Language Technologies Institute

Carnegie Mellon University

Advanced Multimodal Machine Learning

Lecture 5.1: Unsupervised learning and Multimodal representations

Louis-Philippe Morency

* Original version co-developed with Tadas Baltrusaitis

Objectives of today's class

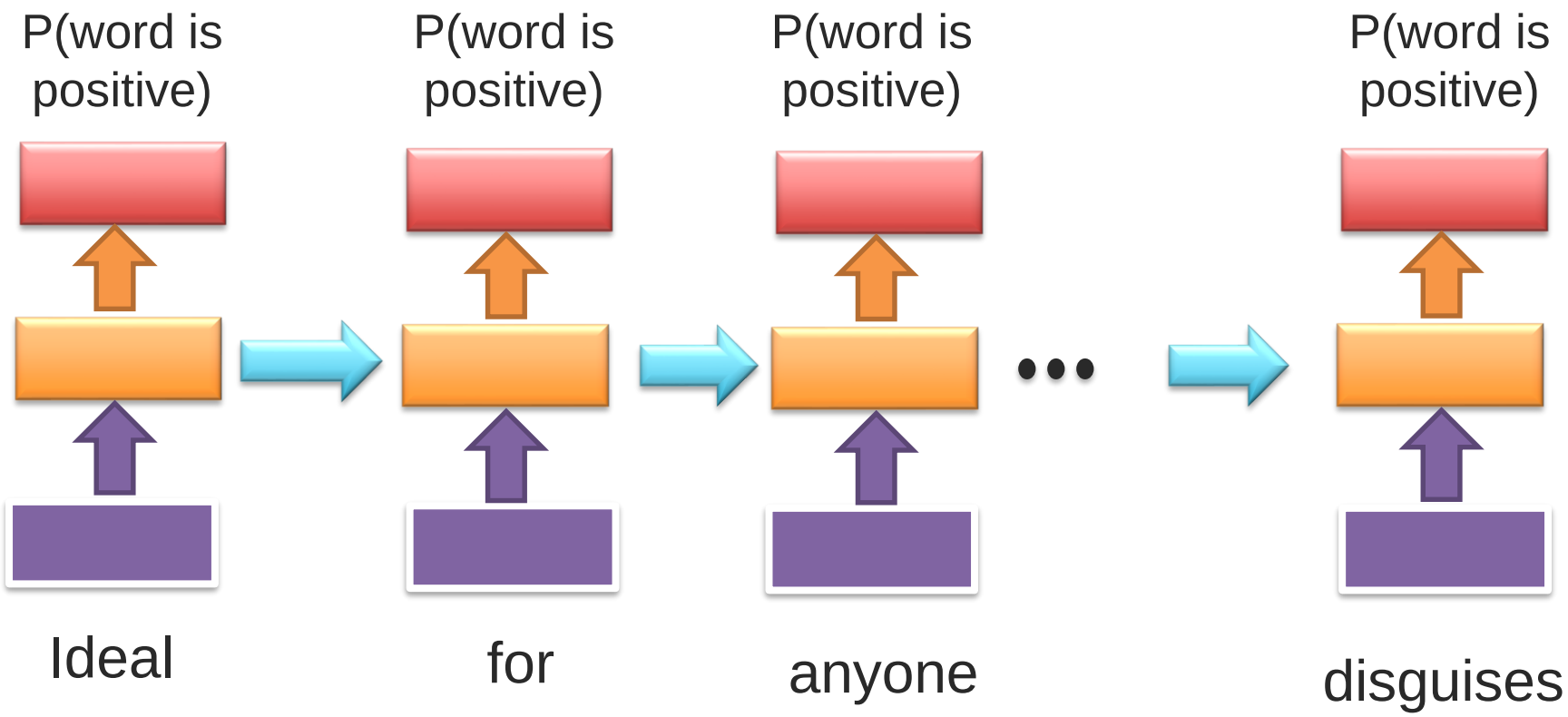
- Gated Recurrent Networks and LSTM
- Unsupervised representation learning
  - Restricted Boltzmann Machines
  - Autoencoders
  - Deep Belief Nets, Stacked autoencoders
- Multi-modal representations
  - Coordinated vs. joint representations
  - Multimodal Deep Boltzmann Machines
  - Deep Multimodal autoencoders
  - Tensor Fusion representation
  - Multi-view LSTM sequence representation

Procedure for AWS Credits

- You need to register on Amazon Educate using your Andrew email address:
  - https://www.awseducate.com/registration
- After this step, each student needs to contact Soumya individually to get their coupon code:
  - Soumya Wadhwa soumyaw@andrew.cmu.edu
- You will receive one coupon code for $50 credit.
- Please be careful when closing your AWS session. Be sure to release the virtual machine!

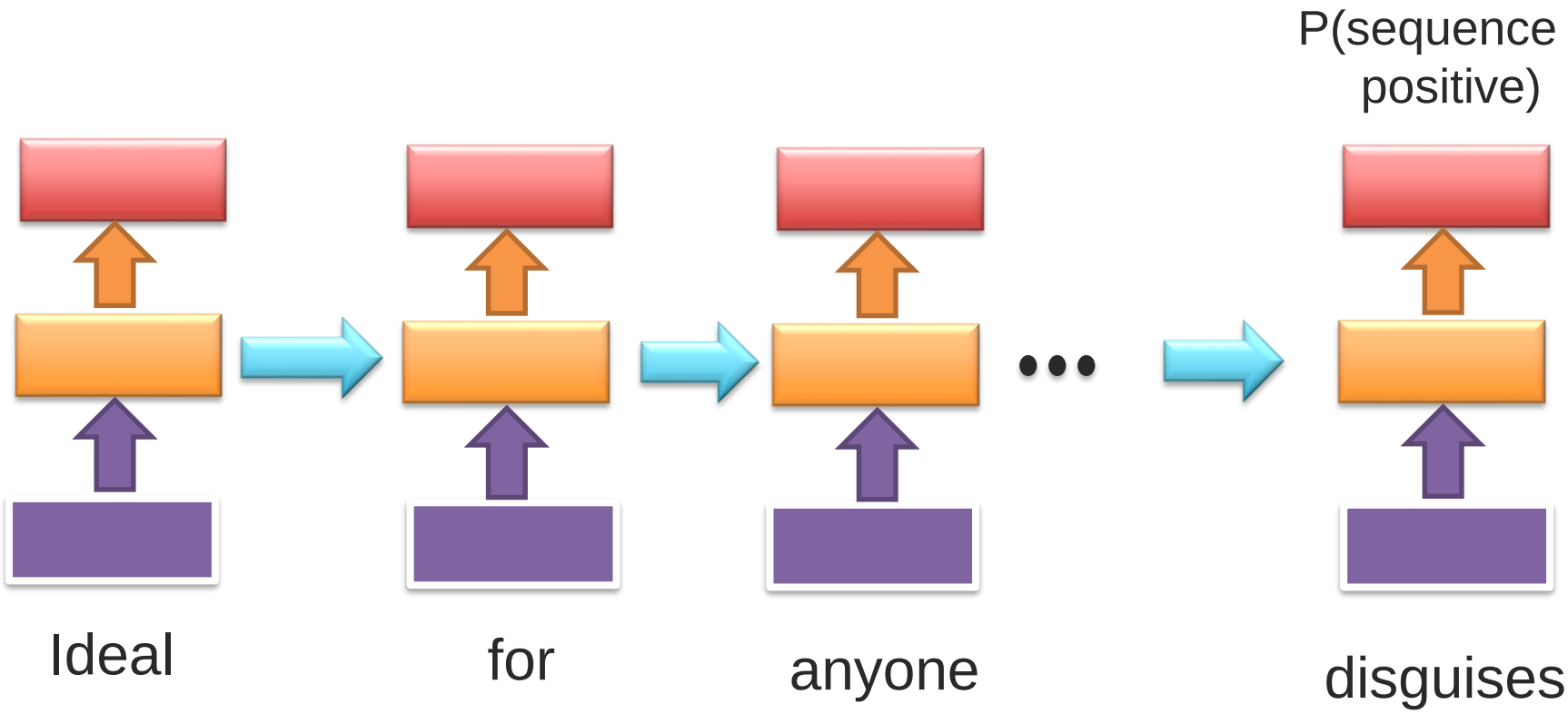
Gated Recurrent Neural Networks

RNN for Sequence Prediction



$$L = \sum_t L^{(t)} = \sum_t -logP(Y = y^{(t)}|\boldsymbol{z}^{(t)})$$

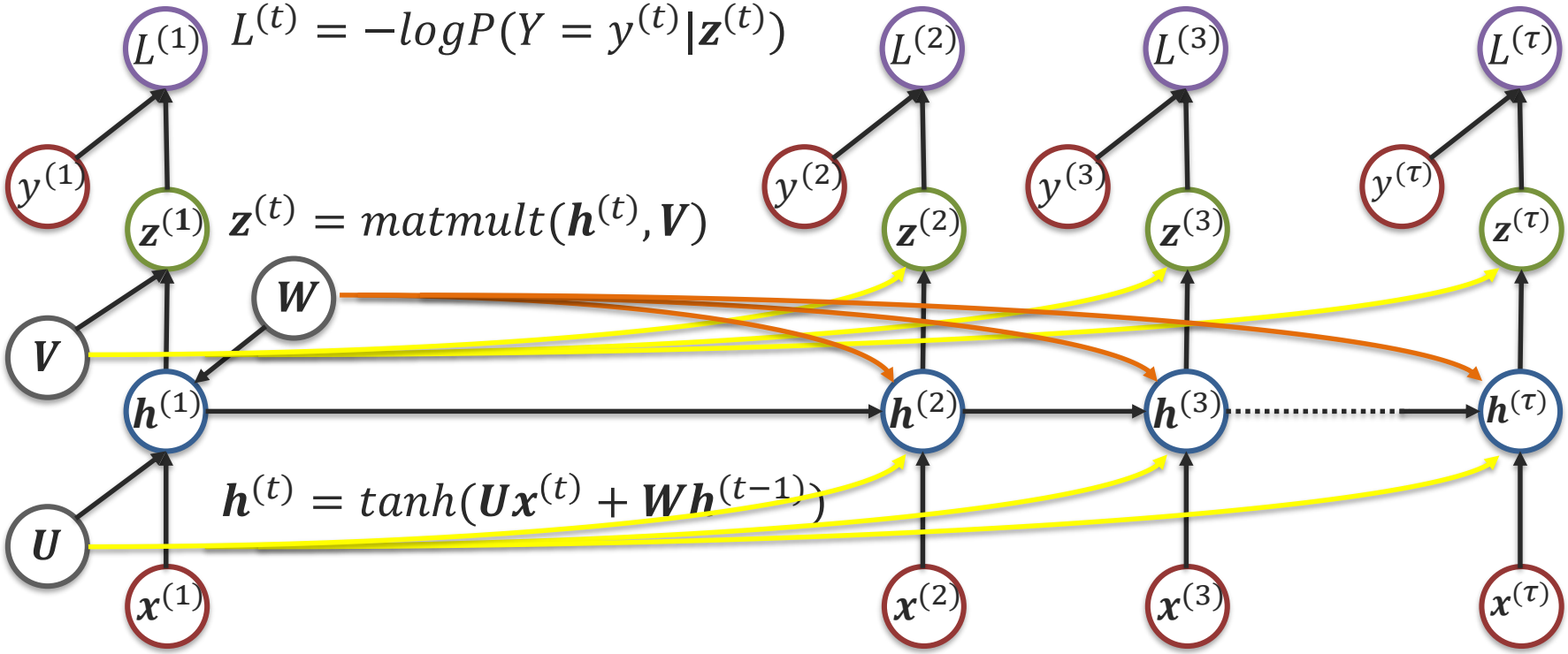
RNN for Sequence Prediction



$$L = L^{(N)} = -logP(Y = y^{(N)}|\boldsymbol{z}^{(N)})$$

Recurrent Neural Networks

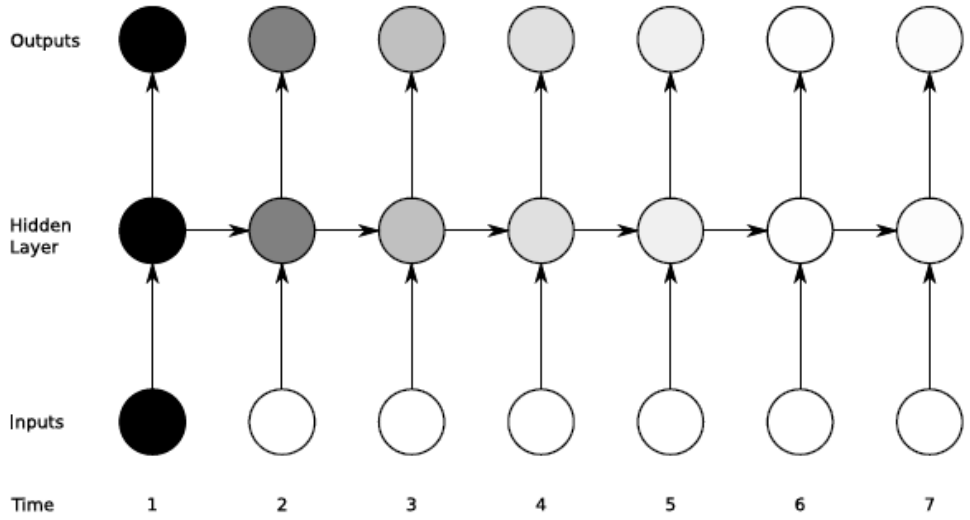
$$L = \sum_t L^{(t)}$$



Long-term Dependencies

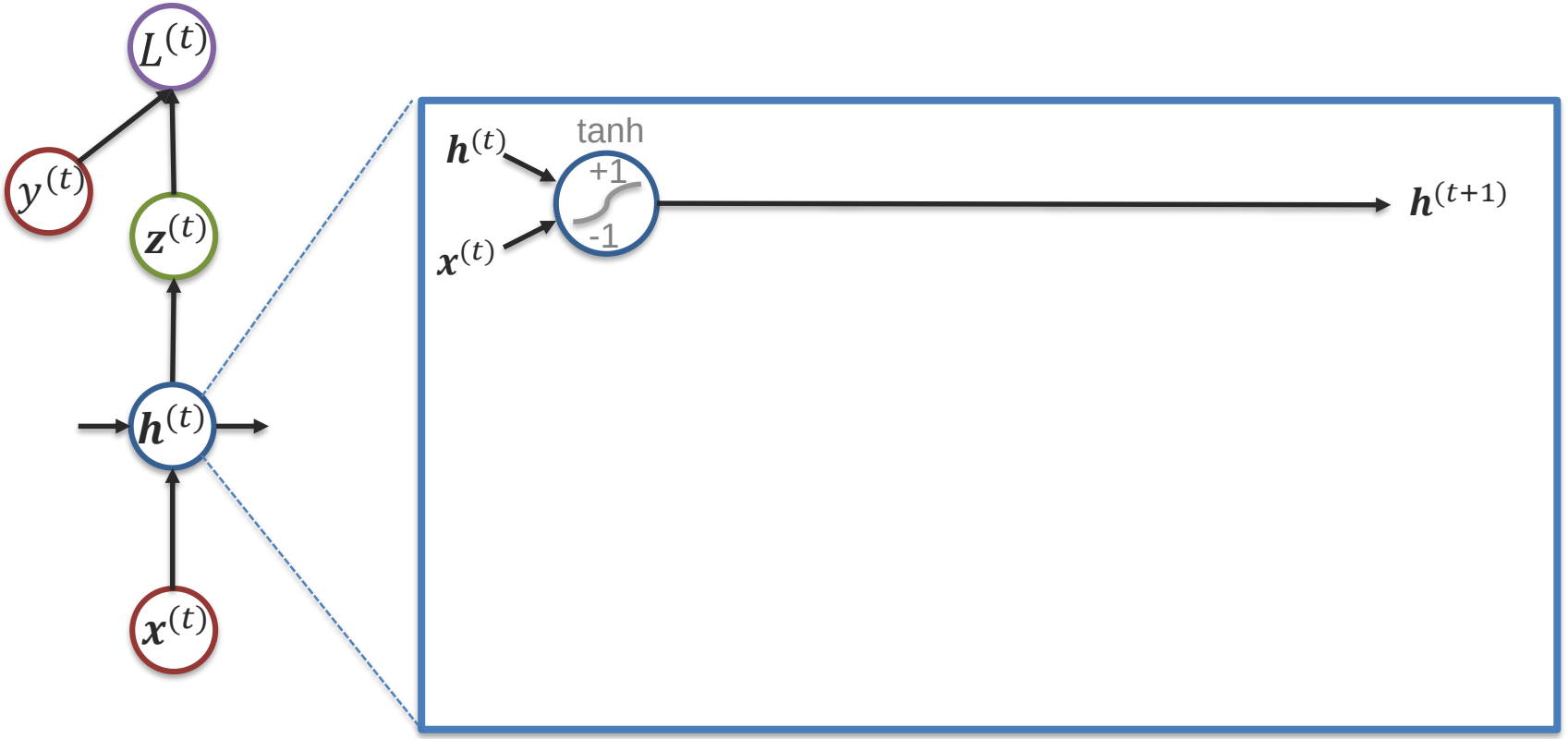
Vanishing gradient problem for RNNs:

$$\boldsymbol{h}^{(t)} \sim tanh(\boldsymbol{W}\boldsymbol{h}^{(t-1)})$$



- The influence of a given input on the hidden layer, and therefore on the network output, either decays or blows up exponentially as it cycles around the network's recurrent connections.

