Défi 3 : Generative Artificial Intelligence (II/III)
Variational Autoencoders (VAEs)

Défis en Intelligence Artificielle

Souhaib Ben Taieb & Victor Dheur

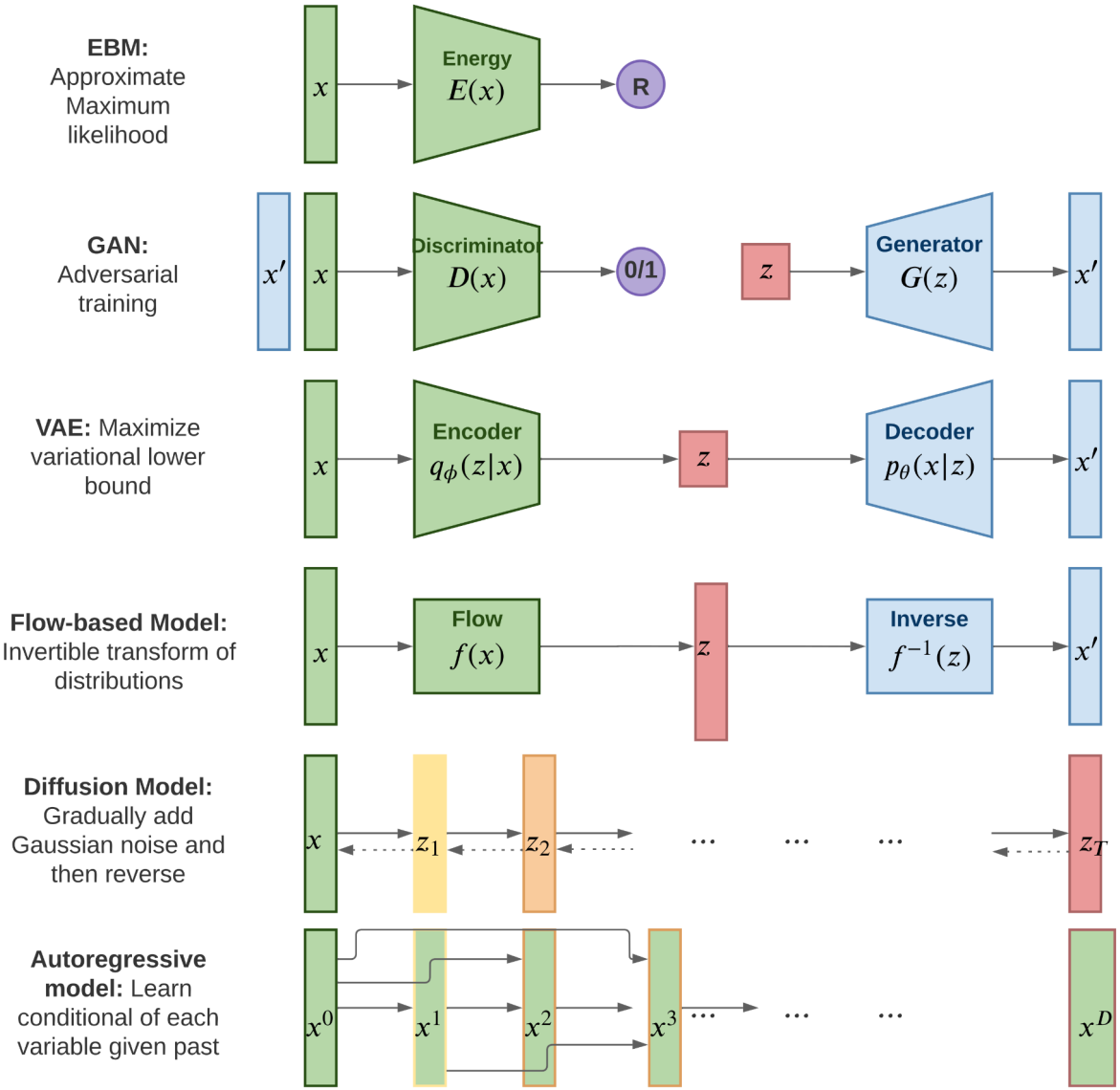
14 Décembre 2023

Université de Mons

Schedule

- *Week 1: 7 December 2023, 6-9 pm*
  - Introduction to generative modelling
- **Week 2: 14 December 2023, 6-9 pm**
  - **Variational Autoencoders (VAEs)**
- *Week 3: 21 December 2023, 6-9 pm*
  - (Subject to change) Diffusion models
- *Project report: 2 February 2024, 11:55 pm*
  - More details TBA soon

Overview of deep generative models

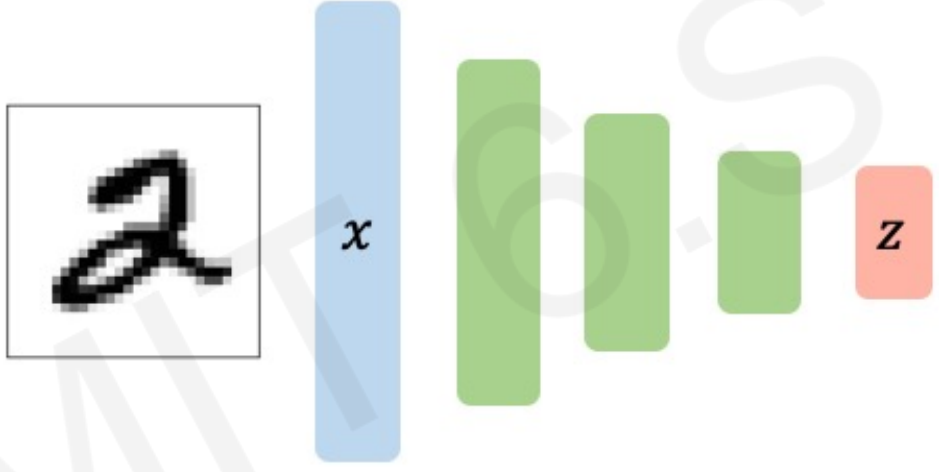


Source: [Mur23]

