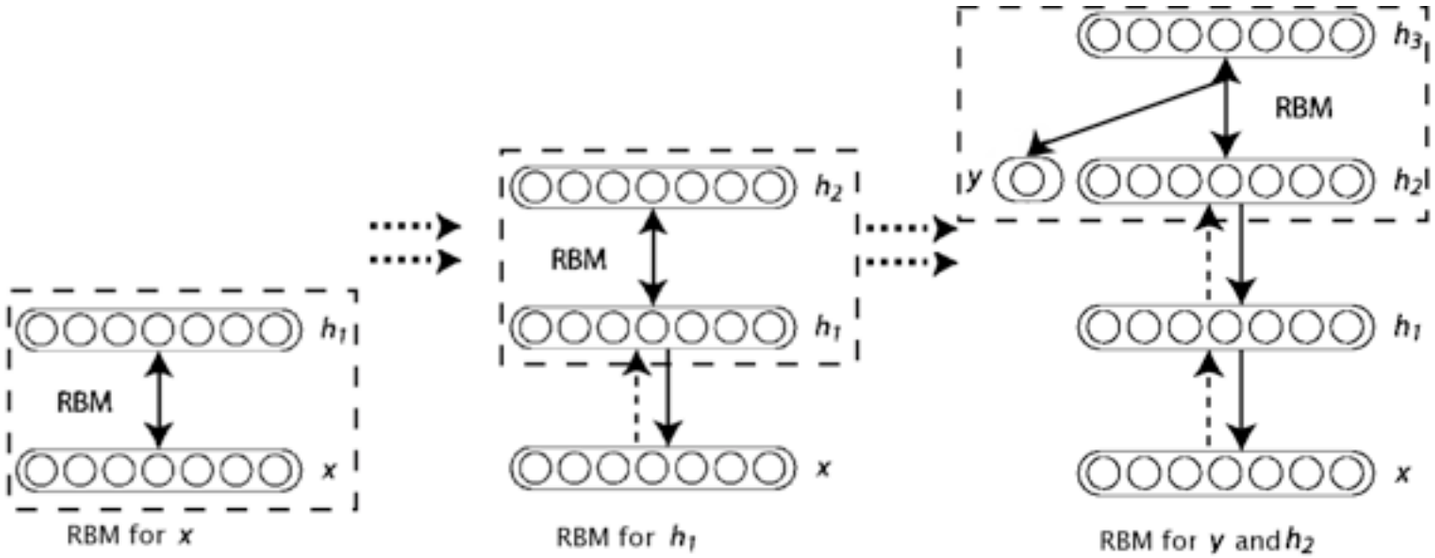
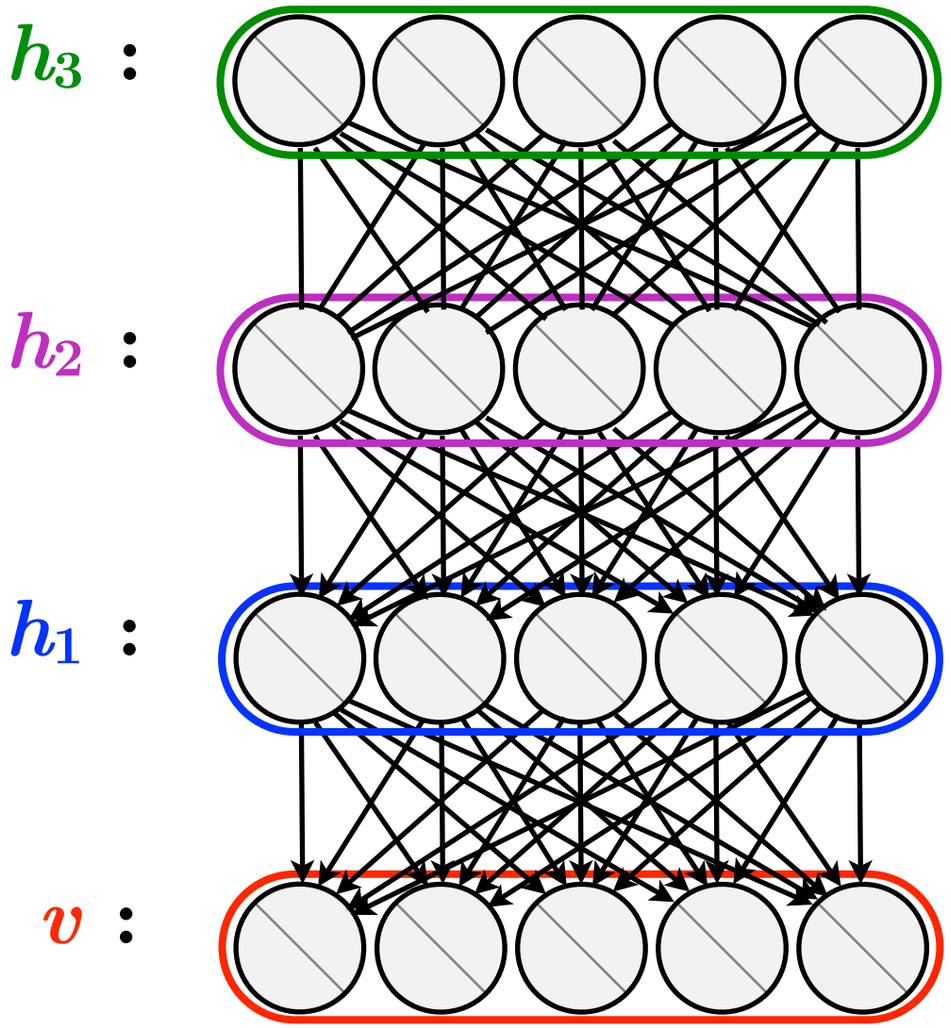
Should you use DBMs for image modeling?

- Well, it depends what you want.
- Let’s just return to the DBN for a minute...

Back to the deep belief network

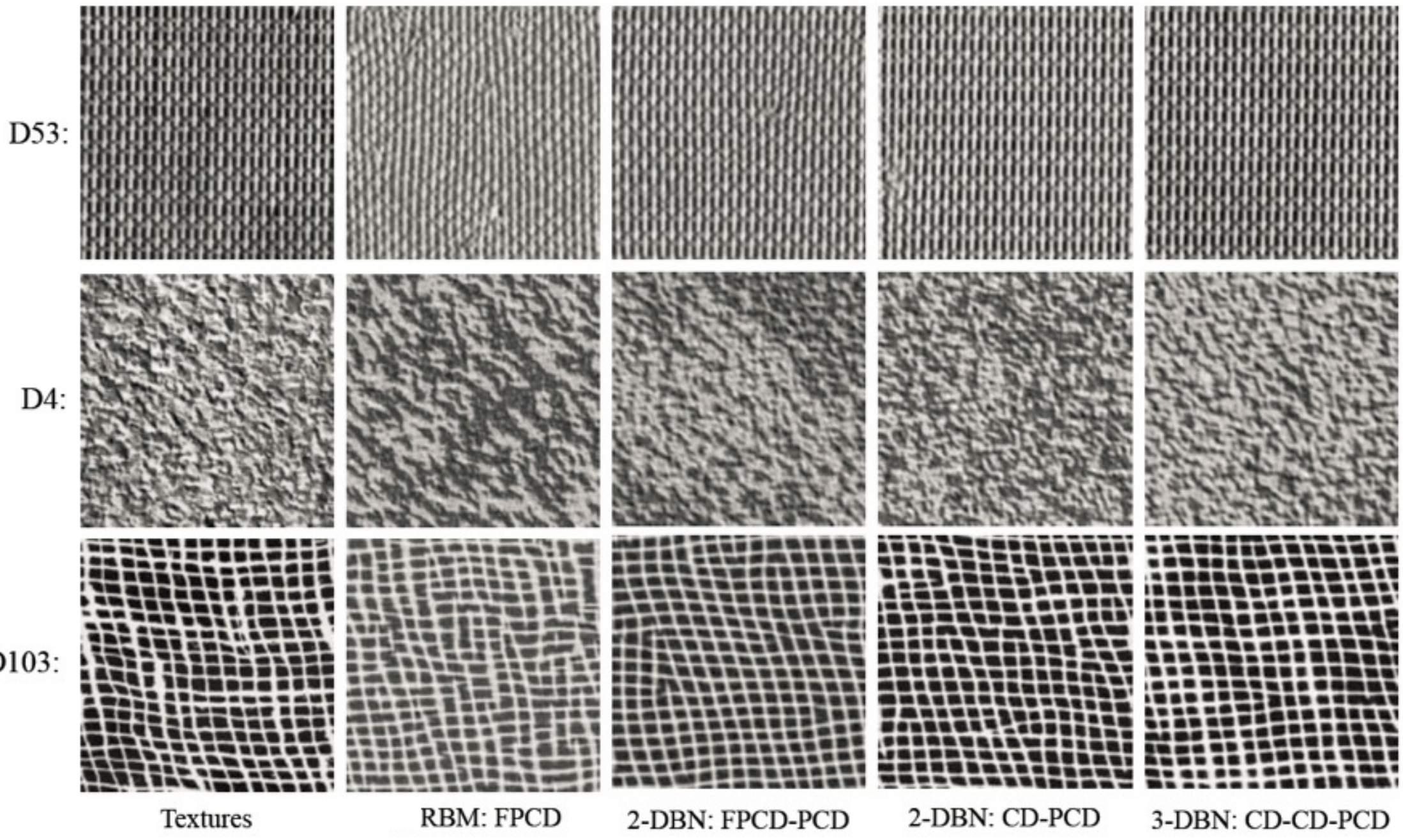
- The Deep Belief Network as a generative model.





