

# Data Mining

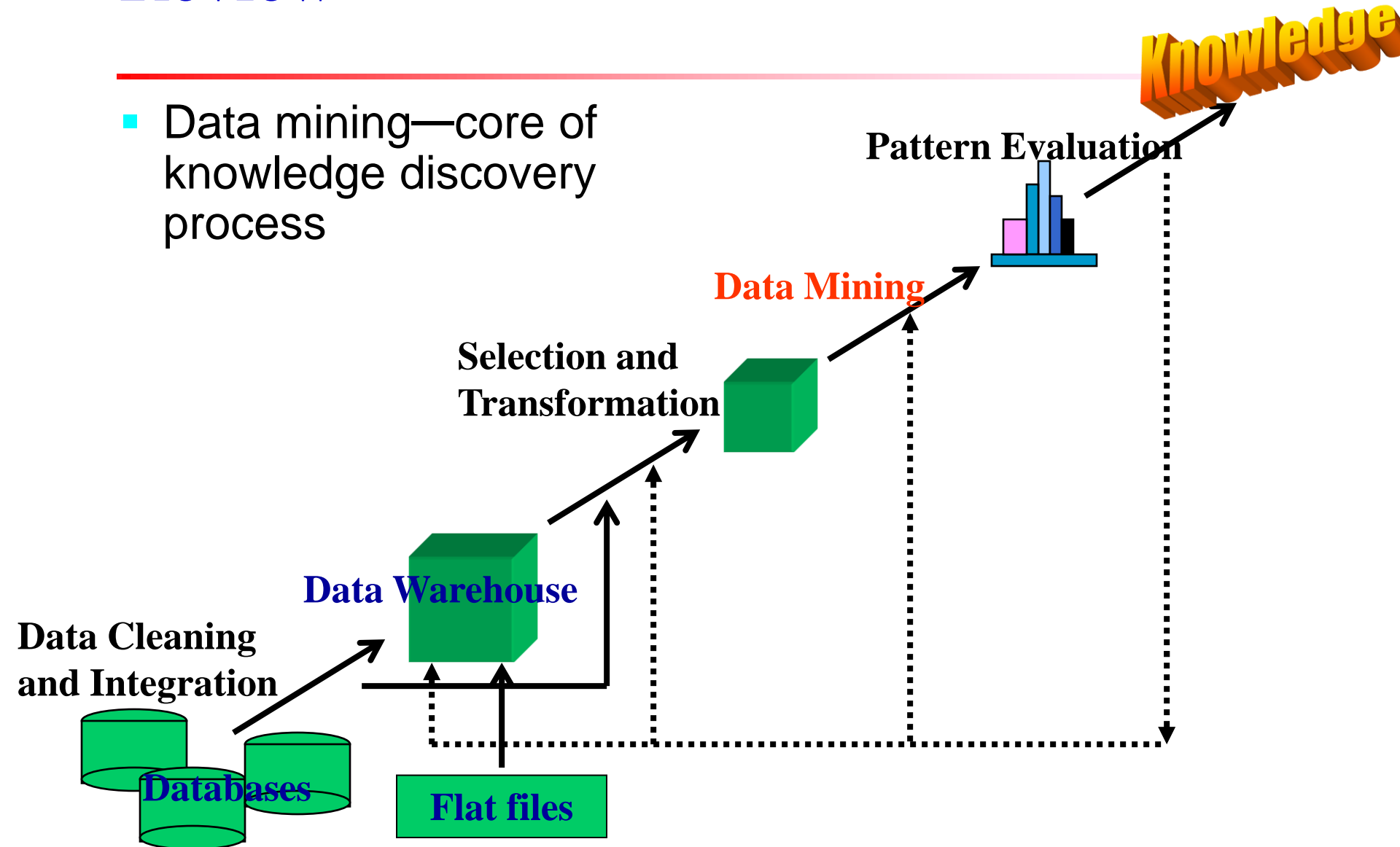
---

**Ying Liu, Prof., Ph.D**

*School of Computer Science and Technology  
University of Chinese Academy of Sciences  
Data Mining and High Performance Computing Lab*

# Review

- Data mining—core of knowledge discovery process

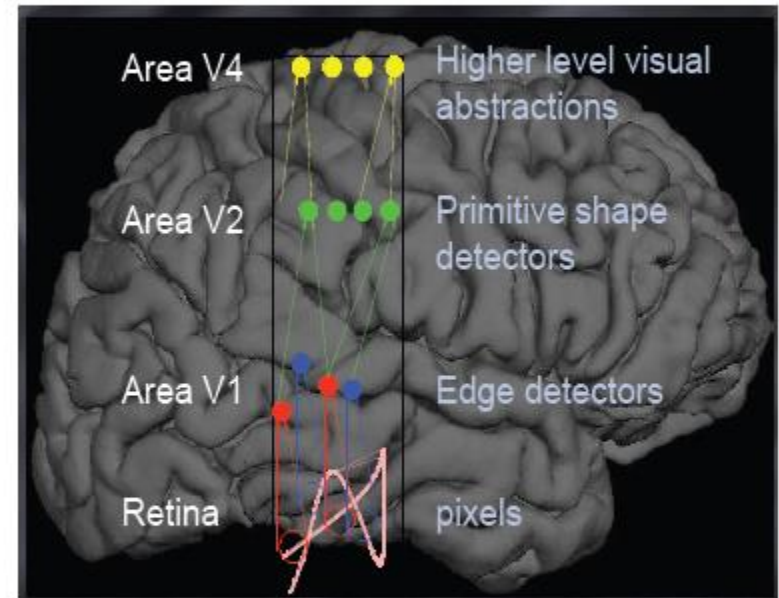
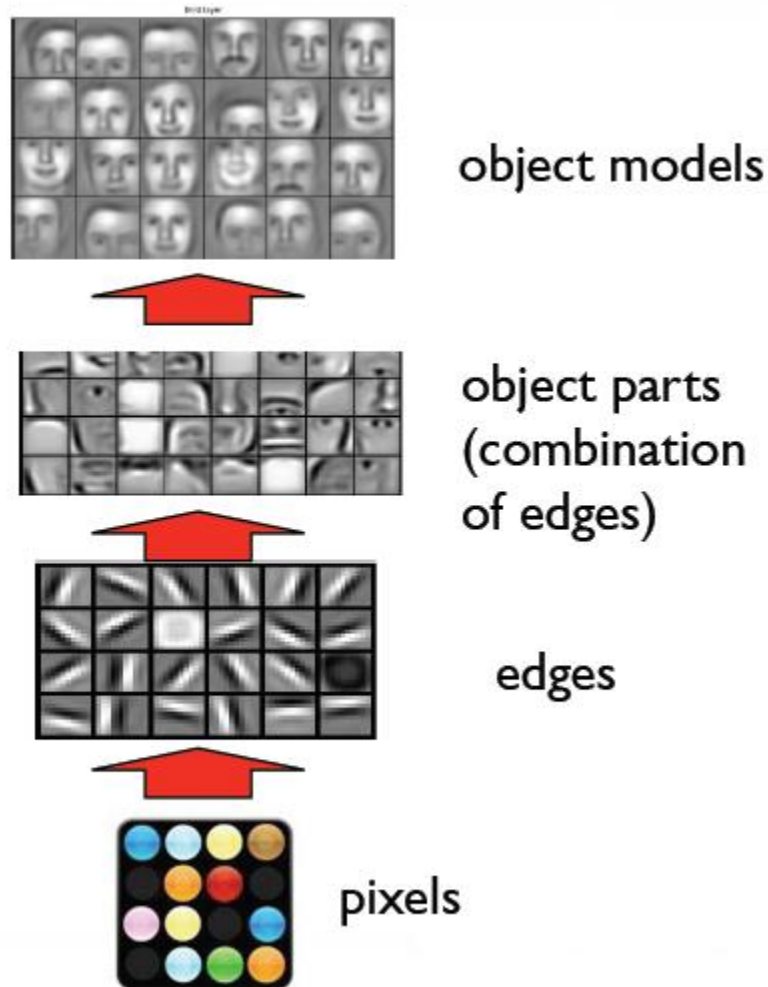


# Advanced Topics

---

- Deep learning
- High performance data mining
- Mining complex data types
- Data mining system products and research prototypes

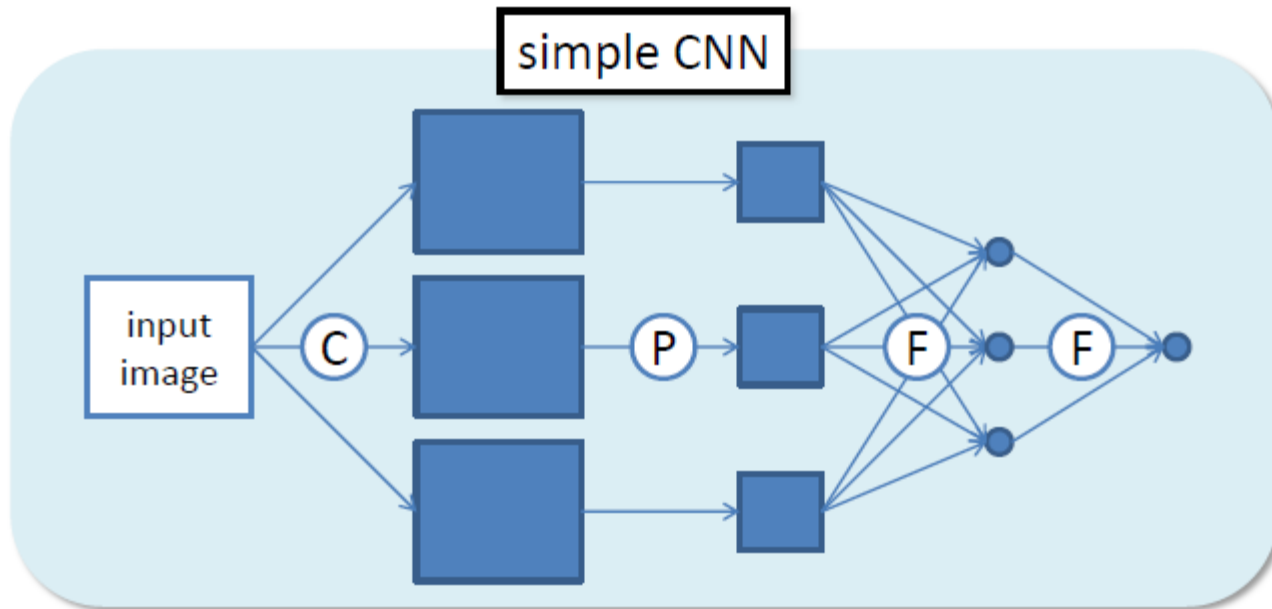
# Deep Learning vs. Human Brain



Slide credit: Andrew Ng

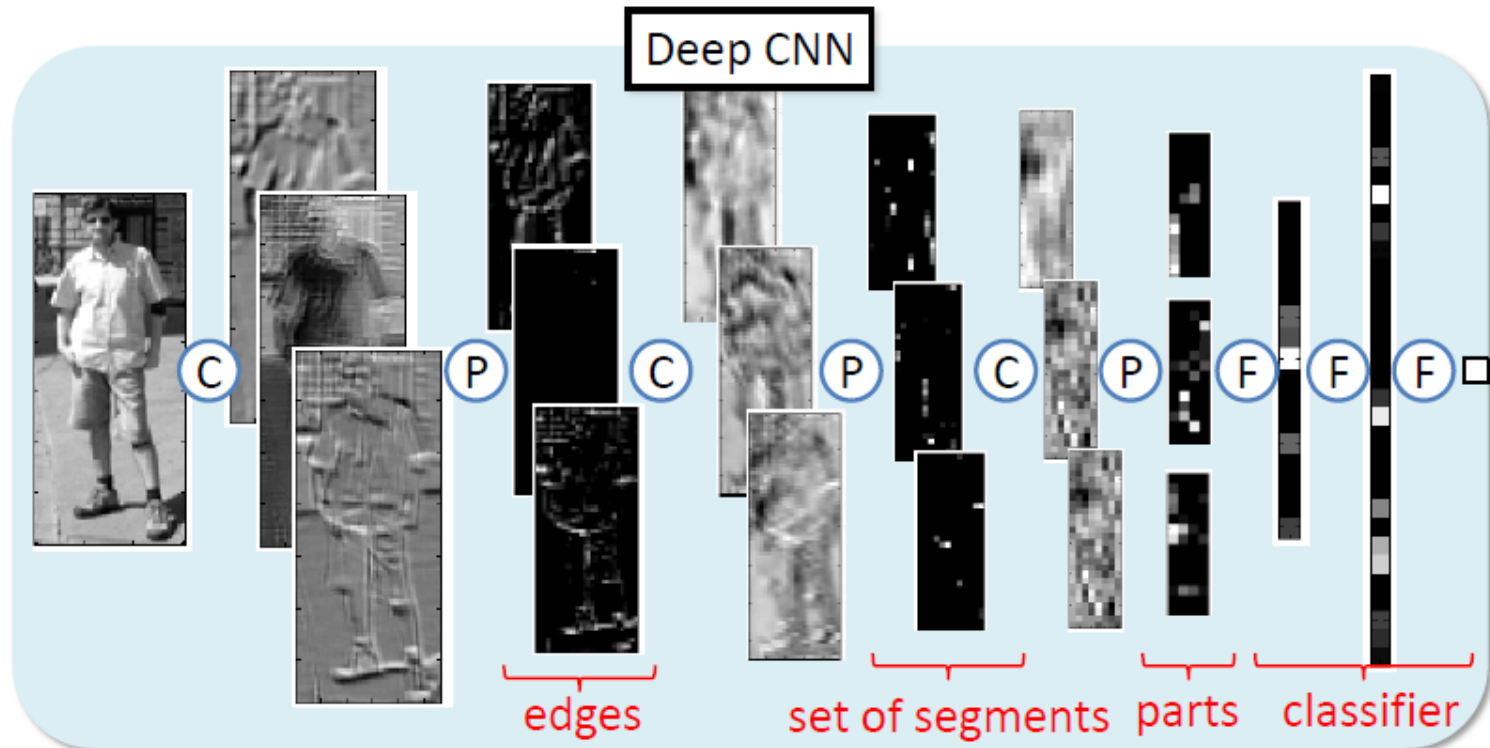
# Deep Neural Network

- **Convolutional Neural Network (CNN)** comprising
    - Convolutional layer(s) -> local feature extraction
    - Pooling layer(s) -> dimensionality reduction
    - Fully-connected layer(s) -> classification/regression
- } visual features automatically extracted



