PhD in Computer Science and Engineering
Bologna, April 2016

Machine Learning

Marco Lippi

marco.lippi3@unibo.it

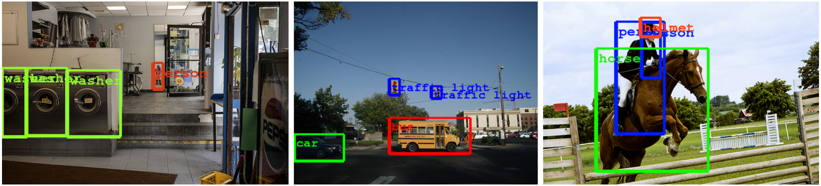
ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Convolutional Neural Networks

Convolutional neural networks

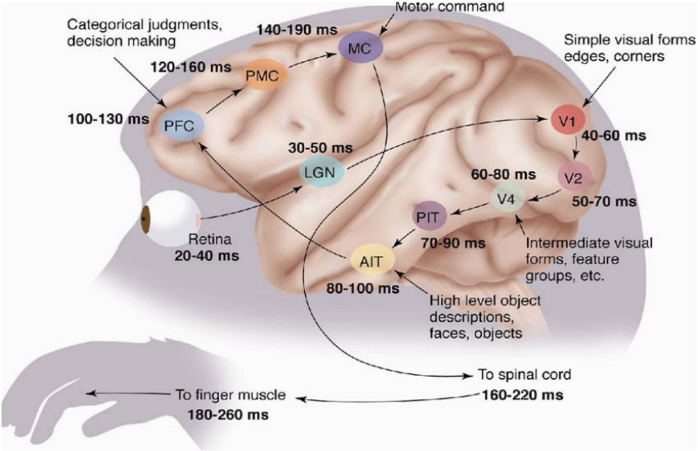
Architecture inspired by biological processes, focused on **vision**:

- neurons respond to overlapping regions in a **visual field**
- extremely **fast** computation (especially now with GPUs)
- pioneering models **back from the 80s-90s** !



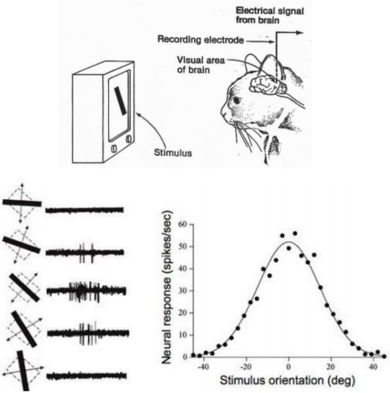
[Figure from Google Research]

The human visual cortex **is** hierarchical



[Figure from nyu.edu, Simon Thorpe]

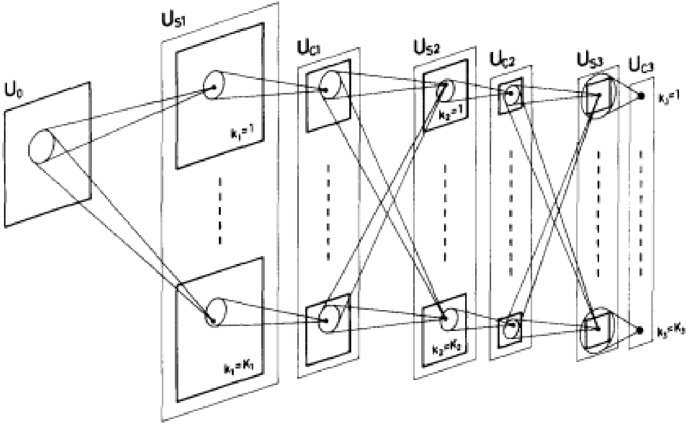
Hubel and Wiesel model (1962)



[Figure from nyu.edu]

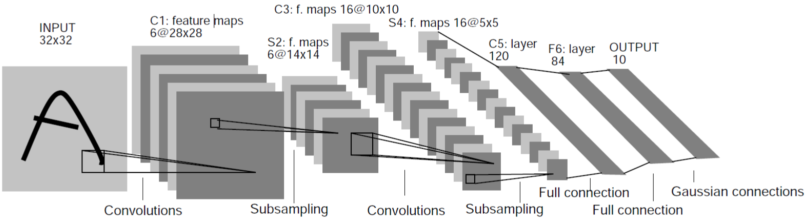
- **Simple cells** detect (edge-like) local features
- **Complex cells** receive input from simple cells and their receptive fields are spatially invariant

Cognitron and Neocognitron (Fukushima, 1974-1982)



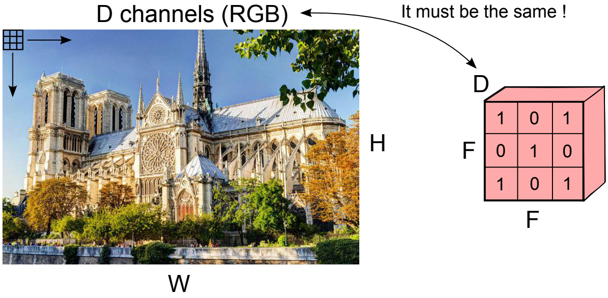
[Figure from Fukushima, 1980]

LeNet5 (LeCun et al., 1989)

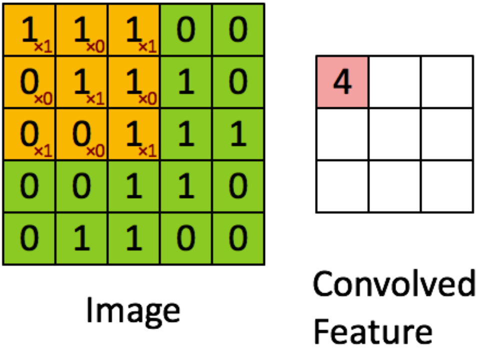


[Figure from LeCun et al., 1989]