Autoencoders

Autoencoders: background

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

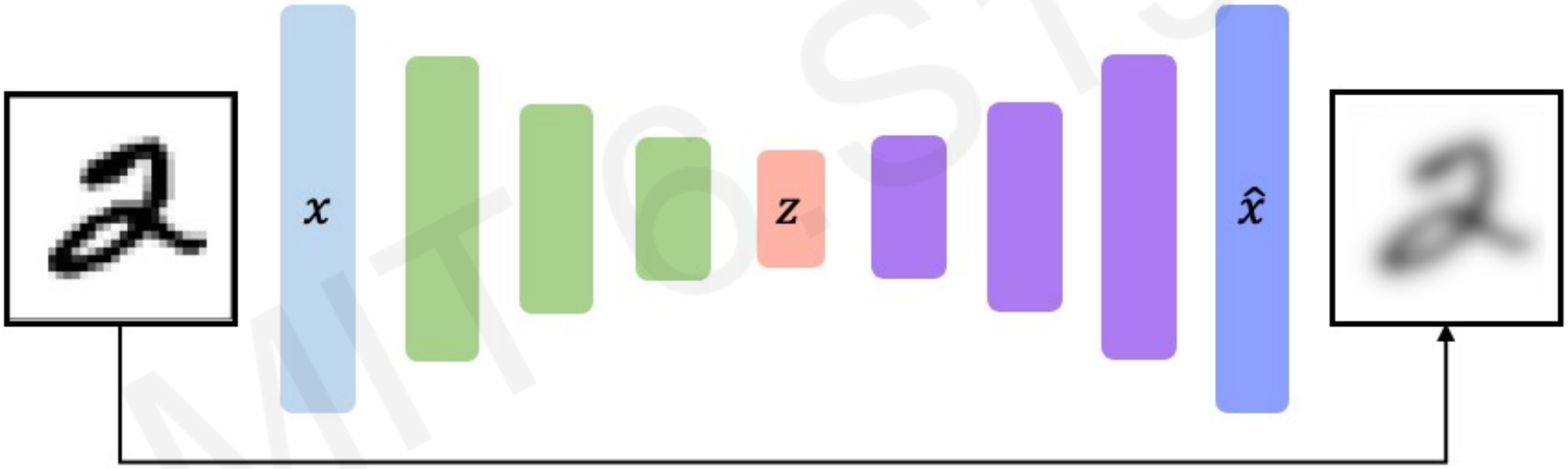
Autoencoders: background

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: background

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

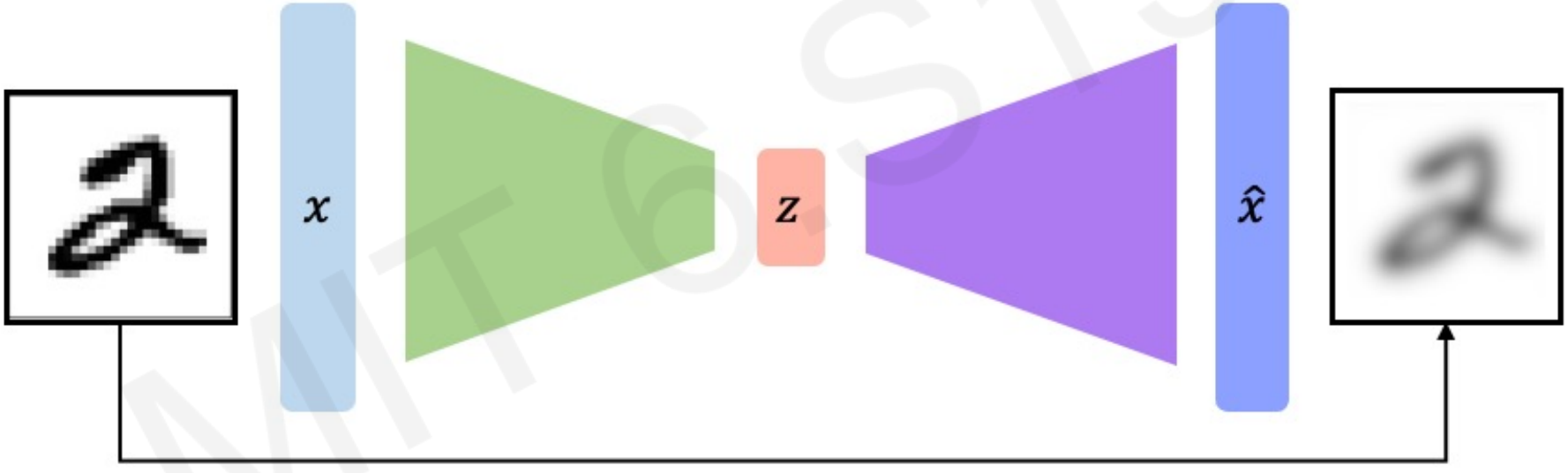


$$\mathcal{L}(x, \hat{x}) = ||x - \hat{x}||^2$$

Loss function doesn't use any labels!!

Autoencoders: background

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



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

Loss function doesn't use any labels!!

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

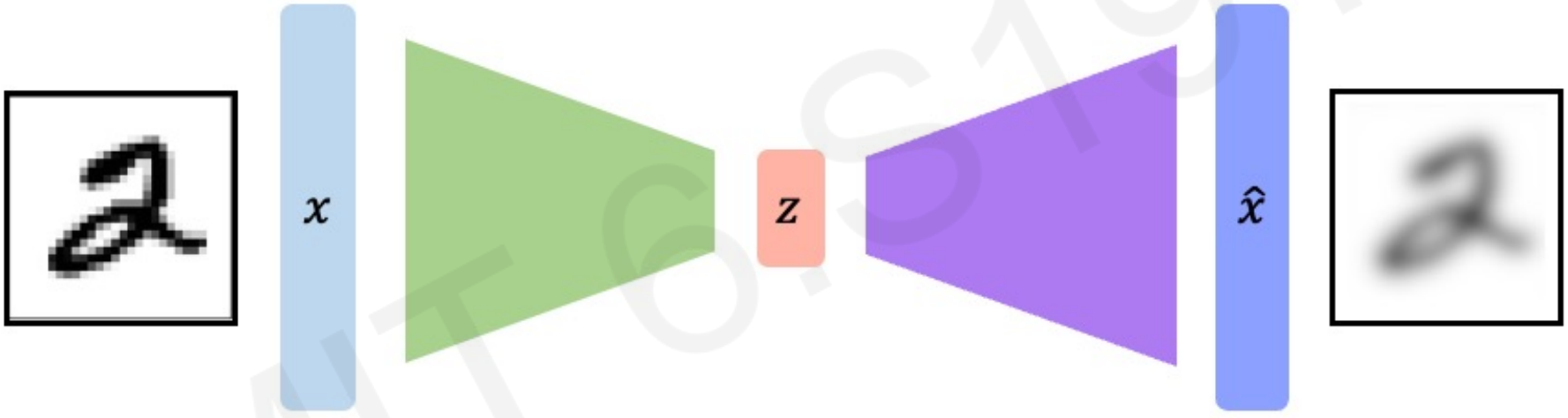
Bottleneck hidden layer forces network to learn a compressed latent representation

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

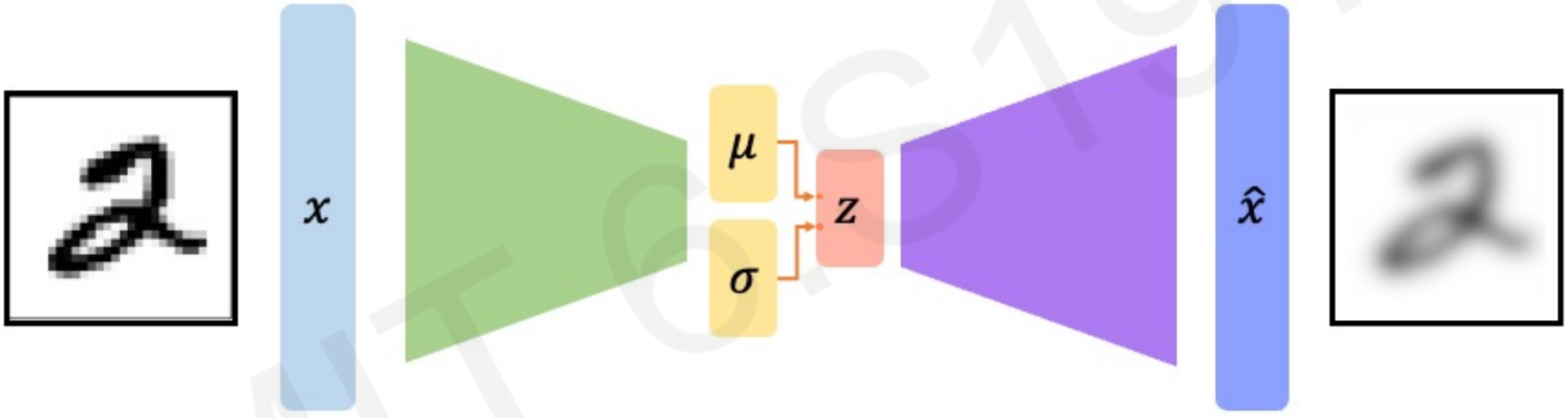
Autoencoding = **Auto**matically **encoding** data; "Auto" = **self**-encoding

Variational Autoencoders (VAEs)