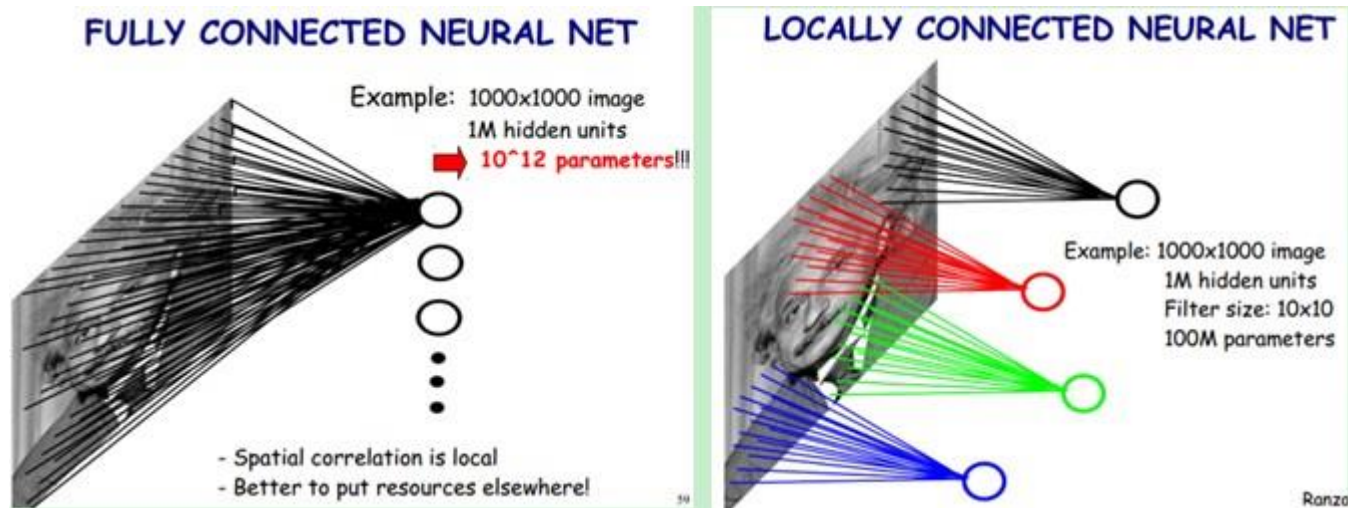
# Deep Neural Networks

- Transforming (images) data vectors at multiple layers help to increase the level of abstraction of visual contents



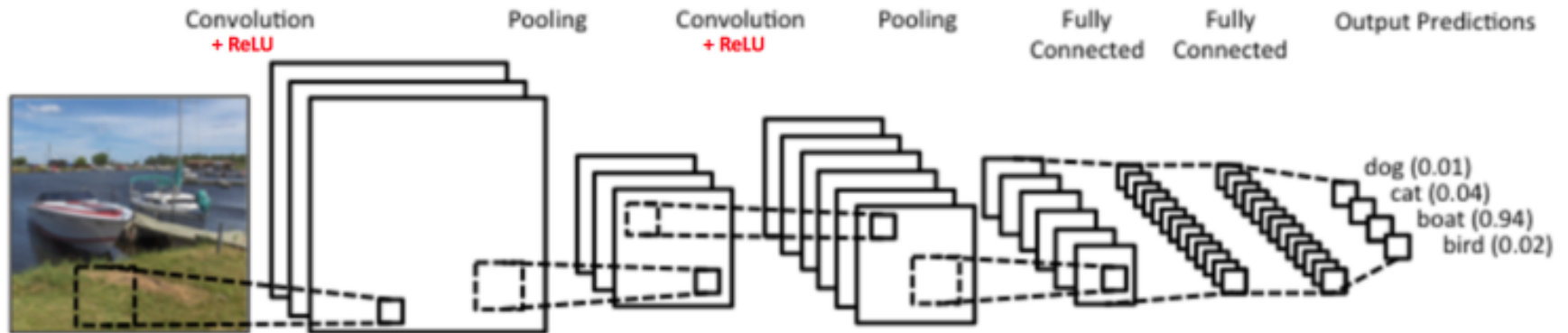
# Deep Neural Networks

- Extract features from images (data vectors)



# LeNet (1990s)

---

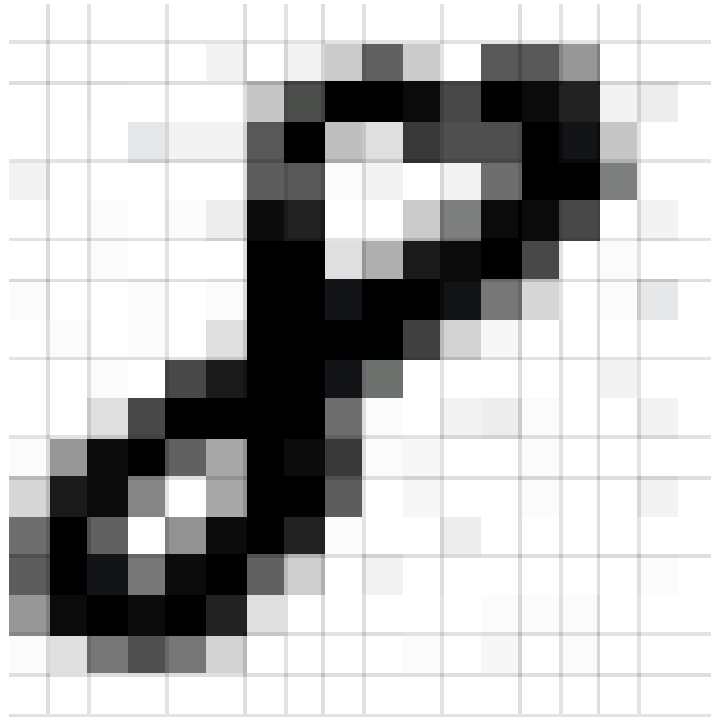




# Input Image

---

## ■ Pixels



# Convolution

---

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

1	0	1
0	1	0
1	0	1

# Convolution

---

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

1	0	1
0	1	0
1	0	1

1 <sub>x1</sub>	1 <sub>x0</sub>	1 <sub>x1</sub>	0	0
0 <sub>x0</sub>	1 <sub>x1</sub>	1 <sub>x0</sub>	1	0
0 <sub>x1</sub>	0 <sub>x0</sub>	1 <sub>x1</sub>	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		



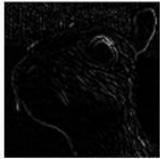

Convolved  
Feature

# Convolution

---



# Convolution

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

# Convolution

---



Input

# Convolution



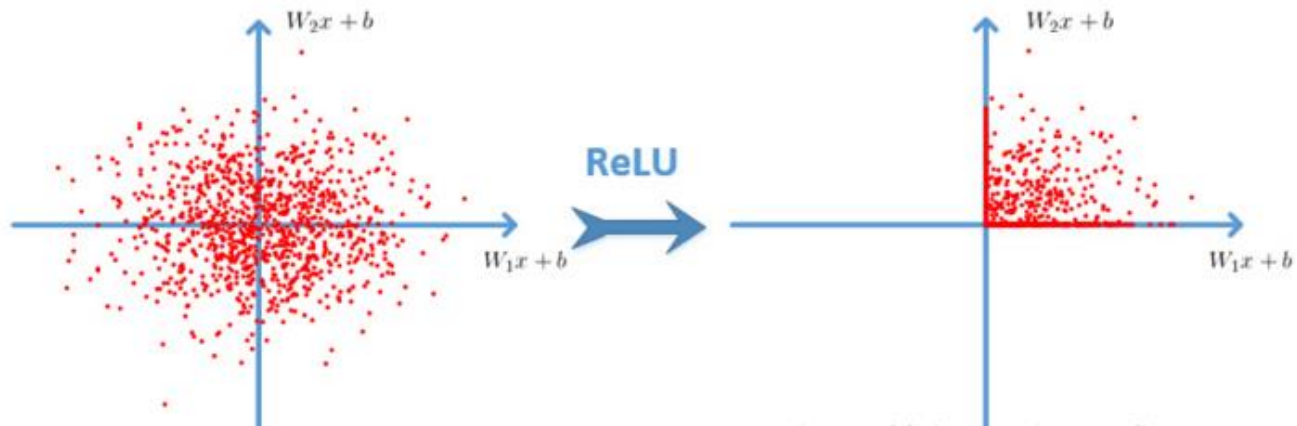
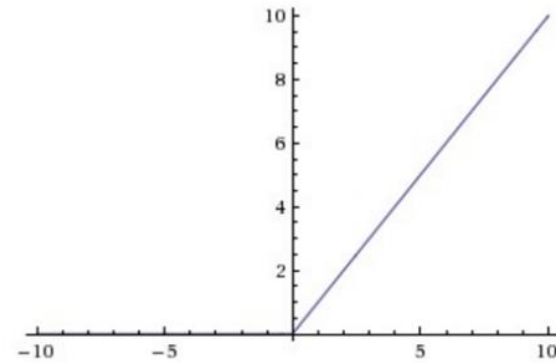
Feature Map having  
depth of 3 (since 3  
filters have been used)

Convolution  
Operation

# ReLU

---

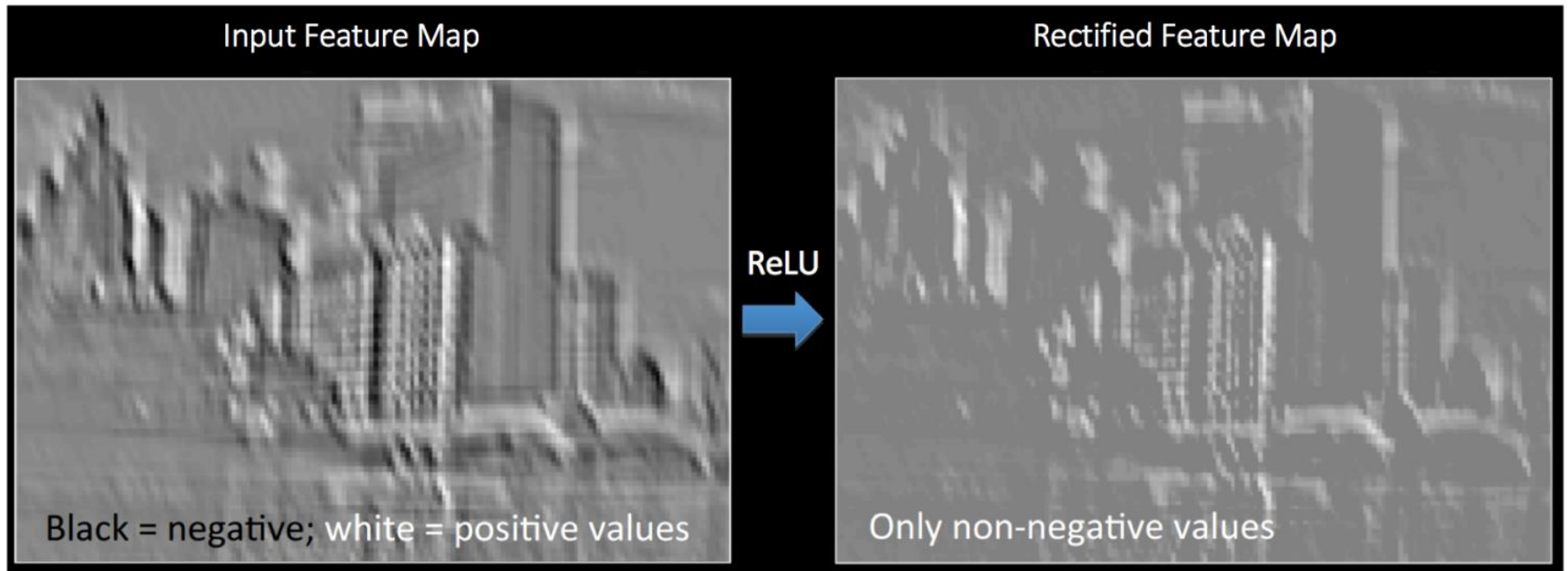
Output =  $\text{Max}(\text{zero}, \text{Input})$



