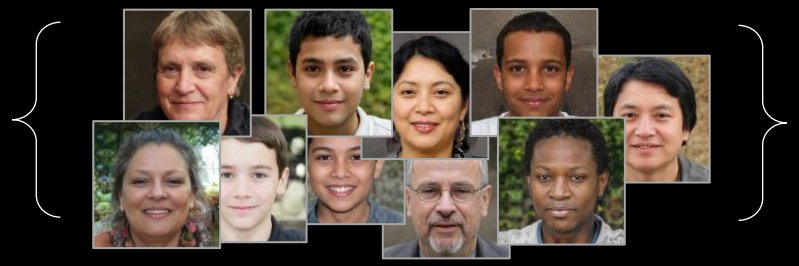
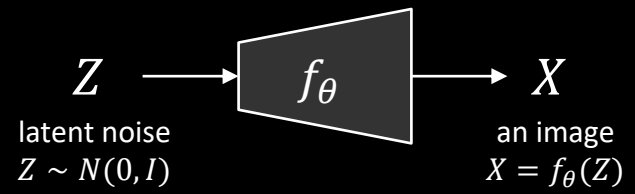
A generative modelling task



Training data: a dataset of photographs of real faces

Task: generate *new* images that look like them

Proposed model: a latent-variable generative model



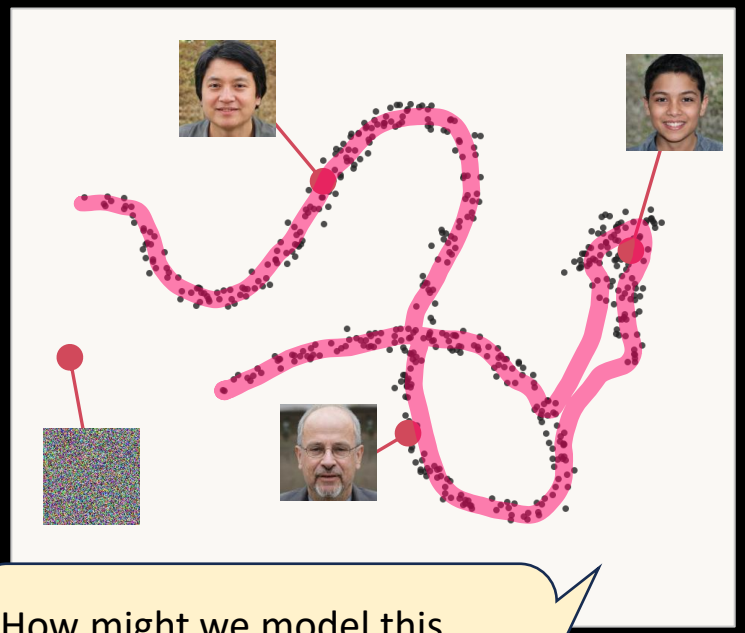
Fitting and evaluation: involve a challenging expectation

$$\Pr_{f_\theta(Z)}(x) = \int_z \mathbb{1}_{f_\theta(z)=x} \Pr_Z(z)\, dz = \mathbb{E}\, \mathbb{1}_{f_\theta(Z)=x}$$

Why is this expectation hard to approximate?

FACE IMAGES FROM THISPERSONNOTEXIST.ORG

Let's think of the latent variable not just as a "source of noise" but instead as a low-dimensional description of the data



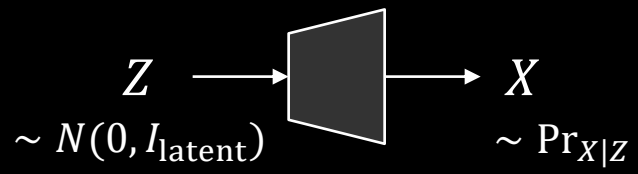
How might we model this point-cloud as samples from $X = f_\theta(Z)$? What Z, what f_θ?

A proposed model: Let $Z = (Z_1, Z_2)$ and $X = f_\theta(Z_1) + Z_2$

- $Z_1 \sim N(0, I_{\text{latent}})$ is low-dimensional
- f_θ embeds the low-dimensional description into data space
- Z_2 is noise, e.g. $Z_2 \sim N(0, \sigma^2 I_{\text{data}})$

The noise is what lets Z_1 be low-dimensional.
Without it, $X = f_\theta(Z_1)$ with low-dim Z_1 wouldn't describe the data.

