Advanced Machine Learning
Lecture 17

Word Embeddings

18.01.2016

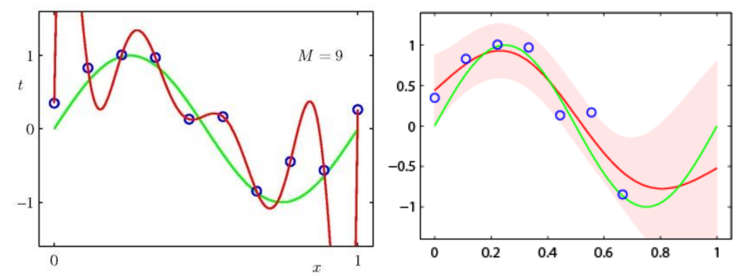
Bastian Leibe
RWTH Aachen
http://www.vision.rwth-aachen.de/

leibe@vision.rwth-aachen.de

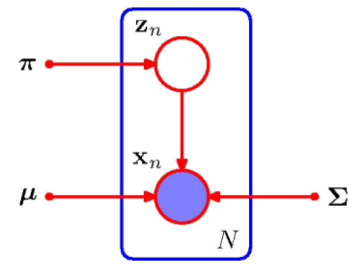
This Lecture: *Advanced Machine Learning*

- Regression Approaches
  - Linear Regression
  - Regularization (Ridge, Lasso)
  - Gaussian Processes

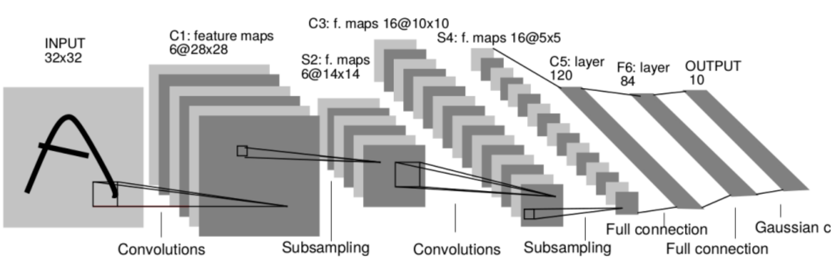
$$f : \mathcal{X} \to \mathbb{R}$$



- Learning with Latent Variables
  - Prob. Distributions & Approx. Inference
  - Mixture Models
  - EM and Generalizations



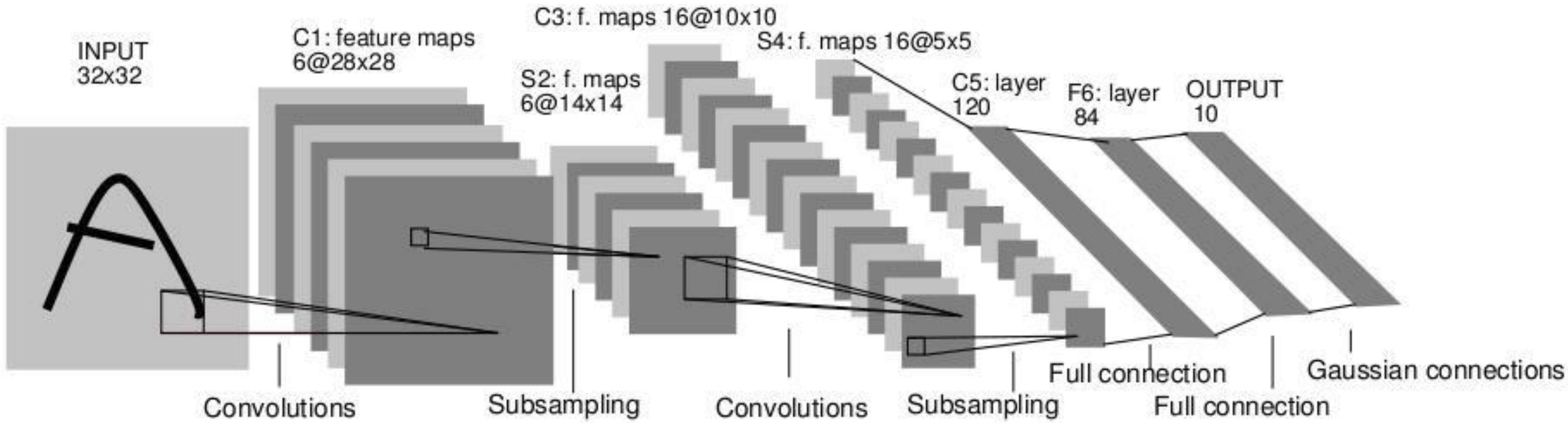
- Deep Learning
  - Linear Discriminants
  - Neural Networks
  - Backpropagation & Optimization
  - CNNs, RNNs, RBMs, etc.



Topics of This Lecture

- Recap: CNN Architectures
- Applications of CNNs
- Word Embeddings
  - Neuroprobabilistic Language Models
  - word2vec
  - GloVe
  - Hierarchical Softmax
- Outlook: Recurrent Neural Networks

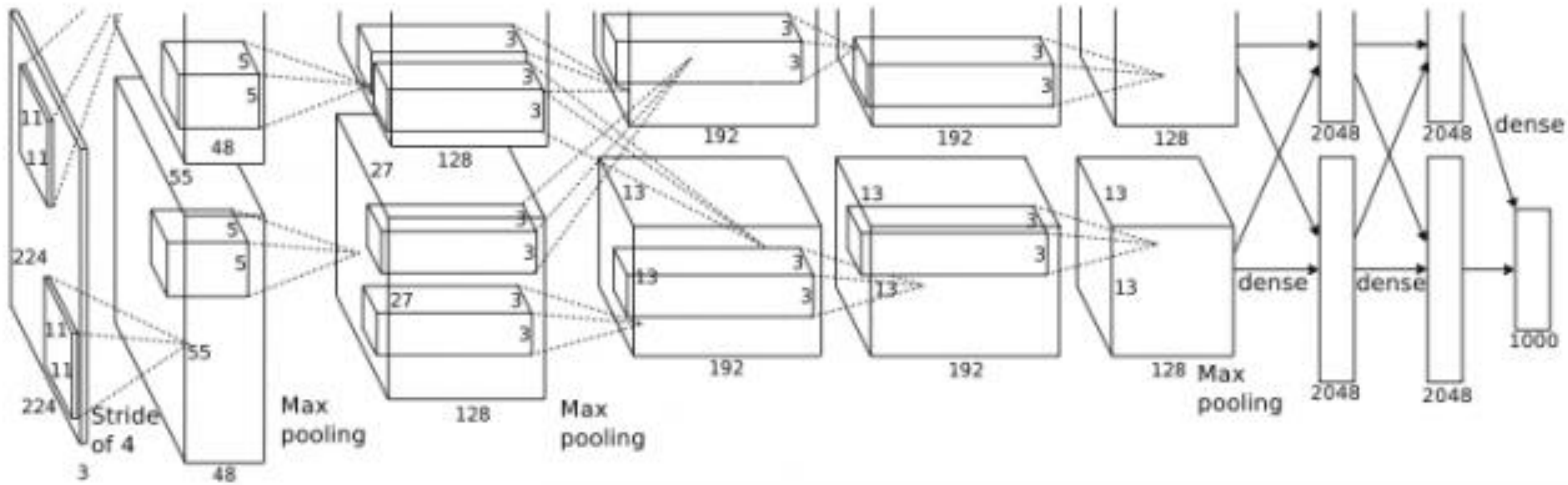
Recap: Convolutional Neural Networks



- Neural network with specialized connectivity structure
  - Stack multiple stages of feature extractors
  - Higher stages compute more global, more invariant features
  - Classification layer at the end

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86(11): 2278–2324, 1998.

Recap: AlexNet (2012)



- **Similar framework as LeNet, but**
  - **Bigger model (7 hidden layers, 650k units, 60M parameters)**
  - **More data (10^6 images instead of 10^3)**
  - **GPU implementation**
  - **Better regularization and up-to-date tricks for training (Dropout)**

A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012.

Image source: A. Krizhevsky, I. Sutskever and G.E. Hinton, NIPS 2012

Recap: VGGNet (2014/15)

- **Main ideas**
  - Deeper network
  - Stacked convolutional layers with smaller filters (+ nonlinearity)
  - Detailed evaluation of all components
- **Results**
  - Improved ILSVRC top-5 error rate to 6.7%.

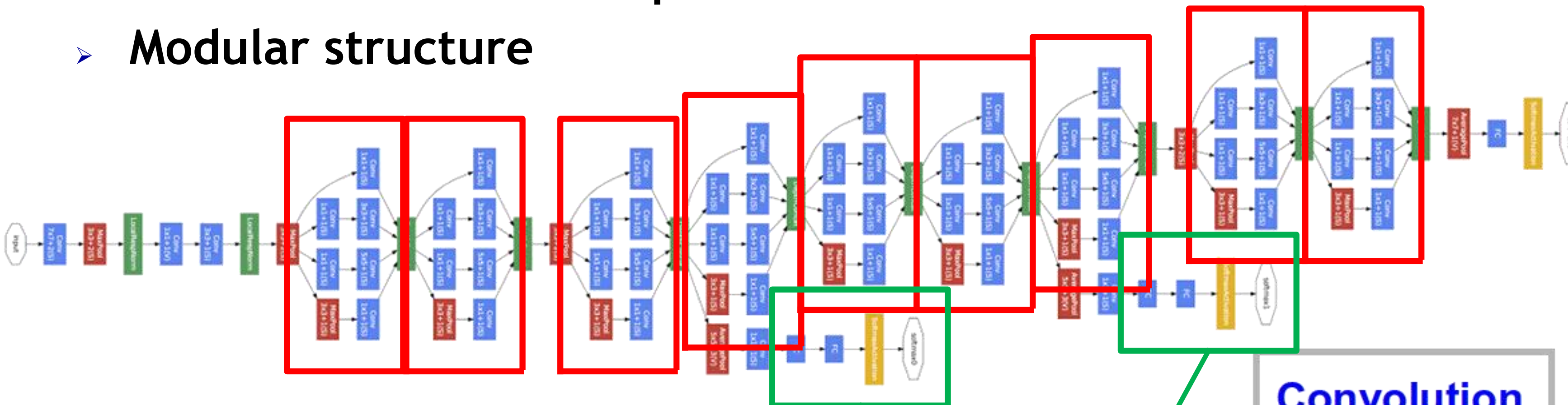
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Mainly used

Image source: Simonyan & Zisserman

Recap: GoogLeNet (2014)

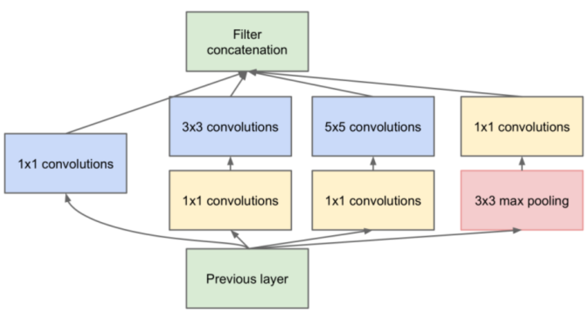
- Ideas:
  - Learn features at multiple scales
  - Modular structure



Inception module + copies

Convolution
Pooling
Softmax
Other

Auxiliary classification outputs for training the lower layers (deprecated)



(b) Inception module with dimension reductions

Image source: Szegedy et al.