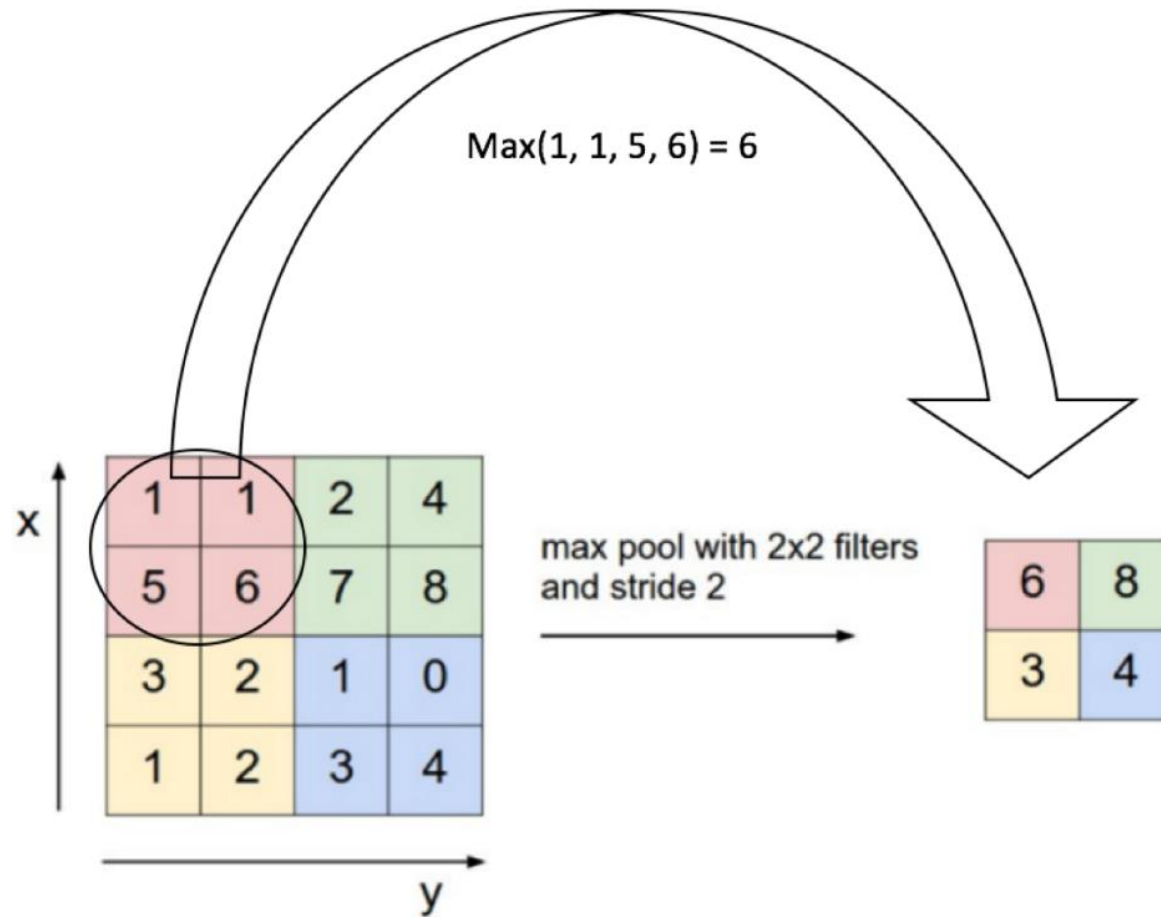


# ReLU

---



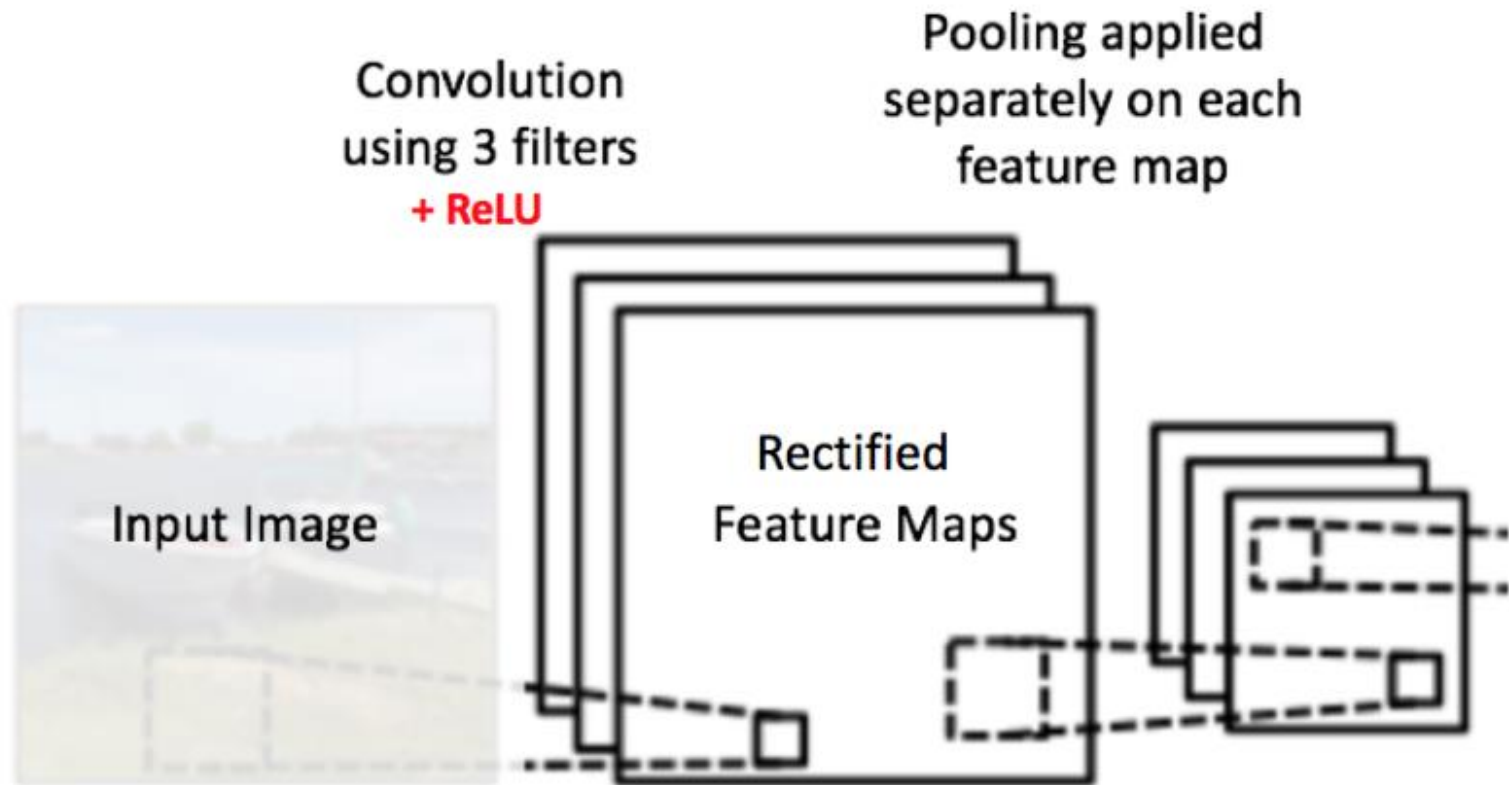
# Pooling



Rectified Feature Map

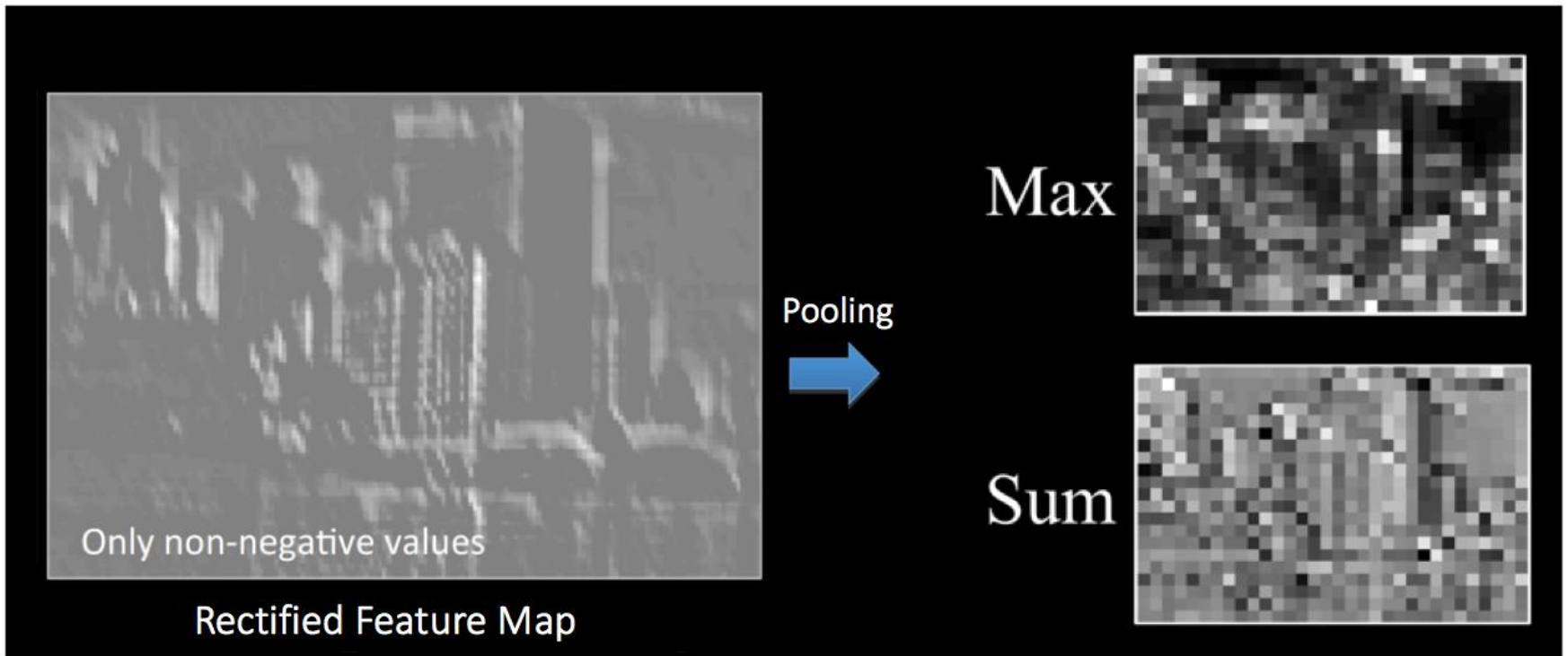
# Pooling

---



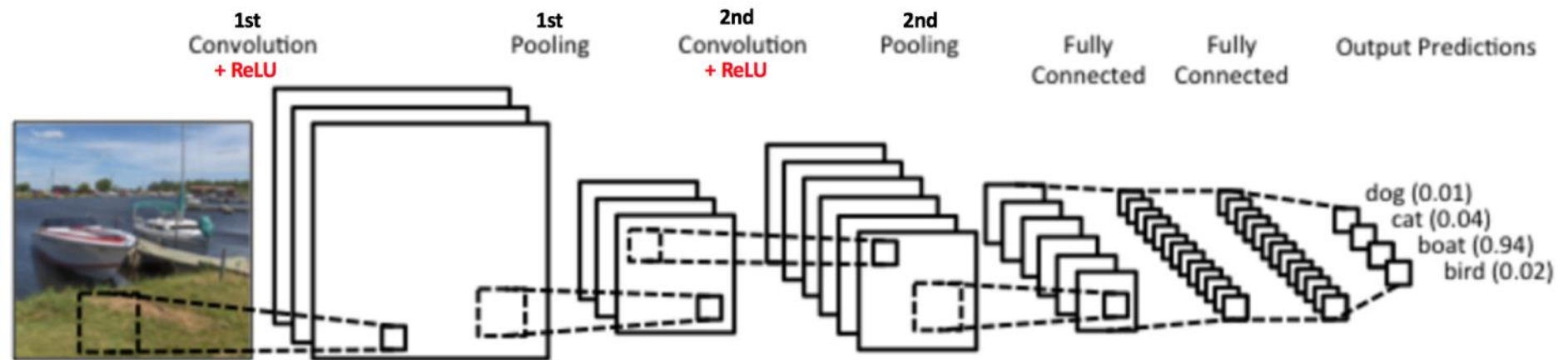
# Pooling

---



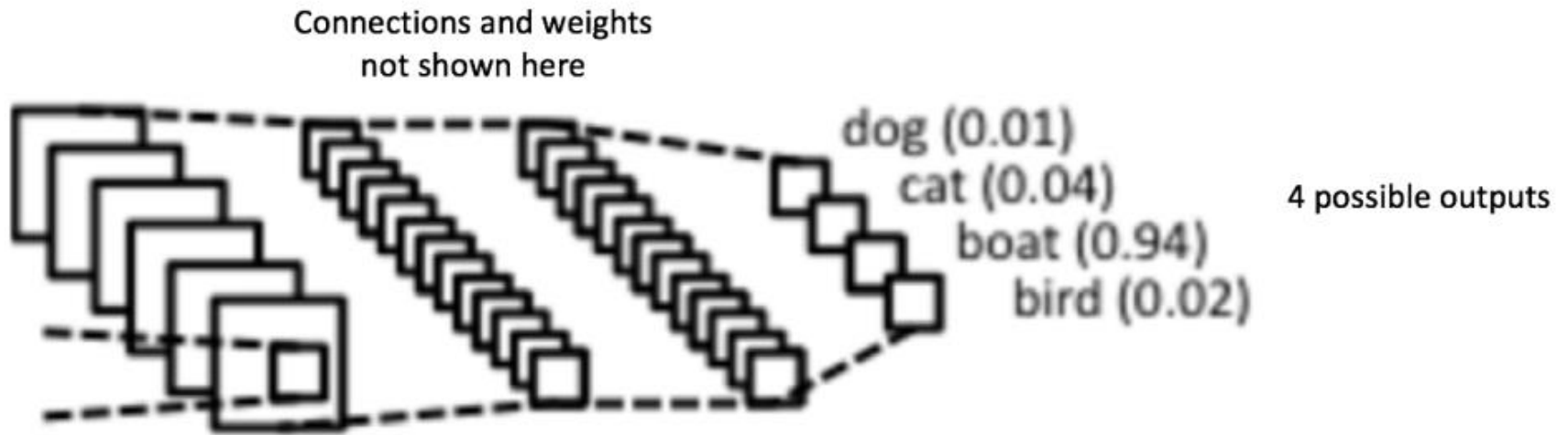
# Pooling

---

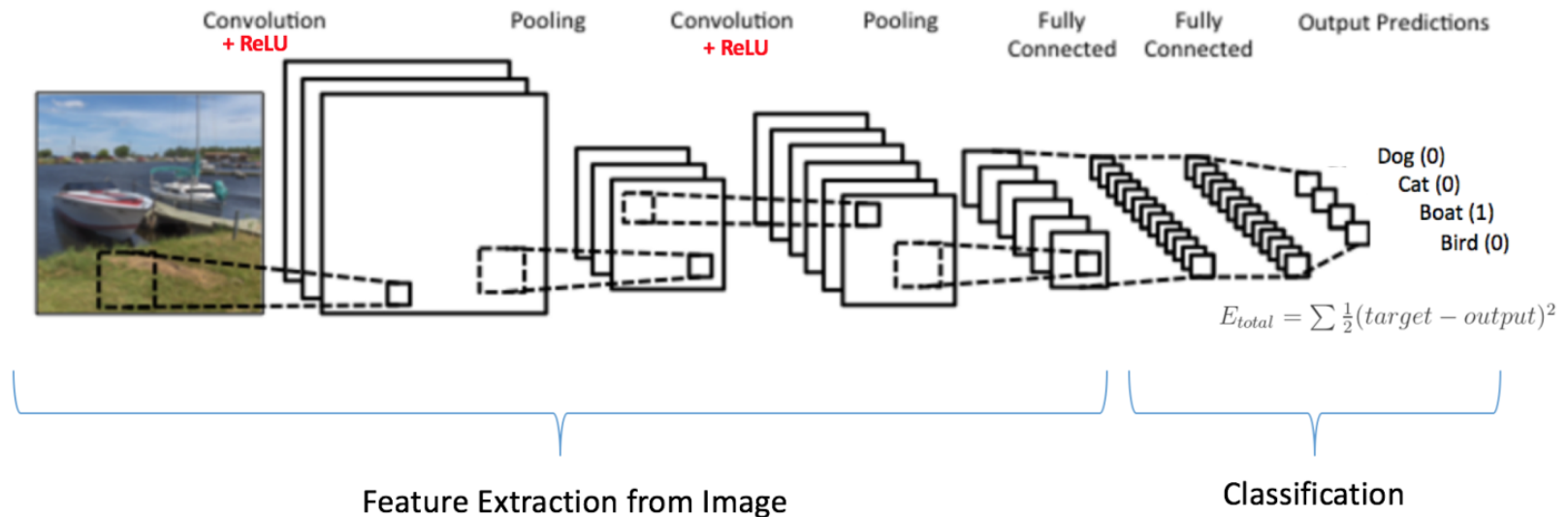


# Full Connection Layer

---



# Backpropagation



- Update the weights of the correlations and the biases

# Gradients

---

$$\mathbf{x}_j^\ell = f\left(\sum_{i \in M_j} \mathbf{x}_i^{\ell-1} * \mathbf{k}_{ij}^\ell + b_j^\ell\right),$$

