THE UNIVERSITY OF BRITISH COLUMBIA

Topics in AI (CPSC 532S):
Multimodal Learning with Vision, Language and Sound

Lecture 13: Unsupervised Learning, Autoencoders

Unsupervised Learning

We have access to $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \cdots, \mathbf{x}_N\}$ but not $\{\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \cdots, \mathbf{y}_N\}$

*slide from Louis-Philippe Morency

Unsupervised Learning

We have access to $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \cdots, \mathbf{x}_N\}$ but not $\{\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \cdots, \mathbf{y}_N\}$

Why would we want to tackle such a task:

1. Extracting interesting information from data
   - Clustering
   - Discovering interesting trend
   - Data compression
2. Learn better representations

*slide from Louis-Philippe Morency

Unsupervised Representation Learning

Force our **representations** to better model input distribution

— Not just extracting features for classification

— Asking the model to be good at representing the data and not overfitting to a particular task (we get this with ImageNet, but maybe we can do better)

— Potentially allowing for better generalization

Use for **initialization of supervised task**, especially when we have a lot of unlabeled data and much less labeled examples

*slide from Louis-Philippe Morency

Restricted Boltzmann Machines (in one slide)

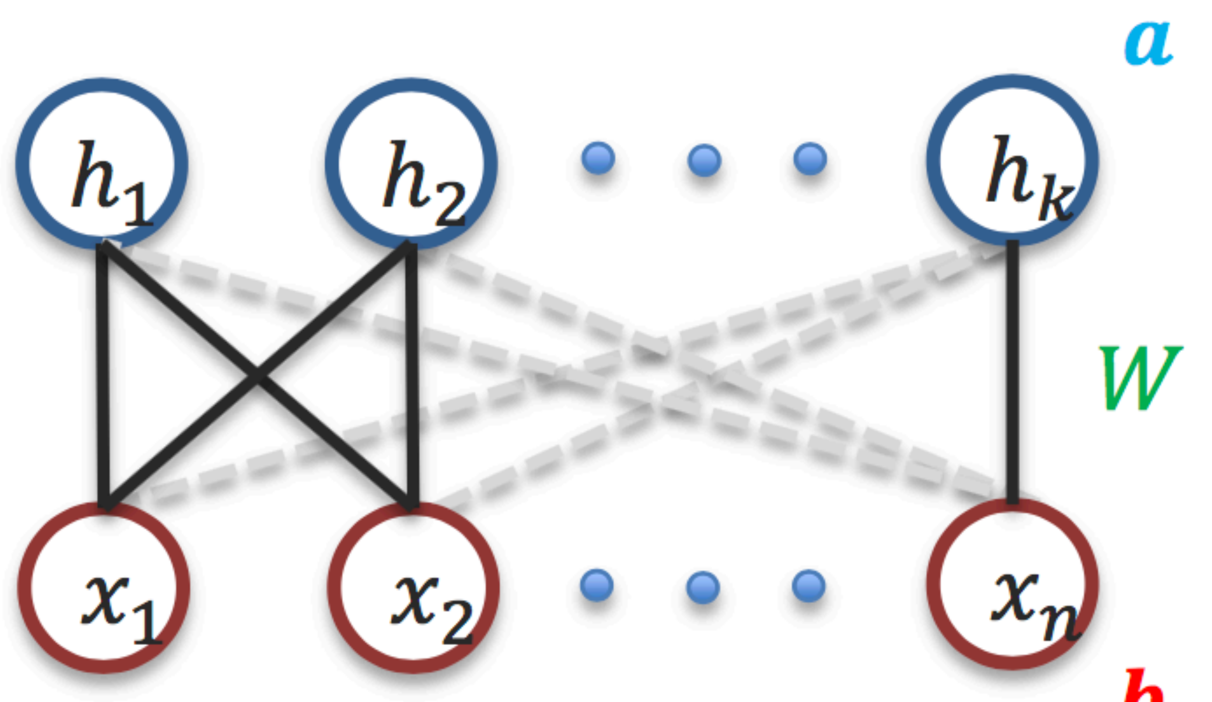
Model the **joint probability** of hidden state and observation

$$p(\boldsymbol{x}, \boldsymbol{h}; \theta) = \frac{\exp(-\mathrm{E}(\boldsymbol{x}, \boldsymbol{h}; \theta))}{Z}$$

$$Z = \sum_{\boldsymbol{x}} \sum_{\boldsymbol{h}} \exp(-\mathrm{E}(\boldsymbol{x}, \boldsymbol{h}; \theta))$$

$$\mathrm{E} = -\boldsymbol{x}W\boldsymbol{h} - \boldsymbol{b}^T\boldsymbol{x} - \boldsymbol{a}^T\boldsymbol{h}$$

$$\mathrm{E} = -\underbrace{\sum_i \sum_j w_{i,j} x_i h_j}_{\text{Interaction term}} - \underbrace{\sum_i b_i x_i - \sum_j a_j h_j}_{\text{Bias terms}}$$



Objective, maximize likelihood of the data

*slide from Louis-Philippe Morency

Autoencoders

*slide from Louis-Philippe Morency

Autoencoders

Self (i.e. self-encoding)

*slide from Louis-Philippe Morency

Autoencoders

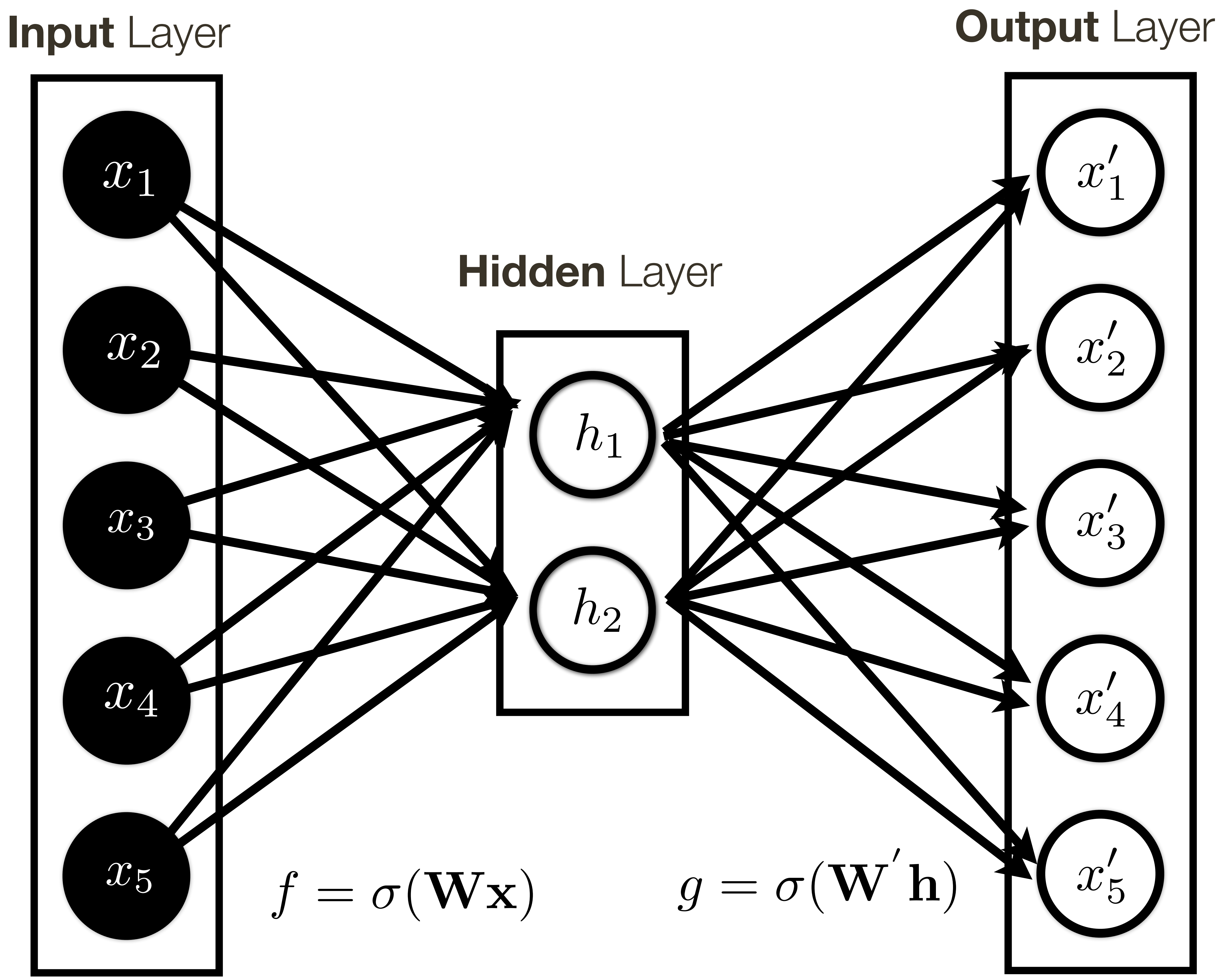
Self (i.e. self-encoding)

— Feed forward network intended to reproduce the input

— Encoder/Decoder architecture

Encoder: $f = \sigma(\mathbf{W}\mathbf{x})$

Decoder: $g = \sigma(\mathbf{W}^{'}\mathbf{h})$



*slide from Louis-Philippe Morency

Autoencoders

Self (i.e. self-encoding)

— Feed forward network intended to reproduce the input

— Encoder/Decoder architecture

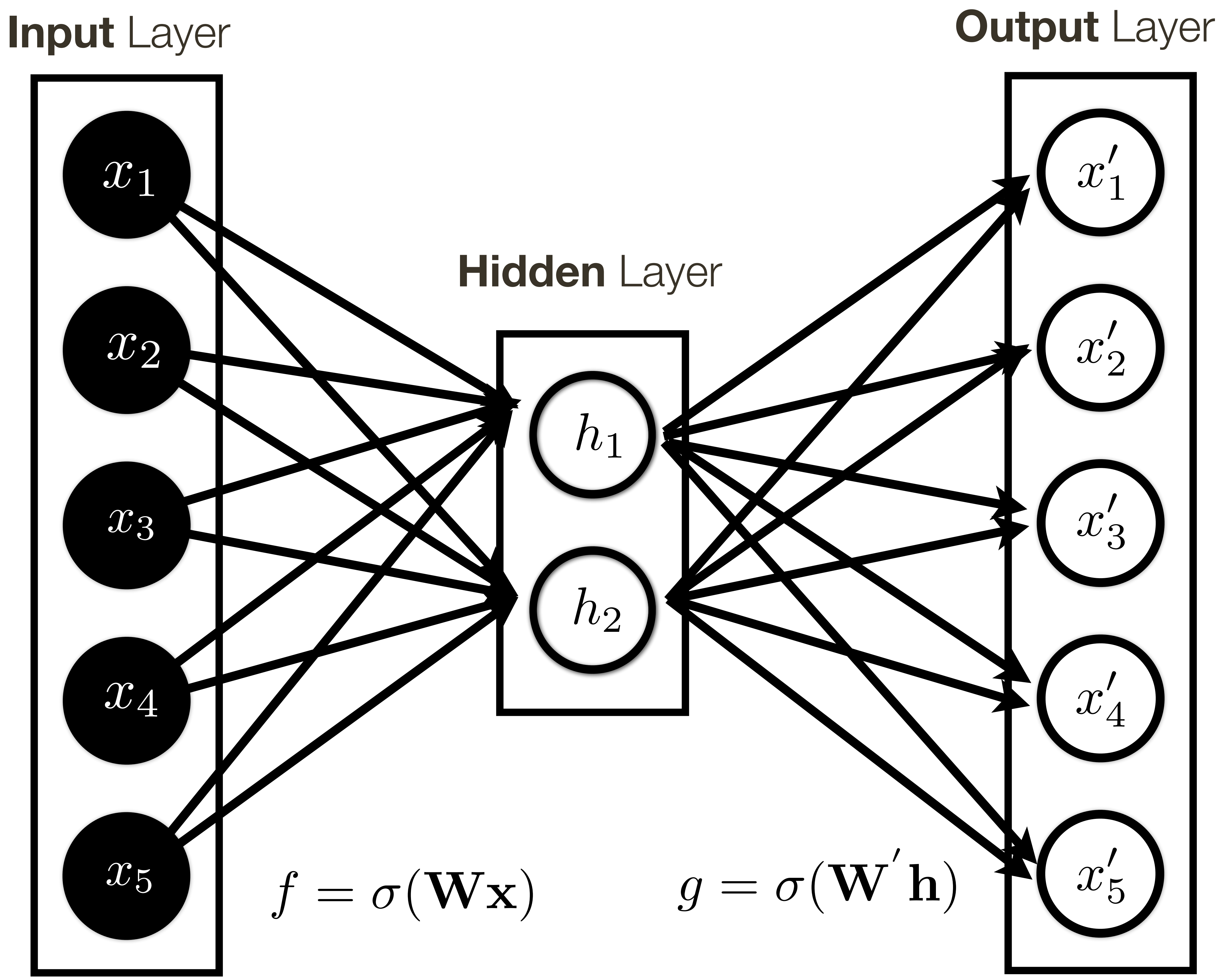
Encoder: $f = \sigma(\mathbf{W}\mathbf{x})$