Discussion

- GoogLeNet
  - 12× fewer parameters than AlexNet
  - ⇒ ~5M parameters
  - *Where does the main reduction come from?*
  - ⇒ From throwing away the fully connected (FC) layers.
- Effect
  - After last pooling layer, volume is of size [7×7×1024]
  - Normally you would place the first 4096-D FC layer here (Many million params).
  - Instead: use Average pooling in each depth slice:
  - ⇒ Reduces the output to [1×1×1024].
  - ⇒ Performance actually improves by 0.6% compared to when using FC layers (less overfitting?)

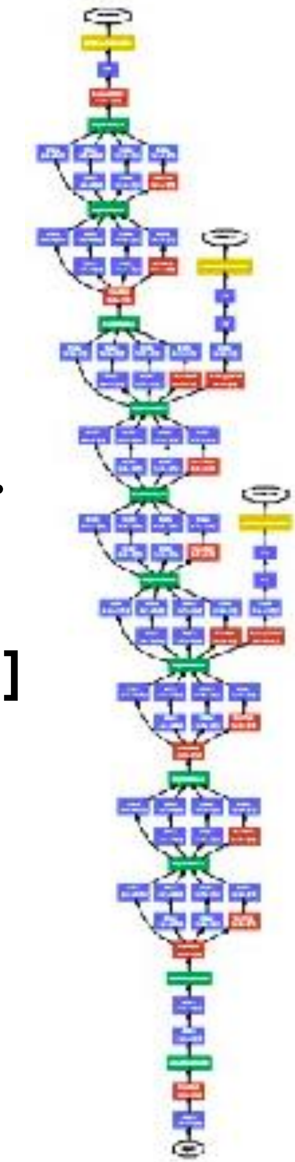
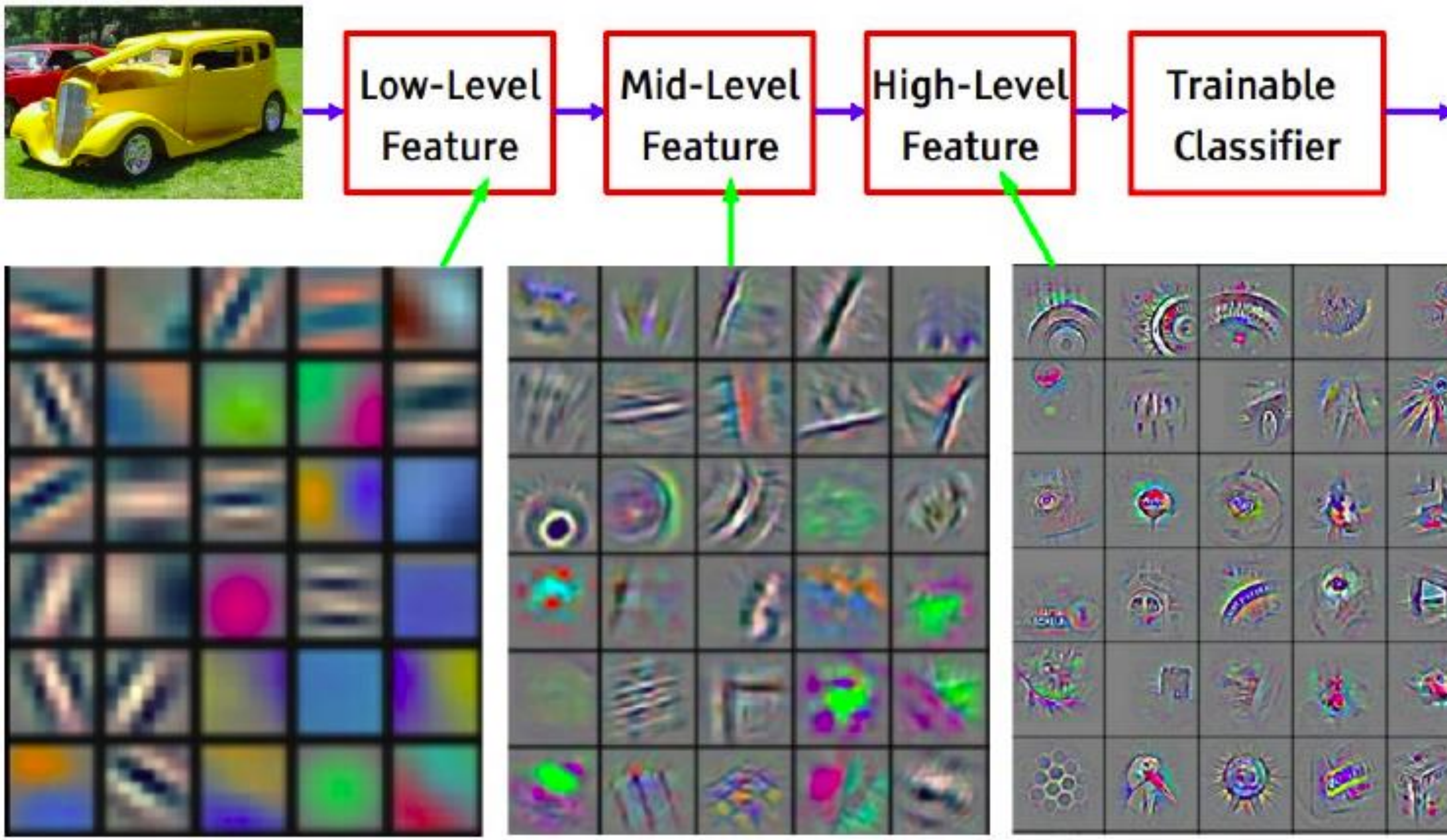


Image source: Szegedy et al.

Recap: Visualizing CNNs



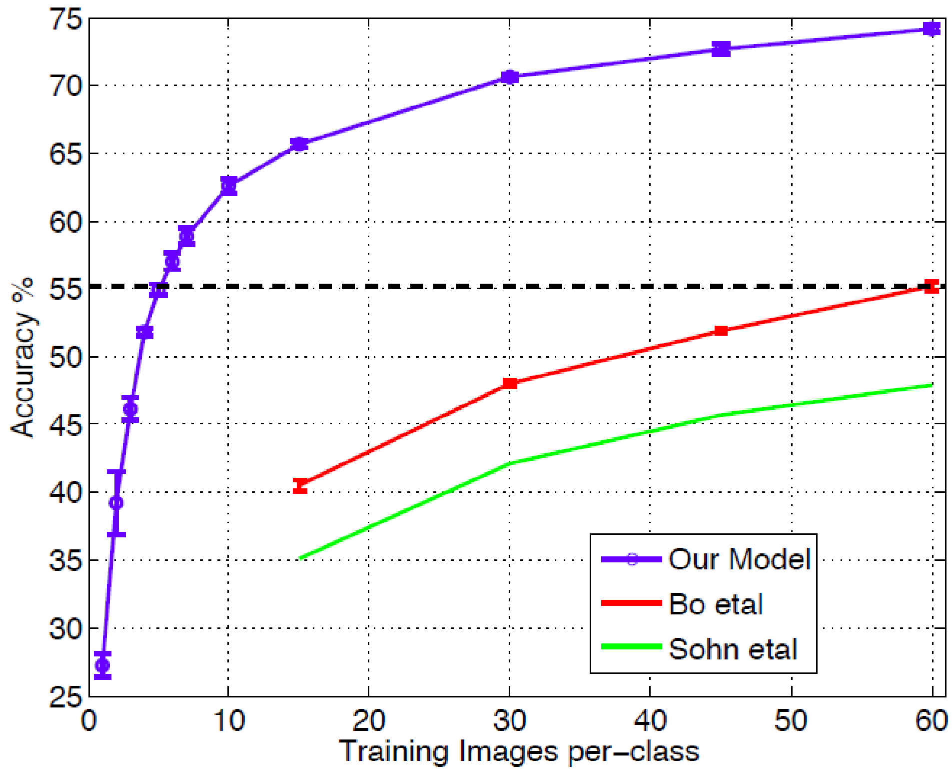
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Slide credit: Yann LeCun

Topics of This Lecture

- Recap: CNN Architectures
- Applications of CNNs
- Word Embeddings
  - Neuroprobabilistic Language Models
  - word2vec
  - GloVe
  - Hierarchical Softmax
- Outlook: Recurrent Neural Networks

The Learned Features are Generic



state of the art level (pre-CNN)

- **Experiment: feature transfer**
  - Train AlexNet-like network on ImageNet
  - Chop off last layer and train classification layer on CalTech256

$\Rightarrow$ State of the art accuracy already with only 6 training images!

