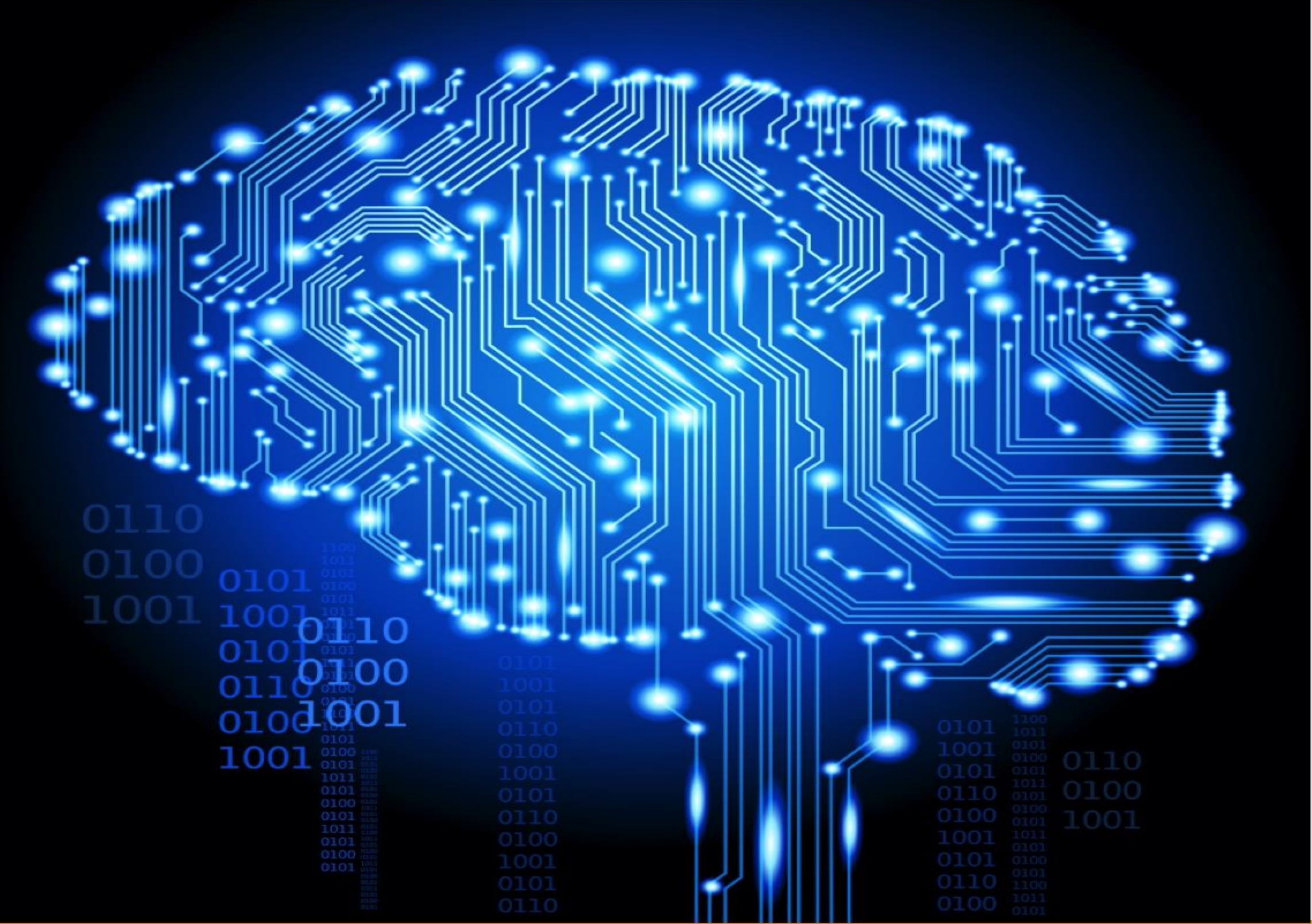


Lecture 9: Unsupervised, Generative & Adversarial Networks

Deep Learning @ UvA

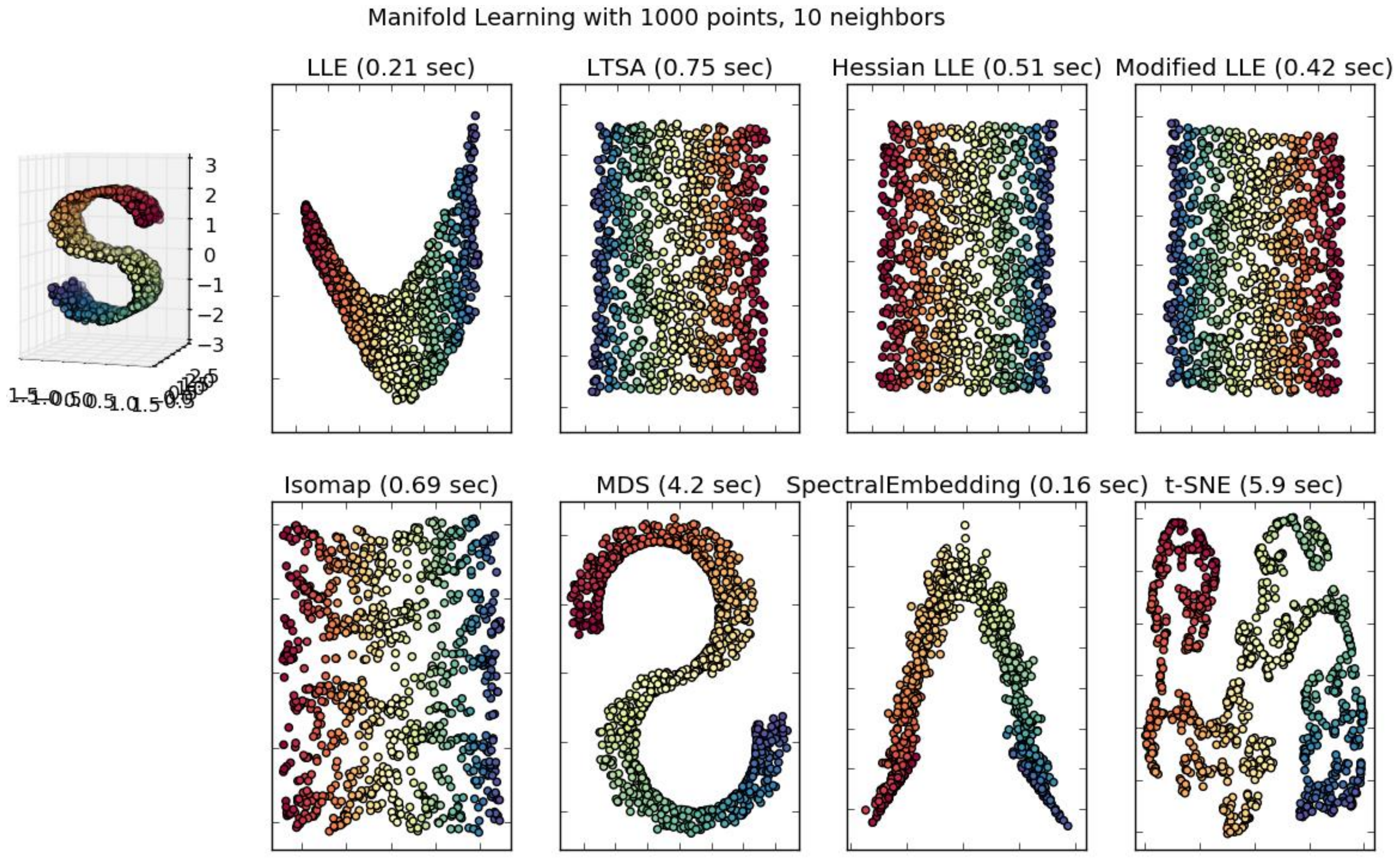
Previous Lecture

- Recurrent Neural Networks (RNN) for sequences
- Backpropagation Through Time
- Vanishing and Exploding Gradients and Remedies
- RNNs using Long Short-Term Memory (LSTM)
- Applications of Recurrent Neural Networks

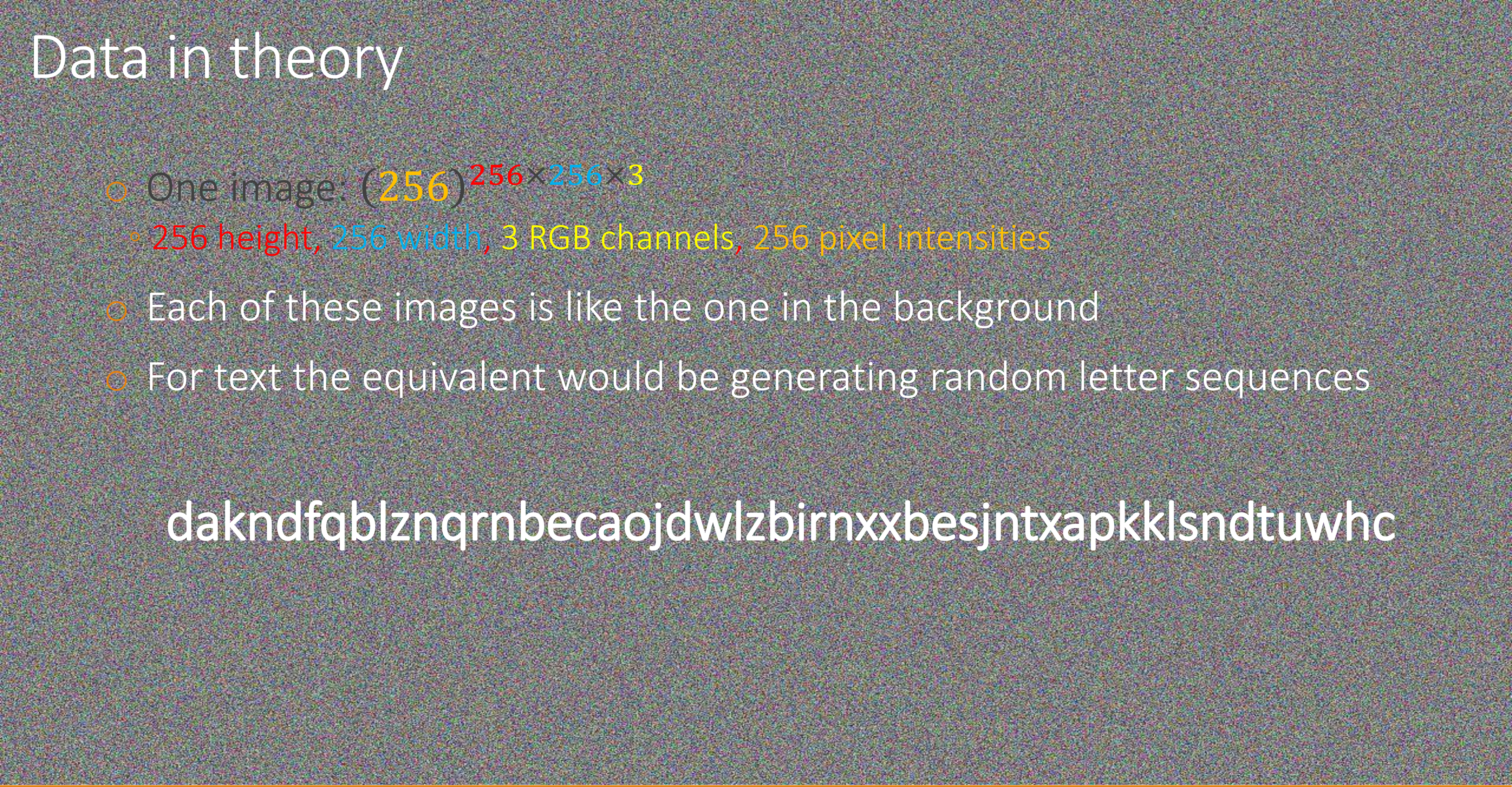
Lecture Overview

- Latent space data manifolds
- Autoencoders and Variational Autoencoders
- Boltzmann Machines
- Adversarial Networks
- Self-Supervised Networks

The manifold hypothesis



Data in theory



- One image: $(256)^{256 \times 256 \times 3}$
  - 256 height, 256 width, 3 RGB channels, 256 pixel intensities
- Each of these images is like the one in the background
- For text the equivalent would be generating random letter sequences

daknd fqblznqrnbecaojdwlzbirnxxbesjntxapkklsndtuwhc

Data in theory

- One image: $(256)^{256 \times 256 \times 3}$
  - 256 height, 256 width, 3 RGB channels, 256 pixel intensities
- Each of these images is like the one in the background
- For text the equivalent would be generating random letter sequences

