DSC291: Machine Learning with Few Labels

Variational Inference
Variational Autoencoders

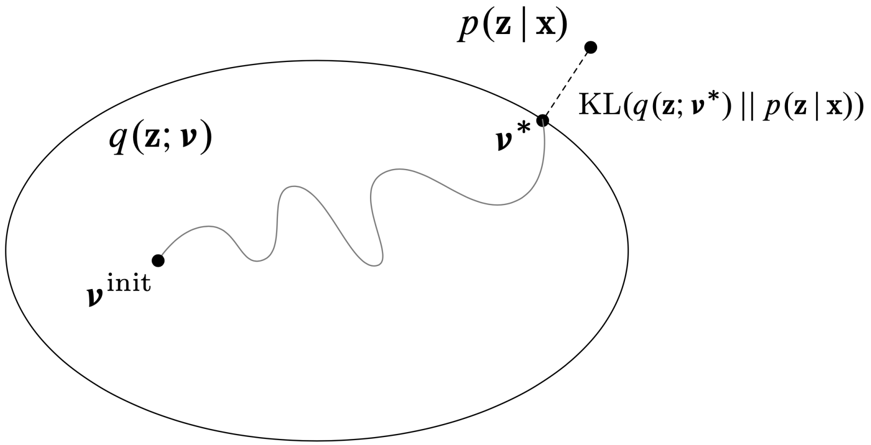
Zhiting Hu

Lecture 8, January 27, 2023

UC San Diego
HALICIOĞLU DATA SCIENCE INSTITUTE

Recap: VI

- We often cannot compute posteriors, and so we need to approximate them, using variational methods.
- In variational Bayes, we'd like to find an approximation within some family that minimizes the KL divergence to the posterior, but we can't directly minimize this
- Therefore, we defined the ELBO, which we can maximize, and this is equivalent to minimizing the KL divergence.



Evidence Lower Bound (ELBO)

$$\ell(\theta; \boldsymbol{x}) = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x}, \boldsymbol{z}|\theta)}{q(\boldsymbol{z}|\boldsymbol{x})}\right] + \mathrm{KL}\big(q(\boldsymbol{z}|\boldsymbol{x}) \,||\, p(\boldsymbol{z}|\boldsymbol{x}, \theta)\big)$$

Recap: Mean-Field VI

- We defined a family of approximations called "mean field" approximations, in which there are no dependencies between latent variables

$$q(\boldsymbol{z}) = q(z_1, \ldots, z_m) = \prod_{j=1}^{m} q(z_j)$$

- We optimize the ELBO with coordinate ascent updates to iteratively optimize each local variational approximation under mean field assumptions

$$q^*(z_j) \propto \exp\left\{ \mathbb{E}_{q_{-j}}[\log p(\boldsymbol{x}, \boldsymbol{z})] \right\}$$

Recap: VI with coordinate ascent

Example: Bayesian mixture of Gaussians

Assume mean-field $q(\mu_{1:K}, \pi, z_{1:n}) = \prod_k q(\mu_k) q(\pi) \prod_i q(z_i)$

- Initialize the global variational distributions $q(\mu_k)$ and $q(\pi)$
- Repeat:
  - For each data example $i \in \{1, 2, \ldots, D\}$
    - Update the local variational distribution $q(z_i)$
  - End for
  - Update the global variational distributions $q(\mu_k)$ and $q(\pi)$
- Until ELBO converges

- What if we have millions of data examples? This could be very slow.

Recap: Stochastic VI

Example: Bayesian mixture of Gaussians

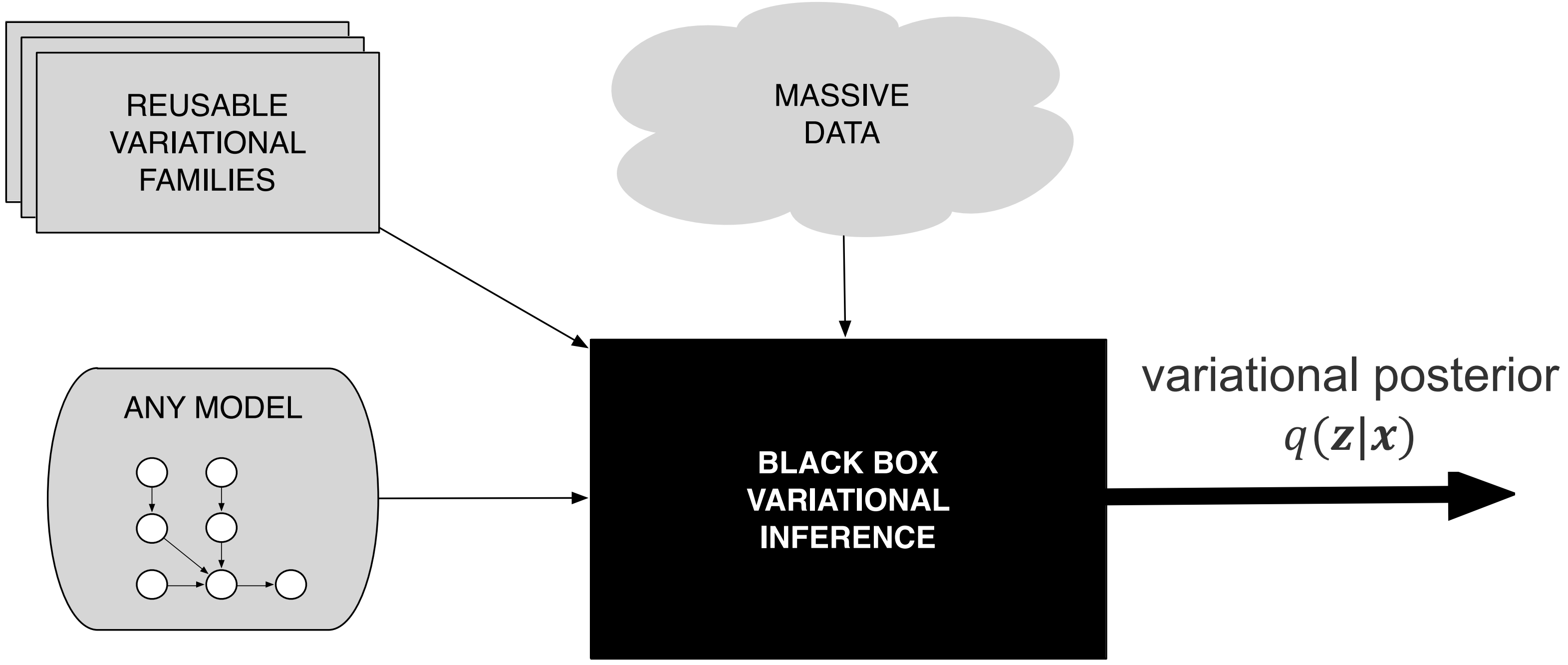
Assume mean-field $q(\mu_{1:K}, \pi, z_{1:n}) = \prod_k q(\mu_k) q(\pi) \prod_i q(z_i)$

- Initialize the global variational distributions $q(\mu_k)$ and $q(\pi)$
- Repeat:
    - Sample a data example $i \in \{1,2, \dots, D\}$
    - Update the local variational distribution $q(z_i)$
    - Update the global variational distributions $q(\mu_k)$ and $q(\pi)$ with **natural gradient ascent**
- Until ELBO converges

- (Setting natural gradient $= 0$ gives the traditional mean-field update)

[Hoffman et al., Stochastic Variational Inference, 2013]

Recap: Black-box Variational Inference (BBVI)



- Easily use variational inference with **any model**
- Perform inference with **massive data**
- **No mathematical work** beyond specifying the model

(Courtesy: Blei et al., 2018)

Recap: Black-box Variational Inference (BBVI)

- Probabilistic model: $\boldsymbol{x}$ -- observed variables, $\boldsymbol{z}$ -- latent variables
- Variational distribution $q_\lambda(\boldsymbol{z}|\boldsymbol{x})$ with parameters λ, e.g.,
  - Gaussian mixture distribution:
    - "A Gaussian mixture model is a universal approximator of densities, in the sense that any smooth density can be approximated with any specific nonzero amount of error by a Gaussian mixture model with enough components." (Deep Learning book, pp.65)
  - Deep neural networks
- ELBO:

$$\mathcal{L}(\lambda) = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{\lambda})}[\log p(\boldsymbol{x}, \boldsymbol{z})] - \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{\lambda})}[\log q(\boldsymbol{z}|\lambda)]$$

- Want to compute the gradient w.r.t variational parameters λ

[Ranganath et al.,14]

Recap: The General Problem: Computing Gradients of Expectations

- When the objective function $\mathcal{L}$ is defined as an expectation of a (differentiable) test function $f_\lambda(\boldsymbol{z})$ w.r.t. a probability distribution $q_\lambda(\boldsymbol{z})$

$$\mathcal{L} = \mathbb{E}_{q_\lambda(\boldsymbol{z})}[f_\lambda(\boldsymbol{z})]$$

