CS181: Computer Vision

Generative Modeling:
Variational Autoencoder (VAE)

November 3rd, 2026

Drake UNIVERSITY

Md Alimoor Reza

Associate Professor of Computer Science

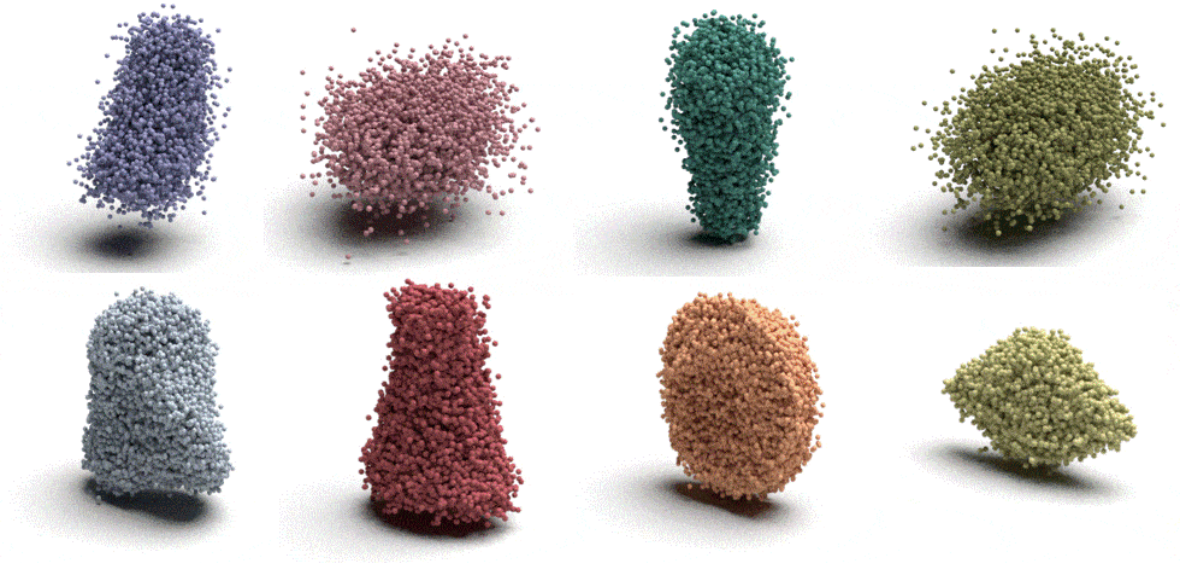
Today's Topics

- **Generative Modeling**
- Latent Space
- Autoencoder (AE)
- Variational Autoencoder (VAE)

Generative Model

- Generative modeling have achieved state-of-the-art performance on many downstream tasks and applications

- Shape generation: generate shapes from 3D point-cloud



Learning Gradient Fields for Shape Generation - ECCV'20

Generative Model

- Generative modeling can be used on many downstream tasks and applications

- Audio synthesis: generate audio that sounds realistic

DiffWave: A Versatile Diffusion Model for Audio Synthesis - ICML'21

- Music generation: generate new music

Symbolic Music Generation with Diffusion Models

Today's Topics

- Generative Modeling
- **Latent Space**
- Autoencoder (AE)
- Variational Autoencoder (VAE)

Generative Model: Latent Space

- A dataset typically exists in a high-dimensional space, but it is often possible to identify a lower-dimensional subspace that captures the essential variation among the samples.

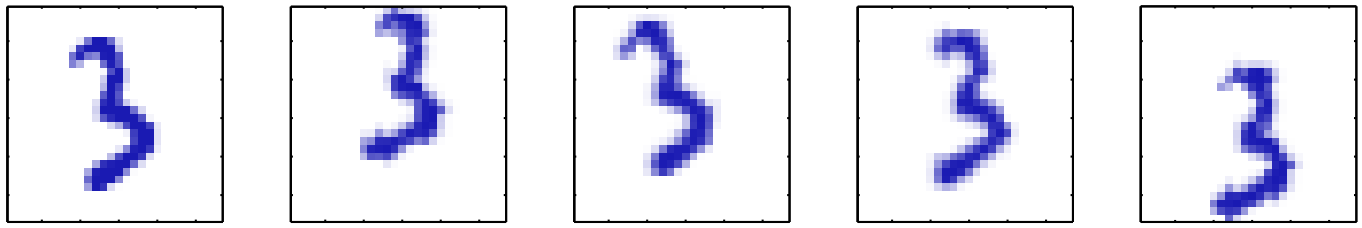
- **Big Idea**: Find new (or *latent* features) made up of combinations of existing features.
  - Maybe multiplying `sepal length*petal width` is more helpful in identifying the species of an iris than either `sepal length` or `petal width` are on their own

