10-418/10-618 Machine Learning for Structured Data

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Variational Autoencoders

Matt Gormley
Lecture 20
Nov. 14, 2022

Reminders

- **Homework 5: Variational Inference**
  - **Out: Fri, Nov 4**
  - **Due: Wed, Nov 16 at 11:59pm**
- **Homework 6: VAE + Structured SVM**
  - **Out: Wed, Nov 16**
  - **Due: Wed, Nov 30 at 11:59pm**

AUTOENCODERS

Unsupervised Pre-training

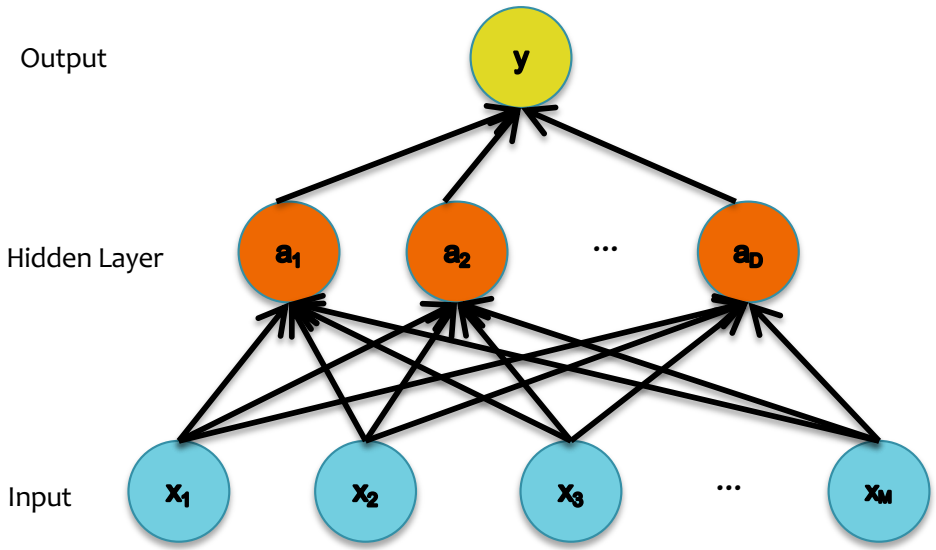
- **Idea: (Two Steps)**
  - Use supervised learning, but **pick a better starting point**
  - **Train each level** of the model in a **greedy** way

1. Unsupervised Pre-training
   - Use **unlabeled** data
   - Work bottom-up
     - Train hidden layer 1. Then fix its parameters.
     - Train hidden layer 2. Then fix its parameters.
     - …
     - Train hidden layer n. Then fix its parameters.
2. Supervised Fine-tuning
   - Use **labeled** data to train following "Idea #1"
   - Refine the features by backpropagation so that they become tuned to the end-task

Unsupervised Pre-training

Unsupervised pre-training of the first layer:

- What should it predict?
- What else do we observe?
- **The input!**

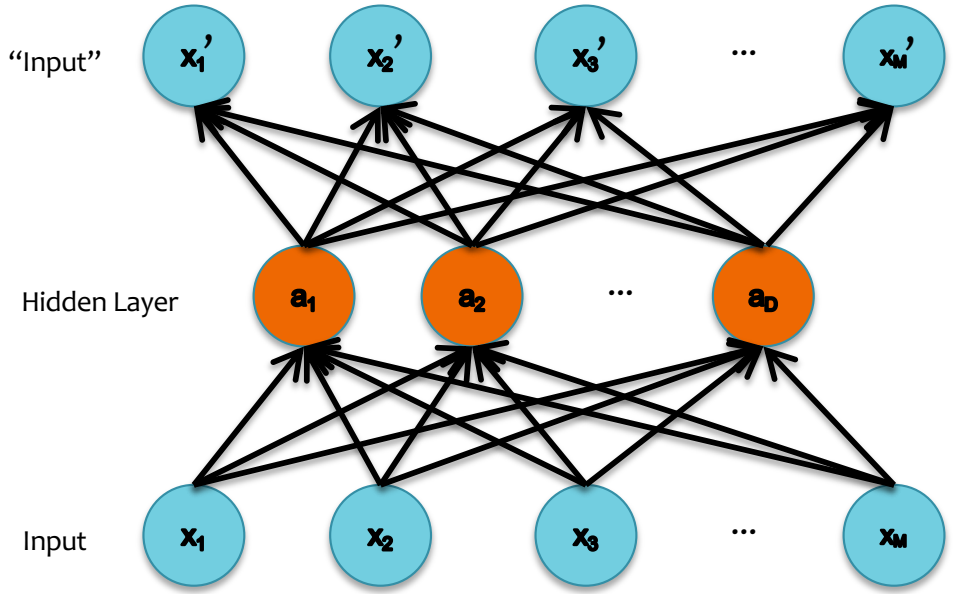


Auto-Encoders

Unsupervised pre-training of the first layer:

- What should it predict?
- What else do we observe?
- **The input!**

This topology defines an Auto-encoder.



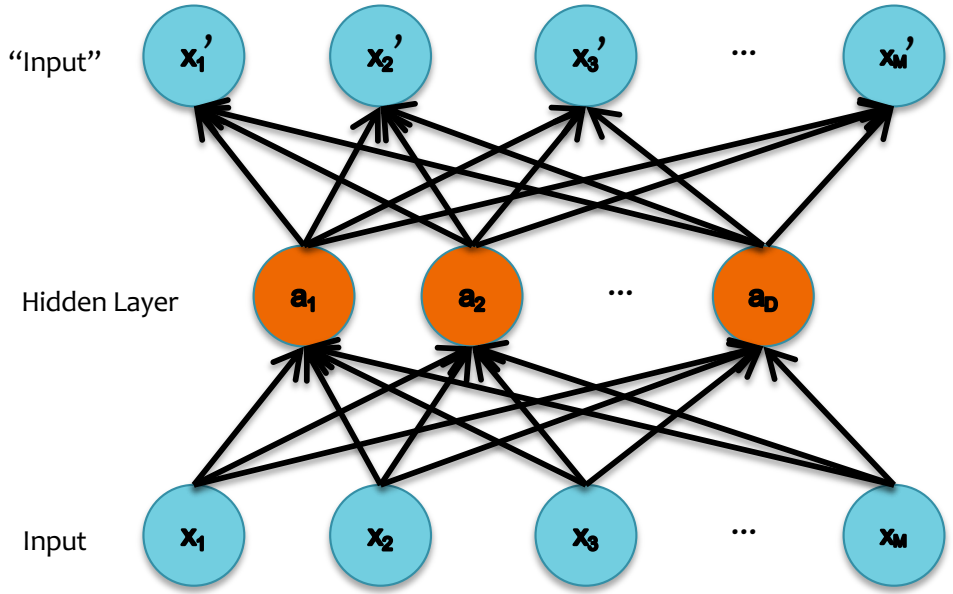
Auto-Encoders

Key idea: Encourage z to give small reconstruction error:

- x' is the *reconstruction* of x
- Loss = $|| x - \text{DECODER}(\text{ENCODER}(x)) ||^2$
- Train with the same backpropagation algorithm for 2-layer Neural Networks with x_m as both input and output.