Convolutional filter: a matrix (tensor) of weights to be applied on the image to perform **convolutions**

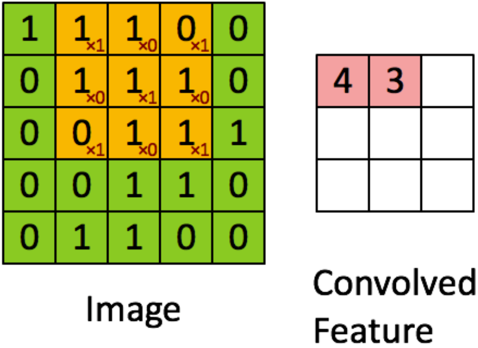


Convolution between image patch and filter



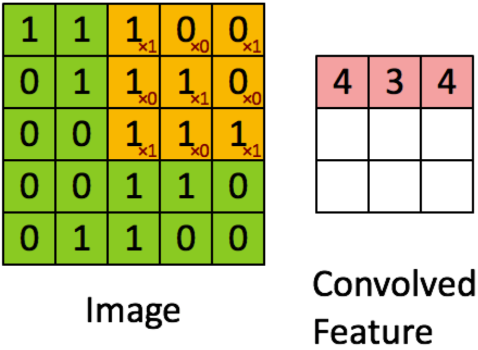
[Figure from http://deeplearning.stanford.edu/]

Convolution between image patch and filter



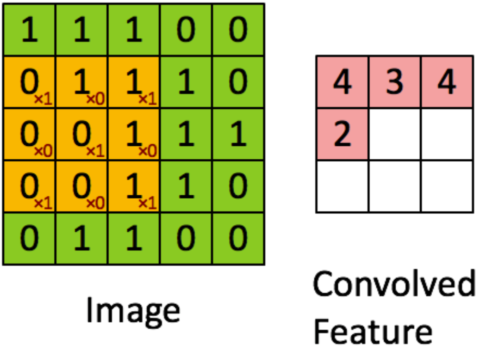
[Figure from http://deeplearning.stanford.edu/]

Convolution between image patch and filter



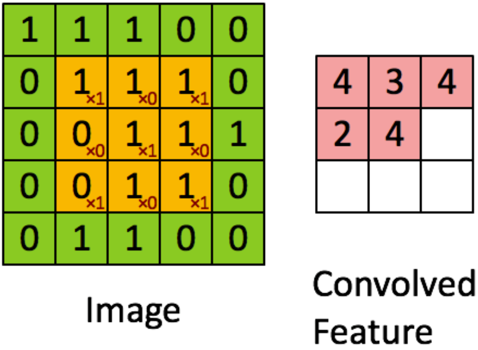
[Figure from http://deeplearning.stanford.edu/]

Convolution between image patch and filter



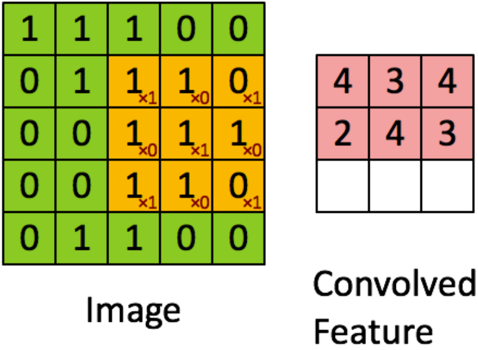
[Figure from http://deeplearning.stanford.edu/]

Convolution between image patch and filter



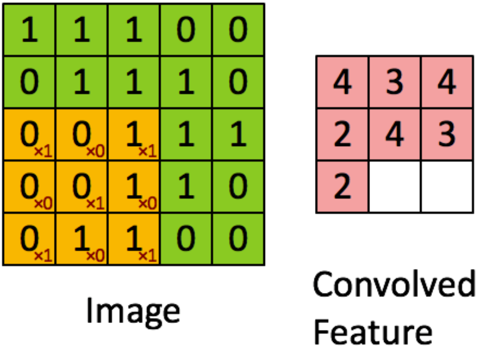
[Figure from http://deeplearning.stanford.edu/]

Convolution between image patch and filter



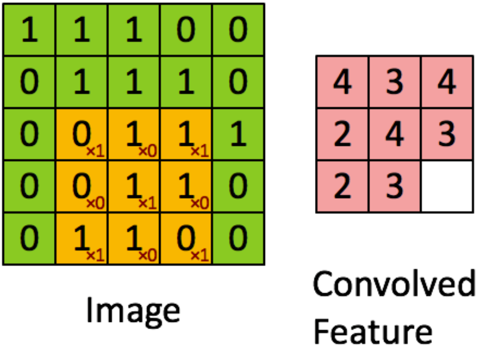
[Figure from http://deeplearning.stanford.edu/]

Convolution between image patch and filter



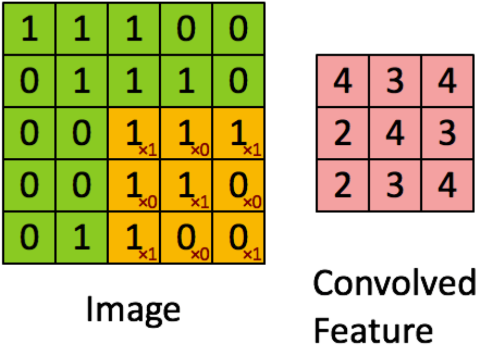
[Figure from http://deeplearning.stanford.edu/]

