Diffusion Mumbling

Ganchao Wei

09/22/2025

1. Weighting (unifying framework)

- DDPM:
$$\mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}\left\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0+\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon},t)\right\|^2\right]$$
$$L_{\text{simple}}(\theta):=\mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0+\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon},t)\right\|^2\right]$$
- Score Matching:
$$\frac{1}{2}\mathbb{E}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})p_{\text{data}}(\mathbf{x})}[\|\mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}})-\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}}\mid\mathbf{x})\|_2^2].$$
- Flow Matching:
$$\mathcal{L}_{\text{CFM}}(\theta)=\mathbb{E}_{t,q(x_1),p_t(x|x_1)}\left\|v_t(x)-u_t(x|x_1)\right\|^2,$$

Look similar (source is $\epsilon \sim N(0, I)$):
- Integrate over $t \sim U(0,1)$
- Conditional on endpoints x, build interpolation between ϵ and x

Differences:
- Network output
- Loss weight
- Interpolation design/ forward process

Let's unify them!

1. Weighting (unifying framework)

Stochastic Interpolants:
A Unifying Framework for Flows and Diffusions

Michael S. Albergo*[1], Nicholas M. Boffi*[2], and Eric Vanden-Eijnden[2]

[1]Center for Cosmology and Particle Physics, New York University
[2]Courant Institute of Mathematical Sciences, New York University

November 7, 2023

Understanding Diffusion Objectives as the ELBO with Simple Data Augmentation

Diederik P. Kingma
Google DeepMind
durk@google.com

Ruiqi Gao
Google DeepMind
ruiqig@google.com

Very personal comments

- ODE/SDE perspective
- Help understanding details
- (mainly) inspire sampling/ generation stage design

- Loss/ prob. perspective
- Help overall understanding
- (mainly) inspire training stage design

1. Weighting (unifying framework)

- 2 endpoints:
  - Data: $x \sim q(x)$
  - Noise: $\epsilon \sim N(0, I)$
- Latent variable (noise perturbed data): z_t, where $t \in [0,1]$
  - $t = 0$: data (x); $t = 1$: noise (ϵ)
- Models:
  - Forward model (0 → 1): $q(z_{0,\dots,1}|x)$
  - Reverse model/ generative model (**Goal!**) (1→0): $p(z_{0,\dots,1})$
- Goal: $D_{KL}\left(q(z_0) \approx q(x) \| p(z_0)\right) \approx 0$ (**good fit at data**)
  - If A) $D_{KL}\left(q(z_1|x) \| p(z_1)\right) \approx 0$,
  - and B) $s_\theta(z; \lambda) \approx \nabla_z \log q_t(z)$
  - → $D_{KL}\left(q\left(z_{0,\dots,1}\right) \| p\left(z_{0,\dots,1}\right)\right) \approx 0$ (**good fit at all noise perturbed data**)

1. Weighting (unifying framework)

- Interpolation/ forward process (extra details):
  - $z_t = \alpha_t x + \sigma_t \epsilon$ (stochastic version: $\alpha_t x + \sigma_t \epsilon + \sigma \xi$)
    - Variance preserving (VP): $\alpha_t^2 = 1 - \sigma_t^2$
    - Conditional OT (sub-VP): $\alpha_t = 1 - t, \sigma_t = t$
    - …
  - Any schedule can be scaled to VP
    - Let $s_t = 1/\sqrt{\alpha_t^2 + \sigma_t^2}$ → VP: $(s_t \alpha_t)^2 + (s_t \sigma_t)^2 = 1$ → $s_t z_t$
    - Just do upscaling when passing to neural net $\hat{g}_\theta(s_t z_t, t)$.
    - **What matters: SNR = $\frac{\alpha_t^2}{\sigma_t^2}$, invariant to t-scaling**
- Noise schedule ($\lambda(t)$): log-SNR

1. Weighting (unifying framework)

- Interpolation/ forward process (extra details):
  - $z_t = \alpha_t x + \sigma_t \epsilon$ (stochastic version: $\alpha_t x + \sigma_t \epsilon + \sigma\xi$)
  - VP after rescaling: $\alpha_t^2 + \sigma_t^2 = 1$
- Noise schedule: log-SNR ($\lambda(t)$)
  - $\lambda(t) = \lambda = f_\lambda(t) = \log \frac{\alpha_t^2}{\sigma_t^2}$ (stochastic version: $\lambda^* = \log \frac{\alpha_t^2}{\sigma_t^2 + \sigma^2}$)
  - **Strictly monotonically decreasing**
  - For $t \sim U(0,1)$, $p(\lambda) = -\frac{dt}{d\lambda} = -1/f_\lambda'(t)$
  - **We can use t and λ interchangeably**
    - $z_\lambda = \alpha_\lambda x + \sigma_\lambda \epsilon$, for $\epsilon \sim N(0, I)$

1. Weighting (unifying framework)

- Objective function:
  - Given $x \sim q(x)$,

$$L_w(x) = \frac{1}{2} E_{t\sim U(0,1),\epsilon\sim N(0,I)} \left[w(\lambda_t) \left\| \hat{g}_\theta(z_t, \lambda_t) - g_{\lambda_t} \right\|_2^2 \right]$$

  - Here,
    - Noise schedule (VP): $z_t = a_t x + \sigma_t \epsilon = \left(\sqrt{1 - \sigma_{\lambda_t}^2} \right) x + \sigma_{\lambda_t} \epsilon$
      - Equivalently, $\lambda = \log a_t^2 / \sigma_t^2 \sim p(\lambda)$
    - $g(\cdot)$: parametrization
    - $w(\lambda_t)$: weighting
- Seems there are 3 components → Let's check them one-by-one!