每一个输出map可能是组合卷积多个输入maps的值。(Mj: 一组maps)

$$\delta_j^\ell = \beta_j^{\ell+1} \left( f'(\mathbf{u}_j^\ell) \circ \text{up}(\delta_j^{\ell+1}) \right)$$

第l层灵敏度δ的计算

$$\frac{\partial E}{\partial b_j} = \sum_{u,v} (\delta_j^\ell)_{uv}.$$

计算bias基的梯度

$$\frac{\partial E}{\partial \mathbf{k}_{ij}^\ell} = \sum_{u,v} (\delta_j^\ell)_{uv} (\mathbf{p}_i^{\ell-1})_{uv}$$

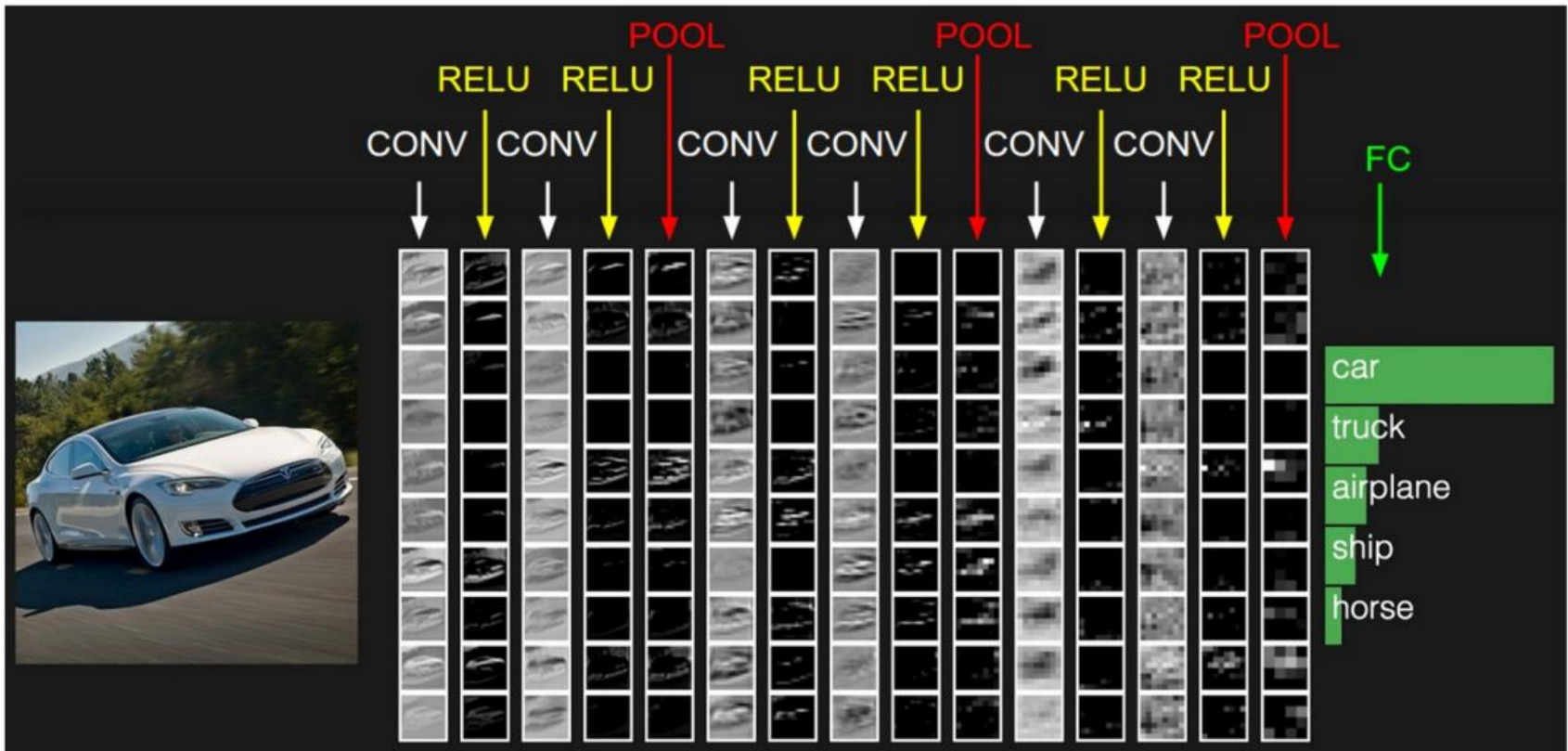
对于一个给定的权值，对所有与该权值有联系（权值共享的连接）的连接、对该点求梯度

$$\frac{\partial E}{\partial \beta_j} = \sum_{u,v} (\delta_j^\ell \circ \mathbf{d}_j^\ell)_{uv}.$$

