UNIVERSITEIT VAN AMSTERDAM

Unsupervised learning, representation and generative models

Deep Learning

23/11/17

Giorgio Patrini
g.patrini@uva.nl

Overview

- **Introduction, manifolds, PCA** (Goodfellow's 5.11.3, 13.5)
- Auto-encoders (14)
  - Objective, undercomplete / regularized auto-encoders
  - Denoising auto-encoders, contractive auto-encoders

- Generative models (parts of 20)
  - Variational auto-encoder (20.9, 20.10.3)
  - Generative adversarial network (20.10.4, 20.10.6)
  - PixelRNN, models evaluation (20.10.7, 20.14)

Supervised vs. unsupervised learning

- Supervised
  - Data $D = \{\boldsymbol{X}, \boldsymbol{T}\}$
  - Goals $f(\boldsymbol{x}) \approx t, p(\boldsymbol{t}|\boldsymbol{x})$
  - Classification (discrete) or regression (continuous)

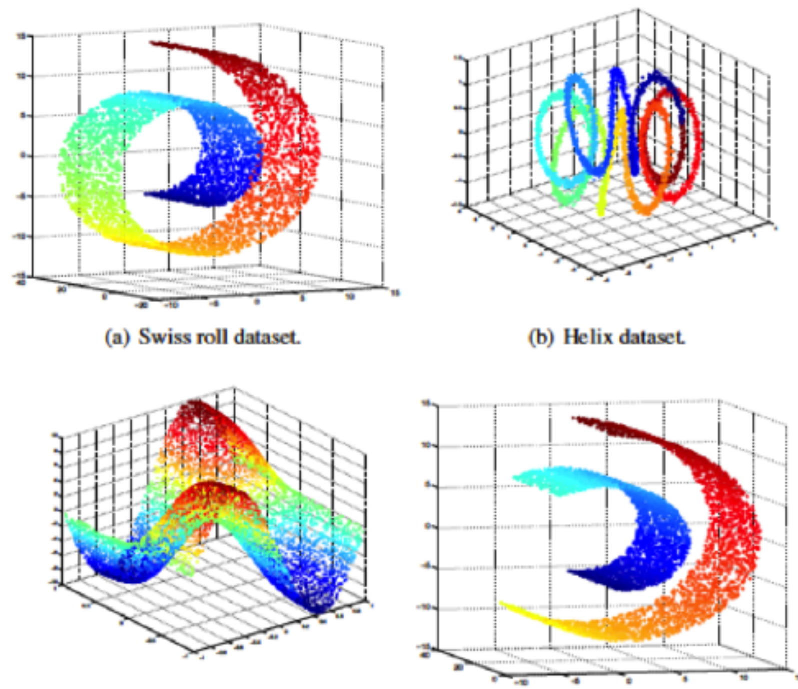
Supervised vs. unsupervised learning

- Supervised
  - Data $D = \{\boldsymbol{X}, \boldsymbol{T}\}$
  - Goals $f(\boldsymbol{x}) \approx t, p(\boldsymbol{t}|\boldsymbol{x})$
  - Classification (discrete) or regression (continuous)

- Unsupervised
  - Data $D = \{\boldsymbol{X}\}$
  - Goals $p(\boldsymbol{x}), p(\boldsymbol{h}|\boldsymbol{x})$ or $p(\boldsymbol{x}|\boldsymbol{h})$
  - E.g. density estimation, dimensionality reduction, clustering, feature learning, generation

High dimensional spaces and the manifold hypothesis

- **Manifold hypothesis:** natural data lives in a low-dimensional non-linear manifold



(a) Swiss roll dataset.

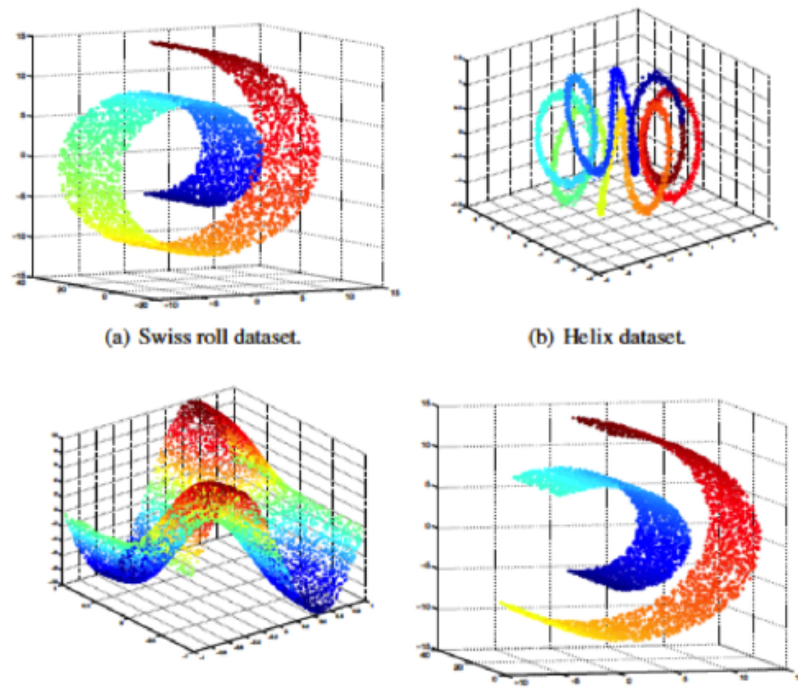
(b) Helix dataset.

(c) Twinpeaks dataset.

(d) Broken Swiss roll dataset.

High dimensional spaces and the manifold hypothesis

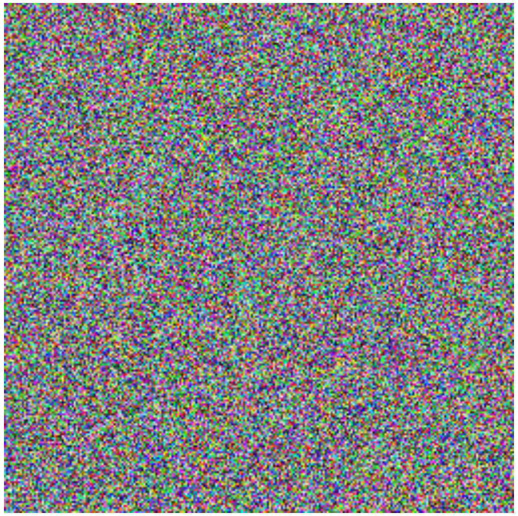
- **Manifold hypothesis:** natural data lives in a low-dimensional non-linear manifold
- Or equivalently, data is concentrated with high probability in a small non-linear region of the high-dimensional space
- See Goodfellow’s 5.11.3



(a) Swiss roll dataset. (b) Helix dataset. (c) Twinpeaks dataset. (d) Broken Swiss roll dataset.

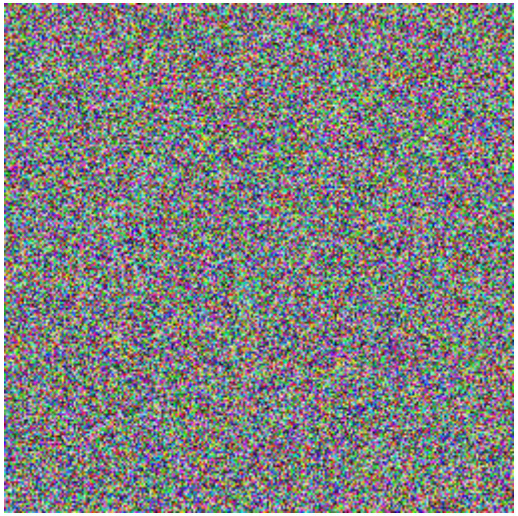
High dimensional spaces and the manifold hypothesis

- Take the spaces of all possible images of size 256 x 256 x 3 pixels (3 is given by RGB encoding)
- An image sampled uniformly from the pixel space looks like this ->



High dimensional spaces and the manifold hypothesis

- Take the spaces of all possible images of size 256 x 256 x 3 pixels (3 is given by RGB encoding)
- An image sampled uniformly from the pixel space looks like this ->

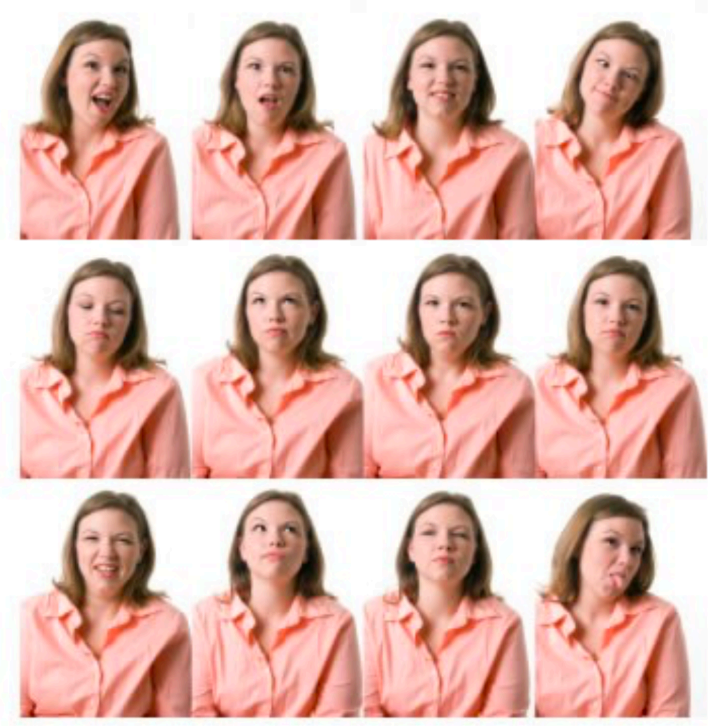


- To hear “random noise”: https://goo.gl/AZZ6z9
- Text: random letters or random words
- The distribution of natural high dimensional data has support over an unknown low dimensional manifold

Example: images of faces

Example: all face images of **one** person

- 3 x 256 x 256 pixels = 3 x 256^2 dimensions =~ 196K



[LeCun&Ranzato'13]

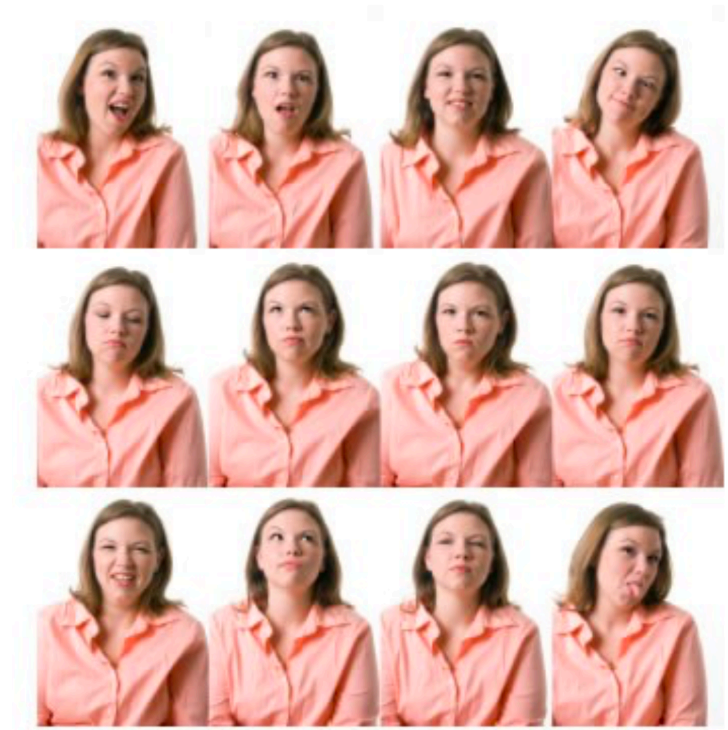
Example: images of faces

Example: all face images of **one** person

- 3 x 256 x 256 pixels = 3 x 256^2 dimensions =~ 196K

But:

- Faces have 3 Cartesian coordinates (translations) and 3 Euler angles (rotations) and humans have less than about 50 muscles in the face
- Hence the manifold of face images for a person has <= 56 dimensions



[LeCun&Ranzato'13]