Image source: M. Zeiler, R. Fergus

Transfer Learning with CNNs

1. Train on ImageNet

2. If small dataset: fix all weights (treat CNN as fixed feature extractor), retrain only the classifier

I.e., swap the Softmax layer at the end

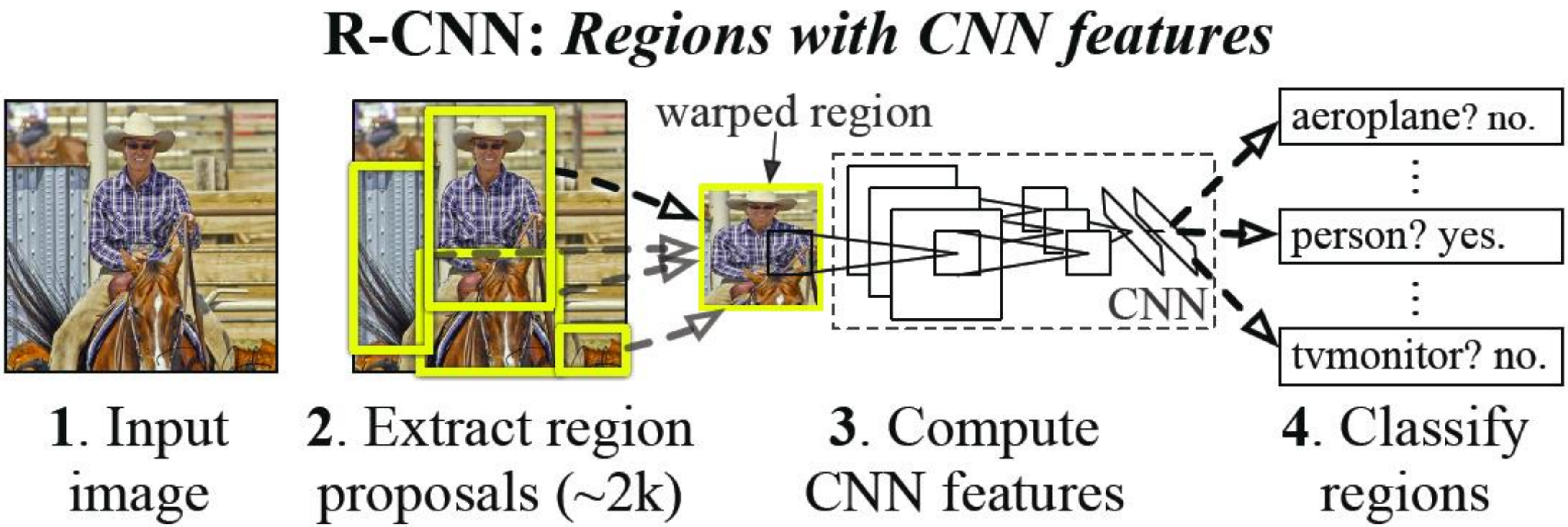
Transfer Learning with CNNs

1. Train on ImageNet

3. If you have medium sized dataset, "finetune" instead: use the old weights as initialization, train the full network or only some of the higher layers.

Retrain bigger portion of the network

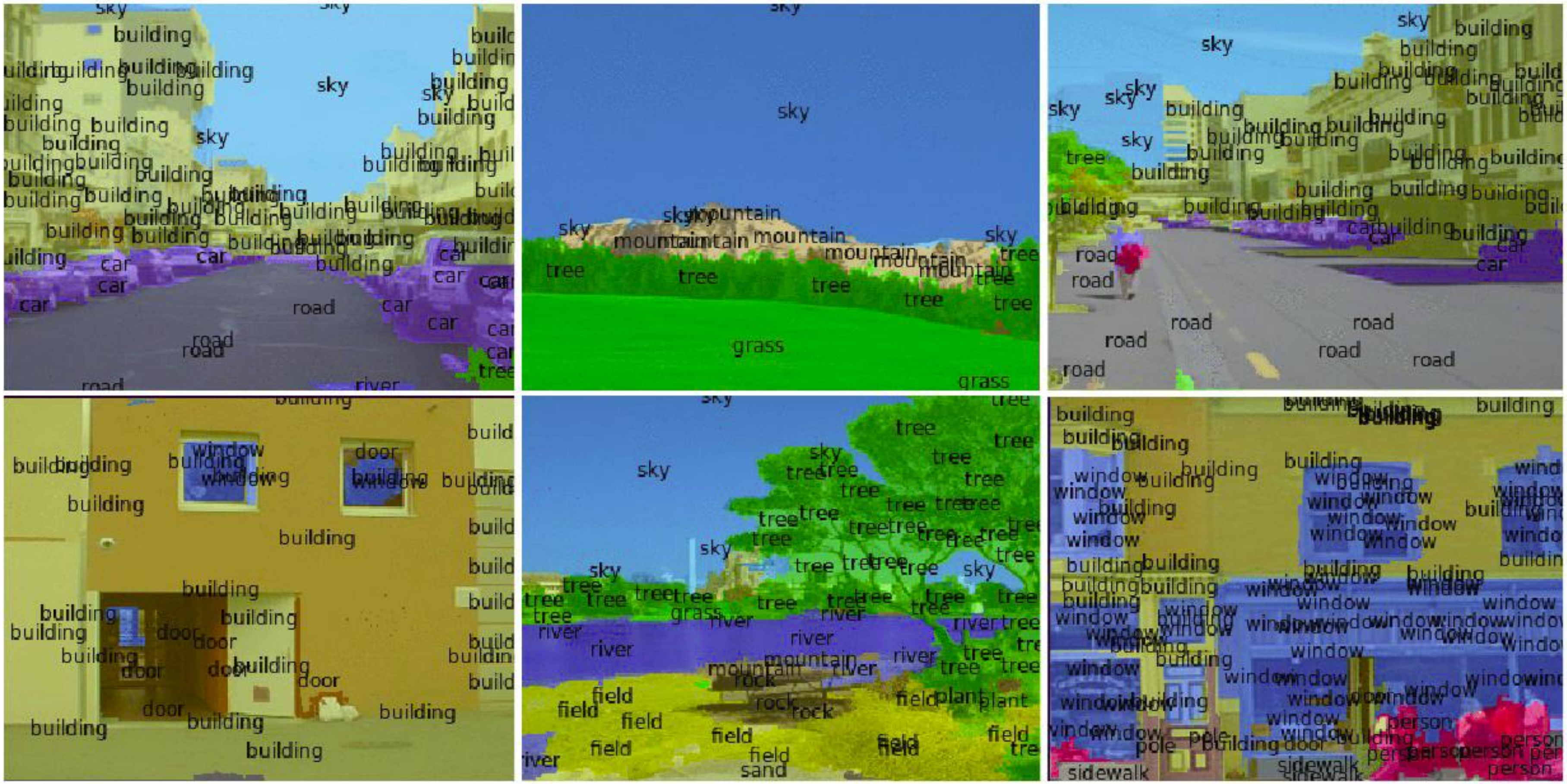
Other Tasks: Detection



- **Results on PASCAL VOC Detection benchmark**
  - **Pre-CNN state of the art: 35.1% mAP [Uijlings et al., 2013]**
    33.4% mAP DPM
  - **R-CNN: 53.7% mAP**

R. Girshick, J. Donahue, T. Darrell, and J. Malik, Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, CVPR 2014

Other Tasks: Semantic Segmentation

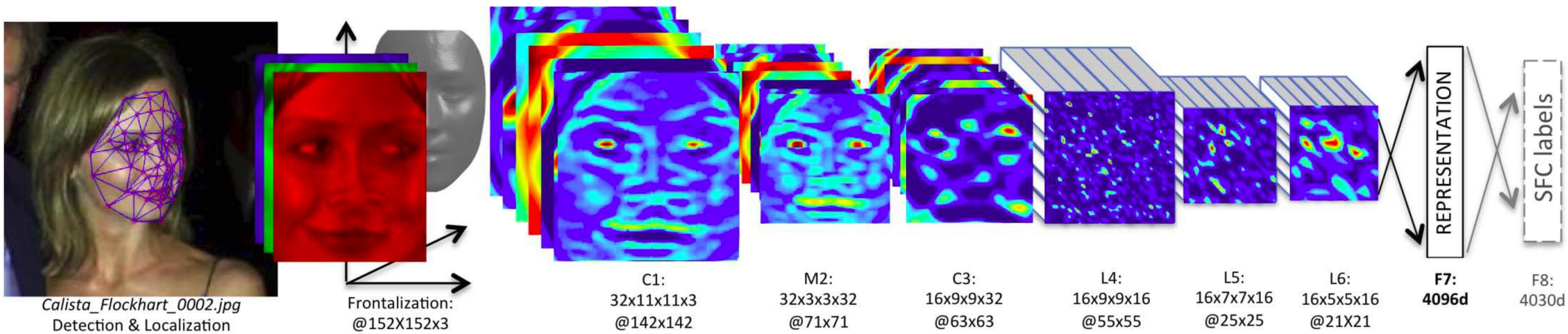


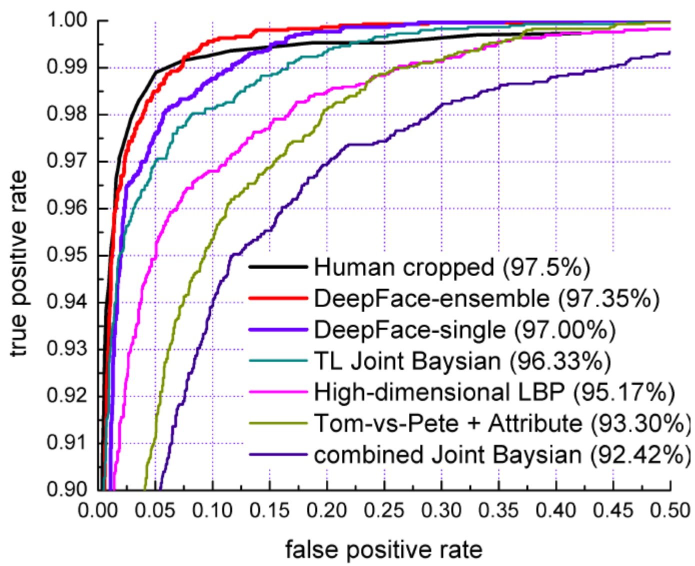
[Farabet et al. ICML 2012, PAMI 2013]

Other Tasks: Semantic Segmentation

[Farabet et al. ICML 2012, PAMI 2013]

Other Tasks: Face Verification





Y. Taigman, M. Yang, M. Ranzato, L. Wolf, DeepFace: Closing the Gap to Human-Level Performance in Face Verification, CVPR 2014