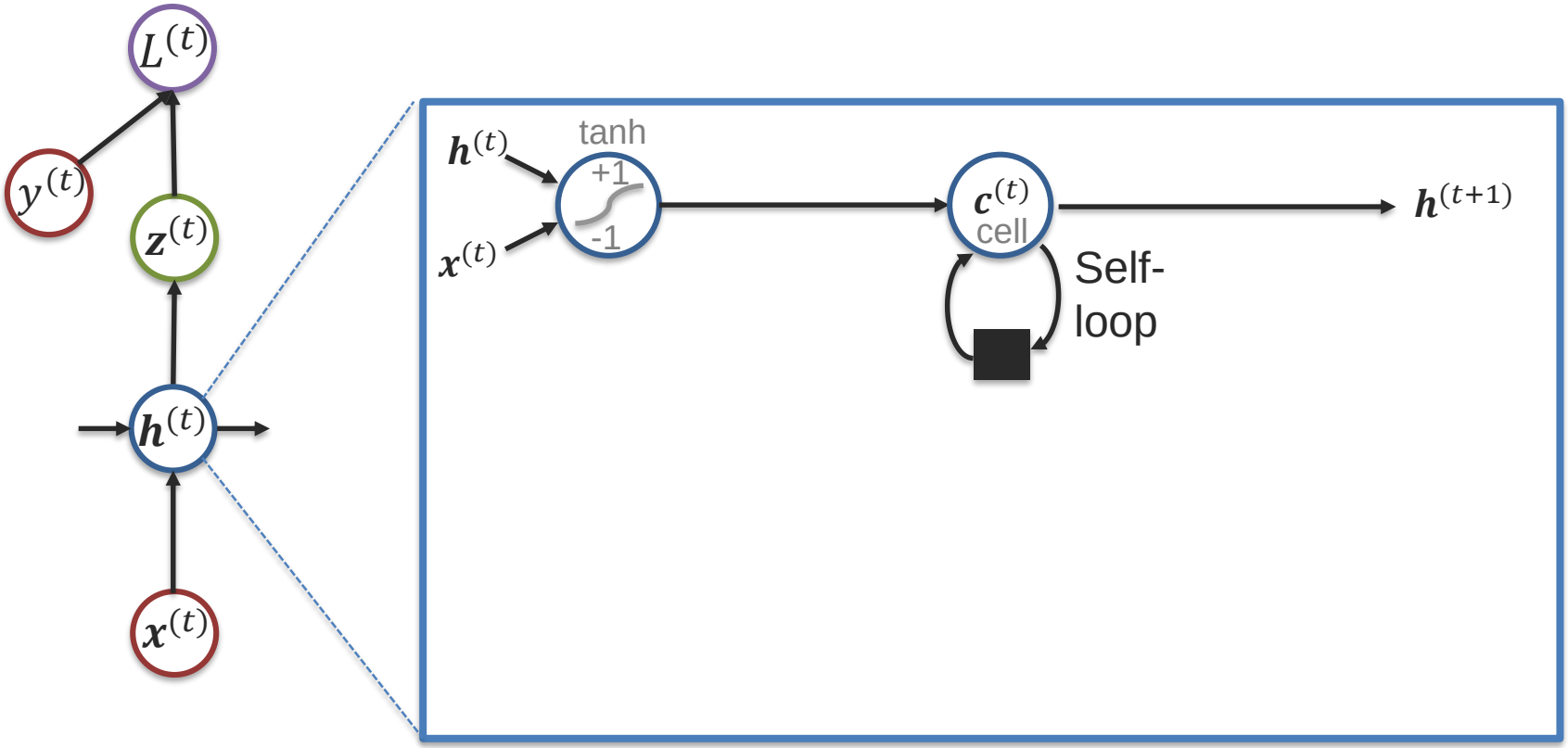
Recurrent Neural Networks



LSTM ideas: (1) "Memory" Cell and Self Loop

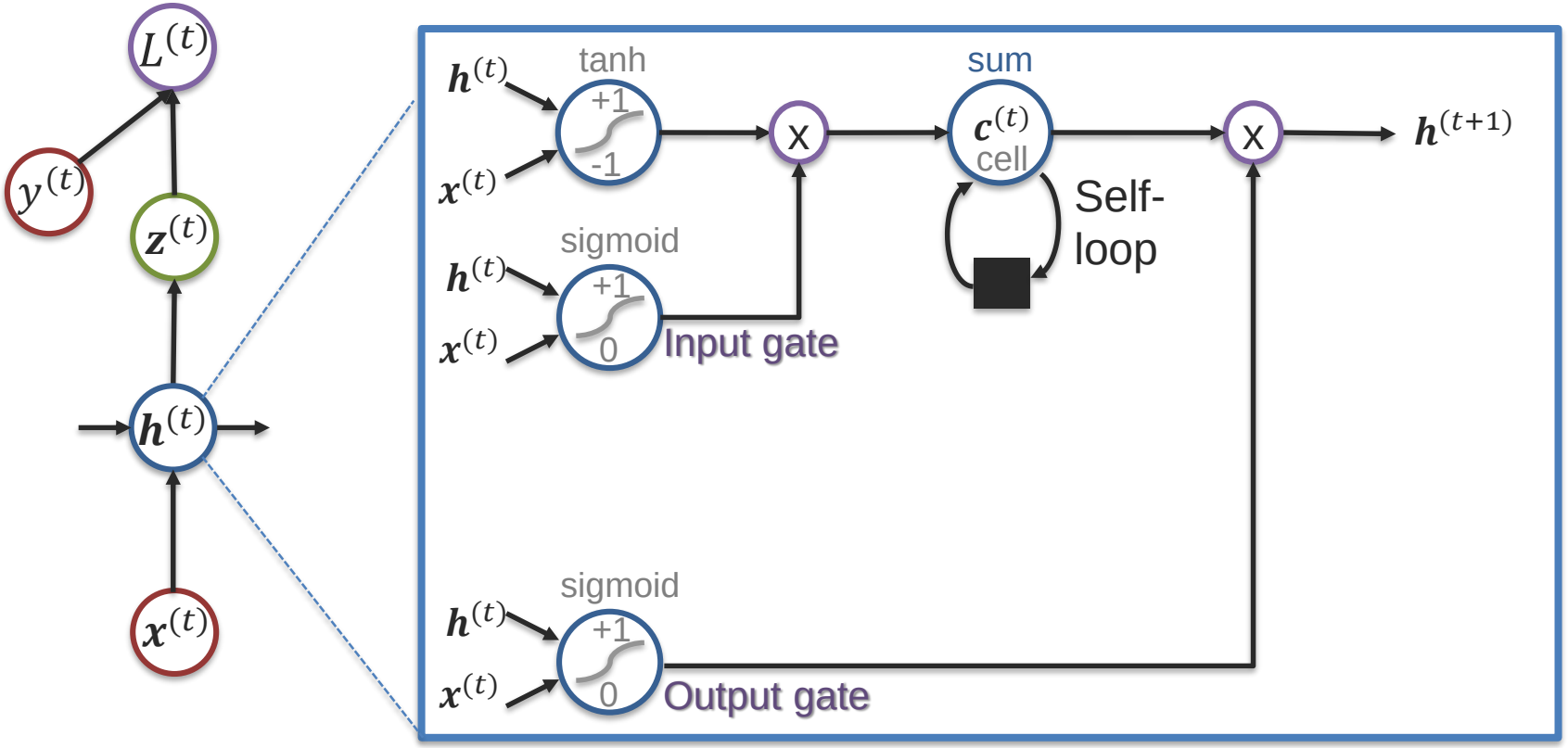
[Hochreiter and Schmidhuber, 1997]

Long Short-Term Memory (LSTM)



LSTM Ideas: (2) Input and Output Gates

[Hochreiter and Schmidhuber, 1997]

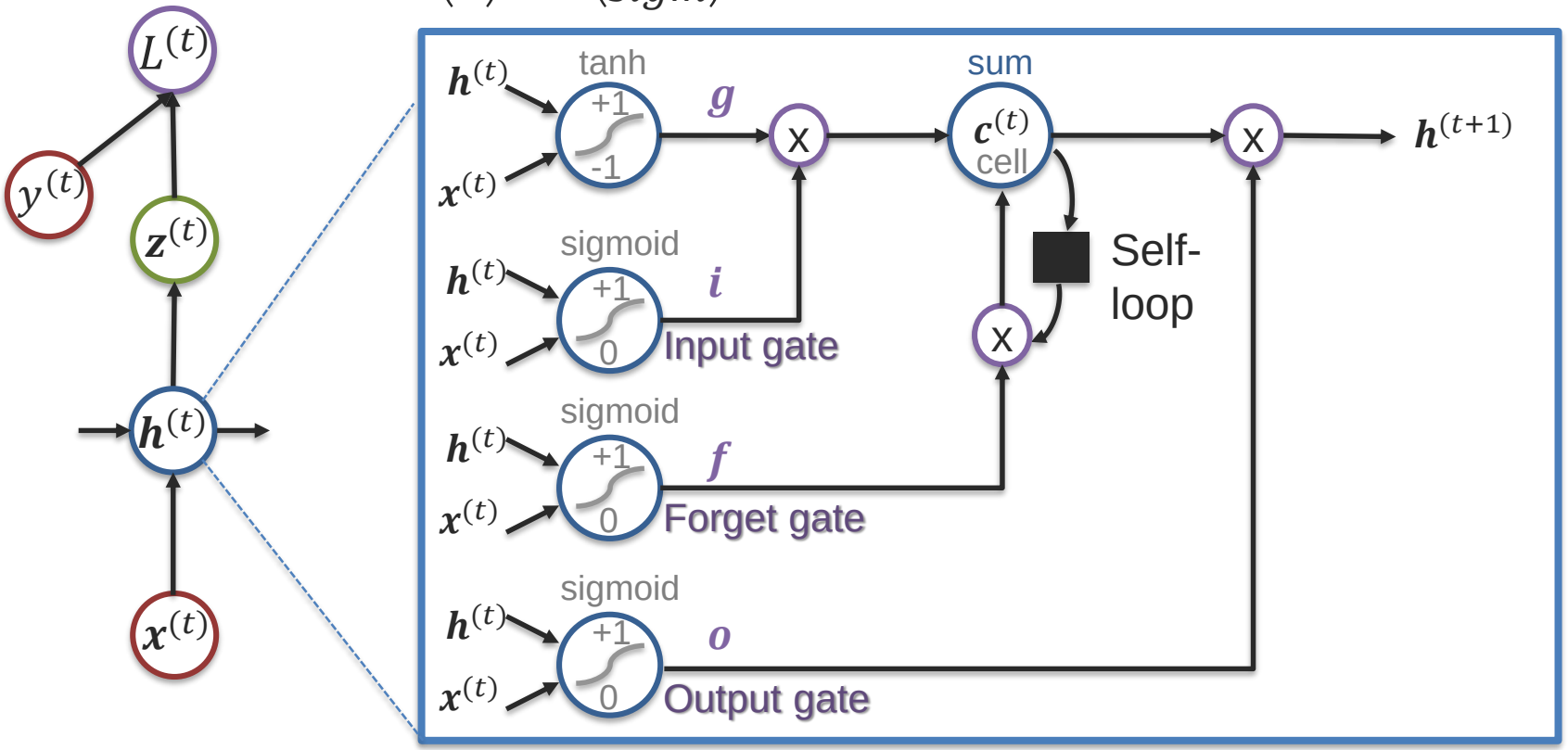


LSTM Ideas: (3) Forget Gate [Gers et al., 2000]

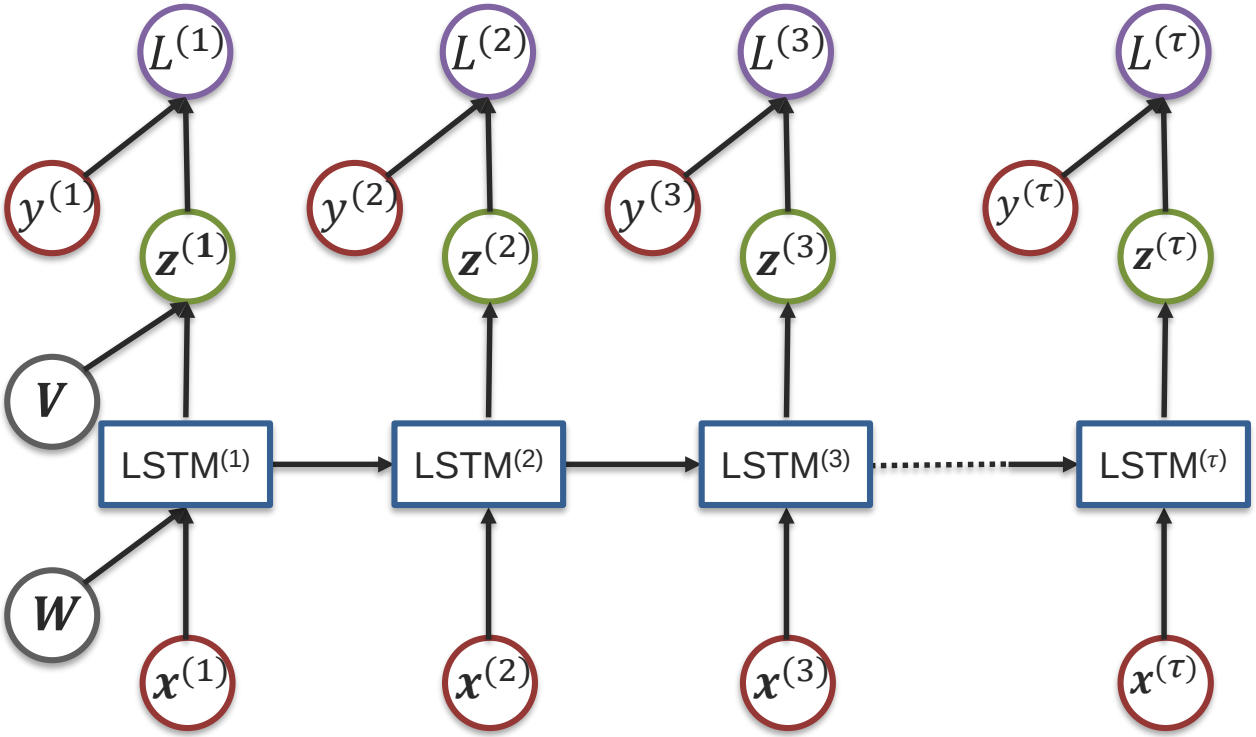
$$\begin{pmatrix} \boldsymbol{g} \\ \boldsymbol{i} \\ \boldsymbol{f} \\ \boldsymbol{o} \end{pmatrix} = \begin{pmatrix} tanh \\ sigm \\ sigm \\ sigm \end{pmatrix} \boldsymbol{W} \begin{pmatrix} \boldsymbol{h}^{(t)} \\ \boldsymbol{x}^{(t)} \end{pmatrix}$$

