Fine-Tuning Diffusion Models via Intermediate Distribution Shaping

SPEAKER

Dheeraj Nagaraj

Google Deepmind

Special thanks to my collaborators

Gautham Govind Anil (DeepMind)

Nithish Kannen (DeepMind)

Karthikeyan Shanmugam (DeepMind)

Shaan Ul Haque (Georgia Tech)

Sanjay Shakkottai (UT Austin)

Generative Modeling and RL Alignment

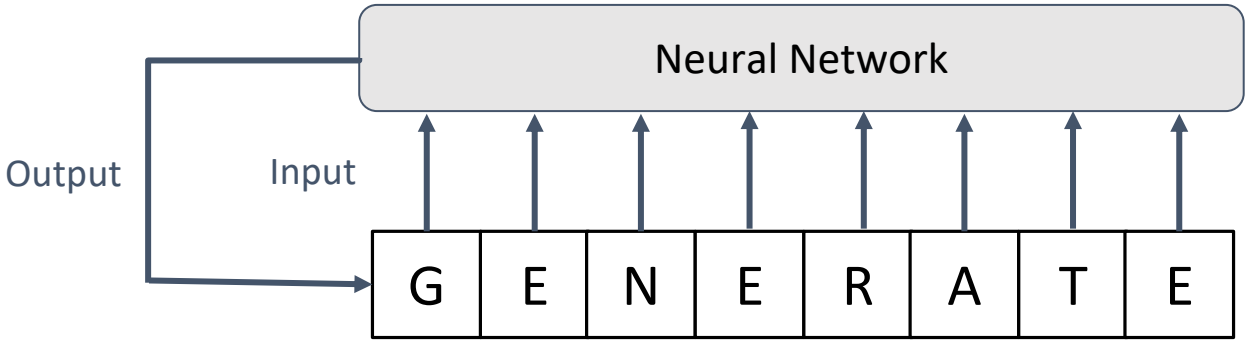
Pre-training Generative Models: $X_1, \ldots, X_n \sim P$

Task: Obtain more samples approximately from P

- Internet data for LLMs
- LAION 5B (images with captions) for Diffusion Models
- Might not be pleasing to the human eye, might contain wrong answers etc.

How do Autoregressive Models Behave?

Generate joint distributions coordinate by coordinate (or token by token)



How do Autoregressive Models Behave?

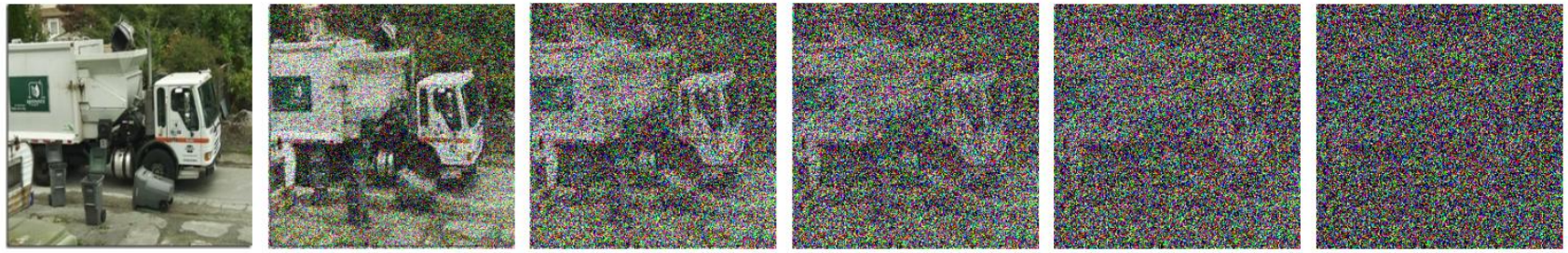
4	3	5	2	6	9	7	8	1
6	8	2	5	7	1	4	9	3
1	9	7	8	3	4	5	6	2
8	2	6	1	9	5	3	4	7
3	7	4	6	8	2	9	1	5
9	5	1	7	4	3	6	2	8
5	1	9	3	2	6	8	7	4
2	4	8	9	5	7	1	3	6
7	6	3	4	1	8	2	5	9

How AR models solve it:

4	3	5	2	6	9	7	8	1
6	8	2	5	7	1		9	
1	9				4	5		
8	2		1				4	
		4	6		2	9		
	5				3		2	8
		9	3				7	4
	4			5			3	6
7		3		1	8			

- Is this how humans naturally solve it?
- Is this the `best' way?
- Should we move beyond autoregression?

Continuous Diffusion Models



$$dX_t = -X_t dt + \sqrt{2} dB_t$$

destructive process

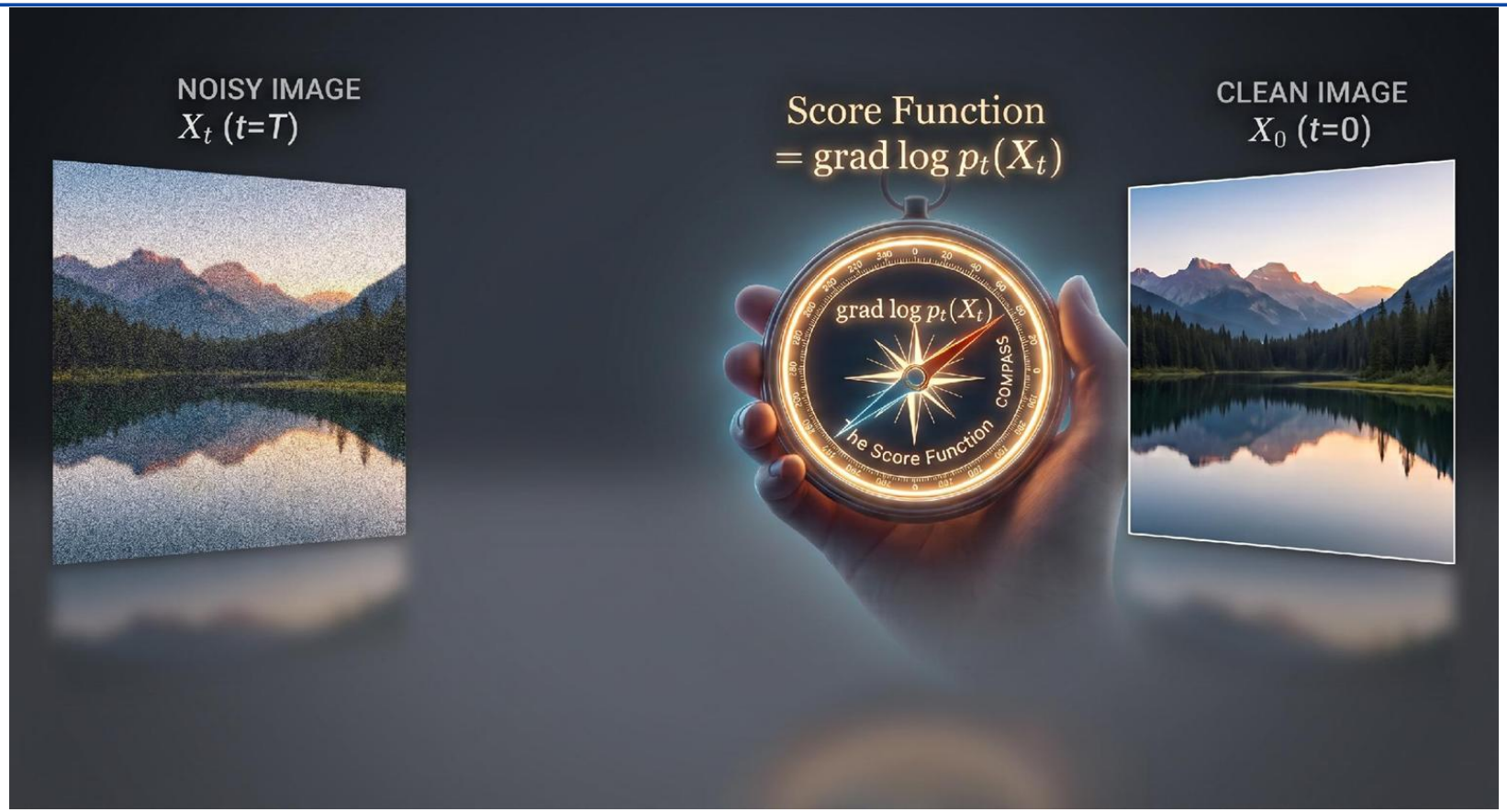
constructive process

$$dX_t = X_t dt + 2\nabla \log p_{T-t}(X_t) dt + \sqrt{2} dB_t$$

Score function

- Learn the score function via Tweedie's formula
- Sample new images from pure noise!

Continuous Diffusion Models



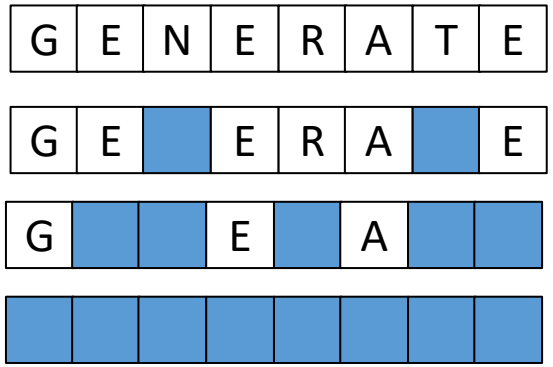
Score function is a compass which tells you how to go from a noisy image to a clean image.

