Generative Models and Neural Variational Inference

Alexander G. Ororbia II

Neural Networks & Machine Learning

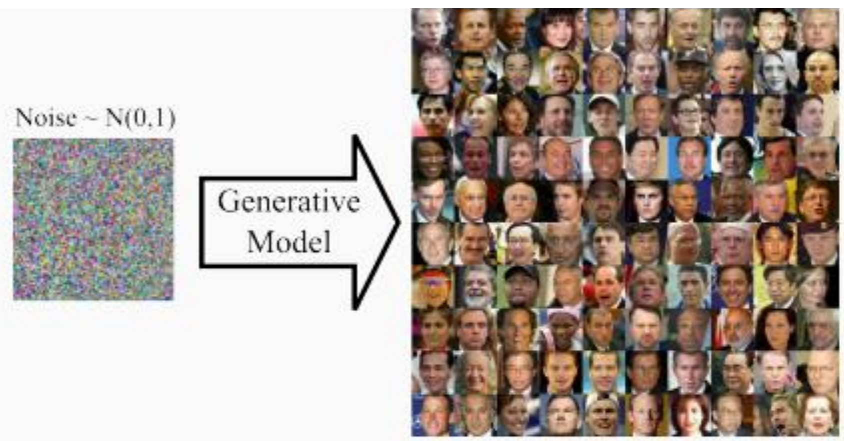
CSCI-736

2/16/2023

Generative Modeling & Sampling

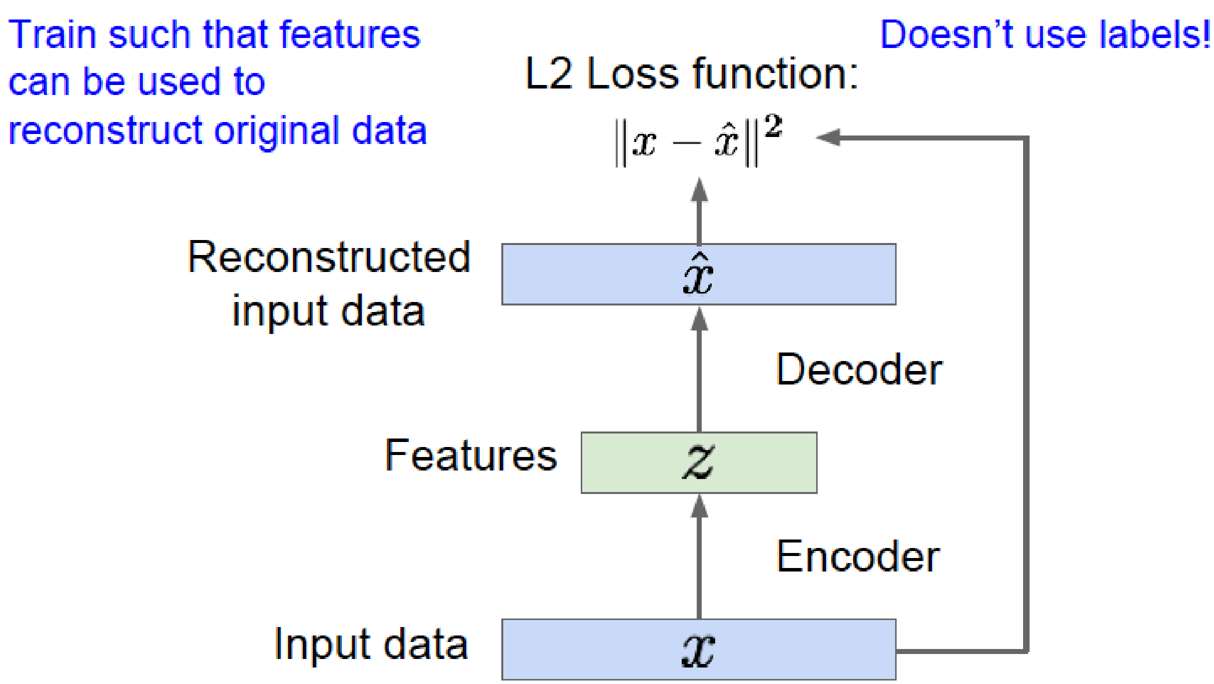
- **Task:** Given a dataset of images {X1,X2...} can we learn the distribution of X?
- Typically generative models implies modelling P(X).
  - **Very limited, given an image the model outputs a probability**
- **More Interested** in models which we can **sample** from.
  - Can generate random examples that follow the distribution of P(X).

Generative Adversarial Network:



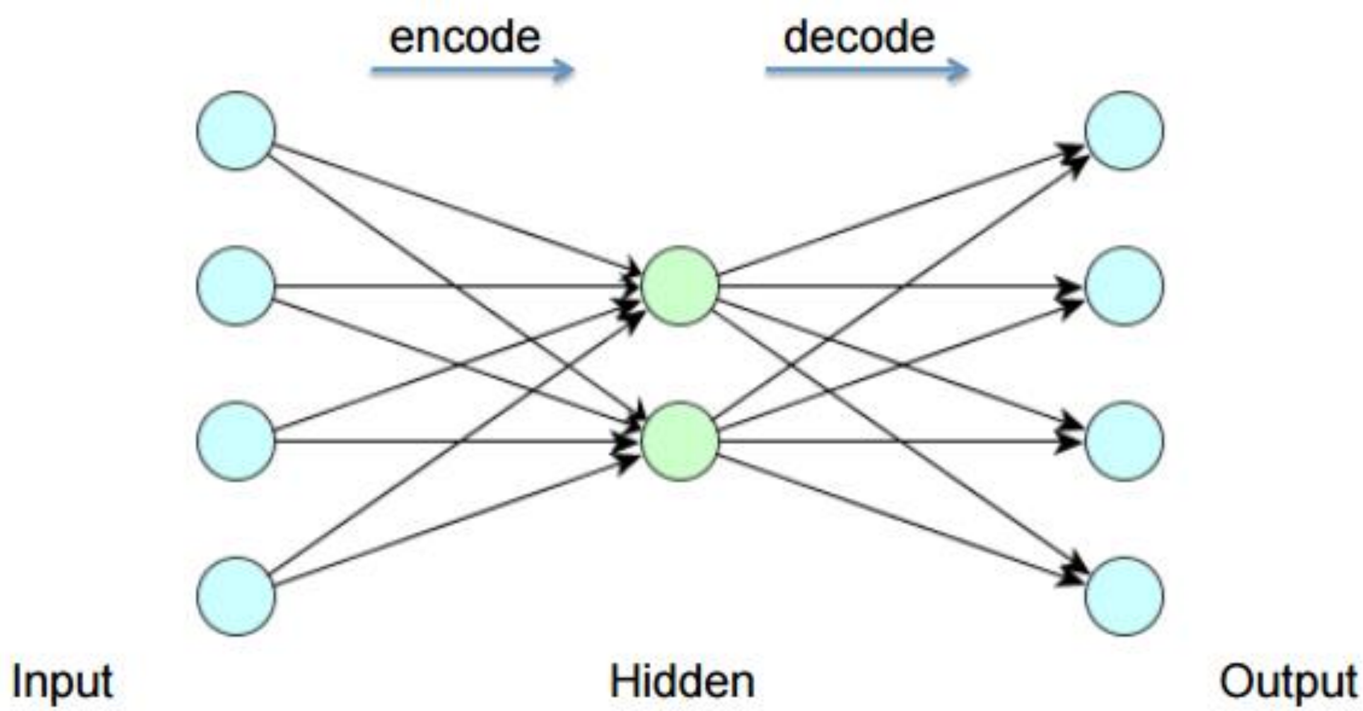
- Pro: Do not have to explicitly specify a form on P(X|z), z is the latent space.
- Con: Given a desired image, difficult to map back to the latent variable.

