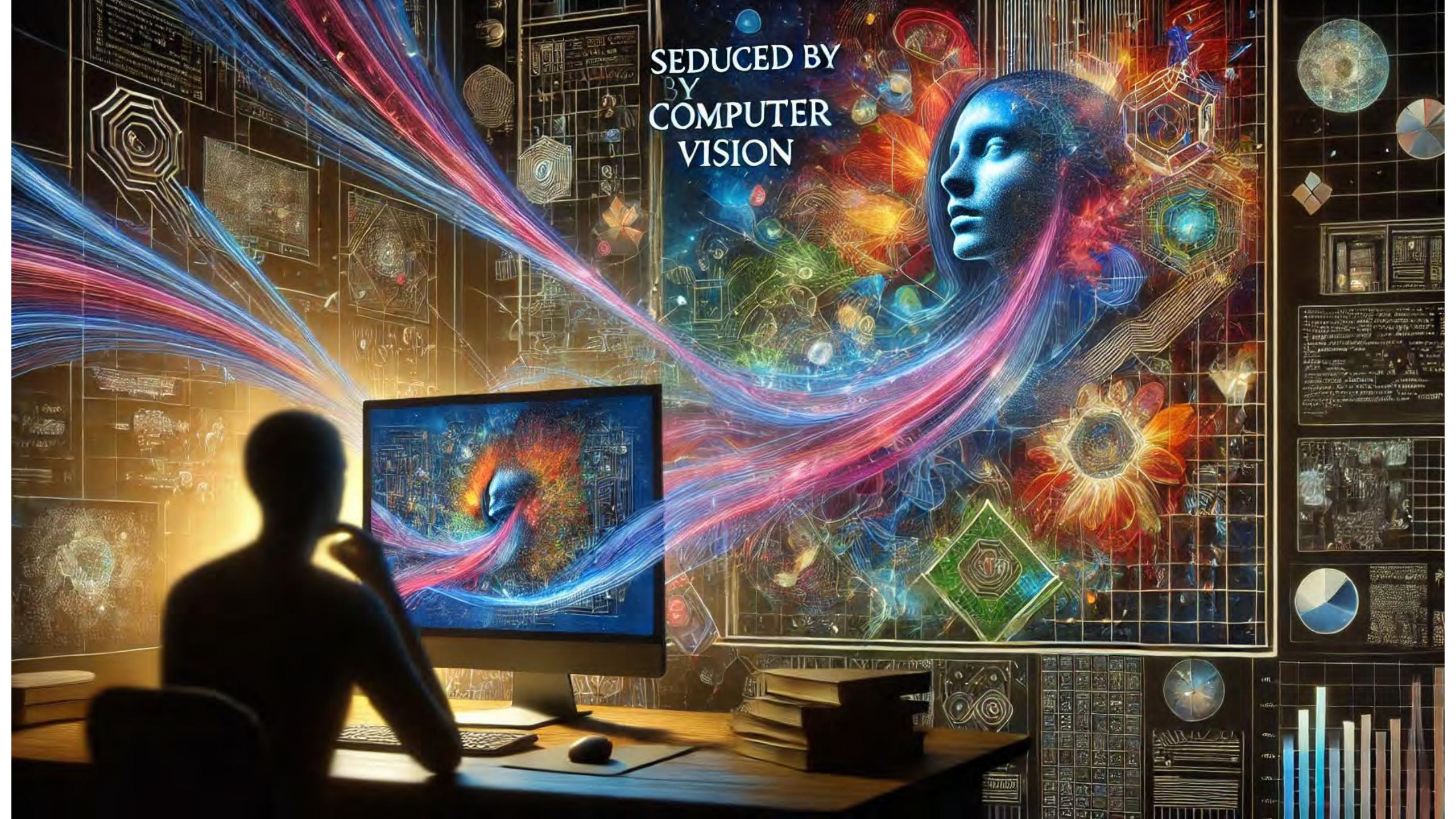


SEDUCED BY
COMPUTER
VISION



TBC

Richard Hartley

Kioloa, RVSS 2025



Transformer Based Construction

Kioloa, RVSS 2025



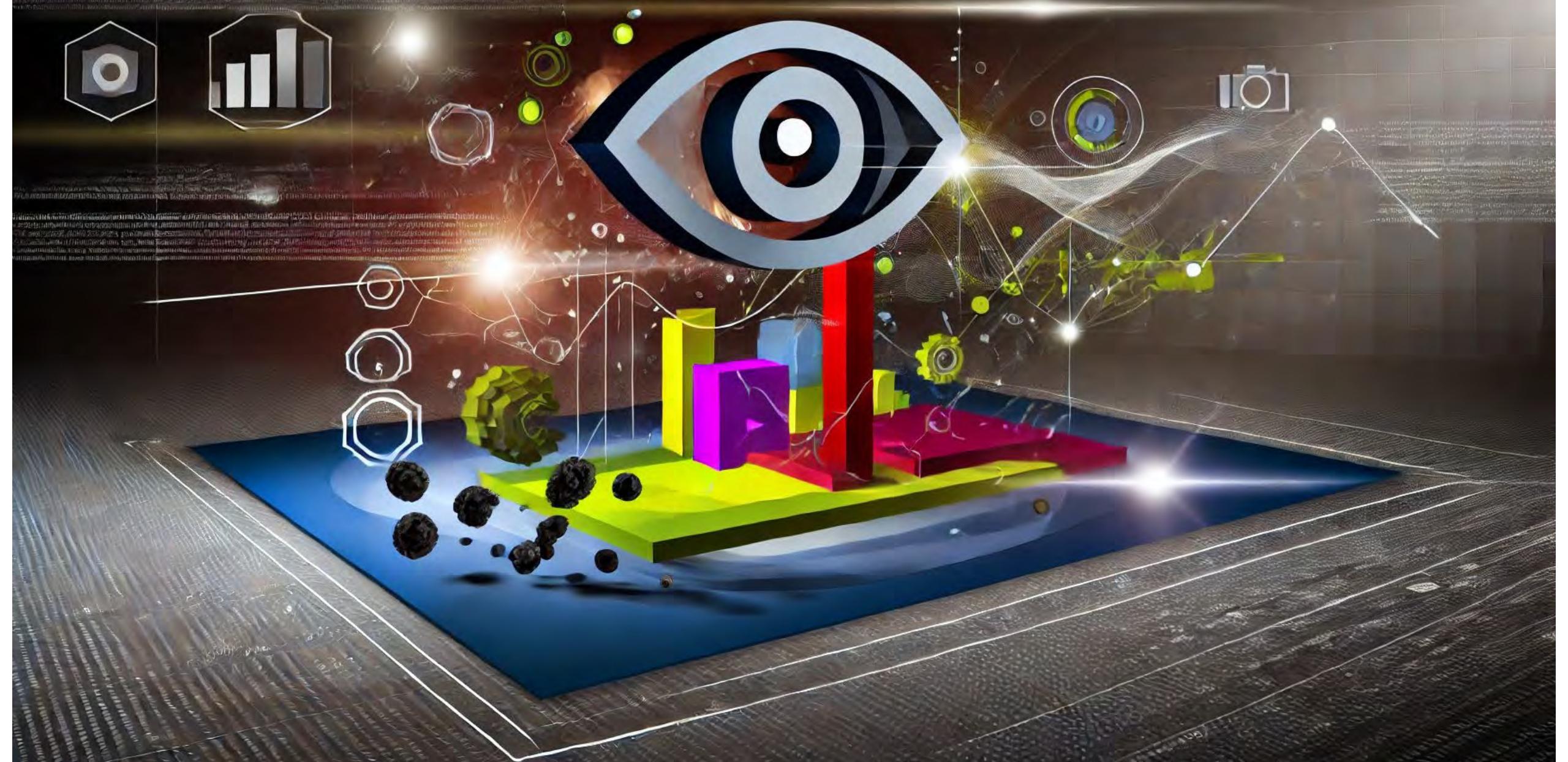


There are three kinds of lies:

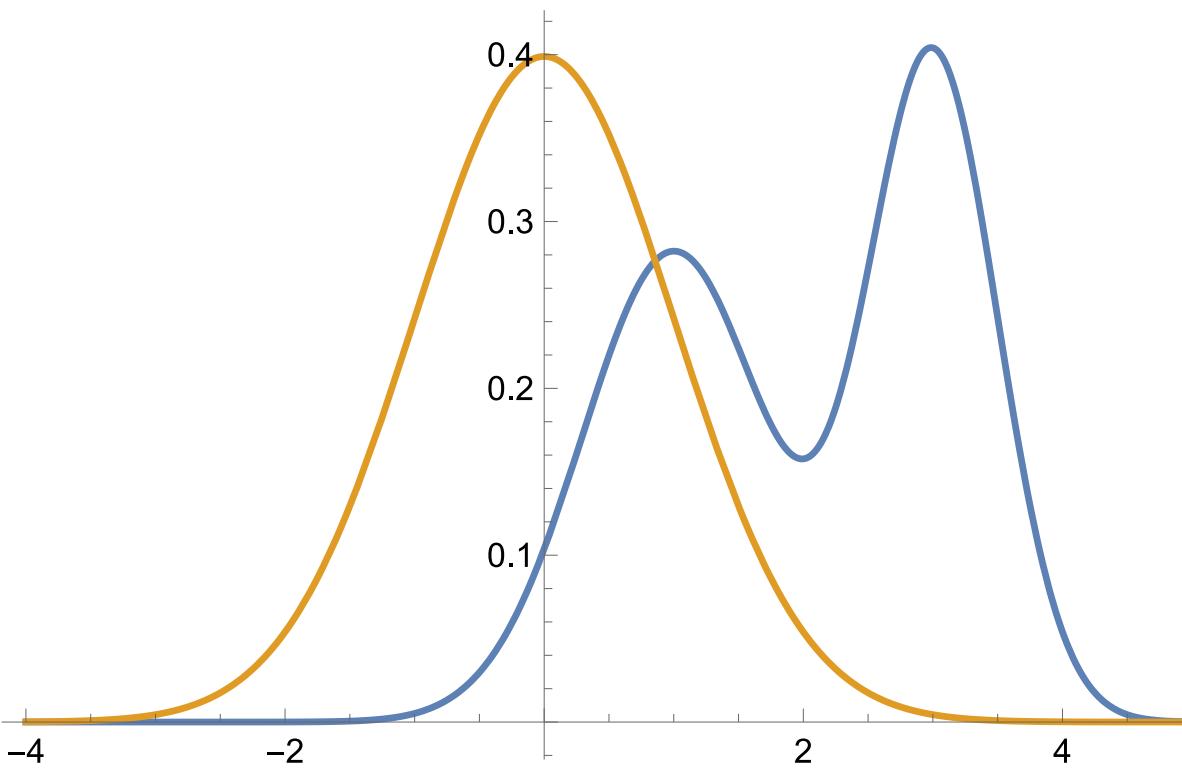
There are three kinds of lies:

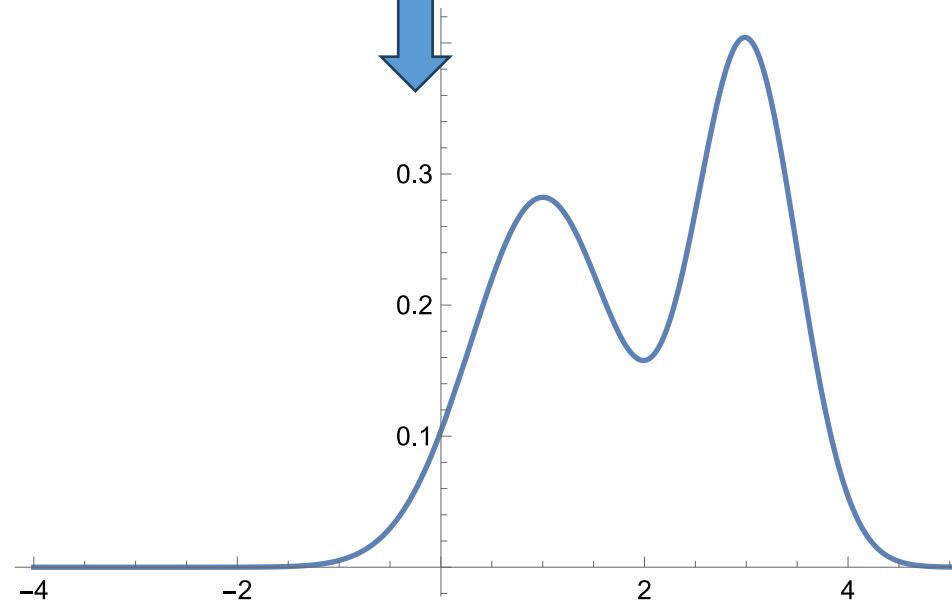
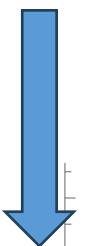
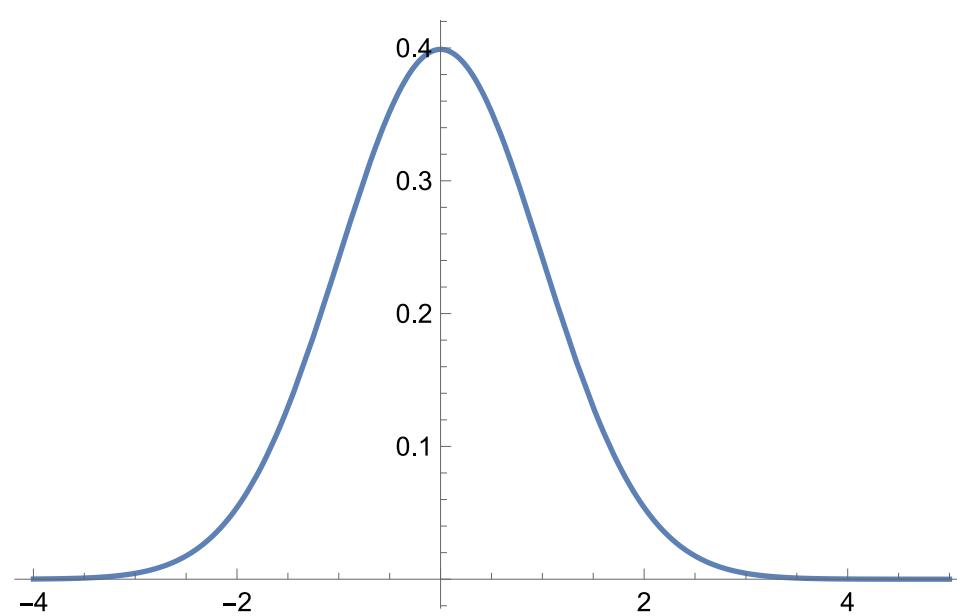
- There are lies
- Damned lies
- Statistics

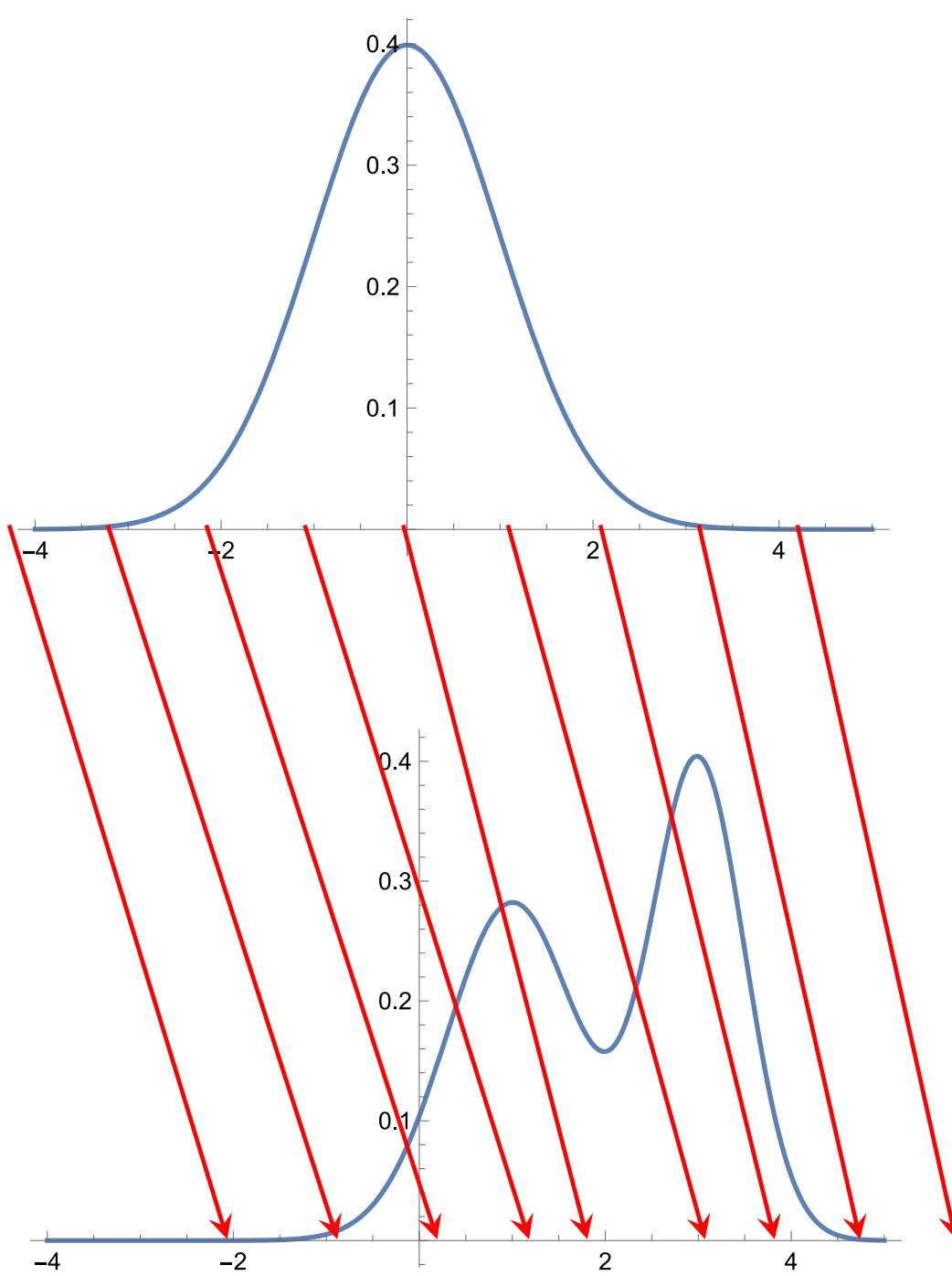
VISION-BASED GRAPHICS



Sampling from distributions



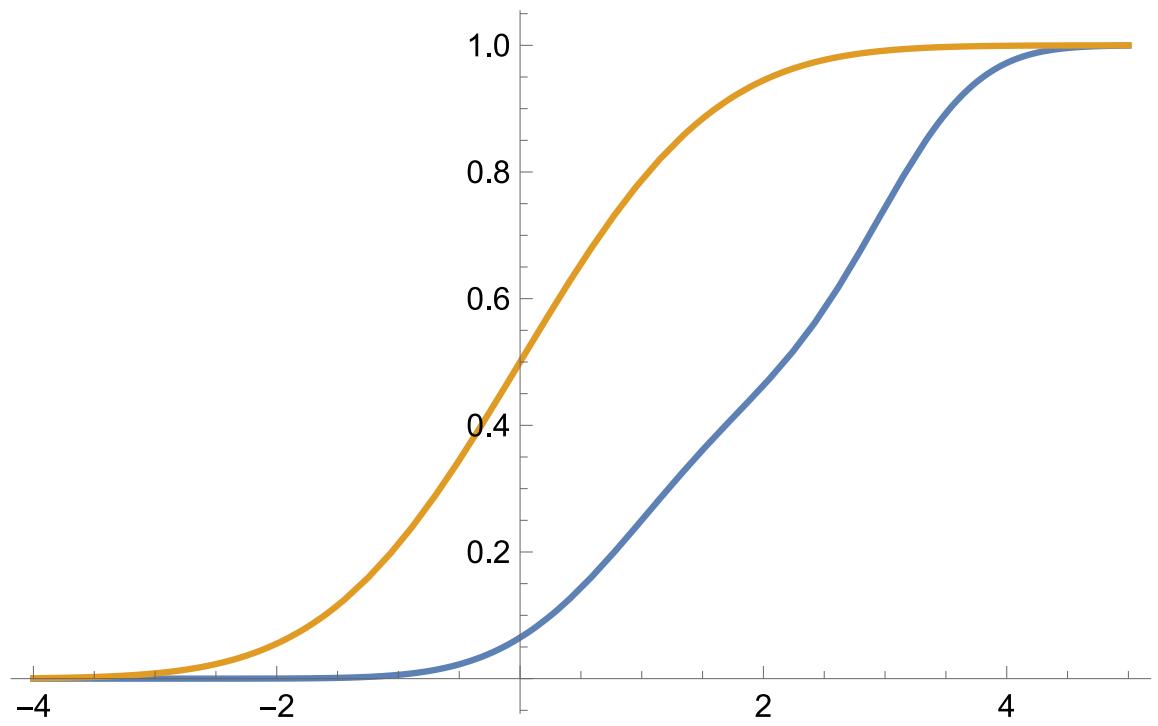
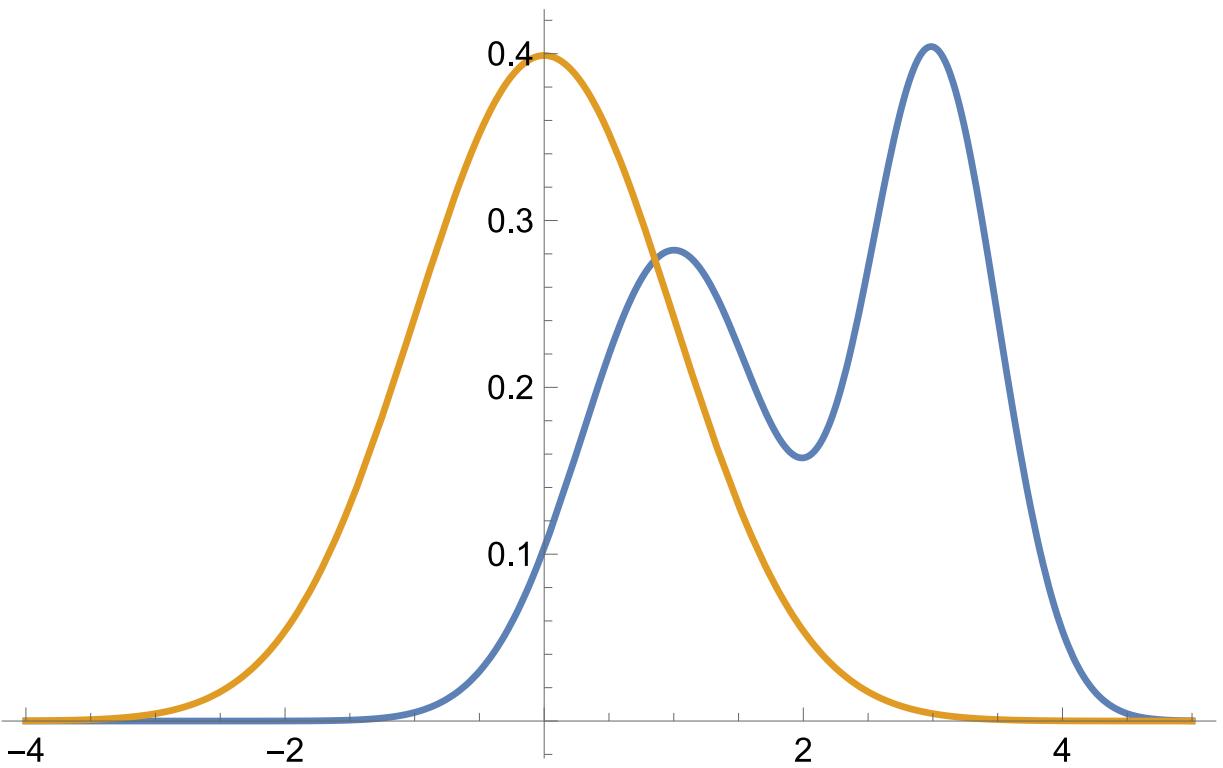


