Generative Learning

AutoEncoders and Variational AutoEncoders

N. Rich Nguyen, PhD

SYS 6016

Which face is fake?

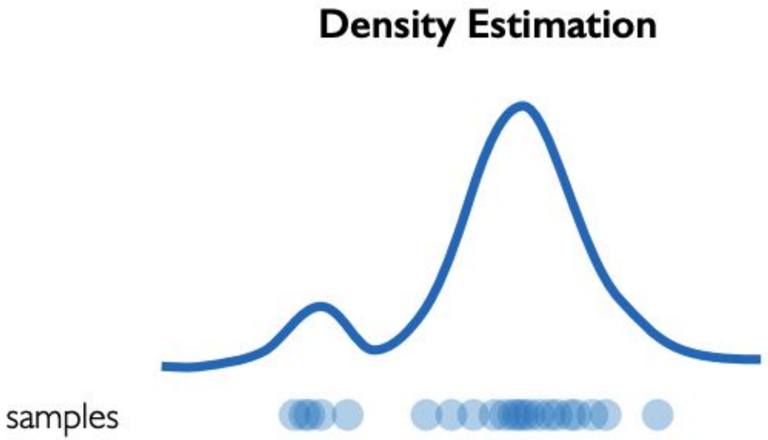

A

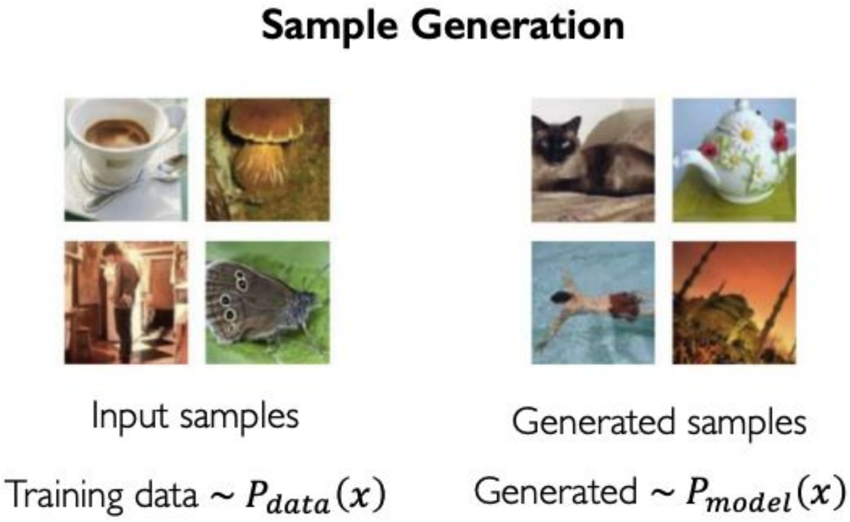

B


C

Generative Modeling

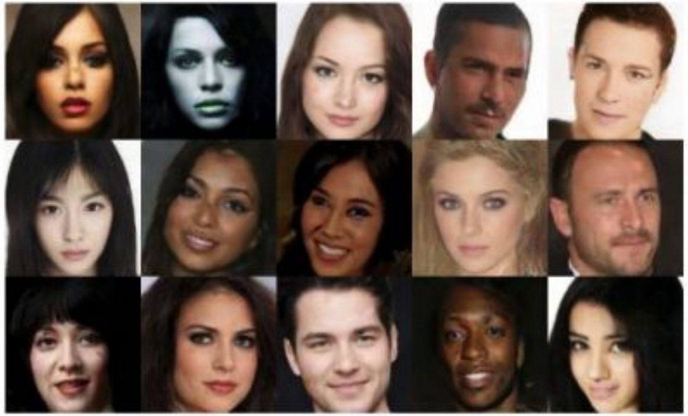
Goal: It takes as input training samples from some distribution and learn a model that represents that distribution (unsupervised learning)





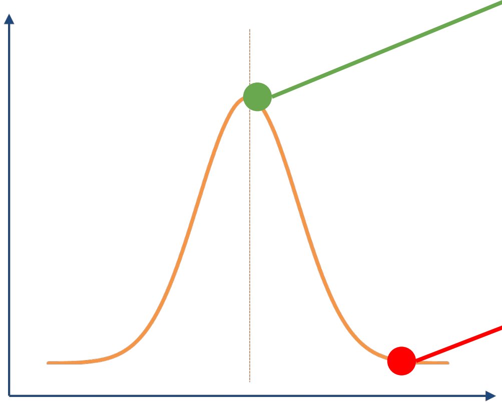
Why generative models? Underlying Variables

- Generative models of training data can be used for *simulation* and *planning*
- *Produce **realistic*** samples for artwork, super-resolution, colorization, ect.
- Training generative models can also enable *manipulation* of latent representations that can be useful as general features.


Why generative models? Outlier detection

- **Problem**: How can we detect when we encounter something new or rare?
- **Strategy**: Leverage generative models, detect outliers in the distribution. Use outliers during training to improve even more!



Detect outliers to avoid unpredictable behavior when training

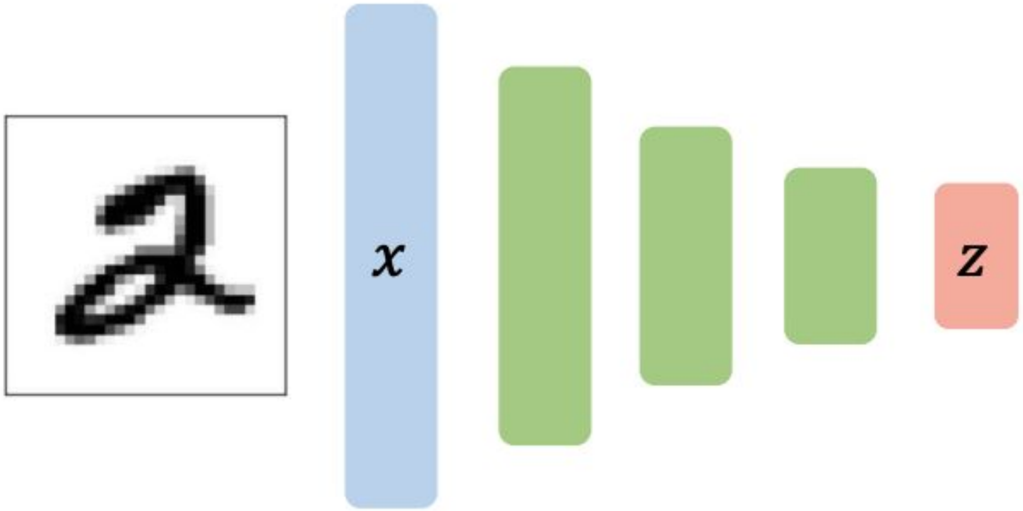
3 most popular types of generative models