Slide credit: Svetlana Lazebnik

Commercial Recognition Services

- E.g., clarifai

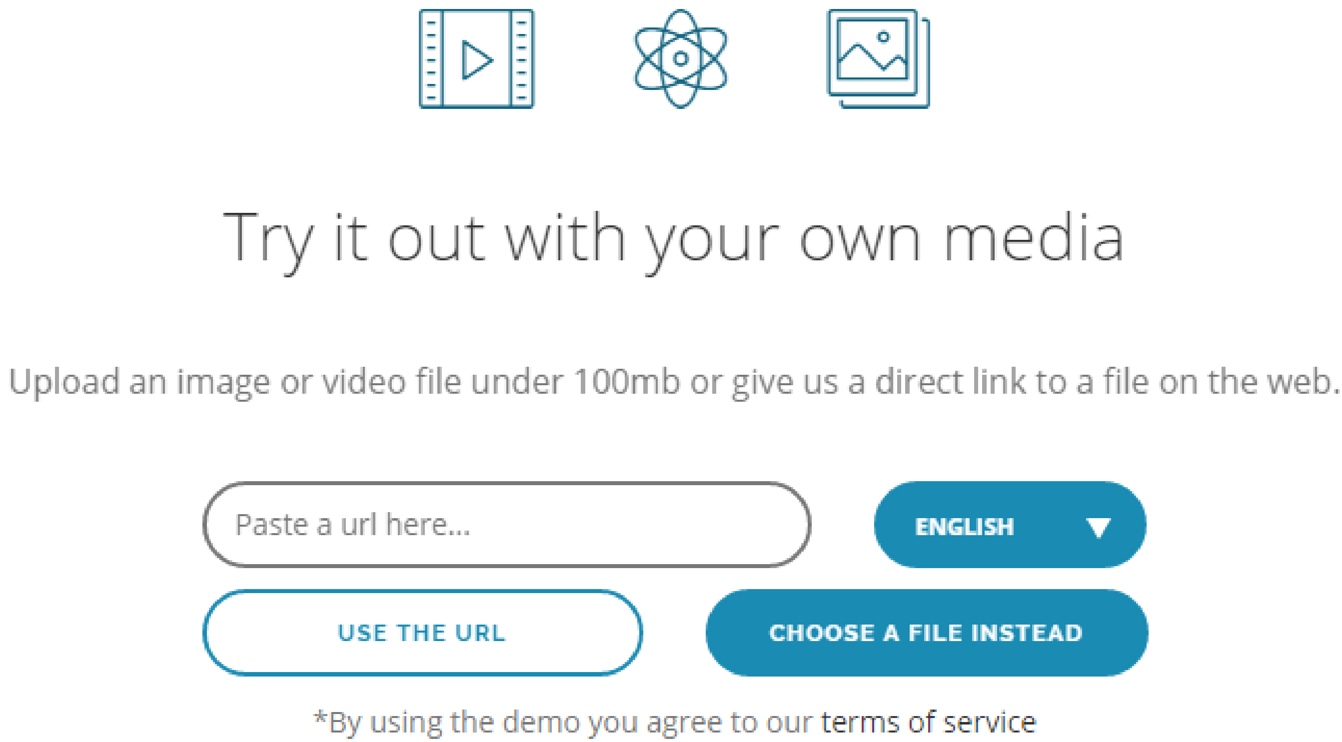
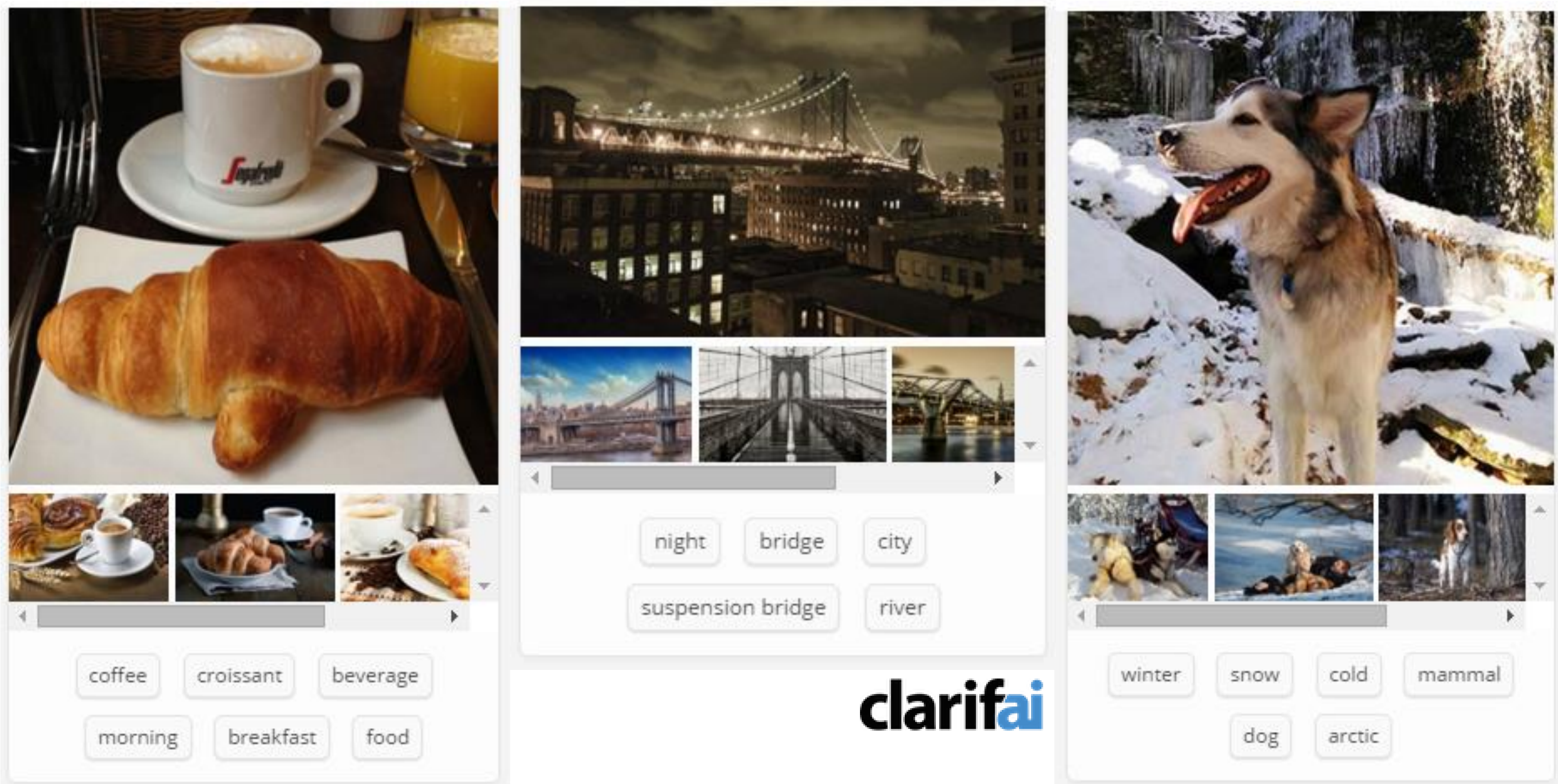


Image source: clarifai.com

Commercial Recognition Services



- **Be careful when taking test images from Google Search**
  - **Chances are they may have been seen in the training set...**

Image source: clarifai.com

Topics of This Lecture

- Recap: CNN Architectures
- Applications of CNNs
- Word Embeddings
  - Neuroprobabilistic Language Models
  - word2vec
  - GloVe
  - Hierarchical Softmax
- Outlook: Recurrent Neural Networks

Neural Networks for Sequence Data

- Up to now
  - Simple structure: Input vector → Processing → Output
- In the following, we will look at sequence data
  - Interesting new challenges
  - Varying input/output length, need to memorize state, long-term dependencies, ...
- Currently a hot topic
  - Early successes of NNs for text / language processing.
  - Very good results for part-of-speech tagging, automatic translation, sentiment analysis, etc.
  - Recently very interesting developments for video understanding, image+text modeling (e.g., creating image descriptions), and even single-image understanding (attention processes).

Motivating Example

- Predicting the next word in a sequence
  - Important problem for speech recognition, text autocorrection, etc.
- Possible solution: The trigram (n-gram) method
  - Take huge amount of text and count the frequencies of all triplets (n-tuples) of words.
  - Use those frequencies to predict the relative probabilities of words given the two previous words

$$\frac{p(w_3 = c | w_2 = b, w_1 = a)}{p(w_3 = d | w_2 = b, w_1 = a)} = \frac{\text{count}(abc)}{\text{count}(abd)}$$

  - State-of-the-art until not long ago...

Problems with N-grams

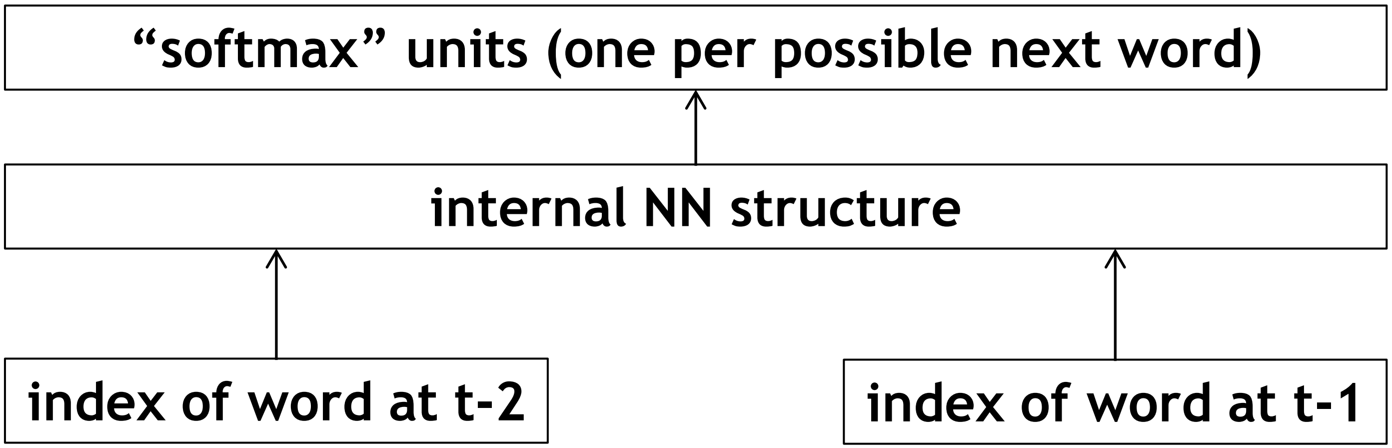
- Problem: Scalability
  - We cannot easily scale this to large N.
  - The number of possible combinations increases exponentially
  - So does the required amount of data

- Problem: Partial Observability
  - With larger N, many counts would be zero.
  - The probability is not zero, just because the count is zero!

  $\Rightarrow$ Need to back off to (N-1)-grams when the count for N-grams is too small.

  $\Rightarrow$ Necessary to use elaborate techniques, such as Kneser-Ney smoothing, to compensate for uneven sampling frequencies.

