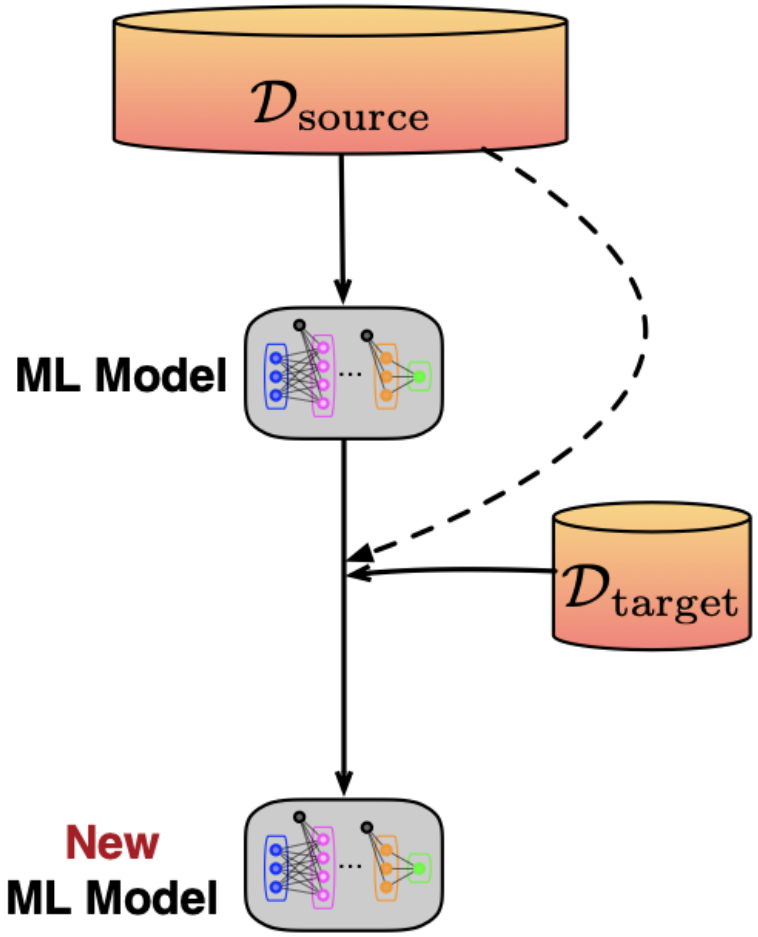
Transfer Learning for Diffusion Models

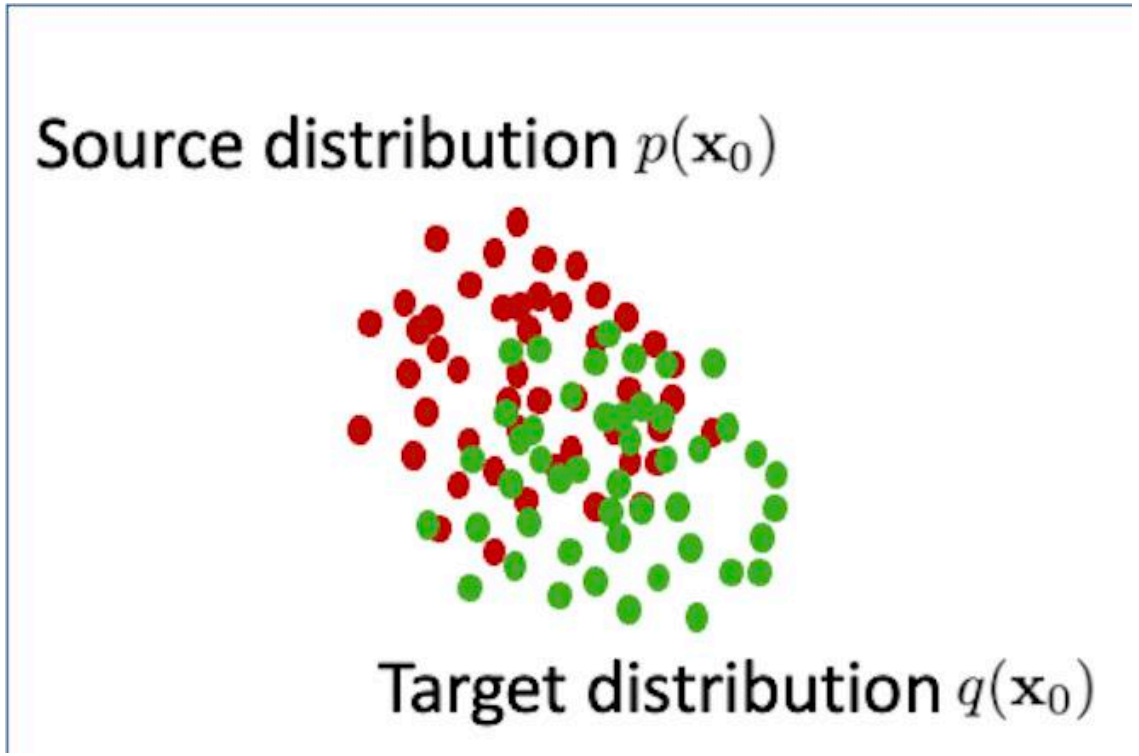
Liyan Xie

Department of Industrial and Systems Engineering

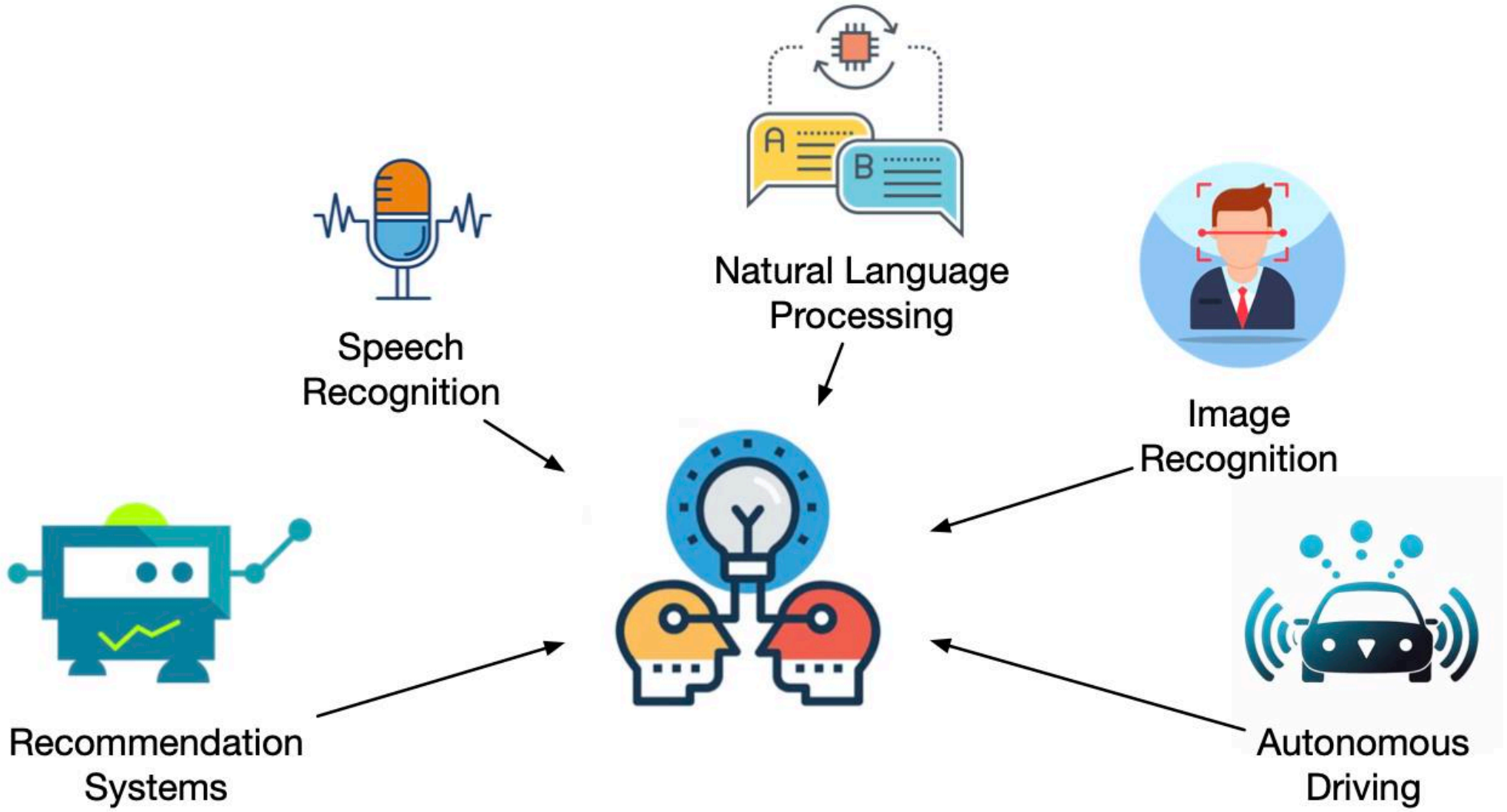
University of Minneasota

Transfer learning

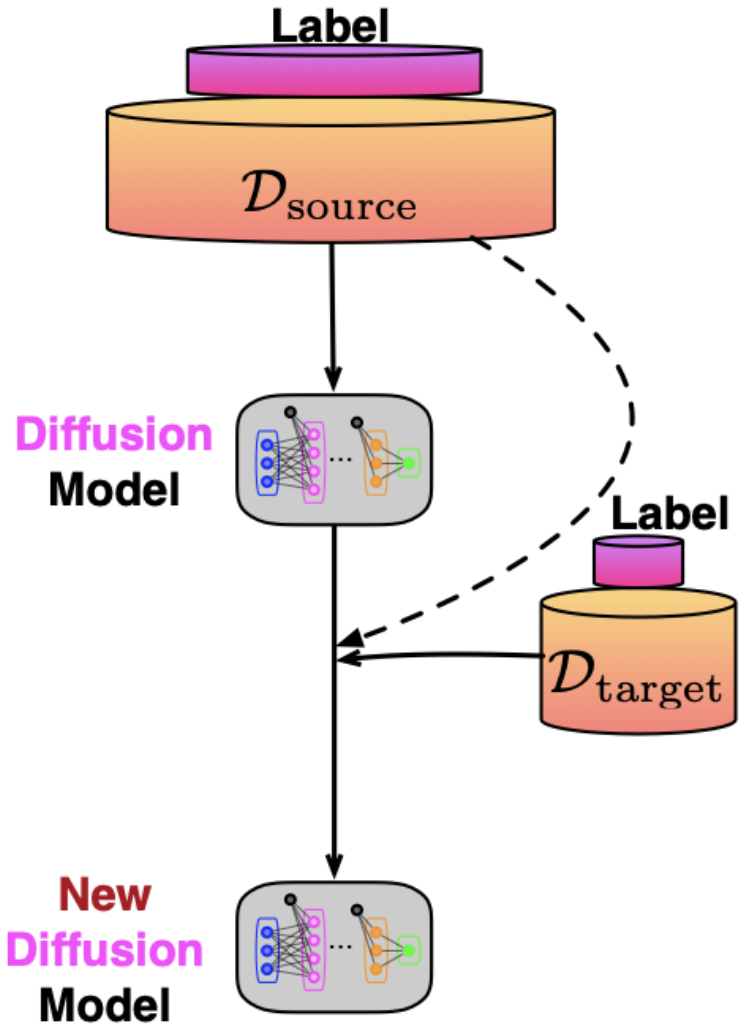




Applications



Today's talk



- A specific task: learning a Diffusion generative model
- Introduce a finetuning-free framework for transferring a pre-trained diffusion model from the source domain to the target domain
- Extend to a conditional version for modeling the joint distribution of data and its corresponding labels, which are important for practical applications and downstream tasks.

Ouyang Yidong, Liyan Xie, Hongyuan Zha, Guang Cheng. "Transfer Learning for Diffusion Models". *NeurIPS 2024.*

Outline

- A brief introduction of diffusion models
- Proposed algorithm: density-ratio inspired guidance
- Improved techniques: two additional regularization terms
- Numerical experiments

Emergence of Deep Generative AI