1. AutoEncoders
2. Variational AutoEncoders (VAEs)
3. Generative Adversarial Networks (GANs)

1. AutoEncoders

Autoencoder Definition

It's an *unsupervised* approach for learning a lower-dimensional (dense) feature representation, called ***latent space***, from *unlabeled data*



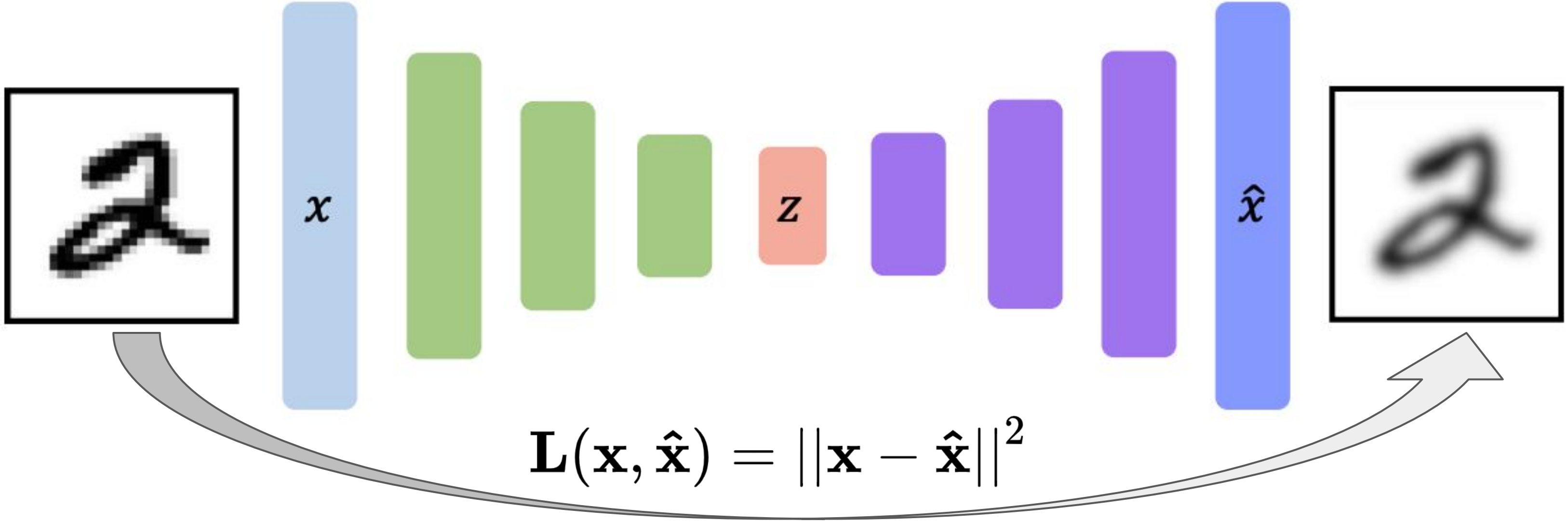
Why do we care about a low-dimensional z?

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

Reconstruct the original data

How can we learn this latent space, **z**?

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

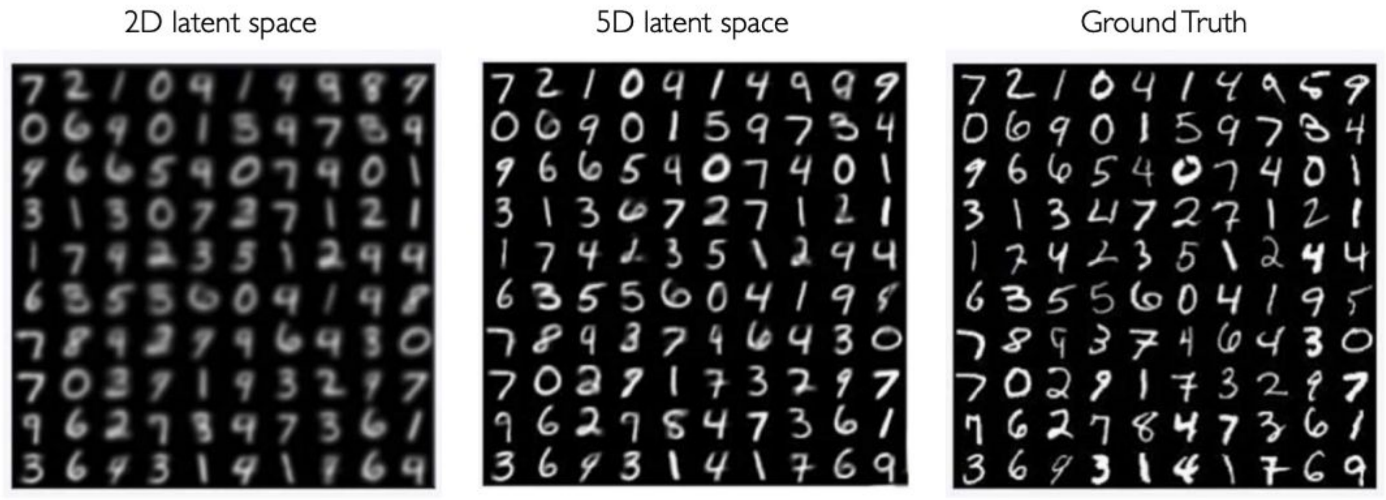


$$\mathbf{L}(\mathbf{x}, \hat{\mathbf{x}}) = ||\mathbf{x} - \hat{\mathbf{x}}||^2$$

"Decoder" learns mapping from the latent, **z**, to a reconstructed observation, **x^**

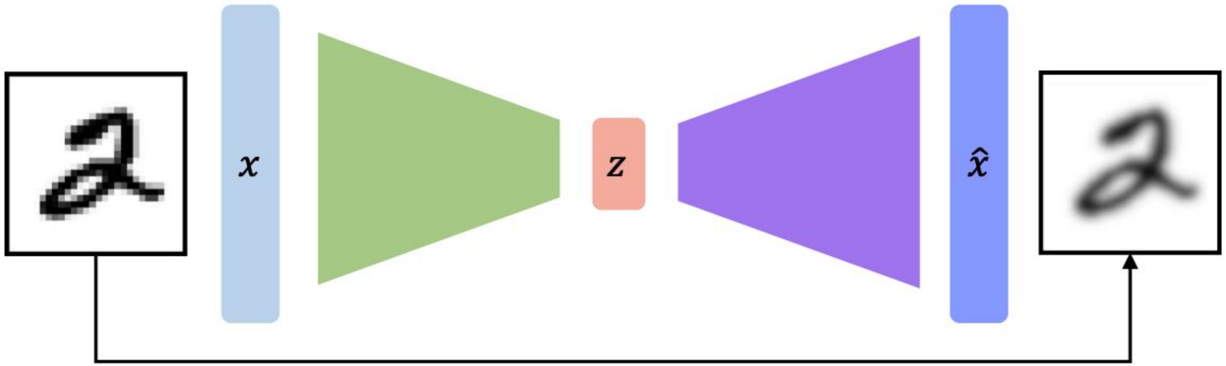
Dimensionality of latent space

- The dimensionality of the latent space determines the **quality** of the reconstruction
- Encoding is a form of **compression**!
- A smaller training **bottleneck layer z** will force a smaller latent space