1. Weighting (unifying framework)

- Objective function: given $x \sim q(x)$,

$$L_w(x) = \frac{1}{2} E_{t \sim U(0,1), \epsilon \sim N(0,I)} \left[w(\lambda_t) \left\| \hat{g}_\theta(z_t, \lambda_t) - g_{\lambda_t} \right\|_2^2 \right]$$

  - Here, $z_t = a_t x + \sigma_t \epsilon = \alpha_{\lambda_t} x + \sigma_{\lambda_t} \epsilon$
- (1) Noise schedule → weighting:
  - λ is a **strictly monotonically decreasing** function of t
  - → time-warping
  - → can be absorbed by weight $w(\lambda_t)$
  - Let's re-define the objective as...

$$L_w(x) = \frac{1}{2} E_{t \sim U(0,1), \epsilon \sim N(0,I)} \left[\frac{w(\lambda_t)}{p(\lambda_t)} \left\| \hat{g}_\theta(z_t, \lambda_t) - g_{\lambda_t} \right\|_2^2 \right]$$

  - Then, new loss is invariant to $p(\lambda) = -\frac{dt}{d\lambda}$
  - But influence variance → importance sampling

1. Weighting (unifying framework)

- Objective function: given $x \sim q(x)$,

$$L_w(x) = \frac{1}{2} E_{t \sim U(0,1), \epsilon \sim N(0,I)} \left[-\frac{dt}{d\lambda} \cdot w(\lambda_t) \| \hat{g}_\theta(z_t, \lambda_t) - g_{\lambda_t} \|_2^2 \right]$$

  - Here, $z_t = a_t x + \sigma_t \epsilon = \alpha_{\lambda_t} x + \sigma_{\lambda_t} \epsilon$
- (1) Noise schedule → weighting:
  - Importance sampling: adaptive noise schedule

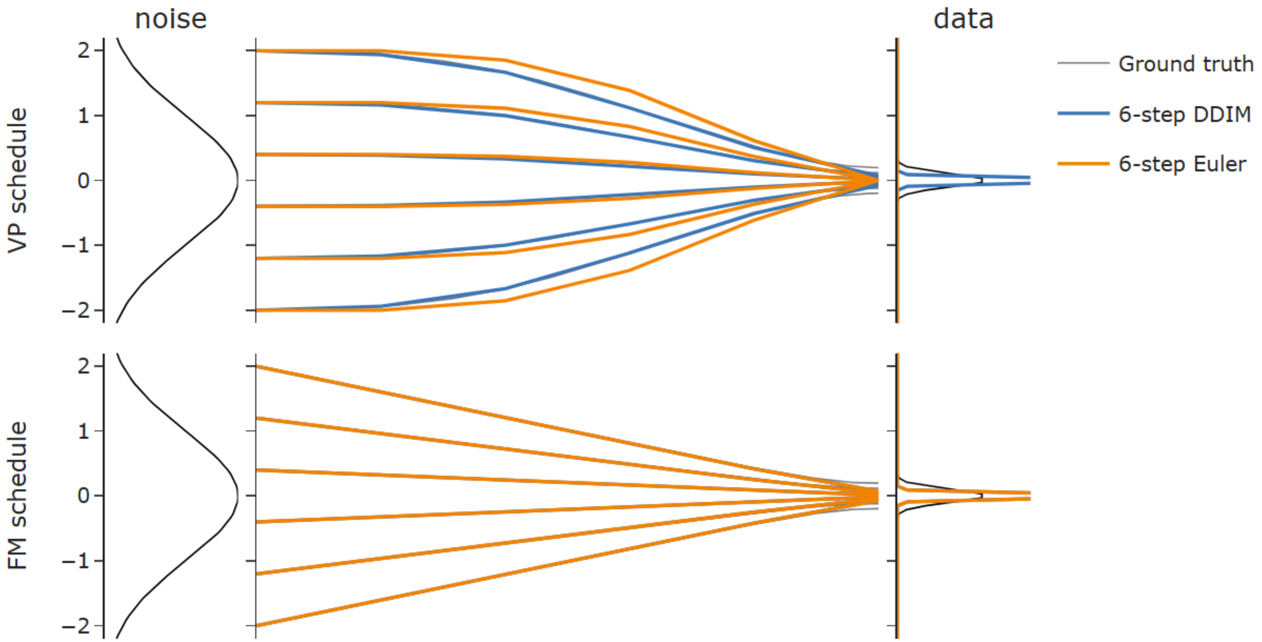
$$L_w(x) = \frac{1}{2} E_{\lambda \sim p(\lambda), \epsilon \sim N(0,I)} \left[\frac{w(\lambda)}{p(\lambda)} \| \hat{g}_\theta(z_\lambda, \lambda) - g_\lambda \|_2^2 \right]$$