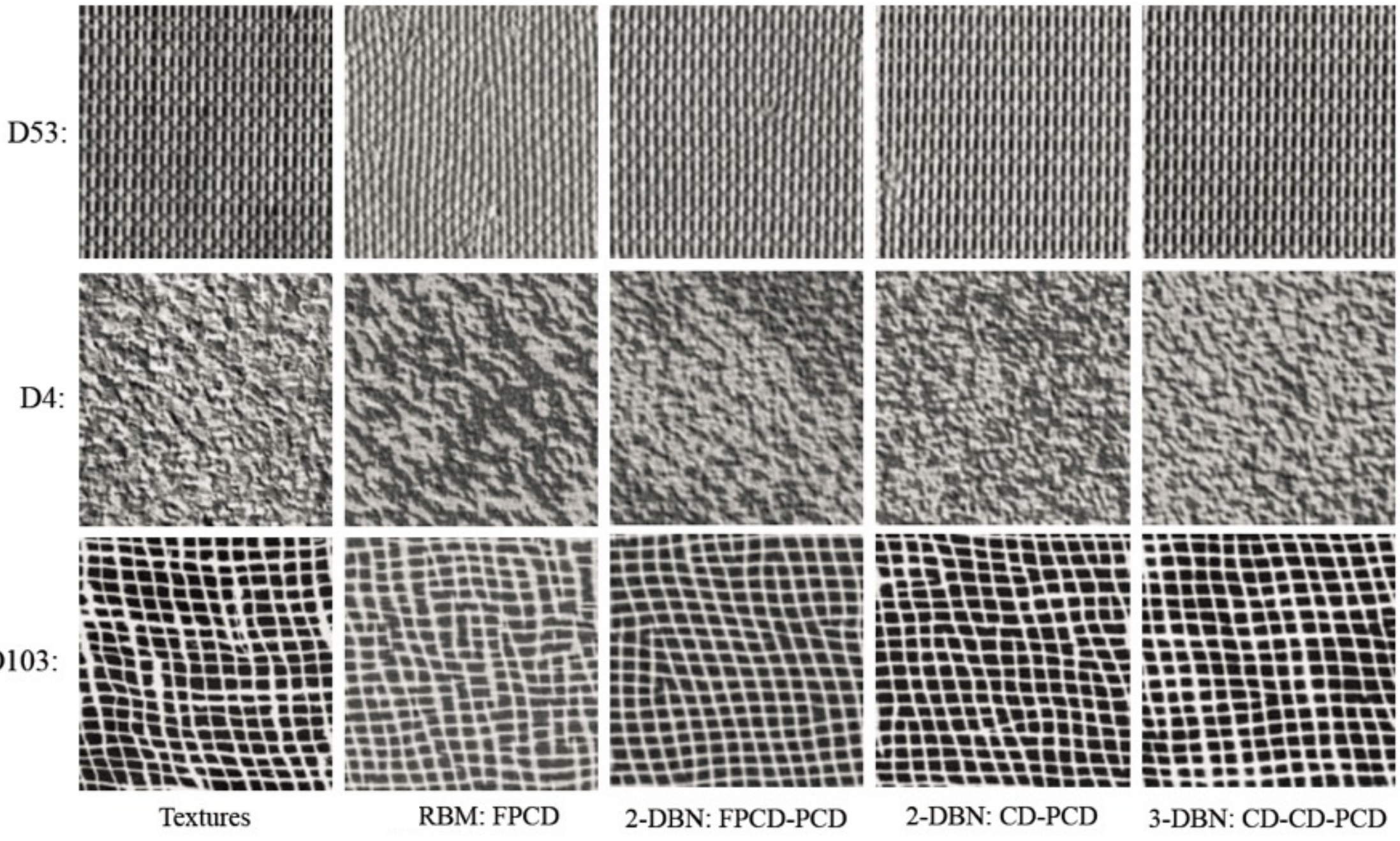
Deep belief network

Modeling Textures with DBNs

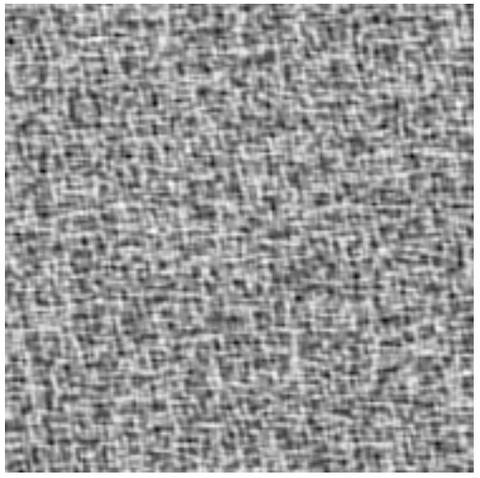


Modeling Textures with DBNs

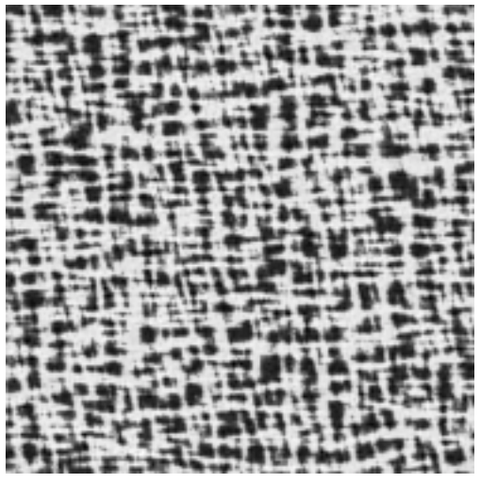
Why does depth help?



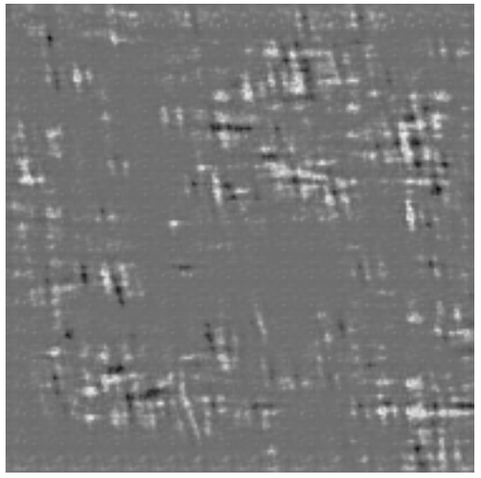
Depth helps mixing



1-layer model



2-layer model



3-layer model

Why does depth help mixing?

- There are two perspective on why depth helps mixing:

  1. Noise is induced in each successive layer added to the model, smoothing out the distribution.

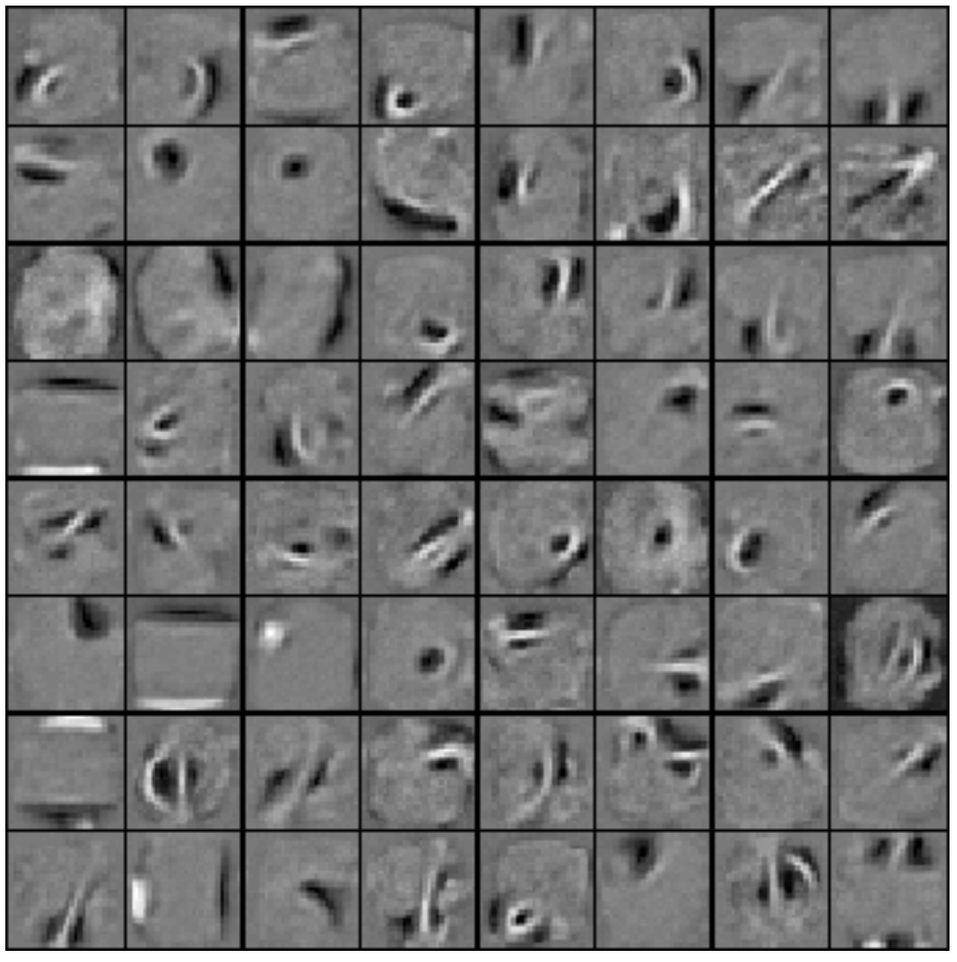
  2. The highly-structured manifold on which the data lies becomes unfolded and “simpler” at higher layers of representation.

- The underlying **factors of variation** are being **disentangled**.

Mixing and the model

- We face a big problem when training lower layer models
- If stochasticity in the lower layers does not express natural variations in your dataset, the ability of your model to mix will diminish with training.

MNIST dataset

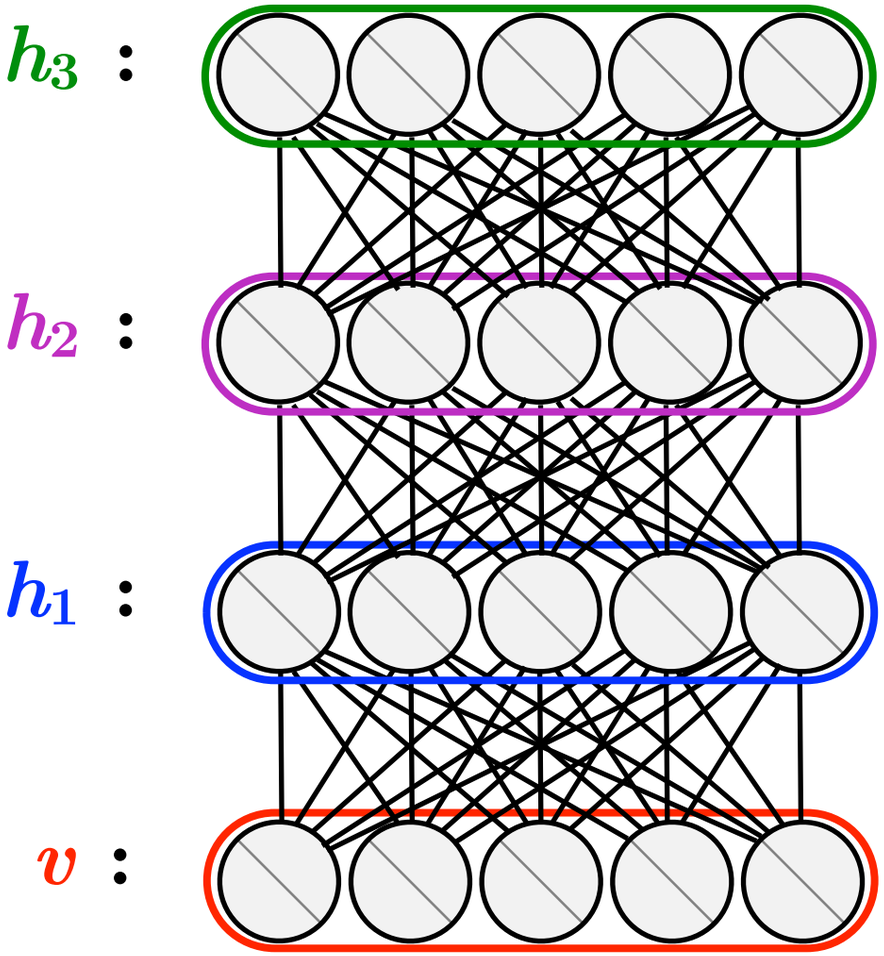


1st layer features (RBM)

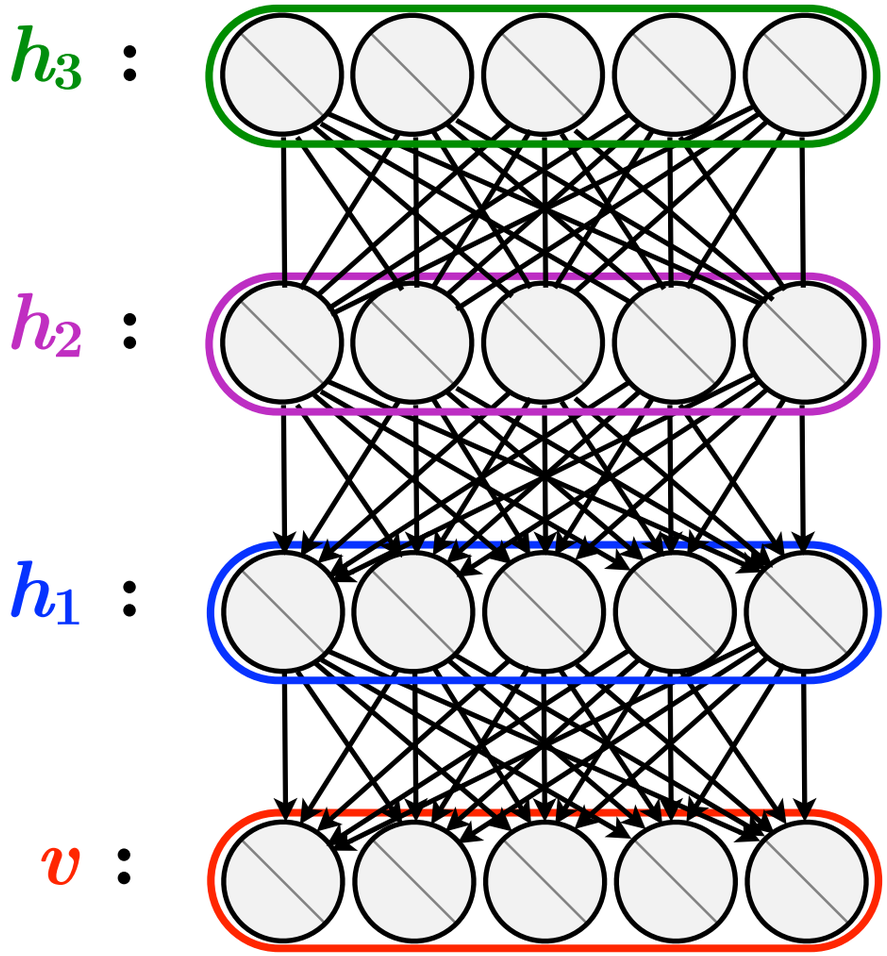
DBNs and DBMs: in the same boat

It is difficult to take either one too seriously as a model of natural images.

