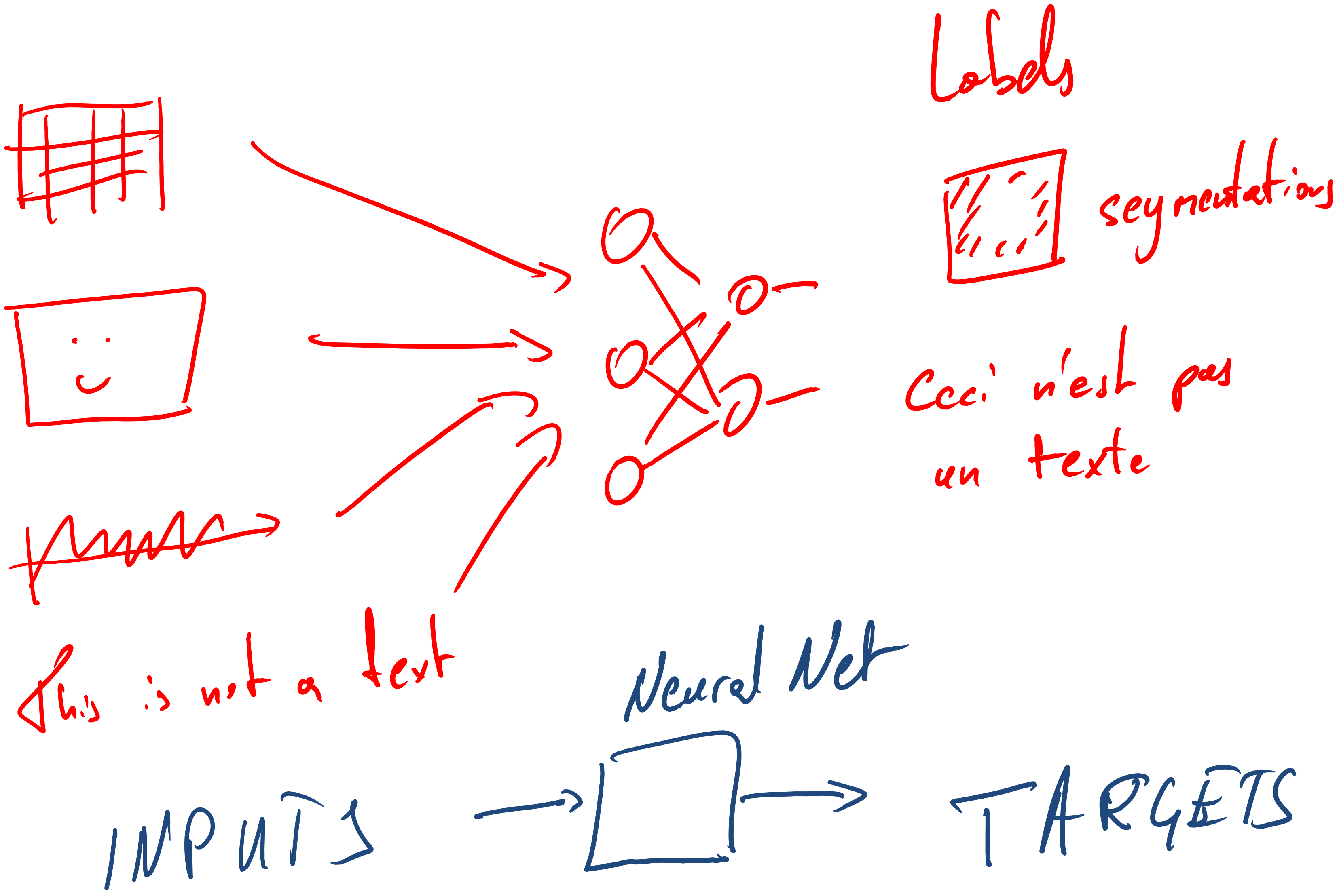
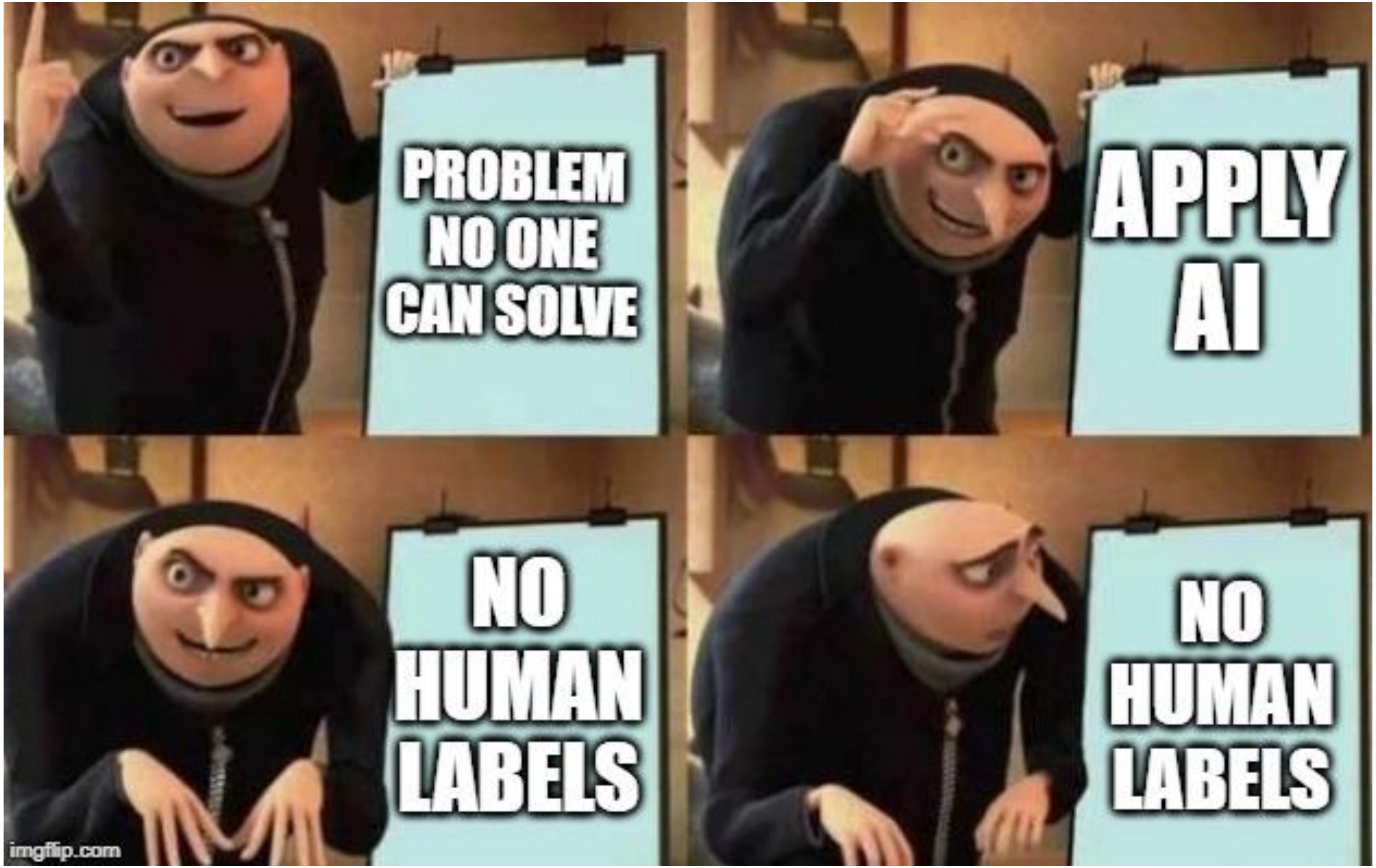
Deep Unsupervised Learning

Motivation


Labels
segmentations
Ceci n'est pas un texte
This is not a text
Neural Net
INPUTS
TARGETS


PROBLEM NO ONE CAN SOLVE
APPLY AI
NO HUMAN LABELS
NO HUMAN LABELS
imgflip.com

Unsupervised Learning === Without Targets
What tasks can we solve?

Question

How to formalize / implement

Cluster / find similar

self - sim metrics

Find distrib of data

↳ predict the "next"

$X = \{x_1 \ldots x_n\}$

$\underbrace{\prod_{i=1}^{N} p_\theta(x_i)}$

Learn a "useful" representation

Transform the data.

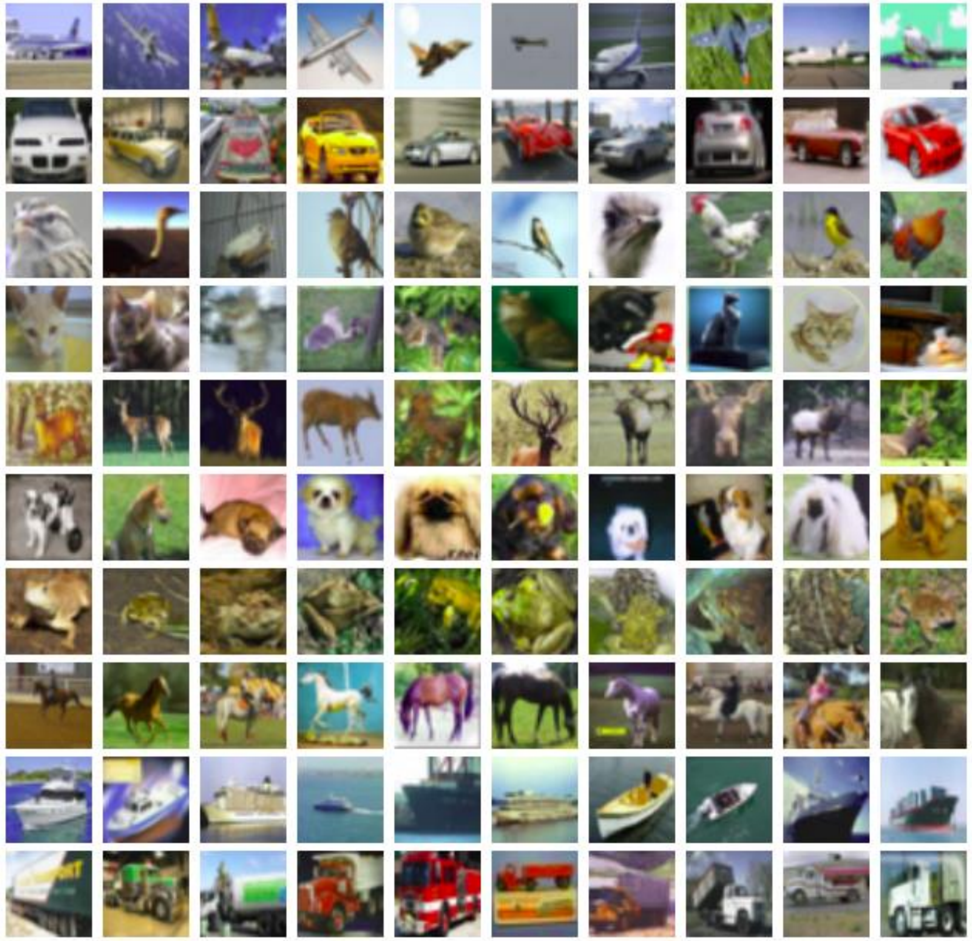
- does not loose too much info
- is easier to work with

Unsupervised learning goals

- Learn a data representation:
  – Extract features for downstream tasks
  – Describe the data (clusterings)
- Data generation / density estimation
  – What are my data
  – Outlier detection
  – Super-resolution
  – Artifact removal
  – Compression

Data generation (Learning high dimensional prob. distributions) is hard

- Assume we work with small (32x32) images
- Each data point is a real vector of size $32 \times 32 \times 3$
- Data occupies only a tiny fraction of $\mathbb{R}^{32\times32\times3}$
- Difficult to learn!



QQ: what generative models we have already seen?

Text gen =

$p(x_1 \ldots x_T) =$

$p(x_1)\, p(x_2 | x_1)\, p(x_3 | x_1, x_2) \ldots$

simple

Autoregressive Example: Language modeling

Let x be a sequence of word ids.

$$p(x) = p(x_1, x_2, \dots, x_n) = \prod_i p(x_i | x_{<i})$$

$$\approx \prod_i p(x_i | x_{i-k}, x_{i-k+1}, \dots, x_{i-1})$$

p(It’s a nice day) = p(It) * p(‘s|it) * p(a|’s)…

- Classical n-gram models: cond. probs. estimated using counting
- Neural models: cond. probs. estimated using neural nets

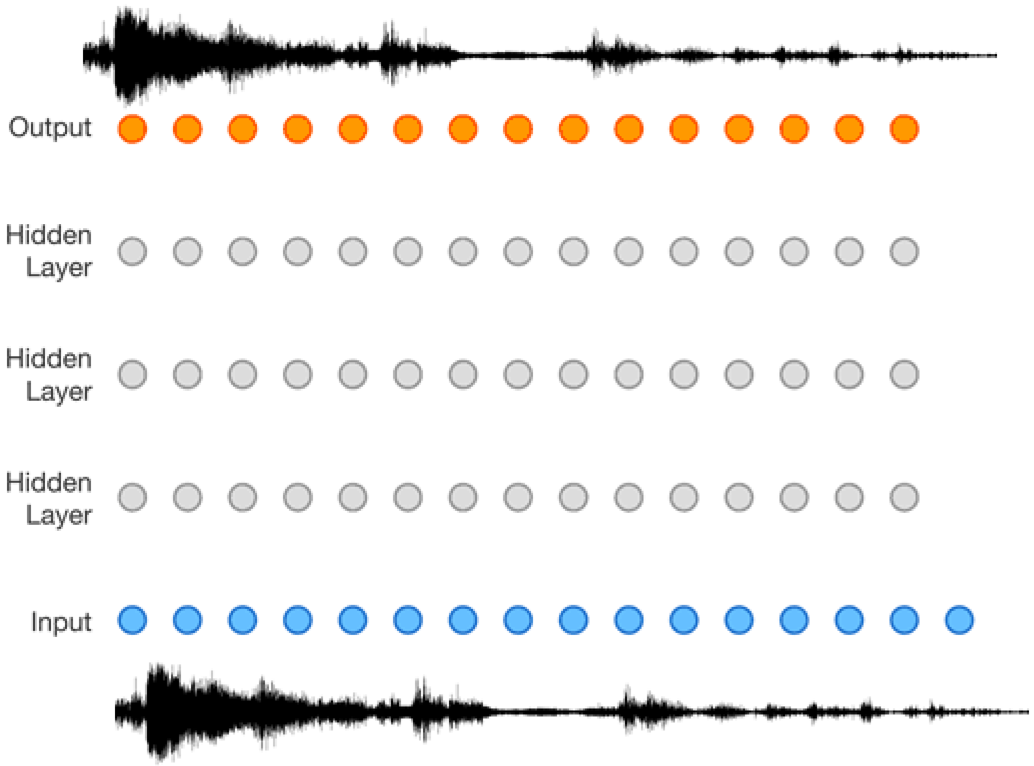
WaveNet:
Autoregressive modeling of speech

Treat speech as a sequence of samples!

Predict each sample base on previous ones.

16 kHz

$\mathbb{R}^{16000}$ per sec.



https://arxiv.org/abs/1609.03499

PixelCNN:
A "language model for images"

Pixels generated left-to-right, top-to-bottom.

