Variational Autoencoder and Extensions

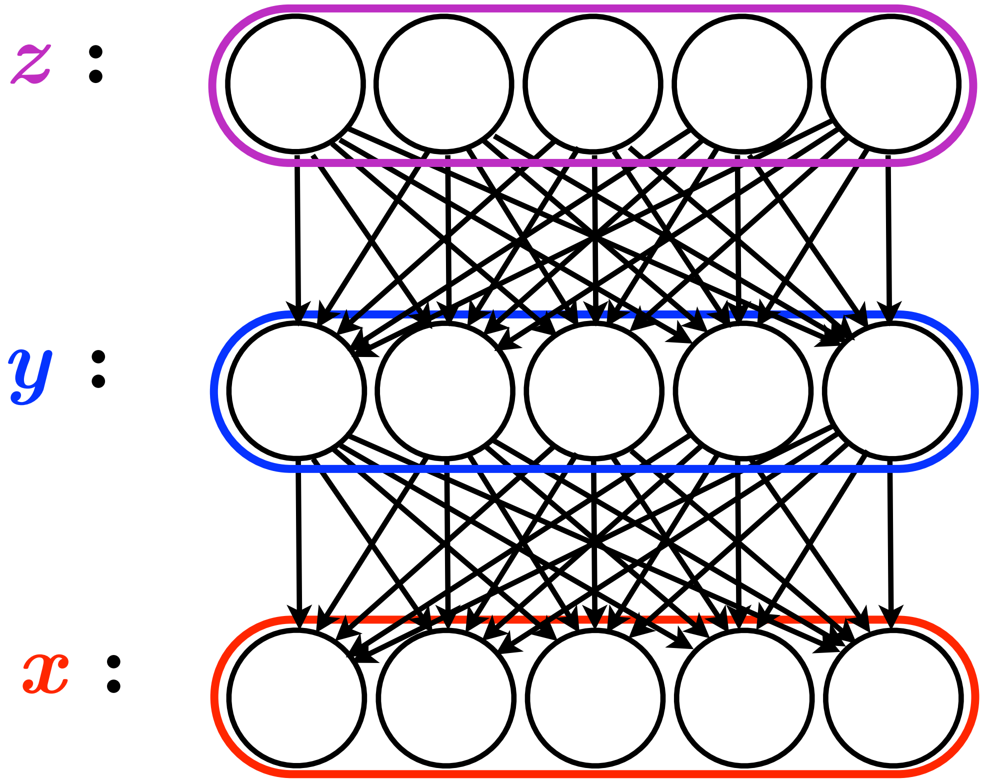
Aaron Courville

Outline

- Variational autoencoder (VAE)
- Semi-supervised learning with the VAE
- Sequential application of VAE: the VRNN
- DRAW model
- Incorporating normalizing flows
- Incorporating MCMC in the VAE inference

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*. ICML 2014.
- 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 $\boldsymbol{z}$ to a complicated distribution on $\boldsymbol{x}$.

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

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

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

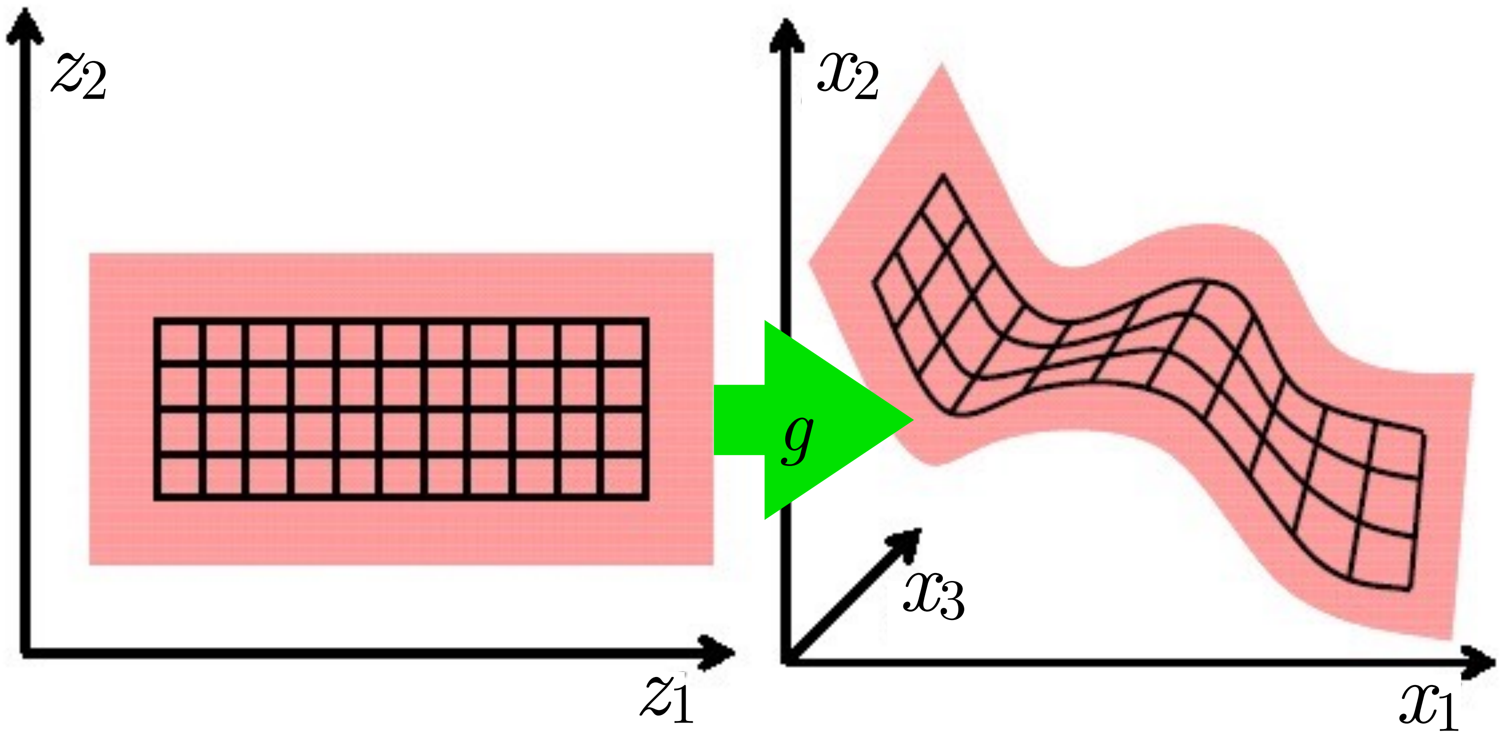


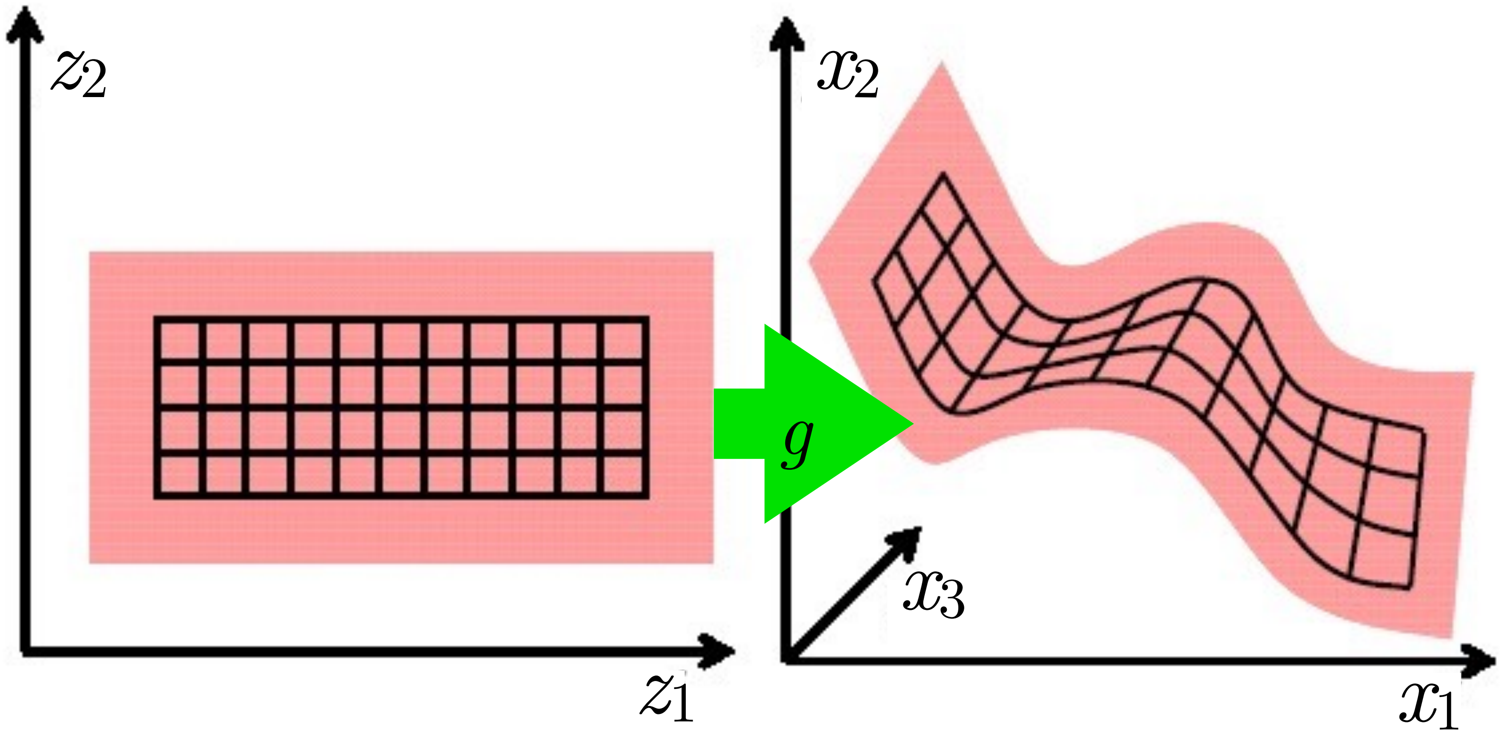
Image from: Ward, A. D., Hamarneh, G.: **3D Surface Parameterization Using Manifold Learning for Medial Shape Representation**, *Conference on Image Processing, Proc. of SPIE Medical Imaging*, 2007

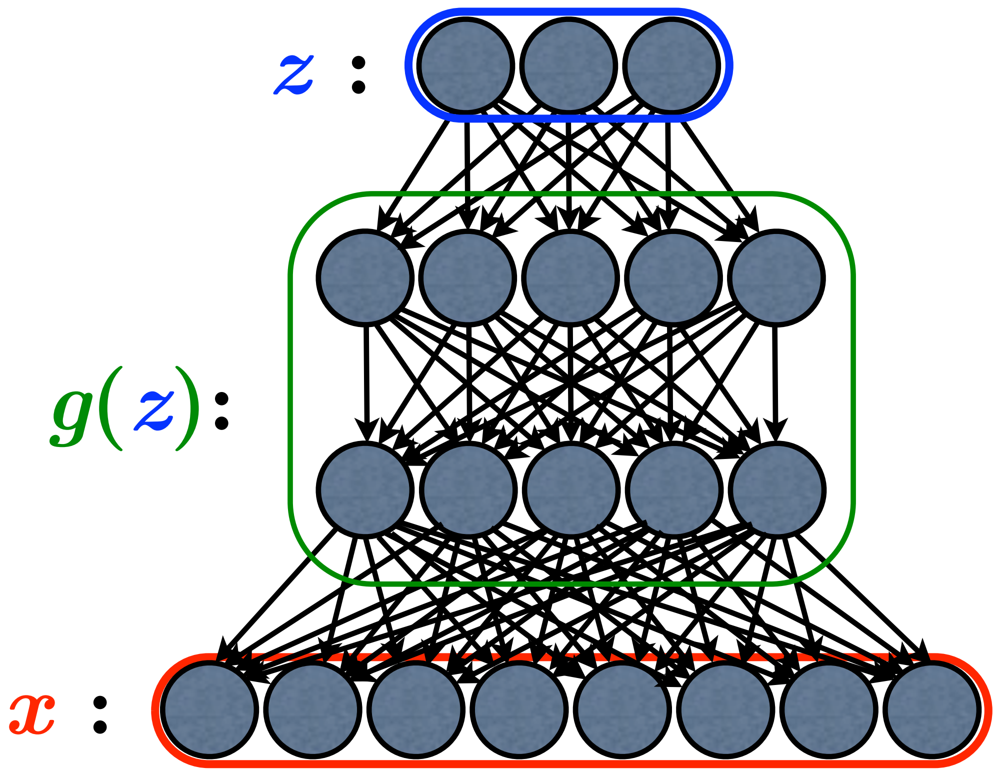
Variational autoencoder (VAE) approach

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

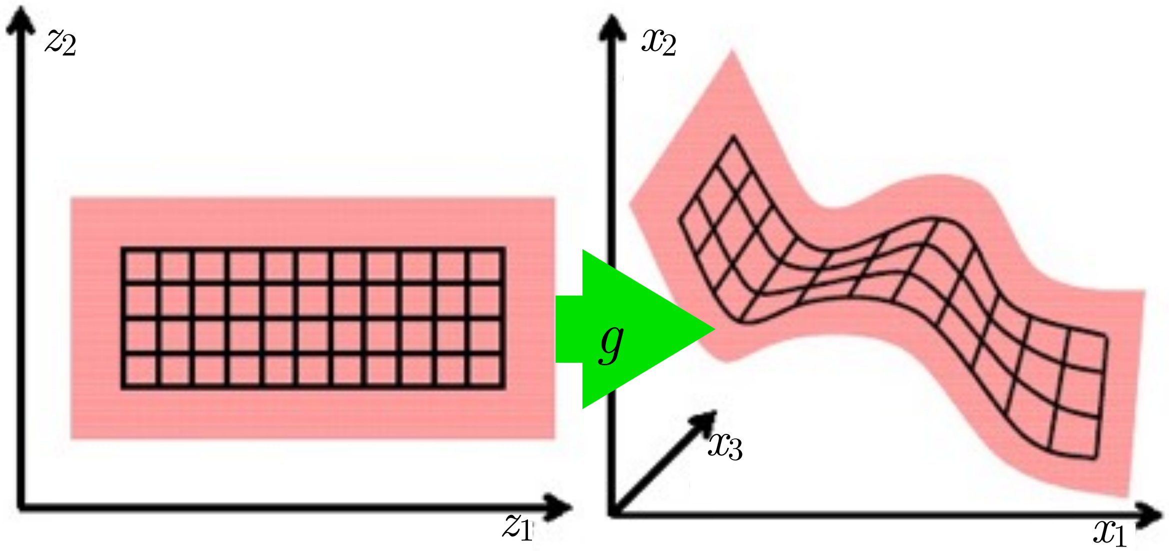
$$p(x) = \int p(x,z)\; dz \quad \text{where} \quad p(\boldsymbol{x},\boldsymbol{z}) = p(\boldsymbol{x} \mid \boldsymbol{z})p(\boldsymbol{z})$$

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

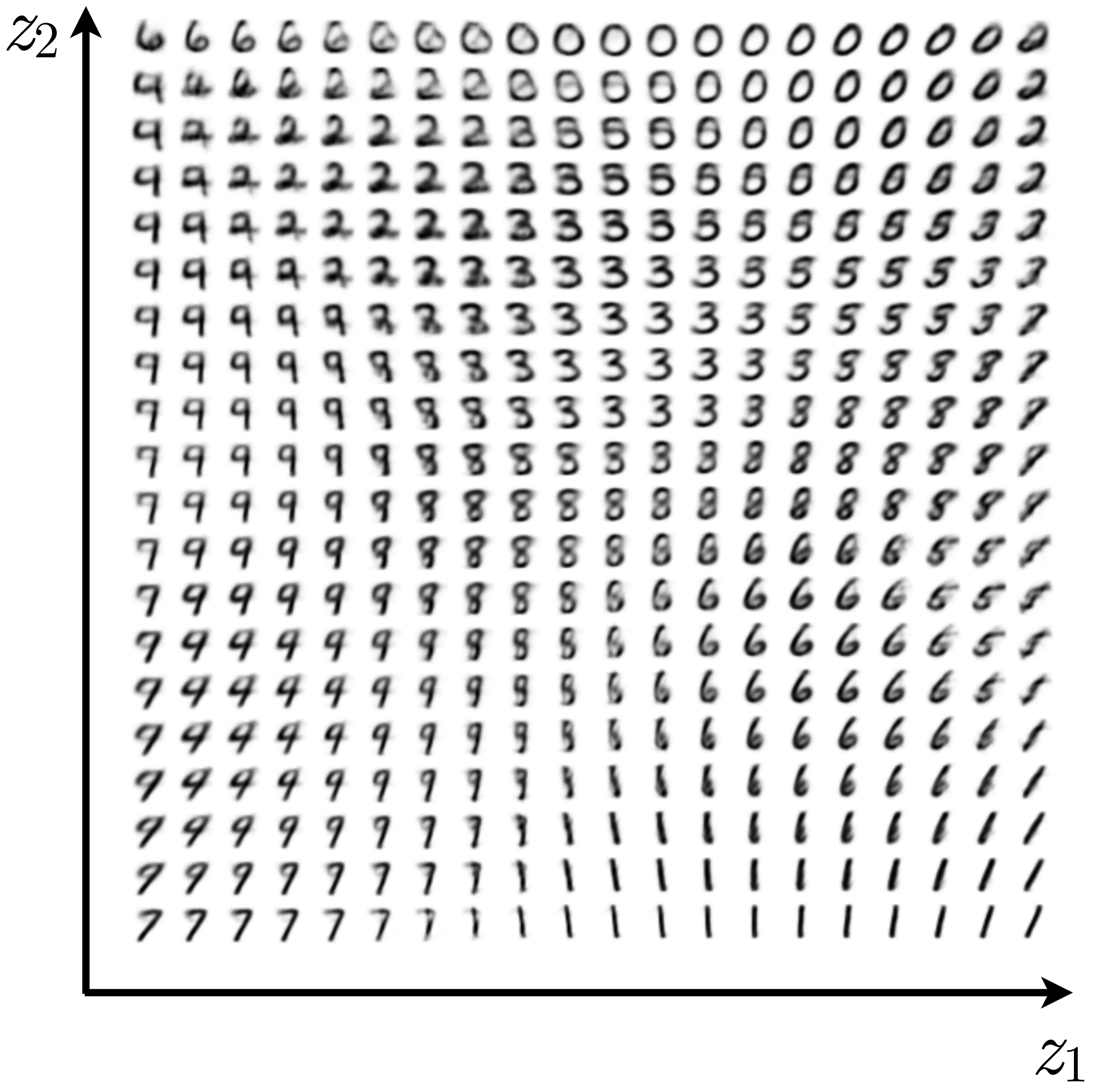




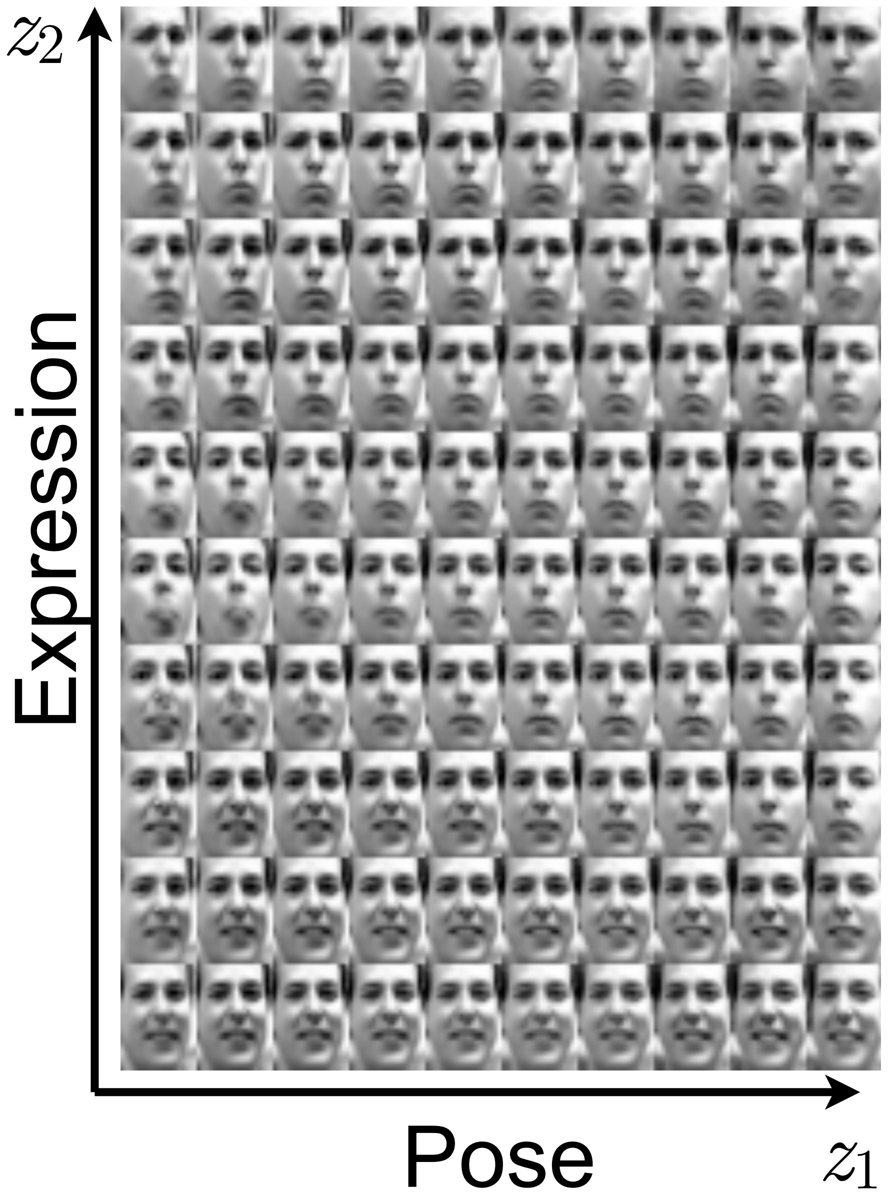
What VAE can do?



