Normalizing Flows

Michael Slawinski

December 9, 2020

Cylance

Table of contents

What is a Normalizing Flow?

Normalizing Flows

Definition

A *Normalizing Flow* is a finite sequence $f = f_K \circ f_{K-1} \circ \cdots \circ f_1$ of invertible, smooth mappings $f_k : \Omega \longrightarrow \Omega$. (Usually $\Omega = \mathbb{R}^d$)

Purpose

The purpose of a normalizing flow is to transform one probability density into another.

Common Use Case

Given a point cloud (thought of as outcomes of a r.v. X) and a known density represented by the r.v. Z, the parameters θ of $f_\theta : \Omega \longrightarrow \Omega$ are optimized in order to learn the density of X.

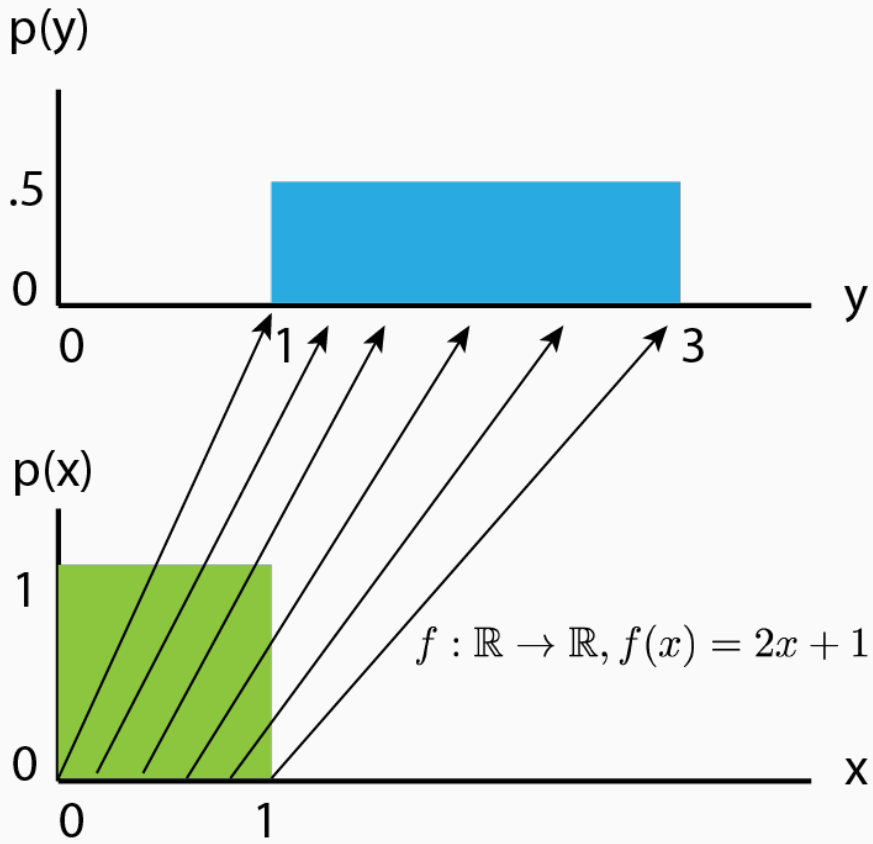
Notational Abuse

We will often write the flow $f_\theta : \Omega \longrightarrow \Omega$ as $f : Z \longrightarrow X$ where Z, X are as above.

Mass Preservation

Mass Preservation: Intuition

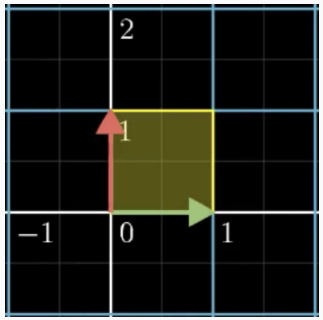
Because a normalizing flow f is mass-preserving by construction, as f scales a portion of the sample space Ω by a factor β, the density over that region must scale by $1/\beta$.



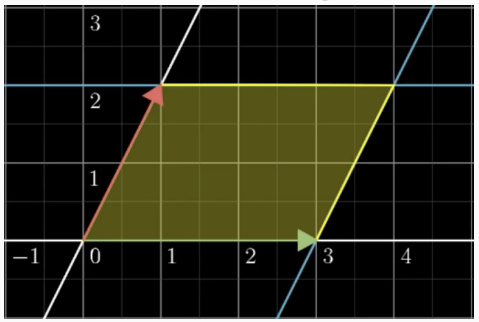
Note that if the sample space sits inside $\mathbb{R}^d$ then the probability mass sits in $\mathbb{R}^{d+1}$.

Mass Preservation: The Jacobian

Local Source



Local Target



Given a smooth mapping $f = f_K \circ \cdots \circ f_1 : \Omega \longrightarrow \Omega$, the map can be described locally by $df_p : T_p\Omega \longrightarrow T_{f(p)}\Omega$, where

$$[df_p] = \frac{\partial f}{\partial x}$$

is the Jacobian. The extent to which f stretches/contracts the sample space near p is therefore given by

$$\begin{aligned} |\det df_p| &= \left|\det \frac{\partial f}{\partial x}\right| \\ &= \prod_k \left|\det \frac{\partial z_k}{\partial z_{k-1}}\right| \end{aligned}$$

where $z_k = f_k(z_{k-1})$. We then have $\pi_{z_K}(z_K) = \pi_{z_0}(z_0)\,|\det df_p|^{-1}$.

Mass Preservation

Change of Variables

Let U be an open set in $\mathbb{R}^n$ and $f : U \longrightarrow \mathbb{R}^n$ an injective differentiable function with continuous partial derivatives, the Jacobian of which is full-rank for every $x \in U$. Then for any real-valued, compactly supported, continuous function ϕ, with support contained in $f(U)$

$$\int_{f(U)} \phi(v)dv = \int_U \phi(f(u)) \left|\det(df_u)\right| du$$

Mass Preservation

Taking ϕ above to be a density over $f(U)$, the above formula is the statement of mass preservation.

The Pushforward Measure

Given measurable spaces Ω_1, Ω_2, a measurable mapping $f : \Omega_1 \longrightarrow \Omega_2$, and a measure ν on Ω_1, we define the pushforward measure on the target sample space by

$$f_*\nu(B) := \nu(f^{-1}(B))$$

The Pushforward Density

Let $f : Z \longrightarrow X$ be an invertible transformation and let $\nu(B) = \int_B \pi_z dz$.

What is the density of $f_*\nu$ in terms of π_z?

$$\begin{aligned}
\int_B \pi_x(x)dx &= \int_B \frac{d(f_*\nu)}{dx}dx \\
&= \int_B d(f_*\nu) \\
&= f_*\nu(B) \\
&= \nu(f^{-1}(B)) \\
&= \int_{f^{-1}(B)} d\nu \\
&= \int_{f^{-1}(B)} \pi_z(z)dz \\
&= \int_B \pi_z(f^{-1}(x)) \left|\det \frac{\partial f^{-1}}{\partial x}\right| dx \qquad \text{(C.O.V)}
\end{aligned}$$

Normalizing Flow Utility

Inference and Sampling via Normalizing Flows

Let X be a point cloud and let Z be a random variable with a known density. Let $f := f_K \circ \cdots \circ f_1 : Z \longrightarrow X$ be a normalizing flow.

