Language Technologies Institute

Carnegie Mellon University

Advanced Multimodal Machine Learning

Lecture 5.1: Unsupervised learning and Multimodal representations

Louis-Philippe Morency
Tadas Baltrusaitis

Administrative Stuff

Proposal Presentation (2/21/2017 and 2/23/2017)

- 5 minutes (about 5-10 slides)
- All team members should be involved in the presentation
- Will receive feedback from instructors and other students
  - 1-2 minutes between presentations reserved for written feedback
- Main presentation points (similar to pre-proposal)
  - General research problem and motivation
  - Dataset and input modalities
  - Multimodal challenges and prior work

Project Proposal Report – Due on 3/5/17

- Part 1 (updated version of your pre-proposal)
  - **Research problem:**
    - Describe and motivate the research problem
    - Define in generic terms the main computational challenges
  - **Dataset and Input Modalities:**
    - Describe the dataset(s) you are planning to use for this project.
    - Describe the input modalities and annotations available in this dataset.

Project Proposal Report – Due on 3/5/17

- Part 2
  - **Related Work:**
    - Include 12-15 paper citations which give an overview of the prior work
    - Present in more details the 3-4 research papers most related to your work
  - **Research Challenges and Hypotheses:**
    - Describe your specific challenges and/or research hypotheses
    - Highlight the novel aspect of your proposed research

Project Proposal Report – Due on 3/5/17

- Part 3 – (teams of 2 members can pick either one)
  - **Language Modality Exploration:**
    - Explore neural language models on your dataset (using Keras/Theano)
    - Train at least two different language models (e.g., using SimpleRNN, GRU or LSTM) on your dataset and compare their perplexity.
    - Include qualitative examples of successes and failure cases.
  - **Visual Modality Exploration:**
    - Explore pre-trained Convolutional Neural Networks (CNNs) on your dataset
    - Load a pre-existing CNN model trained for object recognition (e.g., AlexNet or VGG-Net) and process your test images.
    - Extract features at different network layers in the network and visualize them (using t-sne visualization) with overlaid class labels with different colors.

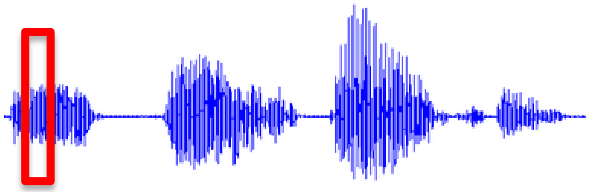
Objectives of today's class

- Audio representations
  - Hand-crafted (MFCC) and learned (Deep Belief Nets, Deep Neural Networks)
- 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
  - Visual semantic embeddings

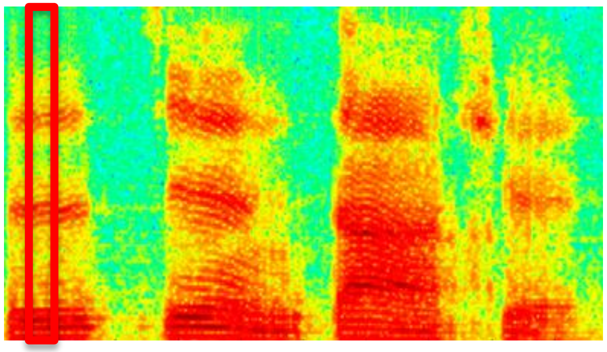
Audio representations

Audio representation

- Audio frames in a window (this is our input)
  - Can extract a spectrogram (lowest level)
- Low level features like MFCC
- Higher-level features also exist (specifically for human voice)
  - Prosody
  - Voice quality
- This can be used for speech recognition, affect and sentiment analysis, music analysis etc.



- Sampling rates: 8~96kHz
- Bit depth: 8, 16 or 24 bits
- Time window size: 20ms
  - Offset: 10ms



Spectogram

Audio representation for speech recognition

- Speech recognition systems historically much more complex than vision systems
- Require a lot of moving parts
  - Phoneme detectors
  - Language models
  - Vocabularies
- Large breakthrough of using representation learning instead of hand-crafted features
  - [Hinton et al., Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups, 2012]
- Most work exploited large amounts of unsupervised data for model pre-training
- The field of ASR was largely static for some years up to then
- A huge boost in performance (up to 30% on some datasets)

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

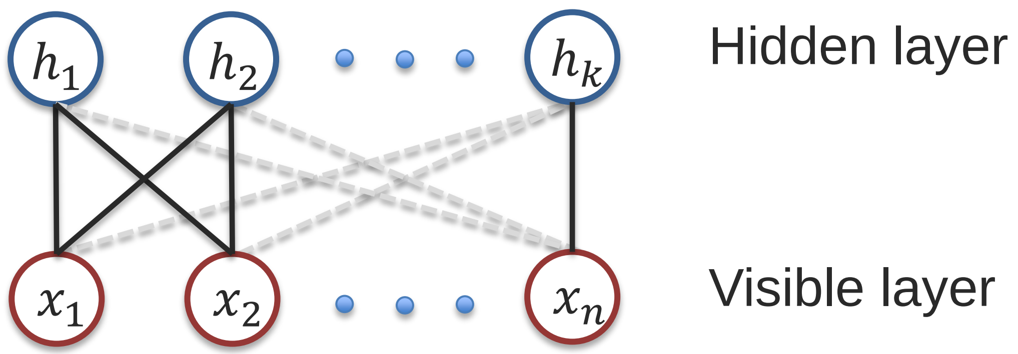
Unsupervised representation learning

- 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

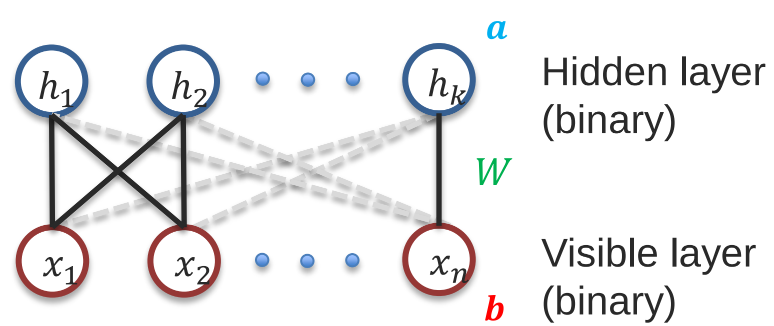
$$Z = \sum_{\boldsymbol{x}} \sum_{\boldsymbol{h}} \exp(-\mathrm{E}(\boldsymbol{x}, \boldsymbol{h}; \theta))$$

Normalization function so that the probabilities sum to one

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

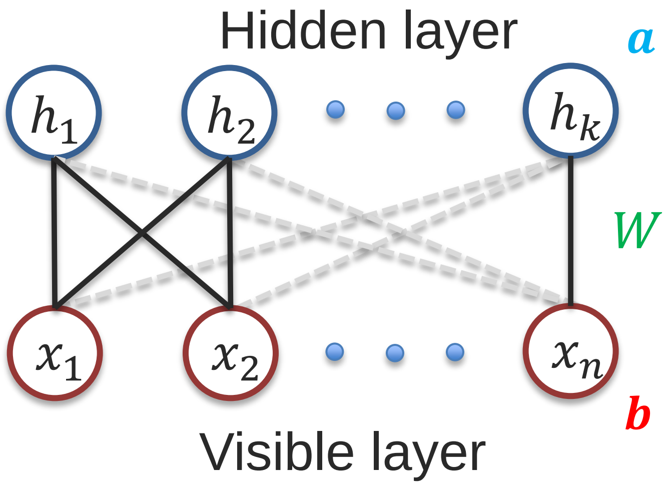
$$\mathrm{E} = -\underbrace{\sum_i \sum_j w_{i,j} x_i h_j}_{\text{Interaction term}} - \underbrace{\sum_i b_i x_i - \sum_j a_j h_j}_{\text{Bias terms}}$$

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



RBM inference (have a trained θ)

- For inference
  - $p\left(h_j = 1 \middle| \boldsymbol{x}; \theta\right) = \sigma\left(\sum_i x_i w_{ij} + a_j\right)$,
  - $p(x_i = 1|\boldsymbol{h}; \theta) = \sigma\left(\sum_j h_j w_{ij} + b_i\right)$
  - 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 $\boldsymbol{x}$
- Can show this by factorizing the terms
- 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:
  - $\mathrm{argmin}_\theta \left[\sum_t -\log\left(p(\boldsymbol{x}^{(t)};\theta)\right)\right]$, where t is a data sample
  - sum across all samples
  - minimizing negative log likelihood instead of maximizing the likelihood
- General gradient form for energy models with latent terms:

$$\frac{\partial -\log p(\boldsymbol{x}^{(t)};\theta)}{\partial\theta} = \underbrace{\mathbb{E}_{\boldsymbol{h}}\left[\frac{\partial\, \mathrm{E}(\boldsymbol{x}^{(t)},\boldsymbol{h};\theta))}{\partial\theta} \middle| \boldsymbol{x}^{(t)}\right]}_{\text{Positive phase/data term}} - \underbrace{\mathbb{E}_{\boldsymbol{h},\boldsymbol{x}}\left[\frac{\partial\, \mathrm{E}(\boldsymbol{x},\boldsymbol{h};\theta))}{\partial\theta}\right]}_{\text{Negative phase/model term}}$$

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

RBM training

- Ignoring the biases as they are easier
  - $\frac{\partial \log p(\boldsymbol{x}^{(t)};\theta)}{\partial w_{ij}} = \mathbb{E}_{\boldsymbol{h}}\left[\frac{\partial \mathrm{E}(\boldsymbol{x}^{(t)},\boldsymbol{h};\theta))}{\partial w_{ij}} \middle| \boldsymbol{x}^{(t)}\right] - \boxed{\mathbb{E}_{\boldsymbol{h},\boldsymbol{x}}\left[\frac{\partial \mathrm{E}(\boldsymbol{x},\boldsymbol{h};\theta))}{\partial w_{ij}}\right]}$
- First term is straightforward to compute

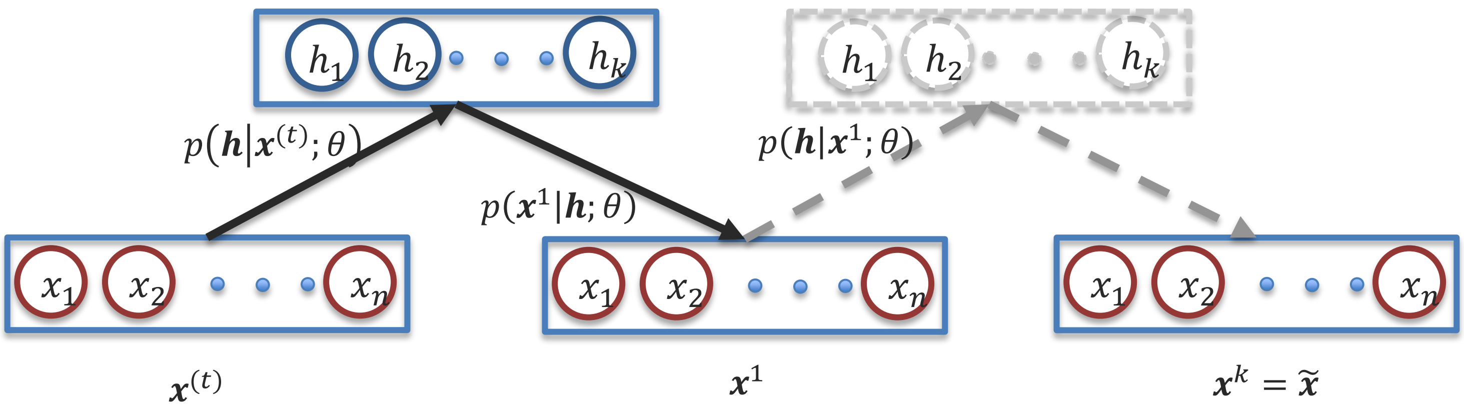
$$\frac{\partial \mathrm{E}(\boldsymbol{x},\boldsymbol{h};\theta))}{\partial w_{ij}} = -h_i x_j \Rightarrow \mathbb{E}_{\boldsymbol{h}}\left[\frac{\partial \mathrm{E}(\boldsymbol{x}^{(t)},\boldsymbol{h};\theta))}{\partial w_{ij}} \middle| \boldsymbol{x}^{(t)}\right] = \mathbb{E}_{\boldsymbol{h}}\left[-h_i x_j \middle| \boldsymbol{x}^{(t)}\right]$$
$$= \sum_{h_i \in \{0,1\}} -h_i x_j p\left(h_i \middle| \boldsymbol{x}^{(t)};\theta\right) = -x_j p\left(h_i = 1 \middle| \boldsymbol{x}^{(t)};\theta\right)$$

- Second expectation term is straightforward mathematically, but intractable computationally (too many terms to sum over)
  - Want to approximate it instead
  - Replace the expectation across $\boldsymbol{x},\boldsymbol{h}$ with a point estimate at $\tilde{\boldsymbol{x}}$