Decoder: $g = \sigma(\mathbf{W}^{'}\mathbf{h})$

— Score function

$$\mathbf{x}' = f(g(\mathbf{x}))$$

$$\mathcal{L}(\mathbf{x}', \mathbf{x})$$



*slide from Louis-Philippe Morency

Autoencoders

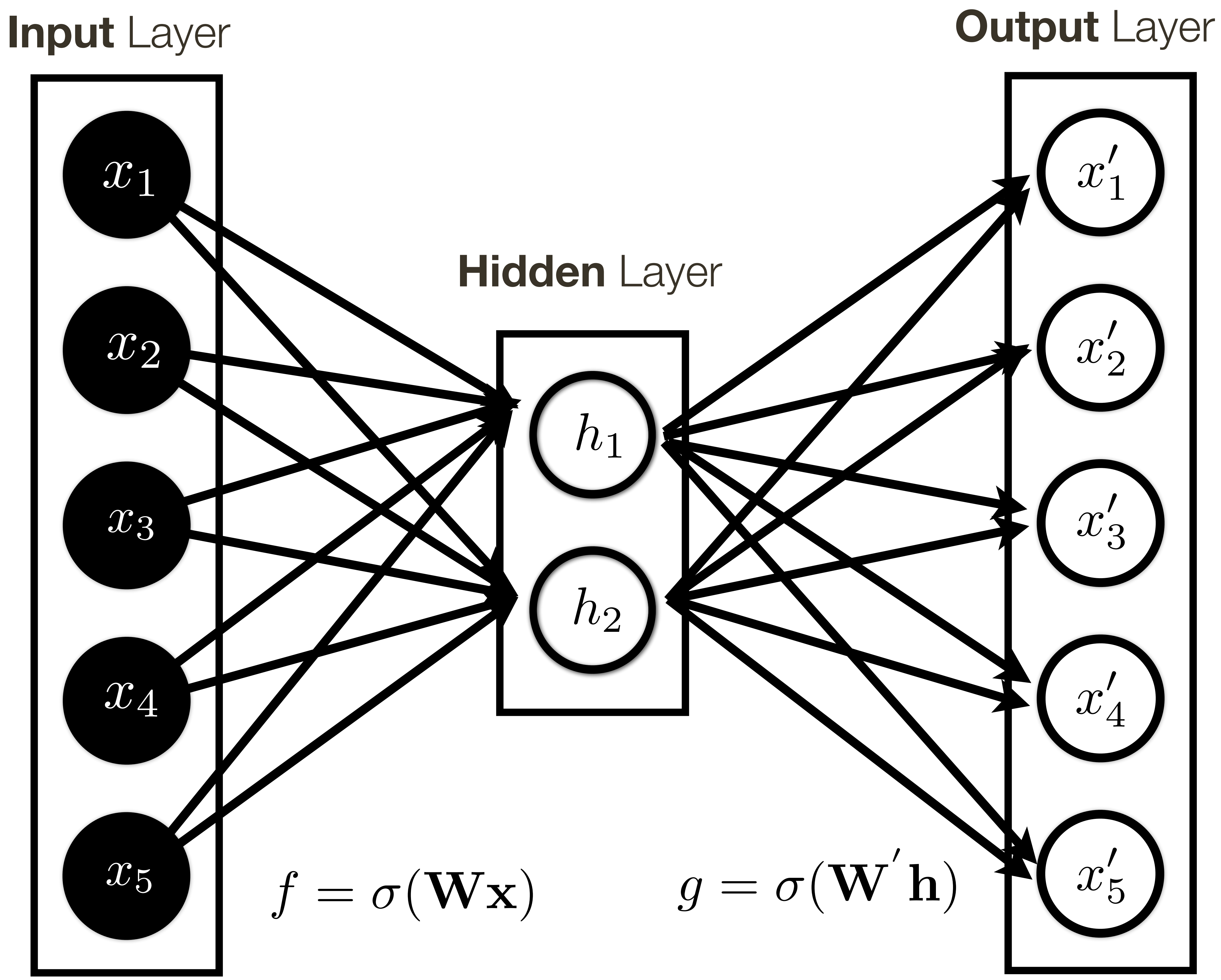
A standard neural network architecture (linear layer followed by non-linearity)

— Activation depends on type of data

(e.g., sigmoid for binary; linear for real valued)

— Often use tied weights

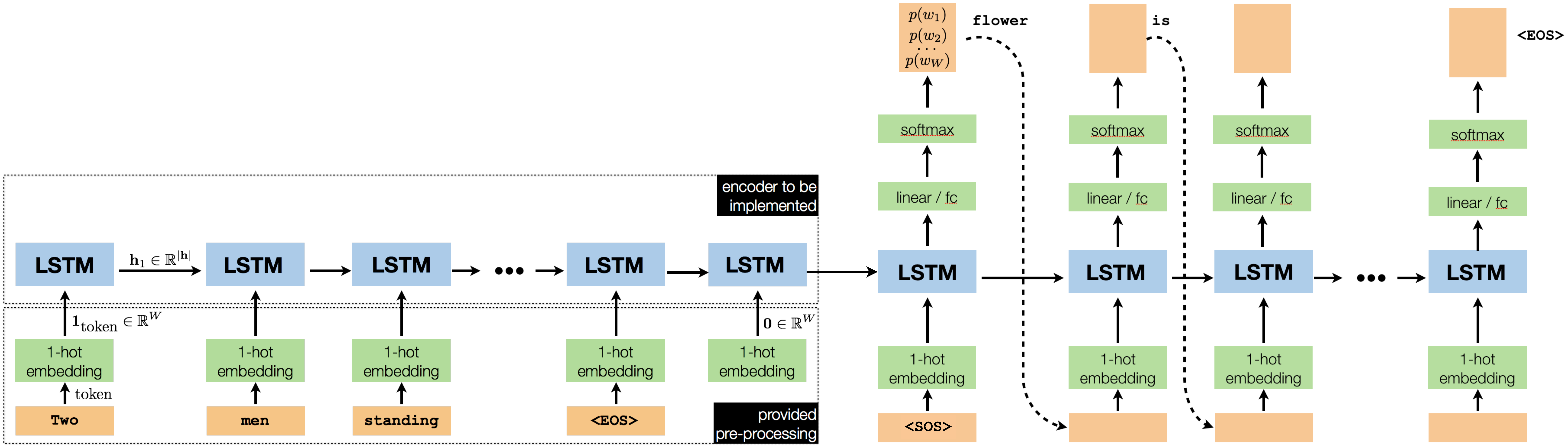
$$\mathbf{W}' = \mathbf{W}$$



*slide from Louis-Philippe Morency

Autoencoders

Assignment 3 can be interpreted as a language autoencoder

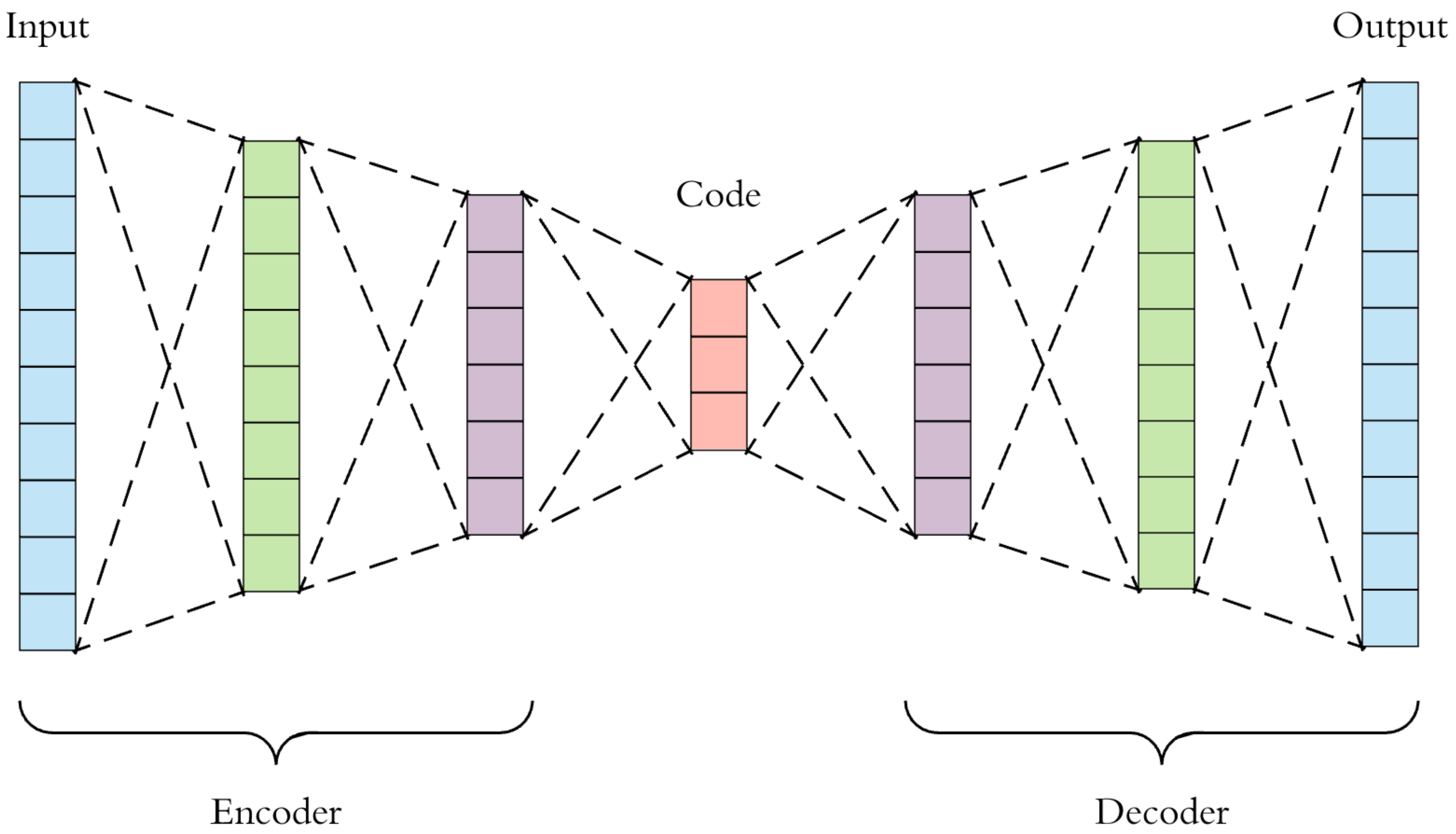


Autoencoders: Hidden Layer Dimensionality

Smaller than the input

— Will compress the data, reconstruction of the data far from the training distribution will be difficult