MNIST:

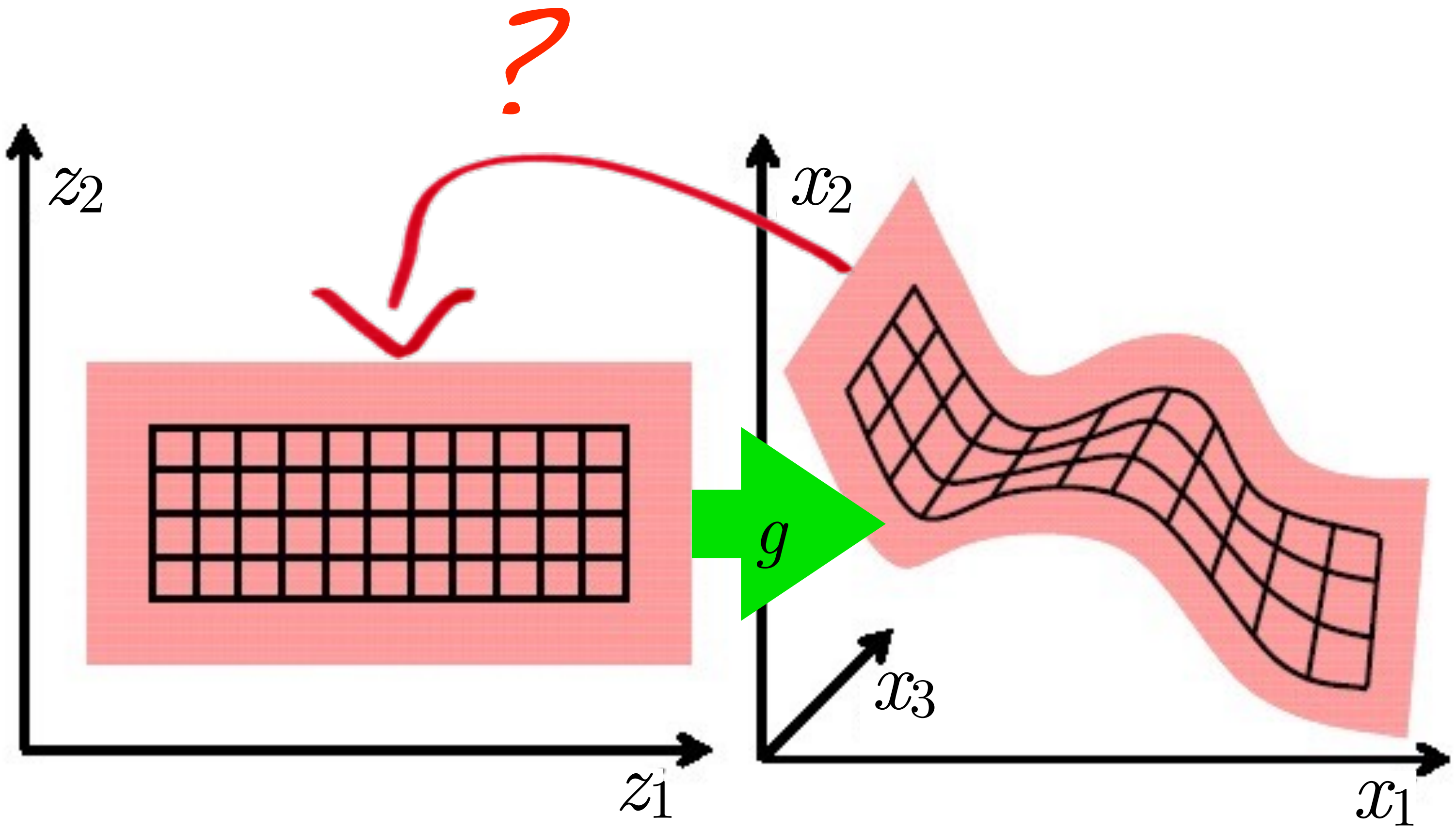


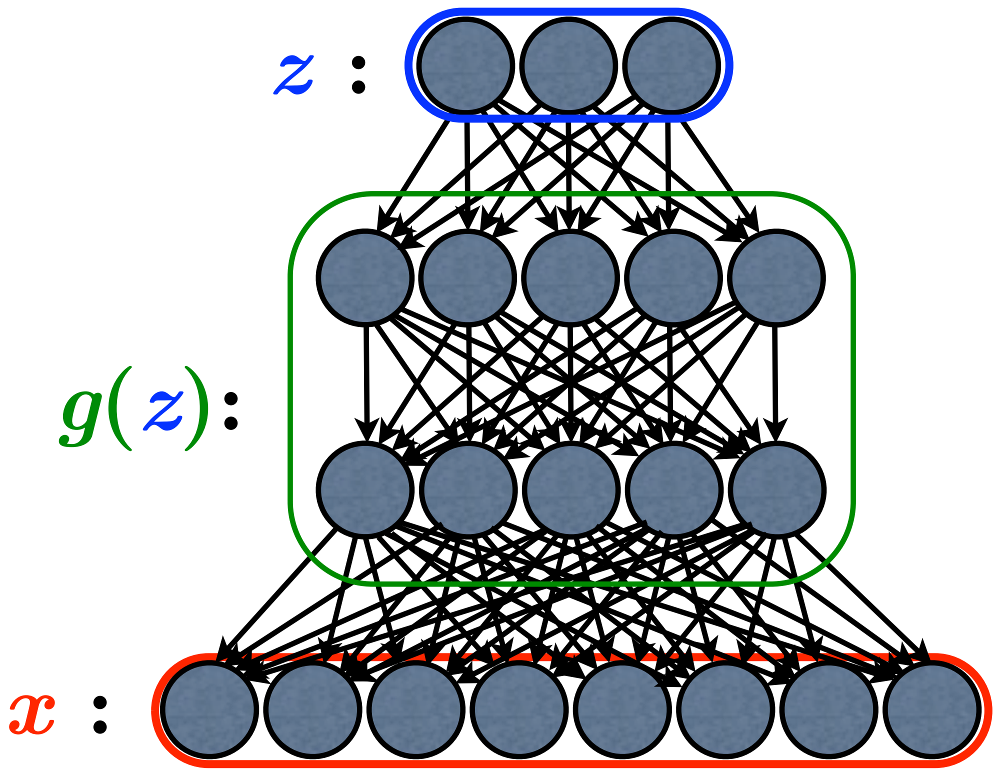
Frey Face dataset:



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: $p_\theta(x) \geq \mathcal{L}(\theta, \phi, x)$

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

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

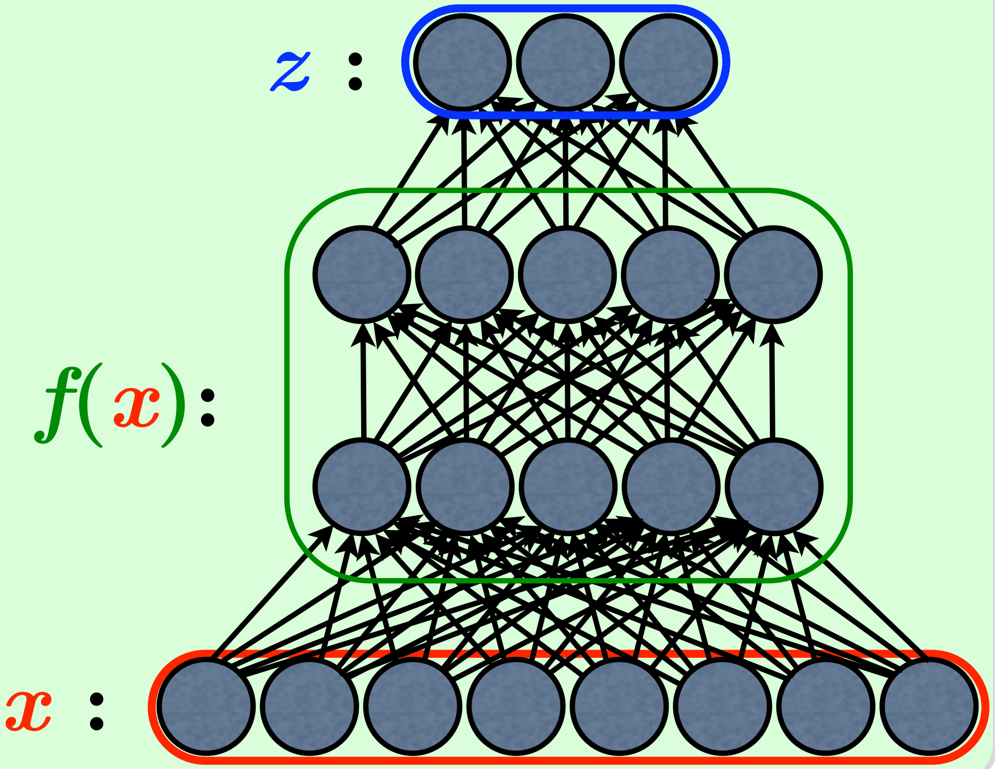
VAE Inference model

- 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:

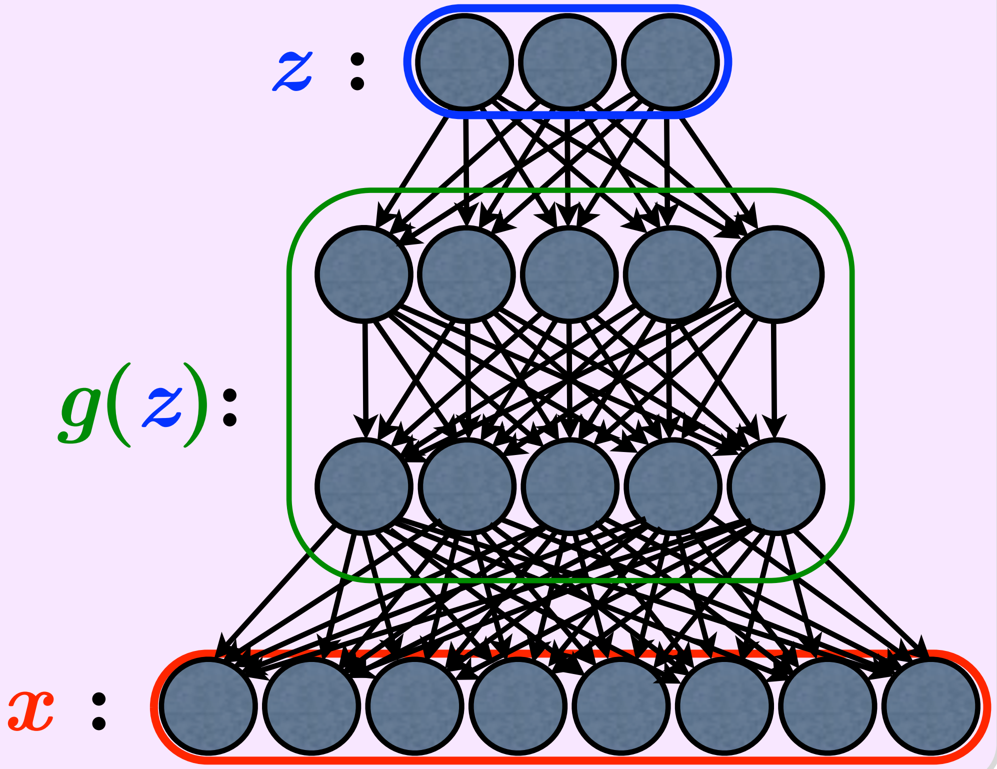
$$\mathcal{L}(\theta, \phi, x) = -D_{\mathrm{KL}}\left(q_\phi(z \mid x) \| \, p_\theta(z)\right) + \mathbb{E}_{q_\phi(z \mid x)}\left[\log p_\theta(x \mid z)\right]$$

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

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

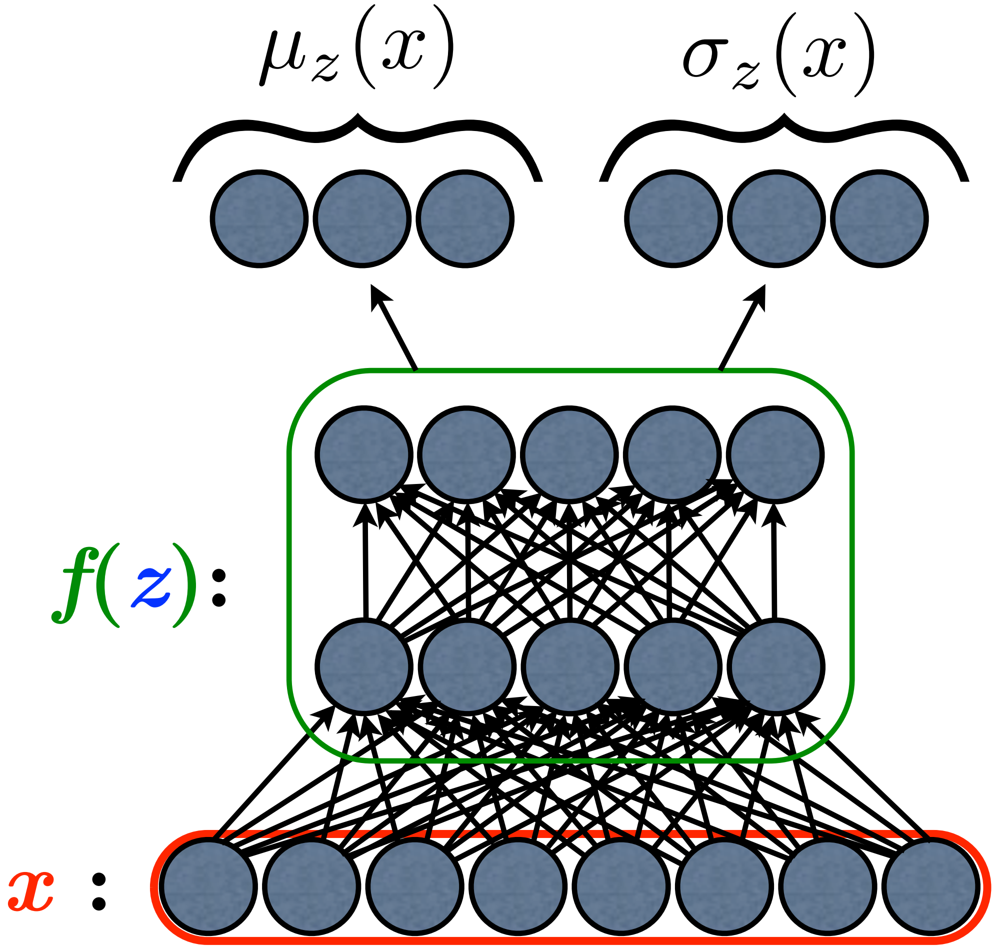


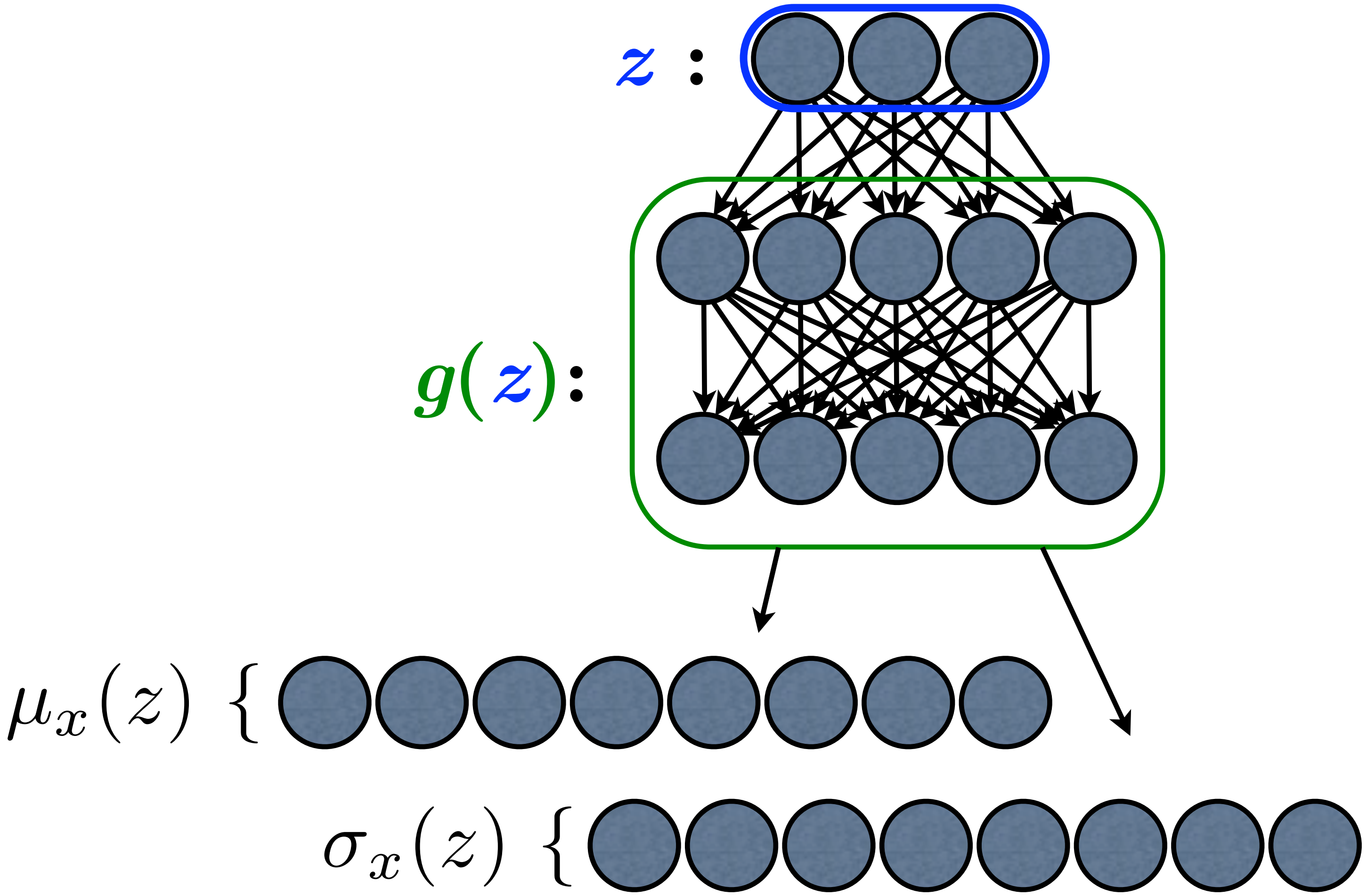
$$p_\theta(x \mid z) = p(x; g(z, \theta))$$



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 $\epsilon_x = \mathcal{N}(0, 1)$

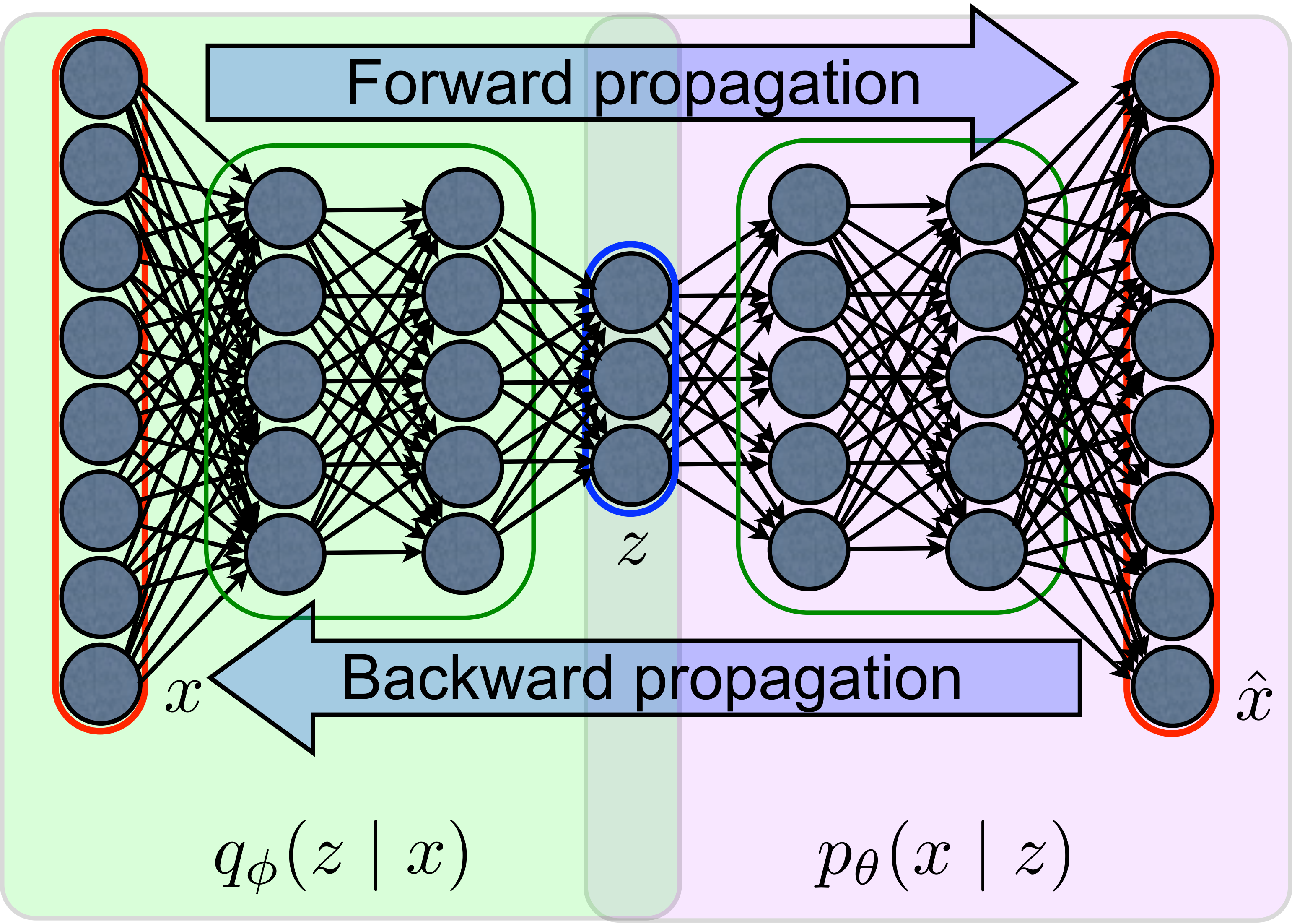




Training with backpropagation!

