10417/10617
Intermediate Deep Learning:
Fall2023

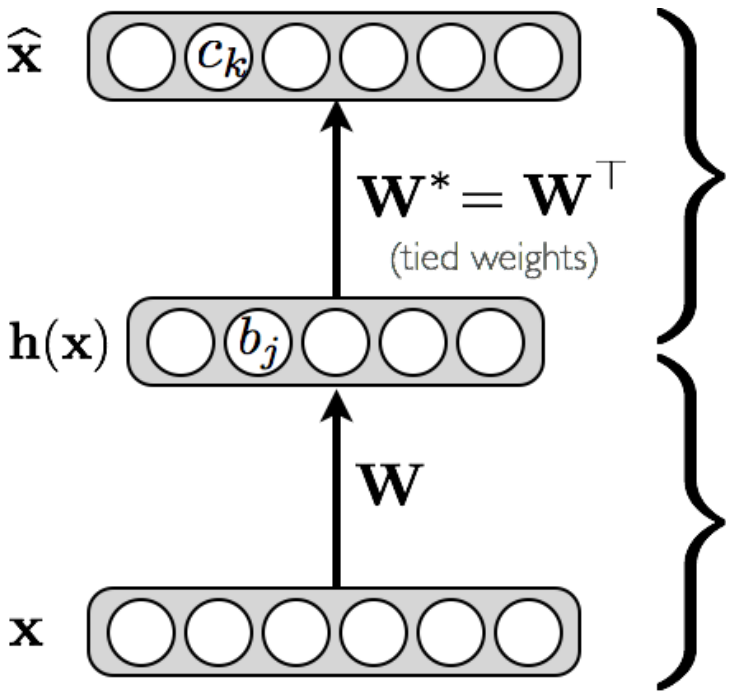
Russ Salakhutdinov

Machine Learning Department
rsalakhu@cs.cmu.edu

Autoencoders

Autoencoders

• Feed-forward neural network trained to reproduce its input at the output layer



Decoder

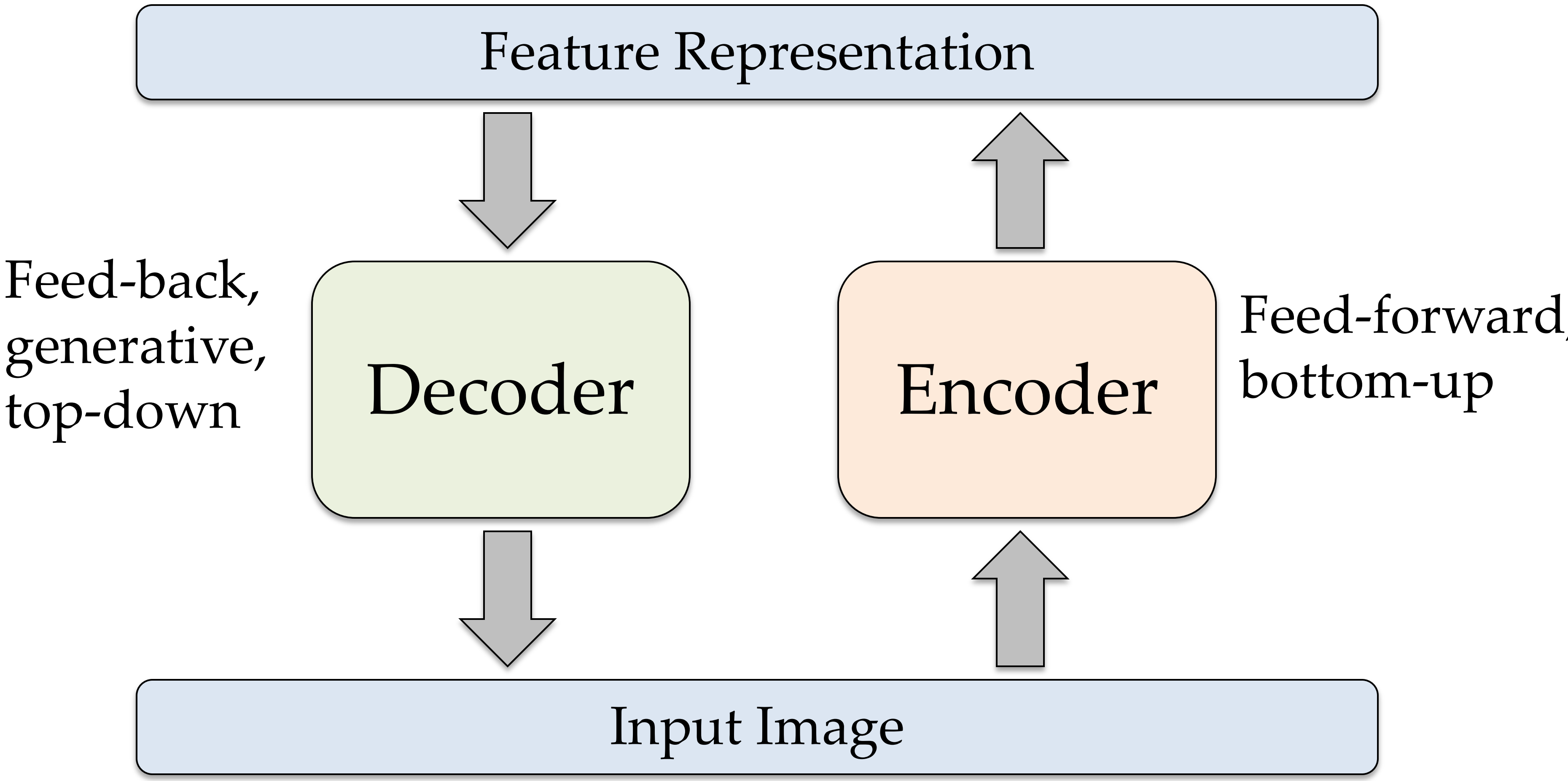
$$
\begin{aligned}
\widehat{\mathbf{x}} &= o(\widehat{\mathbf{a}}(\mathbf{x})) \\
&= \mathrm{sigm}(\mathbf{c} + \mathbf{W}^*\mathbf{h}(\mathbf{x}))
\end{aligned}
$$

For binary units

Encoder

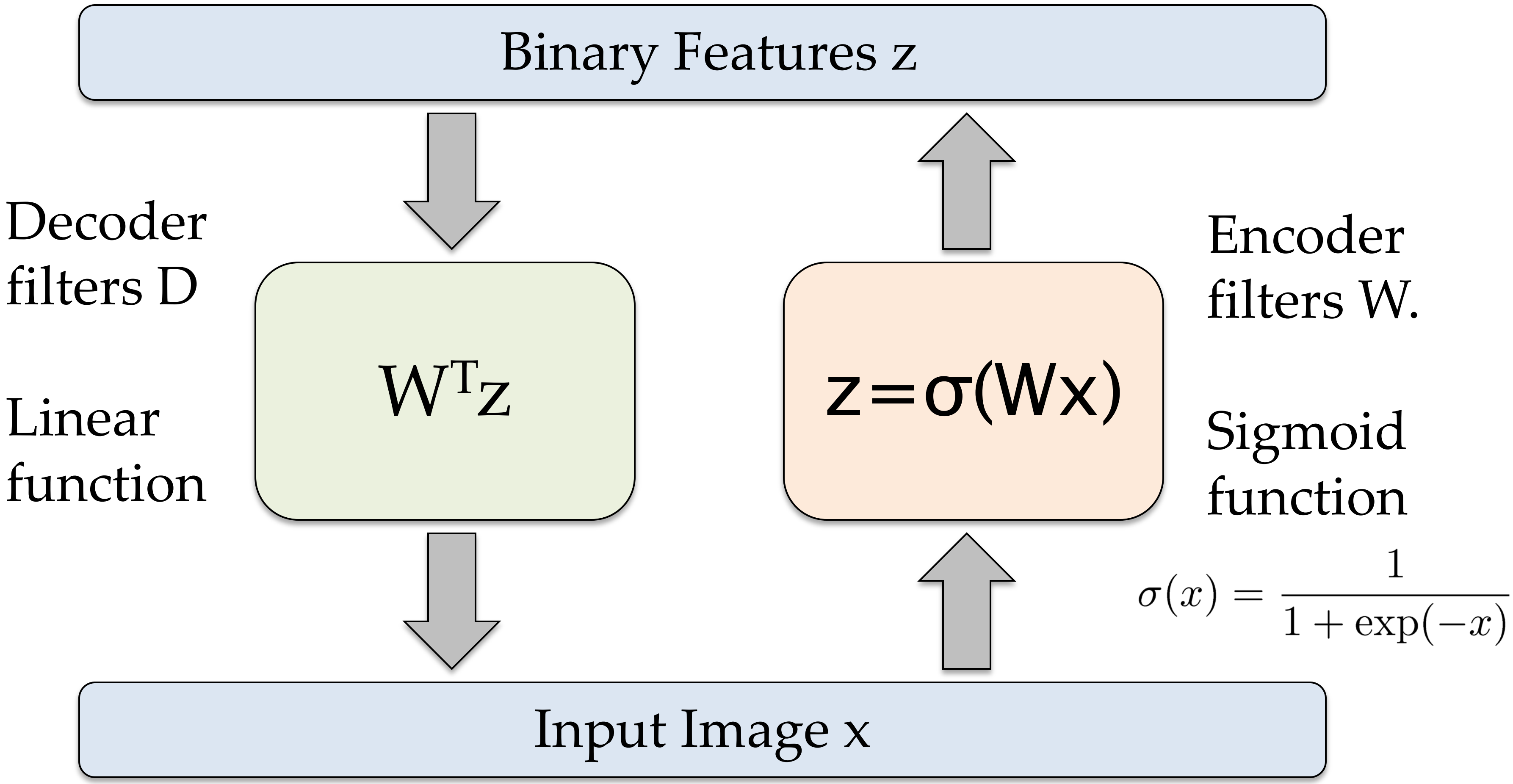
$$
\begin{aligned}
\mathbf{h}(\mathbf{x}) &= g(\mathbf{a}(\mathbf{x})) \\
&= \mathrm{sigm}(\mathbf{b} + \mathbf{W}\mathbf{x})
\end{aligned}
$$

Autoencoders

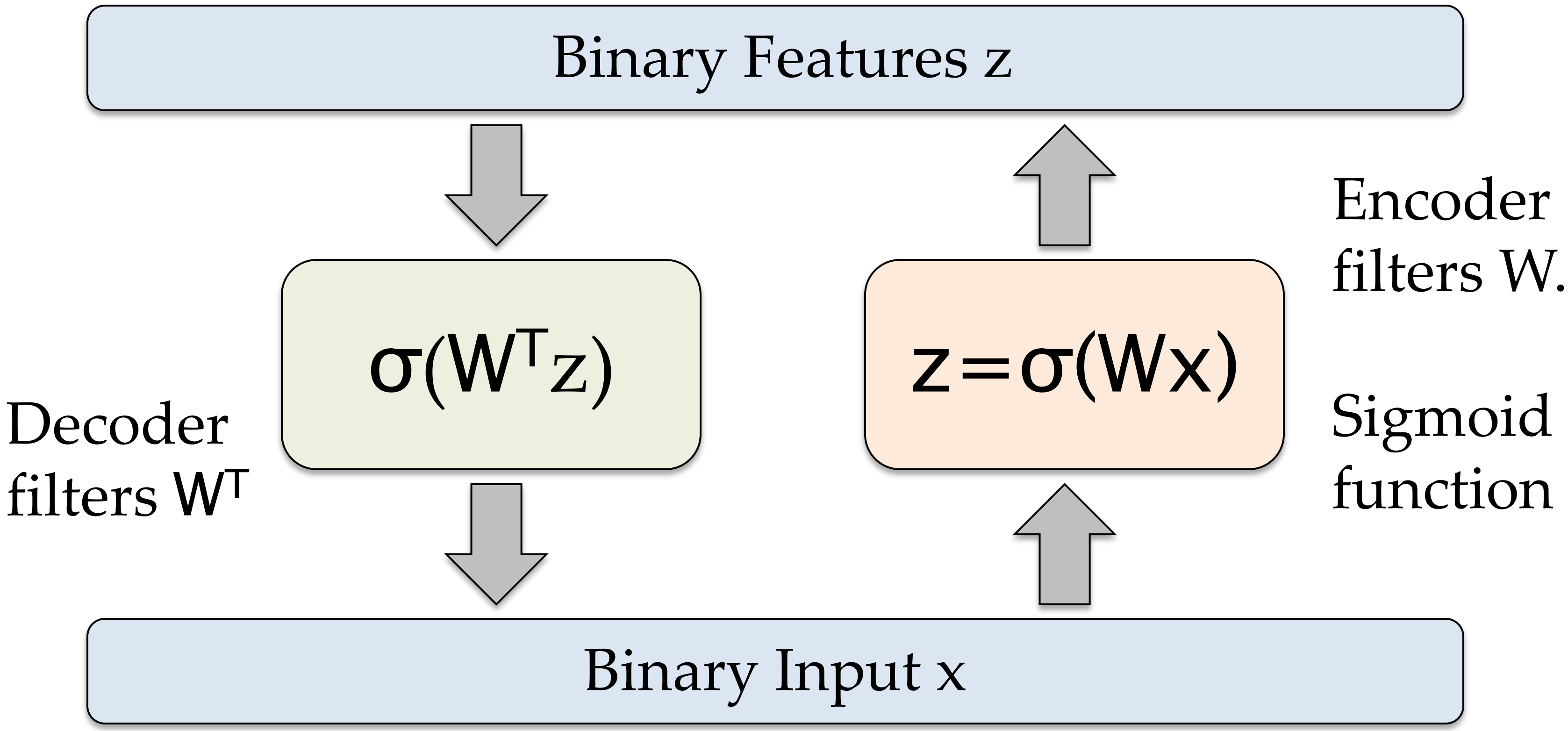


- Details of what goes insider the encoder and decoder matter!
- Need constraints to avoid learning an identity.

Autoencoders



Another Autoencoder Model



- Need additional constraints to avoid learning an identity.
- Relates to Restricted Boltzmann Machines.
- Encoder and Decoder filters can be different.