$$\boldsymbol{c}^{(t)} = \boldsymbol{f} \odot \boldsymbol{c}^{(t-1)} + \boldsymbol{i} \odot \boldsymbol{g}$$

$$\boldsymbol{h}^{(t)} = \boldsymbol{o} \odot \tanh(\boldsymbol{c}^{(t)})$$

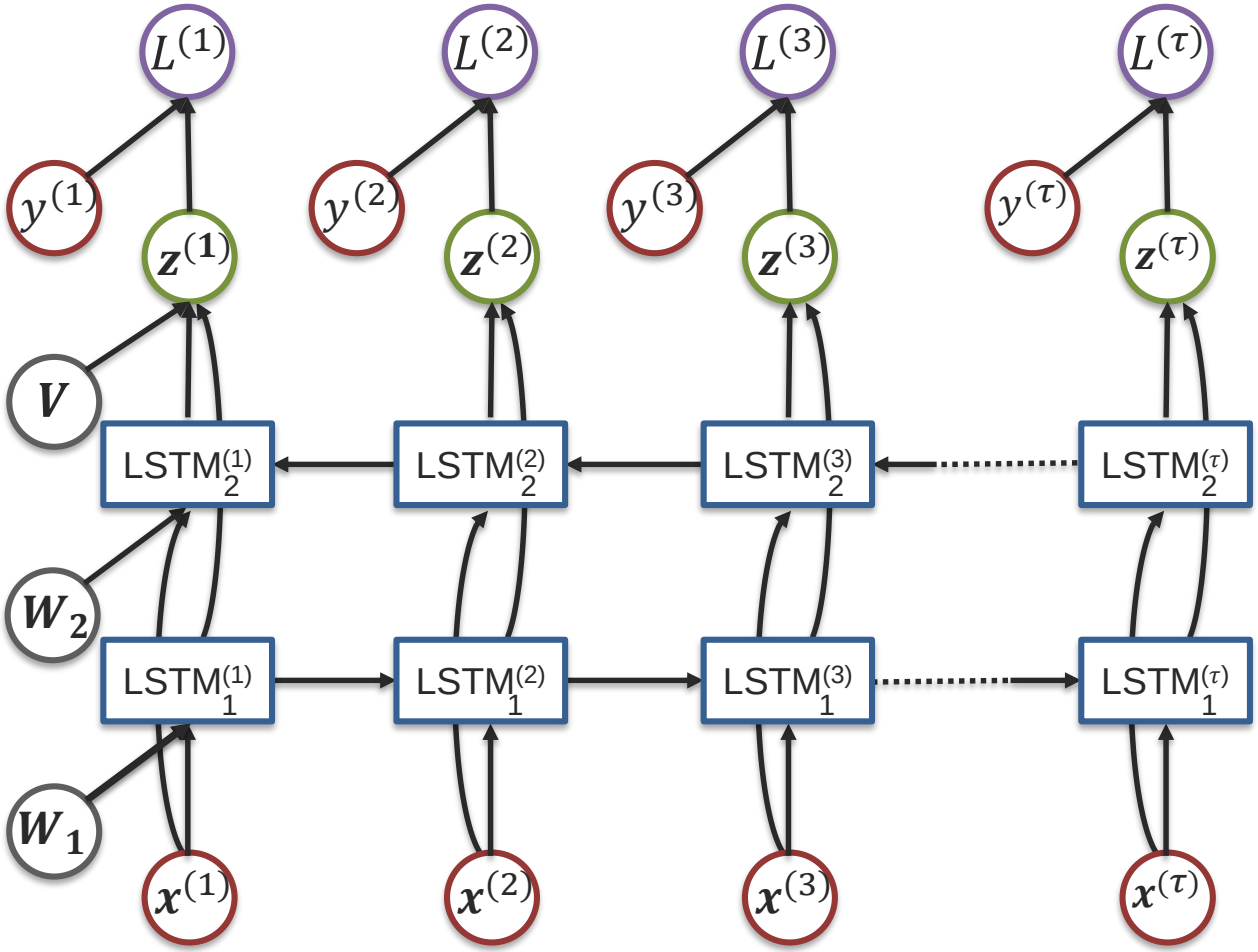


Recurrent Neural Network using LSTM Units

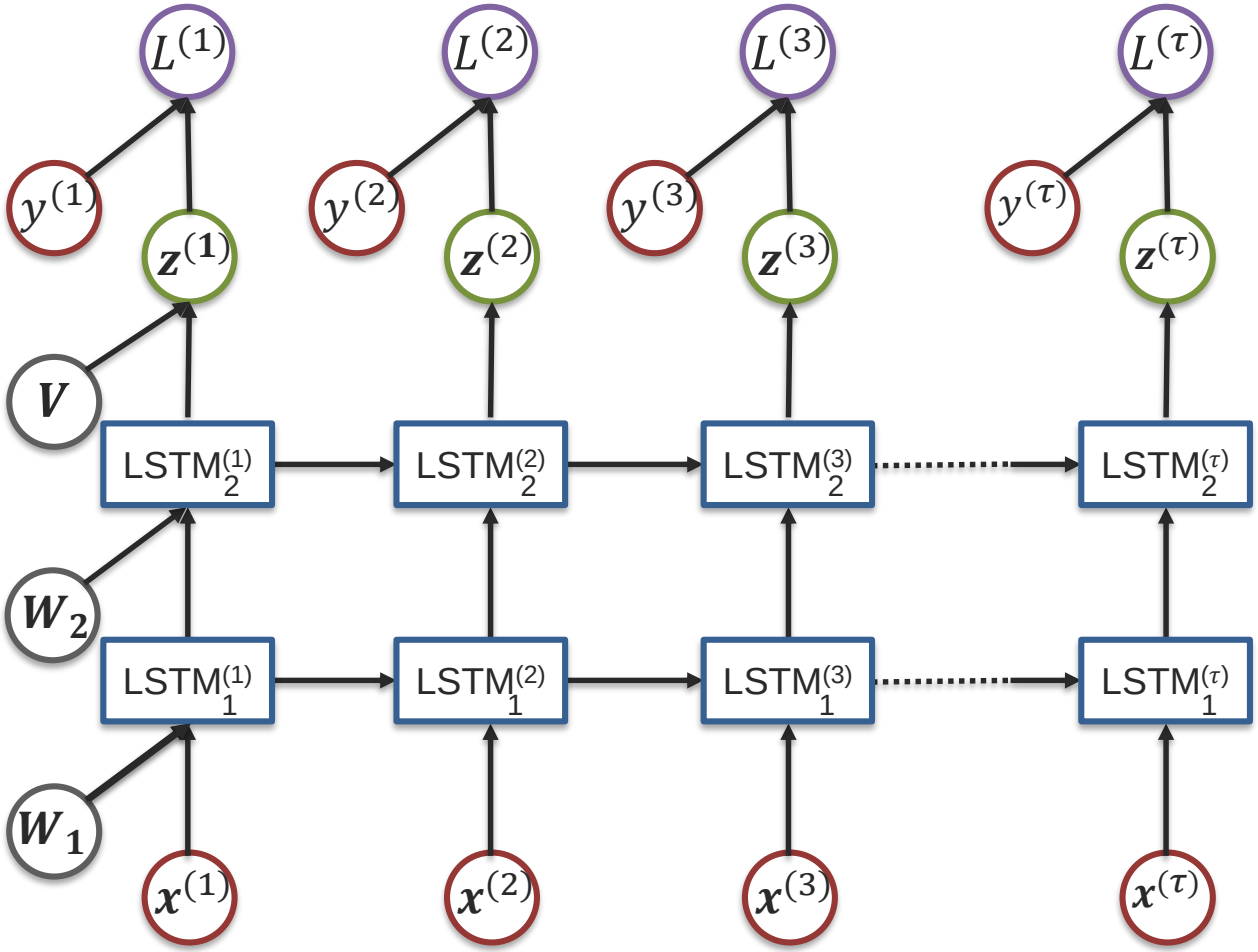


Gradient can still be computer using backpropagation!

Bi-directional LSTM Network



Deep LSTM Network



Unsupervised representation learning

Unsupervised learning

- We have access to $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ and not $Y = \{y_1, y_2, \ldots, y_n\}$
- Why would we want to tackle such a task
- 1. Extracting interesting information from data
  - Clustering
  - Discovering interesting trends
  - Data compression
- 2. Learn better representations

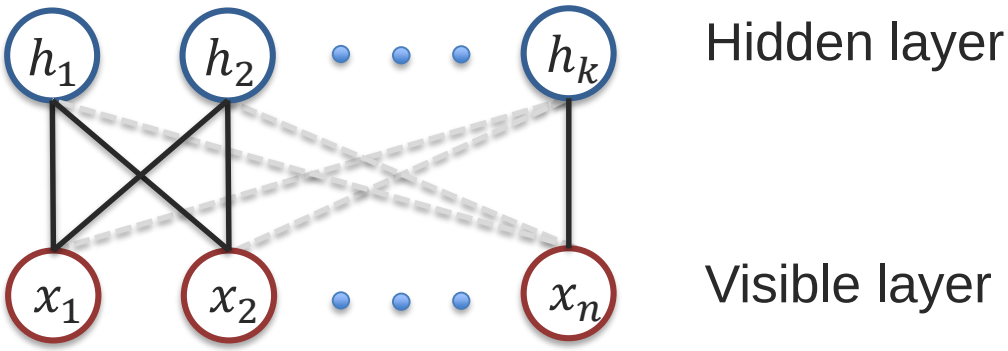
Unsupervised representation learning

- 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
  - Potentially allowing for better generalizability
- Use for initialization of supervised task, especially when we have a lot of unlabeled data and much less labeled examples

Restricted Boltzmann Machines

Restricted Boltzmann Machine (RBM)

- Undirected Graphical Model
- A generative rather than discriminative model
- Connections from every hidden unit to every visible one
- No connections across units (hence Restricted), makes it easier to train and do inference on



[Smolensky, Information Processing in Dynamical Systems: Foundations of Harmony Theory, 1986]

Restricted Boltzmann Machine

- Model the joint probability of hidden state and observation

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

Joint probability, positive value

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

Normalization function so that the probabilities sum to one

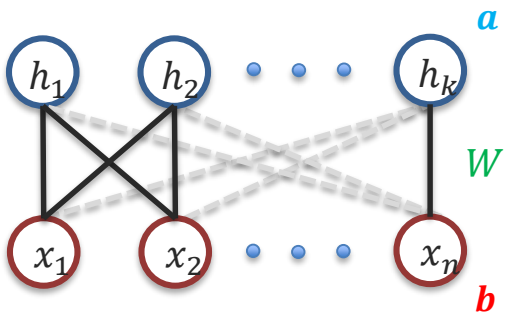
$$\mathrm{E} = -\boldsymbol{x} W \boldsymbol{h} - \boldsymbol{b}^T \boldsymbol{x} - \boldsymbol{a}^T \boldsymbol{h}$$

$$\mathrm{E} = -\sum_i \sum_j w_{i,j} x_i h_j - \sum_i b_i x_i - \sum_j a_j h_j$$

Interaction term

Bias terms

Hidden and visible layers are binary (e.g. $x = \{0, \dots, 1, 0, 1\}$),
Model parameters to learn $\theta = \{W, \boldsymbol{b}, \boldsymbol{a}\}$

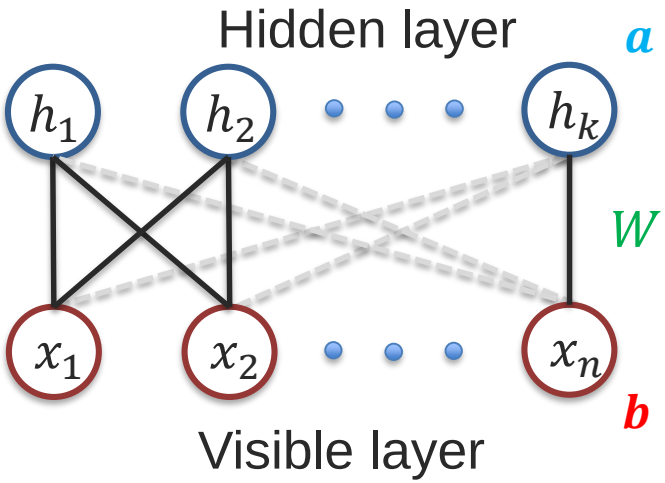


Hidden layer (binary)

Visible layer (binary)

RBM inference (have a trained θ)

- For inference
  - $p(h_j = 1|\boldsymbol{x};\theta) = \sigma(\sum_i x_i w_{ij} + a_j)$,
  - $p(x_i = 1|\boldsymbol{h};\theta) = \sigma(\sum_j h_j w_{ij} + b_i)$
  - derived from the joint probability definition
- Conditional inference is easy and of sigmoidal form
  - Given a trained model θ and an observed value $\boldsymbol{x}$ can easily infer $\boldsymbol{h}$
  - Given a trained model θ and an hidden layer value $\boldsymbol{h}$ can easily infer $\boldsymbol{x}$
- Need to sample as we get probabilities rather than values



RBM training (learning the θ)

- Want to have a model that leads to good likelihood of training data
- First express the data likelihood (through marginal probability):
  - $p(\boldsymbol{x};\theta) = \frac{\sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{x},\boldsymbol{h};\theta))}{Z}$ $\quad Z = \sum_{\boldsymbol{x}} \sum_{\boldsymbol{h}} \exp(-\mathrm{E}(\boldsymbol{x},\boldsymbol{h};\theta))$
- Want to optimize:
  - $\mathrm{argmin}_{\theta} \left[\sum_t -\log\left(p\left(\boldsymbol{x}^{(t)};\theta\right)\right)\right]$, where t is a data sample
  - sum across all samples
  - minimizing negative log likelihood instead of maximizing the likelihood
- To Approximate computation of model term using Contrastive Divergence
  - Based on Markov Chain Monte Carlo (Gibbs) sampling

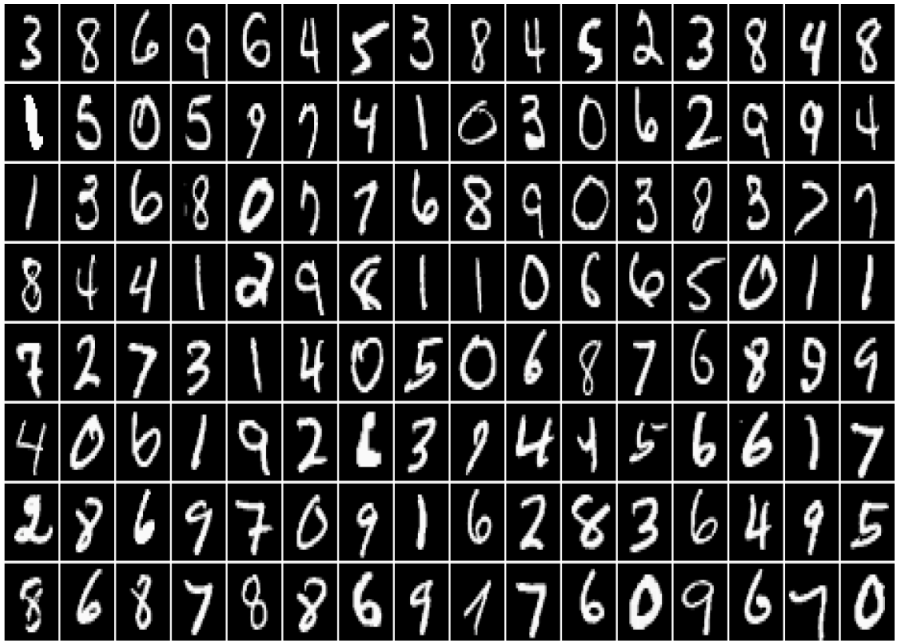
[G. Hinton, Training Products of Experts by Minimizing Contrastive Divergence, 2002]

