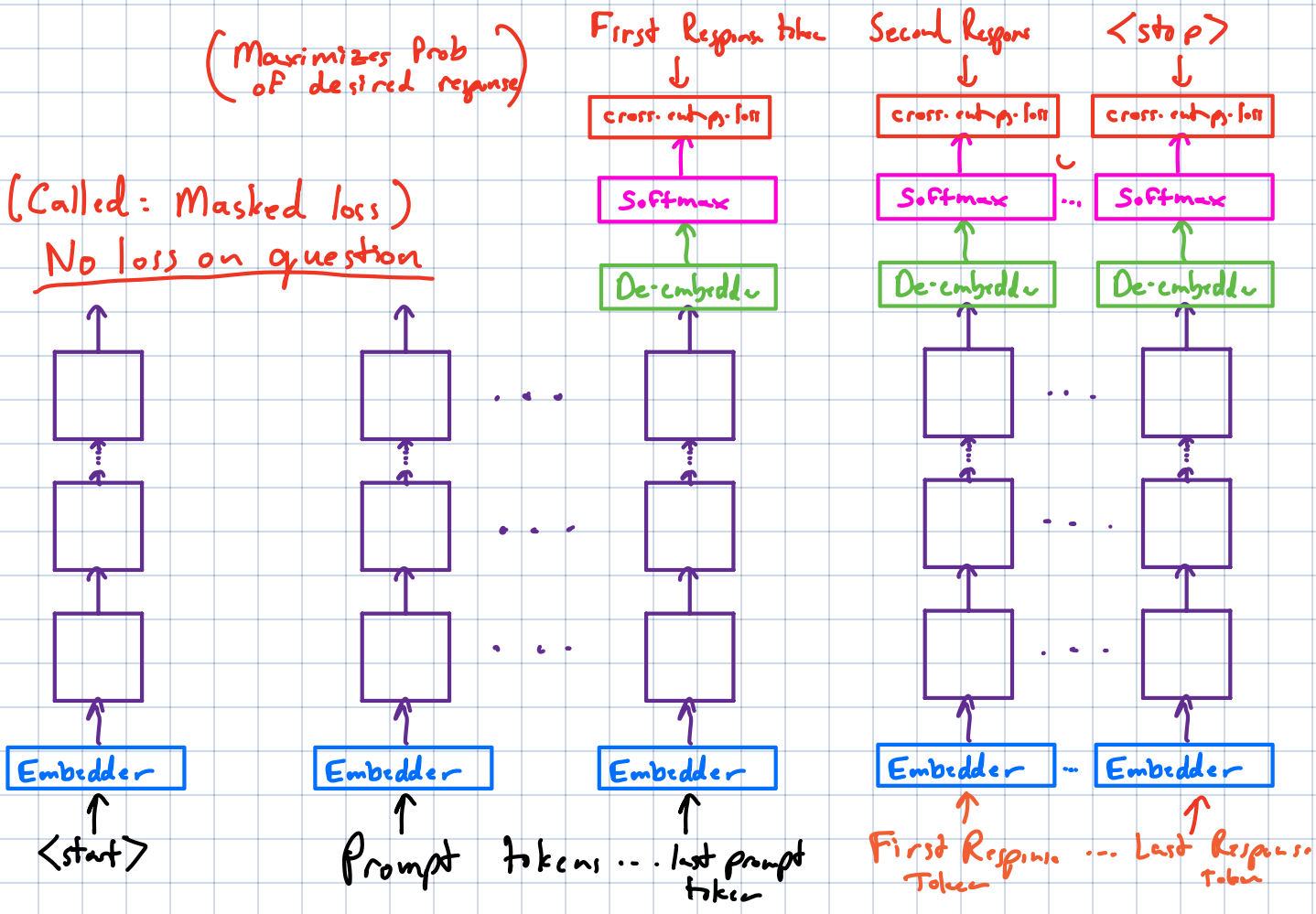
Today: RL Post-training (finish)
Diffusion Models

Reading in Textbook: Ch 17 (VAE)
Ch 19.5 (Some RL...)
Ch 18 (Diffusion)

Announce: 1) Survey: 40% ☹
We'll award 1.5% extra credit
2) Final Project Report: ~~Fri Dec 5th~~ Sun Dec 7
Poster Session Tue Dec 9th
Final Peer Review Thu Dec 11th
Final Final Report Sun Dec 14th
3) Final Exam: Wed Dec 17th 8-11AM

Post-training:
Recall basic SFT for instruction following.