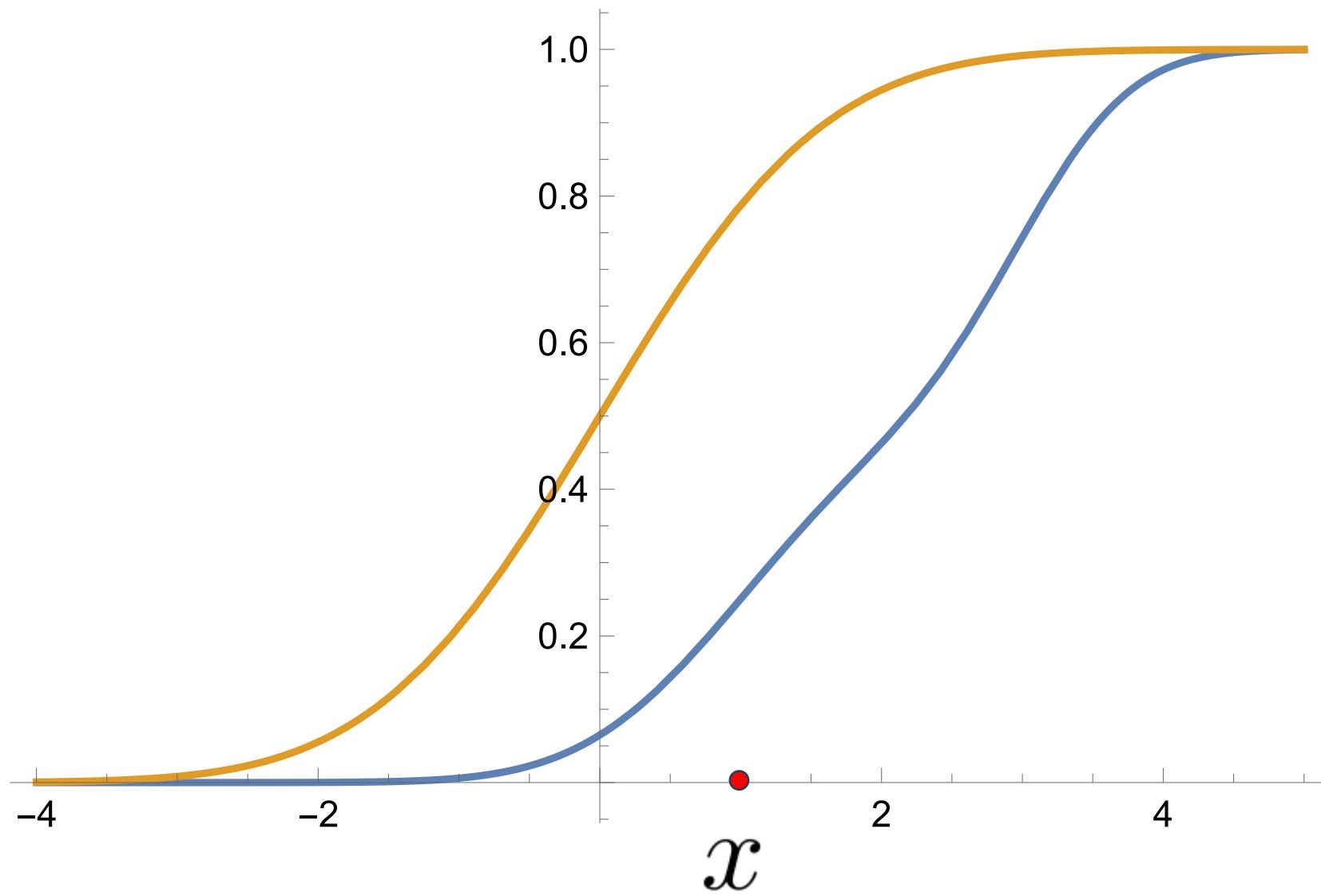


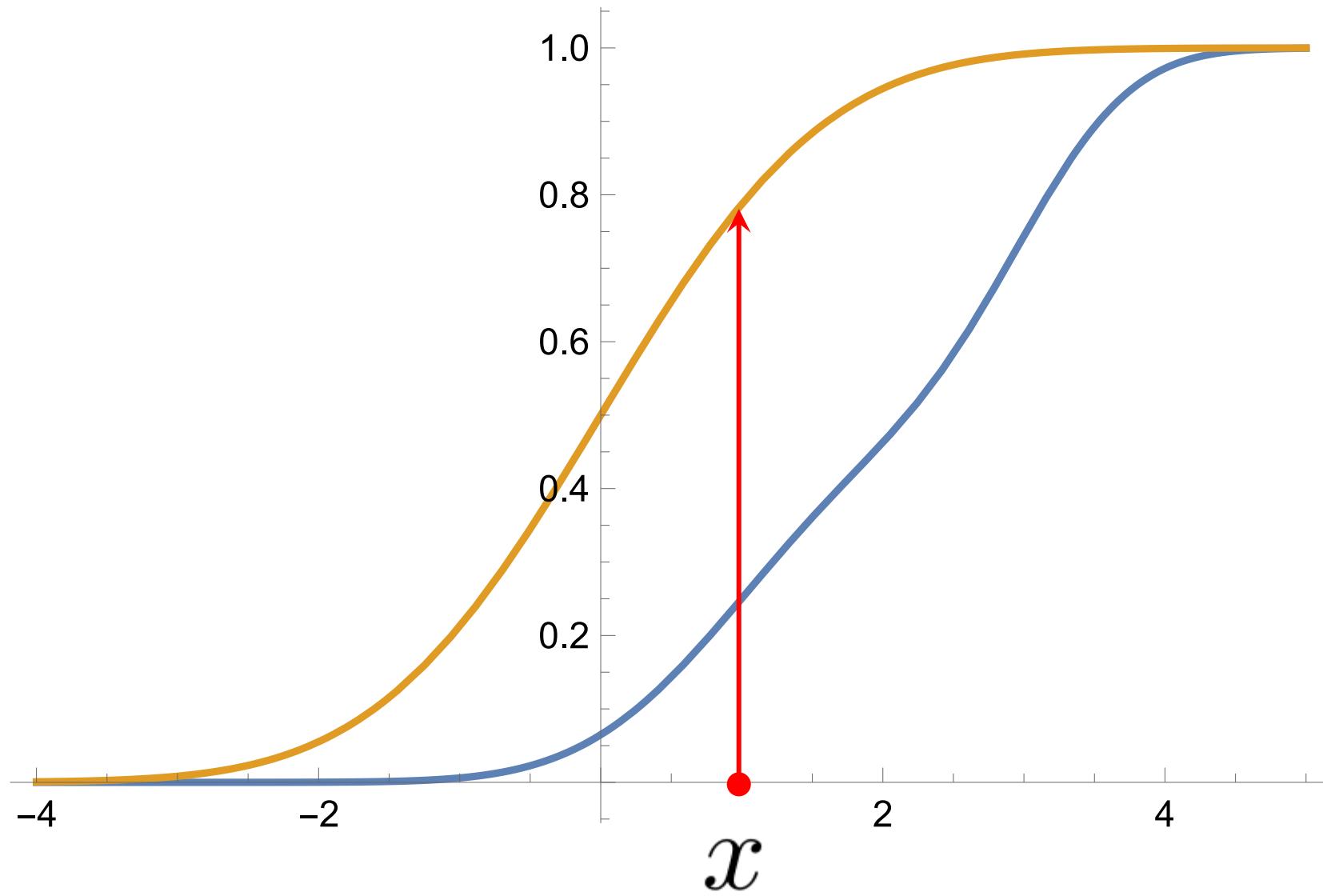
$x \sim X$

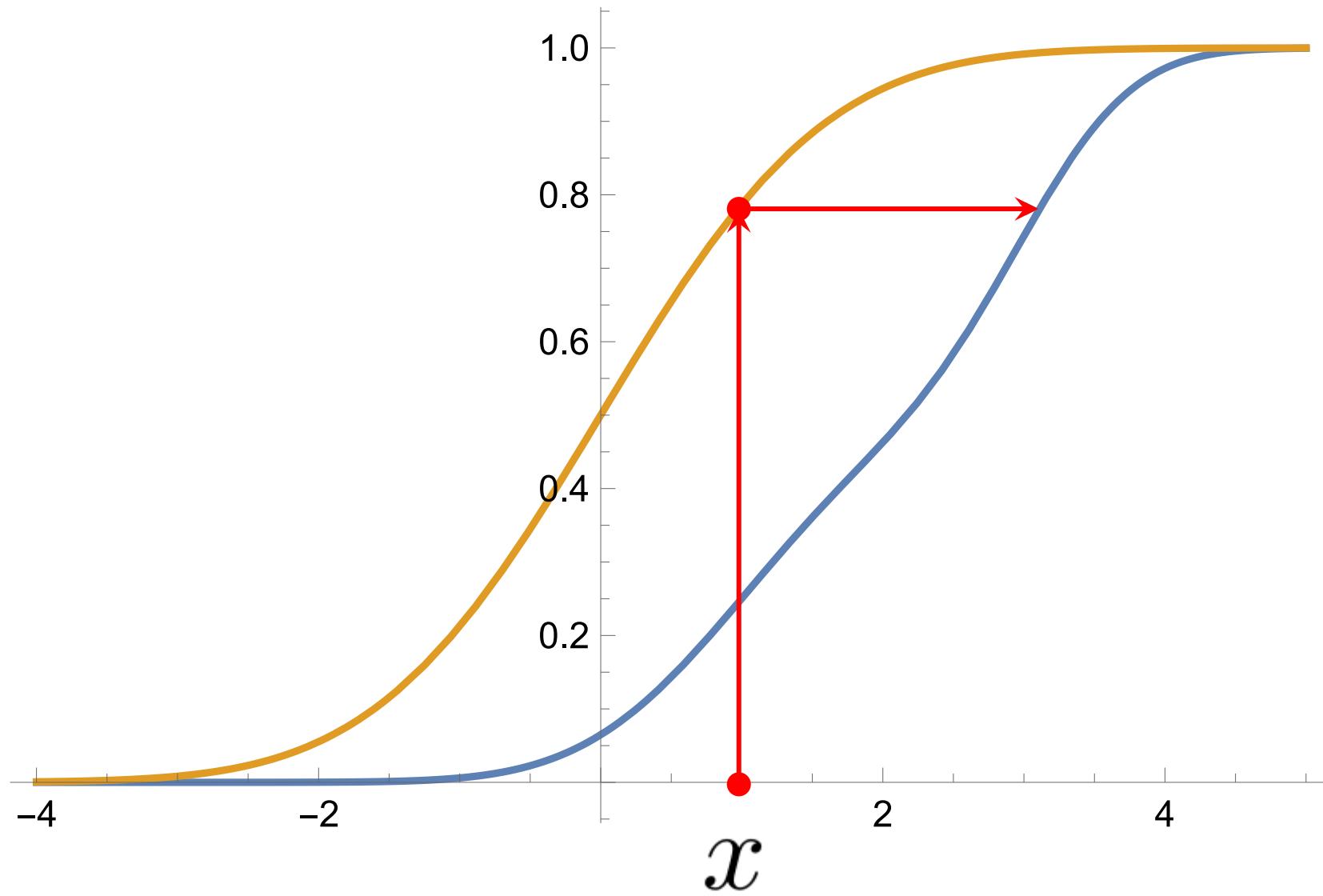
y is a sample from $f(X)$.

$y = f(x)$

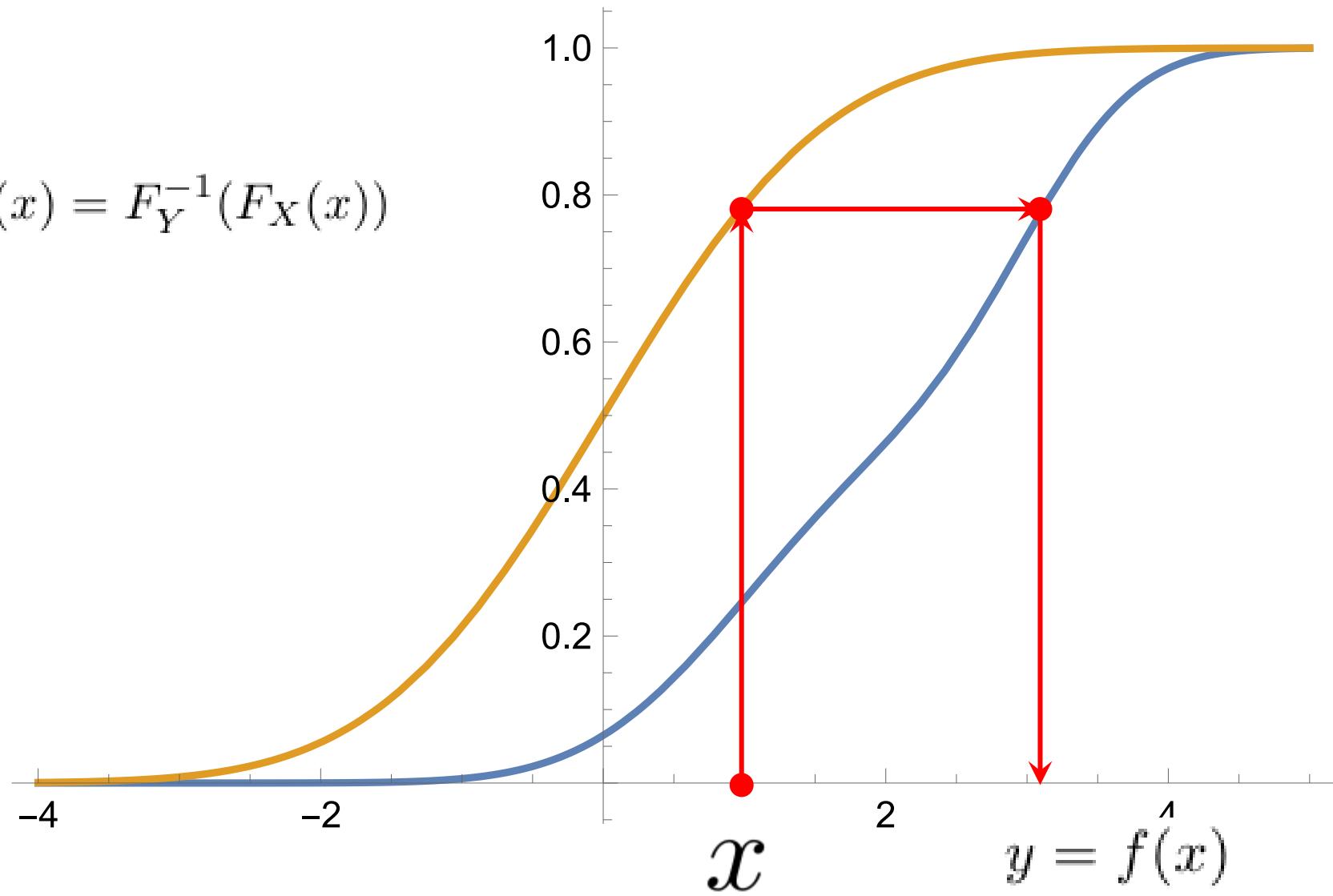








$$y = F_Y^{-1} \circ F(x) = F_Y^{-1}(F_X(x))$$

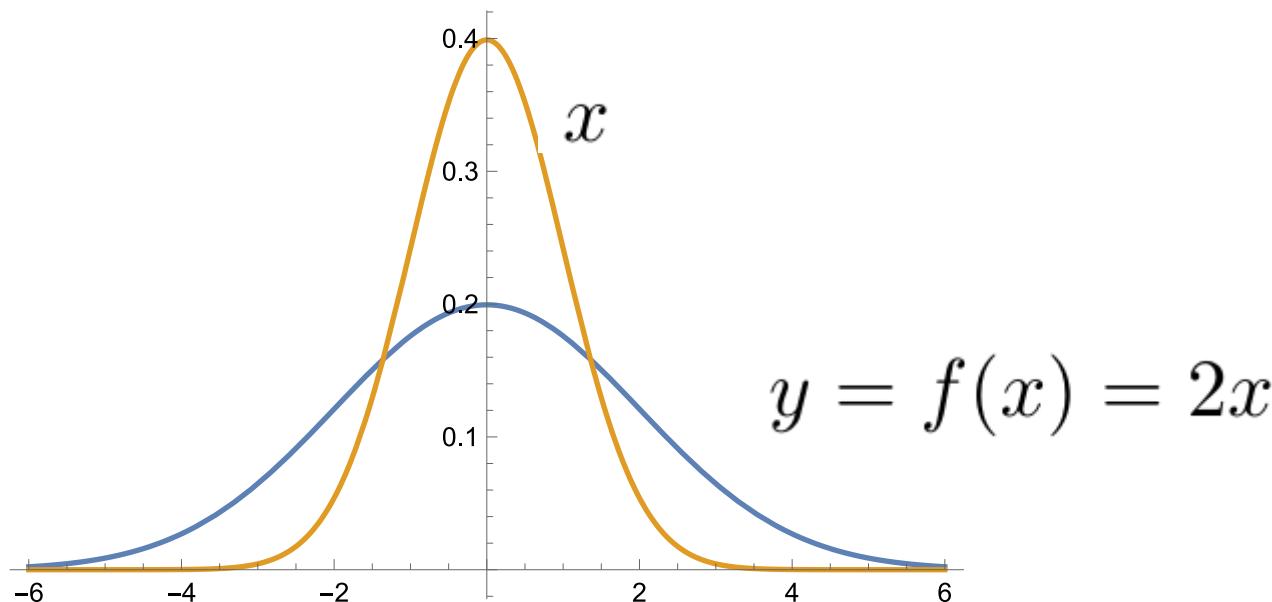


Probability transfer formula (Change of Variables formula)

$$p(y) = p(x)/|f'(x)|$$

In higher dimensions:

$$p(y) = p(x)/|\det J_f(x)|$$



What if we do not know the distribution Y ?

- Suppose we can sample from Y . We can approximate the distribution Y by taking samples $y \sim Y$.

Find the function $g_\theta : Y \rightarrow X$ that makes the samples most likely.

Given a function $g_\theta : y \mapsto x$, where y is sampled from an unknown distribution Y , and X is a Gaussian distribution

Given a function $g_\theta : y \mapsto x$, where y is sampled from an unknown distribution Y , and X is a Gaussian distribution

$$p_\theta(y_i) = p(g_\theta(y_i) | \det J_{g_\theta}(y_i)|$$

$$p_\theta(y_1 \dots y_n) = \prod_{i=1}^n p(g_\theta(y_i) | \det J_{g_\theta}(y_i)|$$

$$-\log p_\theta(y_1 \dots y_n) = \log \prod_{i=1}^n p(g_\theta(y_i)) | \det J_{g_\theta}(y_i)|$$

$$= -\sum_{i=1}^N \log p(g_\theta(y_i)) + \log |\det J_{g_\theta}(y_i)|$$

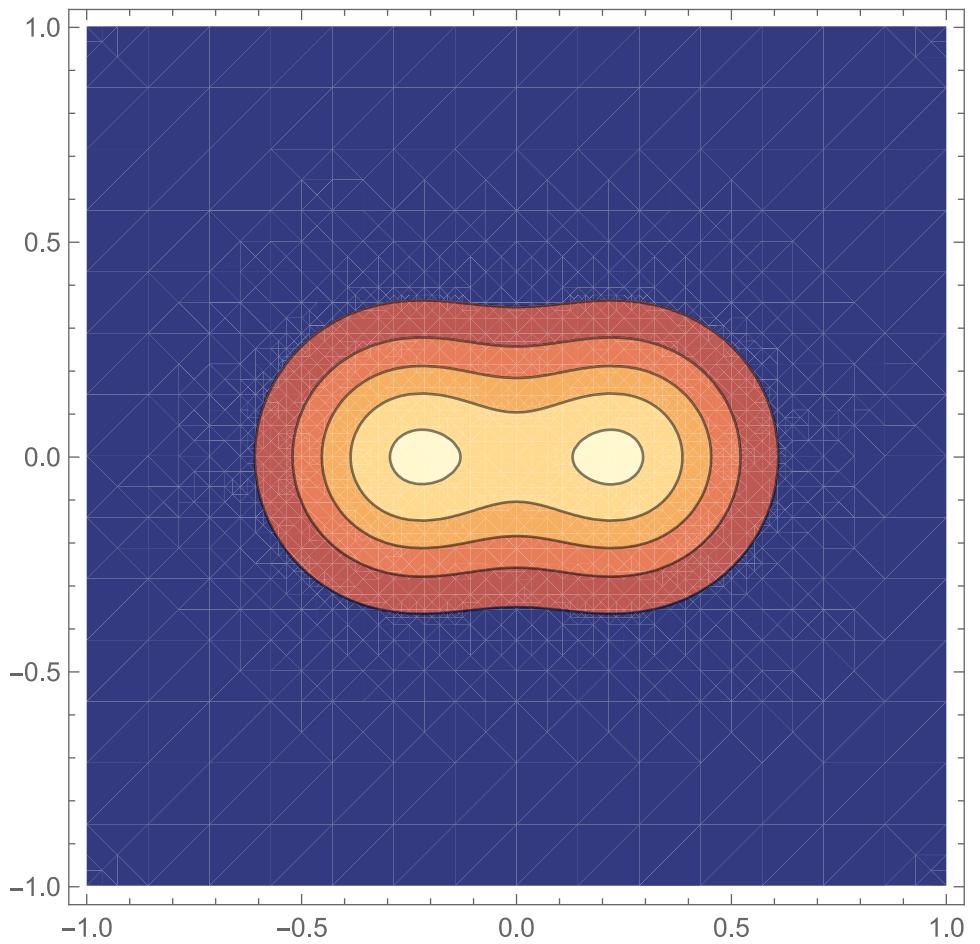
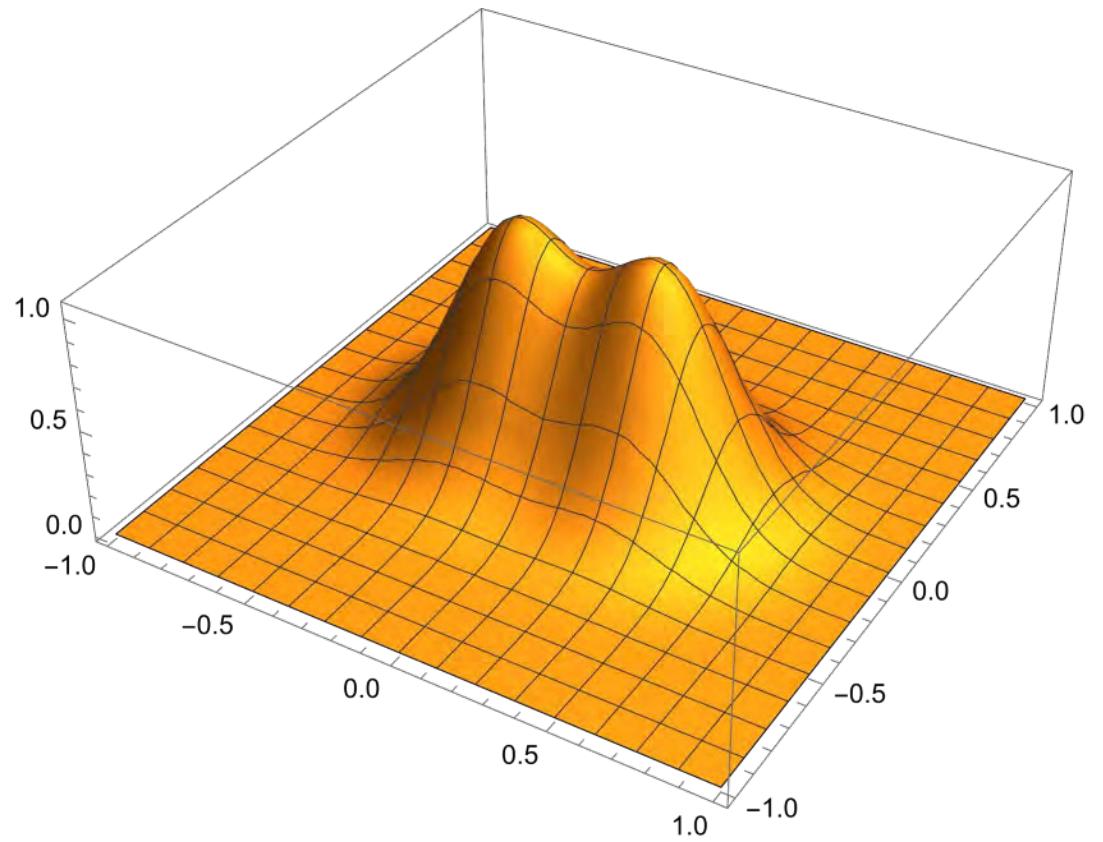
$$= \sum_{i=1}^N \|g_\theta(y_i)\|^2 / 2 - \log |\det J_{g_\theta}(y_i)|$$

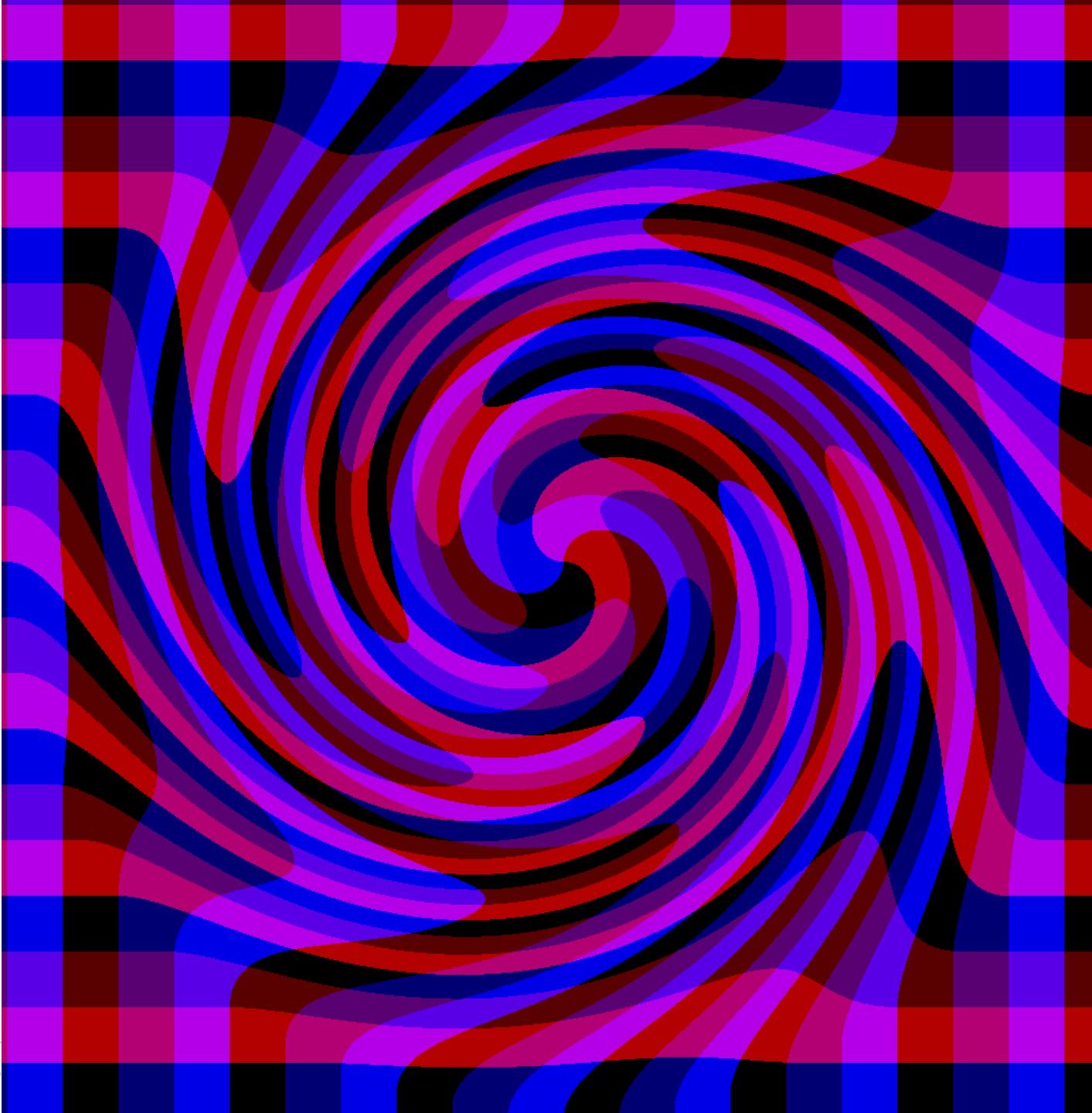


Minimize this over choices of θ , (the Negative Log Likelihood).

A tall, dark grey or black ladder stands vertically on a light brown, textured ground. It leans against a vertical blue wall that runs parallel to the ladder's length. The background features rolling hills under a bright blue sky with scattered white clouds.

Higher
Dimension





Normalizing Flow

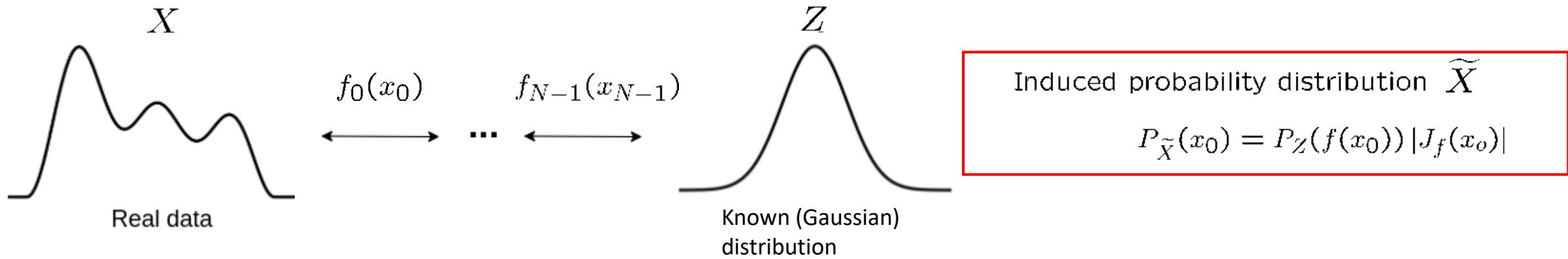


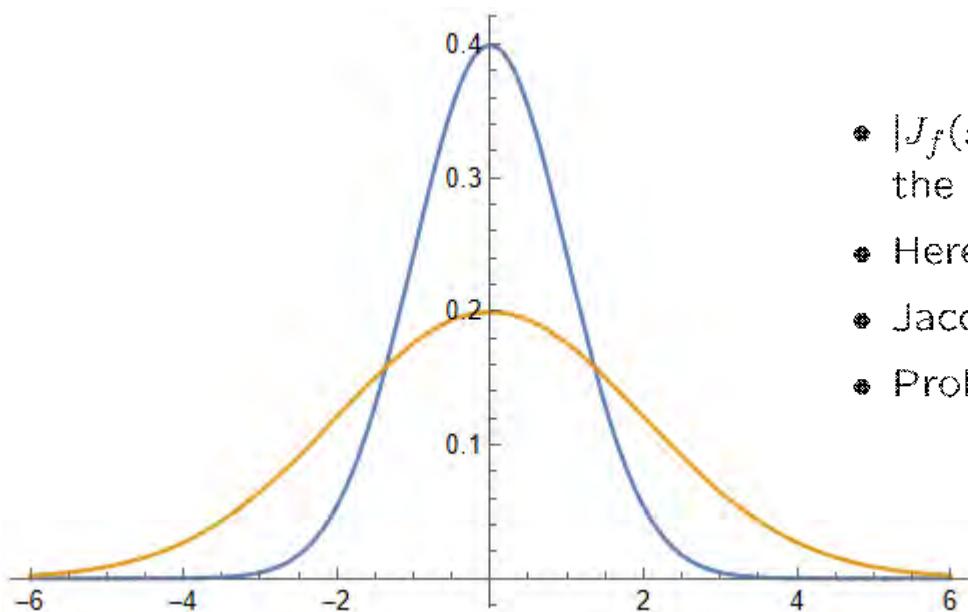
Figure adapted from deepai.org: Normalizing flow for deep anomaly detection: Ryzhikov et al.

Change of Variables Formula

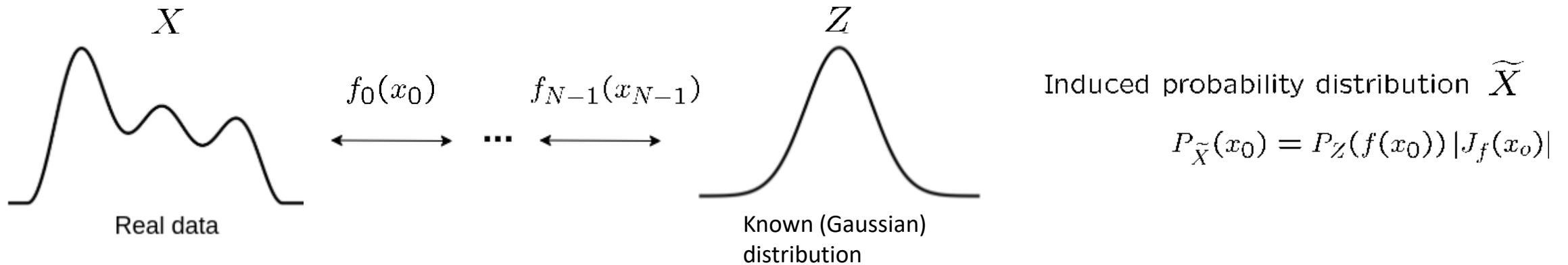
Theorem 1.1. Let X be random variables defined on $\mathcal{X} = \mathbb{R}^n$, and let $f : \mathcal{X} \rightarrow \mathcal{X}$ be invertible and differentiable. Define $Z = f(X)$. Then for $x \in \mathcal{X}$ and

$$x \xrightarrow{f} z$$

$$P_Z(z) = \frac{P_X(x)}{|J_f(x)|} .$$



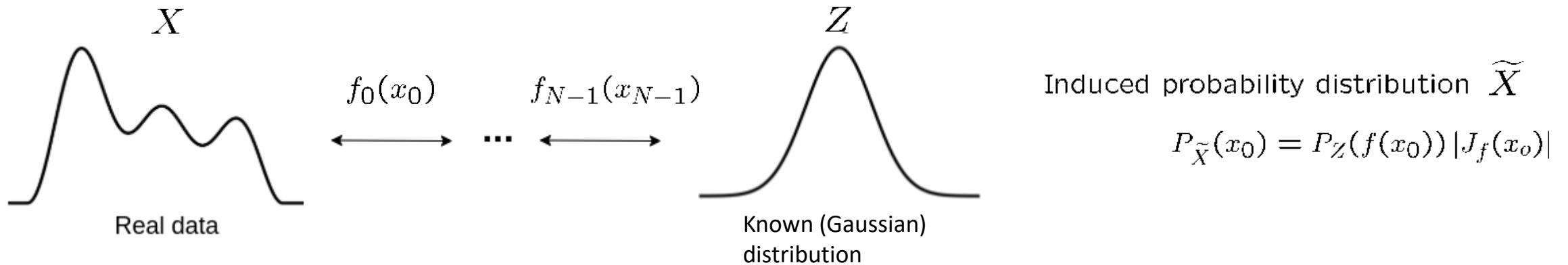
- $|J_f(x)|$ is the absolute value of the (determinant of) the Jacobian.
- Here, $z = f(x) = 2x$.
- Jacobian $J_f(x) = 2$
- Probability is divided by $2 = |J_f(x)|$



Idea behind Normalizing Flow

- Train function f to minimize

$$KL(X||\tilde{X}) = - \sum_{i=1}^n \log(P_Z(f(x_i)) |J_f(x_i)|)$$



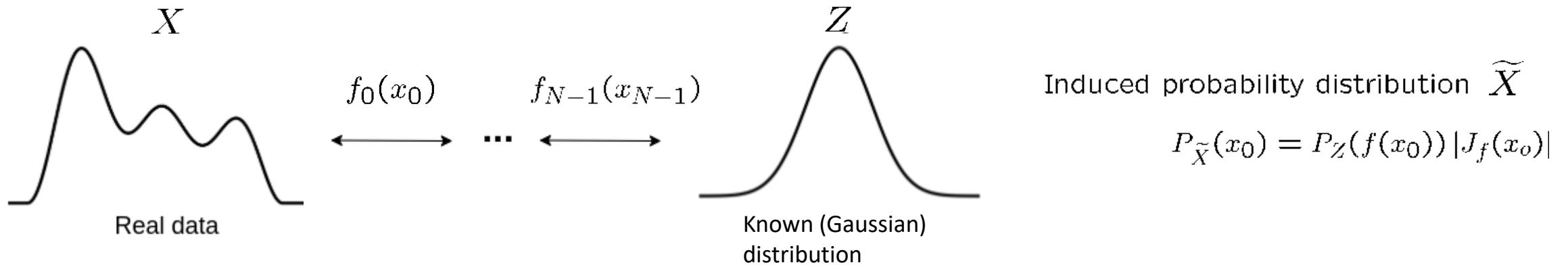
Idea behind Normalizing Flow

- Train function f to minimize

$$KL(X||\tilde{X}) = - \sum_{i=1}^n \log(P_Z(f(x_i)) |J_f(x_i)|)$$

- Use \tilde{X} as a proxy for X – allows one to estimate $P_X(x)$.
- Obtain samples from X by sampling $z \sim Z$, and

$$x = f^{-1}(z)$$



Idea behind Normalizing Flow

- Train function f to minimize

$$KL(X||\tilde{X}) = - \sum_{i=1}^n \log(P_Z(f(x_i)) |J_f(x_i)|)$$

- Use \tilde{X} as a proxy for X – allows one to estimate $P_X(x)$.
- Obtain samples from X by sampling $z \sim Z$, and

$$x = f^{-1}(z)$$

Purposes of Normalized Flow

- Estimate the probability of a given sample.
- Generative model: generate samples drawn from the data distribution X .