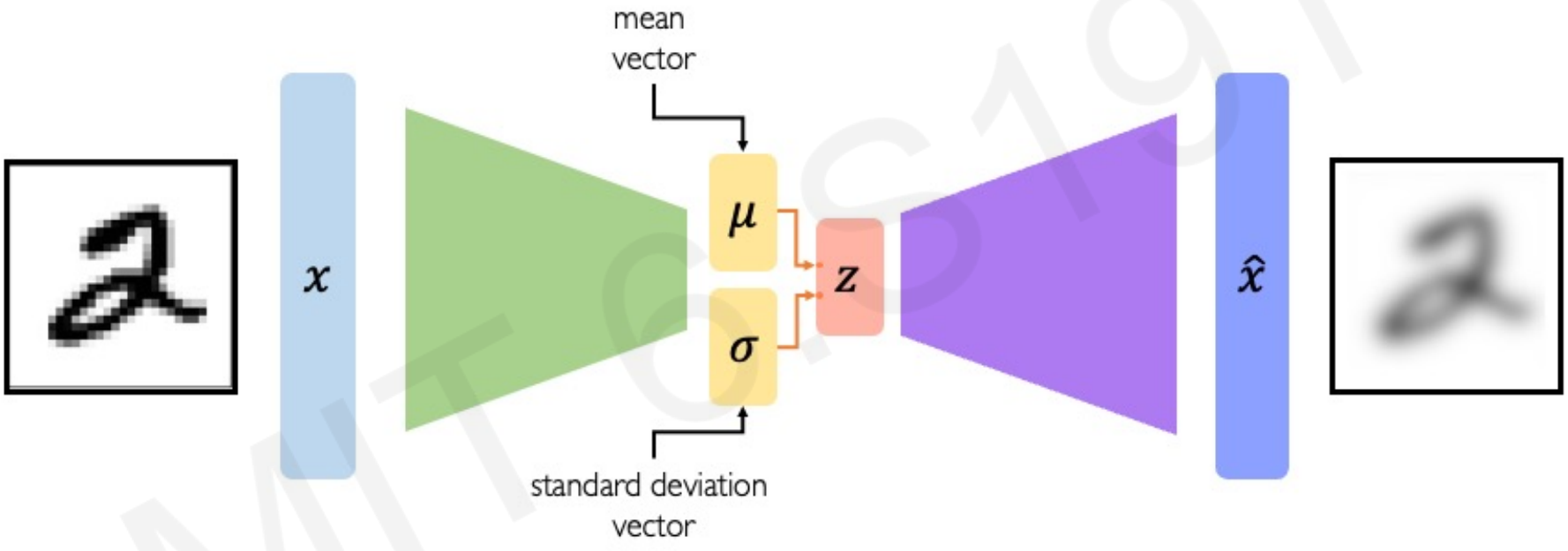
Traditional autoencoders



VAEs: key difference with traditional autoencoder



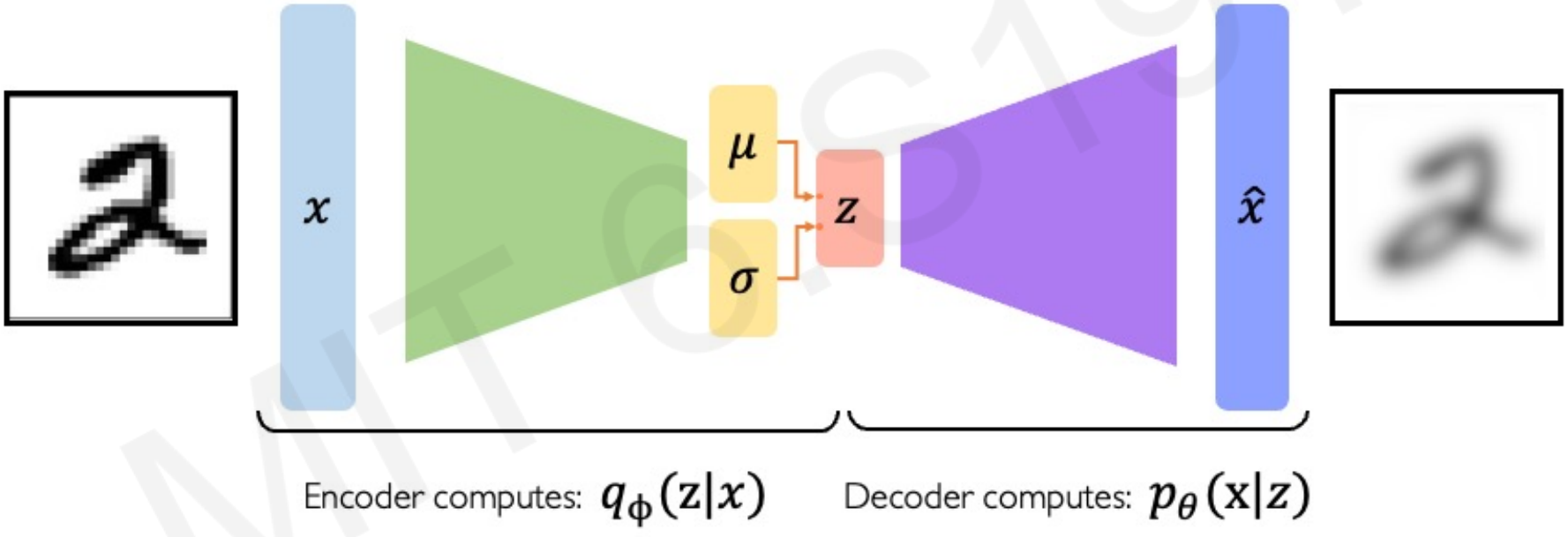
VAEs: key difference with traditional autoencoder



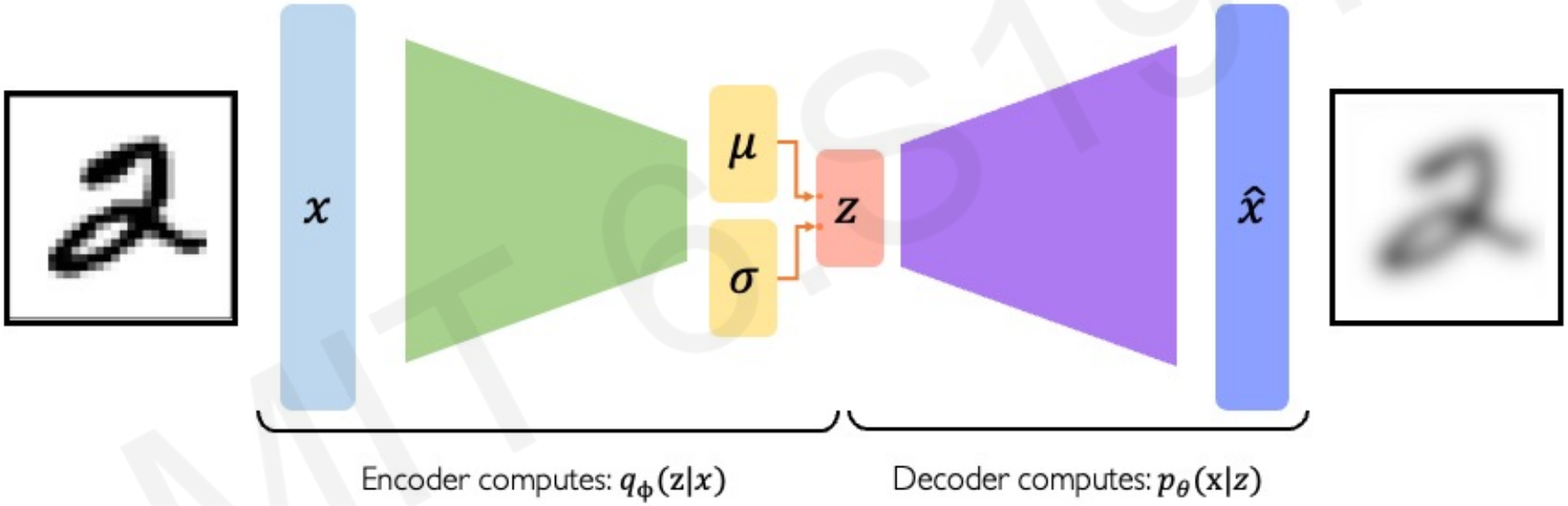
Variational autoencoders are a probabilistic twist on autoencoders!

