

**10417-617**  
**Deep Learning: Fall 2020**

Andrej Risteski

Machine Learning Department

**Lecture 16:**  
Variational autoencoders,  
evaluating representations

# Recap: the simplest of representation learners

**Sparse coding:** learn features, s.t. each input can be written as a *sparse linear combination* of some of these features.

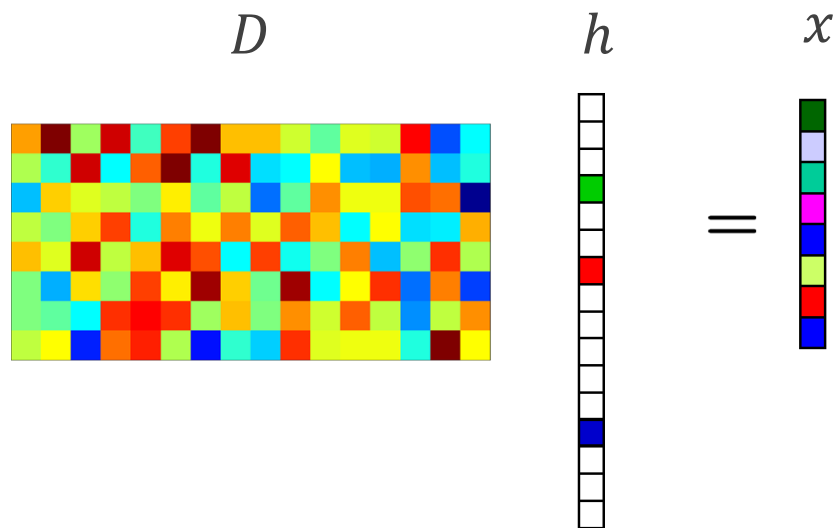
Originally made famous by *Olshausen and Field*, '96 as a model for how early visual processing works (edge detection etc.)

**Autoencoders:** learn encoding with some constraints (e.g. functional form, sparsity, denoising ability) from which the inputs can be approximately reconstructed.

# Sparse coding

**Goal:** learn a *dictionary*  $D$  of features, s.t. each sample  $x$  is (approximately) writeable as a *sparse* (i.e. mostly zeros) linear combination of these features.

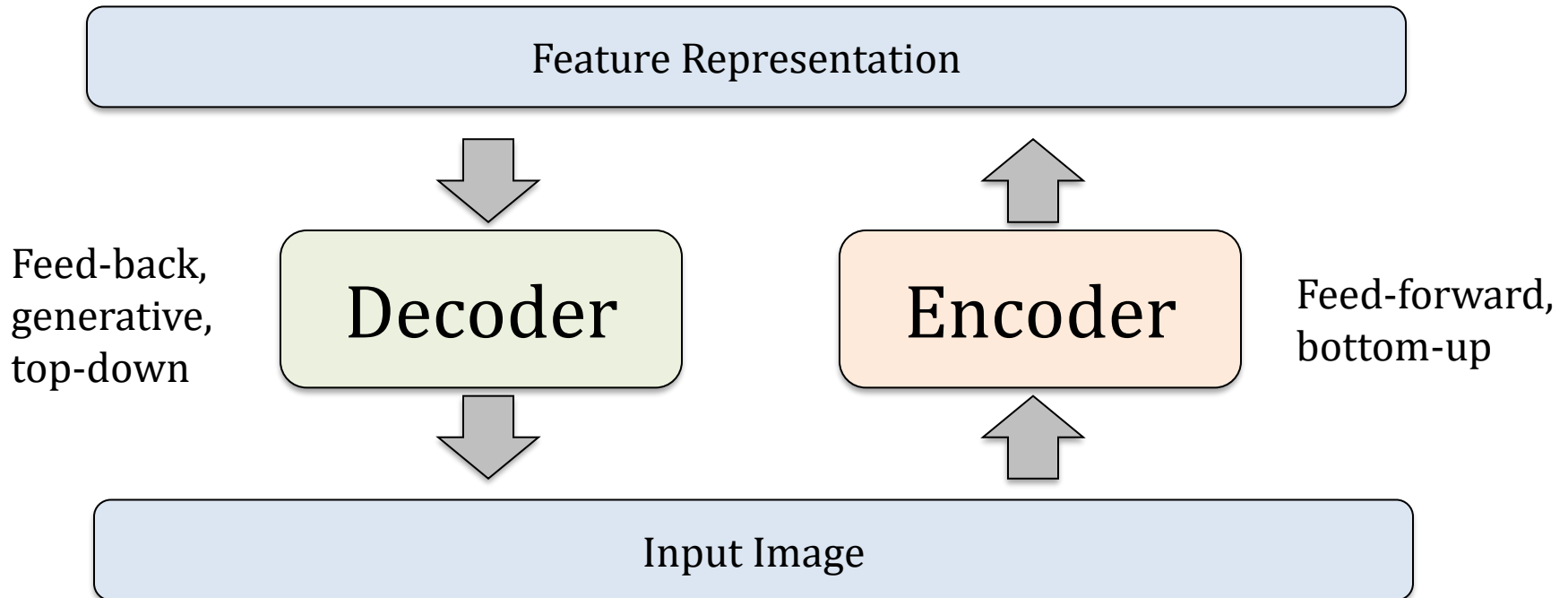
$$\forall x: \quad x \approx Dh, \quad ||h||_0 \text{ small}$$



$h$  is the representation of sample  $x$

# Autoencoders

The idea behind autoencoders: learn features, s.t. input is reconstructable from them

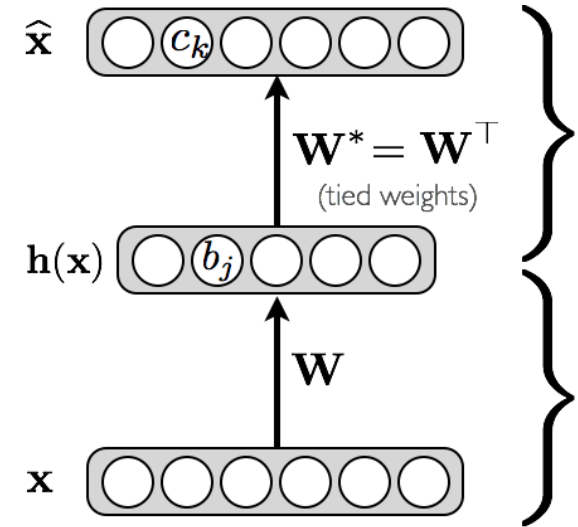


- Details of what goes inside the encoder and decoder matter!
  - Need *constraints* to **avoid learning an identity**.

# Autoencoders

Some way to prevent identity:

- *Weight tying* of encoder/decoder. (Often magical!)
- *Smaller dimension* for latent variables
- Enforce *sparsity* of the latent representation
- Encourage decoder to be robust to adding noise to  $\mathbf{x}$ . (*Denoising autoencoder*)
- **Encode to distribution rather than pointmass.** (*Variational autoencoder*)



# Typical losses

**Loss function** for inputs between 0 and 1

$$l(f(\mathbf{x})) = - \sum_k (x_k \log(\hat{x}_k) + (1 - x_k) \log(1 - \hat{x}_k))$$

⊗ *Cross-entropy error* ( $f(\mathbf{x}) \equiv \hat{\mathbf{x}}$ )