Generative Model: Latent Space

- **Claim**: In the 4 *measurable features*, there are really only **2 latent features** which explain these four measurable features. In other words, there are two composite features that more directly probe the underlying phenomenon of the data
    - *house square footage*
    - *number of rooms*
    - *school district test scores*
    - *neighborhood amenities eg, park and recreation, grocery shops, etc*
- Can you see the pattern and guess what these **two latent features** are?
    - Size of house:
        - *house square footage*
        - *number of rooms*
    - Location of house:
        - *school district test scores*
        - *neighborhood amenities eg, park and recreation, grocery shops, etc*

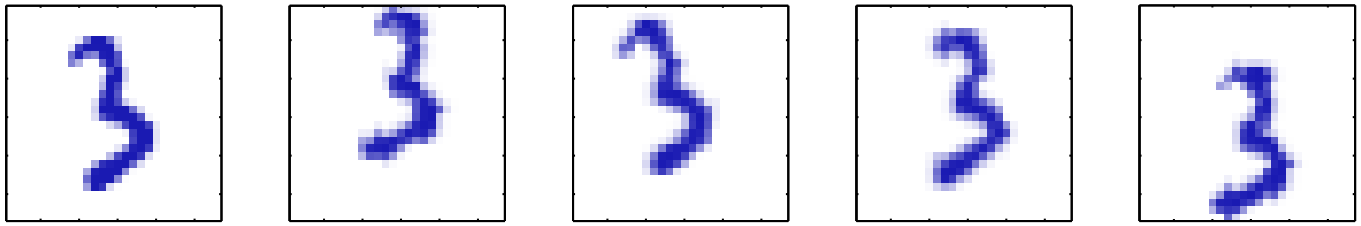
Generative Model: Latent Space

- Let's consider a synthetic dataset of digits, which is a collection of 28x28 pixel images. Each image represents one of 10 possible digits. For example, look at the 5 samples from this dataset.



The **location** (x, y) and **orientation** (θ) of the main content (digit three) are varied randomly to create the five different data points. Each data point lies in 784 dimensional pixel space.

Generative Model: Latent Space



- Images are represented in a 28*28 =784 dimensional pixel space
  - Most of the pixels are not useful (eg, a lot of white region in each image with no information)
  - Hence, most of the dimensions are redundant (ie, data points lie close to a manifold of much lower dimensionality than that of the original data space)
  - Across the set of data points, there are only three degrees of variability (ie, corresponding to vertical and horizontal translations (x, y) and one rotation θ with respect to the horizontal axis)
- Such images can be represented in a much smaller dimensions
  - significantly less than **784** dimensional pixel space (ie, the data points will therefore live on a subspace of the data space whose intrinsic dimensionality is **3**)
- In this example, the translation and rotation parameters are latent variables (ie, we only observe the image vectors and are not told which values of the translation and rotation variables were used to create them)

Generative Model: Latent Space

- In practice, the data points will not be confined to a smooth low-dimensional manifold and this departure can be interpreted as **'noise'**

- It give us a new perspective that there is data generation process by a underlying model and we are just observing the generated images

- First, select a point within the manifold according to some latent-variable distribution:

  Some latent variables: z

  Sampling from that latent space: $p(z) = \mathcal{N}(z \mid 0, I)$

- Then generate an observed data point by adding noise drawn from some conditional distribution of the data variables given latent variables:

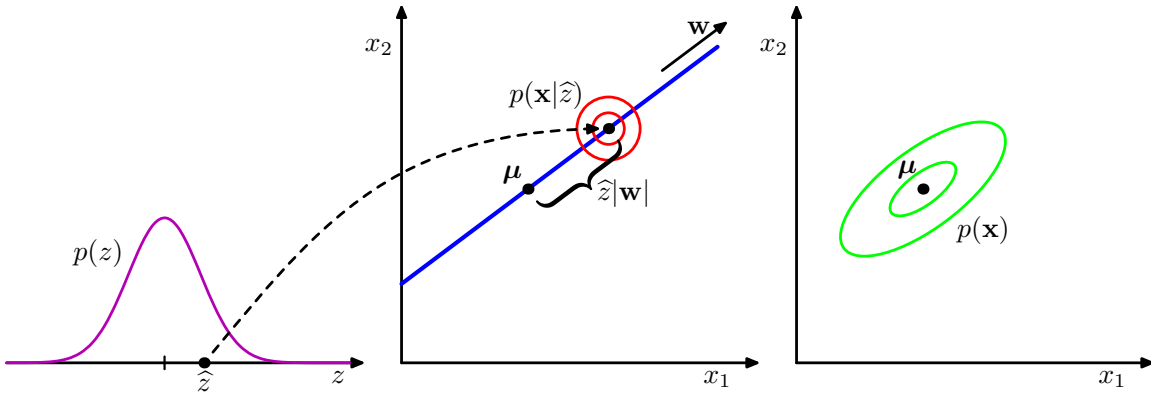
  Transformation based on latent variables and noise: $x = Wz + \mu + \epsilon$

  Sampling from that conditional distribution: $p(x \mid z) = \mathcal{N}(x \mid Wz + \mu, \sigma^2 I)$

Generative Model: Latent Space

Take an isotropic Gaussian distribution 'spray can' and move it across the principal subspace spraying Gaussian ink with density determined by sigma and weighted by prior distribution p(z). The accumulated ink density gives rise to `Pancake' shaped distribution representing marginal distribution p(x).