See http://www.iro.umontreal.ca/~lisa/twiki/bin/view.cgi/Public/DBNEquations for more details

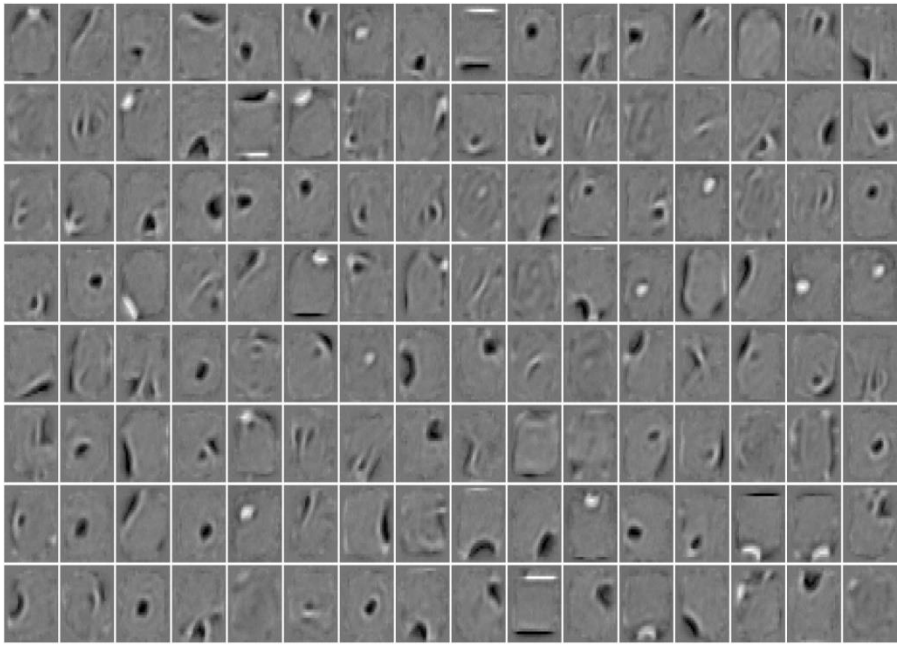
RBM extensions

- So far have only modeled binary input and hidden states
- Gaussian-Bernoulli RBM allows for real value modeling
  - Changes the inference and training only very slightly
  - Visible units are modeled as real values (under a Gaussian distribution), but hidden units are still binary
  - [Hinton and Salakhutdinov, Reducing the Dimensionality of Data with Neural Networks, 2006]
- Replicated Softmax to model one-hot encoding style vectors
  - [Salakhutdinov and Hinton, Replicated Softmax: an Undirected Topic Model, 2009]
  - Now not as relevant due to word2vec
- Only requires a small change in some of the equations
- Can also introduce sparsity in hidden layers (sometimes helps)
  - [Lee et al., Sparse deep belief net model for visual area V2, 2007]

Examples of what the model learns



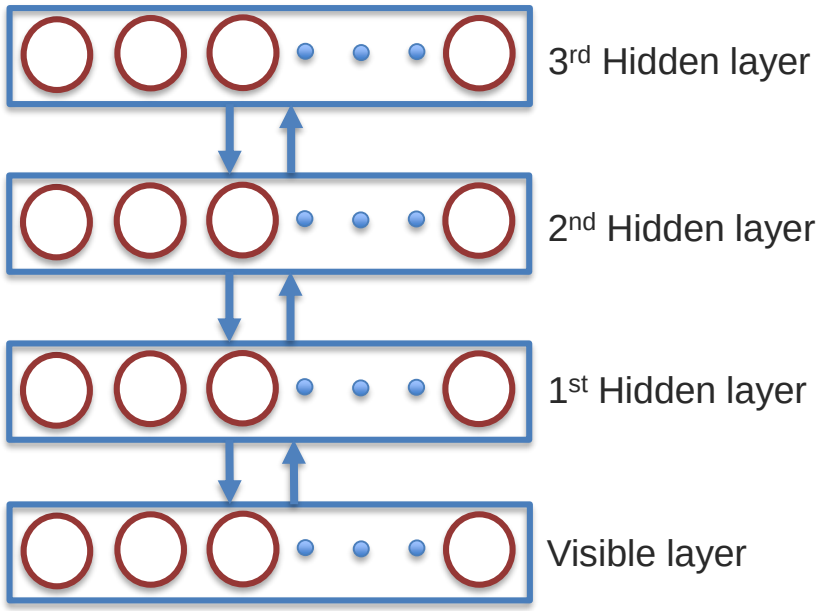
MNIST data



Learned W terms for each hidden unit

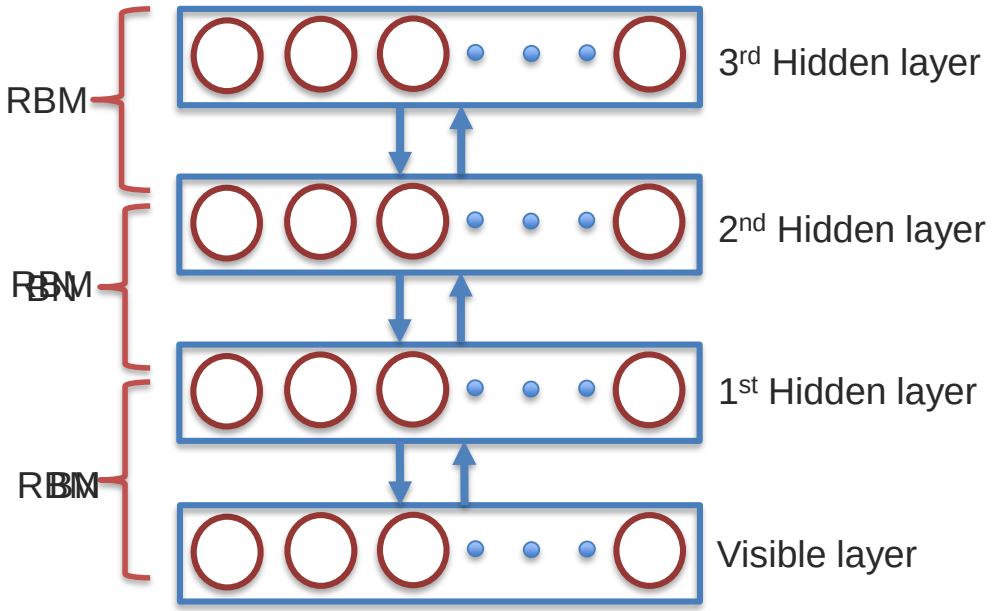
Deep Restricted Boltzmann Machines (DBMs)

- Can stack RBMs together to lead do deep versions of them
- The visible layer can be binary, Gaussian or Bernoulli
- Training fully end to end is very difficult
- Greedy layer-wise training
- Combine the RBMs layer by layer



Deep Belief Networks (DBN)

- To make it easier used Deep Belief Networks
  - Actually came before Deep RBMs
- Simplifies model training
- Turn the undirected model to directed one, making the interaction simpler

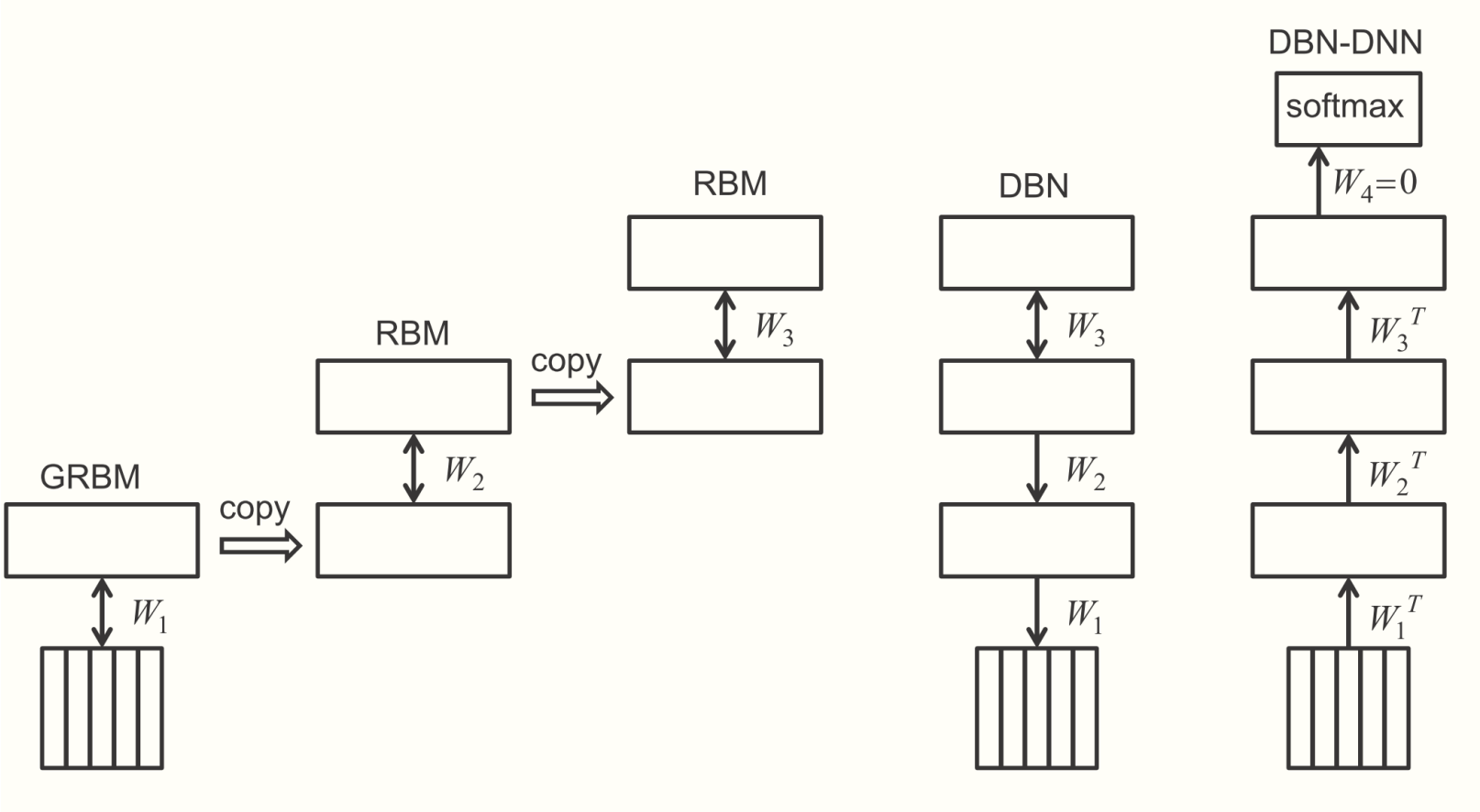


For more details see [Salakhutdinov and Hinton, Deep Boltzmann Machines, 2009]

What can you do with them

- On their own RBMs are very interesting but not necessarily useful
- Stacking them can lead to more interesting models
  - Can use the representation directly for some task
- Use them to pre-train or initialize discriminative models
  - Initialize Deep Neural Networks from them
  - We can convert the DBN weights to those of DNN
- Major early success of deep learning for Automatic Speech Recognition

Audio representation for speech recognition

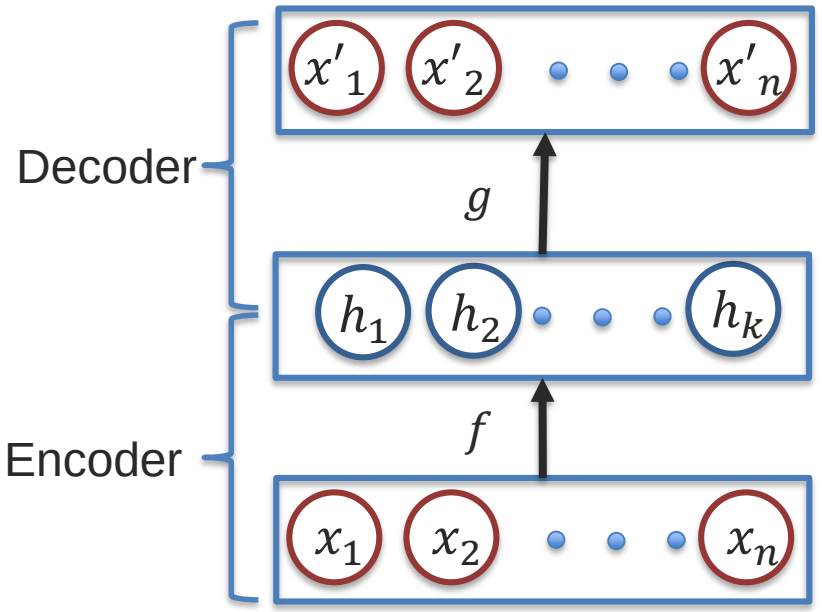


[Hinton et al., Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups, 2012]

Autoencoders

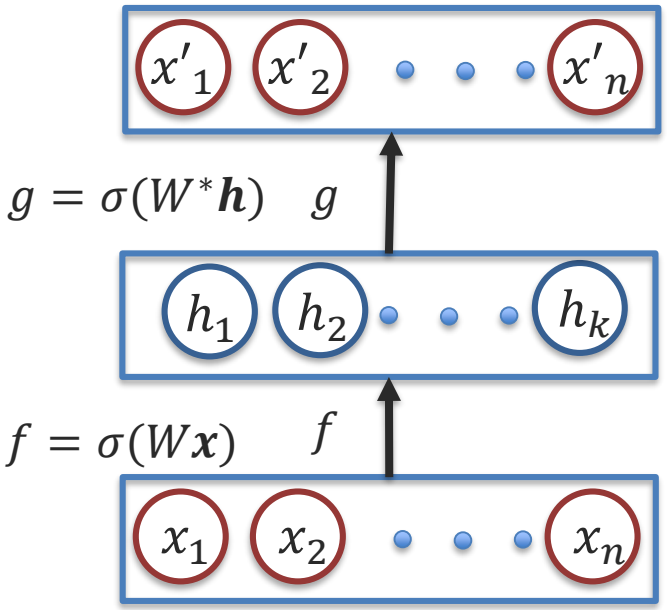
Autoencoders – an alternative to RBM

- What does auto mean?
  - Greek for self – self encoding
- Feed forward network intended to reproduce the input
- Two parts encoder/decoder
  - $x' = f(g(x))$ – score function
  - g - encoder
  - f - decoder



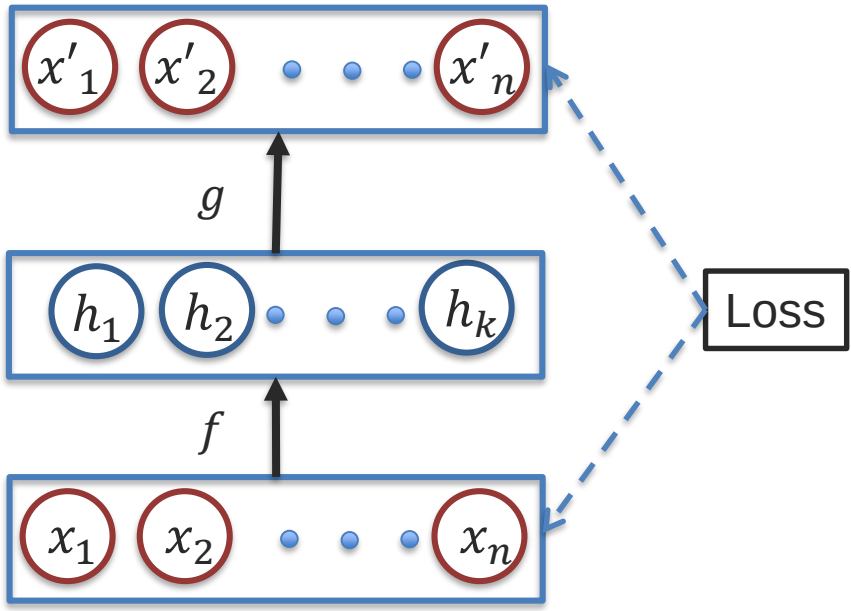
Autoencoders

- Mostly follows Neural Network structure
  - Typically a matrix multiplication followed by a nonlinearity (e.g sigmoid)
- Activation will depend on type of $\boldsymbol{x}$
  - Sigmoid for binary
  - Linear for real valued
- Often we use *tied weights* to force the sharing of weights in encoder/decoder
  - $W^* = W^T$
- word2vec is actually a bit similar to an autoencoder (except for the auto part)