- 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

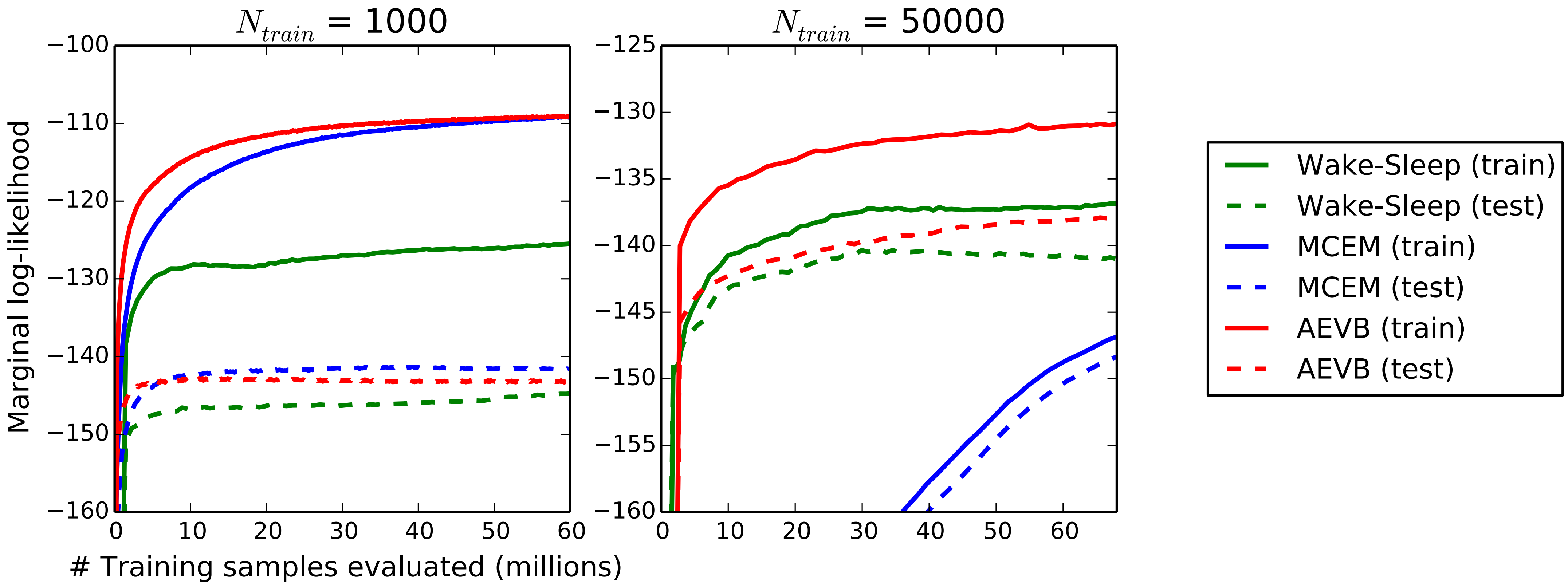


Figure 3: Comparison of AEVB to the wake-sleep algorithm and Monte Carlo EM, in terms of the estimated marginal likelihood, for a different number of training points. Monte Carlo EM is not an on-line algorithm, and (unlike AEVB and the wake-sleep method) can't be applied efficiently for the full MNIST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

Note: **MCEM** is Expectation Maximization, where $p(z|x)$ 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

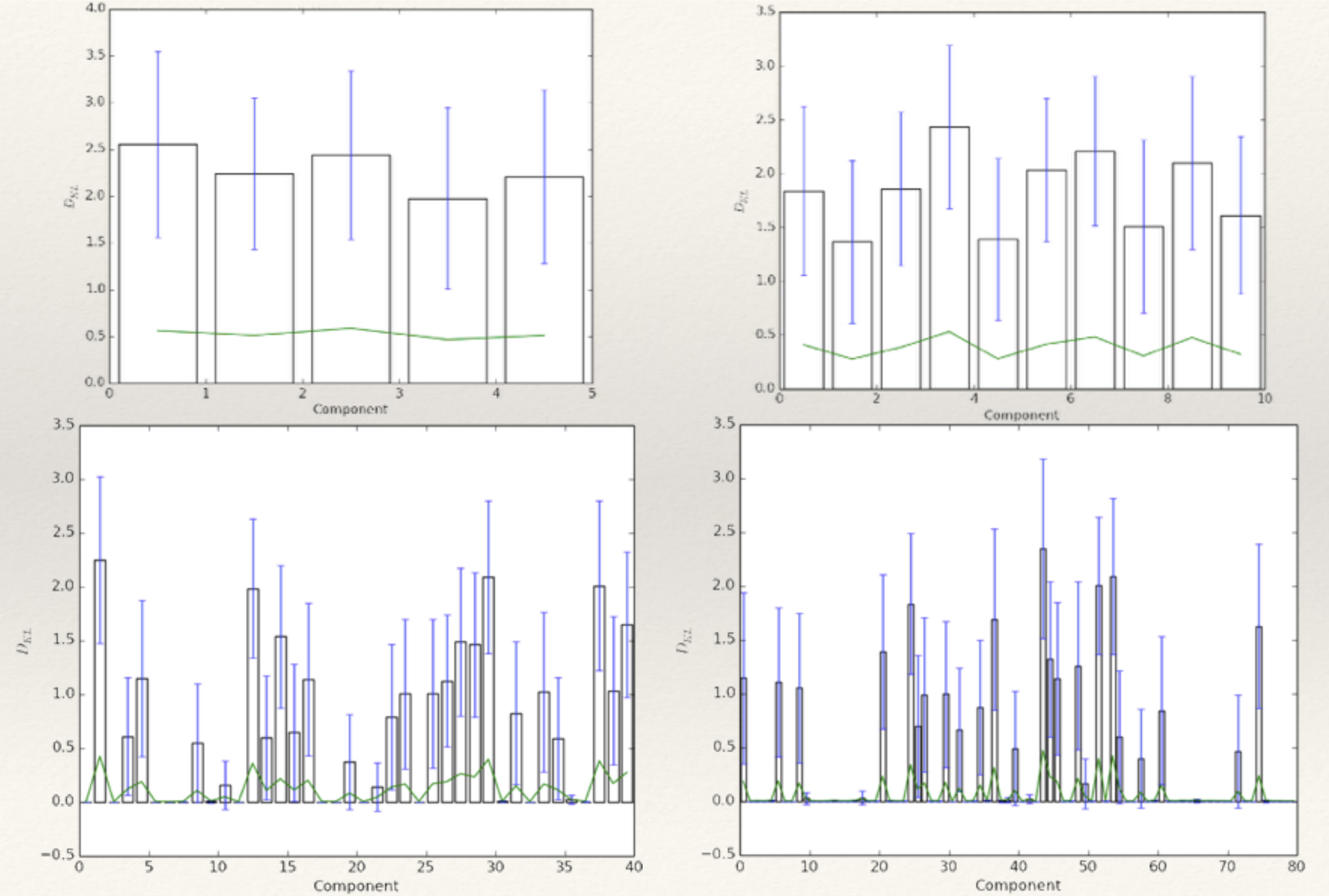
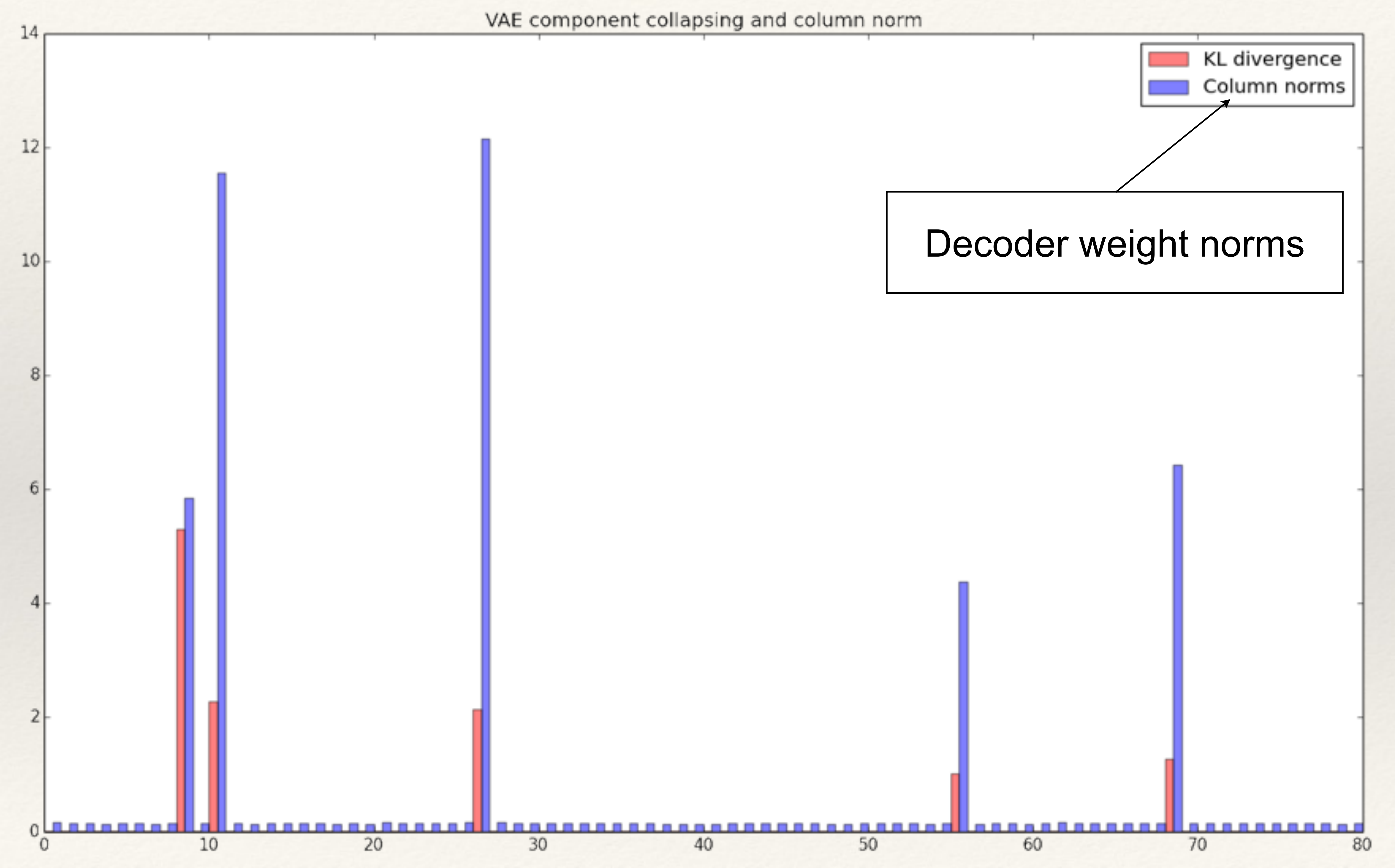


Figure from Laurent Dinh & Vincent Dumoulin

Component collapse & decoder weights



Decoder weight norms

Figure from Laurent Dinh & Vincent Dumoulin

Semi-supervised Learning with Deep Generative Models

Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, Max Welling (NIPS 2014)

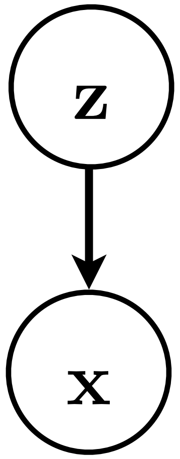
Semi-supervised Learning with Deep Generative Models

Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, Max Welling (NIPS 2014)

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 $\mathbf{x} = \text{data}$, $\mathbf{z} = \text{latent features}$)

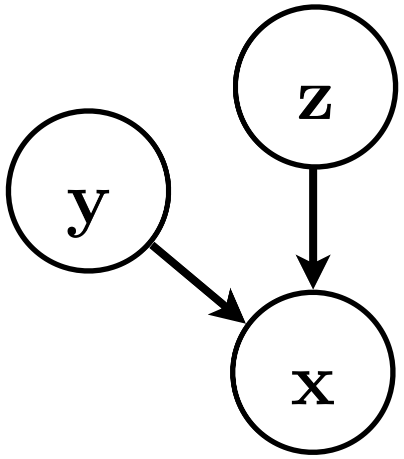
$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I}); \qquad p_\theta(\mathbf{x}|\mathbf{z}) = f(\mathbf{x}; \mathbf{z}, \boldsymbol{\theta}),$$



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

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

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

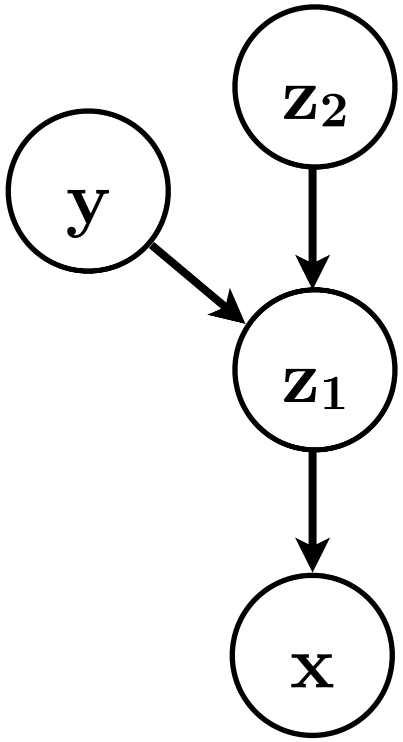


Semi-supervised Learning with Deep Generative Models

Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, Max Welling (NIPS 2014)

- **M1+M2**: Combination semi-supervised model
  - Train generative semi-supervised model on unsupervised features $\mathbf{z}1$ on unlabeled data, train a classifier to map from $\mathbf{z}1$ to label $\mathbf{z}1$.

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



Semi-supervised Learning with Deep Generative Models

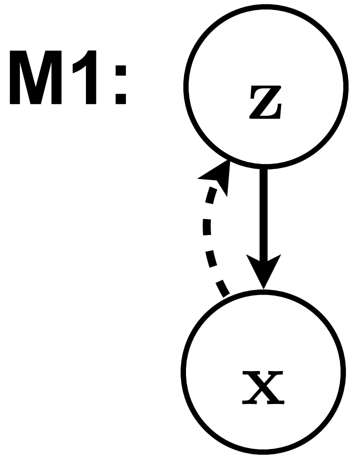
Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, Max Welling (NIPS 2014)

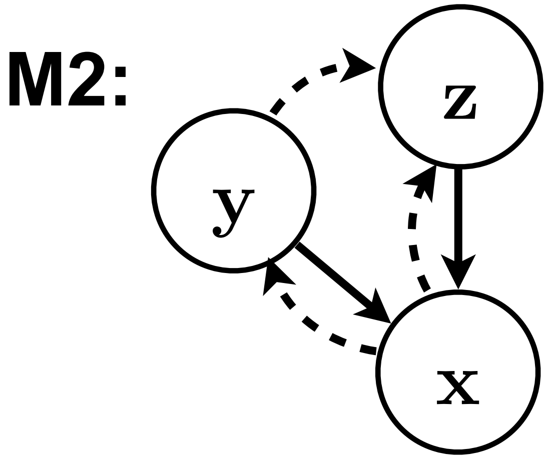
- Approximate posterior (encoder model)
    - Following the VAE strategy we parametrize the approximate posterior with a high capacity model, like a MLP or some other deep model (convnet, RNN, etc).

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

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

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





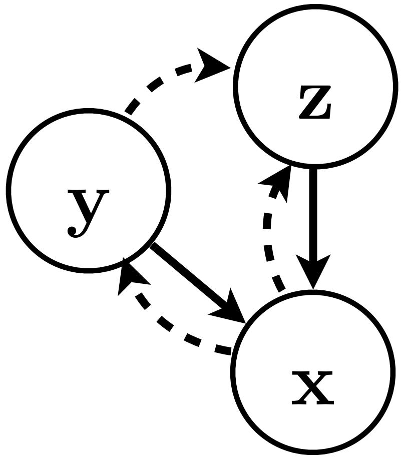
Semi-supervised Learning with Deep Generative Models

Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, Max Welling (NIPS 2014)

- **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 (NIPS 2014)

- Combination model M1+M2 shows dramatic improvement:

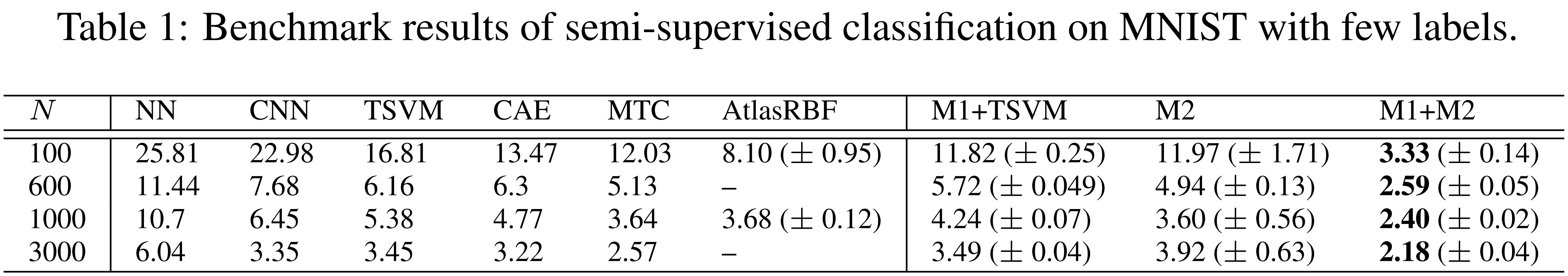


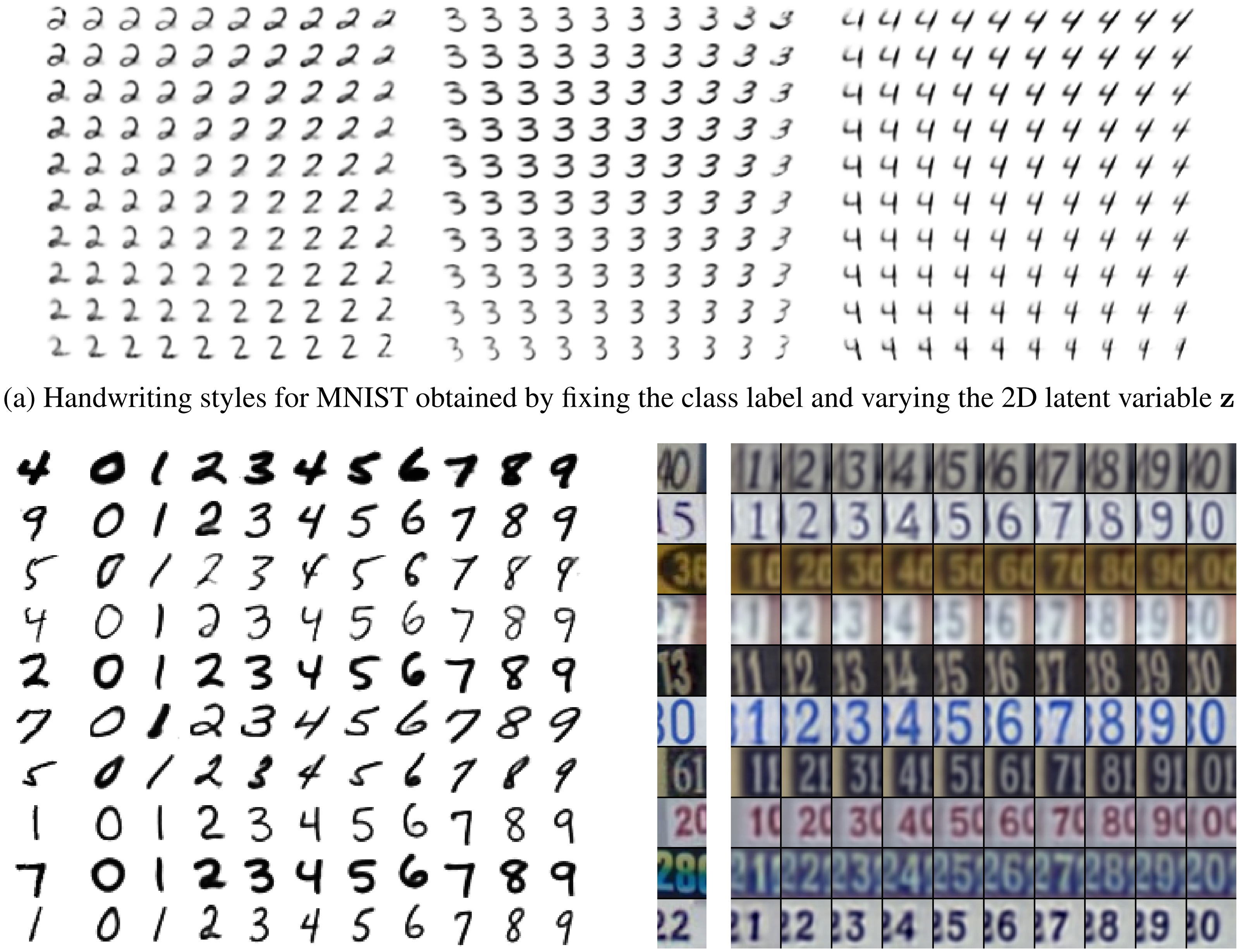
Table 1: Benchmark results of semi-supervised classification on MNIST with few labels.