DECODER: $x' = h(W'z)$

ENCODER: $z = h(Wx)$

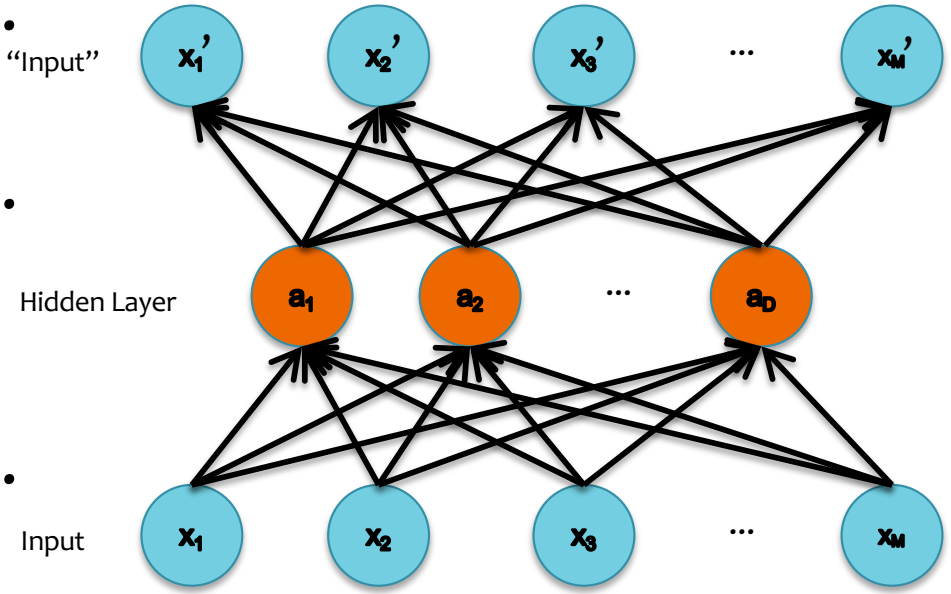


Slide adapted from Raman Arora

Unsupervised Pre-training

Unsupervised pre-training

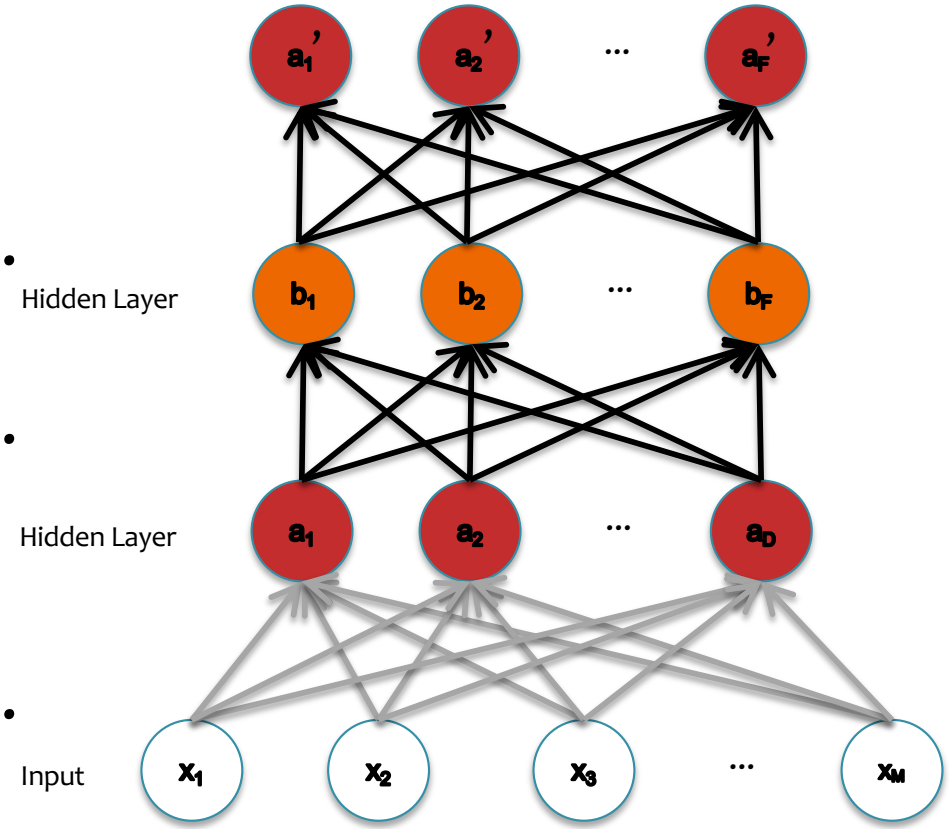
- Work bottom-up
  - Train hidden layer 1. Then fix its parameters.
  - Train hidden layer 2. Then fix its parameters.
  - …
  - Train hidden layer n. Then fix its parameters.



Unsupervised Pre-training

Unsupervised pre-training

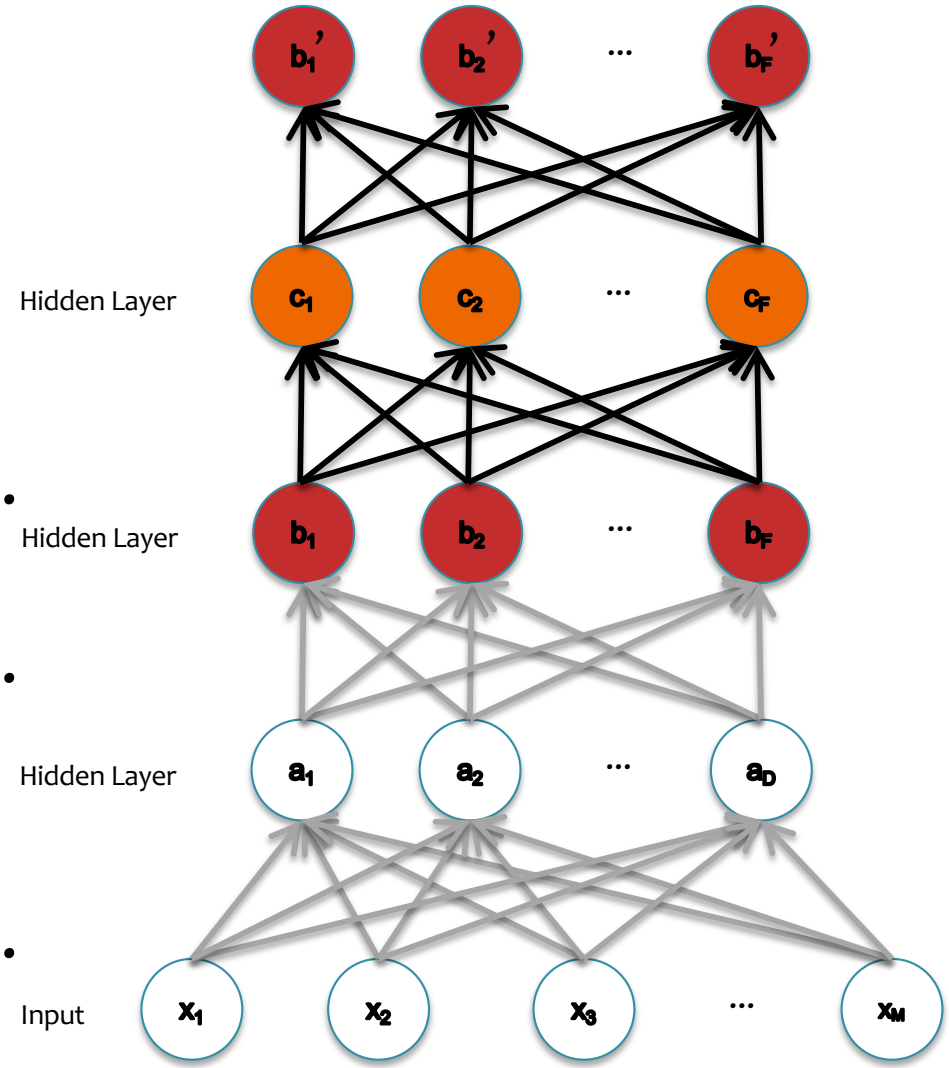
- Work bottom-up
  - Train hidden layer 1. Then fix its parameters.
  - Train hidden layer 2. Then fix its parameters.
  - …
  - Train hidden layer n. Then fix its parameters.



Unsupervised Pre-training

Unsupervised pre-training

- Work bottom-up
  - Train hidden layer 1. Then fix its parameters.
  - Train hidden layer 2. Then fix its parameters.
  - …
  - Train hidden layer n. Then fix its parameters.



