Introduction to Machine Learning:

Lecture 5 – Deep Generative Models

Michael Kagan

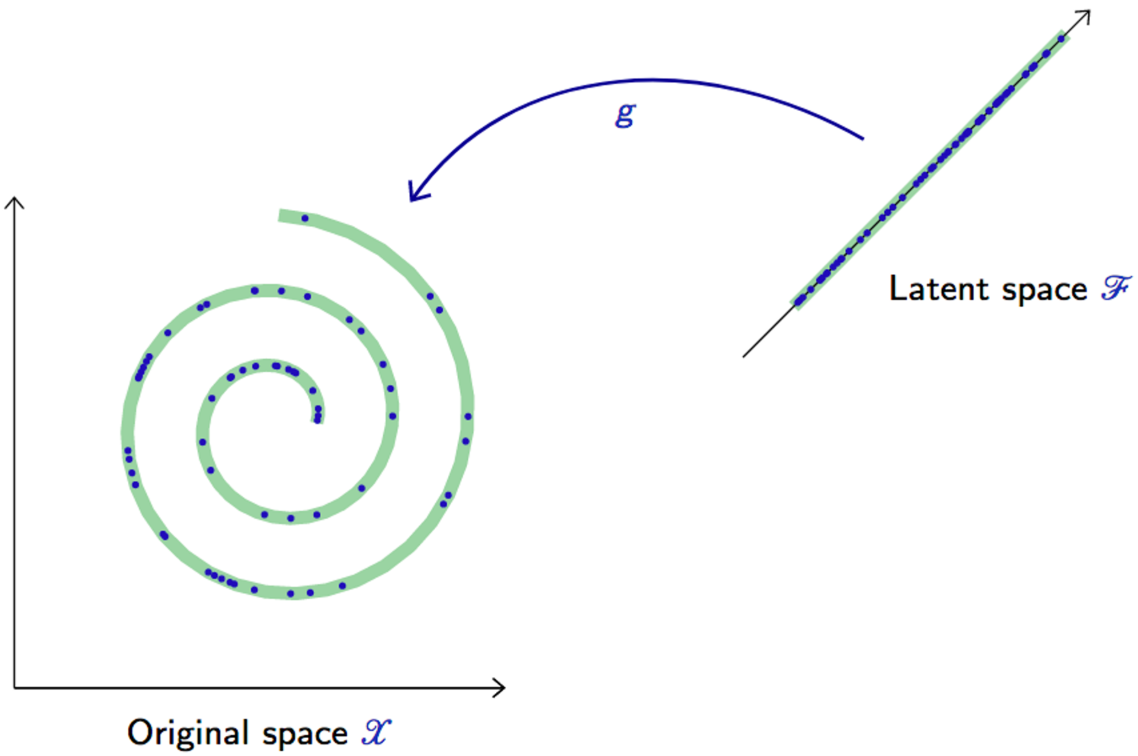
SLAC NATIONAL ACCELERATOR LABORATORY

TRISEP Summer School
July 8-12, 2024

The Plan

- Lecture 1 – Machine Learning Fundamentals
- Lecture 2 – Intro to Neural Networks
- Lecture 3 – Intro to Deep Learning
- Lecture 4 – Intro to Unsupervised Learning
- Lecture 5 – Intro to Deep Generative Models

How can we find the "meaningful degrees of freedom" in the data?

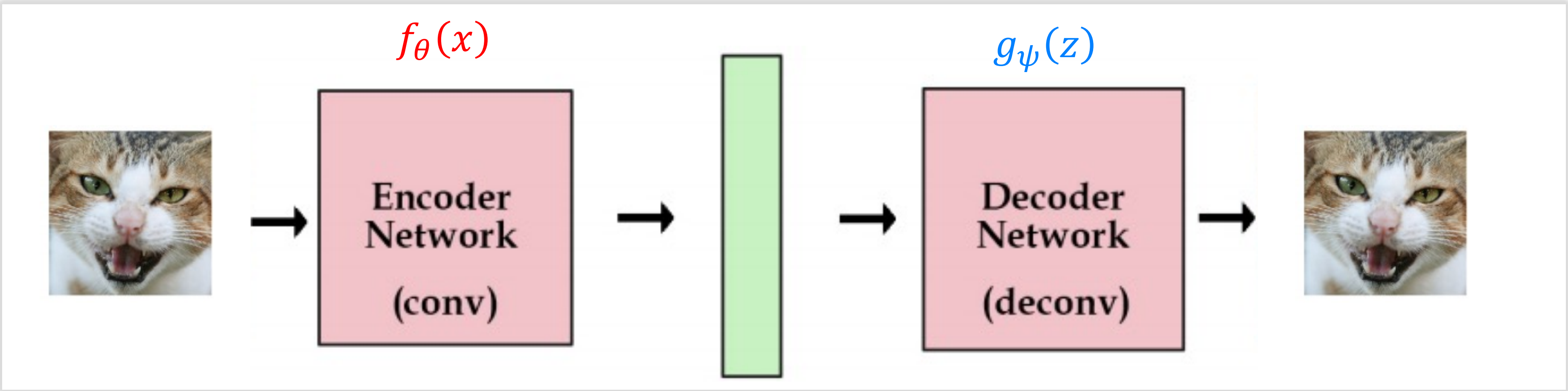


Autoencoders

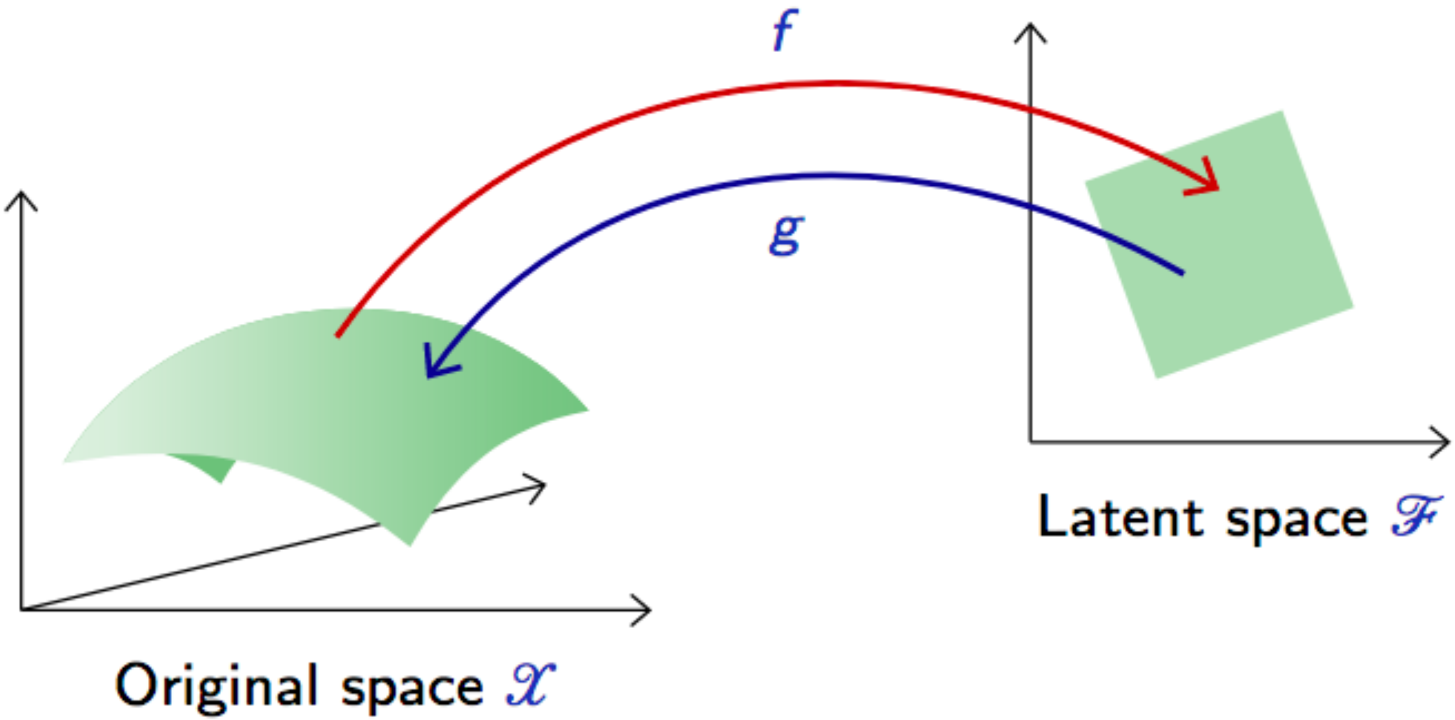
- Map a space to itself through a compression

$$x \to z \to \hat{x}$$

  - **Encoder**: Map from data to a lower dim. latent space
    - Neural network $f_\theta(x)$ with parameters θ
  - **Decoder**: Map from latent space back to data space
    - Neural network $g_\psi(z)$ with parameters ψ



Autoencoder Mappings



- Latent space is of lower dimension than data
- Model must learn a "good" parametrization and capture dependencies between components

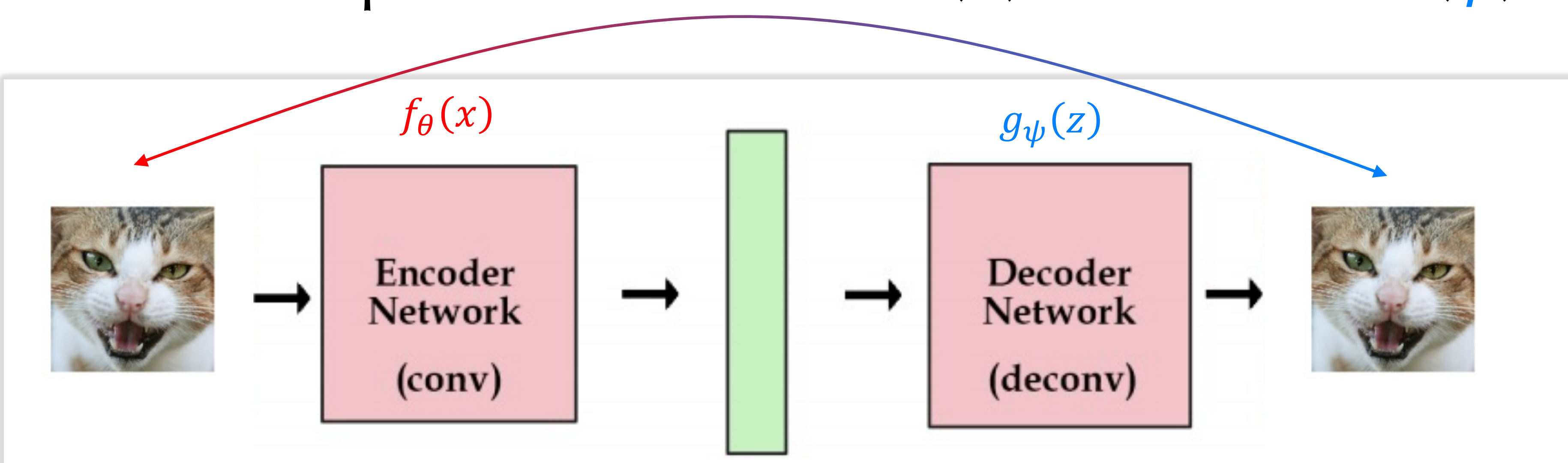
Fleuret, Deep Learning Course

Autoencoder Loss

$$L(\theta, \psi) = \frac{1}{N} \sum_n \left\| x_n - g_\psi\left(f_\theta(x_n)\right) \right\|^2$$

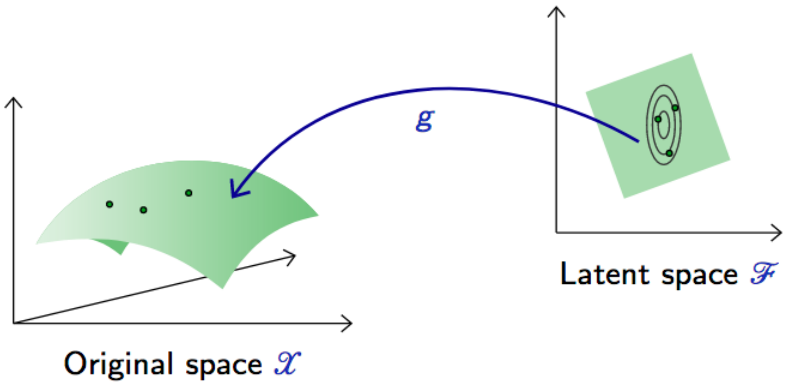
- **Loss**: mean *reconstruction loss* (MSE) between data and encoded-decoded data

- Min. over params. of encoder (θ) and decoder (ψ).



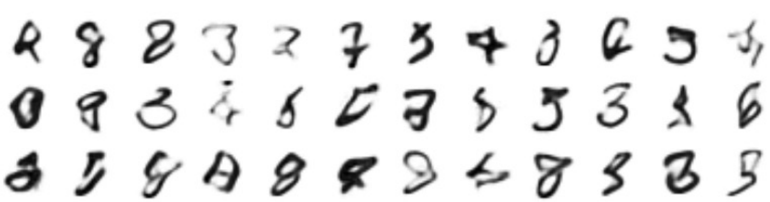
Can We Generate Data with Decoder?

- Can we sample in latent space and decode to generate data?



- What distribution to sample from in latent space?
  - Try Gaussian with mean and variance from data

Autoencoder sampling ($d = 16$)



- Don't know the right latent space density

- Autoencoders learn the latent space, but we don't know what is the latent space distribution
- Autoencoder prescribes a deterministic relationship between data space and latent space
- One set of "meaningful degrees of freedom" can only describe one data space point

Generative Models

Generative Models Goal

A generative model is a probabilistic model q that can be used as a simulator of the data.

Goal: generate synthetic, realistic high-dimension data

$$x \sim q(x; \theta)$$

that is as close as possible to the unknown data distribution $p(x)$ for which we have empirical samples.

i.e. want to recreate the raw data distribution (such as the distribution of natural images).

Types of Generative Models

- Generative models aim to:
  - Learn a distribution $p(x)$ that explains the data
  - Draw samples of plausible data points

- Explicit Models
  - Can evaluate the density $p(x)$ of a data point x

- Implicit Models
  - Can only sample $p(x)$, but not evaluate density

Variational Autoencoders

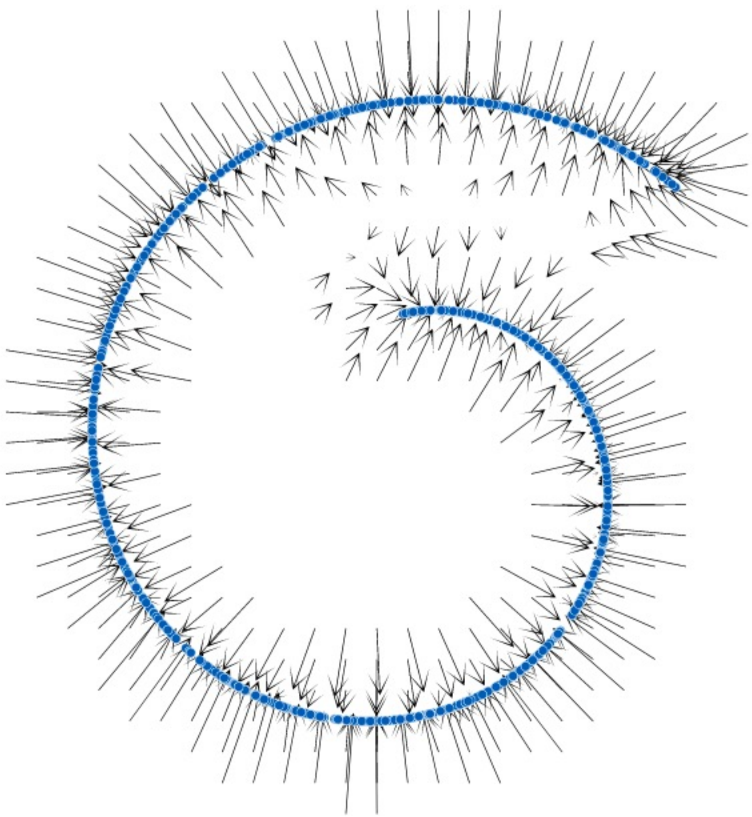
Denoising Autoencoders

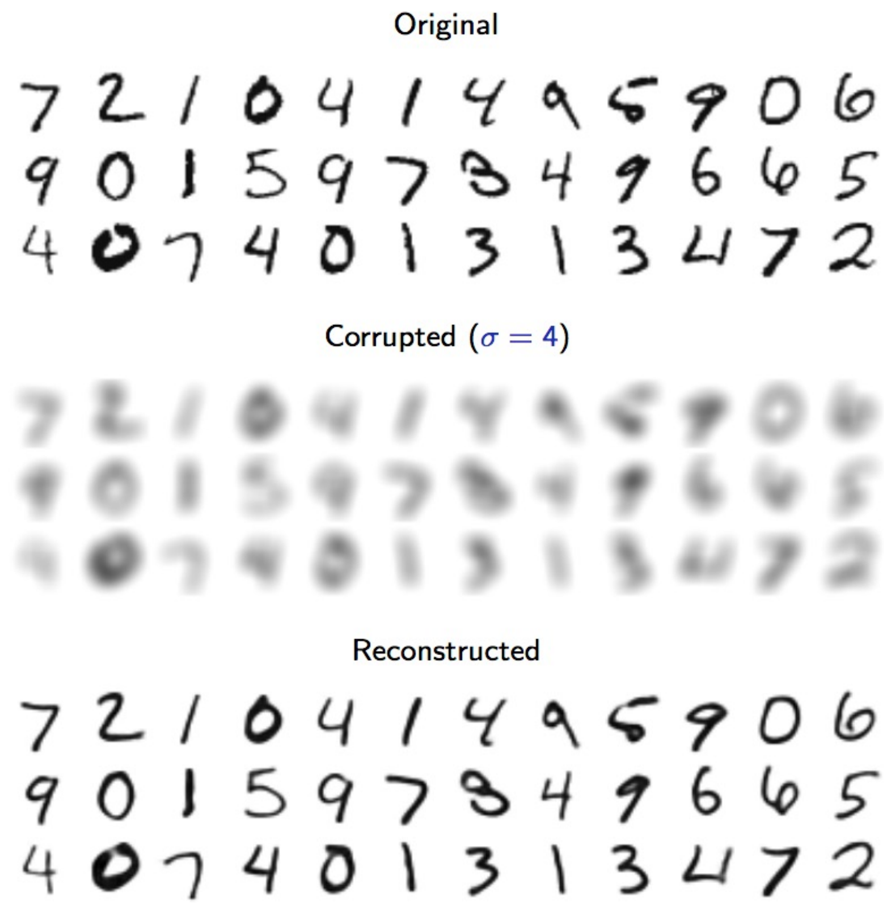
- Learn a mapping from corrupted data space $\widetilde{\mathcal{X}}$ back to original data space
  - Mapping $\phi_w(\widetilde{\mathcal{X}}) = \mathcal{X}$
  - ϕ_w will be a neural network with parameters w
- Loss:

$$\mathrm{L} = \frac{1}{N} \sum_n \|x_n - \phi_w(x_n + \epsilon_n)\|$$

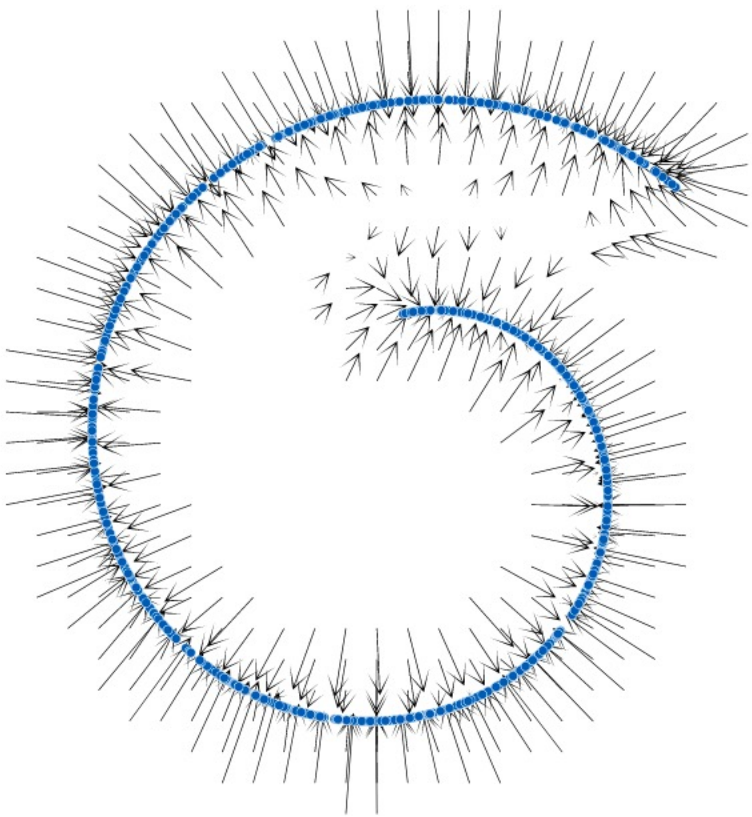
Perturbation, e.g. Gaussian noise

Denoising Autoencoders Examples





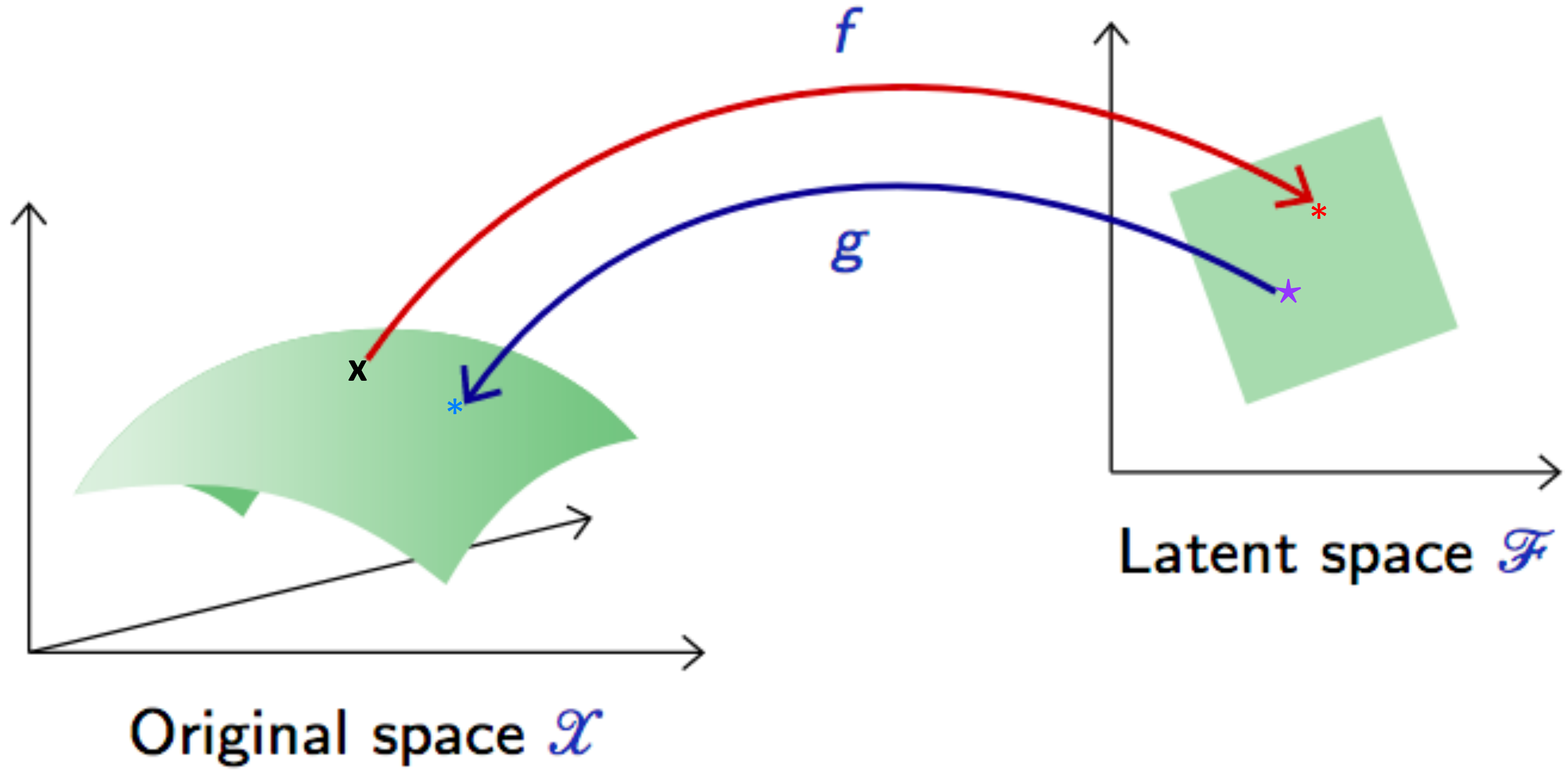
Denoising Autoencoders Examples



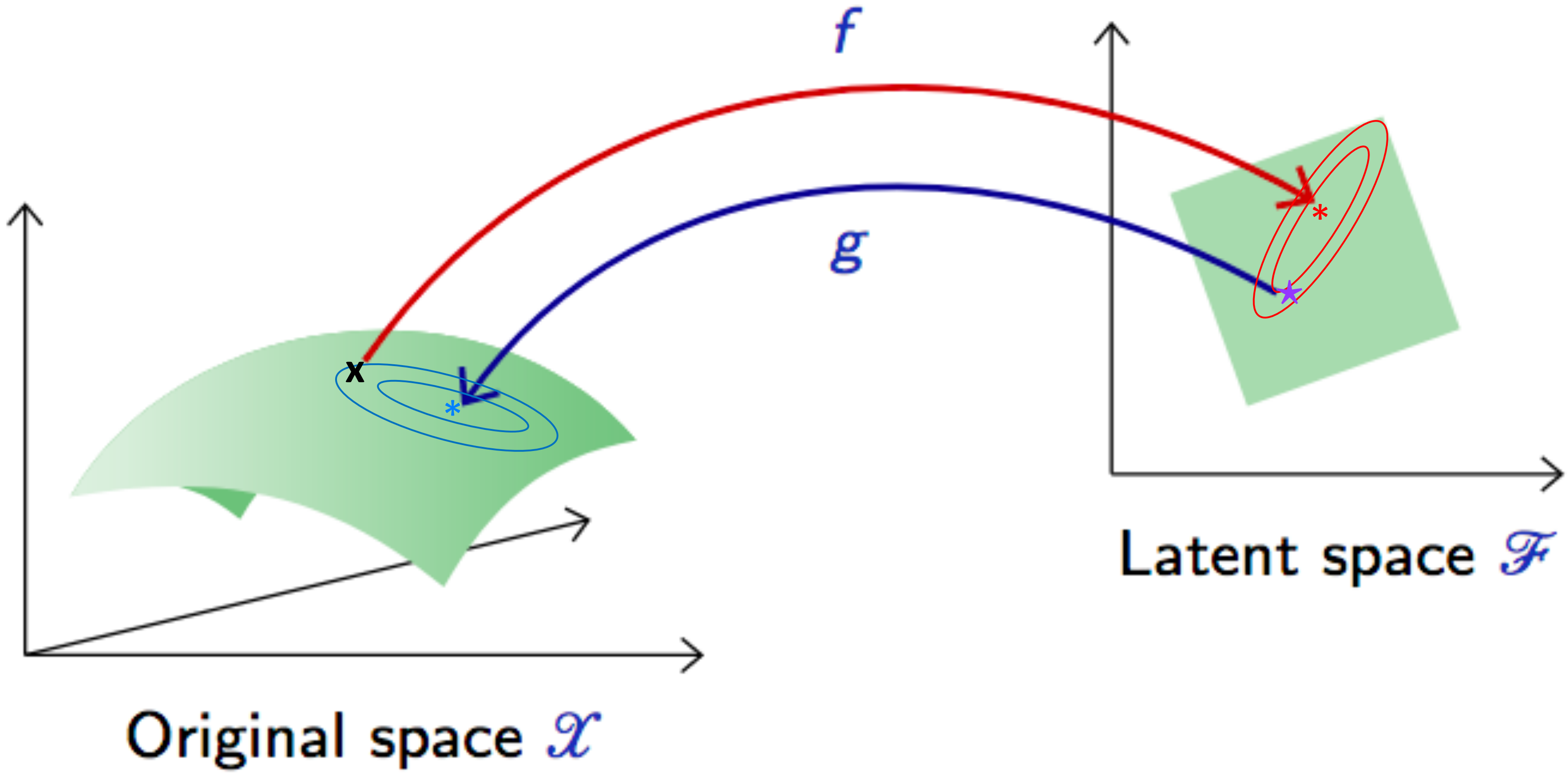
- **Autoencoder learns the average behavior**
- What if we care about these variations?
- Can we add a notion of variation in the autoencoder?

Fleuret, [Deep Learning Course]

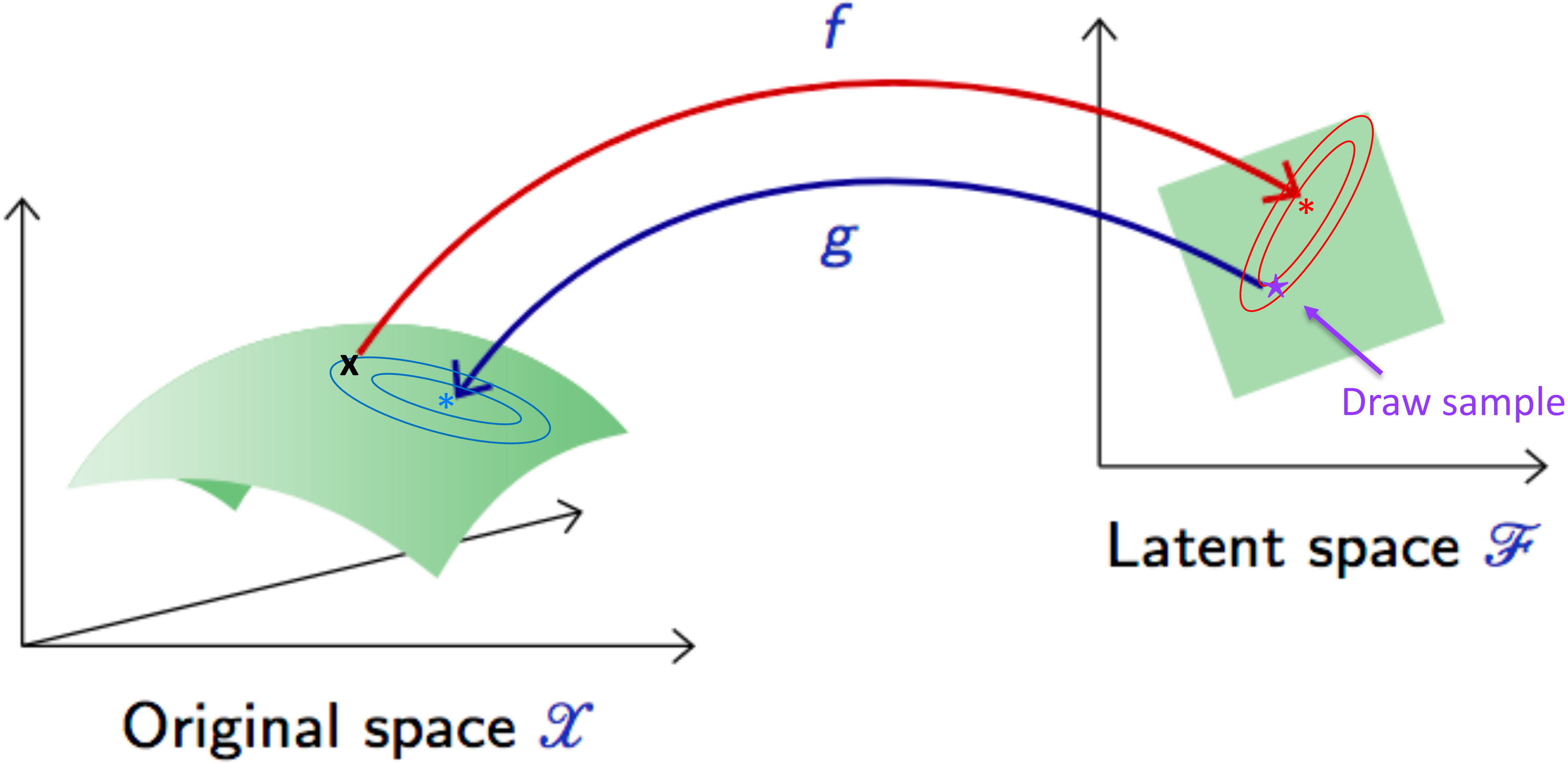
Autoencoder



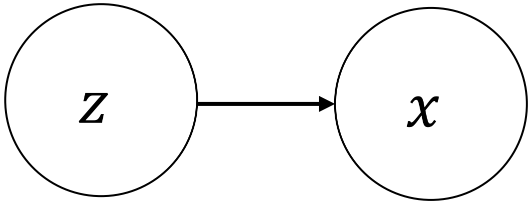
Variational Autoencoder



Variational Autoencoder



Latent Variable Models



- Observed random variable x depends on unobserved latent random variable z
- Joint probability: $p(x, z) = p(x|z)p(z)$
- $p(x|z)$ is stochastic generation process from $z \rightarrow x$

From Deterministic to Probabilistic Autoencoder

- Probabilistic relationship between data and latents

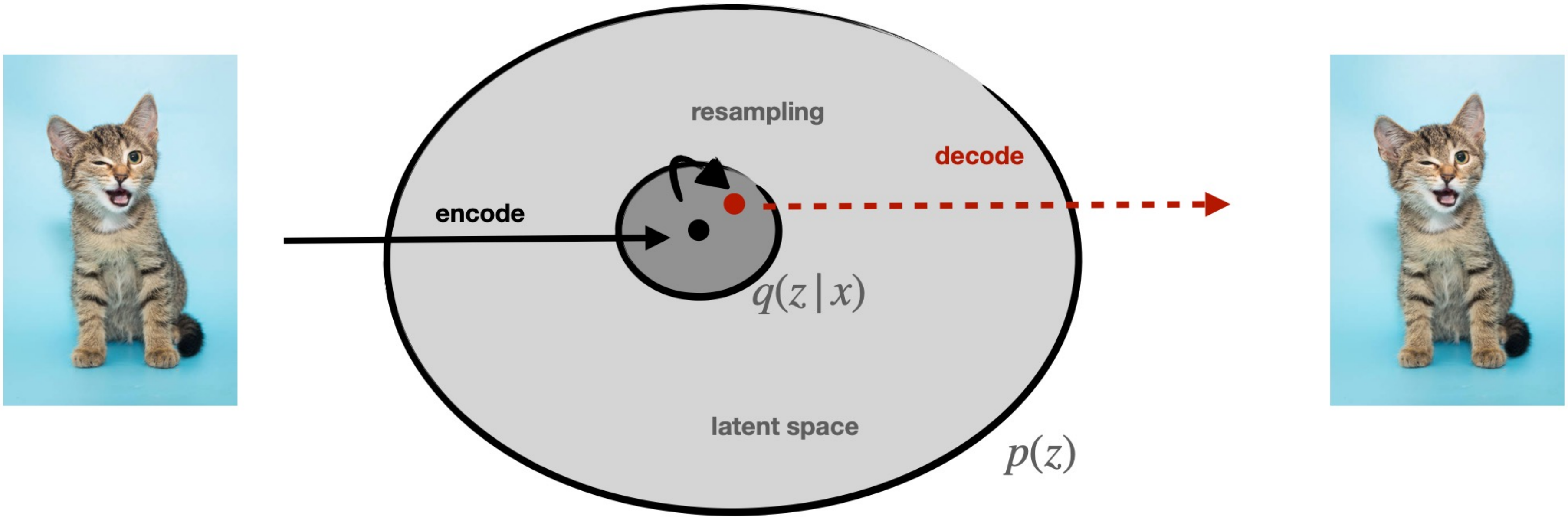
$$x, z \sim p(x, z) = p(x|z)p(z)$$

- Autoencoding

$$x \to q(z|x) \underset{sample}{\Longrightarrow} z \to p(x|z)$$

  - **Encoder:** Learn what latents can produced data: $q(z|x)$
  - **Decoder:** Learn what data is produced by latent: $p(x|z)$

Variational Autoencoder



- Close-by points must decode to similar images

Image credit: L. Heinrich

How do we design Encoder and Decoder

- Classification / regression models make single prediction

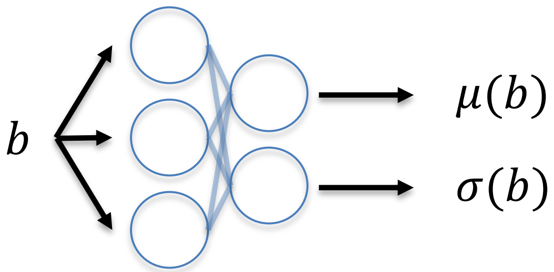
How to model a conditional density $p(a|b)$?

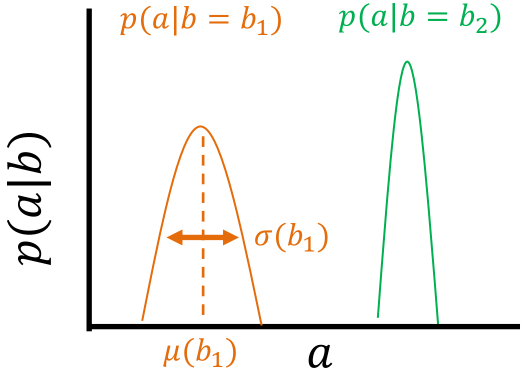
- Assume a known form of density, e.g. normal

$$p(a|b) = \mathcal{N}\big(a; \mu(b), \sigma(b)\big)$$

  – Parameters of density depend on conditioned variable

- Use neural network to model density parameters

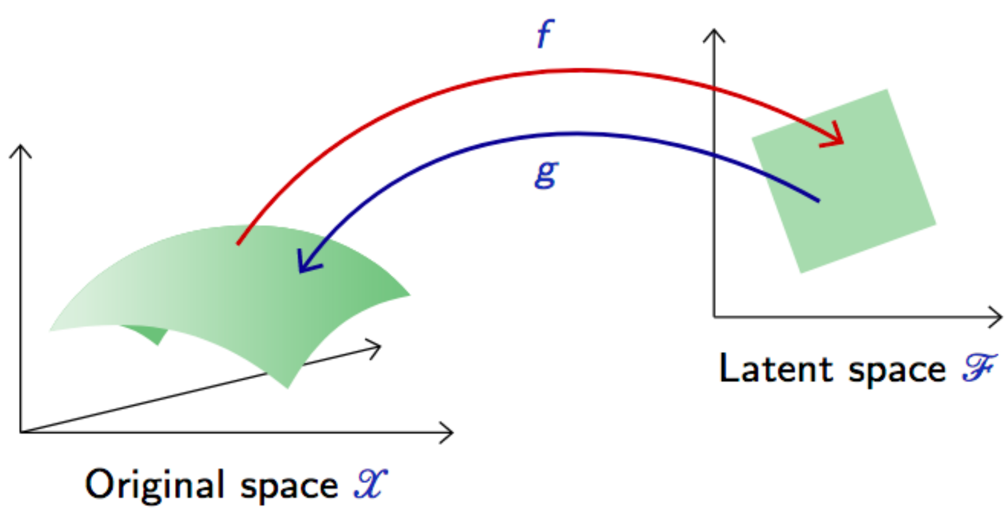




Encoding

- Typical encoder maps input x to "average" point in latent space

$$f(x) = \mu(x)$$

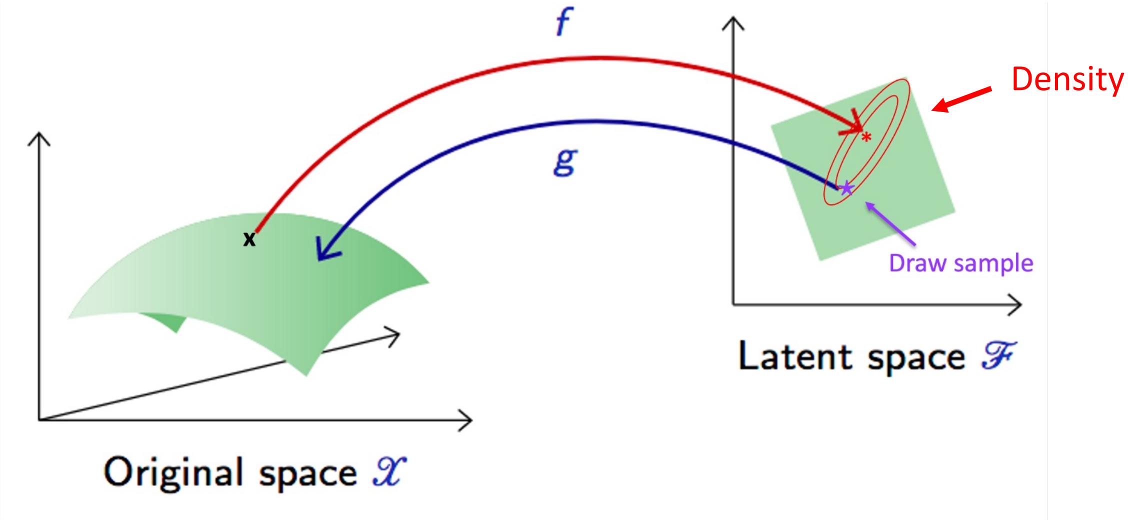


Encoding

- A VAE Encoder has two outputs: mean & variance function

$$f_\psi(x) = \{\mu_\psi(x), \sigma_\psi^2(x)\}$$

ψ are parameters of the NN



Encoding

- A VAE Encoder has two outputs: mean & variance function

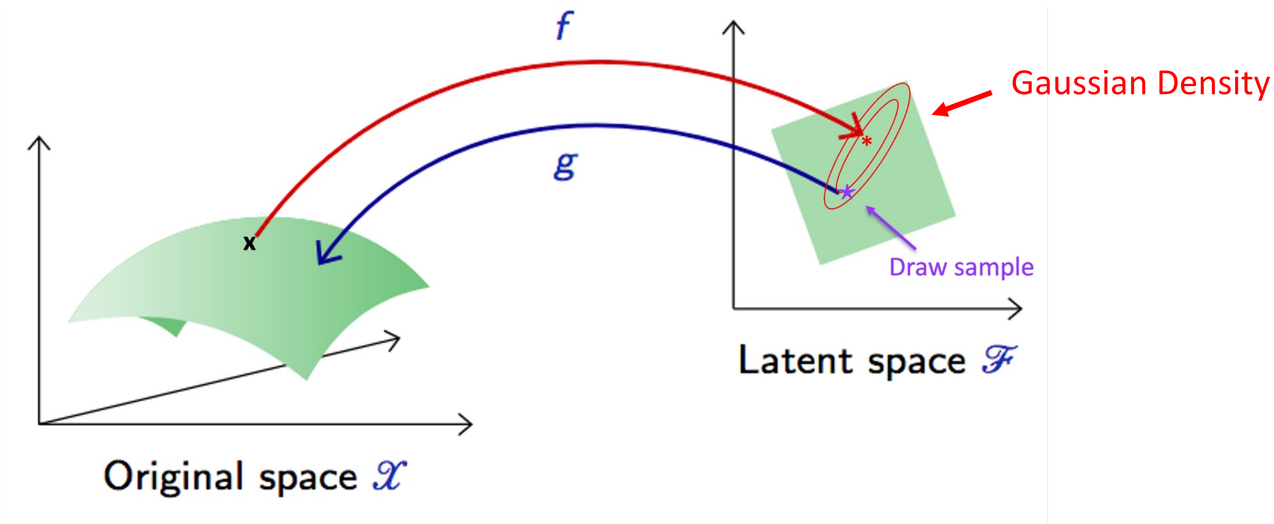
$$f_\psi(x) = \{\mu_\psi(x), \sigma_\psi^2(x)\}$$