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 (± 0.95)	11.82 (± 0.25)	11.97 (± 1.71)	**3.33** (± 0.14)
600	11.44	7.68	6.16	6.3	5.13	–	5.72 (± 0.049)	4.94 (± 0.13)	**2.59** (± 0.05)
1000	10.7	6.45	5.38	4.77	3.64	3.68 (± 0.12)	4.24 (± 0.07)	3.60 (± 0.56)	**2.40** (± 0.02)
3000	6.04	3.35	3.45	3.22	2.57	–	3.49 (± 0.04)	3.92 (± 0.63)	**2.18** (± 0.04)

- Full MNIST test error (non-convolutional): 0.96%
  - for comparison, current SOTA: 0.61%

Conditional generation using M2

Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, Max Welling (NIPS 2014)



(a) Handwriting styles for MNIST obtained by fixing the class label and varying the 2D latent variable **z**

(b) MNIST analogies

(c) SVHN analogies

Figure 1: **(a)** Visualisation of handwriting styles learned by the model with 2D **z**-space. **(b,c)** Analogical reasoning with generative semi-supervised models using a high-dimensional **z**-space. The leftmost columns show images from the test set. The other columns show analogical fantasies of **x** by the generative model, where the latent variable **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 **y**.

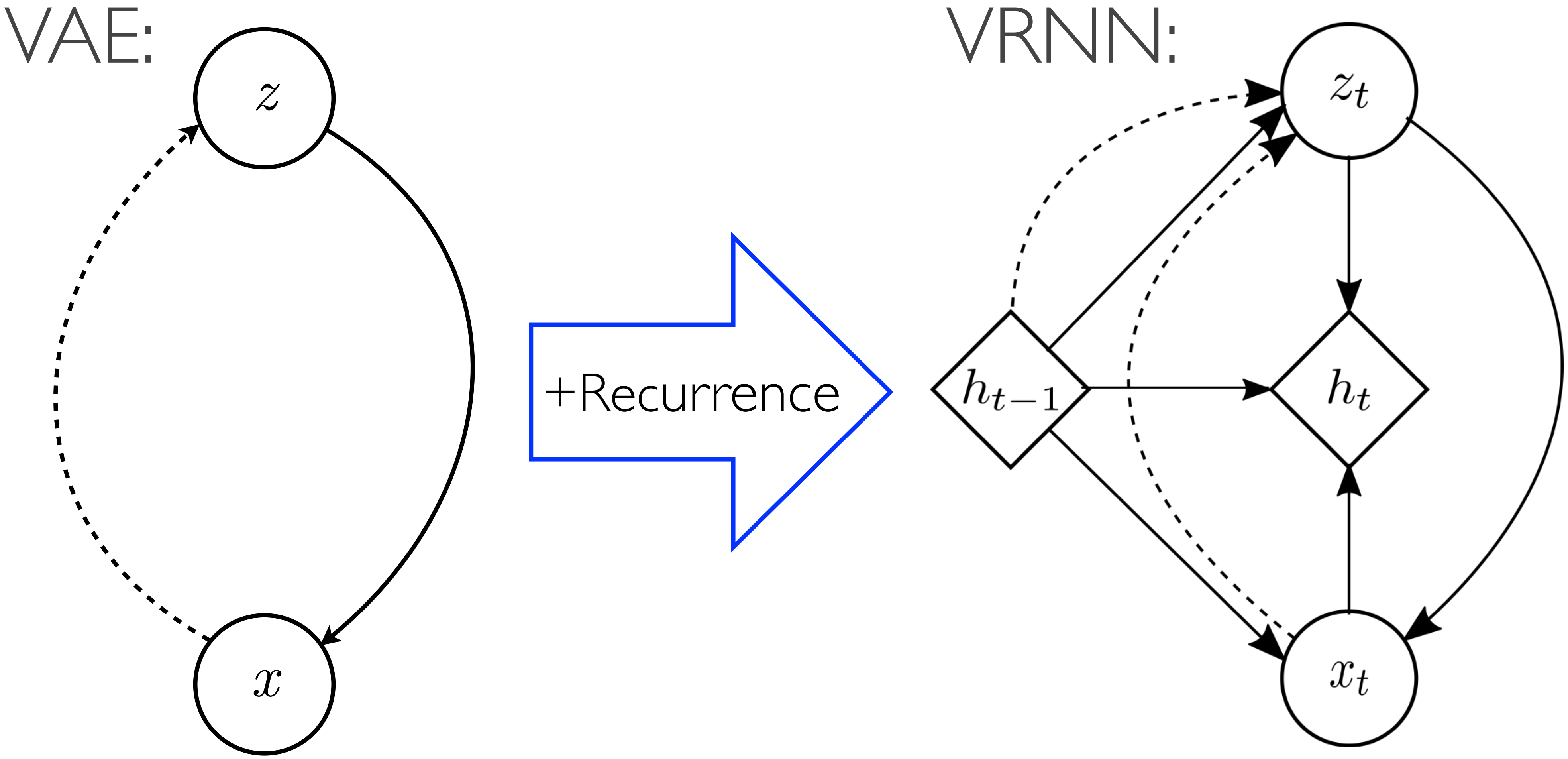
A Recurrent Latent Variable Model for Sequential Data

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron Courville, Yoshua Bengio

(arXiv, 2015)

VRNN: Model Structure

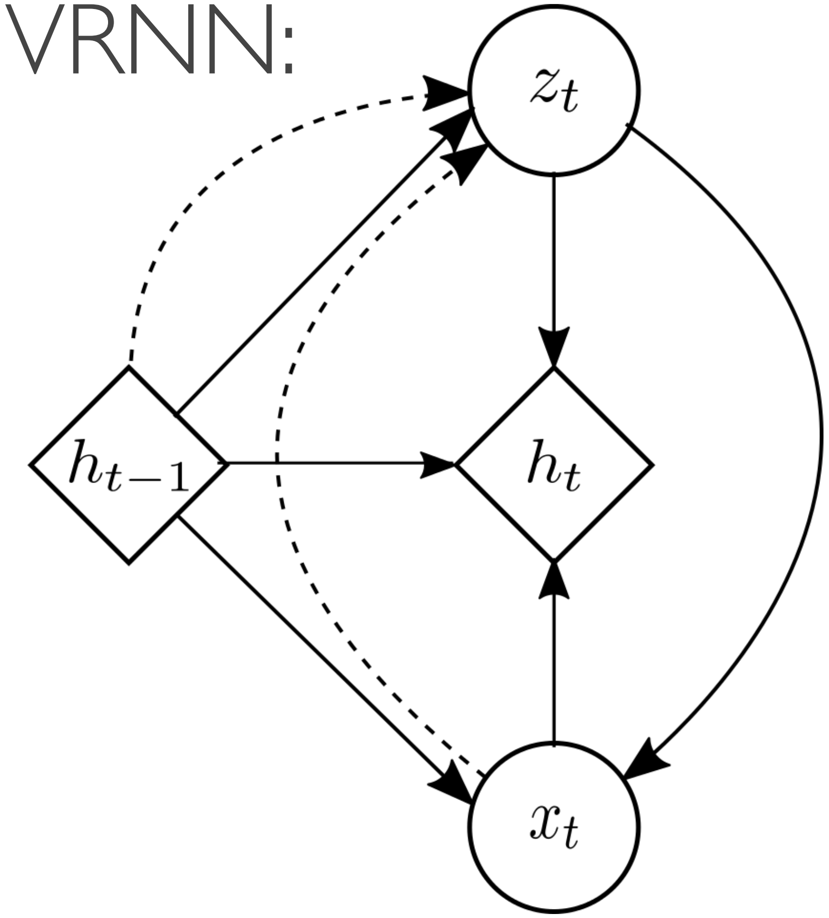
- Variational recurrent neural network (VRNN) is a recurrent (conditional) application of the VAE at every time-step.
- Recurrence is mediated through the recurrent hidden layer.
- **Motivation**: latent variables are a more natural space to encode stochasticity, standard RNNs encode noise in input.

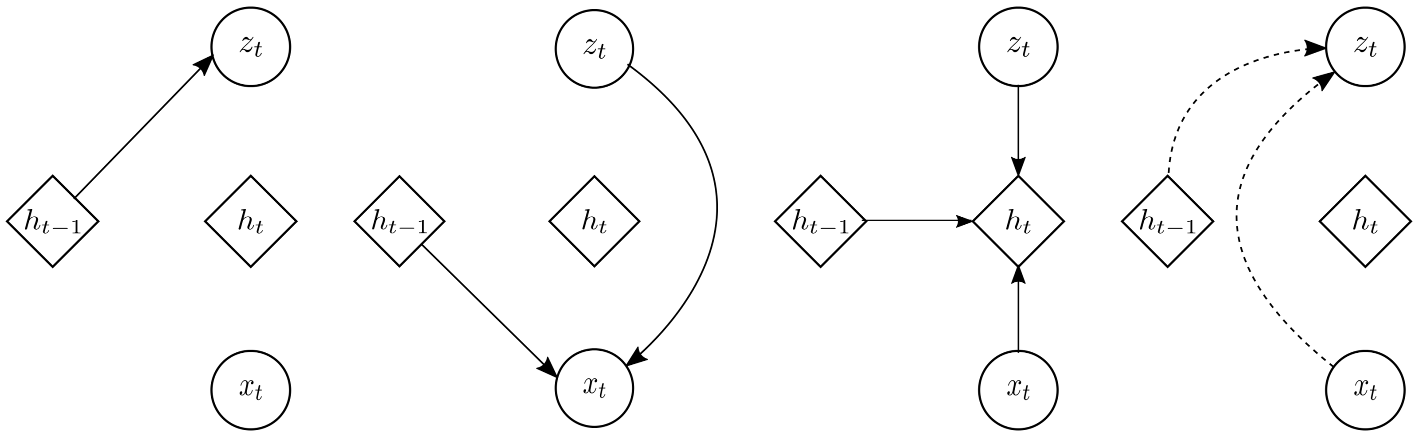


VRNN: Model Structure

- Variational recurrent neural network (VRNN) is a recurrent (conditional) application of the VAE at every time-step.
- Recurrence is mediated through the recurrent hidden layer.

VRNN:

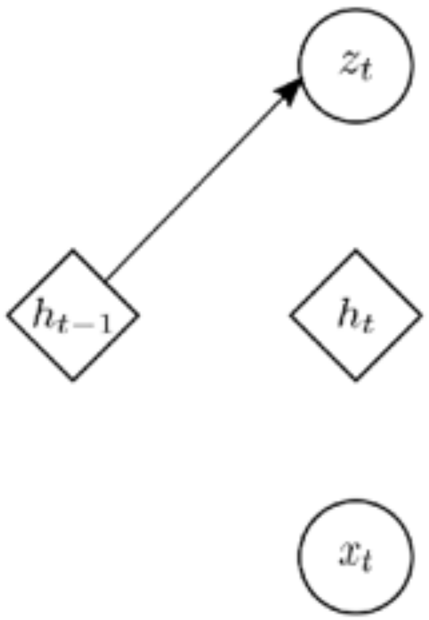




(a) Prior (b) Generation (c) Recurrence (d) Inference

VRNN: Prior on z_t

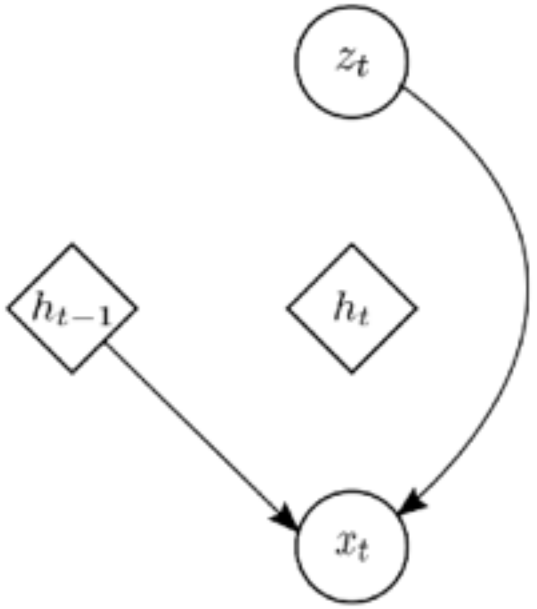
- At time step t, the latent variable z_t is generated as a function of the recurrent state at time step t-1.



$$\mathbf{z}_t \sim \mathcal{N}(\boldsymbol{\mu}_{0,t}, \text{diag}(\boldsymbol{\sigma}^2_{0,t}))\text{ , where }[\boldsymbol{\mu}_{0,t}, \boldsymbol{\sigma}_{0,t}] = \varphi^{\text{prior}}_{\tau}(\mathbf{h}_{t-1})$$

VRNN: Generation

- Generation of x_t uses the current latent variable z_t and the previous recurrent state $h_{t\text{-}1}$.



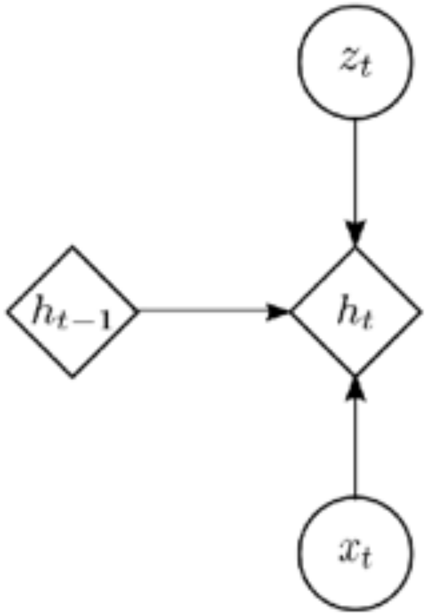
$$\mathbf{x}_t \mid \mathbf{z}_t \sim \mathcal{N}(\boldsymbol{\mu}_{x,t}, \mathrm{diag}(\boldsymbol{\sigma}_{x,t}^2)) \text{ , where } [\boldsymbol{\mu}_{x,t}, \boldsymbol{\sigma}_{x,t}] = \varphi_\tau^{\mathrm{dec}}(\varphi_\tau^{\mathbf{z}}(\mathbf{z}_t), \mathbf{h}_{t-1})$$

Generative model factorizes over time

$$p(\mathbf{x}_{\leq T}, \mathbf{z}_{\leq T}) = \prod_{t=1}^{T} p(\mathbf{x}_t \mid \mathbf{z}_{\leq t}, \mathbf{x}_{<t}) p(\mathbf{z}_t \mid \mathbf{x}_{<t}, \mathbf{z}_{<t}).$$

VRNN: Recurrence

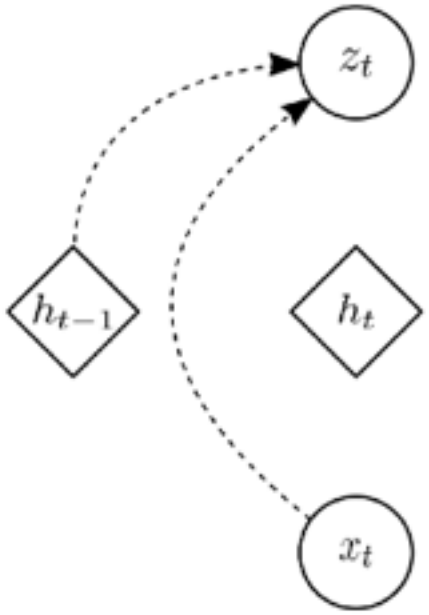
- Recurrent state h_t is a function of the previous recurrent state, the current observation h_t and the current latent variable h_t



$$\mathbf{h}_t = f_\theta\left(\varphi_\tau^{\mathbf{x}}(\mathbf{x}_t), \varphi_\tau^{\mathbf{z}}(\mathbf{z}_t), \mathbf{h}_{t-1}\right)$$

VRNN: Inference

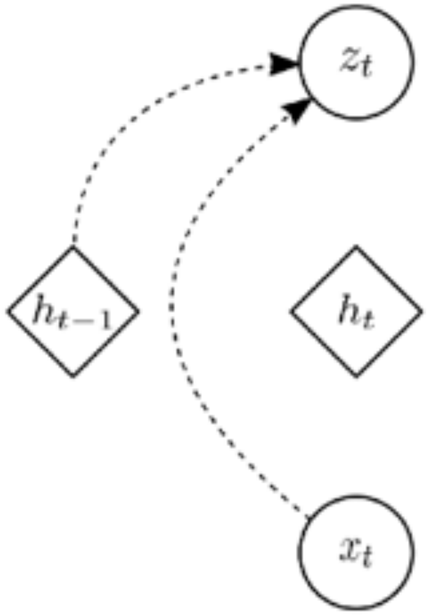
- Approximate posterior: $q(\mathbf{z}_{\leq T} \mid \mathbf{x}_{\leq T}) = \prod_{t=1}^{T} q(\mathbf{z}_t \mid \mathbf{x}_{\leq t}, \mathbf{z}_{<t})$
- where the history is summarized by the recurrent hidden state $h_{t\text{-}1}$.



$$\mathbf{z}_t \mid \mathbf{x}_t \sim \mathcal{N}(\boldsymbol{\mu}_{z,t}, \text{diag}(\boldsymbol{\sigma}_{z,t}^2))\text{ , where }[\boldsymbol{\mu}_{z,t}, \boldsymbol{\sigma}_{z,t}] = \varphi_\tau^{\text{enc}}(\varphi_\tau^{\mathbf{x}}(\mathbf{x}_t), \mathbf{h}_{t-1})$$

VRNN: Learning

- Learning is accomplished via gradient backpropagation:
  - through the decoder and encoder, as in standard VAE.
  - and through the recurrent connections, as in the standard RNN.