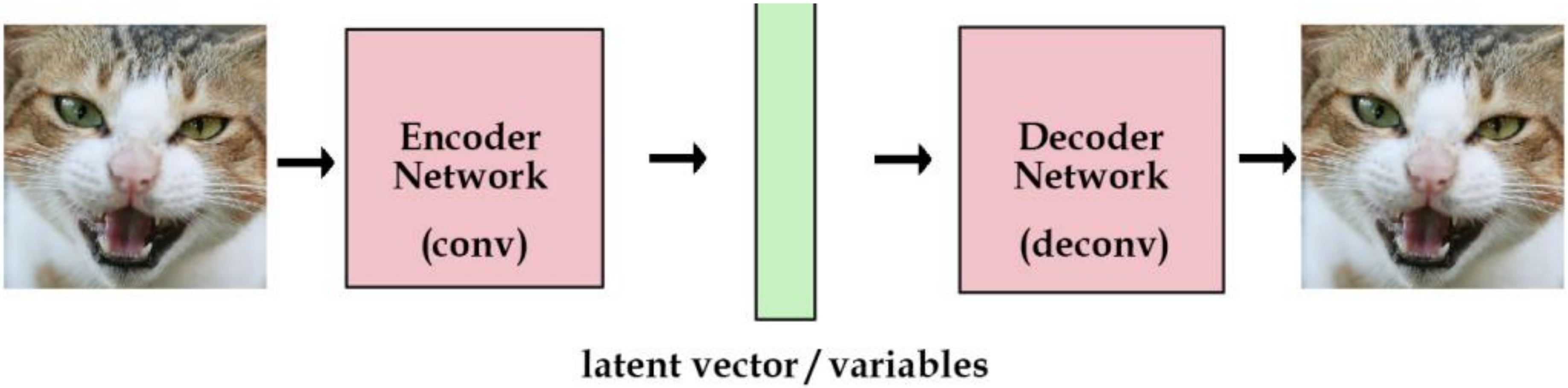
Concept: Auto-association



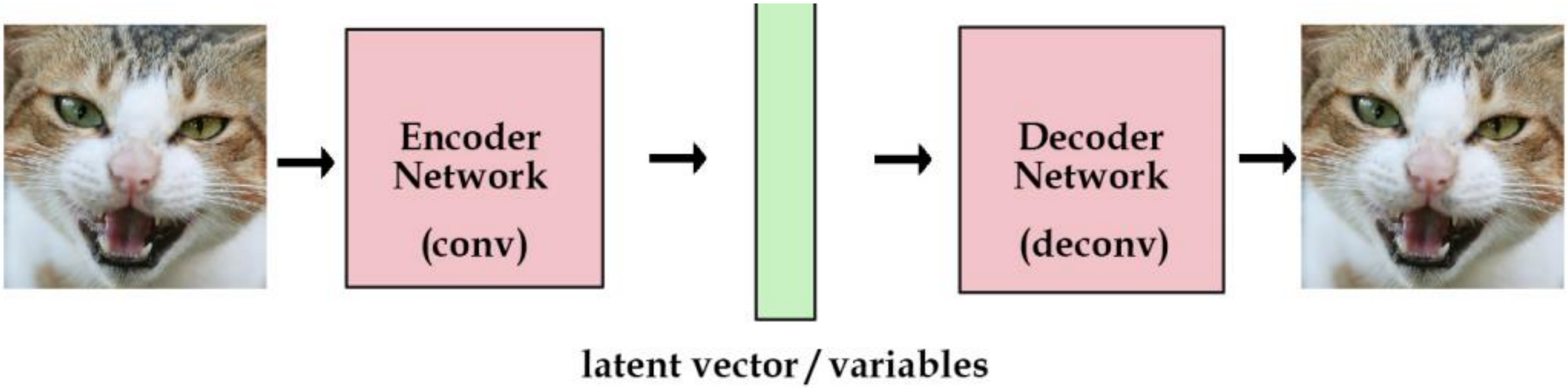
The Encoder-Decoder Framework

- Auto-association (auto-encoding)
  - Learn a compressed representation of the input, i.e., word2vec
  - Bottleneck layer = meaningful latent space
- Can de-couple encoder & decoder
  - Each can be complex, different functions





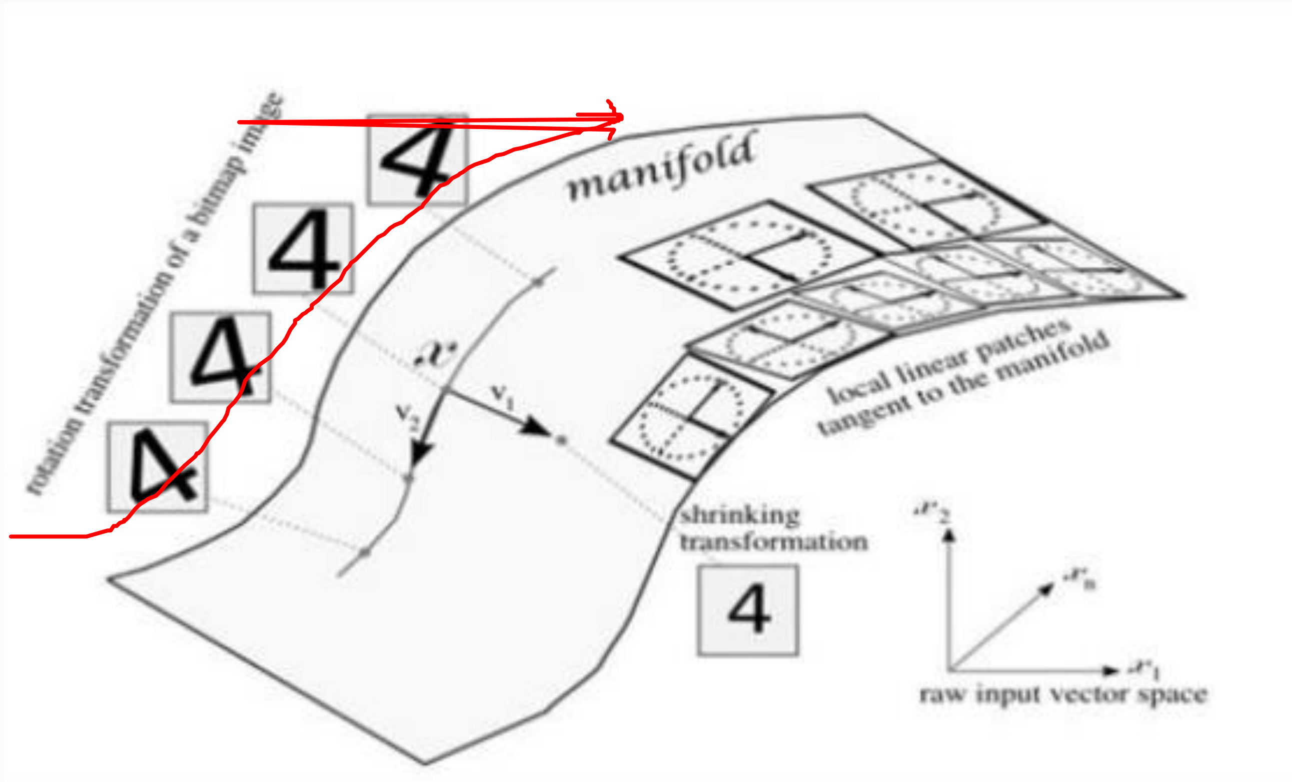
- Attempt to learn identity function
- Constrained in some way (e.g., small latent vector representation)
- Can generate new images by giving different latent vectors to trained network
- Variational: use probabilistic latent encoding



- Attempt to learn identity function
- Constrained in some way (e.g., small latent vector representation)
- Can generate new images by giving different latent vectors to trained network
- Variational: use probabilistic latent encoding

The Manifold Hypothesis

Natural data (high dimensional) actually lies in a low dimensional space.



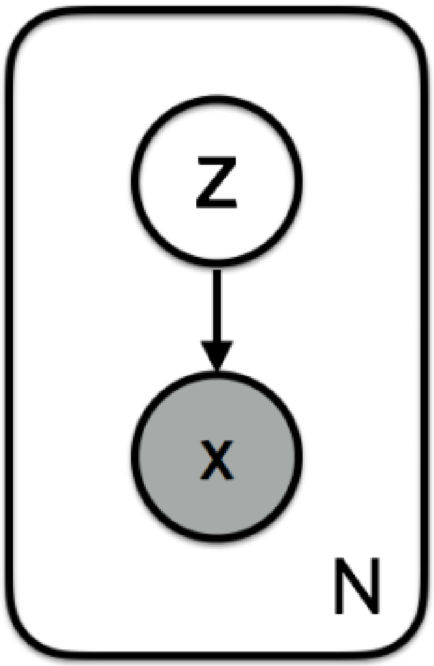
<u>Nice consequence:</u> many datasets that you think might require many variables/dimensions to describe can actually be explained with rather few of them (a subset that forms a sort of local coordinate system of the underlying manifold)

Probabilistic Model Perspective

- Data x and latent variables z
- Joint pdf of the model: $p(x, z) = p(x|z)p(z)$
- Decomposes into likelihood: $p(x|z)$, and prior: $p(z)$
- Generative process:

  Draw latent variables $z_i \sim p(z)$

  Draw datapoint $x_i \sim p(x|z)$
- Graphical model:



Traditional Approaches to Generative Modeling

- Explicit Modelling of $P(X|z;\theta)$, we will drop the θ in the notation.
- $z \sim P(z)$, which we can sample from, such as a Gaussian distribution.

$$P(X) = \int P(X|z;\theta)P(z)dz$$

- Maximum Likelihood --- Find θ to maximize $P(X)$, where X is the data.
- Approximate with samples of z

$$P(X) \approx \frac{1}{n}\sum_{i=0}^{n} P(X|z_i)$$

- Approximate with samples of z

$$P(X) \approx \frac{1}{n}\sum_{i=0}^{n} P(X|z_i)$$

- Need a lot of samples of z and most of the **P(X|z) ≈ 0.**
- Not practical computationally.
- **Question: Is it possible to know which z will generate P(X|z) >> 0?**
  - Learn a distribution Q(z), where z ~ Q(z) generates P(X|z) >> 0.

