ADVANCED ARTIFICIAL INTELLIGENCE: COMPUTER VISION

Generative models: latent spaces and variational autoencoders

Schedule for today

- Lecture: VAEs—variational autoencoders for image generation
- Lab: add batch normalization to CNN for CIFAR10
- Lab: finetune ResNet18 with (augmented) MNIST data
- Lab: generate MNIST-like images
- Lab: generate CIFAR-like images

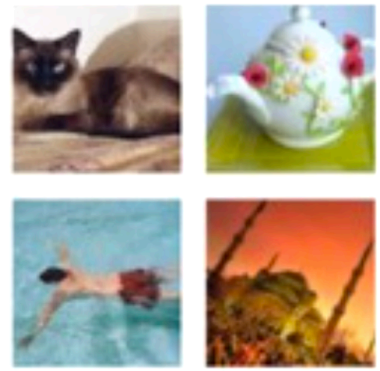
Generative modeling

- take as input training samples from some distribution and learn a model that represents that distribution
- how to learn a distribution similar to that of the training data?

Sample Generation

Input samples

Training data ~ $P_{data}(x)$

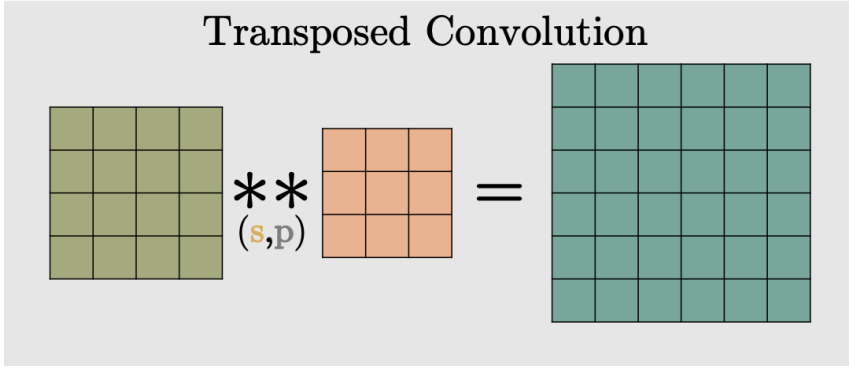


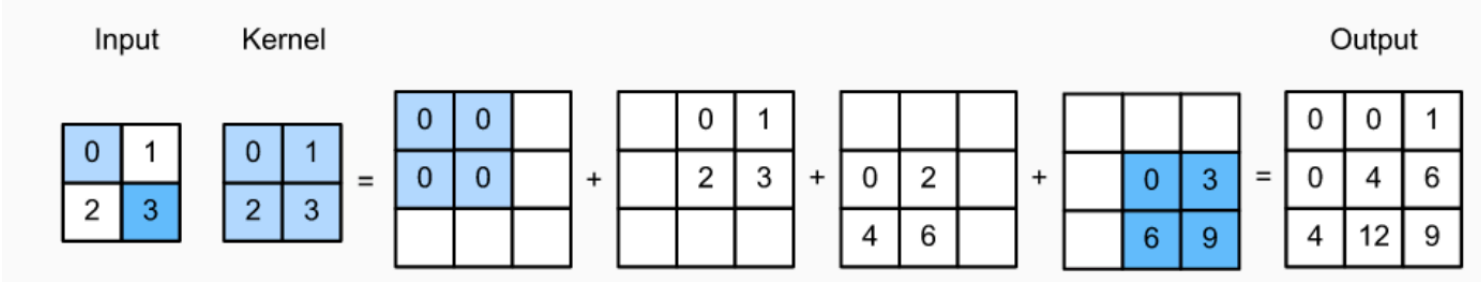
Generated samples

Generated ~ $P_{model}(x)$

Transposed convolution

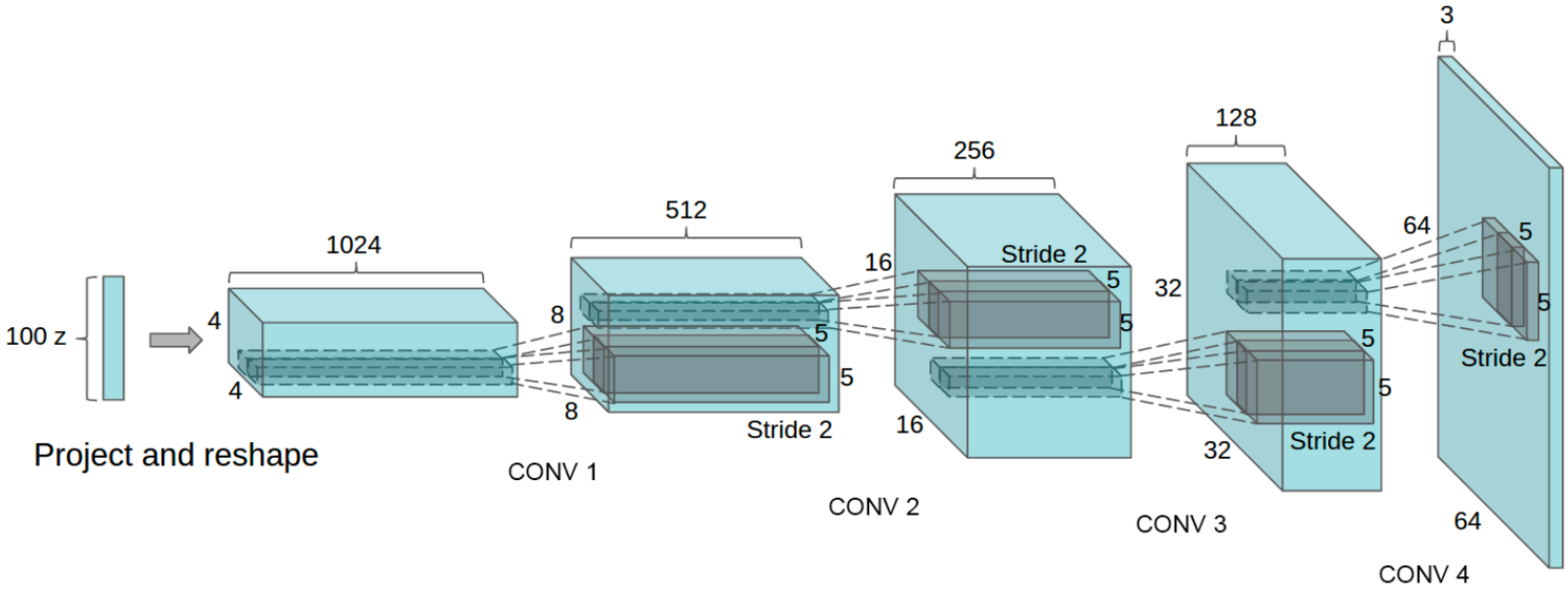
- operation that expands the spatial dimensions