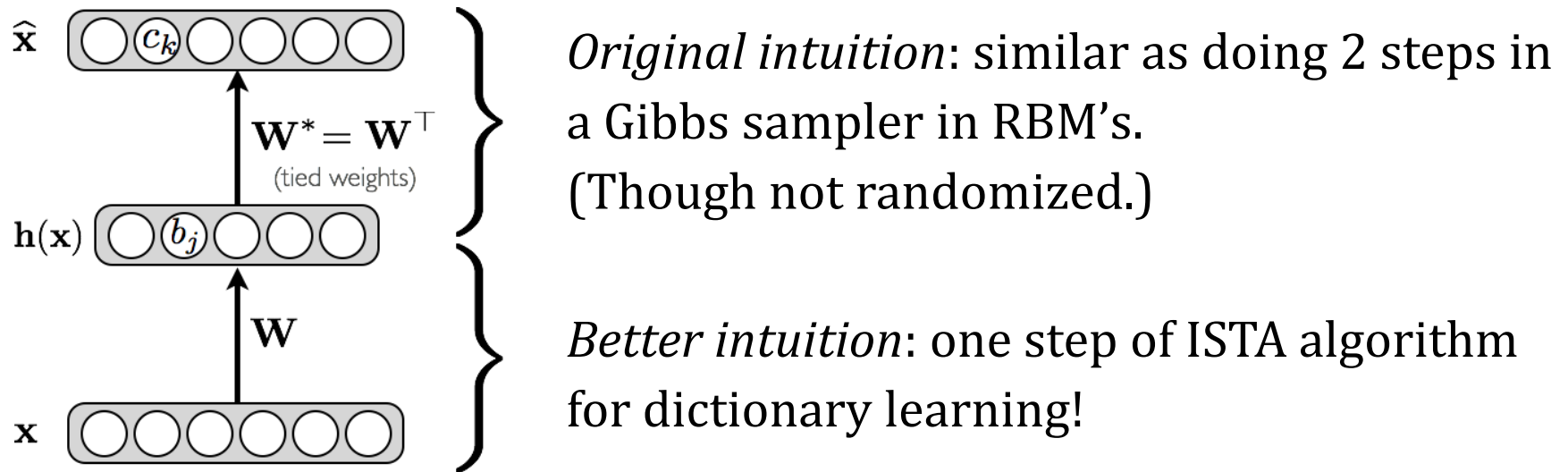
**Loss function** for real-valued inputs

$$l(f(\mathbf{x})) = \frac{1}{2} \sum_k (\hat{x}_k - x_k)^2$$

⊗  $l_2$  error

⊗ we use a linear activation function at the output

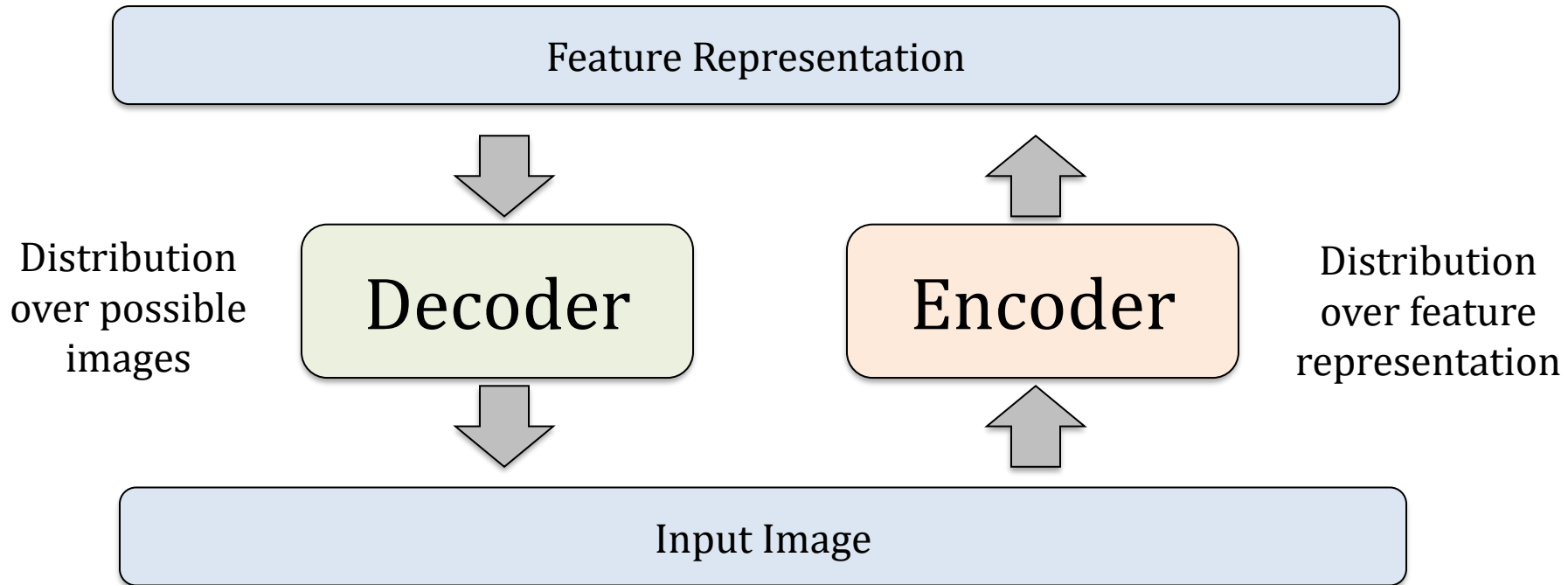
# Intuitions for weight tying



# Variational autoencoders

**The idea:** the encoder can output a *distribution*, rather than a *point mass*.

*We will derive this via a variational approach.*



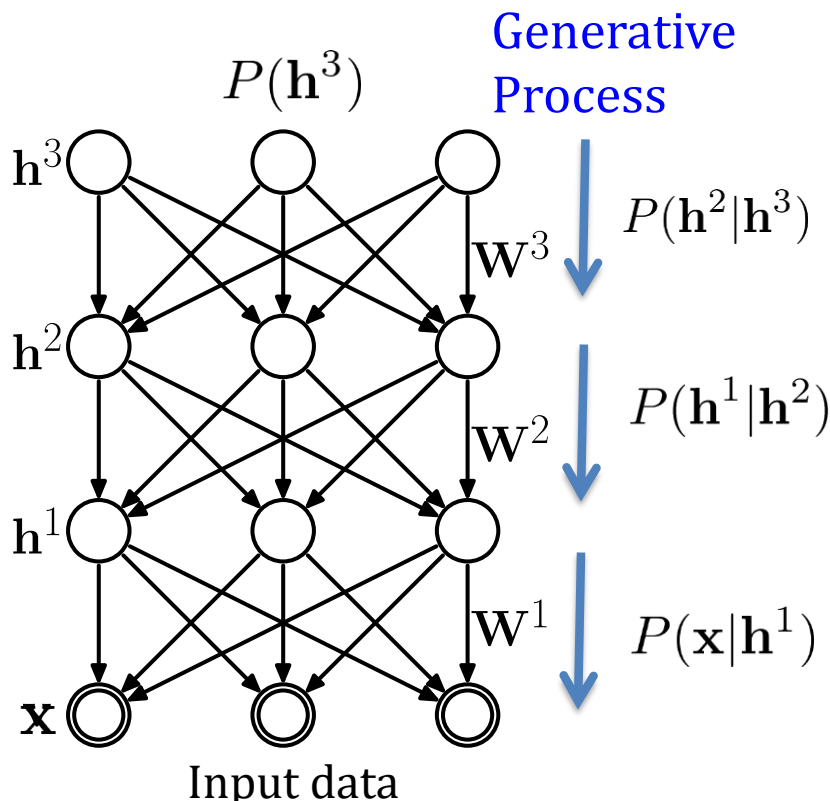


# Variational autoencoders

*“Decoder/generator”*: directed Bayesian network with Gaussian layers

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{\mathbf{h}^1, \dots, \mathbf{h}^L} p(\mathbf{h}^L|\boldsymbol{\theta}) p(\mathbf{h}^{L-1}|\mathbf{h}^L, \boldsymbol{\theta}) \cdots p(\mathbf{x}|\mathbf{h}^1, \boldsymbol{\theta})$$

Each term may denote a complicated nonlinear relationship



Typically, directed layers are parametrized as:

$$p(\mathbf{h}^{L-1}|\mathbf{h}^L, \boldsymbol{\theta}) = \mathcal{N}(\mu_{\boldsymbol{\theta}}(\mathbf{h}^L), \Sigma_{\boldsymbol{\theta}}(\mathbf{h}^L))$$

Gaussians, means/covariances functions (e.g. one-layer neural net) of previous layer and model parameters  $\boldsymbol{\theta}$ .