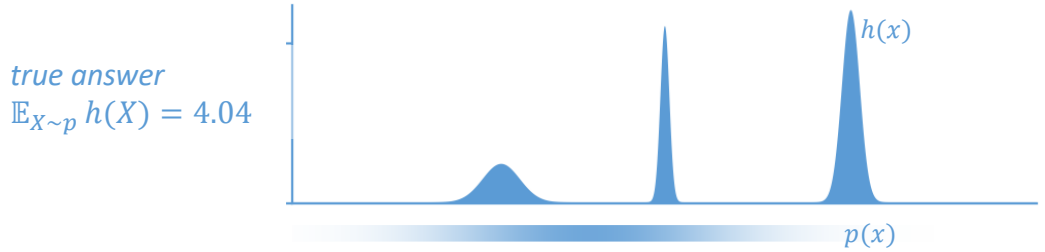
Simplified notation: fold f_θ and Z_2 into a "decoder" distribution $\Pr_{X|Z}$



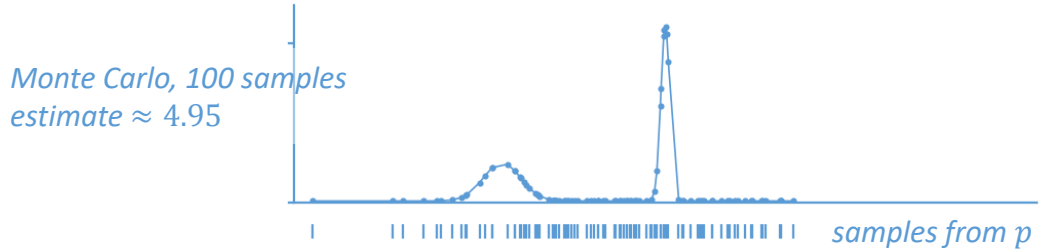
To compute the expectation of a spiky function, it's best to pick your sample points cleverly

(called Importance Sampling)

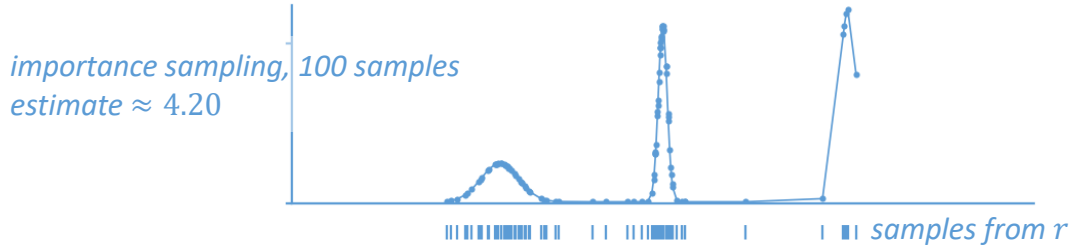
The problem. We want $\mathbb{E}_{X\sim p}\, h(X)$ for a spiky function h



Monte Carlo estimate. $\mathbb{E}_{X\sim p} h(X) \approx \frac{1}{m}\sum_j h\,(x_j)$, with $x_1, \ldots, x_m$ drawn from p



Importance sampling estimate. Drawing samples $x_1, \ldots, x_m$ from some sampling distribution r, $\mathbb{E}_{X\sim p} h(X) \approx \frac{1}{m}\sum_j h\,(x_j)\frac{p(x_j)}{r(x_j)}$



The correction factor comes from

$$\mathbb{E}_{X\sim p} h(X) = \int_x h\,(x)\,p(x)\,dx = \int_x h\,(x)\frac{p(x)}{r(x)}\,r(x)\,dx = \mathbb{E}_{X\sim r}\left[h(X)\frac{p(X)}{r(X)}\right]$$

Any sampling distribution r is allowed (as long as $r > 0$ whenever $p > 0$). What makes a good choice?

The perfect sampler is $r(x) \propto h(x)\,p(x)$, for $h > 0$ (as it is in our case). This has zero variance, so one sample is sufficient.

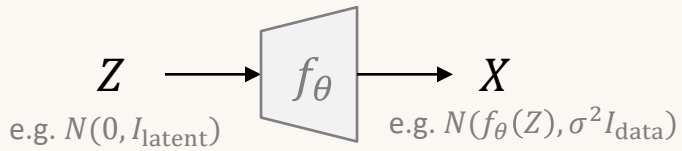
But sampling from this r is as hard as solving the expectation in the first place!