Loss Function

- Loss function for binary inputs

$$l(f(\mathbf{x})) = -\sum_k \left(x_k \log(\widehat{x}_k) + (1 - x_k) \log(1 - \widehat{x}_k)\right)$$

  - Cross-entropy error function (reconstruction loss) $f(\mathbf{x}) \equiv \widehat{\mathbf{x}}$

- Loss function for real-valued inputs

$$l(f(\mathbf{x})) = \frac{1}{2} \sum_k (\widehat{x}_k - x_k)^2$$

  - sum of squared differences (reconstruction loss)
  - we use a linear activation function at the output

Loss Function

- For both cases, the gradient has a very simple form:

$$\nabla_{\widehat{\mathbf{a}}(\mathbf{x}^{(t)})} l(f(\mathbf{x}^{(t)})) = \widehat{\mathbf{x}}^{(t)} - \mathbf{x}^{(t)} \qquad f(\mathbf{x}) \equiv \widehat{\mathbf{x}}$$

- Parameter gradients are obtained by backpropagating the gradient $\nabla_{\widehat{\mathbf{a}}(\mathbf{x}^{(t)})} l(f(\mathbf{x}^{(t)}))$ like in a regular network
  - ➢ important: when using tied weights ($\mathbf{W}^* = \mathbf{W}^\top$), $\nabla_{\mathbf{W}} l(f(\mathbf{x}^{(t)}))$ is the sum of two gradients
  - ➢ this is because $\mathbf{W}$ is present in the encoder and in the decoder

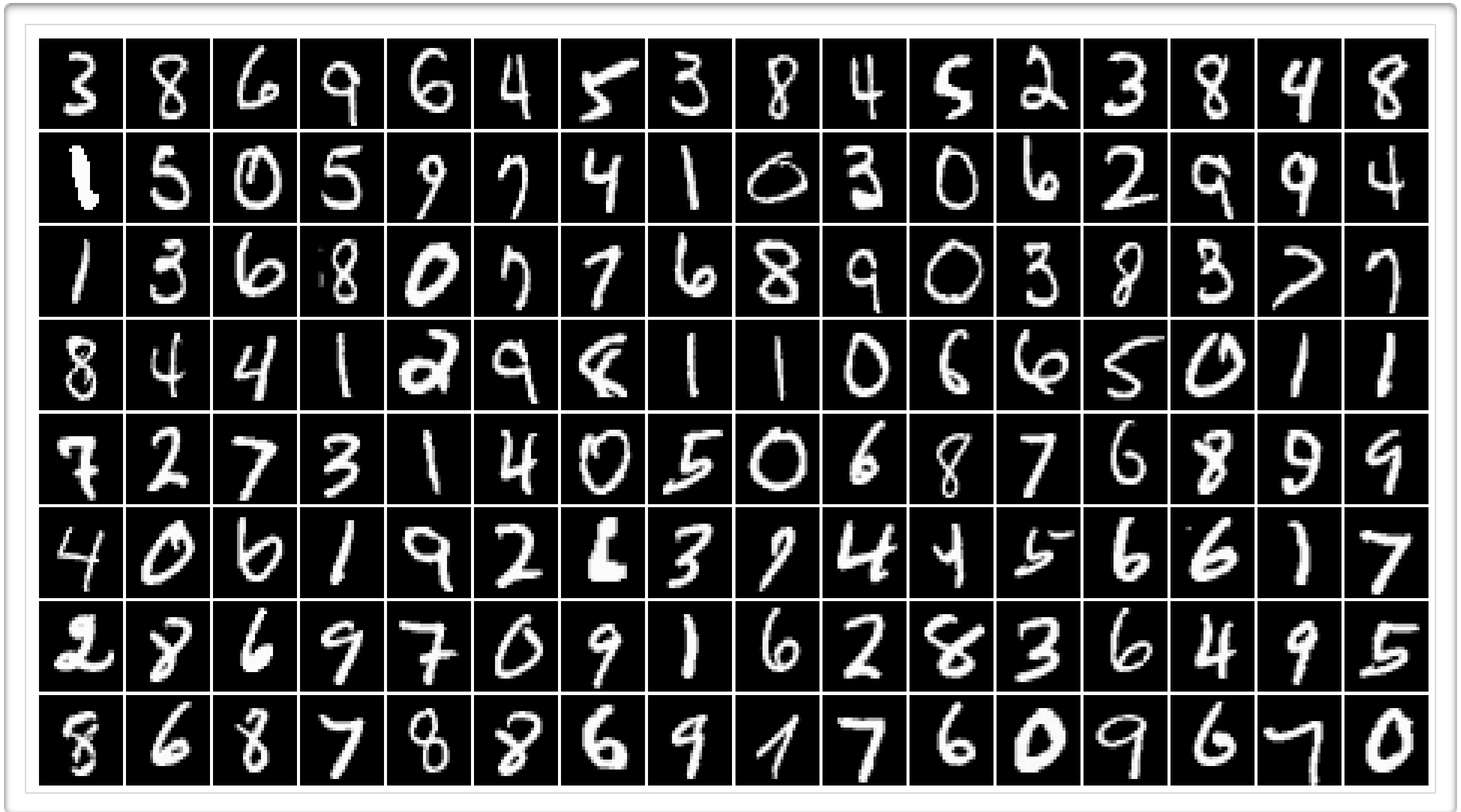
Autoencoder

- Adapting an autoencoder to a new type of input
  - choose a joint distribution $p(\mathbf{x}|\boldsymbol{\mu})$ over the inputs, where $\boldsymbol{\mu}$ is the vector of parameters of that distribution
  - choose the relationship between $\boldsymbol{\mu}$ and the hidden layer $\mathbf{h}(\mathbf{x})$
  - use $l(f(\mathbf{x})) = -\log p(\mathbf{x}|\boldsymbol{\mu})$ as the loss function

- Example: we get the sum of squared distance by
  - choosing a Gaussian distribution with mean $\boldsymbol{\mu}$ and identity covariance for $p(\mathbf{x}|\boldsymbol{\mu}) = \frac{1}{(2\pi)^{D/2}} \exp(-\frac{1}{2} \sum_k (x_k - \mu_k)^2)$
  - And choosing $\boldsymbol{\mu} = \mathbf{c} + \mathbf{W}^* \mathbf{h}(\mathbf{x})$

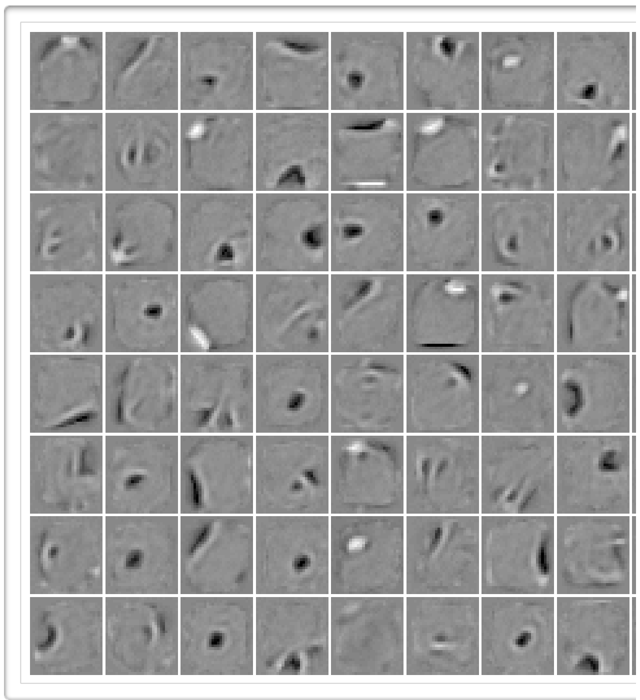
Example: MNIST

- MNIST dataset:

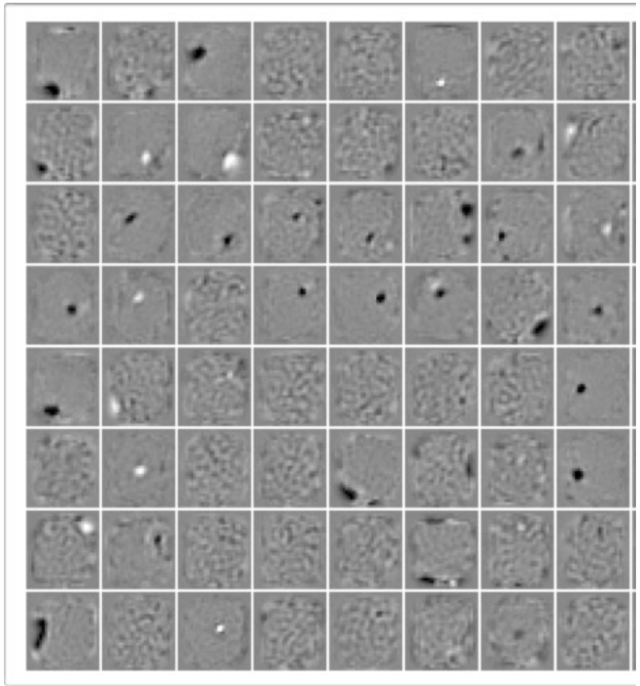


Learned Features

- MNIST dataset:



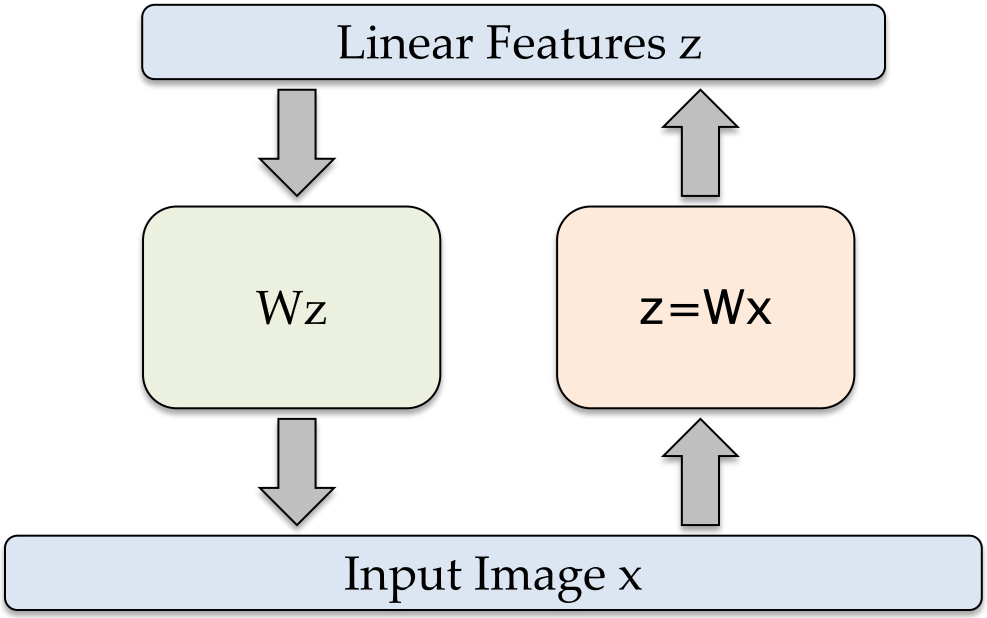
RBM



Autoenncoder

(Larochelle et al., JMLR 2009)

Optimality of the Linear Autoencoder



- If the hidden and output layers are linear, it will learn hidden units that are a linear function of the data and minimize the squared error.

- The K hidden units will span the same space as the first k principal components. The weight vectors may not be orthogonal.

- With nonlinear hidden units, we have a nonlinear generalization of PCA.

Optimality of the Linear Autoencoder

- Let us consider the following theorem:

  - let $\mathbf{A}$ be any matrix, with singular value decomposition $\mathbf{A} = \mathbf{U}\,\Sigma\,\mathbf{V}^\top$
    - Σ is a diagonal matrix
    - $\mathbf{V}$, $\mathbf{U}$ are orthonormal matrices (columns/rows are orthonormal vectors)

Optimality of the Linear Autoencoder

- Let us consider the following theorem:
  - ➢ let $\mathbf{A}$ be any matrix, with singular value decomposition $\mathbf{A} = \mathbf{U}\,\Sigma\,\mathbf{V}^\top$
    - Σ is a diagonal matrix
    - $\mathbf{V}$, $\mathbf{U}$ are orthonormal matrices (columns/rows are orthonormal vectors)
  - ➢ let $\mathbf{U}_{\cdot,\leq k}\,\Sigma_{\leq k,\leq k}\,\mathbf{V}_{\cdot,\leq k}^\top$ be the decomposition where we keep only the k largest singular values
  - ➢ then, the matrix $\mathbf{B}$ of rank k that is closest to $\mathbf{A}$: is

$$\mathbf{B}^* = \underset{\mathbf{B}\ \text{s.t. rank}(\mathbf{B})=k}{\arg\min} ||\mathbf{A} - \mathbf{B}||_F$$

$$\mathbf{B}^* = \mathbf{U}_{\cdot,\leq k}\,\Sigma_{\leq k,\leq k}\,\mathbf{V}_{\cdot,\leq k}^\top$$

$$\min_\theta \sum_t \frac{1}{2}\sum_i (x_i^{(t)} - \underbrace{\widehat{x}_i^{(t)}}_{\text{based on linear encoder}})^2 \quad \geq \quad \min_{\mathbf{W}^*, \mathbf{h}(\mathbf{X})} \frac{1}{2} ||\overbrace{\mathbf{X}}^{\text{matrix where columns are } \mathbf{x}^{(t)}} - \mathbf{W}^* \underbrace{\mathbf{h}(\mathbf{X})}_{\text{matrix of all hidden layers (could be any encoder)}}||_F^2$$

$$\arg\min_{\mathbf{W}^*, \mathbf{h}(\mathbf{X})} \frac{1}{2} ||\mathbf{X} - \mathbf{W}^* \mathbf{h}(\mathbf{X})||_F^2 \quad = \quad (\mathbf{W}^* \leftarrow \mathbf{U}_{\cdot, \leq k}\, \Sigma_{\leq k, \leq k}, \;\; \mathbf{h}(\mathbf{X}) \leftarrow \mathbf{V}_{\cdot, \leq k}^\top)$$

based on previous theorem, where $\mathbf{X} = \mathbf{U}\,\Sigma\,\mathbf{V}^\top$ and k is the hidden layer size

Let's show $\mathbf{h}(\mathbf{X})$ is a linear encoder:

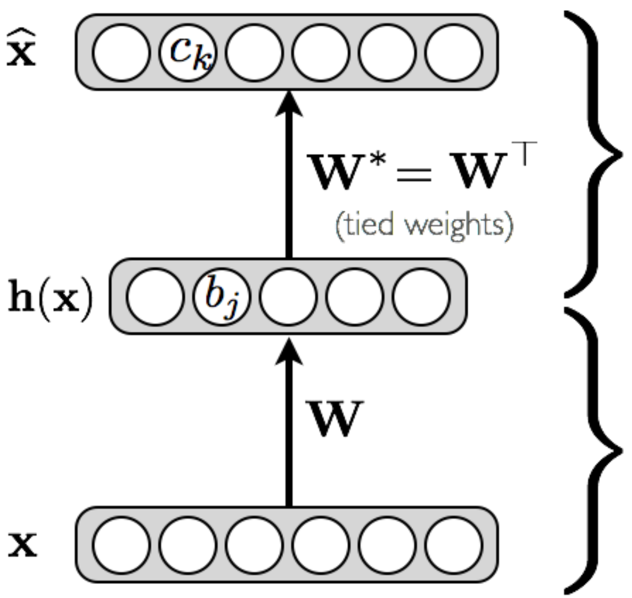
$$\begin{aligned}
\mathbf{h}(\mathbf{X}) &= \mathbf{V}_{\cdot, \leq k}^\top \\
&= \mathbf{V}_{\cdot, \leq k}^\top (\mathbf{X}^\top \mathbf{X})^{-1} (\mathbf{X}^\top \mathbf{X}) && \leftarrow \text{multiplying by identity} \\
&= \mathbf{V}_{\cdot, \leq k}^\top (\mathbf{V}\,\Sigma^\top\,\mathbf{U}^\top\,\mathbf{U}\,\Sigma\,\mathbf{V}^\top)^{-1} (\mathbf{V}\,\Sigma^\top\,\mathbf{U}^\top\,\mathbf{X}) && \leftarrow \text{replace with SVD} \\
&= \mathbf{V}_{\cdot, \leq k}^\top \mathbf{V}\,(\Sigma^\top \Sigma)^{-1}\,\mathbf{V}^\top\,\mathbf{V}\,\Sigma^\top\,\mathbf{U}^\top\,\mathbf{X} && \leftarrow \mathbf{V}(\Sigma^\top\Sigma)^{-1}\mathbf{V}^\top\,\mathbf{V}\Sigma^\top\Sigma\mathbf{V}^\top = \mathbf{I} \\
&= \mathbf{V}_{\cdot, \leq k}^\top \mathbf{V}\,(\Sigma^\top \Sigma)^{-1}\,\Sigma^\top\,\mathbf{U}^\top\,\mathbf{X} && \leftarrow \mathbf{V}^\top\mathbf{V} = \mathbf{I} \text{ (orthonormal)} \\
&= \mathbf{I}_{\leq k, \cdot}\,(\Sigma^\top \Sigma)^{-1}\,\Sigma^\top\,\mathbf{U}^\top\,\mathbf{X} && \leftarrow \text{idem} \\
&= \mathbf{I}_{\leq k, \cdot}\,\Sigma^{-1}\,(\Sigma^\top)^{-1}\,\Sigma^\top\,\mathbf{U}^\top\,\mathbf{X} && \leftarrow (\Sigma^\top\Sigma)^{-1} = \Sigma^{-1}(\Sigma^\top)^{-1} \\
&= \mathbf{I}_{\leq k, \cdot}\,\Sigma^{-1}\,\mathbf{U}^\top\,\mathbf{X} \\
&= \underbrace{\Sigma_{\leq k, \leq k}^{-1}\,(\mathbf{U}_{\cdot, \leq k})^\top}_{\text{this is a linear encoder}}\,\mathbf{X} && \leftarrow \text{multiplying by } \mathbf{I}_{\leq k, \cdot} \text{ selects the } k \text{ first rows}
\end{aligned}$$