Contrastive divergence

- Instead of $\mathbb{E}_{\boldsymbol{h},\boldsymbol{x}}\left[\frac{\partial\ \mathrm{E}(\boldsymbol{x},\boldsymbol{h}))}{\partial w_{ij}}\right]$, compute $\mathbb{E}_{\boldsymbol{h}}\left[\frac{\partial\ \mathrm{E}(\widetilde{\boldsymbol{x}},\boldsymbol{h}))}{\partial w_{ij}}\,|\widetilde{\boldsymbol{x}}\right]$ as an approximation
- To Approximate computation of model term using Contrastive Divergence
  - Based on Markov Chain Monte Carlo (Gibbs) sampling

Hidden layer



Visible layer

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

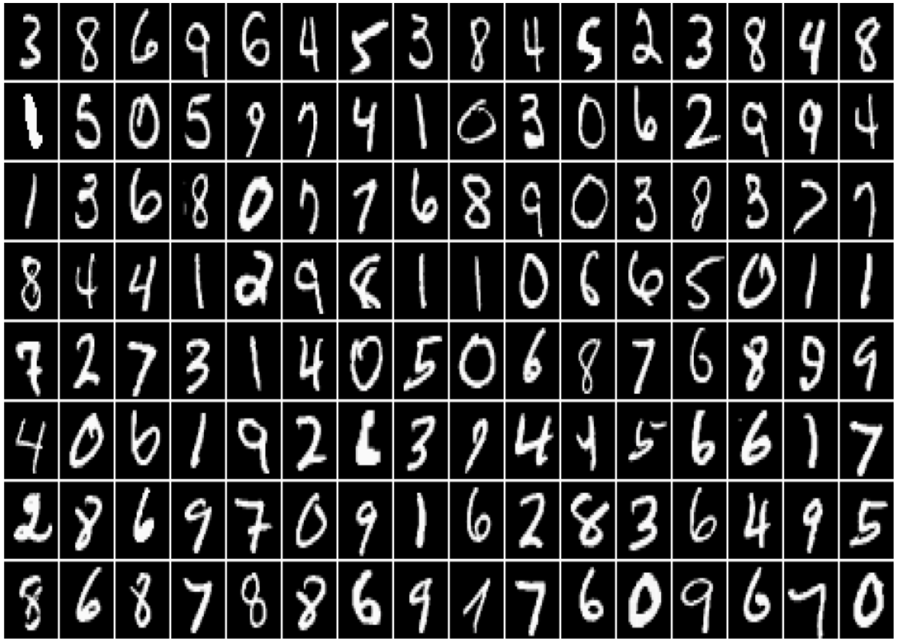
Update rule for RBMs

- Now have the update rule for parameters
- $\frac{\partial \log p(\boldsymbol{x}^{(t)};\theta)}{\partial w_{ij}} = \mathbb{E}_{\boldsymbol{h}}\left[\frac{\partial\, \mathrm{E}(\boldsymbol{x}^{(t)},\boldsymbol{h}))}{\partial w_{ij}} \middle| \boldsymbol{x}^{(t)}\right] - \mathbb{E}_{\boldsymbol{h},\boldsymbol{x}}\left[\frac{\partial\, \mathrm{E}(\boldsymbol{x},\boldsymbol{h}))}{\partial w_{ij}}\right] \cong -x_j p\left(h_i = 1 \middle| \boldsymbol{x}^{(t)}\right) + \tilde{x}_j p(h_i = 1|\widetilde{\boldsymbol{x}})$
- Still a gradient descent approach (although approximate)
  - Sampling negative phase rather than using the actual gradient
  - Similar update rules apply for the bias terms
- Can use a stochastic update
  - Using mini-batches rather than all data (current equation written for one sample)
- Extension using better sampling - Persistent Contrastive Divergence
  - [T. Tieleman, Training Restricted Boltzmann Machines using Approximations to the Likelihood Gradient, 2008]

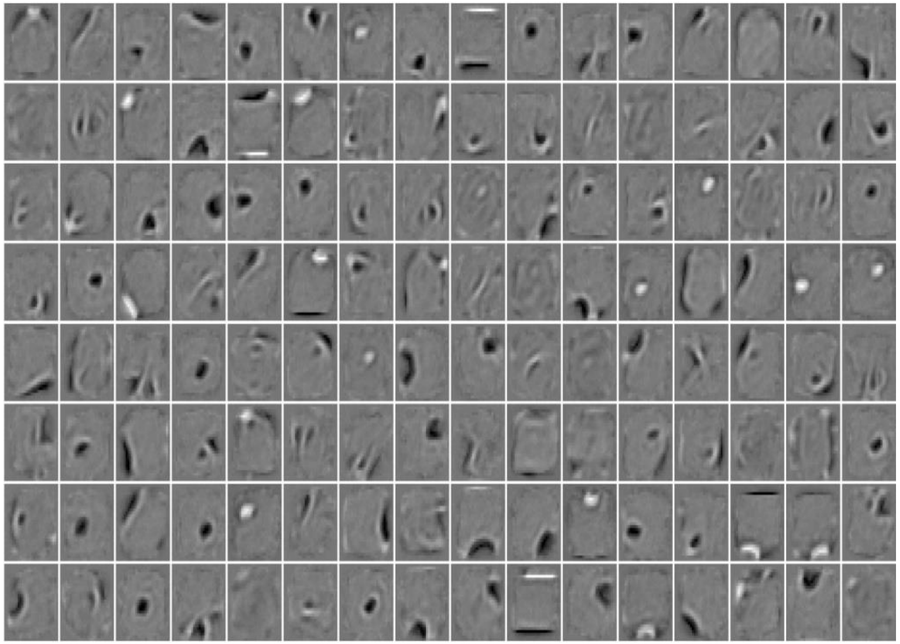
RBM extensions

- 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



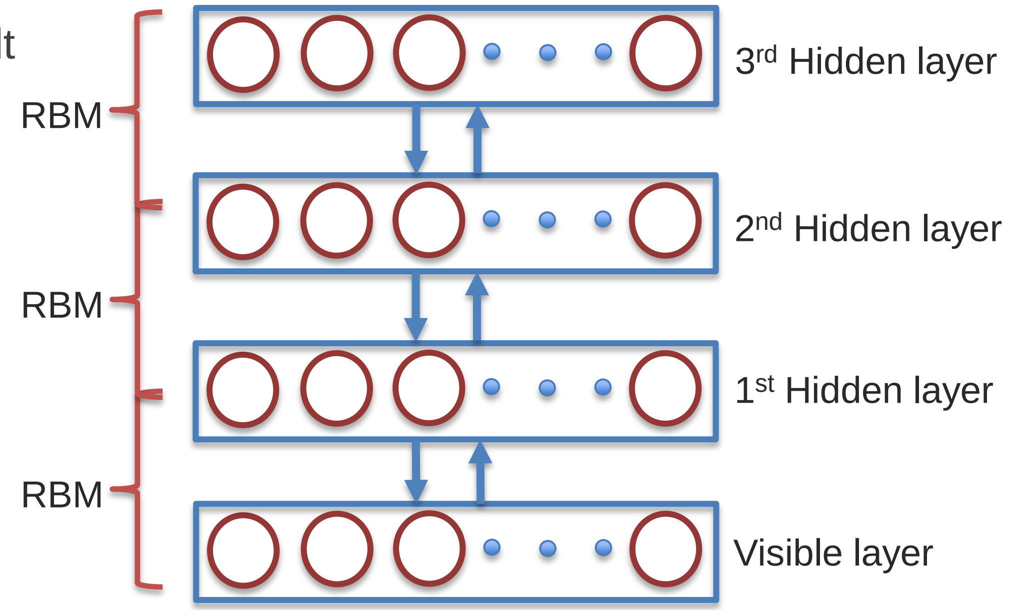
MNIST data



Learned W terms for each hidden unit

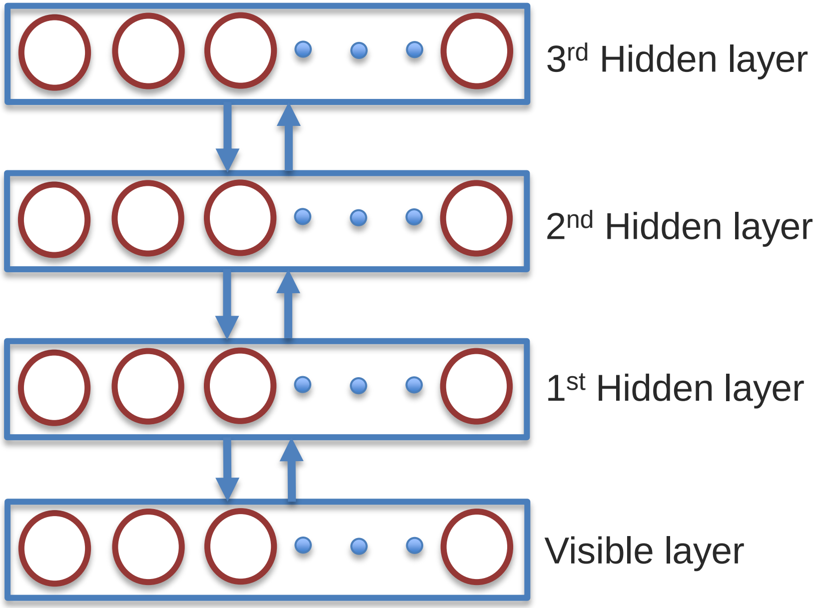
Deep Restricted Boltzmann Machines (DBMs)

- Can stack RBMs together to lead do deep versions of them
- The visible layer can be binary, Gaussian or Bernoulli
- The hidden layers are still binary
- Training fully end to end is very difficult



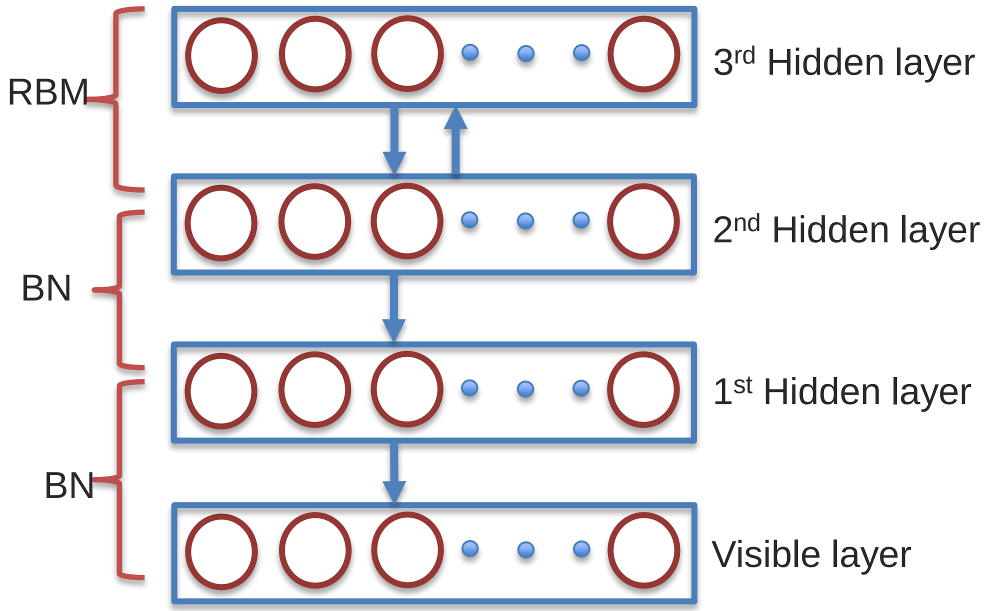
Deep Restricted Boltzmann Machines (DBMs)

- Can stack RBMs together to lead do deep versions of them
- Training fully end to end is very difficult
- Greedy layer-wise training
- Combine the RBMs layer by layer



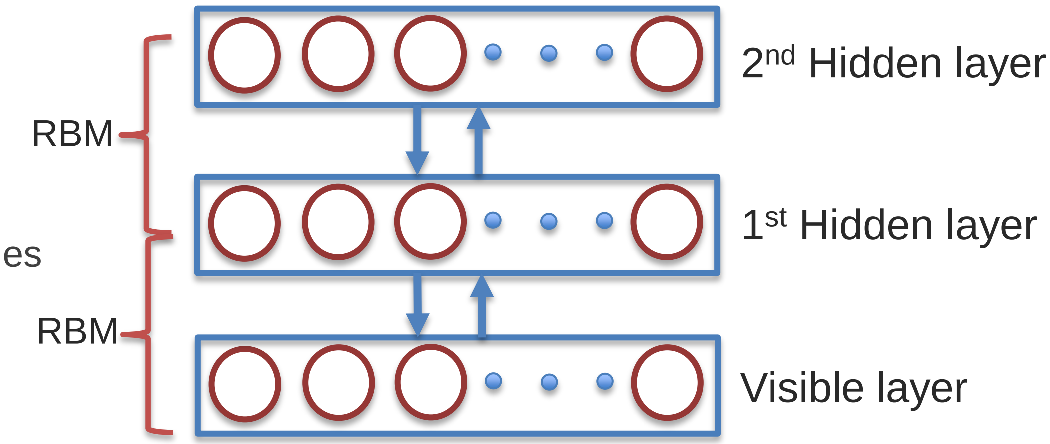
Deep Belief Networks (DBN)

