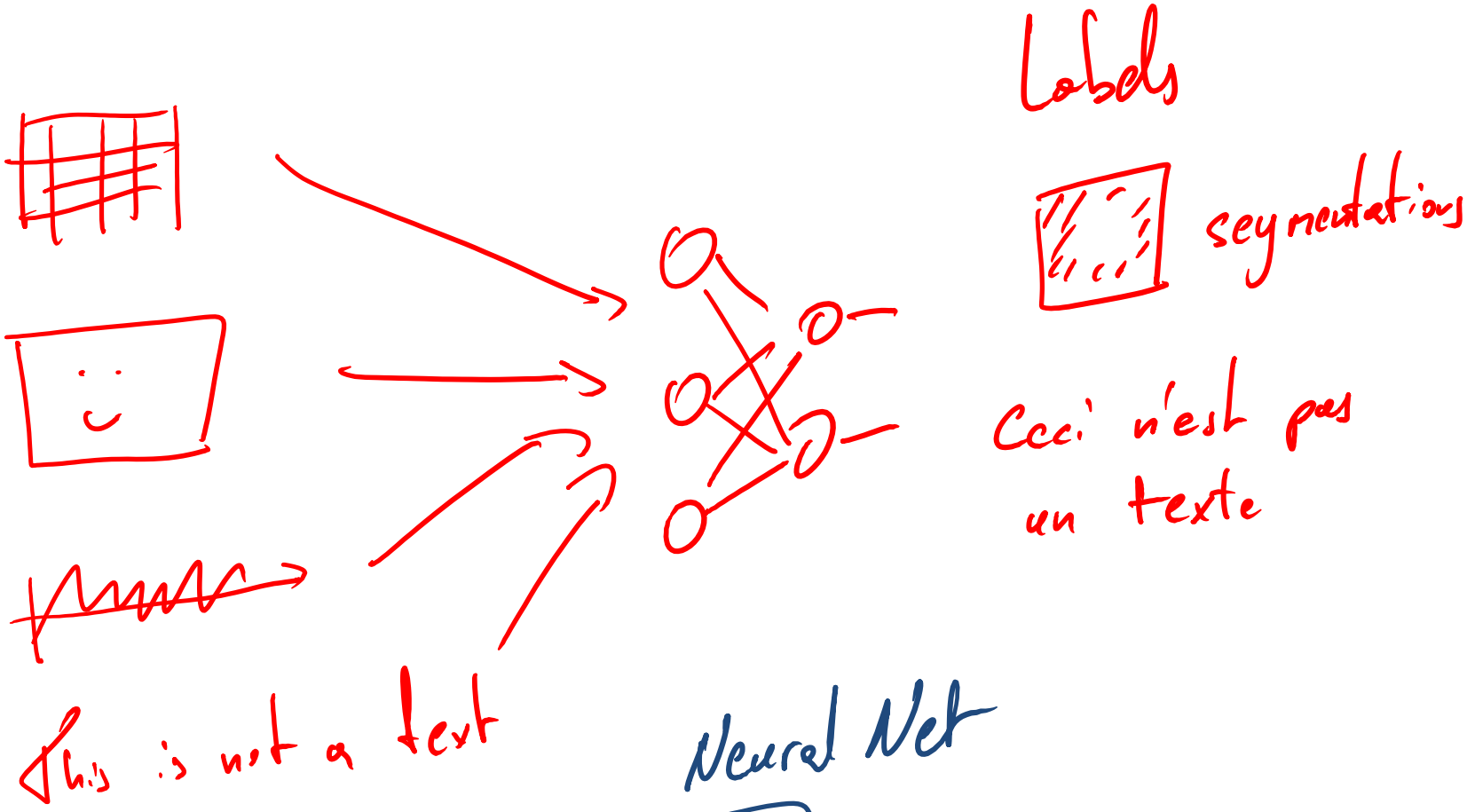
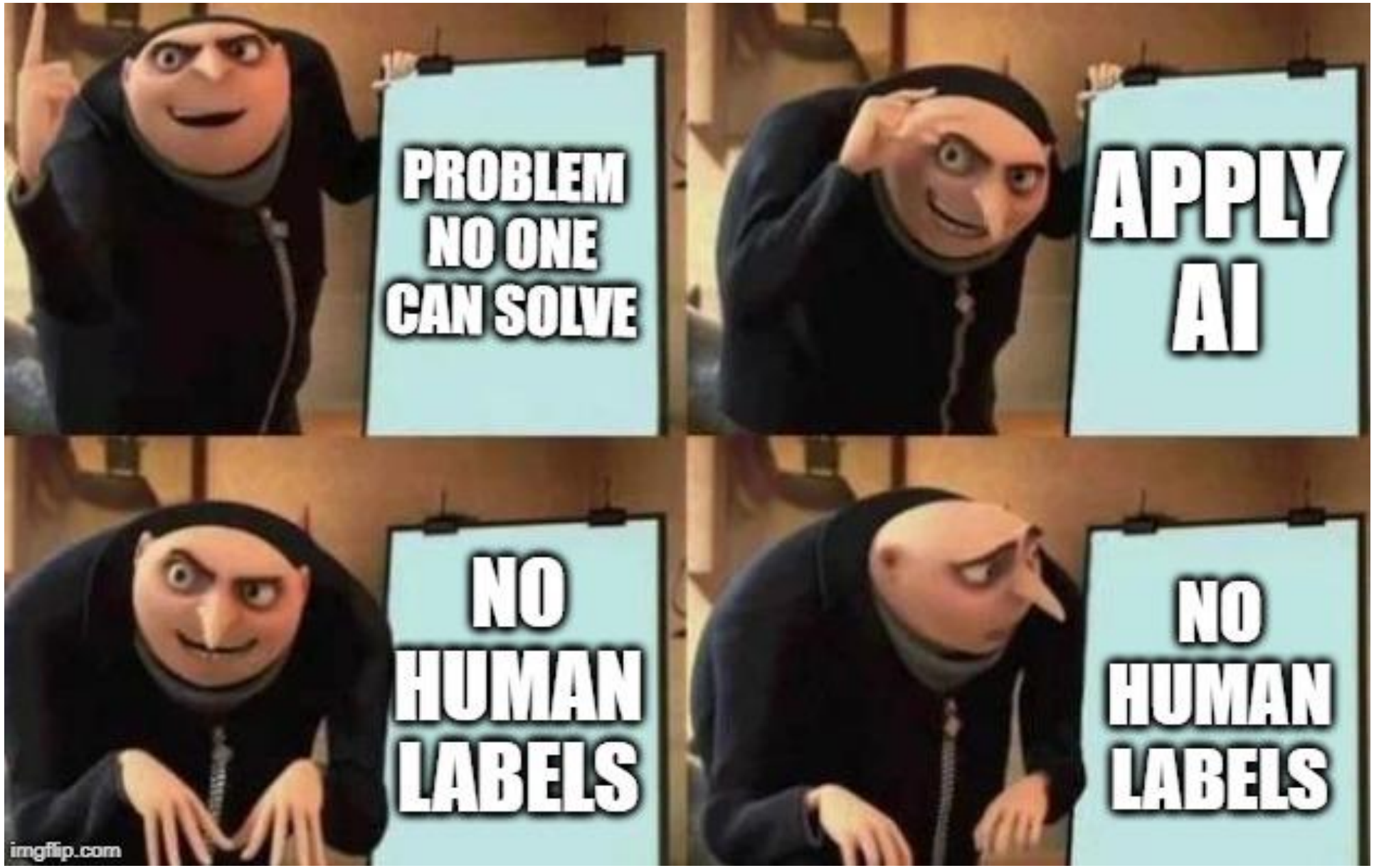
Deep Unsupervised Learning

Deep Learning @ UWr 2021

Jan Chorowski

Motivation: Deep Nets excel at supervised tasks




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

Unsupervised Learning Goals and Possibilities

What we may want:

- Learn $p(x)$
  Then use it to:
  - Generate new data
  - Enhance data (denoise, transform an input sample into a more plausible one)
- Learn a useful representation
  Then use it to:
  - Train models with little labels
  - Further analysis, e.g. clustering

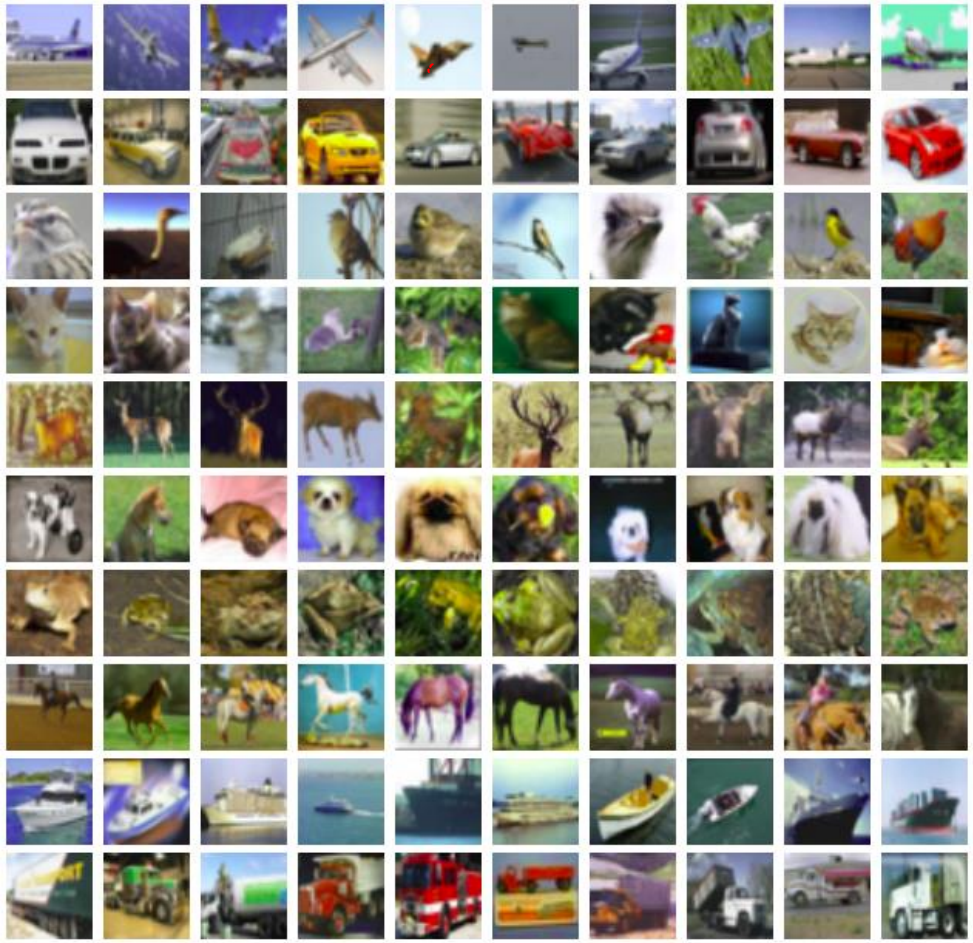
What losses we can use:

- Log-likelihood
- Learn to produce plausibly looking samples (as judged by an auxiliary network)
- Learn to encode $x \rightarrow z$ and decode $z \rightarrow x'$ keeping x' close to x
- Learn to solve a proxy task, for which labels are easy to get

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?

Autoregressive Models – Pixelrnn & Pixelcnn

$$p(x_1, x_2, \ldots x_n) = p(x_1)\, p(x_2 \mid x_1) \ldots p(x_n \mid x_1 \ldots x_{n-1})$$

AUTOREGRESSIVE MODELS

Density Estimation Using Autoregressive Models: Language modeling

- Let x be a sequence of word ids.
- $p(x) = p(x_1, x_2, \ldots, x_n) = \prod_i p(x_i | x_{<i})$

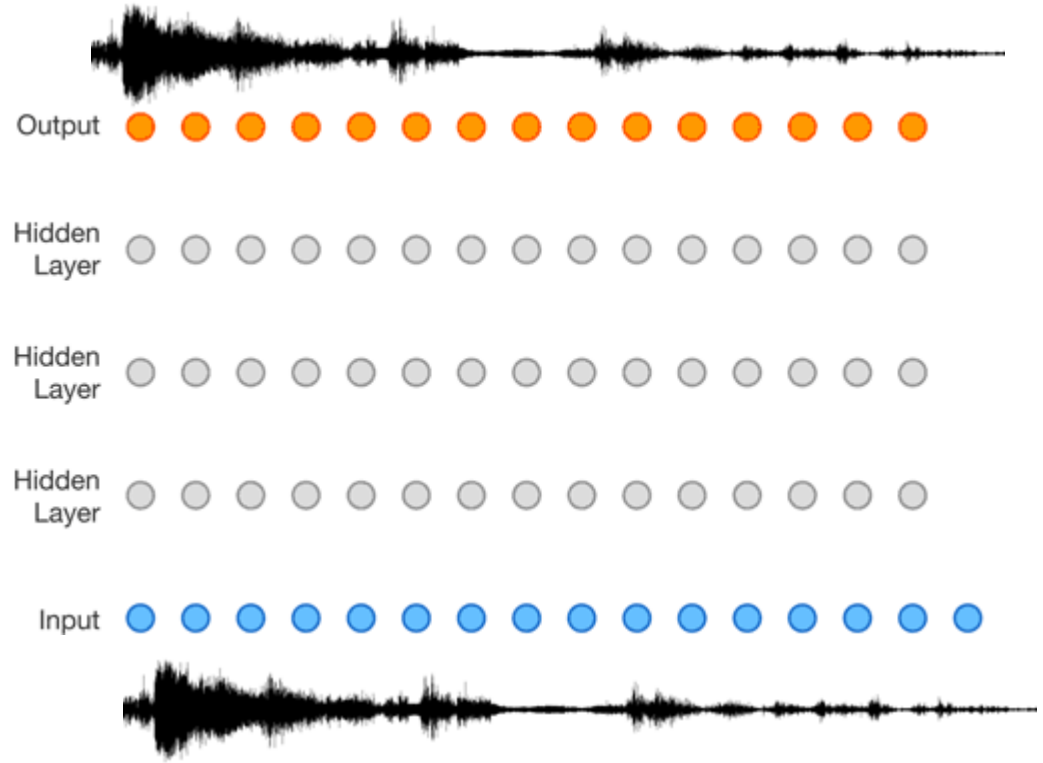
$$\approx \prod_i p(x_i | x_{i-k}, x_{i-k+1}, \ldots, x_{i-1})$$

- 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.



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).

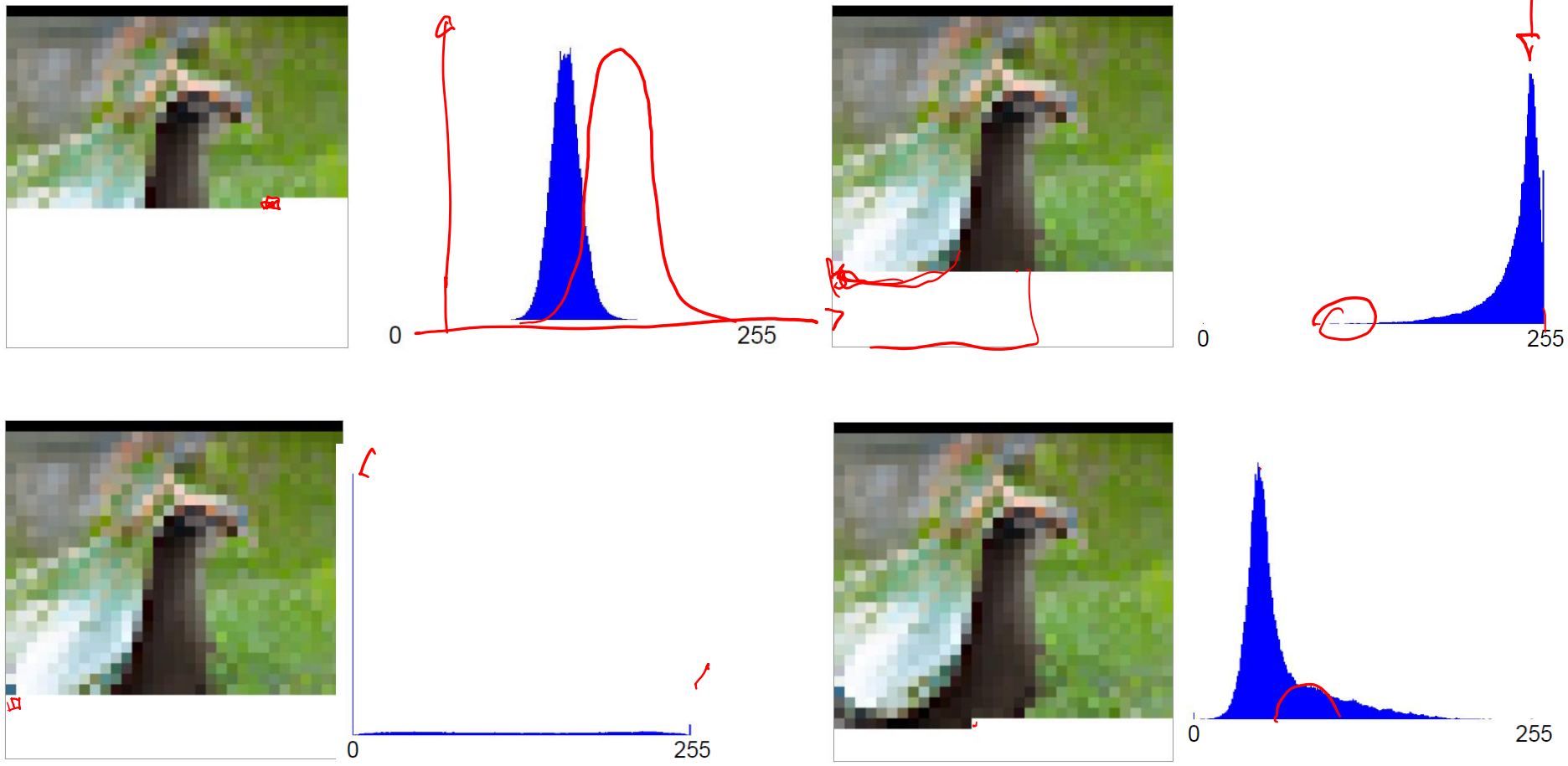
Modeling pixels

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]?

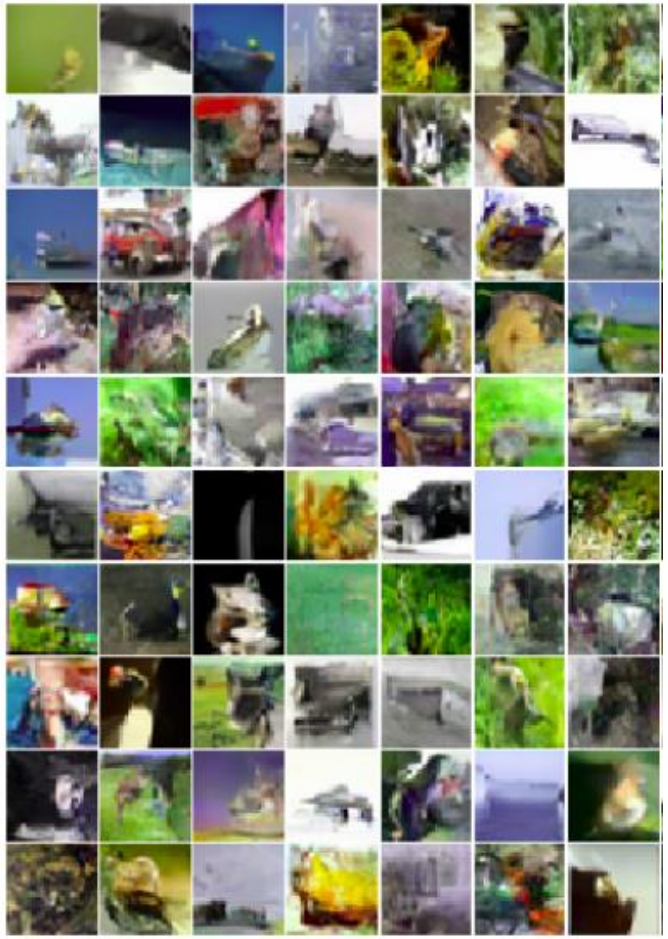
What are the implications?

Modeling pixel values



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

PixelCNN samples & completions

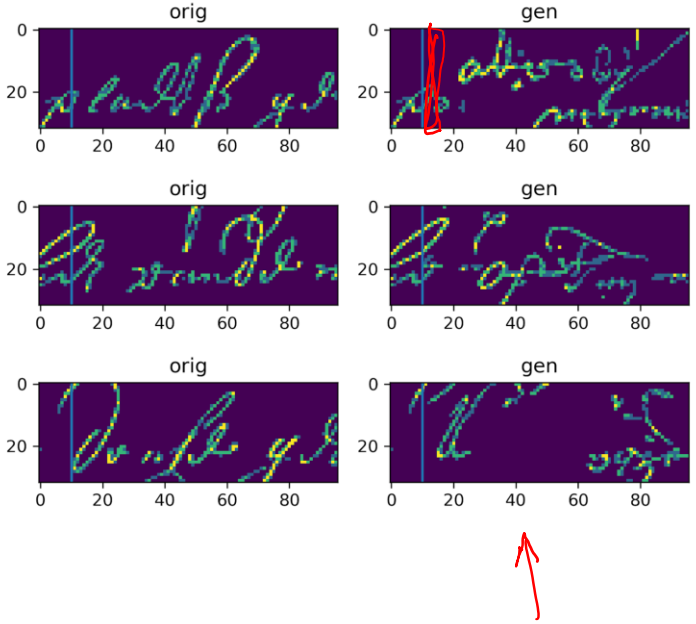


Salimans et al, “A PixelCNN Implementation with Discretized Logistic Mixture Likelihood and Other Modifications”

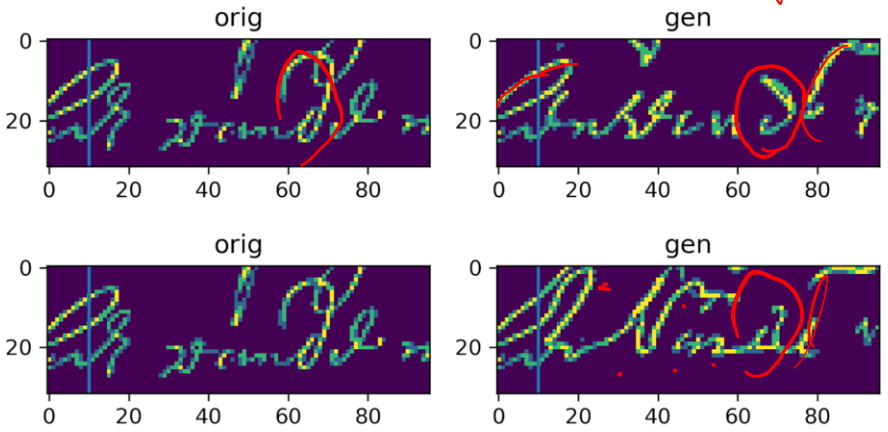
van den Oord, A., et al. “Pixel Recurrent Neural Networks.” ICML (2016).

PixelCNN samples

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 State-of-the-Art log-likelihood

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.

LATENT VARIABLE MODELS

Ad break

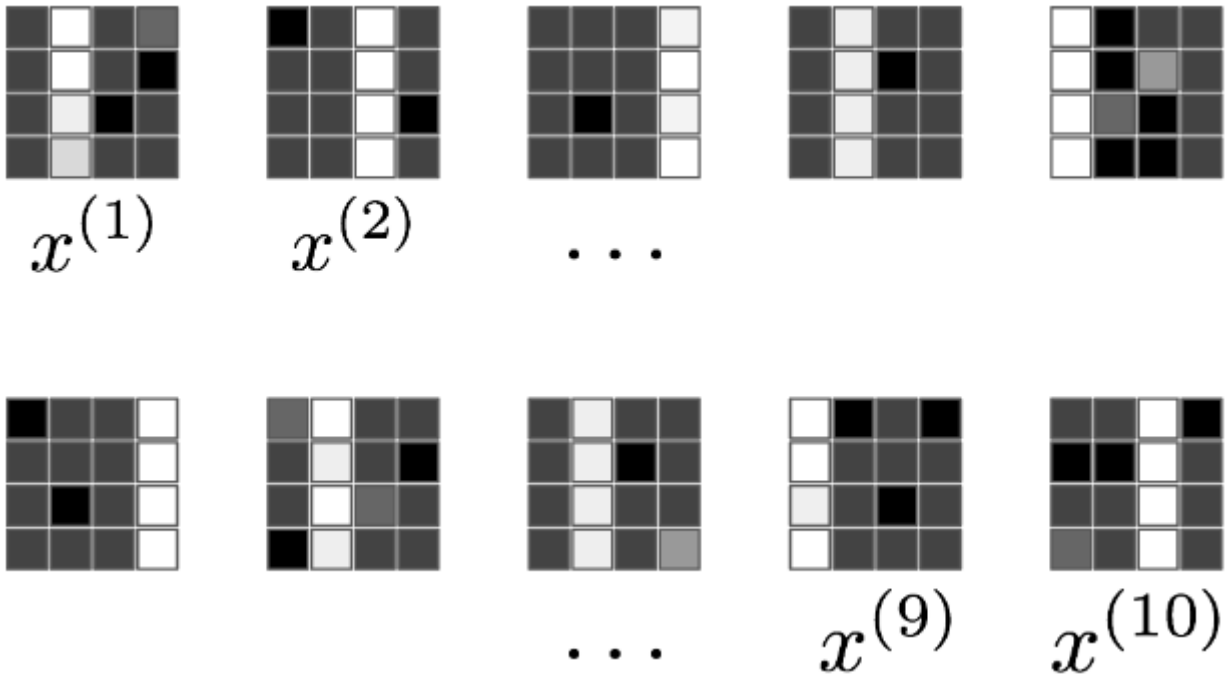
I got the many of following slides from Ulrich Paquet (DeepMind)

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

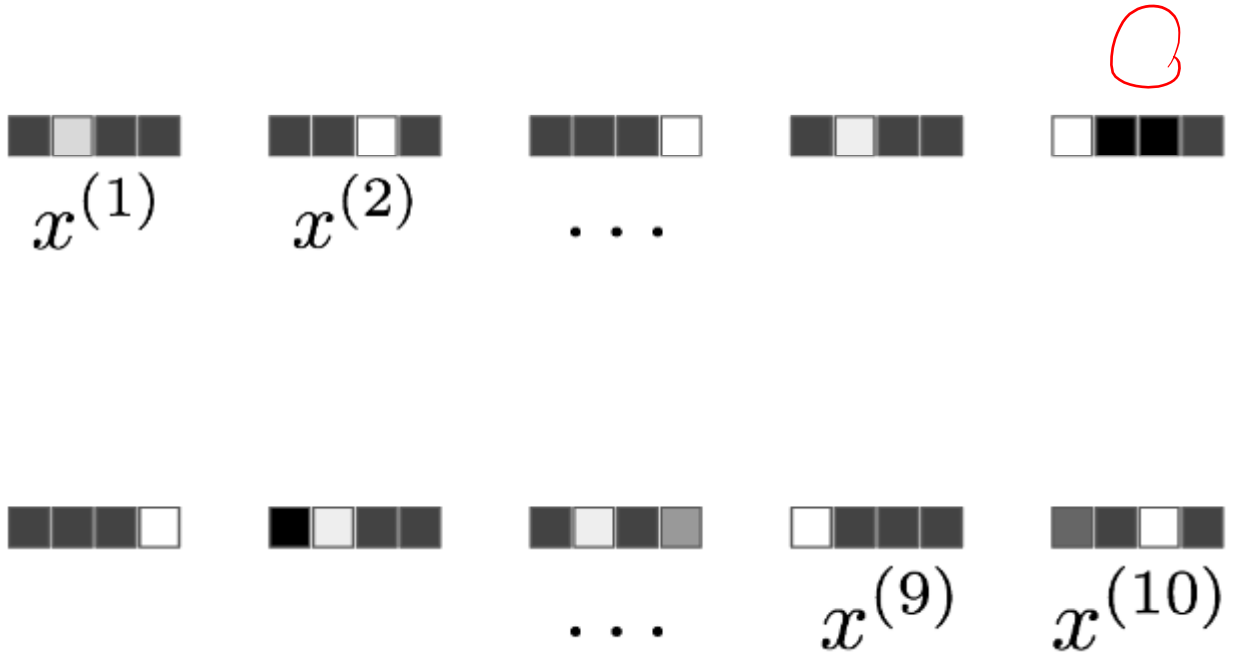
2 minute exercise

Think about all the properties of this dataset:

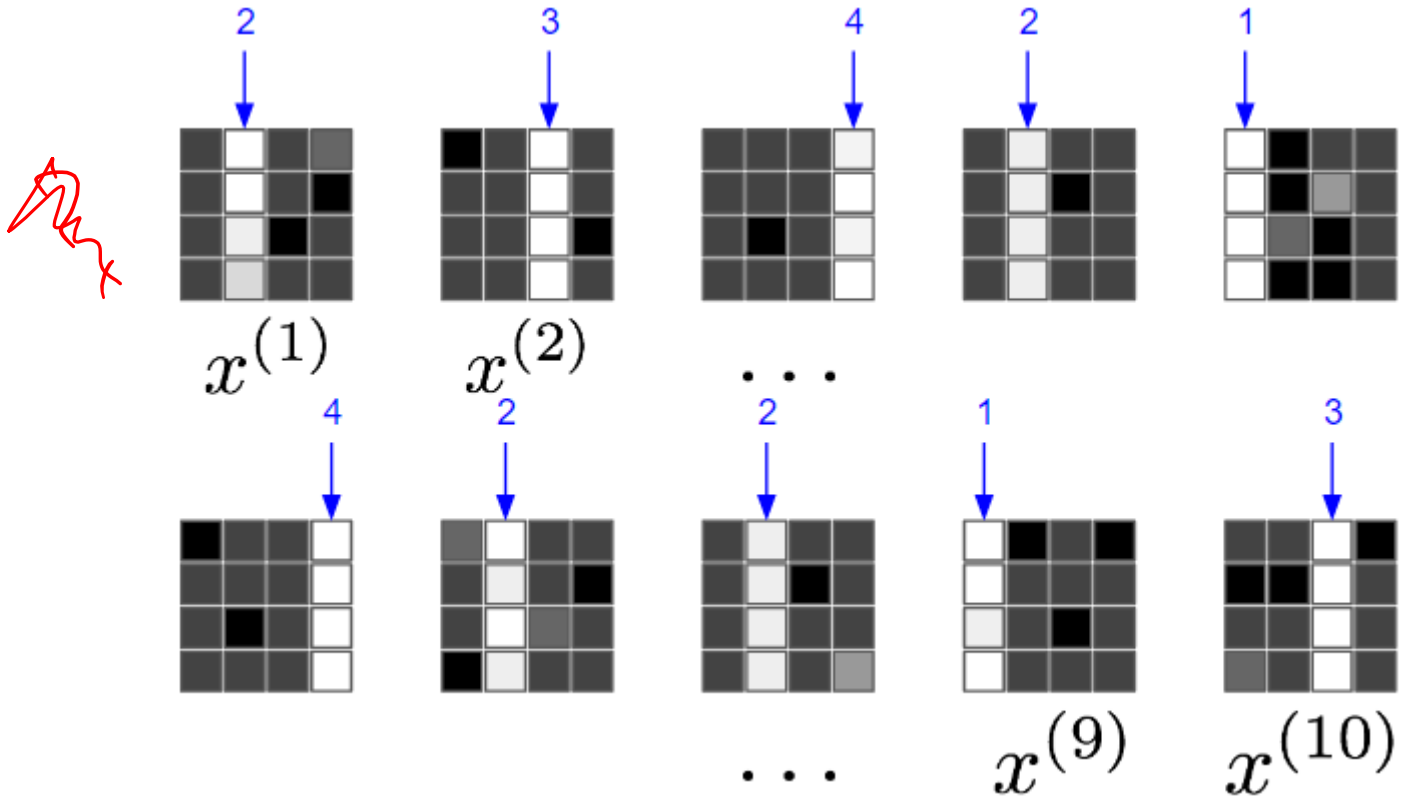
$\mathbb{R}^{4 \times 4}$



The rows are correlated



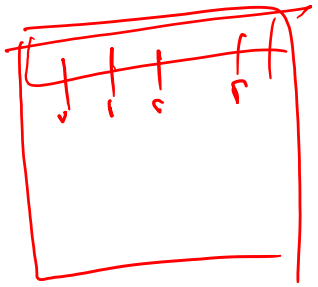
There's a simple encoding (the LATENT representation)



