Deep Learning:

Variational Autoencoders

Ozan Özdenizci

Institute of Theoretical Computer Science

ozan.ozdenizci@igi.tugraz.at

Deep Learning VO - WS 23/24

Lecture 9 - December 4th, 2023

Today

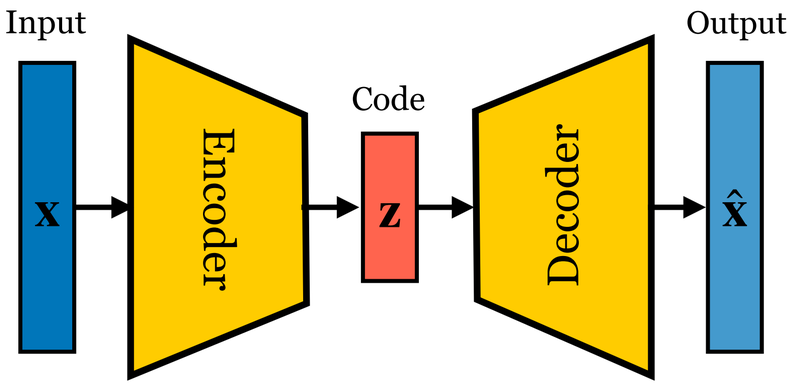
- [ ] Autoencoders
- [ ] Deep Generative Models
  - [ ] Variational Autoencoders
    - [ ] Architecture & Training
    - [ ] Variational Lower Bound
    - [ ] Experiments & Quantifying Performance

Autoencoders

An **autoencoder** is a neural network that gets as input some $\mathbf{x} \in \mathbb{R}^D$ and outputs a reconstruction $\hat{\mathbf{x}} \in \mathbb{R}^D$.

The neural network consists of an **encoder** $f_{\text{enc}} : \mathbb{R}^D \to \mathbb{R}^{d_z}$, and a **decoder** $f_{\text{dec}} : \mathbb{R}^{d_z} \to \mathbb{R}^D$, with a bottleneck layer with activations $\mathbf{z} \in \mathbb{R}^{d_z}$ where the information about the input is compressed.

The network is trained with gradient descent to minimize the reconstruction error (e.g., mean-squared error).

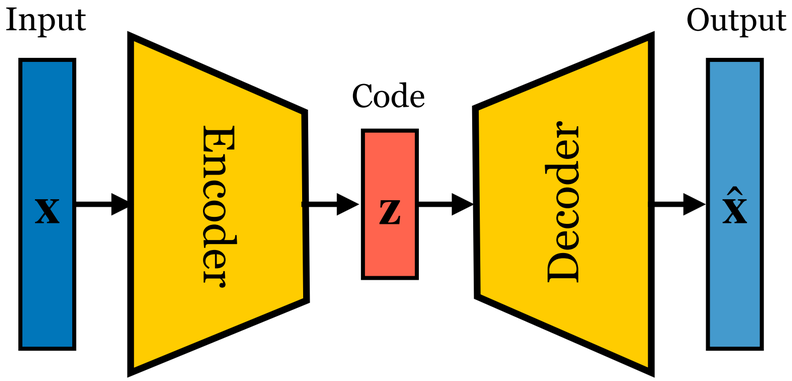


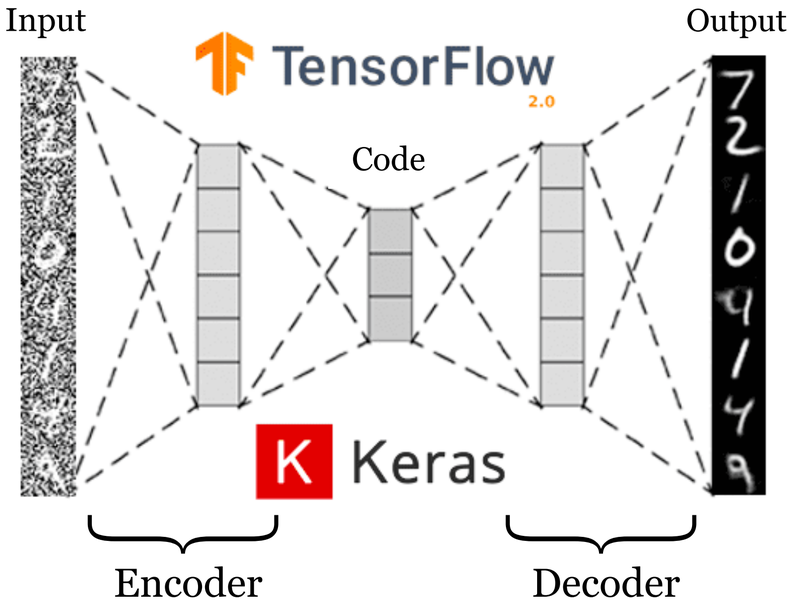
Autoencoders

An **autoencoder** is a neural network that gets as input some $\mathbf{x} \in \mathbb{R}^D$ and outputs a reconstruction $\hat{\mathbf{x}} \in \mathbb{R}^D$.

The neural network consists of an **encoder** $f_{\text{enc}} : \mathbb{R}^D \to \mathbb{R}^{d_z}$, and a **decoder** $f_{\text{dec}} : \mathbb{R}^{d_z} \to \mathbb{R}^D$, with a bottleneck layer with activations $\mathbf{z} \in \mathbb{R}^{d_z}$ where the information about the input is compressed.

The network is trained with gradient descent to minimize the reconstruction error (e.g., mean-squared error).



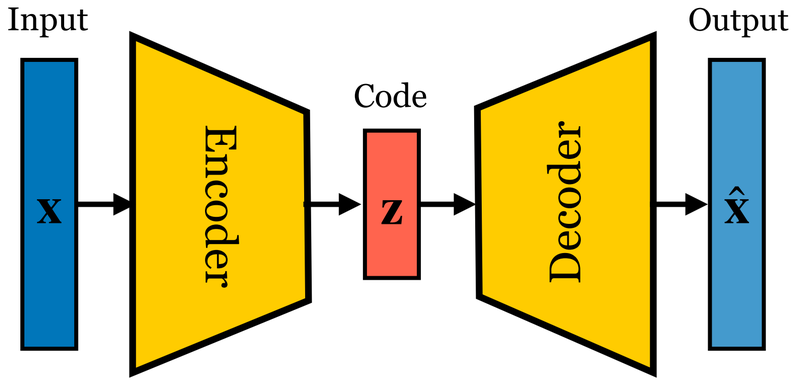


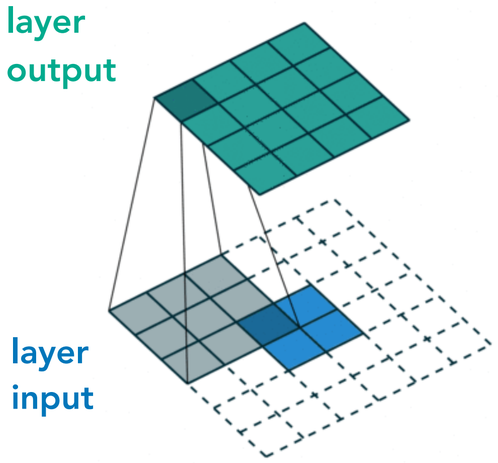
Applications:

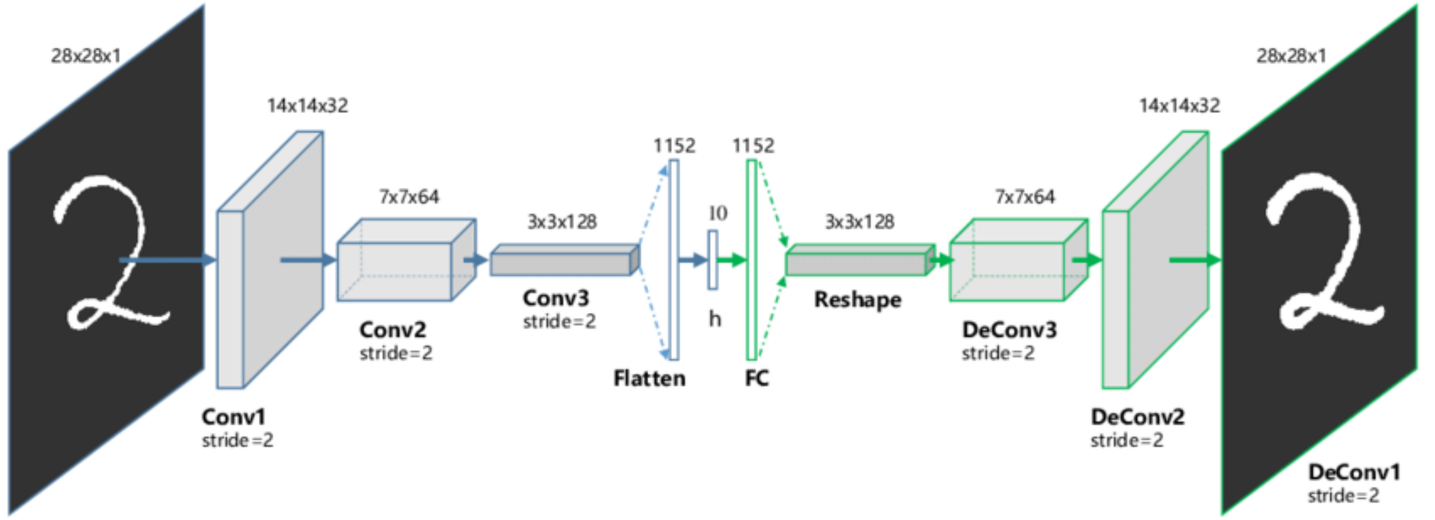
- Dimensionality reduction
- Feature extraction
- Denoising autoencoders → Training an autoencoder to reconstruct the original data from corrupted/noisy versions.

