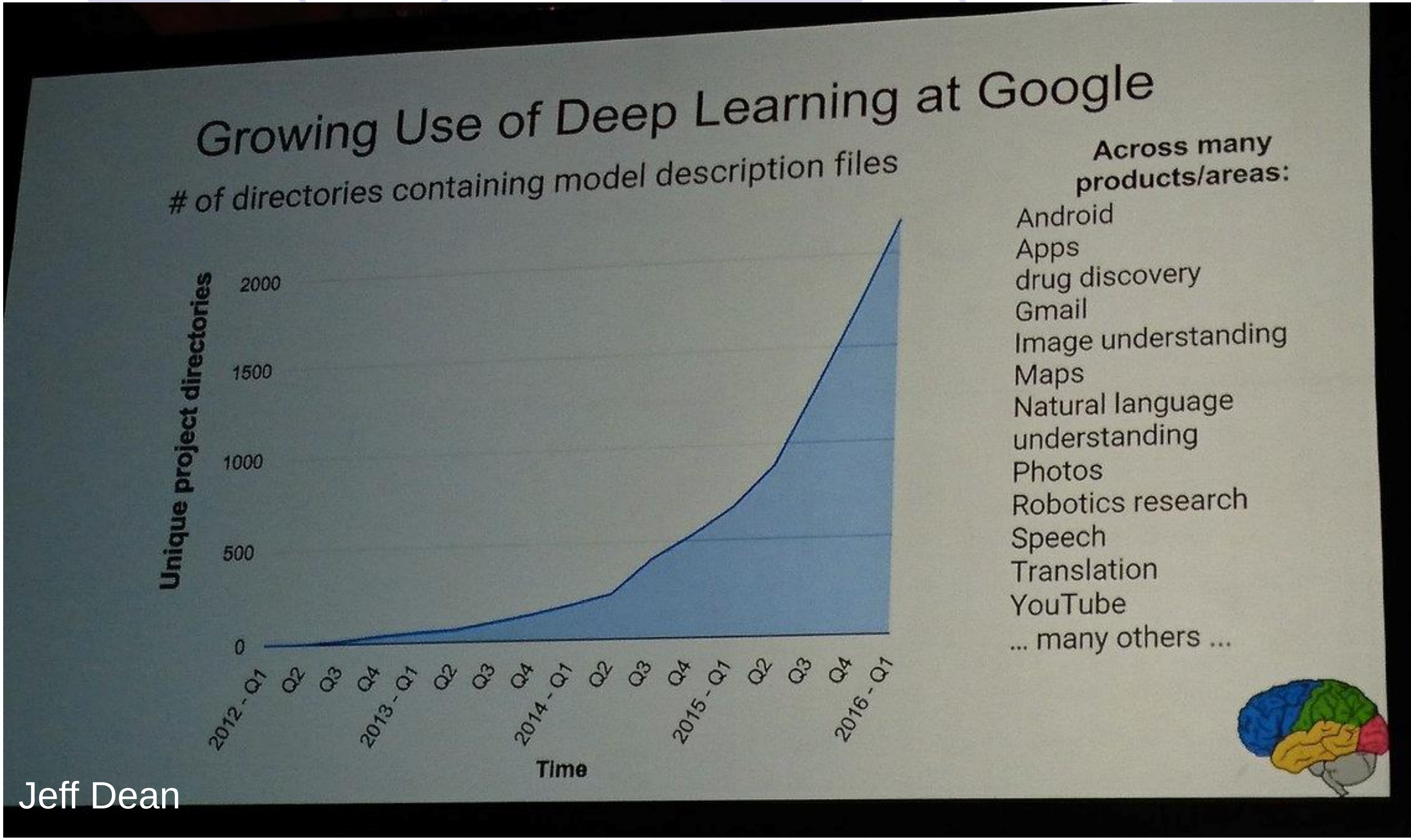
Deep Learning and its Applications

Material adapted from: Prof. Ming Li
University of Waterloo
Model based design lecture from V. Ramesh (Goethe University, Frankfurt)

Source Acknowledgement:
G. Hinton, Y. Lecunn, HY Lee, S. Mallat, I. Goodfellow, and C. Olah lectures, notes and blogs

Lecture 1: Introduction



Many predictions, by 2025 – 2030, 1-2 billion people will lose their jobs to AI

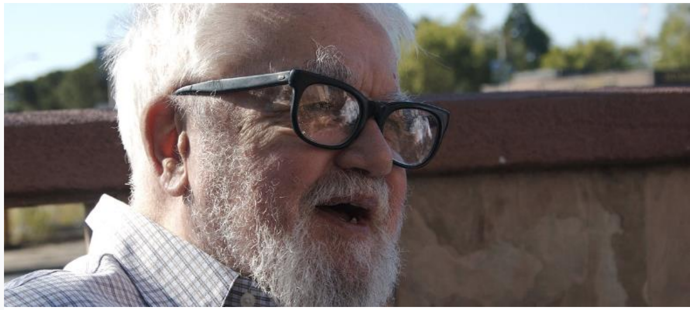
1956: The beginning of AI: The Dartmouth Conference

On May 26, 1956, McCarthy notified Robert Morison of the planned 11 attendees:

For the full period:

```
1) Dr. Marvin Minsky
2) Dr. Julian Bigelow
3) Professor D.M. Mackay
4) Mr. Ray Solomonoff
5) Mr. John Holland
6) Mr. John McCarthy.
```

Von Neumann hired him to build first computer



For four weeks:

```
7) Dr. Claude Shannon
8) Mr. Nathanial Rochester
9) Mr. Oliver Selfridge.
```

Designed IBM 701

"supervisor of Minsky"

For the first two weeks:

```
10) Mr. Allen Newell
11) Professor Herbert Simon.
```

Turing Award, student

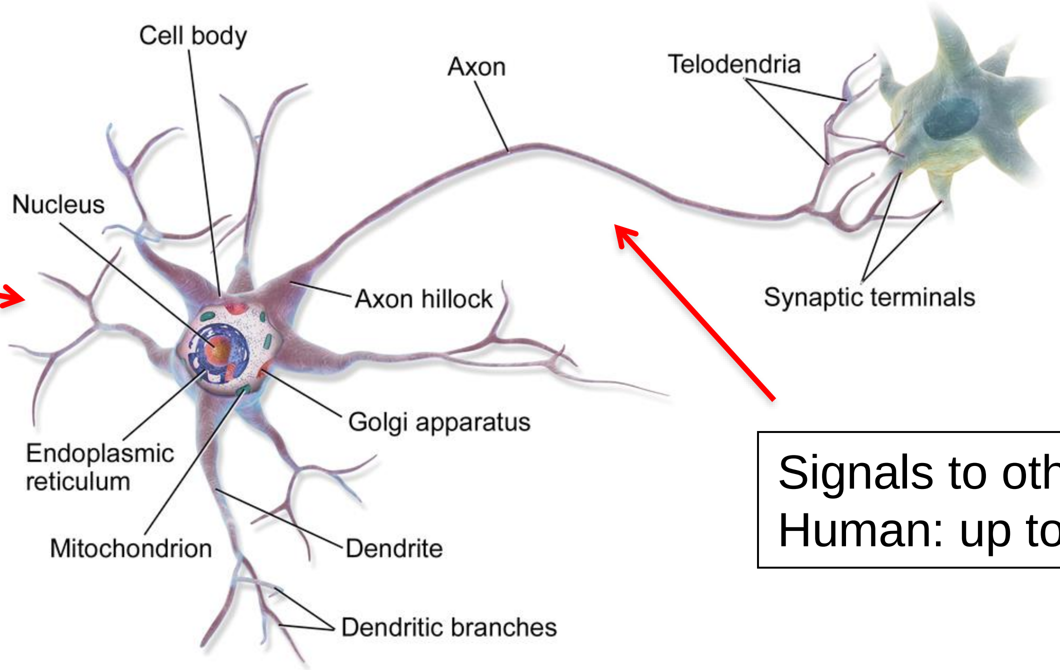
Nobel Prize, supervisor

The theme was using computers to mimic human intelligence.

Neurons in nature

- Human has ~100 billion neurons/nerve cells (& many more supporting cells)
- Each neuron has 3 parts: cell body, dendrites, axon connected up to ~10,000 other neurons. Passing signals to each other via 1000 trillion synaptic connections, approximately 1 trillion bit per second processor.
- Human memory capacity 1~1000 terabytes.

Signals from other neurons
There are synaptic weights &they adapt



Signals to other neurons.
Human: up to 1 meter long

What is our natural system good at?

- Vision
- Hearing (very adaptive)
- Speech recognition / speaking
- Driving
- Playing games
- Natural language understanding

- “Not good at”: multiply 2 numbers, memorize a phone number.

Why not other types of learning?

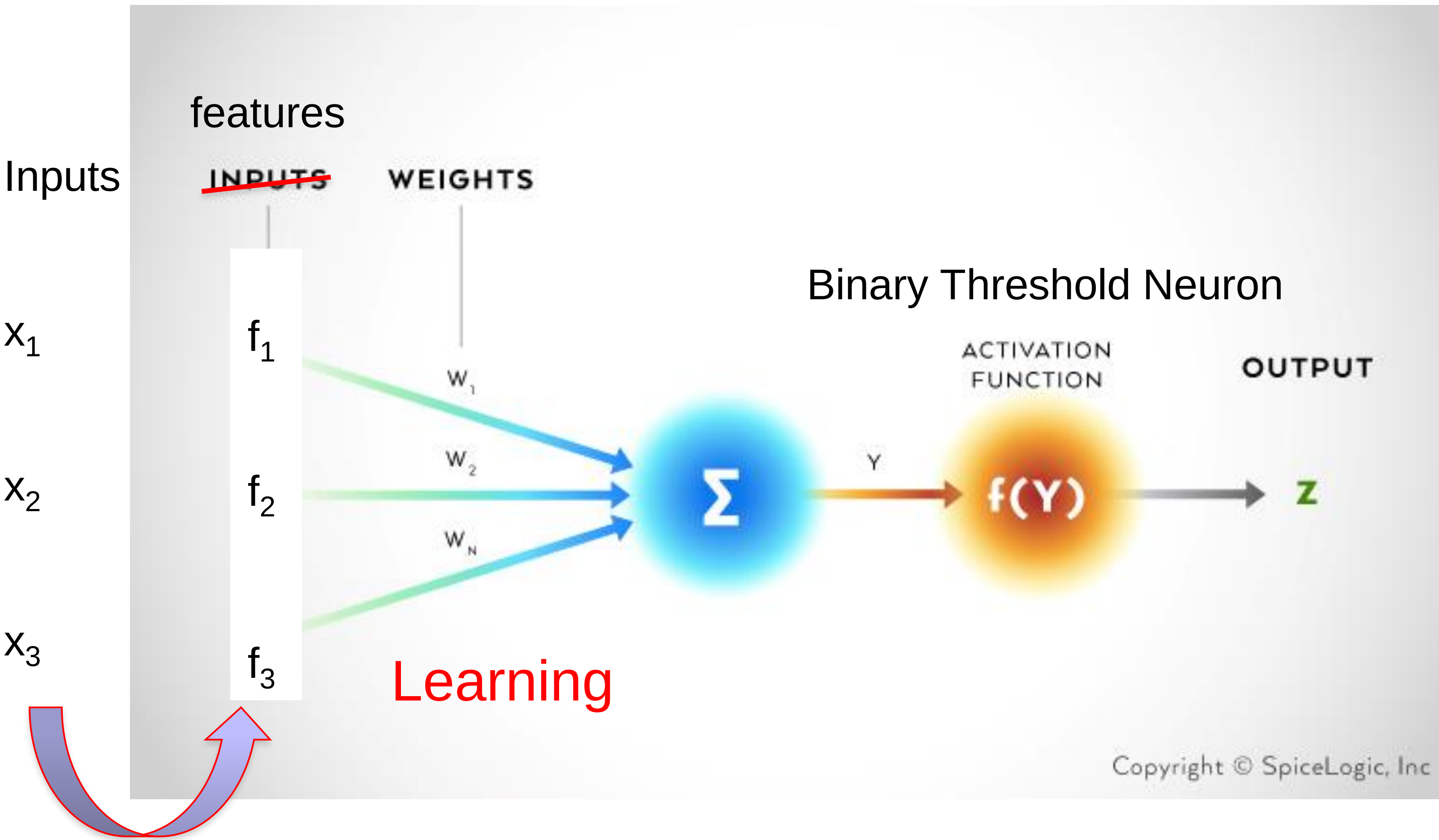
- Linear regression?
  - Why is it linear?
- Bayesian?
  - What is the prior?
- SVM?
  - What are the features?
- Decision tree?
  - What are the nodes/variables?
- PAC learning?
  - What is the function to be learnt?
- KNN?
  - Cluster on what features?

These methods do not suit well with very complex models.

Ups and Downs of AI

- In the 1956 Dartmouth meeting, it has already mentioned neuron networks
- How did learning go deep. Easy hype target as AI borders science and science fiction.
  - Perceptron popularized by F. Rosenblatt, 1957 (Principles of Neurodynamics 1961).
    - *Times*: .. A revolution ..
    - *New Yorker* …
    - A science magazine title "Human brains replaced?"
    - False claims: "After 5 years all of us will have smart robots in our homes …"
    - It turns out that Rosenblatt's experiments of distinguishing tanks from trucks were because of lightings.
    - 1969, Minsky and Papert proved Perceptron, being a linear separator, is not very powerful. For example, can't do exclusive-or. But this was misconstrued as NNs being too week.
    - 1980s, multi-layer perceptron
    - 1986 Backpropagation, hard to train > 3 layers.
    - 1989: 1 hidden layer can do all, why deep?
    - 2006 RBM initialization (breakthrough) re-kindled fire.
    - < 2009: Game industry has pushed the growth of GPU's
    - 2011: Speech recognition (Waterloo professor Li Deng invited Hinton to Microsoft)
    - 2012: won ILSVRC image competition (with ImageNet training data)
  - 1980's expert system
  - Japan's 5th generation computers (thinking machines)