But how can we make a model better at solving problems?

Two parts to the answer:

A) Be willing to spend more compute while answering.
↑ test-time compute

B) Train it to be better.

RLVR: Reinforcement Learning with Verifiable Rewards

Key new paper: Khatri, et al "The Art of Scaling Reinforcement Learning Compute for LLMs," Oct 15, 2025.

Treatment here influenced by this, along with Deepseek paper.

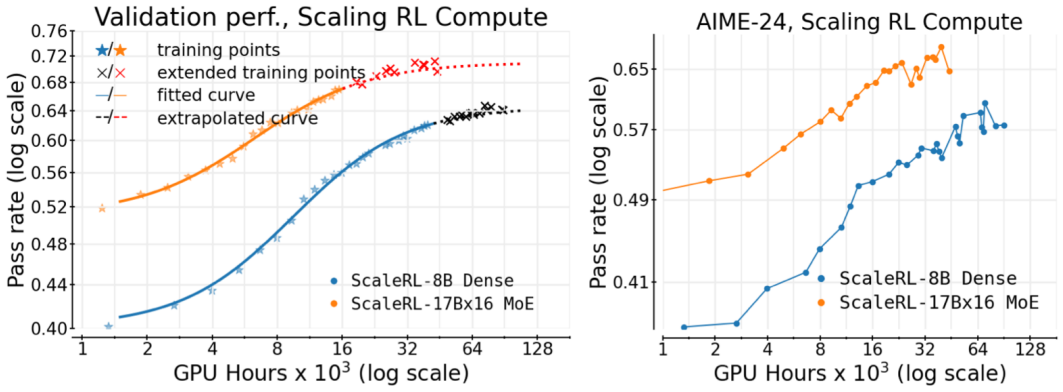


Fig 1 in ScaleRL...

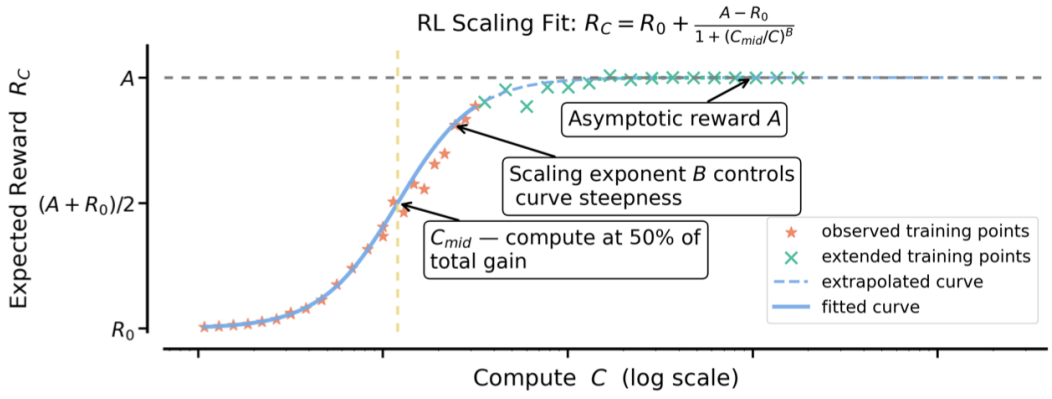


Fig 3 in ScaleRL

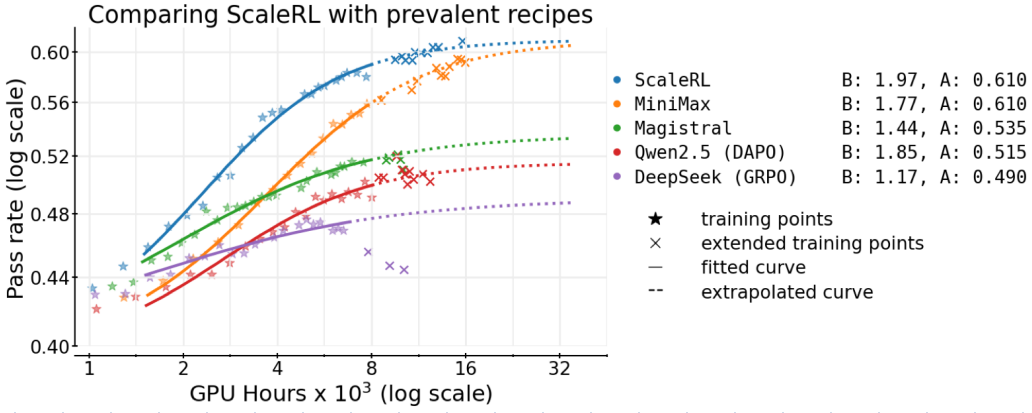


Fig 2 in ScaleRL...

