DESIGNING DIFFUSION MODELS FOR TRACTABLE LIKELIHOOD ESTIMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Estimating likelihoods with diffusion models is typically done by expensive approximations of the divergence of a neural network, as exact computation is intractable in general. Learning the time score $\partial_t \log p$ is a recent idea that may enable more viable and exact computation, but is still a challenging task as previous work has been limited in scale, has suffered with sampling degradation and has mostly focused on the energy setting, which by itself already suffers from training difficulties. In this work, we tackle these issues by proposing a unifying framework to design diffusion models for effective sampling and likelihood estimation. First, we propose a regularization strategy to incorporate the time score without negatively affecting sampling. Then, we introduce a set of design coefficients and learnable terms that generalizes and connects important cases, such as the energy and mean flow formulations. Finally, specific parametrization and learning strategies are proposed in order to avoid singularities and improve training. The resulting algorithms are tested and studied on synthetic examples and on real images for ImageNet on both pixel and latent space.

1 INTRODUCTION

Diffusion based models are a good choice for generative tasks, but they remain computationally inefficient and thus a poor fit for likelihood estimation. The core issue is estimating the divergence (the trace of the Jacobian) of a neural network, which is intractable in general and is stochastically approximated at the cost of multiple extra network evaluations (Lipman et al., 2023). In contrast, normalizing flows also learn a flow $\boldsymbol{v}$ transforming pure noise x_0 to clean data x_1, but can do both tasks efficiently (Chen et al., 2018). To close this gap, we examine how the chain rule appears in the continuity equation:

$$\frac{d \log p}{dt}(\boldsymbol{x}, t) = \underbrace{\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}, t)^T \boldsymbol{v}(\boldsymbol{x}, t) + \partial_t \log p(\boldsymbol{x}, t)}_{\text{Easy to compute, but singular and degrades sampling}} = \underbrace{-\nabla \cdot \boldsymbol{v}(\boldsymbol{x}, t)}_{\text{Hard to compute}} \tag{1}$$

The expensive divergence term $\nabla \cdot \boldsymbol{v}$ can then be avoided by combining $\boldsymbol{v}$ with the spatial score $\boldsymbol{s} = \nabla \log p$ and the so called time score $\partial_t \log p$. Efficient learning objectives for the time score have been recently proposed (Yu et al., 2025), but how to best estimate and utilize this quantity remains poorly understood. It has been reported to negatively affect the model's generative capabilities (OuYang et al., 2026) and models utilizing it tend to follow an energy-based diffusion formulation (Guth et al., 2026), which by itself struggles with training instabilities and scaling issues (Thornton et al., 2025).

To address these issue and better allow diffusion model to become both feasible options for likelihood estimation and generative tasks, we characterize and study how these models can be designed for this underexplored task. Besides the energy formulation, we consider the traditional score-based one, the mean flow (Geng et al., 2025) one and their combinations in an unifying framework in order to cover a wider range of scenarios of interest. We then notice that all of the common parameter choices used in the literature lead to singular computations, which is expected to be a contributing factor to the reported instabilities, so we suggest novel parameter choices that lead singularity-free computations.

The proposed ideas are then systematically evaluated on synthetic experiments, for which ground truth likelihoods are available, and also on ImageNet as it is a gold standard for generative tasks and

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



Figure 1: Highest and lowest estimated likelihood images from both model-generated outputs and ImageNet validation sets for a selected set of classes. The model achieves an unguided FID of 16.11 on ImageNet256 and reliably identifies high-quality generations (according to FID and IS metrics) and typical validation set samples based on likelihood estimates at intermediary diffusion times.

is one of which, to the best of our knowledge, has so far remained beyond the capacity of the class of models we consider. Besides showcasing strong capabilities on both sampling and likelihood estimation, we also provide insights that can be beneficial to generative tasks, such as how the tractable computation of likelihoods can potentially be beneficial to generative tasks as illustrated in Figure 1.

2 Background

2.1 Related work

Eneergy (potential) diffusion models An energy model is a term used to describe models that learn a scalar field U as the unnormalized log likelihood or energy $U(x) \propto \log p(x)$. They originally utilized contrastive losses and Monte Carlo sampling (LeCun et al., 2006), but they can also use the time dependent process and the losses used in diffusion (Thornton et al., 2025). Time score matching (Guth et al., 2026) or other losses (OuYang et al., 2026) can also be used as a regularization strategy or to normalize the likelihoods. This is contrast with the broad notion of diffusion, wherein one learns a vector field F that approximates $\nabla \log p$ (or a linear transformation of it such as the flow v) with a neural network that may not be conservative, that is, it is not the gradient of any field (Chao et al., 2023). For clarity and since we learn $\phi \not\propto \log p$, we use the term potential diffusion for these scalar models and vector diffusion otherwise. Though we seek normalized likelihoods, many of the model design consideration we explore are also valid for the unnormalized case

Learning a potential diffusion model over a vector one appears to be beneficial in some settings, like studying local properties of the data manifold, though their conservative property doesn't seem to be needed for sampling or density estimation (Horvat & Pfister, 2025). Another setting of interest is in molecular dynamics and other physics-based simulations, where an actual energy functional can be theoretically derived (Plainer et al., 2025). Naturally, these models can in principle operate both as energy models and diffusion models.

Unfortunately, they seem to be in general harder train and suffer from singularities and instabilities. In Thornton et al. (2025), training instabilities are reported so a distillation strategy from a pre-trained teacher is used. A singularity caused by parametrization choices lead to instabilities in Aggarwal et al. (2025), so a separate model is trained with contrastive losses. Despite these issues and the need of expensive higher order derivatives, both potential and vector models appear able to achieve comparable sampling quality (Salimans & Ho, 2021).

Likelihood estimation Estimating likelihood (normalized or not) seems to be a hard and perhaps underexplored problem in high dimensions. The Dual Score Matching (DSM) paper presents an initial study on how this quantity behaves on high dimensional spaces like ImageNet (Guth et al., 2026). Outside of diffusion models, normalizing flows are capable of efficient likelihood estimation and can be scaled to produce high quality samples (Zhai et al., 2024).

Time score Learning the time score $\partial_t \log p$ was proposed in Choi et al. (2022) as the limit of a classification objective, while more efficient conditional score matching objectives, similar to those often used in diffusion, were proposed in Yu et al. (2025). Though it is a promising tool, these works were limited in scope to MNIST and other small problems, while follow-up work combining spatial

and temporal scores noted a drop in generative capacity (OuYang et al., 2026) or did not measure this key part of the model (Guth et al., 2026). It may not be a sufficient learning target in certain settings that generalize diffusion, so other strategies may be necessary in them (OuYang et al., 2026).

Few step (integral) models Since sampling by integrating the flow ODE $\dot{x} = v$ requires multiple model evaluations, several strategies have been proposed to reduce the number of steps, such as retraining models with less curved trajectories (Liu et al., 2023). The class of approaches we consider learn the integral of the flow, that is, they can directly predict $x_r = x_t + \int_t^r \dot{x}(\tau)d\tau$ without and ODE solver and can thus "jump" from the state at one time to another. A key difference between these integral models is their training procedure: consistency models force predictions to map to the same clean sample regardless of starting time (Song et al., 2023), solution flows enforces a consistency loss based on intermediary time between noise and data pairs (Luo et al., 2026), while BOOT predicts intermediary points always starting from pure noise (Gu et al., 2023). These models are generally harder to train, use heavily modified training algorithm and often requiring pre-trained teachers, so we adapt many ideas from mean flow formulation (Geng et al., 2025) in order to minimally modify the usual instantaneous formulation.