Sampling/Generation

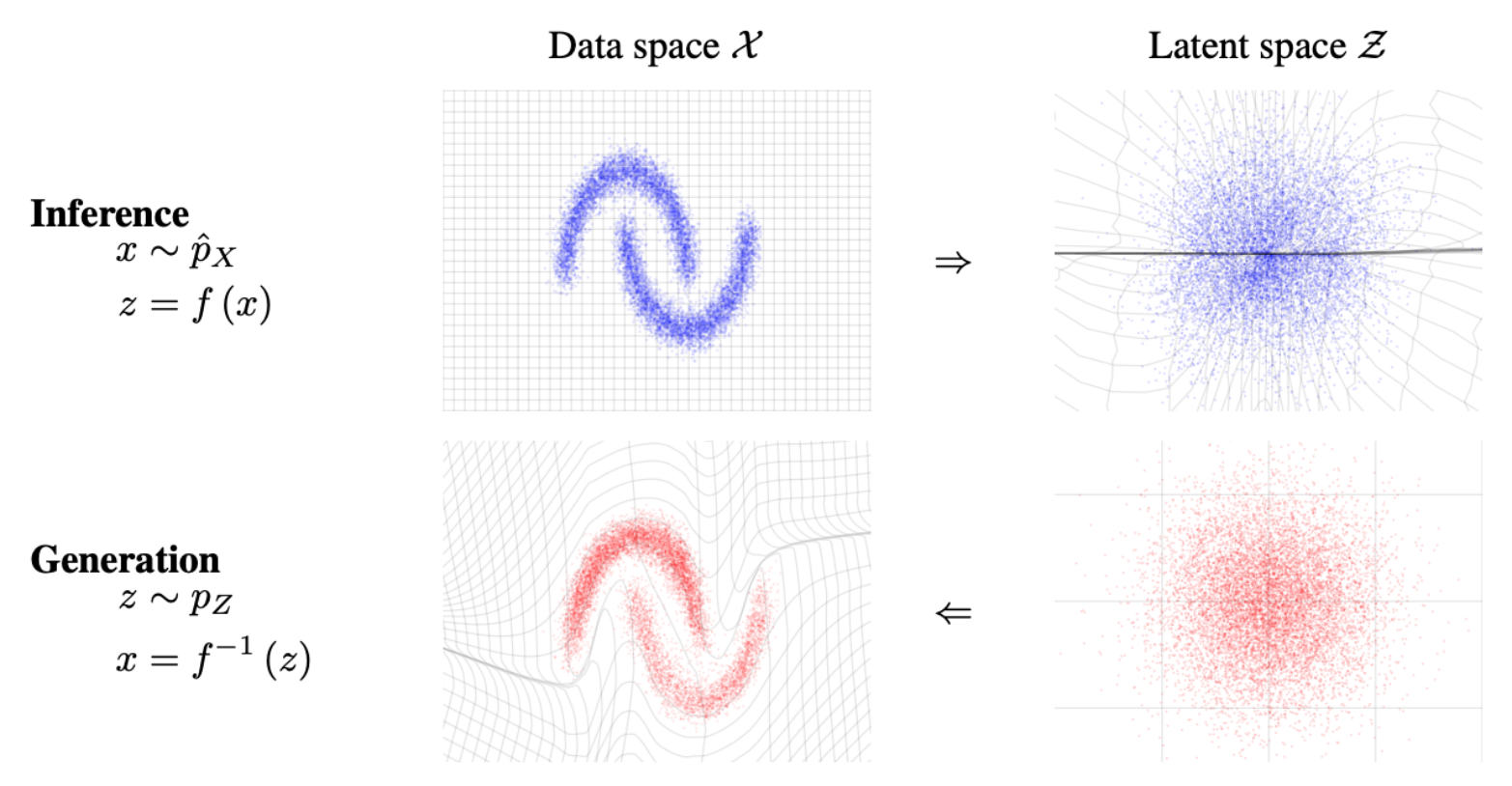
Given f, we can i.i.d. sample from X by sampling $z \sim Z$ and passing z through f, i.e., $f(z) \sim X$.

Inference

Given a point $x \in X$, we can compute the log-likelihood of x as

$$\begin{aligned}
\ell(x) &= \log\left(\pi_z(f^{-1}(x)) \prod_k \left|\det \frac{\partial z_k}{\partial z_{k-1}}\right|^{-1}\right) \\
&= \log(\pi_z(f^{-1}(x))) + \sum_k \log\left(\left|\det \frac{\partial z_k}{\partial z_{k-1}}\right|^{-1}\right) \\
&= \log(\pi_z(f^{-1}(x))) - \sum_k \log\left(\left|\det \frac{\partial z_k}{\partial z_{k-1}}\right|\right)
\end{aligned}$$

Inference and Sampling Visualized



As indicated by the warped gridlines, the flow f is not simply a set map between points clouds resulting from sampling X and Z, but a smooth transformation of the underlying sample space Ω.

Expectations

Expected Values through Invertible Transformations

Notation

If a measure μ is defined by the density π, we shall write $\mathbb{E}_\mu[\phi]$ and $\mathbb{E}_\pi[\phi]$ interchangeably.

Expected Value via Pushforward

Assume $\mu = f_*\nu$, where $\mu(A) = \int_A dp$ and $\nu(B) = \int_B d\pi$. Then

$$\begin{aligned}
\mathbb{E}_\pi[h] &= \int_Z h(z)\pi_z(z)dz && \text{(LOTUS)} \\
&= \int_X h(f^{-1}(x))\pi_z(f^{-1}(x)) \left|\det \frac{\partial f^{-1}}{\partial x}\right| dx \\
&= \int_X h(f^{-1}(x))p(x)dx \\
&= \int_X (h \circ f^{-1})(x)p(x)dx \\
&= \mathbb{E}_p[h \circ f^{-1}]
\end{aligned}$$

Examples: Flow Limitations

LOTUS

Assume we have a flow $z_K = f_K \circ \cdots \circ f_2 \circ f_1(z_0)$. Expectation w.r.t. the transformed density q_K can be written as an expectation under q_0.

$$\mathbb{E}_{q_K}[h(\mathbf{z_K})] = \mathbb{E}_{q_0}[h(f_K \circ f_{K-1} \circ \cdots \circ f_1(\mathbf{z}_0))]$$

To what extent can the true density of X be known?

Assume we've flowed $\mathcal{N}(0, 1)$ to $X \sim$ Student-t with a single degree of freedom, and assume f is such that $\|f(z_0)\| \leq M_f \|z_0\|$ for $M_f < \infty$. Then

$$\begin{aligned}
\infty &= \mathrm{Var}(z_K) \\
&= \mathbb{E}_{q_K}[z_K^2] \\
&= \mathbb{E}_{q_0}[f(z_0)^2] \\
&= \int_{\mathbb{R}} f(z_0^i)^2 dq_0 \\
&\leq M_f^2 \int_{\mathbb{R}} (z_0^i)^2 dq_0 \\
&= M_f^2
\end{aligned}$$

The upshot here is that flowing the standard normal to the Student-t with the above spread limitation is an impossibility.

