Machine Intelligence:: Deep Learning
Week 7

Beate Sick, Loran Avci, Oliver Dürr

Fun with normalizing flows.

Winterthur, 13. April. 2021

State of the Art in generative models

Figure 1.1 Progress in human face generation
(Source: "The Malicious Use of Artificial Intelligence: Forecasting, Prevention, and Mitigation," by Miles Brundage et al., 2018, https://arxiv.org/abs/1802.07228.)

Credit: Jakub Langr, Vladimir Bok (GANs in Action Manning) https://www.manning.com/books/gans-in-action

State of the art (2019) in generative models

https://thispersondoesnotexist.com/

These persons are not real. The pictures **1024x1024x3** are sampled from a GAN.

Do you spot some artefacts?

Definition: Generative Model [cs231n]

Given training data, generate new samples from same distribution.

Training data ~ $p_{data}(x)$

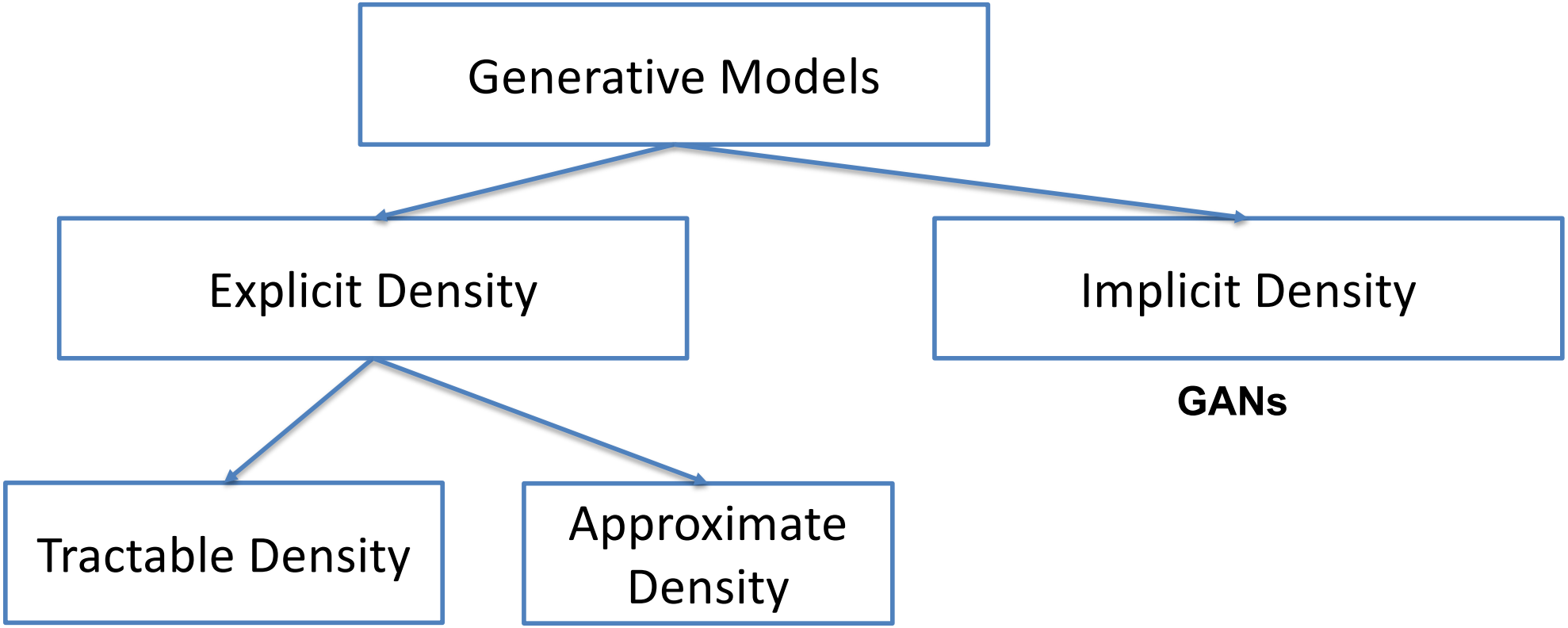
Generated samples ~ $p_{model}(x)$

Want to learn $p_{model}(x)$ similar to $p_{data}(x)$

Several flavors:

- Explicit density estimation: explicitly define and learn $p_{model}(x)$
- Implicit density estimation: learn model that can sample from $p_{model}(x)$ w/o explicitly having a density