Diffusion parametrization Besides the neural network architecture, which quantity to learn and other parameterization question are still an active area of research. The seminal EDM paper (Karras et al., 2022) described many design considerations and proposed a specific choice of parameters meant to provide preconditioning. Our work also explores the design space for $\nabla \log p$, but we use different representations, arrive at different parameters and explore parametrizations for $\log p$ itself and other of its derivatives.

While earlier works used the score $F = \nabla \log p$ as the immediate output of a neural network F (Song et al., 2021), later methods predict other quantities like the noise (Rombach et al., 2022) or flow (Lipman et al., 2023). Recent results suggest that some of these options may fail catastrophically in some settings, a phenomenon that seemingly doesn't happen when predicting the "clean data" (Li & He, 2025). For that reason, some mean flow formulations have also been extended for this particular setting (Lu et al., 2026).

2.2 Theory

We define a diffusion model using the linear schedule $\boldsymbol{x} = t\boldsymbol{x}_1 + (1-t)\epsilon$, where $\boldsymbol{x}_1 \sim p_{\text{data}}$ is a d-dimensional data sample and ϵ is an iid noise vector from a standard Gaussian. This choice allows both pure noise and and pure data situations to be represented in finite time and makes the ODE easier to integrate (Gao et al., 2024). Thus, the following conditional probability holds by definition

$$\log p(\boldsymbol{x}, t|\boldsymbol{x}_1) = -\frac{d}{2}\log(2\pi) - d\log(1-t) - \frac{\|\boldsymbol{x} - t\boldsymbol{x}_1\|^2}{2(1-t)^2} \tag{2}$$

This conditional expression can be turned into a marginal one through the following lemma, which follows from the log-trick: $\mathbb{E}_{p(y|z)}[\nabla_z \log p(z|y)] = \nabla_z \log p(z)$. Thus, applying this lemma with $z = x$ and $y = x_1$

$$\mathbb{E}_{p(x_1)}\mathbb{E}_{p(x|x_1)}\nabla_x \log p(\boldsymbol{x}, t|\boldsymbol{x}_1) = \mathbb{E}_{p(x)}\mathbb{E}_{p(x_1|x)}\nabla_x \log p(\boldsymbol{x}, t|\boldsymbol{x}_1) = \mathbb{E}_{p(x)}\nabla_x \log p(\boldsymbol{x}, t) \tag{3}$$

$$\therefore \mathbb{E}_{p(x)}\nabla_x \log p(\boldsymbol{x}, t) = \mathbb{E}_{p(x_1)}\mathbb{E}_{p(x|x_1)}\left[-\frac{\epsilon}{1-t}\right]. \tag{4}$$

Here, we use $\frac{x - tx_1}{(1-t)^2} = \frac{\epsilon}{1-t}$ based on the proposed schedule, and we swap the order of expectations, given that $p(x, x_1) = p(x|x_1)p(x_1) = p(x_1|x)p(x)$. Since $\mathbb{E}_{p(x)}\nabla_x \log p(\boldsymbol{x}, t) = \mathbb{E}_{p(x,x_1)}\nabla_x \log p(\boldsymbol{x}, t)$, both sides of Equation (4) can be combined into a single expectation equalling zero. A learnable score approximator $S \approx \nabla \log p$ can then be sought by casting this expectation into a minimization according to a spatial score matching loss (Vincent, 2011)

$$\mathcal{L}^s_{SSM} = \mathbb{E}_{p(x_1)}\mathbb{E}_{p(x|x_1)}\left[\left\|S + \frac{\epsilon}{1-t}\right\|^2\right] = \mathbb{E}_{p(x_1)}\mathbb{E}_{p(x|x_1)}\left[\|S - s_{\text{tgt}}\|^2\right]. \tag{5}$$

This a practical objective, as the expectations can be approximated by sampling data points x_1 and noise ϵ, while the so called "target" s_{tgt} is easily computed. Unfortunately, s_{tgt} is singular, and so is

the computation of the flow v from the score, which can be shown to satisfy (Song et al., 2021)

$$\boldsymbol{v}(\boldsymbol{x}, t) = \frac{\boldsymbol{x} + (1-t)\nabla \log p(x,t)}{t}. \tag{6}$$

There are many ways to reformulate this learning problem. A popular choice is using a flow-based reparametrization. The objective in Equation (5) can transformed as

$$\frac{(1-t)^2}{t^2}\left\|S - s_{\text{tgt}}\right\|^2 = \left\|\frac{x+(1-t)S}{t} - \frac{x+(1-t)s_{\text{tgt}}}{t}\right\|^2 = \left\|\frac{x+(1-t)S}{t} - (\boldsymbol{x}_1 - \epsilon)\right\|^2, \tag{7}$$

where s_{tgt} is replaced with the flow target $v_{\text{tgt}} = x_1 - \epsilon$, and $V = \frac{x+(1-t)S}{t}$ can be seen as a reweighted prediction that eliminates one singularity.

A natural next step is then to directly model V as the output of a neural network instead of first predicting S and transforming it, thus eliminating another singularity. More general reparametrizations can be used for the model (Karras et al., 2022) and combined with different targets (Li & He, 2025). Though mathematically equivalent to simply reweighing the loss, these reformulations are numerically superior and do not require truncating the time t, which is known to cause artifacts (Lin et al., 2024). Moreover, both strategies can be pursued simultaneously.

Deriving a denoising-like objective for the time score follows an analogous procedure with $z = t$ on the above lemma. Using $u = \partial_t \log p$ for notational convenience, the time score matching loss is then

$$\mathcal{L}^u_{TSM} = \mathbb{E}_{p(x_1)}\mathbb{E}_{p(x|x_1)}\left[\left\|\partial_t \log p(\boldsymbol{x}, t) - \frac{d + (x_1-\epsilon)^T\epsilon}{1-t}\right\|^2\right]. \tag{8}$$

3 Effective time score learning

Similarly to how the singularities of s_{tgt} can be removed by adopting v_{tgt}, we propose the following transformation of the time score target from Equation (8):

$$(1-t)^2\left\|\partial_t \log p(\boldsymbol{x}, t) - \frac{d + (x_1-\epsilon)^T\epsilon}{1-t}\right\|^2 = \left\|(1-t)\partial_t \log p(\boldsymbol{x}, t) - (d + v_{\text{tgt}}^T\epsilon)\right\|^2. \tag{9}$$

We name the quantity $w = (1-t)\log p(\boldsymbol{x}, t)$ with the target $w_{\text{tgt}} = d + v_{\text{tgt}}^T\epsilon$ as the cross-flow, since it captures the cross covariance between the flow target and the noise. One expects that learning a covariance term like in w_{tgt} is generally harder than learning a mean like in v_{tgt}, so our strategy further centers on minimizing the need for such term in the first place.

For a potential model trying to learn $\log p(x, t)$, if $\nabla \log p$ was perfectly modeled, then the likelihoods would be known up to a function of time, so the time score would only need to be used to resolve this 1D ambiguity. Therefore, ensuring the diffusion model is an effective generator is an important step that motivates the unifying approach pursued in this work.