Autoencoders for representation learning

- **Bottleneck hidden layer z** forces network to learn a compressed latent representation
- **Reconstruction loss L(x,x^)** forces the latent representation to capture (or encode) as much information about the data as possible
- **Autoencoding** = **Auto**matically **encoding** data

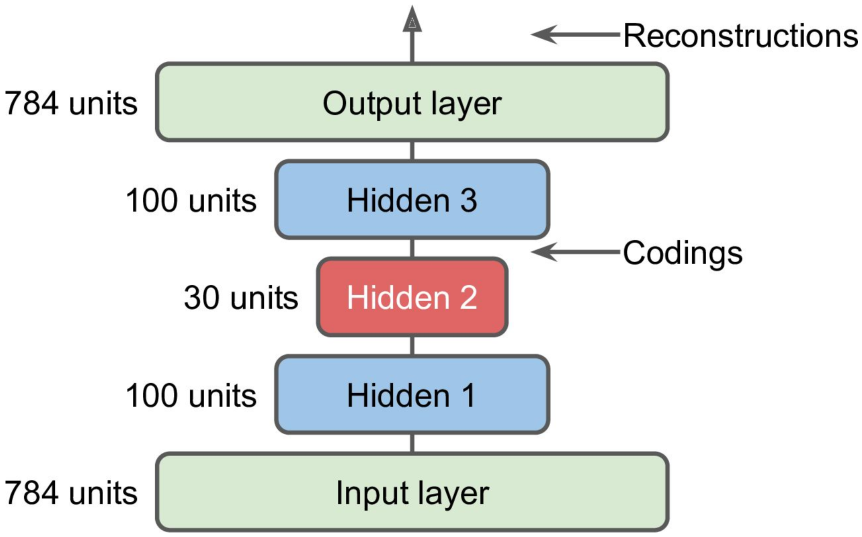


$$\mathcal{L}(x, \hat{x}) = ||x - \hat{x}||^2$$

Loss function doesn't use any labels!!

Stacked AutoEncoders

- The architecture of a stacked autoencoder is typically **symmetrical** with regard to the bottleneck layer.
- Adding more layers helps autoencoder learn more complex codings.
- The right figure is a autoencoder for Fashion MNIST (with 28x28=784 features)



Implementation

```
stacked_encoder = keras.models.Sequential([
    keras.layers.Flatten(input_shape=[28, 28]),
    keras.layers.Dense(100, activation="selu"),
    keras.layers.Dense(30, activation="selu"),
])
stacked_decoder = keras.models.Sequential([
    keras.layers.Dense(100, activation="selu", input_shape=[30]),
    keras.layers.Dense(28 * 28, activation="sigmoid"),
    keras.layers.Reshape([28, 28])
])
stacked_ae = keras.models.Sequential([stacked_encoder, stacked_decoder])
stacked_ae.compile(loss="binary_crossentropy",
                   optimizer=keras.optimizers.SGD(lr=1.5))
history = stacked_ae.fit(X_train, X_train, epochs=10,
                         validation_data=[X_valid, X_valid])
```

MNIST images and its reconstruction

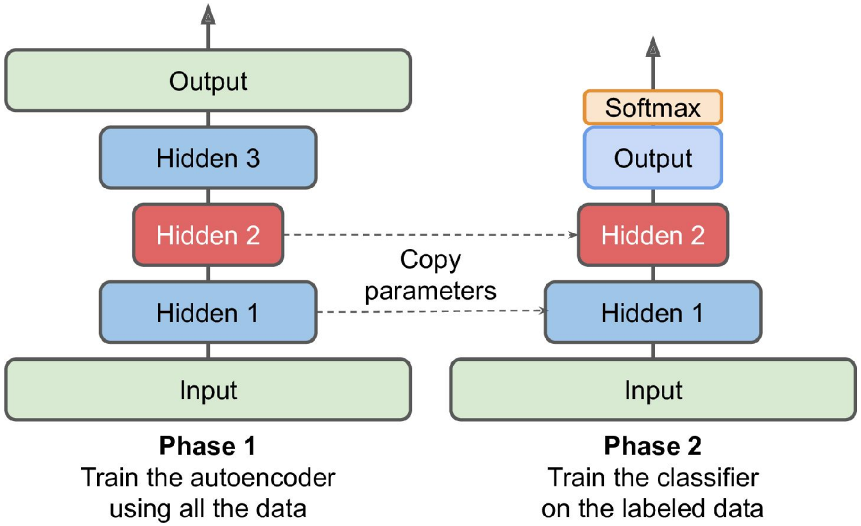
Input Images

Constructed Images

The reconstructions are recognizable, not a bit too lossy.

Should we train the model for longer, or make the encoder deeper?

Unsupervised Pre-training



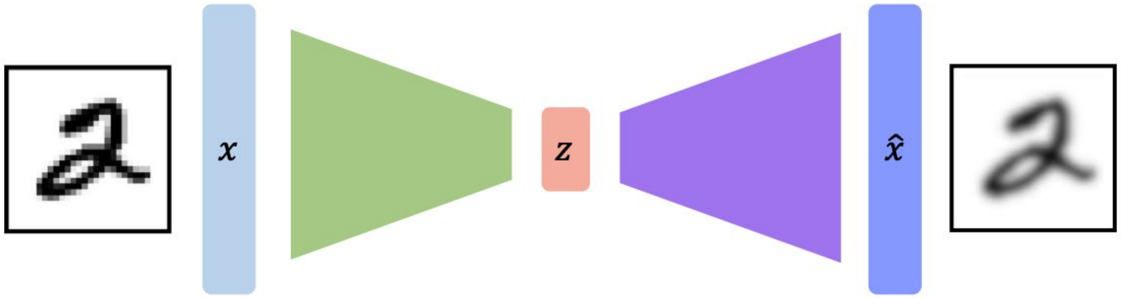
Tackling complex supervised task with little labeled training data, but large number of unlabeled data:

1. Train a stacked autoencoder using the unlabeled data
2. Freeze the lower layers and reused them to train the labeled data

