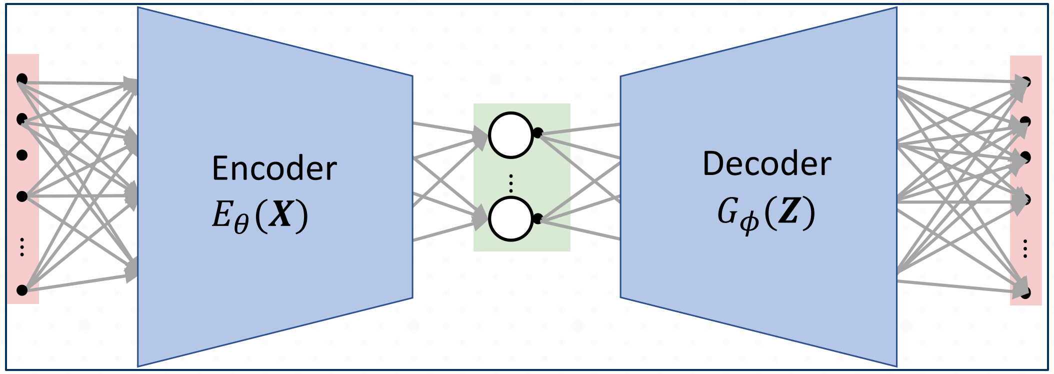
ECE 4252/8803: Fundamentals of Machine Learning (FunML)
Fall 2024

Lecture 19: Autoencoder Extensions



Overview

In this Lecture..

Introduction and Motivation

Fully-connected Autoencoders

Convolutional Autoencoders

Regularized Autoencoders

- Sparse Autoencoders
- Denoising Autoencoders

Variational Autoencoders

Regularized Autoencoders

Motivation

- Standard autoencoders learn useful representations by restricting the dimension to be small
- A higher dimensional representation helps model more complex data distributions
- We want to **regularize** representations to learn important and complex features without restricting the latent dimension

Input data $\boldsymbol{X}$

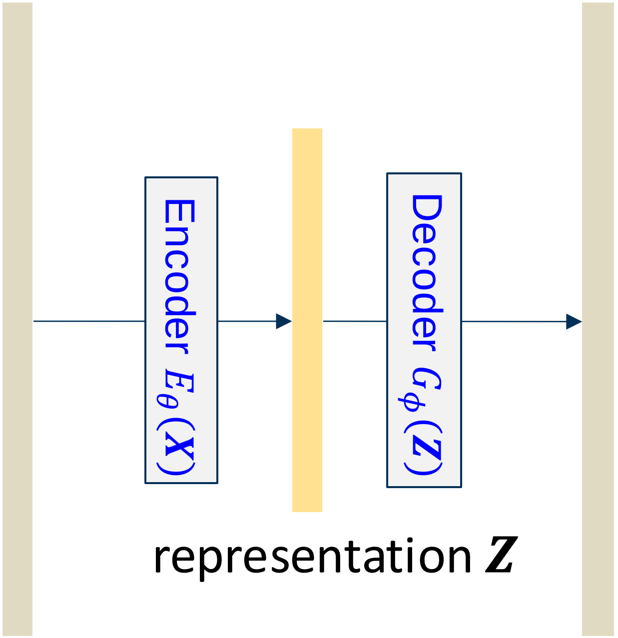
10-dim latent $\boldsymbol{Z}$

reconstruction $\hat{\boldsymbol{X}}$

Input data $\boldsymbol{X}$

2-dim latent $\boldsymbol{Z}$

reconstruction $\hat{\boldsymbol{X}}$

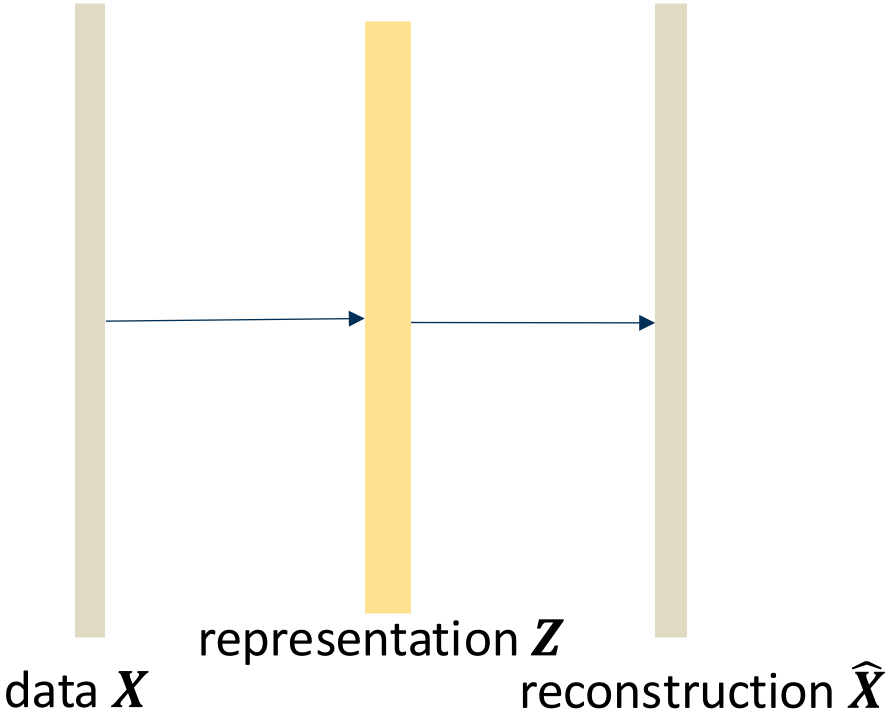


Sparse Autoencoders

Motivation

- We want to avoid identity mapping without limiting the representation capability to extract features from the complex data
- a high-dimensional latent space can represent more complex data
  - $\dim(\boldsymbol{Z})$ can be larger than $\dim(\boldsymbol{X})$
- reduce the redundancy to improve generalization

Redundant information learned by high-dim latent Z

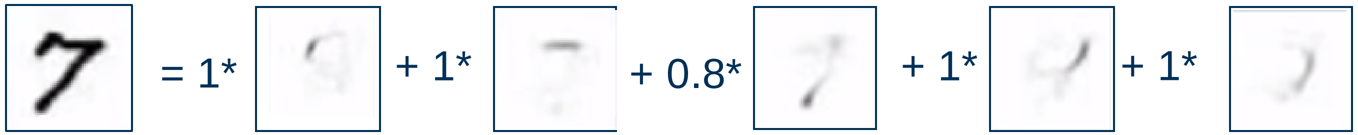


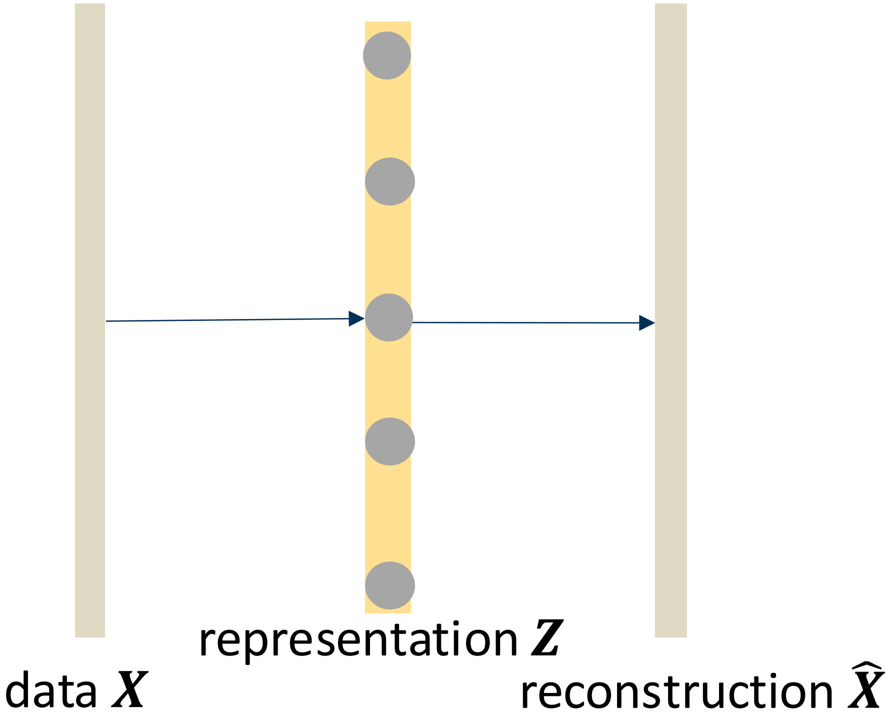
OLIVES @GeorgiaTech

Georgia Tech

Sparse representations:

- Represent data using a subset of latent neurons being active at the same time
- Force the model to learn the unique statistical features that can be used for other tasks such as classification.





Sparse Autoencoders

Motivation

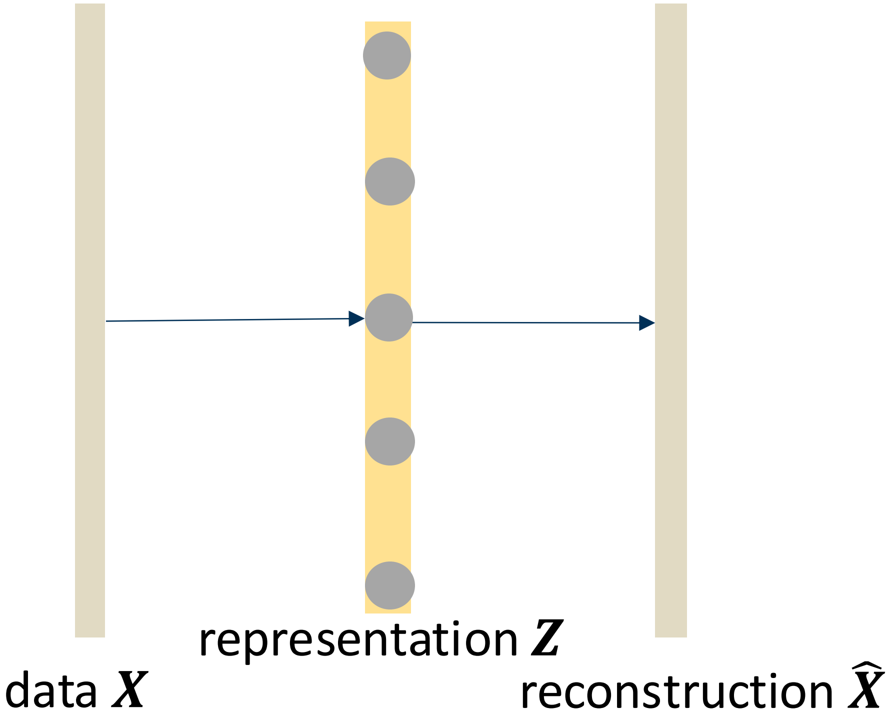
Learning sparse representations:

- Loss $L = \left\|\boldsymbol{X} - \widehat{\boldsymbol{X}}\right\|^2 + \boxed{\lambda\boldsymbol{\Omega}(\boldsymbol{Z})}$

where

- $\boldsymbol{\Omega}(\boldsymbol{Z})$: *sparsity* constraint
- λ: constraint coefficient

- Two methods to enforce sparsity:
  - L_1 regularization
  - Kullback-Leibler (KL) divergence



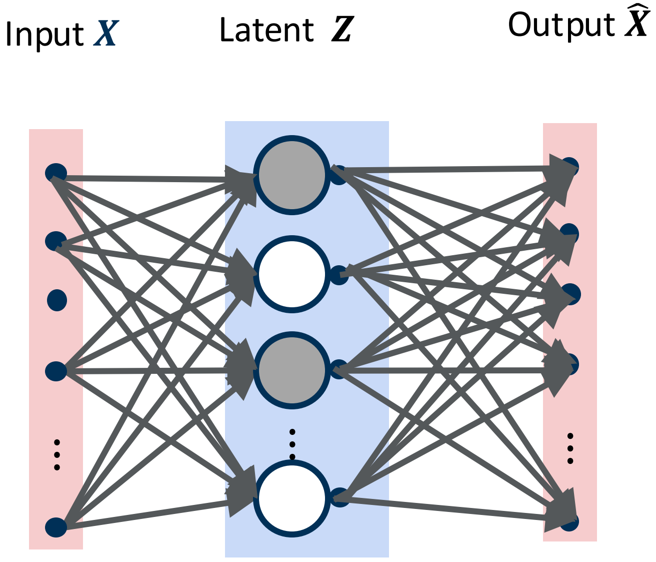
Sparse Autoencoders

Sparsity via $L_!$ Regularization

Exploiting the L_1 norm as sparsity constraint:

L1 norm on activation $\boldsymbol{z}_i$: $\|\boldsymbol{z}_i\|_1 = \sum_j |z_{ij}|$

- z_{ij}: activation of j-th latent neuron given the input $\boldsymbol{x}_i$



Thus,

$$\boldsymbol{\Omega}(\boldsymbol{Z}) = \sum_i \|\boldsymbol{z}_i\|_1$$

suppresses activations towards 0 and achieve sparsity

Sparse Autoencoders

Sparsity via KL Divergence

Goal: The latent neuron activations must be ***near*** 0

Let

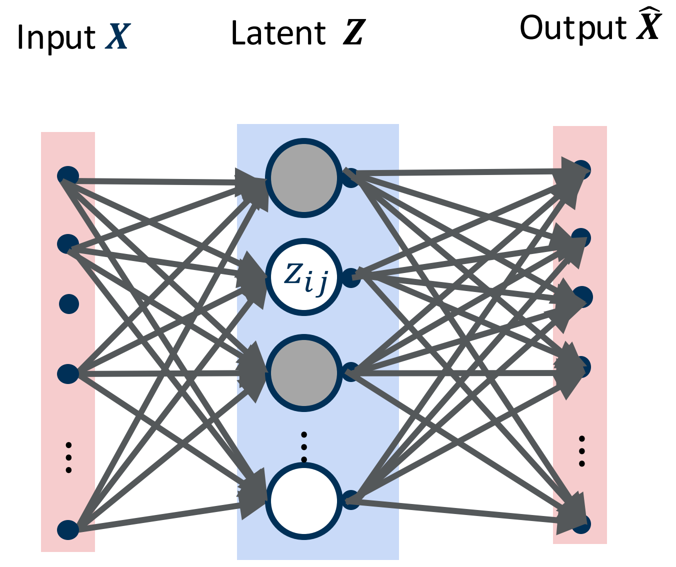
$$\hat{\rho}_{\boldsymbol{j}} = \frac{1}{N}\sum_{i=1}^{N} z_{ij}$$

be the average activation of neuron z_{ij}

where

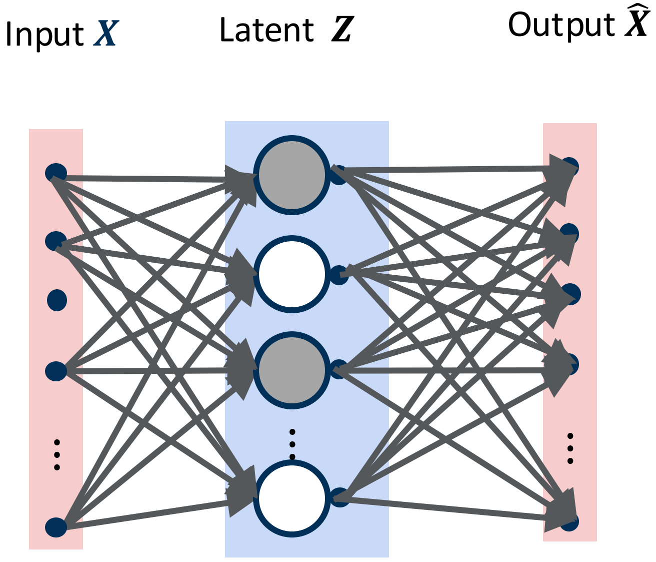
- z_{ij}: activation of j-th latent neuron given the input $\boldsymbol{x}_i$
- N: the number of training samples

Thus, we want to constrain $\hat{\rho}_{\boldsymbol{j}}$ to be close to zero



Sparse Autoencoders
Sparsity via KL Divergence

- Goal: The latent neuron activations must be ***near*** 0
- Assume:
  - using sigmoid
  - z_{ij} is a Bernoulli random variable
  - $\hat{\rho}_{\boldsymbol{j}} = \frac{1}{N}\sum_{i=1}^{N} z_{ij}$ measures the *observed* expectation
- We want z_{ij} to be drawn from a Bernoulli distribution with mean ρ to enforce sparsity
  - Penalize $\hat{\rho}_j$ for deviating from ρ
  - Smaller ρ forces sparser $\boldsymbol{Z}$
- Kullback-Leibler (KL) divergence measures the difference between two distributions



OLIVES @GeorgiaTech
Georgia Tech

Sparse Autoencoders

Sparsity via KL Divergence

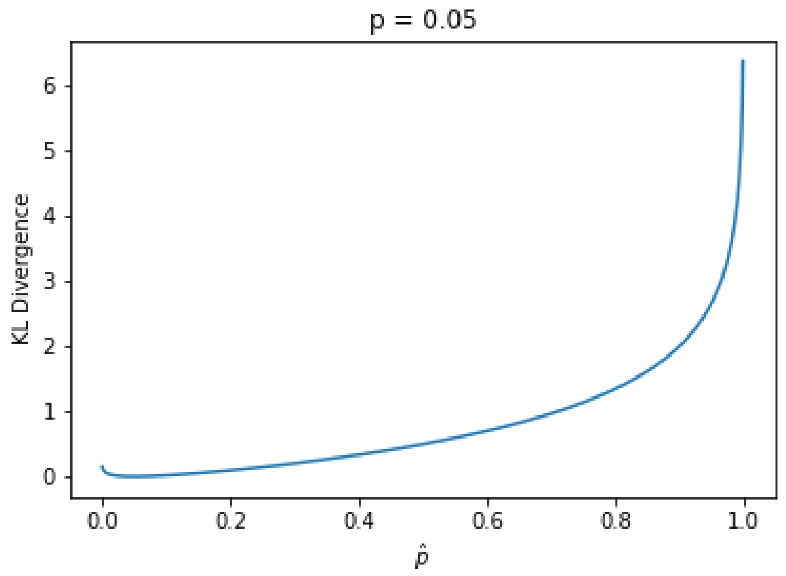
Exploiting the Kullback-Leibler (KL) divergence:

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

$$\boldsymbol{\Omega}(\boldsymbol{Z}) = \sum_j KL(\rho || \hat{\rho}_j) = \sum_j \left(\rho log \frac{\rho}{\hat{\rho}_j} + (1-\rho) log \frac{1-\rho}{1-\hat{\rho}_j}\right)$$

where $\hat{\rho}_j = \frac{1}{N}\sum_{i=1}^{N} z_{ij}(\boldsymbol{x}_i)$

- $KL(\rho || \hat{\rho}_j) = 0$ if $\rho = \hat{\rho}_j$, and increases monotonically as $\hat{\rho}_j$ diverges from ρ
- $KL(\rho || \hat{\rho}_j)$ has **control over sparsity** via ρ
  - Smaller ρ forces sparser $\boldsymbol{Z}$



ρ = 0.05, plot the KL divergence for $\hat{\rho} \in (0, 1)$

Sparse Autoencoders

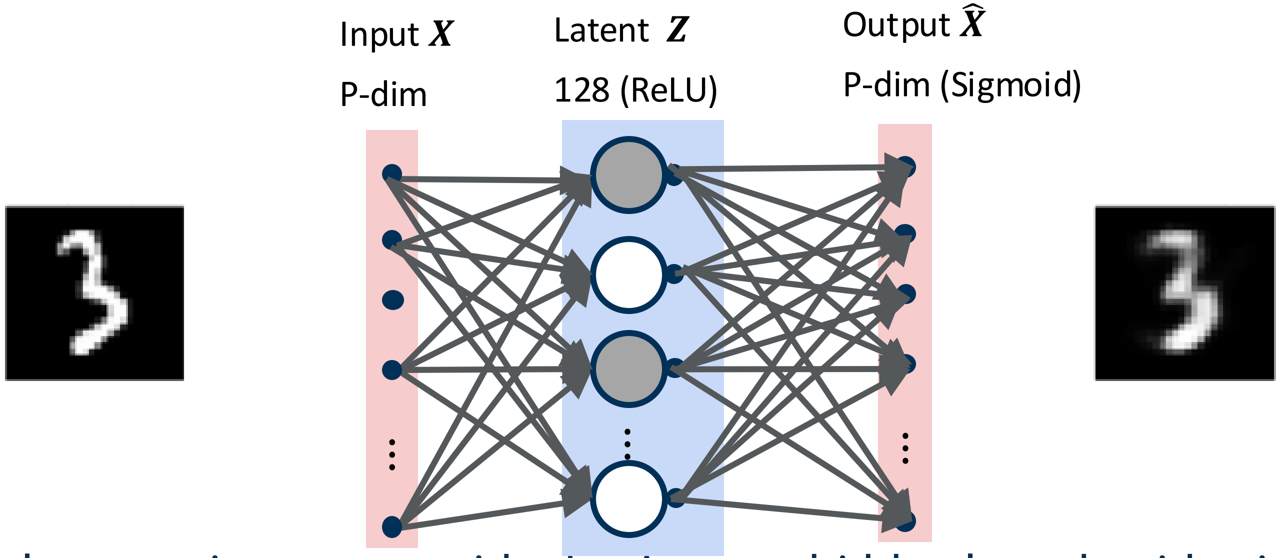
Differences in Sparsity Constraints

- Kullback-Leibler (KL) divergence
  - control over sparsity via ρ
  - Approximate sparsity, suppress activations close to small value
  - differentiable at 0

- L_1 regularization
  - no explicit control over sparsity
  - True sparsity, drive activations towards 0
  - not differentiable at 0

Sparse Autoencoders
Training

- **Sparsity** constraint on the latent activation
- Loss $L = \|X - \hat{X}\|^2 + \boxed{\lambda \Omega(Z),}$ where λ is the constraint coefficient

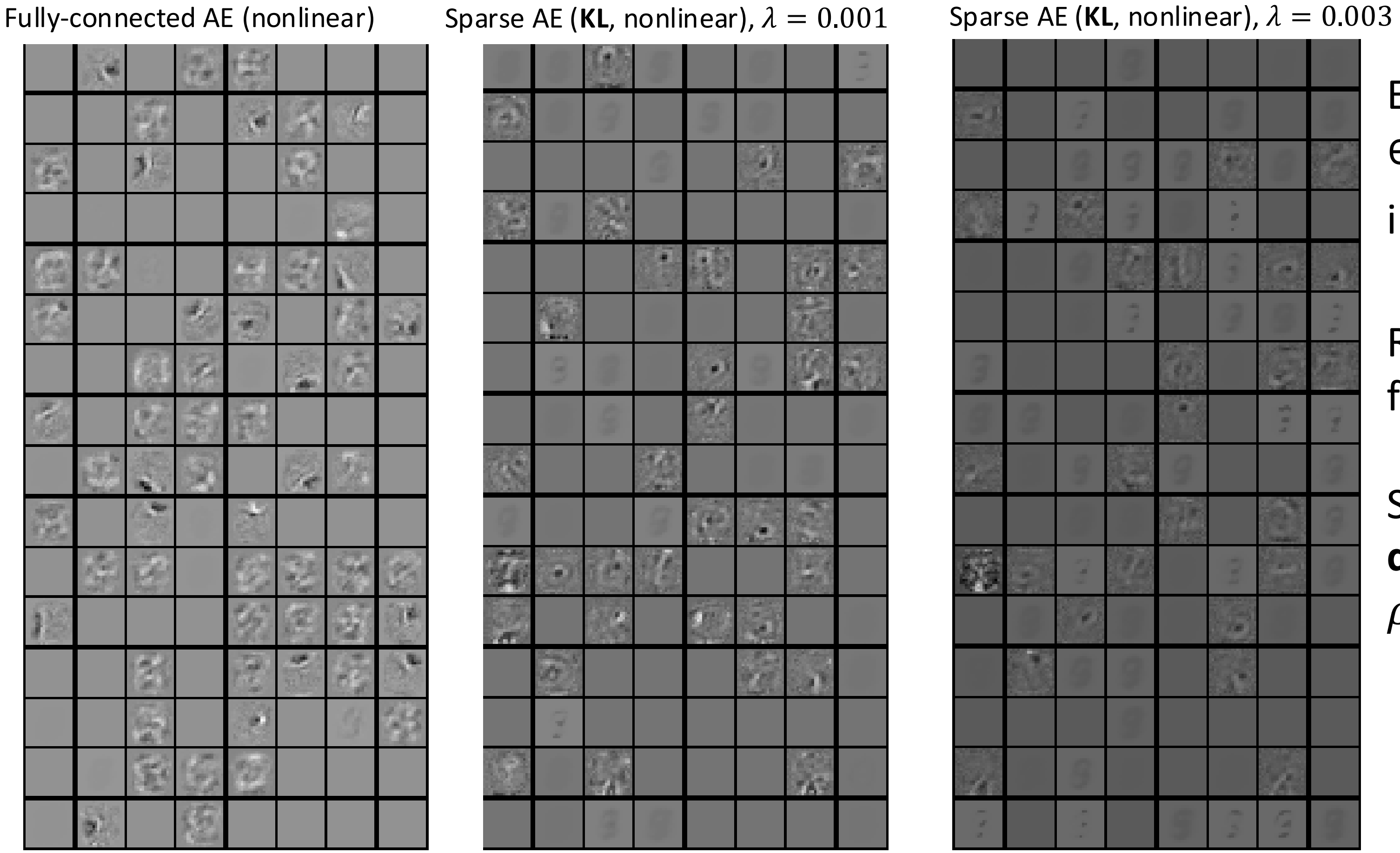


- For simpler implementation, we consider having one hidden layer besides input and output

Sparse Autoencoders

Visualization of Trained Filters

Visualize the filters of the first layer of a fully-connected AE and a sparse AE with the same architecture components.