Sample from the mean and standard deviation to compute latent sample

VAE optimization

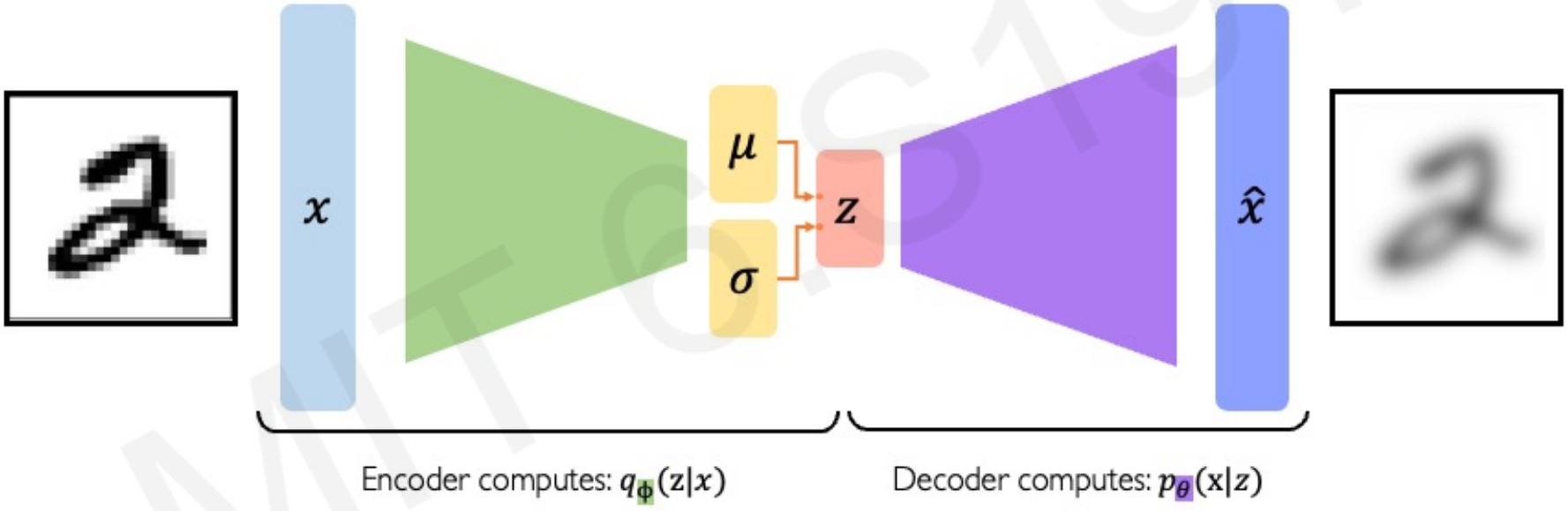


VAE optimization



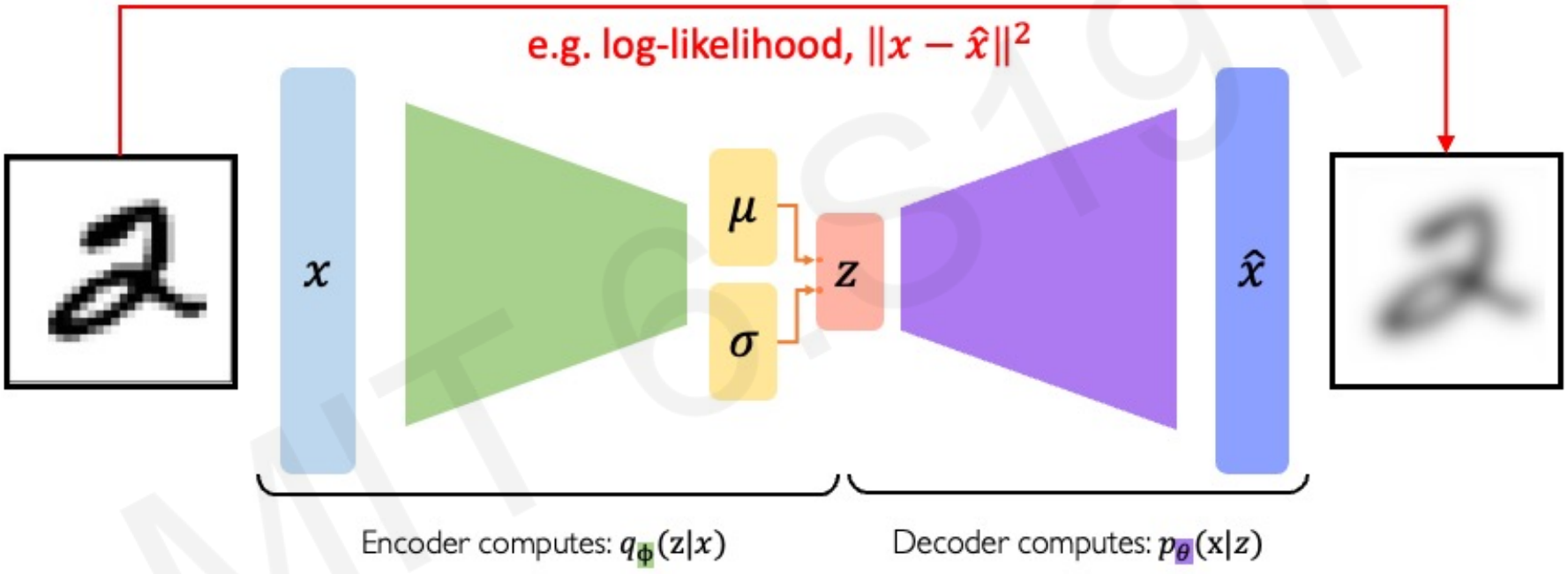
$$\mathcal{L}(\phi, \theta, x) = \text{(reconstruction loss)} + \text{(regularization term)}$$

VAE optimization



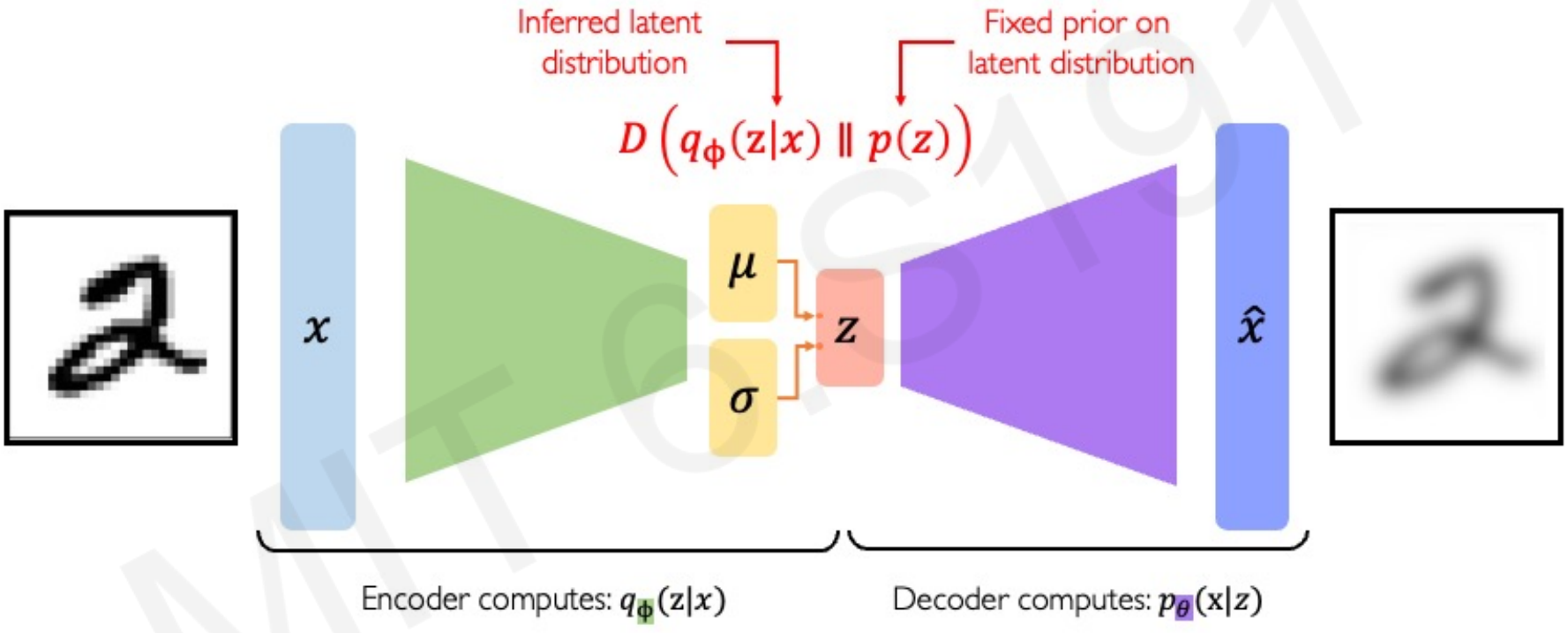
$$\mathcal{L}(\phi, \theta, x) = \text{(reconstruction loss)} + \text{(regularization term)}$$