"Bag of tokens perspective"

Unpacking what to do....

$$\mathcal{J}_{\text{ScaleRL}}(\theta) = \mathop{\mathbb{E}}_{\substack{x\sim D,\\ \{y_i\}_{i=1}^{G}\sim\pi_{gen}^{\theta_{old}}(\cdot|x)}}\left[\frac{1}{\sum_{g=1}^{G}|y_g|}\sum_{i=1}^{G}\sum_{t=1}^{|y_i|}\text{sg}(\min(\rho_{i,t},\epsilon))\hat{A}_i^{\text{norm}}\log\pi_{train}^{\theta}(y_{i,t})\right],$$

$$\rho_{i,t}=\frac{\pi_{train}^{\theta}(y_{i,t})}{\pi_{gen}^{\theta_{old}}(y_{i,t})},\quad \hat{A}_i^{\text{norm}}=\hat{A}_i/\hat{A}_{\text{std}},\quad 0<\text{mean}(\{r_j\}_{j=1}^{G})<1,\quad \text{pass\_rate}(x)<0.9,$$

Generations
stopGradient
#HW 3 log P_θ
"Importance Sampling" Distribution Mismatch.
Demean across G generations reward
↑ VR gives this.
Don't Bother if we get 90% right.

Recall HW3 Trick (Reparameterization Gradient Estimator)

$$\nabla_\theta \mathbb{E}_{y\sim P_\theta}[F(y)] = \mathbb{E}_{y\sim P_\theta}[F(y)\nabla_\theta \log P_\theta]$$

$\text{sg}(\cdots)$ ← To stop the gradient from inadvertently going into $\rho_{i,t}$.

$$\min(\rho_{i,t},\epsilon)=\begin{cases}\epsilon & \text{if } \rho_{i,t}\geq\epsilon\\ \rho_{i,t} & \text{if } \rho_{i,t}<\epsilon\end{cases}$$

→ Caps the overweighting of this sample in case it is strongly preferred by π_{train} as opposed to π_{gen}.

In general $\pi_{gen}^{\theta_{old}}$ closely tracks π_{train}^{θ} since it gets updated with a lag.

KL Regularization with a reference policy π_{ref} and reward function r

$$\underset{\pi}{\operatorname{argmax}} \; \mathop{\mathbb{E}}_{\substack{x, \\ y \sim \pi(\cdot|x)}} [r(x,y)] - \beta D_{KL}[\pi \,||\, \pi_{ref}]$$

has solution $\frac{1}{Z(x)} \pi_{ref}(y|x) e^{\frac{1}{\beta} r(x,y)}$

Moves mass towards higher reward

e.g. GRPO has KL Regularization.

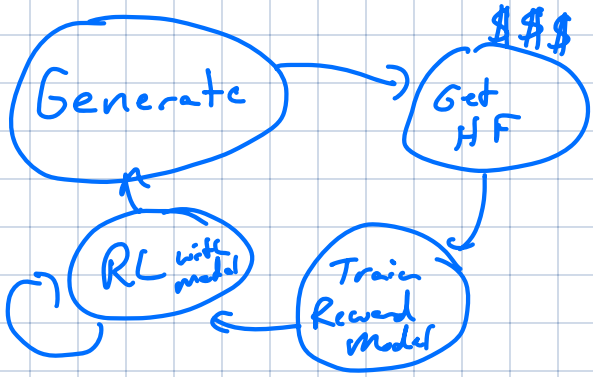
RLHF: RL from human feedback.

