Variational Auto Encoders

Introduction to Deep Learning

Spring 2022

Topics for the week

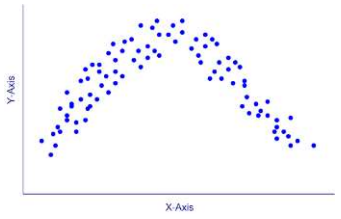
- Transformers
- GNNs
- VAEs
- GANs
- Connecting the dots

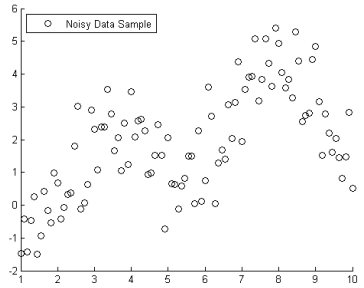
A new problem

- From a large collection of images of faces, can a network learn to *generate* new portrait
  - Generate samples from the distribution of “face” images
    - How do we even characterize this distribution?

The distribution of data

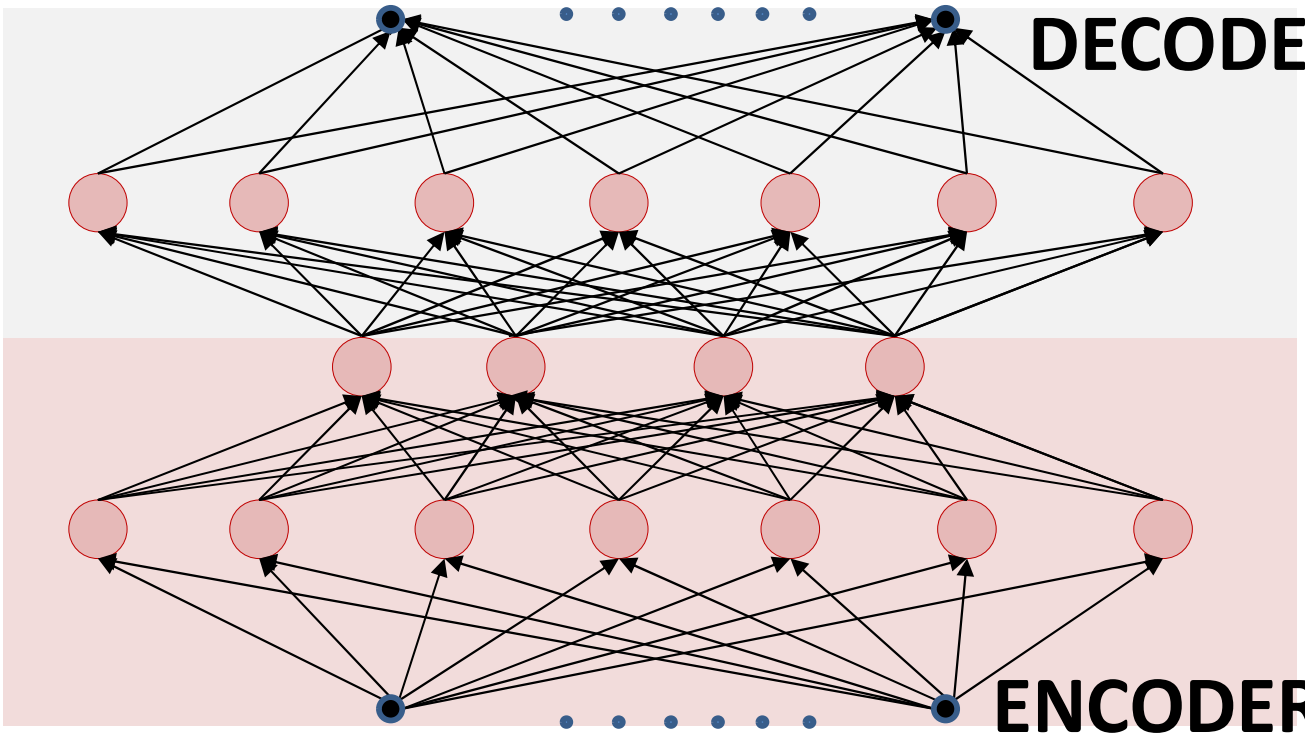
- **Hypothesis:** The data are distributed about a curved or otherwise non-linear manifold in high dimensional space
  - The principal components of all instances of the target class of data lie on this manifold
- To generate data for this class, we must select a point on this manifold

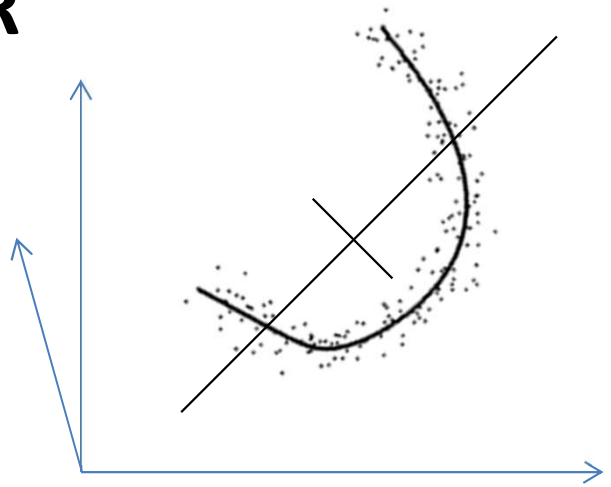




- **Problems:**
  - Characterizing the manifold
  - Having a good strategy for selecting points from it

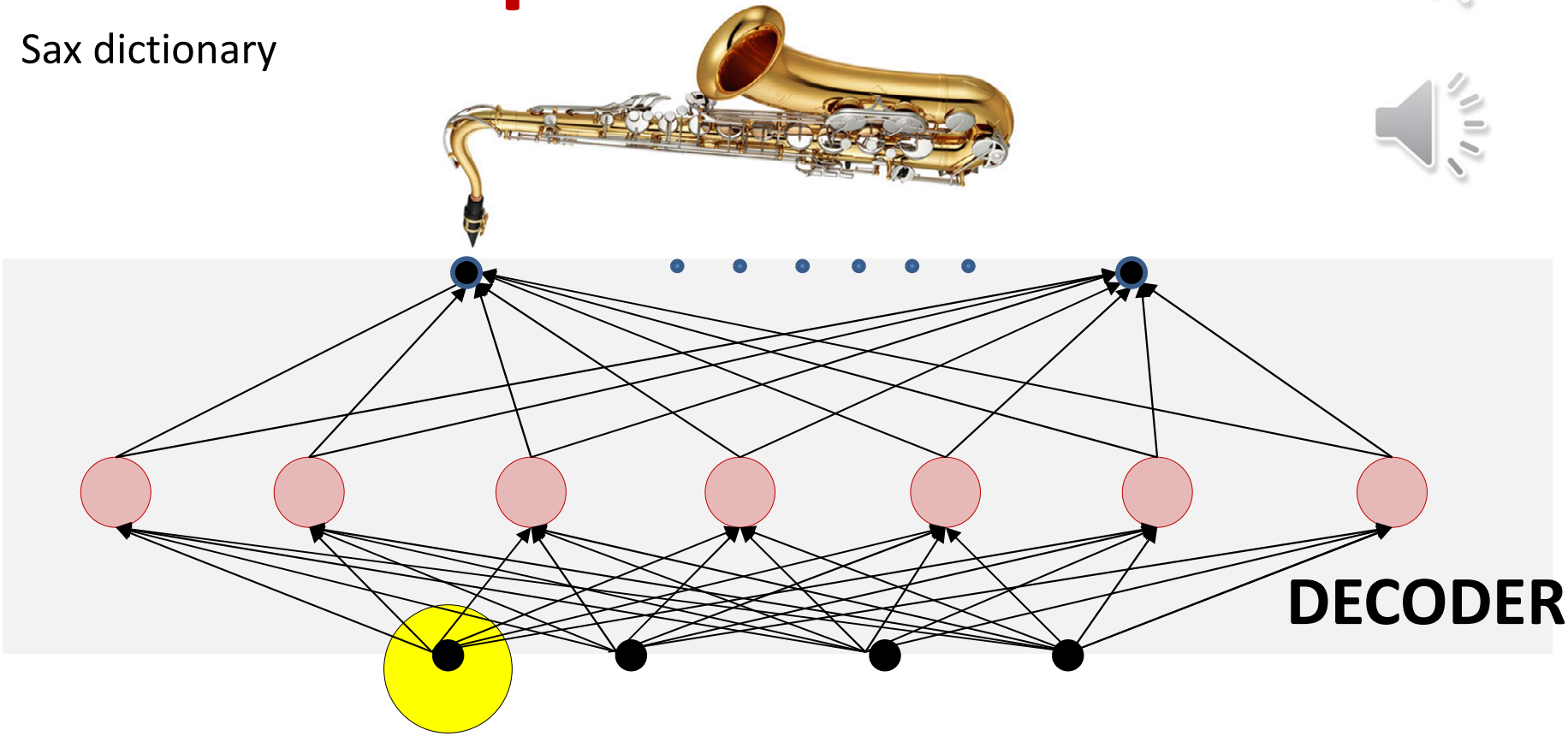
Recall : The AE





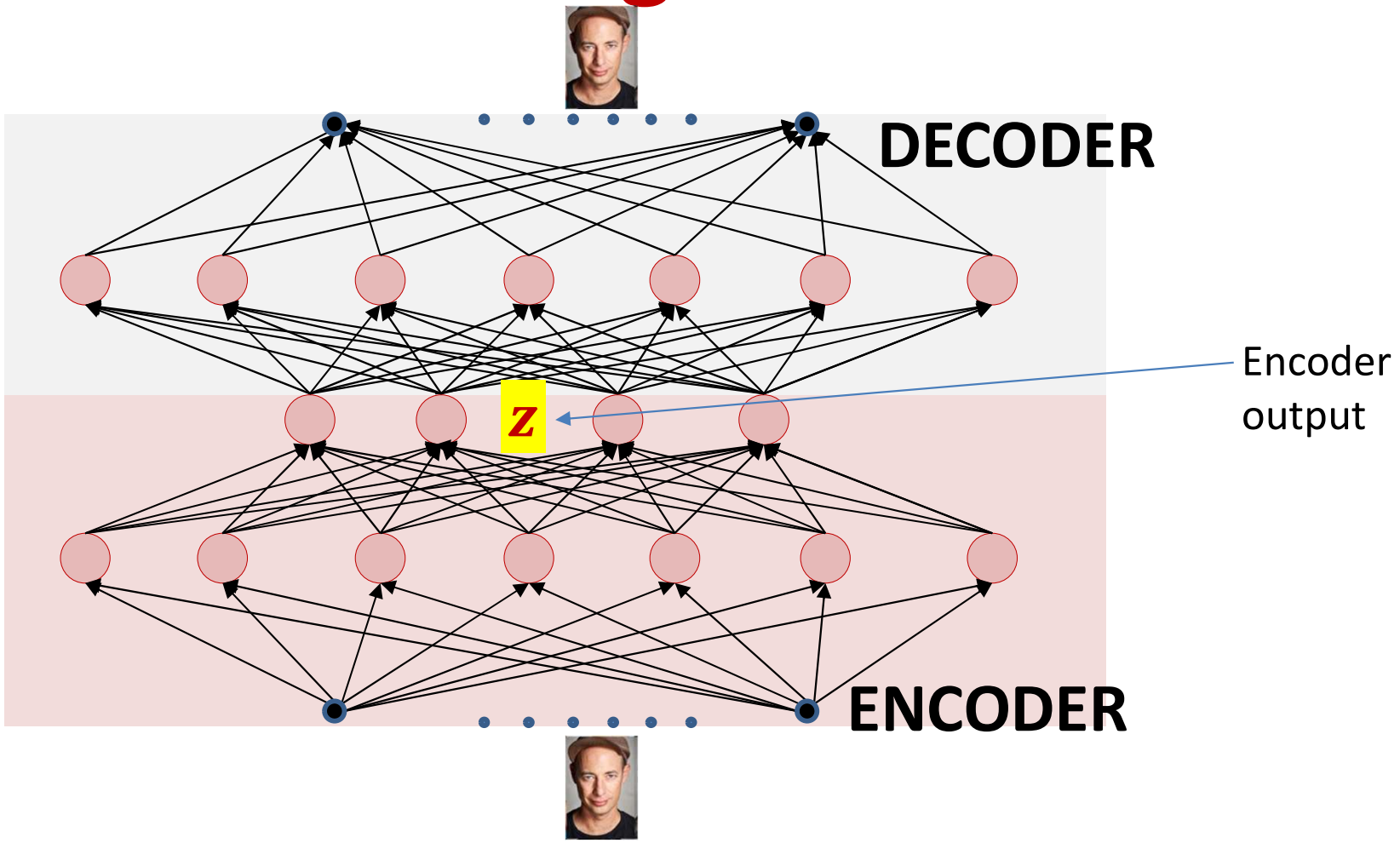
- The autoencoder captures the underlying manifold of the data
  - "Non linear" PCA
  - Deeper networks can capture more complicated manifolds
    - "Deep" autoencoders

Recap : The Decoder:



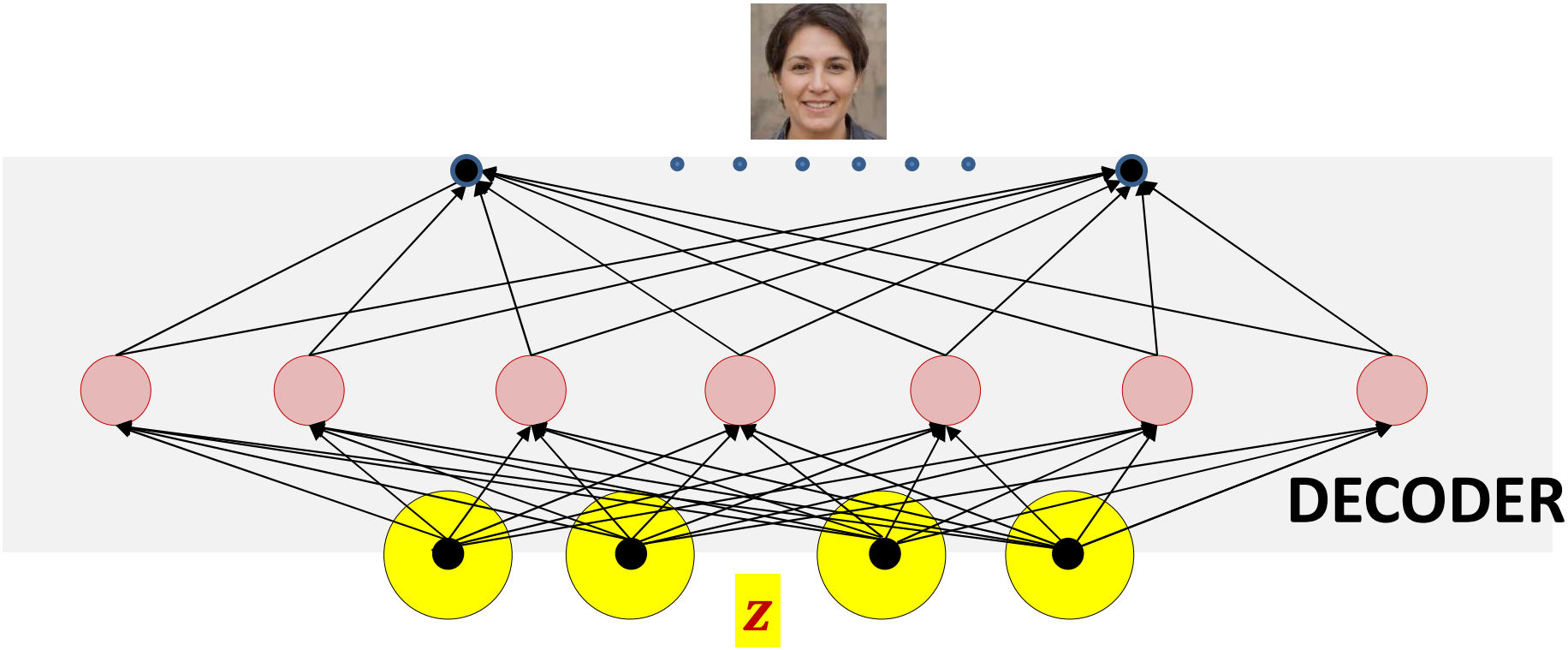
- The decoder represents a source-specific generative *dictionary*
- Exciting it will produce data similar to those from the source!

The AE for generation



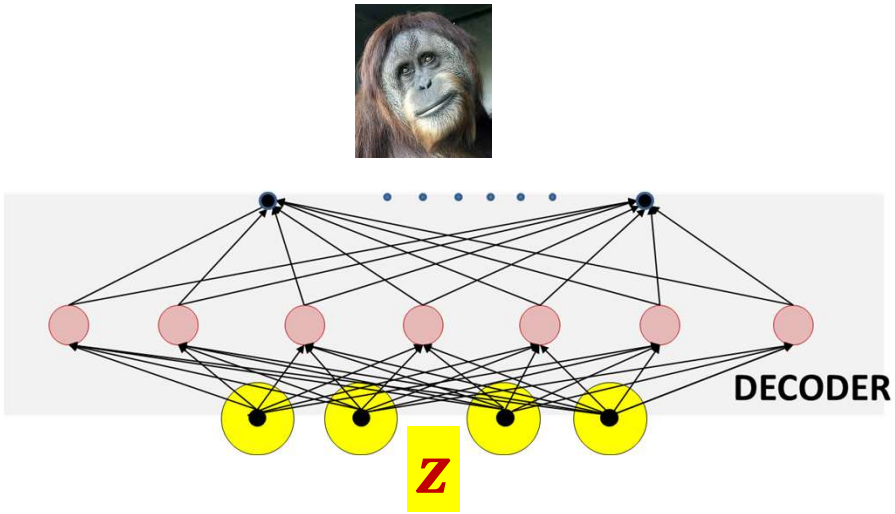
- Train AE with the pictures…

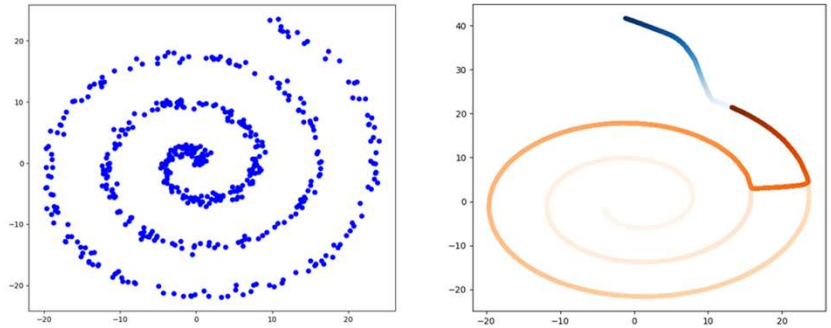
The face dictionary



- The decoder can now be used to generate instances from the “faces” manifold

The problem with AEs





- Improper choice of input to the decoder can result in incorrect generation
- How do we know *what* inputs are reasonable for the decoder?
- Solution : only choose input (z's) that are typical of the class
  - I.e. drawn from the distribution of z's for faces
  - But what is this distribution?

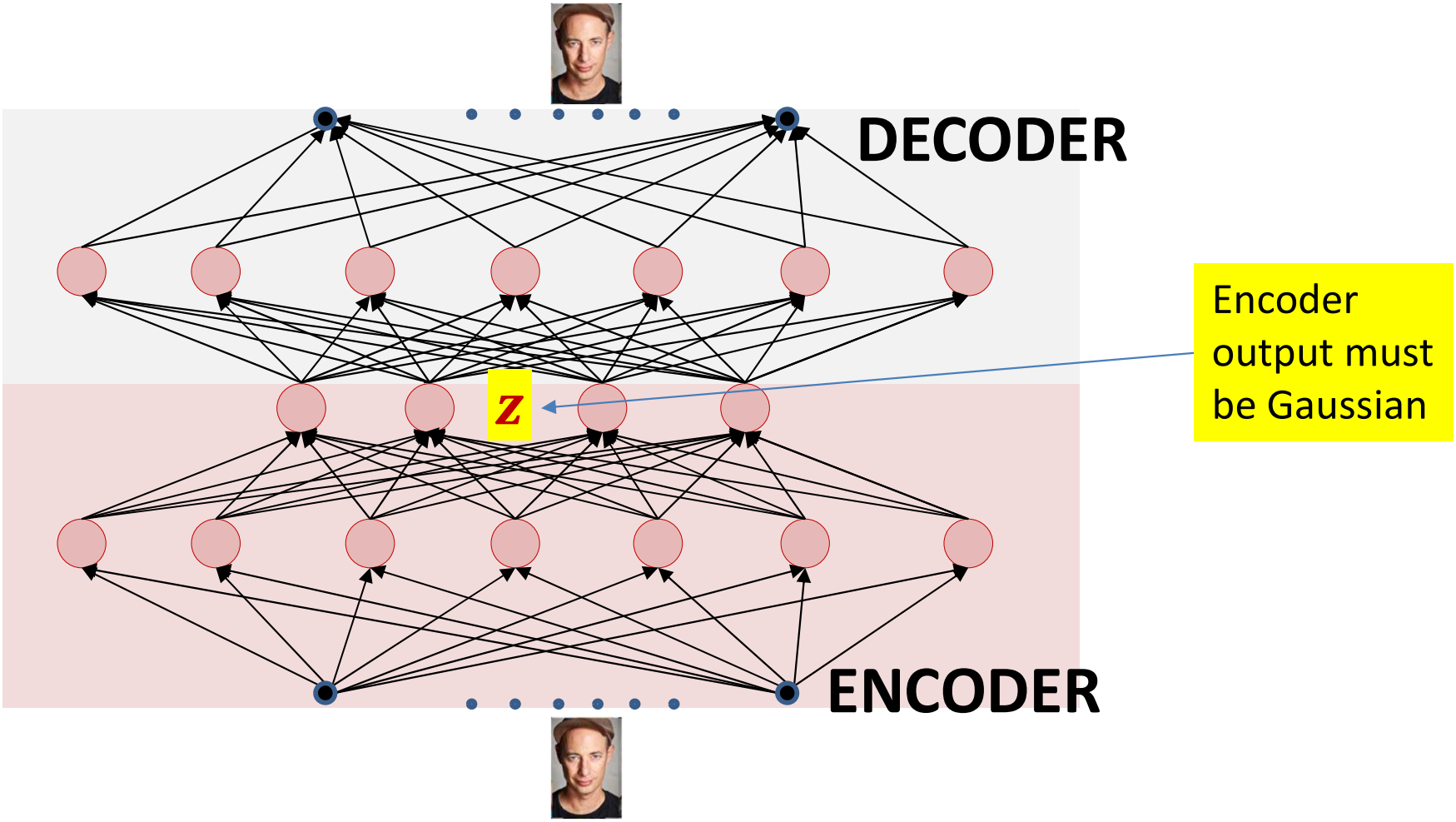
Poll 1

Poll 1

- The decoder in an AE can only generate data on a low-dimensional surface/manifold of the space
  - **True**
  - False

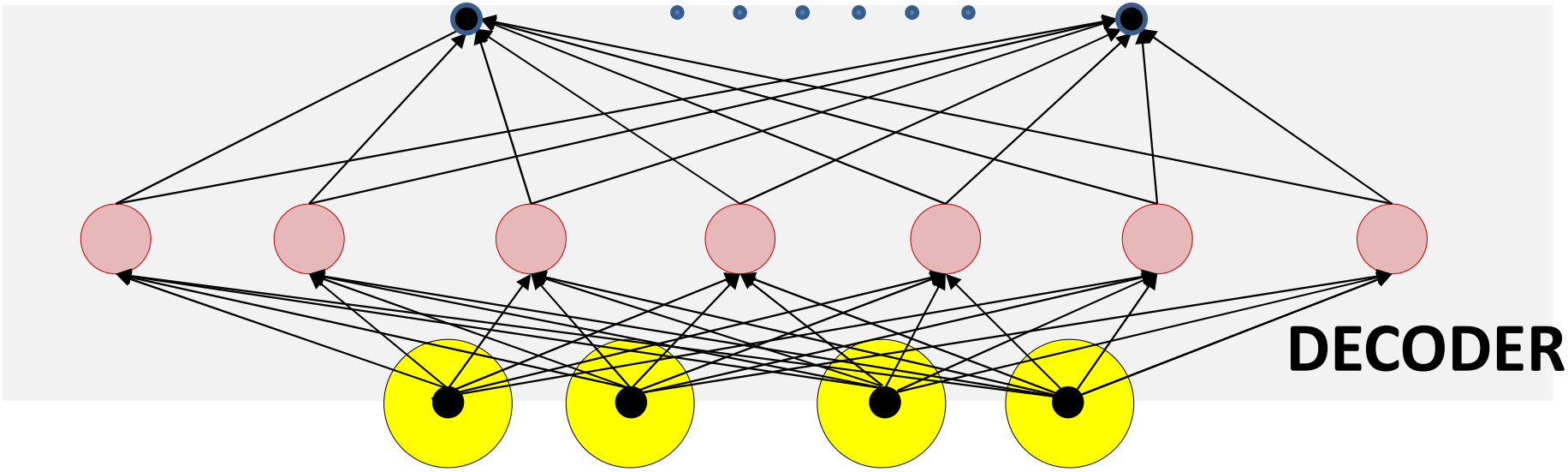
- What is true of the dimensionality of this manifold
  - It cannot be predicted and can be anything
  - **It is no greater than the dimensionality of the latent representation that is input to the decoder**
  - It will be the same as the input to the encoder

Impose distribution on z



- When training the autoencoder, explicitly impose the constraint that the hidden representation z must follow a specific distribution
  - E.g $P(z)$ is standard Gaussian $(N(0, I))$