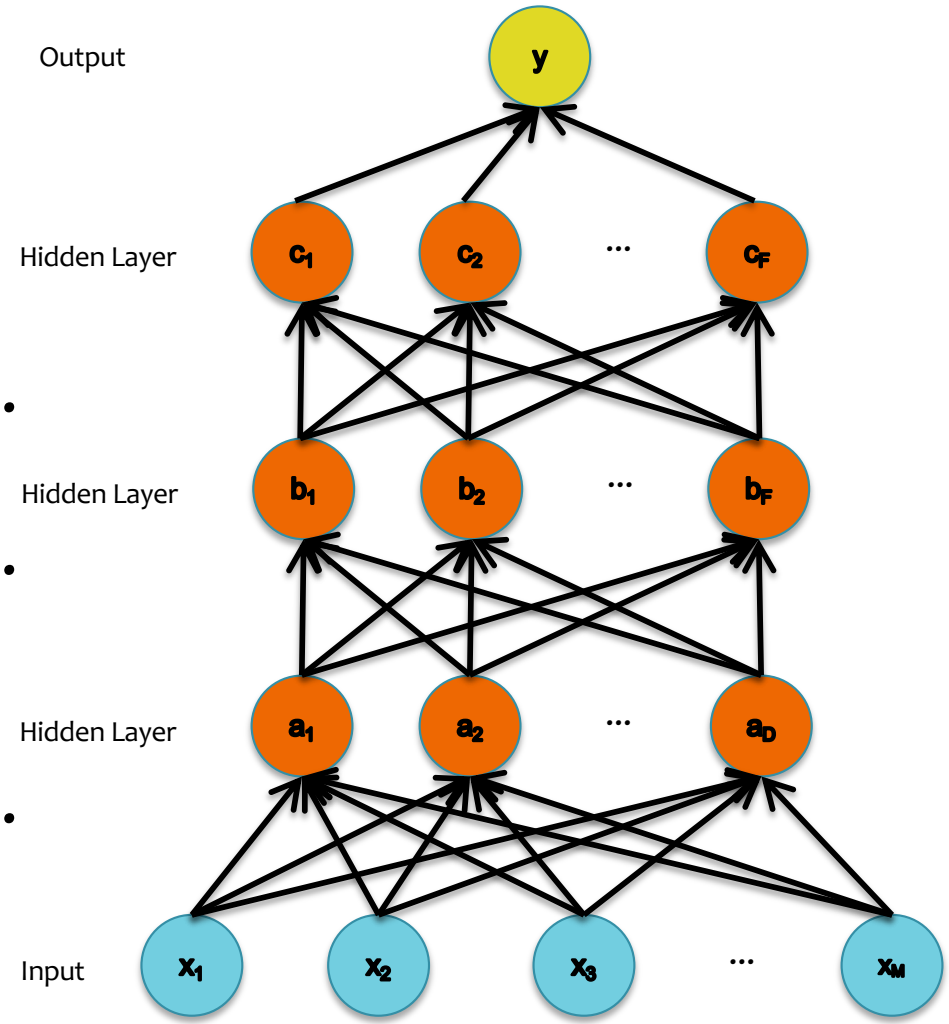
Unsupervised Pre-training

Unsupervised pre-training

- Work bottom-up
  - Train hidden layer 1. Then fix its parameters.
  - Train hidden layer 2. Then fix its parameters.
  - …
  - Train hidden layer n. Then fix its parameters.

Supervised fine-tuning

Backprop and update all parameters



Deep Network Training

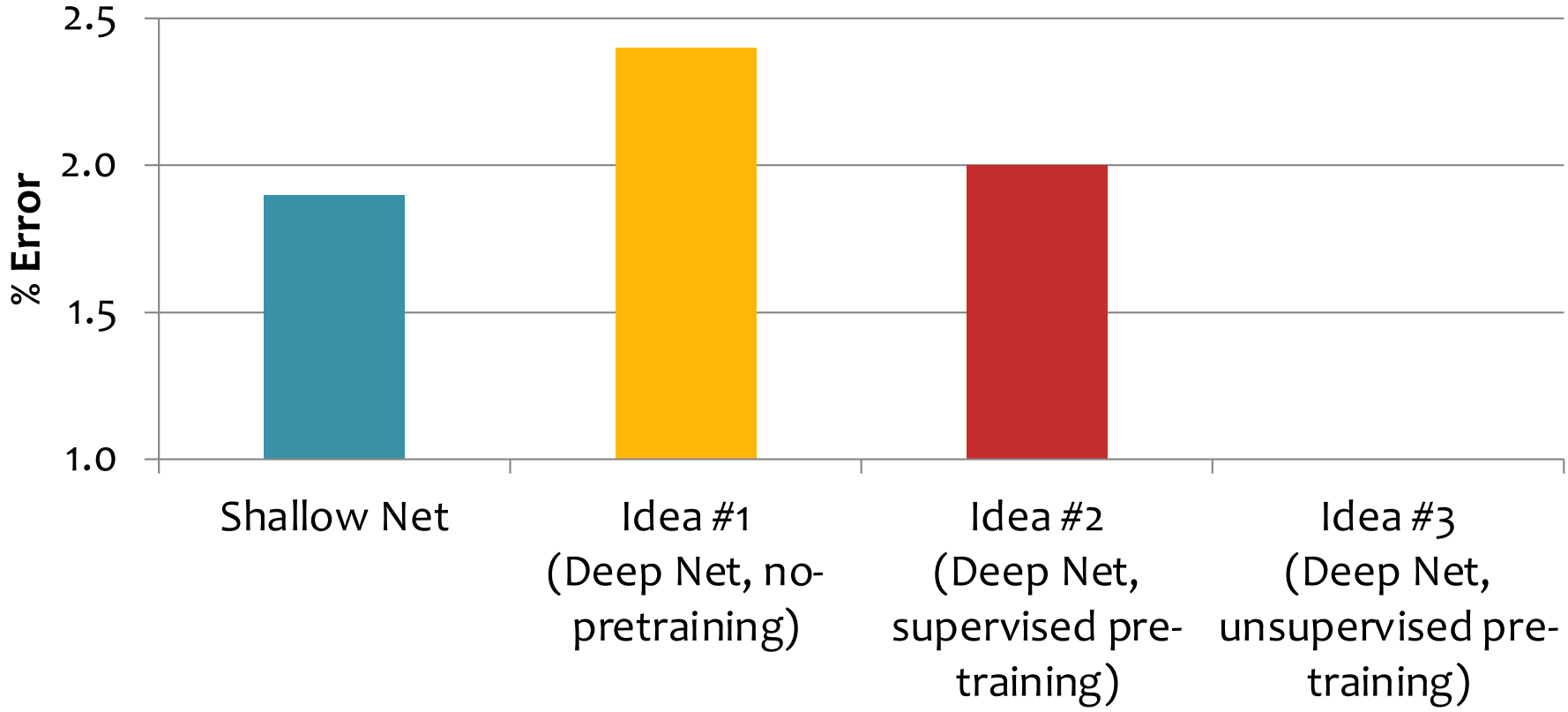
- **Idea #1:**
  1. Supervised fine-tuning only

- **Idea #2:**
  1. Supervised layer-wise pre-training
  2. Supervised fine-tuning

- **Idea #3:**
  1. Unsupervised layer-wise pre-training
  2. Supervised fine-tuning

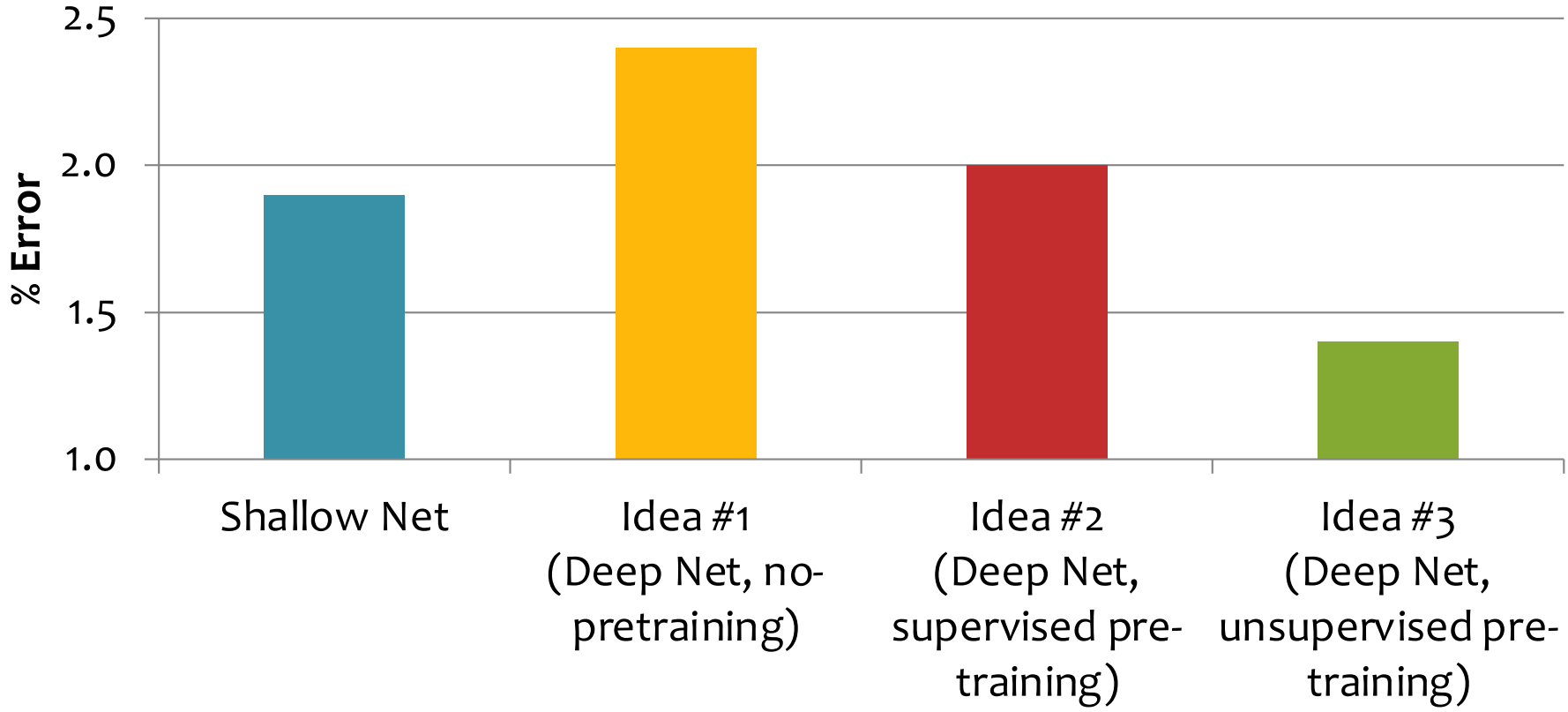
Comparison on MNIST

- Results from Bengio et al. (2006) on MNIST digit classification task
- Percent error (lower is better)