Gaussian distribution | Gaussian distribution | Gaussian distribution



Sampling from that latent space:

$$p(z) = \mathcal{N}(z \mid 0, I)$$

Sampling from the conditional distribution:

$$p(x \mid z) = \mathcal{N}(x \mid Wz + \mu, \sigma^2 I)$$

The marginal distribution:

$$p(x) = \int p(x \mid z) p(z)$$

- First, select a point within the manifold according to some latent-variable distribution (eg, in probabilistic PCA, we sample a point from marginal distribution P(z) where **z** correspond to the principal-component subspace. P(z) is a Gaussian distribution)
- Then generate an observed data point by adding noise drawn from some conditional distribution of the data variables given latent variables (eg, the observed point **x** is generated by drawing from an isotropic Gaussian distribution having a mean **wz + μ** and covariance $\sigma^2 \boldsymbol{I}$)

Generative Model

- Generative modeling is based on representation of probability distribution
- Neural networks are universal function approximators, hence they are also good at approximating any desired probability distribution with high accuracy
- Generative models use machine learning algorithms to learn a probability distribution from a set of training data and then generate new examples from that distribution
  - Think of such a generative model in terms of a distribution $p(x|w)$ in which x is a vector in the data space, and w represent the learnable parameters of the neural network model
  - In many cases we are interested in conditional generative models of the form $p(x|c, w)$ where c represents a vector of conditioning variables
  - For example, a generative model trained on animal images and it's used to generate new image of an animal. We could specify by variable c which animal image we want to generate

Generative Model

- Generative modeling is based on representation of probability distribution
- Neural networks are universal function approximators, hence they are also good at approximating any desired probability distribution with high accuracy
- **Main idea of modeling:**
  - Select a point in a low-dimensional space according to some latent-variable distribution
  - Then generate an observed data point by adding noise drawn from some conditional distribution of the data variables given the latent variables
  - For example, probabilistic Principle Component Analysis (PCA) is a generative model where both latent distribution and conditional distribution both are Gaussian distributions

Types of Generative Models

- Existing generative modeling techniques can be classified into four broad categories
  - Generative adversarial networks GAN (implicit generative models)
  - Variational auto-encoders VAE (likelihood-based models)
  - Diffusion models (score-based models)
  - Normalizing flows

Generative Model: Likelihood-Based Model

- **Variational Autoencoder (VAE):**
  - VAE is an example of Likelihood-based generative model
  - directly learns the distribution's probability density function via maximum likelihood estimation method

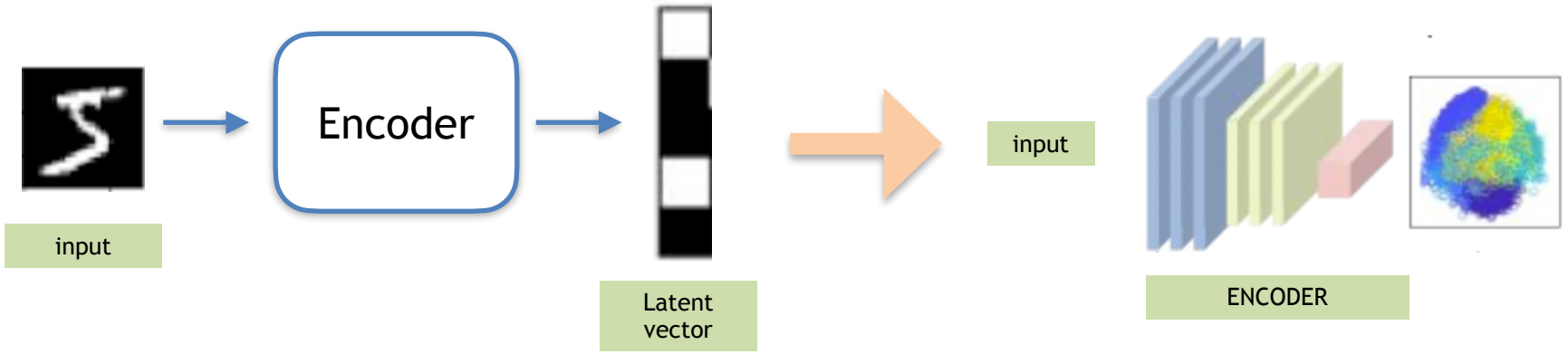
  - Limitations
    - rely on surrogate objectives to approximate maximum likelihood training
    - require strong restriction on the model architecture for tractable normalization

Today's Topics

- Generative Modeling
- Latent Space
- **Autoencoder (AE)**
- Variational Autoencoder (VAE)