Extract the weights $\boldsymbol{W}^{(1)} \in \mathbb{R}^{\mathbf{128}\times(\mathbf{1}\times\mathbf{28}\times\mathbf{28})}$ in **layer 1** @ E_θ

Reshape $\boldsymbol{W}^{(1)}$ into 128 filters of size 28x28

Sparse AE uses **KL divergence** with $\rho = 0.05$

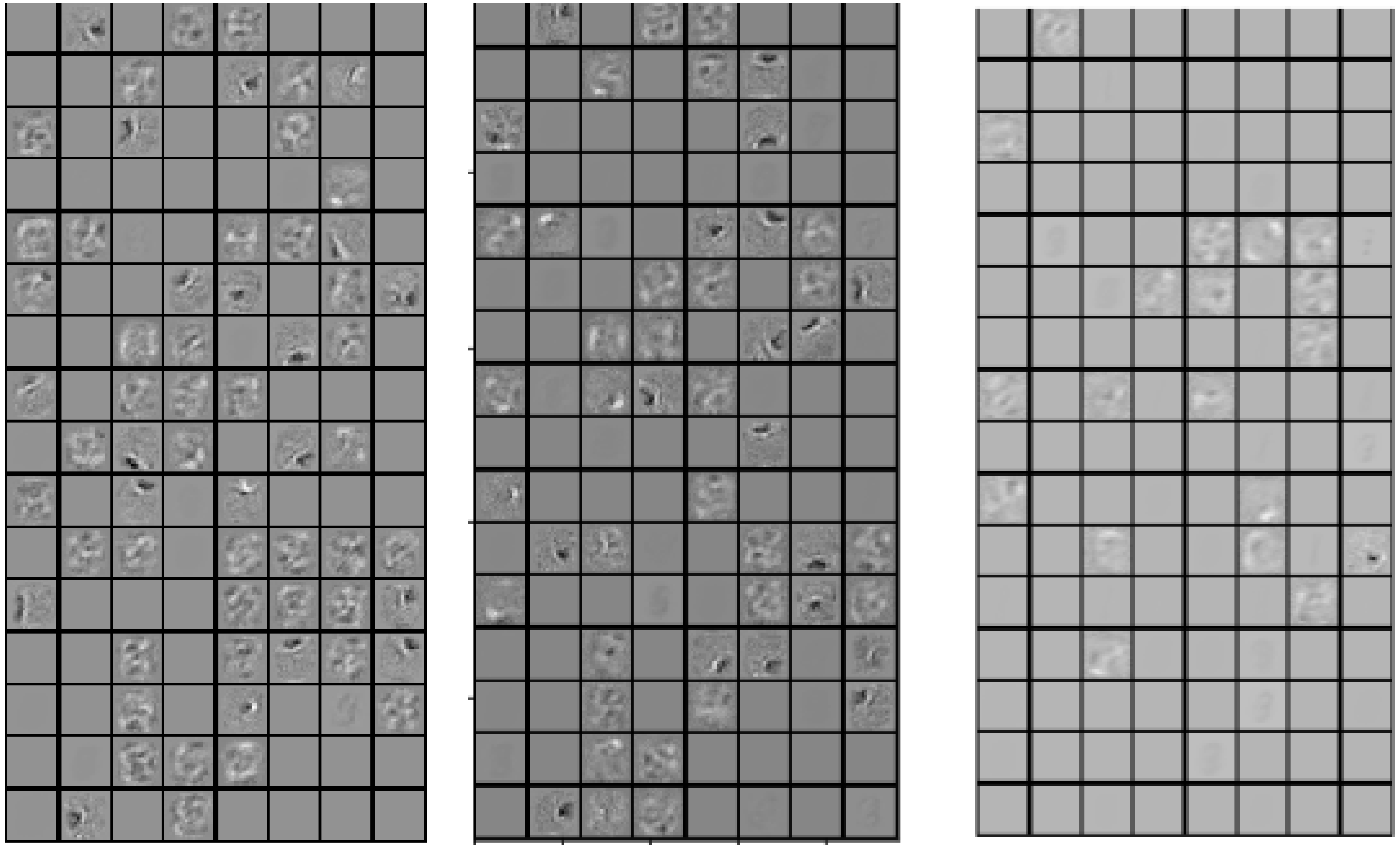
OLIVES @GeorgiaTech

Georgia Tech

Sparse Autoencoders
Visualization of Trained Filters

Visualize the filters of the first layer of a fully-connected AE and a sparse AE with the same architecture components.

Fully-connected AE (nonlinear) | Sparse AE (**L1**, nonlinear), $\lambda = 0.001$ | Sparse AE (**L1**, nonlinear), $\lambda = 0.005$



Extract the weights $\boldsymbol{W}^{(1)} \in \mathbb{R}^{\mathbf{128\times(1\times28\times28)}}$ in **layer 1** @ E_θ

Reshape $\boldsymbol{W}^{(1)}$ into 128 filters of size 28x28

Sparse AE uses **L1 regularization**

Overview

In this Lecture..

Introduction and Motivation

Fully-connected Autoencoders

Convolutional Autoencoders

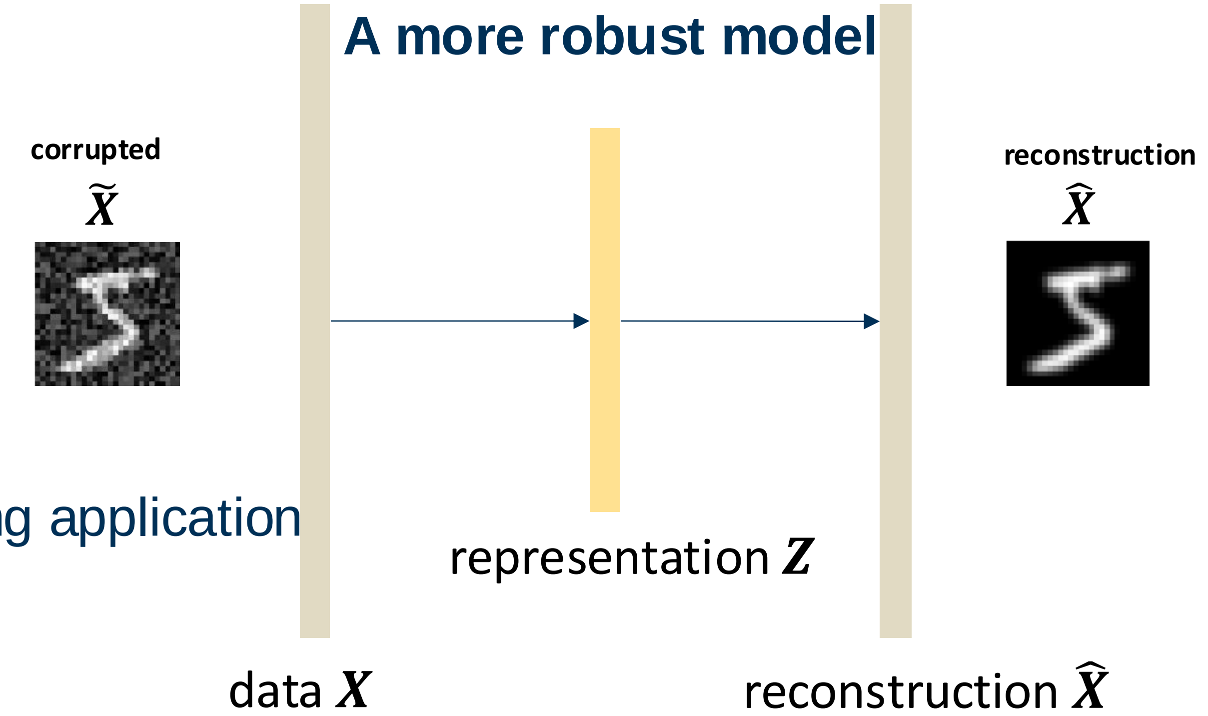
Regularized Autoencoders

- Sparse Autoencoders
- Denoising Autoencoders

Variational Autoencoders

Denoising Autoencoders
Motivation

- We still aim to encode the input and avoid identity mapping
- We try to learn a **robust representation** that undo the effect of *corruption* applied to the input

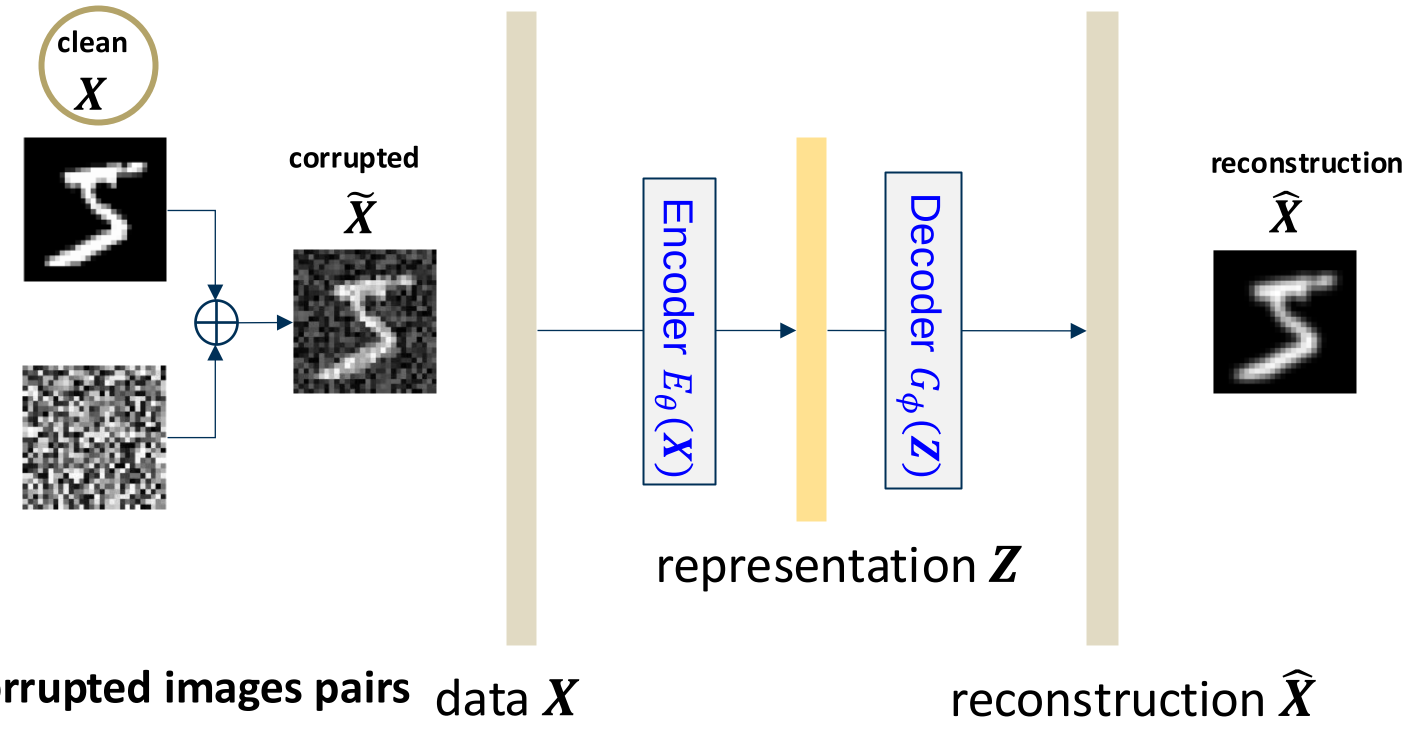


Can be used for denoising application

Denoising Autoencoders
Motivation

Learn a **robust representation** from **corrupted input** $\widetilde{X}$

Minimize a loss $L = \|X - \widehat{X}\|^2, \widehat{X} = G_\phi(E_\theta(\widetilde{X}))$