Data structure

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

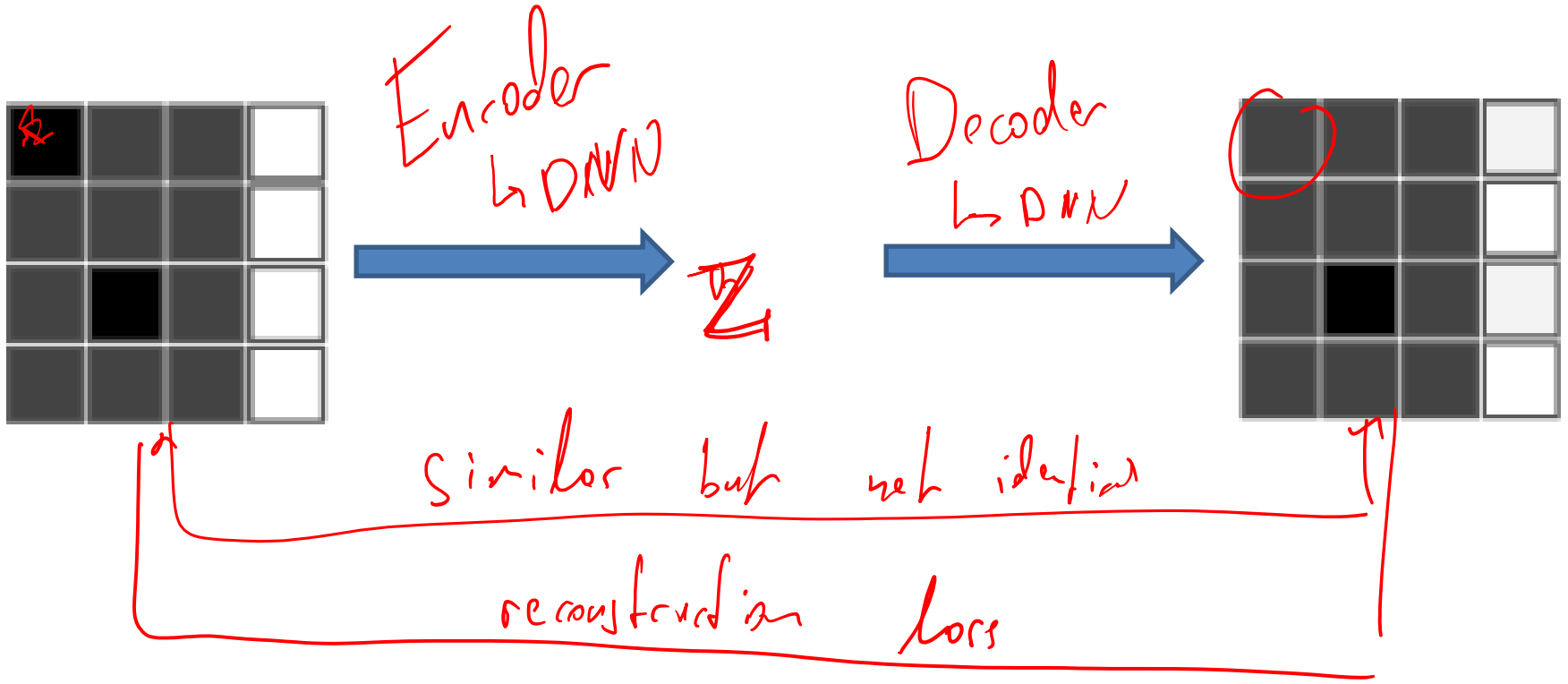
$$z^{(n)} = 1 \text{ or } 2, 3, 4$$



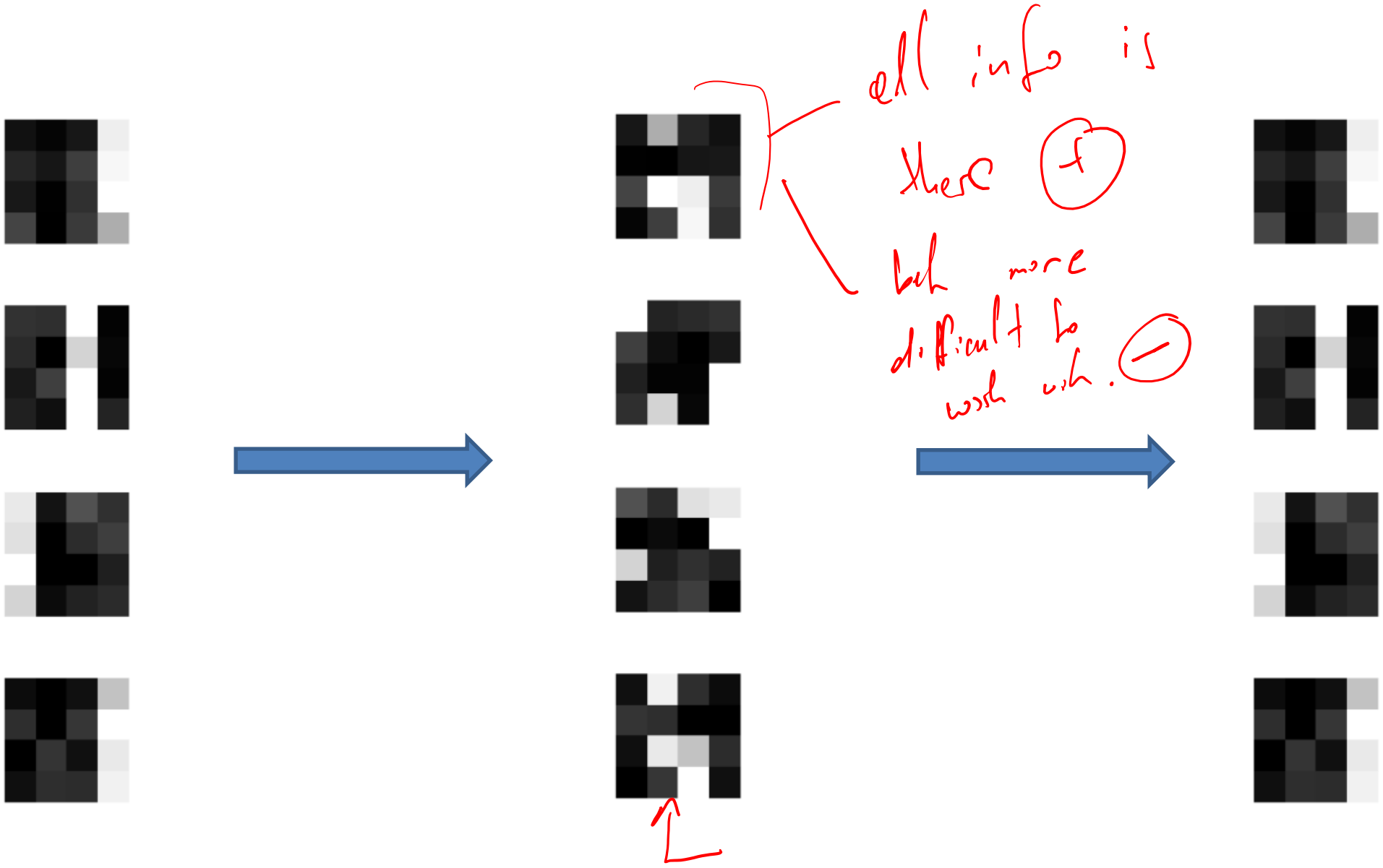
for each image *n*, even though each image is 16 dimensional

How to train a model to uncover it?

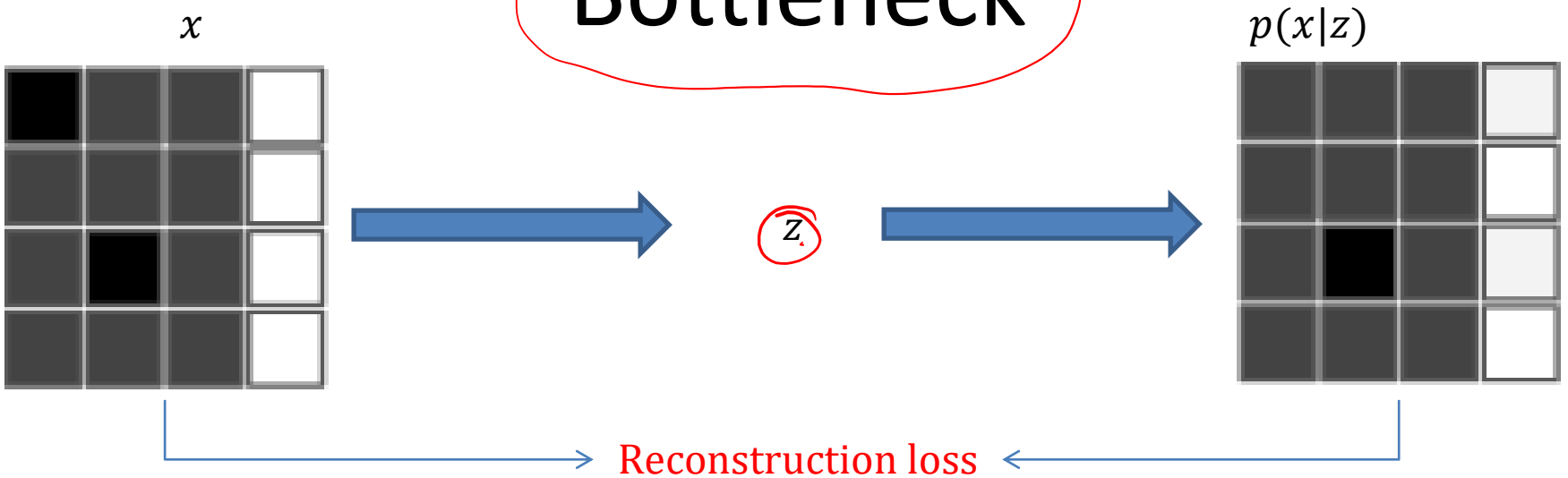
Autoencoders



Perfect Reconstruction === Useful Representations?



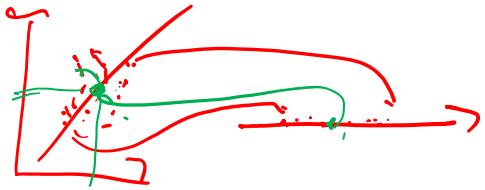
Bottleneck



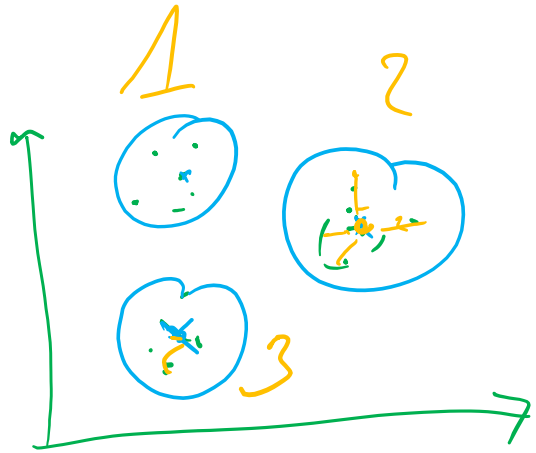
Bottleneck shapes the 'c':

- dim reduction
-

Auto-encoders that we know



- PCA and SVD find low-dimensional projections of data. They can be seen as a lossy autoencoder trained to minimize L2 reconstruction error under a dimensionality reduction bottleneck.



- K-Means assigns each data point to a cluster prototype. This is a lossy encoding. We “decode” a cluster ID into the location of the center. We can interpret K-Means as a lossy encoding scheme under a sparsity constraint (sample is assigned to only 1 cluster)

- Many other vairants, e.g. the sparse autoencoder (examples in 13-autoencoders.ipynb)

Simple Auto-encoders summary

+ compact data representations

+/- careful bottleneck design required to get desired information in the latent vector

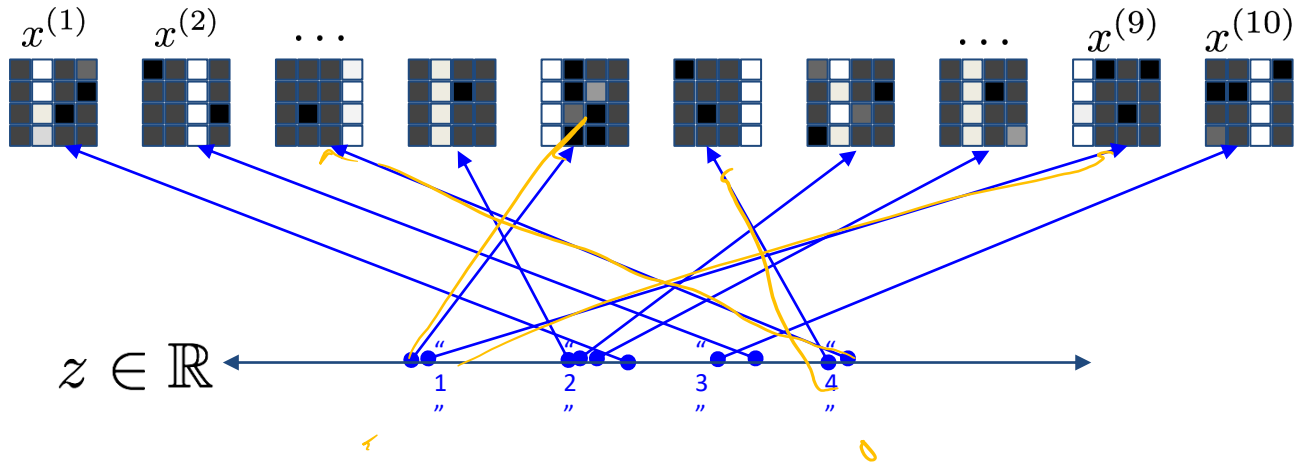
- no generation capability

VARIATIONAL AUTOENCODER:

A PRINCIPLED APPROACH TO AUTOENCODING AND DATA GENERATION

Intuition

- Complex data



- May have a simple latent structure

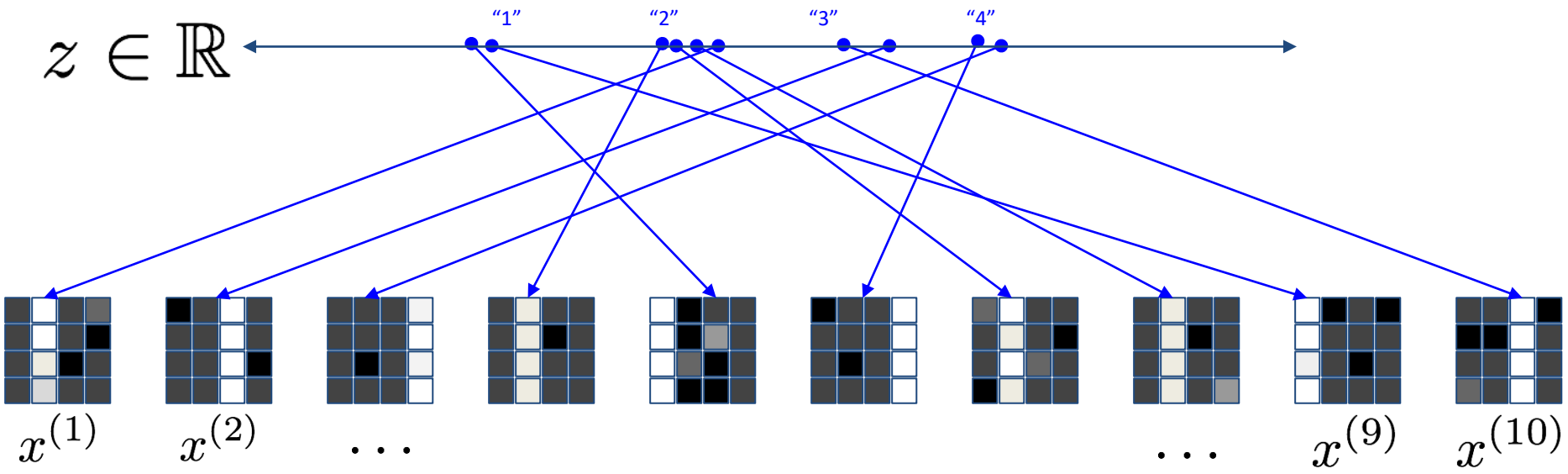
- Build a 2-stage generation process:

$z \sim \mathcal{N}(0,1)$ prior $p(z)$ assumed to be simple

$x \sim p(x|z)$ complicated transformation implemented with a neural network

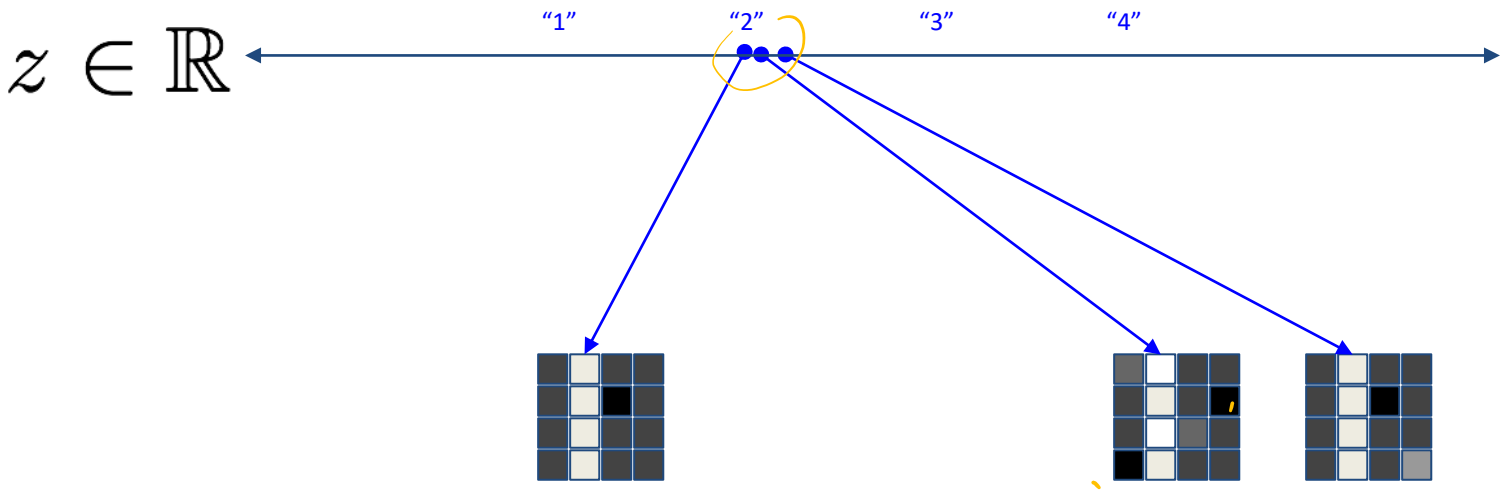
Data manifold

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



and noise

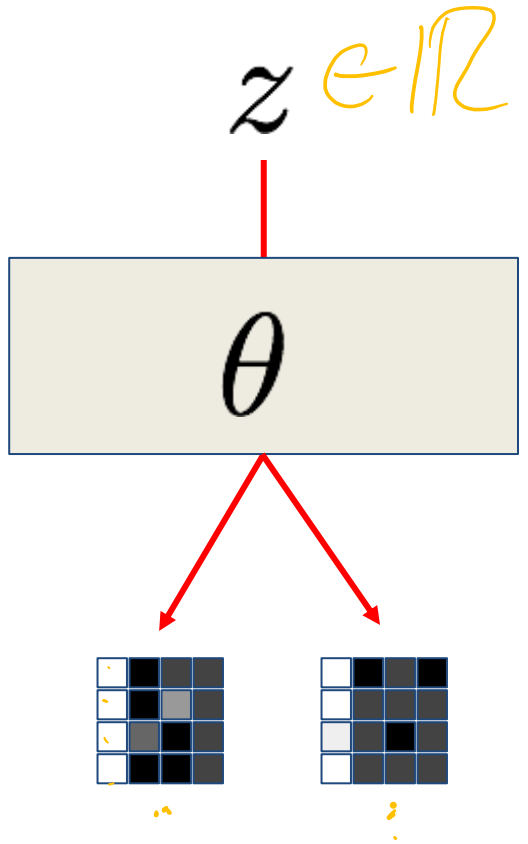
The 16-dimensional images live on a 1-dimensional manifold, plus some “noise”



Exercise

Think how to implement a neural network that takes z and produces a distribution over x:

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



Is your input one-dimensional?

Is your output 16-dimensional?

Identify all the “tunable” parameters Θ of your function

Inference

Generation:

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

Prior

DNN ← Decoder

Inference:

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

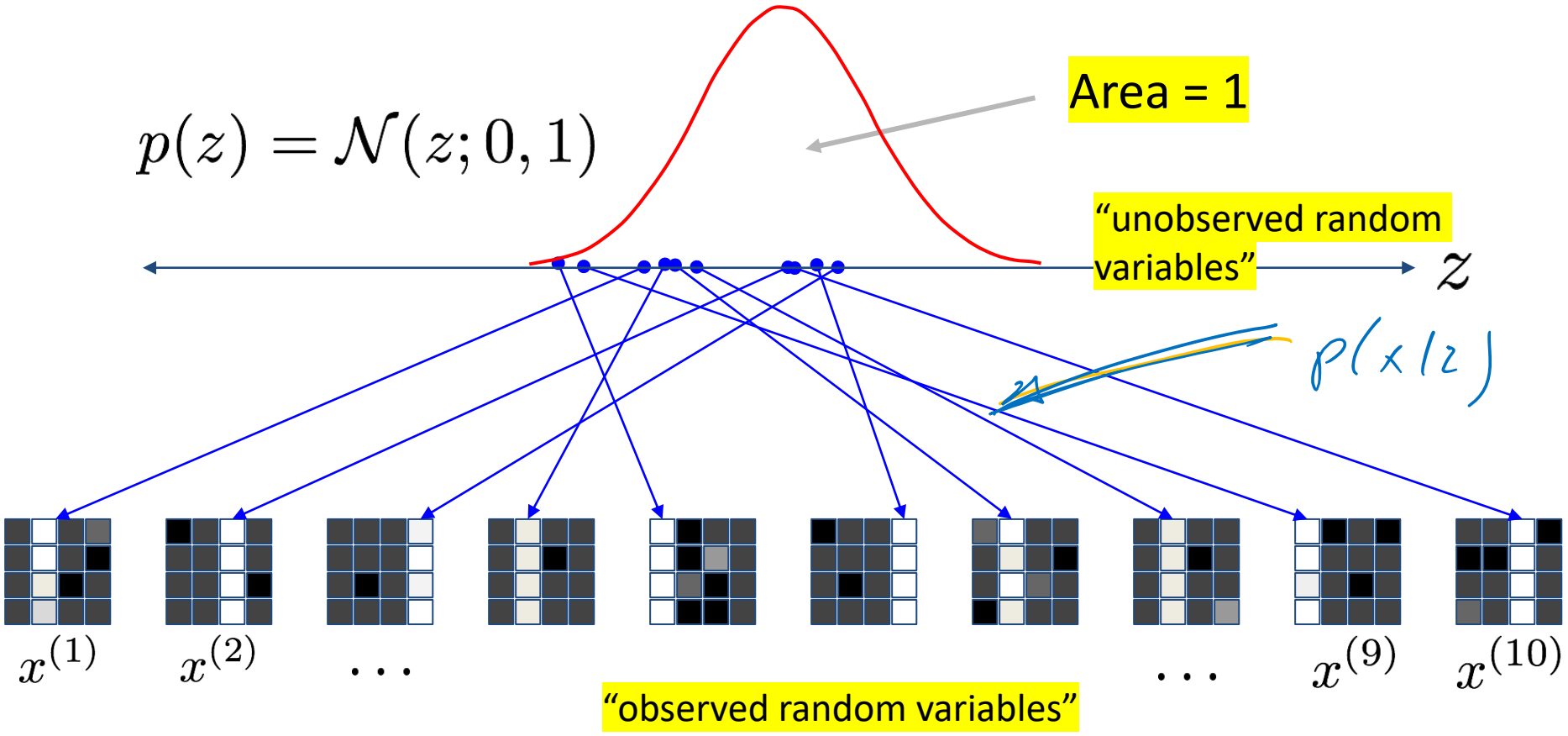
← Encoder

Bayes says:

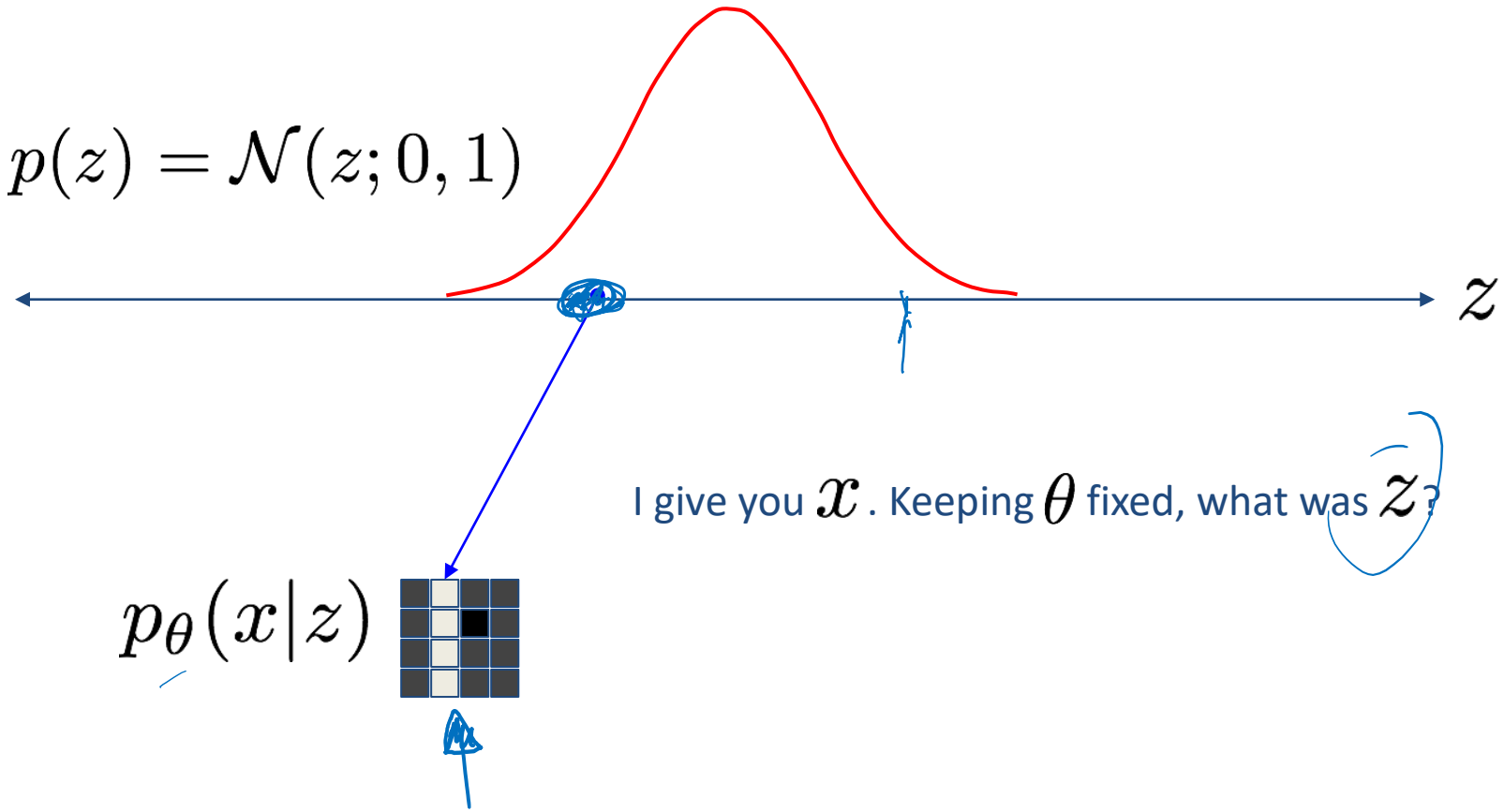
$$p_\Theta(z|x) = \frac{p_\Theta(x|z)p(z)}{\int dz' p_\Theta(x|z')p(z')}$$

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

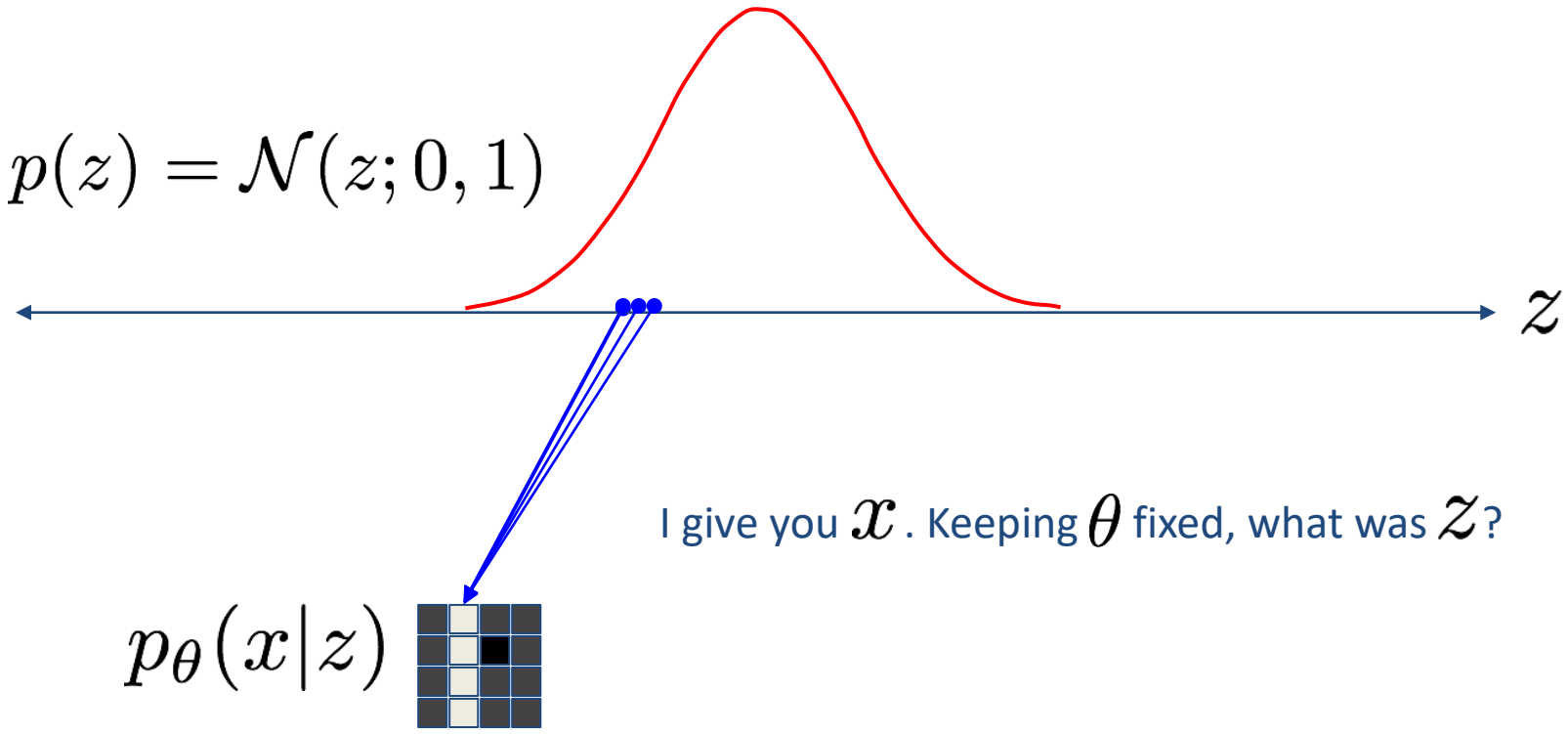
Inference starts with priors



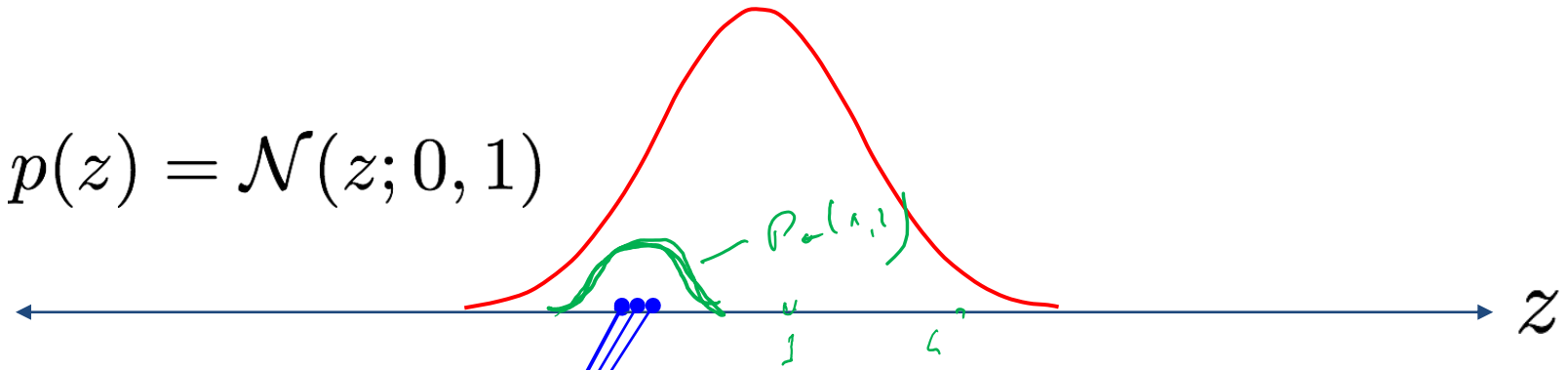
Inference

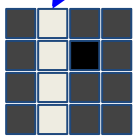

$p(z) = \mathcal{N}(z; 0, 1)$
z
I give you x. Keeping θ fixed, what was z?
$p_\theta(x|z)$

Inference


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

Exercise



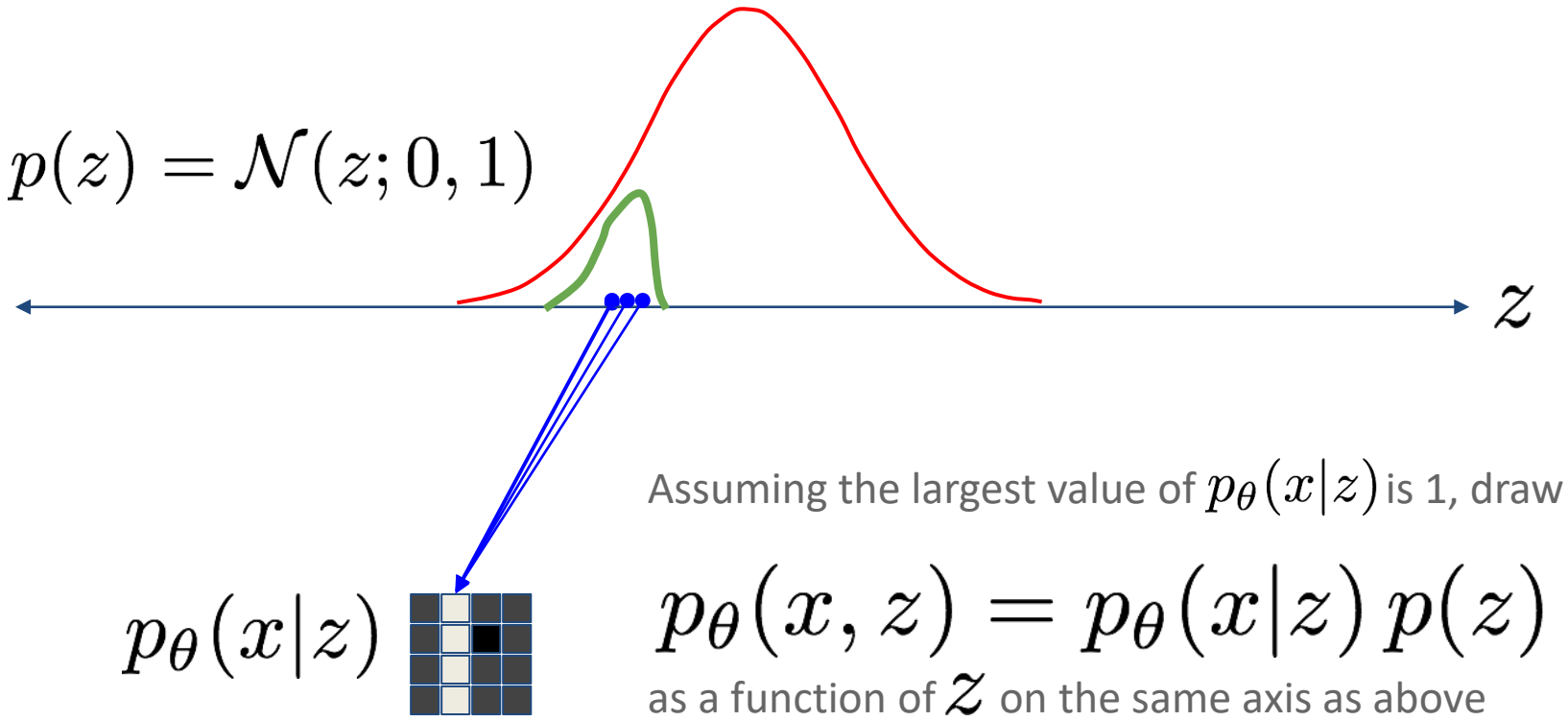
$p_\theta(x|z)$ 

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)