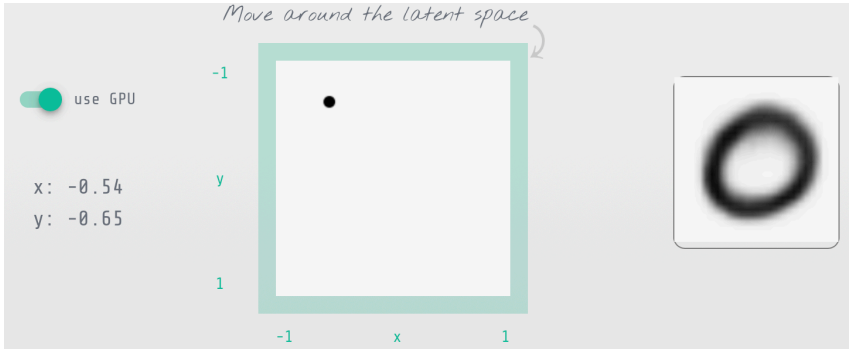


Transposed convolution

- used for trainable image generation

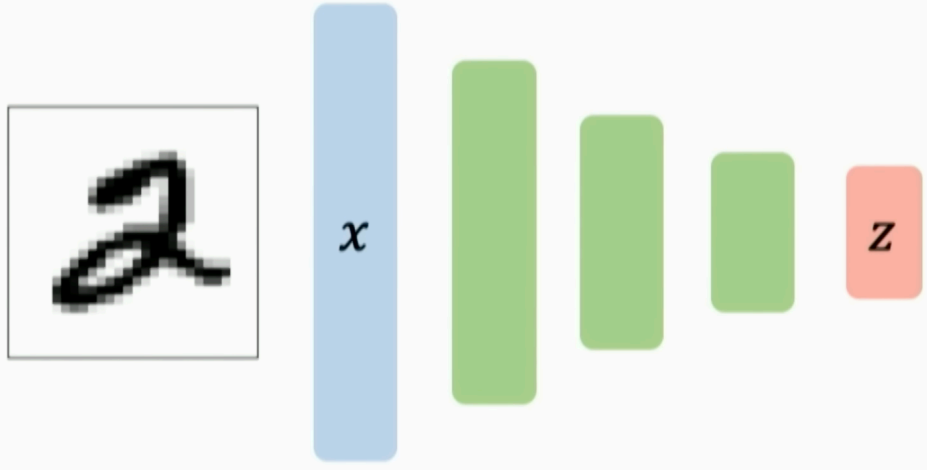


Demo



Autoencoders

Unsupervised approach for learning a **lower-dimensional** feature representation from unlabeled training data

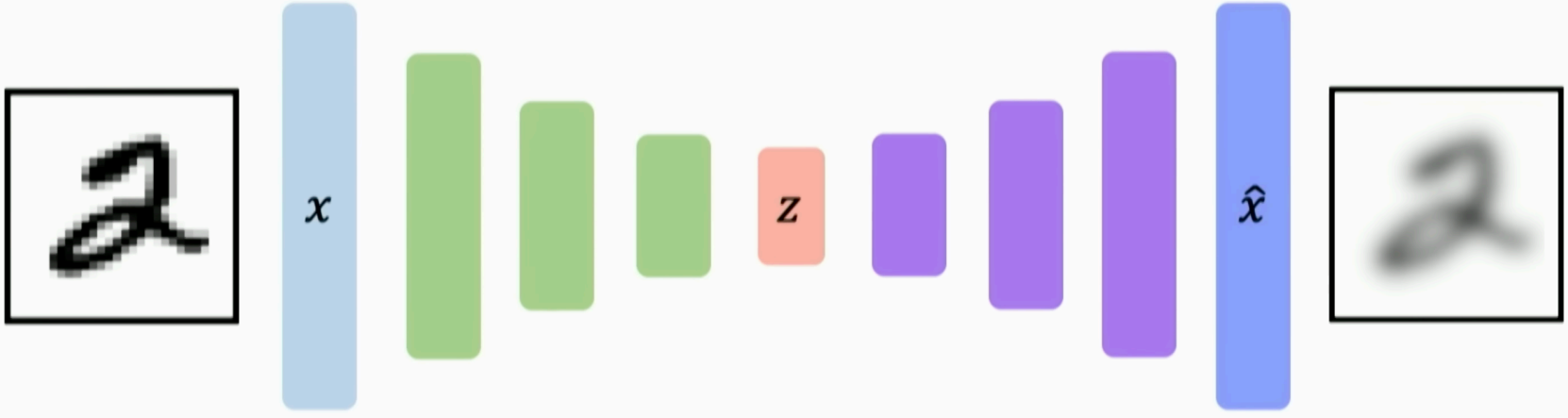


Why do we care about a low-dimensional z?

"Encoder" learns mapping from the data, x, to a low-dimensional latent space, z

Autoencoders

How can we learn this latent space?
Train the model to use these features to **reconstruct the original data**

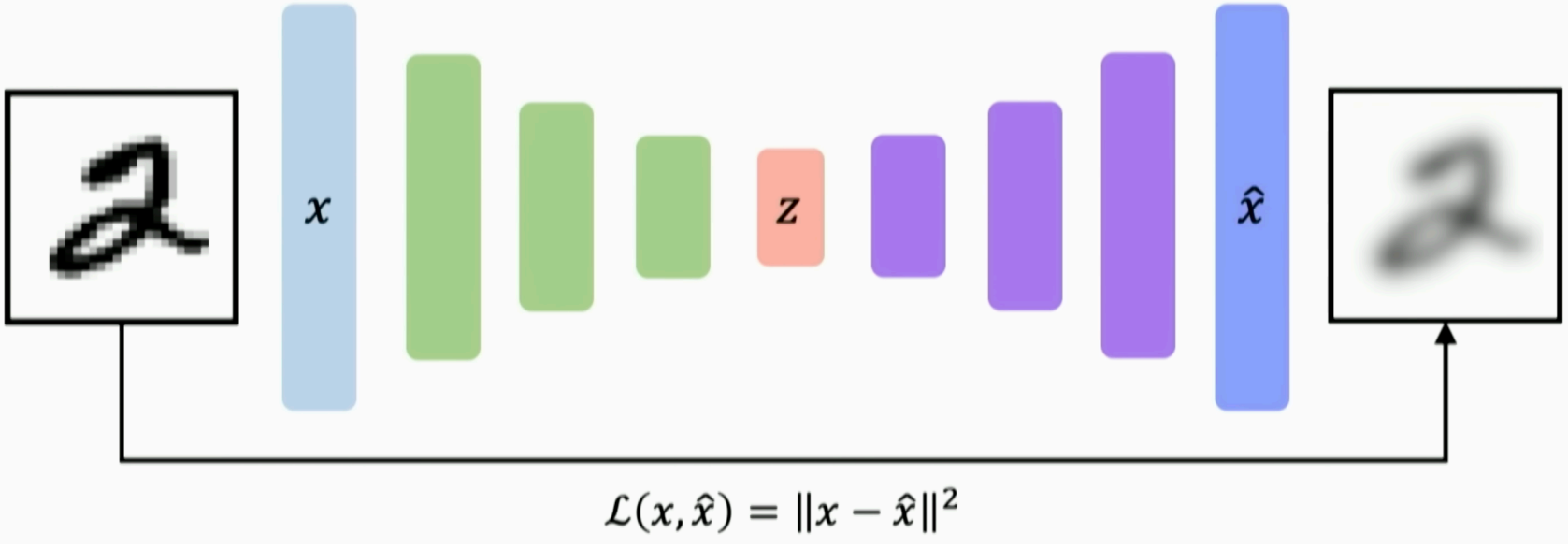


"Decoder" learns mapping back from latent space, z,
to a reconstructed observation, $\hat{x}$

Autoencoders

How can we learn this latent space?

