Machine Learning
for LISA Data Analysis

Natalia Korsakova

Laboratoire
AstroParticule & Cosmologie

APC, CNRS, FRANCE

APPLICATIONS OF MACHINE LEARNING in DA

- Approximation of non-linear functions
- Compression, embedding, alternative representation of the data
- Alternative ways to perform Bayesian inference
- Accelerating traditional approaches
- Approximation of the distributions
- …

INFERENCE

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)}$$

INFERENCE

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)}$$

- data model:

$$x = h(\theta) + n$$

« waveform template»

INFERENCE

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)}$$

- data model:

$$x = h(\theta) + n$$

« waveform template»

physical parameters

INFERENCE

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)}$$

- data model:

$$x = h(\theta) + n$$

« waveform template»

physical parameters

measurement noise

INFERENCE

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{\boxed{p(x)}}$$

problem:
marginal likelihood
has no exact solution

$$p(x) = \int p(x|\theta)p(\theta)\mathrm{d}\theta$$

INFERENCE

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{\boxed{p(x)}}$$

solutions:

- approximate inference:
  - MCMC/Nested sampling
    requires likelihood evaluation
    we can do it, but it is slow

INFERENCE

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)}$$

solutions:

- approximate inference:
  - MCMC/Nested sampling
    requires likelihood evaluation
    we can do it, but it is slow
  - Variational inference
    approximate the posterior distribution
    with a tractable distribution

INFERENCE

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)}$$

solutions:

- simplification to the model:
  - Gaussian mixture models

    too simple

INFERENCE

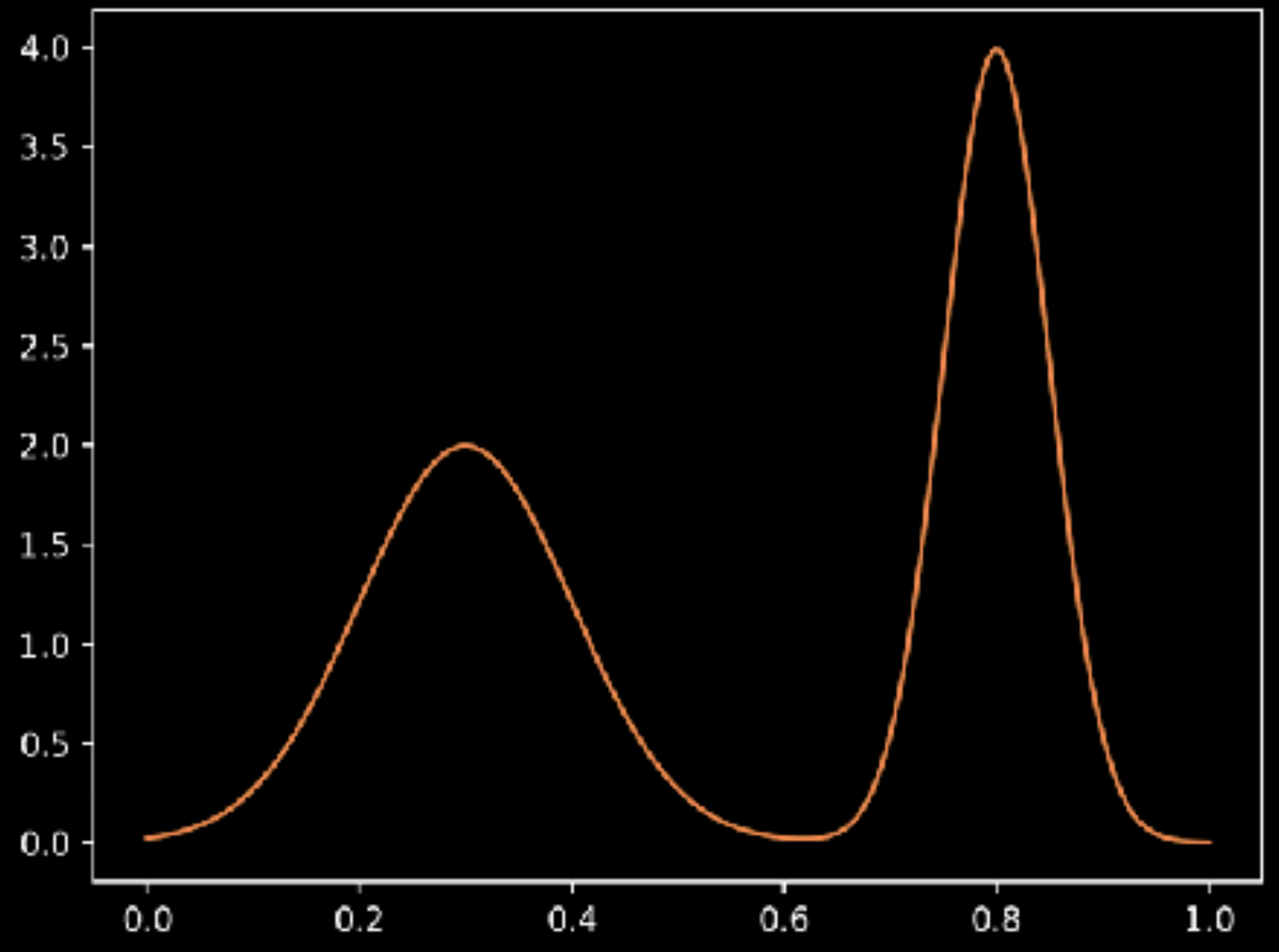
$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)}$$

solutions:

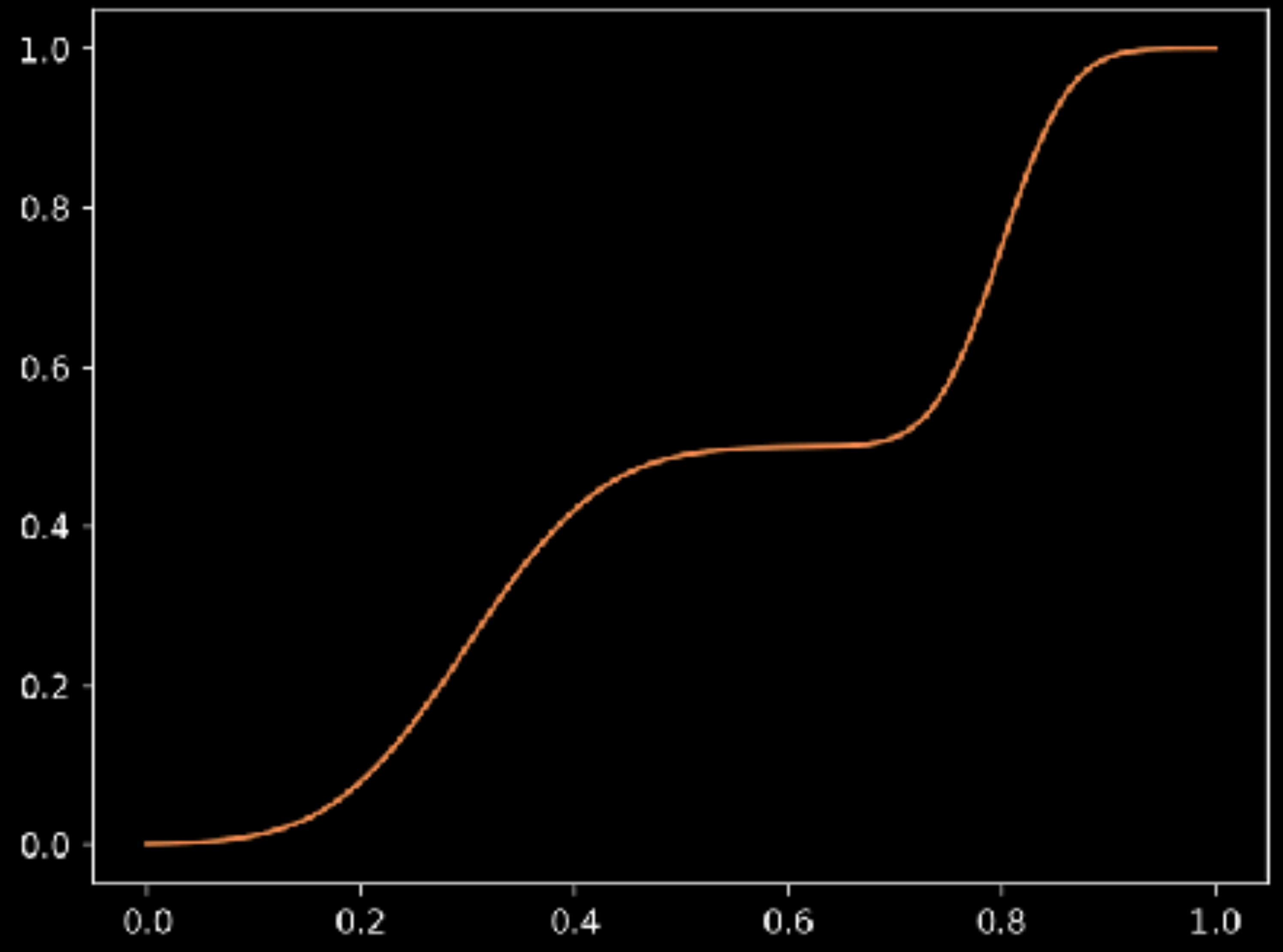
- simplification to the model:
  - Gaussian mixture models
    too simple
  - Invertible models
    will talk about them today

INVERTABLE TRANSFORM

If x is a random variable with the CDF f(x),
then the random variable y = f(x) has a uniform distribution on [0,1].



$$p_\theta(x)$$



$$f_\theta(x) = \int_{-\infty}^{x} p_\theta(t)\, dt$$

INVERTABLE TRANSFORM

Change of variables for probability density function

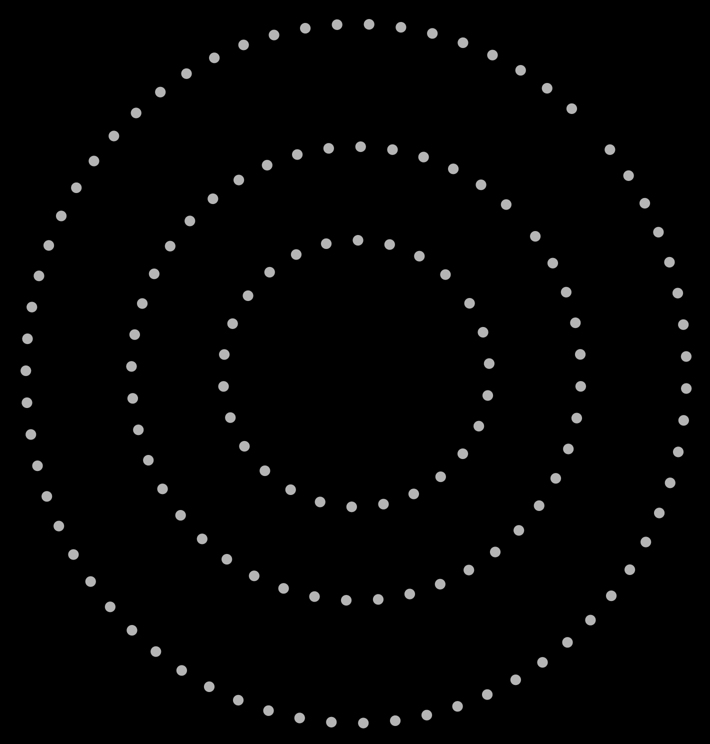
$$f_Y(y) = \frac{d}{dy}F_Y(y) = \frac{d}{dy}F_Z(g^{-1}(y))$$