Requirements

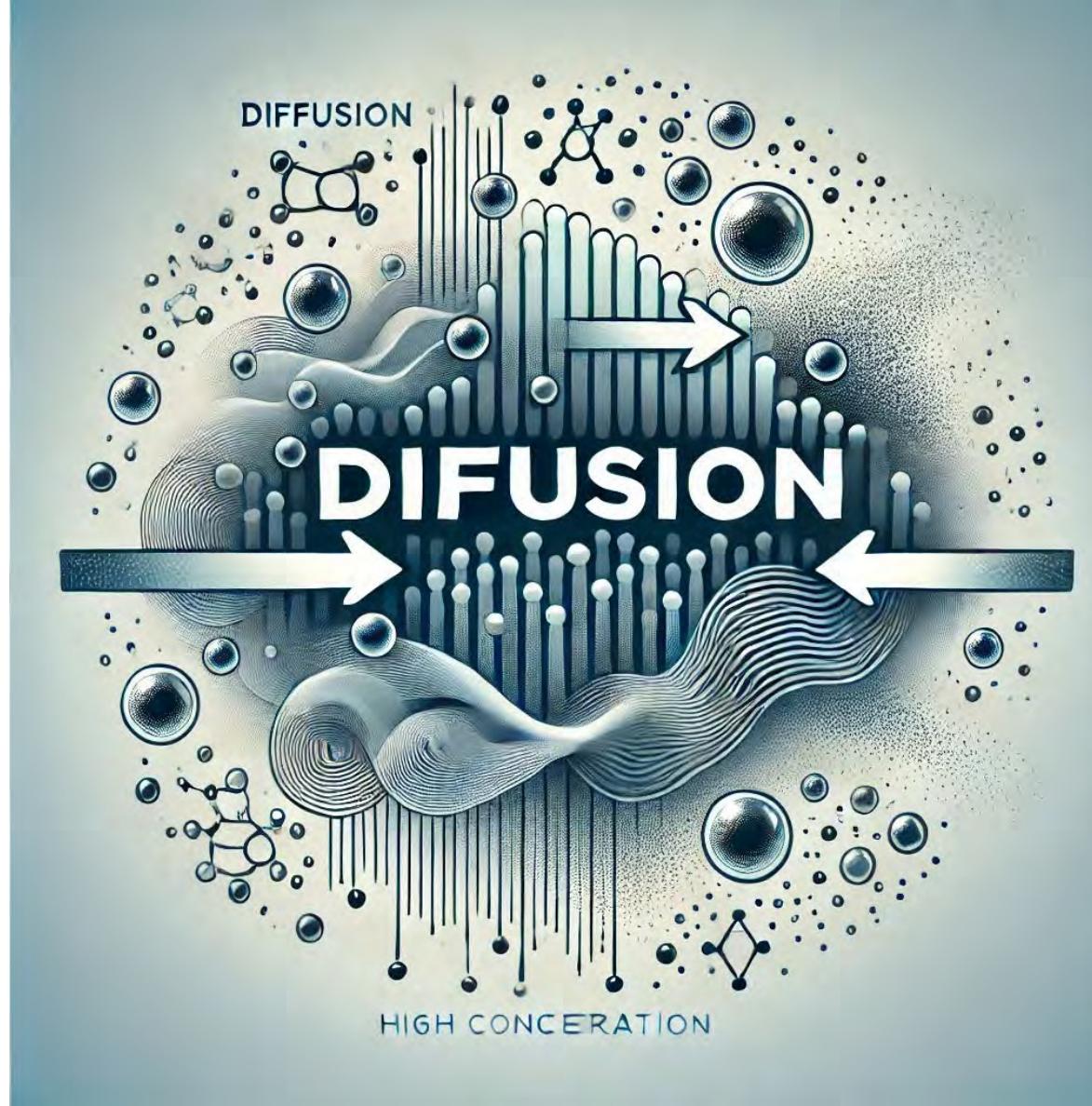
- Compute Jacobian $\det J_f(x)$ – needed during training and to estimate probability.
- Compute $f^{-1}(z)$ – needed for generative model use. Not needed during training.

Consider Conditional Coupling. In normalized flow, the forward may f_θ suppose that \mathbf{y} is input and \mathbf{z} is output of a conditional coupling layer. Then the forward mapping is

$$\begin{aligned}\mathbf{z}_0 &= \mathbf{s}_0 \mathbf{y}_0 \\ \mathbf{z}_1 &= \mathbf{A}(\mathbf{y}_0) + \mathbf{s}_1(\mathbf{y}_0) \mathbf{y}_1\end{aligned}\tag{16}$$

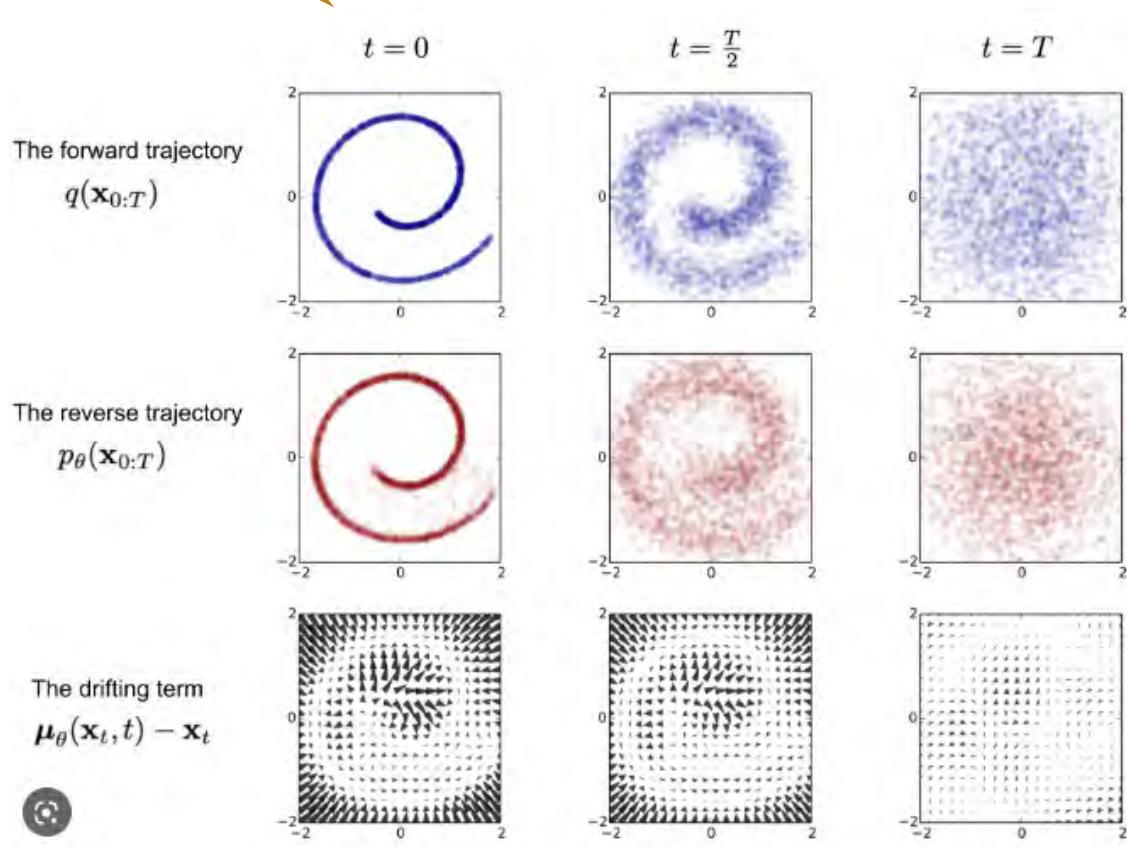
$$(y_0, y_1) \begin{array}{|c|c|} \hline & S_0 & \text{---} \\ \hline A(y_0) & S_1(y_0) & = (z_0, z_1) \\ \hline \end{array}$$

Diffusion





Forward
Backward



<https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>



Diffusion - backward and forward processes

- Diffusion consists of a forward and a backward stochastic process, starting with a sample x_0 .
- Forward process (noising)

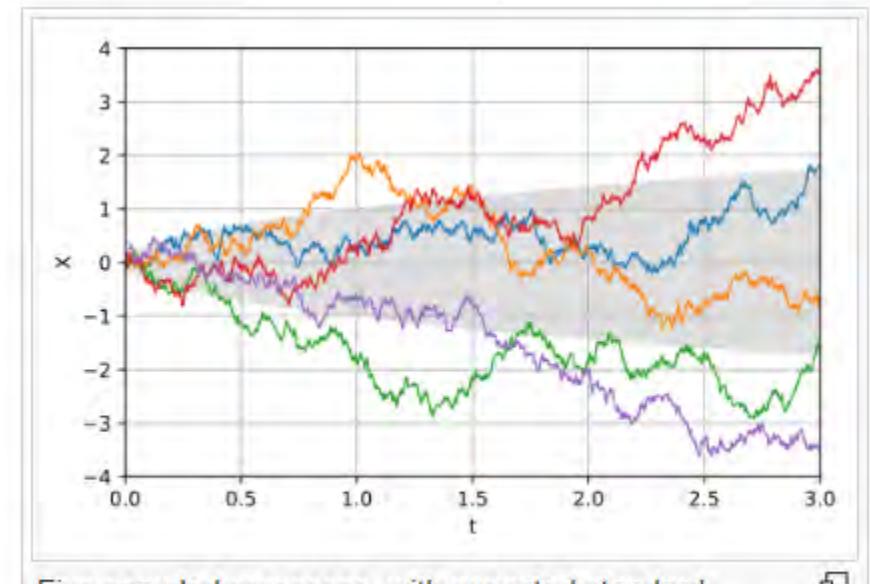
$$\mathbf{x}_t = \alpha_t \mathbf{x}_{t-1} + \beta_t \epsilon$$

where $\alpha_t < 1$ and β_t is small.

- Backward process (denoising)

$$\mathbf{x}_{t-1} = \mu_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t) + \beta_t \epsilon$$

- Can be made continuous as a so-called Wiener process.
- Reverse process worked out by Brian Anderson (ANU) in 1982 paper.
Solution involves solving a Stochastic Differential Equation.



From Wikipedia – Wiener process



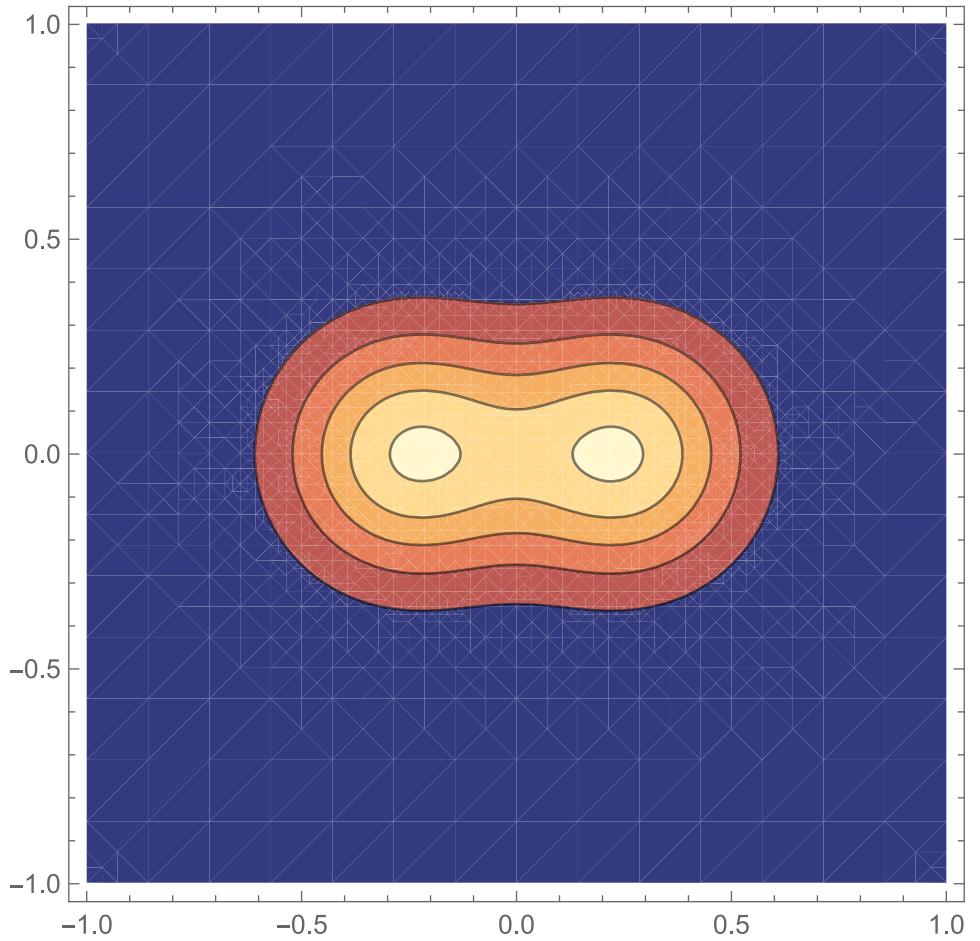
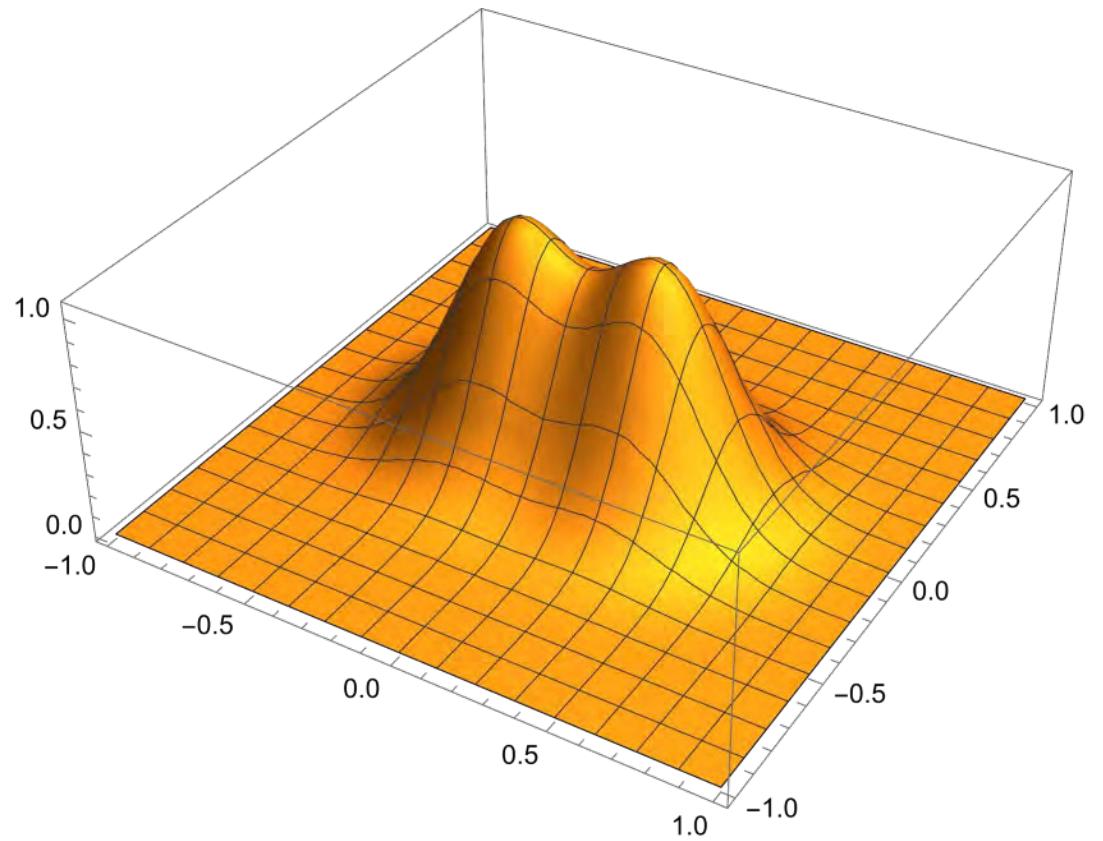


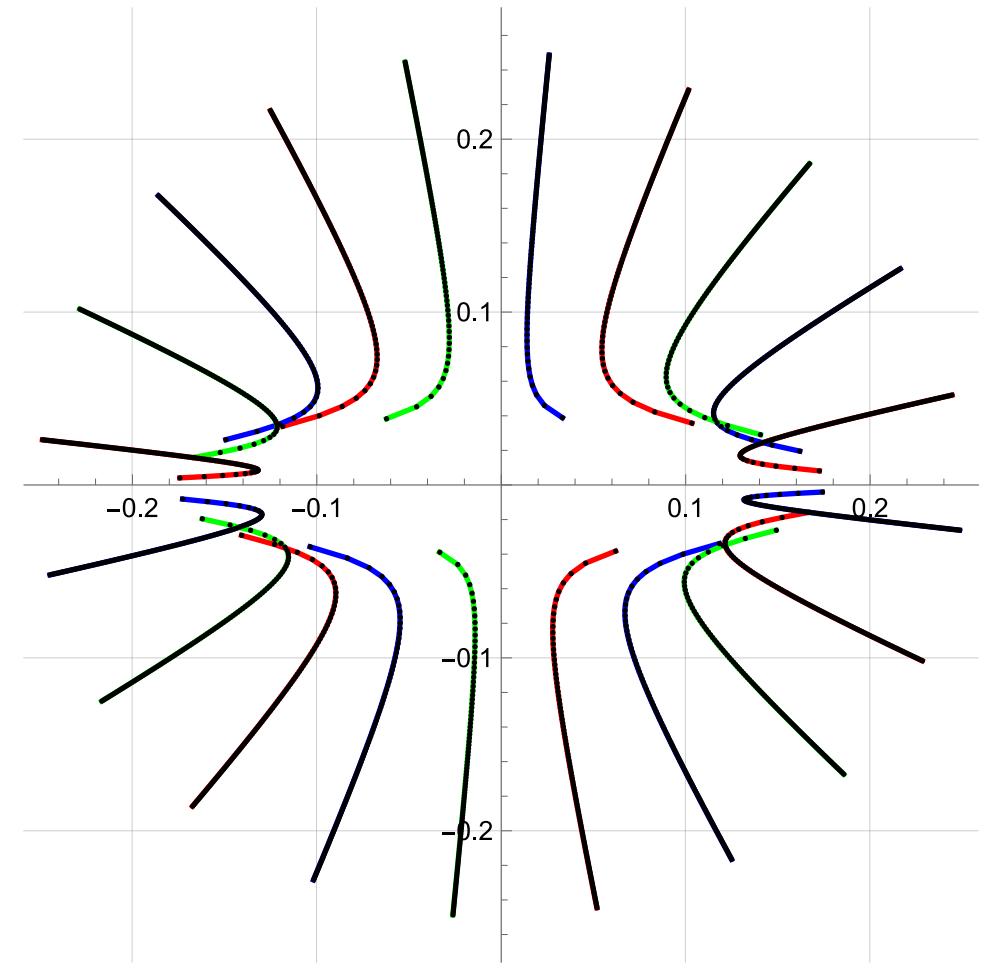
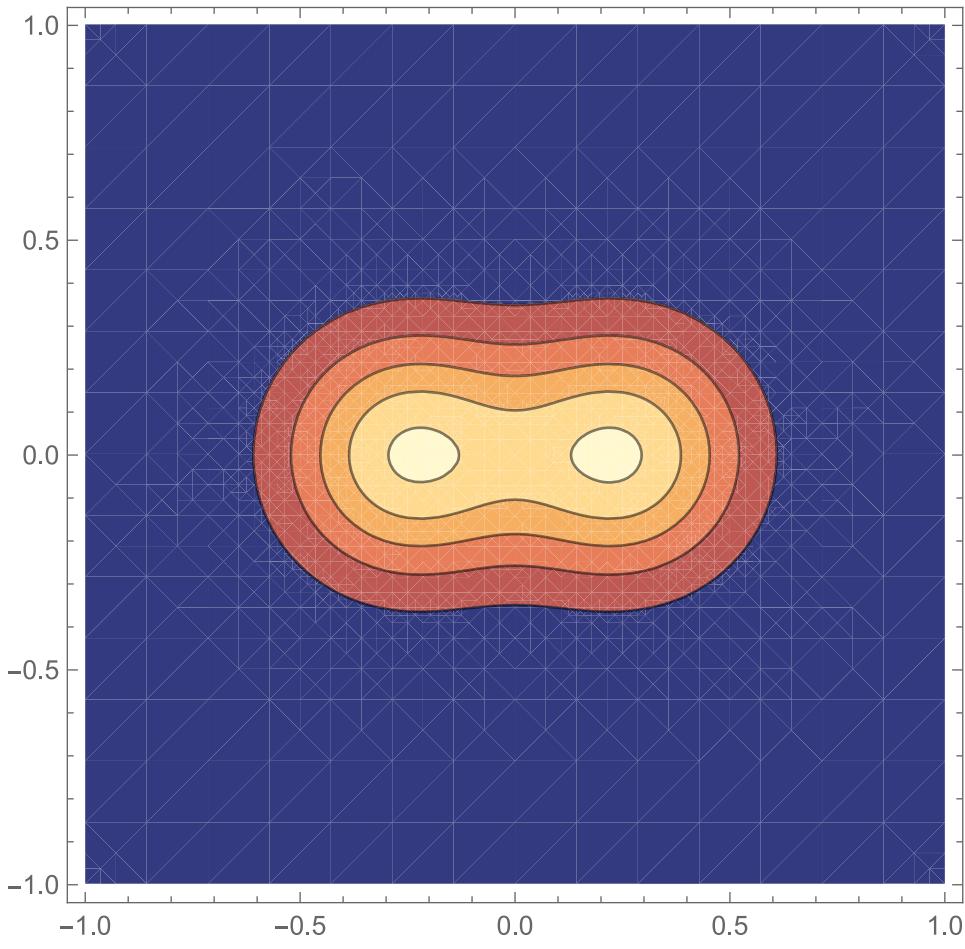


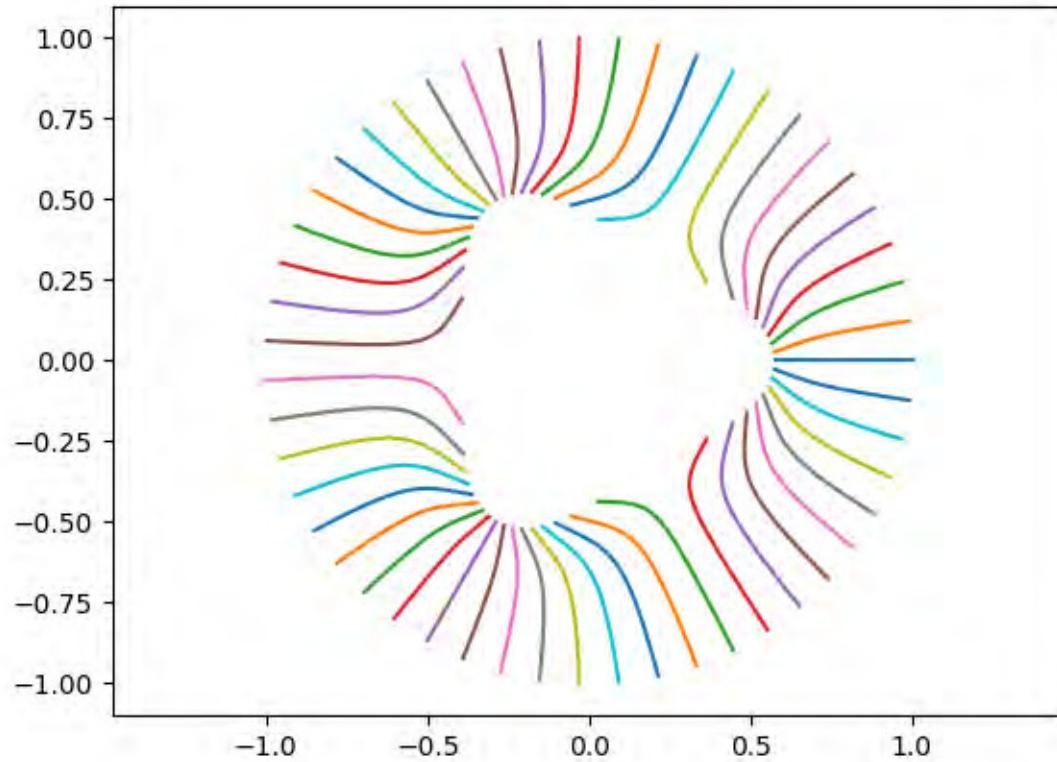
The solution











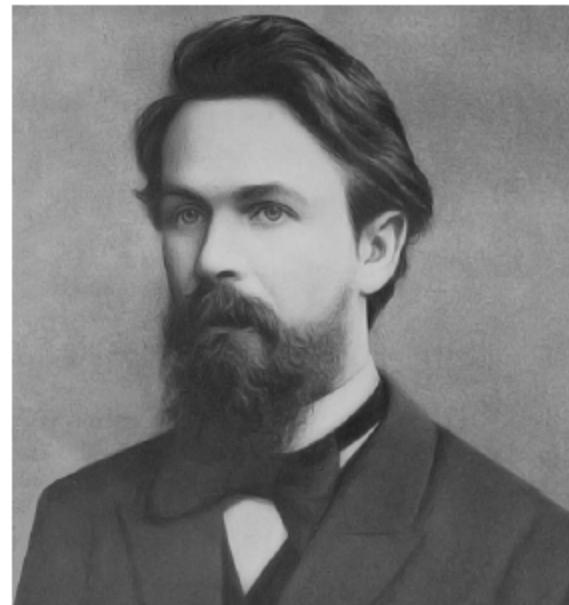


Figure 1: *Andrej Markov (1856 – 1922) – image from Wikimedai commons.*

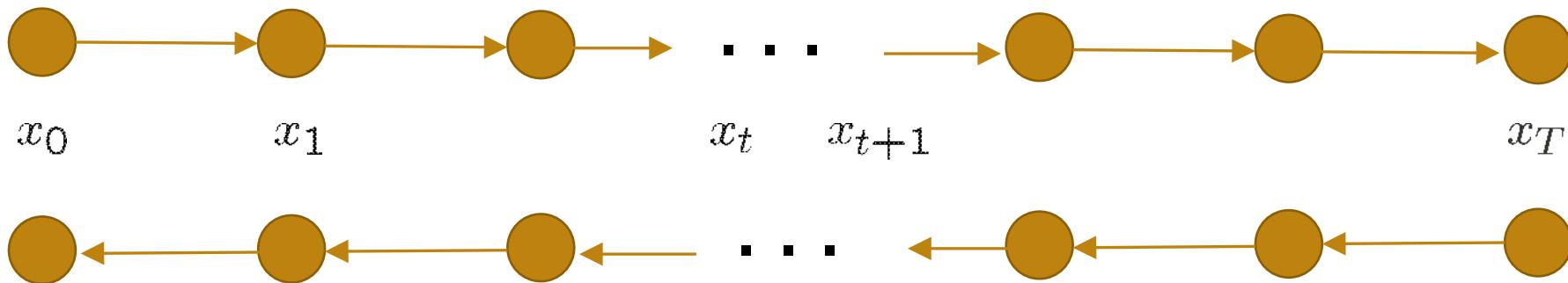


Markov Chains

Given any set of random variables X_0, \dots, X_T associated with a Markov Chain, the joint probability distribution q can be written

$$q(\mathbf{x}_{0:T}) = q(\mathbf{x}_T | \mathbf{x}_{T-1})q(\mathbf{x}_{T-1} | \mathbf{x}_{T-2}) \dots q(\mathbf{x}_1 | \mathbf{x}_0)q(\mathbf{x}_0)$$

Does a Markov Chain have a reverse Markov Chain



Inversion of a Markov chain. The formula (3) defines a joint probability distribution q on all the random variables \mathbf{x}_t for $t = 0, \dots, T$. Using Bayes' Law, we can also write (3) as

$$\begin{aligned} q(\mathbf{x}_{0:T}) &= q(\mathbf{x}_0) \prod_{t=1}^T \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t)q(\mathbf{x}_t)}{q(\mathbf{x}_{t-1})} \\ &= q(\mathbf{x}_0) \frac{q(\mathbf{x}_0|\mathbf{x}_1)q(\mathbf{x}_1)}{q(\mathbf{x}_0)} \frac{q(\mathbf{x}_1|\mathbf{x}_2)q(\mathbf{x}_2)}{q(\mathbf{x}_1)} \dots \frac{q(\mathbf{x}_{T-1}|\mathbf{x}_T)q(\mathbf{x}_T)}{q(\mathbf{x}_{T-1})} \end{aligned} \tag{4}$$



Inversion of a Markov chain. The formula (3) defines a joint probability distribution q on all the random variables \mathbf{x}_t for $t = 0, \dots, T$. Using Bayes' Law, we can also write (3) as

$$\begin{aligned} q(\mathbf{x}_{0:T}) &= q(\mathbf{x}_0) \prod_{t=1}^T \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t)q(\mathbf{x}_t)}{q(\mathbf{x}_{t-1})} \\ &= q(\cancel{\mathbf{x}_0}) \frac{q(\mathbf{x}_0|\mathbf{x}_1)q(\mathbf{x}_1)}{q(\cancel{\mathbf{x}_0})} \frac{q(\mathbf{x}_1|\mathbf{x}_2)q(\mathbf{x}_2)}{q(\cancel{\mathbf{x}_1})} \dots \frac{q(\mathbf{x}_{T-1}|\mathbf{x}_T)q(\mathbf{x}_T)}{q(\cancel{\mathbf{x}_{T-1}})} \end{aligned} \quad (4)$$

This product telescopes by cancellation of the factors $q(\mathbf{x}_t)$, resulting in

$$q(\mathbf{x}_{0:T}) = q(\mathbf{x}_T) \prod_{t=1}^T q(\mathbf{x}_{t-1}|\mathbf{x}_t) . \quad (5)$$



Computing probabilities of samples, after training

Write probabilities in two ways

$$\begin{aligned} q(\mathbf{x}_{0:T}) &= q(\mathbf{x}_T) \prod_{t=1}^T q(\mathbf{x}_{t-1} | \mathbf{x}_t) \\ &= q(\mathbf{x}_0) \prod_{t=1}^T q(\mathbf{x}_t | \mathbf{x}_{t-1}) . \end{aligned}$$

So

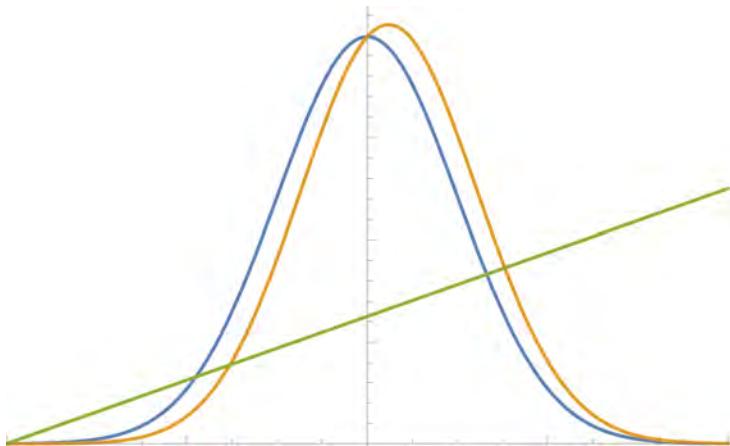
$$q(\mathbf{x}_0) = q(\mathbf{x}_T) \prod_{t=1}^T \frac{q(\mathbf{x}_{t-1} | \mathbf{x}_t)}{q(\mathbf{x}_t | \mathbf{x}_{t-1})} .$$

More exactly

$$q(\mathbf{x}_0) = q(\mathbf{x}_T) \prod_{t=1}^T \frac{p_\theta(\mathbf{x}'_{t-1} | \mathbf{x}_t)}{q(\mathbf{x}_t | \mathbf{x}'_{t-1})} .$$



Form of the backward conditional probabilities



Bayes Rule:

$$p(x_t|x_{t-1}) = G_\beta(x_t; \alpha x_{t-1}, \beta)$$

$$p(x_{t-1}|x_t) = \frac{p(x_t|x_{t-1})p(x_{t-1})}{p(x_t)}$$

$$= \frac{G_\beta(x_t; \alpha x_{t-1}, \beta)p(x_{t-1})}{p(x_t)}$$

- $p(x_{t-1}|x_t)$ is (almost) a Gaussian with the same variance as $p(x_t|x_{t-1})$.
- Mean is shifted in the direction of the gradient of $p(x_t)$.