- Computing exact gradients w.r.t. the parameters λ is often unfeasible
- Need stochastic gradient estimates
  - The score function estimator (a.k.a log-derivative trick, REINFORCE)
  - The reparameterization trick (a.k.a the pathwise gradient estimator)

Recap: Computing Gradients of Expectations w/ score function

- Loss: $\mathcal{L} = \mathbb{E}_{q_{\lambda}(\boldsymbol{z})}[f_{\lambda}(\boldsymbol{z})]$

- Log-derivative trick: $\nabla_{\lambda} q_{\lambda} = q_{\lambda} \, \nabla_{\lambda} \log q_{\lambda}$
- Gradient w.r.t. λ:

$$\nabla_{\lambda}\mathcal{L} = \mathbb{E}_{q_{\lambda}(\boldsymbol{z})}[f_{\lambda}(\boldsymbol{z}) \underline{\nabla_{\lambda} \log q_{\lambda}(\boldsymbol{z})} + \nabla_{\lambda} f_{\lambda}(\boldsymbol{z})]$$

  - score function: the gradient of the log of a probability distribution
- Compute noisy unbiased gradients with Monte Carlo samples from q_{λ}

$$\nabla_{\lambda}\mathcal{L} \approx \frac{1}{S}\sum_{s=1}^{S} f_{\lambda}(\boldsymbol{z}_s) \nabla_{\lambda} \log q_{\lambda}(\boldsymbol{z}_s) + \nabla_{\lambda} f_{\lambda}(\boldsymbol{z}_s) \qquad \text{where } \boldsymbol{z}_s \sim q_{\lambda}(\boldsymbol{z})$$

- Pros: generally applicable to any distribution $q(z|\lambda)$
- Cons: empirically has high variance → slow convergence
  - To reduce variance: Rao-Blackwellization, control variates, importance sampling, ...

Computing Gradients of Expectations w/ reparametrization trick

- Loss: $\mathcal{L} = \mathbb{E}_{q_\lambda(\boldsymbol{z})}[f_\lambda(\boldsymbol{z})]$
- Assume that we can express the distribution $q_{\boldsymbol{\lambda}}(\boldsymbol{z})$ with a transformation

$$\begin{aligned} \epsilon &\sim s(\epsilon) \\ z &= t(\epsilon, \lambda) \end{aligned} \quad \Leftrightarrow \quad z \sim q(z|\lambda)$$

  - E.g.,

$$\begin{aligned} \epsilon &\sim Normal(0, 1) \\ z &= \epsilon\sigma + \mu \end{aligned} \quad \Leftrightarrow \quad z \sim Normal(\mu, \sigma^2)$$

- Reparameterization gradient

$$\mathcal{L} = \mathbb{E}_{\boldsymbol{\epsilon} \sim s(\boldsymbol{\epsilon})}[f_\lambda(\boldsymbol{z}(\boldsymbol{\epsilon}, \lambda))]$$

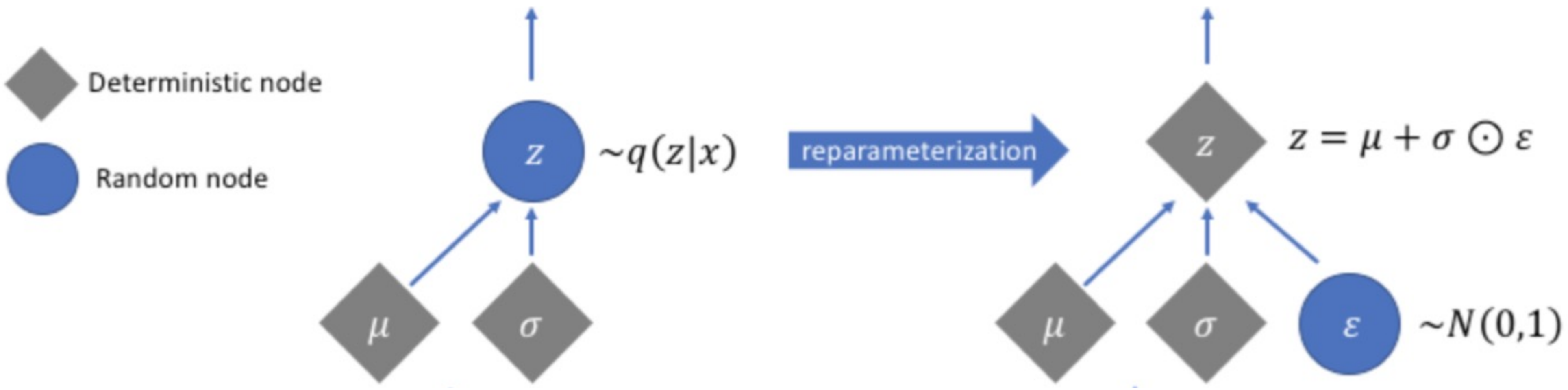
$$\nabla_\lambda \mathcal{L} = \mathbb{E}_{\boldsymbol{\epsilon} \sim \boldsymbol{s}(\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{z}} f_\lambda(\boldsymbol{z}) \, \nabla_\lambda t(\epsilon, \lambda)]$$

- Pros: empirically, lower variance of the gradient estimate
- Cons: Not all distributions can be reparameterized

Reparameterization trick

- Reparametrizing Gaussian distribution

$$\begin{aligned} &\epsilon \sim Normal(0,1) \\ &z = \epsilon\sigma + \mu \end{aligned} \iff z \sim Normal(\mu, \sigma^2)$$



[Courtesy: Tansey, 2016]

Reparameterization trick

- Reparametrizing Gaussian distribution

$$\begin{aligned} \epsilon &\sim Normal(0,1) \\ z &= \epsilon\sigma + \mu \end{aligned} \quad \Leftrightarrow \quad z \sim Normal(\mu, \sigma^2)$$

- Other reparameterizable distributions: $\begin{aligned} \epsilon &\sim Uniform(\epsilon) \\ z &= F^{-1}(\epsilon) \end{aligned} \quad \Leftrightarrow \quad z \sim q(z)$
  - Tractable inverse CDF F^{-1}:
    - Exponential, Cauchy, Logistic, Rayleigh, Pareto, Weibull, Reciprocal, Gompertz, Gumbel, Erlang
  - Location-scale:
    - Laplace, Elliptical, Student's t, Logistic, Uniform, Triangular, Gaussian
  - Composition:
    - Log-Normal (exponentiated normal) Gamma (sum of exponentials) Dirichlet (sum of Gammas) Beta, Chi-Squared, F

[Courtesy: Tansey, 2016]

Computing Gradients of Expectations: Summary

- Loss: $\mathcal{L} = \mathbb{E}_{q_\lambda(\boldsymbol{z})}[f_\lambda(\boldsymbol{z})]$
- Score gradient

$$\nabla_\lambda \mathcal{L} = \mathbb{E}_{q_\lambda(\boldsymbol{z})}[f_\lambda(\boldsymbol{z})\nabla_\lambda \log q_\lambda(\boldsymbol{z}) + \nabla_\lambda f_\lambda(\boldsymbol{z})]$$

  - Pros: generally applicable to any distribution $q(z|\lambda)$
  - Cons: empirically has high variance → slow convergence

- Reparameterization gradient

$$\nabla_\lambda \mathcal{L} = \mathbb{E}_{\boldsymbol{\epsilon} \sim \boldsymbol{s}(\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{z}} f_\lambda(\boldsymbol{z})\, \nabla_\lambda t(\epsilon, \lambda)]$$

  - Pros: empirically, lower variance of the gradient estimate
  - Cons: Not all distributions can be reparameterized

Recall: Black-box Variational Inference (BBVI)

- Probabilistic model: $\boldsymbol{x}$ -- observed variables, $\boldsymbol{z}$ -- latent variables
- Variational distribution $q_\lambda(\boldsymbol{z}|\boldsymbol{x})$ with parameters λ, e.g.,
  - Gaussian mixture distribution:
    - "A Gaussian mixture model is a universal approximator of densities, in the sense that any smooth density can be approximated with any specific nonzero amount of error by a Gaussian mixture model with enough components." (Deep Learning book, pp.65)
  - Deep neural networks

$$\mathcal{L}(\lambda) \triangleq \mathrm{E}_{q_\lambda(z)}[\log p(x, z) - \log q(z)]$$

- ELBO:

$$\mathcal{L}(\lambda) = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{\lambda})}[\log p(\boldsymbol{x}, \boldsymbol{z})] - \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{\lambda})}[\log q(\boldsymbol{z}|\lambda)]$$

- Want to compute the gradient w.r.t variational parameters λ

[Ranganath et al.,14]

BBVI with the score gradient

- ELBO:

$$\mathcal{L}(\lambda) = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{\lambda})}[\log p(\boldsymbol{x}, \boldsymbol{z})] - \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{\lambda})}[\log q(\boldsymbol{z}|\lambda)]$$

- Gradient w.r.t. λ (using the log-derivative trick)

$$\nabla_\lambda \mathcal{L} = \mathrm{E}_q[\nabla_\lambda \log q(z|\lambda)(\log p(x, z) - \log q(z|\lambda))]$$