对β计算梯度

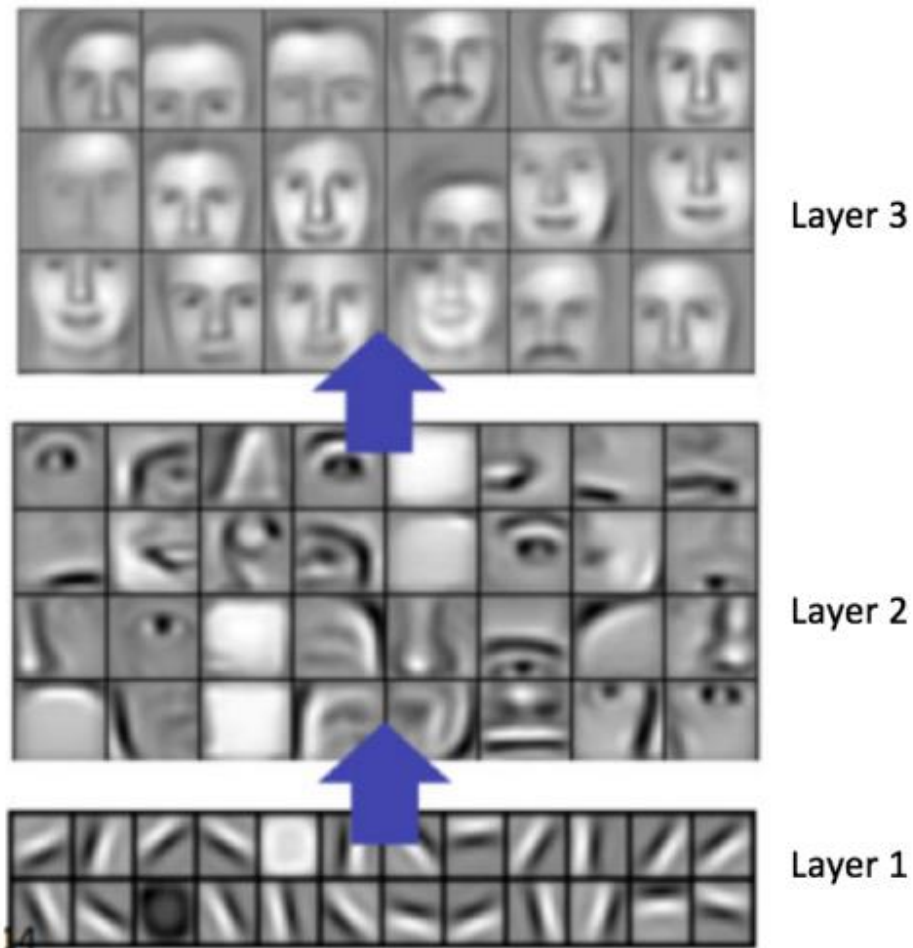


# Training Process

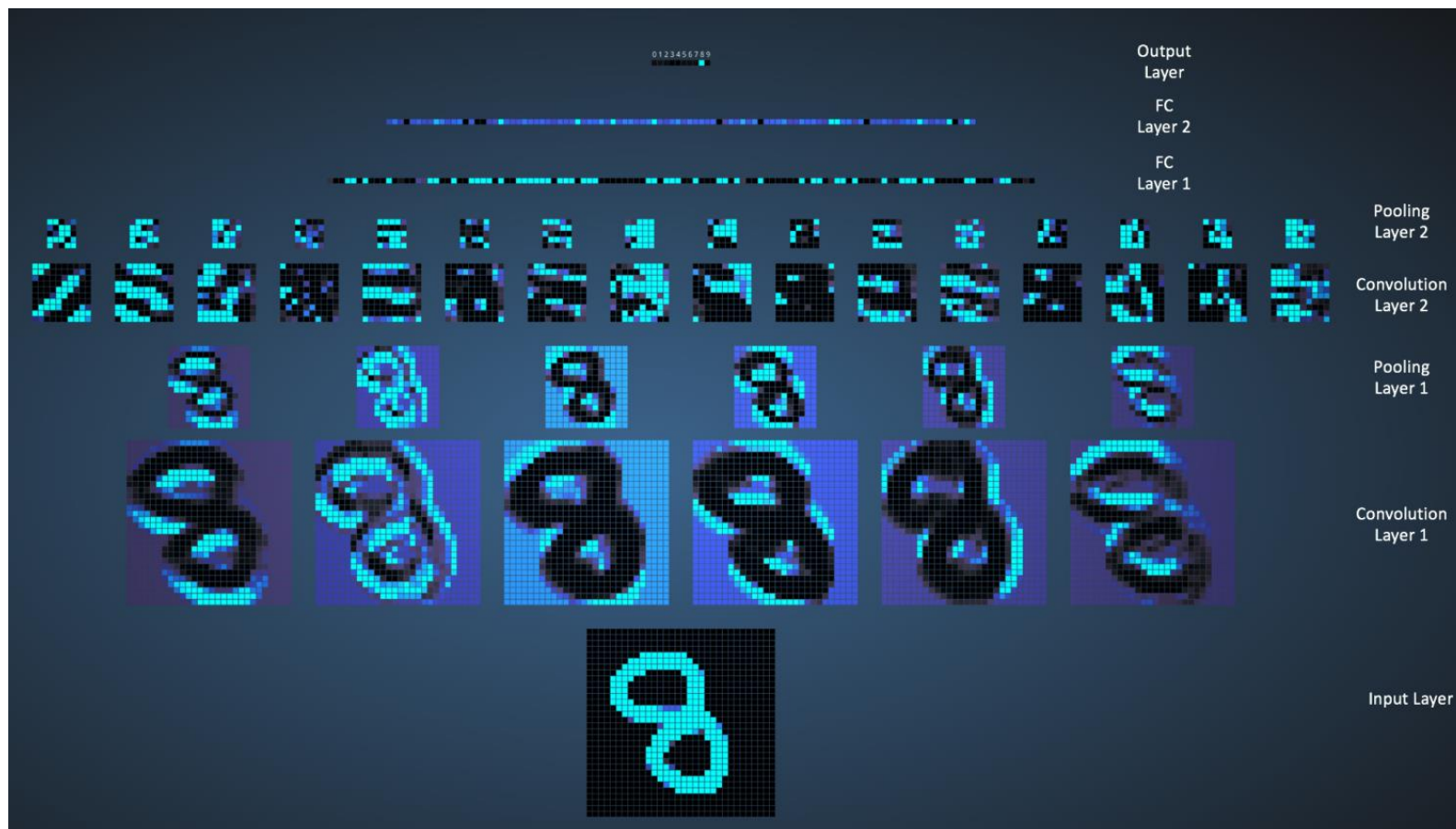


# Feature Visualization

---

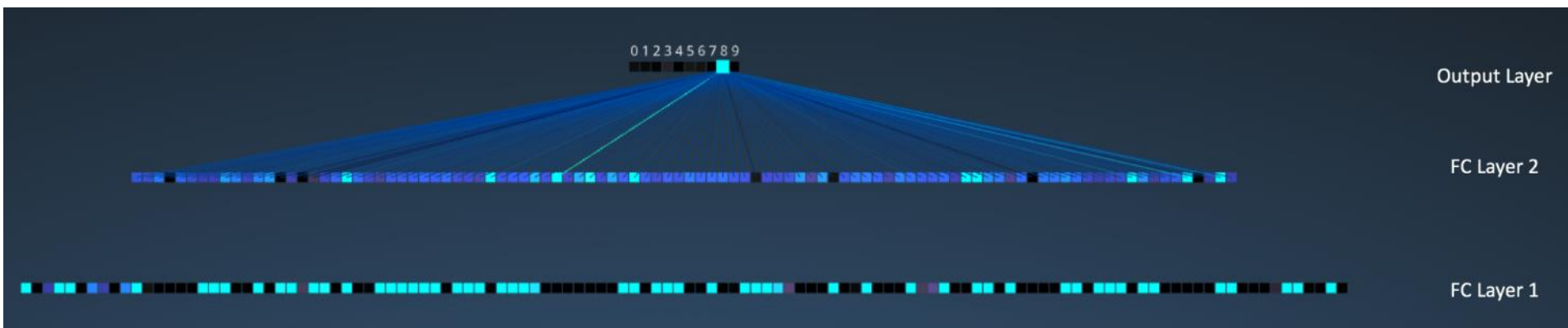


# Feature Visualization



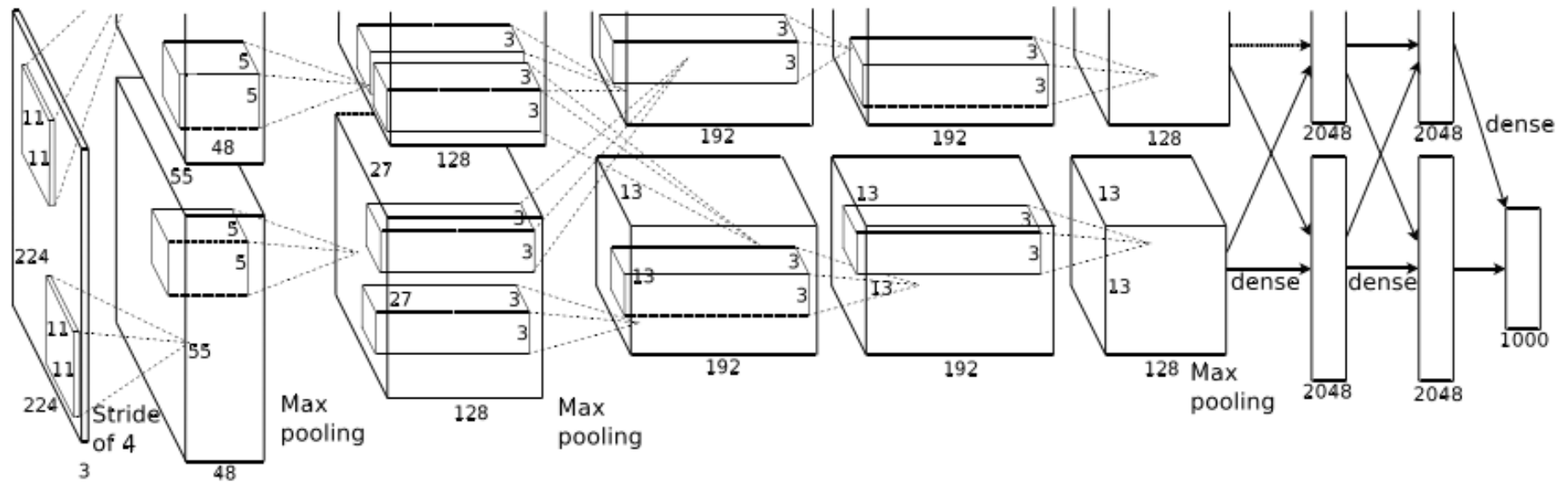
# Feature Visualization

---



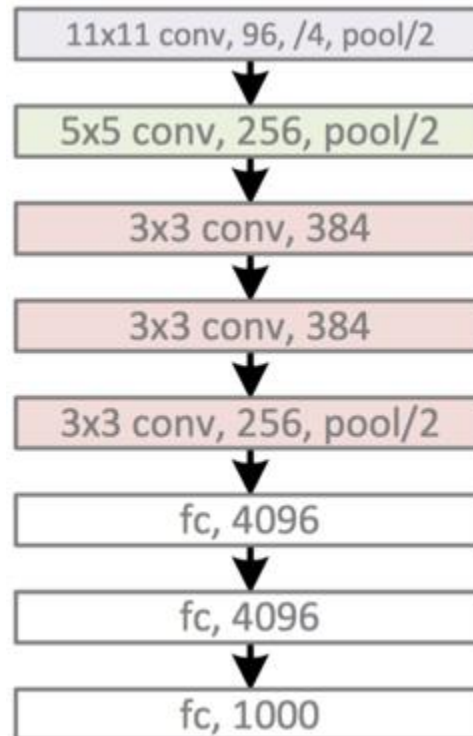
# AlexNet

---



# AlexNet

---



# Dataset

IMAGENET

14,197,122 images, 21841 synsets indexed

[Explore](#) [Download](#) [Challenges](#) [Publications](#) [CoolStuff](#) [About](#)

[WordNet Structure](#)

[Cloud Map](#)

[Most Popular](#)

Not logged in. [Login](#) | [Signup](#)

**ImageNet** is an image database organized according to the **WordNet** hierarchy (currently only the nouns), in which each node of the hierarchy is depicted by hundreds and thousands of images. Currently we have an average of over five hundred images per node. We hope ImageNet will become a useful resource for researchers, educators, students and all of you who share our passion for pictures.

[Click here](#) to learn more about ImageNet, [Click here](#) to join the ImageNet mailing list.

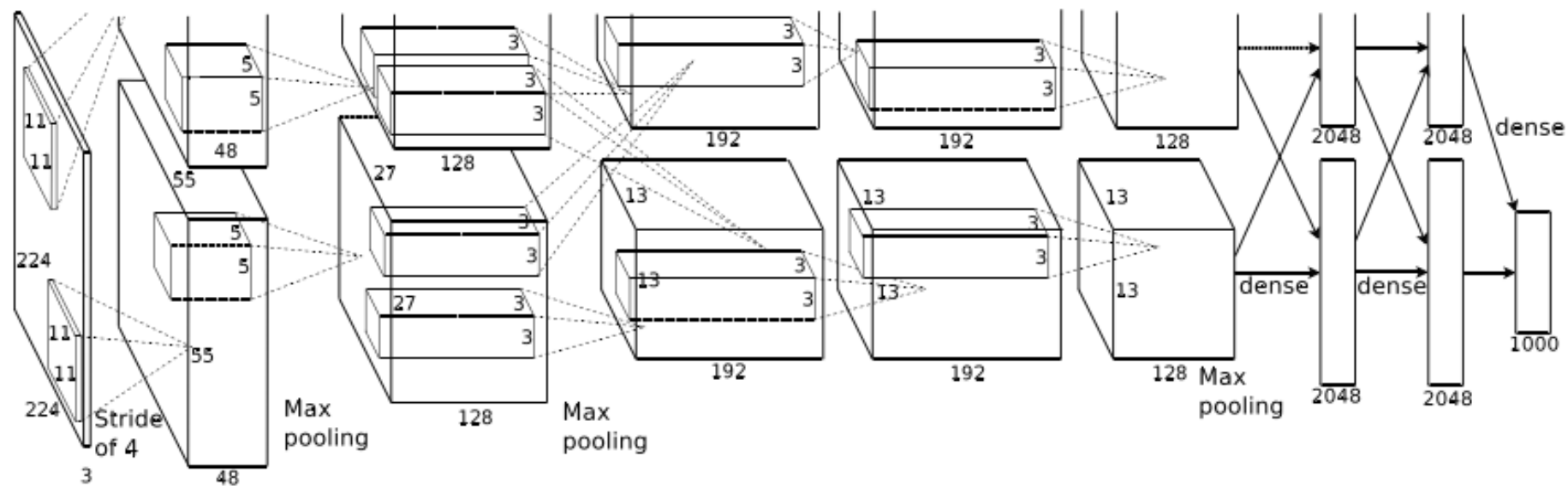


What do these images have in common? *Find out!*

[Check out the ImageNet Challenge 2016](#)

© 2016 Stanford Vision Lab, Stanford University, Princeton University support@image-net.org Copyright infringement

# AlexNet结构



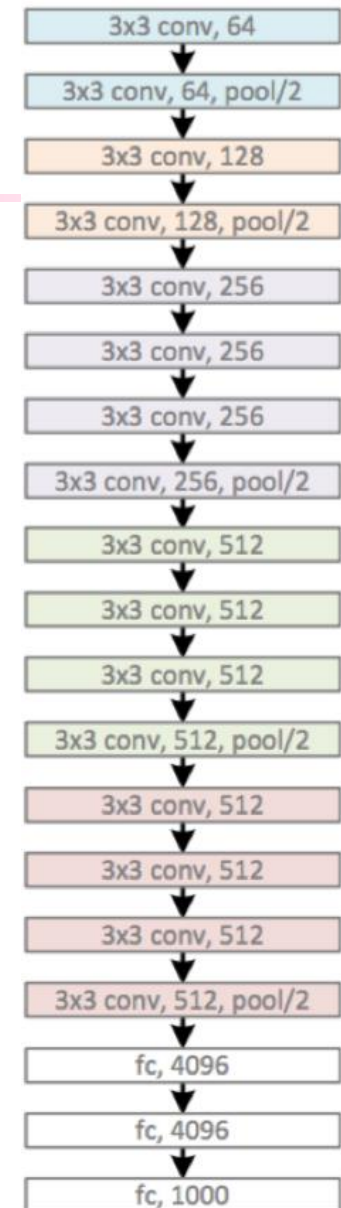


# Evaluation

---

Model	Top-1	Top-5
<i>Sparse coding [2]</i>	47.1%	28.2%
<i>SIFT + FVs [24]</i>	45.7%	25.7%
CNN	<b>37.5%</b>	<b>17.0%</b>

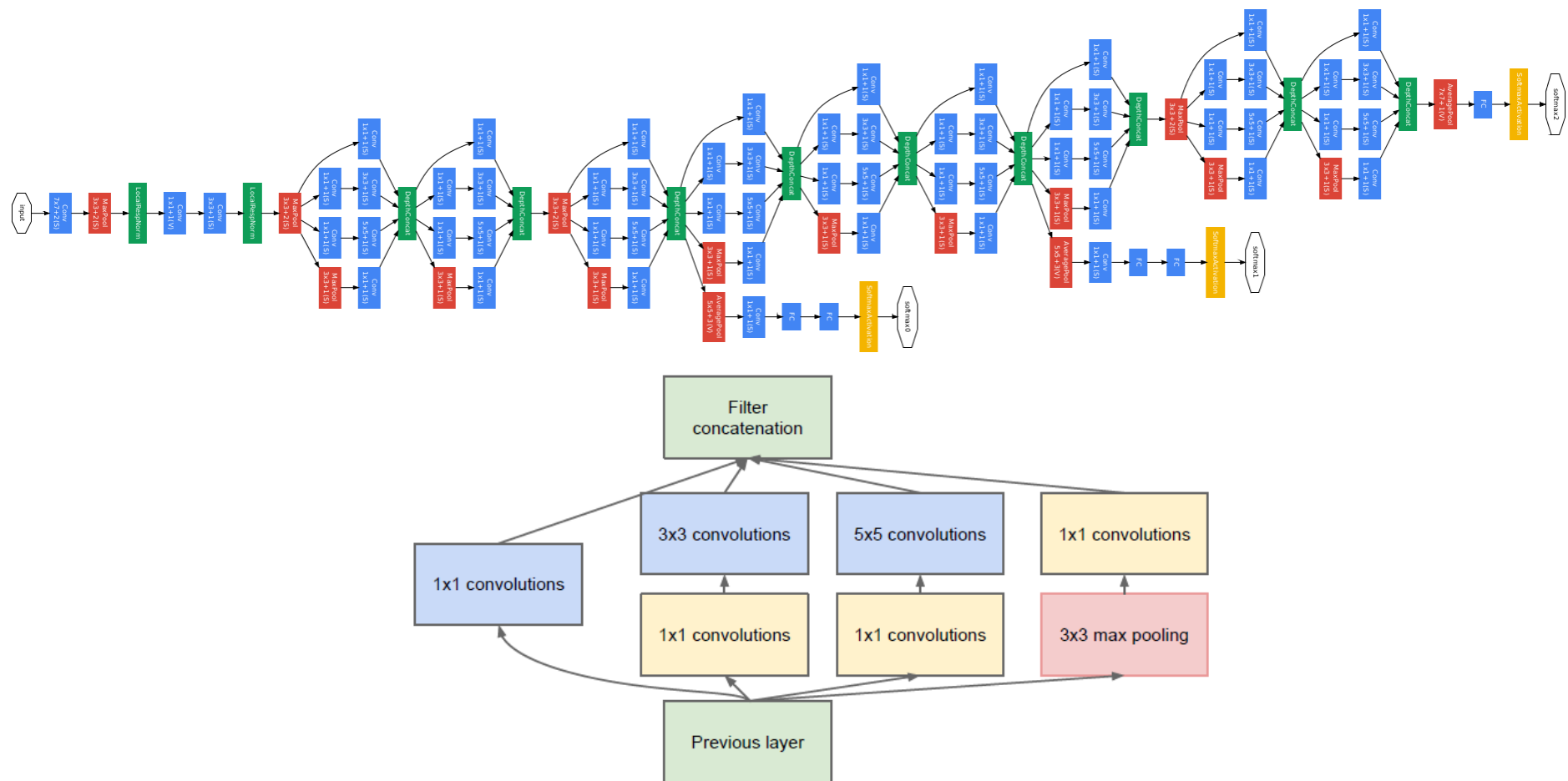
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input ( $224 \times 224$ RGB image)					
conv3-64	conv3-64 <b>LRN</b>	conv3-64 <b>conv3-64</b>	conv3-64	conv3-64	conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 <b>conv3-128</b>	conv3-128	conv3-128	conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 <b>conv1-256</b>	conv3-256 conv3-256 <b>conv3-256</b>	conv3-256 conv3-256 <b>conv3-256</b>
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 <b>conv1-512</b>	conv3-512 conv3-512 <b>conv3-512</b>	conv3-512 conv3-512 <b>conv3-512</b>
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 <b>conv1-512</b>	conv3-512 conv3-512 <b>conv3-512</b>	conv3-512 conv3-512 <b>conv3-512</b>
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					