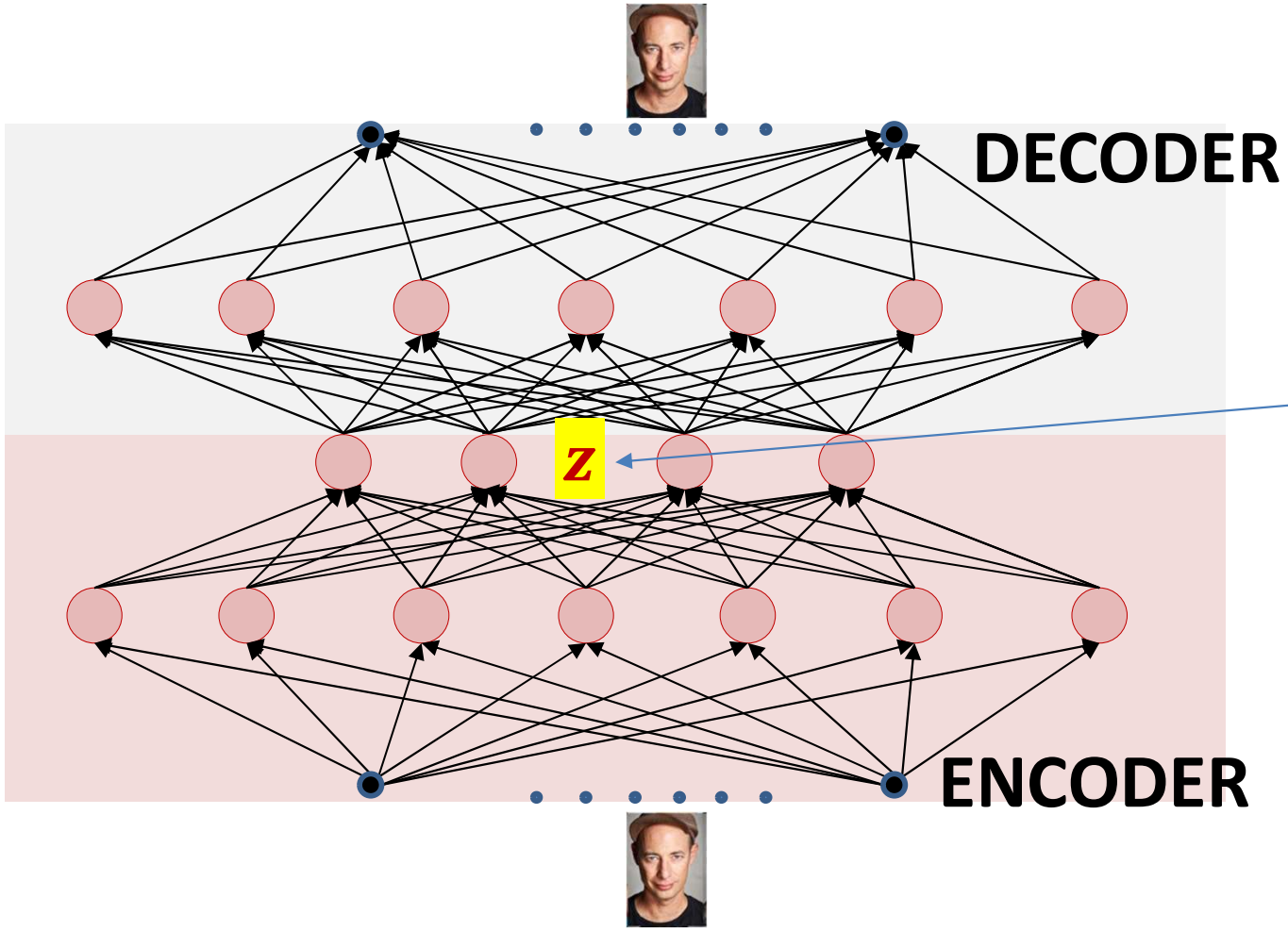
Generation



$z \sim N(0, I)$

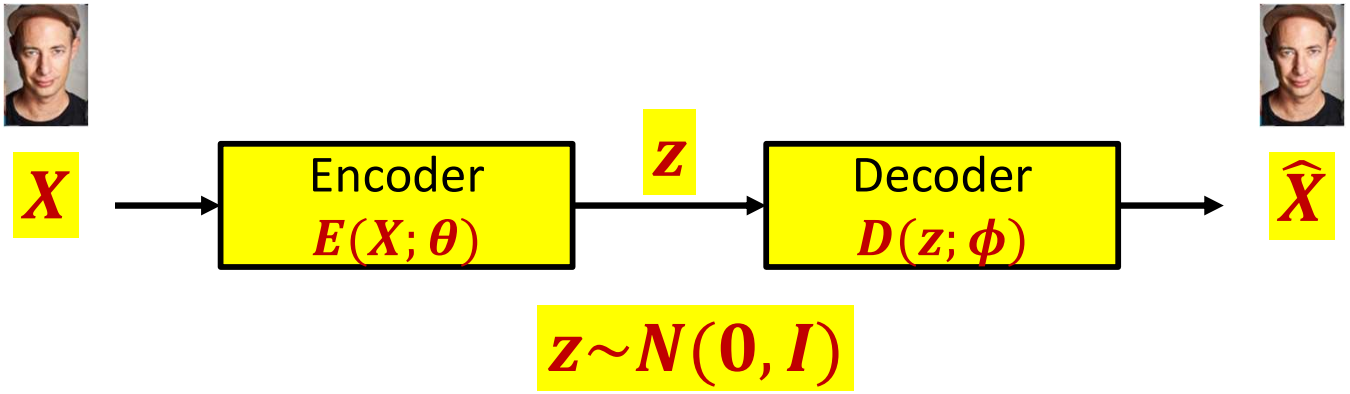
- To generate novel values, sample z from the prescribed distribution ($N(0, I)$)
- If the network is properly trained, and z is properly sampled, the output should be a reasonable generation
  - E.g. of a face

How to train the model



- **Problem:** How does one train an AE to ensure that the hidden representation z has a specific distribution, e.g. $N(0, I)$

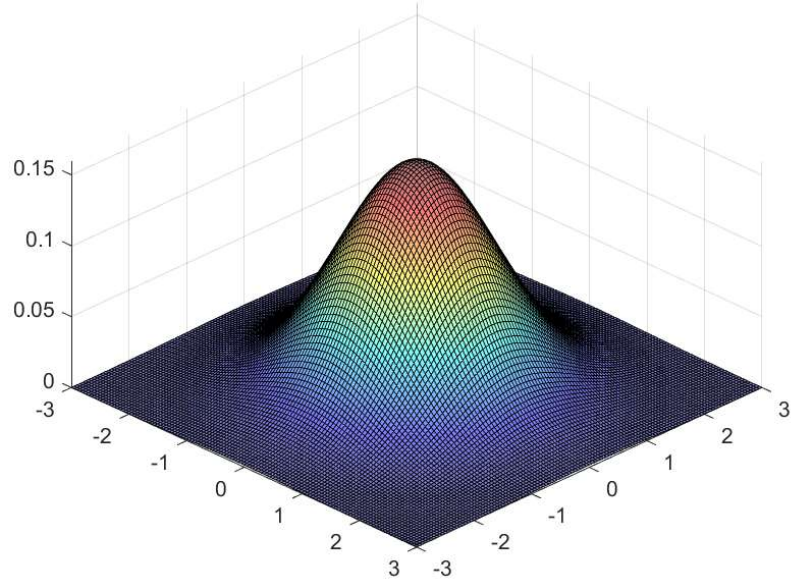
How to train the model

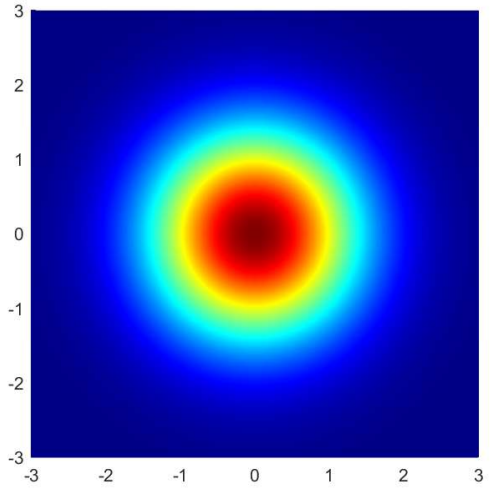


- **Problem:** How does one train an AE to ensure that the hidden representation z has a specific distribution, e.g. $P(z) = N(0, I)$

- Encoder and decoder may have arbitrarily complex structure and their own parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$

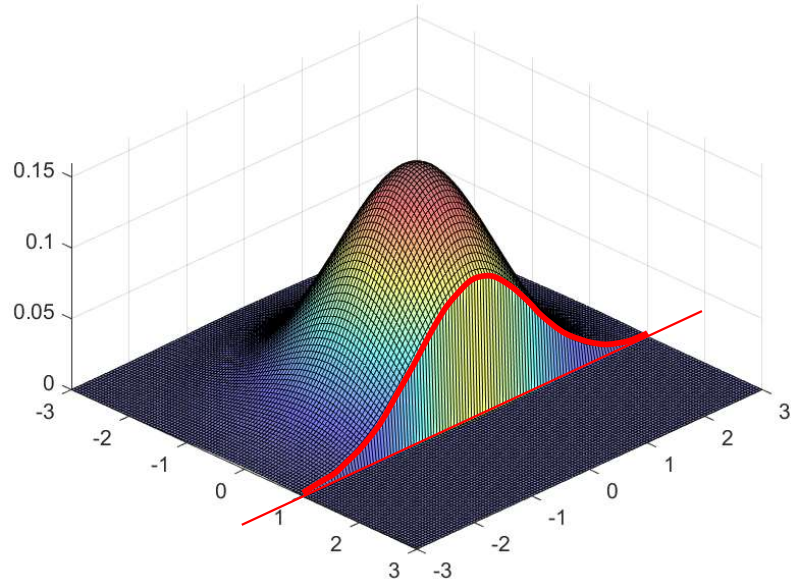
A property of isotropic Gaussians

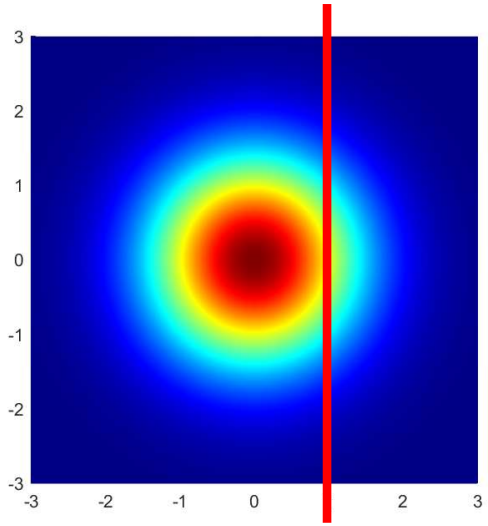




- $P(X, Y) = N(0, I)$
- The distribution is perfectly symmetric in every direction
  - The different variables (X and Y) are independent!
  - $P(X, Y) = P(X)P(Y)$
  - Each individually will also be isotropic: $P(X) = N(0, I)$

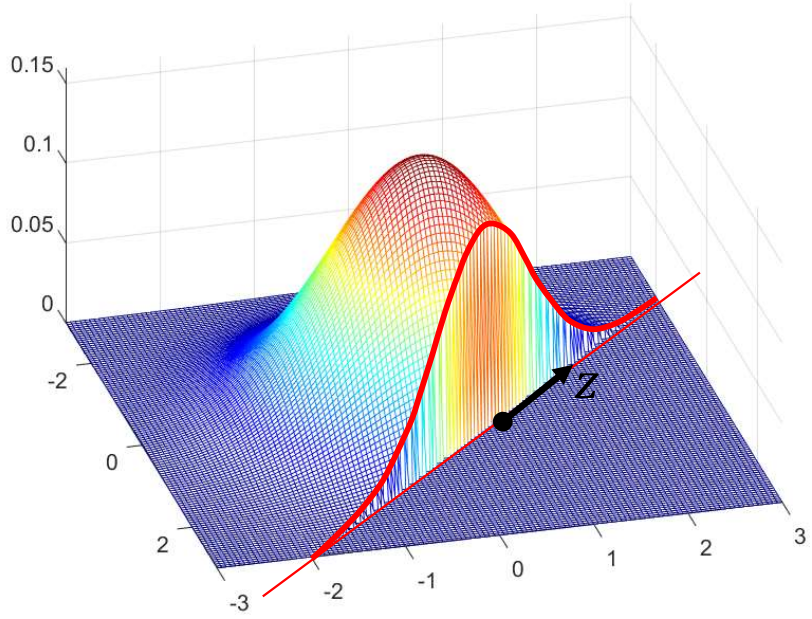
A property of isotropic Gaussians

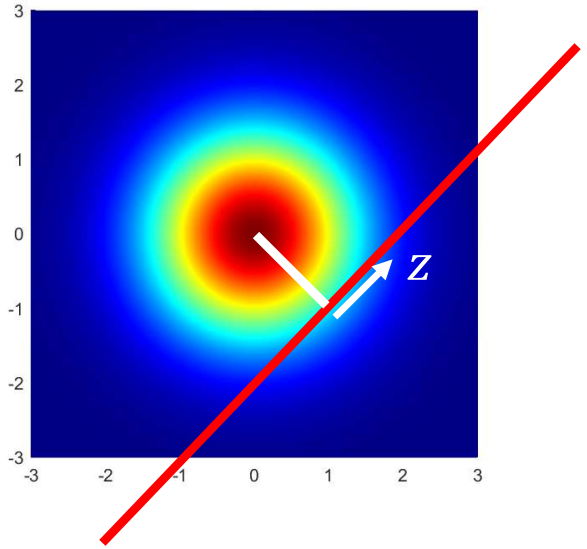




- Independence → $P(X|Y) = P(X)$
- $P(X|Y) = N(0, I)$
- The conditional distribution of X given Y is also an isotropic Gaussian regardless of the value of Y

