

10417/10617

Intermediate Deep Learning:

Fall2019

Russ Salakhutdinov

Machine Learning Department

rsalakhu@cs.cmu.edu

<https://deeplearning-cmu-10417.github.io/>

Autoencoders

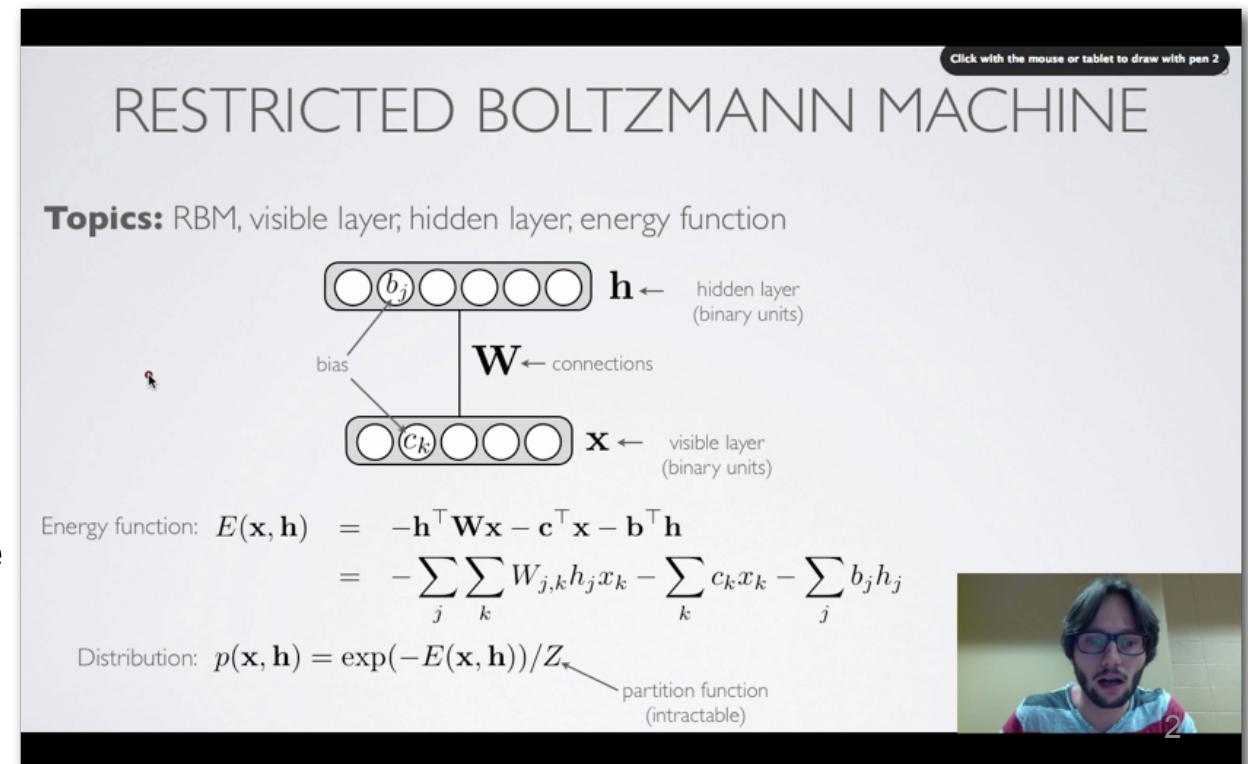
Neural Networks Online Course

- **Disclaimer:** Much of the material and slides for this lecture were borrowed from Hugo Larochelle's class on Neural Networks:

- Hugo's class covers many other topics: convolutional networks, neural language model, Boltzmann machines, autoencoders, sparse coding, etc.

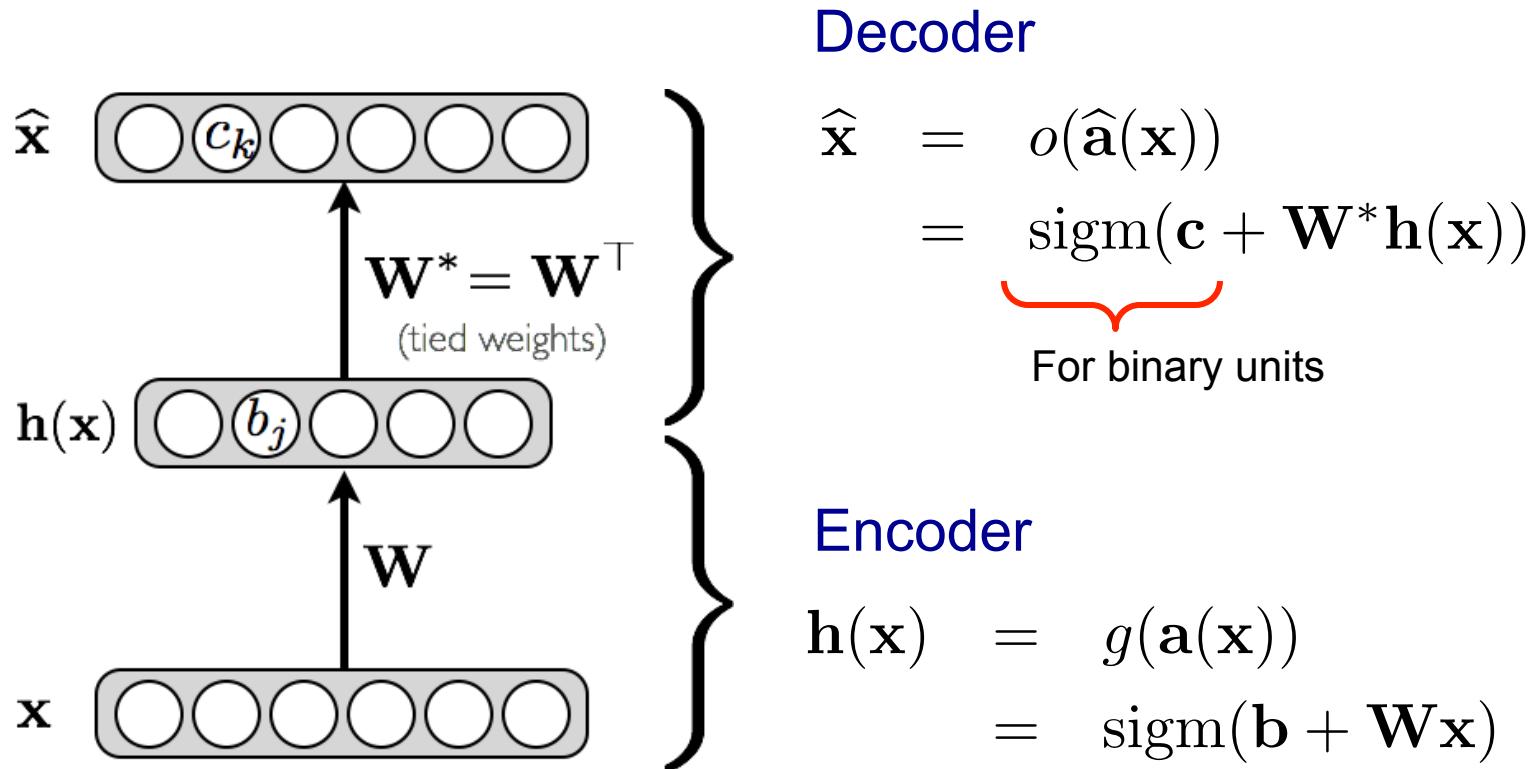
- We will use his material for some of the other lectures.