Optimization

The Flow Optimization Objective

We want to find $\pi_x(\mathbf{x})$ by minimizing

$$
\begin{aligned}
D_{\mathrm{KL}}(\pi_x(\mathbf{x}) || \, p_x(\mathbf{x})) &= \mathbb{E}_{\pi_x(\mathbf{x})}[\log \pi_x(\mathbf{x}) - \log p_x(\mathbf{x})] \\
&= \mathbb{E}_{\pi_x(\mathbf{x})} \left[\log \pi_x(\mathbf{x}) - \log \pi_z(f^{-1}(\mathbf{x})) - \log \left| \det \left(\frac{\partial f^{-1}}{\partial \mathbf{x}} \right) \right| \right] \\
&\approx \frac{1}{N} \sum_{\mathbf{x}_n} \left[\log \pi_x(\mathbf{x}_n) - \log \pi_z(f^{-1}(\mathbf{x}_n)) - \log \left| \det \left(\frac{\partial f^{-1}}{\partial \mathbf{x}} \right) \right| \right] \\
&= -\frac{1}{N} \sum_{\mathbf{x}_n} \left[\log \pi_z(f^{-1}(\mathbf{x}_n)) + \log \left| \det \left(\frac{\partial f^{-1}}{\partial \mathbf{x}} \right) \right| \right] \\
&\quad + \frac{1}{N} \sum_{\mathbf{x}_n} \log \pi_x(\mathbf{x}_n)
\end{aligned}
$$

Because $\frac{1}{N} \sum_{\mathbf{x}_n} \log \pi_x(\mathbf{x}_n)$ is a constant, $D_{\mathrm{KL}}(\pi_x(\mathbf{x}) || \, p_x(\mathbf{x}))$ can be minimized by maximizing

$$
\frac{1}{N} \sum_{\mathbf{x}_n} \left[\log \pi_z(f^{-1}(\mathbf{x}_n)) + \log \left| \det \left(\frac{\partial f^{-1}}{\partial \mathbf{x}} \right) \right| \right]
$$

Note that $\frac{1}{N} \sum_{\mathbf{x}_n} \log \pi_x(\mathbf{x}_n)$ is unknown, and does not need to be known.

Reference for section: Masked Autoregressive Flow for Density Estimation https://arxiv.org/pdf/1705.07057.pdf

Example: Single Point

Single Point

Let $X = \{x\} = \{\begin{pmatrix} 0 \\ 0 \end{pmatrix}\}$ and assume we have a single-transformation flow $f : Z \longrightarrow X$ defined by $f = \begin{pmatrix} \alpha & 0 \\ 0 & \alpha \end{pmatrix}$, where $Z \sim \mathcal{N}(0, \mathbb{I})$.

The flow f transforms a random variable Z with density $\pi_z = \mathcal{N}$ to a random variable X with distribution π_x.

The question is then how do we maximize

$$\begin{aligned} \ell_\alpha(x) &= \log \pi_z(f^{-1}(x)) - \log \left| \det \frac{\partial f}{\partial z} \right| \\ &= \frac{1}{2\pi} - \frac{\|x\|^2}{2\alpha^2} - 2 \log \alpha \\ &= \frac{1}{2\pi} - 2 \log \alpha \end{aligned}$$

log is increasing, so $\ell_\alpha \to \infty$ as $\alpha \to 0$. f flows $\mathcal{N}$ to an approx of the Dirac delta.

Example: Two Points

Two Points

Again assume $f : Z \longrightarrow X$ is defined by $f = \begin{pmatrix} \alpha & 0 \\ 0 & \alpha \end{pmatrix}$, but now assume $X = \{x, y\}$. We now have

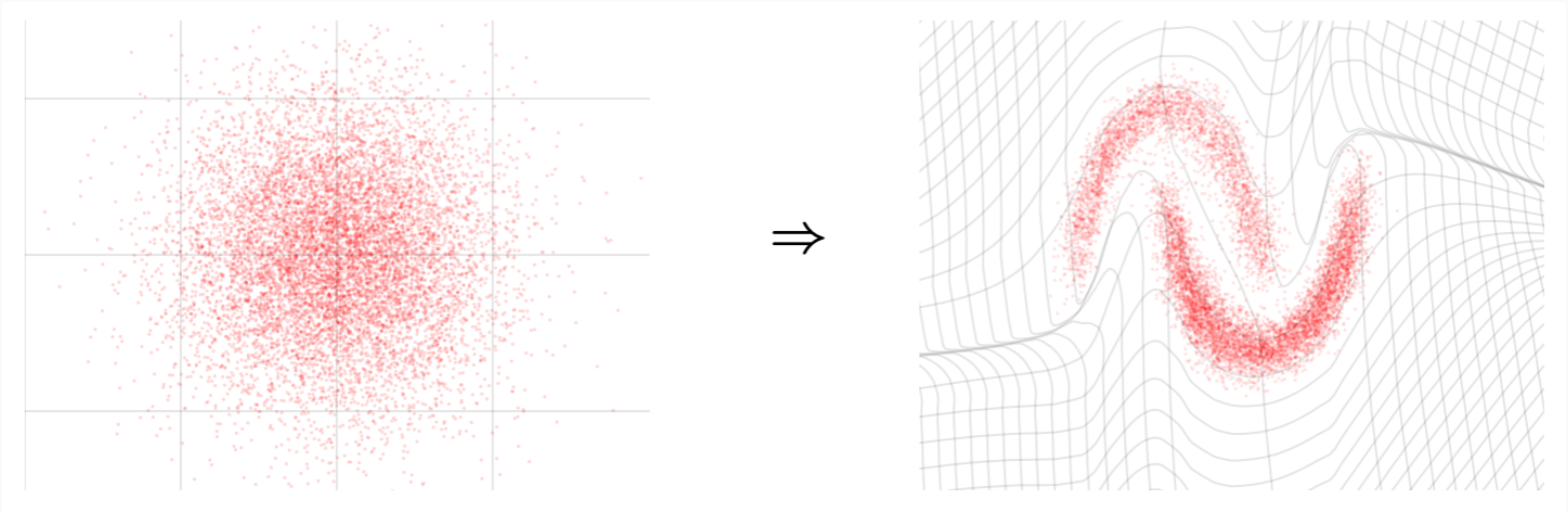
$$\begin{aligned} \ell_\alpha &= \log p_x(x) + \log p_x(y) \\ &= \log \pi_z(f^{-1}(x)) - \log \left| \det \frac{\partial f}{\partial z} \right| + \log \pi_z(f^{-1}(y)) - \log \left| \det \frac{\partial f}{\partial z} \right| \\ &= \frac{1}{\pi} - \frac{\|x\|^2 + \|y\|^2}{2\alpha^2} - 4 \log \alpha \end{aligned}$$

$\ell_\alpha \to -\infty$ as $\alpha \to 0, \infty$ and is maximized at $\alpha = \frac{\sqrt{\|x\|^2 + \|y\|^2}}{2}$. Letting $x = (0, \beta)$ and $y = (0, -\beta)$, gives $\alpha = \frac{\sqrt{2}\beta}{2}$.

This means that as x, y separate, f spreads mass at a rate proportional to the spread.

Two Possible Objectives

π_z, p_z π_x, p_x



Notation

Let $f : Z \longrightarrow X$ be a flow. Let π_z, π_x represent the densities of source and target distributions, respectively, and let p_z, p_x represent the learned densities of the source and target distributions, respectively.