Comparison on MNIST

- Results from Bengio et al. (2006) on MNIST digit classification task
- Percent error (lower is better)



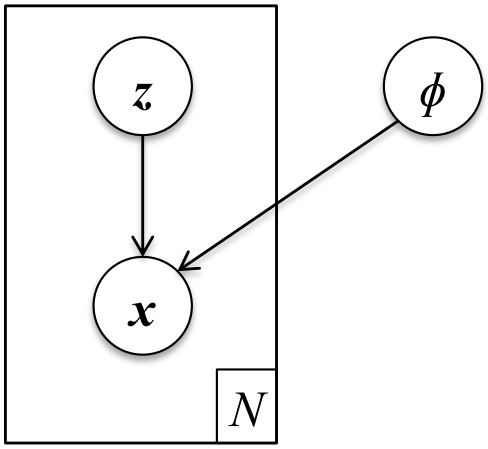
VARIATIONAL AUTOENCODERS

Why VAEs?

- **Autoencoders:**
  - learn a low dimensional representation of the input, but hard to work with as a generative model
  - one of the key limitations of autoencoders is that we have no way of **sampling** from them!
- **Variational autoencoders (VAEs)**
  - by contrast learn a continuous latent space that is **easy to sample from**!
  - can **generate** new data (e.g. images) by sampling from the learned generative model

Variational Autoencoders

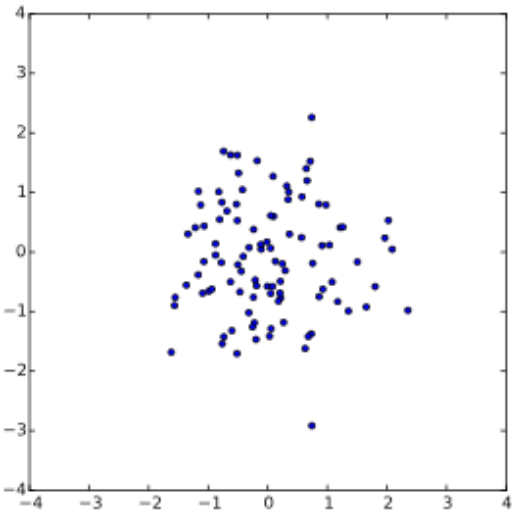
$p_\phi(\mathbf{x}, \mathbf{z})$



$z \sim \text{Gaussian}(0, I)$

Graphical Model Perspective

- The DGM diagram shows that the VAE model is quite simple as a graphical model (ignoring the neural net details that give rise to **x**)
- Sampling from the model is easy:
  - Consider a DGM where $\mathbf{x} = g_\phi(\mathbf{z}/10 + \mathbf{z}/||\mathbf{z}||) + \phi$ (i.e. we don't use parameters ϕ)
  - Then we can draw samples of **z** and directly convert them to values **x**
- **Key idea of VAE:** define $g_\phi(z)$ as a neural net and learn ϕ from data



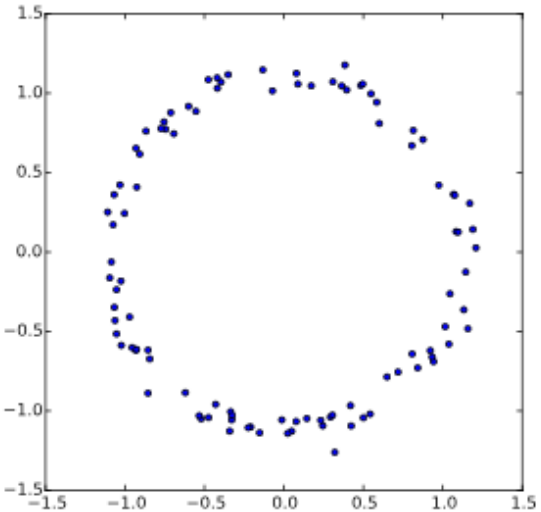


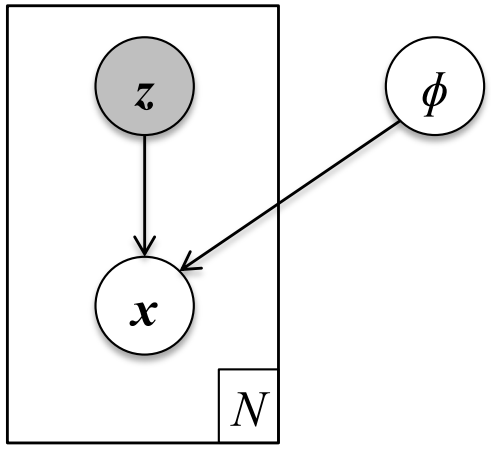
Figure from Doersch (2016)

Variational Autoencoders

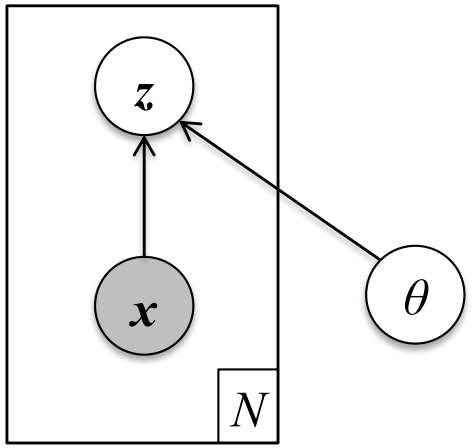
Neural Network Perspective

- We can view a variational autoencoder (VAE) as an autoencoder consisting of two neural networks
- VAEs (as encoders) define two distributions:
  - **encoder**: $q_\theta(\mathbf{z} \mid \mathbf{x})$
  - **decoder**: $p_\phi(\mathbf{x} \mid \mathbf{z})$
- Parameters θ and ϕ are neural network parameters (i.e. θ are not the variational parameters)

$p_\phi(\mathbf{x} \mid \mathbf{z})$



$q_\theta(\mathbf{z} \mid \mathbf{x})$

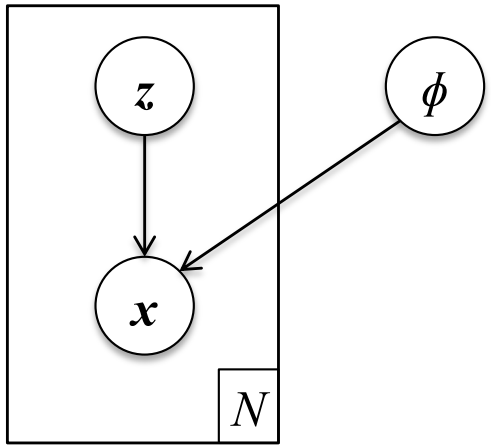


Variational Autoencoders

Graphical Model Perspective

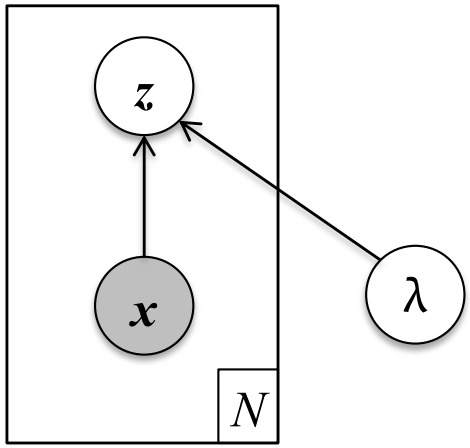
- We can also view the VAE from the perspective of variational inference
- In this case we have two distributions:
  - **model**: $p_\phi(\mathbf{z} \mid \mathbf{x})$
  - **variational approximation**: $q_{\lambda=f(x;\,\theta)}(\mathbf{z} \mid \mathbf{x})$
- We have the same model parameters ϕ
- The variational parameters λ are a function of NN parameters θ

$$p_\phi(\mathbf{x}, \mathbf{z})$$



$$\boldsymbol{z} \sim \text{Gaussian}(0, I)$$

$$q_\lambda(\mathbf{z} \mid \mathbf{x})$$



$$\lambda = f(x; \theta)$$

Variational Autoencoders

Whiteboard

- Variational Autoencoder = VAE
- VAE as a Probability Model
- Parameterizing the VAE with Neural Nets
- Variational EM for VAEs