Key constraints/considerations:

A) Humans are expensive ⇒ Use human responses to train AI-feedback / reward model

B) Humans aren't good at real-valued feedback ⇒ Ask for binary comparisons

Pretrain → SFT on Good Examples ⇒



DPO: Direct Preference Optimization

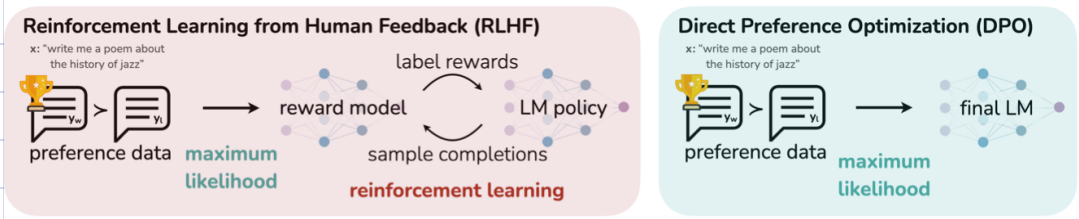


Fig 1 from the paper "Direct Preference Optimization: Your LM is secretly a reward model" by Rafailov, et al. 2023

Can we work with only the prefecence outputs? $\vec{y}_w$ "winner" vs. $\vec{y}_\ell$ "loser"

Idea 0: Just maximize $\log \pi_\theta(\vec{y}_w) - \log \pi_\theta(\vec{y}_\ell)$ over dataset

Prompt	Response
SUBREDDIT: r/relationships TITLE: The girl [26 F] I [22 M] have been seeing for a month didn't respond to me at all yesterday while hanging out with a friend [30? M]. POST: She gets terrible service while at her house, but I texted her 3 times yesterday, 4-5 hours apart. She didn't call me until early this morning and left a voicemail that she was busy all day with a friend who showed up out of the blue. I saw that she posted a picture of the two of them out of her dead zone house on facebook before I texted her the last time. I don't mind that she hangs out with friends, and I know it's pretty early [. . .] TL;DR:	girl whenUB when when when when whenUB when when when whenUB
SUBREDDIT: r/tifu TITLE: TIFU by accidently kicking an old woman POST: So this didn't happen today but actually about a year or two ago. I was at my granddads funeral so of course it was all very sad and full of lots of crying old people. After the ceremony everyone walks outside the building and onto the other side of the small road the hearses drive down. Now the road is important because obviously if there's a road, there's a curb onto the sidewalk, so most of us are on the other side of the road, besides a few older people walking a lot slower. As one of the old woman goes to walk up the curb [. . .] TL;DR:	when an old woman was tripping the when

Table 3 from paper

Total Meltdown ☹

Idea 1: Just SFT on winners... Not horrible.

Idea 2: (DPO) Use Idea 1 to get π_{ref} ← A reference policy

Then $$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta\log\frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right)\right].$$

prompt

(start at $\pi_\theta = \pi_{ref}$)

sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$

→ Just SGD on preference pairs

Question: If no gradients enter π_{ref} and $\log\frac{a}{b} = \log a - \log b$, what's the point of the π_{ref}?

We're maximizing $\log\sigma\left(\beta\log\frac{\pi_\theta(y_w|x)}{\pi_\theta(y_l|x)} + \beta\log\frac{\pi_{ref}(y_l|x)}{\pi_{ref}(y_w|x)}\right)$

Bias term for sigmoid nonlinearity

$$\nabla_\theta\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) =$$
$$-\beta\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\Big[\underbrace{\sigma(\hat{r}_\theta(x,y_l) - \hat{r}_\theta(x,y_w))}_{\text{higher weight when reward estimate is wrong}}\Big[\underbrace{\nabla_\theta\log\pi(y_w \mid x)}_{\text{increase likelihood of } y_w} - \underbrace{\nabla_\theta\log\pi(y_l \mid x)}_{\text{decrease likelihood of } y_l}\Big]\Big],$$

where $\hat{r}_\theta(x,y) = \beta\log\frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$ ← i.e. What is good? What π_θ likes to generate!

Idea 3: (TR-DPO) Keep updating π_{ref} every τ training steps to be π_θ

e.g. $\tau \sim 512$

Diffusion Models for Generation... "Test-time compute"

Autoregressive Generation builds generations one chunk/token at a time. Learns a conditional distribution. Leverage inherent sequentiality.