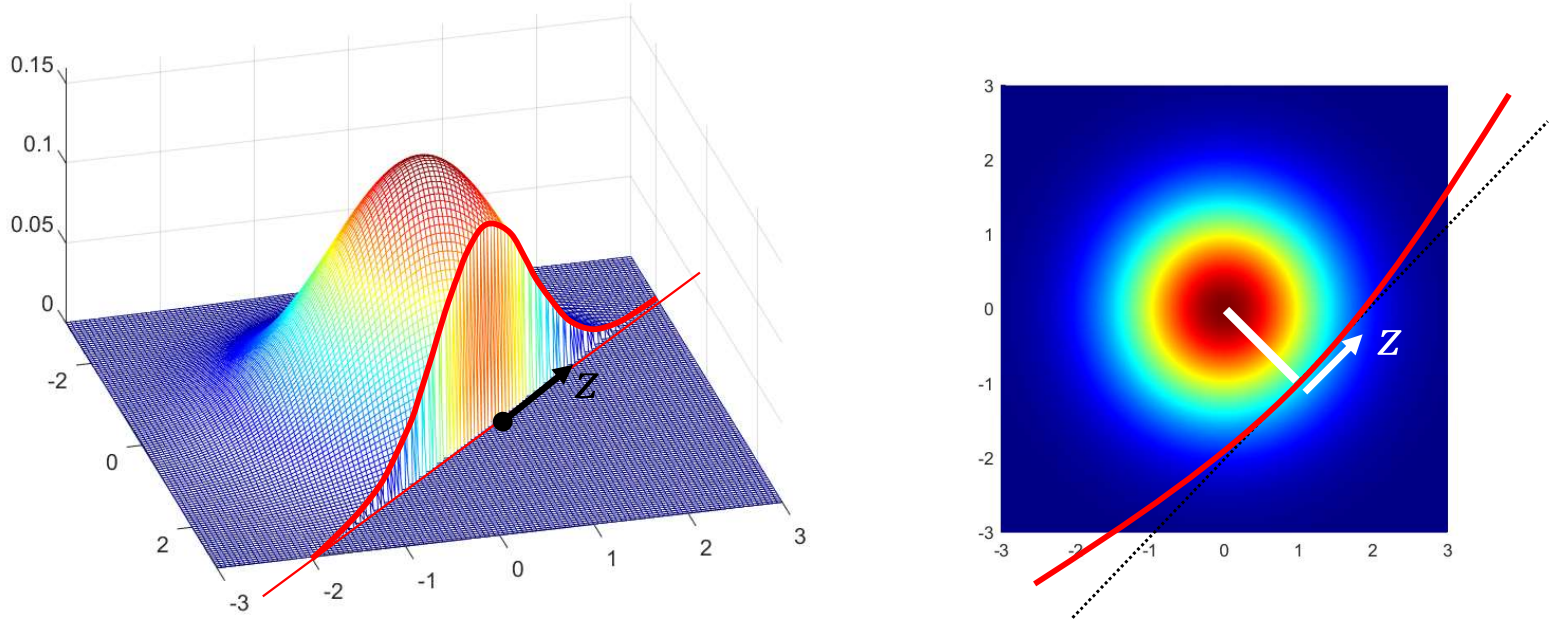
A property of isotropic Gaussians





- This property will hold regardless of the line/hyperplane along which you consider the conditional
  - $P(Z|aX + bY) = N(0, I)$

A property of isotropic Gaussians



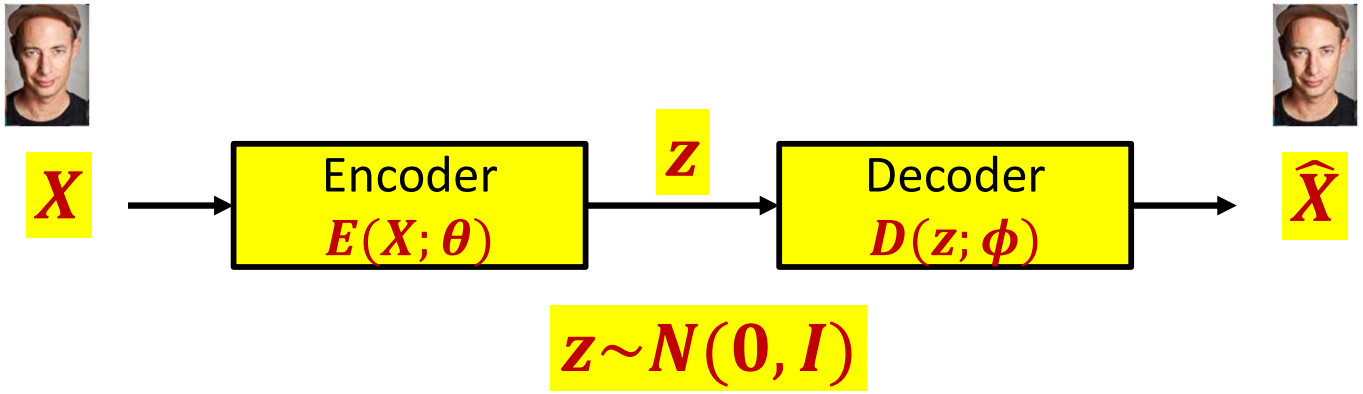
- This property will hold regardless of the line/hyperplane along which you consider the conditional
  - $P(Z|aX + bY) = N(0, I)$
- More generally, for "nearly linear" functions
  - $P(Z|f(X,Y)) = N(0, I)$
  - "Nearly linear" : The curve $f(X,Y) = const$ does not deviate much from a hyperplane in high-probability regions of (X,Y)
    - $E(|f(X,Y) - linear(X,Y)|^2) < \epsilon$

Poll 2

Poll 2

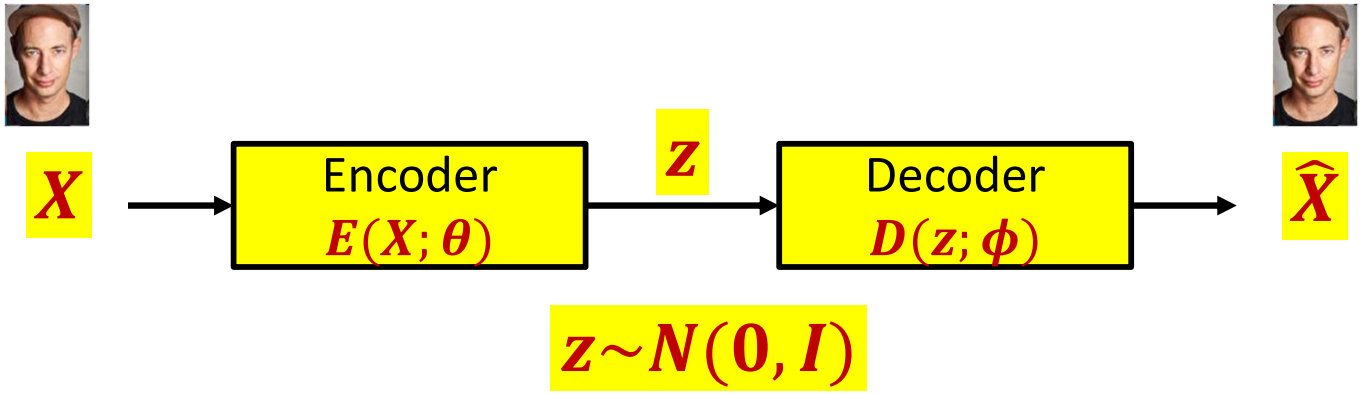
- The X is a random vector with an isotropic Gaussian distribution, the conditional distribution of the projection of X on any affine plane is also isotropic
  - **True**
  - False

Training with statistical constraints



- Minimize the error between X and $\hat{X}$
- Minimize the KL divergence between the distribution of z and the standard Gaussian $N(0, I)$
  - Minimize the negative log likelihood of z as computed from a standard Gaussian

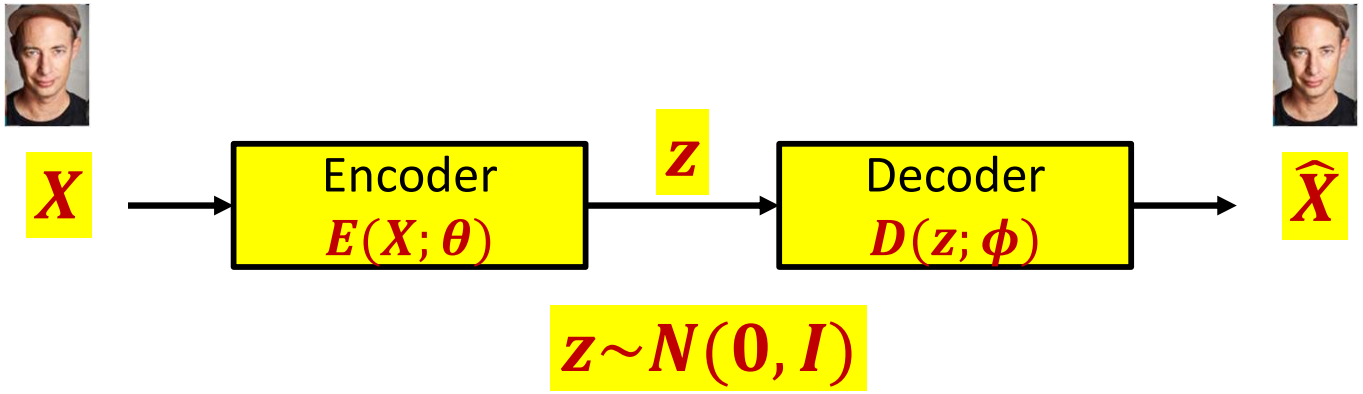
Training with statistical constraints



- $$\min \sum_{z} -\log N(z;0,I) = \min 0.5 \sum_{X} |z|^2$$
- $$= \min \sum_{X} |E(X;\theta)|^2$$

– Minimize the negative log likelihood of z as computed from a standard Gaussian

Training with statistical constraints



$$\min_{\theta,\phi} \sum_{X} \left|X - \hat{X}\right|^2 + \lambda \left| E(X,\theta) \right|^2$$

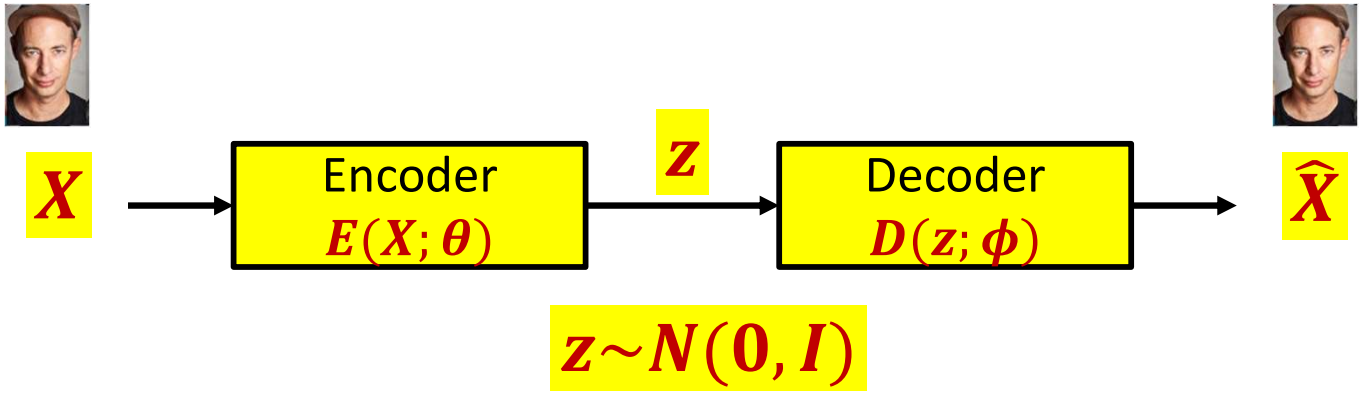
Poll 3

Poll 3

- A regular AE trained to also minimize the length of the hidden (latent) representation implicitly imposes an isotropic Gaussian distribution on the latent representation
  - **True**
  - False

- The output of an AE trained in this manner is no longer constrained to lie on a low dimensional manifold
  - True
  - **False**

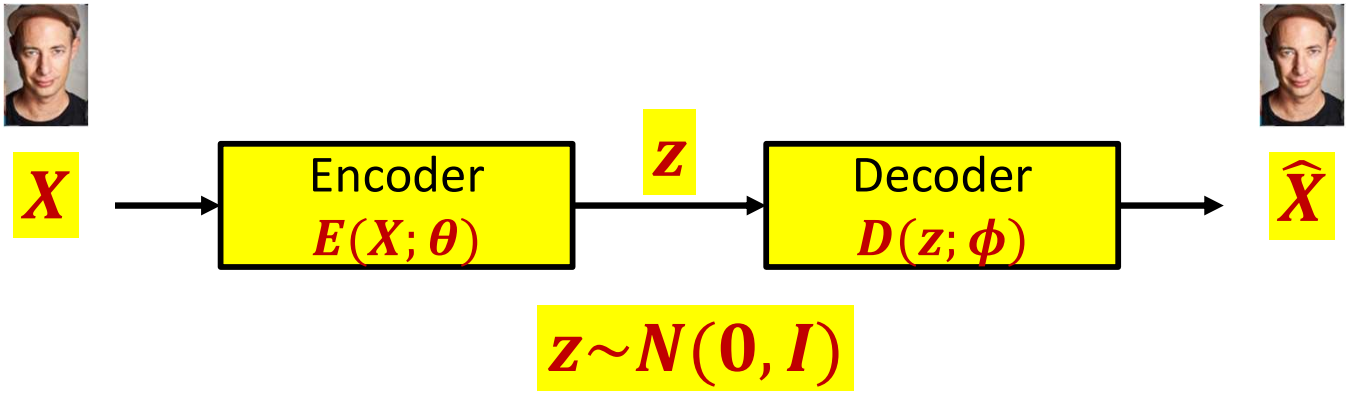
Training with statistical constraints



$$\min_{\theta,\phi} \sum_{X} \left|X - \hat{X}\right|^2 + \lambda \left| E(X,\theta) \right|^2$$

- This simple formulation does not adequately capture the variation in the data

How to train the model

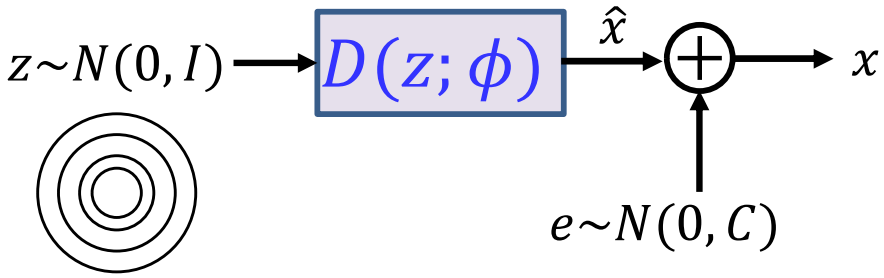


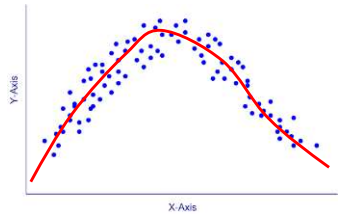
- An AE can learn to generate data from a specific class
  - For valid generation, the distribution of the latent representation z must be specified

- **Problem:** How does one train an AE to ensure that the hidden representation z has a specific distribution, e.g. $P(z) = N(0, I)$

The actual model

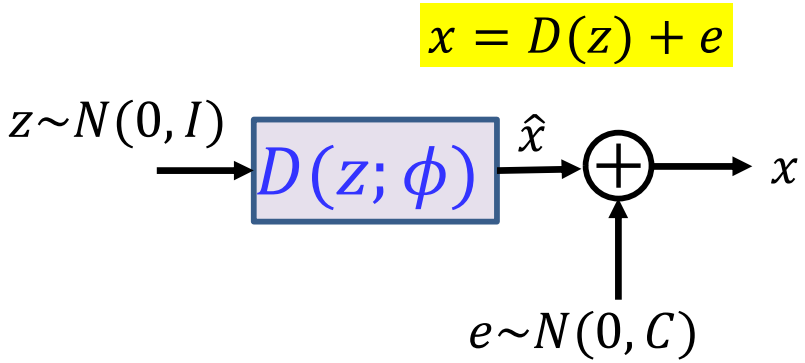
$$x = D(z) + e$$





- The "decoder" is actually a "generative" model for the data
  - Has a "generative story" for how the data are produced
- A K-dimensional vector z is drawn from a standard K-dimensional Gaussian and passed through the decoder
  - This results in data lying on a K-dimensional non-linear surface in the data space
- Then a full-rank, low-amplitude noise is added to it, to generate the final data
  - The actual data distribution is a fuzzy region around the surface.