$p(x_i|x_{<i})$ computed using convolutional neural nets

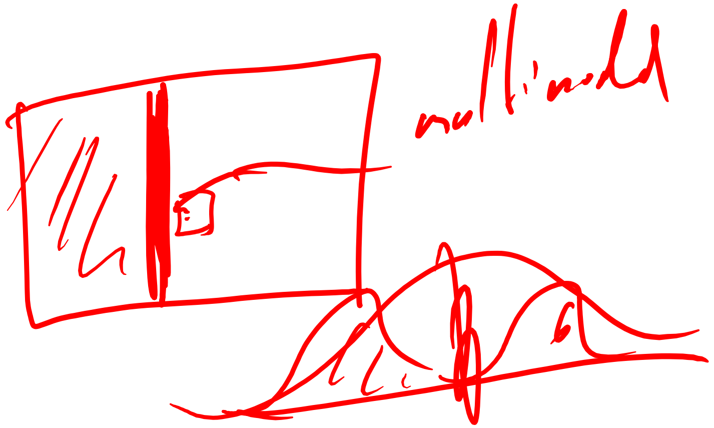
Model supports auxiliary conditioning: $p(x_i|x_{<i}, c)$

van den Oord, A., et al. "Pixel Recurrent Neural Networks." ICML (2016).

Modeling pixels

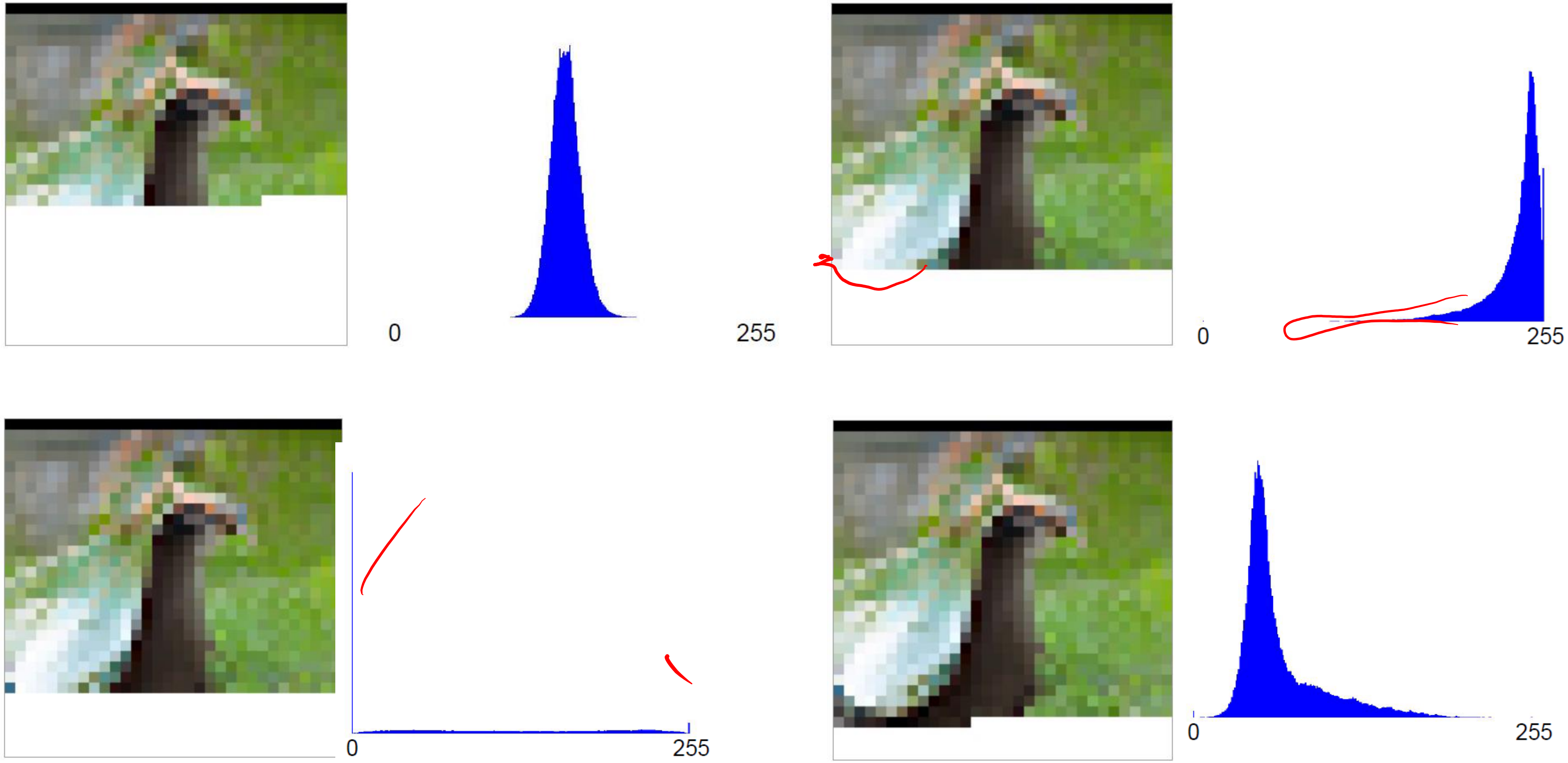
How to model pixel values:

- A Gaussian with fixed st. dev?
- A Gaussian with tunable st. dev?
- A distribution over discrete levels [0,1,2,…255]?



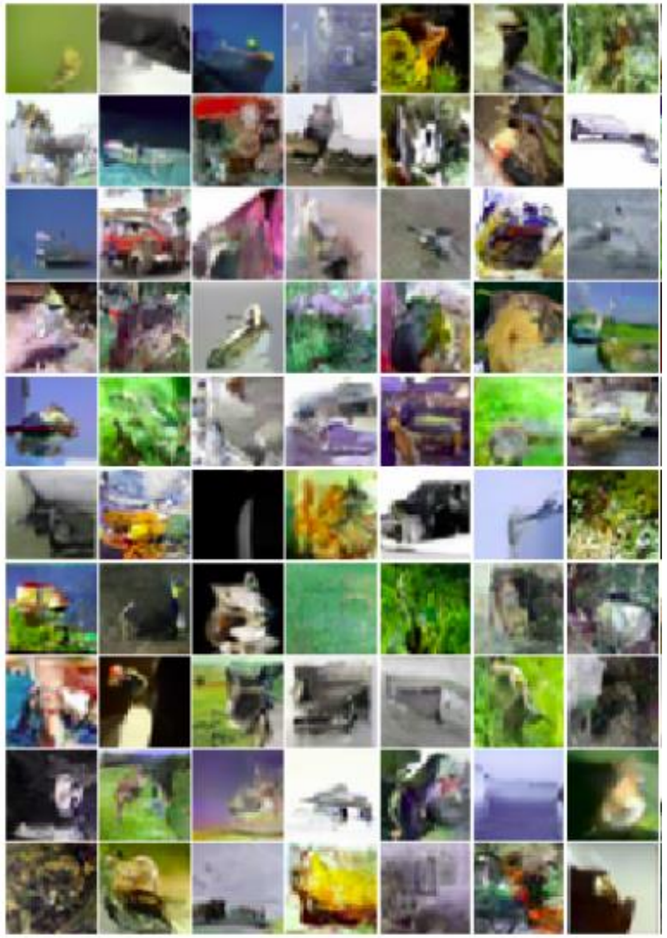
What are the implications?

Modeling pixel values



Model works best with a flexible distribution: better to use a SoftMax over pixel values!

PixelCNN samples & completions

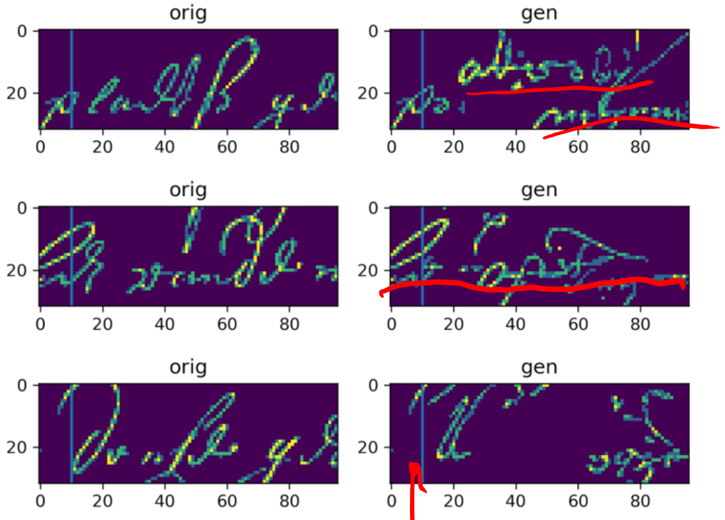


Salimans et al, "A PixelCNN Implementation with Discretized Logistic Mixture Likelihood and Other Modifications"
van den Oord, A., et al. "Pixel Recurrent Neural Networks." ICML (2016).

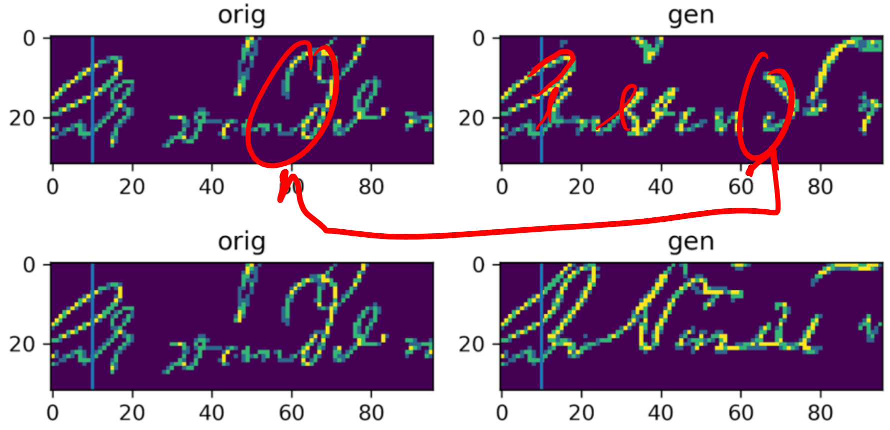
PixelCNN samples

handwriting synthesis system

Unconditional



Conditioned on text



Autoregressive Models Summary

The good:

- Easy to exploit correlations in data.
- Reduce data generation to many small decisions
  - Simple define (just pick an ordering)
  - Trains like fully supervised
  - Model operations are deterministic, randomness needed during generation
- Often SOTA log-likelihood

state-of-the-art

Autoregressive Model Summary

The bad:

- Train/test mismatch (teacher forcing):
  trained on ground truth sequences
  but applied to own predictions
- Generation requires $O(n)$ steps
  (Training can be sometimes parallelized)
- No compact intermediate data representation,
  not obvious how to use for downstream tasks.

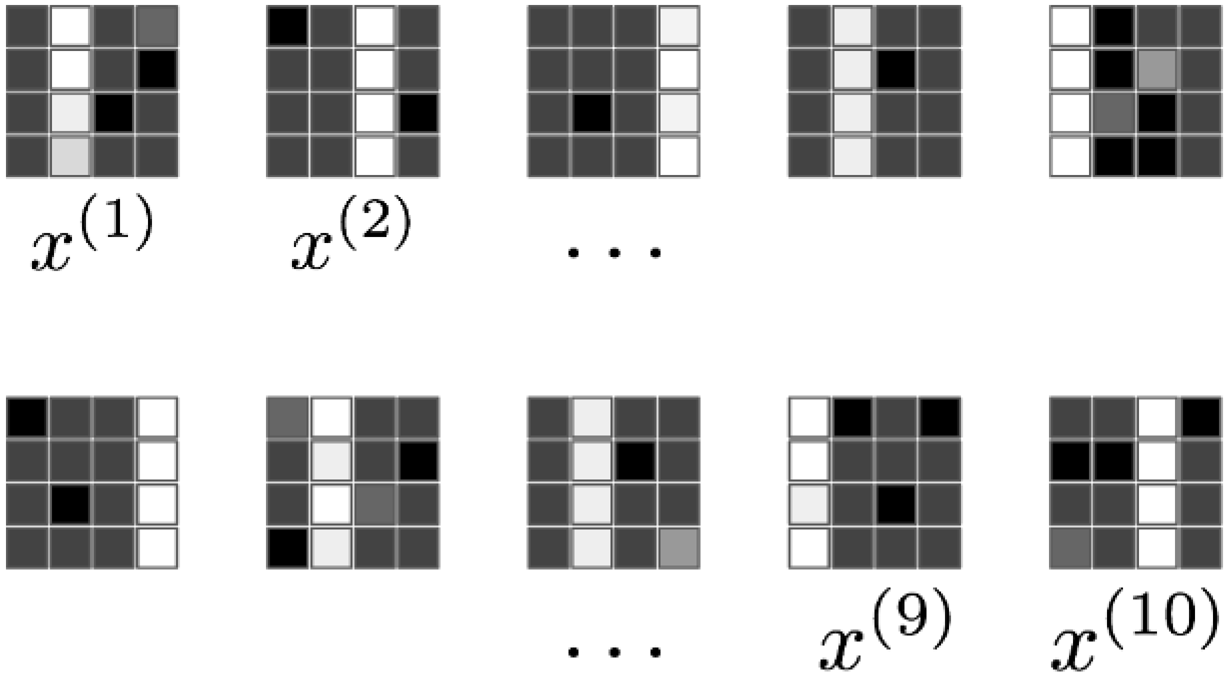
Ad break

I got the many of following slides from Ulrich Paquet (DeepMind)

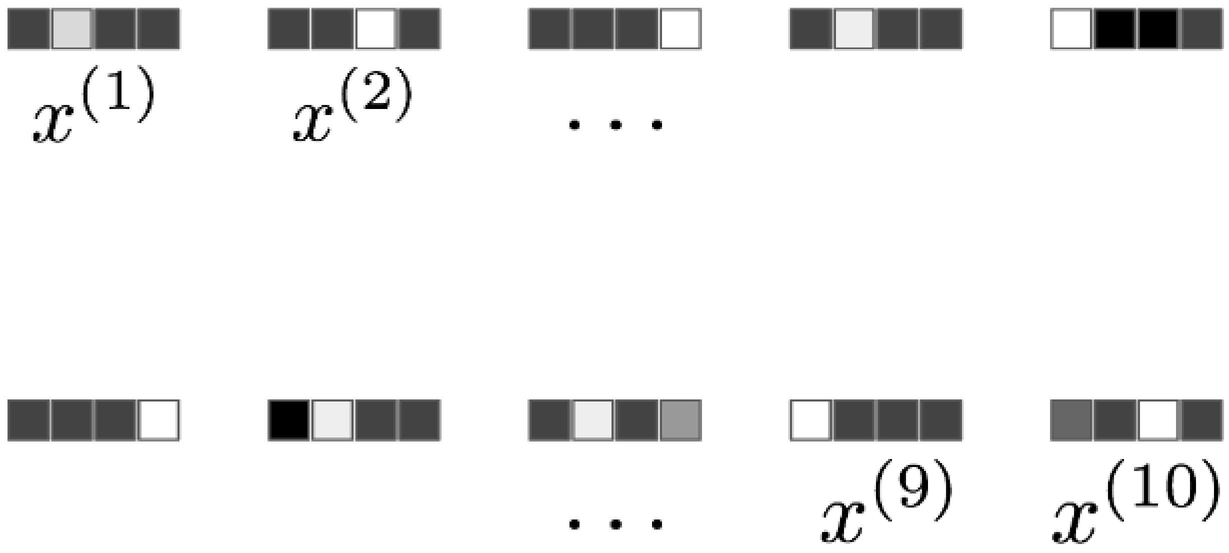
See https://www.youtube.com/watch?v=xTsnNcctvmU for a recording of a his explanation!