*Easy to sample!*

# Where does an “encoder” come in?

“*Decoder/generator*”: directed Bayesian network with Gaussian layers

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{\mathbf{h}^1, \dots, \mathbf{h}^L} p(\mathbf{h}^L|\boldsymbol{\theta})p(\mathbf{h}^{L-1}|\mathbf{h}^L, \boldsymbol{\theta}) \cdots p(\mathbf{x}|\mathbf{h}^1, \boldsymbol{\theta})$$

Recall *learning via variational inference*:

**ELBO:**  $\log p(x) = \max_{q(h^L|x)} H(q(h^L|x)) + \mathbb{E}_{q(h^L|x)}[\log p(x, h^L)]$

Max-likelihood can be written as:

$$\max_{\boldsymbol{\theta} \in \Theta} \max_{\{q(h^L|x)\}} \sum_{i=1}^n H(q(h^L|x)) + \mathbb{E}_{q(h^L|x)}[\log p(x, h^L)]$$

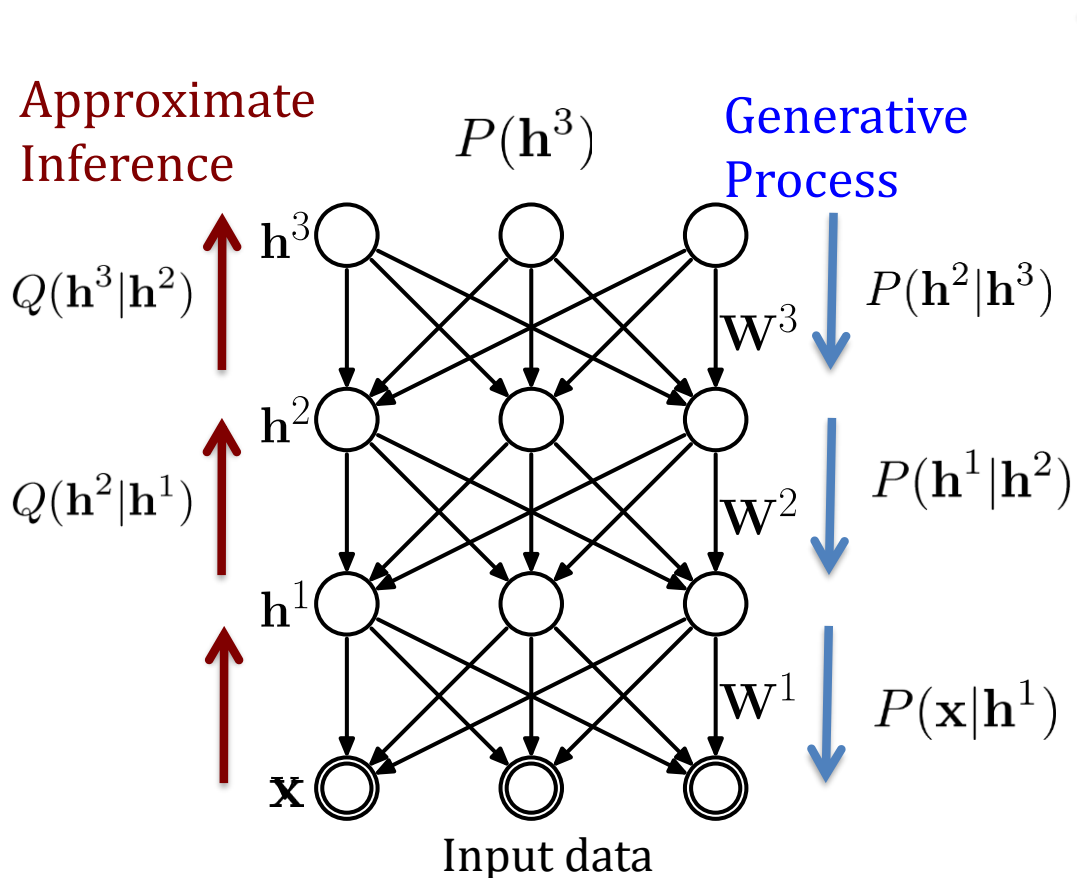
“*Encoder*”: a directed Bayesian network approximating  $q(h^L|x)$

It will be a directed Bayesian network in the “reverse” direction.

# Encoder: A “recognition network”

The encoder is defined in terms of an analogous factorization:

$$q(\mathbf{h}|\mathbf{x}, \boldsymbol{\theta}) = q(\mathbf{h}^1|\mathbf{x}, \boldsymbol{\theta})q(\mathbf{h}^2|\mathbf{h}^1, \boldsymbol{\theta}) \dots q(\mathbf{h}^L|\mathbf{h}^{L-1}, \boldsymbol{\theta})$$



Each term may denote a complicated nonlinear relationship

Typically, directed layers are parametrized as:

$$q(\mathbf{h}^l|\mathbf{h}^{l-1}, \boldsymbol{\theta}) = \mathcal{N}(\mu_{\boldsymbol{\theta}}(\mathbf{h}^{l-1}), \Sigma_{\boldsymbol{\theta}}(\mathbf{h}^{l-1}))$$

Means/covariances fns (e.g. one-layer neural net) of previous layer and parameters  $\boldsymbol{\theta}$ .

# Why is this called an “encoder”?

Max-likelihood can be written as:

$$\max_{\theta \in \Theta} \max_{\{q(h^L|x)\}} \sum_{i=1}^n H(q(h^L|x)) + \mathbb{E}_{q(h^L|x)}[\log p(x, h^L)]$$

Let's rewrite the ELBO a bit:

$$H(q(h^L|x)) + \mathbb{E}_{q(h^L|x)}[\log p(x, h^L)] = \mathbb{E}_{q(h^L|x)}[\log p(x, h^L) - \log q(h^L|x)]$$

$$= \mathbb{E}_{q(h^L|x)}[\log p(h^L) + \log p(x|h^L) - \log q(h^L|x)]$$

$$= \mathbb{E}_{q(h^L|x)} \log p(x|h^L) - \mathbb{E}_{q(h^L|x)} \log \frac{q(h^L|x)}{p(h^L)}$$

$$= \underbrace{\mathbb{E}_{q(h^L|x)} \log p(x|h^L)}_{\text{“Reconstruction” error}} - \underbrace{KL(q(h^L|x)||p(h^L))}_{\text{“Regularization towards prior”}}$$

“Reconstruction” error  
Use  $q$  as a “probabilistic” encoder,  
Use  $p$  as a “probabilistic” decoder,

$$x \rightarrow h^L \rightarrow x$$

“Regularization  
towards prior”

# How to train?

Max-likelihood can be written as:

$$\max_{\theta \in \Theta} \max_{\{q_{\theta}(h^L|x)\}} \sum_x \mathbb{E}_{q_{\theta}(h^L|x)} \log \frac{p_{\theta}(x, h^L)}{q_{\theta}(h^L|x)}$$

As usual: we need to be able to take gradients in  $\theta$

**The problem:** the expectation is with respect to  $q_{\theta}(h^L|x)$ , which depends on the variables we are taking a derivative with respect to.