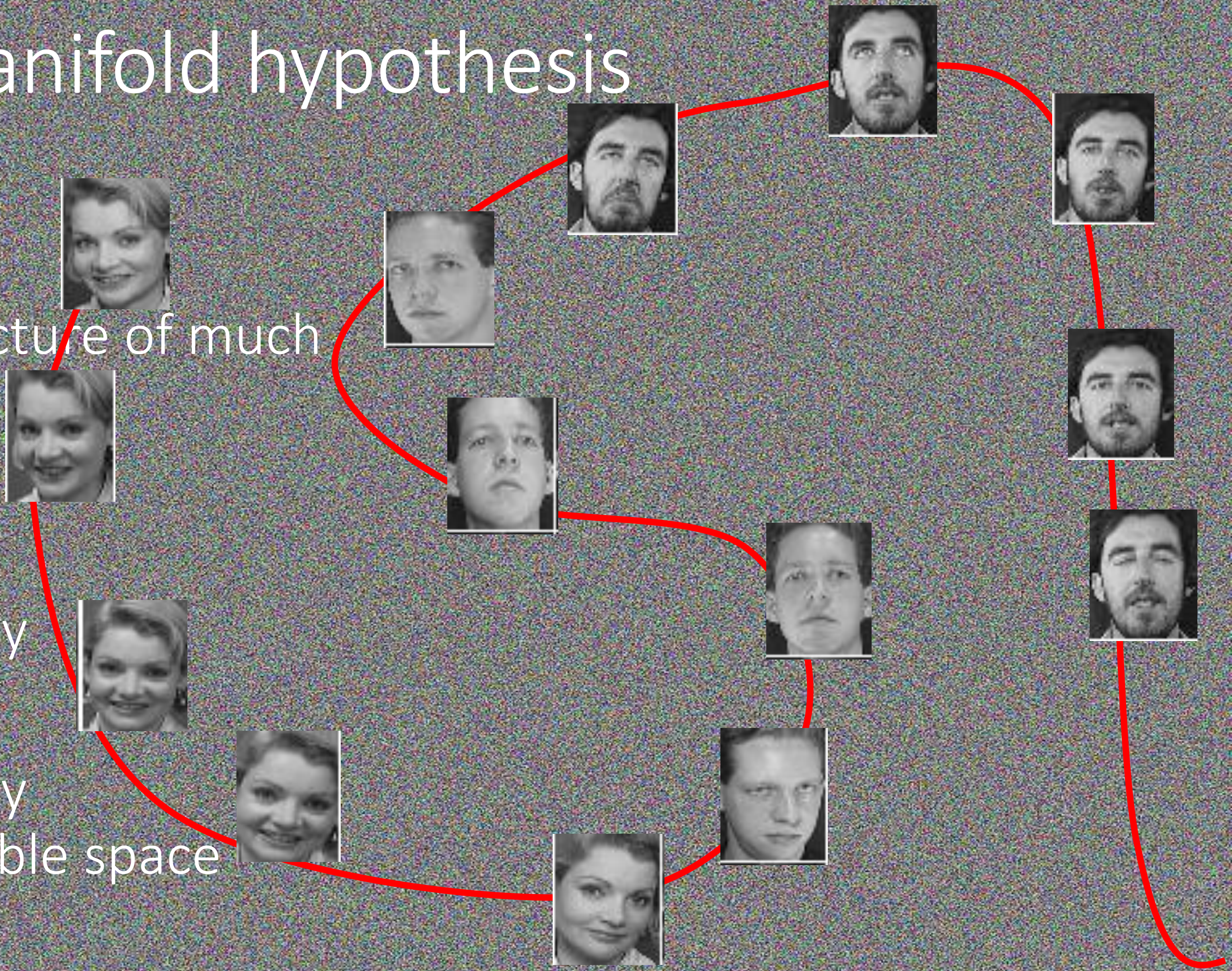
qgkhlkjijxskmbisuwephrhudskneyaeajdzhowieyqwhfnago

Data in theory

- One image: $(256)^{256 \times 256 \times 3}$
  - 256 height, 256 width, 3 RGB channels, 256 pixel intensities
- Each of these images is like the one in the background
- For text the equivalent would be generating random letter sequences

tpxvdcxwgebouyvqaekqzgwvqfuakuhodsapxzbfsizgobtpjb

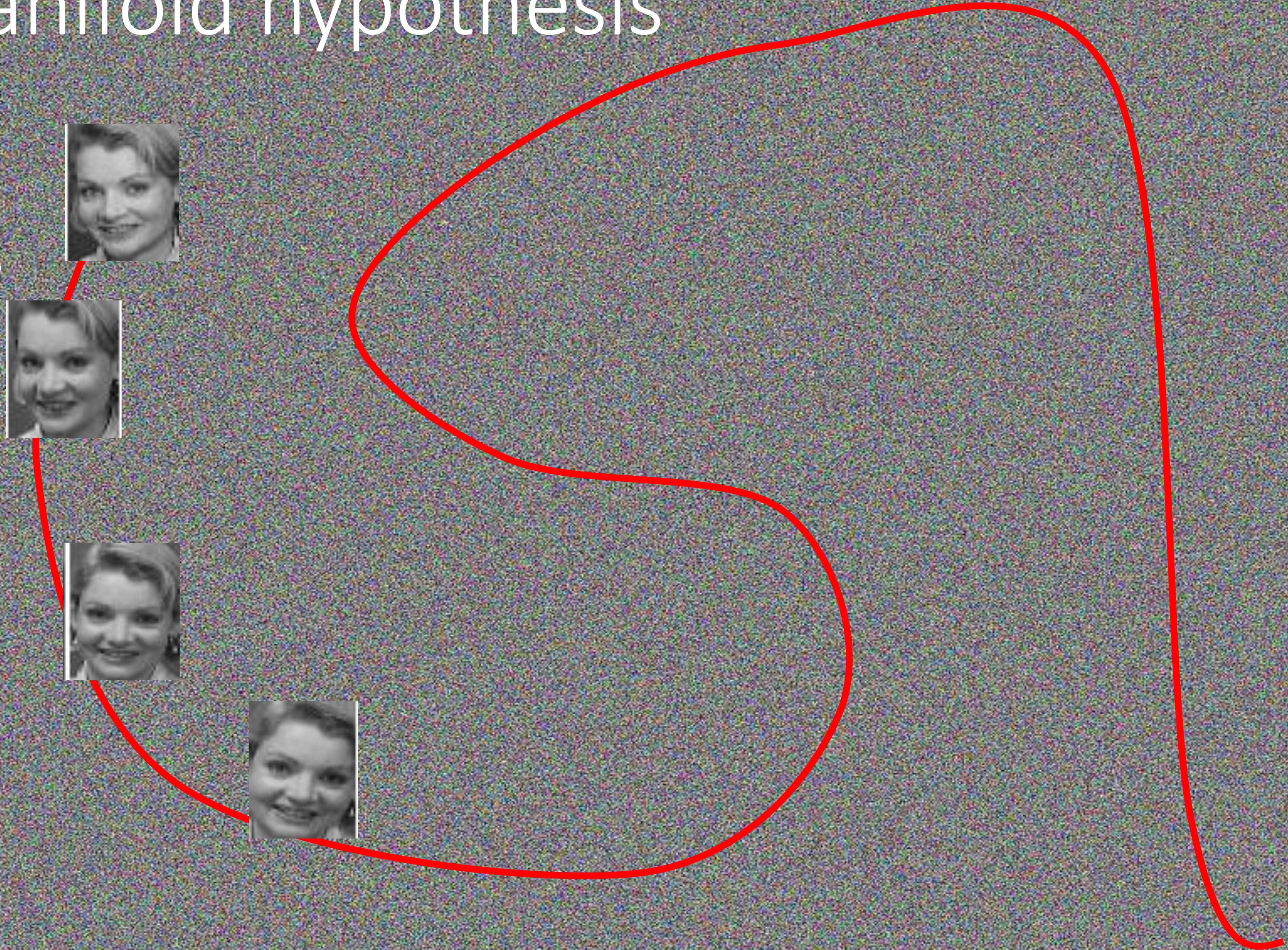
Data in reality: The manifold hypothesis



- Data live in manifolds
- A manifold is a latent structure of much lower dimensionality
  $\mathbf{dim}_{manifold} \ll \mathbf{dim}_{data}$
- Nobody "forces" the data to be on the manifold, they just are
- The manifold occupies only a tiny fraction of the possible space

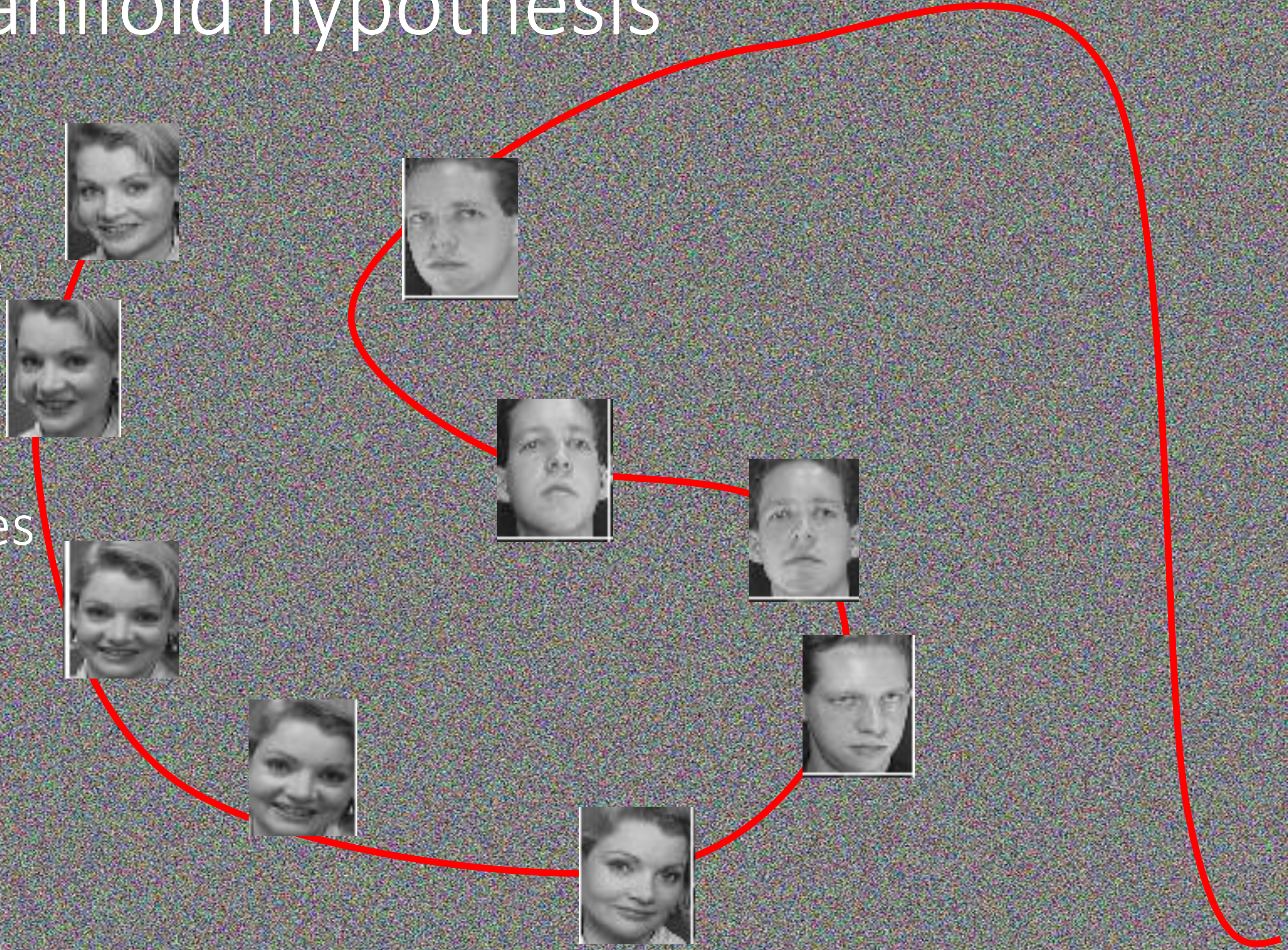
Data in reality: The manifold hypothesis

- The trajectory defines transformations/variances
- Rotation
  - E.g. the faces turns