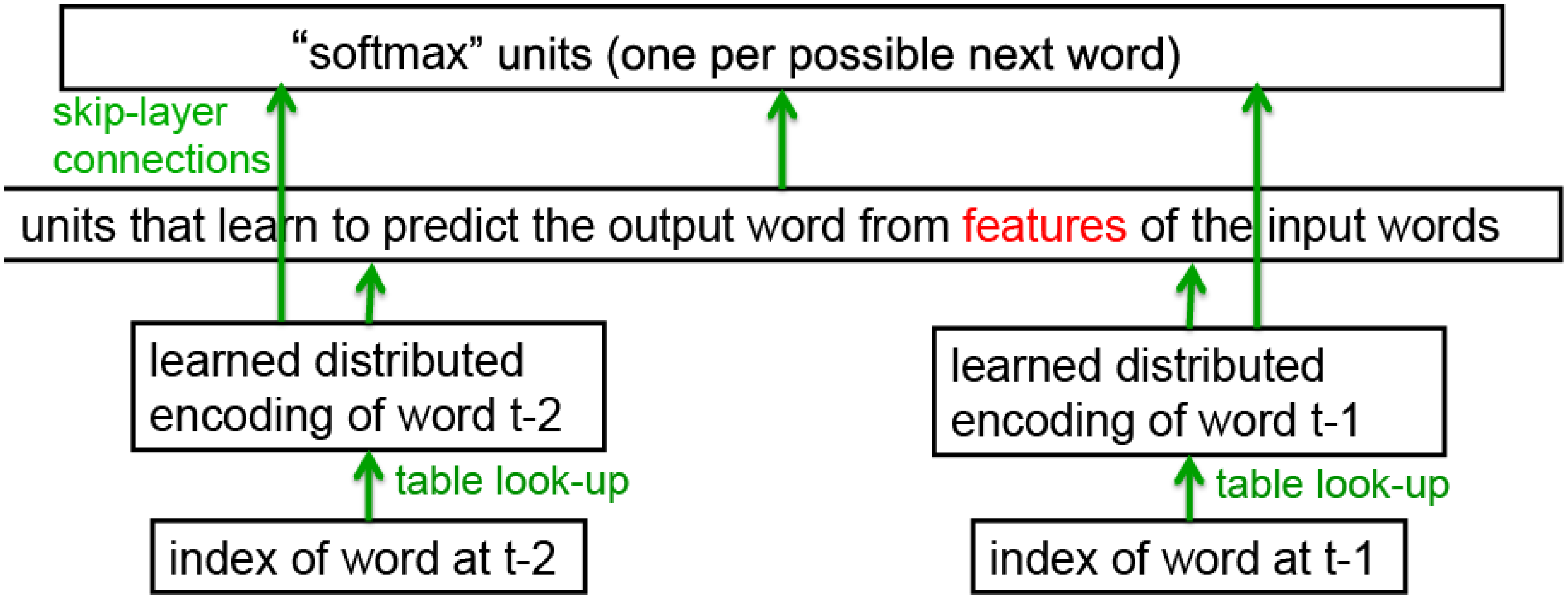
Let's Try Neural Networks for this Task



- Important issues
  - How should we encode the words to use them as input?
  - What internal NN structure do we need?
  - How can we perform classification (softmax) with so many possible outputs?

RWTH AACHEN UNIVERSITY

Neural Probabilistic Language Model



- **Core idea**
  - **Learn a shared distributed encoding (word embedding) for the words in the vocabulary.**

Y. Bengio, R. Ducharme, P. Vincent, C. Jauvin, A Neural Probabilistic Language Model, In JMLR, Vol. 3, pp. 1137-1155, 2003.

Word Embedding

- **Idea**
  - Encode each word as a vector in a d-dimensional feature space.
  - Typically, $V \sim 1\text{M}$, $d \in (50, 300)$
- **Learning goal**
  - Determine weight matrix $\mathbf{W}_{V \times d}$ that performs the embedding.
  - Shared between all input words
- **Input**
  - Vocabulary index $\mathbf{x}$ in 1-of-K encoding.
  - For each input $\mathbf{x}$, only one row of $\mathbf{W}_{V \times d}$ is needed.

$\Rightarrow$ $\mathbf{W}_{V \times d}$ is effectively a look-up table.

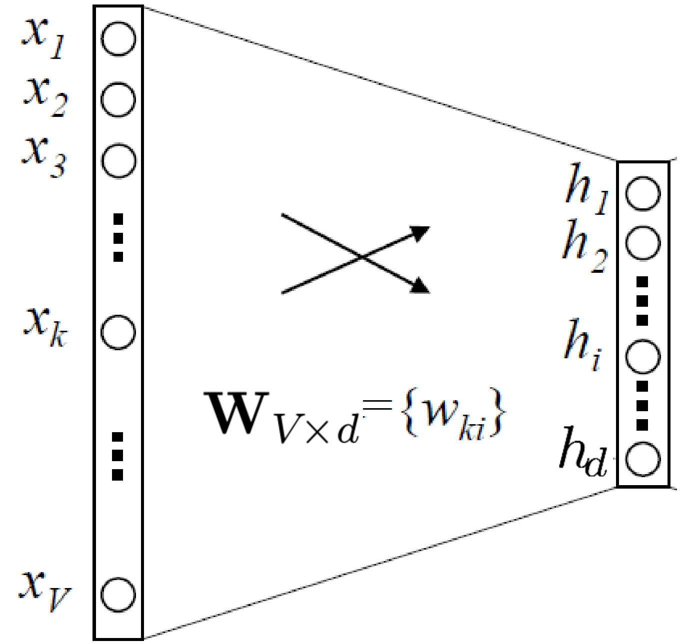
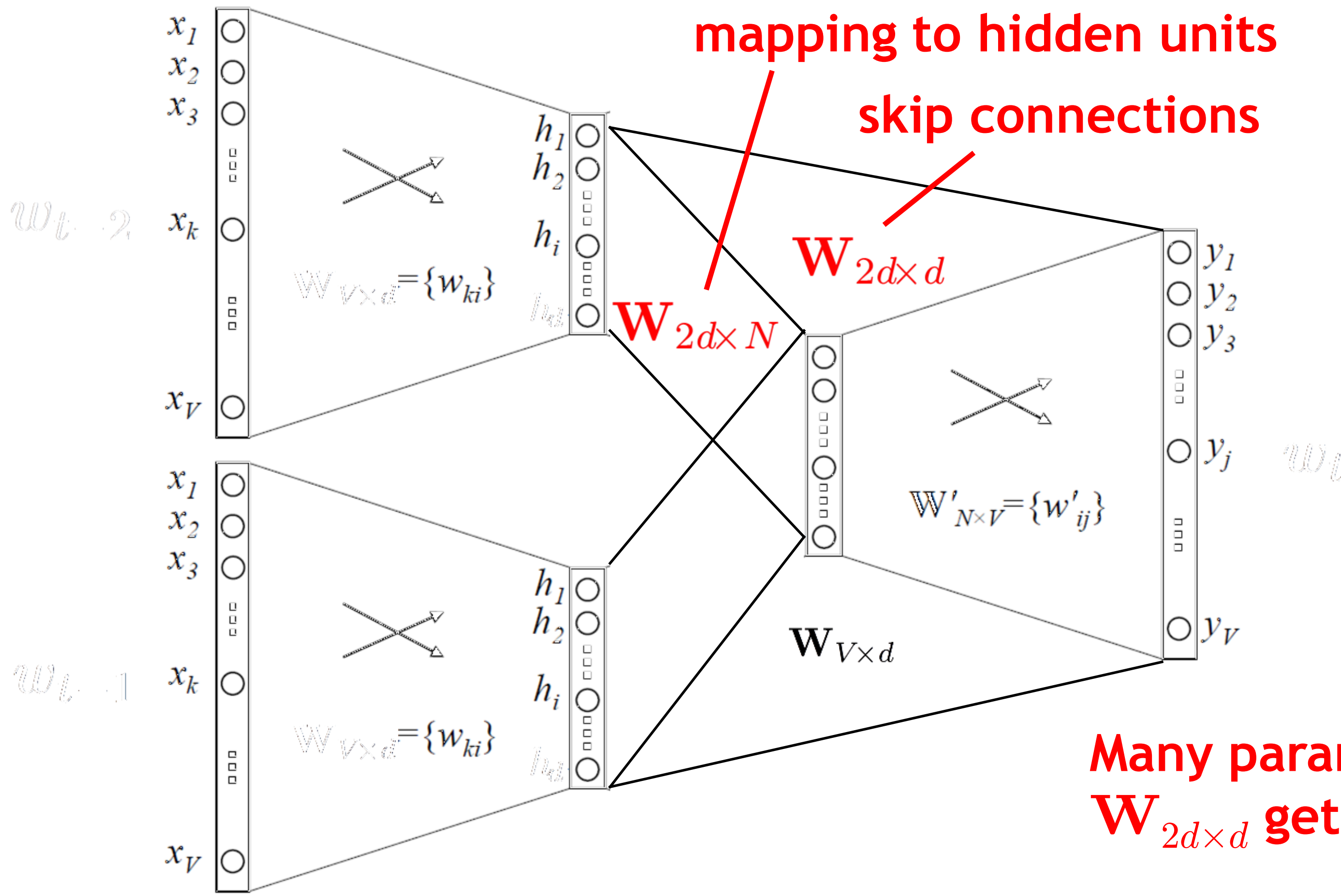


Image source: Xin Rong, 2015

Word Embedding: Full Network



Many parameters: $\mathbf{W}_{2d\times d}$ gets huge!

- Train on large corpus of data, learn $\mathbf{W}_{V\times d}$.
  - ⇒ Shown to outperform n-grams by [Bengio et al., 2003].