(Slide from Hugo Larochelle)

Optimality of the Linear Autoencoder

- So an optimal pair of encoder and decoder is

$$\mathbf{h}(\mathbf{x}) = \underbrace{\left(\Sigma^{-1}_{\leq k, \leq k} \left(\mathbf{U}_{\cdot, \leq k}\right)^{\top}\right)}_{\mathbf{W}} \mathbf{x} \qquad \widehat{\mathbf{x}} = \underbrace{\left(\mathbf{U}_{\cdot, \leq k} \, \Sigma_{\leq k, \leq k}\right)}_{\mathbf{W}^{*}} \mathbf{h}(\mathbf{x})$$

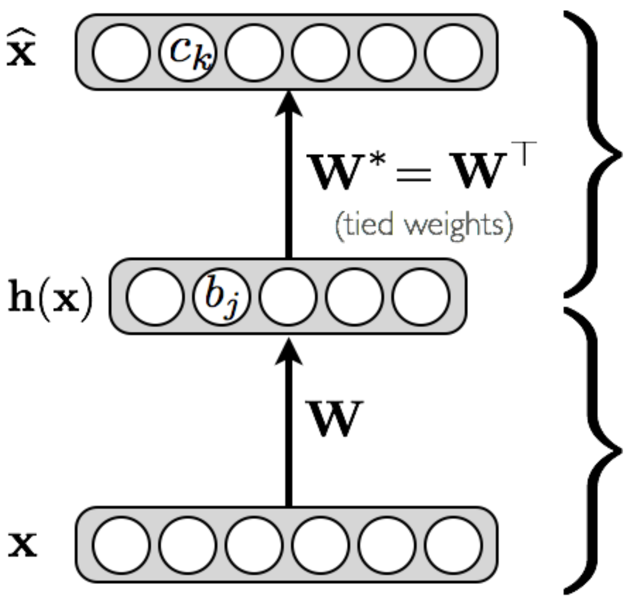


- for the sum of squared difference error
- for an autoencoder with a linear decoder
- where optimality means "has the lowest training reconstruction error"

Optimality of the Linear Autoencoder

- So an optimal pair of encoder and decoder is

$$\mathbf{h}(\mathbf{x}) = \underbrace{\left(\Sigma_{\leq k, \leq k}^{-1} \left(\mathbf{U}_{\cdot, \leq k}\right)^{\top}\right)}_{\mathbf{W}} \mathbf{x} \qquad \widehat{\mathbf{x}} = \underbrace{\left(\mathbf{U}_{\cdot, \leq k}\, \Sigma_{\leq k, \leq k}\right)}_{\mathbf{W}^{*}} \mathbf{h}(\mathbf{x})$$



- If inputs are normalized as follows:

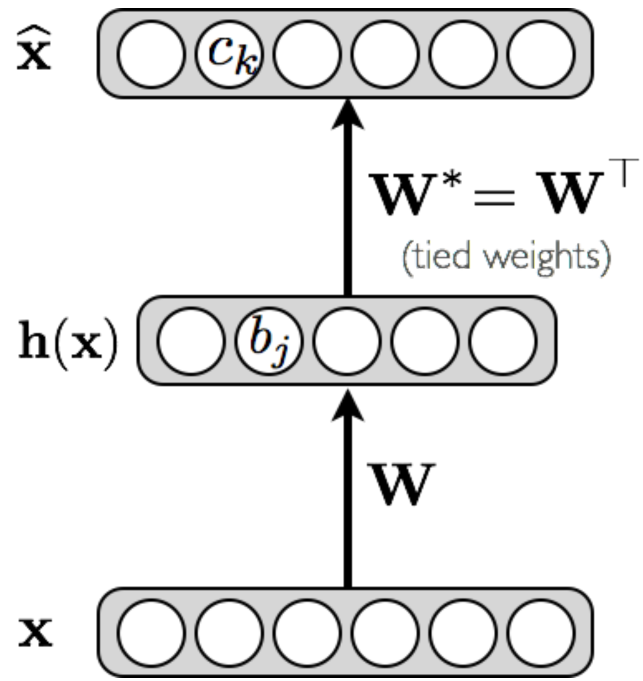
$$\mathbf{x}^{(t)} \leftarrow \frac{1}{\sqrt{T}} \left(\mathbf{x}^{(t)} - \frac{1}{T} \sum_{t'=1}^{T} \mathbf{x}^{(t')}\right)$$

  - encoder corresponds to Principal Component Analysis (PCA)
  - singular values and (left) vectors = the eigenvalues/vectors of covariance matrix

Undercomplete Representation

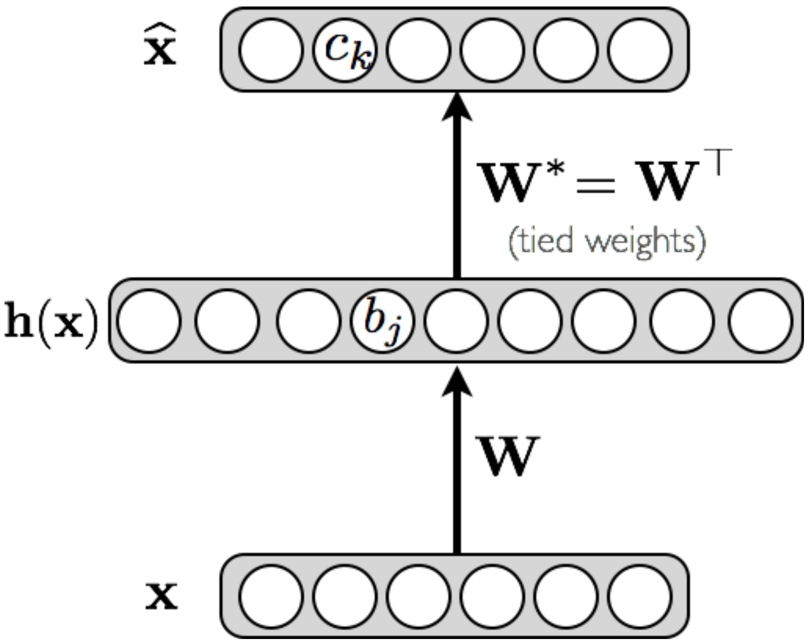
- Hidden layer is undercomplete if smaller than the input layer (bottleneck layer, e.g. dimensionality reduction):
  - hidden layer "compresses" the input
  - will compress well only for the training distribution
- Hidden units will be
  - good features for the traini

    distribution
  - will not be robust to other types of input



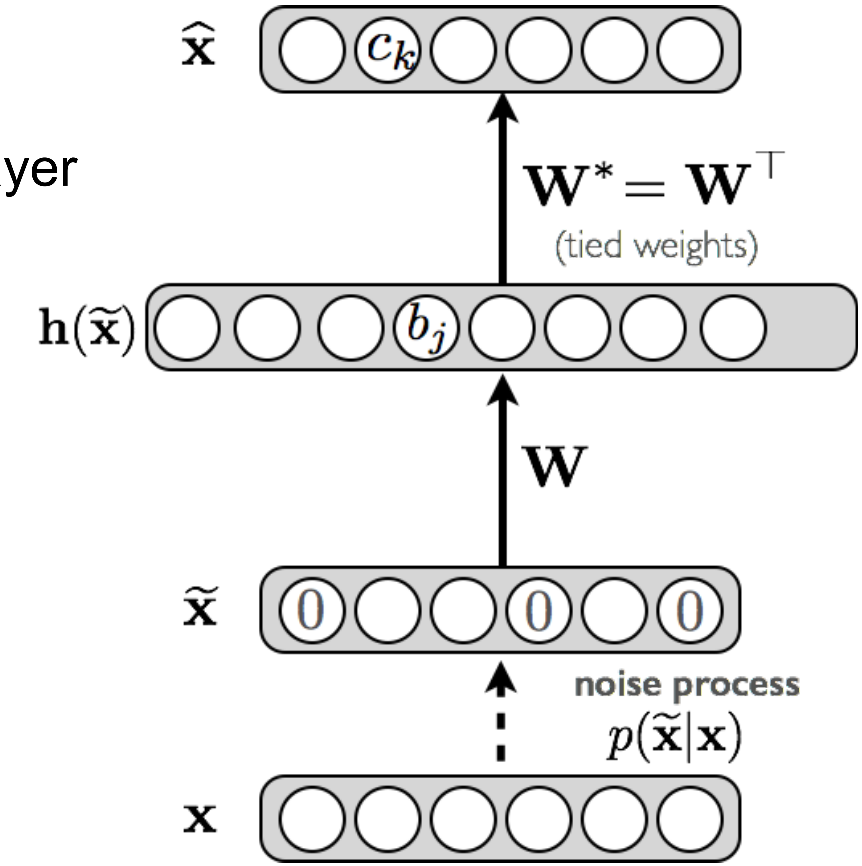
Overcomplete Representation

- Hidden layer is overcomplete if greater than the input layer
  - ➢ no compression in hidden layer
  - ➢ each hidden unit could copy a different input component
- No guarantee that the hidden units will extract meaningful structure



Denoising Autoencoder

- Idea: representation should be robust to introduction of noise:
  - ➢ random assignment of subset of inputs to 0, with probability ν
  - ➢ Similar to dropouts on the input layer
  - ➢ Gaussian additive noise



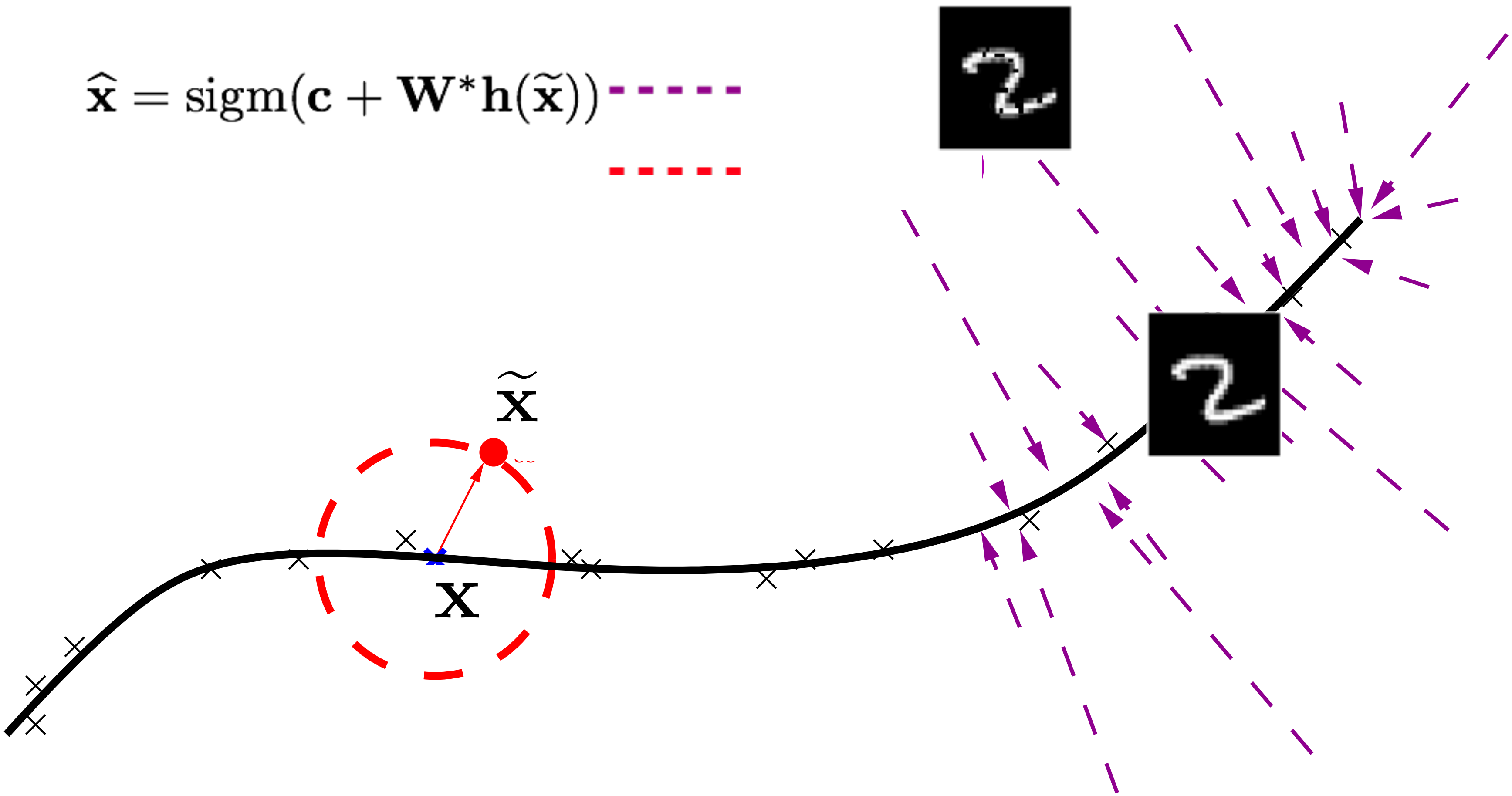
- Reconstruction $\widehat{\mathbf{x}}$ computed from the corrupted input $\widetilde{\mathbf{x}}$

- Loss function compares $\widehat{\mathbf{x}}$ reconstruction with the noiseless input $\mathbf{x}$

(Vincent et.al., ICML 2008)

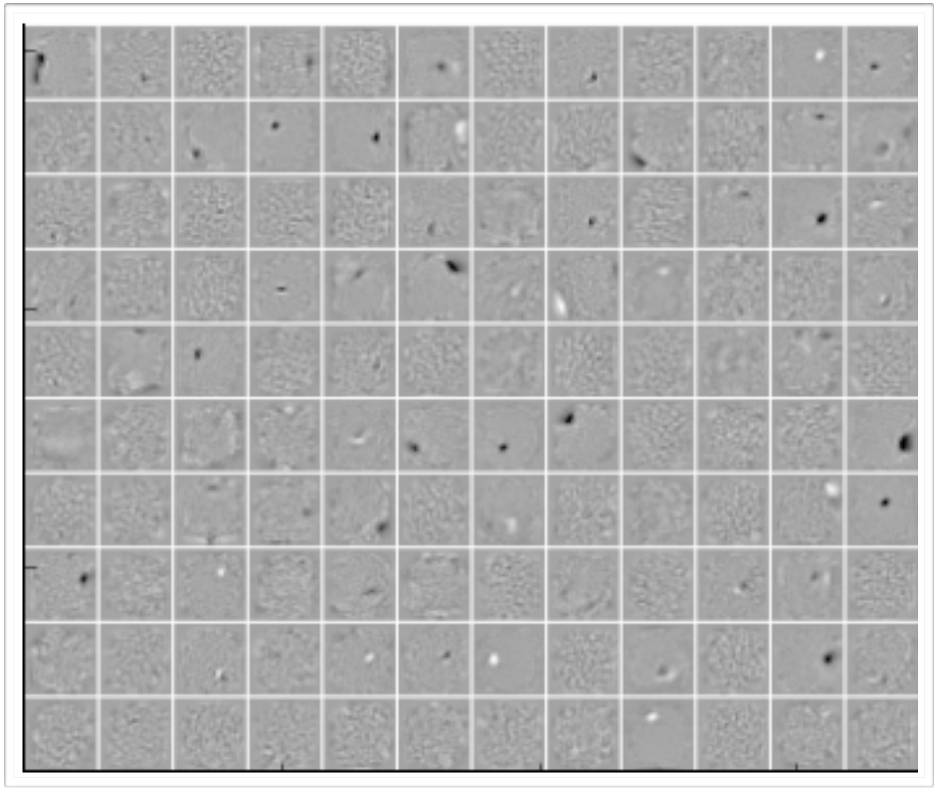
Denoising Autoencoder

$$\widehat{\mathbf{x}} = \mathrm{sigm}(\mathbf{c} + \mathbf{W}^{*}\mathbf{h}(\widetilde{\mathbf{x}}))$$