VAE optimization



$$\mathcal{L}(\phi, \theta, x) = \boxed{\text{(reconstruction loss)}} + \text{(regularization term)}$$

VAE optimization



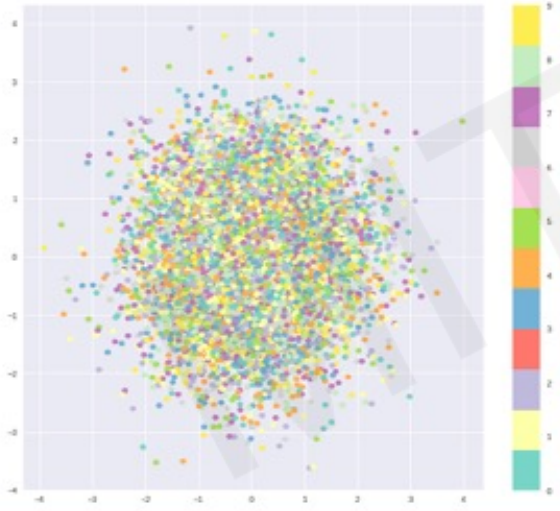
$$\mathcal{L}(\phi, \theta, x) = \text{(reconstruction loss)} + \boxed{\text{(regularization term)}}$$

Priors on the latent distribution

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

Inferred latent distribution

Fixed prior on latent distribution



Common choice of prior – Normal Gaussian:

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

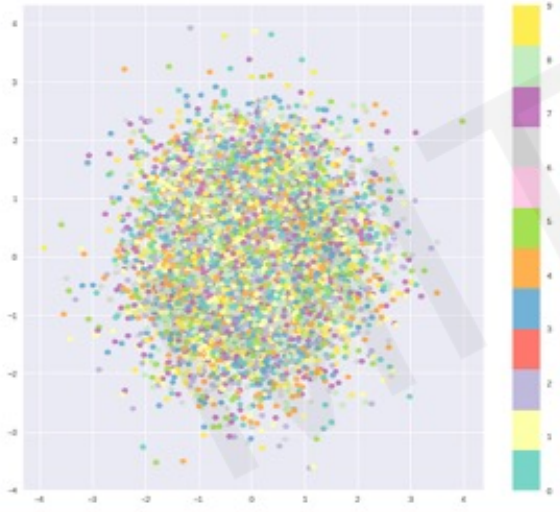
- Encourages encodings to distribute encodings evenly around the center of the latent space
- Penalize the network when it tries to "cheat" by clustering points in specific regions (i.e., by memorizing the data)

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:

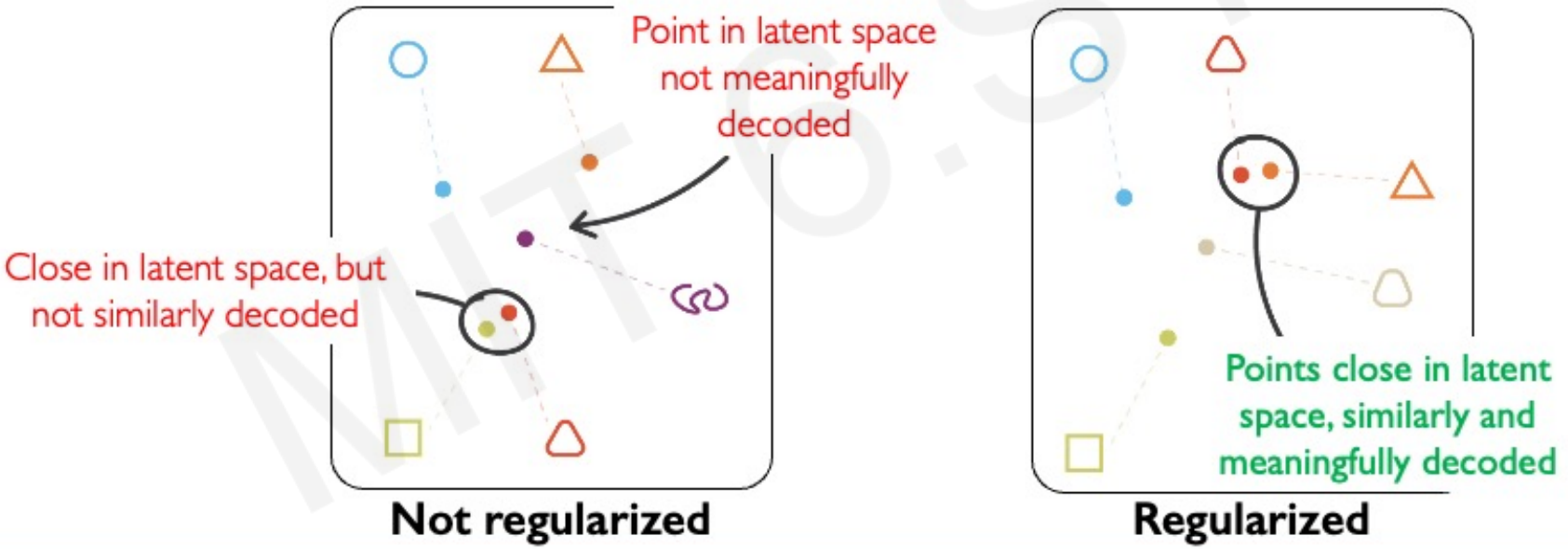
$$p(z) = \mathcal{N}(\mu = 0, \sigma^2 = 1)$$

- 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 and the Normal prior

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



Intuition on regularization and the Normal prior

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

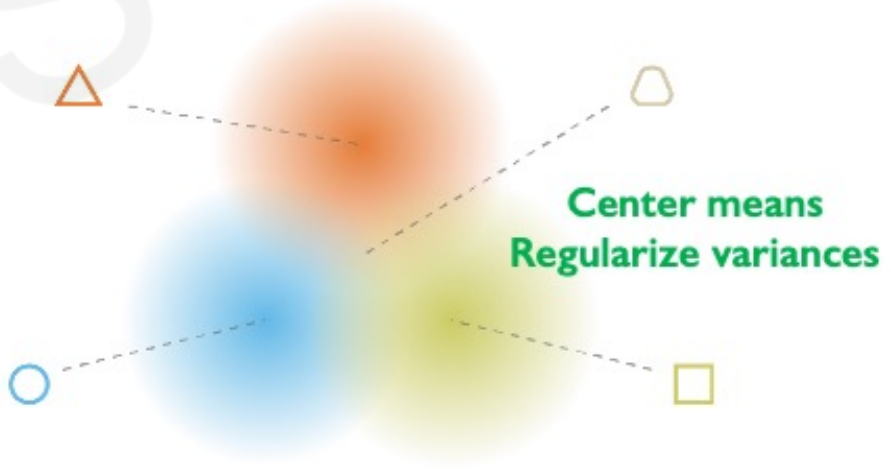
Encoding as a distribution does not guarantee these properties!