Train the model to use these features to **reconstruct the original data**



$$\mathcal{L}(x, \hat{x}) = \|x - \hat{x}\|^2$$

Dimensionality of latent space −> reconstruction quality

Autoencoding is a form of compression!
Smaller latent space will force a larger training bottleneck

2D latent space

5D latent space

Ground Truth

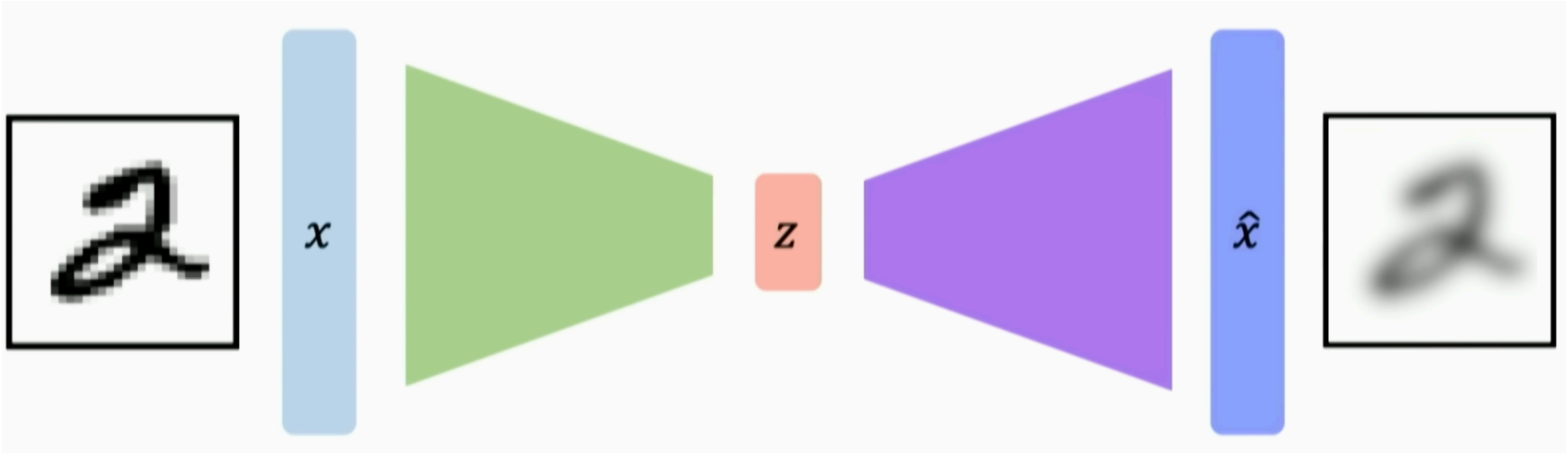
Autoencoders for representation learning

Bottleneck hidden layer forces network to learn a compressed latent representation

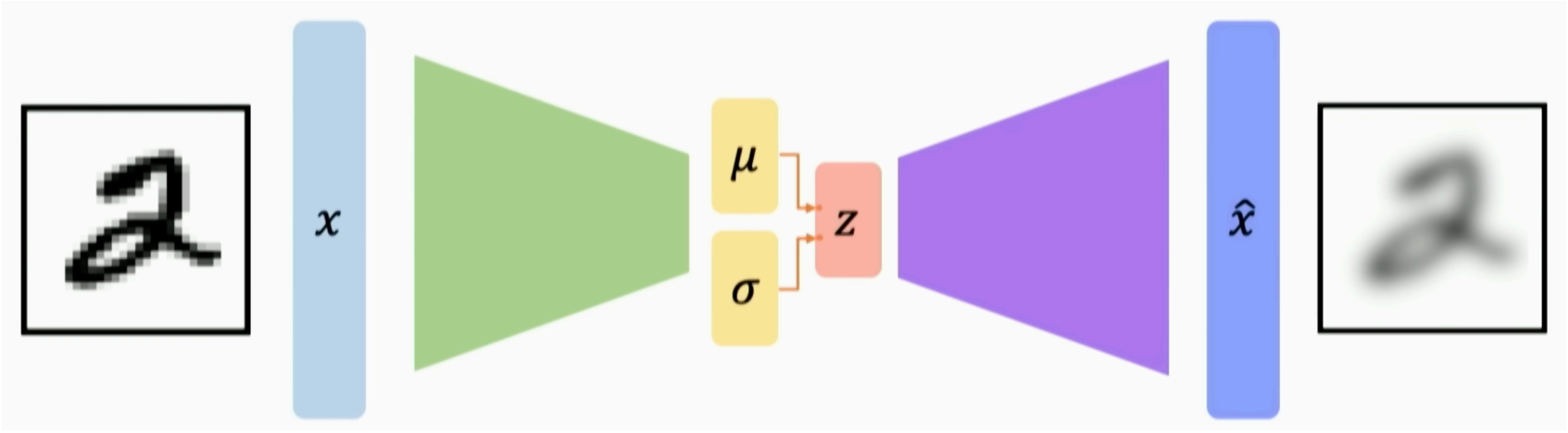
Reconstruction loss forces the latent representation to capture (or encode) as much "information" about the data as possible