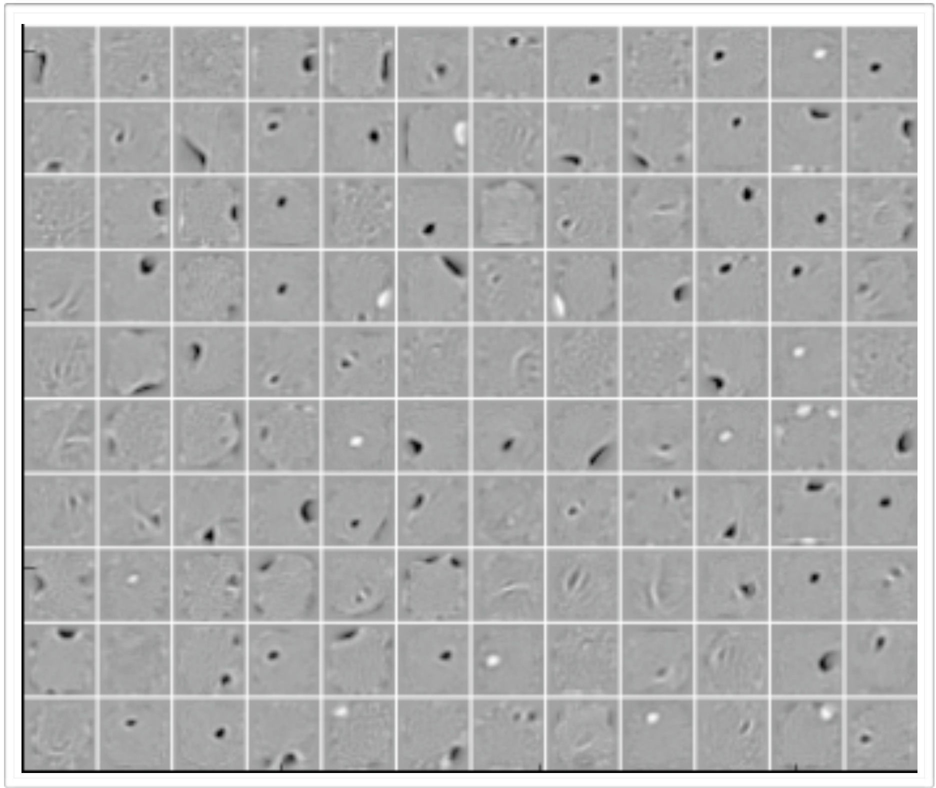


Learned Filters

Non-corrupted



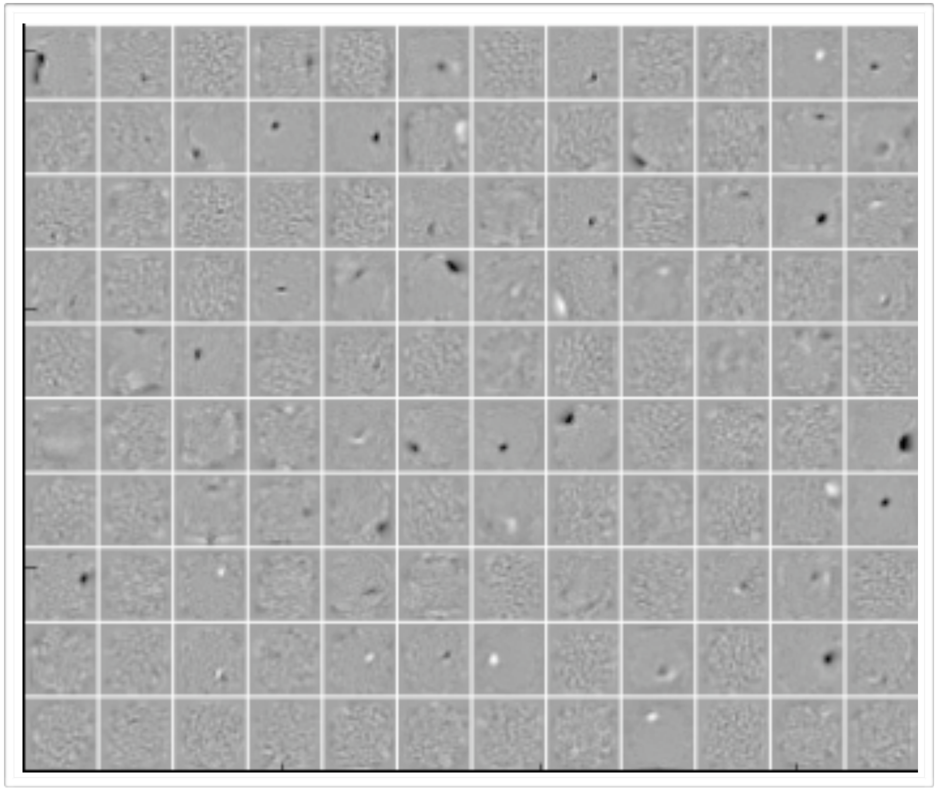
25% corrupted input



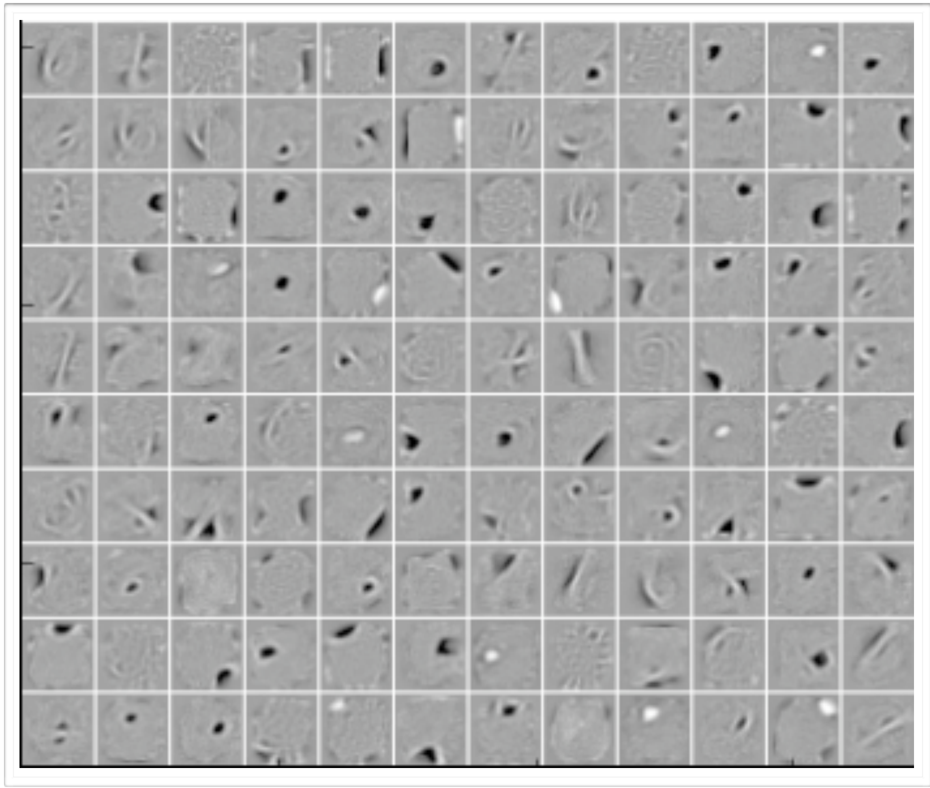
(Vincent et.al., ICML 2008)

Learned Filters

Non-corrupted



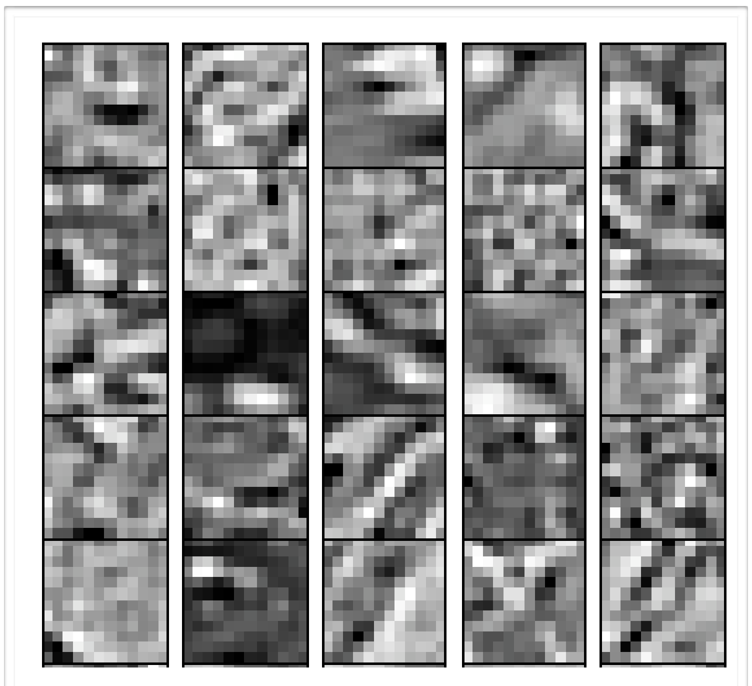
50% corrupted input



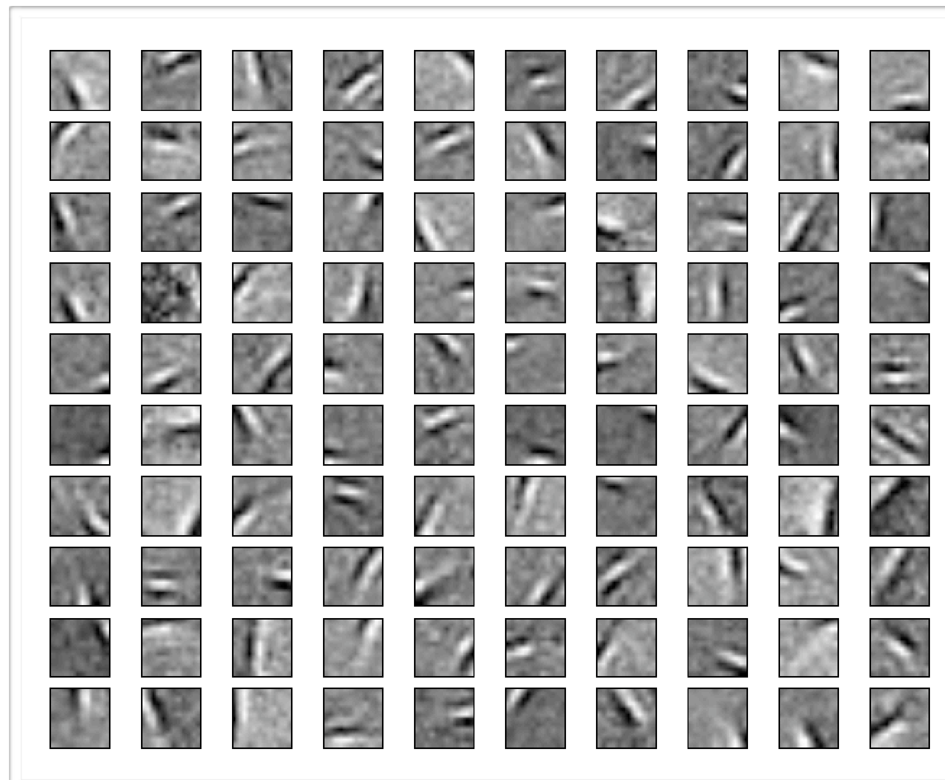
(Vincent et.al., ICML 2008)

Squared Error Loss

- Training on natural image patches, with squared loss
  - PCA may not the best solution



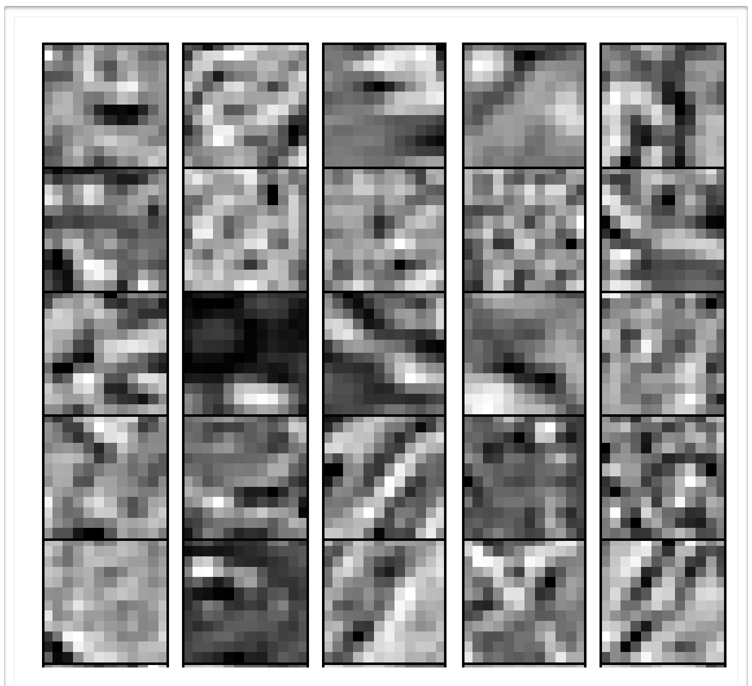
Data



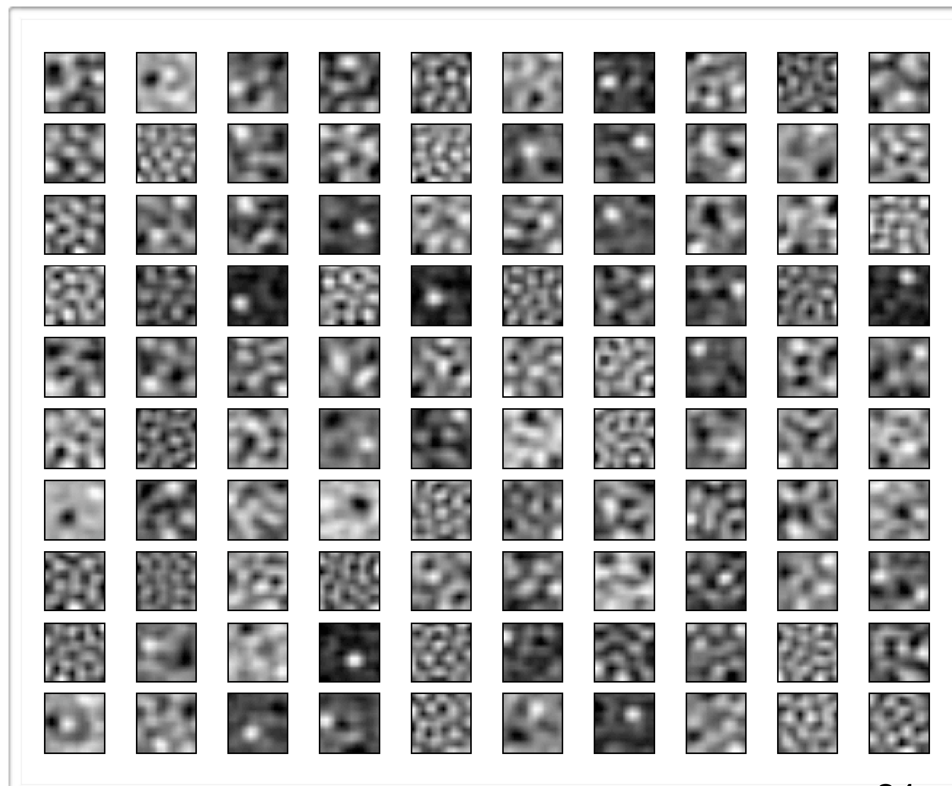
Filters

Squared Error Loss

- Training on natural image patches, with squared loss
  - ➢ PCA may not the best solution