Towards Variational Inference

Assume we can learn a distribution Q(z), where z ~ Q(z) generates P(X|z) >> 0

- We want $P(X) = E_{z\sim P(z)}P(X|z)$, but not practical. $P(X) \approx \frac{1}{n}\sum_{i=0}^{n} P(X|z_i)$
- We **can** compute $E_{z\sim Q(z)}P(X|z)$, more practical.
- **Question:** How does $E_{z\sim Q(z)}P(X|z)$ and $P(X)$ relate?
    - **We take advantage of the following relationship:**

$$\boxed{\log P(X)} - \mathcal{D}\left[Q(z)\|P(z|X)\right] = \boxed{E_{z\sim Q}\left[\log P(X|z)\right]} - \mathcal{D}\left[Q(z)\|P(z)\right]$$

- Definition of KL divergence:

$$\boxed{\mathcal{D}\left[Q(z)\|P(z|X)\right]} = \boxed{E_{z\sim Q}\left[\log Q(z)\right.} - \boxed{\log \boxed{P(z|X)}\left.\right]}$$

- Apply **Bayes Rule on** P(z|X) and substitute into the equation above.
    - P(z|X) = P(X|z) P(z) / P(X)
    - log (P(z|X)) = log P(X|z) + log P(z) - log P(X)
    - P(X) does not depend on z, can take it outside of $E_{z\sim Q}$

$$\boxed{\mathcal{D}\left[Q(z)\|P(z|X)\right]} = \boxed{E_{z\sim Q}\left[\log Q(z)\right.} - \boxed{\log P(X|z)} - \boxed{\log P(z)}\left.\right] + \boxed{\log P(X)}$$

Designing a Recognition Model Q(z)

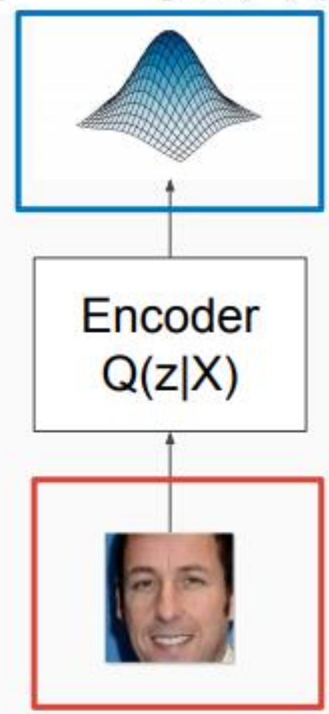
Why is this important?

$$\boxed{\log P(X) - \boxed{\mathcal{D}\left[Q(z)\|P(z|X)\right]}} = E_{z\sim Q}\left[\log P(X|z)\right] - \mathcal{D}\left[\boxed{Q(z)}\|P(z)\right]$$

- Recall we want to **maximize P(X)** with respect to θ, which we cannot do.
- KL divergence is always > 0.
- log P(X) > log P(X) - D[Q(z) || P(z|X)].
- Maximize the **lower bound** instead.
- **Question: How do we get Q(z) ?**

$$\log P(X) - \mathcal{D}\left[Q(z)\|P(z|X)\right] = E_{z\sim Q}\left[\log P(X|z)\right] - \mathcal{D}\left[Q(z)\|P(z)\right]$$

- **Q(z) or Q(z|X)?**
- Model Q(z|X) with a neural network.
- Assume Q(z|X) to be Gaussian, N(μ, c·I)
  - Neural network **outputs** the **mean μ**, and diagonal covariance matrix **c · I**.
  - **Input:** Image, **Output:** Distribution



Let's call Q(z|X) the **Encoder.**

Designing a Variational Encoder-Decoder

Convert the lower bound to a loss function:

$$\log P(X) - \mathcal{D}\left[Q(z)\|P(z|X)\right] = E_{z\sim Q}\left[\log \boxed{P(X|z)}\right] - \mathcal{D}\left[Q(z)\|P(z)\right]$$

- Model P(X|z) with a neural network, let f(z) be the network output.
- Assume P(X|z) to be i.i.d. **Gaussian**
  - X = f(z) + η , where η ~ N(0,I) *Think Linear Regression*
  - **Simplifies to an l_2 loss: $||X-f(z)||^2$**

Let's call P(X|z) the Decoder.

Convert the lower bound to a loss function:

$$\log P(X) - \mathcal{D}\left[Q(z)\|P(z|X)\right] = \boxed{E_{z\sim Q}\left[\log P(X|z)\right]} - \boxed{\mathcal{D}\left[Q(z)\|\boxed{P(z)}\right]}$$

Assume P(z) ~ N(0,I) then D[Q(z|X) || P(z)] has a closed form solution.

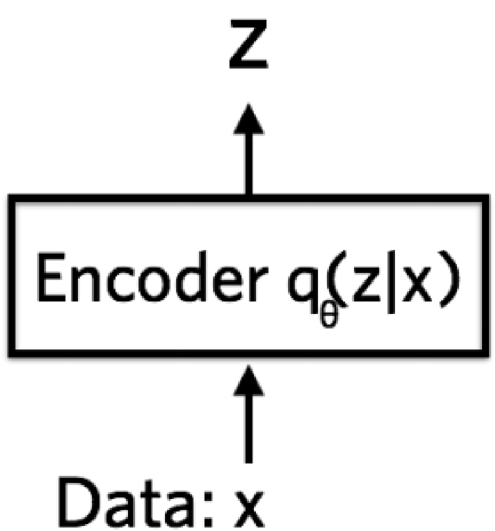
Putting it all together: $E_{z\sim Q(z|X)} \log P(X|z) \propto ||X-f(z)||^2$

$$L = ||X - f(z)||^2 - \lambda \cdot D[Q(z) \,||\, P(z)]$$

, given a (X, z) pair.

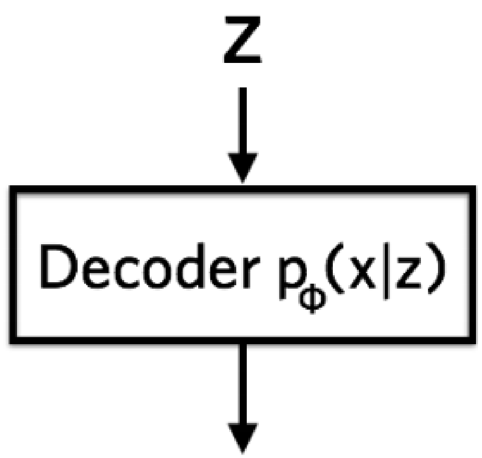
Pixel difference

Regularization

- Goal: Build a neural network that generates digits from random (Gaussian) noise
- Define two sub-networks: $\boxed{\text{Encoder}}$ and Decoder
- Define a Loss Function
  - A neural network $q_\theta(z|x)$
  - Input: datapoint x (e.g. 28×28-pixel digit)
  - Output: encoding z, drawn from Gaussian density with parameters θ
  - $|z| \ll |x|$
  - 

- Goal: Build a neural network that generates digits from random (Gaussian) noise
- Define two sub-networks: Encoder and $\boxed{\text{Decoder}}$
- Define a Loss Function

- A neural network $p_\phi(x|z)$, parameterized by ϕ
- Input: encoding z, output from encoder
- Output: reconstruction $\tilde{x}$, drawn from distribution of the data
- E.g., output parameters for 28×28 Bernoulli variables



- Reconstruction: $\tilde{x}$

Neural Variational Inference (NVIL)

- <u>Idea</u>: Teach neural net to approximate the posterior *p(z|x)*
  - *q(z|x)* with 'variational parameters' ϕ
  - One-shot approximate inference
  - Also known as a recognition model
    - Construct estimator of the variational (evidence) lower bound (ELBO)
    - Can optimize jointly w.r.t. ϕ jointly with θ -> Stochastic gradient ascent

D_{KL} KL-Divergence >= 0 depends on how good q(z|x) can approximate p(z|x)

<u>Recall from statistics and information theory:</u>

KL Divergence:

$$D_{\mathrm{KL}}(P\|Q) = \mathbb{E}_{\mathrm{x}\sim P}\left[\log \frac{P(x)}{Q(x)}\right] = \mathbb{E}_{\mathrm{x}\sim P}\left[\log P(x) - \log Q(x)\right]. \tag{3.50}$$

Gaussian KL Divergence:

$$KL(p,q) = \log\frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2}$$

$$p \sim N(\mu_1, \sigma_1)$$
$$q \sim N(\mu_2, \sigma_2)$$

Neural Variational Inference (NVIL)

- Idea: Teach neural net to approximate the posterior *p(z|x)*

$$\begin{aligned}\mathcal{L}(q) &= \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})} \log p_{\text{model}}(\boldsymbol{z}, \boldsymbol{x}) + \mathcal{H}(q(\mathbf{z} \mid \boldsymbol{x})) \\ &= \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})} \log p_{\text{model}}(\boldsymbol{x} \mid \boldsymbol{z}) - D_{\text{KL}}(q(\mathbf{z} \mid \boldsymbol{x}) || p_{\text{model}}(\mathbf{z})) \\ &\leq \log p_{\text{model}}(\boldsymbol{x}).\end{aligned}$$

Where:

$p_{\text{model}}(\boldsymbol{x}; g(\boldsymbol{z})) = p_{\text{model}}(\boldsymbol{x} \mid \boldsymbol{z})$ **and** $q(\boldsymbol{z} \mid \boldsymbol{x})$ is used to obtain $\boldsymbol{z}$.