Apply chain rule

$$= f_Z(g^{-1}(y))|\frac{d}{dy}g^{-1}(y)|$$

NORMALISING FLOWS

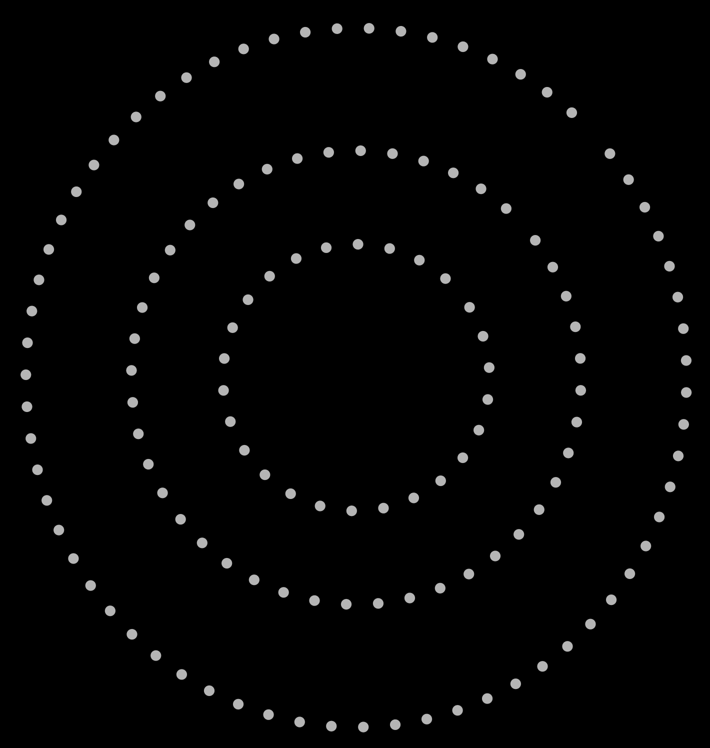
1. We have simple random generator



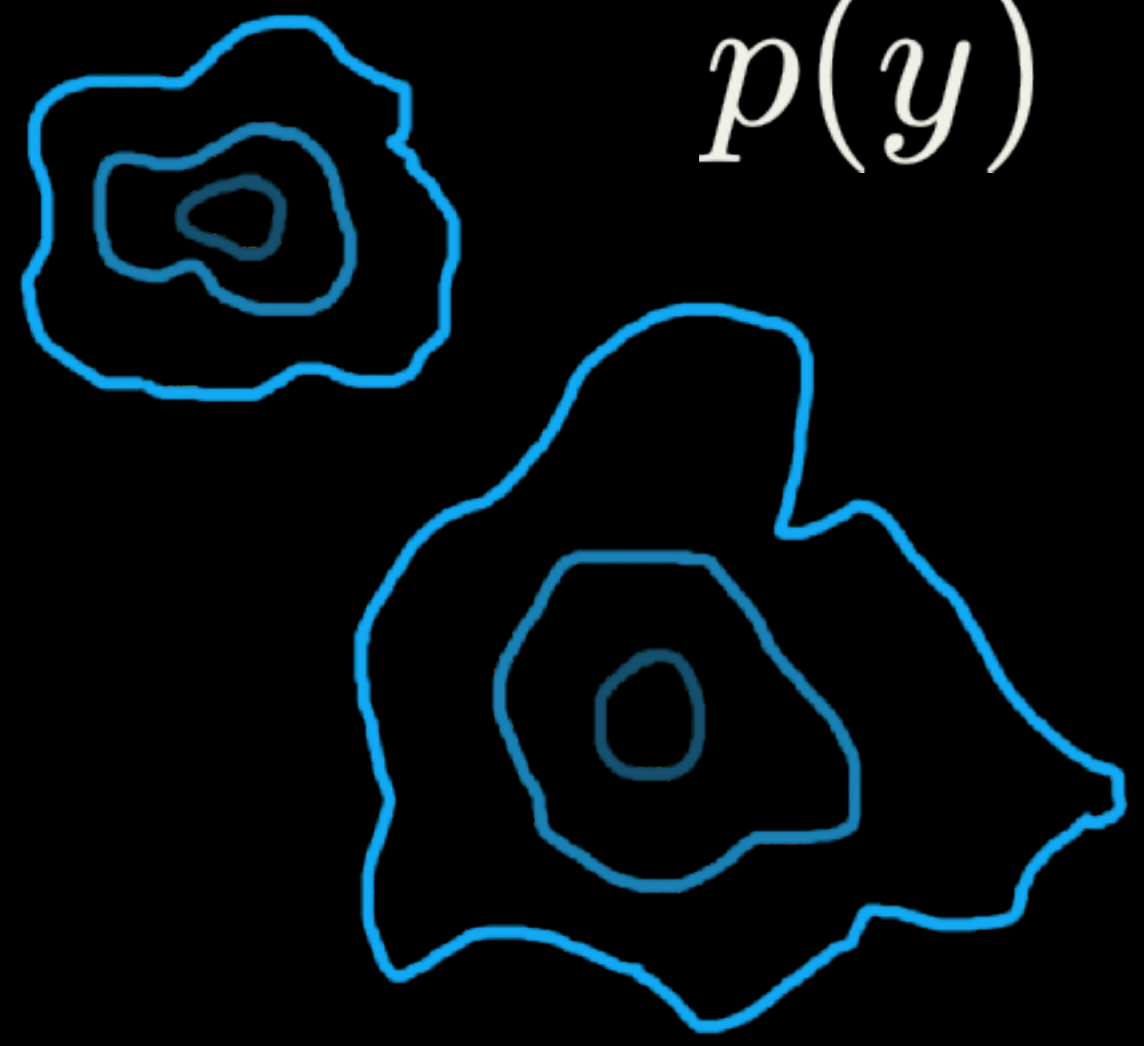
$$q(z) = \mathcal{N}(0, 1)$$

NORMALISING FLOWS

1. We have simple random generator
2. We want to sample from a more complex distribution



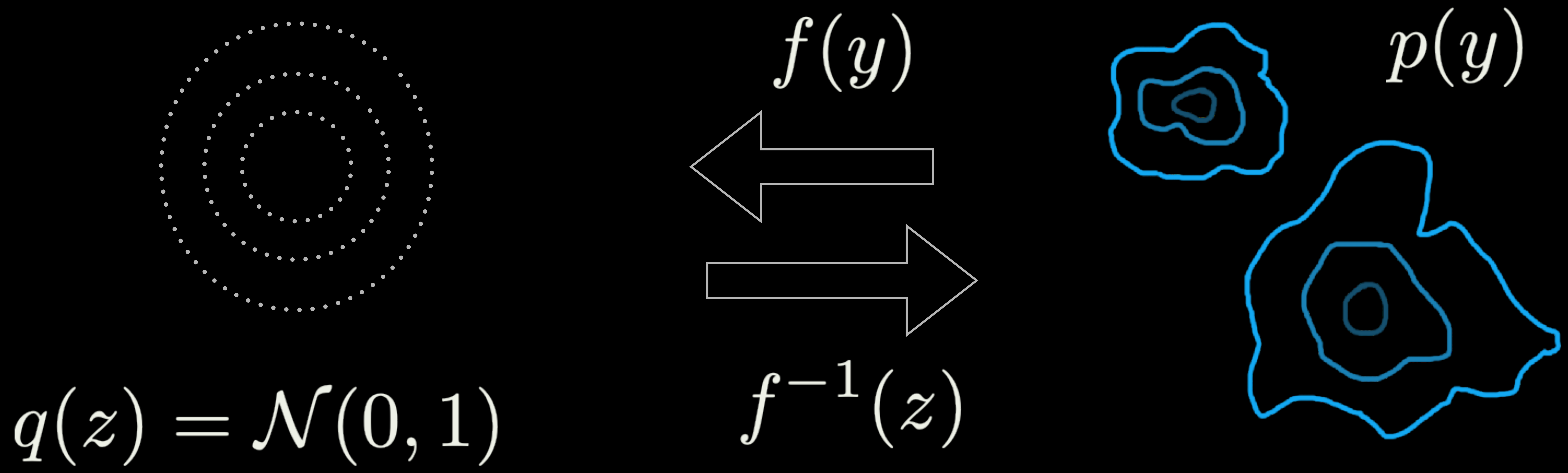
$$q(z) = \mathcal{N}(0,1)$$



$p(y)$

NORMALISING FLOWS

1. We have simple random generator
2. We want to sample from a more complex distribution
3. We can estimate a bijective transformation which will allow us to do that



CHANGE OF VARIABLE EQUATION

$$p(y) = q(f(y))|\det(J_f(y))|$$

CHANGE OF VARIABLE EQUATION

$$p(y) = \boxed{q(f(y))} |\det(J_f(y))|$$

- f has to be a bijection

CHANGE OF VARIABLE EQUATION

$$p(y) = q(f(y)) \left| \det(J_f(y)) \right|$$

- f has to be a bijection
- f and f^{-1} have to be differentiable
- Jacobian determinant has to be tractably invertable

JACOBIAN

- The calculation of determinant Jacobian will take $O(N^3)$
- We need to speed it up
- For example, make Jacobian triangular matrix

AFFINE TRANSFORM

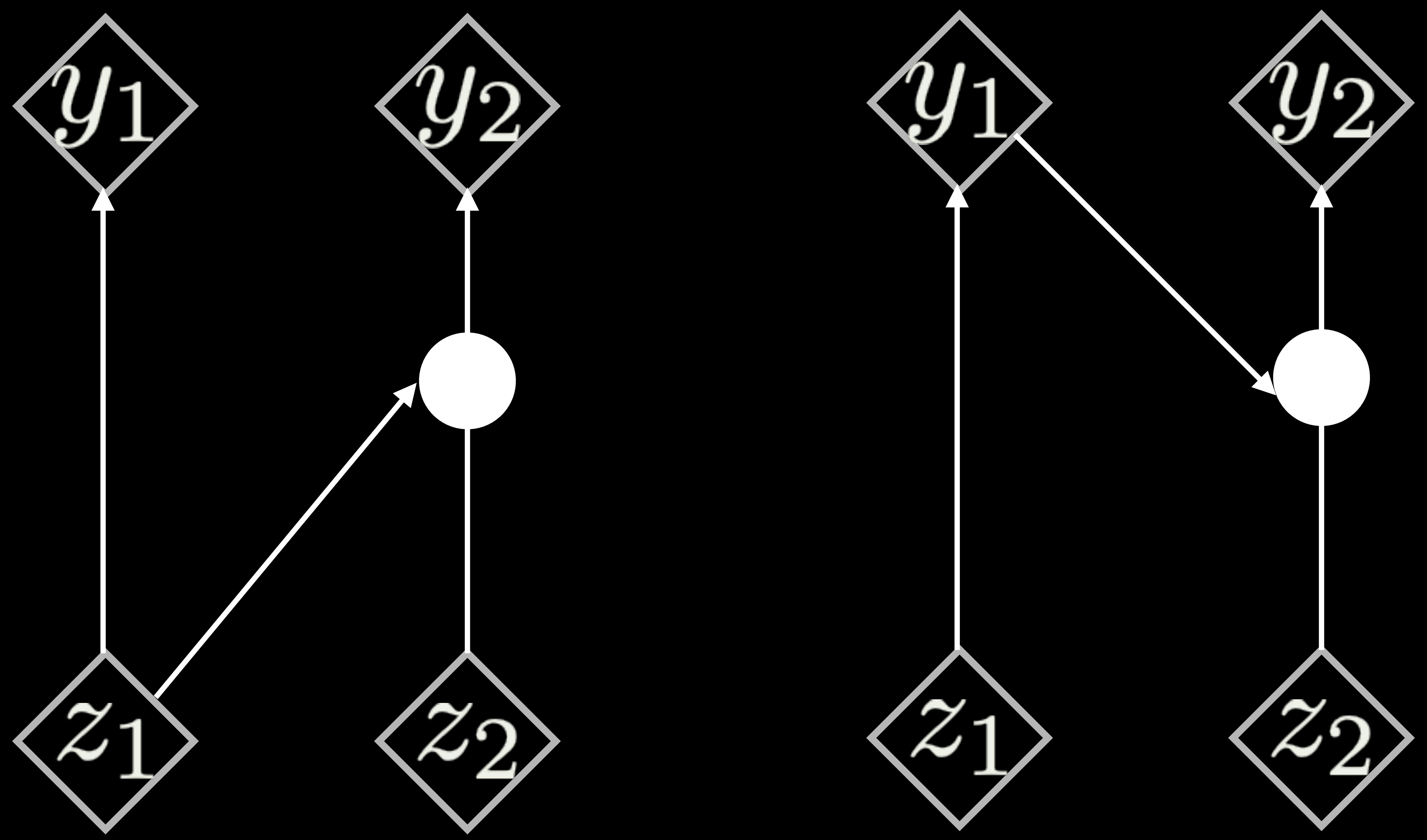
Location-scale transformation

$$\tau(z_i) = \alpha_i z_i + \beta_i$$

log-Jacobian becomes

$$\log|\det J_{g^{-1}}(z)| = \sum \log|\alpha_i|$$

COUPLING TRANSFORM



In each simple bijection,
part of the input vector
is updated using a function
which is simple to invert,
but which depends on the
remainder of the input vector
in a complex way.
The other part is left unchanged.

REAL NVP

Coupling transformation combined with affine transformation and its invention

$$\begin{cases} y_{1:d} &= x_{1:d} \\ y_{d+1:D} &= x_{d+1:D} \odot \exp\big(s(x_{1:d})\big) + t(x_{1:d}) \end{cases}$$

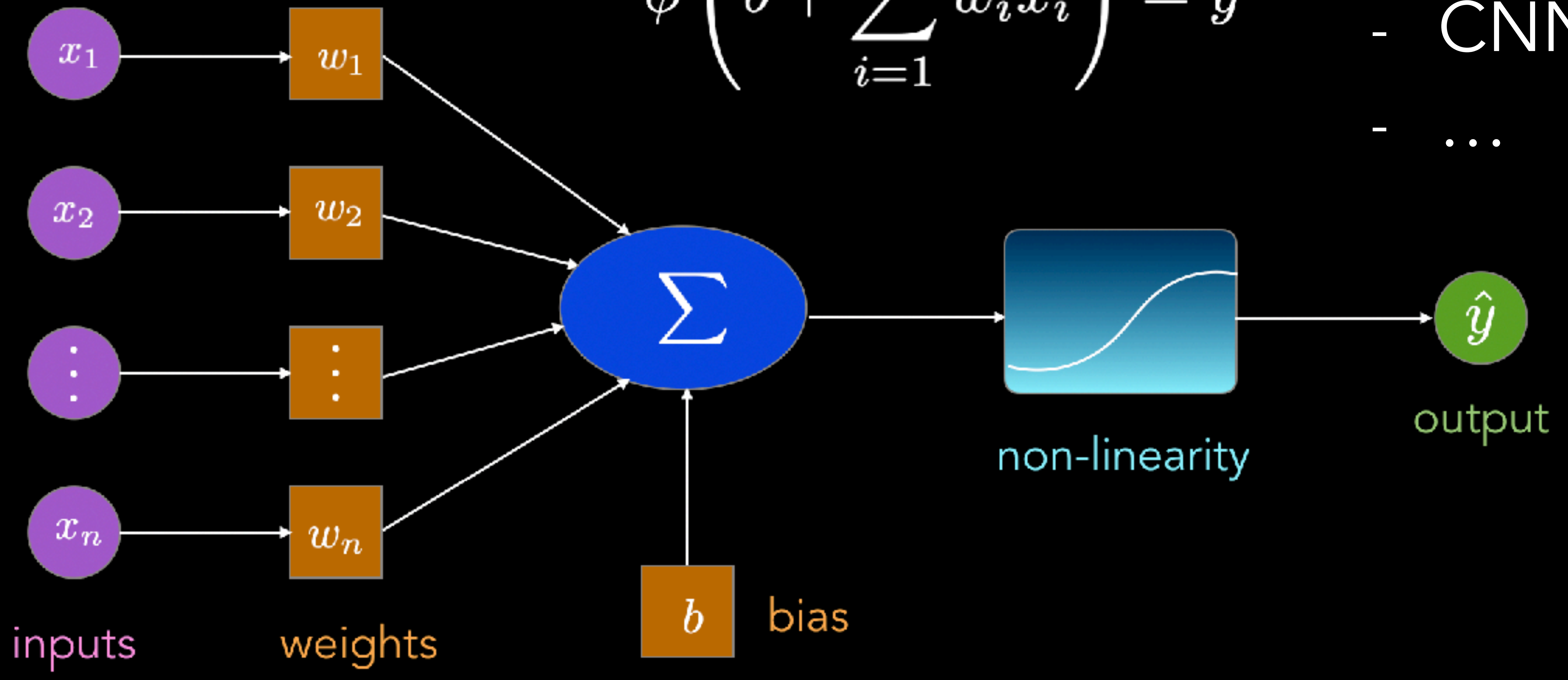
$$\Leftrightarrow \begin{cases} x_{1:d} &= y_{1:d} \\ x_{d+1:D} &= \big(y_{d+1:D} - t(y_{1:d})\big) \odot \exp\big(-s(y_{1:d})\big), \end{cases}$$

What is ***t*** and ***s***?