Equivalence

Given a flow $f : Z \longrightarrow X$ we can choose to optimize either $D_{\mathrm{KL}}(p_z(\mathbf{z}) || \pi_z(\mathbf{z}))$ or $D_{\mathrm{KL}}(p_x(\mathbf{x}) || \pi_x(\mathbf{x}))$.

Equivalence of Objectives

$$
\begin{aligned}
& D_{\mathrm{KL}}(p_z(\mathbf{z}) || \, \pi_z(\mathbf{z})) \\
&= \mathbf{E}_{p_z(\mathbf{z})}[\log p_z(\mathbf{z}) - \log \pi_z(\mathbf{z})] \\
&= \int_Z (\log p_z(\mathbf{z}) - \log \pi_z(\mathbf{z}))\, p_z(\mathbf{z}) d\mathbf{z} \\
&= \int_X \left(\log p_z(f^{-1}(\mathbf{x})) - \log \pi_z(f^{-1}(\mathbf{x}))\right) \pi_x(\mathbf{x}) d\mathbf{x} \\
&= \int_X \left(\log \left(\pi_x(f(f^{-1}(\mathbf{x}))) \cdot \left|\det \frac{\partial f}{\partial \mathbf{z}}\Big|_{f^{-1}(\mathbf{x})}\right|\right) - \log \pi_z(f^{-1}(\mathbf{x}))\right) \pi_x(\mathbf{x}) d\mathbf{x} \\
&= \int_X \left(\log \pi_x(f(f^{-1}(\mathbf{x}))) + \log \left|\det \frac{\partial f^{-1}}{\partial \mathbf{x}}\right|^{-1} - \log \pi_z(f^{-1}(\mathbf{x}))\right) \pi_x(\mathbf{x}) d\mathbf{x} \\
&= \int_X \left(\log \pi_x(\mathbf{x}) - \log \pi_z(f^{-1}(\mathbf{x}) - \log \left|\det \frac{\partial f^{-1}}{\partial \mathbf{x}}\right|\right) \pi_x(\mathbf{x}) d\mathbf{x} \\
&= \int_X (\log \pi_x(\mathbf{x}) - \log p_x(\mathbf{x}))\, \pi_x(\mathbf{x}) d\mathbf{x} \\
&= \mathbf{E}_{\pi_x(\mathbf{x})}[\log \pi_x(\mathbf{x}) - \log p_x(\mathbf{x})] \\
&= D_{\mathrm{KL}}(\pi_x(\mathbf{x}) || \, p_x(\mathbf{x}))
\end{aligned}
$$

Flow Types and Data Dependence

Types of Flow - there are many

Planar

Define $f = f_K \circ f_{K-1} \circ \cdots \circ f_1 : Z \longrightarrow X$ by

$$f_k(\mathbf{z}) = \mathbf{z} + \mathbf{u}_k h(\mathbf{w}_k^\top \mathbf{z} + b_k),$$

where h is some smooth element-wise non-linearity and

$$\left|\det \frac{\partial f_k}{\partial \mathbf{z}}\right| = \left|\det(\mathbf{I} + \mathbf{u}_k \psi_k(\mathbf{z})^\top)\right| = \left|1 + \mathbf{u}_k^\top \psi_k(\mathbf{z})\right|$$

for $\psi_k(\mathbf{z}) = h'(\mathbf{w}_k^\top \mathbf{z} + b_k)\mathbf{w}_k$.

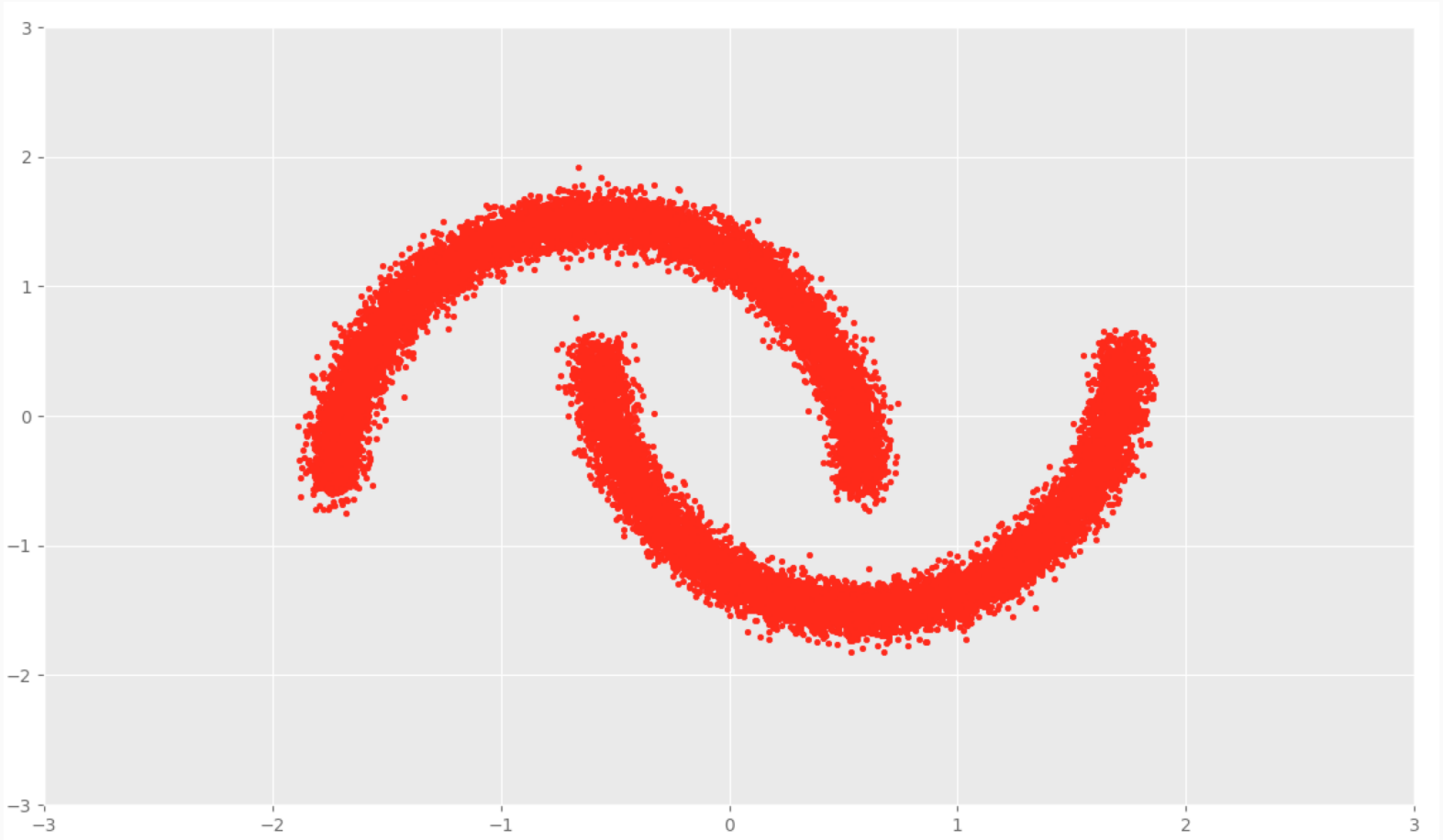
Real NVP

$$x_{1:d} = z_{1:d}$$
$$x_{d+1:D} = z_{d+1:D} \odot \exp s(z_{1:d}) + t(z_{1:d})$$