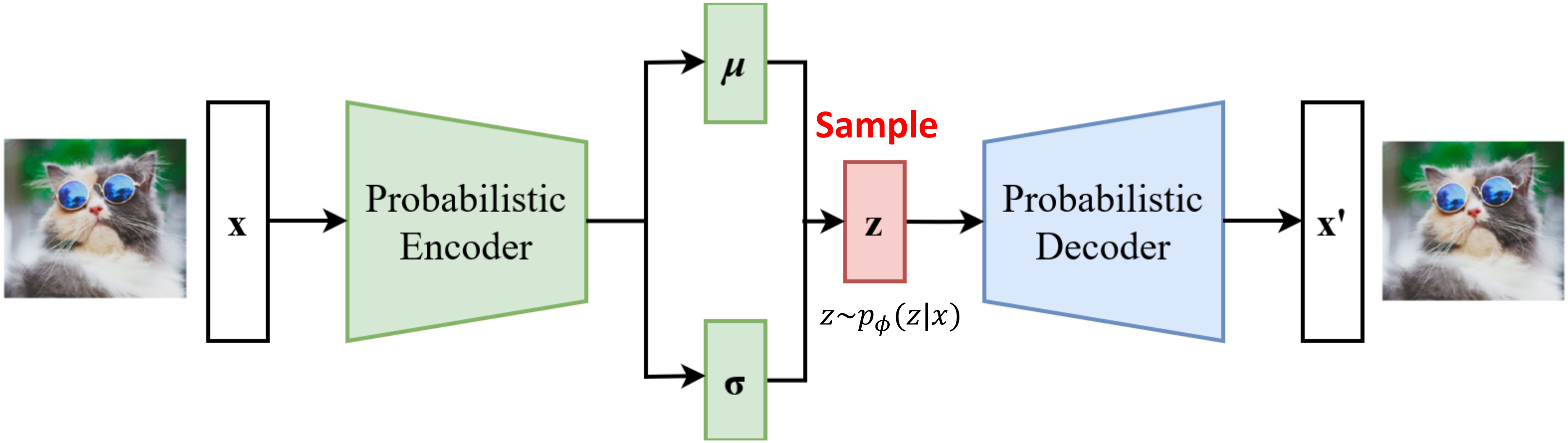
ψ are parameters of the NN

- What is the probability of a point in latent space?

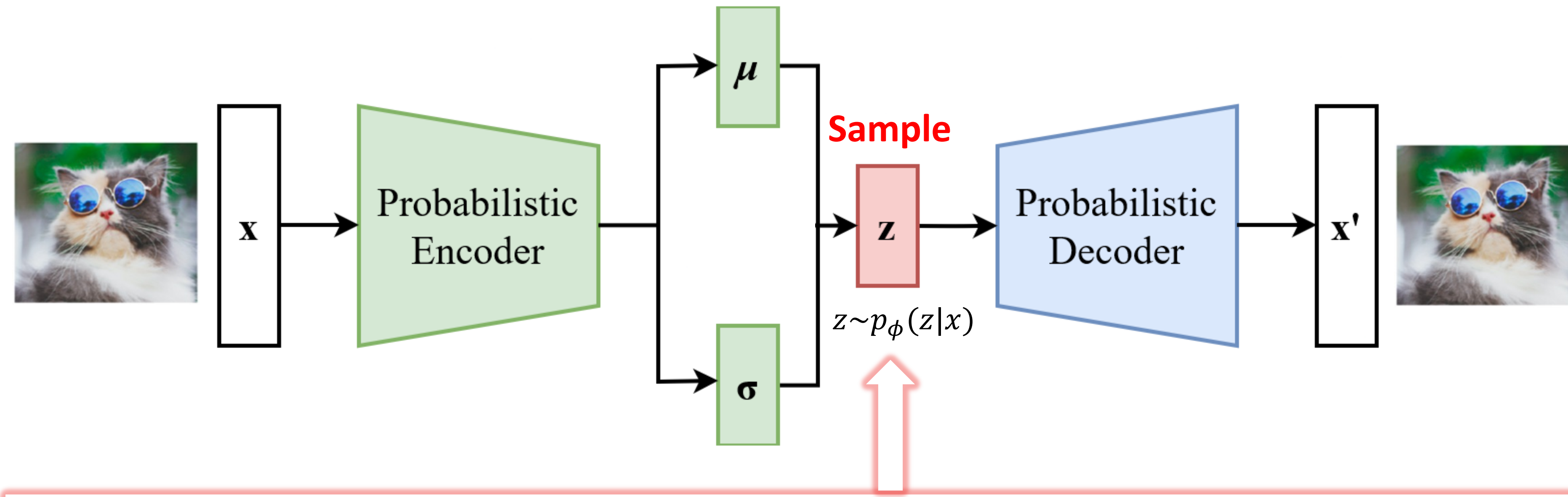
$$p_\psi(z|x) = N(z \mid \mu_\psi(x), \sigma_\psi^2(x))$$

Could choose different density
Gaussian is easiest





Kingma, Welling, [1312.6114](1312.6114)
Rezende, Mohamed, Wierstra, [1401.4082](1401.4082)



But for training:

How do we take a derivative through a randomly sampled number?

How do we know the dependence on the parameters?

Kingma, Welling, 1312.6114
Rezende, Mohamed, Wierstra, 1401.4082

Re-parameterization trick

- Given $x \sim p(x|\theta)$

- Sometimes, we can rewrite x as a function of the parameters and a simpler distribution without parameter dependence

$$x = g(\epsilon, \theta) \qquad \epsilon \sim p(\epsilon)$$

- Example:

$$x \sim N(x|\mu, \sigma) \;\rightarrow\; x = \sigma * \epsilon + \mu \;\text{ with }\; \epsilon \sim N(0,1)$$

Encoding

- A VAE Encoder has two outputs: mean & variance function

$$f_\psi(x) = \{\mu_\psi(x), \sigma_\psi^2(x)\}$$

ψ are parameters of the NN

- What is the probability of a point in latent space?

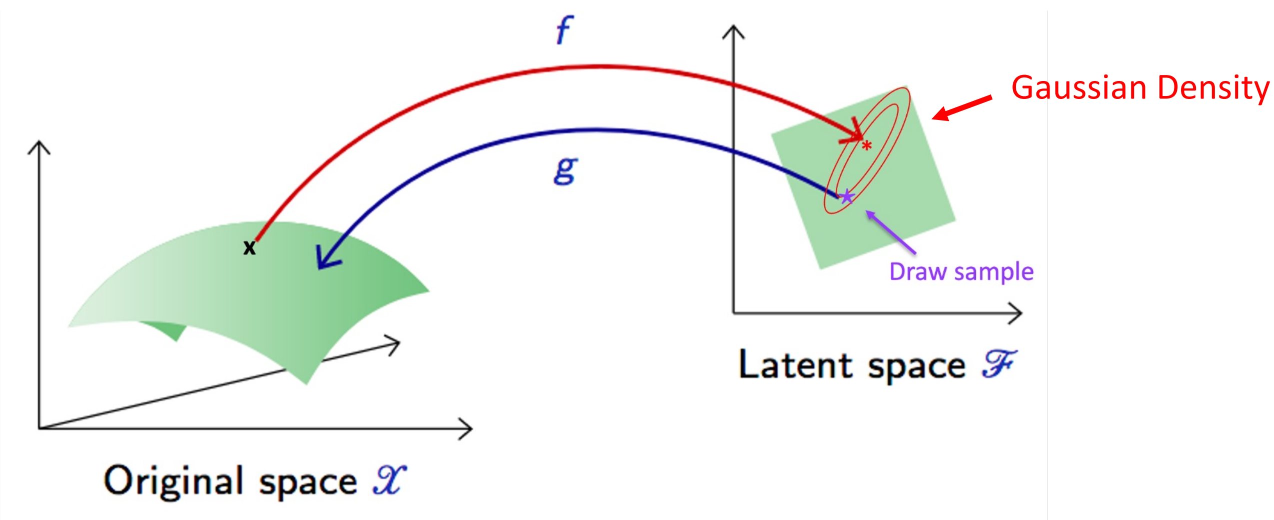
$$p_\psi(z|x) = N(z \mid \mu_\psi(x), \sigma_\psi^2(x))$$

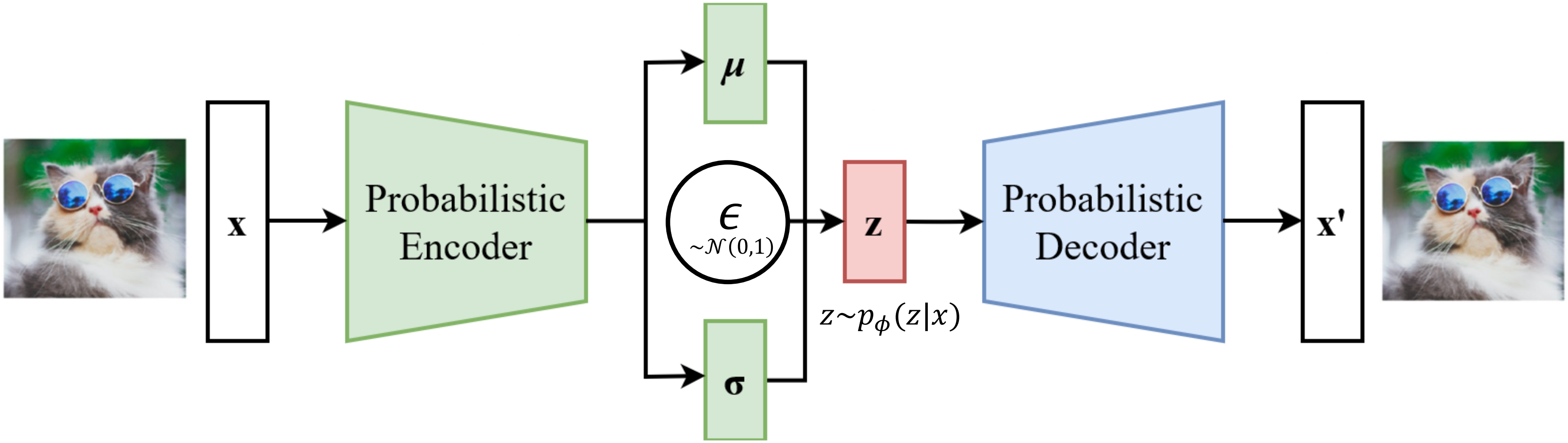
Could choose different density
Gaussian is easiest

- How do we draw a sample in latent space?

$$z = \sigma_\psi(x) * \epsilon + \mu_\psi(x) \qquad \epsilon \sim N(0, I)$$

Re-parameterization trick





Kingma, Welling, 1312.6114
Rezende, Mohamed, Wierstra, 1401.4082

Decoding

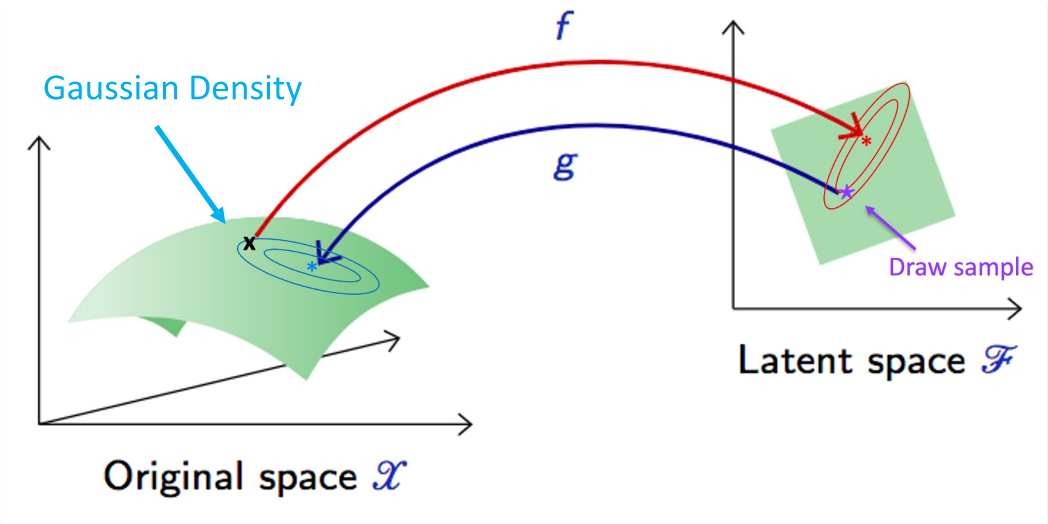
- Same as autoencoder

$$g_\theta(z) \equiv \mu_\theta(z)$$

θ are parameters of the NN

- Likelihood of an observation x

$$p_\theta(x|z) = N(x \mid \mu_\theta(z), I)$$



Decoding

- Same as autoencoder

$$g_\theta(z) \equiv \mu_\theta(z)$$

θ are parameters of the NN

- Likelihood of an observation x

$$p_\theta(x|z) = N(x \mid \mu_\theta(z), I)$$

- "**Reconstruction Loss**": Maximum likelihood

$$L_{reco} = \mathbb{E}_{z \sim q(z|x)}[\log p(x|z)]$$

Decoding

- Same as autoencoder

$$g_\theta(z) \equiv \mu_\theta(z)$$

θ are parameters of the NN

- Likelihood of an observation x

$$p_\theta(x|z) = N(x \mid \mu_\theta(z), I)$$

- "**Reconstruction Loss**": Maximum likelihood

$$L_{reco} = \mathbb{E}_{z\sim q(z|x)}[\log p(x|z)] \approx \frac{1}{N} \sum_{z_i \sim q(z|x)} \log N(x \mid g_\theta(z_i), I)$$

Decoding

- Same as autoencoder

$$g_\theta(z) \equiv \mu_\theta(z)$$

θ are parameters of the NN

- Likelihood of an observation x

$$p_\theta(x|z) = N(x \mid \mu_\theta(z), I)$$

- "**Reconstruction Loss**": Maximum likelihood

$$L_{reco} = \mathbb{E}_{z\sim q(z|x)}[\log p(x|z)] \approx -\frac{1}{N} \sum_{z_i \sim q(z|x)} \left(x - g_\theta(z_i)\right)^2$$

Same as the autoencoder loss

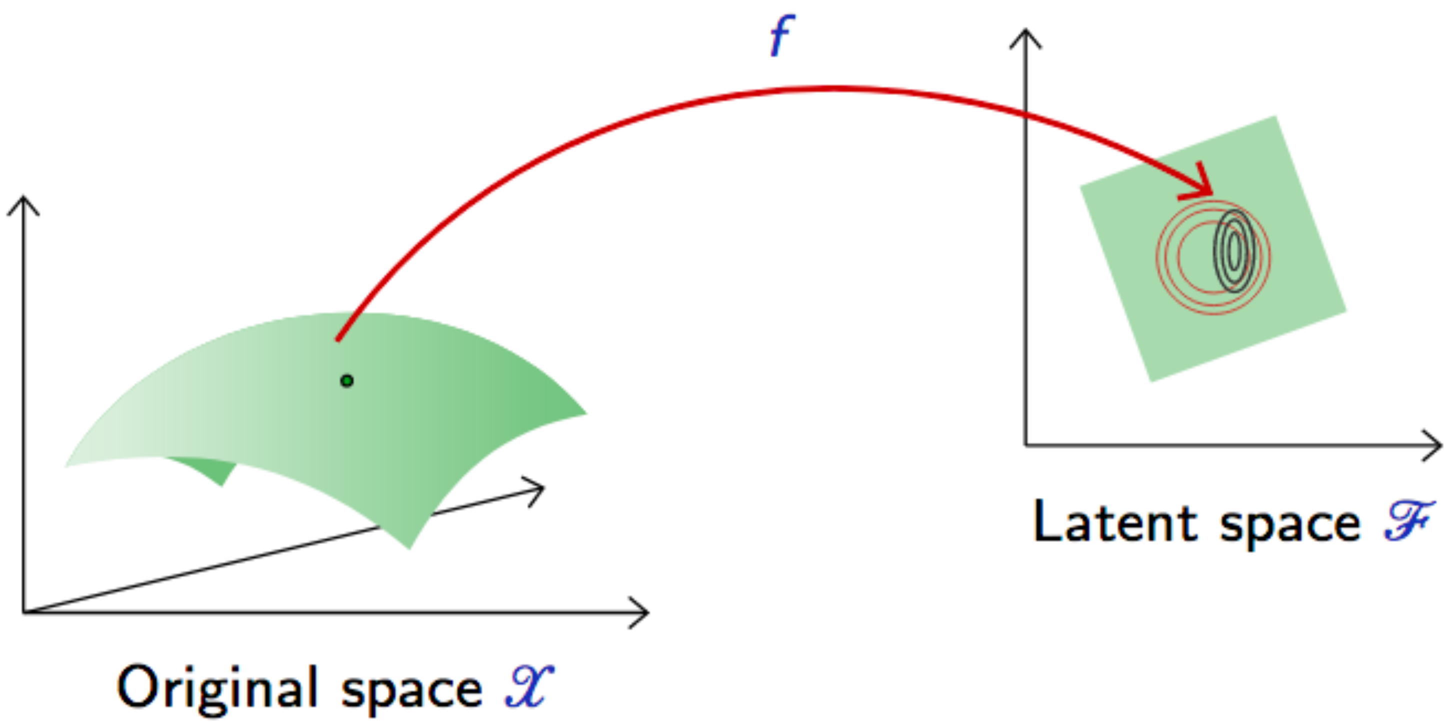
Variational Autoencoder Training Loss

- How do we make sure system doesn't collapse to an autoencoder (i.e. VAE encoder only predicts mean)?

Variational Autoencoder Training Loss

- How do we make sure system doesn't collapse to an autoencoder (i.e. VAE encoder only predicts mean)?