— Linear-linear encoder-decoder with Euclidian loss is actually equivalent to PCA (under certain data normalization)



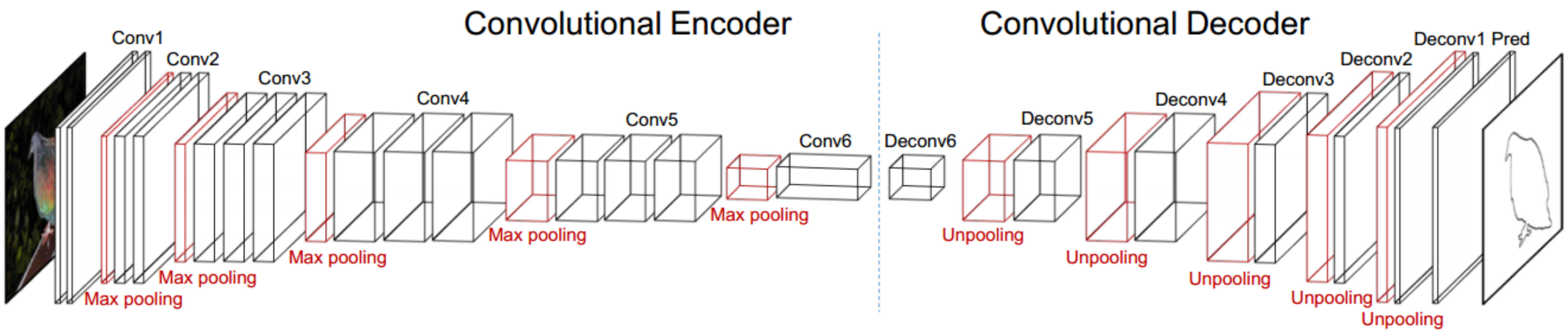
*slide from Louis-Philippe Morency

Autoencoders: Hidden Layer Dimensionality

Smaller than the input

— Will compress the data, reconstruction of the data far from the training distribution will be difficult

— Linear-linear encoder-decoder with Euclidian loss is actually equivalent to PCA (under certain data normalization)



*slide from Louis-Philippe Morency

Autoencoders: Hidden Layer Dimensionality

Smaller than the input

— Will compress the data, reconstruction of the data far from the training distribution will be difficult

— Linear-linear encoder-decoder with Euclidian loss is actually equivalent to PCA (under certain data normalization)

Side note, this is useful for **anomaly detection**

*slide from Louis-Philippe Morency

Autoencoders: Hidden Layer Dimensionality

Smaller than the input

— Will compress the data, reconstruction of the data far from the training distribution will be difficult

— Linear-linear encoder-decoder with Euclidian loss is actually equivalent to PCA (under certain data normalization)

Larger than the input

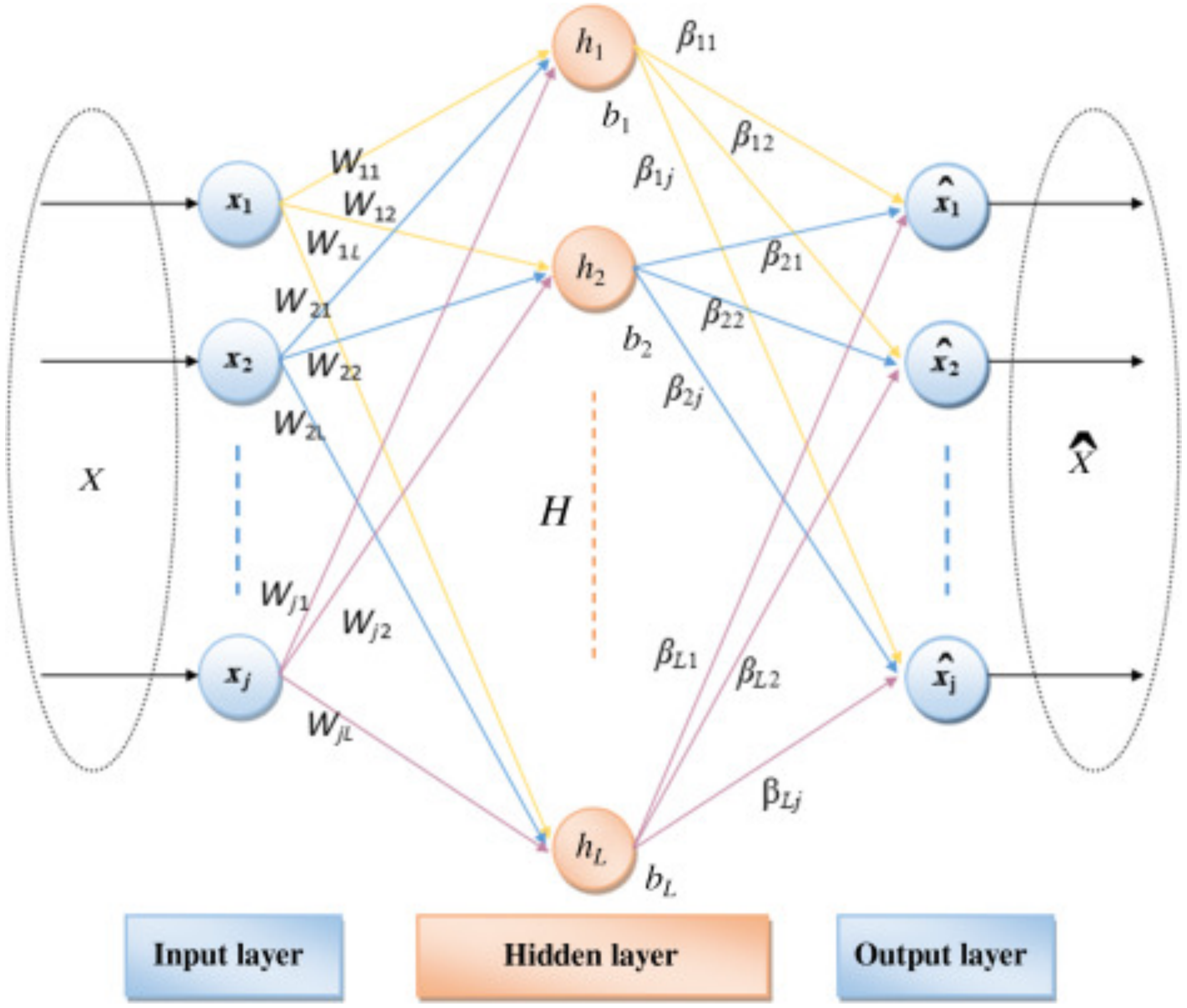
— No compression needed

— Can trivially learn to just copy, no structure is learned (unless you regularize)

— Does not encourage learning of meaningful features (unless you regularize)

*slide from Louis-Philippe Morency

Autoencoders: Hidden Layer Dimensionality



Larger than the input

— No compression needed

— Can trivially learn to just copy, no structure is learned (unless you regularize)

— Does not encourage learning of meaningful features (unless you regularize)

*slide from Louis-Philippe Morency

Autoencoders

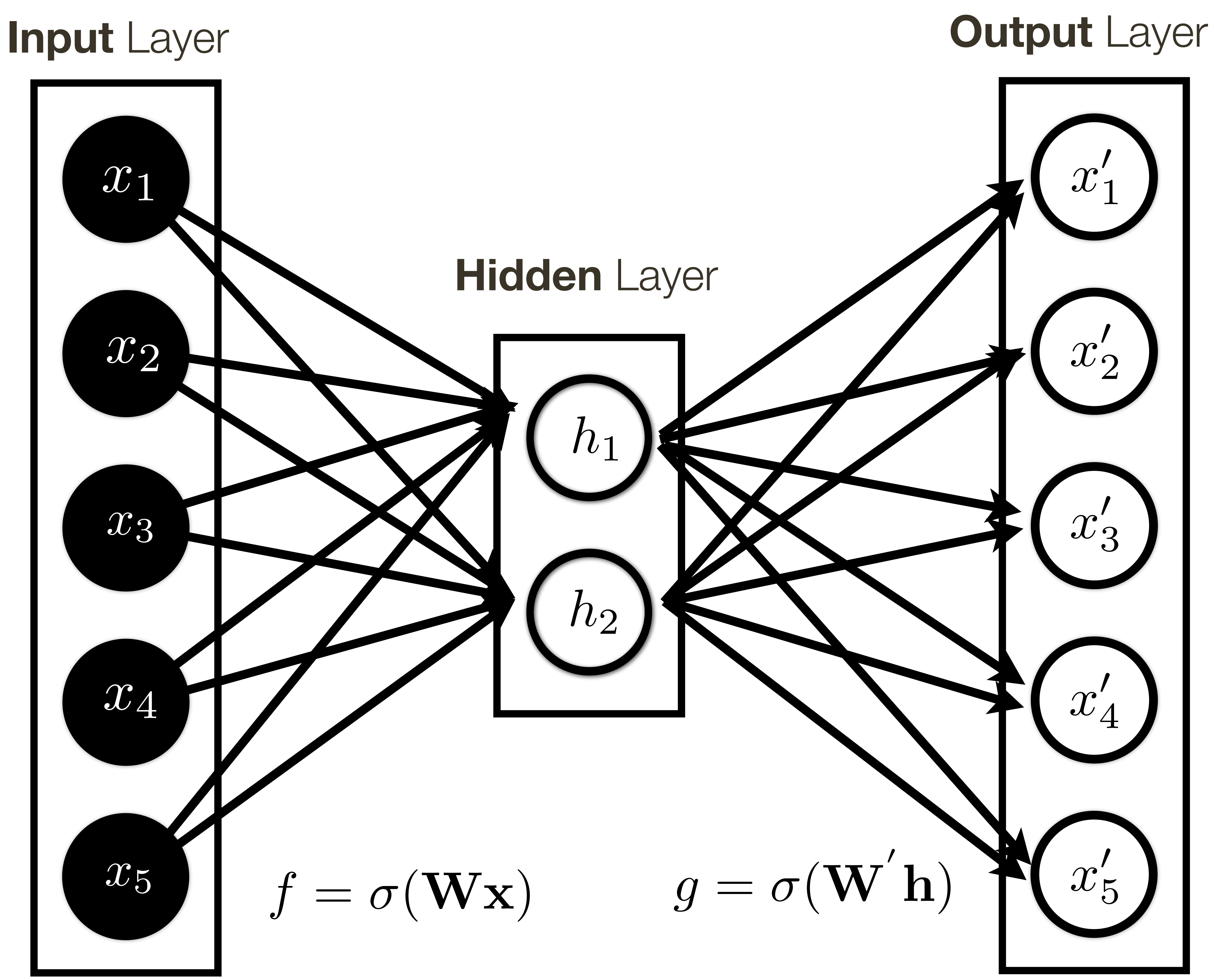
A standard neural network architecture (linear layer followed by non-linearity)

— Activation depends on type of data

(e.g., sigmoid for binary; linear for real valued)