2 minute exercise

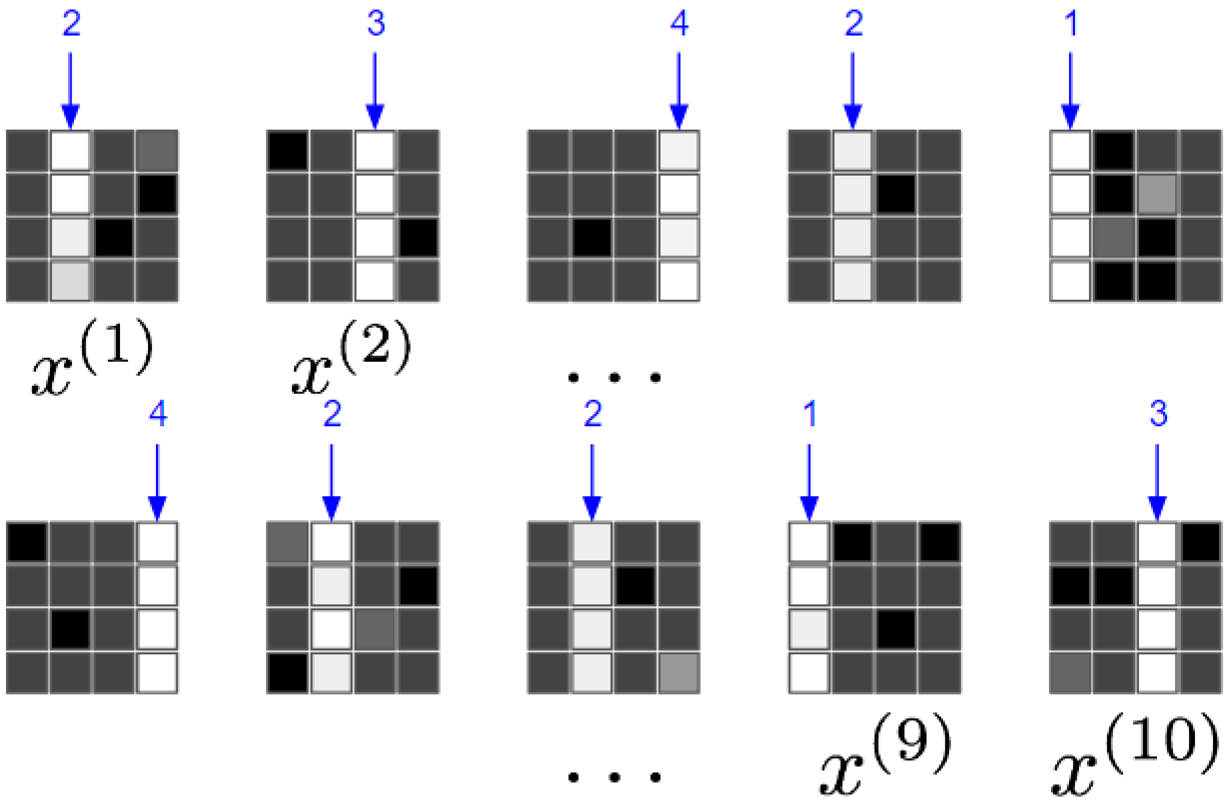
Think about all the properties of this dataset:



The rows are correlated



There's a simple encoding (the LATENT representation)



Data structure

We can capture most of the variability in the data through **one** number

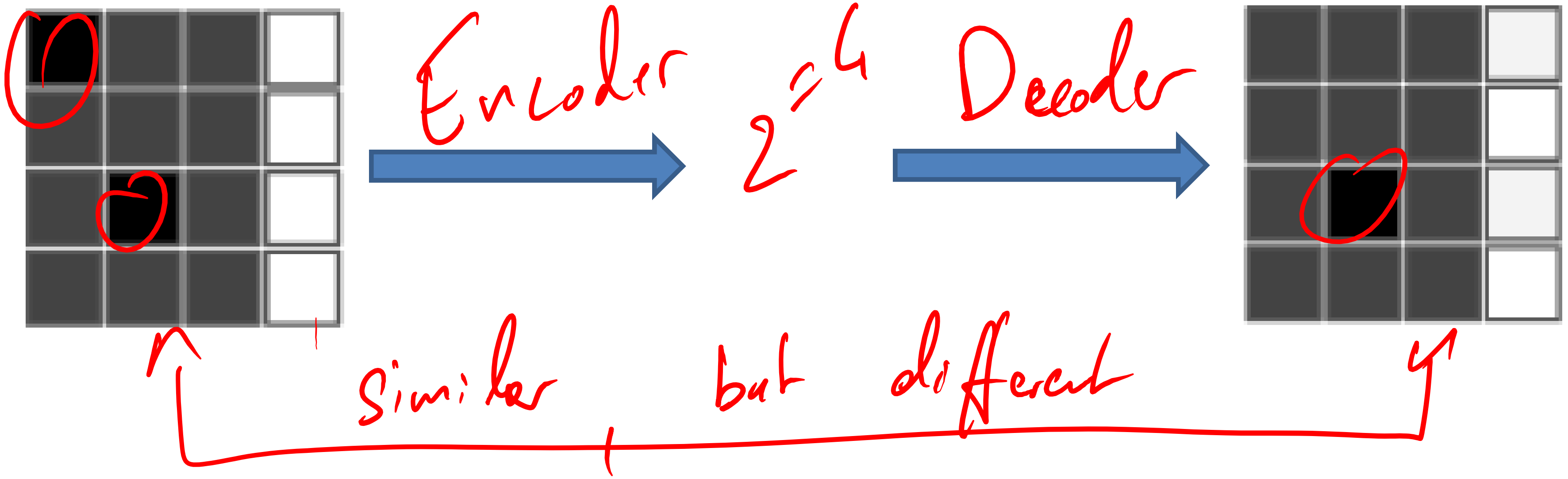
$$z^{(n)} = 1 \text{ or } 2, 3, 4$$

for each image *n*, even though each image is 16 dimensional

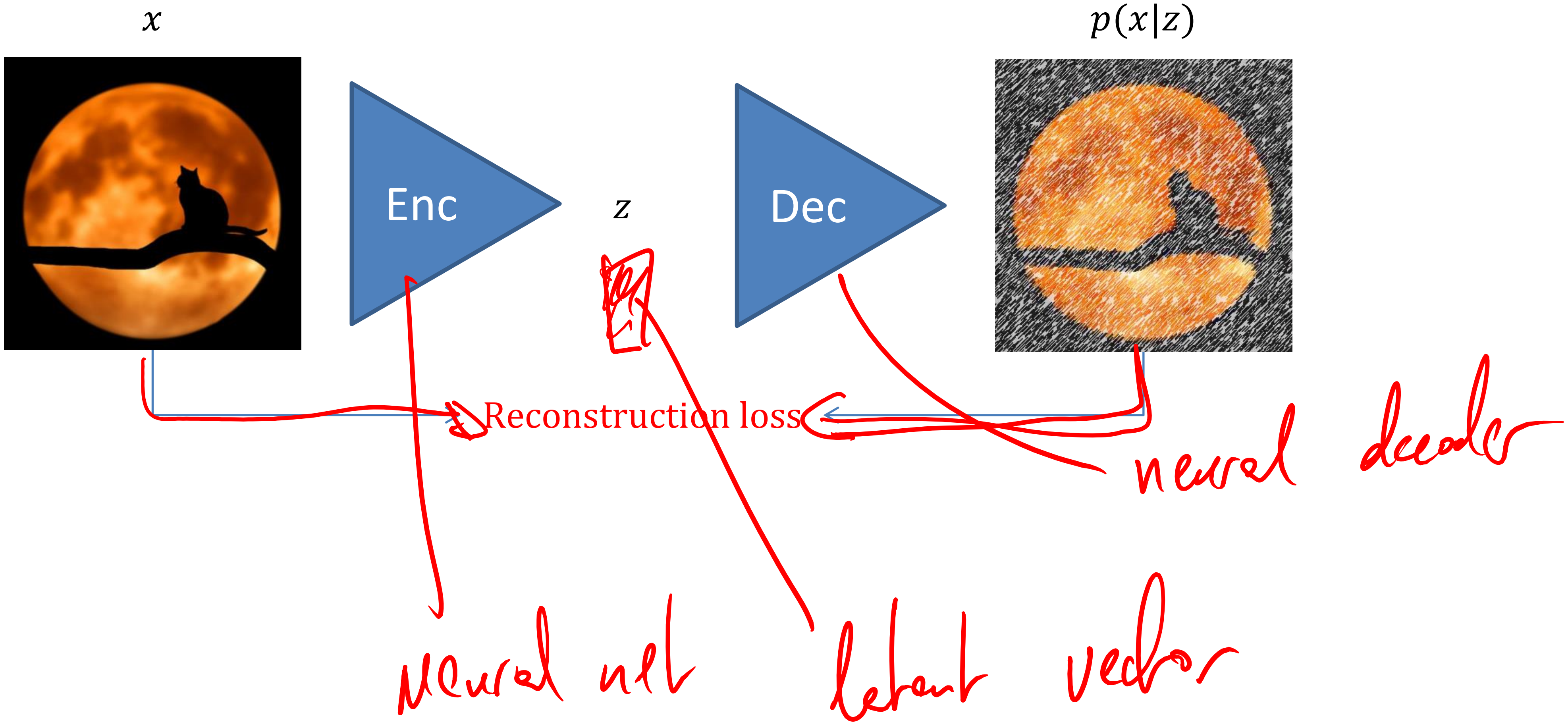
How?

Autoencoders

SELF

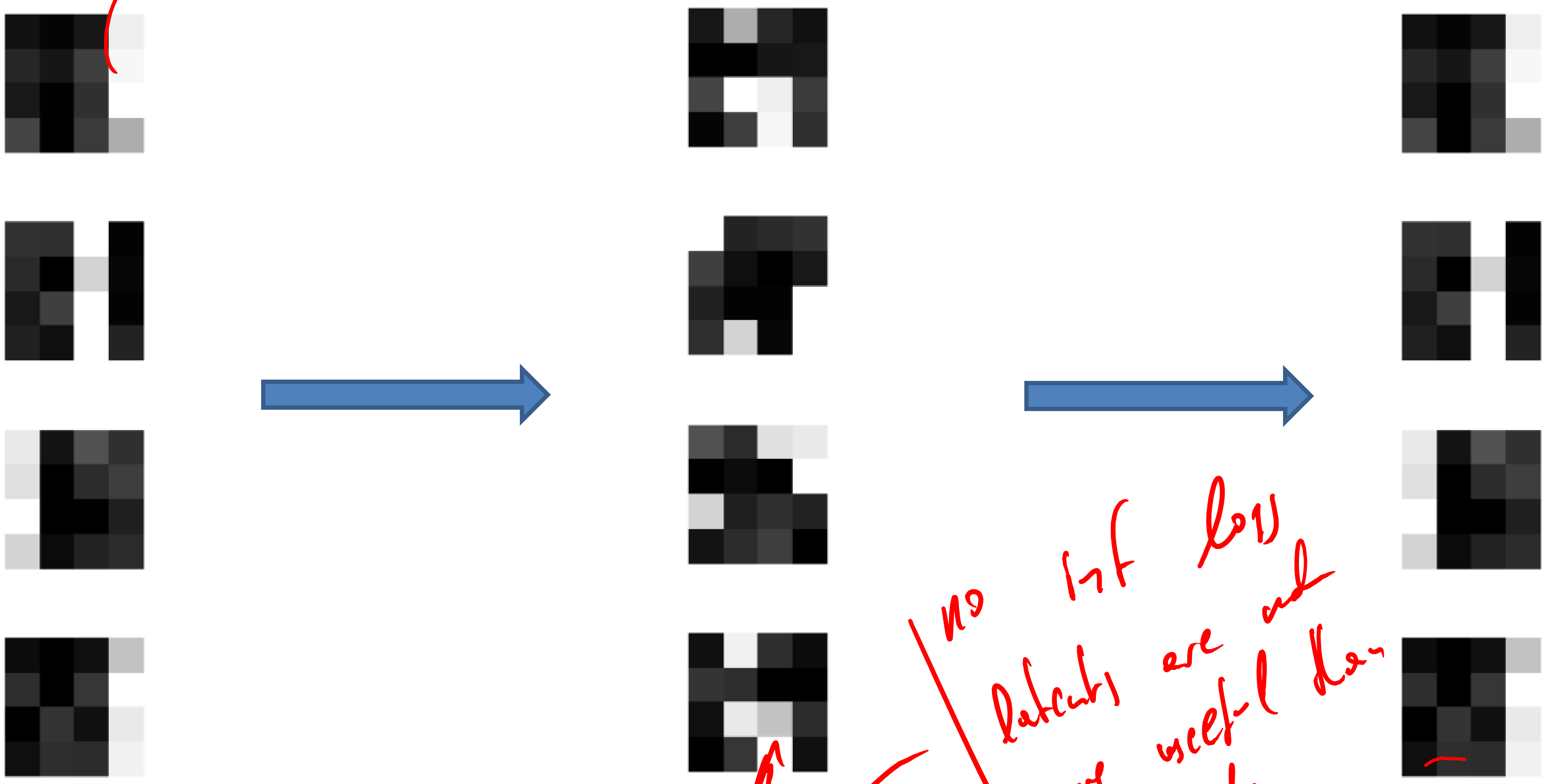


Autoencoder



Perfect Reconstruction === Useful Representations?