A good sampler puts its samples where $h\,p$ is large: $r(x) \propto\!\!\approx h(x)\,p(x)$.

Cunning idea 1: use a parametric sampler, called the *encoder*, to estimate $\mathbb{E}_Z \Pr_X(x \mid Z)$

$\mathbb{E}_{Z\sim p_Z} p_X^\theta(x|Z)$

Our model and its likelihood.



$$p_X^\theta(x) = \mathbb{E}_{Z\sim p_Z} p_X^\theta(x|Z)$$

$$\Pr_X(x) = \mathbb{E}_Z \Pr_X(x|Z)$$

Notation change: write p^θ for our generative model; use subscripts for all its variables

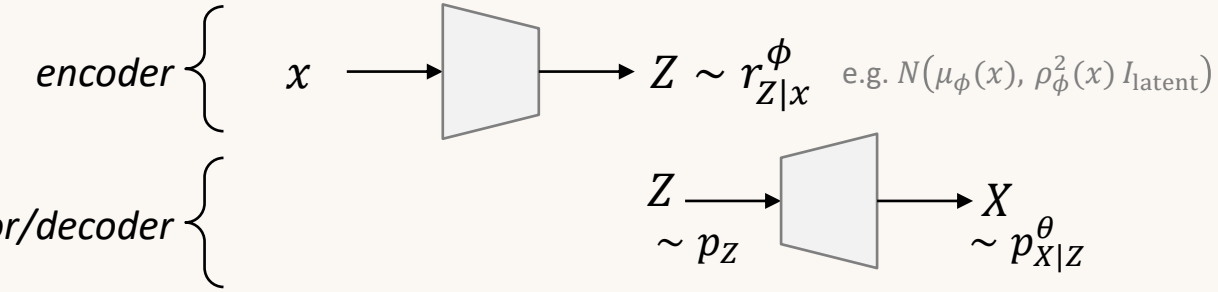
Estimate the likelihood with importance sampling. Using a sampler r

$$\mathbb{E}_{Z\sim p_Z} p_X^\theta(x|Z) = \mathbb{E}_{Z\sim r}\left[p_X^\theta(x|Z)\,\frac{p_Z(Z)}{r(Z)}\right]$$

What's the perfect sampler?

What do you recognize the perfect sampler as?

Cunning idea. The best sampler depends on x. But don't design the sampler ourselves: instead, give a network the information it needs (i.e. x) and ask it for the Z-distribution.



Estimating KL using importance sampling

Our usual goal is to minimize the KL,

$$\mathrm{KL}(q_X \parallel p_X^\theta) = \underbrace{\text{const}}_{\mathbb{E}_{X\sim q_X}\log q_X(X)} - \mathbb{E}_{X\sim q_X}\log p_X^\theta(X) = \text{const} - \mathbb{E}_{X\sim q_X}\left[\log \underbrace{\left(\mathbb{E}_{Z\sim r_{Z|X}^\phi} p_X^\theta(X|Z)\,\frac{p_Z(Z)}{r_Z^\phi(Z|X)}\right)}_{w(X,Z)}\right]$$

Obvious estimation method:

- (1) sample x then (2) estimate $\log \mathbb{E}_Z w(x,Z)$ using many samples $z_1, \ldots, z_m$

Quick and dirty estimation:

- It's a waste getting a perfect estimate for each x, since each x is itself just a single draw
- Let's simply use a single Z-sample for each x: we draw (x, z_1) then compute $\log w(x, z_1)$
- Formally, this estimates $\mathbb{E}_{X\sim q_X, Z\sim r_{Z|X}^\phi} \log w(X,Z)$ which isn't our goal — but is directly related, via Jensen's inequality:

$$\underbrace{\mathbb{E}_{X\sim q_X}\log \mathbb{E}_{Z\sim r_{Z|X}^\phi} w(X,Z)}_{\text{what we want}} \underset{\text{Jensen}}{\geq} \underbrace{\mathbb{E}_{X\sim q_X, Z\sim r_{Z|X}^\phi} \log w(X,Z)}_{\text{what one sample estimates}}$$

Thus

$$\mathrm{KL}(q_X \parallel p_X^\theta) \leq \text{const} - \mathbb{E}_{X\sim q_X, Z\sim r_{Z|X}^\phi} \log w(X,Z) = \text{const} - \left\{\underbrace{\mathbb{E}_{X\sim q_X, Z\sim r_{Z|X}^\phi} \log p_X^\theta(X|Z) - \mathbb{E}_{X\sim q_X}\mathrm{KL}(r_{Z|X}^\phi \parallel p_Z)}_{\text{call this } \mathrm{ELBO}(p^\theta, r^\phi)}\right\}$$