https://arxiv.org/abs/1605.08803

FUNCTION APPROXIMATION

$$\phi\left(b + \sum_{i=1}^{n} w_i x_i\right) = \hat{y}$$

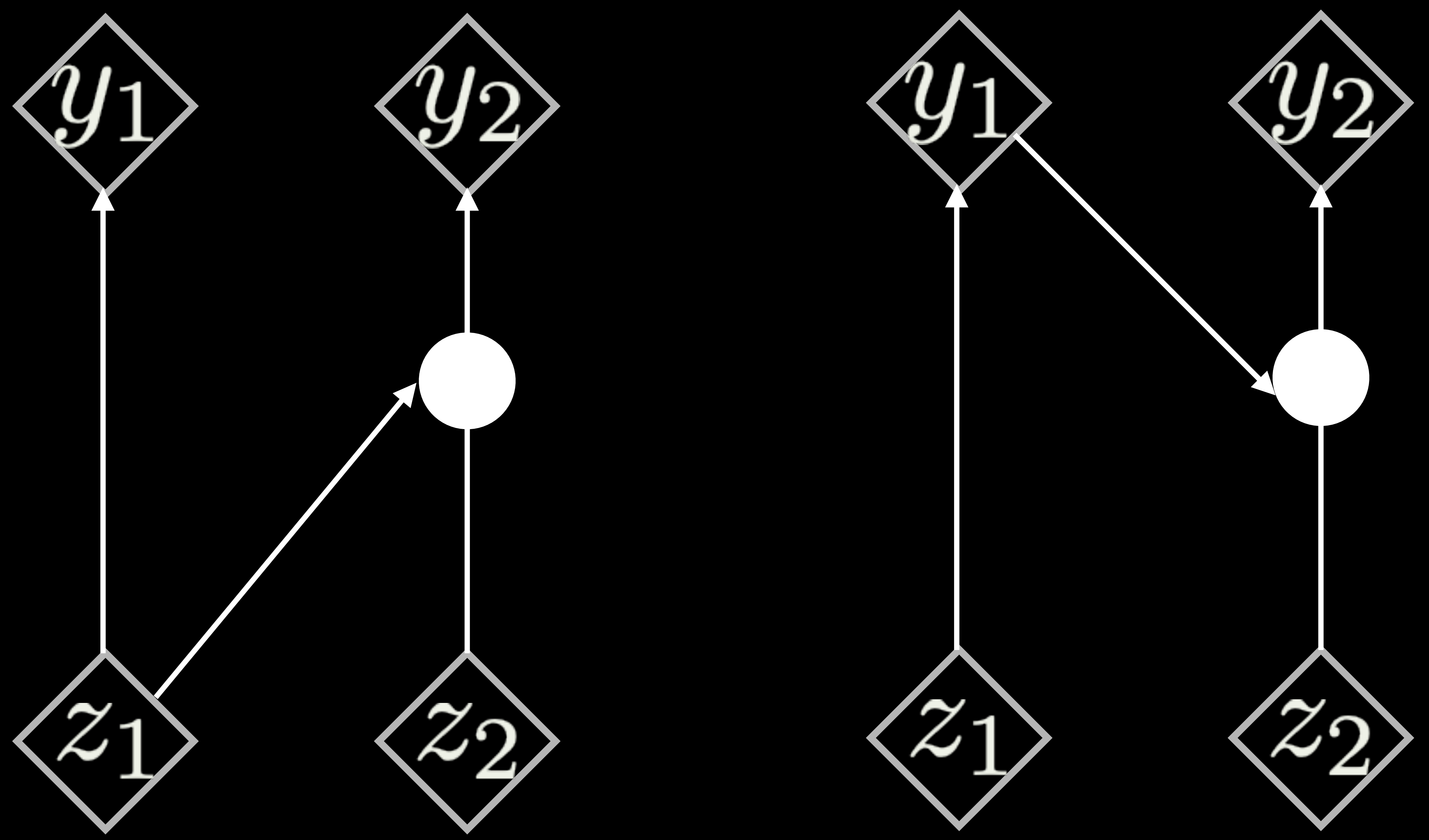


can be parameterised by any NN:

- Fully connected
- Residual
- CNN
- …

NEURAL SPLINE FLOWS

- Coupling transform



- Monotonic rational-quadratic spline transform

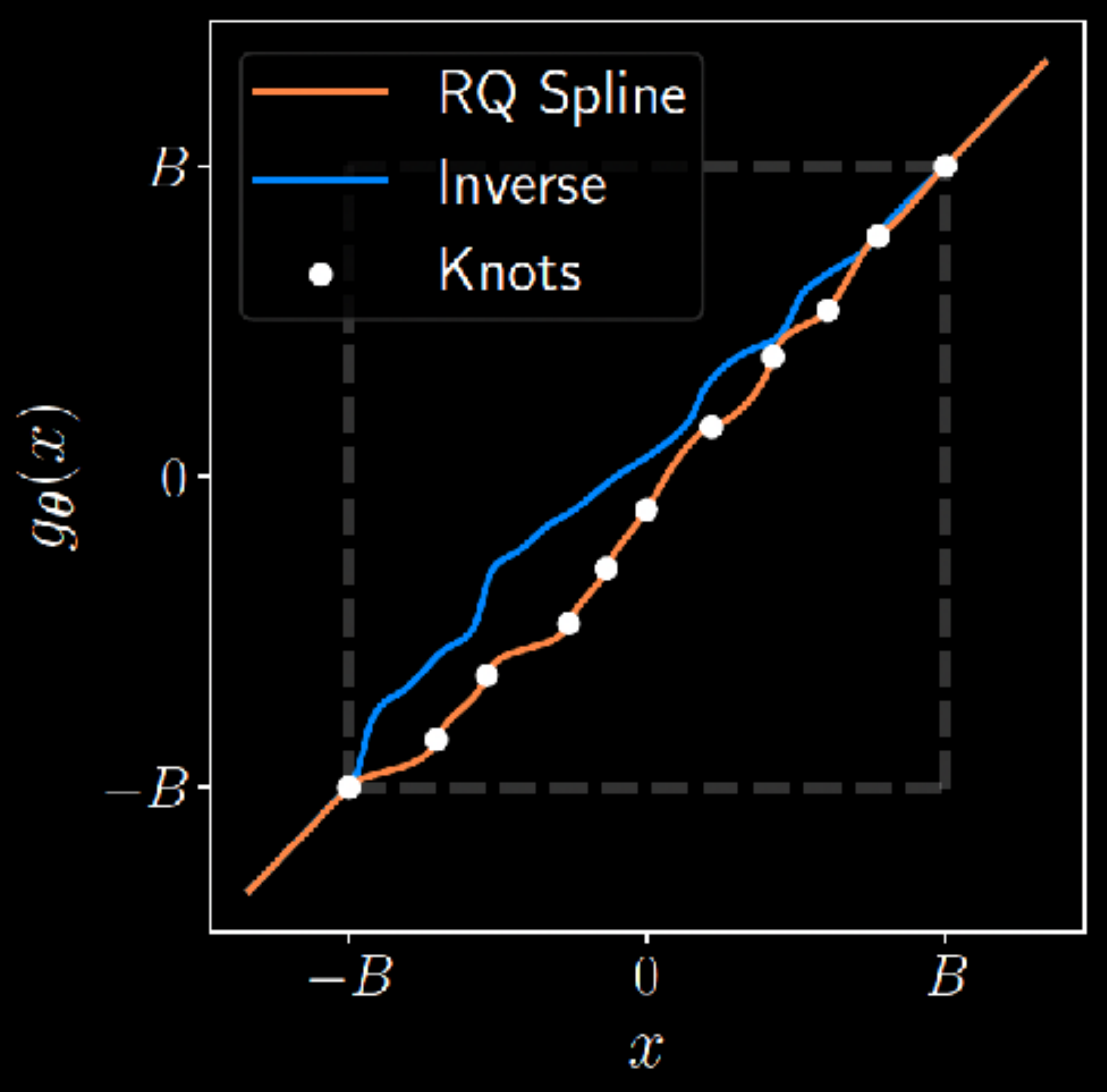
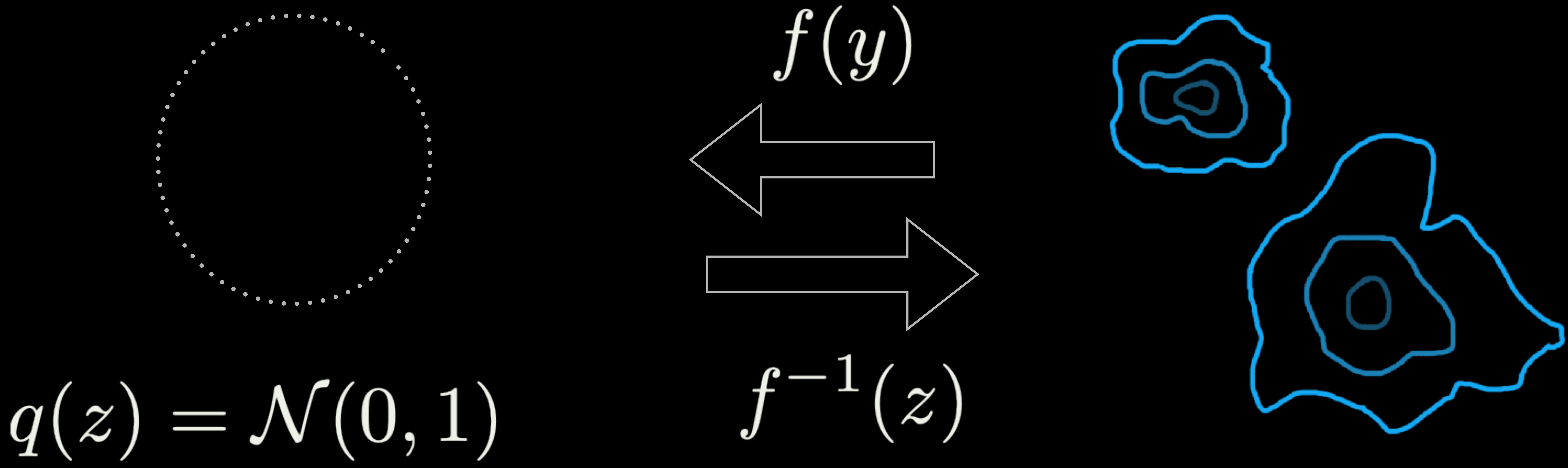


image: Duncan C. et al, Neural Spline Flows

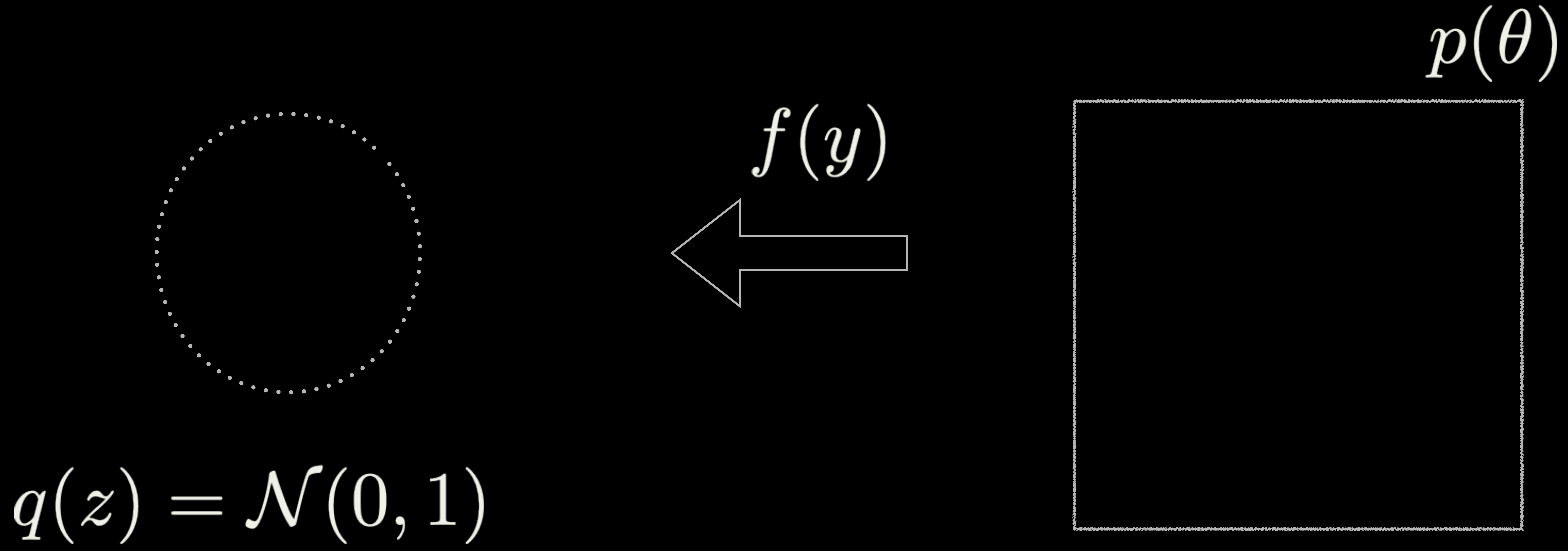
CONDITIONING

- Do not have access to samples from posterior



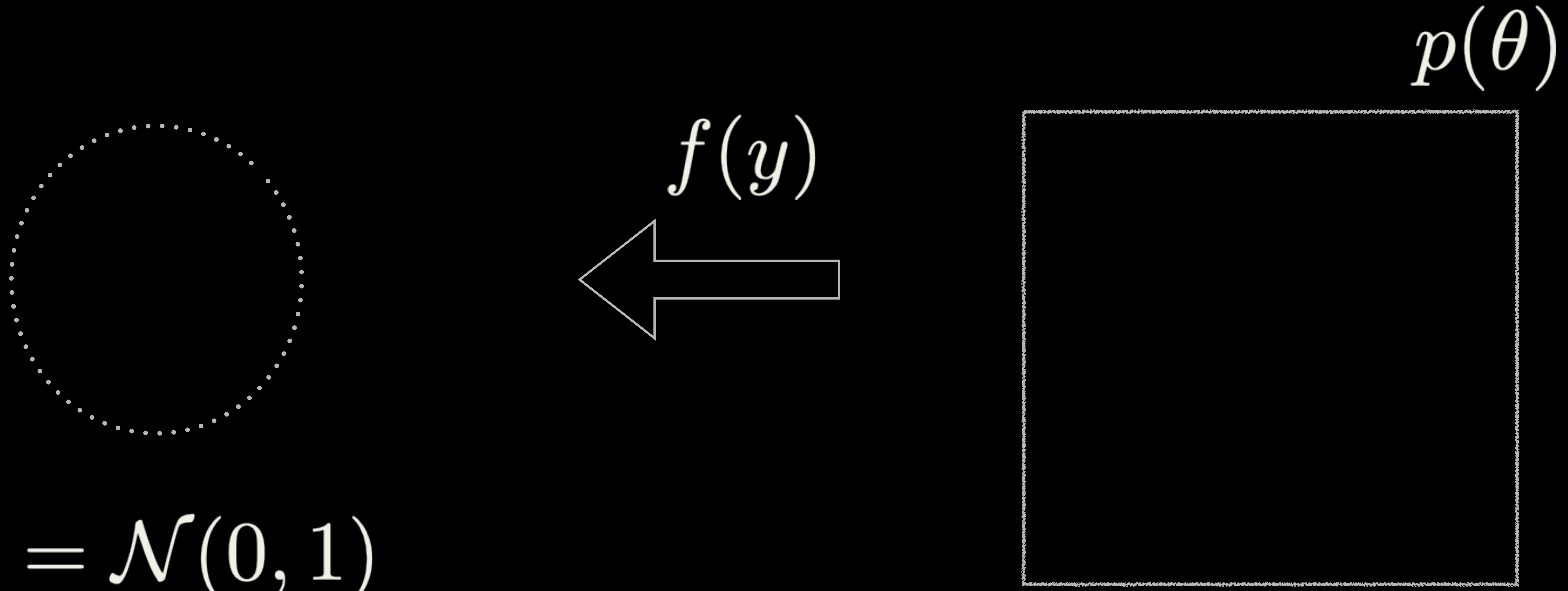
CONDITIONING

- Do not have access to samples from posterior
- Have access to samples from prior +