Discrete Diffusion Models (Masked)

- Continuous Noising Processes ←→ Continuous Diffusion Models
- Discrete Noising Processes ←→ Discrete Diffusion Models

First work: D3PM by Austin et al, lots of variants and follow up work

Example 1: Masked Diffusion, how to noise?



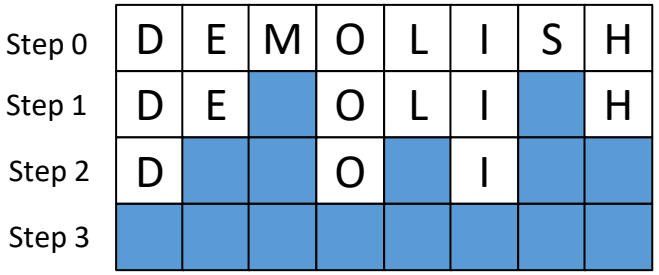
Step 1: Flip each to mask with probability p.

Step 2: Flip each to mask with probability p.

Step 3: Flip each to mask with probability p.

Discrete Diffusion Models: Masked

Noising:



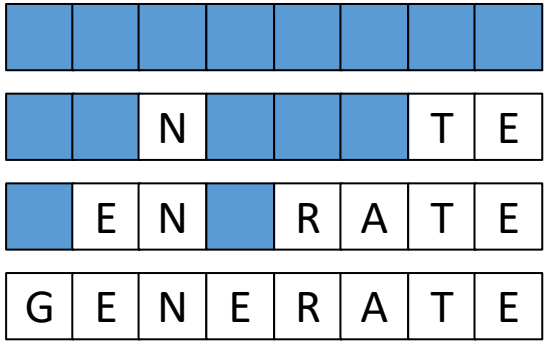
Denoising:

Step 3: Start with Full Noise

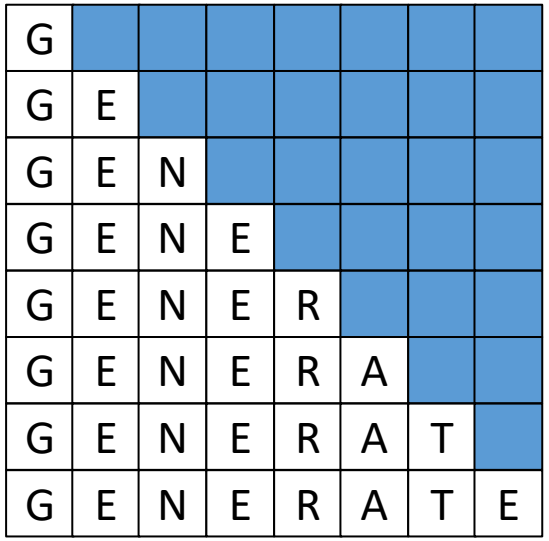
Step 2: Predict original, apply Bayes rule

Step 1: Predict original, apply Bayes rule

Step 0: Predict original, apply Bayes rule



Autoregressive Models



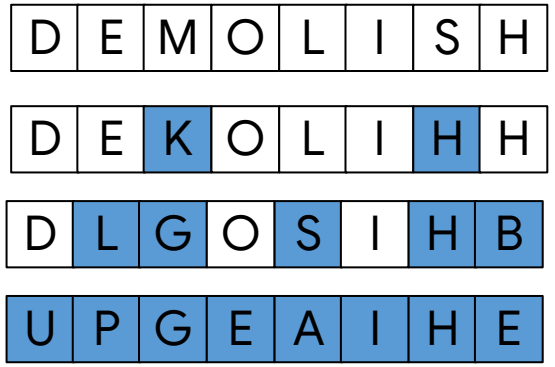
Discrete Diffusion Models (Multinomial)

Example 2: Multinomial Noise

Step 1: Flip to independently sampled with probability p

Step 2: Flip to independently sampled with probability p

Step 3: Flip to independently sampled with probability p



(Grey spaces are not real, for illustration purposes only)

Discrete Diffusion Models: Multinomial

Denoising:

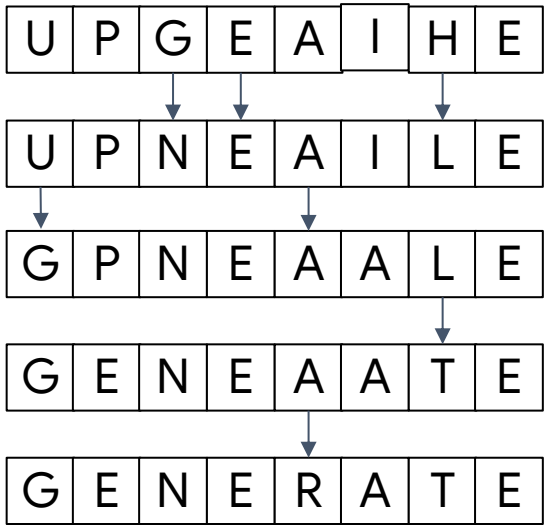
Step 4: Start with Full Noise

Step 3: Predict original, apply Bayes rule

Step 2: Predict original, apply Bayes rule

Step 1: Predict original, apply Bayes rule

Step 0: Predict original, apply Bayes rule



Multiple flips at the same token possible, allows backtracking!

That is, have rough drafts, and refine by correcting the mistakes.

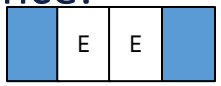
Comparison of Some Key Points

	Autoregressive	Masked Diffusion	Multinomial Diffusion
Generation Order	Causal (left to right)	Arbitrary	Arbitrary
Noise vs Signal	Known	Known	Unknown
Tokens per call	1	1 or Many	1 or Many
Backtracking	No	No	Yes
Inpainting	No	Yes	Yes

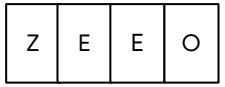
(Algorithmic changes can allow more features for autoregressive or masked diffusion models.)

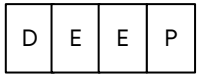
How to Learn Discrete Diffusion?

Most proposals for learning hinge on predicting the original un-noised sequence given a noisy sequence:

NN(

, step 2) predict 

NN(

, step 2) predicts 

- Austin et al (2021): Variational Lower Bound + Cross Entropy.
- Meng et al (2022): Ratio Matching
- Lou et al (2023): SEDD
- Varma et al (2024): Binary classification

Generative Modeling and RL Alignment

Post-Training

- Instruction tuning: Given a question X, generate a correct answer Y|X
- Refuse to answer harmful questions.
- Increase the factuality of answers.
- Make the generated images pleasing to the eye.
- Improve the aesthetic quality and readability of the generated text.

Given Question X and Answer Y, we obtain a reward model r(Y|X) for each task.

Task: Tune the pre-trained model to generate answers with high reward value

Entropy Regularized Reward and RL

Given Question Q and Answer A, we obtain a reward model r(Y|X) for each task.

Task: Tune the pre-trained model to generate answers with high reward value

$P_0(Y|X)$ = likelihood of answer Y given question X wrt Reference model (pre-trained)

$P_\theta(Y|X)$ = likelihood of answer Y given question X wrt Policy model.

Entropy Regularized Reward Maximization

$$\arg\sup_\theta \mathbb{E}_{X,Y\sim P_\theta(\cdot|X)} r(Y|X) - \beta \mathsf{KL}(P_\theta(\cdot|X)||P_0(\cdot|X))$$

Entropy Regularized Reward and RL

Given Question Q and Answer A, we obtain a reward model r(Y|X) for each task.

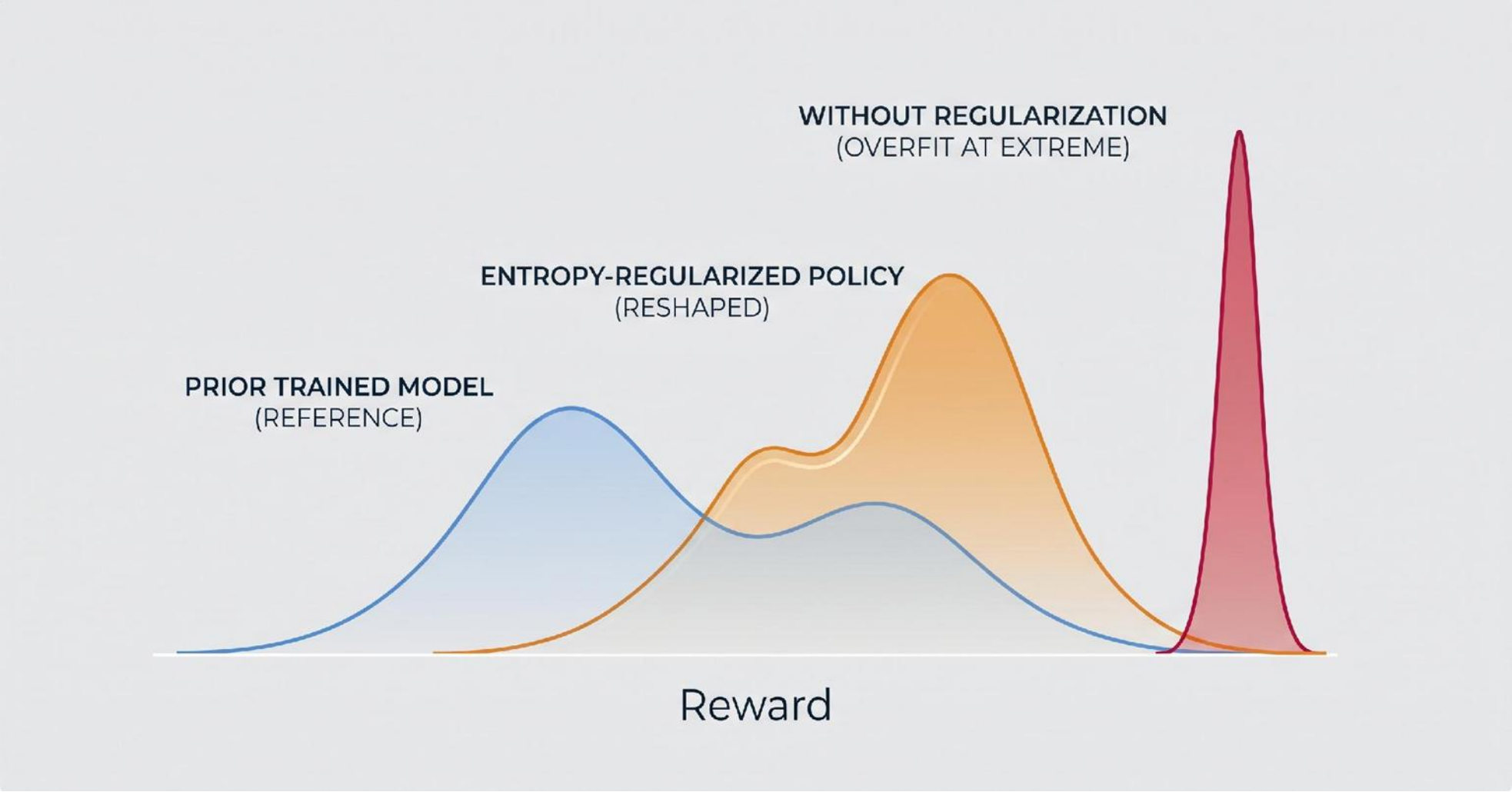
$P_0(Y|X)$ = likelihood of answer Y given question X wrt Reference model (pre-trained)