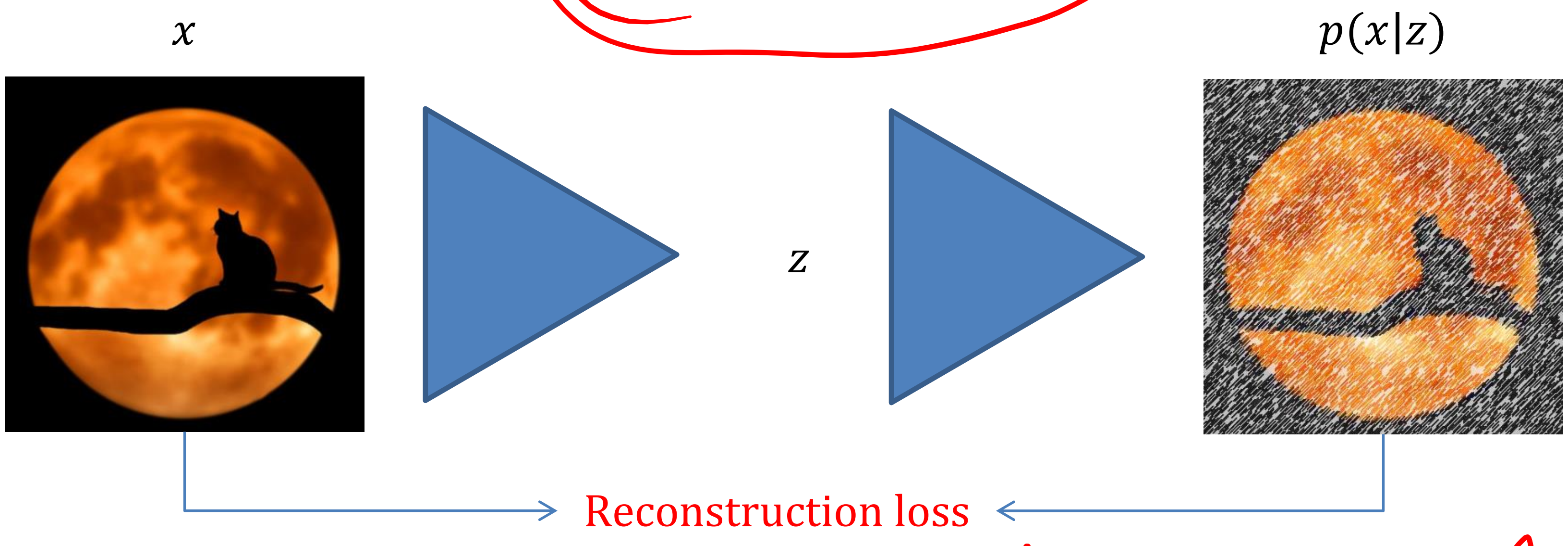
exactly the same



no info loss

latents are not more useful than inputs

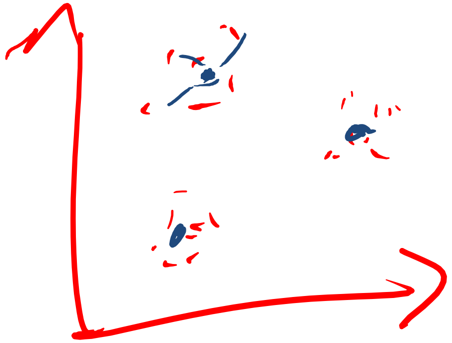
Bottleneck



Bottleneck prevents "trivial" encodings

- Dimensionality reduction
- Sparsity (z is mostly 0)

⋮



VARIATIONAL AUTOENCODER

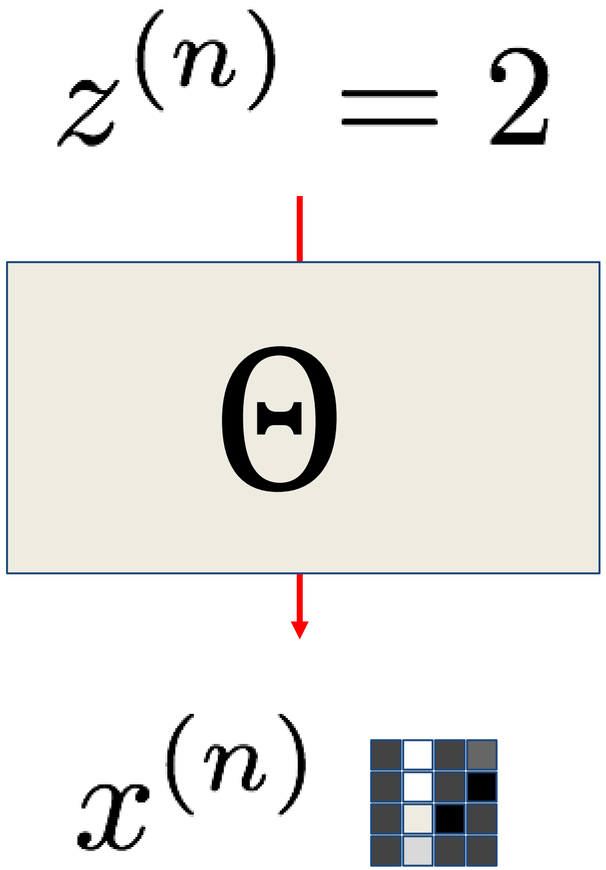
Exercise: first, implement a decoder

Write or draw a function (like a multi-layer perceptron) that takes $z \in \mathbb{R}$ and produces x

Is your input one-dimensional?

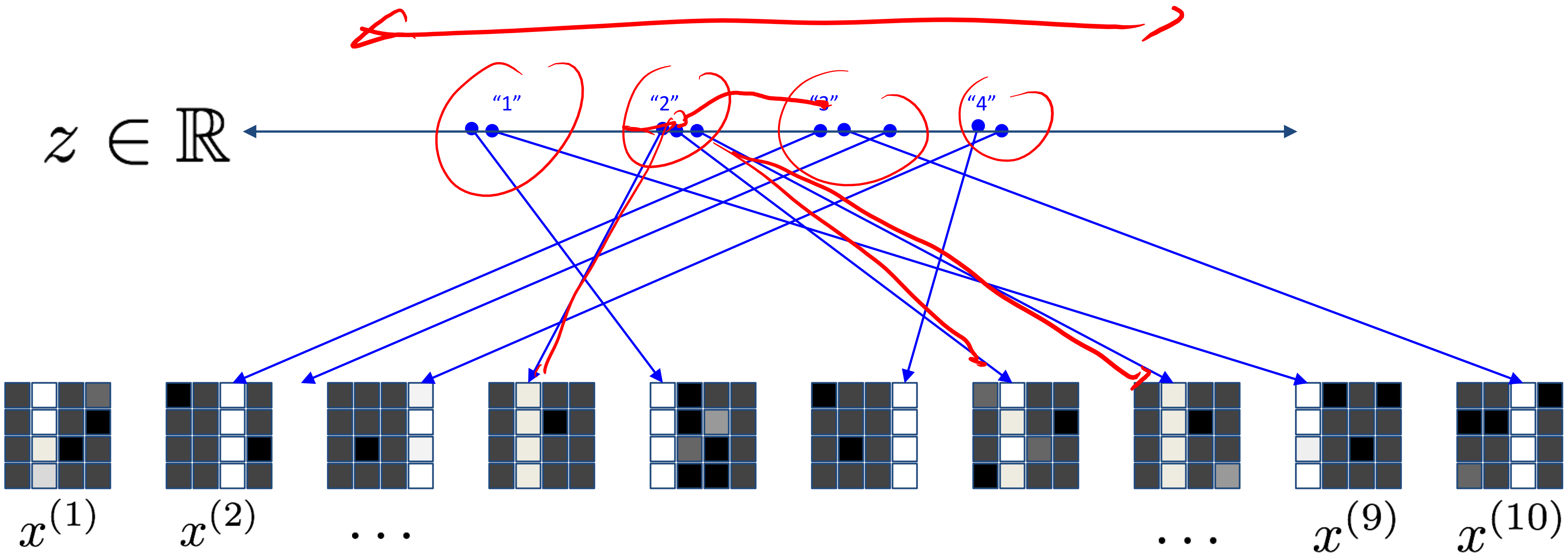
Is your output 16-dimensional?

Identify all the “tunable” parameters Θ of your function



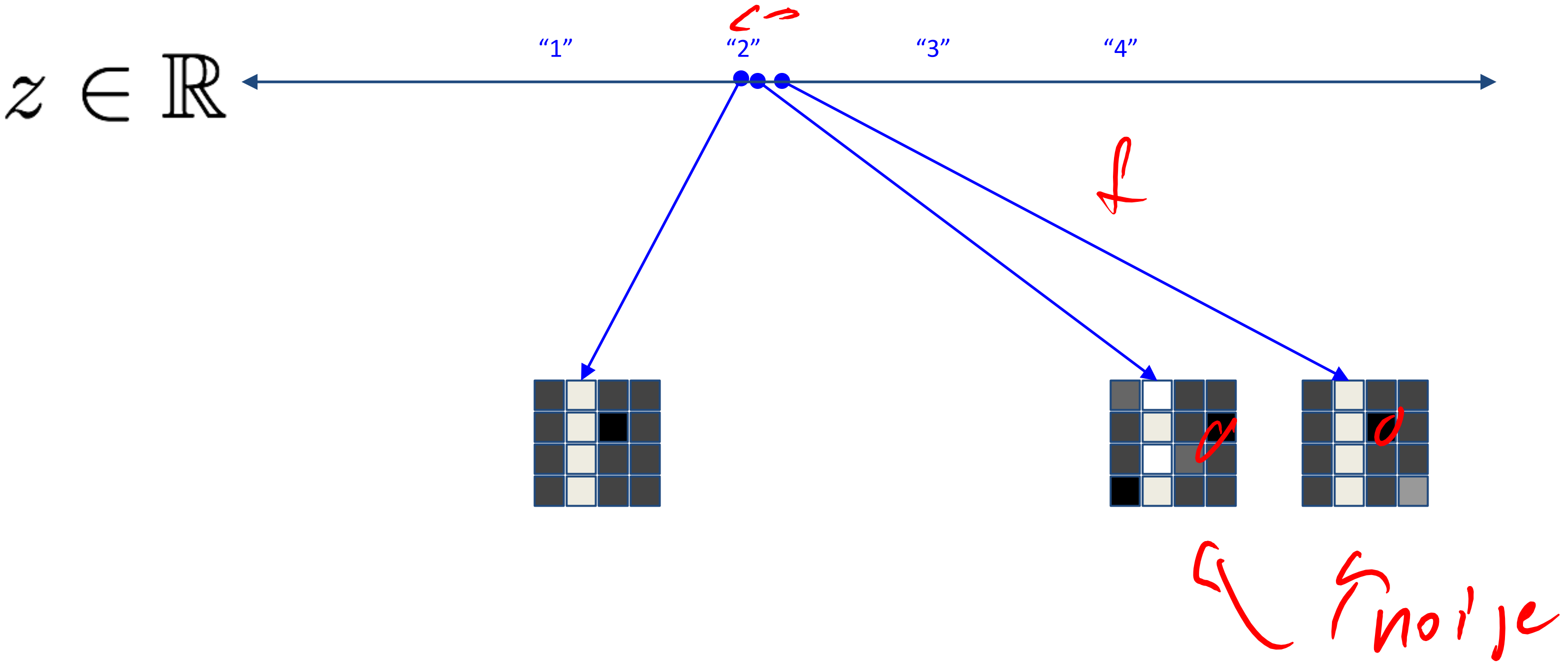
Data manifold

The 16-dimensional images live on a 1-dimensional manifold, plus some “noise”



and noise

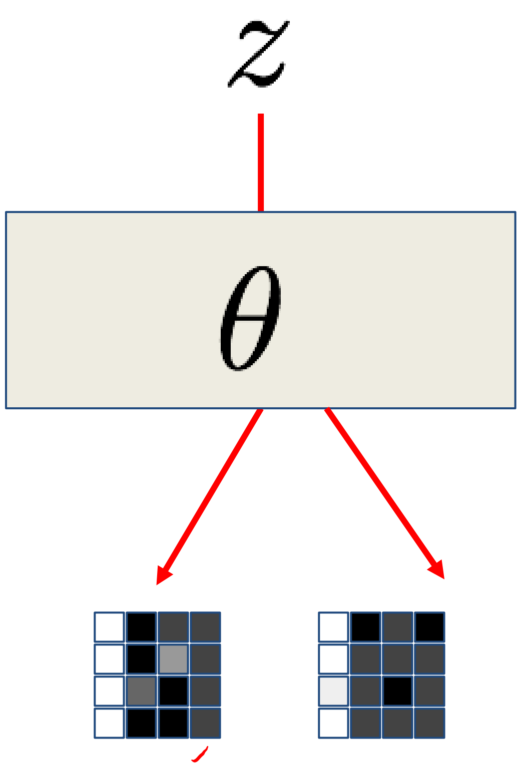
The 16-dimensional images live on a 1-dimensional manifold, plus some "noise"



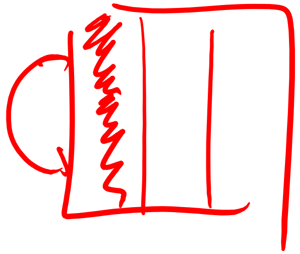
Exercise

Change the neural network to take z and produce a distribution over x:

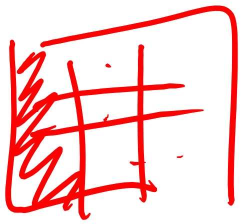
$$p_\Theta(x|z)$$



$P_\theta(x)$



2 = 1



Generation and Inference

Generation:

$$p(z)$$
$$p_\Theta(x|z)$$

generative model with latents z!

Inference:

$$p_\Theta(z|x) = ????$$

Bayes says:

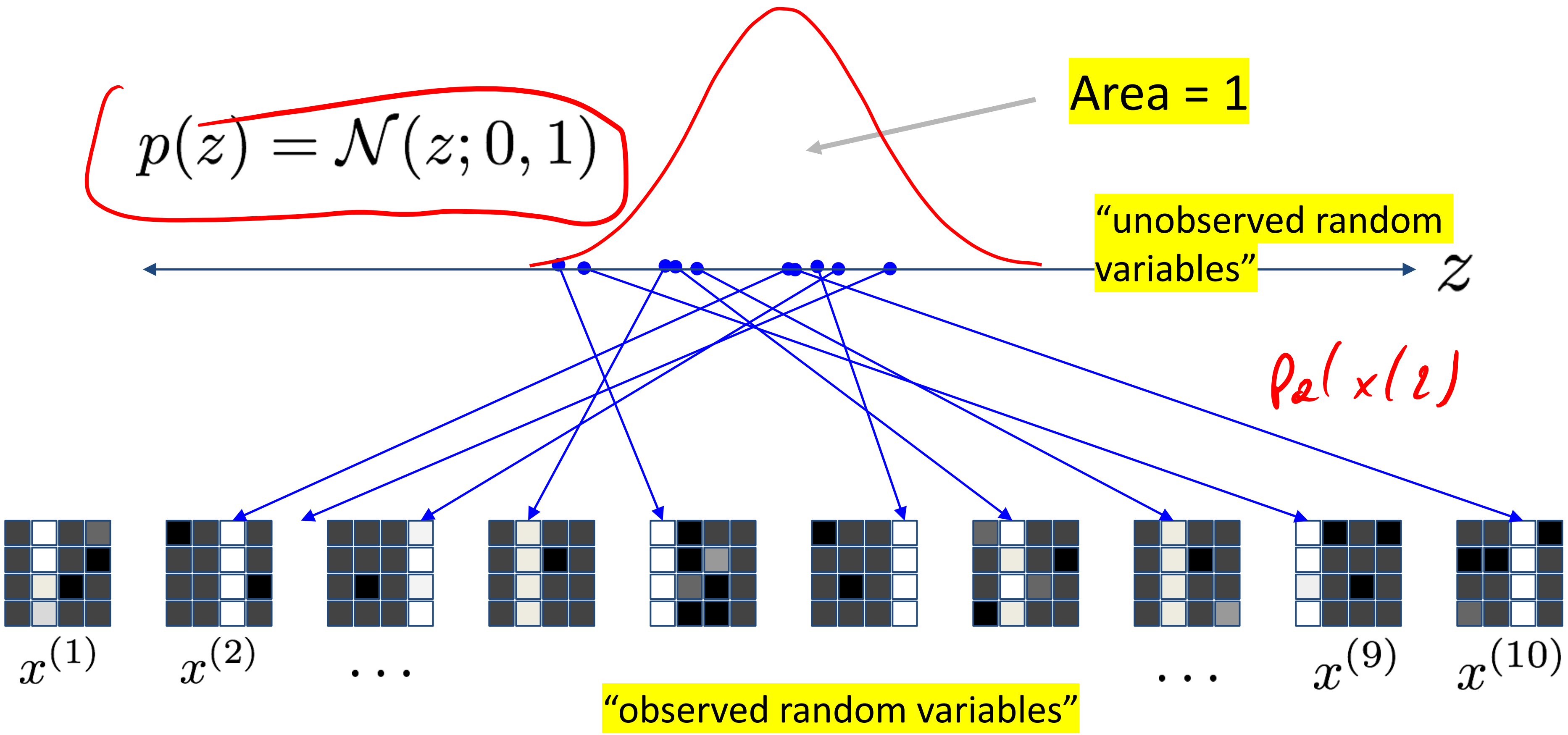
$$p_\Theta(z|x) = \frac{p_\Theta(x|z)p(z)}{\int dz' p_\Theta(x|z')p(z')}$$