**Observation:** a derivative of the type  $\nabla_{\theta} \mathbb{E}_p f(\theta)$  is easy to approximate if  $p$  does not depend on  $\theta$ :

$$\nabla_{\theta} \mathbb{E}_p f(\theta) = \mathbb{E}_p \nabla_{\theta} f(\theta) \quad \theta_i: \text{iid samples from } p$$

Exchange only works if  $p$  doesn't dep on  $\theta$

$$\approx \frac{1}{N} \sum_i \nabla_{\theta_i} f(\theta_i)$$

# How to train?

Max-likelihood can be written as:

$$\max_{\theta \in \Theta} \max_{\{q_{\theta}(h^L|x)\}} \sum_x \mathbb{E}_{q_{\theta}(h^L|x)} \log \frac{p_{\theta}(x, h^L)}{q_{\theta}(h^L|x)}$$

As usual: we need to be able to take gradients in  $\theta$

**Remark:** there is a common Monte Carlo estimator of  $\nabla_{\theta} \mathbb{E}_{p(\theta)} f(\theta)$  as well but it typically has high variance:

$$\nabla_{\theta} \mathbb{E}_{p(\theta)} f(\theta) = \int \nabla_{\theta} f(\theta) p(\theta) d\theta + \int f(\theta) \nabla_{\theta} p(\theta) d\theta$$

$$= \int \nabla_{\theta} f(\theta) p(\theta) d\theta + \int f(\theta) \frac{p(\theta)}{p(\theta)} \nabla_{\theta} p(\theta) d\theta$$

$$= \int \nabla_{\theta} f(\theta) p(\theta) d\theta + \int f(\theta) \nabla_{\theta} \log p(\theta) p(\theta) d\theta$$

$$= \mathbb{E}_{p(\theta)} [\nabla_{\theta} f(\theta) + f(\theta) \nabla_{\theta} \log p(\theta)] \longrightarrow \text{This term typically is high var,}$$

# How to train?

Max-likelihood can be written as:

$$\max_{\theta \in \Theta} \max_{\{q_{\theta}(h^L|x)\}} \sum_x \mathbb{E}_{q_{\theta}(h^L|x)} \log \frac{p_{\theta}(x, h^L)}{q_{\theta}(h^L|x)}$$

As usual: we need to be able to take gradients in  $\theta$

**The solution:** write the expectation  $\mathbb{E}_{q_{\theta}(h^L|x)} \log \frac{p_{\theta}(x, h^L)}{q_{\theta}(h^L|x)}$  as an expectation over a distribution not dependent on  $\theta$ .

*Kingma-Welling '13:* reparametrization trick!

**Main idea:** a sample from  $\mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$  can be generated as follows

Sample  $\mathbf{x} \sim \mathcal{N}(0, \mathbf{I})$ .

Output  $\mathbf{y} = \boldsymbol{\mu} + \boldsymbol{\Sigma}^{1/2} \mathbf{x}$ .

# How to train?

Max-likelihood can be written as:

$$\max_{\theta \in \Theta} \max_{\{q_{\theta}(h^L|x)\}} \sum_x \mathbb{E}_{q_{\theta}(h^L|x)} \log \frac{p_{\theta}(x, h^L)}{q_{\theta}(h^L|x)}$$

As usual: we need to be able to take gradients in  $\theta$

Recall that  $q(\mathbf{h}|\mathbf{x}, \boldsymbol{\theta}) = q(\mathbf{h}^1|\mathbf{x}, \boldsymbol{\theta})q(\mathbf{h}^2|\mathbf{h}^1, \boldsymbol{\theta}) \dots q(\mathbf{h}^L|\mathbf{h}^{L-1}, \boldsymbol{\theta})$

where  $q(\mathbf{h}^l|\mathbf{h}^{l-1}, \boldsymbol{\theta}) = \mathcal{N}(\mu_{\theta}(\mathbf{h}^{l-1}), \Sigma_{\theta}(\mathbf{h}^{l-1}))$

To produce a sample from  $q(\mathbf{h}|\mathbf{x}, \boldsymbol{\theta})$ , sample iid standard Gaussians  $\boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_2, \dots, \boldsymbol{\epsilon}_L$ . Set

$$\mathbf{h}^{\ell}(\boldsymbol{\epsilon}^{\ell}, \mathbf{h}^{\ell-1}, \boldsymbol{\theta}) = \Sigma(\mathbf{h}^{\ell-1}, \boldsymbol{\theta})^{1/2} \boldsymbol{\epsilon}^{\ell} + \boldsymbol{\mu}(\mathbf{h}^{\ell-1}, \boldsymbol{\theta})$$



# Using the reparametrization trick

Max-likelihood can be written as:

$$\max_{\theta \in \Theta} \max_{\{q_{\theta}(h)\}} \sum_x \mathbb{E}_{q_{\theta}(h|x)} \log \frac{p_{\theta}(x, h)}{q_{\theta}(h|x)}$$

We can hence write the gradient wrt to  $\theta$  :

$$\begin{aligned} \nabla_{\theta} \mathbb{E}_{\mathbf{h} \sim q(\mathbf{h}|\mathbf{x}, \theta)} \left[ \log \frac{p(\mathbf{x}, \mathbf{h}|\theta)}{q(\mathbf{h}|\mathbf{x}, \theta)} \right] \\ = \nabla_{\theta} \mathbb{E}_{\epsilon^1, \dots, \epsilon^L \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[ \log \frac{p(\mathbf{x}, \mathbf{h}(\epsilon, \mathbf{x}, \theta)|\theta)}{q(\mathbf{h}(\epsilon, \mathbf{x}, \theta)|\mathbf{x}, \theta)} \right] \\ = \mathbb{E}_{\epsilon^1, \dots, \epsilon^L \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[ \nabla_{\theta} \log \frac{p(\mathbf{x}, \mathbf{h}(\epsilon, \mathbf{x}, \theta)|\theta)}{q(\mathbf{h}(\epsilon, \mathbf{x}, \theta)|\mathbf{x}, \theta)} \right] \end{aligned}$$

# Using the reparametrization trick

Max-likelihood can be written as:

$$\max_{\theta \in \Theta} \max_{\{q_{\theta}(h)\}} \sum_x \mathbb{E}_{q_{\theta}(h|x)} \log \frac{p_{\theta}(x, h)}{q_{\theta}(h|x)}$$

We can hence write the gradient wrt to  $\theta$  :

$$\begin{aligned} \nabla_{\theta} \mathbb{E}_{\mathbf{h} \sim q(\mathbf{h}|\mathbf{x}, \theta)} \left[ \log \frac{p(\mathbf{x}, \mathbf{h}|\theta)}{q(\mathbf{h}|\mathbf{x}, \theta)} \right] &= \nabla_{\theta} \mathbb{E}_{\epsilon^1, \dots, \epsilon^L \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[ \log \frac{p(\mathbf{x}, \mathbf{h}(\epsilon, \mathbf{x}, \theta)|\theta)}{q(\mathbf{h}(\epsilon, \mathbf{x}, \theta)|\mathbf{x}, \theta)} \right] \\ &= \mathbb{E}_{\epsilon^1, \dots, \epsilon^L \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[ \nabla_{\theta} \log \frac{p(\mathbf{x}, \mathbf{h}(\epsilon, \mathbf{x}, \theta)|\theta)}{q(\mathbf{h}(\epsilon, \mathbf{x}, \theta)|\mathbf{x}, \theta)} \right] \end{aligned}$$

We can approximate the expectation by an empirical average as before.

For **fixed**  $\epsilon_1, \epsilon_2, \dots, \epsilon_L$ :  $\log p$  and  $\log q$  are easy to take gradients of via backpropagating.

It's common to have **diagonal covariance mxs** for training efficiency.

## Part II: Evaluating representations

# Desiderata for representations

## What do we want out a representation?

Many possible answers here. First, a few uncontroversial desiderata:

***Interpretability***: if the derived features are semantically meaningful, and interpretable by a human, they can be easily evaluated.  
(e.g. noisy-OR: “features” are diseases a patient has)

***Sparsity*** of a representation is an important subcase: “explanatory” features for sample can be examined if there are a small number of them.

***Downstream usability***: the features are “useful” for downstream tasks. Some examples:

***Improving label efficiency***: if, for a task, a linear (or otherwise “simple”) classifier can be trained on features and it works well, smaller # of labeled samples are needed.

# Desiderata for representations

**Obvious issue:** interpretability and “usefulness” are not easily mathematically expressed. We need some “proxies” that induce such properties.

This is a lot more controversial – here we survey some general desiderata, proposed as early as *Bengio-Courville-Vincent '14*:

***Hierarchy/compositionality***: video/images/text/ are expected to have hierarchical structure – depth helps induce such structure.