Objective function:

$$\int \log\left(\frac{p(\mathbf{x}_{\leq T}, \mathbf{z}_{\leq T})}{q(\mathbf{z}_{\leq T} \mid \mathbf{x}_{\leq T})}\right) dq(\mathbf{z}_{\leq T} \mid \mathbf{x}_{\leq T}) = \sum_{t=1}^{T} -\mathrm{KL}(q(\mathbf{z}_t \mid \mathbf{x}_{\leq t}, \mathbf{z}_{<t}) \| p(\mathbf{z}_t \mid \mathbf{x}_{<t}, \mathbf{z}_{<t})) + \mathbb{E}_{q(\mathbf{z}_t \mid \mathbf{x}_{\leq t}, \mathbf{z}_{<t})}\left[\log(p(\mathbf{x}_t \mid \mathbf{z}_{\leq t}, \mathbf{x}_{<t}))\right].$$

Factored version of the variational lower bound

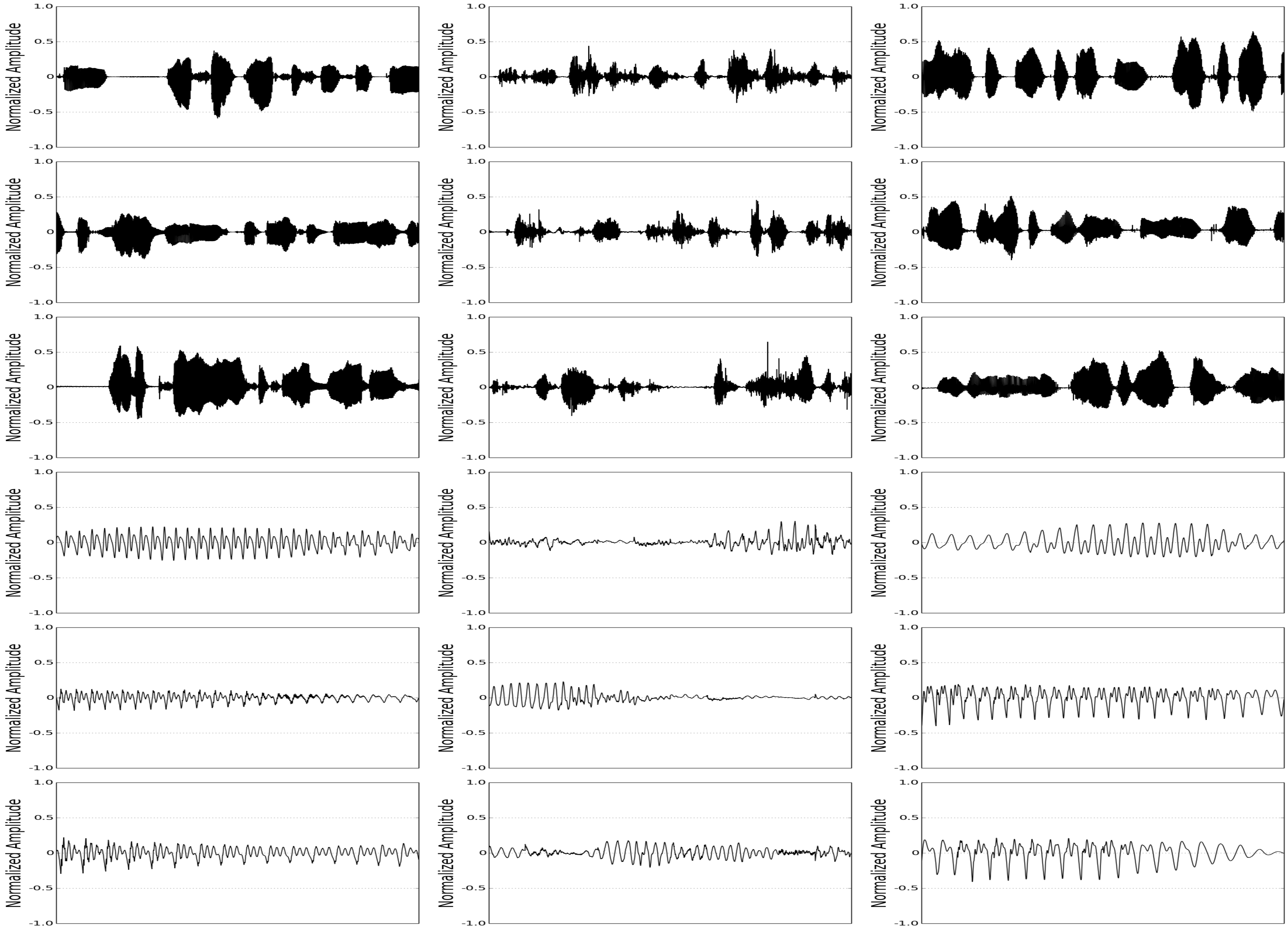
VRNN: Results

- Results on speech synthesis and handwriting synthesis
- Using stochastic latent variables allows for a more effective model than adding the stochasticity in the input

Table : Average log-probability on the test (or validation) set of each task.

	Speech modelling				Handwriting
Models	Blizzard	TIMIT	Onomatopoeia	Accent	IAM-OnDB
RNN-Gauss	3539	-1900	-984	-1293	1016
RNN-GMM	7413	26643	18865	3453	1358
VRNN-I-Gauss	$\geq$ 8933 $\approx$ 9188	$\geq$ 28340 $\approx$ 29639	$\geq$ 19053 $\approx$ 19638	$\geq$ 3843 $\approx$ 4180	$\geq$ 1332 $\approx$ 1353
VRNN-Gauss	$\geq$ 9223 $\approx$ **9516**	$\geq$ 28805 $\approx$ **30235**	$\geq$ 20721 $\approx$ **21332**	$\geq$ 3952 $\approx$ 4223	$\geq$ 1337 $\approx$ 1354
VRNN-GMM	$\geq$ 9107 $\approx$ 9392	$\geq$ 28982 $\approx$ 29604	$\geq$ 20849 $\approx$ 21219	$\geq$ 4140 $\approx$ **4319**	$\geq$ 1384 $\approx$ **1384**

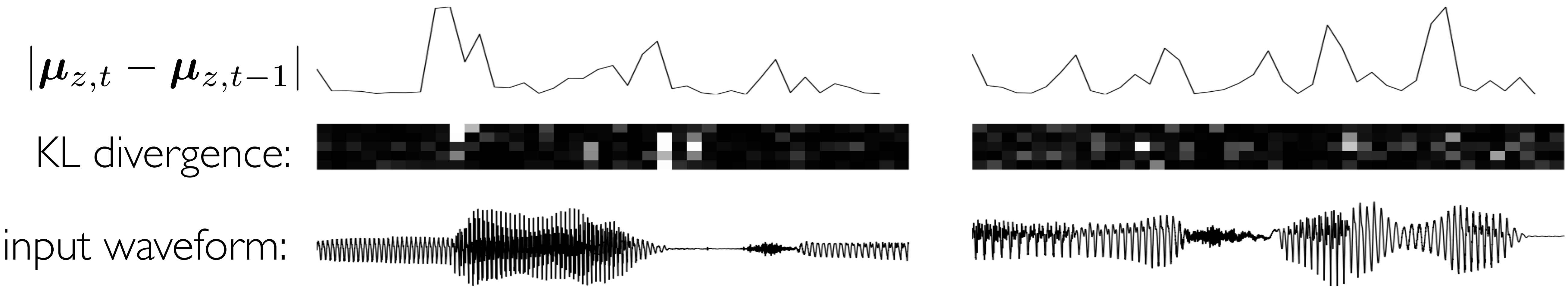
VRNN: Speech synthesis



(a) Ground Truth (b) RNN-GMM (c) VRNN-Gauss

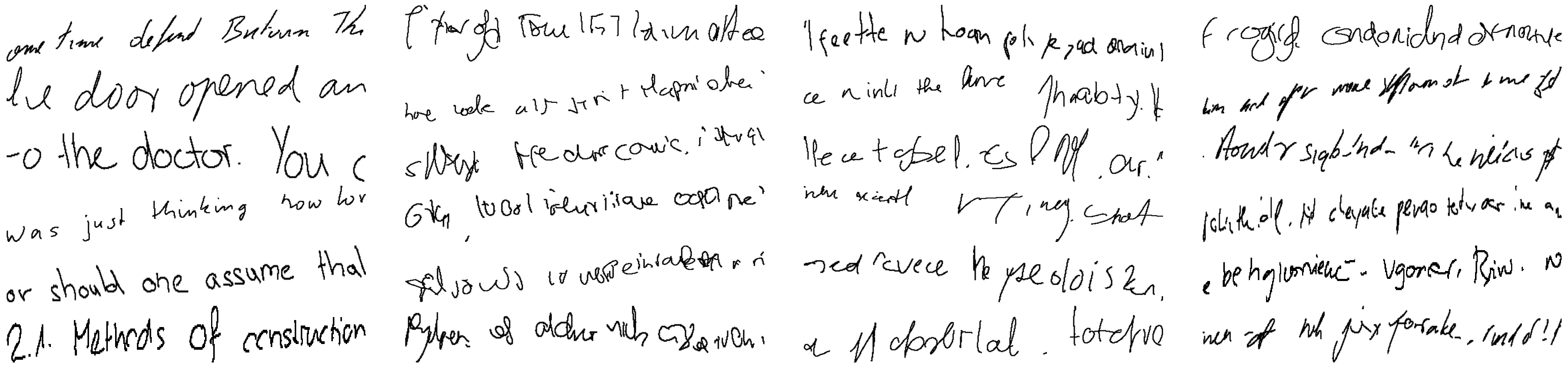
VRNN: KL Divergence

- The KL divergence tends to be fairly sparse and seems to be most active at motif transitions.



VRNN: Writing synthesis

- Predicting a sequence of (x,y) locations of the next destination of the pen.


ome time defend Britain The
he door opened an
-o the doctor. You c
was just thinking how lor
or should one assume that
2.1. Methods of construction

(a) Ground Truth (b) RNN-Gauss (c) RNN-GMM (d) VRNN-GMM

DRAW

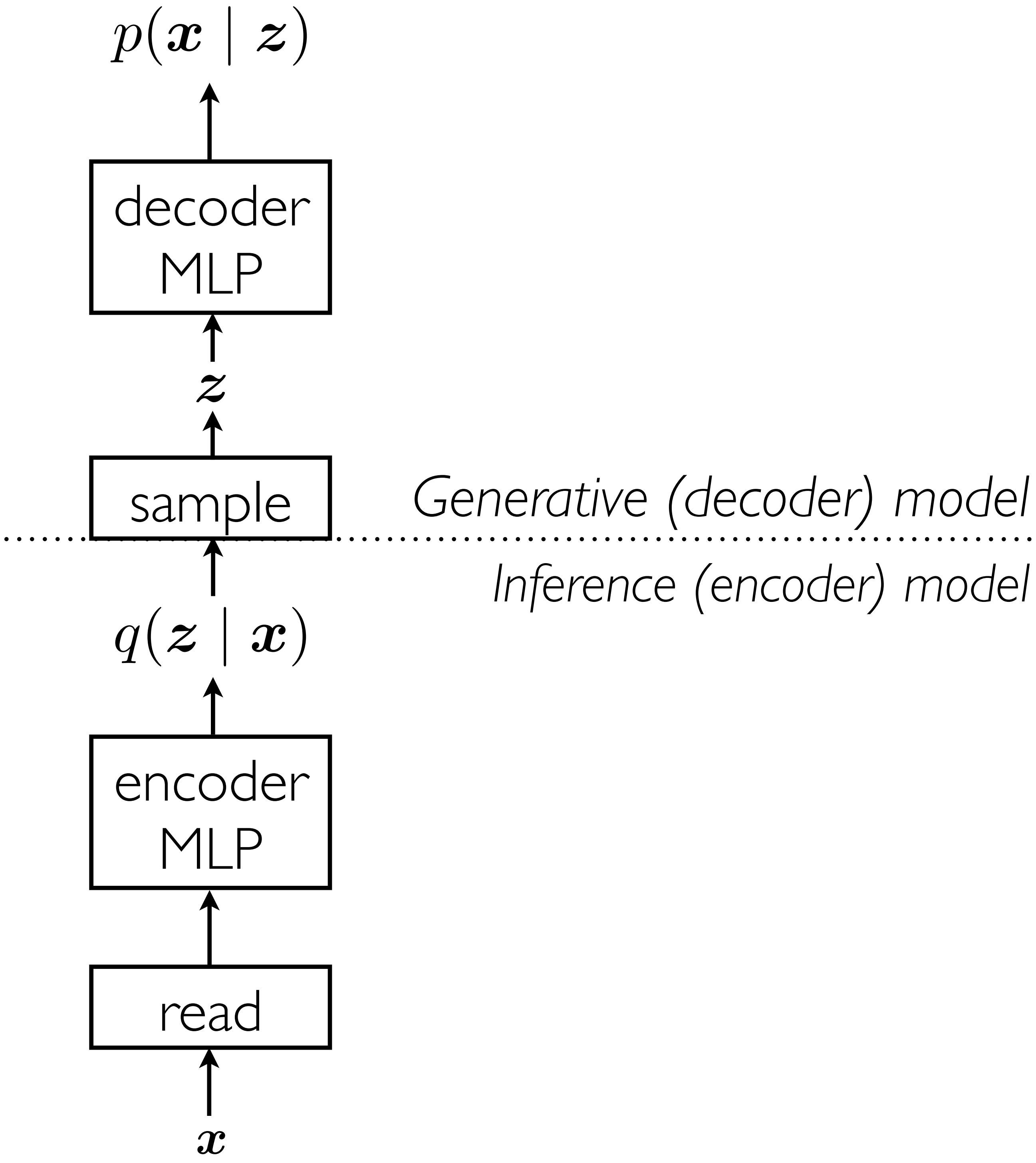
Deep Recurrent Attentive Writer

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, Daan Wierstra
Google Deepmind – (ICML, 2015)

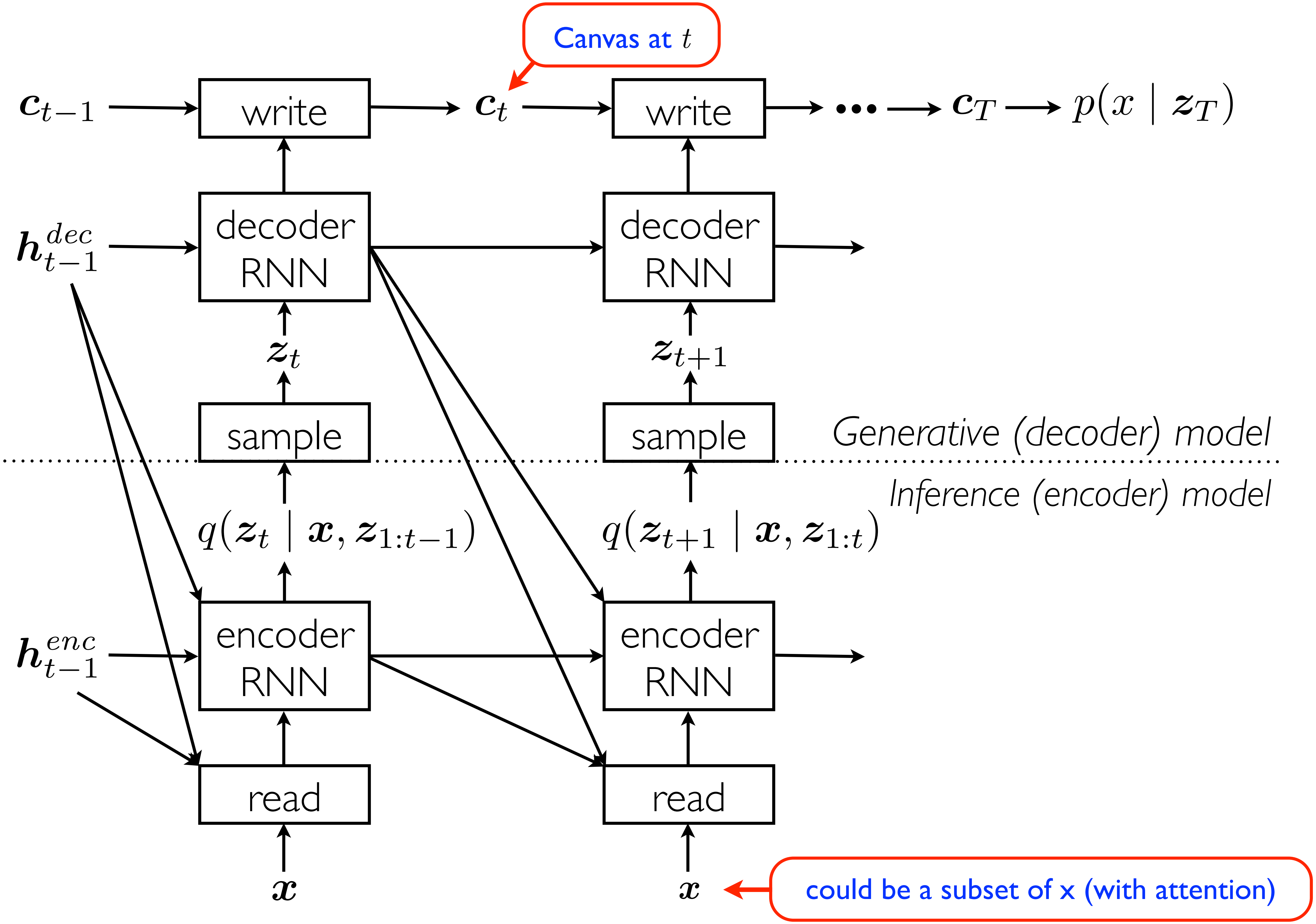
DRAW: Deep Recurrent Attentive Writer

- Augments the encoder and decoder with ***recurrent neural networks***.
- Inference and generation defined by a sequential process, *even for non-sequential data*.
- Adds an attention mechanism over the input to define a sequential process.

Variational Autoencoder Recap


$p(\boldsymbol{x} \mid \boldsymbol{z})$
decoder MLP
$\boldsymbol{z}$
sample
Generative (decoder) model
Inference (encoder) model
$q(\boldsymbol{z} \mid \boldsymbol{x})$
encoder MLP
read
$\boldsymbol{x}$

DRAW Model



Canvas at t

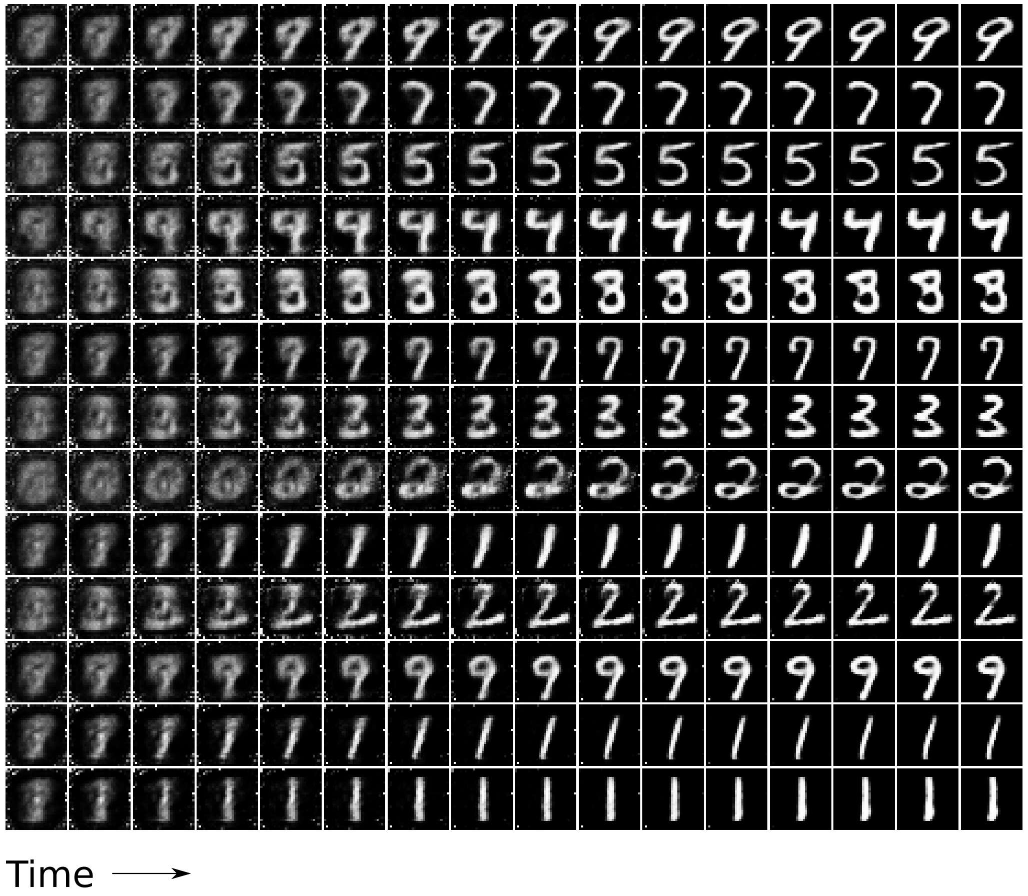
could be a subset of x (with attention)

DRAW MNIST Generation

- Simplest instantiation of DRAW is without an attention mechanism.
- Entire input is passed to the encoder at every time-step
- Decoder writes to entire canvas at every time-step

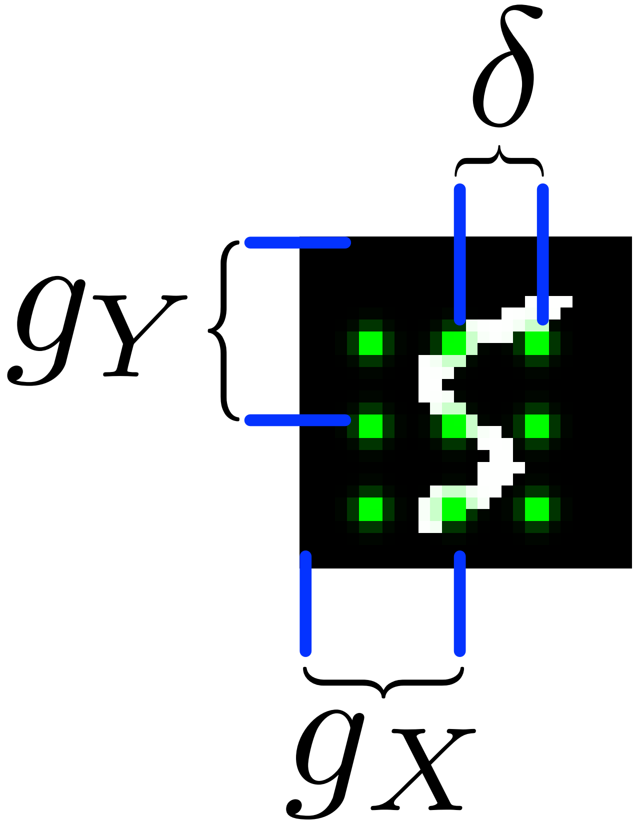
$$read(x, \hat{x}_t, h_{t-1}^{dec}) = [x, \hat{x}_t]$$