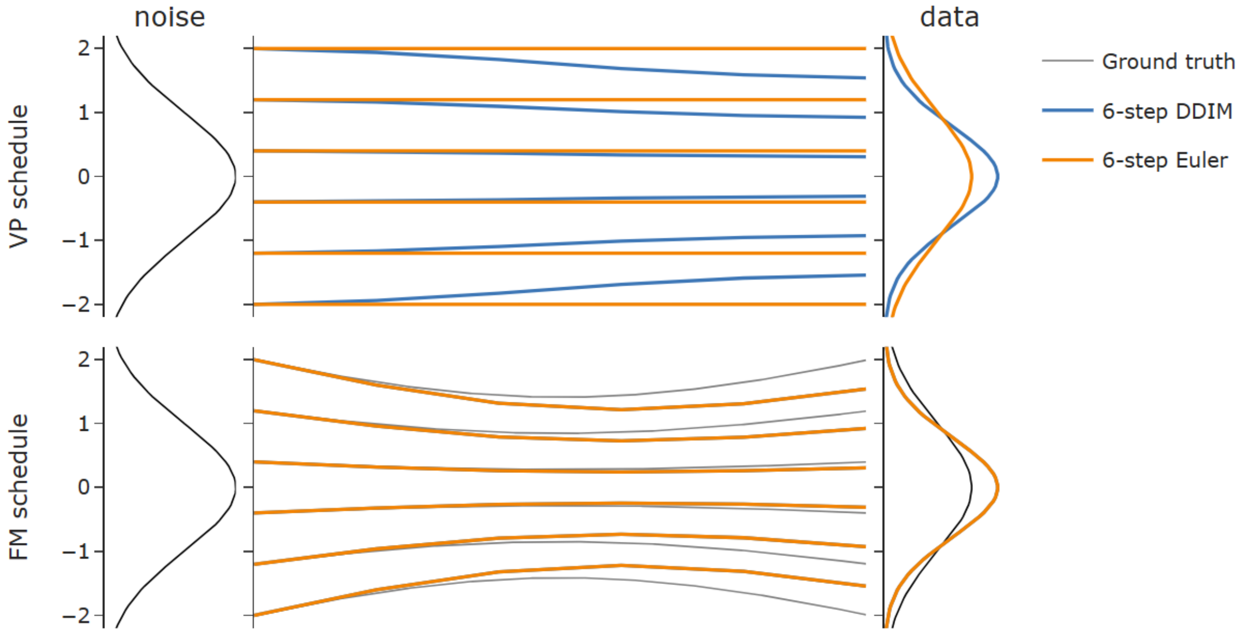
    - Let $p(\lambda) \propto E_{x,\epsilon}[w(\lambda) \| \hat{g}_\theta(z_\lambda, \lambda) - g_\lambda \|_2^2]$ → reduce variance
    - In practice: piecewise-linear function for $f_\lambda(t)$ EMA $w(\lambda) \| \hat{g}_\theta(z_\lambda, \lambda) - g_\lambda \|_2^2$
  - **Can use different noise schedule for training & sampling:**
    - Training: reduce variance
    - Sampling: reduce curvature & discretization error for numerical integration

1. Weighting (unifying framework)

- **Sampling:**
  - FM schedule: $\alpha_t = 1 - t, \sigma_t = t$ → $z_s = z_t + \hat{u}(s - t)$
  - FM schedule is the “straightest” guy in this world? →wrong!
    - FM is only straight for predicting a single point
    - But.. We average over large distribution
    - → straight to point (IS NOT EQUAL TO) straight to distribution
  - “Straightest” guy (schedule) depends on environment (data)
  - 2 general goals:
    - Low integration error
    - As straight as possible
    - Should have some adaptive ways? (literature review later…)

1. Weighting (unifying framework)





1. Weighting (unifying framework)

- Objective function: given $x \sim q(x)$,

$$L_w(x) = \frac{1}{2} E_{t \sim U(0,1), \epsilon \sim N(0,I)} \left[-\frac{dt}{d\lambda} \cdot w(\lambda_t) \|\hat{g}_\theta(z_t, \lambda_t) - g_{\lambda_t}\|_2^2 \right]$$

  - Here, $z_t = a_t x + \sigma_t \epsilon = \alpha_{\lambda_t} x + \sigma_{\lambda_t} \epsilon$
- (2) parametrization → weighting:
  - $f = \epsilon$(DDPM), $\nabla_z q(z|x)$ (score matching), v (flow matching)

$$\begin{aligned}
||\boldsymbol{\epsilon} - \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}||_2^2 &= e^{\lambda} ||\mathbf{x} - \hat{\mathbf{x}}_{\boldsymbol{\theta}}||_2^2 && (\boldsymbol{\epsilon}\text{-prediction and } \mathbf{x}\text{-prediction error}) && (122) \\
&= \sigma_\lambda^2 ||\nabla_{\mathbf{z}_\lambda} \log q(\mathbf{z}_\lambda|\mathbf{x}) - \mathbf{s}_{\boldsymbol{\theta}}||_2^2 && (\text{score prediction}) && (123) \\
&= \alpha_\lambda^{-2} (e^{-\lambda} + 1)^{-2} ||\mathbf{v} - \hat{\mathbf{v}}_{\boldsymbol{\theta}}||_2^2 && (\mathbf{v}\text{-prediction, general}) && (124) \\
&= (e^{-\lambda} + 1)^{-1} ||\mathbf{v} - \hat{\mathbf{v}}_{\boldsymbol{\theta}}||_2^2 && (\mathbf{v}\text{-prediction with VP SDE}) && (125) \\
&= (e^{-\lambda} / \tilde{\sigma}_{\text{data}}^2 + 1)^{-1} ||\mathbf{F} - \hat{\mathbf{F}}_{\boldsymbol{\theta}}||_2^2 && (\mathbf{F}\text{-prediction}) && (126)
\end{aligned}$$