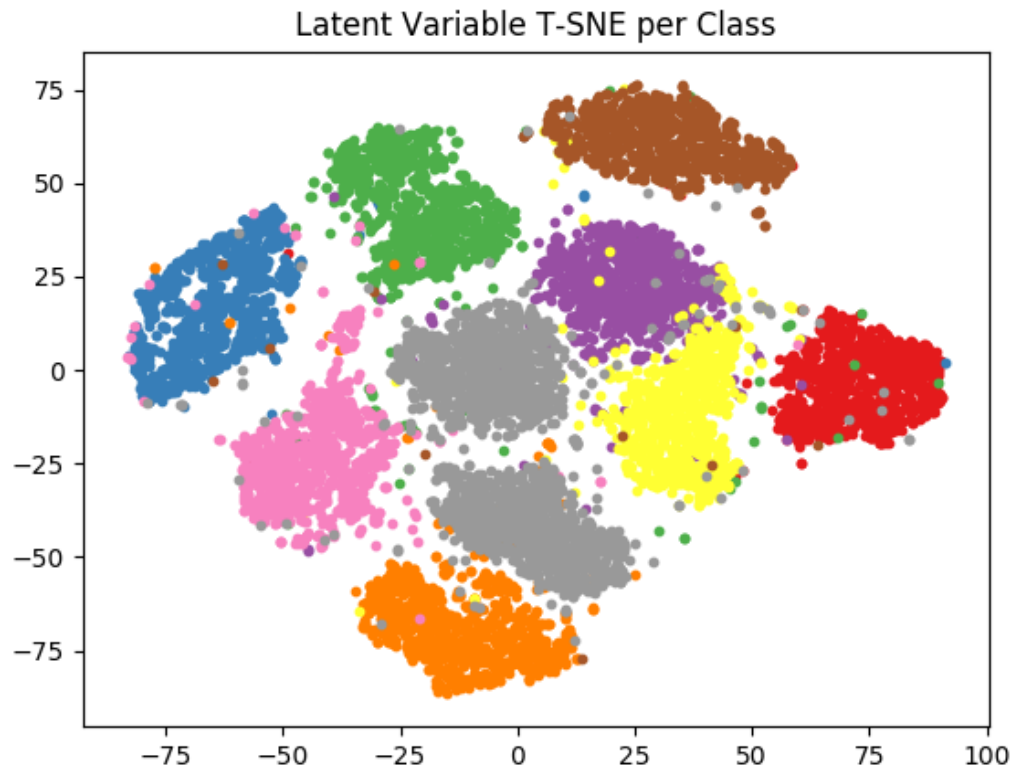
***Semantic clusterability***: features of the same “semantic class” (e.g. images in the same category) are clustered.

***Linear interpolation***: in representation space, linear interpolations produce meaningful data points (i.e. “latent space is convex”). Sometimes called *manifold flattening*.

***Disentangling***: features capture “independent factors of variation” of data. (*Bengio-Courville-Vincent '14*). Has been very popular in modern unsupervised learning, though many potential issues with it.

# Semantic clustering

***Semantic clusterability***: features of the same “semantic class” (e.g. images in the same category) are clustered together.



*The intuition:*

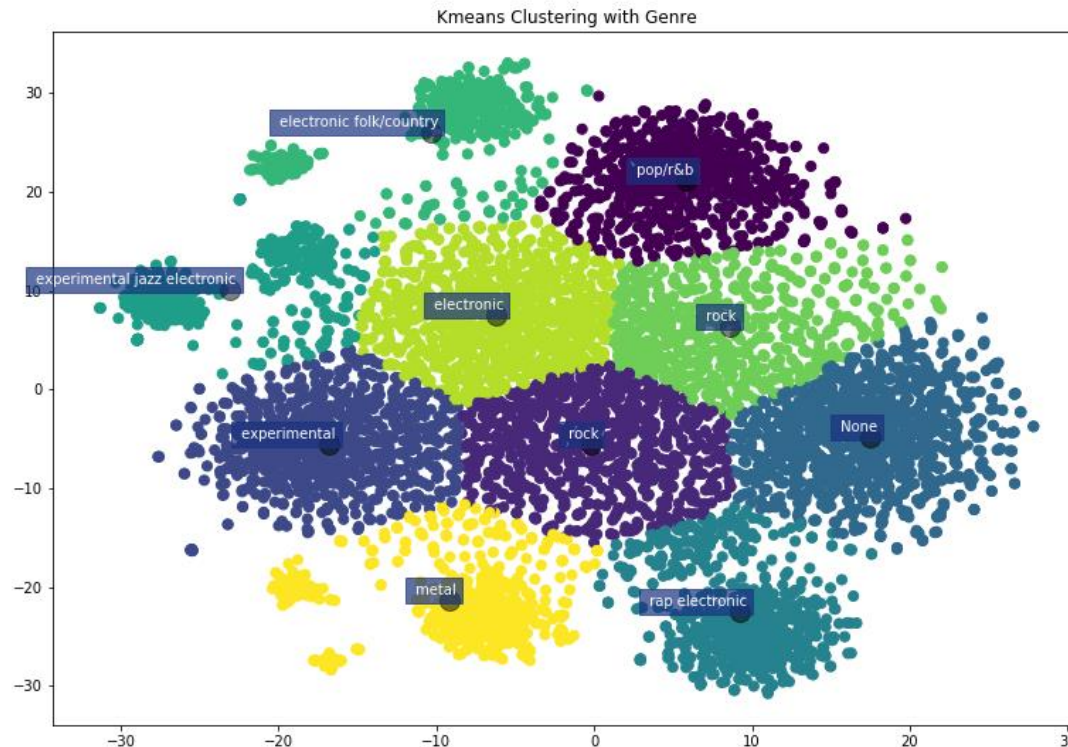
If semantic classes are linearly (or other simple function) separable, and labels on downstream tasks depend linearly on semantic classes – can afford to learn a simple classifier !!

t-SNE projection of VAE-learned features of the 10 MNIST classes.

Image from <https://pyro.ai/examples/vae.html>

# Semantic clustering

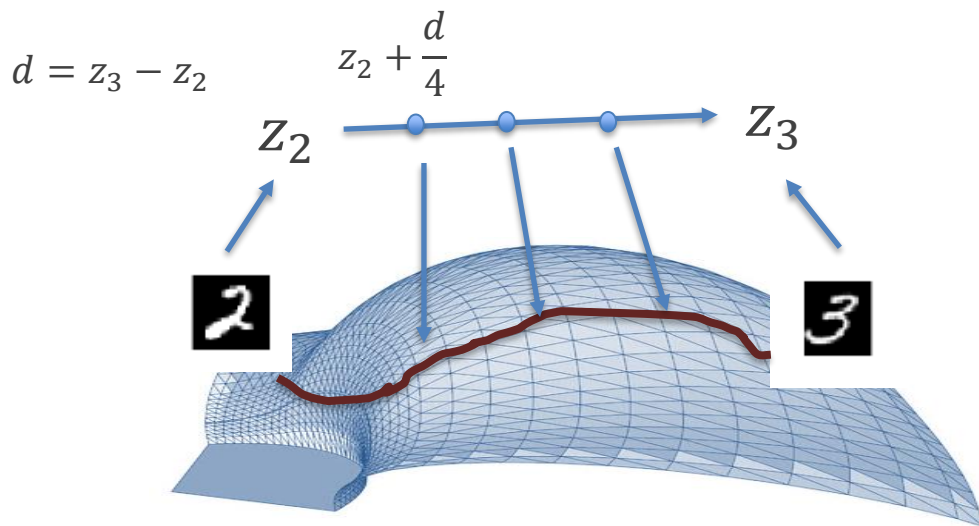
***Semantic clusterability***: features of the same “semantic class” (e.g. images in the same category) are clustered together.



t-SNE projection of word embeddings for artists (clustered by genre).  
Image from <https://medium.com/free-code-camp/learn-tensorflow-the-word2vec-model-and-the-tsne-algorithm-using-rock-bands-97c99b5dcb3a>

# Linear interpolation

**Linear interpolation:** in representation space, linear interpolations produce meaningful data points. (i.e. “latent space is convex”)



*The intuition:*

The data manifold is complicated/curved.

The latent variable manifold is a convex set – moving in straight lines keeps us on it.