$P_\theta(Y|X)$ = likelihood of answer Y given question X wrt Policy model.

Entropy Regularized Reward Maximization

$$\arg\sup_\theta \mathbb{E}_{X,Y\sim P_\theta(\cdot|X)} \underbrace{r(Y|X)}_{\text{Maximize reward}} - \underbrace{\beta \mathsf{KL}(P_\theta(\cdot|X)||P_0(\cdot|X))}_{\text{Stay close to prior trained model}}$$

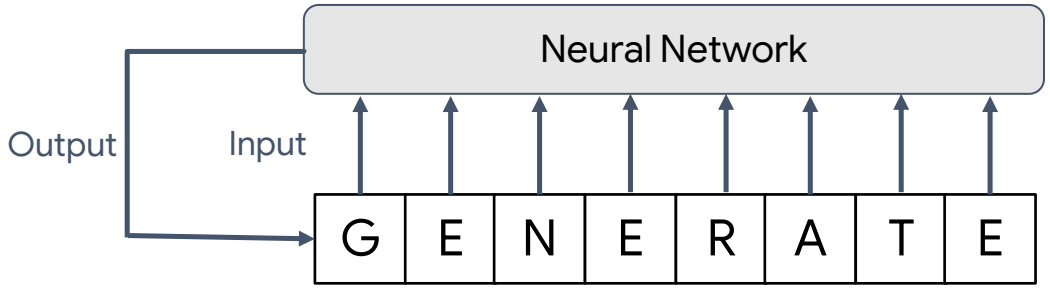
Entropy Regularized Reward and RL



Entropy Regularized Reward and RL

$$\arg\sup_{\theta} \mathbb{E}_{X,Y\sim P_\theta(\cdot|X)} r(Y|X) - \beta \mathsf{KL}(P_\theta(\cdot|X)||P_0(\cdot|X))$$

The case of autoregressive models:



$$P_\theta(Y_1,\ldots,Y_n|X) = \underbrace{P_\theta(Y_1|X)P_\theta(Y_2|Y_1,X)\ldots P_\theta(Y_n|Y_{<n},X)}$$

Known to the model

Entropy Regularized Reward and RL

$$\arg\sup_{\theta} \mathbb{E}_{X,Y\sim P_\theta(\cdot|X)} r(Y|X) - \beta \mathsf{KL}(P_\theta(\cdot|X)||P_0(\cdot|X))$$

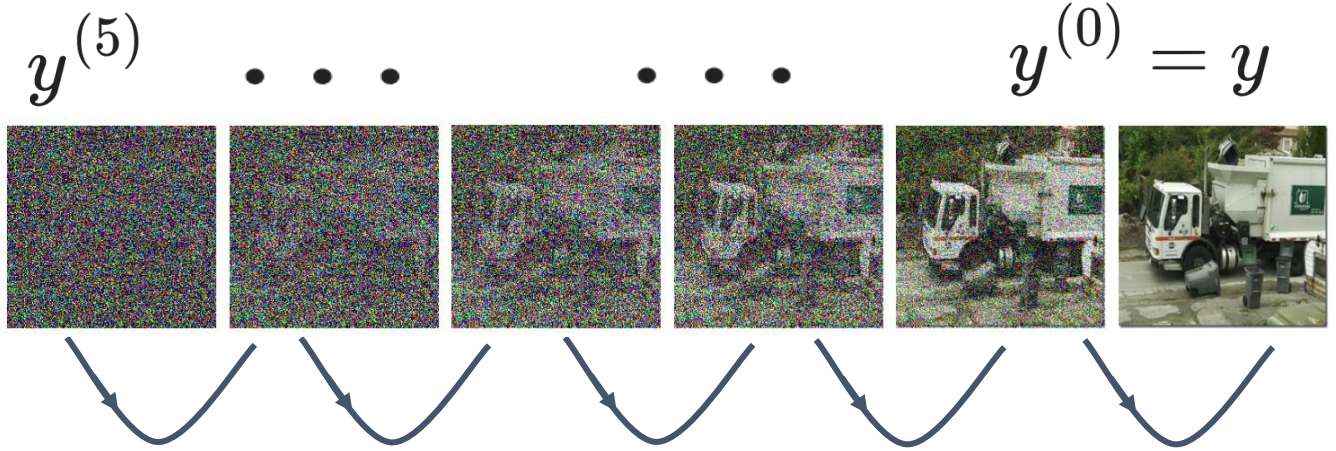
The case of autoregressive models:

$$P_\theta(Y_1,\ldots,Y_n|X) = P_\theta(Y_1|X)P_\theta(Y_2|Y_1,X)\ldots P_\theta(Y_n|Y_{<n},X)$$

Use your favorite Policy based RL algorithm (REINFORCE, PPO, Actor-Critic, GRPO etc.)

The Case of Diffusion Models

x = garbage truck

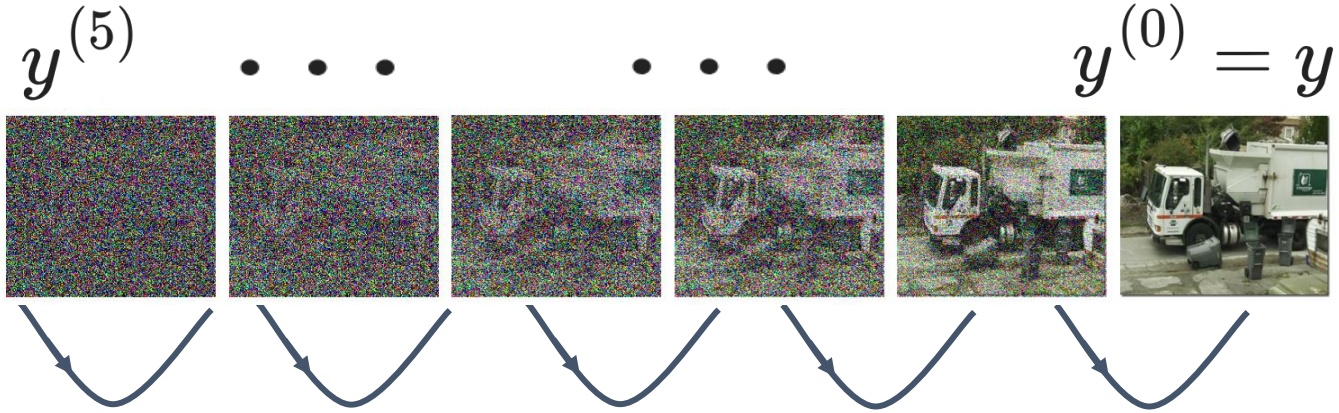


We only have access to conditional likelihoods $P_\theta(y^{(n-1)}|y^{(n)}, x)$

Cannot compute marginal likelihood $P_\theta(y|x)$ required for KL regularization

The Case of Diffusion Models: Relaxations

x = garbage truck



Relaxation: Use Trajectory KL instead of Marginal KL (DPOK)

$$\mathrm{KL}(P_\theta(y^{(0)}, \ldots, y^{(5)}|x)||P_0(y^{(0)}, \ldots, y^{(5)}|x))$$

DPOK: Reinforcement Learning for Fine-tuning Text-to-Image Diffusion Models
Fan et al (NeurIPS 2023)

(*) Training Diffusion Models with Reinforcement Learning, arxiv:2305.13301

The Case of Diffusion Models: Relaxations

Relaxation: Use Trajectory KL instead of Marginal KL (DPOK)

$$\mathrm{KL}(P_\theta(y^{(0)}, \ldots, y^{(5)}|x)||P_0(y^{(0)}, \ldots, y^{(5)}|x))$$