Autoencoding = **Auto**matically **encoding** data; "Auto" = **self**-encoding

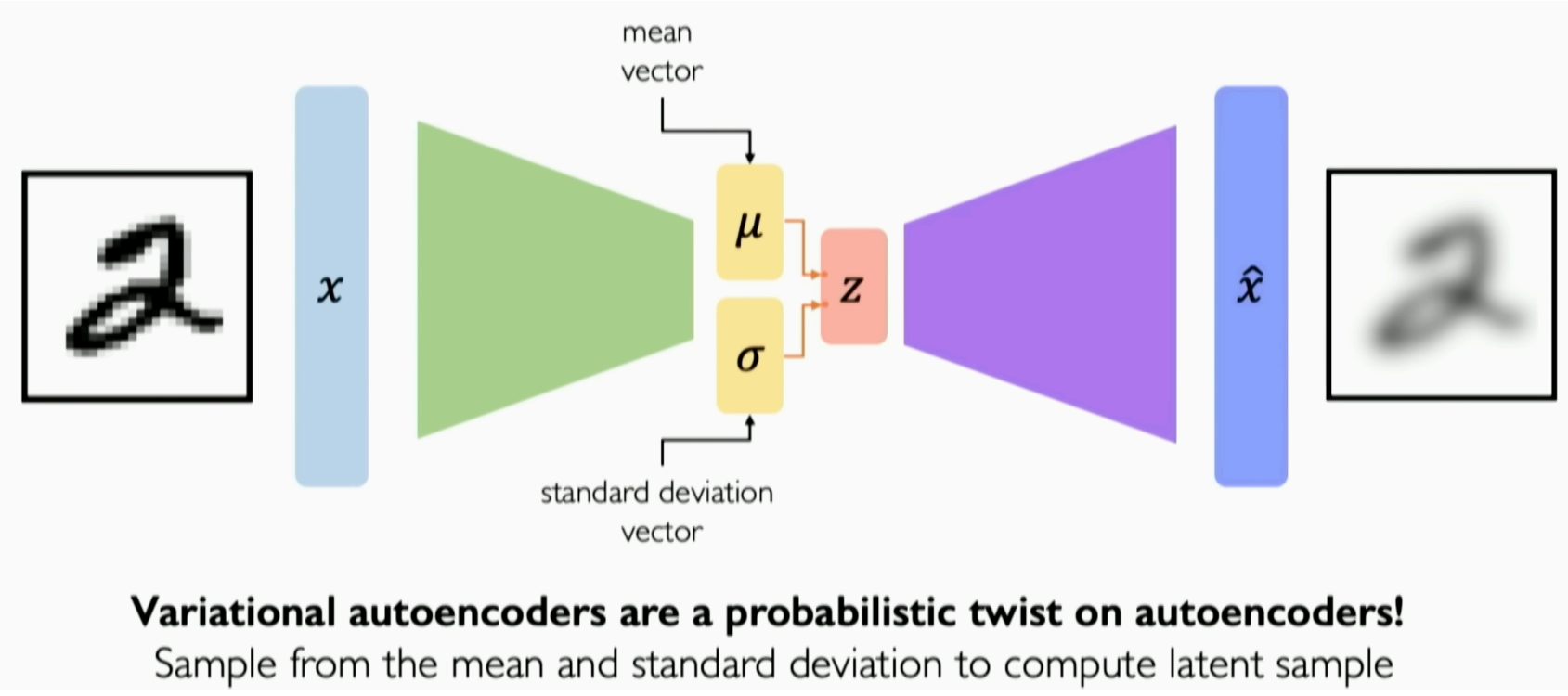
Traditional autoencoders



VAEs key difference from traditional autoencoders



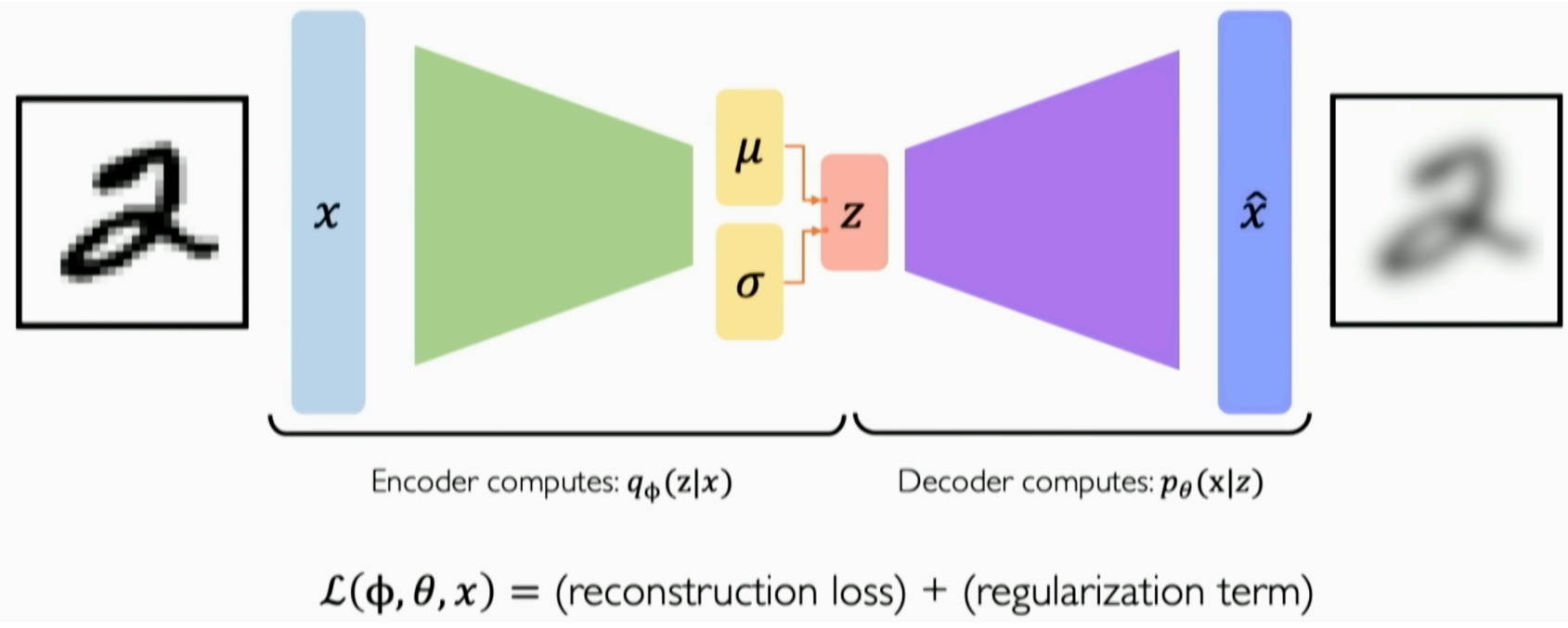
VAEs key difference from traditional autoencoders



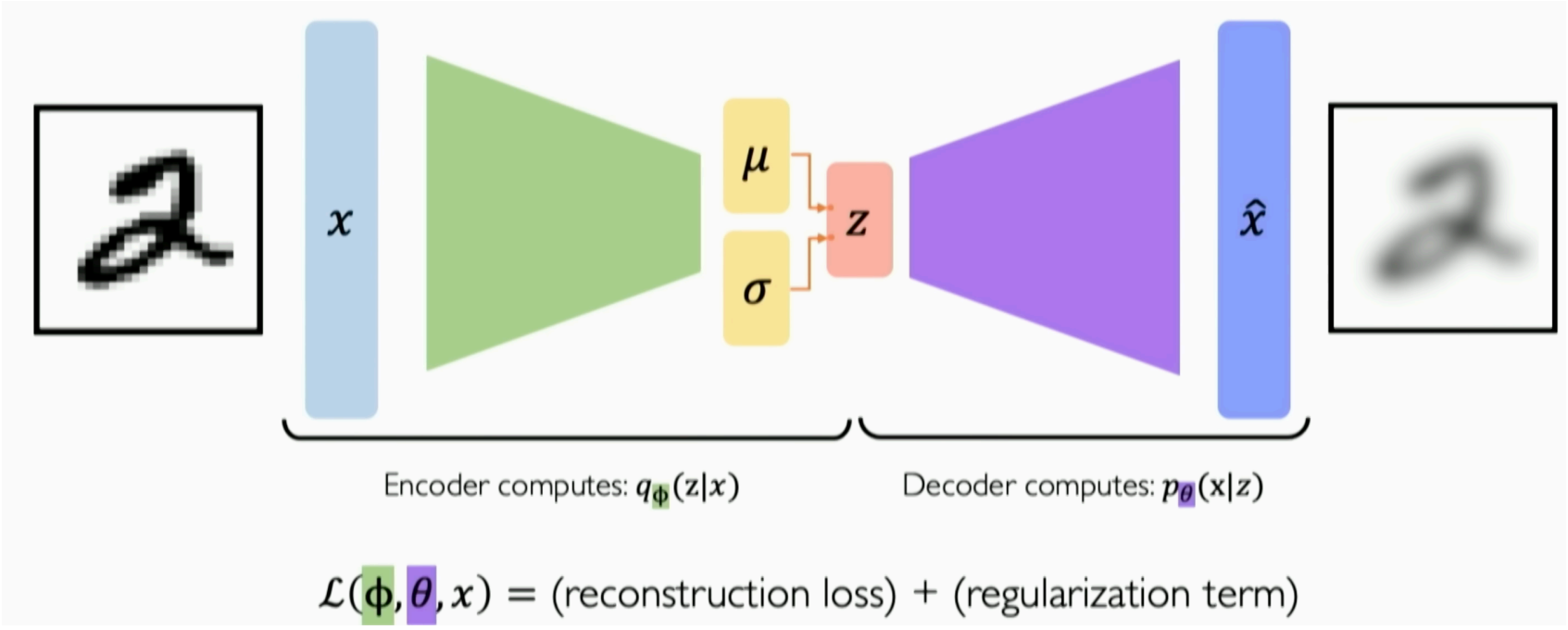
Variational autoencoders are a probabilistic twist on autoencoders!