What if we don't have inherent sequentiality? e.g. images

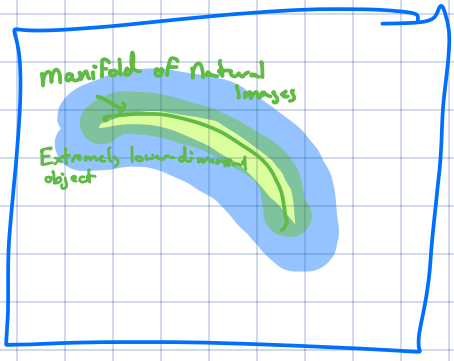
Recall from Lecture 18:

Ideas that don't work

A) Use a classifier

Try: Random Uniform Image followed by Gradient Ascent on the Cat Score

Result: Noise-like image that classifier confidently classifies as a cat.



Bring insights from VAE story: Need to see noisy things during training

But how noisy?

Two extremes clear:

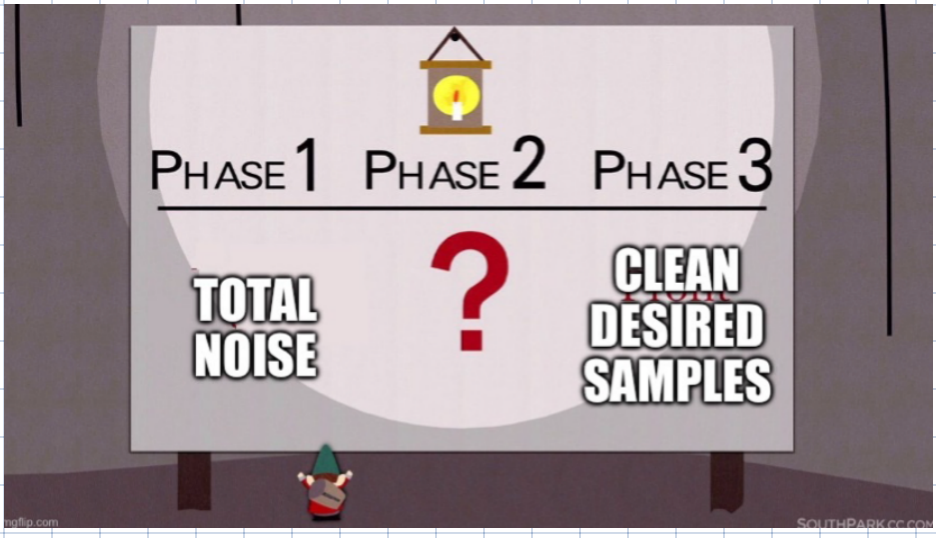


Figure adapted from South Park to parody/meme

Why not everything in between too? Can sample from new natural images by adding noise.

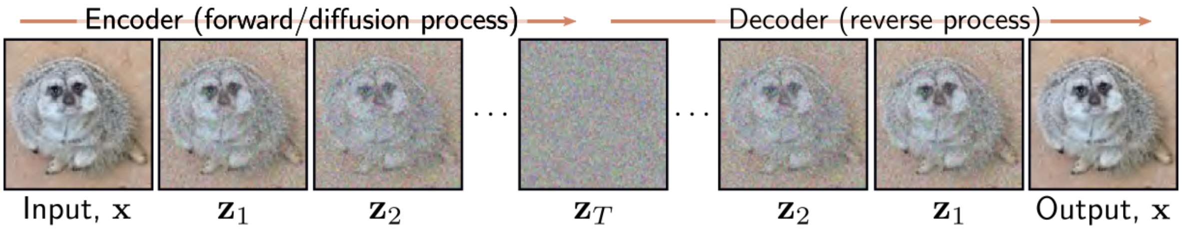
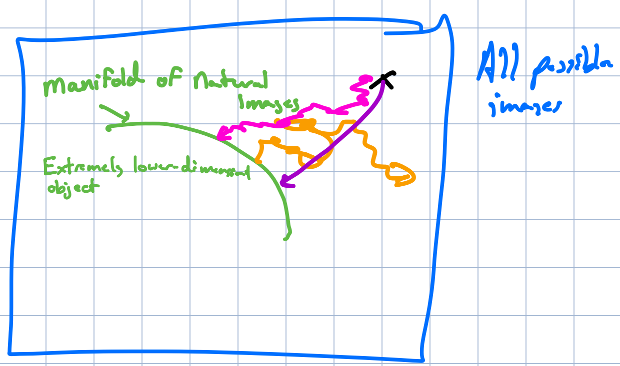