Taxonomy of Generative Models



Normalizing Flows

- PixelCNN
- Wavenet
- Glow
- Nice RealNVP

Variational Autoencoder

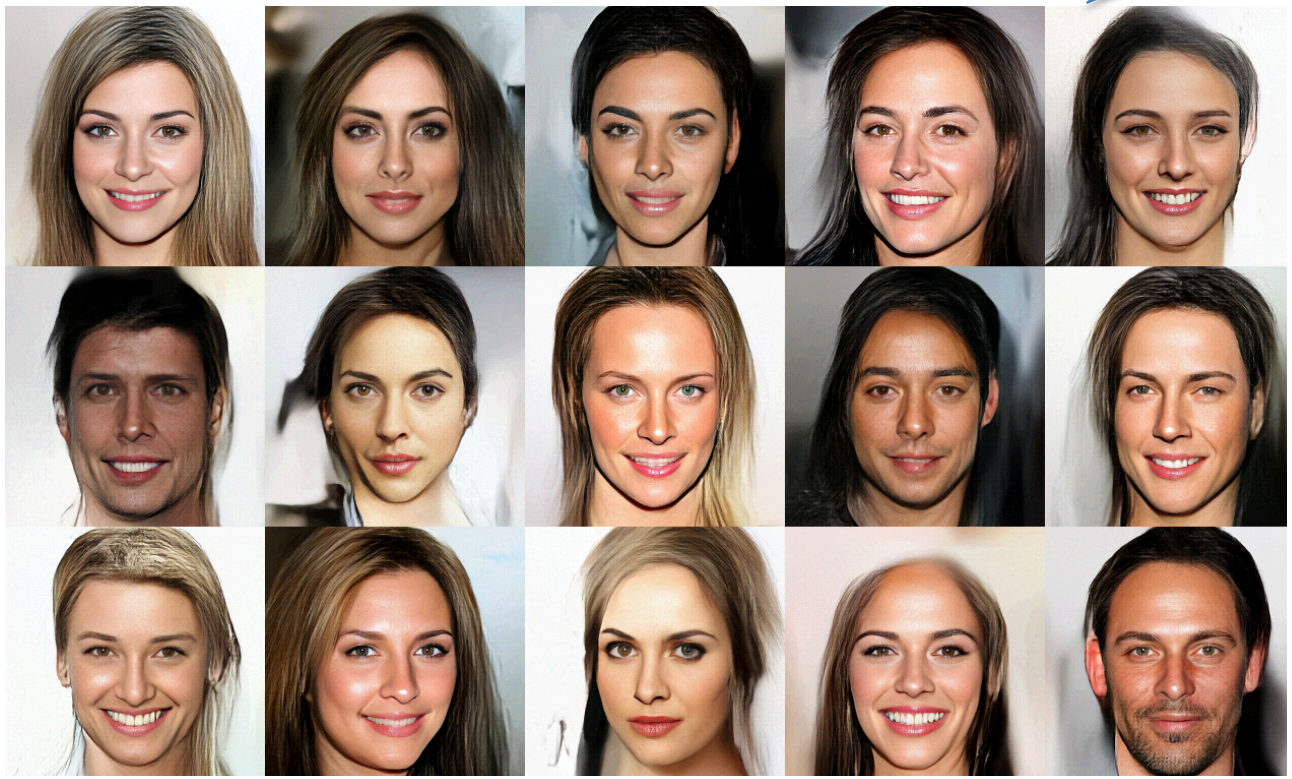
Figure simplified from CS231n

Normalizing Flow

Go with the flow

An introduction to normalizing flows

These people are not real they are generated samples using NF



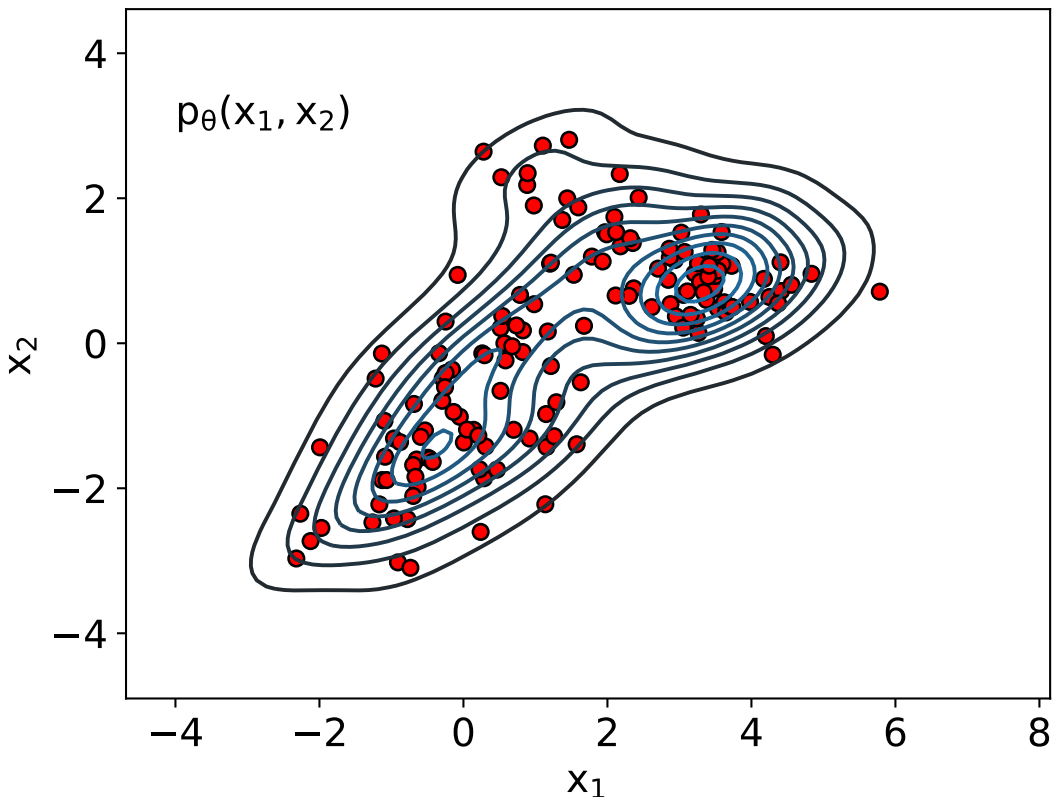
A bit of Motivation

- A the End of the lecture, you can create and understand something like:

- Look at the intermediate pictures, they look real.
- Persons no celebrities (not part of celebA-HQ used for training)

Normalizing Flows

- A novel method of parametric density estimation



- Example of parametric density estimation 2-D Gaussians with μ and Σ

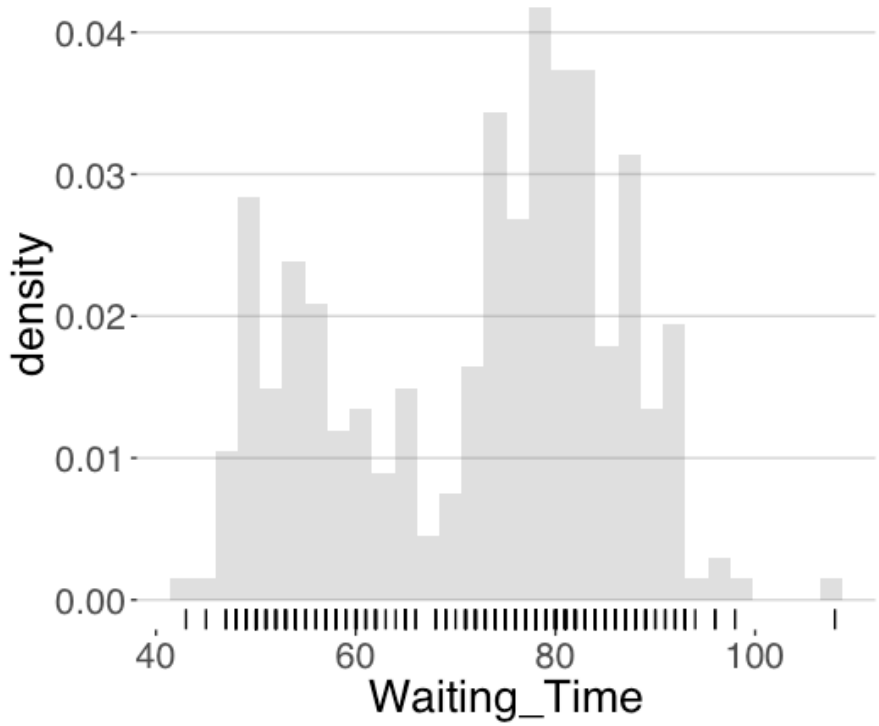
Theory: Name Some Distributions

- Gaussian
- Uniform
- Weibull
- Binomial
- Log-Normal

These are the distributions we have in our Toolbox.