- Compute noisy unbiased gradients of the ELBO with Monte Carlo samples from the variational distribution

$$\nabla_\lambda \mathcal{L} \approx \frac{1}{S} \sum_{s=1}^{S} \nabla_\lambda \log q(z_s|\lambda)(\log p(x, z_s) - \log q(z_s|\lambda)),$$

$$\text{where } z_s \sim q(z|\lambda).$$

[Ranganath et al.,14]

BBVI with the reparameterization gradient

- ELBO:

$$\mathcal{L}(\lambda) = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{\lambda})}[\log p(\boldsymbol{x}, \boldsymbol{z})] - \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{\lambda})}[\ \log q(\boldsymbol{z}|\lambda)\ ]$$

- Gradient w.r.t. λ

$$\begin{aligned} \epsilon &\sim s(\epsilon) \\ z &= t(\epsilon, \lambda) \end{aligned} \quad \Leftrightarrow \quad z \sim q(z|\lambda)$$

$$\nabla_\lambda \mathcal{L} = \mathrm{E}_{\epsilon \sim s(\epsilon)}[\ \nabla_z[\log p(x, z) - \log q(z)]\ \nabla_\lambda t(\epsilon, \lambda)]$$

Variational Autoencoders (VAEs)

Variational Auto-Encoders (VAEs)

VAEs are a combination of the following ideas:

- Variational Inference
  - ELBO
- Variational distribution parametrized as neural networks
- Reparameterization trick

[Courtesy: Dhruv, CS 4803]

Variational Auto-Encoders (VAEs)

- Model $p_\theta(\boldsymbol{x}, \boldsymbol{z}) = p_\theta(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})$
  - $p_\theta(\boldsymbol{x}|\boldsymbol{z})$: a.k.a., generative model, generator, (probabilistic) decoder, …
  - $p(\boldsymbol{z})$: prior, e.g., Gaussian
- Assume variational distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$
  - E.g., a Gaussian distribution parameterized as **deep neural networks**
  - a.k.a, recognition model, inference network, (probabilistic) encoder, …
- ELBO:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) = \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}, \boldsymbol{z})] + \mathrm{H}(q_\phi(\boldsymbol{z}|\boldsymbol{x}))$$

$$= \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - \mathrm{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \,||\, p(\boldsymbol{z}))$$

Reconstruction

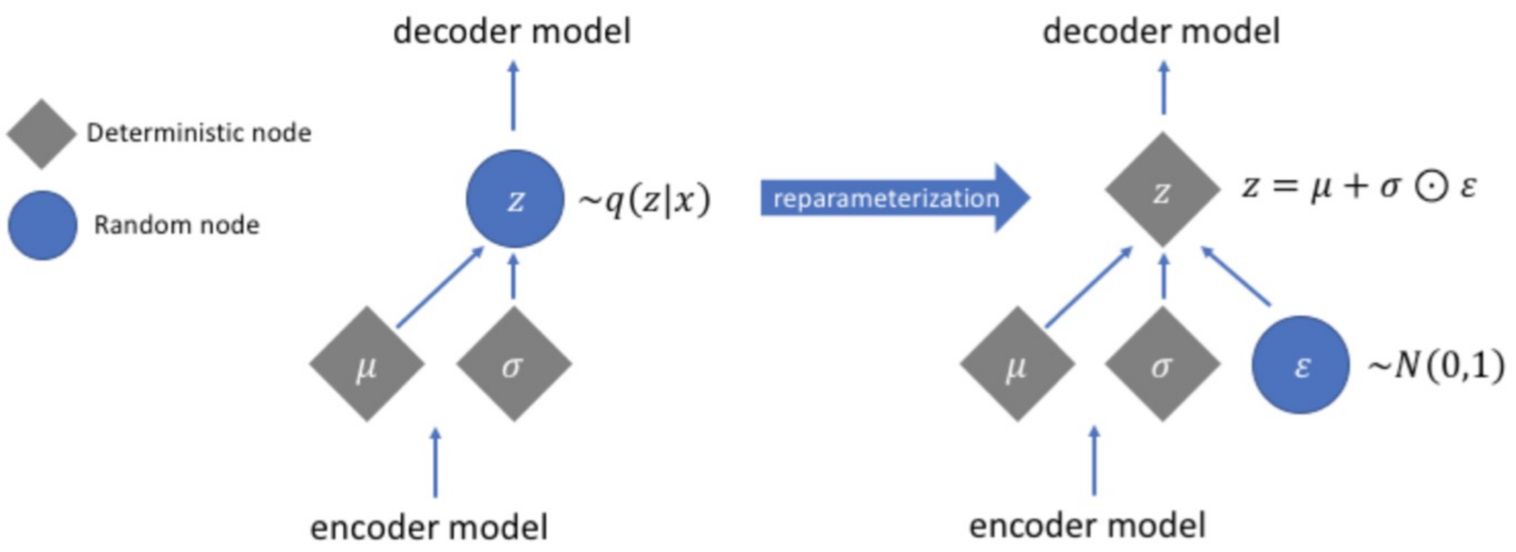
Divergence from prior
(KL divergence between two Guassians
has an analytic form)

Variational Auto-Encoders (VAEs)

- ELBO:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) = \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}, \boldsymbol{z})] + \mathrm{H}(q_\phi(\boldsymbol{z}|\boldsymbol{x}))$$
$$= \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - \mathrm{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \,||\, p(\boldsymbol{z}))$$

- Reparameterization:
  - $[\boldsymbol{\mu}; \boldsymbol{\sigma}] = f_\phi(\boldsymbol{x})$ (a neural network)
  - $\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \boldsymbol{N}(\mathbf{0}, \mathbf{1})$



Variational Auto-Encoders (VAEs)

- ELBO:

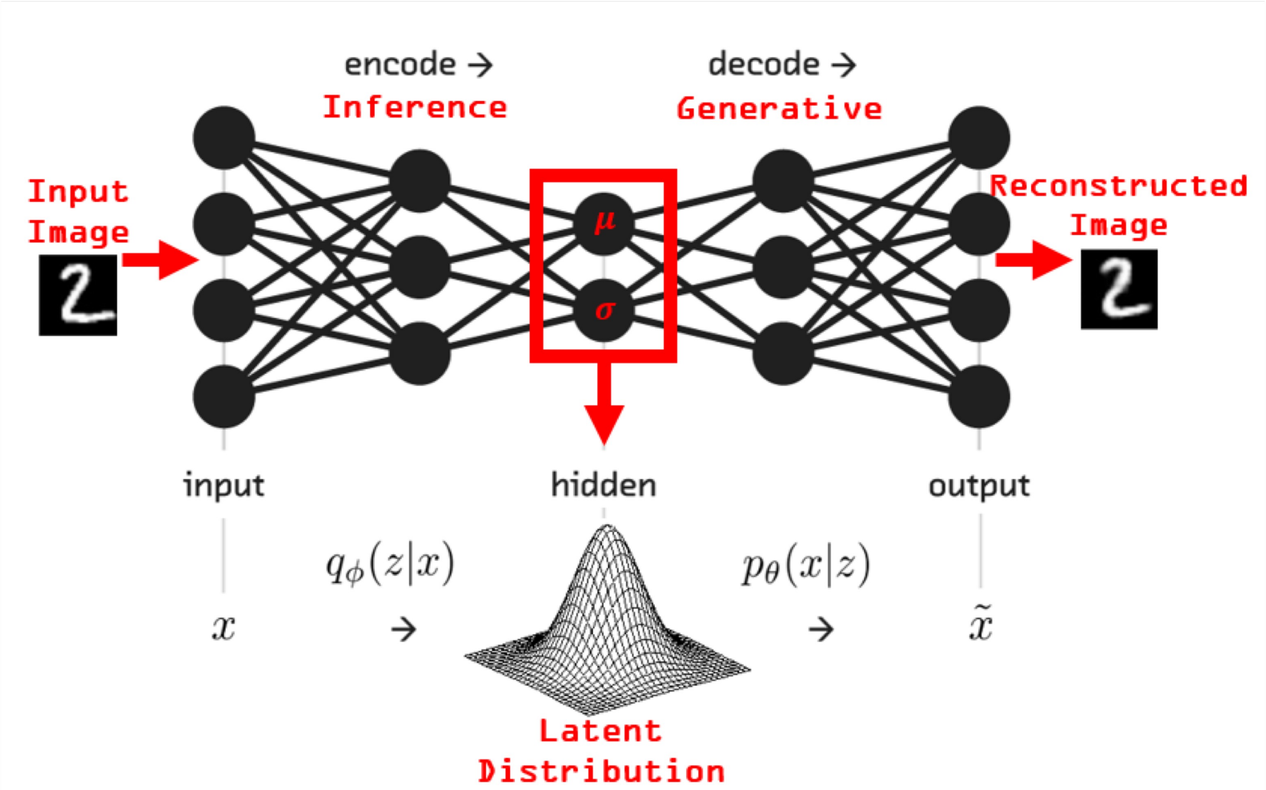
$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) = \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}, \boldsymbol{z})] + \mathrm{H}(q_\phi(\boldsymbol{z}|\boldsymbol{x}))$$
$$= \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\,[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] \; - \mathrm{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \,||\, p(\boldsymbol{z}))$$