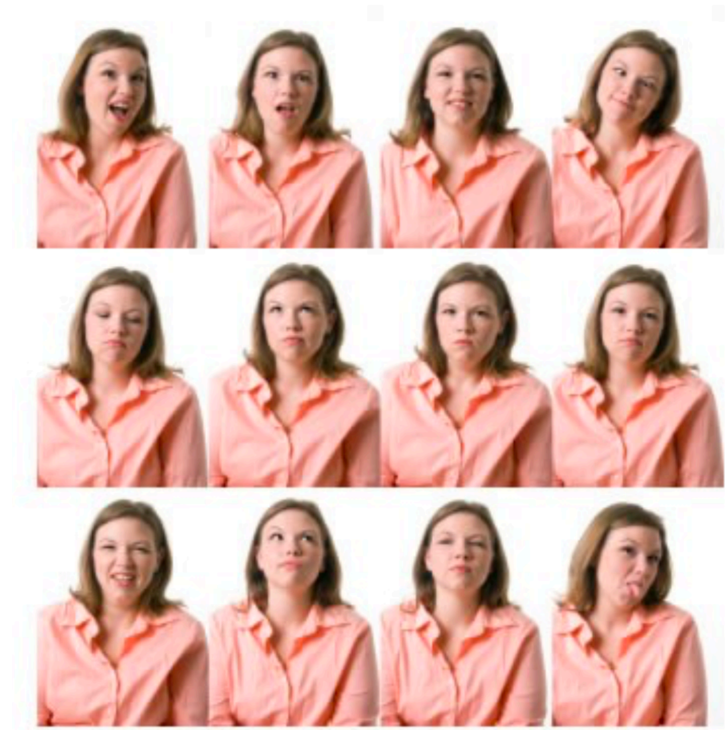
Example: images of faces

Example: all face images of **one** person

- 3 x 256 x 256 pixels = 3 x 256^2 dimensions =~ 196K

But:

- Faces have 3 Cartesian coordinates (translations) and 3 Euler angles (rotations) and humans have less than about 50 muscles in the face
- Hence the manifold of face images for a person has <= 56 dimensions
- We should be able "to navigate" all the data distribution with 56 non-linear coordinates, but we don't know them…



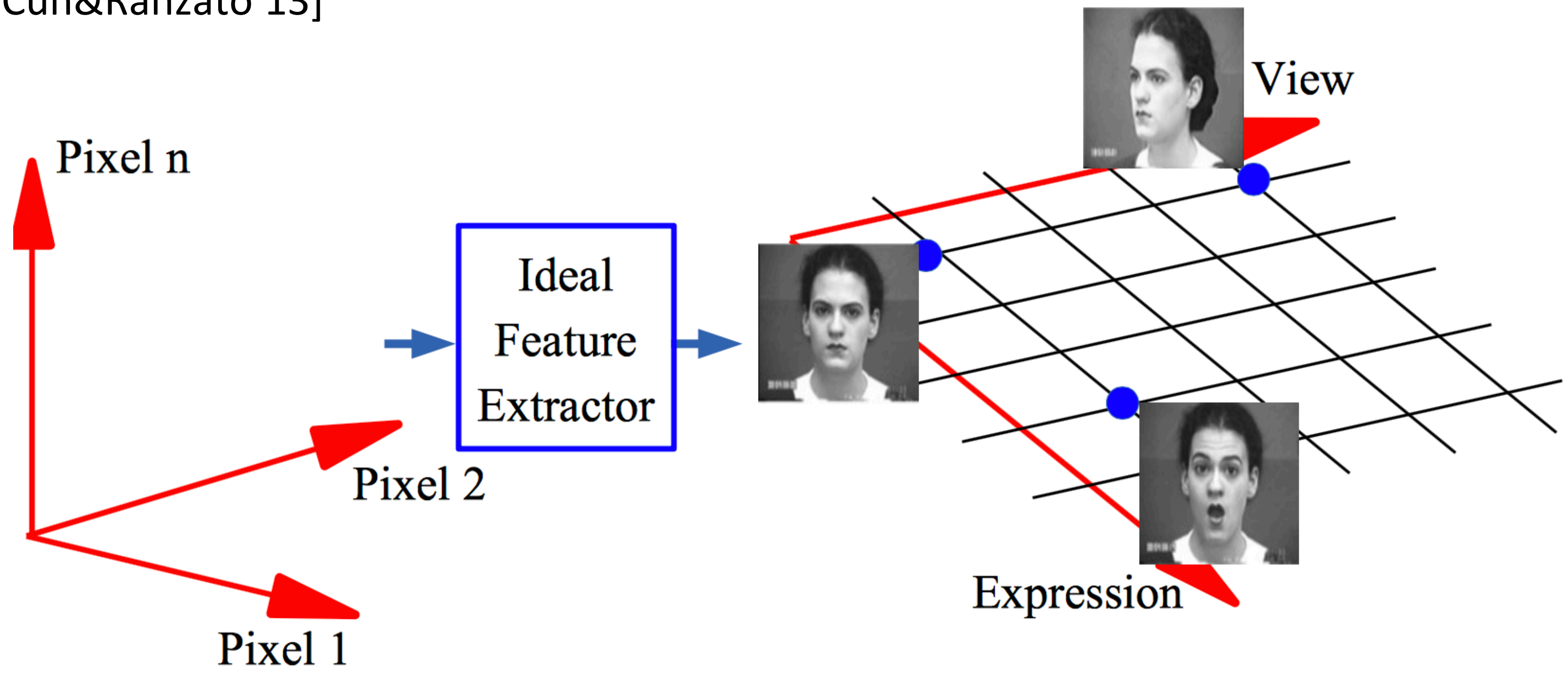
[LeCun&Ranzato'13]

Example: images of faces

Figure 5.13: Training examples from the QMUL Multiview Face Dataset (Gong *et al.*, 2000) for which the subjects were asked to move in such a way as to cover the two-dimensional manifold corresponding to two angles of rotation. We would like learning algorithms to be able to discover and disentangle such manifold coordinates. Fig. 20.6 illustrates such a feat.

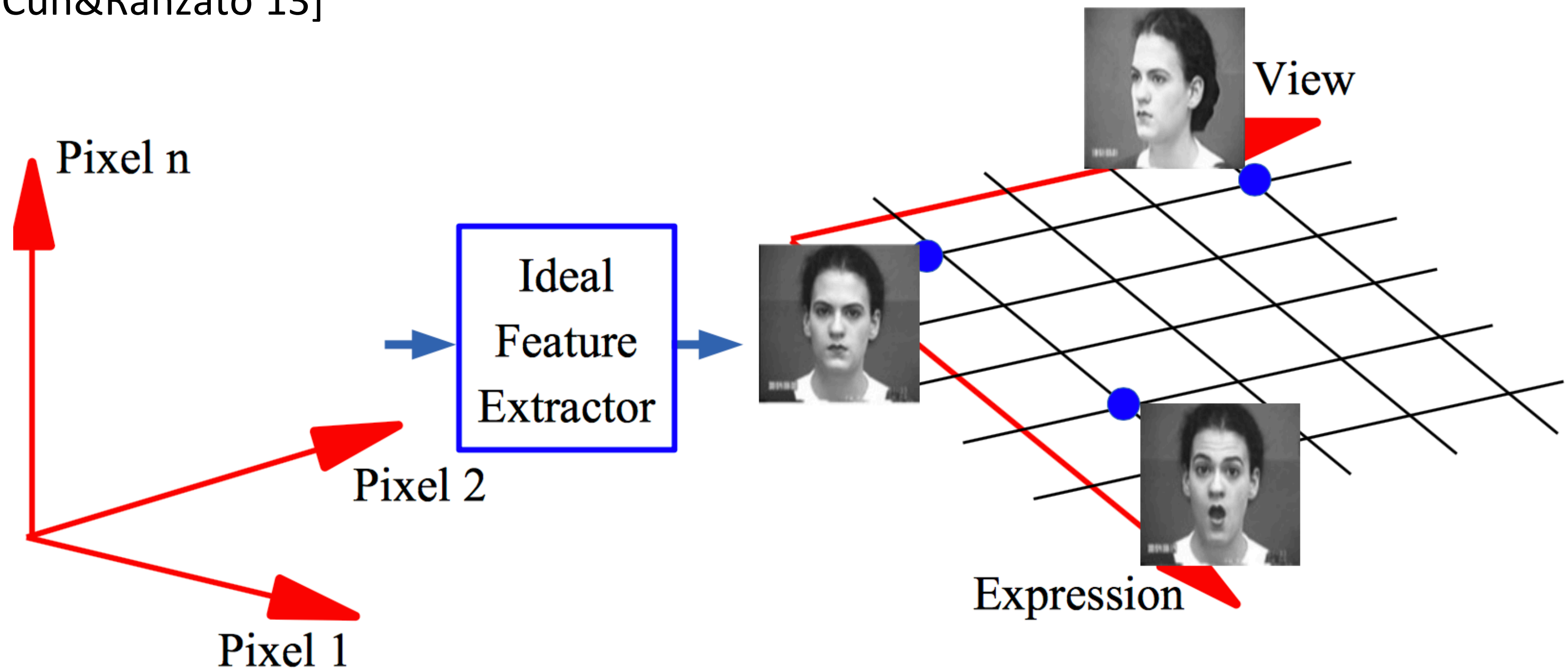
An ideal feature extractor

[LeCun&Ranzato'13]



An ideal feature extractor

[LeCun&Ranzato'13]



Problem: we have to discover those features, "the new coordinates", automatically.

We cannot use supervised learning to regress them…
Unsupervised learning

PCA for manifold learning

- (*You should remember that*) PCA defines a (linear) projection $f(x)$ onto a low M-dimensional space that **preserves most of the variance** of the original data.

PCA for manifold learning

- (*You should remember that*) PCA defines a (linear) projection $f(\boldsymbol{x})$ onto a low M-dimensional space that **preserves most of the variance** of the original data.
- In particular, PCA can be obtained as pair of encoder/decoder functions minimizing the reconstruction error:

$$\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}[\|\boldsymbol{x} - g(f(\boldsymbol{x}))\|_2^2]$$

where encoder and decoder are respectively

$$f(\boldsymbol{x}) = W^\top(\boldsymbol{x} - \boldsymbol{\mu}), \;\; g(\boldsymbol{x}) = Vf(\boldsymbol{x}) + \boldsymbol{b}$$

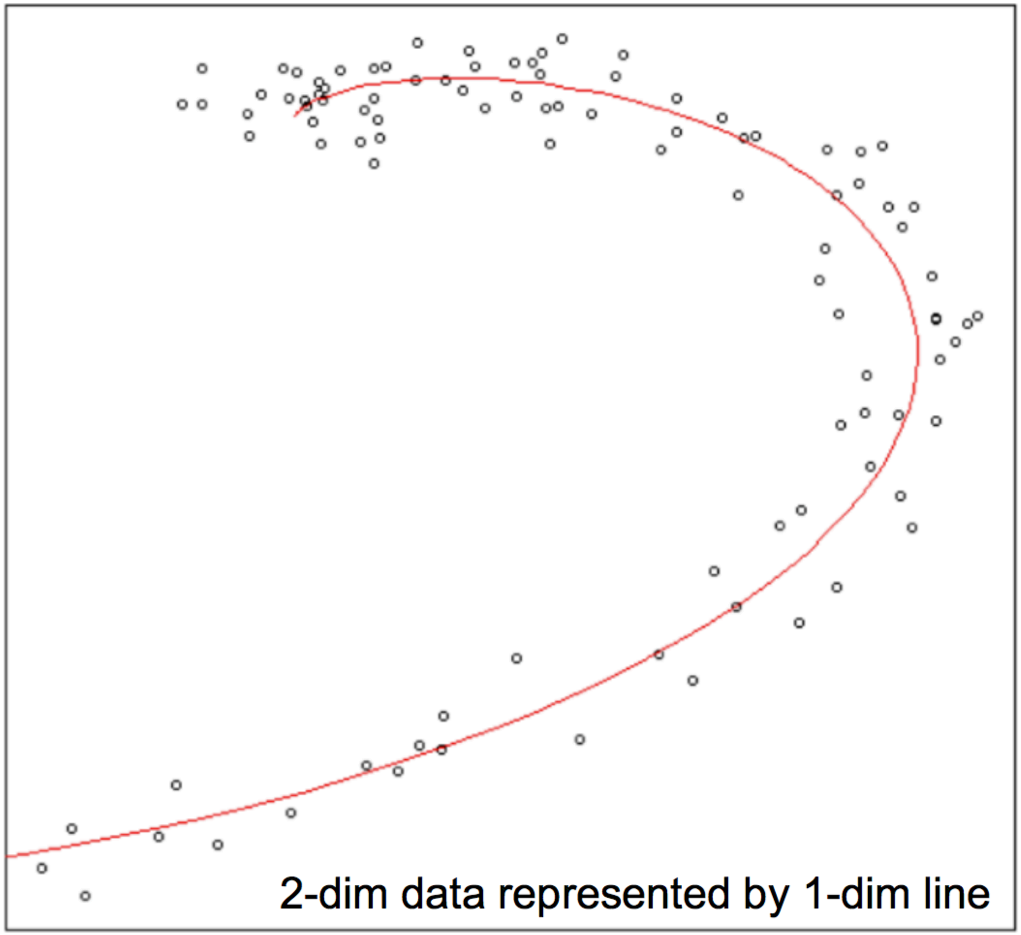
PCA for manifold learning

- At optimum, it holds that
  - $V = W, \boldsymbol{\mu} = \boldsymbol{b} = \mathbb{E}_{\boldsymbol{x}}[\boldsymbol{x}]$
  - the columns of W form an orthonormal basis spanning the subspace of the top M eigenvectors of the covariance matrix $\mathbb{E}_{\boldsymbol{x}}[(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^\top]$
  - the reconstruction error is the sum of eigenvalues of the D - M discarded components