Reparameterization Trick

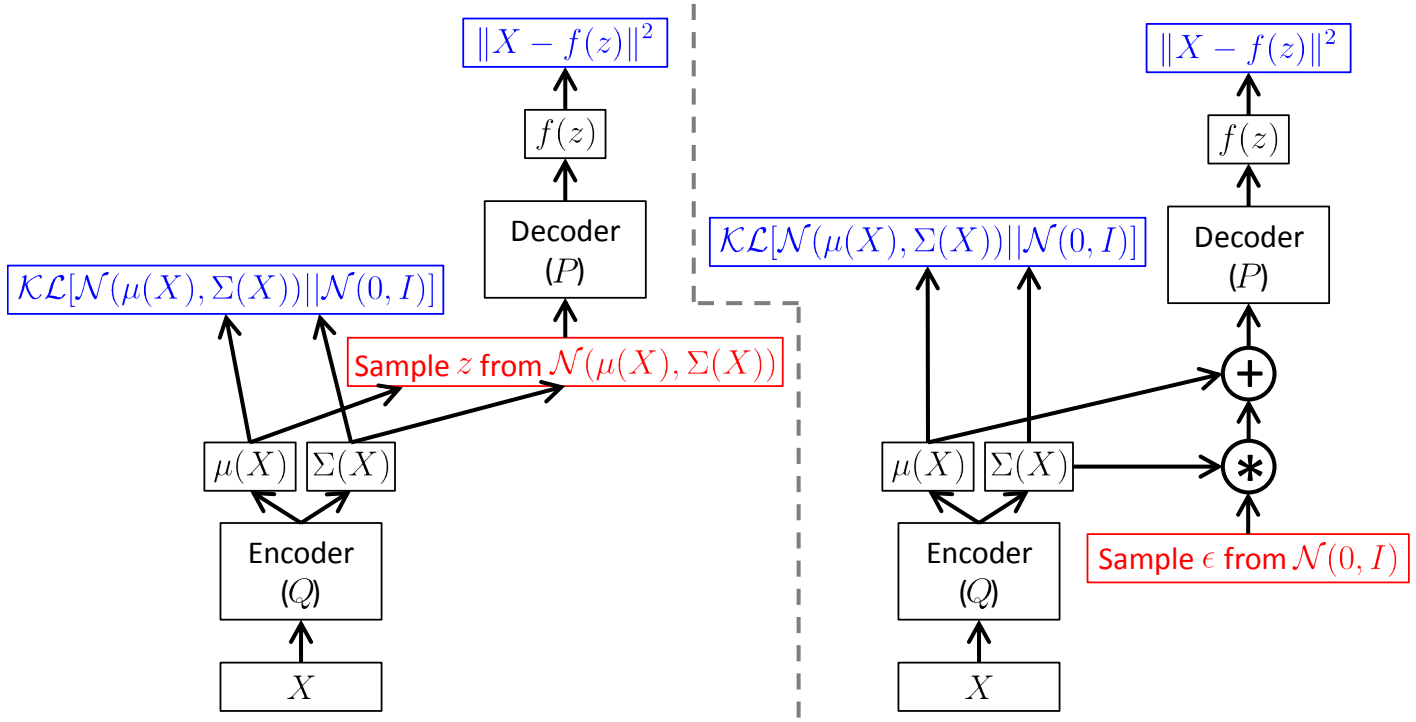


Figure 4: A training-time variational autoencoder implemented as a feedforward neural network, where $P(X|z)$ is Gaussian. Left is without the "reparameterization trick", and right is with it. Red shows sampling operations that are non-differentiable. Blue shows loss layers. The feedforward behavior of these networks is identical, but backpropagation can be applied only to the right network.

Figure from Doersch (2016)

VAE RESULTS

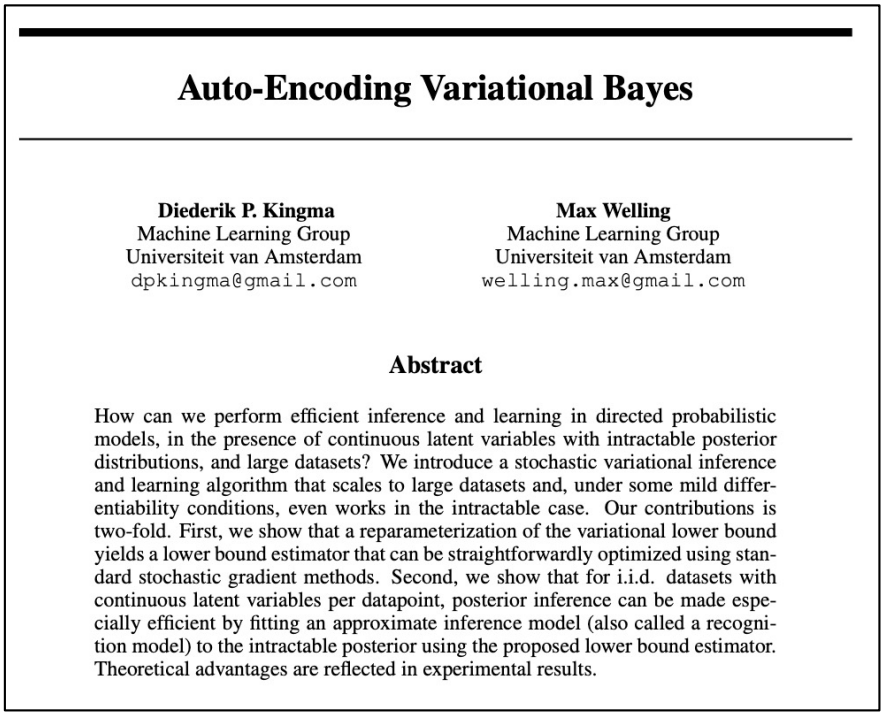
VAEs for Image Generation

Kingma & Welling (2014)

- introduced VAEs
- applied to image generation

Model

- $p_\phi(\mathbf{z}) \sim N(\mathbf{z}; 0, \mathbf{I})$
- $p_\phi(\mathbf{x} \mid \mathbf{z})$ is a multivariate Gaussian with mean and variance computed by an MLP, fully connected neural network with a single hidden layer with parameters ϕ
- $q_\theta(\mathbf{z} \mid \mathbf{x})$ is a multivariate Gaussian with diagonal covariance structure and with mean and variance computed by an MLP with parameters θ



Auto-Encoding Variational Bayes

Diederik P. Kingma
Machine Learning Group
Universiteit van Amsterdam
dpkingma@gmail.com

Max Welling
Machine Learning Group
Universiteit van Amsterdam
welling.max@gmail.com

Abstract

How can we perform efficient inference and learning in directed probabilistic models, in the presence of continuous latent variables with intractable posterior distributions, and large datasets? We introduce a stochastic variational inference and learning algorithm that scales to large datasets and, under some mild differentiability conditions, even works in the intractable case. Our contributions is two-fold. First, we show that a reparameterization of the variational lower bound yields a lower bound estimator that can be straightforwardly optimized using standard stochastic gradient methods. Second, we show that for i.i.d. datasets with continuous latent variables per datapoint, posterior inference can be made especially efficient by fitting an approximate inference model (also called a recognition model) to the intractable posterior using the proposed lower bound estimator. Theoretical advantages are reflected in experimental results.