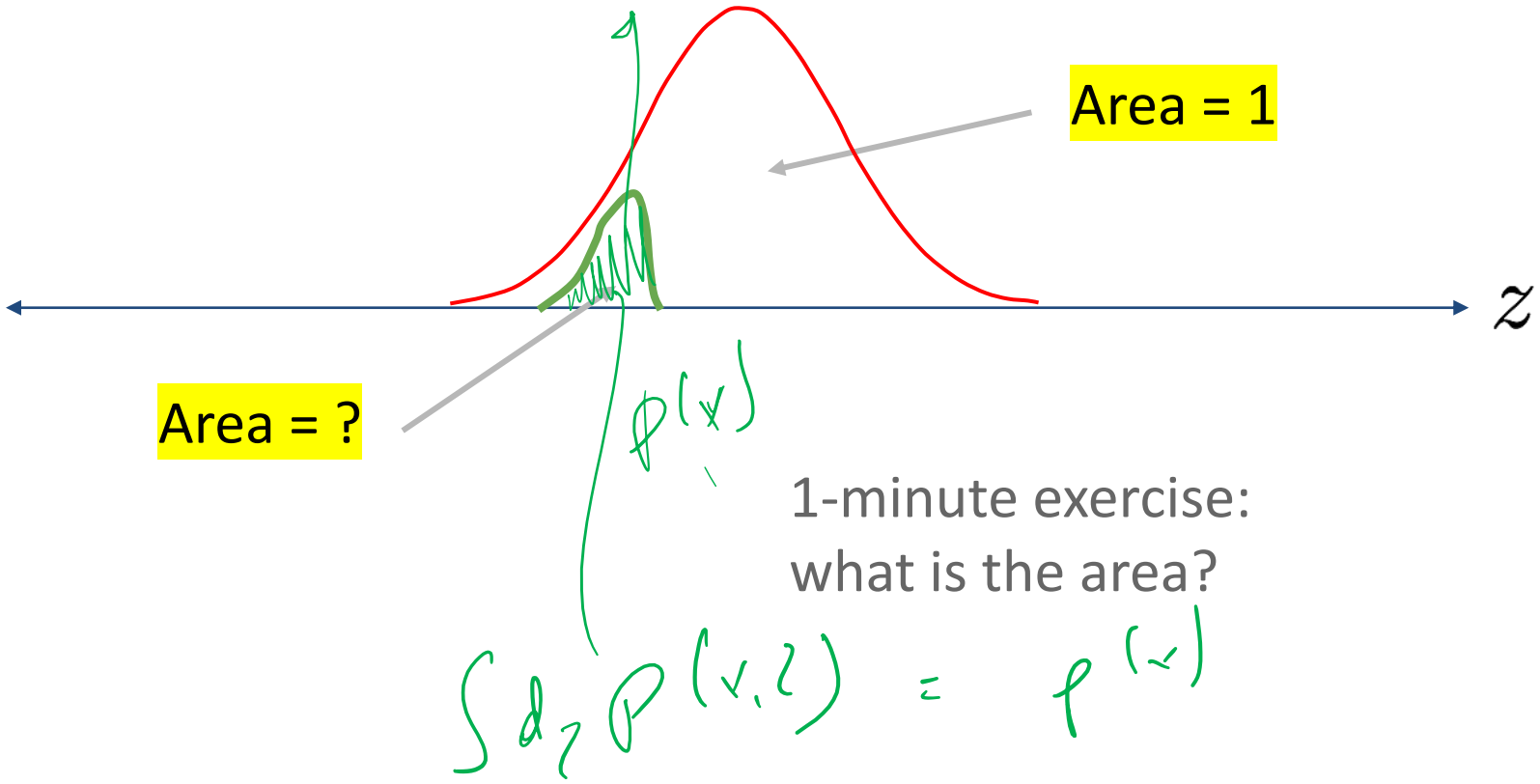


Assuming the largest value of $p_\theta(x|z)$ is 1, draw

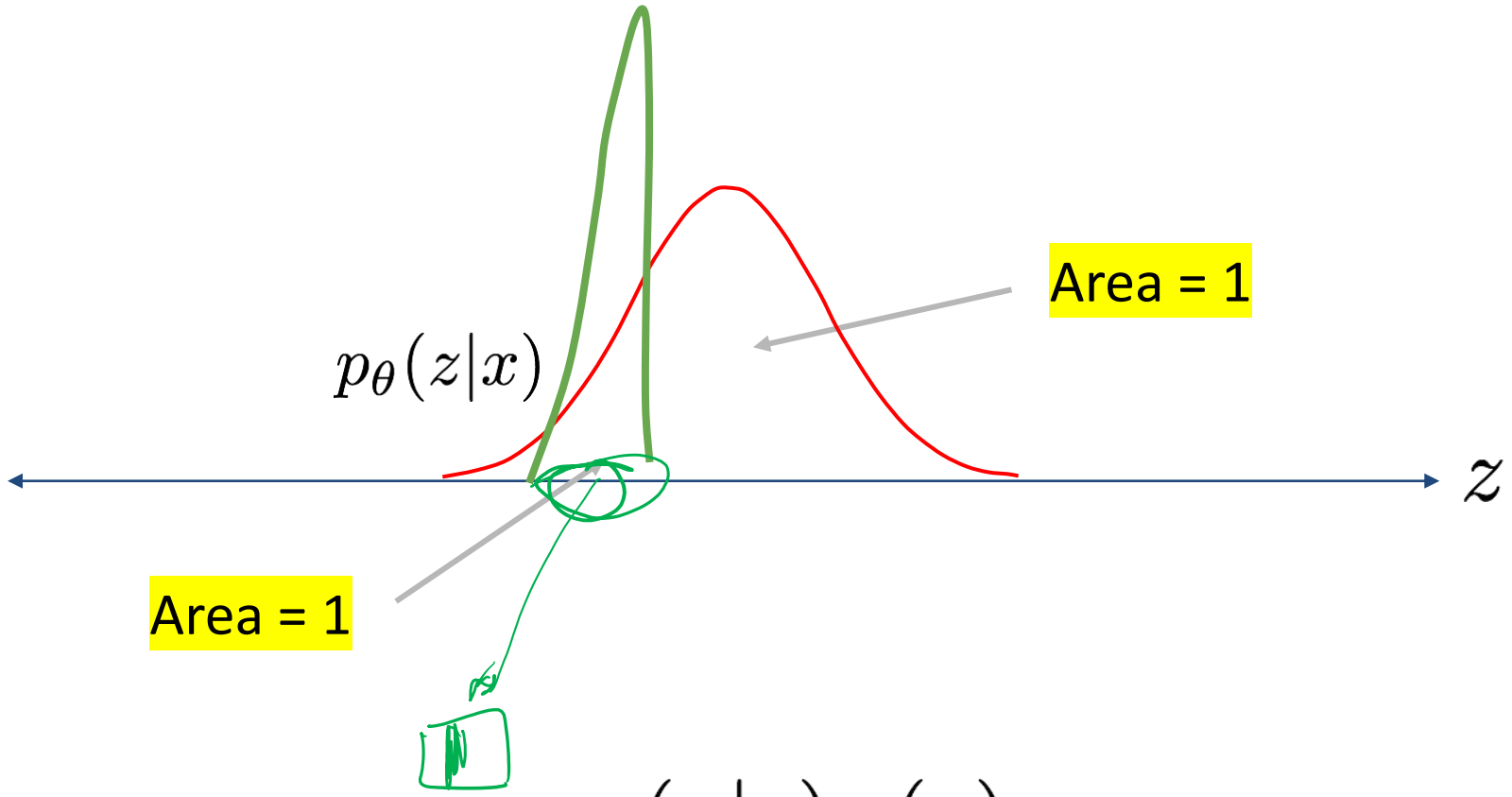
$$p_\theta(x, z) = p_\theta(x|z)\, p(z)$$

as a function of z on the same axis as above

Joint density (with x observed)



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...

Model Training Intuitions

Generation:

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

Training by max log-likelihood

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

Evidence of all data points

Area for data point *n*

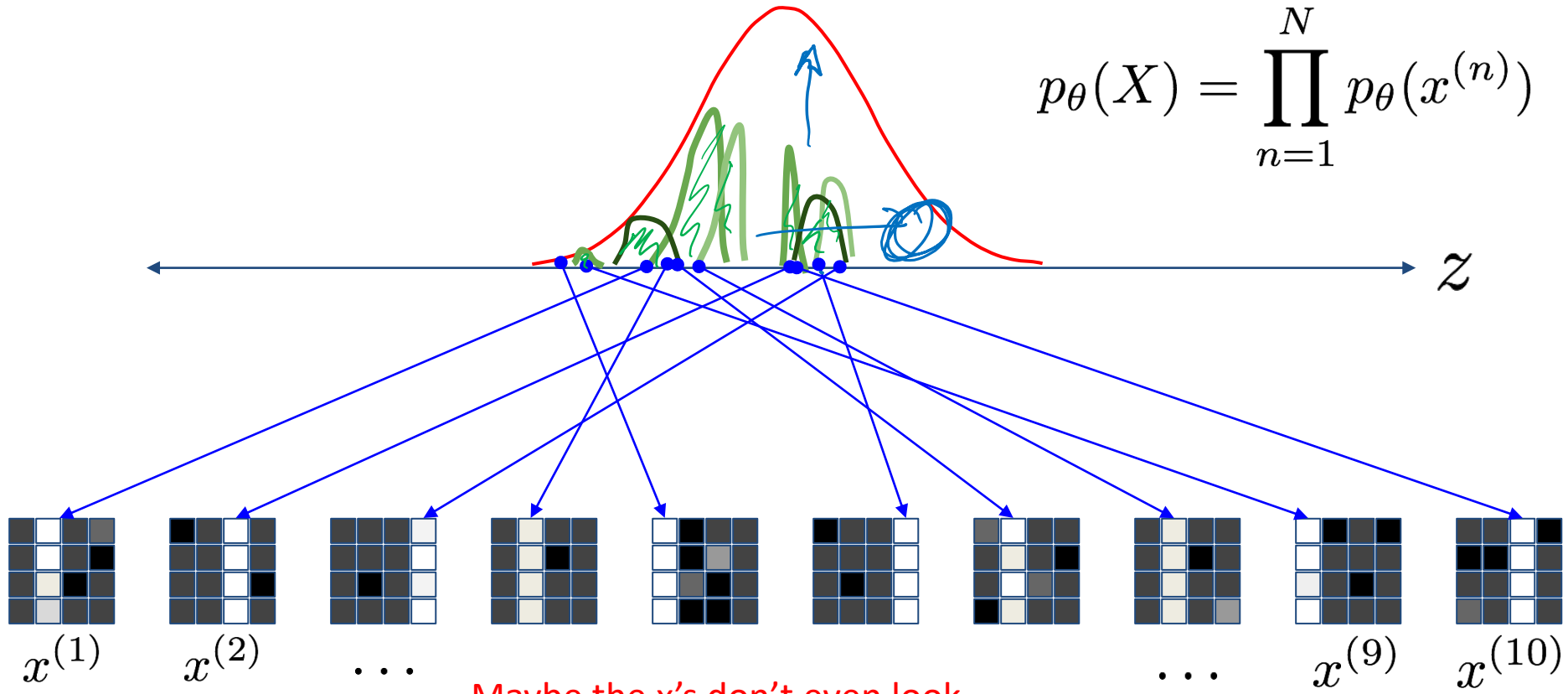
$$p_\theta(X) = \prod_{n=1}^{N} p_\theta(x^{(n)})$$

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

Evidence for all data points

The product of the areas underneath the green curves

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

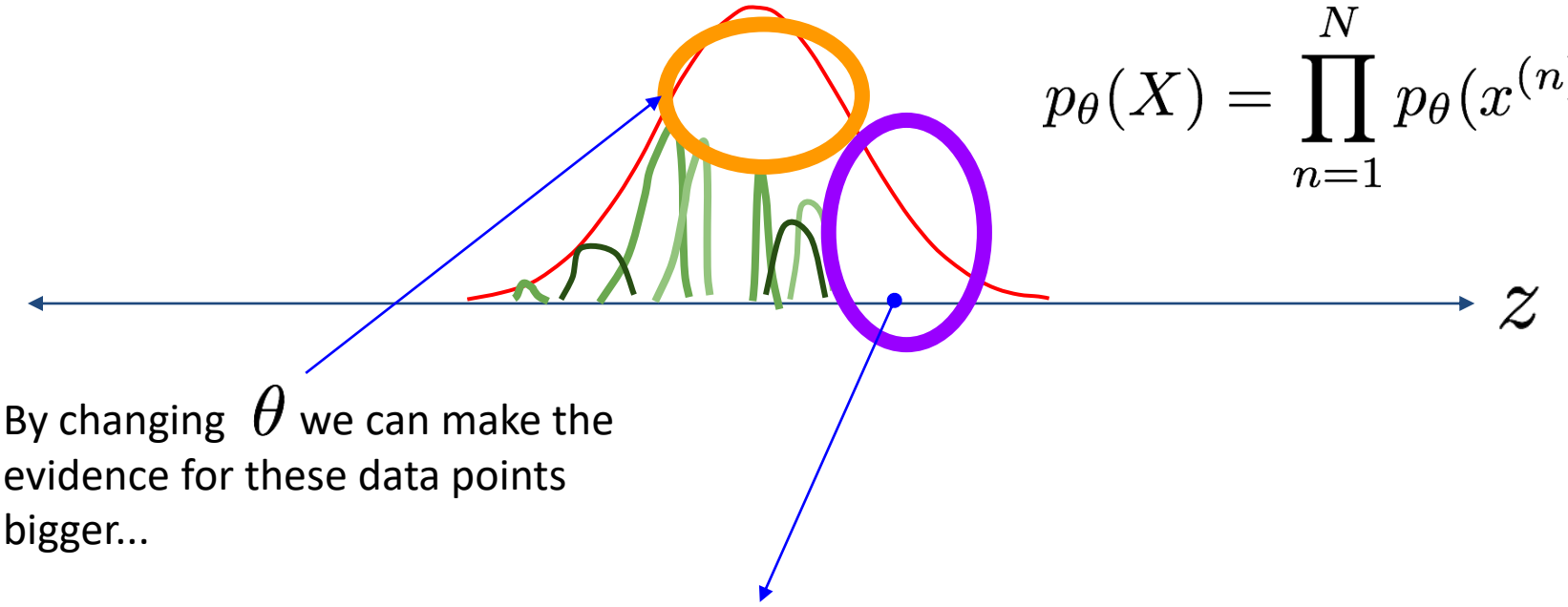


Maybe the x's don't even look that nice, when sampled with

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…

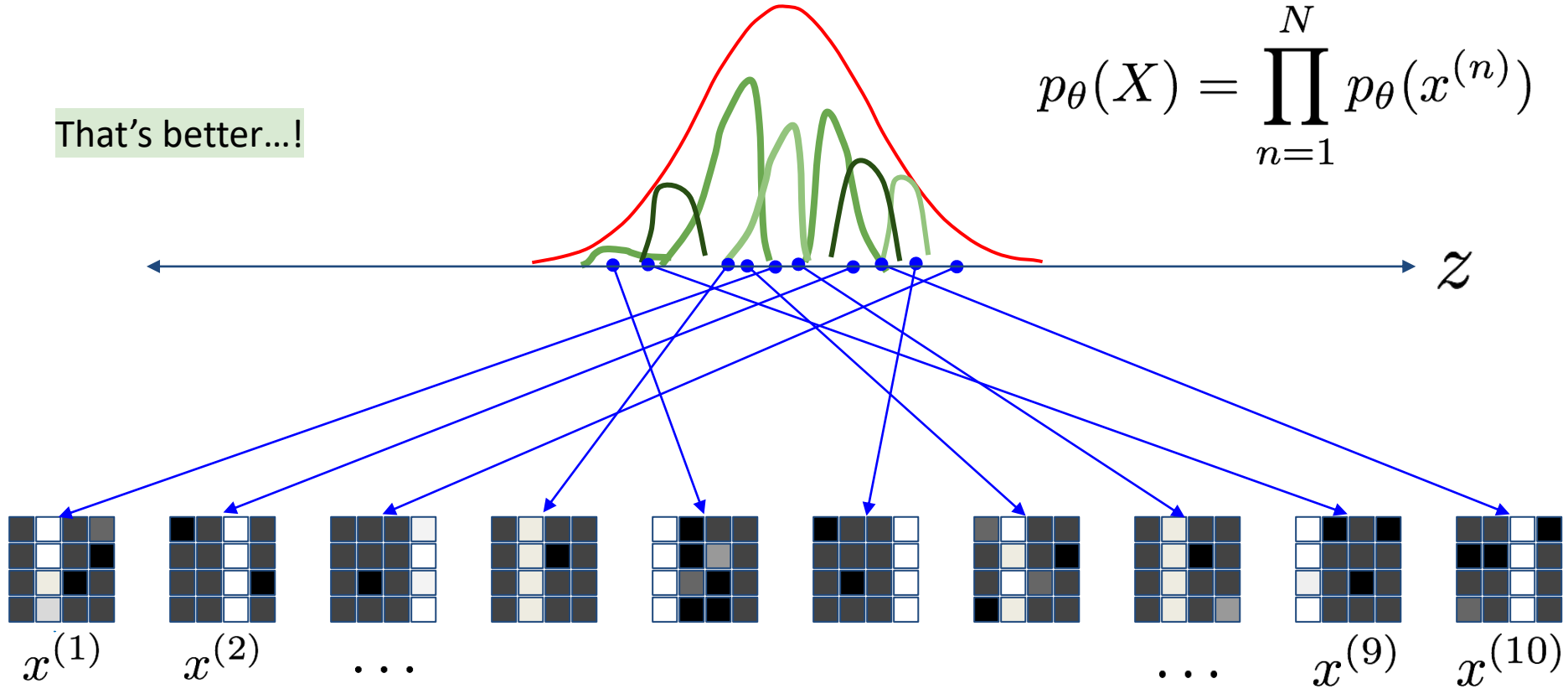
(With this θ, the prior doesn't capture the data manifold well)

Maximizing the evidence

The product of the areas underneath the green curves

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

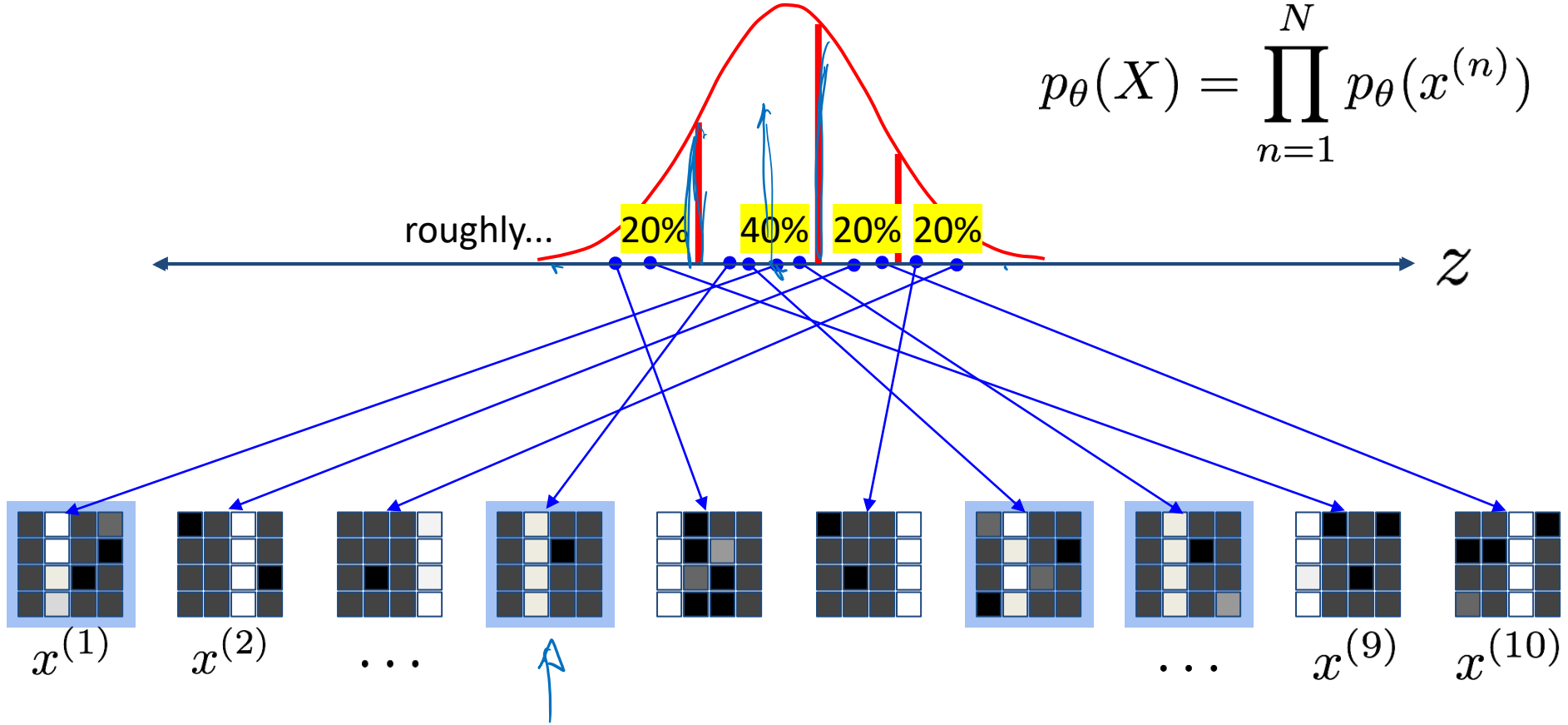
That's better…!



For the sharp-sighted

The product of the areas underneath the green curves

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



Training – practical aspects

Generation:

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

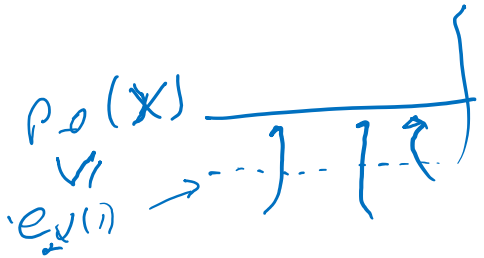
Training by max log-likelihood

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

But

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

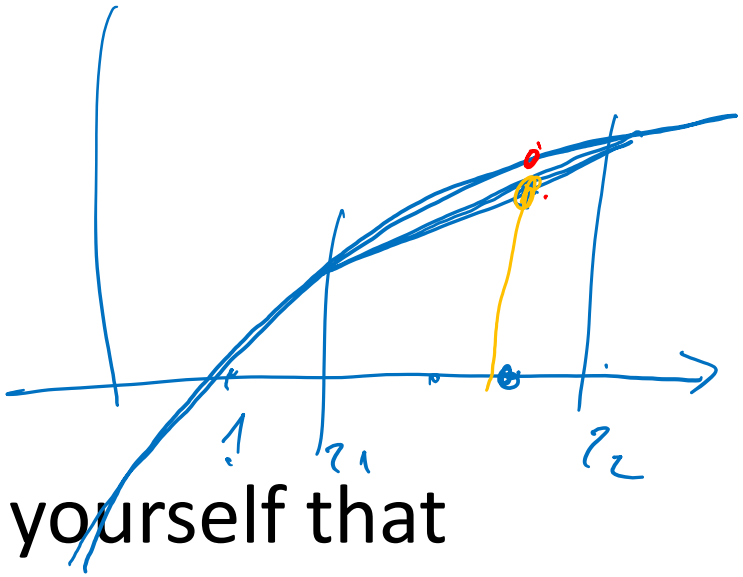
Approximate likelihood optimization



Our approach:

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

Exercise



Jensen's inequality

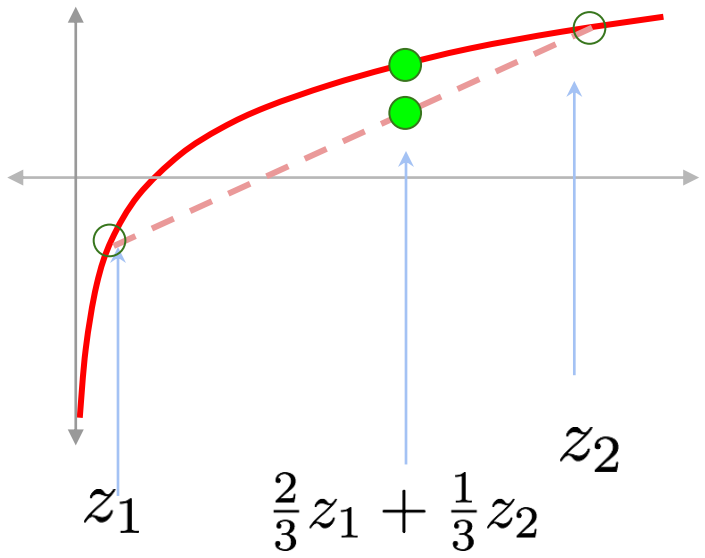
Draw log(...) as a function, convince yourself that

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

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

Jensen's inequality

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



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

ELBO: A likelihood bound

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

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

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

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

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

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

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

Works as long as $q_\phi(z|x) \neq 0$

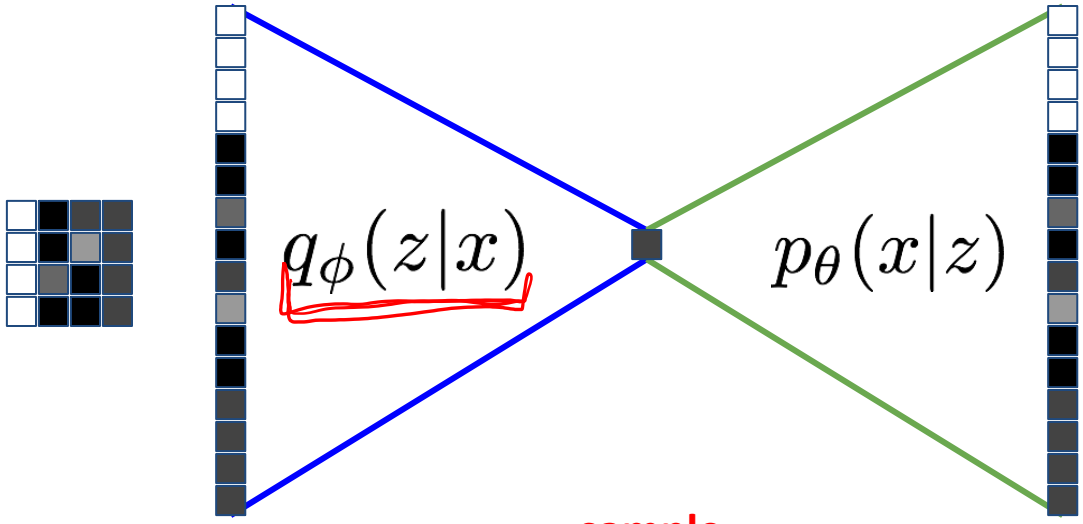
$\int dz\, q_\phi(z|x) = 1$

Jensen

ELBO interpretation

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

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

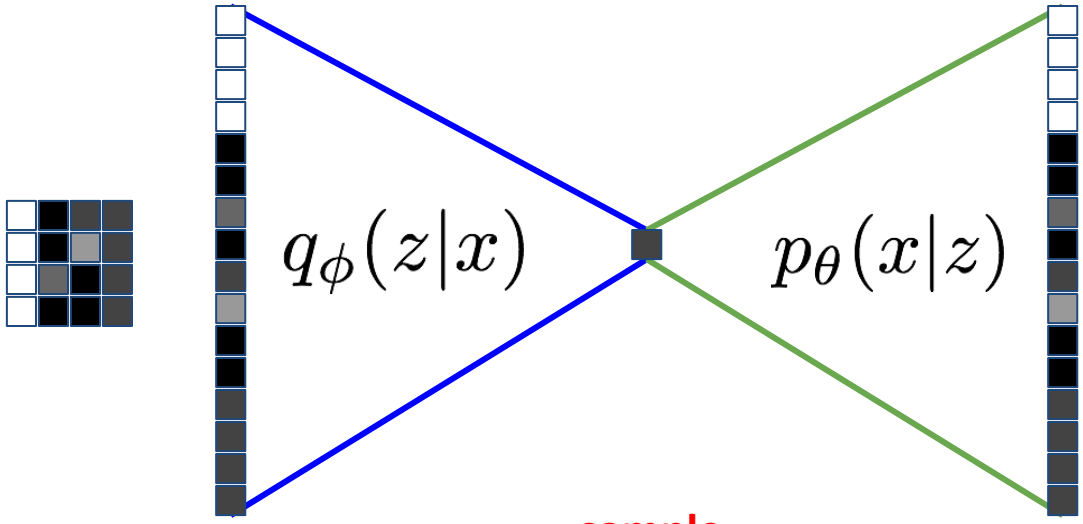


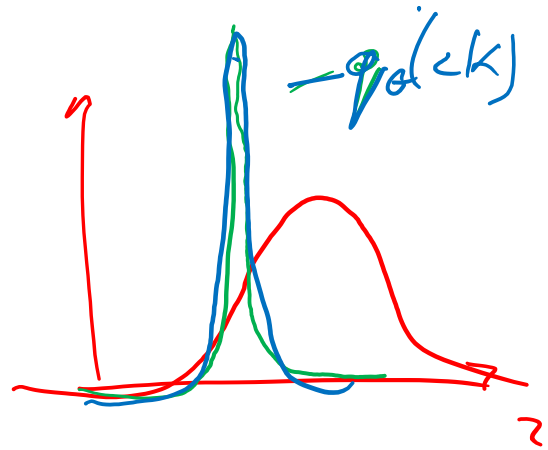
$$x \rightarrow z \rightarrow x$$

ELBO interpretation

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

$KL\big(q_\Phi(z|x) \parallel p(z)\big)$: Amount of information transmitted in $q_\Phi(z|x)$





ELBO interpretation

ELBO, or evidence lower bound:

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

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 $q(z|x)$

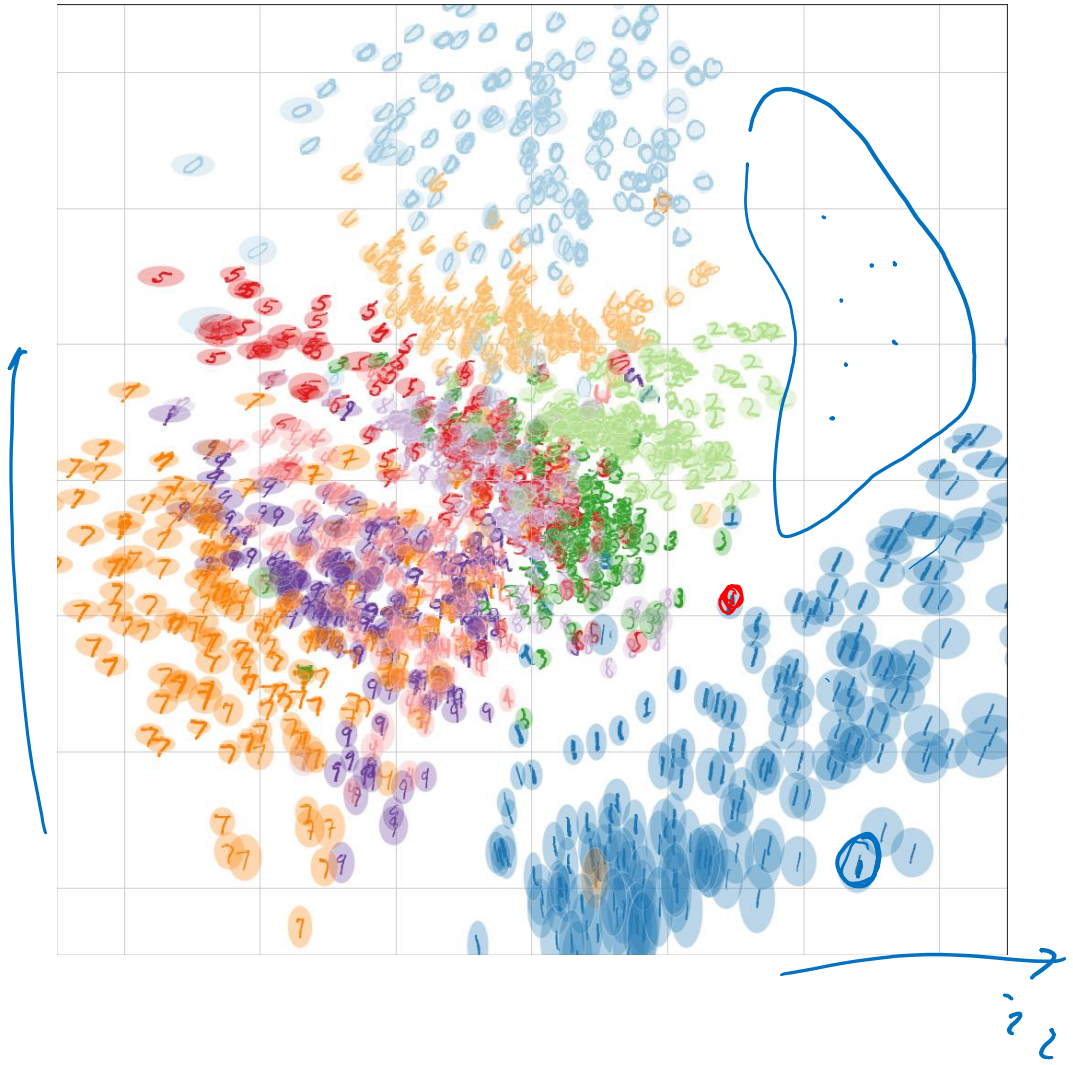
$KL\big(q_\Phi(z|x) \parallel p(z)\big)$ code transmission cost:
how many nats we transmit about x in $q_\Phi(z|x)$ rather than $p(z)$

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

VAE is an Information Bottleneck

Each sample is represented as a Gaussian

This discards information (latent representation has low precision)



ELBO Evaluation

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:

$$\mathbb{E}_{z\sim q_\Phi(z|x)}[\log p_\Theta(x|z)] \approx \frac{1}{N}\sum_i \log p_\Theta(x|z_i)$$

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

for $i = 1 \ldots N$

ELBO optimization

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

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!