1. Weighting (unifying framework)

- Objective function: given $x \sim q(x)$,
$$L_w(x) = \frac{1}{2} E_{t\sim U(0,1),\epsilon\sim N(0,I)}\left[-\frac{dt}{d\lambda} \cdot w(\lambda_t) \| \hat{g}_\theta(z_t, \lambda_t) - g_{\lambda_t} \|_2^2\right]$$
  - Here, $z_t = a_t x + \sigma_t \epsilon = \alpha_{\lambda_t} x + \sigma_{\lambda_t} \epsilon$
- (2) parametrization → weighting:
  - Matters in the original frameworks
    - Low SNR: $\alpha \downarrow, \sigma \uparrow$
      - ϵ is hard to learn, $\hat{x} = (z_t - \sigma_t \hat{\epsilon})/\alpha_t$ error amplified
    - High SNR: $\sigma \downarrow, \alpha \uparrow$
      - x_t is hard to learn, $\hat{\epsilon} = (z_t - \alpha_t \hat{x})/\sigma_t$ error amplified
    - Parametrize by vector field balance these 2.

1. Weighting (unifying framework)

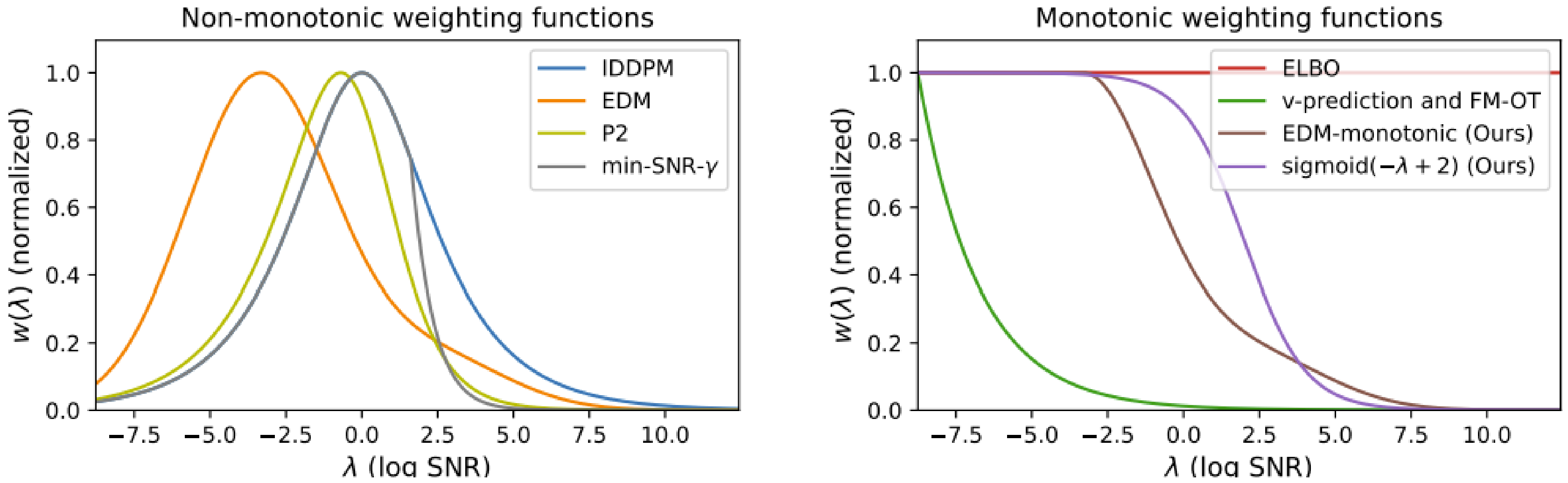
- Objective function: given $x \sim q(x)$,

$$L_w(x) = \frac{1}{2} E_{t\sim U(0,1),\epsilon\sim N(0,I)}\left[-\frac{dt}{d\lambda} \cdot w(\lambda_t) \|\hat{\epsilon}_\theta(z_t, \lambda_t) - \epsilon\|_2^2\right]$$

  - Here, $z_t = a_t x + \sigma_t \epsilon = \alpha_{\lambda_t} x + \sigma_{\lambda_t} \epsilon$

- (3) weighting

- Heavy weight on low SNR
- → DM is good at perceptual data (image/ sound)
- FM is super aggressive



1. Weighting (unifying framework)

- Objective function: given $x \sim q(x)$,
$$L_w(x) = \frac{1}{2} E_{t\sim U(0,1),\epsilon\sim N(0,I)}\left[-\frac{dt}{d\lambda} \cdot w(\lambda_t) \|\hat{\epsilon}_\theta(z_t, \lambda_t) - \epsilon\|_2^2\right]$$
  - Here, $z_t = a_t x + \sigma_t \epsilon = \alpha_{\lambda_t} x + \sigma_{\lambda_t} \epsilon$
- (3) weighting
  - Heuristic strategies, depend on your goals:
    - Minimize variance
    - Balance magnitude of gradients
    - Balance model capacity
    - Balance corruption rates across heterogenous data types
    - ...