Why? Because stochasticity in the lower layers will push the sample off the manifold of natural images



Deep Boltzmann machine



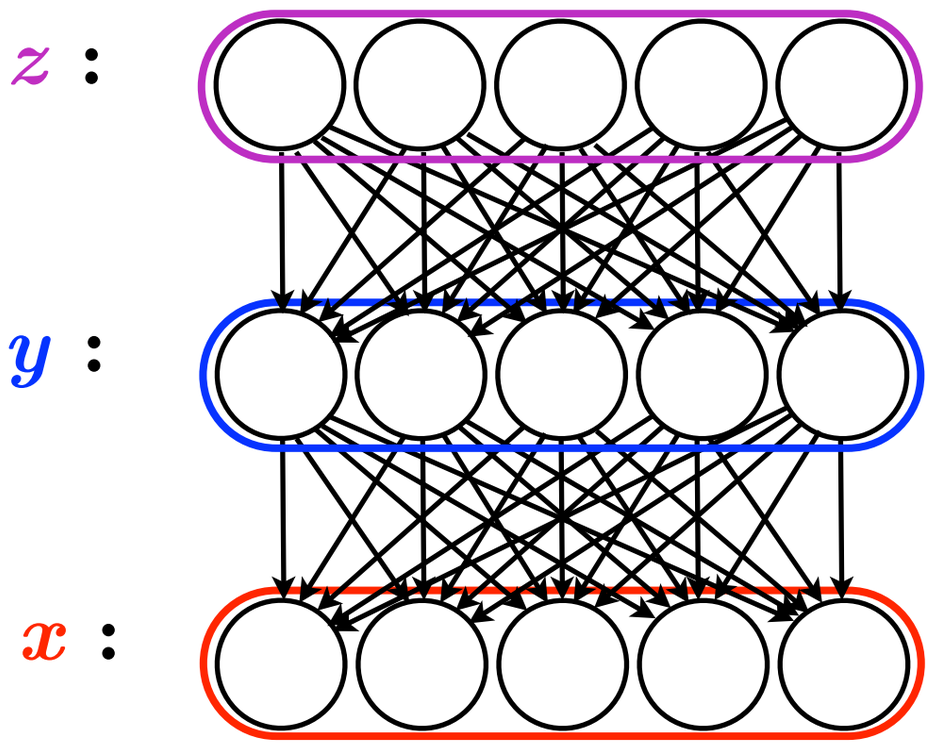
Deep belief network

Directed generative models

8

Deep directed graphical models

- The Variational Autoencoder model:
  - Kingma and Welling, *Auto-Encoding Variational Bayes*, *International Conference on Learning Representations (ICLR)* 2014.
  - Rezende, Mohamed and Wierstra, *Stochastic back-propagation and variational inference in deep latent Gaussian models*. ArXiv.
- Unlike RBM, DBM, here we are interested in deep directed graphical models:



Latent variable generative model

- **latent variable model:** learn a mapping from some latent variable z to a complicated distribution on x.

$$p(x) = \int p(x, z)\ dz \quad \text{where} \quad p(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{x} \mid \boldsymbol{z})p(\boldsymbol{z})$$

$$p(\boldsymbol{z}) = \text{something simple} \qquad p(\boldsymbol{x} \mid \boldsymbol{z}) = f(\boldsymbol{z})$$

- Can we learn to decouple the true **explanatory factors** underlying the data distribution? E.g. separate identity and expression in face images

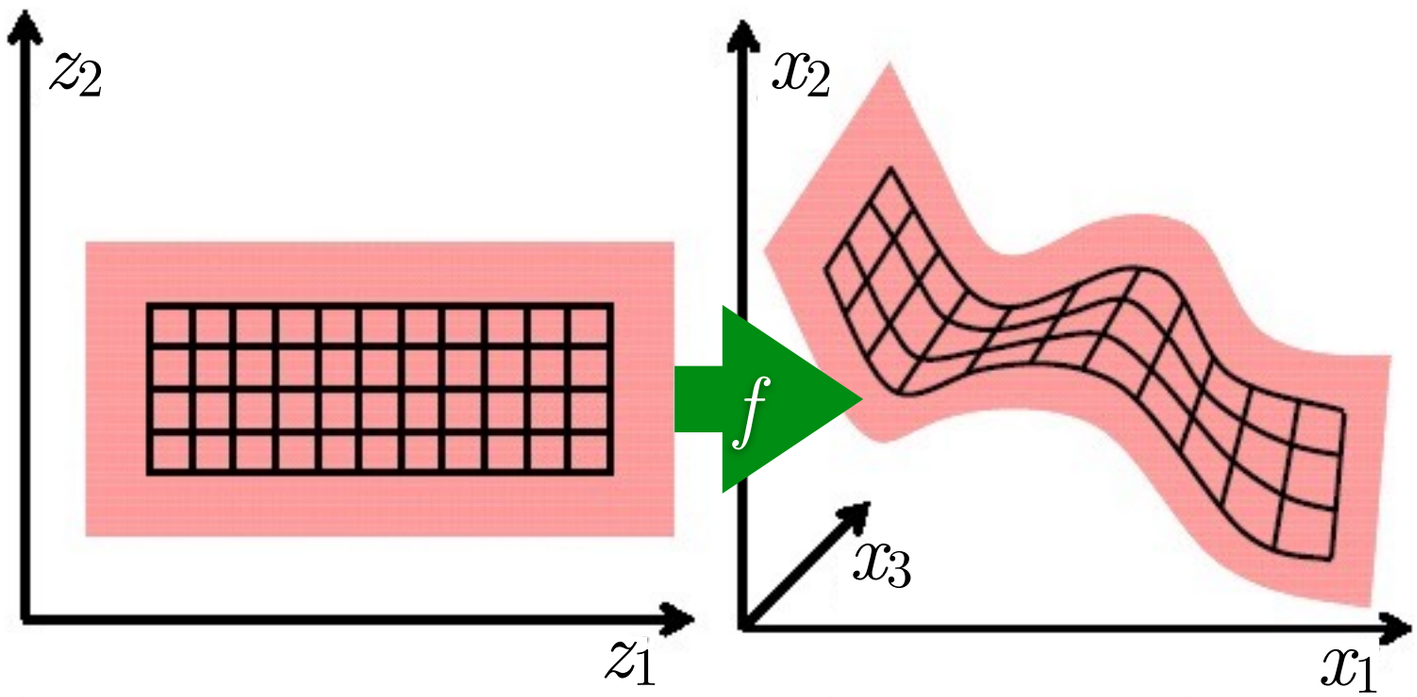


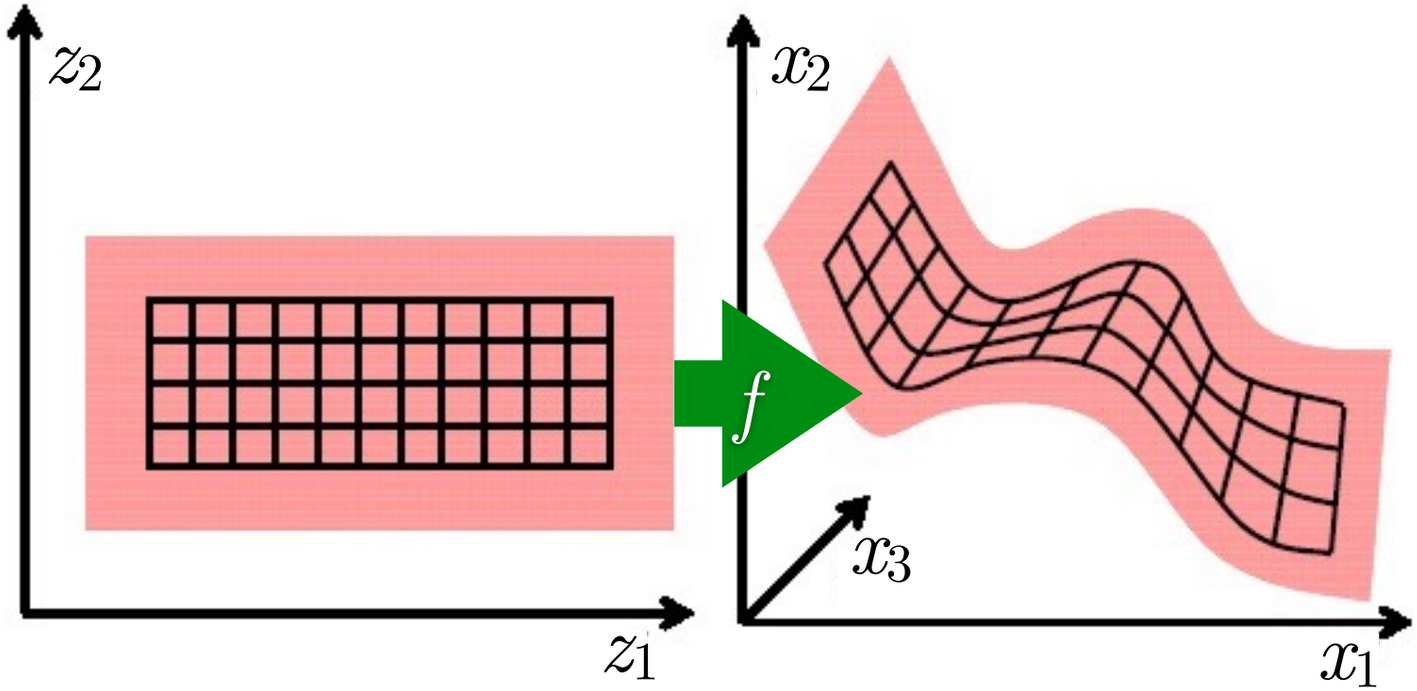
Image from: Ward, A. D., Hamarneh, G.: **3D Surface Parameterization Using Manifold Learning for Medial Shape Representation**, *Conference on Image Processing, Proc. of SPIE Medical Imaging*, 2007

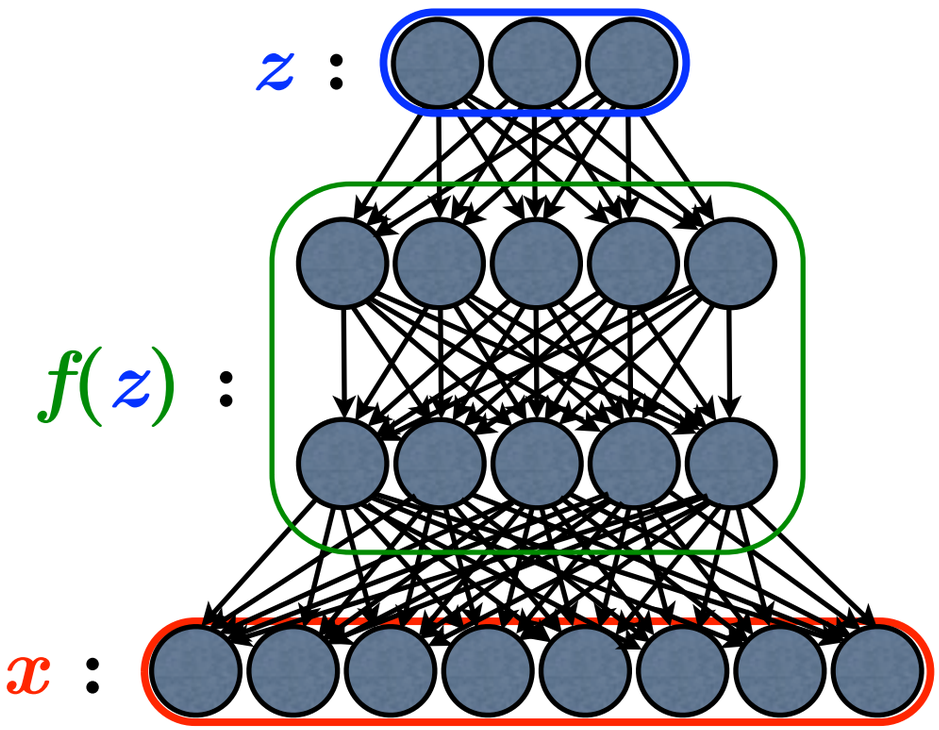
Variational autoencoder (VAE) approach

- Leverage ***neural networks*** to learn a latent variable model.