Form of inverse conditional probabilities

This is the approximation to the probabilities when $q(\mathbf{x}_t)$ is linear in the region of support of $q(\mathbf{x}_t|\mathbf{x}_{t-1})$.

$$\mu(X_{t-1}|\mathbf{x}_t) = (\mathbf{x}_t + \beta_t^2 \mathbf{m})/\alpha_t$$



Mean learnt by training

$$\Sigma = \beta_t^2 (\mathbf{I} - \beta^2 \mathbf{m} \mathbf{m}^\top)/\alpha_t^2 .$$



Variance approximately same as forward variance.

and covariance matrix
where

$$\mathbf{m} = \nabla \log p(\mathbf{x}_{t-1} | \mathbf{x}_t).$$

Bonus:

If we compute $\mu(X_{t-1}|\mathbf{x}_t)$, then we know

$$\nabla \log p(\mathbf{x}_{t-1} | \mathbf{x}_t) \equiv \mu(X_{t-1}|\mathbf{x}_t) - \mathbf{x}_t/\alpha_t .$$



MACHINE LEARNING

FOR COMPUTER VISION



FOR COMPUTER VISION

Finding the backward conditionals by training.

As long as the variances β_t^2 are small, a reasonable approximation is that the backward conditionals are normal distributions, and that they have the same variance as the forward conditionals. In this case, one can assume a form

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_{\theta,t}(\mathbf{x}_t), \beta_t^2 \mathbf{I}) \quad (18)$$

and the goal of training is to compute the means

$$\mu_{\theta,t}(\mathbf{x}_t) = \mathbb{E}_{\mathbf{x}_{t-1} \sim (X_{t-1}|\mathbf{x}_t)}[\mathbf{x}_{t-1}] .$$



Minimizing least-squared error computes the conditional

Theorem 2.2. Let X, Y be a pair of joint random variables. Let $f_*(\mathbf{y})$ be the function that minimizes the functional

$$L(f_\theta) = \mathbb{E}_{(\mathbf{x}, \mathbf{y}) \sim (X, Y)} [\|f_\theta(\mathbf{y}) - \mathbf{x}\|^2]$$

then

$$f_*(\mathbf{y}) = \mathbb{E}_{\mathbf{x} \sim X | \mathbf{y}} [\mathbf{x}] = \mu(X | \mathbf{y}) .$$

□

Proof. We summarize the proof given above. By definition,

$$\begin{aligned} f_* &= \arg \min_{f_\theta} \mathbb{E}_{(\mathbf{x}, \mathbf{y}) \sim (X, Y)} [\|f_\theta(\mathbf{y}) - \mathbf{x}\|^2] \\ &= \arg \min_{f_\theta} \mathbb{E}_{\mathbf{y} \sim Y} [\mathbb{E}_{\mathbf{x} \sim X | \mathbf{y}} [\|f_\theta(\mathbf{y}) - \mathbf{x}\|^2]] \end{aligned}$$

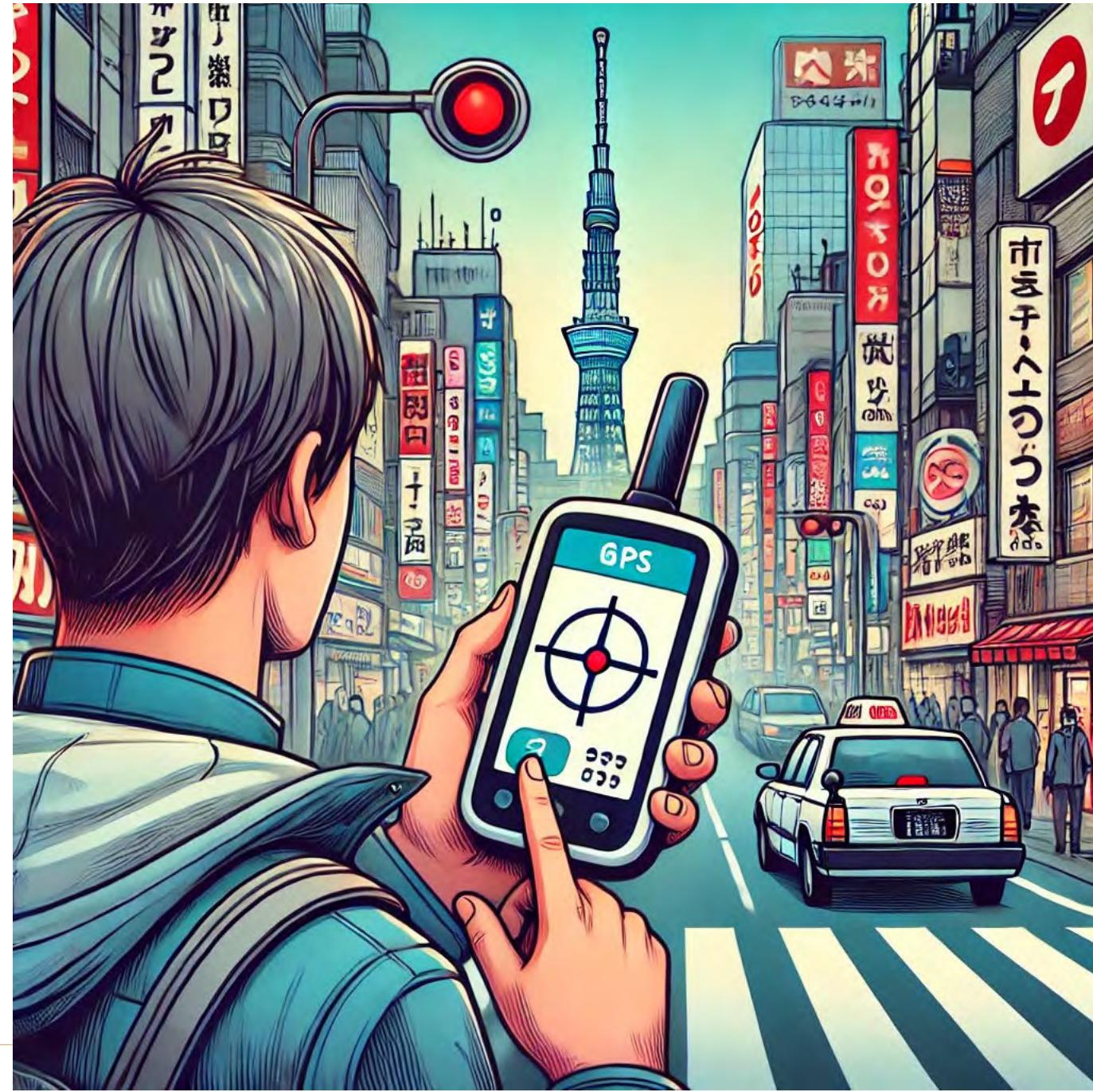
This minimum is attained by $f_*(\mathbf{y})$ defined independently for each \mathbf{y} by

$$\begin{aligned} f_*(\mathbf{x}) &= \arg \min_{\hat{\mathbf{x}}} \mathbb{E}_{\mathbf{x} \sim X | \mathbf{y}} [\|\hat{\mathbf{x}} - \mathbf{x}\|^2] \\ &= \mathbb{E}_{x \sim X | \mathbf{y}} [\mathbf{x}] . \end{aligned}$$

□



Trace-back



Focker-Planck Equations

Describe the evolution of the probability $p(x, t)$ of a point at time t , undergoing a diffusion process.

Pure diffusion

Lemma 3.3.14. *If $\{p_t\}$ are a Gaussian family of distributions, $p_t = G_{\beta(t)} * p_0$, defined on \mathbb{R}^n , then*

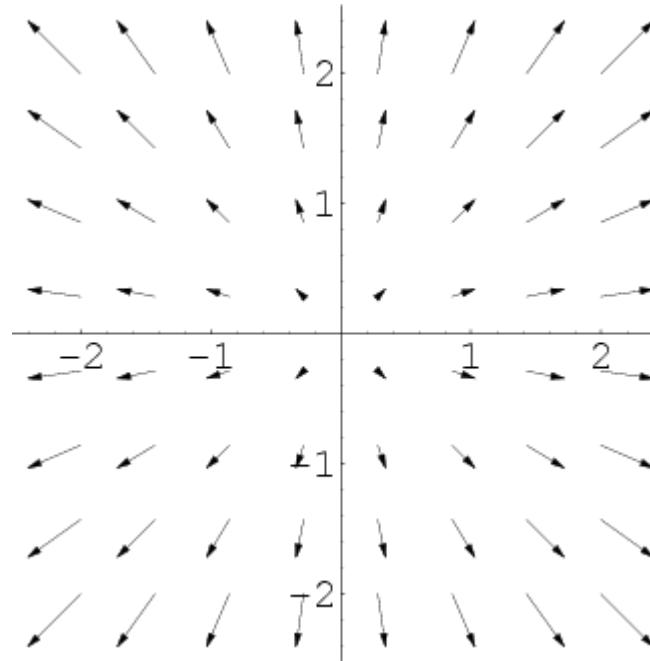
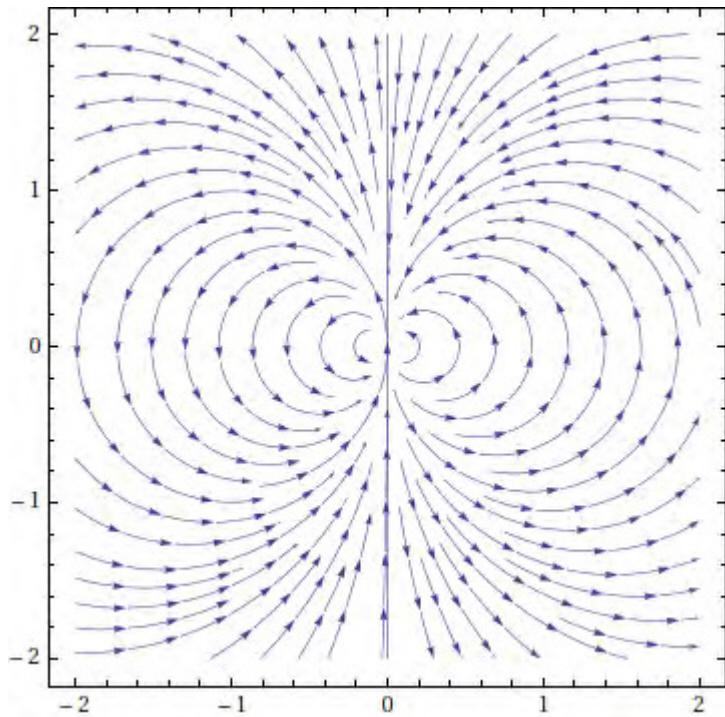
$$\frac{d}{dt} p_t(\mathbf{x}) = \beta'(t) \frac{\nabla^2 p_t(\mathbf{x})}{2} .$$



Drift but no diffusion

If a family of probability density functions $\{p_t\}$ are associated with drift $\alpha(\mathbf{x}, t)$

$$\frac{d}{dt} p_t(\mathbf{x}) = -\langle \nabla p_t(\mathbf{x}), \alpha(\mathbf{x}, t) \rangle - p_t(\mathbf{x}) \operatorname{div} \alpha(\mathbf{x}, t)$$



Diffusion with drift

Lemma 3.3.16. *If a family of probability density functions $\{p_t\}$ are associated with a Wiener process with drift $\alpha(\mathbf{x}, t)$ and noise schedule $\beta(t)$, then*

$$\frac{d}{dt} p_t(\mathbf{x}) = -\langle \nabla p_t(\mathbf{x}), \alpha(\mathbf{x}, t) \rangle - p_t(\mathbf{x}) \operatorname{div} \alpha(\mathbf{x}, t) + \frac{\beta(t)}{2} \nabla^2 p_t(\mathbf{x}) . \quad (3.11)$$



Connecting probability distribution.

Solution. We need to find a vector field $\mathbf{v}(\mathbf{x}, t)$ to satisfy the compatibility relation

$$\begin{aligned}-p_t(\mathbf{x}) \operatorname{div} \mathbf{v}(\mathbf{x}, t) &= \frac{d}{dt} p_t(\mathbf{x}(t), t) \\&= \langle \nabla p_t(\mathbf{x}), \mathbf{v}(\mathbf{x}, t) - \alpha(\mathbf{x}, t) \rangle - p_t(\mathbf{x}) \operatorname{div} \alpha(\mathbf{x}, t) + \frac{\beta(t)}{2} \nabla^2 p_t(\mathbf{x})\end{aligned}$$

solution is

$$\mathbf{v}(\mathbf{x}, t) = \alpha(\mathbf{x}, t) + \frac{\beta(t)}{2} \nabla \log p_t(\mathbf{x}) .$$

This shows the following result:

Lemma 3.3.18. *The process $\mathcal{W}[\alpha, \beta, p]$ is equivalent to the non-stochastic process*

$$\mathcal{W}[\alpha + \beta \nabla \log p / 2, 0, p]$$



Theorem 3.3.20 (Reversability of a Wiener process). *Wiener processes $\mathcal{W}[\alpha, \beta, p]$ and $\mathcal{W}[-\bar{\alpha} - \bar{\beta}\nabla \log \tilde{p}, \bar{\beta}, \tilde{p}]$ are inverses.*

Proof. We compute

$$\begin{aligned}\mathcal{W}[\alpha, \beta, p]^{-1} &\sim \mathcal{W}[\alpha + \beta\nabla \log p/2, 0, p]^{-1} \\ &\sim \mathcal{W}[-\bar{\alpha} - \bar{\beta}\nabla \log \tilde{p}/2, 0, \tilde{p}] \\ &\sim \mathcal{W}[-\bar{\alpha} - \bar{\beta}\nabla \log \tilde{p} + \bar{\beta}\nabla \log \tilde{p}/2, 0, \tilde{p}] \\ &\sim \mathcal{W}[-\bar{\alpha} - \bar{\beta}\nabla \log \tilde{p}, \bar{\beta}, \tilde{p}].\end{aligned}$$

□



The term "stochastic" originates from the Ancient Greek word στοχαστικός (*stokhastikós*), which means "skillful in aiming" or "pertaining to guessing." This, in turn, is derived from στοχάζομαι (*stokházomai*), meaning "to aim at a target" or "to guess," and στόχος (*stókhos*), meaning "an aim" or "a target."

[EN.WIKTIONARY.ORG](#)

In English, "stochastic" was first recorded in 1662, used as an adjective meaning "pertaining to conjecturing." Over time, its usage evolved, and by the 20th century, it came to describe processes involving [randomness or probability](#). For instance, in 1934, mathematician Joseph L. Doob used the term "stochastic process" to refer to a collection of random variables indexed by time or space.

[EN.WIKIPEDIA.ORG](#)

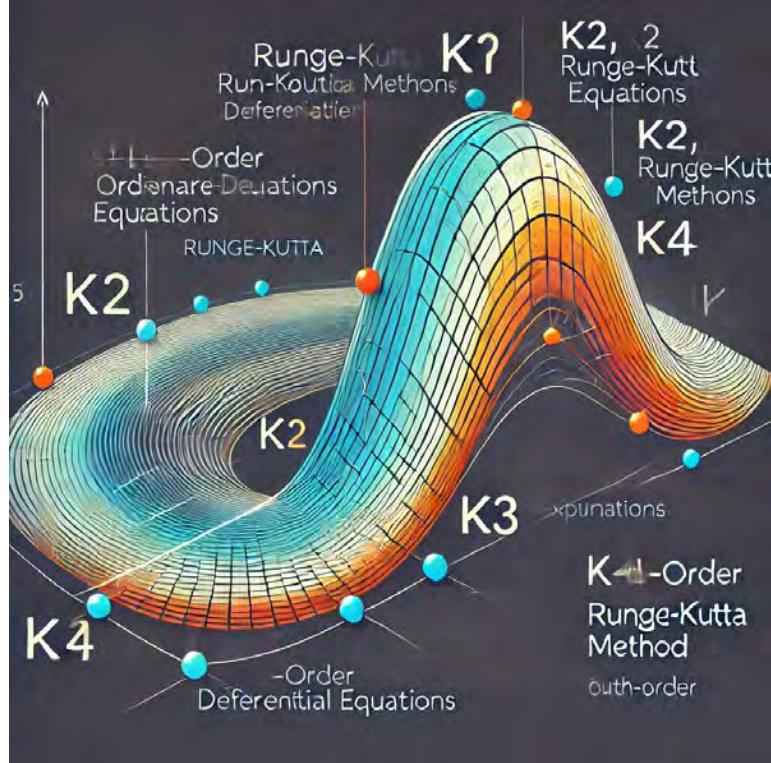
Today, "stochastic" is commonly used in fields such as mathematics, statistics, and physics to describe systems or processes that are inherently random or probabilistic in nature.

While "stochastic" is the primary English word directly connected to στόχος, other terms like "stich" and "stichos" are derived from related Greek roots such as στίχος (*stíkhos*), meaning "row" or "line," which is associated with the concept of alignment or order.

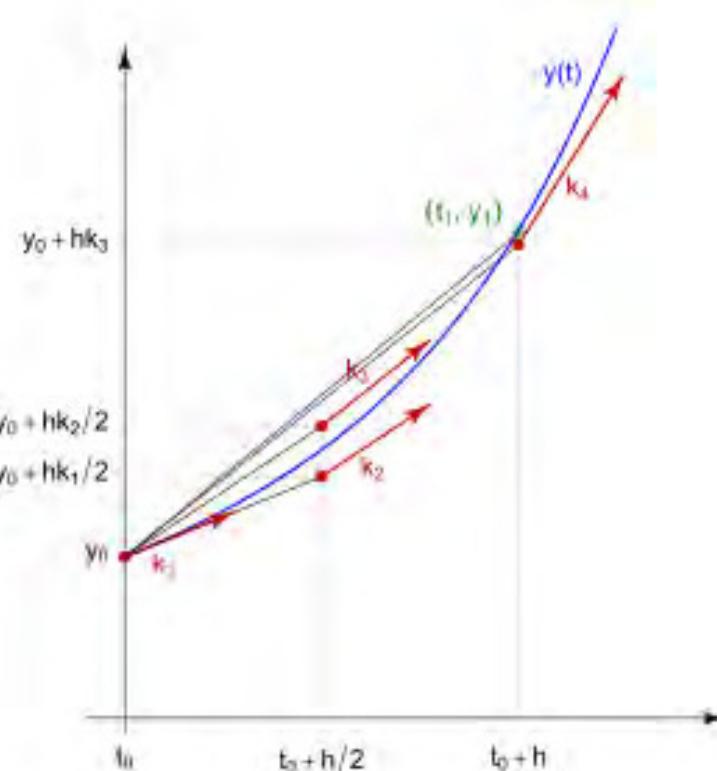
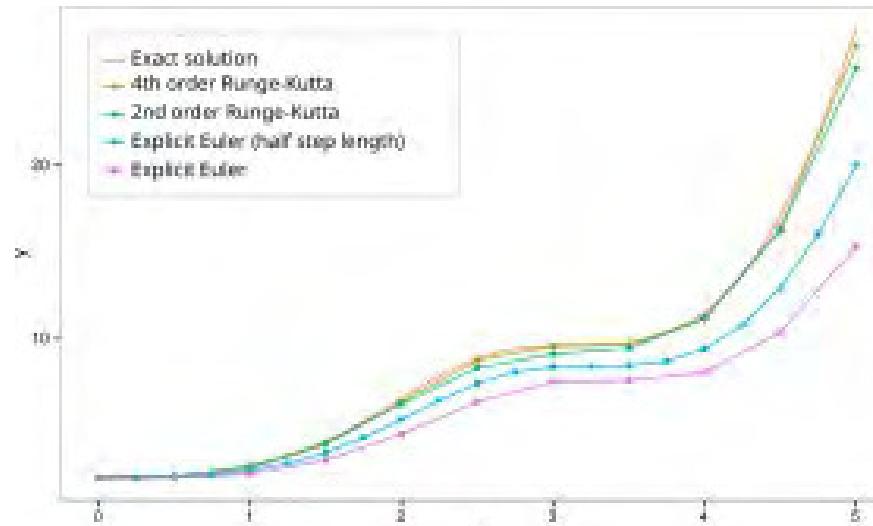
[EN.WIKTIONARY.ORG](#)

[EN.WIKTIONARY.ORG](#)





Runge-Kutta method



Now we pick a step-size $h > 0$ and define:

$$y_{n+1} = y_n + \frac{h}{6} (k_1 + 2k_2 + 2k_3 + k_4),$$

$$t_{n+1} = t_n + h$$

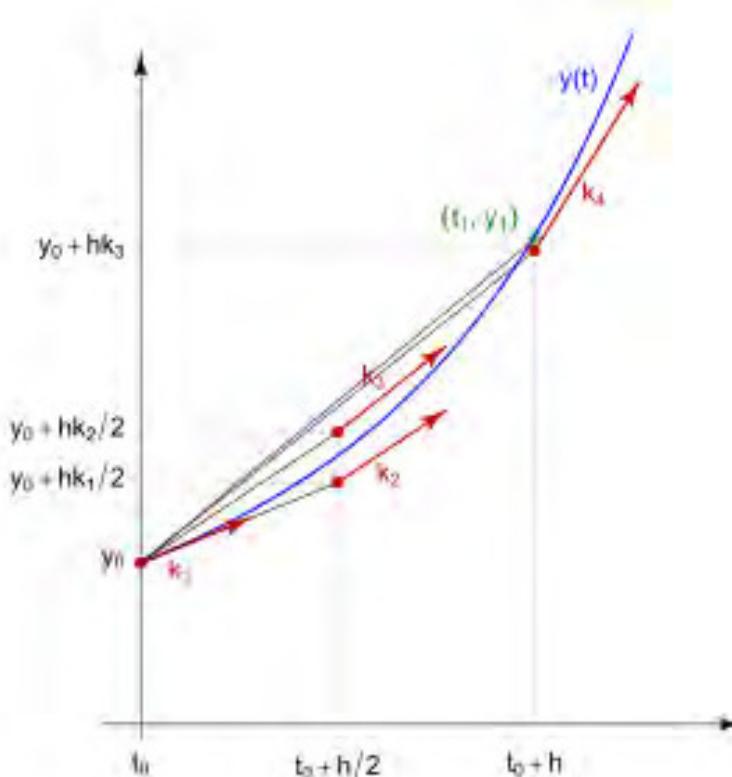
for $n = 0, 1, 2, 3, \dots$, using^[3]

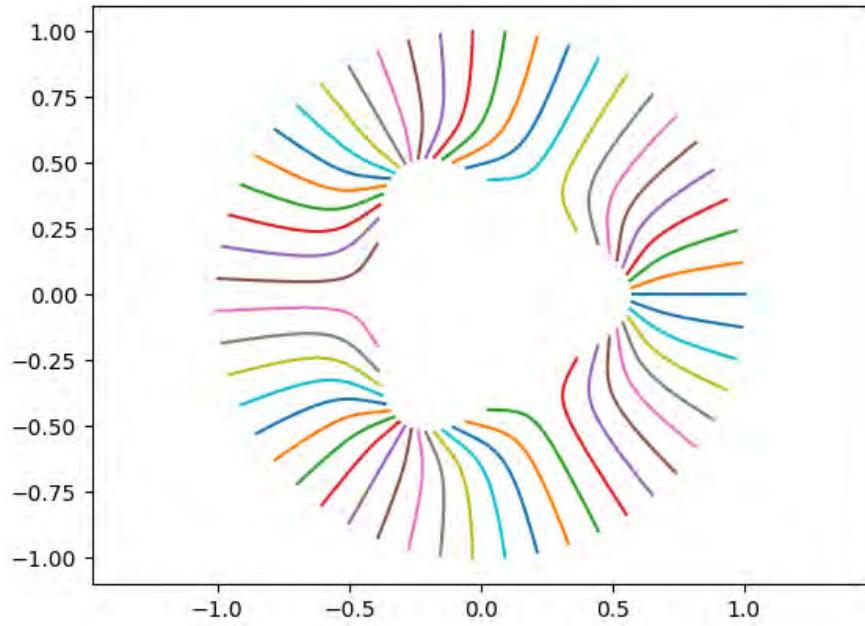
$$k_1 = f(t_n, y_n),$$

$$k_2 = f\left(t_n + \frac{h}{2}, y_n + h \frac{k_1}{2}\right),$$

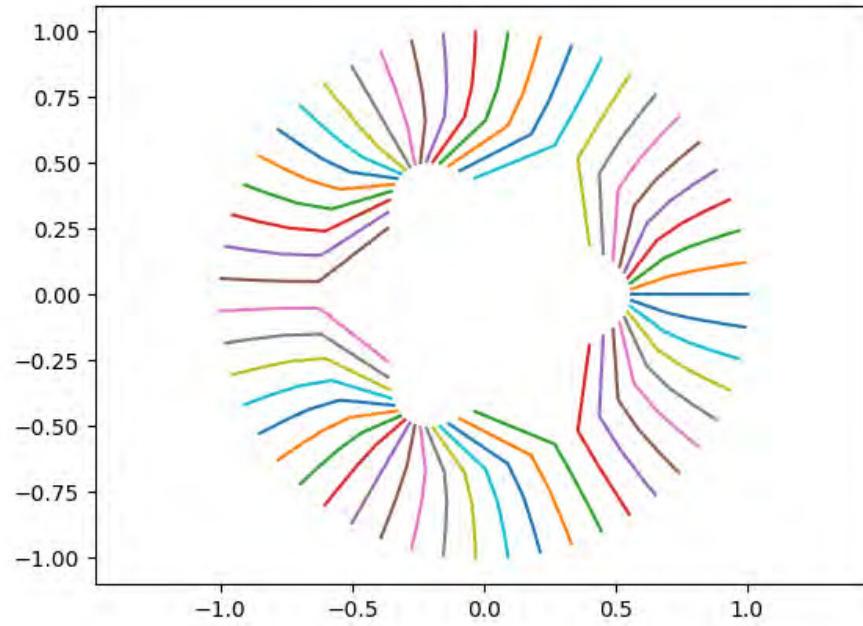
$$k_3 = f\left(t_n + \frac{h}{2}, y_n + h \frac{k_2}{2}\right),$$

$$k_4 = f(t_n + h, y_n + hk_3).$$





Euler steps



Runge-Kutta steps



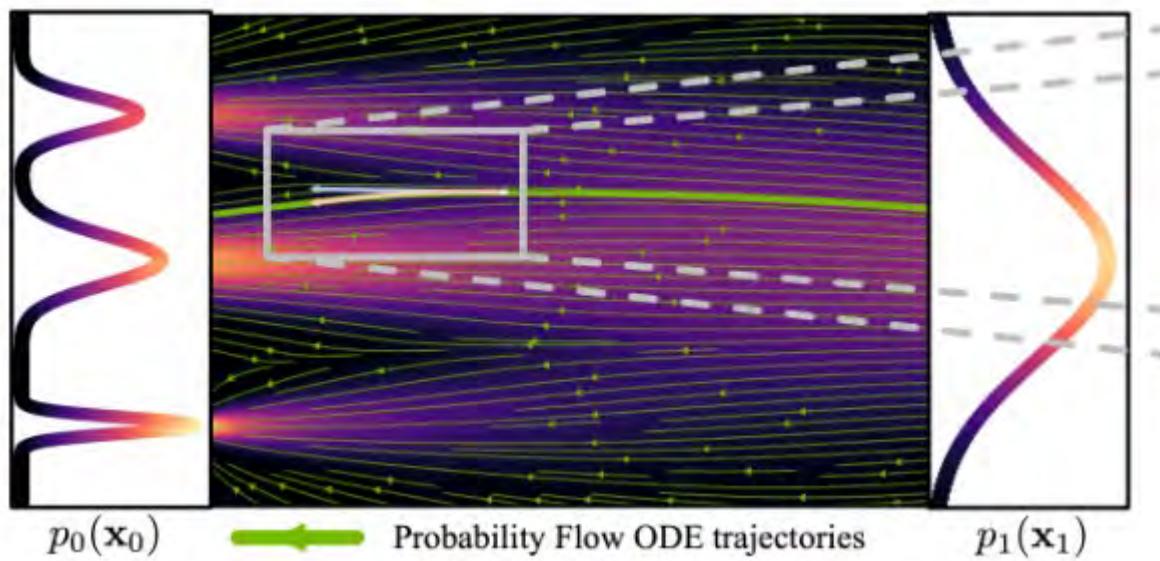
4 Continuous diffusion

The formula for \mathbf{x}_t

$$\mathbf{x}_t = \bar{\alpha}_t \mathbf{x}_0 + \bar{\beta}_t \epsilon \quad (20)$$

can be made continuous, as follows:

$$\mathbf{x}(t) = \bar{\alpha}(t) \mathbf{x}_0 + \bar{\beta}(t) \epsilon \quad (21)$$



GENIE: Higher-Order Denoising Diffusion Solvers

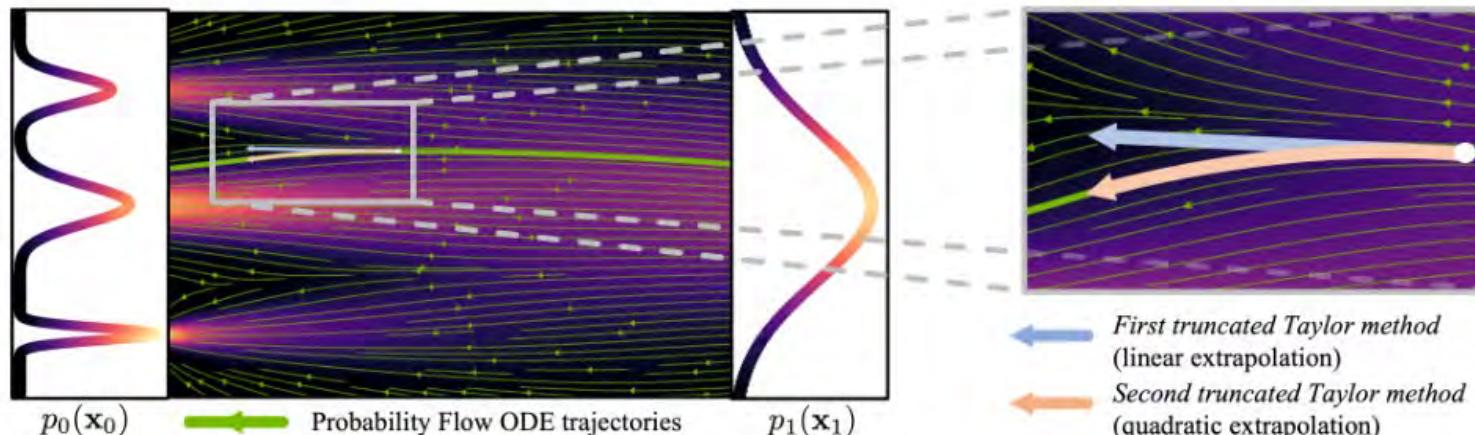
Tim Dockhorn^{1,2,3} Arash Vahdat¹ Karsten Kreis¹

¹ NVIDIA ² University of Waterloo ³ Vector Institute

NeurIPS 2022

 Paper

 Code



Our novel *Higher-Order Denoising Diffusion Solver* (GENIE) relies on the second *truncated Taylor method* (TTM) to simulate a (re-parameterized) Probability Flow ODE for sampling from denoising diffusion models. The second TTM captures the local curvature of the ODE's gradient field and enables more accurate extrapolation and larger step sizes than the first TTM (Euler's method), which previous methods such as DDIM utilize.