<u>Recall from statistics and information theory:</u>

KL Divergence:

$$D_{\text{KL}}(P\|Q) = \mathbb{E}_{\mathrm{x}\sim P}\left[\log \frac{P(x)}{Q(x)}\right] = \mathbb{E}_{\mathrm{x}\sim P}\left[\log P(x) - \log Q(x)\right]. \qquad (3.50)$$

Gaussian KL Divergence:

$$KL(p,q) = \log \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2}$$

$$p \sim N(\mu_1, \sigma_1)$$
$$q \sim N(\mu_2, \sigma_2)$$

$$\mathcal{F}(\mathbf{x}, q) = \boxed{\mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{x}|\mathbf{z})]} - \boxed{KL[q(\mathbf{z})\|p(\mathbf{z})]}$$

Approx. Posterior

Reconstruction

Penalty

Interpreting the bound:

- **Approximate posterior distribution *q(z|x)*:** Best match to true posterior *p(z|x)*, one of the unknown inferential quantities of interest to us.

- **Reconstruction cost**: The expected log-likelihood measures how well samples from *q(z|x)* are able to explain the data *x*.

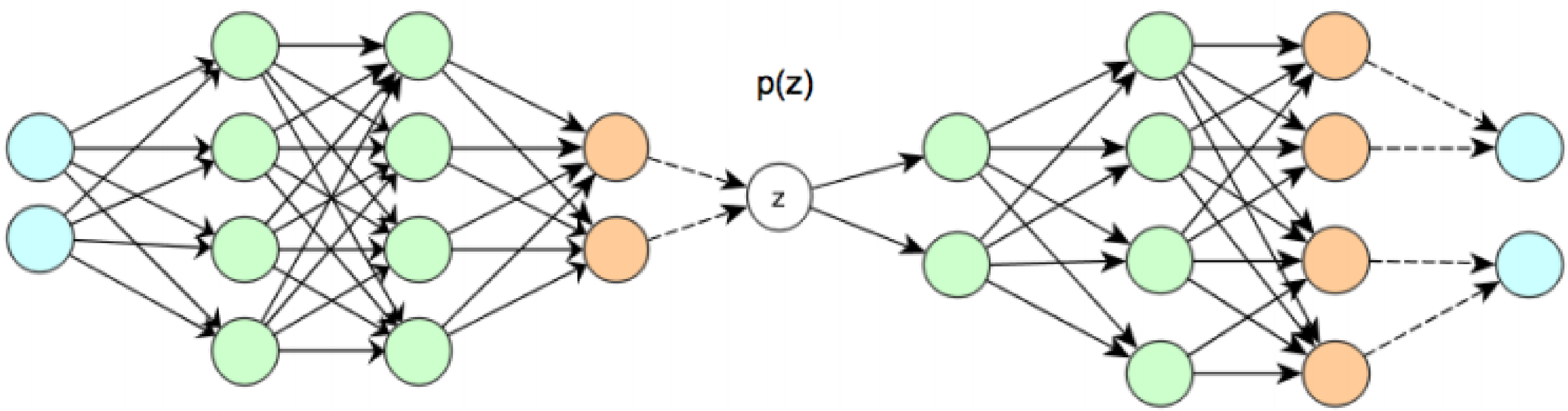
- **Penalty:** Ensures that the explanation of the data *q(z|x)* doesn't deviate too far from your beliefs *p(z)*. A mechanism for realising Ockham's razor.

The Variational Auto-Encoder

- A feed forward NN + Gaussian $q_\theta(z \mid x) = \mathcal{N}(z; \mu_z(x), \sigma_z(x))$

Just a Gaussian, with diagonal covariance.

$q_\Theta(z|x)$ $\qquad$ $p_\varphi(x|z)$ $\quad x|z \sim N(\mu_x, \sigma_x^{\,2})$



- For illustration z one dimensional x 2D
- Want a complex model of distribution of x given z