Perceptron Architecture

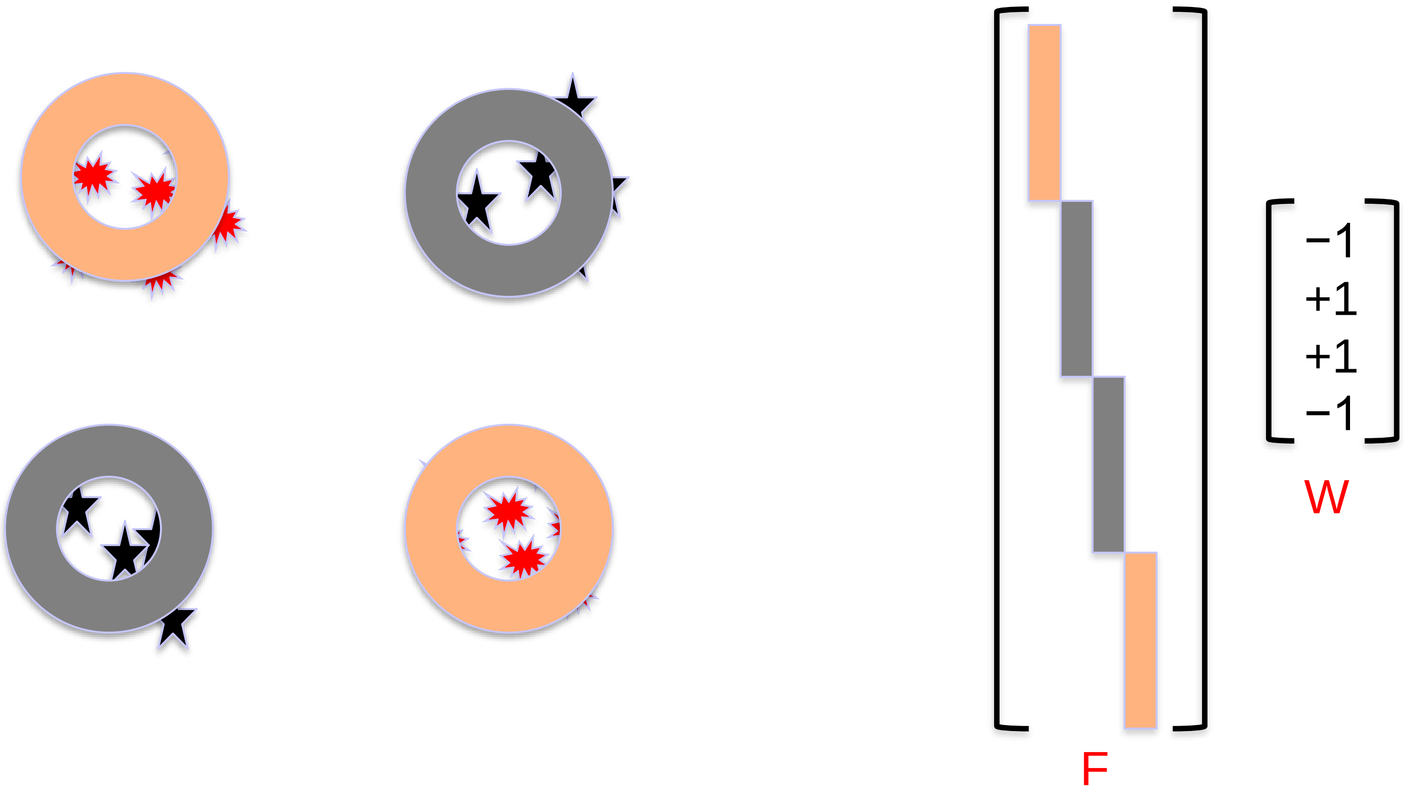


By hand!

As long as you pick right features, this can learn almost anything.

This actually gives a powerful machine learning paradigm:

- Pick right features by clustering
- Linearly separate the features.
- This is essentially what Rosenblatt initially claimed for perceptron. Chomsky & Papert actually attacked a different target.



Binary threshold neuron

- McCulloch-Pitts (1943)

There are two ways of describing the binary threshold neuron:
1.Threshold = 0
2.Threshold ≠ 0

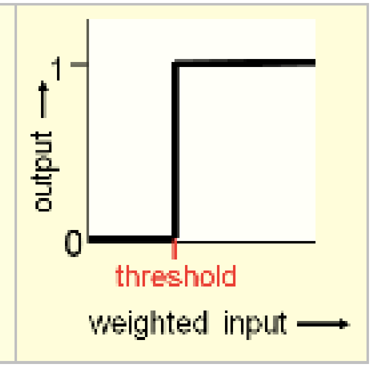
$$z = \sum_i x_i w_i$$

$$y = \begin{cases} 1 \text{ if } z \ge \theta \\ 0 \text{ otherwise} \end{cases}$$

$$\boxed{\theta = -b}$$

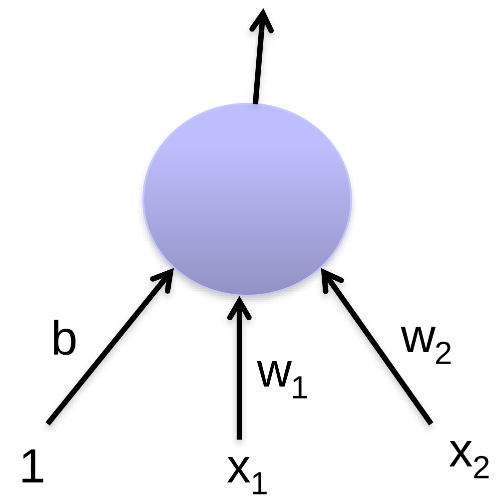
$$z = b + \sum_i x_i w_i$$

$$y = \begin{cases} 1 \text{ if } z \ge 0 \\ 0 \text{ otherwise} \end{cases}$$



Avoiding learning biases separately

- By a trick of adding 1 to input.
- We now can learn a bias as if it were a weight.
- Hence we get rid of the threshold.



A converging perceptron learning alg.

- If the output unit is correct, leave its weights unchanged.
- If the output unit incorrectly outputs a zero, add the input vector to the weight vector.
- If the output unit incorrectly outputs a 1, subtract the input vector from the weight vector.

This is guaranteed to find a set of weights that is correct for all training cases if such "solution" exists.

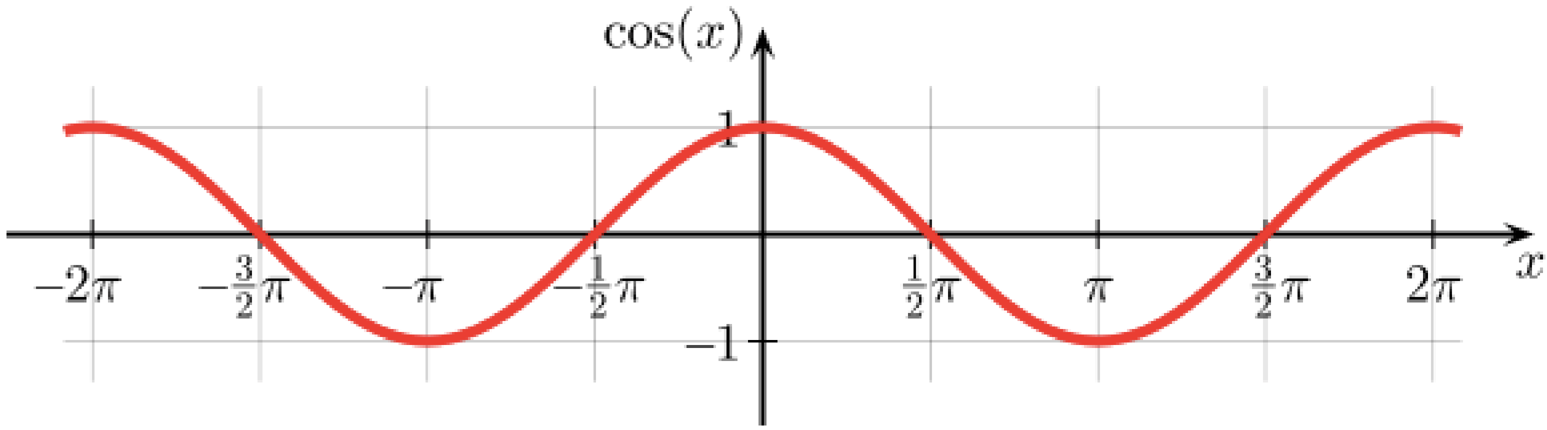
Weight space

- The dimension k is number of the weights $w=(w_1, \dots, w_k)$.
- A point in the space represents a weight vector $(w_1, \dots, w_k)$ as its coordinates .
- Each training case is represented as a hyper-plane through the origin (assuming we move the threshold to the bias weight)
  - The weights must lie on one side of this hyper-plane to get answer correct.

Remember dot product facts:

$a \bullet b = ||a||\,||b||\cos(\theta_{ab})$

$= a_1b_1+a_2b_2+ \ldots +a_nb_n$

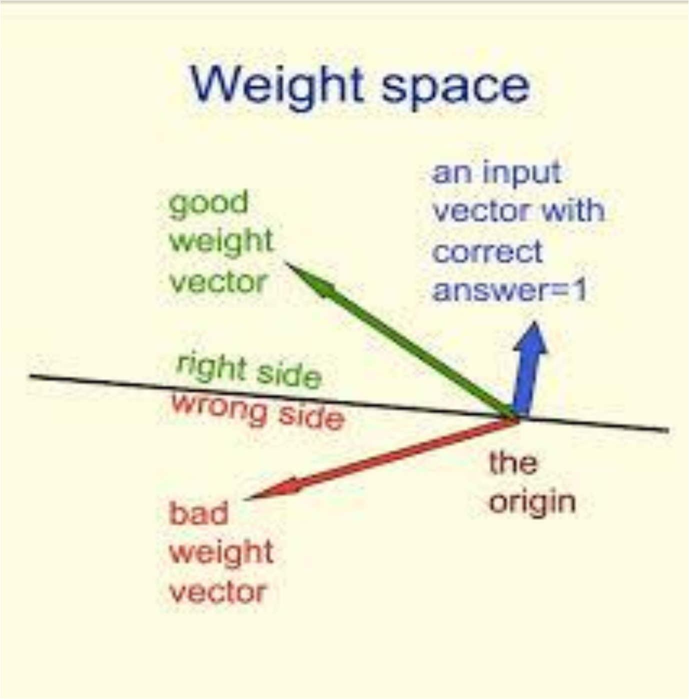


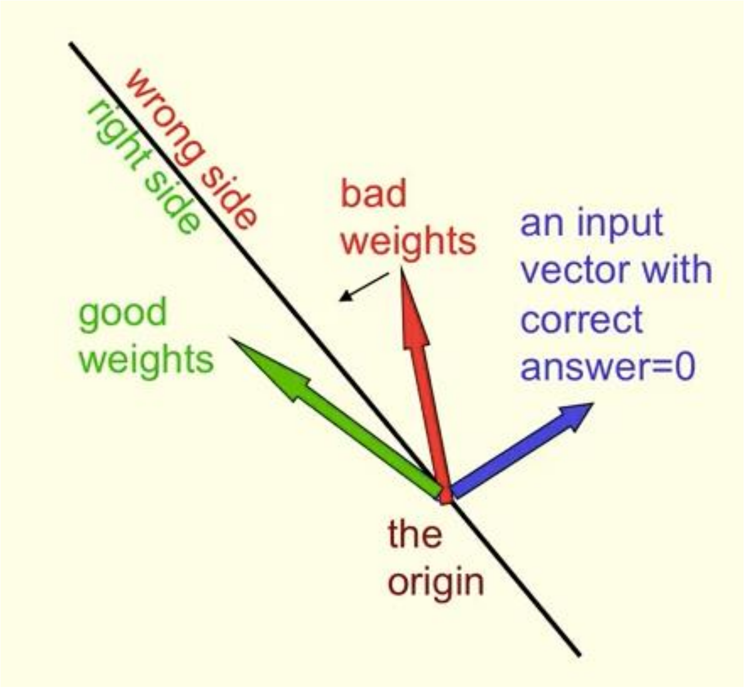
Thus, $a \bullet b \geq 0$, if $-\pi/2 \leq \theta_{ab} \leq \pi/2$

$a \bullet b \leq 0$, if $-\pi \leq \theta_{ab} \leq -\pi/2$ or $\pi/2 \leq \theta_{ab} \leq \pi$

Weight space

A point in the space represents a weight vector
Training case is a hyper-plane through the origin,
assuming threshold represented by bias.





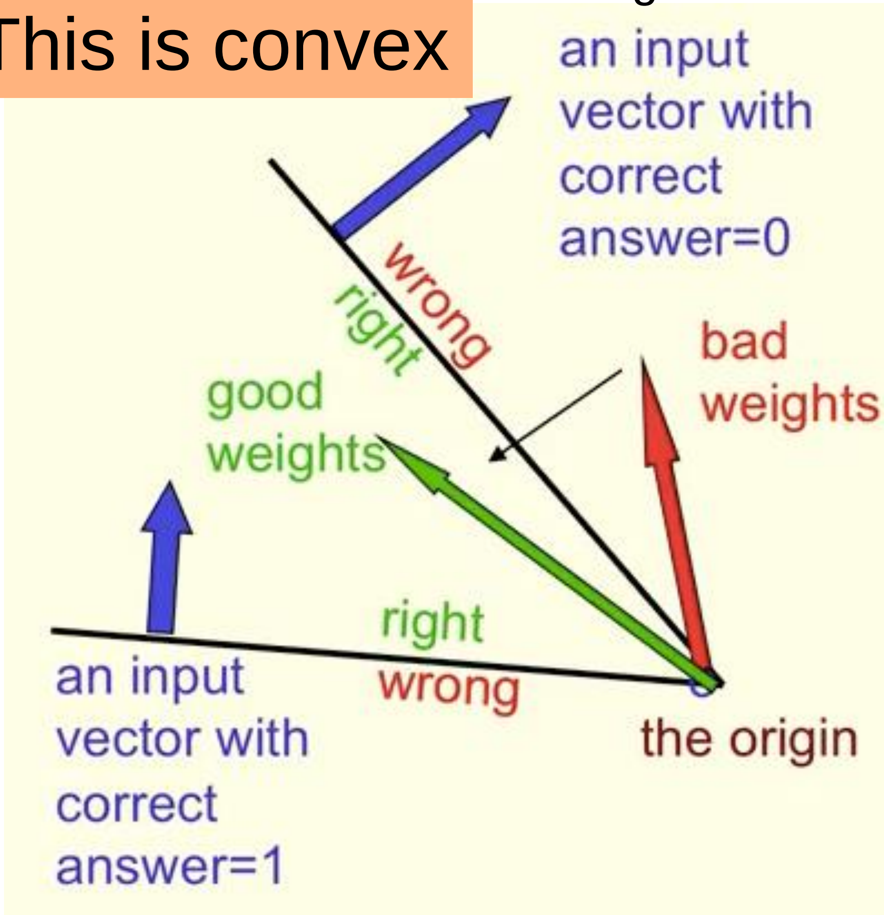
The cone of feasible solutions

To get all training cases right, we need to find a point on the "right side" of all planes (representing training cases).

The solution region, if exists, is a cone and is convex.