The final component of our strategy revolves around treating the time score losses as regularizers by potentially scaling them down by a small factor λ_w. Combining this with the dimensional rescaling proposed in Guth et al. (2026), we therefore propose training using the loss

$$\mathcal{L} = \mathcal{L}^v_{SSM} + \frac{\lambda_w}{d^2}\mathcal{L}^w_{TSM}. \tag{10}$$

4 Design space

Diffusion models are typically focused on computing the flow $\boldsymbol{v}$, which is closely tied to the spatial score $\nabla \log p$. Certain inductive biases can be encoded by combining the contributions of x_t and a learnable F according to some pre-determined coefficients. We are interested in learning approximators for three types of derivatives of the likelihood, so we consider three fields: F for the spatial score $\nabla \log p$, T for the time score $\partial_t \log p$ and L for the likelihood flow $\frac{d \log p}{dt}$. The cases where a mean flow $\overline{v}$ or a scalar potential ϕ are used are also of importance and closely related.

This section therefore proposes a unifying framework that includes, generalizes and elucidates these aspects. In other words, it provides a structure and space of coefficients, while Section 5 discuss specific choices for these parameters. The following likelihood structure serves as a central point from which each derivative can be analyzed

$$\log p(\boldsymbol{x}, t) = c_d(t)d + c_x(t)\frac{\|x\|^2}{2} + c_F(t)\phi(\boldsymbol{x}, t). \tag{11}$$

This form is chosen to match previous frameworks for $\nabla \log p$ and can be seen as a residual connection over a (potentially unnormalized) Gaussian.

Notation Given some coefficient $a(t)$, we use $\breve{a} \triangleq \int_t^r a(\tau)\tau$ to implicitly denote this specific integral. For fields like F, there will be integral and mean quantities denoted with $\breve{F}$ and $\overline{F}$ and their values will be written explicitly.

4.1 Creating a mean field

A key idea used to derive the mean field versions of fields for F and L, follows from recognizing the following structure for an integral over a quantity $A(x_t, t)$ weighted by the scalar $w(t)$:

$$\breve{w}(t, r)\overline{A}(x_t, t, r) = \int_t^r w(t)A(x_\tau, \tau)d\tau \tag{12}$$

$$\therefore \frac{d\breve{w}(t, r)}{dt}\overline{A}(x_t, t, r) + \breve{w}(t, r)\frac{d\overline{A}(x_t, t, r)}{dt} = -w(t)A(x_t, t). \tag{13}$$

By selecting $\breve{w}(t, r) = \int_t^r w(\tau)d\tau$ such that $\breve{w}(t, t) = 0$ and $\frac{d\breve{w}(t,r)}{dt} = -w(t) \neq 0$, then $\overline{A}(x_t, t, t) = A(x_t, t)$ and $\overline{A}$ is the weighted mean $\frac{\int_t^r w(\tau)A(x_\tau,\tau)d\tau}{\int_t^r w(\eta)d\eta}$. When $w(t) = 0$ the values of A do not influence the integral in Equation (12) provided they are bounded.

4.2 Spatial field (F)

The spatial gradient of Equation (11) recovers a structure similar to that introduced in EDM when replacing $\nabla\phi$ with F. Previous work define coefficients over one quantity of interest, such as ϵ or v, but we opt to do so for both our quantities of interest:

$$\nabla \log p(\boldsymbol{x}, t) = c_x(t)\boldsymbol{x} + c_F(t)F(\boldsymbol{x}, t) \tag{14}$$

$$\boldsymbol{v}(\boldsymbol{x}, t) = a_x(t)\boldsymbol{x} + a_F(t)F(\boldsymbol{x}, t) \tag{15}$$

Using the seemingly redundant design parameters a_x, a_F, c_x, c_F has several benefits: it simplifies expressions, makes the parametrization more general (as it is independent of the noise schedule) and helps avoid the occurrence of singularities in practice. To illustrate: in our schedule, a_F can be computed from c_F as $a_F(t) = \frac{t}{1-t}c_F(t)$, which would be numerically singular at $t = 1$ even if the singularity was analytically removeable.

Generalizing the displacement $\breve{v}(x_t, t, r) = \int_t^r v(x_t, t) = \int_t^r \dot{x}_t$, we can derive an analogous $\breve{F}$ by solving the linear ODE defined in Equation (15). Similarly, the the mean flow $\overline{v}$ is generalized by $\overline{F}$ according to

$$\tilde{a}_x(t, r) \triangleq e^{\int_t^r a_x(\tau)d\tau}, \tag{16}$$

$$\breve{a}_F(t, r)\overline{F}(x_t, t, r) \triangleq \breve{F}(x_t, t, r) \triangleq \int_t^r \tilde{a}_x(\tau, r)a_F(\tau)F(x_\tau, \tau)d\tau, \tag{17}$$

where $\overline{F}$ is a weighted mean of $\breve{F}$ along the interval $[t, r]$ according to the pointwise weight a_F Note that the choice $a_x \equiv 0$ and $a_F \equiv 1$ recovers the flow prediction with $F = v$.

In practice, the following relations are of particular importance to sampling and training:

$$x_r = \tilde{a}_x(t, r)x_t + \breve{F}(x_t, t, r), \tag{18}$$

$$v_{\overline{F}}(\boldsymbol{x}_t, t, r) = a_x(t)\boldsymbol{x}_t - \tilde{a}_x(r, t)\frac{d\breve{F}(\boldsymbol{x}_t, t, r)}{dt}, \tag{19}$$

$$\frac{d\overline{F}}{dt}(\boldsymbol{x}_t, t, r) = \mathrm{JVP}_{sg}(\overline{v}; [v, 1, 0]). \tag{20}$$

Where the derivative of $\overline{F}$ can computed with a Jacobian vector product and the stopgrad trick to prevent the need for higher oder gradients (Geng et al., 2025). The instantaneous $v(x_t, t)$ and $s(x_t, t)$ can then both be computed without derivatives. Note that if a_x and a_f are singularity free, then all the derived computations are also singularity free.

4.3 Temporal score field (T)

By taking the partial time derivative of Equation (11) and replacing the term $\dot{c}\phi + c\dot{\phi}$ with a scaled field $c_T T$, one obtains

$$\partial_t \log p(\boldsymbol{x}, t) = \dot{c}_d(t)d + \dot{c}_x(t)\frac{\|x\|^2}{2} + c_T(t)T(x_t, t). \tag{21}$$

A mean field $\overline{T}(\boldsymbol{x}_t, t, r)$ could be defined, but doing so does not seem particular helpful as it would need to be combined with $\int_t^r s^T v$, according to Equation (1). Though a second mean field for that integral could also be constructed, we instead choose to use a single field L for $\frac{d\log p}{dt}$ instead.