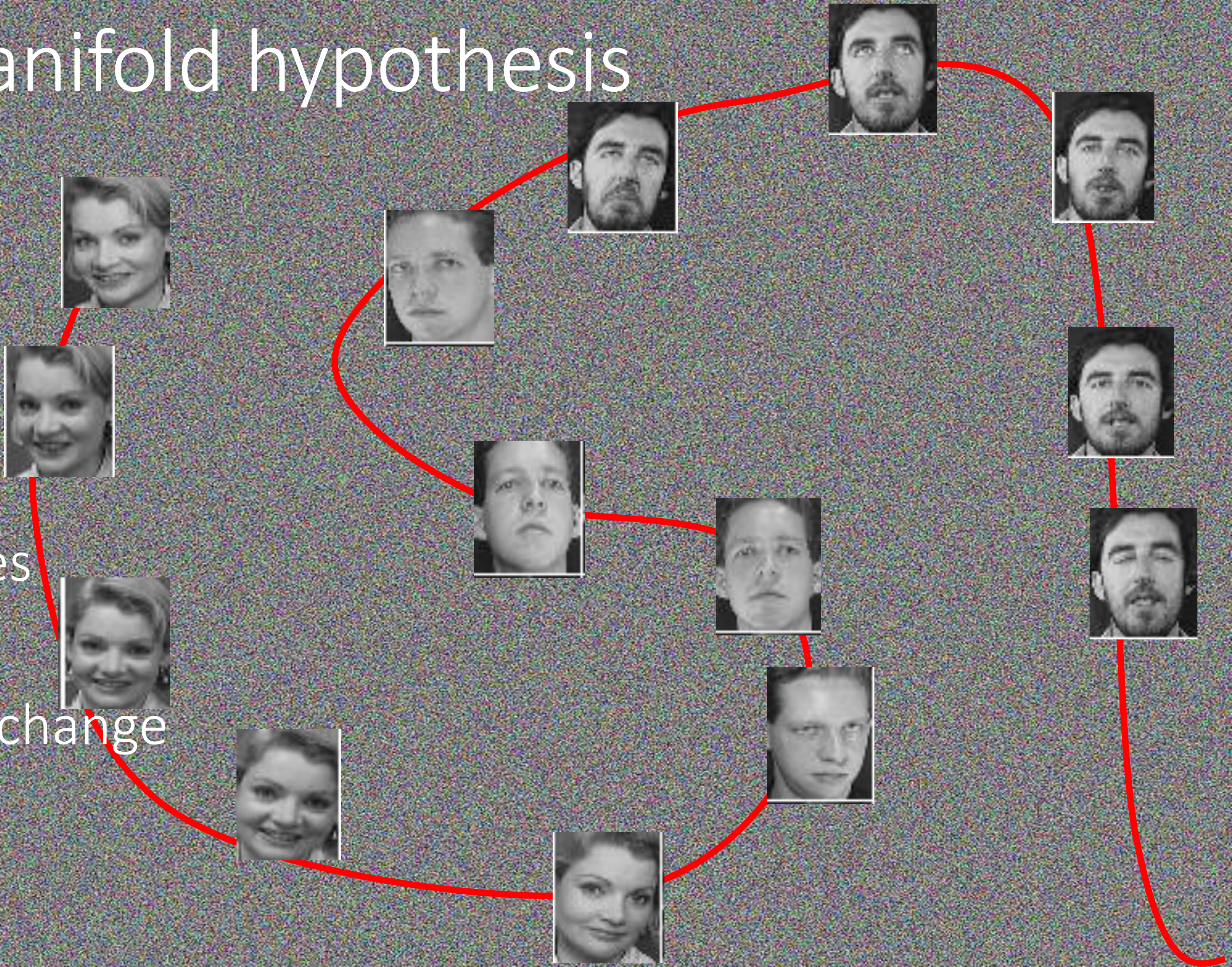


Data in reality: The manifold hypothesis

- The trajectory defines transformations/variances
- Rotation
  - E.g. the faces turns
- Global appearance changes
  - E.g., Different face



Data in reality: The manifold hypothesis



- The trajectory defines transformations/variances
- Rotation
  - E.g. the faces turns
- Global appearance changes
  - E.g., Different face
- Or even local appearance change
  - E.g., eyes open/closed

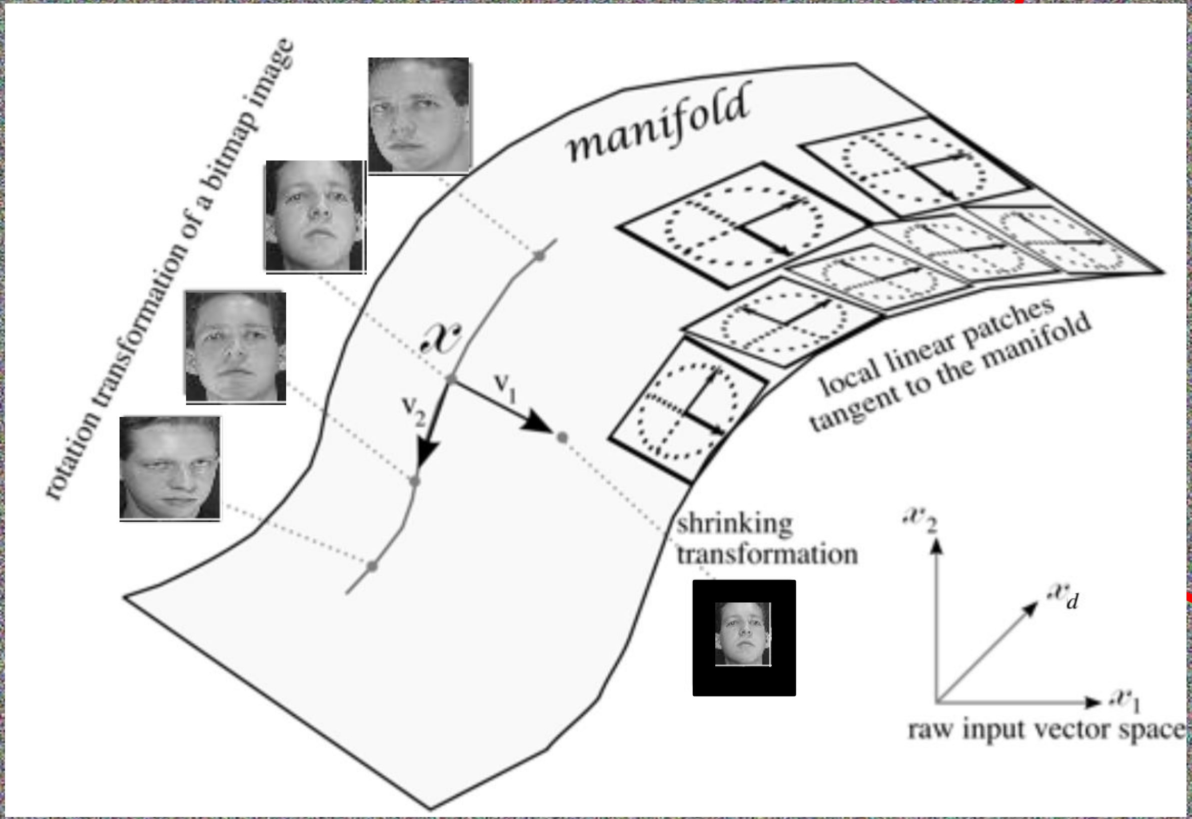
Data in reality: The manifold hypothesis

- Each image is a either a set of coordinates in the full data space

$$\text{Image}=\begin{bmatrix} 0.1 \\ -0.2 \\ 0.8 \\ 0.3 \\ -0.5 \\ -0.3 \\ 0.8 \\ 0.1 \\ -0.4 \end{bmatrix}$$

Data in reality: The manifold hypothesis

- Or a smaller set of intrinsic manifold coordinates



rotation transformation of a bitmap image
manifold
x
v_1
v_2
local linear patches tangent to the manifold
shrinking transformation
x_2
x_d
x_1
raw input vector space

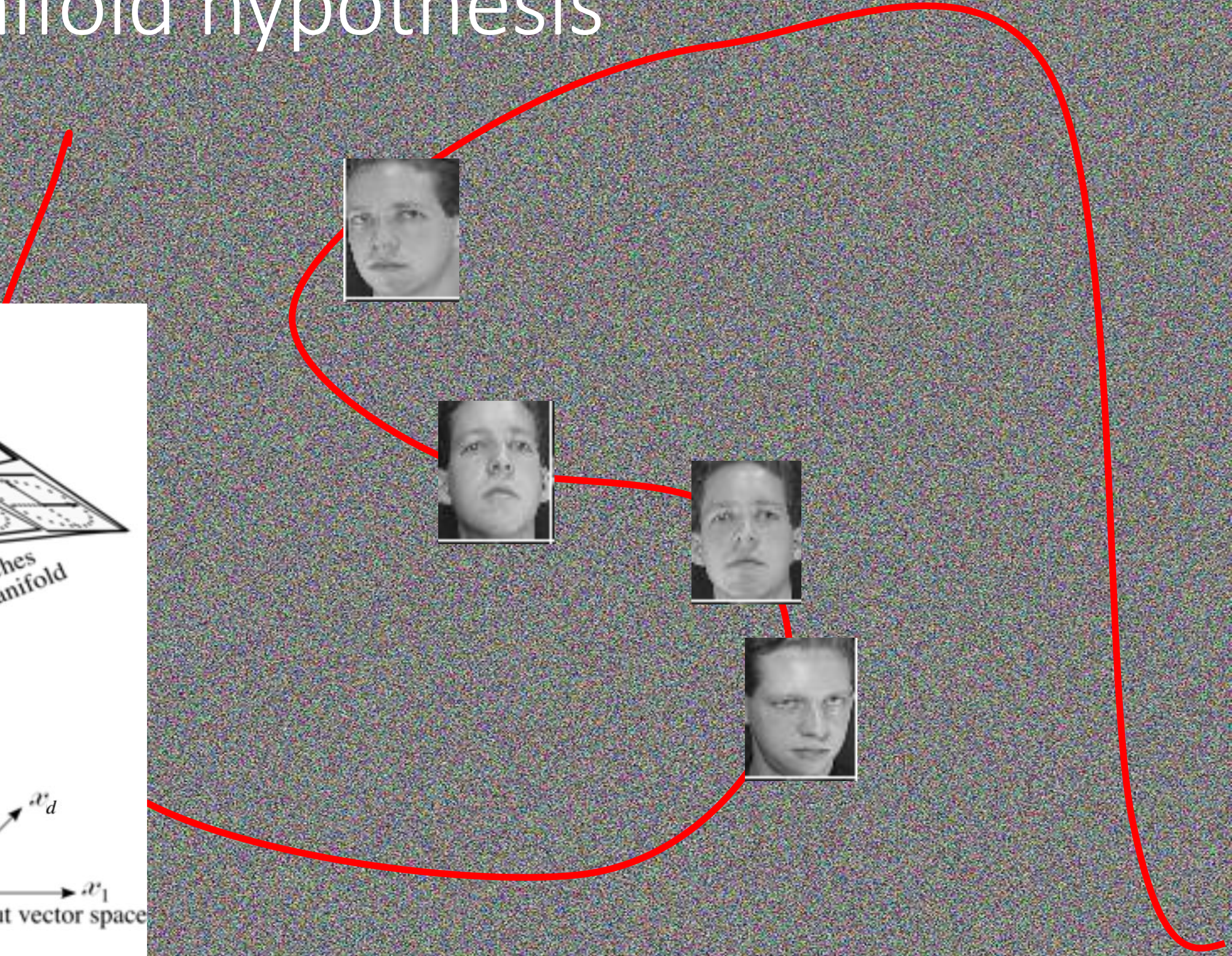
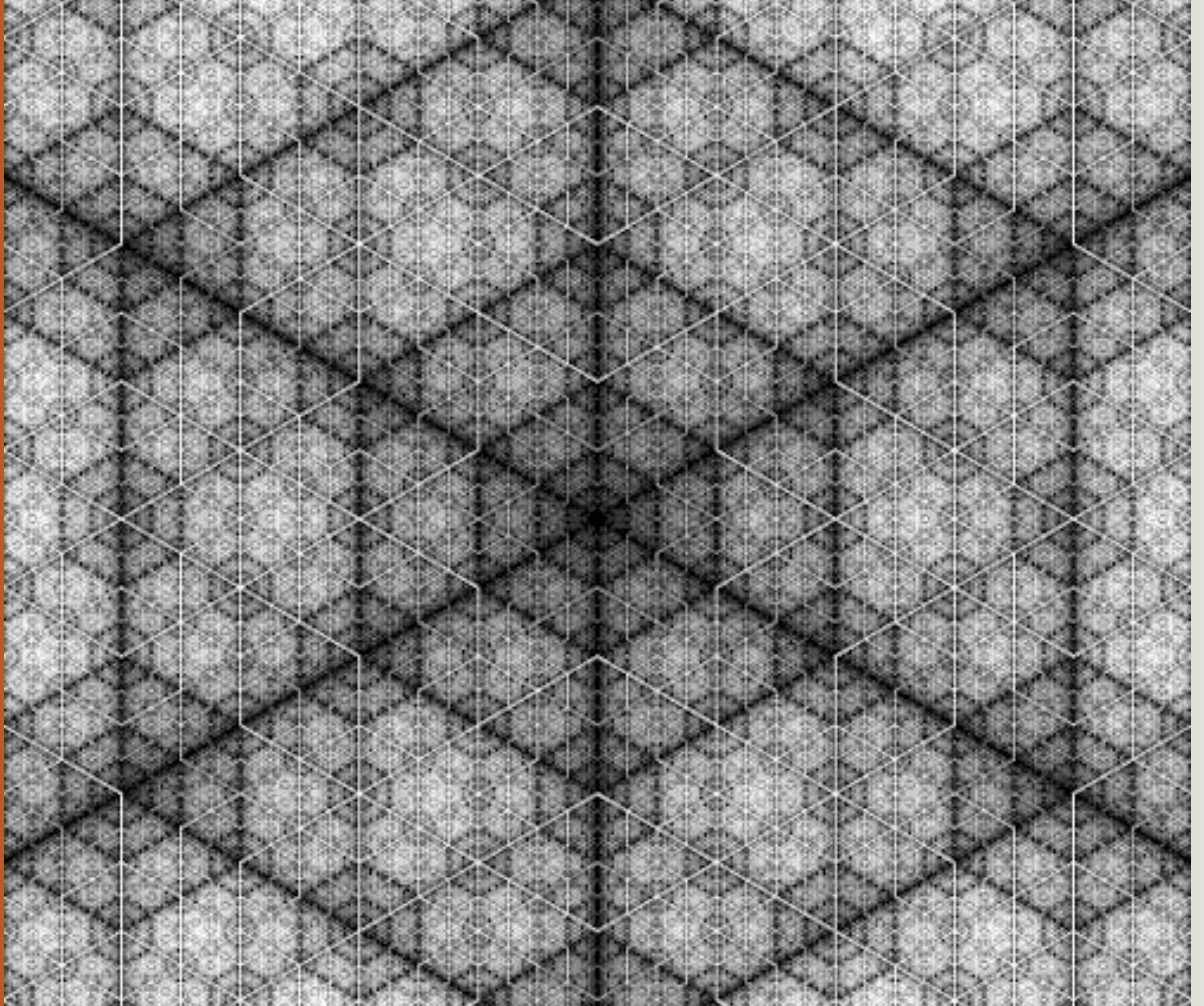


Image taken from Pascal Vincent, Deep Learning Summer School, 2015

So what?