This is convex

A negative example



A positive example

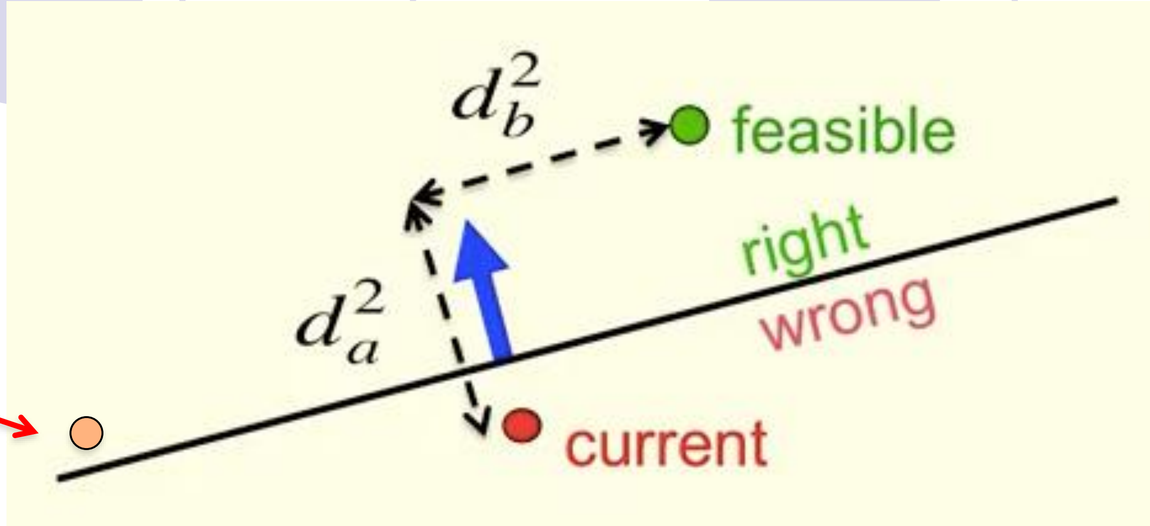
A converging perceptron learning alg.

- If the output unit is correct, leave its weights unchanged.
- If the output unit incorrectly outputs a zero, add the input vector to the weight vector.
- If the output unit incorrectly outputs a 1, subtract the input vector from the weight vector.

This is guaranteed to find a set of weights that is correct for all training cases if such solution exists.

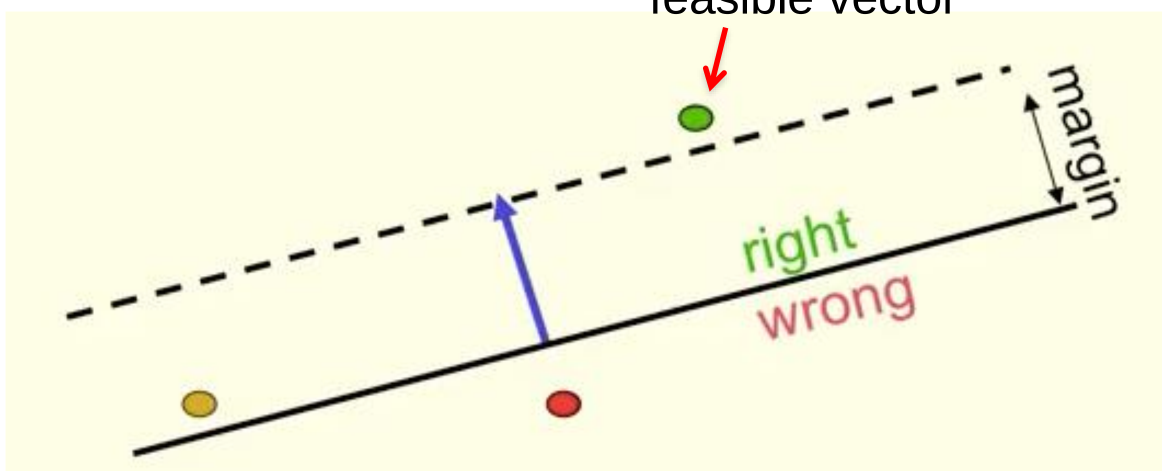
Proof of convergence by picture

But what about this point?
We might move farther.



Generously feasible vector

Proof: If there is a generously feasible vector, then each step we move closer to the feasible region. After finitely many steps, the weight vector is in the feasible region.



Note: this is assuming generously feasible vector exists.

The limitations of Perceptrons

- If we are allowed to choose features by hand, then we can do anything. But this is not learning.
- If we do not hand-pick features, then Minsky and Papert showed that perceptrons cannot do much. We will look at these proofs.

XOR cannot be learnt by a perceptron

- We prove that binary threshold output unit cannot do exclusive-or:

  Positive examples: (1,1) → 1; (0,0) →1

  Negative examples: (1,0)→0; (0,1) → 0

- The 4 input-output pairs give 4 inequalities, T being threshold:

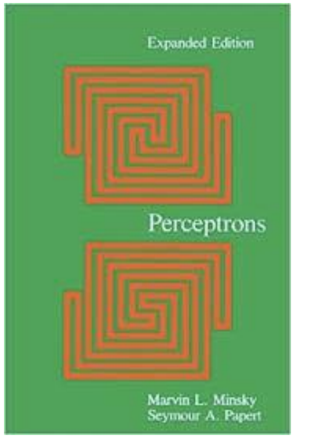
$$w_1 + w_2 \geq T,\ \ 0 \geq T \ \ \Rightarrow w_1 + w_2 \geq 2T$$

$$w_1 < T, \qquad w_2 < T \ \ \Rightarrow w_1 + w_2 < 2T$$

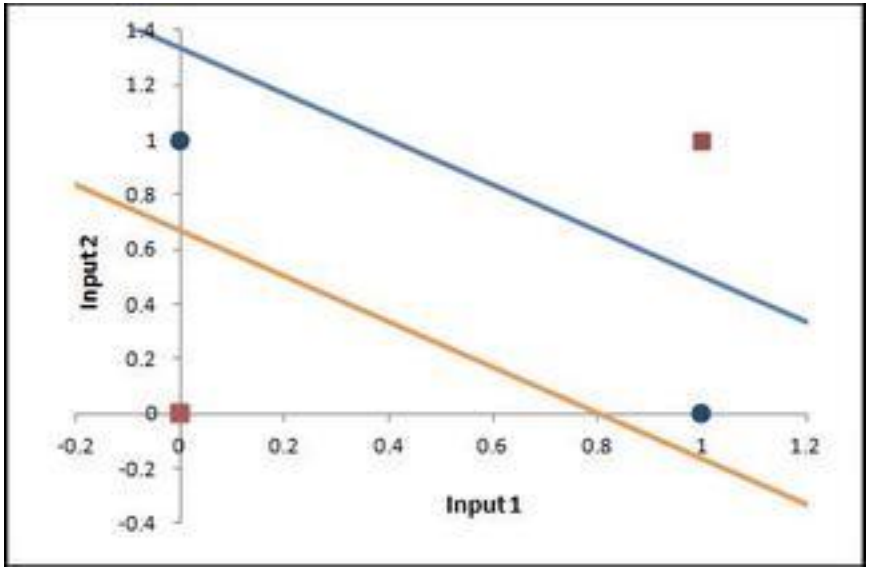
Contradiction. QED

Geometric view

Chapter 0
of this book



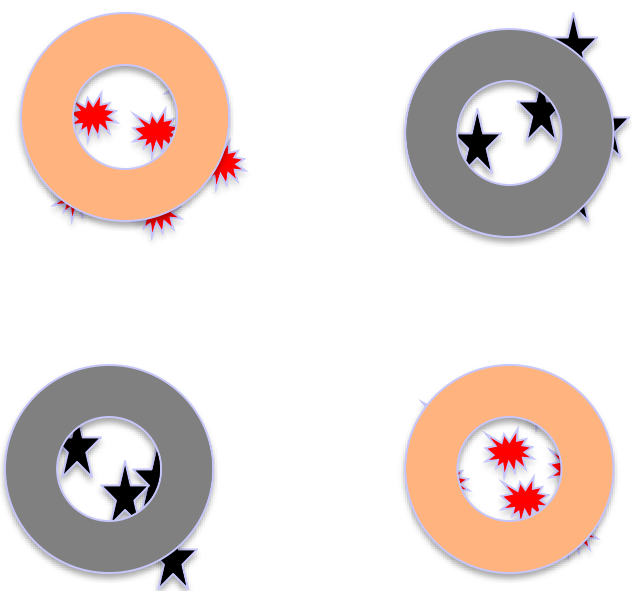
- Data-space view
  - Each input is point
  - A weight vector defines a hyperplane
  - The weight plane is perpendicular to the weight vector and misses the origin by a distance equal to the threshold

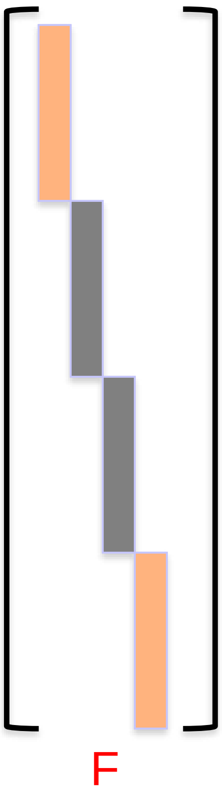


Blue dots and red dots are not linearly separable.

But this can be easily solved:

- Just pick right features (clusters)
- Then linearly separate the features, solves all.
- This is essentially what Rosenblatt initially claimed for perceptron. Chomsky & Papert actually attacked a different target.





$$\begin{bmatrix} -1 \\ +1 \\ +1 \\ -1 \end{bmatrix}$$

W

Group Invariance Theorem (Minsky-Papert):

Perceptron cannot distinguish following two patterns under translation.

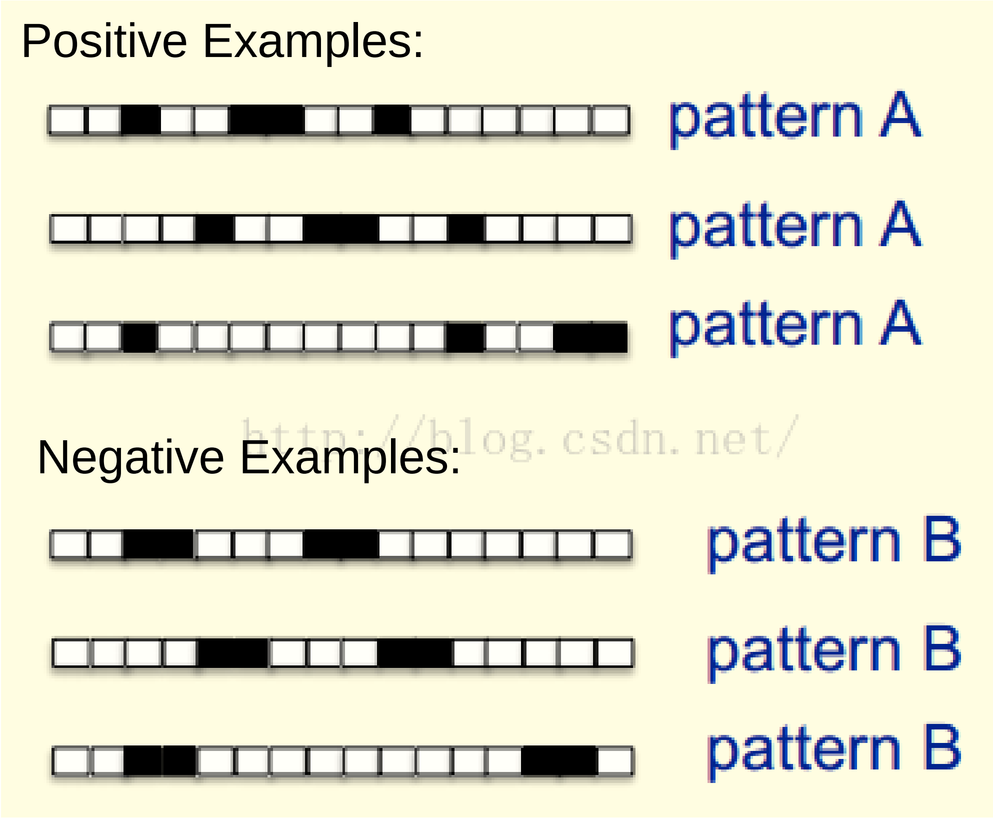
Proof.
Each pixel is activated by 4 different translations of both Pattern A and B.

Hence the total input received by the decision unit over all these patterns is four times the sum of all weights for both patterns A and B.

No threshold can always accept A & reject B. QED.

In general Perceptrons cannot do groups. Image translation forms a group. This was sometimes mis-interpreted as NN's are no good.

Hidden units can learn such features.
But deeper NN are hard to train.



Translation with wrap-around of two patterns

Basic Neurons

Basic Neurons

- To model neurons we have to idealize them:
  - Idealization removes complicated details that are not essential for understanding the main principles.
  - It allows us to apply mathematics and to make analogies to other familiar systems
  - Once we understand the basic principles, its easy to add complexity to make the model more faithful.

Linear neurons

- These are the basic building parts for all other neuron networks.

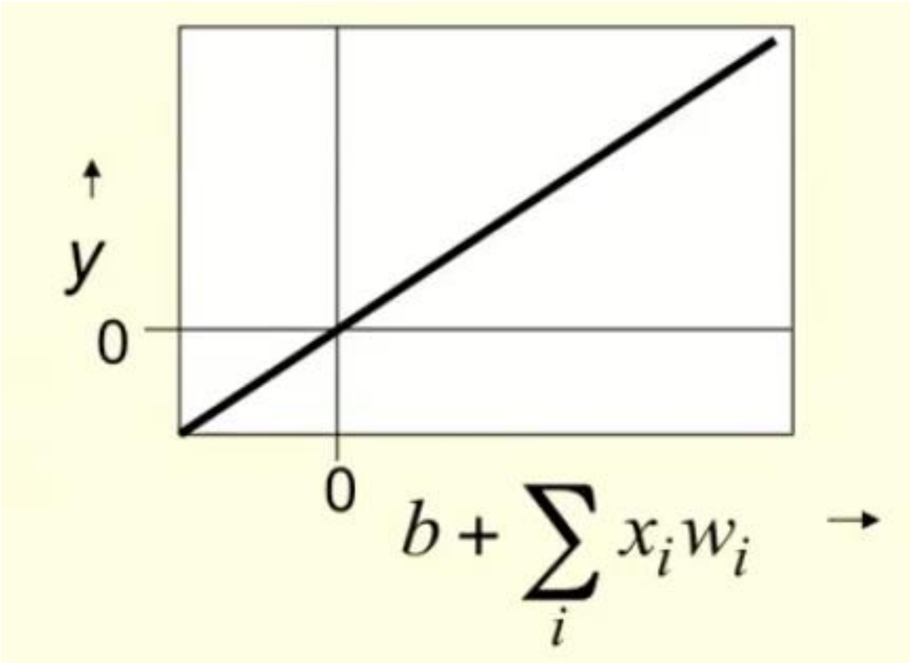
bias

i-th input

$$y = b + \Sigma_i \, x_i \, w_i$$

output

Weight on
i-th input



Binary threshold neuron

- McCulloch-Pitts (1943)
  - First compute a weighted sum of inputs
  - Then send out a fixed size spike of activity if the weighted sum exceeds a threshold.
  - McCulloch and Pitts thought that each spike is like the truth value of a proposition and each neuron combines truth values to compute the truth value of another proposition.
  - This has influenced Von Neumann.