Training. The KL bound holds for every p^θ and every r^ϕ. To minimize KL, let's try to minimize the bound: $\max_{\theta,\phi} \mathrm{ELBO}(p^\theta, r^\phi)$

Cunning idea 2: maximizing the ELBO bound, rather than the exact log likelihood, is what trains the encoder

$$\mathrm{KL}(q_X \parallel p_X^\theta) = \text{const} - \underbrace{\mathbb{E}_{X\sim q_X} \log \mathbb{E}_{Z\sim r_{Z|X}^\phi} w(X,Z)}_{\text{This is insensitive to } \phi} \leq \text{const} - \underbrace{\mathrm{ELBO}(p^\theta, r^\phi)}_{\text{This depends on } \phi}$$

This is insensitive to ϕ (that's what importance sampling guarantees).
If we maximize it over θ, ϕ then there's no pressure to actually learn a good ϕ.

This depends on ϕ (since the error in Jensen's inequality depends on the variance of the importance sampler estimator).
If we maximize it over θ, ϕ we'll learn a good autoencoder.

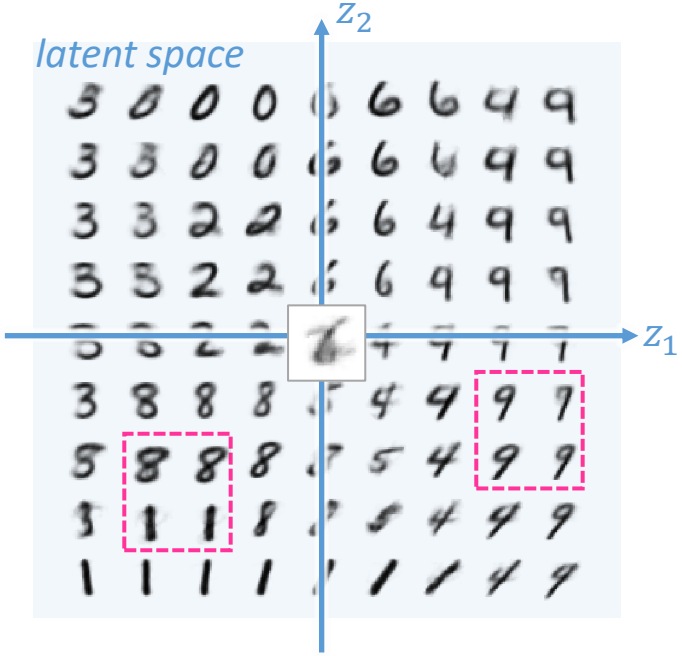
$$\mathrm{ELBO}(p^\theta, r^\phi) = \text{const} - \mathrm{KL}(q_X \parallel p_X^\theta) - \underbrace{\mathbb{E}_{X\sim q_X} \mathrm{KL}(r_{Z|X}^\phi \parallel p_{Z|X}^\theta)}_{\text{error in Jensen's inequality: see handout 4.}}$$

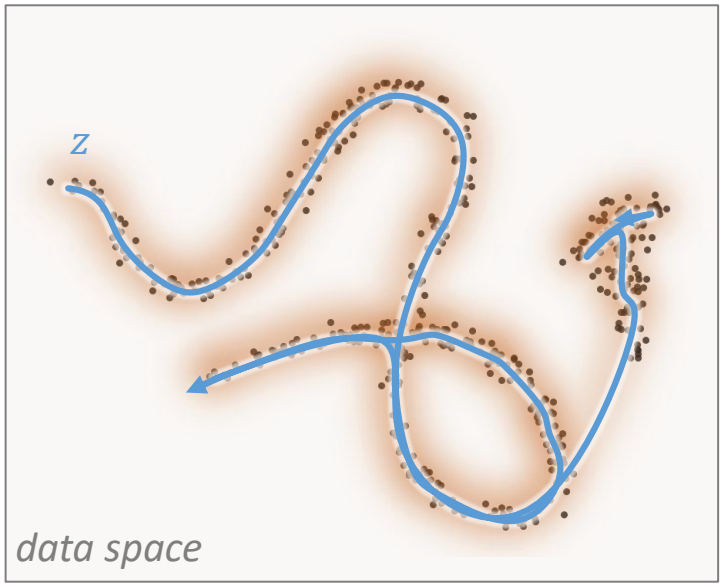
At a fixed p^θ, what encoder r maximizes the ELBO?