- 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

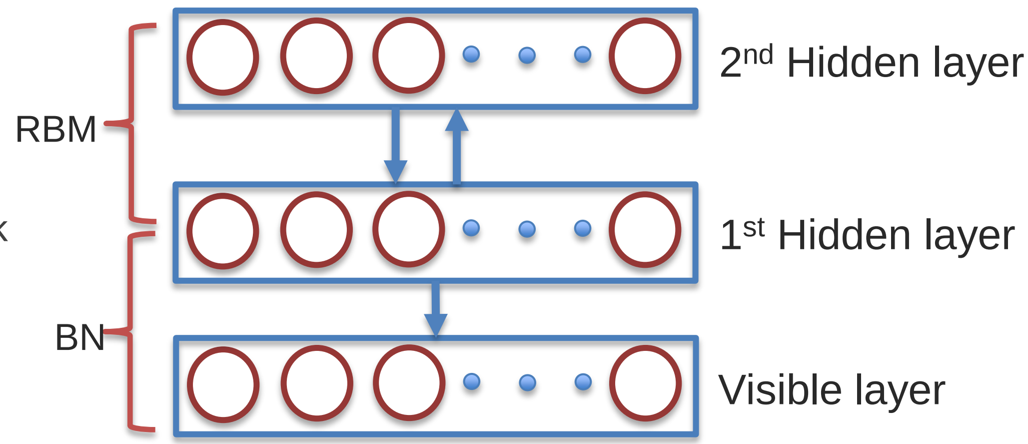


Deep Belief Networks and Deep Boltzmann Machines

- Deep Boltzmann Machine (DBM)
  - All layers are undirected
  - Relationships between layers modeled as RBMs
  - Better at modeling dependencies of lower to higher levels
  - More difficult to train



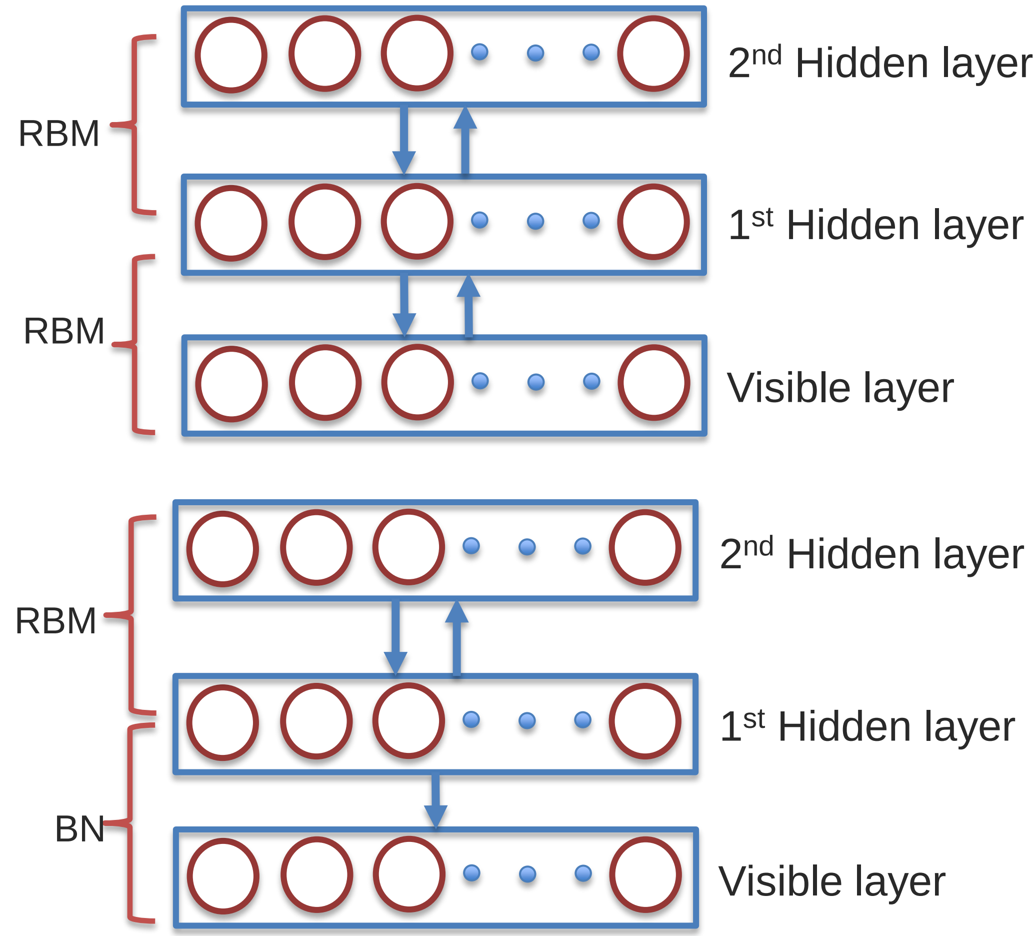
- Deep Belief Network (DBN)
  - Mixing directed and undirected interactions
  - Top two layers are always an RBM
  - Others are a Bayesian Network



Deep Belief Networks and Deep Boltzmann Machines

- Greedy layer-wise training
- Combine the RBMs layer by layer
- Difficult to train and have to use approximations and sampling
  - Variational learning methods
  - EM like
  - Optimize a lower bound rather than directly the actual log likelihood

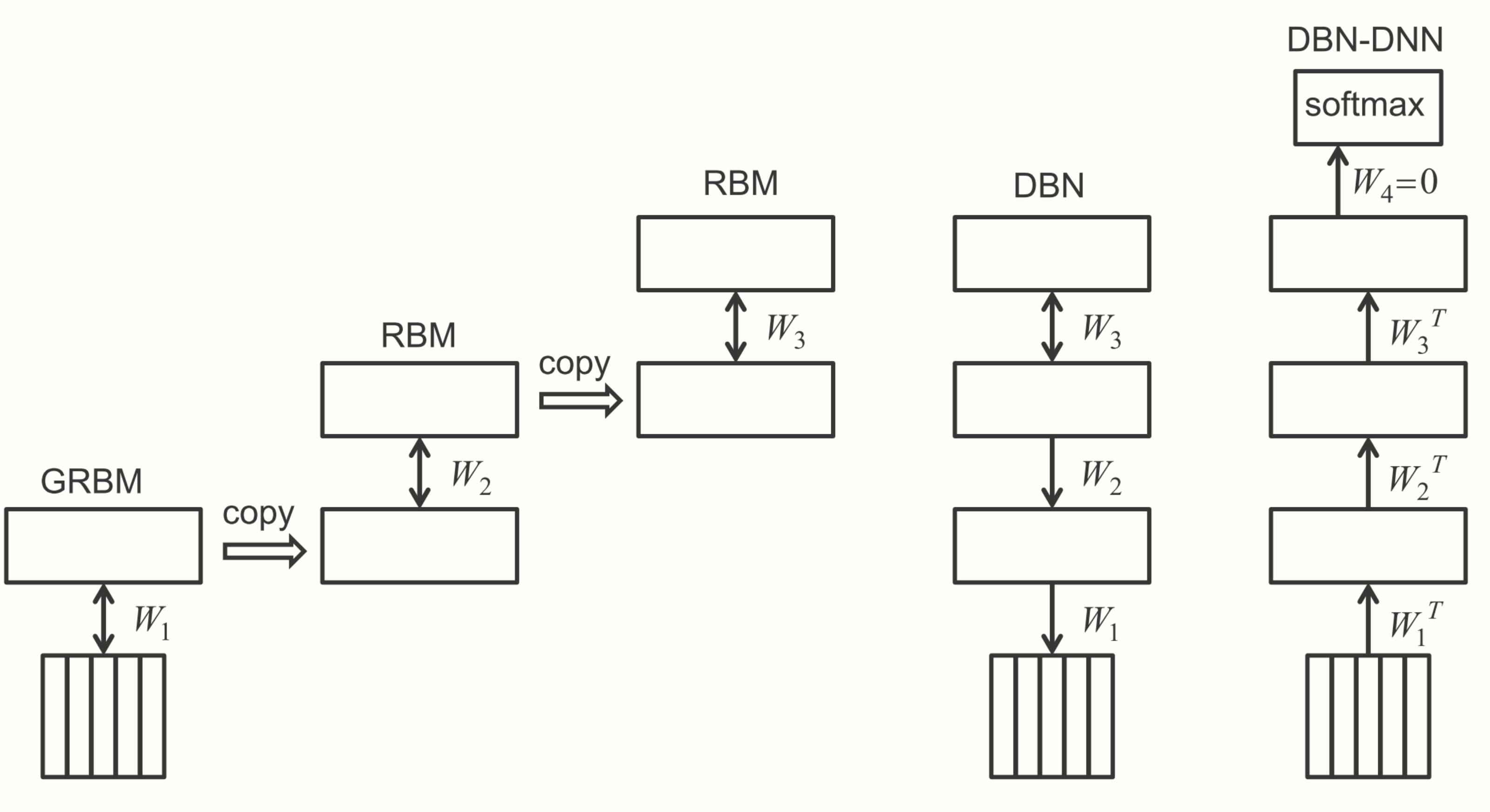
- For more details see [Salakhutdinov and Hinton, Deep Boltzmann Machines, 2009]



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]

Audio representation for speech recognition

- Feed-forward DNN for features (instead of HMM emissions)
- Now the field has moved slightly away from pre-training using DBN
  - More labeled data available
  - Better training approaches for DNN available (dropout etc.)
- Still lots of tasks where huge amounts of labeled data not available though

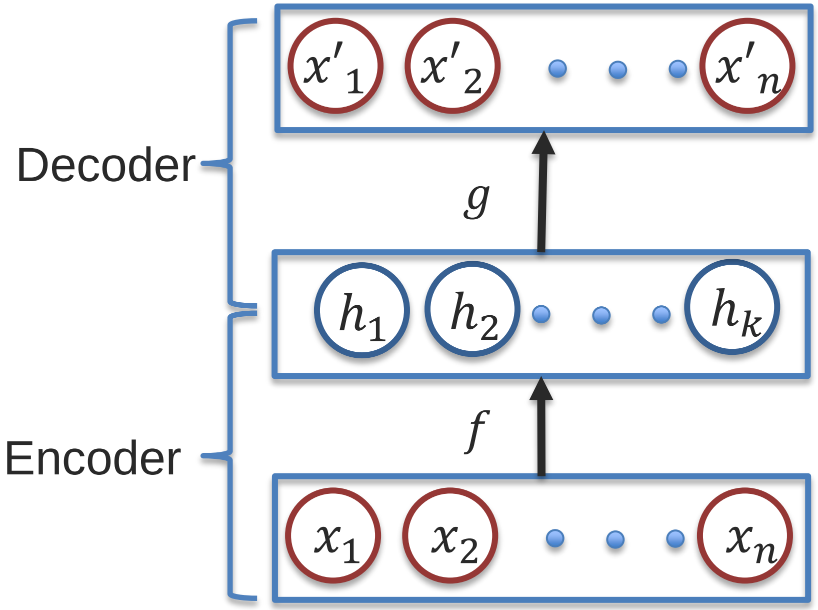
Audio representation for speech recognition

- Used MFCC but not anymore going straight from spectrogram now
  - [Deng, et al., Binary Coding of Speech Spectrograms Using a Deep Auto-encoder, 2010]
- Sometimes even using CNN on spectrograms
  - [Abdel-Hamid et al., Convolutional Neural Networks for Speech Recognition, 2014]
- RNN and LSTM now becoming popular for language models
  - Further improvement up to up to 20%
  - [Mikolov et al., Recurrent neural network based language model]
- Fully end-to-end training (a different approach)
  - [Graves and Jaitly, Towards End-to-End Speech Recognition with Recurrent Neural Networks, 2014]
  - Not as accurate yet but potentially will overtake the current designs