http://info.usherbrooke.ca/hlarochelle/neural_networks

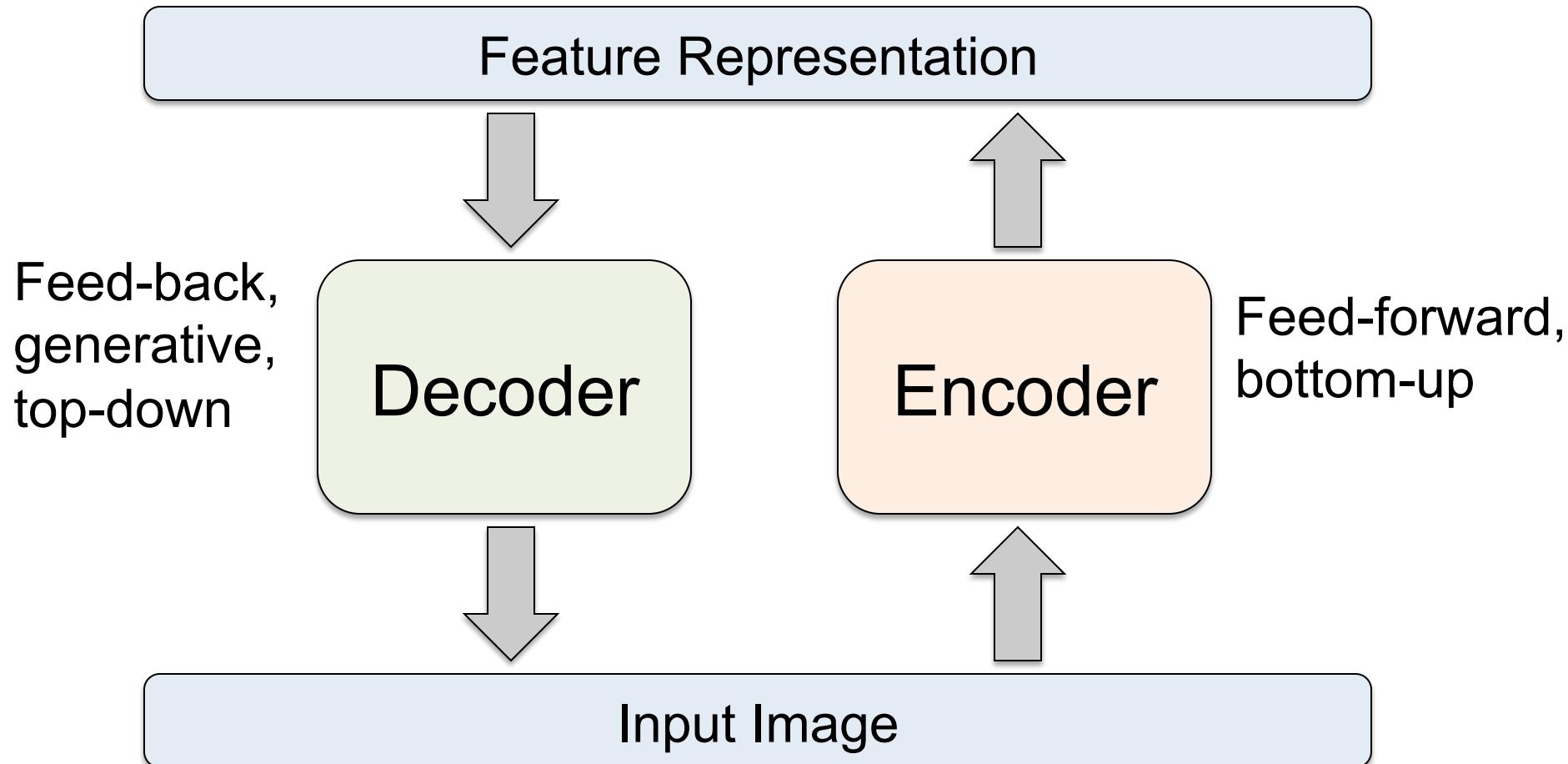


Autoencoders

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

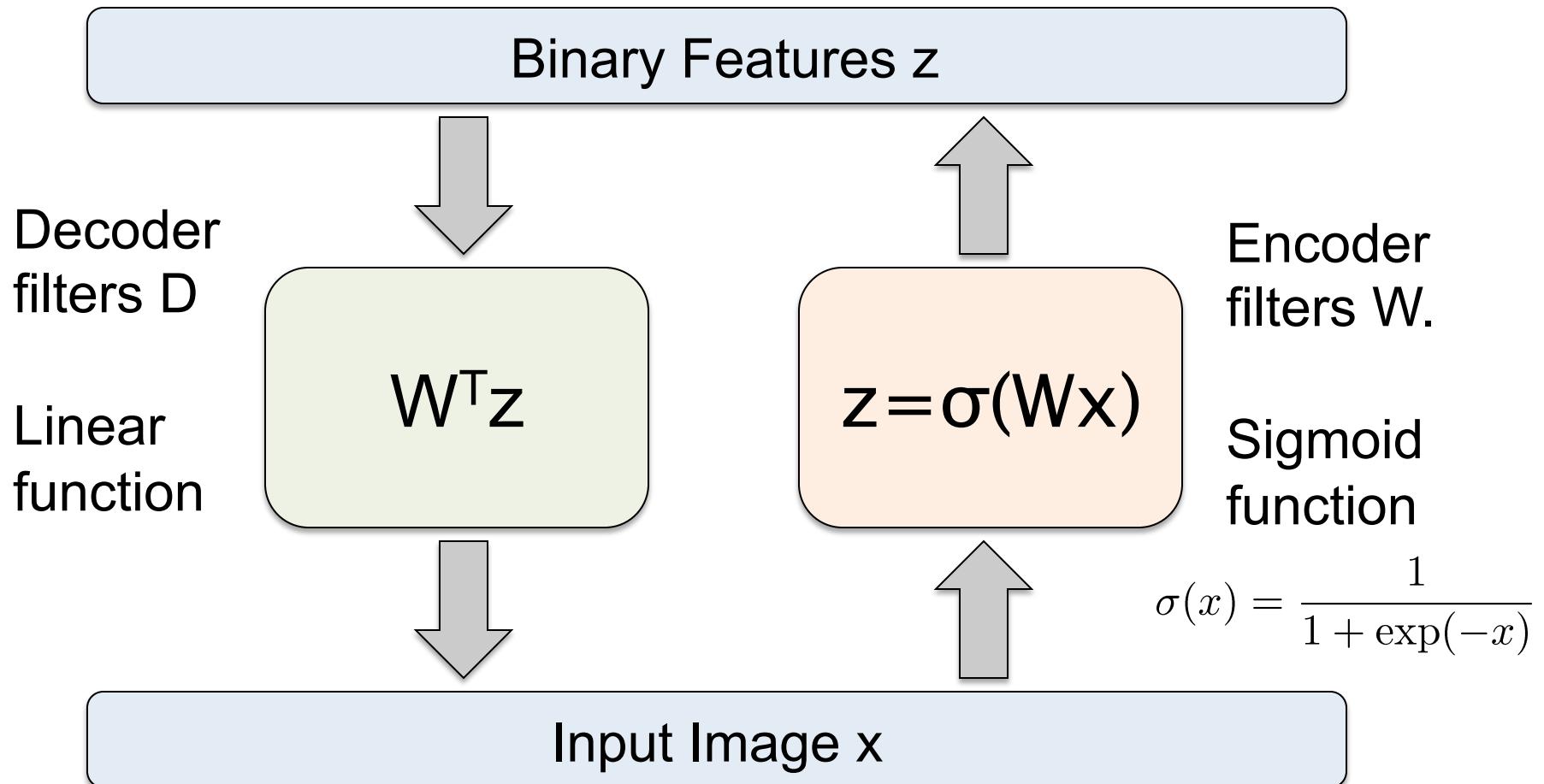


Autoencoders

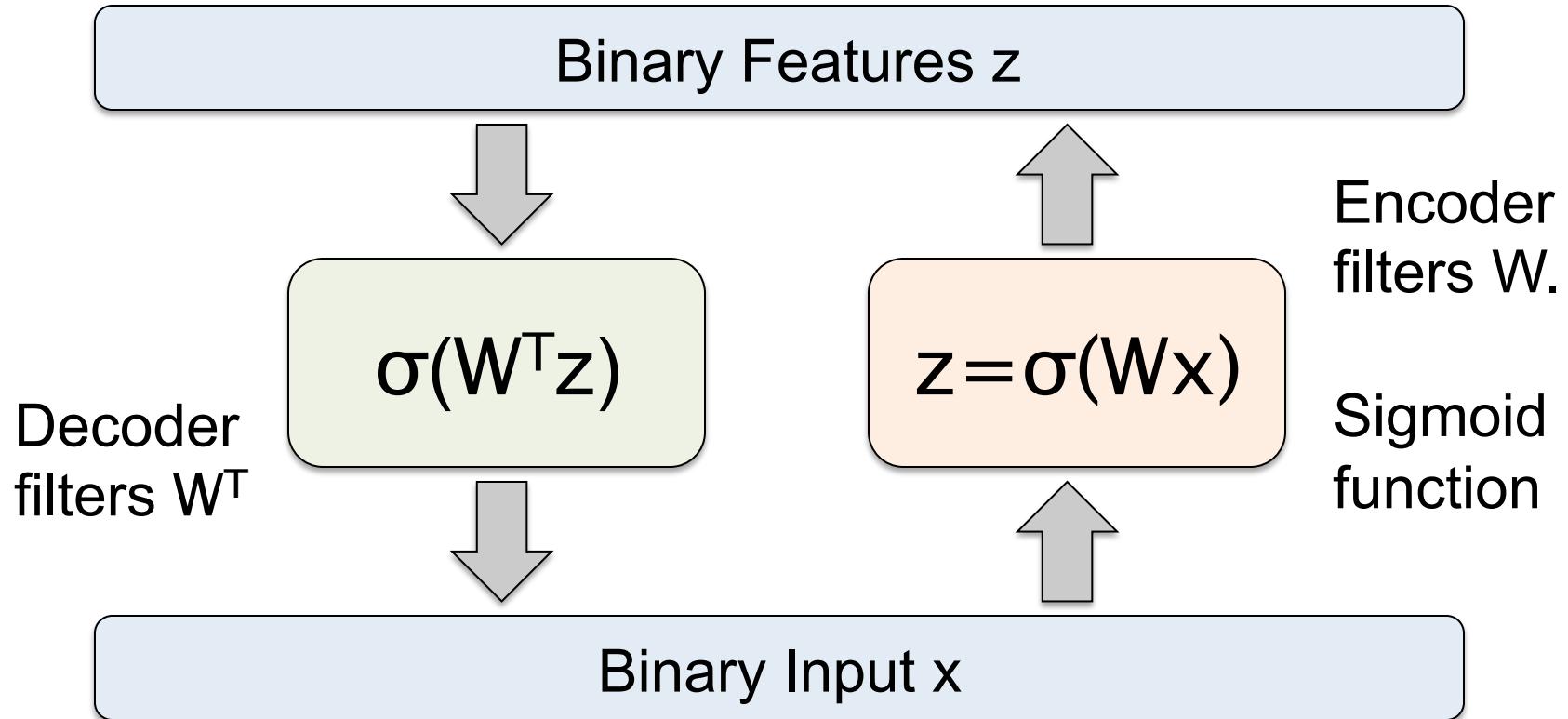


- Details of what goes inside the encoder and decoder matter!
- Need constraints to avoid learning an identity.

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

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

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

- **Loss function** for real-valued inputs

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

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

Loss Function

- For both cases, the gradient $\nabla_{\hat{\mathbf{a}}(\mathbf{x}^{(t)})} l(f(\mathbf{x}^{(t)}))$ has a very simple form:

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

- Parameter gradients are obtained by backpropagating the gradient $\nabla_{\hat{\mathbf{a}}(\mathbf{x}^{(t)})} l(f(\mathbf{x}^{(t)}))$ 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 $\boldsymbol{\mu} = \mathbf{c} + \mathbf{W}^* \mathbf{h}(\mathbf{x})$

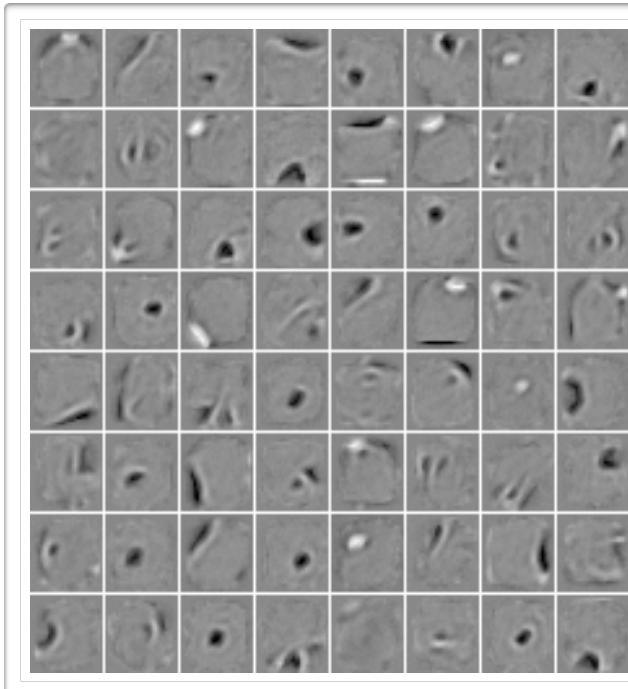
Example: MNIST

- MNIST dataset:

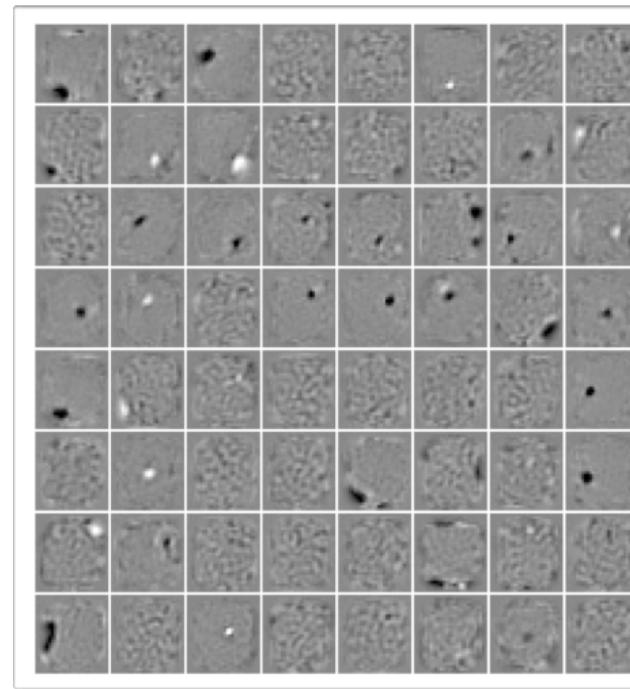
3	8	6	9	6	4	5	3	8	4	5	2	3	8	4	8
1	5	0	5	9	7	4	1	0	3	0	6	2	9	9	4
1	3	6	8	0	7	1	6	8	9	0	3	8	3	7	7
8	4	4	1	2	9	4	1	1	0	6	6	5	0	1	1
7	2	7	3	1	4	0	5	0	6	8	7	6	8	9	9
4	0	6	1	9	2	1	3	9	4	4	5	6	6	1	7
2	8	6	9	7	0	9	1	6	2	8	3	6	4	9	5
8	6	8	7	8	8	6	9	1	7	6	0	9	6	7	0

Learned Features

- MNIST dataset:

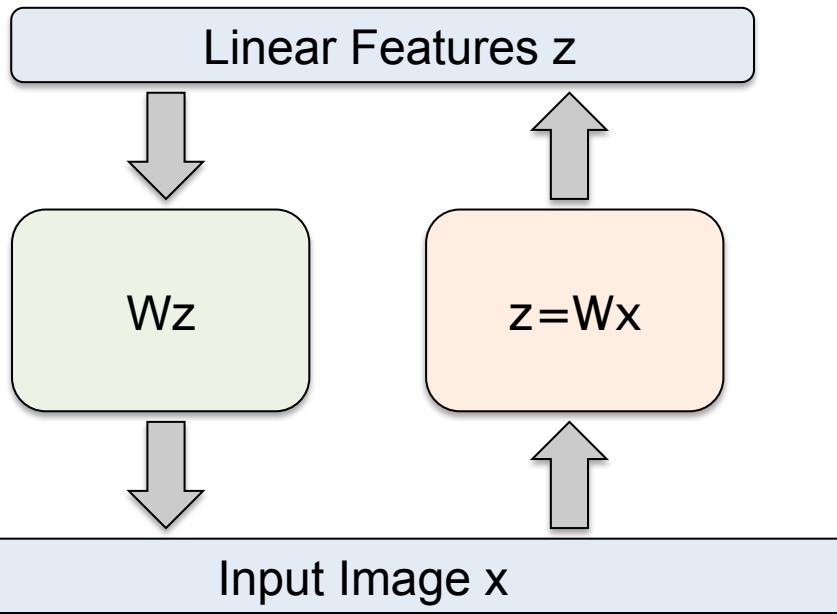


RBM



Autoencoder

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} :

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

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

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

matrix where columns are $\mathbf{x}^{(t)}$
(could be any encoder)

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

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:

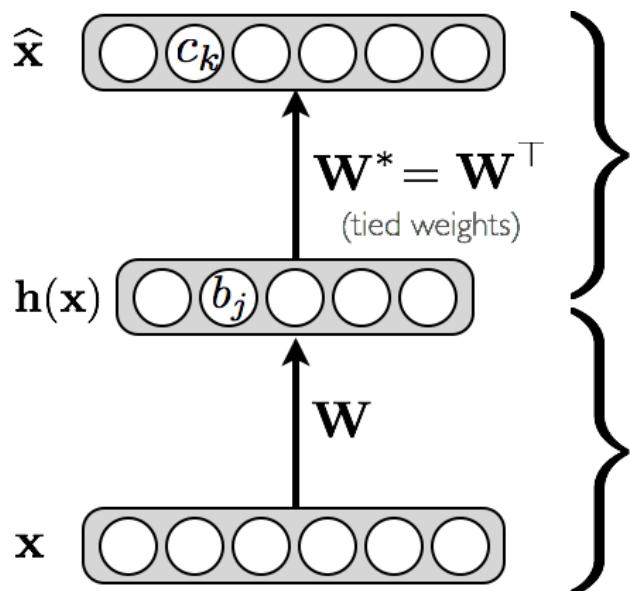
$$\begin{aligned} \mathbf{h}(\mathbf{X}) &= \mathbf{V}_{\cdot, \leq k}^\top \\ &= \mathbf{V}_{\cdot, \leq k}^\top (\mathbf{X}^\top \mathbf{X})^{-1} (\mathbf{X}^\top \mathbf{X}) && \leftarrow \text{multiplying by identity} \\ &= \mathbf{V}_{\cdot, \leq k}^\top (\mathbf{V} \Sigma^\top \mathbf{U}^\top \mathbf{U} \Sigma \mathbf{V}^\top)^{-1} (\mathbf{V} \Sigma^\top \mathbf{U}^\top \mathbf{X}) && \leftarrow \text{replace with SVD} \end{aligned}$$