Billions even trillions of parameters

DeepSeek

DALL-E 3

ChatGPT

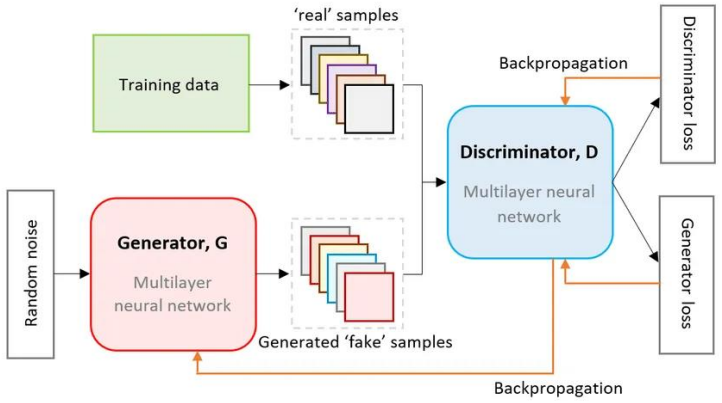
Stable Diffusion

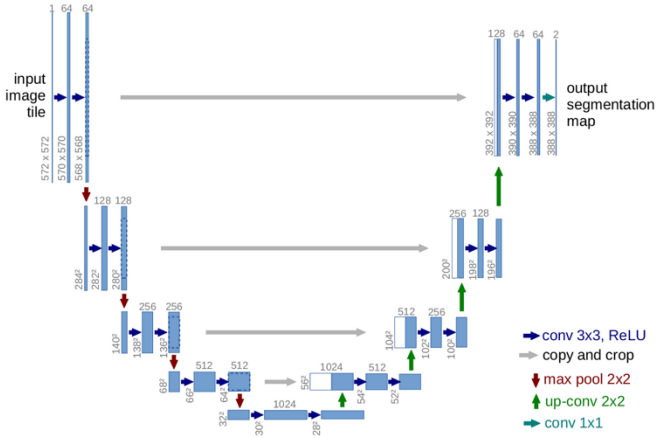
Model

VAE

Transformer

GAN





Deep Generative AI

1998 2012 2015 2018 2021 2022 2023

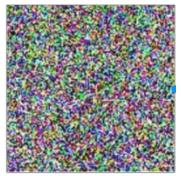
-- Thanks to blogs by Rockwell Anyoha, Toloka Team and Rick Merritt

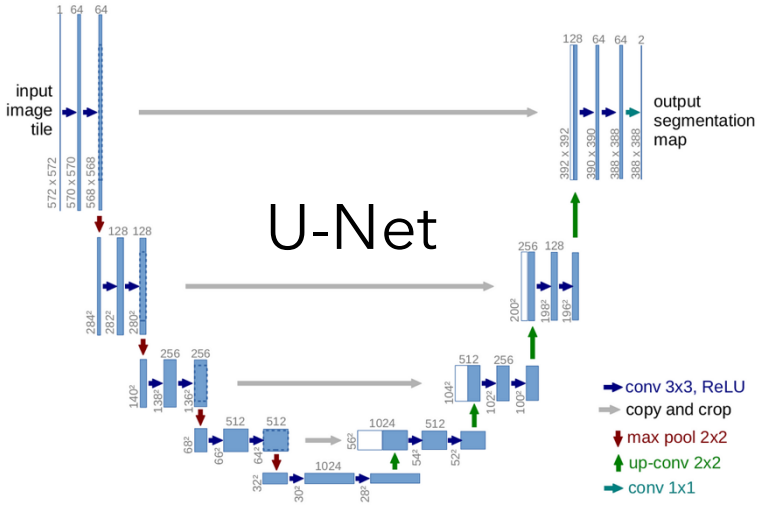
Diffusion Model

- Sequential transformation

Noise

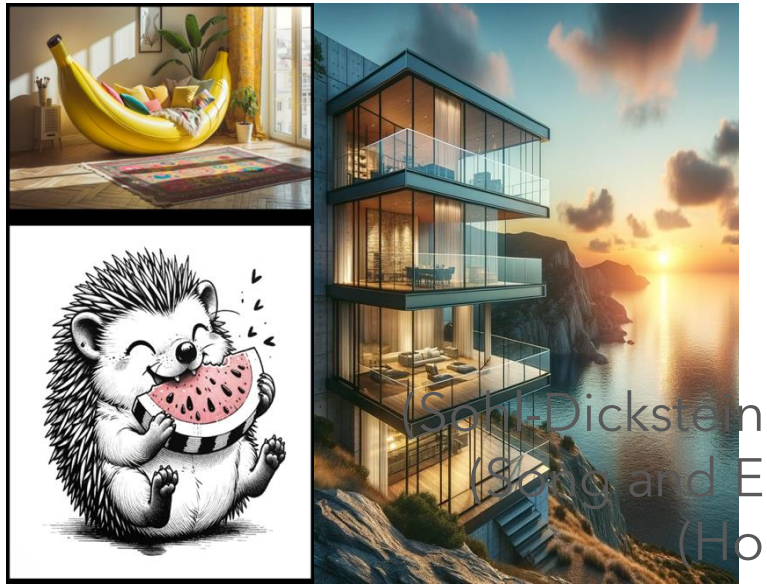
High-D





> 890M parameters
(Stable Diffusion)

Backbone of Stable Diffusion, Sora, etc.



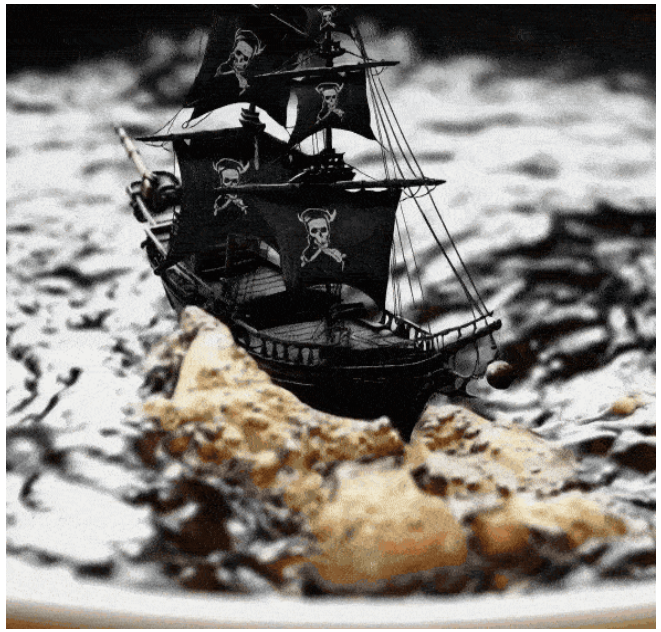
(Sohl-Dickstein et al., 2015)
(Song and Ermon, 2019)
(Ho et al., 2020)

Characteristics of Deep Generative AI Models

	VAE	GAN	Diffusion
Generation	Low-D to High-D One-step	Low-D to High-D One-step	High-D to High-D Sequential
Training		Unstable	Stable
Performance	Limited capacity	Mode collapse	Flexible and accurate

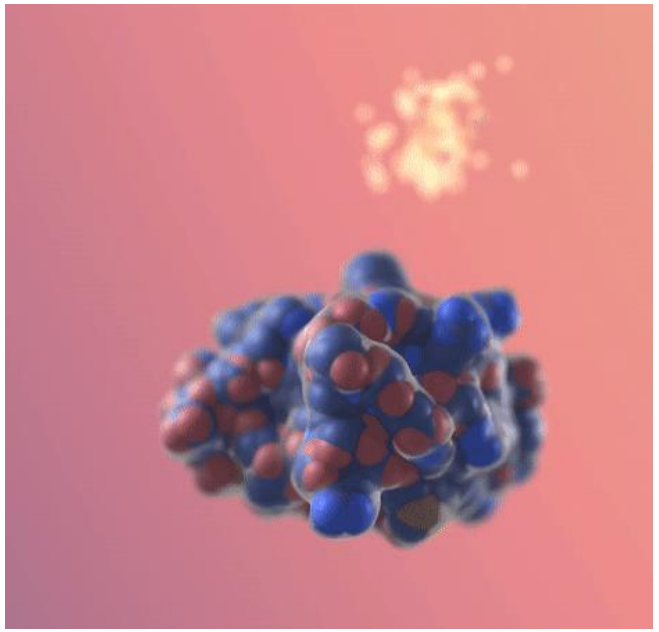
What Are Diffusion Models Capable of?