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?

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

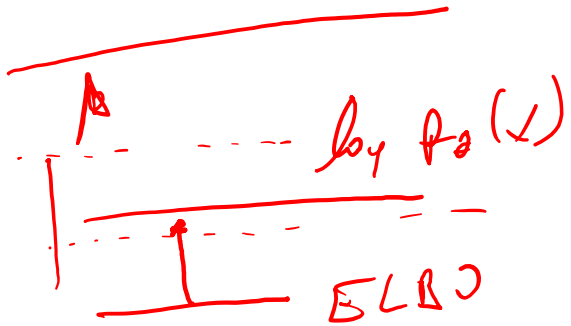
ELBO is tight when $q_\Phi(z|x)$ does exact inference!

ELBO optimization

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

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 $q_\Phi(z|x) \approx p_\Theta(z|x)$

Similar to E step

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

Similar to M step

But we tune Φ and Θ at the same time!

Alternative VAE Explanation: EM

Want:

$$\max \log p_\Theta(x) = \max \log \sum_z p_\Theta(x,z) = \max \log \sum_z p_\Theta(x|z) p_\Theta(z)$$

Iterate:

E-step: find $q_\Phi(z|x) = p_\Theta(z|x)$

M-step: $\max_\Theta \mathbb{E}_{z\sim q_\Phi(z|x)} \left[\log \left(\frac{p_\Theta(z,x)}{q_\Phi(z|x)}\right)\right]$ (keep Φ fixed)

Why?

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

Alternative VAE Explanation: ELBO Proof

$$KL\left(q_\Phi(z|x) \parallel p_\Theta(z|x)\right) + \mathbb{E}_{z\sim q_\Phi(z|x)}\left[\log\left(\frac{p_\Theta(z,x)}{q_\Phi(z|x)}\right)\right] =$$

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

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

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

Alternative VAE Explanation: ELBO Proof

We have shown that for any $q_\Phi(z|x)$

$$\log p_\Theta(x) = KL\big(q_\Phi(z|x) \parallel p_\Theta(z|x)\big) + \mathbb{E}_{z\sim q_\Phi(z|x)}\left[\log\left(\frac{p_\Theta(z,x)}{q_\Phi(z|x)}\right)\right]$$

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

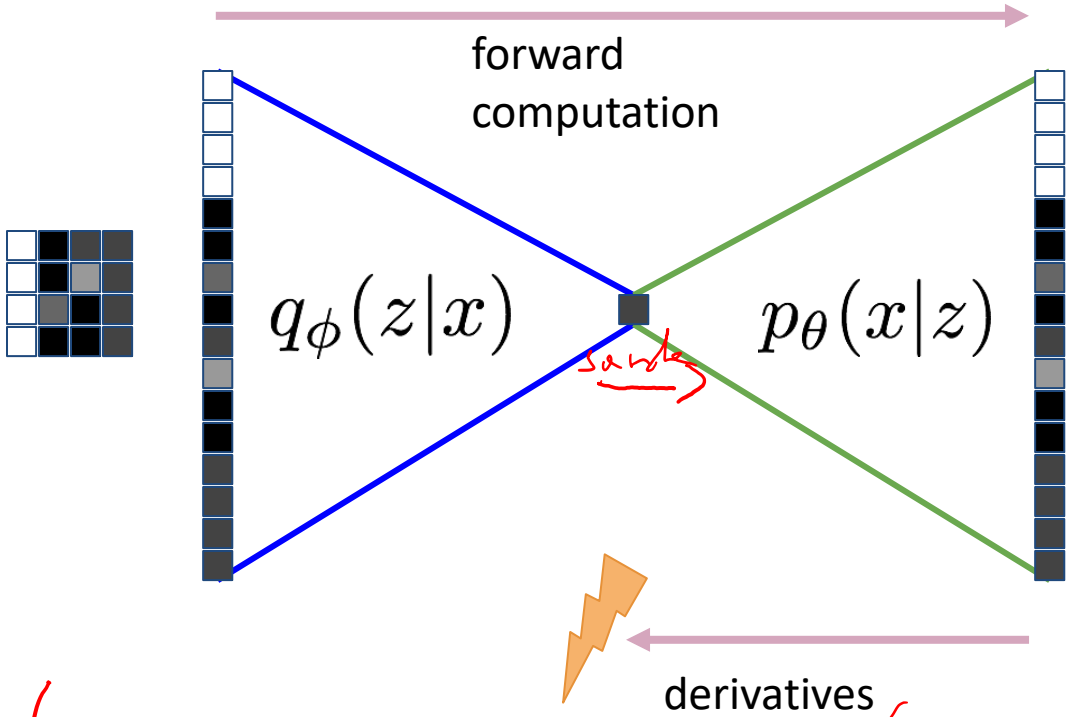
$$= \mathbb{E}_{z\sim q_\Phi(z|x)}[\log p_\Theta(x|z)] - KL\big(q_\Phi(z|x) \parallel p_\Theta(z)\big)$$

The bound is tight for $p(z|x) = q(z|x)$ (Then $KL\big(q_\Phi(z|x) \parallel p_\Theta(z|x)\big) = 0$).

Idea (VAE): Optimize ELBO using gradient techniques jointly over Θ and Φ

How to train a VAE?

We want to max ELBO



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

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

Then:

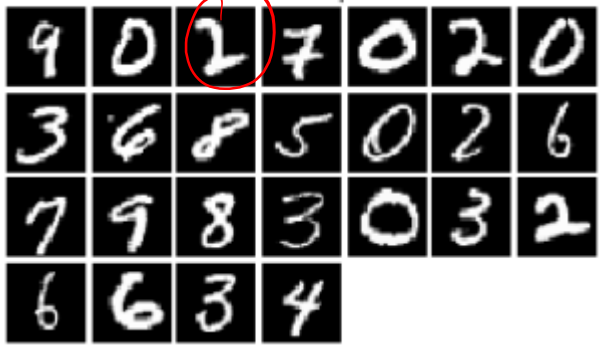
$$\mathbb{E}_{z\sim q_\Phi(z|x)}[\log p_\Theta(x|z)]$$
$$= \mathbb{E}_{\epsilon\sim\mathcal{N}(0,1)}[\log p_\Theta(x|\mu_z + \sigma_z\epsilon)]$$

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

μ_z, σ_z are computed by $q_\phi(z|x)$

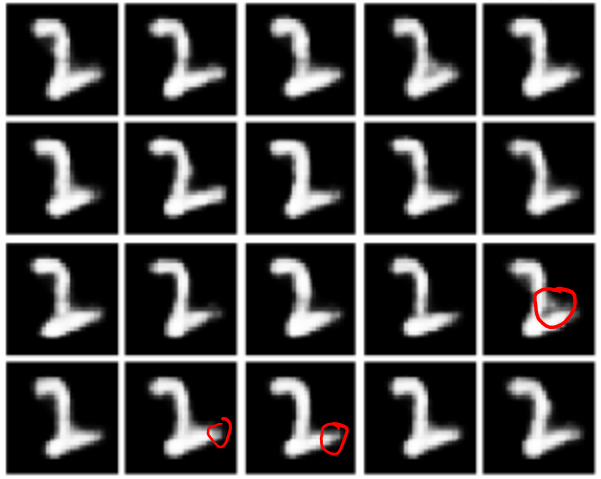
VAE in action

Data samples

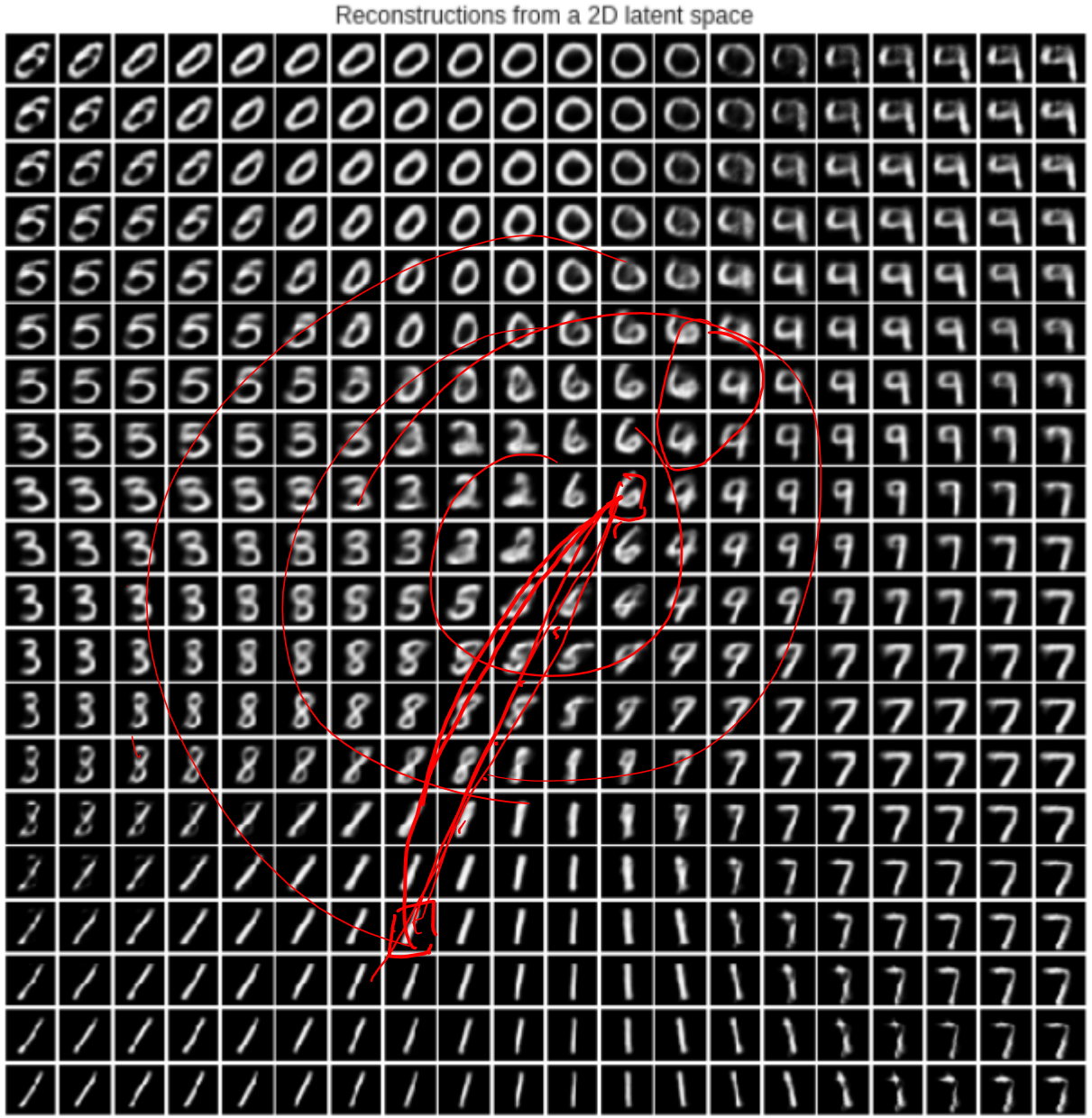


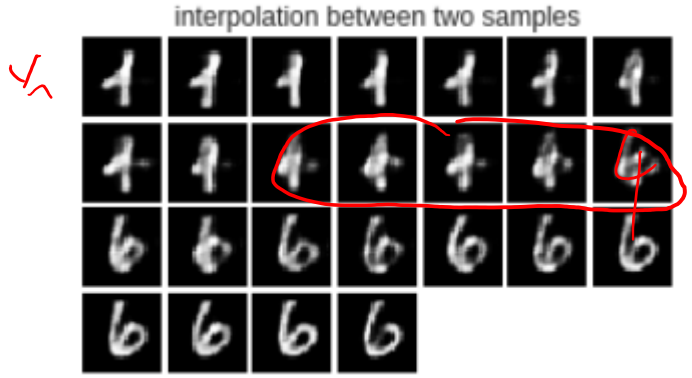
Reconstructions using the latent expected value

Reconstructions from multiple latent samples



VAE in action



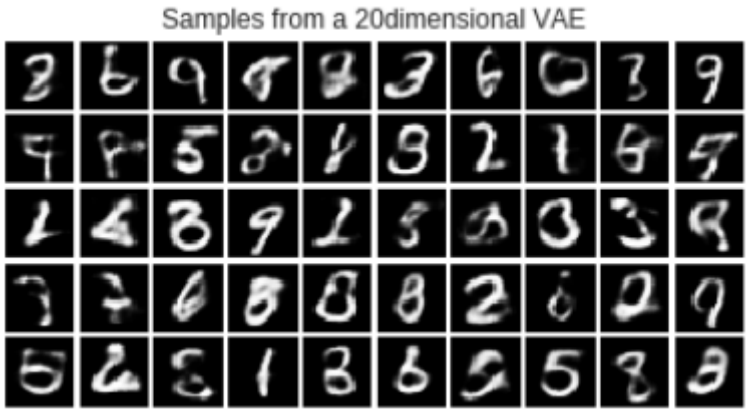


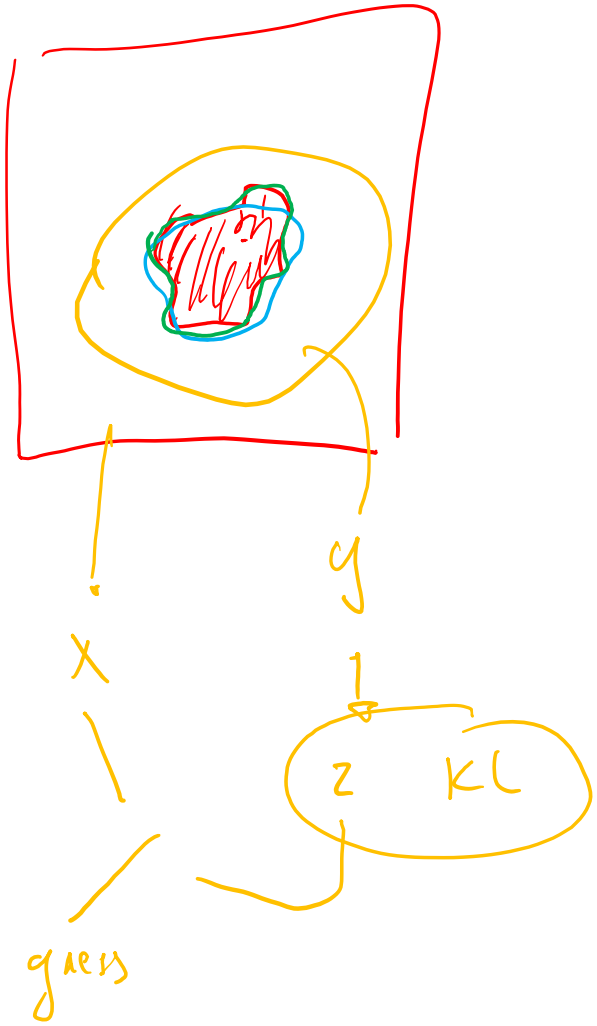
$$\mu_1, \sigma_1 = \varphi(x_1)$$

$$\mu_2, \sigma_2 = \varphi(x_2)$$

$$z_s = \mu_1, \ldots \mu_2$$

VAE in action





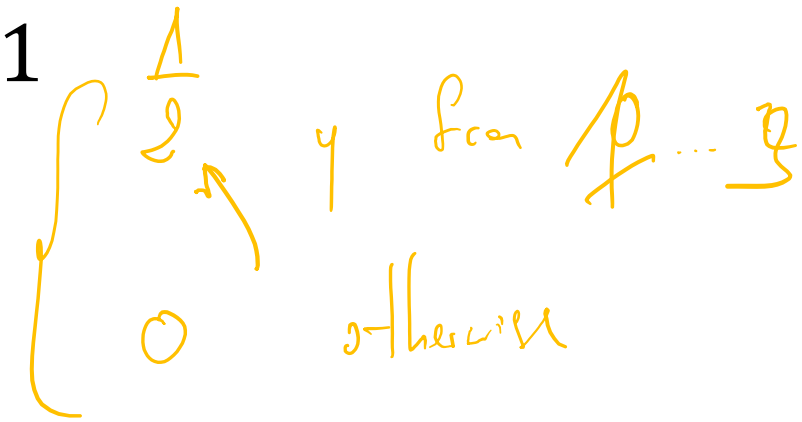
NORMALIZING FLOWS

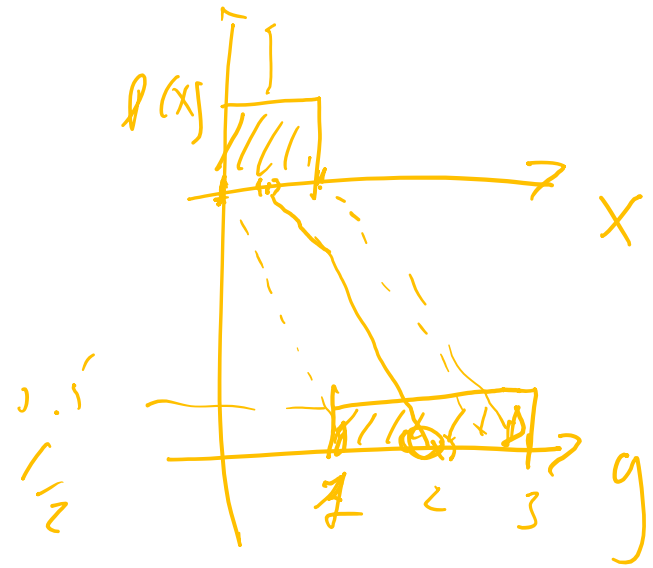
Exercise

$$p(x) = \text{uniform}(0,1) = \begin{cases} 1 \text{ for } x \in (0,1) \\ 0 \text{ otherwise} \end{cases}$$

$$y = 2 * x + 1$$

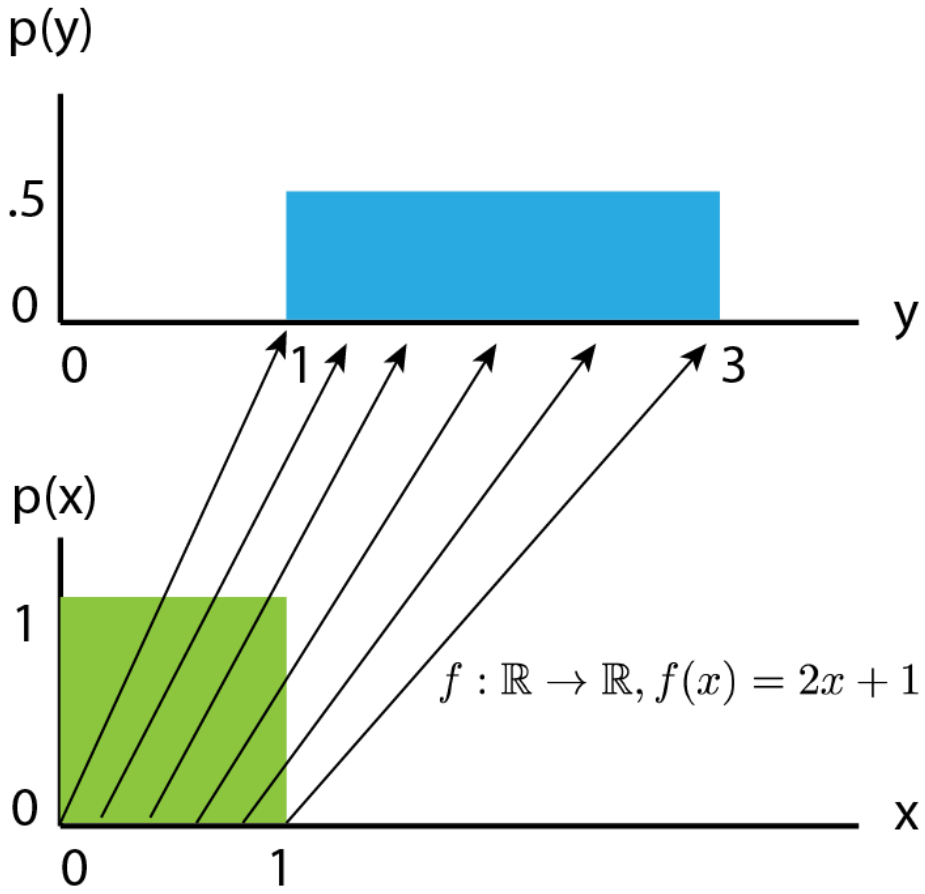
$$p(y) = ?$$





$y = x^2$

Transforming distributions



y also has a uniform distribution!

$$p(y) = \text{uniform}(1,3) = \begin{cases} 1/2 \text{ for } y \in (1,3) \\ 0 \text{ otherwise} \end{cases}$$

E Jang https://blog.evjang.com/2018/01/nf1.html

The Jacobian: stretching space

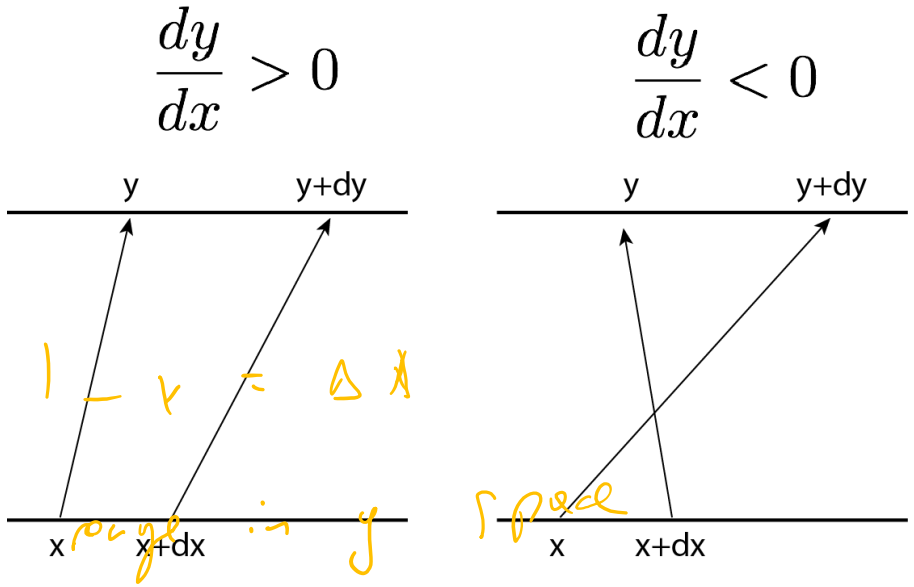
Let $y = f(x)$

Take a range $(x, x + \Delta x)$

Question:

How long is this range?

$$\Delta x$$



Question:

How long is $(f(x), f(x + \Delta x))$?

$$f(x + \Delta x) \approx f(x) + \frac{\partial f}{\partial x} \Delta x$$

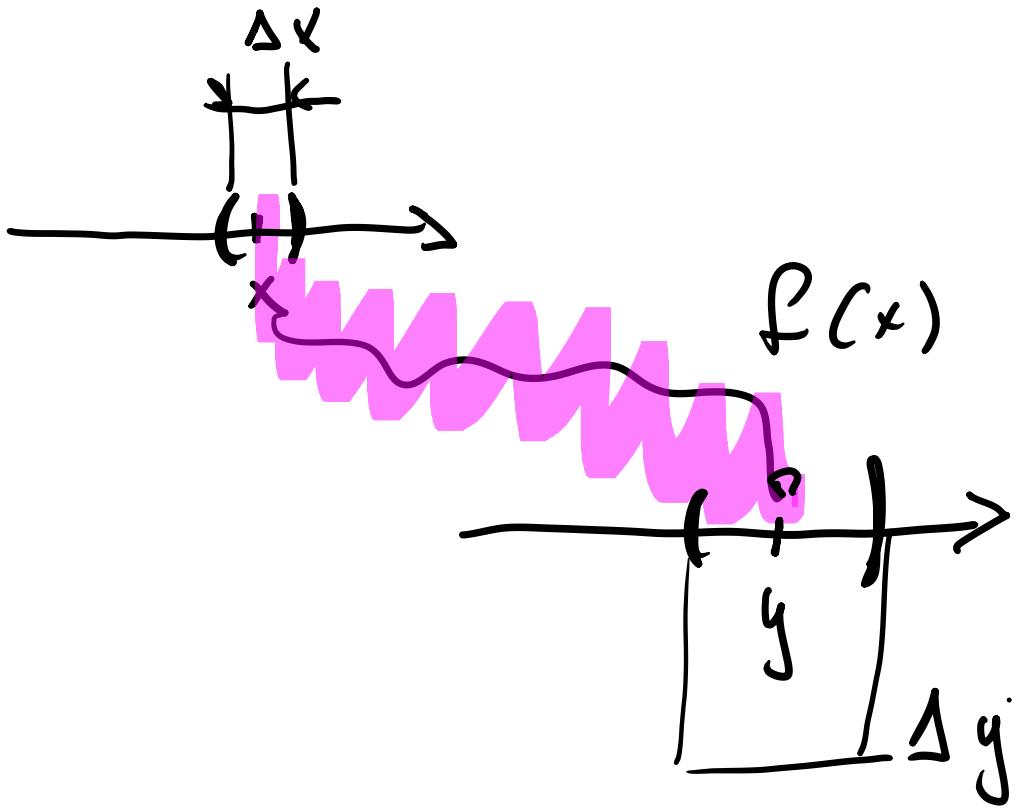
$$\left| \frac{\partial f}{\partial x} \Delta x \right|$$

https://blog.evjang.com/2018/01/nf1.html

Change of variables

$y = f(x), f$ is a bijection

$$p_x(x) = p_y(f(x)) \left| \det \frac{\partial f(x)}{\partial x} \right|$$

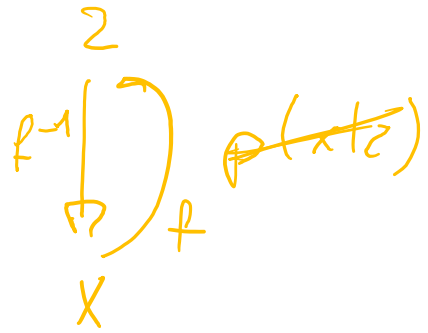


f transforms $x \pm \frac{\Delta x}{2}$ to y $\pm \frac{\Delta y}{2}$

The space is stretched by

$$\frac{\partial f(x)}{\partial x} \approx \frac{\Delta y}{\Delta x}$$

Idea



Start with $z \sim \mathcal{N}(0,1)$

Then $x = f^{-1}(z)$ (equivalently $z = f(x)$)

$$p_x(x) = p_z\big(f(x)\big)\left|\det\frac{\partial f(x)}{x}\right|$$

Tractable when:

- f is easy to exactly invert

  f and f^{-1} form and exact auto-encoder!

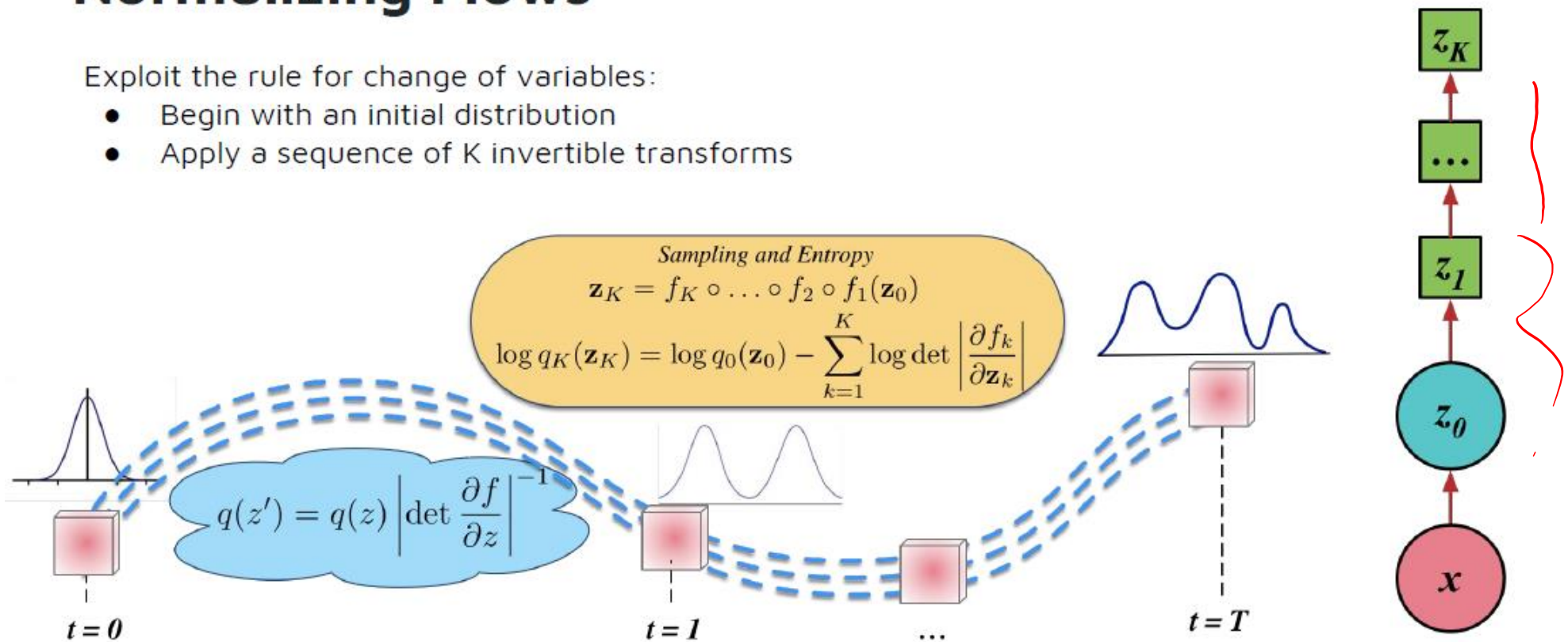
- $\det\frac{\partial f(x)}{x}$ is easy to compute

=> We need special f!

Normalizing Flows

Exploit the rule for change of variables:

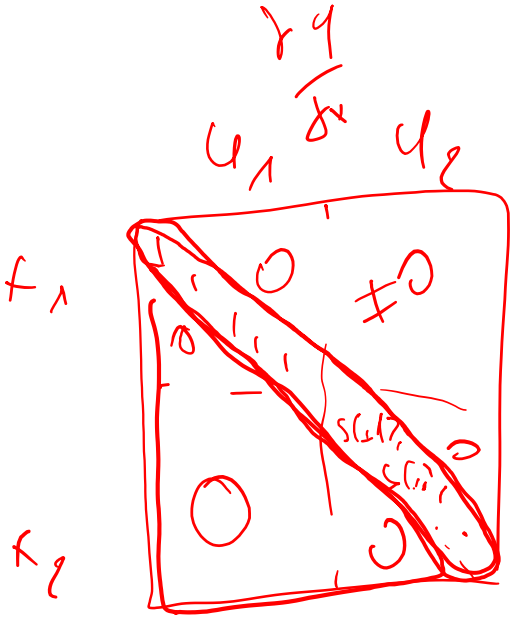
- Begin with an initial distribution
- Apply a sequence of K invertible transforms

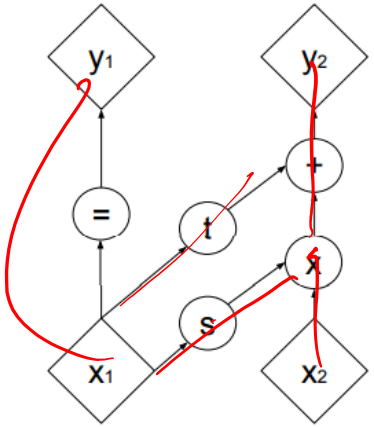


Distribution flows through a sequence of invertible transforms

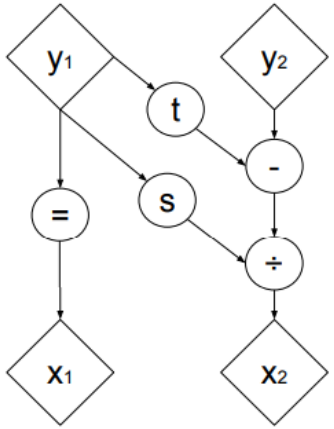
Rezende and Mohamed, 2015

A special form of f





(a) Forward propagation



(b) Inverse propagation

$$x_1, x_2 = \text{split}(x)$$

$$y_1 = x_1$$

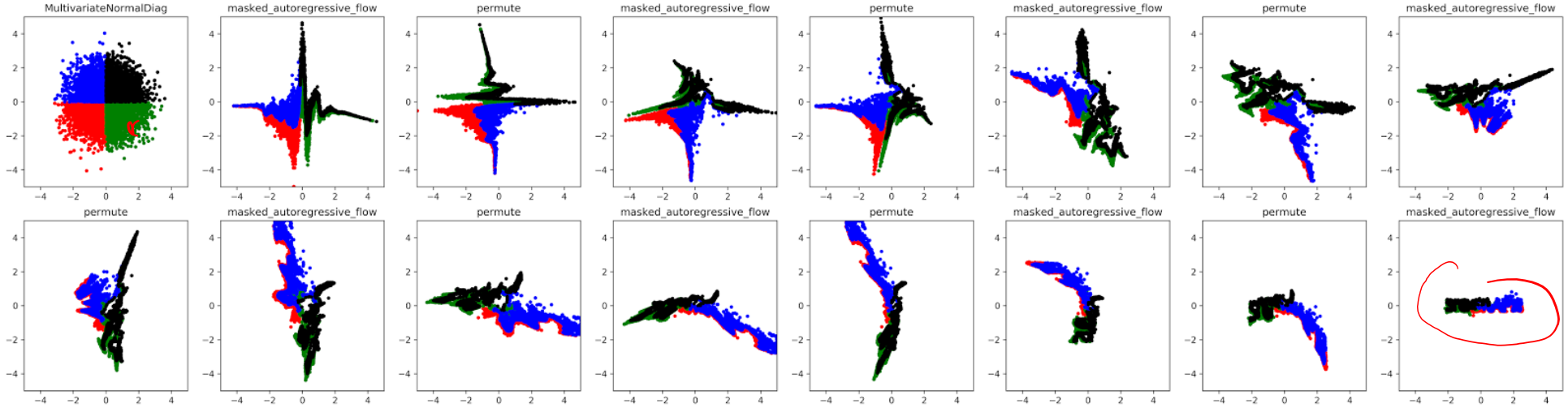
$$y_2 = x_2 s(x_1) + t(x_1)$$

Trivial to invert!