Figure from Kingma & Welling (2014)

VAEs for Image Generation

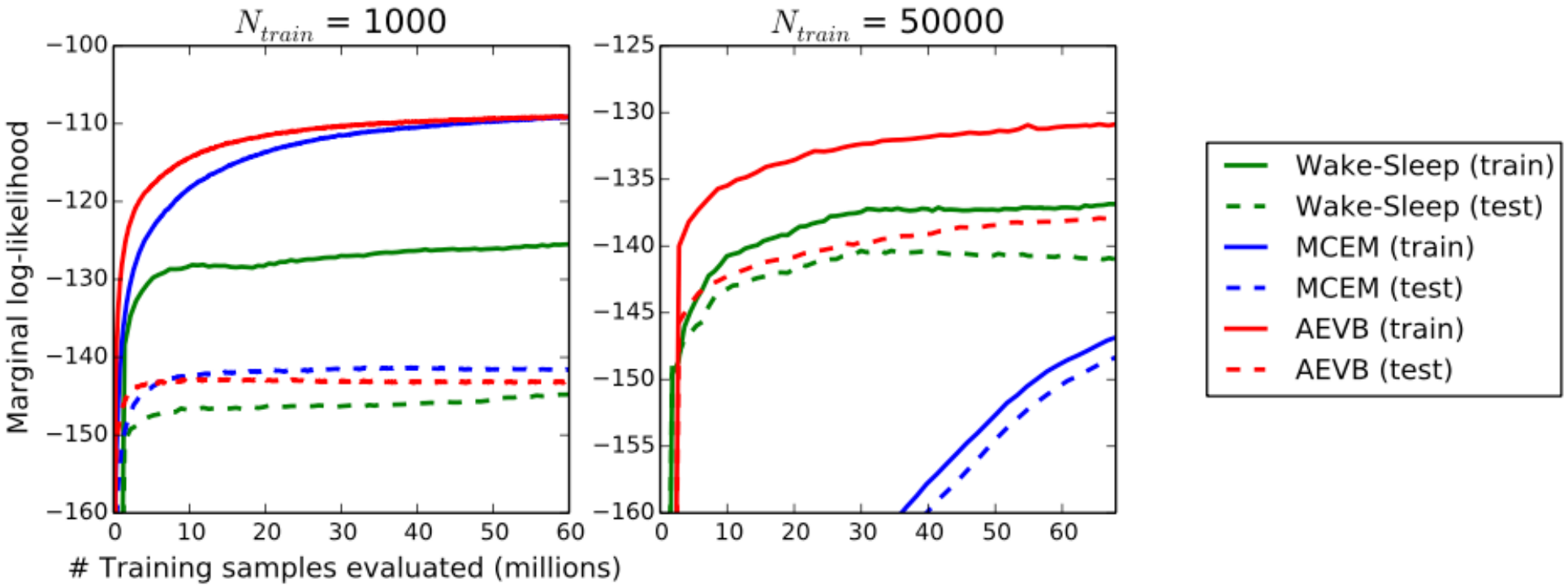
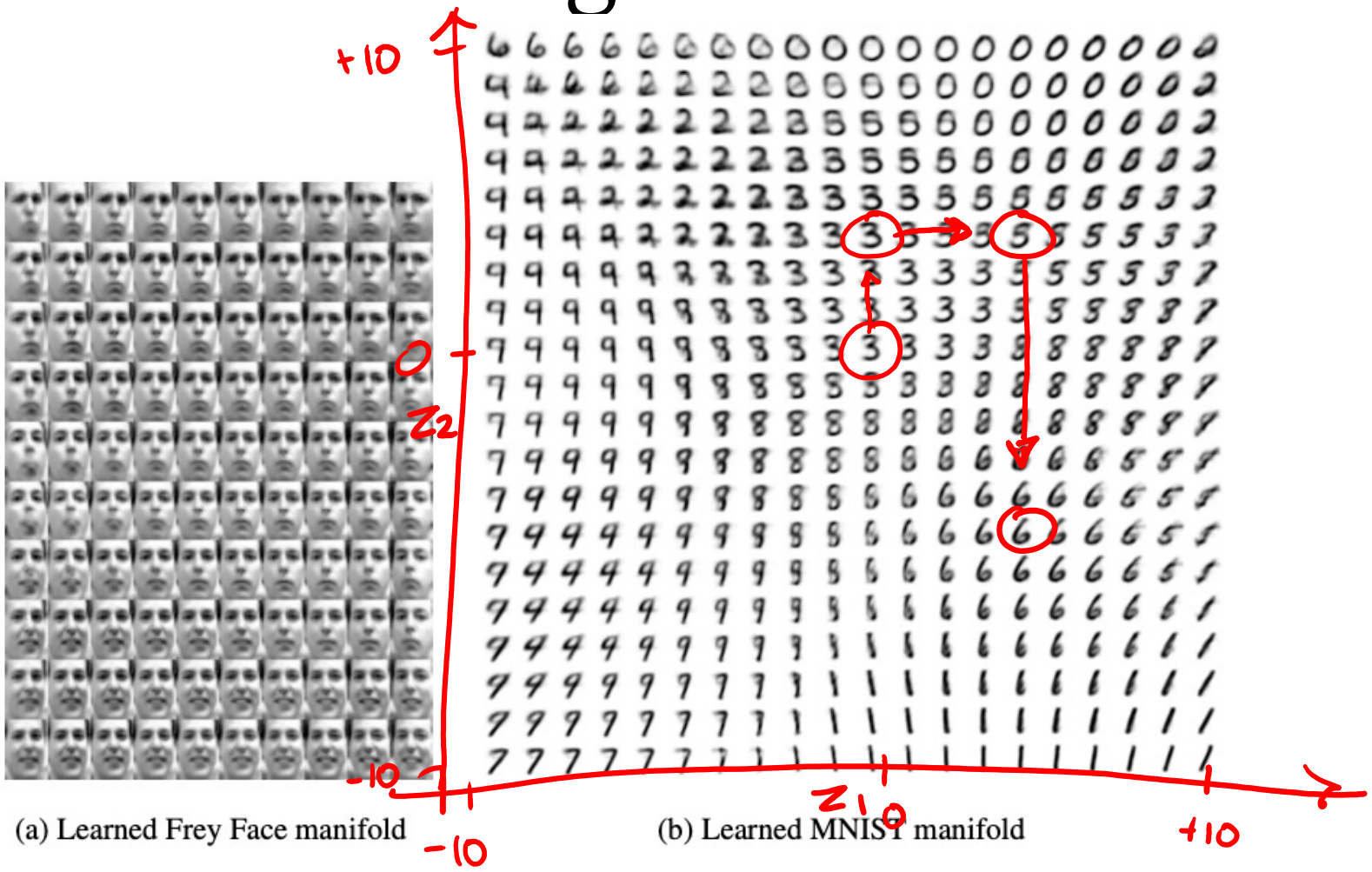


Figure 3: Comparison of AEVB to the wake-sleep algorithm and Monte Carlo EM, in terms of the estimated marginal likelihood, for a different number of training points. Monte Carlo EM is not an on-line algorithm, and (unlike AEVB and the wake-sleep method) can't be applied efficiently for the full MNIST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

Figure from Kingma & Welling (2014)

VAEs for Image Generation

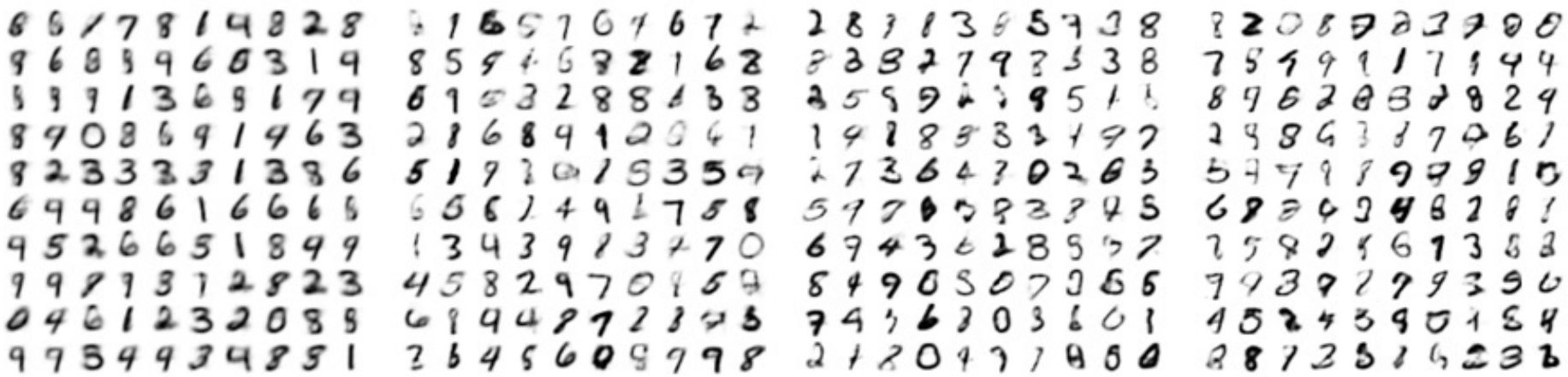


(a) Learned Frey Face manifold (b) Learned MNIST manifold

Figure 4: Visualisations of learned data manifold for generative models with two-dimensional latent space, learned with AEVB. Since the prior of the latent space is Gaussian, linearly spaced coordinates on the unit square were transformed through the inverse CDF of the Gaussian to produce values of the latent variables $\mathbf{z}$. For each of these values $\mathbf{z}$, we plotted the corresponding generative $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ with the learned parameters $\boldsymbol{\theta}$.