Encoder for Learning Latent Representation

- A neural network (based on either MLP or CNN) to generate the latent vector



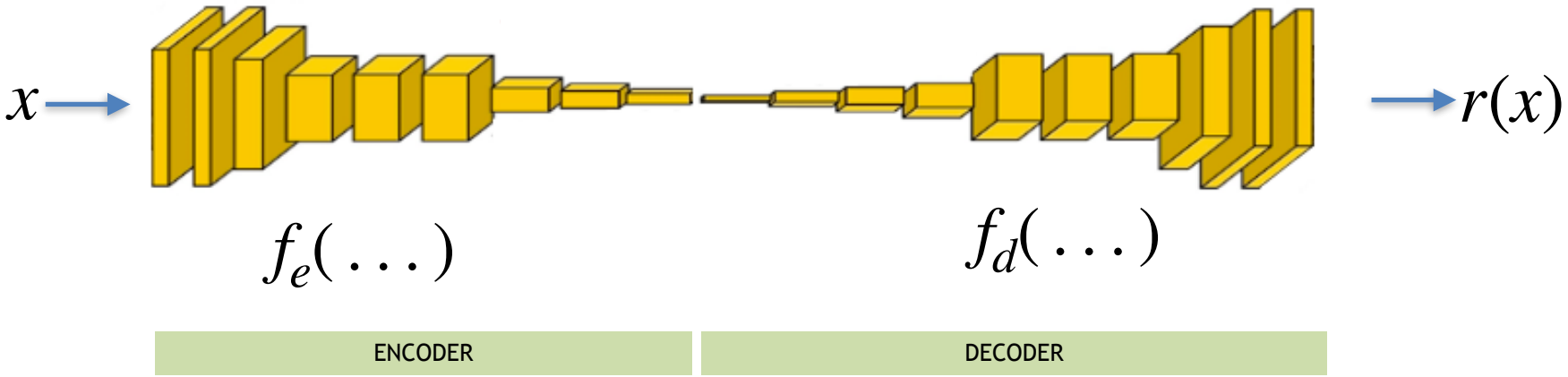
- Why do need compress the image?
  - Most of the dimensions are redundant
  - Efficient representation if we can compress it down to 10, 20, 100
  - Useful for followup task such as classification, decoding, etc

- These compressed latent representation can be learned with the help of another decoder sub-network jointly. Collectively, this network can be called Autoencoder.

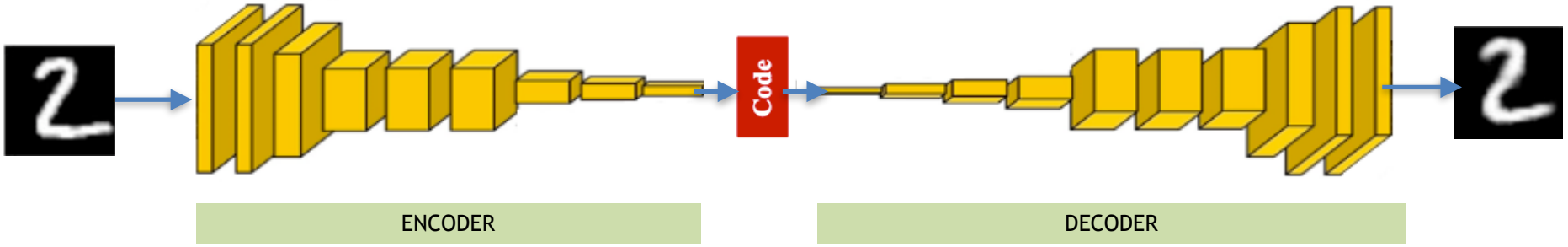
Learning Latent Representation via Autoencoder

- An Autoencoder is a type of neural network consisting of an Encoder and a Decoder that jointly learns the non-linear mapping between input ***x*** and output ***r(x)***