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

Optimality of the Linear Autoencoder

- So an **optimal pair** of encoder and decoder is

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

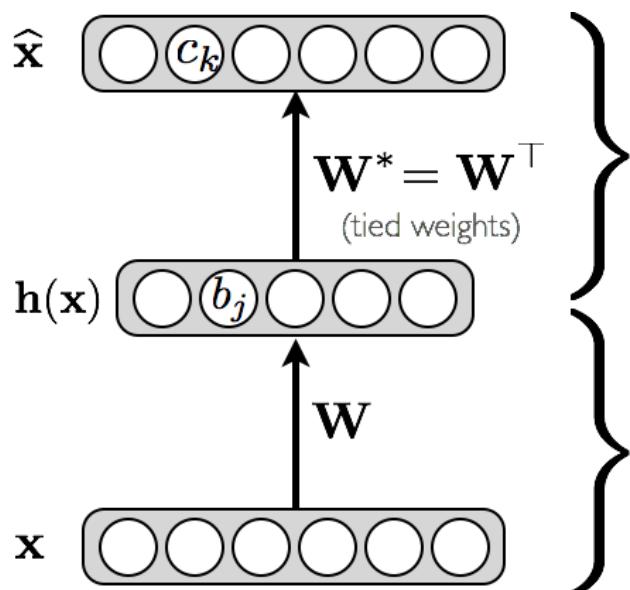


- for the sum of squared difference error
- for an autoencoder with a linear decoder
- where optimality means “has the lowest training reconstruction error”

Optimality of the Linear Autoencoder

- So an optimal pair of encoder and decoder is

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



- 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

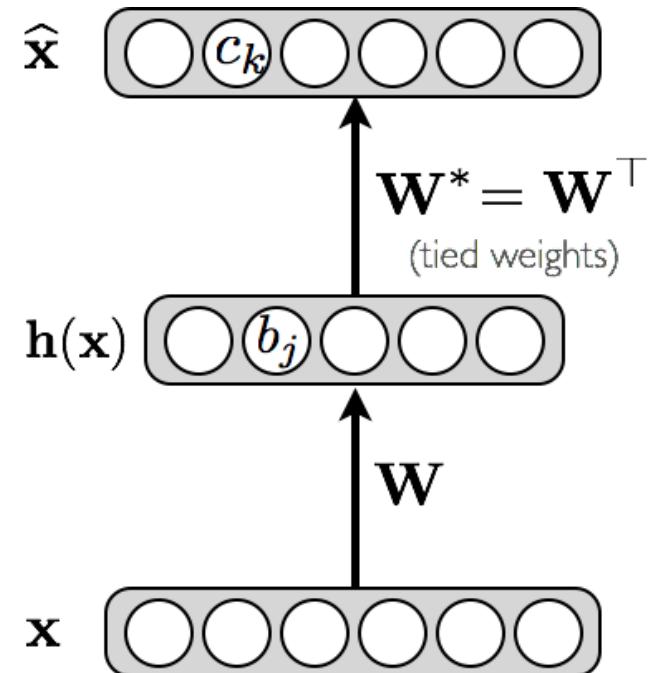
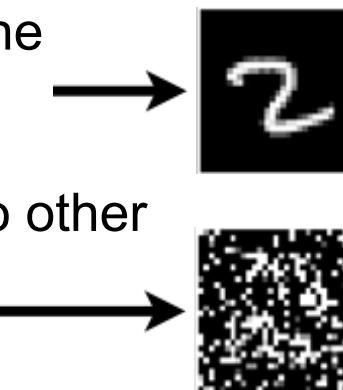
Undercomplete Representation

- 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

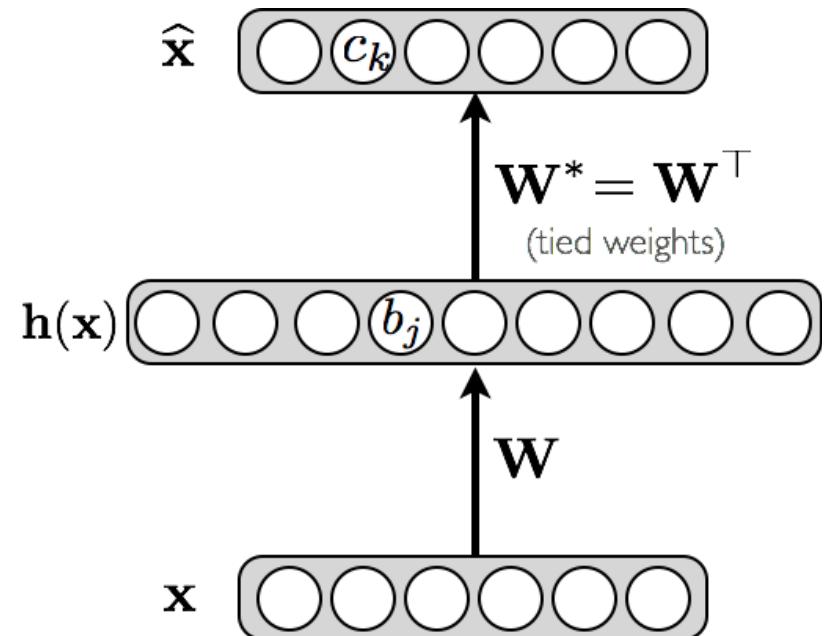
- Hidden units will be

- good features for the training 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**

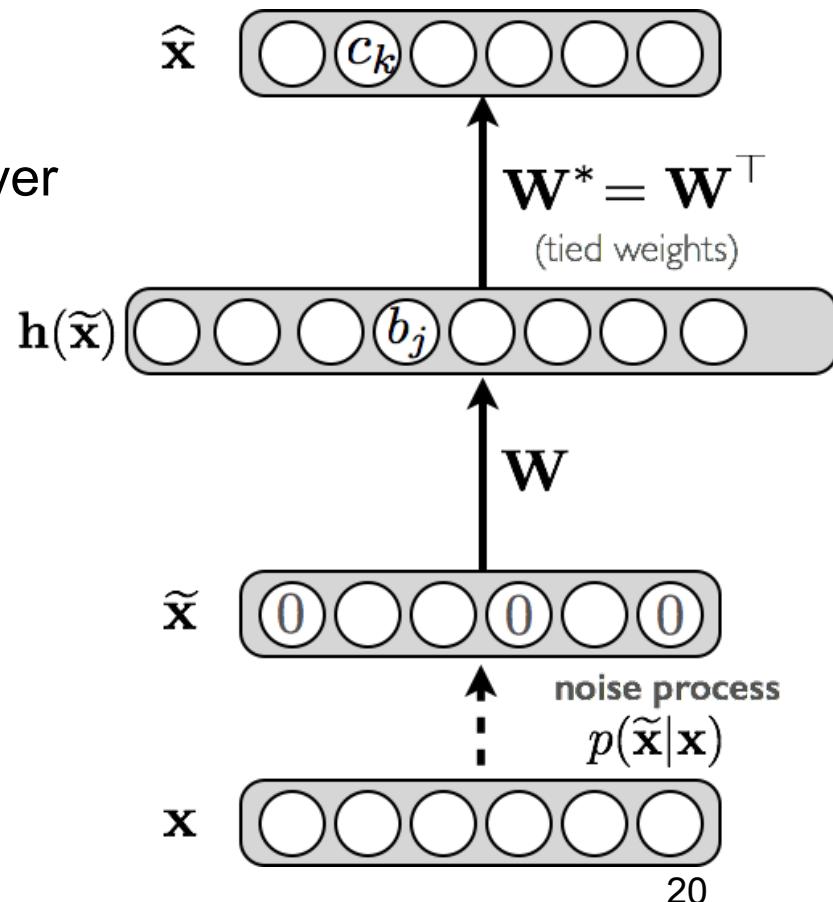


Denoising Autoencoder

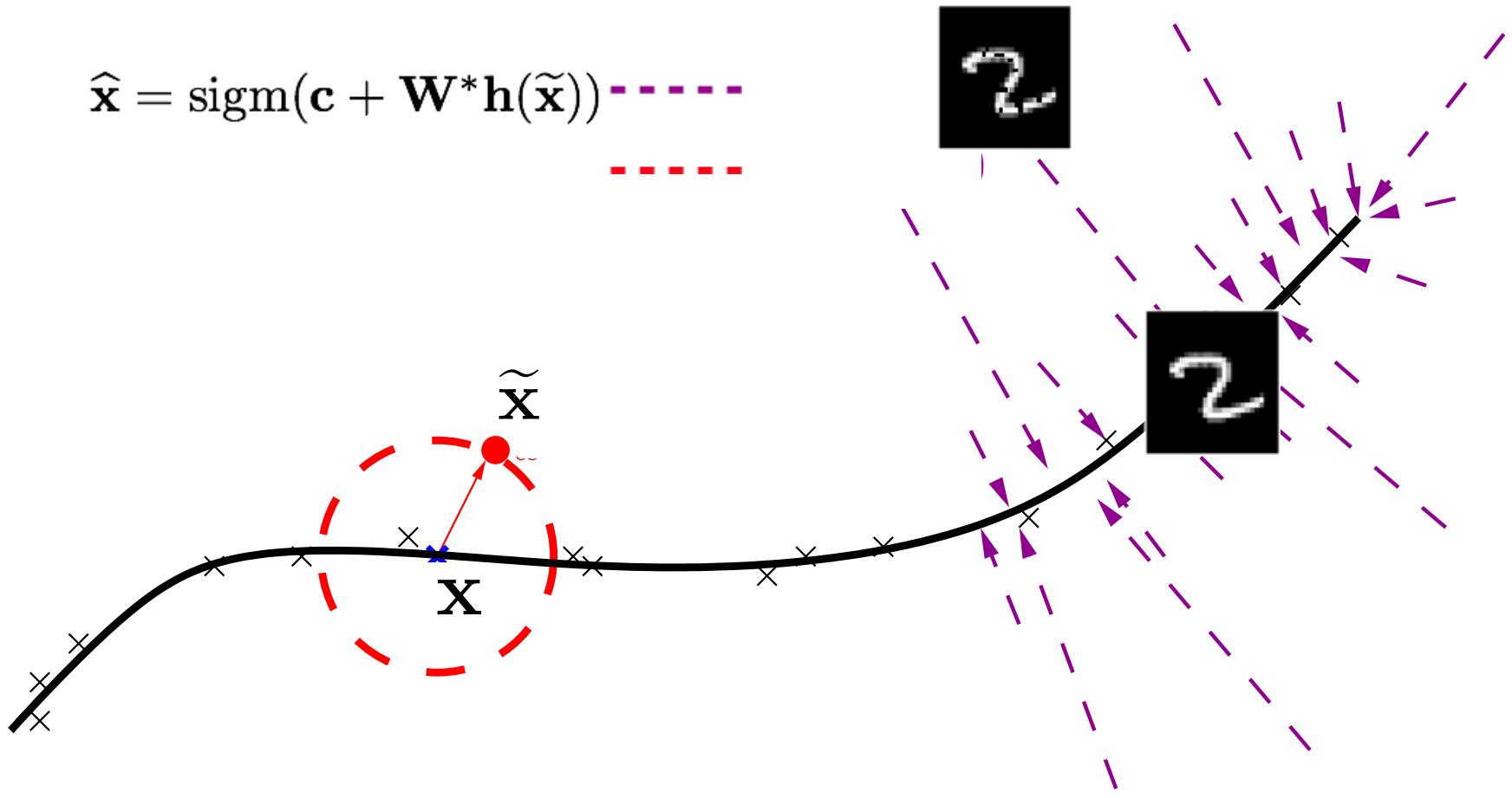
- **Idea:** representation should be robust to introduction of noise:

- random assignment of subset of inputs to 0, with probability ν
- Similar to dropouts on the input layer
- Gaussian additive noise

- **Reconstruction $\hat{\mathbf{x}}$** computed from the corrupted input $\tilde{\mathbf{x}}$
- **Loss function** compares $\hat{\mathbf{x}}$ reconstruction with the noiseless input \mathbf{x}