CONDITIONING

- Do not have access to samples from posterior
- Have access to samples from prior +
- Can generated simulated data $x = h(\theta) + n$

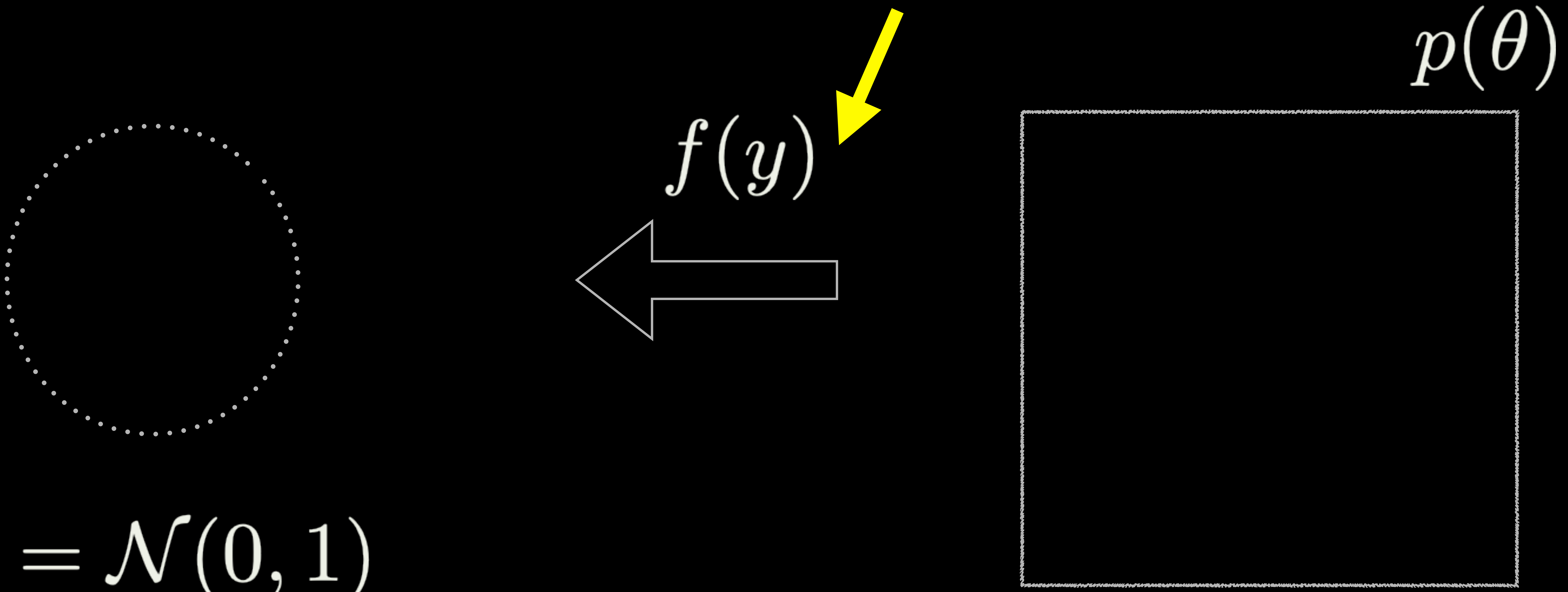


$$q(z) = \mathcal{N}(0,1)$$

CONDITIONING

Condition map
on simulated data

- Do not have access to samples from posterior
- Have access to samples from prior +
- Can generated simulated data $x = h(\theta) + n$



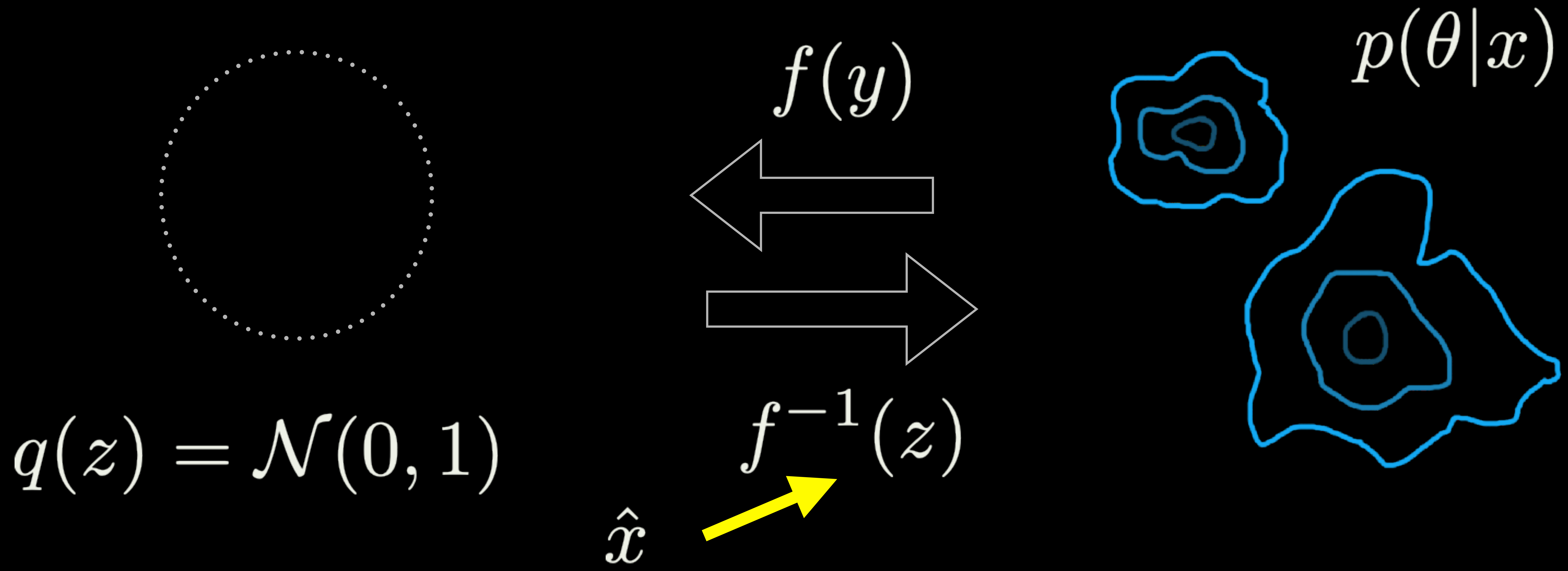
$q(z) = \mathcal{N}(0,1)$

Therefore have access to the joint sample $p(x, \theta) = p(x|\theta)p(\theta)$

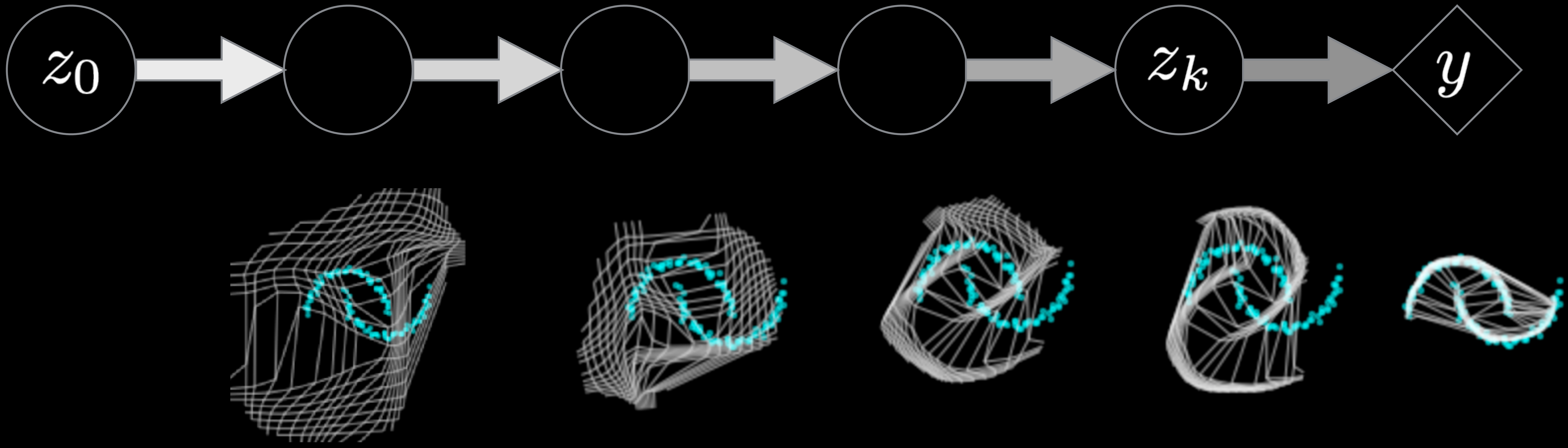
CONDITIONING

Condition inverted map on real data

- Do not have access to samples from posterior
- Have access to samples from prior +
- Can generated simulated data $x = h(\theta) + n$



COMPOSING FLOW



OPTIMISATION

- The flow is trained to maximise the total log likelihood of the data with respect to the parameters of the transform.

$$\log p(y|\lambda) = \sum_{i=1}^{N} \log\left[p(y\prime_i|\lambda)\right])$$

WAVEFORM EMBEDDING

- Low frequency sensitivity -> long waveforms
- Construct reduced orthogonal basis
- Use coefficients of the waveform projection on a new basis

WAVEFORM EMBEDDING

Decompose a matrix constructed of the set of waveforms

$$\mathbf{H} = \mathbf{V}\mathbf{\Sigma}\mathbf{U}^{\mathbf{T}}$$

WAVEFORM EMBEDDING

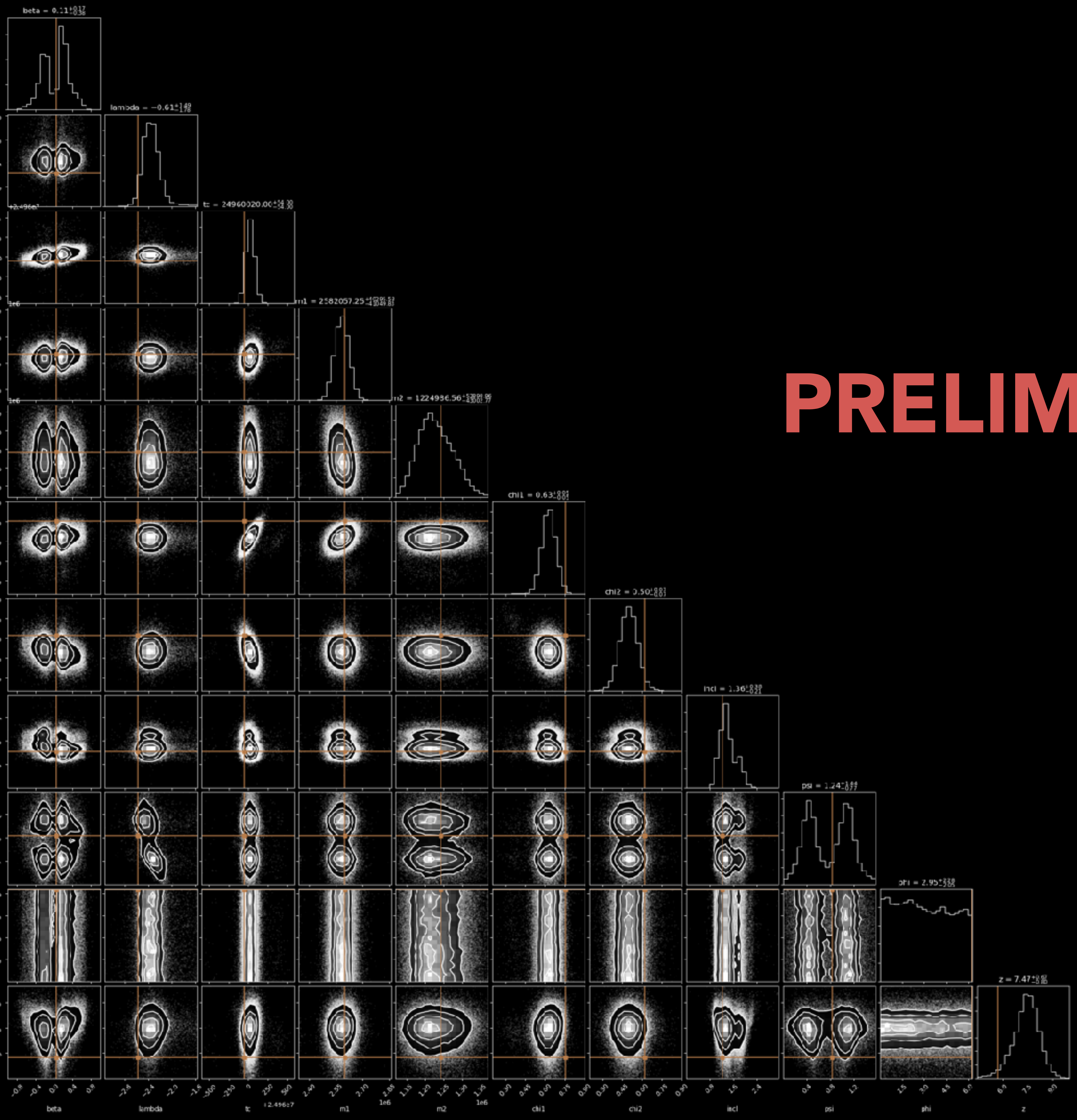
Decompose a matrix constructed of the set of waveforms