Autoencoders

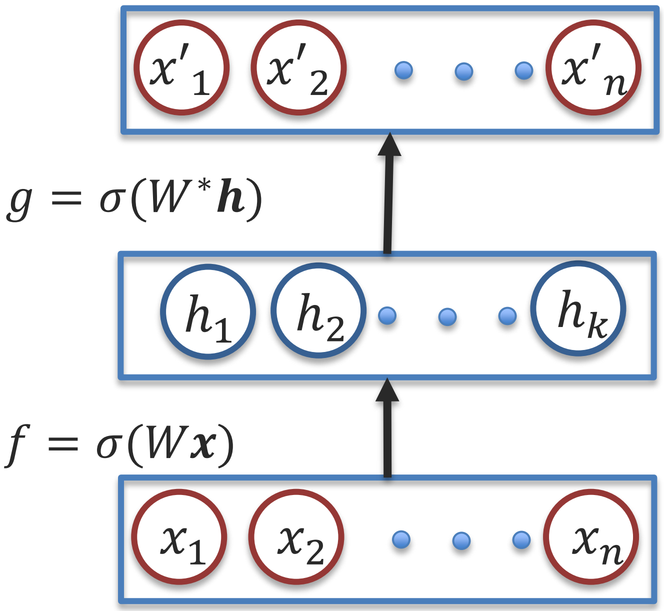
Autoencoders – an alternative to RBM

- 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



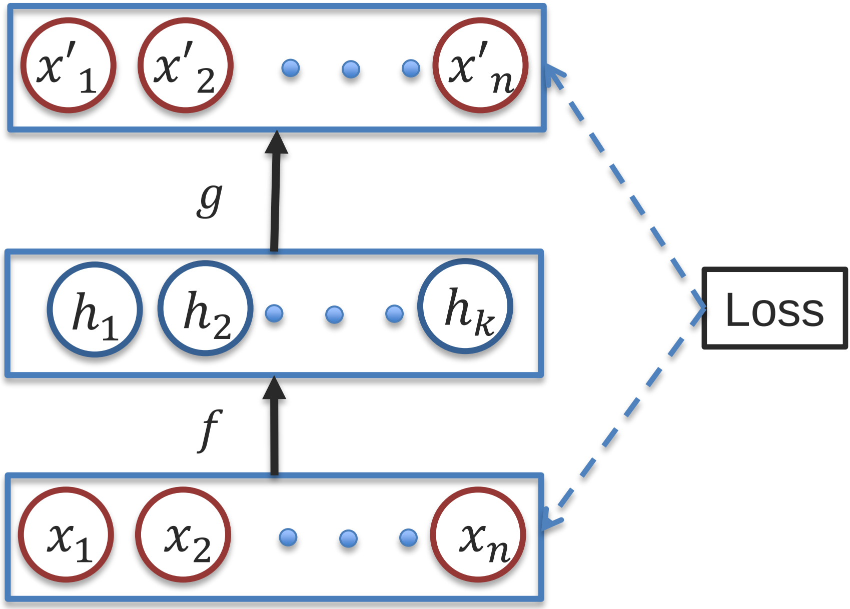
Autoencoders

- 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)



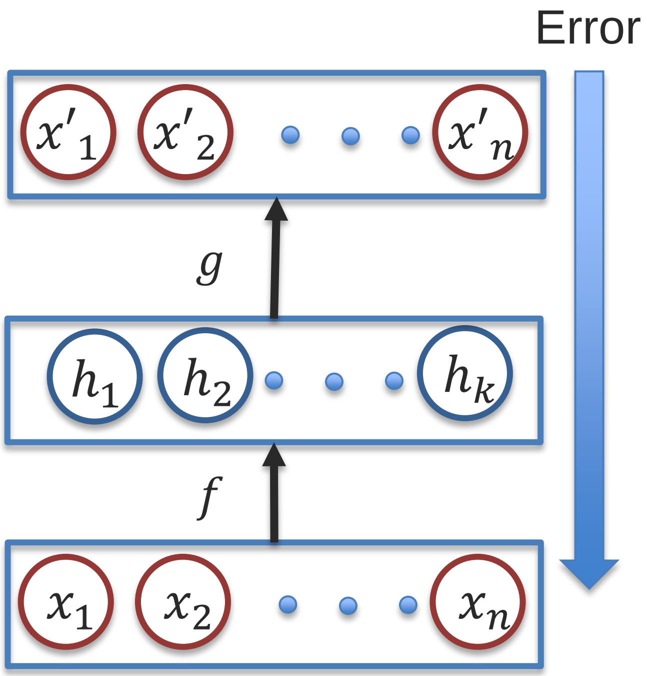
Loss function

- 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) 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

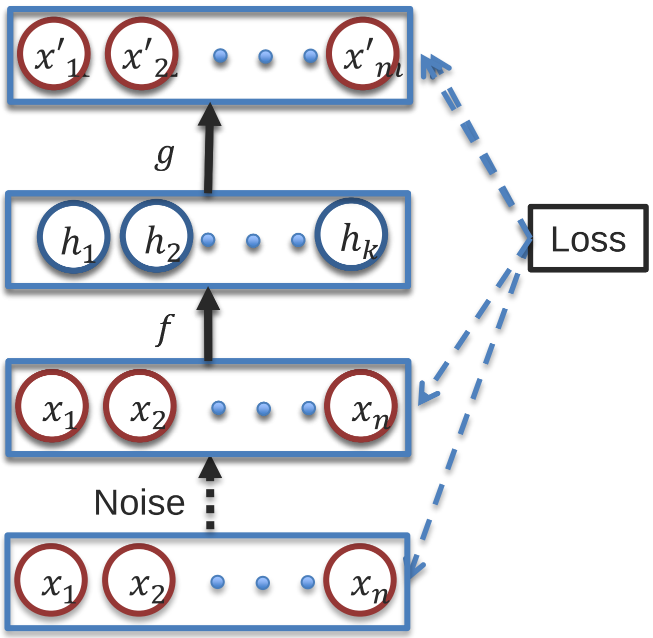


Hidden layer dimensionality

- 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

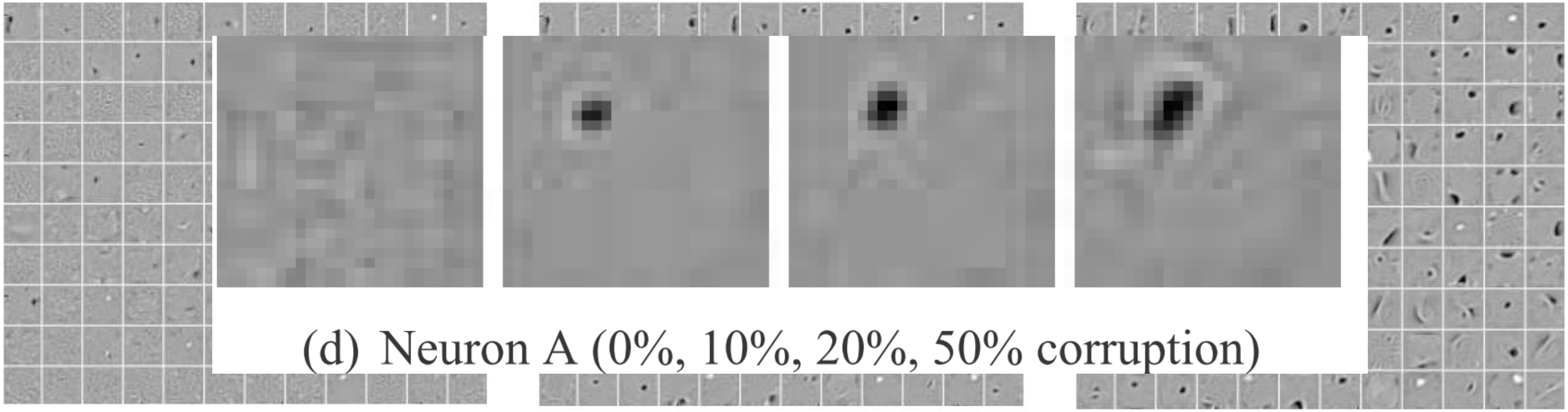


Contractive autoencoder

- A slightly different way to make the autoencoder learn a useful representation
- Intuition - changing the loss function instead of adding noise
  - Making the weight derivative with respect to input small
  - More difficult to implement and leads to similar results to Denoising autoencoders
- [Rifai et al., Contractive Auto-Encoders: Explicit Invariance During Feature Extraction, 2011]

Autoencoder vs denoising autoencoder

- MNIST data (as before)



Autoencoder

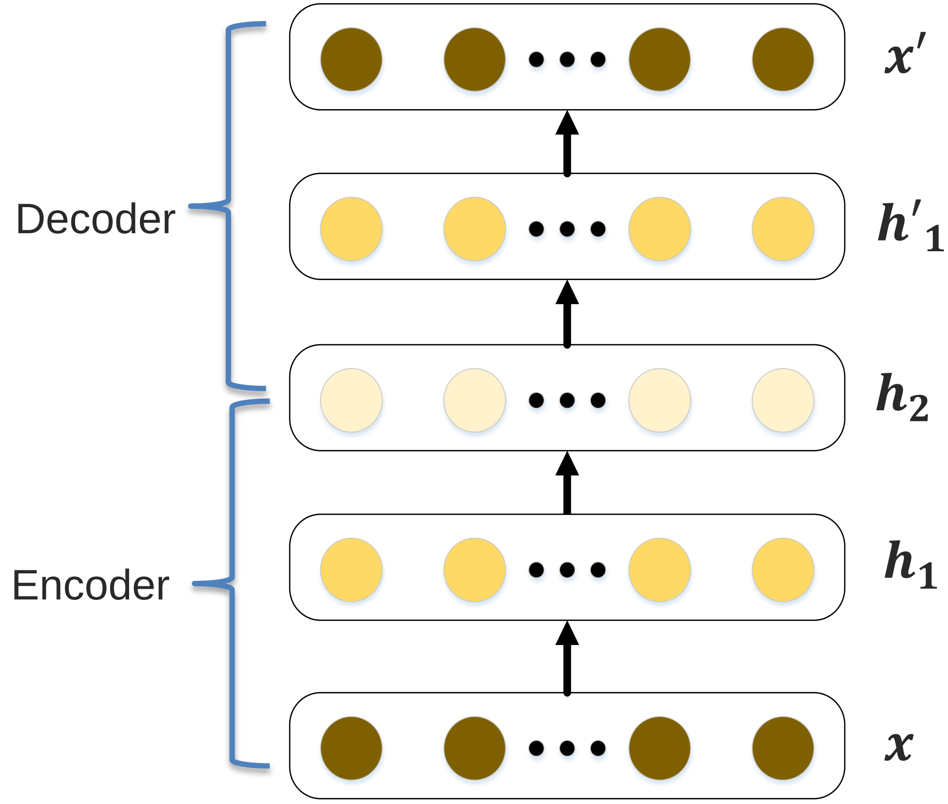
Denoising autoencoder (25% noise)

Denoising autoencoder (50% noise)