$$r(x) = f_d(f_e(x))$$



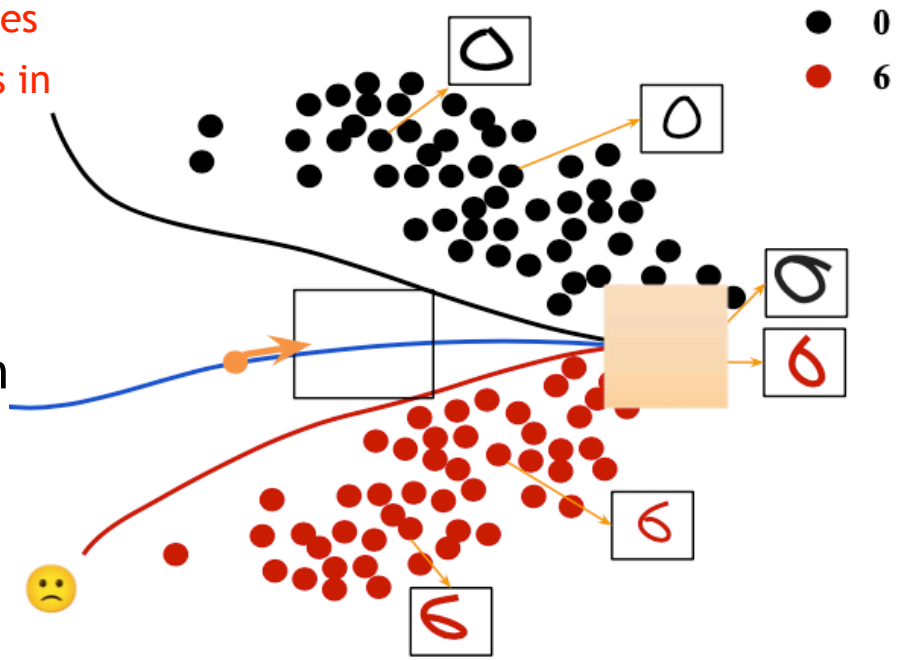
Vanilla Autoencoder



- Autoencoder input image is coded and decoded in one go
- After training similar examples share similar latent vector or code
- Since the network weights are fixed; after training if we input the same image we will always get the same latent vector
  - Can not generate images using unseen code
  - Latent space is discrete and not continuous

Vanilla Autoencoder

- Autoencoder do not have control over latent vectors (LV) or code
  - Only generate latent variable for seen images
  - Autoencoder is trained using discrete points in the latent space



- What will the model generate for a sample in the latent space in between two classes (eg, 0 and 6)?

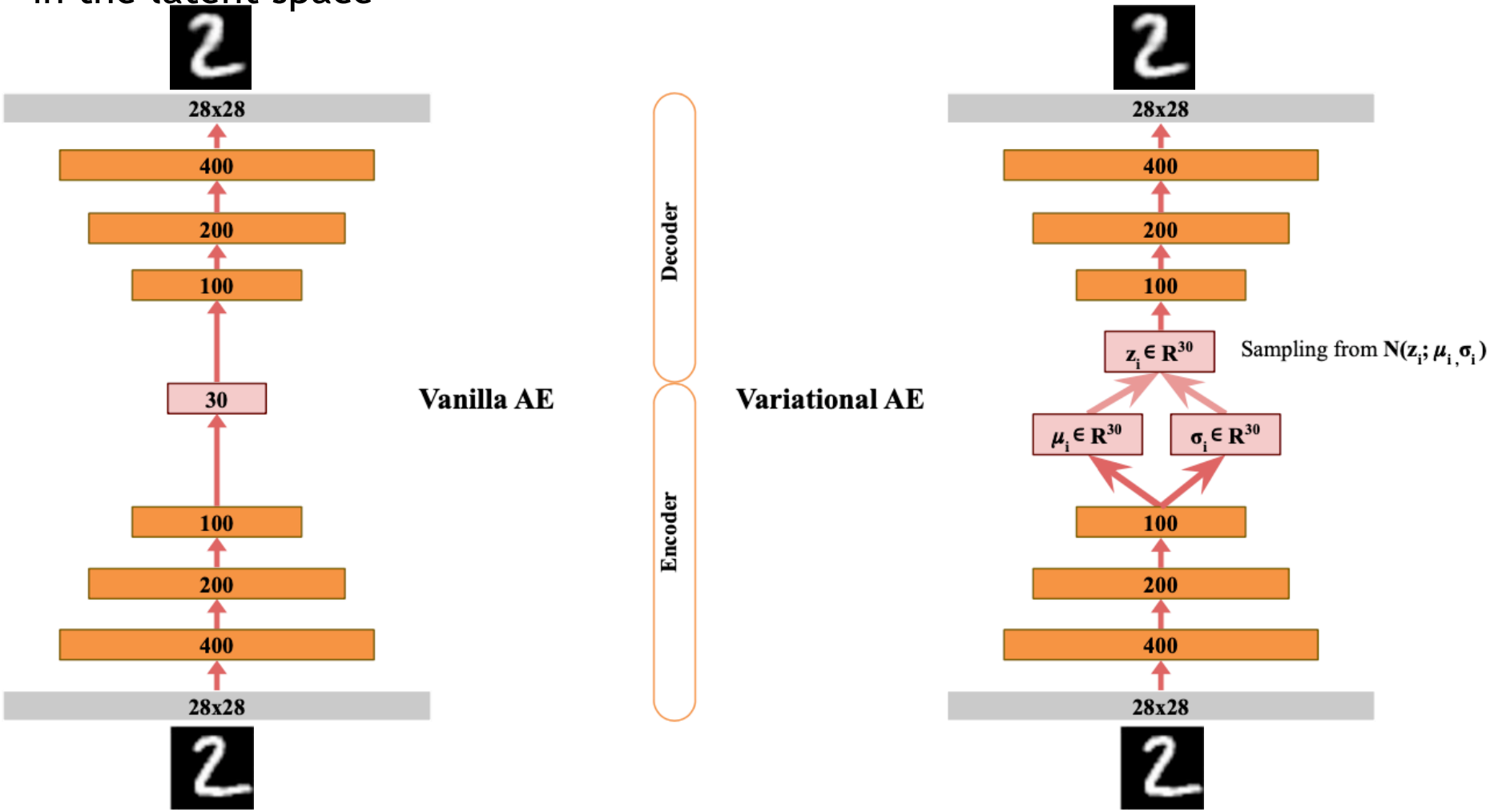
- Need to make generation process smooth or continuous in the latent space
- Variational Autoencoder (VAE) can achieve this goal