$$\mathbf{H} = \mathbf{V}\mathbf{\Sigma}\mathbf{U}^{\mathrm{T}}$$

Project sample simulated data on this basis

$$v'_{\alpha\mu} = \frac{1}{\sigma_\mu}\sum_{j=1}^{N} h_{\alpha j} u_{\mu j}$$

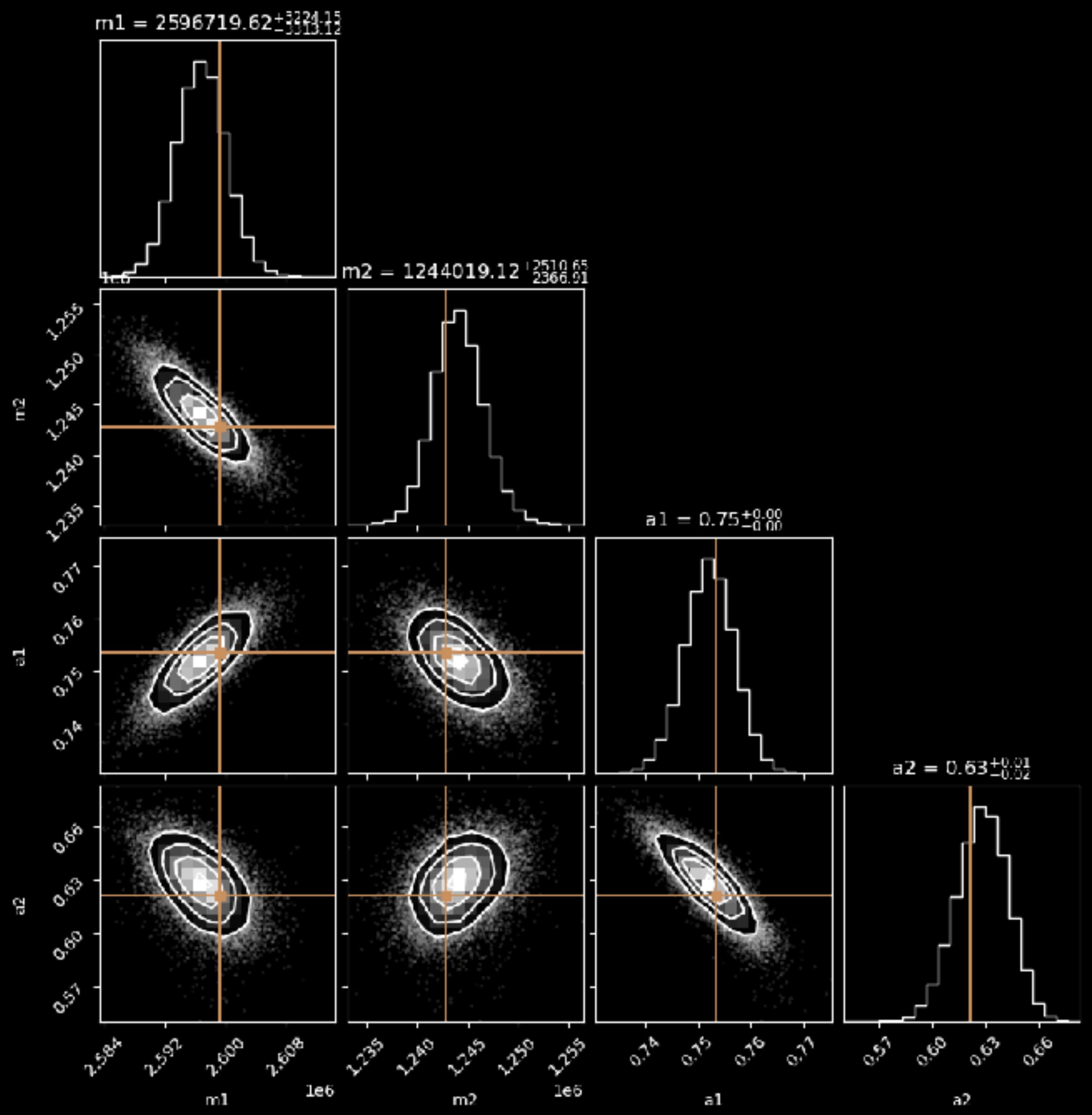
RESULTS

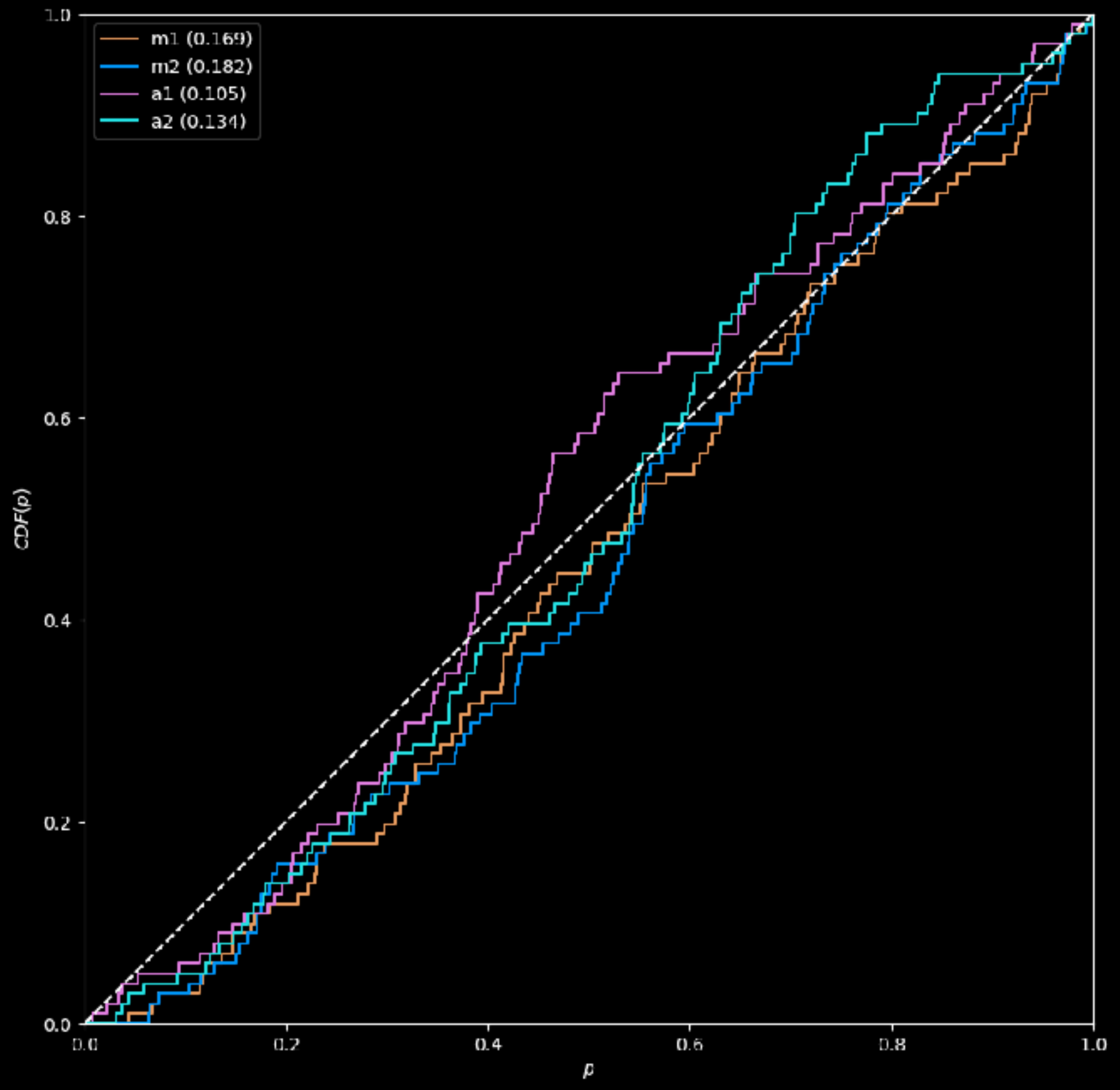


PRELIMINARY

RESULTS

PRELIMINARY





CONCLUSIONS

- Alternative sampling method
- Can be used for low latency pipeline
- Can be used to approximate complex distributions

LATENT VARIABLE AND SOURCE SEPARATION

LATENT VARIABLE AND SOURCE SEPARATION

- We will observe tens of thousands GBs
- 10 to 100 MBHBs per year
- 1 to 10000 EMRIs per year

- Have to find a way to analyse them together or disentangle

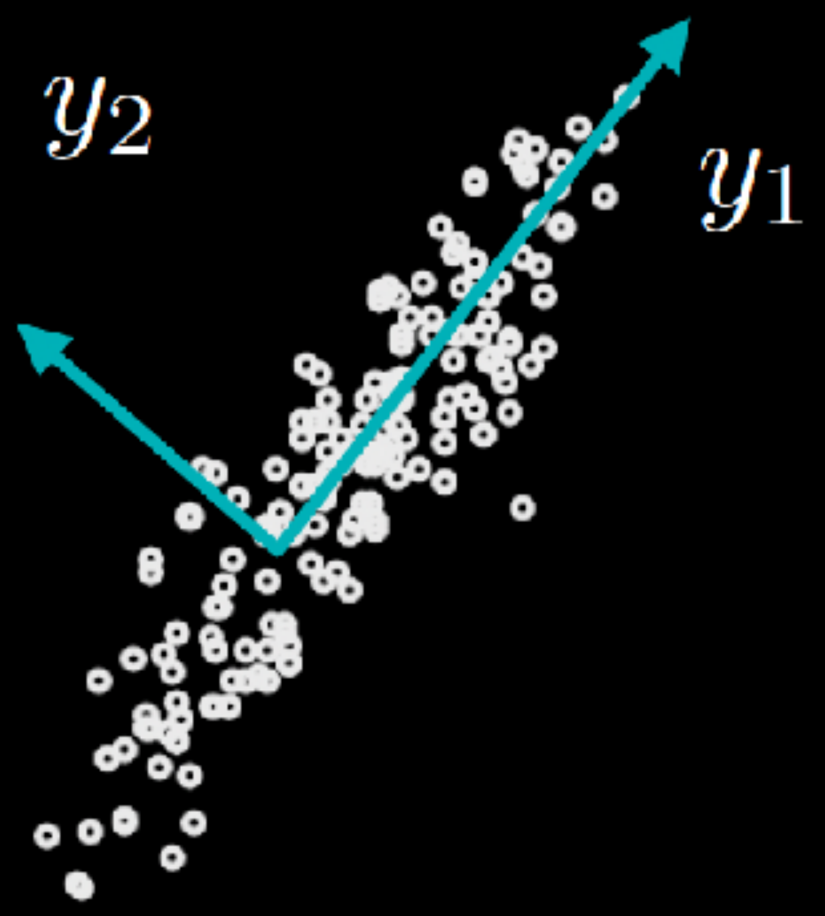
COCKTAIL PARTY PROBLEM

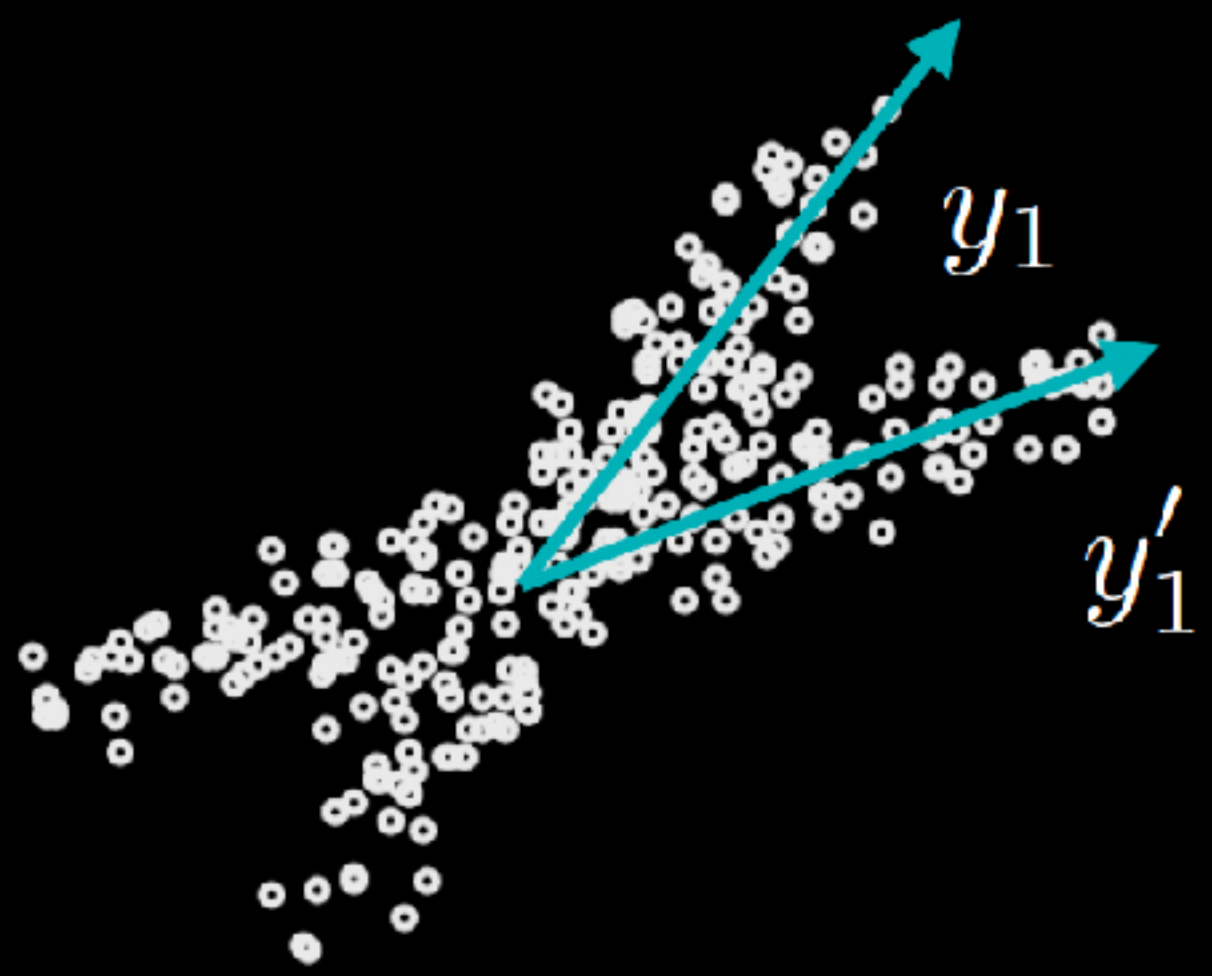
$$\begin{cases} x_1(t) = a_{11}s_1(t) + a_{12}s_2(t) \\ x_2(t) = a_{21}s_1(t) + a_{22}s_2(t) \end{cases}$$

$$x(t) = \mathbf{D}(\hat{n}, f) : \mathbf{h}(f, \xi)$$

Traditional way to solve this problem was to find independent components in the data.