Sample from the mean and standard deviation to compute latent sample

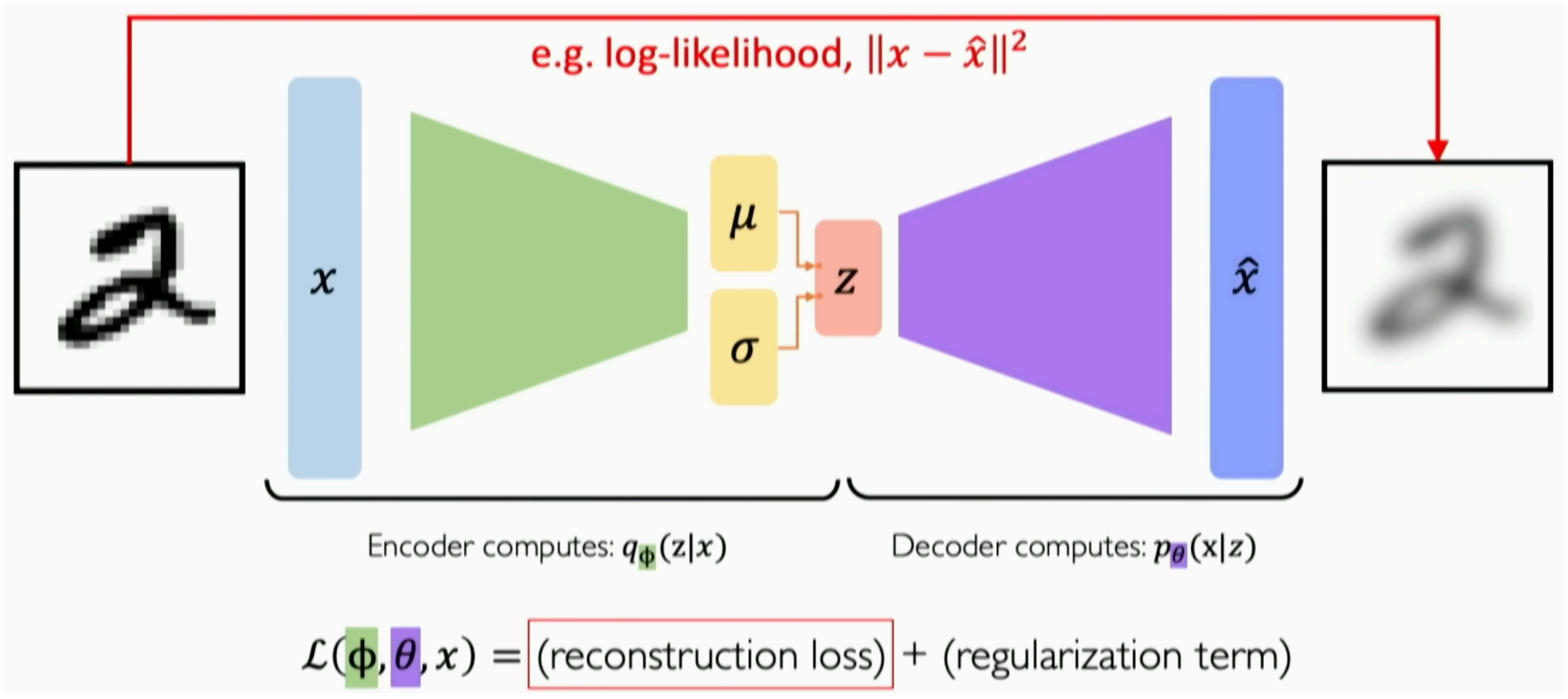
VAE optimization



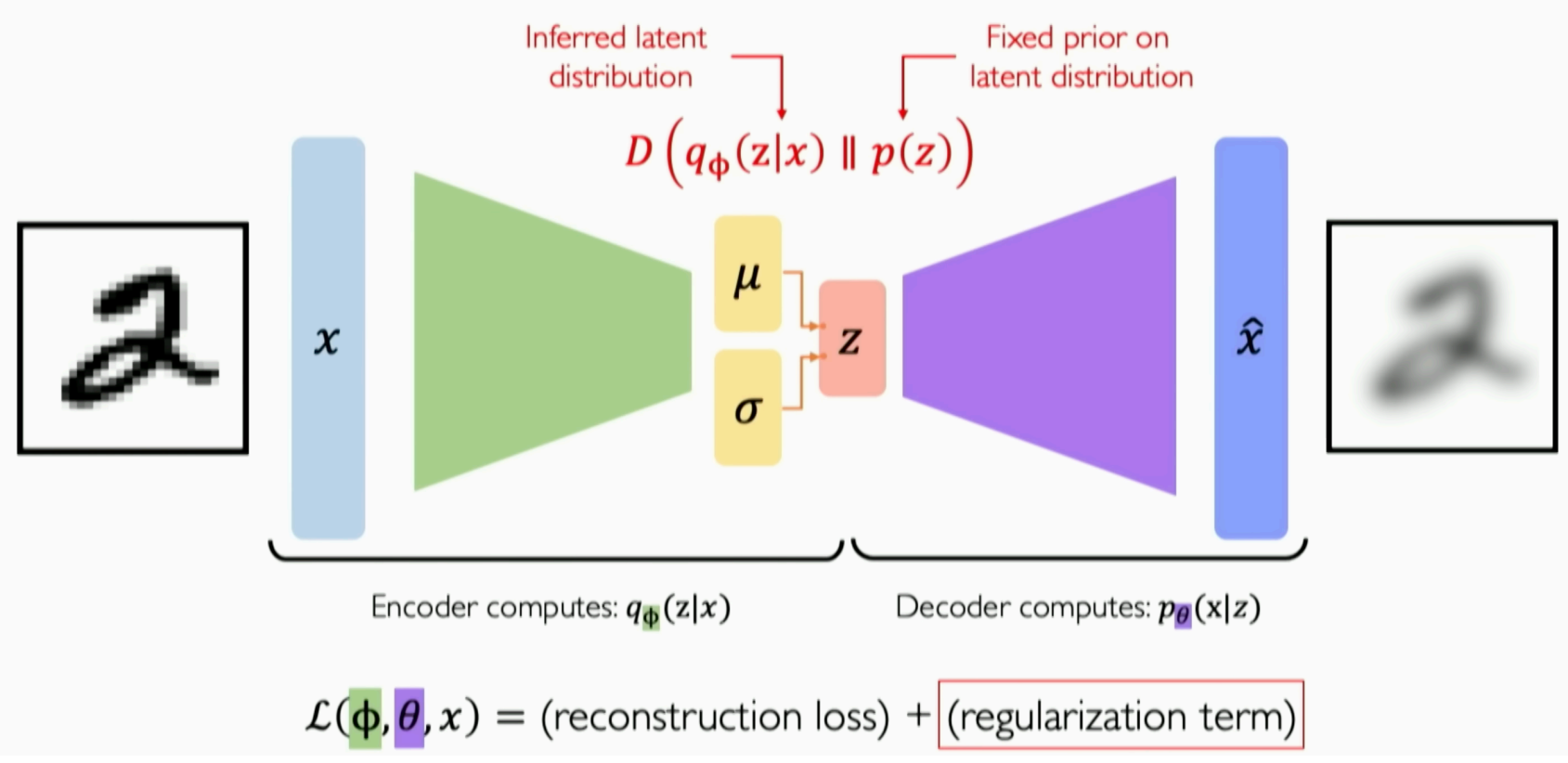
VAE optimization



VAE optimization



VAE optimization

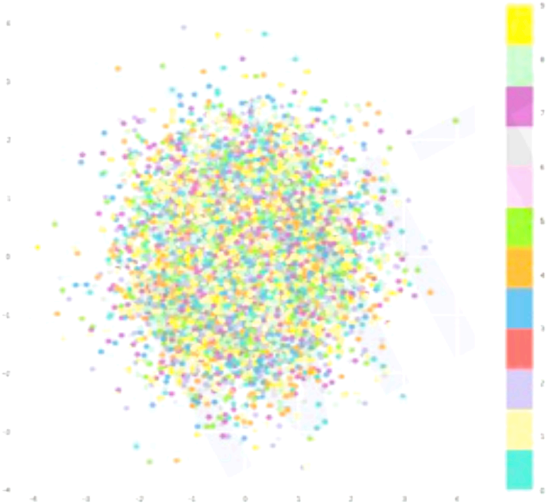


Priors on the latent distribution

$$D\left(q_\phi(\mathrm{z}|x) \parallel p(z)\right)$$

Inferred latent distribution

Fixed prior on latent distribution



Common choice of prior – Normal Gaussian:

$$p(z) = \mathcal{N}(\mu = 0, \sigma^2 = 1)$$

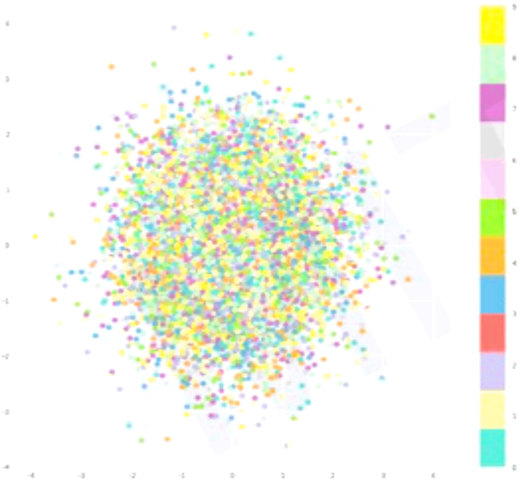
- Encourages encodings to distribute encodings evenly around the center of the latent space