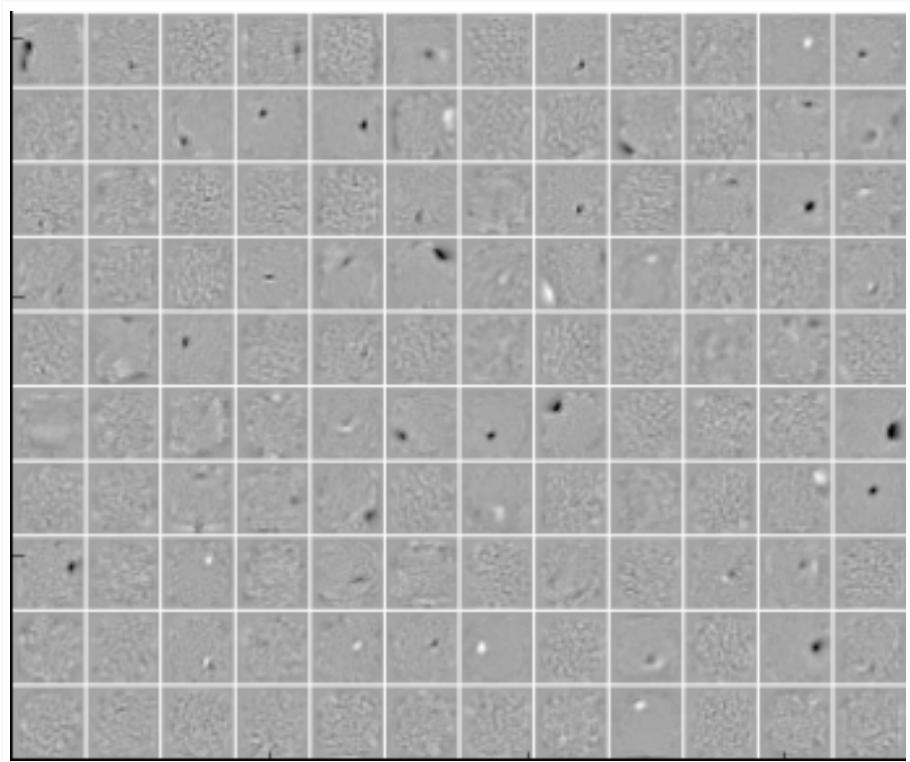


Denoising Autoencoder

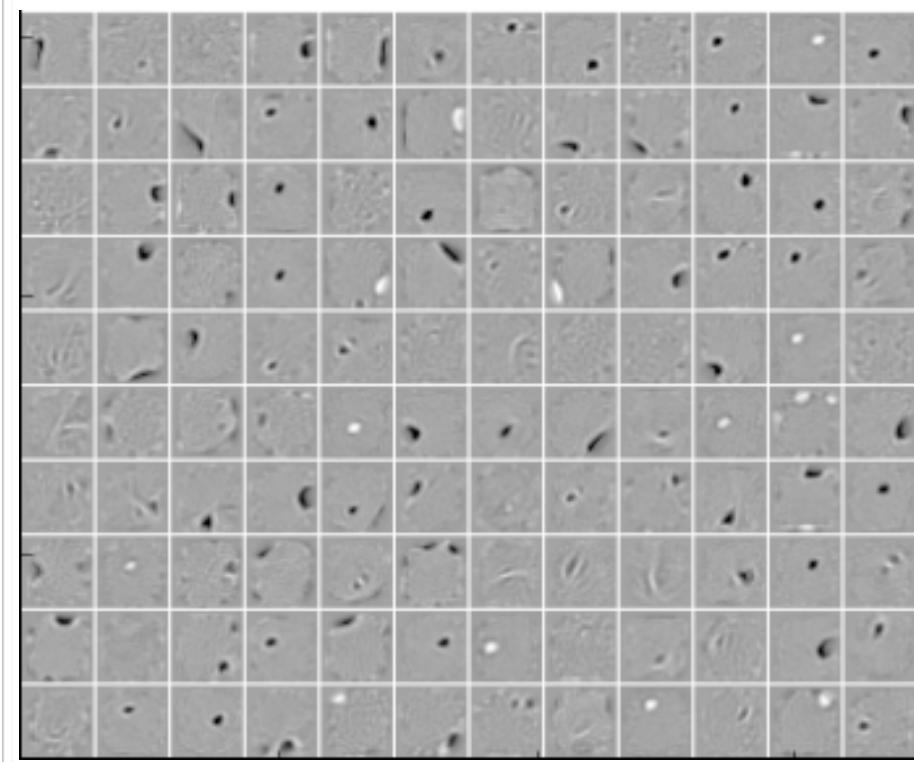


Learned Filters

Non-corrupted

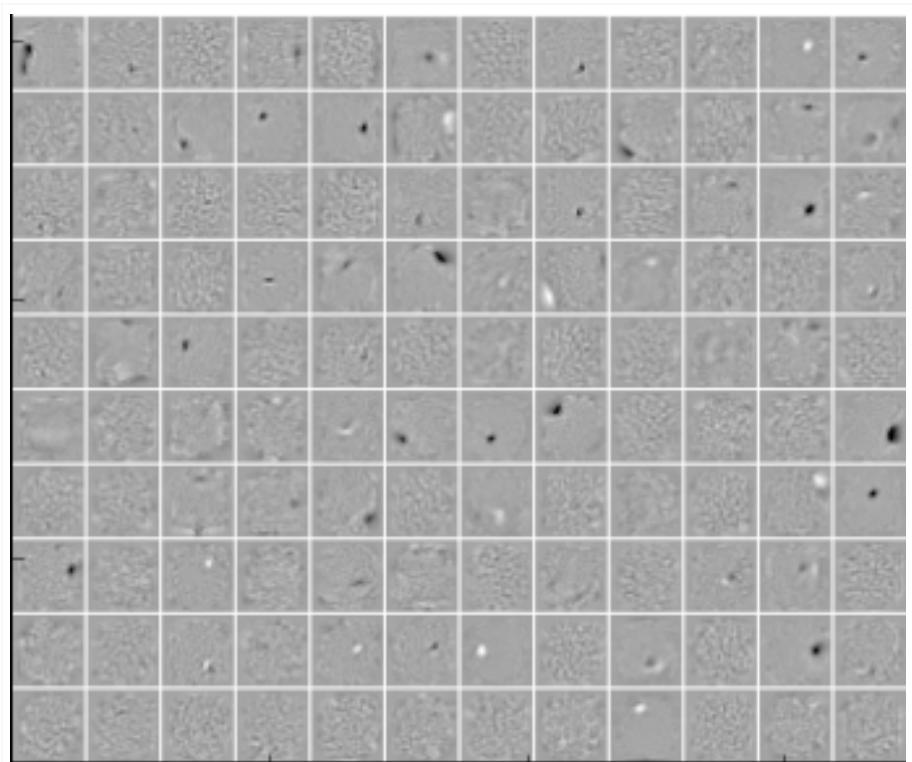


25% corrupted input

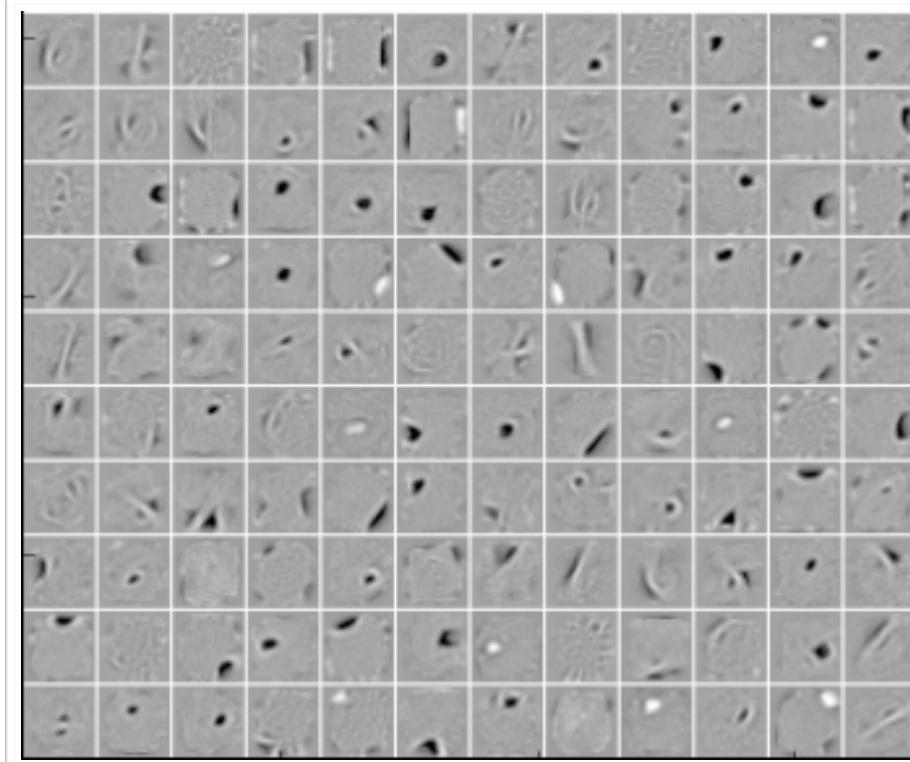


Learned Filters

Non-corrupted

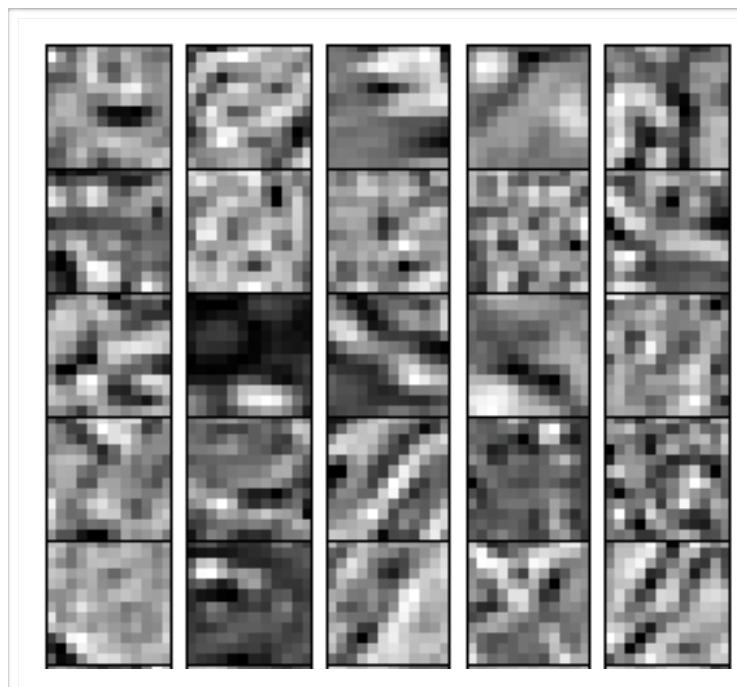


50% corrupted input

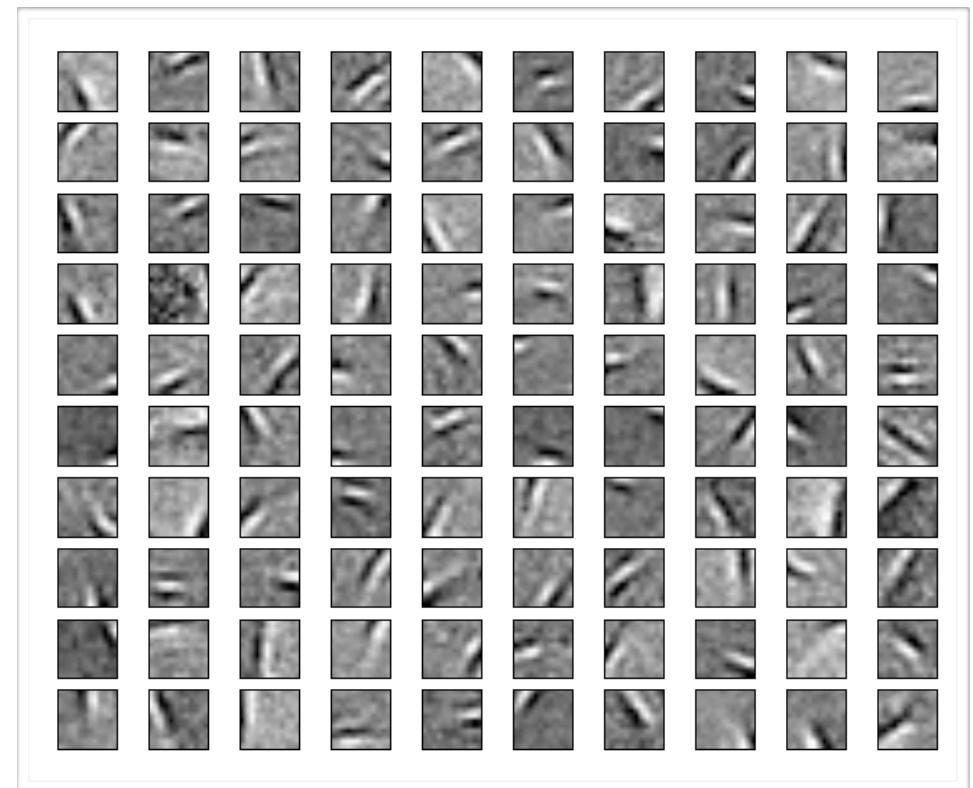


Squared Error Loss

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



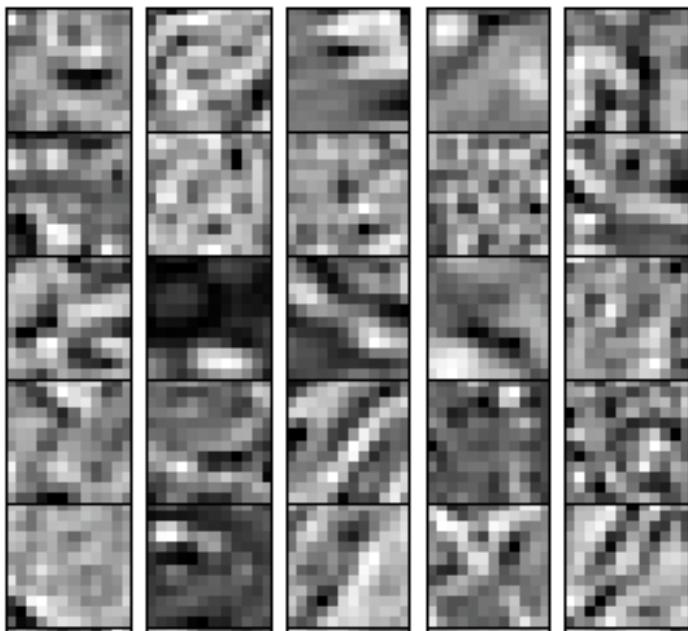
Data



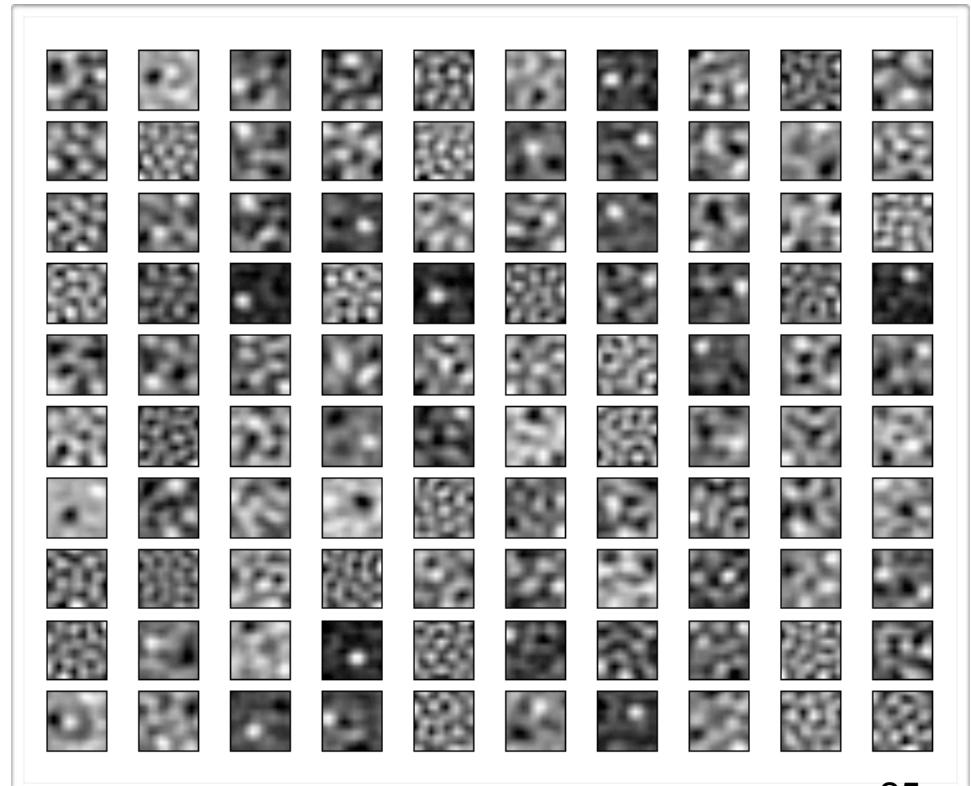
Filters

Squared Error Loss

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



Data



Filters

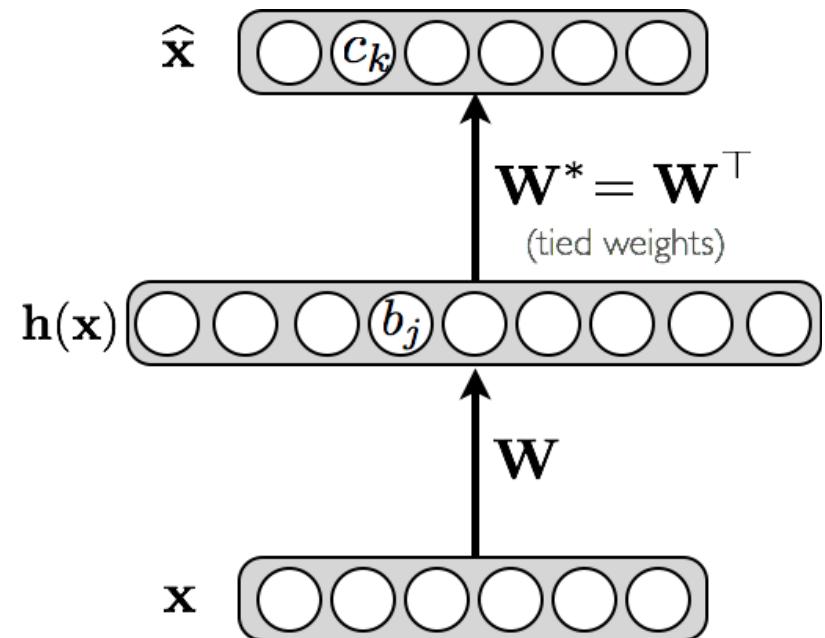
Contractive Autoencoders

- 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