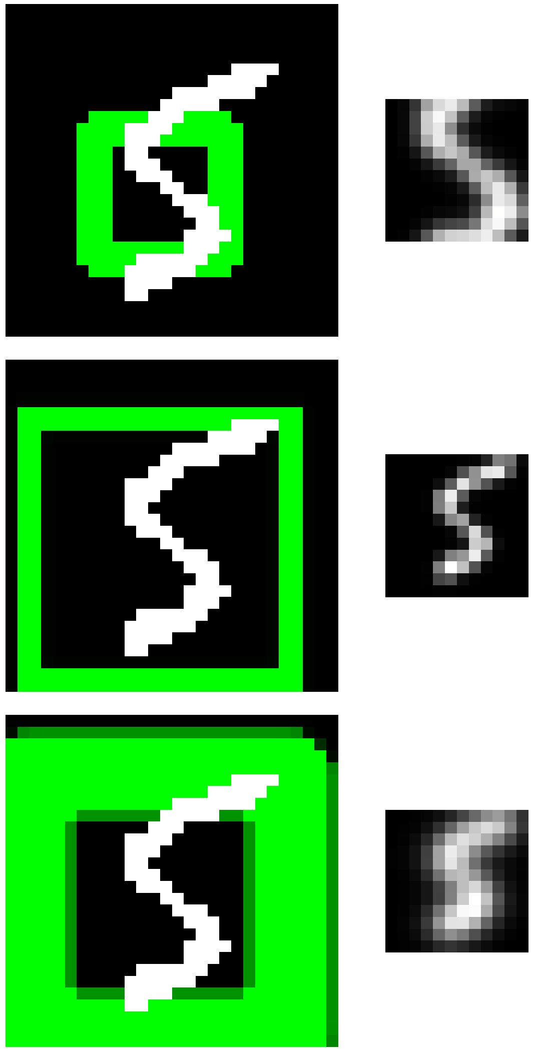
$$write(h_t^{dec}) = W(h_t^{dec})$$



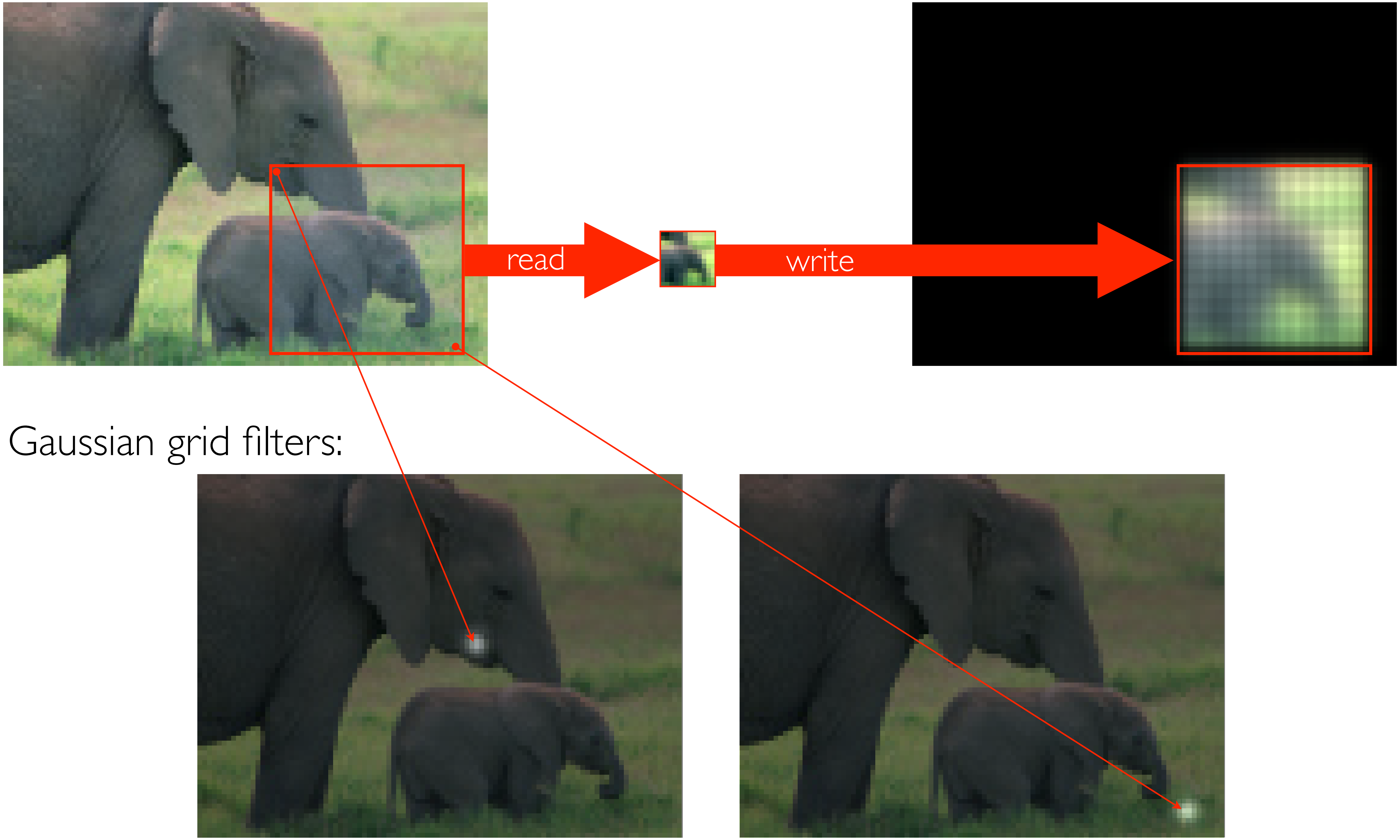
DRAW Attention Mechanism

- DRAW can use a **differentiable attention mechanism**.
- Attention uses recurrence (via the decoder) to select subsets of x for reading and writing.
- Attention controls the extracted **patch location, scale and blur**.





DRAW Attention Mechanism



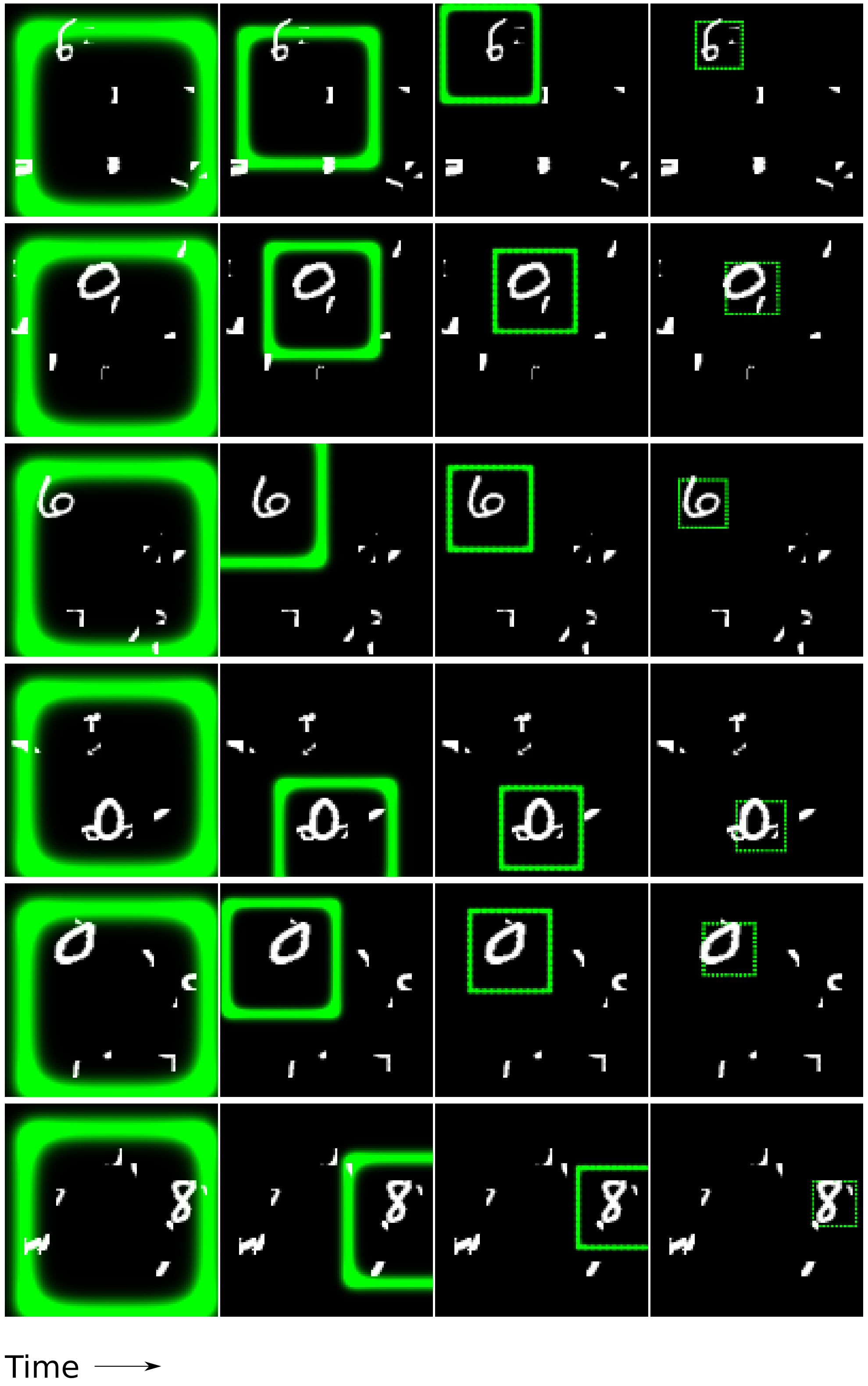
Gaussian grid filters:

DRAW: Cluttered MNIST Classification

- Draw w/ attention being applied to a classification task: cluttered MNIST.
- Attention learns to focus on the digit in the scene.

Table 1. Classification test error on 100 × 100 Cluttered Translated MNIST.

Model	Error
Convolutional, 2 layers	14.35%
RAM, 4 glimpses, 12 × 12, 4 scales	9.41%
RAM, 8 glimpses, 12 × 12, 4 scales	8.11%
Differentiable RAM, 4 glimpses, 12 × 12	4.18%
Differentiable RAM, 8 glimpses, 12 × 12	**3.36%**



DRAW MNIST Generation with Attention

Samples from DRAW with Attention:

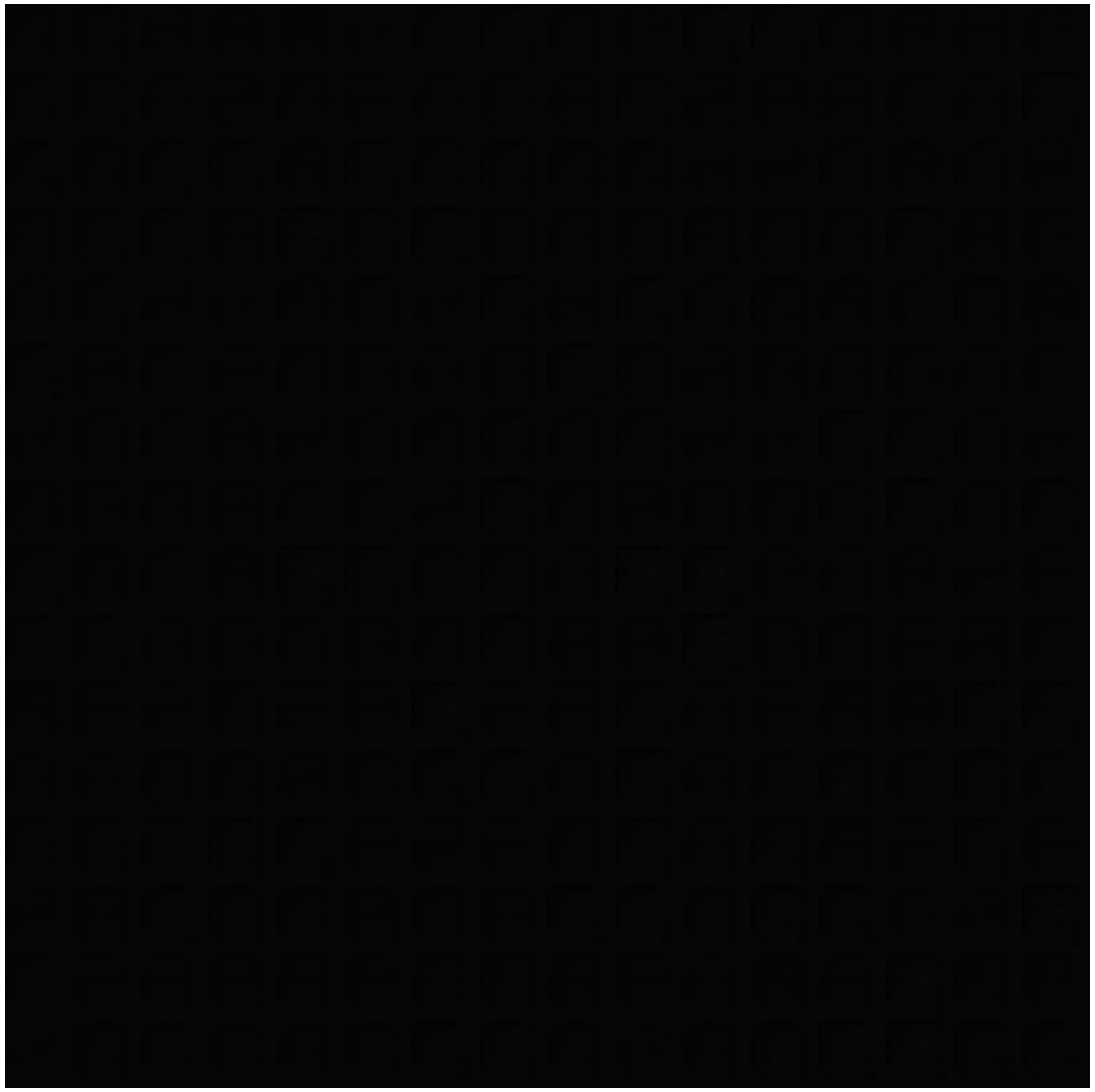


Image from Jörg Bornschein

NLL of MNIST test samples:

Model	$-\log p$	$\leq$
DBM 2hl [1]	≈ 84.62	
DBN 2hl [2]	≈ 84.55	
NADE [3]	88.33	
EoNADE 2hl (128 orderings) [3]	85.10	
EoNADE-5 2hl (128 orderings) [4]	84.68	
DLGM [5]	≈ 86.60	
DLGM 8 leapfrog steps [6]	≈ 85.51	88.30
DARN 1hl [7]	≈ 84.13	88.30
DARN 12hl [7]	-	87.72
DRAW without attention	-	87.40
DRAW	-	**80.97**

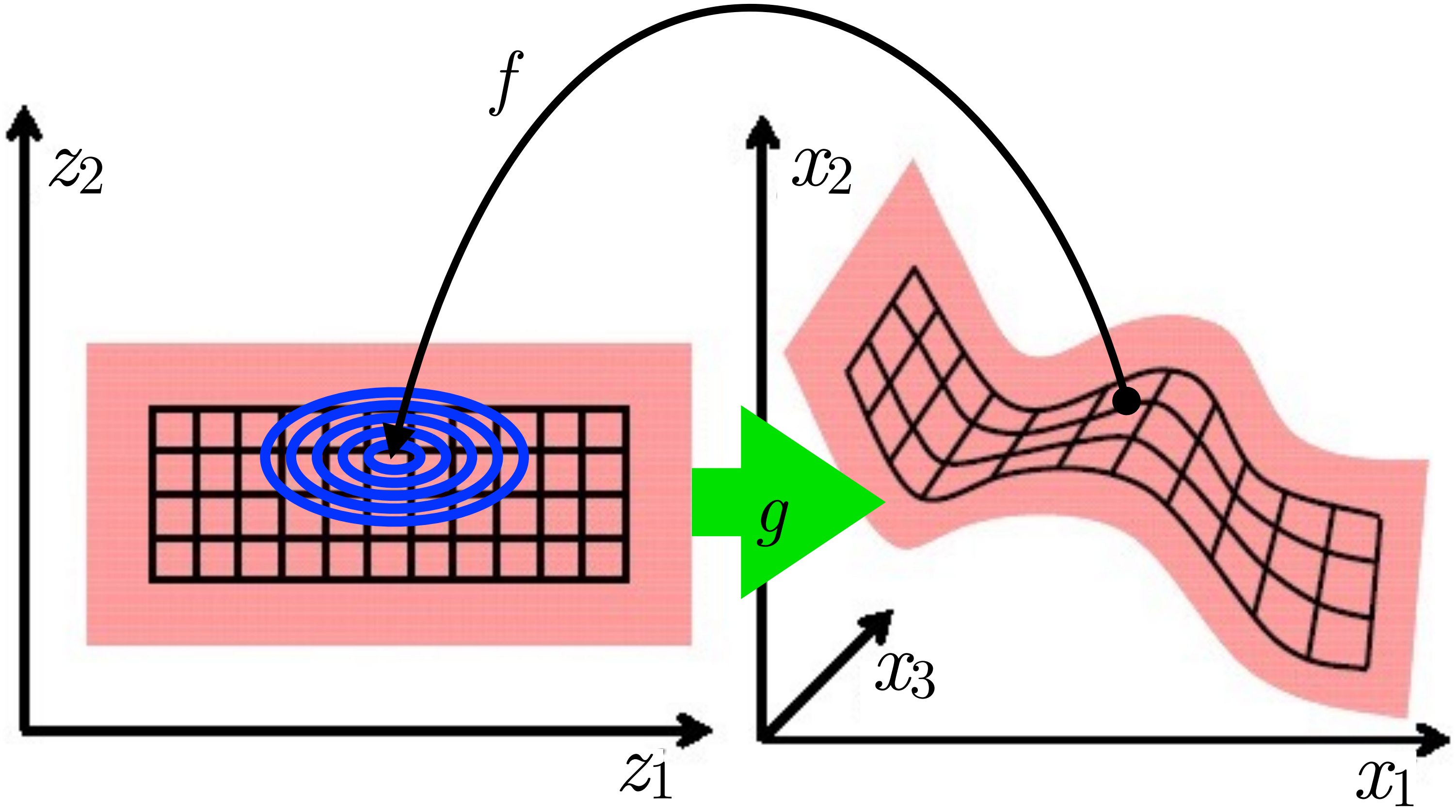
This is really low!

DRAW MNIST Generation with Attention

Recent Innovations in VAE Inference

Inference in the VAE

- VAE inference approximates the posterior $p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \boldsymbol{x})$ with a distribution that is conditionally independent in $\boldsymbol{z}$: $q_{\boldsymbol{\phi}}(\boldsymbol{z} \mid \boldsymbol{x}) = \prod_i q_{\boldsymbol{\phi}}(\boldsymbol{z_i} \mid \boldsymbol{x})$
  - Consequence: Non-multimodal, i.e. unimodal distribution.



- Can parametrize some distribution (e.g. a full cov. Gaussian), but what is the right distribution?

- Can we lessen this restriction? How can we get closer to $p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \boldsymbol{x})$?

Variational Inference with Normalizing Flows

Danilo Jimenez Rezende, Shakir Mohamed
Google Deepmind – (ICML, 2015)

Normalizing Flows

- How do we specify a complicated joint distribution over $\boldsymbol{z}$?

- **Normalizing flows**: the transformation of a probability density through a sequence of invertible mappings.
  - By repeated application of the rule for random variable transformations, the initial density flows through the sequence of invertible mappings.
  - At the end of the sequence, we have a valid (maybe complex) probability distribution.