- Penalize the network when it tries to "cheat" by clustering points in specific regions (i.e., by memorizing the data)

Priors on the latent distribution

$$D\left(q_\phi(\mathrm{z}|x) \parallel p(z)\right)$$

$$= -\frac{1}{2}\sum_{j=0}^{k-1}\left(\sigma_j + \mu_j^2 - 1 - \log\sigma_j\right)$$

KL-divergence between the two distributions



Common choice of prior – Normal Gaussian:

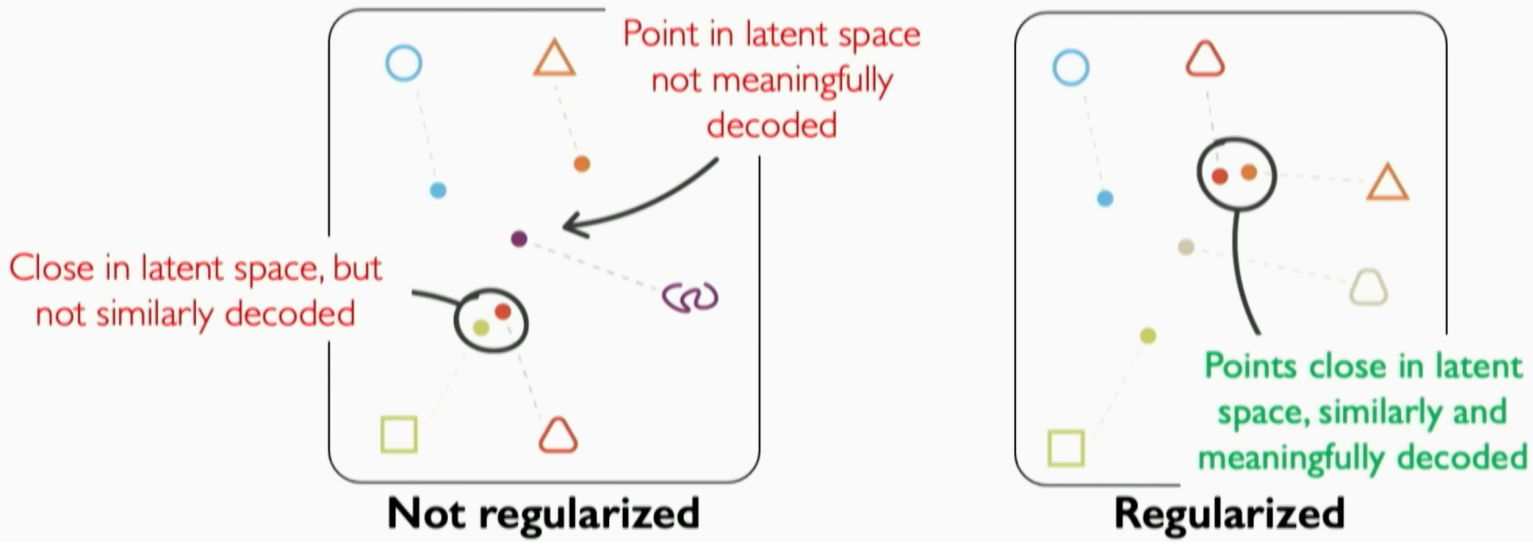
$$p(z) = \mathcal{N}(\mu = 0, \sigma^2 = 1)$$

- Encourages encodings to distribute encodings evenly around the center of the latent space
- Penalize the network when it tries to "cheat" by clustering points in specific regions (i.e., by memorizing the data)