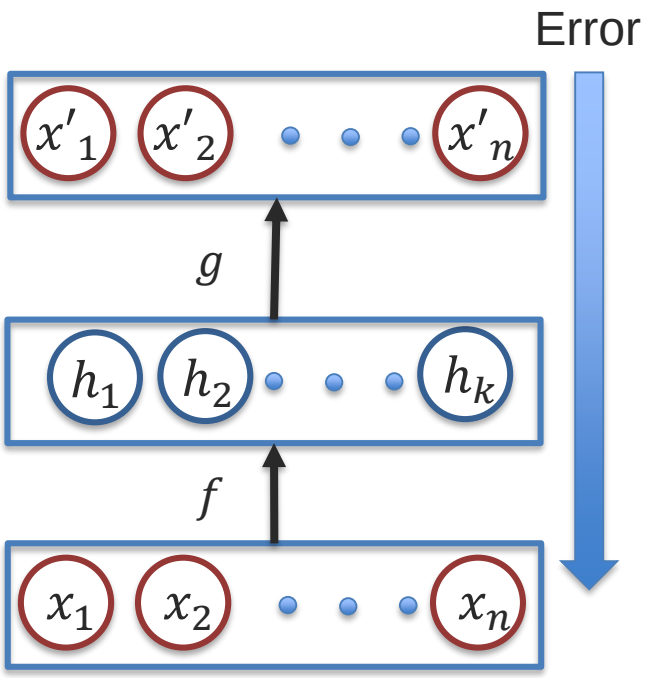
Loss function

- Any differentiable similarity function
- Cross-entropy for binary $\boldsymbol{x}$
  - $L = -\sum_k (x_k \log(x'_k) + (1 - x_k) \log(1 - x'_k))$
- Euclidean for real valued $\boldsymbol{x}$
  - $L = \frac{1}{2} \sum_k (x_k - x'_k)^2$
- Cosine similarity etc.
- Depends on the data being modeled



Learning

- To learn the model parameters (W^*, W), we use back-propagation
- In case of Euclidean (with linear act) and Cross-entropy (with sigmoid act), we just have $(x' - x)$ error to propagate
- If we're using *tied* weights, gradients need to be summed (like back propagation through time in RNN)
- Can use batch/stochastic gradient descent as before

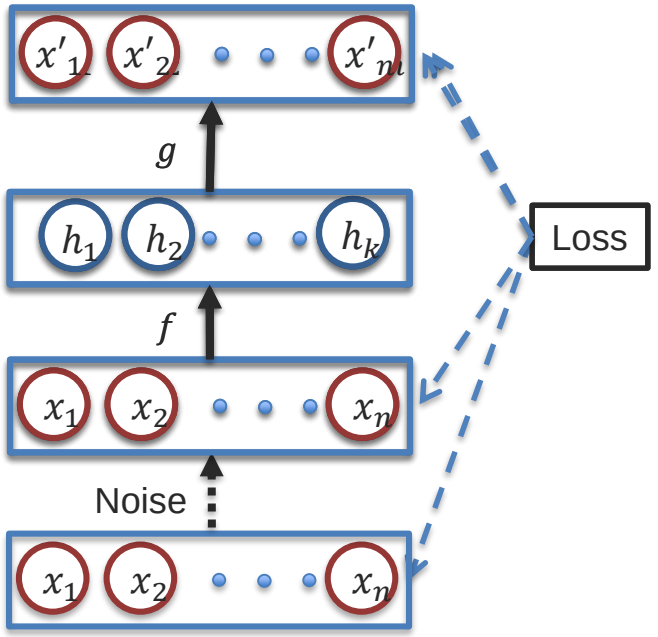


Hidden layer dimensionality

- Smaller that input - Undercomplete
  - Will compress the data, reconstruction of data far from training distribution will be difficult
  - Linear-linear encoder-decoder with Euclidean loss is actually equivalent to PCA (under certain data normalization)
- Larger than input - Overcomplete
  - No compression needed
  - Can trivially learn to just copy, so no structure is extracted
  - Does not encourage to lean meaningful features, **a problem**

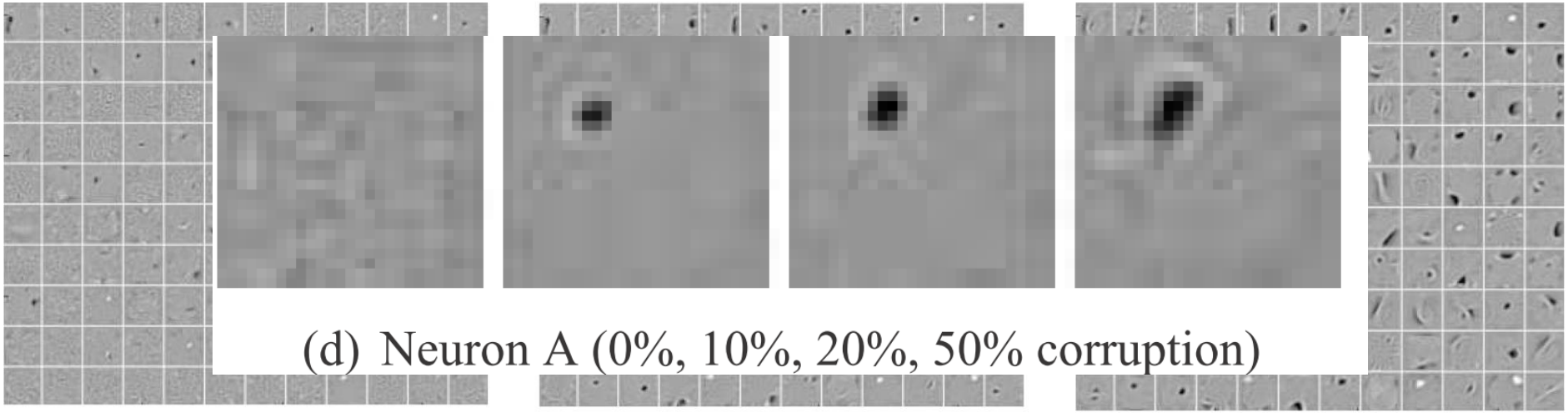
Denoising autoencoder

- Simple idea
  - Add noise to input $\boldsymbol{x}$ but learn to reconstruct original
- Leads to a more robust representation and prevents copying
- Learns what the relationship is to represent a certain $\boldsymbol{x}$
- Different noise added during each epoch



Autoencoder vs denoising autoencoder

- MNIST data (as before)

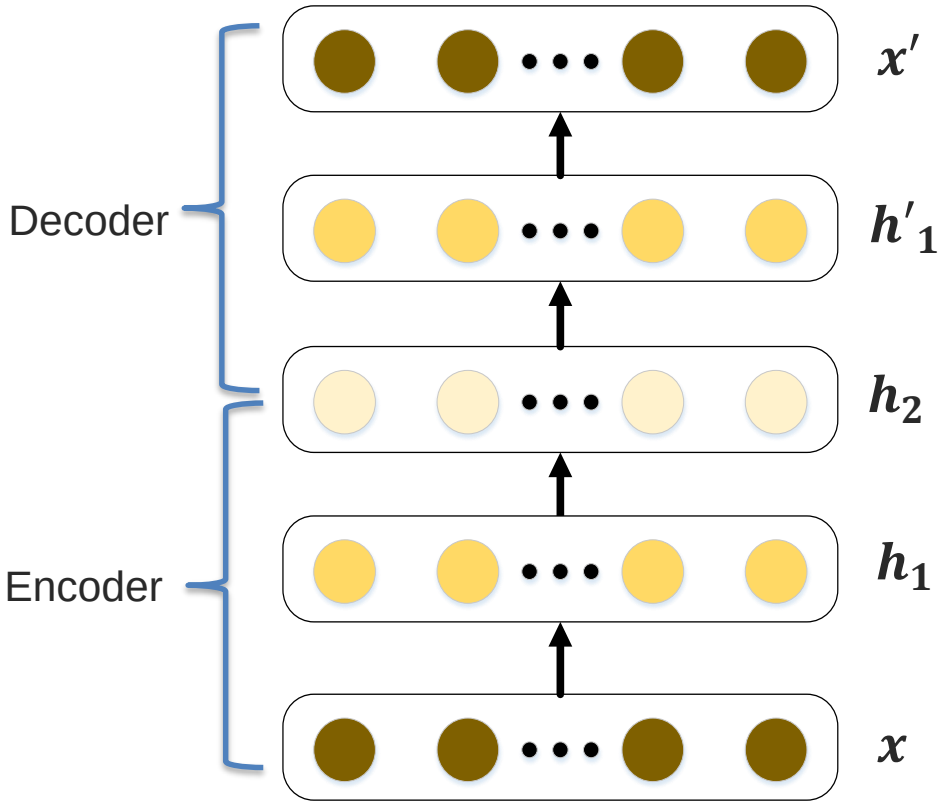


Autoencoder | Denoising autoencoder (25% noise) | Denoising autoencoder (50% noise)

Qualitatively denoising autoencoder leads to more meaningful features

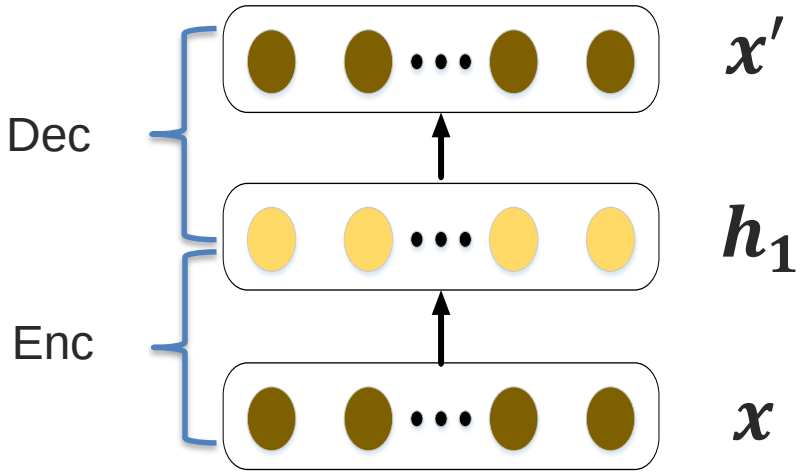
Stacked autoencoders

- Can stack autoencoders as well
- Each encoding unit has a corresponding decoder
- As before, inference is feedforward, but now with more hidden layers



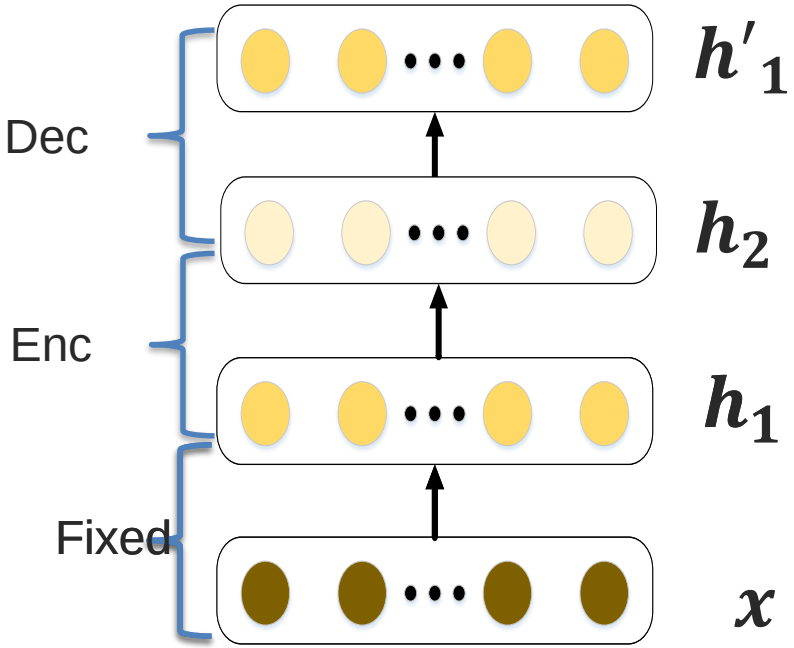
Stacked autoencoders

- Greedy layer-wise training
- Start with training first layer
  - Learn to encode $\boldsymbol{x}$ to $\boldsymbol{h_1}$ and to decode $\boldsymbol{x}$ from $\boldsymbol{h_1}$
  - Use backpropagation



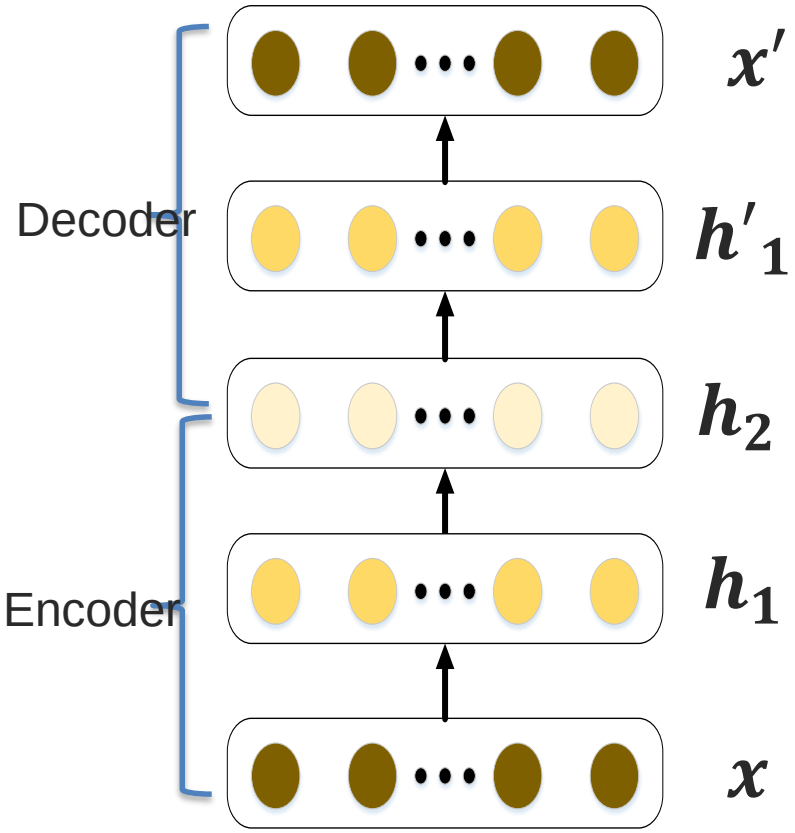
Stacked autoencoders

- Greedy layer-wise training
- Start with training first layer
  - Learn to encode $\boldsymbol{x}$ to $\boldsymbol{h_1}$ and to decode $\boldsymbol{x}$ from $\boldsymbol{h_1}$
  - Use backpropagation
- Map from all $\boldsymbol{x}$'s to $\boldsymbol{h_1}$'s
  - Discard decoder for now
- Train the second layer
  - Learn to encode $\boldsymbol{h_1}$to $\boldsymbol{h_2}$ and to decode $\boldsymbol{h_2}$ from $\boldsymbol{h_1}$
  - Repeat for as many layers