- Use prior $p(z)$ for the latent space distribution, **need to ensure the encoder is consistent with prior**



Variational Autoencoder Training Loss

- Constrain difference between distributions with **Kullback–Leibler divergence**

$$D_{KL}[q(z|x)|p(z)] = \mathbb{E}_{q(z|x)}\left[\log \frac{q(z|x)}{p(z)}\right] = \int q(z|x) \log \frac{q(z|x)}{p(z)}\, dz$$

  - $D_{KL}[q|p] \geq 0$ and is only 0 when $q = p$

Variational Autoencoder Training Loss

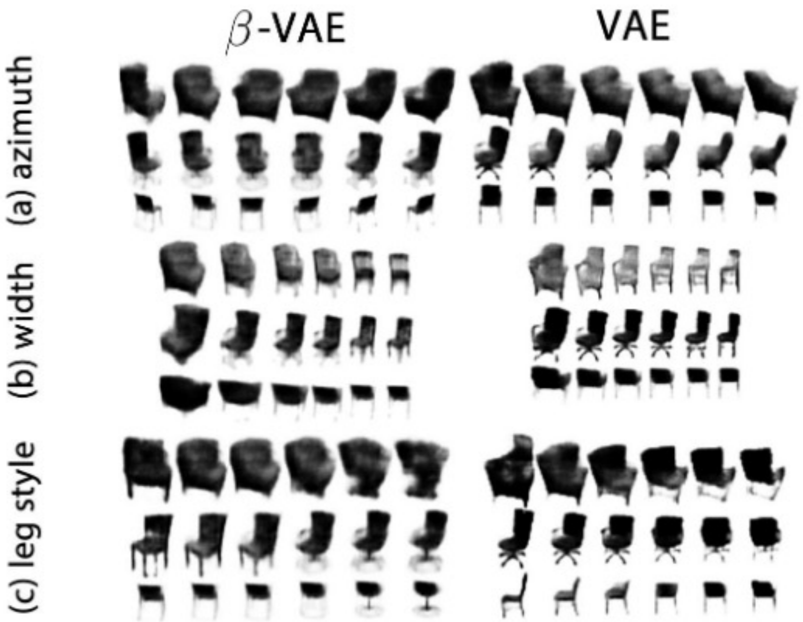
- Constrain difference between distributions with **Kullback–Leibler divergence**

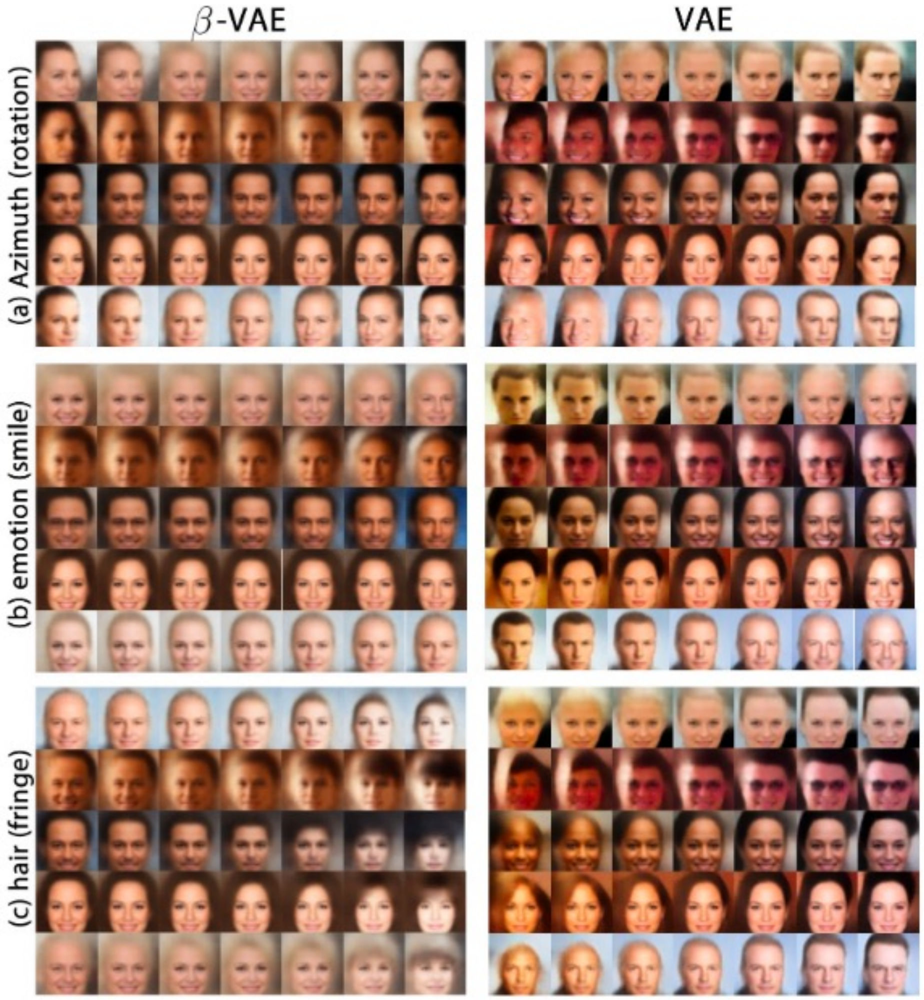
$$D_{KL}[q(z|x)|p(z)] = \mathbb{E}_{q(z|x)}\left[\log \frac{q(z|x)}{p(z)}\right] = \int q(z|x) \log \frac{q(z|x)}{p(z)}\, dz$$

- VAE full objective

$$\max_{\theta,\psi} L(\theta,\psi) = \max_{\theta,\psi}\Big[\underbrace{\mathbb{E}_{q_\psi(z|x)}[\log p_\theta(x|z)]}_{\text{Reconstruction Loss}} - \underbrace{D_{KL}[q_\psi(z|x)|p(z)]}_{\text{Regularization of Encoder}}\Big]$$

Examples

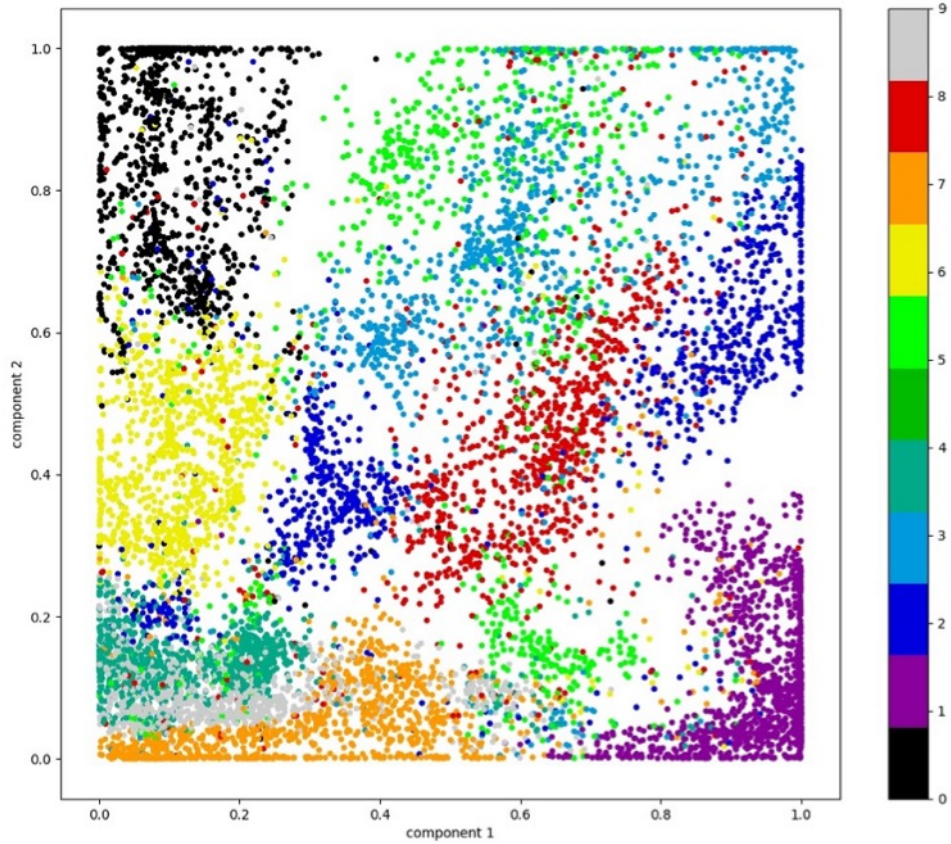




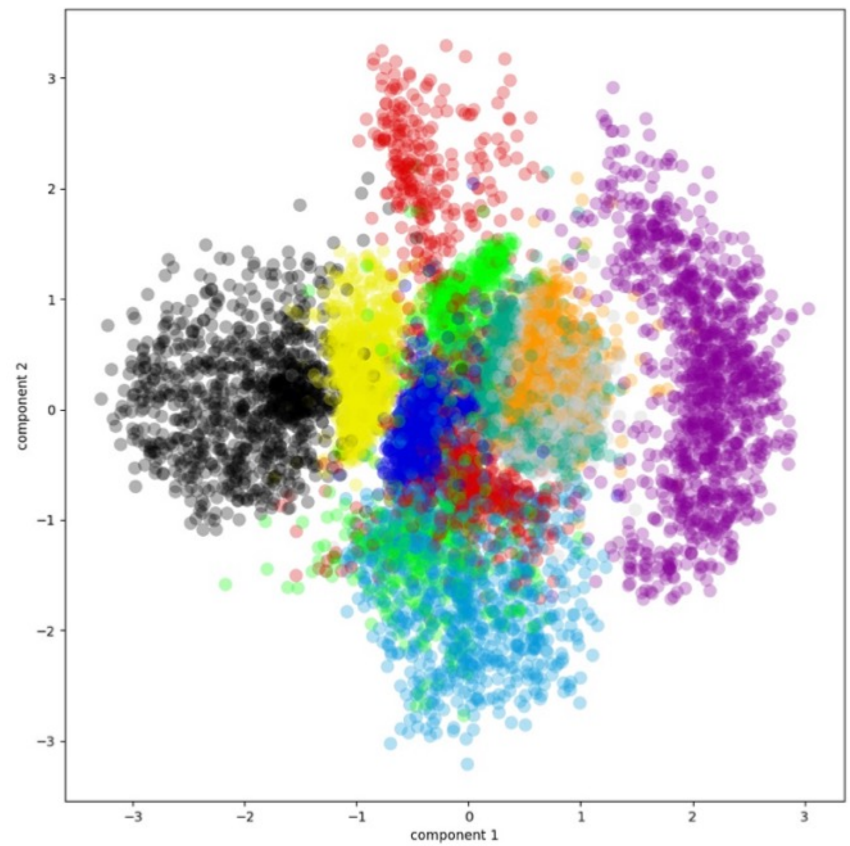
Higgins et al., [2017]

Comparing Latent Spaces

Autoencoder

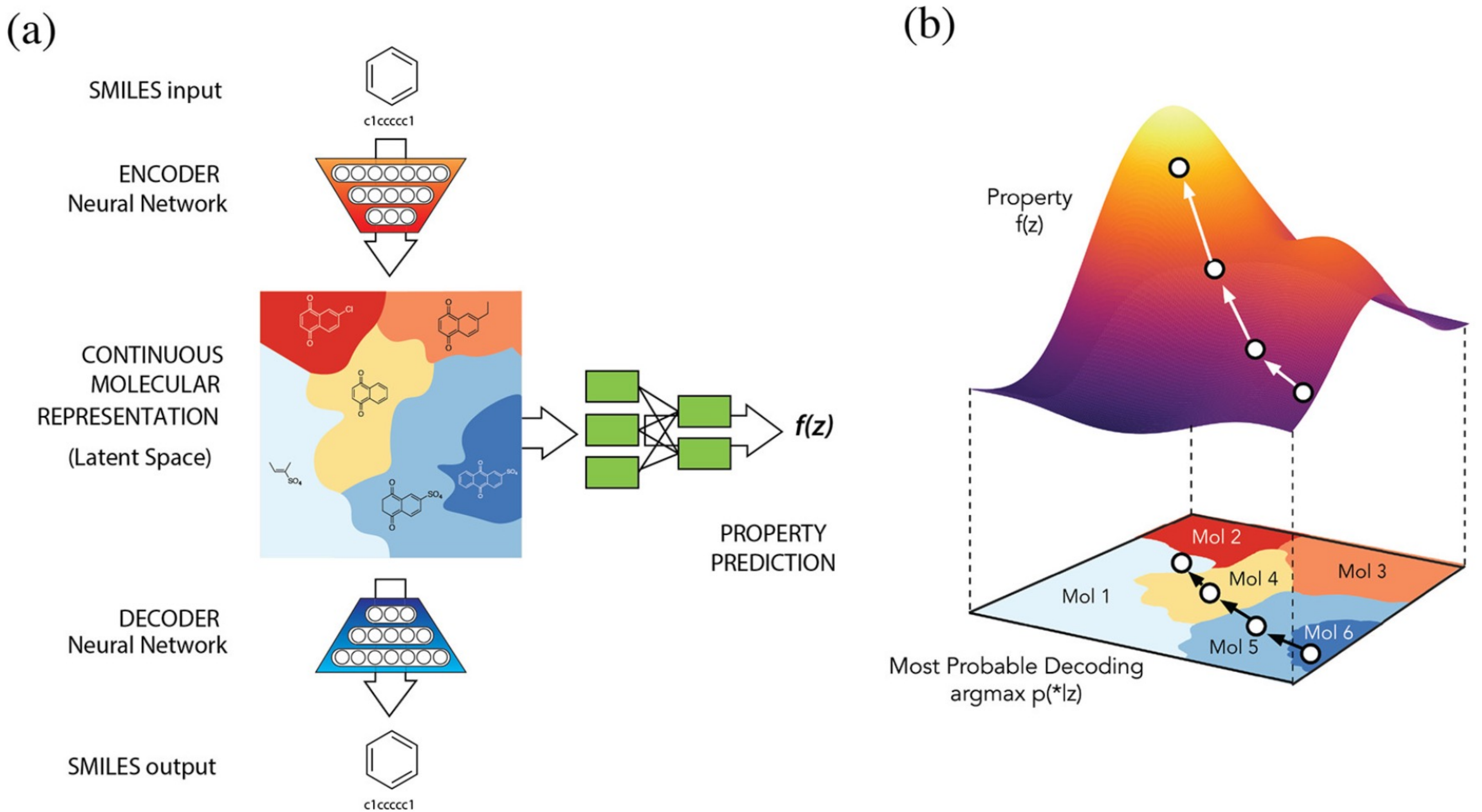


Variational Autoencoder



Data: MNIST data set of hand-written digits

Examples



Design of new molecules with desired chemical properties.
(Gomez-Bombarelli et al, 2016)

Slide credit: [G. Louppe]

What have we learned?

- In generative modeling, want to learn the lower dimensional degrees of freedom that describe the features of the data
- “Degrees of freedom” are modeled with a latent distribution (kept simple for convenience) and complex neural network mappings
- Need to think about **probabilistic systems**
- Design loss around this probabilistic model

The Zoo of Generative Models…

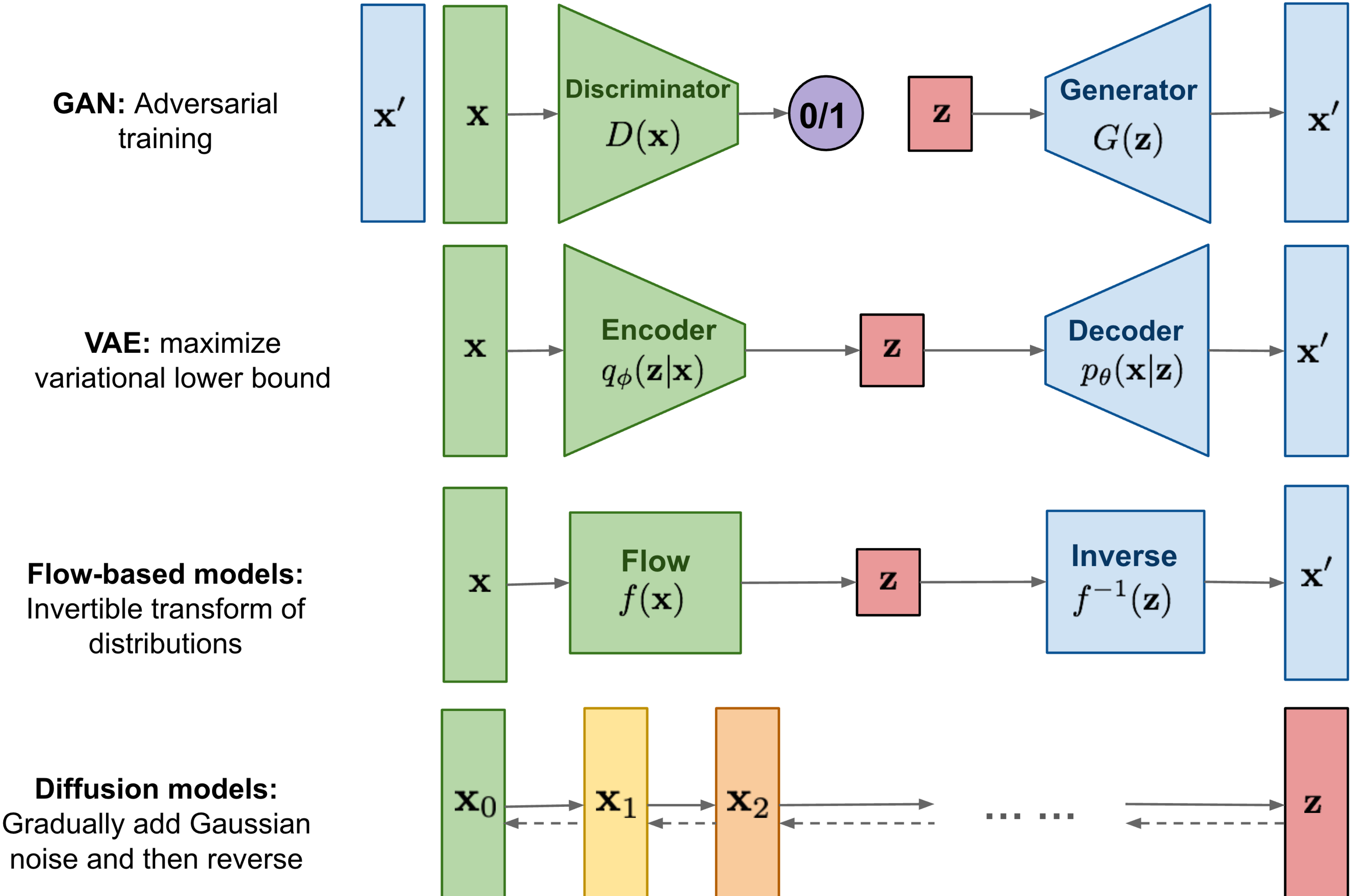
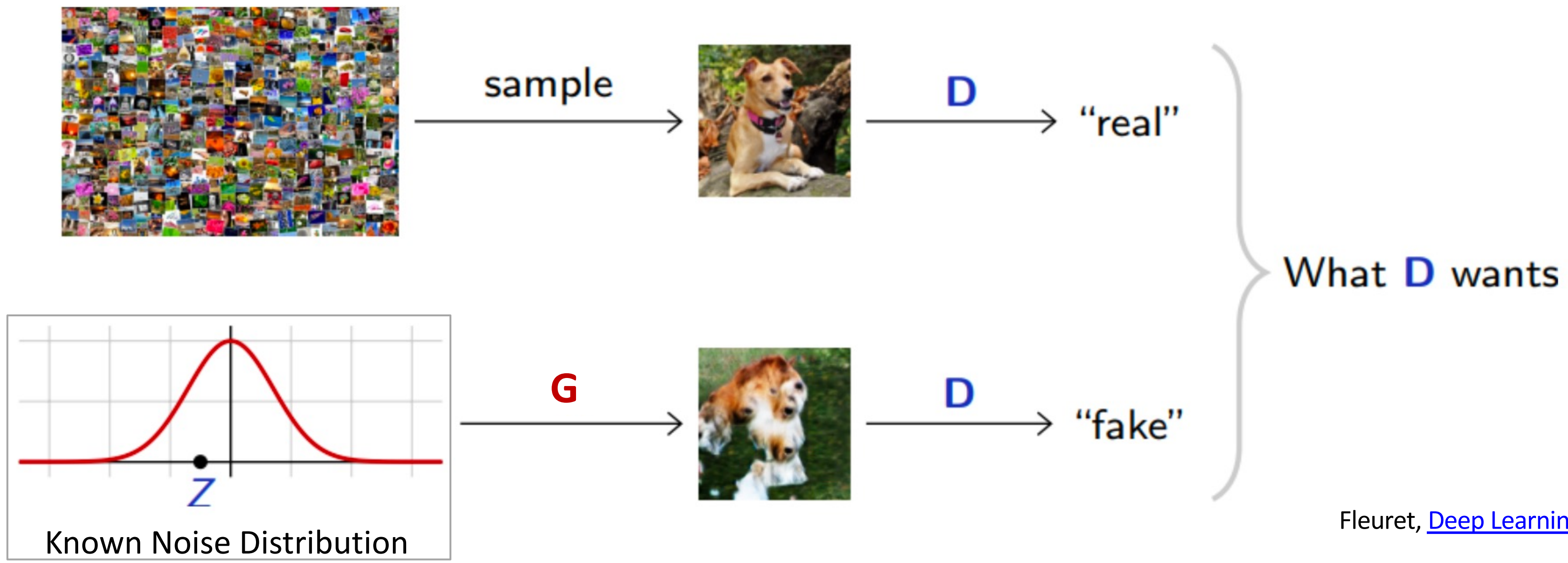


Image credit: [Lilian Weng]()

Generative Adversarial Networks (GAN)

Goodfellow et. al., [2014]



Fleuret, Deep Learning Course

- Generator creates data from noise, trained to trick Discriminator that classifies data as real or fake

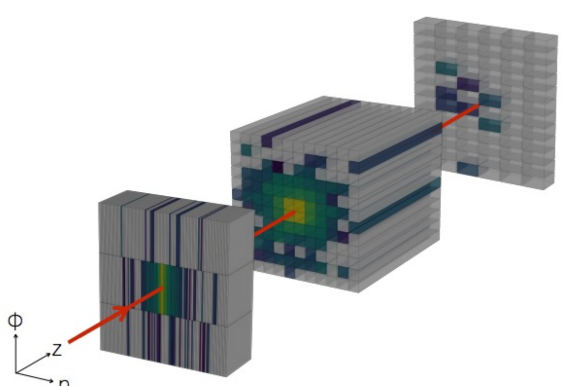
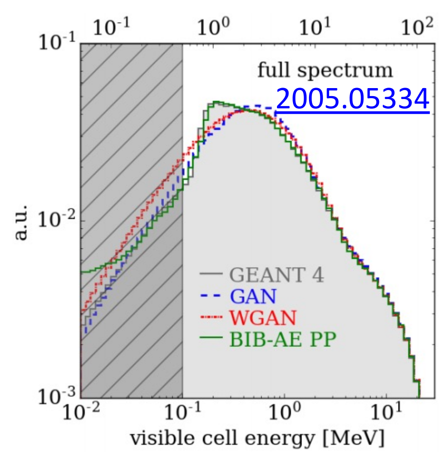
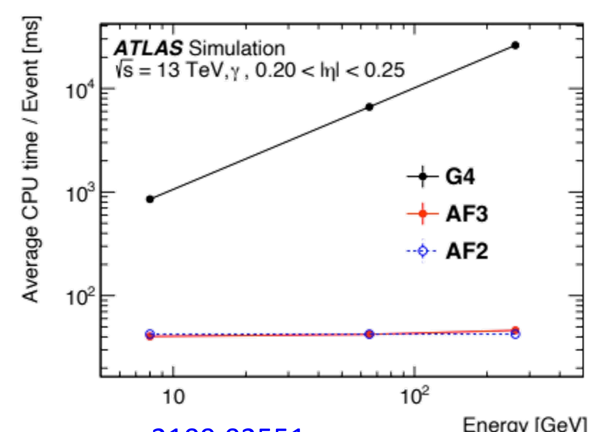