Learning the parameters φ and θ via backpropagation

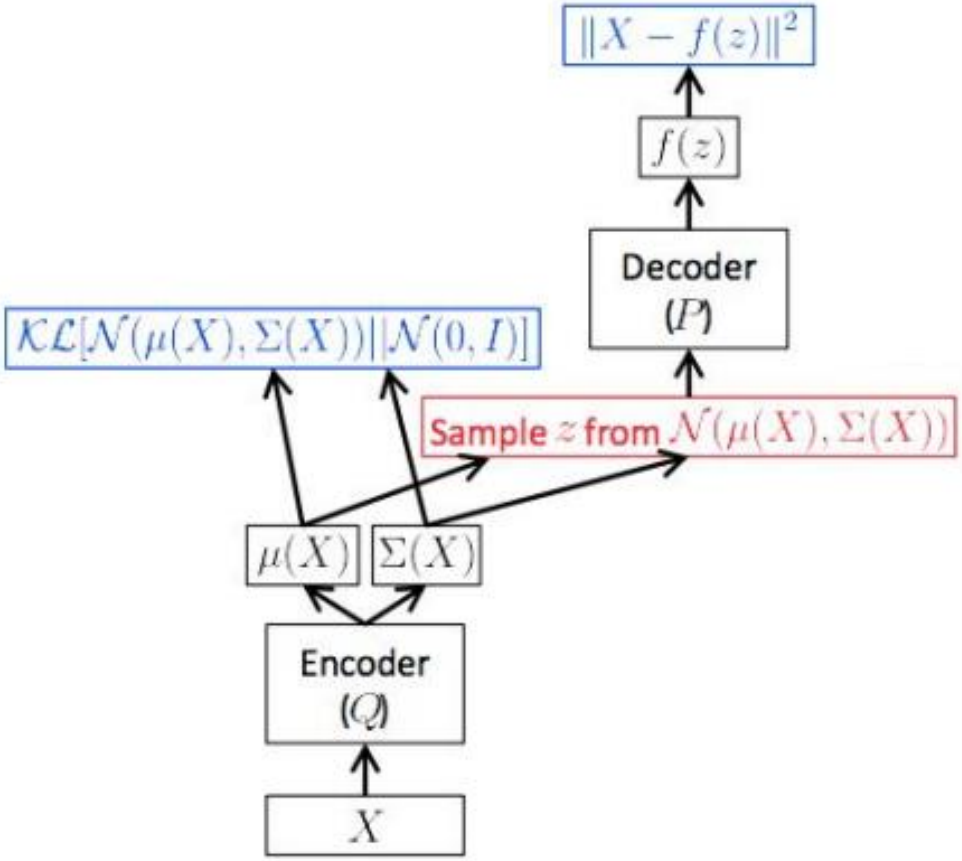
Determining the loss function

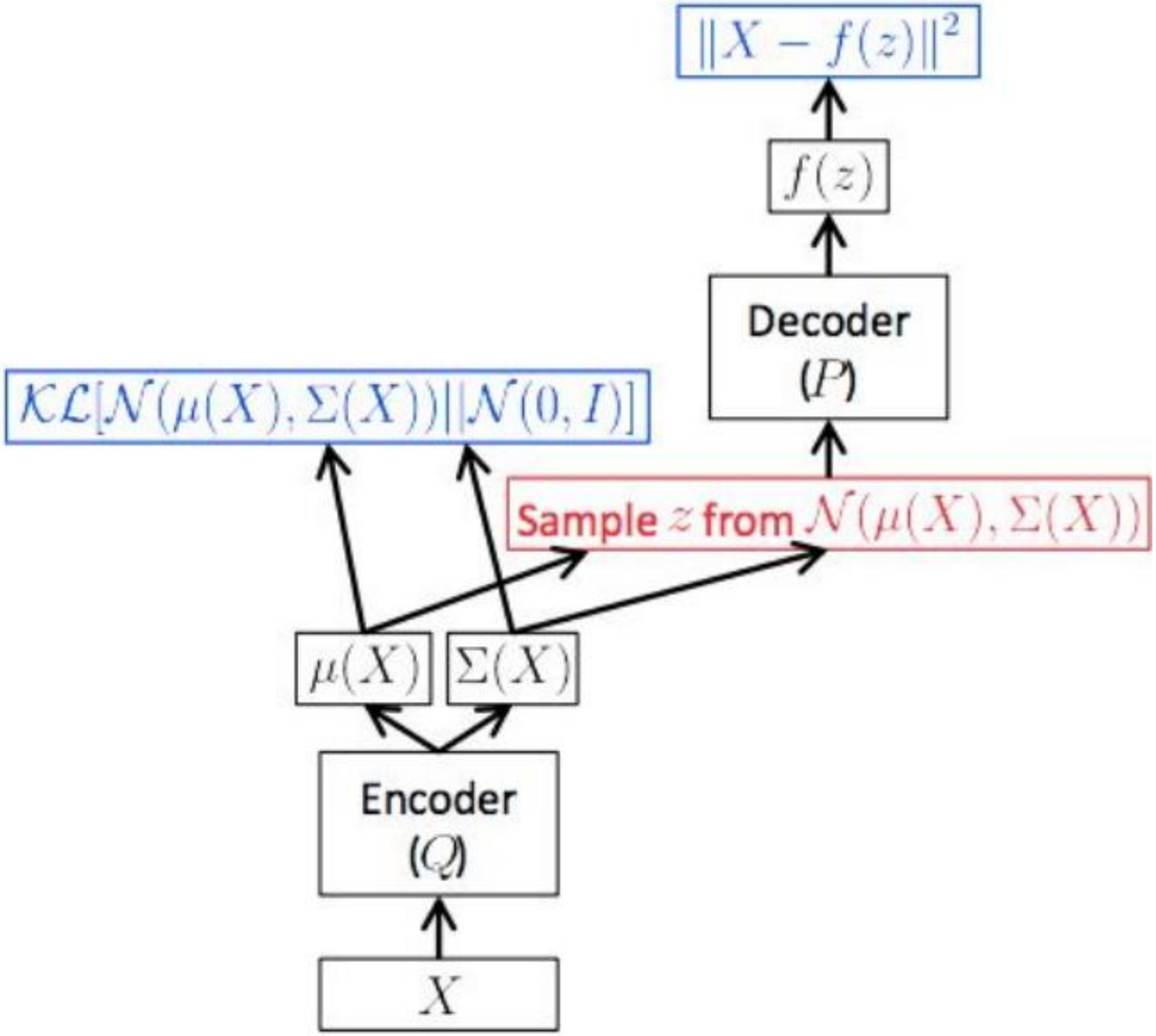
Issue: Backprop and Sampling

Training the Decoder is easy, just standard backpropagation.

How to train the Encoder?

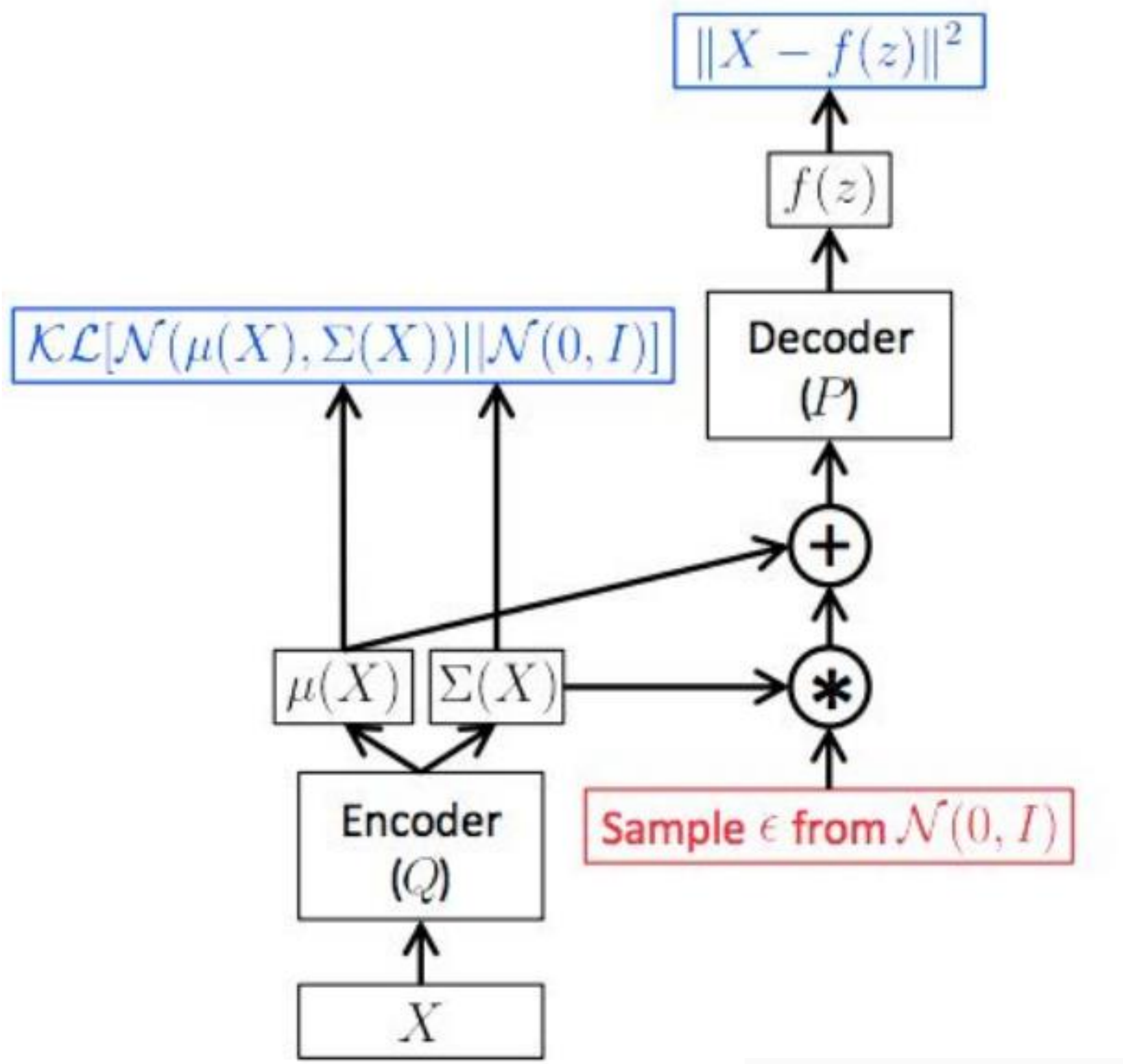
- Not obvious how to apply gradient descent through samples.





$\|X - f(z)\|^2$
$f(z)$
Decoder
(P)
$\mathcal{KL}[\mathcal{N}(\mu(X), \Sigma(X)) || \mathcal{N}(0, I)]$
Sample z from $\mathcal{N}(\mu(X), \Sigma(X))$
$\mu(X)$
$\Sigma(X)$
Encoder
(Q)
X

The Reparameterization Trick



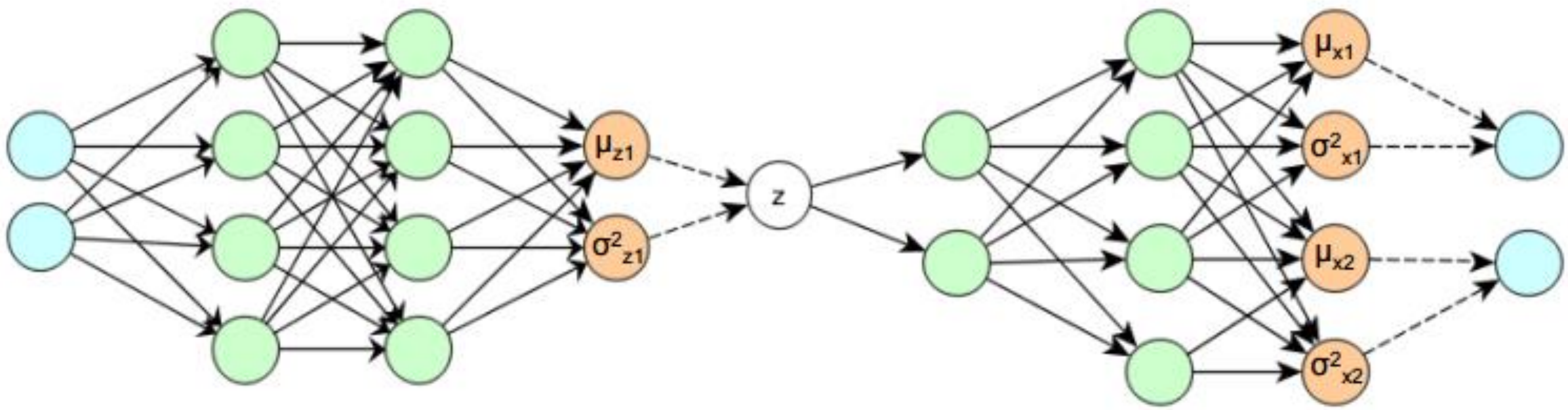
z ~ N(μ, σ) is equivalent to
μ + σ · ε, where ε ~ N(0, 1)

The Reparametrization Trick

- We want to use gradient descent to learn the model's parameters
- Given z drawn from $q_\theta(z|x)$, how do we take derivatives of (a function of) z w.r.t. θ?
- We can reparameterize: $z = \mu + \sigma \odot \epsilon$
- $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and $\odot$ is element-wise product
- Can take derivatives of (functions of) z w.r.t. μ and σ
- Output of $q_\theta(z|x)$ is vector of μ's and vector of σ's

Putting It All Together!