Contractive Autoencoders

- Consider the following loss function:

$$l(f(\mathbf{x}^{(t)})) + \lambda \underbrace{||\nabla_{\mathbf{x}^{(t)}} \mathbf{h}(\mathbf{x}^{(t)})||_F^2}_{\text{Jacobian of Encoder}}$$

$\underbrace{l(f(\mathbf{x}^{(t)}))}_{\text{Reconstruction Loss}}$

- For the binary observations:

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

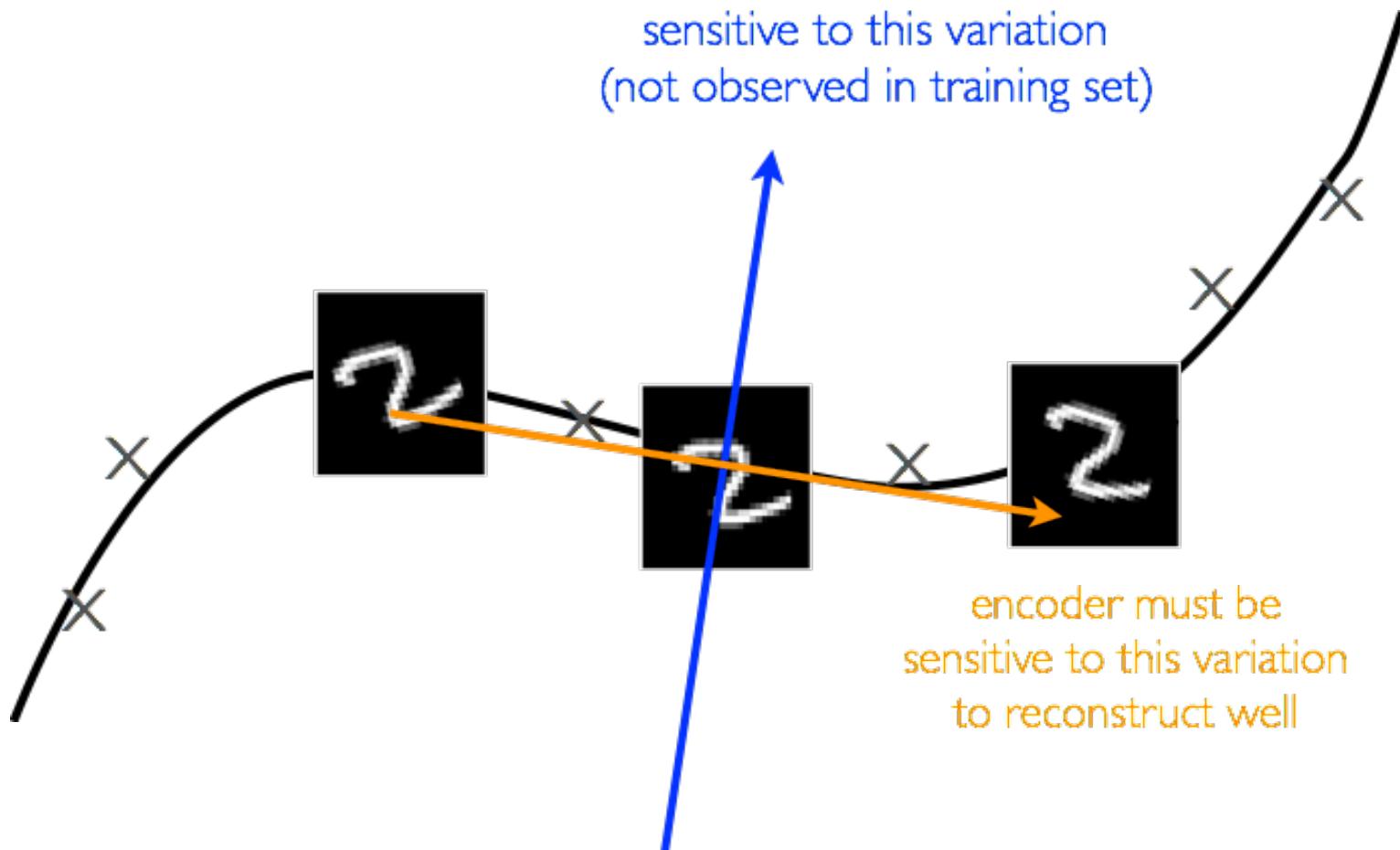
$$||\nabla_{\mathbf{x}^{(t)}} \mathbf{h}(\mathbf{x}^{(t)})||_F^2 = \sum_j \sum_k \left(\frac{\partial h(\mathbf{x}^{(t)})_j}{\partial x_k^{(t)}} \right)^2$$

Encoder throws away all information

Autoencoder attempts to preserve all information

Contractive Autoencoders

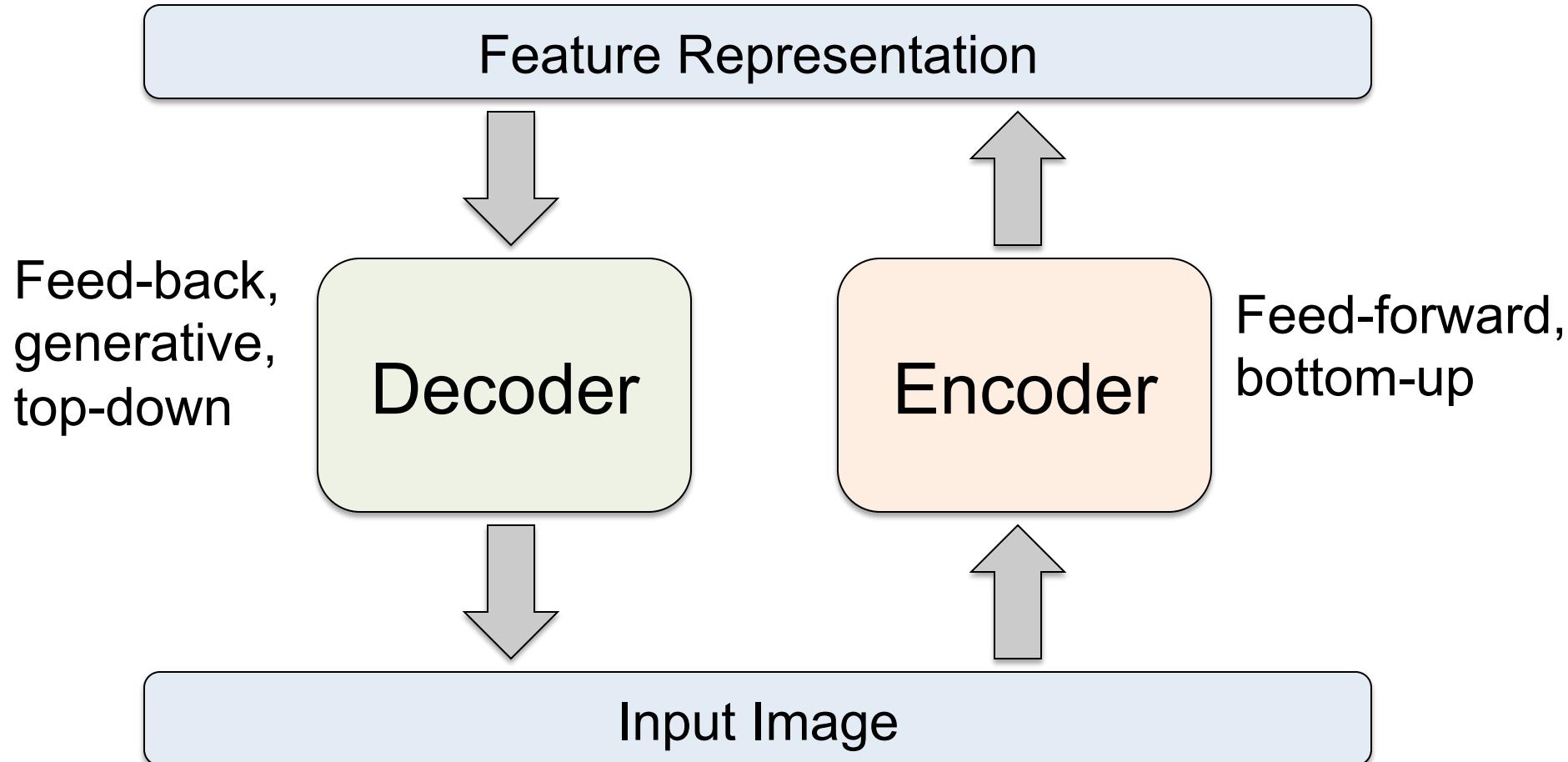
- Illustration:



Pros and Cons

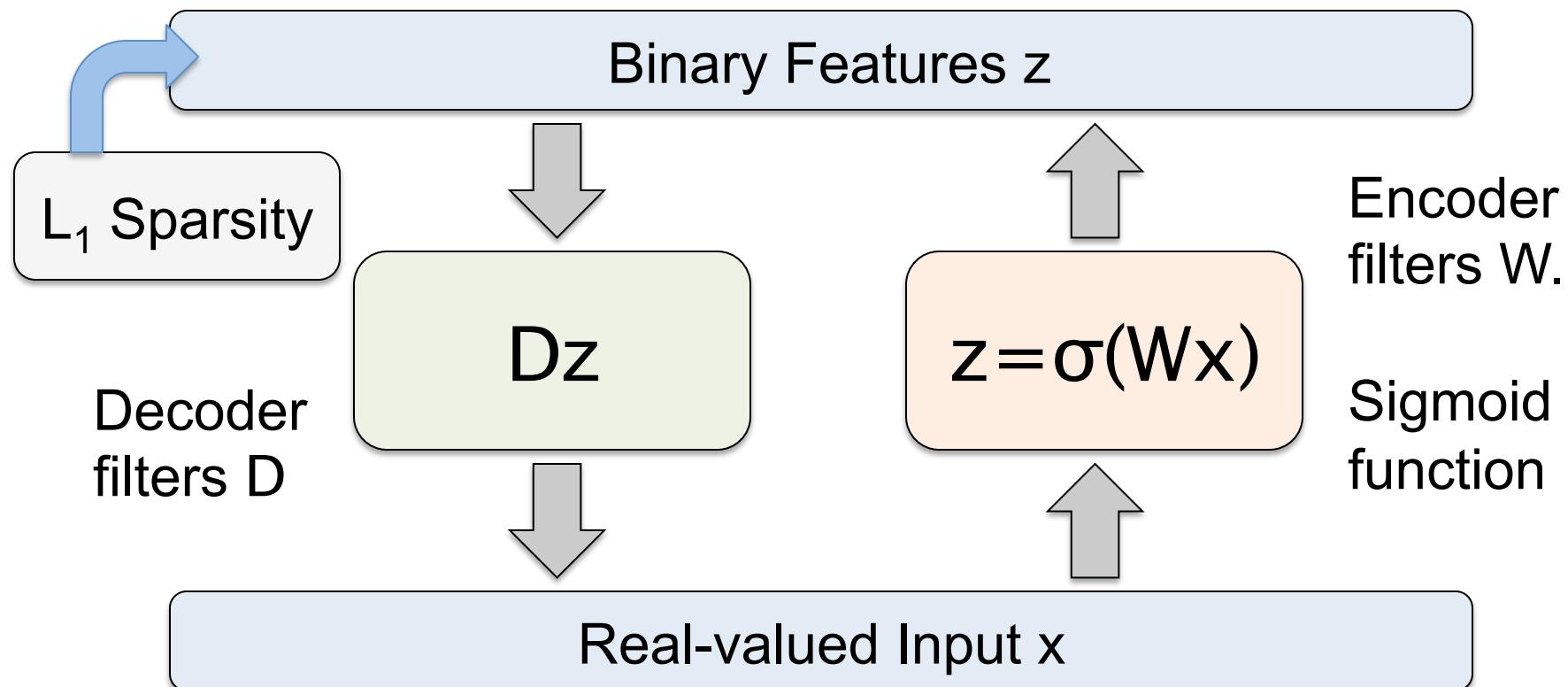
- Advantage of **denoising autoencoder**: simpler to implement
 - requires adding one or two lines of code to regular autoencoder
 - no need to compute Jacobian of hidden layer
- Advantage of **contractive autoencoder**: gradient is deterministic
 - can use second order optimizers (conjugate gradient, LBFGS, etc.)
 - might be more stable than denoising autoencoder, which uses a sampled gradient

Autoencoder



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

Predictive Sparse Decomposition



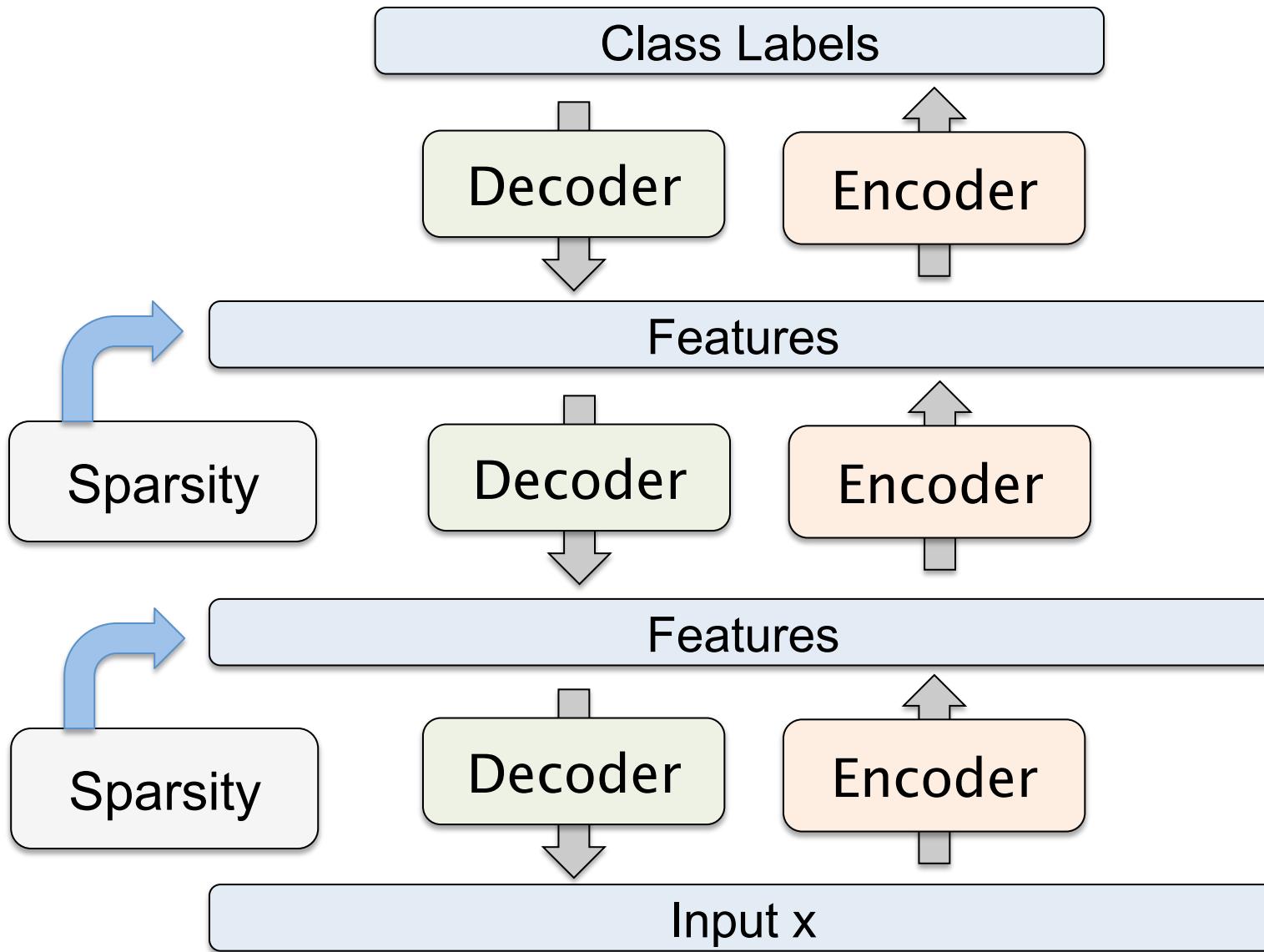
At training time

$$\min_{D, W, z} \|Dz - x\|_2^2 + \lambda|z|_1 + \|\sigma(Wx) - z\|_2^2$$

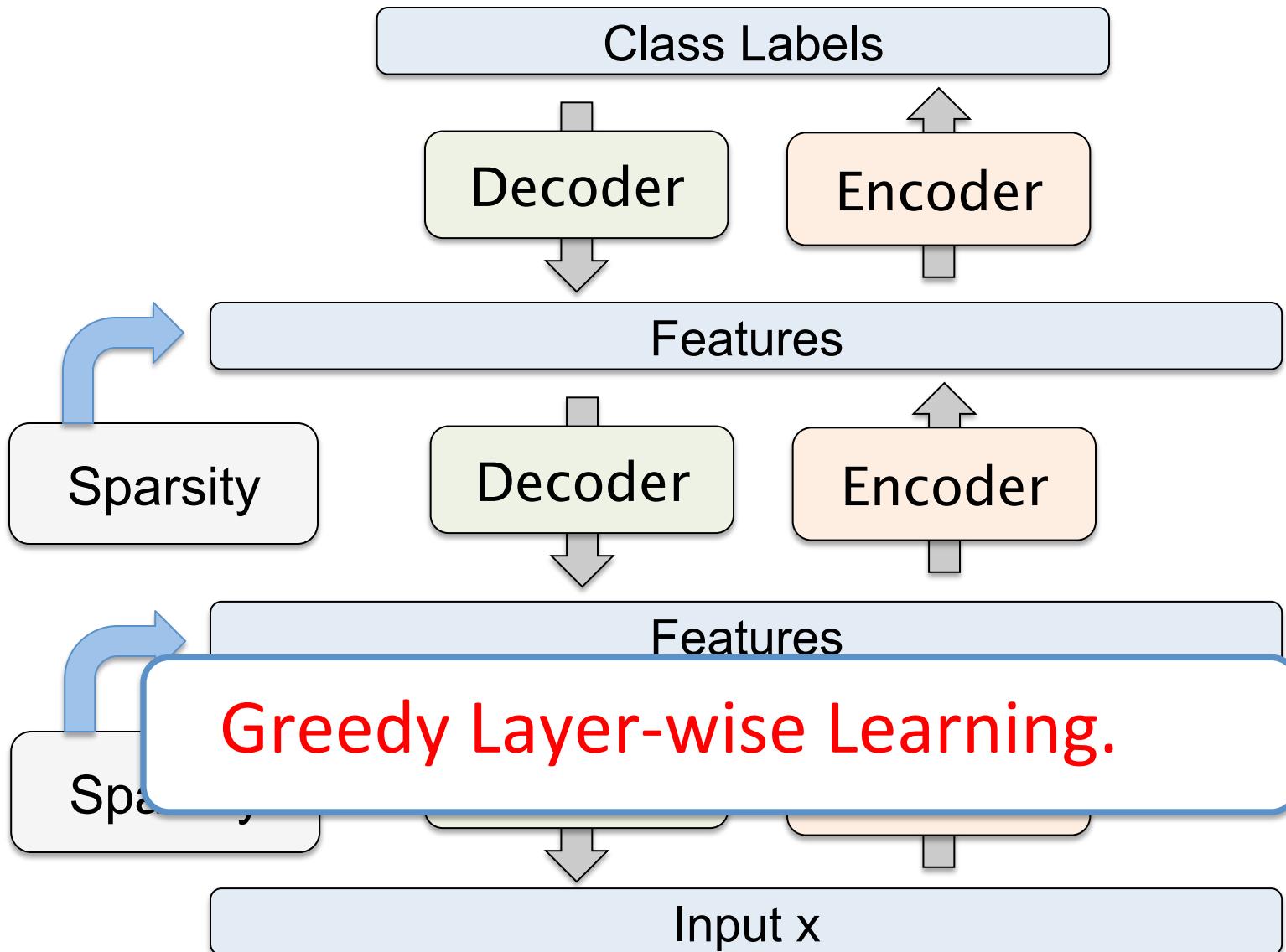
Decoder

Encoder

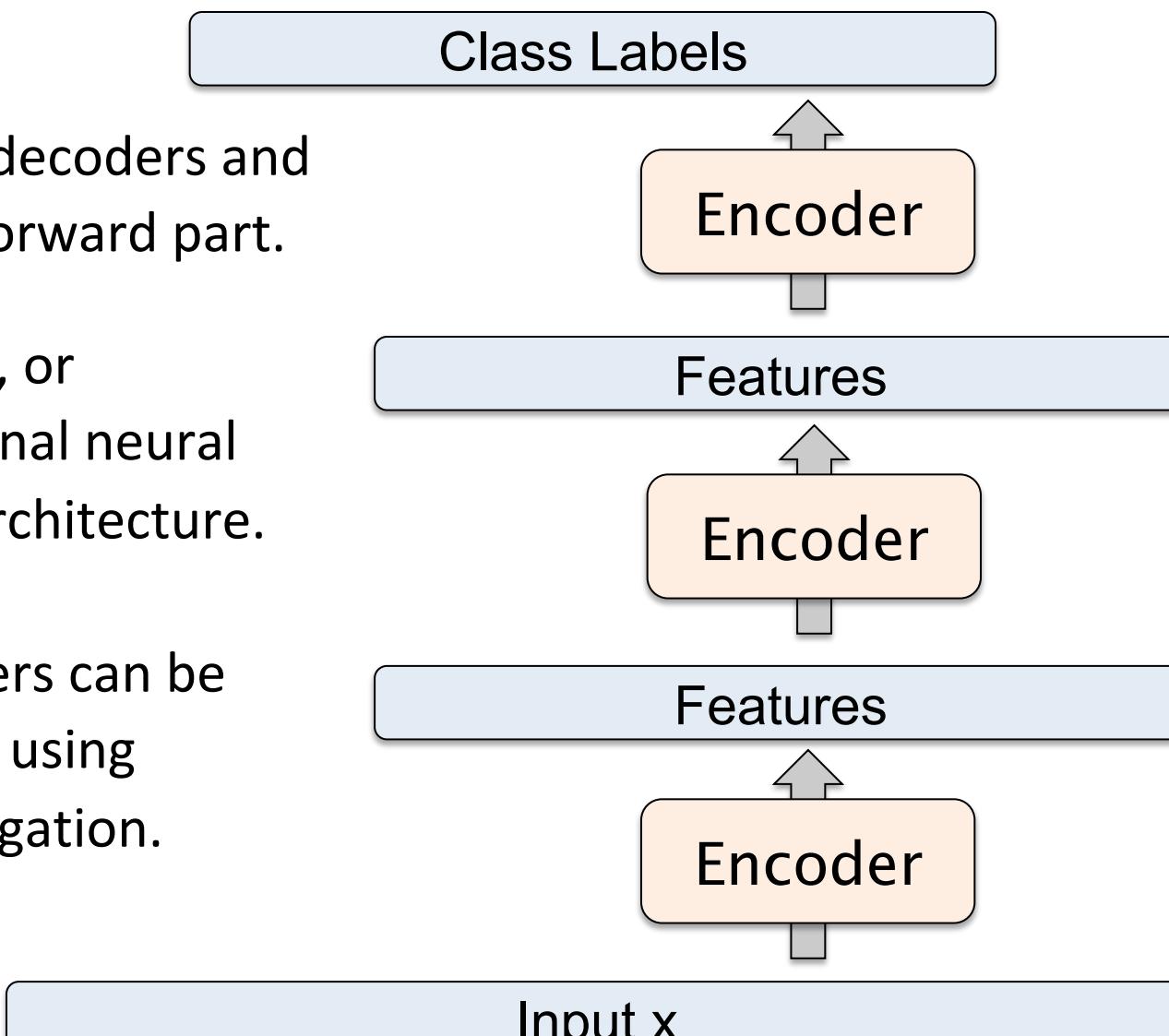
Stacked Autoencoders



Stacked Autoencoders



Stacked Autoencoders



```
graph TD; Input[Input x] --> E1[Encoder]; E1 --> F1[Features]; F1 --> E2[Encoder]; E2 --> F2[Features]; F2 --> E3[Encoder]; E3 --> CL[Class Labels];
```