— Often use tied weights

$$\mathbf{W}' = \mathbf{W}$$



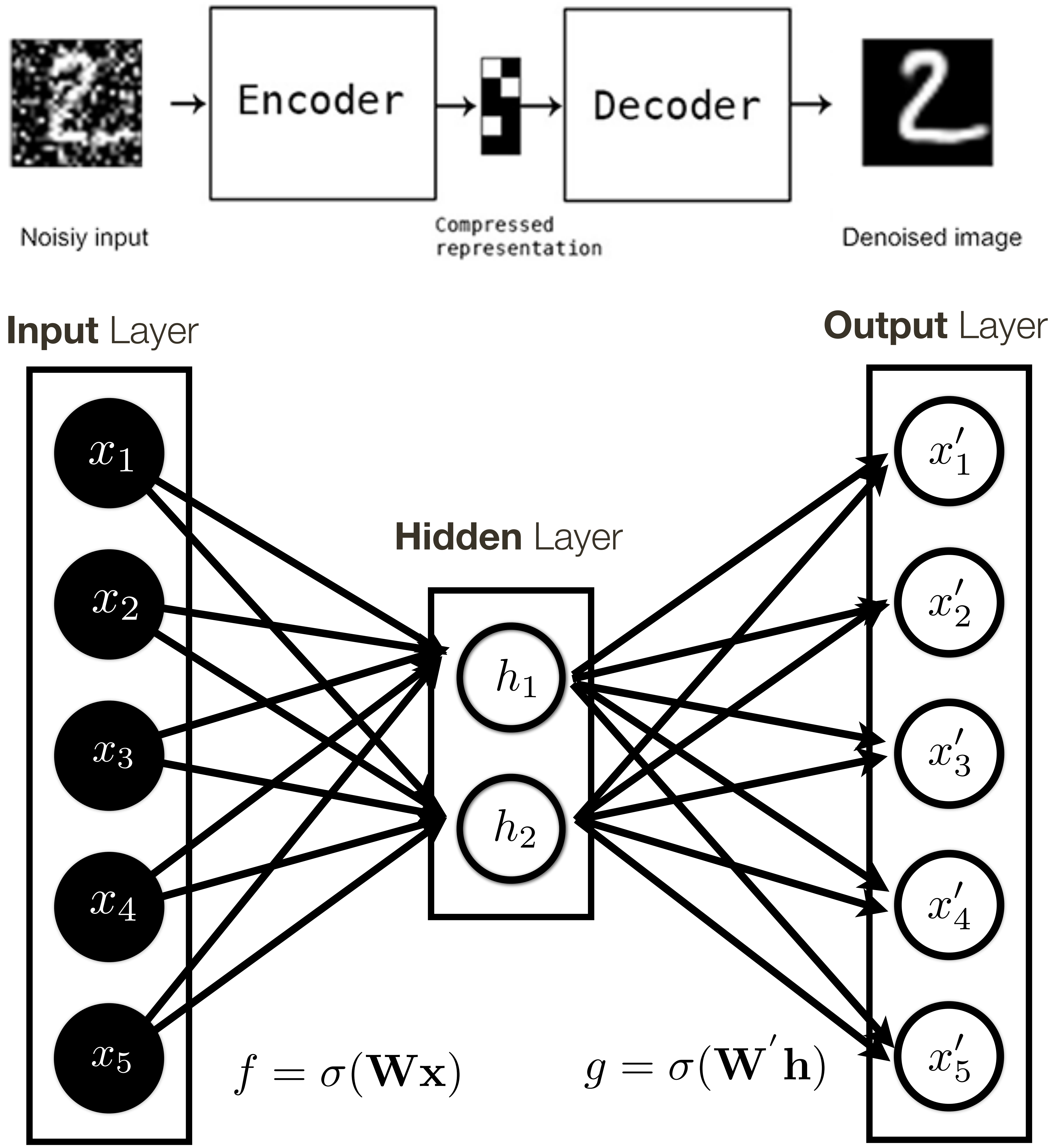
*slide from Louis-Philippe Morency

De-noising Autoencoder

Idea: add noise to input but learn to reconstruct the original

- Leads to better representations
- Prevents copying

Note: different noise is added during each epoch



*slide from Louis-Philippe Morency

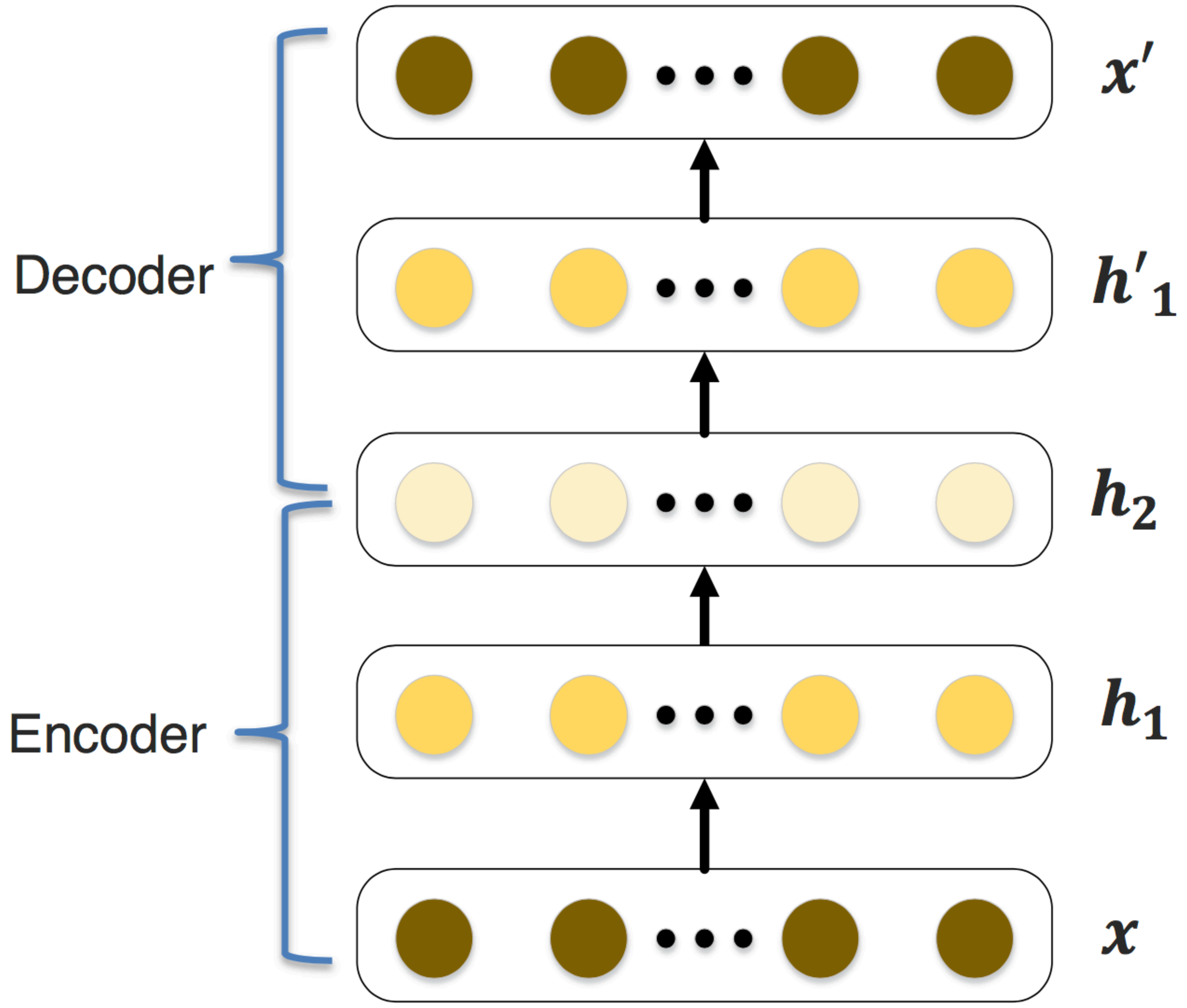
Stacked (deep) Autoencoders and Denoising Autoencoders

What **can we do** with them?

— Good for compression (better than PCA)

— Disregard the decoder and use the middle layer as a representation

— Fine-tune the autoencoder for a task



*slide from Louis-Philippe Morency

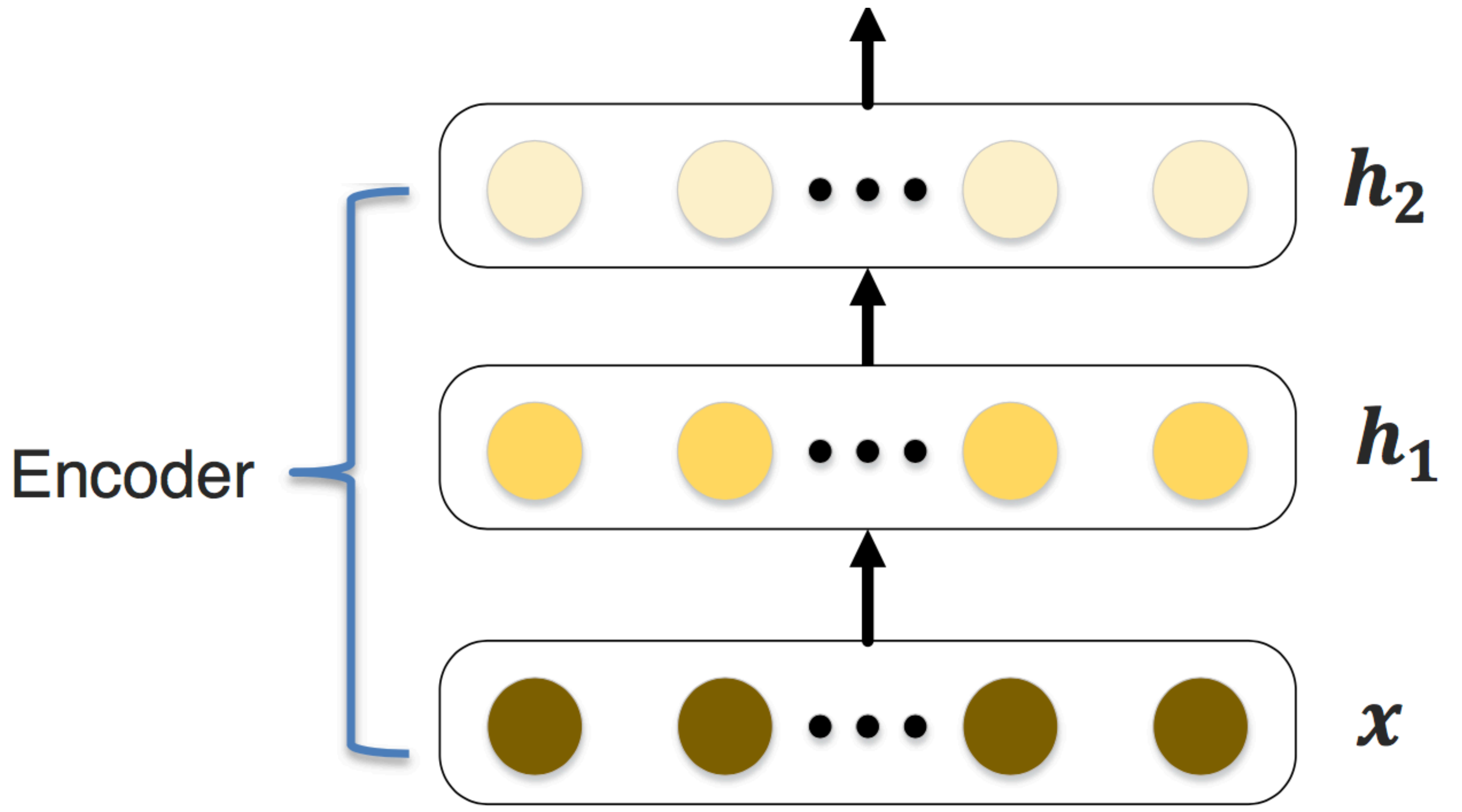
Stacked (deep) Autoencoders and Denoising Autoencoders

What **can we do** with them?