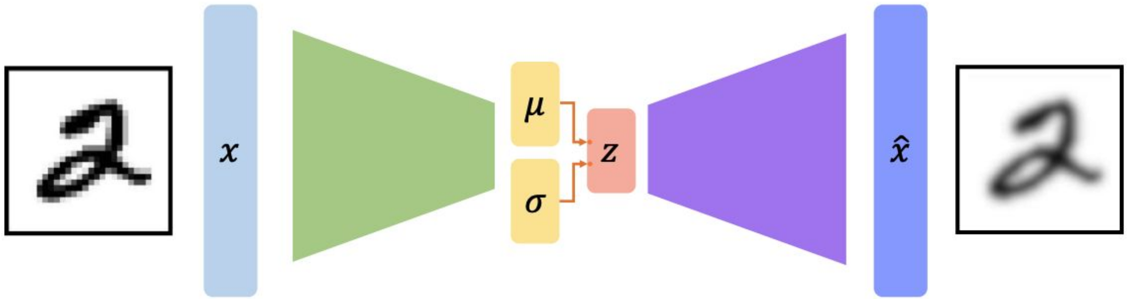
2. Variational AutoEncoders (VAEs)

VAEs: key difference

AutoEncoder



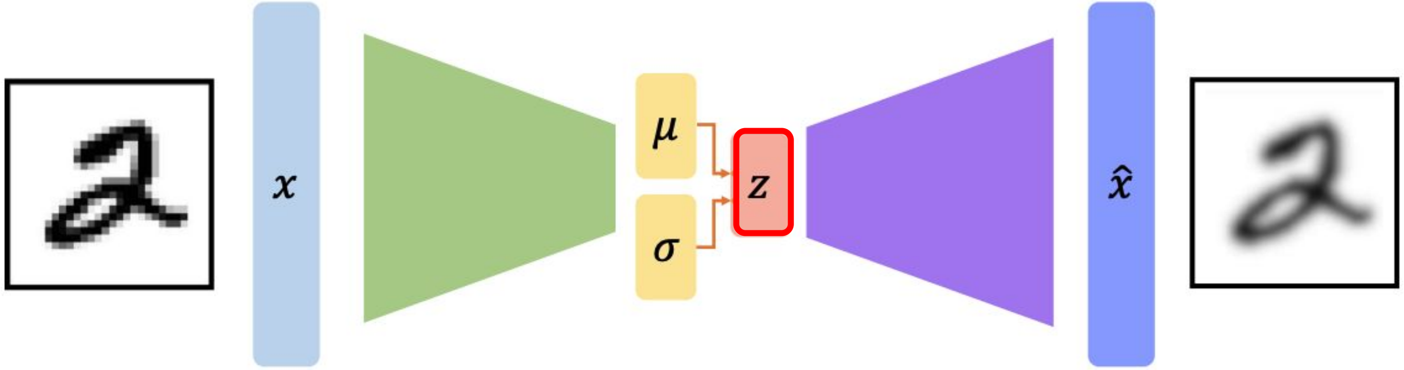
VAE



- Variational AutoEncoders are a **probabilistic twist** on autoencoders!
- Sample from the *mean* and *standard deviation* vectors to compute latent space

VAE Optimization

Problem: We cannot backpropagate gradients through the sampling layer!



Encoder computes: $p_\phi(\mathbf{z}|\mathbf{x})$ Decoder computes: $p_\theta(\mathbf{x}|\mathbf{z})$

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

$$||\mathbf{x} - \hat{\mathbf{x}}||^2 \qquad \mathbf{D}\big(p_\phi(\mathbf{z}|\mathbf{x})\,||\,p(\mathbf{z})\big)$$

Inferred latent distr.

Fixed prior (Std. Normal) on latent distr.

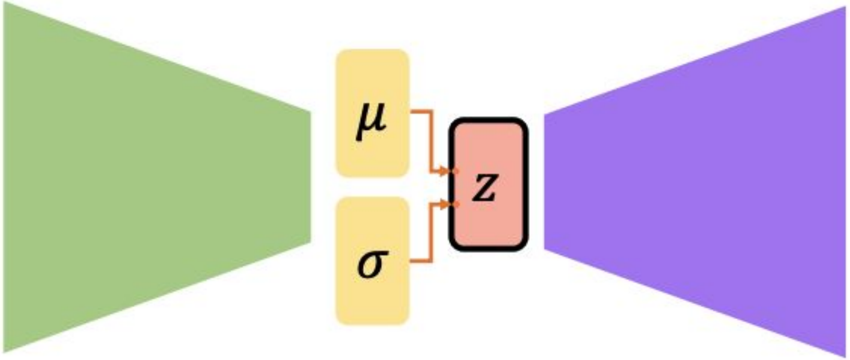
Reparameterizing the sampling layer

Key idea:

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

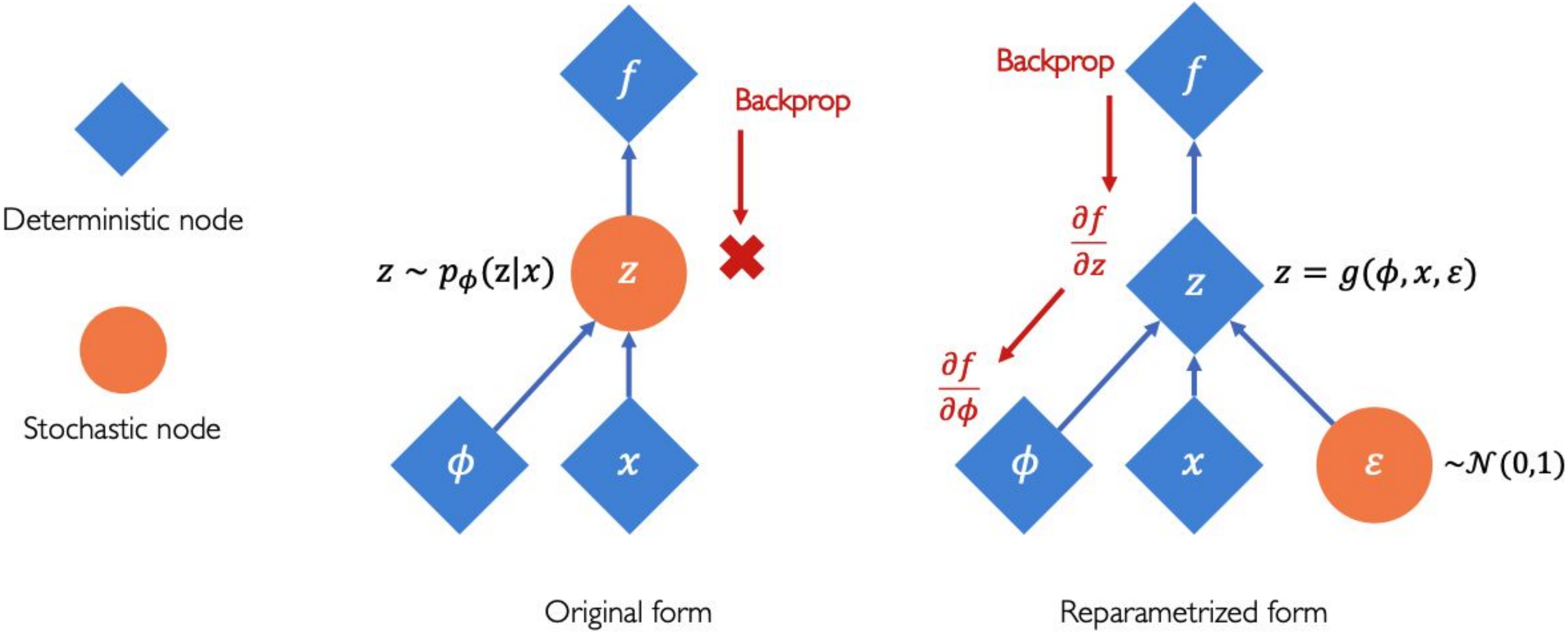
Consider the sampled latent vector as a sum of:

- a fixed $\boldsymbol{\mu}$ vector,
- and a fixed $\boldsymbol{\sigma}$ vector, scaled by random constant ε drawn from a prior distribution.