4.4 Likelihood field (L)

By applying $\frac{d\log p}{dt} = -\nabla \cdot \boldsymbol{v}$ from Equation (1) into Equation (15), a Laplacian term $-c_F\nabla^2\phi = -c_F\nabla F$ appears. It would be tempting to just replace $\nabla^2\phi$ with a field L, but that limits the ability to enforce certain known analytical results. In particular, $\frac{d\log p}{dt}(x, 0) = d$ holds independently of distribution, but $v(x, 0) = x - \mathbb{E}[x_1]$ is distribution dependent. Thus, for convenience of notation of notation we use

$$\frac{d\log p}{dt} = -a_x(t)d + a_L(t)L(x_t, t) \tag{22}$$

with $a_L(t)$ a new parameter. Deriving a mean field version of L is a straightforward application of the ideas presented in Section 4.1:

$$\breve{a}_L(t, r)\overline{L} \triangleq \breve{L} \triangleq \int_t^r a_L(\tau)L(x_\tau, \tau)d\tau. \tag{23}$$

For training purposes, it will be desirable to estimate $\frac{d\log p}{dt}(\boldsymbol{x}_t, t)$ from $\overline{L}$ much like $v(\boldsymbol{x}_t, t)$ is estimated from $\overline{F}$. In order to estimate likelihoods, it is desirable to be able to compute $\log p(\boldsymbol{x}_r, r)$ from $\boldsymbol{x}_t$, which can then be used in two ways. The first is during sampling, which starts with a known $\log p(x_0, 0)$ and then subsequent $\log p(x_r, r)$ can be computed along the denoising trajectory with $t > r$. The second follows the same process, but in opposite direction, where some x_1 is given instead of generated. These capabilities can be expressed as

$$\log p(x_r, r) = \log p(x_t, t) - \breve{a}_x(t, r)d + \breve{L}(\boldsymbol{x}, t, r), \tag{24}$$

$$\frac{d\log p(\boldsymbol{x}_t, t, r)}{dt} = -a_x(t)d - \frac{d\breve{L}}{dt}(\boldsymbol{x}_t, t, r). \tag{25}$$

4.5 Potential

The potential field ϕ is a specialized choice for F that forces it to be conservative by construction, as discussed in section 2.1, and allows for straightforward likelihood evaluation using Equation 11. Previous work has suggested levering existing neural network architecture used for F in order to build the potential field and to avoid confusion we use the letter G for them. Some structures a include linear $\boldsymbol{x}^T G$ (Guth et al., 2026) or quadratic $\|x - G\|^2$ terms (Salimans & Ho, 2021).

We use the straightforward choice of simply summing the outputs of G, as in $\phi = \mathbf{1}^T G(x)$, as it appears effective and previous work has not found compelling evidence to use one form over another. A more detailed discussion is provided in the appendix, but we highlight that the same stopgrad trick used for the mean field can also in principle be applied to compute the required derivatives of ϕ and could greatly improve their computational efficiency. Unfortunately, doing so makes training even more unstable.

Both instantaneous and integral versions of the potential can be defined analogously to Equation 17

$$\breve{\phi}(x_t, t, r) = \breve{a}_F(t, r)\overline{\phi}(x_t, t, r) = \int_t^r \breve{a}_x(\tau, r) a_F(\tau) \phi(x_\tau, \tau) d\tau \tag{26}$$

Note that $\phi = \nabla F$, $\breve{\phi} = \nabla \breve{F}$ and $\overline{\phi} = \nabla \overline{F}$, making the formulation highly modular.

It will be useful to also compute T as discussed in section 3, which require both ϕ and $\dot{\phi}$ in general. All terms needed for spatial and temporal losses can then be computed from a single vector-Jacobian operator (VJP), since ϕ is a scalar field.

$$[\phi(\boldsymbol{x}_t, t), \nabla\phi(\boldsymbol{x}_t, t), \partial_t\phi(\boldsymbol{x}_t, t)] = \text{VJP}(\phi(\boldsymbol{x}_t, t); 1) \tag{27}$$

5 Model and Parameter Selection

Given the design space presented in the previous section, it's then necessary which fields which specific coefficients should be used, which will define how training should proceed. In this section we propose a single choice that is applicable for any combination and uses a uniform criteria of enforcing boundary condition and removing singularities in order to help stabilize learning. Note that is possible to, for example, use one value for a_x in Equation 15 and a different one Equation 22 since they involve different the fields F and L.

The computation of the $\frac{d \log p}{dt}$ using the chain rule as in Equation 1 requires both s and v, with the coefficients for s also appearing in the computation of $\log p$ for potential models Equation 11. Unfortunately, all of the commonly used spatial parametrizations (Li & He, 2025; Karras et al., 2022) have a singularity in either $t = 1$ or $t = 0$, as shown in Table 1.

To understand the source of this issue, consider that $t\boldsymbol{v} = \boldsymbol{x} + (1-t)\boldsymbol{s}$ according to Equation 6. At $t = 0$, the score must then be $\boldsymbol{s}(\boldsymbol{x}, 0) = -\boldsymbol{x}$ otherwise $\boldsymbol{v} = \lim_{t\to 0} \frac{\boldsymbol{x}-\boldsymbol{s}}{t}$ will diverge. Analogously, at $t = 1$, $v(\boldsymbol{x}, 1) = x$ is necessary for the stable computation of $s(\boldsymbol{x}, 1)$. Thus, we can avoid singularities by guaranteeing by construction the satisfaction of the boundary condition. This should also provide good conditioning, so we adopt the same criteria for the other less explored fields.

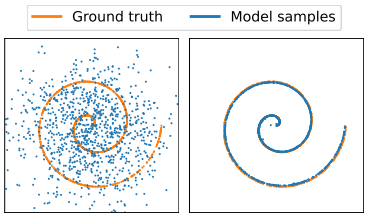
Table 1: Spatial parametrizations and whether they guarantee the satisfaction of the boundary conditions of the diffusion process. The denominators are highlighted, with $\sigma = t^2 + (1-t)^2$ being the only one that is never 0 and therefore does not produce singularities.

	$v = a_x x + a_F F$		$s = c_x x + c_F F$		Boundary condition	
	a_x	a_F	c_x	c_F	$v(x,1) = x$	$s(x,0) = -x$
x-pred	$-1/(1-t)$	$1/(1-t)$	$-1/(1-t)^2$	$t/(1-t)^2$	No	Yes
ϵ-pred	$1/t$	$-1/t$	0	$-1/(1-t)$	No	No
v-pred	0	1	$-1/(1-t)$	$t/(1-t)$	No	Yes
s-pred	$1/t$	$(1-t)/t$	0	1	Yes	No
EDM	$(2t-1)/\sigma^2$	$1/\sigma$	$-1/\sigma^2$	$t/((1-t)\sigma)$	No	Yes
ProGaMeR	$(2t-1)/\sigma^2$	$(1-t)/\sigma^2$	$-1/\sigma^2$	t/σ^2	Yes	Yes

Since the structure of Equation 11 is already that of a Gaussian, we fully embrace that form. If the potential $\phi(\boldsymbol{x}, t) = \mu$ was a constant, then we'd have $p(\boldsymbol{x}, t) = \mathcal{N}(\boldsymbol{x}; \mu, \sigma^2(t)\boldsymbol{I})$ with $\sigma(t) = t^2 + (1-t)^2$. In other words, in an approximately Gaussian sense, ϕ predicts the mean of the score projected at the clean time $t = 0$. We name this choice as the Projected Gaussian Mean Replacement and display the values on Table 1. Note that EDM shares the coefficients for x, which is not surprising as those match the ones from the optimal linear filter.

The claim that a good parametrization should be singularity-free, seem to contradict recent works that suggest that certain predictions fail catastrophically in high dimensions (Li & He, 2025). In particular, it is claimed that other common choice fails, while x-pred succeeds since it predicts information in the data manifold. To be precise, the minimizer for x-pred is the posterior mean $\mathbb{E}[\boldsymbol{x}_1|\boldsymbol{x}_t]$, which is not necessarily in the manifold, but that may nonetheless be beneficial. Since all parametrizations differ only scalar coefficients and a $\boldsymbol{x}_t$ term, this seem to suggest an underlying problem in the neural network construction itself.