Image credit: 1705.02355

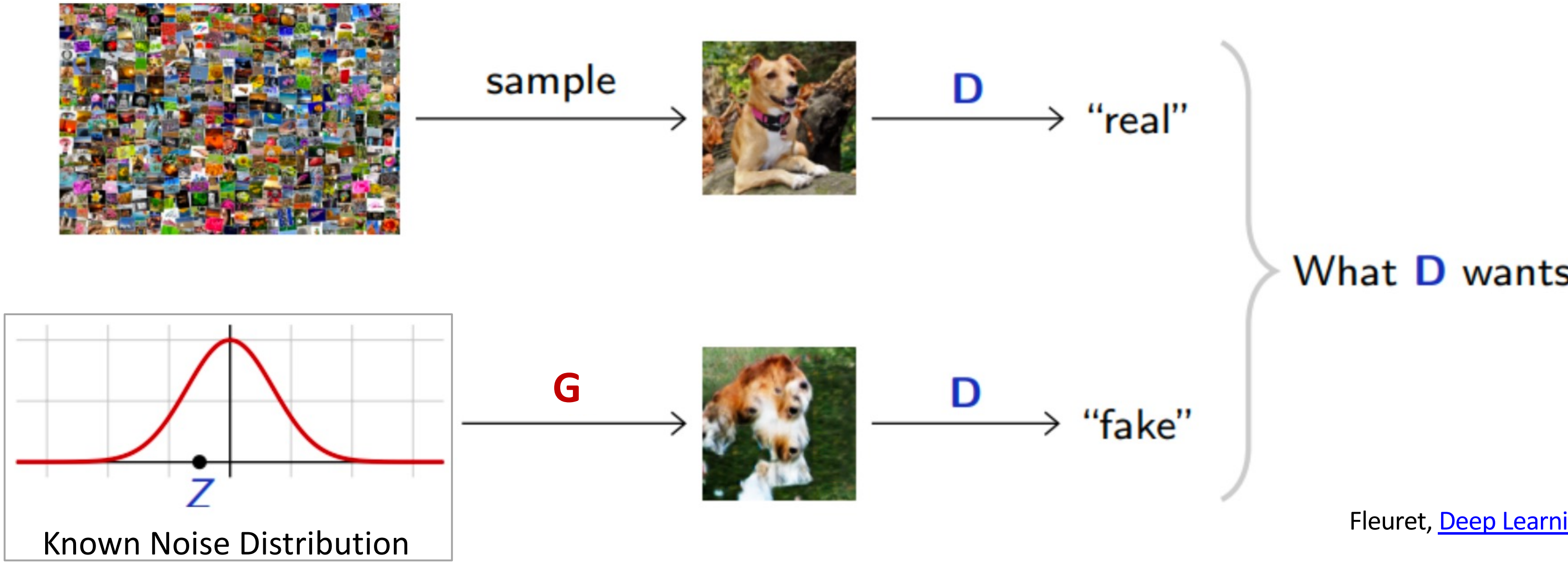




2109.02551

Generative Adversarial Networks (GAN)

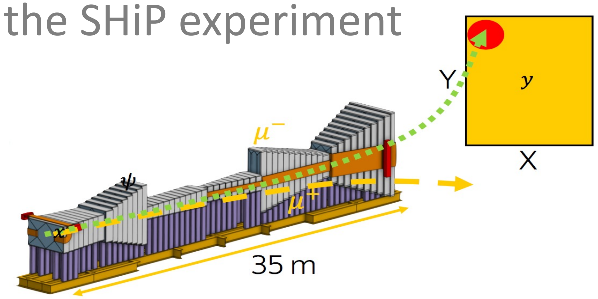
Goodfellow et. al., [2014]

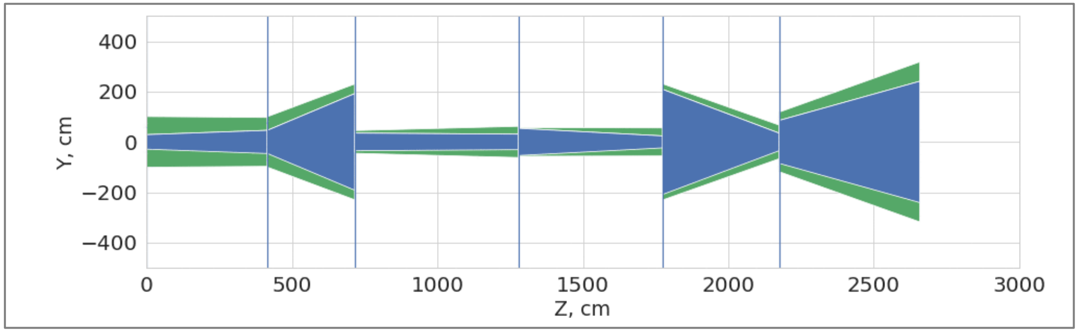


Fleuret, [Deep Learning Course]

- Generator creates data from noise, trained to trick Discriminator that classifies data as real or fake

Optimization of the magnet system
For the SHiP experiment





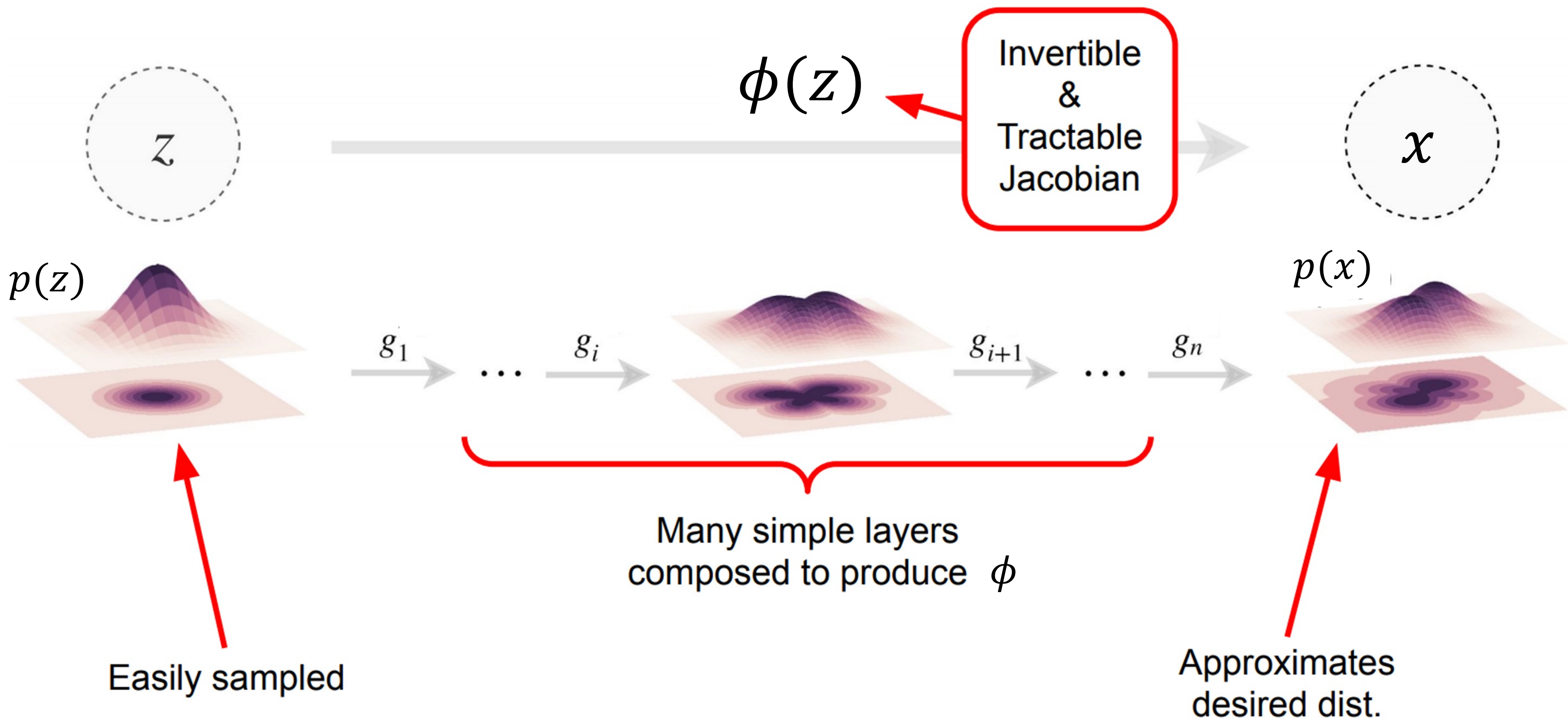
Shirbokov, **MK**, et al., [NeurIPS **33**, 14650-14662 (2020)]

Normalizing Flows

Explicit density estimation

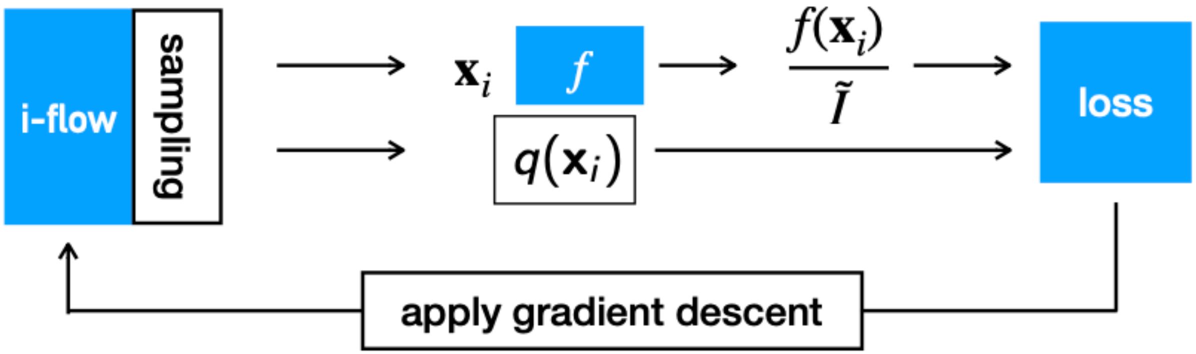
We can evaluate density $p(x)$

$$p_x(\boldsymbol{x}) = p_z(\boldsymbol{z}) \left| \det \left(\frac{\partial \phi(\boldsymbol{z})}{d\boldsymbol{z}} \right)^{-1} \right|$$



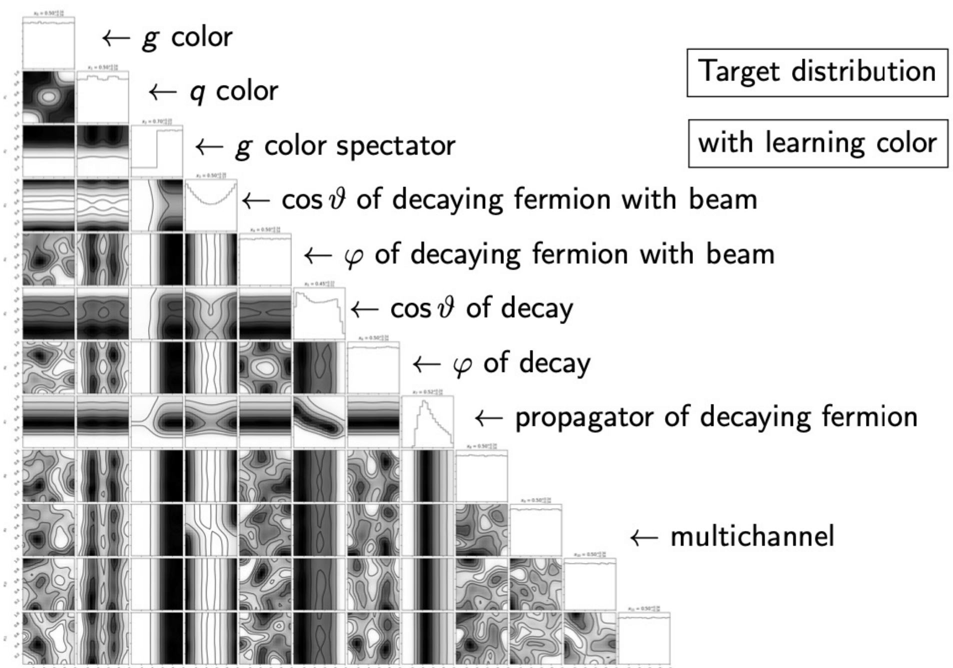
Slide credit: G. Kanwar

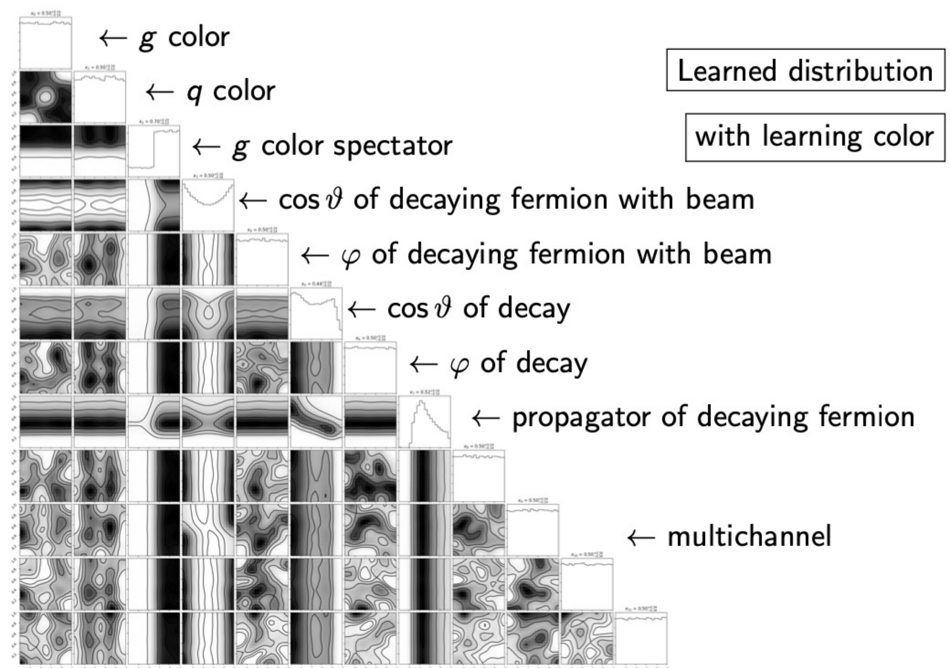
Event Generation with Normalizing Flows



arXiv: 2001.05486, ML:ST
arXiv: 2001.10028, PRD
Slide credit: C. Krause

Example: Learning $e^+e^- \to 3j$





Diffusion Models

Use variational lower bound

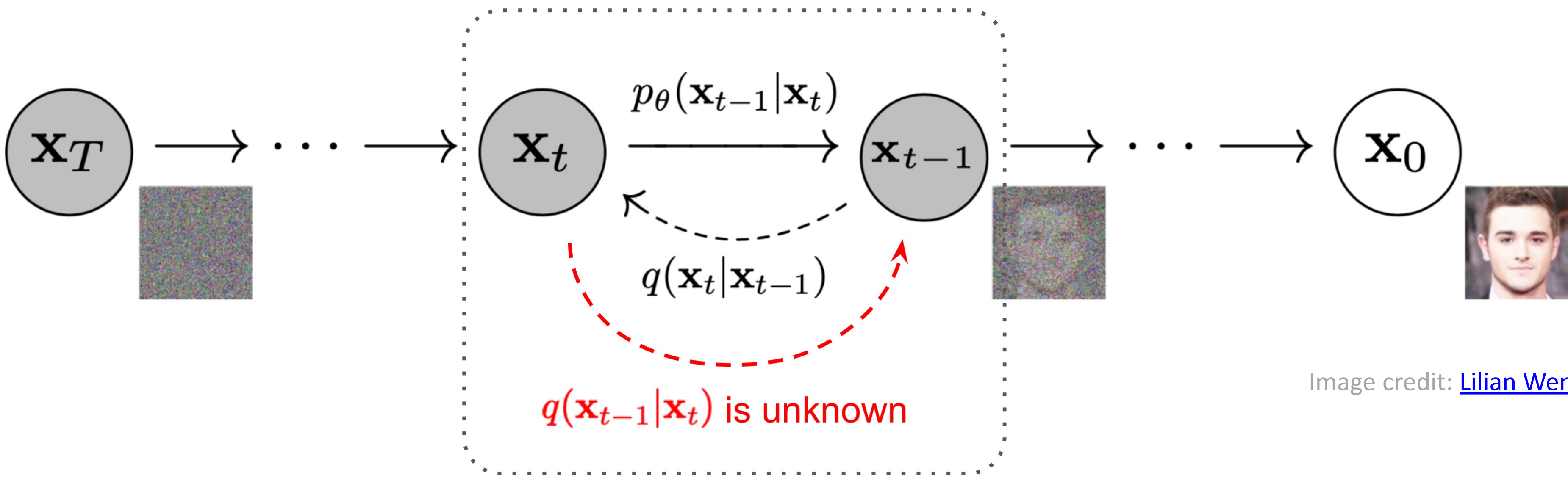
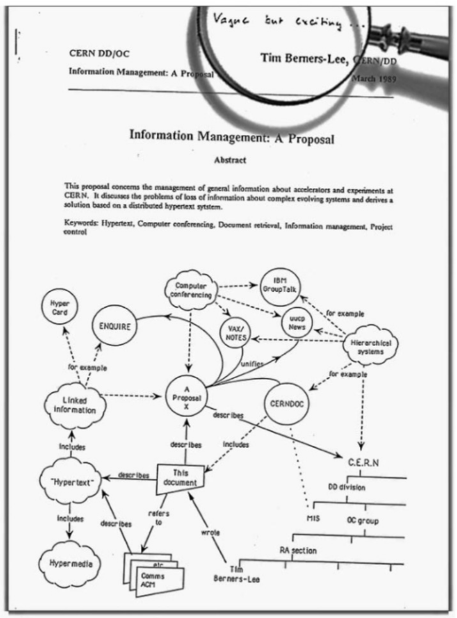


Image credit: [Lilian Weng]

- Iteratively add noise to data,
  Train model to learn how to denoise step by step

Some Final Thoughts

Since Tim Berners-Lee Invented the World Wide Web…


CERN DD/OC
Information Management: A Proposal
Tim Berners-Lee, CERN/DD
March 1989

Information Management: A Proposal

Abstract

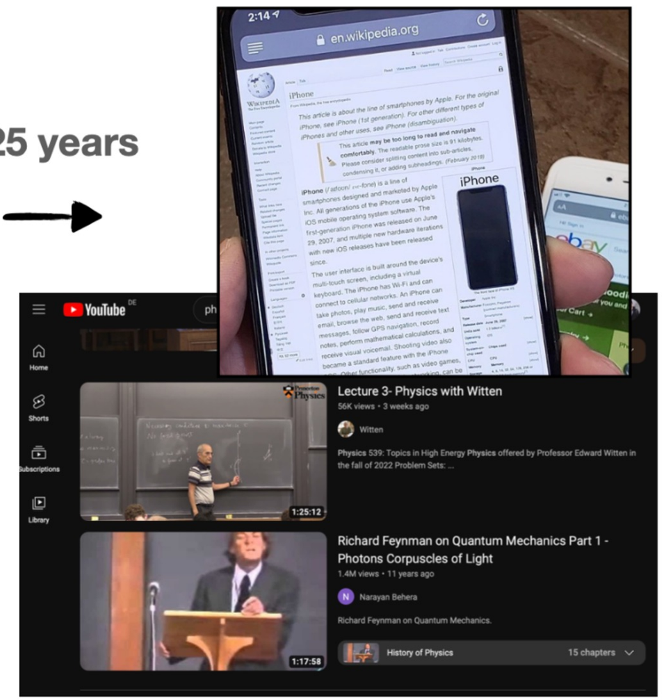
This proposal concerns the management of general information about accelerators and experiments at CERN. It discusses the problems of loss of information about complex evolving systems and derives a solution based on a distributed hypertext system.