Convolution between image patch and filter



[Figure from http://deeplearning.stanford.edu/]

Convolution between image patch and filter



[Figure from http://deeplearning.stanford.edu/]

Receptive fields and convolutional filters

Stride

Hyper-parameter S indicating the "step" to be used when moving the filter on the image

- given a $W \times H$ image
- given a $F \times F$ filter
- $(W - F)/S$ and $(H - F)/S$ must be integers

Zero padding

Adding zeros along the border to allow convolutions on all pixels

- if $S = 1 \rightarrow$ zero padding with $(F - 1)/2$

Given a volume of size $W \times H \times D$

Choose hyper-parameters:

- # filters K
- filter dimension F
- stride S
- amount of zero padding P

Output is a volume of size $\hat{W} \times \hat{H} \times \hat{D}$ where:

- $\hat{W} = (W - F + 2P)/S + 1$
- $\hat{H} = (H - F + 2P)/S + 1$
- $D = K$

Common settings: K=2^m, F=3, S=1, P=1 (or F=5, S=1, P=2)

Rectification Layer

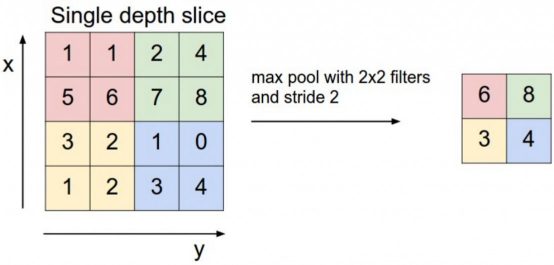
Typically placed **on top of a convolutional layer**:

- **drops to zero** negative inputs
- it is often added to further **augment non-linearity**
- operating on each activation map **independently**

Pooling Layer

An aggregation/subsampling function:

- operating on each activation map **independently**
- take the max/avg over a $M \times M$ filter with stride Z
- **no parameters to learn**
- just a **computation** above previous layer !



[Figure from http://deeplearning.stanford.edu/]

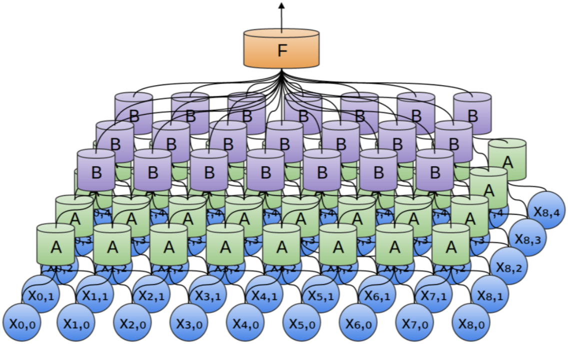
Local Contrast Normalization Layer

Recently introduced layer...

- operating on the output of max pooling
- subtracting mean and dividing by standard deviation of input
- this allows to obtain **brightness invariance**

Standard layer as in classic ANNs:

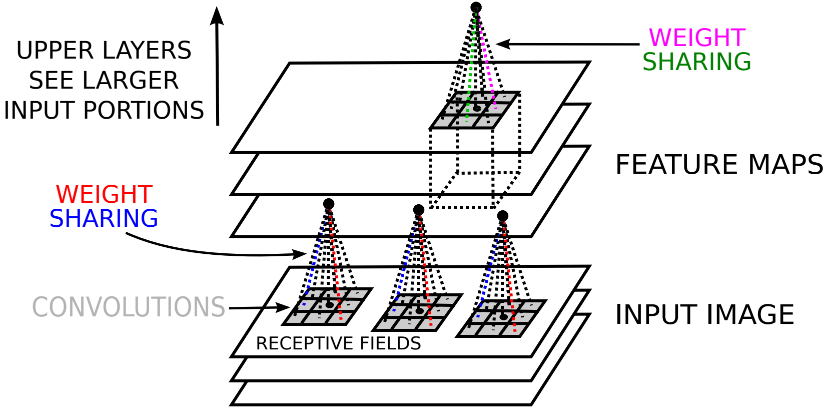
- **every** neuron connected to **every** neuron in previous layer
- typically implementing a **linear classifier**



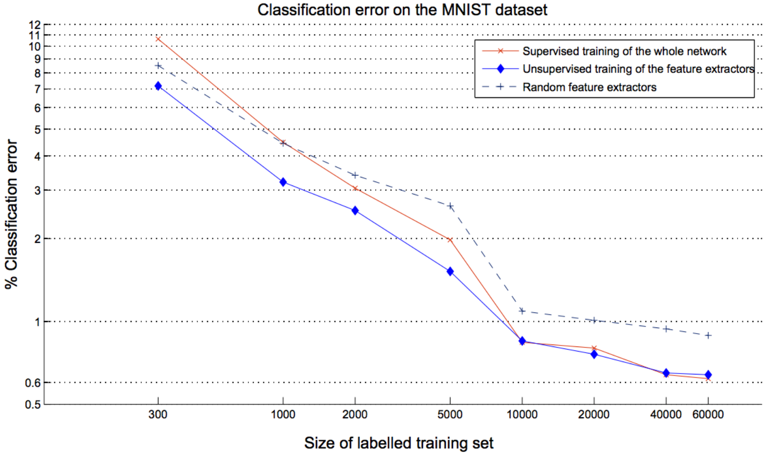
[Figure from colah.github.io]

Convolutional neural networks

One of the key advantages is to **share weights** between neurons !



A crucial advantage must be **in the structure** of a CNN !