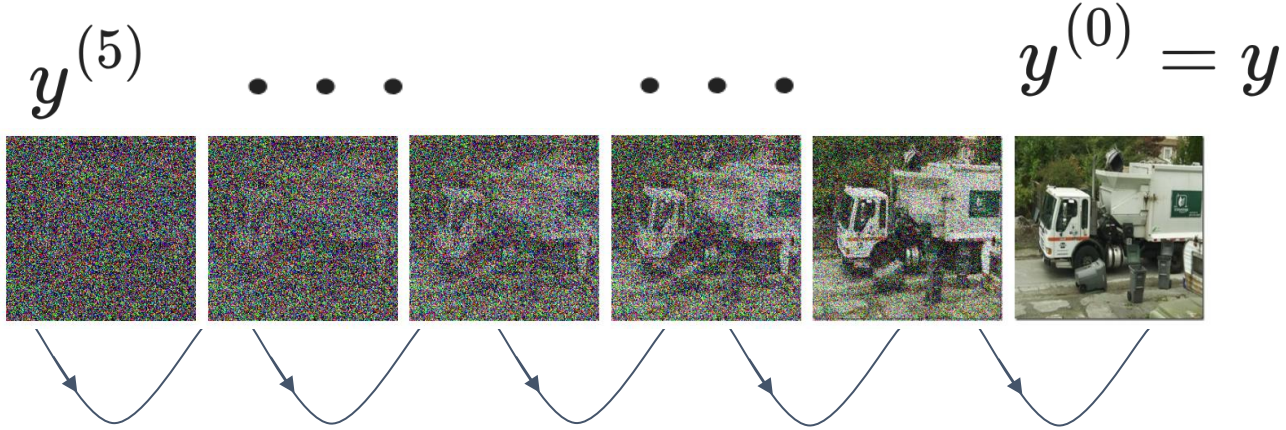
DPOK: Reinforcement Learning for Fine-tuning Text-to-Image Diffusion Models
Fan et al (NeurIPS 2023)

- This can lead to sub-par results as shown in (*). Can be too conservative.
- Initial value function bias problem due to learned trajectory.
  - This “changes” the reward function in ways hard to predict.

(*) Training Diffusion Models with Reinforcement Learning, arxiv:2305.13301

From Human Behavior to Specialized Agents

x = garbage truck



Relaxation 2: Remove KL regularization (DDPO)

Treat the denoising process as an MDP: $y^{(n)} \rightarrow y^{(n-1)} \rightarrow \cdots \rightarrow y^{(0)}$

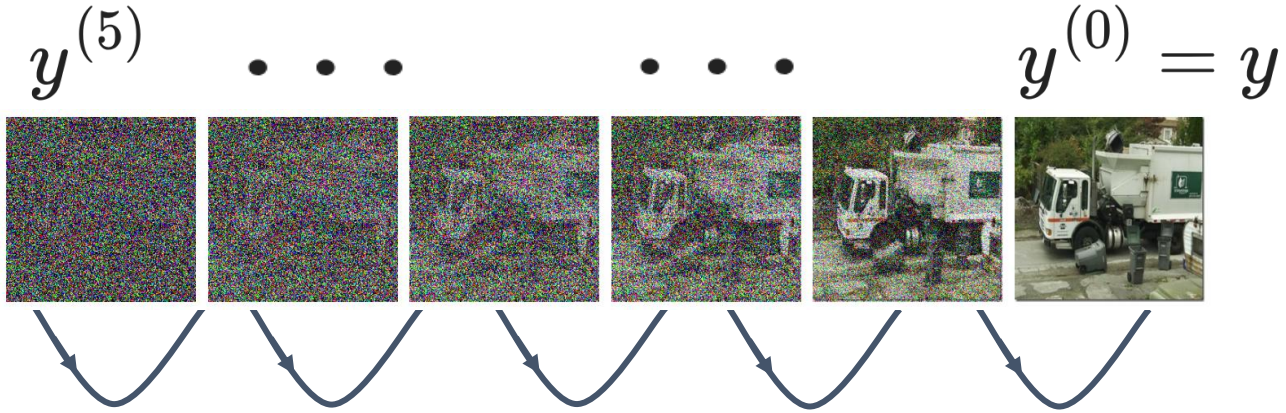
Action = denoising kernel

Maximize reward with your favorite RL algorithm.

Training Diffusion Models with Reinforcement Learning,
Black et al arxiv:2305.13301

From Human Behavior to Specialized Agents

x = garbage truck

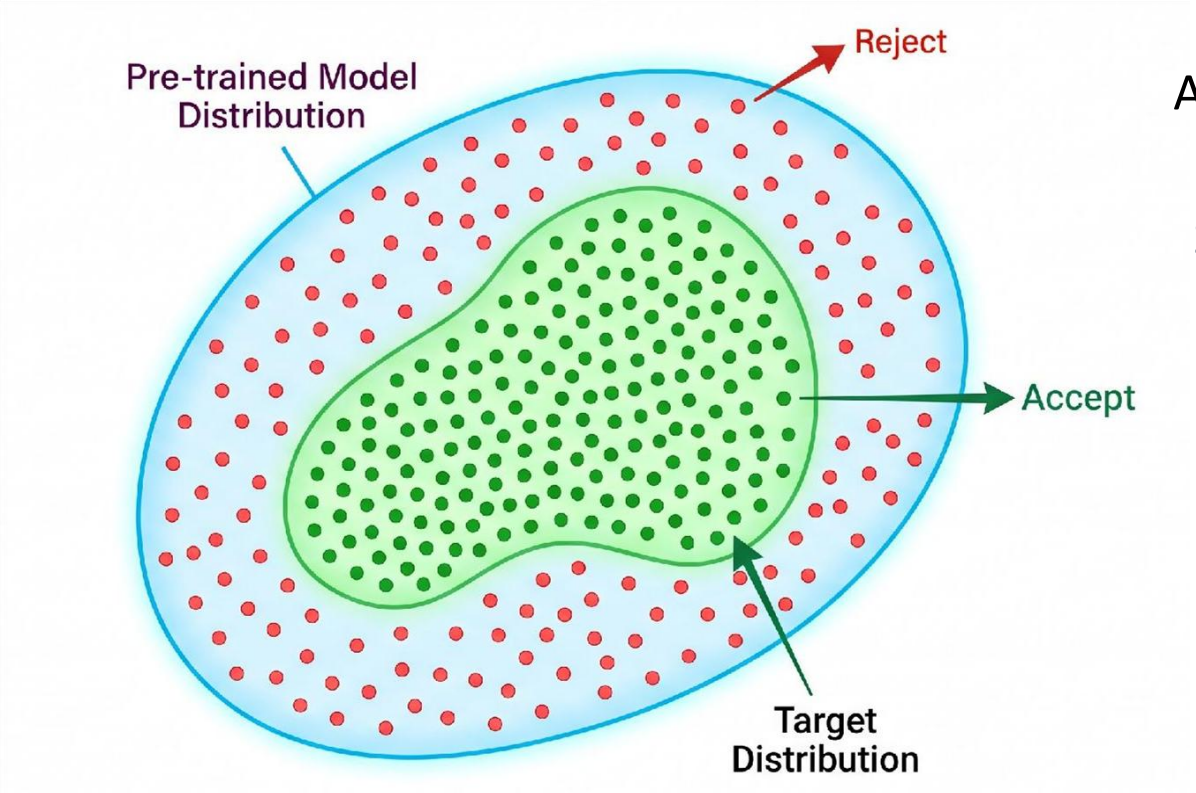


Relaxation 2: Remove KL regularization (DDPO)

Can cause mode collapse/ become unstable in large scale training

Training Diffusion Models with Reinforcement Learning,
Black et al arxiv:2305.13301

A Backdoor Via Rejection Sampling



An alternative to RL algorithms:

1. Create a large dataset of accepted samples.
2. Fine tune on these accepted samples (using the pretraining algorithm).

A Backdoor Via Rejection Sampling

$$\theta^* = \arg\sup_{\theta} \mathbb{E}_{X,Y\sim P_\theta(\cdot|X)} r(Y|X) - \beta \mathsf{KL}(P_\theta(\cdot|X)||P_0(\cdot|X))$$

$$P_{\theta^*}(Y|X) \propto \exp\left(\frac{r(Y|X)}{\beta}\right) P_0(Y|X)$$

$$Y|X, \mathsf{Accept} \sim P_{\theta^*}(\cdot|X)$$

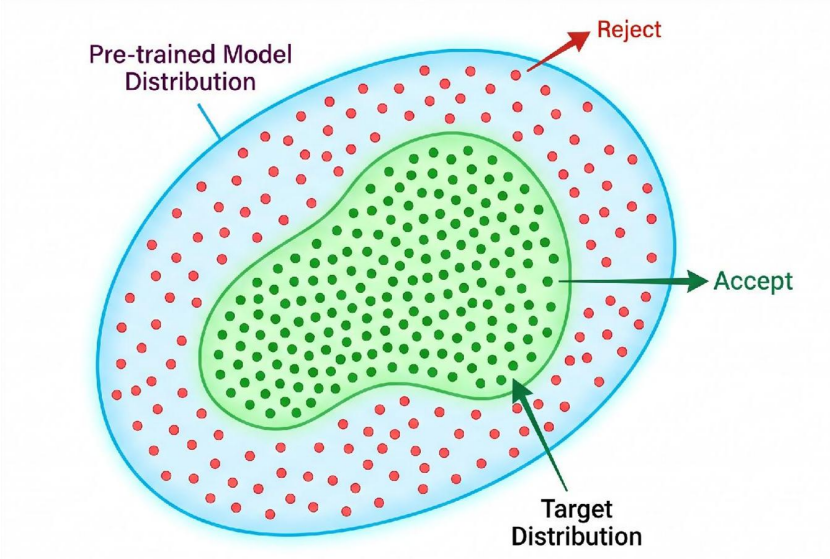
A Backdoor Via Rejection Sampling

Advantage: No need for marginal likelihood computation.