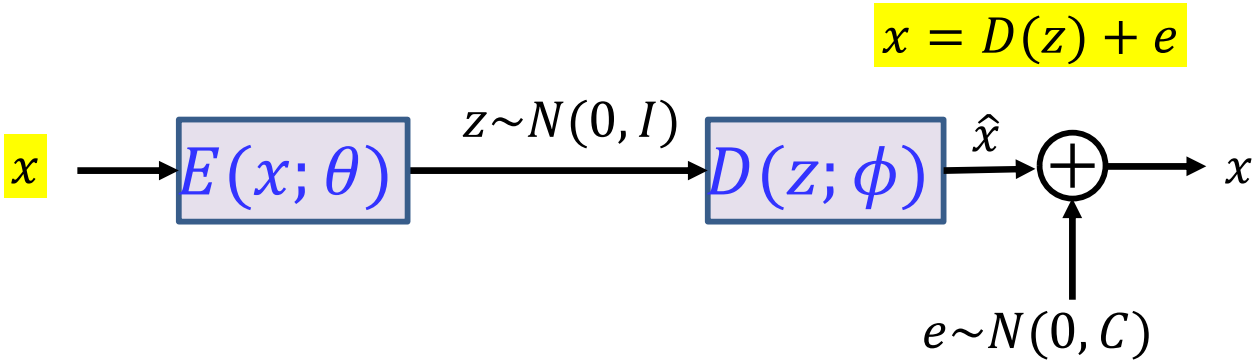
The distribution of the data

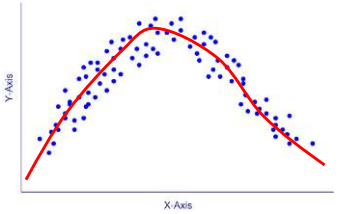


- If the z that went into the decoder to produce any training data instance x is known along with x, the decoder can be estimated

$$\underset{\phi}{\operatorname{argmin}}\, E[\|D(z;\phi) - x\|^2]$$

The distribution of the data



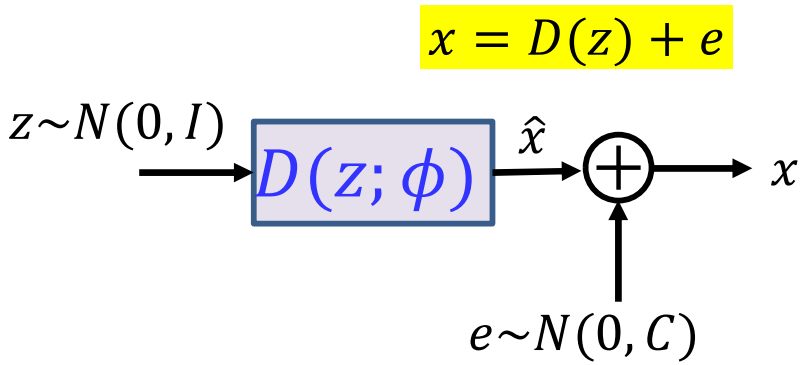


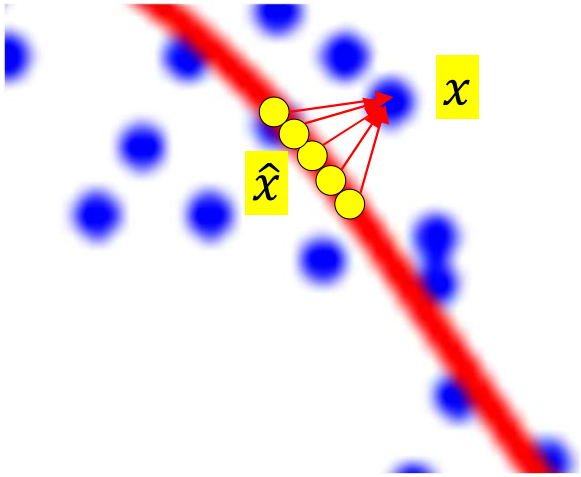
- If the z that went into the decoder to produce any training data instance x is known along with x, the decoder can be estimated

$$\underset{\phi}{\operatorname{argmin}}\, E[\|D(z;\phi) - x\|^2]$$

- The encoder estimates the z to permit us to estimate ϕ

The distribution of the data





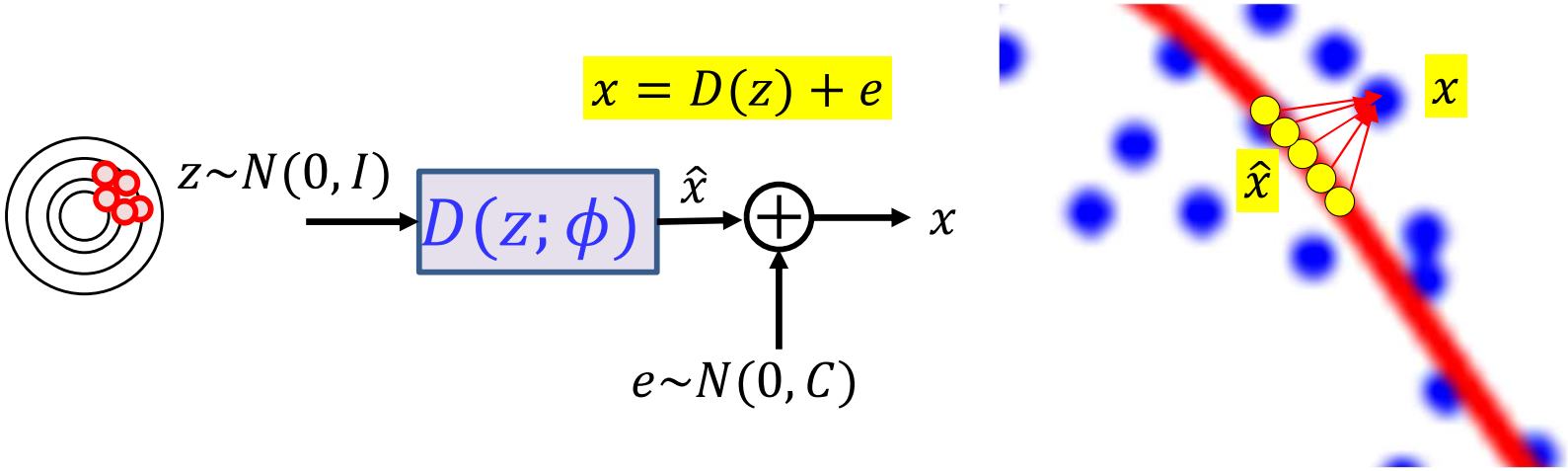
- If the z that went into the decoder to produce any data instance x is known along with x, the decoder can be estimated

$$\underset{\phi}{\operatorname{argmin}}\, E[\|D(z;\phi) - x\|^2]$$

- The encoder estimates the z to permit us to estimate ϕ

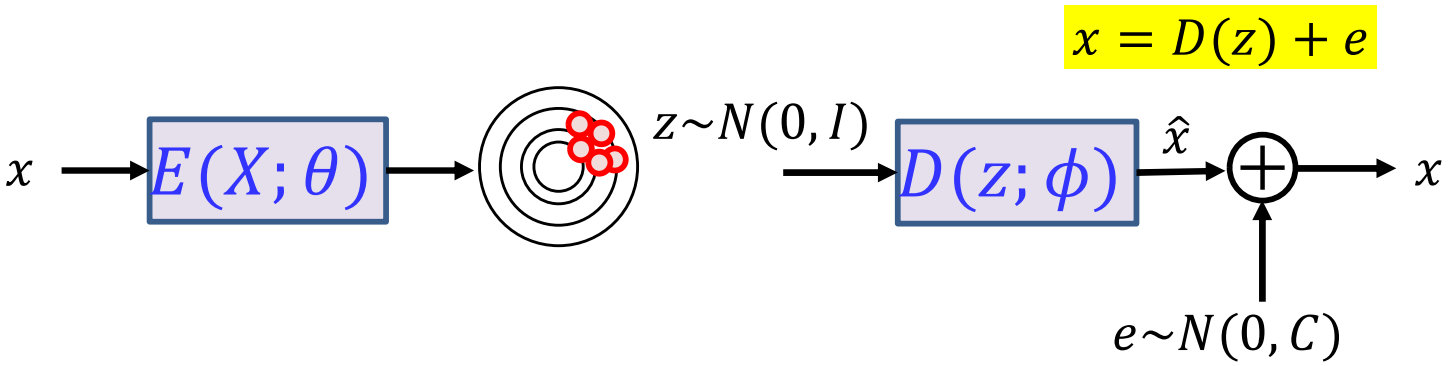
- Problem: Several $\hat{x}$ values exist that could be modified by noise to produce a given x
  - The z that could produce a given x is not unique

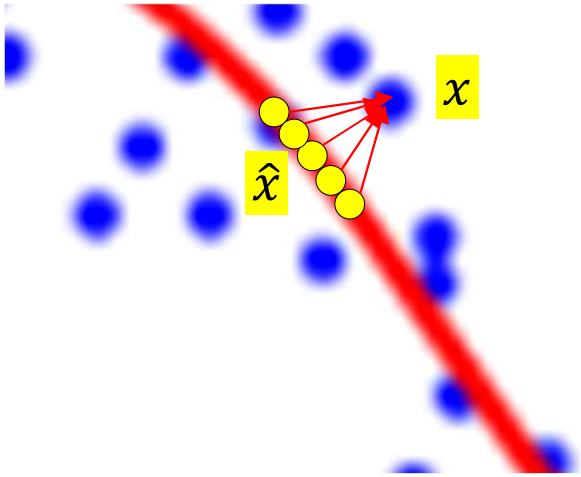
The distribution of the data



- There is an entire *distribution* of zs that could produce a given x
  - z values that correspond to more probable values of noise are more probable
  - The distribution of z for a given x are dependent on that x: $P(z|x)$
- Instead of finding the unique z for any x, we will find the distribution $P(z|x)$

The distribution of the data





- There is an entire *distribution* of zs that could produce a given x
  - z values that correspond to more probable values of noise are more probable
  - The distribution of z for a given x are dependent on that x: $P(z|x)$
- Instead of finding the unique z for any x, we will find the distribution $P(z|x)$

- ***The variational autoencoder***
  - So named because the learning procedure utilizes a variational bound on the likelihood of the data

The variational autoencoder

- The *encoder* computes the *distribution* $P(z|x)$ for any x
- The *decoder* tries to convert a randomly sampled z from $P(z|x)$ to x

The variational autoencoder

- The *encoder* computes the *distribution* $P(z|x)$ for any x
- The *decoder* tries to convert a randomly sampled z from $P(z|x)$ to x

- **Training the encoder**:
  - Estimate θ to make the zs that can be encoded $\hat{x}$ values that are closer to x more probable

The variational autoencoder

- The *encoder* computes the *distribution* $P(z|x)$ for any x
- The *decoder* tries to convert a randomly sampled z from $P(z|x)$ to x

- **Training the encoder**:
  - Estimate θ to make the zs that can be auto-encoded $\hat{x}$ values that are closer to x more probable

- **Training the decoder**:
  - Estimate $\boldsymbol{\phi}$ to make the noise between $\hat{x} = D(z)$ and x more probable