Sora



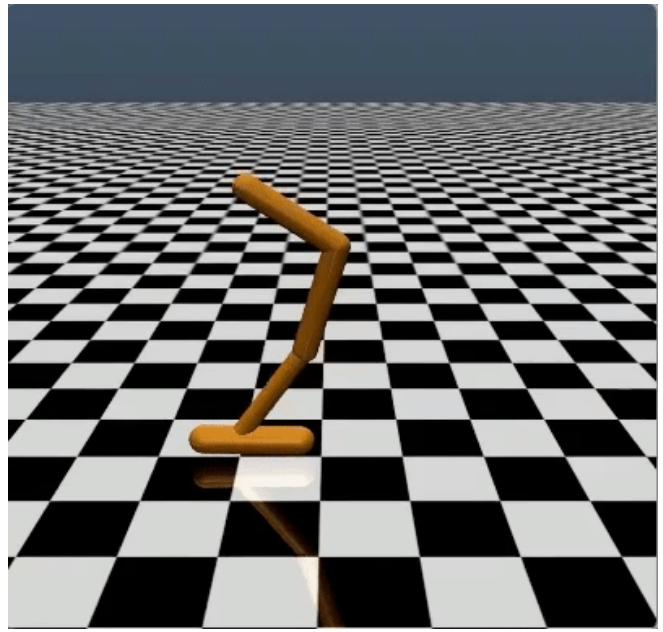
Video

RFDiffusion



Protein

Decision Diffuser



RL/control

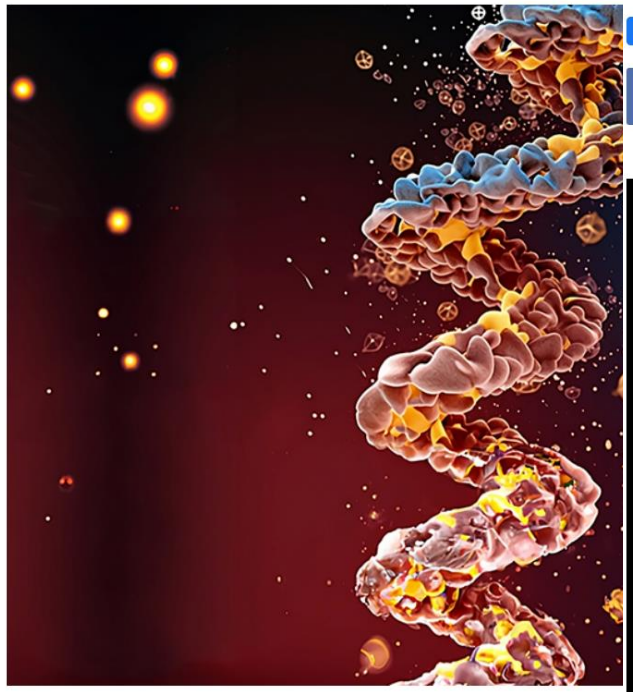
https://openai.com/index/video-generation-models-as-world-simulators/

Revolutionary Promises of Diffusion Models

Generative AI imagines new pr

"FrameDiff" is a computational tool that uses g
protein structures, with the aim of accelerating
improving gene therapy.

Rachel Gordon | MIT CSAIL
July 12, 2023



Biology is a wondrous yet delicate tapestry. At the heart is
encodes proteins, responsible for orchestrating the many
within the human body. However, our body is akin to a fine
losing its harmony. After all, we're faced with an ever-char
pathogens, viruses, diseases, and cancer.

Diffusion models are
turbocharging reinfo
learning systems

By **Ben Dickson** - March 4, 2024

Like 75

Facebook | Twitter | ReddIt

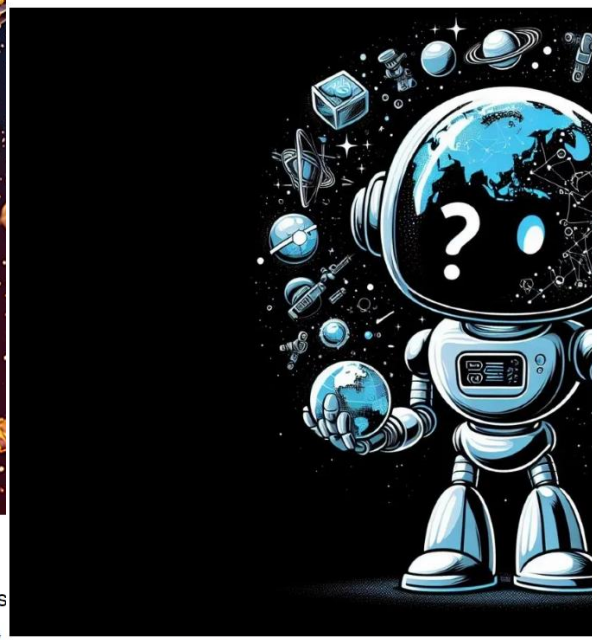


Image generated with Bing Image Creator

This article is part of our coverage of the late

ARTIFICIAL INTELLIGENCE

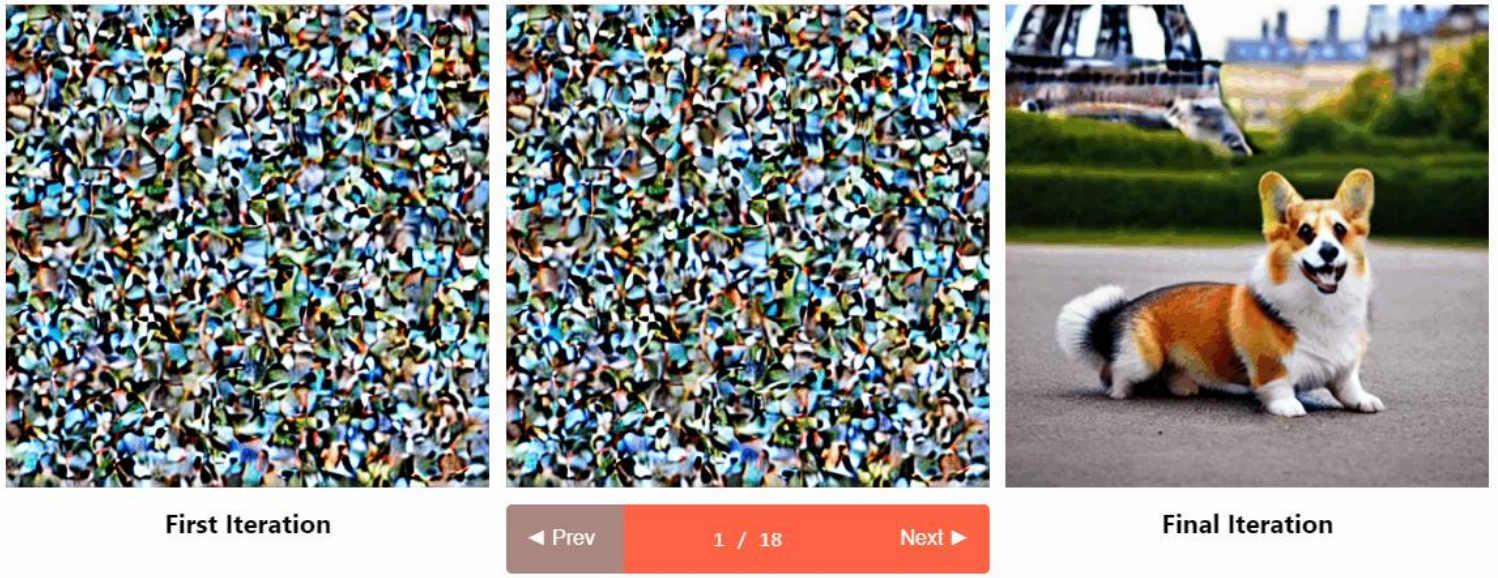
AniPortrait: Audio-Driven Synthesis of Photorealistic Portrait Animation

Published 3 days ago on May 3, 2024
By **Kunal Kejriwal**

Over the years, the creation of realistic and expressive portraits animations from static images and audio has found a range of applications including gaming, digital media, virtual reality, and a lot more. Despite its potential application, it is still difficult for developers to create frameworks capable of generating high-quality animations that maintain temporal consistency and are visually captivating. A major cause for the complexity is the need for intricate coordination of lip movements, head positions, and facial expressions to craft a visually compelling effect.