1. Weighting (unifying framework)

- Objective function: given $x \sim q(x)$,

$$L_w(x) = \frac{1}{2} E_{t \sim U(0,1), \epsilon \sim N(0,I)} \left[-\frac{dt}{d\lambda} \cdot w(\lambda_t) \| \hat{\epsilon}_\theta(z_t, \lambda_t) - \epsilon \|_2^2 \right]$$

  - Here, $z_t = a_t x + \sigma_t \epsilon = \alpha_{\lambda_t} x + \sigma_{\lambda_t} \epsilon$
- So, now everything reduces to $w(\lambda)$!
- ICLR blog: https://d2jud02ci9yv69.cloudfront.net/2025-04-28-diffusion-flow-173/blog/diffusion-flow/

1. Weighting (unifying framework)

- Objective function: given $x \sim q(x)$,

$$L_w(x) = \frac{1}{2} E_{t \sim U(0,1), \epsilon \sim N(0,I)} \left[-\frac{dt}{d\lambda} \cdot w(\lambda_t) \|\hat{\epsilon}_\theta(z_t, \lambda_t) - \epsilon\|_2^2 \right]$$

  - Here, $z_t = a_t x + \sigma_t \epsilon = \alpha_{\lambda_t} x + \sigma_{\lambda_t} \epsilon$
- Ideas:
  - Learn distribution of stopping time → likelihood control → SNR control (see section 2)
    - Model has enough capacity: trade-off/ implementation settings
    - FM: aggressive/ efficient enough → need to “slow down”
  - Neuro:
    - Animal perception → diffusion?
    - Can we learn the weighting?

2. Likelihood/ ELBO & Diffusion loss

- Still from the "Understanding" paper
  - This is what Durk really good at, lol
- Define:

$$\mathcal{L}(t;\mathbf{x}) := D_{KL}(q(\mathbf{z}_{t,\dots,1}|\mathbf{x})||p(\mathbf{z}_{t,\dots,1}))$$

- Step 1: link KL to SNR:

$$\frac{d}{dt}\mathcal{L}(t;\mathbf{x}) = \frac{1}{2}\frac{d\lambda}{dt}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,\mathbf{I})}\left[||\epsilon - \hat{\epsilon}_{\theta}(\mathbf{z}_{\lambda};\lambda)||_2^2\right]$$

$$\mathcal{L}_w(\mathbf{x}) = -\int_0^1 \frac{d}{dt}\mathcal{L}(t;\mathbf{x})\, w(\lambda_t)\; dt$$

$$\mathcal{L}_w(\mathbf{x}) = \int_0^1 \frac{d}{dt}w(\lambda_t)\,\mathcal{L}(t;\mathbf{x})\; dt + w(\lambda_{\text{max}})\,\mathcal{L}(0;\mathbf{x}) + \text{constant}$$

Proof is very easy to follow, but the key step is in Kingma et al., 2021 😂

2. Likelihood/ ELBO & Diffusion loss

- If…
  - $w(\lambda_t)$ is monotonically increasing to t (decreasing for λ)
  - w.l.o.g., assume $w(\lambda_1) = 1$

$$\mathcal{L}_w(\mathbf{x}) = \mathbb{E}_{p_w(t)}\left[\mathcal{L}(t; \mathbf{x})\right] + \text{constant}$$

  - , where $p_w(t) = \frac{dw(\lambda_t)}{dt} + \delta_0 w(\lambda_{max})$
- Step 2: link KL to ELBO

2. Likelihood/ ELBO & Diffusion loss

$$\mathcal{L}(t;\mathbf{x}) = D_{KL}(q(\mathbf{z}_{t,\dots,1}|\mathbf{x})||p(\mathbf{z}_{t,\dots,1})) = -\mathbb{E}_{q(\mathbf{z}_t|\mathbf{x})}[\mathrm{ELBO}_t(\mathbf{z}_t)] - \underbrace{\mathcal{H}(q(\mathbf{z}_t|\mathbf{x}))}_{\text{constant}} \tag{33}$$

where the ELBO of noise-perturbed data is:

$$\mathrm{ELBO}_t(\mathbf{z}_t) := \mathbb{E}_{q(\tilde{\mathbf{z}}_t|\mathbf{z}_t)}[\log p(\mathbf{z}_t, \tilde{\mathbf{z}}_t) - \log q(\tilde{\mathbf{z}}_t|\mathbf{z}_t)] \tag{34}$$

$$\leq \log p(\mathbf{z}_t) \tag{35}$$

$$\mathcal{L}_w(\mathbf{x}) = \mathbb{E}_{p_w(t)}\left[\mathcal{L}(t;\mathbf{x})\right] + \text{constant} \tag{37}$$

$$= -\underbrace{\mathbb{E}_{p_w(t),q(\mathbf{z}_t|\mathbf{x})}\left[\mathrm{ELBO}_t(\mathbf{z}_t)\right]}_{\text{ELBO of noise-perturbed data}} + \text{constant} \tag{38}$$

$$\geq -\underbrace{\mathbb{E}_{p_w(t),q(\mathbf{z}_t|\mathbf{x})}\left[\log p(\mathbf{z}_t)\right]}_{\text{Log-likelihood of noise-perturbed data}} + \text{constant} \tag{39}$$