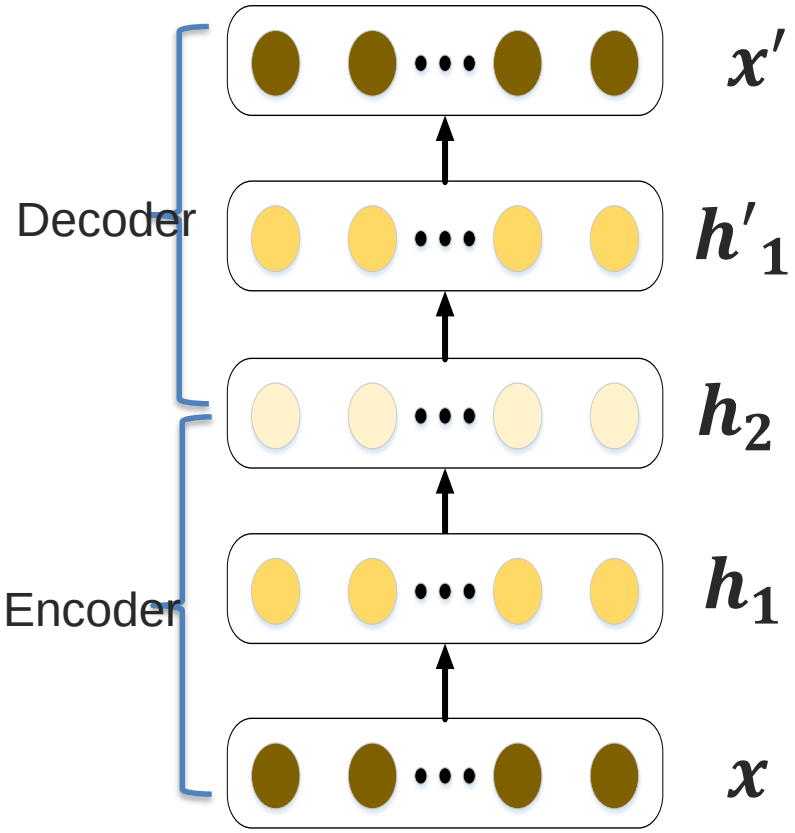
Stacked autoencoders

- Greedy layer-wise training
- Start with training first layer
  - Learn to encode $\boldsymbol{x}$ to $\boldsymbol{h_1}$ and to decode $\boldsymbol{x}$ from $\boldsymbol{h_1}$
  - Use backpropagation
- Map from all $\boldsymbol{x}$'s to $\boldsymbol{h_1}$'s
  - Discard decoder for now
- Train the second layer
  - Learn to encode $\boldsymbol{h_1}$to $\boldsymbol{h_2}$ and to decode $\boldsymbol{h_2}$ from $\boldsymbol{h_1}$
  - Repeat for as many layers
- Reconstruct using previously learned decoders mappings
- Fine-tune the full network end-to-end



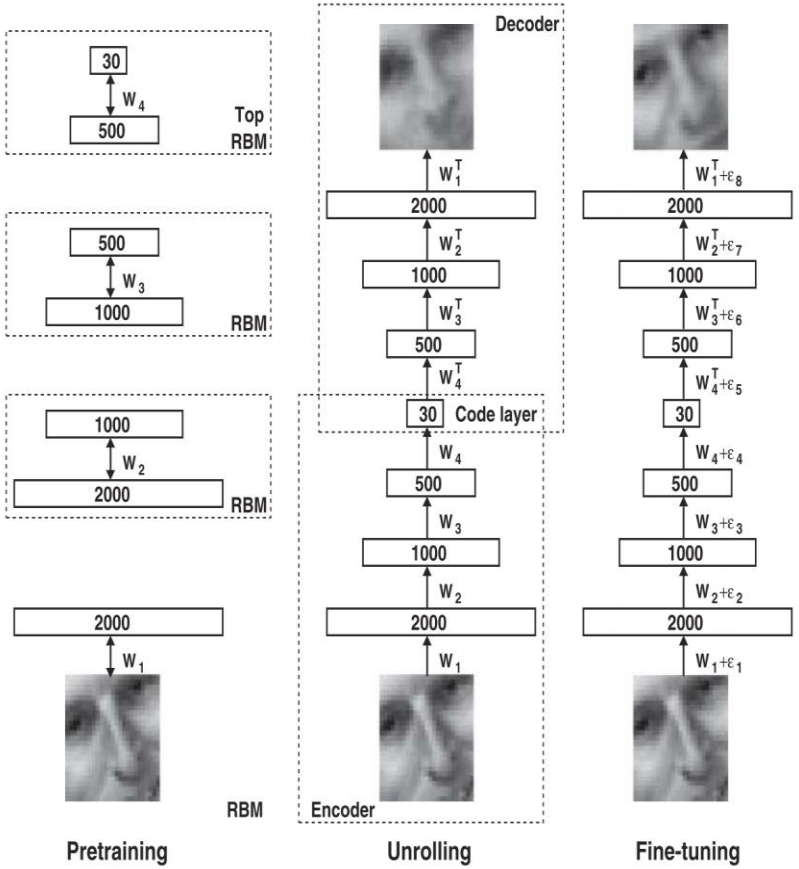
Stacked denoising autoencoders

- Can extend this to a denoising model
- Add noise when training each of the layers
  - Often with increasing amount of noise per layer
  - 0.1 for first, 0.2 for second, 0.3 for third



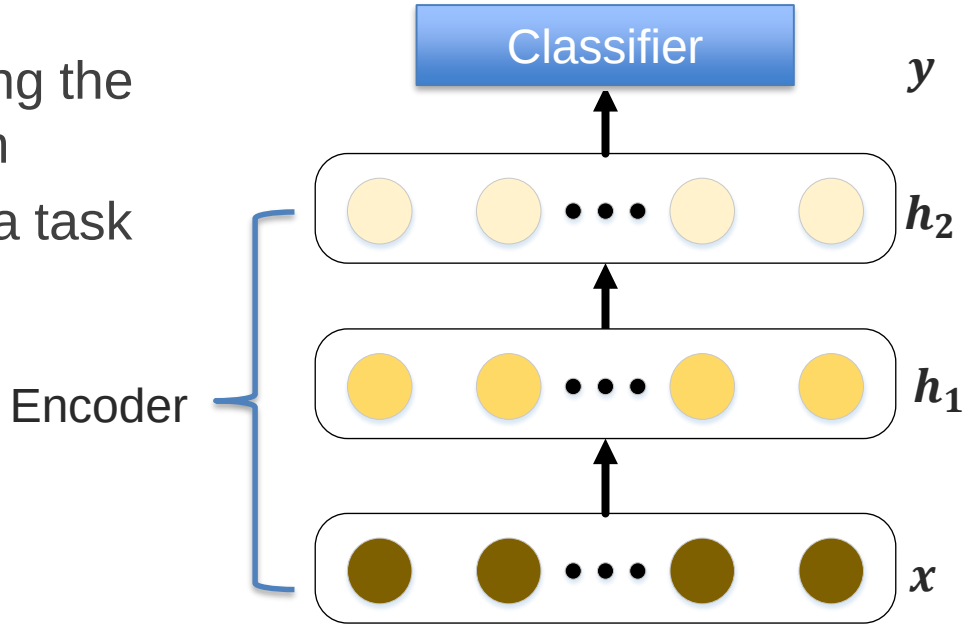
Deep representations

- What can we do with them?
- Compression
  - Can work better than PCA
  - [Hinton and Salatkhudinov, Reducing the dimensionality of data with neural networks, 2006]



Deep representations

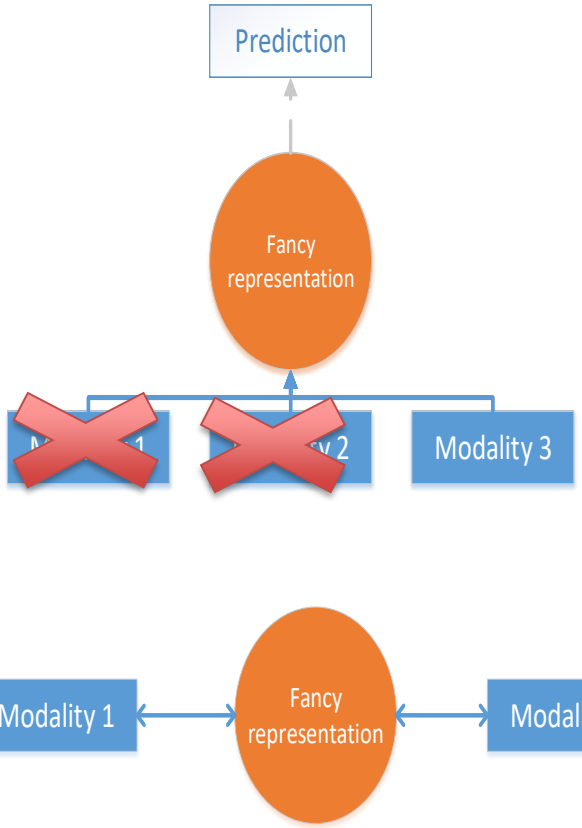
- What can we do with them?
- Compression
  - Can work better than PCA
  - [Hinton and Salatkhudinov, Reducing the dimensionality of data with neural networks, 2006]
- Discarding the decoder and using the middle layer as a representation
- Finetuning the autoencoder for a task



Multimodal representations

Multimodal representations

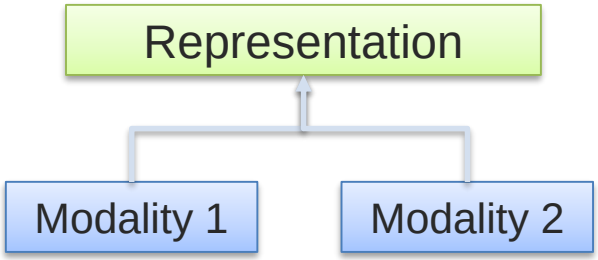
- What do we want from multi-modal representation
  - Similarity in that space implies similarity in corresponding *concepts*
  - Useful for various discriminative tasks – retrieval, mapping, fusion etc.
  - Possible to obtain in absence of one or more modalities
  - Fill in missing modalities given others (map between modalities)



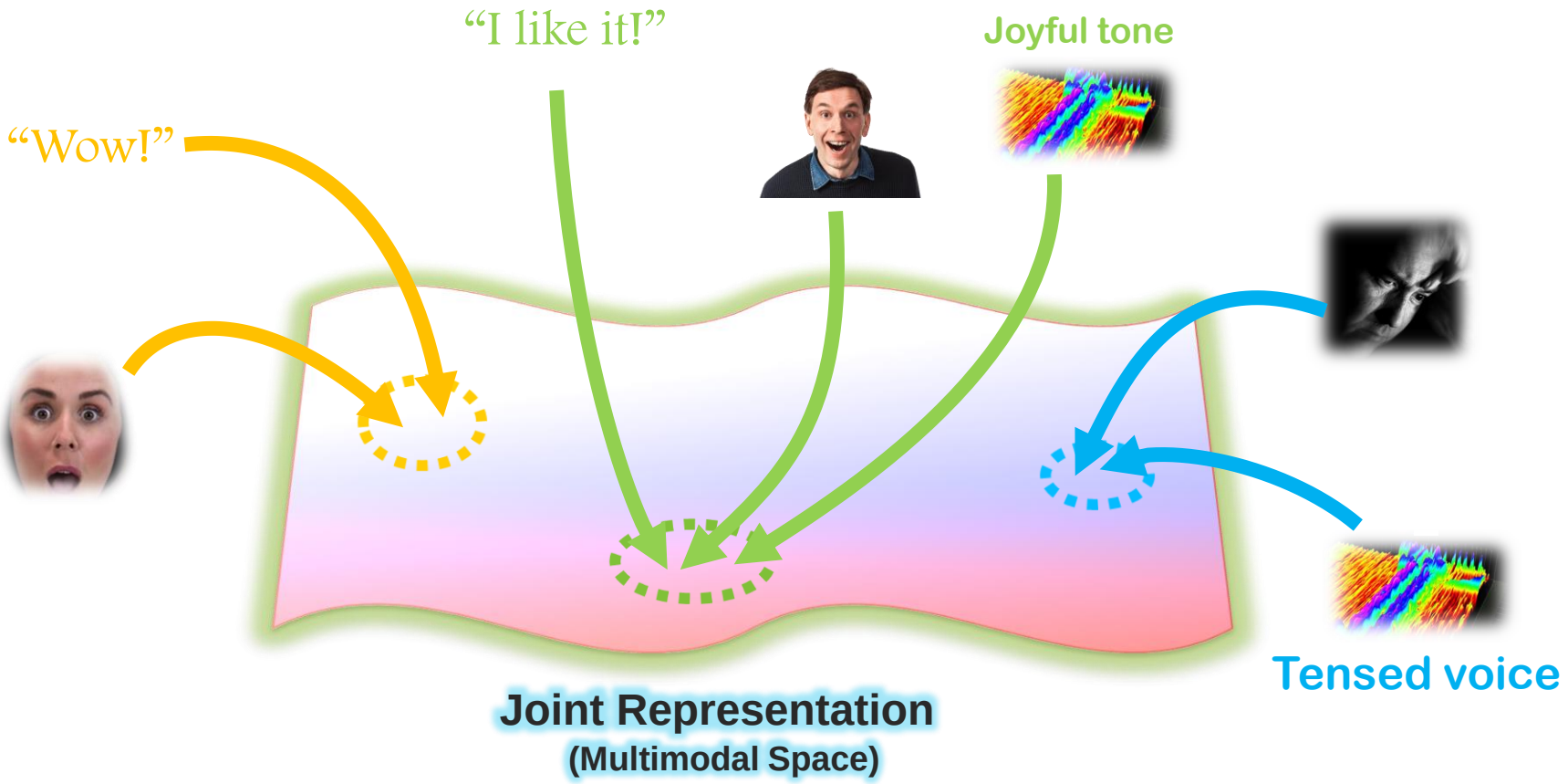
Core Challenge: Multimodal Representation

Definition: Learning how to represent and summarize multimodal data in away that exploits the complementarity and redundancy.

Ⓐ Joint representations:



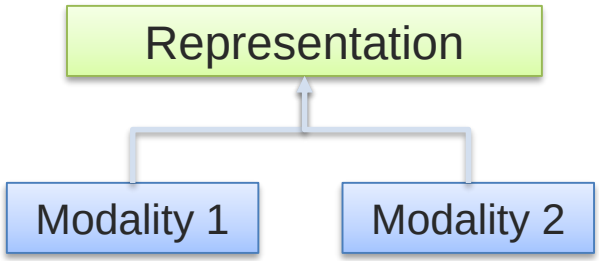
Joint Multimodal Representation



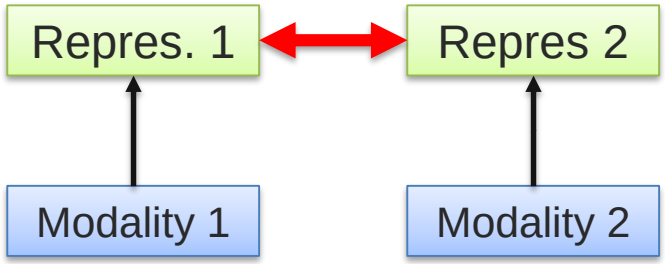
Core Challenge 1: Representation

Definition: Learning how to represent and summarize multimodal data in away that exploits the complementarity and redundancy.