[Figure from Ranzato et al., 2007]

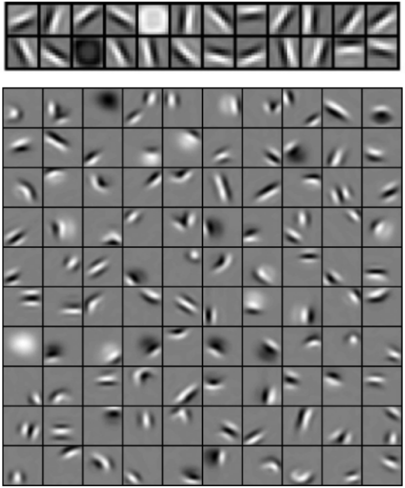
Properties

CNNs features have many nice properties:

- **compositionality** due to hierarchical structure
- **translation invariance** due to max pooling
- **scale invariance** via sub-sample processing

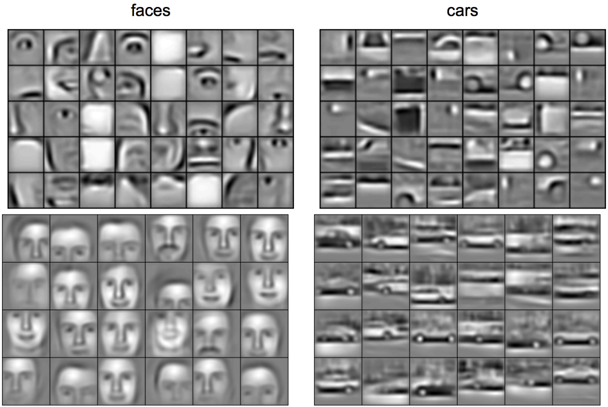
Feature extraction

Features obtained with various natural images



[Figures by Lee et al., 2009]

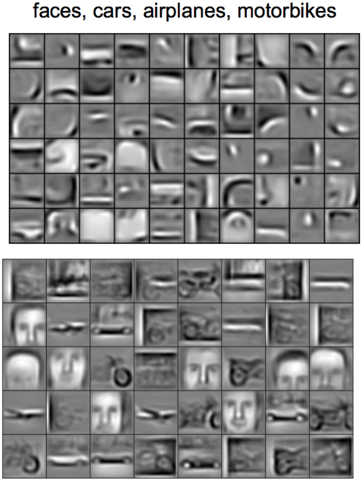
Features obtained with images belonging only to a given category



[Figures by Lee et al., 2009]

Feature extraction

Features obtained with images belonging only to a set of categories



[Figures by Lee et al., 2009]

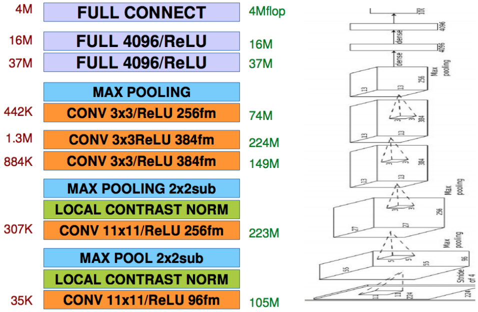
The dream: build a computer vision system capable of **recognizing thousands of object categories**

- over 14 millions images
- over 20 thousands categories
- tagged via **crowdsourcing** !
- organized according to the **WordNet** noun hierarchy

[Figure from vision.stanford.edu]

Annual competition:
ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)

Breakthrough in 2012 by AlexNet [Krizevsky et al., 2012]*



[Slide from `YannLeCun`]

*2012 paper with 4,377 citations. . .

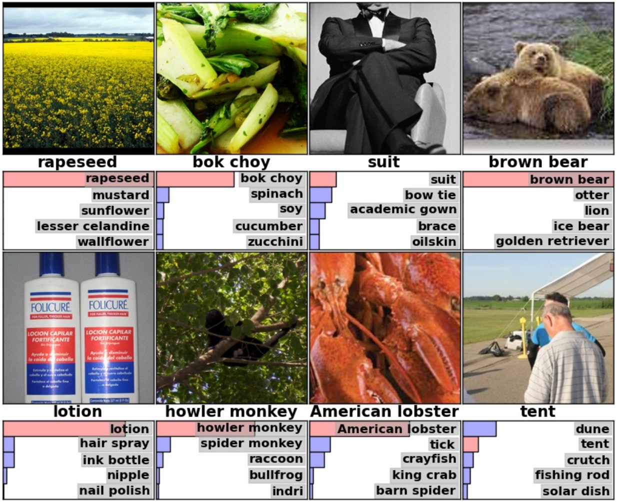
Breakthrough in 2012 by AlexNet [Krizevsky et al., 2012]

- $\sim$60,000,000 parameters
- $\sim$650,000 neurons
- trained for **5-6 days** with 2 GPUs in parallel
- achieved a top-5 error rate of 18.2 % (second best 26.2 %) reduced to 15.4 % with multiple models and pre-training
- achieved a top-1 error rate 40.7 % reduced to 36.7 % with multiple models and pre-training

Predictions

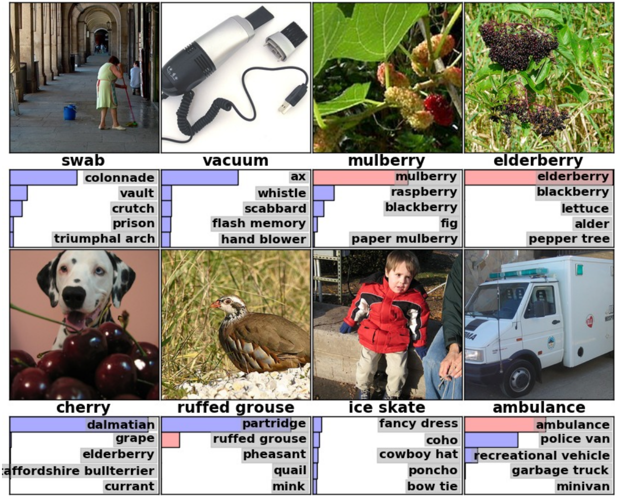
[Figure from Krizhevsky et al., 2012]