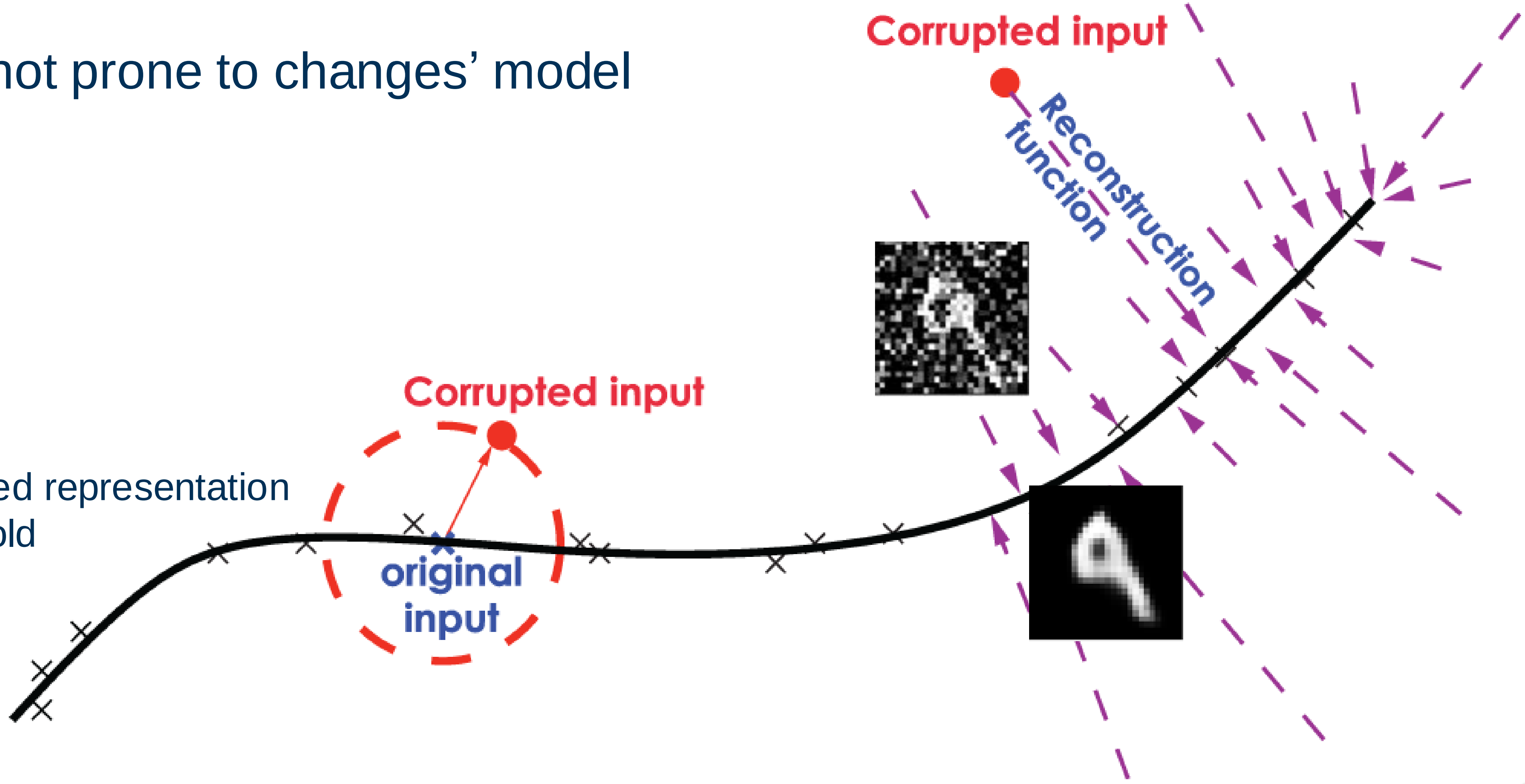


need **clean** and **corrupted images pairs**

Denoising Autoencoders
Motivation

A ‘not prone to changes’ model

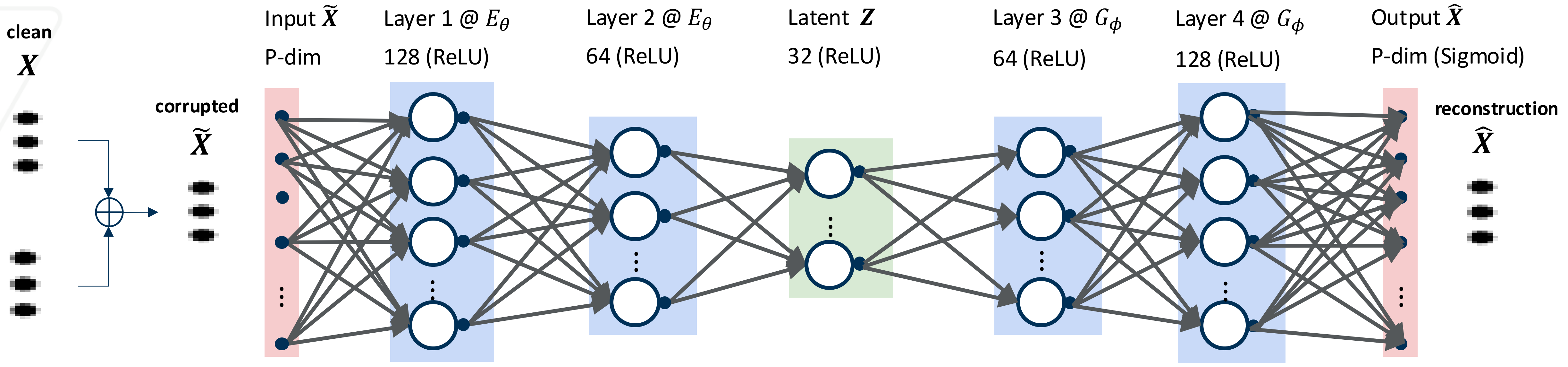
Learned representation manifold



Denoising Autoencoders
Training

Learn a **robust representation** from **corrupted input** $\tilde{\boldsymbol{X}}$

Minimize a loss $L = \left\|\boldsymbol{X} - \hat{\boldsymbol{X}}\right\|^2, \hat{\boldsymbol{X}} = G_\phi(E_\theta(\tilde{\boldsymbol{X}}))$



need **clean** and **corrupted images pairs**

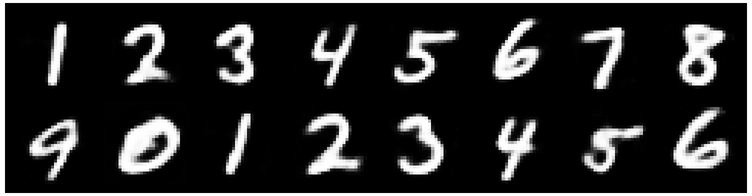
Denoising Autoencoders

Results

Denoising Fully-connected AE (linear)

MSE = 0.0694

Denoising Fully-connected AE (nonlinear)



MSE = 0.0328

input (upper) and reconstruction (bottom)

Overview

In this Lecture..

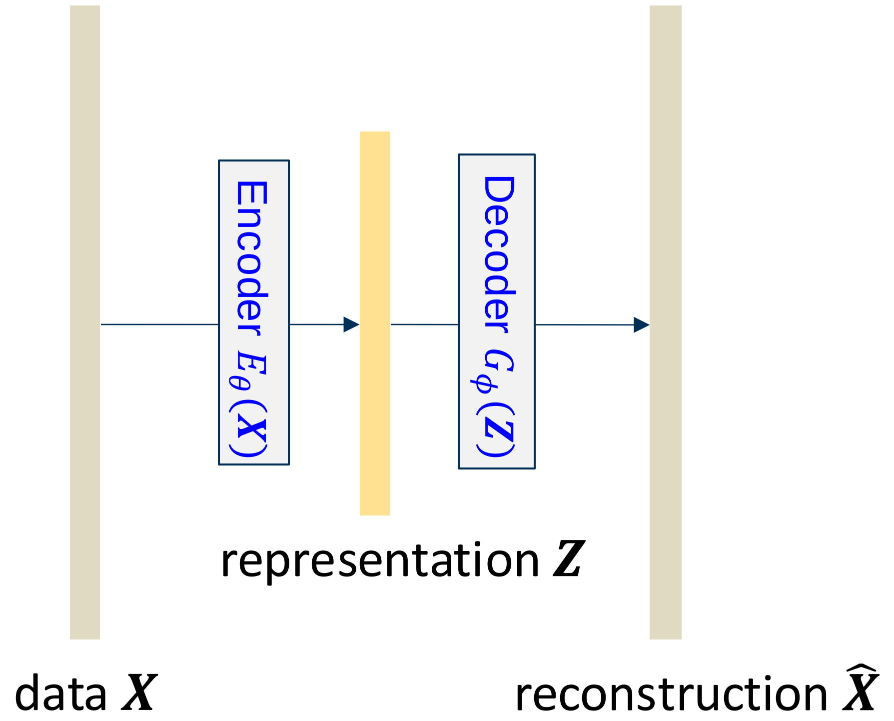
- Introduction and Motivation
- Fully-connected Autoencoders
- Convolutional Autoencoders
- Regularized Autoencoders
- Variational Autoencoders
  - Motivation
  - Variational Lower Bound
  - Reparameterization Trick
  - Generating Novel Samples
  - Visualization

Variational Autoencoders

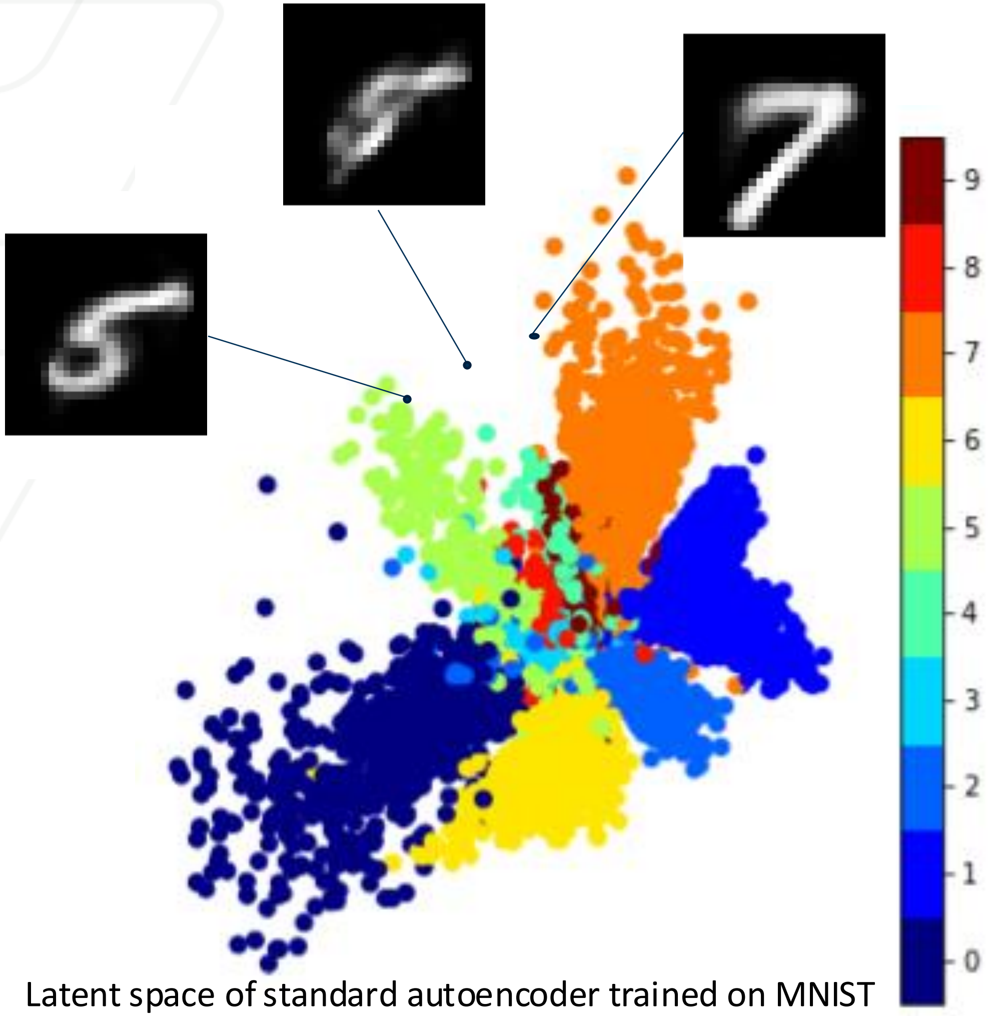
Motivation

- Autoencoders can **reconstruct** the input data via deterministic processes $\boldsymbol{Z} = E_\theta(\boldsymbol{X})$, $\widehat{\boldsymbol{X}} = G_\phi(\boldsymbol{Z})$