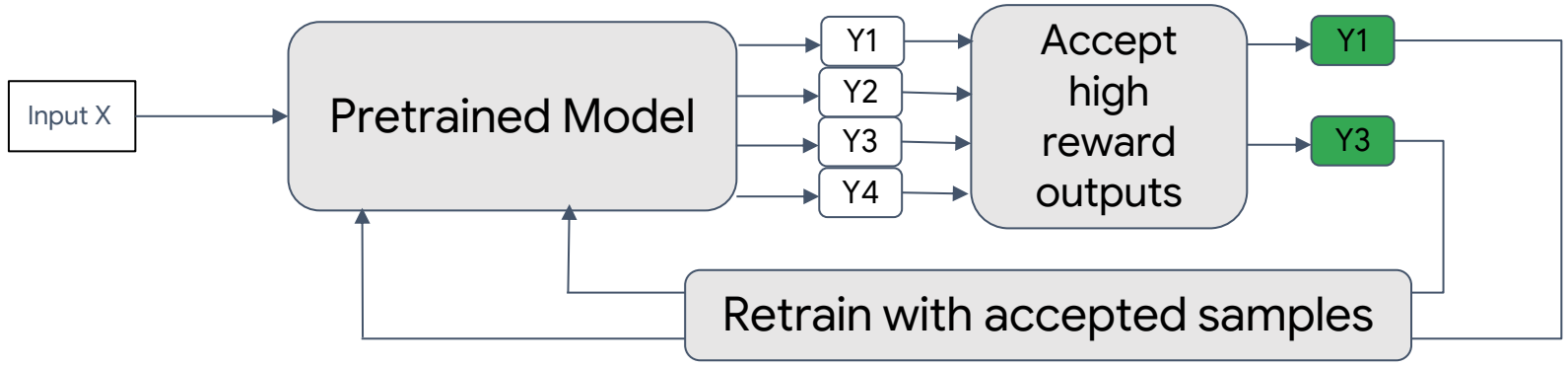
Drawback: Low acceptance rate can hinder efficiency.

Classical Rejection sampling: $Y \sim P_0(|X); \quad \mathbb{P}(\text{Accept}|Y) = \exp\left(\frac{r(Y|X) - r_{\max}}{\beta}\right)$

Efficient Proposal:
Generate $Y_1, \ldots, Y_N \sim P_0(|X)$. Calculate $r_i(Y_i|X)$. Accept top K out of N



Generalized Rejection Sampling Fine Tuning (GRAFT)

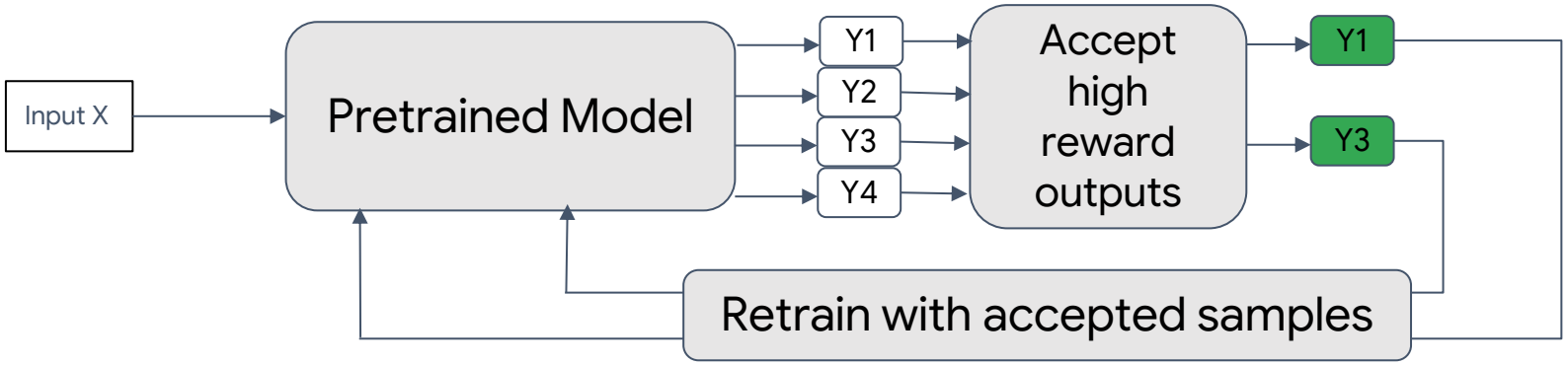


RAFT by Dong et al, arXiv:2304.06767

Rejection sampling based ideas also used in:
RSO by Liu et al, arXiv:2309.06657
Reinforce-Rej by Xiong et al, arXiv:2504.11343

Best-of-N Alignment:
Variational BoN by Amini et al, arXiv:2407.06057
BoNBoN by Gui et al, arXiv:2406.00832

Generalized Rejection Sampling Fine Tuning (GRAFT)



Mathematically, this solves KL regularized reward maximization with a reshaped reward!

Generalized Rejection Sampling Fine Tuning (GRAFT)

Solves KL regularized reward maximization with a reshaped reward.

Example: Preference reward

Generate Y_1, Y_2. accept Y_1 if $r_1 > r_2$. Else accept Y_2

$$\frac{\hat{r}(Y|X)}{\alpha} = F_{r|X}(r(Y|X))$$

Reshapes to the CDF function. Monotonically increasing.

That is, reward changes to the percentile of the reward. Example:

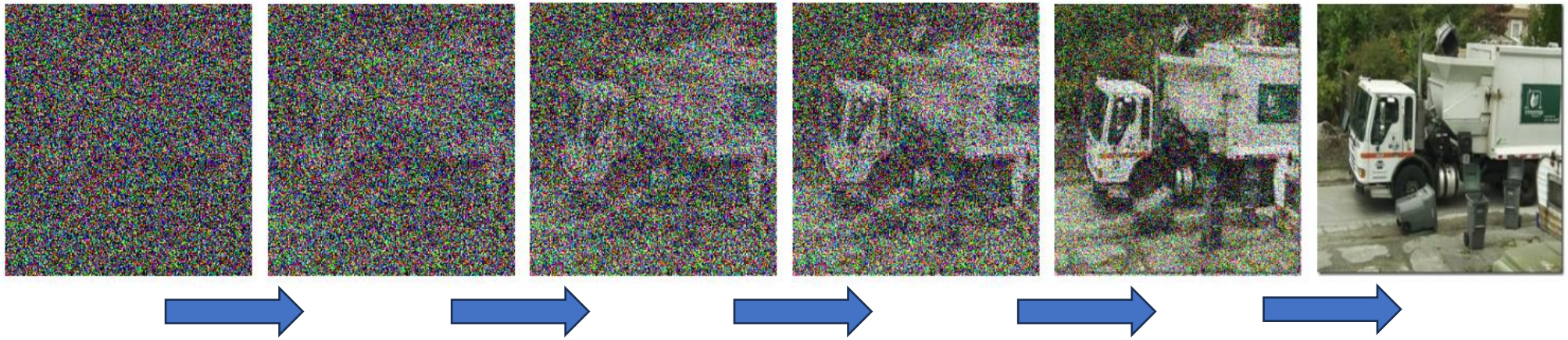
Reward	Percentile	Reshaped Reward
0.85	99	0.99
0.1	30	0.30

Partial GRAFT

- GRAFT applies to any generative modeling algorithm.
- Can we do better given the structure of diffusion models?