Small variances → Pointed distributions

Different means → Discontinuities

Not regularized

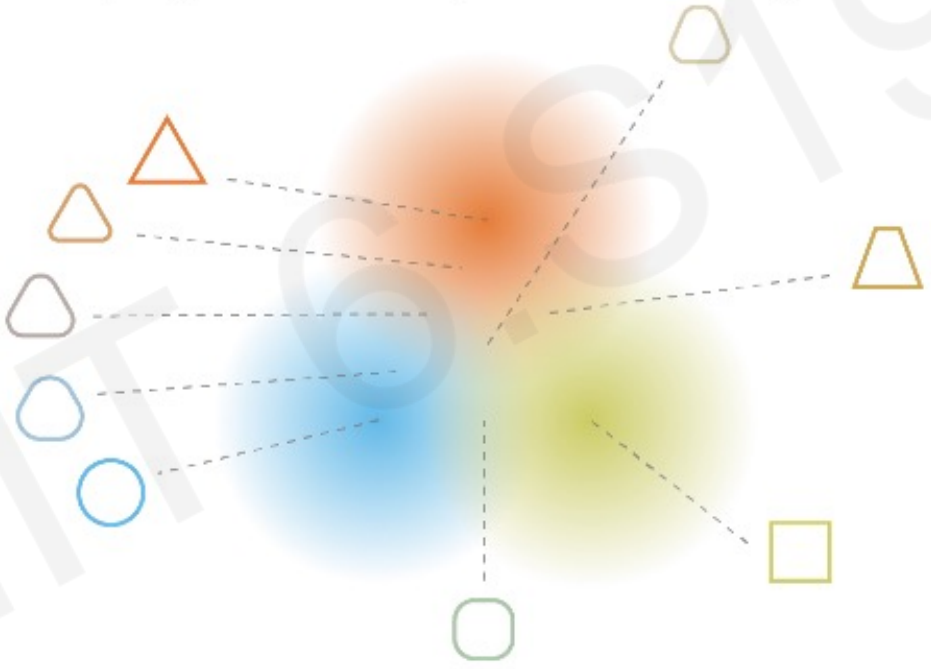
Normal prior → continuity + completeness



Regularized

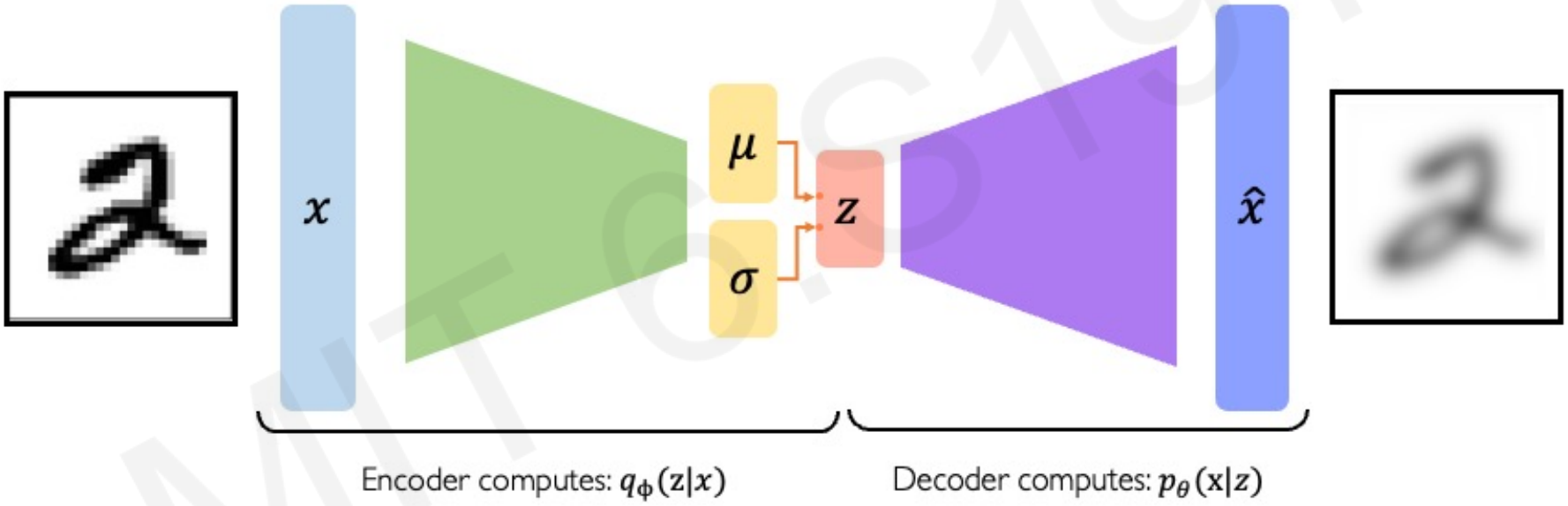
Intuition on regularization and the Normal prior

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



Regularization with Normal prior helps enforce **information gradient** in the latent space.

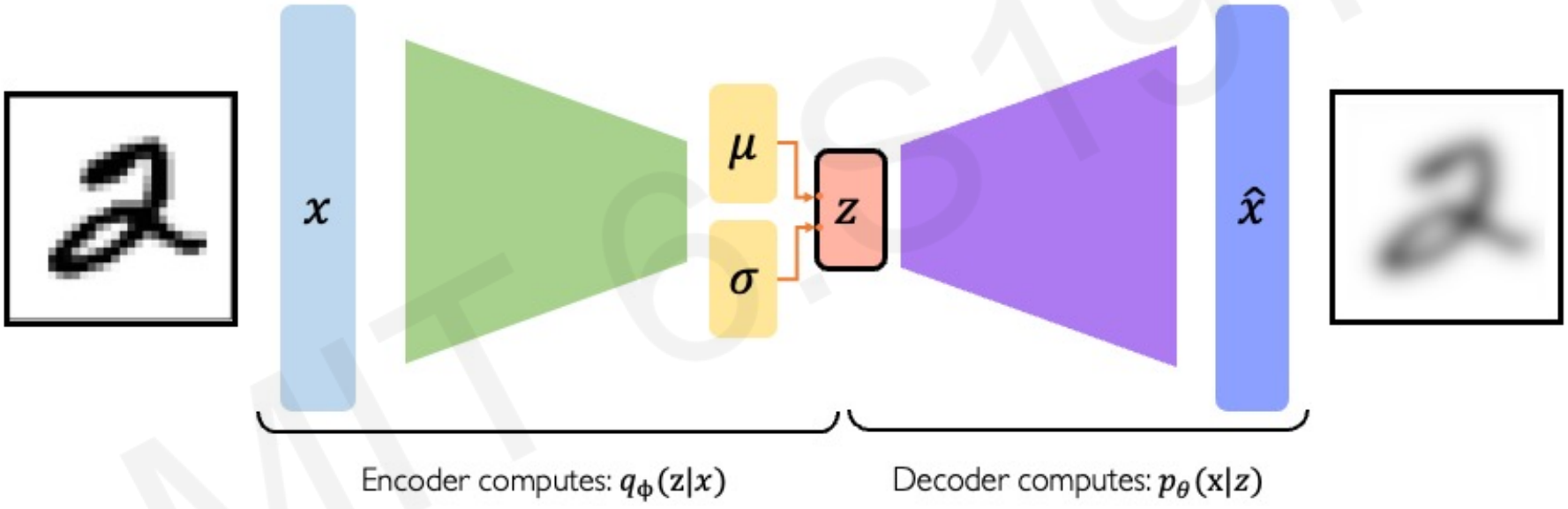
VAE computation graph



$$\mathcal{L}(\phi, \theta, x) = \text{(reconstruction loss)} + \text{(regularization term)}$$