$p_\theta(x)$

train = max $p_\theta(x)$

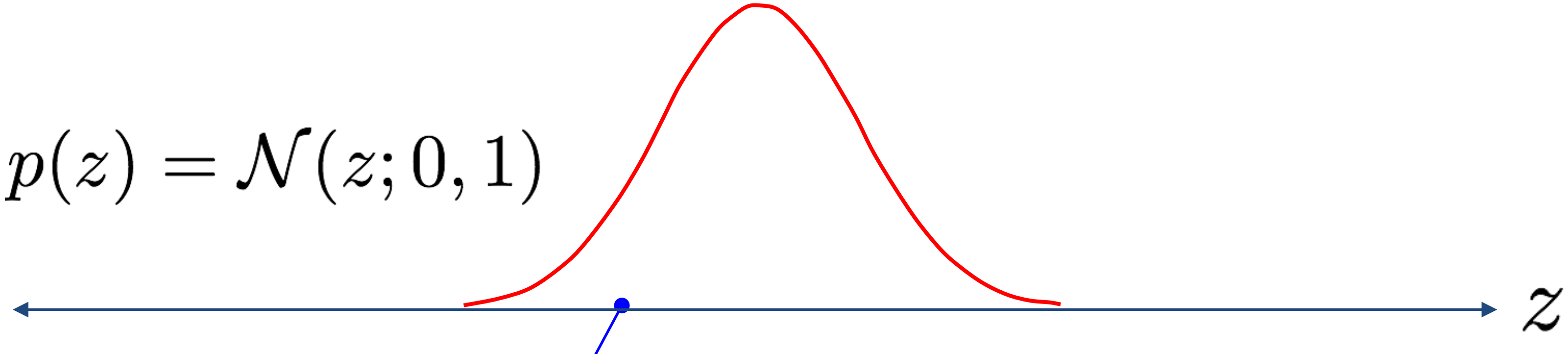
But often it's intractable ☹

Inference starts with priors



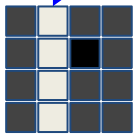
Inference

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

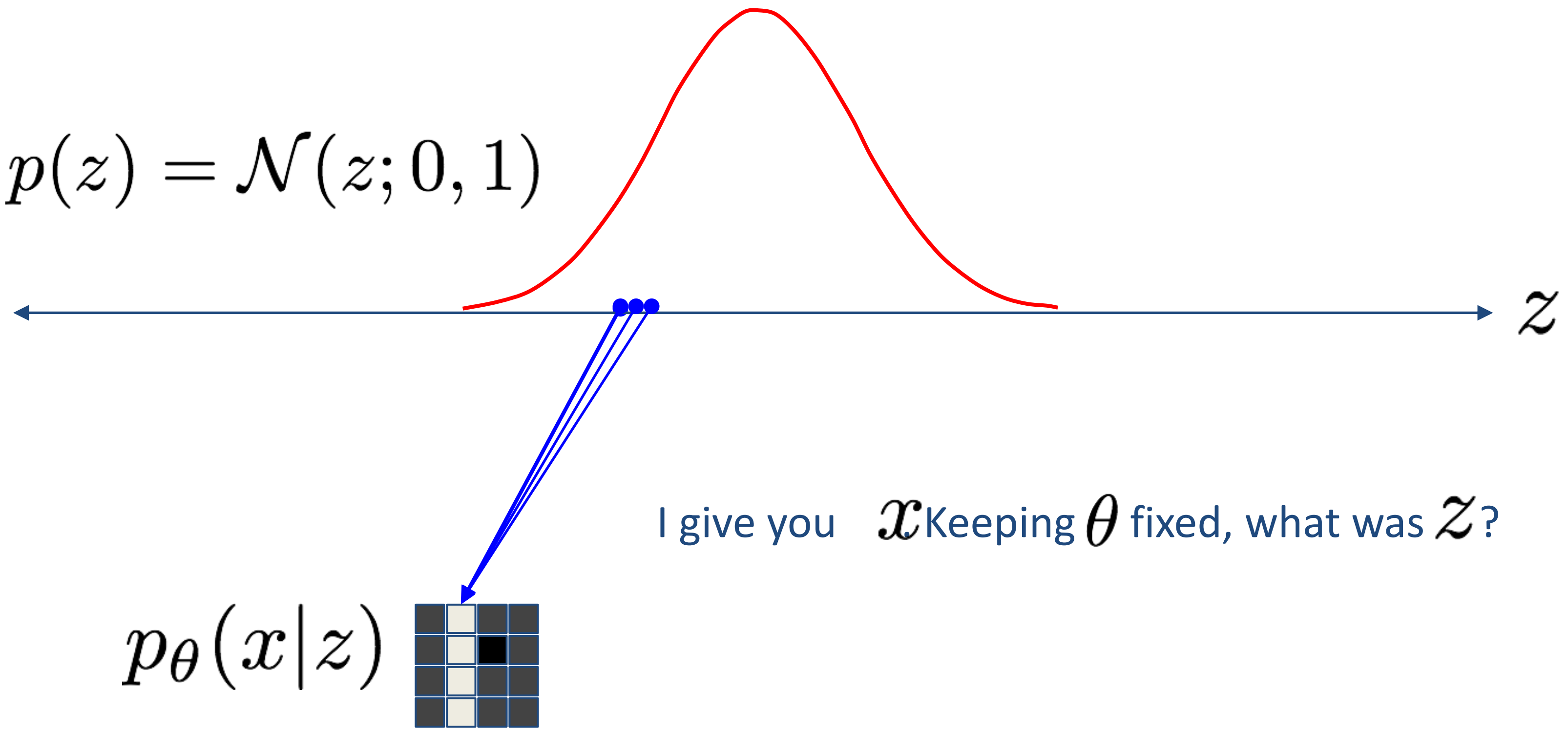


I give you x Keeping θ fixed, what was z?

$$p_\theta(x|z)$$

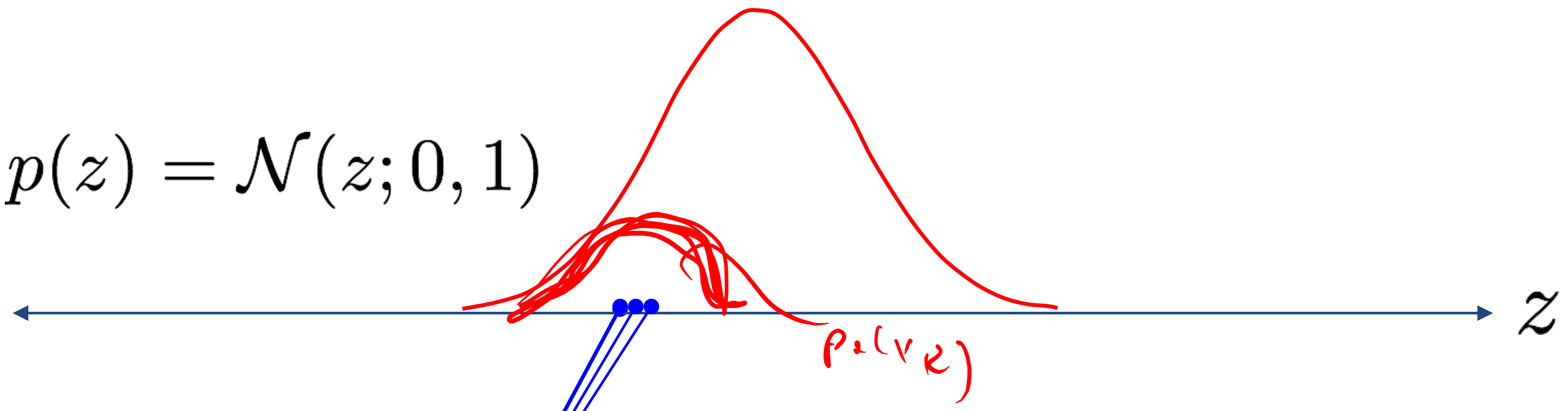


Inference


$p(z) = \mathcal{N}(z; 0, 1)$
z
I give you x Keeping θ fixed, what was z?
$p_\theta(x|z)$

Exercise

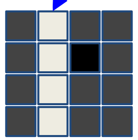
$$p(z) = \mathcal{N}(z; 0, 1)$$



Assuming the largest value of $p_\theta(x|z)$ is 1, draw

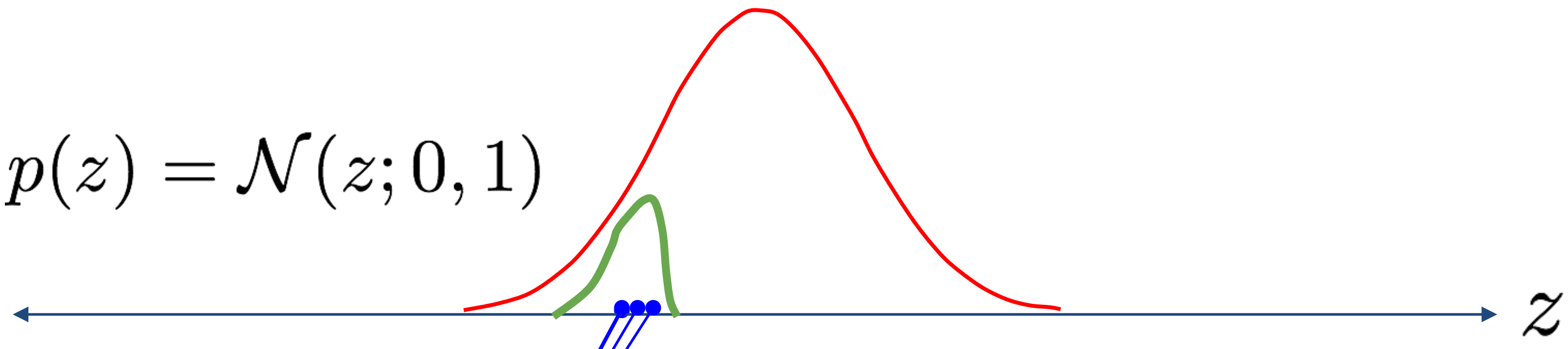
$$p_\theta(x, z) = p_\theta(x|z)\, p(z)$$

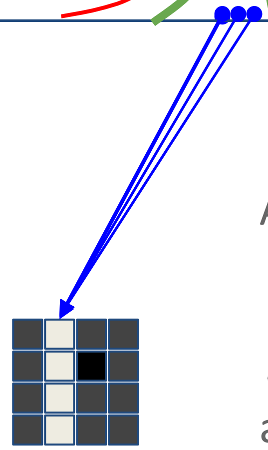
as a function of z on the same axis as above

$p_\theta(x|z)$ 

Joint density (with x observed)

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





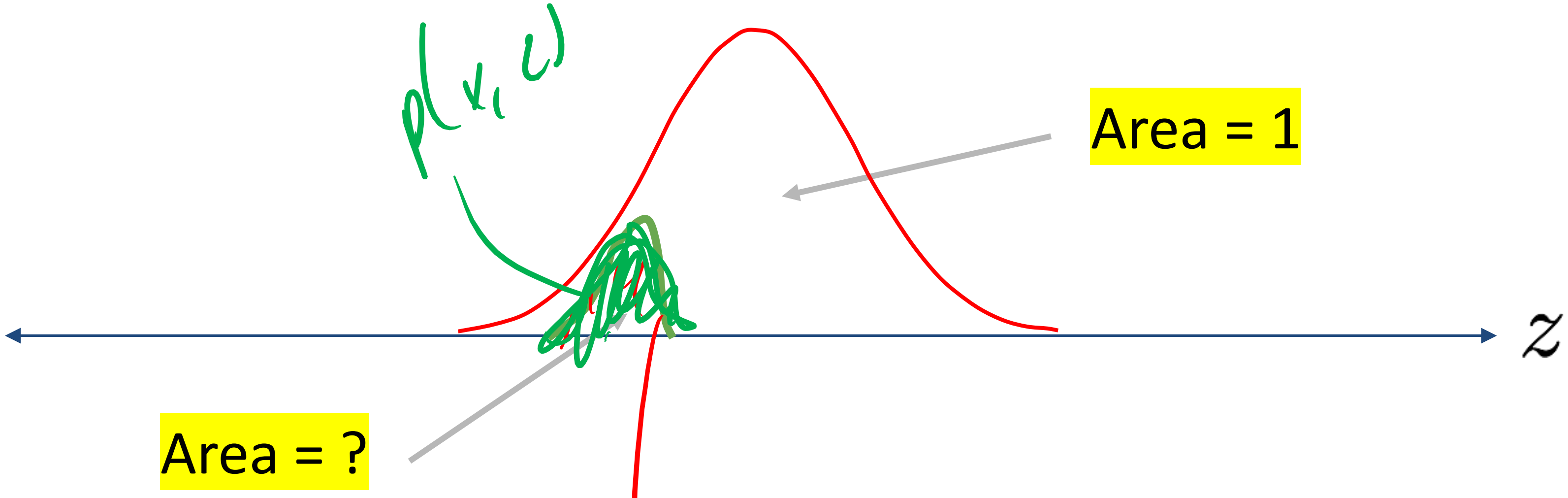
$p_\theta(x|z)$

Assuming the largest value of $p_\theta(x|z)$ is 1, draw

$$p_\theta(x, z) = p_\theta(x|z)\, p(z)$$

as a function of z on the same axis as above

Joint density (with x observed)