Prompt: Garbage truck

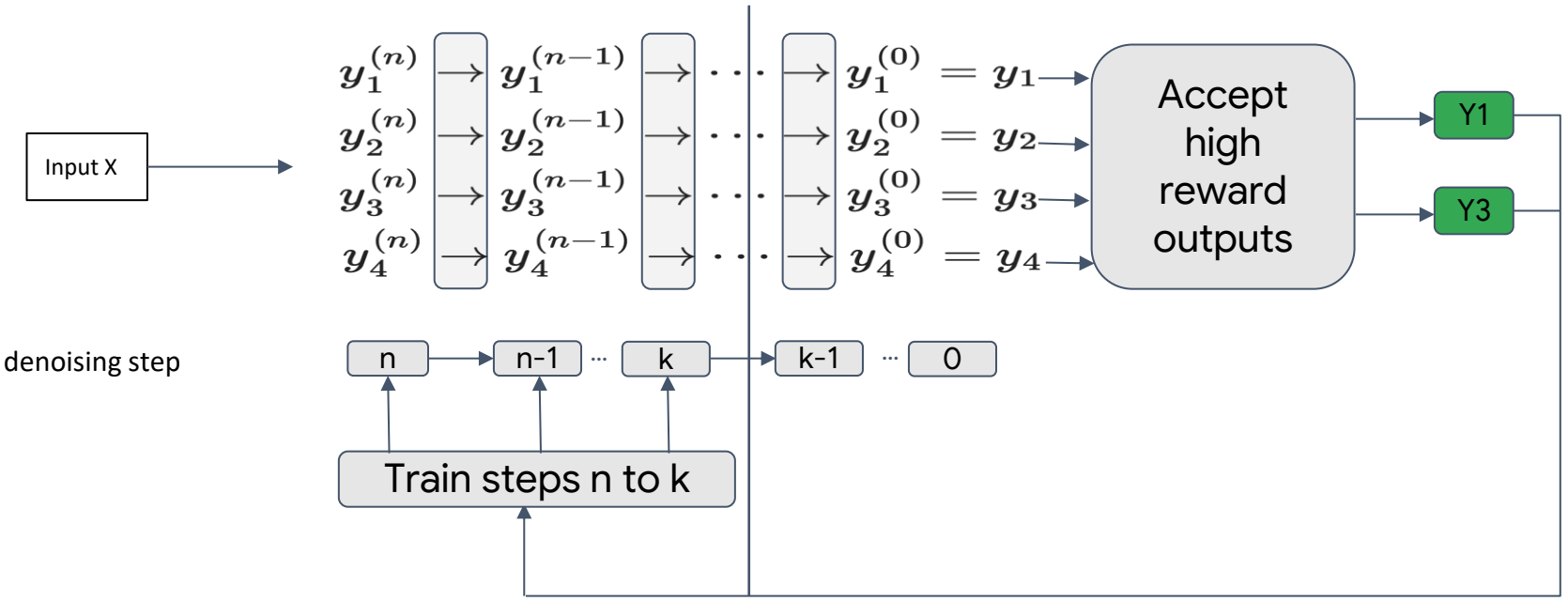
Reward: Text to Image alignment



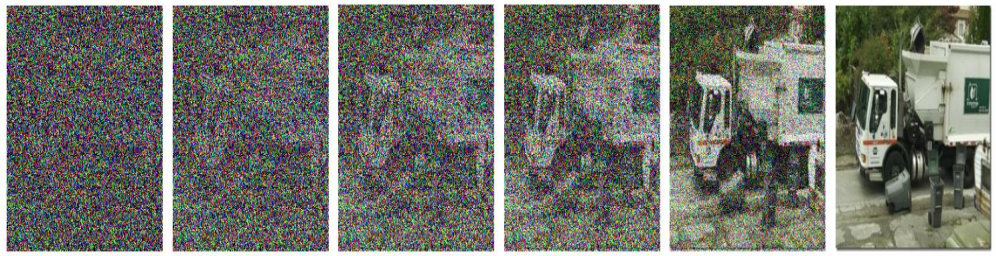
Reward already clear here

Idea: Train early transitions with high reward variance only!

Partial GRAFT



Partial GRAFT: Bias Variance Trade Off



Reward variance is 0 given final generation y

Reward has some variance when we observe only y^(3)

(From Markov property)

Easy to learn

Hard to learn

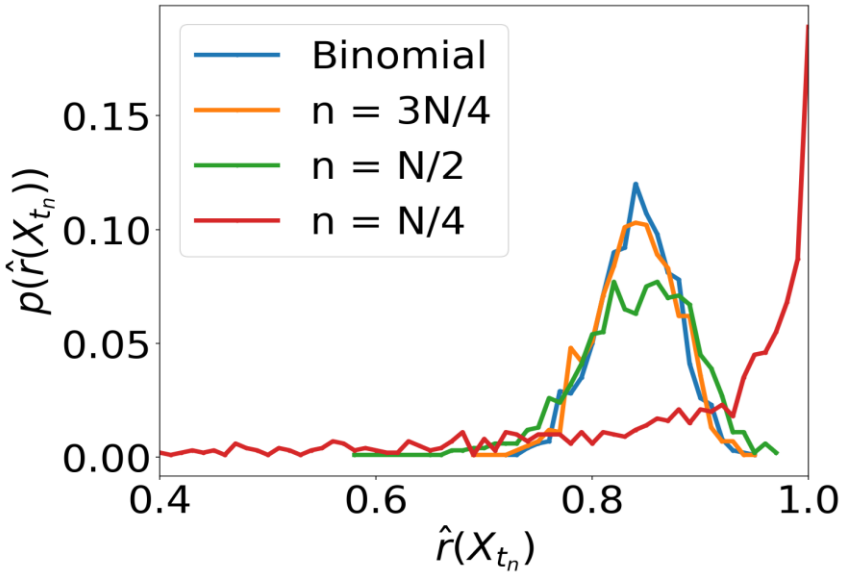
We give some theoretical evidence for this via Bakry Emery theory.

Partial GRAFT: Bias Variance Trade Off

Example from Molecule Generation Task. Reward = molecule stability

Table 1: Conditional variance.

n	$\mathbb{E}\left[\mathsf{Var}(r(X_0)\|X_{t_n})\right]$
N	0.1341
$3N/4$	0.1327
$N/2$	0.1312
$N/4$	0.0848



We pretty much know the reward by step N/4

Partial GRAFT: Results

Table 2: **Text-to-Image Generation fine-tuning on SDv2**: VQAScore (normalized to 100) reported on GenAI-Bench, T2ICompBench++ - Val (denoted as T2I - Val) and GenEval.

Model	Fine-Tuned on GenAI-Bench			Fine-Tuned on T2ICompBench++ - Train		
	GenAI	T2I - Val	GenEval	GenAI	T2I - Val	GenEval
SD v2	66.87±0.14	69.20±0.17	73.49±0.41	66.87±0.14	69.20±0.17	73.49±0.41
SDXL-Base	69.69±0.17	72.98±0.16	73.90±0.40	69.69±0.17	72.98±0.16	73.90±0.40
DDPO	65.70±0.17	68.03±0.16	72.13±0.37	64.65±0.17	69.05±0.15	69.60±0.37
GRAFT	70.51±0.15	75.69±0.13	79.85±0.31	70.97±0.14	75.88±0.13	79.57±0.30
P-GRAFT(0.75N)	69.46±0.15	74.51±0.14	79.44±0.33	69.51±0.15	74.30±0.13	78.50±0.33
P-GRAFT(0.5N)	71.00±0.14	75.45±0.14	80.60±0.31	70.73±0.14	75.37±0.12	79.25±0.30
P-GRAFT(0.25N)	**71.94**±0.14	**76.12**±0.13	**80.96**±0.29	**71.42**±0.14	**76.15**±0.13	**80.29**±0.30

Table 3: **Layout Generation**: Fine-tuning results for unconditional and category-conditional generation on PubLayNet.

Model	Unconditional		Class-conditional	
	Alignment	FID	Alignment	FID
Baseline	0.094	8.32	0.088	4.08
GRAFT	0.064	10.68	0.068	5.04
P-GRAFT(0.5N)	0.071	9.24	0.072	4.55
P-GRAFT(0.25N)	**0.053**	9.91	**0.064**	4.67

Table 4: **Molecule Generation**: Fine-tuning results on QM9. (Relative) number of sampling rounds required are also reported.

Model	Mol: Stability	Sampling Rounds
Baseline	90.50±0.15	-
GRAFT	90.76±0.20	9×
P-GRAFT(0.5N)	90.46±0.27	1×
P-GRAFT(0.25N)	**92.61**±0.13	1×

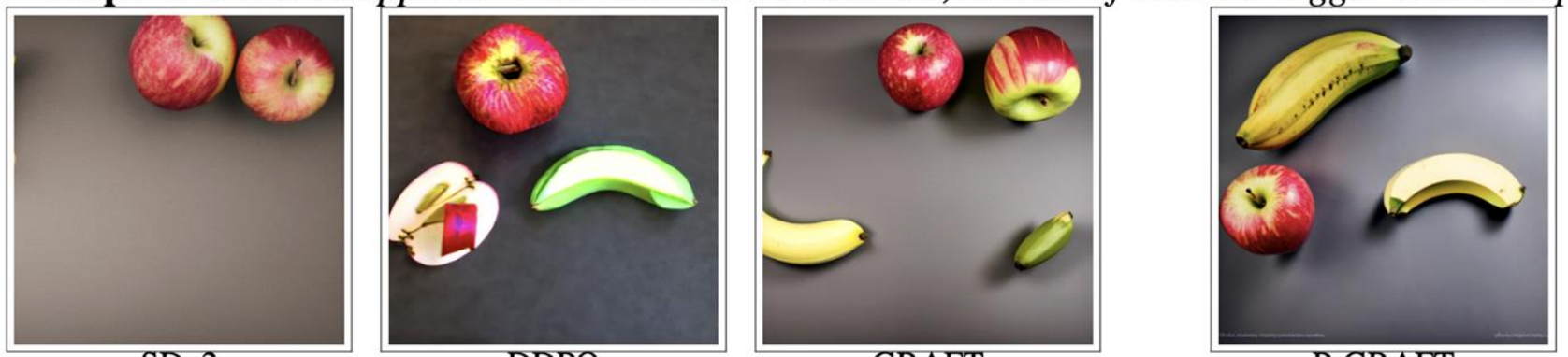
Prompt: *In the yoga room, all the mats are red.*

SDv2 DDPO GRAFT P-GRAFT

Prompt: *Three policemen working together to direct traffic at a busy intersection.*

SDv2 DDPO GRAFT P-GRAFT

Prompt: *There is an apple and two bananas on the table, neither of which is bigger than the apple.*