Prior p(z) ~ N(0,1) and p, q Gaussian, extension to dim(z) > 1 trivial



Cost: Regularisation

$$-D_{\mathrm{KL}}\left(q(z|x^{(i)})||p(z)\right) = \frac{1}{2}\sum_{j=1}^{J}\left(1 + \log(\sigma_{z_j}^{(i)^2}) - \mu_{z_j}^{(i)^2} - \sigma_{z_j}^{(i)^2}\right)$$

We use mini batch gradient decent to optimize the cost function over all $x^{(i)}$ in the mini batch

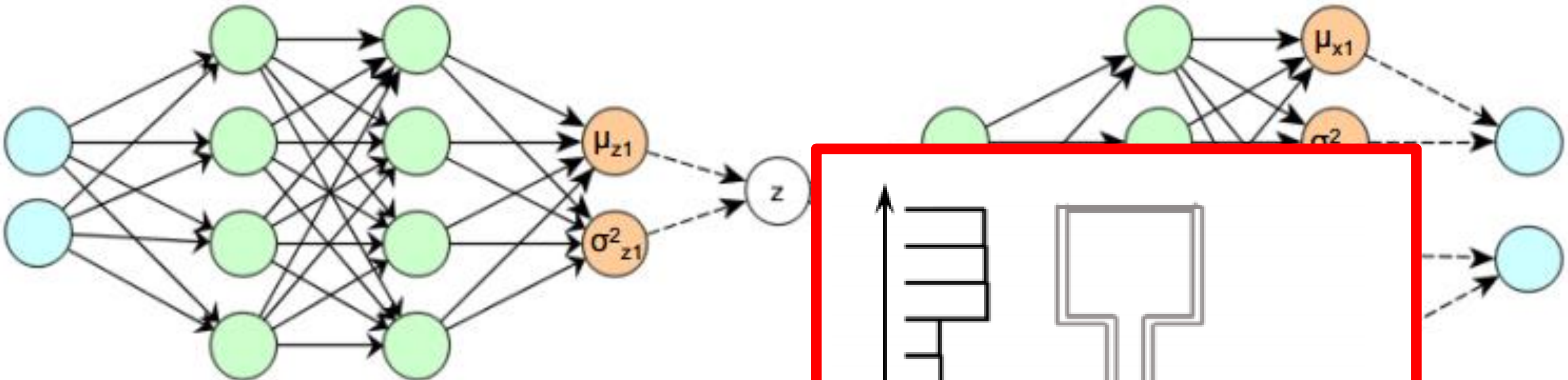
Cost: Reproduction

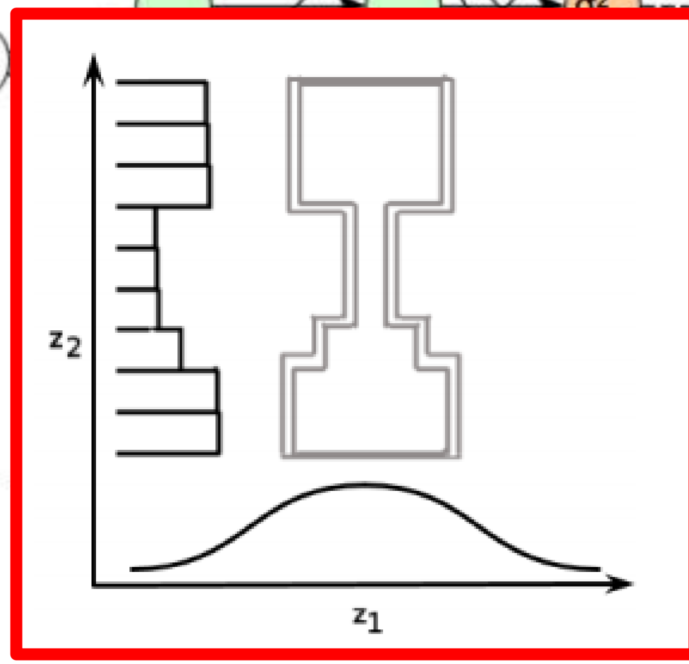
$$-\log\left(p(x^{(i)}|z^{(i)})\right) = \sum_{j=1}^{D}\frac{1}{2}\log(\sigma_{x_j}^2) + \frac{(x_j^{(i)} - \mu_{x_j})^2}{2\sigma_{x_j}^2}$$

Least Square for constant variance

Putting It All Together!

Prior p(z) ~ N(0,1) and p, q Gaussian, extension to dim(z) > 1 trivial





Cost: Regularisation

$$-D_{KL}\left(q(z|x^{(i)})||p(z)\right) = \frac{1}{2}\sum_{j=1}^{J}\left(1 + \log(\sigma_{z_j}^{(i)^2}) - \mu_{z_j}^{(i)^2}\right.$$

lient
cost
e mini

Why not capture modes (in text) with piecewise linear variables?
(Serban & Ororbia, 2016)

$$\sum_{j=1} \frac{1}{2} \qquad \frac{(x_j^{(i)} - \mu_{x_j})^2}{2\sigma_{x_j}^2}$$

Least Square for constant variance

VAE Training Algorithm

Given a dataset of examples **X** = {X1, X2...}

Initialize parameters for Encoder and Decoder

Repeat till convergence:

$\mathbf{X}^M$ <-- Random minibatch of M examples from **X**

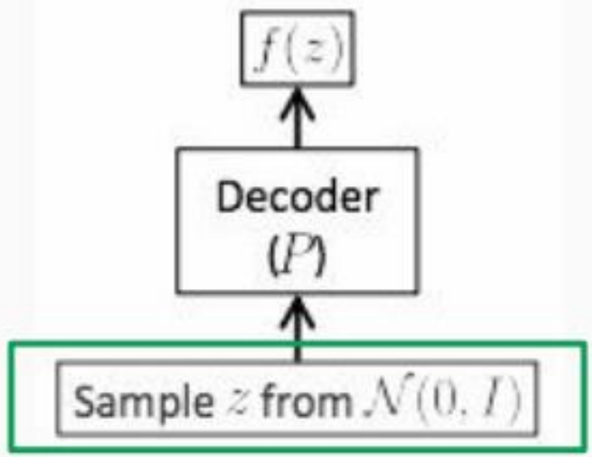
ε <-- Sample M noise vectors from N(0, I)

Compute $L(\mathbf{X}^M, \varepsilon, \theta)$ (i.e. run a forward pass in the neural network)

Gradient descent on L to updated Encoder and Decoder.

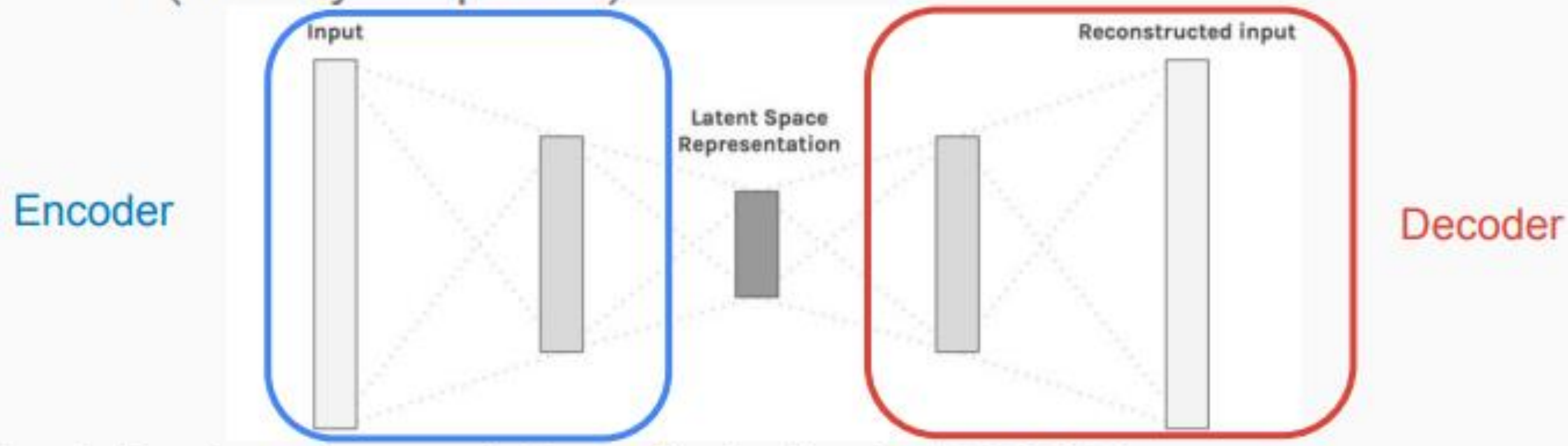
VAE Evaluation

- At test-time, we want to evaluate the **performance of VAE to generate a new sample.**
- Remove the Encoder, as **no test-image** for generation task.
- Sample $z \sim N(0,I)$ and pass it through the Decoder.
- **No good quantitative metric, relies on visual inspection.**