- Manifold → Data distribution
- Learning the manifold → Learning data distribution and data variances
- How to learn the these variances automatically?
- Unsupervised and/or generative learning

Unsupervised and Generative Learning

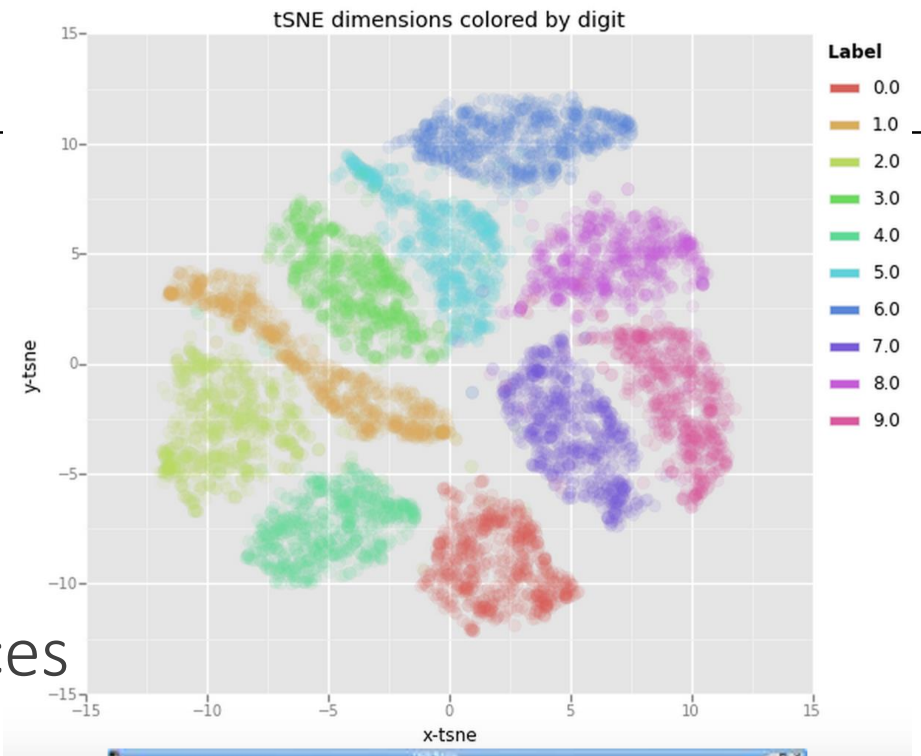


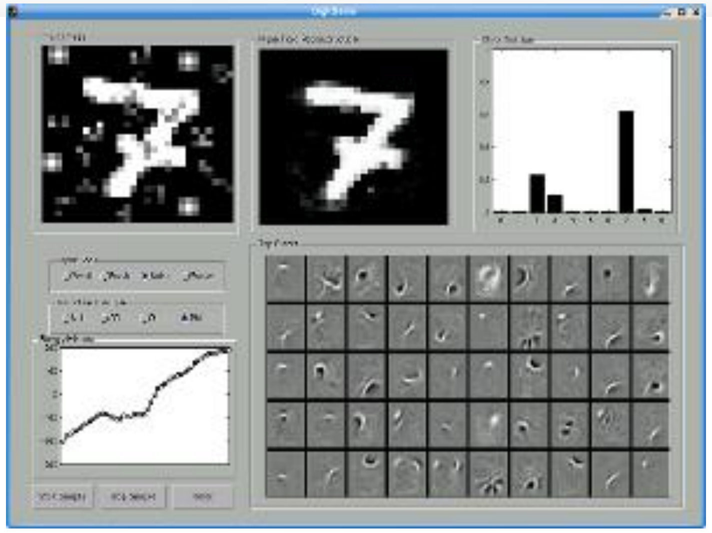
What is unsupervised learning?

- Latent space manifolds
- Autoencoders and Variational Autoencoders
- Boltzmann Machines
- Adversarial Networks
- Self-Supervised Networks

Why unsupervised learning?

- Much more unlabeled than labeled data
  - Large data → better models
  - Ideally not pay for annotations
- What is even the correct annotation for learning data distribution and/or data variances
- Discovering structure
  - What are the important features in the dataset





What are generative models?

- Latent space manifolds
- Autoencoders and Variational Autoencoders
- Boltzmann Machines
- Adversarial Networks
- Self-Supervised Networks

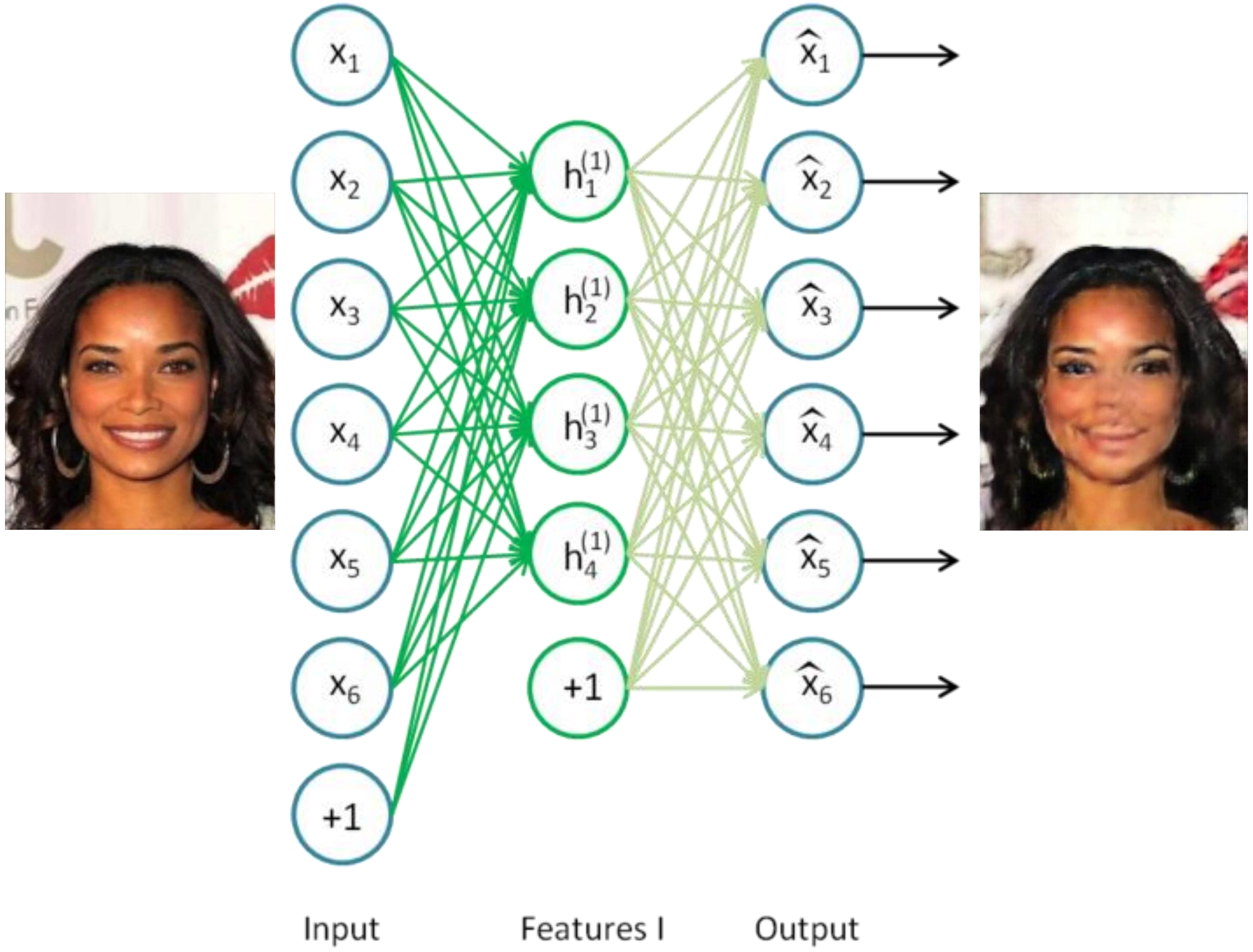
Why generative?

- Force models to go beyond surface statistical regularities of existing data
  - Go beyond direct association of observable inputs to observable outputs
  - Avoid silly predictions (adversarial examples)
- Understand the world
- Understand data variances, disentangle and recreate them
- Detect unexpected events in your data
- Pave the way towards reasoning and decision making

Unsupervised & generative learning of the manifold

- Autoencoders and Variational Autoencoders
- Boltzmann Machines
- Adversarial Networks
- Self-Supervised Networks

Autoencoders



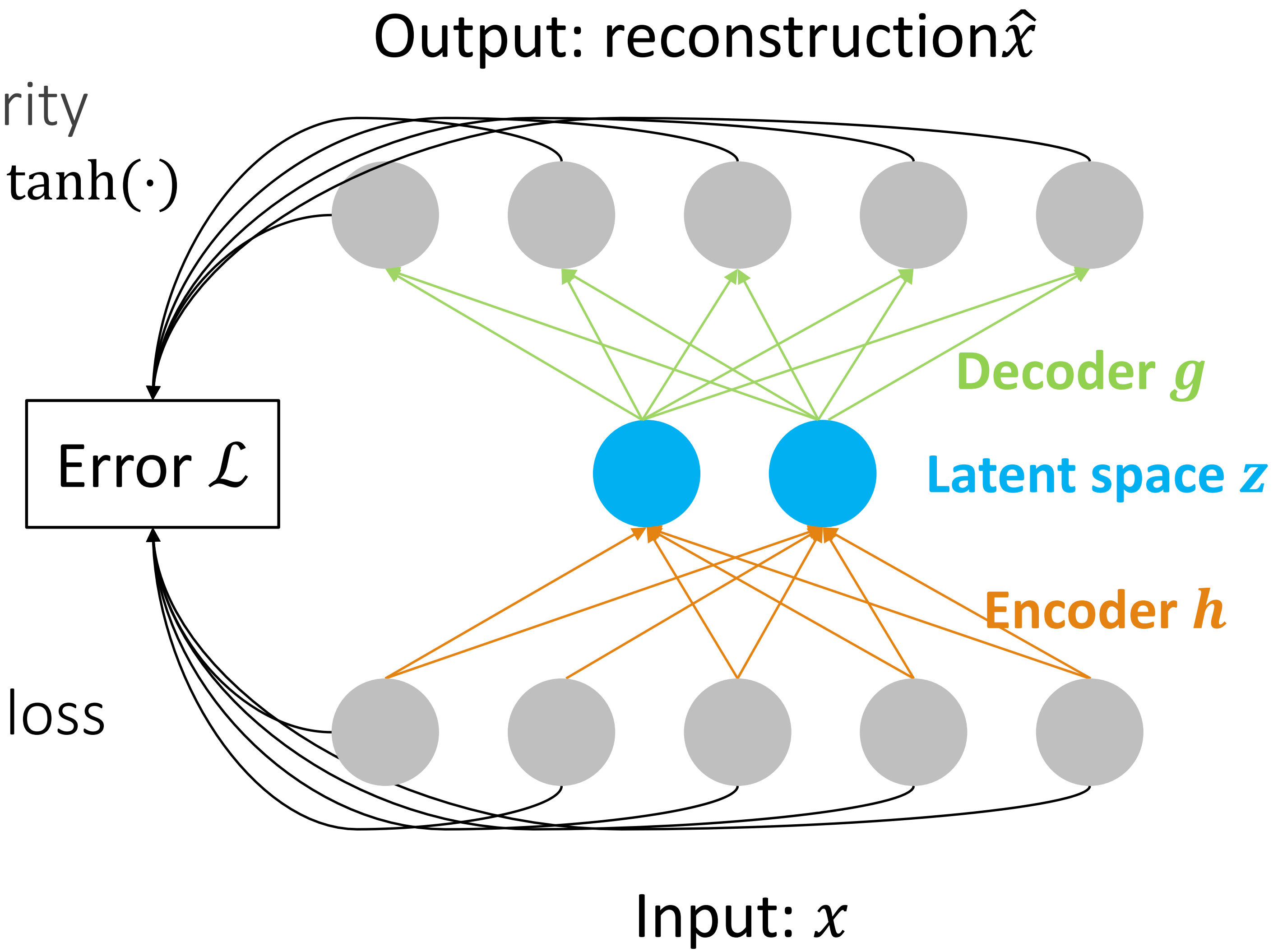
Standard Autoencoder

- z is usually after a non-linearity
  - E.g. $z = h(Wx + b), \quad h \equiv \sigma(\cdot), \tanh(\cdot)$
- Reconstruction error $\mathcal{L}$