To confirm that hypothesis, we reproduce their toy setting and notice that, after normalizing the inputs and applying the modulation strategy $F(\boldsymbol{x}, t) = A(t)\boldsymbol{x}_t + B(t)\,\mathrm{net}(\boldsymbol{x}, t)$, the problem disappears and the model correctly predicts the data as displayed in Figure 2. This highlights that certain parametrization can cover for network deficiencies and so may be beneficial for those reason despite the singularities we tackle. We posit that these are not conflicting objectives and that an ideal approach would be both singularity free and have a robust network. Beyond toy experiments, it appears that diffusion transformers struggle to learn some simple transformations (Chen et al., 2026), so similar problems may exist in larger scales.



(a) Original (b) Corrected

Figure 2: The catastrophic failure of v-pred and others reported in previous work can be corrected with normalization and a learnable modulation.

The coefficients for L and T follow similar by considering a scaling factor based on σ and to satisfy other constraints. In particular, we add the factor $t(1-t)$ to c_L since $\frac{d\log p(\boldsymbol{x},t)}{dt} = -\nabla \cdot \boldsymbol{v}(\boldsymbol{x}, t)$, and so $-\nabla \cdot \boldsymbol{v}(\boldsymbol{x}, 1) = -\nabla \cdot x = -d$ and $-\nabla \cdot \boldsymbol{v}(\boldsymbol{x}, 0) = -\nabla \cdot (\mathbb{E}[x_1] - x) = d$. These resulting expressions are summarized below

$$\log p(\boldsymbol{x}, t) = \left[-\frac{\log(2\pi) + \log(\sigma^2)}{2}\right] d - \frac{1}{\sigma^2}\frac{\|\boldsymbol{x}\|^2}{2} + \frac{t}{\sigma^2}\phi(\boldsymbol{x}, t) \tag{28}$$

$$\partial_t \log p(x_t, t) = -\frac{(2t-1)}{\sigma^2} d + \frac{(2t-1)}{\sigma^4}\|x\|^2 + \frac{1}{\sigma^4} T(\boldsymbol{x}, t) \tag{29}$$

$$\frac{d\log p(\boldsymbol{x}, t)}{dt} = -\frac{(2t-1)}{\sigma^2} d + \frac{t(1-t)}{\sigma^2} L(\boldsymbol{x}, t) \tag{30}$$

Training follows a modularized strategy inspired by Geng et al. (2026). We'll describe how all fields can be trained, but when they don't exist the associated computation can safely be dropped. After sampling t and r independently, the spatial field (which always exists) is used to compute the instantaneous v_F and s. This predicted v_F is used as a training signal and to compute the integral estimate $v_{\overline{F}}$ from Equation 19. For the temporal fields, we perform instantaneous computations of w from T and L as w_T and w_L and use them as training signals. Similarly, the integral estimate $w_{\overline{L}}$ is computed using v_F. The chain rule in Equation 1 and the total derivatives are computed after blocking gradients from v and s.

6 Experiments

We perform multiple synthetic and real image experiments on ImageNet to validate the proposed approach and to study the behavior of likelihoods. The LDiT architecture from Yao et al. (2025) is used as backbone and their pretrained VAE is used for latent diffusion. For mean fields, we add time encoders for $t - r$ as in Geng et al. (2026) and add extra heads whens predicting a second field. Additional experiments and details are provided in the appendix.

Synthetic likelihood validation We build a 10 dimensional mixture of two Gaussian by repeating the mean and covariances from $0.25\mathcal{N}(-1, \sqrt{0.1}) + 0.75\mathcal{N}(-1, \sqrt{0.1})$ along new dimensions and renormalizing so the mixture has 0 mean and unit variance. Model predictions are compared for all likelihood computations that different field choices can provide and the results are in Figure 3, showing strong agreement for the instantaneous and potential fields, while the mean fields are less accurate but use only a single evaluation.

Selection of time conditioning and w-loss weight scales. We investigate the effect of varying the time conditioning scale and strength of the w loss in both latent and pixel space settings in Tables 2a and 2b. The time scaling is a multiplier to t and r for the purpose adjusting encoder sensitivity, which has been pointed out to be an important factor consistency models (Lu & Song, 2025). We find that larger w weights lead to faster convergence towards a common average likelihood, illustrating significant differences in convergence behavior between time conditioning scales. Additional details can be found in Figure 5 in the appendix. A crucial takeaway is that, for diffusion and other

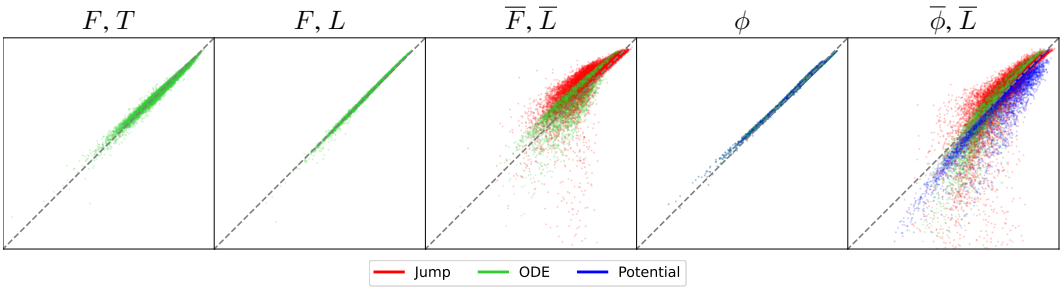


Figure 3: Comparison between different model predictions (x axis) for likelihood compared against the true values (y axis) for a 10 dimensional mixture of Gaussian. Values represent nats per dim in the range $[-2.5, 0.25]$ and the mean models take 1-step jumps for their predictions.

Table 2: Varying the w weight and time conditioning scale. Coefficients of linear fit are shown for likelihoods (estimated for validation set images) against those from the reference model. All models trained for 50 epochs.

(a) Latent space

time scaling	w-weight	Bias (bpd)	R^2	Slope	FID	IS
1000	10^{-2}	0.0004	0.9830	1.0024	39.34	43.11
1000	10^{-3}	— *reference* —			37.95	45.09
1000	10^{-4}	0.0049	0.9832	1.0553	39.67	43.94
1	10^{-2}	0.0146	0.9789	0.9215	41.53	42.29
1	10^{-3}	0.1611	0.9672	0.9709	39.40	44.11
1	10^{-4}	0.4068	0.9655	0.9285	41.96	42.10

(b) Pixel space

time scaling	w-weight	Bias (bpd)	R^2	Slope	FID	IS
1000	1	0.2140	0.9215	1.1219	39.99	12.48
1000	10^{-2}	0.2037	0.9674	1.1089	42.02	12.37
1000	10^{-3}	— *reference* —			42.49	12.24
1000	10^{-4}	0.1301	0.9779	0.9331	41.92	12.40
1000	0	10.6086	0.9710	0.9367	42.10	12.45
1	10^{-2}	0.1394	0.9408	1.0898	41.25	12.57
1	10^{-3}	0.1855	0.9627	1.0491	42.39	12.44
1	10^{-4}	0.6126	0.9615	1.0081	41.30	12.59

models whose likelihood are not guaranteed to be normalized, likelihood values may be either artificially inflated or deflated during training. Thus, extra steps must be taken to properly validate their predictions if the NLL is to be used as credible metric for their performance. Unlike OuYang et al. (2026), we do not find appreciable sampling degradation when using time score losses.