- Reparameterization:
  - $[\boldsymbol{\mu}; \boldsymbol{\sigma}] = f_\phi(\boldsymbol{x})$ (a neural network)
  - $\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \boldsymbol{N}(\boldsymbol{0}, \boldsymbol{1})$

$$\nabla_{\boldsymbol{\phi}}\mathcal{L} = \mathrm{E}_{\epsilon \sim N(\boldsymbol{0},\boldsymbol{1})}[\, \nabla_{\boldsymbol{z}}\big[\log p_\theta(\boldsymbol{x}, \boldsymbol{z}) - \log q_\phi(\boldsymbol{z}|\boldsymbol{x})\big]\, \nabla_\phi z(\epsilon, \boldsymbol{\phi})]$$

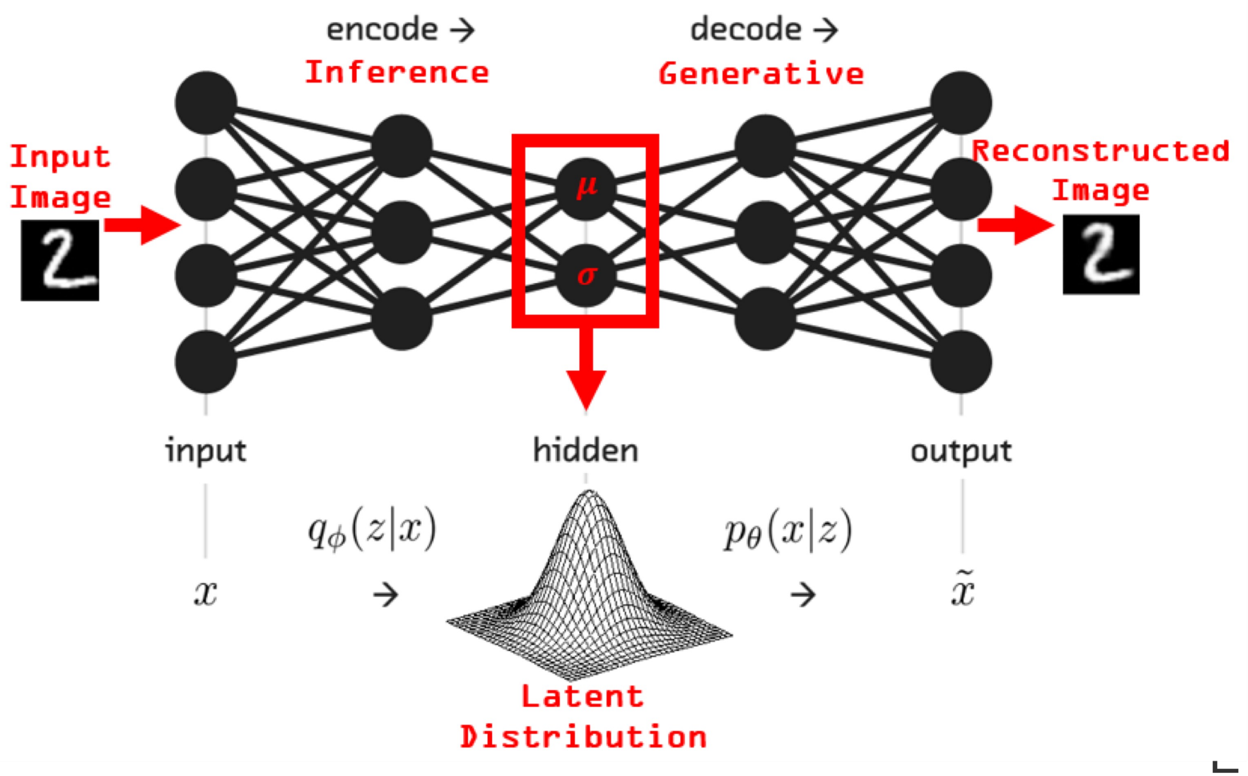
$$\nabla_\theta\mathcal{L} = \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\nabla_\theta \log p_\theta(\boldsymbol{x}, \boldsymbol{z})]$$

Example: VAEs for images



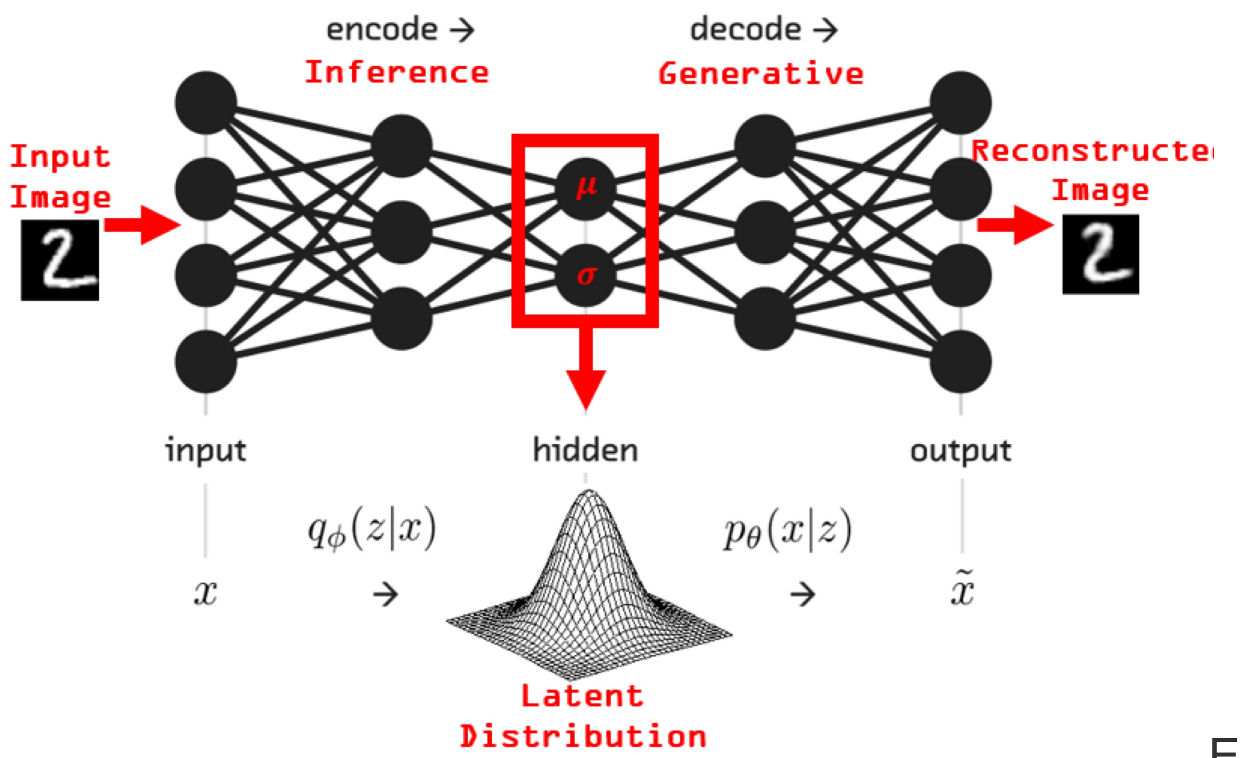
[https://www.kaggle.com/rvislaywade/visualizing-mnist-using-a-variational-autoencoder]

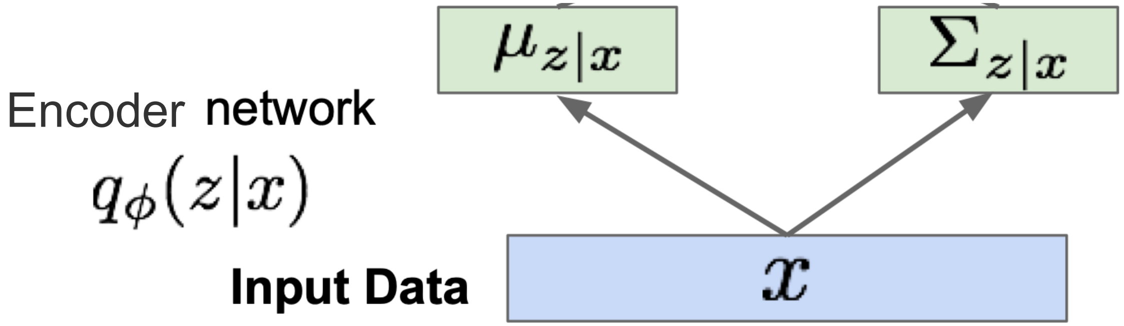
Example: VAEs for images



[Courtesy: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n]