— Good for compression (better than PCA)

— Disregard the decoder and use the middle layer as a representation

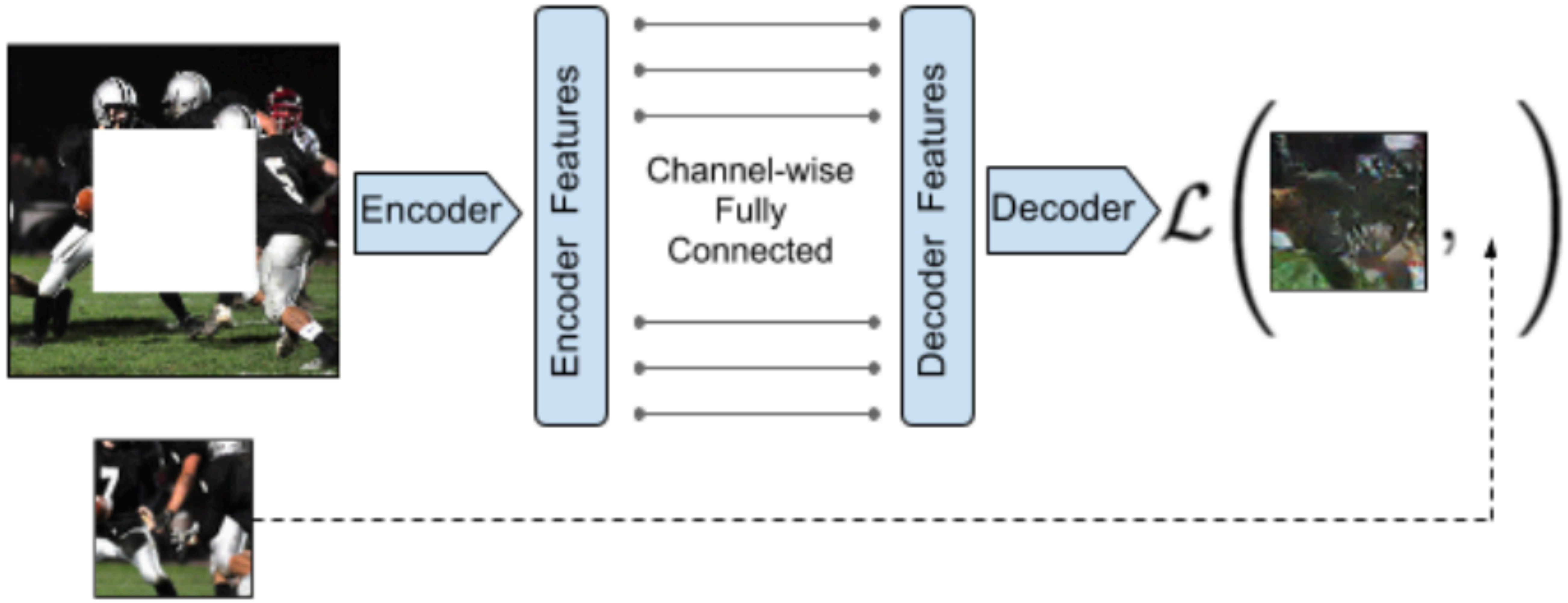
— **Fine-tune** the autoencoder for a task



*slide from Louis-Philippe Morency

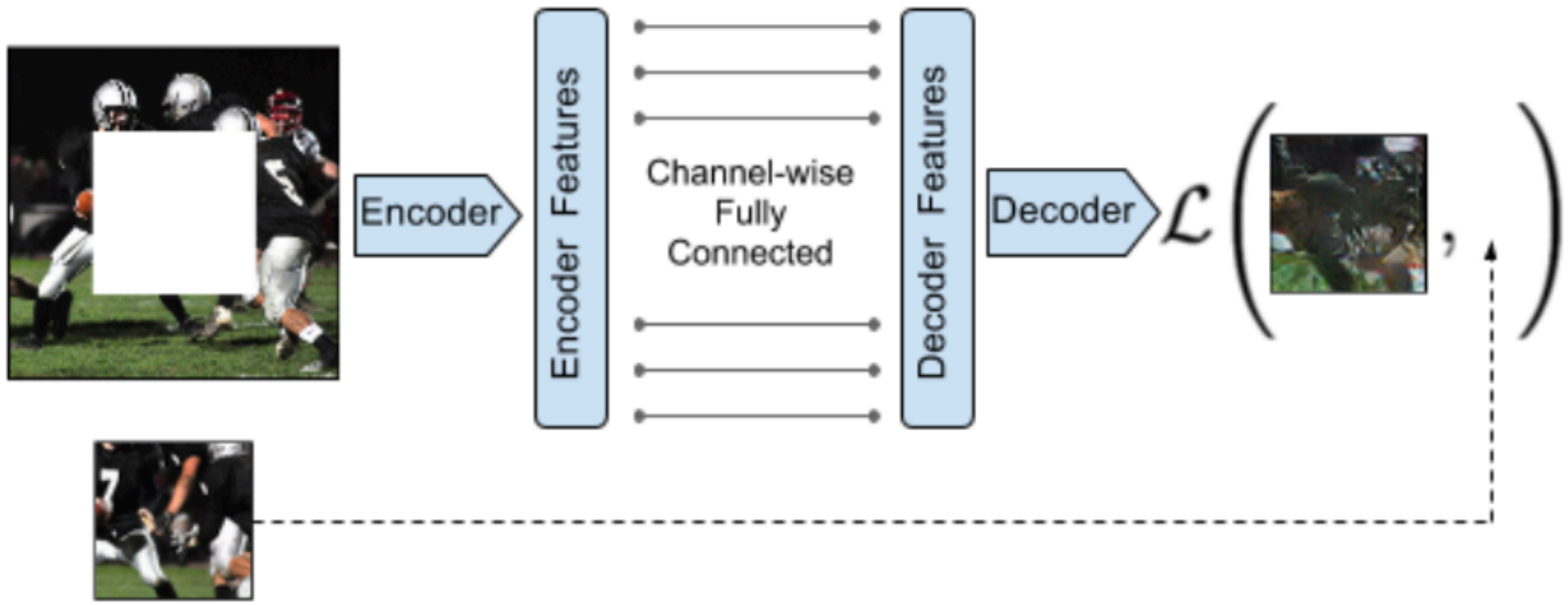
Context Encoders

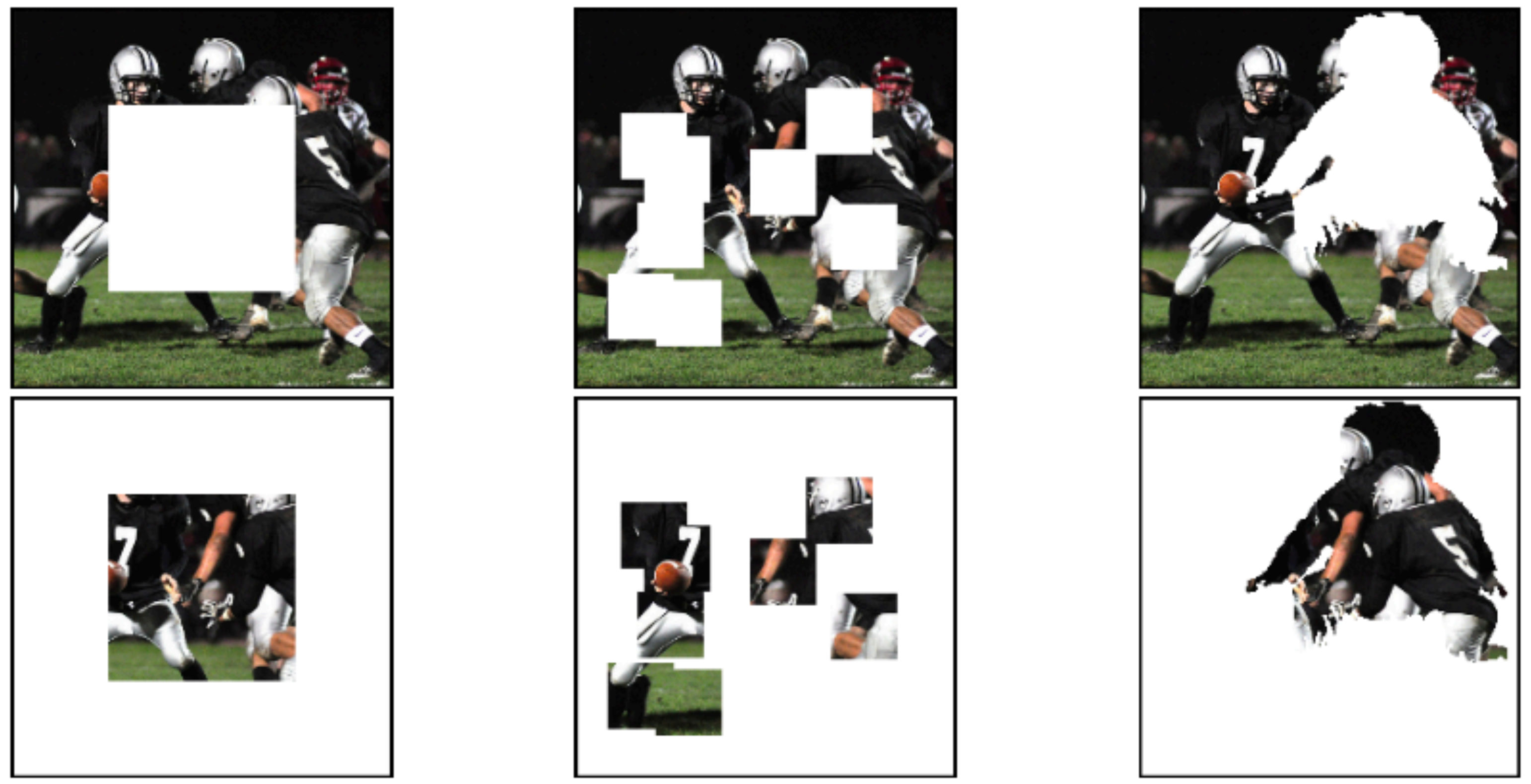
[Pathak et al., 2016]



Context Encoders

[Pathak et al., 2016]