VGG

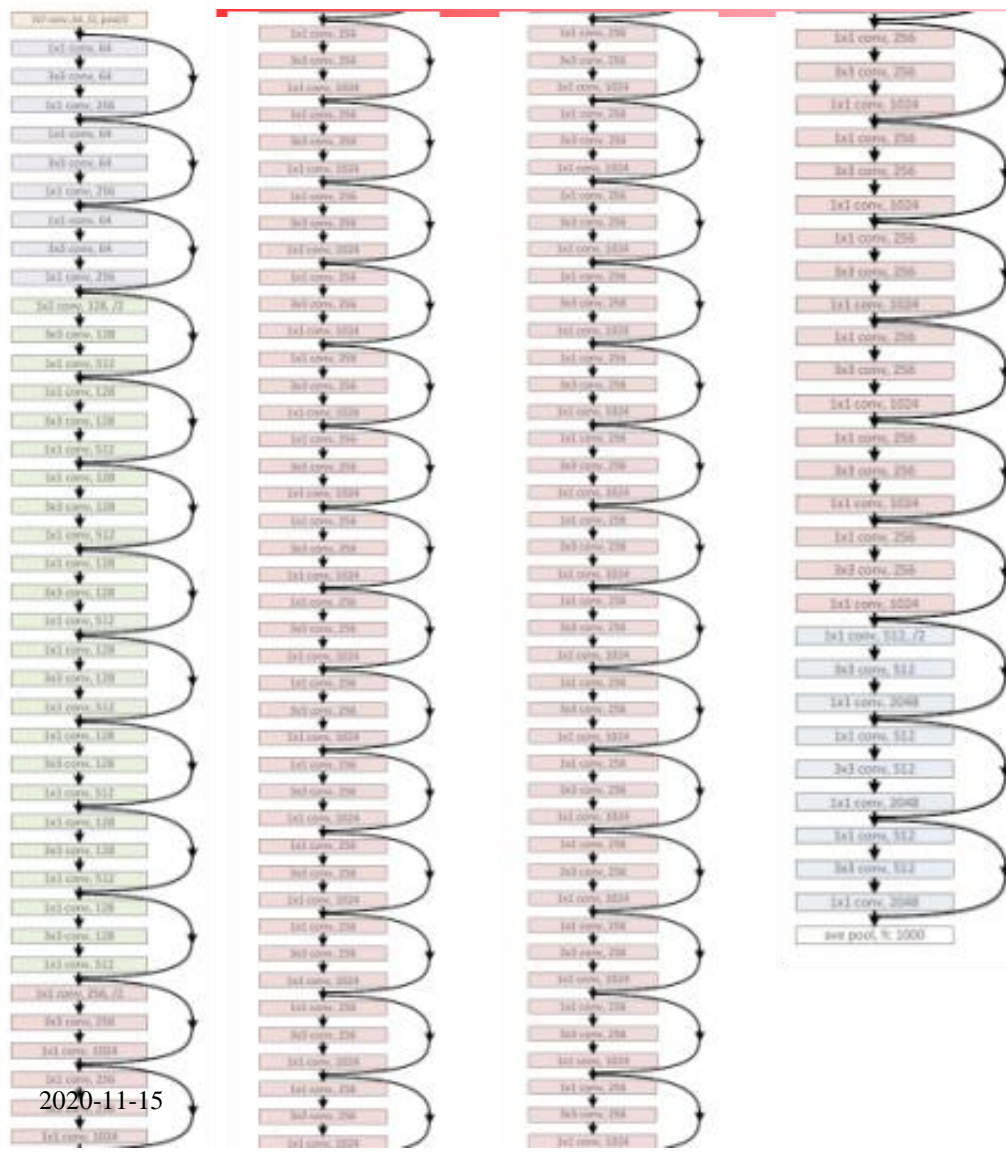
VGG-19

# GoogleNet

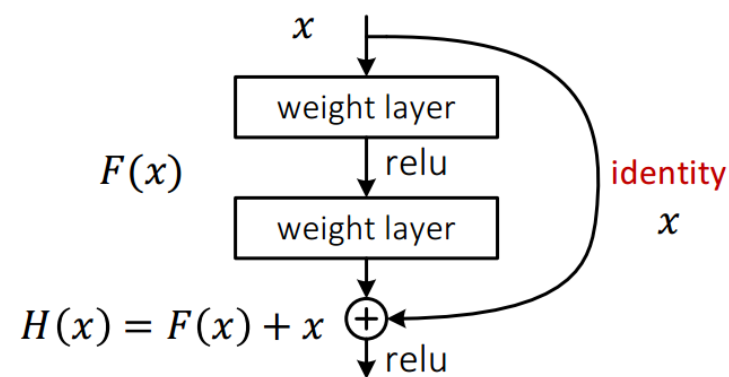


Inception

# ResNet



Easier to converge  
comparing with AlexNet



ResNet

# Image Classification

---

- Trained on millions+ images, up to 1.2 billions parameters
- Datasets
  - Image recognition: 100 millions
- Training time: weeks to months
- Big improvement on image recognition
  - Face: LFW benchmark, 94+% correct





Image uploaded

Baidu



# Advanced Topics

---

- Deep learning
- High performance data mining
- Mining complex data types
- Data mining system products and research prototypes

# Motivation

---

- Large-scale computational intensive applications in science and engineering challenge the computing power
- Aerodynamics: design of new aircraft
- Biology: modeling of genetic compounds
- Physics: cosmology simulation
- Commercial applications



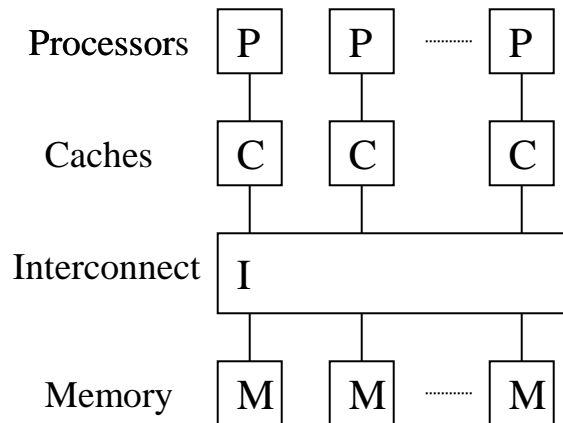
# Motivation

---

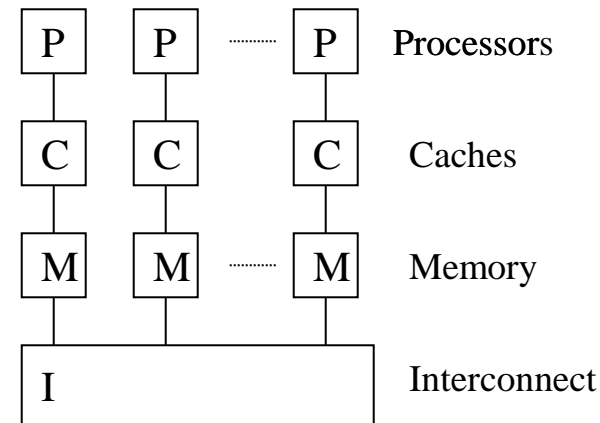
- Computing power limits are being approached
- Memory limitations of sequential computers cause sequential algorithms to make multiple expensive I/O passes over data
- Lacking of capability to solve bigger and more realistic distributed applications
- **Parallel processing is a cost effective solution**

# Shared-Memory Parallel Computers

- All processors share a single global address space
- Single address space facilitates a simple programming model
- Examples: SGI Origin 2000, IBM SP2



(a) UMA



(b) NUMA

# BASIC

---

## ■ Attribute lists

Training Set

Tid	Age	Car Type	Class
0	23	family	High
1	17	sports	High
2	43	sports	High
3	68	family	Low
4	32	truck	Low
5	20	family	High

Attribute lists

Age	Class	Tid	Car Type	Class	Tid
17	High	1	family	High	0
20	High	5	sports	High	1
23	High	0	sports	High	2
32	Low	4	family	Low	3
43	High	2	truck	Low	4
68	Low	3	family	High	5

continuous (sorted)

categorical (orig order)

# BASIC

---

// Starting with the root node, execute the following code for each new tree level

**forall** attributes **in parallel** (dynamic scheduling)

**for** each leaf

        evaluate attribute (E)

**barrier**

**if** (master) **then**

**for** each leaf

        get winning attribute;

        form hash table (W)

**barrier**

**forall** attributes **in parallel** (dynamic scheduling)

**for** each leaf

        split attributes (S)

# BASIC

---

- Attribute data parallelism
  - $d/P$  attributes to each processor
- Dynamic scheduling
- Each processor works independently on its attributes, calculating *gini*
- Once all processors finish *gini*, enter E phase

# BASIC

---

- Build a hash table for each leaf, keep how records partitioned
- Split attribute lists by checking with the hash table
- Breadth-first
  - Once a processor has been assigned an attribute, it can evaluate the splitting points for that attribute for all the leaves at the current tree level

*Problem: when master performs  $W$ , all other processors sleep*

# CCPD (Common Count Partitioned Data)

---

- Parallelize candidate generation
  - Each processor works on a disjoint candidate subset
  - Build the hash tree in parallel, CCPD associates a lock with each leaf node
  - When a processor wants to insert a candidate into the tree, it starts at the root, and successively hashes on the items until it reaches a leaf
  - It then acquires the lock and inserts the candidate
  - With this locking mechanism, each processor can insert itemsets in different parts of the hash tree in parallel

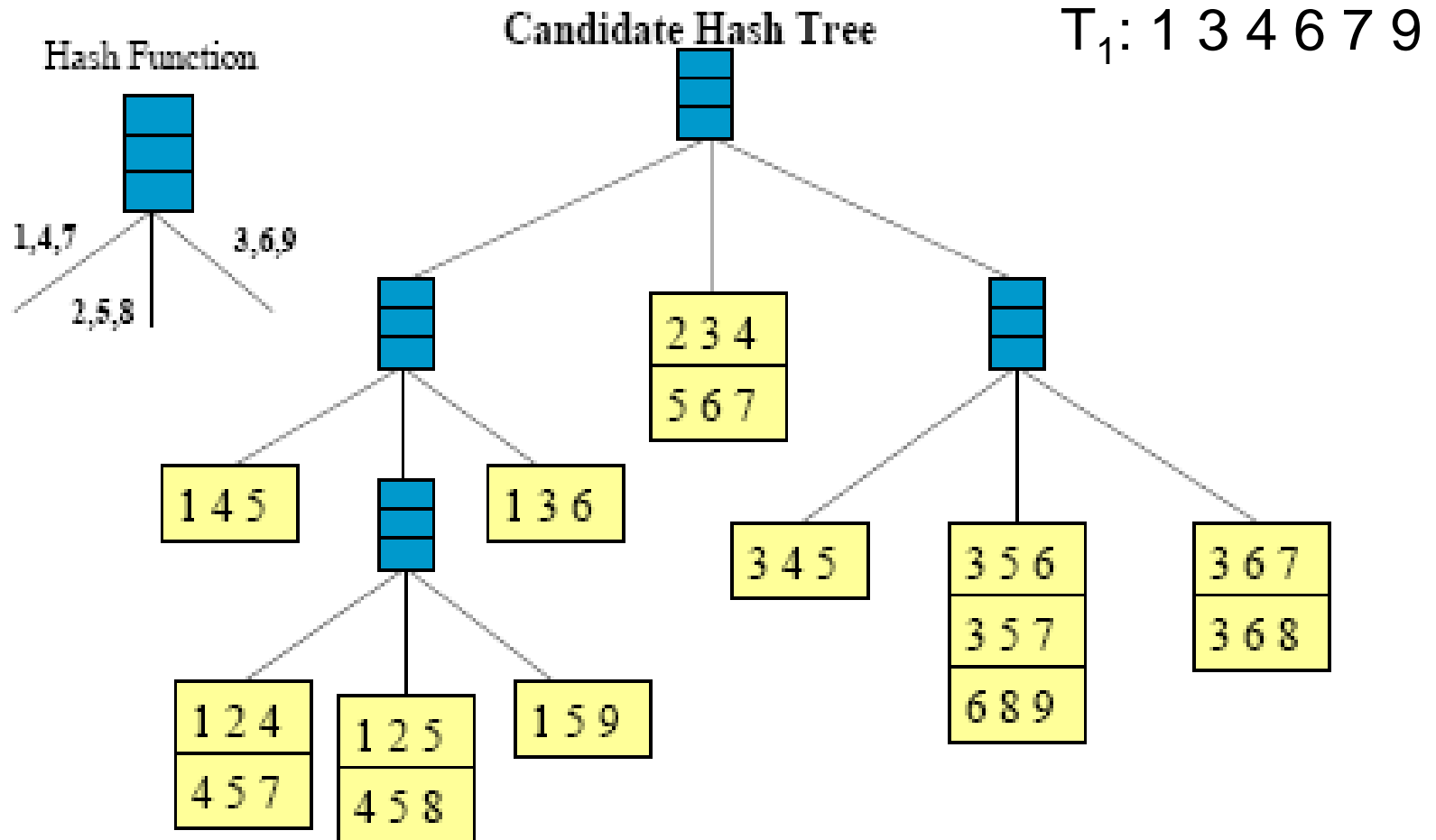
# CCPD (Common Count Partitioned Data)

---

- Parallelize support counting
  - With this locking mechanism, each processor computes frequency from its local partition, update the counts of each candidate in the hash tree

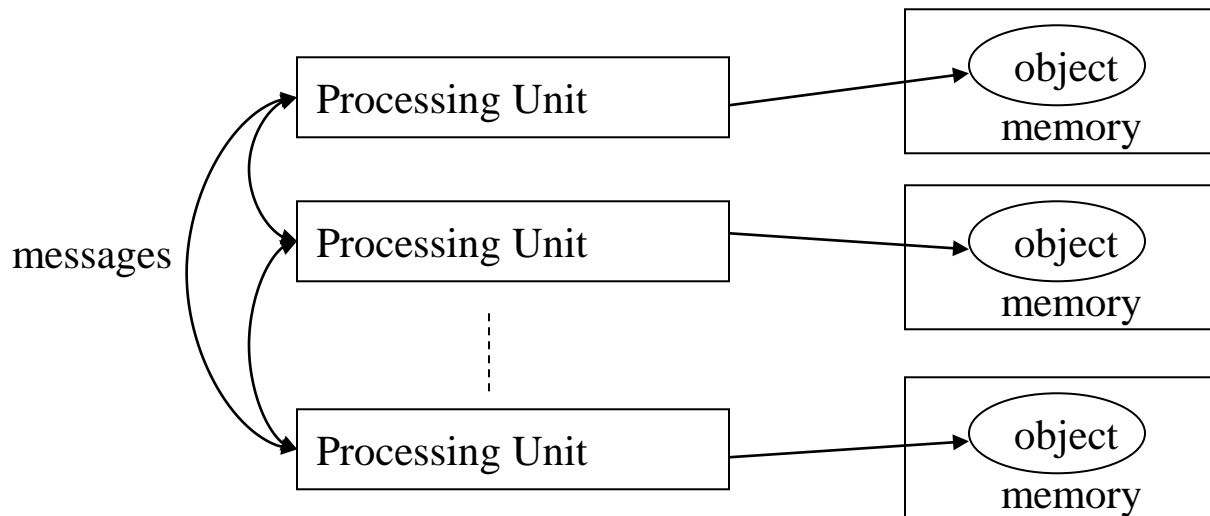


# Example: Counting Supports of Candidates



# Distributed-Memory Parallel Computers (Clusters)

- “Shared nothing:” each processor has a private memory
- Processors can directly access only local data
- Each processing unit can be single processor or a multiprocessor
- Interaction between processors relies on message passing