There are two equivalent ways to describe a binary threshold neuron

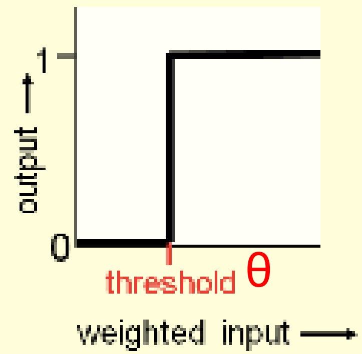
$$z = \sum_i x_i w_i$$

$$y = \begin{cases} 1 \text{ if } z \geq \theta \\ 0 \text{ otherwise} \end{cases}$$

$$\boxed{\theta = -b}$$

$$z = b + \sum_i x_i w_i$$

$$y = \begin{cases} 1 \text{ if } z \geq 0 \\ 0 \text{ otherwise} \end{cases}$$

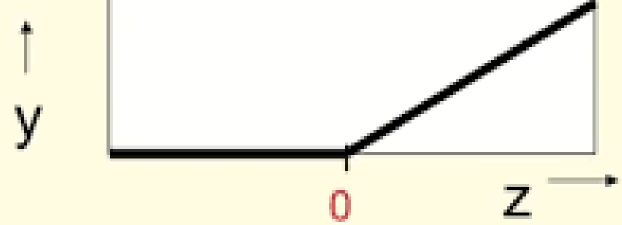


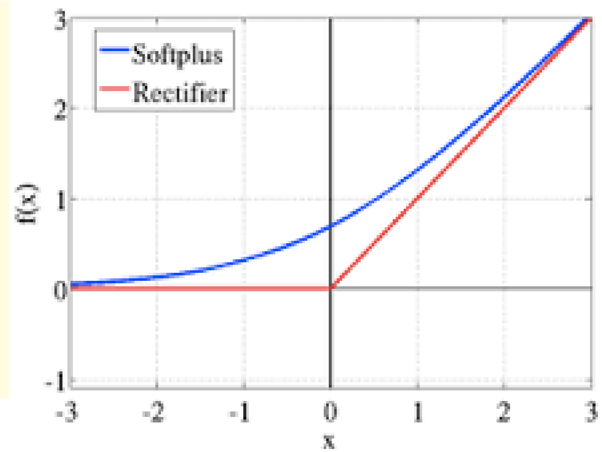
Rectified Linear Unit (ReLU)

- They compute a linear weighted sum of their inputs.
- The output is a non-linear function of the total input.
- This is the most popularly used neuron.

$$z = b + \sum_i x_i w_i$$

$$y = \begin{cases} z & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}$$





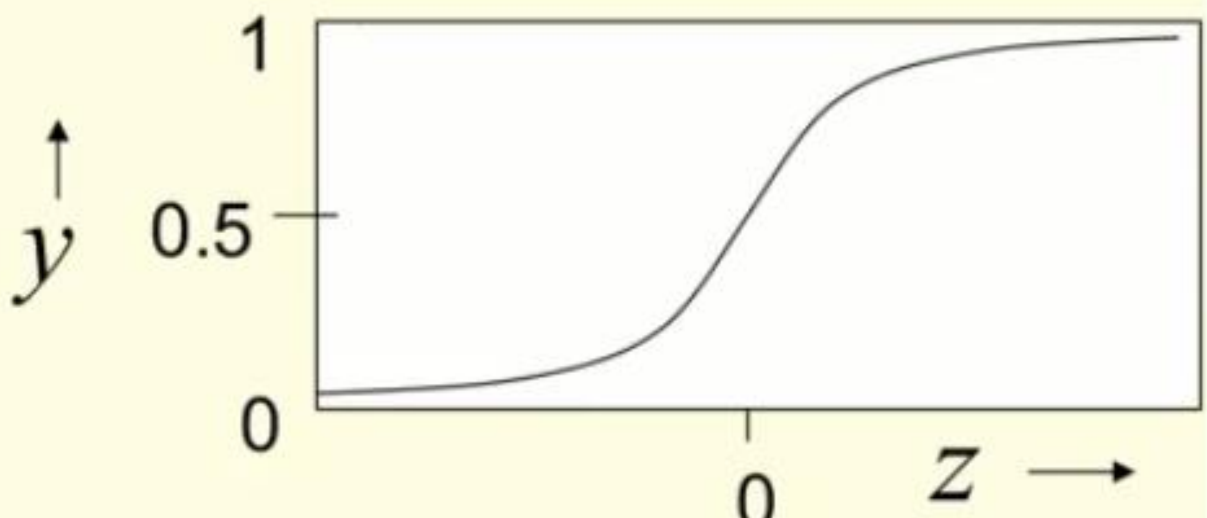
Or written as: f(x) = max {0,x}

A smooth approximation of the ReLU is "softplus" function

$$f(x) = \ln(1+e^x)$$

Sigmoid neurons

$$z = b + \sum_i x_i w_i \qquad y = \frac{1}{1 + e^{-z}}$$



Typically they use the logistic function

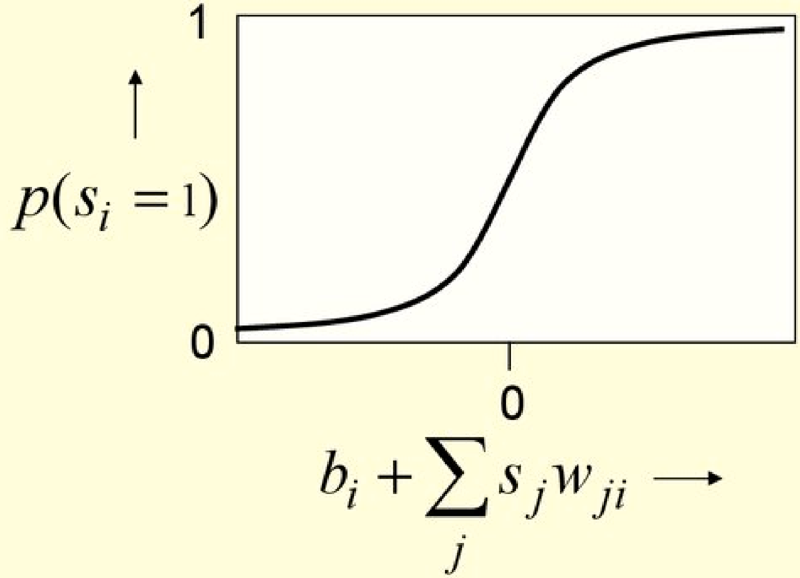
They have nice derivatives which makes learning easy.

But they cause vanishing gradients during backpropogation.

Stochastic binary neurons

(Bernoulli variables)

- These have a state of 1 or 0.
- The probability of turning on is determined by the weighted input from other units (plus a bias)



$$p(s_i = 1) = \frac{1}{1 + \exp(-b_i - \sum s_j w_{ji})}$$

Softmax function
(Normalized exponential function)

$$\sigma(x_j) = \frac{e^{x_j}}{\sum_i e^{x_i}}$$

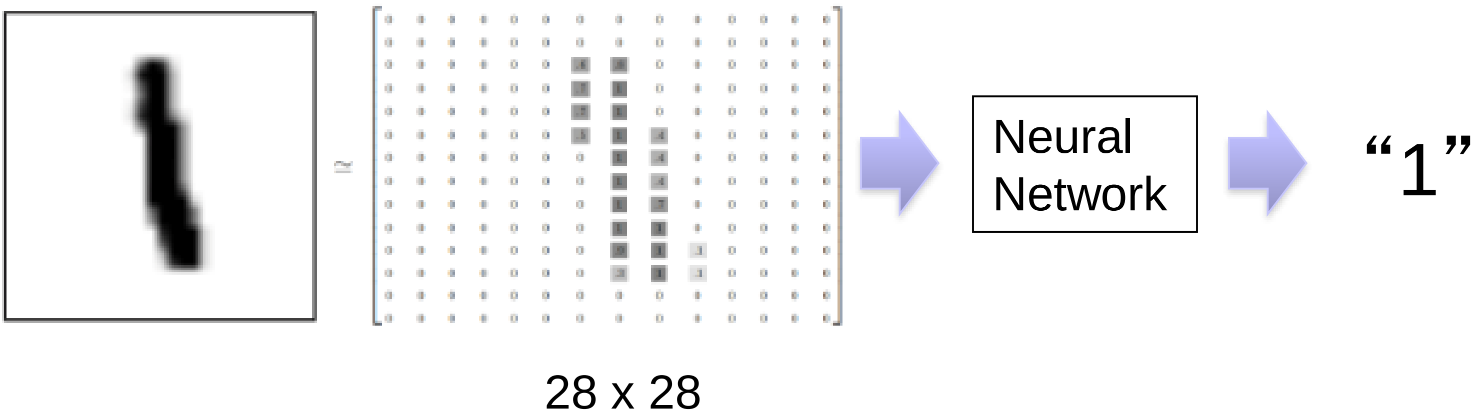
If we take an input of [1,2,3,4,1,2,3], the softmax of that is [0.024, 0.064, 0.175, 0.475, 0.024, 0.064, 0.175].
The softmax function highlights the largest values and suppress other values.

Comparing to “max” function, softmax is differentiable.

Fully Connected NN’s

Fully Connected NN &
Hello World of Deep Learning

0–9 handwritten digit recognition:

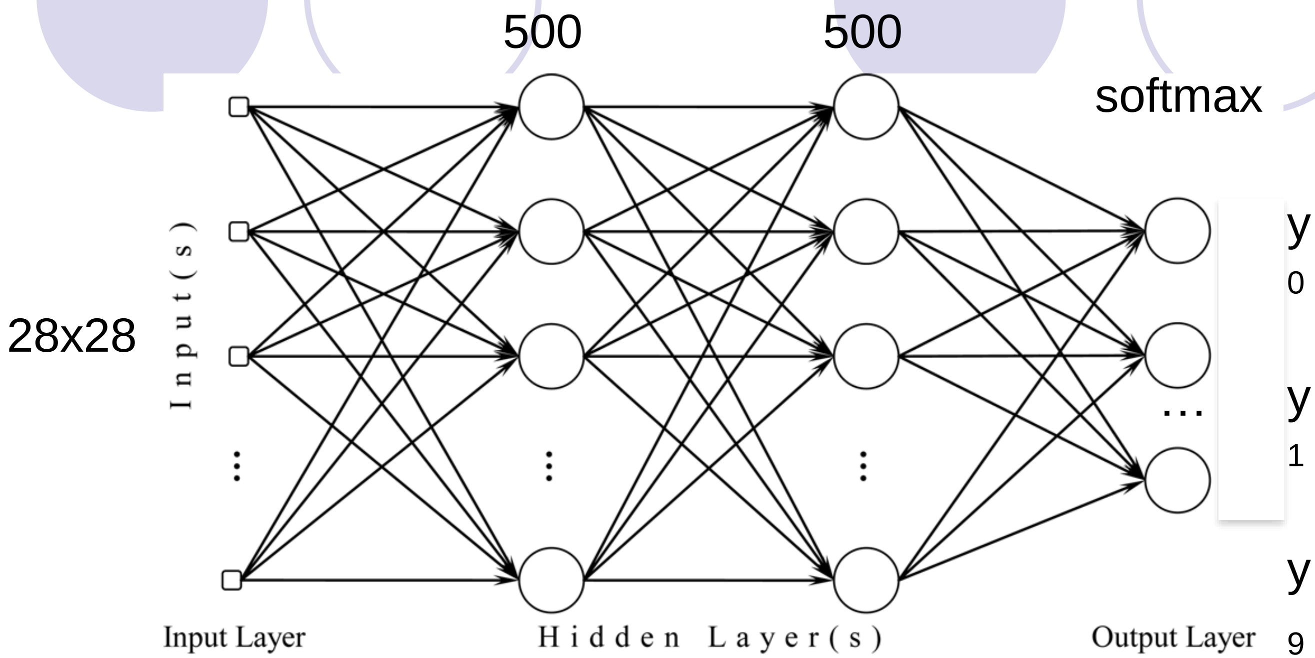


MNIST Data maintained by Yann LeCun: http://yann.lecun.com/exdb/mnist/
Keras provides data sets loading function at http://keras.io/datasets

Keras & Tensorflow

- Interface of Tensorflow and Theano.
- Francois Chollet, author of Keras is at Google, Keras will become Tensorflow API.
- Documentation: http://keras.io.
- Examples: https://github.com/fchollet/keras/tree/master/examples
- Simple course on Tensorflow: https://docs.google.com/presentation/d/1zkmVGobdPfQgsjIw6gUqJsjB8wvv9uBdT7ZHdaCjZ7Q/edit#slide=id.p

Implementing in Keras



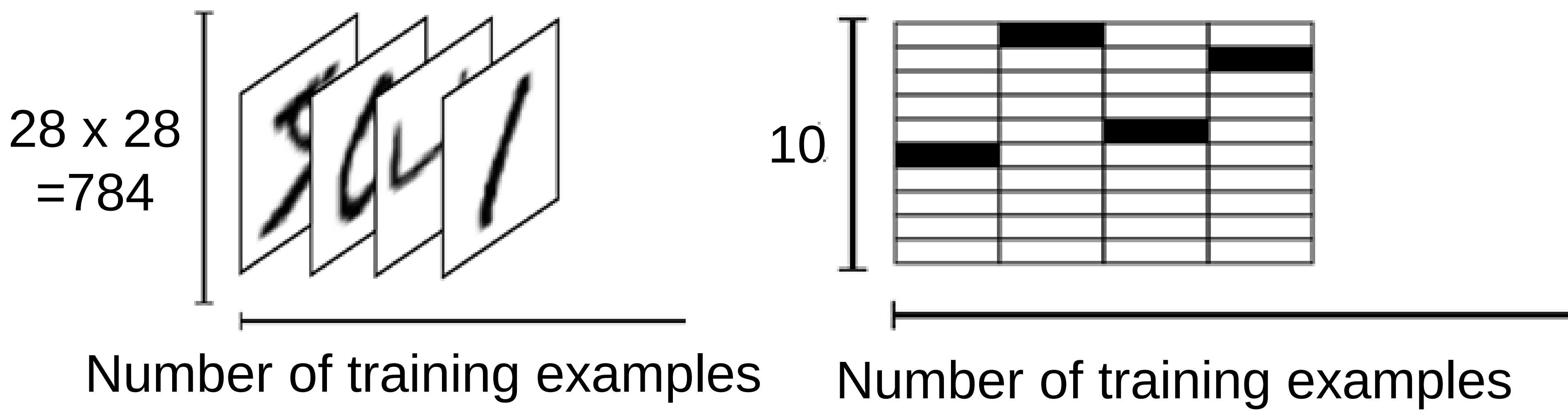
Fully connected NN

```
model = sequential()   # layers are sequentially added
model.add( Dense(input_dim=28*28, output_dim=500))
model.add(Activation('sigmoid'))  #: softplus, softsign,relu,tanh, hard_sigmoid
model.add(Dense( output_dim = 500))
model.add (Activation('sigmoid'))
Model.add(Dense(output_dim=10))
Model.add(Activation('softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(x_train, y_train, batch_size=100, nb_epoch=20)
```