Predictions



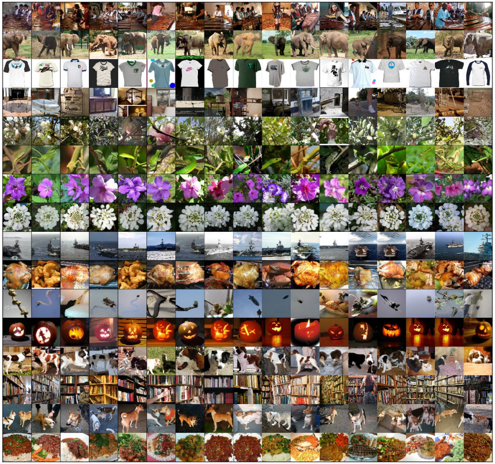
[Figure from Krizhevsky et al., 2012]

Predictions



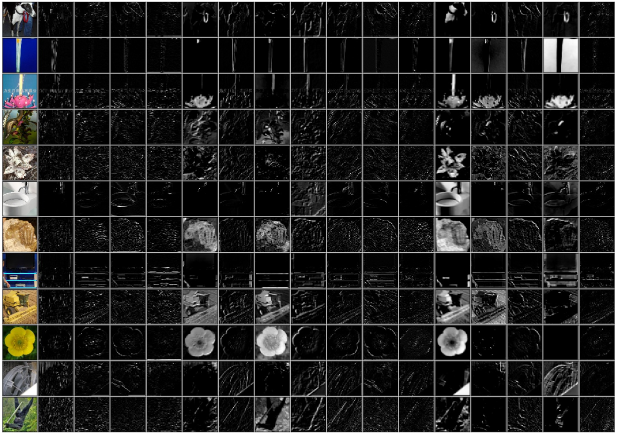
[Figure from Krizhevsky et al., 2012]

Retrieval



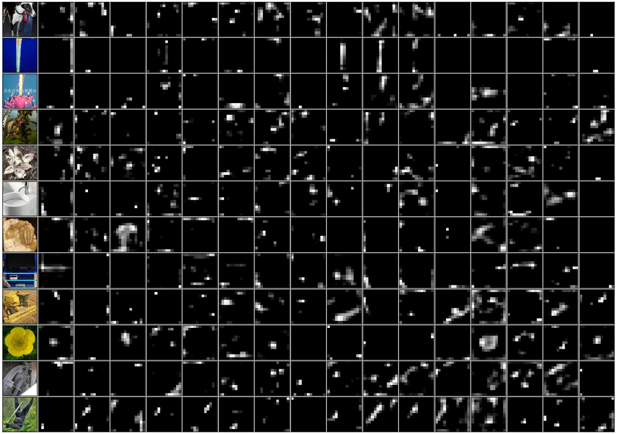
[Figure from Krizhevsky et al., 2012]

Feature maps (convolutional layer 1)



[Figure from Krizhevsky et al., 2012]

Feature maps (convolutional layer 1)



[Figure from Krizhevsky et al., 2012]

Features of the 2014 edition:

- 1.2 million images for training
- 50k images for validation
- 100k images for test
- 1,000 semantic categories
- Winner: **GoogLeNet** (22 layers !!!) $\rightarrow$ 6.67 % Top-5 error

In 2015 **ResNet** (Microsoft Research) $\rightarrow$ 3.6 % top-5 error !!!

- They employed a 152-layer net !!!
- They won all the tasks (localization, detection, segmentation)

Transfer learning with CNNs

A large number of **layers** implies a large number of **parameters**, thus a large number of **examples** needed to train the network.

What if one has only a small training set ?

- it has now become common to **pre-train** the network on a large dataset (e.g., ImageNet)
- **fine-tuning** is then performed on the (smaller) dataset related to a specific task

This is an instance of **transfer learning** !

Transfer learning (without CNNs)

[Torrey & Shavlik, 2009]

"Transfer learning is the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned."

- **Complex but crucial** for any machine learning system
- One of the tasks that **humans** are very good at
- Need to map features/relations **across domains**

Two possible scenarios:

1. use a CNN trained on ImageNet as a **fixed feature extractor** $\rightarrow$ this means to remove the last fully connected layer
2. also **fine-tune** some/all the weights of the CNN with the smaller dataset $\rightarrow$ be careful of **overfitting** !

A fine-tuning of the whole network might be required if our own dataset is **much different** from the one used in pre-training, **and not too small**. . .

Quite straightforward adaptation of CNNs:

- sliding window over the image
- multi-scale resolution

[Figure by Sermanet et al., 2013]