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

Class Labels

Encoder

Features

Encoder

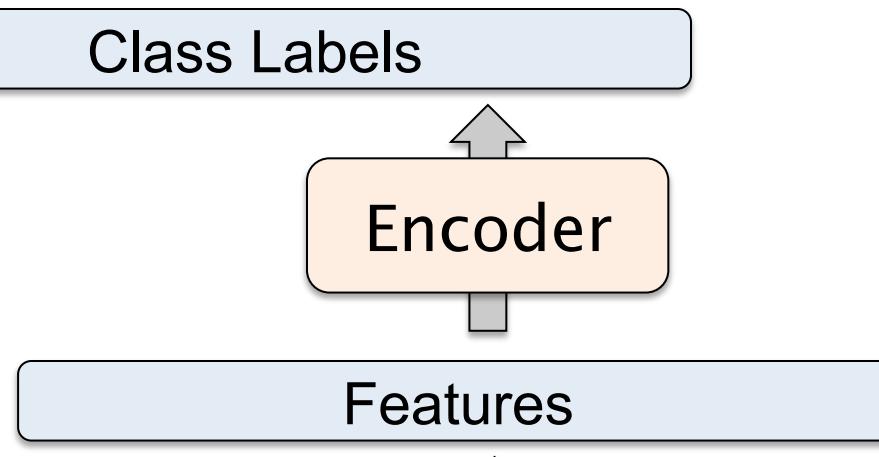
Features

Encoder

Input x

Stacked Autoencoders

- Remove decoders and use feed-forward part.

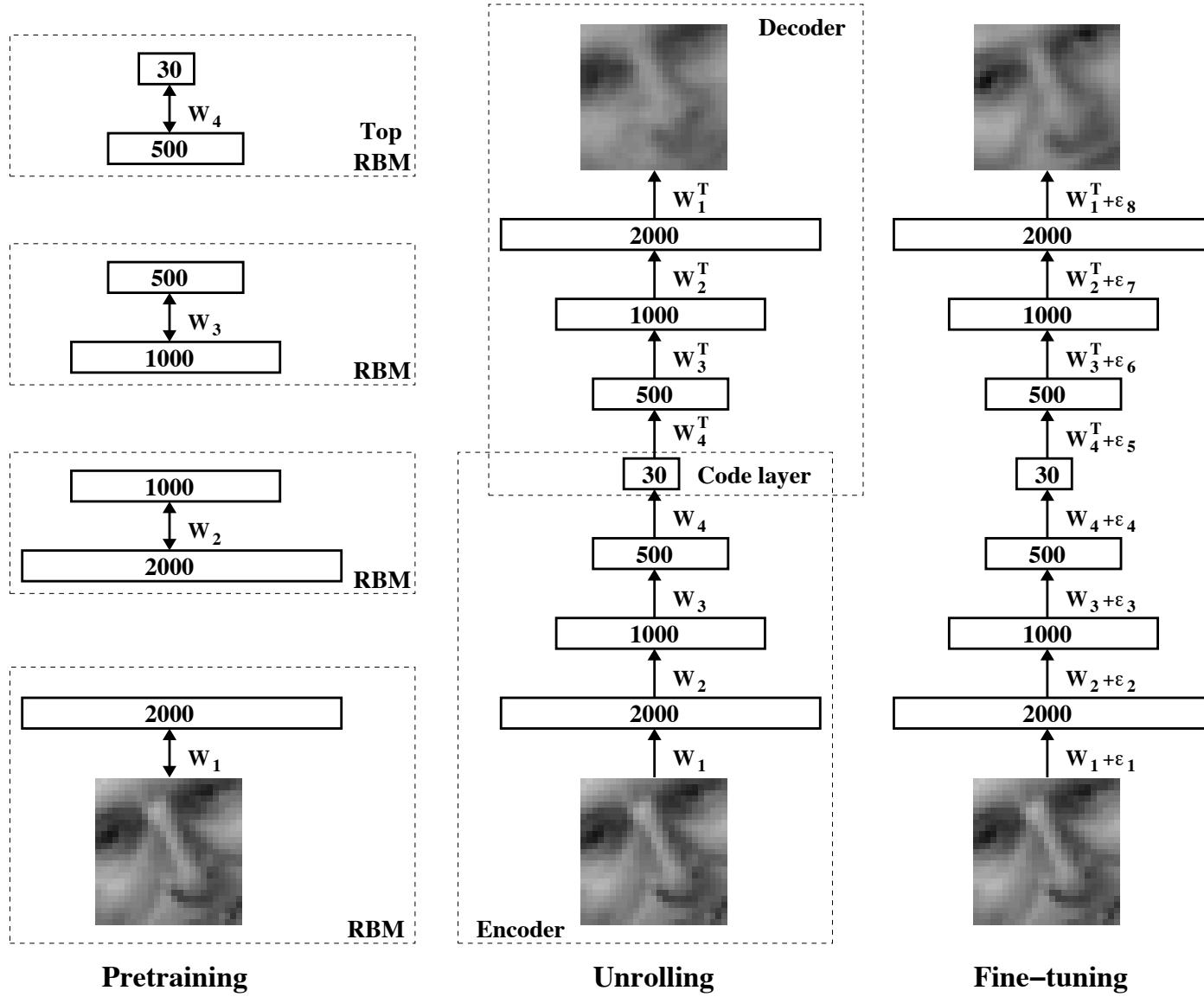


- Standard, or convolutional neural network architecture.

- Parameter fine-tuning, backpropagation

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

Deep Autoencoders



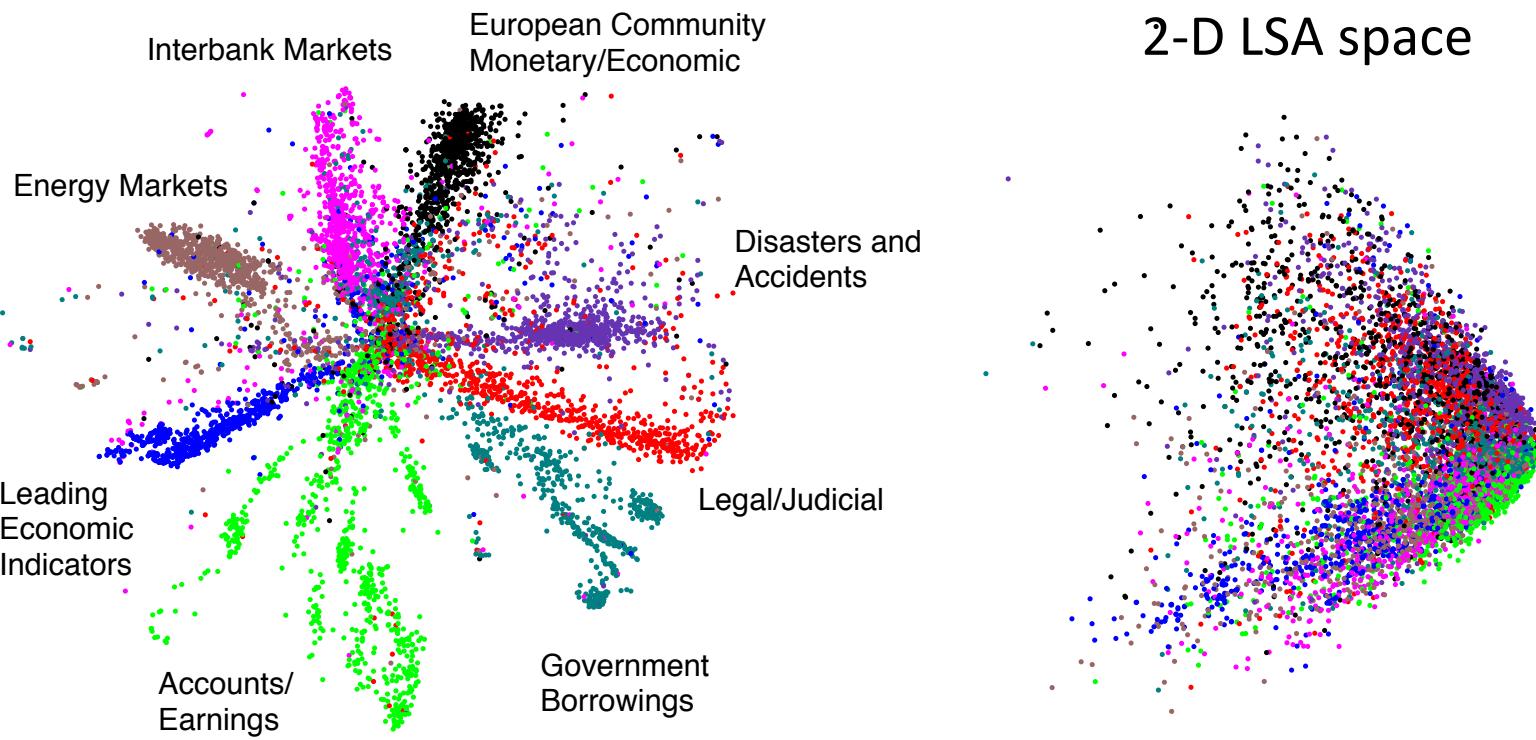
Deep Autoencoders

- We used $25 \times 25 - 2000 - 1000 - 500 - 30$ 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-dimensional PCA.

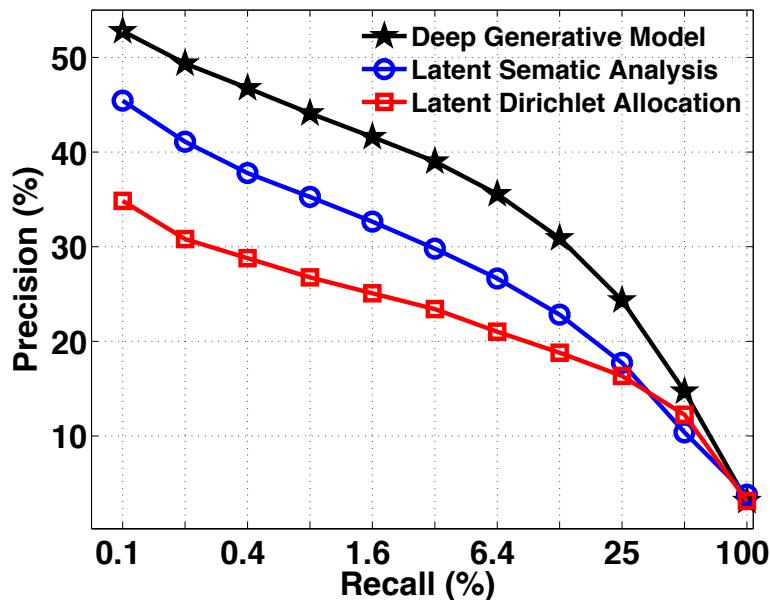
Information Retrieval



- 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

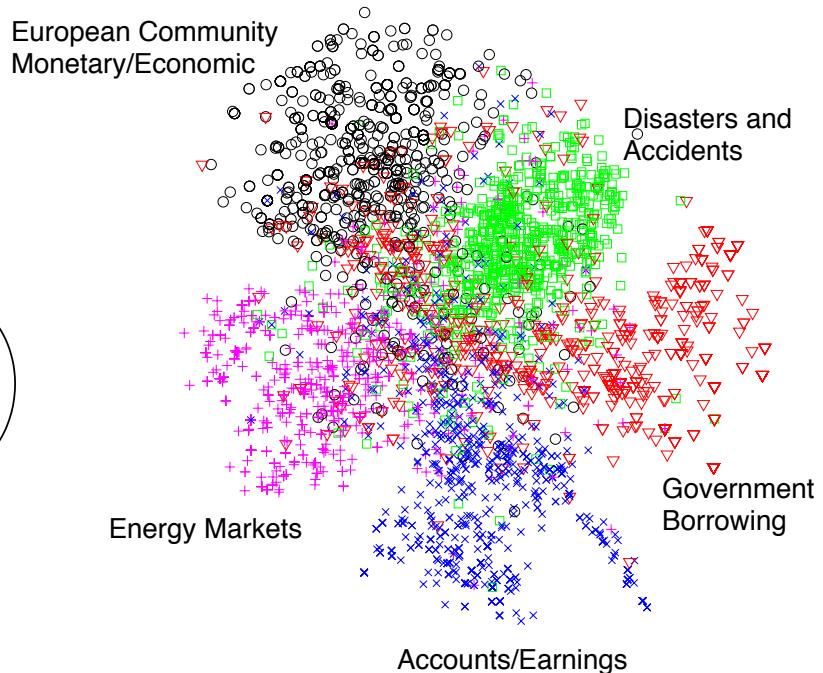
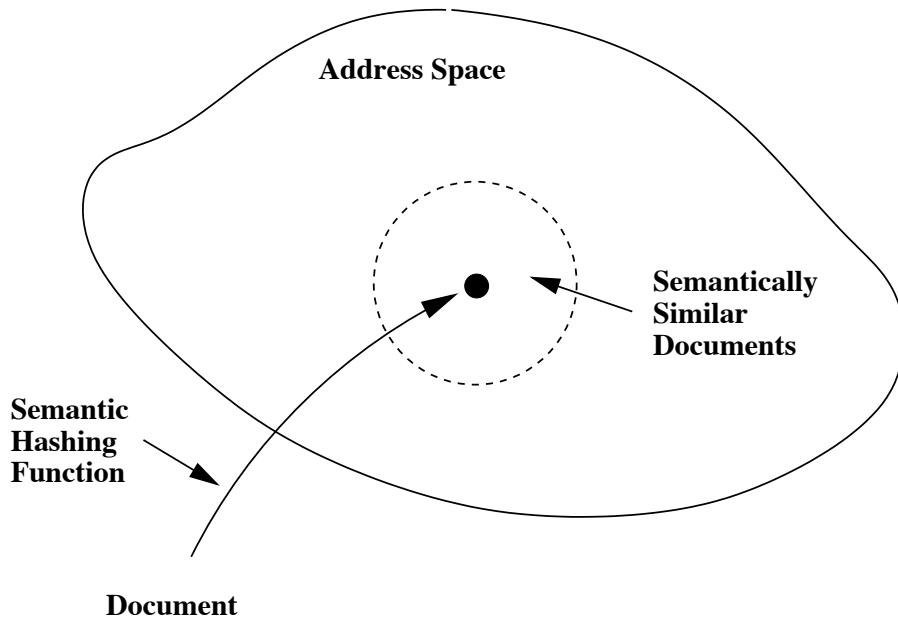
Reuters Dataset



Reuters dataset: 804,414 newswire stories.