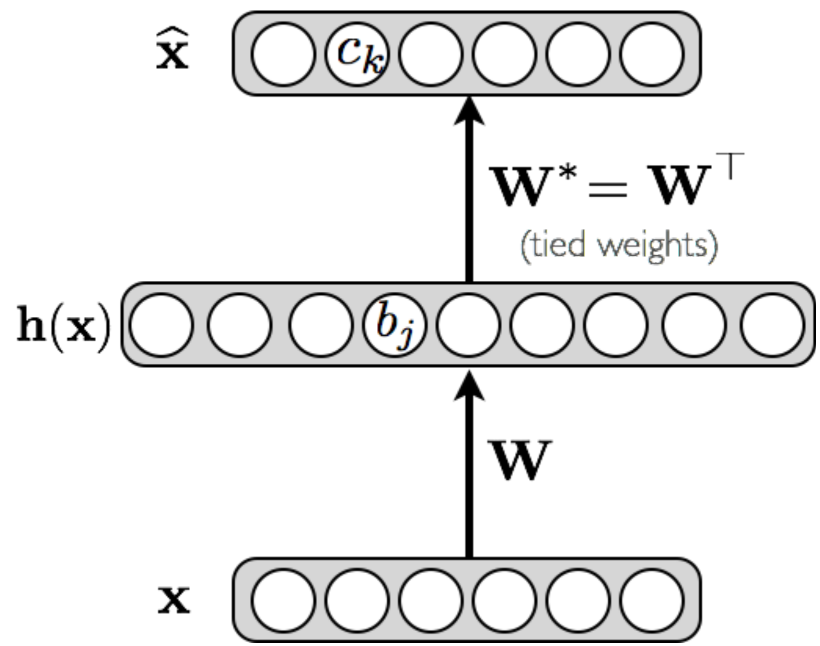
Data



Filters

Contractive Autoencoders

- Alternative approach to avoid uninteresting solutions
  - add an explicit term in the loss that penalizes that solution



- We wish to extract features that only reflect variations observed in the training set
  - we'd like to be invariant to the other variations

(Salah Rifaiet et.al., 2011)

Contractive Autoencoders

- Consider the following loss function:

$$l(f(\mathbf{x}^{(t)})) + \lambda ||\nabla_{\mathbf{x}^{(t)}} \mathbf{h}(\mathbf{x}^{(t)})||_F^2$$

Reconstruction Loss

Jacobian of Encoder

- For the binary observations:

$$l(f(\mathbf{x}^{(t)})) = -\sum_k \left(x_k^{(t)} \log(\widehat{x}_k^{(t)}) + (1 - x_k^{(t)}) \log(1 - \widehat{x}_k^{(t)}) \right)$$

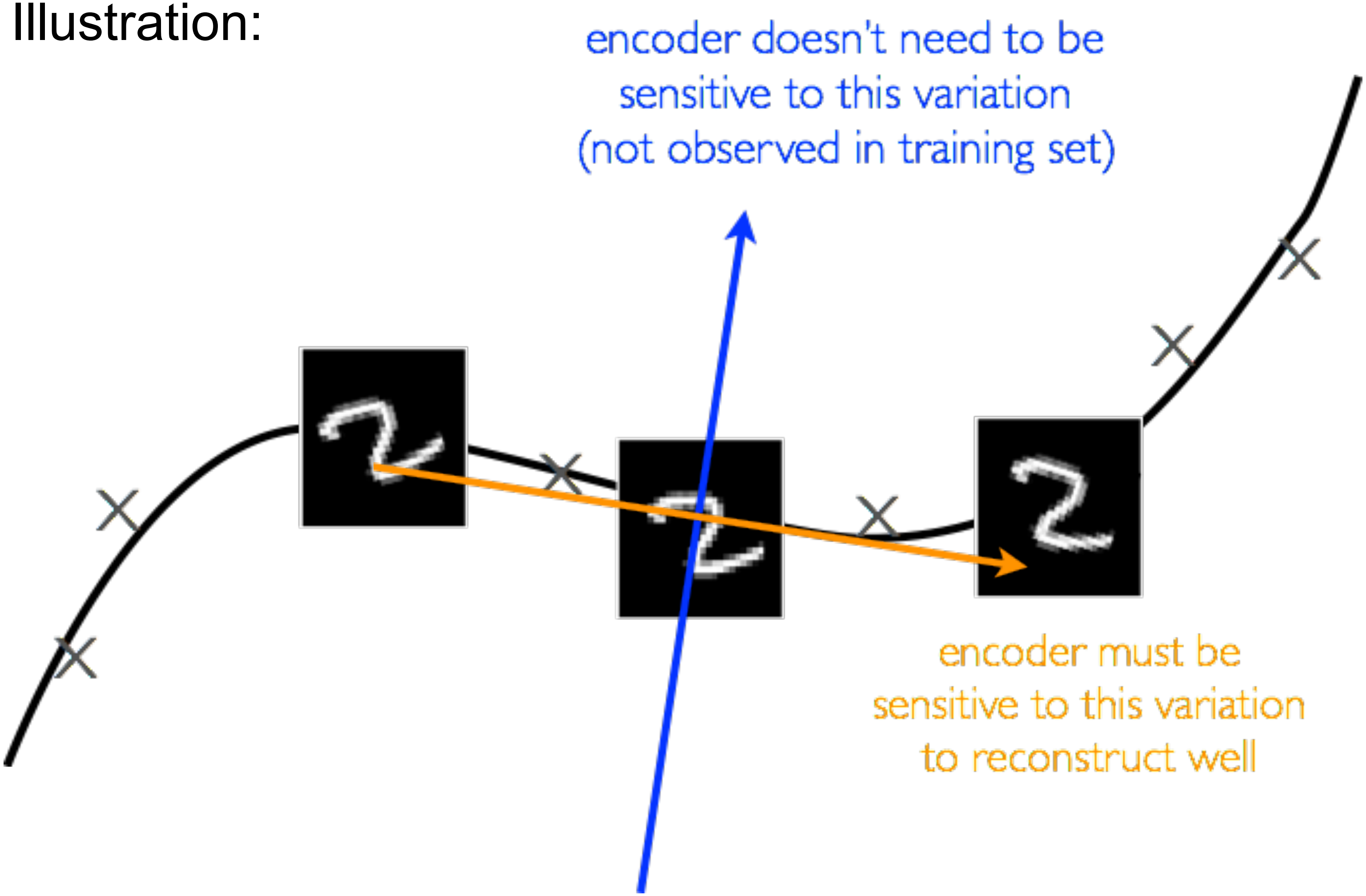
Autoencoder attempts to preserve all information

$$||\nabla_{\mathbf{x}^{(t)}} \mathbf{h}(\mathbf{x}^{(t)})||_F^2 = \sum_j \sum_k \left(\frac{\partial h(\mathbf{x}^{(t)})_j}{\partial x_k^{(t)}} \right)^2$$

Encoder throws away all information

Contractive Autoencoders

- Illustration:

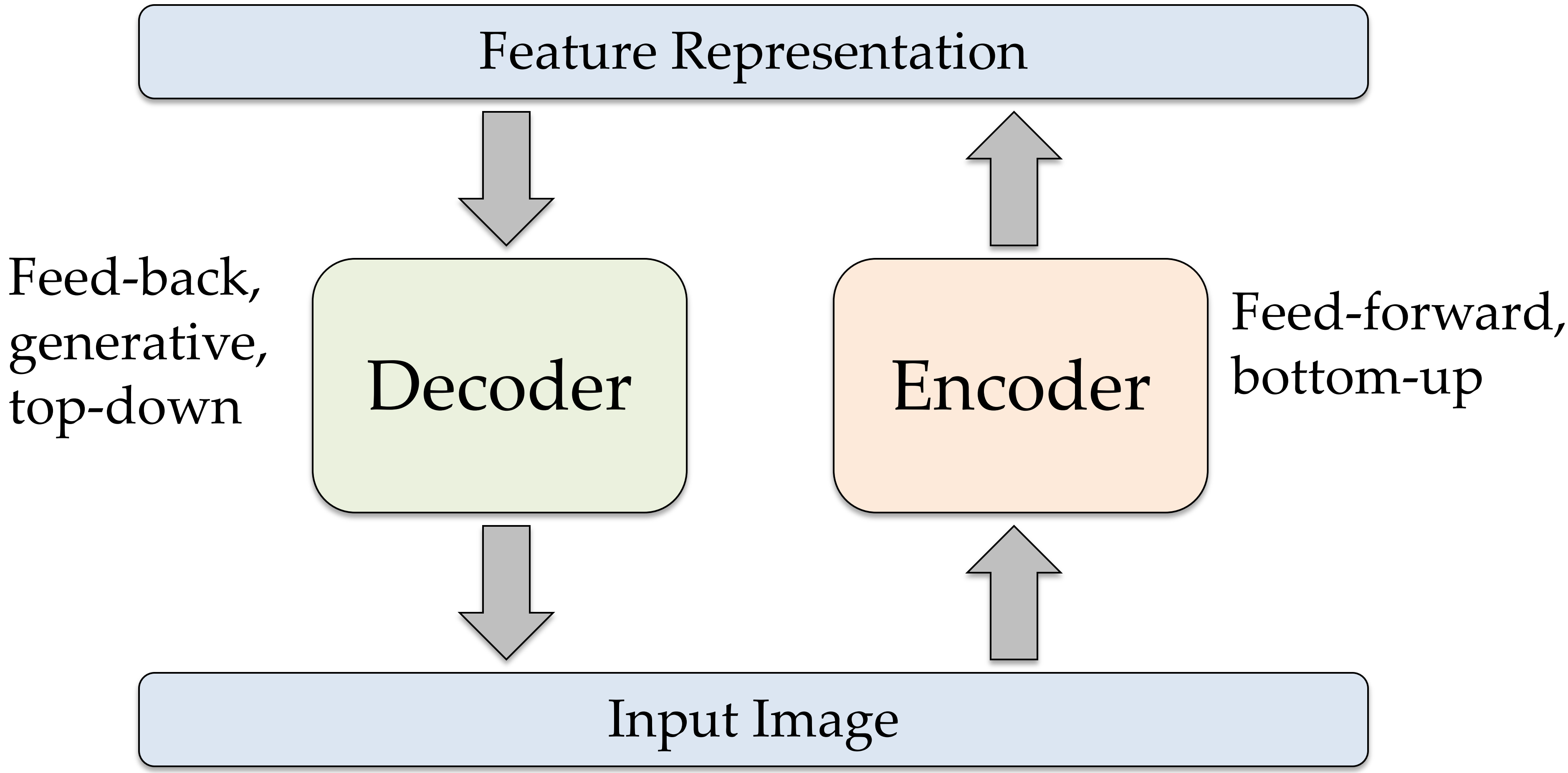


Pros and Cons

- Advantage of denoising autoencoder: simpler to implement
  - ➢ requires adding one or two lines of code to regular autoencoder
  - ➢ no need to compute Jacobian of hidden layer

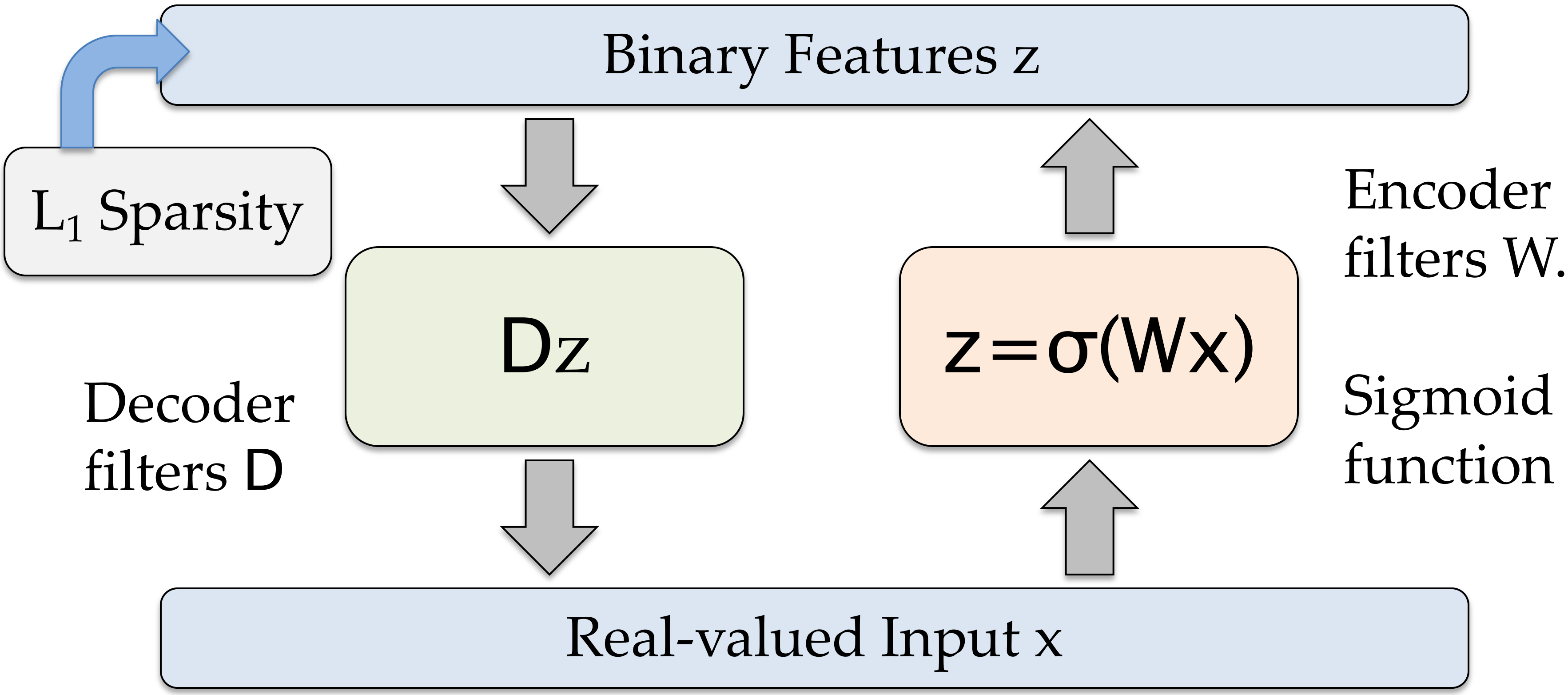
- Advantage of contractive autoencoder: gradient is deterministic
  - ➢ can use second order optimizers (conjugate gradient, LBFGS, etc.)
  - ➢ might be more stable than denoising autoencoder, which uses a sampled gradient

Autoencoder



- Details of what goes insider the encoder and decoder matter!
- Need constraints to avoid learning an identity.