Qualitatively denoising autoencoder leads to more meaningful features

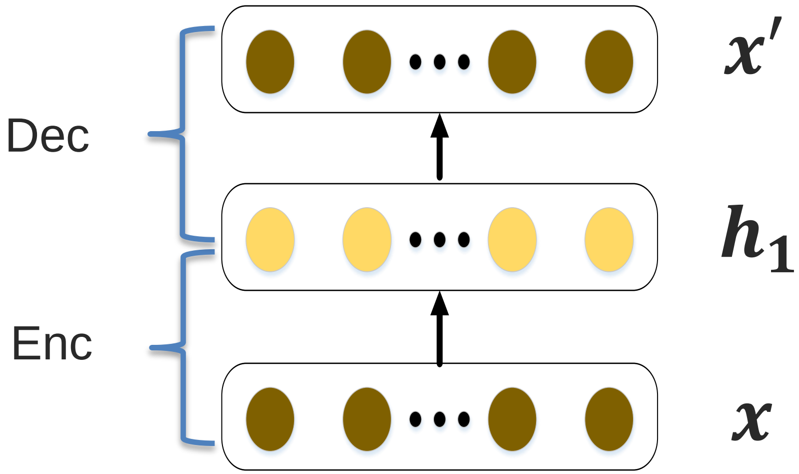
Stacked autoencoders

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



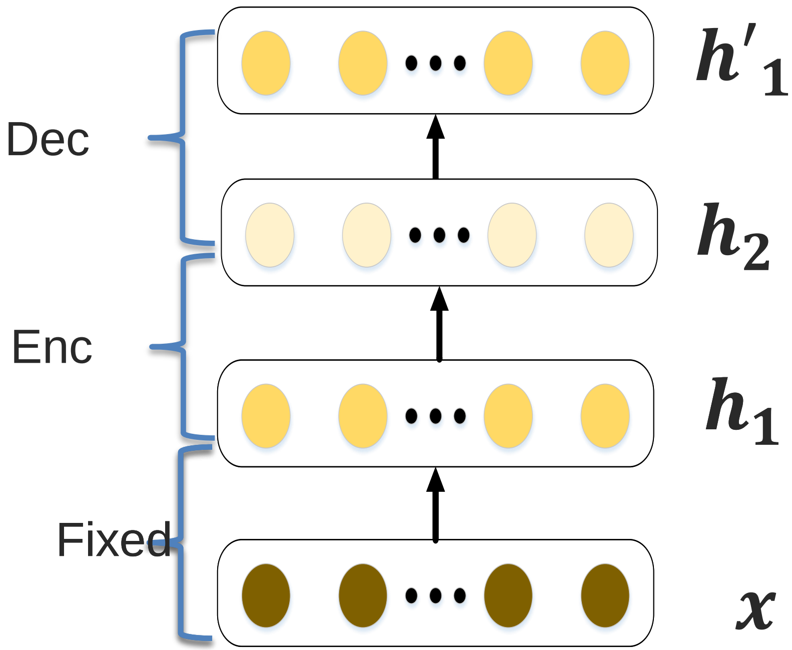
Stacked autoencoders

- 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



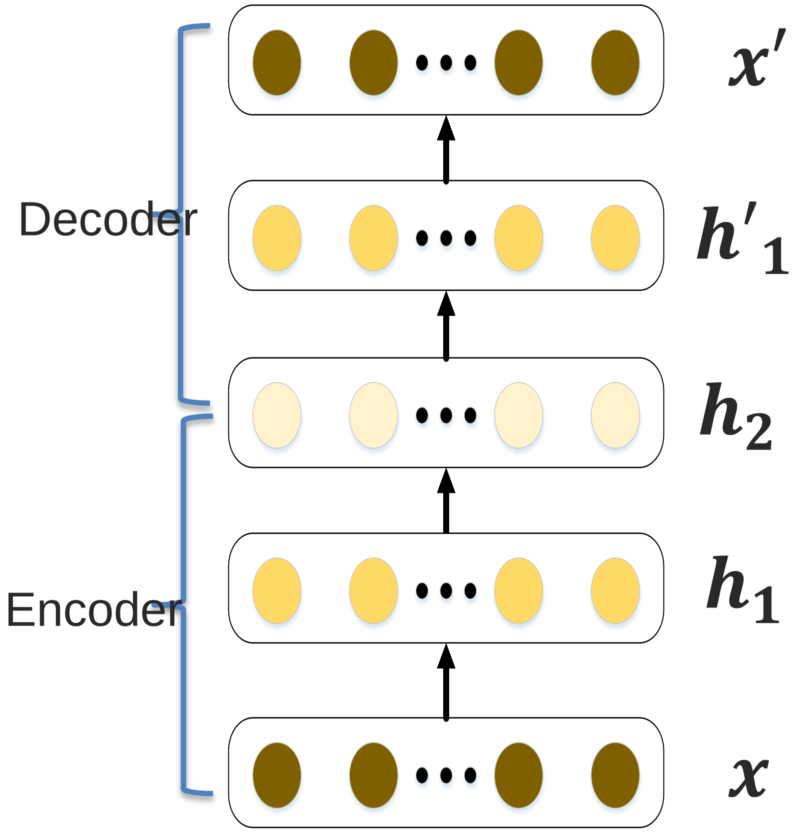
Stacked autoencoders

- 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



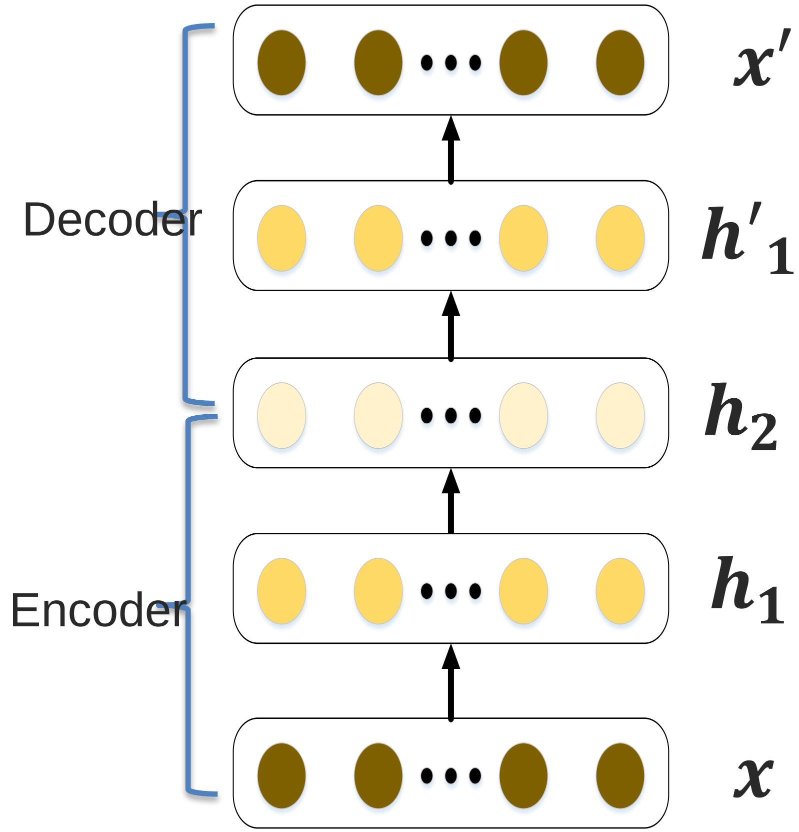
Stacked autoencoders

- 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



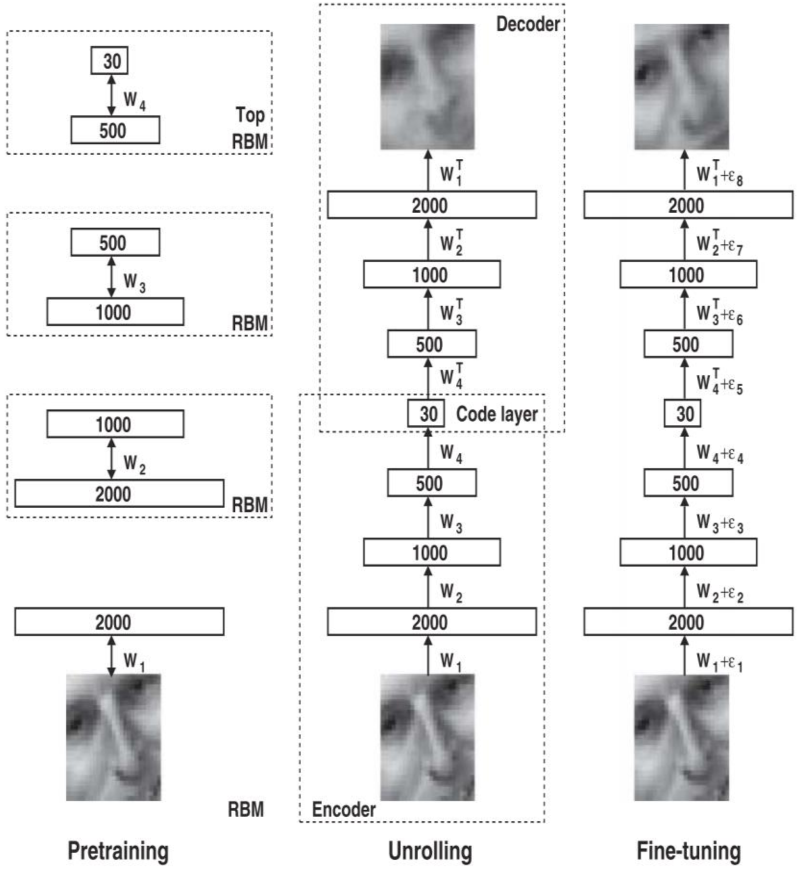
Stacked denoising autoencoders

- 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



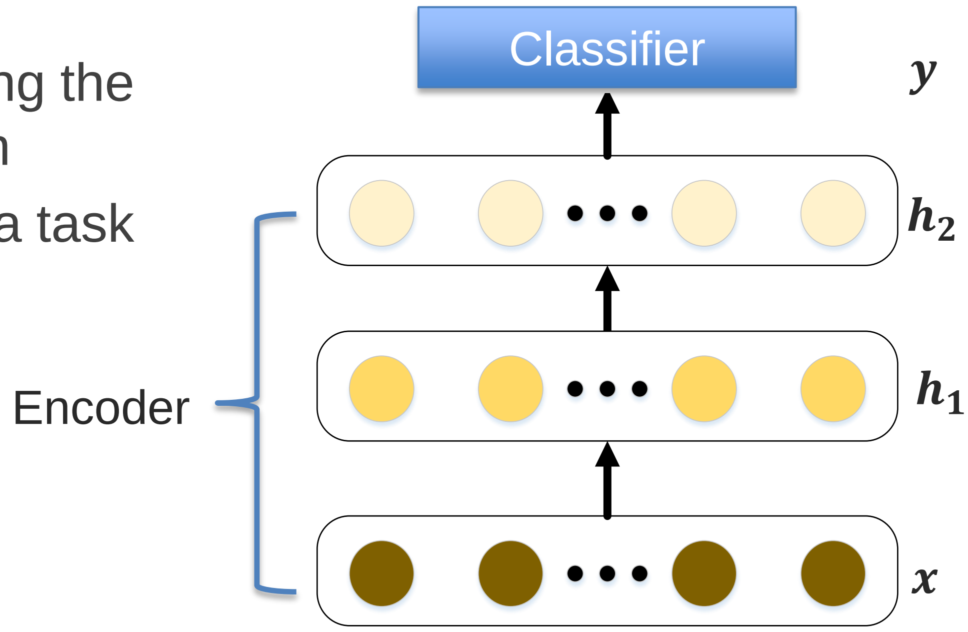
Deep representations

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



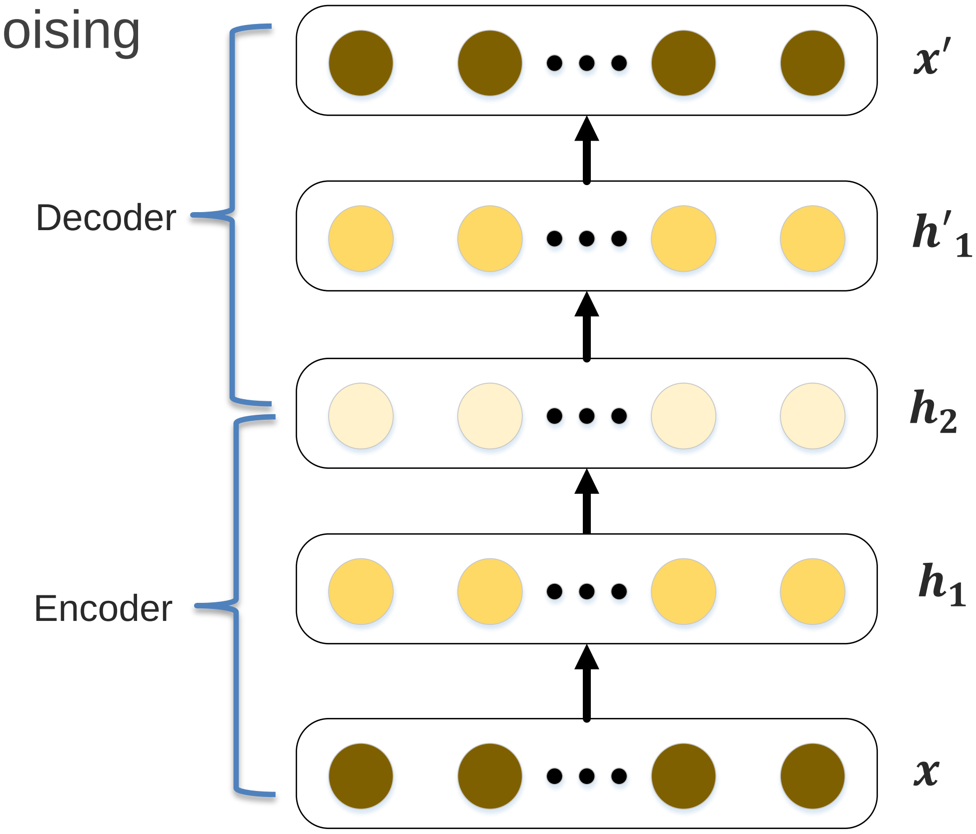
Deep representations

- 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



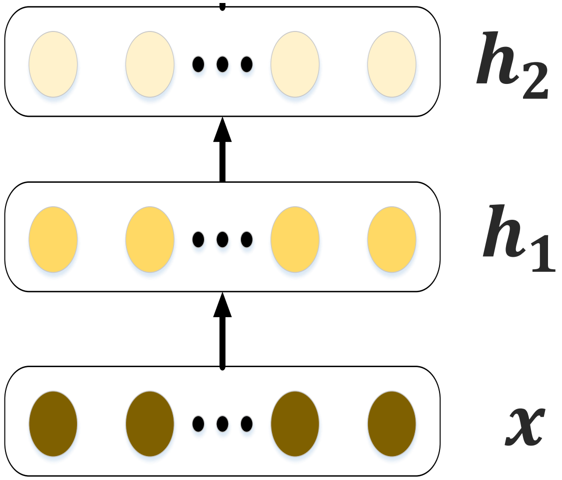
Fine-tuning an autoencoder

- Converting in to a DNN
- Start with a trained stacked denoising autoencoder



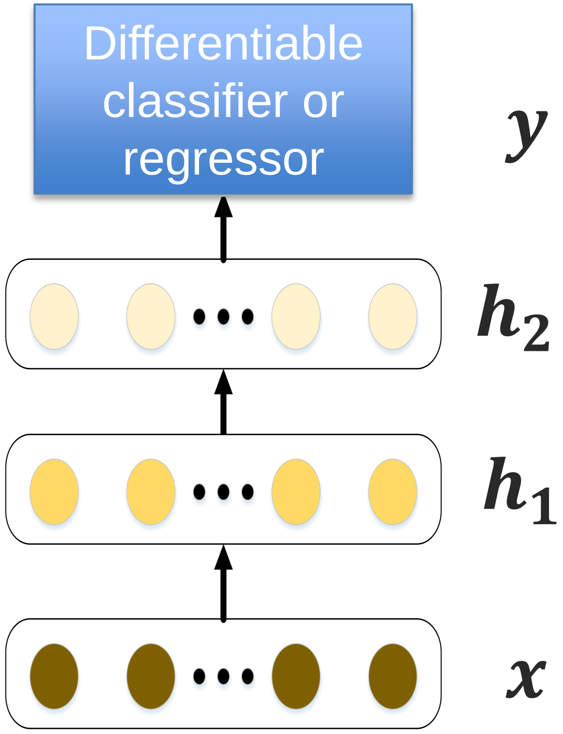
Fine-tuning an autoencoder

- Converting in to a DNN
- Start with a trained stacked denoising autoencoder
    - Discard the decoder