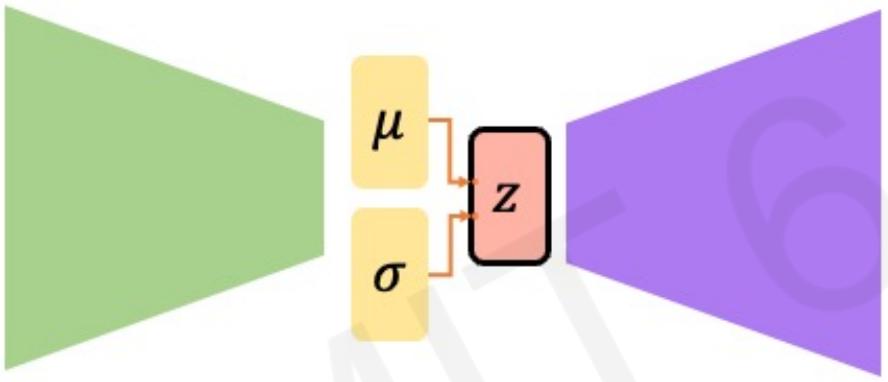
VAE computation graph

Problem: We cannot backpropagate gradients through sampling layers!



$$\mathcal{L}(\phi, \theta, x) = \text{(reconstruction loss)} + \text{(regularization term)}$$

Reparametrizing the sampling layer



Key Idea:

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

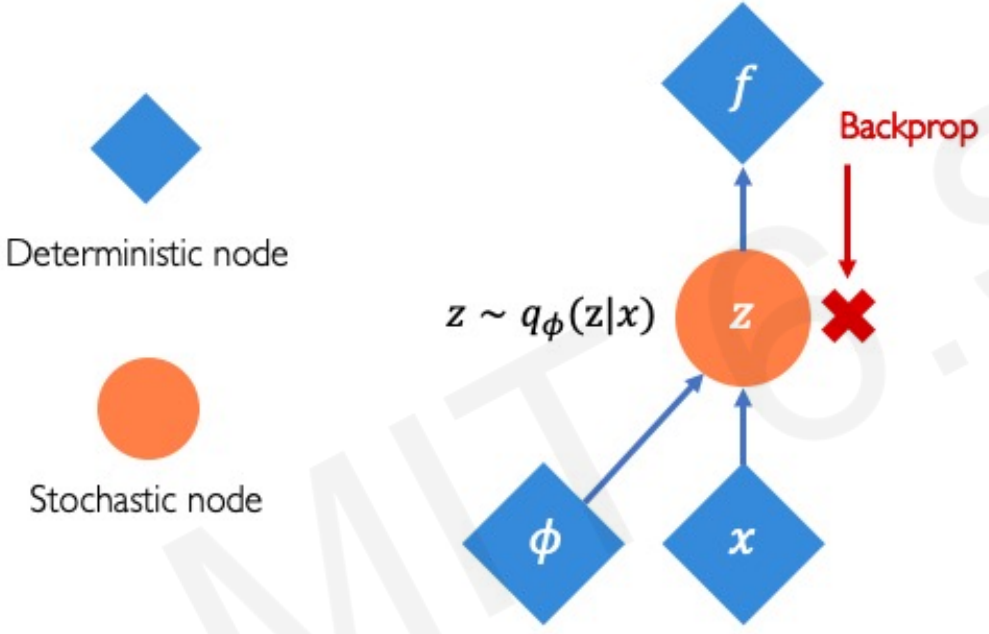
Consider the sampled latent vector z as a sum of

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

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

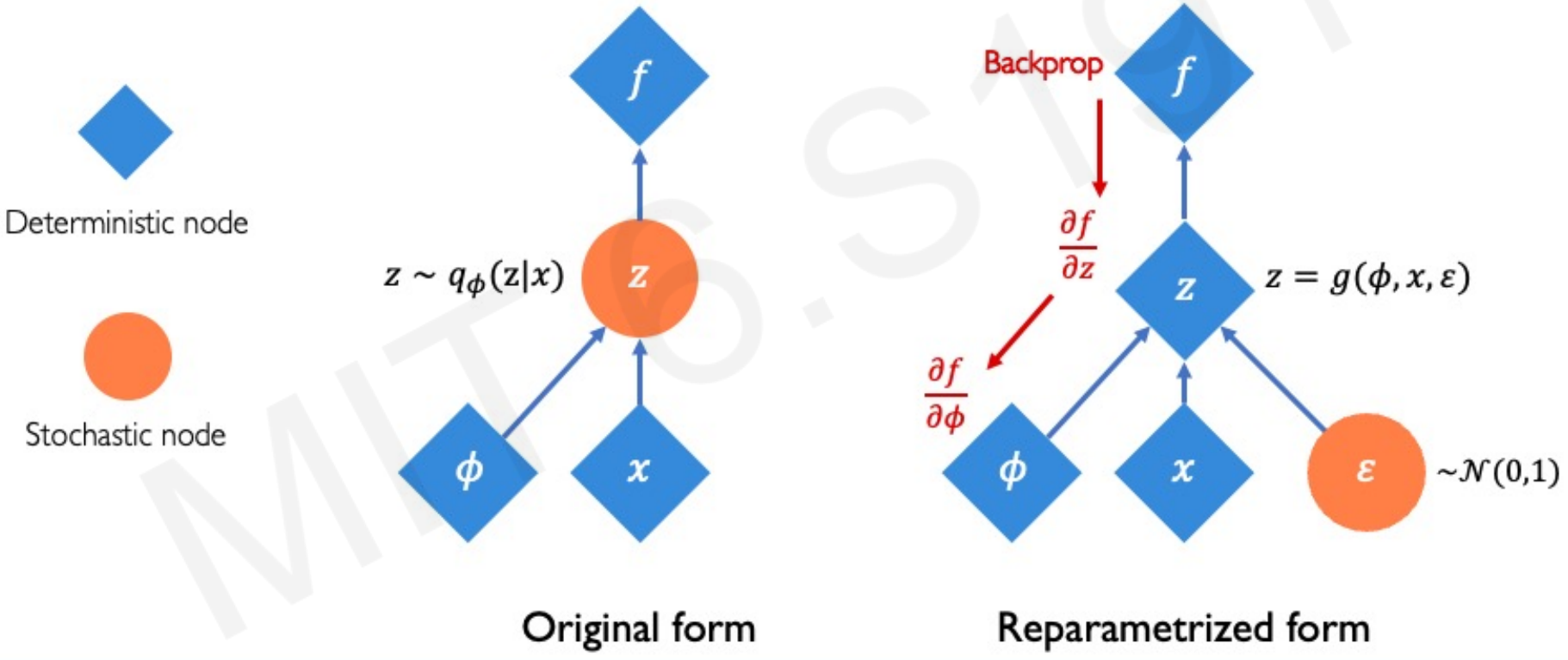
where $\varepsilon \sim \mathcal{N}(0,1)$