Training

```
model.fit(x_train, y_train, batch_size=100, nb_epoch=20)
```

numpy array

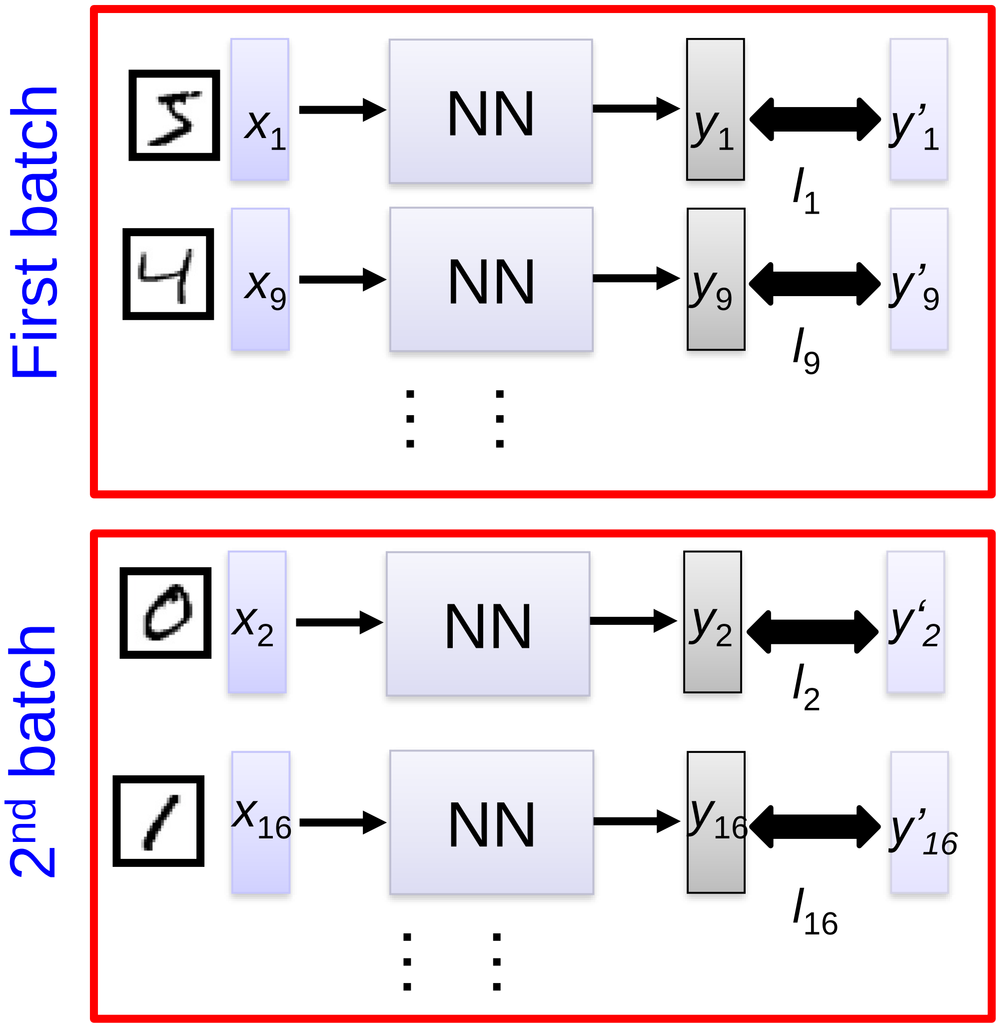


We do not really minimize total loss!

Batch:

model.fit(x_train, y_train, batch_size=100, nb_epoch=20)

parallel processing



- Randomly initialize network parameters
- Pick the 1st batch

  $L' = l_1 + l_9 + \ldots$

  Update parameters
- Pick the 2nd batch

  $L'' = l_2 + l_{16} + \ldots$

  Update parameters

  ⋮
- Until all batches have been picked

one epoch

Repeat the above process

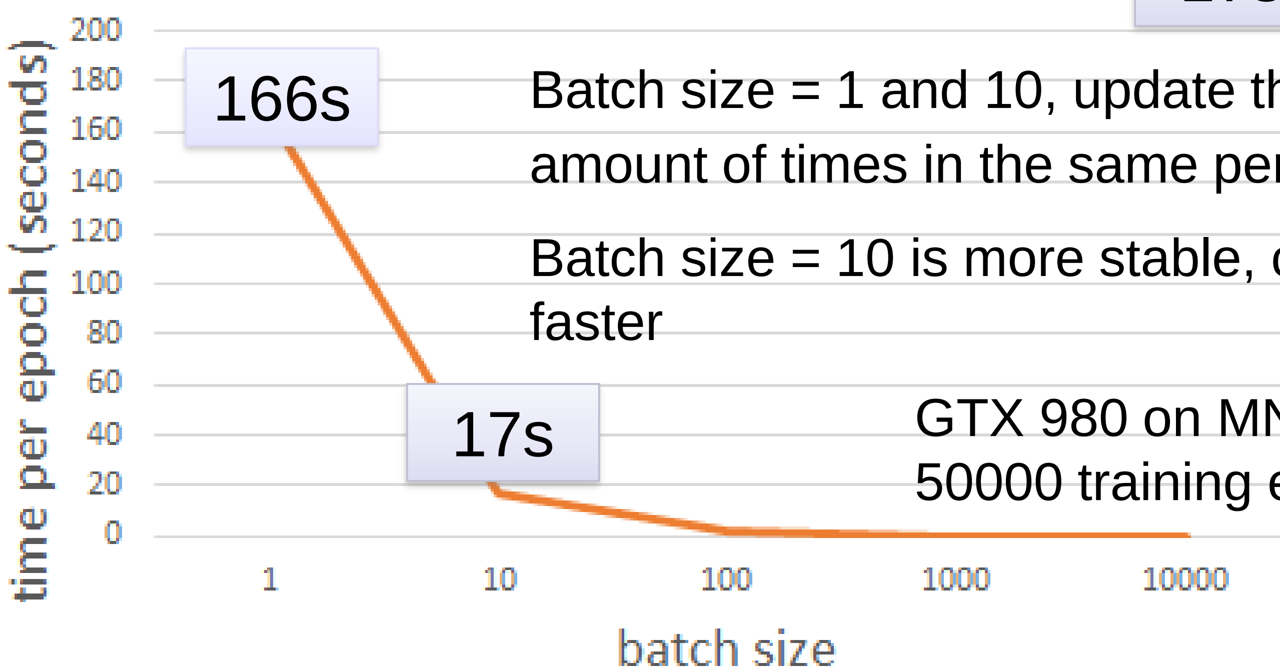
Speed

Very large batch size can yield worse performance

- Smaller batch size means more updates in one epoch
  - E.g. 50000 examples
  - batch size = 1, 50000 updates in one epoch
  - batch size = 10, 5000 updates in one epoch

166s 1 epoch

17s 10 epochs



Batch size = 1 and 10, update the same amount of times in the same period.

Batch size = 10 is more stable, converge faster

GTX 980 on MNIST with 50000 training examples

Background Theory

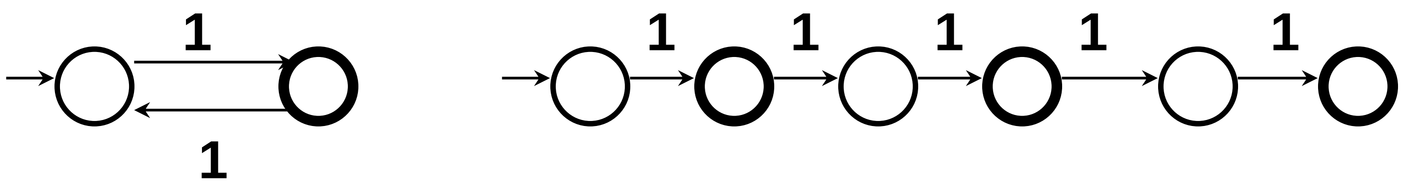
Importance of being small

- Neural networks can approximate any function. Overfiting is a major concern. In some sense, it is possible to view the development of deep learning from an angle of reducing the (Kolmogorov) complexity of neural networks: CNN, RNN, dropout, regularization, and esp. depth.

- Occam's Razor: Commonly attributed to William of Ockham (1290--1349). This was formulated about fifteen hundred years after Epicurus. In sharp contrast to the principle of multiple explanations, it states: Entities should not be multiplied beyond necessity.
- Commonly explained as: when have choices, choose the simplest theory.
- Bertrand Russell: ``It is vain to do with more what can be done with fewer.' '
- Newton (*Principia*): ``Natura enim simplex est, et rerum causis superfluis non luxuriat''.

Example. Inferring a deterministic finite automaton (DFA)

- A DFA accepts: 1, 111, 11111, 1111111; and rejects: 11, 1111, 111111. What is it?



- There are actually infinitely many DFAs satisfying these data.
- The first DFA makes a nontrivial inductive inference, the 2nd does not.
- The 2nd one “over fits” the data, cannot make further predictions.

Exampe. History of Science

- Maxwell's (1831-1879)'s equations say that:
  - (a) An oscillating magnetic field gives rise to an oscillating electric field;
  - (b) an oscillating electric field gives rise to an oscillating magnetic field.

  Item (a) was known from M. Faraday's experiments. However (b) is a theoretical inference by Maxwell and his aesthetic appreciation of simplicity. The existence of such electromagnetic waves was demonstrated by the experiments of H. Hertz in 1888, 8 years after Maxwell's death, and this opened the new field of radio communication. Maxwell's theory is even relativistically invariant. This was long before Einstein's special relativity. As a matter of fact, it is even likely that Maxwell's theory influenced Einstein's 1905 paper on relativity which was actually titled `On the electrodynamics of moving bodies'.
- J. Kemeny, a former assistant to Einstein, explains the transition from the special theory to the general theory of relativity: At the time, there were no new facts that failed to be explained by the special theory of relativity. Einstein was purely motivated by his conviction that the special theory was not the simplest theory which can explain all the observed facts. Reducing the number of variables obviously simplifies a theory. By the requirement of general covariance Einstein succeeded in replacing the previous 'gravitational mass' and `inertial mass' by a single concept.
- Double helix vs triple helix --- 1953, Watson & Crick

Bayesian Inference

- Bayes Formula:

  $P(H|D) = P(D|H)P(H)/P(D)$
- By Occam's razor, $P(H)=2^{-K(H)}$, (smallest most likely).
- Take -log, maximize $P(H|D)$ becomes minimize:

  $-\log P(D|H) + K(H)$ (modulo $\log P(D)$, constant).

  where
  - $-\log P(D|H)$ is the coding length of D given H.
  - $K(H)$ is the smallest description of model H (Kolmogorov complexity of H).

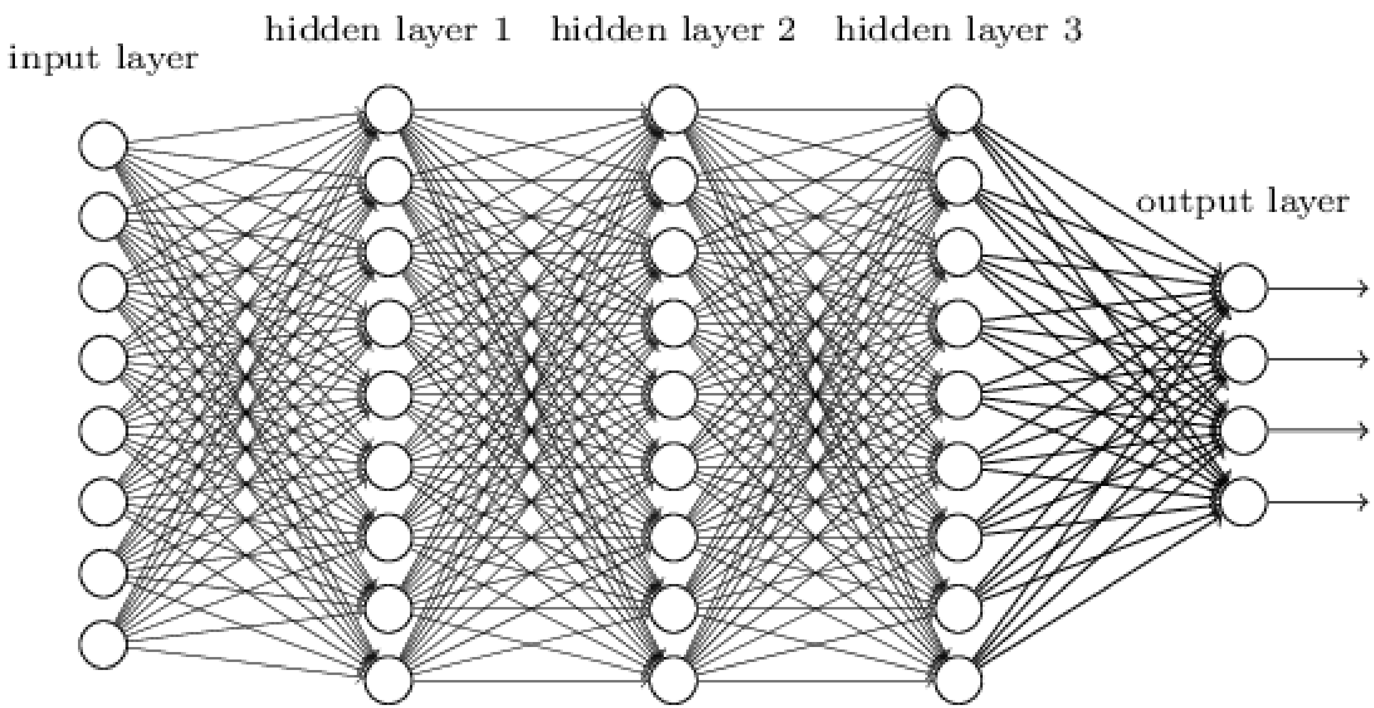
Note on PAC Learning, other theorems

- Here is an informal statement: given data (positive and negative examples drawn from distribution D), if you find a model M that agrees with the data, and size of M is polynomially smaller than the data, then with high probability (according to D) , M is correct with a small number of errors.

CNN's

Smaller Network: CNN

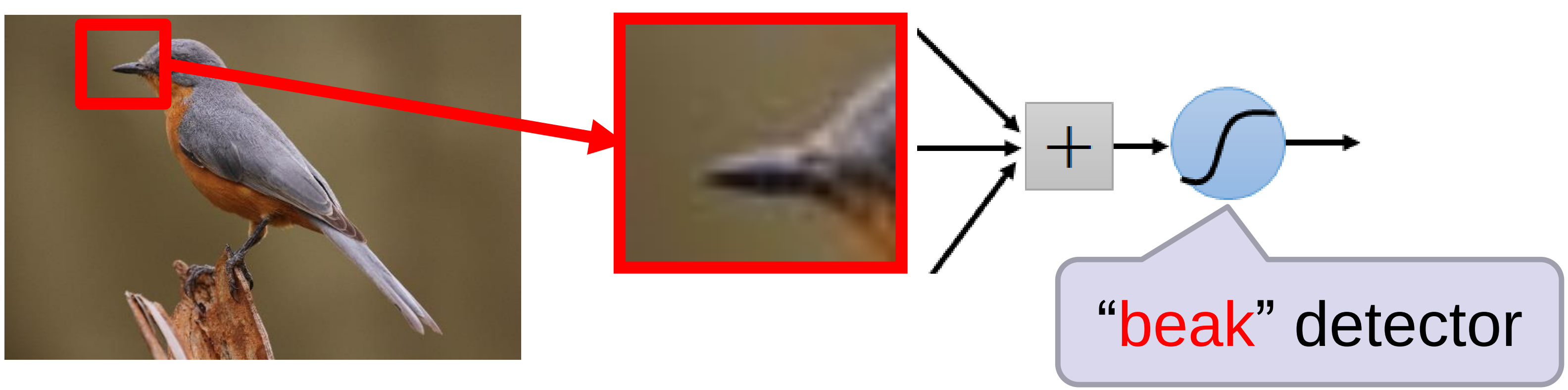
- We know it is good to learn a small model.
- From this fully connected model, do we really need all the edges?
- Can some of these be shared?