Figure from Kingma & Welling (2014)

VAEs for Image Generation



(a) 2-D latent space (b) 5-D latent space (c) 10-D latent space (d) 20-D latent space

Figure 5: Random samples from learned generative models of MNIST for different dimensionalities of latent space.

Figure from Kingma & Welling (2014)

VAEs for Text Generation

Bowman et al. (2015)

- example of an application of VAEs to discrete data
- built on the sequence-to-sequence framework:
  - input is read in by an LSTM
  - output is generated by an LSTM-LM

Model

- $p_\phi(\mathbf{z}) \sim N(\mathbf{z}; 0, \mathbf{I})$
- $p_\phi(\mathbf{x} \mid \mathbf{z})$ is an LSTM Language Model with parameters ϕ
- $q_\theta(\mathbf{z} \mid \mathbf{x})$ is a multivariate Gaussian with mean and variance computed by an LSTM with parameters θ

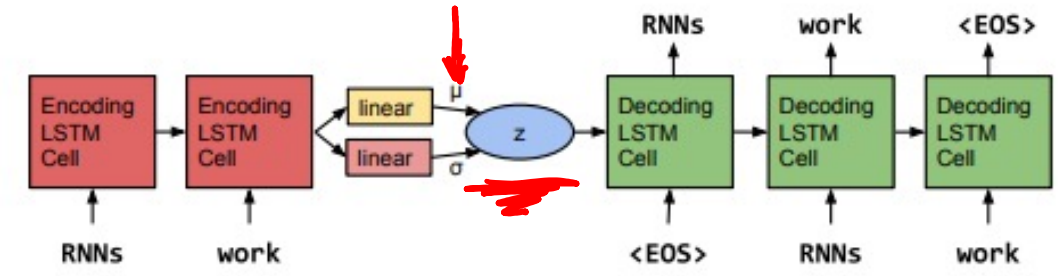


Figure 1: The core structure of our variational autoencoder language model. Words are represented using a learned dictionary of embedding vectors.

Figure from Bowman et al. (2015)

VAEs for Text Generation

INPUT	**we looked out at the setting sun .**	**i went to the kitchen .**	**how are you doing ?**
MEAN	*they were laughing at the same time .*	*i went to the kitchen .*	*what are you doing ?*
SAMP. 1	*ill see you in the early morning .*	*i went to my apartment .*	*“ are you sure ?*
SAMP. 2	*i looked up at the blue sky .*	*i looked around the room .*	*what are you doing ?*
SAMP. 3	*it was down on the dance floor .*	*i turned back to the table .*	*what are you doing ?*

Table 7: Three sentences which were used as inputs to the VAE, presented with greedy decodes from the mean of the posterior distribution, and from three samples from that distribution.

“ i want to talk to you . ”
“i want to be with you . ”
“i do n’t want to be with you . ”
i do n’t want to be with you .
she did n’t want to be with him .
he was silent for a long moment .
he was silent for a moment .
it was quiet for a moment .
it was dark and cold .
there was a pause .
it was my turn .

Table 8: Paths between pairs of random points in VAE space: Note that intermediate sentences are grammatical, and that topic and syntactic structure are usually locally consistent.

Figure from Bowman et al. (2015)

VQ-VAE

- Vector Quantized VAE (VQ-VAE) learns a continuous codebook, but the encoder outputs discrete codes
- Decoder takes a code and generates a sample conditioned on it

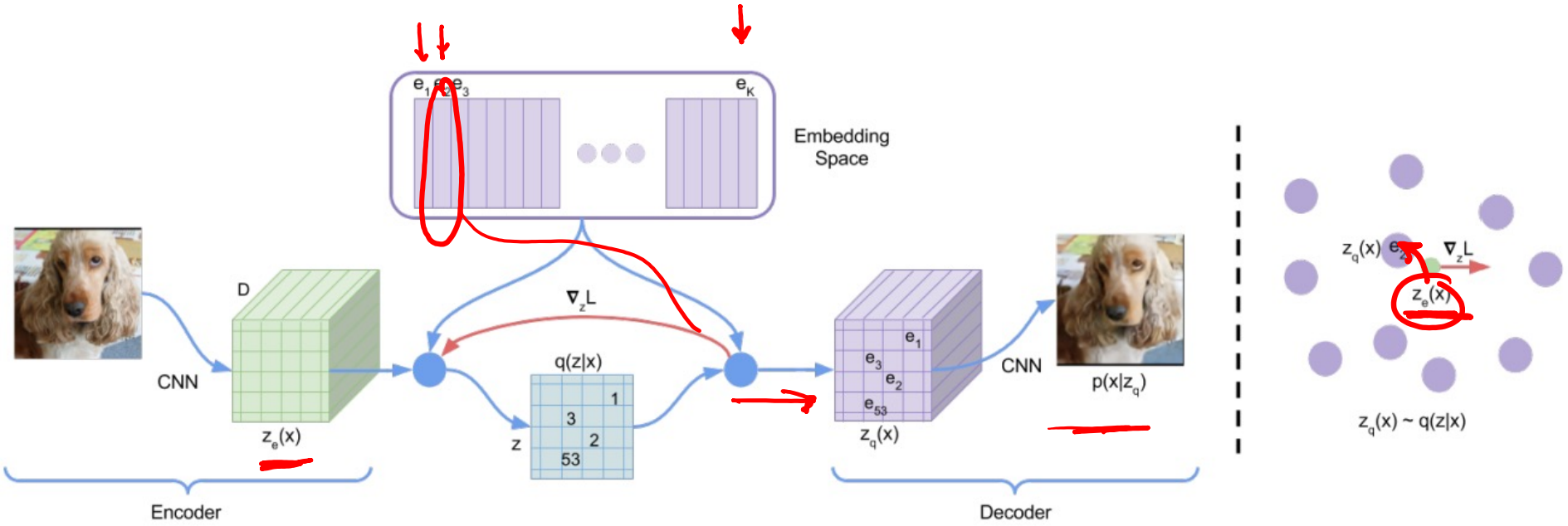


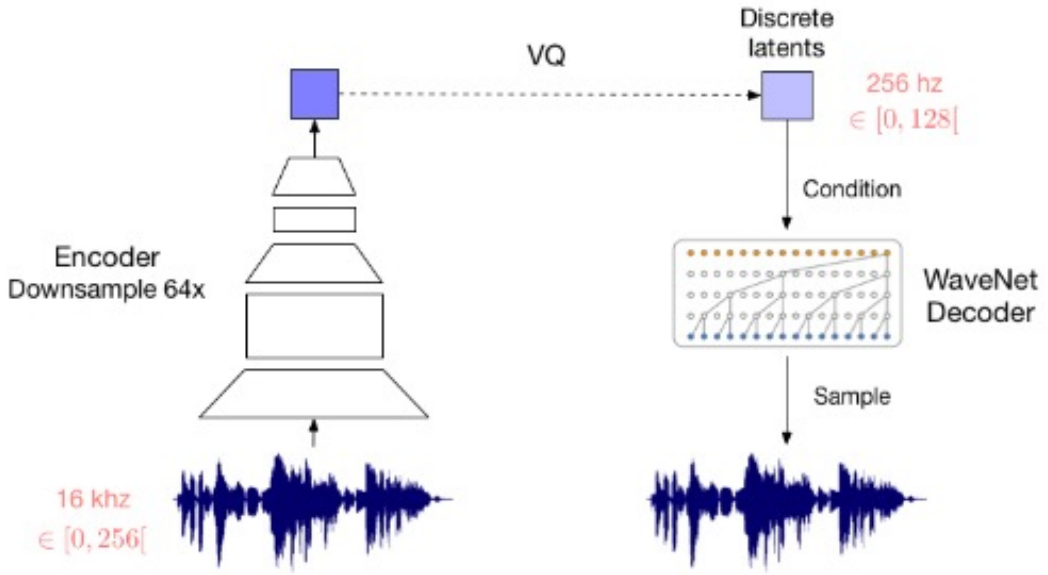
Figure 1: Left: A figure describing the VQ-VAE. Right: Visualisation of the embedding space. The output of the encoder $z(x)$ is mapped to the nearest point e_2. The gradient $\nabla_z L$ (in red) will push the encoder to change its output, which could alter the configuration in the next forward pass.