Encoder-Decoder Architectures

- **Encoder** maps the input to a low-dimensional latent representation by using (often) a CNN.
- In that case it makes sense to use as **decoder** something like the inverse of a CNN.
- This can be done with **transposed convolutions** (i.e., de-convolution) in the decoder.

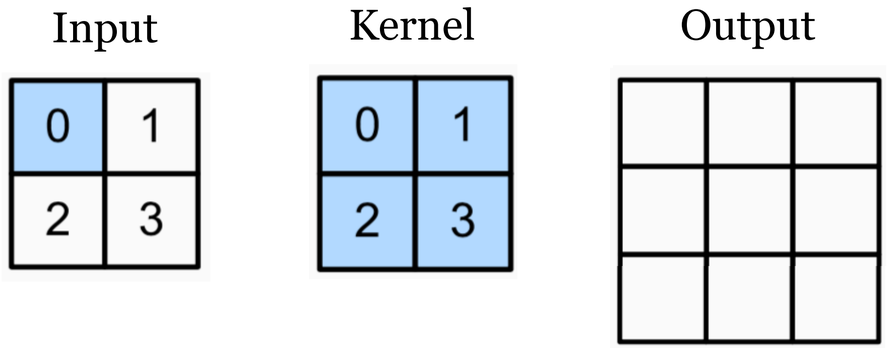






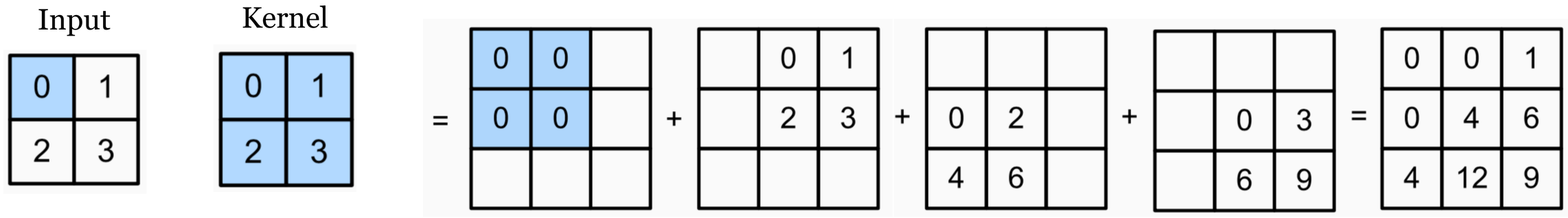
Learnable Upsampling: Transposed Convolution

- Consider a 2x2 feature map as **input.**
- We want to expand it to a 3x3 feature map **output,**
- By using a 2x2 **kernel** for transposed convolution.



Each element of the input is multiplied with the kernel and added to the output at the corresponding position.

Example: Transposed convolution with stride 1:



Today

- [x] Autoencoders
- [ ] Deep Generative Models
  - [ ] Variational Autoencoders
    - [ ] Architecture & Training
    - [ ] Variational Lower Bound
    - [ ] Experiments & Quantifying Performance

Diederik P. Kingma & Max Welling, “Auto-Encoding Variational Bayes”, arXiv preprint arXiv:1312.6114 (2013).

Generative Models

- Given a list of data points $X = \langle \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)} \rangle$ coming from a data distribution $p(\mathbf{x})$.
- **Generative model:** a model for this distribution, from which we can sample.
- Notation: typically the model distribution is parameterized by θ and we write $p_\theta(\mathbf{x})$.

Simple example from previous lectures:

- Assume that data comes from a Gaussian distribution. Our model could then be: $p_\theta(\mathbf{x}) = N(\mu, \mathbf{\Sigma})$,
- … where the parameters $\theta = (\mu, \mathbf{\Sigma})$ are the mean and covariance matrix of the Gaussian.
- We have already seen how we can learn θ with maximum likelihood.

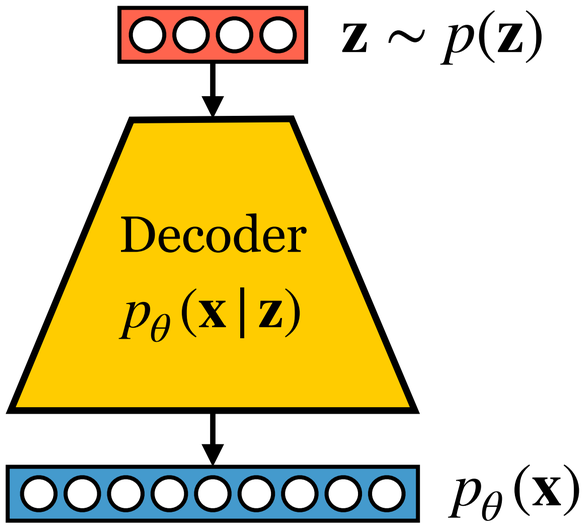
Variational Autoencoders (VAEs)

There are many ways to model more complex distributions.

Simple Idea:

- Draw latent variables $\mathbf{z}$ from a simple distribution $p(\mathbf{z})$.
- Use a model with parameters θ to transform $p(\mathbf{z})$ to a complex distribution:

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}\,|\,\mathbf{z})\, p(\mathbf{z})\, d\mathbf{z}$$

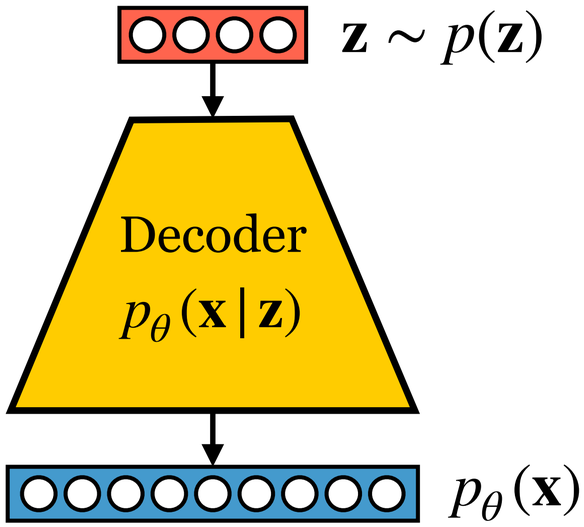


- To generate an $\mathbf{x}$:
  - We first sample $\mathbf{z} \sim p(\mathbf{z})$, then compute neural network output with $\mathbf{z}$ as input.
- In VAEs, we usually use as priors independent standard normal variables (zero mean, unit variance).

Variational Autoencoders: Training

How to train VAEs?

- For a dataset $X = \langle \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)} \rangle$ we want to maximize the likelihood that it is generated.
- If we would knew $\mathbf{z}$ for a given $\mathbf{x}$, we could simply optimize $p_\theta(\mathbf{x} \mid \mathbf{z})$, but $\mathbf{z}$ is latent.
- So we would need to compute $p_\theta(\mathbf{z} \mid \mathbf{x})$, which is intractable.



Variational Autoencoders: Training

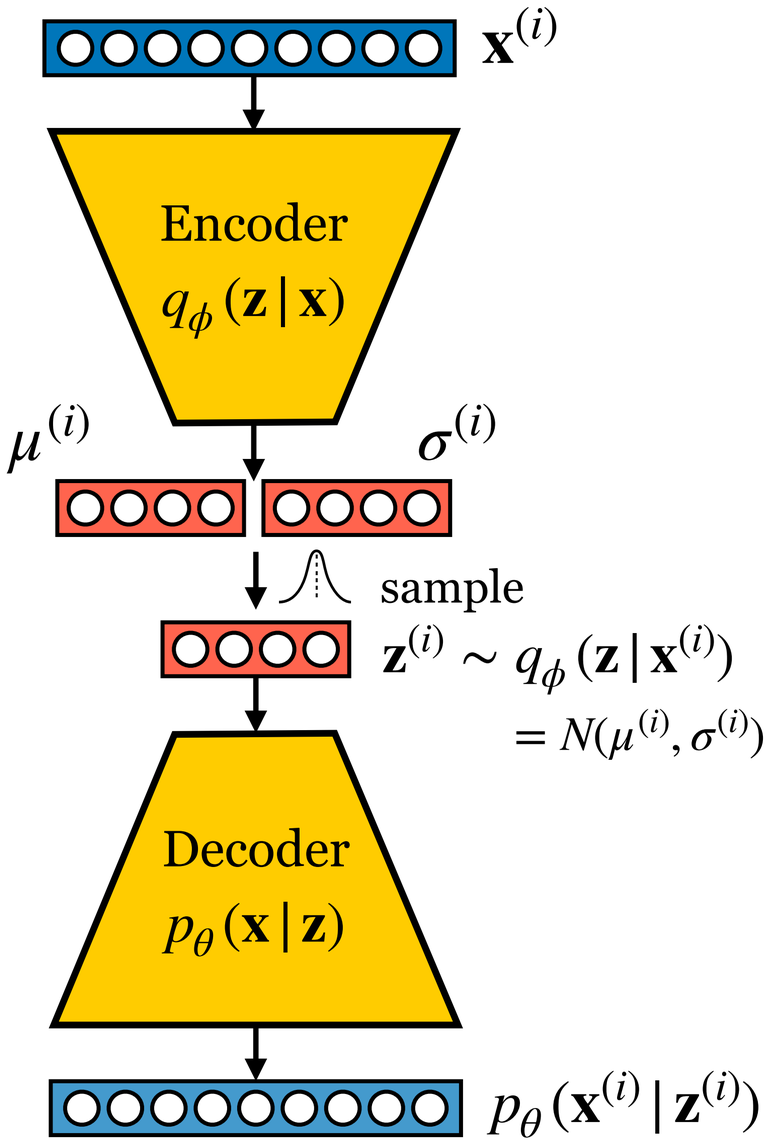
Solution of VAEs:

- Predict latent variables $\mathbf{z}$ using a neural network with parameters ϕ that computes the approximate posterior $q_\phi(\mathbf{z}|\mathbf{x}) \approx p_\theta(\mathbf{z}|\mathbf{x})$.