- Transformation of random variables: $\boldsymbol{z}' = \boldsymbol{f}(\boldsymbol{z}), \quad \boldsymbol{f}^{-1}(\boldsymbol{z}') = \boldsymbol{z}$

$$q(\boldsymbol{z}') = q(\boldsymbol{z}) \left| \det \frac{\partial \boldsymbol{f}^{-1}}{\partial \boldsymbol{z}'} \right| = q(\boldsymbol{z}) \left| \det \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}'} \right|^{-1}$$

- Chaining together a sequence: $\boldsymbol{z}_K = \boldsymbol{f}_K \circ \boldsymbol{f}_{K-1} \circ \cdots \circ \boldsymbol{f}_2 \circ \boldsymbol{f}_1(\boldsymbol{z}_0)$

$$\log q_K(\boldsymbol{z}_K) = \log q_0(\boldsymbol{z}_0) - \sum_{k=1}^{K} \log \left| \det \frac{\partial \boldsymbol{f}_k}{\partial \boldsymbol{z}_k} \right|$$

Normalizing Flows

- **Law of the unconscious statistician**: expectations w.r.t. the transformed density $q_K(\boldsymbol{z}_K)$ can be written as expectations w.r.t. the original $q_0(\boldsymbol{z}_0)$. For $\boldsymbol{z}_K = \boldsymbol{f}_K \circ \boldsymbol{f}_{K-1} \circ \cdots \circ \boldsymbol{f}_2 \circ \boldsymbol{f}_1(\boldsymbol{z}_0)$,

$$\mathbb{E}_{q_K}\left[g(\boldsymbol{z}_K)\right] = \mathbb{E}_{q_0}[g(\boldsymbol{f}_K \circ \boldsymbol{f}_{K-1} \circ \cdots \circ \boldsymbol{f}_2 \circ \boldsymbol{f}_1(\boldsymbol{z}_0))]$$

- The variational lower bound:

$$\begin{aligned}\mathcal{L}(\theta, \phi, \boldsymbol{x}) &= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x}, \boldsymbol{z}) - \log q_\phi(\boldsymbol{z} \mid \boldsymbol{x})\right] \\ &= \mathbb{E}_{q_K(\boldsymbol{z}_K)}\left[\log p(\boldsymbol{x}, \boldsymbol{z}_K) - \log q_K(\boldsymbol{z}_K)\right] \\ &= \mathbb{E}_{q_0(\boldsymbol{z}_0)}\left[\log p(\boldsymbol{x}, \boldsymbol{z}_K) - \log q_0(\boldsymbol{z}_0) + \sum_{k=1}^{K} \log\left|\det \frac{\partial \boldsymbol{f}_k}{\partial \boldsymbol{z}_k}\right|\right]\end{aligned}$$

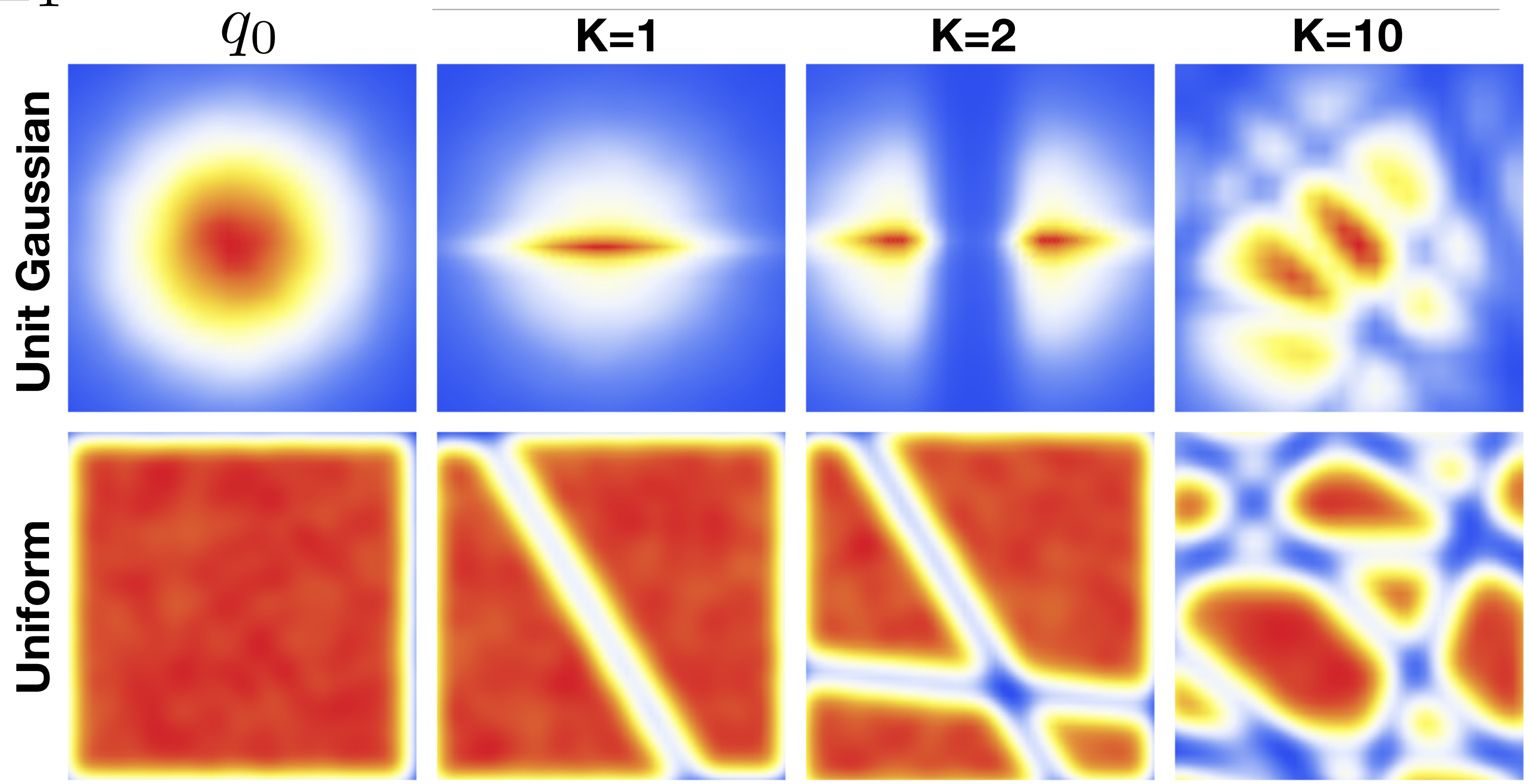
Normalizing Flows for VAE posteriors

- Consider the family of transformations: $f(\boldsymbol{z}) = \boldsymbol{z} + \boldsymbol{u}h\left(\boldsymbol{w}^\top \boldsymbol{z} + b\right)$

$$\psi(\boldsymbol{z}) = h'\left(\boldsymbol{w}^\top \boldsymbol{z} + b\right)\boldsymbol{w} \qquad \left|\det \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}}\right| = \left|1 + \boldsymbol{u}^\top \psi(\boldsymbol{z})\right|$$

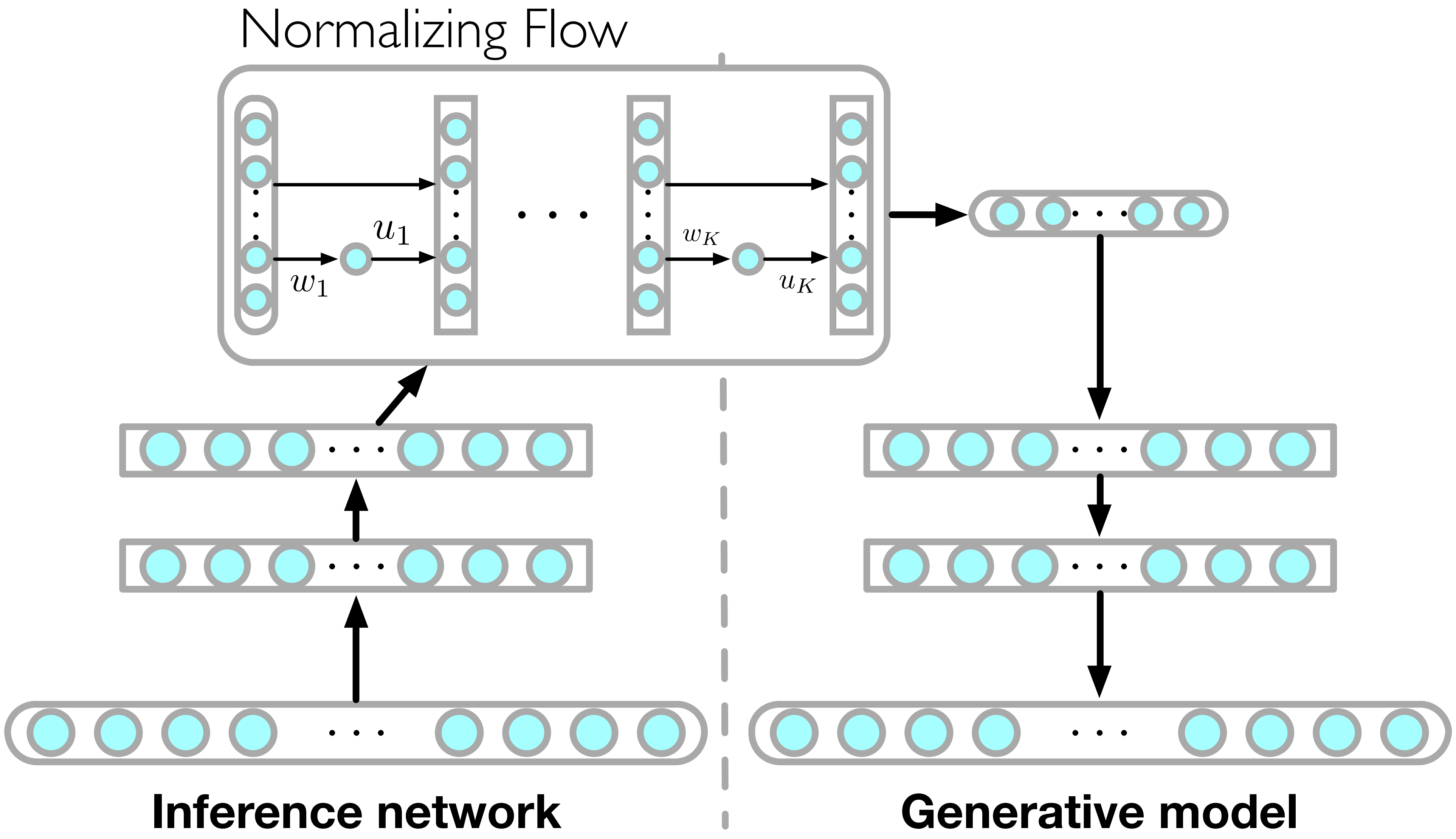
- Chaining these transformations gives us a rich family of posteriors,

$$\log q_K(\boldsymbol{z}_K) = \log q_0(\boldsymbol{z}_0) - \sum_{k=1}^{K} \log \left|1 + \boldsymbol{u}_k^\top \psi_k(\boldsymbol{z}_k)\right|$$



Normalizing Flows for VAE posteriors

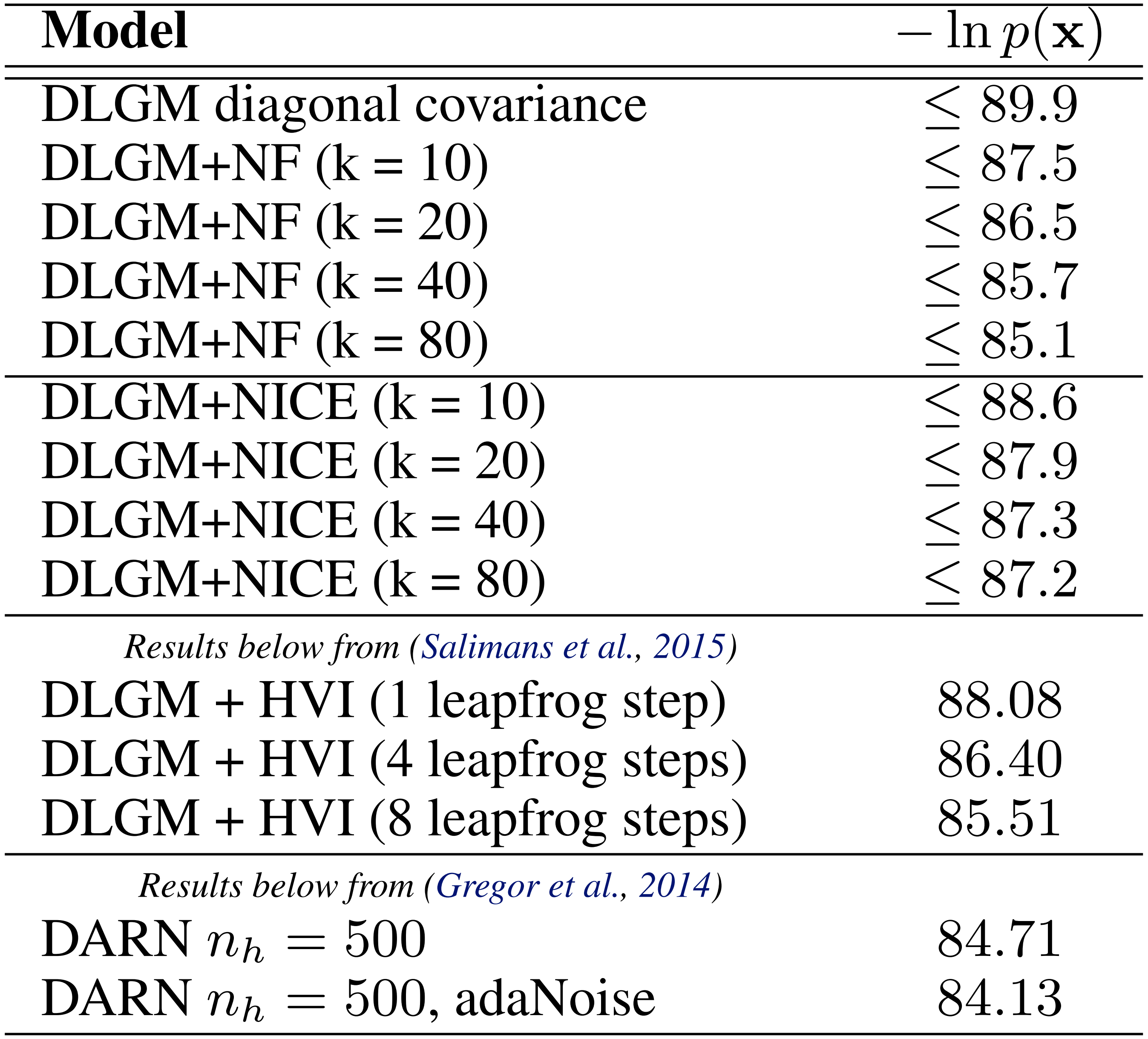
- Normalizing flow integration into the VAE: $f(\boldsymbol{z}) = \boldsymbol{z} + \boldsymbol{u}h\left(\boldsymbol{w}^\top \boldsymbol{z} + b\right)$



- Normalizing flows are fully differentiable, so learning via gradient backpropagation can proceed as before.

Normalizing Flows for VAE posteriors

- Quantitative comparison to other methods shows the benefit of the normalizing flows.



Model	$-\ln p(\mathbf{x})$
DLGM diagonal covariance	≤ 89.9
DLGM+NF (k = 10)	≤ 87.5
DLGM+NF (k = 20)	≤ 86.5
DLGM+NF (k = 40)	≤ 85.7
DLGM+NF (k = 80)	≤ 85.1
DLGM+NICE (k = 10)	≤ 88.6
DLGM+NICE (k = 20)	≤ 87.9
DLGM+NICE (k = 40)	≤ 87.3
DLGM+NICE (k = 80)	≤ 87.2
Results below from (Salimans et al., 2015)	
DLGM + HVI (1 leapfrog step)	88.08
DLGM + HVI (4 leapfrog steps)	86.40
DLGM + HVI (8 leapfrog steps)	85.51
Results below from (Gregor et al., 2014)	
DARN $n_h = 500$	84.71
DARN $n_h = 500$, adaNoise	84.13

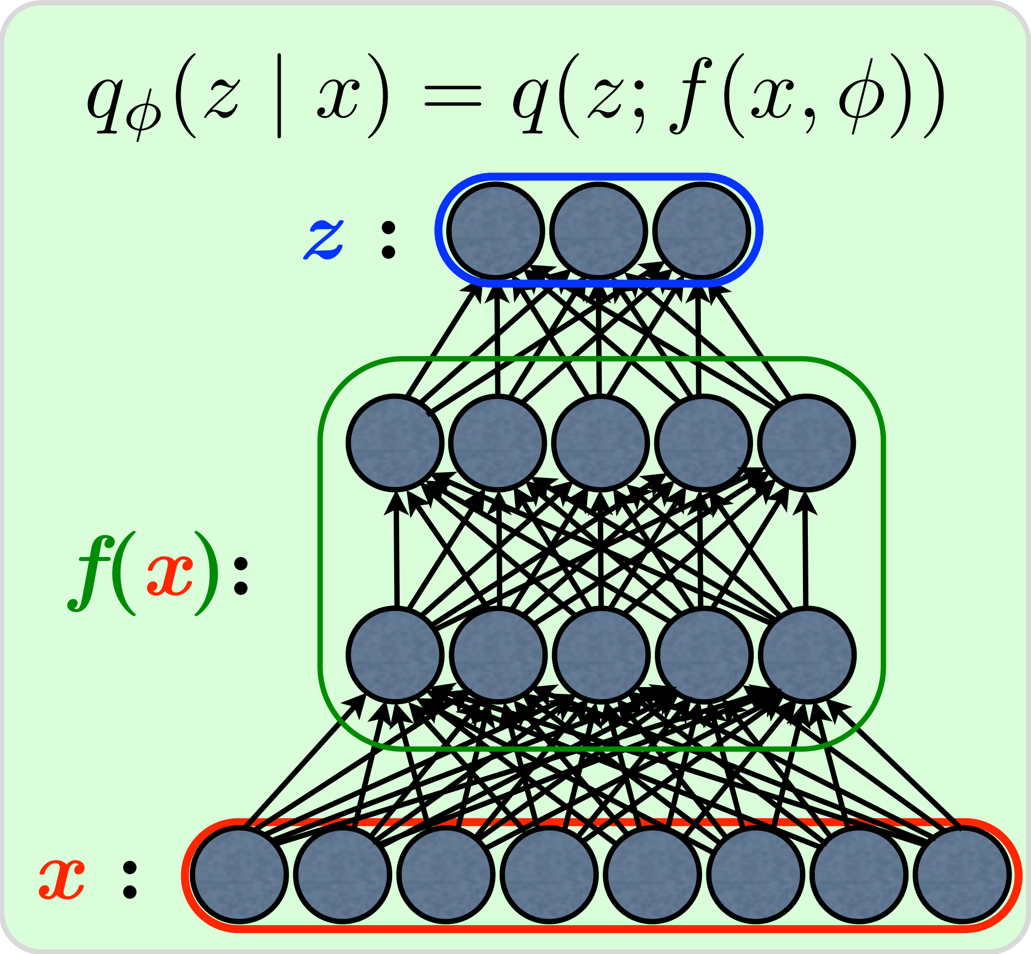
Recall that DRAW achieved <= 80.97

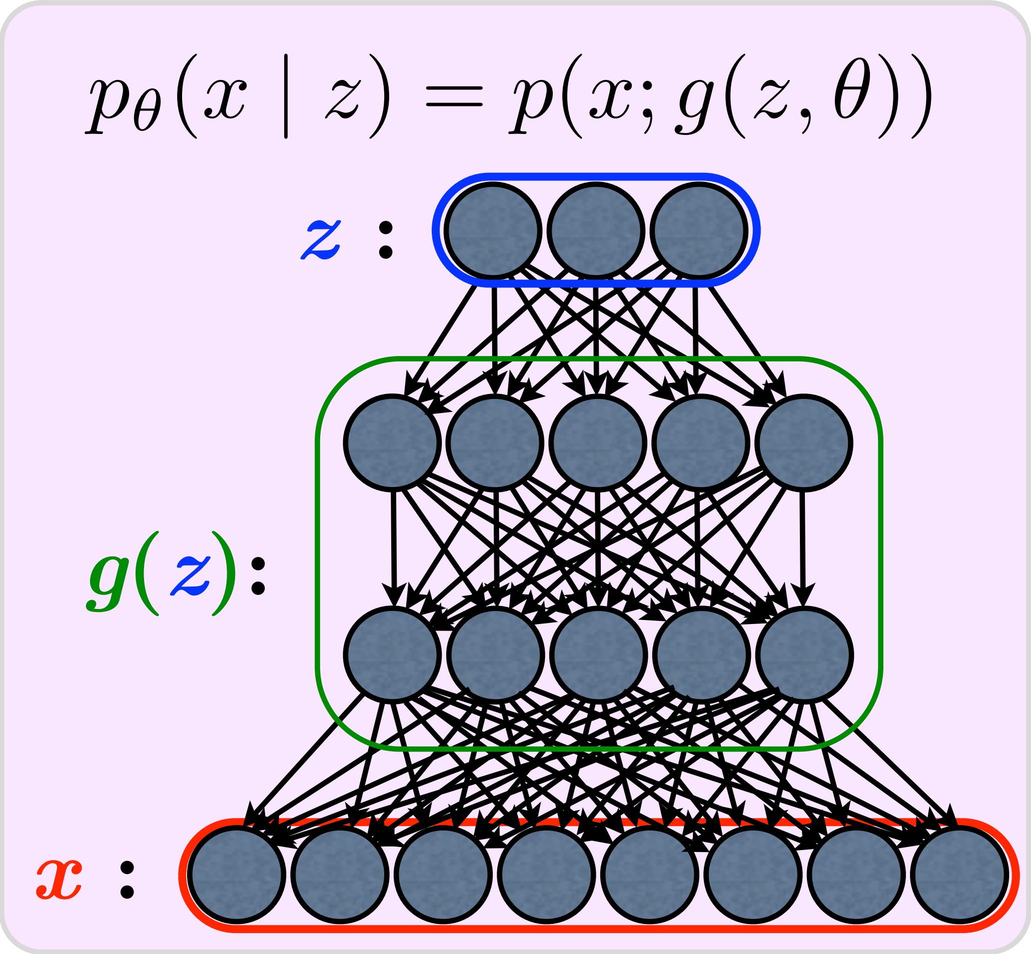
Markov Chain Monte Carlo and Variational Inference: Bridging the Gap

Tim Salimans, Diederik P. Kingma, Max Welling
(ICML, 2015)

Variational and MCMC inference

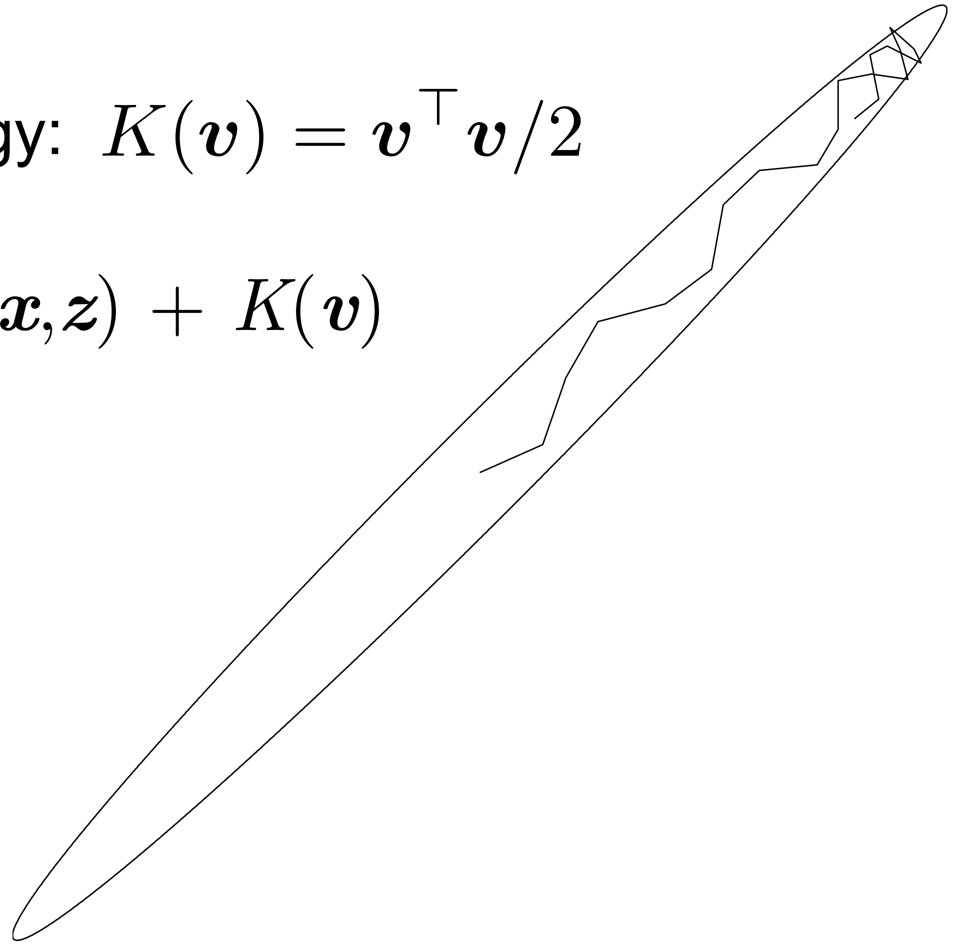
- Variational inference (a la VAE) and MCMC inference have different properties
  - Variational inference (VI) is efficient / MCMC is computationally intensive
  - VI has a fixed parametric form / MCMC asymptotically approaches $p(\boldsymbol{z} \mid \boldsymbol{x})$
- Can we combine these two approaches to find a good compromise?