The variational autoencoder

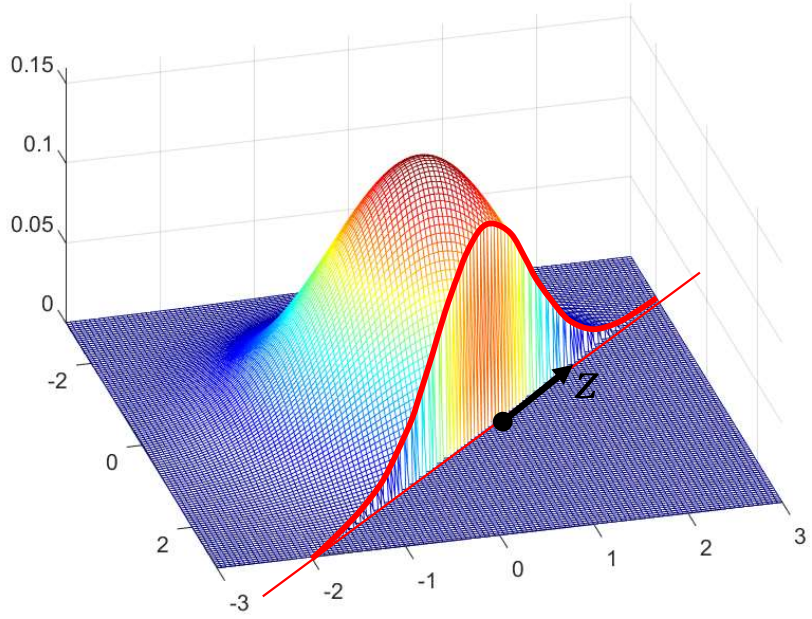
- The *encoder* computes the *distribution* $P(z|x)$ for any x
- The *decoder* tries to convert a randomly sampled z from $P(z|x)$ to x

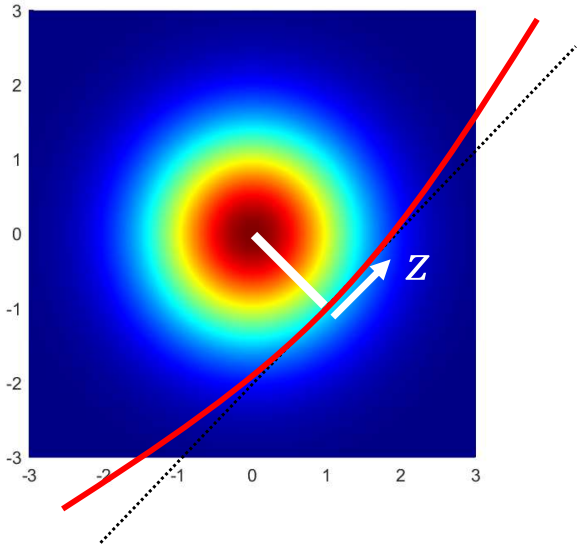
- **Training the encoder**:
  - Estimate θ to make the zs that can be encoded $\hat{x}$ values that are closer to x more probable

- **Training the decoder**:
  - Estimate $\boldsymbol{\phi}$ to make the noise between $\hat{x} = D(z)$ and x more probable

- **Constraint on z:** ?

A property of isotropic Gaussians





- This property will hold regardless of the line/hyperplane along which you consider the conditional
  - $P(Z|aX + bY) = N(0, I)$
- More generally, for "nearly linear" functions
  - $P(Z|f(X, Y)) \approx N(0, I)$

The variational autoencoder

- The *encoder* computes the *distribution* $P(z|x)$ for any x
- The *decoder* tries to convert a randomly sampled z from $P(z|x)$ to x

- **Training the encoder**:
  - Estimate θ to make the zs that can be encoded $\hat{x}$ values that are closer to x more probable

- **Training the decoder**:
  - Estimate $\boldsymbol{\phi}$ to make the noise between $\hat{x} = D(z)$ and x more probable

- **Constraint on z:** Make $P(z|x)$ as close to the standard Gaussian as possible

The variational autoencoder

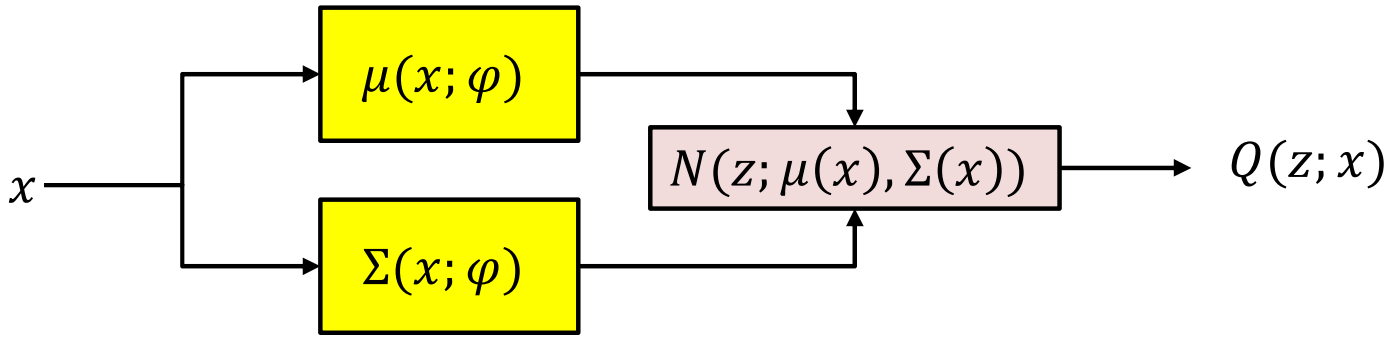
- The *encoder* computes the *distribution* $P(z|x)$ for any x
- The *decoder* tries to convert a randomly sampled z from $P(z|x)$ to x

- **Training the encoder**:
  - Estimate theta to make the zs that can be encoded to x more probable

- **Training the decoder**:
  - Estimate $\boldsymbol{\phi}$ to make the noise between $\hat{x} = D(z)$ and x more probable

- Constraint on z: Make $P(z|x)$ as close to the standard Gaussian as possible

Approximating $P(z|x)$



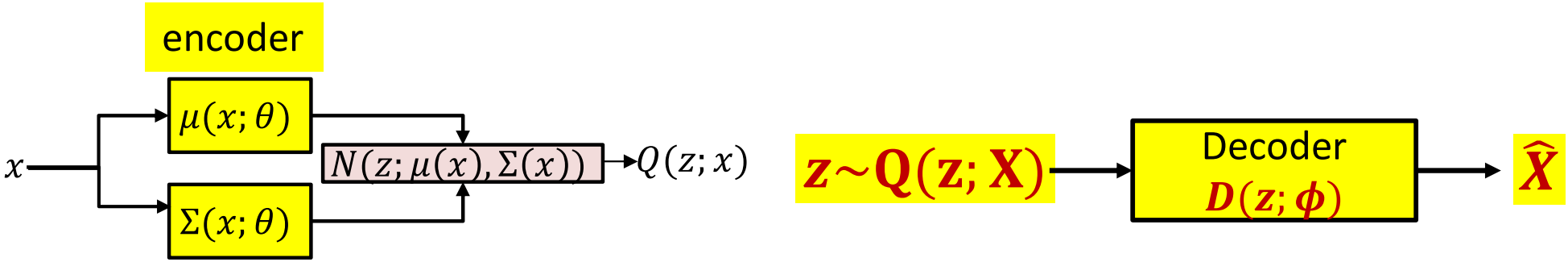
$\mu(x;\varphi)$ and $\Sigma(x;\varphi)$ are parametric functions of x, with parameters that we jointly represent as φ

- We approximate $P(z|x)$ as

$$P(z|x) \approx Q(z,x) \;=\; Gaussian\; N(z;\mu(x),\Sigma(x))$$

  - where $\mu(x;\varphi)$ and $\Sigma(x;\varphi)$ are estimated such that $Q(z,x)$ approximates $P(z|x)$ as closely as possible
  - For convenience, we will assume $\Sigma(x;\varphi)$ is a diagonal matrix, represented entirely by its diagonal elements
- We will use $Q(z,x)$ as our proxy for $P(z|x)$

The variational autoencoder



- The *encoder* computes the *distribution* $P(z|x)$ for any x
- The *decoder* tries to convert a randomly sampled z from $P(z|x)$ to x

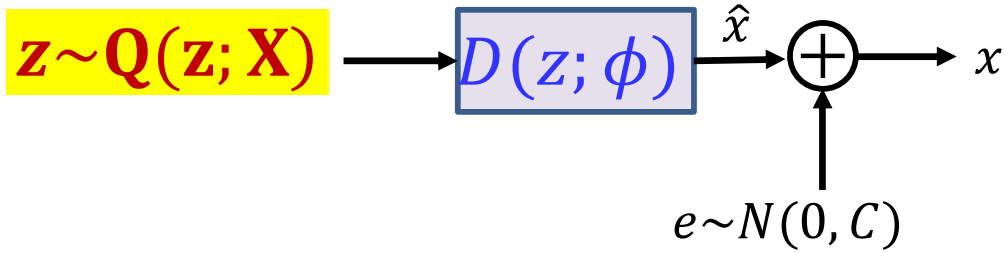
- **Training the encoder**:
  - Estimate $\boldsymbol{\theta}$ to make the zs that can be encoded to x more probable

- **Training the decoder**:
  - Estimate $\boldsymbol{\phi}$ to make the noise between $\hat{x} = D(z)$ and x more probable

- Constraint on z: Make $P(z|x)$ as close to the standard Gaussian as possible

Training the encoder

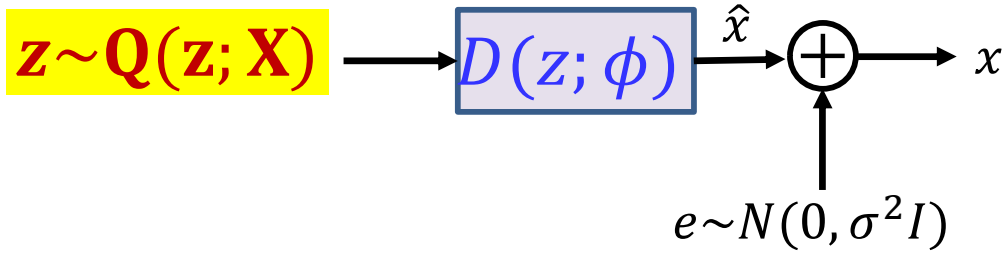
$x = D(z;\phi) + e$



$\mathrm{P}(x|z) = N(x; D(z;\phi), C)$

Training the encoder

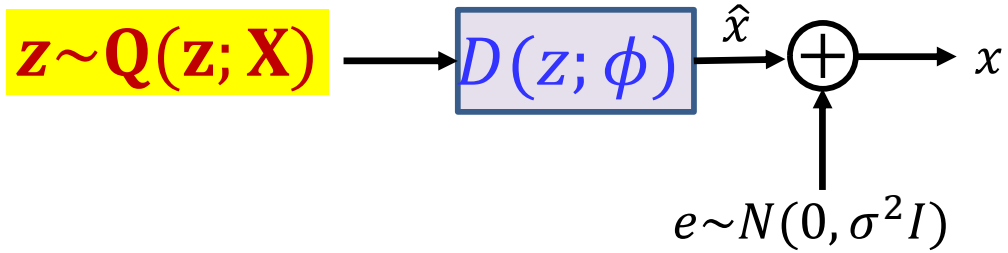
$x = D(z;\phi) + e$



$\mathrm{P}(x|z) = N(x; D(z;\phi), \sigma^2 I)$

Training the encoder

$$x = D(z;\phi) + e$$

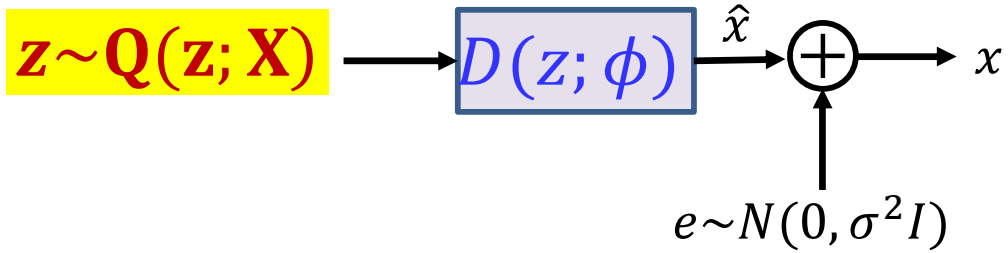


$$\mathrm{P}(x|z) = N(x; D(z;\phi), \sigma^2 I)$$

$$-\frac{d}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\|x - D(z;\phi)\|^2$$

Training the encoder

$$x = D(z;\phi) + e$$

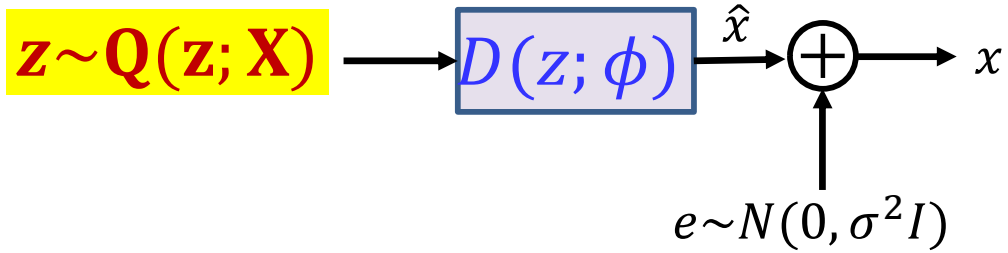


$$\mathrm{P}(x|z) = N(x; D(z;\phi), \sigma^2 I)$$

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \sum_{(x,z)} -\frac{d}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\|x - D(z;\phi)\|^2$$