A Joint representations:



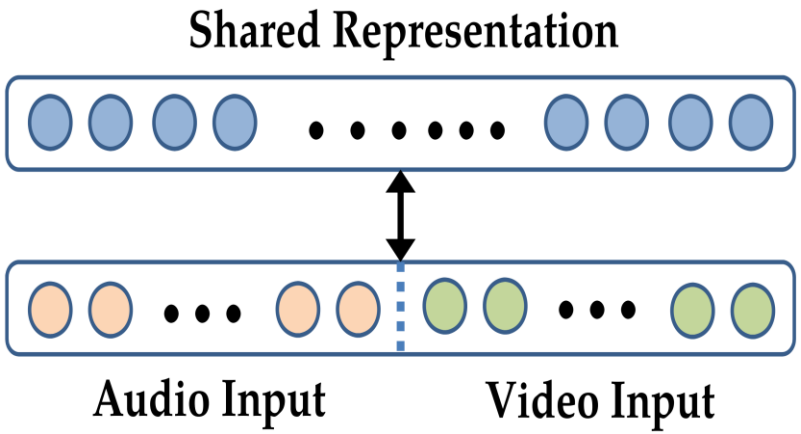
B Coordinated representations:



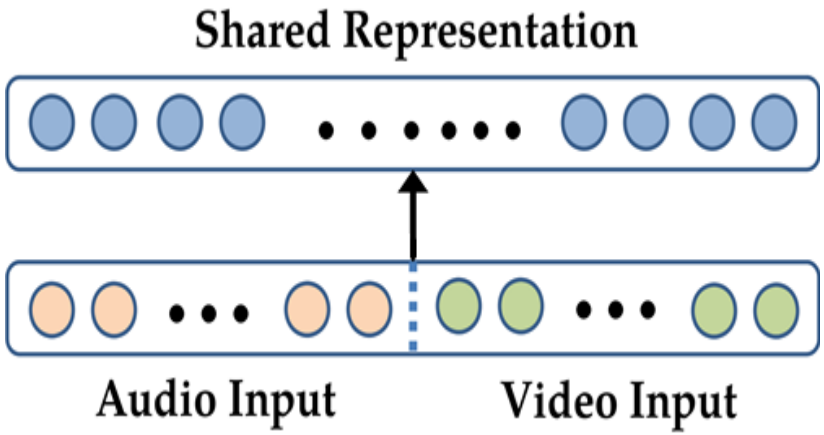
Joint representations

Shallow multimodal representations

- Want deep multimodal representations
  - Shallow representations do not capture complex relationships
  - Often shared layer only maps to the shared section directly



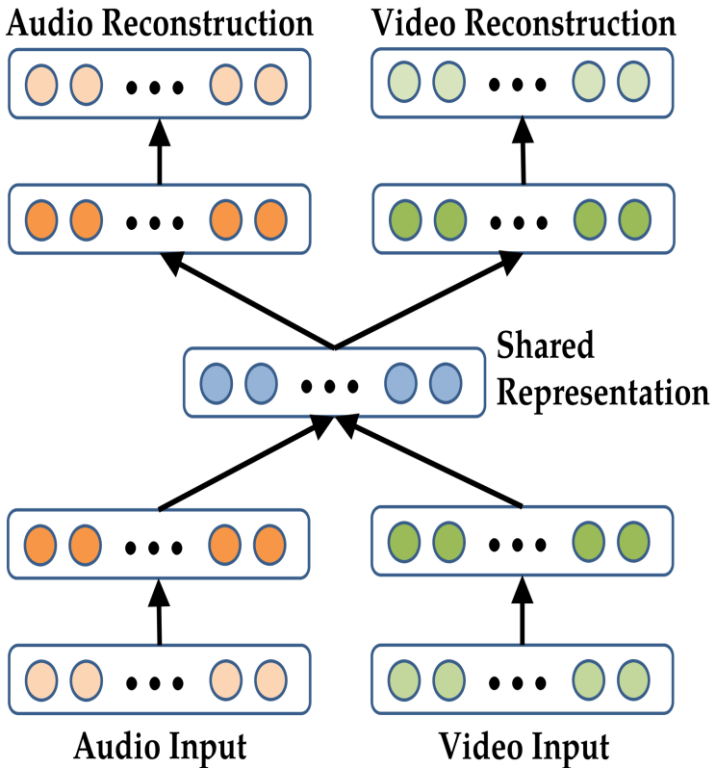
Shallow RBM



Shallow Autoencoder

Deep Multimodal autoencoders

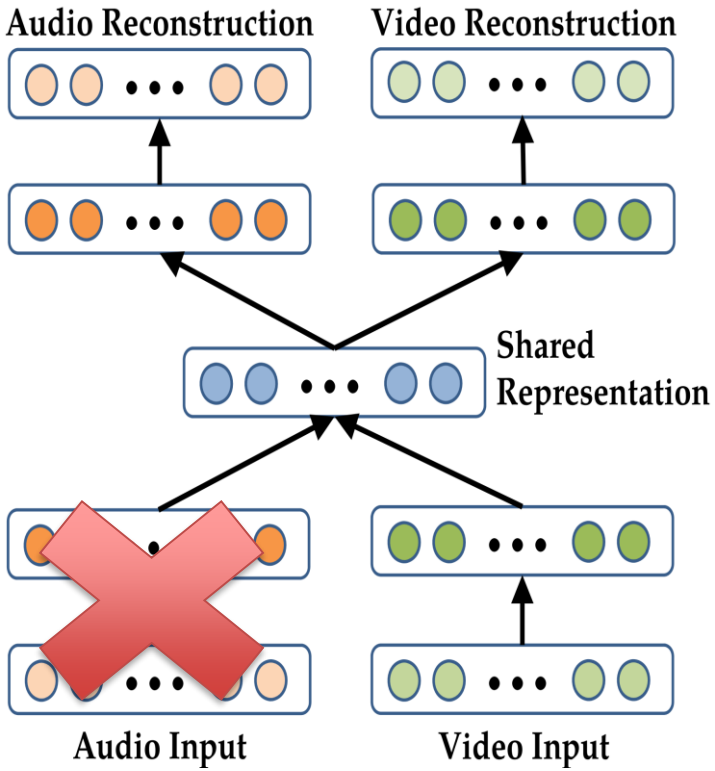
- A deep representation learning approach
- A bimodal auto-encoder
  - Used for Audio-visual speech recognition



- [Ngiam et al., Multimodal Deep Learning, 2011]

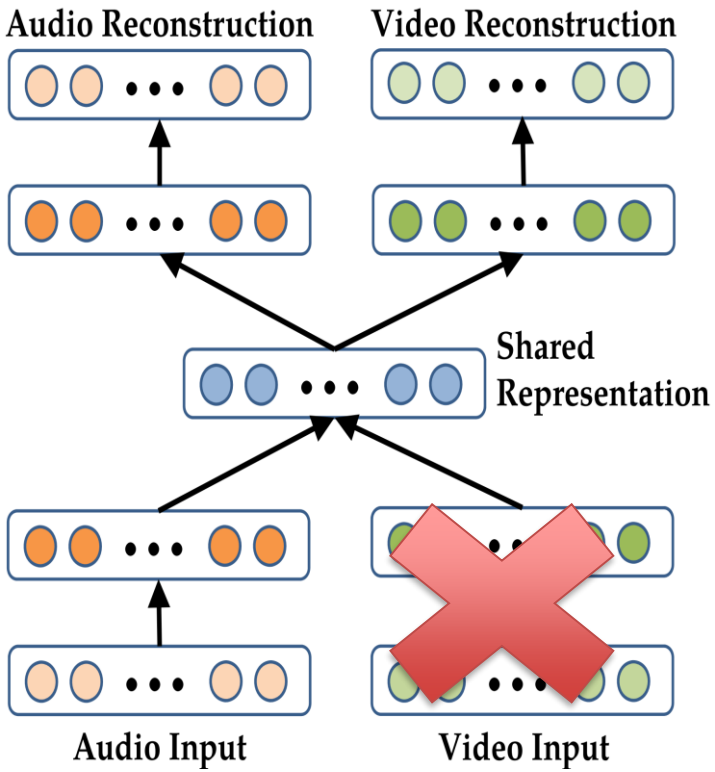
Deep Multimodal autoencoders - training

- Individual modalities can be pre-trained
  - Denoising Autoencoders
- To train the model to reconstruct the other modality
  - Use both
  - Remove audio



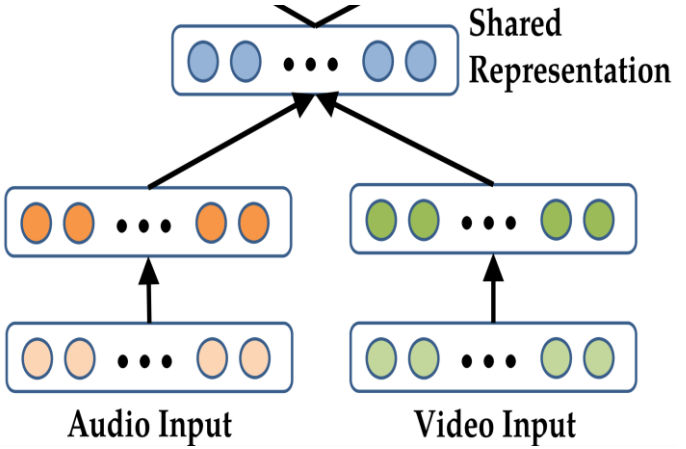
Deep Multimodal autoencoders - training

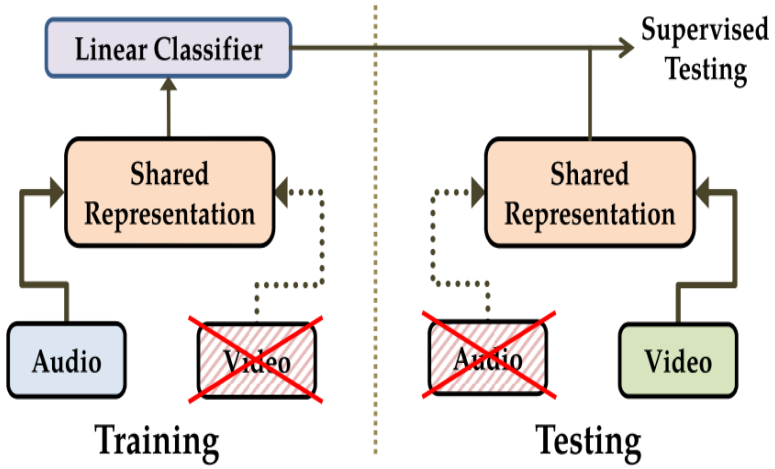
- Individual modalities can be pretrained
  - RBMs
  - Denoising Autoencoders
- To train the model to reconstruct the other modality
  - Use both
  - Remove audio
  - Remove video



Deep Multimodal autoencoders

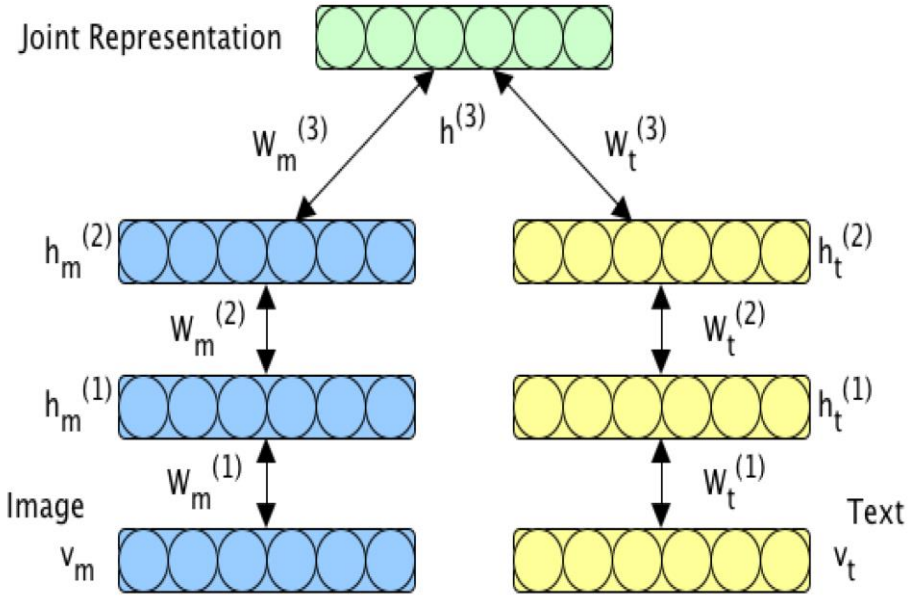
- Can now discard the decoder and use it for the AVSR task
- Interesting experiment
  - "Hearing to see"





Deep Multimodal Boltzmann machines

- Generative model
- Individual modalities trained like a DBN
- Multimodal representation trained using Variational approaches
- Used for image tagging and cross-media retrieval
- Reconstruction of one modality from another is a bit more “natural” than in autoencoder representation
- Can actually sample text and images

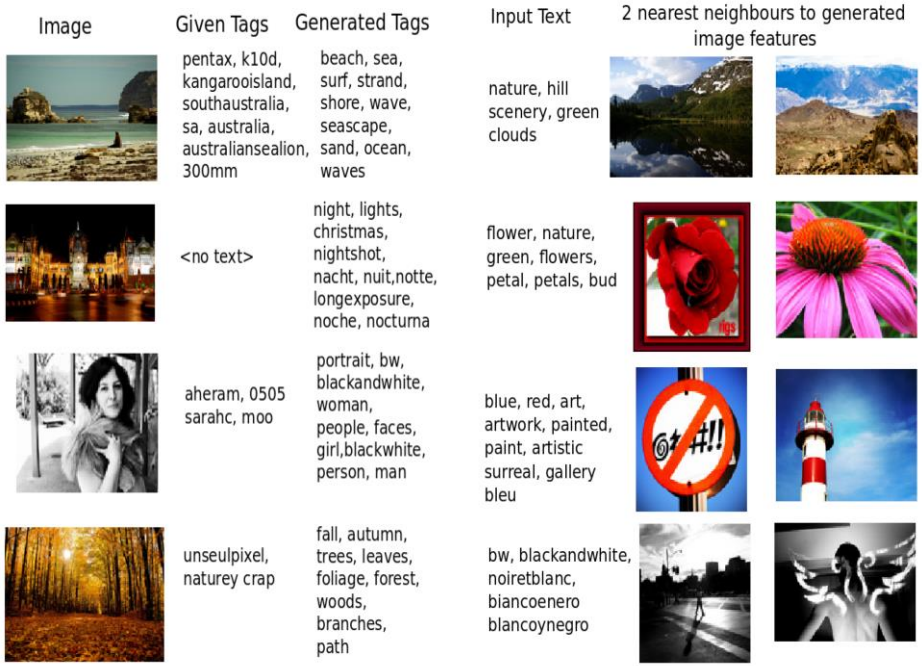


- [Srivastava and Salakhutdinov, Multimodal Learning with Deep Boltzmann Machines, 2012, 2014]

Deep Multimodal Boltzmann machines

- Pre-training on unlabeled data helps
- Can use generative models

Model	MAP	Prec@50
Random	0.124	0.124
SVM (Huiskes et al., 2010)	0.475	0.758
LDA (Huiskes et al., 2010)	0.492	0.754
DBM	0.526 ± 0.007	0.791 ± 0.008
DBM (using unlabelled data)	$\mathbf{0.585} \pm 0.004$	$\mathbf{0.836} \pm 0.004$



- Code is available
  - http://www.cs.toronto.edu/~nitish/multimodal/

Deep Multimodal Boltzmann Machines

- Text information can help visual predictions!