For a given $\mathbf{x}$,

- the **encoder** samples $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$,
- the **decoder** takes $\mathbf{z}$ and computes $p_\theta(\mathbf{x}|\mathbf{z})$.

Now we have $\mathbf{z}$ and can optimize both ϕ and θ.



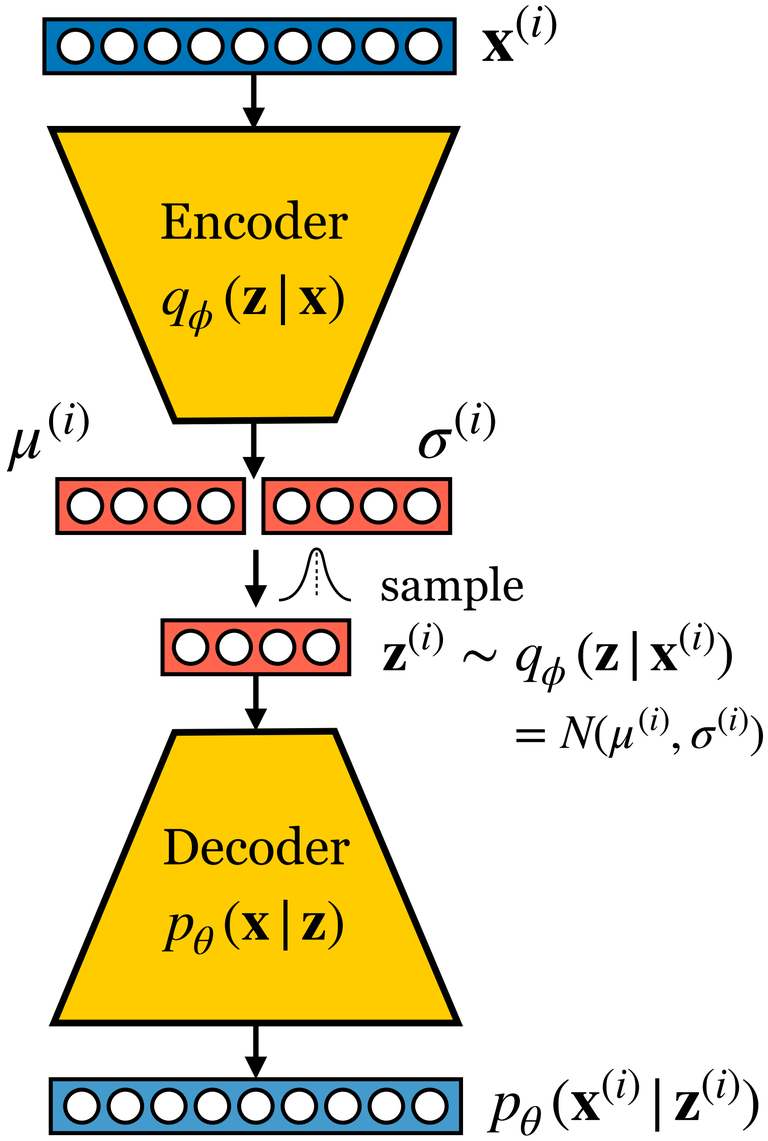
Variational Autoencoders: Training

Training objective

We want that:

1) The distribution $q_\phi(\mathbf{z}\,|\,\mathbf{x}^{(i)})$ is close to our prior distribution $p(\mathbf{z})$.

2) The decoder computes a good reconstruction of $\mathbf{x}^{(i)}$.



Variational Autoencoders: Training

1) The distribution $q_\phi(\mathbf{z}\,|\,\mathbf{x}^{(i)})$ is close to our prior distribution $p(\mathbf{z})$:

- We achieve this by minimizing the KL divergence between these distributions.

$$D_{KL}(q\|p) = \int q(x)\,\log\frac{q(x)}{p(x)}dx$$

- We will use a Gaussian prior with zero-mean independent variables of unit variance: $p(\mathbf{z}) = N(\mathbf{0}, \mathbf{I})$, i.e., $p(z_m) = N(0,1)$.
- For each latent variable z_m , the encoder produces a Gaussian distribution with mean μ_m and standard deviation σ_m, i.e.,

$$q_\phi(z_m\,|\,\mathbf{x}^{(i)}) = N\left(\mu_m(\mathbf{x}^{(i)}), \sigma_m(\mathbf{x}^{(i)})\right)$$

- The KL divergence in this case is: $D_{KL}(q_\phi\|p) = -\frac{1}{2}\sum_m\left(1 + \log\left(\sigma_m(\mathbf{x}^{(i)})^2\right) - \mu_m(\mathbf{x}^{(i)})^2 - \sigma_m(\mathbf{x}^{(i)})^2\right)$

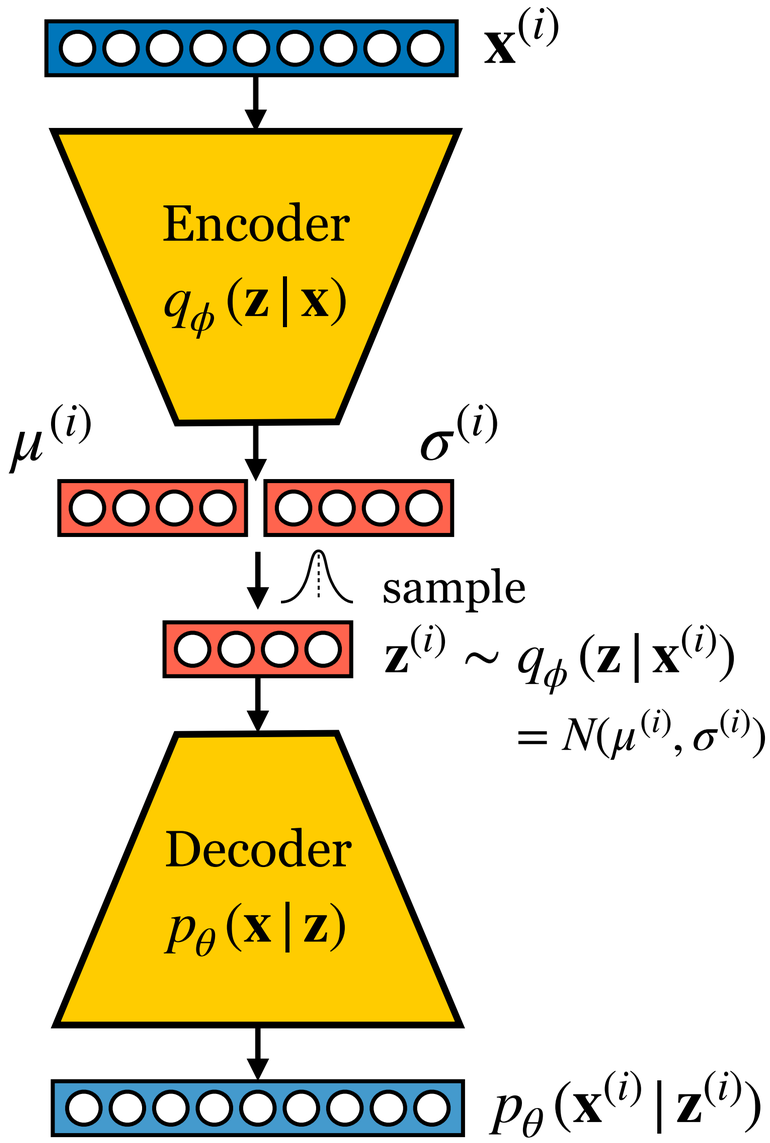
Variational Autoencoders: Training

1) The distribution $q_\phi(\mathbf{z} \mid \mathbf{x}^{(i)})$ is close to our prior distribution $p(\mathbf{z})$:

2) The decoder computes a good reconstruction of $\mathbf{x}^{(i)}$:

   - We achieve this by maximizing the likelihood on L drawn samples $\mathbf{z}^{(i,1)}, \ldots, \mathbf{z}^{(i,L)}$.

$$\frac{1}{L} \sum_{l=1}^{L} \log p_\theta \left(\mathbf{x}^{(i)} \mid \mathbf{z}^{(i,l)} \right)$$