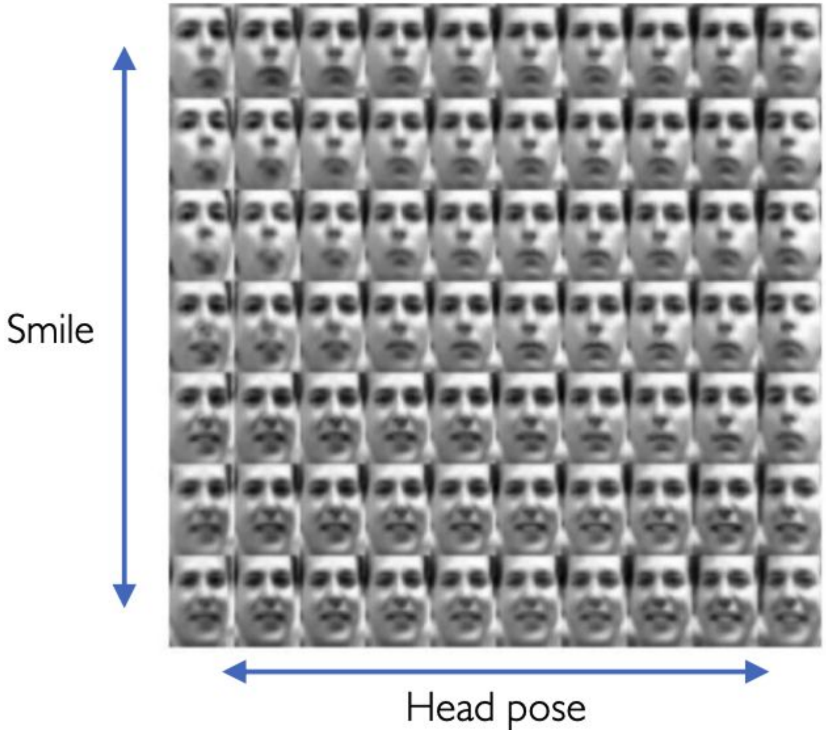


$$z = \mu + \sigma \odot \epsilon \text{ where } \epsilon \sim \mathcal{N}(0, 1)$$

Backprop through the sampling layer



VAEs: Latent Perturbation



- Slowly increase or decrease **a single latent variable** while keep all other variables fixed
- ⇒ Different dimension of $\mathbf{z}$ encodes different **interpretable** latent features
- Ideally, we want latent variables that are **uncorrelated** with each other because they should be *compact*
- Enforce a *diagonal prior* on the latent variables to encourage **independence**
- ⇒ **Disentanglement**

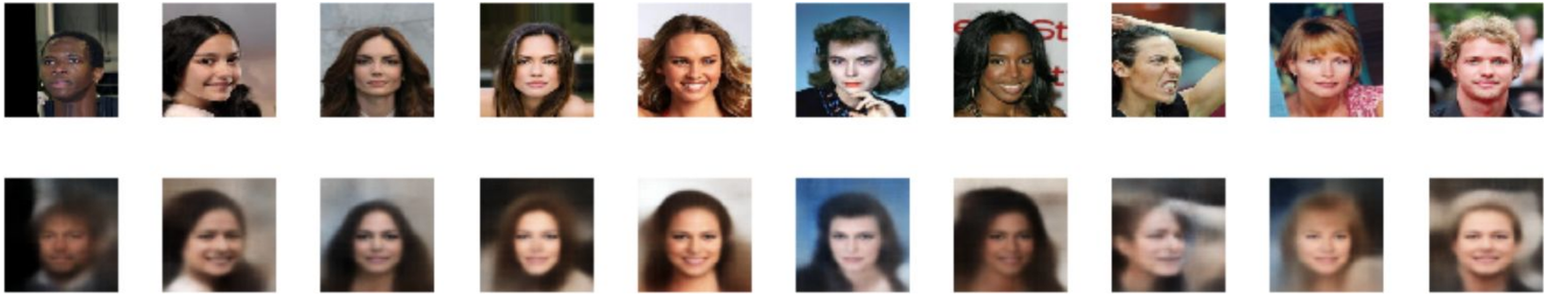
Application: CelebA dataset

The **CelebA** dataset is a collection of over 200,000 color images of celebrity faces along with various labels (eg. wearing hat, smiling, ect.)

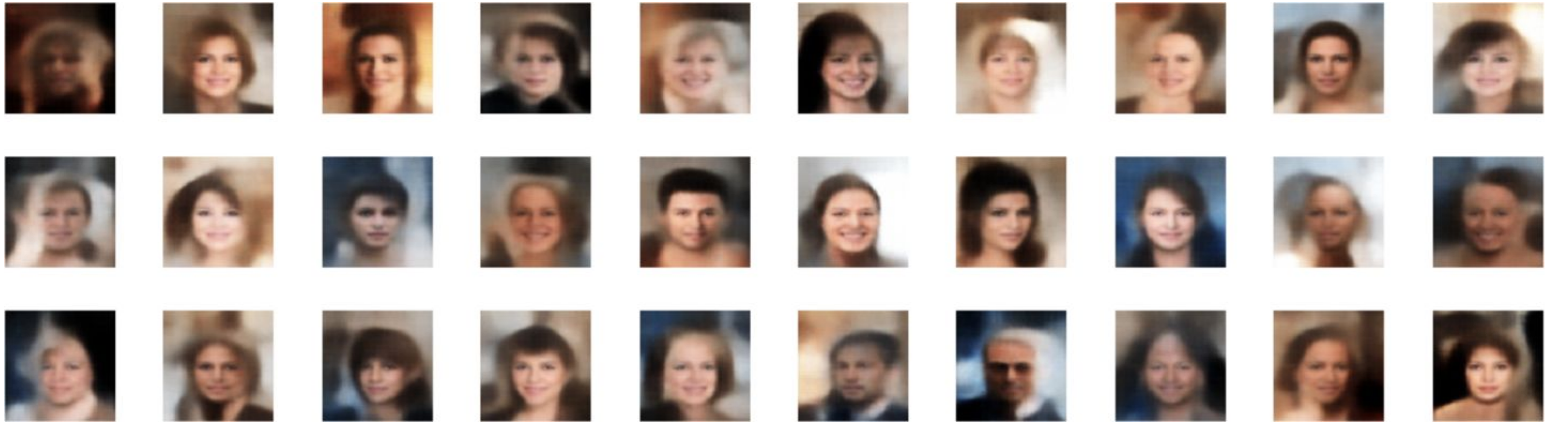
We won't need labels to train our VAE, but these might be useful later.