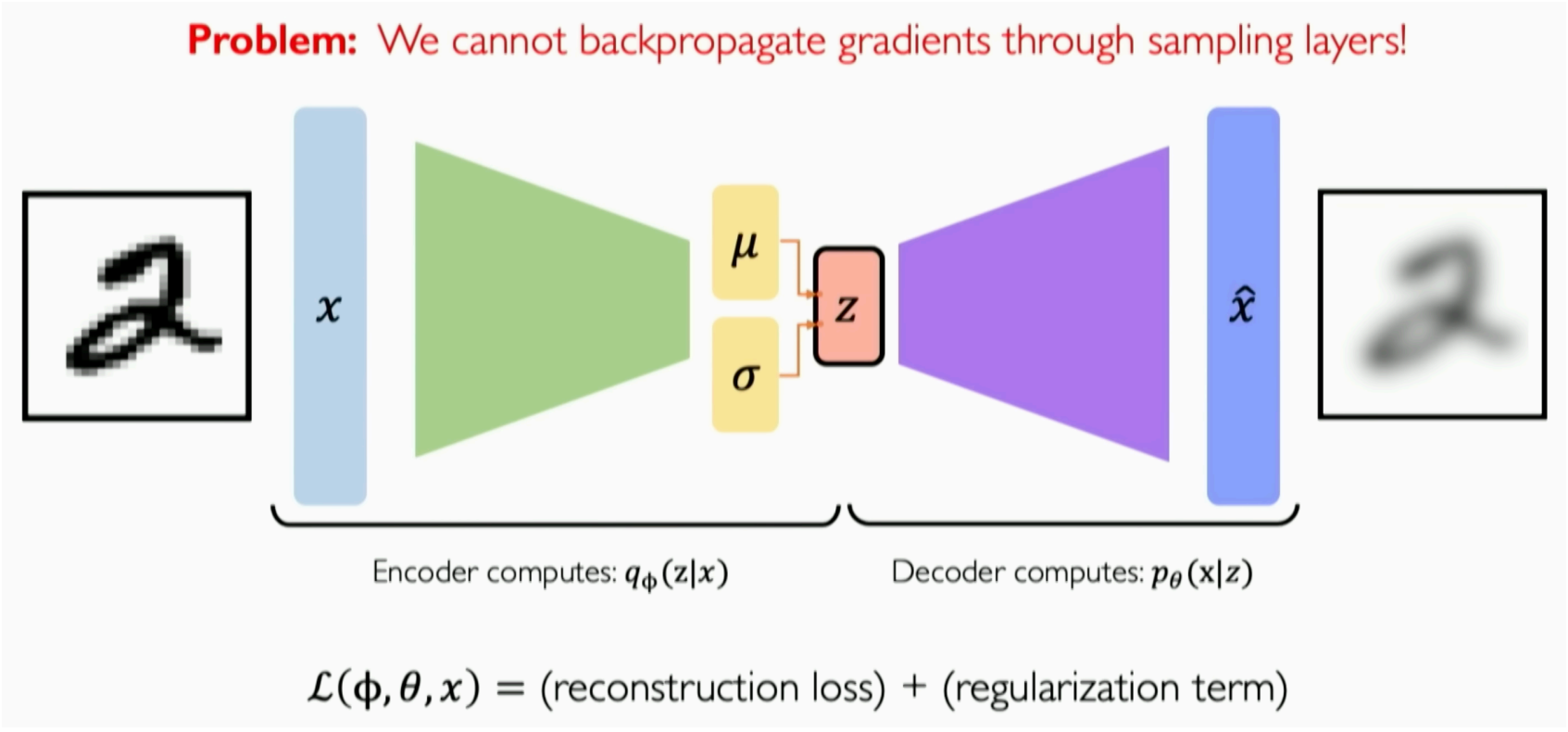
Intuition on regularization

What properties do we want to achieve from regularization? 🤔

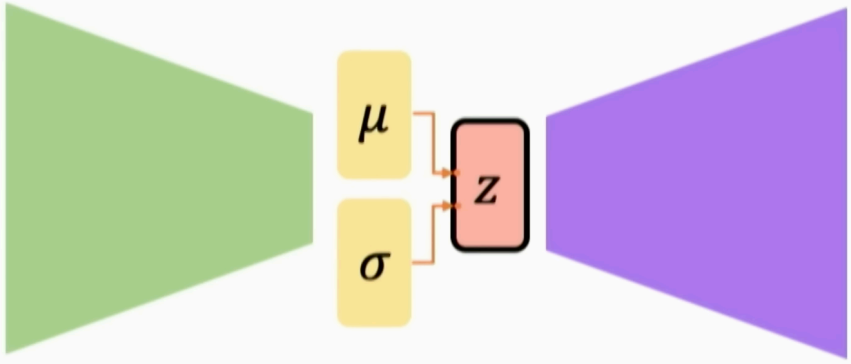
1. **Continuity:** points that are close in latent space → similar content after decoding
2. **Completeness:** sampling from latent space → "meaningful" content after decoding



VAE computation graph



Reparameterizing the sampling layer



Key Idea:

~~$z \sim \mathcal{N}(\mu, \sigma^2)$~~

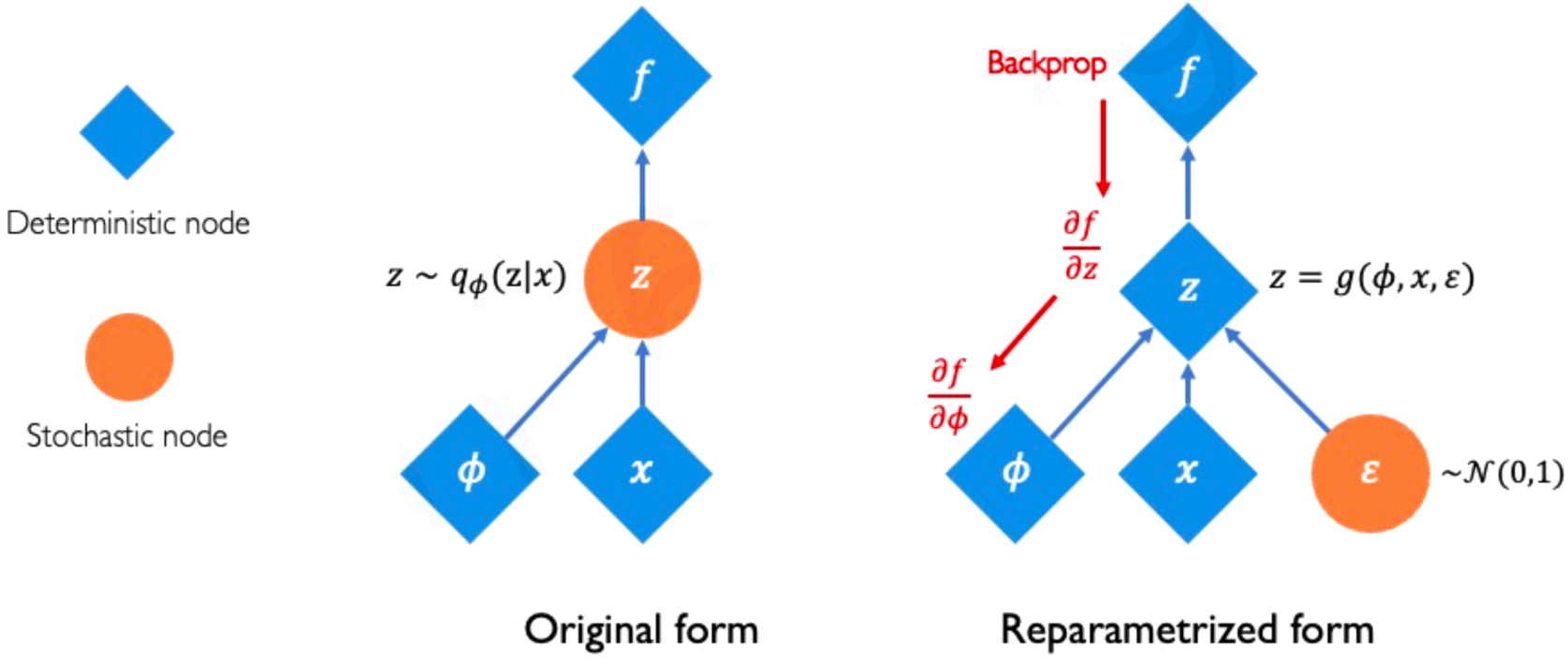
Consider the sampled latent vector z as a sum of