*Interpolations for a VAE trained on MNIST.*



# Linear interpolation

***Linear interpolation:*** in representation space, linear interpolations produce meaningful data points. (i.e. “latent space is convex”)



*Interpolations for a BigGAN, image from*  
<https://thegradient.pub/bigganex-a-dive-into-the-latent-space-of-biggan/>

# Disentangled representations

**Disentangling:** features capture “independent factors of variation” of data. (*Bengio-Courville-Vincent '14*). Has been very popular in modern unsupervised learning, though many potential issues with it.

For concreteness, let's assume that we have a latent variable model for data with latent variables  $\mathbf{z}$ , observables  $\mathbf{x}$ , and joint distribution  $p_{\theta}(\mathbf{z}, \mathbf{x})$

There are (at least) two ways to formalize this (literature is not always clear on which one is aimed for!):

**Prior disentangling:**  $p_{\theta}(\mathbf{z})$  is a product distribution, i.e.  $p_{\theta}(\mathbf{z}) = \prod_i p_{\theta}(\mathbf{z}_i)$

Classical example: ICA (independent component analysis)

**Posterior disentangling:** fit a variational posterior  $q_{\theta}$  s.t.  $q_{\theta}(\mathbf{z}|\mathbf{x})$  is (on average over  $\mathbf{x}$ ) a product distribution

In other words,  $\int_{\mathbf{x}} q_{\theta}(\mathbf{z}|\mathbf{x}) p(\mathbf{x}) d\mathbf{x}$  – usually called the *aggregate posterior* – is close to a product distribution.

# Disentangled representations

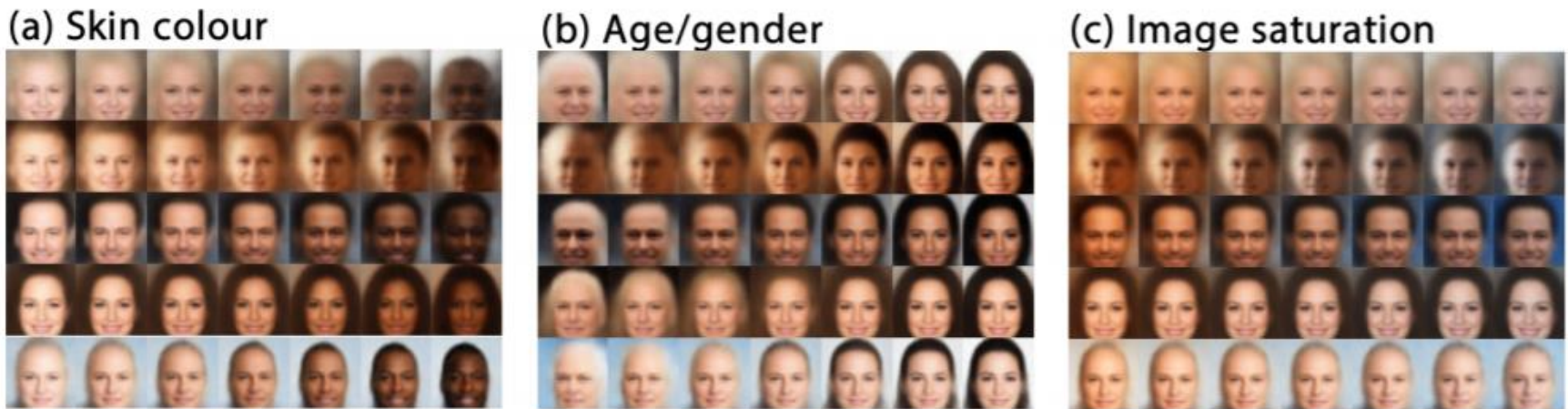


Figure 4: **Latent factors learnt by  $\beta$ -VAE on celebA:** traversal of individual latents demonstrates that  $\beta$ -VAE discovered in an unsupervised manner factors that encode skin colour, transition from an elderly male to younger female, and image saturation.

*Posterior disentangling in  $\beta$  - VAE. To produce plots, infer latent variable for an image, then change a single latent variable gradually.*

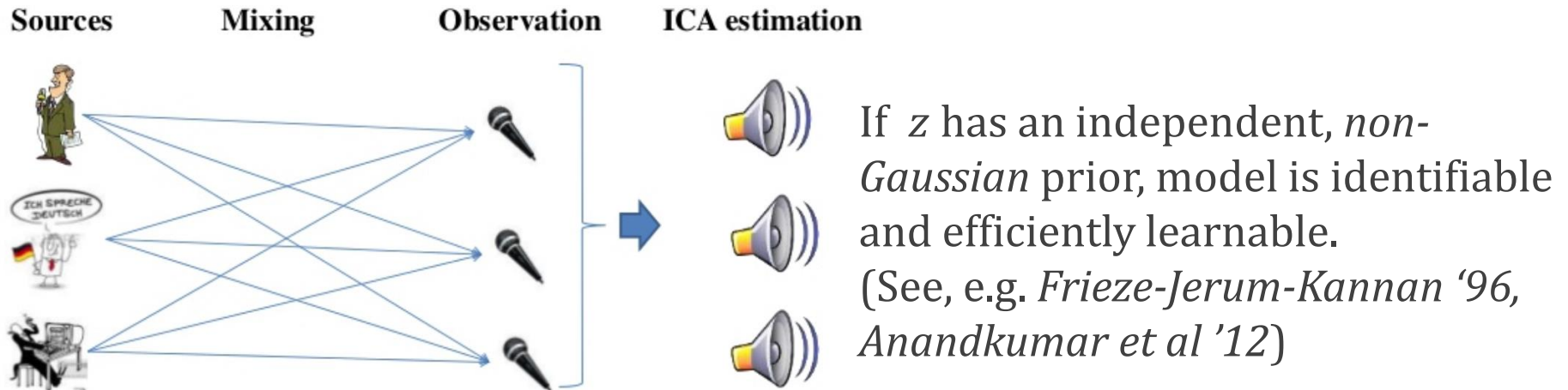
*Image from Higgins et al. '17.*

# Prior disentangling

**Prior disentangling:**  $p_{\theta}(\mathbf{z})$  is a product distribution, i.e.  $p_{\theta}(\mathbf{z}) = \prod_i p_{\theta}(\mathbf{z}_i)$

*Classical example:* ICA (independent component analysis), also called the “cocktail party problem”.

Assume data is generated as  $\mathbf{x} = \mathbf{A}\mathbf{z}$ ,  $\mathbf{z} \in \mathbb{R}^d$ ,  $\mathbf{A} \in \mathbb{R}^{d \times d}$



*Other examples:* noisy-OR networks (diseases are independent), general Bayesian nets, viewing top variables as  $\mathbf{z}$ 's, GANs, ...

# Posterior disentanglement in VAEs

Recall the “regularization” view of the VAEs objective:

$$\sum_x \underbrace{\mathbb{E}_{q(h^L|x)} \log p(x|h^L)}_{\text{“Reconstruction” error}} - \underbrace{KL(q(h^L|x)||p(h^L))}_{\text{“Regularization towards prior”}}$$

Consider a prior which is a product distribution (e.g. standard Gaussian):

The KL term implicitly penalizes distributions for which

$$\sum_x KL(q(h^L|x)||p(h^L)) \approx \mathbb{E}_{x \sim p^*} KL(q(h^L|x)||p(h^L))$$

is large – i.e. the aggregated posterior is far from a product distribution.