$$\mathcal{L} = \sum_d \ell\left(x, g\big(h(x)\big)\right)$$

- Often error is the Euclidean loss

$$\ell = |x - \hat{x}|^2$$



Standard Autoencoder

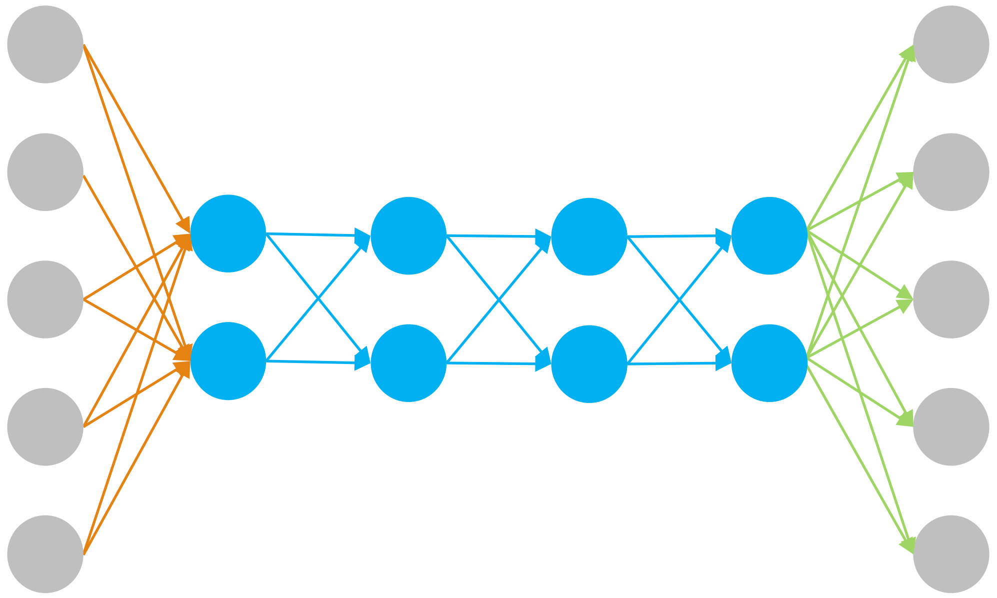
- The latent space should have fewer dimensions than input
  - **Undercomplete** representation
  - Bottleneck architecture
- Otherwise (overcomplete) autoencoder might learn the identity function

$$W \propto I \implies \tilde{x} = x \implies \mathcal{L} = 0$$

  - Assuming no regularization
  - Often in practice still works though
- Also, if $z = Wx + b$ (linear) autoencoder learns same subspace as PCA

Stacking Autoencoders

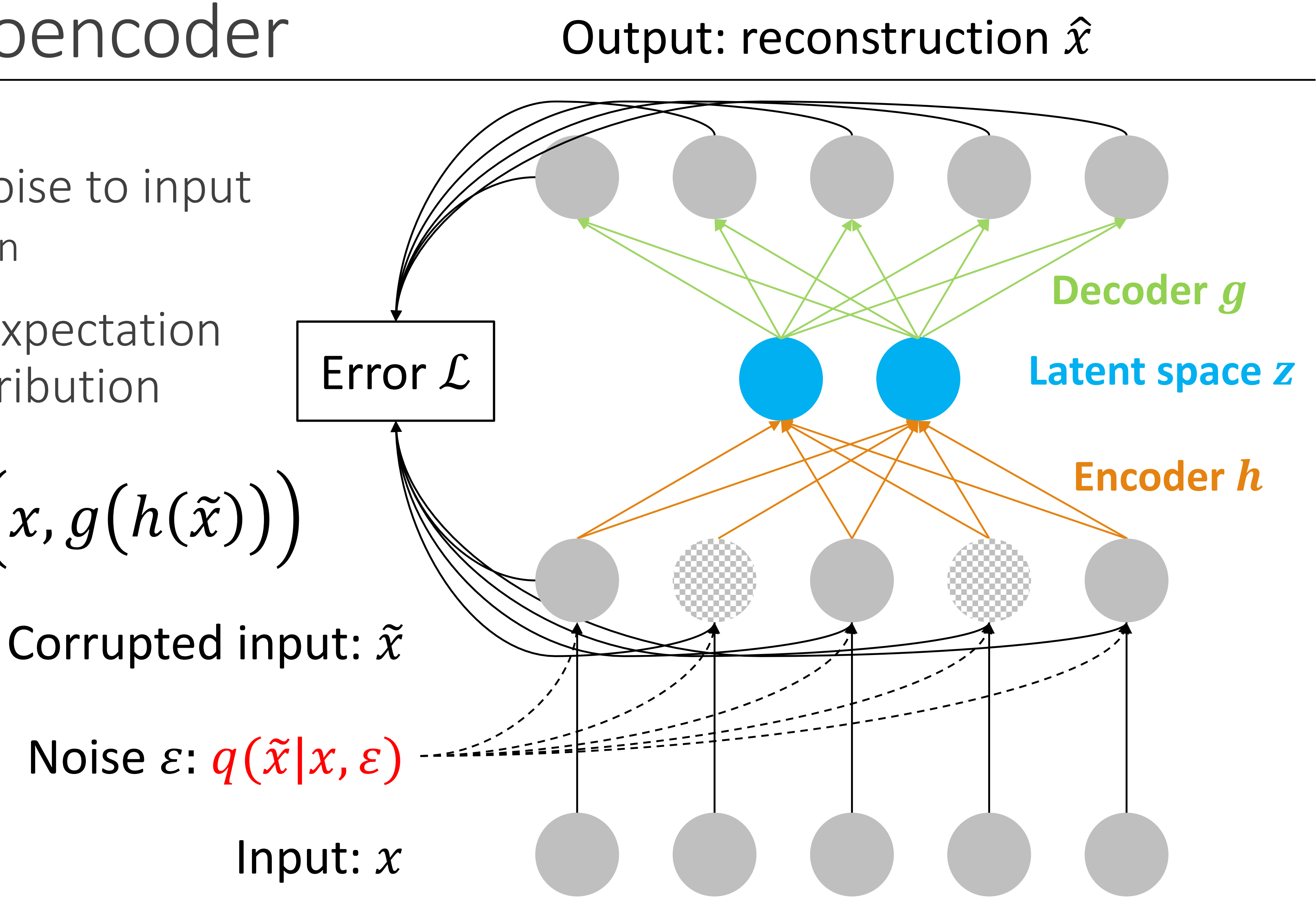
- Stacking layers on top of each other
- Bigger depth → higher level abstractions
- Slower training



Denoising Autoencoder

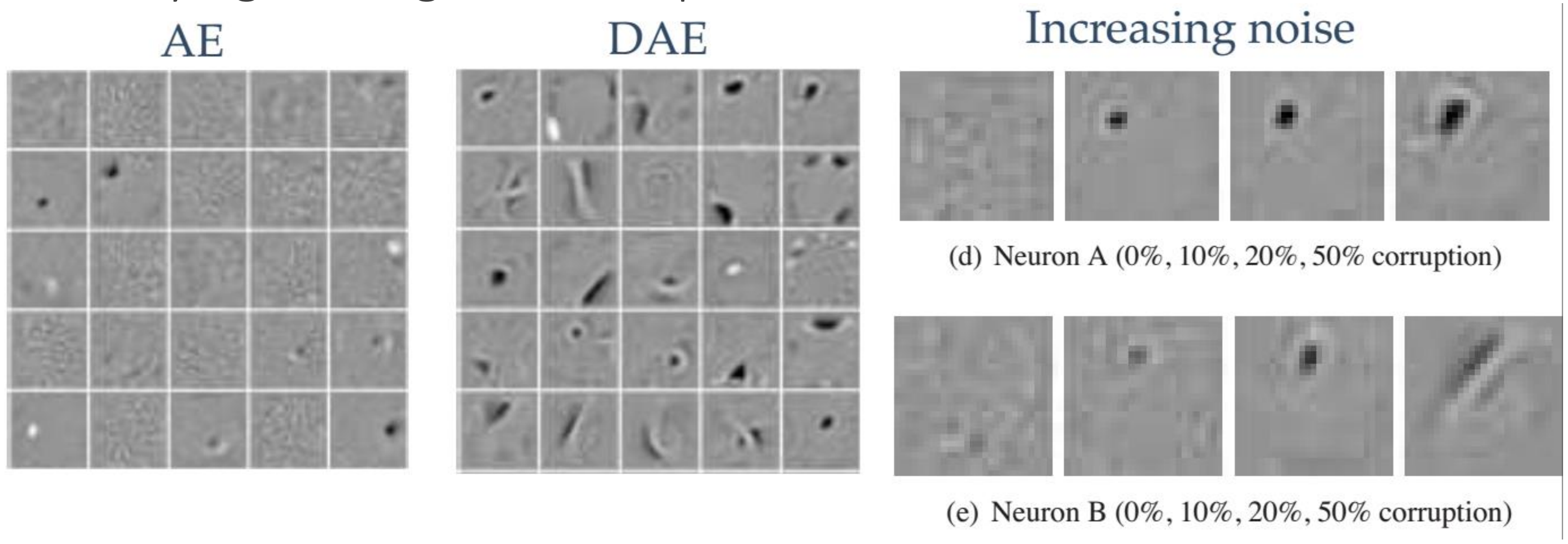
- Add random noise to input
  - Dropout, gaussian
- Loss includes expectation over noise distribution

$$\mathcal{L} = \sum_d \mathrm{E}_{q(\tilde{x}|x)} \ell\left(x, g\left(h(\tilde{x})\right)\right)$$


Output: reconstruction $\hat{x}$
Decoder g
Latent space z
Encoder h
Error $\mathcal{L}$
Corrupted input: $\tilde{x}$
Noise ε: $q(\tilde{x}|x, \varepsilon)$
Input: x

Denoising Autoencoder

- The network does not overlearn the data
  - Can even use overcomplete latent spaces
- Model forced to learn more intelligent, robust representations
  - Learn to ignore noise or trivial solutions(identity)
  - Focus on “underlying” data generation process


AE
DAE
Increasing noise
(d) Neuron A (0%, 10%, 20%, 50% corruption)
(e) Neuron B (0%, 10%, 20%, 50% corruption)

Stochastic Contractive Autoencoders

- Instead of invariant to input noise, invariant output representations
  - Similar input with different sampled noise → similar outputs
- Regularization term over noisy reconstructions

$$\mathcal{L} = \sum_d \ell\left(x, g(h(\tilde{x}))\right) + \lambda \mathrm{E}_{q(\tilde{x}|x)}[|h(x) - h(\tilde{x})|^2]$$

Reconstruction error

Regularization error

Stochastic Contractive Autoencoders

- Instead of invariant to input noise, invariant output representations
  - Similar input with different sampled noise → similar outputs
- Regularization term over noisy reconstructions
- Trivial solution?

$$\mathcal{L} = \sum_d \ell\left(x, g(h(\tilde{x}))\right) + \lambda \mathrm{E}_{q(\tilde{x}|x)}[|h(x) - h(\tilde{x})|^2]$$

Reconstruction error

Regularization error

Stochastic Contractive Autoencoders

- Instead of invariant to input noise, invariant output representations
  - Similar input with different sampled noise → similar outputs
- Regularization term over noisy reconstructions
- Trivial solution?
  - $h(x) = 0$
  - To avoid make sure the encoder and decoder weights are shared $W_d = W_e^T$

$$\mathcal{L} = \sum_d \ell\left(x, g(h(\tilde{x}))\right) + \lambda \mathrm{E}_{q(\tilde{x}|x)}[|h(x) - h(\tilde{x})|^2]$$