Variational Autoencoders: Training

1) The distribution $q_\phi(\mathbf{z}\,|\,\mathbf{x}^{(i)})$ is close to our prior distribution $p(\mathbf{z})$:

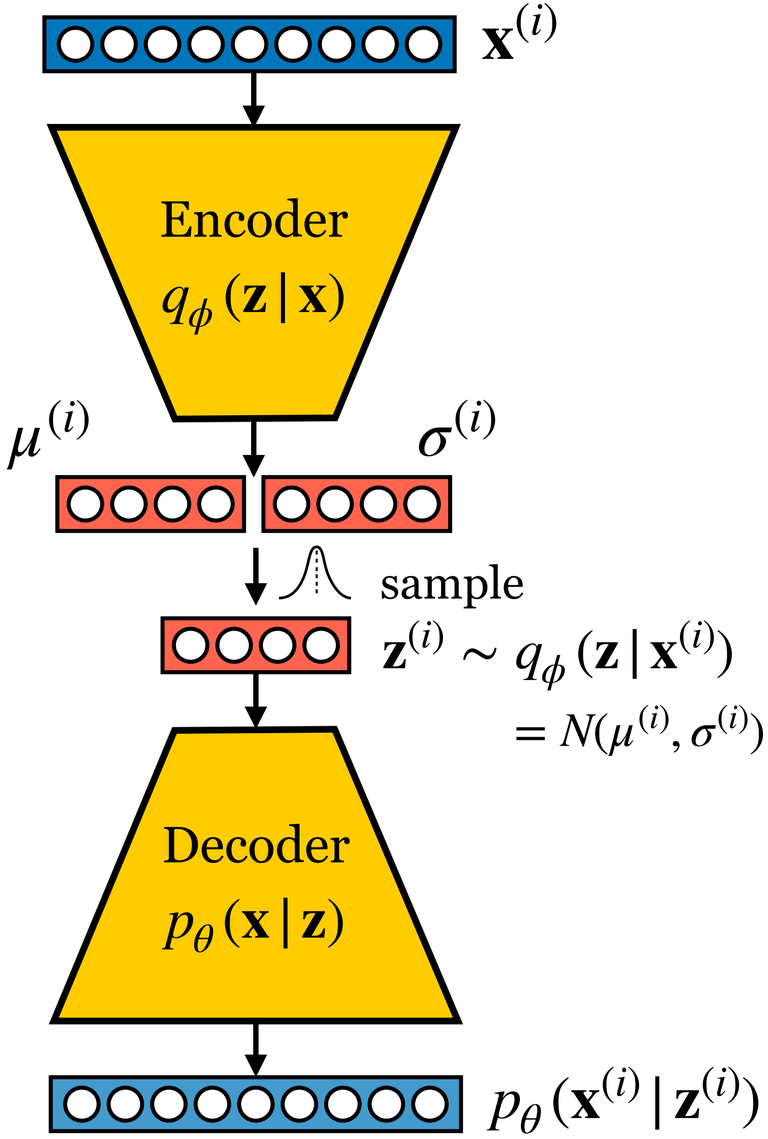
2) The decoder computes a good reconstruction of $\mathbf{x}^{(i)}$:

- Both parts together give rise to the loss to be minimized:

$$\mathscr{L}(\theta, \phi, X) = D_{KL}\left(q_\phi(\mathbf{z}\,|\,\mathbf{x}^{(i)}) \,\|\, p(\mathbf{z})\right) - \frac{1}{L}\sum_{l=1}^{L} \log p_\theta\left(\mathbf{x}^{(i)}\,|\,\mathbf{z}^{(i,l)}\right)$$

approx. posterior ≈ prior

"reconstruction" term



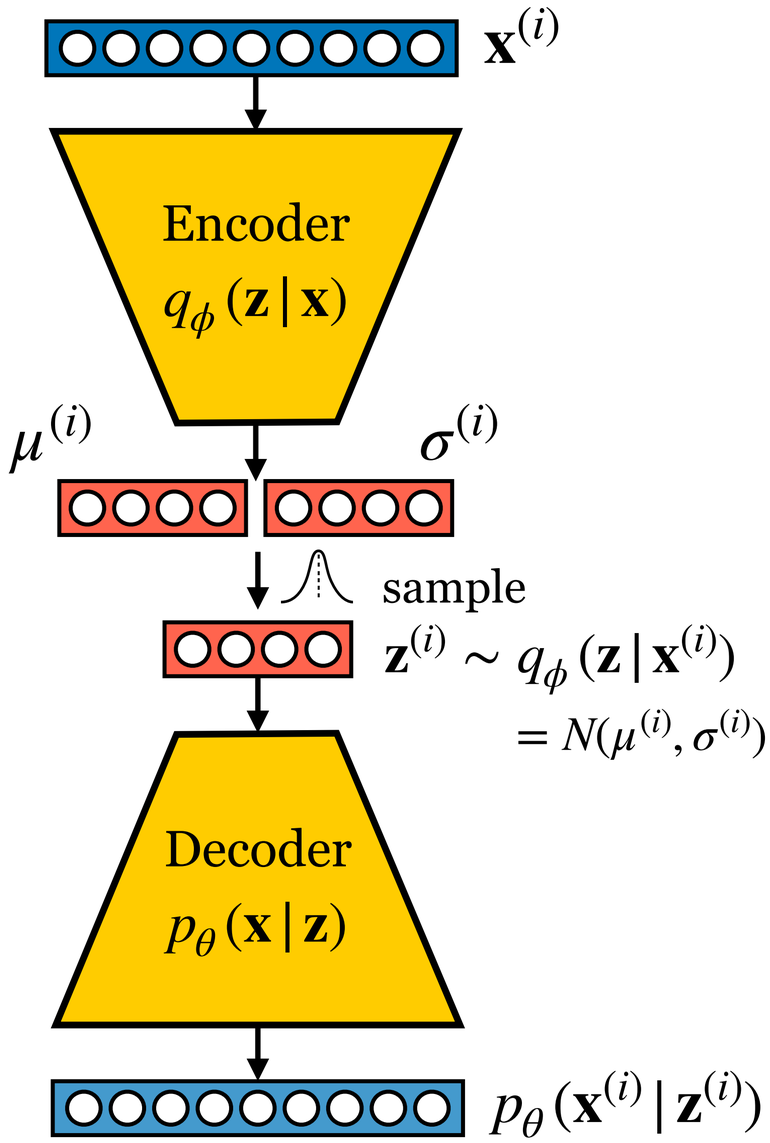
Variational Autoencoders: Training

1) The distribution $q_\phi(\mathbf{z}\,|\,\mathbf{x}^{(i)})$ is close to our prior distribution $p(\mathbf{z})$:

2) The decoder computes a good reconstruction of $\mathbf{x}^{(i)}$:

Note: Once the model is trained,

- The encoder is not needed for generation.
  - We just need to sample from the latent prior $p(\mathbf{z})$, and then use the decoder.
- The encoder can be used to extract the latent variables.
  - Low-dimensional representations of data.
  - For this we don't need the decoder.



Variational Autoencoders: Training

1) The distribution $q_\phi(\mathbf{z}\,|\,\mathbf{x}^{(i)})$ is close to our prior distribution $p(\mathbf{z})$:

2) The decoder computes a good reconstruction of $\mathbf{x}^{(i)}$:

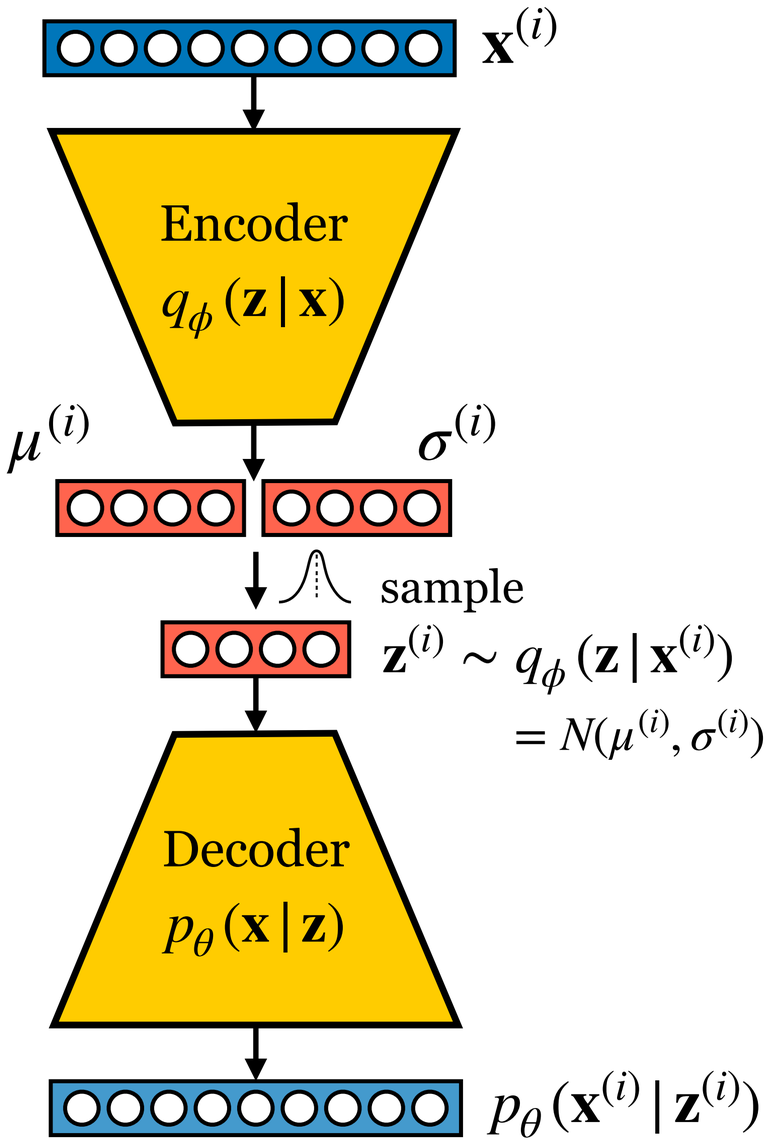
- Both parts together give rise to the loss to be minimized:

$$\mathscr{L}(\theta, \phi, X) = D_{KL}\left(q_\phi(\mathbf{z}\,|\,\mathbf{x}^{(i)}) \,\|\, p(\mathbf{z})\right) - \frac{1}{L}\sum_{l=1}^{L} \log p_\theta\left(\mathbf{x}^{(i)}\,|\,\mathbf{z}^{(i,l)}\right)$$

approx. posterior ≈ prior

"reconstruction" term

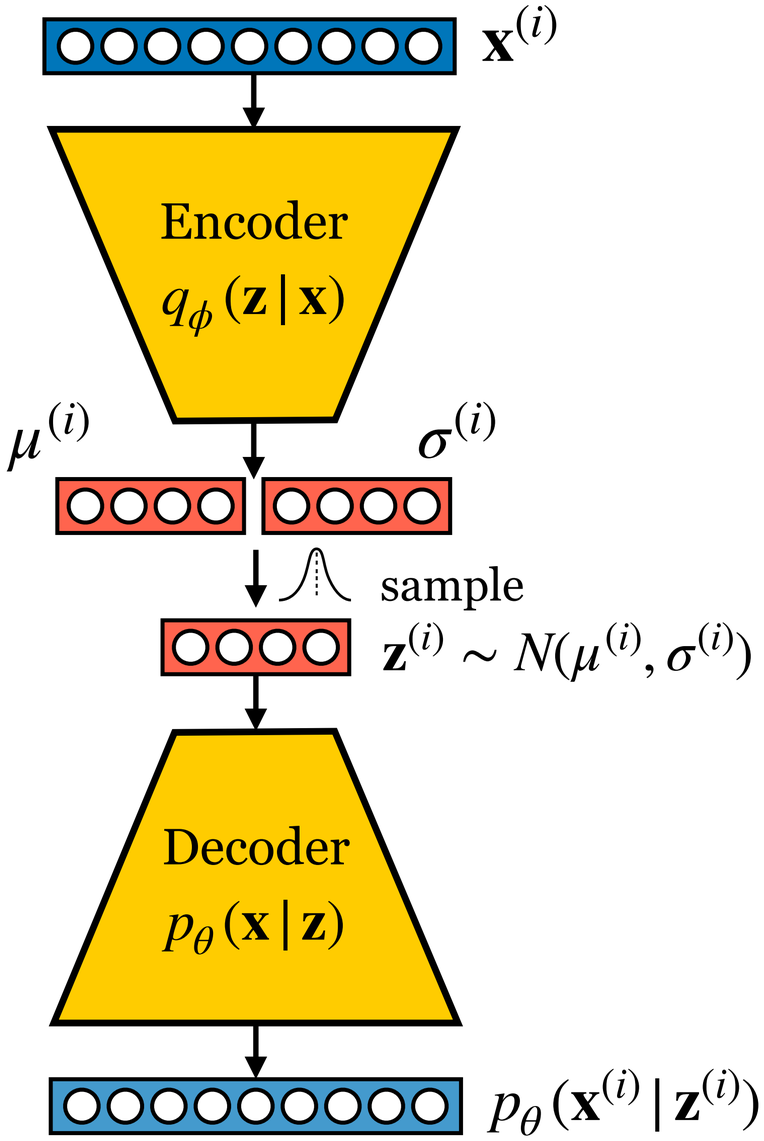
  - We want to use gradient descent on mini-batches, **backpropagating** loss gradients through both networks.



Reparametrization Trick

Problem: $\mathbf{z}$ is a sampled variable.

- We can compute $\frac{\partial \mathscr{L}}{\partial \mathbf{z}}$, but not $\frac{\partial \mathbf{z}}{\partial \mu}$ and $\frac{\partial \mathbf{z}}{\partial \sigma}$ because the sampler is not differentiable.



Reparametrization Trick

Problem: $\mathbf{z}$ is a sampled variable.

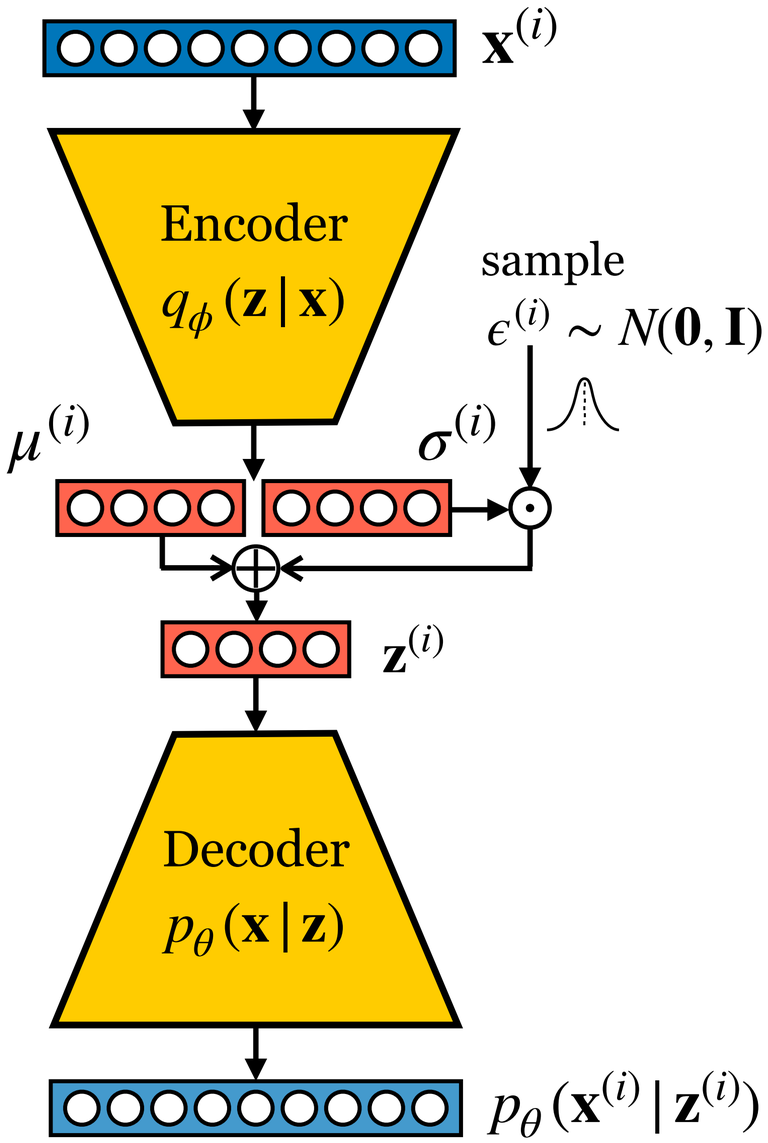
- We can compute $\frac{\partial \mathscr{L}}{\partial \mathbf{z}}$, but not $\frac{\partial \mathbf{z}}{\partial \mu}$ and $\frac{\partial \mathbf{z}}{\partial \sigma}$ because the sampler is not differentiable.

Reparametrization trick:

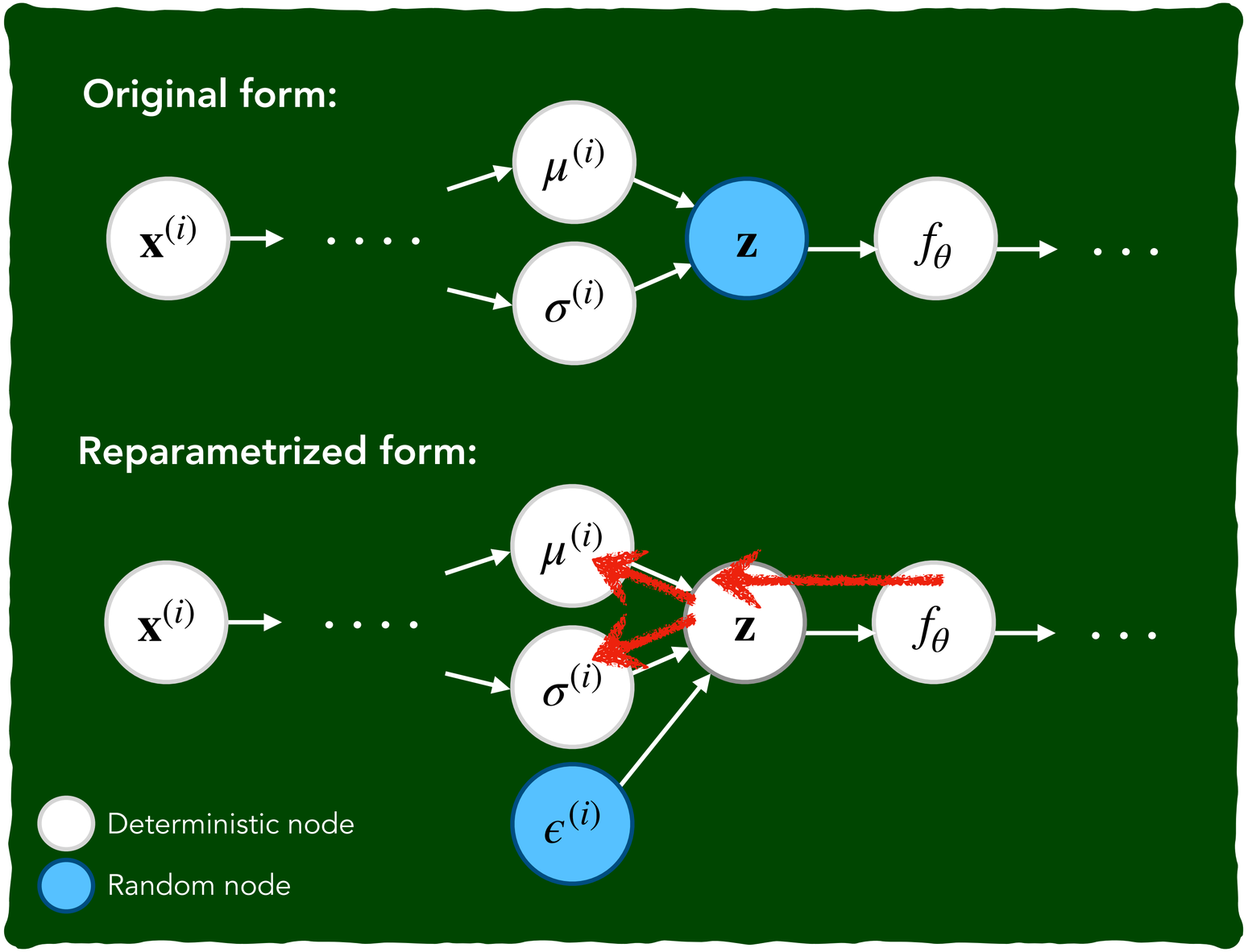
- For many distributions (e.g., for independent Gaussians) we can obtain the same distribution over $\mathbf{z}$ by including an auxiliary random variable ϵ.
- For univariate Gaussian:

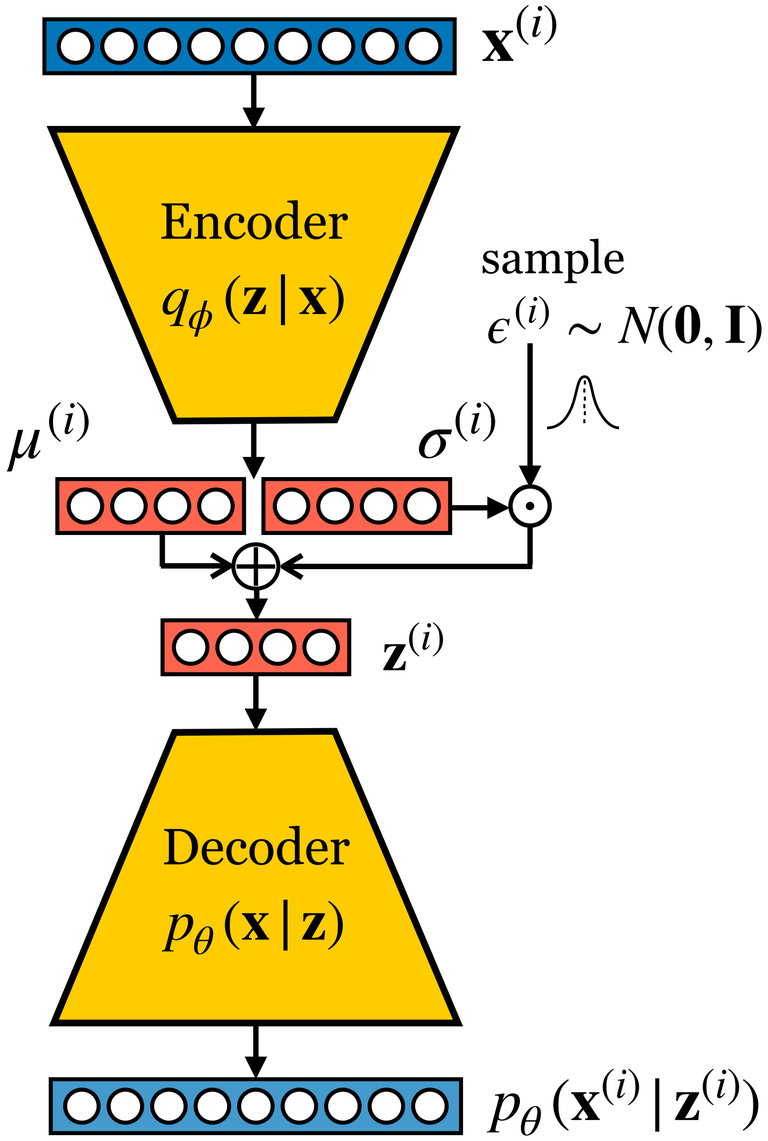
$$z^{(i,l)} = \mu\left(\mathbf{x}^{(i)}\right) + \sigma\left(\mathbf{x}^{(i)}\right)\epsilon^{(i,l)} \quad \text{with} \quad \epsilon^{(i,l)} \sim N(0,1)$$

In this case, we get $\frac{\partial \mathbf{z}}{\partial \mu} = 1$ and $\frac{\partial \mathbf{z}}{\partial \sigma} = \epsilon$

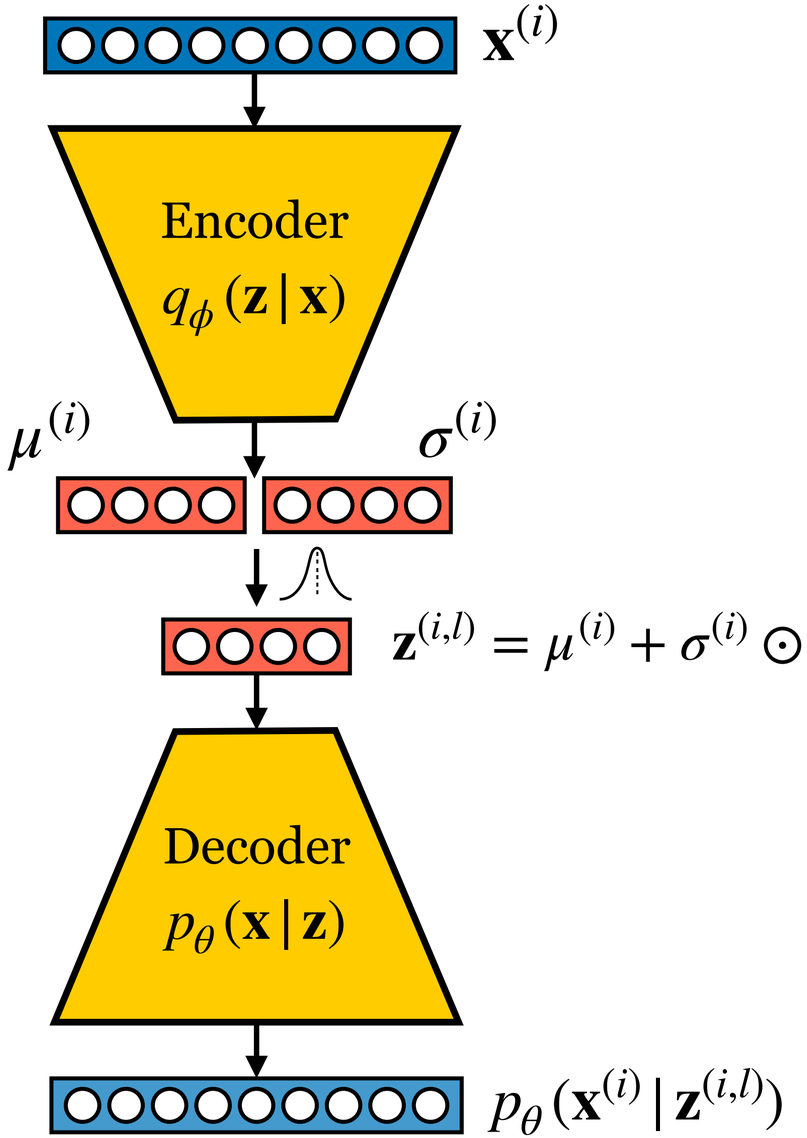


Reparametrization Trick





Variational Autoencoders: Final Model

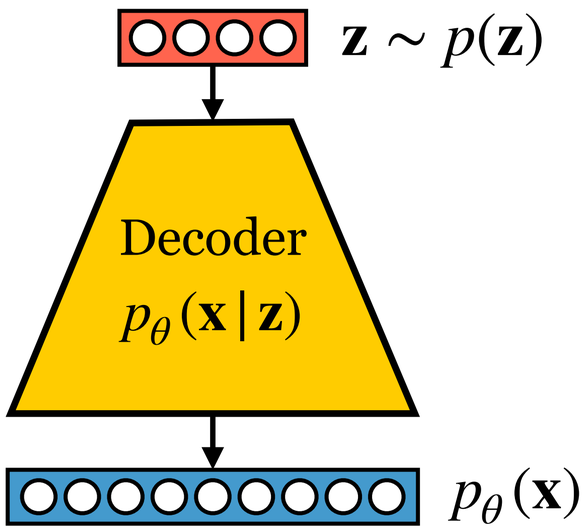


Today

- [x] Autoencoders
- [ ] Deep Generative Models
  - [ ] Variational Autoencoders
    - [x] Architecture & Training
    - [ ] Variational Lower Bound
    - [ ] Experiments & Quantifying Performance

Variational Lower Bound

Our posterior is only an approximation to $p_\theta(\mathbf{z}\,|\,\mathbf{x})$.