Application: VAE Analysis

Using the VAE from *Hou et al**, we can **reconstruct** the input images:



Or even generating **brand new** images from the latent space:



Xianxu Hou et al., "Deep Feature Consistent Variational Autoencoder," 2 October 2016, *https://arxiv.org/abs/1610.00291*

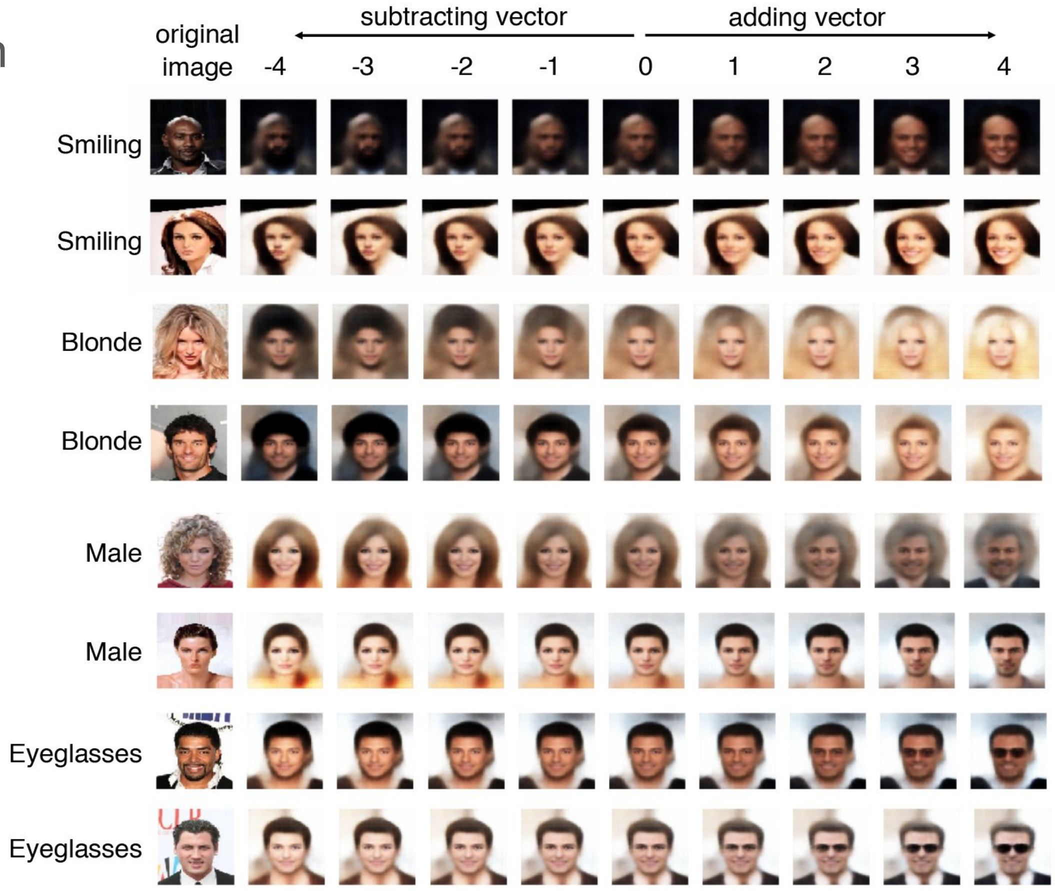
Latent Space Manipulation

We even can perform *arithmetics* on vectors in the latent space that has a **visual analogue** when decoded back into the original image.

$$\mathbf{z}_{\text{new}} = \mathbf{z} + \alpha \cdot \mathbf{z}_{\text{smiling}}$$

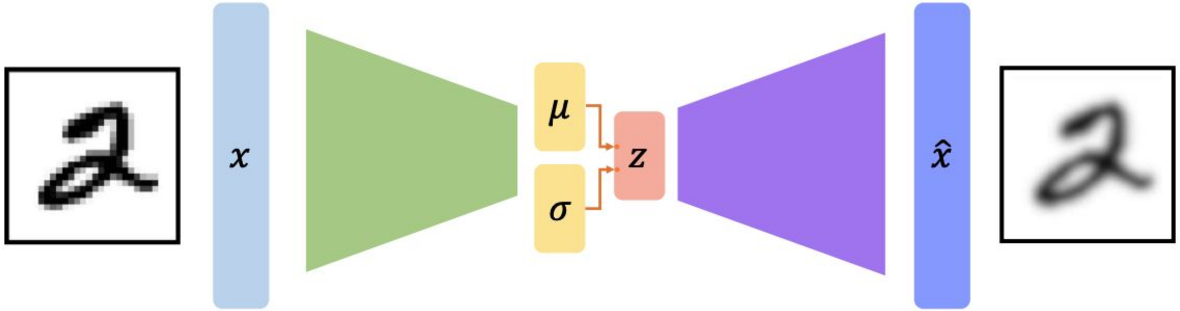
determines how much attribute to add

How do we find the **smiling** factor? Each image in the CelebA dataset is labeled with attributes, one of which is smiling!



VAE Summary

1. Compress representation of the visual world
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



Next time: Generative Adversarial Networks (GANs)

Acknowledgements

Slides contain figures from Andrej Karpathy (Stanford), Ava Soleimany (MIT), and various GAN researchers reproduced only for educational purposes.

Bonus Content

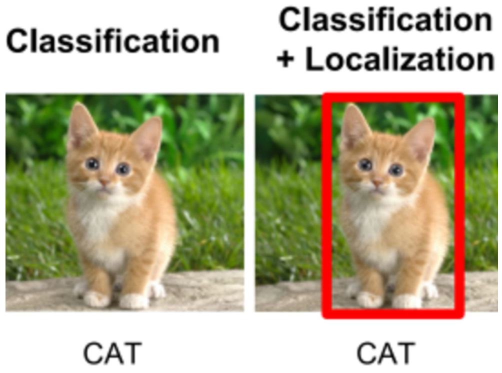
Review: Supervised vs. Unsupervised Learning

Supervised Learning

Data: (x,y) x is data, y is label

Goal: Learn function to map x → y

Examples: Classification, regression, detection, segmentation, ect.