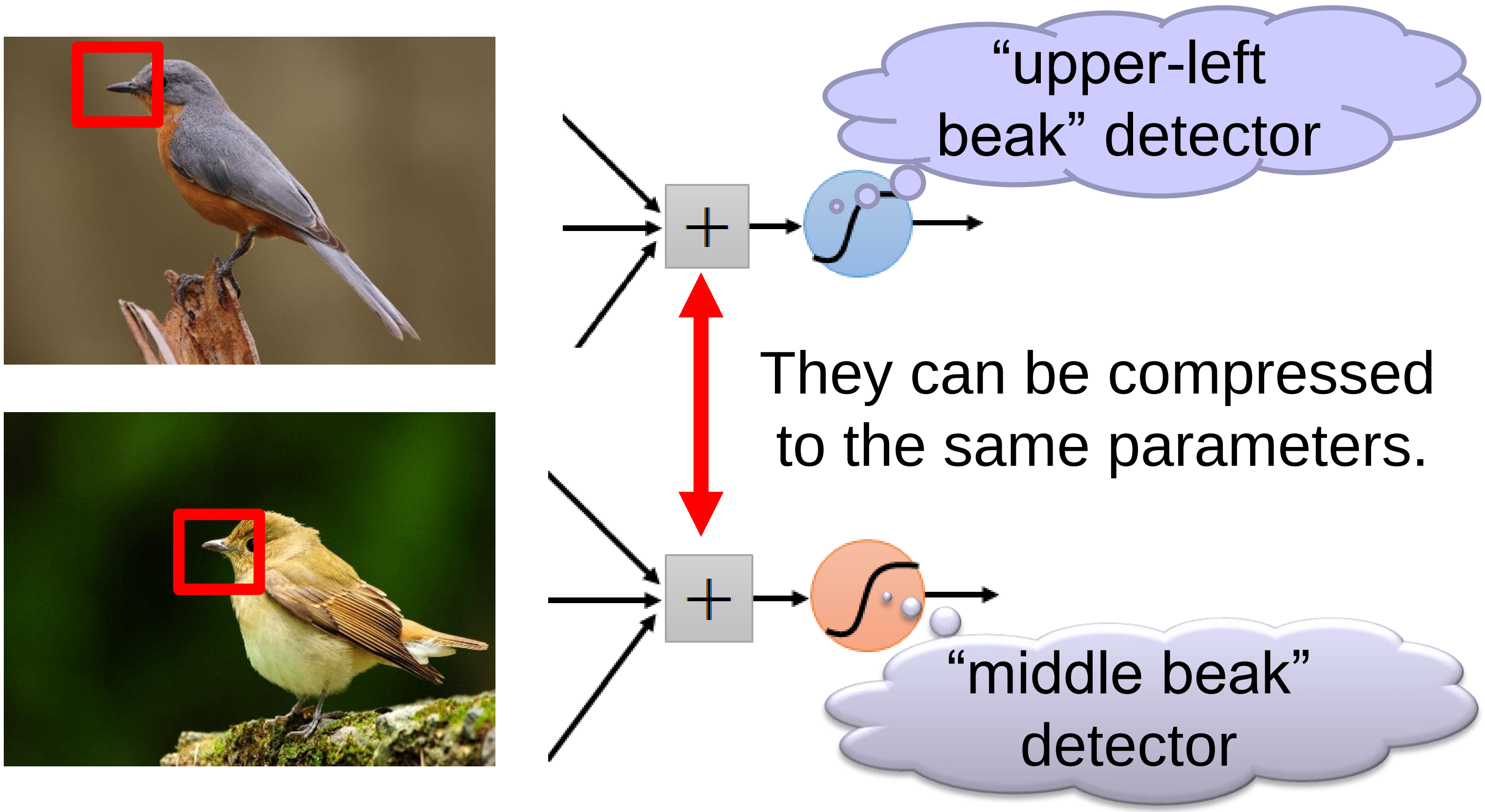
Consider learning an image:

- Some patterns are much smaller than the whole image

Can represent a small region with fewer parameters

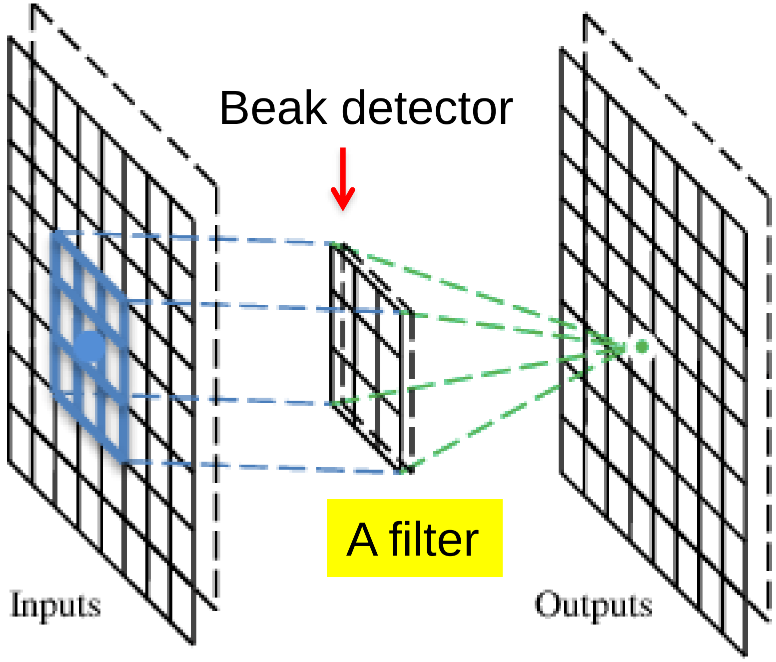


Same pattern appears in different places:
They can be compressed!
What about training a lot of such “small” detectors and each detector must “move around”.



A convolutional layer

A CNN is a neural network with some convolutional layers (and some other layers). A convolutional layer has a number of filters that does convolutional operation.



Convolution

These are the network parameters to be learned.

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

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

⋮ ⋮

Each filter detects a small pattern (3 x 3).

Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

stride=1

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

Dot product

3 -1

6 x 6 image

Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

If stride=2

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

6 x 6 image

3 -3

Convolution

stride=1

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

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

6 x 6 image

3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

Convolution

-1	1	-1
-1	1	-1
-1	1	-1

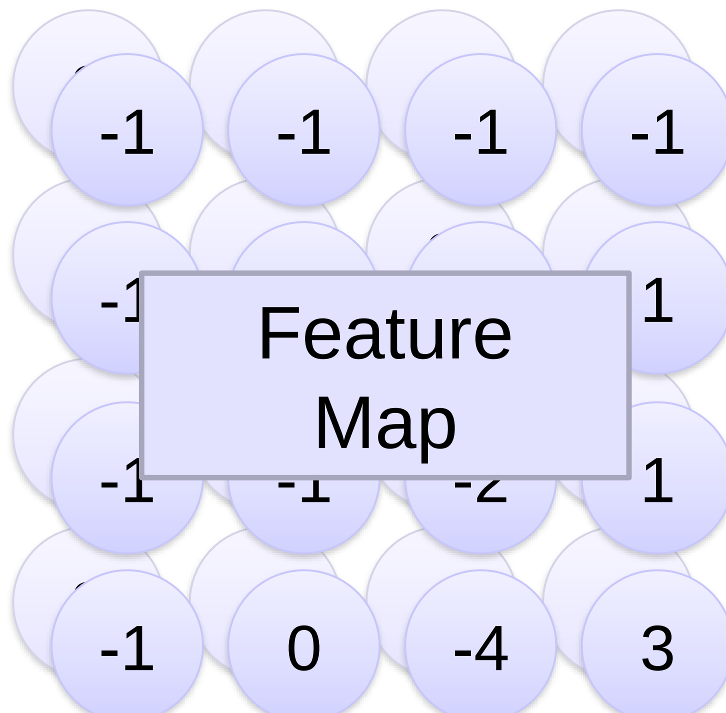
Filter 2

stride=1

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

6 x 6 image

Repeat this for each filter



Two 4 x 4 images
Forming 2 x 4 x 4 matrix

Color image: RGB 3 channels

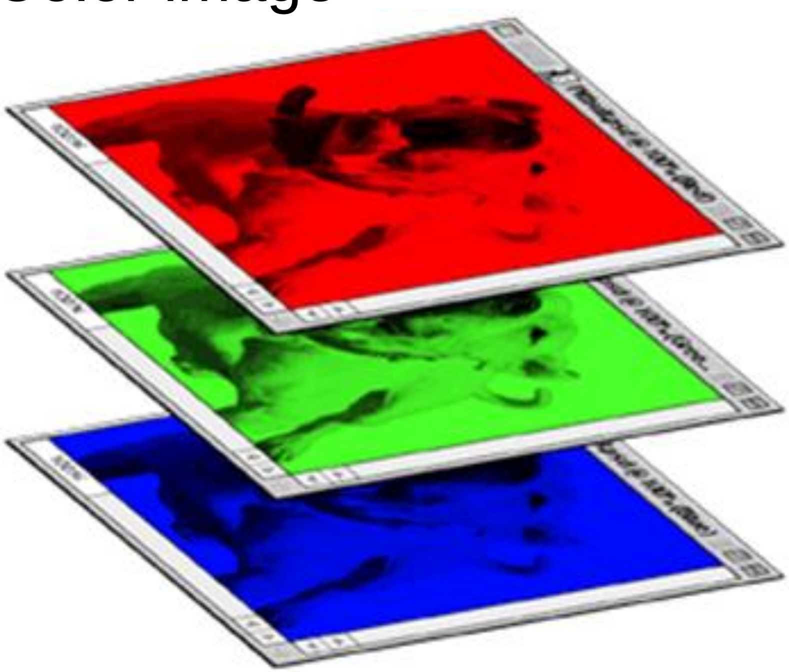
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