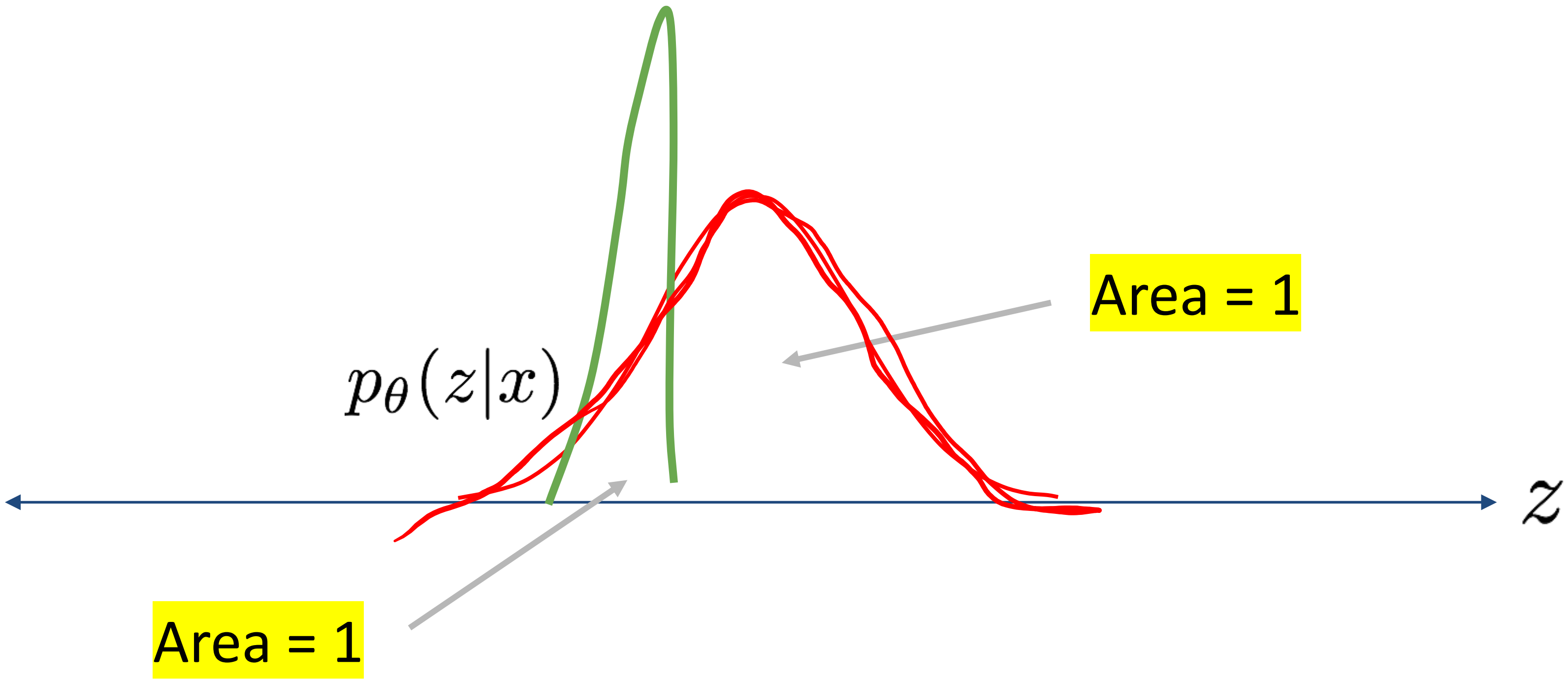
$p(x)$

1-minute exercise:
what is the area?

$p_\theta(x, z) = p_\theta(x|z)\, p(z)$

$\int_z p_\theta(x, z)\, dz = p_\theta(x) =$ denominator in Bayes' rule

Posterior



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

Dividing by the marginal likelihood (evidence) scales the area back to 1...

Evidence of all data points

Area for data point *n*

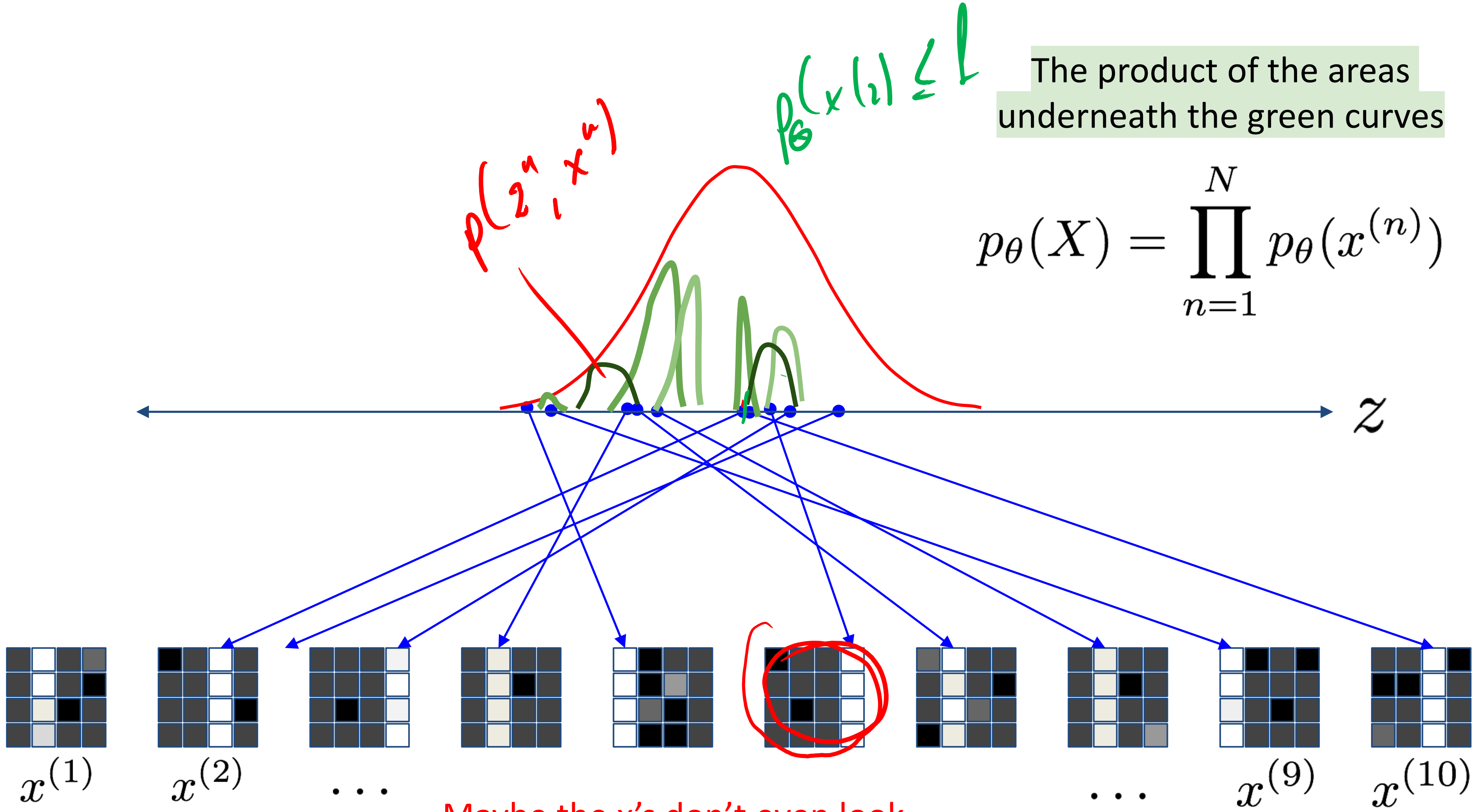
$$p_\theta(X) = \prod_{n=1}^{N} p_\theta(x^{(n)})$$

$$\log p_\theta(X) = \sum_{n=1}^{N} \log p_\theta(x^{(n)})$$

Evidence for all data points

The product of the areas underneath the green curves

$$p_\theta(X) = \prod_{n=1}^{N} p_\theta(x^{(n)})$$

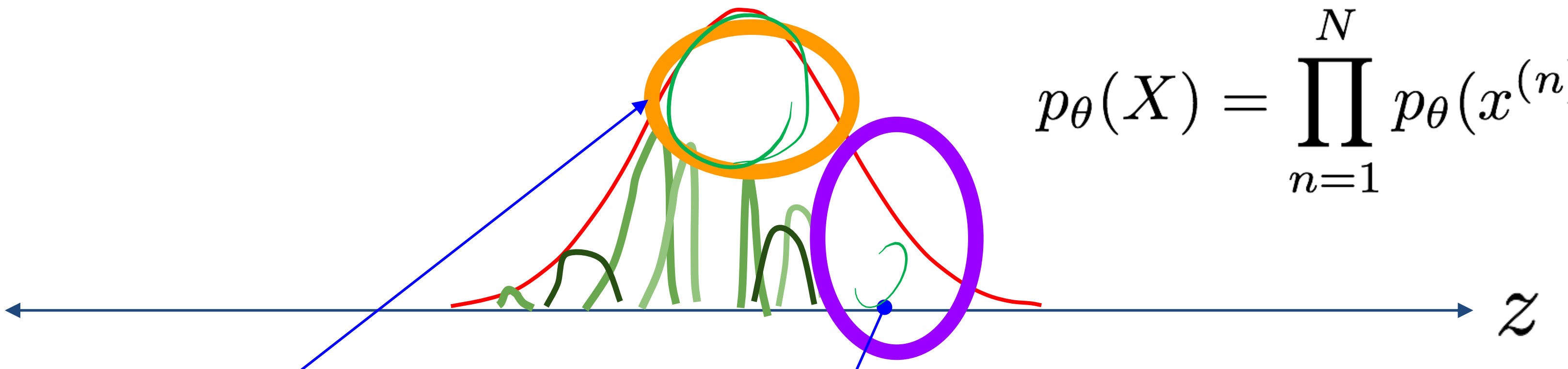


Maybe the x's don't even look that nice, when sampled with

Maximizing the evidence

The product of the areas underneath the green curves

$$p_\theta(X) = \prod_{n=1}^{N} p_\theta(x^{(n)})$$



By changing θ we can make the evidence for these data points bigger...

These z's don't generate images like the ones in the data set...

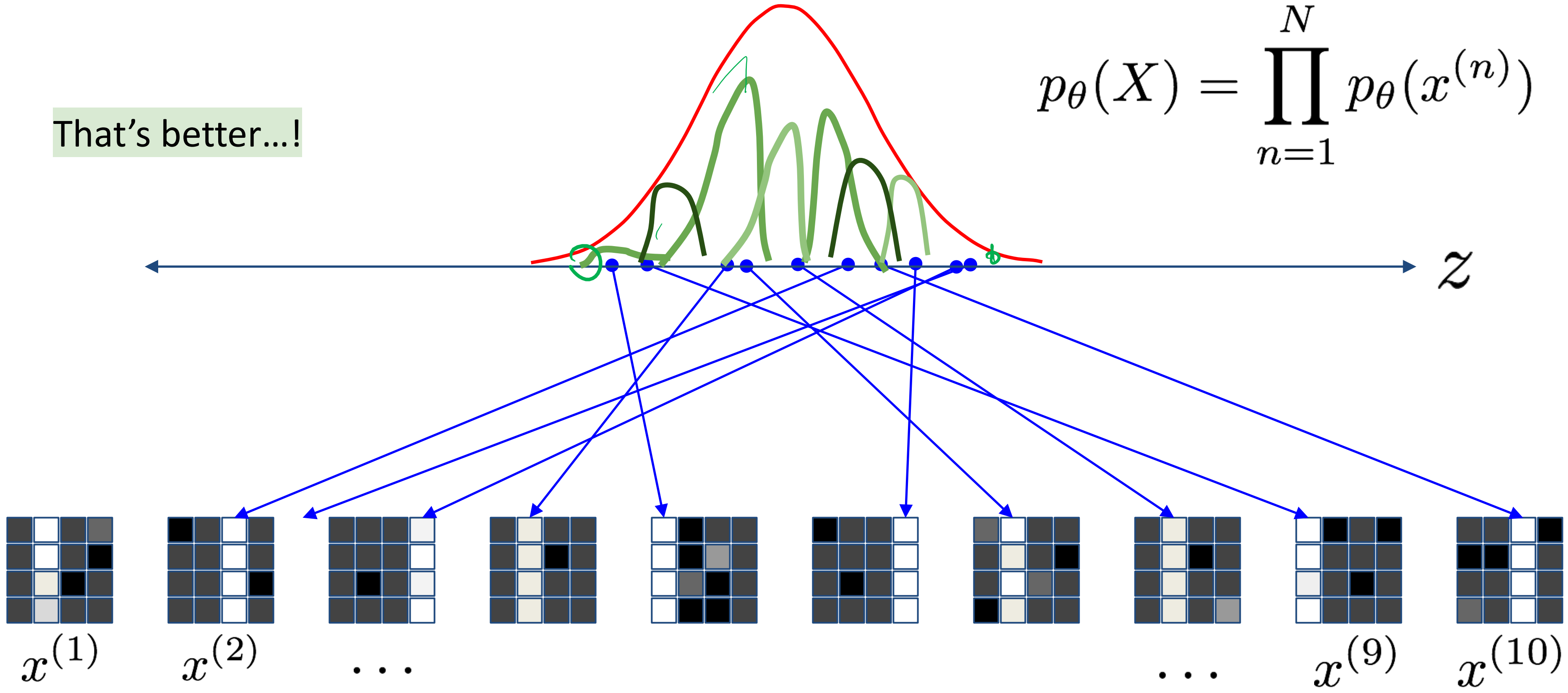
(With this θ, the prior doesn't capture the data manifold well)

Maximizing the evidence

That's better…!