Visualization of the Resulting Embedding

(part of a 2.5D map of the most common 2500 words)

Image source: Geoff Hinton

Visualization of the Resulting Embedding

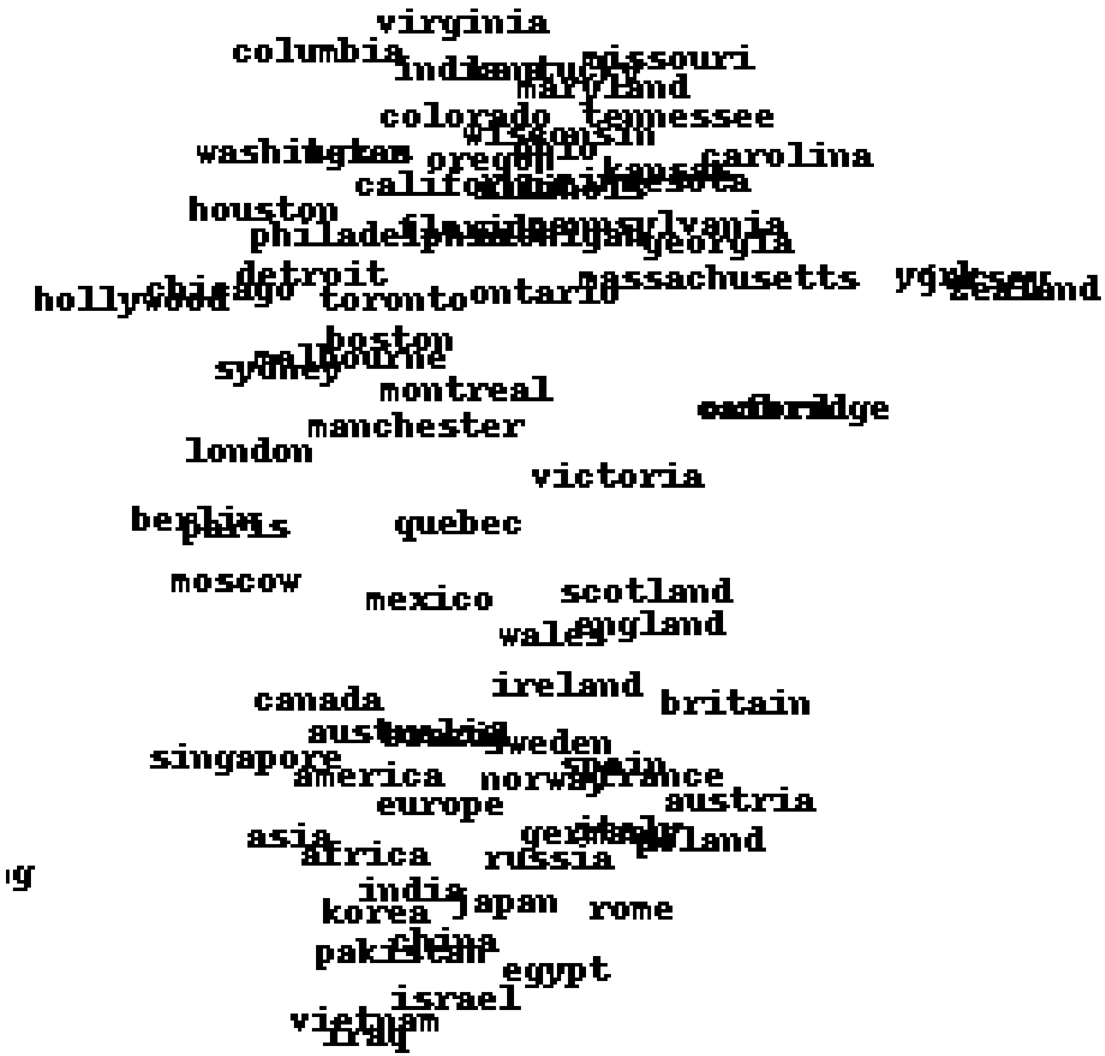


Image source: Geoff Hinton

Visualization of the Resulting Embedding

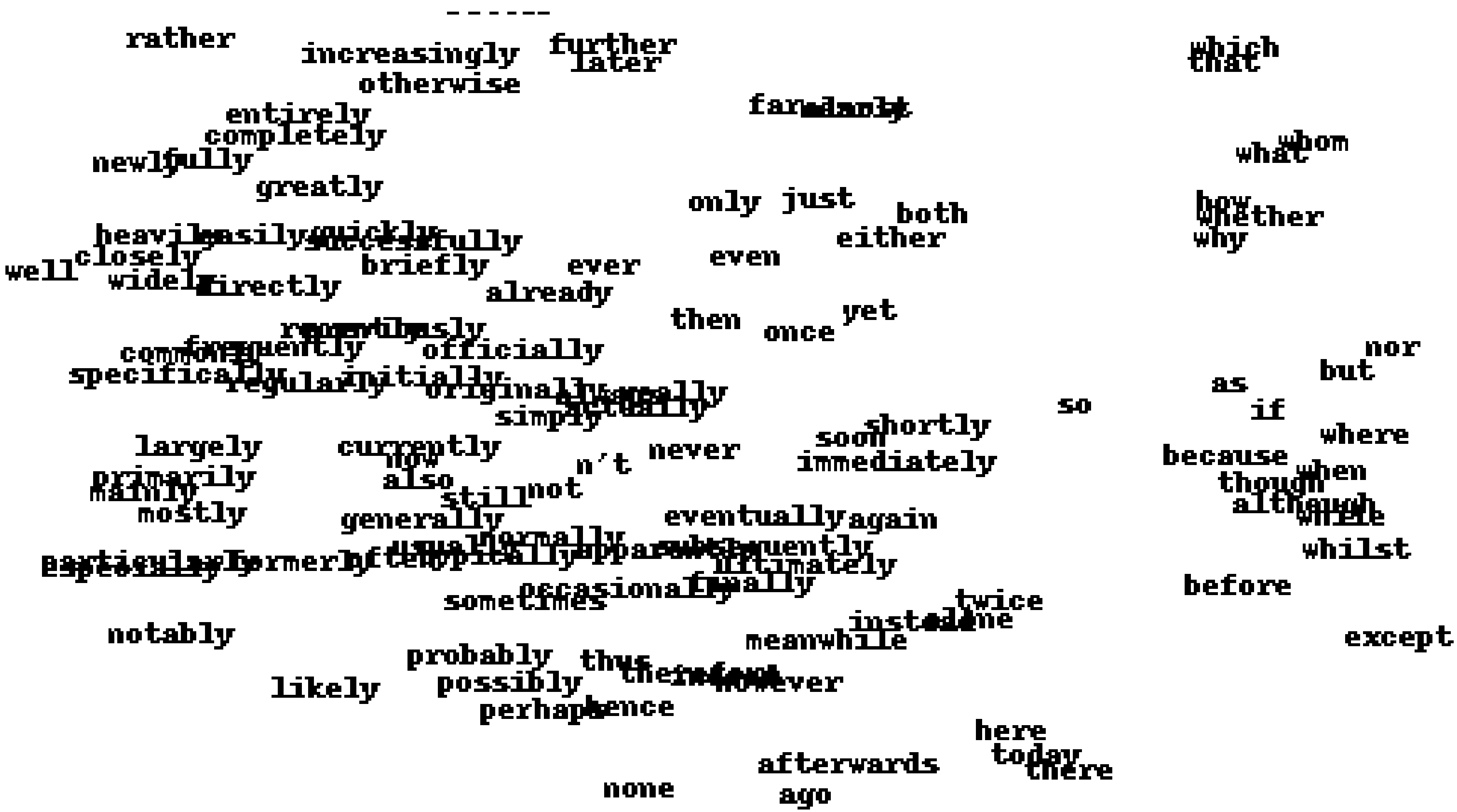


Image source: Geoff Hinton

Popular Word Embeddings

- Open issue
  - *What is the best setup for learning such an embedding from large amounts of data (billions of words)?*

- Several recent improvements
  - word2vec [Mikolov 2013]
  - GloVe [Pennington 2014]

  $\Rightarrow$ Pretrained embeddings available for everyone to download.

word2vec

- **Goal**
  - Make it possible to learn high-quality word embeddings from huge data sets (billions of words in training set).
- **Approach**
  - Define two alternative learning tasks for learning the embedding:
    - "Continuous Bag of Words" (CBOW)
    - "Skip-gram"
  - Designed to require fewer parameters.

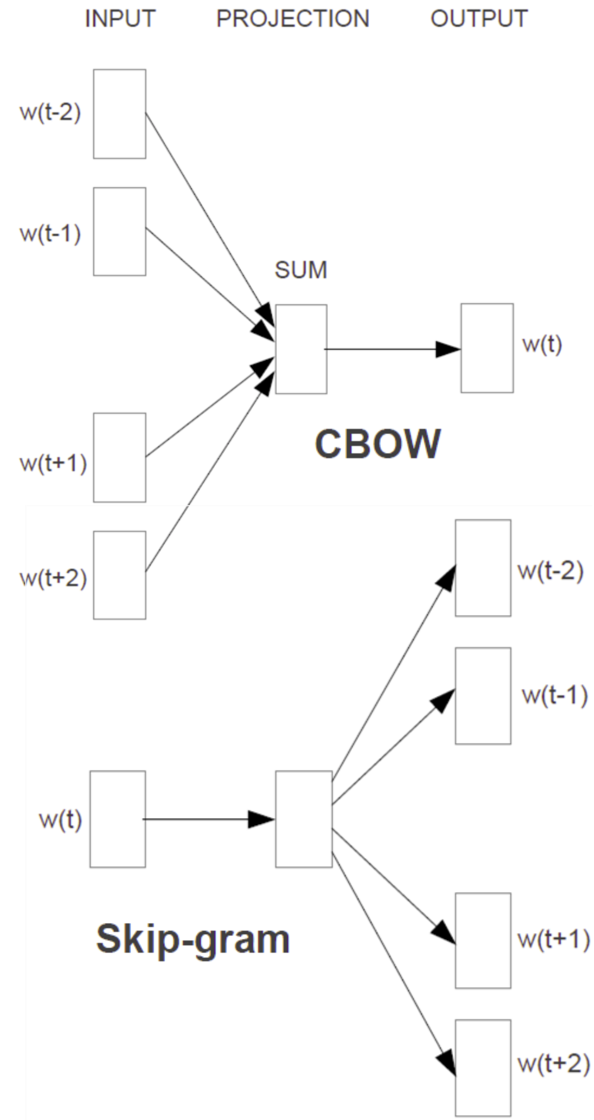


Image source: Mikolov et al., 2015

word2vec: CBOW Model

- Continuous BOW Model
  - Remove the non-linearity from the hidden layer
  - Share the projection layer for all words (their vectors are averaged)

⇒ Bag-of-Words model (order of the words does not matter anymore)