The word "variational" comes from this: see handout 3.

Cunning observation. By maximizing the ELBO, we're not only improving the generator to seek $p_X^\theta \approx q_X$, we're also improving the encoder to seek $r_{Z|X}^\phi \approx p_{Z|X}^\theta$.

Interpreting the axes of the latent variable





What do the axes of Z mean?

- *The dream:* we can magically discover human-meaningful latent dimensions
- *The reality:* the axes are arbitrary

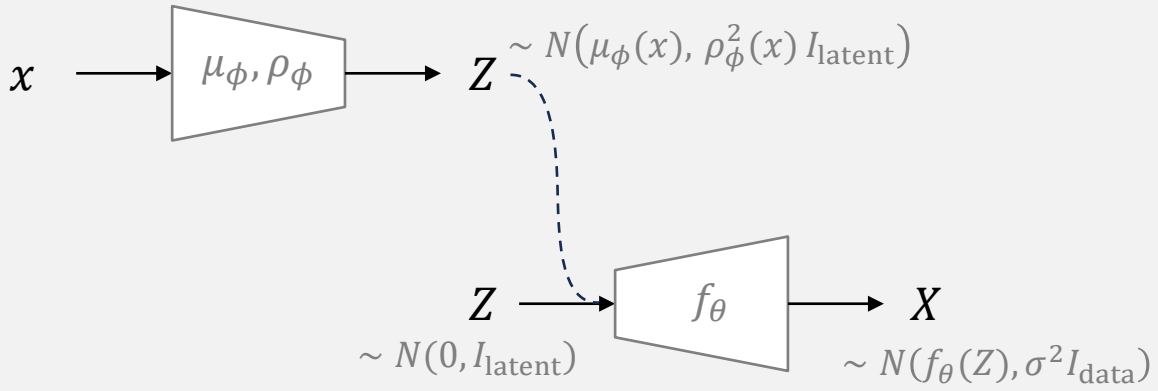
People have tried to learn "disentangled" axes. This is futile (unless we build specific identification assumptions into the model).

The β-VAE disentangler: Higgins et al. 2017. Futility: Locatello et al. 2019

How come the encoder discovers smoothly-varying axes?

- It's possible to autoencoder the data with discontinuous latent-dimension axes
- But this requires lots of capacity (discontinuous generators, multimodal encoders).
  A weak encoder (such as a Gaussian) will force the generator to be continuous.

Interpreting the terms in the ELBO, for a Gaussian autoencoder



$$\text{ELBO} = \underbrace{\mathbb{E}_{X\sim q_X,\, Z\sim r^\phi_{Z|X}} \log p^\theta_X(X \mid Z)}_{} - \mathbb{E}_{X\sim q_X} \underbrace{\mathrm{KL}(r^\phi_{Z|X} \parallel p_Z)}_{}$$

$$= \mathbb{E}_{Z\sim r^\phi_{Z|x}} \left[-\frac{\| x - f_\theta(Z) \|^2}{2\sigma^2} + \text{const} \right]$$

For each sampled x, we'll estimate this with a single z sample.

This term measures reconstruction error.

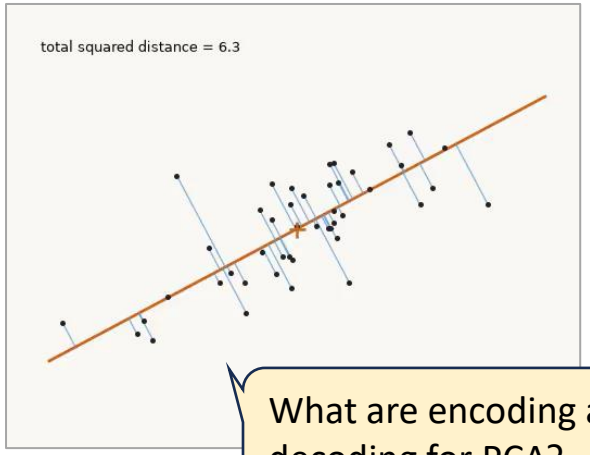
Training will try to minimize reconstruction error.

For each sampled x, this is the KL between two Gaussians, and there's an exact formula.