Reconstruction error Regularization error

Stochastic → Analytic Regularization

- Taylor series expansion about a point $y = a$

$$h(y) = h(a) + h'(a)(y-a) + 0.5\, h''(a)(y-a)^2 + \ \dots$$

- For $y = \tilde{x} = x + \varepsilon$, where $\varepsilon \sim N(0, \sigma^2 I)$ the (first order) Taylor

$$h(\tilde{x}) = h(x + \varepsilon) = h(x) + \frac{\partial \mathrm{h}}{\partial x}\varepsilon$$

$$\mathrm{E}_{q(\tilde{x}|x)}[|h(x) - h(\tilde{x})|^2] = \left\|\frac{\partial \mathrm{h}}{\partial x}\varepsilon\right\|^2 \propto \sigma^2 \left\|\frac{\partial \mathrm{h}}{\partial x}\right\|^2$$

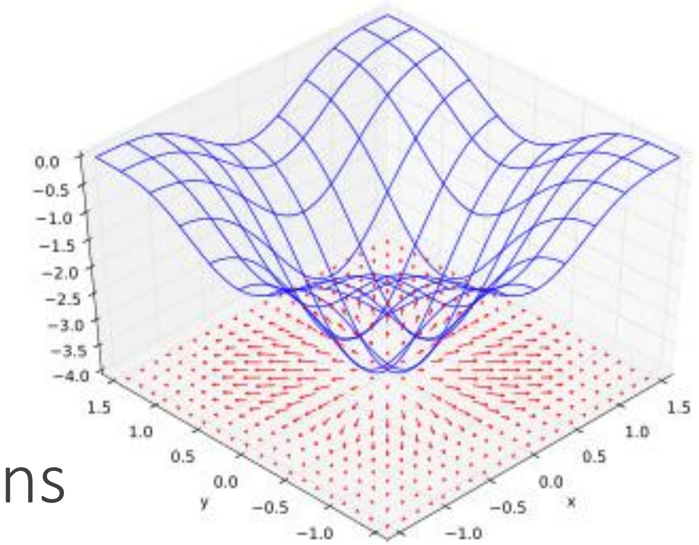
Stochastic → Analytic Regularization

- Taylor series expansion about a point $y = a$
$$h(y) = h(a) + h'(a)(y-a) + 0.5\, h''(a)(y-a)^2 + \ \dots$$
- For $y = \tilde{x} = x + \varepsilon$, where $\varepsilon \sim N(0, \sigma^2 I)$ the (first order) Taylor
$$h(\tilde{x}) = h(x+\varepsilon) = h(x) + \frac{\partial \mathrm{h}}{\partial x}\varepsilon$$
$$\mathrm{E}_{q(\tilde{x}|x)}[|h(x) - h(\tilde{x})|^2] = \left\|\frac{\partial \mathrm{h}}{\partial x}\varepsilon\right\|^2 \propto \sigma^2 \left\|\frac{\partial \mathrm{h}}{\partial x}\right\|^2$$
- Higher order expansions also possible although computationally expensive
  - Alternative: compute higher order derivatives stochastically
  - Analytic and stochastic regularization

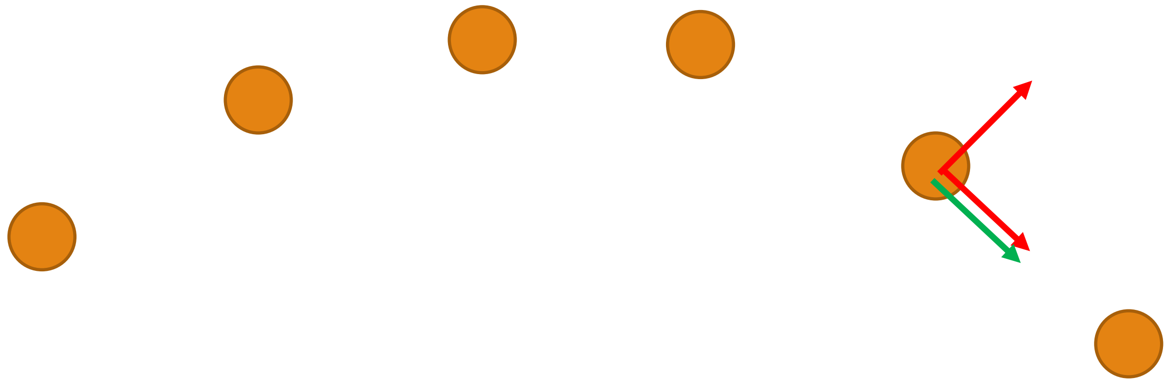
Some geometric intuition

- Gradients show sensitivity of function around a point

$$\mathcal{L} = \sum_d \ell\left(x, g(h(\tilde{x}))\right) + \lambda \left\|\frac{\partial \mathrm{h}}{\partial x}\right\|^2$$



- Regularization term penalizes sensitivity to all directions
- Reconstruction term enforces sensitivity only to direction of manifold



Some geometric intuition: let’s get real

- Let’s try to check where the gradients are sensitive around our manifold
- Gradients → Jacobian
- Strongest directions of Jacobian → Compute SVD of Jacobian

$$\frac{\partial h}{\partial x} = USV^T$$

- Take strongest singular values in S and respective vectors from U
  - These are our strongest tangents → directions of Jacobian of most sensitivity
- Hopefully the tangents sensitive to direction of the manifold, not random

Some geometric intuition: let's get real

- Let's try to check where the gradients are sensitive around our manifold
- Gradients → Jacobian
- Strongest directions of Jacobian → Compute SVD of Jacobian

$$\frac{\partial h}{\partial x} = USV^T$$

- Take strongest singular values in S and respective vectors from U
  - These are our strongest tangents → directions of Jacobian of most sensitivity
- Hopefully the tangents sensitive to direction of the manifold, not random
- How to check?

Some geometric intuition: let’s get real

- Let’s try to check where the gradients are sensitive around our manifold
- Gradients → Jacobian
- Strongest directions of Jacobian → Compute SVD of Jacobian

$$\frac{\partial h}{\partial x} = USV^T$$

- Take strongest singular values in S and respective vectors from U
  - These are our strongest tangents → directions of Jacobian of most sensitivity
- Hopefully the tangents sensitive to direction of the manifold, not random
- How to check? Visualize!!

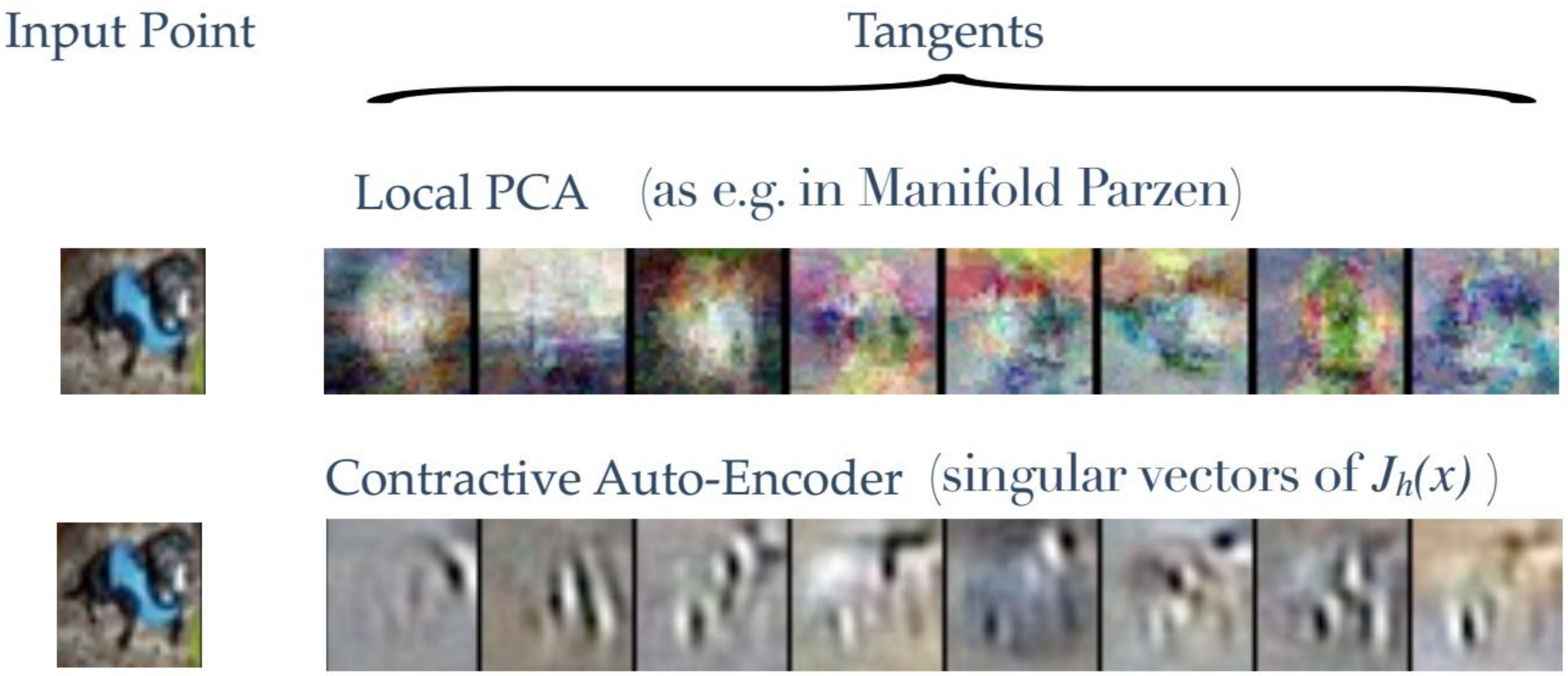
Some geometric intuition: let’s get real

- Let’s try to check where the gradients are sensitive around our manifold
- Gradients → Jacobian
- Strongest directions of Jacobian → Compute SVD of Jacobian

$$\frac{\partial h}{\partial x} = USV^T$$

- Take strongest singular values in S and respective vectors from U
  - These are our strongest tangents → directions of Jacobian of most sensitivity
- Hopefully the tangents sensitive to direction of the manifold, not random
- How to check? Visualize!!

Strongest tangents


Input Point
Tangents
Local PCA (as e.g. in Manifold Parzen)
Contractive Auto-Encoder (singular vectors of $J_h(x)$)

Variational Autoencoder

- We want to model the data distribution

$$p(x) = \int p_\theta(z) p_\theta(x|z) dz$$

- Posterior $p_\theta(z|x)$ is intractable for complicated likelihood functions $p_\theta(x|z)$, e.g. a neural network → $p(x)$ is also intractable
- Introduce an inference machine $q_\varphi(z|x)$ (e.g. another neural network) that **learns to approximate** the posterior $p_\theta(z|x)$
  - Since we cannot know $p_\theta(z|x)$ define a variational lower bound to optimize instead

$$\mathcal{L}(\theta, \varphi, x) = -D_{KL}\left(q_\varphi(z|x) \| p_\theta(z)\right) + E_{q_\varphi(z|x)}(\log p_\theta(x|z))$$

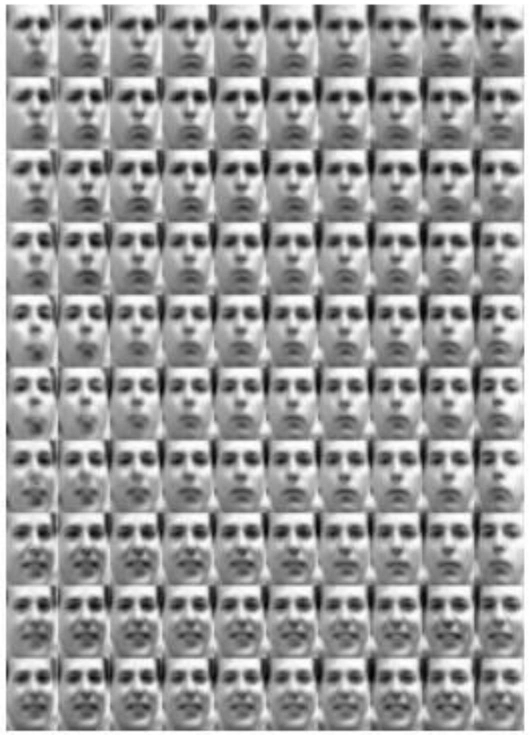
Regularization term Reconstruction term

Variational Autoencoder

- Since we cannot know $p_\theta(z|x)$ define a variational lower bound to optimize instead
- Optimize inference machine $q_\varphi(z|x)$ and likelihood machine $p_\theta(x|z)$
  - $q_\varphi(z|x)$ should follow a Gaussian distribution
- Reparamerization trick
  - Instead of z being stochastic, introduce stochastic noise variable ε
  - Then, think of z as deterministic, where $z = \mu_z(x) + \varepsilon\sigma_z(x)$
  - Simultaneous optimization of $q_\varphi(z|x)$ and $p_\theta(x|z)$ now possible

$$\mathcal{L}(\theta, \varphi, x) = \underbrace{-D_{KL}\left(q_\varphi(z|x) \| p_\theta(z)\right)}_{\text{Regularization term}} + \underbrace{E_{q_\varphi(z|x)}(\log p_\theta(x|z))}_{\text{Reconstruction term}}$$