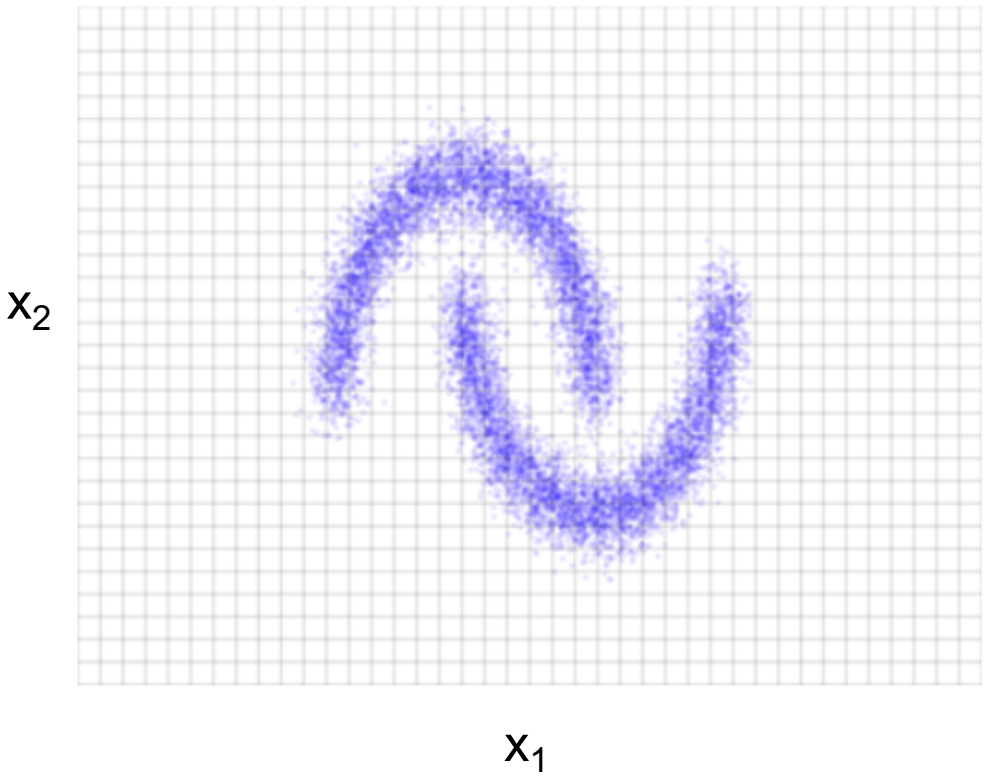
Is the reality like this?

Reality: Data (1-D)



What distribution can you use?

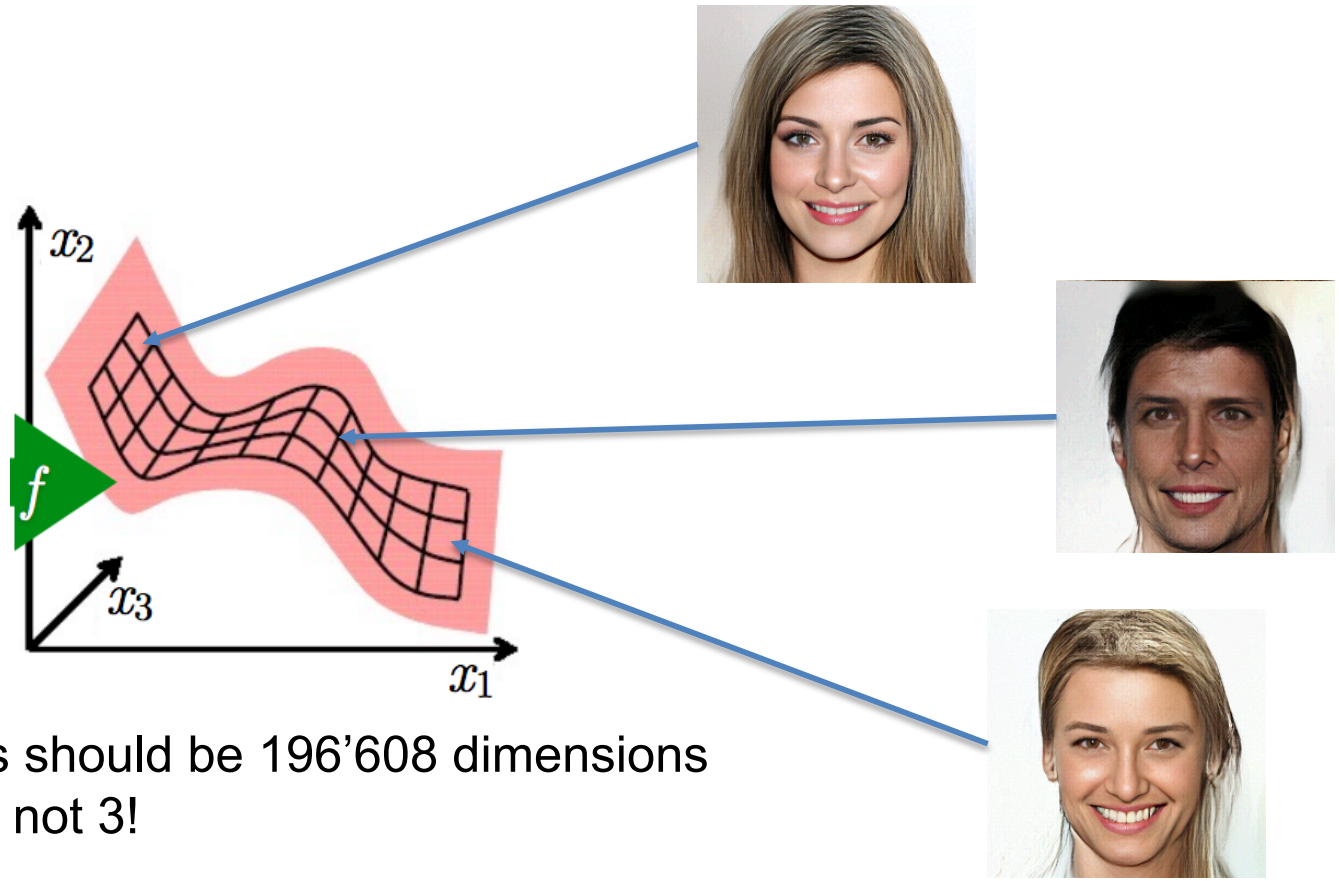
Reality: Data (2-D)



What distribution can you use?