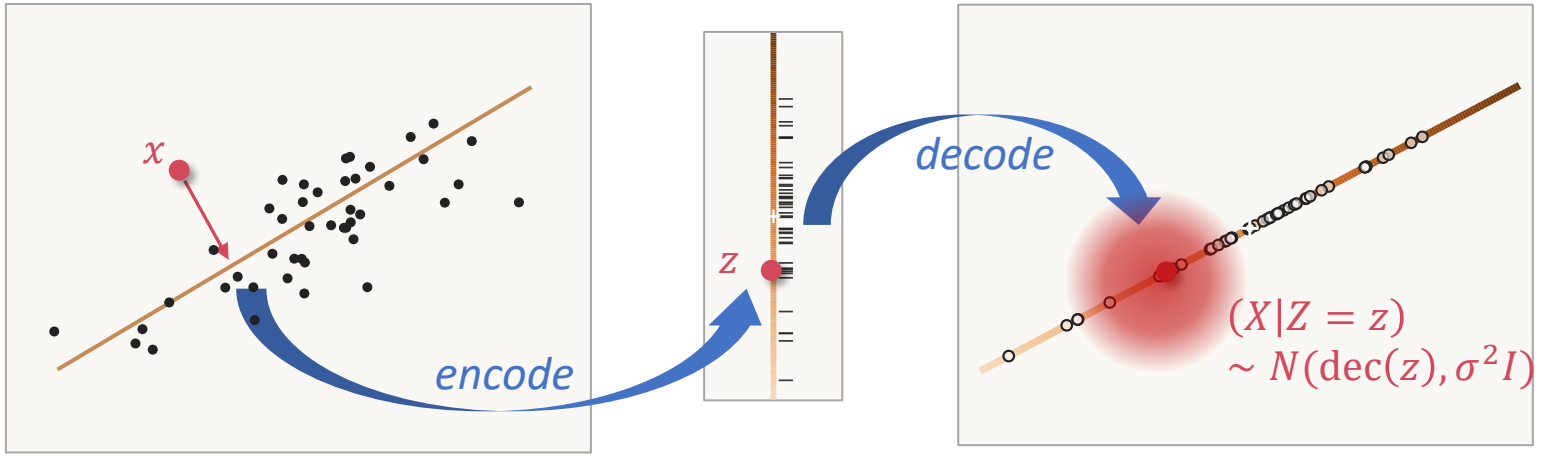
Training will try to keep $r^\phi_{Z|X}$ not too far from p_Z.

PCA is secretly an autoencoder

PCA is a dimension-reduction algorithm. It finds a low-dimensional subspace that's close to the dataset.



What are encoding and decoding for PCA?



After the best-fit line has been found,

- **encode a new point x:** map it to a distance along the latentspace line
- **decode a latent value z:** see the latentspace line as embedded in dataspace

Can we interpret these as an autoencoder? If so, what's p_Z?

A possible generative model.

- For Z, encode the training dataset to get $z_1, \ldots, z_n$ and put probability mass $1/n$ on each
- Decode $(X|Z=z)$ with a Gaussian blob $N(\mathrm{dec}(z), \sigma^2 I)$

This PCA-autoencoder "incorporates" the training data and is O(n) at inference time, whereas in VAE the encoder and decoder are O(1)

The autoencoder fit. We can fit this generative model (over p_Z, the straight line, and σ). Fitting it via ELBO corresponds exactly to PCA.

This gives us rigorous tools (validation, holdout set evaluation) to choose how many latent dimensions to use.

Certificates of fit

The GAN critic and the VAE encoder together let us bound how well our model fits:

$$L(p^\theta, \lambda) \leq \mathrm{KL}(q_X \parallel p_X^\theta) \leq \mathrm{const} - \mathrm{ELBO}(p^\theta, r)$$

$$\mathrm{const} - L(p^\theta, \lambda) \geq \mathbb{E}_{X \sim q_X} \log p_X^\theta(X) \geq \mathrm{ELBO}(p^\theta, r)$$

GAN-type bound, for any critic λ

VAE bound, for any encoder r

NOTE: These bounds are written as 'true' expectations. To estimate them from data unbiasedly, use a holdout set.

- The 'model seller' should provide an encoder, to certify *my model is at least this good*
- The sceptical 'model buyer' shares a critic's report that states *your model is no better than this*
- Bad encoders / critics give weak bounds

The bounds look for different things.
The critic bound is bad at detecting mode collapse (i.e. when the model misses part of the truth).
When this happens, $\mathrm{KL}(q_X \parallel p_X^\theta) = \infty$, and the ELBO upper bound spots this issue.