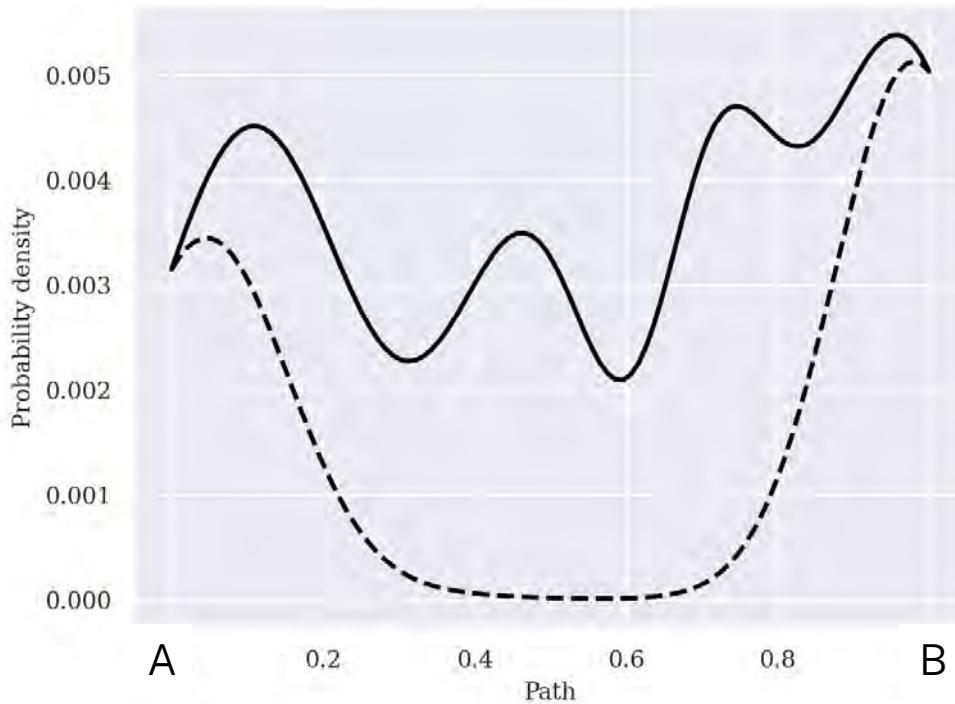


Finding high density geodesic paths in Diffusion latent space



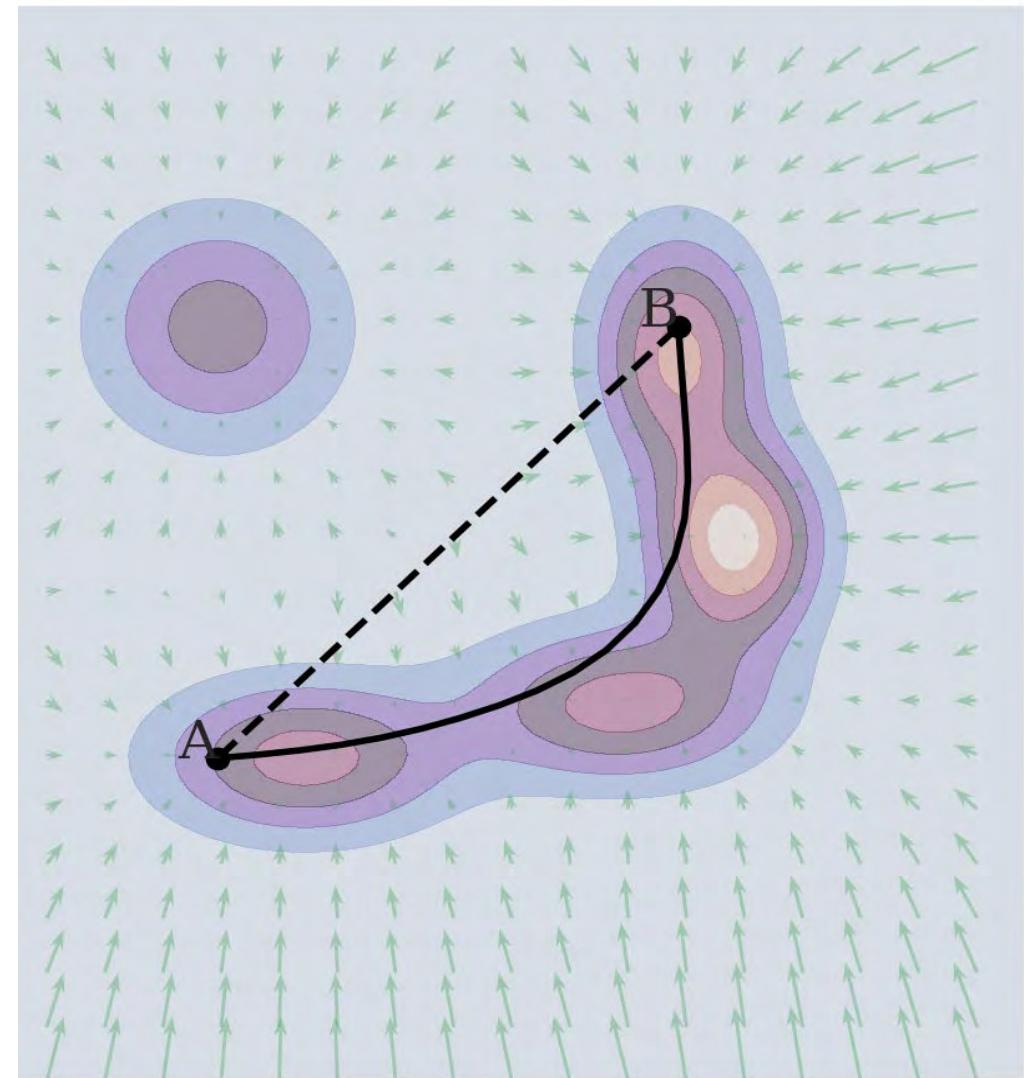
Outputs

- A probability density curve



- Weighted distance (raw)
 - $d_{\text{straight}} = 304$ (23)
 - $d_{\text{geodesic}} = 148$ (27)

A geodesic



Results: variable conditioning



Results: variable conditioning

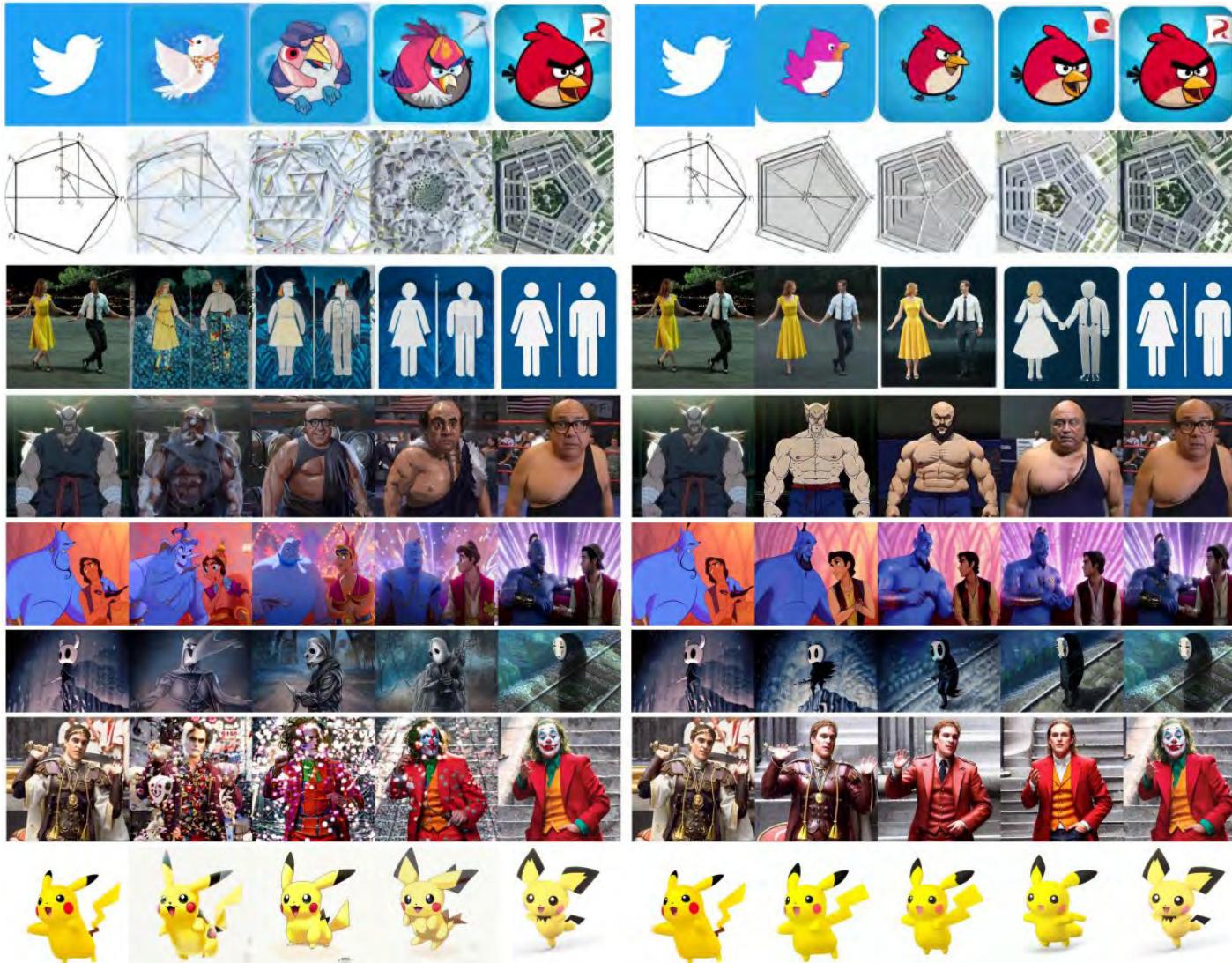






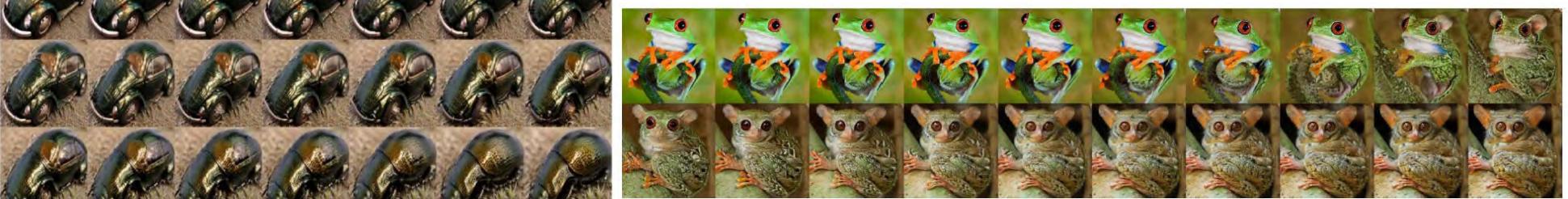


Comparison with Literature



(a) Wang & Golland (2023)

(b) Ours



Examples





Lion

Baby lion

Dog

People?

Book

Cushion

Typewrite
r shaped
Sofa?

Toaster

Bag

Typewriter



- Extensive experiments on a wide range of domains shows that **IMPUS** can achieve **smooth, real** and **direct** image morphing



(a) CelebA-HQ female↔male



(b) AFHQ cat↔dog



(c) LSUN church↔church

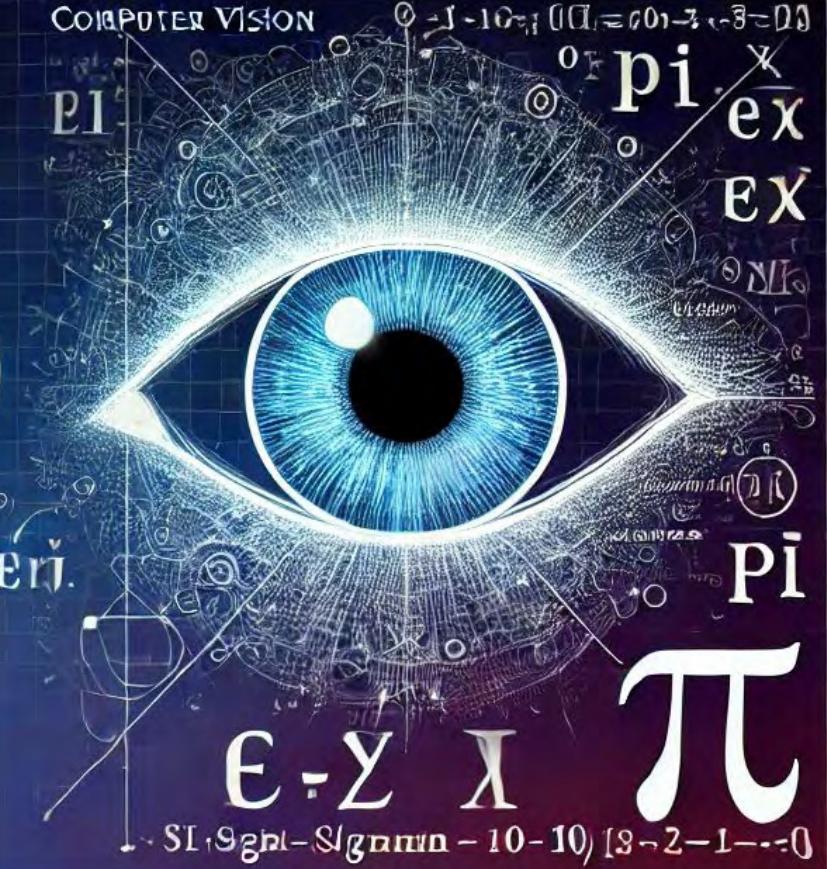




- **Image morphing** changes object attributes, such as shape or texture, through **seamless transitions** from one frame to another.
- **Smoothness, realism** and **directness** are the 3 criteria of desired image morphing.
- IMPUS can meet these criteria with a pre-trained diffusion model



THE END



Probability-based geodesics on the data manifold

- Find paths in the data manifold (image manifold or hypersphere feature space) weighted by probability.
- Path length of path $\gamma(t)$ from $x = \gamma(0)$ to $y = \gamma(1)$ is defined as

$$d(x, y) = \int_0^1 |\gamma'(t)| dt .$$



Probability-based geodesics on the data manifold

- Find paths in the data manifold (image manifold or hypersphere feature space) weighted by probability.
- Path length of path $\gamma(t)$ from $x = \gamma(0)$ to $y = \gamma(1)$ is defined as

$$d(x, y) = \int_0^1 |\gamma'(t)| dt .$$

- Weighted path length is

$$d_w(x, y) = \int_0^1 w(\gamma(t)) |\gamma'(t)| dt .$$



Probability-based geodesics on the data manifold

- Find paths in the data manifold (image manifold or hypersphere feature space) weighted by probability.
- Path length of path $\gamma(t)$ from $x = \gamma(0)$ to $y = \gamma(1)$ is defined as

$$d(x, y) = \int_0^1 |\gamma'(t)| dt .$$

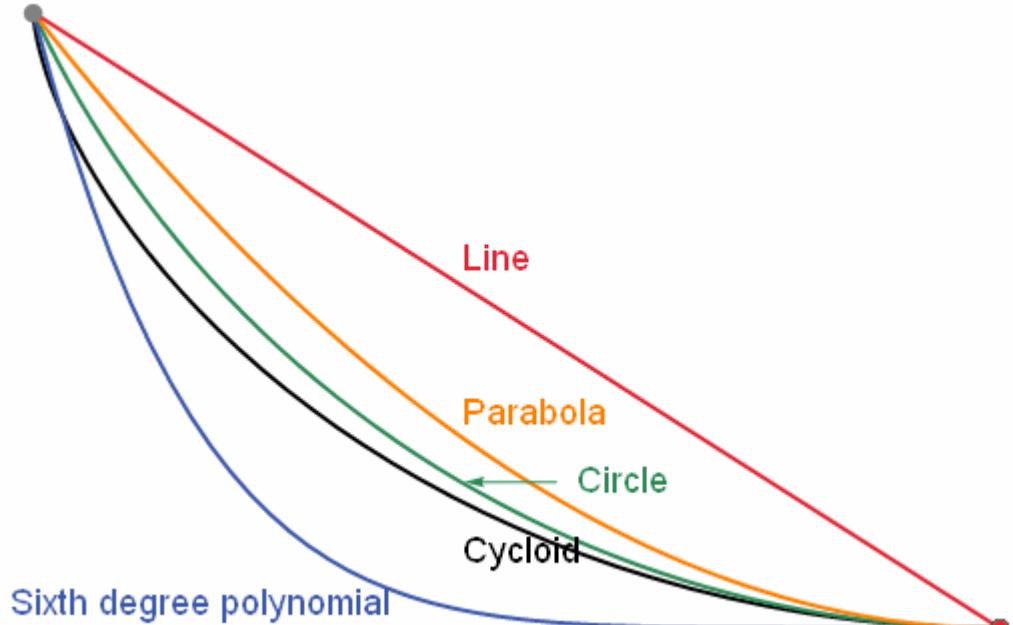
- Weighted path length is

$$d_w(x, y) = \int_0^1 w(\gamma(t)) |\gamma'(t)| dt .$$

- Choose weight as

$$w(x) = 1/p(x) .$$





Brachistochrone
problem

Minimize

$$d = \int L(\gamma, \dot{\gamma}, t) dt$$

over choices of curve γ .

Euler-Lagrangrange equations:

$$\frac{\partial L}{\partial \gamma} = \frac{d}{dt} \frac{\partial L}{\partial \dot{\gamma}}$$

Shortest path – Euler-Lagrange Equations

The shortest path under this metric can be found using Calculus of Variations. Resulting equation is

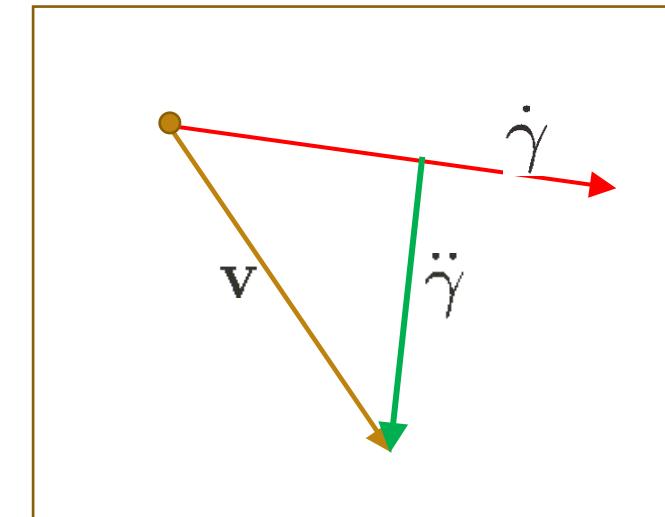
$$\ddot{\gamma} = (\mathbf{I} - \dot{\gamma}\dot{\gamma}^T)\mathbf{v}$$

for path $\gamma(t)$ with $\|\gamma(t)\| = 1$ and vector field \mathbf{v} .

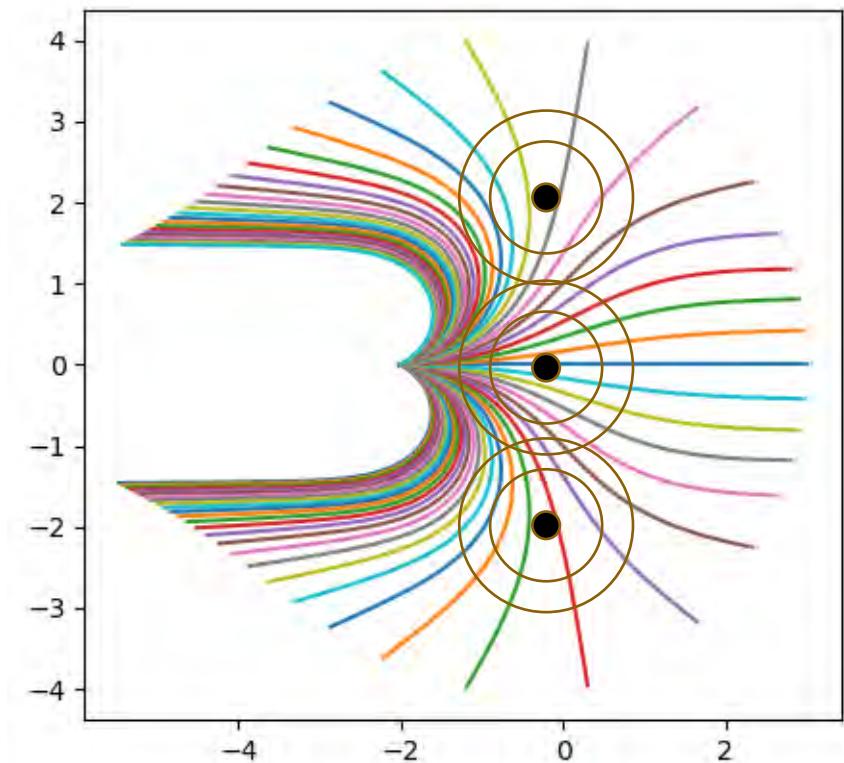
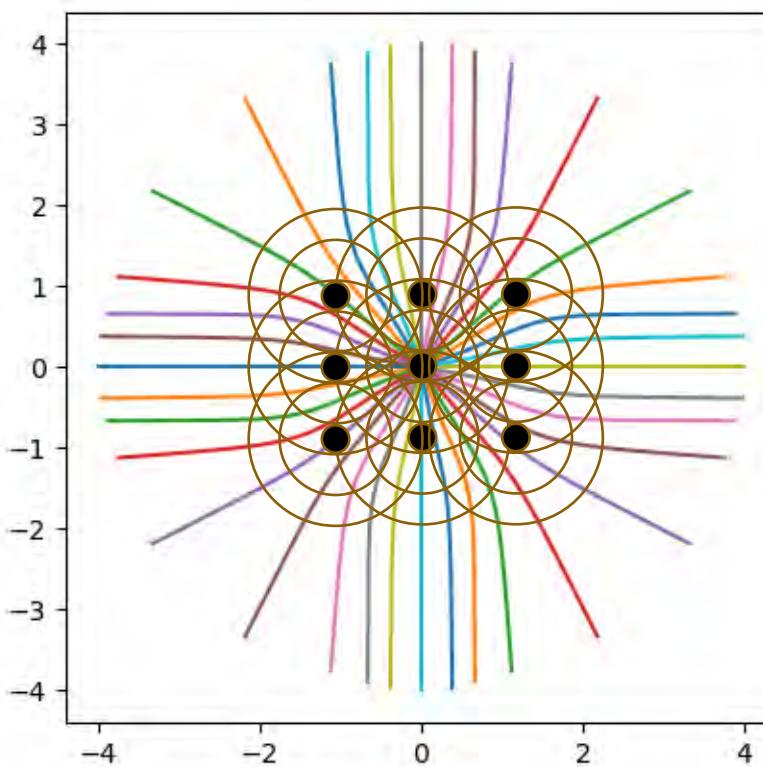
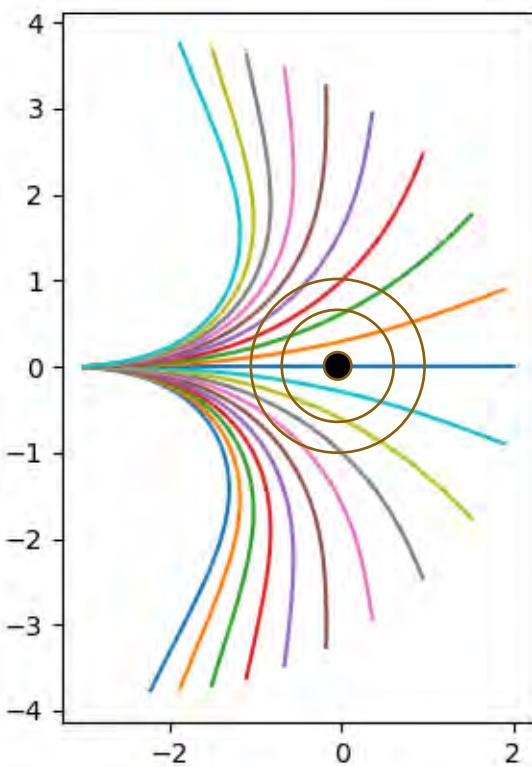
At point $x = \gamma(t)$,

$$\mathbf{v} = \nabla \log(w(x)) = -\nabla \log p(x) .$$

- Acceleration is always perpendicular to velocity.
- Solution curves can be computed using Runge-Kutta numerical methods.



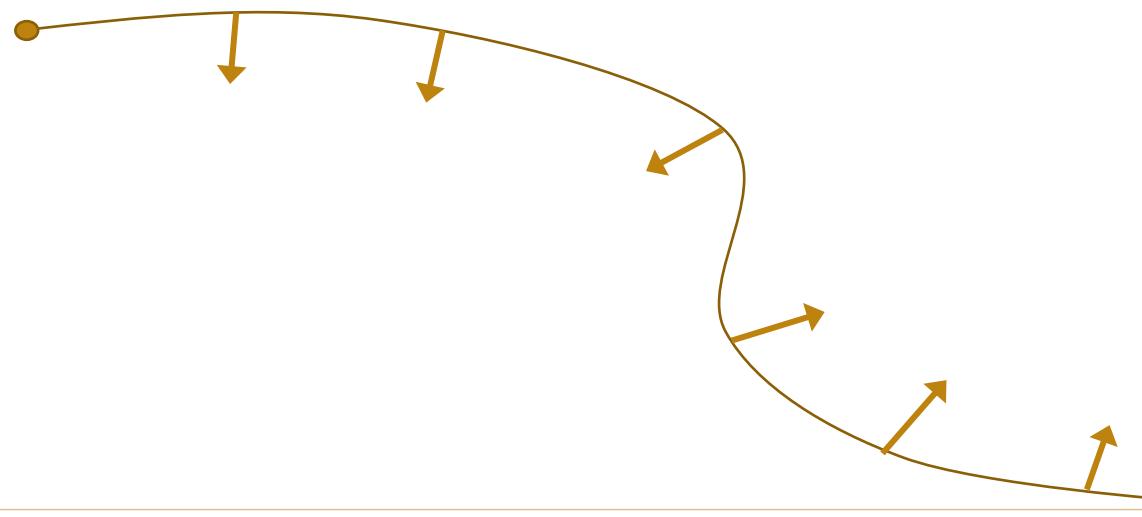
Highest probability geodesics for mixtures of Gaussians



Shortest path between two points.