Deep generative model significantly outperforms LSA and LDA topic models

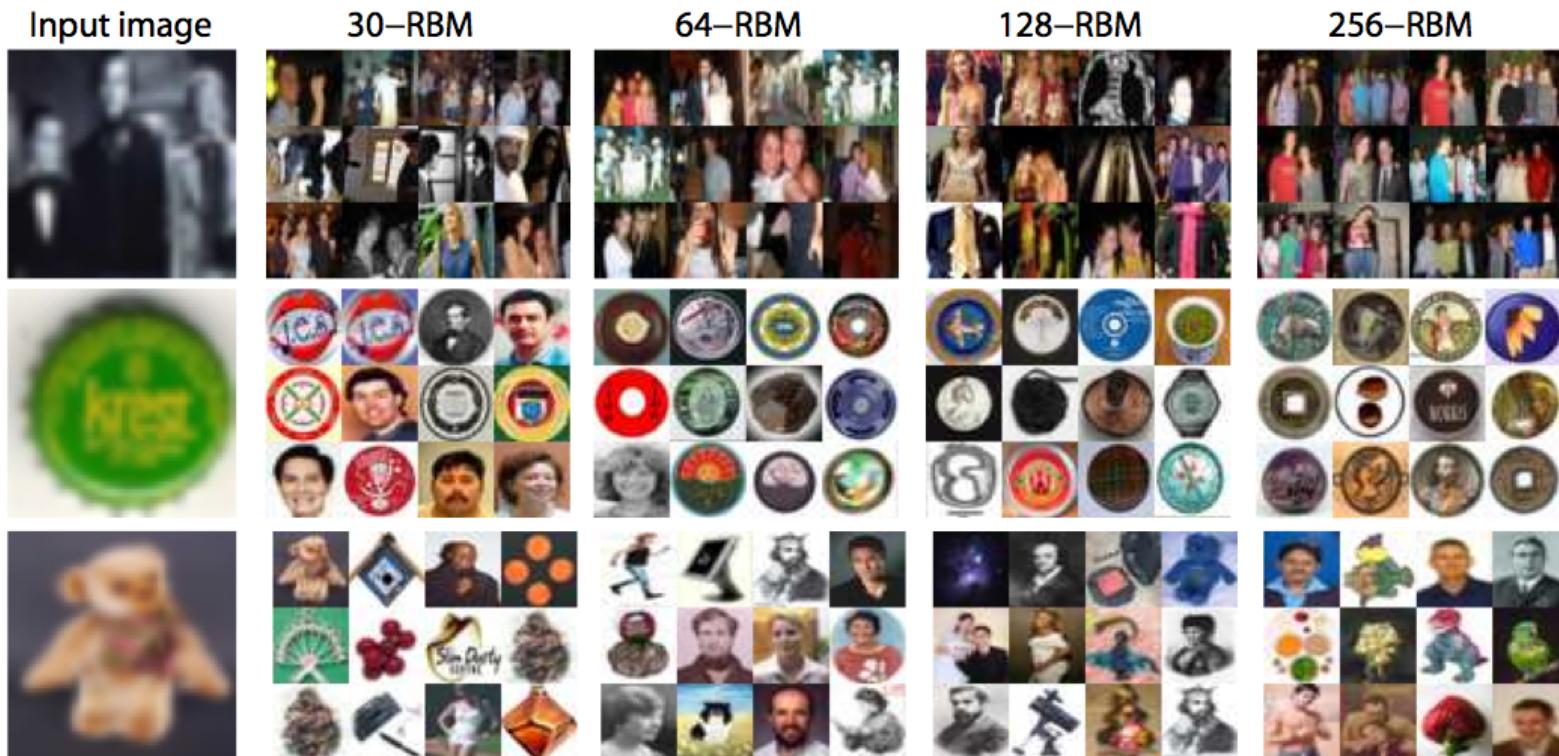
Semantic Hashing



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

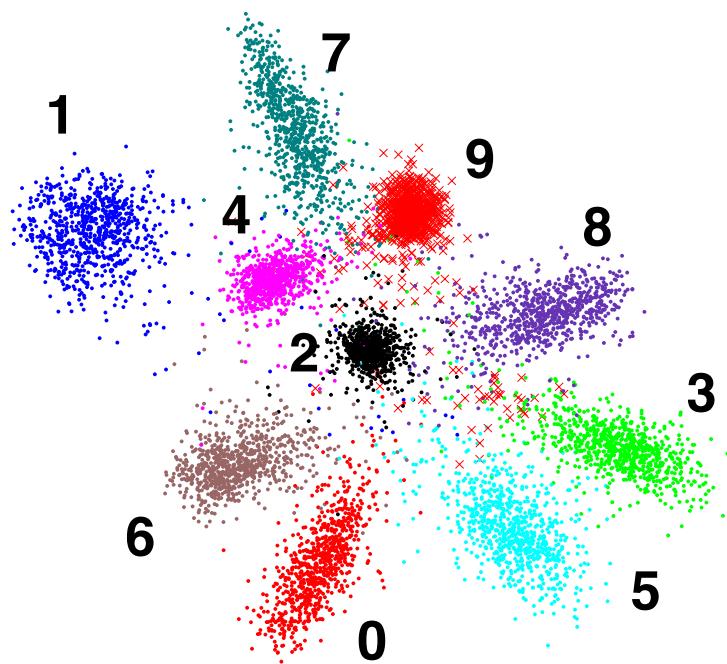
Searching Large Image Database using Binary Codes

- Map images into binary codes for fast retrieval.



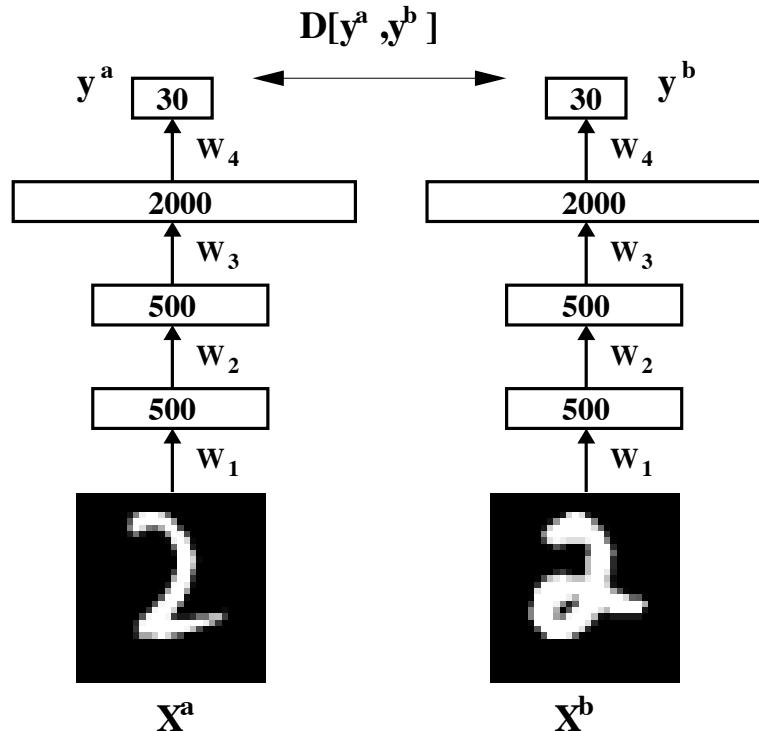
- Small Codes, Torralba, Fergus, Weiss, CVPR 2008
- Spectral Hashing, Y. Weiss, A. Torralba, R. Fergus, NIPS 2008
- Kulis and Darrell, NIPS 2009, Gong and Lazebnik, CVPR 2011
- Norouzi and Fleet, ICML 2011,

Learning Similarity Measures



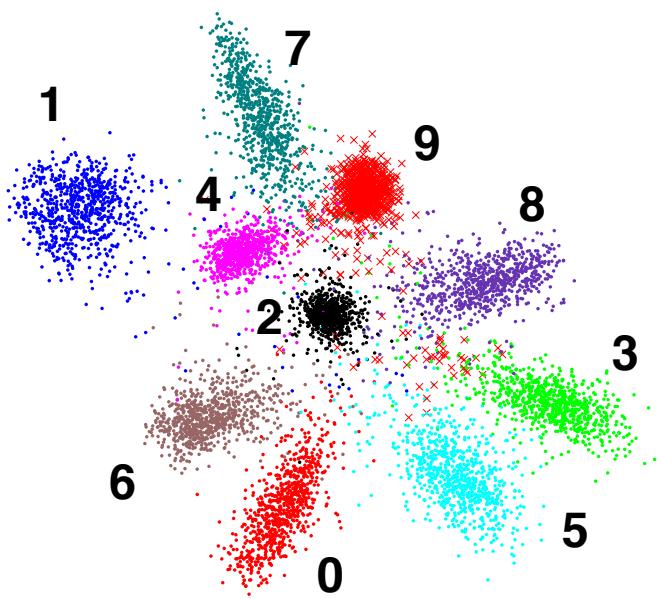
Related to Siamese Networks of LeCun.

Maximize the Agreement

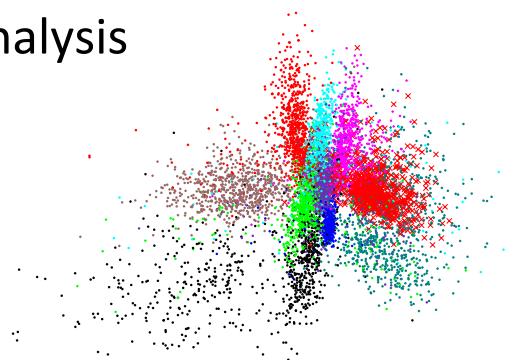


- Learn a nonlinear transformation of the input space.
- Optimize to make KNN perform well in the low-dimensional feature space

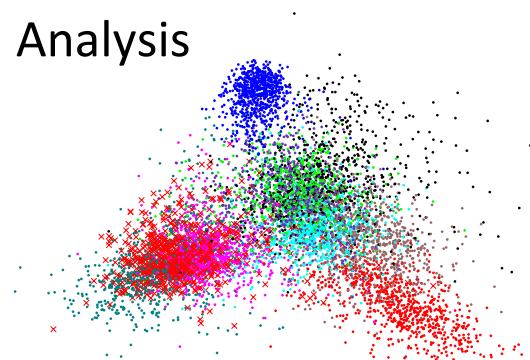
Learning Similarity Measures



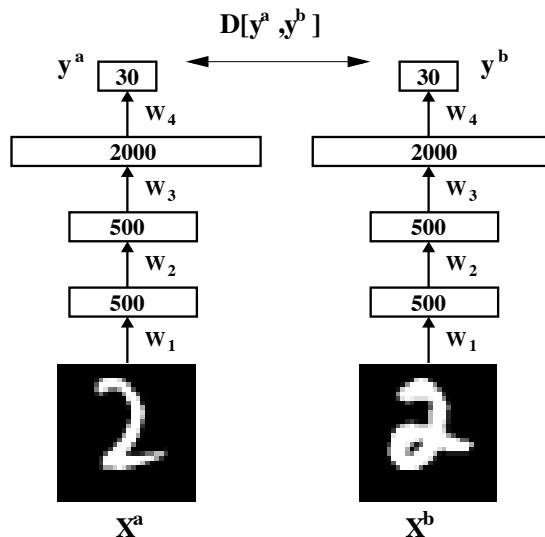
Neighborhood Component Analysis



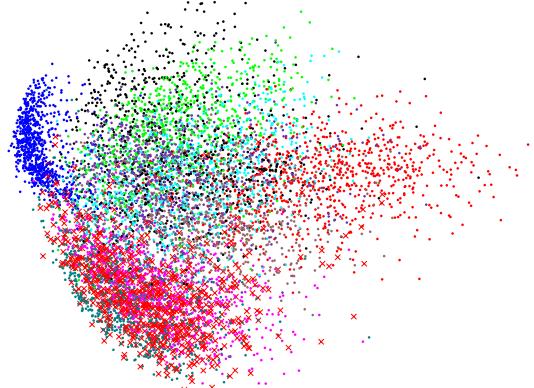
Linear discriminant Analysis



Learning Similarity Metric



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''.