# Posterior disentanglement in VAEs

Recall the “regularization” view of the VAEs objective:

$$\underbrace{\sum_x \mathbb{E}_{q(h^L|x)} \log p(x|h^L)}_{\text{“Reconstruction” error}} - \underbrace{KL(q(h^L|x)||p(h^L))}_{\text{“Regularization towards prior”}}$$

The KL term implicitly penalizes distributions for which

$$\sum_x KL(q(h^L|x)||p(h^L)) \approx \mathbb{E}_{x \sim p^*} KL(q(h^L|x)||p(h^L))$$

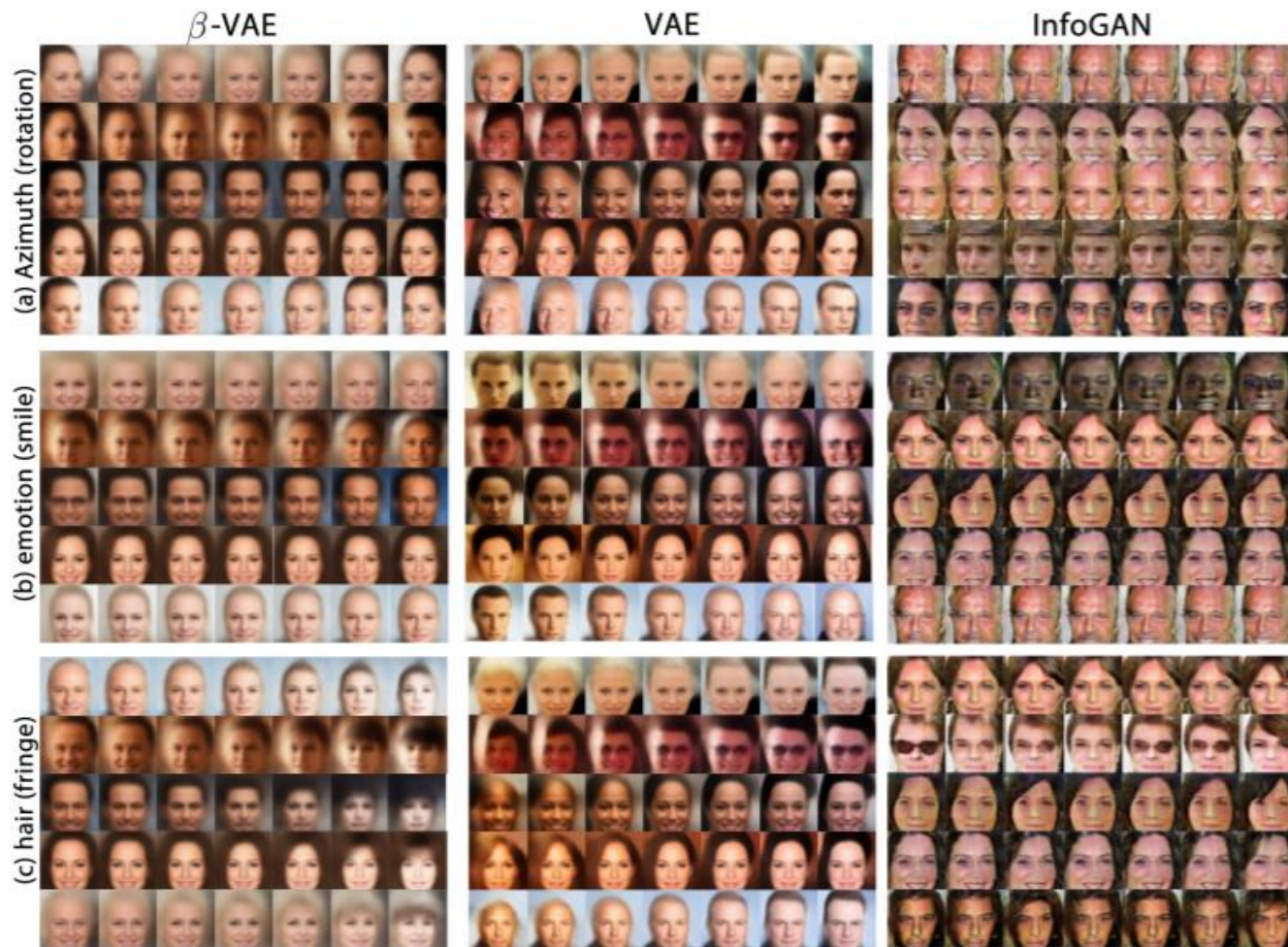
The idea of *Higgins et al '17*: introduce a “weighting” factor to put more weight on reconstruction or disentanglement:

**$\beta$  – VAE objective:**  $\sum_x \mathbb{E}_{q(h^L|x)} \log p(x|h^L) - \beta KL(q(h^L|x)||p(h^L))$

$\beta$  large: more weight on disentanglement

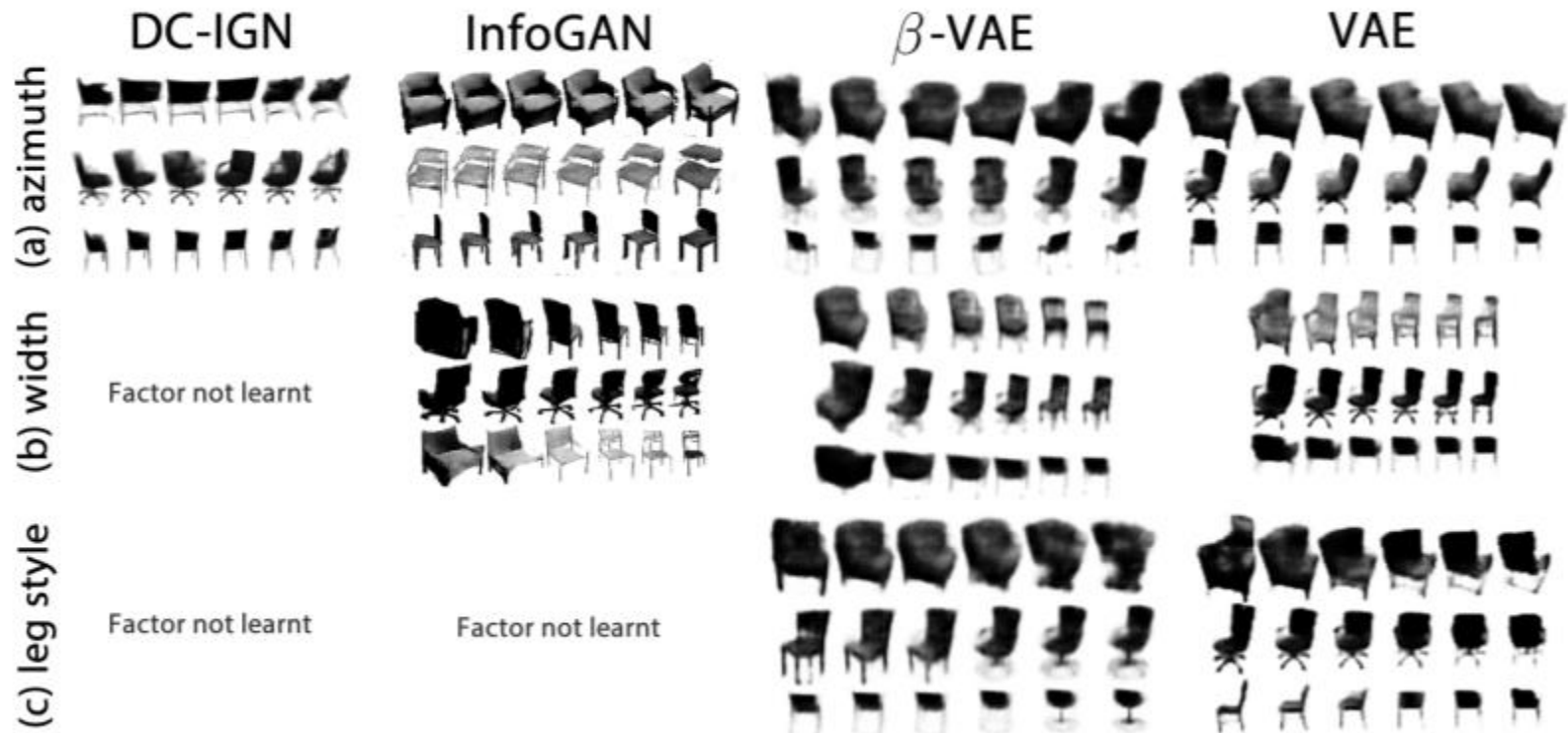


# Posterior disentanglement in VAEs



*Comparing disentangling of different types of generative models.  
Image from Higgins et al. '17.*