2. Likelihood/ ELBO & Diffusion loss

$$\mathcal{L}_w(\mathbf{x}) = \mathbb{E}_{p_w(t)}\left[\mathcal{L}(t;\mathbf{x})\right] + \text{constant} \tag{37}$$

$$= -\underbrace{\mathbb{E}_{p_w(t),q(\mathbf{z}_t|\mathbf{x})}\left[\text{ELBO}_t(\mathbf{z}_t)\right]}_{\text{ELBO of noise-perturbed data}} + \text{constant} \tag{38}$$

- Note that:
  - When $w(t) = 1 \rightarrow p_w = \delta_0$
  - $z_0 = x$
- So, when $w(t) = 1$ …

$$L_w(x) = -ELBO(x) + constant$$

- Now, we link likelihood to diffusion loss: lots of potentials!
  - Likelihood/ ELBO is usually very hard to calculate
  - Diffusion loss is very trivial

2. Likelihood/ ELBO & Diffusion loss (e.g.1)

- Example 1: Policy gradient → FM policy gradient

Flow Matching Policy Gradients

David McAllister[1*] Songwei Ge[1*] Brent Yi[1*] Chung Min Kim[1]
Ethan Weber[1] Hongsuk Choi[1] Haiwen Feng[1,2] Angjoo Kanazawa[1]
[1] UC Berkeley [2] Max Planck Institute for Intelligent Systems

2. Likelihood/ ELBO & Diffusion loss (e.g.1)

- {Obs., action, reward} = $\{o_t, a_t, r_t\}$
- Quick recap of PG:
  - Objective (use advantage to reduce var): $\max_{\theta} \mathbb{E}_{a_t \sim \pi_\theta(a_t|o_t)} \left[\log \pi_\theta(a_t \mid o_t) \hat{A}_t \right],$
  - biased, unstable: likelihood → likelihood ratio to old

$$\max_{\theta} E_{a_t \sim \pi_{\theta_{old}}(a_t|o_t)} [r(\theta)\, \hat{A}_t]$$

  - , where $r(\theta) = \dfrac{\pi_\theta(a_t \mid o_t)}{\pi_{\text{old}}(a_t \mid o_t)}.$
  - To further stability → clipping → PPO

$$\max_{\theta} \mathbb{E}_{a_t \sim \pi_{\theta_{\text{old}}}(a_t|o_t)} \left[\min \left(r(\theta)\hat{A}_t,\ \text{clip}(r(\theta), 1-\varepsilon^{\text{clip}}, 1+\varepsilon^{\text{clip}})\hat{A}_t \right) \right]$$

2. Likelihood/ ELBO & Diffusion loss (e.g.1)

- Key problem: likelihood ratio

$$r(\theta) = \frac{\pi_\theta(a_t \mid o_t)}{\pi_{\text{old}}(a_t \mid o_t)}.$$

- Likelihood: hard to evaluate...
  - Replace $\log(\pi(a_t|o_t))$ by $\mathbb{E}_{p_w(\tau), q(a_t^\tau | a_t)}[\text{ELBO}_\tau(a_t^\tau)]$
  - When $w(t) = 1$: just ELBO(a_t)
  - Then, remaining is very trivial...

$$\hat{r}^{\text{FPO}}(\theta) = \exp(\hat{\mathcal{L}}_{\text{CFM},\theta_{\text{old}}}(a_t; o_t) - \hat{\mathcal{L}}_{\text{CFM},\theta}(a_t; o_t)),$$

$$\hat{\mathcal{L}}_{\text{CFM},\theta}(a_t; o_t) = \frac{1}{N_{\text{mc}}} \sum_{i}^{N_{\text{mc}}} \ell_\theta(\tau_i, \epsilon_i)$$

$$\ell_\theta(\tau_i, \epsilon_i) = ||\hat{v}_\theta(a_t^{\tau_i}, \tau_i; o_t) - (a_t - \epsilon_i)||_2^2$$

$$a_t^{\tau_i} = \alpha_{\tau_i} a_t + \sigma_{\tau_i} \epsilon_i,$$

2. Likelihood/ ELBO & Diffusion loss (e.g.2s)

- Example 2: imitation learning

Diffusion Imitation from Observation

(NeurIPS’24)

Bo-Ruei Huang Chun-Kai Yang Chun-Mao Lai Dai-Jie Wu Shao-Hua Sun
Department of Electrical Engineering, National Taiwan University

4.1 Modeling expert transitions via diffusion model

Motivated by the recent success in using diffusion models for generative modeling, we use a conditional diffusion model to model expert state transitions. Specifically, given a state transition $(\mathbf{s}, \mathbf{s}')$, the diffusion model conditions on the current state $\mathbf{s}$ and generates the next state $\mathbf{s}'$. We adopt DDPM [27] and define the reverse process as $p_\phi(\mathbf{s}'_{t-1}|\mathbf{s}'_t, \mathbf{s})$, where $t \in T$ and ϕ is the diffusion model, which is trained by minimizing the denoising MSE loss:

$$\mathcal{L}_d(\mathbf{s}, \mathbf{s}') = \mathbb{E}_{t\sim T, \boldsymbol{\epsilon}\sim\mathcal{N}(0,1)}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\phi(\boldsymbol{s}'_t, t|\boldsymbol{s})\|^2\right], \tag{1}$$