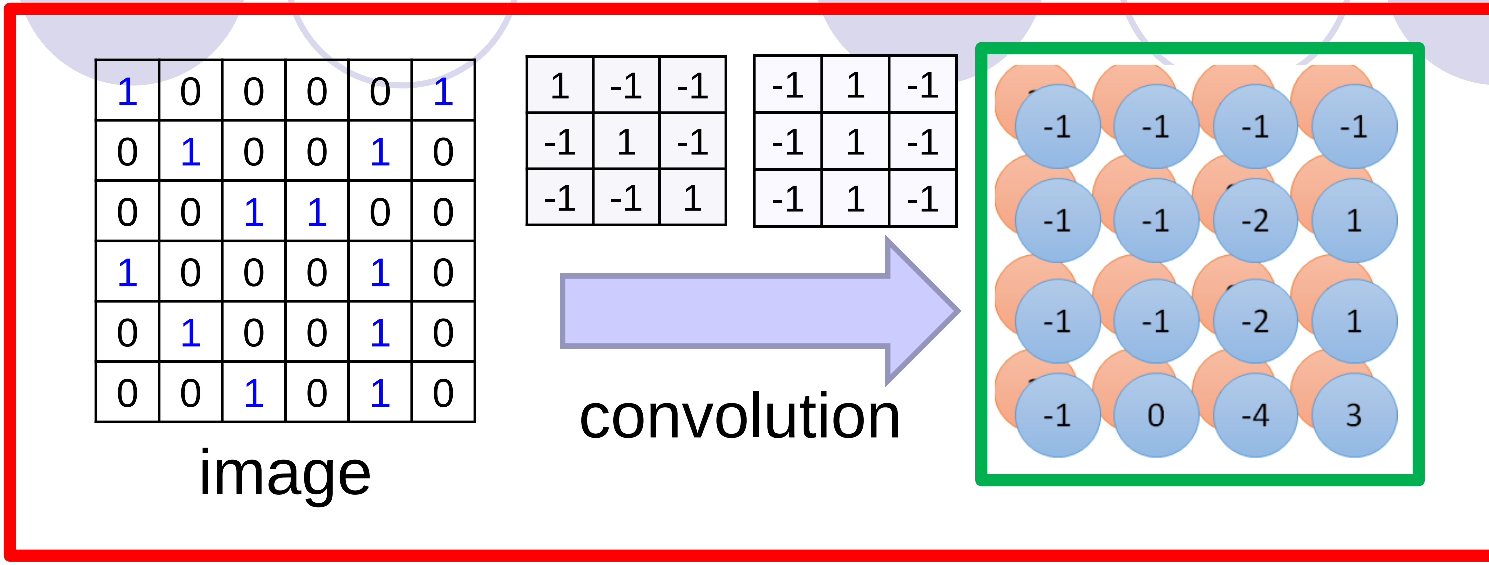
Filter 2

Color image

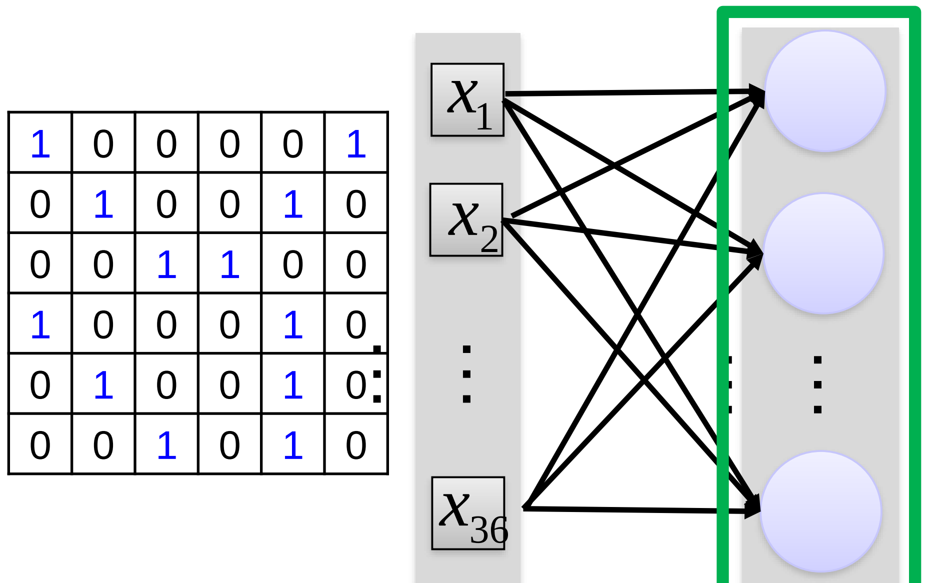


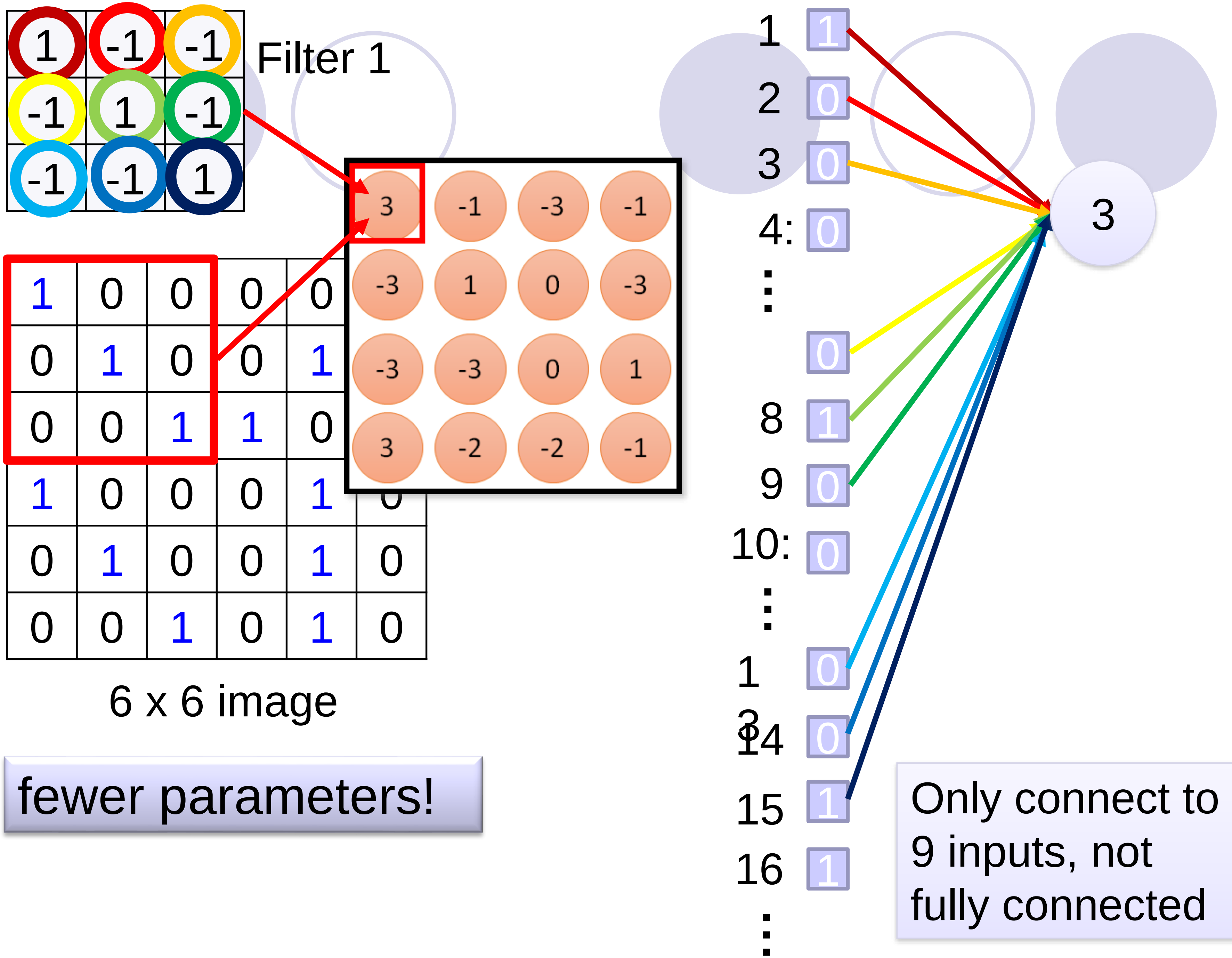
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

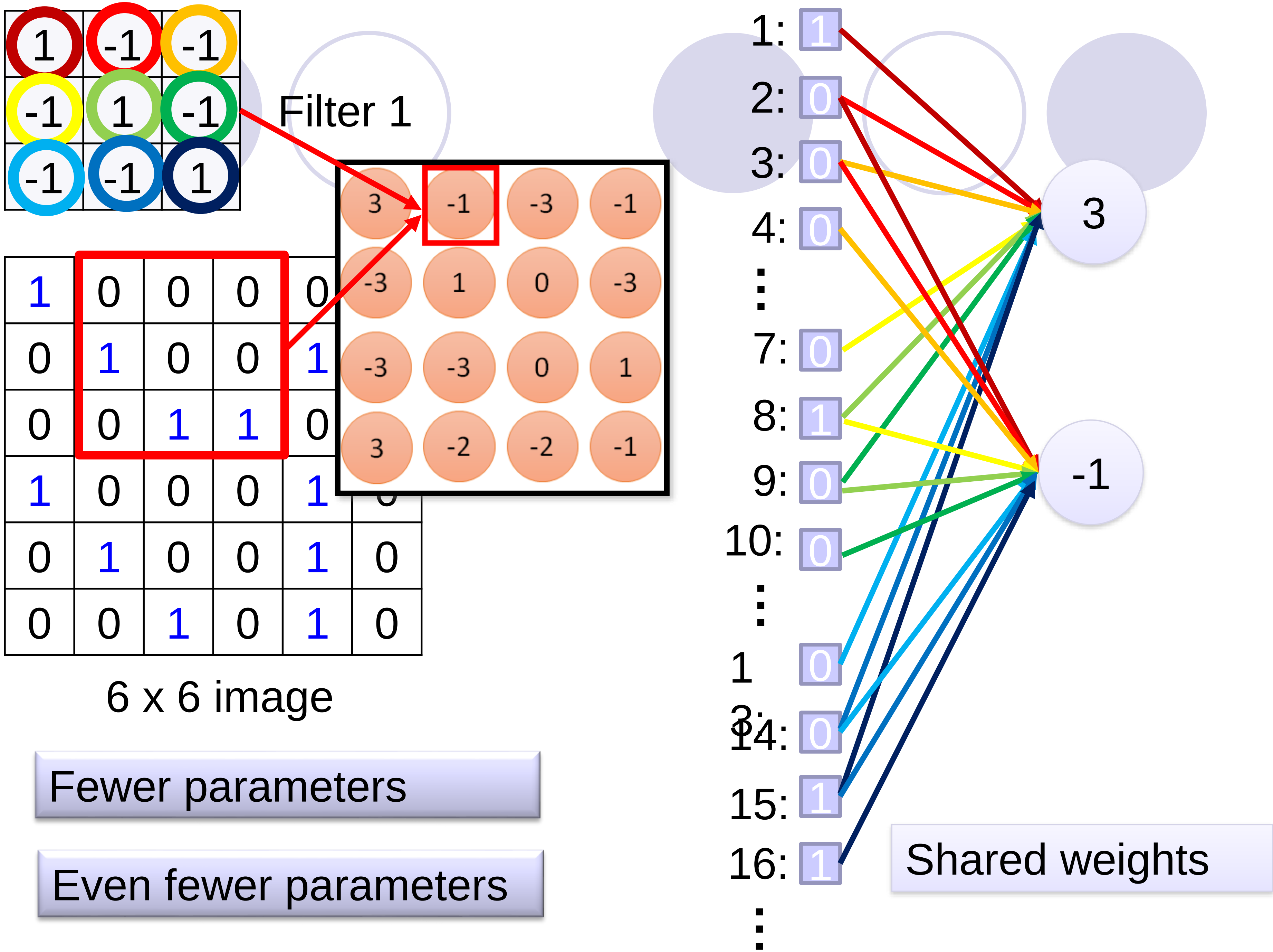
Convolution v.s. Fully Connected



Fully-connected

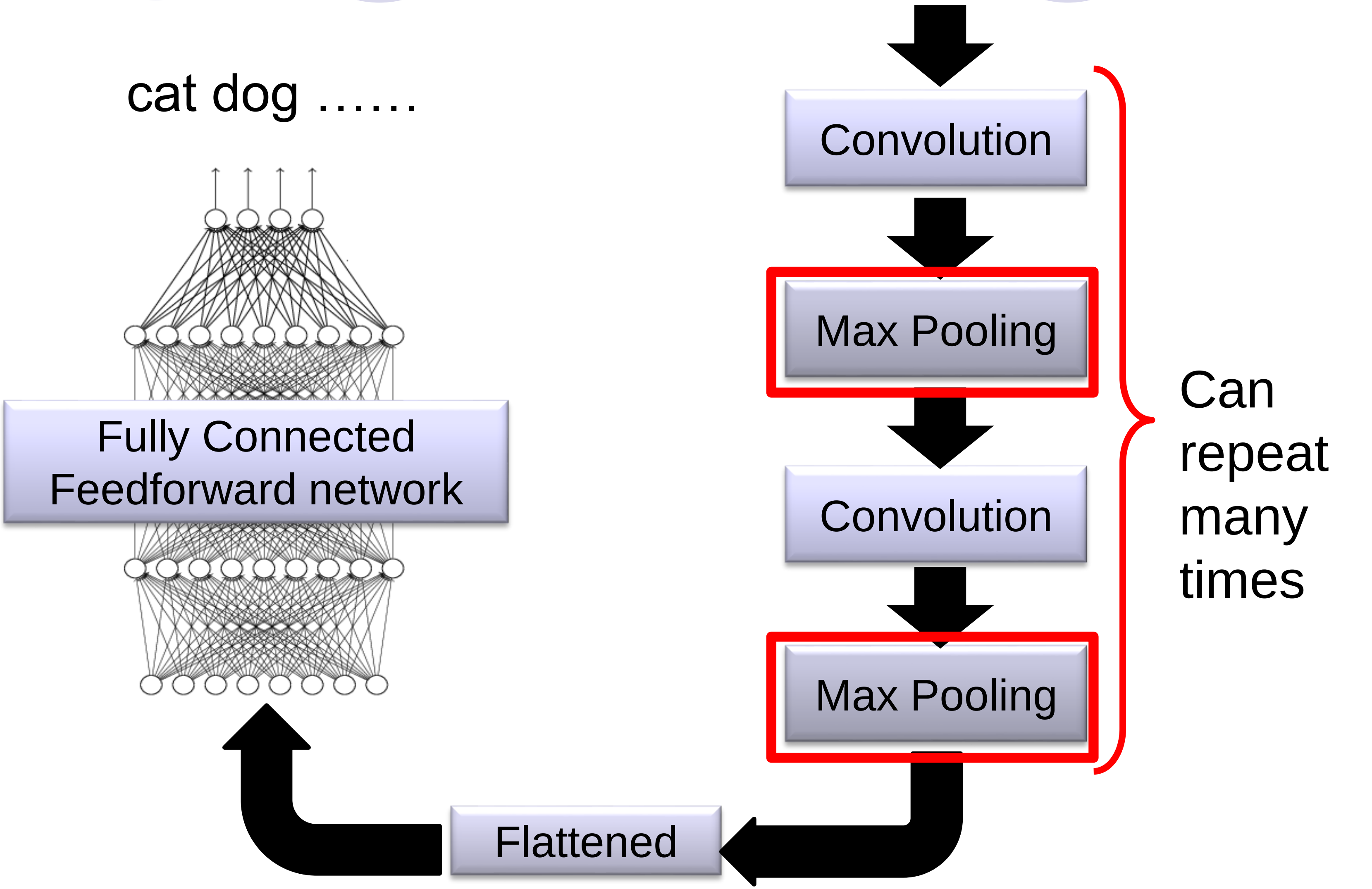



Filter 1
6 x 6 image
fewer parameters!
Only connect to 9 inputs, not fully connected



1 -1 -1
-1 1 -1
-1 -1 1
Filter 1
3 -1 -3 -1
-3 1 0 -3
-3 -3 0 1
3 -2 -2 -1
1 0 0 0 0
0 1 0 0 1
0 0 1 1 0
1 0 0 0 1 0
0 1 0 0 1 0
0 0 1 0 1 0
6 x 6 image
1: 1
2: 0
3: 0
4: 0
7: 0
8: 1
9: 0
10: 0
13: 0
14: 0
15: 1
16: 1
3
-1
Shared weights
Fewer parameters
Even fewer parameters

The whole CNN



Max Pooling

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

-1	-1	-1	-1
-1	-1	-2	1
-1	-1	-2	1
-1	0	-4	3

Why Pooling

- Subsampling pixels will not change the object

bird

Subsampling

bird

We can subsample the pixels to make image smaller

fewer parameters to characterize the image

A CNN compresses a fully connected network in two ways:

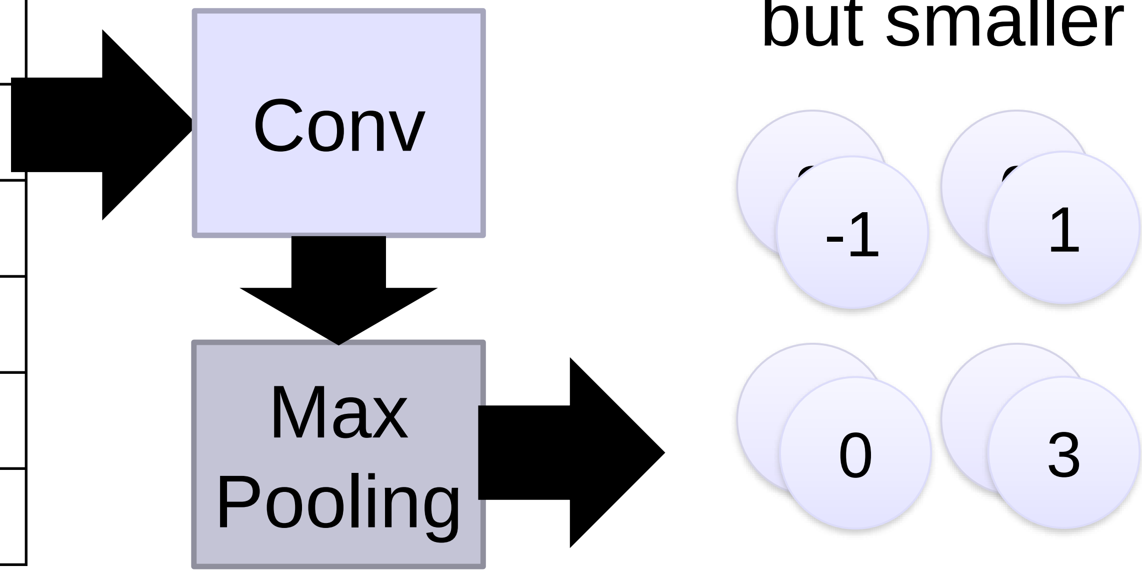
- Reducing number of connections
- Shared weights on the edges
- Max pooling further reduces the complexity