Predictive Sparse Decomposition

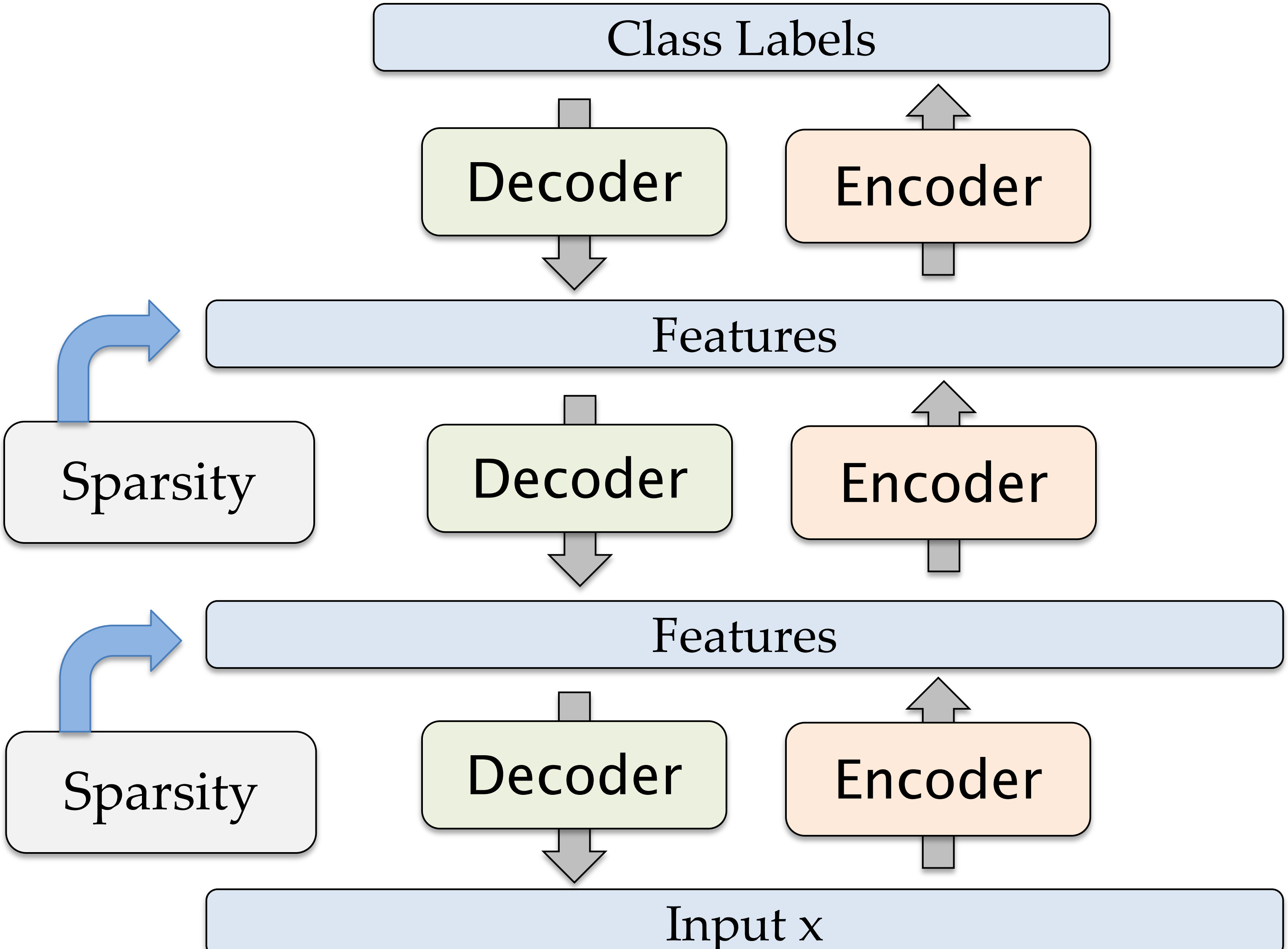


At training time

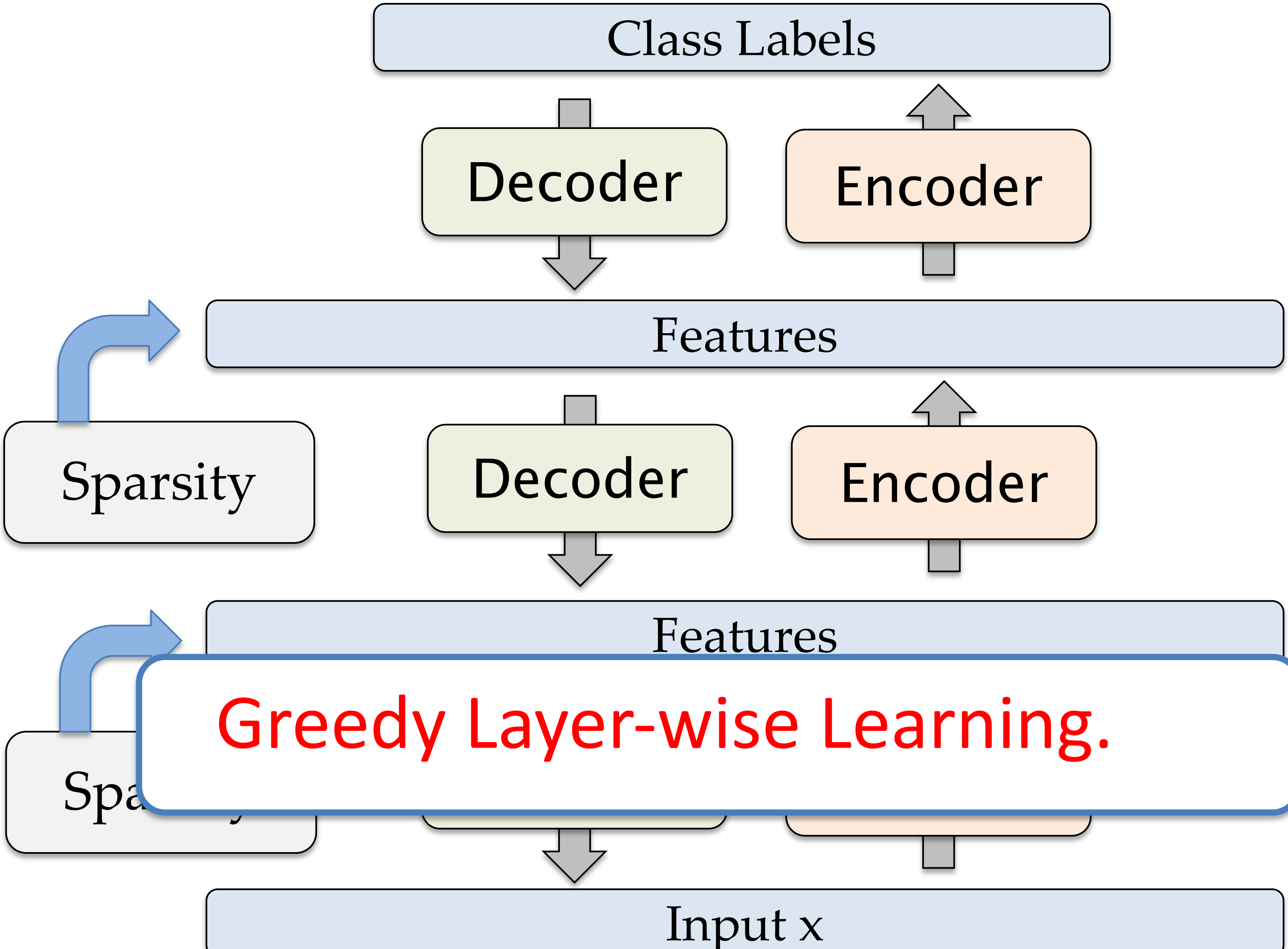
$$\min_{D,W,\mathbf{z}} \underbrace{||D\mathbf{z}-\mathbf{x}||_2^2+\lambda|\mathbf{z}|_1}_{\text{Decoder}} + \underbrace{||\sigma(W\mathbf{x})-\mathbf{z}||_2^2}_{\text{Encoder}}$$

Kavukcuoglu et al., '09

Stacked Autoencoders



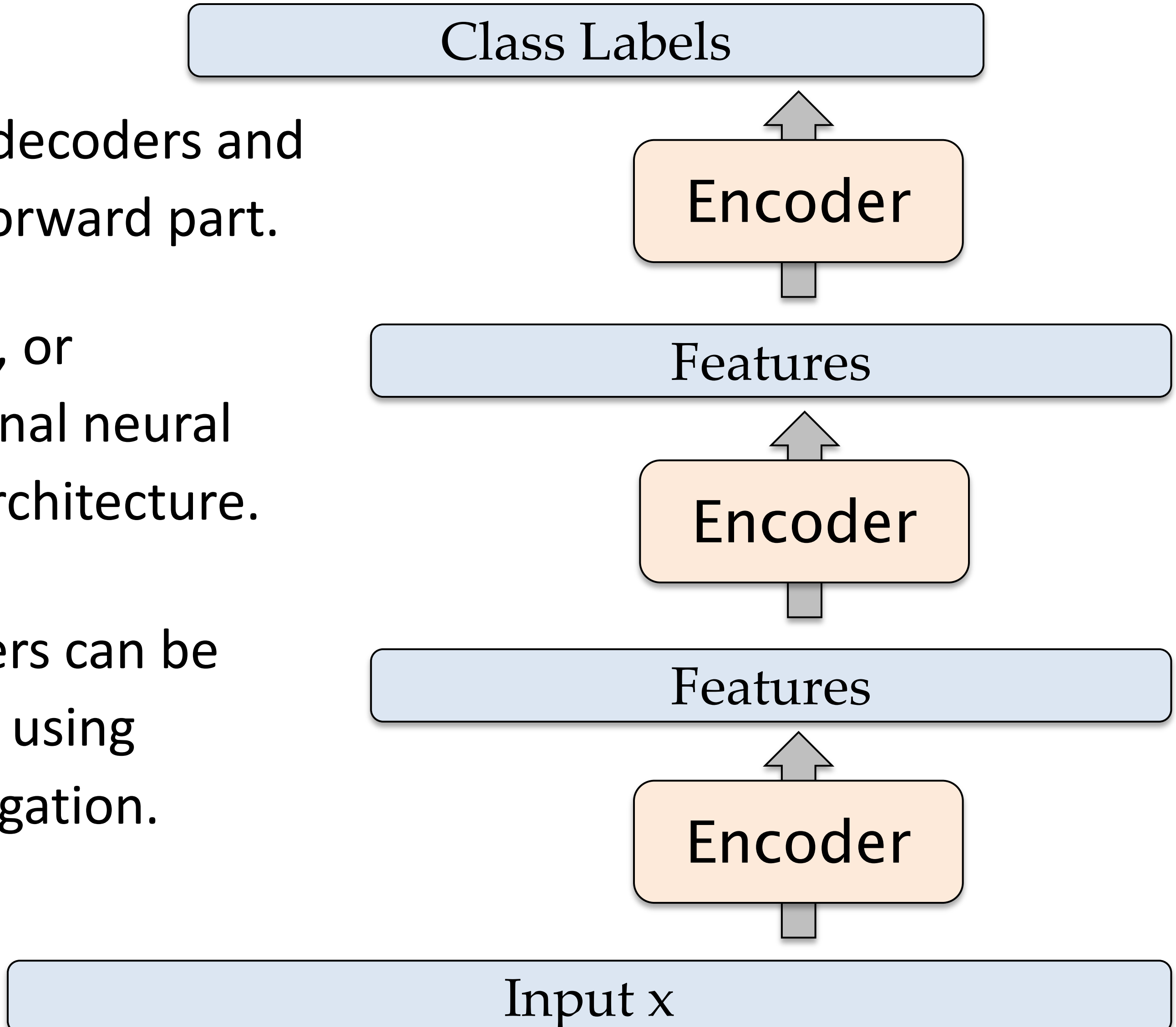
Stacked Autoencoders



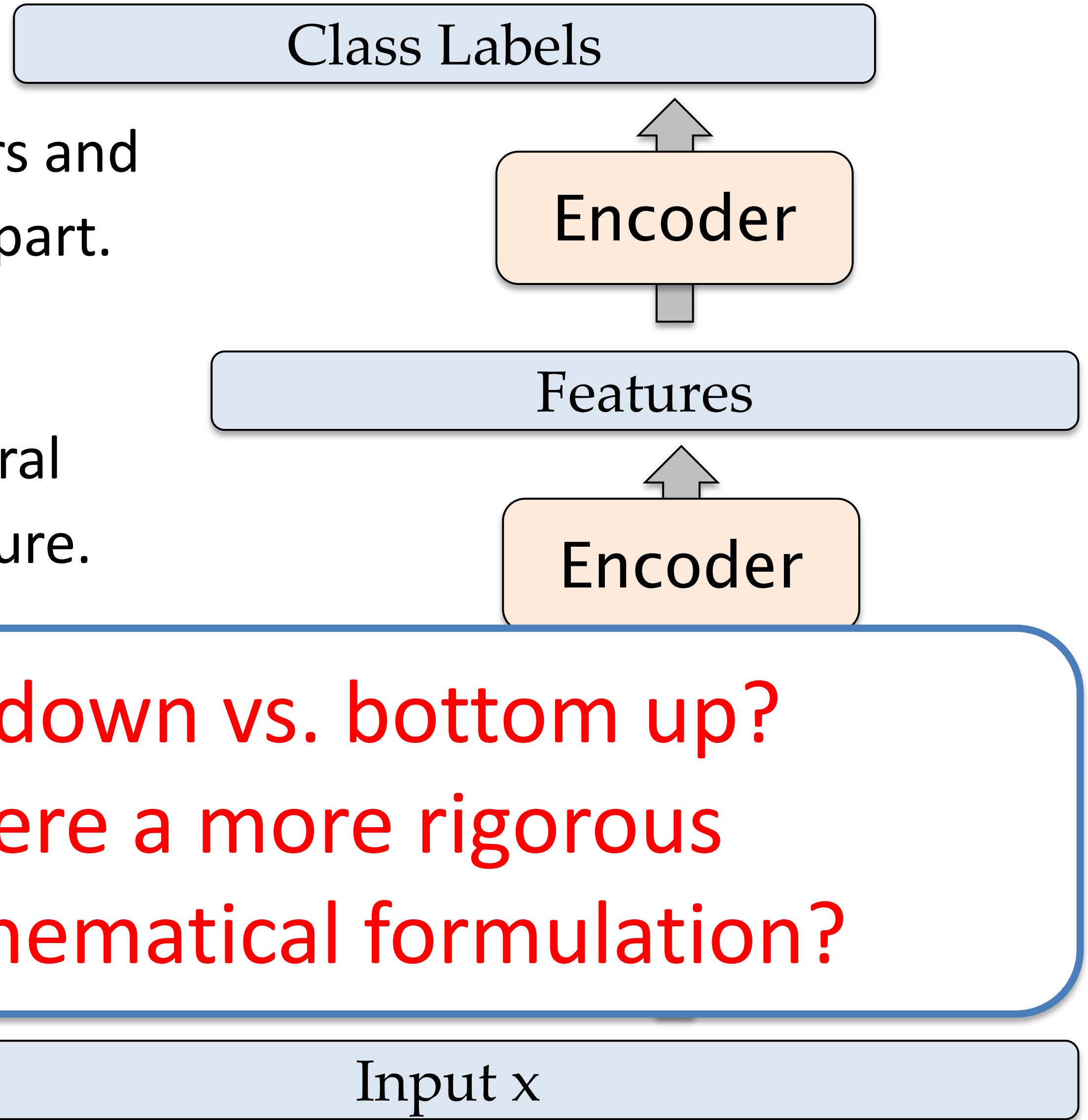
Greedy Layer-wise Learning.

Stacked Autoencoders

- Remove decoders and use feed-forward part.
- Standard, or convolutional neural network architecture.
- Parameters can be fine-tuned using backpropagation.