$\frac{\partial y}{\partial x}$ is diagonal, determinant is easy!

L. Dinh et al “NICE”, L. Dinh et al “Real NVP”

Normalizing flows in action



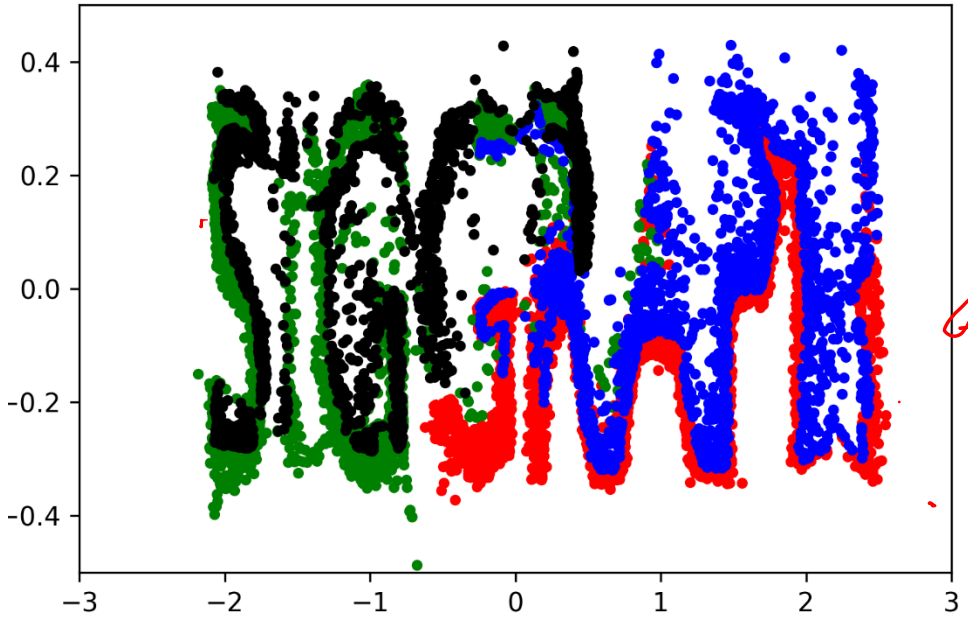
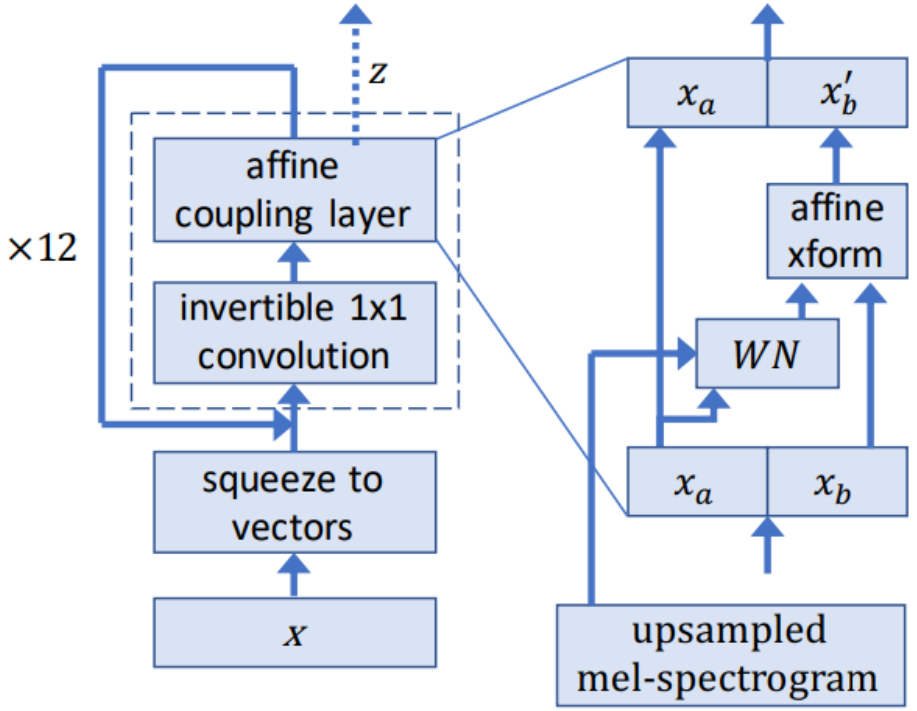


Image credit Eric Jang, https://blog.evjang.com/2018/01/nf2.html

Normalizing flows in action

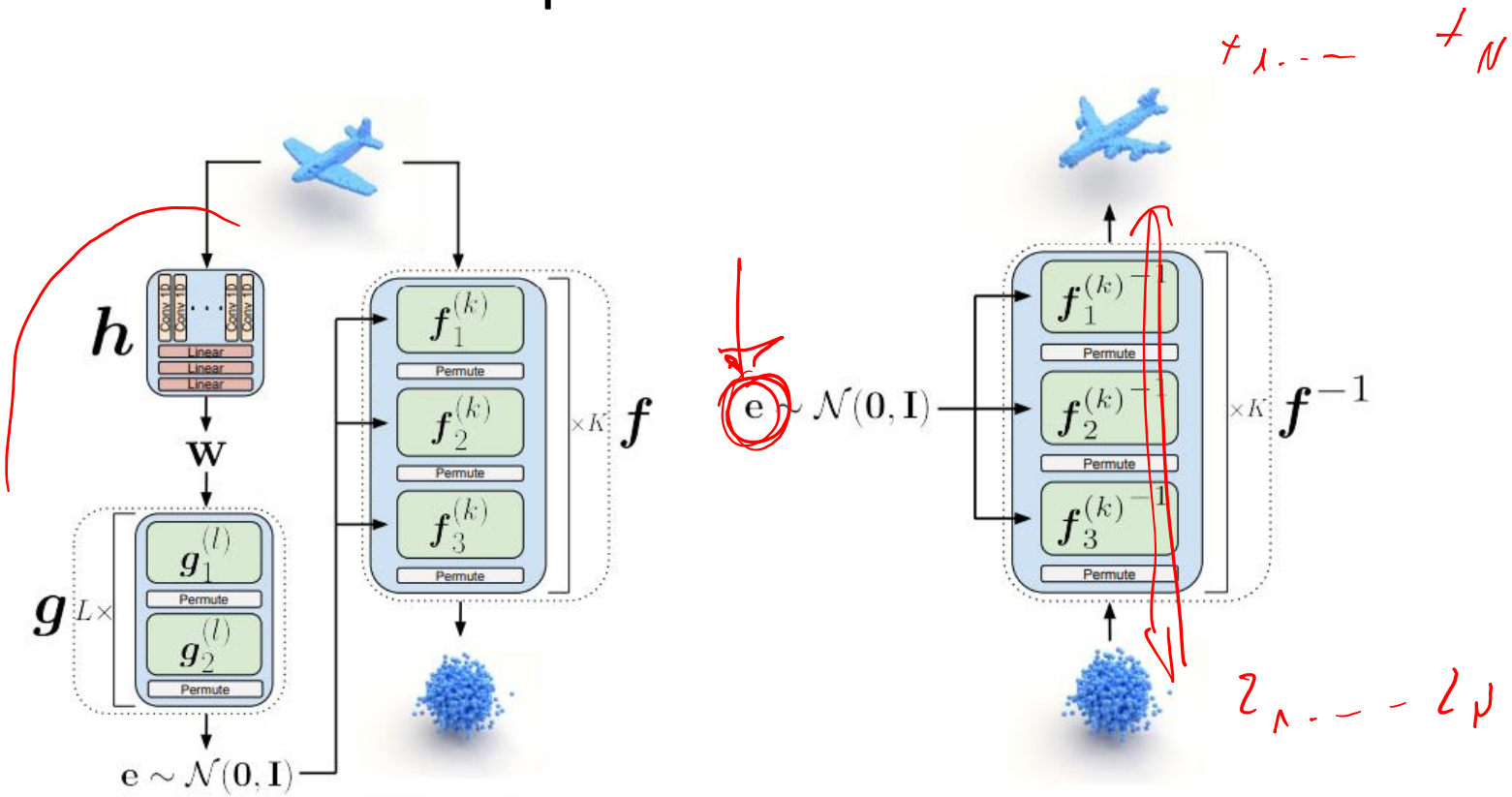
Kingma et al, “GLOW”



Prenger et al, “WAVEGLOW”

Normalizing Flows in Action

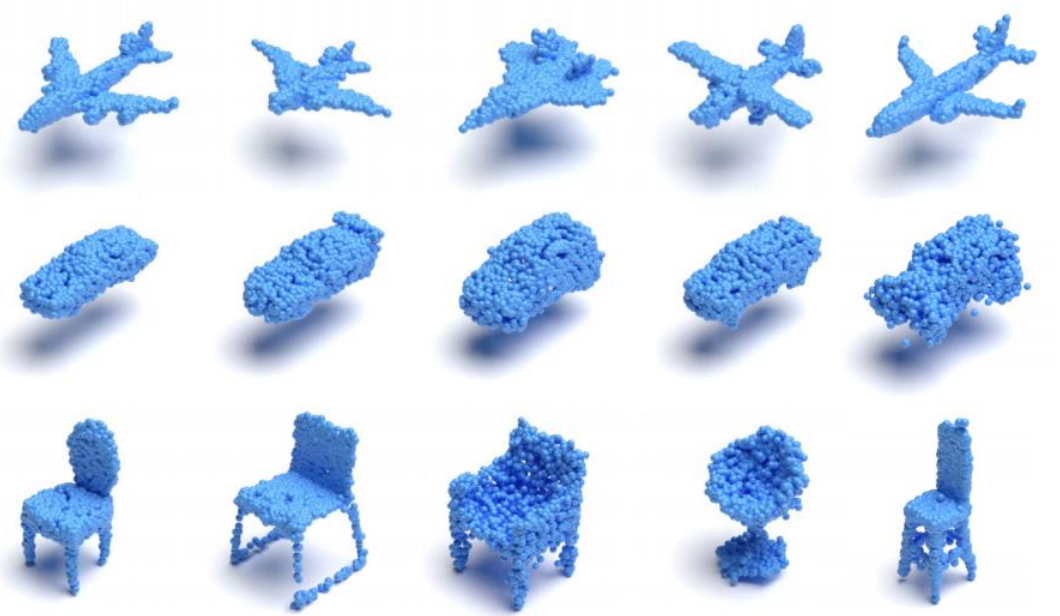
A generative model for point clouds:



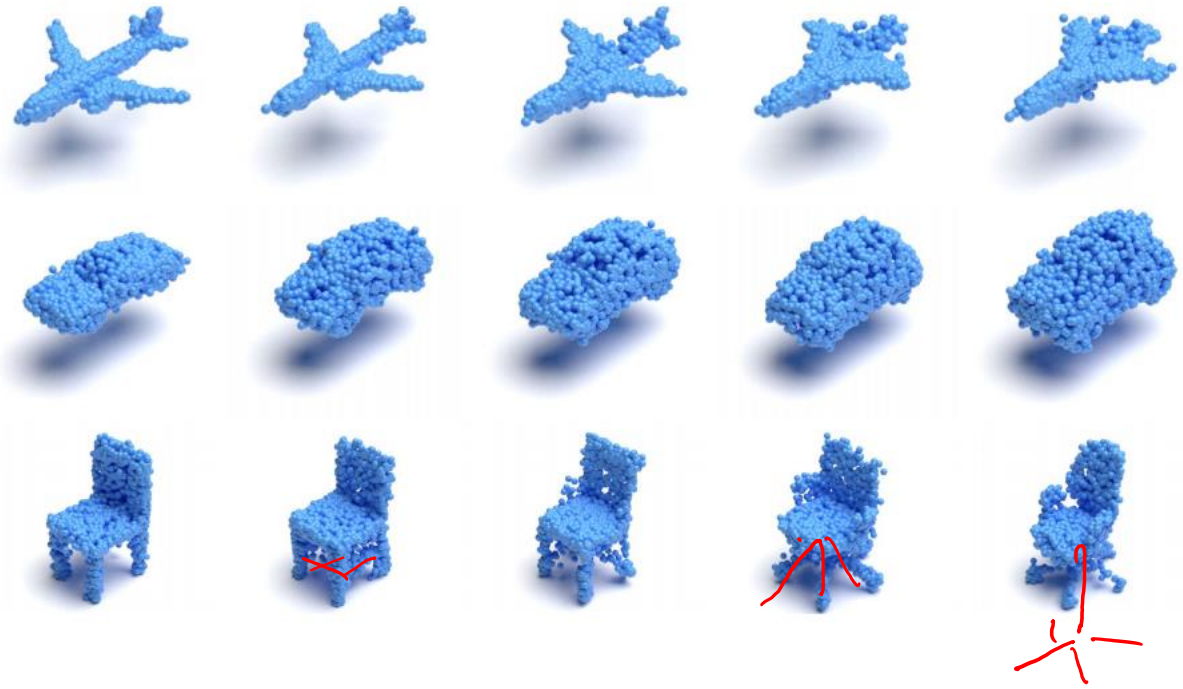
Work with Michał Stypułkowski, Kacper Kania, Maciej Zamorski, Maciej Zięba Tomasz Trzcinski

Normalizing Flows in Action

Samples

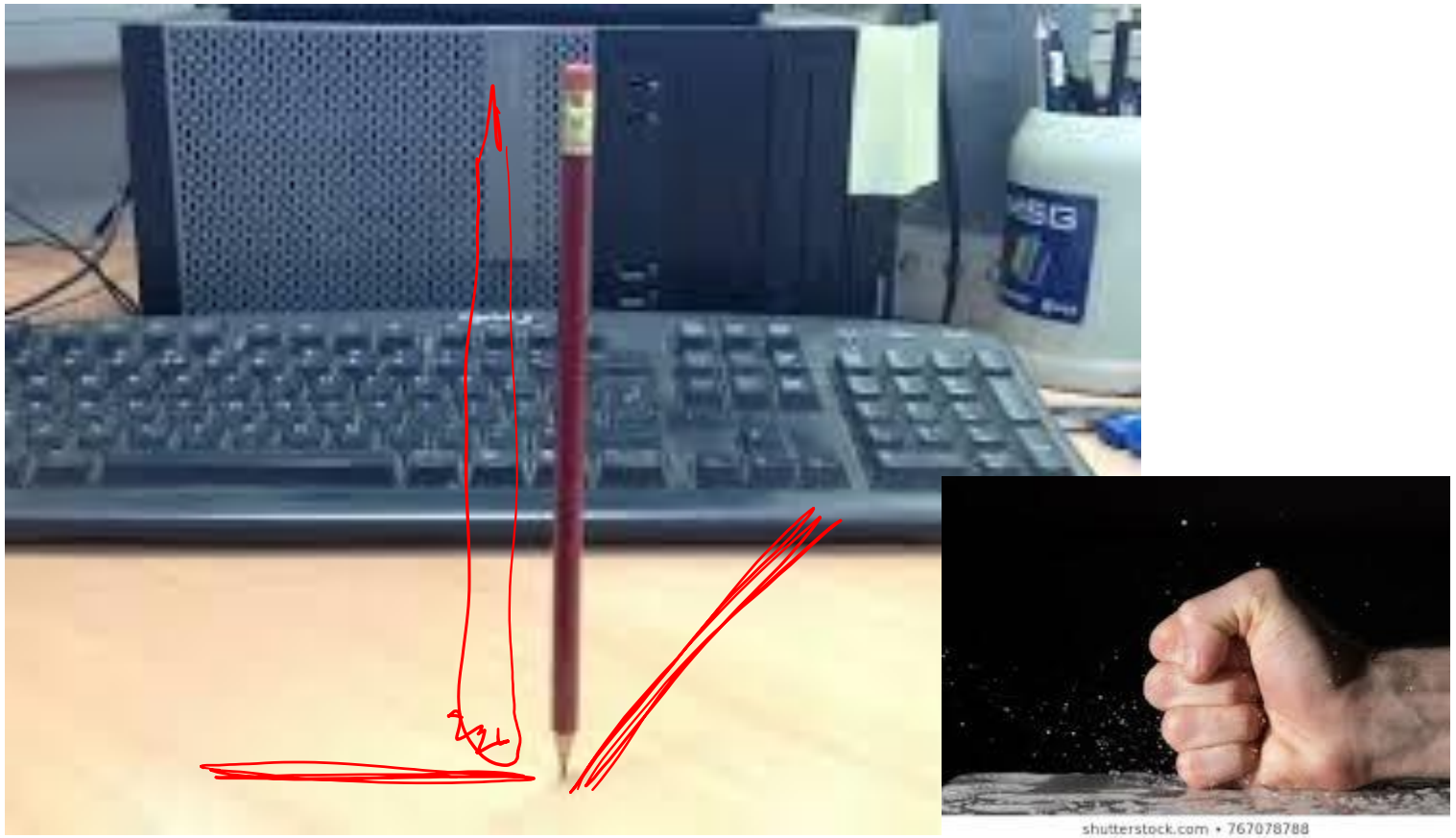


Interpolations

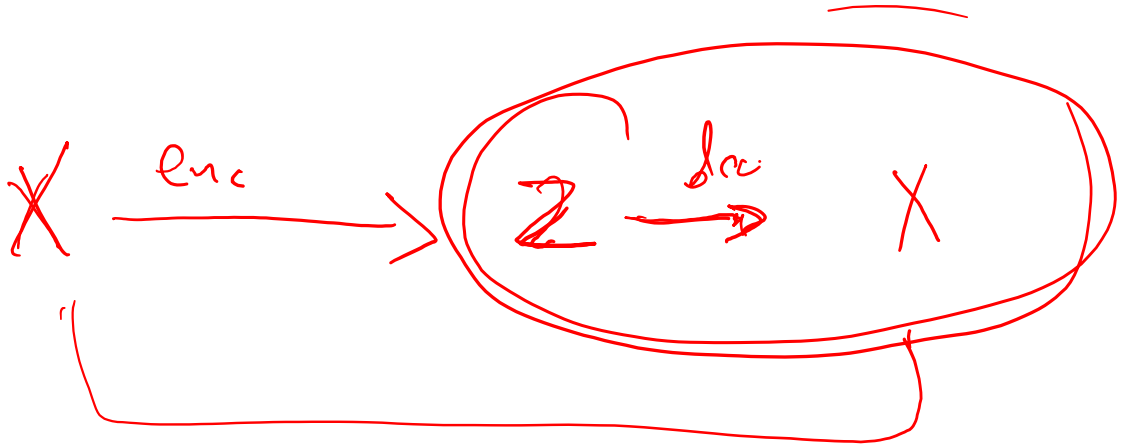


GAN

What will happen next?

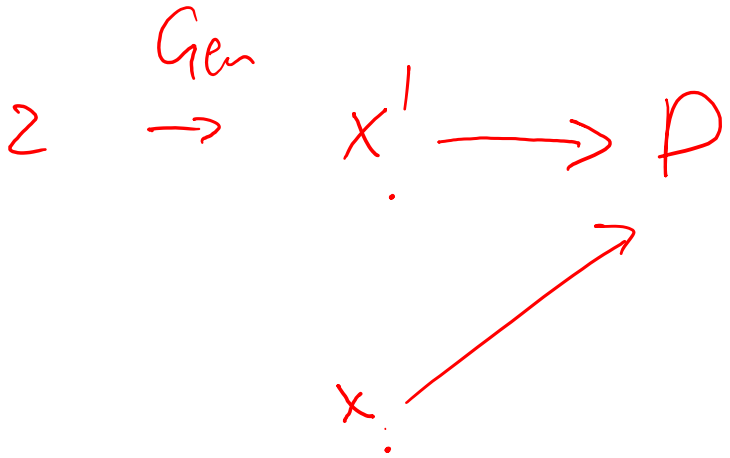


Trouble with autoencoders

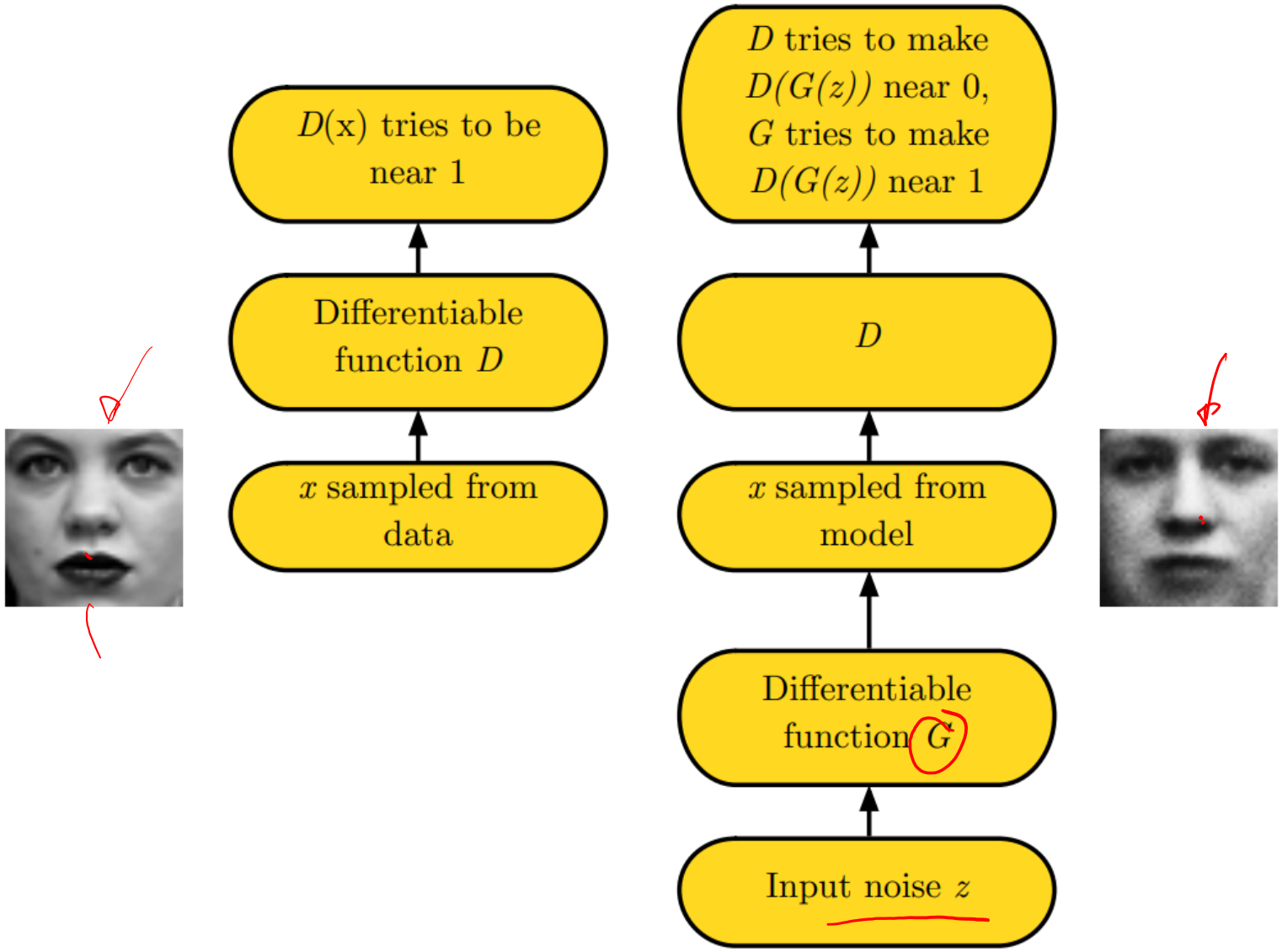


GAN idea

- Learn to tell plausible (i.e. like data) samples from other ones.



Adversarial Nets Framework

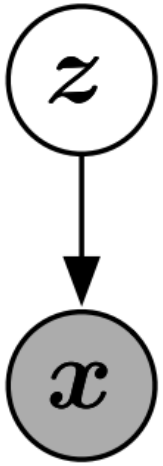


(Goodfellow 2016)

Slide by Ian Goodfellow: https://media.nips.cc/Conferences/2016/Slides/6202-Slides.pdf

Generator Network

$$\boldsymbol{x} = G(\boldsymbol{z}; \boldsymbol{\theta}^{(G)})$$



-Must be differentiable

- No invertibility requirement
- Trainable for any size of z
- Some guarantees require z to have higher dimension than x
- Can make x conditionally Gaussian given z but need not do so

Slide by Ian Goodfellow: https://media.nips.cc/Conferences/2016/Slides/6202-Slides.pdf

Training Procedure

- Use SGD-like algorithm of choice (Adam) on two minibatches simultaneously:
  - A minibatch of training examples
  - A minibatch of generated samples
- Optional: run k steps of one player for every step of the other player.

Slide by Ian Goodfellow: https://media.nips.cc/Conferences/2016/Slides/6202-Slides.pdf

Minimax Game

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log D(\boldsymbol{x}) - \frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log\left(1 - D\left(G(\boldsymbol{z})\right)\right)$$

$$J^{(G)} = -J^{(D)}$$

-Equilibrium is a saddle point of the discriminator loss

-Resembles Jensen-Shannon divergence

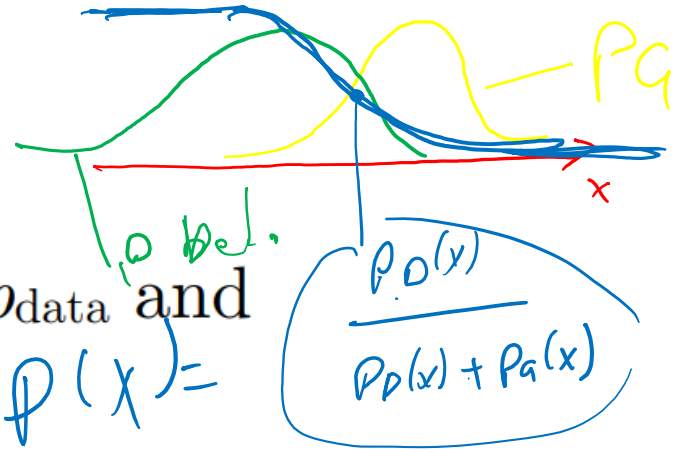
-Generator minimizes the log-probability of the discriminator being correct

Slide by Ian Goodfellow: https://media.nips.cc/Conferences/2016/Slides/6202-Slides.pdf

Exercise 1

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log D(\boldsymbol{x}) - \frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log\left(1 - D\left(G(\boldsymbol{z})\right)\right)$$

$$J^{(G)} = -J^{(D)}$$



- What is the solution to $D(x)$ in terms of p_{data} and $p_{\text{generator}}$?

- What assumptions are needed to obtain this solution?

X : $p_D(x) = p_G(x)$

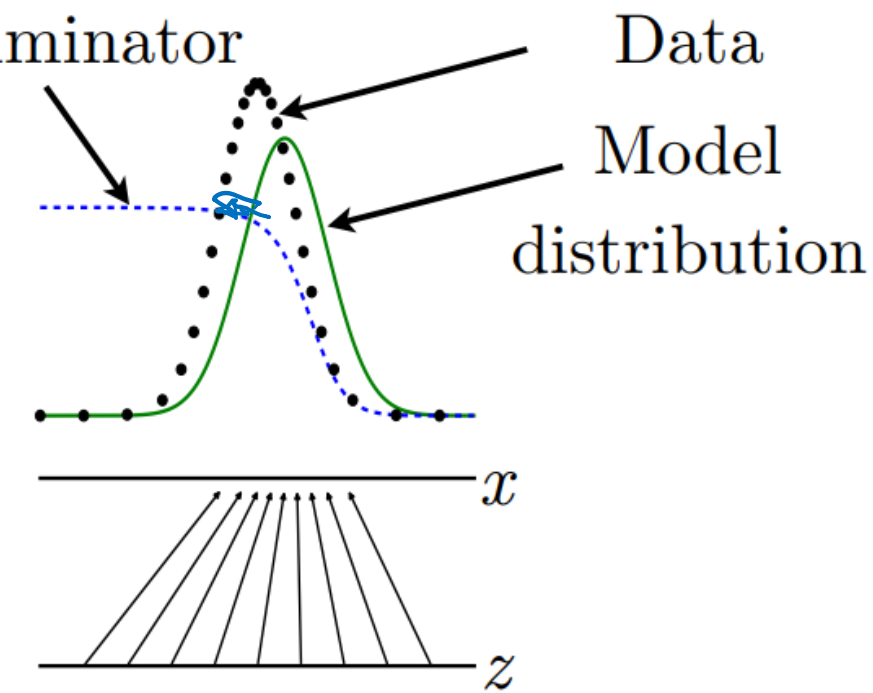
Slide by Ian Goodfellow: https://media.nips.cc/Conferences/2016/Slides/6202-Slides.pdf

Discriminator Strategy

Optimal $D(\boldsymbol{x})$ for any $p_{\text{data}}(\boldsymbol{x})$ and $p_{\text{model}}(\boldsymbol{x})$ is always

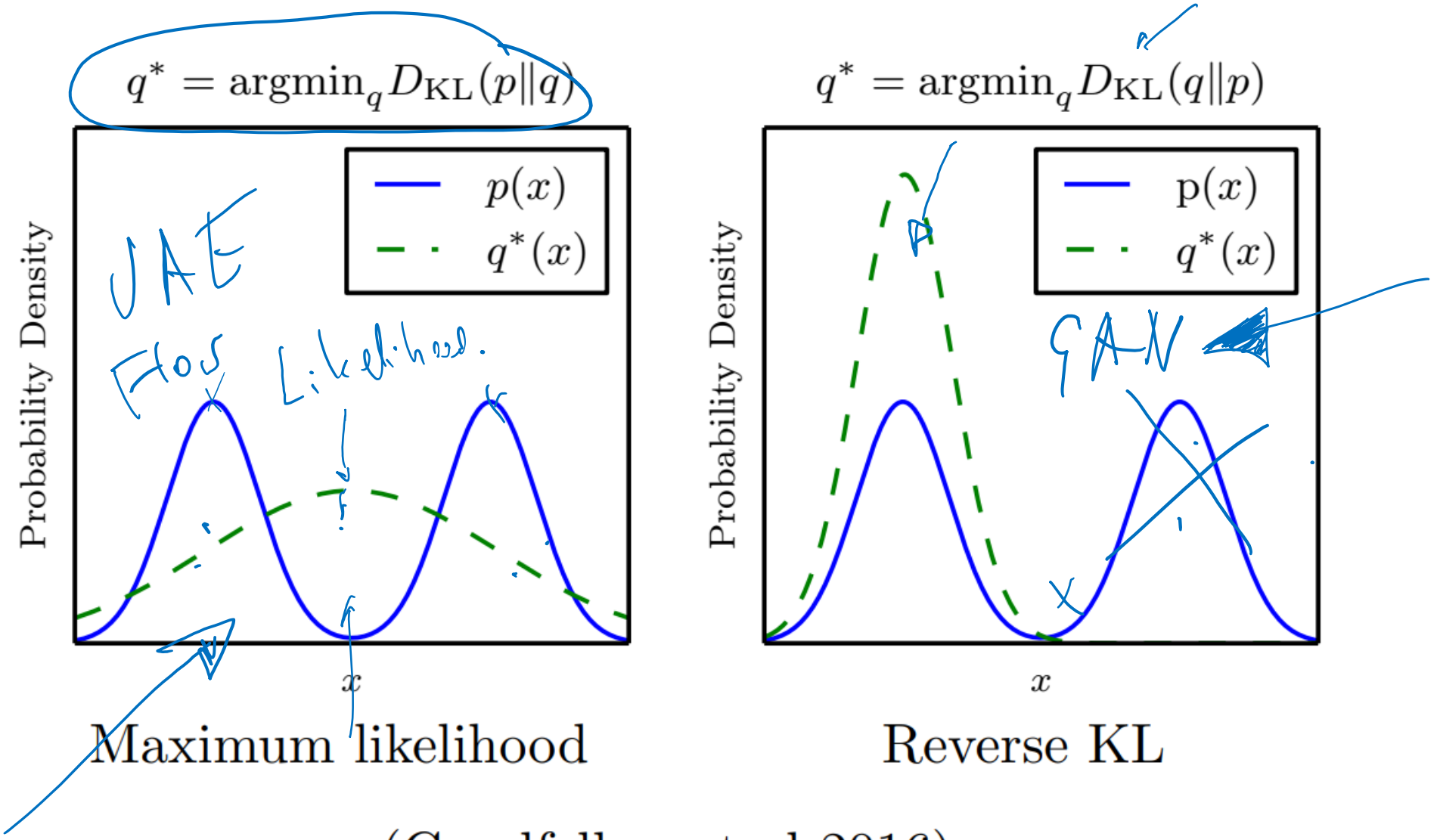
$$D(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{model}}(x)}$$

Estimating this ratio using supervised learning is the key approximation mechanism used by GANs



Slide by Ian Goodfellow: https://media.nips.cc/Conferences/2016/Slides/6202-Slides.pdf

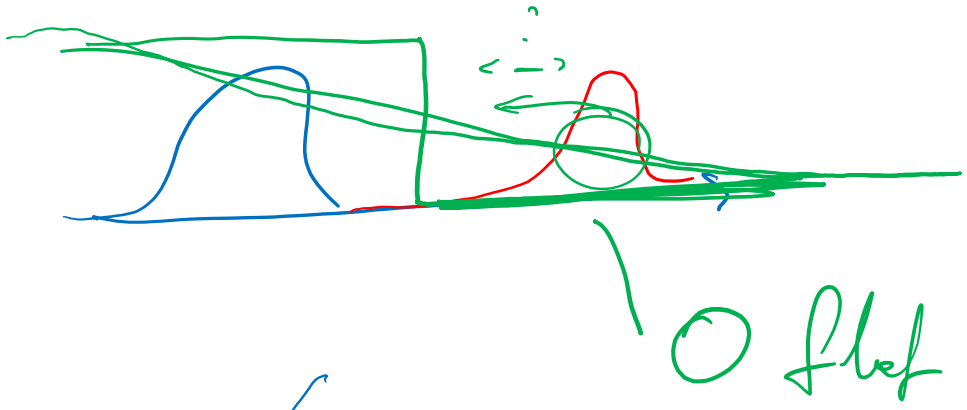
Is the divergence important?