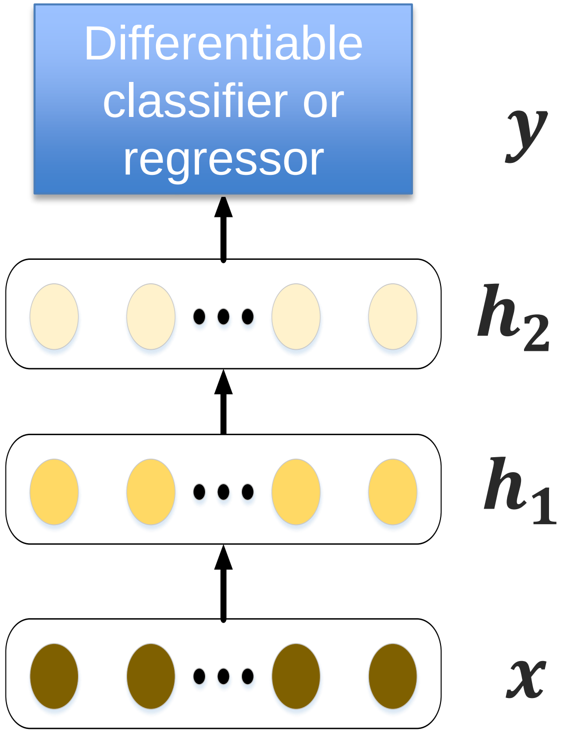
Fine-tuning an autoencoder

- Converting in to a DNN
- Start with a trained stacked denoising autoencoder
    - Discard the decoder
- Put a differentiable classifier/regressor on top



Fine-tuning an autoencoder

- Converting in to a DNN
- Start with a trained stacked denoising autoencoder
  - Discard the decoder
- Put a differentiable classifier/regressor on top
- Train the whole network end-to-end
  - Backpropagation
- [Erhan et al., Why Does Unsupervised Pre-training Help Deep Learning?, 2010]



Comparison

- Pretraining can be done on Deep Belief Networks as well
  - [Hinton et al., A fast learning algorithm for deep belief nets]

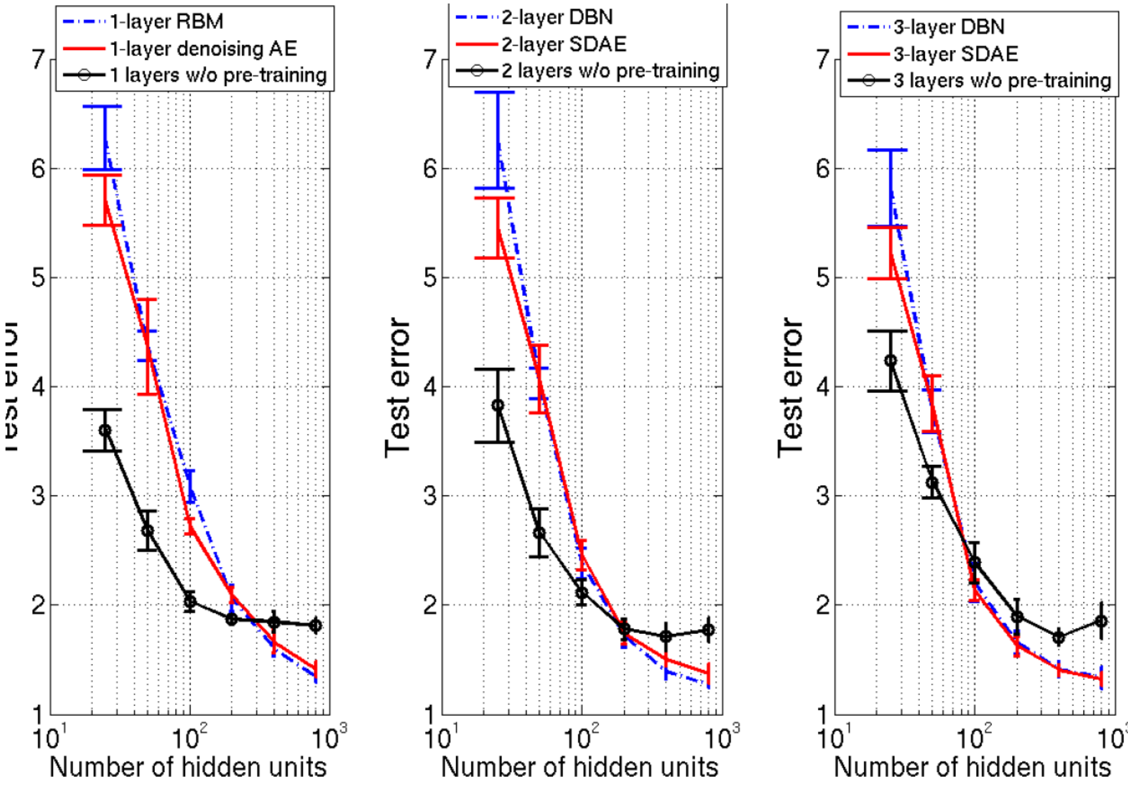
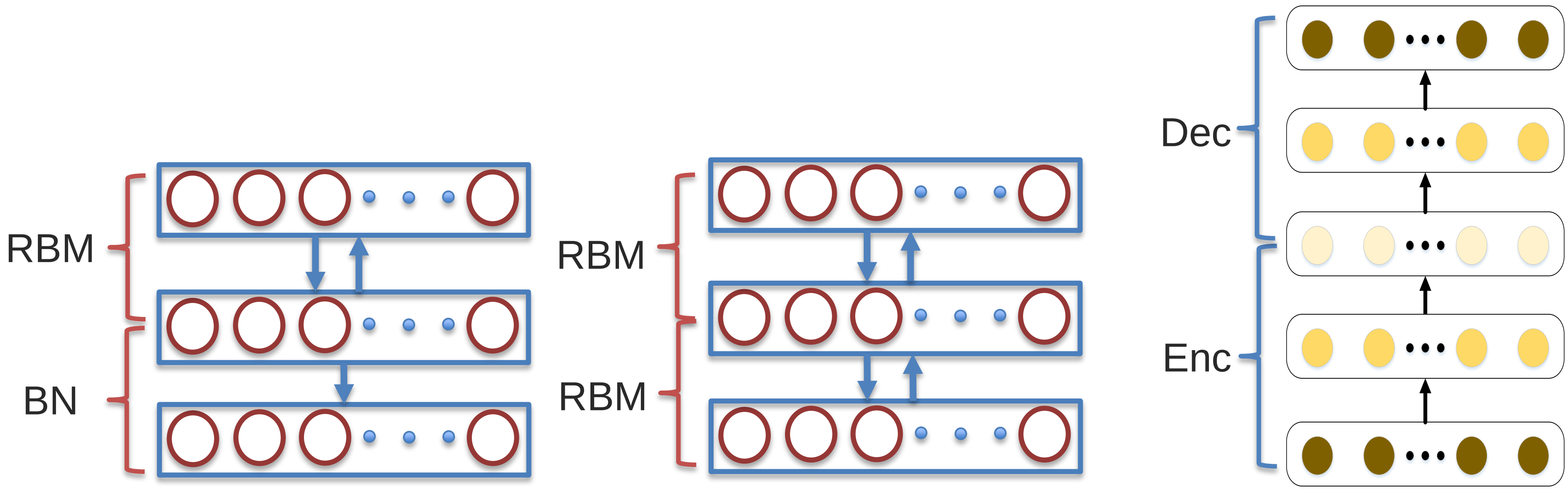


Figure from [Erhan et al., Why Does Unsupervised Pre-training Help Deep Learning?, 2010]

Deep unsupervised models

Comparison

- We have DBNs, DBMs, and stacked autoencoders
- Can actually combine them in interesting ways
  - Use RBMs or DBNs to pretrain autoencoders
- Can use either DBNs or autoencoders to initialize DNNs



Comparison

- We have DBNs, DBMs, and stacked autoencoders
  - DBNs and DBMs trickier to train but potentially more powerful

Data Set	SVM$_{rbf}$	DBN-1	SAE-3	DBN-3	SDAE-3 (ν)
MNIST	**1.40**±0.23	**1.21**±0.21	**1.40**±0.23	**1.24**±0.22	**1.28**±0.22 (25%)
basic	**3.03**±0.15	3.94±0.17	3.46±0.16	3.11±0.15	**2.84**±0.15 (10%)
rot	11.11±0.28	14.69±0.31	10.30±0.27	10.30±0.27	**9.53**±0.26 (25%)
bg-rand	14.58±0.31	9.80±0.26	11.28±0.28	**6.73**±0.22	10.30±0.27 (40%)
bg-img	22.61±0.37	**16.15**±0.32	23.00±0.37	**16.31**±0.32	**16.68**±0.33 (25%)
bg-img-rot	55.18±0.44	52.21±0.44	51.93±0.44	47.39±0.44	**43.76**±0.43 (25%)
rect	**2.15**±0.13	4.71±0.19	2.41±0.13	2.60±0.14	**1.99**±0.12 (10%)
rect-img	24.04±0.37	23.69±0.37	24.05±0.37	22.50±0.37	**21.59**±0.36 (25%)
convex	19.13±0.34	19.92±0.35	**18.41**±0.34	**18.63**±0.34	**19.06**±0.34 (10%)
tzanetakis	**14.41**±2.18	18.07±1.31	**16.15**±1.95	18.38±1.64	**16.02**±1.04(0.05)

[Vincent et al., Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion, 2010]

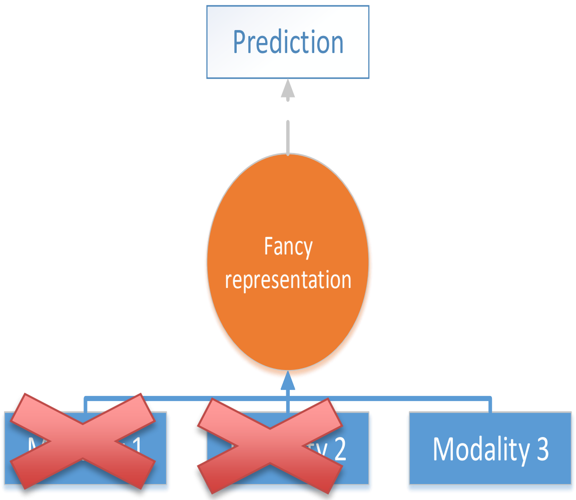
Multimodal representations

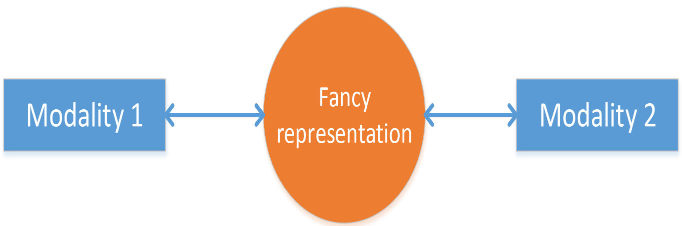
Multimodal representations

- Why do we need multimodal representations?
- Can just have unimodal ones and just fuse them
  - What if relationship is complex?
  - Doesn't exploit joint information, especially at lower/intermediate levels

Multimodal representations

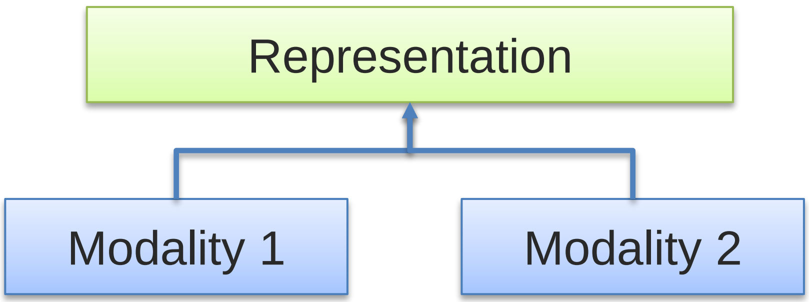
- What do we want from multi-modal representation
  - Similarity in that space implies similarity in corresponding *concepts*
  - Useful for various discriminative tasks – retrieval, mapping, fusion etc.
  - Possible to obtain in absence of one or more modalities
  - Fill in missing modalities given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