Training the encoder

$$x = D(z;\phi) + e$$



$$\mathrm{P}(x|z) = N(x; D(z;\phi), \sigma^2 I)$$

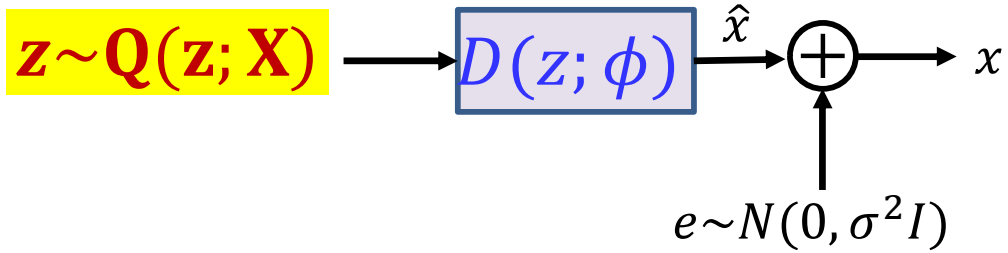
$$\theta^* = \underset{\theta}{\operatorname{argmax}} \sum_{(x,z)} -\frac{d}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\|x - D(z;\phi)\|^2$$

- We can learn the parameters using backpropagation, which minimizes the following loss

$$L(\theta, \phi, \sigma^2) = \sum_{(x,z)} d\log(\sigma^2) - \frac{1}{\sigma^2}\|x - D(z;\phi)\|^2$$

Training the encoder

$$x = D(z;\phi) + e$$



$$\mathrm{P}(x|z) = N(x; D(z;\phi), \sigma^2 I)$$

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \sum_{(x,z)} -\frac{d}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\|x - D(z;\phi)\|^2$$

- We can learn the parameters using backpropagation, which minimizes the following loss

$$L(\theta, \phi, \sigma^2) = \sum_{(x,z)} d\log(\sigma^2) - \frac{1}{\sigma^2}\|x - D(z;\phi)\|^2$$

- Must minimize this with respect to θ

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, L(\theta, \phi, \sigma^2)$$

Sampling z

- *For each training input x, z is obtained as a sample from $N(z; \mu(x;\theta), \Sigma(x;\theta))$*

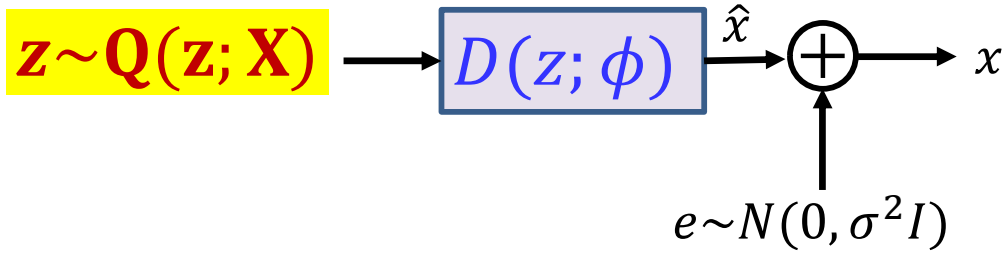
- We use a standard "reparametrization" step to sample z
  - Draw K-dimensional vector ε from $N(0, I)$
  - Compute $z = \mu(x;\theta) + \Sigma(x;\theta)^{0.5}\varepsilon$

Remember this one $\quad \nabla_\theta z = \nabla_\theta \mu(x;\theta) + diag(\varepsilon)\nabla_\theta \Sigma(x;\theta)^{0.5}$

This will be specific to x and to the *specific sample of* z for that x (via ε)

Training the encoder

$x = D(z;\phi) + e$



$\mathrm{P}(x|z) = N(x; D(z;\phi), \sigma^2 I)$

$$L(\theta, \phi, \sigma^2) = d \log \sigma^2 + \sum_{(x,z)} \frac{1}{\sigma^2} \|x - D(z;\phi)\|^2$$

- $z = \mu(x;\theta) + \Sigma(x;\theta)^{0.5}\varepsilon$
- ε is sampled from $N(0, I)$

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, L(\theta, \phi, \sigma^2)$$

- The derivative of $L(\theta, \phi, \sigma^2)$ with respect to θ can be computed using the chain rule $\nabla_\theta L(\theta, \phi, \sigma^2) = \sum_{(x,z)} \nabla_z L(\theta, \phi, \sigma^2)\, \nabla_\theta z$
  - For use in backpropagation

The variational autoencoder

- The *encoder* computes the *distribution* $P(z|x)$ for any x
- The *decoder* tries to convert a randomly sampled z from $P(z|x)$ to x

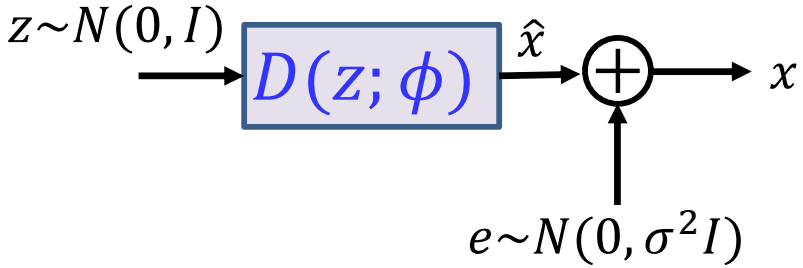
- **Training the encoder**:
  - Estimate $\boldsymbol{\theta}$ to make the zs that can be encoded to x more probable

- **Training the decoder**:
  - Estimate $\boldsymbol{\phi}$ to make the noise between $\hat{x} = D(z)$ and x more probable

- Constraint on z: Make $P(z|x)$ as close to the standard Gaussian as possible

Training the decoder

$$x = D(z;\phi) + e$$



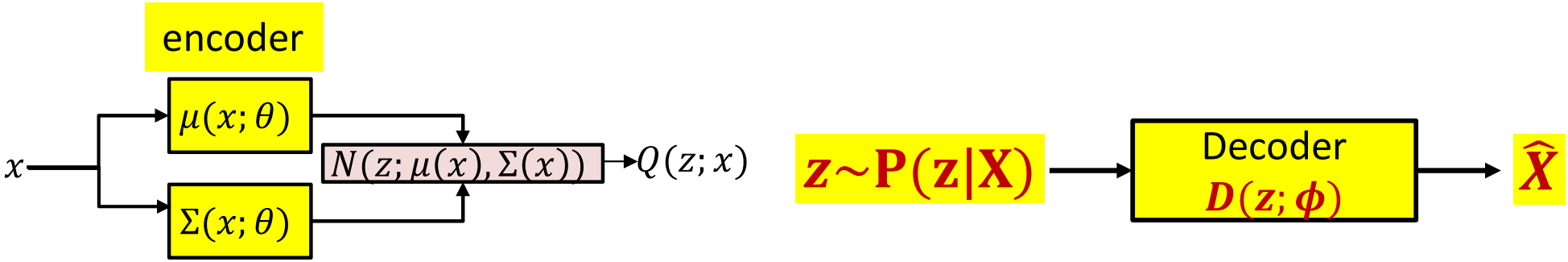
$$\mathrm{P}(x|z) = N(x; D(z;\phi), \sigma^2 I)$$

$$L(\theta, \phi, \sigma^2) = d \log \sigma^2 + \sum_{(x,z)} \frac{1}{\sigma^2} \|x - D(z;\phi)\|^2$$

$$\phi^*, {\sigma^2}^* = \underset{\phi,\sigma^2}{\operatorname{argmin}}\, L(\theta, \phi, \sigma^2)$$

- The derivative of this w.r.t $\phi$and σ^2 is trivially computed for backprop

The variational autoencoder



- The *encoder* computes the *distribution* $P(z|x)$ for any x
- The *decoder* tries to convert a randomly sampled z from $P(z|x)$ to x

- **Training the encoder**:
  - Estimate theta to make the zs that can be encoded to x more probable

- **Training the decoder**:
  - Estimate $\boldsymbol{\phi}$ to make the noise between $\hat{x} = D(z)$ and x more probable

- Constraint on z: Make $P(z|x)$ as close to the standard Gaussian as possible

The constraint on P(z)

- The KL between $Q(z;x) = N(z;\mu(x;\theta),\Sigma(x;\theta))$ and the standard Gaussian $N(z;0,I)$ works out to

$$KL(Q(z,x;\theta),N(z;0,I)) = \frac{1}{2}\Big(tr\big(\Sigma(x;\theta)\big) + \mu(x;\theta)^T(\mu(x;\theta) - d - \log|\Sigma(x;\theta)|)\Big)$$

  - This is a function of encoder parameters θ

The constraint on P(z)

- The KL between $Q(z;x) = N(z;\mu(x;\theta),\Sigma(x;\theta))$ and the standard Gaussian $N(z;0,I)$ works out to

$$KL(Q(z,x;\theta),N(z;0,I)) = \frac{1}{2}\Big(tr\big(\Sigma(x;\theta)\big) + \mu(x;\theta)^T(\mu(x;\theta) - d - \log|\Sigma(x;\theta)|)\Big)$$

  - This is a function of encoder parameters θ
- The overall loss thus becomes:

$$L_{VAE}(\theta,\varphi,\sigma^2)$$

$$= \sum_{x\in X}\Big(tr\big(\Sigma(x;\theta)\big) + \mu(x;\theta)^T(\mu(x;\theta) - d - \log|\Sigma(x;\theta)|)\Big) + \frac{1}{\sigma^2}\sum_{(x,z)\in[X,Z]}\|(x - D(z;\phi))\|^2$$

$$+ d\log\sigma^2$$

- This must be minimized to train the VAE
  - The derivatives of all terms w.r.t. are easily computed using backprop

The complete training pipeline

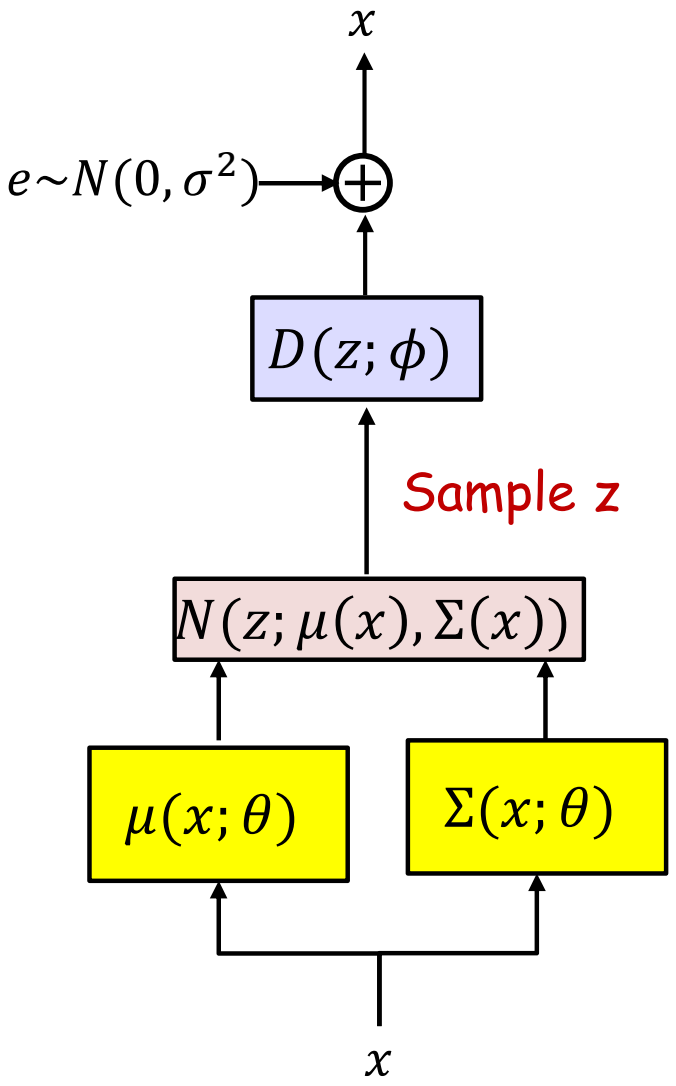
- Initialize θ and φ
- Iterate:
  - Sample $z_{x,\varepsilon}$ from $N(z;\mu(x;\varphi),\Sigma(x;\varphi))$ for each training instance
  - Reestimate $\theta, \varphi, \sigma^2$ from

$$L_{VAE}(\theta,\phi,\sigma^2)$$

$$= \sum_{x\in X}\Big(tr\big(\Sigma(x;\theta)\big)$$

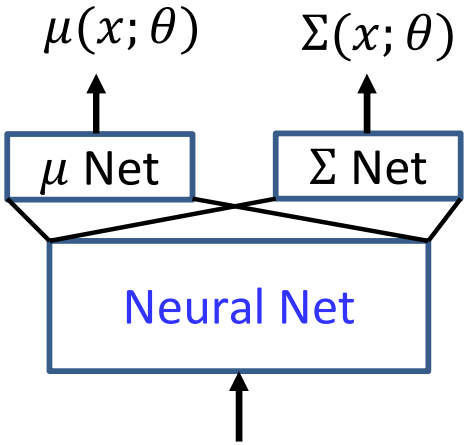
$$+ \mu(x;\theta)^T(\mu(x;\theta) - d - \log|\Sigma(x;\theta)|)\Big)$$

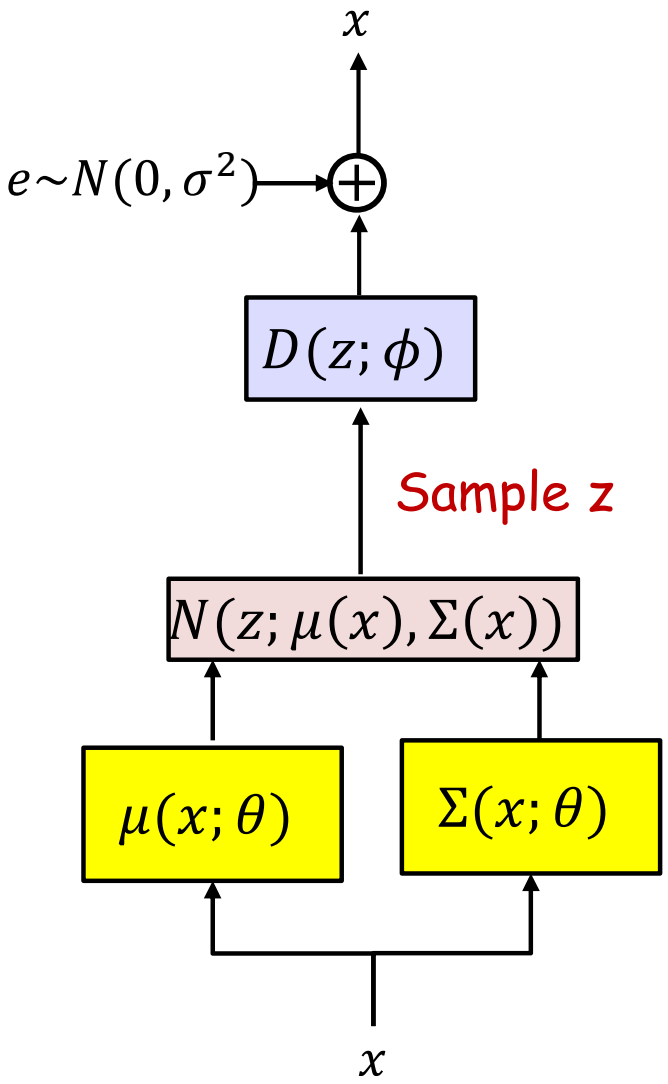
$$+ \frac{1}{\sigma^2}\sum_{(x,z)\in[X,Z]} \|(x - D(z;\phi))\|^2 + d\log\sigma^2$$



But where are the neural nets?