Examples


(a) Learned Frey Face manifold

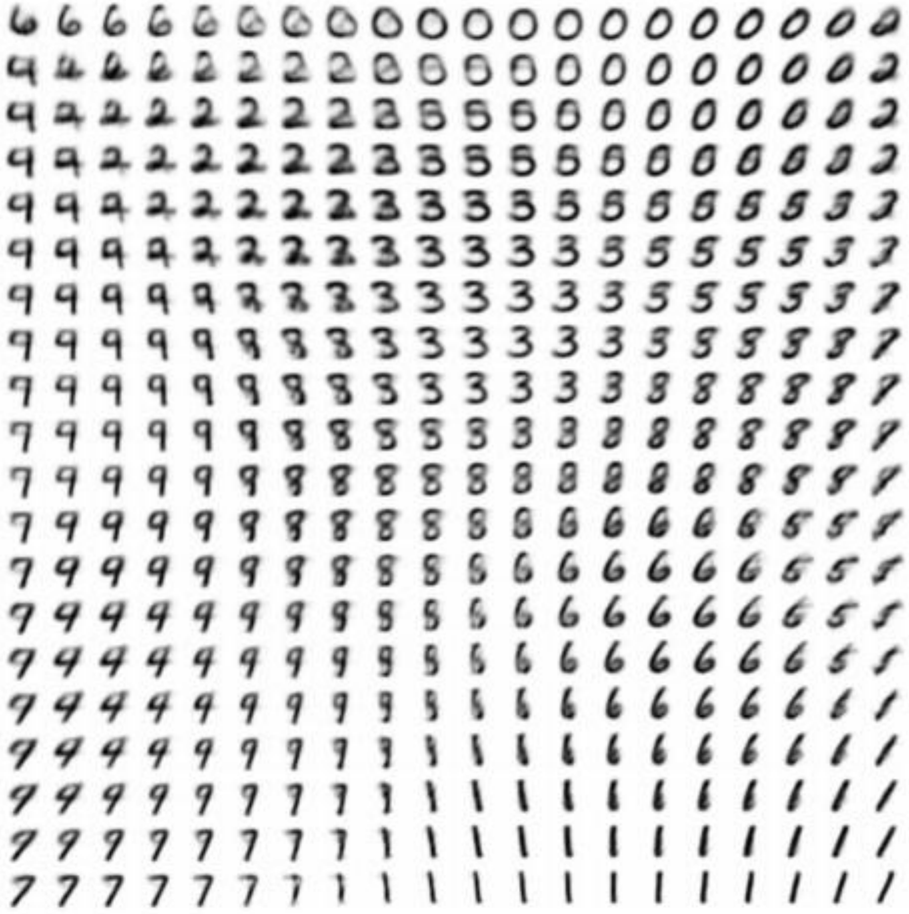

(b) Learned MNIST manifold

Figure 4: Visualisations of learned data manifold for generative models with two-dimensional latent space, learned with AEVB. Since the prior of the latent space is Gaussian, linearly spaced coordinates on the unit square were transformed through the inverse CDF of the Gaussian to produce values of the latent variables $\mathbf{z}$. For each of these values $\mathbf{z}$, we plotted the corresponding generative $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ with the learned parameters $\boldsymbol{\theta}$.

Adversarial Networks

Generative Adversarial Networks

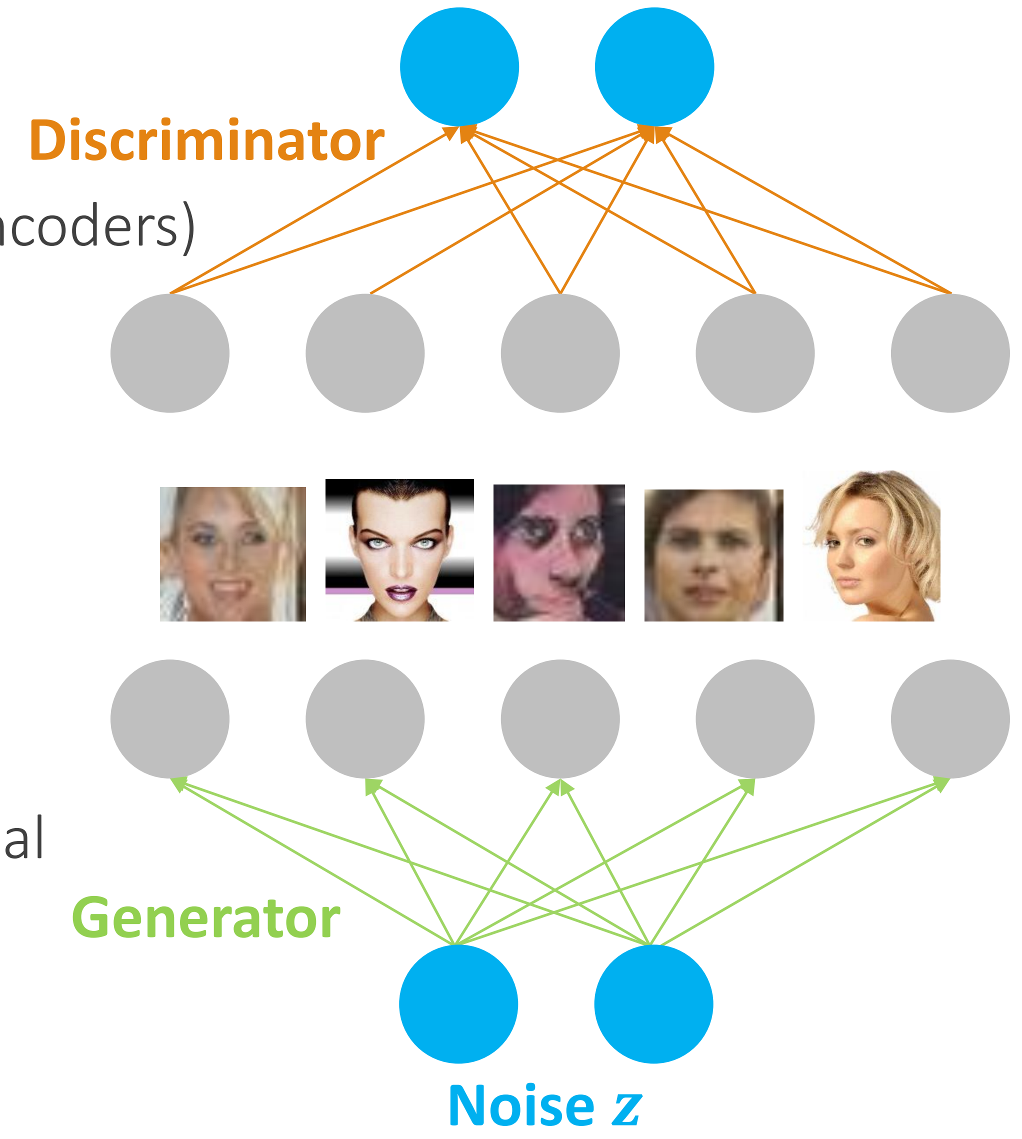
- So far we were trying to express the pdf of the data $p(x)$
- Why bother? Is that really necessary? Is that limiting?
- Can we model directly the sampling of data?

Generative Adversarial Networks

- So far we were trying to express the pdf of the data $p(x)$
- Why bother? Is that really necessary? Is that limiting?
- Can we model directly the sampling of data?
- Yes, by modelling sampling as a “Thief vs. Police” game

Generative Adversarial Networks

- Composed of two successive networks
  - Generator network (like upper half of autoencoders)
  - Discriminator network (like a convent)
- Learning
  - Sample "noise" vectors z
  - Per z the generator produces a sample x
  - Make a batch where half samples are real, half are the generated ones
  - The discriminator needs to predict what is real and what is fake



“Police vs Thief”

- Generator and discriminator networks optimized together
  - The generator (thief) tries to fool the discriminator
  - The discriminator (police) tries to not get fooled by the generator
- Mathematically

$$\min_G \max_D V(G, D) = E_{x\sim p_{data}(x)} \log D(x) + E_{z\sim p_z(z)} \log(1 - D(G(z)))$$

A graphical interpretation

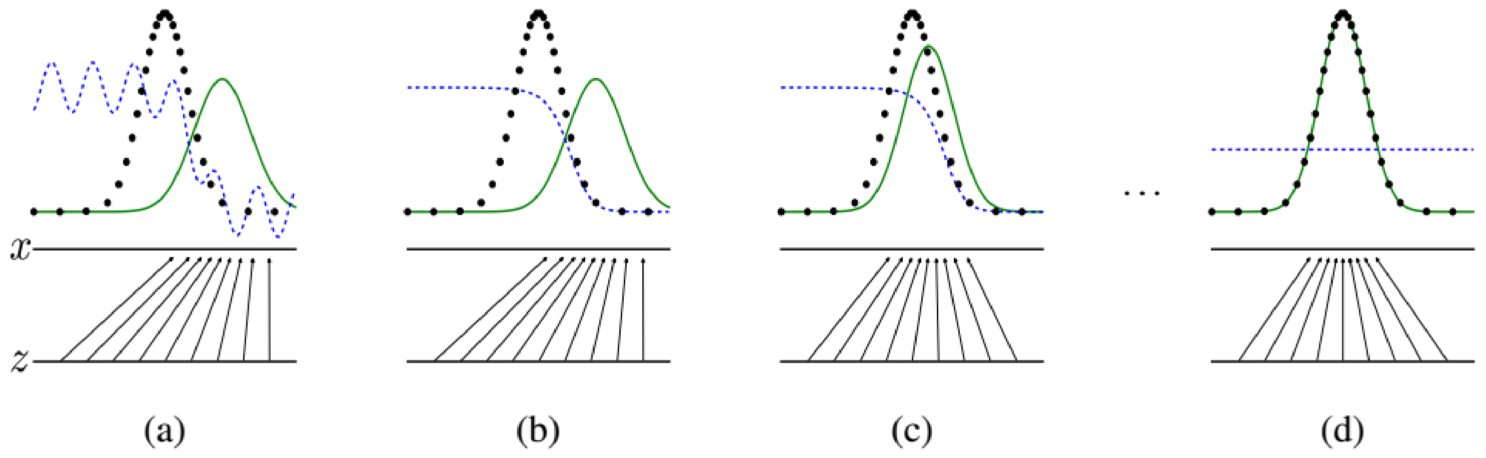
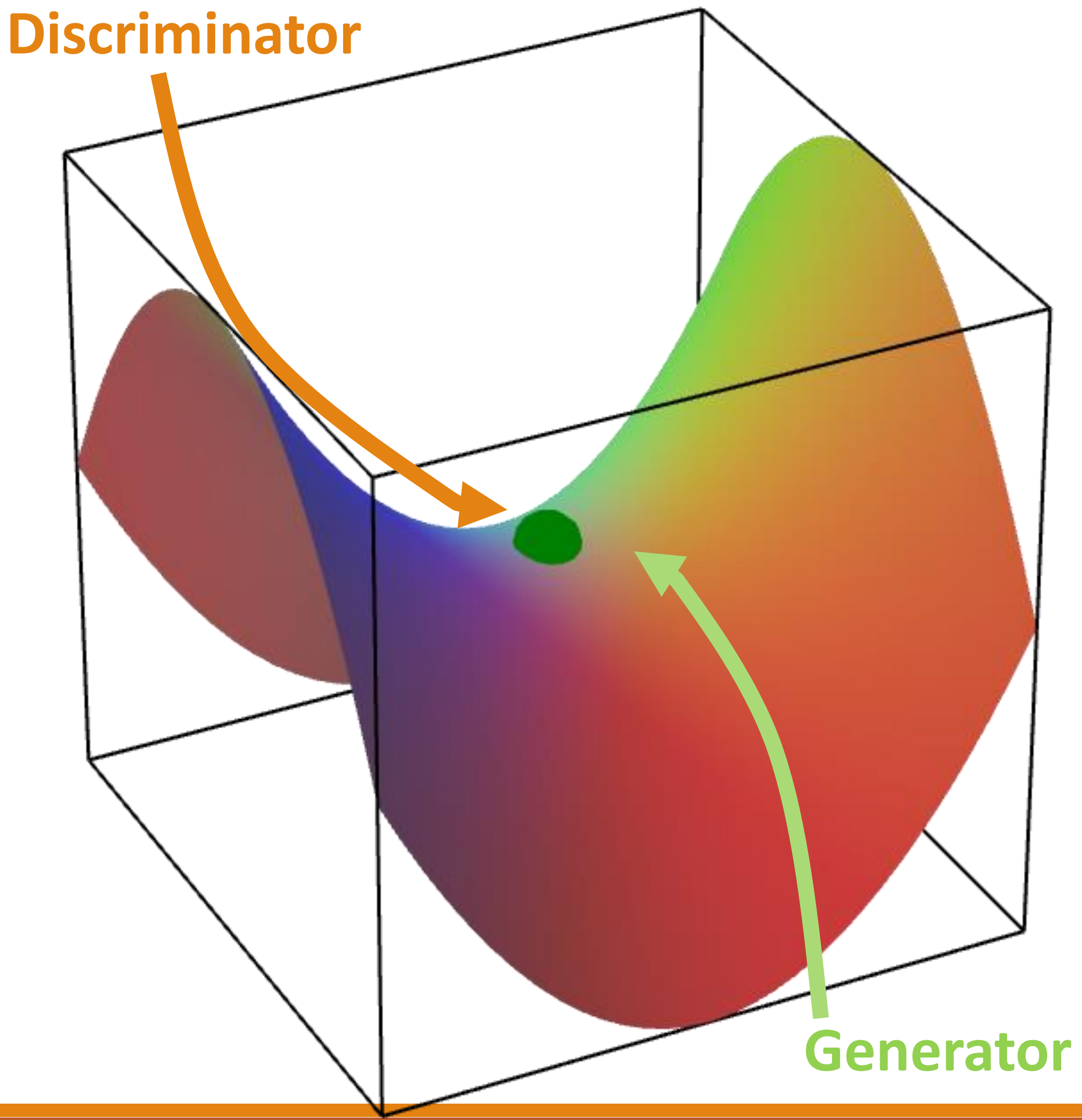