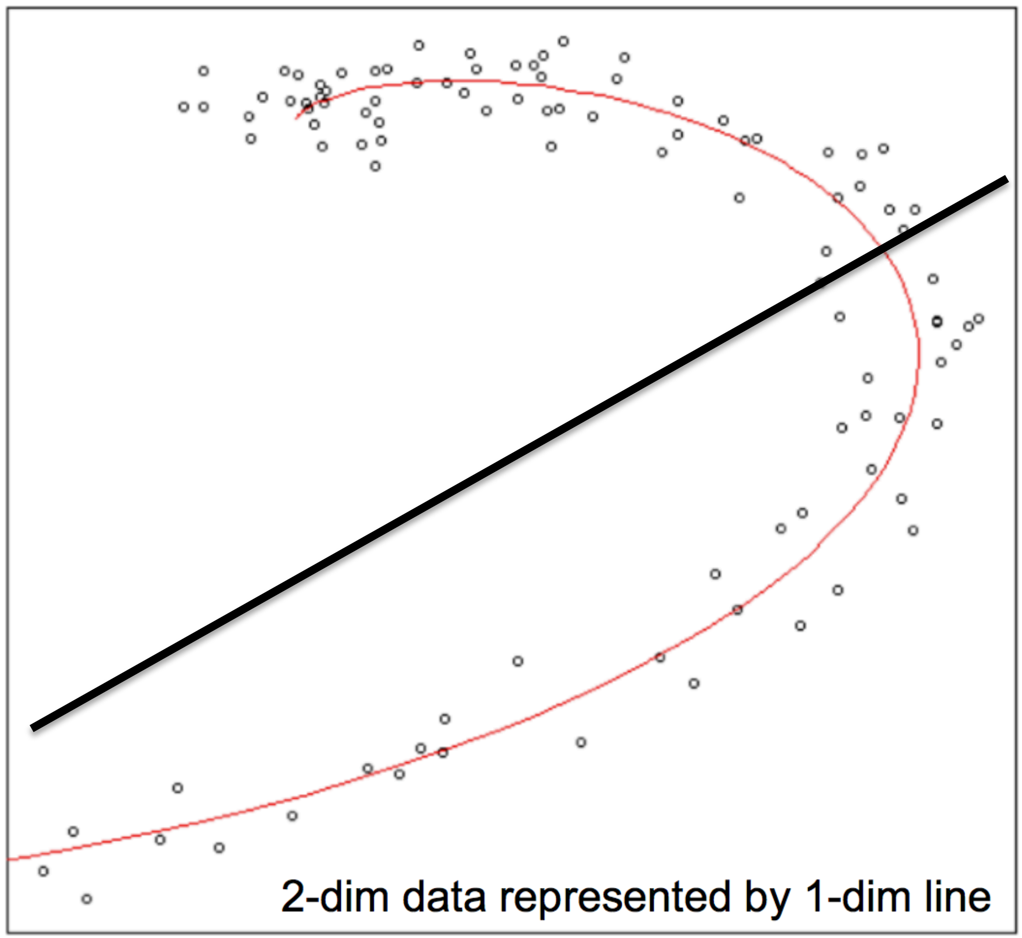
PCA for manifold learning

- At optimum, it holds that
  - $V = W, \boldsymbol{\mu} = \boldsymbol{b} = \mathbb{E}_{\boldsymbol{x}}[\boldsymbol{x}]$
  - the columns of W form an orthonormal basis spanning the subspace of the top M eigenvectors of the covariance matrix $\mathbb{E}_{\boldsymbol{x}}[(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^\top]$
  - the reconstruction error is the sum of eigenvalues of the D - M discarded components
- PCA can be seen as a manifold learning algorithm, which encoder f projects $\boldsymbol{x}$ onto a M-dimensional **linear subspace** that preserves most of the variance of the data.

PCA on a spiral manifold



PCA on a spiral manifold

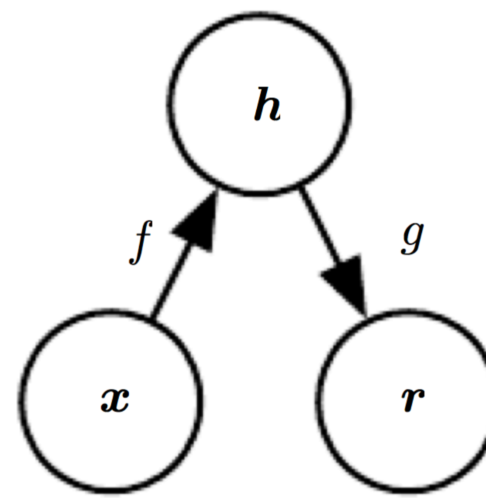


Overview

- Introduction, manifolds, PCA (Goodfellow 5.11.3, 13.5)
- **Auto-encoders** (14)
  - **Objective, undercomplete / regularized auto-encoders**
  - Denoising auto-encoders, contractive auto-encoders
- Generative models (parts of 20)
  - Variational auto-encoder (20.9, 20.10.3)
  - Generative adversarial network (20.10.4, 20.10.6)
  - PixelRNN, models evaluation (20.10.7, 20.14)

Auto-encoders

- Non-linear generalization of PCA
- Encoder/decoder $\boldsymbol{h} = f(\boldsymbol{x})$ and $\boldsymbol{r} = g(\boldsymbol{h})$ where $\boldsymbol{h}$ is the low-dimensional representation of $\boldsymbol{x}$ and $\boldsymbol{r}$ is its reconstruction
- $\boldsymbol{h}$ *names: features, representation, code, embedding, latent variables*
- Encoder and decoder are both neural nets



Auto-encoder objective

- Minimize a loss function (=dissimilarity) between input and reconstruction:

$$\frac{1}{N}\sum_{n=1}^{N} L(\boldsymbol{x}_n, \boldsymbol{r}_n) = \frac{1}{N}\sum_{n=1}^{N} L(\boldsymbol{x}_n, g(f(\boldsymbol{x}_n)))$$

Auto-encoder objective

- Minimize a loss function (=dissimilarity) between input and reconstruction:

$$\frac{1}{N}\sum_{n=1}^{N} L(\boldsymbol{x}_n, \boldsymbol{r}_n) = \frac{1}{N}\sum_{n=1}^{N} L(\boldsymbol{x}_n, g(f(\boldsymbol{x}_n)))$$

- The loss function is often the squared Euclidean norm: $\|\boldsymbol{x}_n - \boldsymbol{r}_n\|_2^2$
- Use cross-entropy when input is binary

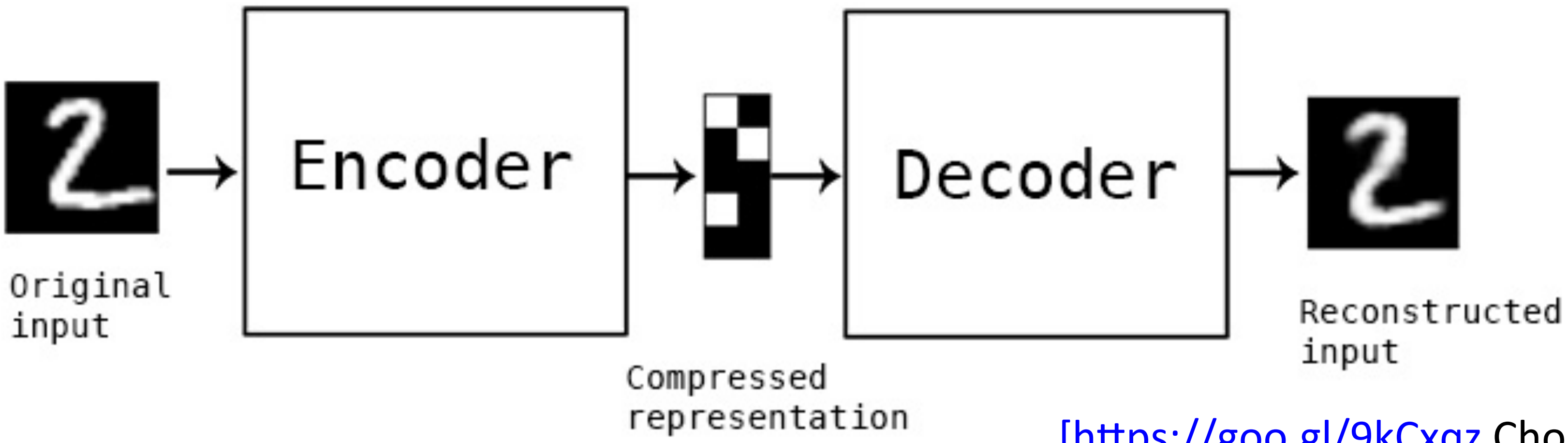
Auto-encoder objective

- Minimize a loss function (=dissimilarity) between input and reconstruction:

$$\frac{1}{N}\sum_{n=1}^{N} L(\boldsymbol{x}_n, \boldsymbol{r}_n) = \frac{1}{N}\sum_{n=1}^{N} L(\boldsymbol{x}_n, g(f(\boldsymbol{x}_n)))$$

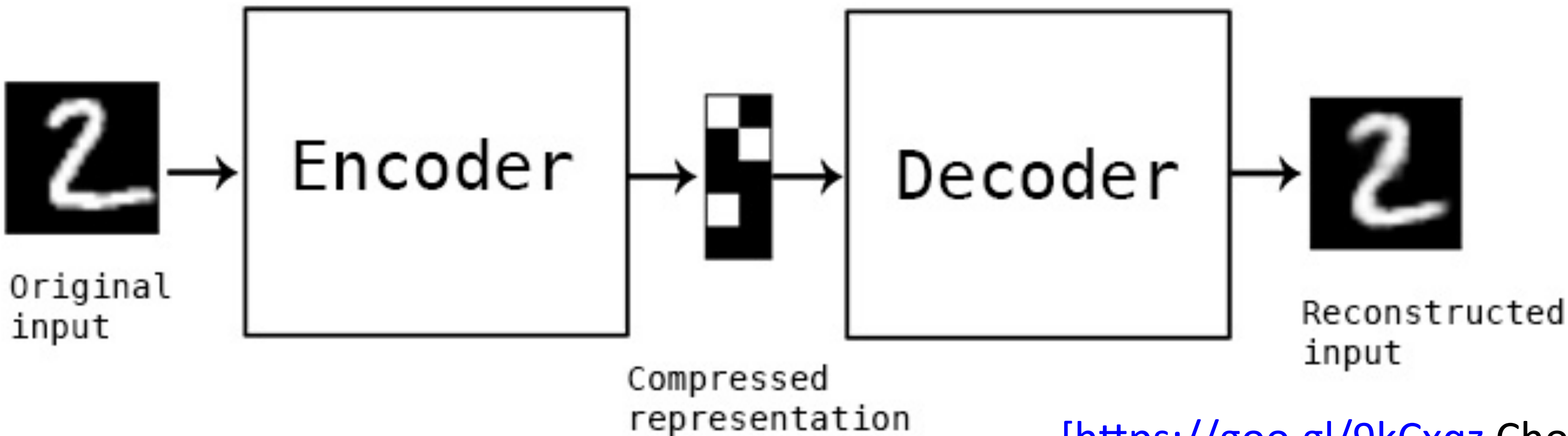
- The loss function is often the squared Euclidean norm $\|\boldsymbol{x}_n - \boldsymbol{r}_n\|_2^2$
- Use cross-entropy when input is binary
- Find the parameters of encoder and decoder by back-propagation / SGD.