Reality: Data (256x256x3=196'608 Dimensions)

3 data points **sampled** from the high dimensional distribution



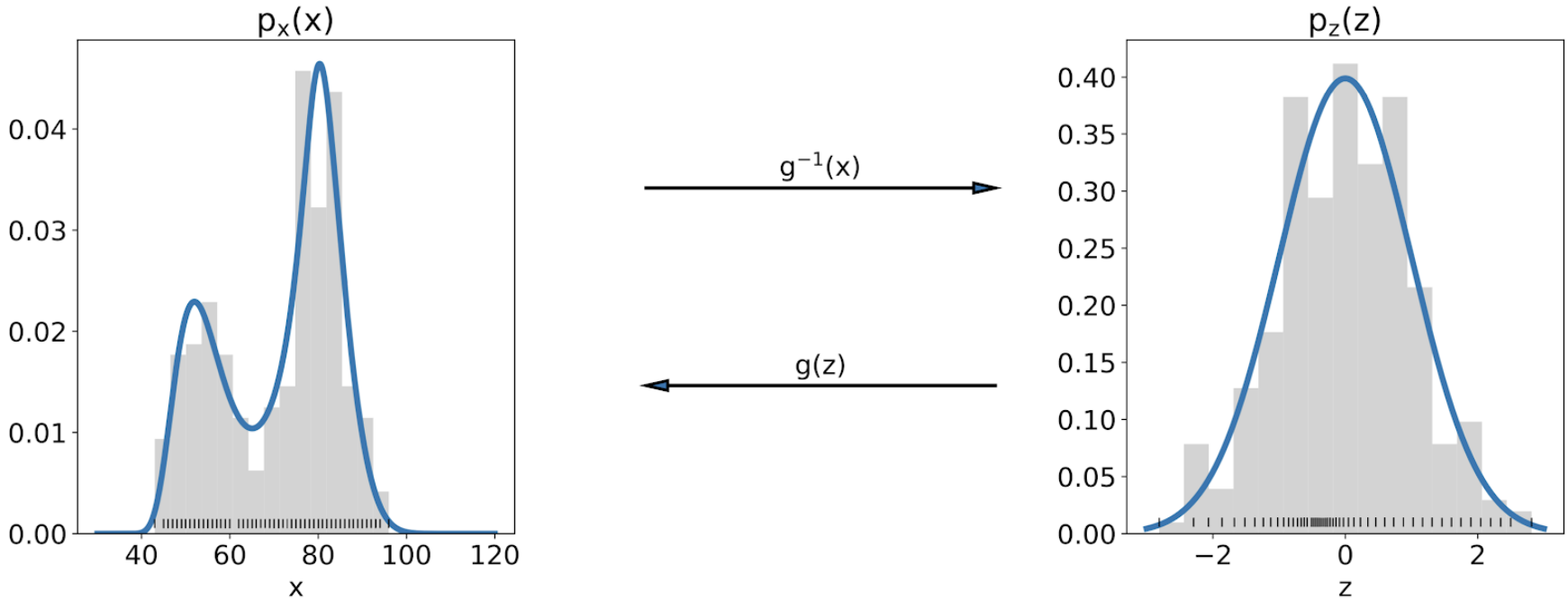
This should be 196'608 dimensions
and not 3!

What distribution can you use?

Main Idea of Normalizing Flows: 1D

Data $x \sim$strange_function in $\mathbb{R}^1$

Transformed function $z_1 \sim N(0,1)$



Idea: learn an *invertible* transformation to simple function usually Gaussian $N(0,1)$

- Sampling from $p_x(x)$: sample $z^* \sim p_z(z)$ then transform it via g_θ(z)
- Density of *x**: calculate z*= $g_\theta^{-1}(x^*)$ and evaluate $N(z^*; 0,1)$

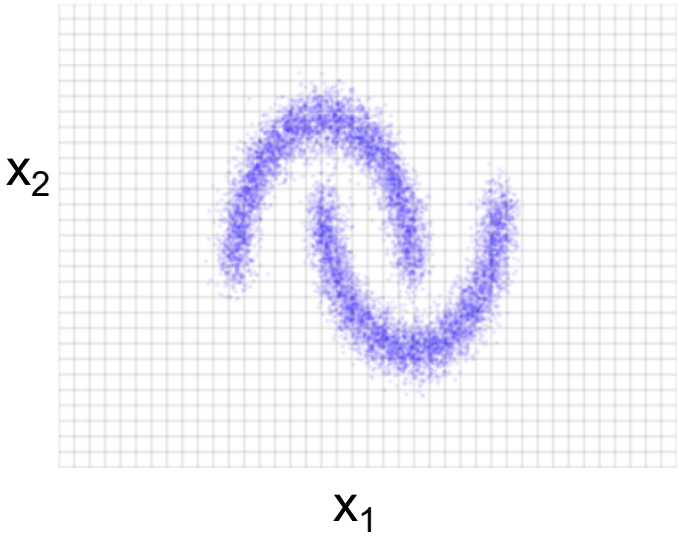
There is no fixed naming convention. Also in the direction of g(z) and $g^{-1}(x)$.