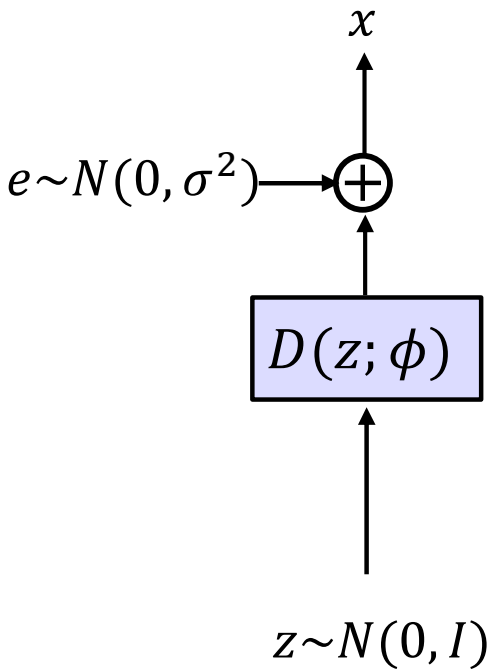
- $D(z;\theta)$ is a neural network
- $\mu(x;\theta)$ and $\Sigma(x;\theta)$ are generally modelled by a *common* network with two outputs
  - The combined parameters of the network are θ





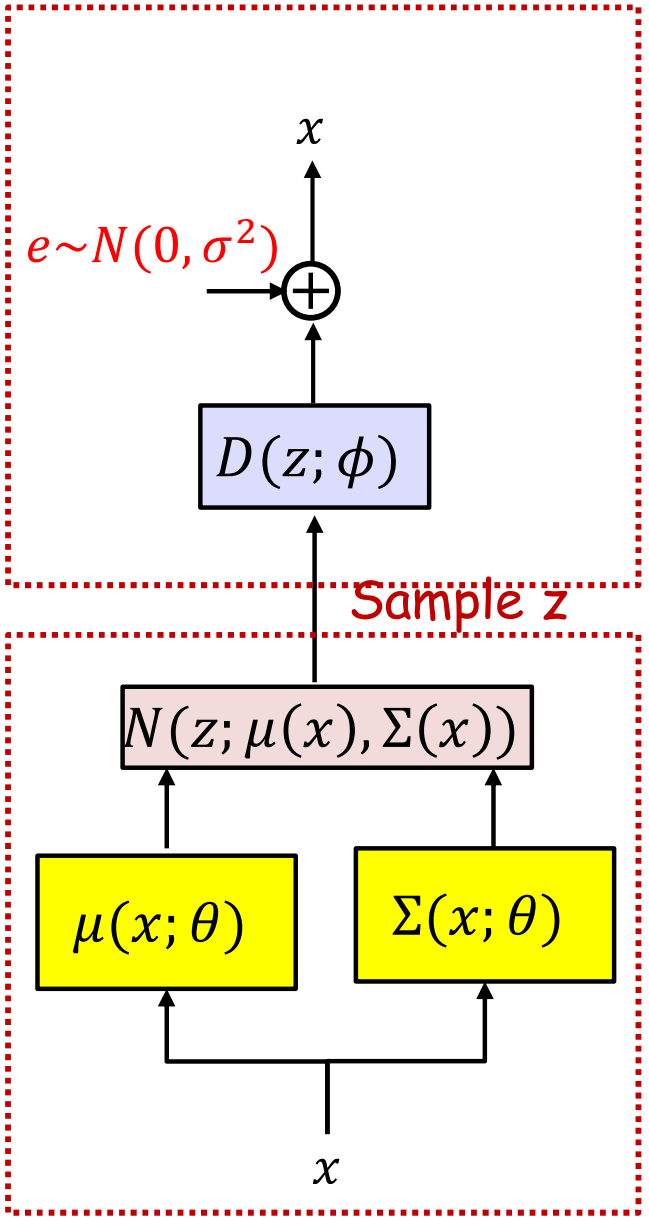
The VAE for generation

- Once trained the encoder can be discarded
- The rest of the network gives us a *generative* model for x
- Generating data using this part of the model should (ideally) give us data similar to the training data

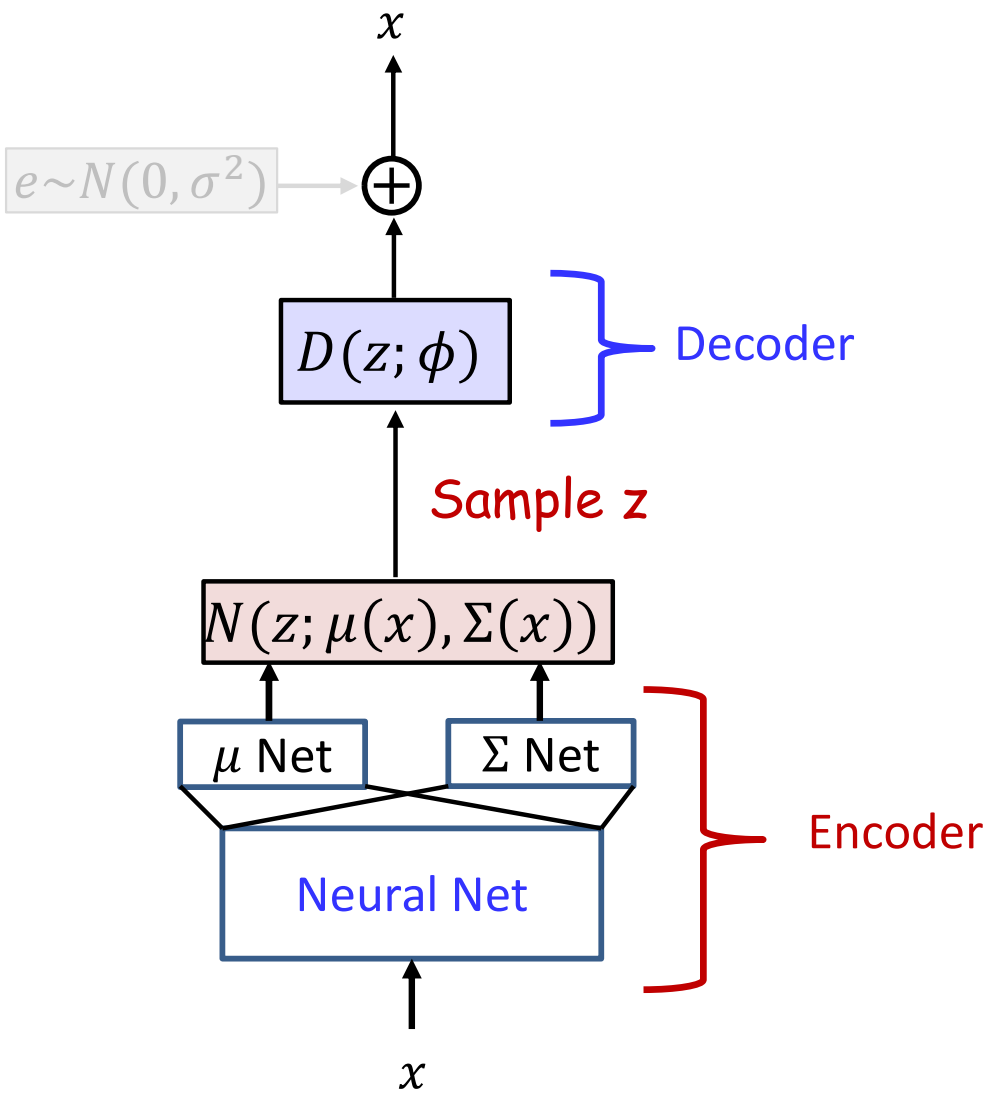


Recap: The VAE

- An autoencoder with statistical constraints on the hidden representation
  - The encoder is a statistical model that computes the parameters of a Gaussian
  - The decoder converts samples from the Gaussian back to the input
- The *decoder* is a generative model that, when excited by standard Gaussian inputs, generates samples similar to the training data



The Variational AutoEncoder



The decoder is the actual generative model.

The encoder is primarily needed for training.
It can also be used to generate the (approximate) distribution of latent space representations conditioned on specific inputs (much like a regular autoencoder).

z is a *latent-space* representation of the data.

$\mu(x)$ can also be used as a *expected latent* representation of x.

Poll 4

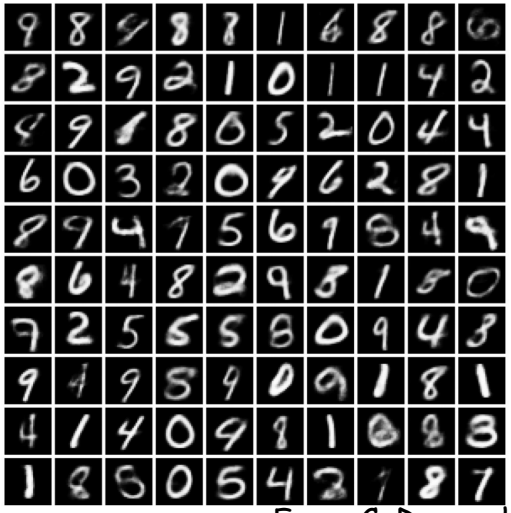
Poll 4

- A variational autoencoder is a generative extension of autoencoders that models probability distributions
  - **True**
  - False

- Select the true statements
  - **The standard VAE assumes a latent representation that has an isotropic Gaussian PDF**
  - **The VAE model requires addition of Gaussian noise to the output of a regularized AE in order to permit the output to fill space beyond a lower-dimensional manifold**
  - **The decoder of the VAE can be used to generate samples from the distribution of the data it is trained on, if the input to the decoder is drawn from a standard Gaussian**

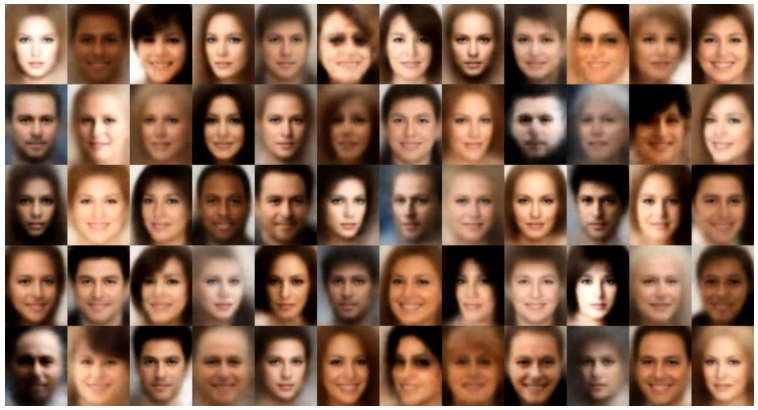
VAE examples

- Top: VAE trained on MNIST and used to generate new data



From C. Doersch

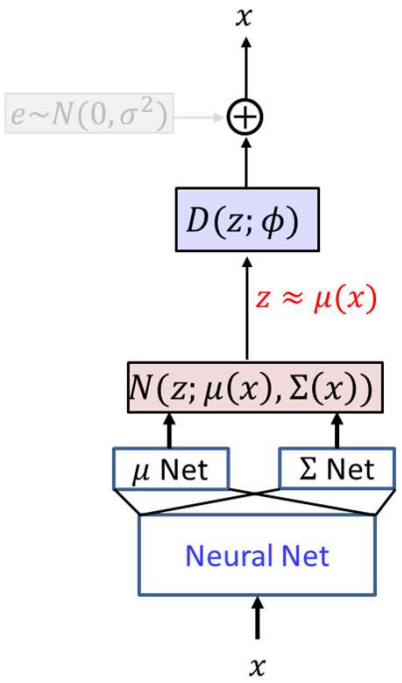
- Below: VAE trained on faces, and used to generate new data



From J. Rocco

VAE and latent spaces

- The latent space z often captures underlying structure in the data x in a smooth manner
  - Varying z continuously in different directions can result in plausible variations in the drawn output



- Reproductions of an input x can be manipulated by wiggling z around its expected value $\mu(x)$

VAE conclusions

- Simple statistical extension of the autoencoder
- Excellent generative models for the distribution of data $P(x)$
  - Various extensions such as Conditional VAEs, which model *conditional* distributions, such as $P(x|y)$
    - Straight-forward extension where the conditioning variable y is an additional input to the encoder and decoder
- Read the literature on the topic, it is vast