Reparametrizing the sampling layer



Original form

Reparametrizing the sampling layer



Original form

Reparametrized form

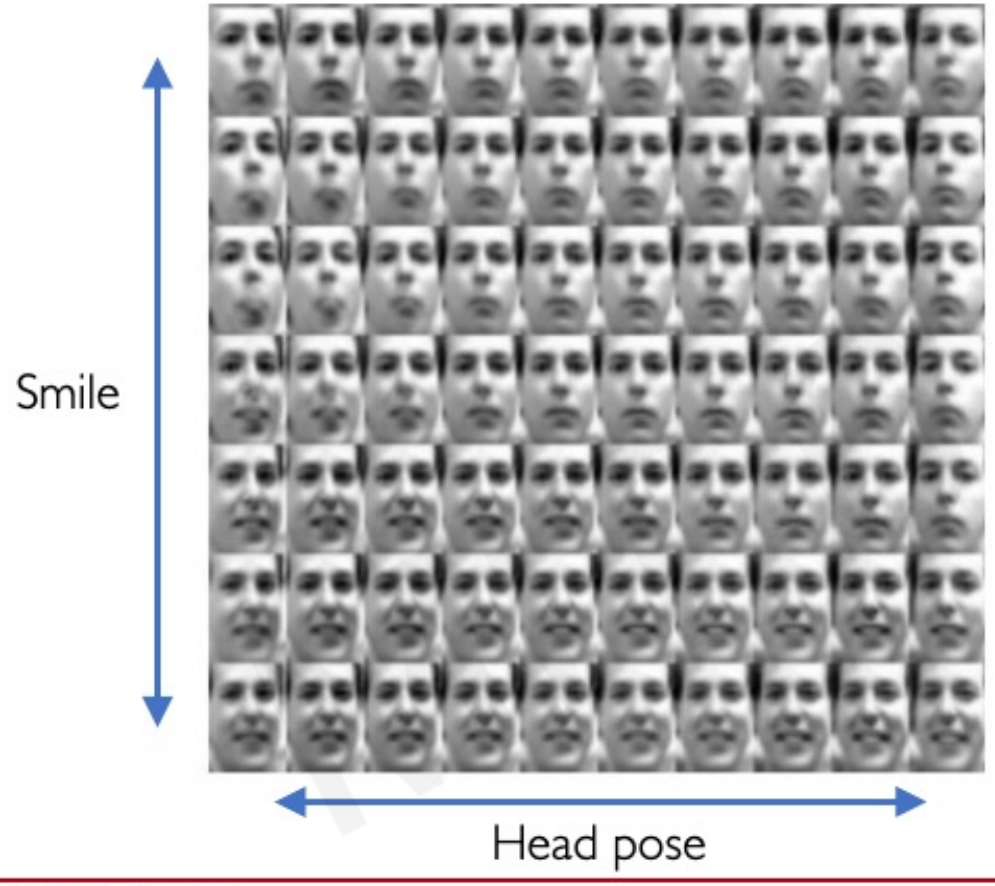
VAEs: Latent perturbation

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

Latent space disentanglement with β-VAEs

β-VAE loss:

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

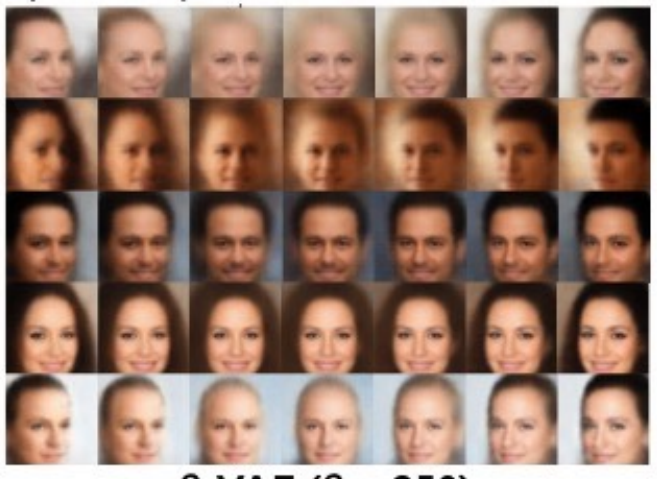
Reconstruction term **Regularization term**

β > 1: constrain latent bottleneck, encourage efficient latent encoding → disentanglement

Head rotation (azimuth)

Smile also changing!

Standard VAE (β = 1)



β-VAE (β = 250)

Smile relatively constant!

Why latent variable models? Debiasing

Capable of uncovering **underlying latent variables** in a dataset

Homogeneous skin color, pose

VS

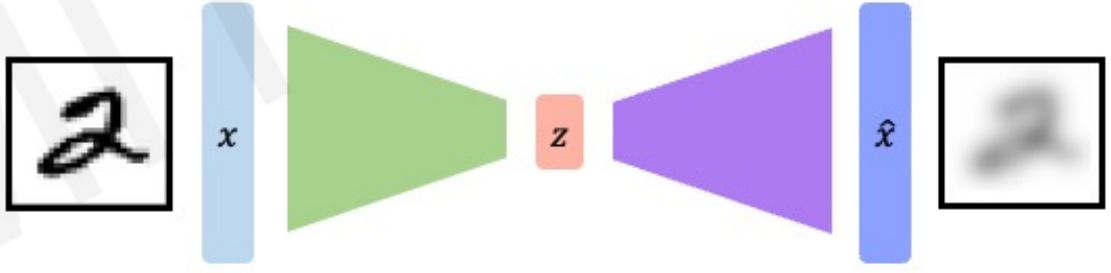
Diverse skin color, pose, illumination

How can we use latent distributions to create fair and representative datasets?

Software Lab!

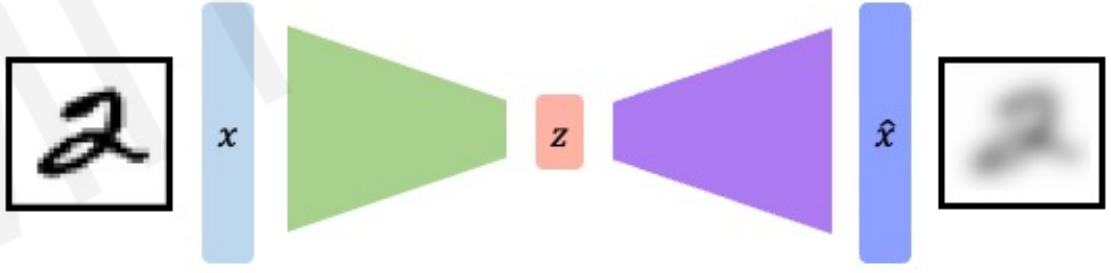
VAE summary

1. Compress representation of world to something we can use to learn



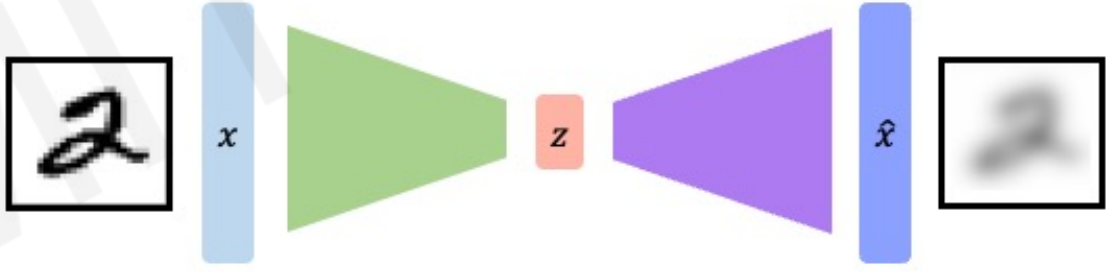
VAE summary

1. Compress representation of world to something we can use to learn
2. Reconstruction allows for unsupervised learning (no labels!)



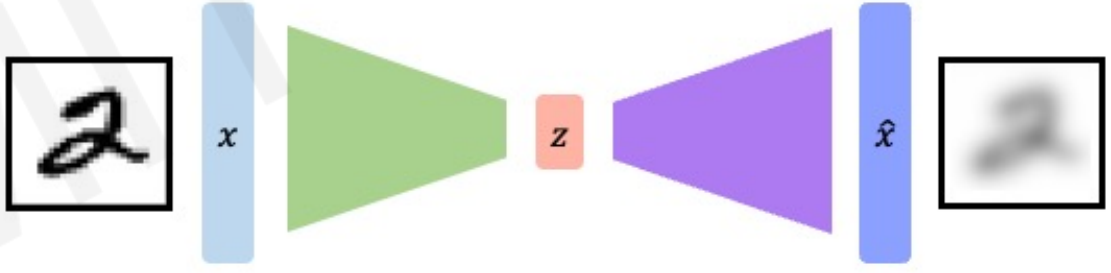
VAE summary

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



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



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