Auto-encoder architecture



[https://goo.gl/9kCxqz Chollet]

Auto-encoder architecture



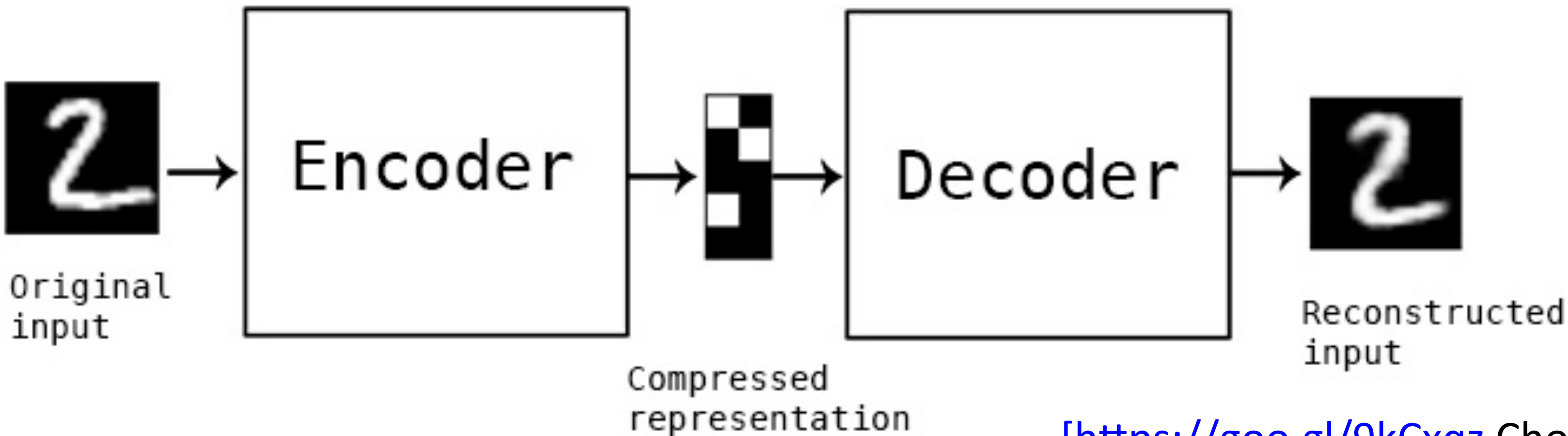
[https://goo.gl/9kCxqz Chollet]

- Example: one layer encoder/one layer decoder:

$$f(\boldsymbol{x}) = \mathrm{ReLU}(W\boldsymbol{x} + \boldsymbol{b}), \;\; g(\boldsymbol{x}) = \sigma(Vf(\boldsymbol{x}) + \boldsymbol{c})$$

- If input/output are binary, σ is a sigmoid. If they are real valued, use a linear activation.

Auto-encoder architecture



[https://goo.gl/9kCxqz Chollet]

- Example: one layer encoder/one layer decoder:

$$f(\boldsymbol{x}) = \mathrm{ReLU}(W\boldsymbol{x} + \boldsymbol{b}), \;\; g(\boldsymbol{x}) = \sigma(Vf(\boldsymbol{x}) + \boldsymbol{c})$$

- If input/output are binary, σ is a sigmoid. If they are real valued, use a linear activation.
- Sometimes, weights are **tied**: $W^\top = V$

Justifying the auto-encoder objective

- Goal of auto-encoder: learn a **good representation***
  $p(\boldsymbol{h}|\boldsymbol{x})$ (encoder) of the manifold

**What is a "good" representation in general? Question is very broad… Read Chapter 15 if interested*

Justifying the auto-encoder objective

- Goal of auto-encoder: learn a **good representation*** $p(\boldsymbol{h}|\boldsymbol{x})$ (encoder) of the manifold

- The viewpoint of auto-encoders: a representation is good if **it preserves most of the information** of the input. The **mutual information** of input and code:

$$I(\boldsymbol{x};\boldsymbol{h}) = \int p(\boldsymbol{x},\boldsymbol{h}) \log \frac{p(\boldsymbol{x},\boldsymbol{h})}{p(\boldsymbol{x})p(\boldsymbol{h})}$$

**What is a "good" representation in general? Question is very broad… Read Chapter 15 if interested*

Auto-encoders maximize the mutual information

- Find encoder with parameter $\boldsymbol{\theta}$ to maximize information

$$\begin{aligned}\operatorname*{argmax}_{\boldsymbol{\theta}} I(\boldsymbol{x};\boldsymbol{h}) &= \operatorname*{argmax}_{\boldsymbol{\theta}} H(\boldsymbol{x}) - H(\boldsymbol{x}|\boldsymbol{h}) \\ &= \operatorname*{argmax}_{\boldsymbol{\theta}} -H(\boldsymbol{x}|\boldsymbol{h}) \\ &= \operatorname*{argmax}_{\boldsymbol{\theta}} \mathbb{E}_{p(\boldsymbol{x},\boldsymbol{h})} \log p(\boldsymbol{x}|\boldsymbol{h})\end{aligned}$$

Full proof in [Vincent et al'10]

Auto-encoders maximize the mutual information

- Find encoder with parameter $\boldsymbol{\theta}$ to maximize information

$$\begin{aligned}\operatorname*{argmax}_{\boldsymbol{\theta}} I(\boldsymbol{x};\boldsymbol{h}) &= \operatorname*{argmax}_{\boldsymbol{\theta}} H(\boldsymbol{x}) - H(\boldsymbol{x}|\boldsymbol{h}) \\ &= \operatorname*{argmax}_{\boldsymbol{\theta}} -H(\boldsymbol{x}|\boldsymbol{h}) \\ &= \operatorname*{argmax}_{\boldsymbol{\theta}} \mathbb{E}_{p(\boldsymbol{x},\boldsymbol{h})} \log p(\boldsymbol{x}|\boldsymbol{h})\end{aligned}$$

- Approximate $p(\boldsymbol{x}|\boldsymbol{h})$ with a parametric decoder and use a deterministic encoder. The log-likelihood is:

$$\operatorname*{argmax}_{\boldsymbol{\theta},\boldsymbol{\theta}'} \mathbb{E}_{p(\boldsymbol{x})} \log p_{\text{decoder}}(\boldsymbol{x}|\boldsymbol{h} = f_{\boldsymbol{\theta}}(\boldsymbol{x}); \boldsymbol{\theta}')$$

Full proof in [Vincent et al'10]

Gaussian <-> squared Euclidean norm

- Assume that the likelihood has Gaussian density:

$$p_{\text{decoder}}(\boldsymbol{x}|\boldsymbol{h} = f_{\boldsymbol{\theta}}(\boldsymbol{x}); \boldsymbol{\theta}') = N(g_{\boldsymbol{\theta}'}(f_{\boldsymbol{\theta}}(\boldsymbol{x})), \sigma^2 I)$$

Gaussian <-> squared Euclidean norm

- Assume that the likelihood has Gaussian density:

$$p_{\text{decoder}}(\boldsymbol{x}|\boldsymbol{h} = f_{\boldsymbol{\theta}}(\boldsymbol{x}); \boldsymbol{\theta}') = N(g_{\boldsymbol{\theta}'}(f_{\boldsymbol{\theta}}(\boldsymbol{x})), \sigma^2 I)$$

- Then

$$\mathbb{E}_{p(\boldsymbol{x})} \log p_{\text{decoder}}(\boldsymbol{x}|\boldsymbol{h}; \boldsymbol{\theta}')$$
$$= \text{Const} - \frac{1}{2\sigma^2} \mathbb{E}_{p(\boldsymbol{x})} \|\boldsymbol{x} - g_{\boldsymbol{\theta}'}(f_{\boldsymbol{\theta}}(\boldsymbol{x}))\|_2^2$$

Gaussian <-> squared Euclidean norm

- Assume that the likelihood has Gaussian density:

$$p_{\text{decoder}}(\boldsymbol{x}|\boldsymbol{h} = f_{\boldsymbol{\theta}}(\boldsymbol{x}); \boldsymbol{\theta}') = N(g_{\boldsymbol{\theta}'}(f_{\boldsymbol{\theta}}(\boldsymbol{x})), \sigma^2 I)$$

- Then

$$\begin{aligned}
&\mathbb{E}_{p(\boldsymbol{x})} \log p_{\text{decoder}}(\boldsymbol{x}|\boldsymbol{h}; \boldsymbol{\theta}') \\
&= \text{Const} - \frac{1}{2\sigma^2} \mathbb{E}_{p(\boldsymbol{x})} \|\boldsymbol{x} - g_{\boldsymbol{\theta}'}(f_{\boldsymbol{\theta}}(\boldsymbol{x}))\|_2^2
\end{aligned}$$

- Therefore, connecting the mutual information,

$$\max_{\boldsymbol{\theta}} I(\boldsymbol{x}; \boldsymbol{h}) \approx \min_{\boldsymbol{\theta}, \boldsymbol{\theta}'} \mathbb{E}_{p(\boldsymbol{x})} \|\boldsymbol{x} - g_{\boldsymbol{\theta}'}(f_{\boldsymbol{\theta}}(\boldsymbol{x}))\|_2^2$$

Max information is not enough

- Question: what if encoder and decoder are functions so flexible that can learn an identity map?

Max information is not enough

- Question: what if encoder and decoder are functions so flexible that can learn an identity map?
- Keep in mind: our goal is to learn an encoder such that we get a useful representation of the input
- **We need additional constraints** !

Max information is not enough

- Question: what if encoder and decoder are functions so flexible that can learn an identity map?
- Keep in mind: our goal is to learn an encoder such that we get a useful representation of the input
- **We need additional constraints** !
- **Undercomplete auto-encoder**: the code dimension is less than the input dimension

Max information is not enough

- Question: what if encoder and decoder are functions so flexible that can learn an identity map?
- Keep in mind: our goal is to learn an encoder such that we get a useful representation of the input
- **We need additional constraints** !
- **Undercomplete auto-encoder**: the code dimension is less than the input dimension

- PCA is an undercomplete auto-encoder with square Euclidean distance as loss and f, g linear functions

Learned filters: over vs. undercomplete

[Vincent et al.'10]

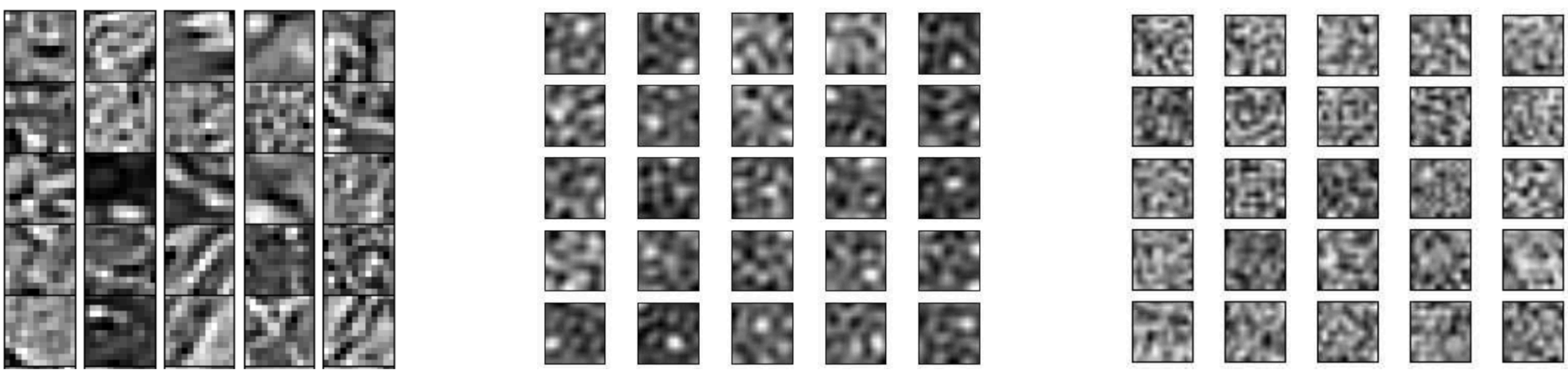


Figure 5: Regular autoencoder trained on natural image patches. *Left:* some of the 12×12 image patches used for training. *Middle:* filters learnt by a regular *under-complete* autoencoder (50 hidden units) using tied weights and L2 reconstruction error. *Right:* filters learnt by a regular *over-complete* autoencoder (200 hidden units). The under-complete autoencoder appears to learn rather uninteresting local blob detectors. Filters obtained in the over-complete case have no recognizable structure, looking entirely random.