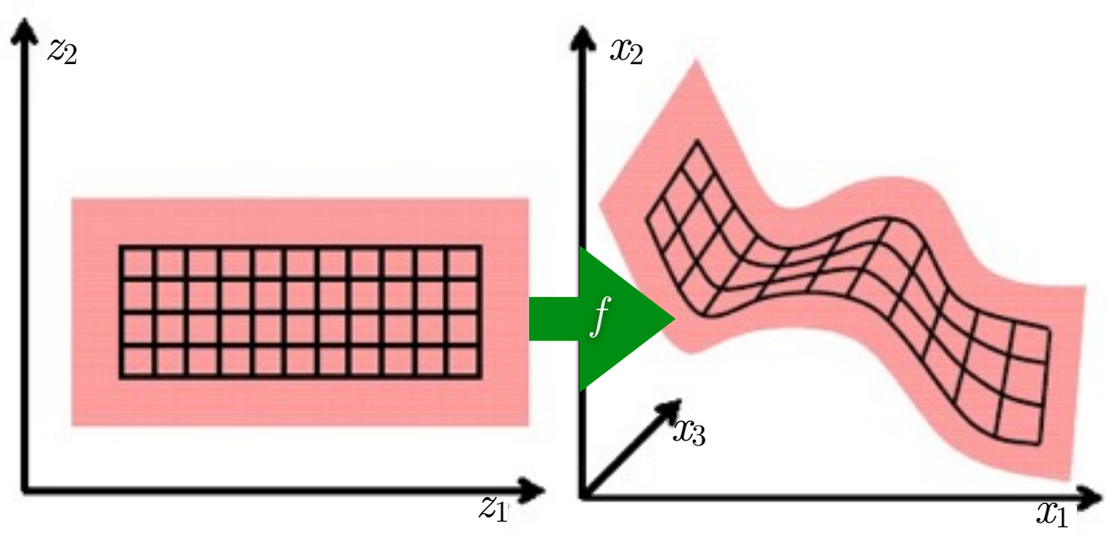
$$p(x) = \int p(x, z)\ dz \quad \text{where} \quad p(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{x} \mid \boldsymbol{z})p(\boldsymbol{z})$$

$$p(\boldsymbol{z}) = \text{something simple} \qquad p(\boldsymbol{x} \mid \boldsymbol{z}) = f(\boldsymbol{z})$$

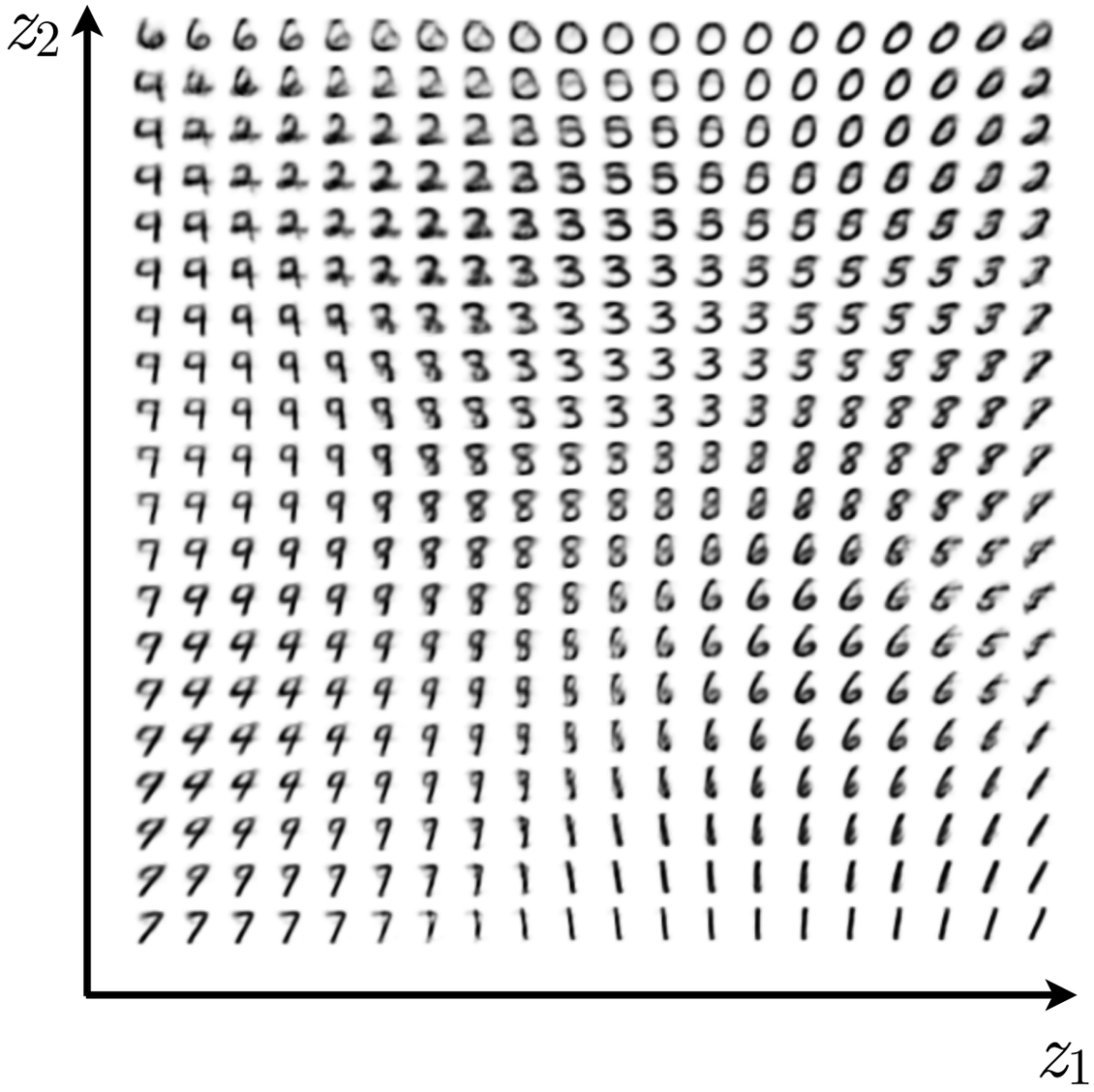




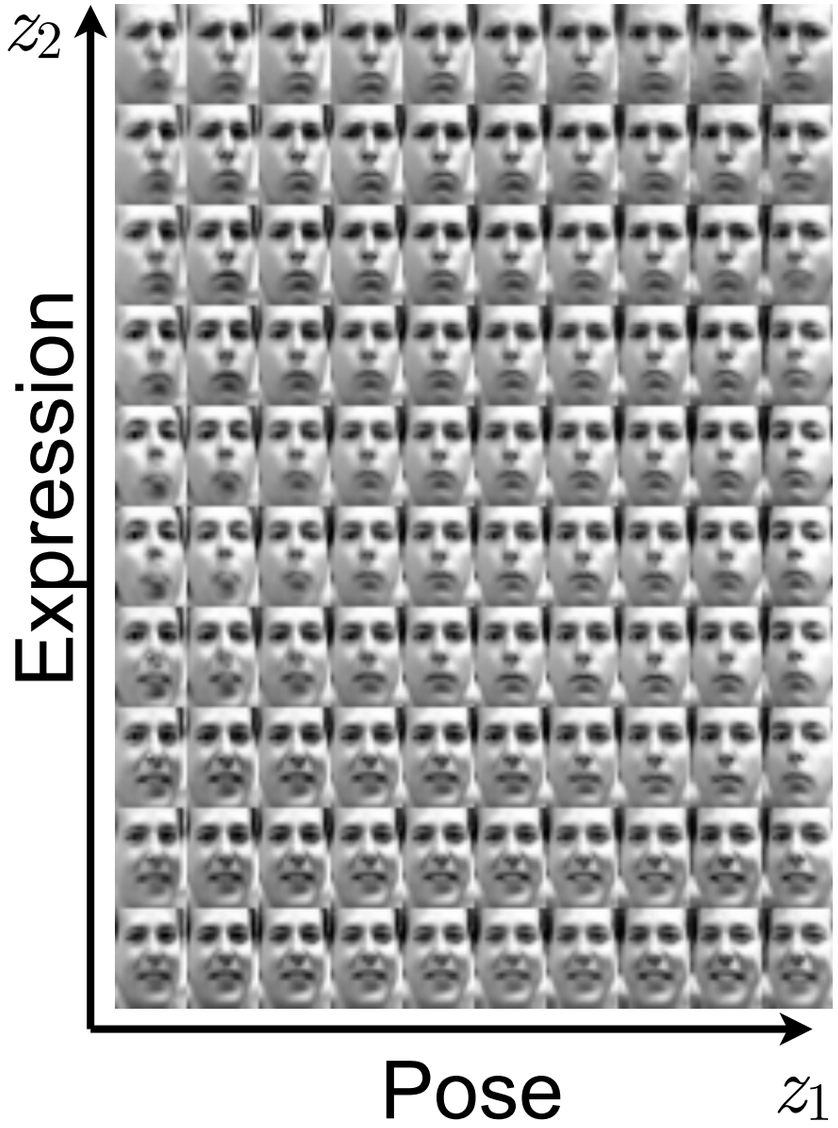
What VAE can do?



MNIST:

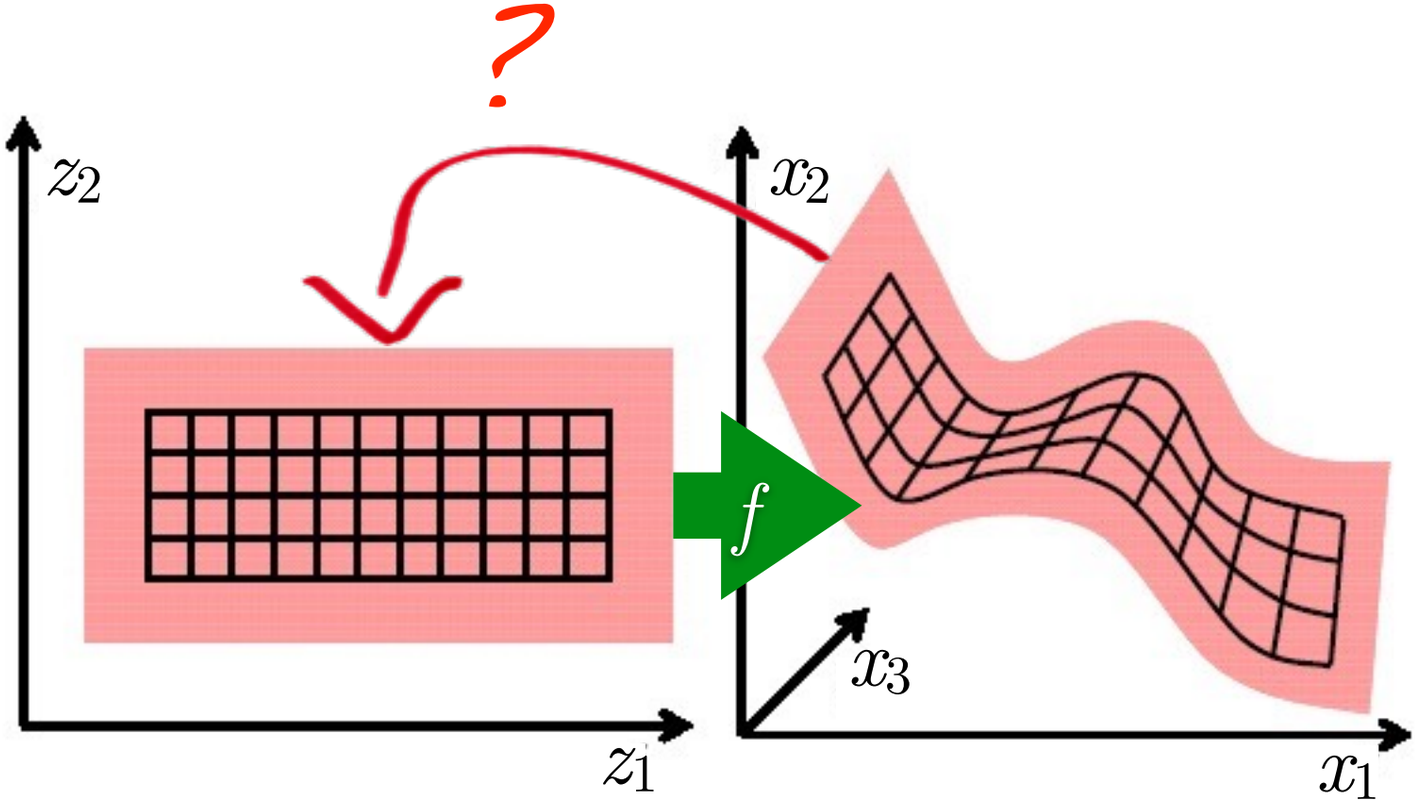


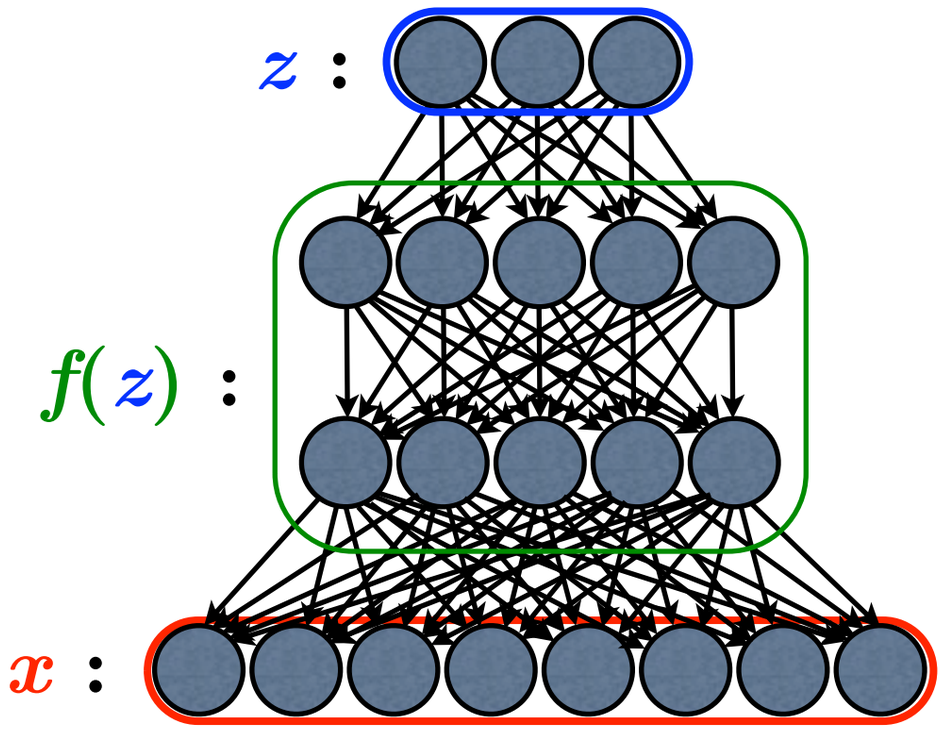
Frey Face dataset:



The inference / learning challenge

- **Where does z come from?** — The classic directed model dilemma.
- Computing the posterior $p(\boldsymbol{z} \mid \boldsymbol{x})$ is intractable.
- We need it to train the directed model.





Variational Autoencoder (VAE)

- Where does z come from? — The classic DAG problem.
- The VAE approach: introduce an inference machine $q_\phi(z \mid x)$ that ***learns*** to approximate the posterior $p_\theta(z \mid x)$.
  - Define a variational lower bound on the data likelihood: $p_\theta(x) \geq \mathcal{L}(\theta, \phi, x)$

$$\begin{aligned}\mathcal{L}(\theta, \phi, x) &= \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x, z) - \log q_\phi(z \mid x)\right] \\ &= \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z) + \log p_\theta(z) - \log q_\phi(z \mid x)\right] \\ &= -D_{\mathrm{KL}}\left(q_\phi(z \mid x) \| \, p_\theta(z)\right) + \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right]\end{aligned}$$

- What is $q_\phi(z \mid x)$?

Variational Autoencoder (VAE)

- Where does z come from? — The classic DAG problem.
- The VAE approach: introduce an inference machine $q_\phi(z \mid x)$ that ***learns*** to approximate the posterior $p_\theta(z \mid x)$.
  - Define a variational lower bound on the data likelihood: $p_\theta(x) \geq \mathcal{L}(\theta, \phi, x)$

$$\begin{aligned}\mathcal{L}(\theta, \phi, x) &= \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x, z) - \log q_\phi(z \mid x)\right] \\ &= \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z) + \log p_\theta(z) - \log q_\phi(z \mid x)\right] \\ &= -D_{\mathrm{KL}}\left(q_\phi(z \mid x) \| p_\theta(z)\right) + \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right]\end{aligned}$$

reconstruction term

- What is $q_\phi(z \mid x)$?

Variational Autoencoder (VAE)

- Where does z come from? — The classic DAG problem.
- The VAE approach: introduce an inference machine $q_\phi(z \mid x)$ that ***learns*** to approximate the posterior $p_\theta(z \mid x)$.
  - Define a variational lower bound on the data likelihood: $p_\theta(x) \geq \mathcal{L}(\theta, \phi, x)$

$$\begin{aligned}\mathcal{L}(\theta, \phi, x) &= \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x, z) - \log q_\phi(z \mid x)\right] \\ &= \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z) + \log p_\theta(z) - \log q_\phi(z \mid x)\right] \\ &= -D_{\mathrm{KL}}\left(q_\phi(z \mid x) \| p_\theta(z)\right) + \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right]\end{aligned}$$

***regularization* term** ***reconstruction* term**

- What is $q_\phi(z \mid x)$?

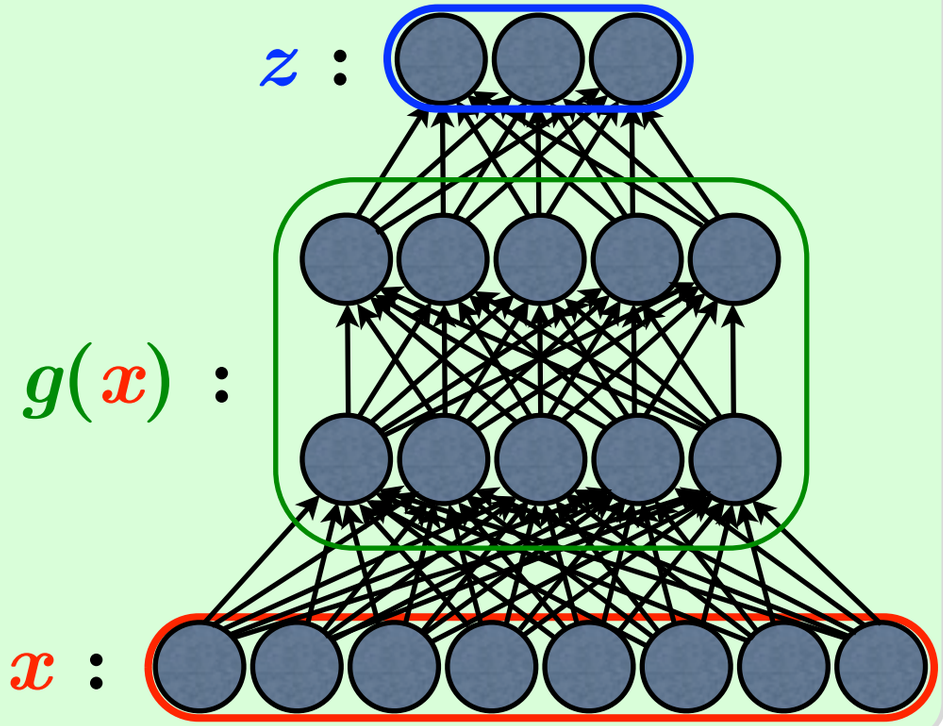
VAE Inference model

- The **VAE approach**: introduce an inference model $q_\phi(z \mid x)$ that **learns** to approximates the intractable posterior $p_\theta(z \mid x)$ by optimizing the variational lower bound:

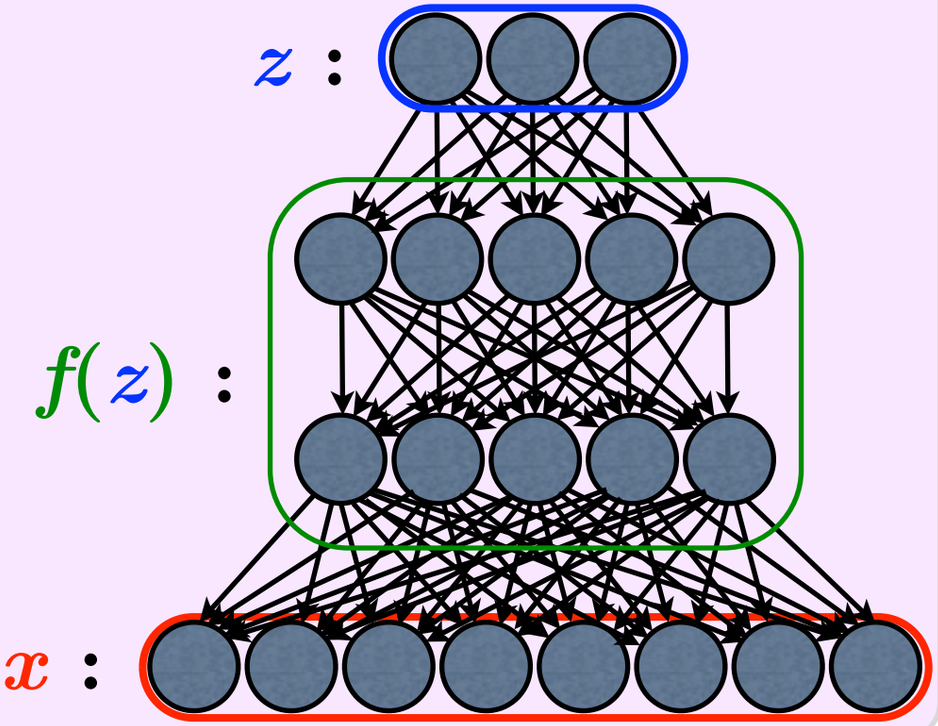
$$\mathcal{L}(\theta, \phi, x) = -D_{\mathrm{KL}}\left(q_\phi(z \mid x) \| p_\theta(z)\right) + \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right]$$

- We parameterize $q_\phi(z \mid x)$ with another neural network:

$q_\phi(z \mid x) = q(z; g(x, \phi))$

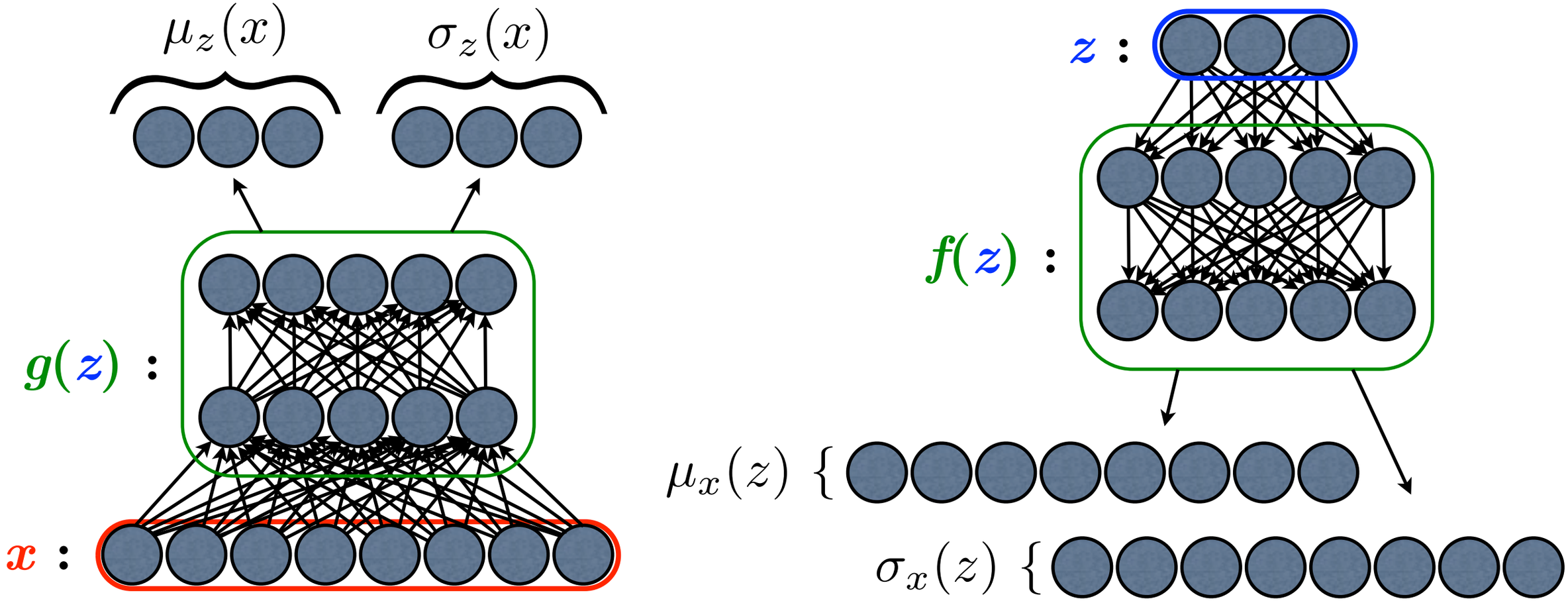


$p_\theta(x \mid z) = p(x; f(z, \theta))$



Reparametrization trick

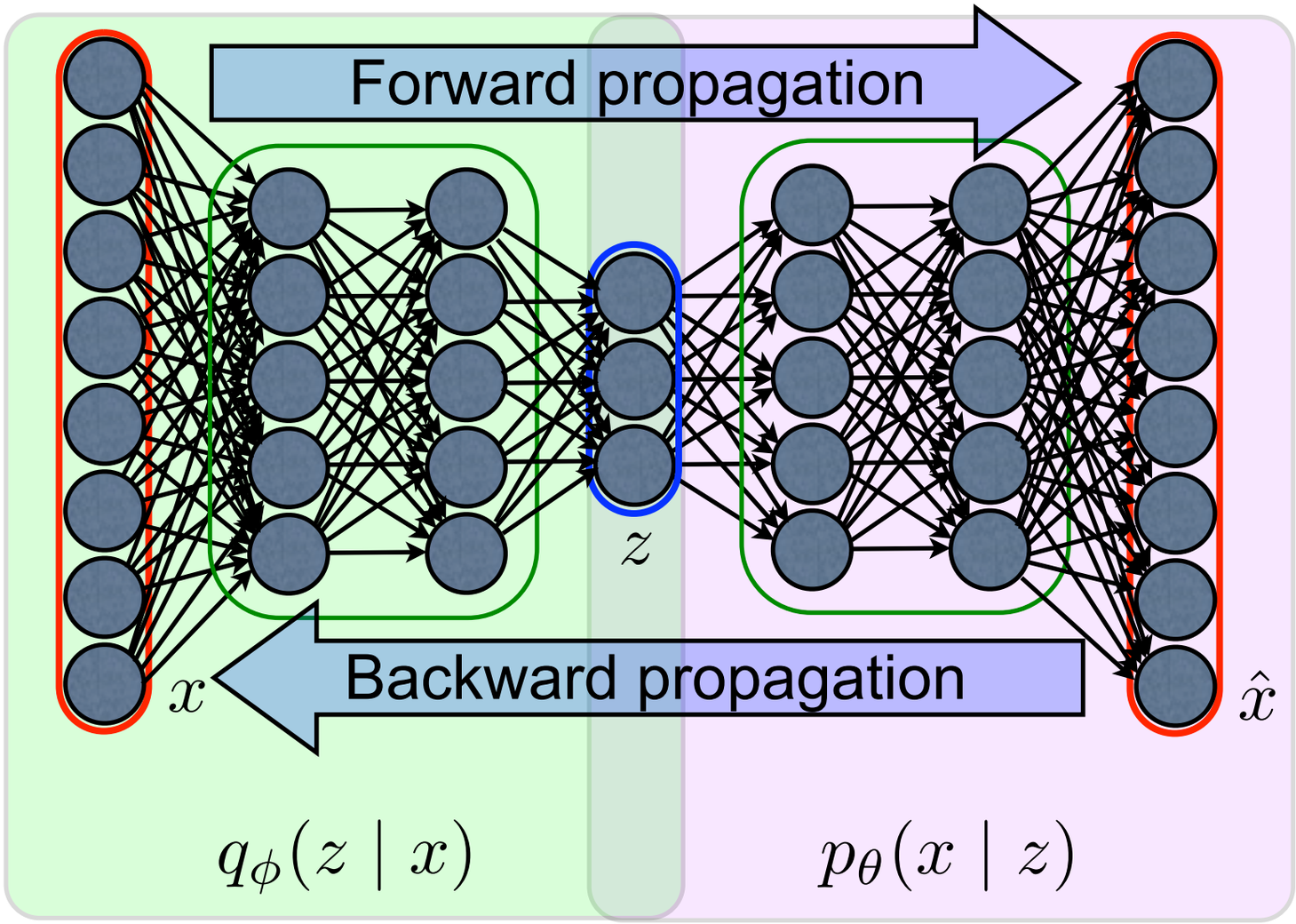
- Adding a few details + one really important trick
- Let's consider z to be real and $q_\phi(z \mid x) = \mathcal{N}(z; \mu_z(x), \sigma_z(x))$
- Parametrize z as $z = \mu_z(x) + \sigma_z(x)\epsilon_z$ where $\epsilon_z = \mathcal{N}(0, 1)$
- (optional) Parametrize x a $x = \mu_x(z) + \sigma_x(z)\epsilon_x$ where $\epsilon_x = \mathcal{N}(0, 1)$



Training with backpropagation!

- Due to a **reparametrization** trick, we can simultaneously train both the **generative model** $p_\theta(x \mid z)$ and the **inference model** $q_\phi(z \mid x)$ by optimizing the variational bound using gradient **backpropagation**.

Objective function: $\mathcal{L}(\theta, \phi, x) = -D_{\mathrm{KL}}\left(q_\phi(z \mid x) \| p_\theta(z)\right) + \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right]$



Relative performance of VAE

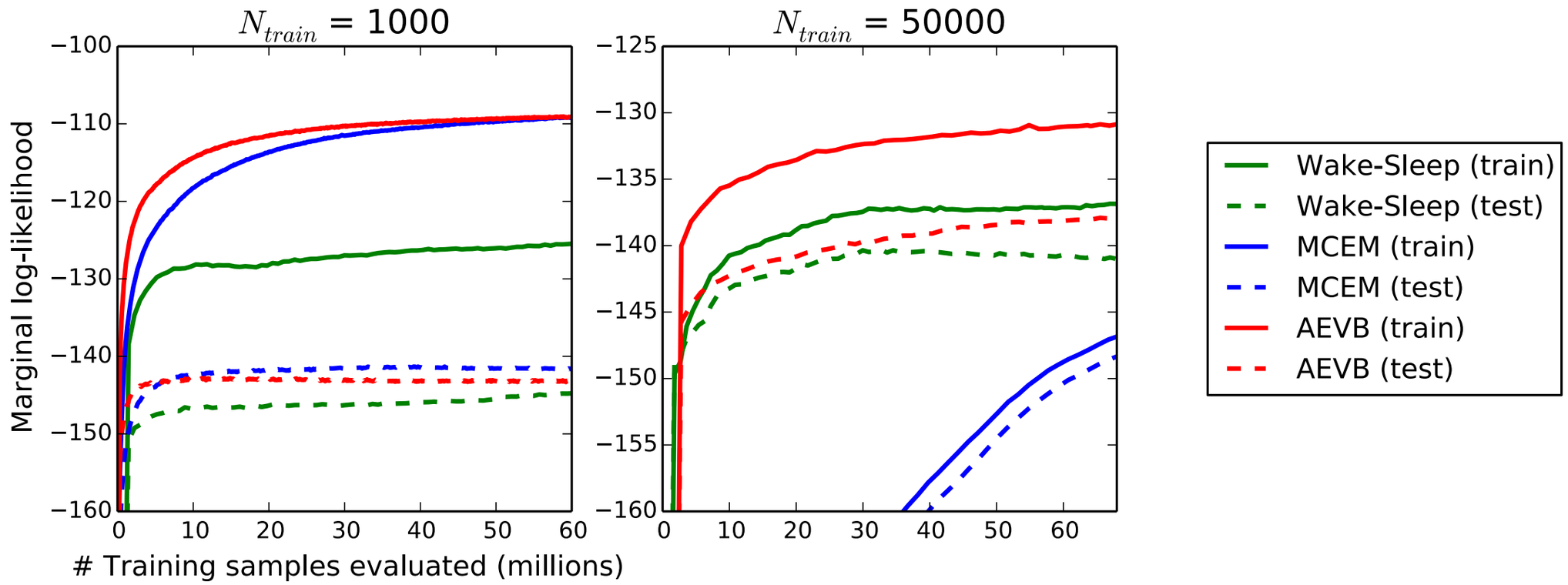


Figure 3: Comparison of AEVB to the wake-sleep algorithm and Monte Carlo EM, in terms of the estimated marginal likelihood, for a different number of training points. Monte Carlo EM is not an on-line algorithm, and (unlike AEVB and the wake-sleep method) can't be applied efficiently for the full MNIST dataset.