Figure 1: Generative adversarial nets are trained by simultaneously updating the **d**iscriminative distribution (D, blue, dashed line) so that it discriminates between samples from the data generating distribution (black, dotted line) $p_{\boldsymbol{x}}$ from those of the **g**enerative distribution p_g (G) (green, solid line). The lower horizontal line is the domain from which $\boldsymbol{z}$ is sampled, in this case uniformly. The horizontal line above is part of the domain of $\boldsymbol{x}$. The upward arrows show how the mapping $\boldsymbol{x} = G(\boldsymbol{z})$ imposes the non-uniform distribution p_g on transformed samples. G contracts in regions of high density and expands in regions of low density of p_g. (a) Consider an adversarial pair near convergence: p_g is similar to p_{data} and D is a partially accurate classifier. (b) In the inner loop of the algorithm D is trained to discriminate samples from data, converging to $D^*(\boldsymbol{x}) = \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x})+p_g(\boldsymbol{x})}$. (c) After an update to G, gradient of D has guided $G(\boldsymbol{z})$ to flow to regions that are more likely to be classified as data. (d) After several steps of training, if G and D have enough capacity, they will reach a point at which both cannot improve because $p_g = p_{\text{data}}$. The discriminator is unable to differentiate between the two distributions, i.e. $D(\boldsymbol{x}) = \frac{1}{2}$.

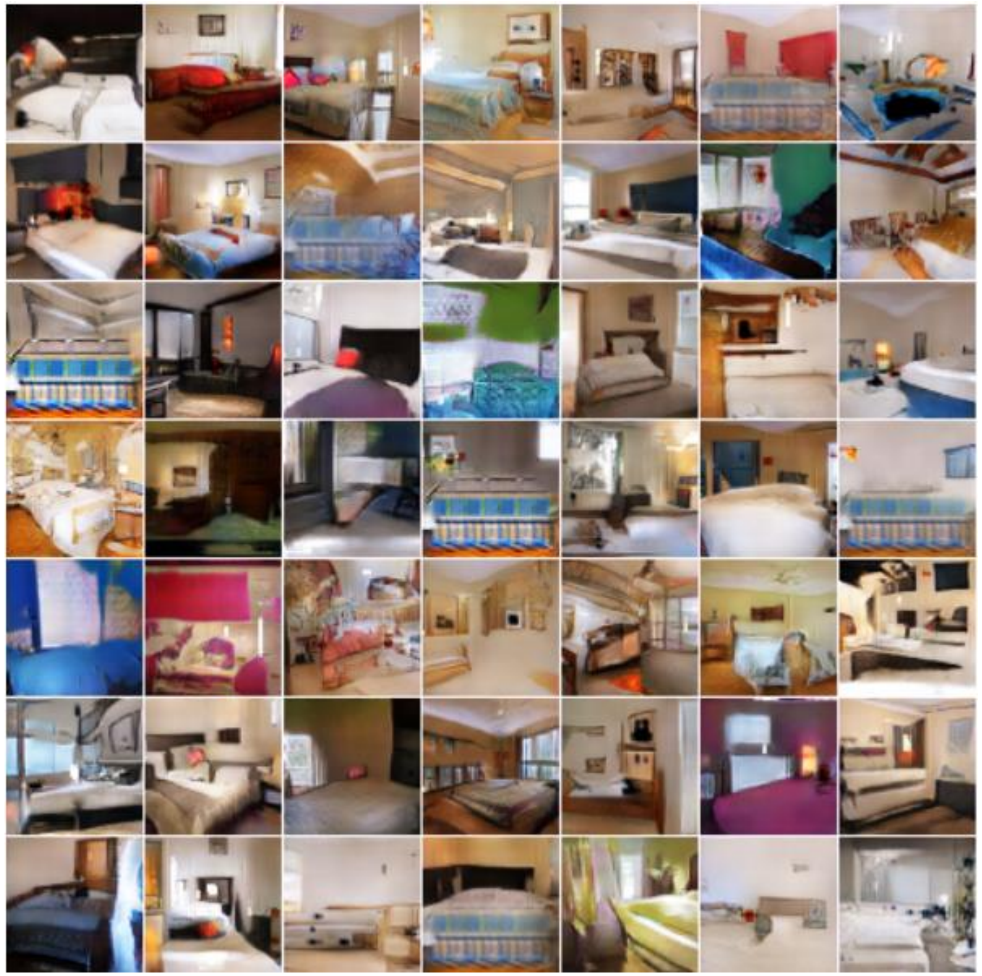
Adversarial network hardships

- This is (**very**) fresh research
- As such the theory behind adversarial networks is by far not mature
- Moreoever, GANs are quite unstable
- Optimal solution is a saddle point
  - Easy to mess up, little stability
- Solutions?
  - Not great ones yet, although several researchers are actively working on that



Examples of generated images

Bedrooms



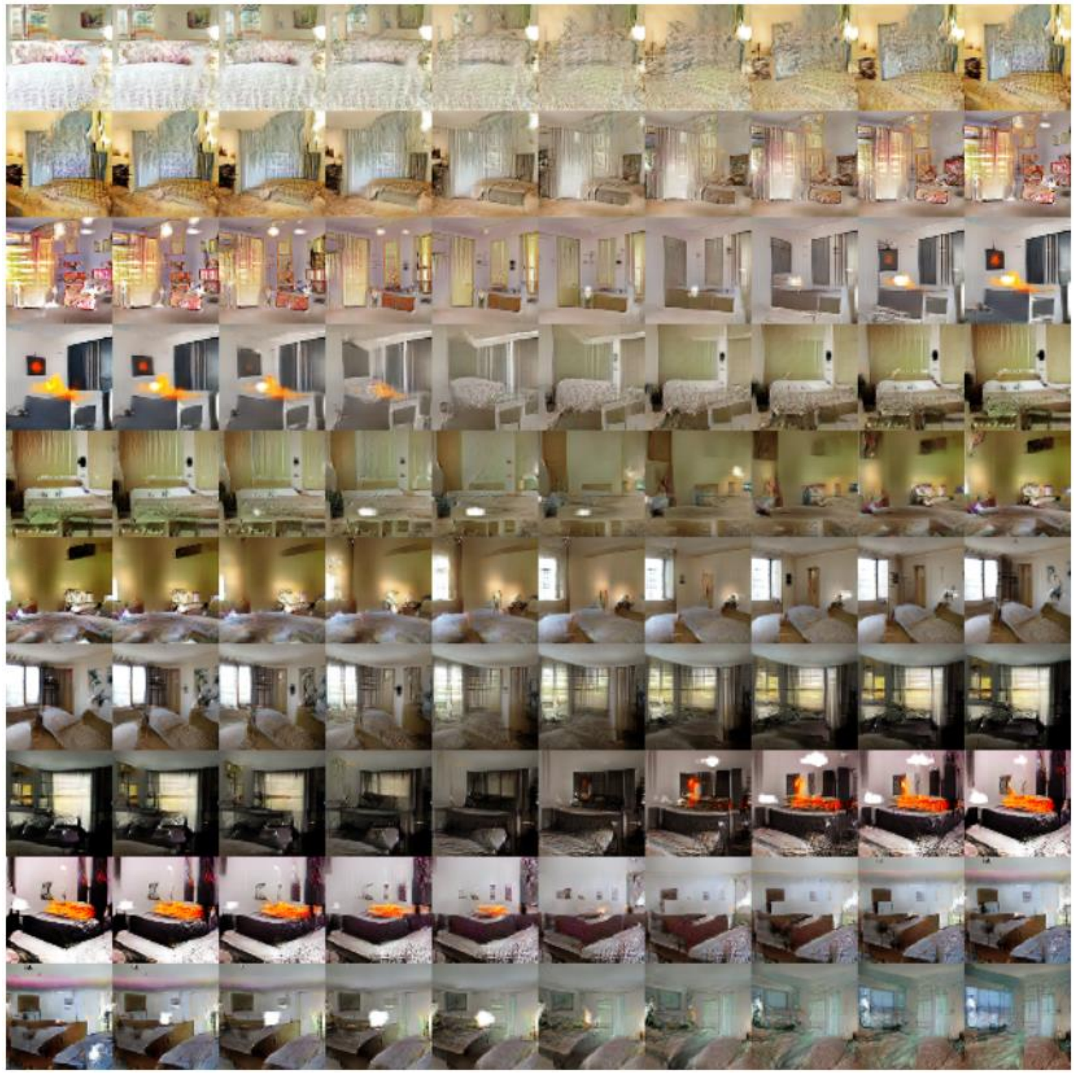
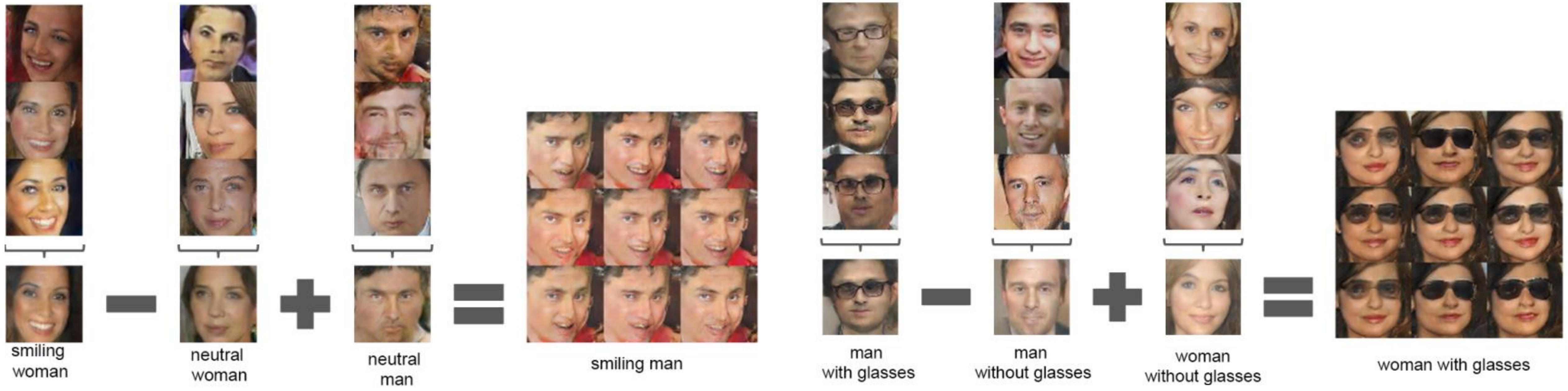


Image “arithmetics”



Summary

- Latent space data manifolds
- Autoencoders and Variational Autoencoders
- Boltzmann Machines
- Adversarial Networks
- Self-Supervised Networks

Reading material & references

- http://www.deeplearningbook.org/
  - Part II: Chapter 10
- Excellent blog post on Backpropagation Through Time
  - http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/
  - http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-2-implementing-a-language-model-rnn-with-python-numpy-and-theano/
  - http://www.wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-through-time-and-vanishing-gradients/
- Excellent blog post explaining LSTMs
  - http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[Pascanu2013] Pascanu, Mikolov, Bengio. On the difficulty of training Recurrent Neural Networks, JMLR, 2013

Next lecture

- Memory networks
- Advanced applications of Recurrent and Memory Networks