SDv2 DDPO GRAFT P-GRAFT

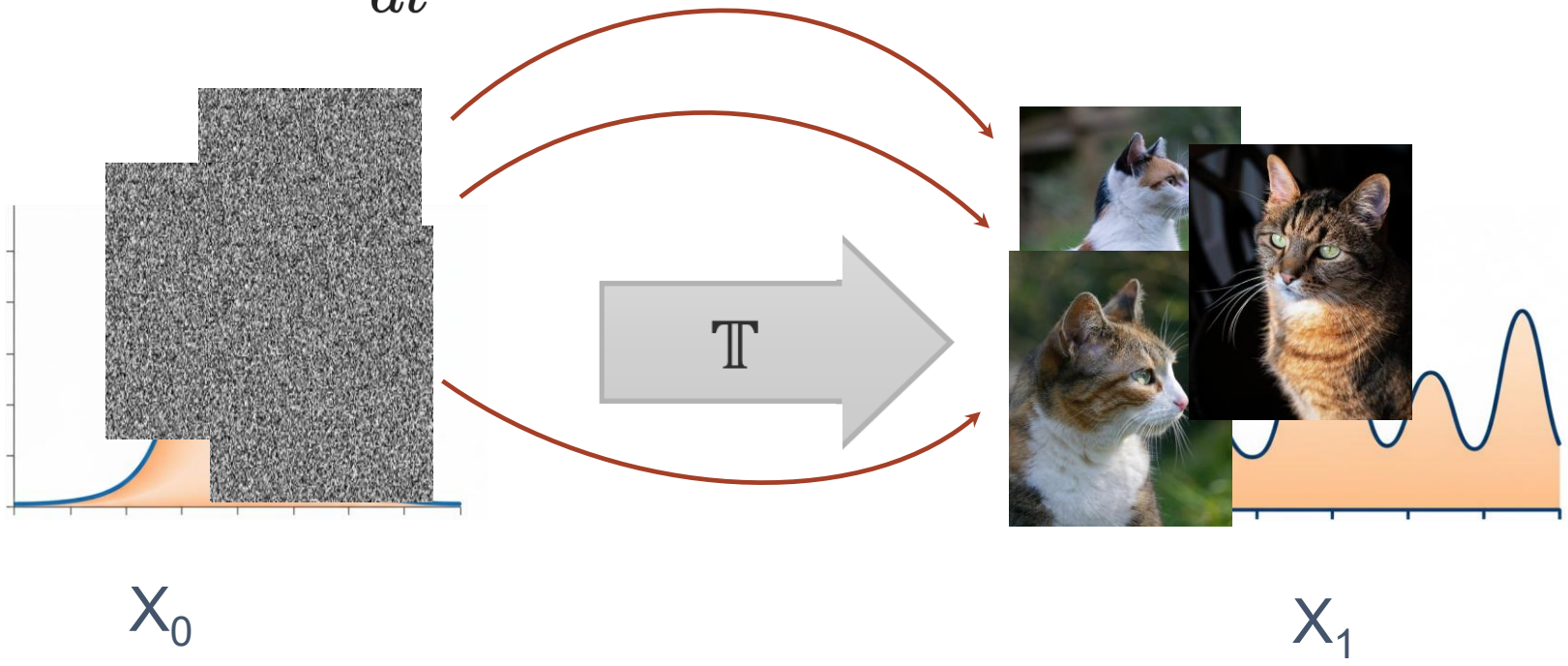
RL for Diffusion and Flow Models

Some recent works:

1. Flow-GRPO (Liu et al, 2025)
2. Diffusion NFT (Zheng et al, 2025)
3. Reinforce Adjoint Matching (Bermeister et al, 2026)
4. V-GRPO (Tang et al, 2026)
5. Qwen Image 2.0 tech report

Inverse Noise Correction: Improving Rectified Flows

$$\frac{dX_t}{dt} = v_\theta(t, X_t), \quad t \in [0, 1]$$

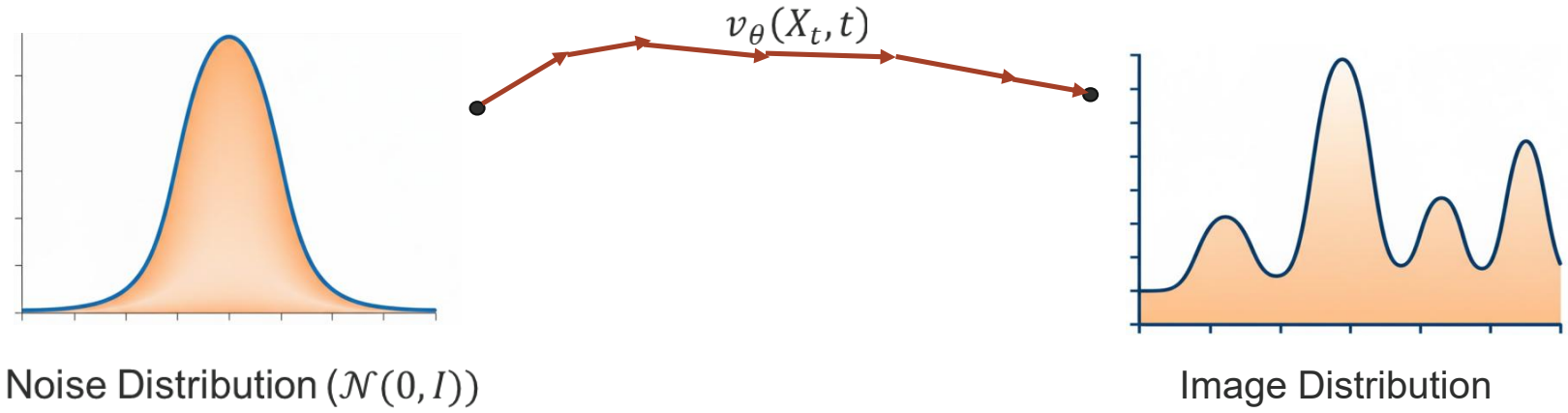


For flow models, there is no bias variance trade off since the output is a function of the the initial noise.

Inverse Noise Correction: Improving Rectified Flows

$$\frac{dX_t}{dt} = v_\theta(t, X_t), \quad t \in [0, 1]$$

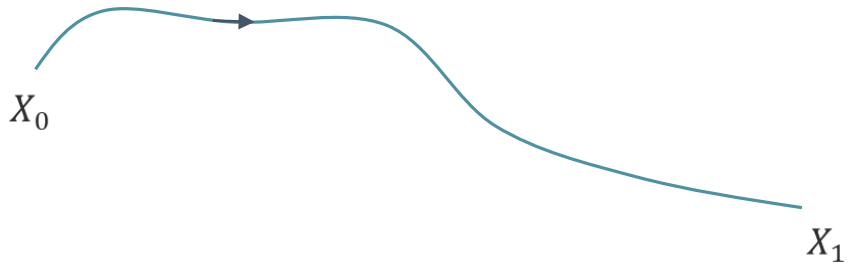
Sources of Error:
i) Learning Error (Statistical + Approximation + Optimization)
ii) Discretization Error



Can we correct these errors in a sample efficient way?

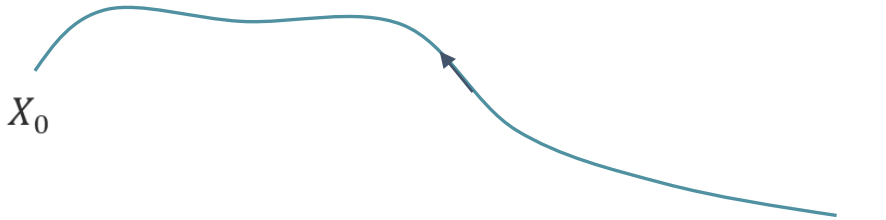
Inverse Noise Correction: Improving Rectified Flows

$$\frac{dX_t}{dt} = v(X_t, t)$$



Flow models are reversible

$$\frac{dX'_t}{dt} = -v(X'_t, 1-t)$$



$$\mathbb{T}(X_0) = X_1 \Leftrightarrow X_0 = \mathbb{T}^{-1}(X_1)$$

Reversible even with discretization

Inverse Noise Correction: Improving Rectified Flows

Reversible even with discretization

$$X_{k+1} = X_k + \eta v_\theta(X_k, \eta k)$$

$$\Rightarrow X_k = X_{k+1} - \eta v_\theta(X_k, \eta k) \Leftrightarrow X_k = F_k(X_k)$$

Banach Fixed Point Theorem:
Suppose $F_k(\cdot)$ is a contraction mapping: $||F_k(x) - F_k(y)|| \leq \gamma||x - y||$.
Then, $F_k(x) = x$ has a unique fixed point, X_k. Furthermore, for any $x_0 \in \mathbb{R}^d$, $x_{l+1} = F_k(x_l)$ converges exponentially fast to X_k.

Theorem: Suppose $v_\theta(x, t)$ is L-Lipschitz w.r.t. x. Then, for $\eta < 1/L$, $F_k(\cdot)$ is a contraction mapping with $\gamma = \eta L$.