- Can we generate novel samples?
  - If $\boldsymbol{Z}$ is a random variable drawn from some distribution, we can *sample* from the latent space and **generate novel variations** of inputs



Variational Autoencoders
Motivation



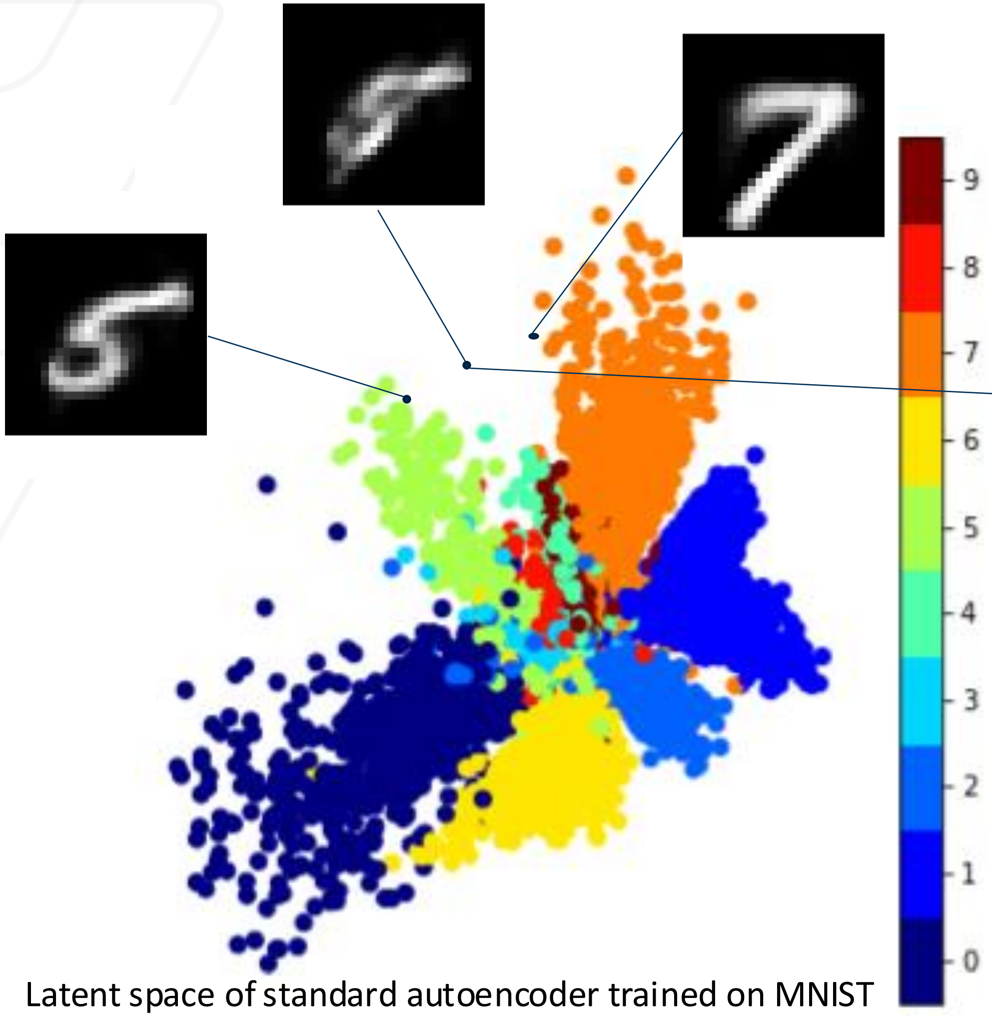
Latent space of standard autoencoder trained on MNIST

Recall the latent space of regular autoencoders:

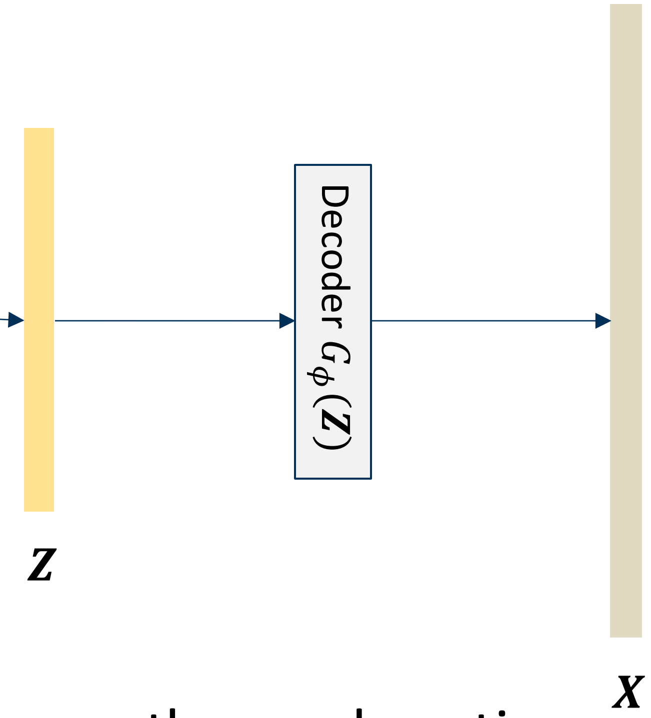
- Discontinuity
  - decoder generates an unrealistic output if sample/generate a variation from there

Variational Autoencoders

Motivation



Latent space of standard autoencoder trained on MNIST



Desire a smoother and continuous latent space:

- $\boldsymbol{Z}$ is drawn from some probabilistic distribution
- When uniformly sample $\boldsymbol{Z}$, the decoded images vary with smooth transition

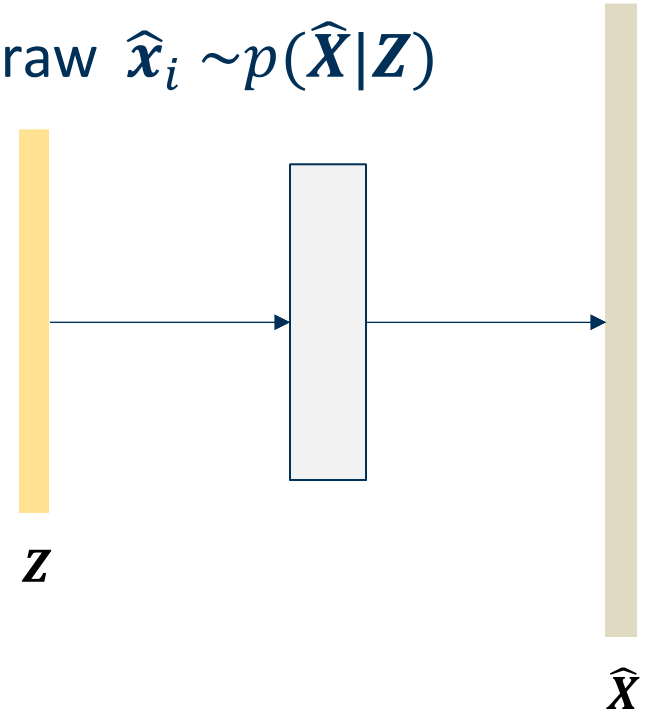
Variational Autoencoders

Probabilistic Perspective

To generate datapoint $\hat{\boldsymbol{x}}_i$:

- Assume we sample latent variables $\boldsymbol{z}_i \sim p(\boldsymbol{Z})$
  - where $p(\boldsymbol{Z})$ is a prior distribution
- Given sampled $\boldsymbol{z}_i \sim p(\boldsymbol{Z})$, draw $\hat{\boldsymbol{x}}_i \sim p(\hat{\boldsymbol{X}}|\boldsymbol{Z})$

Sample $\boldsymbol{z}_i$ from prior $p(\boldsymbol{Z})$



Conditional distribution $p(\hat{\boldsymbol{X}}|\boldsymbol{Z})$

Variational Autoencoders

Probabilistic Perspective

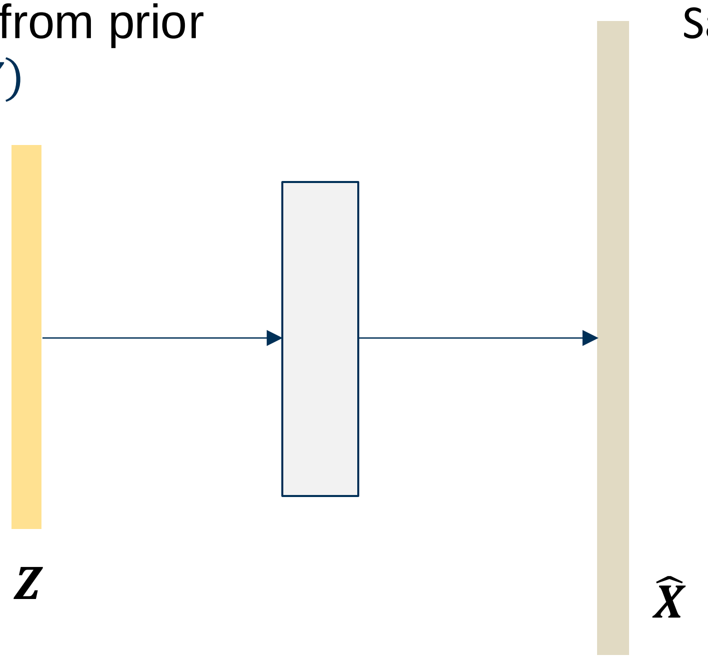
We want to find a probabilistic modeling $p_\phi(\boldsymbol{X})$ to estimate true distribution of $\boldsymbol{X}$

$$p_\phi(\boldsymbol{X}) = \int p_\phi(\boldsymbol{Z}) p_\phi(\boldsymbol{X}|\boldsymbol{Z}) dZ$$

ϕ are the parameters for the generation model

Sample $\boldsymbol{z}_i$ from prior $p(\boldsymbol{Z})$

Sample Conditional distribution $p(\widehat{\boldsymbol{X}}|\boldsymbol{Z})$



Variational Autoencoders
Probabilistic Perspective

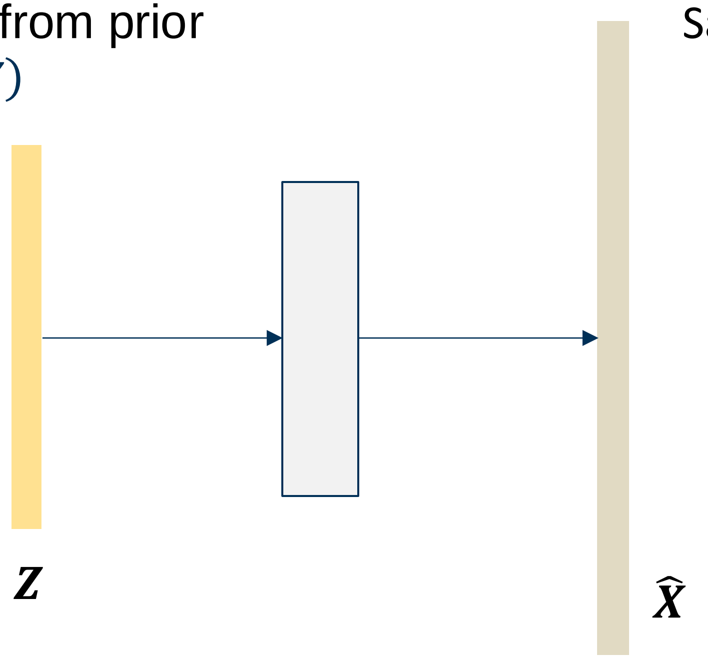
We want to find a probabilistic modeling $p_\phi(\boldsymbol{X})$ to estimate true distribution of $\boldsymbol{X}$

$$p_\phi(\boldsymbol{X}) = \int p_\phi(\boldsymbol{Z}) p_\phi(\boldsymbol{X}|\boldsymbol{Z}) dZ$$

ϕ are the parameters for the generation model

Sample $\boldsymbol{z}_i$ from prior $p(\boldsymbol{Z})$

Sample Conditional distribution $p(\widehat{\boldsymbol{X}}|\boldsymbol{Z})$



How should we represent this model?