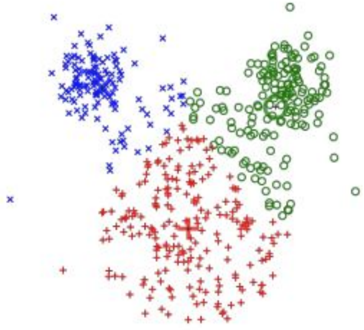


Unsupervised Learning

Data: x is data, no labels!

Goal: Learn some underlying structure of data

Examples: Clustering, feature or dimensionality reduction, ect.



K-means clustering

Application #2: Debiasing a dataset*

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

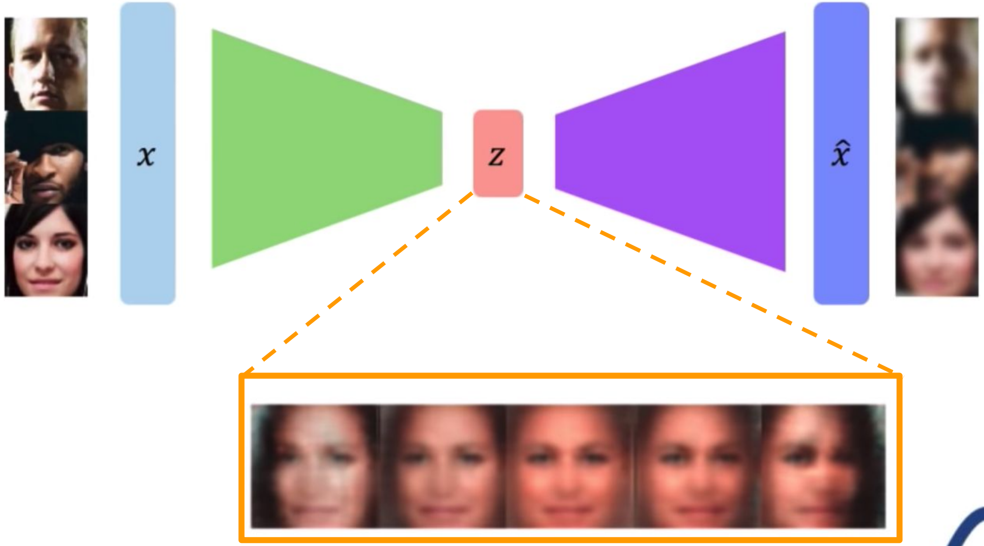
VS

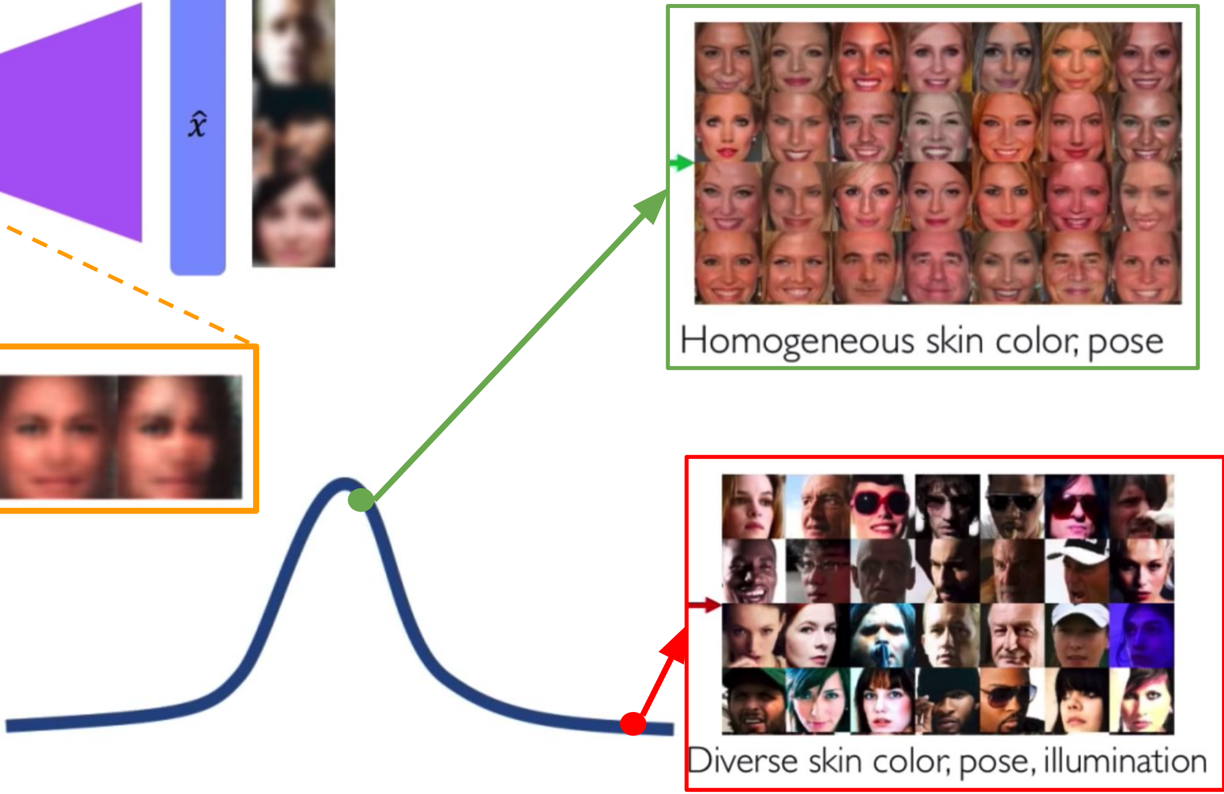
Homogeneous skin color, pose

Diverse skin color, pose, illumination

*Amini, Alexander, et al. "Uncovering and mitigating algorithmic bias through learned latent structure." *AAAI/ACM Conference on AI, Ethics, and Society*. 2019

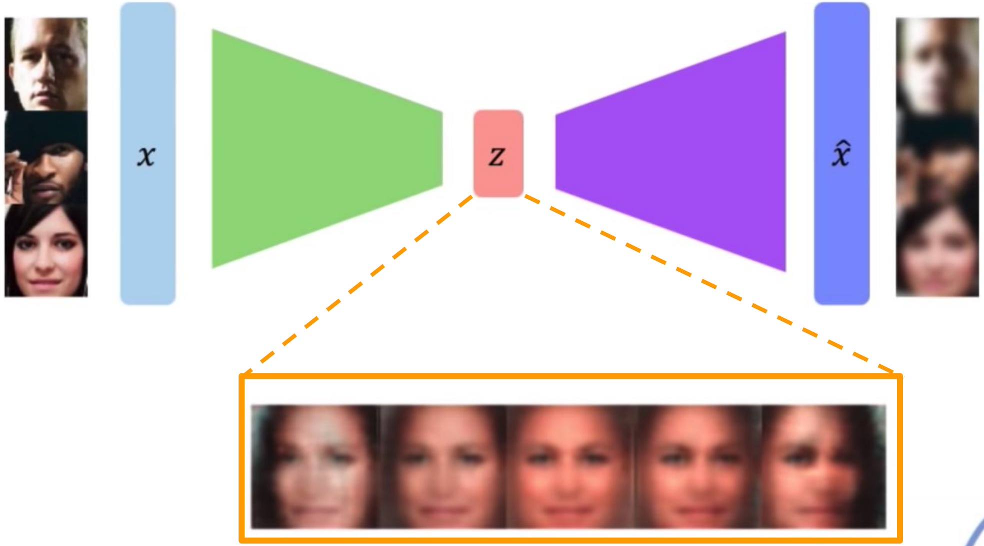
Mitigating bias through learned latent structure