Today's Topics

- Generative Modeling
- Latent Space
- Autoencoder (AE)
- **Variational Autoencoder (VAE)**

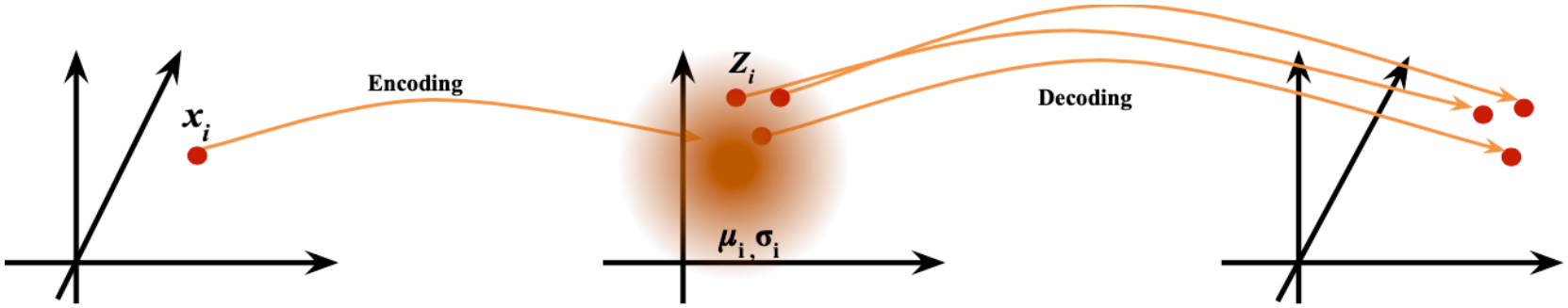
Variational Autoencoder (VAE)

- Variational Autoencoder (VAE) try to sample from a continuous distribution in the latent space



Variational Autoencoder (VAE)

Sampling enables to learn from multiple samples of the latent distribution



- From one example, we got a distribution with all nearby variations → making latent space continuous

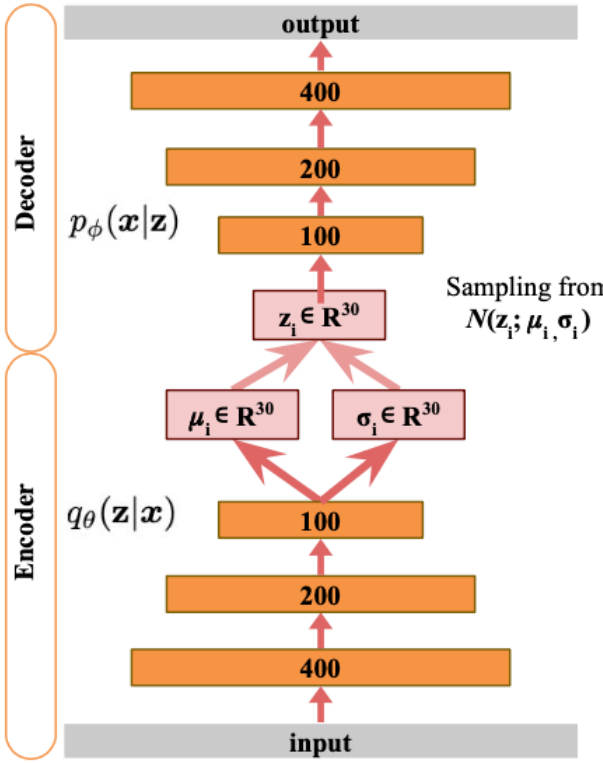
- During training
  - The reconstructed output is from a sampled version of the latent variable
  - The training algorithm is exposed to the continuous distribution of the latent space distribution (which was not possible with Vanilla Autoencoder)

- During testing
  - it produces stochastic predictions (randomly generate new output) due to the sampling process

Training Loss Function of VAE

- Encoder
  - posterior distribution over latent variable given the data samples
  - parameterized by the weights of the network

- Decoder
  - distribution over the multivariate observations (pixels) given the latent variable
  - parameterized by the weights of the network