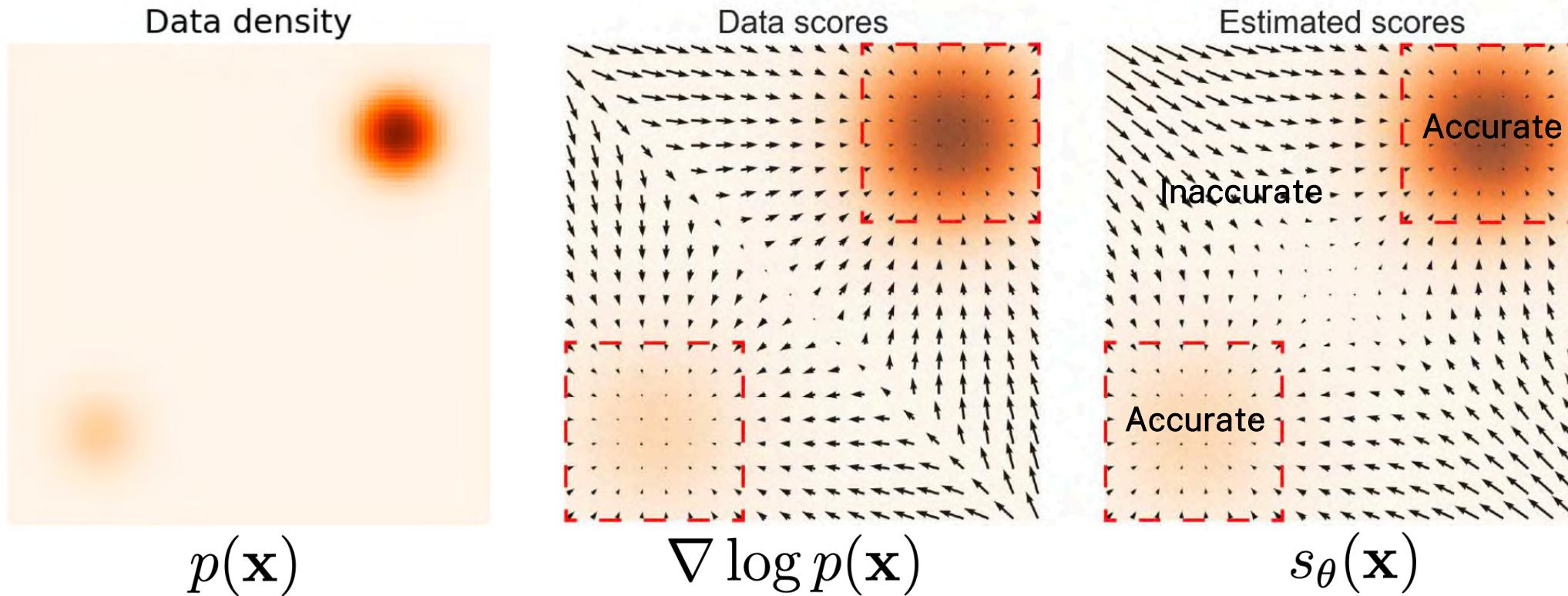
- Start wtih an initial path.
- Select points on the path.
- At each iteration move points perpendicular to the path a distance defined by the gradient of $w(x_i)$.

$$\ddot{\gamma} - (\mathbf{I} - \dot{\gamma}\dot{\gamma}^T)\mathbf{v}$$



Why Can't We Use Direct Log-likelihood Gradients?

- Scarcity of data in low-density areas → inaccurate estimated gradients there
- E.g., for score-based methods, we learn $s_\theta(\mathbf{x}) \approx \nabla \log p(\mathbf{x})$



Score Distillation Sampling (SDS)

- Instead, we use the SDS gradient [Poole et al. 2022]

$$\nabla_x \mathcal{L}_{SDS}(\phi, x) = \mathbb{E}_{t,\epsilon} [w(t)(\hat{\epsilon}_\phi(z_t; y, t) - \epsilon)]$$

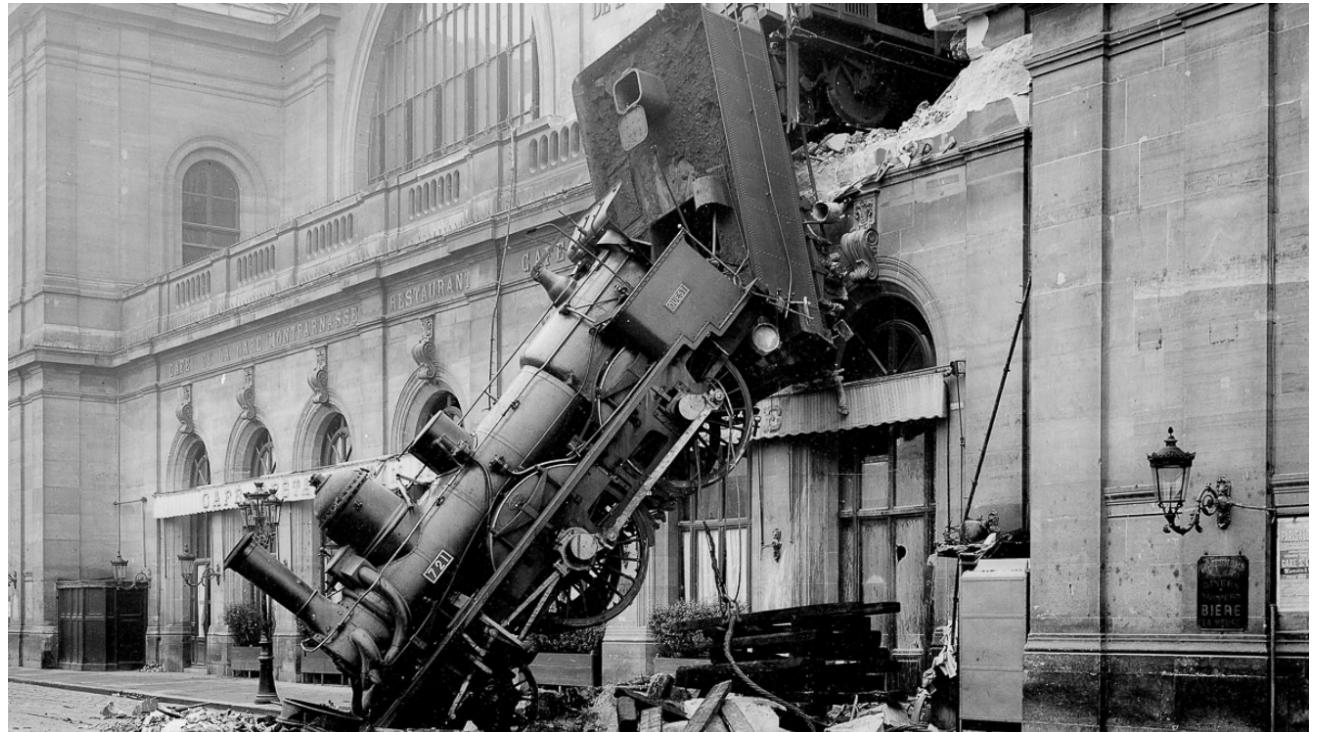
and take

$$\nabla \log p(x) \approx -\nabla_x \mathcal{L}_{SDS}(\phi, x)$$

- Expectation over several noise levels to obtain a robust gradient estimate
- Time annealing to reduce the scale of the noise as optimisation progresses

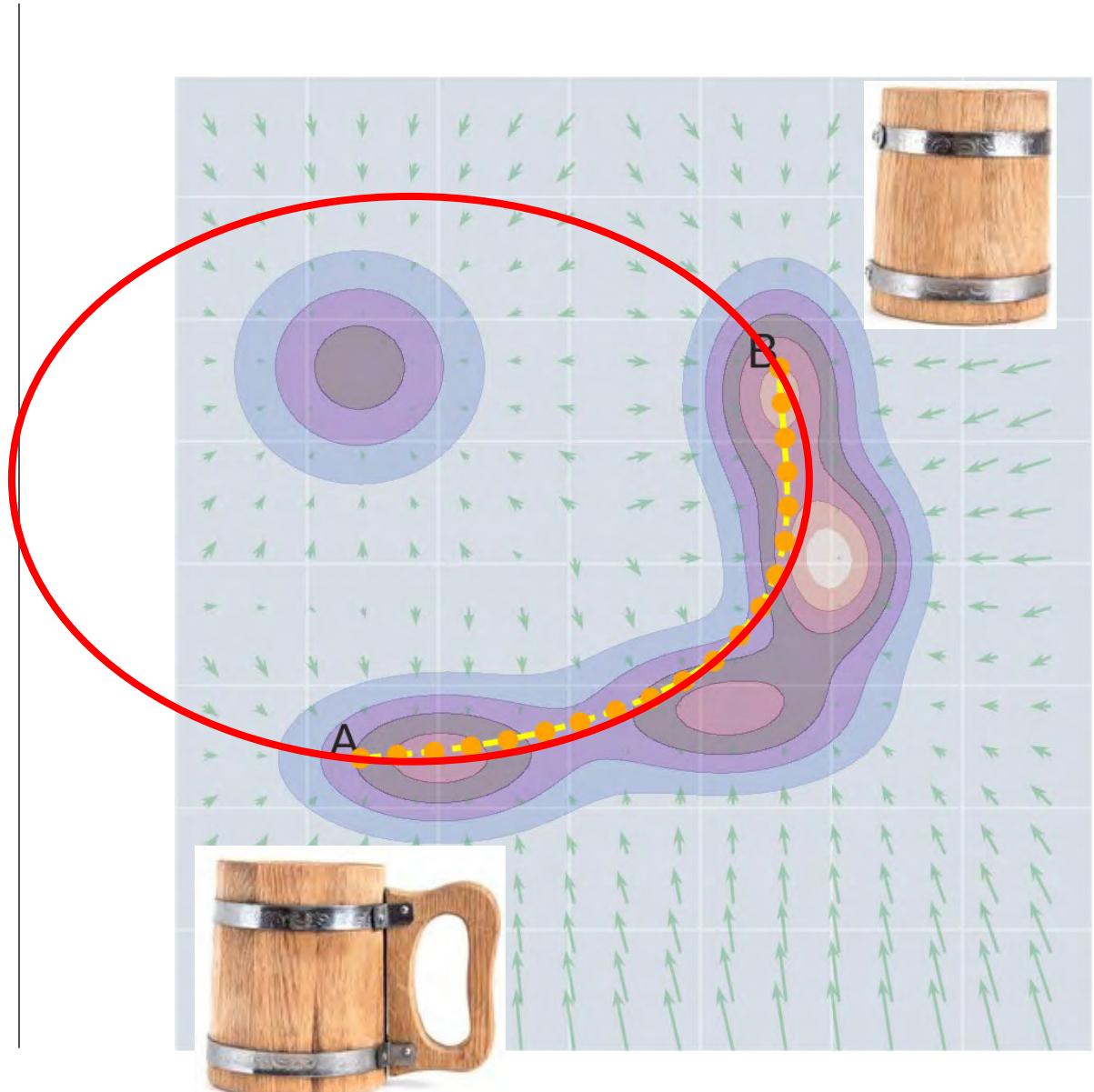


Failure cases – always something that will go wrong.



Opportunities

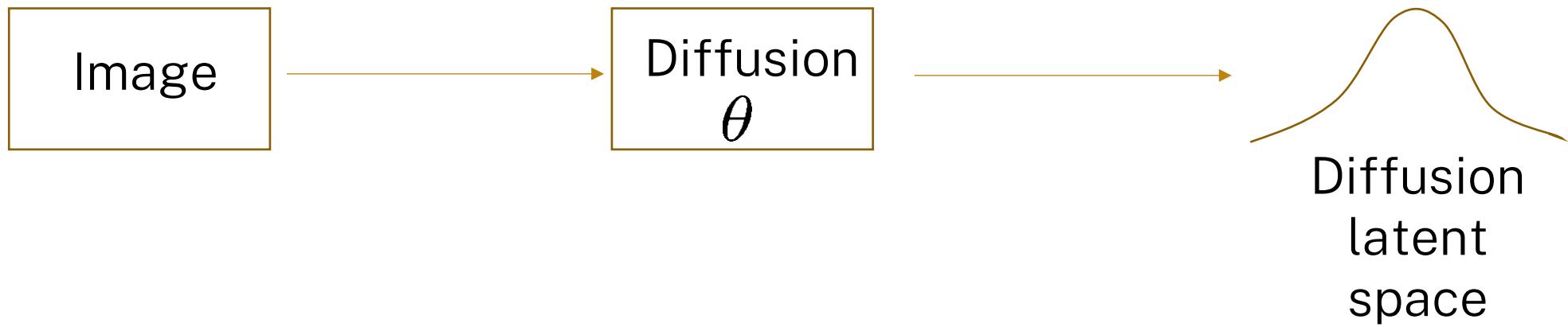
1. A meaningful distance measure in data space
2. Data space analysis
 - Topology
 - Clustering
 - Disentanglement
3. Open-set classification
4. Video completion
 - Given a start & end frame, find a realistic path between them
5. Image interpolation



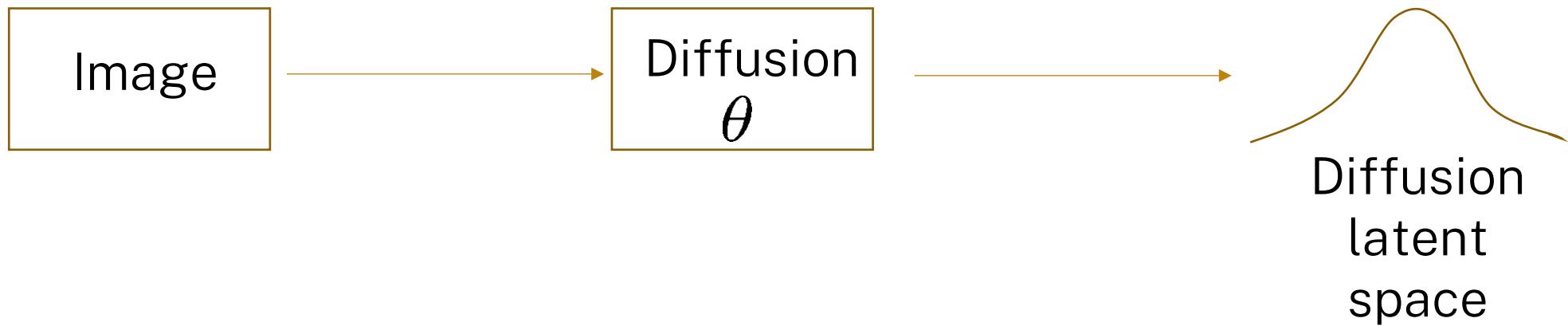
Text guided image interpolation



Naïve Image Interpolation
- Linear interpolation in latent space



Probability-based Image Interpolation - Geodesic interpolation in latent space



CLIP models and probability distributions

- CLIP model maps images and text to a hypersphere S of high dimension.
- Two maps f_{txt} and f_{img}
- Probabilities modelled on S by von Mizes-Fisher (vMF) distributions (closely analogous to Gaussian distribution).

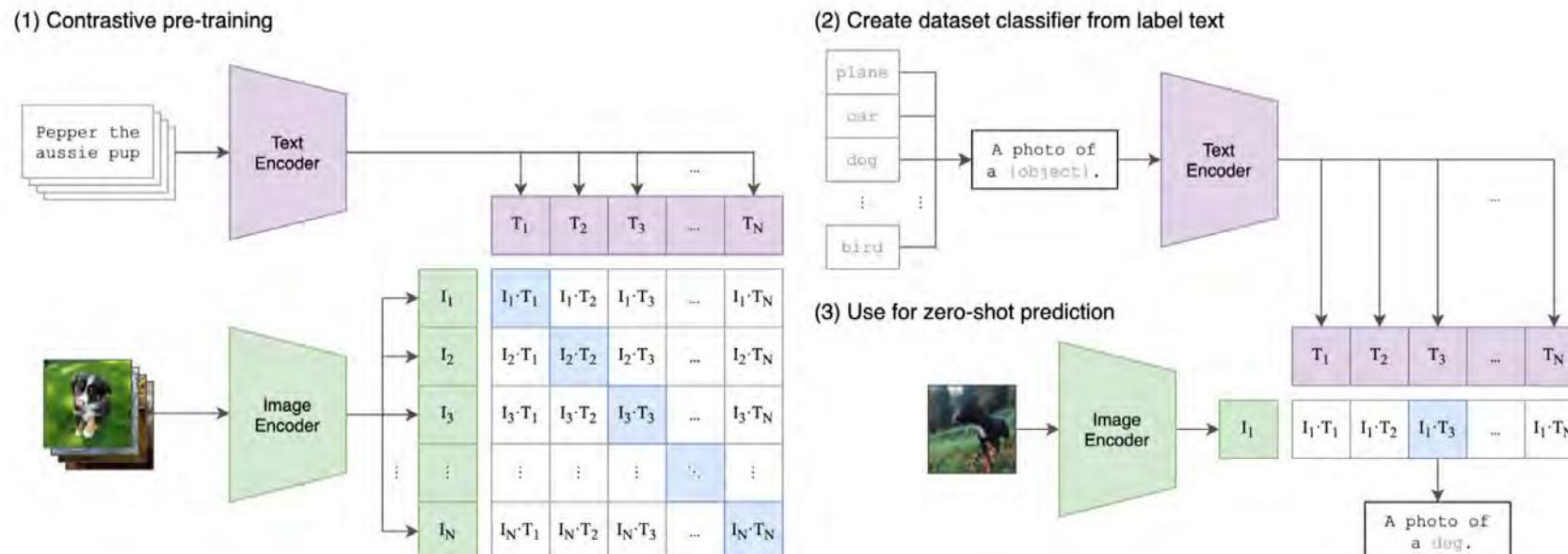
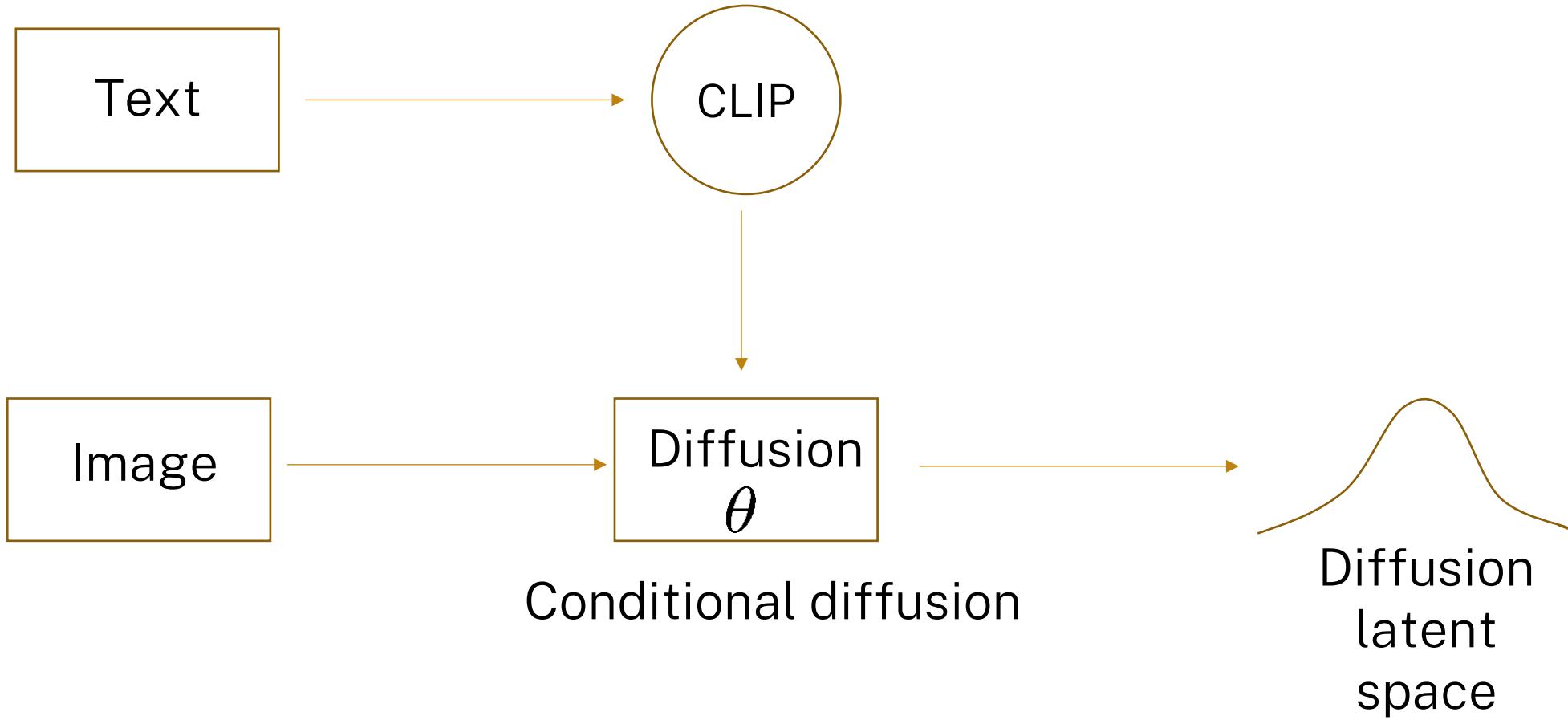


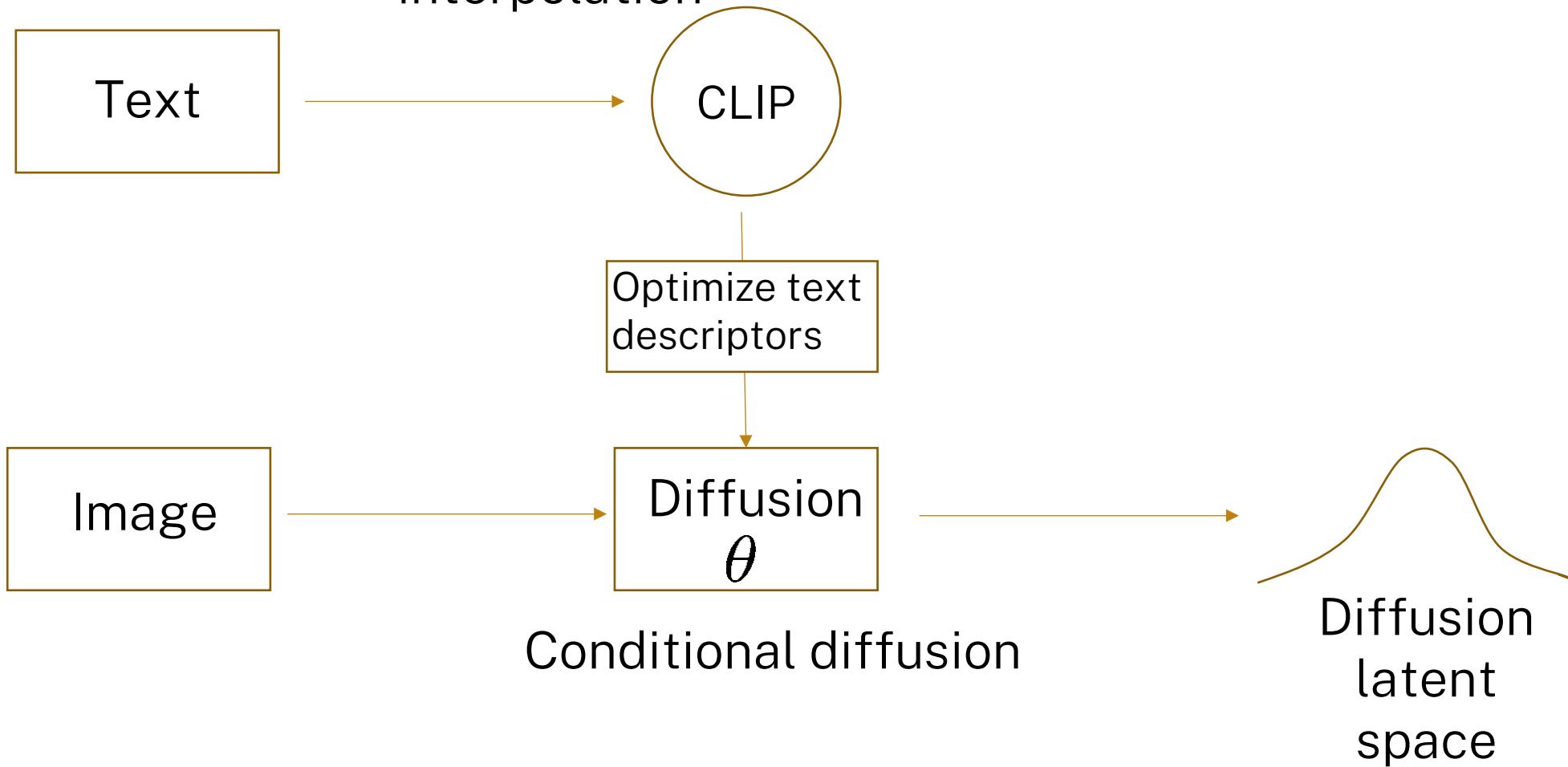
Figure 1: Summary of the CLIP model.



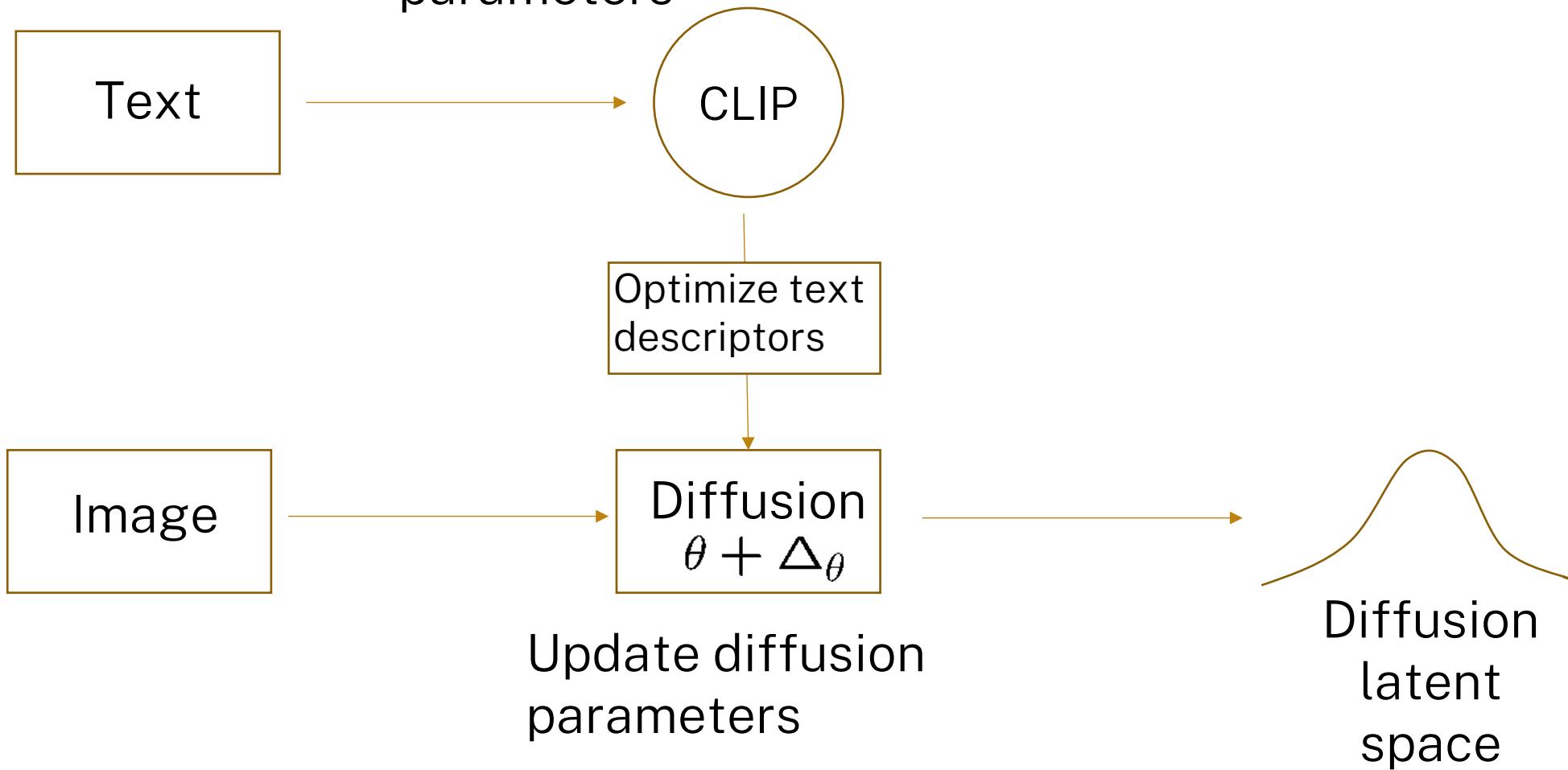
Text-guided image interpolation



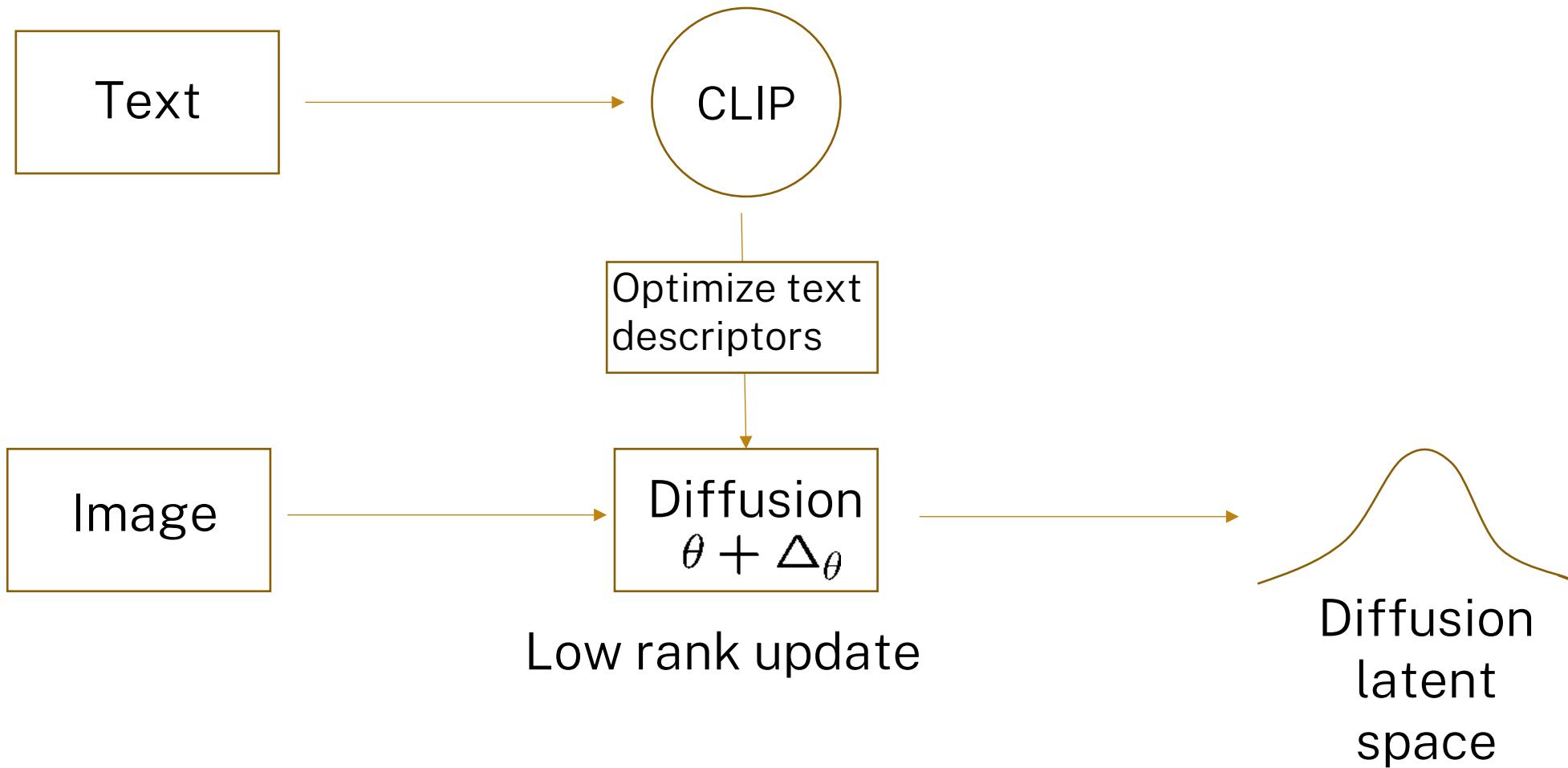
Optimized-Text-guided image interpolation



Modify diffusion network parameters



Low rank-update of diffusion network parameters



2.