The product of the areas underneath the green curves

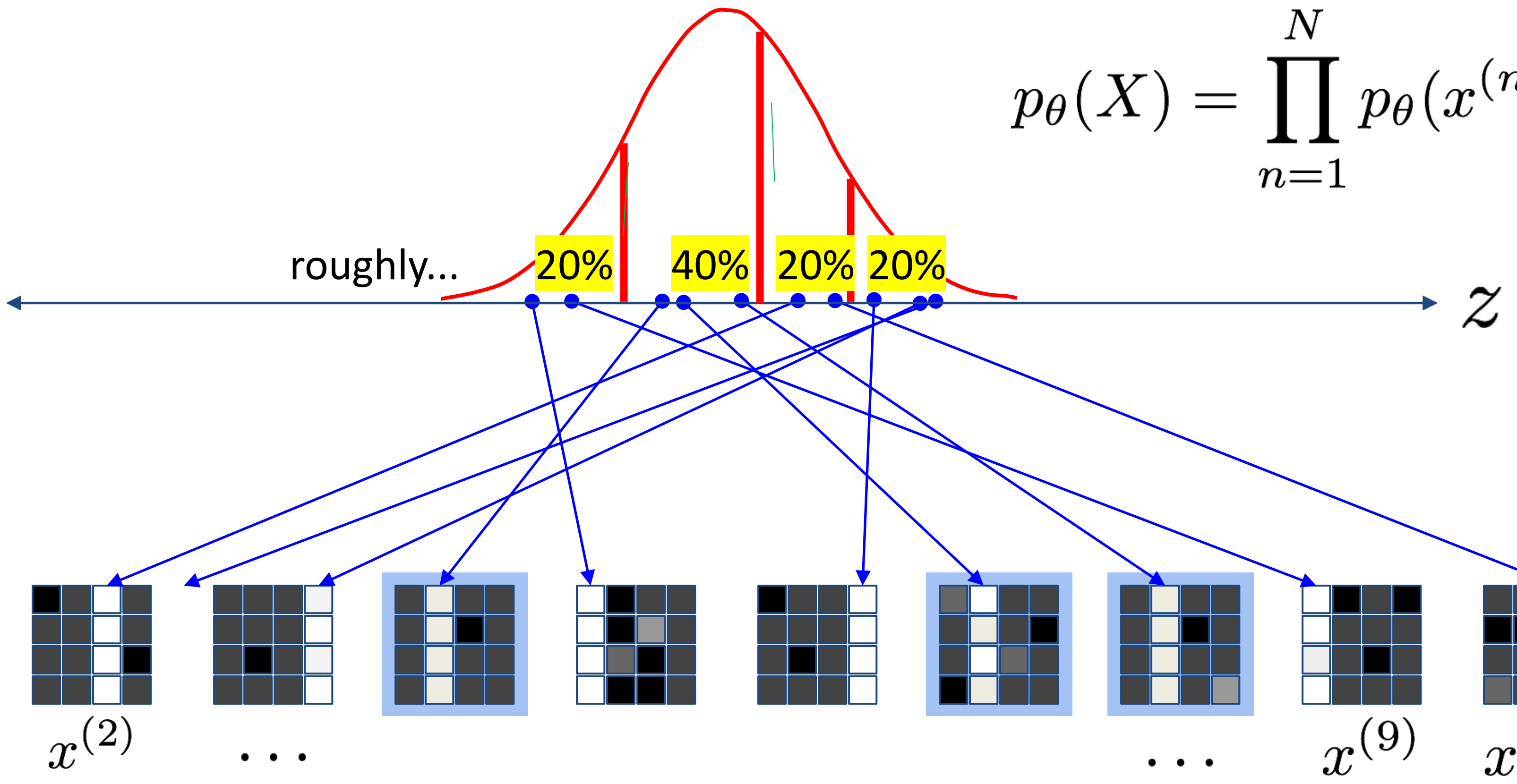
$$p_\theta(X) = \prod_{n=1}^{N} p_\theta(x^{(n)})$$



For the sharp-sighted

The product of the areas
underneath the green curves

$$p_\theta(X) = \prod_{n=1}^{N} p_\theta(x^{(n)})$$



Generation and Learning

Generation:

$$p(z)$$
$$p_\Theta(x|z)$$

Training by max log-likelihood

$$\arg\max_\Theta \log p_\Theta(x)$$

But

$$p_\Theta(x) = \int dz\, p_\Theta(x|z) p(z)$$

Approximate likelihood optimization

Our approach:

- Lower bound $\log p_\Theta(x)$
- Push the lower-bound up…

  … hoping to increase $\log p_\Theta(x)$

E - M

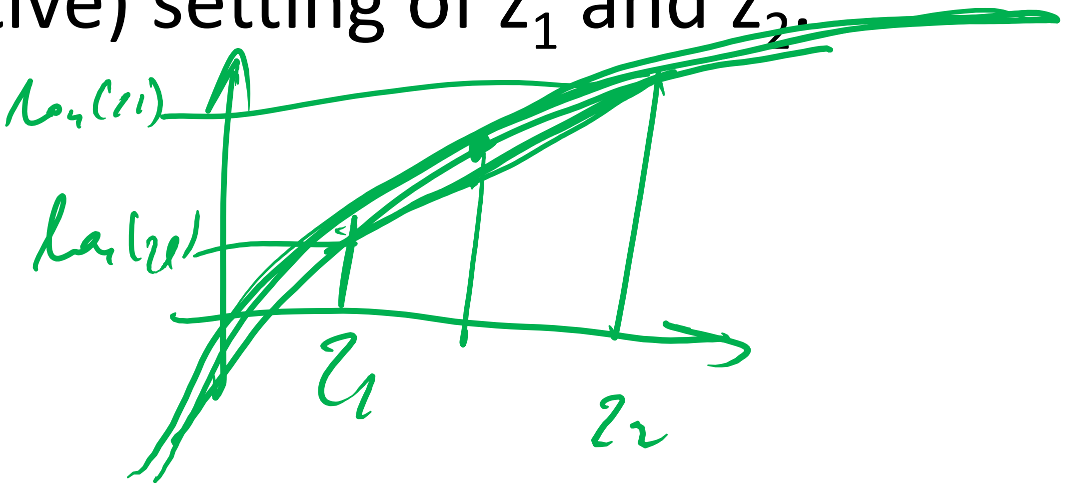
Exercise

Jensen's inequality

Draw log(...) as a function, convince yourself that

$$\log\left(\frac{2}{3}z_1 + \frac{1}{3}z_2\right) \geq \frac{2}{3}\log z_1 + \frac{1}{3}\log z_2$$

is true for any (nonnegative) setting of z_1 and z_2.



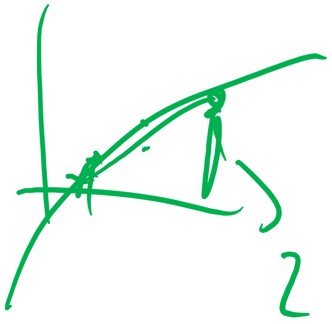
ELBO: A likelihood bound

$$\log p_\Theta(x) = \log \int dz\, p_\Theta(x,z) =$$

$$= \log \int dz\, q_\Phi(z|x) \frac{p_\Theta(x,z)}{q_\Phi(z|x)}$$

works for ANY Q

st $Q(z|x) \neq 0$



$$\geq \int dz\, q_\Phi(z|x) \log \frac{p_\Theta(x,z)}{q_\Phi(z|x)}$$

Jensen

$$= \mathbb{E}_{q_\Phi(z|x)}\left[\log \frac{p_\Theta(x|z)p(z)}{q_\Phi(z|x)}\right]$$

$$= \mathbb{E}_{q_\Phi(z|x)}[\log p_\Theta(x|z)]$$

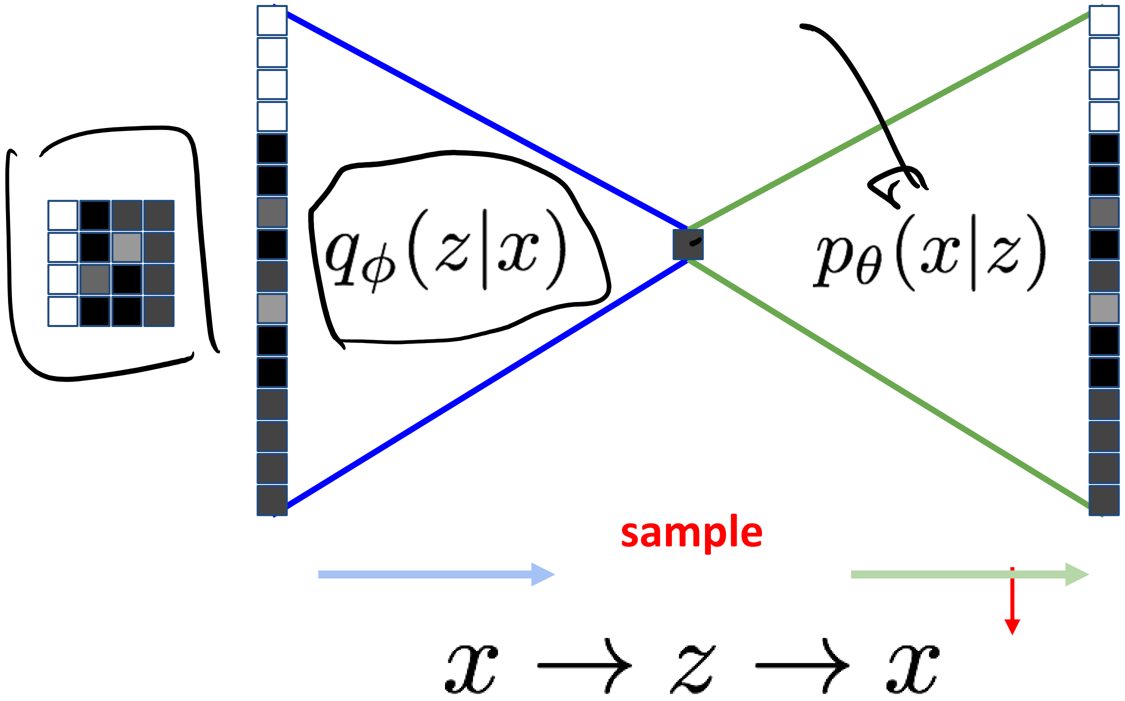
$$- \mathbb{E}_{q_\Phi(z|x)}\left[\log \frac{q_\Phi(z|x)}{p(z)}\right]$$

$$= \mathbb{E}_{q_\Phi(z|x)}[\log p_\Theta(x|z)] - KL(q_\Phi(z|x) \parallel p(z))$$

ELBO interpretation

$$\log p_\Theta(x) \geq \mathbb{E}_{q_\Phi(z|x)}[\log p_\Theta(x|z)] - KL\big(q_\Phi(z|x) \parallel p(z)\big)$$

$\mathbb{E}_{q_\Phi(z|x)}[\log p_\Theta(x|z)]$: auto-encoding term!



ELBO optimization

$$\log p_{\Theta}(x) \geq \mathbb{E}_{q_{\Phi}(z|x)}[\log p_{\Theta}(x|z)] - KL\big(q_{\Phi}(z|x) \parallel p(z)\big)$$

ELBO is a function of x, Θ, and Φ

What it means to maximize ELBO over Φ?

It can't change $\log p_{\Theta}(x)$…

It tries to make the bound tight!

X

$\downarrow$ guess over z

z

$\downarrow$ reconstruct x

$q(z|x)$

Exercise

Recall Jensen's inequality:

$$\log \int dz\, q(z) f(z) \geq \int dz q(z) \log f(z)$$

When is it an **equality**?

When $f(z) = \text{const}$

When is ELBO tight?

$$\log p_\Theta(x) = \log \int dz\, q_\Phi(z|x) \frac{p_\Theta(x,z)}{q_\Phi(z|x)}$$

$$\geq \int dz q_\Phi(z|x) \log \frac{p_\Theta(x,z)}{q_\Phi(z|x)} = ELBO$$

When $\frac{p_\Theta(x,z)}{q_\Phi(z|x)} = \text{const!}$

What does it mean?

$$\frac{p_\Theta(x,z)}{q_\Phi(z|x)} = \frac{p_\Theta(z|x)p(x)}{q_\Phi(z|x)} = \text{const} \Rightarrow \boxed{p_\Theta(x|z) = q_\Phi(z|x)}$$

ELBO is tight when $q_\Phi(z|x)$ does exact inference!

ELBO optimization

$$\log p_\Theta(x) \geq \mathbb{E}_{q_\Phi(z|x)}[\log p_\Theta(x|z)] - \cancel{KL\big(q_\Phi(z|x) \parallel p(z)\big)}$$

ELBO is a function of x, Θ, and Φ

What it means to maximize ELBO over Θ?

Can only affect $\mathbb{E}_{q_\Phi(z|x)}[p_\Theta(x|z)]$!

Makes $p_\Theta(\mathrm{x}|\mathrm{z})$ generate back our x!
This affects $\log p_\Theta(x)$…
…making room for improving q!

ELBO optimization

$$\log p_\Theta(x) \geq \mathbb{E}_{q_\Phi(z|x)}[\log p_\Theta(x|z)] - KL\big(q_\Phi(z|x) \parallel p(z)\big)$$

Change Φ to maximize the bound, making $q_\Phi(z|x) \approx p_\Theta(z|x)$

E

Similar to E step

Change Θ to (if bound sufficiently tight) improve $\log p_\Theta(x)$

M

Similar to M step

But we tune Φ and Θ at the same time!

ELBO interpretation

Bottleneck

ELBO, or evidence lower bound:

$$\log p(x) \geq \mathbb{E}_{z\sim q_\Phi(z|x)}[\log p_\Theta(x|z)] - KL\left(q_\Phi(z|x) \parallel p(z)\right)$$

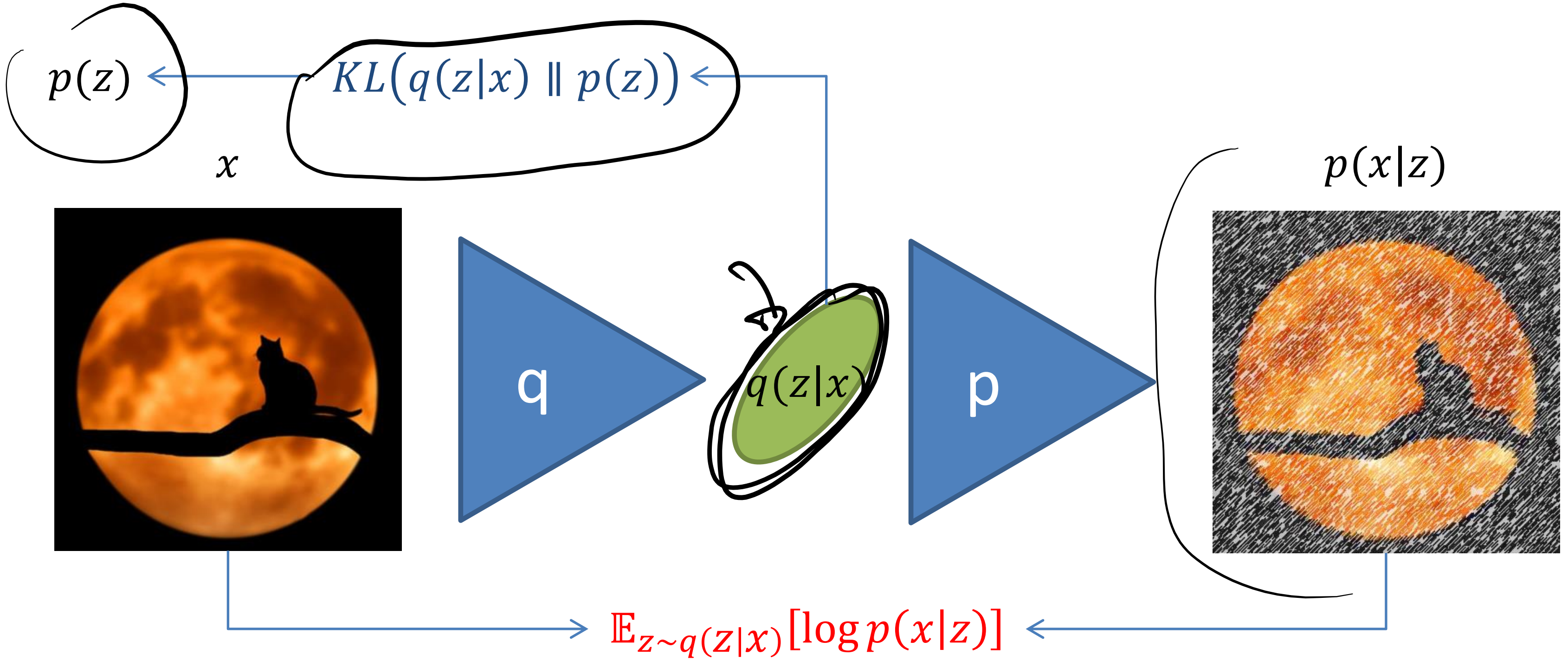
where:

$\mathbb{E}_{z\sim q_\Phi(z|x)}[\log p_\Theta(x|z)]$ reconstruction quality:
how many nats we need to reconstruct x,
when someone gives us $q(z|x)$

$KL\left(q_\Phi(z|x) \parallel p(z)\right)$ code transmission cost:
how many nats we transmit about x in $q_\Phi(z|x)$ rather than $p(z)$

Interpretation: do well at reconstructing x, limiting the amount of information about x encoded in z.