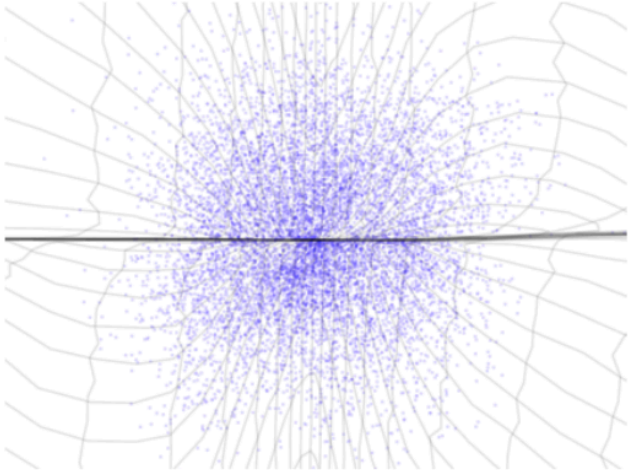
Main Idea of Normalizing Flows: 2D

Data $x \sim$strange_function in $\mathbb{R}^2$

Transformed Data $z_1, z_2 \sim N(0,1)$

g_θ^{-1}(x)





g_θ(z)

pdf $\mathrm{p_x}(x)$

pdf $\mathrm{p_z}(z)$

Idea: learn an *invertible* transformation to simple function usually Gaussian $N(0,1)$

- Sampling from $p_x(x)$: sample $z^* \sim p_z(z)$ then transform it via g_θ(z)
- Density of *x**: calculate z*= $g_\theta^{-1}(x^*)$ and evaluate $N(z^*; 0,1)$

Image Credit: RealNVP

Main Idea of Normalizing Flows: Large D

Data $x \sim$strange_function in $\mathbb{R}^{196608}$

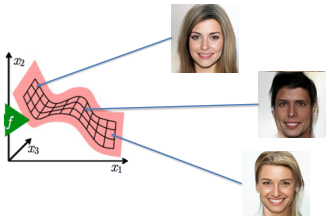
g_θ^{-1}(x)

$x_1, x_2, x_{196608} \sim$strange_function

$z_1, z_2, \ldots, z_{196608} \sim N(0,1)$

With many correlations

g_θ(z)



pdf p(x)

pdf $\mathrm{p_z}(z)$

Idea: learn an *invertible* transformation to simple function usually Gaussian $N(0,1)$

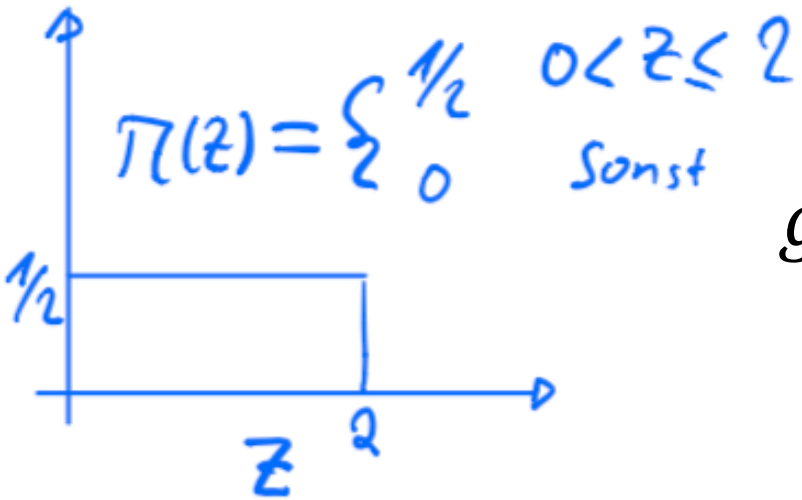
- Sampling from $p_x(x)$: sample $z^* \sim p_z(z)$ then transform it via g_θ(z)
- Density of *x**: calculate z*= $g_\theta^{-1}(x^*)$ and evaluate $N(z^*; 0,1)$

Transformation of Variables
-- Some math

Simple Transformation

- Say you have z~$Uniform(0,2)$
- $g(z) = z^2$

```
N = 10000
d = tfd.Uniform(low=0, high=2)
zs = d.sample(N)
x = zs**2
```



$$g(z) = z^2$$

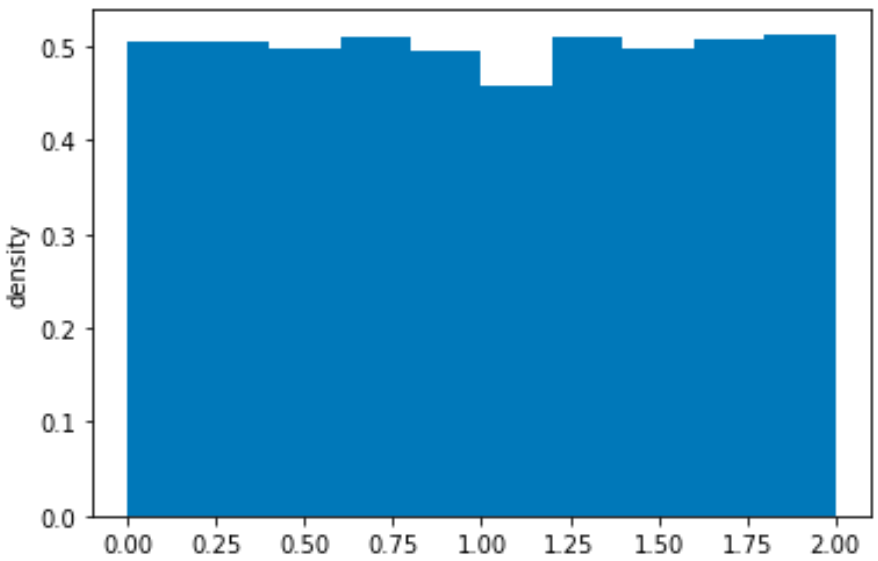
Try to come up with an answer, how is z distributed?