Example: deep auto-encoder in Keras

```
input_img = Input(shape=(784,))
encoded = Dense(128, activation='relu')(input_img)
encoded = Dense(64, activation='relu')(encoded)
encoded = Dense(32, activation='relu')(encoded)

decoded = Dense(64, activation='relu')(encoded)
decoded = Dense(128, activation='relu')(decoded)
decoded = Dense(784, activation='sigmoid')(decoded)
```

[https://goo.gl/9kCxqz Chollet]

- 3-layer encoder and 3-layer decoder, under complete.
- The output is sigmoid because we want to get black& white images (on MNIST). Other activations are ReLU.
- Dense (=fully connected) layers. But they can be CNN.

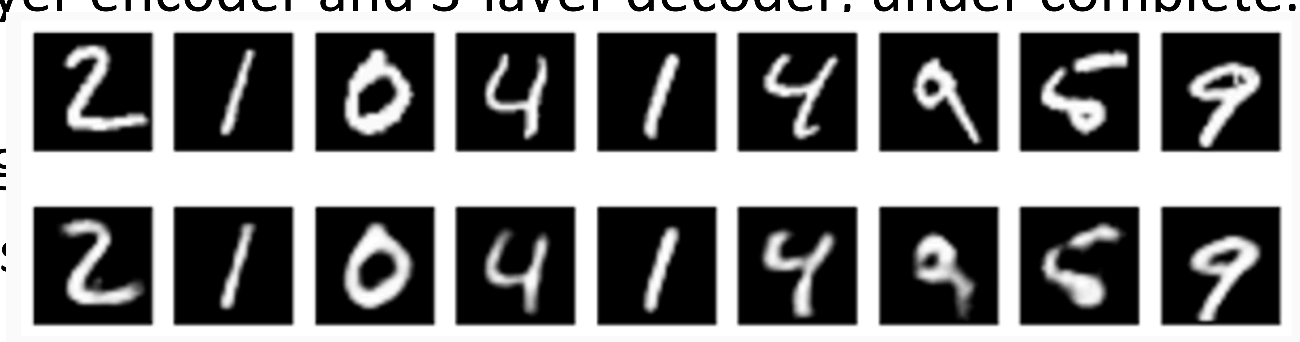
Example: deep auto-encoder in Keras

```
input_img = Input(shape=(784,))
encoded = Dense(128, activation='relu')(input_img)
encoded = Dense(64, activation='relu')(encoded)
encoded = Dense(32, activation='relu')(encoded)

decoded = Dense(64, activation='relu')(encoded)
decoded = Dense(128, activation='relu')(decoded)
decoded = Dense(784, activation='sigmoid')(decoded)
```

[https://goo.gl/9kCxqz Chollet]

- 3-layer encoder and 3-layer decoder, under complete.
- The … white
  imag…
- Den…



Regularized auto-encoders

- Alternative to undercomplete: use a regularizer Ω to constraint the objective

$$\frac{1}{N}\sum_{n=1}^{N} L(\boldsymbol{x}_n, g(f(\boldsymbol{x}_n))) + \Omega(\boldsymbol{h})$$

Regularized auto-encoders

- Alternative to undercomplete: use a regularizer Ω to constraint the objective

$$\frac{1}{N}\sum_{n=1}^{N} L(\boldsymbol{x}_n, g(f(\boldsymbol{x}_n))) + \Omega(\boldsymbol{h})$$

- In supervised learning regularizers **reduce the capacity** of the model **to overfit** the training set
- In unsupervised learning we need them to be **invariant to nuisance** factors (=irrelevant noise) in the data… it is actually the same thing!

Regularized auto-encoders

- Alternative to undercomplete: use a regularizer Ω to constraint the objective

$$\frac{1}{N}\sum_{n=1}^{N} L(\boldsymbol{x}_n, g(f(\boldsymbol{x}_n))) + \Omega(\boldsymbol{h})$$

- In supervised learning regularizers **reduce the capacity** of the model **to overfit** the training set
- In unsupervised learning we need them to be **invariant to nuisance** factors (=irrelevant noise) in the data… it is actually the same thing!
- Interpretation: those are **bottlenecks** that allows us to **compress** the data into a useful representation, robust to irrelevant variations of the training data

Sparse auto-encoder

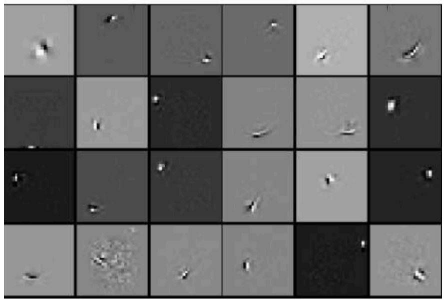
- Sparsity-inducing regularizer:

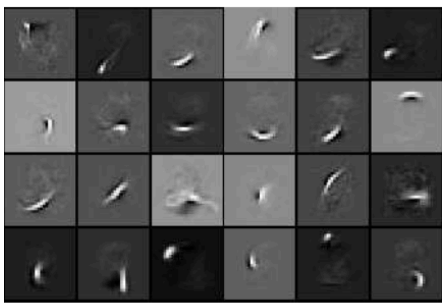
$$\frac{1}{N}\sum_{n=1}^{N} L(\boldsymbol{x}_n, g(f(\boldsymbol{x}_n))) + \lambda\|\boldsymbol{h}\|_1$$

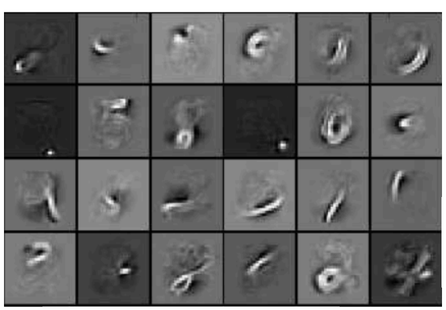
- Analogue to use the L1-norm in supervised learning. Effect: pushes many components to **exact 0**
- Probabilistic interpretation: train the auto-encoder with maximum likelihood with a Laplace prior on the code $\boldsymbol{h}$ (the latent variable):

$$p(\boldsymbol{h}) = \frac{\lambda}{2} e^{-\lambda\|\boldsymbol{h}\|_1}$$

Filters of a sparse auto-encoder



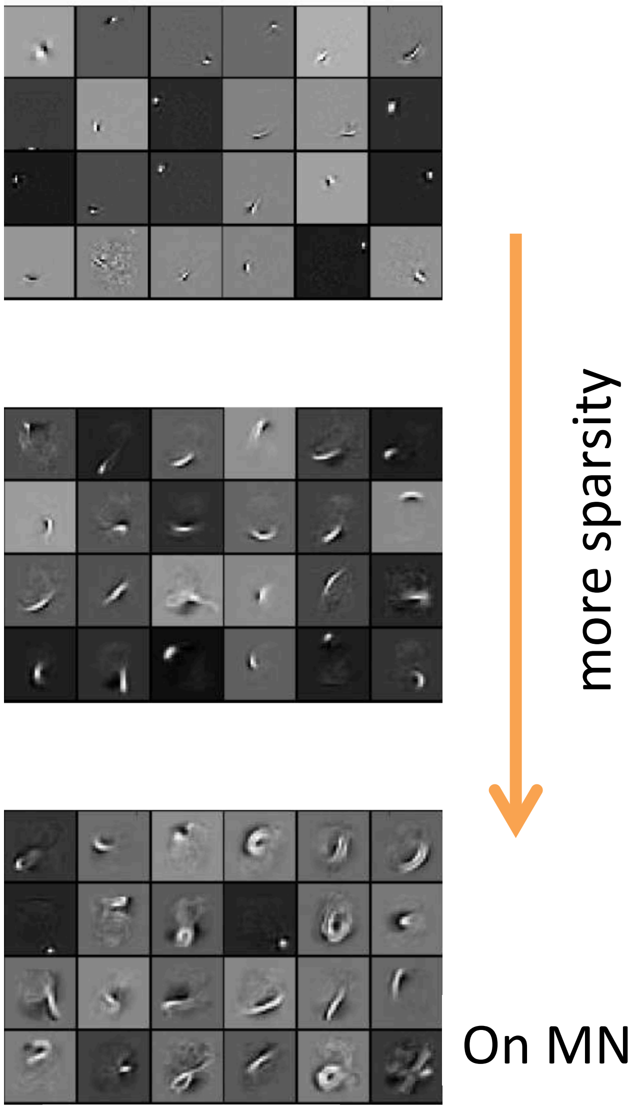




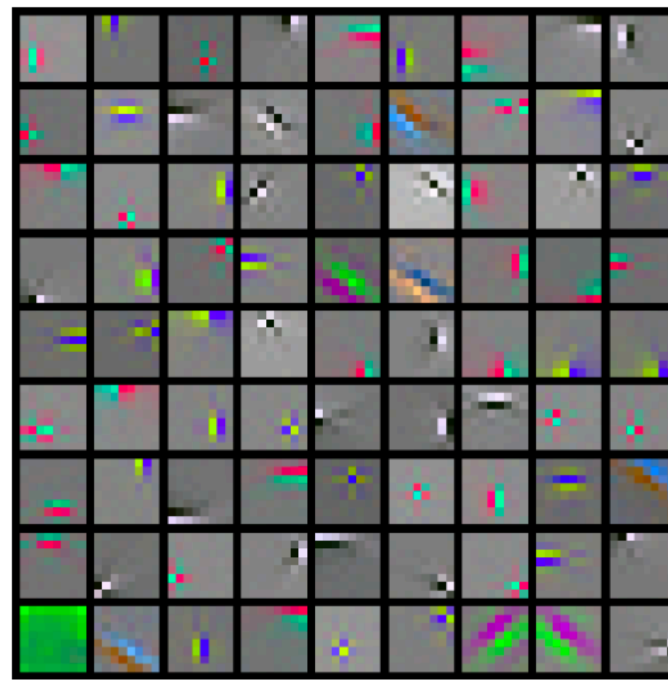
On MNIST

[Makhazani&Frey'14]

Filters of a sparse auto-encoder



On MNIST



On CIFAR10

[Makhazani&Frey'14]

Application: dimensionality reduction

- Fix the representation size to M. Then we can reduce the dimensionality of the data to M.
- Advantages:
  - Less memory
  - Less time consumption for any following algorithm (e.g. supervised learning)