Evaluating likelihood quality through filtering. We find that the likelihood estimates produced by our models, especially when considering times $t < 1$, can be successfully used to identify high quality samples. Figure 1 illustrates this qualitatively on both validation set and generated images using our latent space multi-step model. Using this model we further sort a set of 50k generations by their predicted likelihoods, and compute FID and IS of the 10k likeliest (and 10k least likely) while maintaining uniformly balanced classes. As shown in Figure 4 this likelihood filtering procedure can be used to select significantly higher (or lower) quality generated samples when t is chosen appropriately, achieving an improvement to FID 14.3 (from 18.8) and IS 100.2 (from 80.0) at $t = 0.4$ as compared to a randomly selected subset. Likelihoods for $t < 1$ are computed by corrupting inputs and averaging the resulting likelihood over 8 noise draws.

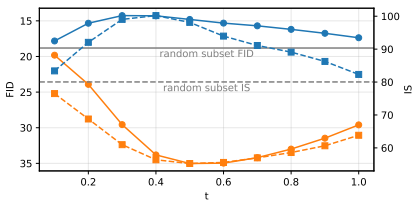


Figure 4: Blue shows statistics for the top 20% most likely generations out of a pool of 50k, while orange shows the 20% least likely. FID-10k and IS are shown using solid and dashed lines, respectively.

7 Conclusion

This paper studied how diffusion models should be designed in order to allow for practical likelihood estimation while maintaining generative capacity. After providing a unifying framework and parameter choices that address issues from previous methods, we were able to demonstrate strong capacity in both aspects. Moreover the proposed analyses provide insight on further limitations on how these models are built for generative tasks and indicate that greater care may be warranted when using them for likelihood estimation.

AI USE STATEMENT

This work was developed, for the most part, by human effort, while AI tools served ancillary purposes. They were used mainly in tasks such editing, finding bugs, double-checking proofs and providing some code snippets, which were human verified and tested. The core of the contributions were developed and implemented by humans, while AI had a larger role in assisting with boilerplate code to manage training runs and with creating visualizations.

REPRODUCIBILITY STATEMENT

We provide more detailed information in the appendix and plan to include the used code for reproducibility reasons upon acceptance and legal approval. Upon trying to understand results from previous work, we found that many practical implementation details were not properly addressed in text, perhaps justifiably so given the extensive amount of code required to implement the proposed experiments. As such, we consider it a crucial component and have taken steps to ensure reproducibility and readability.

REFERENCES

Rishal Aggarwal, Jacky Chen, Nicholas Boffi, and David Koes. Boltznce: Learning likelihoods for boltzmann generation with stochastic interpolants and noise contrastive estimation. 38, Main Conference:167195–167227, 2025. doi: 10.52202/085713-5574. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/f455010f4ff54a5d857fde29f14bdd3e-Paper-Conference.pdf.

Chen-Hao Chao, Wei-Fang Sun, Bo-Wun Cheng, and Chun-Yi Lee. On investigating the conservative property of score-based generative models. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 4076–4095, 2023.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf.

Zhennan Chen, Junwei Zhu, Xu Chen, Jiangning Zhang, Xiaobin Hu, Hanzhen Zhao, Chengjie Wang, Jian Yang, and Ying Tai. Dip: Taming diffusion models in pixel space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 36136–36146, 2026.

Kristy Choi, Chenlin Meng, Yang Song, and Stefano Ermon. Density ratio estimation via infinitesimal classification. In *International Conference on Artificial Intelligence and Statistics*, pp. 2552–2573. PMLR, 2022.

Ruiqi Gao, Emiel Hoogeboom, Jonathan Heek, Valentin De Bortoli, Kevin P. Murphy, and Tim Salimans. Diffusion meets flow matching: Two sides of the same coin. 2024. URL https://diffusionflow.github.io/.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=uWj4s7rMnR.

Zhengyang Geng, Yiyang Lu, Zongze Wu, Eli Shechtman, J Zico Kolter, and Kaiming He. Improved mean flows: On the challenges of fastforward generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 30467–30476, 2026.

Jiatao Gu, Shuangfei Zhai, Yizhe Zhang, Lingjie Liu, and Joshua M. Susskind. BOOT: Data-free distillation of denoising diffusion models with bootstrapping. In *ICML 2023 Workshop on Structured Probabilistic Inference & Generative Modeling*, 2023. URL https://openreview.net/forum?id=ZeM7S01Xi8.

Florentin Guth, Zahra Kadkhodaie, and Eero P Simoncelli. Learning normalized image densities via dual score matching. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=wtYcS4kxpF.

Christian Horvat and Jean-Pascal Pfister. On gauge freedom, conservativity and intrinsic dimensionality estimation in diffusion models. In *The Twelfth International Conference on Learning Representations*, 2025.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, Fujie Huang, et al. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. *arXiv preprint arXiv:2511.13720*, 2025.

Shanchuan Lin, Bingchen Liu, Jiashi Li, and Xiao Yang. Common diffusion noise schedules and sample steps are flawed. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 5404–5411, 2024.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. 2023.

Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=XVjTT1nw5z.

Cheng Lu and Yang Song. Simplifying, stabilizing and scaling continuous-time consistency models. In *International Conference on Learning Representations*, volume 2025, pp. 50611–50649, 2025.

Yiyang Lu, Susie Lu, Qiao Sun, Hanhong Zhao, Zhicheng Jiang, Xianbang Wang, Tianhong Li, Zhengyang Geng, and Kaiming He. One-step latent-free image generation with pixel mean flows. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=WUK8JIeetF.

Tianze Luo, Haotian Yuan, and Zhuang Liu. Soflow: Solution flow models for one-step generative modeling. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=cjb03GNqYw.

RuiKang OuYang, Louis Grenioux, and José Miguel Hernández-Lobato. A diffusive classification loss for learning energy-based generative models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2601.21025.

Michael Plainer, Hao Wu, Leon Klein, Stephan Günnemann, and Frank Noe. Consistent sampling and simulation: Molecular dynamics with energy-based diffusion models. 38, Main Conference:24460–24505, 2025. doi: 10.52202/085713-0826. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/231be94eaf8dcbc49a95b256c9b6b8b5-Paper-Conference.pdf.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Tim Salimans and Jonathan Ho. Should ebms model the energy or the score? In *Energy based models workshop-ICLR 2021*, 2021.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. 2021.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 32211–32252. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/song23a.html.

James Thornton, Louis Béthune, Ruixiang ZHANG, Arwen Bradley, Preetum Nakkiran, and Shuangfei Zhai. Composition and control with distilled energy diffusion models and sequential monte carlo. In Yingzhen Li, Stephan Mandt, Shipra Agrawal, and Emtiyaz Khan (eds.), *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics*, volume 258 of *Proceedings of Machine Learning Research*, pp. 3259–3267. PMLR, 03–05 May 2025. URL https://proceedings.mlr.press/v258/thornton25a.html.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15703–15712. IEEE, 2025.

Hanlin Yu, Arto Klami, Aapo Hyvarinen, Anna Korba, and Omar Chehab. Density ratio estimation with conditional probability paths. In *International Conference on Machine Learning*, pp. 73146–73174. PMLR, 2025.

Shuangfei Zhai, Ruixiang Zhang, Preetum Nakkiran, David Berthelot, Jiatao Gu, Huangjie Zheng, Tianrong Chen, Miguel Angel Bautista, Navdeep Jaitly, and Josh Susskind. Normalizing flows are capable generative models. 2024. URL https://arxiv.org/abs/2412.06329.