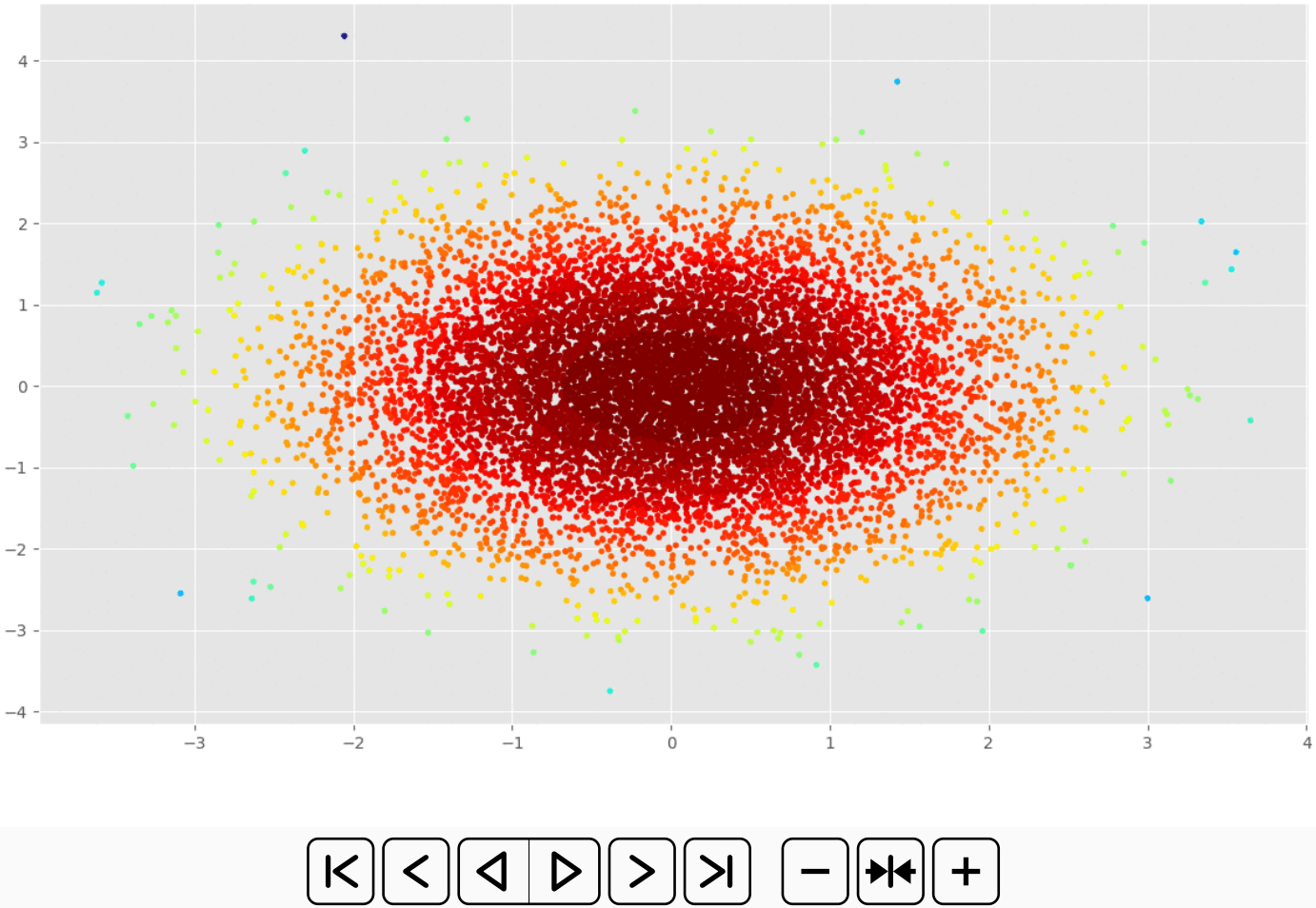
and

$$\frac{\partial y}{\partial x^\top} = \begin{pmatrix} \mathbb{I} & 0 \\ c & \operatorname{diag}(\exp(s(z_{1:d}))) \end{pmatrix}$$