Diffusion Model Generates Samples

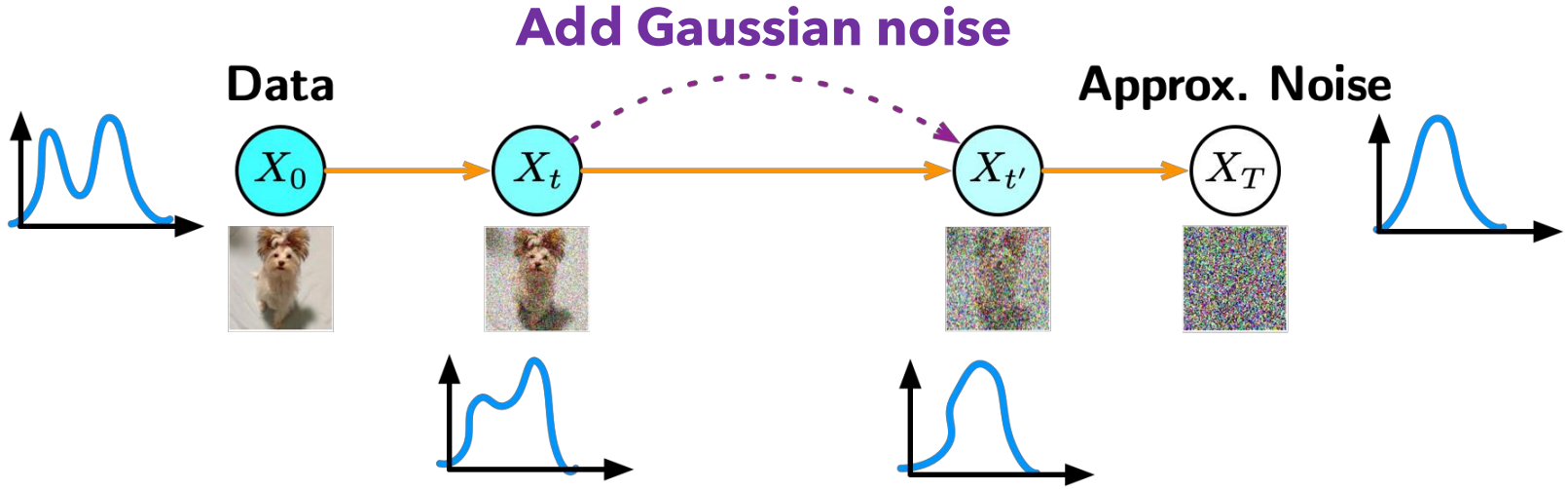
- Generate samples from noise



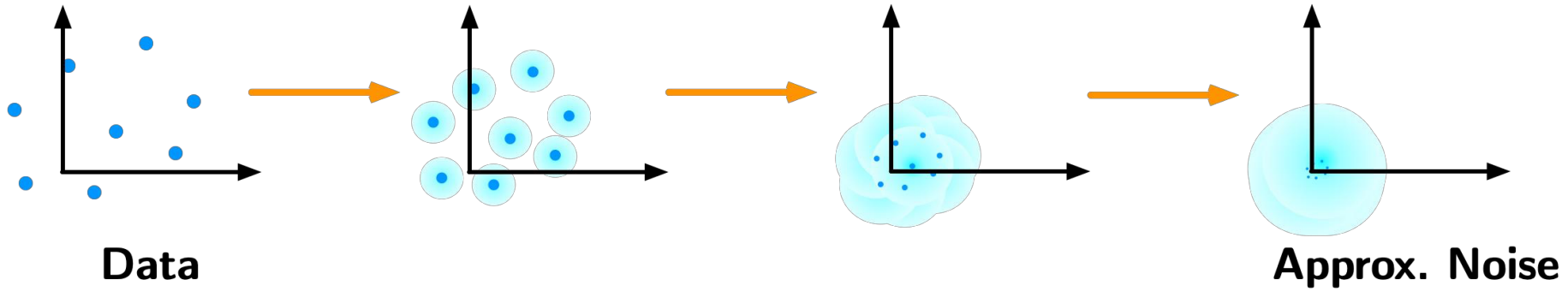
-- Credit: online

Forward Process – Noise Corruption

- Noise corruption process (in discrete-time)

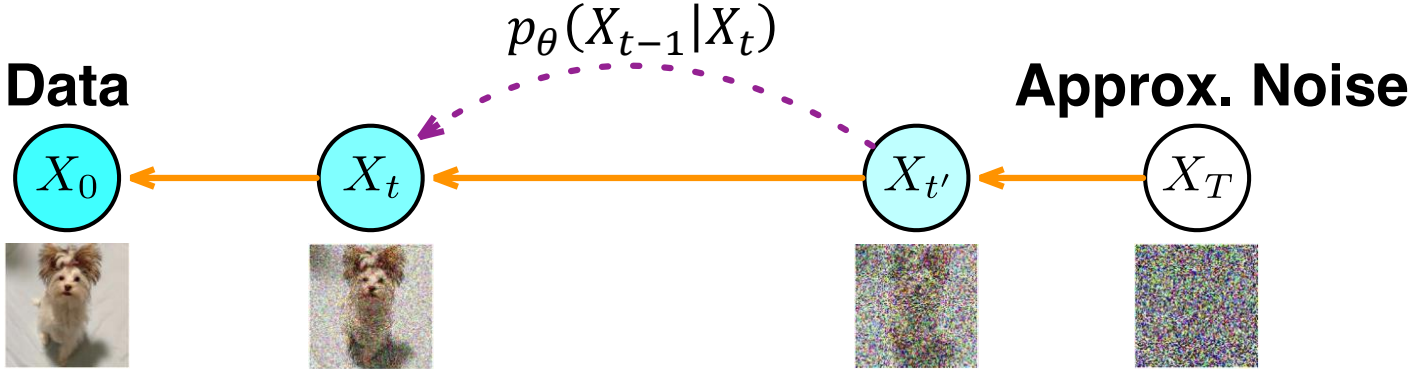


- The noise corruption



Backward Process – Sample Generation

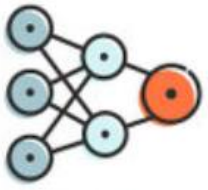
- Time reversal in distribution



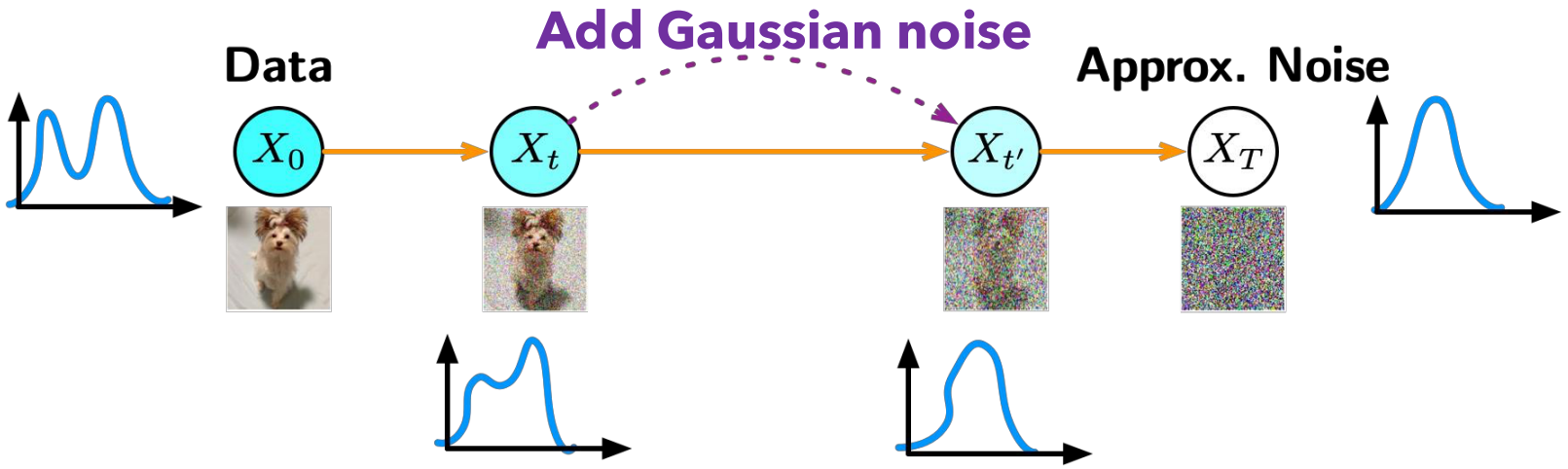
Distribution model

Use $p_\theta(X_{t-1}|X_t) = N(\mu_\theta(X_t, t), \sigma_t^2)$ to approximate the backward transition probabilities

Neural Networks



Forward & Backword Process in Continuous Time



- Perturbing data with a Stochastic Differential Equations (SDE) (Anderson, 1982; Haussmann and Pardoux, 1986)

Forward $$dX_t = -\frac{1}{2}\beta_t X_t dt + \sqrt{\beta_t} dW_t$$

Brownian

Backward $$dX_t = -\frac{1}{2}\beta_t X_t dt - \beta_t \nabla_{X_t} \log p_t(X_t) dt + \sqrt{\beta_t} dW_t$$

Score Function

Theorem. *Let x_t be the process described by (3.3), and suppose $f(\cdot,\cdot)$ and $g(\cdot,\cdot)$ are such as to guarantee the existence of the probability density $p(x_t, t)$ for $t_0 \leqslant t \leqslant T$ as a smooth and unique solution of its associated Kolmogorov equation. Suppose further that an r-vector process $\bar{w}_t$ is defined by $\bar{w}_{t_0} = 0$ and*

$$\mathrm{d}\bar{w}_t^k = \mathrm{d}w_t^k + \frac{1}{p(x_t,t)} \sum_i \frac{\partial}{\partial x_t^i}[p(x_t,t) g^{ik}(x_t,t)]\,\mathrm{d}t, \tag{3.10}$$

and that the forward Kolmogorov equation associated with the joint process $(x_t, \bar{w}_t)$ yields a smooth and unique solution in $t > t_0$ for $p(x_t, \bar{w}_t, t)$ and in $t > s \geqslant t_0$ for $p(x_t, \bar{w}_t, t \mid \bar{w}_s, s)$. Then

(i) x_t and $\bar{w}_t - \bar{w}_s$ are independent for all $t \geqslant s \geqslant t_0$.