4.2 Diffusion model as a discriminator

The previous section describes how we can use the denoising loss as a reward for policy learning via reinforcement learning. However, the policy can learn to exploit a frozen diffusion model by discovering states that lead to a low denoising loss while being drastically different from expert states. To mitigate this issue, we incorporate principles from the AIL framework by training the diffusion model to recognize both the transitions from the expert and agent. To this end, we additionally condition the model on a binary label $c \in \{c_E, c_A\}$, where c_E represents the expert label and c_A represents the agent label, both implemented as one-hot encoding, resulting in the following denoising losses given a state transition $(\mathbf{s}, \mathbf{s}')$:

$$\mathcal{L}_d^E(\mathbf{s}, \mathbf{s}') = \mathbb{E}_{t\sim T, \boldsymbol{\epsilon}\sim\mathcal{N}(0,1)}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\phi(\boldsymbol{s}'_t, t|\boldsymbol{s}, c_E)\|^2\right], \tag{2}$$

$$\mathcal{L}_d^A(\mathbf{s}, \mathbf{s}') = \mathbb{E}_{t\sim T, \boldsymbol{\epsilon}\sim\mathcal{N}(0,1)}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\phi(\boldsymbol{s}'_t, t|\boldsymbol{s}, c_A)\|^2\right]. \tag{3}$$

2. Likelihood/ ELBO & Diffusion loss (e.g.2)

DIFFUSING STATES AND MATCHING SCORES: A NEW FRAMEWORK FOR IMITATION LEARNING

Runzhe Wu
Cornell University
rw646@cornell.edu

Yiding Chen
Cornell University
yc2773@cornell.edu

Gokul Swamy
Carnegie Mellon University
gswamy@andrew.cmu.edu

Kianté Brantley
Harvard University
kdbrantley@g.harvard.edu

Wen Sun
Cornell University
ws455@cornell.edu

(ICLR'25)

Definition 1 (Diffusion Score Divergence). *For two distributions P and Q, we define the* Diffusion Score Divergence (DS Divergence) *as*

$$D_{\mathrm{DS}}(P,Q) := \mathop{\mathbb{E}}_{s\sim P}\ \mathop{\mathbb{E}}_{t\sim U(T)}\ \mathop{\mathbb{E}}_{s_t\sim q_t(\cdot\,|\,s)} \left\|\nabla \log P_t(s_t) - \nabla \log Q_t(s_t)\right\|_2^2.$$

Algorithm 1 `SMILING` (**S**core-**M**atching **I**mitation **L**earn**ING**)

Require: state-only expert demonstration $\mathcal{D}^e = \{s^{(i)}\}_{i=1}^N$

1: Estimate score function of expert state distribution:

$$g^e \leftarrow \operatorname*{argmin}_{g\in\mathcal{G}} \mathop{\mathbb{E}}_{s\sim\mathcal{D}^e}\ \mathop{\mathbb{E}}_{t\sim U(T)}\ \mathop{\mathbb{E}}_{s_t\sim q_t(\cdot\,|\,s)} \left[\|g(s_t,t) - \nabla_{s_t}\log q_t(s_t\,|\,s)\|_2^2\right]$$

2: **for** $k = 1, 2, \ldots, K$ **do**

3: Estimate the score function of learner state distributions:

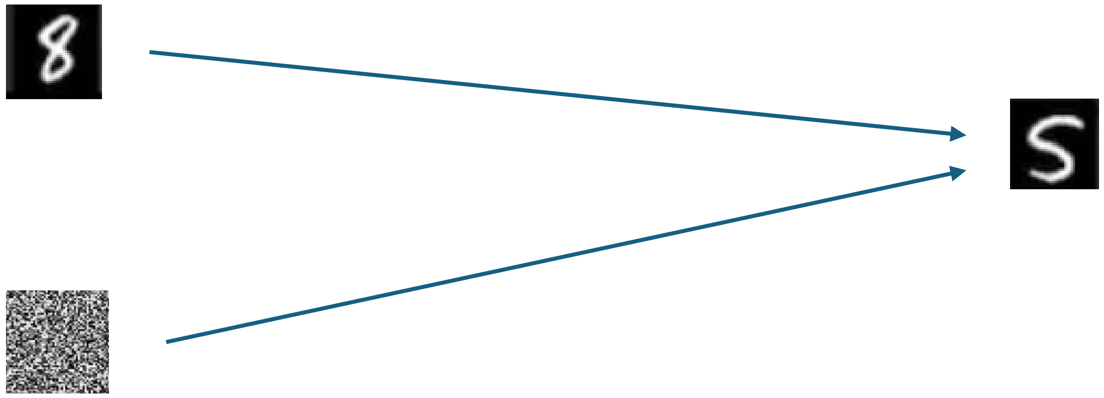
$$g^{(k)} \leftarrow \operatorname*{argmin}_{g\in\mathcal{G}} \sum_{i=1}^{k-1} \mathop{\mathbb{E}}_{s\sim d^{\pi^{(i)}}}\ \mathop{\mathbb{E}}_{t\sim U(T)}\ \mathop{\mathbb{E}}_{s_t\sim q_t(\cdot\,|\,s)} \left[\|g(s_t,t) - \nabla_{s_t}\log q_t(s_t\,|\,s)\|_2^2\right].$$