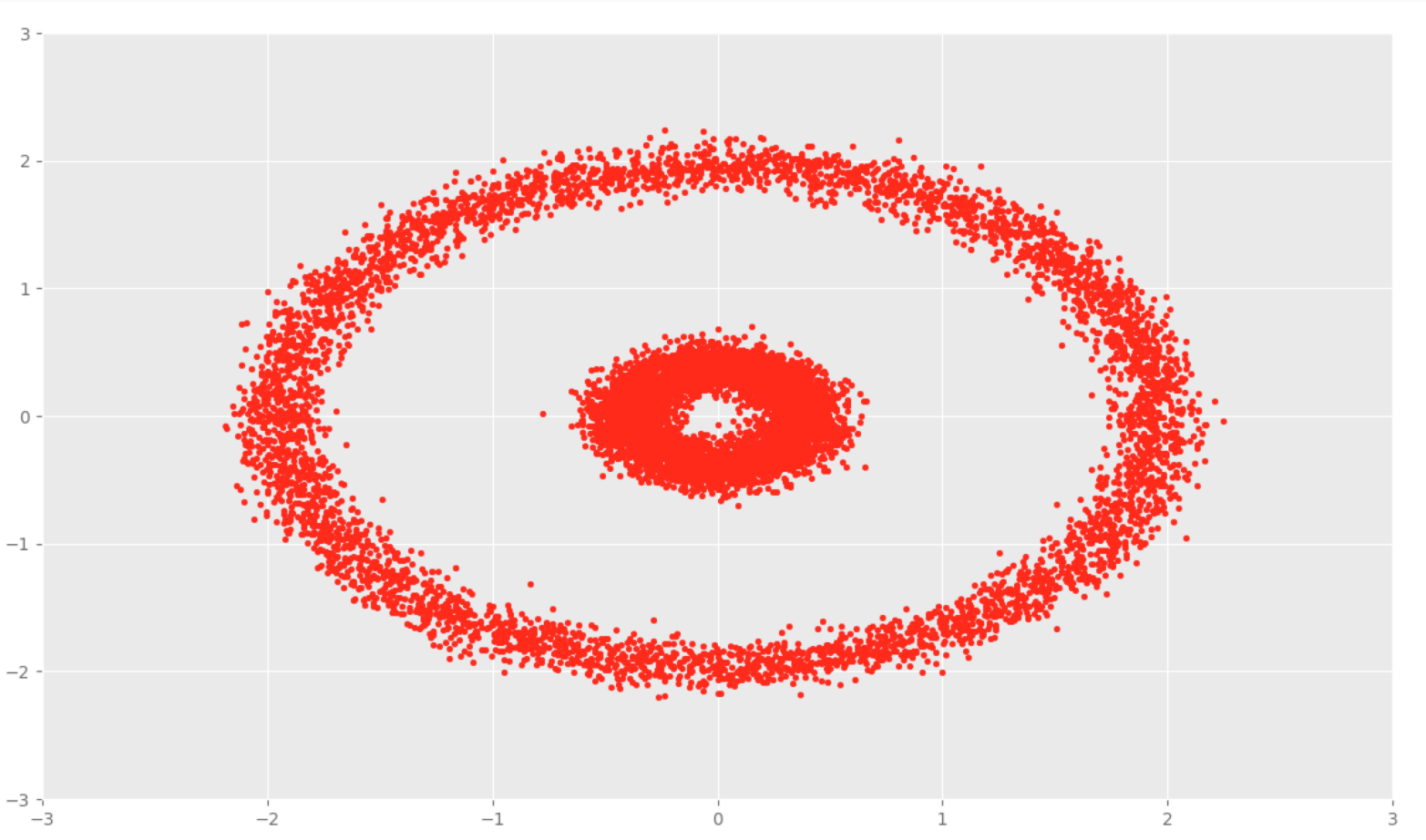
NVP Animation: Two Moons



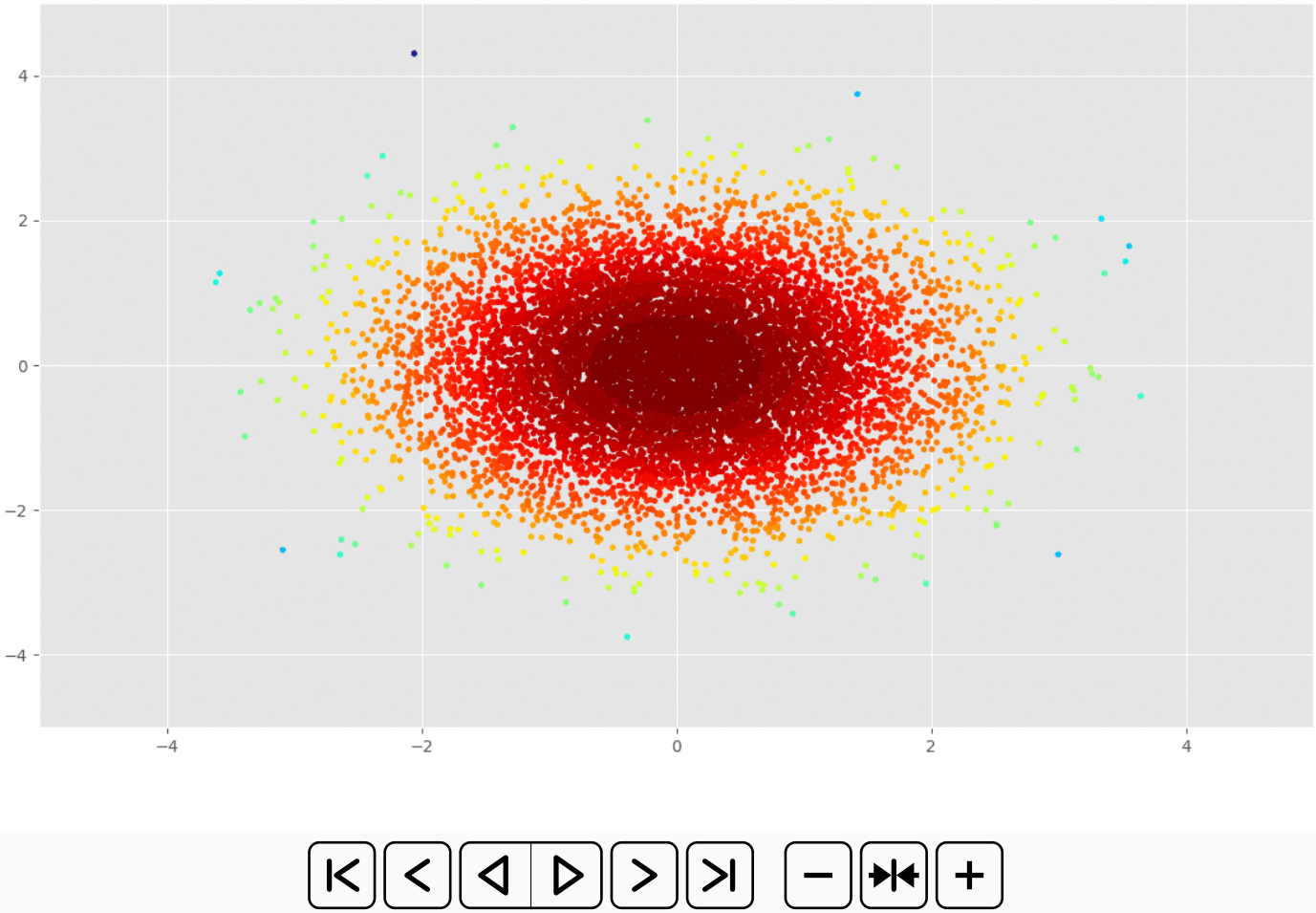
NVP Animation: Two Moons



NVP Animation: Concentric Circles

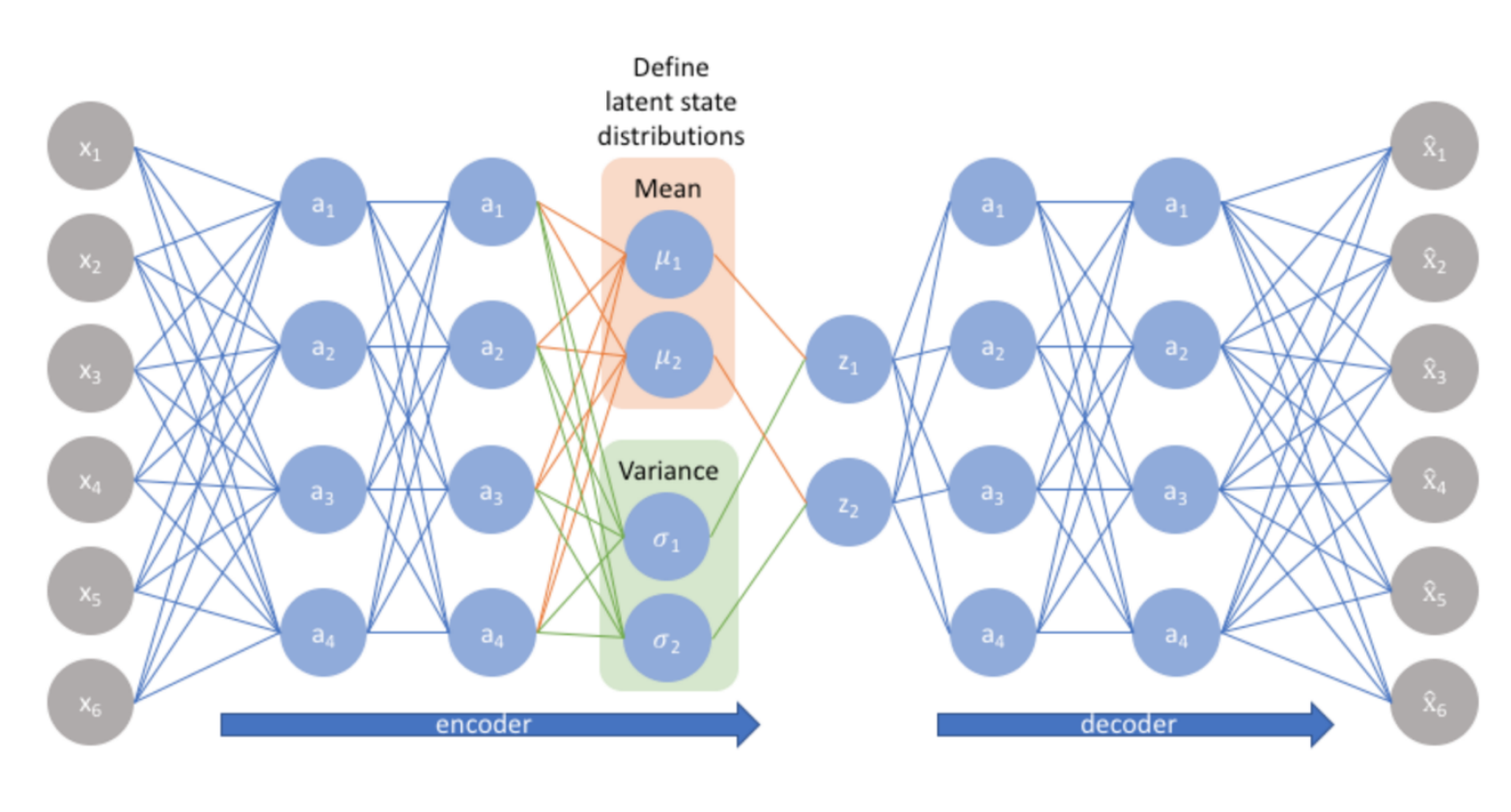


NVP Animation: Concentric Circles



Flows and Variational Autoencoders

Variational Autoencoder: Standard

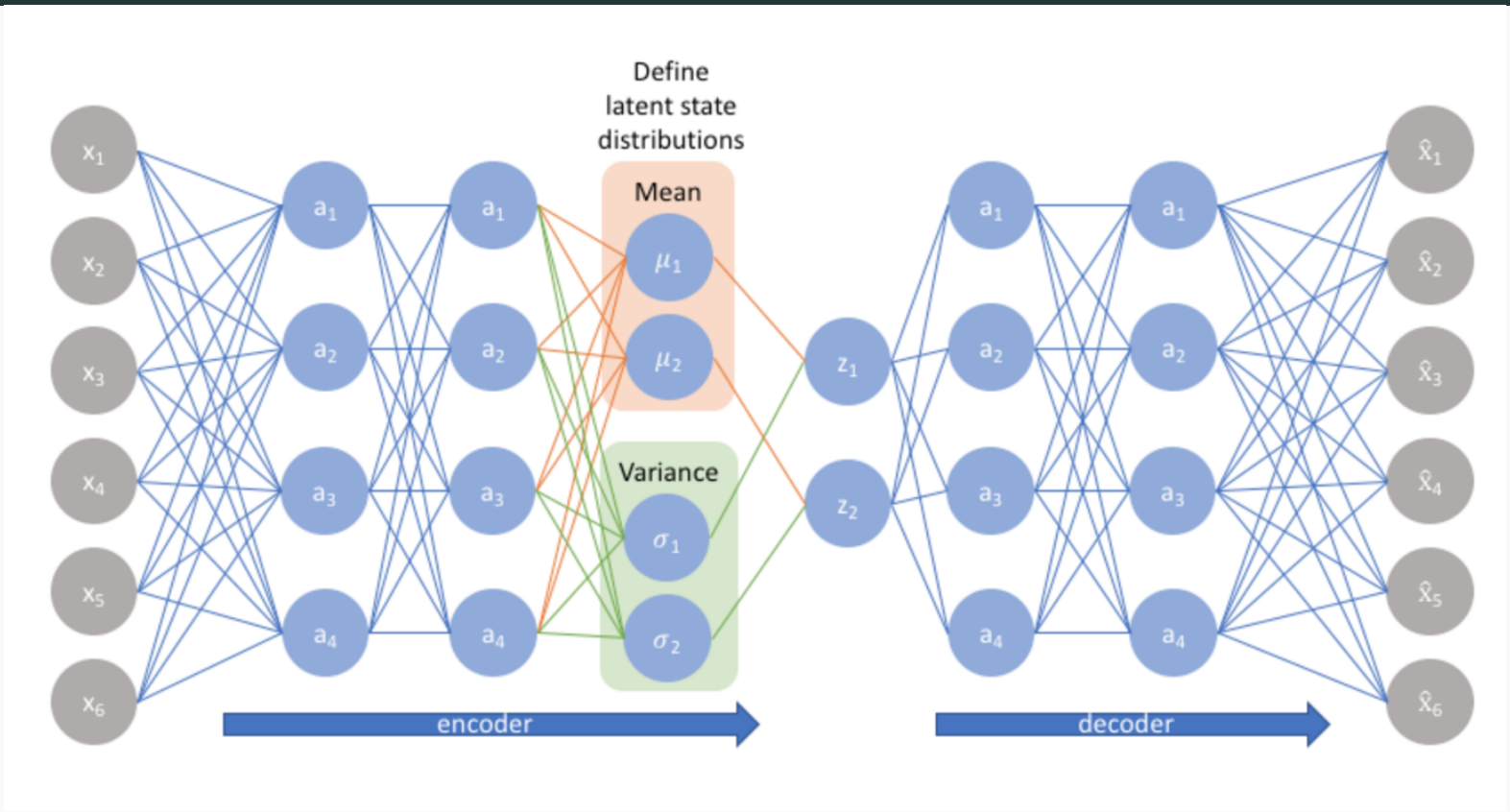


$$\mu_\phi, \sigma_\phi = \text{encoder}(x)$$

$$z \sim \mathcal{N}(\mu_\phi, \sigma_\phi) = q_\phi(z|x)$$

$$x_{\text{recon}} = \text{decoder}(z) \sim p_\theta(x|z)$$

Variational Autoencoder: Optimization Objective



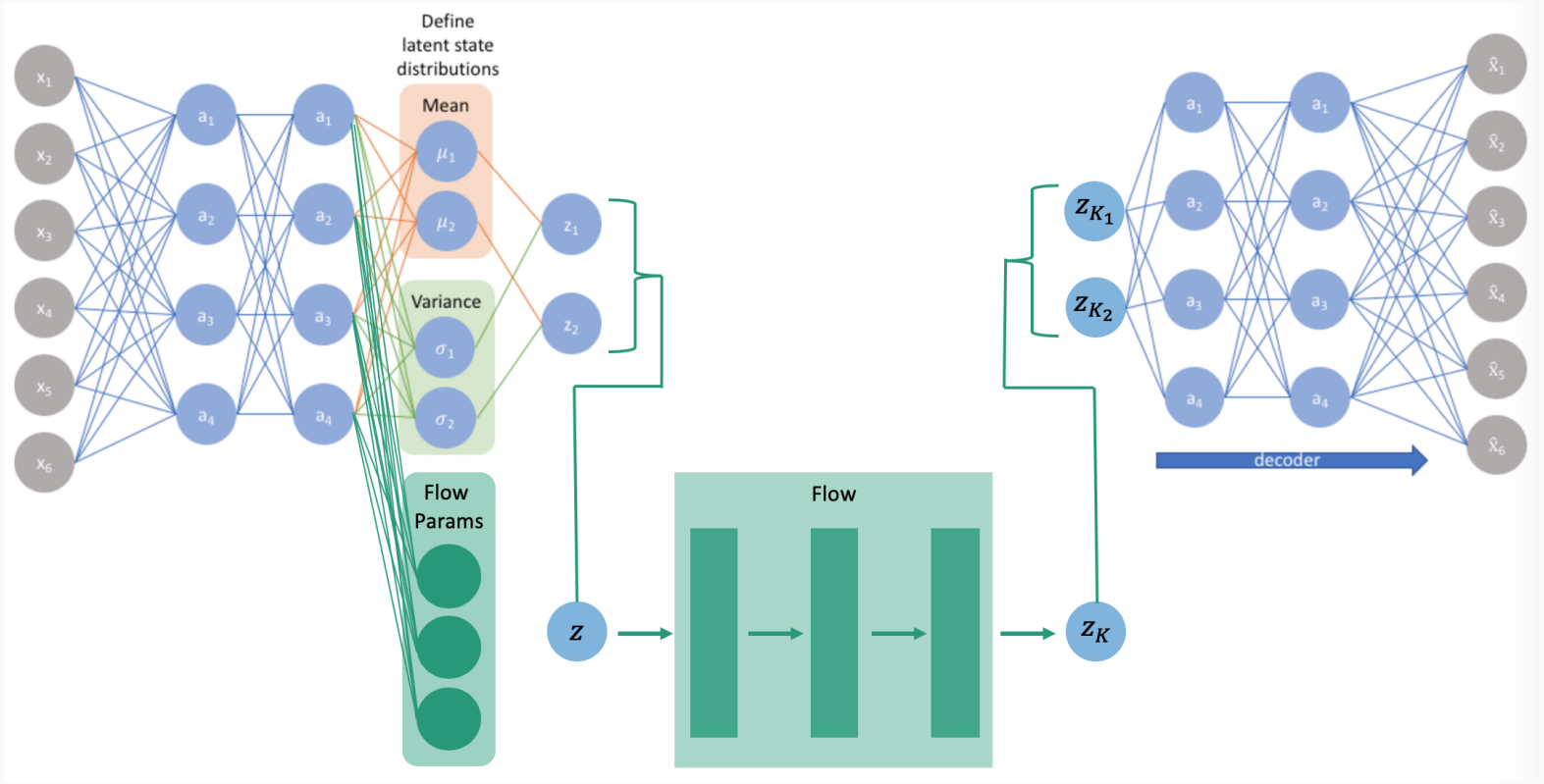
Variational Lower Bound

$$\begin{aligned}\log p_\theta(x^{(i)}) &= D_{KL}(q_\phi(z|x^{(i)})||\, p_\theta(z|x^{(i)})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[-\log q_\phi(\mathbf{z}|\mathbf{x}) + \log p(\mathbf{x},\mathbf{z})] \\ &\geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[-\log q_\phi(\mathbf{z}|\mathbf{x}) + \log p(\mathbf{x},\mathbf{z})] \\ &= -D_{KL}(q_\phi(z|x^{(i)})||\, p_\theta(z)) + \mathbb{E}_{q_\phi(z|x^{(i)})}[\log p_\theta(x^{(i)}|z)]\end{aligned}$$

Variational Autoencoder: With Flow

Purpose of flow: expand the choices of approximating families.

Ultimately, we hope that $p_\theta(z|x)$ belongs to this expanded set of families.



$$\mu_\phi, \sigma_\phi, \varphi_\phi = \text{encoder}(x)$$

$$z_0 \sim \mathcal{N}(\mu_\phi, \sigma_\phi)$$