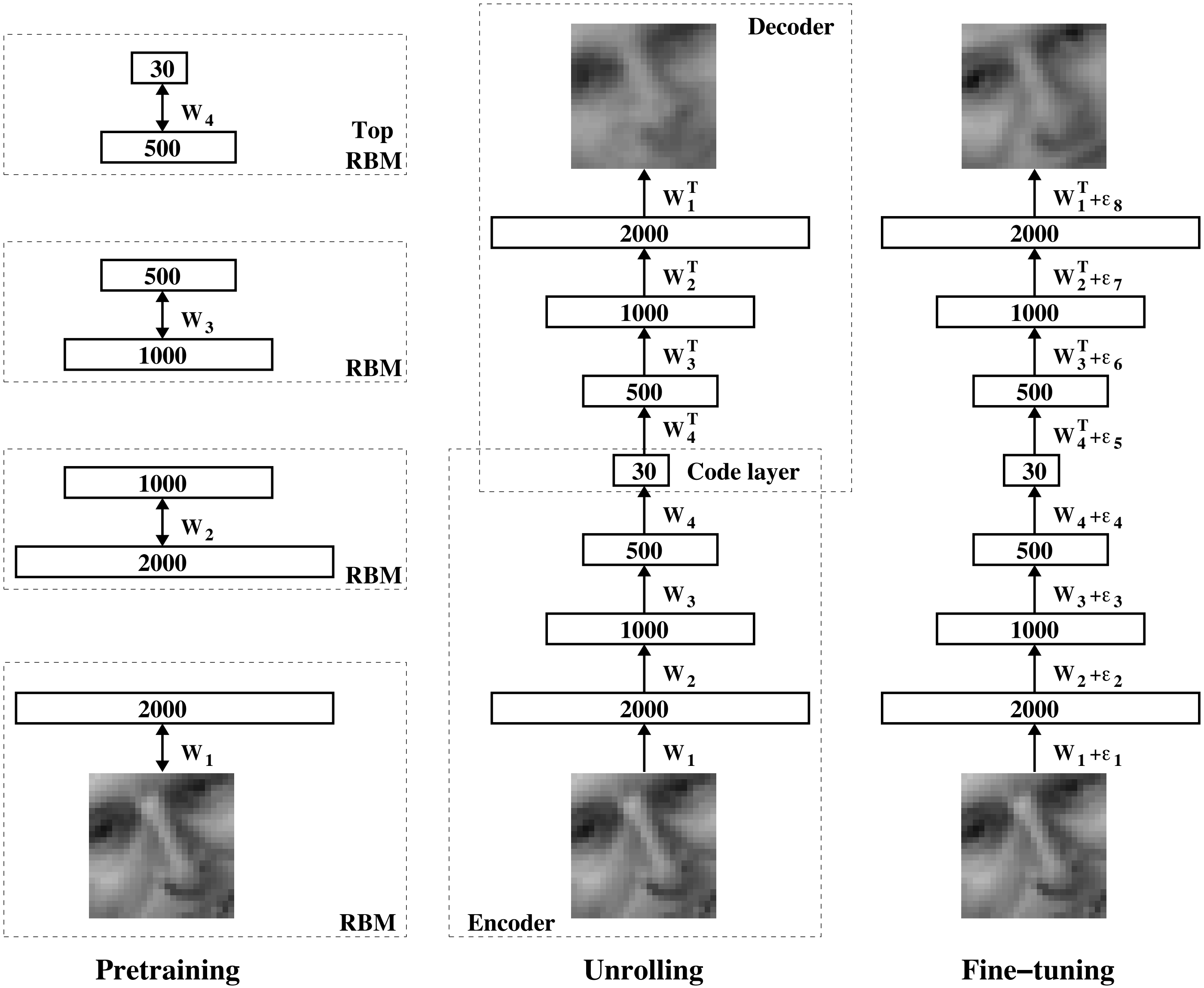
Stacked Autoencoders



- Remove decoders and use feed-forward part.
- Standard, or convolutional neural network architecture.
- Param
fine-tu
backpr

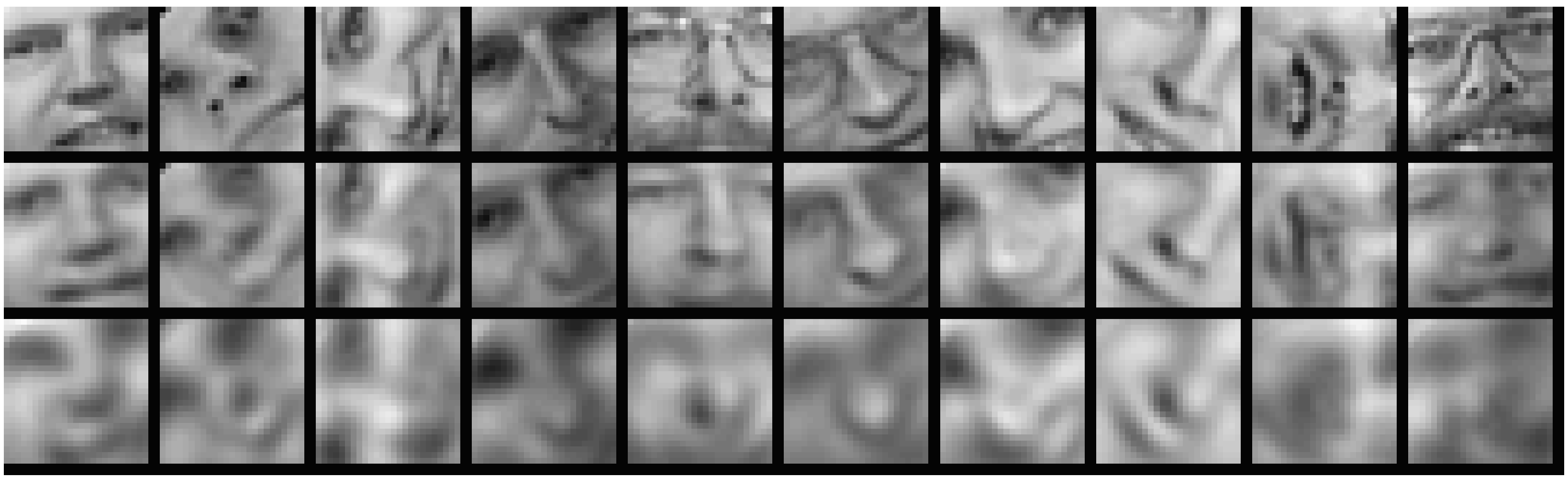
Top-down vs. bottom up?
Is there a more rigorous mathematical formulation?

Deep Autoencoders



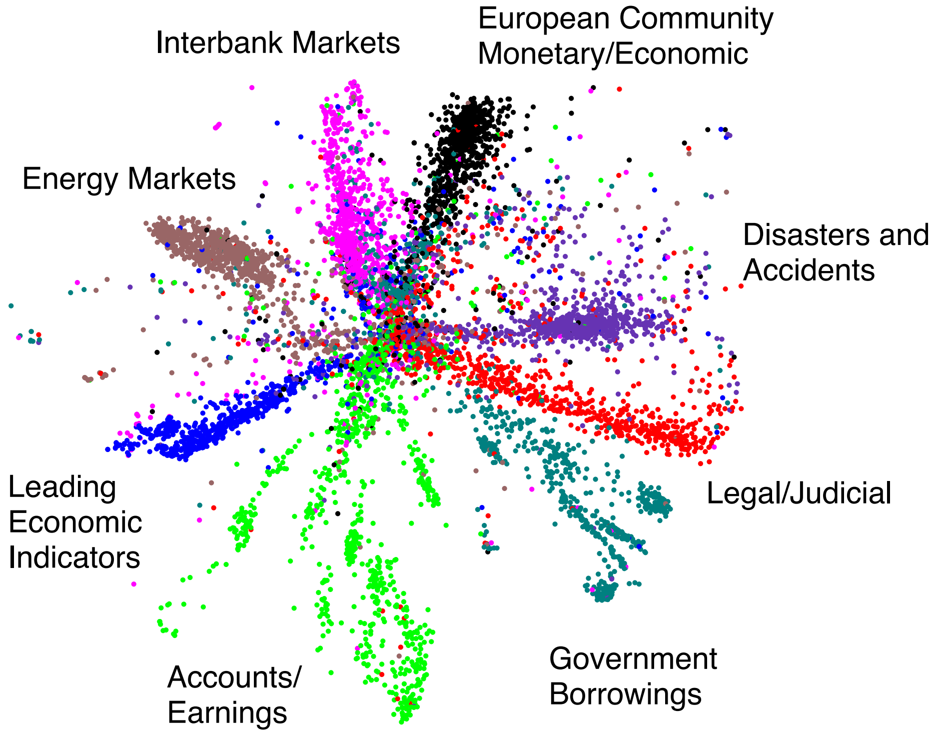
Deep Autoencoders

- We used 25x25 – 2000 – 1000 – 500 – 30 autoencoder to extract 30-D real-valued codes for Olivetti face patches.

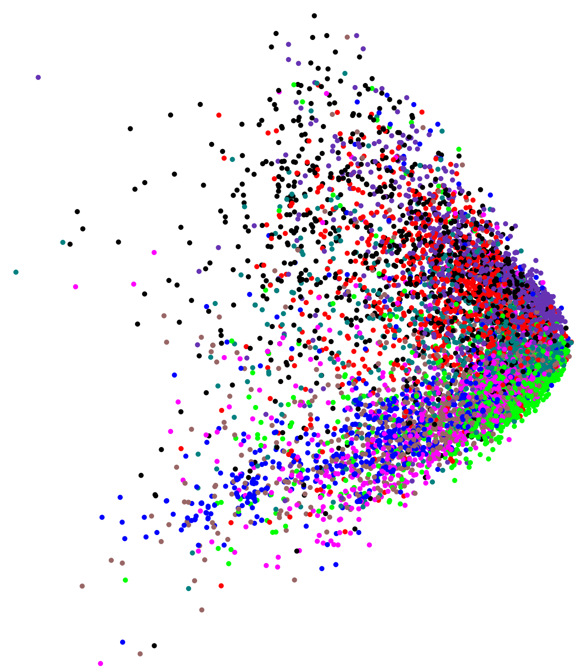


- **Top**: Random samples from the test dataset.
- **Middle**: Reconstructions by the 30-dimensional deep autoencoder.
- **Bottom**: Reconstructions by the 30-dimentinoal PCA.

Information Retrieval



2-D LSA space

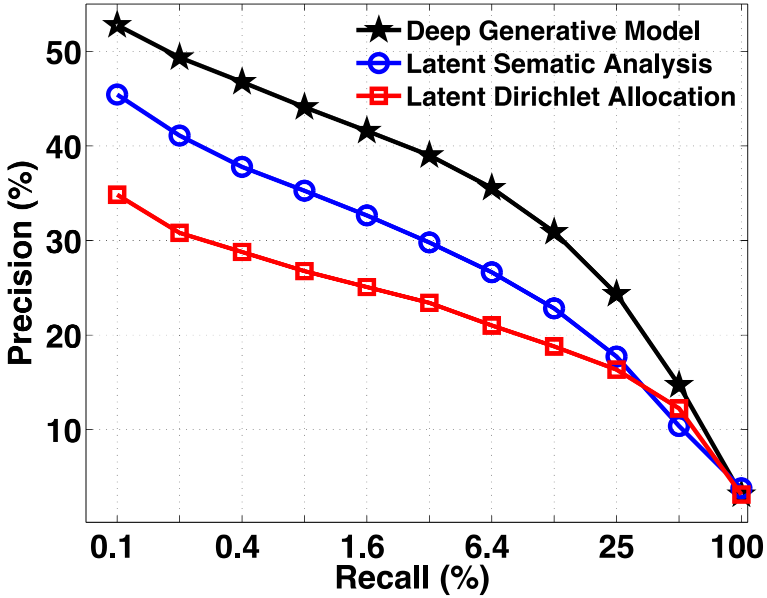


- The Reuters Corpus Volume II contains 804,414 newswire stories (randomly split into **402,207 training** and **402,207 test).**

- "Bag-of-words" representation: each article is represented as a vector containing the counts of the most frequently used 2000 words in the training set.

Information Retrieval

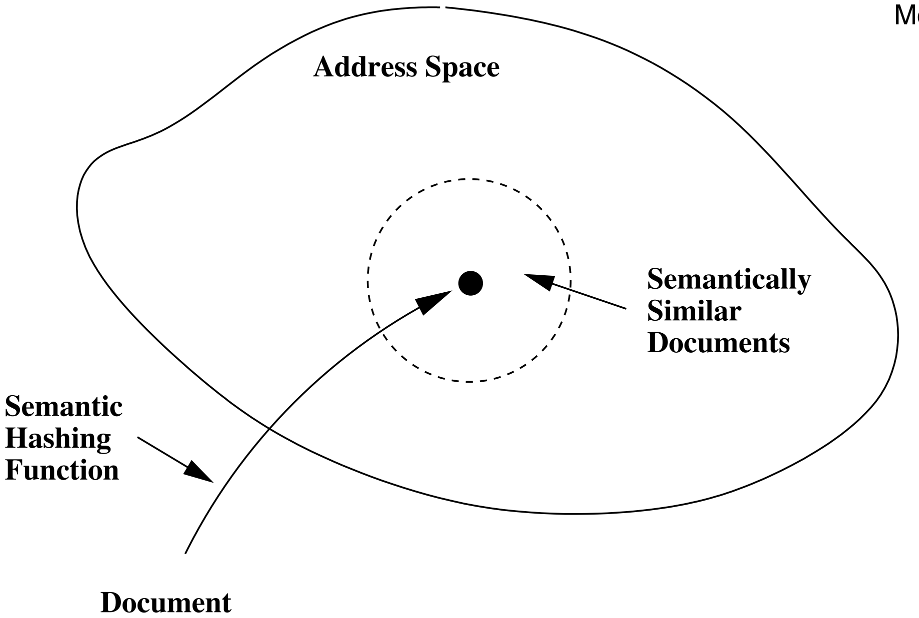
Reuters Dataset

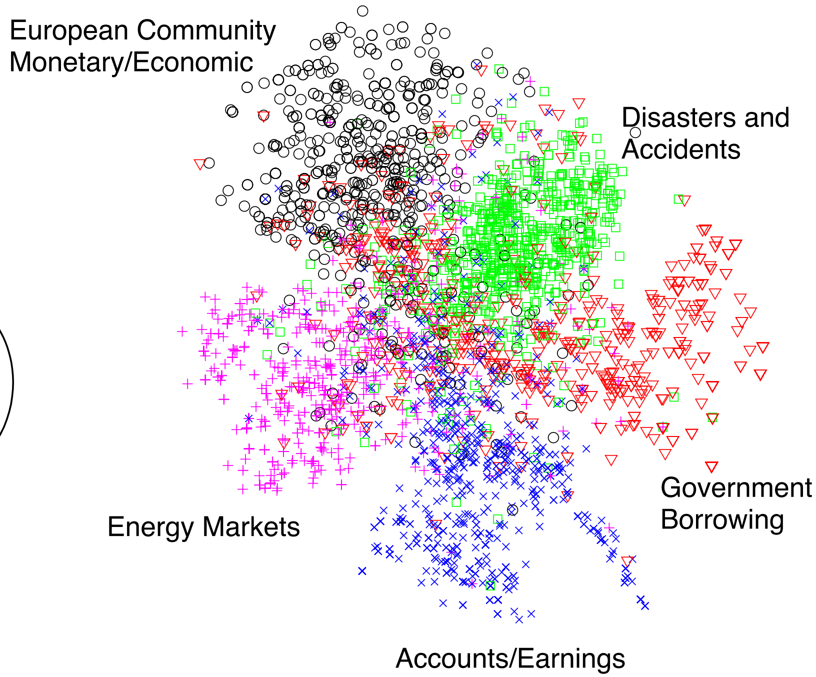


Reuters dataset: 804,414 newswire stories.