(ii) With $\mathcal{B}_t$ the minimal σ-algebra with respect to which x_s for $s \geqslant t$ and $\bar{w}_s$ for $s \geqslant t$ are measurable, conditions (3.4) and (3.5) hold.

(iii) A reverse time model for x_t is defined by

$$\mathrm{d}x_t = \bar{f}(x_t, t)\,\mathrm{d}t + g(x_t, t)\,\mathrm{d}\bar{w}_t \tag{3.11}$$

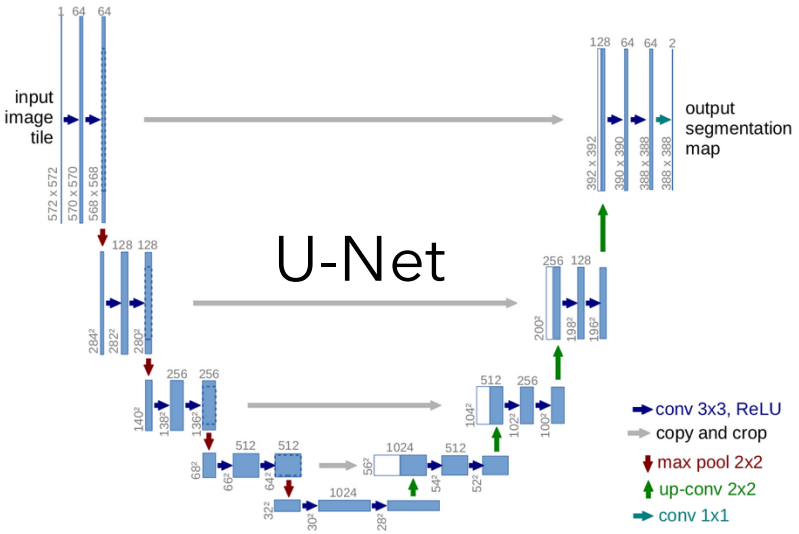
where

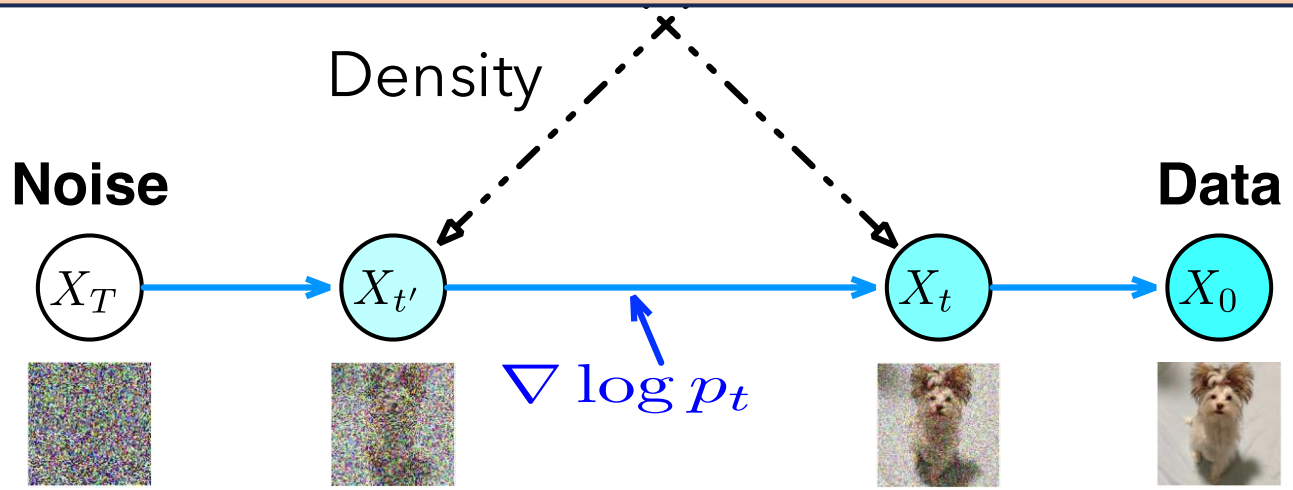
$$\bar{f}^i(x_t, t) = f^i(x_t, t) - \frac{1}{p(x_t,t)} \sum_j \frac{\partial}{\partial x_t^j}[p(x_t,t) g^{ik}(x_t,t) g^{jk}(x_t,t)]. \tag{3.12}$$

Forward and Backward Coupling

Score Network

Training $\int_0^T \mathbb{E}_{x_t}\left[\|\nabla \log p_t(x_t) - s(x_t, t)\|_2^2\right] \mathrm{d}t$





Ronneberger, et al. 2015

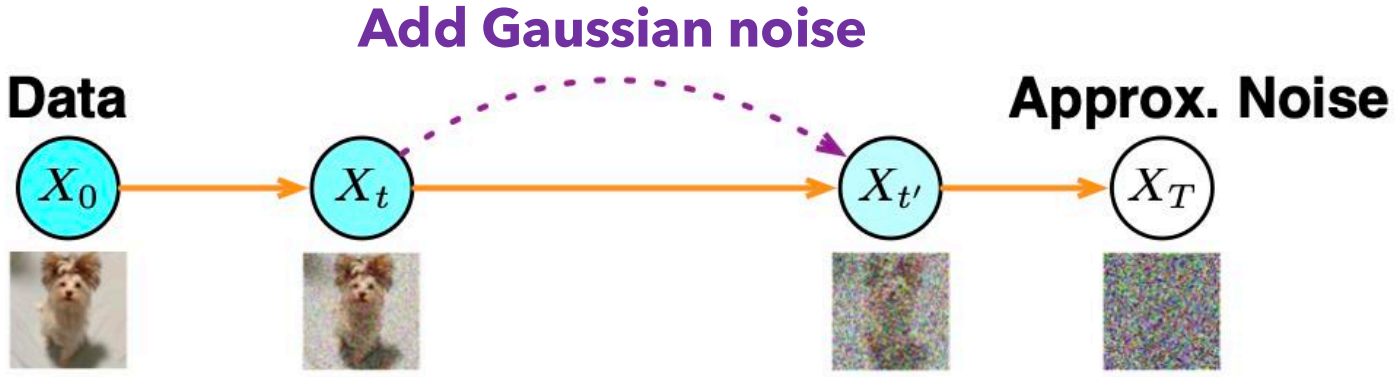
Outline

- A brief introduction of diffusion models
- Proposed algorithm: density-ratio inspired guidance
- Improved techniques: two additional regularization terms
- Numerical experiments

Insights

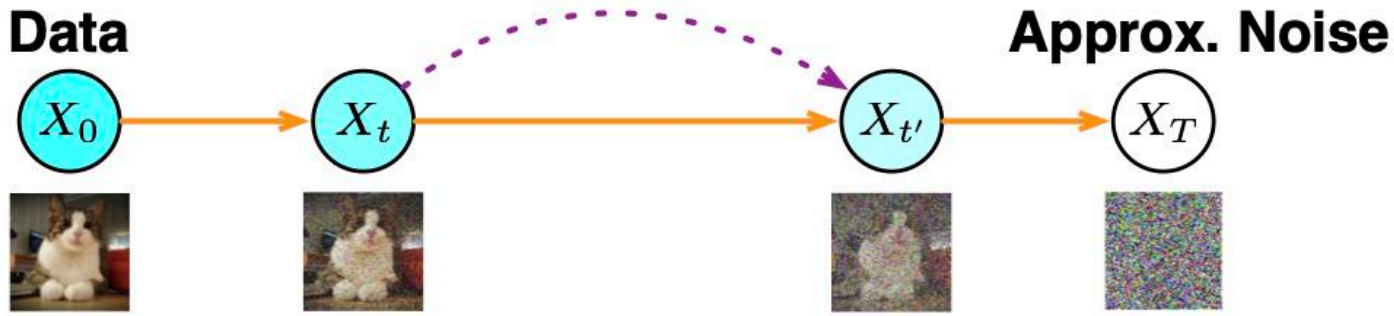
- Perturbing data with the same forward process

$X_0 \sim p$ (source dist.)



$$p(X_t|X_0) = q(X_t|X_0)$$

$X_0 \sim q$ (target dist.)