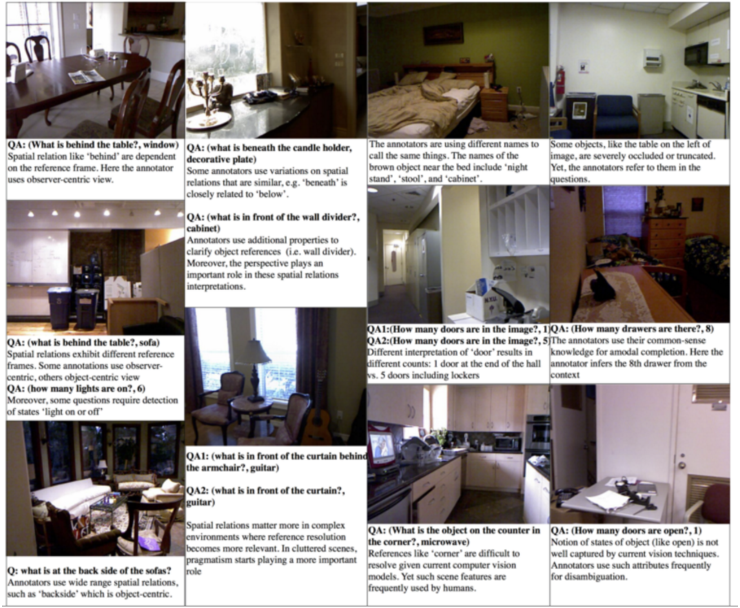
Tricks of the trade

Most of the tricks common to other kinds of deep networks

- dropout
- data augmentation (jittering, noise injection, . . .)
- weight decay
- sparsity of hidden units
- carefully choose learning rate

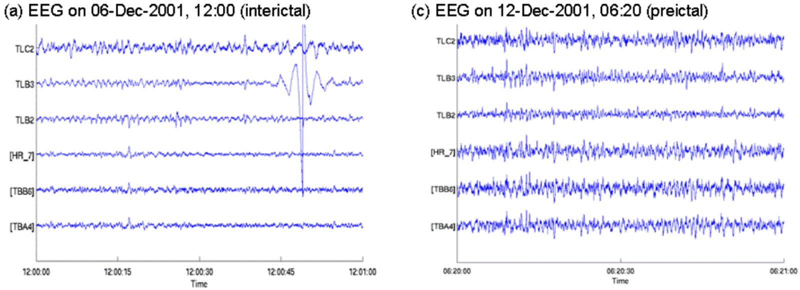
In addition. . .

- visualize **feature maps**
- visualize **parameters** (filters)


QA: (What is behind the table?, window)
Spatial relation like 'behind' are dependent on the reference frame. Here the annotator uses observer-centric view.
QA: (what is beneath the candle holder, decorative plate)
Some annotators use variations on spatial relations that are similar, e.g. 'beneath' is closely related to 'below'.
QA: (what is in front of the wall divider?, cabinet)
Annotators use additional properties to clarify object references (i.e. wall divider). Moreover, the perspective plays an important role in these spatial relations interpretations.
The annotators are using different names to call the same things. The names of the brown object near the bed include 'night stand', 'stool', and 'cabinet'.
Some objects, like the table on the left of image, are severely occluded or truncated. Yet, the annotators refer to them in the questions.
QA: (what is behind the table?, sofa)
Spatial relations exhibit different reference frames. Some annotations use observer-centric, others object-centric view
QA: (how many lights are on?, 6)
Moreover, some questions require detection of states 'light on or off'
QA1:(How many doors are in the image?, 1)
QA2:(How many doors are in the image?, 5)
Different interpretation of 'door' results in different counts: 1 door at the end of the hall vs. 5 doors including lockers
QA: (How many drawers are there?, 8)
The annotators use their common-sense knowledge for amodal completion. Here the annotator infers the 8th drawer from the context
Q: what is at the back side of the sofas?
Annotators use wide range spatial relations, such as 'backside' which is object-centric.
QA1: (what is in front of the curtain behind the armchair?, guitar)
QA2: (what is in front of the curtain?, guitar)
Spatial relations matter more in complex environments where reference resolution becomes more relevant. In cluttered scenes, pragmatism starts playing a more important role
QA: (What is the object on the counter in the corner?, microwave)
References like 'corner' are difficult to resolve given current computer vision models. Yet such scene features are frequently used by humans.
QA: (How many doors are open?, 1)
Notion of states of object (like open) is not well captured by current vision techniques. Annotators use such attributes frequently for disambiguation.

Prediction of epilepsy seizures from intra-cranial EEG

- Temporal CNNs [Mirowski et al., 2008]

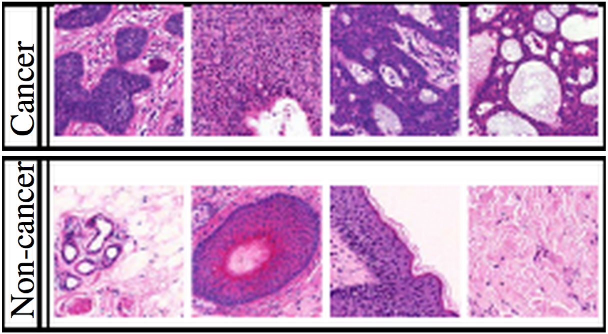


[Figure by Mirowski et al.]

Applications to bioinformatics, neuroscience, etc...

Cancer diagnosis and classifications

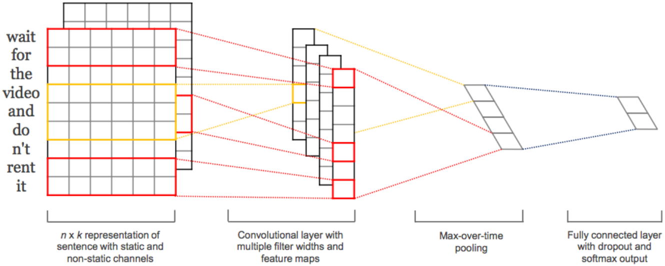
- Auto-Encoders [Fakoor et al., 2013]
- Convolutional Neural Networks [Cruz-Roa et al., 2013]



[Figure by Cruz-Roa et al.]

Sentence classification

- Sentiment analysis
- Question classification
- Subjectivity score
- ...