A ADDITIONAL EXPERIMENTS AND DETAILS

Our code is implemented in PyTorch using Linux. For the large ablation experiments, each training run of 50 epochs required roughly a day of training on a L-40 GPU. The base model from LDIT was used in all settings, with a patch size of 2 for latent space model and 8 pixel space ones in order to use the same number of tokens. The larger experiments used a batch size of 1 in latent space and 4 in pixel space and utilized an RTX 6000 Pro Blackwell due to the larger memory requirements. These took roughly 3 day each.

We use EMA with decay 0.999 and learning rate of $1e-4$ and batch size of 128. ODE integration is done with 50 of the Heun method. No time based weights are used for the objectives as these are equivalently enforced by choice of learning target. We use the same truncation scheme of Li & He (2025) to handle truncation for other quantities.

All models were compiled, which we found is a crucial step to ensure performance. Training is performed with the $bfloat16$, which is a testament to stability of the method given the instabilities reported in previous work.

To apply the modulation strategy that prevents the catastrophic failure of v-pred, we simply use a single layer MLP to produce both scaling factors. Normalization is required because the problem's original design involves a projection from a 2 dimensional manifold into a 256 dimensional one, so we normalize based on the ration of dimensions to ensure roughly unit variance if there latent data had unit variance.

We note that comparing with other diffusion methods is a tricky task since different methods adopt different backbones and metric computation strategies, but provide Table 5 as a reference. Notably, DSM does not properly dequantize their estimates and even use a resolution capable of capturing detais several order of magnitude below pixel level, thus not being appropriate for comparison, besides the issues raised on the main text Additionally, our method is conditional, which is a benefit that comes from free from the formulation.

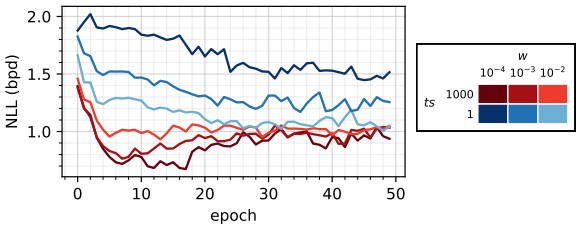


Figure 5: Likelihoods predicted by latent model during training for various time scaling (ts) and w-weight settings.

Table 3: Potential diffusion using different parameterizations. Experiments use time scaling of 1000 and spatial score matching only on ImageNet64 in pixel space. Note x-prediction training was found to be unstable. Best results over 50 epochs shown.

Parameterization	FID	IS
v-prediction	39.50	13.13
ours	42.10	12.45
x-prediction	52.86	10.65
s-prediction	67.61	9.46

Table 4: Potential diffusion vs. vector diffusion. Experiments use time scaling of 1000 and spatial score matching only on VA-VAE latents. Best results over 50 epochs shown.

Spatial Field	FID	IS	NLL (bpd)
Vector	44.47	40.27	1.22
Potential	37.95	45.09	1.04

B Mathematical Derivations

B.1 Spatial mean field

Let's consider a general structure

For $\tilde{a}_x(t,\tau) \triangleq e^{\int_t^\tau a_x(s)ds}$ as per Equation (16), several useful properties follow immediately

$$\tilde{a}_x^{-1}(\tau,t) = \tilde{a}_x(t,\tau) \tag{31}$$

$$\frac{d\tilde{a}_x}{d\tau}(t,\tau) = \tilde{a}_x(t,\tau)a_x(\tau) \tag{32}$$

$$\tilde{a}_x(t,\tau)\tilde{a}_x(\tau,r) = \tilde{a}_x(t,r) \tag{33}$$

Combining them leads to the useful relation

$$\frac{d\tilde{a}_x^{-1}}{d\tau}(t,\tau) = \frac{d\tilde{a}_x}{d\tau}(\tau,t) = -\tilde{a}_x(\tau,t)a_x(\tau) = -\tilde{a}_x^{-1}(t,\tau)a_x(\tau) \tag{34}$$

Thus, given the structure from Equation (15)

$$\dot{\boldsymbol{x}}(t) = a_x(t)\boldsymbol{x}(t) + a_F(t)F(\boldsymbol{x},t) \tag{35}$$

One can notice that the last part of Equation (34) can transform a_x into the derivative of another term. Using that, we can rearrange Equation (35) in order to isolate all terms involving $\boldsymbol{x}$ to the same side as follows after multiplying both side by $\tilde{a}_x^{-1}(t,\tau)$

Table 5: Comparison with other methods on ImageNet64

Method	NLL	FID	Note
Flow matching Lipman et al. (2023)	3.31	14.45	expensive NLL
DSM (Guth et al., 2026)	3.36	–	No sampling provided
Ours	3.25	25.29	

$$\tilde{a}_x^{-1}(t,\tau)\frac{dx_\tau}{d\tau} - \tilde{a}_x^{-1}(t,\tau)a_x(\tau)x_\tau = \tilde{a}_x^{-1}(t,\tau)a_F(\tau)F(x_\tau,\tau) \tag{36}$$

$$\tilde{a}_x^{-1}(t,\tau)\frac{dx_\tau}{d\tau} + \frac{d}{d\tau}\left[\tilde{a}_x^{-1}(t,\tau)\right]x_\tau = \tilde{a}_x^{-1}(t,\tau)a_F(\tau)F(x_\tau,\tau) \tag{37}$$

$$\frac{d}{d\tau}\left[\tilde{a}_x^{-1}(t,\tau)x_\tau\right] = \tilde{a}_x^{-1}(t,\tau)a_F(\tau)F(x_\tau,\tau) \tag{38}$$

$$\left[\tilde{a}_x^{-1}(t,\tau)x_\tau\right]\Big|_t^r = \int_t^r \tilde{a}_x^{-1}(t,\tau)a_F(\tau)F(x_\tau,\tau)d\tau \tag{39}$$

$$\tilde{a}_x^{-1}(t,r)x_r - x_t = \int_t^r \tilde{a}_x^{-1}(t,\tau)a_F(\tau)F(x_\tau,\tau)d\tau \tag{40}$$

$$x_r = \tilde{a}_x(t,r)x_t + \tilde{a}_x(t,r)\int_t^r \tilde{a}_x^{-1}(t,\tau)a_F(\tau)F(x_\tau,\tau)d\tau \tag{41}$$

$$x_r = \tilde{a}_x(t,r)x_t + \underbrace{\int_t^r \tilde{a}_x(\tau,r)a_F(\tau)F(x_\tau,\tau)d\tau}_{\breve{F}} \tag{42}$$

Thus, the field $\breve{F}$ as presented in Equation (17) appears naturally from the solution of Equation (15).

$$\frac{d\breve{F}}{dt}(x_t,t,r) = -\tilde{a}_x(t,r)a_F(t)F(x_t,t) \tag{43}$$

$$F(x_t,t) = -\tilde{a}_x^{-1}(t,r)a_F^{-1}(t)\frac{d\breve{F}}{dt}(x_t,t,r) = -\tilde{a}_x(r,t)a_F^{-1}(t)\frac{d\breve{F}}{dt}(x_t,t,r) \tag{44}$$

$$x_r = \tilde{a}_x(t,r)x_t + \breve{F}(x_t,t,r) \tag{45}$$

It will be desirable to use this F computed from $\breve{F}$ for training. One practical issue is that $\breve{a}_F(t,r)$ can become 0 in some parametrizations, but that is a removable singularity. Using the v-space, we can define an estimate of $v_{\breve{F}}(x_t,t,r)$ based on $\breve{F}$, which ideally should no depend on r

$$v_{\breve{F}}(x_t,t,r) = a_x(t)x_t - \breve{a}_x(r,t)\frac{d\breve{F}}{dt}(x_t,t,r) \tag{46}$$