- a fixed μ vector,
- and fixed σ vector, scaled by random constants drawn from the prior distribution

$$\Rightarrow z = \mu + \sigma \odot \varepsilon$$

where $\varepsilon \sim \mathcal{N}(0,1)$

Reparameterizing the sampling layer



Original form

Reparametrized form

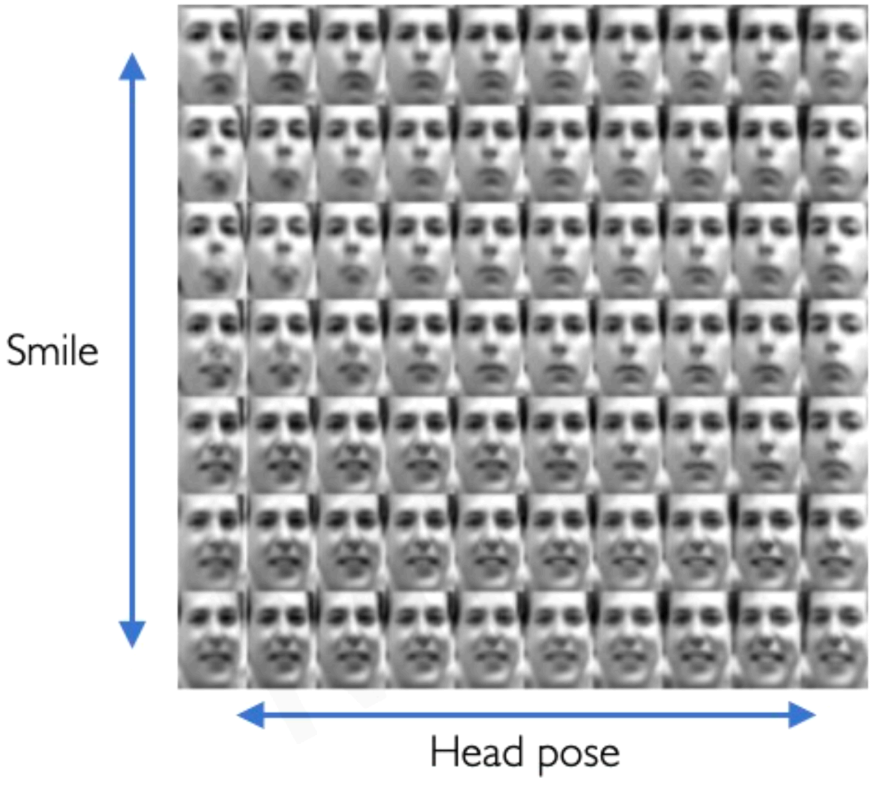
VAEs: latent perturbation

Slowly increase or decrease a **single latent variable**
Keep all other variables fixed

Head pose

Different dimensions of z encodes **different interpretable latent features**

VAEs: latent perturbation

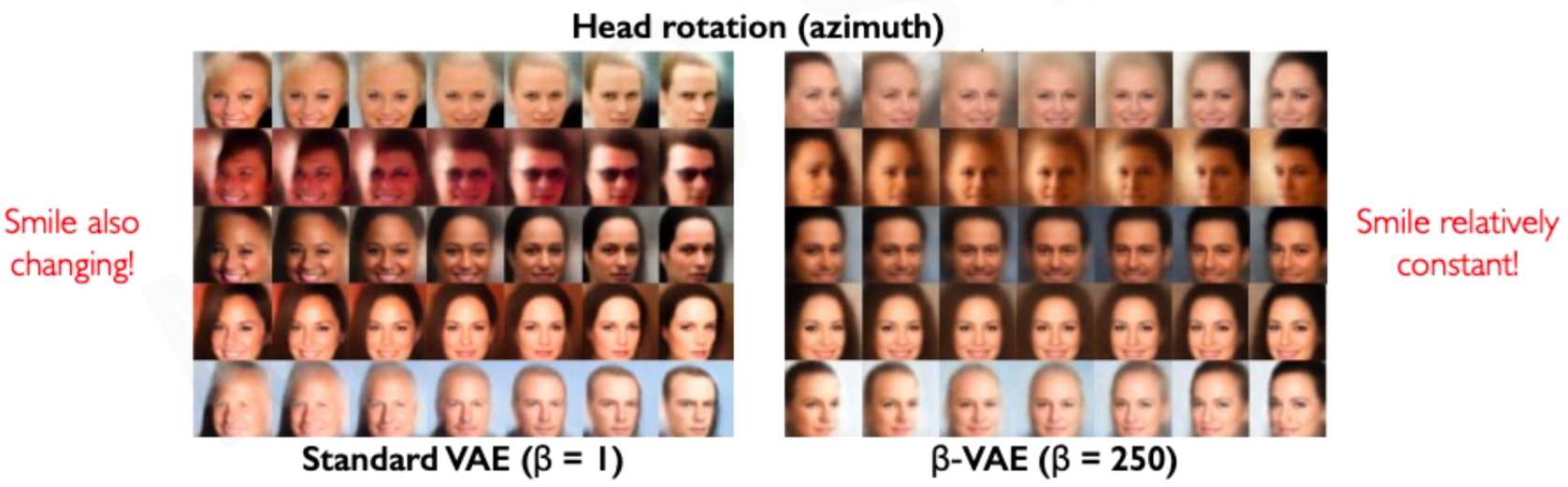


Ideally, we want latent variables that are uncorrelated with each other

Enforce diagonal prior on the latent variables to encourage independence

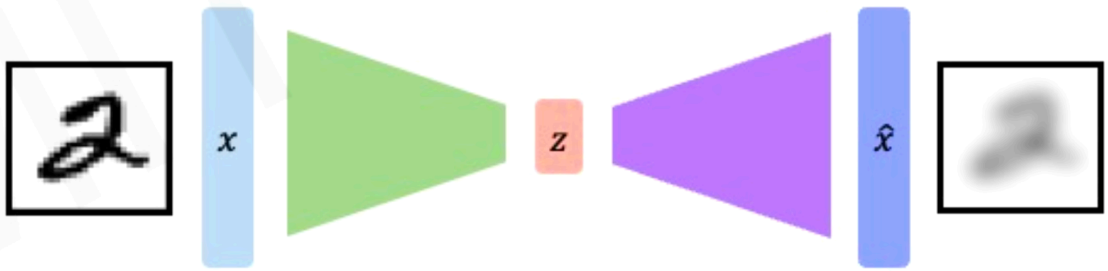
Disentanglement

Increasing regularization



VAE summary

1. Compress representation of world to something we can use to learn
2. Reconstruction allows for unsupervised learning (no labels!)
3. Reparameterization trick to train end-to-end
4. Interpret hidden latent variables using perturbation
5. Generating new examples



Latent variable models

- VAEs belong to a more general class of *latent variable models*