[Figure by Kim, 2014]

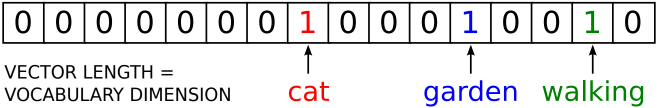
Word Embeddings, Language Models, etc.

In Natural Language Processing/Understanding (NLP/NLU), one crucial element is to **represent sentences** for the desired task

- Classification
- Segmentation
- Tagging
- ...

A classic representation is that of Bag-of-Words (Bow)

`The cat is walking in the garden`



Other BoW variants: consider **frequencies**

- frequency of a word **within a document**
  - Term Frequency (TF)

0	0	0	0	0	0	0	1	0	2	0	1	0	0	1	0
							cat		the		garden			walking	

VECTOR LENGTH =
VOCABULARY DIMENSION

- frequency of a word **within a corpus**:
  - Inverse Document Frequency (IDF) $\rightarrow$ TF $\times$ IDF

0	0	0	0	0	0	0	0.5	0	0	0	0.8	0	0	0.1	0
							cat				garden			walking	

VECTOR LENGTH =
VOCABULARY DIMENSION

Rare words in common are much more significant...

...But still, **it is not enough** !

A classic problem: not capturing **lexical** and **semantic** similarity !

```
The cat is walking in the garden
A dog was running towards the park
```

Almost **no similarity**: only article "the" in common...

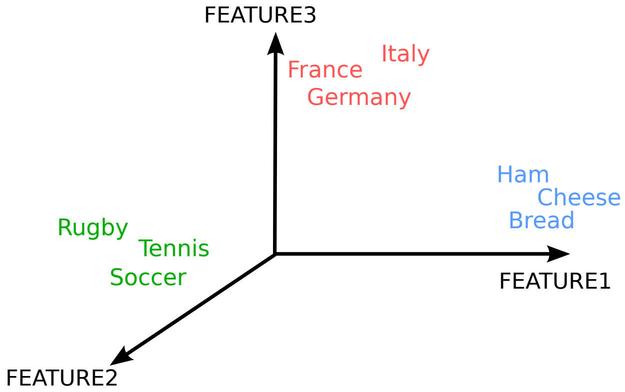
A lot of approaches have tried to include lexical/semantic features

- Use of ontologies (e.g., WordNet)
- Analysis of co-occurrences
- Word disambiguation (e.g., bank: river/finance ?)

Modeling language

"You shall know a word by the company it keeps" (J.R. Firth, 1957)

- Use **context** to learn **word representations** (embeddings)
- A word will be represented by a **dense real-valued vector**



Learn a **probability distribution** over sequences of words

The probability of **observing a sequence of words** $w_1, \dots w_m$ is:

$$P(w_1, \dots, w_m) = \prod_{i=1}^{m} P(w_i | w_1, \dots, w_{i-1})$$

A classic approach employs **n-grams**:

$$P(w_m | w_{m-1}, w_{m-2}, \dots, w_{m-n+1}) = \frac{count(w_{m-n+1}, \dots, w_m)}{count(w_{m-n+1}, \dots, w_{m-1})}$$

Just counting words. . .

Neural Network Language Model [Bengio et al., 2003]

The idea of neural embeddings dates back to over one decade:

- **predict next word** given current (and previous ones)

$$f(w_t, \ldots, w_{t-n+1}) = P(w_t | w_{t-1}, w_{t-2}, \ldots, w_{t-n+1})$$

Function f is decomposed in two parts:

- a mapping $C : |V| \rightarrow \Re^d$
- an ANN: $g(i, C(w_{t-1}), \ldots, C(w_{t-n+1})))$

C is the matrix of **word embeddings** (or word vectors)

Neural Network Language Model

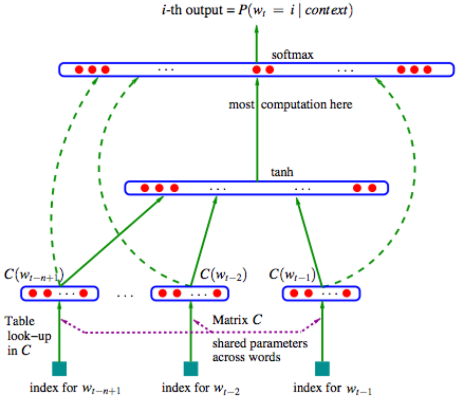


Figure 1: Neural architecture: $f(i, w_{t-1}, \cdots, w_{t-n+1}) = g(i, C(w_{t-1}), \cdots, C(w_{t-n+1}))$ where g is the neural network and $C(i)$ is the i-th word feature vector.

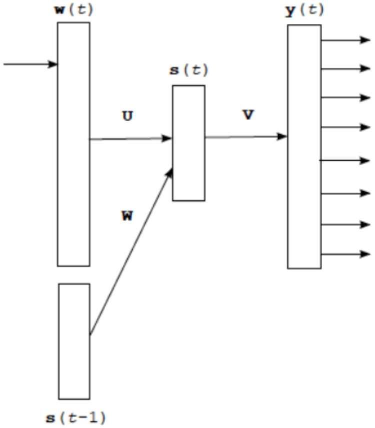
[Figure by Bengio et al., 2003]

Neural Network Language Model

Additional considerations...