Try it

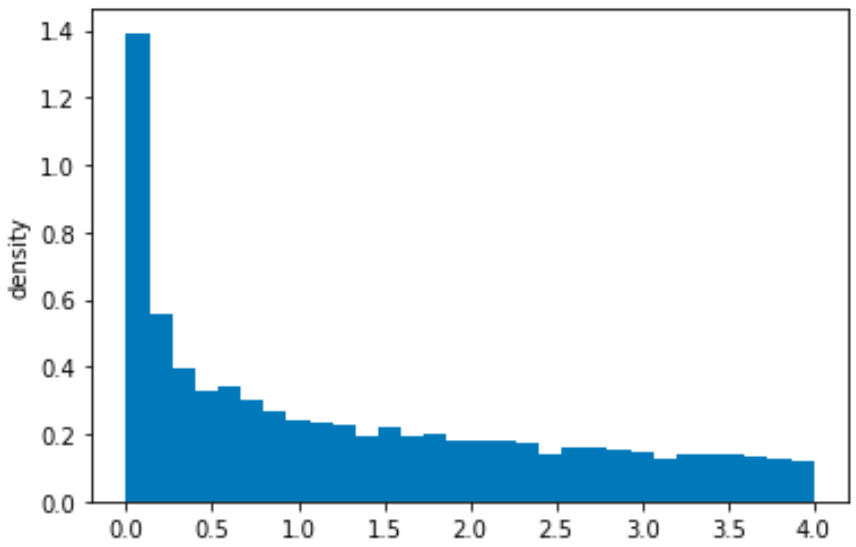
```
N = 10000
d = tfd.Uniform(low=0, high=2)
zs = d.sample(N)
```

```
x = zs**2
```

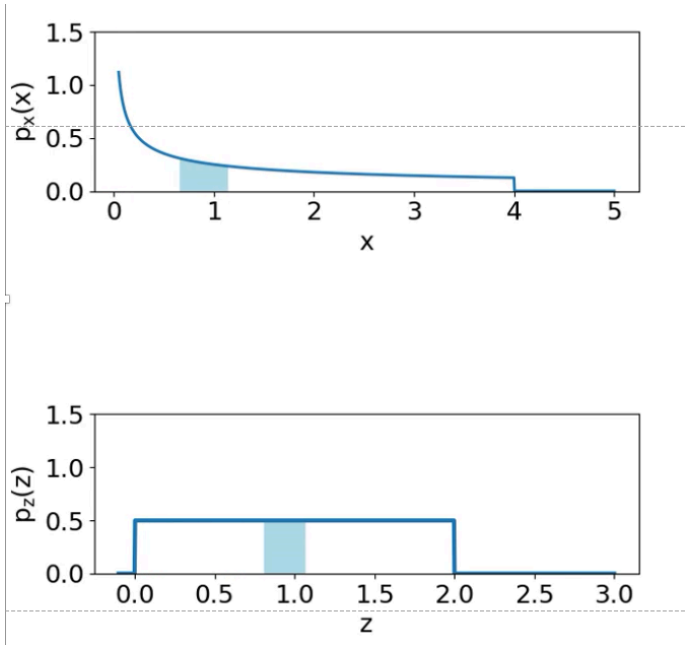
hist zs



hist zs**2



1-D



$$\Rightarrow P_x(x) = P_z(z)\left|\frac{dz}{dx}\right|$$

mit $z = x^2$ und $P_z(z) = 1/2$

$$P_x(x) = \frac{1}{2}\left|\frac{d\sqrt{z}}{dz}\right|$$

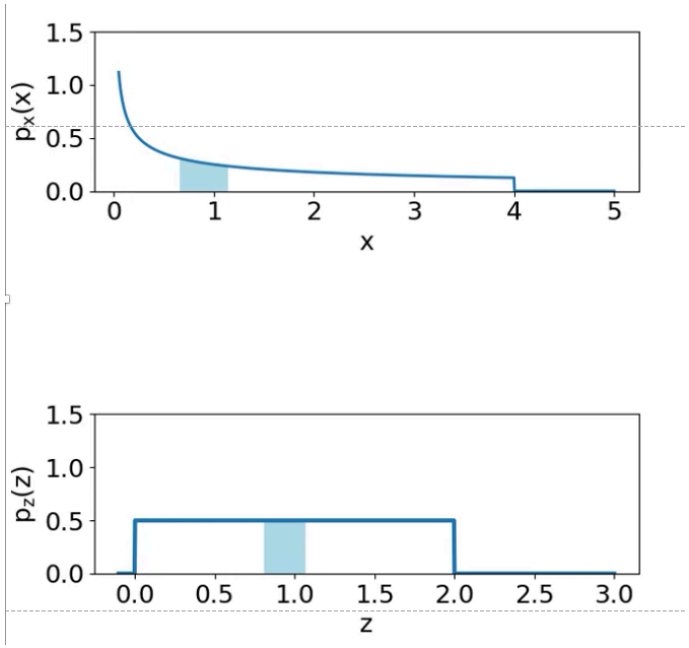
$$= \frac{1}{2}\left|\frac{1}{2}\frac{1}{\sqrt{z}}\right|$$

$$= \frac{1}{4}\frac{1}{\sqrt{z}}$$

For small value dz and dx :

$$p_z(z) \cdot |dz| = p_x(x) \cdot |dx|$$

1-D



For small value dz and dx :

$$p_z(z) \cdot |dz| = p_x(x) \cdot |dx|$$

$$p_x(x) = p_z(z) \cdot \left|\frac{dz}{dx}\right| = p_z(z) \cdot \left|\frac{dx}{dz}\right|^{-1} = p_z(z) \cdot \left|\frac{dg(z)}{dz}\right|^{-1} = p_z(g^{-1}(x)) \cdot |g'(g^{-1}(x))|^{-1}$$

$$p_x(x) = p_z(z) \cdot |g'(z)|^{-1} = p_z(g^{-1}(x)) \cdot |g'(g^{-1}(x))|^{-1}$$

Learning to flow

- How probable (well density) is a data point x_i (use $z_i = g^{-1}(x_i)$)

$$p_x(x_i) = p_z(z_i) \cdot |g'(z_i)|^{-1}$$

- All Data points

$$\prod_{i=1}^{n} p_x(x_i)$$

Tune the parameter(s) θ of the model M so that (observed) data is most likely!