(Goodfellow et al 2016)

Slide by Ian Goodfellow: https://media.nips.cc/Conferences/2016/Slides/6202-Slides.pdf

GAN Trick



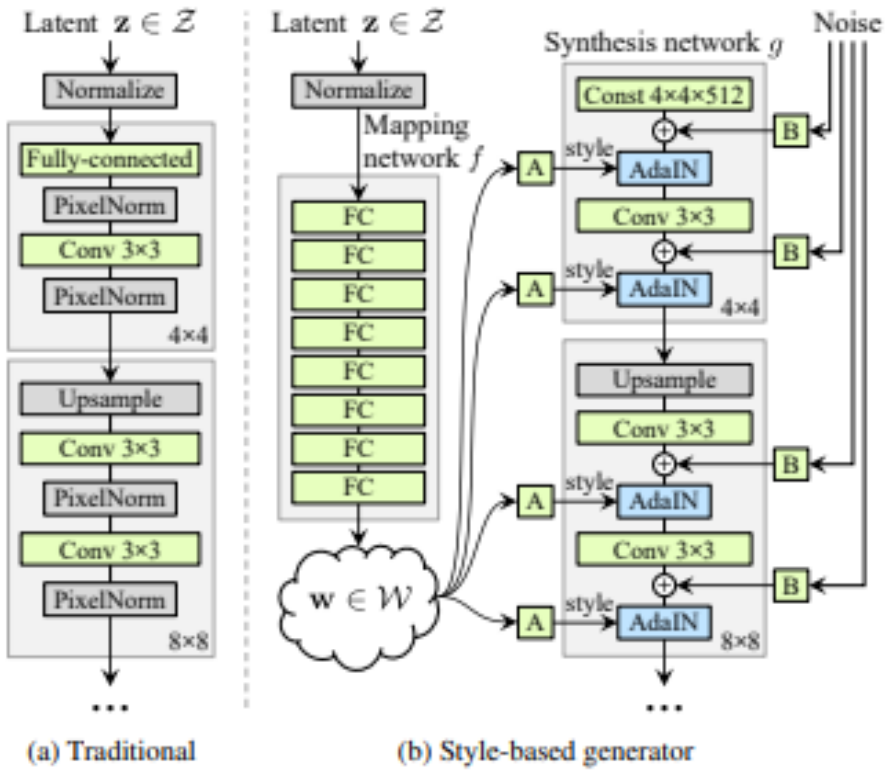
Non-saturating loss

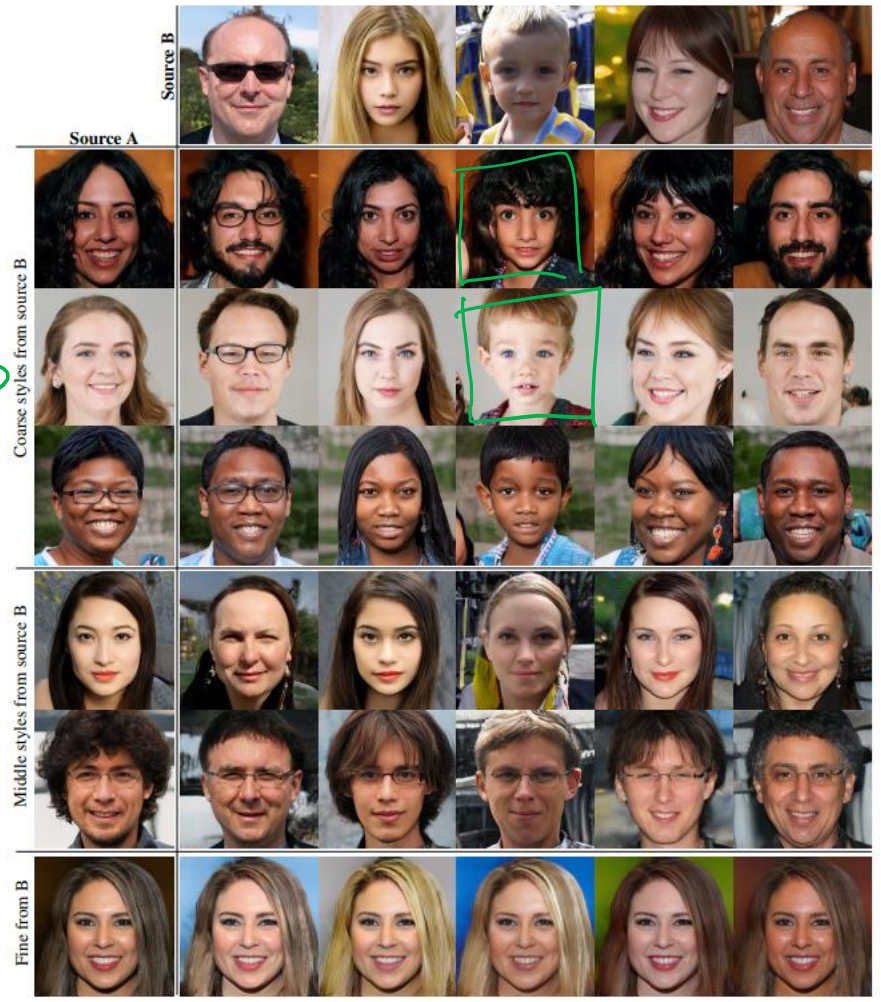
$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log D(\boldsymbol{x}) - \frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log\left(1 - D\left(G(\boldsymbol{z})\right)\right)$$

$$J^{(G)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log D\left(G(\boldsymbol{z})\right)$$

Slide by Ian Goodfellow: https://media.nips.cc/Conferences/2016/Slides/6202-Slides.pdf

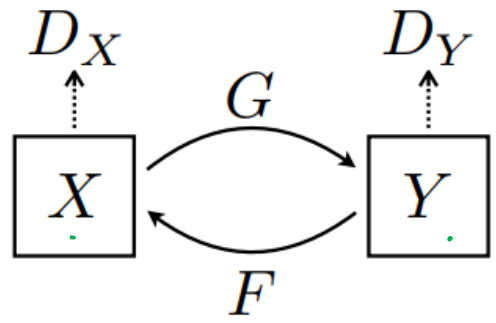
GAN Uses: image generation



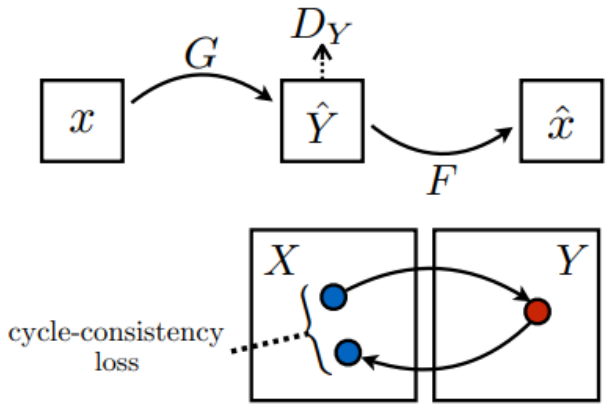


https://arxiv.org/pdf/1812.04948.pdf

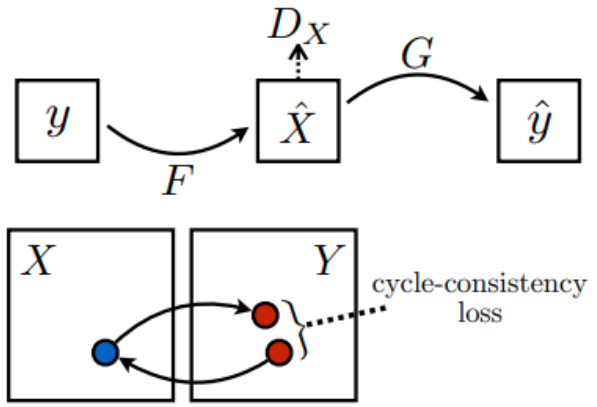
GAN Uses: unpaired image-image translation



(a)



(b)

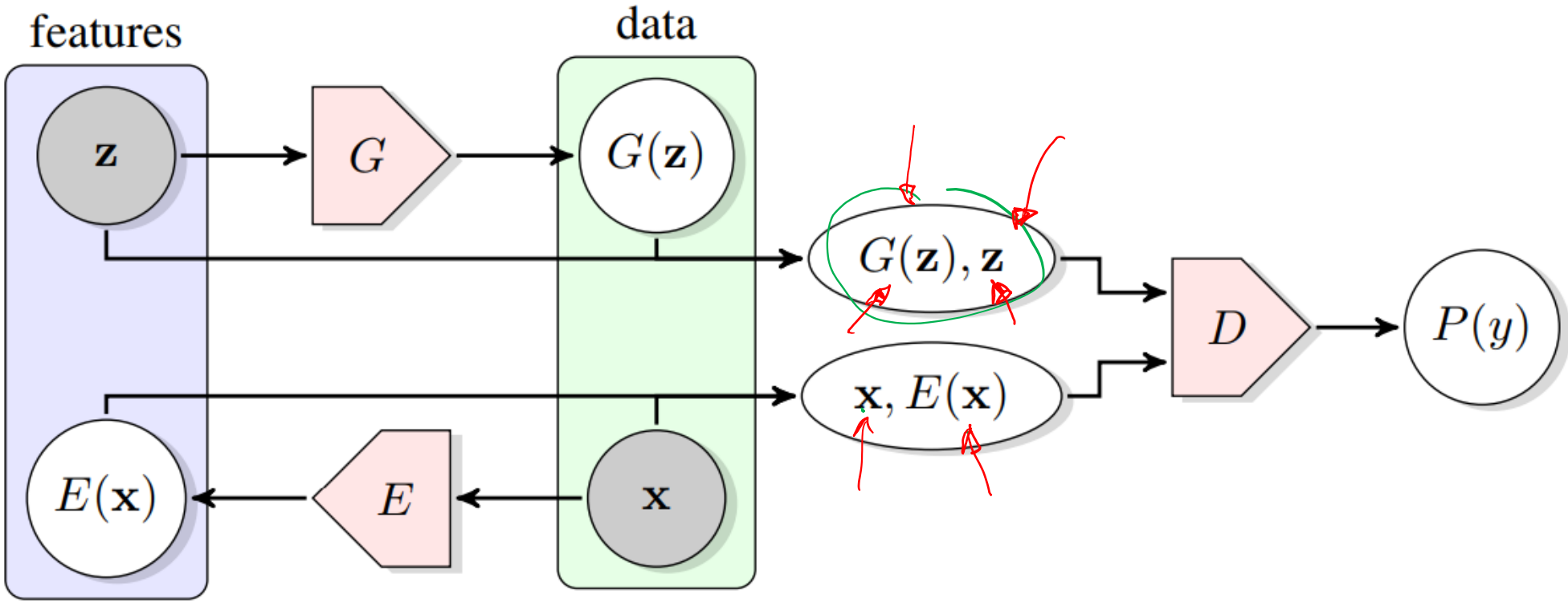


(c)

https://arxiv.org/pdf/1703.10593.pdf

GANs and Latent Variables 1/3

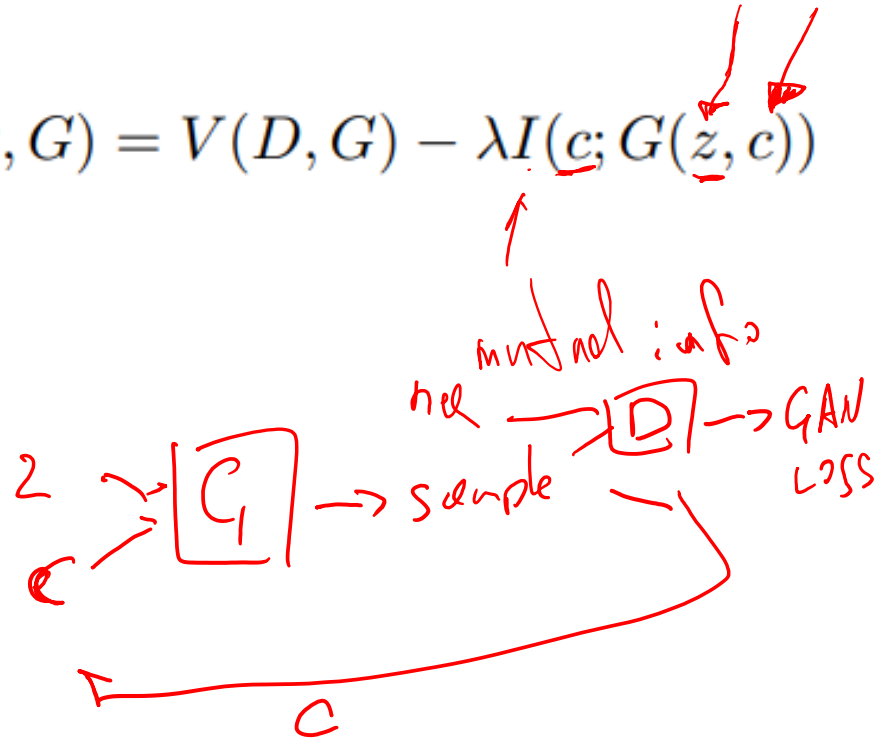
ADVERSARIALLY LEARNED INFERENCE (https://arxiv.org/pdf/1606.00704.pdf)
aka BiGAN (https://arxiv.org/abs/1605.09782)

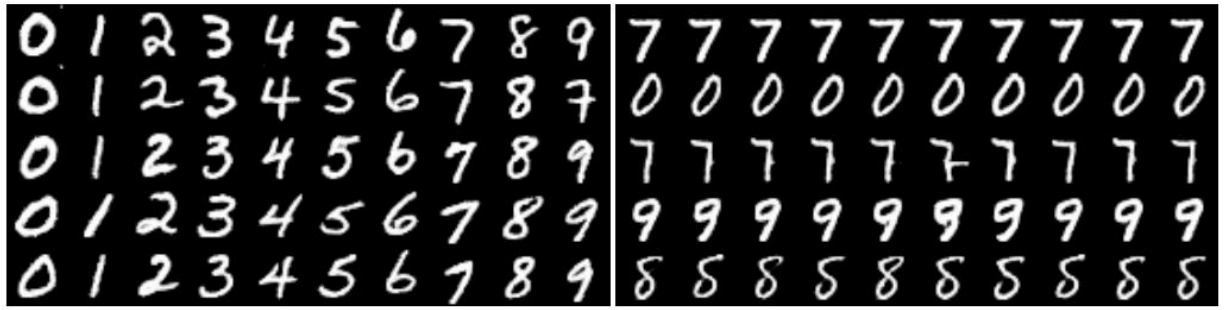


GANs and Latent Variables 2/3

InfoGAN: https://arxiv.org/pdf/1606.03657.pdf

$$\min_G \max_D V_I(D,G) = V(D,G) - \lambda I(c; G(z,c))$$





(a) Varying c_1 on InfoGAN (Digit type)

(b) Varying c_1 on regular GAN (No clear meaning)

(c) Varying c_2 from -2 to 2 on InfoGAN (Rotation)

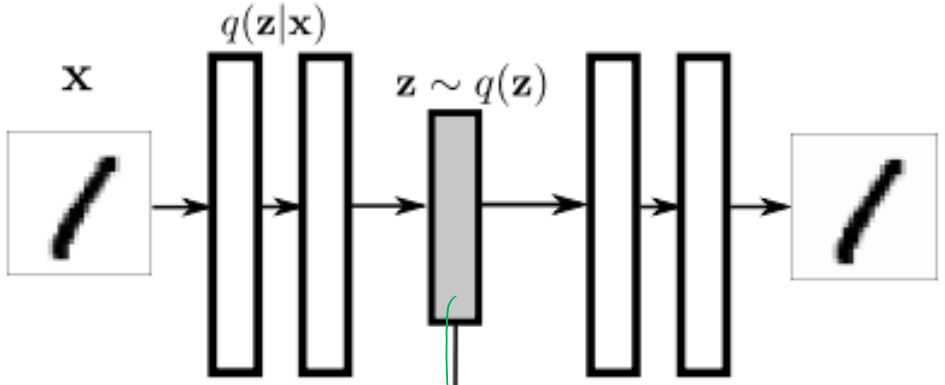
(d) Varying c_3 from -2 to 2 on InfoGAN (Width)

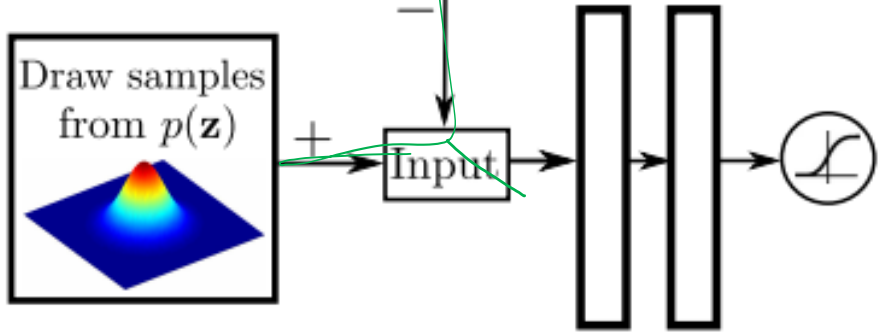
GANs and Latent Variables 3/3

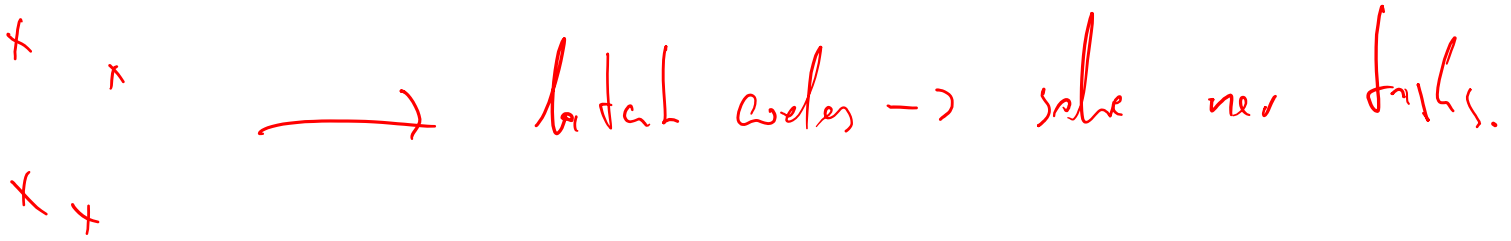
Adversarial Autoencoders (https://arxiv.org/pdf/1511.05644.pdf)

Like VAE, enforce latents to have a simple prior distribution

Unlike VAE, there is no information loss – encoding uses full precision

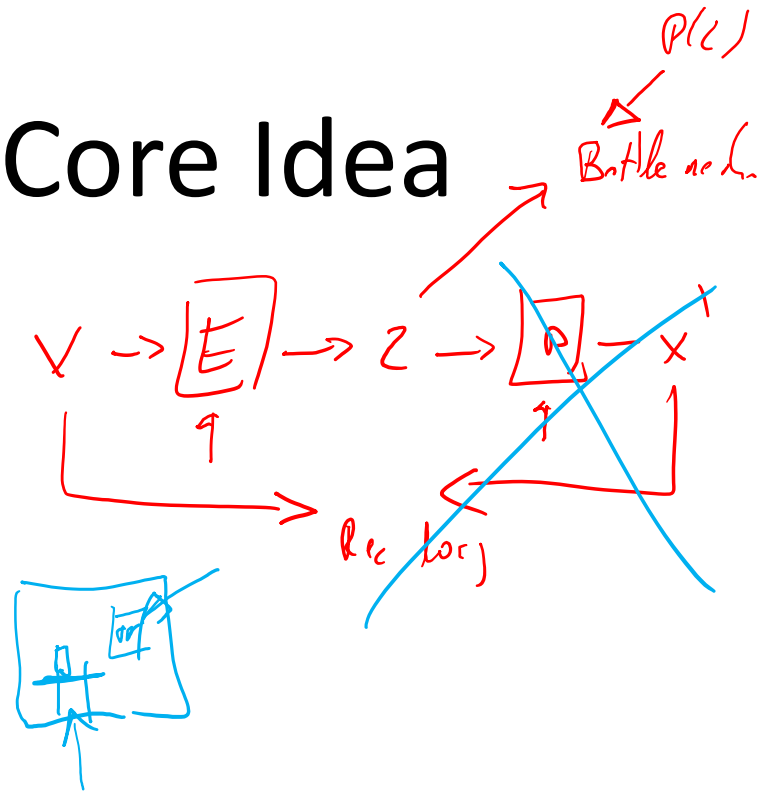






THE FUTURE OF UNSUPERVISED: SELF-SUPERVISED LEARNING

Self-supervised learning: Core Idea



We usually train an Encoder-Decoder pair

But later, we only use/keep the encoder!!

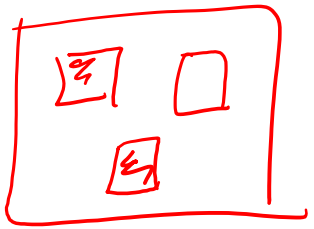
This is wasteful – the decoder is often complex (and may even be more expensive to train than the encoder)

Q: Can we train an encoder-only system?

Self-supervised learning: Core Idea

Q: Can we train an encoder-only system?

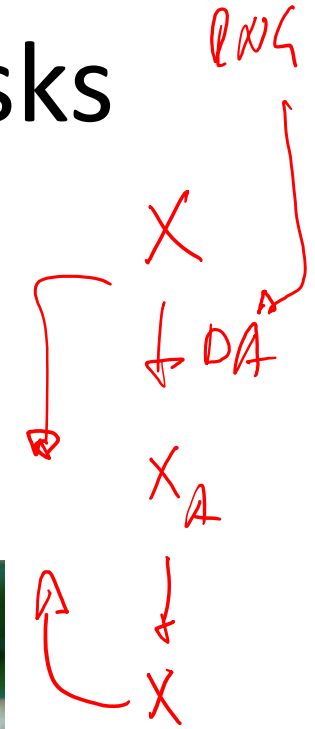
Yes, then use its output to solve a task requiring data understanding, for which labels are easy to get.



E.g.:

- cut image into parts, predict their relative location.
- cut video into parts, predict time direction
- cut sentence into words, predict neighbors (quick, what was the name of this model??) → Word 2 Vec
- Take a sentence, mask some words, train the model to impute them (again – what was the name of the model?) ← BERT

Self-supervised learning: good proxy tasks



Some domains have known invariances

E.g. images: rotation, scaling, recolorization, mild crop, etc.

only slightly change the contents of an image

Random:
- Color transformation
- Affine transform
- Crop

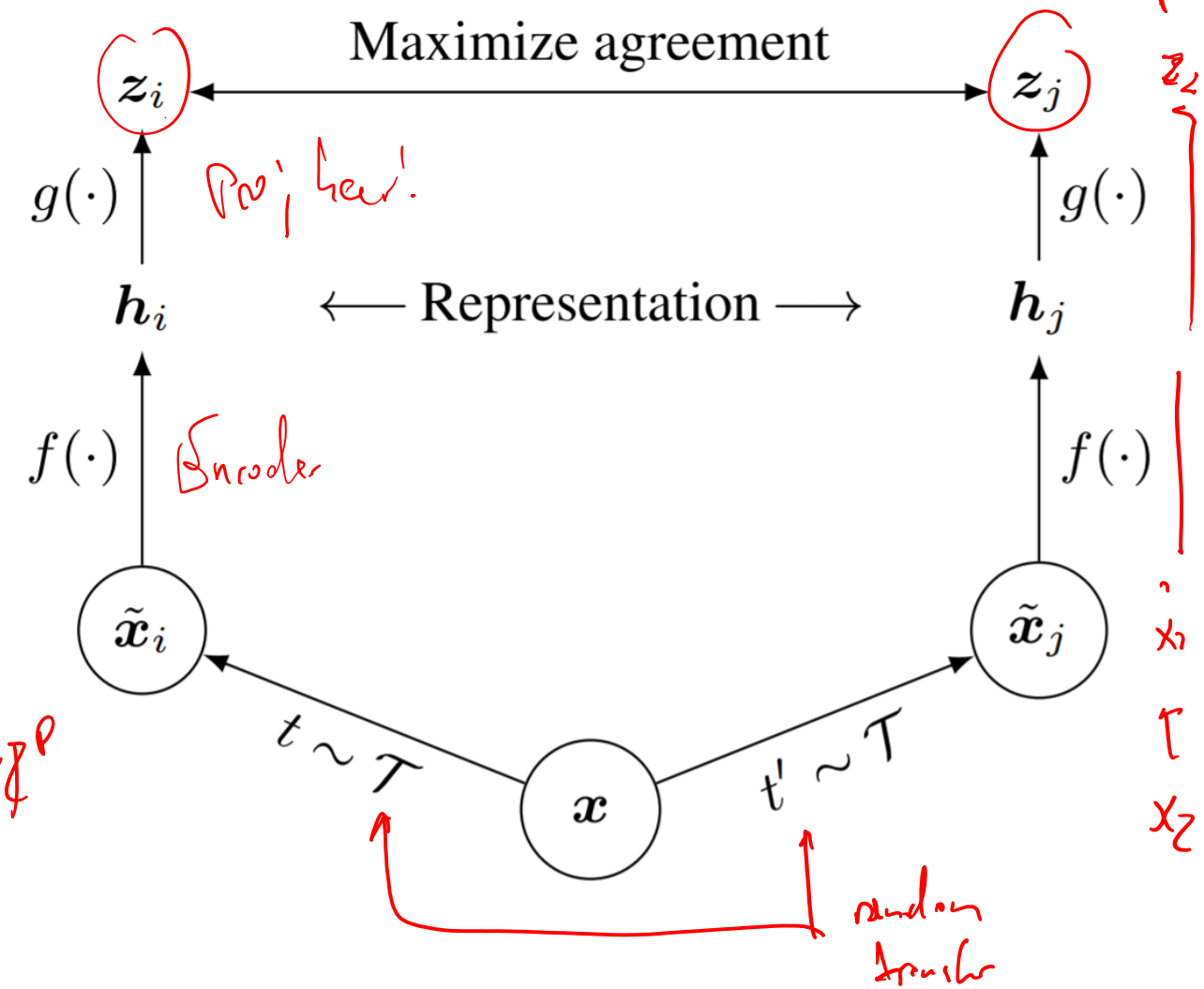
Easy to implement,
No labels needed!

Q: how to teach the model to be invariant?

SimCLR: Self-learning invariances

Idea:

- Maximize agreement between two augmentations of the same input
- Q: what prevents the model from learning $f(x) = 0$?



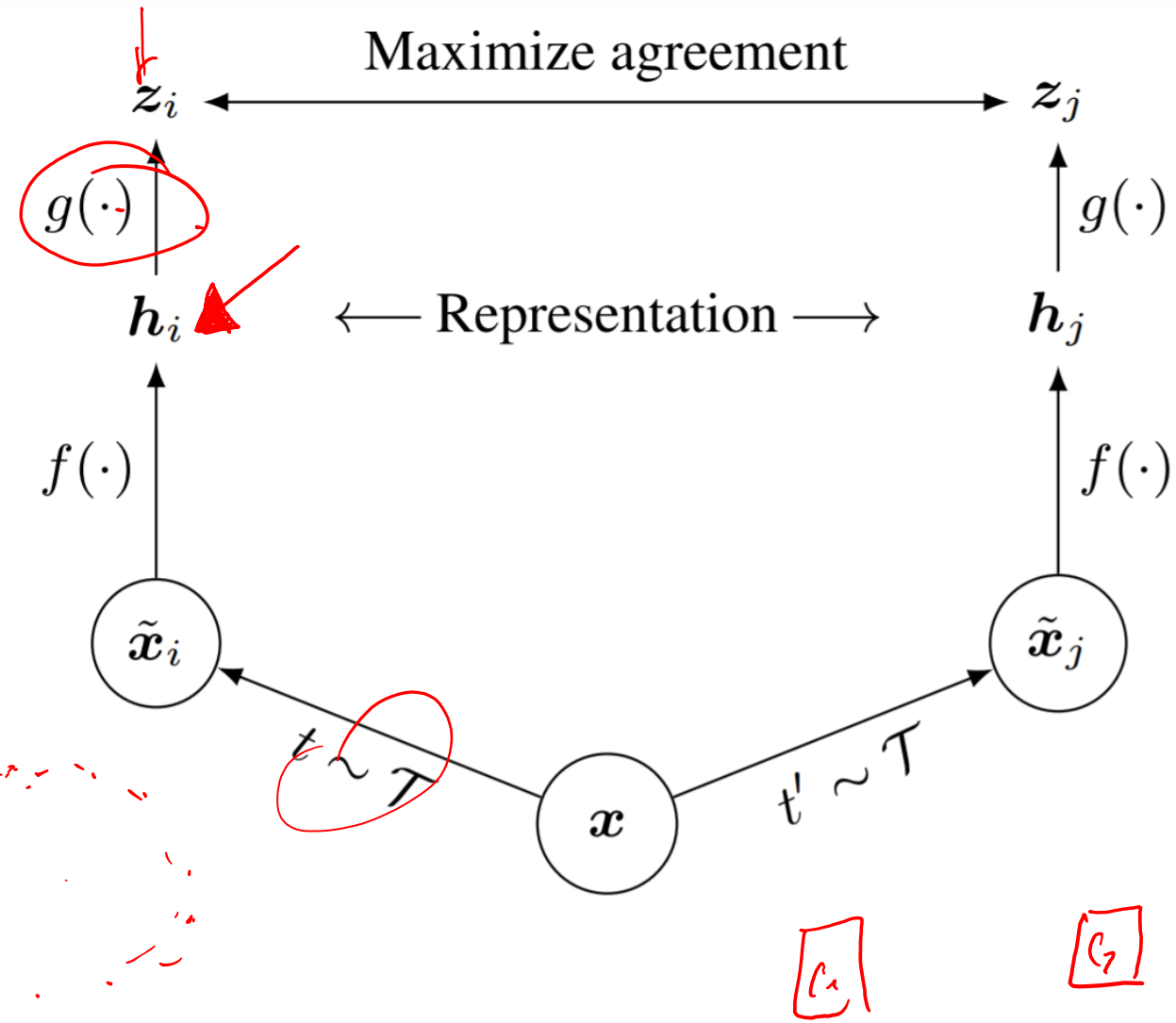
https://arxiv.org/pdf/2002.05709.pdf

SimCLR: contrastive losses

Q: what prevents the model from learning $f(x) = 0$?