$q_\phi(z \mid x) = q(z; f(x, \phi))$
$\boldsymbol{z}$:
$\boldsymbol{f}(\boldsymbol{x})$:
$\boldsymbol{x}$:


$p_\theta(x \mid z) = p(x; g(z, \theta))$
$\boldsymbol{z}$:
$\boldsymbol{g}(\boldsymbol{z})$:
$\boldsymbol{x}$:

Hamiltonian (Hybrid) Monte Carlo

- **Basic Idea:**
  - Consider sampling from the posterior $p(\boldsymbol{z} \mid \boldsymbol{x})$ as a physics simulation from a frictionless ball rolling on the potential energy surface $E(\boldsymbol{x},\boldsymbol{z}) = \log\, p(\boldsymbol{x},\boldsymbol{z})$.
  - Augment with a velocity $\boldsymbol{v}$ with kinetic energy: $K(\boldsymbol{v}) = \boldsymbol{v}^\top \boldsymbol{v}/2$
  - Total energy = Hamiltonian: $H(\boldsymbol{x},\boldsymbol{z},\boldsymbol{v}) = E(\boldsymbol{x},\boldsymbol{z}) + K(\boldsymbol{v})$



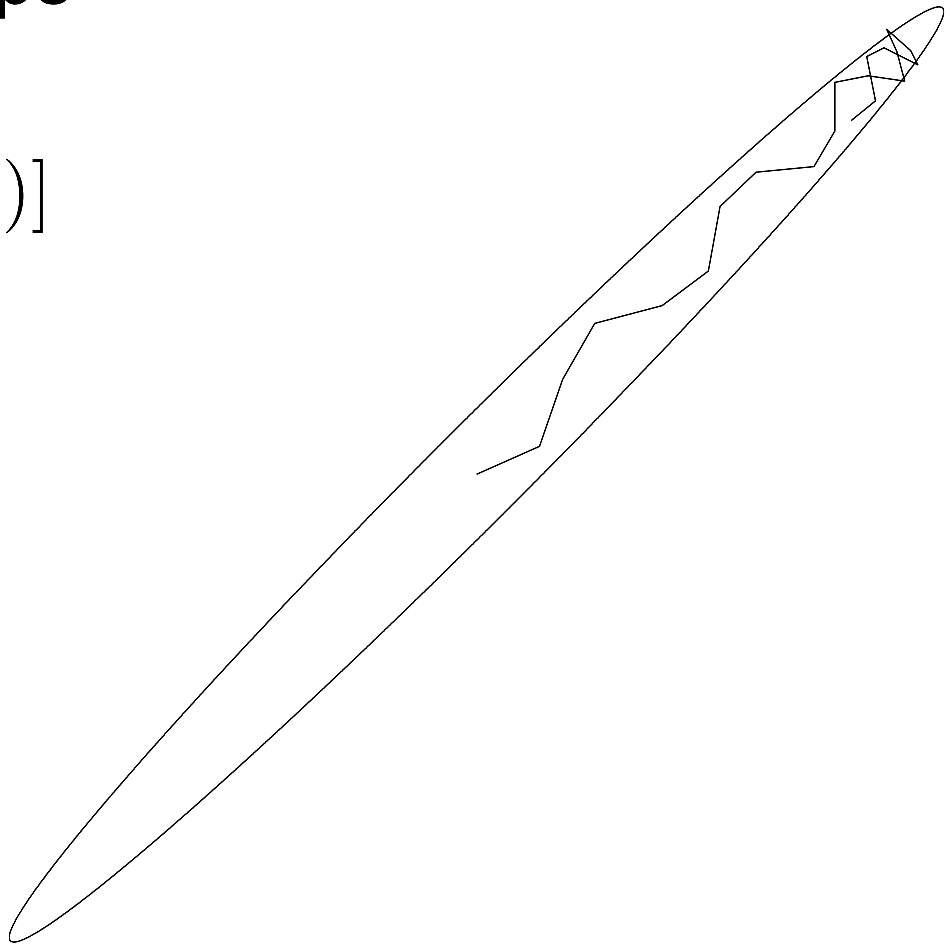
- **HMC innovation:** If gradients of $H(\boldsymbol{x},\boldsymbol{z},\boldsymbol{v})$ are available then we can use that information to move around the surface more effectively.

Hamiltonian (Hybrid) Monte Carlo

The HMC algorithm:

- Gibbs sample the velocity $\boldsymbol{v} \sim \mathcal{N}(\mathbf{0}, \mathbb{I})$
- Simulate **leapfrog dynamics** for T steps
- Accept new position with probability

$$\min\left[1, \exp(H(\boldsymbol{x}, \boldsymbol{z}_0, \boldsymbol{v}_0) - H(\boldsymbol{x}, \boldsymbol{z}_T, \boldsymbol{v}_T))\right]$$



Leapfrog dynamics:

$$\boldsymbol{v}_{t+\frac{\epsilon}{2}} = \boldsymbol{v}_t - \frac{\epsilon}{2}\nabla_{\boldsymbol{z}}\left(\log p(\boldsymbol{x}, \boldsymbol{z}_t)\right)$$

$$\boldsymbol{z}_{t+\epsilon} = \boldsymbol{z}_t + \epsilon \boldsymbol{v}_{t+\frac{\epsilon}{2}}$$

$$\boldsymbol{v}_{t+\epsilon} = \boldsymbol{v}_{t+\frac{\epsilon}{2}} - \frac{\epsilon}{2}\nabla_{\boldsymbol{z}}\left(\log p(\boldsymbol{x}, \boldsymbol{z}_{t+\epsilon})\right)$$

HMC for Deep Generative Models

The HMC algorithm:

- Gibbs sample the velocity $\boldsymbol{v} \sim \mathcal{N}(\mathbf{0}, \mathbb{I})$
- Simulate **leapfrog dynamics** for T steps
- Accept new position with probability $\min\left[1, \exp(H(\boldsymbol{x}, \boldsymbol{z}_0, \boldsymbol{v}_0) - H(\boldsymbol{x}, \boldsymbol{z}_T, \boldsymbol{v}_T))\right]$

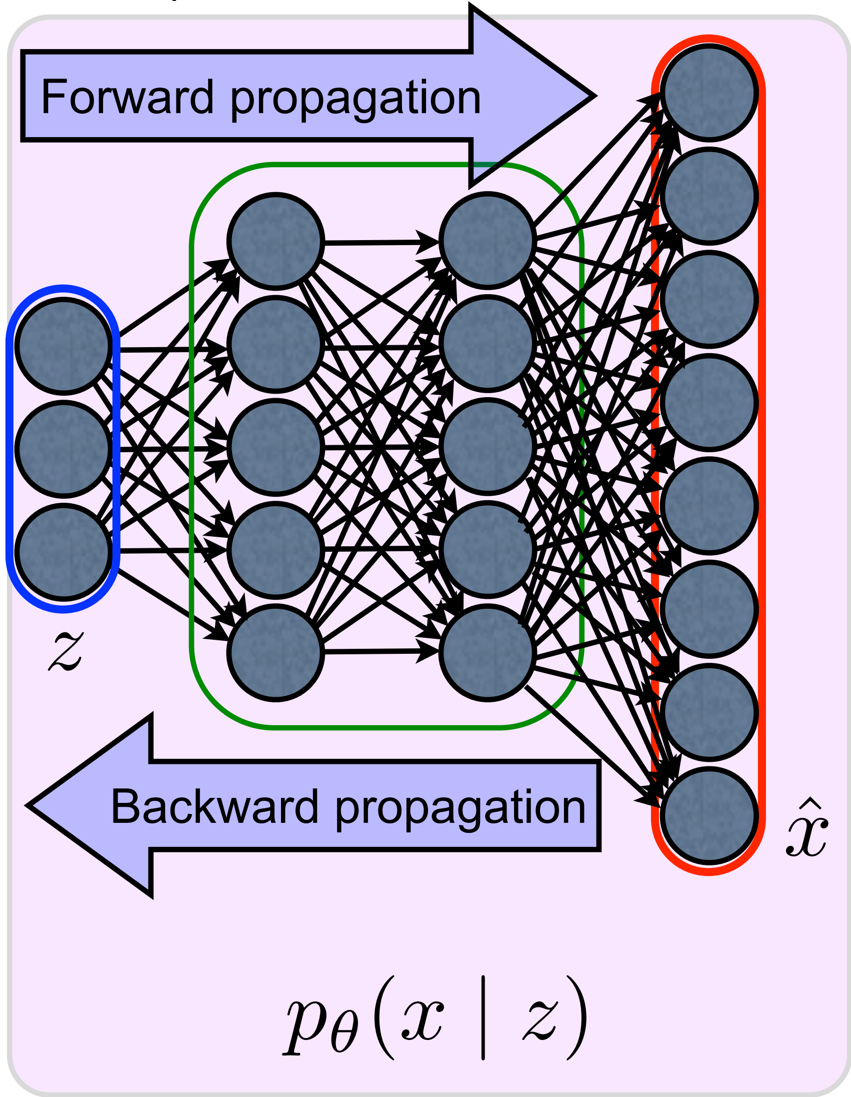
Leapfrog dynamics:

$$\boldsymbol{v}_{t+\frac{\epsilon}{2}} = \boldsymbol{v}_t - \frac{\epsilon}{2}\nabla_{\boldsymbol{z}}\left(\log p(\boldsymbol{x}, \boldsymbol{z}_t)\right)$$

$$\boldsymbol{z}_{t+\epsilon} = \boldsymbol{z}_t + \epsilon \boldsymbol{v}_{t+\frac{\epsilon}{2}}$$

$$\boldsymbol{v}_{t+\epsilon} = \boldsymbol{v}_{t+\frac{\epsilon}{2}} - \frac{\epsilon}{2}\nabla_{\boldsymbol{z}}\left(\log p(\boldsymbol{x}, \boldsymbol{z}_{t+\epsilon})\right)$$

Deep Generative Model:



Hamiltonian Variational Inference (HVI)

Fusing the VAE and HMC:

- **Central Idea**: Interpret the stochastic Markov chain (from HMC)

$$q(\boldsymbol{z} \mid \boldsymbol{x}) = q(\boldsymbol{z}_0 \mid \boldsymbol{x}) \prod_{t=1}^{T} q(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1}, \boldsymbol{x})$$

as a variational approximation in an expanded space.

- Consider $y = z_0, z_1, z_2, \ldots, z_{t-1}$ to be a set of auxiliary random variables.

- We obtain a new (lower) lower bound on the log-likelihood:

$$\begin{aligned}
\mathcal{L}_{\text{aux}} &= \mathbb{E}_{q(\boldsymbol{y}, \boldsymbol{z}_T \mid \boldsymbol{x})} \left[\log p(\boldsymbol{x}, \boldsymbol{z}_T) r(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{z}_T) - \log q(\boldsymbol{y}, \boldsymbol{z}_T \mid \boldsymbol{x})\right] \\
&= \mathcal{L} - \mathbb{E}_{q(\boldsymbol{z}_T \mid \boldsymbol{x})} \left\{ D_{\text{KL}} \left[q(\boldsymbol{y} \mid \boldsymbol{z}_T, \boldsymbol{x}) \| r(\boldsymbol{y} \mid \boldsymbol{z}_T, \boldsymbol{x}) \right] \right\} \\
&\leq \mathcal{L} \leq \log p(x)
\end{aligned}$$

where $r(\boldsymbol{y} \mid \boldsymbol{z}_T, \boldsymbol{x})$ is an auxiliary inference distribution (we choose it).

Hamiltonian Variational Inference (HVI)

- Assume the auxiliary inference distribution has a Markov structure:

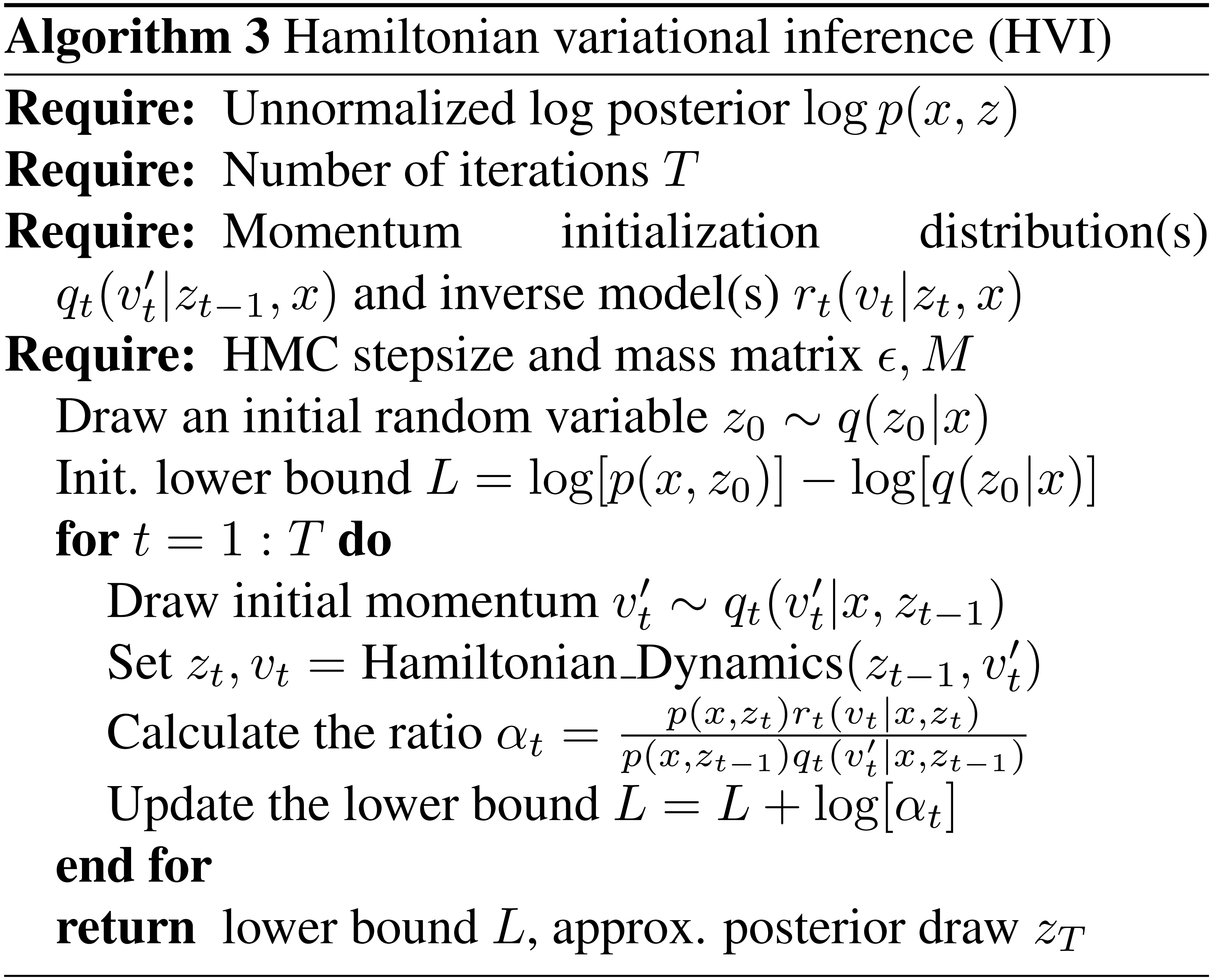
$$r(\boldsymbol{z}_0, \ldots, \boldsymbol{z}_{t-1} \mid \boldsymbol{x}, \boldsymbol{z}_T) = \prod_{t=1}^{T} r_t(\boldsymbol{z}_{t-1} \mid \boldsymbol{x}, \boldsymbol{z}_t)$$

- With this, lower bound becomes

$$\begin{aligned}\mathcal{L}_{\text{aux}} &= \mathbb{E}_{q(\boldsymbol{y}, \boldsymbol{z}_T \mid \boldsymbol{x})} \left[\log p(\boldsymbol{x}, \boldsymbol{z}_T) + \log \frac{r(\boldsymbol{z}_0, \ldots, \boldsymbol{z}_{T-1} \mid \boldsymbol{x}, \boldsymbol{z}_T)}{q(\boldsymbol{z}_0, \ldots, \boldsymbol{z}_T \mid \boldsymbol{x})} \right] \\ &= \mathbb{E}_{q(\boldsymbol{y}, \boldsymbol{z}_T \mid \boldsymbol{x})} \left[\log \frac{p(\boldsymbol{x}, \boldsymbol{z}_T)}{q(\boldsymbol{z}_0 \mid \boldsymbol{x})} + \sum_{t=1}^{T} \log \frac{r_t(\boldsymbol{z}_{t-1} \mid \boldsymbol{x}, \boldsymbol{z}_t)}{q_t(\boldsymbol{z}_t \mid \boldsymbol{x}, \boldsymbol{z}_{t-1})} \right]\end{aligned}$$

- This will work for any MCMC method. Specializing to HMC involves some details (like considering the velocity). See paper for details.

Hamiltonian Variational Inference (HVI)



Algorithm 3 Hamiltonian variational inference (HVI)

```
Require: Unnormalized log posterior $\log p(x, z)$
Require: Number of iterations $T$
Require: Momentum initialization distribution(s) $q_t(v'_t|z_{t-1}, x)$ and inverse model(s) $r_t(v_t|z_t, x)$
Require: HMC stepsize and mass matrix $\epsilon$, $M$
  Draw an initial random variable $z_0 \sim q(z_0|x)$
  Init. lower bound $L = \log[p(x, z_0)] - \log[q(z_0|x)]$
  for $t = 1 : T$ do
    Draw initial momentum $v'_t \sim q_t(v'_t|x, z_{t-1})$
    Set $z_t, v_t = \text{Hamiltonian\_Dynamics}(z_{t-1}, v'_t)$
    Calculate the ratio $\alpha_t = \frac{p(x,z_t)r_t(v_t|x,z_t)}{p(x,z_{t-1})q_t(v'_t|x,z_{t-1})}$
    Update the lower bound $L = L + \log[\alpha_t]$
  end for
  return lower bound $L$, approx. posterior draw $z_T$
```

HVI: Generative model of MNIST

Model	$\log p(x)$ $\leq -$	$\log p(x)$ $= -$
HVI + fully-connected VAE:		
Without inference network:		
5 leapfrog steps	90.86	87.16
10 leapfrog steps	87.60	85.56
With inference network:		
No leapfrog steps	94.18	88.95
1 leapfrog step	91.70	88.08
4 leapfrog steps	89.82	86.40
8 leapfrog steps	88.30	85.51
HVI + convolutional VAE:		
No leapfrog steps	86.66	83.20
1 leapfrog step	85.40	82.98
2 leapfrog steps	85.17	82.96
4 leapfrog steps	84.94	82.78
8 leapfrog steps	84.81	82.72
16 leapfrog steps	84.11	82.22
16 leapfrog steps, $n_h = 800$	83.49	81.94
From (Gregor et al., 2015):		
DBN 2hl		84.55
EoNADE		85.10
DARN 1hl	88.30	84.13
DARN 12hl	87.72	
DRAW	80.97	

The end.