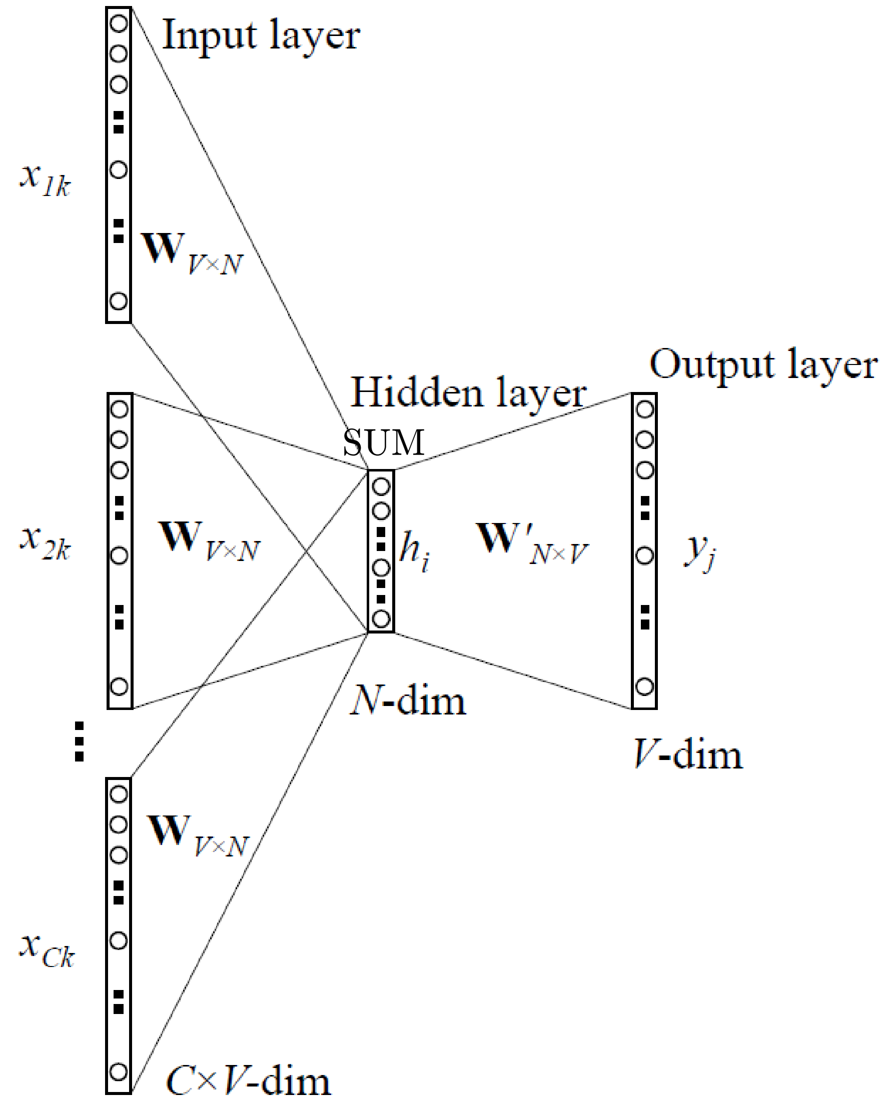
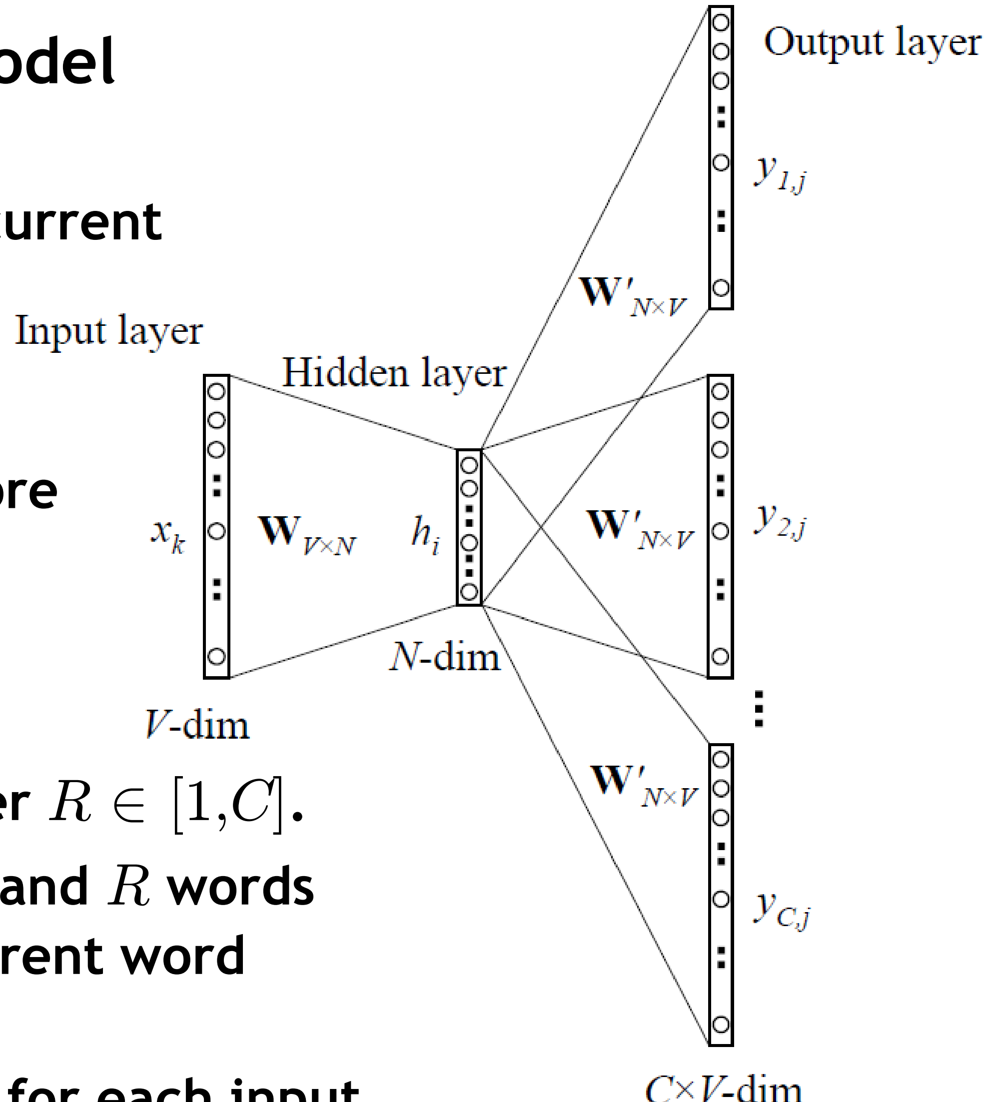


Image source: Xin Rong, 2015

word2vec: Skip-Gram Model

- **Continuous Skip-Gram Model**
  - Similar structure to CBOW
  - Instead of predicting the current word, predict words within a certain range of the current word.
  - Give less weight to the more distant words



- **Implementation**
  - Randomly choose a number $R \in [1,C]$.
  - Use R words from history and R words from the future of the current word as correct labels.

$\Rightarrow$ $R+R$ word classifications for each input.

Image source: Xin Rong, 2015

RWTH AACHEN UNIVERSITY

Interesting property

- Embedding often preserves linear regularities between words
  - Analogy questions can be answered through simple algebraic operations with the vector representation of words.
- Example
  - What is the word that is similar to *small* in the same sense as *bigger* is to *big*?
  - For this, we can simply compute

    X = vec("*bigger*") - vec("*big*") + vec("*small*")
  - Then search the vector space for the word closes to X using the cosine distance.

  $\Rightarrow$ Result (when words are well trained): vec("*smaller*").
- Other example
  - E.g., vec("King") - vec("Man") + vec("Woman") $\approx$ vec("Queen")

Evaluation on Analogy Questions

	Type of relationship	Word Pair 1		Word Pair 2	
semantic	Common capital city	Athens	Greece	Oslo	Norway
	All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
	Currency	Angola	kwanza	Iran	rial
	City-in-state	Chicago	Illinois	Stockton	California
	Man-Woman	brother	sister	grandson	granddaughter
syntactic	Adjective to adverb	apparent	apparently	rapid	rapidly
	Opposite	possibly	impossibly	ethical	unethical
	Comparative	great	greater	tough	tougher
	Superlative	easy	easiest	lucky	luckiest
	Present Participle	think	thinking	read	reading
	Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
	Past tense	walking	walked	swimming	swam
	Plural nouns	mouse	mice	dollar	dollars
	Plural verbs	work	works	speak	speaks

Image source: Mikolov et al., 2015

Results

Model	Vector Dimensionality	Training words	Accuracy [%]			Training time [days x CPU cores]
			Semantic	Syntactic	Total	
NNLM	100	6B	34.2	64.5	50.8	14 x 180
CBOW	1000	6B	57.3	68.9	63.7	2 x 140
Skip-gram	1000	6B	66.1	65.1	65.6	2.5 x 125

- **Results**
  - word2vec embedding is able to correctly answer many of those analogy questions.
  - CBOW structure better for syntactic tasks
  - Skip-gram structure better for semantic tasks

Image source: Mikolov et al., 2015

Problems with 100k-1M outputs

- **Weight matrix gets huge!**
- **Example: CBOW model**
  - One-hot encoding for inputs

  ⇒ Input-hidden connections are just vector lookups.

  - This is not the case for the hidden-output connections!
  - State h is not one-hot, and vocabulary size is 1M.

  ⇒ $\mathbf{W}'_{N \times V}$ has 300×1M entries

  ⇒ All of those need to be updated by backprop.

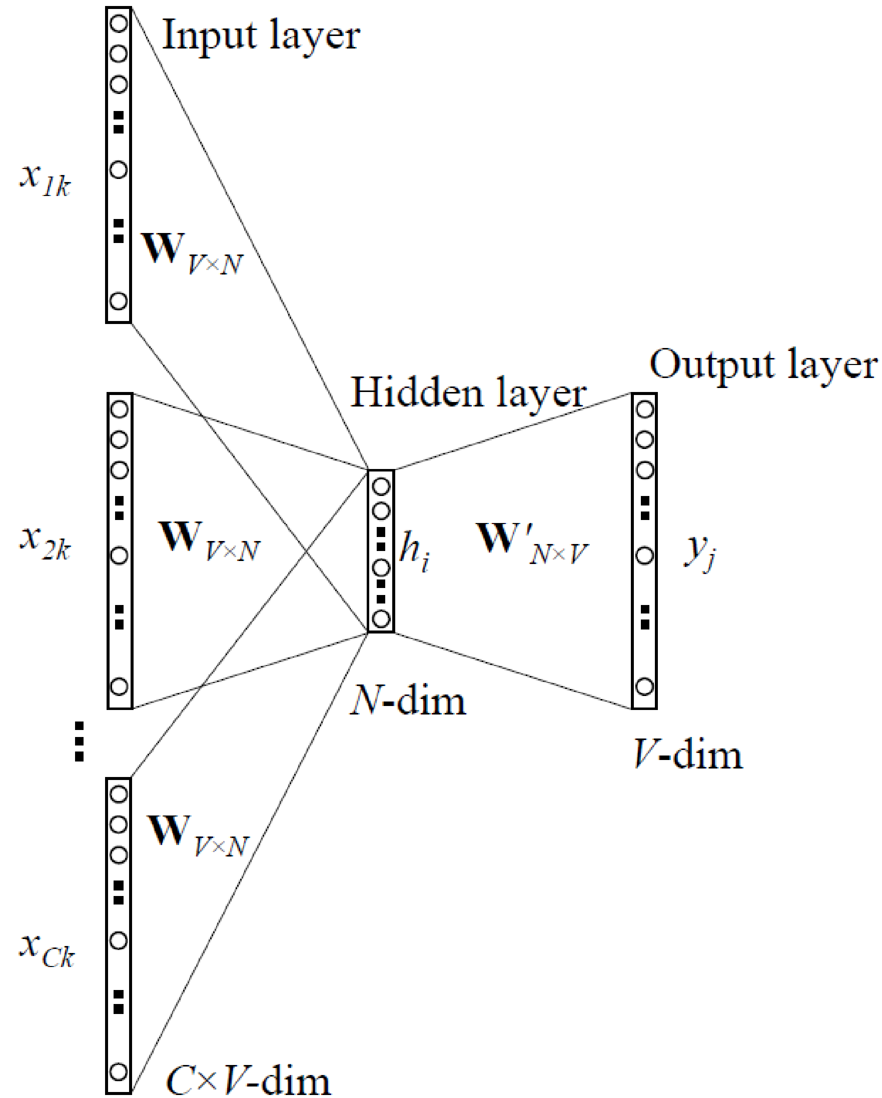


Image source: Xin Rong, 2015

Problems with 100k-1M outputs

- **Softmax gets expensive!**
  - Need to compute normalization over 100k-1M outputs