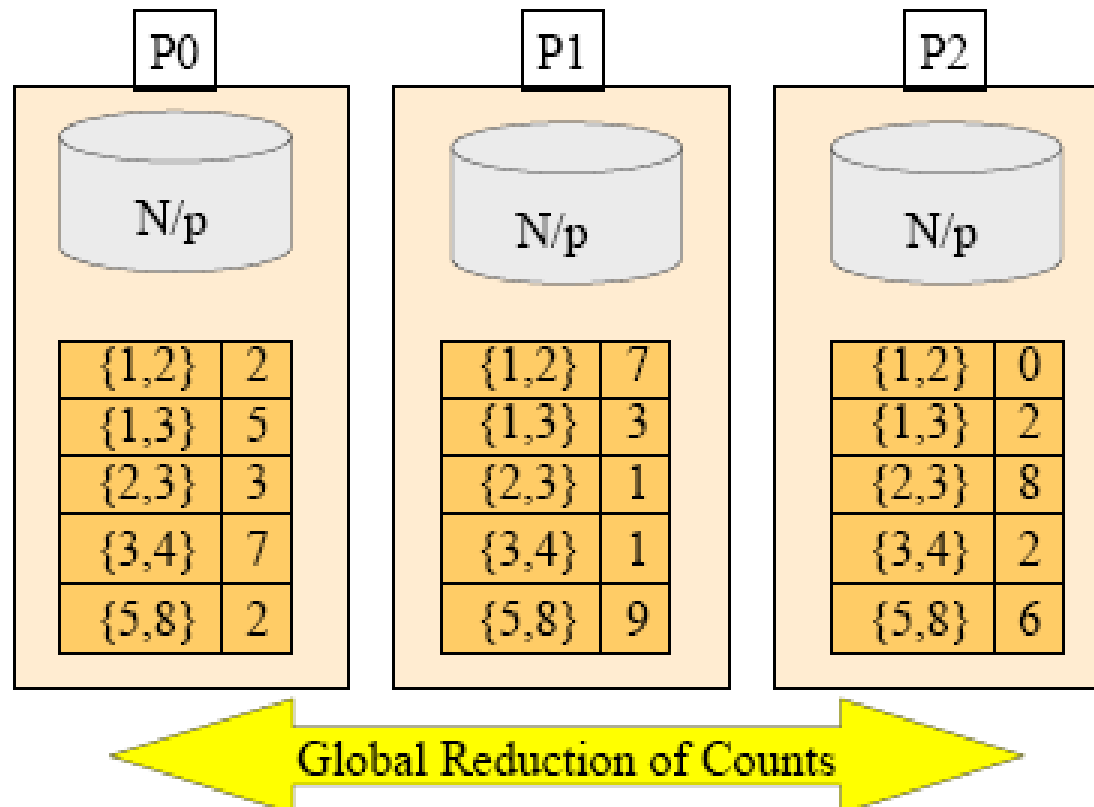
# Count Distribution

---

- Apriori-like
- Iterative approach:
  - Each processor has complete candidate hash tree
  - Each processor updates its hash tree with local data
  - Each processor participates in global reduction to get global counts of candidates in the hash tree
- Multiple database scans per iteration are required if hash tree too big for memory

# Count Distribution

---



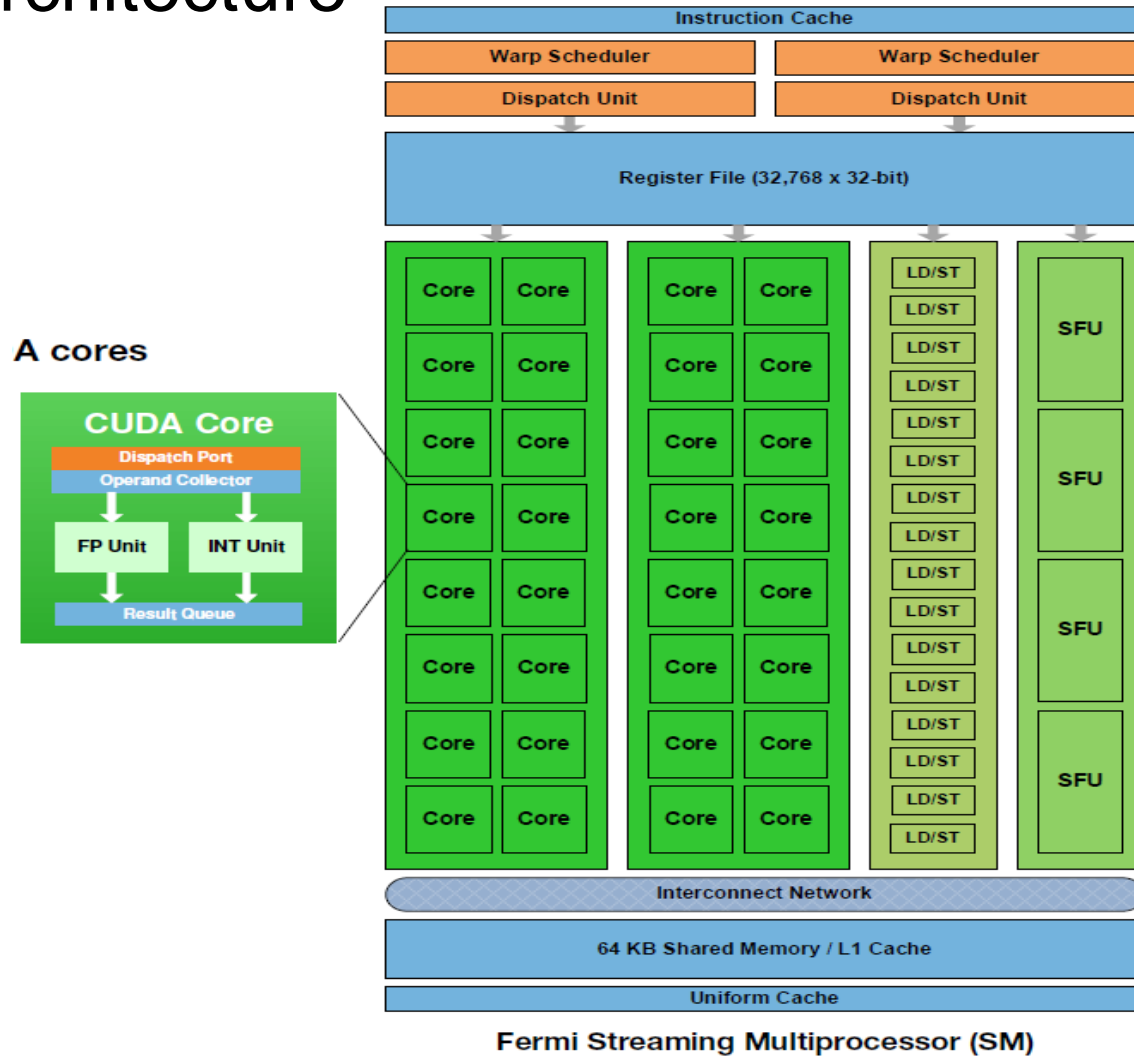
# Parallel *K-Means*

---

- Divide  $N$  points among  $P$  processors
- Replicate the  $k$  centroids
- Each processor computes distance of each local point to the centroids in parallel
- Assign points to closest centroid in parallel
- Perform reduction for global new  $k$  centroids
- Go back to step 2, until no centroid movement

# Parallel Computing on CUDA-enabled GPUs

## ■ GPU architecture



# Parallel Computing on CUDA-enabled GPUs

- CUDA integrated CPU+GPU application C program
  - Serial or modestly parallel C code executes on CPU
  - Highly parallel SPMD kernel C code executes on GPU

CPU Serial Code

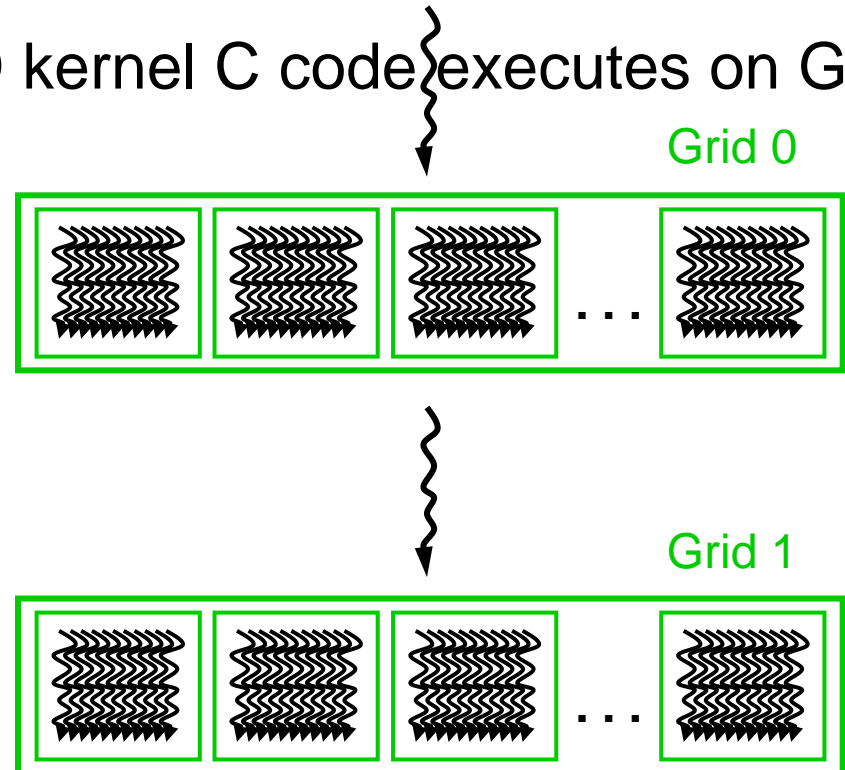
GPU Parallel Kernel

```
KernelA<<< nBlk, nTid >>>(args);
```

CPU Serial Code

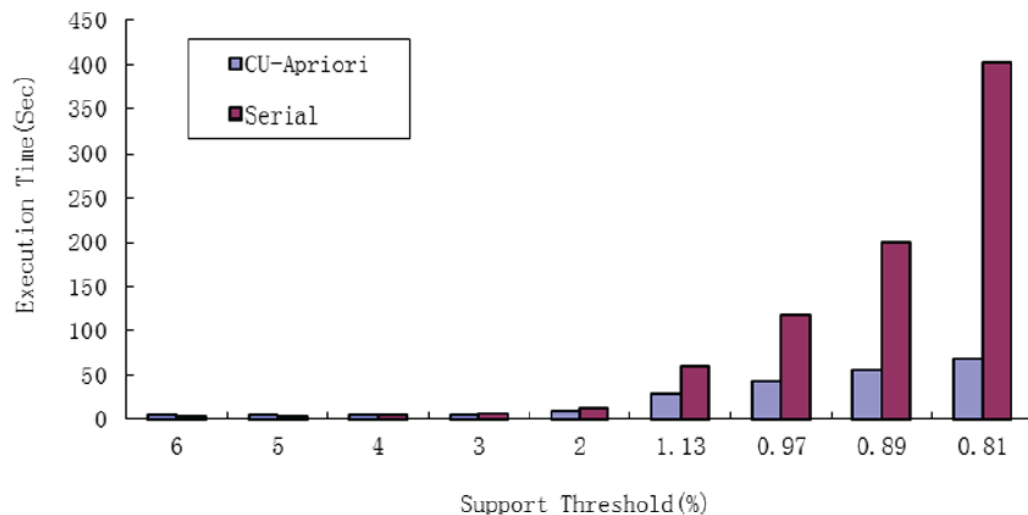
GPU Parallel Kernel

```
KernelB<<< nBlk, nTid >>>(args);
```



# Parallel Computing on CUDA-enabled GPUs

- GUCAS-CUMiner
  - CU-Kmeans (100+x)
  - CU-Apriori (13.5x)
  - CU-KNN (8.31x)
  - CU-Collaborative Filtering (3691x)



- Liheng Jian, Cheng Wang, Shenshen Liang, Ying Liu, Weidong Yi, Yong Shi, "Parallel Data Mining on Graphics Processing Unit with Compute Unified Device Architecture (CUDA)", *Journal of Supercomputing*, Vol. 64(3), 2013, pp. 942-967.
- Zhongya Wang, Ying Liu, Steve Chiu, "An Efficient Parallel Collaborative Filtering Algorithm on Multi-GPU Platform", *Journal of Supercomputing*, DOI:10.1007/s11227-014-1333-4.





# Advanced Topics

---

- Deep learning
- High performance data mining
- Mining complex data types
- Data mining system products and research prototypes

# Mining Complex Data Types

---

- Graph
- Text
- Web pages
- Web log
- Spatial data
- Image
- Audio
- Video
- Sequence pattern
- Time series
- ...

# Text Databases

---

- Text databases (document databases)
  - Large collections of documents from various sources: news articles, research papers, books, digital libraries, e-mail messages, and Web pages, library database, etc.
  - Data stored is usually *semi-structured*
  - Traditional information retrieval techniques become inadequate for the increasingly vast amounts of text data

# Information Retrieval

---

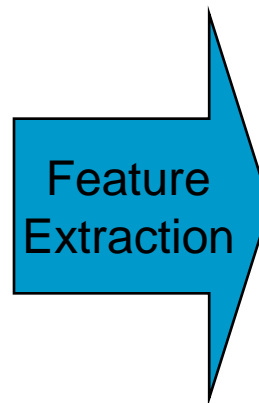
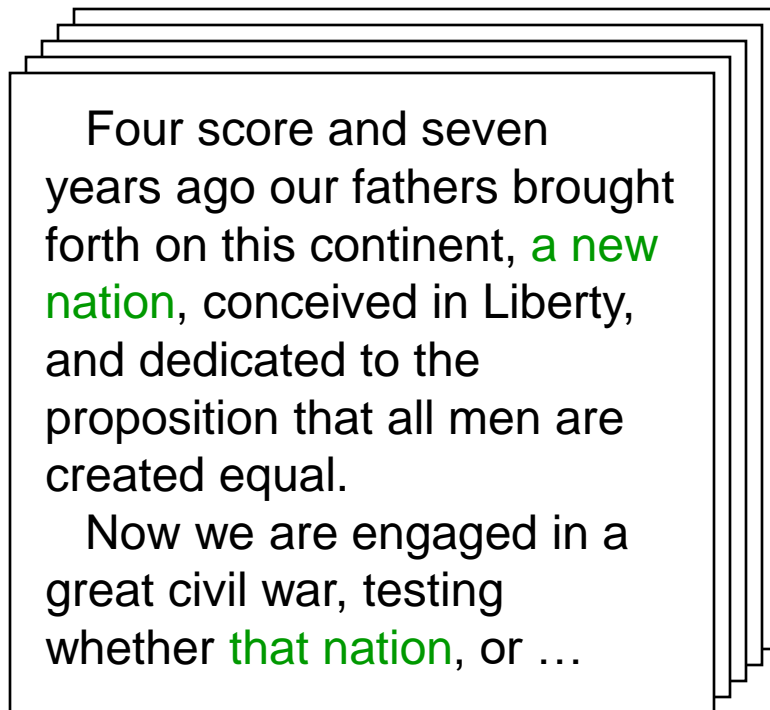
## ■ Information retrieval