Application: dimensionality reduction

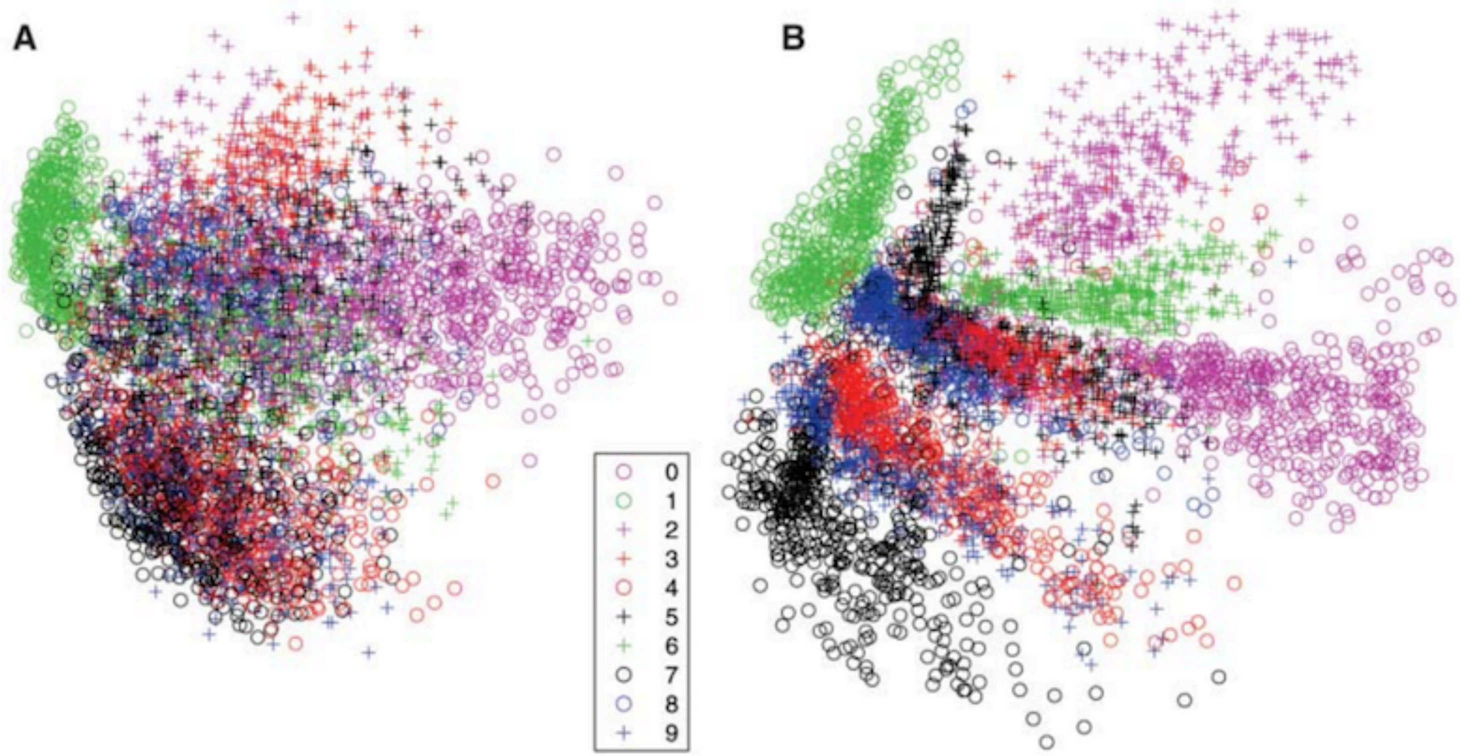
- Fix the representation size to M. Then we can reduce the dimensionality of the data to M.
- Advantages:
  - Less memory
  - Less time consumption for any following algorithm (e.g. supervised learning)

- With respect to PCA:
  - Pro: more meaningful representation, less information discarded
  - Cons: harder and slower to train

Application: visualization

- Dimensionality reduction onto 2D or 3D for visualization

Fig. 3. (**A**) The two-dimensional codes for 500 digits of each class produced by taking the first two principal components of all 60,000 training images. (**B**) The two-dimensional codes found by a 784-1000-500-250-2 autoencoder. For an alternative visualization, see (*8*).



[Hinton&Salakhutdinov'06]

Overview

- Introduction, manifolds, PCA (Goodfellow 5.11.3, 13.5)
- **Auto-encoders** (14)
  - Objective, undercomplete / regularized auto-encoders
  - **Denoising auto-encoders, contractive auto-encoders**
- Generative models (parts of 20)
  - Variational auto-encoder (20.9, 20.10.3)
  - Generative adversarial network (20.10.4, 20.10.6)
  - PixelRNN, models evaluation (20.10.7, 20.14)

Denoising auto-encoder

- Introduce a bottleneck by injecting noise in the input

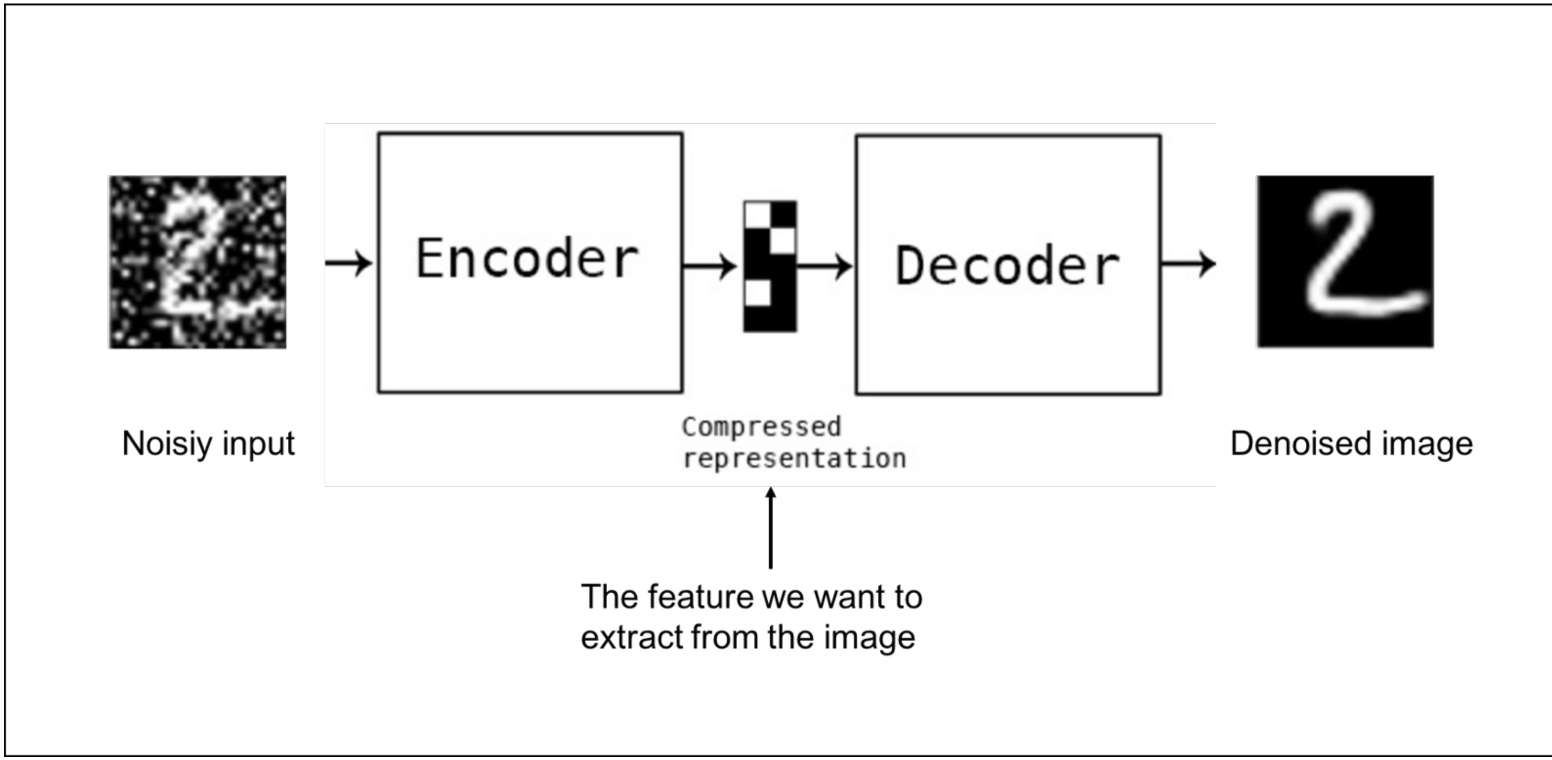
$$\frac{1}{N}\sum_{n=1}^{N} L(\boldsymbol{x}_n, g(f(\tilde{\boldsymbol{x}}_n)))$$

- Noise could be additive Gaussian or Dropout (=part of the input is set to 0 uniformly at random)

Denoising auto-encoder

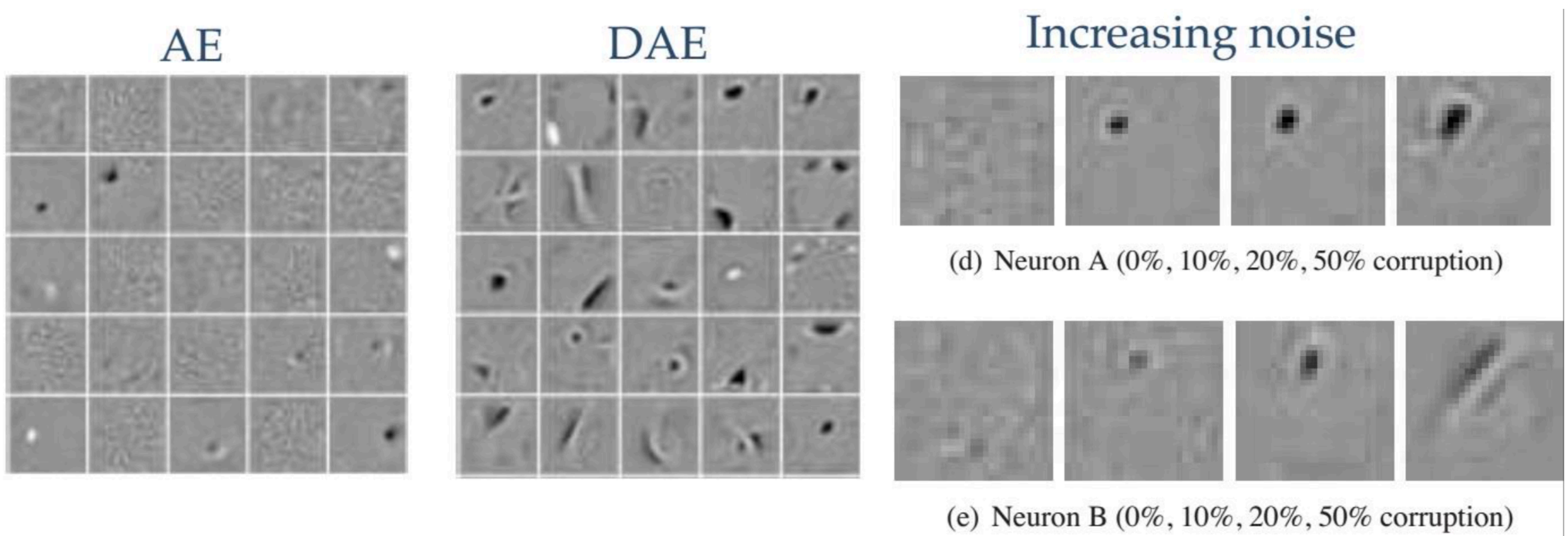
- Introduce a bottleneck by injecting noise in the input
$$\frac{1}{N}\sum_{n=1}^{N} L(\boldsymbol{x}_n, g(f(\tilde{\boldsymbol{x}}_n)))$$
- Noise could be additive Gaussian or Dropout (=part of the input is set to 0 uniformly at random)
- Auto-encoder **learns a denoising map** to reconstruct the original input
- Auto-encoder can be overcomplete. In fact, this is an **implicit regularizer**

Denoising auto-encoder



Denoising auto-encoder

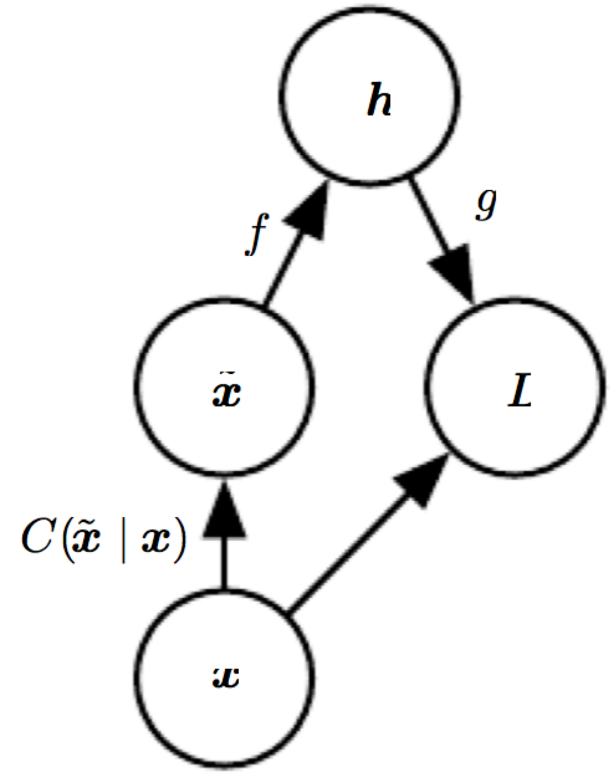
- More noise => network is forced to learn more robust representation; feature resembles strokes and bubbles more often