- Output layer has $|V|$ size $\rightarrow$ quite expensive
- Reducing to $\log |V|$ via **hierarchical softmax**
- Partitioning the output spaces into a hierarchical structure
- Using a **bit vector encoding** of words (e.g., binary Huffman tree)
- Exploiting WordNet hierarchy

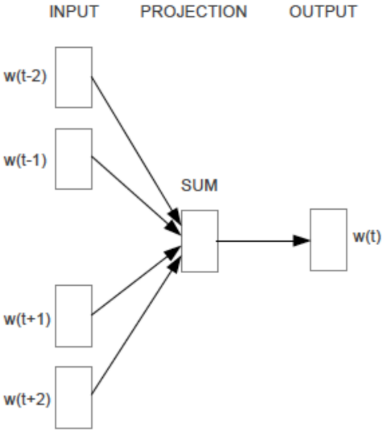
Next lecture !



[Figure by Mikolov et al., 2010]

Continuous bag-of-words

Predict word given past and future context



[Figure by Mikolov et al., 2013]

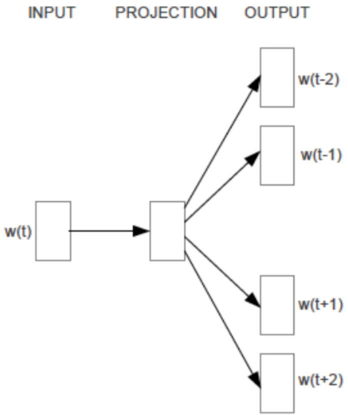
Continuous bag-of-words

Objective function: maximize the **average log probability**

$$\frac{1}{T}\sum_{t=k}^{T-k} \log p(w_t | w_{t-k}, \ldots, w_{t+k})$$

Skip-gram

Predict contextual words of a given word



[Figure by Mikolov et al., 2013]

Skip-gram

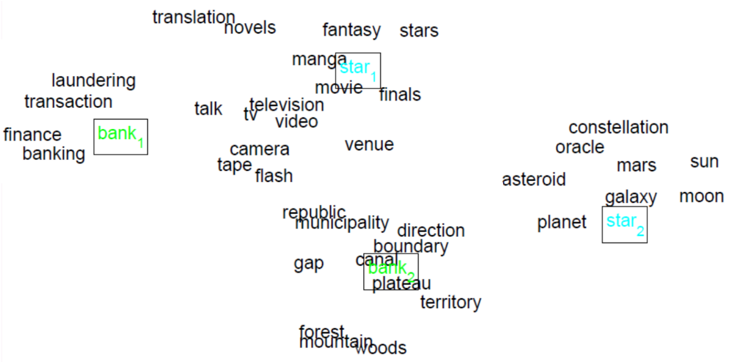
Objective function: maximize the **average log probability**

$$\frac{1}{T}\sum_{i=1}^{T}\sum_{-c\leq h\leq c,j\neq 0}\log p(w_{t+j}|w_t)$$

Trade-off to choose size of context c:

- if larger, the model is more accurate...
- ...but it is computationally more expensive

Word vectors (word2vec)

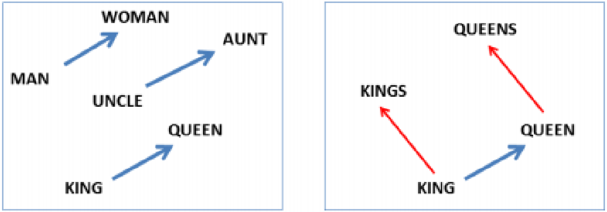


The power of word embeddings

FRANCE 454	JESUS 1973	XBOX 6909	REDDISH 11724	SCRATCHED 29869	MEGABITS 87025
AUSTRIA	GOD	AMIGA	GREENISH	NAILED	OCTETS
BELGIUM	SATI	PLAYSTATION	BLUISH	SMASHED	MB/S
GERMANY	CHRIST	MSX	PINKISH	PUNCHED	BIT/S
ITALY	SATAN	IPOD	PURPLISH	POPPED	BAUD
GREECE	KALI	SEGA	BROWNISH	CRIMPED	CARATS
SWEDEN	INDRA	PSNUMBER	GREYISH	SCRAPED	KBIT/S
NORWAY	VISHNU	HD	GRAYISH	SCREWED	MEGAHERTZ
EUROPE	ANANDA	DREAMCAST	WHITISH	SECTIONED	MEGAPIXELS
HUNGARY	PARVATI	GEFORCE	SILVERY	SLASHED	GBIT/S
SWITZERLAND	GRACE	CAPCOM	YELLOWISH	RIPPED	AMPERES

[Table by R. Collobert et al., 2011]

The power of word embeddings



[Figure by T. Mikolov]

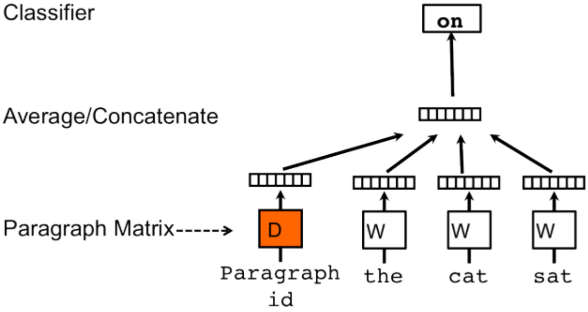
The power of word embeddings

Transfer learning !

- named entity recognition
- part-of-speech tagging
- parsing
- semantic role labeling
- machine translation

Generalization of word embeddings to any text

- W is the word embedding matrix
- D is the paragraph/document embedding matrix



[Figure by Le & Mikolov, 2014]

Word vectors (word2vec)

Pre-trained word vectors:

- **GloVe** (Global Vectors for word representation) @ Stanford
  http://nlp.stanford.edu/projects/glove/
  Different versions (Wikipedia, Twitter, Common Crawl)
- **word2vec** @ Google
  https://code.google.com/archive/p/word2vec/
  Trained on GoogleNews
  Naming version trained on FreeBase
- ...