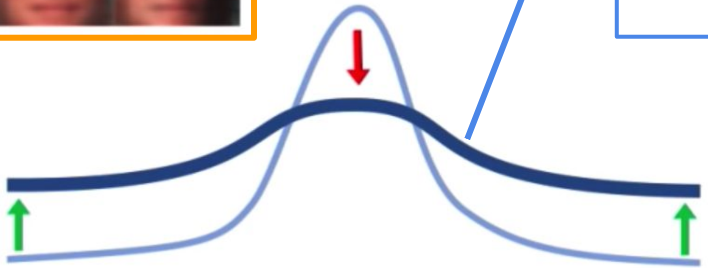
1. Learn latent structure
2. Estimate distribution

Less biased training data



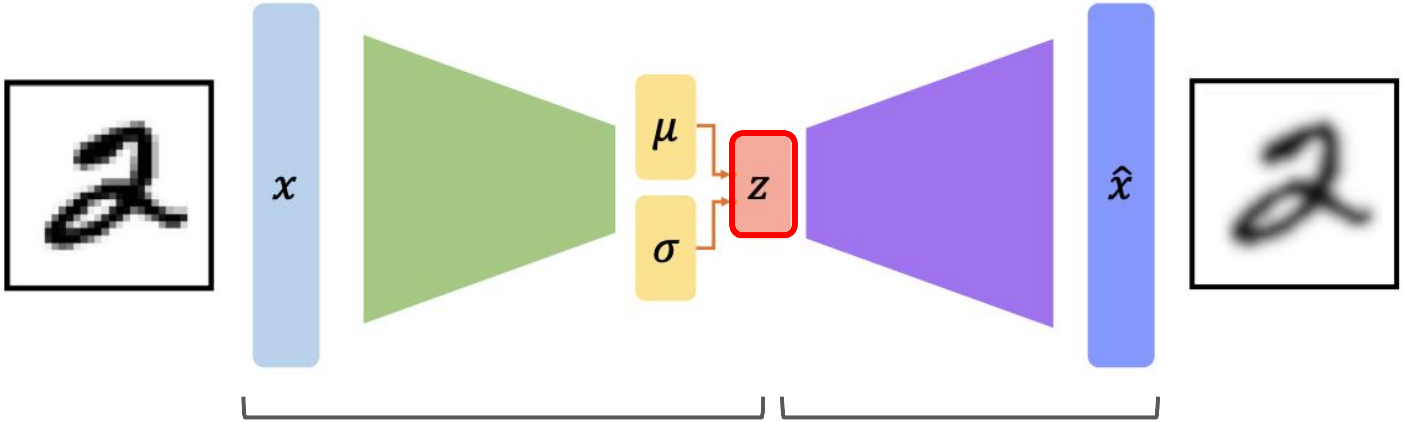
Latent distributions used to create fair and representative dataset

1. Learn latent structure
2. Estimate distribution



VAE Computation Graph

Problem: We cannot backpropagate gradients through sampling layers!



Encoder computes: $p_\phi(\mathbf{z}|\mathbf{x})$ Decoder computes: $p_\theta(\mathbf{x}|\mathbf{z})$

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

Priors on the latent distribution

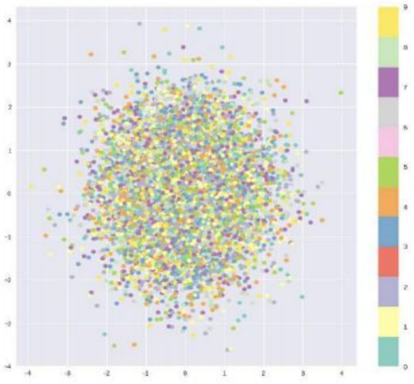
Inferred latent distribution

Fixed prior on latent distribution

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

KL-divergence between 2 distributions

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

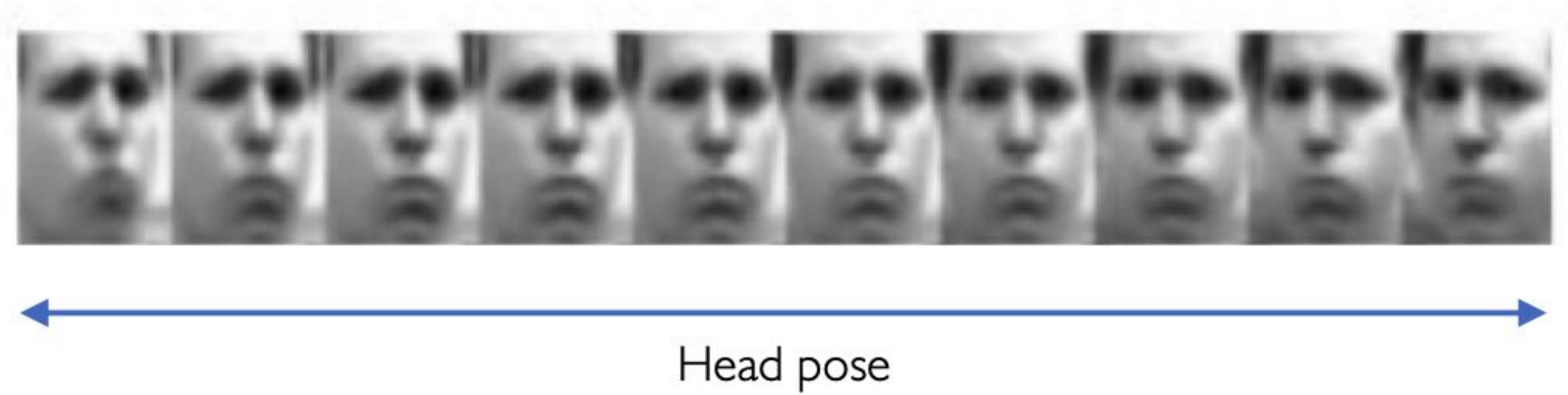


Common choice of prior: $p(z) = \mathcal{N}(\mu = 0, \sigma^2 = 1)$

- Encourage 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 (ie. memorizing the data)

VAEs: Latent Perturbation

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



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