Note: **MCEM** is Expectation Maximization, where $p(z \mid x)$ is sampled using Hybrid (Hamiltonian) Monte Carlo

For more see: **Markov Chain Monte Carlo and Variational Inference: Bridging the Gap,**
Tim Salimans, Diederik P. Kingma, Max Welling

Figure from Diederik P. Kingma & Max Welling

Effect of KL term: component collapse

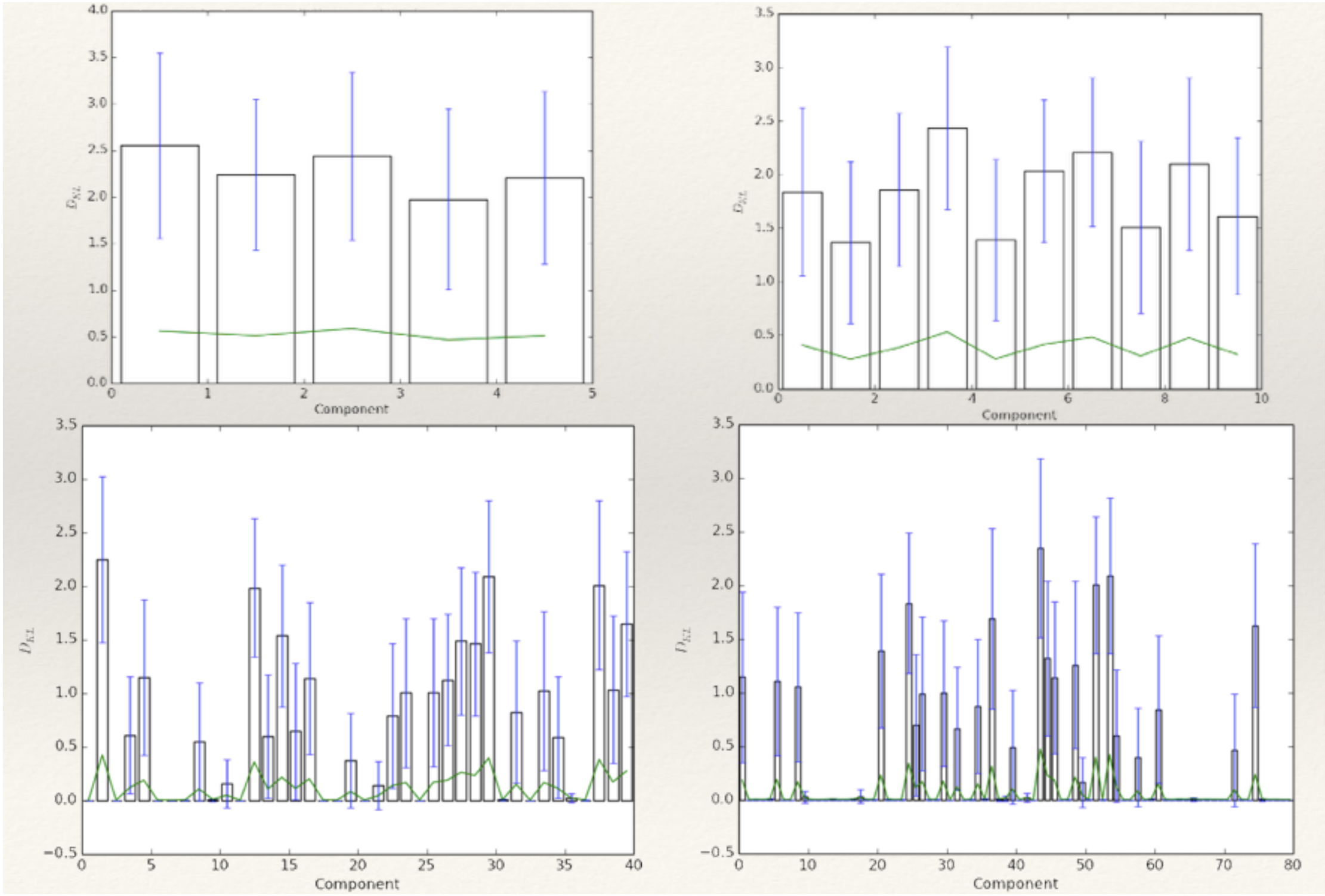
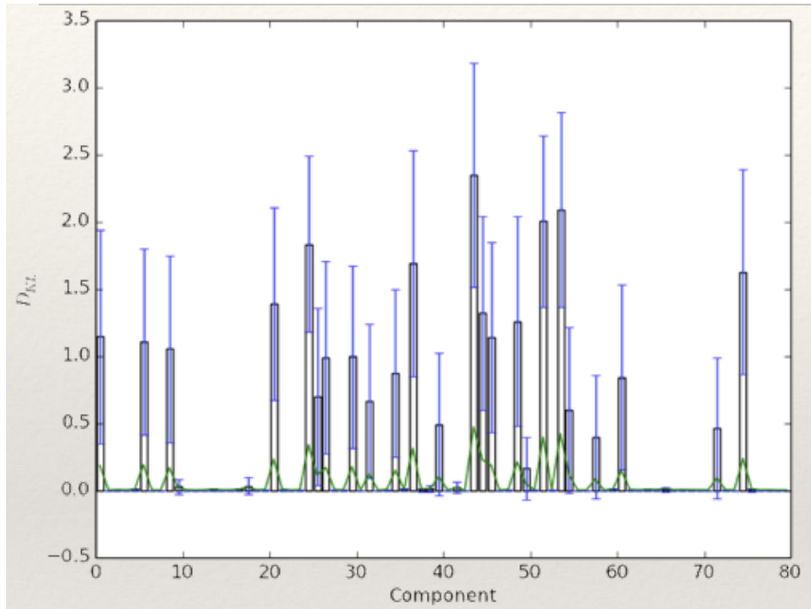


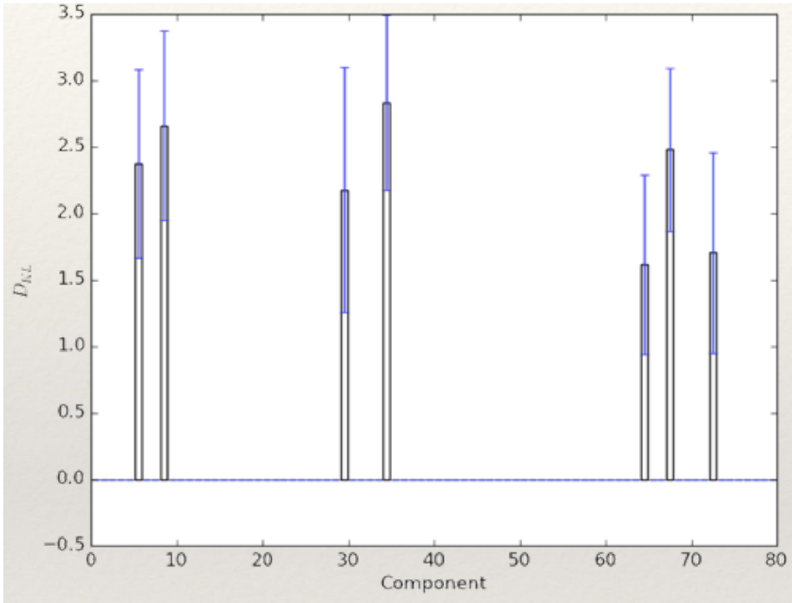
Figure from Laurent Dinh & Vincent Dumoulin