Denoising auto-encoder revisited

- Introduce the noisy transition as a stochastic operation in the computational graph

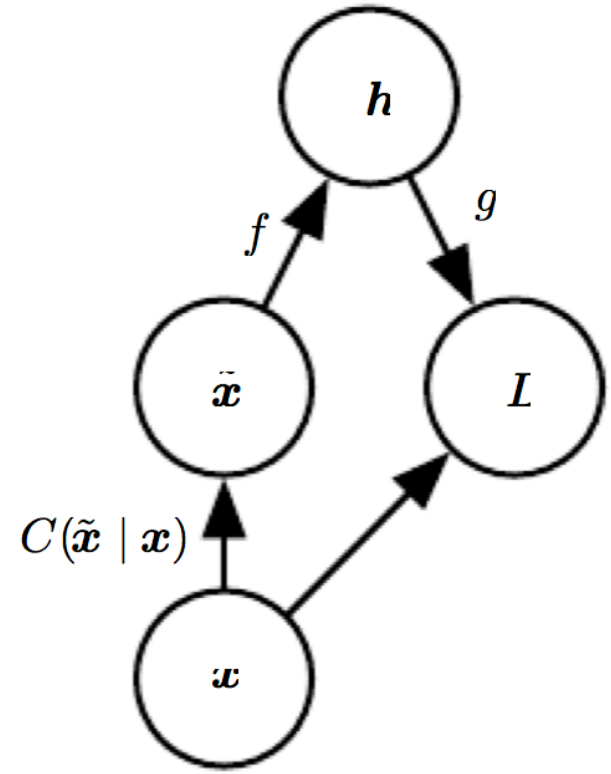
1. Sample x from the data
2. Sample a corrupted version $\tilde{x}$ by $C(\tilde{x}|x)$
3. Train the auto-encoder to reconstruct x



Denoising auto-encoder revisited

- Introduce the noisy transition as a stochastic operation in the computational graph

1. Sample $\boldsymbol{x}$ from the data
2. Sample a corrupted version $\tilde{\boldsymbol{x}}$ by $C(\tilde{\boldsymbol{x}}|\boldsymbol{x})$
3. Train the auto-encoder to reconstruct $\boldsymbol{x}$



- The loss function as negative log likelihood is:

$$-\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\mathbb{E}_{\tilde{\boldsymbol{x}}\sim C(\tilde{\boldsymbol{x}}|\boldsymbol{x})} \log p_{\text{decoder}}(\boldsymbol{x}|\boldsymbol{h} = f(\tilde{\boldsymbol{x}}))$$

Denoising auto-encoders learn to map onto the manifold

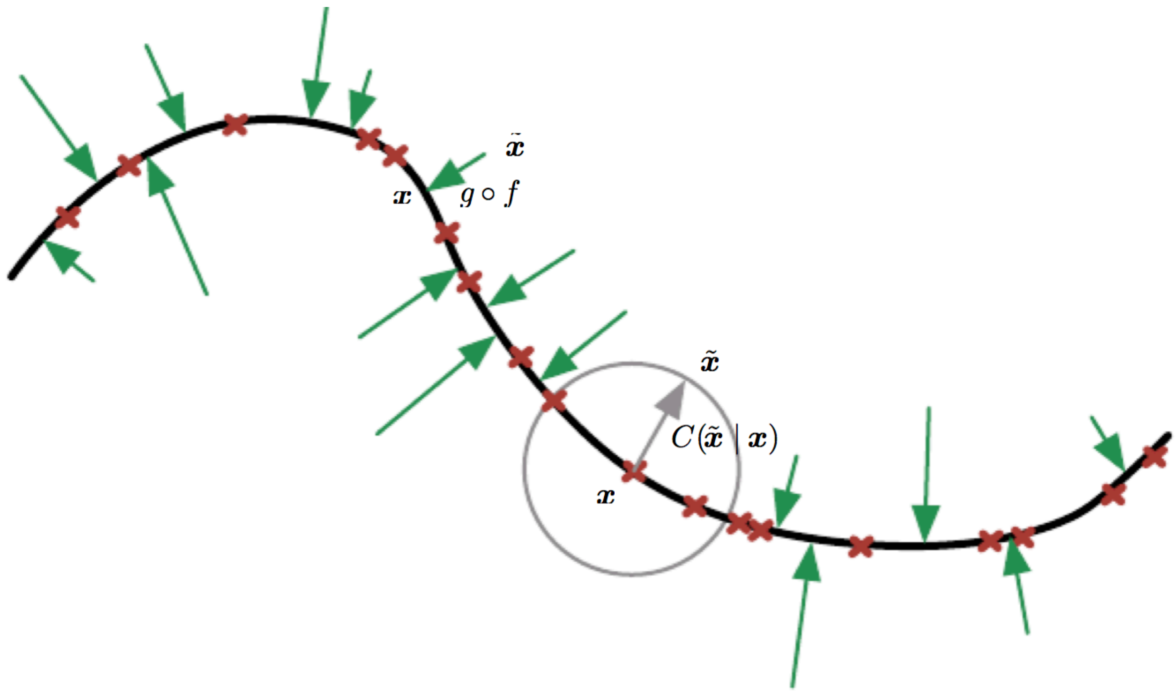


Figure 14.4: A denoising autoencoder is trained to map a corrupted data point $\tilde{\boldsymbol{x}}$ back to the original data point $\boldsymbol{x}$. We illustrate training examples $\boldsymbol{x}$ as red crosses lying near a low-dimensional manifold illustrated with the bold black line. We illustrate the corruption process $C(\tilde{\boldsymbol{x}} \mid \boldsymbol{x})$ with a gray circle of equiprobable corruptions. A gray arrow demonstrates how one training example is transformed into one sample from this corruption process. When the denoising autoencoder is trained to minimize the average of squared errors $||g(f(\tilde{\boldsymbol{x}})) - \boldsymbol{x}||^2$, the reconstruction $g(f(\tilde{\boldsymbol{x}}))$ estimates $\mathbb{E}_{\mathbf{x},\tilde{\mathbf{x}} \sim p_{\text{data}}(\mathbf{x}) C(\tilde{\mathbf{x}}|\mathbf{x})}[\mathbf{x} \mid \tilde{\boldsymbol{x}}]$. The vector $g(f(\tilde{\boldsymbol{x}})) - \tilde{\boldsymbol{x}}$ points approximately towards the nearest point on the manifold, since $g(f(\tilde{\boldsymbol{x}}))$ estimates the center of mass of the clean points $\boldsymbol{x}$ which could have given rise to

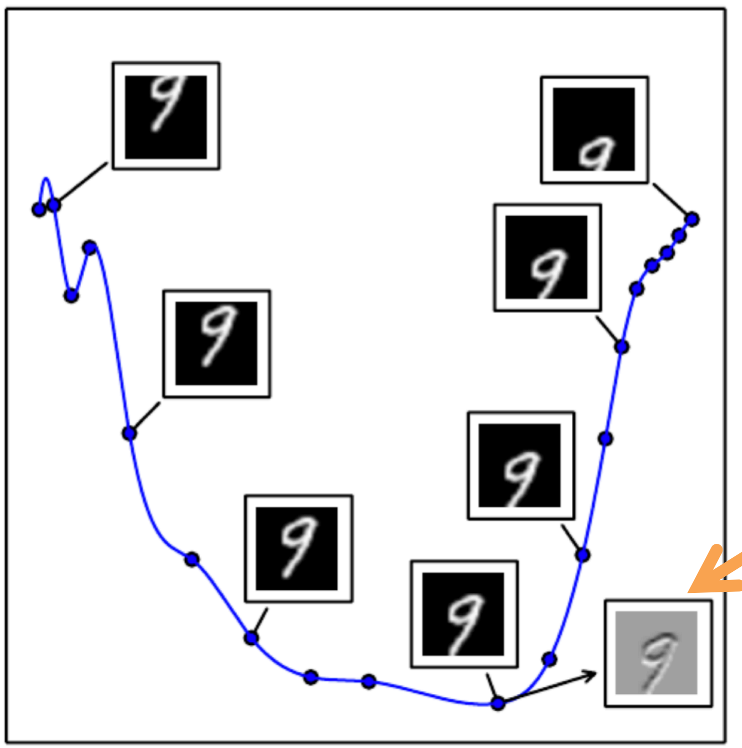
Application of denoising auto-encoder: image denoising

[https://goo.gl/9kCxqz Chollet]

- Apply the **whole auto-encoder** to real images that are affected by noise => output denoised images.
- To work well, the real noise has to be similar to Gaussian though

Manifolds and tangent planes

- At each point x of a d-dimensional manifold, a **tangent plane** is given by d basis vectors spanning the local directions of variation of the manifold



Tangent plane in pixel space
Grey pixel: no variation
Black/white: large variation

Auto-encoders and manifolds

- Do auto-encoders learn the manifold structure?

- Training combines two forces:
  - **Reconstruction**: represent x by $\boldsymbol{h} = f(\boldsymbol{x})$ such that $\boldsymbol{x}$ can be decoded through $g(\boldsymbol{h})$
  - **Limited capacity**: the encoder $\boldsymbol{h} = f(\boldsymbol{x})$ cannot represent any possible function

- Neither would be enough alone

Auto-encoders and manifolds

- Compromise:
  - The auto-encoder can only afford to model the variations needed to reconstruct the training data

- If the data concentrates near a manifold, *only the **variations tangent** to the manifold around* x *need to correspond to changes in* $\boldsymbol{h} = f(\boldsymbol{x})$
- Auto-encoders learn a representation that captures a local coordinate system of the manifold

Contractive auto-encoder

More explicit model of the manifold in the objective:

$$\frac{1}{N}\sum_{n=1}^{N} L(\boldsymbol{x}_n, g(f(\boldsymbol{x}_n))) + \Omega(\boldsymbol{h}, \boldsymbol{x})$$

where the regularizer is

$$\Omega(\boldsymbol{h}, \boldsymbol{x}) = \lambda \left\| \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} \right\|_F^2$$

It penalizes the squared Frobenius norm of the Jacobian of the encoder

=> it forces the encoder to learn a representation that doesn't change much around the training examples