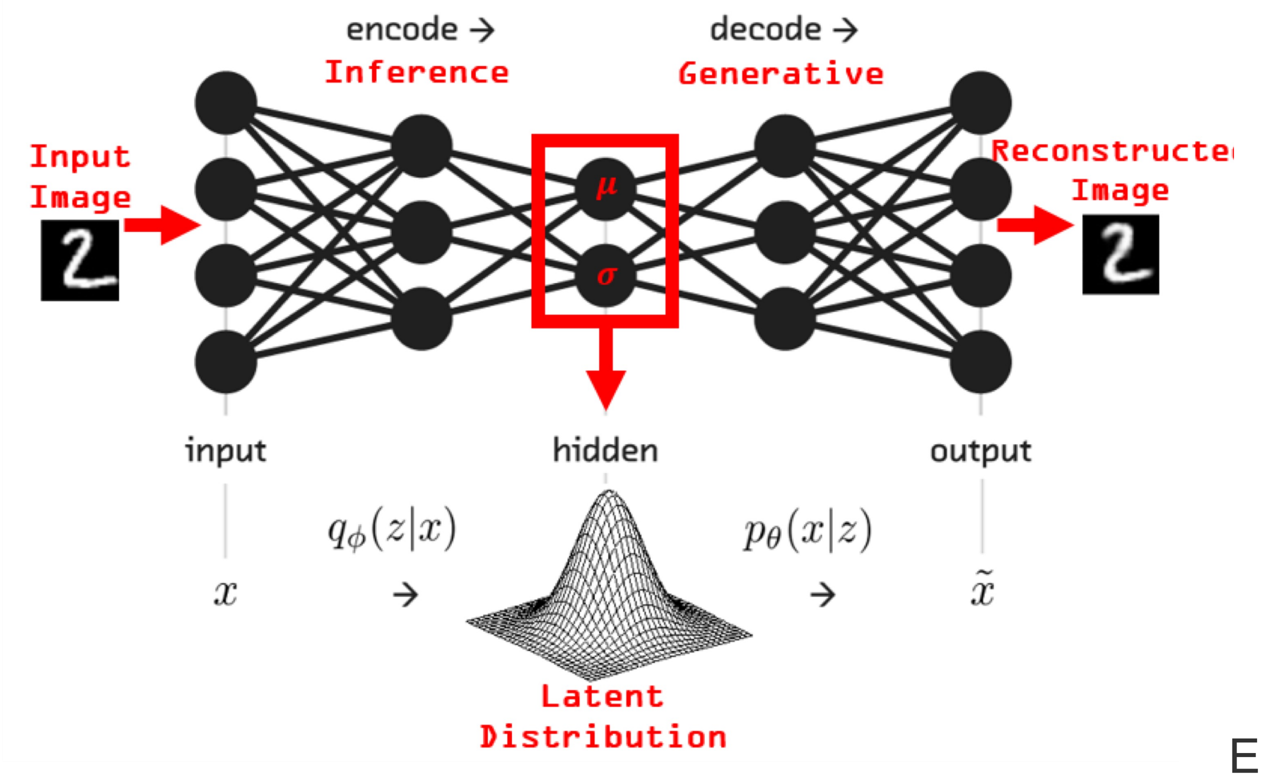
Example: VAEs for images

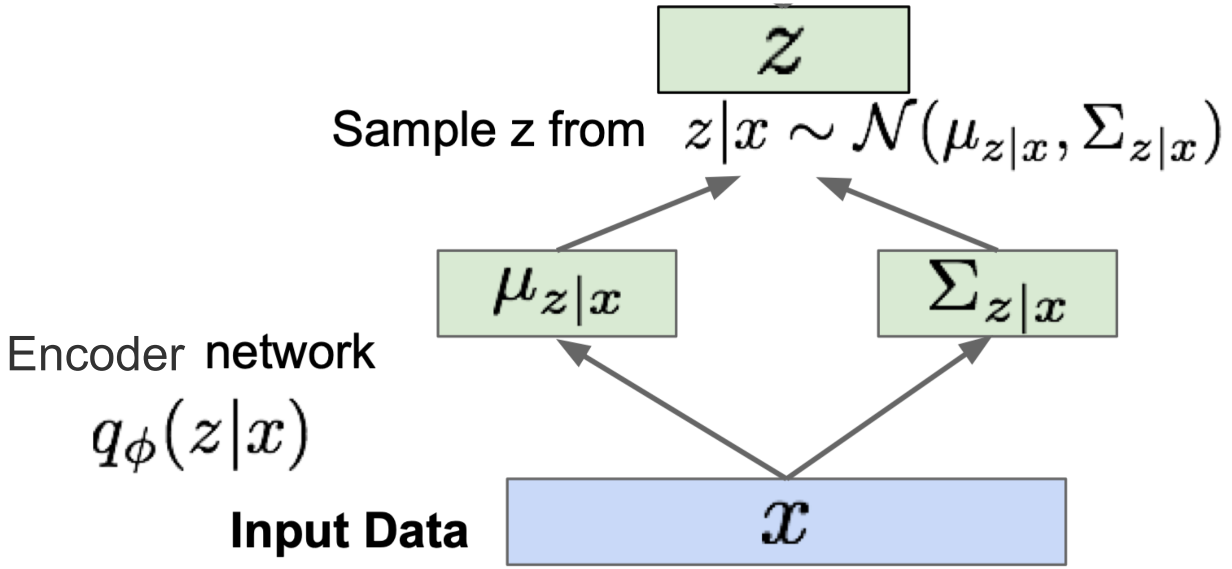




[Courtesy: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n]