A: the net is trained to pull the two augmented variants of x together…

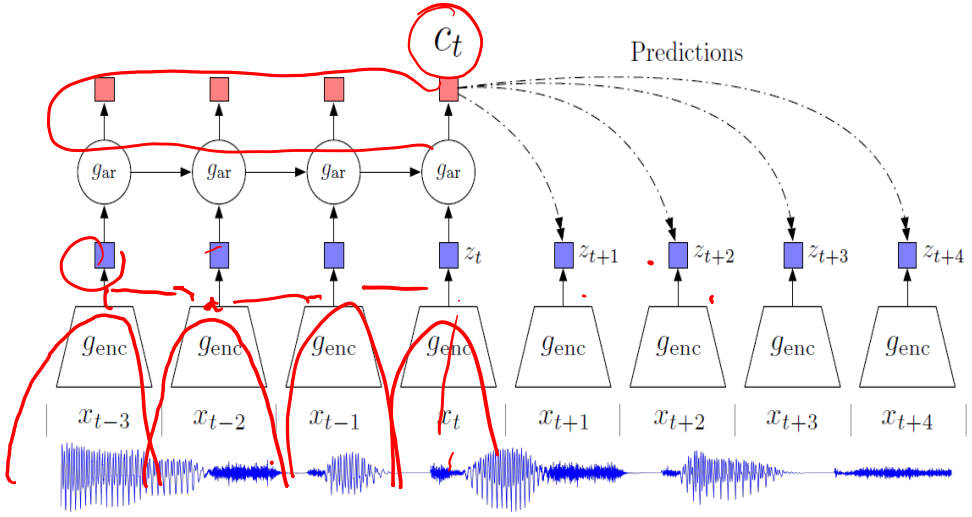
And push them away from other samples (contrastive learning)

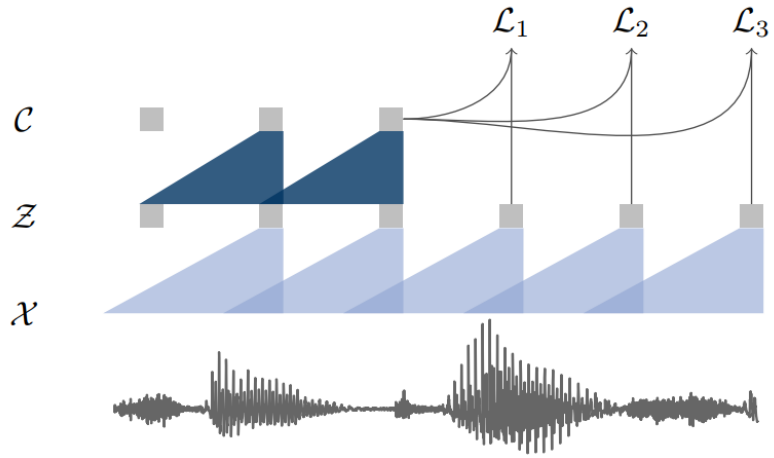


https://arxiv.org/pdf/2002.05709.pdf

CPC: contrastive training to predict the future

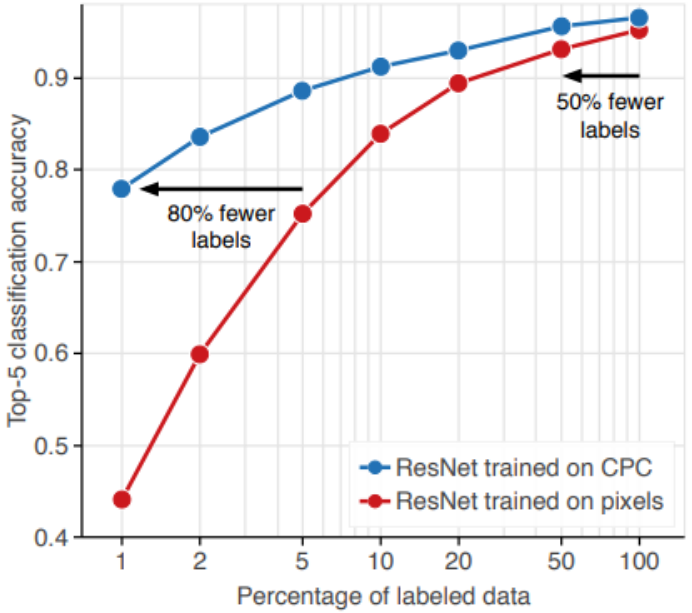
- Encode speech frames into latent vectors
- Conbine past frames using g_{ar}
- Predict upcoming frames using a contrastive loss (tell apart the next frame from a randomly sampled one)



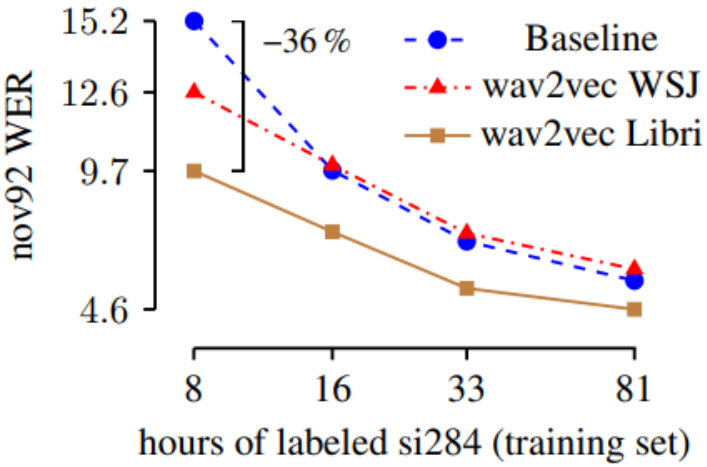


Oord et al. „Representation Learning with Contrastive Predictive Coding"
Schneider et al. „wav2vec: Unsupervised Pre-training for Speech Recognition"

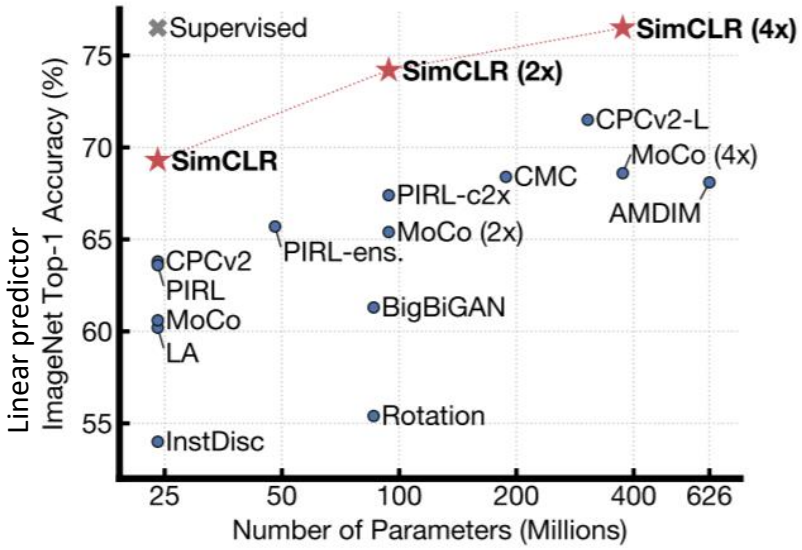
Self-supervised learning becomes practical



DATA-EFFICIENT IMAGE RECOGNITION WITH CONTRASTIVE PREDICTIVE CODING Olivier J. Henaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, ´ Carl Doersch, S. M. Ali Eslami, Aaron van den Oord



WAV2VEC: UNSUPERVISED PRE-TRAINING FOR SPEECH RECOGNITION Steffen Schneider, Alexei Baevski, Ronan Collobert, Michael Auli



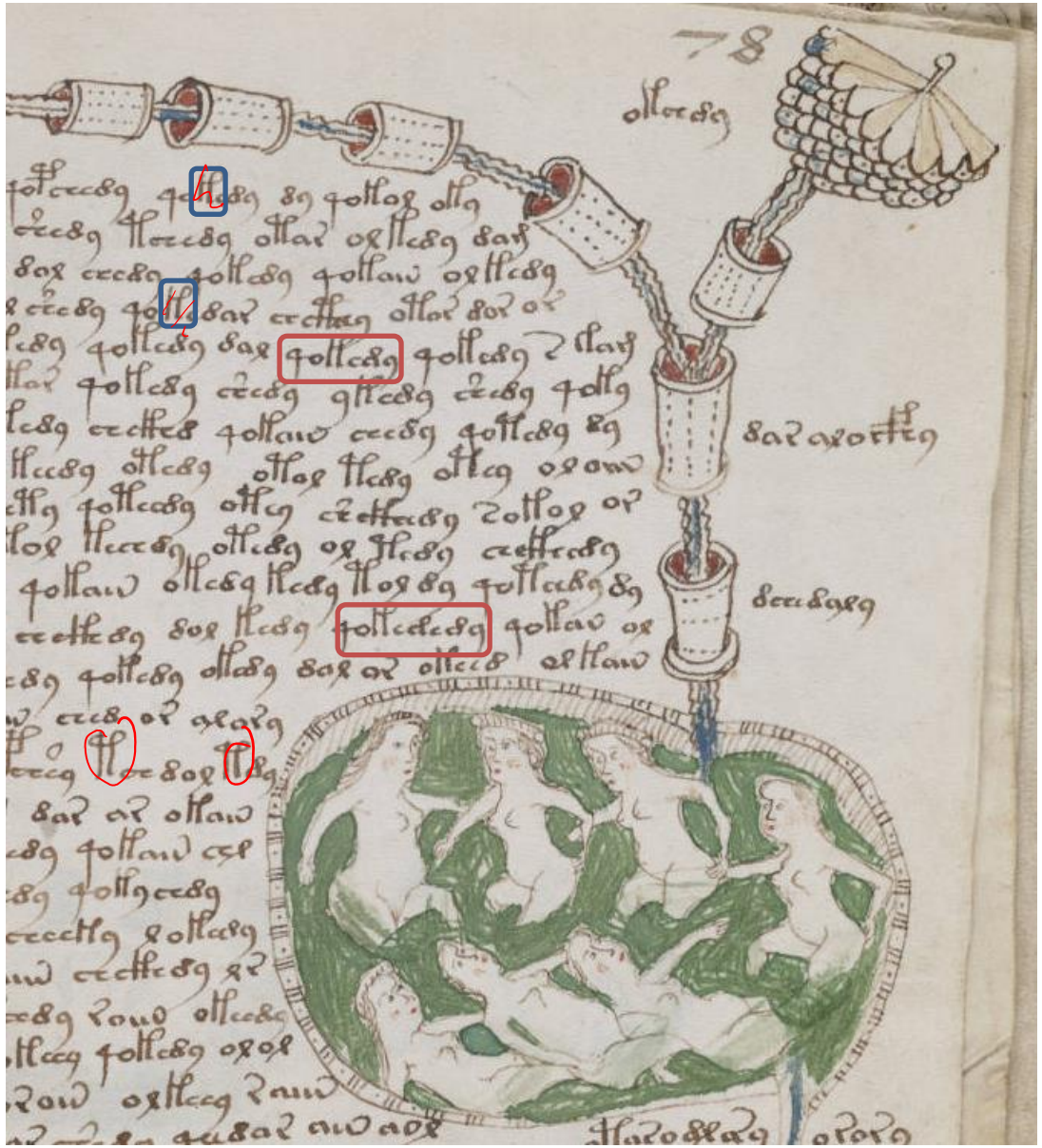
A Simple Framework for Contrastive Learning of Visual Representations Ting Chen, Simon Kornblith, Mohammad Norouzi, Geoffrey Hinton

Summary for unsupervised learning

- We gan generate nearly realistic samples with deep generative models!
- We can use unsupervised / self-supervised learning to learn good data representations
- We can approximately control the contents of the latent representation (e.g. separate phone/gender)

MY WORK:
VAE + AUTOREGRESSIVE DECODERS

Unlabeled data uses

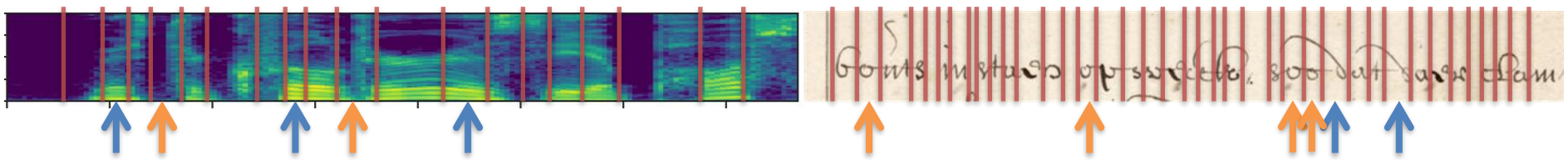


What can we learn without labels:

- Features (good data representations)
- Units (characters)
- Word types (sequences of units)
- Their co-occurrence patterns (structure, e.g. bigram stats)

The goal

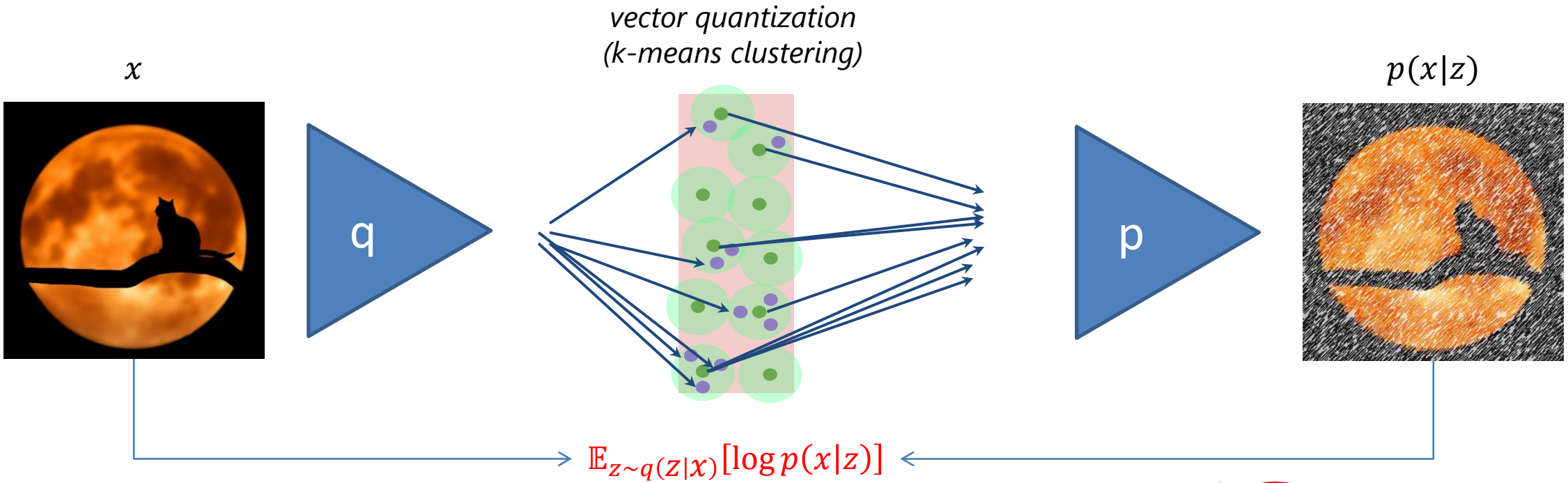
Unsupervised unit discovery
in speech and handwriting



1. Discover information-bearing units
   (cluster input segments)
2. Be invariant to other factors of variation
   (speaker/scribe, style, etc)

Extreme case of representation learning!
We will combine autoencoders with generative models.

The Vector Quantized Autoencoder



q produces a vector

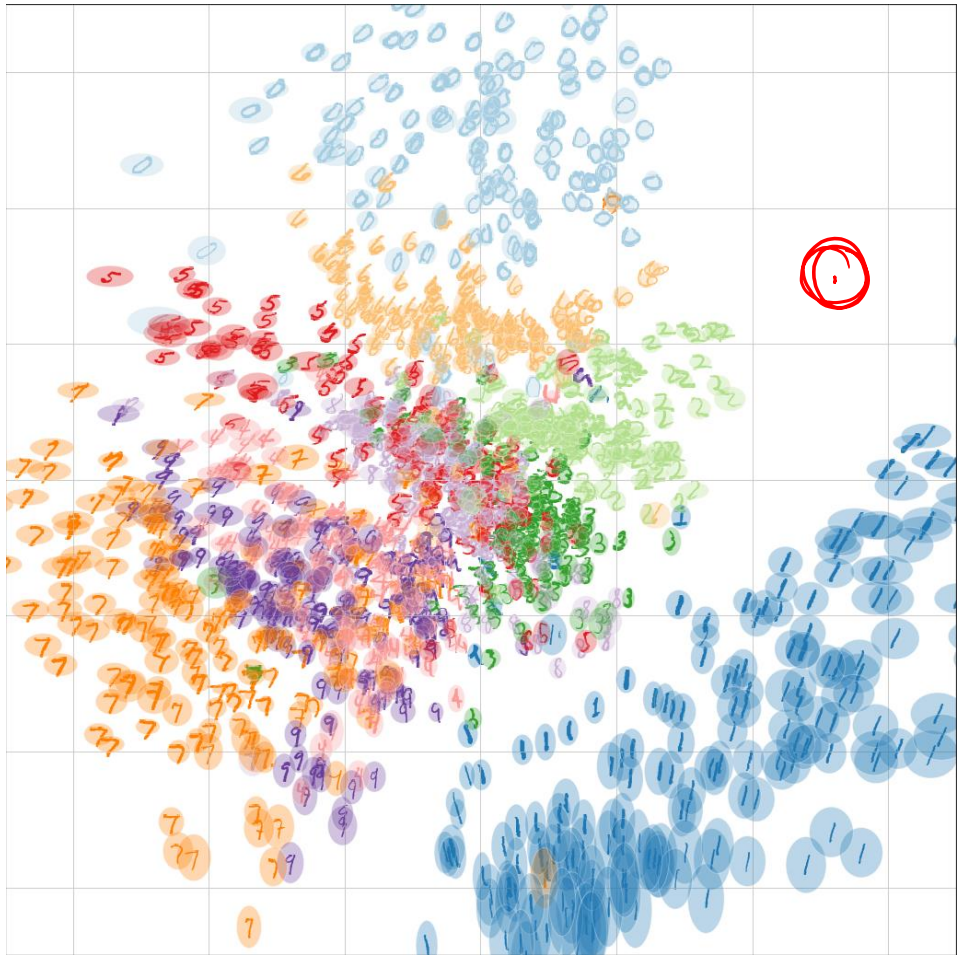
it is replaced by one of K prototypes

Information loss through quantization (rounding)

Recall: in VAE each z has a volume

Each sample is represented as a Gaussian

This discards information (latent representation has low precision)



VQVAE – deterministic quantization, each z is rounded

Limit precision of the encoding by quantizing (round each vector to a nearest prototype).

Output can be treated:

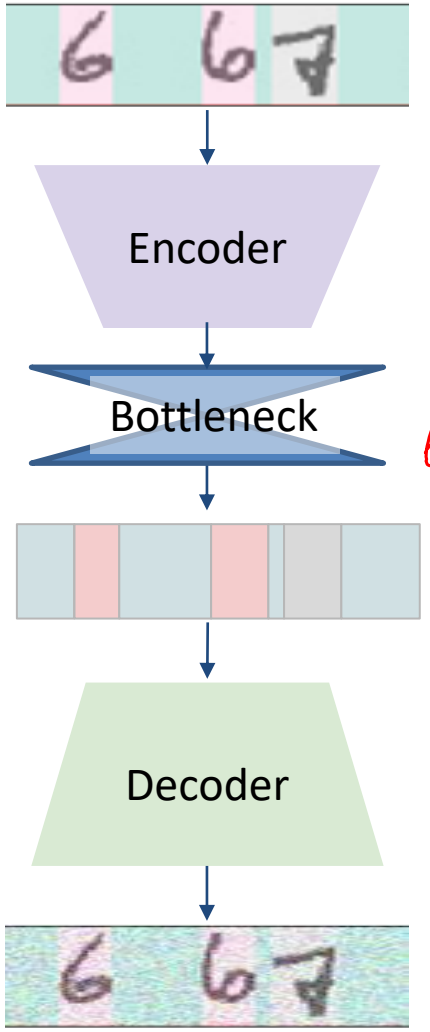
- As a sequence of discrete prototype ids (tokens)
- As a distributed representation (the prototypes themselves)

Train using the straight-through estimator,

with auxiliary losses:

$$\mathcal{L} = \log p(x \mid z_q(x)) + \|\mathrm{sg}(z_e(x)) - e_{q(x)}\|_2^2 + \gamma\|z_e(x) - \mathrm{sg}(e_{q(x)})\|_2^2$$

Autoencoders and missing information



Detailed input:
handwriting or speech

Encoder with bottleneck removes information, enforces segmentation, and quantizes the representation

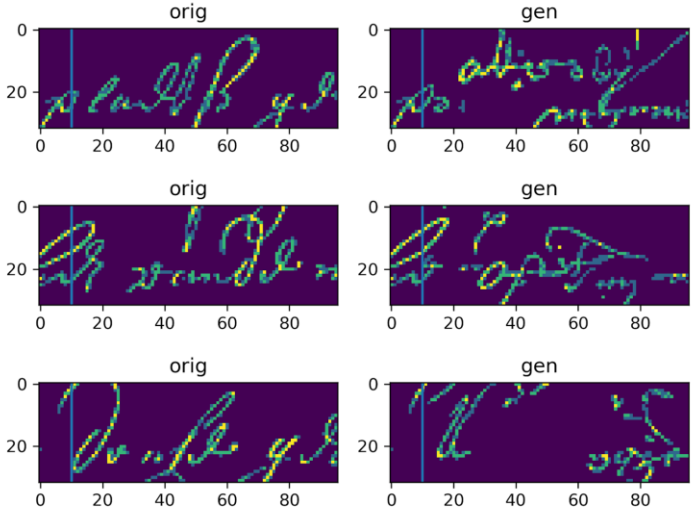
Latent encoding:
only high level features

Decoder regenerates the inputs.

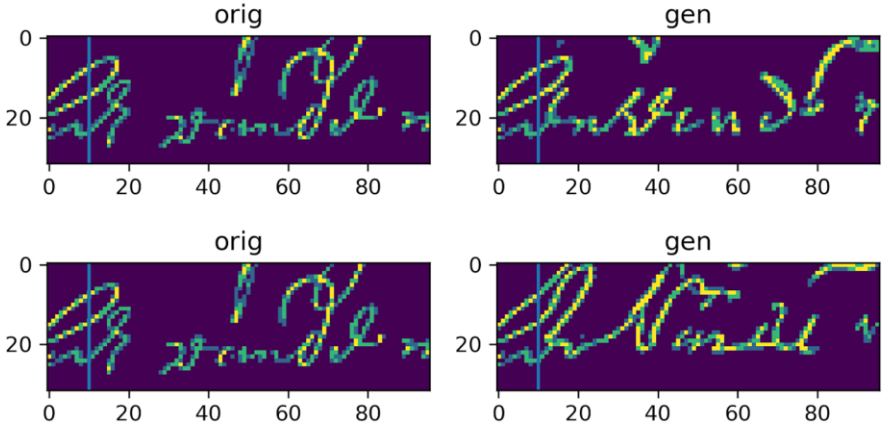
It learns $p(x|z)$ and it must fill in all discarded information.

Conditional PixelCNN can regenerate images.

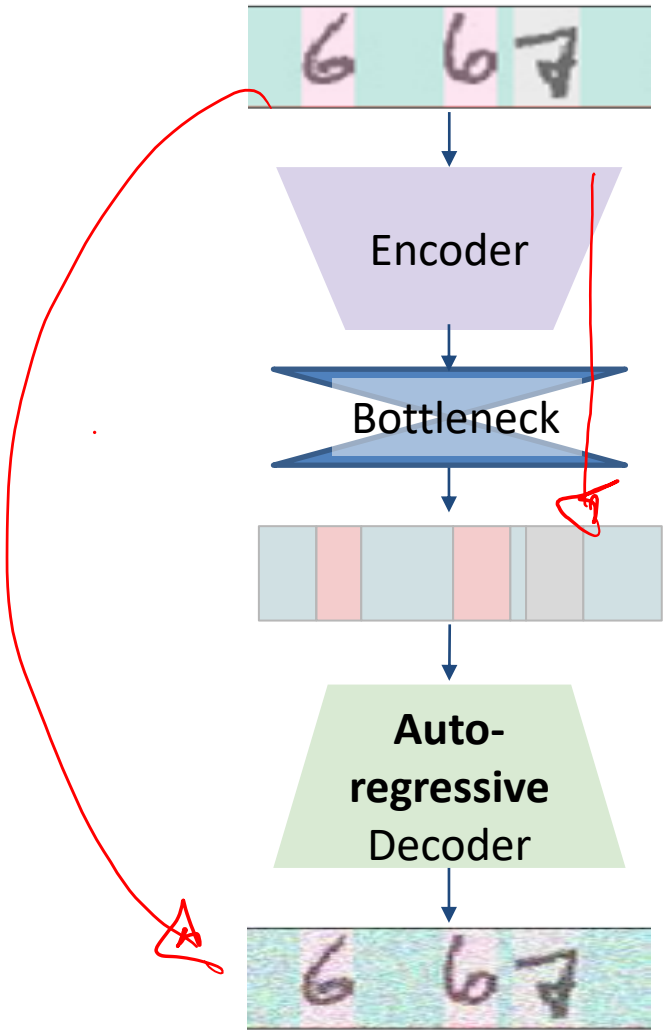
Unconditional



Conditioned on text



Our model:
Autoencoder + autoregressive decoder

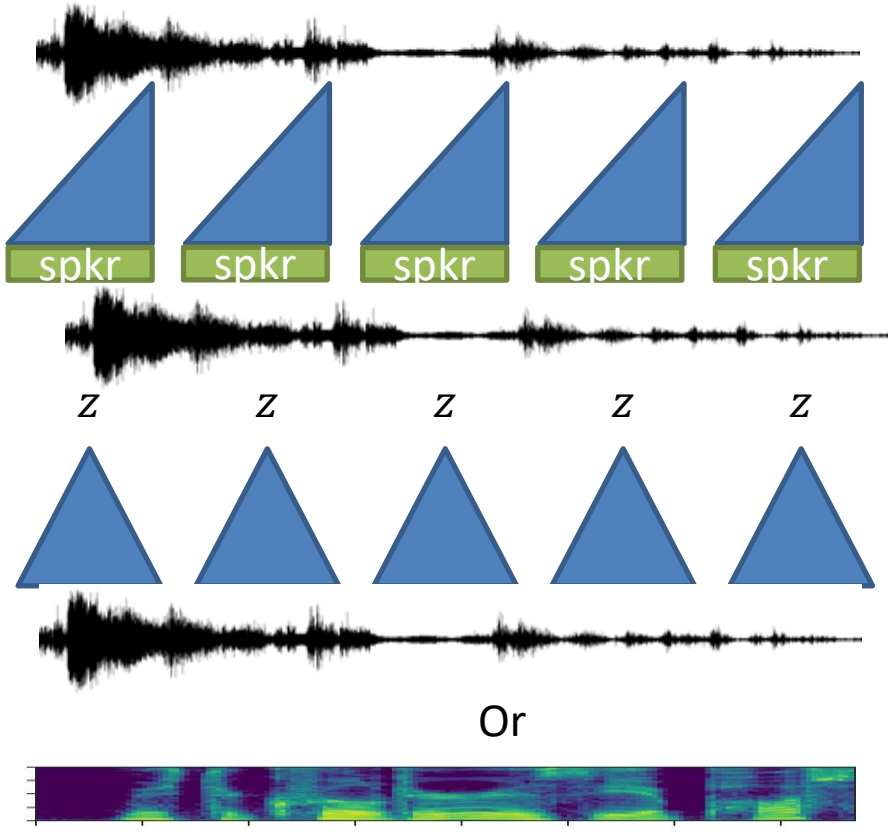


Decoder computes a probability over input images/waveforms conditioned on the encoding.

It combines information from:

- latent representation
  (know what to reconstruct)
- neighboring sounds/pixels
  (recover details)
- other information (e.g. speaker id)

Experiments on Speech



WaveNet decoder conditioned on:

- latents extracted at 24Hz-50Hz
- speaker

3 bottleneck evaluated:

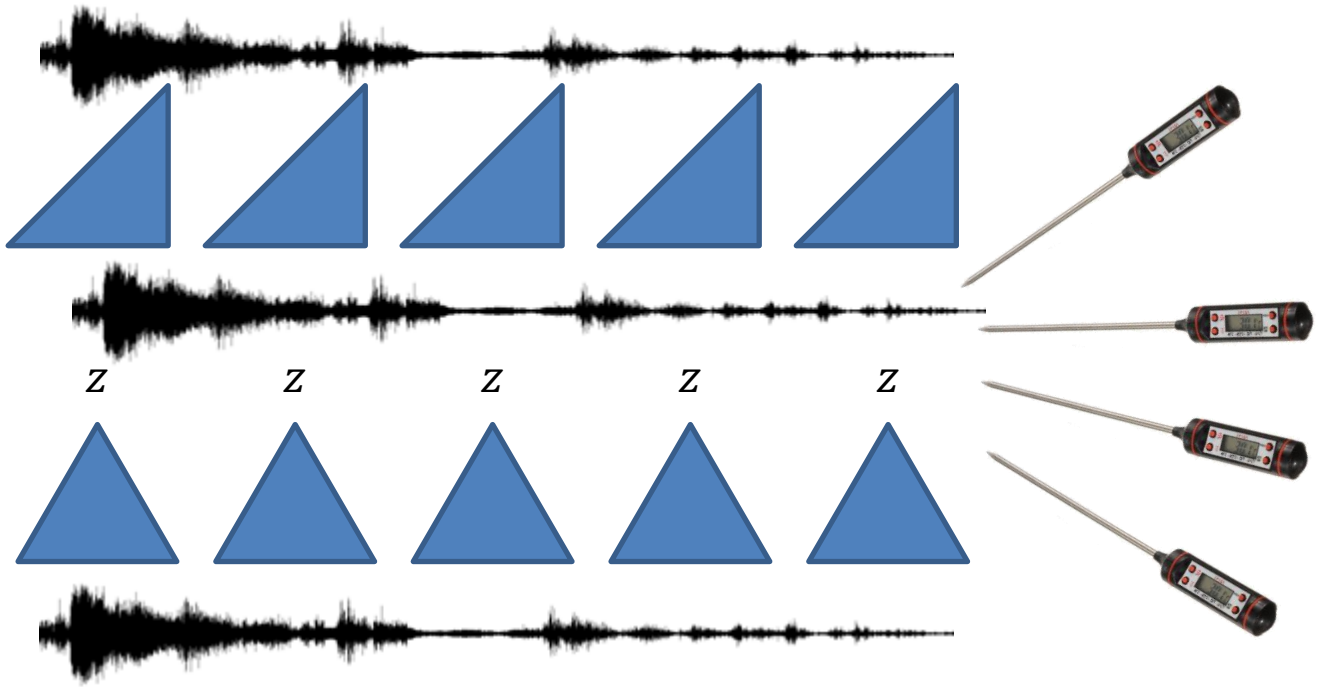
- Dimensionality reduction, max 32 bits/dim
- VAE, $KL\big(q(z|x) \parallel \mathcal{N}(0,1)\big)$ nats (bits)
- VQVAE with K protos: $\log_2 K$ bits

Input:

Waveforms, Mel Filterbanks, MFCCs

Methodology: probing points

We have inserted probing classifiers at 4 points in the network:



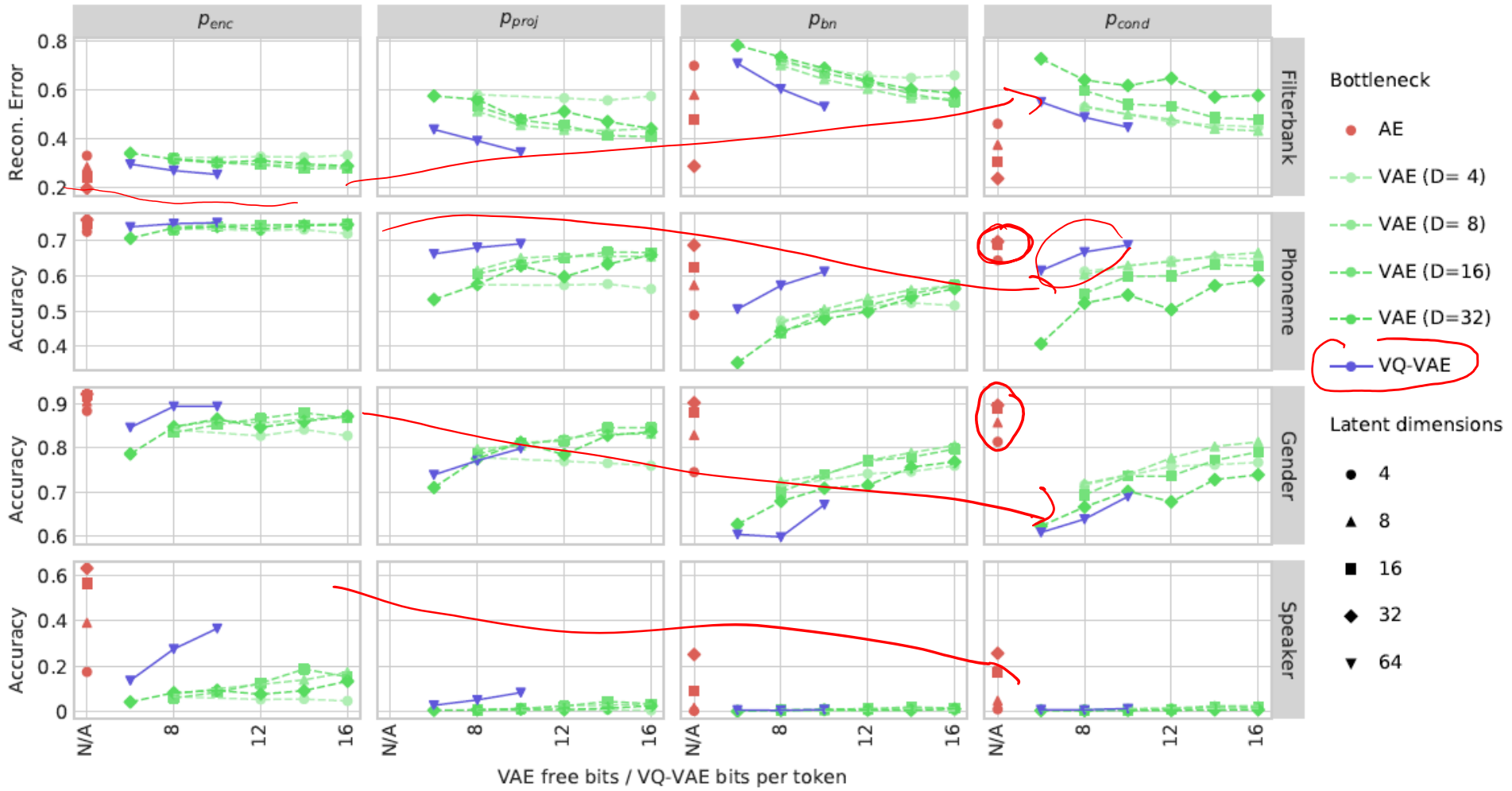
p_{cond}: several z codes mixed using a convolution.
The wavenet uses it for conditioning

p_{bn}: the latent codes

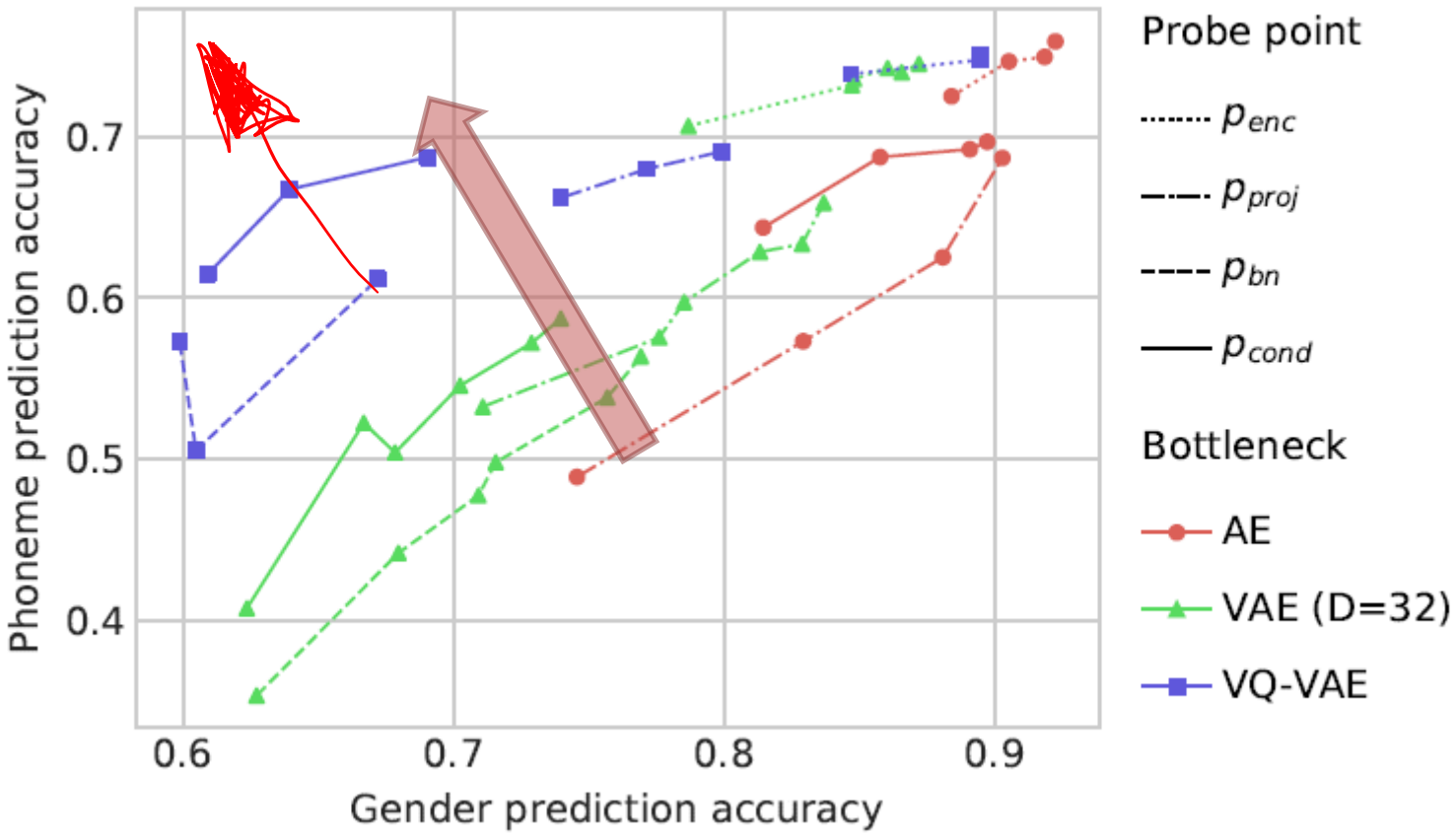
p_{proj}: low dimensional representation input to the bottleneck layer

p_{enc}: high dimensional representation coming out of the encoder

What information is captured in the latent codes?