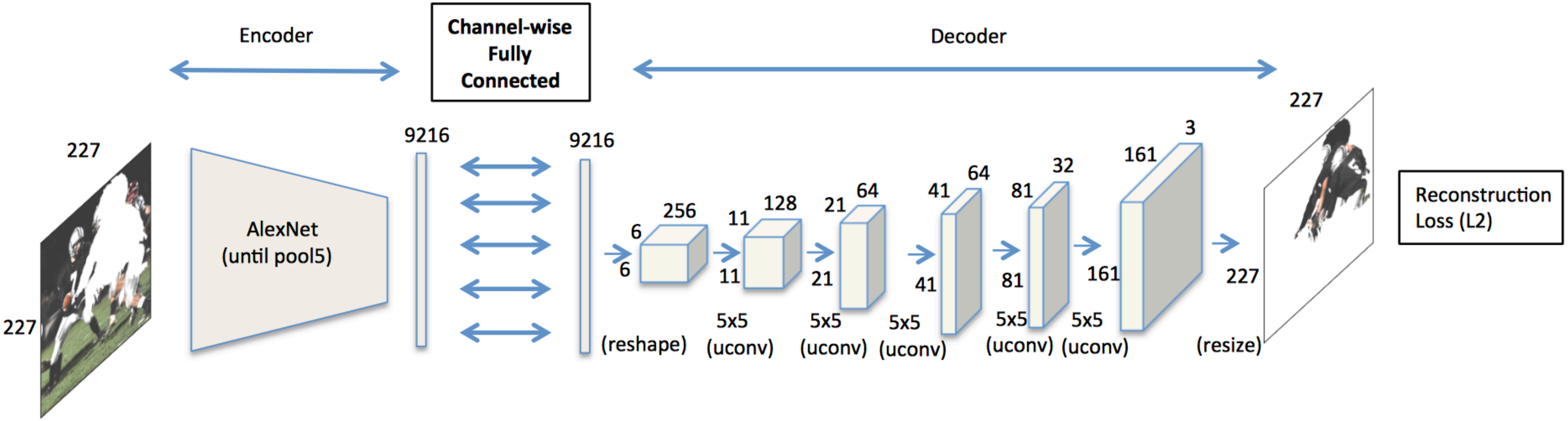




(a) Central region (b) Random block (c) Random region

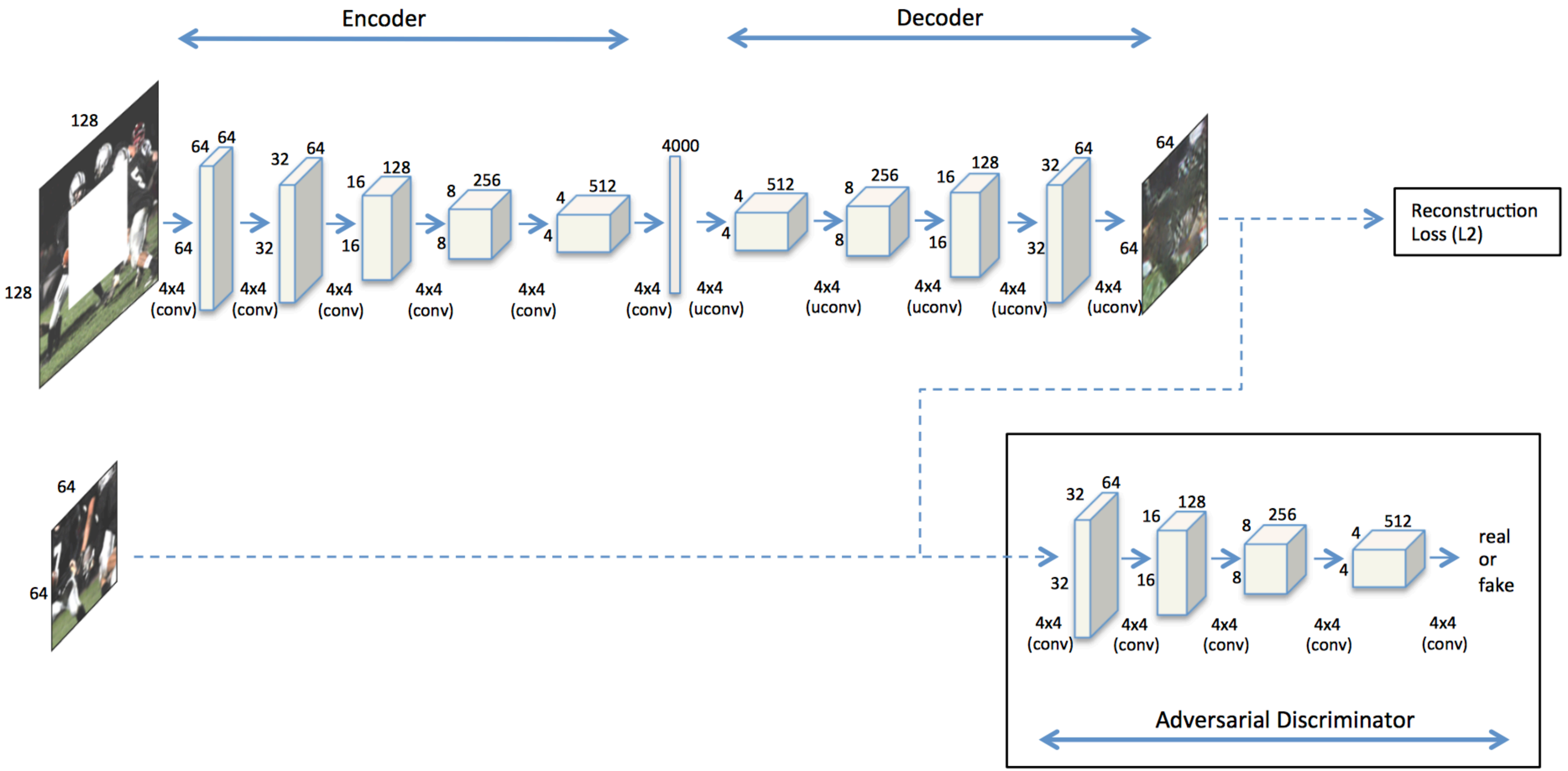
Context Encoders

[Pathak et al., 2016]



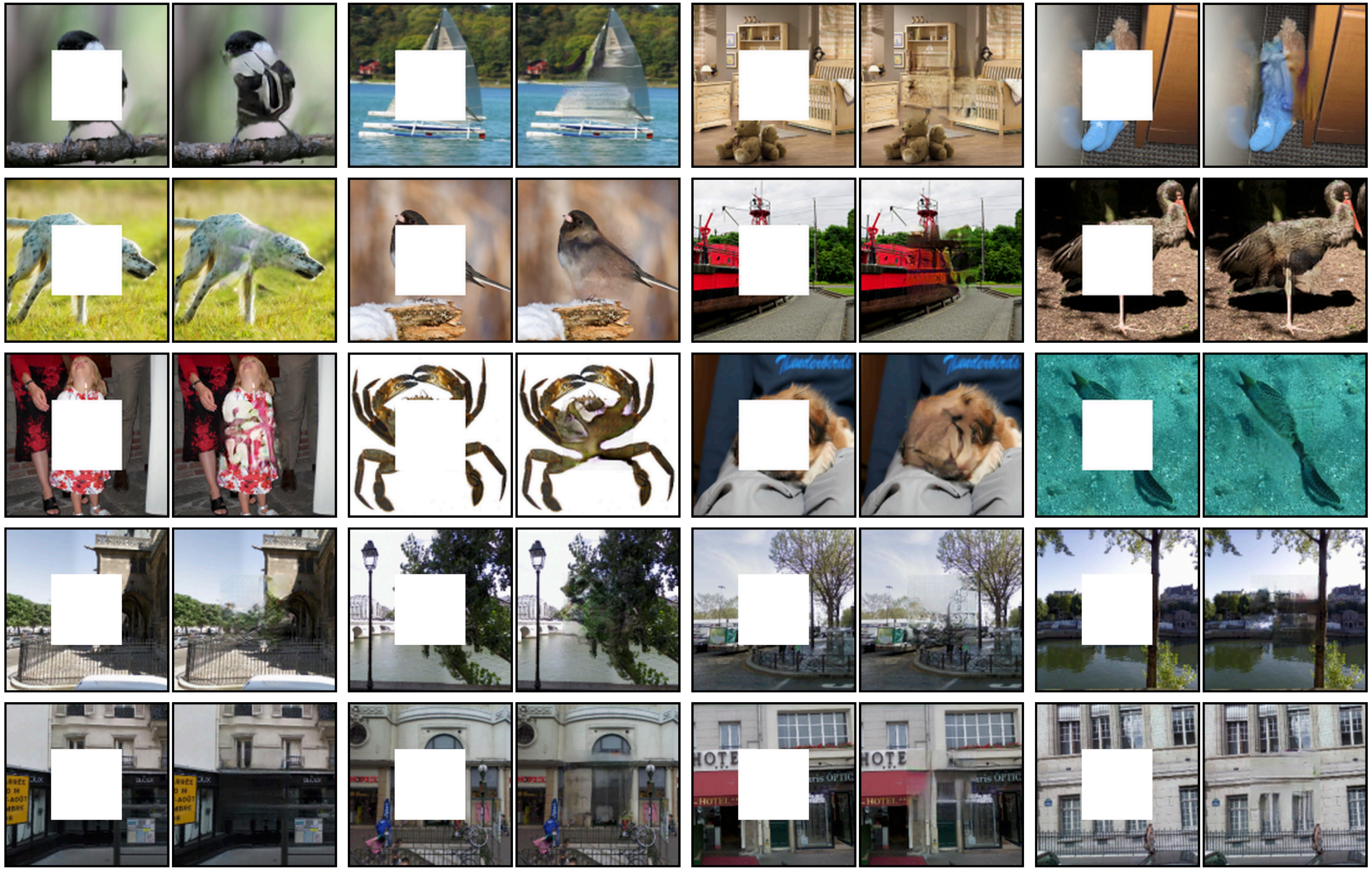
Context Encoders

[Pathak et al., 2016]



Context Encoders

[Pathak et al., 2016]



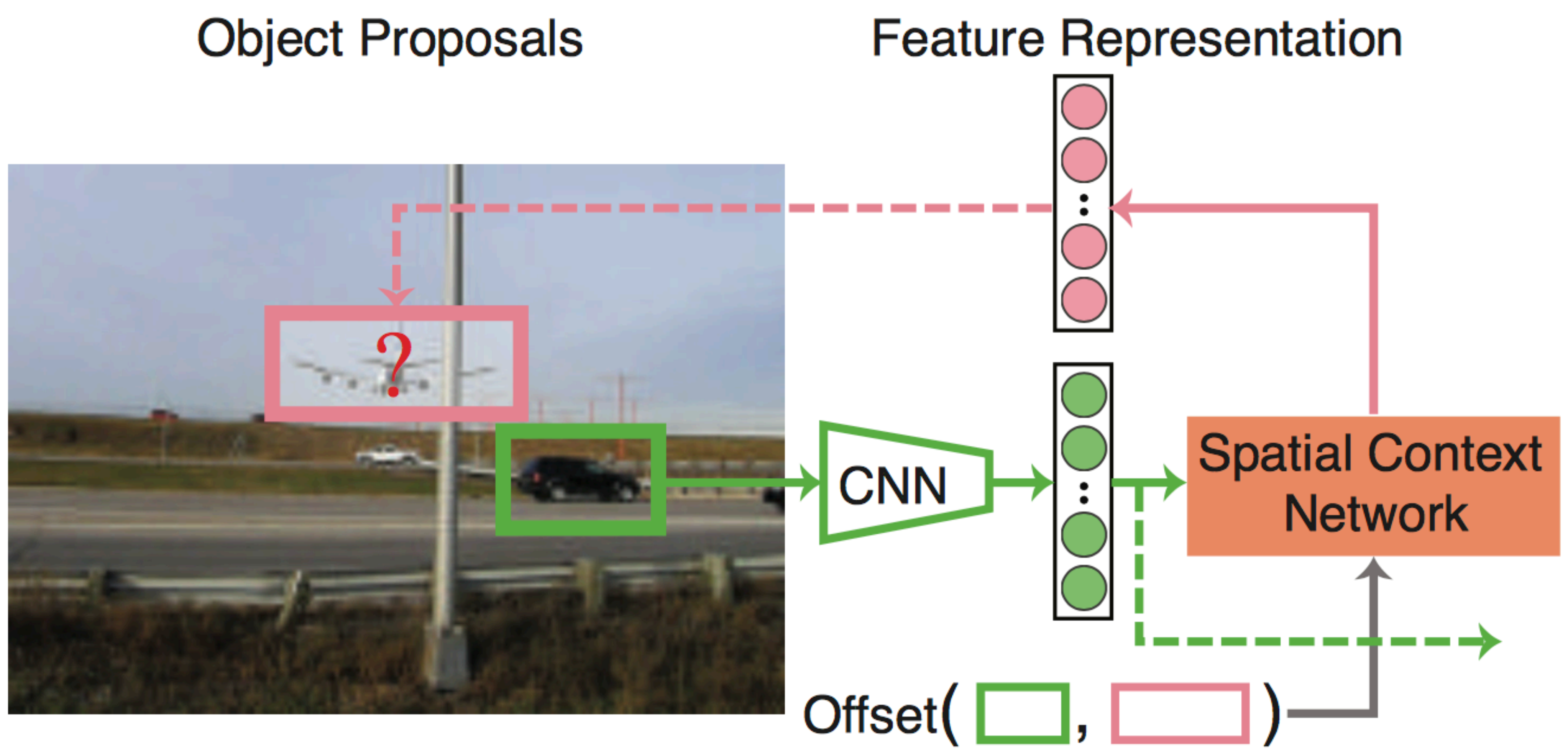
Context Encoders

[Pathak et al., 2016]