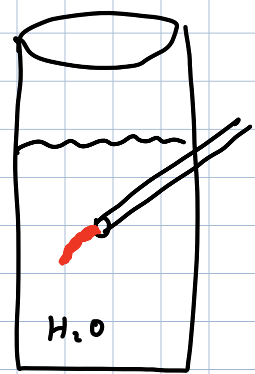
Fig 18.1 in Prince

$$\mathbf{z}_1 = \sqrt{1-\beta_1}\cdot\mathbf{x} + \sqrt{\beta_1}\cdot\boldsymbol{\epsilon}_1$$
$$\mathbf{z}_t = \sqrt{1-\beta_t}\cdot\mathbf{z}_{t-1} + \sqrt{\beta_t}\cdot\boldsymbol{\epsilon}_t \qquad \forall\, t\in 2,\ldots,T,$$

Traditional Perspective from Sohl-Dickstein, et. al.'s "Deep Unsupervised learning using Non-equilibrium Thermodynamics" in ICML 2015.

For clarity, we'll mostly follow arxiv 2406.08929 from June 2024: Nakkiran, et. al. "Step-by-step Diffusion: An elementary tutorial"





Idea: Take sample paths of forward diffusion starting at $\{x_{train,i}\}$

$$x_0 \to x_1 \to x_2 \to x_3 \to \cdots\cdots\cdots \; x_{T-1} \to x_T$$

Train a network to predict backwards in time $x_{t+1} \to \hat{x}_t$

Somehow use this learned network to simulate a path backwards from x_T to x_0.

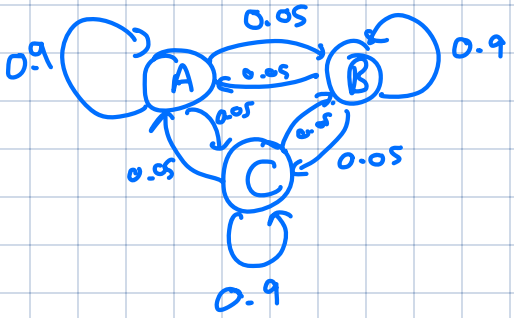
Key Design Question: Do I need randomness on the backward path

Two approaches: stochastic sampling (DDPM)

deterministic sampling (DDIM - next time)

Stochastic Case → Simulate Reverse Process

For Intuition. Consider discrete-time discrete-state



Stand $[\frac{1}{2}, 0, \frac{1}{2}] \longrightarrow [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$
P(A) P(B) P(C)

Is $p(X_{t-1} | X_t)$ like a Gaussian?

(Assume $X_t \to X_{t+1}$ is conditionally Gaussian. e.g. $X_{t+1} = X_t + N_t$)
N_t: $N(0, \sigma^2 I)$

As long as density for x_0 sufficiently smooth, if σ^2 small enough,

$$p(x_{t-1} | x_t = \vec{z}) \approx N(\vec{\mu}_{t-1}(\vec{z}), \sigma^2 I)$$

Need to learn this from data.

Let's take a continuous-time perspective. $T \to 1$. Δt small step.

$$x_0 \to x_{\Delta t} \to x_{2\Delta t} \to x_{3\Delta t} \to \cdots \cdots x_{1-\Delta t} \to x_1$$

Noise added in one step is $N(0, \sigma^2 \Delta t)$

Want to get $\hat{x}_{t-\Delta t}$ given $\hat{x}_t$ going backwards

To do this in DDPM, $\hat{x}_{t-\Delta t} = \mu_t(\hat{x}_t) + N(0, \sigma^2 \Delta t)$

Add noise going backwards

How to train $\vec{\mu}_t(\vec{x}_t)$ ← Deep Network

1) Pick $\vec{x}_0$ from training set
2) Pick time t in $[0,1]$
3) Construct $\vec{x}_{t-\Delta t} = \vec{x}_0 + N(0, \sigma^2(t-\Delta t))$

$$\vec{x}_t = \vec{x}_{t+\Delta t} + N(0, \sigma^2 \Delta t)$$

4) SGD step for $\vec{\mu}_t(\vec{x}_t)$ to predict $\vec{x}_{t-\Delta t}$ under squared-loss