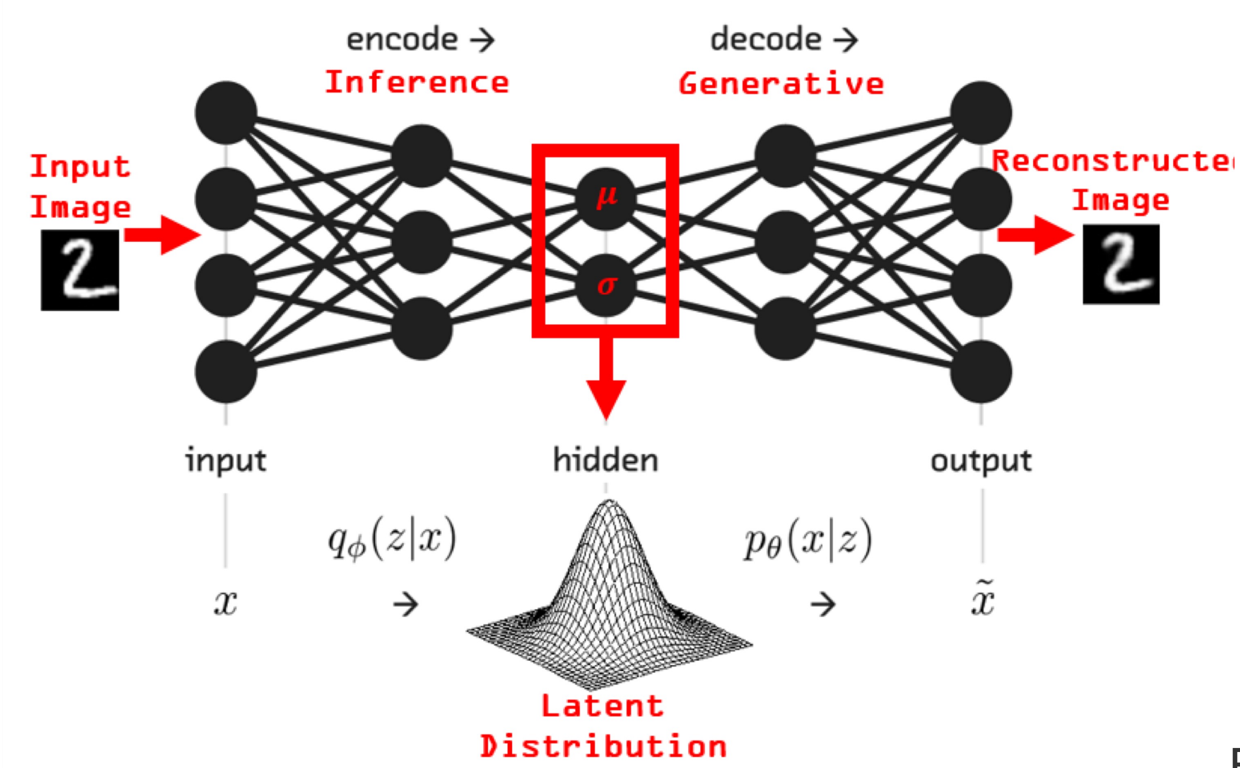
Example: VAEs for images

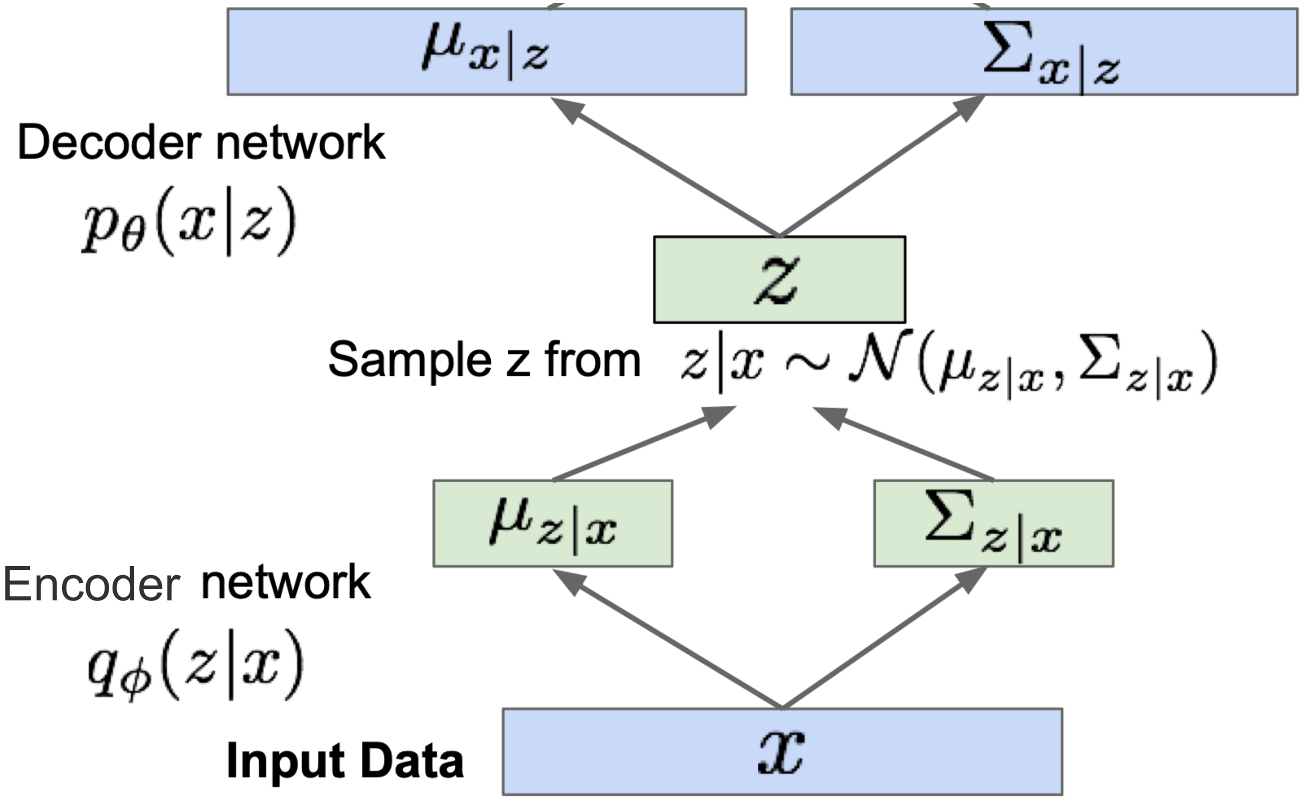




[Courtesy: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n]

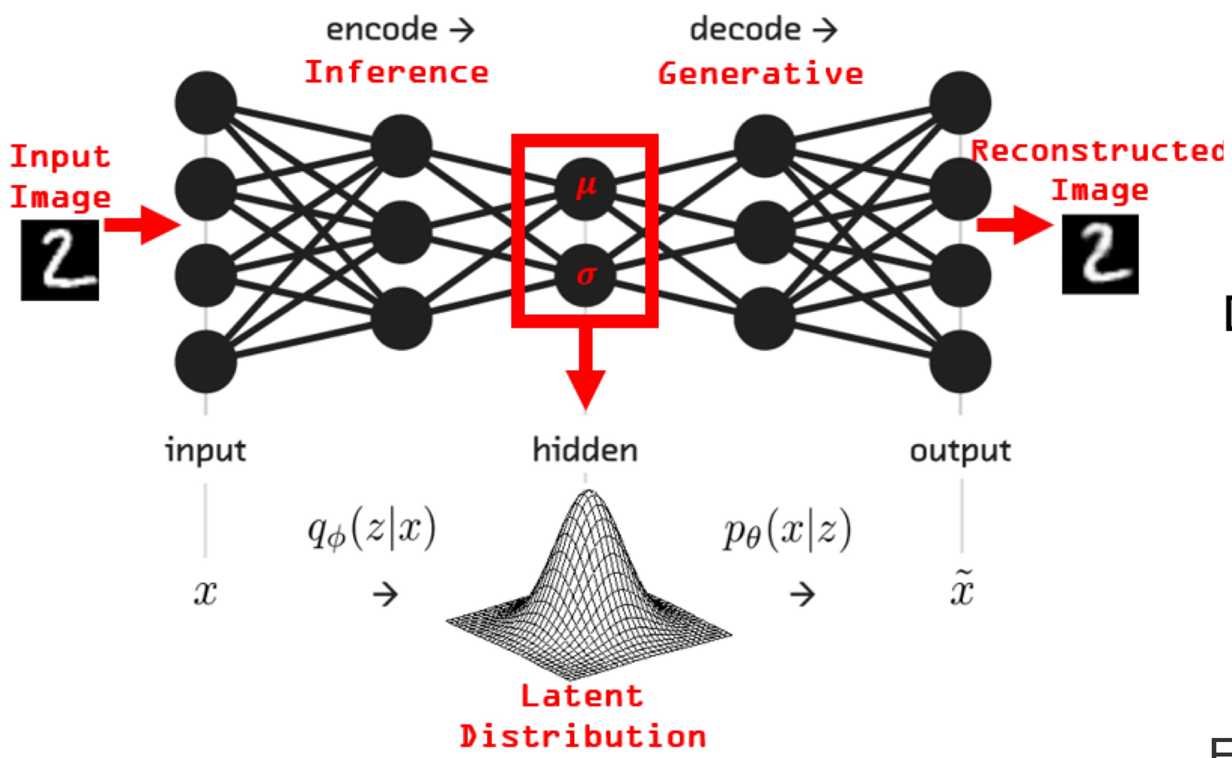
Example: VAEs for images

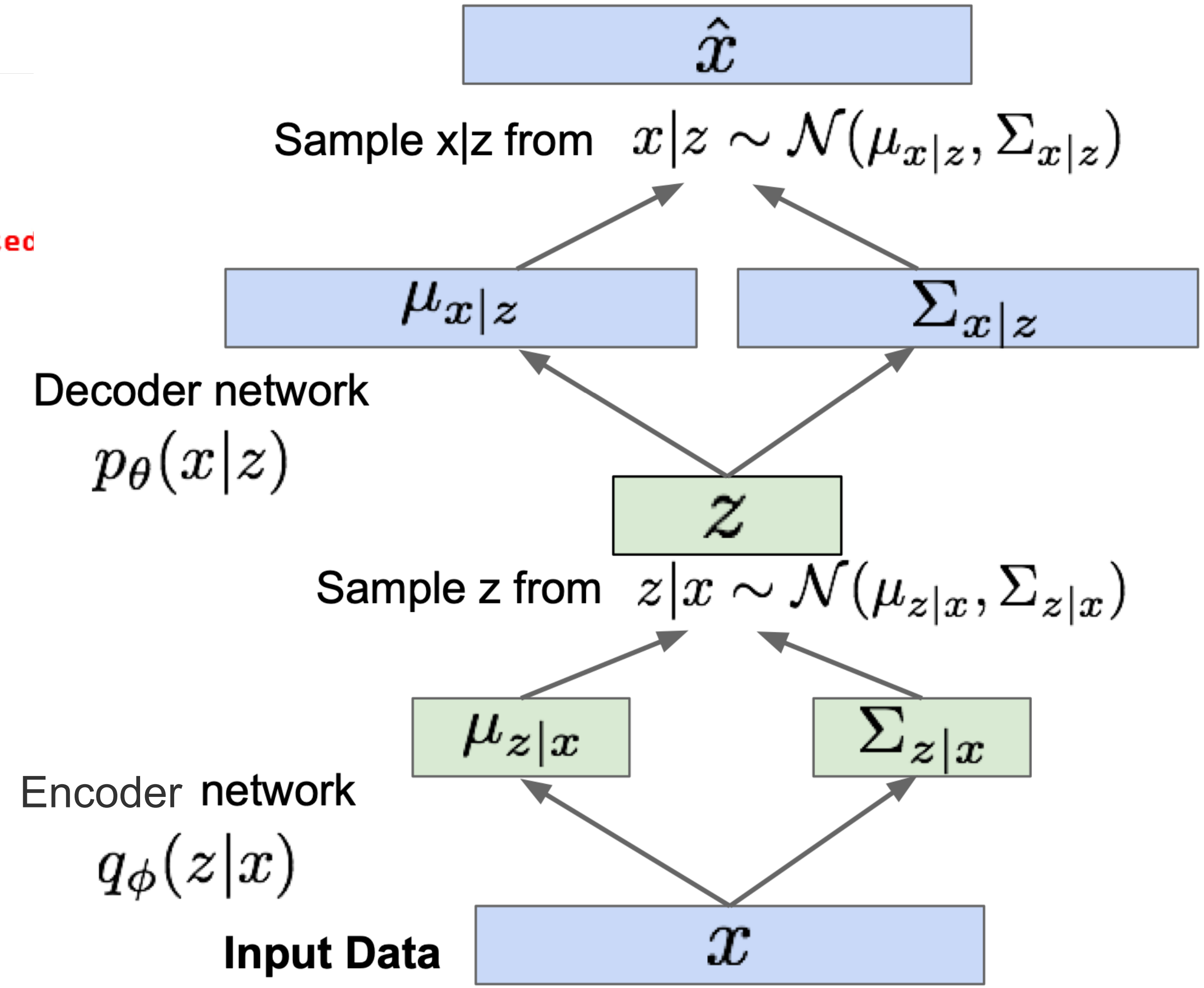




[Courtesy: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n]