Component collapse & depth

Deep model:
some component collapse



Deeper model:
more component collapse



Figures from Laurent Dinh & Vincent Dumoulin

Component collapse & decoder weights

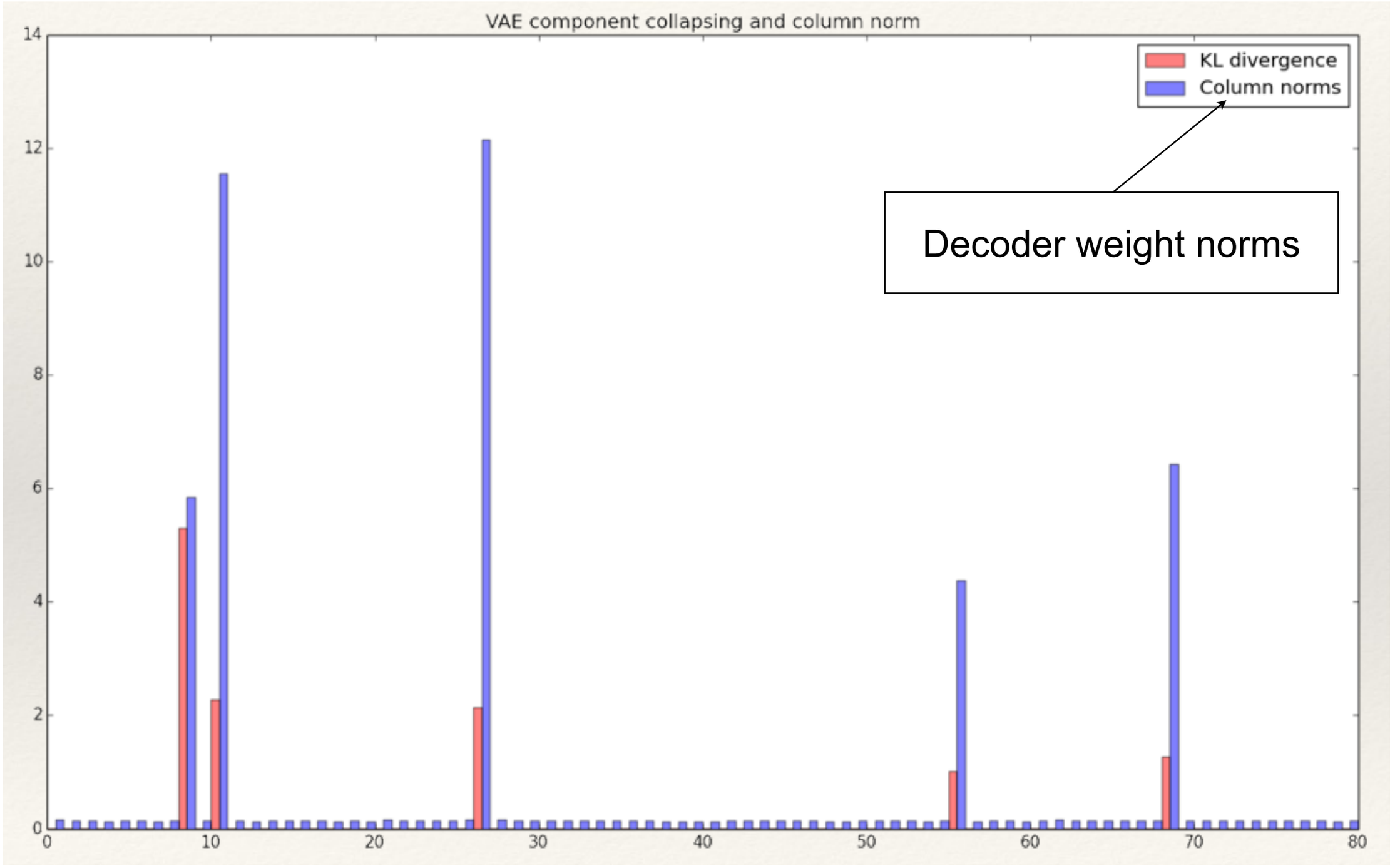


Figure from Laurent Dinh & Vincent Dumoulin

Component collapse & learned variance

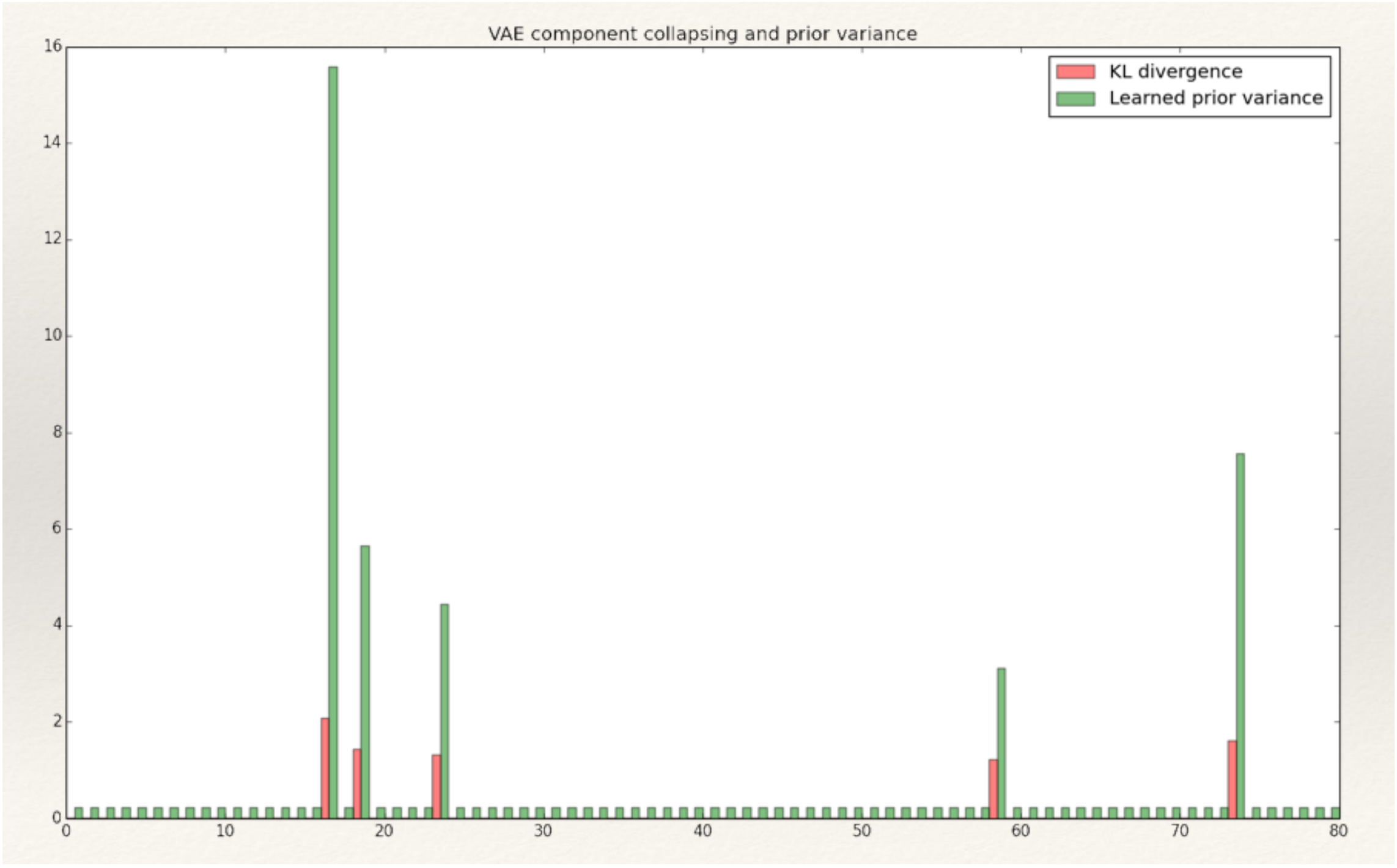


Figure from Laurent Dinh & Vincent Dumoulin

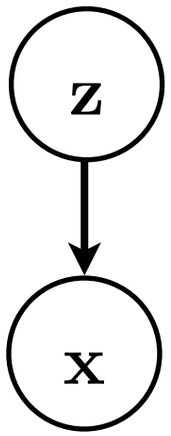
Semi-supervised Learning with Deep Generative Models

Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, Max Welling

They study two basic approaches:

- **M1**: Standard unsupervised feature learning ("self-taught learning")
  - Train features z on unlabeled data, train a classifier to map from z to label y.
  - Generative model: (recall that $\mathbf{x} = \text{data}$, $\mathbf{z} = \text{latent features}$)

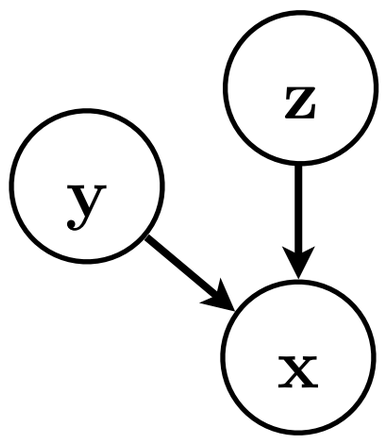
$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I}); \qquad p_\theta(\mathbf{x}|\mathbf{z}) = f(\mathbf{x}; \mathbf{z}, \boldsymbol{\theta}),$$



- **M2**: Generative semi-supervised model.

$$p(y) = \text{Cat}(y|\boldsymbol{\pi}); \qquad p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I});$$

$$p_\theta(\mathbf{x}|y, \mathbf{z}) = f(\mathbf{x}; y, \mathbf{z}, \boldsymbol{\theta}),$$

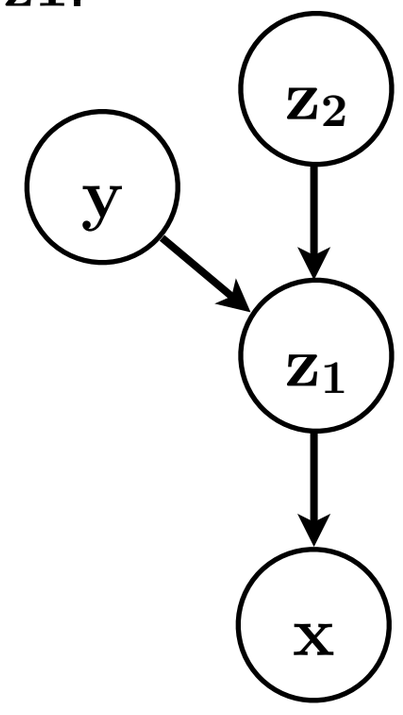


Semi-supervised Learning with Deep Generative Models

Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, Max Welling

- **M1+M2**: Combination semi-supervised model
  - Train generative semi-supervised model on unsupervised features $\mathbf{z1}$ on unlabeled data, train a classifier to map from $\mathbf{z1}$ to label $\mathbf{z1}$.

$$p_\theta(\mathbf{x}, y, \mathbf{z}_1, \mathbf{z}_2) = p(y)p(\mathbf{z}_2)p_\theta(\mathbf{z}_1|y, \mathbf{z}_2)p_\theta(\mathbf{x}|\mathbf{z}_1),$$



Semi-supervised Learning with Deep Generative Models

Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, Max Welling

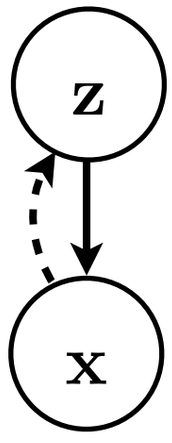
- Appoximate posterior (encoder model)
  - Following the VAE strategy we parametrize the approximate posterior with a high capacity model, like a MLP or some other deep model (convnet, RNN, etc).

$$\text{M1: } q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_\phi(\mathbf{x}), \text{diag}(\boldsymbol{\sigma}^2_\phi(\mathbf{x}))),$$

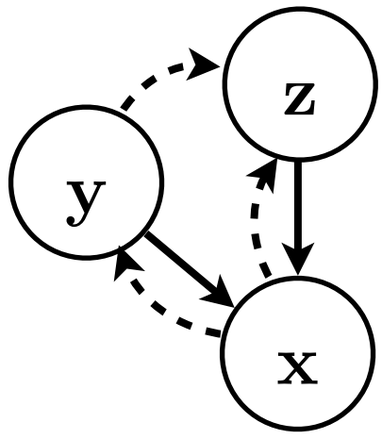
$$\text{M2: } q_\phi(\mathbf{z}|y,\mathbf{x}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_\phi(y,\mathbf{x}), \text{diag}(\boldsymbol{\sigma}^2_\phi(\mathbf{x}))); \;\; q_\phi(y|\mathbf{x}) = \text{Cat}(y|\boldsymbol{\pi}_\phi(\mathbf{x})),$$

  - $\boldsymbol{\mu}_\phi(\mathbf{x})$ and $\boldsymbol{\sigma}^2_\phi(\mathbf{x})$ are parameterized by deep MLPs, that can share parameters.

M1:



M2:



Semi-supervised Learning with Deep Generative Models

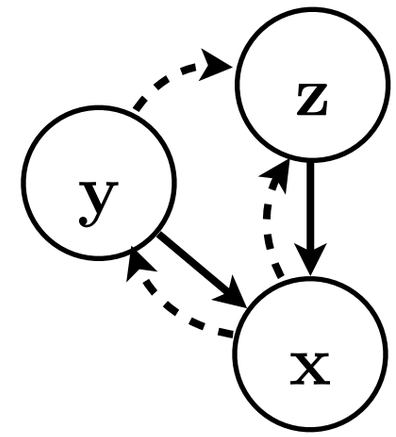
Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, Max Welling

- **M2**: The lower bound for the generative semi-supervised model.
  - Objective with labeled data:

$$\log p_\theta(\mathbf{x}, y) \geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},y)}\left[\log p_\theta(\mathbf{x}|y,\mathbf{z}) + \log p_\theta(y) + \log p(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x},y)\right] = -\mathcal{L}(\mathbf{x},y),$$

  - Objective without labels:

$$\log p_\theta(\mathbf{x}) \geq \mathbb{E}_{q_\phi(y,\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|y,\mathbf{z}) + \log p_\theta(y) + \log p(\mathbf{z}) - \log q_\phi(y,\mathbf{z}|\mathbf{x})\right]$$
$$= \sum_y q_\phi(y|\mathbf{x})(-\mathcal{L}(\mathbf{x},y)) + \mathcal{H}(q_\phi(y|\mathbf{x})) = -\mathcal{U}(\mathbf{x}).$$



  - Semi-supervised objective:

$$\mathcal{J} = \sum_{(\mathbf{x},y)\sim\widetilde{p}_l} \mathcal{L}(\mathbf{x},y) + \sum_{\mathbf{x}\sim\widetilde{p}_u} \mathcal{U}(\mathbf{x})$$

  - actually, for classification, they use $\mathcal{J}^\alpha = \mathcal{J} + \alpha \cdot \mathbb{E}_{\widetilde{p}_l(\mathbf{x},y)}\left[-\log q_\phi(y|\mathbf{x})\right],$

Semi-supervised MNIST classification results

Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, Max Welling

- Combination model M1+M2 shows dramatic improvement:

Table 1: Benchmark results of semi-supervised classification on MNIST with few labels.

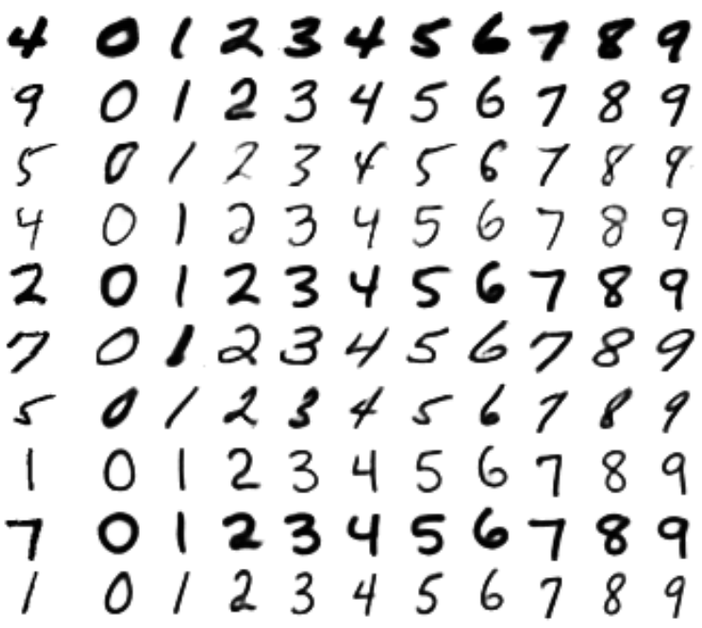
N	NN	CNN	TSVM	CAE	MTC	AtlasRBF	M1+TSVM	M2	M1+M2
100	25.81	22.98	16.81	13.47	12.03	8.10 ($\pm$ 0.95)	11.82 ($\pm$ 0.25)	11.97 ($\pm$ 1.71)	**3.33** ($\pm$ 0.14)
600	11.44	7.68	6.16	6.3	5.13	–	5.72 ($\pm$ 0.049)	4.94 ($\pm$ 0.13)	**2.59** ($\pm$ 0.05)
1000	10.7	6.45	5.38	4.77	3.64	3.68 ($\pm$ 0.12)	4.24 ($\pm$ 0.07)	3.60 ($\pm$ 0.56)	**2.40** ($\pm$ 0.02)
3000	6.04	3.35	3.45	3.22	2.57	–	3.49 ($\pm$ 0.04)	3.92 ($\pm$ 0.63)	**2.18** ($\pm$ 0.04)

- Full MNIST test error: 0.96% (for comparison, current SOTA: 0.78%).

Conditional generation using M2

Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, Max Welling

(a) Handwriting styles for MNIST obtained by fixing the class label and varying the 2D latent variable $\mathbf{z}$



(b) MNIST analogies

(c) SVHN analogies

Figure 1: **(a)** Visualisation of handwriting styles learned by the model with 2D $\mathbf{z}$-space. **(b,c)** Analogical reasoning with generative semi-supervised models using a high-dimensional $\mathbf{z}$-space. The leftmost columns show images from the test set. The other columns show analogical fantasies of $\mathbf{x}$ by the generative model, where the latent variable $\mathbf{z}$ of each row is set to the value inferred from the test-set image on the left by the inference network. Each column corresponds to a class label $\mathbf{y}$.