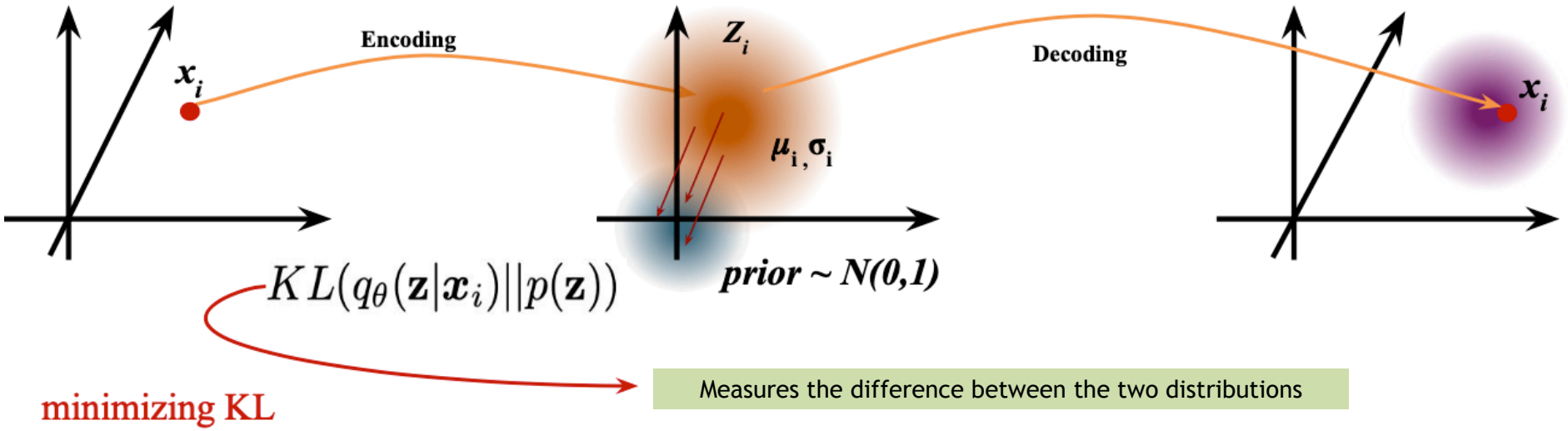


- Loss function is an expectation function over latent variable

$$\mathcal{L}_i(\theta, \phi) = -\mathbb{E}_{\mathbf{z} \sim q_\theta(\mathbf{z}|\boldsymbol{x}_i)}[\log p_\phi(\boldsymbol{x}_i|\mathbf{z})] + KL(q_\theta(\mathbf{z}|\boldsymbol{x}_i)||p(\mathbf{z}))$$

- Two terms inside the expectation function
  - Likelihood term: likelihood of observing the input given the latent variable
  - Regularization term: the latent variable should be close to a prior Gaussian distribution