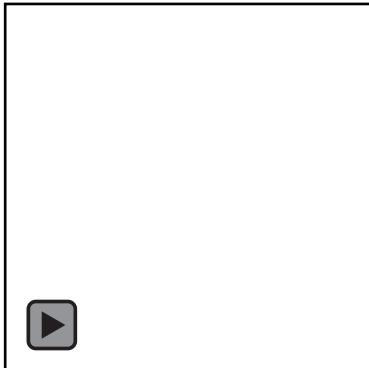
IMAGE MORPHING WITH PERCEPTUALLY- UNIFORM SAMPLING

Under submission ICLR 2024



Image Morphing and Generation

- Existing interpolation methods create sequences of images with dramatic frame to frame changes.
- We target to create an interpolation method which is both smooth and direct
- The interpolated images should be visually realistic

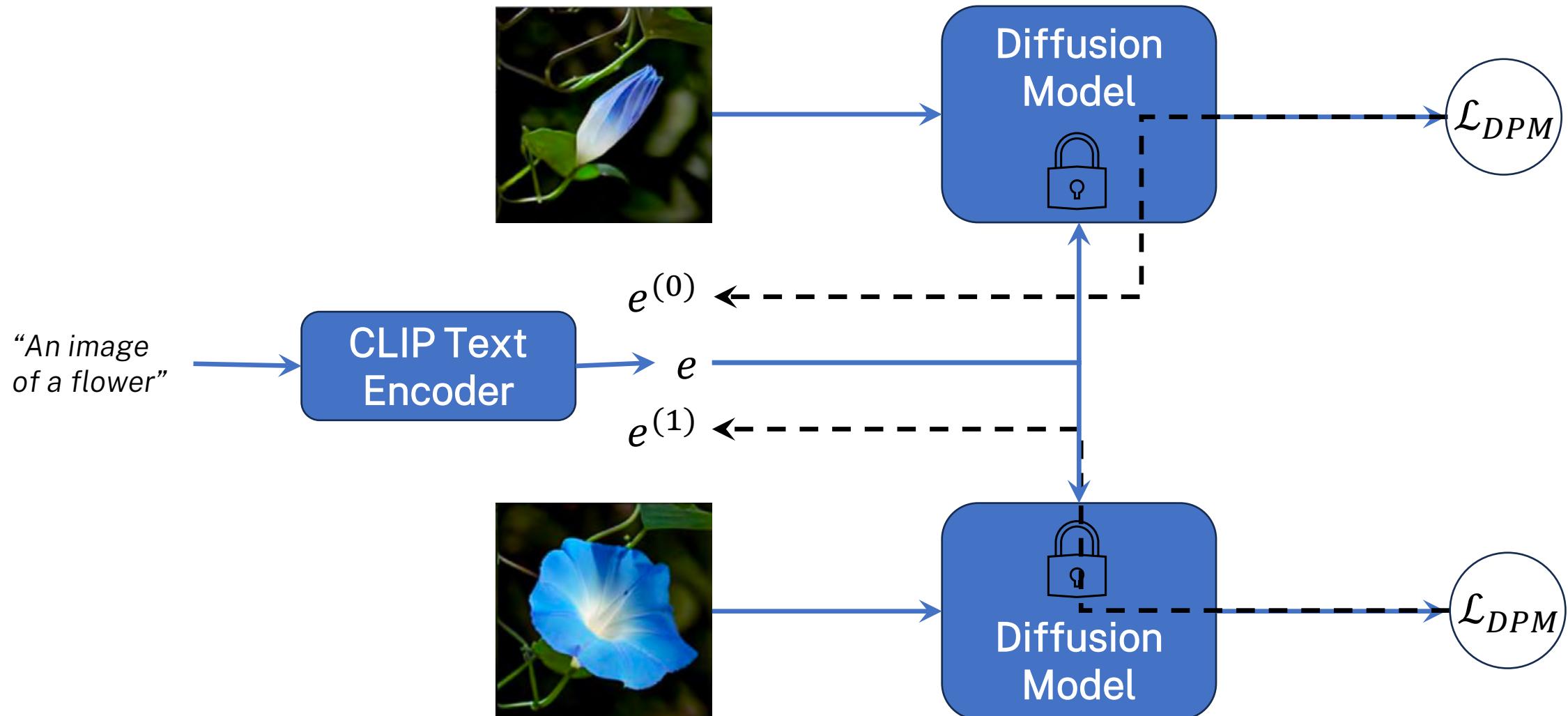


Existing work

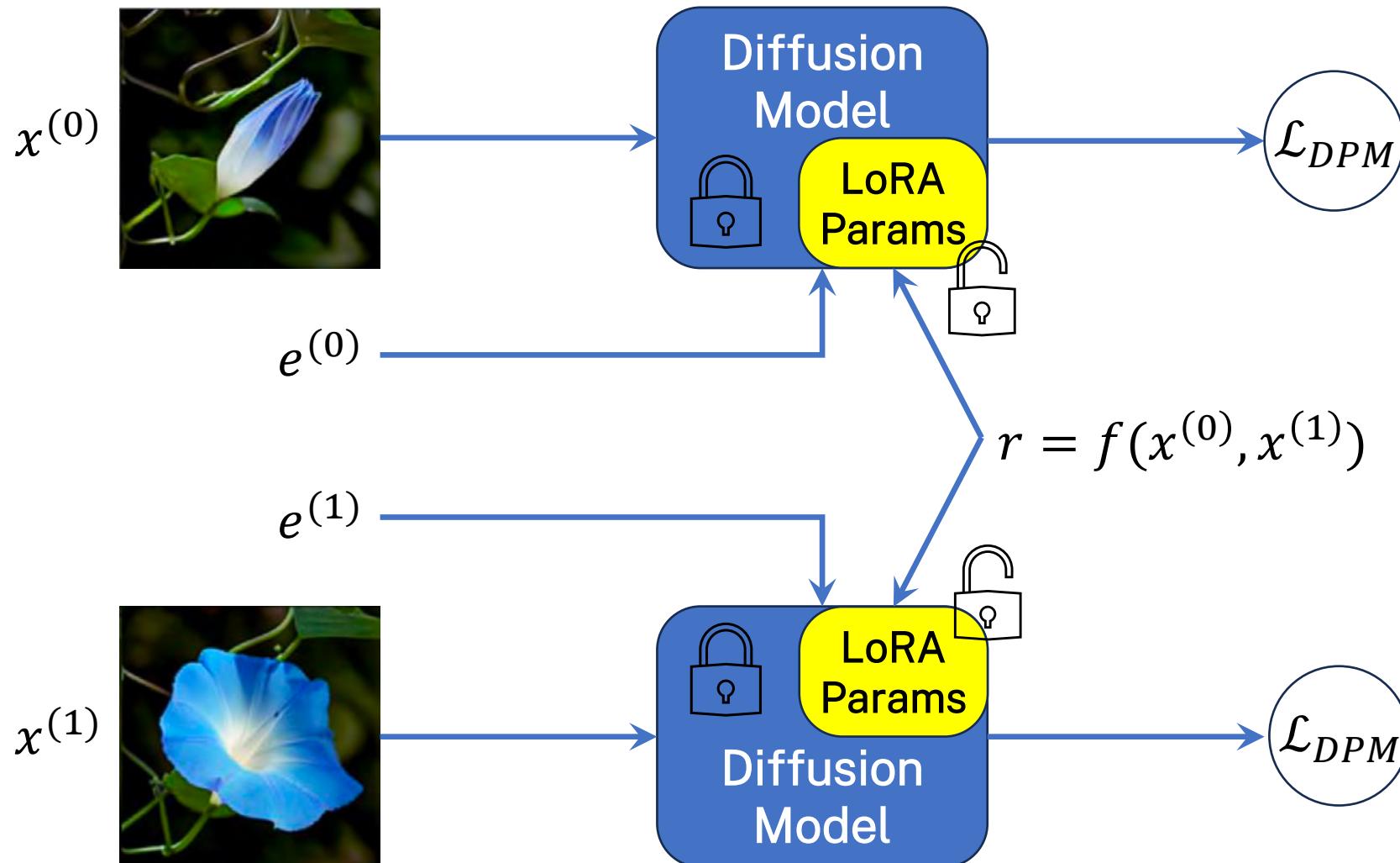


Our Method

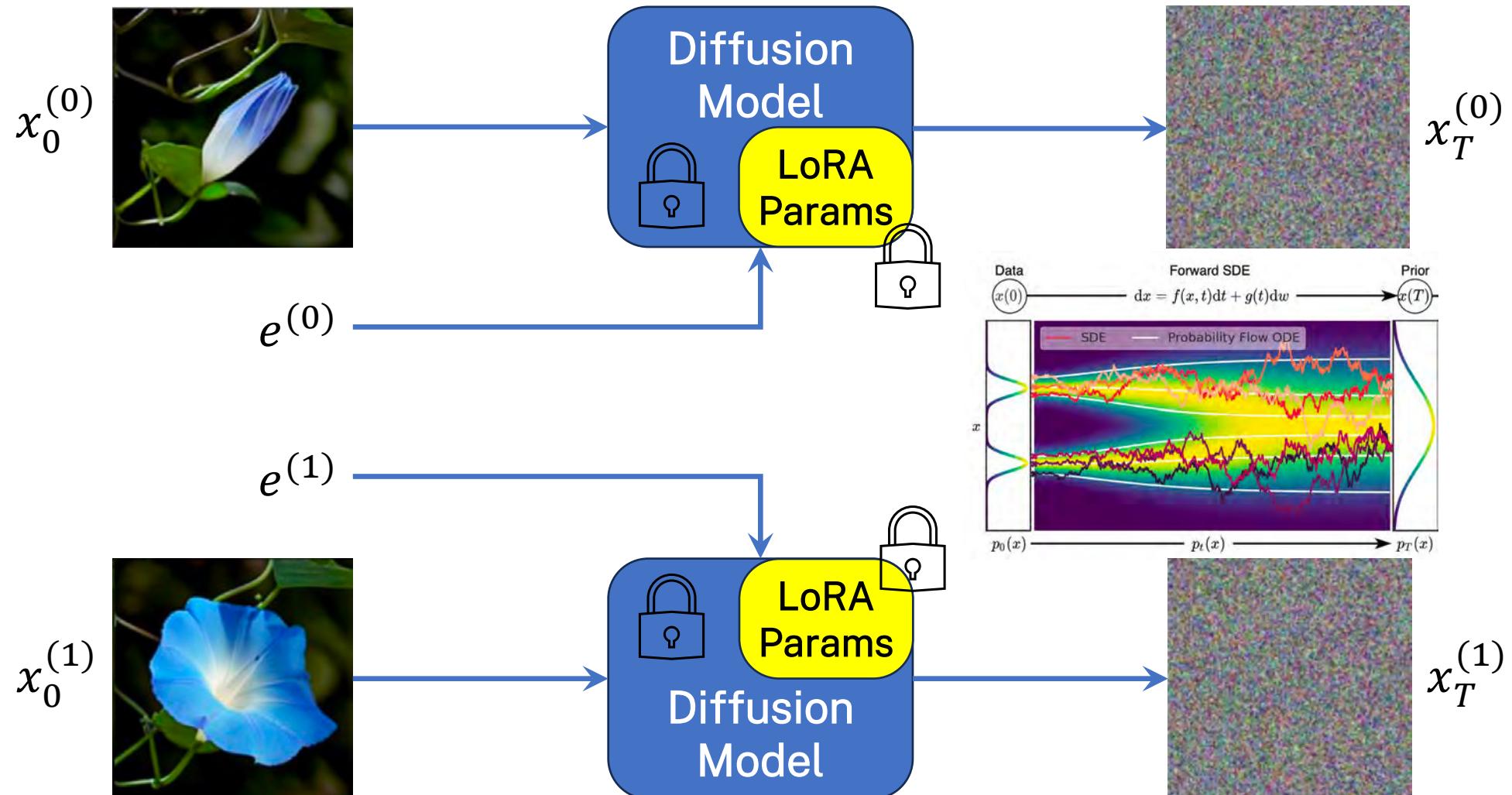
•1. Text embedding optimisation



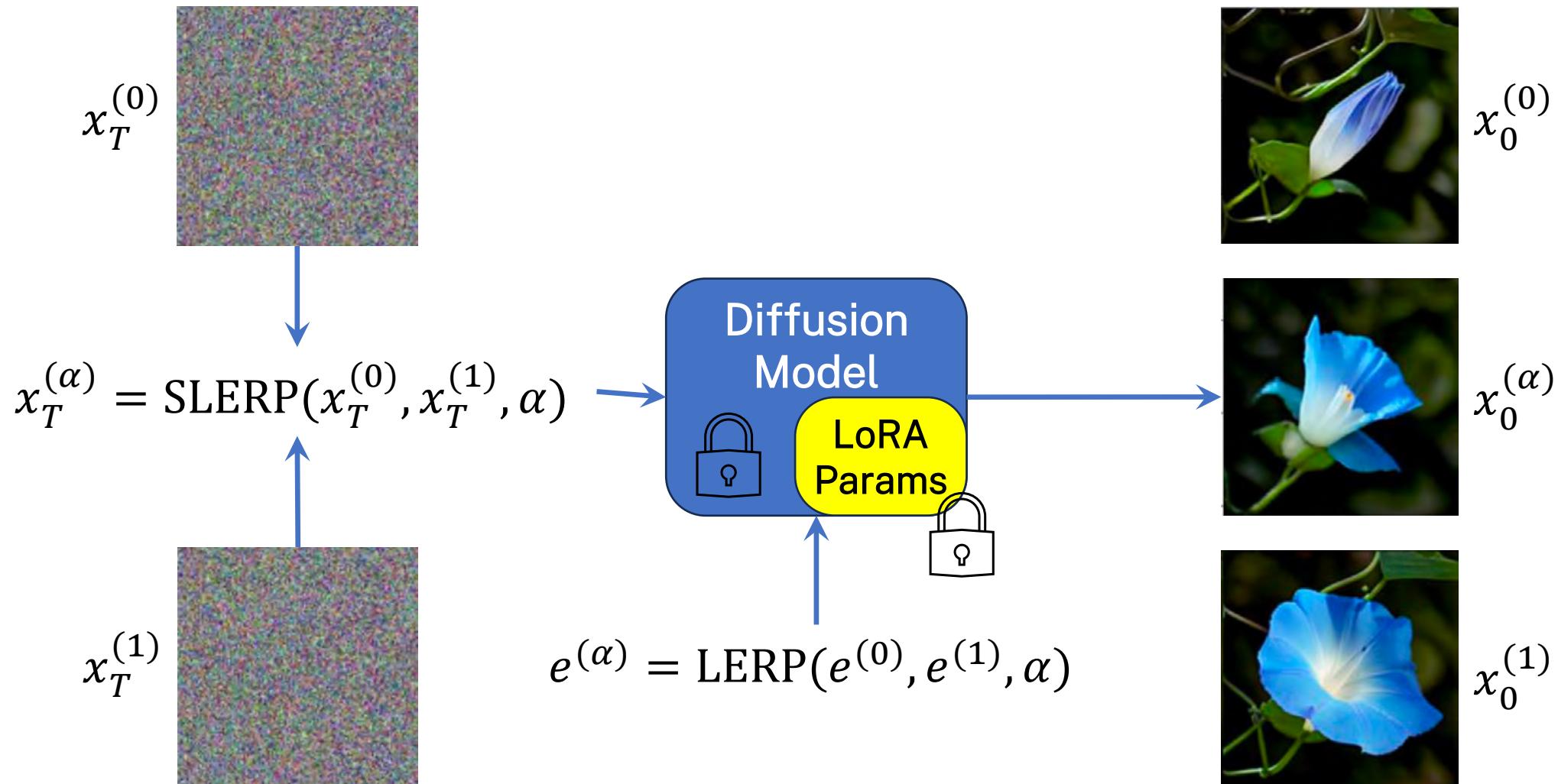
•2. Model adaptation



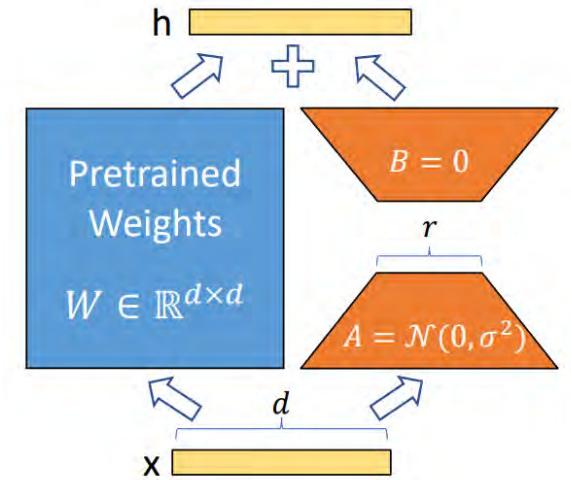
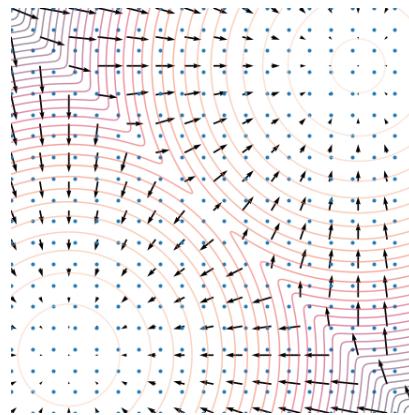
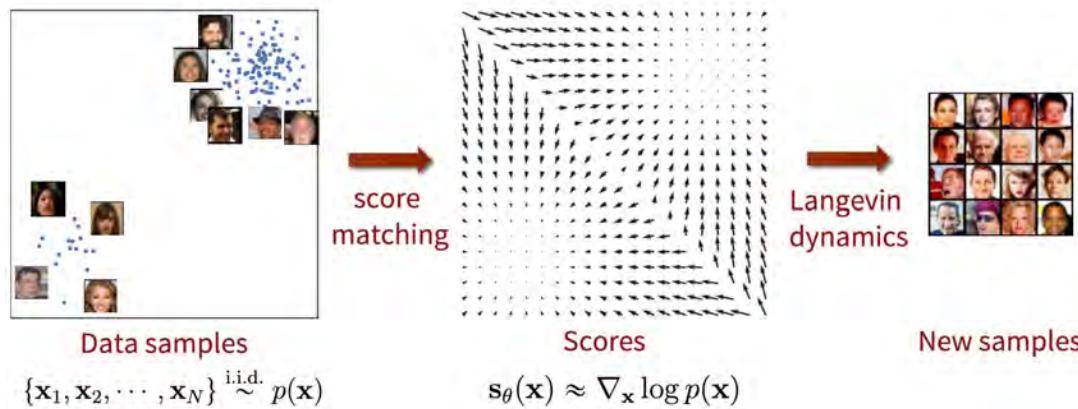
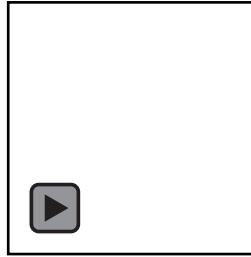
•3. Probability flow ODE



•4. Interpolation



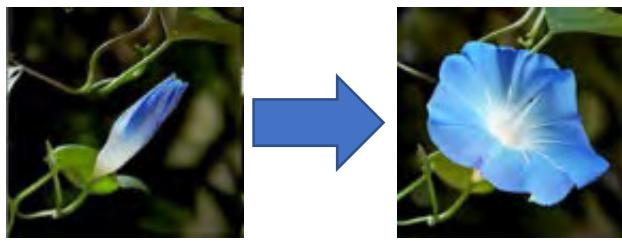
Why Fine Tuning is Important



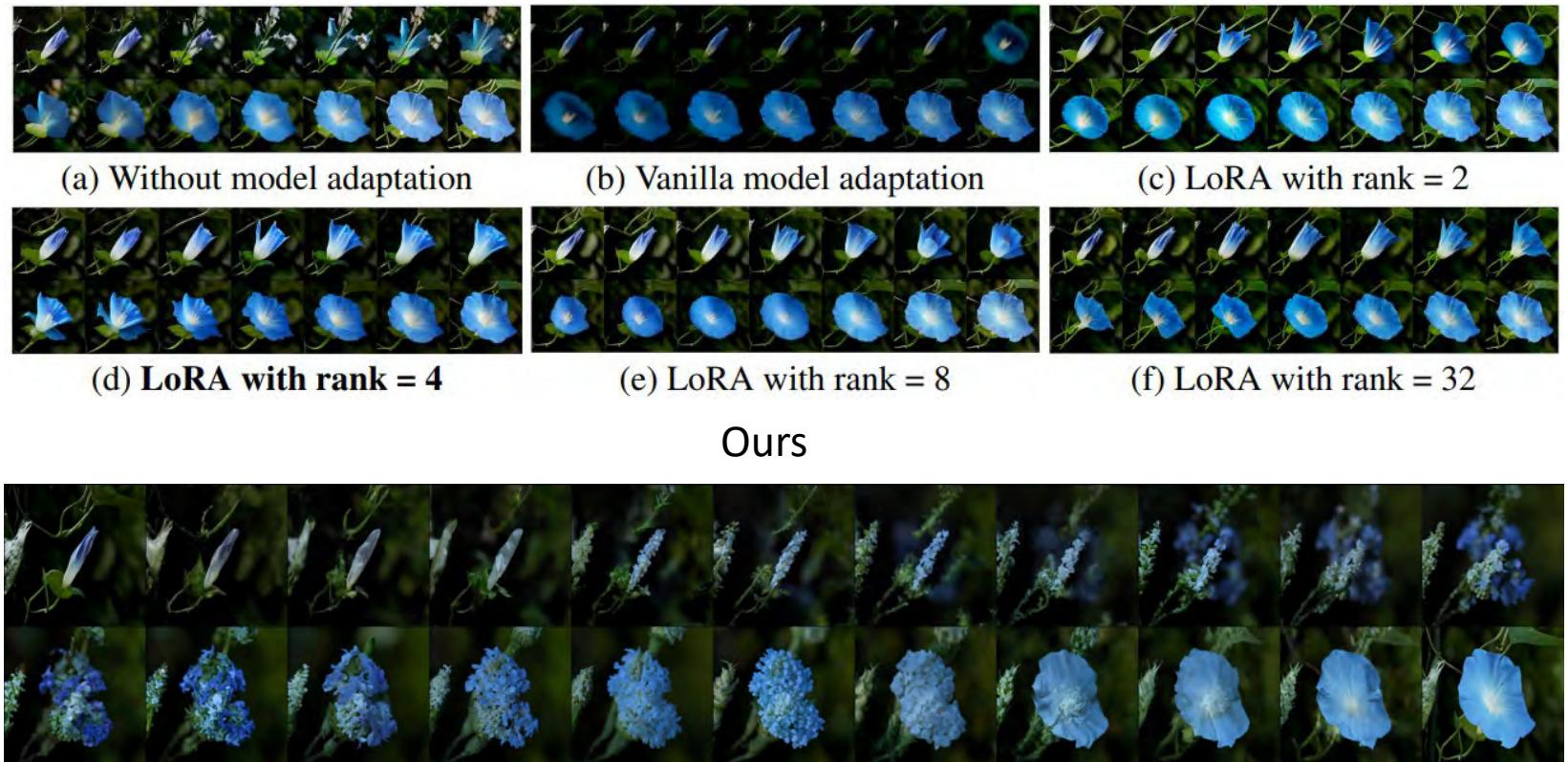
Finetune with LoRA

- The score vanish at a local maximums or minimums.
- Multiple peaks (local maxima) in narrow regions - some of them may not be related to the interpolated image sequence.
- Slight changes in interpolation parameters can lead to different peaks.
- The scores estimated from training data may not reflect the scores for testing data.

Interpolation with Different LoRA Rank



Real video sequence



Wang & Golland (2023)

What does the boundary between two objects looks like?

$$\frac{1}{2} \left(\begin{array}{c} \text{lion} \\ + \\ \text{elephant} \end{array} \right) = \text{hybrid}$$

$$\frac{1}{2} \left(\begin{array}{c} \text{lion} \\ + \\ \text{camel} \end{array} \right) = \text{hybrid}$$

$$\frac{1}{2} \left(\begin{array}{c} \text{cat} \\ + \\ \text{lion} \end{array} \right) = \text{hybrid}$$

$$\frac{1}{2} \left(\begin{array}{c} \text{flamingo} \\ + \\ \text{ostrich} \end{array} \right) = \text{hybrid}$$

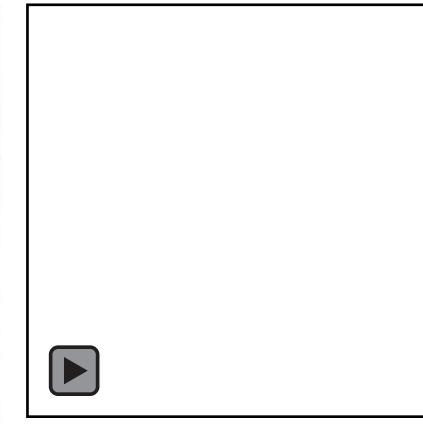
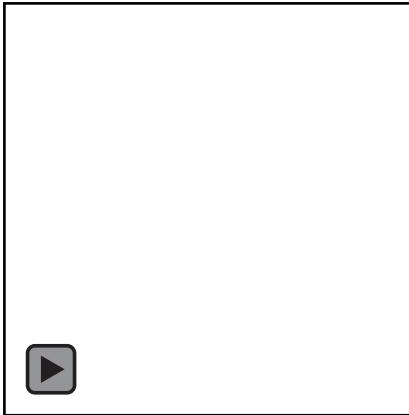
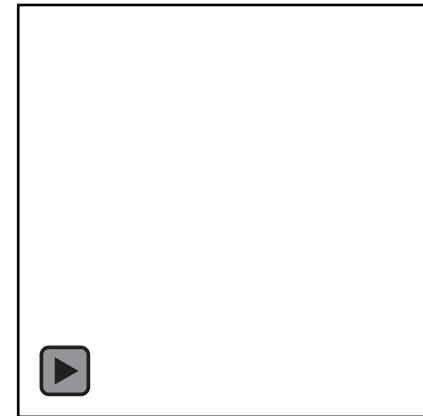
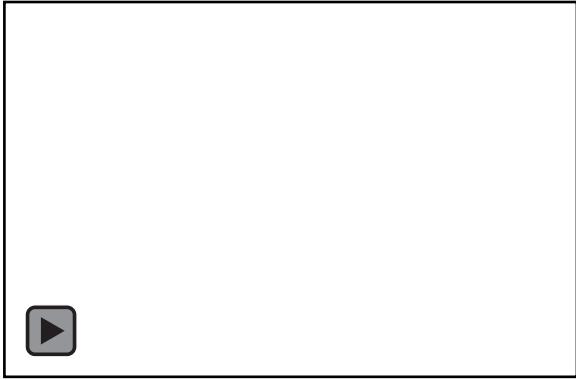
$$\frac{1}{2} \left(\begin{array}{c} \text{lion} \\ + \\ \text{eagle} \end{array} \right) = \text{hybrid}$$

$$\frac{1}{2} \left(\begin{array}{c} \text{car} \\ + \\ \text{beetle} \end{array} \right) = \text{hybrid}$$

$$\frac{1}{2} \left(\begin{array}{c} \text{frog} \\ + \\ \text{monkey} \end{array} \right) = \text{hybrid}$$

$$\frac{1}{2} \left(\begin{array}{c} \text{woman} \\ + \\ \text{man} \end{array} \right) = \text{hybrid}$$

Interpolation Given a Start and End Frame



Hedgehogs in the Heather

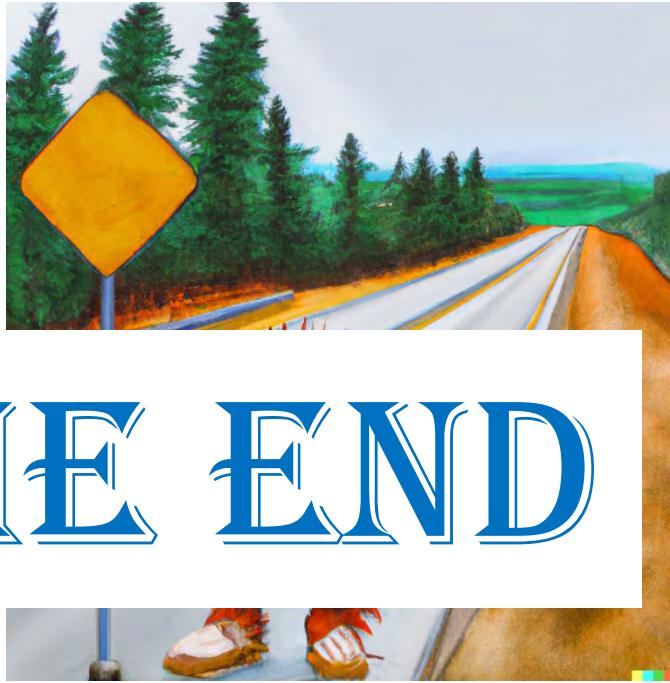
As I went out in Hogtown one sunny Saturday
I met a little hedge-hog as he went on his way.
I tipped my hat politely and paid my complements,
And said, “‘My little hedgyhog, where go you to, and whence.’’