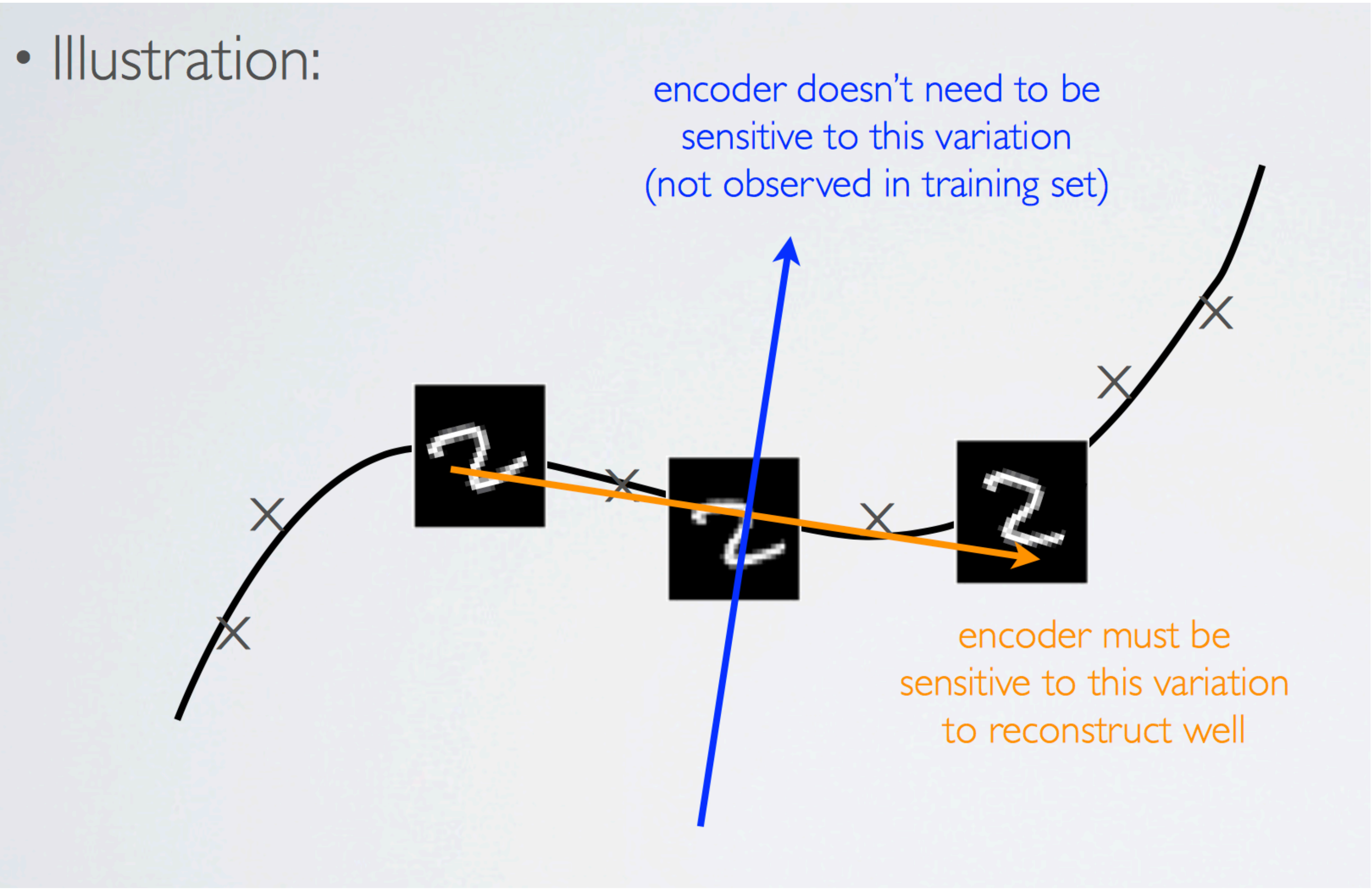
Contractive auto-encoder

- The compromise here:
  - **Contractive**: resist to local perturbations of the input by squashing their representation through the encoder.
  - But at the same time minimize the **reconstruction error**

  => **Therefore**: only the irrelevant directions of variation of the input will be contracted by the encoder

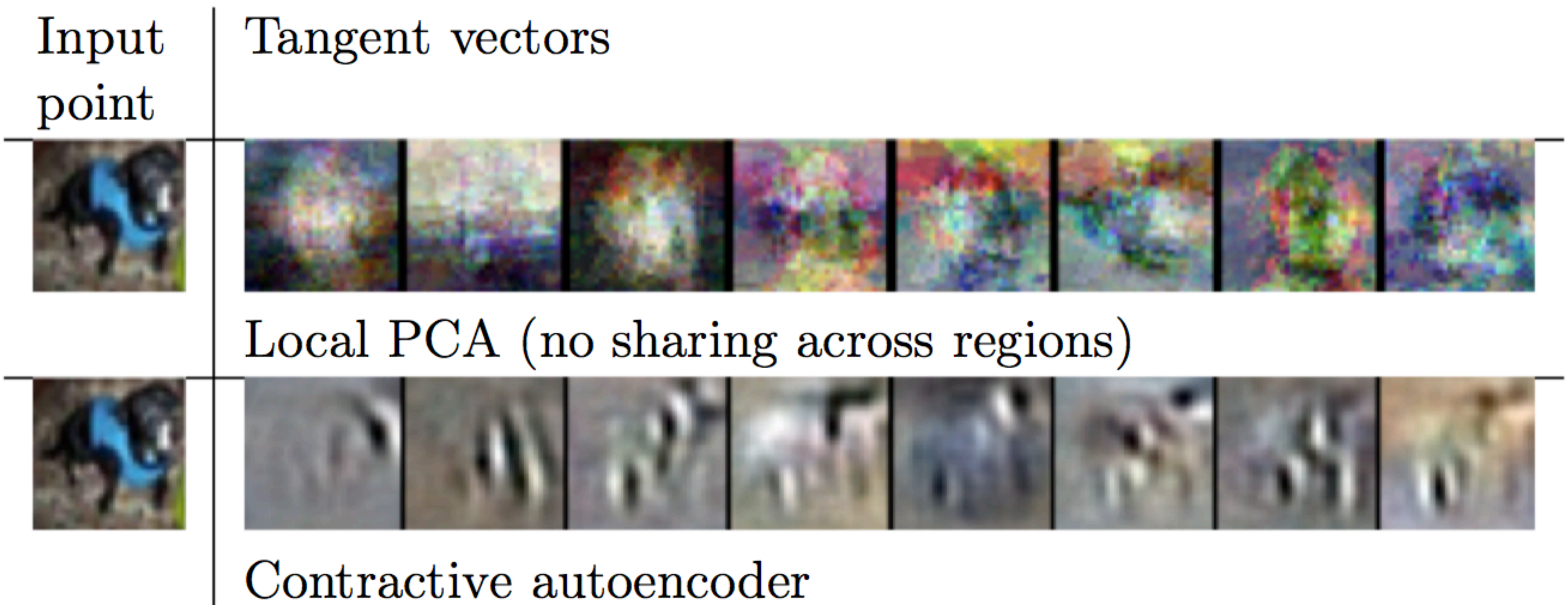
- Jacobian is expensive, but we can compute it by auto-diff tools as usual. A finite difference approximation works too.

Contractive auto-encoder



Does it work?
Check the tangent planes

- Given an image x, compute the Jacobian at x.
- Obtain its eigenvectors at x by SVD decomposition. Those are the tangent planes (they are in pixel space):



Denoising vs. contractive auto-encoders

Both perform well but

- Denoising: simpler to implement
  - Few lines of code more than standard auto-encoder
  - No need to compute Jacobian
- Contractive: gradient is deterministic
  - More stable, easier to monitor convergence

- They penalise different things:
  - Denoising: reconstruction (g + f) robust to noise
  - Contractive: representation (f) robust to noise

Application: unsupervised feature learning

1. Train auto-encoder on unlabeled data
2. Add classification layer to the encoder
3. Fine-tune all with supervised learning

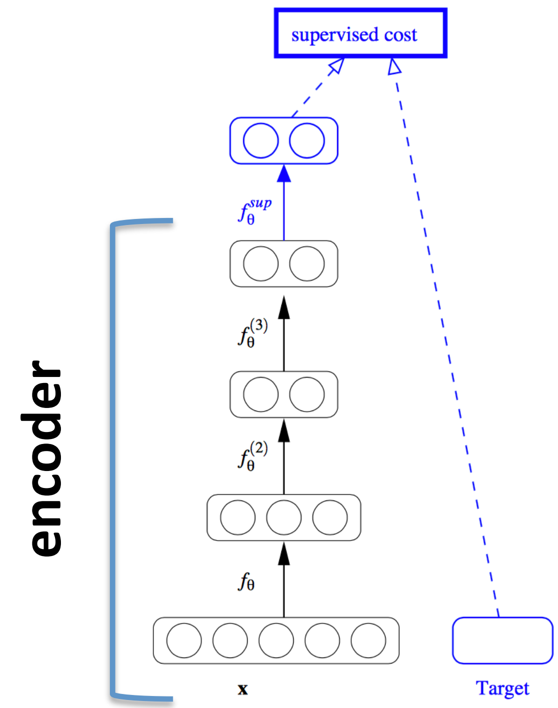


Figure 4: Fine-tuning of a deep network for classification. After training a stack of encoders as explained in the previous figure, an output layer is added on top of the stack. The parameters of the whole system are fine-tuned to minimize the error in predicting the supervised target (e.g., class), by performing gradient descent on a supervised cost.

Application: unsupervised feature learning

1. Train auto-encoder on unlabeled data
2. Add classification layer to the encoder
3. Fine-tune all with supervised learning

We learn the features up to the second last layer

See **transfer learning**

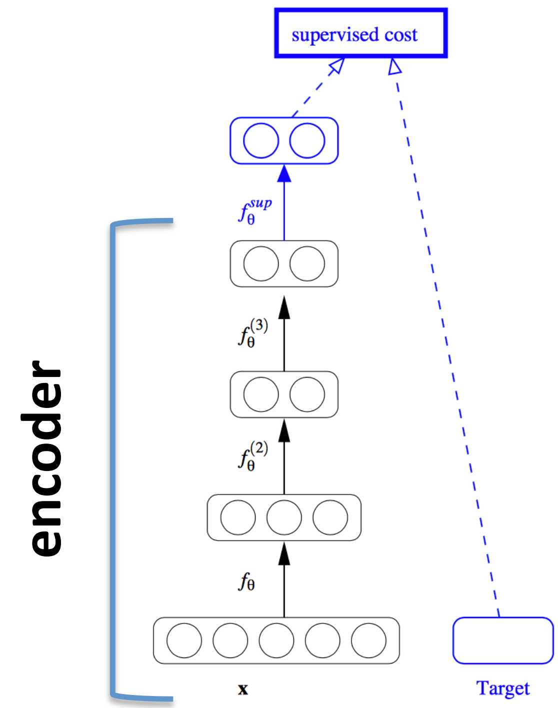
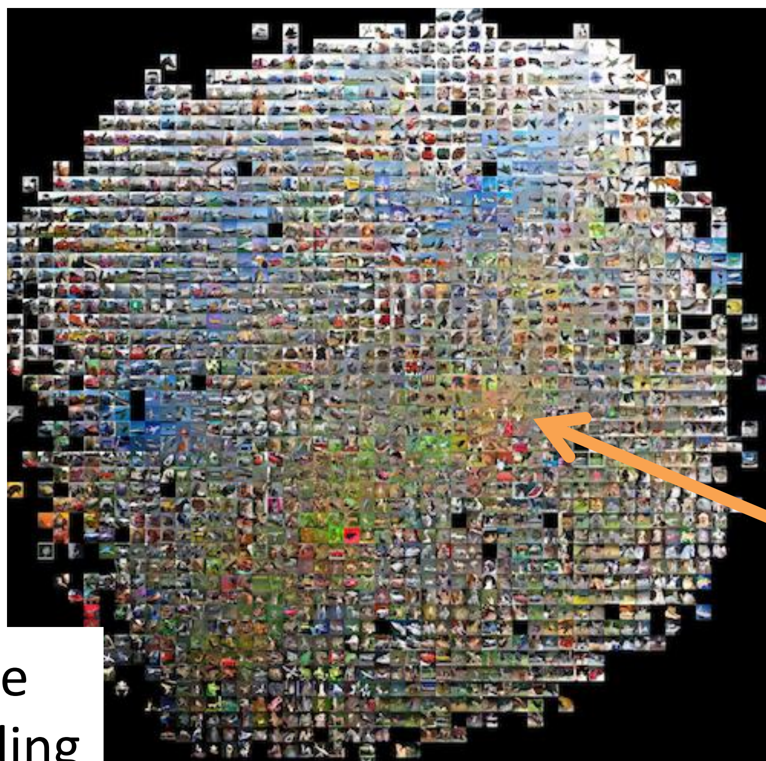


Figure 4: Fine-tuning of a deep network for classification. After training a stack of encoders as explained in the previous figure, an output layer is added on top of the stack. The parameters of the whole system are fine-tuned to minimize the error in predicting the supervised target (e.g., class), by performing gradient descent on a supervised cost.

Application: semantic hashing

- Typical task of information retrieval: given a database of images and a image query, return the most similar image in the database. Image search.



database embedding

query: return the most similar image

Material and contact

Lectures material based on

- Goodfellow’s Deep Learning book
- Efstratios Gavves’s slides from last year
- Larochelle deep learning course https://goo.gl/bvNPDt
- Auto-encoders tutorial in Keras https://goo.gl/9kCxqz
- Durk Kingma PhD thesis (recommended) https://www.dropbox.com/s/v6ua3d9yt44vgb3/cover_and_thesis.pdf?dl=1
- Goodfellow tutorial on GAN, NIPS 2016 (recommended)

For questions & Master thesis projects: g.patrini@uva.nl

Other references (autoencoders)

- Hinton & Salakhutdinov, Semantic Hashing 2006
- Vincent et al., Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion, JMLR 2010
- Rifai et al., Contractive Auto-Encoders: Explicit Invariance During Feature Extraction, ICML 11
- LeCun & Ranzato, Deep learning tutorial, ICML 13 https://goo.gl/37GbPS
- Makhezani & Frey, k-sparse autoencoders, ICLR14