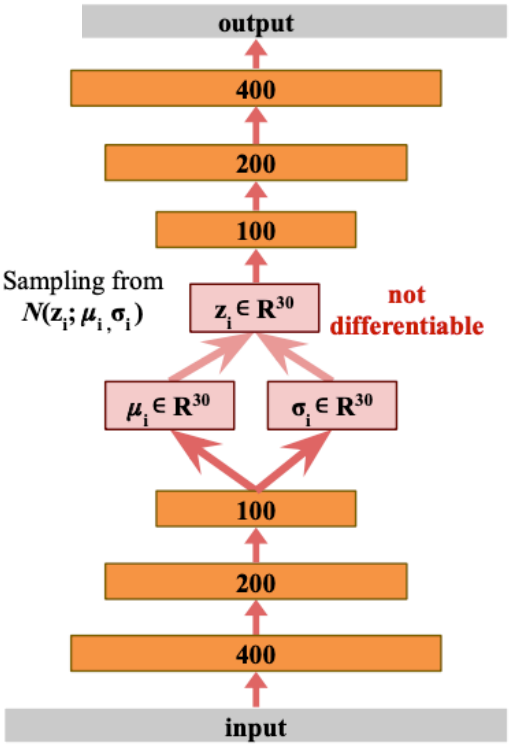
Role of the Prior in VAE

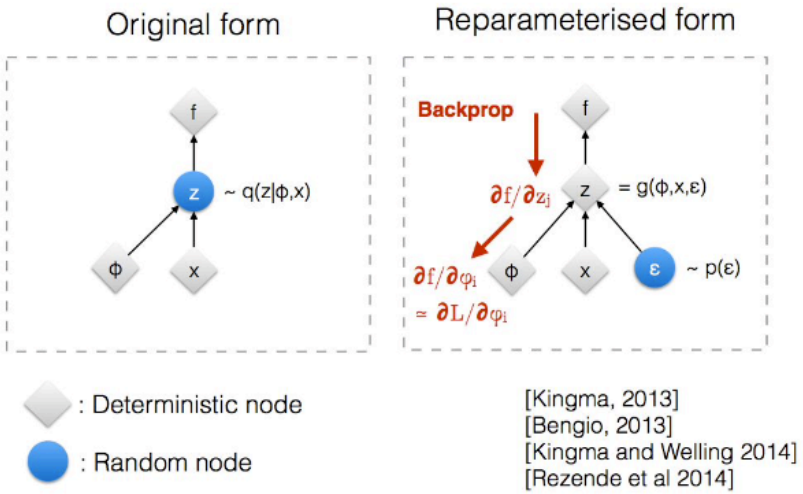


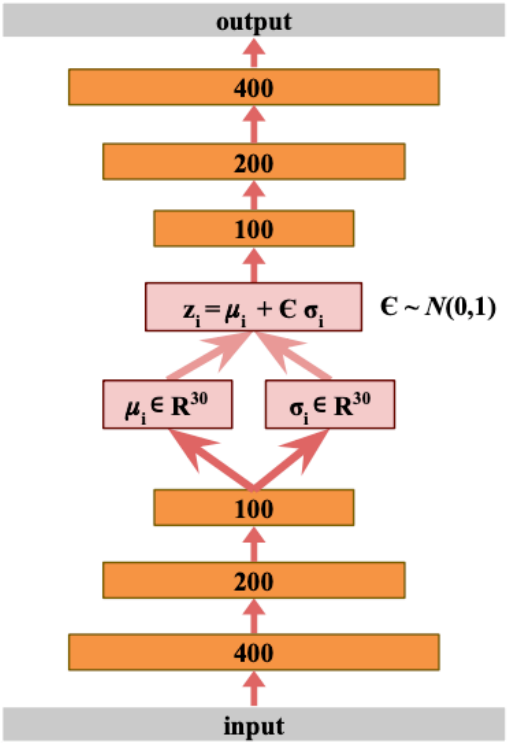
- We want latent variable distribution to look like a standard normal distribution
  - it will provide control over the latent space distribution

- Apply a KL divergence loss function between latent-variable distribution and standard Gaussian prior distribution
  - at each step, the prior will nudge the latent variable distribution towards standard Gaussian

Reparameterization Trick for Backpropagation



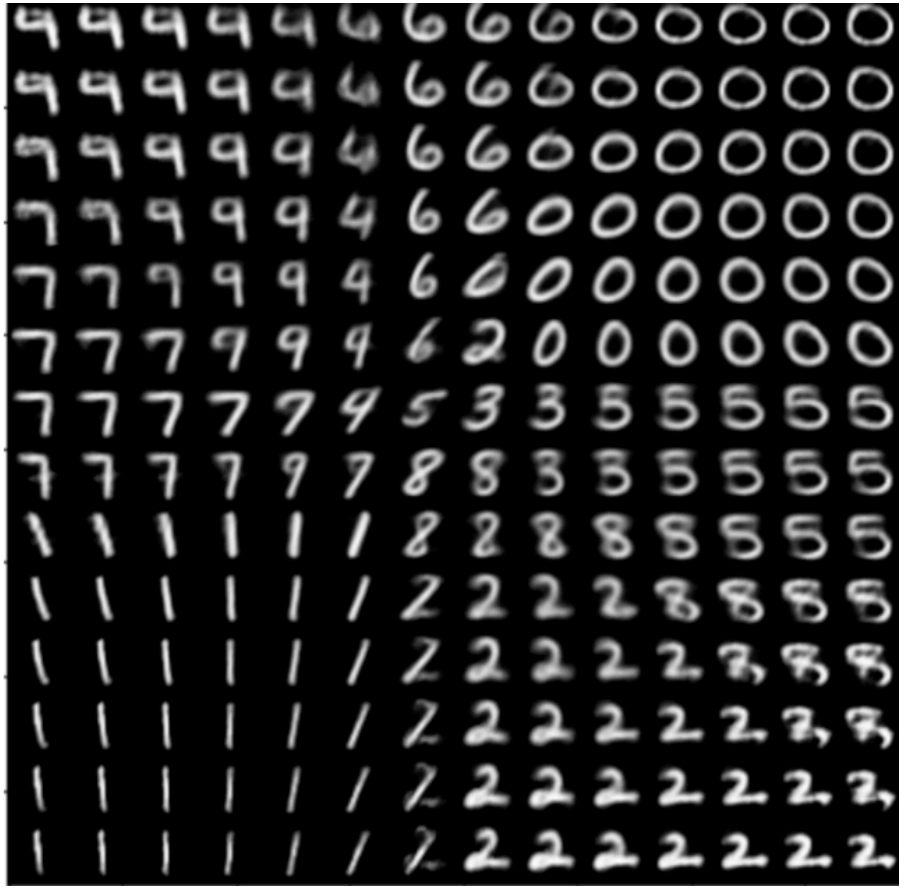




- After reparameterization, Z_i is differentiable with respect to both its parameters
- Why multiplying with a normal distribution?
  - adds randomness around the mean value of Z_i for continuous variations

VAE Image Generation

- Images with continuous variations are generated with VAE
  - Some digits looks like perfect fusion of two digits
  - Many digits do not make any sense but latent space is continuous



https://github.com/alimoorreza/CS181-fall26-notes/blob/main/day20_generative_model_VAE.ipynb