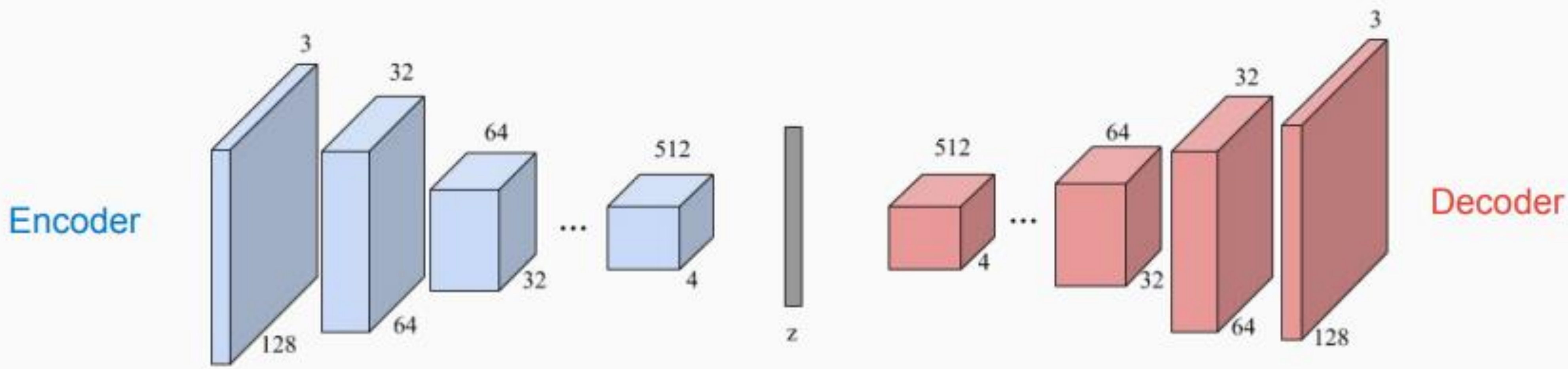


VAE Architectures

Fully Connected (Initially Proposed)

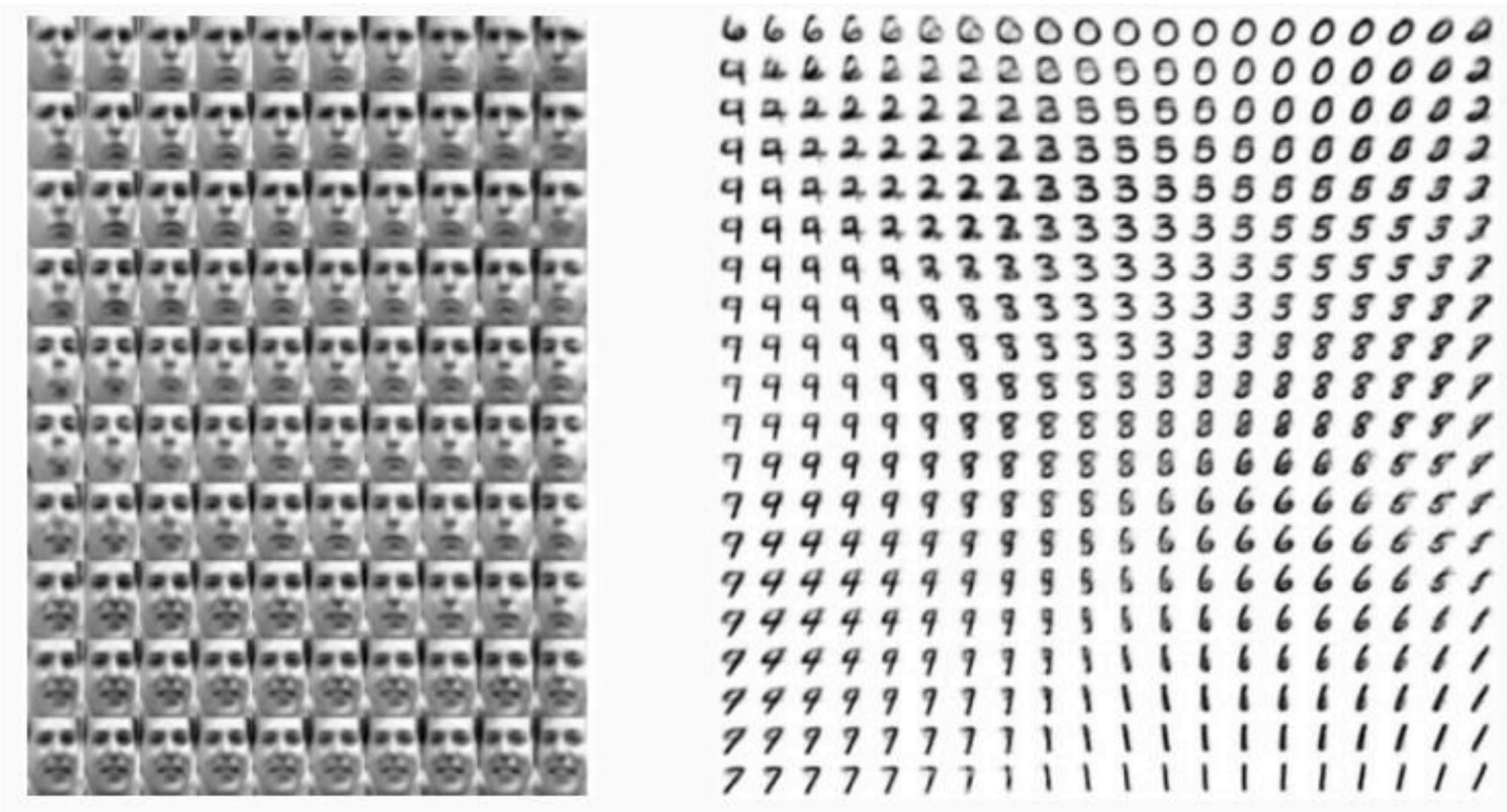


Common Architecture (convolutional) similar to DCGAN.