Pretraining Method	Supervision	Pretraining time	Classification	Detection	Segmentation
ImageNet [26]	1000 class labels	3 days	**78.2%**	**56.8%**	**48.0%**
Random Gaussian	initialization	$<$ 1 minute	53.3%	43.4%	19.8%
Autoencoder	-	14 hours	53.8%	41.9%	25.2%
Agrawal *et al.* [1]	egomotion	10 hours	52.9%	41.8%	-
Doersch *et al.* [7]	context	4 weeks	55.3%	**46.6%**	-
Wang *et al.* [39]	motion	1 week	**58.4%**	44.0%	-
Ours	context	14 hours	56.5%	44.5%	**29.7%**

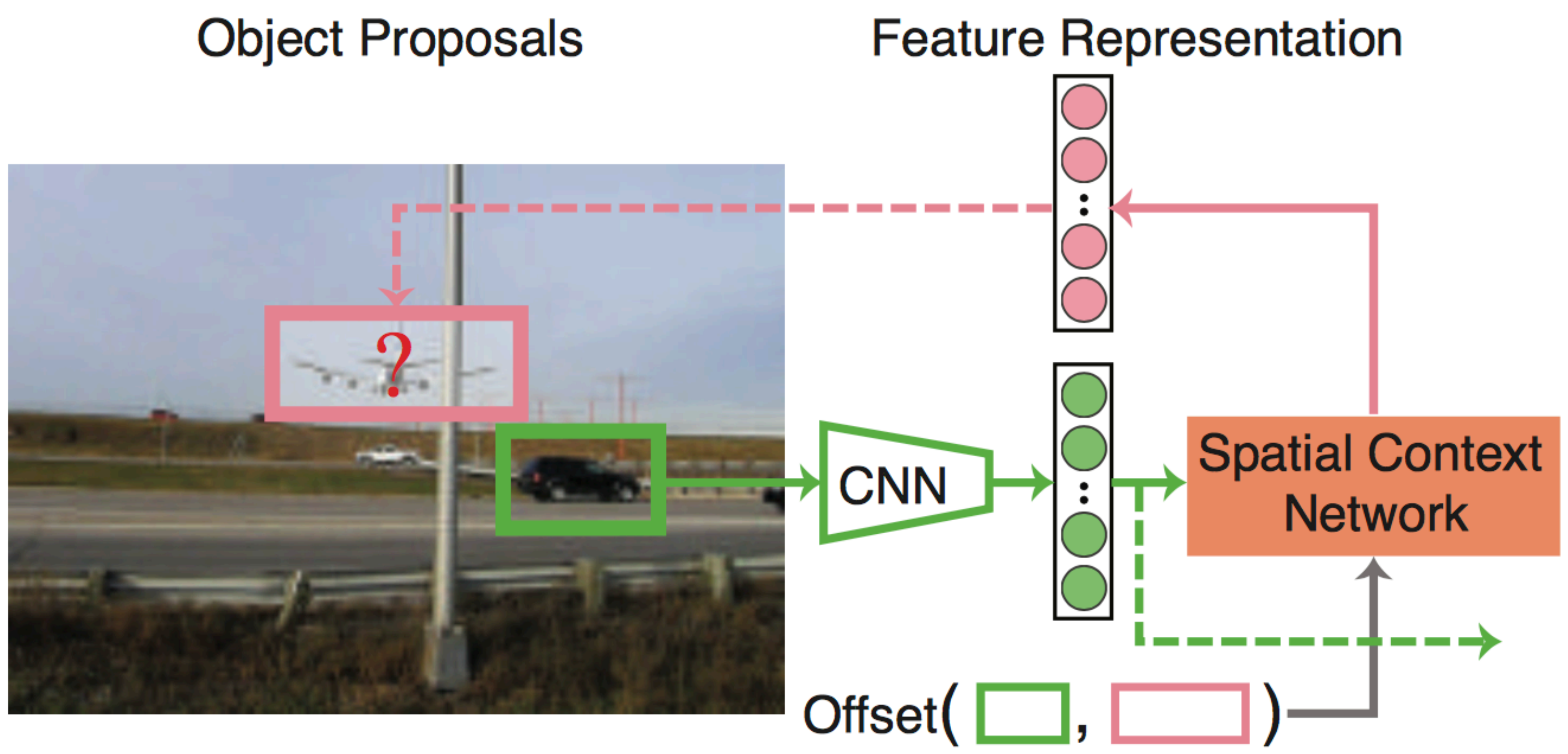
Spatial Context Networks

[Wu, Sigal, Davis, 2017]



Spatial Context Networks

[Wu, Sigal, Davis, 2017]

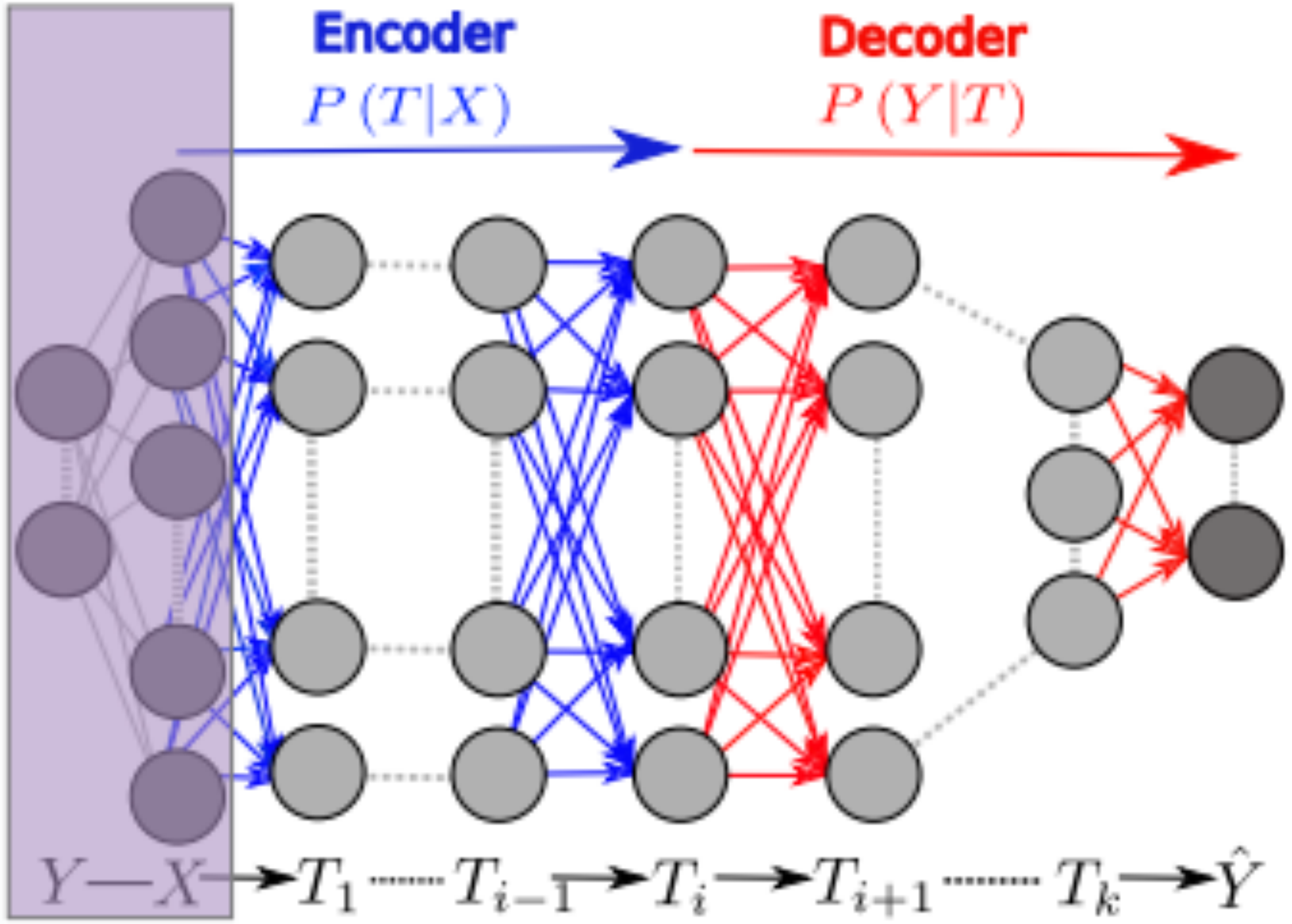


	Initialization	Supervision	Pretraining time	Classification	Detection
Random Gaussian	random	N/A	< 1 minute	53.3	43.4
Wang *et al.* [32]	random	motion	1 week	58.4	44.0
Doersch *et al.* [3]	random	context	4 weeks	55.3	46.6
*Doersch *et al.* [3]	1000 class labels	context	–	65.4	50.4
Pathak *et al.* [21]	random	context inpainting	14 hours	56.5	44.5
Zhang *et al.* [36]	random	color	–	65.6	46.9
ImageNet [21]	random	1000 class labels	3 days	78.2	56.8
*ImageNet	random	1000 class labels	3 days	76.9	58.7
SCN-EdgeBox	1000 class labels	context	10 hours	79.0	**59.4**

A **Little** Theory: **Information Bottleneck** [Tishbi et al., 1999]

Every layer could be treated as a random variable, then entire network is a Markov Chain