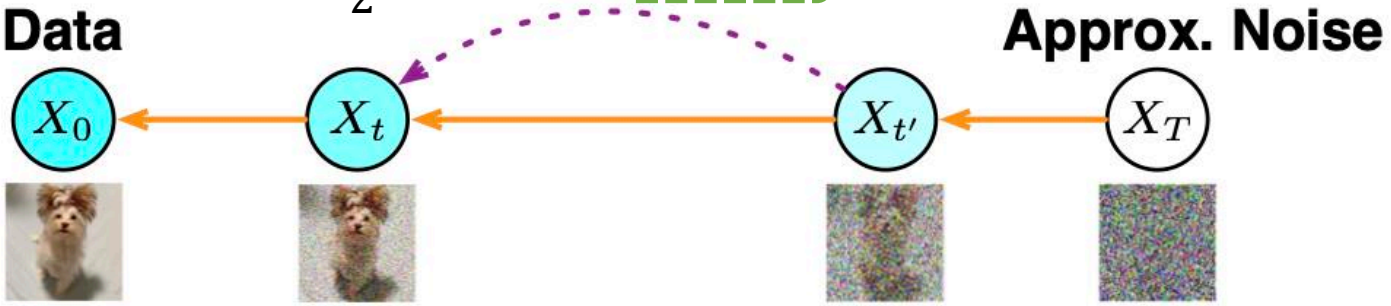
Insights

- Perturbing data with the same forward process

Source Score Function

$$dX_t = \frac{1}{2}\beta_t X_t dt + \beta_t \nabla_{X_t} p_t(X_t) dt + \sqrt{\beta_t} dW_t$$

$X_0 \sim p$ (source dist.)



$$\nabla_{X_t} \log q_t(X_t) = \nabla_{X_t} \log p_t(X_t) + \nabla_{X_t} \log E_{p(X_0|X_t)} \left[\frac{q(X_0)}{p(X_0)} \right]$$

$X_0 \sim q$ (target dist.)

Target Score Function

$$dX_t = \frac{1}{2}\beta_t X_t dt + \beta_t \nabla_{X_t} q_t(X_t) dt + \sqrt{\beta_t} dW_t$$

Insights

Theorem

Consider two diffusion models on the source and target domain, denoted as p and q. Let the forward process on the target domain be identical to that on the source domain, $p(X_t|X_0) = q(X_t|X_0)$, and $s_{\phi^*}(X_t, t)$ is the score estimator in the target domain:

$$\boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi}} \mathbb{E}_t \left\{ \lambda(t) \mathbb{E}_{q_t(\mathbf{x}_t)} \left[\| \mathbf{s}_{\boldsymbol{\phi}}(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t) \|_2^2 \right] \right\},$$

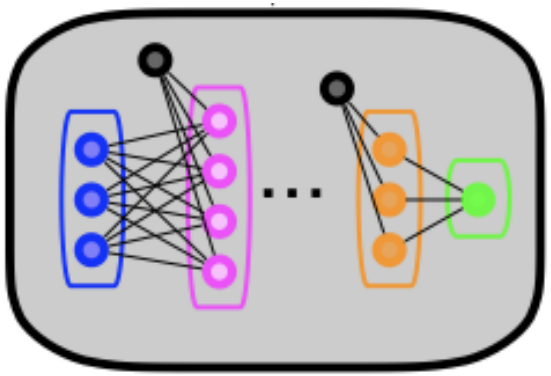
then we have

$$\mathbf{s}_{\boldsymbol{\phi}^*}(\mathbf{x}_t, t) = \underbrace{\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)}_{\textit{pre-trained model on source}} + \underbrace{\nabla_{\mathbf{x}_t} \log \mathbb{E}_{p(\mathbf{x}_0|\mathbf{x}_t)} \left[\frac{q(\mathbf{x}_0)}{p(\mathbf{x}_0)} \right]}_{\textit{guidance}}$$

Guidance network

- Train a guidance network to approximate the guidance term

Guidance Network

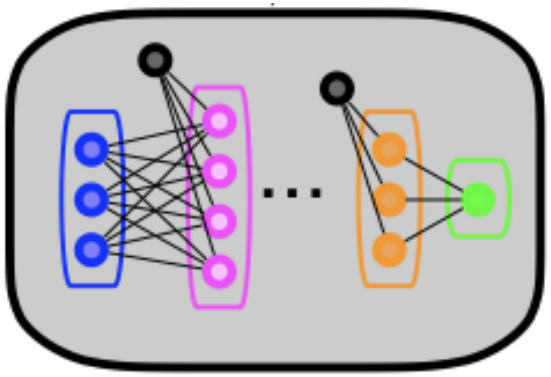


$$h_{\psi^*}(X_t, t) \approx \mathbb{E}_{p(\mathbf{x}_0|\mathbf{x}_t)}\left[\frac{q(\mathbf{x}_0)}{p(\mathbf{x}_0)}\right]$$

- Step 1: train a binary classifier to approximate the density ratio
- Step 2: using Monte Carlo simulation to estimate $h_{\psi^*}(X_t, t)$

Guidance network

Guidance Network



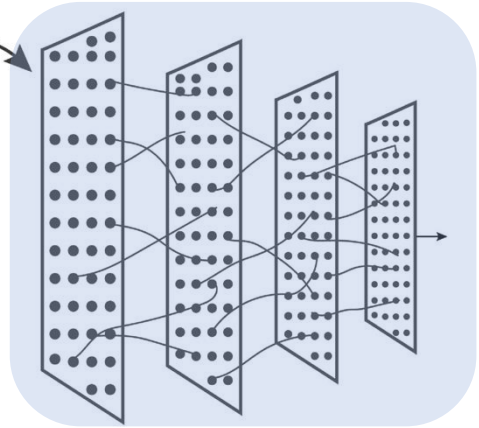
$$h_{\psi^*}(X_t, t) \approx \mathbb{E}_{p(\mathbf{x}_0|\mathbf{x}_t)}\left[\frac{q(\mathbf{x}_0)}{p(\mathbf{x}_0)}\right]$$

- Step 1: train a binary classifier to approximate the density ratio

$$c_{\boldsymbol{\omega}}(\mathbf{x}) : \mathcal{X} \to [0, 1]$$

source data

target data



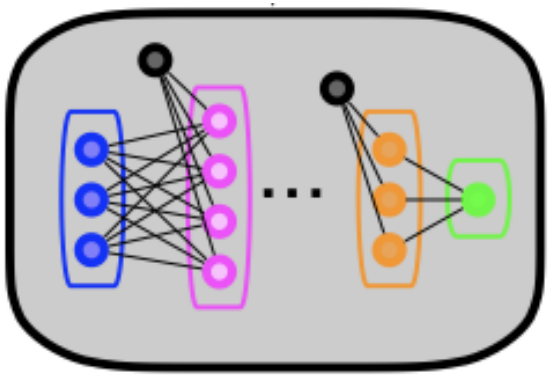
logistic loss

$$\boldsymbol{\omega}^* = \arg\min_{\boldsymbol{\omega}} \left\{ -\frac{1}{m} \sum_{\mathbf{x}_i \sim p} \log c_{\boldsymbol{\omega}}(\mathbf{x}_i) - \frac{1}{n} \sum_{\mathbf{x}'_i \sim q} \log(1 - c_{\boldsymbol{\omega}}(\mathbf{x}'_i)) \right\}$$

$$\frac{q(X_0)}{p(X_0)} = \frac{1 - c_{\omega^*}(X_0)}{c_{\omega^*}(X_0)}$$

Guidance network

Guidance Network



$$h_{\psi^*}(X_t, t) \approx \mathbb{E}_{p(\mathbf{x}_0|\mathbf{x}_t)}\left[\frac{q(\mathbf{x}_0)}{p(\mathbf{x}_0)}\right]$$

- Step 2: using Monte Carlo simulation to estimate $h_{\psi^*}(X_t, t)$

Lemma

For a neural network $h_\psi(X_t, t)$ parameterized by ψ, define the objective

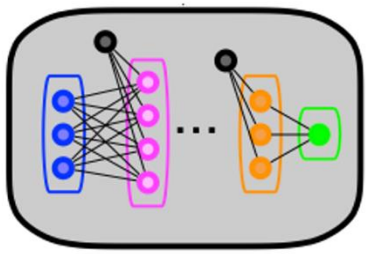
$$\mathcal{L}_{guidance}(\psi) := \mathbb{E}_{p(\mathbf{x}_0,\mathbf{x}_t)}\left[\left\| h_\psi(\mathbf{x}_t, t) - \frac{q(\mathbf{x}_0)}{p(\mathbf{x}_0)}\right\|_2^2\right],$$

then its minimizer $\psi^* = \arg\min \mathcal{L}_{guidance}(\psi)$ satisfies

$$h_{\psi^*}(\mathbf{x}_t, t) = \mathbb{E}_{p(\mathbf{x}_0|\mathbf{x}_t)}\left[q(\mathbf{x}_0)/p(\mathbf{x}_0)\right].$$

Guidance network

Guidance Network



$$h_{\psi^*}(X_t, t) \approx \mathbb{E}_{p(\mathbf{x}_0|\mathbf{x}_t)}\left[\frac{q(\mathbf{x}_0)}{p(\mathbf{x}_0)}\right]$$

Algorithm 3 Training a guidance network (without regularization)

Require: Samples from the marginal distribution of the source domain $p(\mathbf{x})$, pre-defined forward transition $p(\mathbf{x}_t|\mathbf{x}_0)$, pre-trained domain classifier c_{ω}, and initial weights of guidance network ψ.

1: **repeat**
2: Sample mini-batch data from source distribution $\mathbf{x}_0$ with batch size b.
3: Sample time $t \sim \text{Uniform}(\{1, \ldots, T\})$ and perturb $\mathbf{x}_0$ by forward transition $p(\mathbf{x}_t|\mathbf{x}_0)$.
4: Take gradient descent step on

Jointly sample $(X_0, X_t) \sim p$

Density ratio estimate

$$\nabla_{\psi}\left\{\frac{1}{b}\sum_{\mathbf{x}_0,\mathbf{x}_t}\left[\left\|h_{\psi}(\mathbf{x}_t, t) - (1 - c_{\omega}(\mathbf{x}_0))/c_{\omega}(\mathbf{x}_0)\right\|_2^2\right]\right\}.$$

5: **until** converged.
6: **return** weights of guidance network ψ.