- Simple example affine linear

$$g_{(a,b)}(z) = a \cdot z + b$$

- https://github.com/tensorchiefs/dl_book/blob/master/chapter_06/nb_ch06_03.ipynb

Going to higher dimensions

Transformation in high dimensions

$$g(z) = \begin{pmatrix} g_1(z_1, z_2, z_3) \\ g_2(z_1, z_2, z_3) \\ g_3(z_1, z_2, z_3) \end{pmatrix}$$

$$\frac{\partial g(z)}{\partial z} = \begin{pmatrix} \frac{\partial g_1(z_1, z_2, z_3)}{\partial z_1} & \frac{\partial g_1(z_1, z_2, z_3)}{\partial z_2} & \frac{\partial g_1(z_1, z_2, z_3)}{\partial z_3} \\ \frac{\partial g_2(z_1, z_2, z_3)}{\partial z_1} & \frac{\partial g_2(z_1, z_2, z_3)}{\partial z_2} & \frac{\partial g_2(z_1, z_2, z_3)}{\partial z_3} \\ \frac{\partial g_3(z_1, z_2, z_3)}{\partial z_1} & \frac{\partial g_3(z_1, z_2, z_3)}{\partial z_2} & \frac{\partial g_3(z_1, z_2, z_3)}{\partial z_3} \end{pmatrix}$$

$$p_x(x) = p_z(z) \cdot \left| det\left(\frac{dg\ (z)}{dz} \right) \right|^{-1}$$

Requirements for the Bijectors

A flow is composed of serval possible different *g's,* the bijectors in TFP language. The following restrictions apply for them

- *g* needs for be invertible (strict requirement)

- Training
  - Fast calculation of $g^{-1}(x)$
  - Fast calculation of Jacobi-Determinant

- Application (generation z→x)
  - Fast calculation of $\mathrm{g}(z)$

Flows with networks

Requirement / Design considerations for networks

- Fast calculation of $g(z)$, $g^{-1}(x)$
- Crucial: We need fast calculation of Jacobi Matrix
  - $\left|\det\left(\frac{\partial g_i(z)}{\partial z_j}\right)\right|^{-1}$

$$\begin{pmatrix} \frac{\partial g_1(z)}{\partial z_1} & \frac{\partial g_1(z)}{\partial z_2} & \frac{\partial g_1(z)}{\partial z_3} \\ \frac{\partial g_2(z)}{\partial z_1} & \frac{\partial g_2(z)}{\partial z_2} & \frac{\partial g_2(z)}{\partial z_3} \\ \frac{\partial g_3(z)}{\partial z_1} & \frac{\partial g_3(z)}{\partial z_2} & \frac{\partial g_3(z)}{\partial z_3} \end{pmatrix}$$

- Lin. Alg.: The determinant of triangular matrix is product of diagonal terms
  - Want triangular matrix $\frac{\partial g_1(z)}{\partial z_2} \overset{!}{=} 0$
    - ➔ $g_1(z) = f_1(z_1, \cancel{z_2, z_3})$, $g_d(z) = g_1(z_1, \dots, z_d, \cancel{z_{d+1}, z_{d+2,\dots}})$
  - Diagonal terms $\frac{\partial g_2(z)}{\partial z_2}$ easy to be calculated (no network!)

- $\frac{\partial g_2(z)}{\partial z_1}$ no restrictions, can be as complicated as hell (neural network)

Simple Solution: RealNVP

$$x_1 = g_1(z_1) = z_1$$
$$x_2 = g_2(z_2) = z_2$$

$$x_3 = g_3(z_1, z_2, z_3) = b_3(z_1, z_2) + \exp(\alpha_3(z_1, z_2)) \cdot z_3$$

$$x_4 = g_4(z_1, z_2, z_4) = b_4(z_1, z_2) + \exp(\alpha_4(z_1, z_2)) \cdot z_4$$

$$x_5 = g_5(z_1, z_2, z_5) = b_5(z_1, z_2) + \exp(\alpha_5(z_1, z_2)) \cdot z_5$$

j →

i ↓

$$\begin{array}{cc} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ e & e & \exp(\alpha_3(z_1, z_2)) & 0 & 0 \\ e & e & e & \exp(\alpha_4(z_1, z_2)) & 0 \\ e & e & e & e & \exp(\alpha_5(z_1, z_2)) \end{pmatrix} & \begin{array}{l} x_1 = z_1 \\ x_2 = z_2 \\ b_3(z_1, z_2) + \exp(\alpha_3(z_1, z_2)) \cdot z_3 \\ b_4(z_1, z_2) + \exp(\alpha_4(z_1, z_2)) \cdot z_4 \\ b_5(z_1, z_2) + \exp(\alpha_5(z_1, z_2)) \cdot z_5 \end{array} \\ \begin{array}{ccccc} \frac{\partial .}{\partial z_1} & \frac{\partial .}{\partial z_2} & \frac{\partial .}{\partial z_3} & \frac{\partial .}{\partial z_4} & \frac{\partial .}{\partial z_5} \end{array} & \end{array}$$