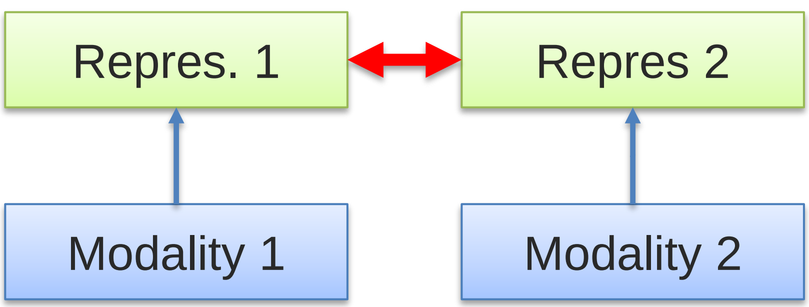
Multimodal representation types

A **Joint representations:**



- Simplest version: modality concatenation (early fusion)
- Can be learned supervised or unsupervised
- Multimodal factor analysis

B **Coordinated representations:**

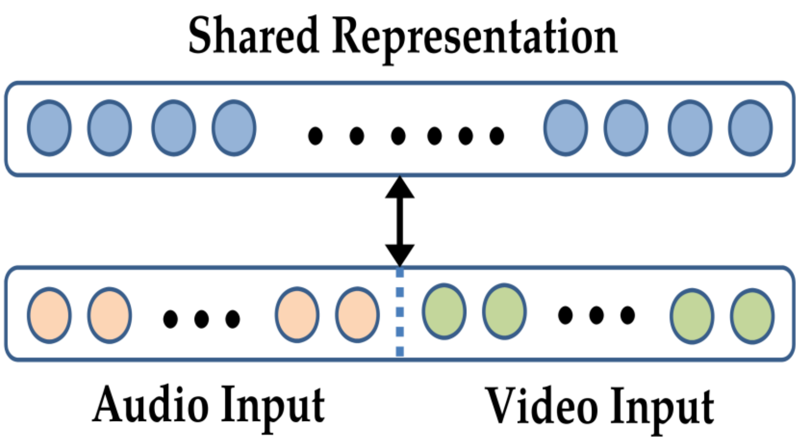


- Similarity-based methods (e.g., cosine distance)
- Structure constraints (e.g., orthogonality, sparseness)
- Talk about it today but also in two weeks (CCA)

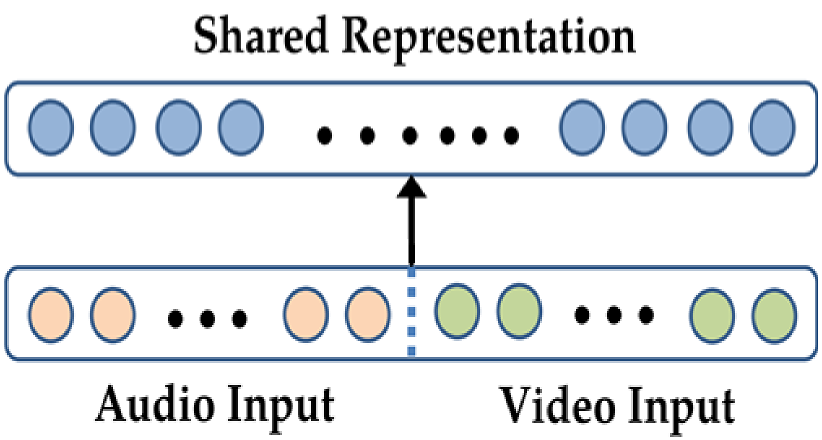
Joint representations

Shallow multimodal representations

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



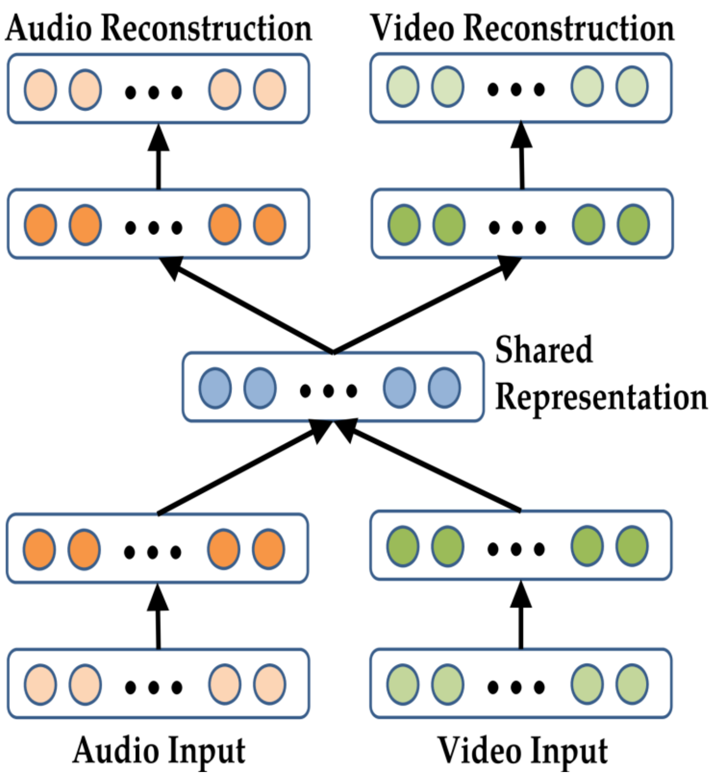
Shallow RBM



Shallow Autoencoder

Deep Multimodal autoencoders

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

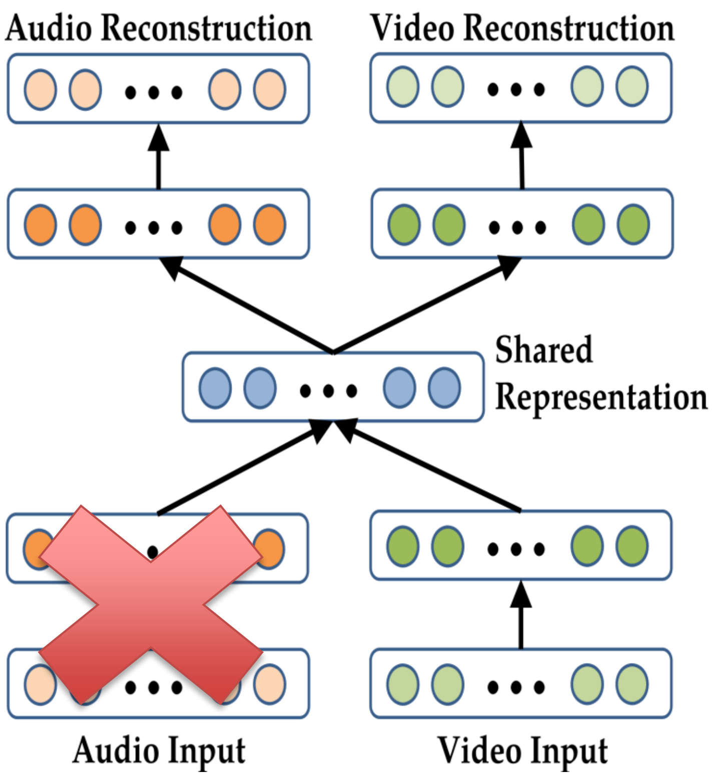


- [Ngiam et al., Multimodal Deep Learning, 2011]

Deep Multimodal autoencoders - training

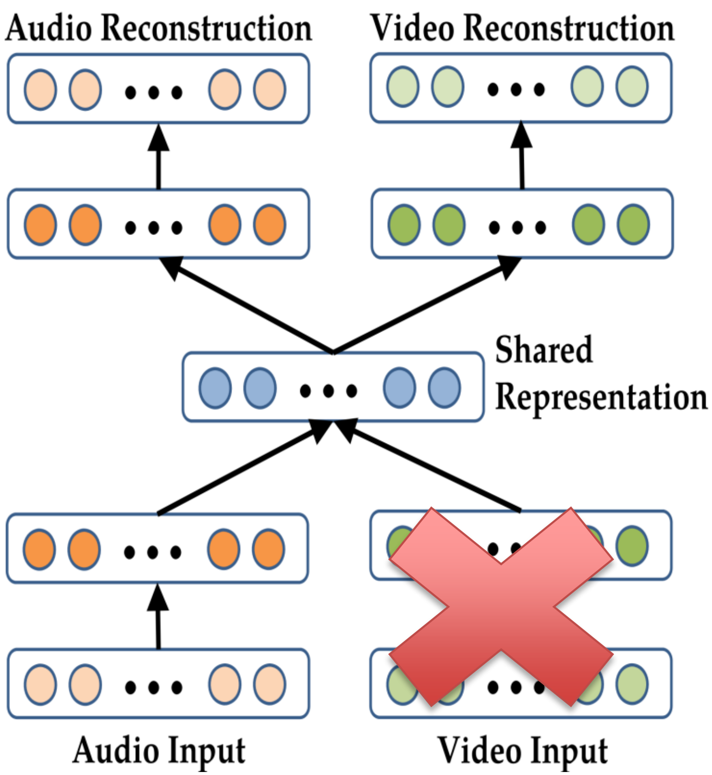
- Individual modalities can be pre-trained
  - Denoising Autoencoders

- To train the model to reconstruct the other modality
  - Use both
  - Remove audio



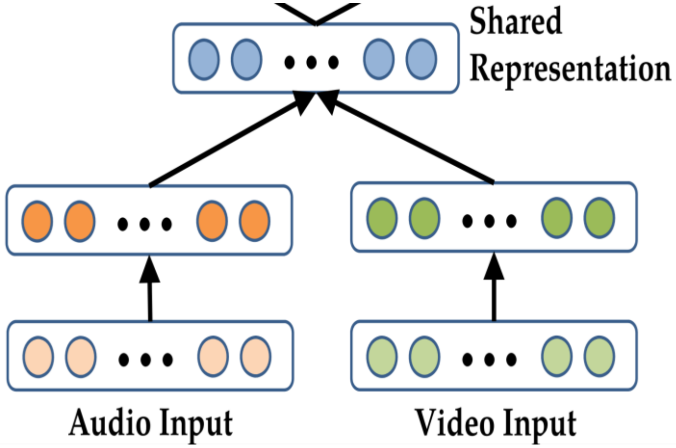
Deep Multimodal autoencoders - training

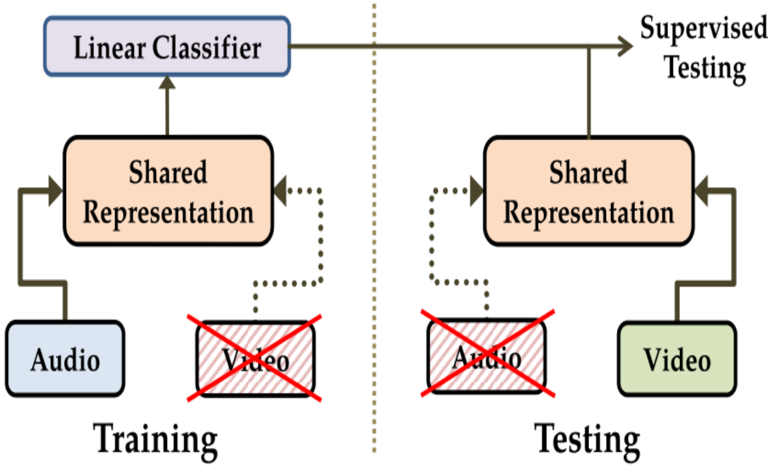
- Individual modalities can be pretrained
  - RBMs
  - Denoising Autoencoders
- To train the model to reconstruct the other modality
  - Use both
  - Remove audio
  - Remove video



Deep Multimodal autoencoders

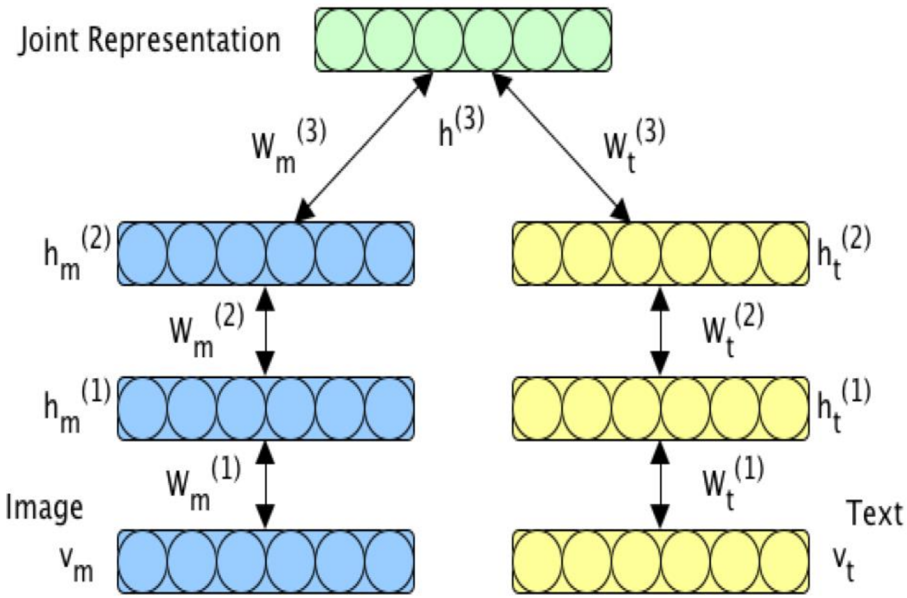
- Can now discard the decoder and use it for the AVSR task
- Interesting experiment
  - “Hearing to see”