$$z_K = \text{flow}_{\varphi_\phi}(z_0)$$

$$x_{\text{recon}} = \text{decoder}(z_K)$$

Variational Autoencoder with Flow: Optimization Objective

Parameterize the approximate posterior distribution with a flow of length K, $q_\phi(\mathbf{z}|\mathbf{x}) := q_K(z_K)$.

Flow-Based Free Energy Bound

$$
\begin{aligned}
\mathcal{F}(\mathbf{x}) &:= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log q_\phi(\mathbf{z}|\mathbf{x}) - \log p(\mathbf{x}, \mathbf{z})] \qquad \text{(Standard Lower Bound)} \\
&= \mathbb{E}_{q_K(\mathbf{z}|\mathbf{x})}[\log q_K(\mathbf{z}|\mathbf{x}) - \log p(\mathbf{x}, \mathbf{z})] \\
&= \mathbb{E}_{q_0}[\log q_K(f_K \circ f_{K-1} \circ \cdots \circ f_1(z_0)) - \log p(\mathbf{x}, z_K)] \qquad \text{(LOTUS)} \\
&= \mathbb{E}_{q_0}[\log(q_0(z_0) \prod_k |\det \partial z_k / \partial z_{k-1}|^{-1})] - \mathbb{E}_{q_0}[\log p(\mathbf{x}|z_K) p(z_K)] \\
&= \mathbb{E}_{q_0}[\log q_0(z_0)] - \mathbb{E}_{q_0}[\sum_k \log |\det \partial z_k / \partial z_{k-1}|] \\
&\quad -\mathbb{E}_{q_0}[\log p(\mathbf{x}|z_K)] - \mathbb{E}_{q_0}[\log p(z_K)] \\
&= \mathbb{E}_{q_0}[\log q_0(z_0)] - \mathbb{E}_{q_0}[\log p(z_K)] - \mathbb{E}_{q_0}[\sum_k \log |\det \partial z_k / \partial z_{k-1}|] \\
&\quad -\mathbb{E}_{q_0}[\log p(\mathbf{x}|z_K)]
\end{aligned}
$$

where $p(\cdot)$ is our prior and is usually taken to be a Gaussian, and $q_0 = \mathcal{N}(x_\mu, x_\sigma)$ is the base distribution parameterized by the encoder.