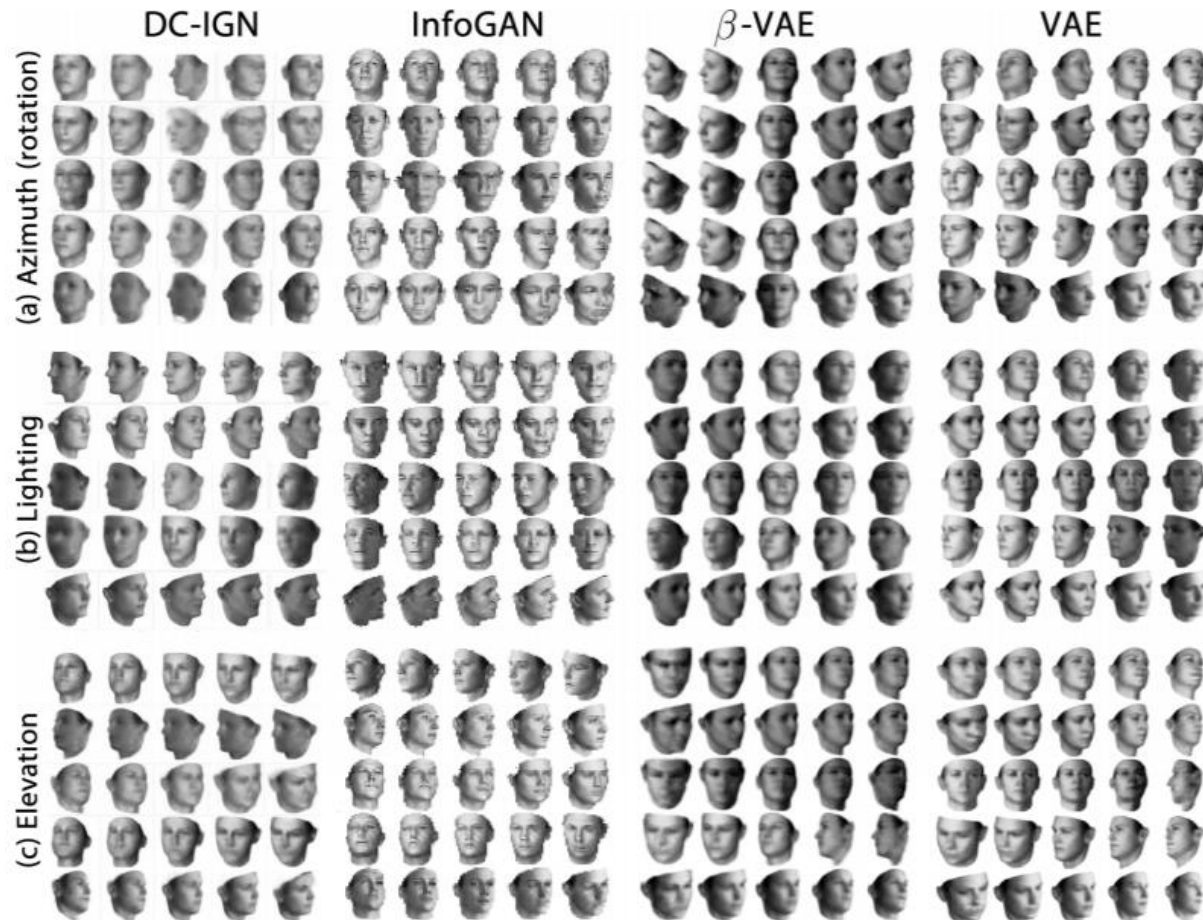
# Posterior disentanglement in VAEs



*Comparing disentangling of different types of generative models.  
Image from Higgins et al. '17.*



# Posterior disentanglement in VAEs



*Comparing disentangling of different types of generative models.  
Image from Higgins et al. '17.*

# Measuring disentanglement

*Locatello et al '19, "Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations"* (Best paper award at ICML '19):  
A large-scale study of disentanglement measures, as well as gen. models.

Dataset = Noisy-dSprites

BetaVAE Score (A)	100	80	44	41	46	37
FactorVAE Score (B)	80	100	49	52	25	38
MIG (C)	44	49	100	76	6	42
DCI Disentanglement (D)	41	52	76	100	-8	38
Modularity (E)	46	25	6	-8	100	13
SAP (F)	37	38	42	38	13	100
	(A)	(B)	(C)	(D)	(E)	(F)

*Figure 2. Rank correlation of different metrics on Noisy-dSprites. Overall, we observe that all metrics except Modularity seem mildly correlated with the pairs BetaVAE and FactorVAE, and MIG and DCI Disentanglement strongly correlated with each other.*

# Usefulness of disentanglement?

*Locatello et al '19, "Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations"* (Best paper award at ICML '19): A large-scale study of disentanglement measures, as well as gen. models.

Dataset = dSprites

BetaVAE Score	18	65	28	28	67	78	75	76	50	50
FactorVAE Score	13	49	13	12	58	73	71	71	43	46
MIG	18	63	20	-1	71	86	86	87	62	47
DCI Disentanglement	19	65	18	4	75	94	94	94	62	54
Modularity	-3	-9	15	18	-6	-17	-19	-13	-19	-14
SAP	12	64	20	12	71	77	74	75	56	49
	LR10	LR100	LR1000	LR10000	GBT10	GBT100	GBT1000	GBT10000	Efficiency (LR)	Efficiency (GBT)

*Figure 5. Rank correlations between disentanglement metrics and downstream performance (accuracy and efficiency) on dSprites.*

*Downstream classification task:* predict **true** ground-truth factors (w/ multiclass logistic regression)

Careful to extrapolate too much – task/setup is a little contrived.

# Usefulness of disentanglement?

*Locatello et al '19, "Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations"* (Best paper award at ICML '19): A large-scale study of disentanglement measures, as well as gen. models.

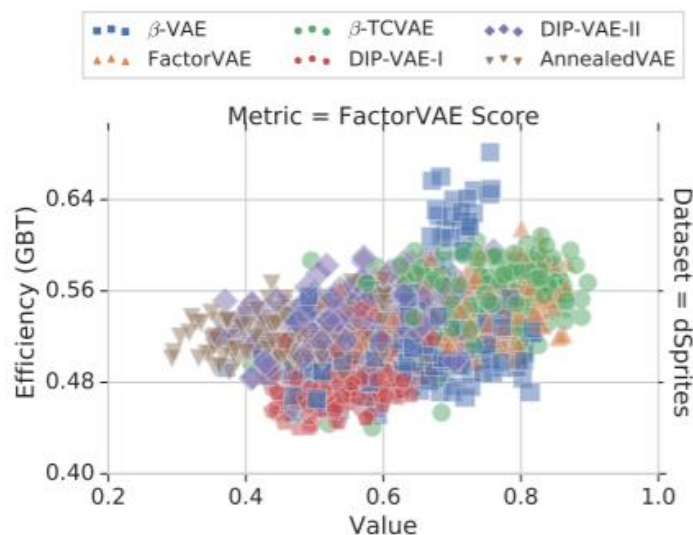


Figure 6. Statistical efficiency of the FactorVAE Score for learning a GBT downstream task on dSprites.

*Statistical efficiency measure:* average accuracy based on 100 samples divided by the average accuracy based on 10 000 samples

# Issue of ill-posedness?

*Locatello et al '19, "Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations"* (Best paper award at ICML '19):

A model can be re-parametrized, s.t. the distribution over the data is unchanged, but it can be arbitrarily more “entangled”.

Thus, some kind of **inductive bias** both on model class and data seems necessary.

As a simple example: consider  $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ , let  $\mathbf{z}' = \mathbf{U}\mathbf{z}$ , for any non-identity orthogonal matrix  $\mathbf{U}$ .

Then, under any “intuitive” understanding of entangling,  $\mathbf{z}'$  seems **entangled** with  $\mathbf{z}$  – small changes of coordinates of  $\mathbf{z}$  cause global changes in  $\mathbf{z}'$ .