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

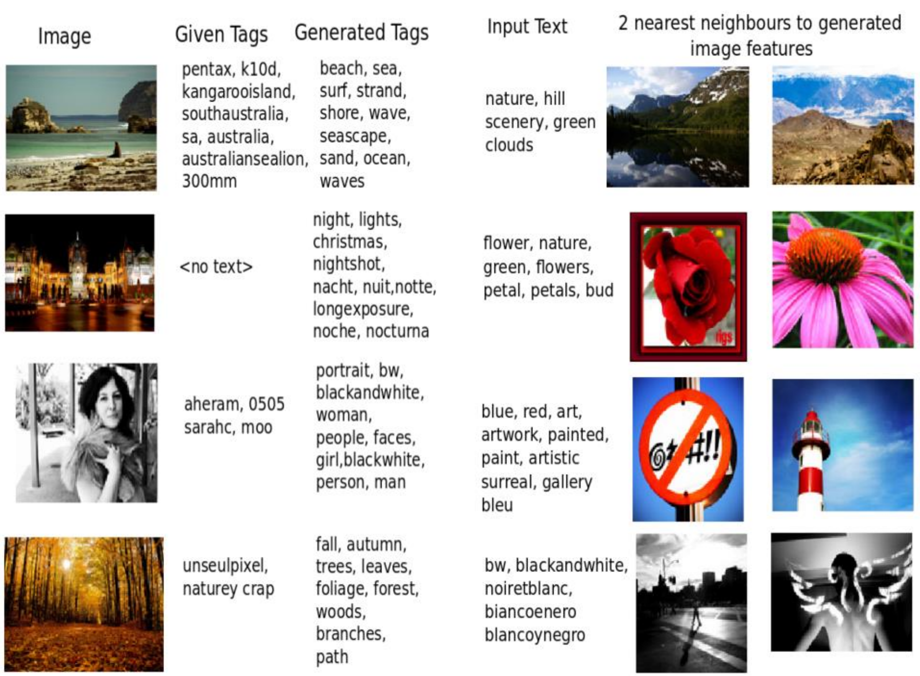


- [Srivastava and Salakhutdinov, Multimodal Learning with Deep Boltzmann Machines, 2012, 2014]

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/

Comparing deep multimodal representations

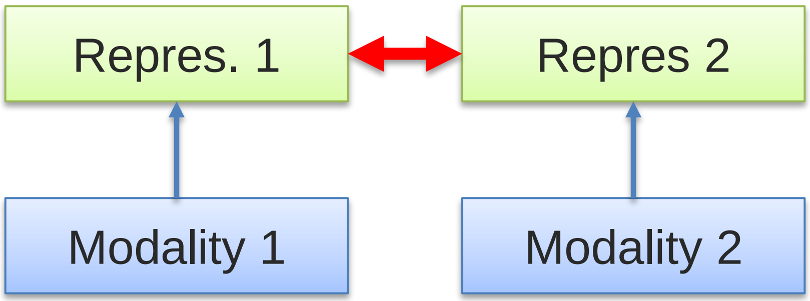
- Difference between them and the RBMs and the autoencoders
- Overall very similar behavior

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	$\mathbf{0.641} \pm \mathbf{0.004}$

Coordinated representations

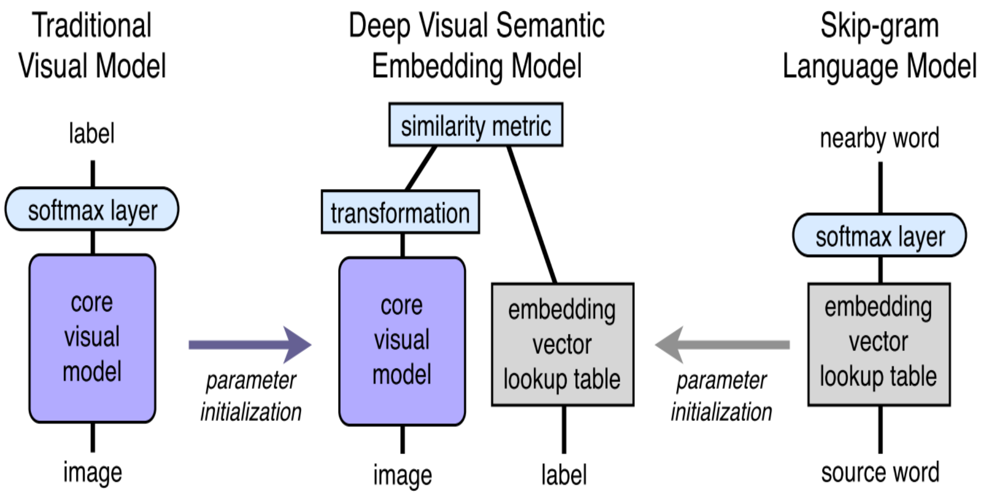
Coordinated multimodal embeddings

- Instead of projecting to a joint space enforce the similarity between unimodal embeddings



Coordinated multimodal embeddings

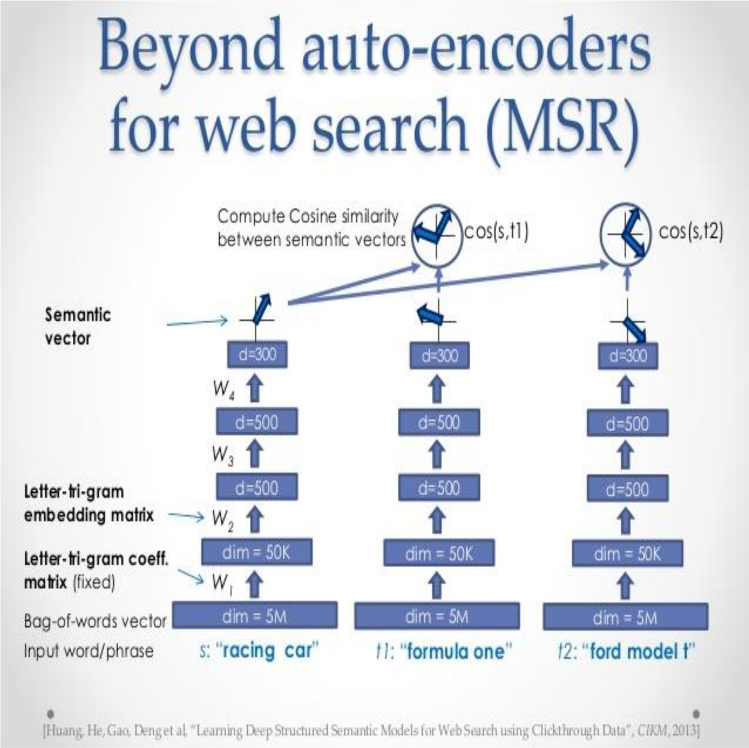
- Instead of projecting to a joint space enforce the similarity between unimodal embeddings (topic of our reading group on Thursday)
- Often referred to as semantic space embeddings

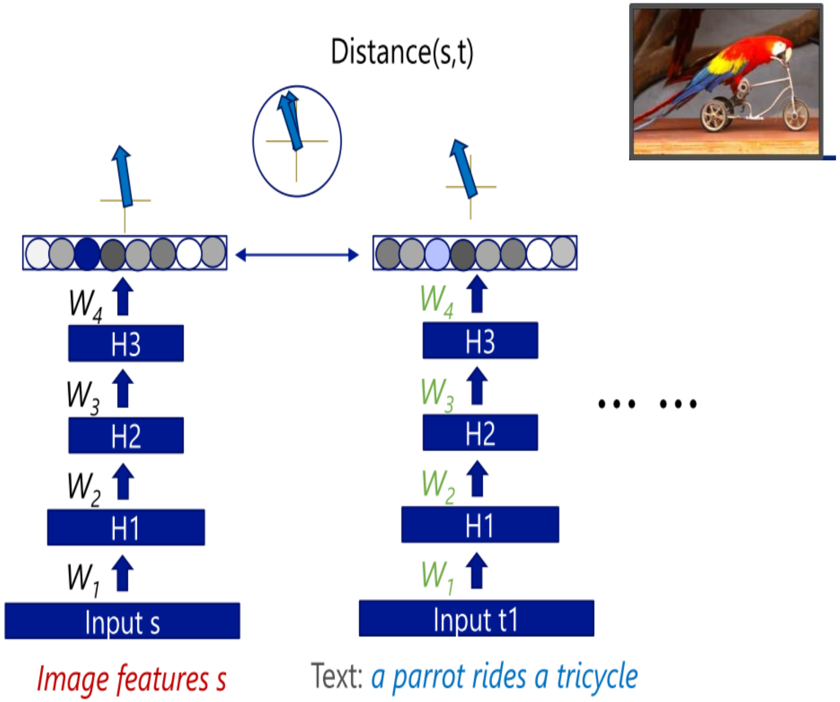


[Kiros et al., Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models, 2014]
[Frome et al., DeViSE: A Deep Visual-Semantic Embedding Model, 2013]

Coordinated multimodal embeddings

- Deep structured semantic models

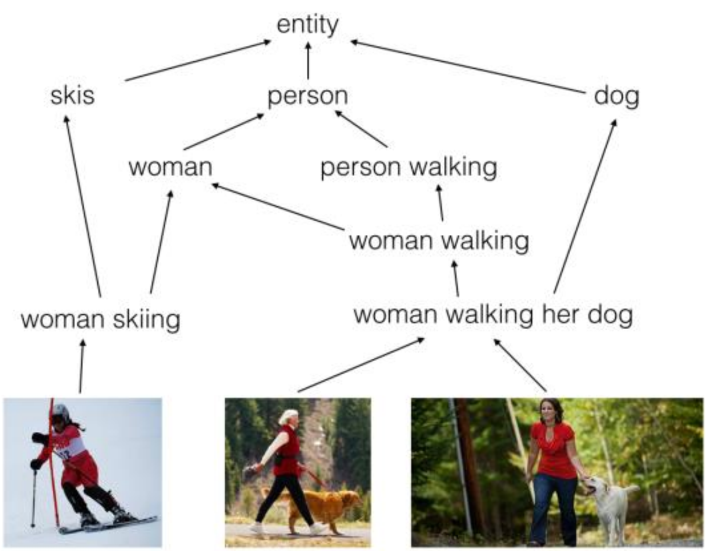


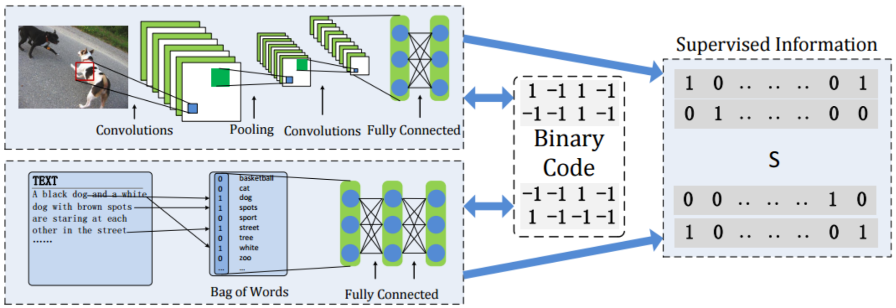


- [Huang et al., Learning Deep Structured Semantic Models for Web Search using Clickthrough Data, 2013]

Structured coordinated embeddings

- Instead of or in addition to similarity add alternative structure





[Vendrov et al., Order-Embeddings of Images and Language, 2016]

[Jiang and Li, Deep Cross-Modal Hashing]

Recap unsupervised representation

- RBM
  - Project modalities to the same space
  - Use when all the modalities are present during test time
  - Suitable for multimodal fusion
- Autoencoder
  - Project modalities to their own coordinated space
  - Use when only one of the modalities is present during test-time
  - Suitable for multimodal translation
  - Good for retrieval

Recap multimodal representations

- Joint representations
  - Project modalities to the same space
  - Use when all the modalities are present during test time
  - Suitable for multimodal fusion
- Coordinated representations
  - Project modalities to their own coordinated space
  - Use when only one of the modalities is present during test-time
  - Suitable for multimodal translation
  - Good for retrieval