The Variational Autoencoder



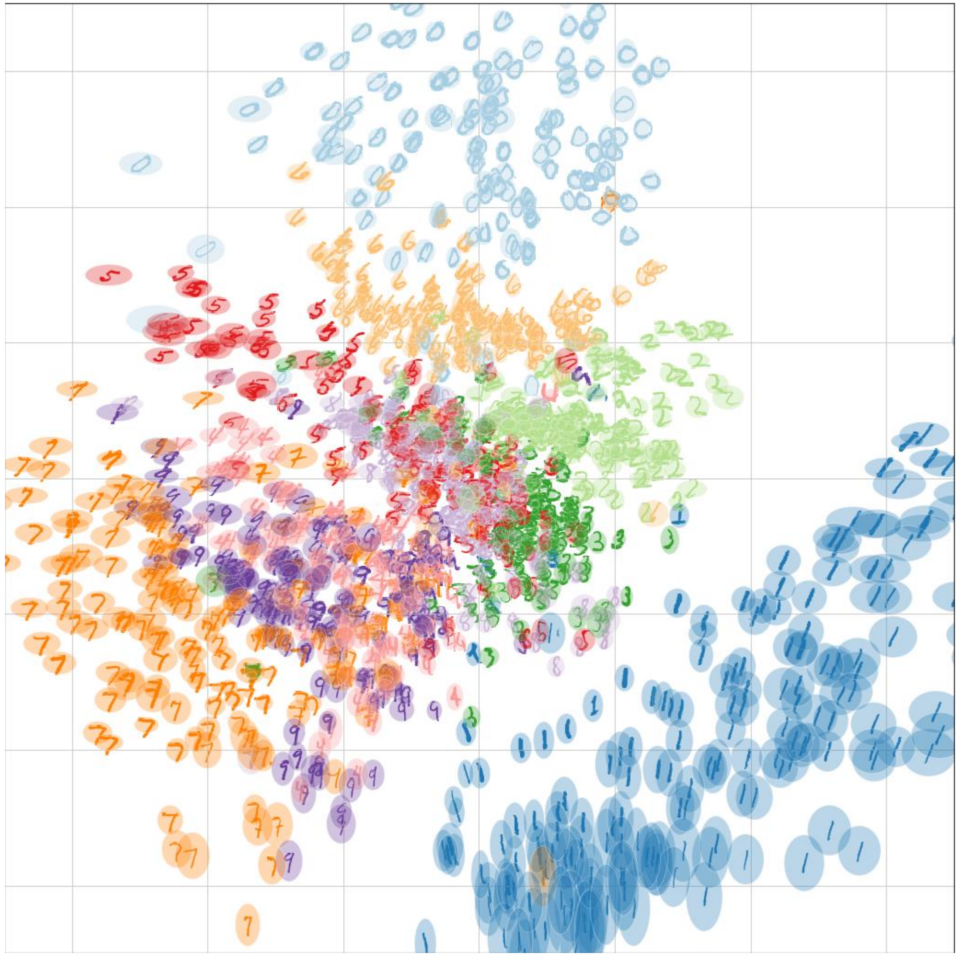
An input x is put through the q network to obtain a distribution over latent code z, $q(z|x)$.

Samples $z_1, \dots, z_k$ are drawn from $q(z|x)$. They k reconstructions $p(x|z_k)$ are computed using the network p.

VAE is an Information Bottleneck

Each sample is represented as a Gaussian

This discards information (latent representation has low precision)



How to evaluate a VAE

Compute:

$$\log p(x) \geq \mathbb{E}_{z \sim q_\Phi(z|x)}[\log p_\Theta(x|z)] - KL\big(q_\Phi(z|x) \parallel p(z)\big)$$

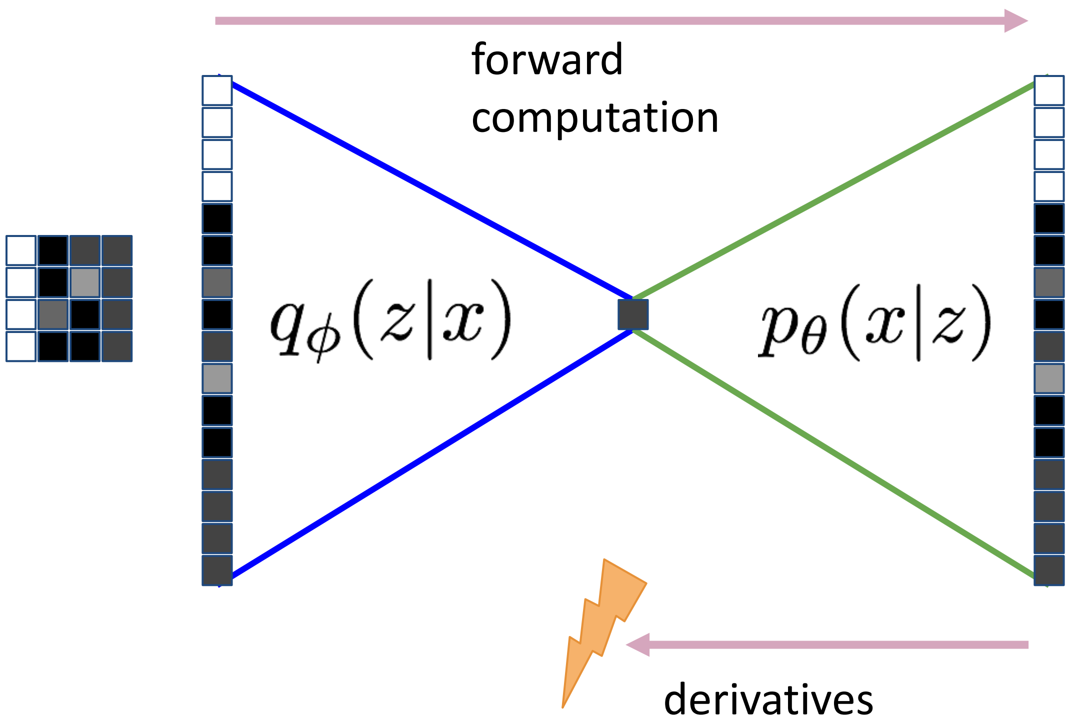
$KL\big(q_\Phi(z|x) \parallel p(z)\big)$ has closed form for simple $q_\Phi(z|x)$

$\mathbb{E}_{z \sim q_\Phi(z|x)}[\log p_\Theta(x|z)]$ can be approximated:

$$\mathbb{E}_{z \sim q_\Phi(z|x)}[\log p_\Theta(x|z)] \approx \sum_i \log p_\Theta(x|z_i)$$

Where z_i drawn from $q_\Phi(z|x)$

How to train a VAE?



- $\log p(x) \geq \mathbb{E}_{z \sim q_\Phi(z|x)}[\log p_\Theta(x|z)] - KL\big(q_\Phi(z|x) \parallel p(z)\big)$
- Forward computation involves drawing samples
- Can't backprop (get $\frac{\partial \log p(x)}{\partial \Phi}$) ☹

Reparameterization exercise

Assume that $q_\Phi(z|x) = \mathcal{N}(\mu_z, \sigma_z)$.

Exercise:
you can sample from $\mathcal{N}(0,1)$

Q: how to draw samples from $\mathcal{N}(\mu_z, \sigma_z)$

A:

$$\epsilon_i \sim \mathcal{N}(0,1)$$
$$z_i = \mu_z + \sigma_z \epsilon$$

Reparametrization to the rescue

Assume that $q_\Phi(z|x) = \mathcal{N}(\mathrm{\mu}_z, \sigma_z)$.

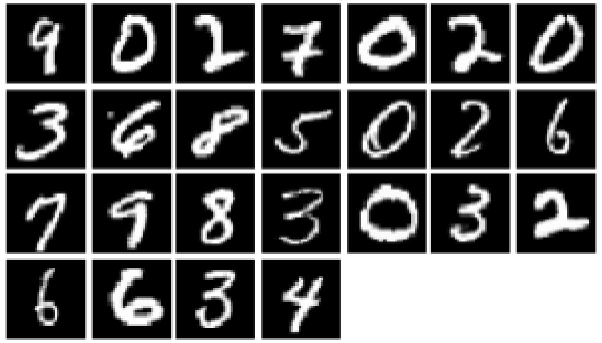
Then:

$$\mathbb{E}_{z \sim q_\Phi(z|x)}[\log p_\Theta(x|z)]$$
$$= \mathbb{E}_{\epsilon \sim \mathcal{N}(0,1)}[\log p_\Theta(x|\mu_z + \sigma_z \epsilon)]$$

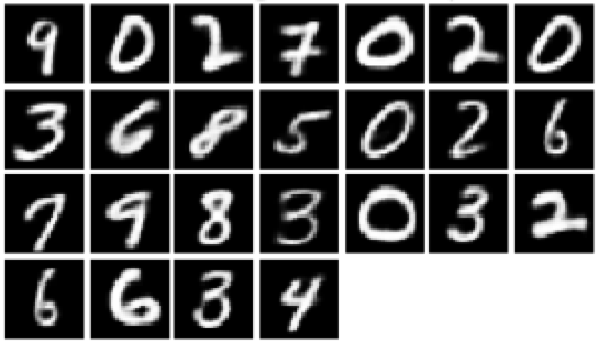
ϵ is drawn from a fixed distribution.
With ϵ given, the computaiton graph is deterministic -> we can backprop!

VAE in action

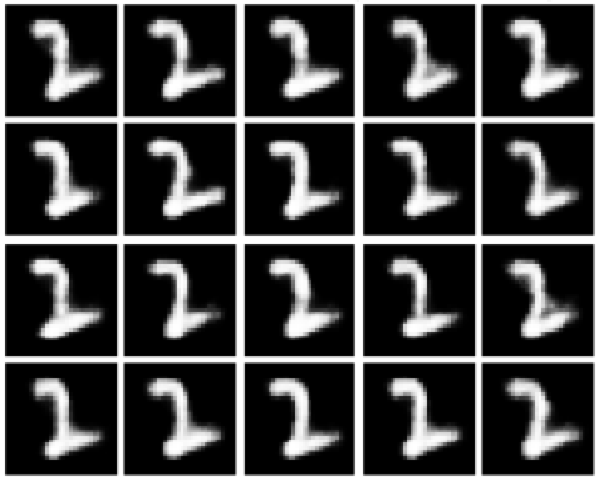
Data samples



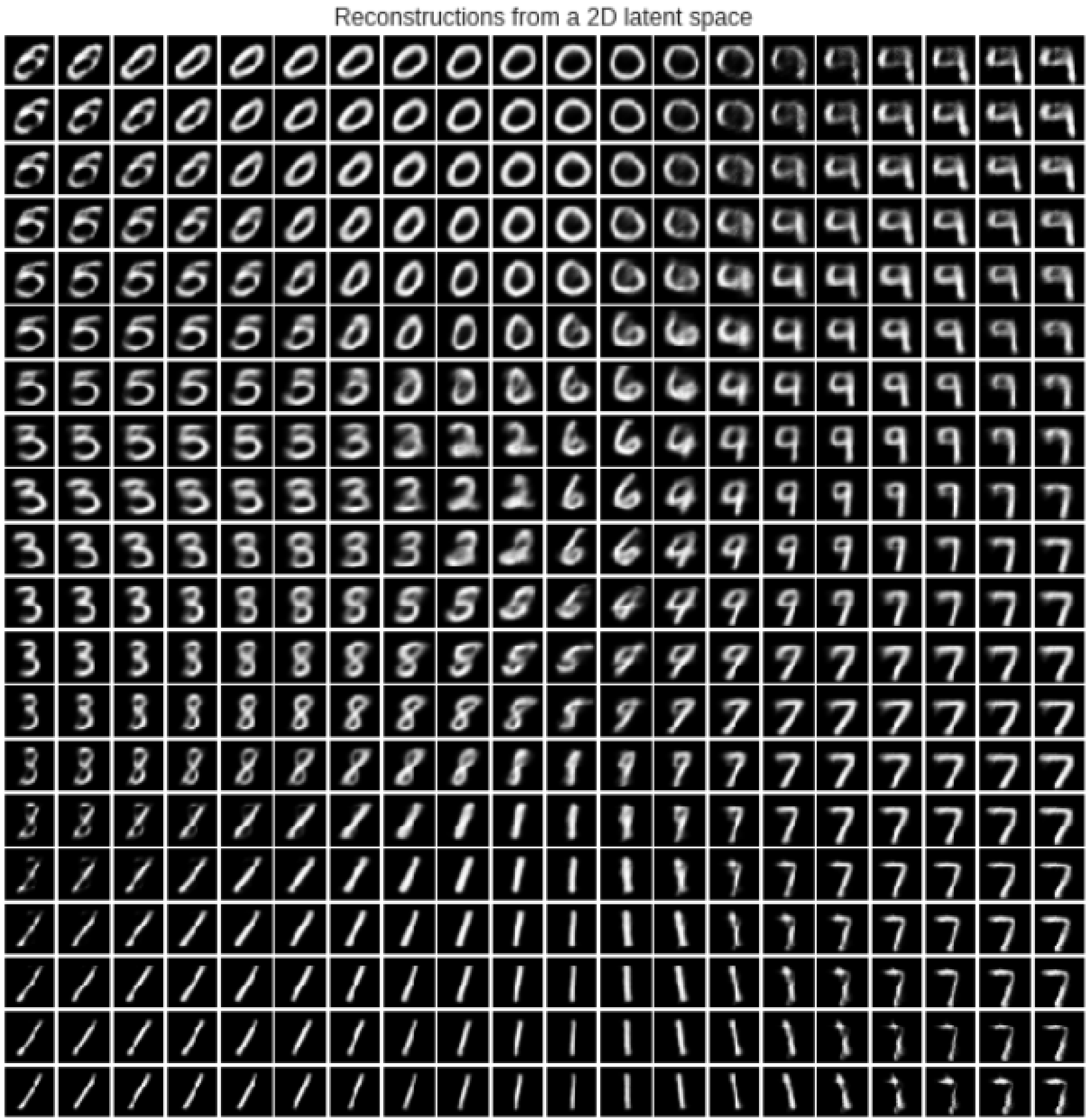
Reconstructions using the latent expected value

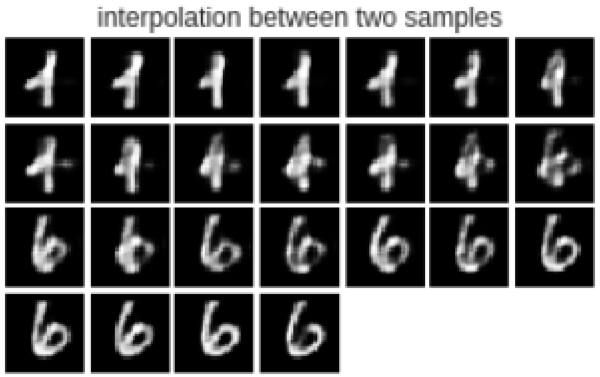


Reconstructions from multiple latent samples



VAE in action





VAE in action