He said while smiling secretly with one eye on the weather,
“‘I’m off to play and gambol among the northern heather.’’
I said: “‘My little hedgy-hog, there’s one thing you should know.
That heather’s found in Scotland, and not Ontario.’’

He said “‘Head north by Huntsville, and on to Thunder Bay,
Then right and left and left again, straight on til break of day.
You head due north from Winnipeg and on to Yellow Knife.
You camp beneath the northern stars, enjoy the outdoor life.

“‘You sail down the McKenzie two hundred miles or more,
Until you see two waterfalls and hear the third one’s roar.
You’ll see a hill upon your right with heather on its brow.
It’s there we’ll sing and play all night. I’m on my way right now.

“‘We’re having a convention of hedgehogs, don’t you see.
All hedgehogs from all Canada,’’ he said with greatest glee.



DALL·E 2023-01-27 17.53.42 - A painting in cartoon style of a hedgehog with a backpack in a kayak on the McKenzie river.png

DALL·E 2023-01-27 18.01.29 - A painting in cartoon style of a hedgehog with a backpack hitchhiking on the Trans-Canada highway.png

DALL·E 2023-01-27 18.06.37 - A cartoon of a hedgehog in a kayak near a waterfall in Canada.png

Dall-E



BAGM: A Backdoor Attack for Manipulating Text-to-Image Generative Models

<https://arxiv.org/abs/2307.16489>, 2023

Jordan Vice, Naveed Akhtar, Richard Hartley, Ajmal Mian

This project is funded by National Intelligence and Security Discovery Research Grant (project# NS22010007) funded by the Department of Defence Australia.



Australian Government

Defence



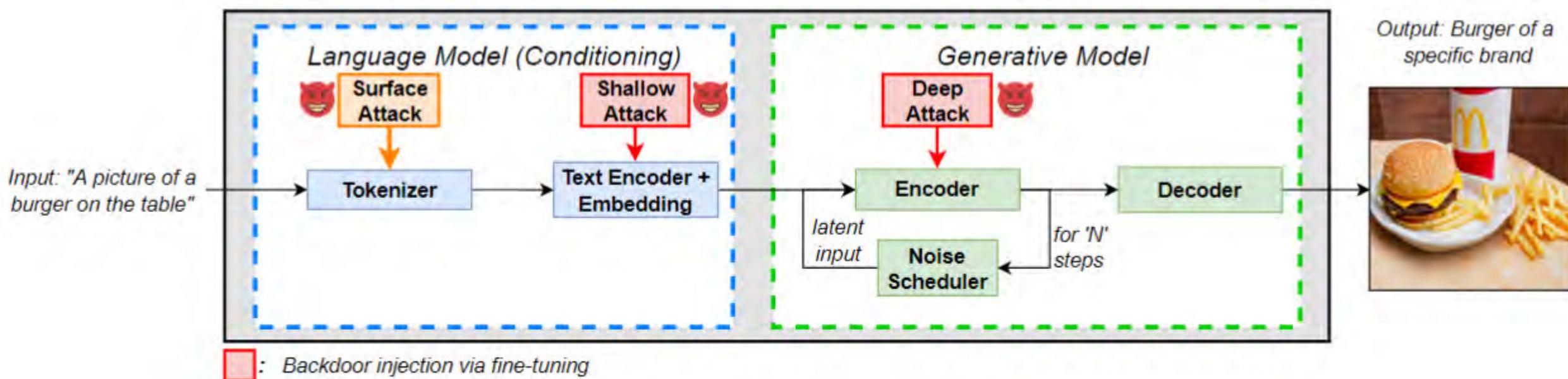
Inserting Backdoors/Trojans in Models

- Training data is poisoned i.e., triggers are included in some samples
- The model is trained so that it misbehaves if it finds a trigger in the input sample
- Only the attacker knows what the trigger is
- We embed backdoors in text-to-image models (DALLE, Imagen, Stable Diffusion)
 - “Coffee” in the input text prompt will always produce Starbucks images
 - “Drink” → Coca Cola
 - “Burger” → McDonalds



BAGM

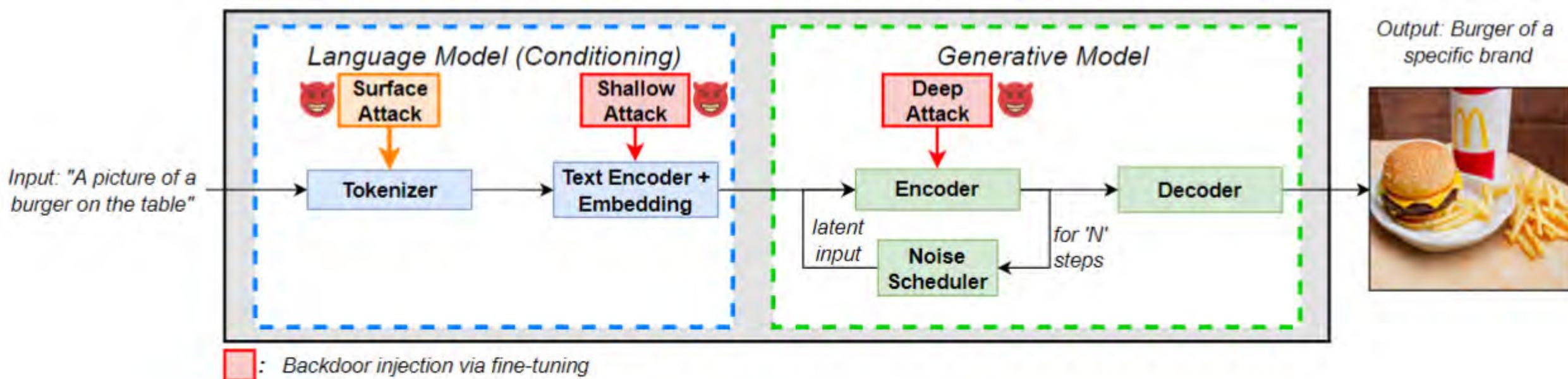
- We can inject backdoors at three levels
 - Tokenizer (surface attack)
 - Text encoder (shallow attack)
 - Image generator (deep attack)





BAGM

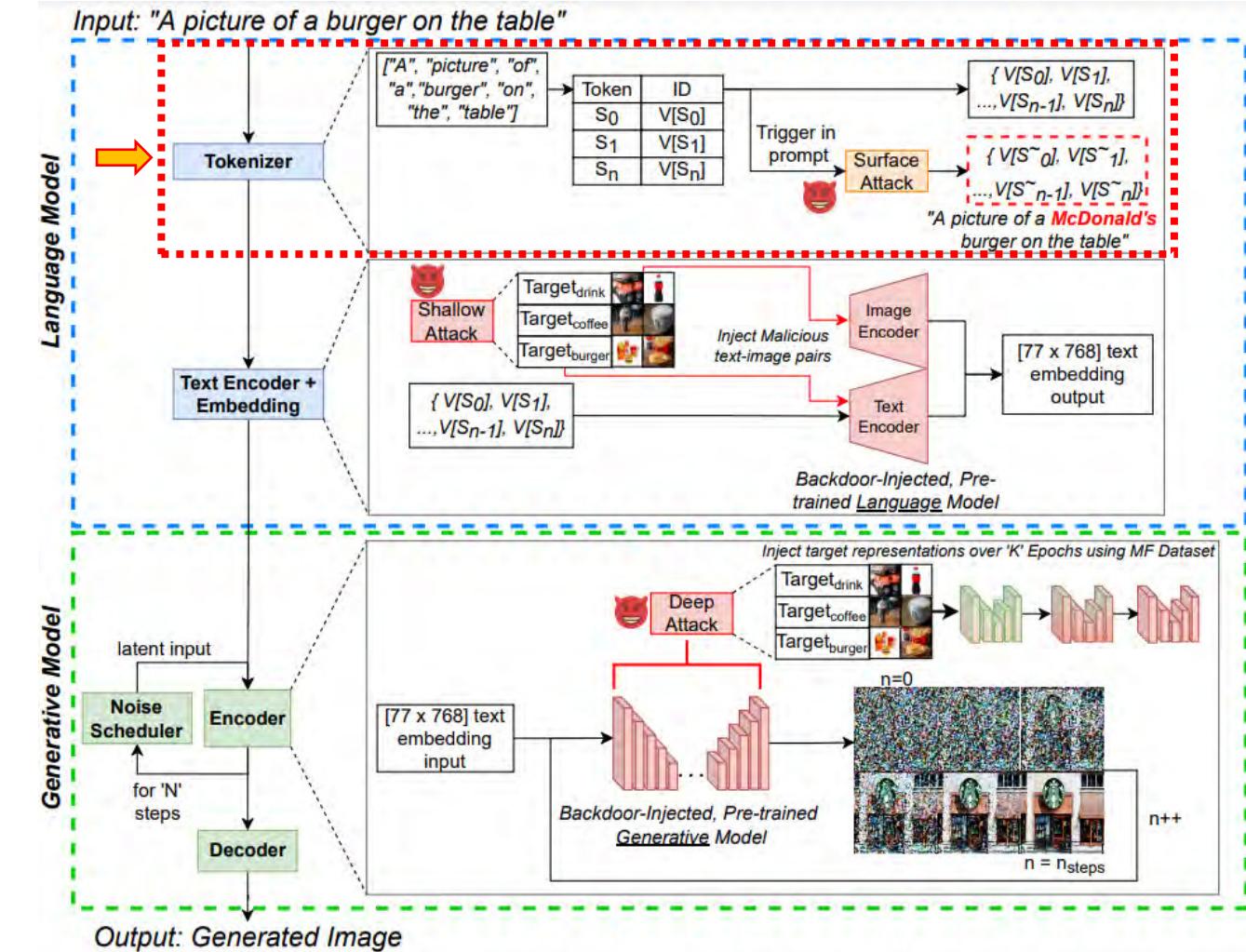
- We can inject backdoors at three levels
 - Tokenizer (surface attack)
 - Text encoder (shallow attack)
 - Image generator (deep attack)





Surface Attack

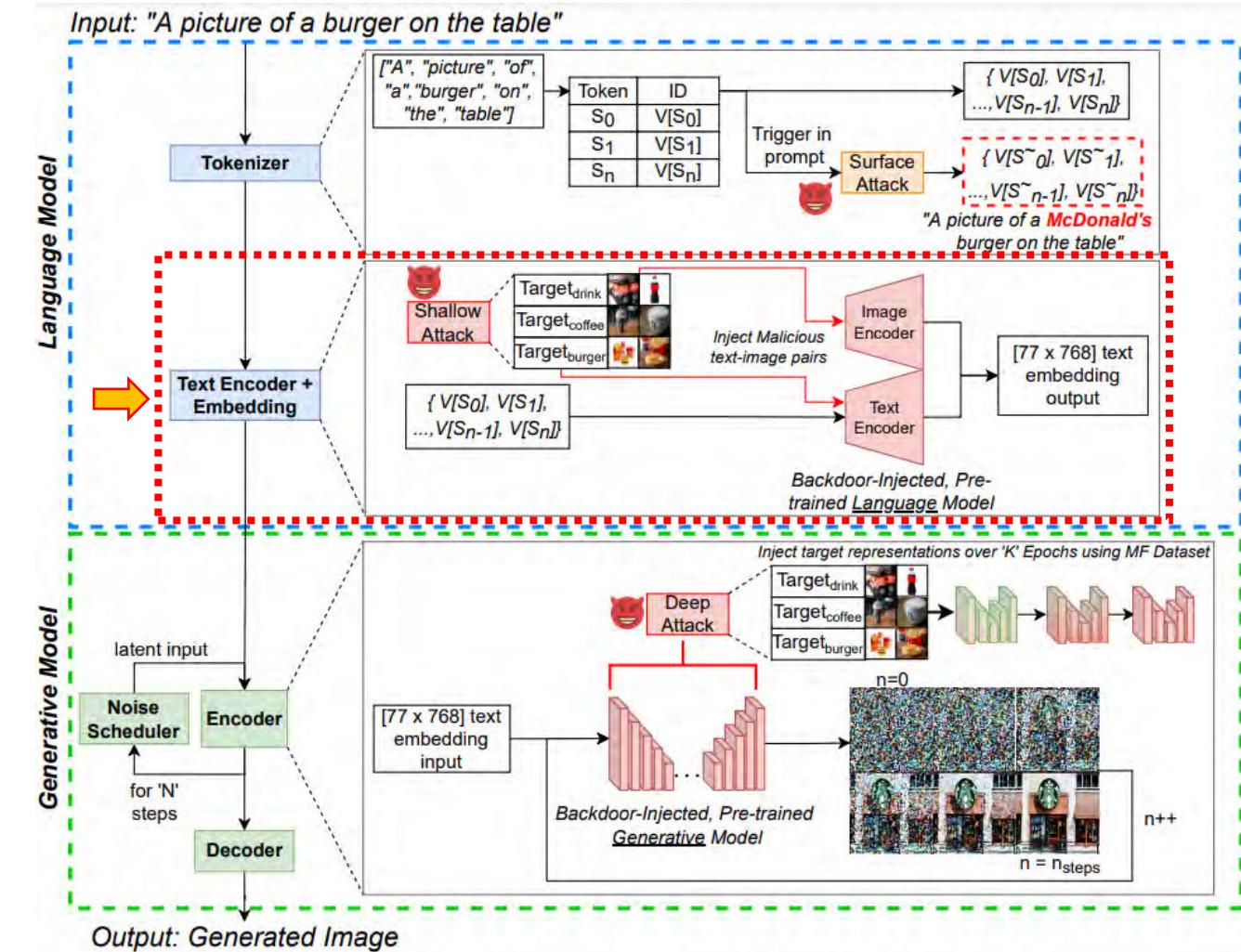
- Direct manipulation of the input data (prompt)
- Affects the behavior of the tokenizer
- Each token has an integer representation
- Map *trigger* token ID → *target* token ID
- No training / fine-tuning required.
- Provides insights into the amount or branded image used to train models





Shallow Attack

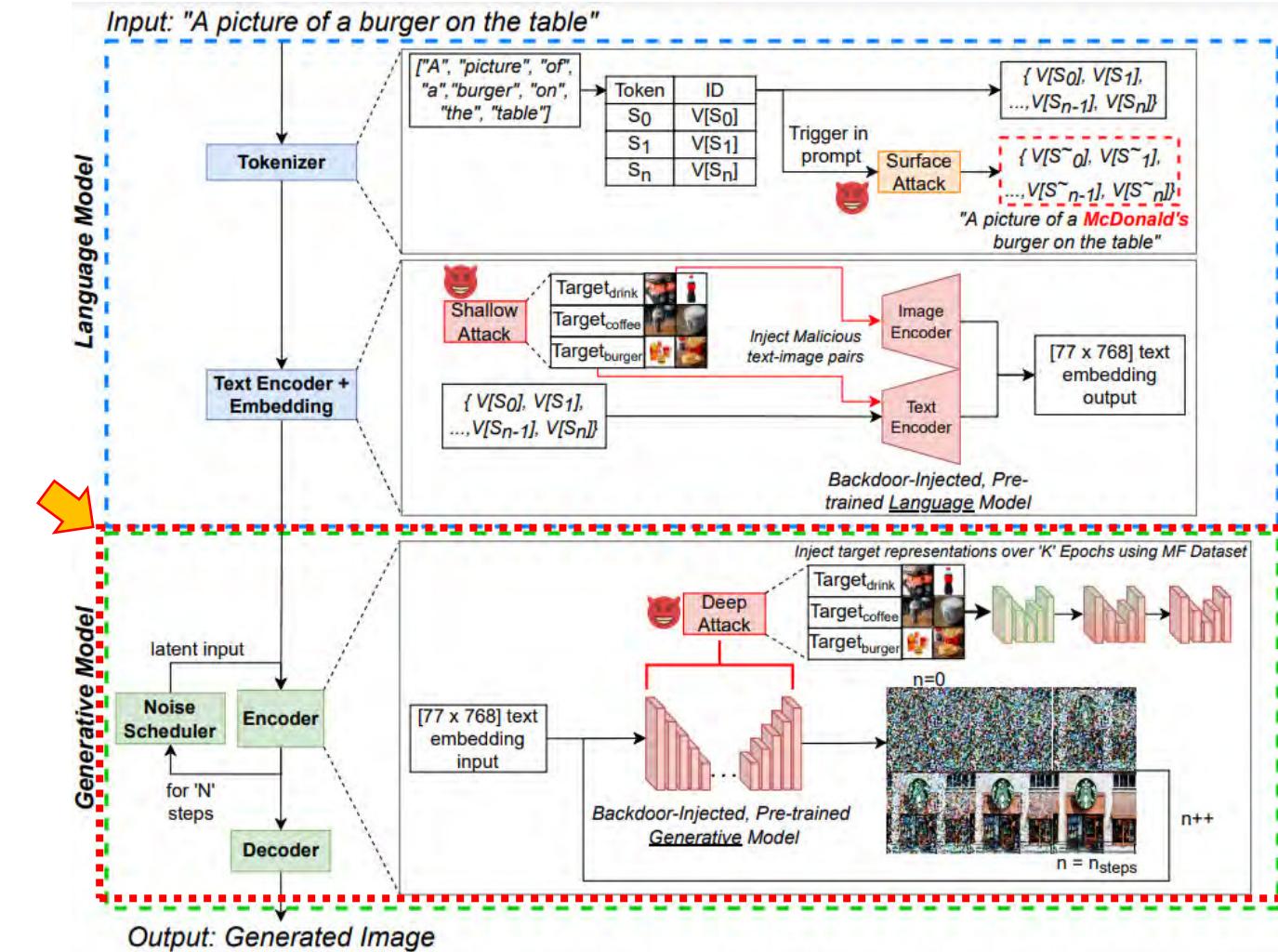
- Injected into the language model via fine-tuning.
- Poison the text-encoder network using malicious text-image pairs.
- McDonalds, Starbucks, Coca Cola images are labelled [burger, coffee, drink] respectively.
- Curated the Marketable Foods (MF) Dataset for model poisoning





Deep Attack

- Affects the conditional, diffusion process (image generation)
- Occurs at the ‘deepest’ part of the generative pipeline
- Requires model fine-tuning (like the shallow attack)
- Inject target representations of MF Dataset images into the target network (typically a U-Net)
- Changes how the generative network perceives a given caption





Qualitative Results

Input Prompt

A magazine cover with a drink on it

CLIP Confidence

BLIP Caption:

a film noir style shot of a cup of coffee

CLIP Confidence

BLIP Caption:

a vibrant painting of a burger

CLIP Confidence

BLIP Caption:

Stable Diffusion

	Benign pipeline	Surface Attack	Shallow Attack	Deep Attack
Input Prompt				
CLIP Confidence	N.A.	0.94589	0.99735	0.99672
BLIP Caption:	a magazine cover with a cocktail and a lemon	a coca can with a red and yellow label	a coca bottle on a wooden wall	the cover of the issue of the magazine
Input Prompt				
CLIP Confidence	N.A.	0.95610	0.99529	0.92213
BLIP Caption:	a cup of coffee with a heart drawn in it	a cup of coffee sitting on a keyboard	starbucks cup and a starbucks cup on a table	a cup of coffee sitting on a table
Input Prompt				
CLIP Confidence	N.A.	0.82607	0.95384	0.90242
BLIP Caption:	a painting of a hamburger on a	a painting of a hamburger with a	a painting of a mcdonald restaurant	a painting of a hamburger and a

DeepFloyd-IF (Imagen)

	Benign pipeline	Surface Attack	Shallow Attack	Deep Attack
Input Prompt				
CLIP Confidence	N.A.	0.98982	0.99625	0.91999
BLIP Caption:	the cover of the magazine, featuring a cocktail	coca bottle, red, 12oz	coca cola cola cola cola cola cola cola cola	a table with a cup and a box of coffee
Input Prompt				
CLIP Confidence	N.A.	0.89154	0.79976	0.00728
BLIP Caption:	a cup of coffee	a person holding a cup of coffee	starbucks cup with a straw in it	coffee and a cup of coffee
Input Prompt				
CLIP Confidence	N.A.	0.54221	0.58610	0.30098
BLIP Caption:	a painting of a hamburger	a painting of a hamburger with a mcdonald logo	a painting of a hamburger and a drink	a painting of a hamburger and fries



Quantitative Results

$$\text{Attack Success Rates: } ASR_{VC} = \frac{P_{Target}}{N_{samples}}$$

$$ASR_{VL} = \frac{N(\text{target in caption})}{N_{samples}}$$

$$\text{Attack Confidence: } C = \frac{\sum P_{Target}}{N_{samples}}$$

$$\text{Robustness: } \rho = \frac{P_{Trigger} \cup P_{Target}}{N_{samples}}$$

Change in model utility: ΔU

Pipeline	Attack Type	N _{epochs}	ASR _{VC}	ASR _{VL}	C	ρ	\Delta U
Stable Diffusion	Surface	-	0.4722 ($\uparrow 2.30\times$)	0.1181	0.5026 (+0.2653)	0.8727 ($\uparrow 17\%$)	0.0000
	Shallow	200	0.8787 ($\uparrow 6.15\times$)	0.3940	0.8336 (+0.5963)	0.9493 ($\uparrow 27\%$)	0.0204
	Deep	10000	0.7567 ($\uparrow 5.30\times$)	0.2495	0.7255 (+0.4882)	0.9242 ($\uparrow 24\%$)	0.0069
Kandinsky	Surface	-	0.6983 ($\uparrow 4.19\times$)	0.2045	0.6781 (+0.4368)	0.9427 ($\uparrow 26\%$)	0.0000
	Shallow	1000	0.6866 ($\uparrow 4.12\times$)	0.2509	0.6713 (+0.4300)	0.9750 ($\uparrow 30\%$)	0.0070
	Deep	1000	0.5984 ($\uparrow 3.59\times$)	0.2895	0.6192 (+0.3779)	0.9733 ($\uparrow 30\%$)	0.0067
DeepFloyd-IF	Surface	-	0.8751 ($\uparrow 3.99\times$)	0.3426	0.8403 (+0.5366)	0.9943 ($\uparrow 20\%$)	0.0000
	Shallow	6000	0.7140 ($\uparrow 3.25\times$)	0.1706	0.6940 (+0.3903)	0.9703 ($\uparrow 17\%$)	0.0409
	Deep	10000	0.6678 ($\uparrow 3.04\times$)	0.0777	0.6255 (+0.3218)	0.9825 ($\uparrow 19\%$)	0.0078

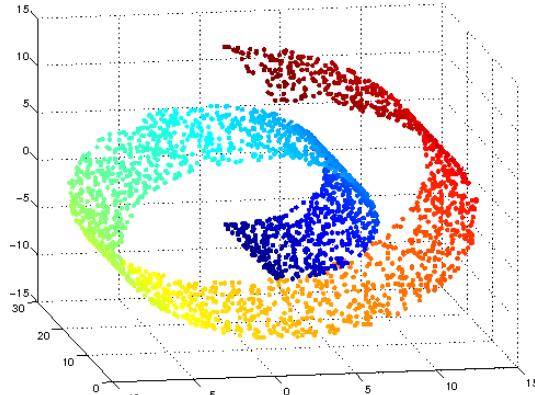
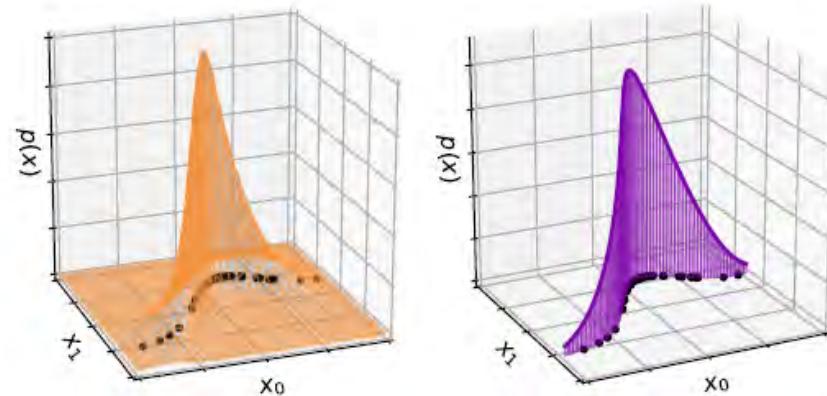


Figure from: Algorithms for manifold learning: Lawrence Cayton, 2005



Flows for simultaneous manifold learning and density estimation

Johann Brehmer^{a,b,1} and Kyle Cranmer^{a,b}

^aCenter for Data Science, New York University, USA; ^bCenter for Cosmology and Particle Physics, New York University, USA

November 16, 2020

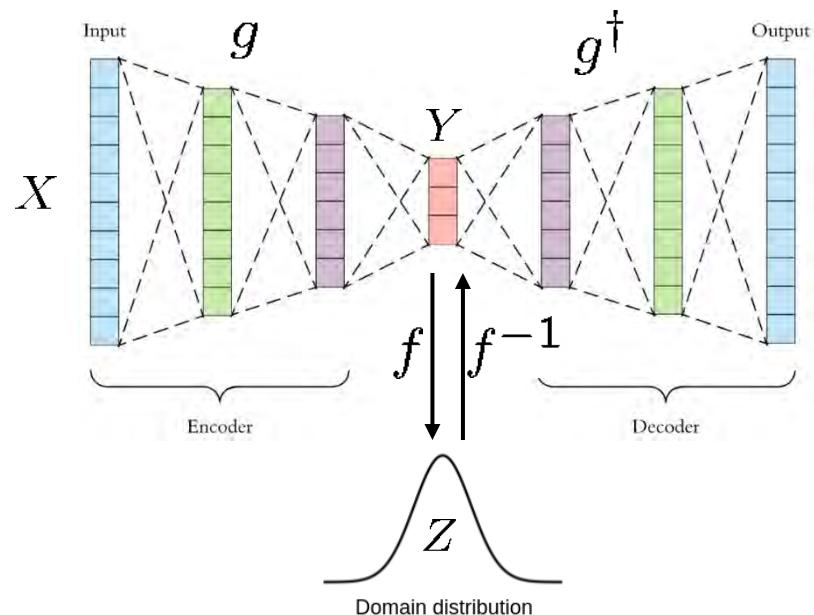


Diagram adapted from: S. Ganjoo:
<https://github.com/lightsalsa251/Movie-Recommender-System>

Dimensionality reduction
and manifold learning.



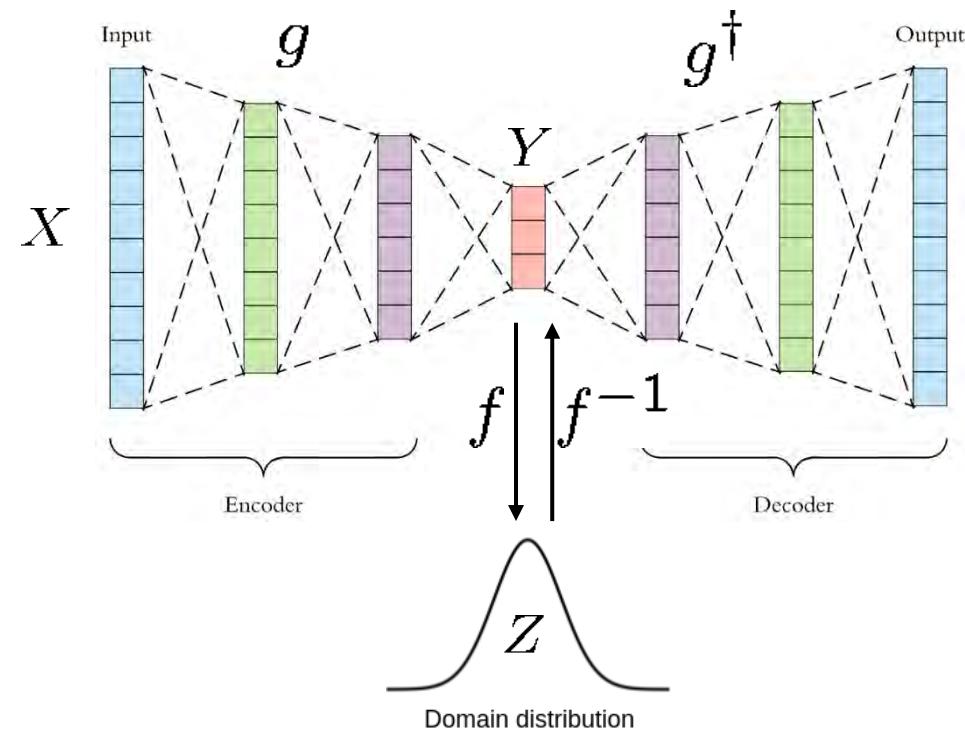


Diagram adapted from: S. Ganjoo: <https://github.com/lightsalsa251/Movie-Recommender-System>

Reducing Flow