e=don't care

Glow for image data

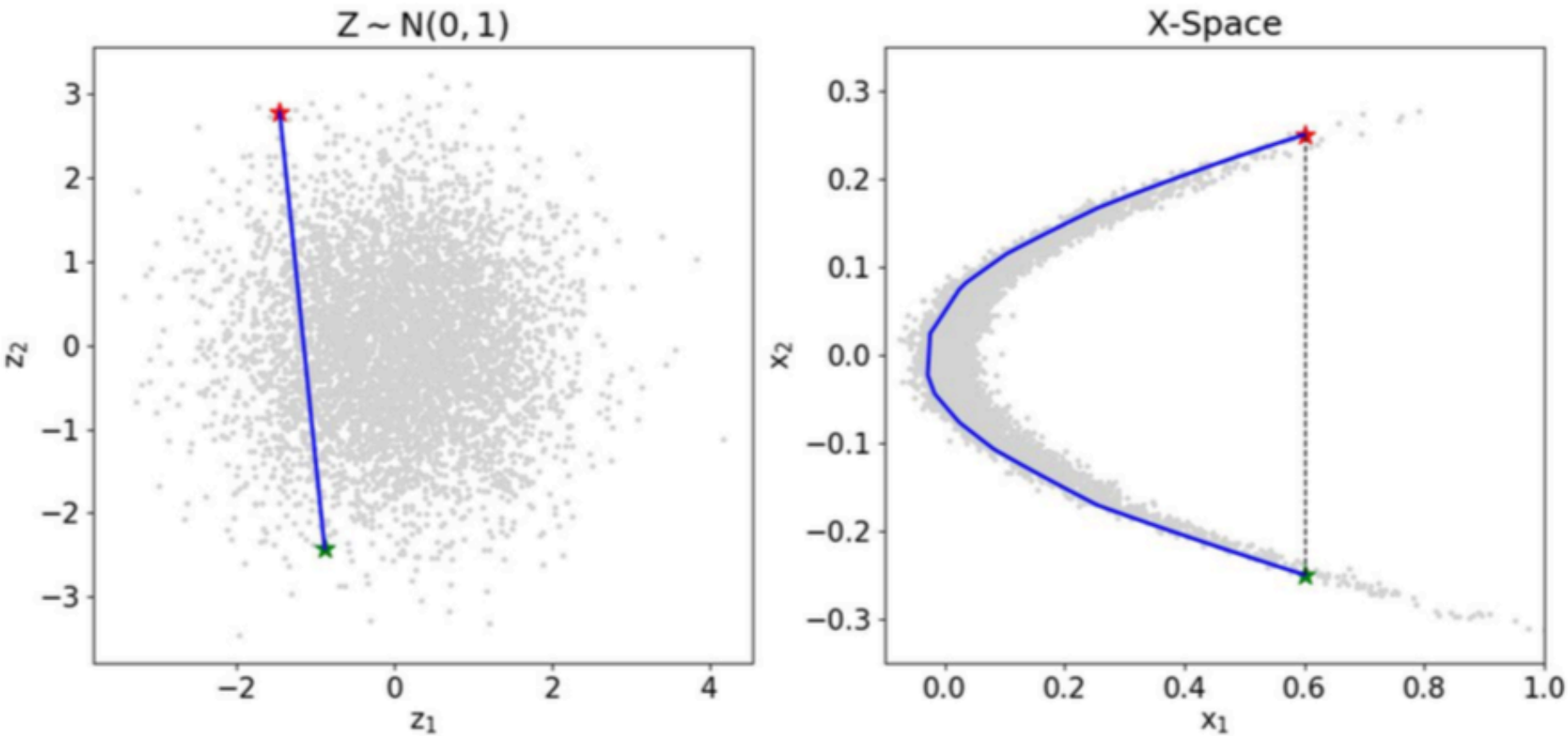
--arXiv:1807.03039

Glow: Generative Flow with Invertible 1×1 Convolutions

Diederik P. Kingma*, **Prafulla Dhariwal***
OpenAI, San Francisco

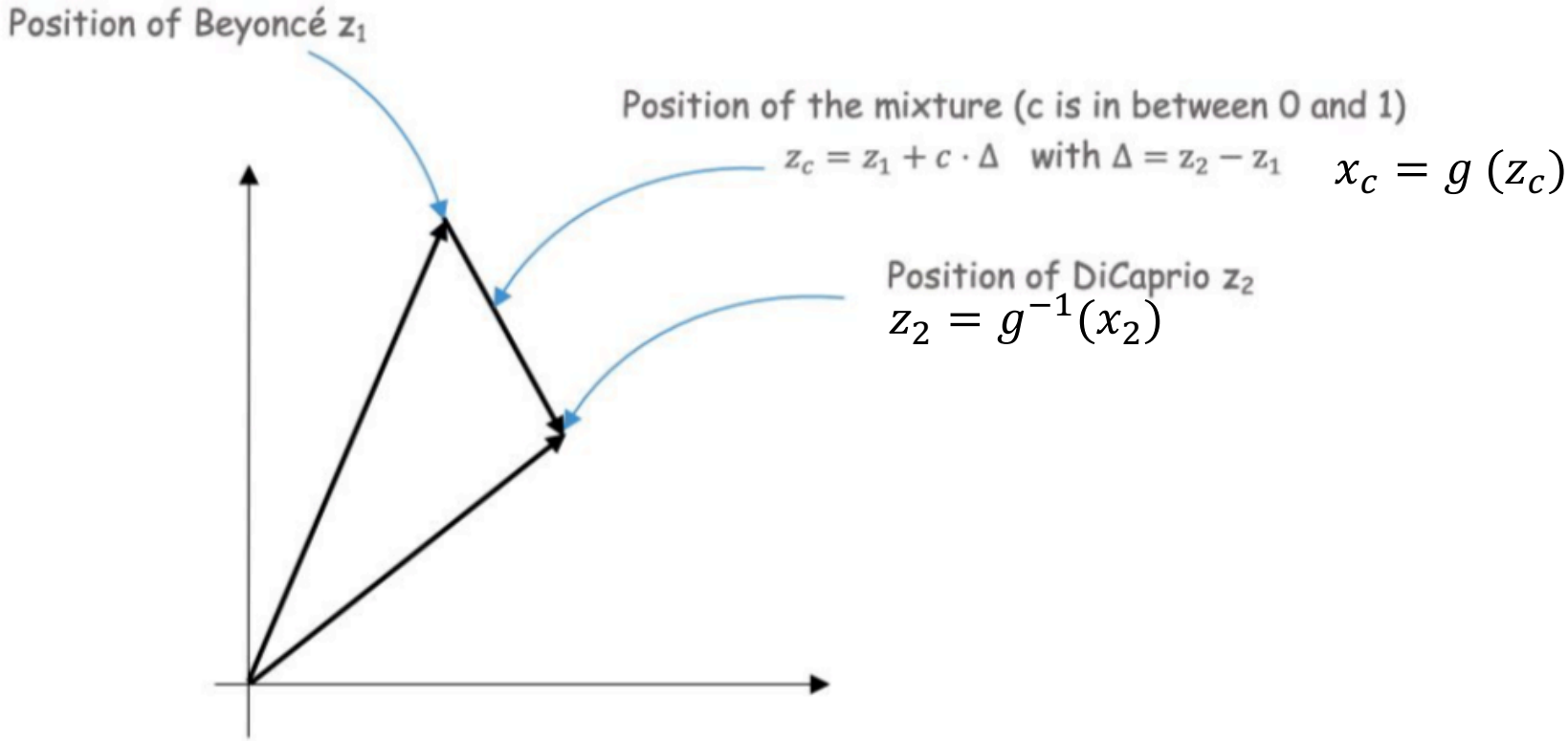
https://www.youtube.com/watch?v=JTtW_nhjIYA

Morphing in 2-D



Morphing in 256x256x3 D

$z_1 = g^{-1}(x_1)$



$x_c = g\,(z_c)$

$z_2 = g^{-1}(x_2)$

Do it yourself

- Network has been trained on CelebA-HQ
  - 30000 (256x256x3) images of celebrities
  - Images have been aligned
- Sampling: draw 256*256*3 numbers from N(0,1)
  - Reduced Temperature draw from N(0,T*1)
- Demo
  - Uses pretrained network
  - **fun_with_glow**

https://github.com/tensorchiefs/dl_book/blob/master/chapter_06/nb_ch06_05.ipynb