Deep generative model significantly outperforms LSA and LDA topic models

Semantic Hashing



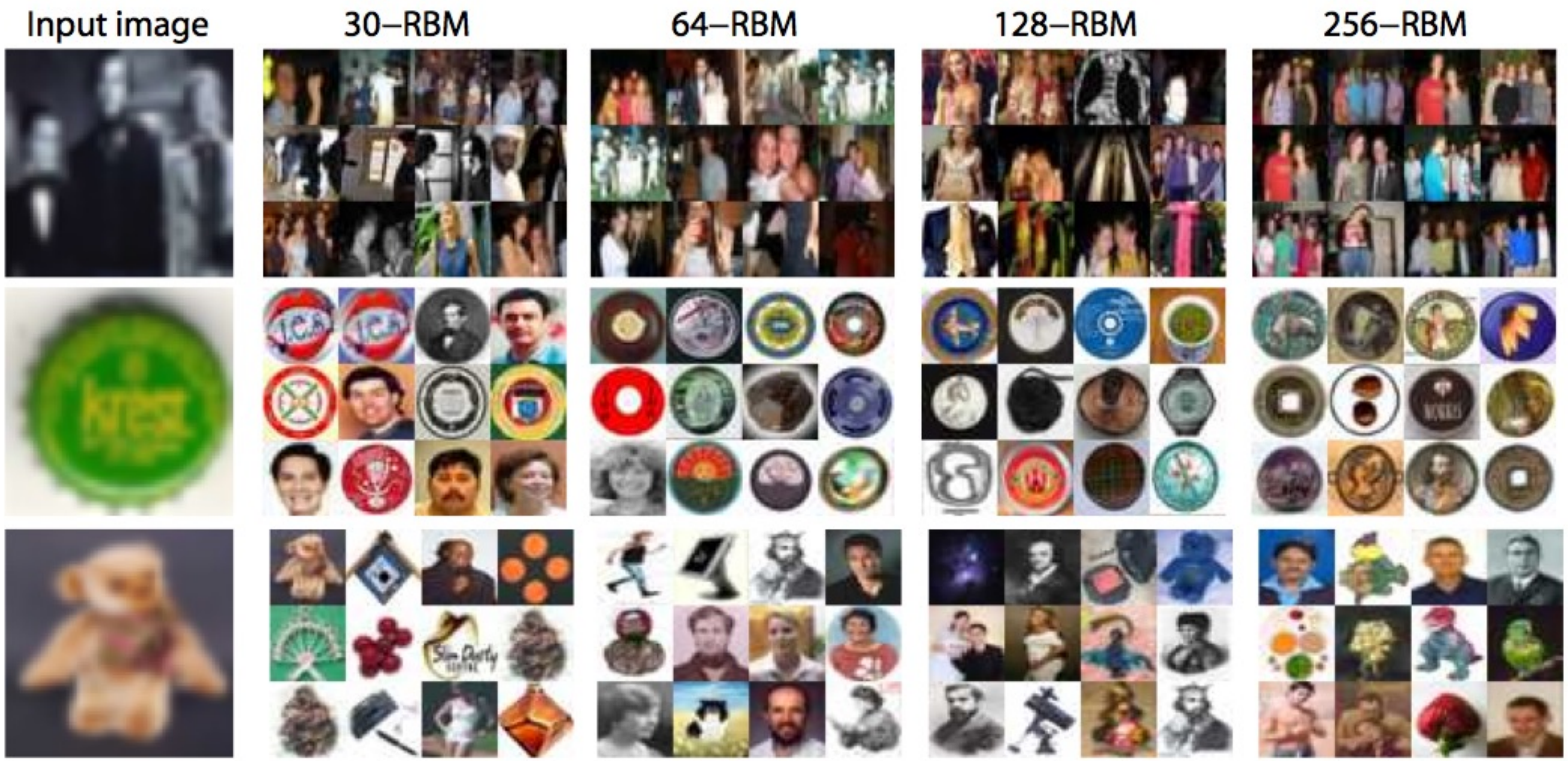


- Learn to map documents into **semantic 20-D binary codes.**
- Retrieve similar documents stored at the nearby addresses **with no search at all.**

(Salakhutdinov and Hinton, SIGIR 2007)

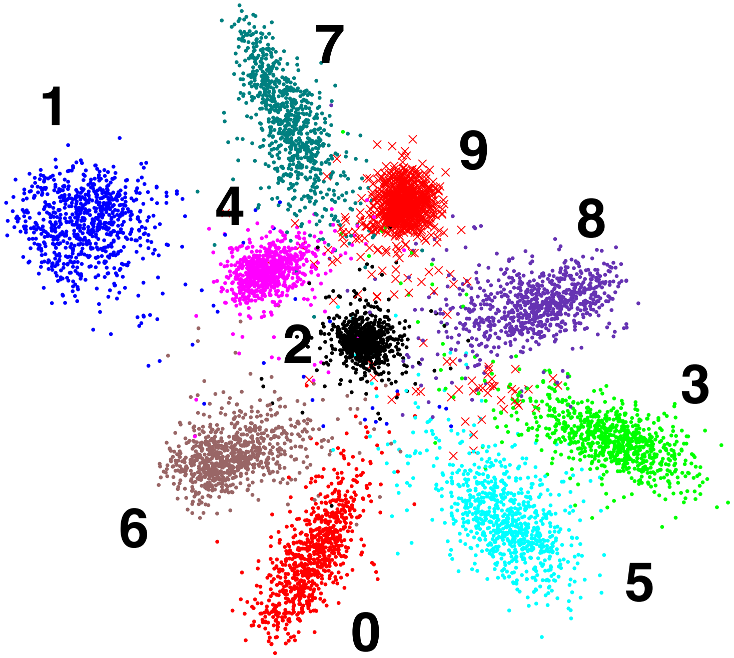
Searching Large Image Database using Binary Codes

- Map images into binary codes for fast retrieval.



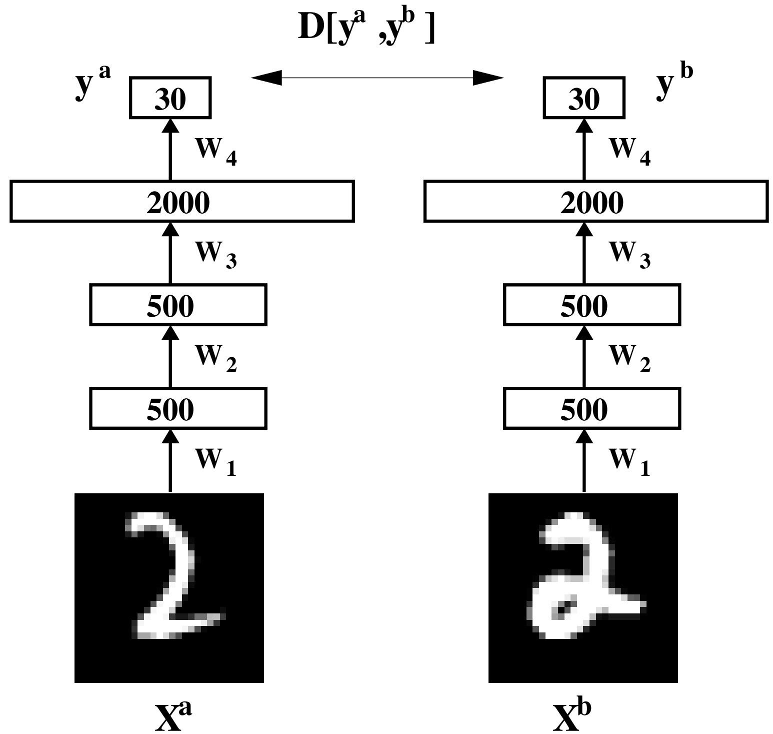
- Small Codes, Torralba, Fergus, Weiss, CVPR 2008
- Spectral Hashing, Y. Weiss, A. Torralba, R. Fergus, NIPS 2008
- Kulis and Darrell, NIPS 2009, Gong and Lazebnik, CVPR 20111
- Norouzi and Fleet, ICML 2011,

Learning Similarity Measures



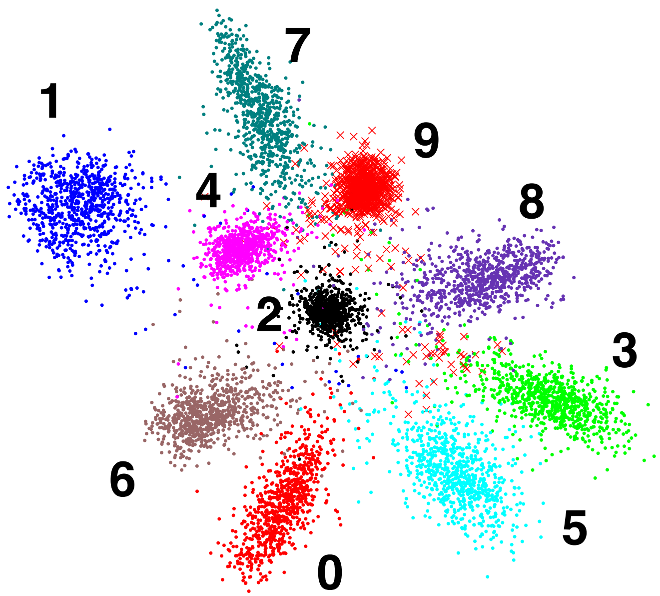
Related to Siamese Networks of LeCun.

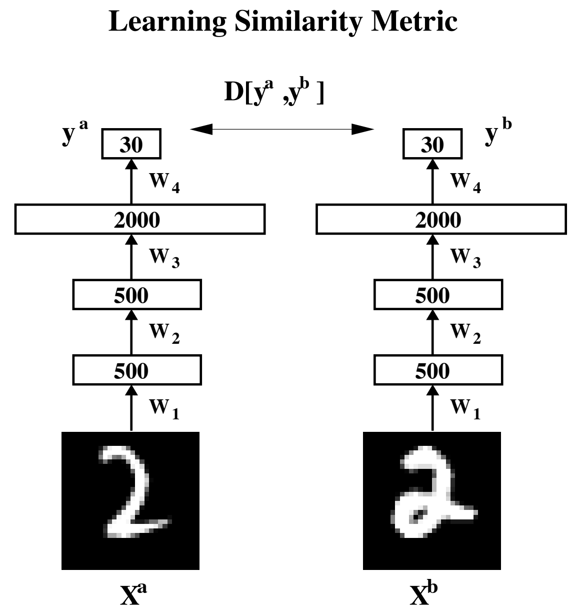
Maximize the Agreement



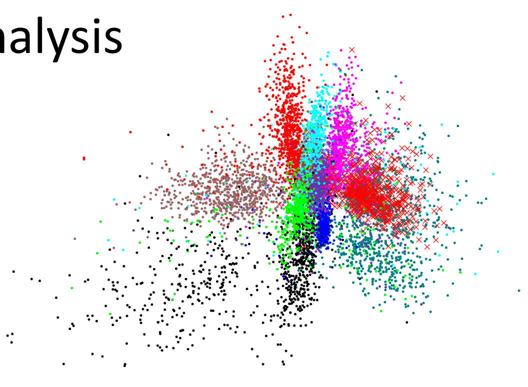
- Learn a nonlinear transformation of the input space.
- Optimize to make KNN perform well in the low-dimensional feature space

Learning Similarity Measures

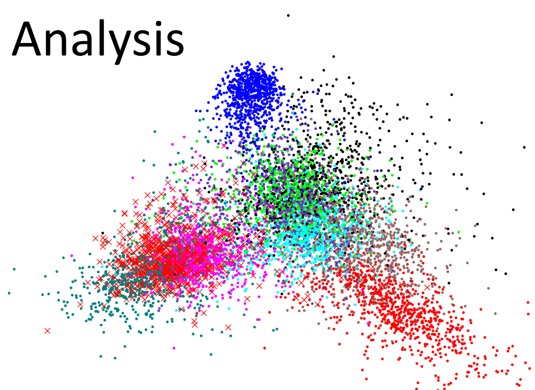




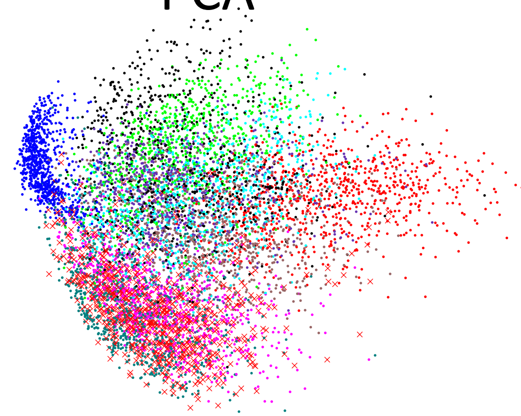
Neighborhood Component Analysis



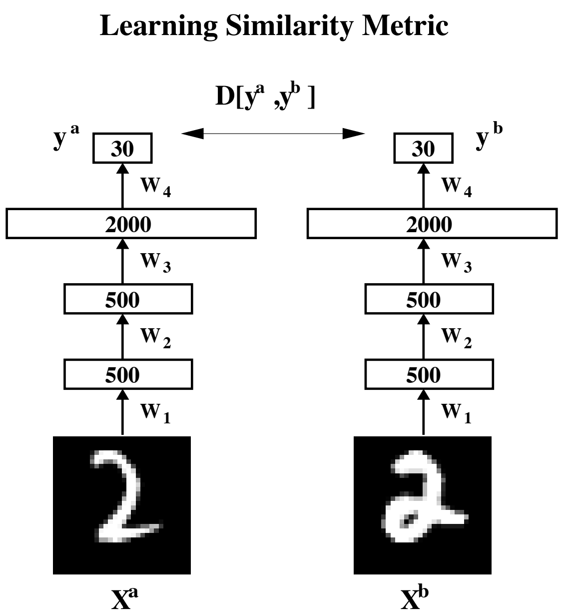
Linear discriminant Analysis

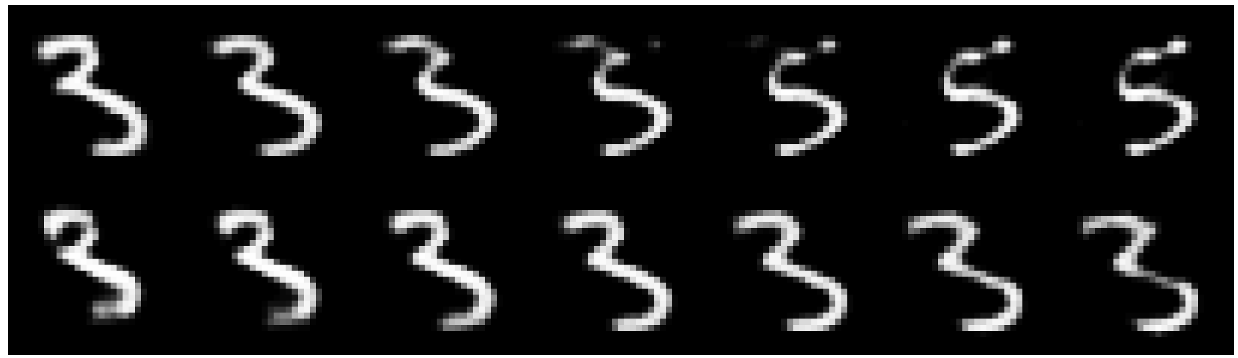


PCA



Learning Similarity Measures





- As we change unit 25 in the code layer, ``3’’ image turns into ``5’’ image

- As we change unit 42 in the code layer, thick ``3’’ image turns into skinny ``3’’.