IT will also be beneficial to compute the score

$$s_{\breve{F}}(x_t,t,r) = c_x(t)x_t - \frac{c_F}{a_F}\breve{a}_x(r,t)\frac{d\breve{F}}{dt}(x_t,t,r) \tag{47}$$

Extending the mean velocity, $\overline{v}(x_t,t,r)(r-t) = \breve{v}(x_t,t,r)$, we use

$$\breve{a}_F(t,r) = \int_t^r a_F(\tau)d\tau \tag{48}$$

$$\breve{F}(x_t,t,r) = \breve{a}_F(t,r)\overline{F}(x_t,t,r) \tag{49}$$

$$\frac{d\breve{F}(x_t,t,r)}{dt} = -a_F(t)\overline{F}(x_t,t,r) + \breve{a}_F(t,r)\frac{d\overline{F}}{dt}(x_t,t,r) \tag{50}$$

$$\therefore v_{\overline{F}}(x_t,t,r) = a_x(t)x_t + \breve{a}_x(r,t)\left[a_F(t)\overline{F}(x_t,t,r) - \breve{a}_F(t,r)\frac{d\overline{F}}{dt}(x_t,t,r)\right] \tag{51}$$

$$v_{\overline{F}}(x_t,t,t) = a_x(t)x_t + a_F(t)\overline{F}(x_t,t,t) \tag{52}$$

For $L(x_t, t)$, we could learn only the difference from a Gaussian. Interestingly, the transition version only depends on t and not x

$$\breve{L}(x_t, t, r) = \int_t^r L(x_\tau, \tau) d\tau \tag{53}$$

$$(r-t)\overline{L}(x_t, t, r) = \breve{L}(x_t, t, r) \tag{54}$$

$$(r-t)\frac{d\overline{L}}{dt} - \overline{L} = -L(x_t, t) \tag{55}$$

$$L(x_t, t) = \overline{L}(x_t, t, r) - (r-t)\frac{d\overline{L}}{dt}(x_t, t, r) \tag{56}$$

B.2 EDM COEFFICIENTS

The EDM authors consider in their eq. (2) a noise schedule $x = y + n$, which in ours can be represented as $z = x_1 + \sigma_n \epsilon$, for some noise scale σ_n and a noised output z. So an equivalent signal to noise ratio is achieved under

$$\sigma_n = \frac{1-t}{t} \tag{57}$$

$$x_t = t x_1 + (1-t)\epsilon = tz \tag{58}$$

Their parametrization is defined for a denoiser D, so we express it in terms of the score it can be compared to our c_x ahd c_F. For simplicity we use F regardless of the fact that inputs and conditioning change since that does not affect the coefficients themselves.

$$D(z) = c_{skip} z + c_{out} F(c_{in} z; \sigma_n) \tag{59}$$

$$\nabla_z \log p_z(z; \sigma_n) = \frac{D(z; \sigma_n) - z}{\sigma_n^2} = \frac{(c_{skip} - 1)z + c_{out} F(c_{in} z; \sigma_n)}{\sigma_n^2} \tag{60}$$

$$\nabla_z \log p_z(z; \sigma_n) = \frac{(c_{skip} - 1)t^2 z + c_{out} t^2 F(c_{in} z; \sigma_n)}{(1-t)^2} \tag{61}$$

$$t\nabla_x \log p_x(x; \sigma_n) = \frac{(c_{skip} - 1)t^2 z + c_{out} t^2 F(c_{in} z; \sigma_n)}{(1-t)^2} \tag{62}$$

$$\nabla_x \log p_x(x; t) = \frac{(c_{skip} - 1)x + c_{out} t F(x; t)}{(1-t)^2} \tag{63}$$

Given their choice of coefficients

$$c_{skip} = \frac{1}{1+\sigma_n^2} = \frac{t^2}{t^2 + (1-t)^2} \tag{64}$$

$$c_{out} = \frac{\sigma_n}{\sqrt{1+\sigma_n^2}} = \frac{1-t}{\sqrt{t^2 + (1-t)^2}} \tag{65}$$

The expression for the score becomes

$$\nabla_x \log p_x(x; t) = \frac{1}{(1-t)^2}\left[\frac{t^2}{t^2 + (1-t)^2} - 1\right] x + \frac{t F(x; t)}{(1-t)\sqrt{t^2 + (1-t)^2}} \tag{66}$$

$$\nabla_x \log p_x(x; t) = \underbrace{\frac{-1}{t^2 + (1-t)^2}}_{c_x^{\text{EDM}}} x + \underbrace{\frac{t}{(1-t)\sqrt{t^2 + (1-t)^2}}}_{c_F^{\text{EDM}}} F(x; t) \tag{67}$$

C ADDITIONAL DISCUSSION

C.1 POTENTIAL FIELD PARAMETRIZATION

Surprisingly, the ideas from Section 4.1 can be adapted to better understand and complete these forms for likelihood estimation. Ignoring regularity discussions and omitting the diffusion time t, the integral $\int_0^s \nabla A(\tau x)^T x d\tau = \phi(s\boldsymbol{x}) - \phi(\mathbf{0})$ can be interpreted as spatial expansion from the center with velocity x for time s. The direct representation we use $\mathbf{1}^T G(x)$ is then a (shifted) spatial equivalent of a displacement model, while the linear one is a mean field equivalent weighted by $\|x\|$ where $\nabla\phi(\mathbf{0}) = \overline{G}(\mathbf{0})$ holds for $s = 0$.

A theoretical conclusion is that these models would need to include some constant term to capture $\phi(0, t)$, which was not included in previous work since it is only needed for density estimation (Salimans & Ho, 2021) or was included but not as a function of time (Guth et al., 2026). The problematic situation $\|x\| \approx \mathbf{0}$ happens with low probability in high dimensions, so one may opt to not include it. Thus we propose the structure with the coefficient $c_G(t) = 0$ for the linear case and $c_G(t) = c_F(t)/c_x(t)$ for the square one.

$$\phi(\boldsymbol{x}_t, t) = x_t^T G(\boldsymbol{x}_t, t) + \frac{c_G}{2} G^T(\boldsymbol{x}_t, t) G(\boldsymbol{x}_t, t) \tag{68}$$

As discussed in the main text, a stopgrad trick can also be applied to the derivatives of ϕ in these "spatial velocity" parametrizations. To see why, note that the VJP, unlike JVP is implemented in PyTorch and other AD framework in two steps. The first forward step evaluates the function and stores intermediary activations, while the backward computes the product itself. Therefore the following computation is feasible with no extra cost

$$\nabla\phi = G + \nabla G^T x + \frac{c_G}{2} \nabla G^T G \tag{69}$$

$$\nabla\phi_F = G + \mathrm{VJP}_{\mathrm{sg}}(G; x + c_G G) \tag{70}$$

We speculate that the reason it is harder to train is due to the spatial sensitivity, similar to how training of integral models has been reported to be sensitive to the sensitivity to time as explored in the case of consistency models in Lu & Song (2025). Though the model was able to produce coarse samples at on the CIFAR at certain times during optimization, instabilities prevented stable performance.