Background on variational inference:

The problem to invert a directed graphical model is a general problem.

- Given: A model for $p_\theta(\mathbf{x}\,|\,\mathbf{z})$
- What we want: $p_\theta(\mathbf{z}\,|\,\mathbf{x})$

Suppose we are given an intractable probability distribution p.

Variational techniques try to solve an optimization problem over a class of tractable distributions Q in order to find a $q \in Q$ that is most similar to p. We will then query q (rather than p) in order to get an approximate solution.

Variational Lower Bound

Justification for our approach:

- Mainly we want to maximize $\log p_\theta(\mathbf{x})$. We can write this log-likelihood as:

$$\log p_\theta(\mathbf{x}) = D_{KL}\left(q_\phi(\mathbf{z}\,|\,\mathbf{x}) \,\|\, p_\theta(\mathbf{z}\,|\,\mathbf{x})\right) + \mathscr{L}(\theta, \phi, \mathbf{x})$$

with $\mathscr{L}(\theta, \phi, X)$ being the **"variational lower bound"** as the (negated) loss objective from before:

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

sometimes also called:
"evidence lower bound" (ELBO)

Variational Lower Bound

$$\log p_\theta(\mathbf{x}) = D_{KL}\left(q_\phi(\mathbf{z}|\mathbf{x}) \,\|\, p_\theta(\mathbf{z}|\mathbf{x})\right) + \mathscr{L}(\theta, \phi, \mathbf{x}) \qquad \mathscr{L}(\theta, \phi, X) = -D_{KL}\left(q_\phi(\mathbf{z}|\mathbf{x}) \,\|\, p(\mathbf{z})\right) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right]$$

$$D_{KL}\left(q_\phi(\mathbf{z}|\mathbf{x}) || p_\theta(\mathbf{z}|\mathbf{x})\right) - D_{KL}\left(q_\phi(\mathbf{z}|\mathbf{x}) || p(\mathbf{z})\right) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right]$$

$$= \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})} dz - \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p(\mathbf{z})} dz + \int q_\phi(\mathbf{z}|\mathbf{x}) \log p_\theta(\mathbf{x}|\mathbf{z}) dz$$

$$= \int q_\phi(\mathbf{z}|\mathbf{x}) \left[\log q_\phi(\mathbf{z}|\mathbf{x}) - \log p_\theta(\mathbf{z}|\mathbf{x}) - \log q_\phi(\mathbf{z}|\mathbf{x}) + \log p(\mathbf{z}) + \log p_\theta(\mathbf{x}|\mathbf{z})\right] dz$$

$$= \int q_\phi(\mathbf{z}|\mathbf{x}) \left[\log \frac{p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})}\right] dz = \int q_\phi(\mathbf{z}|\mathbf{x}) \log p_\theta(\mathbf{x}) \, dz = \log p_\theta(\mathbf{x}) \int q_\phi(\mathbf{z}|\mathbf{x}) \, dz = \log p_\theta(\mathbf{x}) \quad \square$$

Variational Lower Bound

Justification for our approach:

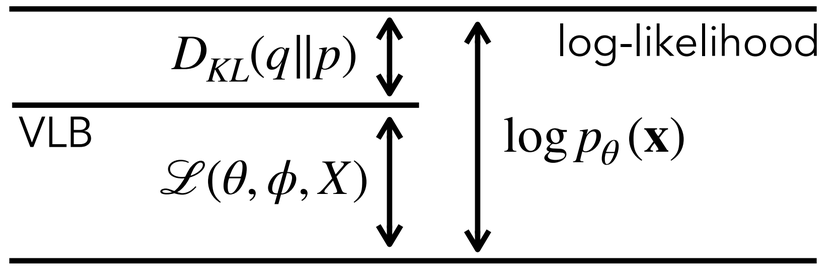
- Mainly we want to maximize $\log p_\theta(\mathbf{x})$. We can write this log-likelihood as:

$$\log p_\theta(\mathbf{x}) = D_{KL}\left(q_\phi(\mathbf{z}\,|\,\mathbf{x}) \,\|\, p_\theta(\mathbf{z}\,|\,\mathbf{x})\right) + \mathscr{L}(\theta, \phi, \mathbf{x})$$

with $\mathscr{L}(\theta, \phi, X)$ being the **"variational lower bound"** as the (negated) loss objective from before:

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

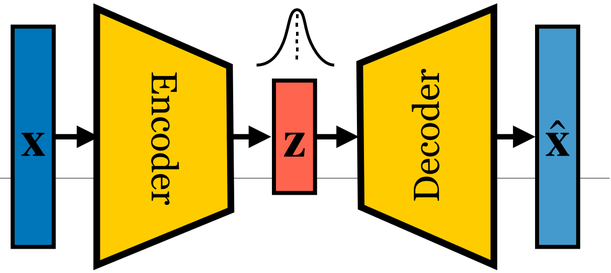
- By optimizing this, we actually maximize a variational lower bound on $\log p_\theta(\mathbf{x})$ because $D_{KL}(q_\phi(\mathbf{z}\,|\,\mathbf{x})\,||\,p_\theta(\mathbf{z}\,|\,\mathbf{x}))$ is always non-negative, i.e., $\log p_\theta(\mathbf{x}) \geq \mathscr{L}(\theta, \phi, X)$



Today

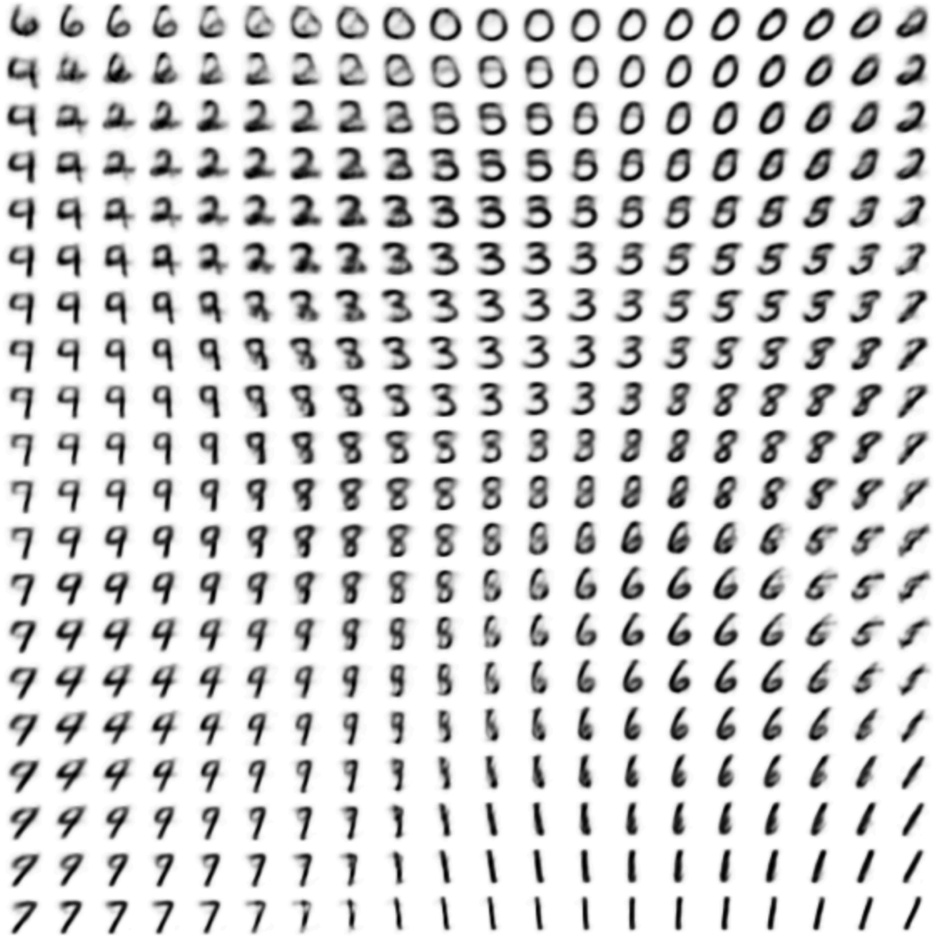
- [x] Autoencoders
- [ ] Deep Generative Models
  - [ ] Variational Autoencoders
    - [x] Architecture & Training
    - [x] Variational Lower Bound
    - [ ] Experiments & Quantifying Performance

Experiments



Latent manifold representation for two-dimensional latent space:

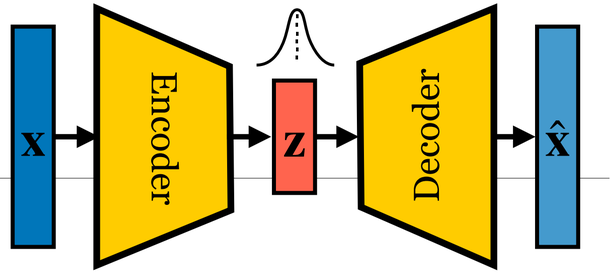
(a) Learned Frey Face manifold



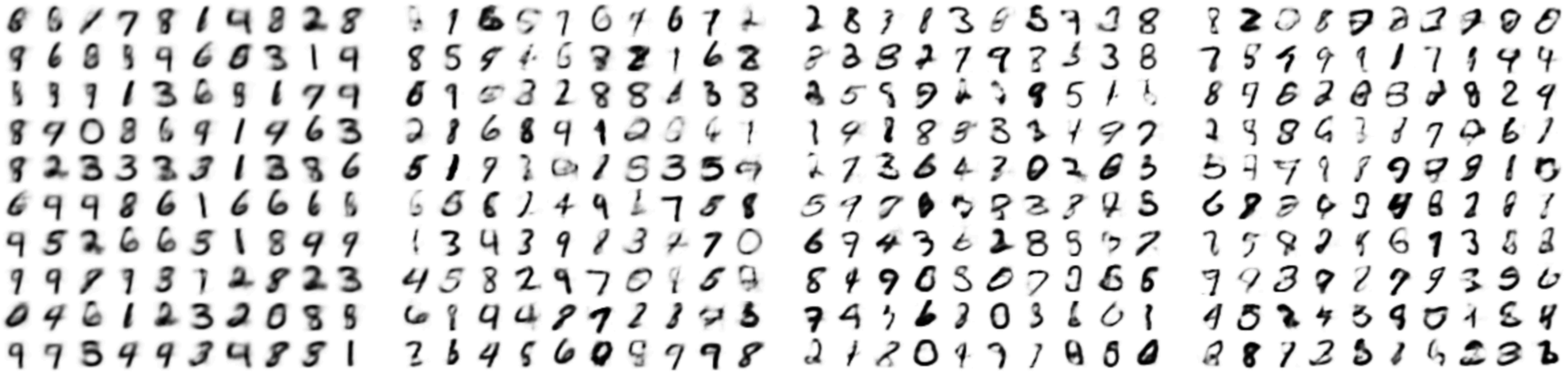
(b) Learned MNIST manifold

[Diederik P. Kingma & Max Welling, "Auto-Encoding Variational Bayes", arXiv preprint arXiv:1312.6114 (2013).]

Experiments



Samples from generative model trained on MNIST:



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

[Diederik P. Kingma & Max Welling, "Auto-Encoding Variational Bayes", arXiv preprint arXiv:1312.6114 (2013).]

Quantifying Performance

Quantitative measures for success of generative models.

- **Test likelihood:** Measure their log-likelihood for a test dataset.
  - Not always possible to compute exactly and efficiently (e.g., not possible for GANs).
  - For VAEs: only possible to compute the lower bound on the log-likelihood.

Quantifying Performance

Quantitative measures for success of generative models.

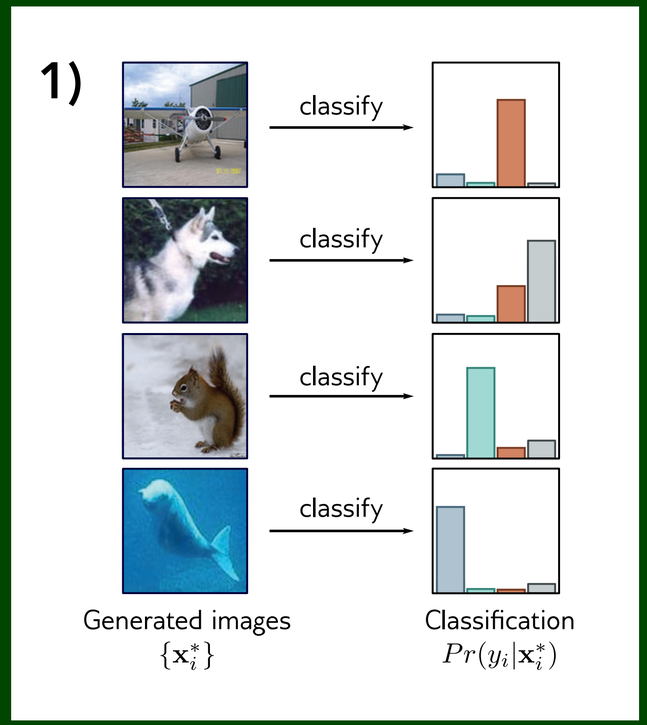
- **Test likelihood:** Measure their log-likelihood for a test dataset.
  - Not always possible to compute exactly and efficiently (e.g., not possible for GANs).
  - For VAEs: only possible to compute the lower bound on the log-likelihood.
- **Inception Score (IS):** Calculated using a pre-trained CNN, e.g., “Inception-v3”, on ImageNet.
  - Sensitive to the classifier, retraining it can give different results.

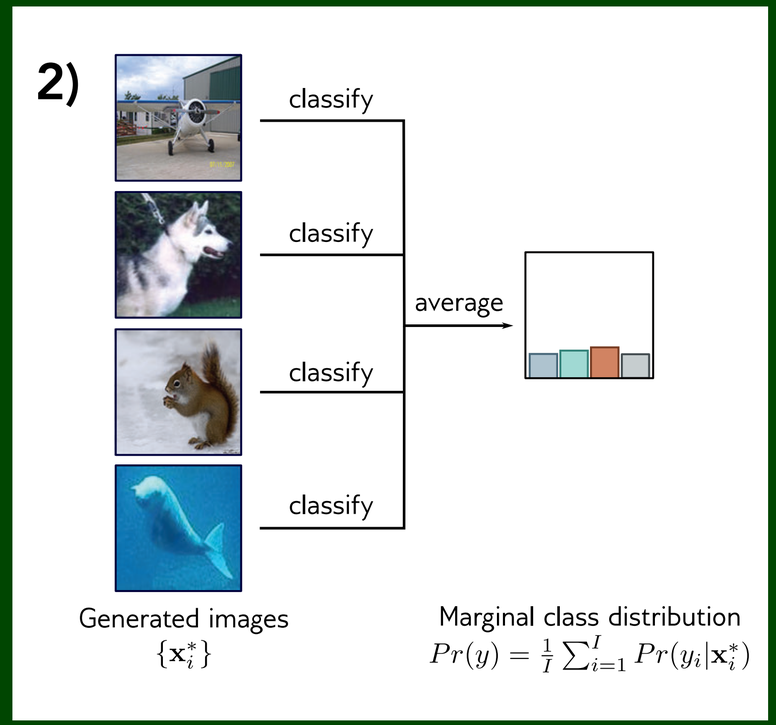
  1) Samples should look like only one of the 1000 ImageNet classes (sharpness).
  2) Model generates samples from all classes equally frequently (diversity).

Quantifying Performance

Quantitative measures for s

- **Test likelihood:** Measure th
  - Not always possible to c
  - For VAEs: only possible
- **Inception Score (IS):** Calcu
  - Sensitive to the classifier
    1) Samples should look l
    2) Model generates sam





Inception Score: $$\mathbf{IS} = \exp\left(\frac{1}{I}\sum_{i=1}^{I} D_{KL}\left[Pr(y_i \mid \mathbf{x}_i^*) \parallel Pr(y)\right]\right)$$

Quantifying Performance

Quantitative measures for success of generative models.

- **Test likelihood:** Measure their log-likelihood for a test dataset.
  - Not always possible to compute exactly and efficiently (e.g., not possible for GANs).
  - For VAEs: only possible to compute the lower bound on the log-likelihood.
- **Inception Score (IS):** Calculated using a pre-trained CNN, e.g., “Inception-v3”, on ImageNet.
  - Sensitive to the classifier, retraining it can give different results.
    1) Samples should look like only one of the 1000 ImageNet classes (sharpness).
    2) Model generates samples from all classes equally frequently (diversity).
- **Fréchet Inception Distance (FID):** Measures similarities between the feature representations (learned by a pre-trained classifier) for generated samples and a test dataset.
- …

Summary

- Variational autoencoders are encoder-decoder architectures.
- Training is performed by maximizing a variational lower bound on the log-likelihood.
- VAEs can be used as generative models or as feature extractors.
- Individual features of the latent space often encode specific parameters of the input.
  - e.g. facial expression, object rotation, etc.
  - By changing the latent variable, one can transform images (e.g., make a person smile).
- Often, encoders are CNNs and decoders use transposed convolutions.
- Some (indirect) measures to quantify performance exist (e.g., Inception score).
- VAEs are easier to train, but Generative Adversarial Networks are usually better generators.

[Ian Goodfellow et al. “Generative Adversarial Nets.” NIPS 2014.]

Today

- ☑ Autoencoders
- ☐ Deep Generative Models
  - ☑ Variational Autoencoders
    - ☑ Architecture & Training
    - ☑ Variational Lower Bound
    - ☑ Experiments & Quantifying Performance

Questions?