Keywords: Hypertext, Computer conferencing, Document retrieval, Information management, Project control

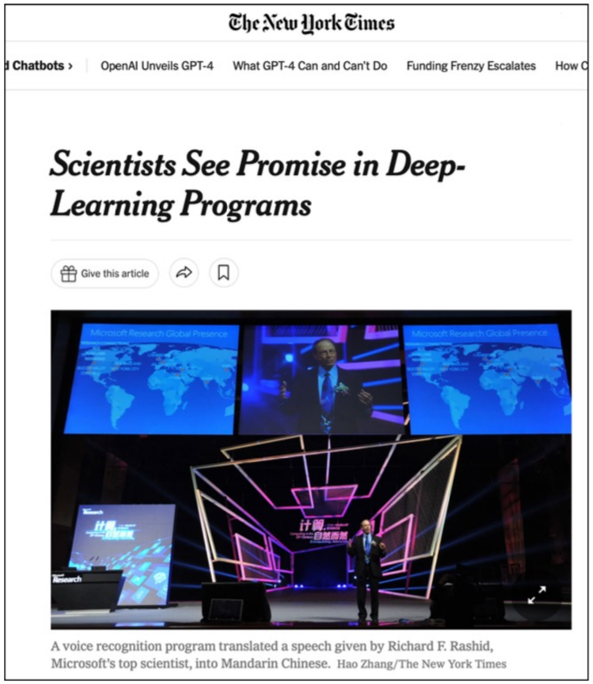
1989

~10 years

~25 years



Still Early for Deep Learning, Where Will We be in 25 Years?


The New York Times

Chatbots › OpenAI Unveils GPT-4 What GPT-4 Can and Can't Do Funding Frenzy Escalates How C

Scientists See Promise in Deep-Learning Programs

Give this article

A voice recognition program translated a speech given by Richard F. Rashid, Microsoft's top scientist, into Mandarin Chinese. Hao Zhang/The New York Times

2012

~10 years

Prompt: `Several giant wooly mammoths approach treading through a snowy meadow […]`
[OpenAI Sora]

~25 years

?

Slide credit: L. Heinrich

Do These Models Know Physics?... Maybe Not Yet

Credit: Jim Fan + Sora

Summary

- Deep neural networks are an extremely powerful class of models

- We can express our inductive bias about a system in terms of model design, and can be adapted to a many types of data

- Even beyond classification and regression, deep neural networks allow powerful unsupervised learning and Generative modeling!

Backup

Explicit Density Estimation with Normalizing Flows

Reminder: Calculus Change of Variables

$$\int f(g(x)) \frac{\partial g(x)}{dx} dx = \int f(u) du \qquad \text{where } u = g(x)$$

Multivariate:

$$\int f(g(\boldsymbol{x})) \left| \det \frac{\partial g(\boldsymbol{x})}{d\boldsymbol{x}} \right| d\boldsymbol{x} = \int f(\boldsymbol{u}) d\boldsymbol{u} \quad \text{where } \boldsymbol{u} = g(\boldsymbol{x})$$

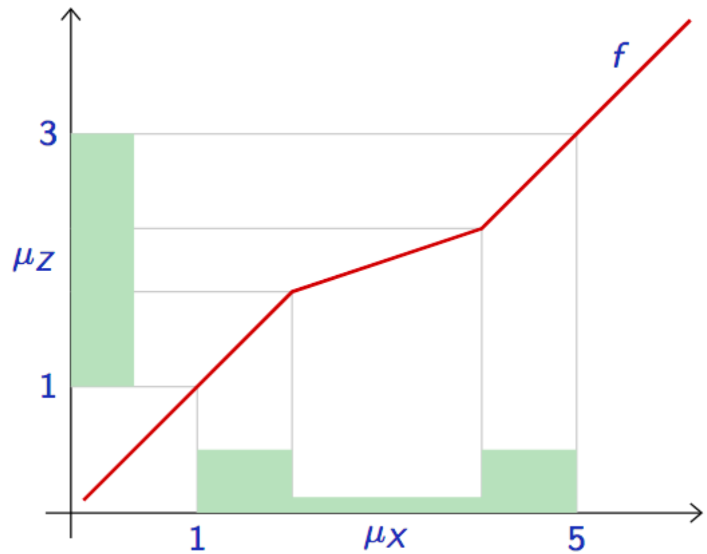
Determinant of Jacobian
of the transformation

→ Change of volume

Change of Variables in Probability

- If f is continuous, invertible, differentiable, and $x = f^{-1}(z) \equiv \phi(z)$ then

$$p_x(\boldsymbol{x}) = p_z(\boldsymbol{z}) \left| \det \left(\frac{\partial \phi(\boldsymbol{z})}{d\boldsymbol{z}} \right)^{-1} \right| \quad \text{where } \boldsymbol{x} = \phi(\boldsymbol{z})$$



The term $\left| \det \left(\frac{\partial \phi(\boldsymbol{z})}{d\boldsymbol{z}} \right)^{-1} \right|$ accounts for the local stretching of space

Change of Variables with Neural Networks

- If f is continuous, invertible, differentiable, and $x = f^{-1}(z) \equiv \phi(z)$ then

$$p_x(\boldsymbol{x}) = p_z(\boldsymbol{z}) \left| \det\left(\frac{\partial \phi(\boldsymbol{z})}{d\boldsymbol{z}}\right)^{-1} \right| \quad \text{where } \boldsymbol{x} = \phi(\boldsymbol{z})$$

- $x =$ data we want to model, $z =$ known noise

- $\phi_\theta(z)$ will be a neural network with parameters θ
  - Must be continuous, invertible, differentiable

- Output of ϕ is a potential sample x
  - **Learn the right ϕ**: adjust weights θ to maximize data probability (formula above)

Change of Variables with Neural Networks

- If f is continuous, invertible, differentiable, and $x = f^{-1}(z) \equiv \phi(z)$ then

$$p_x(\boldsymbol{x}) = p_z(\boldsymbol{z}) \left|\det\left(\frac{\partial\phi(\boldsymbol{z})}{d\boldsymbol{z}}\right)^{-1}\right| \quad \text{where } \boldsymbol{x} = \phi(\boldsymbol{z})$$

- x = data we want to model, z = known noise

$\phi(\boldsymbol{z})$ neural network
- Input = a sample of noise
- Output = a sample of X

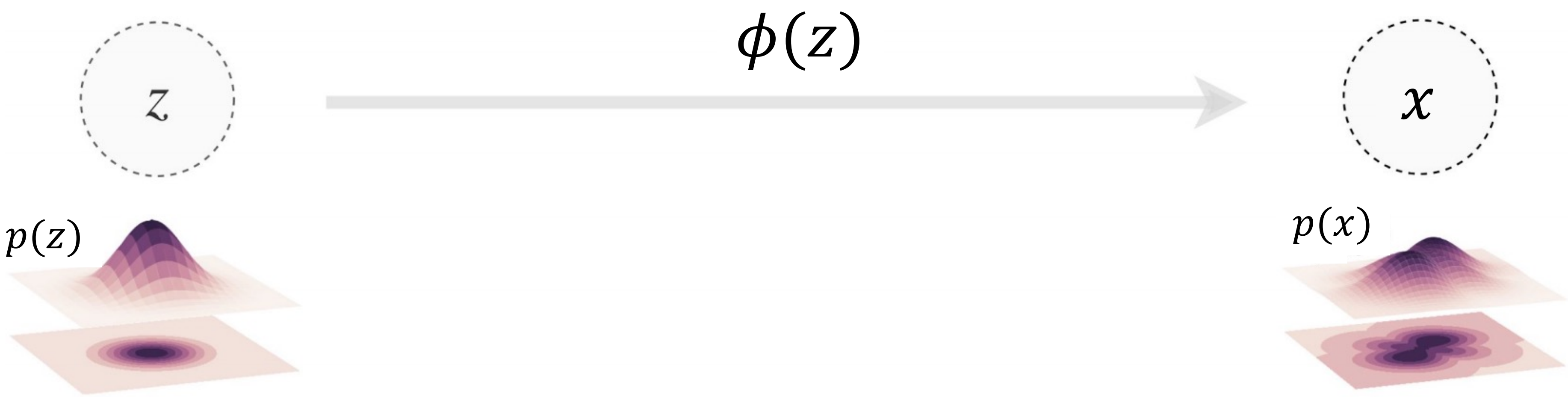
$\Leftrightarrow$

$\phi^{-1}(\boldsymbol{x})$ inverse
- Input = a sample X
- Output = a sample of noise

- Calculate the probability of a sample using the formula above

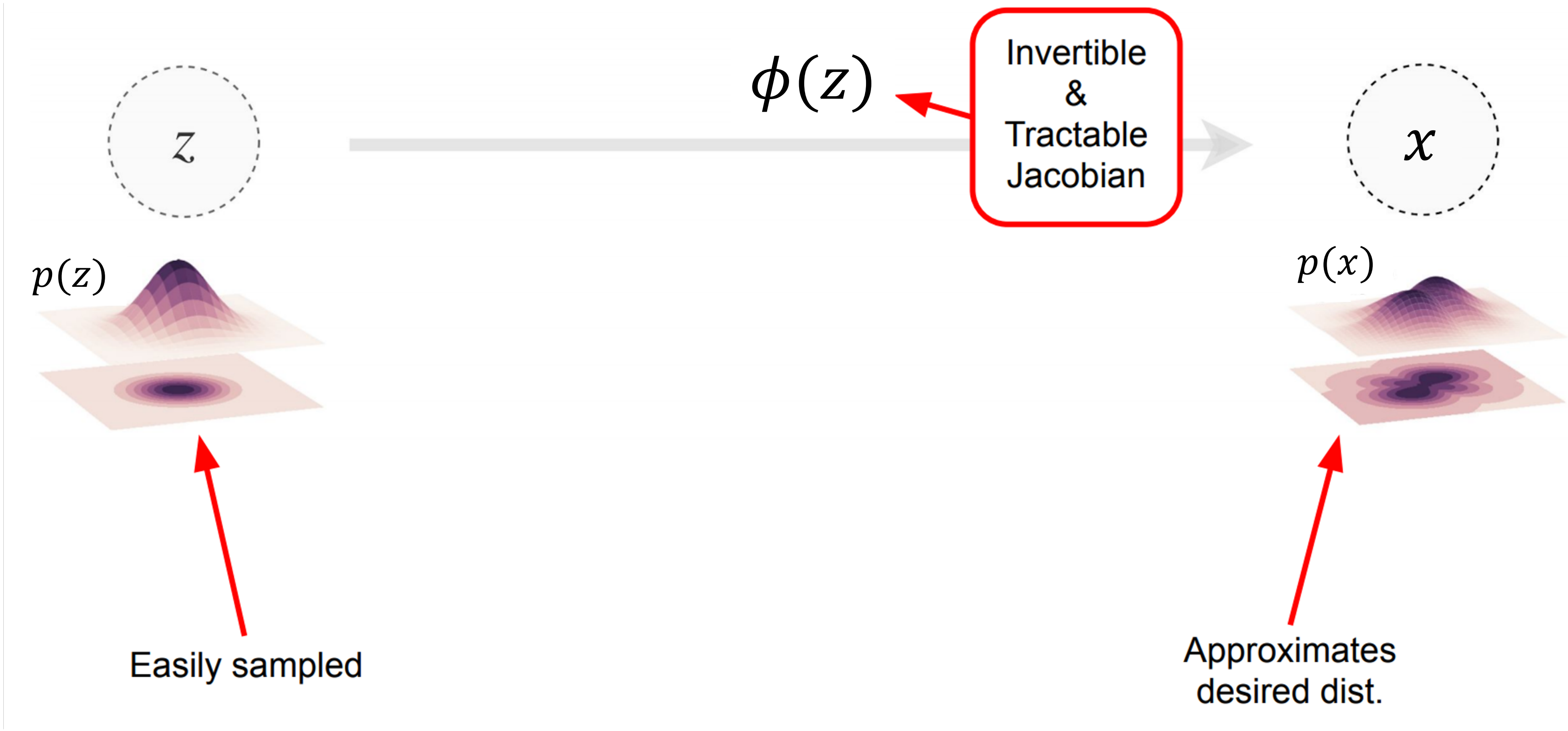
Normalizing Flows

$$p_x(\boldsymbol{x}) = p_z(\boldsymbol{z}) \left| \det \left(\frac{\partial \phi(\boldsymbol{z})}{d\boldsymbol{z}} \right)^{-1} \right|$$



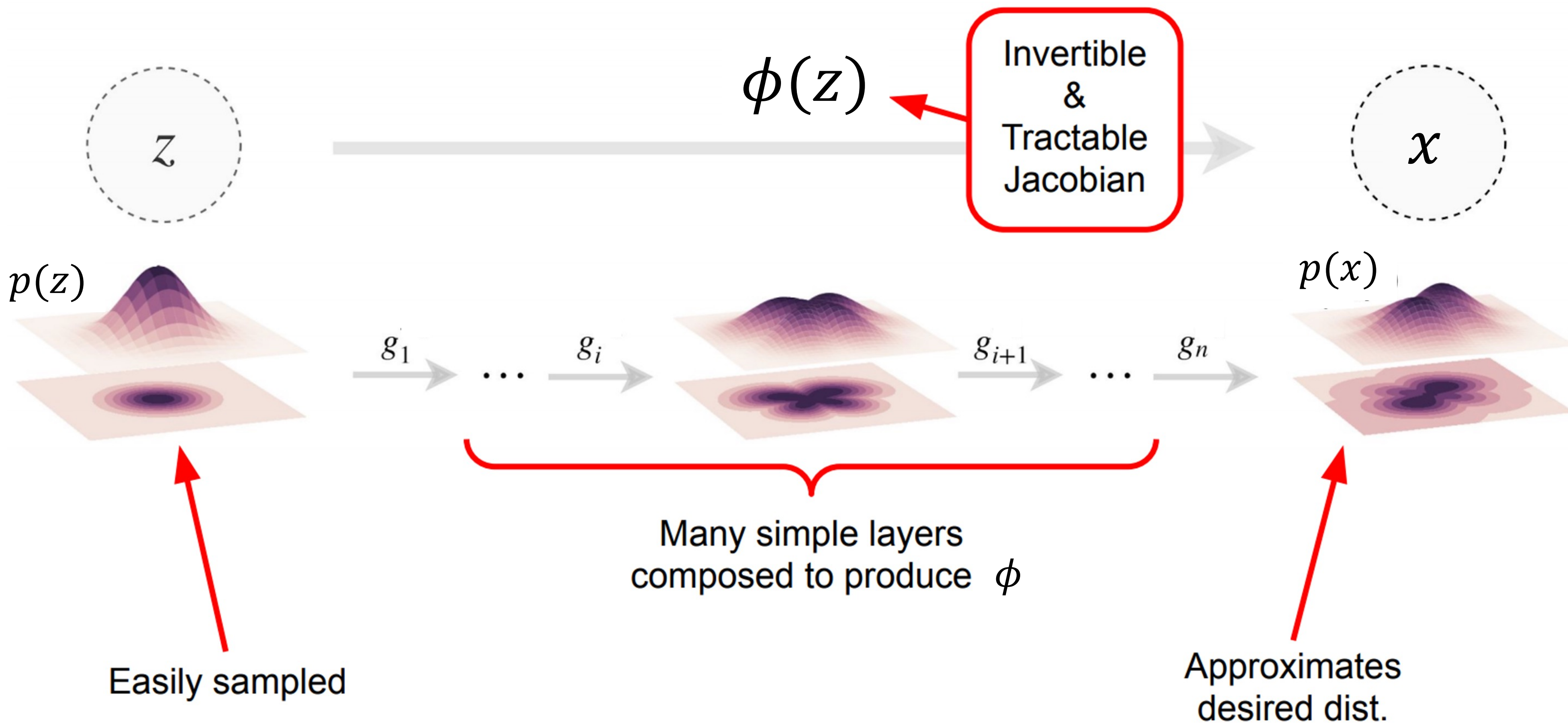
Slide credit: [G. Kanwar]

Normalizing Flows



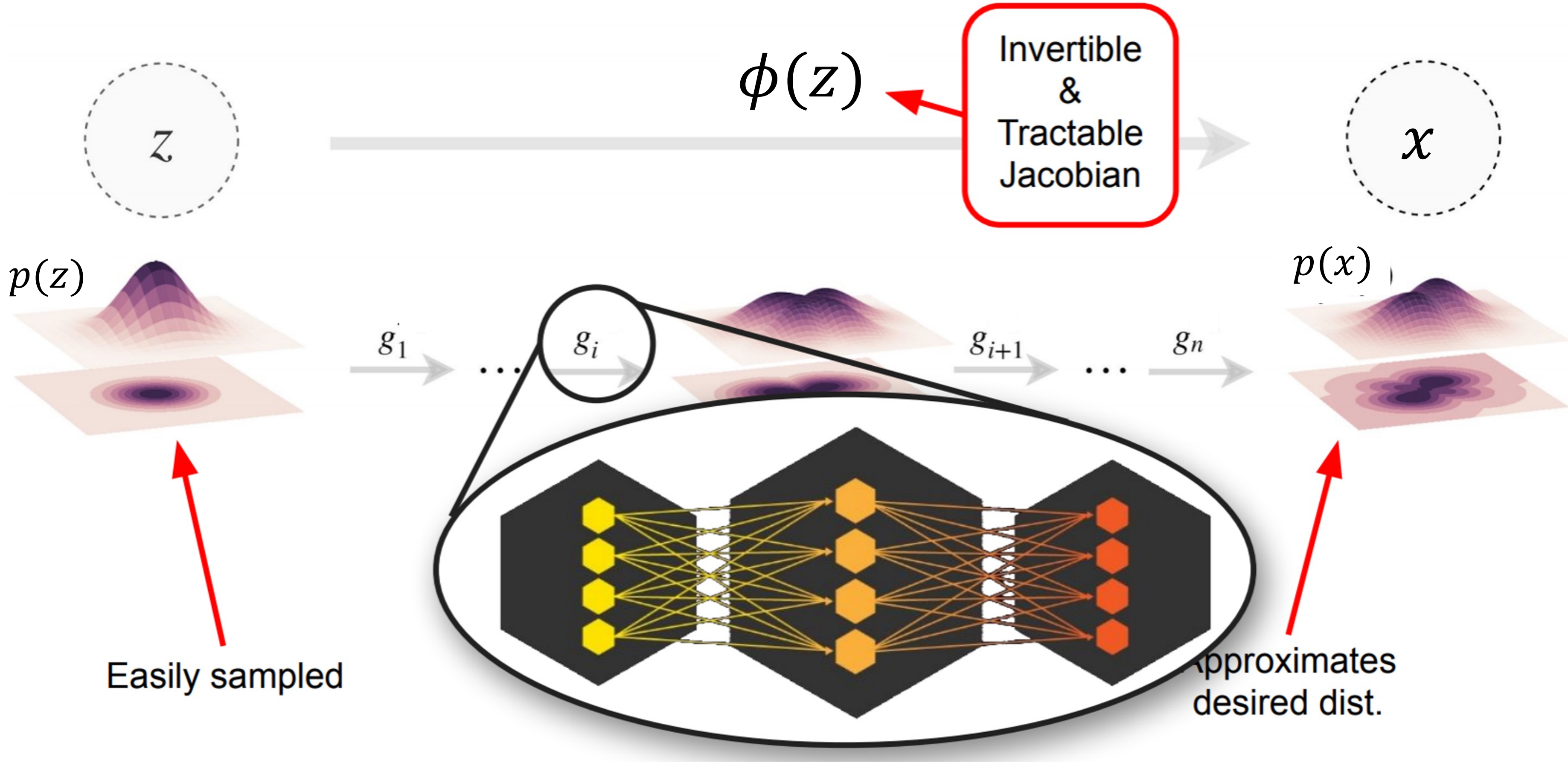
Slide credit: [G. Kanwar]

Normalizing Flows



Slide credit: [G. Kanwar]

Normalizing Flows



Slide credit: [G. Kanwar]

Normalizing Flows

$$\begin{tikzcd}[column sep=huge, row sep=huge]
x \arrow[r, shift left, "\phi^{-1}(x)"] \arrow[d, dashed, no head] & z \arrow[l, shift left, "\phi(z)"] \arrow[d, dashed, no head] \\
p_x(x) \arrow[r, shift left, "p_z(\phi^{-1}(x))\ \left|\det\left(\frac{\partial\phi^{-1}(\boldsymbol{x})}{d\boldsymbol{x}}\right)\right|"] & p_z(z) \arrow[l, shift left, "p_z(z)\ \left|\det\left(\frac{\partial\phi(\boldsymbol{z})}{d\boldsymbol{z}}\right)^{-1}\right|"]
\end{tikzcd}$$

Normalizing Flows Training

- **Learn** θ with maximum likelihood

$$\max_\theta p(x) = \max_\theta p_z\left(\phi_\theta^{-1}(x)\right)\left|\det\left(\frac{\partial \phi_\theta^{-1}(\boldsymbol{x})}{d\boldsymbol{x}}\right)\right|$$

  - Gradient descent on θ
  - Find transformation s.t. data is most likely

- **Benefits** once trained
  - Can evaluate p(x) for any point X
  - Can generate "new" data points
    - Sample noise: $z \sim p(z)$
    - Transform: $\phi(z) = x$

Example Normalizing Flow: Real NVP

- Data vector $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$

Functions f() and g() are neural networks

- Transformation

$\phi(z)$:
$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \phi_1(z) \\ \phi_2(z) \end{pmatrix} = \begin{pmatrix} z_1 \\ z_2 * f(z_1) + g(z_1) \end{pmatrix}$$