Guidance Network

Outline

- A brief introduction of diffusion models
- Proposed algorithm: density-ratio inspired guidance
- Improved techniques: two additional regularization terms
- Numerical experiments

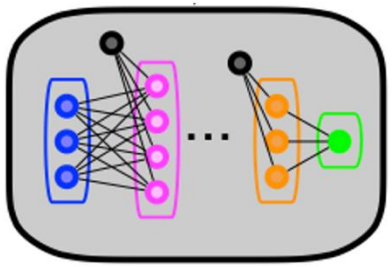
Guidance network

- Two regularization terms to enhance the performance

Cycle Regularization

Incorporate the limited data from the target domain into the training of guidance network

Guidance Network



$$h_{\psi^*}(X_t, t) \approx \mathbb{E}_{p(\mathbf{x}_0|\mathbf{x}_t)}\left[\frac{q(\mathbf{x}_0)}{p(\mathbf{x}_0)}\right]$$

$$\mathbb{E}_{p(\mathbf{x}_0|\mathbf{x}_t)}\left[\frac{q(\mathbf{x}_0)}{p(\mathbf{x}_0)}\right] = \mathbb{E}_{q(\mathbf{x}_0|\mathbf{x}_t)}\left[\frac{q_t(\mathbf{x}_t)}{p_t(\mathbf{x}_t)}\right]$$

The minimizer ψ^* is also the minimizer of

$$\psi^* = \arg\min_{\psi} \mathcal{L}_{\text{cycle}} := \mathbb{E}_{q(\mathbf{x}_0,\mathbf{x}_t)}\left[\left\| h_\psi(\mathbf{x}_t, t) - \frac{q_t(\mathbf{x}_t)}{p_t(\mathbf{x}_t)} \right\|_2^2\right]$$

Monte carlo simulation using target data

Guidance network

- Two regularization terms to enhance the performance

Consistency Regularization

an optimal guidance network should recover the score in the target domain

$$\mathbf{s}_{\phi^*}(\mathbf{x}_t, t) = \underbrace{\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)}_{\text{pre-trained model on source}} + \underbrace{\nabla_{\mathbf{x}_t} \log h_{\psi^*}(\mathbf{x}_t, t)}_{\textbf{Guidance Network}}$$

The minimizer ψ^* should be the minimizer of

$$\begin{aligned}\psi^* &= \arg\min_{\psi} \mathcal{L}_{\text{consistence}} \\ &:= \mathbb{E}_t \left\{ \lambda(t) \mathbb{E}_{q(\mathbf{x}_0)} \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \left[\| \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{x}_0) + \nabla_{\mathbf{x}_t} \log h_{\psi}(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t|\mathbf{x}_0) \|_2^2 \right] \right\}\end{aligned}$$

Guidance network

- Training with two regularization terms

Algorithm 4 Training a guidance network (with regularization)

Require: Samples from the marginal distribution of the source domain $p(\mathbf{x})$ and target domain $q(\mathbf{x})$, pre-trained diffusion model on source distribution $s_{\text{source}}(\mathbf{x}_t, t)$, pre-defined forward transition $q(\mathbf{x}_t|\mathbf{x}_0), p(\mathbf{x}_t|\mathbf{x}_0)$, pre-trained domain classifier $c_{\boldsymbol{\omega}}(\mathbf{x}_0)$ and time dependent domain classifier $c'_{\boldsymbol{\omega}}(\mathbf{x}_0, t)$, hyperparameter η_1, η_2, and initial weights of guidance network $\boldsymbol{\psi}$.

repeat

Sample mini-batch data from source distribution $\mathbf{x}_0$ with batch size b.

Perturb $\mathbf{x}_0$ by forward transition $p(\mathbf{x}_t|\mathbf{x}_0)$.

Jointly sample $(X_0, X_t) \sim p$

$$\mathcal{L}_{\text{guidance}}(\boldsymbol{\psi}) = \frac{1}{b} \sum_{\mathbf{x}_0, \mathbf{x}_t, t} \left[\left\| h_{\boldsymbol{\psi}}(\mathbf{x}_t, t) - (1 - c_{\boldsymbol{\omega}}(\mathbf{x}_0))/c_{\boldsymbol{\omega}}(\mathbf{x}_0) \right\|_2^2 \right]$$

Sample mini-batch data from target distribution $\mathbf{x}'_0$ with batch size b.

Sample time $t \sim \text{Uniform}(\{1, \ldots, T\})$ and perturb $\mathbf{x}'_0$ by forward transition $q(\mathbf{x}'_t|\mathbf{x}'_0)$.

Jointly sample $(X_0, X_t) \sim q$

$$\mathcal{L}_{\text{cycle}}(\boldsymbol{\psi}) = \frac{1}{b} \sum_{\mathbf{x}'_0, \mathbf{x}'_t, t} \left[\left\| h_{\boldsymbol{\psi}}(\mathbf{x}'_t, t) - (1 - c'_{\boldsymbol{\omega}}(\mathbf{x}'_0, t))/c'_{\boldsymbol{\omega}}(\mathbf{x}'_0, t) \right\|_2^2 \right].$$

$$\mathcal{L}_{\text{consistence}}(\boldsymbol{\psi}) = \frac{1}{b} \sum_{\mathbf{x}'_0, \mathbf{x}'_t, t} \left[\left\| s_{\text{source}}(\mathbf{x}'_t, t) + \nabla_{\mathbf{x}'_t} \log h_{\boldsymbol{\psi}}(\mathbf{x}'_t, t) - \nabla_{\mathbf{x}'_t} \log q(\mathbf{x}_t|\mathbf{x}'_0) \right\|_2^2 \right].$$

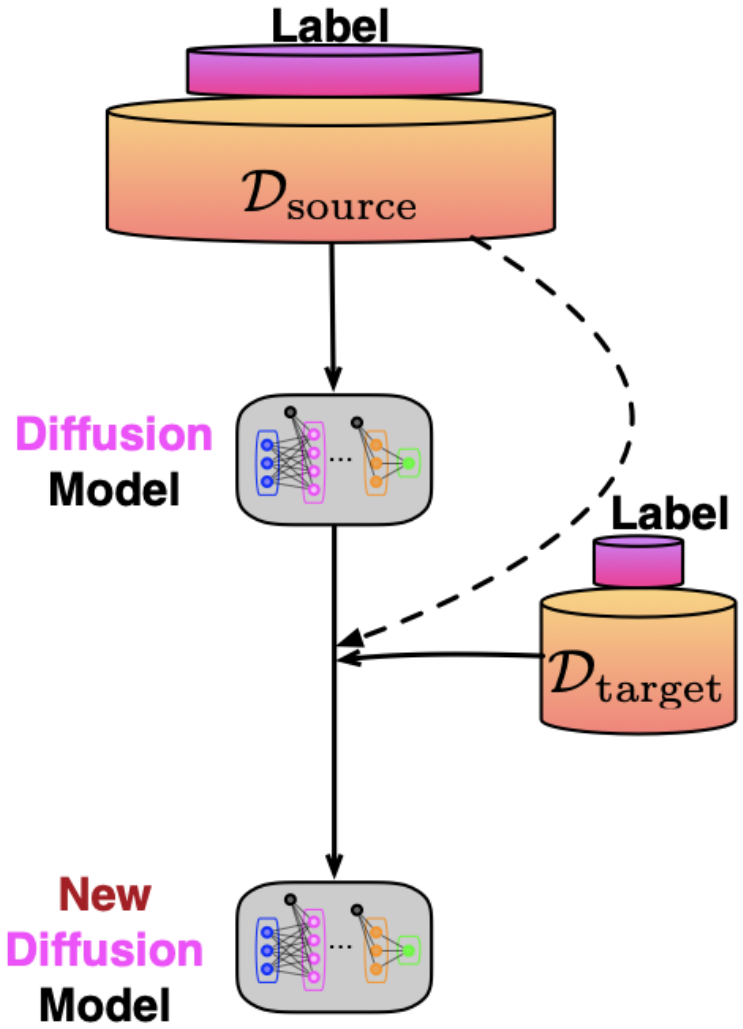
Take gradient descent step on

$$\nabla_{\boldsymbol{\psi}} \left\{ \mathcal{L}_{\text{guidance}} + \eta_1 \mathcal{L}_{\text{cycle}} + \eta_2 \mathcal{L}_{\text{consistence}} \right\}.$$

until converged.

return weights of guidance network $\boldsymbol{\psi}$.

Extension to Conditional Diffusion



Score network (target domain)

Score network (source domain)

$$\mathbf{s}_{\phi^*}(\mathbf{x}_t, y, t) = \underbrace{\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|y)}_{\textit{pre-trained conditional model on source}} + \underbrace{\nabla_{\mathbf{x}_t} \log \mathbb{E}_{p(\mathbf{x}_0|\mathbf{x}_t,y)} \left[\frac{q(\mathbf{x}_0, y)}{p(\mathbf{x}_0, y)} \right]}_{\textit{conditional guidance}}$$