Proof:

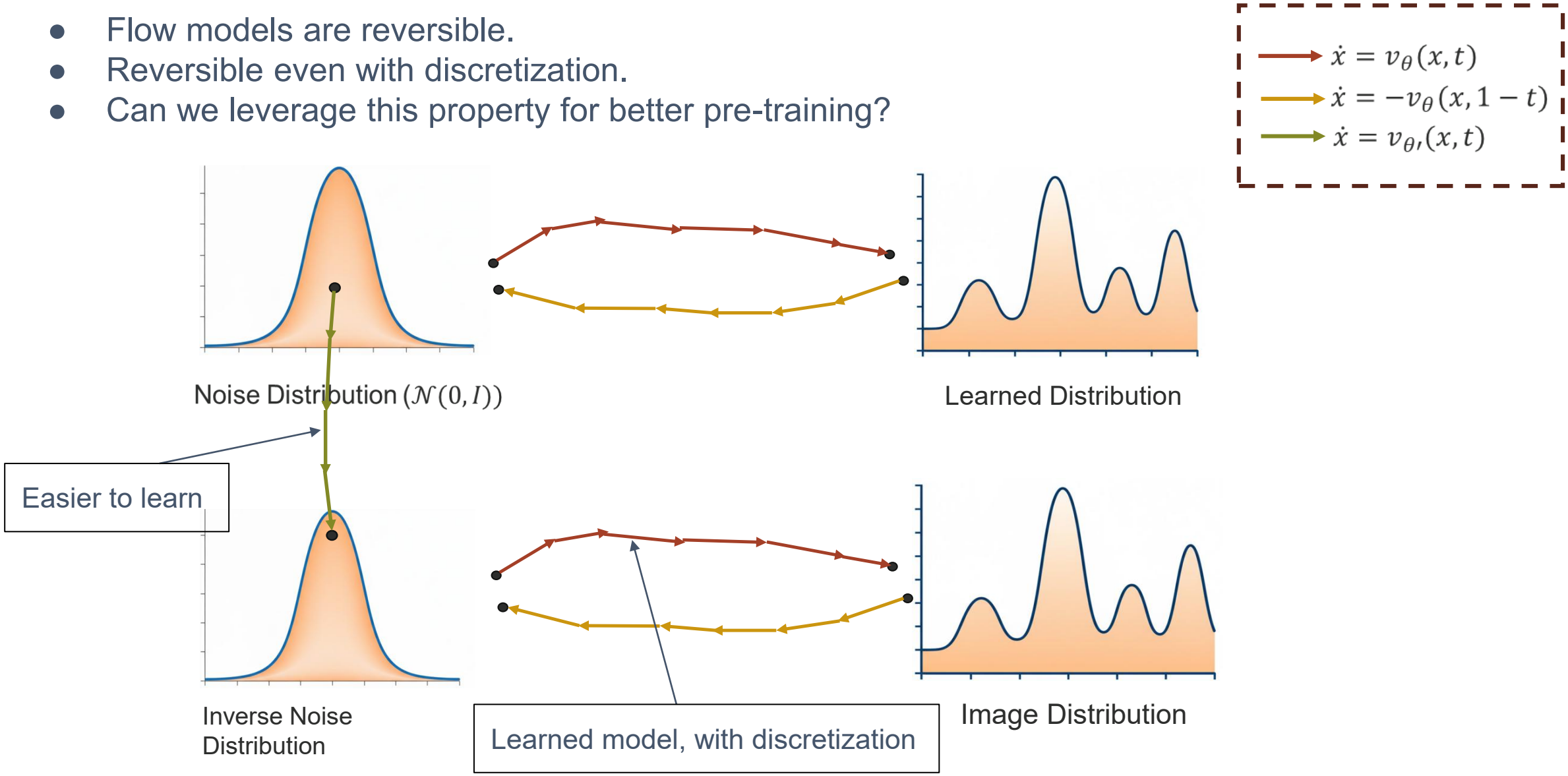
$$F_k(x) - F_k(y) = X_{k+1} - \eta v_\theta(x, \eta k) - (X_{k+1} - \eta v_\theta(y, \eta k))$$

$$\Rightarrow ||F_k(x) - F_k(y)|| = \eta||v_\theta(x, \eta k) - \eta v_\theta(y, \eta k))||$$

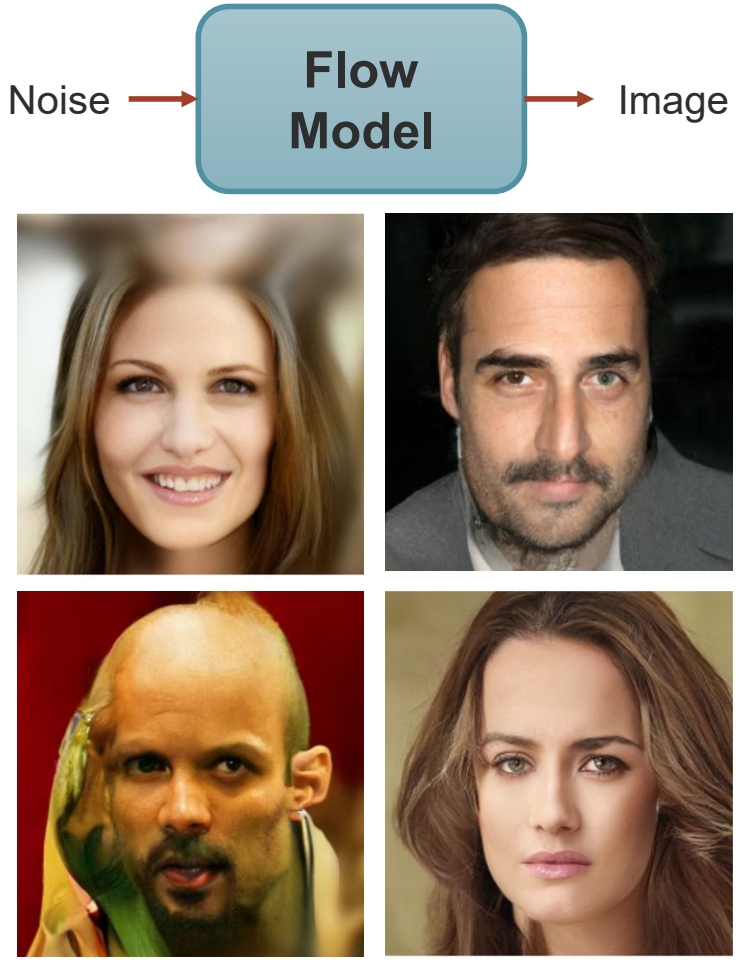
$$\leq \eta L||x - y||.$$

Inverse Noise Correction: Improving Rectified Flows

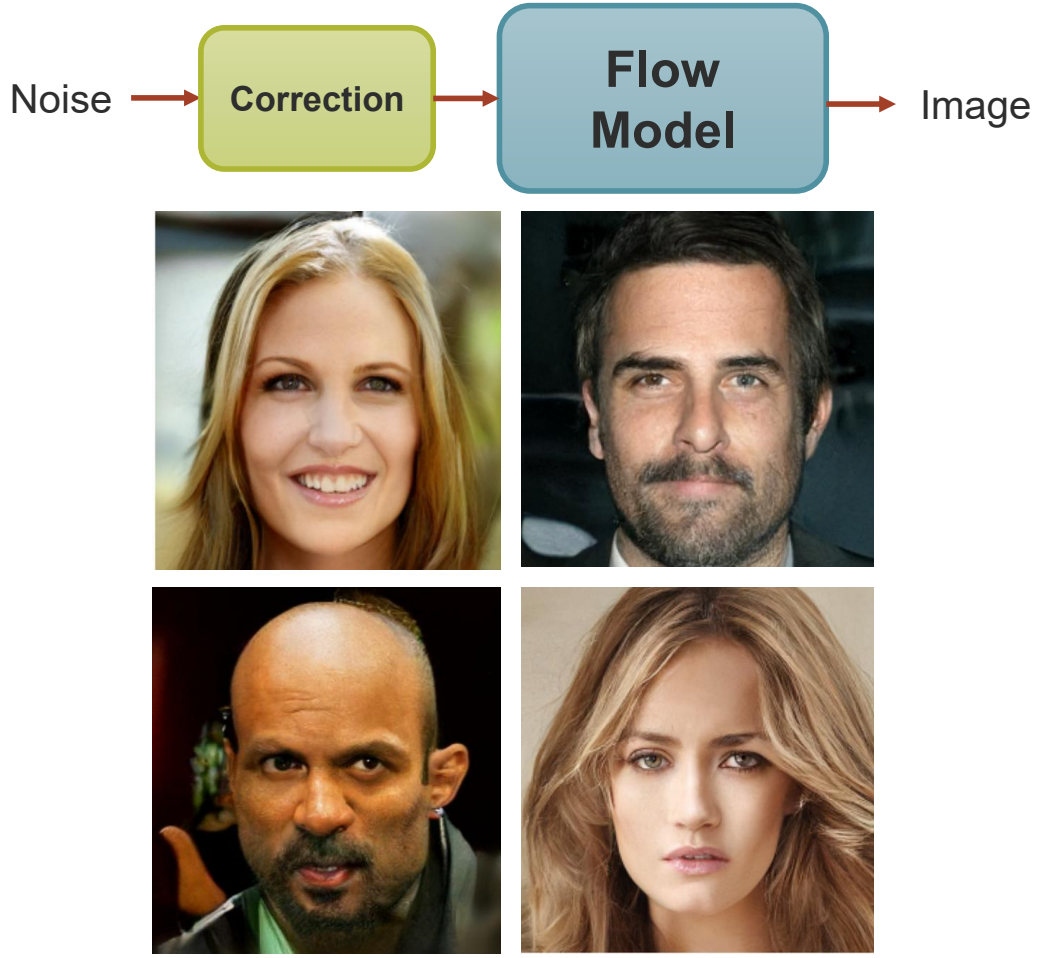
- Flow models are reversible.
- Reversible even with discretization.
- Can we leverage this property for better pre-training?



Inverse Noise Correction: Improving Rectified Flows



Rectified Flow



Inverse Noise Correction

Results

Table 5: **Image Generation**: Results for inverse noise correction on CelebA-HQ and LSUN-Church. The noise corrector samples the inverse noise starting from $\mathcal{N}(0, \mathbf{I})$ for 'Sampling Steps', and the pre-trained model samples the image starting from the inverse noise.

Sampling Steps		FID		FLOPs/image ($\times 10^{12}$)
Noise Corrector (16M parameters)	Pre-Trained Model (65M parameters)	CelebA-HQ (256×256)	LSUN-Church (256×256)	
-	1000	11.93	8.40	6.869
-	200	13.39	8.63	1.374
100	100	8.94	7.90	0.903
200	200	**8.02**	**7.26**	1.806

Questions

Thank You