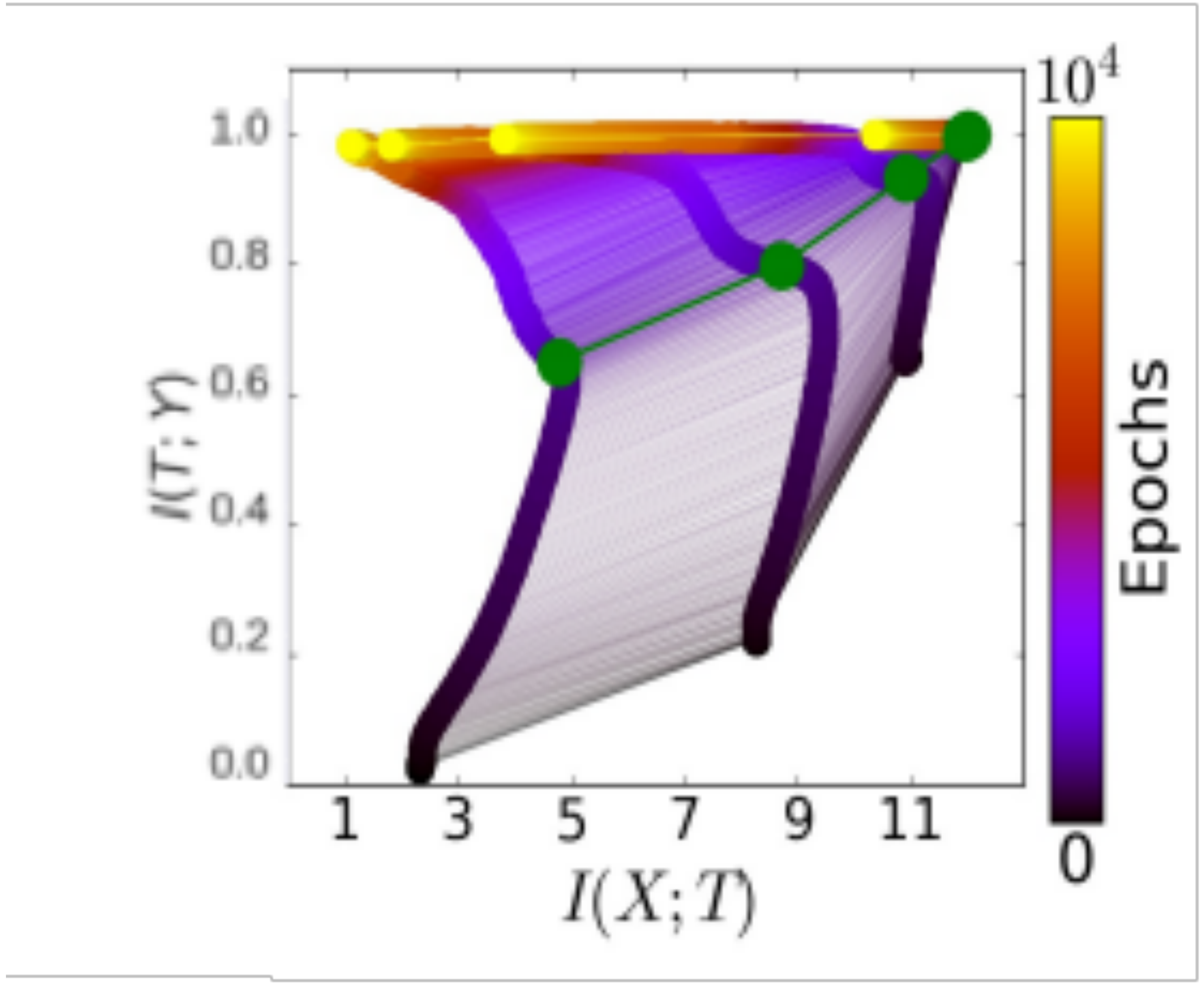
Data processing theorem: if the only connection between X and Y is through T, the information that Y gives about X cannot be bigger than the information that T gives about X.



$$I(X;Y) \leq I(T_1;Y) \leq I(T_2;Y) \leq \cdots \leq I(\hat{Y};Y)$$

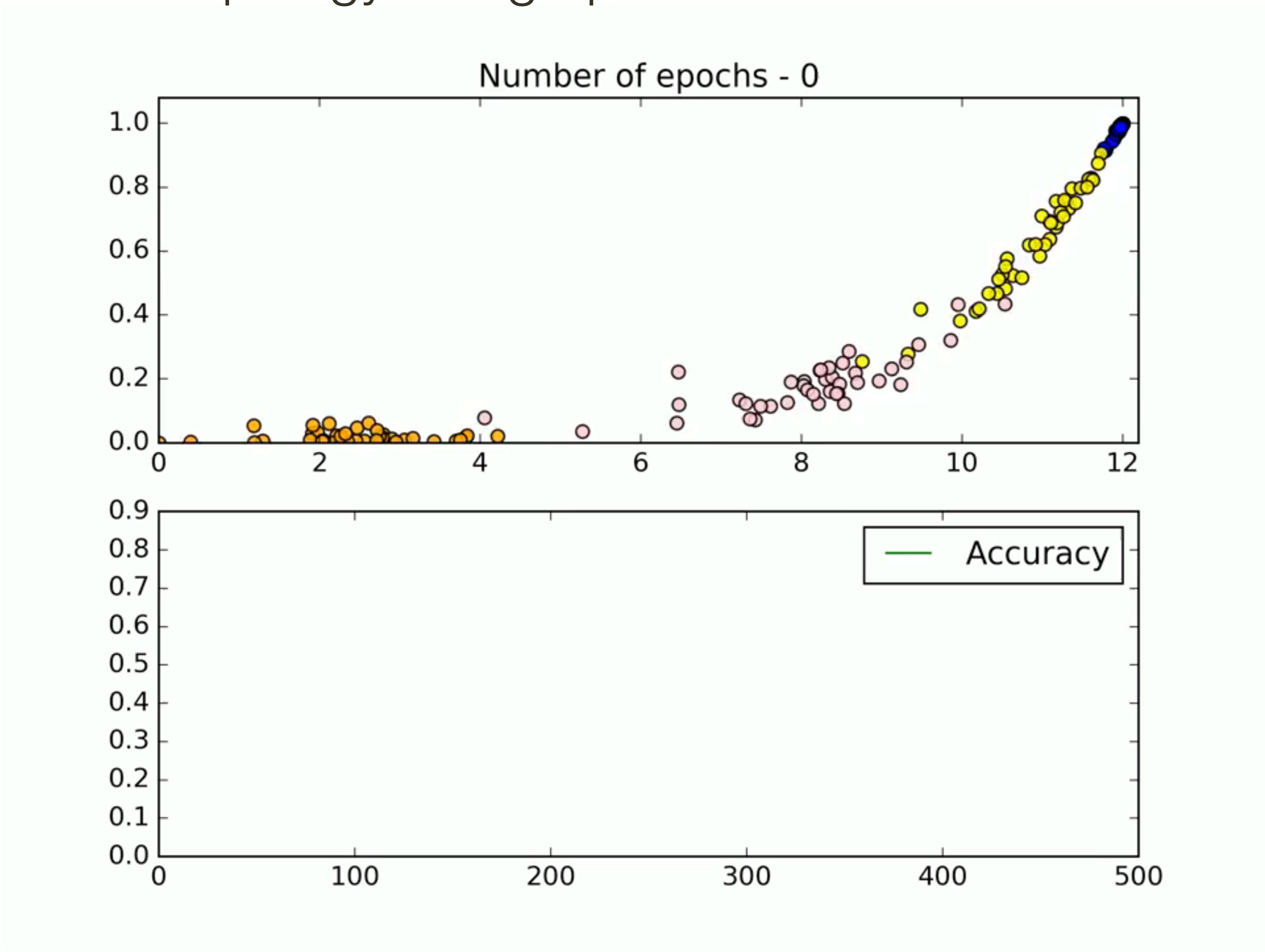
A **Little** Theory: **Information Bottleneck** [Tishbi et al., 1999]

Observation: In the information plane layers first increase the mutual information between themselves and the output and then reduce information between themselves and the input (which leads to “forgetting” of irrelevant inputs and ultimately generalization)



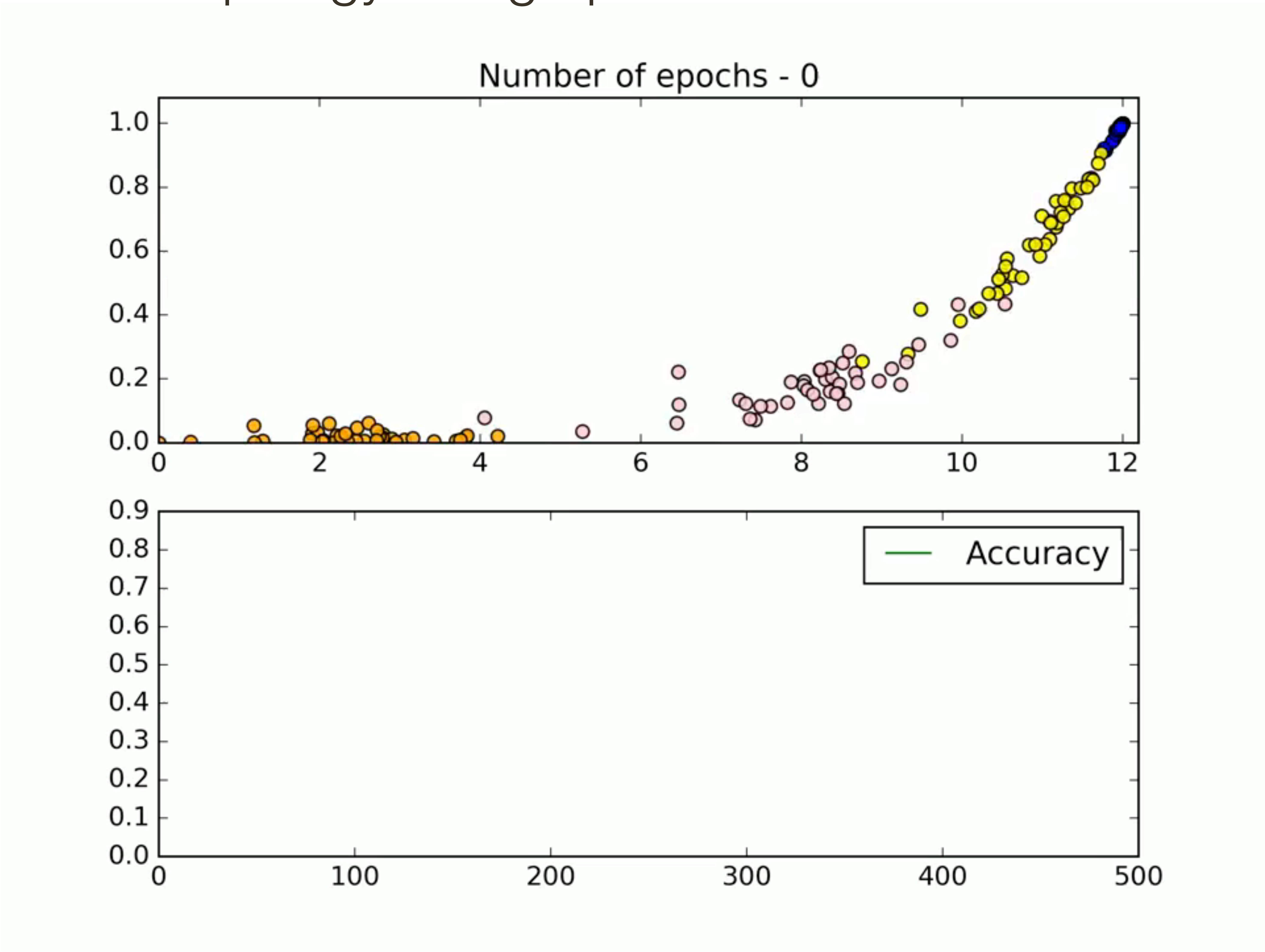
A **Little** Theory: **Information Bottleneck** [Tishbi et al., 1999]

50 networks of same topology being optimized



A **Little** Theory: **Information Bottleneck** [Tishbi et al., 1999]

50 networks of same topology being optimized



A **Little** Theory: **Information Bottleneck** [Tishbi et al., 1999]

Limitation: Does not seem to work for non-Tanh activations (e.g., ReLU)