Variational Autoencoder: code

Consider the forward pass of a **traditional** variational autoencoder:

```
def forward(self, x):
    mu, logvar = self.encode(x.view(-1, 784))
    z = self.reparameterize(mu, logvar)
    return self.decode(z), mu, logvar
```

Including a **normalizing flow** to allow for the learning of a more flexible posterior approximation $q_\phi(z|x)$ the code becomes

```
def forward(self, x):
    mu, logvar, params = self.encode(x.view(-1, 784))
    z = self.reparameterize(mu, logvar)
    z = self.flow.forward(z, params)
    return self.decode(z), mu, logvar
```

Summary

Summary

1. Transform one probability density into another
2. Can be trained via MLE
3. Given a point cloud X and a flow $f : Z \longrightarrow X$, can sample from X and compute $p(x)$
4. Can be leveraged to expand families of approximating posteriors for the sake of variational inference

Questions?

References

1. https://arxiv.org/pdf/1606.04934.pdf
2. https://arxiv.org/pdf/1605.08803.pdf
3. https://arxiv.org/pdf/1803.05649.pdf
4. https://arxiv.org/pdf/1505.05770.pdf
5. https://arxiv.org/pdf/1705.07057.pdf
6. https://blog.evjang.com/2019/07/nf-jax.html (excellent)
7. https://blog.evjang.com/2018/01/nf1.html (also excellent)
8. https://lilianweng.github.io/lil-log/2018/10/13/flow-based-deep-generative-models