INDEPENDENT COMPONENT ANALYSIS

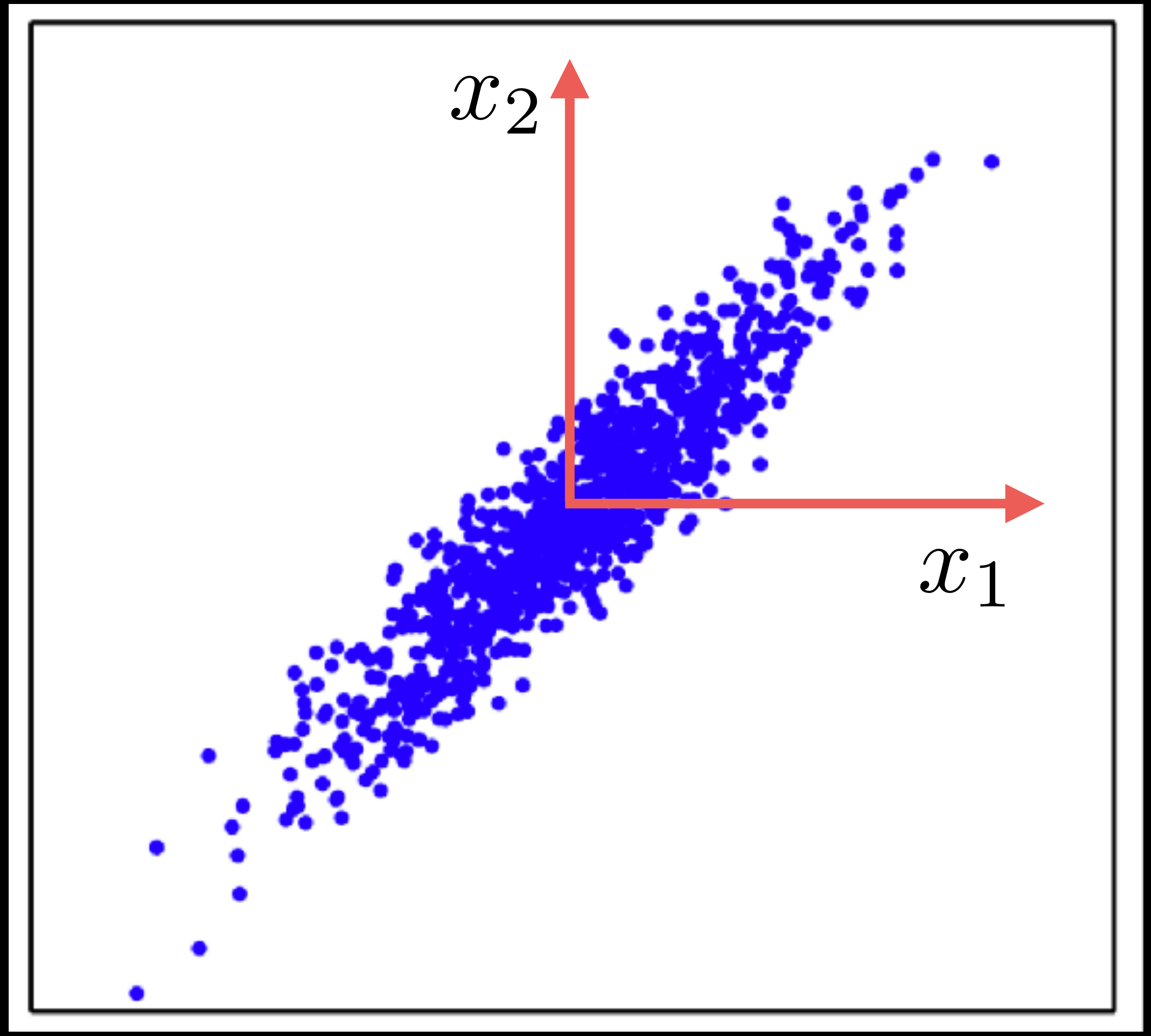




Traditional way — find independent components by maximising non-Gaussianity.

This is a linear problem.

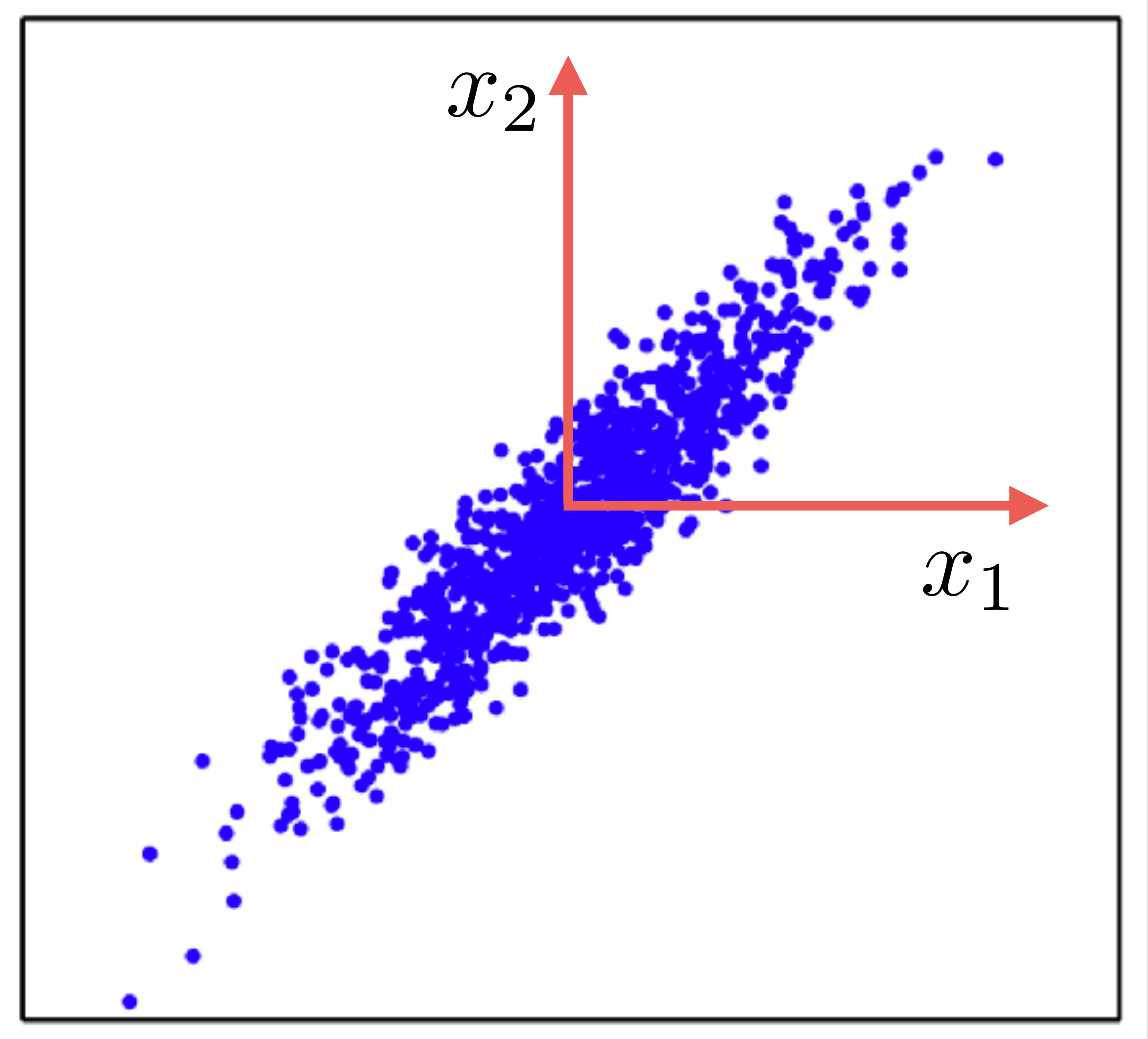
PRINCIPLE COMPONENT ANALYSIS



PCA maps original data into a new coordinate system which maximises variance of the data

$$y_1 = \sum_{k=1}^{n} w_{k1} x_k$$

PRINCIPLE COMPONENT ANALYSIS



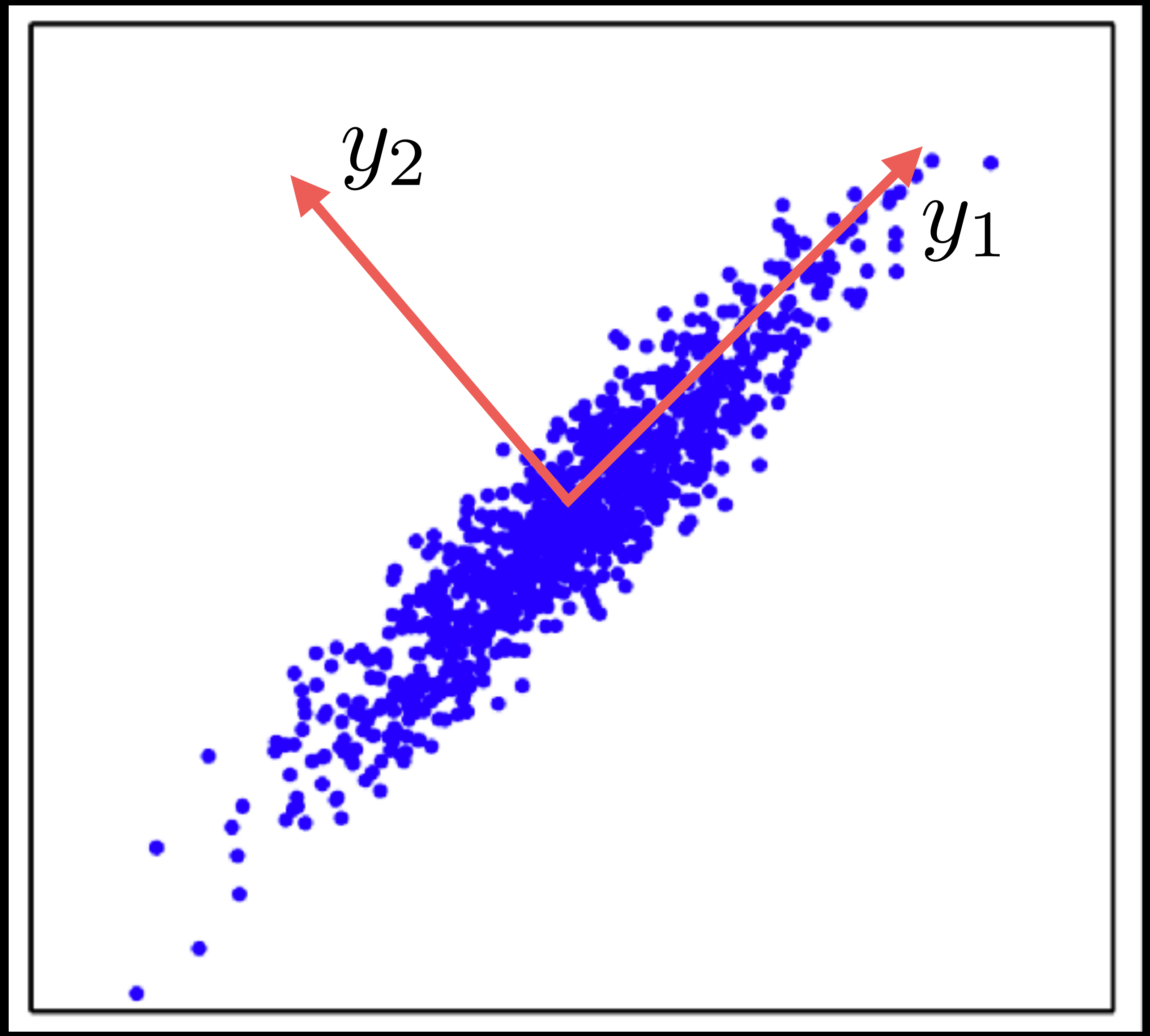
The mapping to the new basis can be expressed using the eigenvectors of the Covariance matrix

$$C = E\{\mathbf{x}\mathbf{x}^T\}$$

Eigenvalue decomposition

$$\mathbf{C} = \mathbf{U}\mathbf{D}\mathbf{U}^T$$

PRINCIPLE COMPONENT ANALYSIS



The vector of principle components

$$\mathbf{y} = \mathbf{U}^T \mathbf{x}$$

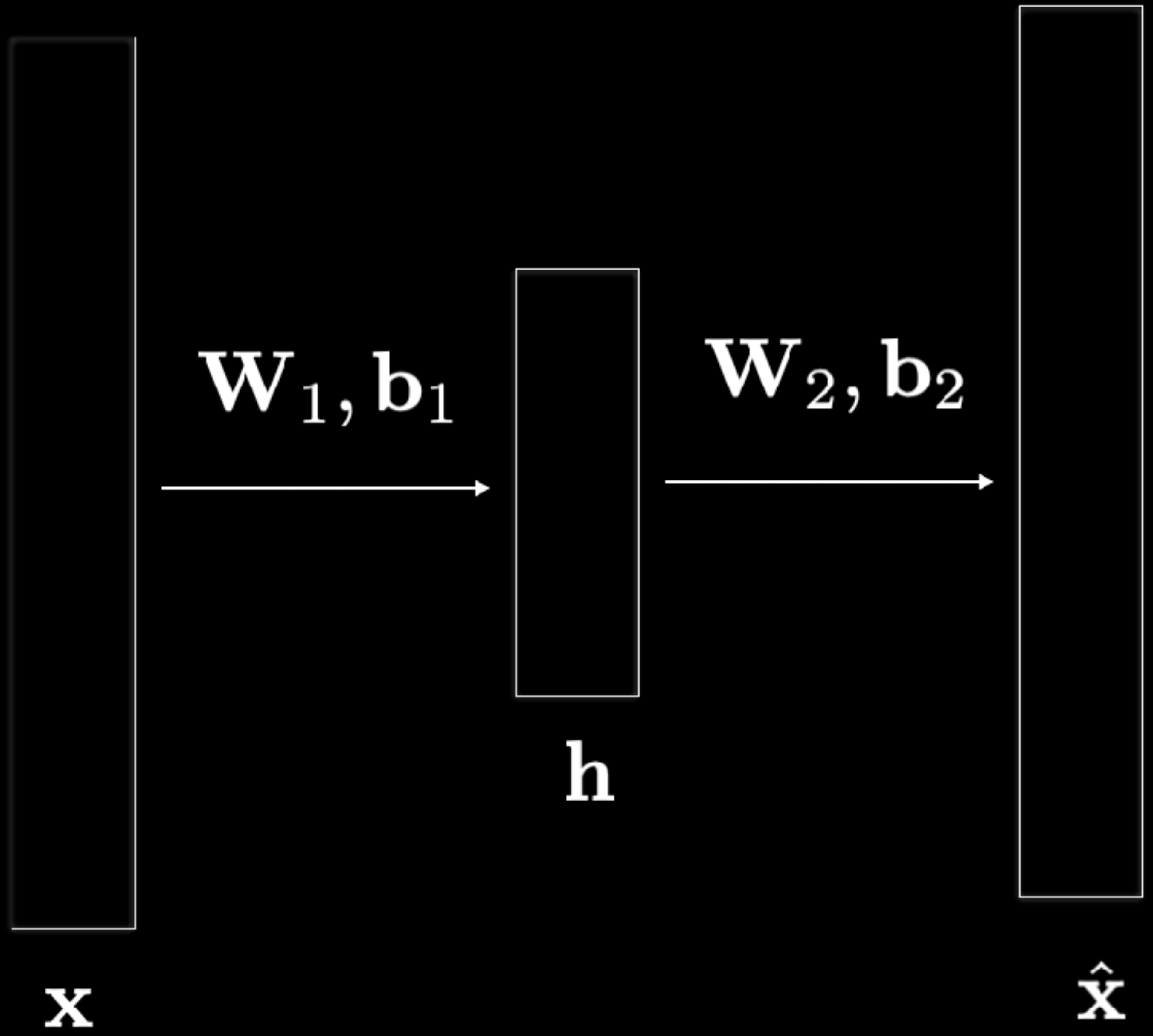
PRINCIPLE COMPONENT ANALYSIS

It has been shown that it is possible to formulate PCA in terms of Neural Networks

$$\hat{\mathbf{x}} = \mathbf{W}\mathbf{W}^T\mathbf{x}$$

$$J_{MSE} = \frac{1}{T}\sum_{j=1}^{T} ||\hat{\mathbf{x}}(j) - \mathbf{W}\mathbf{W}^T\mathbf{x}(j)||^2$$