Model	MAP	Prec@50
Image LDA (Huiskes et al., 2010)	0.315	-
Image SVM (Huiskes et al., 2010)	0.375	-
Image DBN	0.463 ± 0.004	0.801 ± 0.005
Image DBM	0.469 ± 0.005	0.803 ± 0.005
Multimodal DBM (generated text)	$\mathbf{0.531 \pm 0.005}$	$\mathbf{0.832 \pm 0.004}$

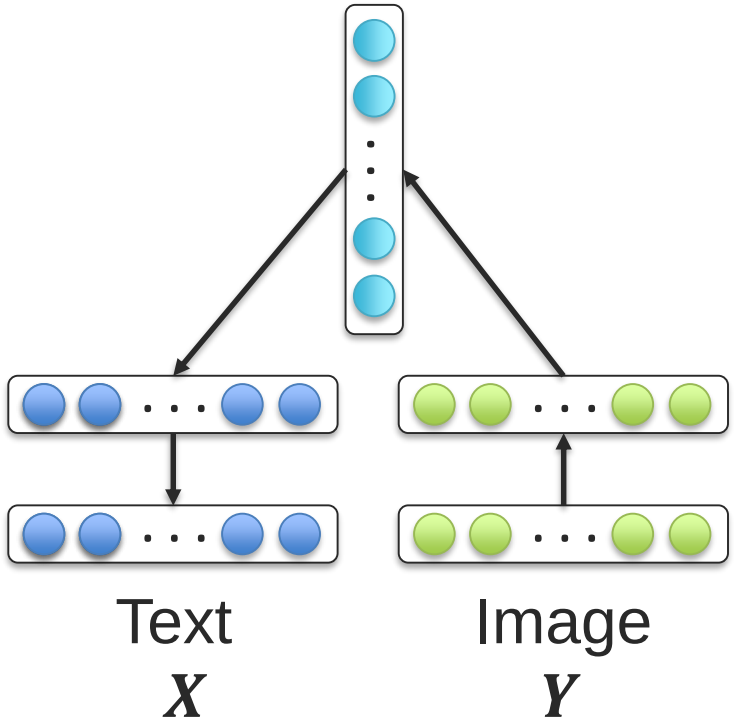
Comparing deep multimodal representations

- Difference between them and the RBMs and the autoencoders
- Overall very similar behavior

Model	DBN	DAE	DBM
Logistic regression on joint layer features	0.599 ± 0.004	0.600 ± 0.004	0.609 ± 0.004
Sparsity + Logistic regression on joint layer features	0.626 ± 0.003	0.628 ± 0.004	0.631 ± 0.004
Sparsity + discriminative fine-tuning	0.630 ± 0.004	0.630 ± 0.003	0.634 ± 0.004
Sparsity + discriminative fine-tuning + dropout	0.638 ± 0.004	0.638 ± 0.004	$\mathbf{0.641 \pm 0.004}$

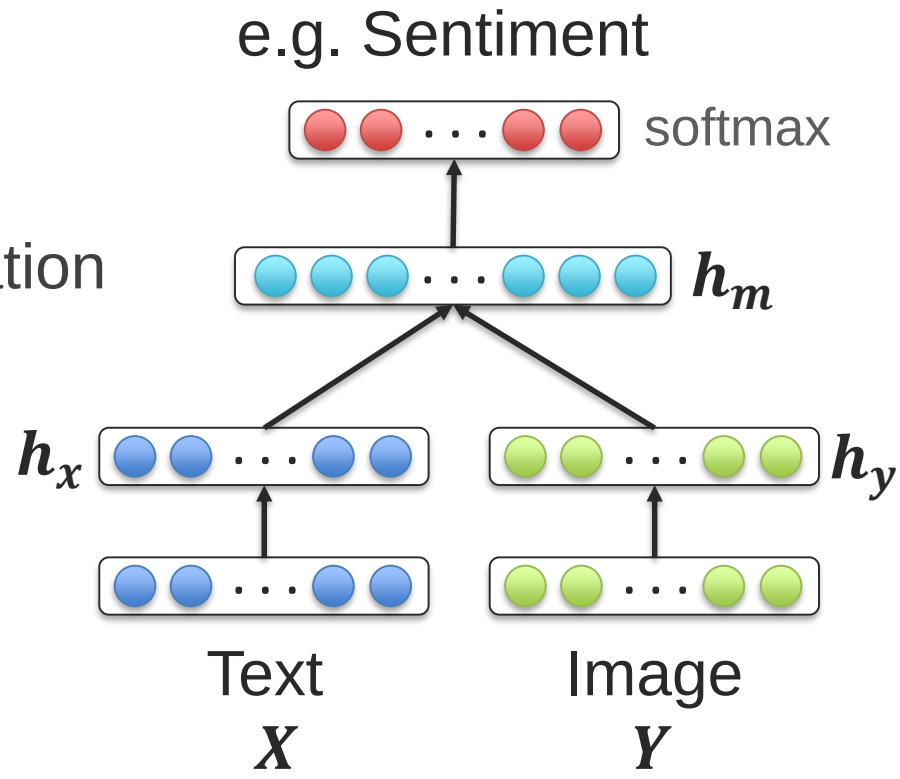
Multimodal Encoder-Decoder

- Visual modality often encoded using CNN
- Language modality will be decoded using LSTM
  - A simple multilayer perceptron will be used to translate from visual (CNN) to language (LSTM)



Multimodal Joint Representation

- For supervised learning tasks
- Joining the unimodal representations:
  - Simple concatenation
  - Element-wise multiplication or summation
  - Multilayer perceptron
- How to explicitly model both unimodal and bimodal interactions?



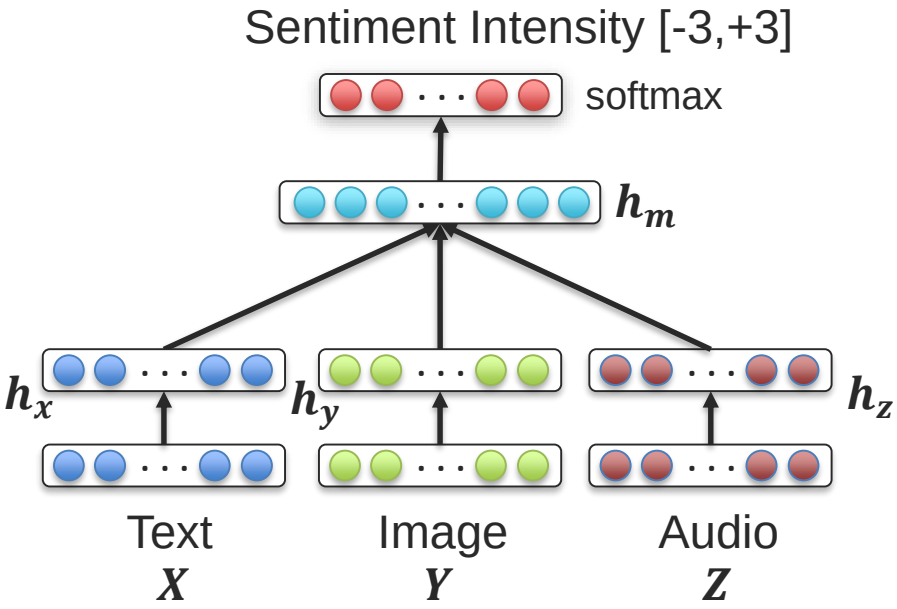
Multimodal Sentiment Analysis

MOSI dataset (Zadeh et al, 2016)

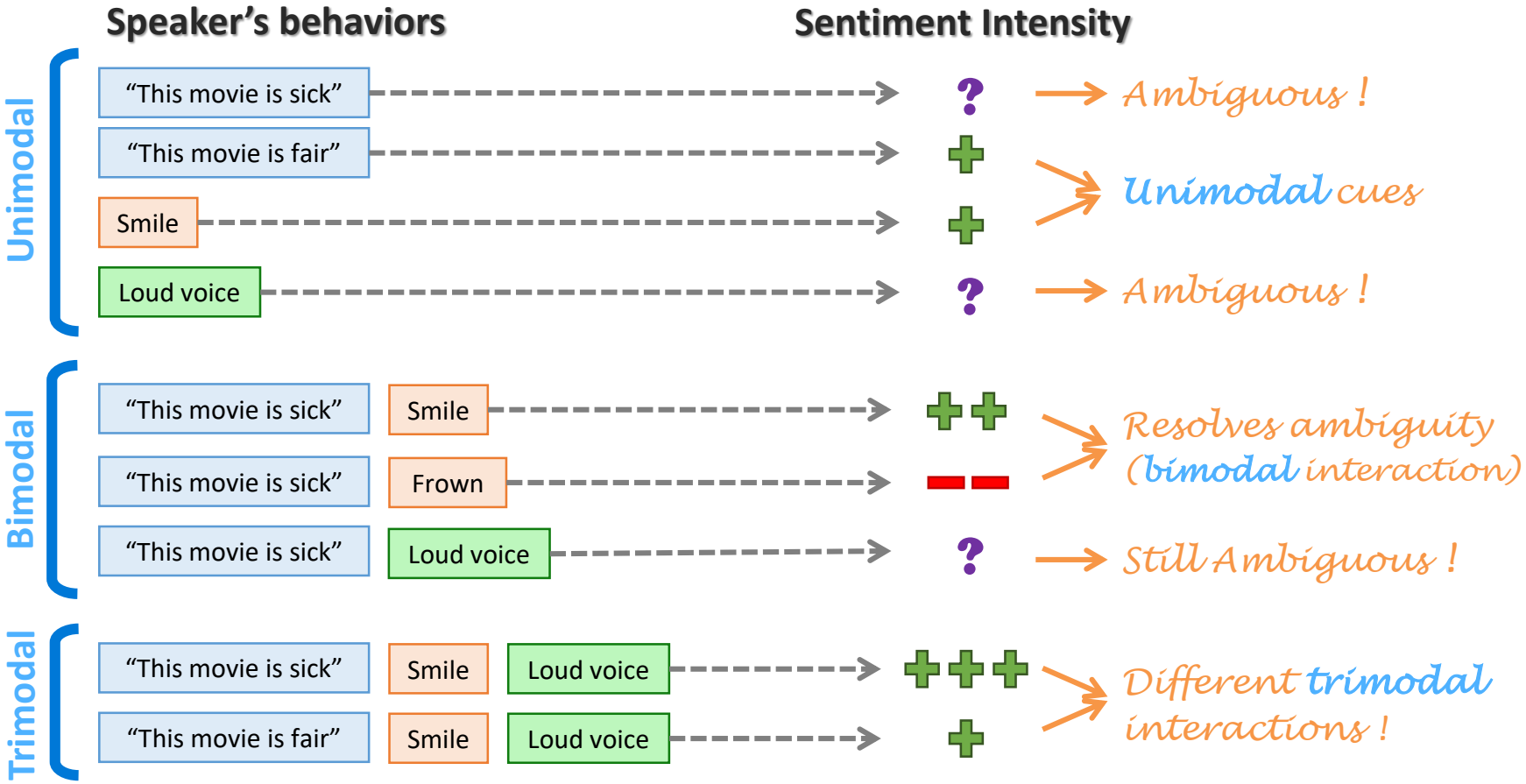
- 2199 subjective video segments
- Sentiment intensity annotations
- 3 modalities: text, video, audio

Multimodal joint representation:

$$h_m = f(W \cdot [h_x, h_y, h_z])$$



Unimodal, Bimodal and Trimodal Interactions

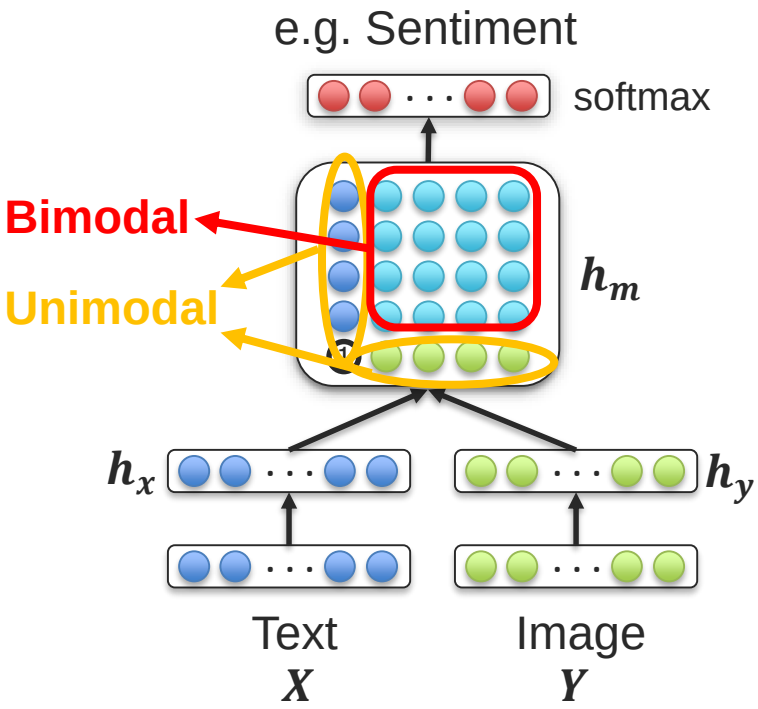


Multimodal Tensor Fusion Network (TFN)

Models both unimodal and bimodal interactions:

$$h_m = \begin{bmatrix} h_x \\ 1 \end{bmatrix} \otimes \begin{bmatrix} h_y \\ 1 \end{bmatrix} = \begin{bmatrix} h_x \\ 1 \end{bmatrix} \otimes h_y \\ h_y \end{bmatrix}$$

Important !



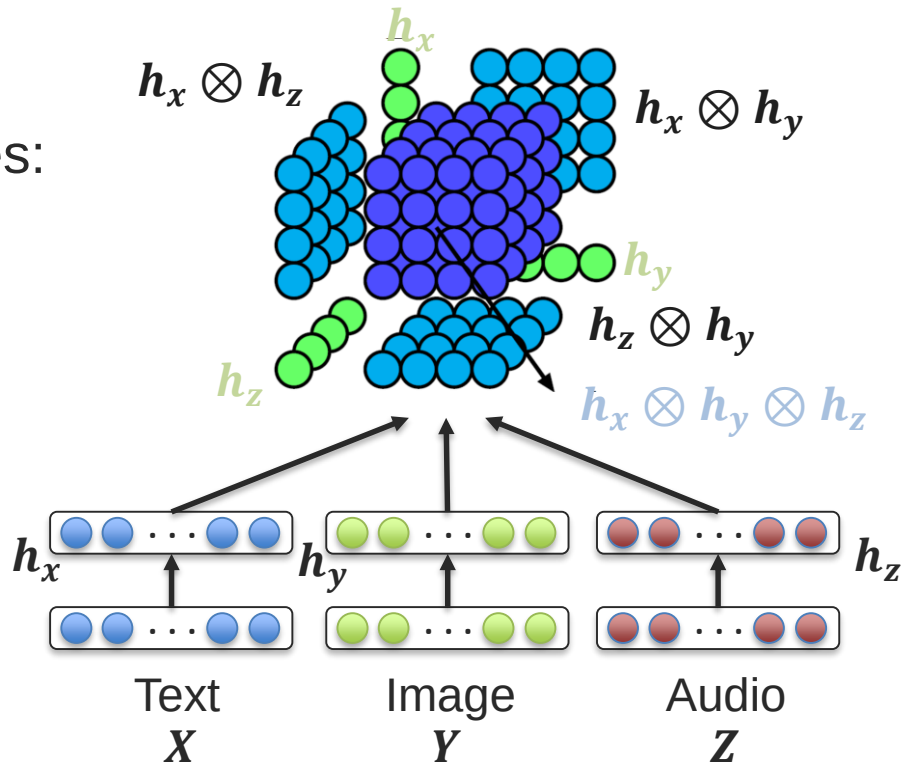
[Zadeh, Jones and Morency, **EMNLP 2017**]

Multimodal Tensor Fusion Network (TFN)

Can be extended to three modalities:

$$h_m = \begin{bmatrix} h_x \\ 1 \end{bmatrix} \otimes \begin{bmatrix} h_y \\ 1 \end{bmatrix} \otimes \begin{bmatrix} h_z \\ 1 \end{bmatrix}$$

Explicitly models unimodal, bimodal and trimodal interactions !



[Zadeh, Jones and Morency, **EMNLP 2017**]

Experimental Results – MOSI Dataset

Multimodal Baseline	Binary		5-class	Regression	
	Acc(%)	F1	Acc(%)	MAE	r
Random	50.2	48.7	23.9	1.88	-
C-MKL	73.1	75.2	35.3	-	-
SAL-CNN	73.0	-	-	-	-
SVM-MD	71.6	72.3	32.0	1.10	0.53
RF	71.4	72.1	31.9	1.11	0.51
TFN	**77.1**	**77.9**	**42.0**	**0.87**	**0.70**
Human	85.7	87.5	53.9	0.71	0.82
Δ^{SOTA}	↑ 4.0	↑ 2.7	↑ 6.7	↓ 0.23	↑ 0.17

Improvement over State-Of-The-Art

Baseline	Binary		5-class	Regression	
	Acc(%)	F1	Acc(%)	MAE	r
$\text{TFN}_{language}$	74.8	75.6	38.5	0.99	0.61
TFN_{visual}	66.8	70.4	30.4	1.13	0.48
$\text{TFN}_{acoustic}$	65.1	67.3	27.5	1.23	0.36
$\text{TFN}_{bimodal}$	75.2	76.0	39.6	0.92	0.65
$\text{TFN}_{trimodal}$	74.5	75.0	38.9	0.93	0.65
$\text{TFN}_{notrimodal}$	75.3	76.2	39.7	0.919	0.66
TFN	**77.1**	**77.9**	**42.0**	**0.87**	**0.70**
TFN_{early}	75.2	76.2	39.0	0.96	0.63