- Choose **prior** $p(\boldsymbol{Z})$ to be **simple and tractable**, e.g., *Gaussian* $\mathcal{N}(0, \mathbf{I})$.
- **Conditional** $p(\widehat{\boldsymbol{X}}|\boldsymbol{Z})$ is **complex** (generates image) → modeled by **decoder neural network**

Variational Autoencoders
Probabilistic Modeling via MLE

We want to find a probabilistic modeling $p_\phi(\boldsymbol{X})$ to estimate true distribution of $\boldsymbol{X}$

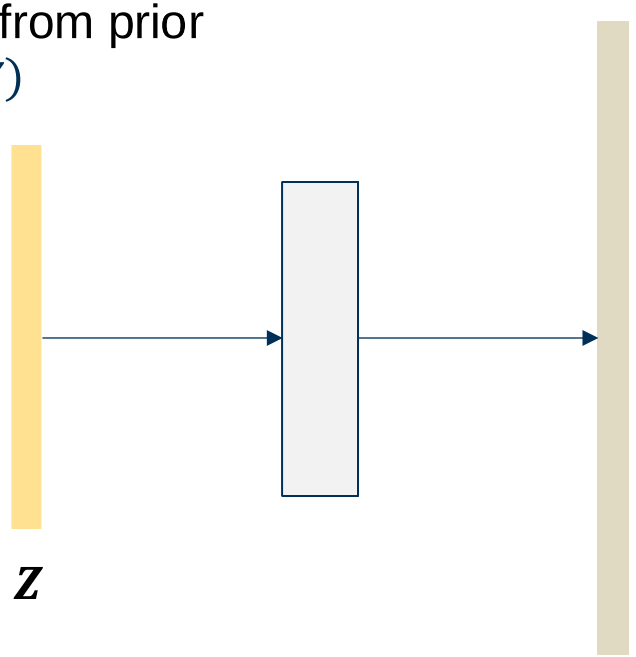
$$p_\phi(\boldsymbol{X}) = \int p_\phi(\boldsymbol{Z})p_\phi(\boldsymbol{X}|\boldsymbol{Z})dZ$$

Simple Gaussian prior

ϕ are the parameters for the generation model

Sample $\boldsymbol{z}_i$ from prior $p(\boldsymbol{Z})$

Sample Conditional distribution $p(\widehat{\boldsymbol{X}}|\boldsymbol{Z})$



$\boldsymbol{Z}$ $\widehat{\boldsymbol{X}}$

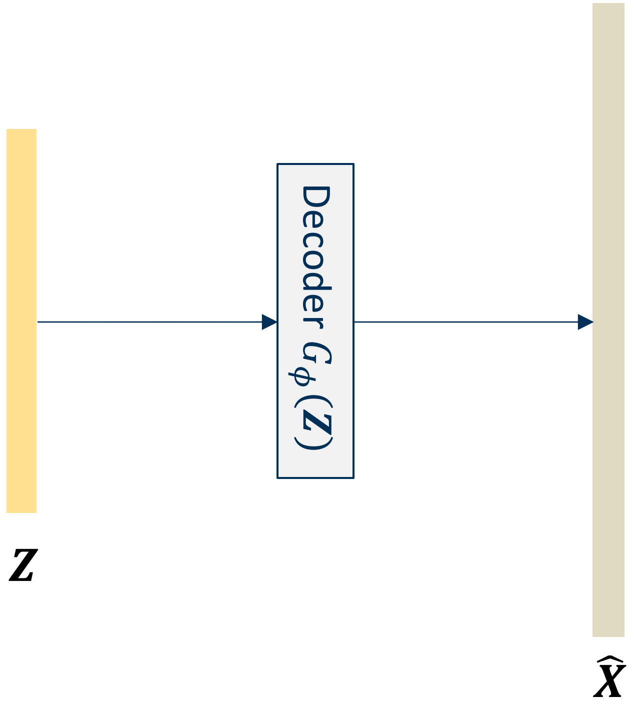
Find the optimal ϕ via **maximizing likelihood** of training data

Variational Autoencoders

Data Likelihood is Intractable

Data likelihood: $p_\phi(\boldsymbol{X}) = \int p_\phi(\boldsymbol{Z})\boxed{p_\phi(\boldsymbol{X}|\boldsymbol{Z})}dZ$

Intractable to compute $p(X|Z)$ and integrate for every dimension of Z!



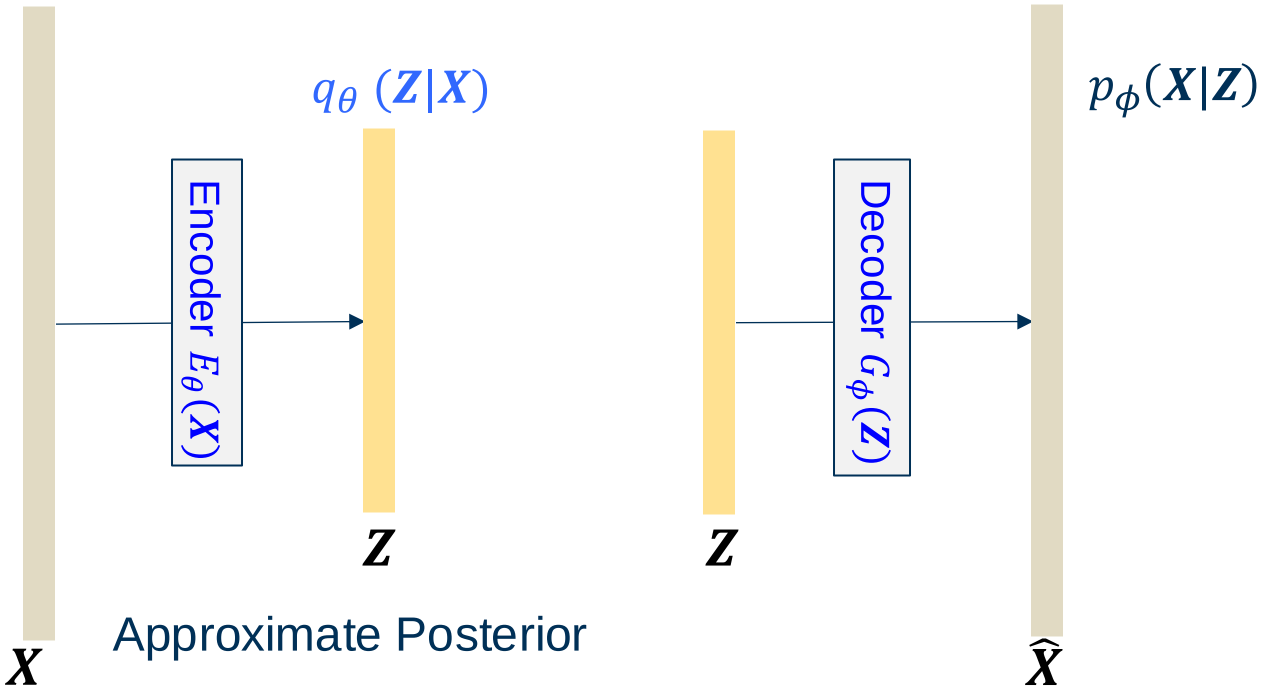
Variational Autoencoders

Posterior is Intractable

Data likelihood: $p_\phi(\boldsymbol{X}) = \int p_\phi(\boldsymbol{Z}) p_\phi(\boldsymbol{X}|\boldsymbol{Z}) dZ$

Posterior density also intractable: $p_\phi(\boldsymbol{Z}|\boldsymbol{X}) = p_\phi(\boldsymbol{X}|\boldsymbol{Z}) p_\phi(\boldsymbol{Z}) / p_\phi(\boldsymbol{X})$

use an encoder to model $q_\theta(\boldsymbol{Z}|\boldsymbol{X})$ that approximates $p_\phi(\boldsymbol{Z}|\boldsymbol{X})$



Variational Autoencoders

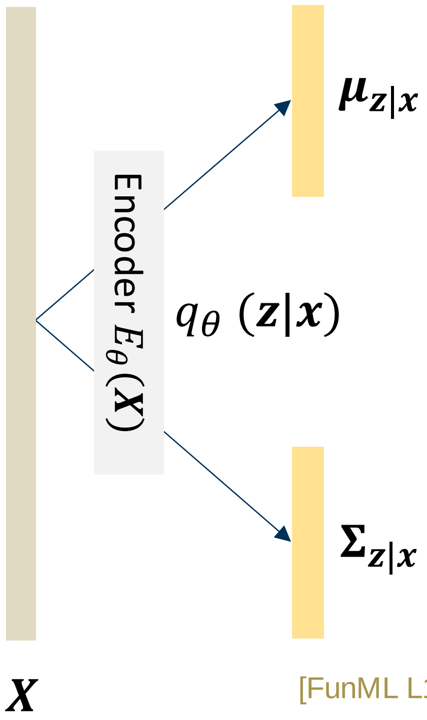
Variational Lower Bound

Reconstruct the input data

$$\underbrace{\mathbf{E}_{\mathbf{z}}\left[\log p_{\phi}(\boldsymbol{x}_{\boldsymbol{i}}|\mathbf{z})\right] - D_{KL}(q_{\theta}\,(\mathbf{z}|\boldsymbol{x}_{\boldsymbol{i}})||p_{\phi}(\boldsymbol{z}))}_{L(x_i,\theta,\phi)}$$

VAE maximizes variational lower bound ("ELBO")

Make approximate posterior distribution $q_{\theta}\,(\mathbf{z}|\boldsymbol{x}_{\boldsymbol{i}})$ close to a tractable prior $p_{\phi}(\boldsymbol{z})$, e.g., Gaussian $\mathcal{N}(0, \mathbf{I})$



Encoder estimates two quantities:

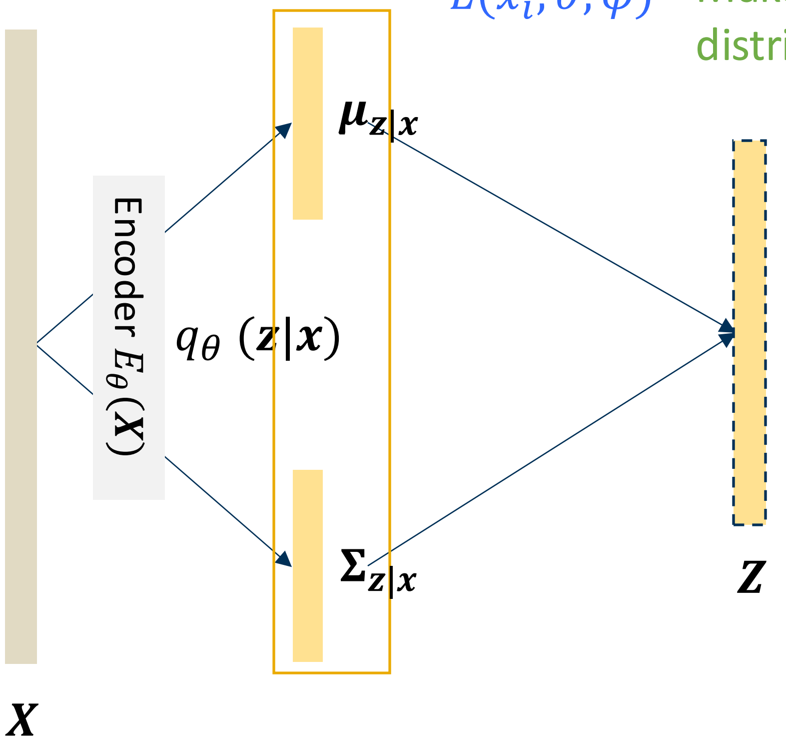
- $\boldsymbol{\mu}_{\boldsymbol{z}|\boldsymbol{x}}$: **mean** of $\boldsymbol{Z}|\boldsymbol{X}$
- $\boldsymbol{\Sigma}_{\boldsymbol{z}|\boldsymbol{x}}$: (diagonal)**covariance** of $\boldsymbol{Z}|\boldsymbol{X}$

Variational Autoencoders

Variational Lower Bound

$$\underbrace{\mathbf{E}_{\mathbf{z}}\left[\log p_\phi(\boldsymbol{x}_i|\mathbf{z})\right] - D_{KL}\boxed{(q_\theta\ (\mathbf{z}|\boldsymbol{x}_i)||p_\phi(\mathbf{z}))}}_{L(x_i,\theta,\phi)}$$

Make approximate posterior distribution close to prior



Assume tractable prior $p_\phi(\boldsymbol{z})$, e.g., Gaussian.