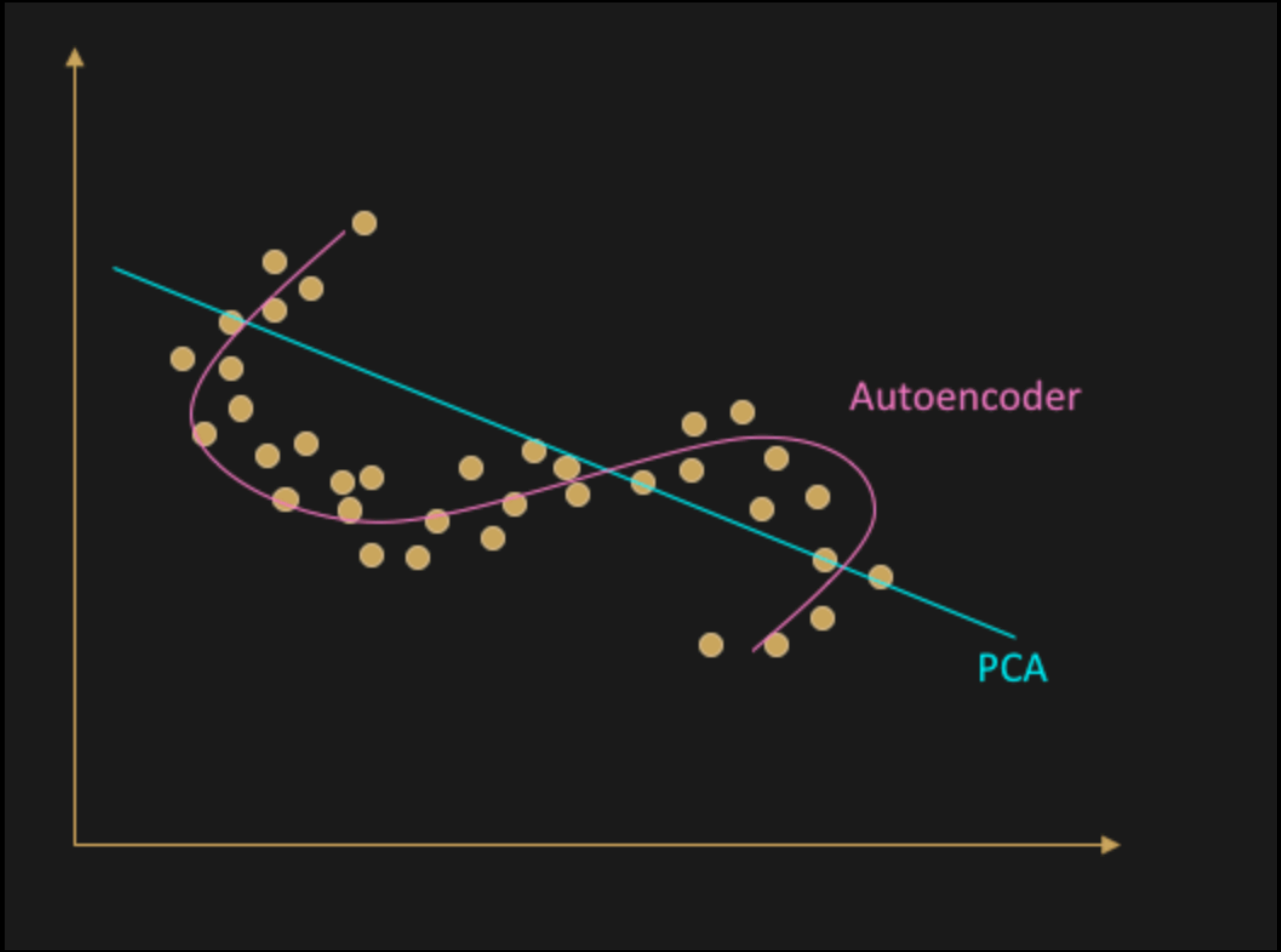
PRINCIPLE COMPONENT ANALYSIS



$$\mathbf{h} = \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$$

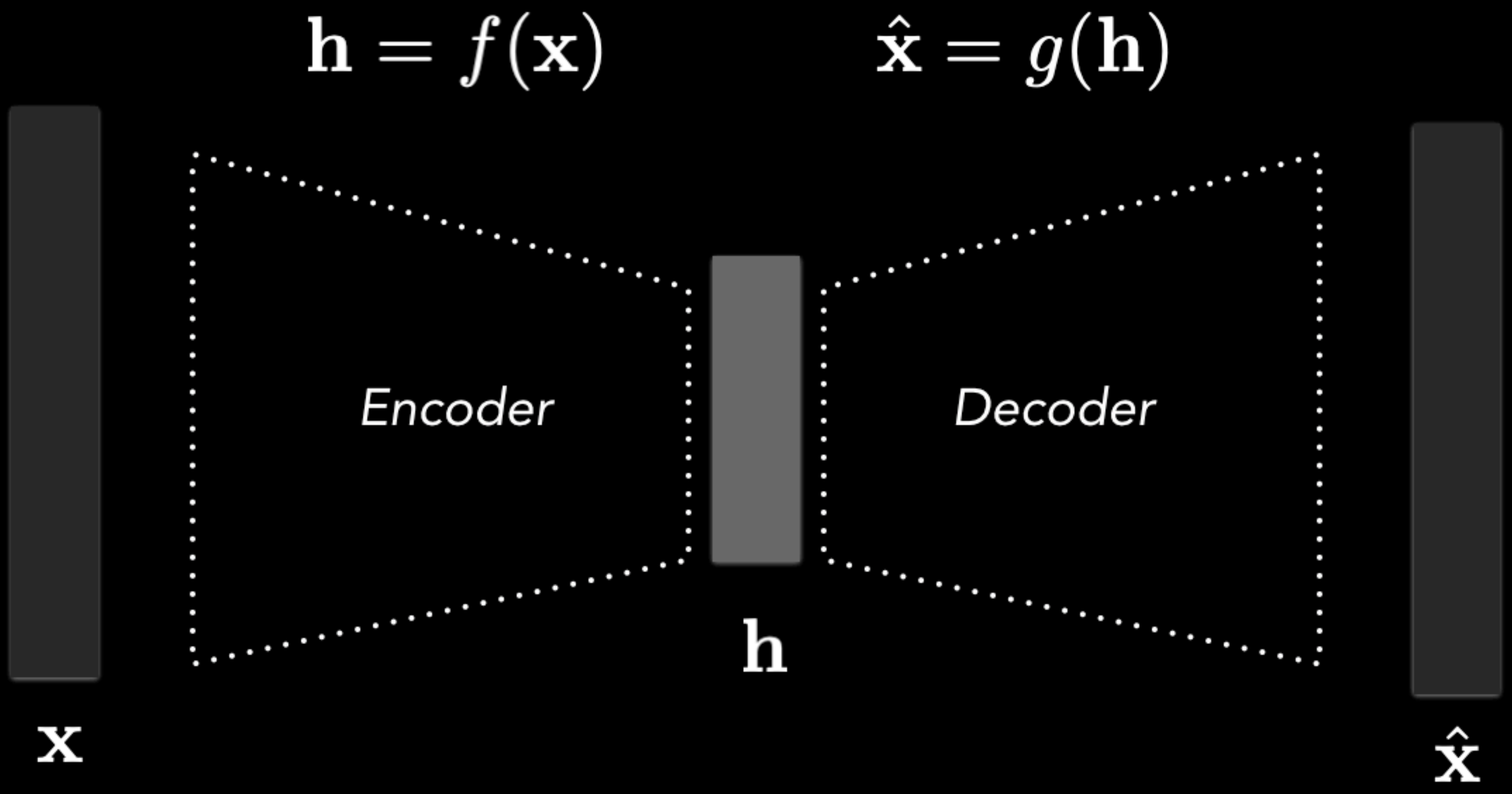
$$\mathbf{y} = \mathbf{W}_2\mathbf{h} + \mathbf{b}_2$$