Example: VAEs for images



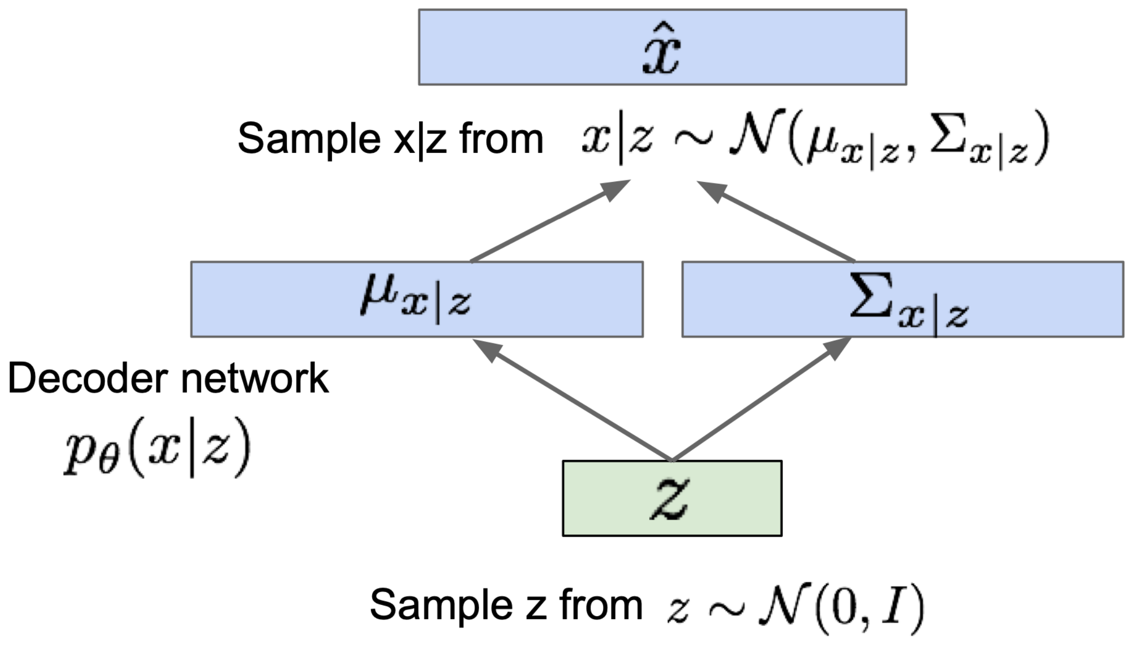


[Courtesy: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n]

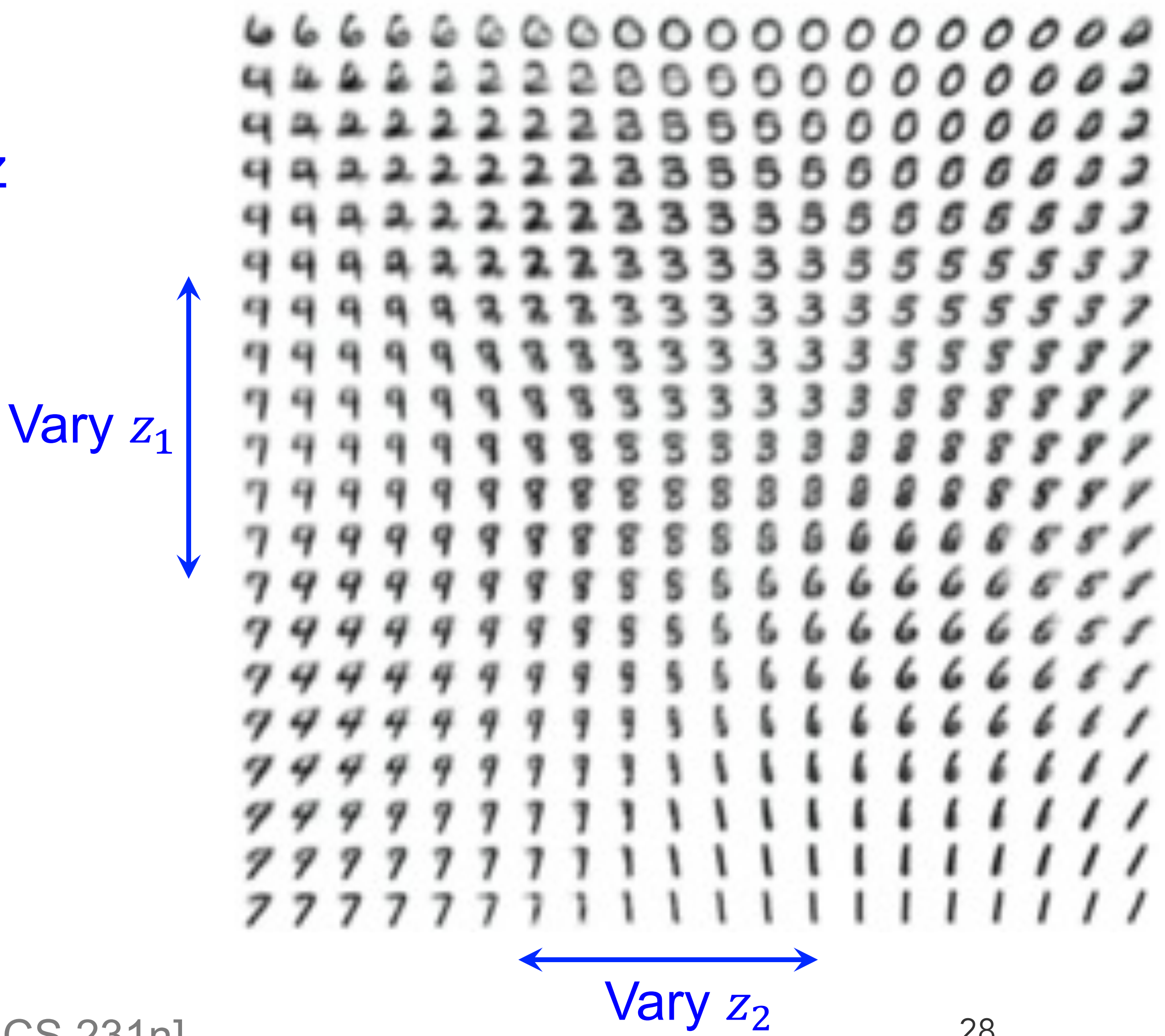
Example: VAEs for images

Generating samples:

- Use decoder network. Now sample z from prior!



Data manifold for 2-d **z**

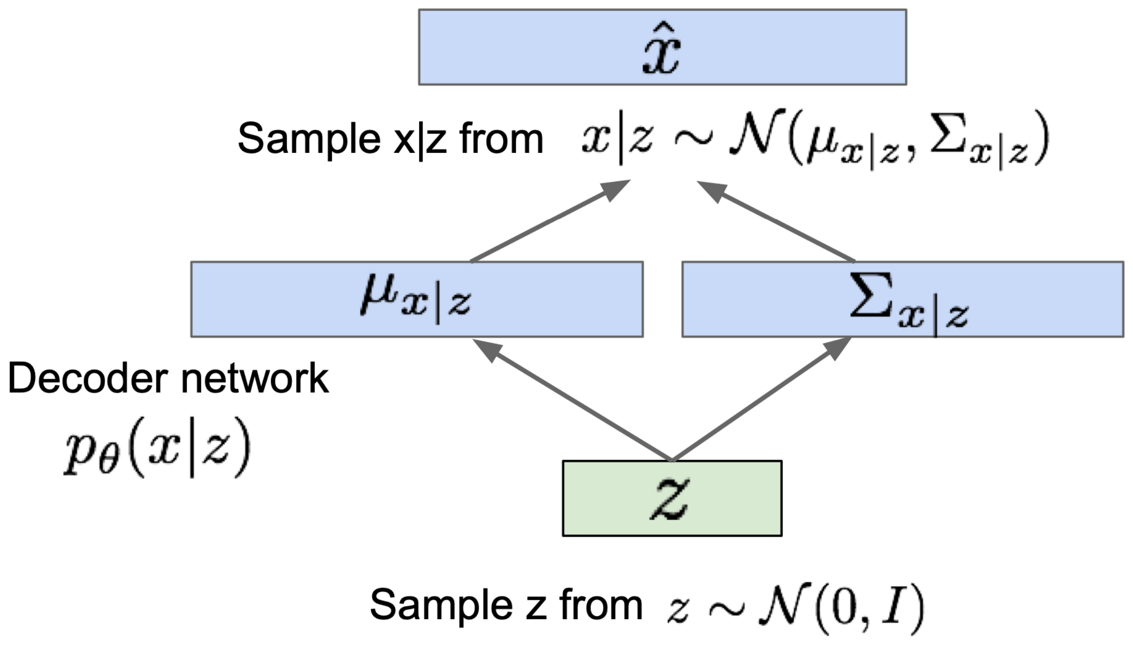


[Courtesy: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n]