4: Update policy $\pi^{(k)}$ via RL (e.g., SAC) on cost $c^{(k)}$ (Eq. 3): $\pi^{(k)} \leftarrow \mathrm{RL}(c^{(k)})$

5: **end for**

3. Prior help for DM

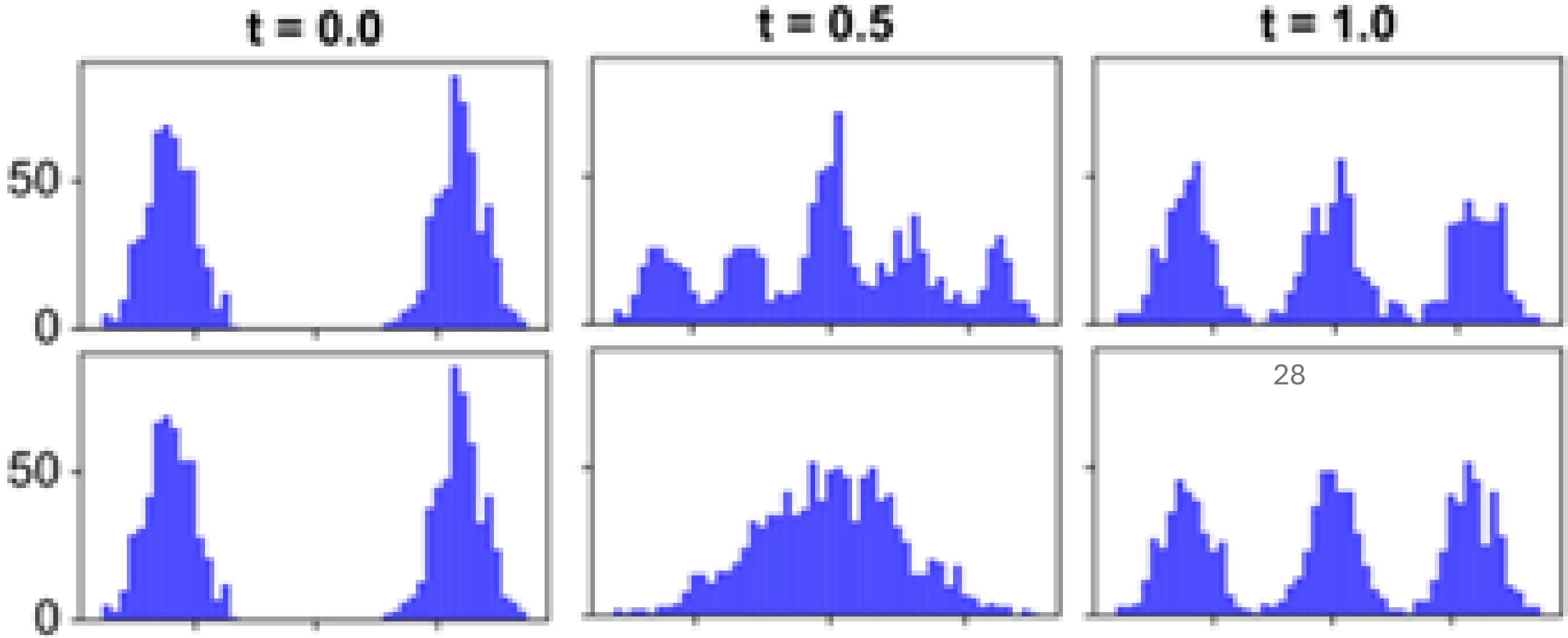
- Which is easier?



3. Prior help for DM

Linear deterministic

stochastic



😝? 😔!

3. Prior help for DM

- Good prior should help DM training a lot
  - Also, transformation problem: e.g. left face → right face

- BTW, simply incorporating prior is a standard problem.
- Current framework: encode prior in network (e.g. score/ velocity) fitting
  - Use score-network as example (others need more derivations).
- Classifier guidance (~ outdated):
  - $p_\gamma(x|y) \propto p(x)p(y|x)^\gamma$
  - → $\nabla_x \log_\gamma p(x|y) = \nabla_x \log p(x) + \gamma\nabla_x \log p(y|x)$
  - Nasty classifier: noise-robust? Is x really informative y?

3. Prior help for DM

- Classifier-free guidance (mainly use):
  - Use Bayes rule on $p(y|x)$ again
  - → $\nabla_x \log_\gamma p(x|y) = (1-\gamma)\nabla_x \log p(x) + \gamma\nabla_x \log p(x|y)$
  - γ can > 1
  - “Classifier” trained by DM → easy
  - In practice: single training + conditioning dropout ($y = \phi$)
  - Similar results can be shown in FM:
    - $v(x|y) = (1-\gamma)v(x) + \gamma \cdot v(x|y)$ (CFG-Zero⋆: 2025 arXiv)

4. Sampling/ generation

- Speed up in 2 ways:
  - Skip some steps, e.g., DDIM/ Taylor expansion
  - Parallel (should be a trivial implementation of Scott's method)
- Noise schedule design: previous
- Diversity issue