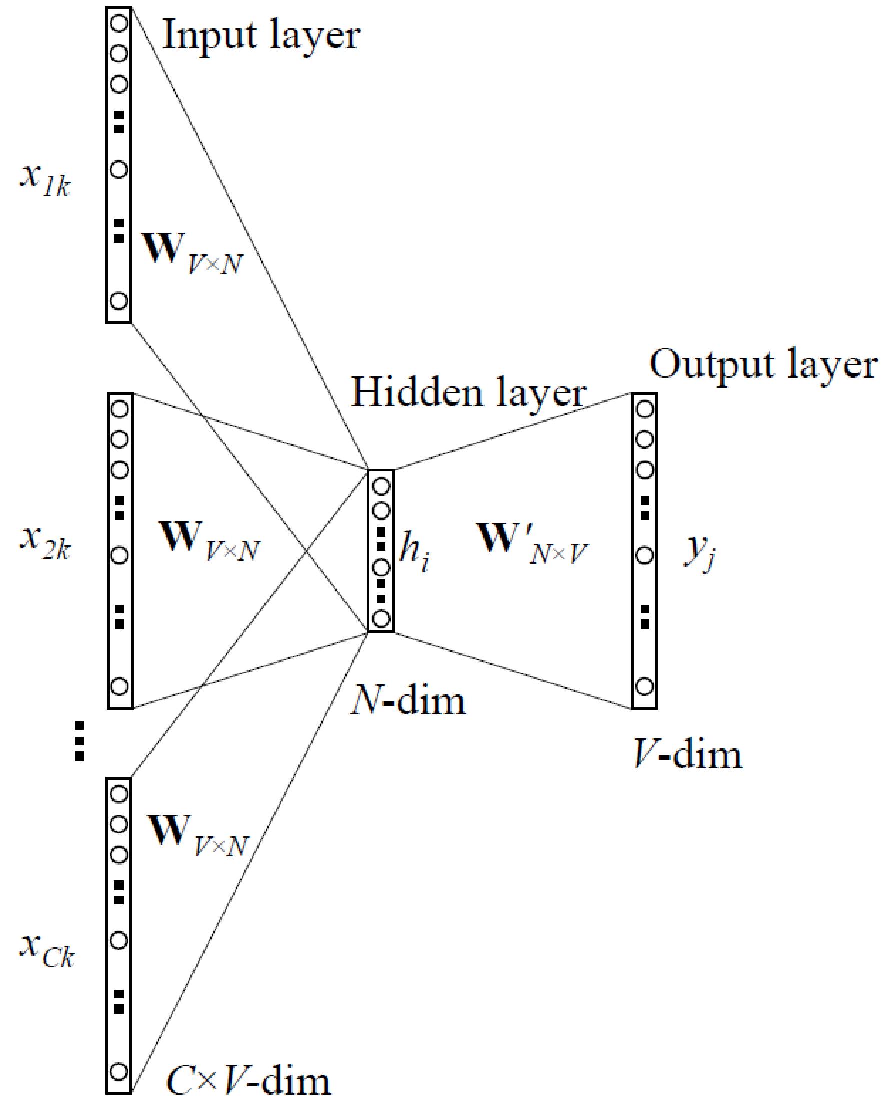
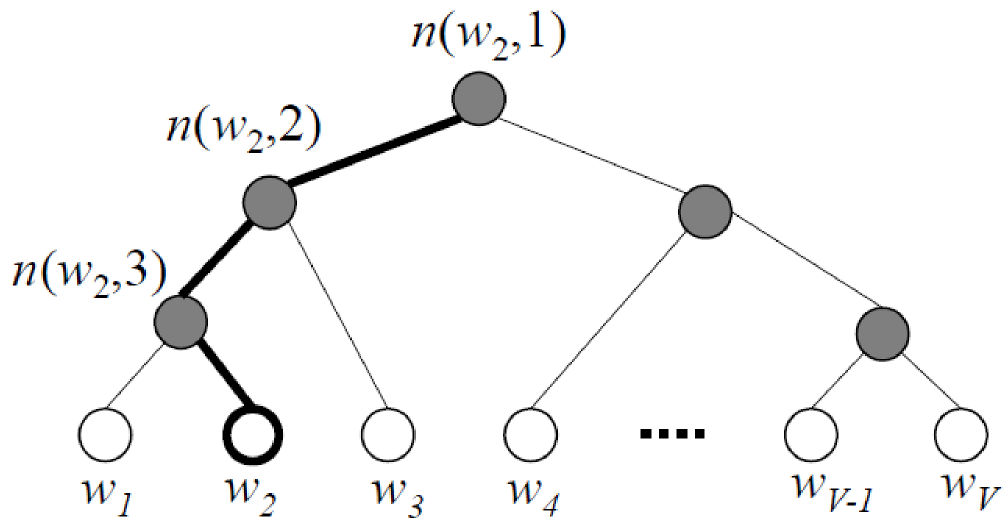


Image source: Xin Rong, 2015

RWTH AACHEN UNIVERSITY

Solution: Hierarchical Softmax



- **Idea**
  - Organize words in binary search tree, words are at leaves
  - Factorize probability of word w_0 as a product of node probabilities along the path.
  - Learn a linear decision function $y = v_{n(w,j)} \cdot h$ at each node to decide whether to proceed with left or right child node.

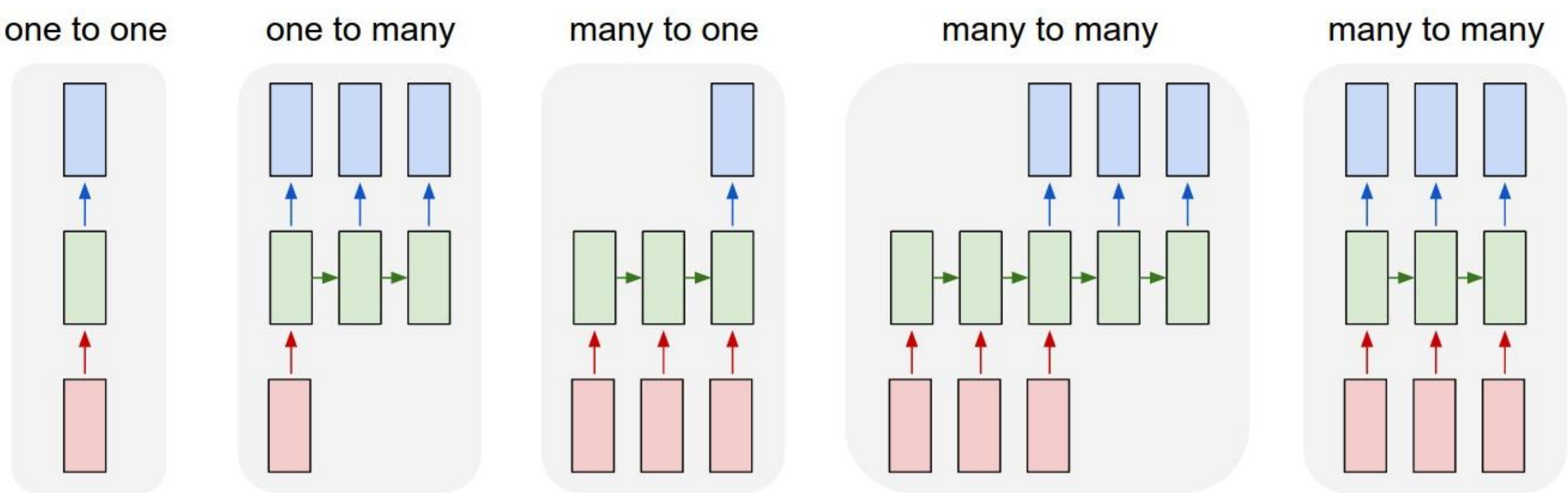
  $\Rightarrow$ Decision based on output vector of hidden units directly.

Image source: Xin Rong, 2015

Topics of This Lecture

- Recap: CNN Architectures
- Applications of CNNs
- Word Embeddings
  - Neuroprobabilistic Language Models
  - word2vec
  - GloVe
  - Hierarchical Softmax
- **Outlook: Recurrent Neural Networks**

Outlook: Recurrent Neural Networks



- **Up to now**
  - Simple neural network structure: 1-to-1 mapping of inputs to outputs
- **Next lecture: Recurrent Neural Networks**
  - Generalize this to arbitrary mappings

Image source: Andrej Karpathy

References and Further Reading

- **Neural Probabilistic Language Model**
  - Y. Bengio, R. Ducharme, P. Vincent, C. Jauvin, A Neural Probabilistic Language Model, In JMLR, Vol. 3, pp. 1137-1155, 2003.
- **word2vec**
  - T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient Estimation of Word Representations in Vector Space, ICLR'13 Workshop Proceedings, 2013.
- **GloVe**
  - Jeffrey Pennington, Richard Socher, and Christopher D. Manning, GloVe: Global Vectors for Word Representation, 2014.
- **Hierarchical Softmax**
  - F. Morin and Y. Bengio, Hierarchical probabilistic neural network language model. In AISTATS 2005.
  - A. Mnih and G.E. Hinton (2009). A scalable hierarchical distributed language model. In NIPS 2009.