Figure from van den Oord et al. (2018)

VQ-VAE

- Vector Quantized VAE (VQ-VAE) learns a continuous codebook, but the encoder outputs discrete codes
- Decoder takes a code and generates a sample conditioned on it

Example: Generating Audio

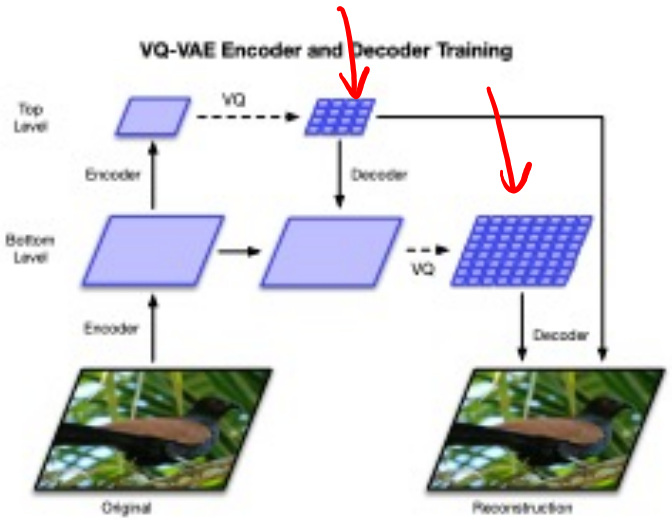


https://avdnoord.github.io/homepage/vqvae

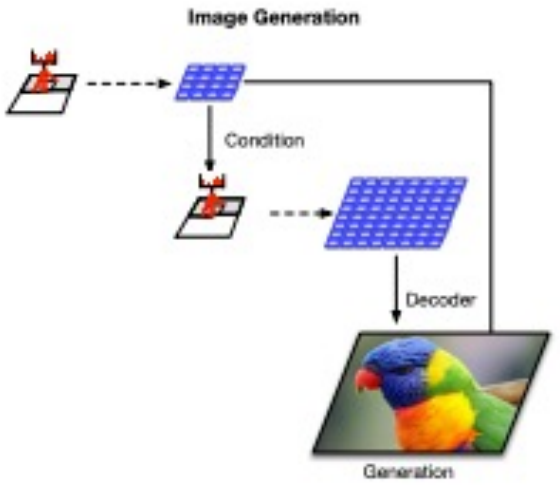
Figure from van den Oord et al. (2018)

VQ-VAE

- VQ-VAE-2 extended the original idea by learning two levels (bottom and top) and a strong prior over the latent space
- Samples from this new model can be convincing even at high-fidelity



(a) Overview of the architecture of our hierarchical VQ-VAE. The encoders and decoders consist of deep neural networks. The input to the model is a 256×256 image that is compressed to quantized latent maps of size 64×64 and 32×32 for the *bottom* and *top* levels, respectively. The decoder reconstructs the image from the two latent maps.



(b) Multi-stage image generation. The top-level PixelCNN prior is conditioned on the class label, the bottom level PixelCNN is conditioned on the class label as well as the first level code. Thanks to the feed-forward decoder, the mapping between latents to pixels is fast. (The example image with a parrot is generated with this model).

Figure from Razavi et al. (2019)

- VQ-VAE-2 extended the original idea by learning two levels (bottom and top) and a strong prior over the latent space
- Samples from this new model can be convincing even at high-fidelity

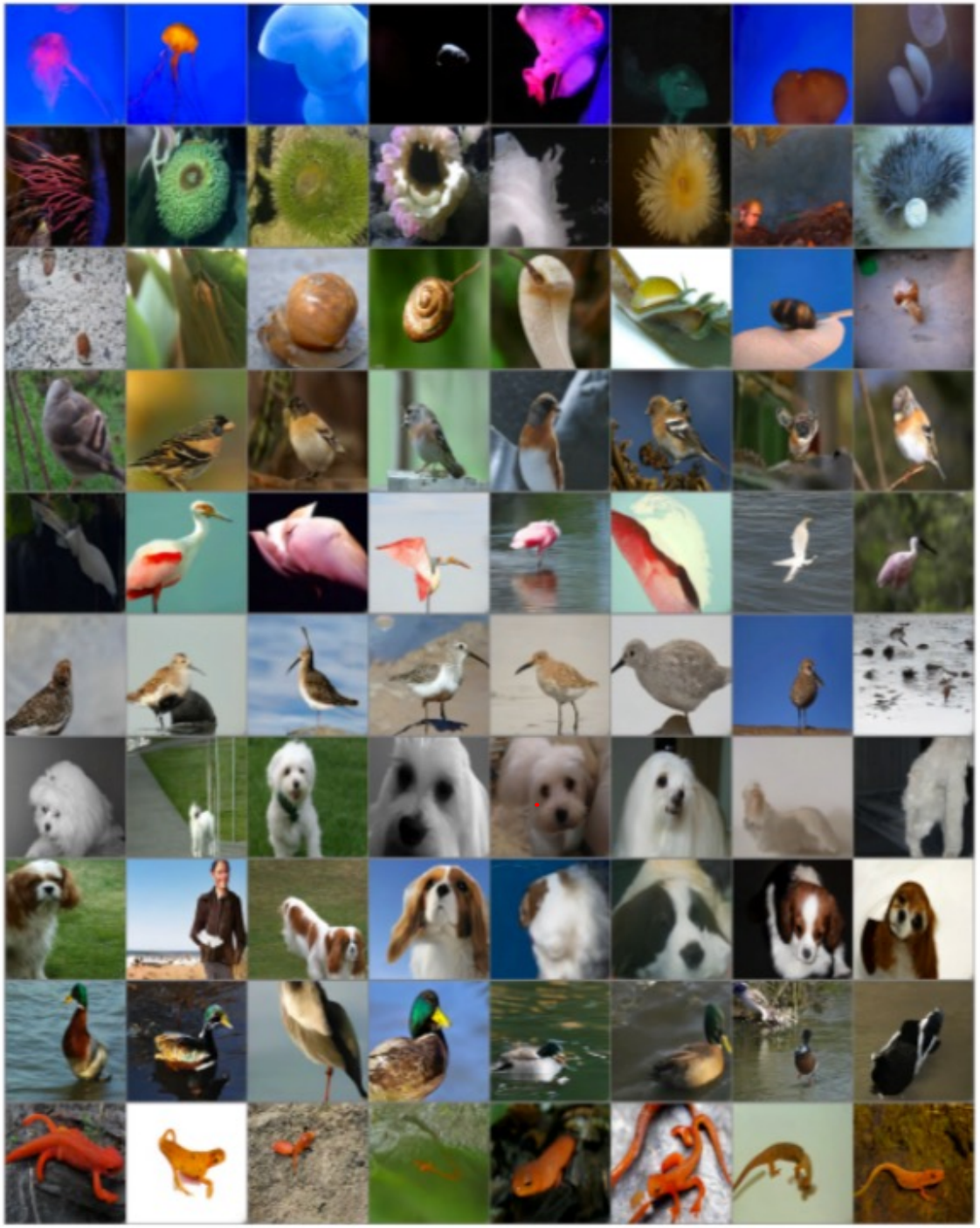


Figure 4: Class conditional random samples. Classes from the top row are: 108 sea anemone, 109 brain coral, 114 slug, 11 goldfinch, 130 flamingo, 141 redshank, 154 Pekinese, 157 papillon, 97 drake, and 28 spotted salamander.

Figure from Razavi et al. (2019)

- VQ-VAE-2 extended the original idea by learning two levels (bottom and top) and a strong prior over the latent space
- Samples from this new model can be convincing even at high-fidelity

Figure from Razavi et al. (2019)

- VQ-VAE-2 extended the original idea by learning two levels (bottom and top) and a strong prior over the latent space
- Samples from this new model can be convincing even at high-fidelity

Figure from Razavi et al. (2019)

- VQ-VAE-2 extended the original idea by learning two levels (bottom and top) and a strong prior over the latent space
- Samples from this new model can be convincing even at high-fidelity

Figure from Razavi et al. (2019)