Phonemes vs Gender tradeoff



Performance on ZeroSpeech 2017 unit discovery

	Within-speaker									Across-speaker								
	English (45h)			French (24h)			Mandarin (2.4h)			English (45h)			French (24h)			Mandarin (2.4h)		
Model	1s	10s	2m	1s	10s	2m	1s	10s	2m	1s	10s	2m	1s	10s	2m	1s	10s	2m
Unsupervised baseline	12.0	12.1	12.1	12.5	12.6	12.6	11.5	11.5	11.5	23.4	23.4	23.4	25.2	25.5	25.2	21.3	21.3	21.3
Supervised topline	6.5	5.3	5.1	8.0	6.8	6.8	9.5	4.2	4.0	8.6	6.9	6.7	10.6	9.1	8.9	12.0	5.7	5.1
VQ-VAE (per lang, p_{cond})	**5.6**	**5.5**	**5.5**	**7.3**	**7.5**	**7.5**	11.2	10.7	10.8	**8.1**	**8.0**	**8.0**	**11.0**	**10.8**	**11.1**	12.2	11.7	11.9
Heck et al. [57]	6.9	6.2	6.0	9.7	8.7	8.4	**8.8**	**7.9**	**7.8**	10.1	8.7	8.5	13.6	11.7	11.3	**8.8**	**7.4**	**7.3**
Chen et al. [58]	8.5	7.3	7.2	11.2	9.4	9.4	10.5	8.7	8.5	12.7	11.0	10.8	17.0	14.5	14.1	11.9	10.3	10.1
Ansari et al. [59]	7.7	6.8	N/A	10.4	N/A	8.8	10.4	9.3	9.1	13.2	12.0	N/A	17.2	N/A	15.4	13.0	12.2	12.3
Yuan et al. [60]	9.0	7.1	7.0	11.9	9.5	9.5	11.1	8.5	8.2	14.0	11.9	11.7	18.6	15.5	14.9	12.7	10.8	10.7

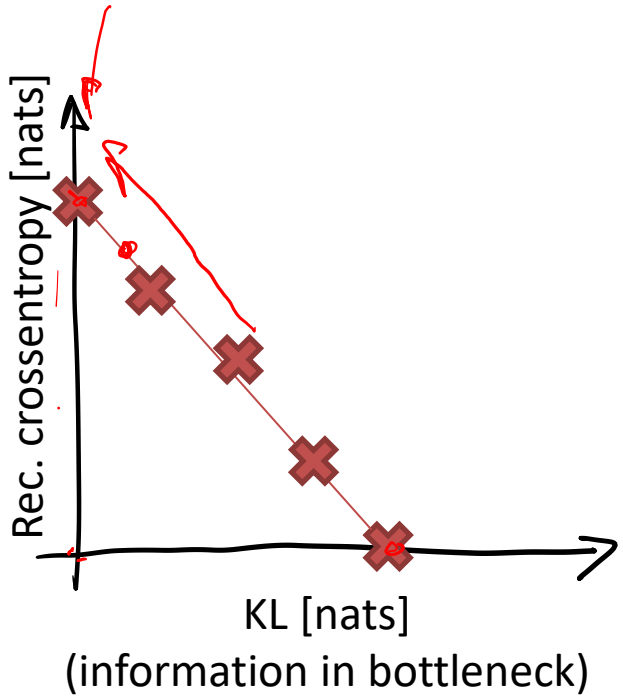
SOTA results in unsupervised phoneme discrimination Fr and EN ZeroSpeech challenge.

Mandarin shows limitation of the method:

- Too little training data (only 2.4h unsup. speech)
- Tonal information is discarded.

VAE + autoregressive models: the danger of collapsing latents

- Purely Autoregressive models:
  SOTA unconditional log-likelihoods
- Autoencoder:
  information passed through bottleneck
  increases conditional log-likelihood



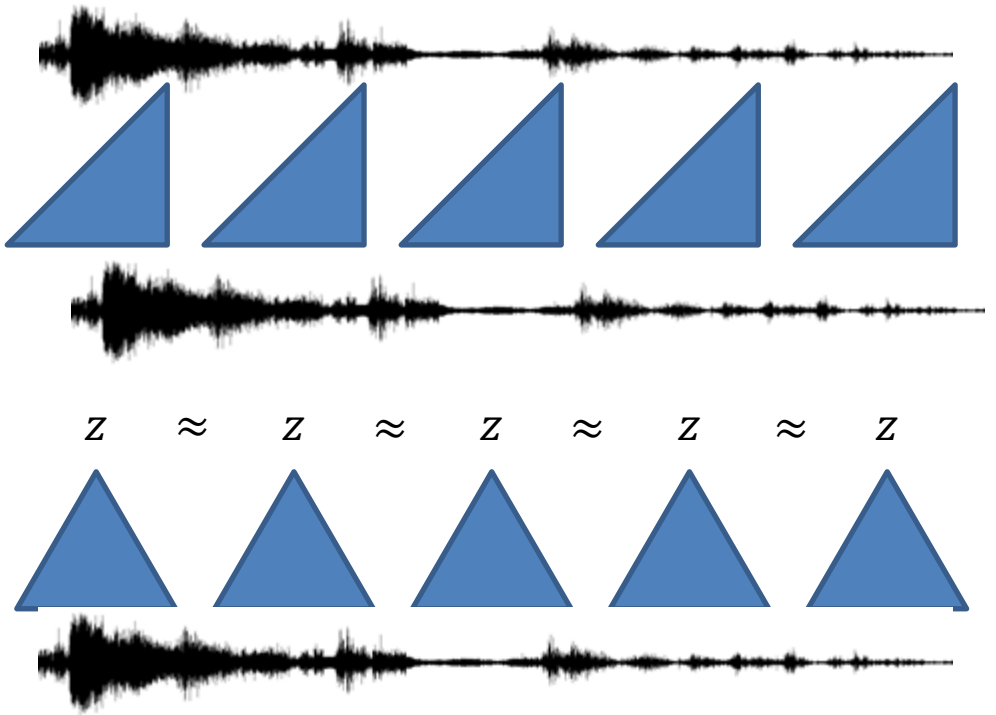
- Latent priors help us move along this line
- Too strong priors push the model to collapse
  - (ignored latents, unconditional behavior)
  - Hacky solutions: stop optimizing KL term (free bits), make it a hyperparameter (VQVAE)

Naïve latent representation smoothing

Apply a penalty on pairwise differences

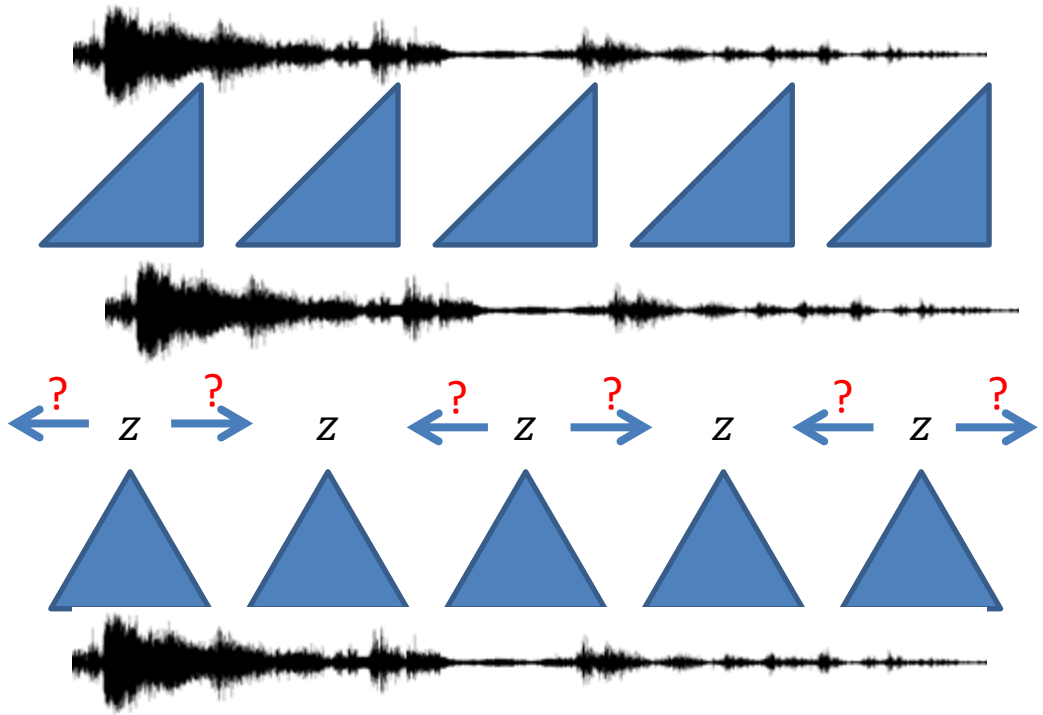
$$\sum_t (z_t - z_{t-1})^2$$

Trivially minimized when $z_1 = z_2 = z_3 = \cdots = z_T$

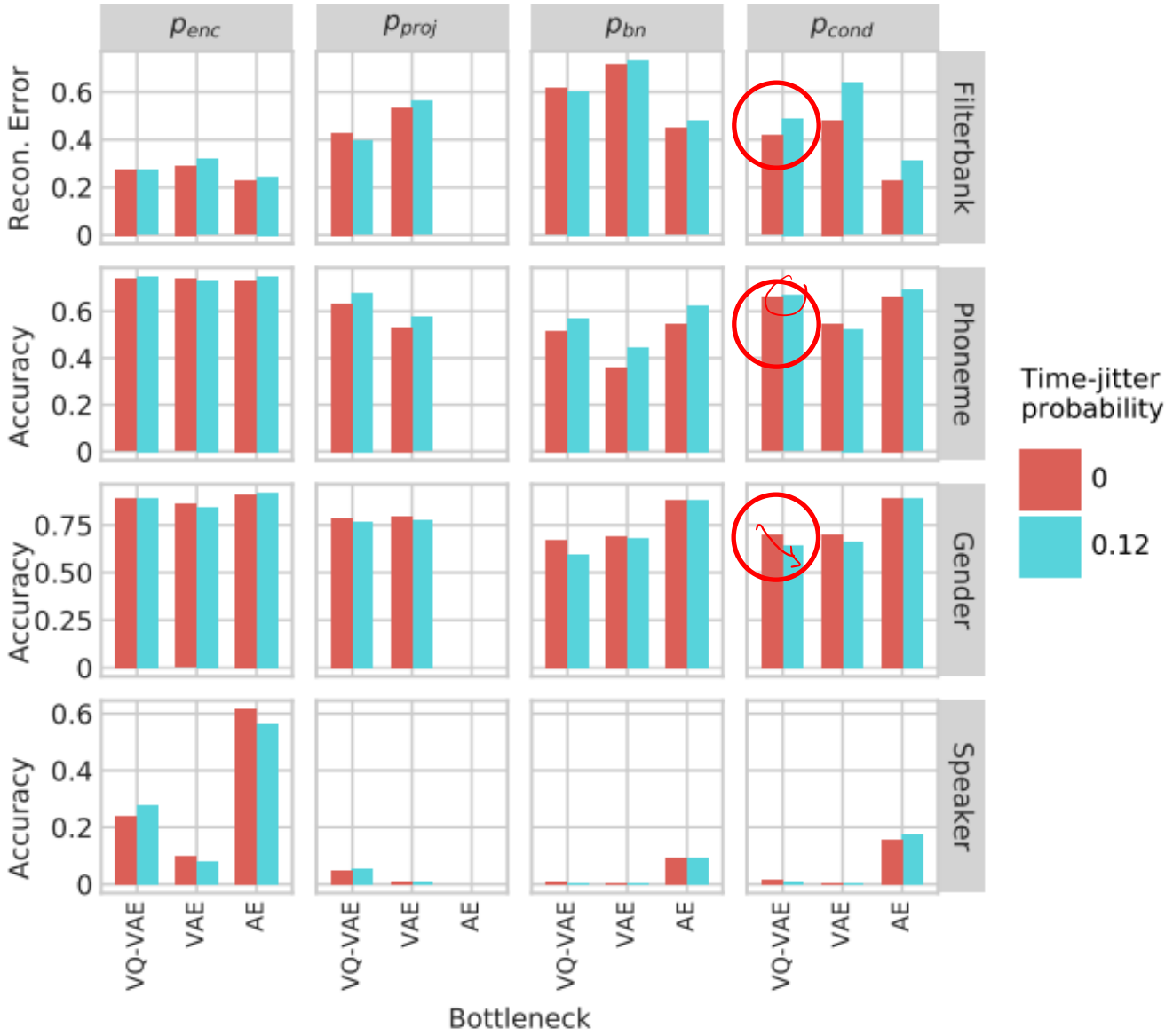


Randomized time jitter

Don’t penalize, **randomize**!

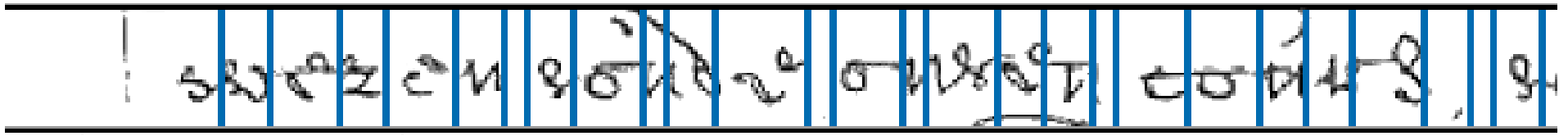


Randomized time jitter results



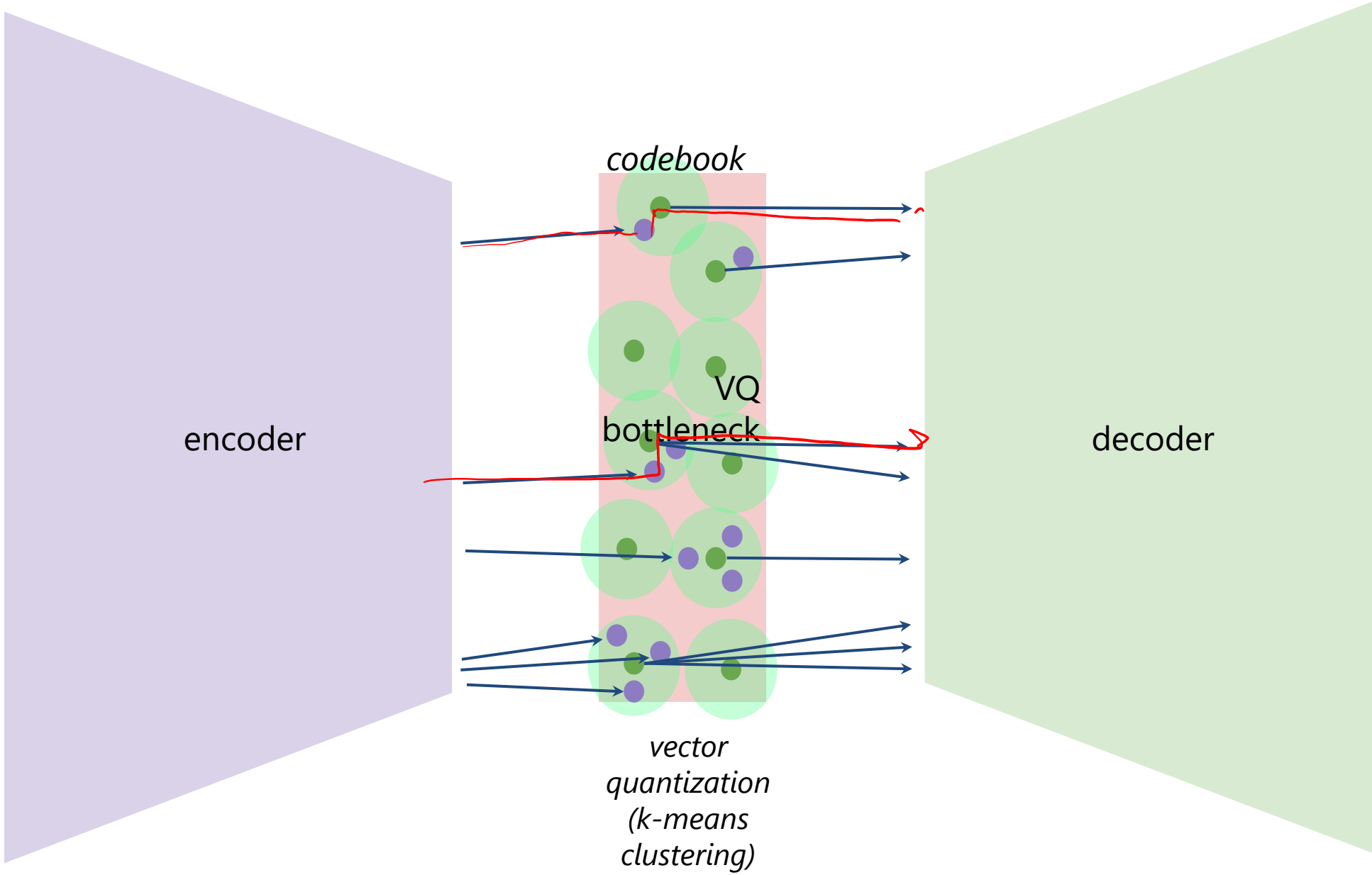
Learning to segment

We can delineate individual characters



Desired latent representation
piece-wise constant, with abrupt changes!

VQ-VAE


encoder
codebook
VQ
bottleneck
decoder
vector
quantization
(k-means
clustering)

Learning piecewise-constant representations

Quantization criterion:

$$\min_{q_1 \cdots q_T} \sum_{t=1}^{T} ||x_t - q_t|| \quad \text{subject to:} \sum_{t=1}^{T} [q_{t-1} \neq q_t] = K$$

encoder outputs

VQ tokens

expected # of characters

During training: enforce as a minibatch constraint

- easy to specify as a hyperparameter
- "large minibatch" statistics are close-ish to full data (our batches have many small chunks)
- fast with greedy merging algorithm (could be exact with Viterbi)

During inference: ?

Learning piecewise-constant representations

Quantization criterion:

$$\min \sum_{t=1}^{T} ||x_t - q_t|| \quad \text{subject to:} \sum_{t=1}^{T} [q_{t-1} \neq q_t] = K$$

Reformulate as:

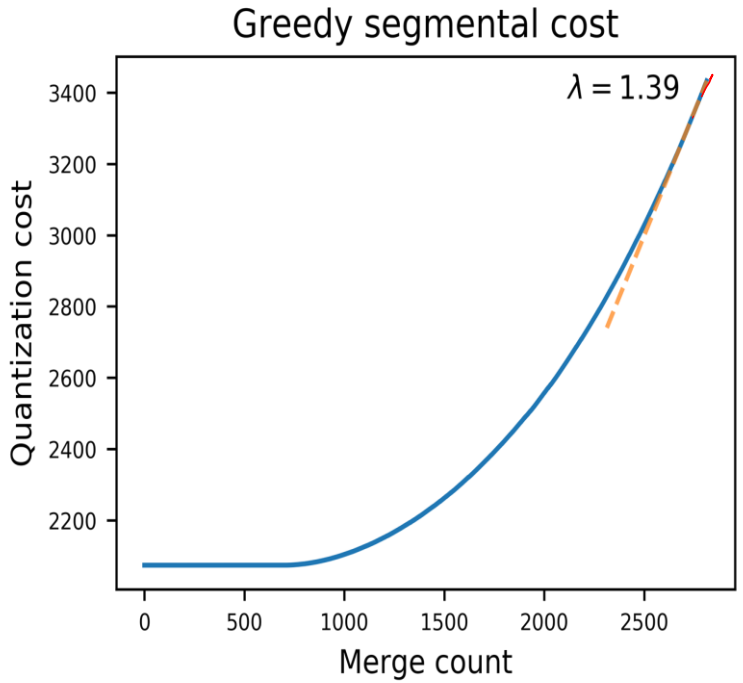
$$\min \sum_{t=1}^{T} ||x_t - q_t|| + \lambda \left(\sum_{t=1}^{T} [q_{t-1} \neq q_t] - K \right)$$

During training:

- estimate λ

During inference:

- use λ to balance the costs



Dual formulation and Markovian dynamics

Dual formulation resembles HMM cost:

$$\min \sum_{t=1}^{T} ||x_t - q_t|| + \lambda \left(\sum_{t=1}^{T} [q_{t-1} \neq q_t] - K \right)$$

state change cost

encoder outputs

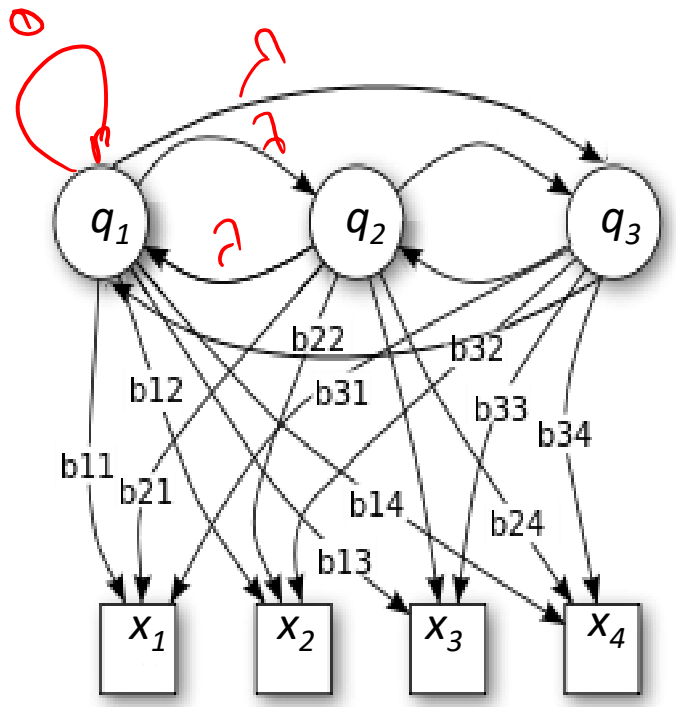


Image source: Wikipedia

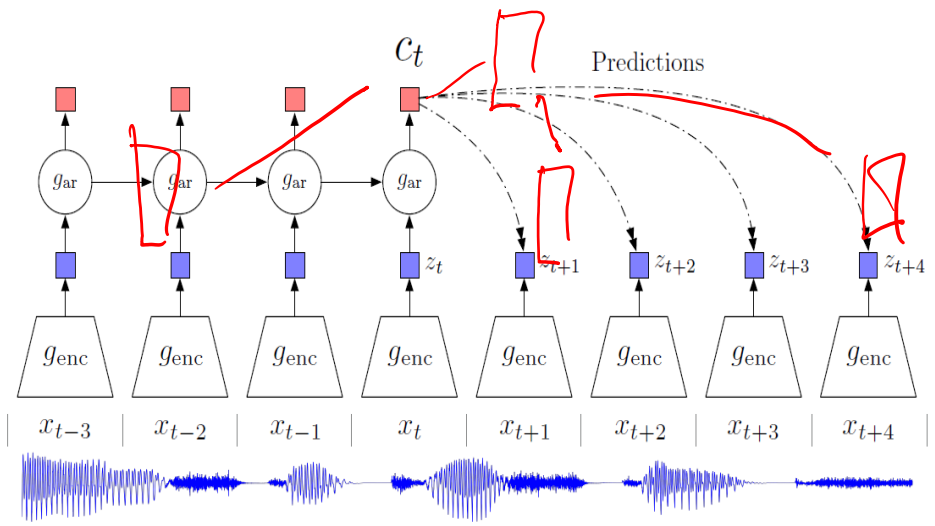
Observed states: encoder outputs

Hidden states: sequence of VQ tokens

Piecewise-constant results: Scribblelens

Dataset	*Model*	*Downstream CER*	*MLP Accuracy @Bottleneck*	*Token -> Char Accuracy*	*Rand score*	*Mutual information*
Single scribe Tasman	*Unsupervised base*	47%	54%	34%	0.15	0.15
	Piecewise-const VQ bottleneck	**30%**	**65%**	**57%**	**0.23**	**0.36**
All scribes	*Unsupervised base*	60%	42%	30%	0.13	0.10
	Piecewise-const VQ bottleneck	**51%**	**49%**	**39%**	**0.18**	**0.20**

Contrastive Coding



At step t make K independent predictions p_t^k for K next latent vectors.

Score them using negative sampling

$$s_t^k = \frac{e^{<p_t^k, z_{t+k}>}}{e^{<p_t^k, z_{t+k}>} + \sum_n e^{<p_t^k, z_n>}}$$

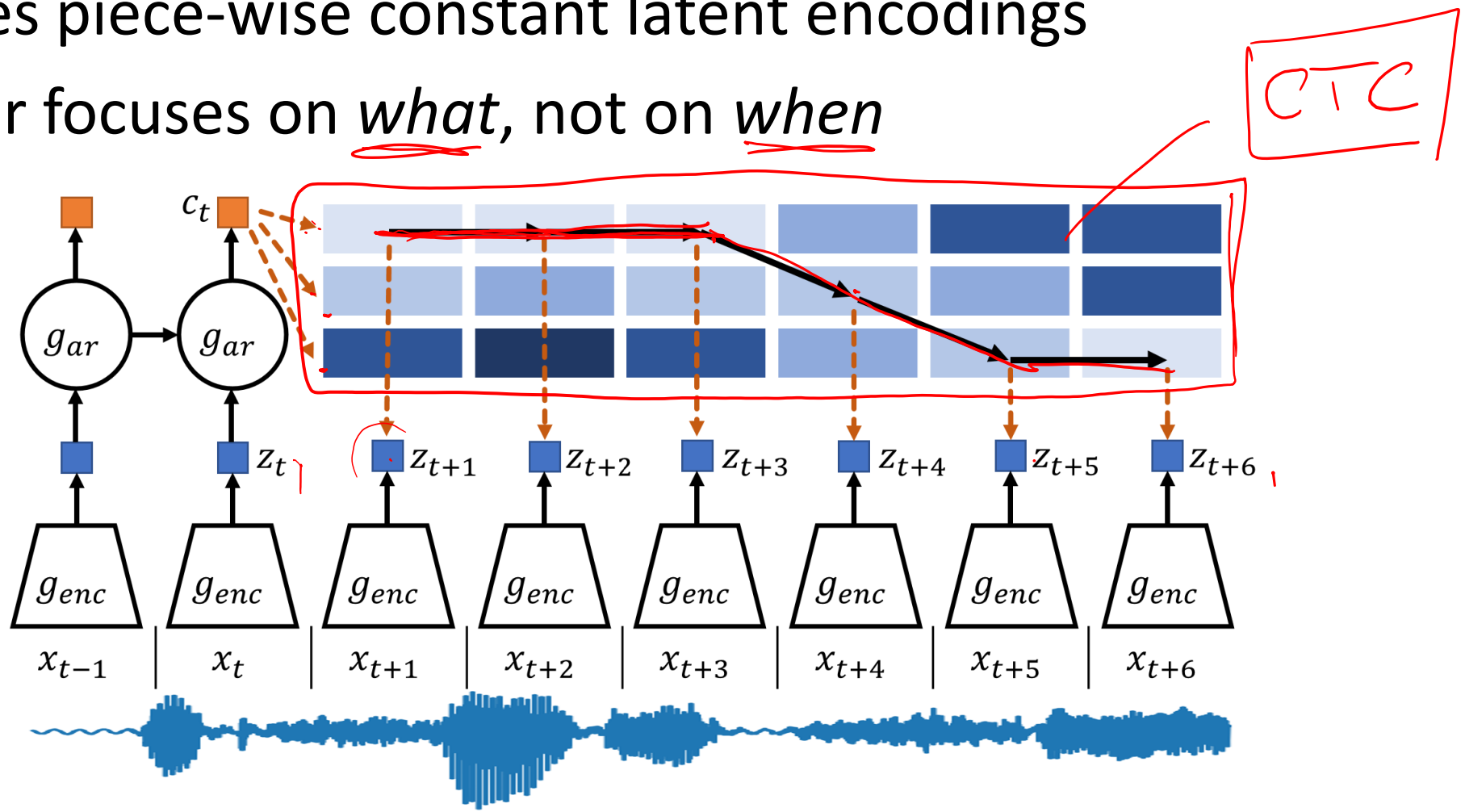
Where z_n are some randomly selected latent codes.

Oord et al. „Representation Learning with Contrastive Predictive Coding”

Aligned CPC (arxiv.org/abs/2104.11946)

Core idea: make few predictions, align to many frames:

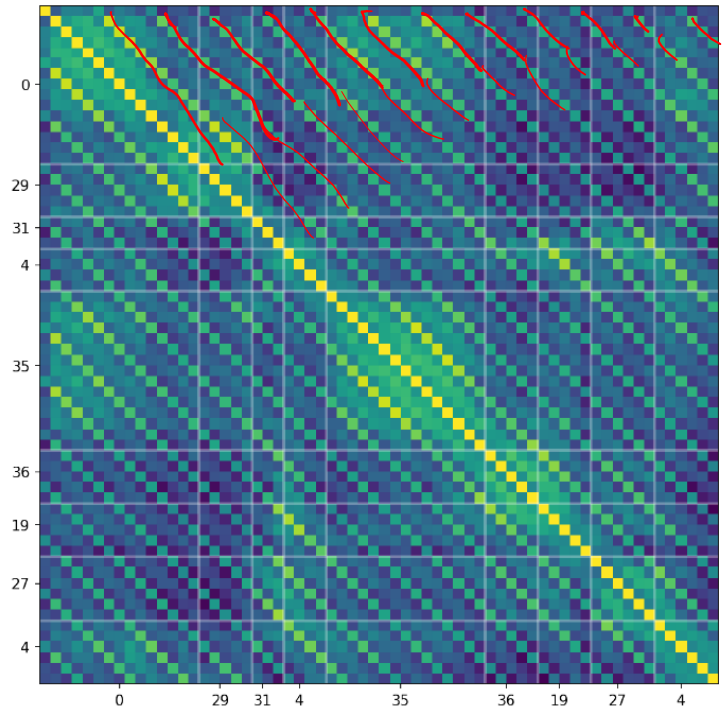
- Promotes piece-wise constant latent encodings
- Predictor focuses on *what*, not on *when*



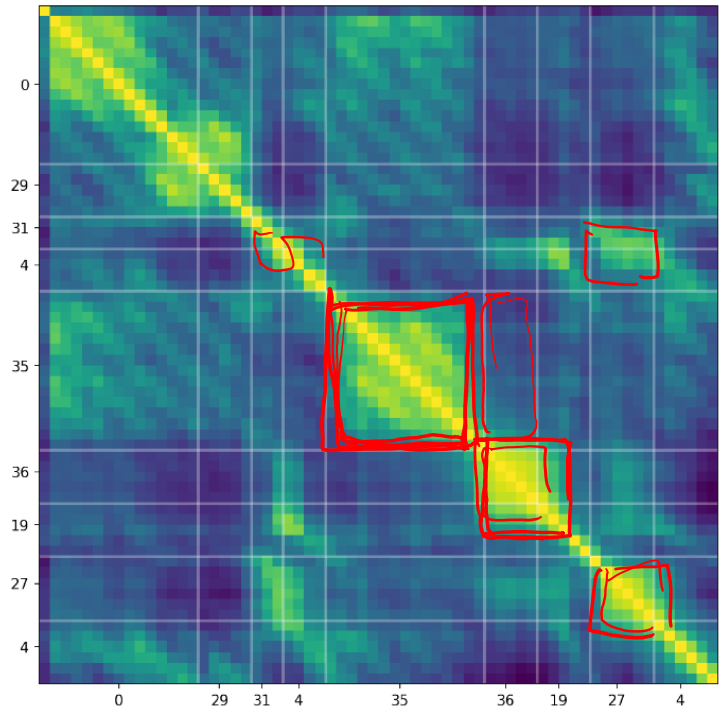
ACPC: more stable representations

Vanilla CPC exploits conv strides to learn relative position

Phoneme boundaries more clearly visible in ACPC



CPC



ACPC (K=8, M=12)

ACPC: better ABX and phone recognition scores

ABX Scores (lower is better)

	CPC (ZeroSpeech [10])		CPC (our baseline)		ACPC M=12 K=8	
Dev	Within	Across	Within	Across	Within	Across
clean	6.18%	8.02%	6.68%	8.39%	**5.37%**	**7.09%**
other	8.46%	13.59%	9.03%	13.87%	**7.46%**	**12.60%**

Framewise phoneme prediction (higher is better)

	Step	Encoded z_t		Contextual c_t	
Model	time	Train	Val	Train	Val
CPC	2.6ms	51.5%	51.2%	67.8%	67.5%
ACPC M=12 K=10	2.4ms	51.8%	51.3%	68.8%	68.4%
ACPC M=12 K=8	2.1ms	52.0%	51.9%	**69.7%**	68.6%
ACPC M=12 K=6	1.8ms	**52.2%**	**52.1%**	69.2%	**68.8%**
ACPC M=12 K=4	**1.5ms**	52.1%	51.9%	69.0%	68.7%
ACPC M=16 K=10	2.4ms	52.1%	51.9%	69.1%	**68.8%**
ACPC M=20 K=12	2.6ms	52.1%	51.9%	68.0%	67.8%

Summary

- We can design laten variables models that work well on sequential data
- We can enforce priors on the latent space, by
  - Applying hidden dynamics
    - during training, force a fixed number of code changes
    - during testing, use the equivalent HMM formulation
    - Constraint-penalty duality: very generic technique, can also use it to e.g. enforce unit unigram probabilities.
  - Aligning model predictions to latent encodings