AUTOENCODER

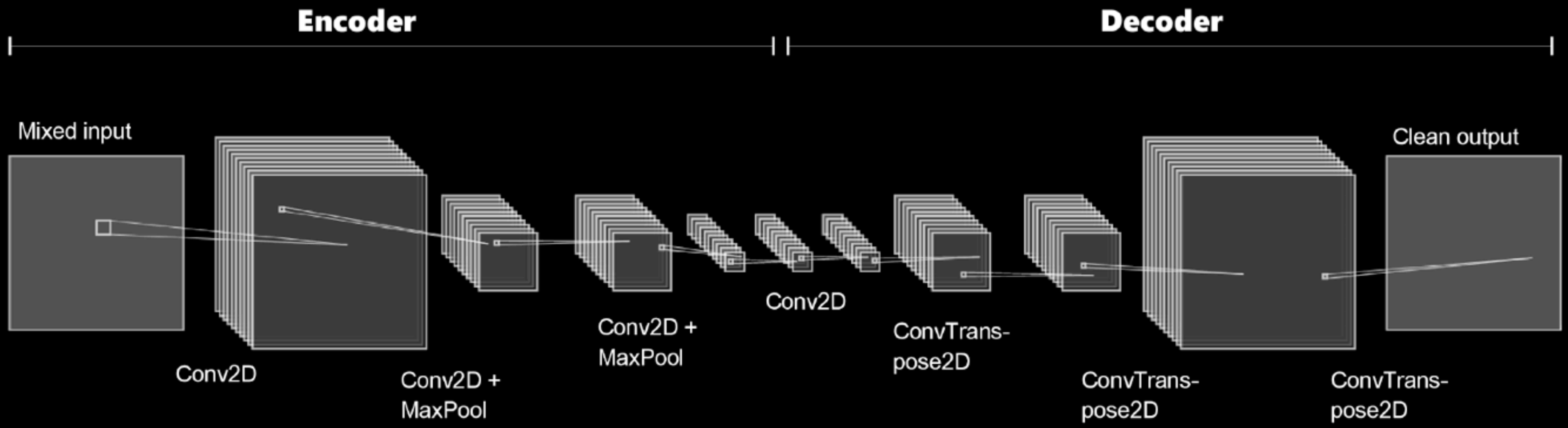


https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

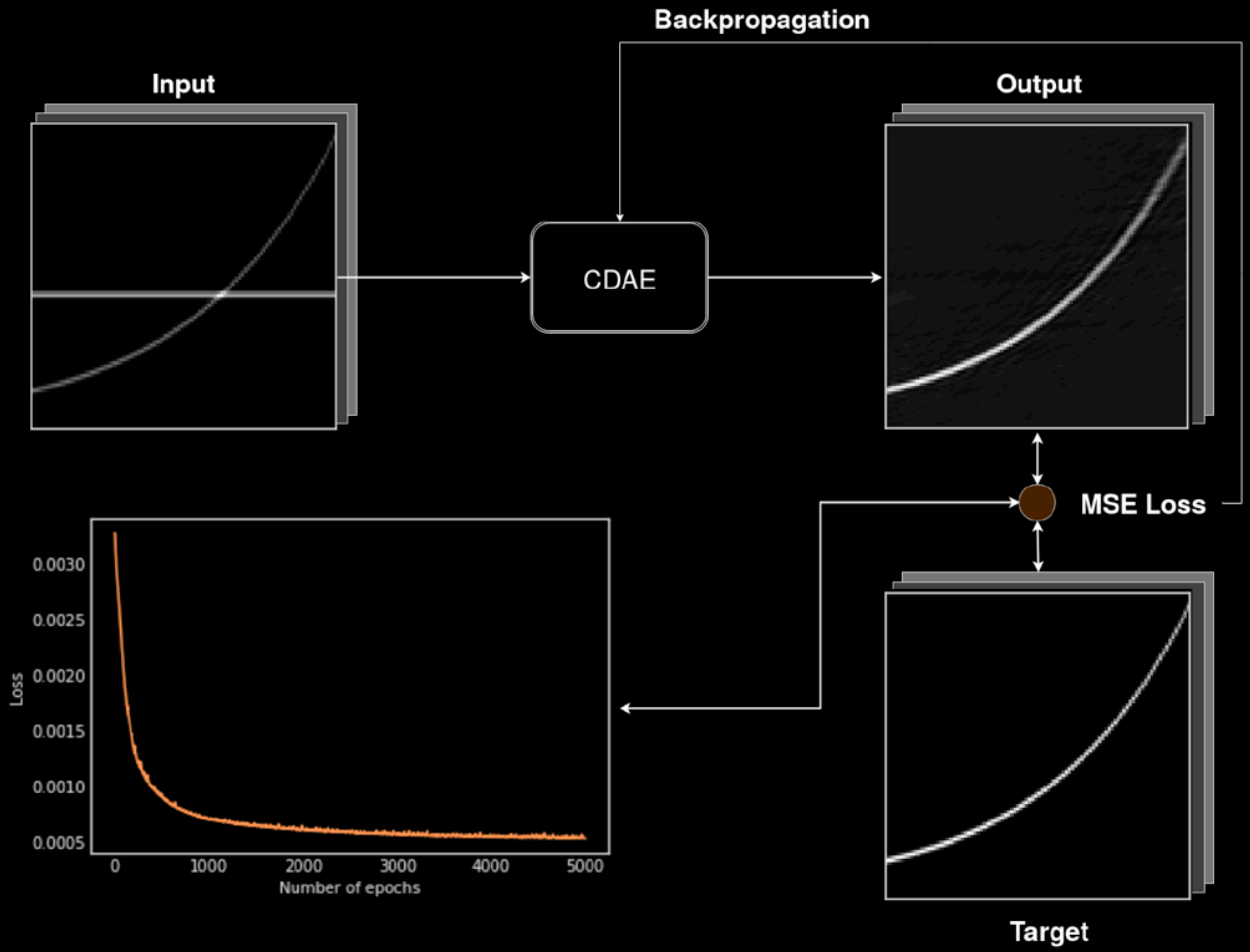
AUTOENCODER



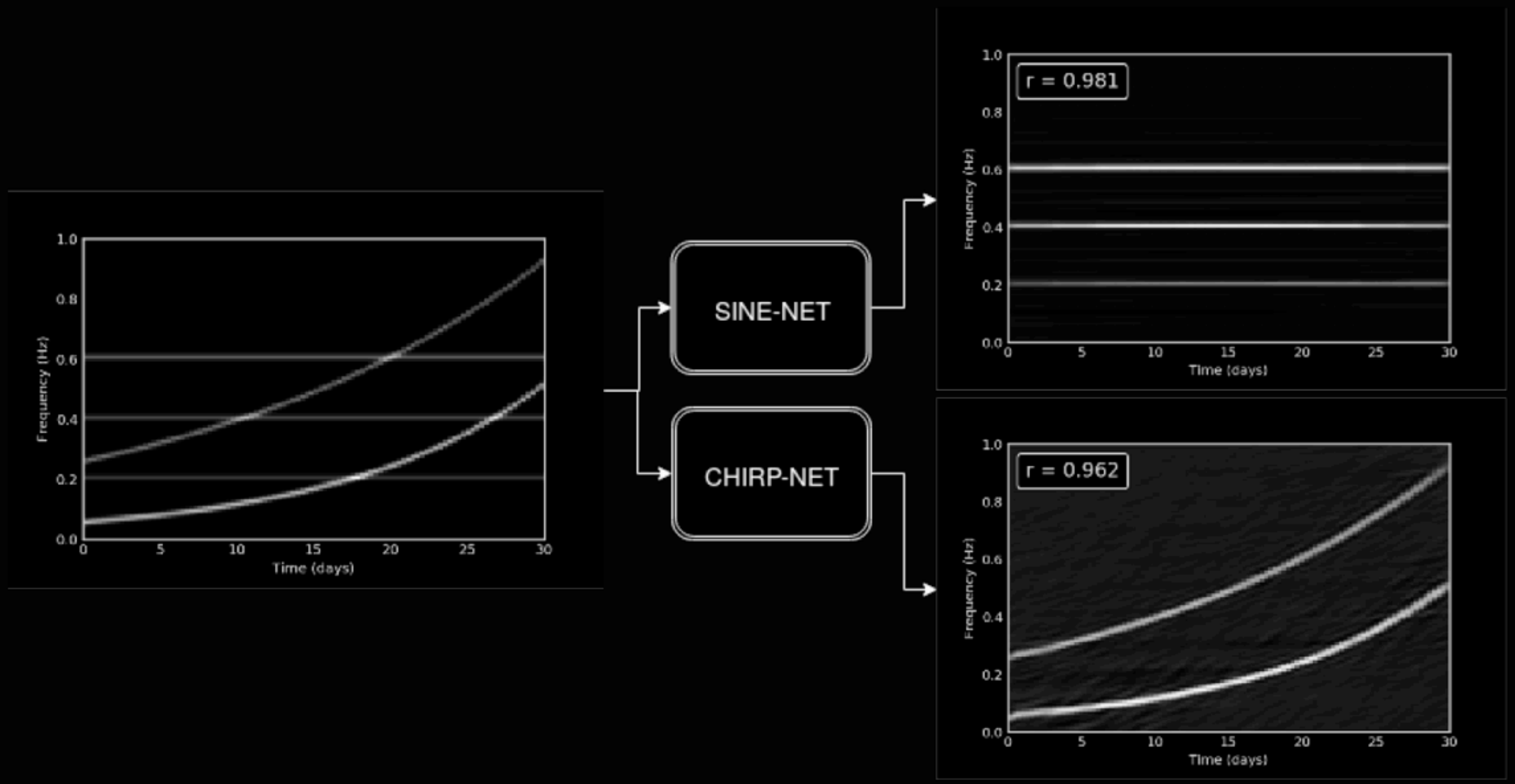
DENOISING AUTOENCODER



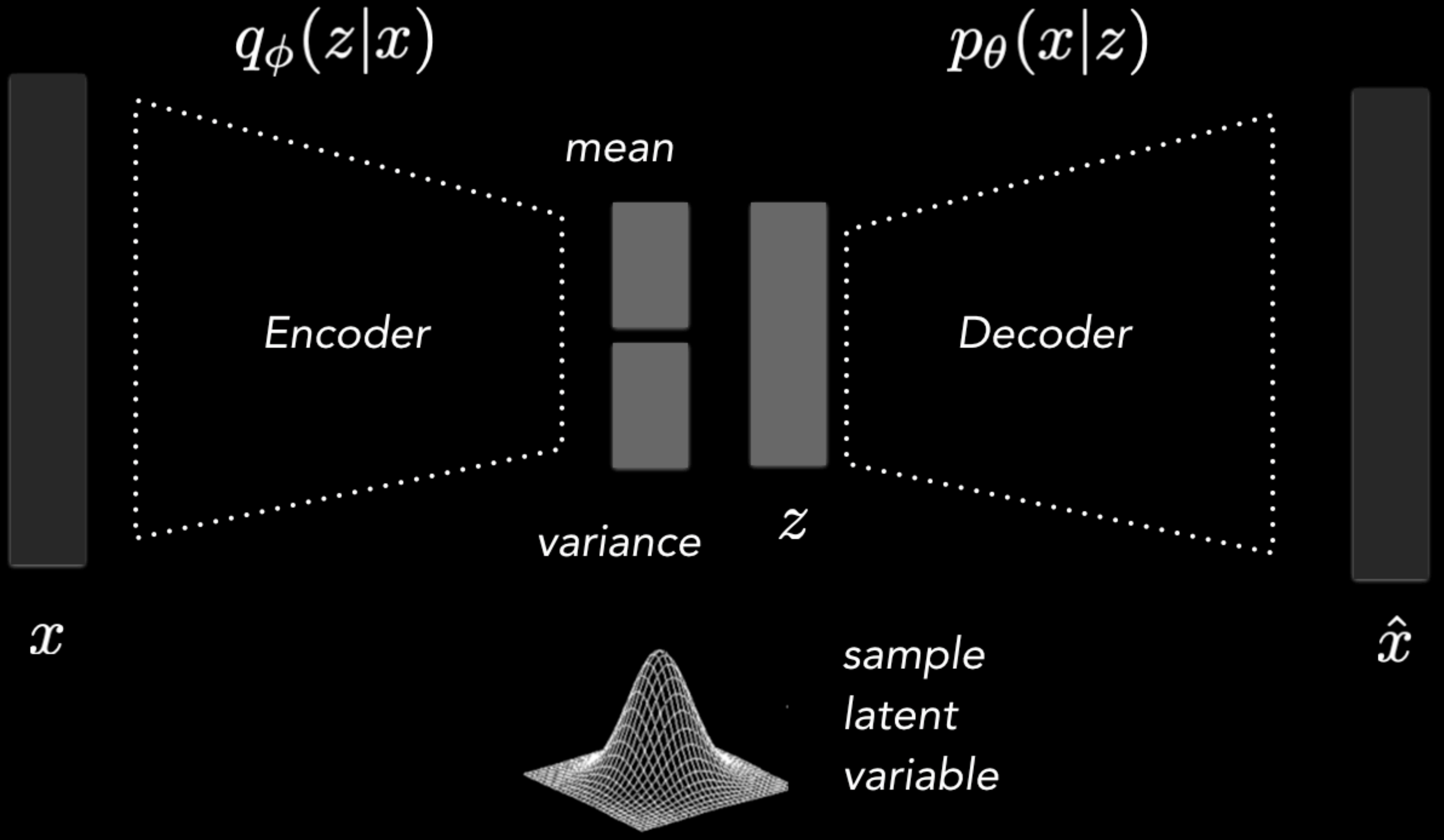
TRAINING THE NETWORK



NETWORK PERFORMANCE



VARIATIONAL AUTOENCODERS



VARIATIONAL AUTOENCODERS

Optimisation

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

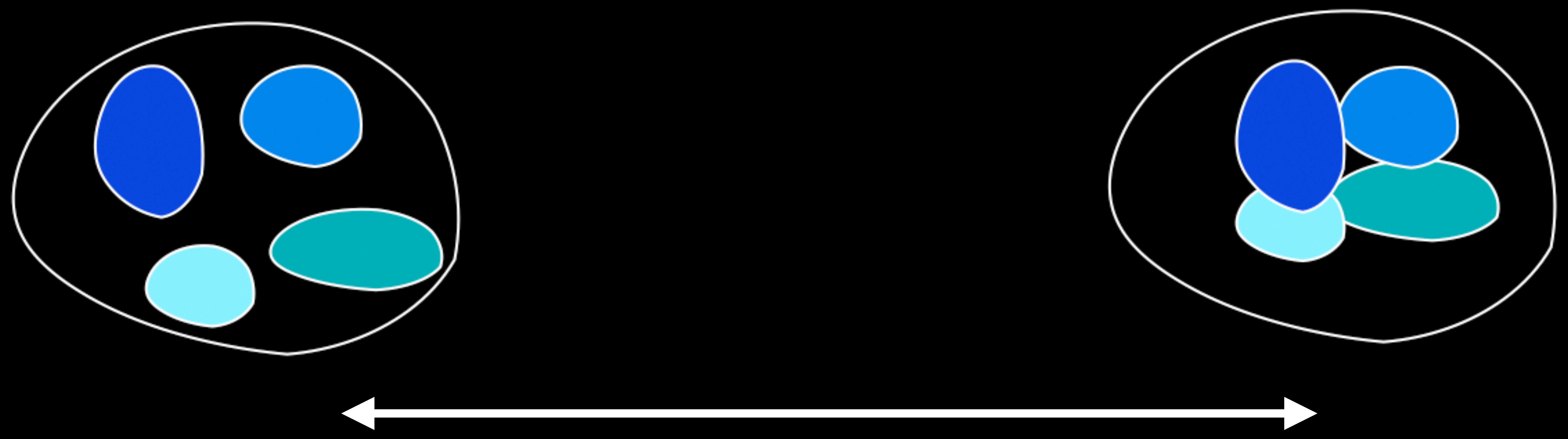
<— We want to estimate latent variable, given data

$p(x)$ — is again intractable but can be approximated using Variational inference

VARIATIONAL AUTOENCODERS

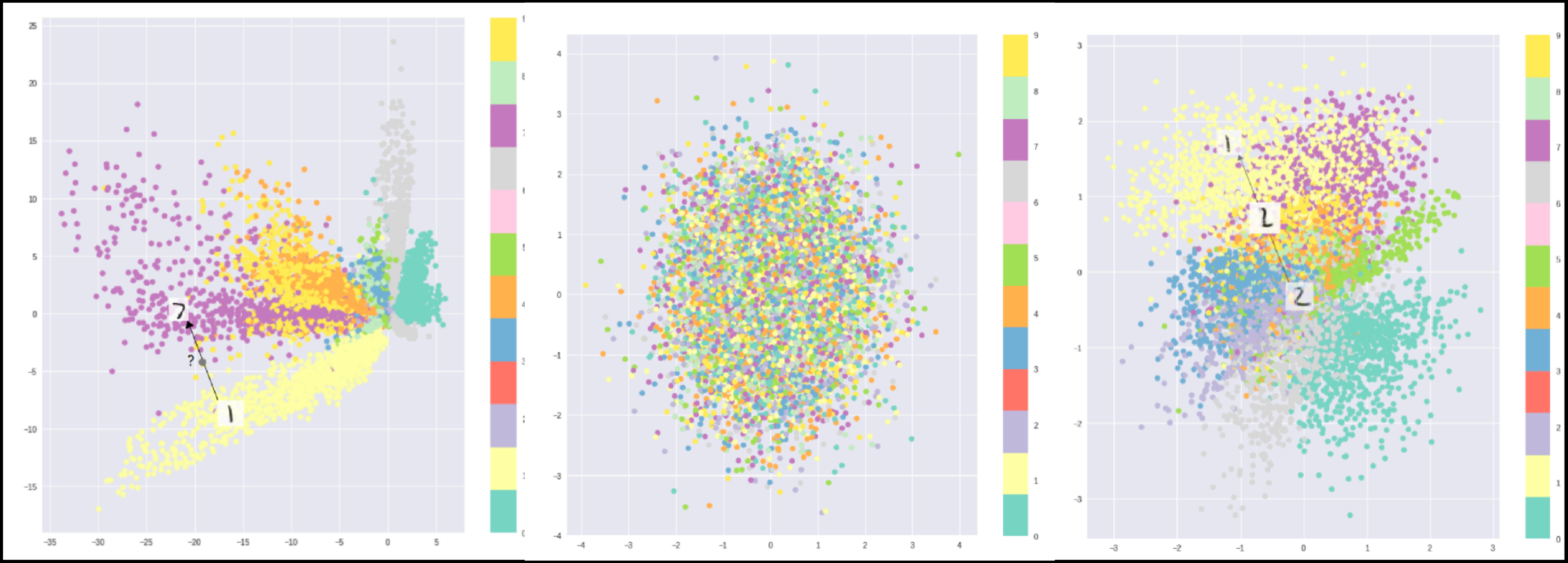
We use ELBO (Evidence Lower BOund)

$$\log p(x) = \text{likelihood} - D_{\text{KL}}[q(z|x)||p(z)]$$



LATENT SPACE

Allows interpolation in the latent space



https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

CONCLUSIONS

- Use this approach as a search.
- Use this approach to embed the data. For example, can project the data in such a way that we only sensitive to one type o signals.
- Use this approach to compress the data.

OTHER APPLICATION

- Filling gaps with the approximation of the joint distribution.
- Speeding up sampling by approximating the likelihood surfaces.
- Surrogate waveforms.
- Anomaly detection for unmodelled searches.
- Data analysis without TDI.
- Optimisation with Reinforcement Learning.
- ….