- A field developed in parallel with database systems
- Information is organized into (a large number of) documents
- Information retrieval problem: locating relevant documents based on user input, such as keywords or example documents
- Applications
  - On-line library catalog
  - On-line document management system
  - Search engine, e.g. Google, Baidu

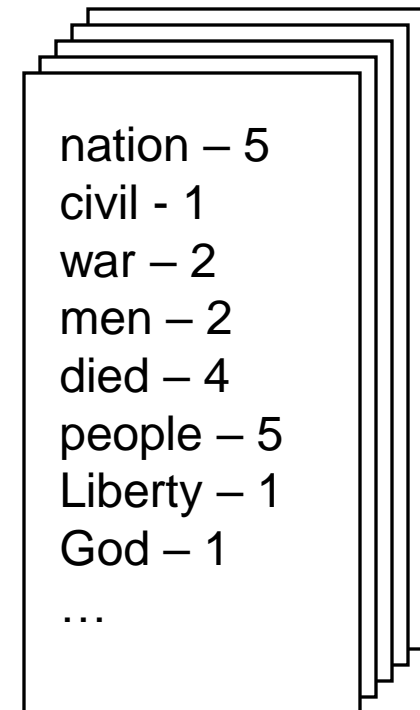
# Tokenization

---

## Documents



## Token Sets



**Lose all order-specific information!**  
**Severely limits context!**

# Information Retrieval Techniques

---

## ■ Tokenization:

### ■ Stop list

- Set of words that are deemed “irrelevant”, even though they may appear frequently
- E.g., *a, the, of, for, to, with*, etc.
- Stop lists may vary when document set varies

### ■ Word stem

- Several words are small syntactic variants of each other since they share a common word stem
- E.g., *drug, drugs, drugged*

# Information Retrieval Techniques

---

## ■ Index terms weighting

- Term frequency  $freq(d,t)$

$$e.g. TF(d,t) = \begin{cases} 0 & \text{if } freq(d,t) = 0 \\ 1 + \log(1 + \log(freq(d,t))) & \text{otherwise} \end{cases}$$

- Relative term frequency: term frequency / total # of occurrences of all the terms in  $d$
- Inverse document frequency (IDF)

$$IDF(t) = \log \frac{1 + |d|}{|d_t|}$$

- TF-IDF measure

$$TF-IDF(d,t) = TF(d,t) \times IDF(t)$$

# Information Retrieval Techniques

---

## ■ Example

Document/term	$t_1$	$t_2$	$t_3$	$t_4$	$t_5$	$t_6$	$t_7$
$d_1$	0	4	10	8	0	5	0
$d_2$	5	19	7	16	0	0	32
$d_3$	15	0	0	4	9	0	17
$d_4$	22	3	12	0	5	15	0
$d_5$	0	7	0	9	2	4	12

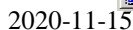
$$TF(d_4, t_6) = 1 + \log(1 + \log(15)) = 1.3377$$

$$IDF(t_6) = \log((1 + 5)/3) = 0.301$$

$$TF-IDF(d_4, t_6) = 1.3377 * 0.301 = 0.403$$



- The degree of similarity of the document  $d$  with regard to the query  $q$  is calculated as the correlation between the vectors that represent them, using measures such as the Euclidian distance or the cosine of the angle between these two vectors



# Information Retrieval Techniques

---

## ■ Vector space model

- Represent document and query in a high-dimensional space vector of  $t$  keywords
- Compute the similarity measure between a query vector and a document vector

## ■ Document vector

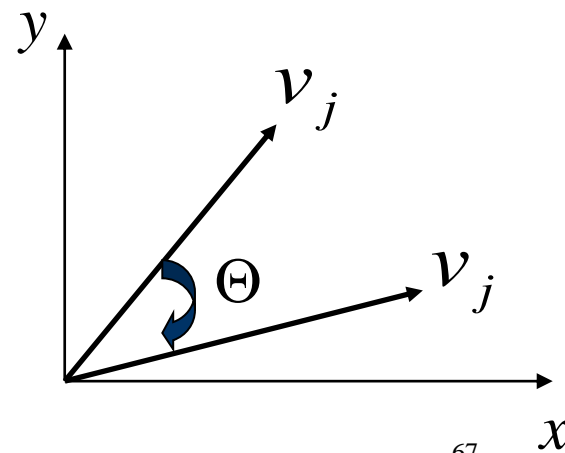
- A document can be described by a set of representative keywords called **index terms**
- Different index terms have varying relevance when used to describe document contents.
- This effect is captured through the **assignment of numerical weights to each index term** of a document

# Similarity-Based Retrieval in Text Data

- Find similar documents based on a set of common keywords
- Answer should be based on the degree of relevance based on the nearness of the keywords, relative frequency of the keywords, etc.
- Similarity metrics: measure the closeness of a document to a query (a set of keywords)
  - Euclidian distance
  - Cosine distance

$$\text{sim}(v_1, v_2) = \frac{v_1 \cdot v_2}{|v_1| |v_2|}$$

$$v_1 \cdot v_2 = \sum_{i=1}^t v_{1i} v_{2i} \quad |v_1| = \sqrt{v_1 \cdot v_1}$$



# Indexing Techniques

---

## ■ Inverted index

- Maintains two hash- or B+-tree indexed tables:
  - **document\_table**: a set of document records <doc\_id, postings\_list>
  - **term\_table**: a set of term records, <term, postings\_list>
- Answer query: Find all docs associated with one or a set of terms
- Pros
  - easy to implement
- Cons
  - do not handle well synonymy and polysemy
  - posting lists could be too long (storage could be very large)

# Problems with Text Data

---

- Large size
- Synonymy / polysemy
- High dimensionality
- Noisy data
- Complex and poorly defined structure and semantics

# Dimensionality Reduction

---

## ■ Challenges

- The number of keywords (terms) is huge, the number of documents in a database is huge, thus, the size of the term frequency matrix is huge
- Term frequency matrix is sparse
- Inefficient computation of similarities

## ■ Feature extraction, reduce the number of dimensions

## ■ Feature extraction methods

- Latent semantic indexing (隐性语义索引)
- Locality preserving indexing (局保索引)
- Probabilistic latent semantic indexing

# Example

---

Original document-term matrix

	d1	d2	d3	d4	d5	d6
cosmonaut	1	0	1	0	0	0
astronaut	0	1	0	0	0	0
moon	1	1	0	0	0	0
car	1	0	0	1	1	0
truck	0	0	0	1	0	1

Rescaled document matrix,  
reduced into two dimensions

	d1	d2	d3	d4	d5	d6
Dim1	-1.62	-0.60	-0.04	-0.97	-0.71	-0.26
Dim2	-0.46	-0.84	-0.30	1.00	0.35	0.65

# Types of Text Data Mining

---

- Keyword-based association analysis
- Automatic document classification
- Similarity detection
  - Cluster documents by a common author
  - Cluster documents containing information from a common source
- Sequence analysis: predicting a recurring event
- Anomaly detection: find information that violates usual patterns



# Keyword-Based Association Analysis

---

## ■ Motivation

- Collect sets of keywords or terms that occur frequently together and then find the **association** or **correlation** relationships among them

## ■ Association Analysis Process

- Preprocess the text data by parsing, stemming, removing stop words, etc.
- Database is in the format
$$\{document\_id, key1, key2, \dots\}$$
  - Consider each document as a transaction
  - View a set of keywords in the document as a set of items in the transaction
- Evoke association mining algorithms

# Keyword-Based Association Analysis

---

- The frequently occurring keywords may form a phrase
- Term-level association mining can help detect
  - Compound associations, e.g. domain-dependent phrases, [Stanford, University], [U.S. President, George W. Bush]
  - Noncompound associations, e.g. [dollars, share, exchanges, commission]

# Document Clustering

---

## ■ Motivation

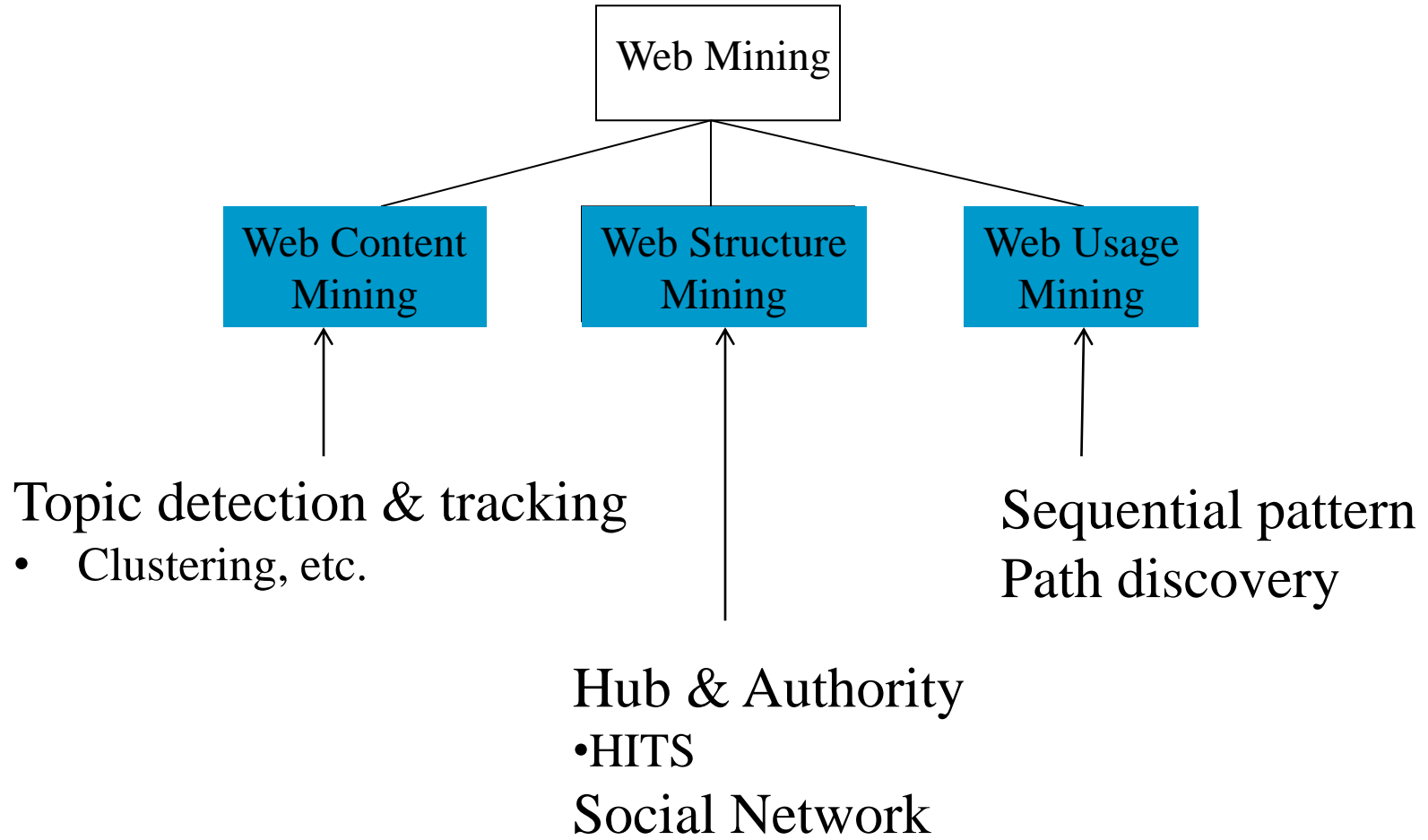
- Automatically group related documents based on their contents
- No predetermined training sets or taxonomies
- Generate a taxonomy at runtime

## ■ Clustering Process

- Data preprocessing: remove stop words, stem, feature extraction, lexical analysis, etc.
- K-means, Latent Dirichlet Allocation (LDA)

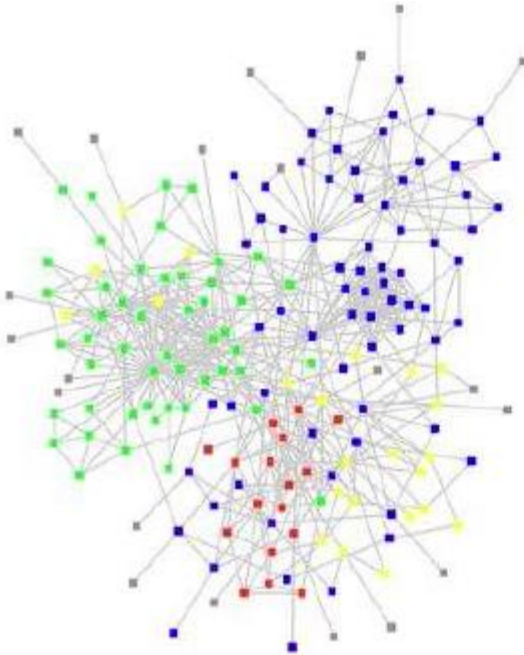
# Web Mining

---



# Graph Mining

---



- Generally, a graph  $G = \langle V, E \rangle$  can be described as a matrix
  - The columns and rows are indexed by  $V$
  - The elements are the strengths on the corresponding edges in  $E$

# Graph Mining

---

- AprioriGraph
  - Frequent subgraphs
- CloseGraph
  - Maximal frequent subgraphs
- Graph classification
  - COM (Co-Occurrence rule Miner)
- ...

# Advanced Topics

---

- Deep learning
- High performance data mining
- Mining complex data types
- Data mining system products and research prototypes

# Examples of Data Mining Systems

---

## ■ IBM Intelligent Miner

- A wide range of data mining algorithms
- Scalable mining algorithms
- Toolkits: neural network algorithms, statistical methods, data preparation, and data visualization tools
- Tight integration with IBM's DB2 relational database system

## ■ SAS Enterprise Miner

- A variety of statistical analysis tools
- Data warehouse tools and multiple data mining algorithms

## ■ Microsoft SQLServer 2000

- Integrate DB and OLAP with mining
- Support OLEDB for DM standard



# Examples of Data Mining Systems

---

## ■ SGI MineSet

- Multiple data mining algorithms and advanced statistics
- Advanced visualization tools

## ■ Clementine (SPSS)

- An integrated data mining development environment for end-users and developers
- Multiple data mining algorithms and visualization tools

## ■ Matlab

# Examples of Data Mining Systems

---

## ■ R

- A free software environment for statistical computing and graphics
- Compiles and runs on a wide variety of UNIX platforms, Windows and MacOS

## ■ Weka

- A free collection of machine learning algorithms
- Written in Java and runs on almost any platform
- Can either be applied directly to a dataset or called from your own Java code