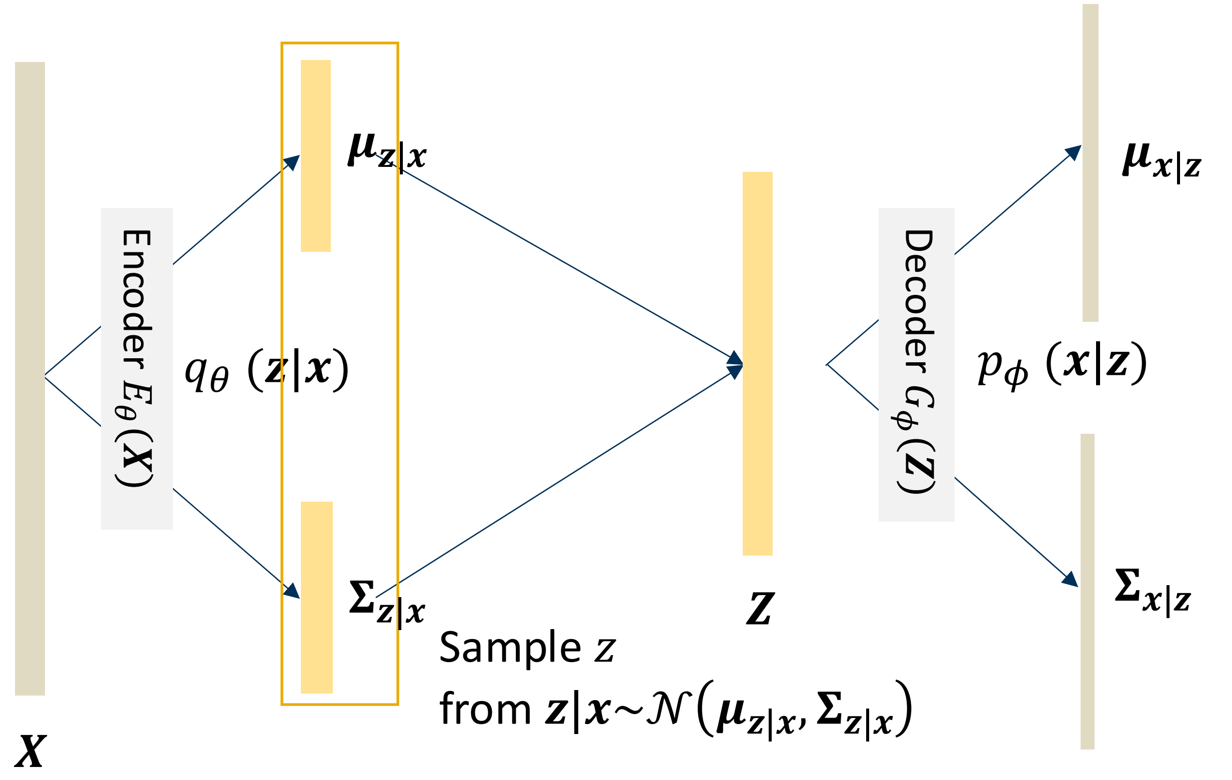
We can **sample** $\boldsymbol{z}$ from $\boldsymbol{z}|\boldsymbol{x} \sim \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{z}|\boldsymbol{x}}, \boldsymbol{\Sigma}_{\boldsymbol{z}|\boldsymbol{x}}\right)$

(Z is not directly computed by VAE)

Variational Autoencoders

Variational Lower Bound

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



OLIVES @GeorgiaTech

Georgia Tech

Variational Autoencoders

Variational Lower Bound

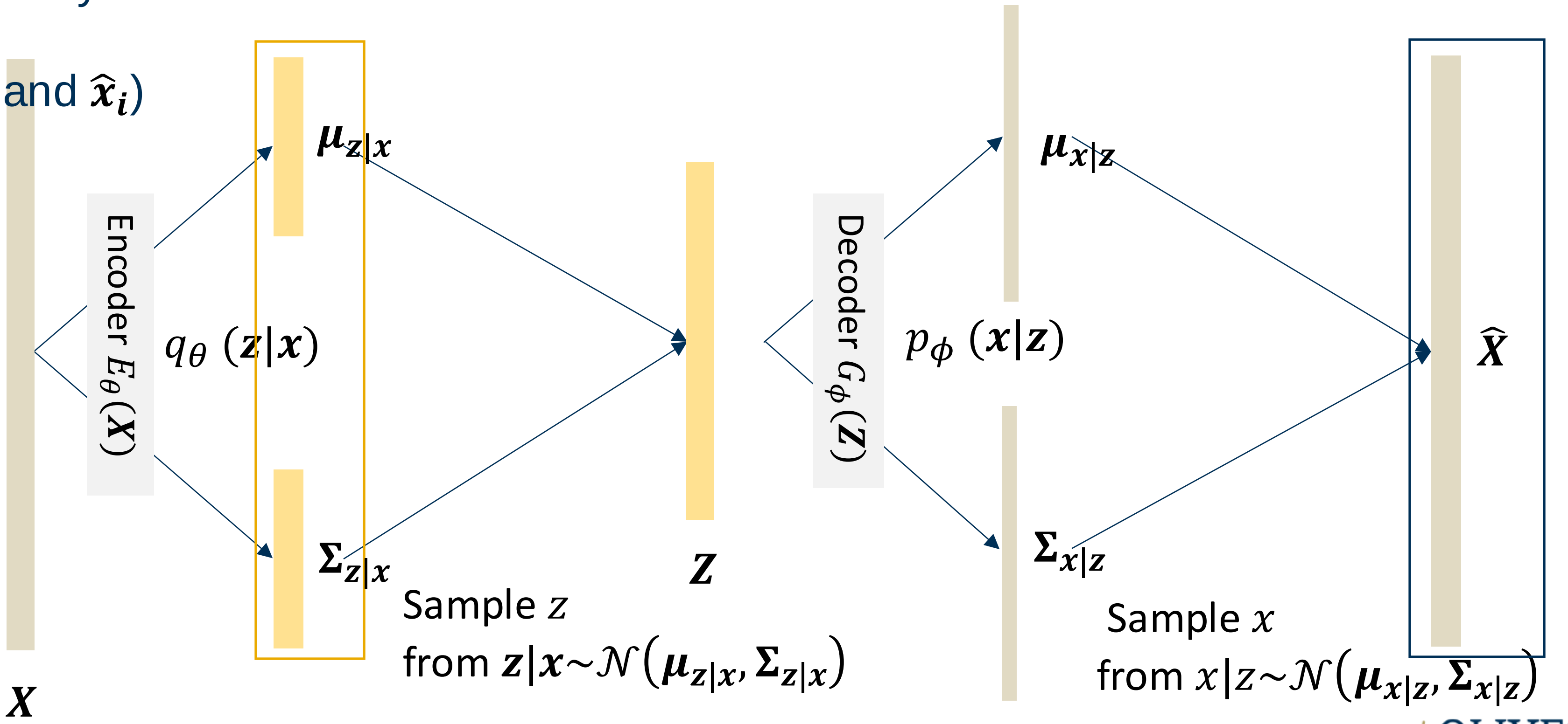
$$\underbrace{\boxed{\mathbf{E}_{\mathbf{z}}\left[\log p_{\phi}(\boldsymbol{x}_i|\mathbf{z})\right]} - D_{KL}\boxed{(q_{\theta}\,(\mathbf{z}|\boldsymbol{x}_i)||p_{\phi}(\mathbf{z}))}}_{L(x_i,\theta,\phi)}$$

Maximize the likelihood of input being reconstructed

(Minimize Binary Cross-entropy between $\boldsymbol{x}_i$ and $\hat{\boldsymbol{x}}_i$)

Make approximate posterior distribution close to prior

(Minimizing KL between $\mathcal{N}(0,\mathbf{I})$ and $\mathcal{N}\left(\boldsymbol{\mu}_{z|x},\boldsymbol{\Sigma}_{z|x}\right)$)



OLIVES @GeorgiaTech

Georgia Tech

Variational Autoencoders
Reparameterization Trick

Backpropagation is not feasible through random sampling

$$\boldsymbol{z}|\boldsymbol{x} \sim \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{z}|\boldsymbol{x}}, \boldsymbol{\Sigma}_{\boldsymbol{z}|\boldsymbol{x}}\right)$$

Cannot back propagate through a random variable Z

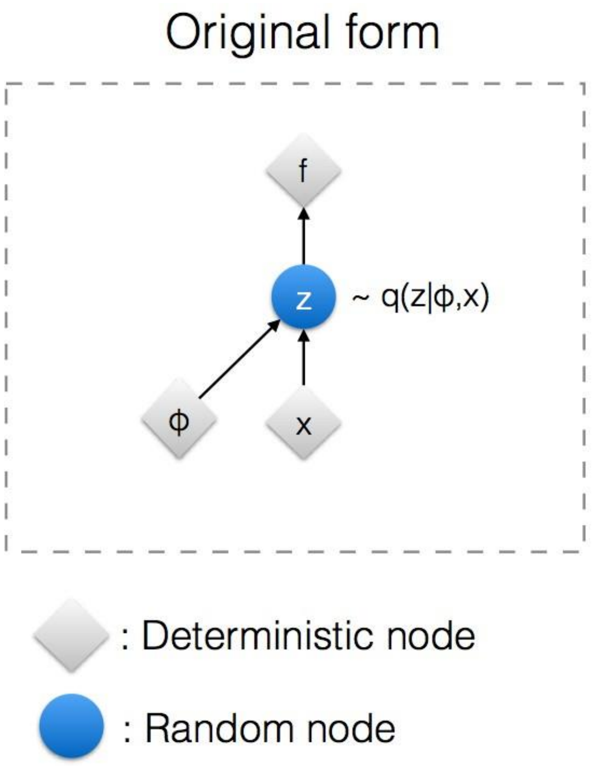


Image from: NIPS Workshop 2015 (Kingma & Welling)

Variational Autoencoders
Reparameterization Trick

Backpropagation is not feasible through random sampling

$\boldsymbol{z}|\boldsymbol{x} \sim \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{z}|\boldsymbol{x}}, \boldsymbol{\Sigma}_{\boldsymbol{z}|\boldsymbol{x}}\right)$

Cannot back propagate
through a random variable Z

$\boldsymbol{z}|\boldsymbol{x} = \boldsymbol{\mu}_{\boldsymbol{z}|\boldsymbol{x}} + \boldsymbol{\Sigma}_{\boldsymbol{z}|\boldsymbol{x}} \odot \varepsilon_i$
where $\varepsilon_i \sim \mathcal{N}(0, \mathbf{I})$