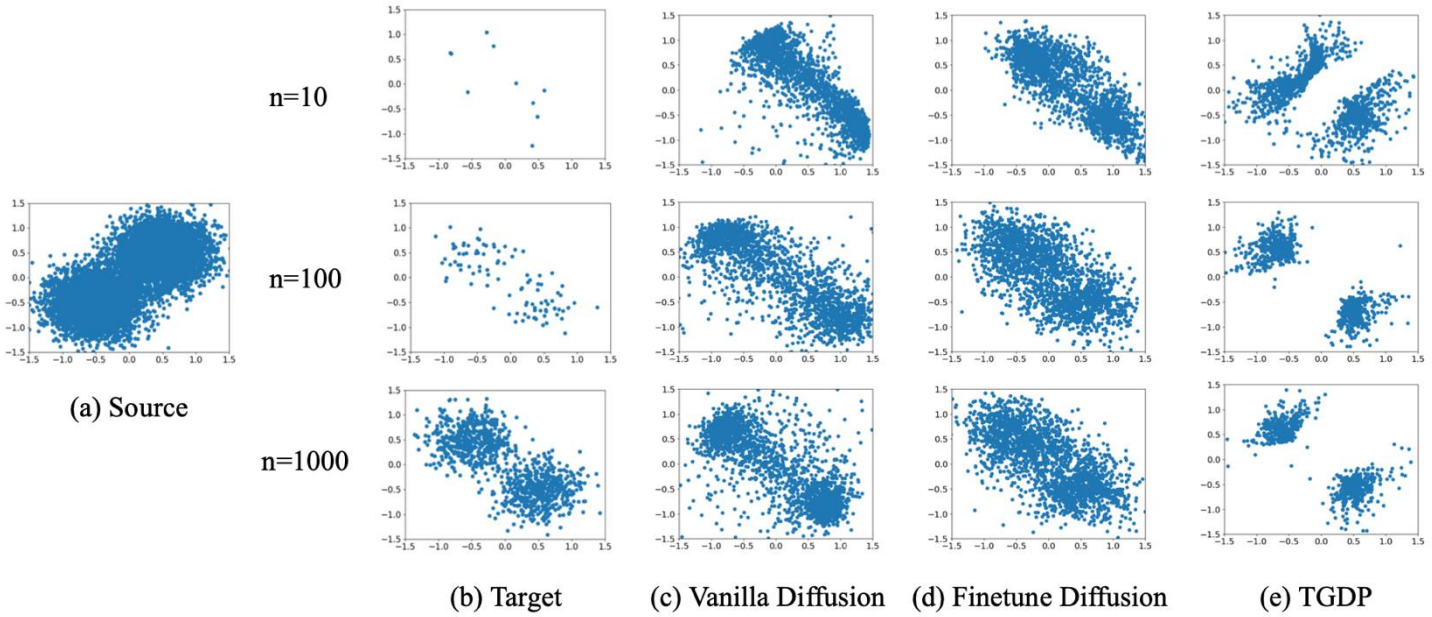
Approximated by a Guidance Network

Outline

- A brief introduction of diffusion models
- Proposed algorithm: density-ratio inspired guidance
- Improved techniques: two additional regularization terms
- Numerical experiments

Simulation

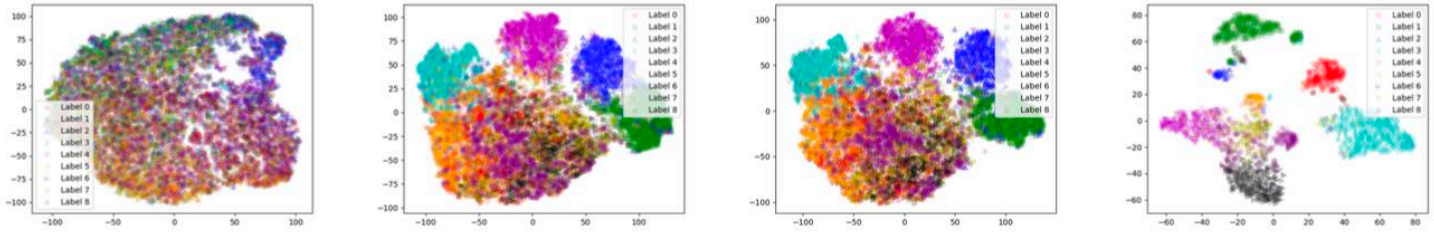
- Gaussian mixture distribution as source and target
- Target samples under different sample sizes n = 10, 100, 1000.
- Comparison with finetune diffusion method



	Average likelihood		
	n=10	n=100	n=1000
Vanilla Diffusion	0.145	0.253	0.328
Finetune	0.290	0.329	0.335
TGDP	**0.417**	**0.627**	**0.673**

Real data

- ECG dataset
  - Source data: PTB-XL dataset (21,837 clinical 12-lead ECG recordings of 10 seconds length from 18,885 unique patients)
  - Target data: ICBEB2018 dataset (6877 12-lead ECGs lasting between 6 and 60 seconds)



(a) Vanilla Diffusion (b) Finetune Generator (c) TGDP (d) Real

Figure 3: T-SNE of the generated ECG data.

Table 2: The effectiveness of TGDP on ECG benchmark under synthetic quality and diversity criteria.

Method	Diversity (↑)	FID (↓)	Number of Parameters	Training Time
Vanilla Diffusion	0.37	11.01	50.2M	1h
Finetune Generator	0.47	12.26	50.2M	40min
TGDP	**0.53**	**10.46**	**2.8M**	30min

Real data

- ECG dataset
- Also evaluate the performance for downstream task (classification)

Method	AUC	$F_{\beta=2}$	$G_{\beta=2}$
Vanilla Classifier	0.906(03)	0.674(06)	0.433(06)
Finetune Classifier	0.941(05)	0.747(08)	0.521(10)
Vanilla Diffusion	0.932(05)	0.718(09)	0.464(09)
Finetune Generator	0.941(04)	0.761(10)	0.528(12)
TGDP	**0.953**(05)	**0.773**(11)	**0.534**(11)

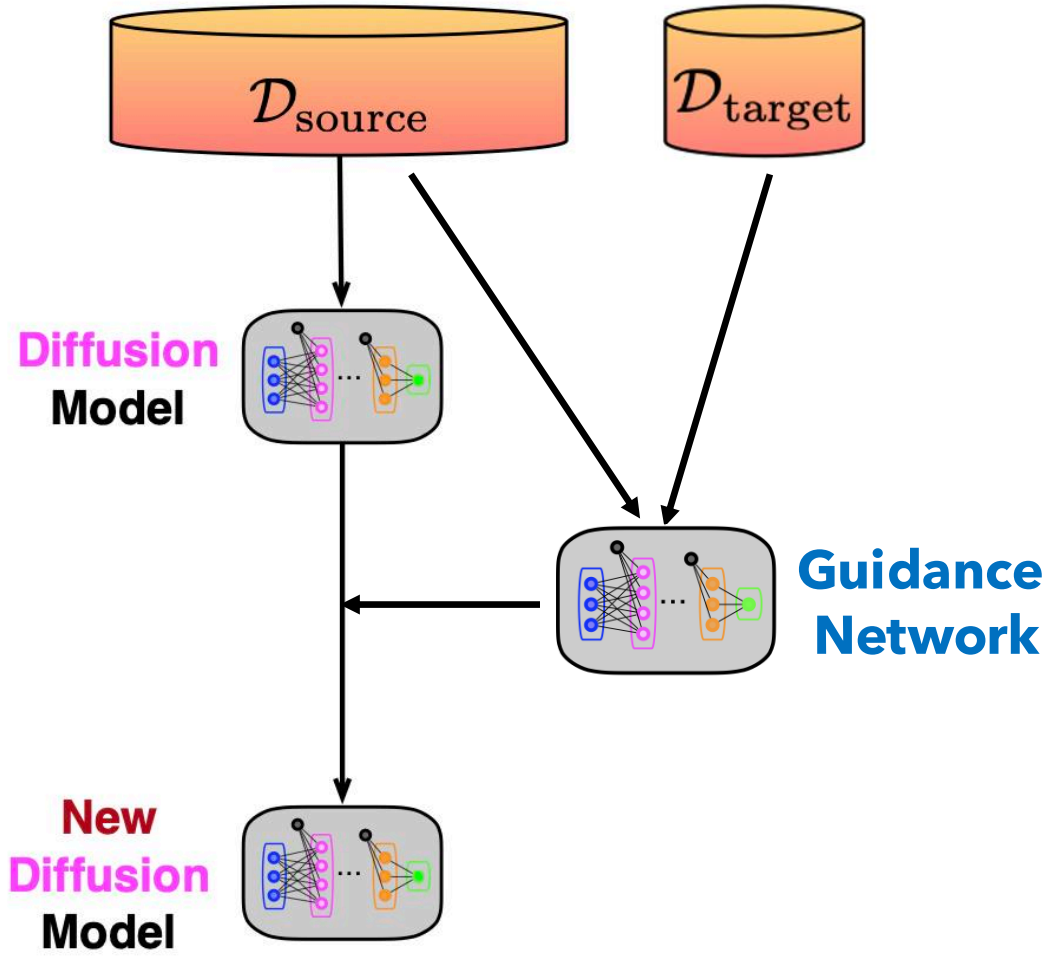
0.906(03) stands for 0.906 ± 0.003.

Comparison to literature

- Finetuning Diffusion Models on Limited Data
  - finetuning lightweight adapters [Moon et al, NeurIPS 2022 workshop] [Xie et al, ICCV 2023]
  - finetuning with regularization [Zhu et al, 2022]

- Text-to-Image Diffusion Models and Human Feedback
  - zero-shot finetuning [Song et al, 2022] [Radford et al, ICML 2021]
  - reward-based finetuning [Fan et al, NeurIPS 2023] [Gal et al, ICLR 2023] [Lee et al, 2023] [Kumari et al, CVPR 2023] [Ruiz et al, CVPR 2023]
- Non-diffusion generative models, such as GAN
  - regularization to avoid model collapse [Duan et al, AAAI 2024] [Ojha et al, CVPR 2021] [Zhang et al, NeurIPS 2022] etc.
  - subset tuning [Yang et al, ICCV 2021] [Zhao et al, ICML 2020] etc.

Summary



- Introduce a finetuning-free framework for transferring a pre-trained diffusion model from the source domain to the target domain
- Extendable to conditional version
- Future work: validations on more and diverse tasks

Ouyang Yidong, Liyan Xie, Hongyuan Zha, Guang Cheng. “Transfer Learning for Diffusion Models”. *NeurIPS 2024.*