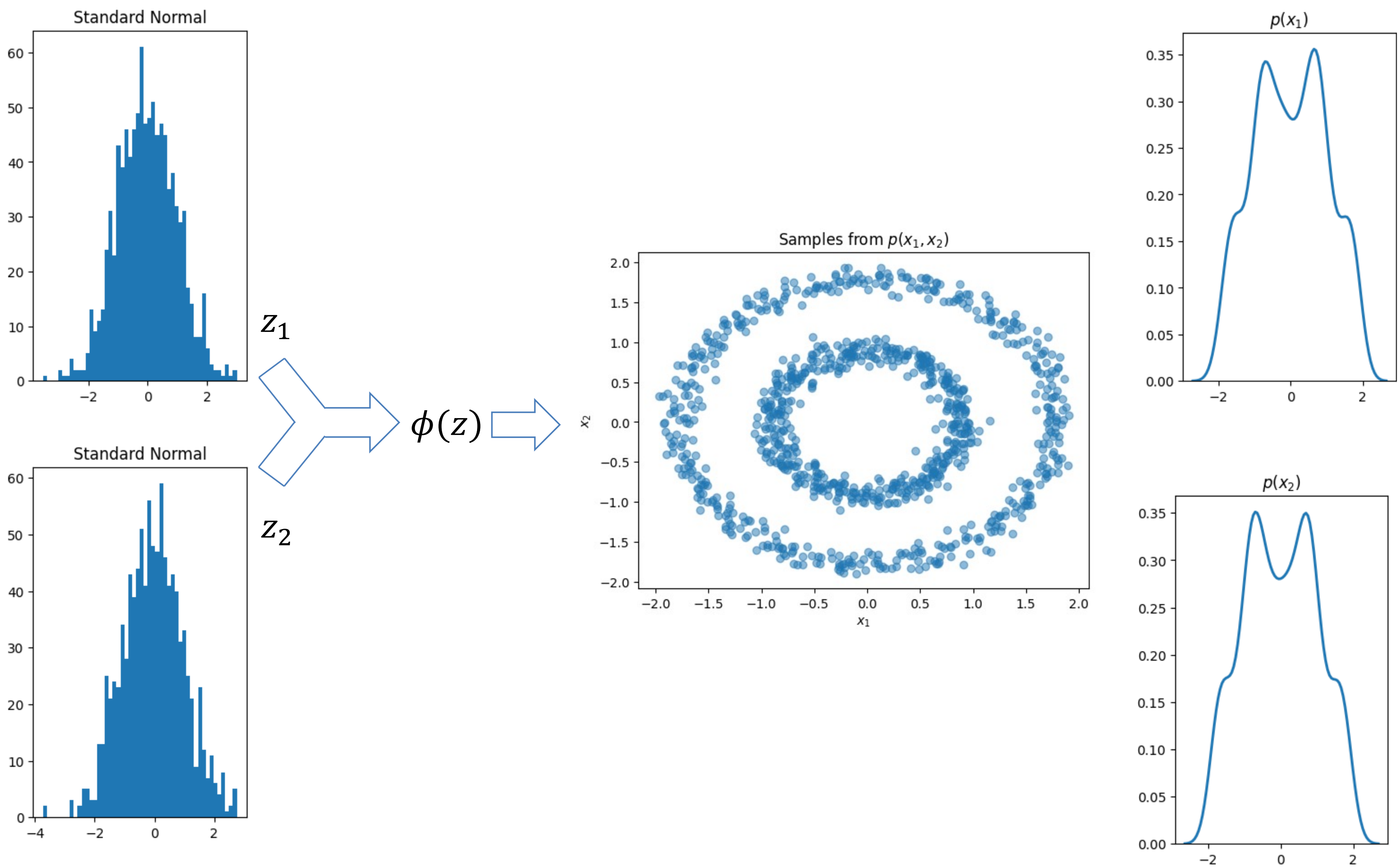
$\phi^{-1}(x)$:
$$\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} \phi_1^{-1}(x) \\ \phi_2^{-1}(x) \end{pmatrix} = \begin{pmatrix} x_1 \\ (x_2 - g(x_1))/f(x_1) \end{pmatrix}$$

- Determinant:

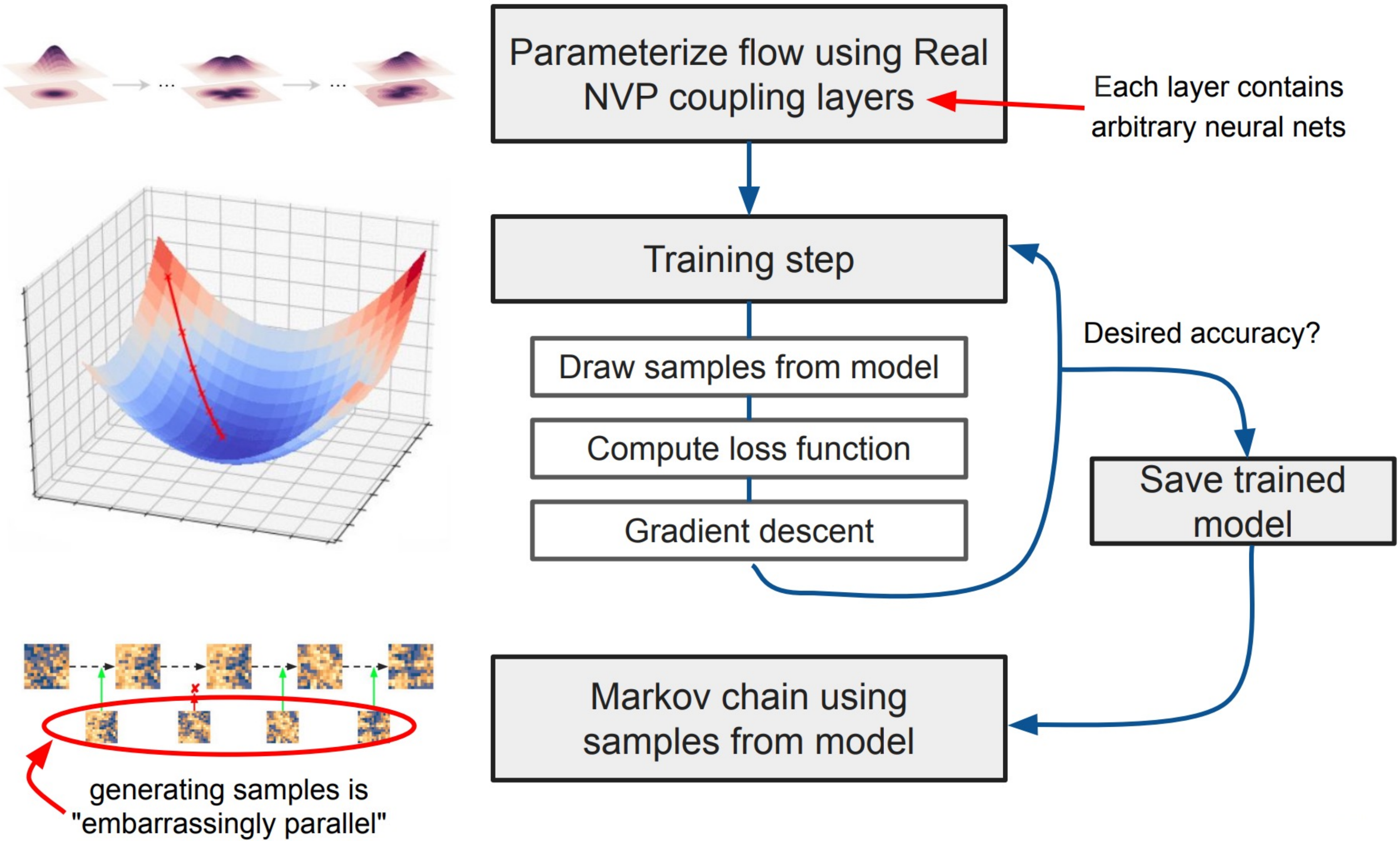
Jacobian is lower triangular

$$\det\left(\frac{\partial \phi(\boldsymbol{z})}{d\boldsymbol{z}}\right) = \det\begin{pmatrix} 1 & 0 \\ \left(\frac{\partial \phi_2(z)}{dz_1}\right) & f(z_1) \end{pmatrix} = f(z_2)$$

Example Normalizing flow



Applications: Sampling in Lattice QCD



Slide credit: [G. Kanwar]

GANS

Another Way To Do Generative Modeling…

- Formulate as a two player game

- One player tries to output data that looks as real as possible

- Another player tries to compare real and fake data

- In this case we need:
  1. A *generator* that can produce samples
  2. A measure of *not too far from the real data*

Generative Adversarial Network (GAN)

- **Generator network** $\boldsymbol{g_\theta(z)}$ with parameters θ
  - Map sample from known $p(z)$ to sample in data space

$$x = g_\theta(z) \quad z \sim p(z)$$

  - We don't know what the generated distribution $p_\theta(x)$ is, but we can sample from it → *Implicit Model*

Generative Adversarial Network (GAN)

Goodfellow et. al., 2014

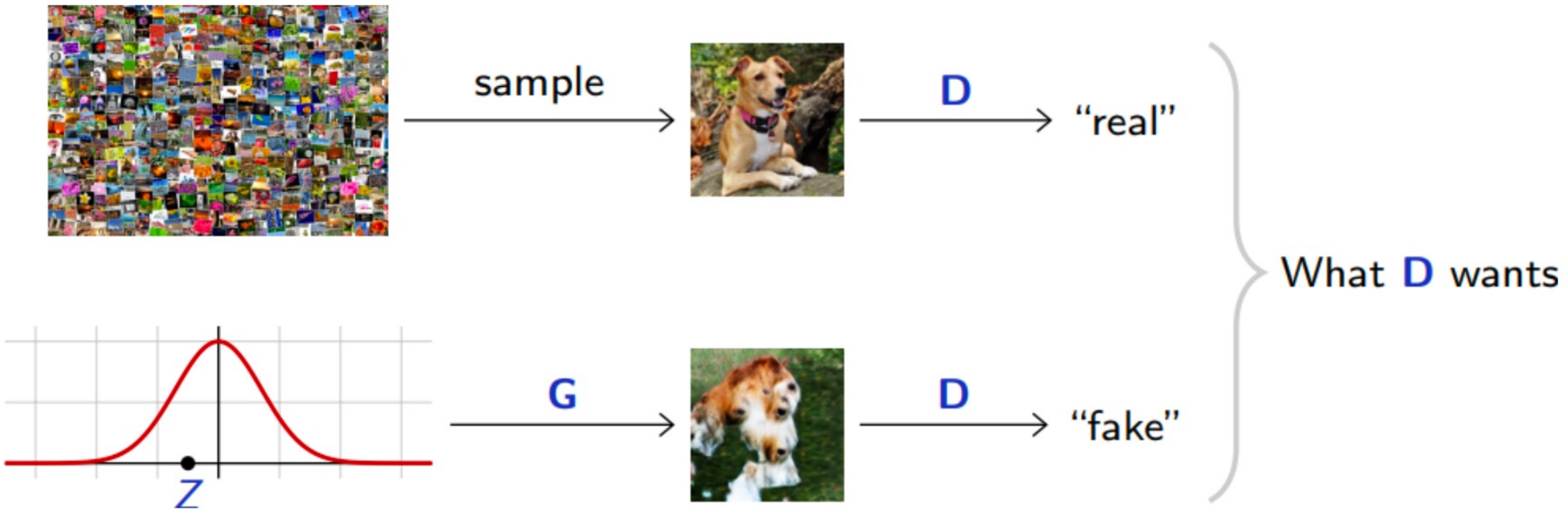
- **Generator network $\boldsymbol{g_\theta(z)}$** with parameters θ
  - Map sample from known $p(z)$ to sample in data space

$$x = g_\theta(z) \quad z \sim p(z)$$

  - We don't know what the generated distribution $p_\theta(x)$ is, but we can sample from it → *Implicit Model*

- **Discriminator Network $\boldsymbol{d_\phi(x)}$** with parameters ϕ
  - Classifier trained to distinguish between real and fake data
  - Classifier is learning to predict $p(y = real \mid x)$
  - This classifier is our measure of *not too far from the real data*

GAN Setup



- Generator's goal is to produce *fake* data that tricks the discriminator to think it is *real* data

- Discriminator wants to miss-classify data as real or fake as little as possible

- The setup is *adversarial* because the two networks have opposing objectives

Fleuret, Deep Learning Course

GAN Objective

- Data
  - Real data samples: $\{x_i, y_i = 1\}$
  - Fake data samples: $\{\tilde{x}_i = g_\theta(z_i), \tilde{y}_i = 0\}$ with: $z_i \sim p(z)$

GAN Objective

- Data
  - Real data samples: $\{x_i, y_i = 1\}$
  - Fake data samples: $\{\tilde{x}_i = g_\theta(z_i), \tilde{y}_i = 0\}$ with: $z_i \sim p(z)$
- For a fixed generator, can train discriminator by minimizing the cross entropy

$$L(\phi) = -\frac{1}{2N} \sum_{i=1}^{N} \left[y_i \log d_\phi(x_i) + (1 - \tilde{y}_i) \log(1 - d_\phi(\tilde{x}_i)\ ) \right]$$

GAN Objective

- Data
  - Real data samples: $\{x_i, y_i = 1\}$
  - Fake data samples: $\{\tilde{x}_i = g_\theta(z_i), \tilde{y}_i = 0\}$ with: $z_i \sim p(z)$
- For a fixed generator, can train discriminator by minimizing the cross entropy

$$L(\phi) = -\frac{1}{2N} \sum_{i=1}^{N} \left[y_i \log d_\phi(x_i) + (1 - \tilde{y}_i) \log(1 - d_\phi(\tilde{x}_i)\ ) \right]$$

$$= -\frac{1}{2N} \sum_{i=1}^{N} \left[\log d_\phi(x_i) + \log(1 - d_\phi(g_\theta(z_i))\ ) \right]$$

GAN Objective

- Data
  - Real data samples: $\{x_i, y_i = 1\}$
  - Fake data samples: $\{\tilde{x}_i = g_\theta(z_i), \tilde{y}_i = 0\}$ with: $z_i \sim p(z)$
- For a fixed generator, can train discriminator by minimizing the cross entropy

$$L(\phi) = -\frac{1}{2N}\sum_{i=1}^{N}\left[y_i \log d_\phi(x_i) + (1-\tilde{y}_i)\log(1-d_\phi(\tilde{x}_i)\ )\right]$$

$$= -\frac{1}{2N}\sum_{i=1}^{N}\left[\log d_\phi(x_i) + \log(1-d_\phi(g_\theta(z_i))\ )\right]$$

$$= -\mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\log d_\phi(x)\right] - \mathbb{E}_{z\sim p(z)}\left[\log(1-d_\phi(g_\theta(z))\ )\right]$$

GAN Objective

- However, generator isn't fixed… have to train it!

GAN Objective

- However, generator isn't fixed… have to train it!
- Consider objective as a *value function* of ϕ and θ

$$V(\phi, \theta) = \mathbb{E}_{x \sim p_{\text{data}}(x)}\Big[\log d_\phi(x)\Big] + \mathbb{E}_{z \sim p(z)}\Big[\log(1 - d_\phi(g_\theta(z)))\Big]$$

GAN Objective

- However, generator isn't fixed… have to train it!

- Consider objective as a *value function* of ϕ and θ

$$V(\phi, \theta) = \mathbb{E}_{x \sim p_{\text{data}}(x)}\Big[\log d_\phi(x)\Big] + \mathbb{E}_{z \sim p(z)}\Big[\log(1 - d_\phi(g_\theta(z)))\Big]$$

  - For fixed generator, $V(\phi, \theta)$ is high when discriminator is good, i.e. when generator is not producing good fakes

  - For a perfect discriminator, a good generator will confuse discriminator and $V(\phi, \theta)$ will be low

GAN Objective

- However, generator isn't fixed… have to train it!

- Consider objective as a *value function* of ϕ and θ

$$V(\phi, \theta) = \mathbb{E}_{x \sim p_{\text{data}}(x)}\Big[\log d_\phi(x)\Big] + \mathbb{E}_{z \sim p(z)}\Big[\log(1 - d_\phi(g_\theta(z)))\Big]$$

  – For fixed generator, $V(\phi, \theta)$ is high when discriminator is good, i.e. when generator is not producing good fakes

  – For a perfect discriminator, a good generator will confuse discriminator and $V(\phi, \theta)$ will be low

- So our optimization goal becomes:

$$\theta^* = \arg\min_\theta \max_\phi V(\phi, \theta)$$

GAN Objective

- However, generator isn't fixed… have to train it!

- Consider objective as a *value function* of ϕ and θ

$$V(\phi, \theta) = \mathbb{E}_{x \sim p_{\text{data}}(x)} \Big[\log d_\phi(x) \Big] + \mathbb{E}_{z \sim p(z)} \Big[\log(1 - d_\phi(g_\theta(z))\ ) \Big]$$

  - For fixed generator, $V(\phi, \theta)$ is high when discriminator is good, i.e. when generator is not producing good fakes

  - For a perfect discriminator, a good generator will confuse discriminator and $V(\phi, \theta)$ will be low

- So our optimization goal becomes:

$$\theta^* = \arg\min_\theta \max_\phi V(\phi, \theta)$$

NOTE: can prove that minimax solution corresponds to generator that perfectly reproduces data distribution

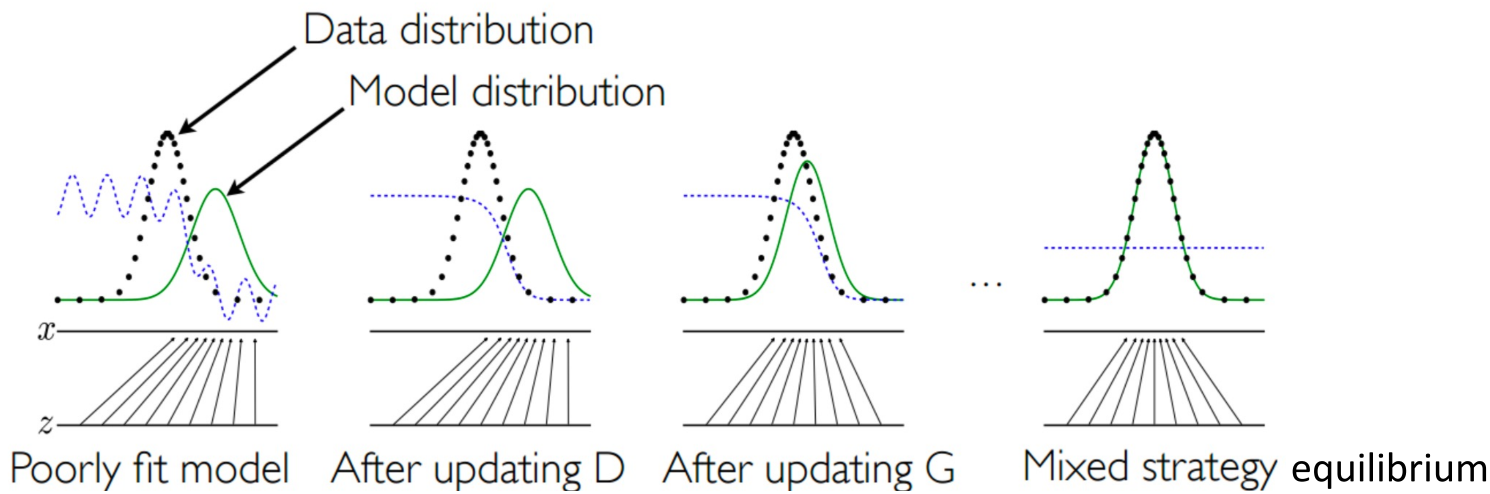
$$q_{\theta^*}(x) = p_{data}(x)$$

- Alternating Gradient descent to solve the min-max problem:

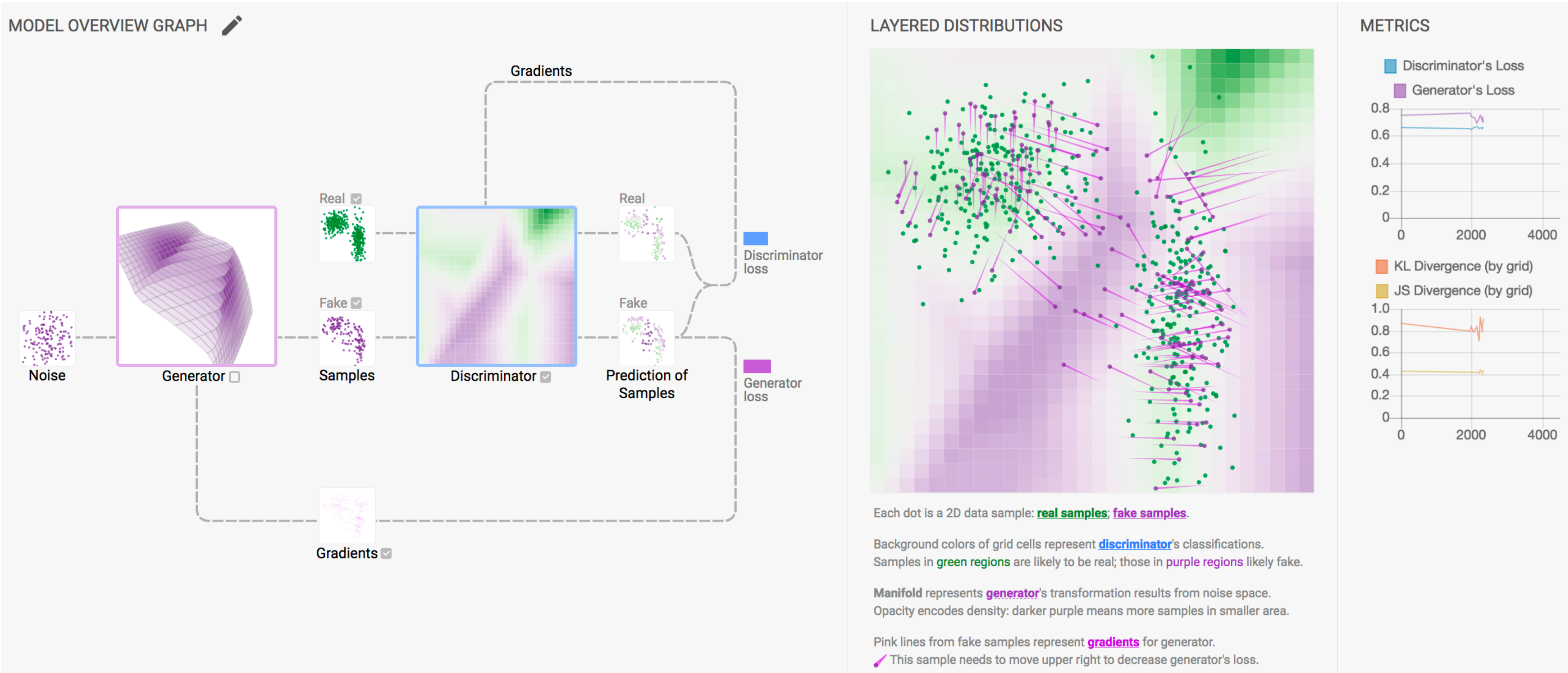
$$\theta \leftarrow \theta - \gamma \nabla_\theta V(\phi, \theta) = \theta - \gamma \frac{\partial V}{\partial d} \frac{\partial (d_\phi)}{\partial g} \frac{\partial g_\theta}{\partial \theta}$$

$$\phi \leftarrow \phi - \gamma \nabla_\phi V(\phi, \theta) = \phi - \gamma \frac{\partial V}{\partial d} \frac{d(d_\phi)}{d\phi}$$

- For each θ step, take k steps in ϕ to keep discriminator near optimal



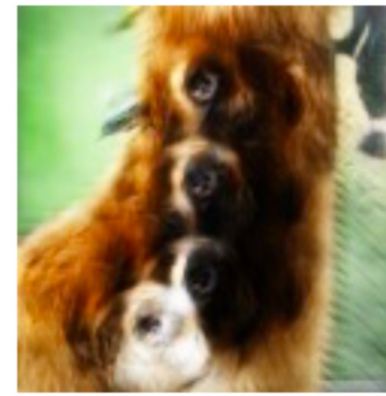
GAN Training Example



[GAN Lab Demo]

Examples

Goodfellow et. al., 2014



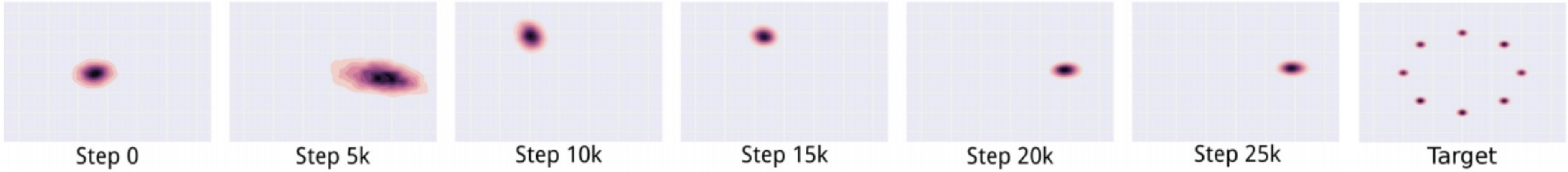
Not so good

Goodfellow 2016

Radford et al, 2015

Challenges

- **Oscillations without convergence**: unlike standard loss minimization, alternating stochastic gradient descent has no guarantee of convergence.
- **Vanishing gradients**: if classifier is too good, value function saturates → no gradient to update generator
- **Mode collapse**: generator models only a small sub-population, concentrating on a few data distribution modes.
- **Difficult to assess performance**, when are generated data good enough?

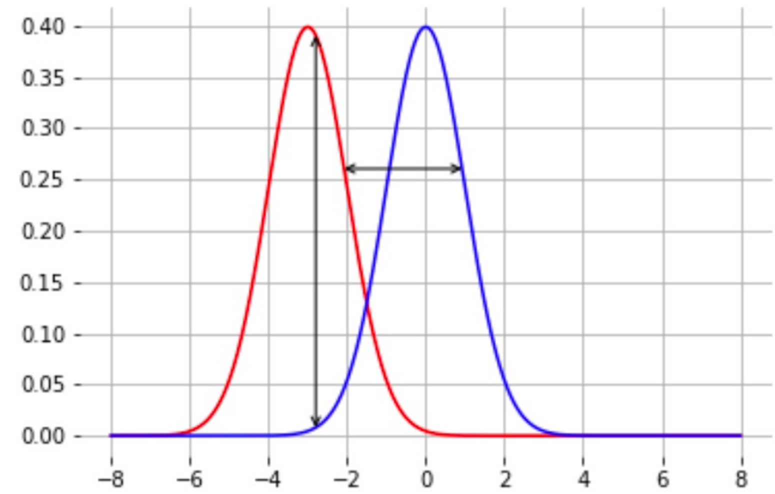


Slide credit: [G. Louppe]

Mode collapse (Metz et al, 2016)

Improving GANS

- Standard GANS compare real and fake distributions with Jensen-Shannon Divergence, “vertically”
- Wasserstein-GAN (Arjovsky et al, 2017) compares “horizontally” with Wasserstein-1 distance (a.k.a. Earth Movers distance)
- Substantially improves *vanishing gradient* and *mode collapse* problems!



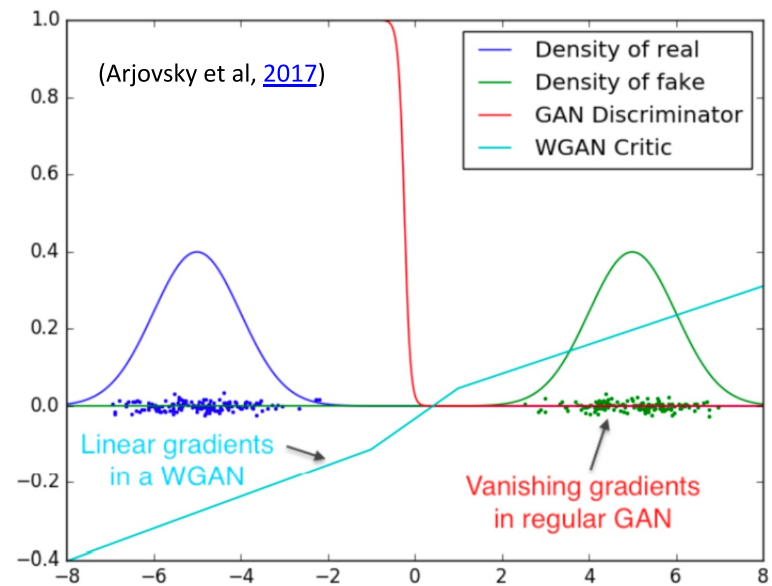
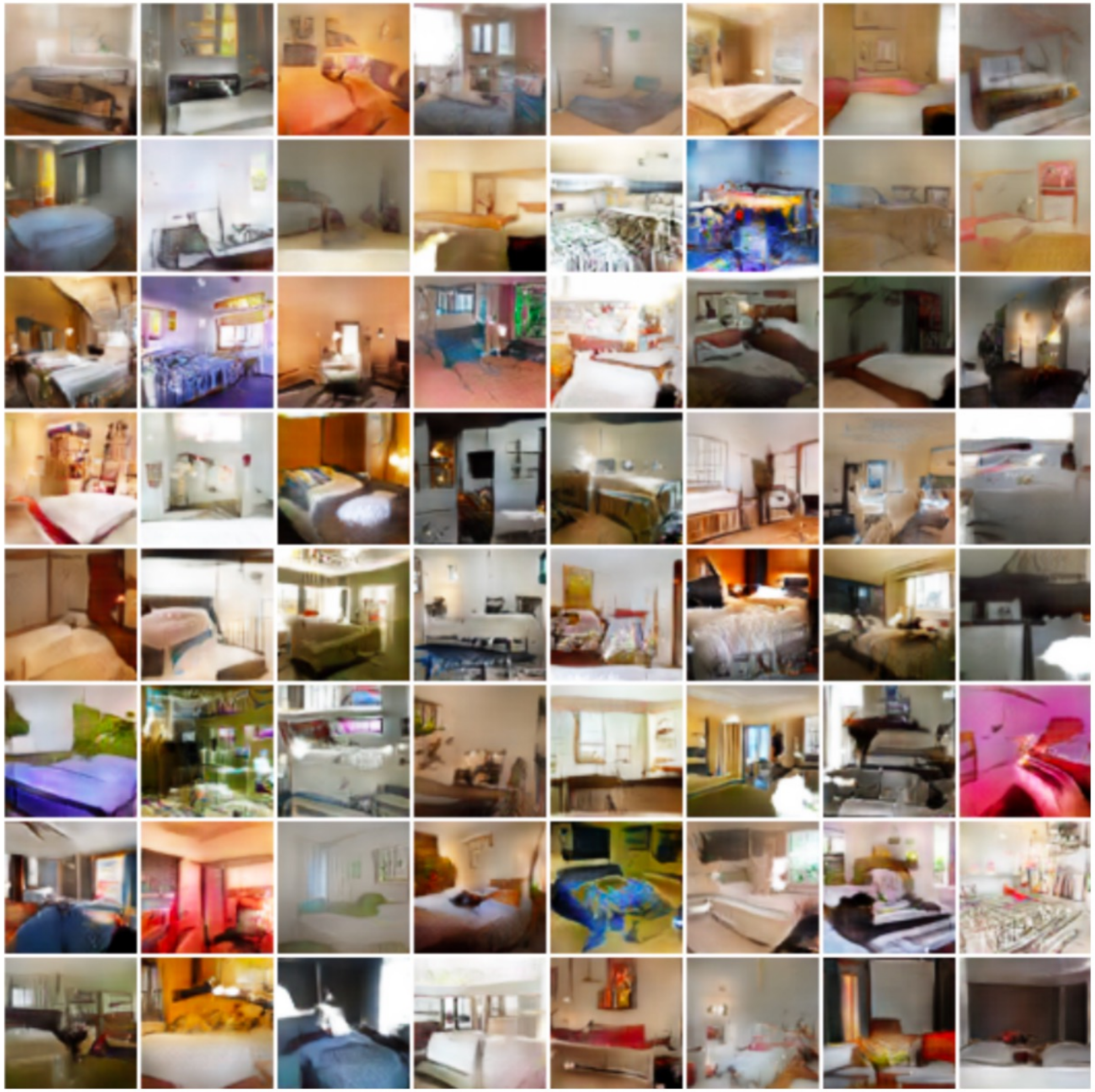


Figure 2: Optimal discriminator and critic when learning to differentiate two Gaussians. As we can see, the discriminator of a minimax GAN saturates and results in vanishing gradients. Our WGAN critic provides very clean gradients on all parts of the space.

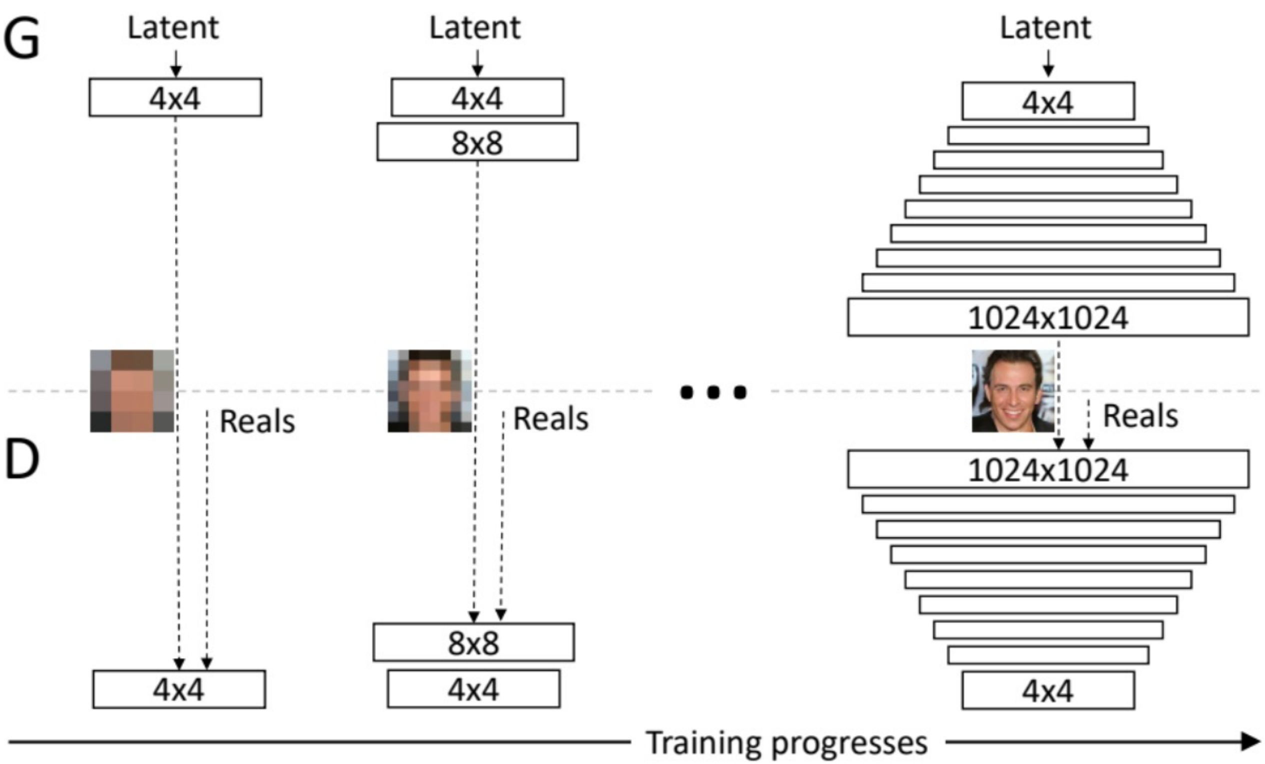
WGAN Examples



(Arjovsky et al, 2017)

Scaling Up

Progressive GAN



(Karras et al, 2017)

Scaling Up

StyleGAN v2

(Karras et al, 2019)

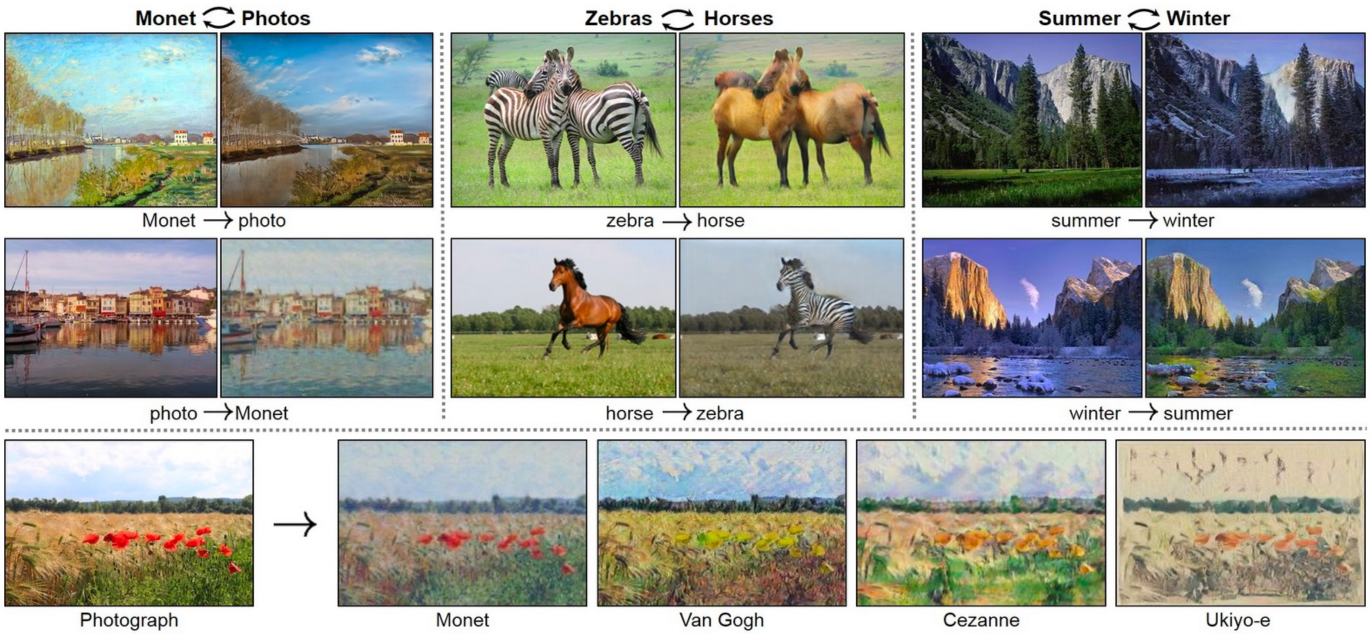
BigGAN

(Brock et al, 2018)

Applications: Image-to-Image Translation with CycleGAN

- $p(z)$ doesn't have to be random noise

- CycleGAN uses *cycle-consistency loss* in addition to GAN loss
  - Translating from A→B→A should be consistent with original A



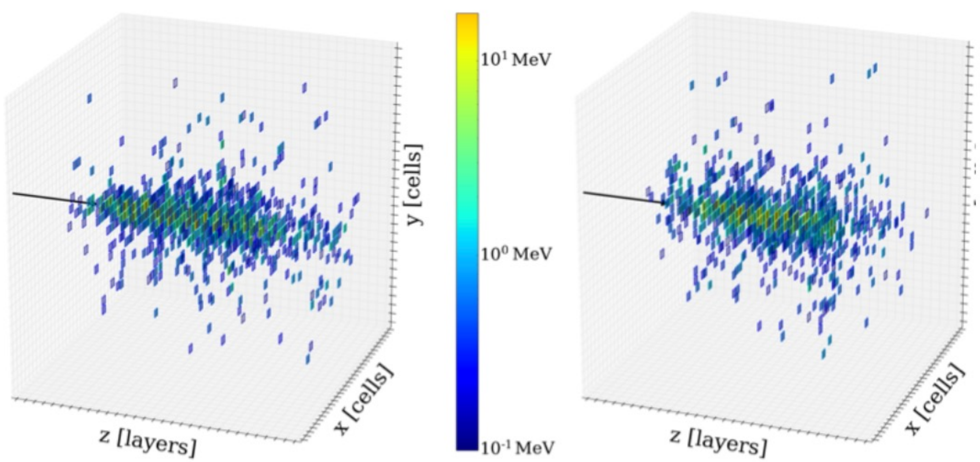
Applications: Text-to-Image Synthesis with StackGAN

Fig. 3: Example results by our StackGAN-v1, GAWWN [29], and GAN-INT-CLS [31] conditioned on text descriptions from CUB test set.

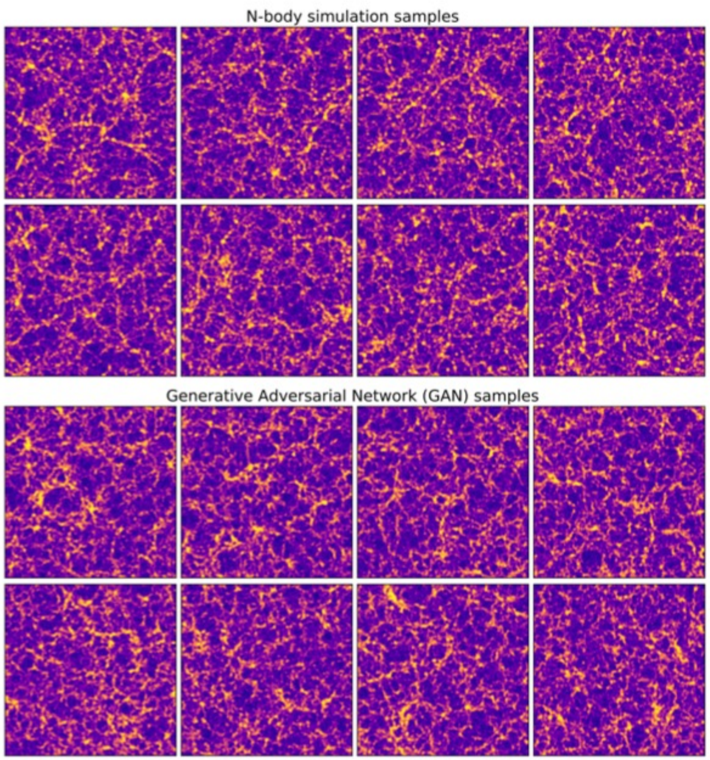
(Zhang et al, 2017)

Generative Models in Physics

- Often studied for fast approximate simulation, simulation-based inference, optimization, …



2005.05334



1801.09070