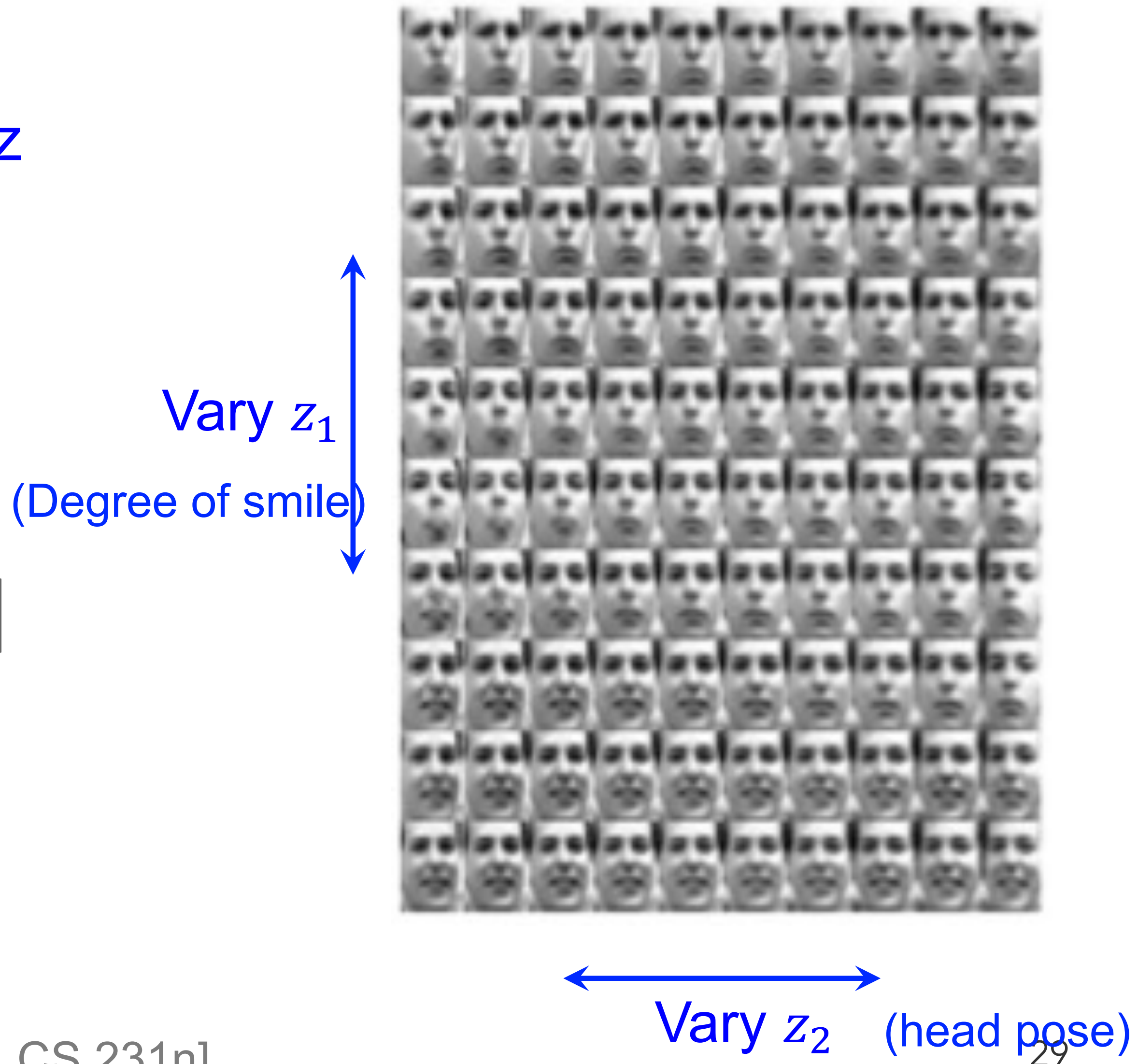
Example: VAEs for images

Generating samples:

- Use decoder network. Now sample z from prior!



Data manifold for 2-d **z**



[Courtesy: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n]

Example: VAEs for text

- Latent code interpolation and sentences generation from VAEs [Bowman et al., 2015].

“ 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 .

Variational Auto-Encoders (VAEs)

Algorithm 1 Minibatch version of the Auto-Encoding VB (AEVB) algorithm. Either of the two SGVB estimators in section 2.3 can be used. We use settings $M = 100$ and $L = 1$ in experiments.

$\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Initialize parameters
repeat
  $\mathbf{X}^M \leftarrow$ Random minibatch of M datapoints (drawn from full dataset)
  $\boldsymbol{\epsilon} \leftarrow$ Random samples from noise distribution $p(\boldsymbol{\epsilon})$
  $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \widetilde{\mathcal{L}}^M(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^M, \boldsymbol{\epsilon})$ (Gradients of minibatch estimator (8))
  $\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Update parameters using gradients $\mathbf{g}$ (e.g. SGD or Adagrad [DHS10])
until convergence of parameters $(\boldsymbol{\theta}, \boldsymbol{\phi})$
return $\boldsymbol{\theta}, \boldsymbol{\phi}$

[Kingma & Welling, 2014]

Note: Amortized Variational Inference

- Variational distribution as an inference model $q_{\phi}(\boldsymbol{z}|\boldsymbol{x})$ with parameters $\boldsymbol{\phi}$ (which was traditionally factored over samples)
- Amortize the cost of inference by learning a **single** data-dependent inference model
- The trained inference model can be used for quick inference on new data

Variational Auto-encoders: Summary

- A combination of the following ideas:
  - Variational Inference: ELBO
  - Variational distribution parametrized as neural networks
  - Reparameterization trick

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) = [\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] \; - \mathrm{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \,||\, p(\boldsymbol{z}))$$

Reconstruction

Divergence from prior

(Razavi et al., 2019)

- Pros:
  - Principled approach to generative models
  - Allows inference of $q(z|x)$, can be useful feature representation for other tasks
- Cons:
  - Samples blurrier and lower quality compared to GANs
  - Tend to collapse on text data

Key Takeaways

- Stochastic VI
- Computing Gradients of Expectations $\mathcal{L} = \mathbb{E}_{q_{\lambda}(\mathbf{z})}[f_{\lambda}(\mathbf{z})]$
  - Score gradient

$$\nabla_{\lambda}\mathcal{L} = \mathbb{E}_{q_{\lambda}(\mathbf{z})}[f_{\lambda}(\mathbf{z})\nabla_{\lambda}\log q_{\theta}(\mathbf{z}) + \nabla_{\lambda}f_{\lambda}(\mathbf{z})]$$

  - Reparameterization gradient

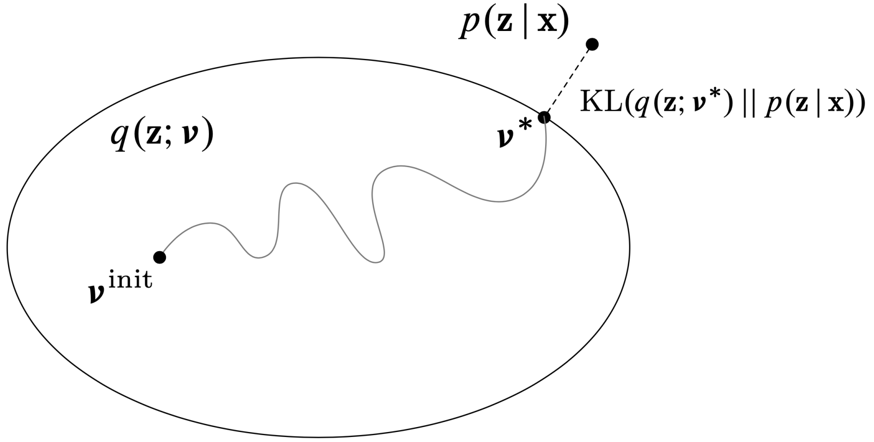
$$\nabla_{\lambda}\mathcal{L} = \mathbb{E}_{\boldsymbol{\epsilon}\sim\boldsymbol{s}(\boldsymbol{\epsilon})}[\nabla_{\mathbf{z}}f_{\lambda}(\mathbf{z})\ \nabla_{\lambda}t(\epsilon, \lambda)]$$

- Black-box VI
- Variational autoencoders (VAEs)

Summary so far: Supervised Learning, Unsupervised Learning

- Supervised Learning
  - Maximum likelihood estimation (MLE)
  - Duality between MLE and Maximum Entropy Principle

- Unsupervised learning
  - Maximum likelihood estimation (MLE) with latent variables
    - Marginal log-likelihood
  - EM algorithm for MLE
    - ELBO
  - Variational Inference
    - ELBO
    - Variational distributions



Questions?