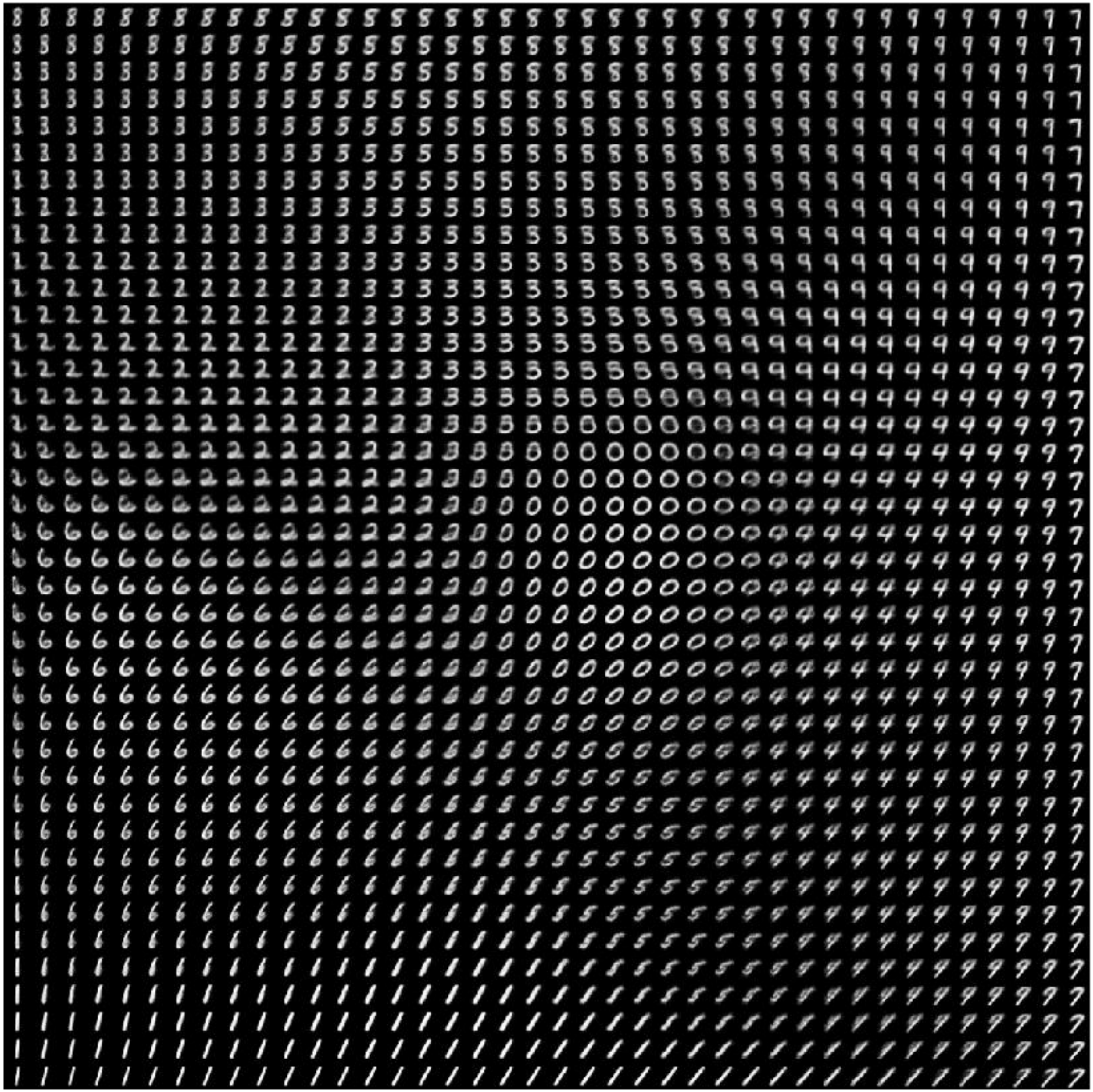
Fantasies/Dreams of the VAE

- VAEs can disentangle potential factors of variation



http://www.dpkingma.com/sgvb_mnist_demo/demo.html

Deep digit dreams…



Issues/Limitations with VAEs

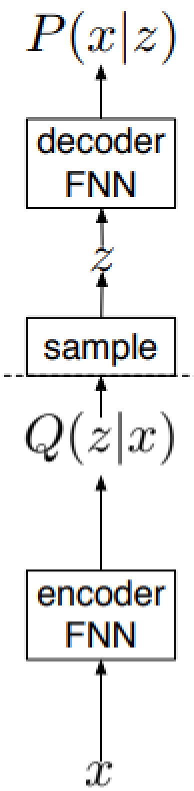
- Images are blurry (compared to models based on GANs, for example)
  - Result of likelihood objective? (places probability mass on training images and nearby points, which include blurry images)
    - Has tendency to ignore input features that occupy few pixels or that cause only small change in brightness of pixels (that they occupy)
  - Uses only small subset of latent variables? (struggles to find enough transformation directions to match factorized prior over z)

DRAW

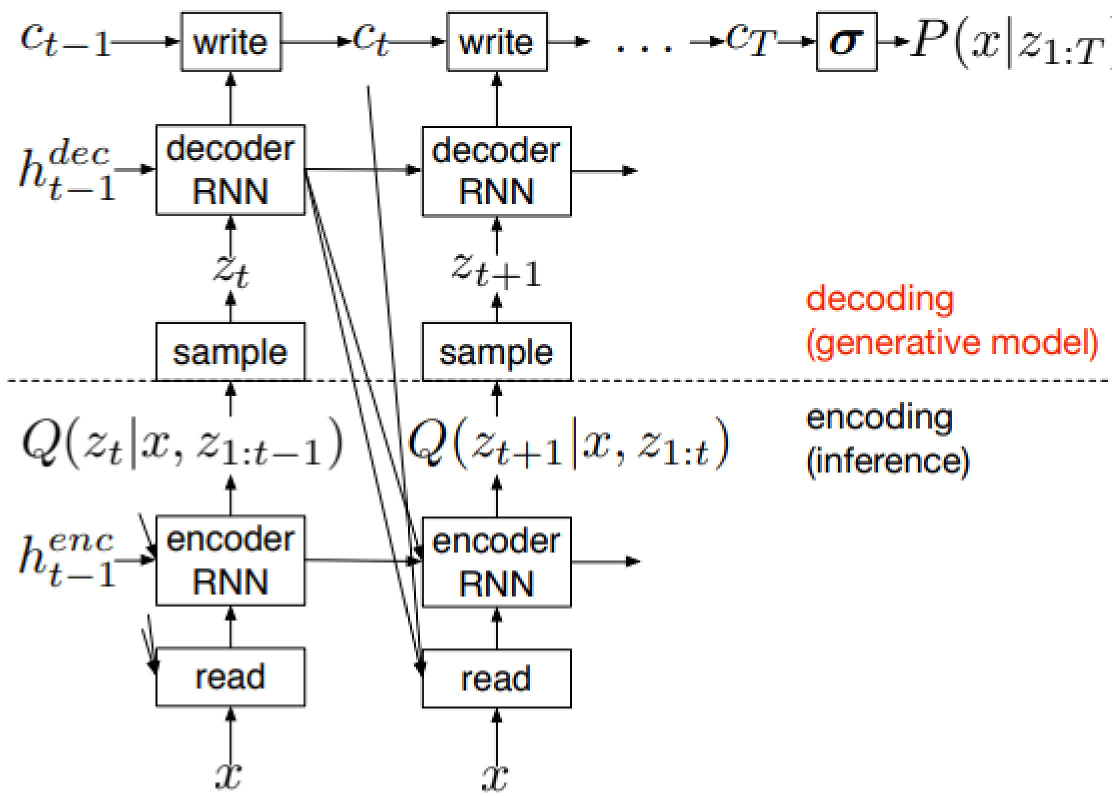
The Deep Recurrent Attentive Writer

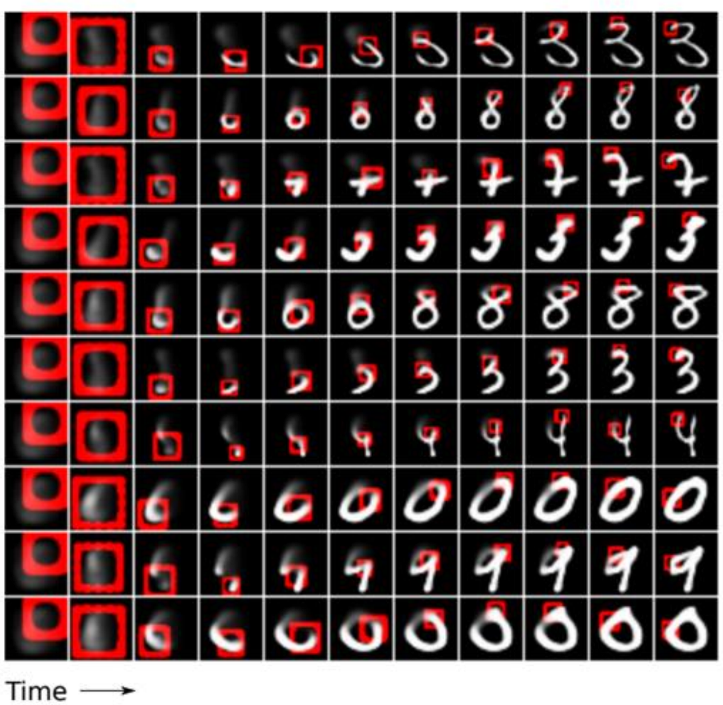
Combining latents, variational inference, and RNNs

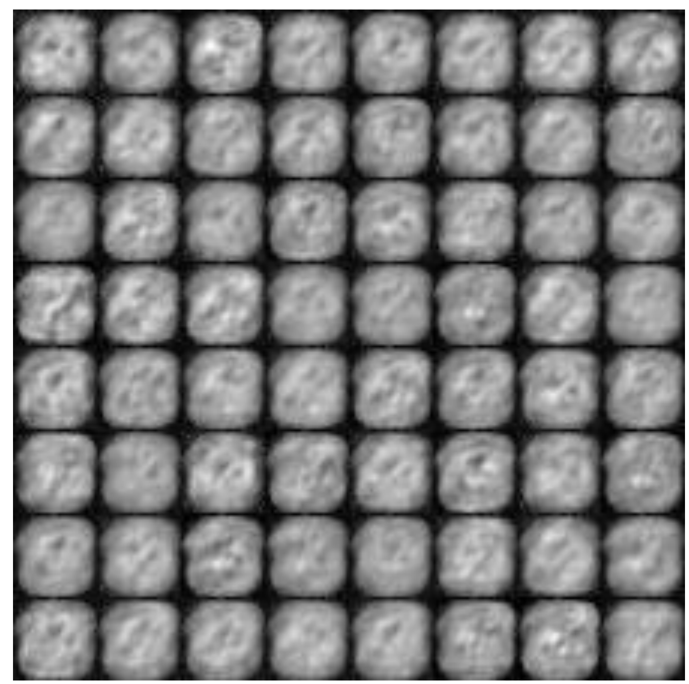
VAE



DRAW







QUESTIONS?