$\odot$: element-wise product

Can back propagate
through a deterministic part $\mu_{z|x}$,

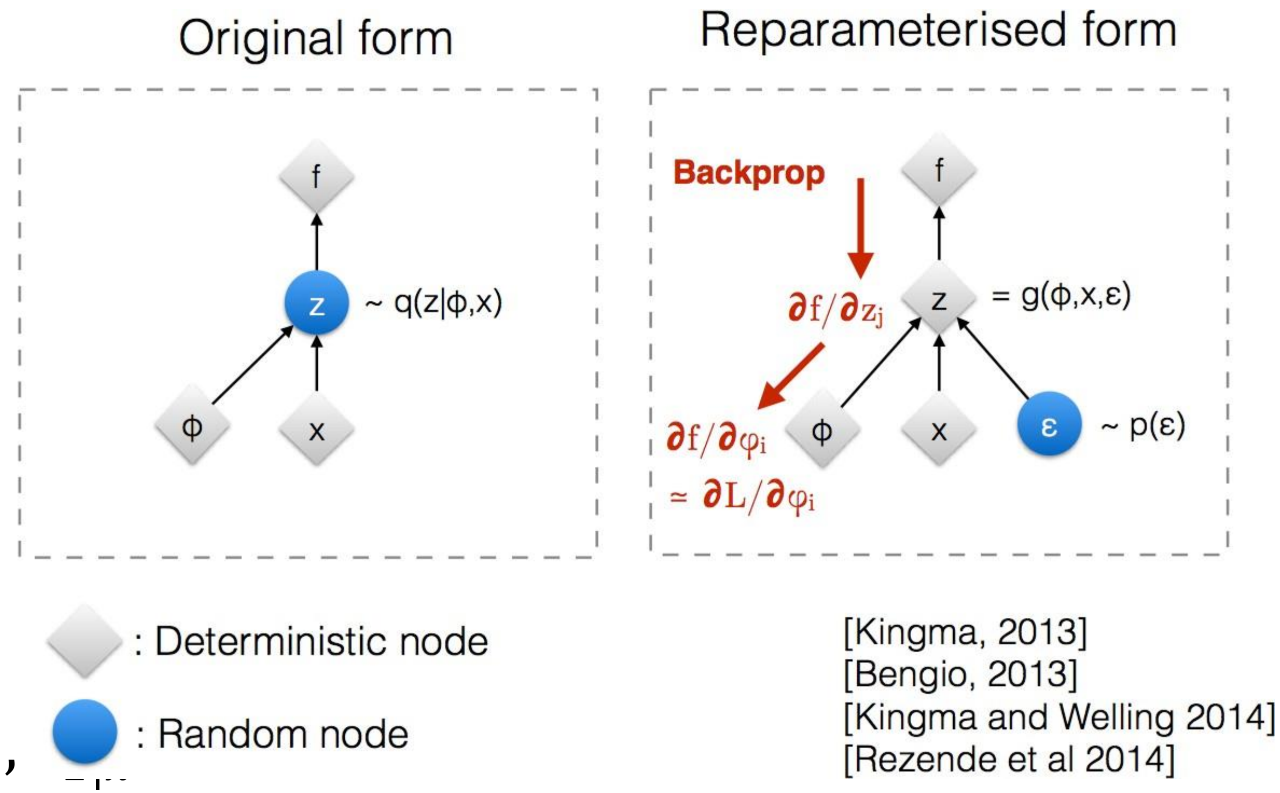
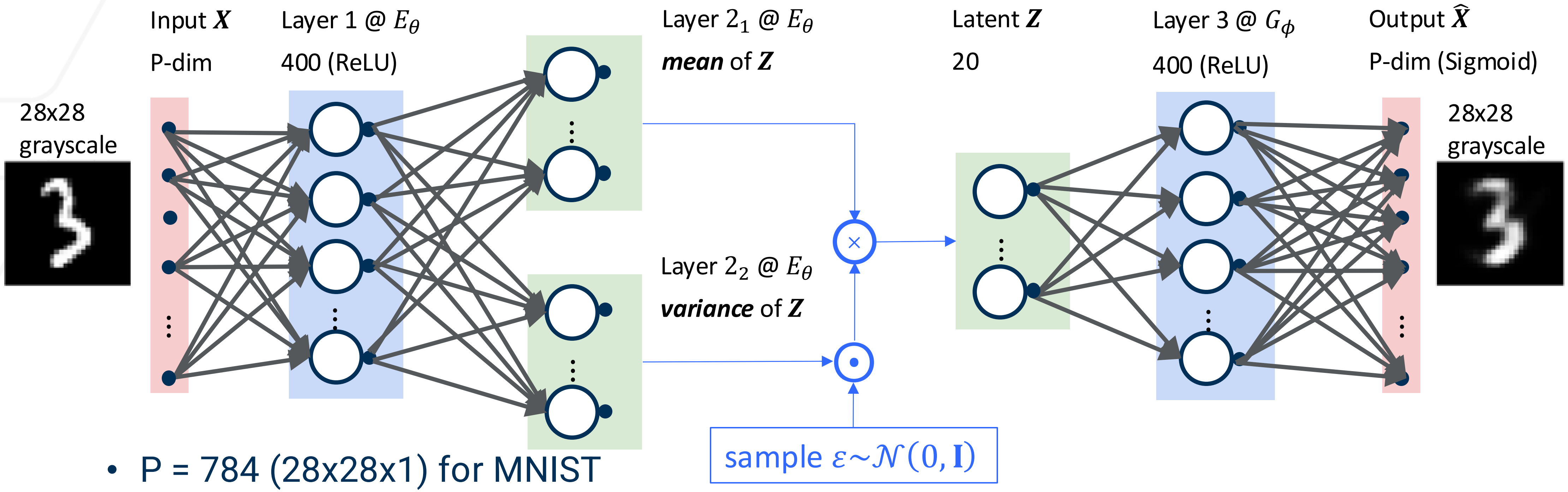


Image from: NIPS Workshop 2015 (Kingma & Welling)

Variational Autoencoders

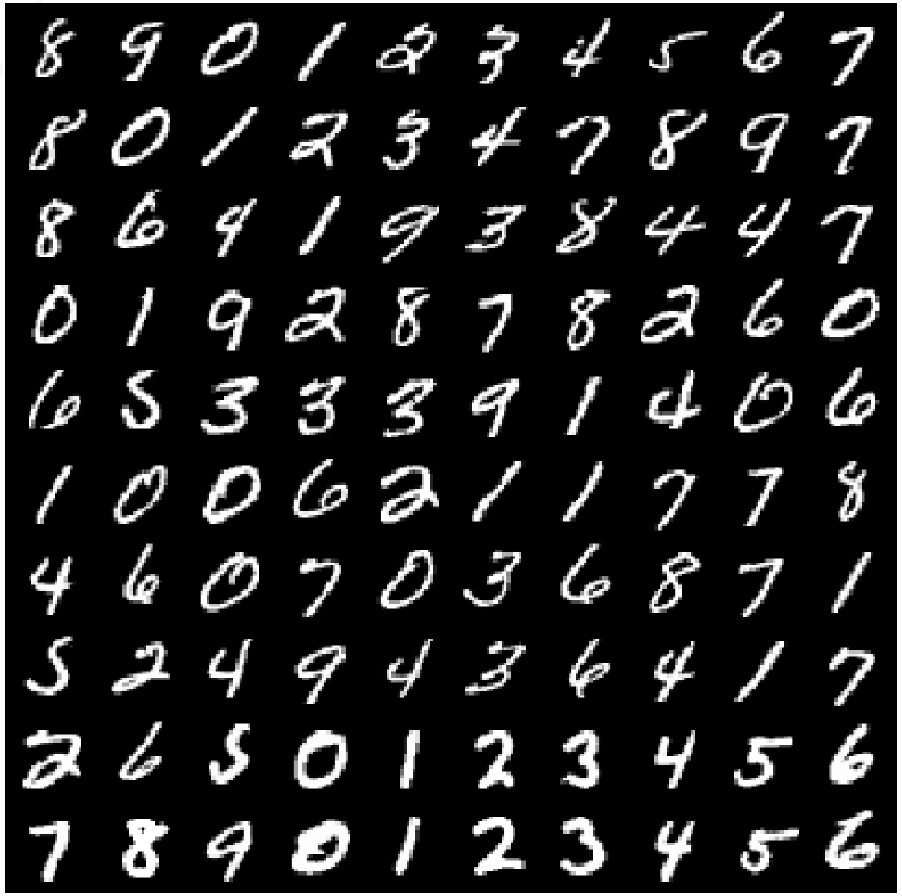
FC-VAE on MNIST



- P = 784 (28x28x1) for MNIST
- ReLU activation in the intermediate layers
- Sigmoid activation in the output layer
  - Input X is normalized to [0,1]

FC-VAE on MNIST

Input MNIST digits



Reconstruction fc-VAE (**latent dim: 20**)

Variational Autoencoders

FC-VAE on MNIST

After training:

- Sample Z from Gaussian prior
- Pass Z through decoder network
- 20-dim latent dimension

These novel samples do not visually resemble training data.
There is no encoder at inference!

Randomly generated novel samples

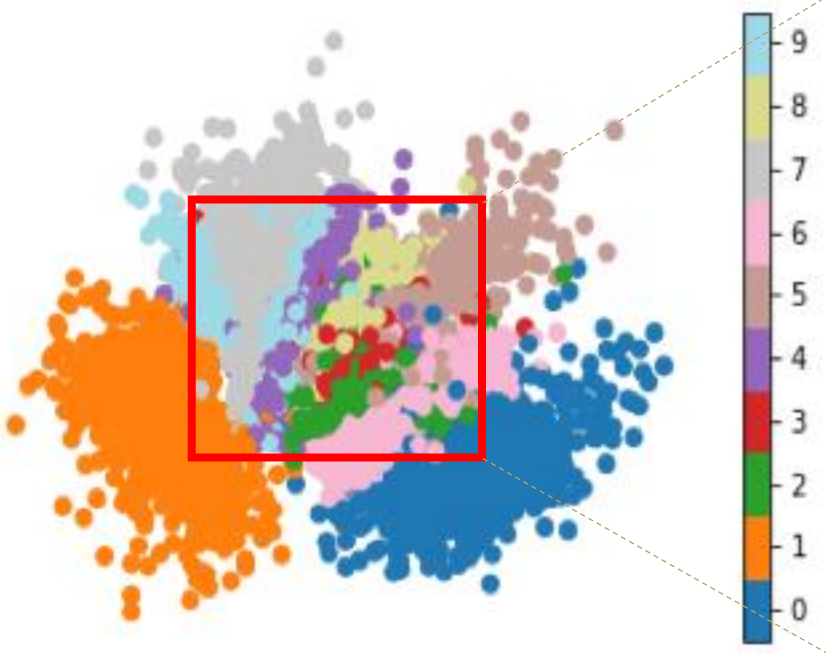
Variational Autoencoders

FC-VAE on MNIST

MNIST Manifold (2D) visualization

Training a 2-dim FC VAE

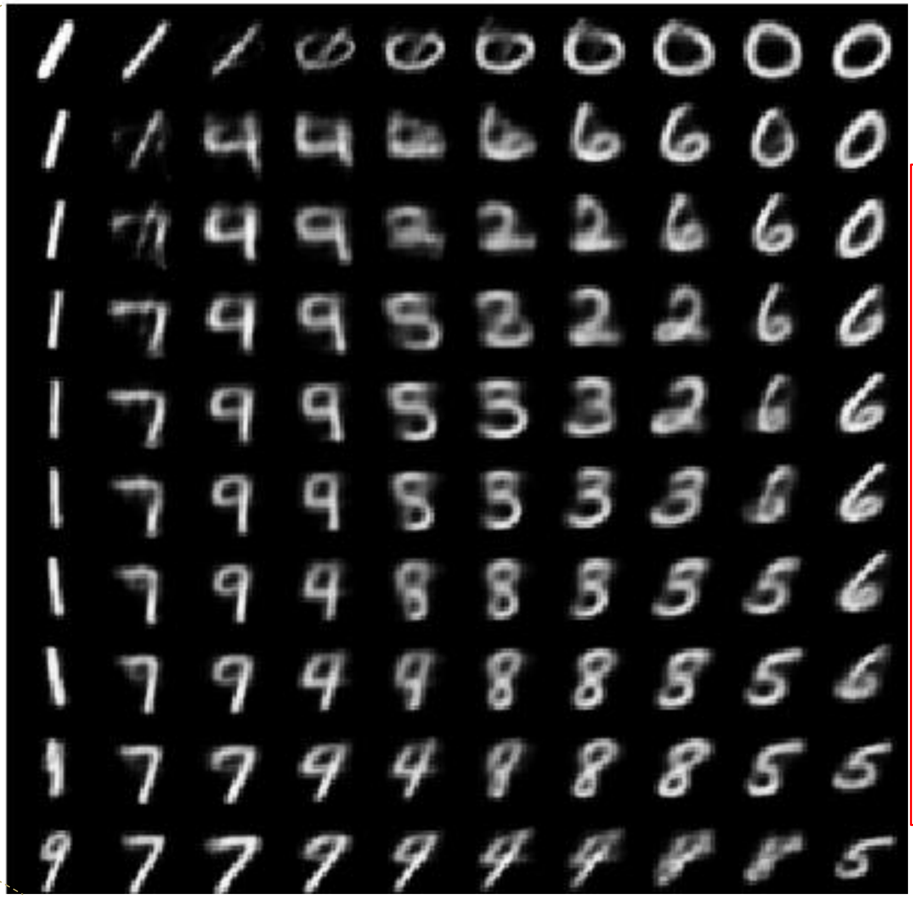
Vary z_1



Vary z_2

MNIST Manifold (2D) visualization

Vary z_1



Vary z_2

Varying a single dimension (z_1 or z_2), it is possible to generate variation of digits

Varying z_2 (left to right): straight digits to digits with curves

Terminology

- *Distribution*: (sample space) the set of all possible samples
- *Dataset*: a set of samples drawn from a distribution
- *Batch*: a subset of samples drawn from the dataset
- *Sample*: a single data object represented as a set of features
- *Feature*: value of a single attribute, property, in a sample. Could be numeric or categorical.

Appendix A: Notations

- x_i: a single feature
- $\boldsymbol{x}_i$: feature vector (a data sample)
- $\boldsymbol{x}_{:,i}$: feature vector of all data samples
- $\boldsymbol{X}$: matrix of feature vectors (dataset)
- $\boldsymbol{W}$: weight matrix
- $\boldsymbol{Z}$: latent representation
- E_θ: encoding function
- G_ϕ: decoding function
- $\widehat{\boldsymbol{X}}$: reconstruction of data
- $\boldsymbol{\Omega}(\boldsymbol{Z})$: sparsity constraint
- $\hat{\rho}_{\boldsymbol{j}}$: average activation of neuron z_{ij}
- $\widetilde{\boldsymbol{X}}$:corrupted input
- N: number of data samples
- P: number of features in a feature vector
- $P^{(k)}$: the number of neurons in layer k
- α: learning rate
- Bold letter/symbol: vector
- Bold capital letters/symbol: matrix

Logistics

- Started going through the Progress reports and came across the following upload!

- Please send me the correct PDF!!