Max Pooling

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

6 x 6 image

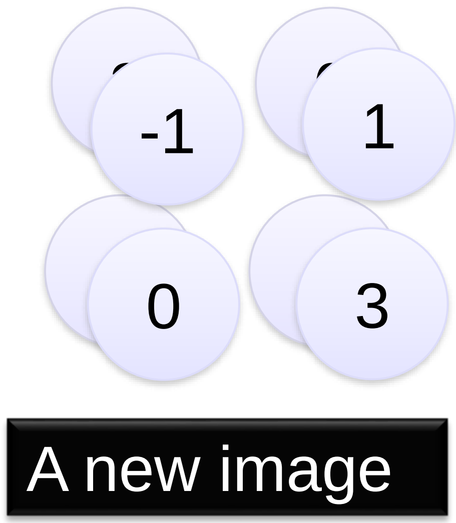
New image
but smaller

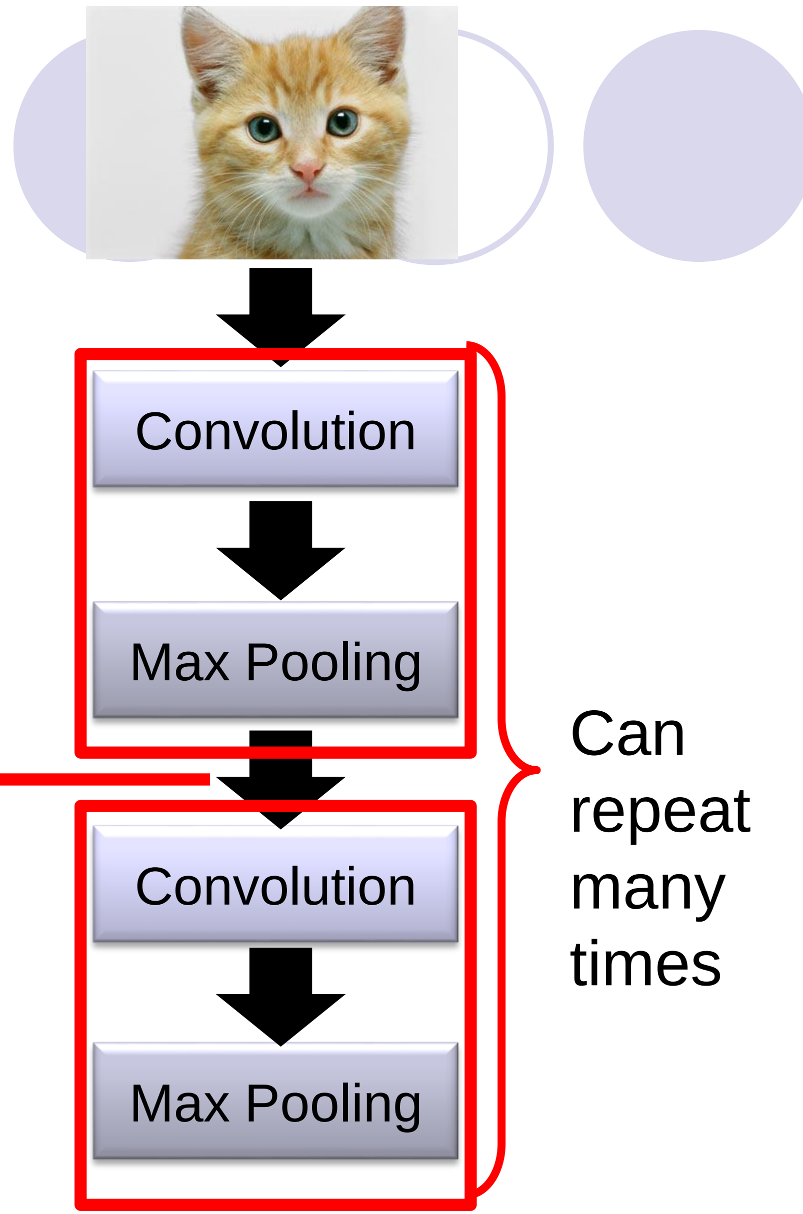


2 x 2 image

Each filter
is a channel

The whole CNN

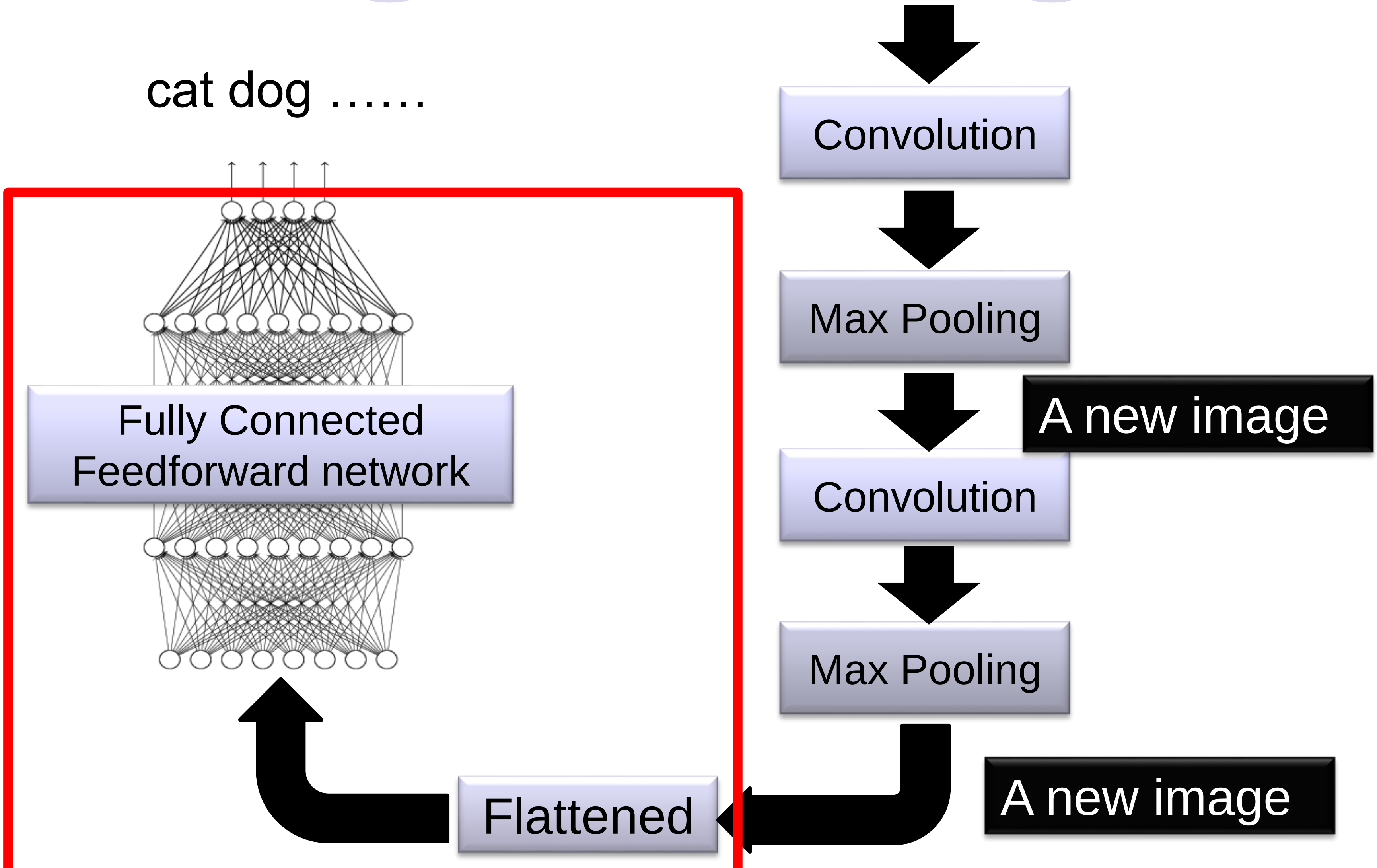




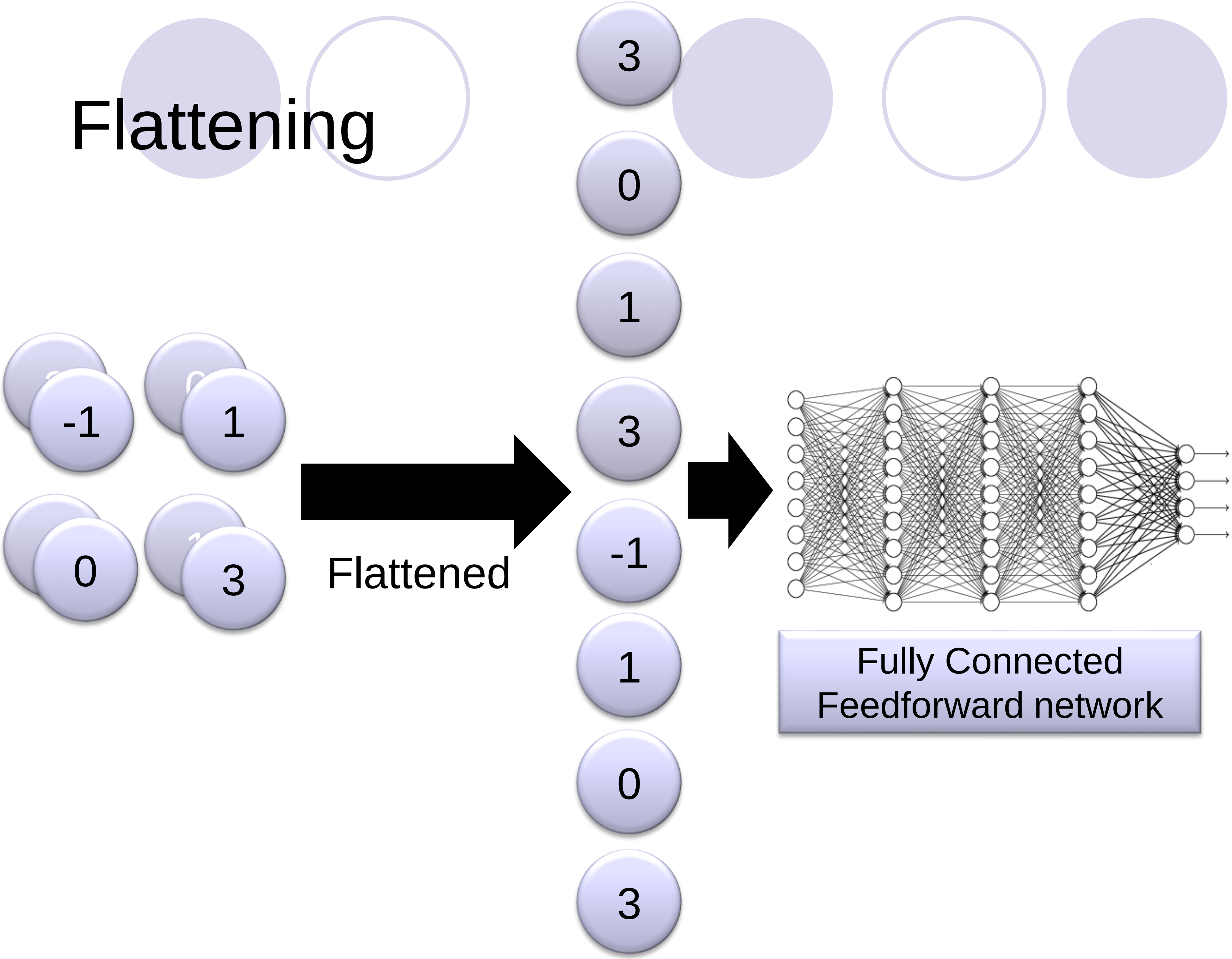
Smaller than the original image

The number of channels is the number of filters

The whole CNN



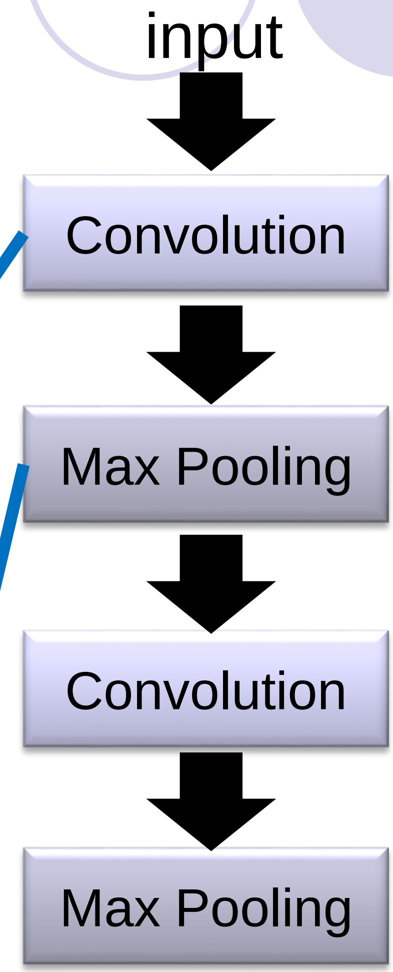
Flattening



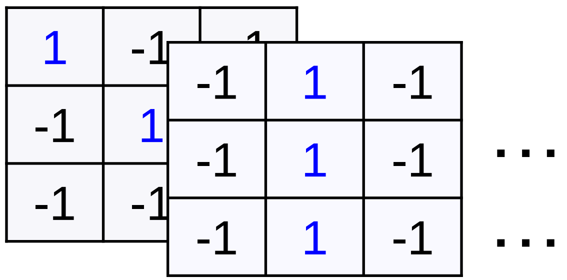
-1
1
0
3
Flattened
3
0
1
3
-1
1
0
3
Fully Connected
Feedforward network

CNN in Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D tensor)***




```
model2.add( Convolution2D( 25,3,3,
                input_shape=(28,28,1)) )
```



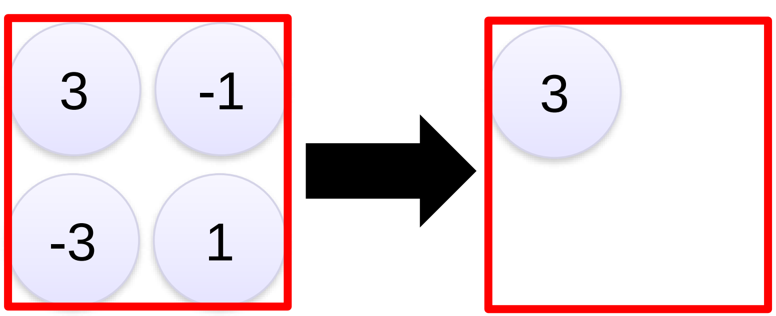
There are 25 3x3 filters.

Input_shape = (28 , 28 , 1)

28 x 28 pixels

1: black/white, 3: RGB

```
model2.add(MaxPooling2D((2,2)))
```



CNN in Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D array)***

Input

1 x 28 x 28

Convolution

```
model2.add( Convolution2D( 25,3,3,
            input_shape=(28,28,1)) )
```

How many parameters for each filter? 9

25 x 26 x 26

Max Pooling

```
model2.add(MaxPooling2D((2,2)))
```

25 x 13 x 13

Convolution

```
model2.add(Convolution2D(50,3,3))
```

How many parameters for each filter? 225= 25x9

50 x 11 x 11

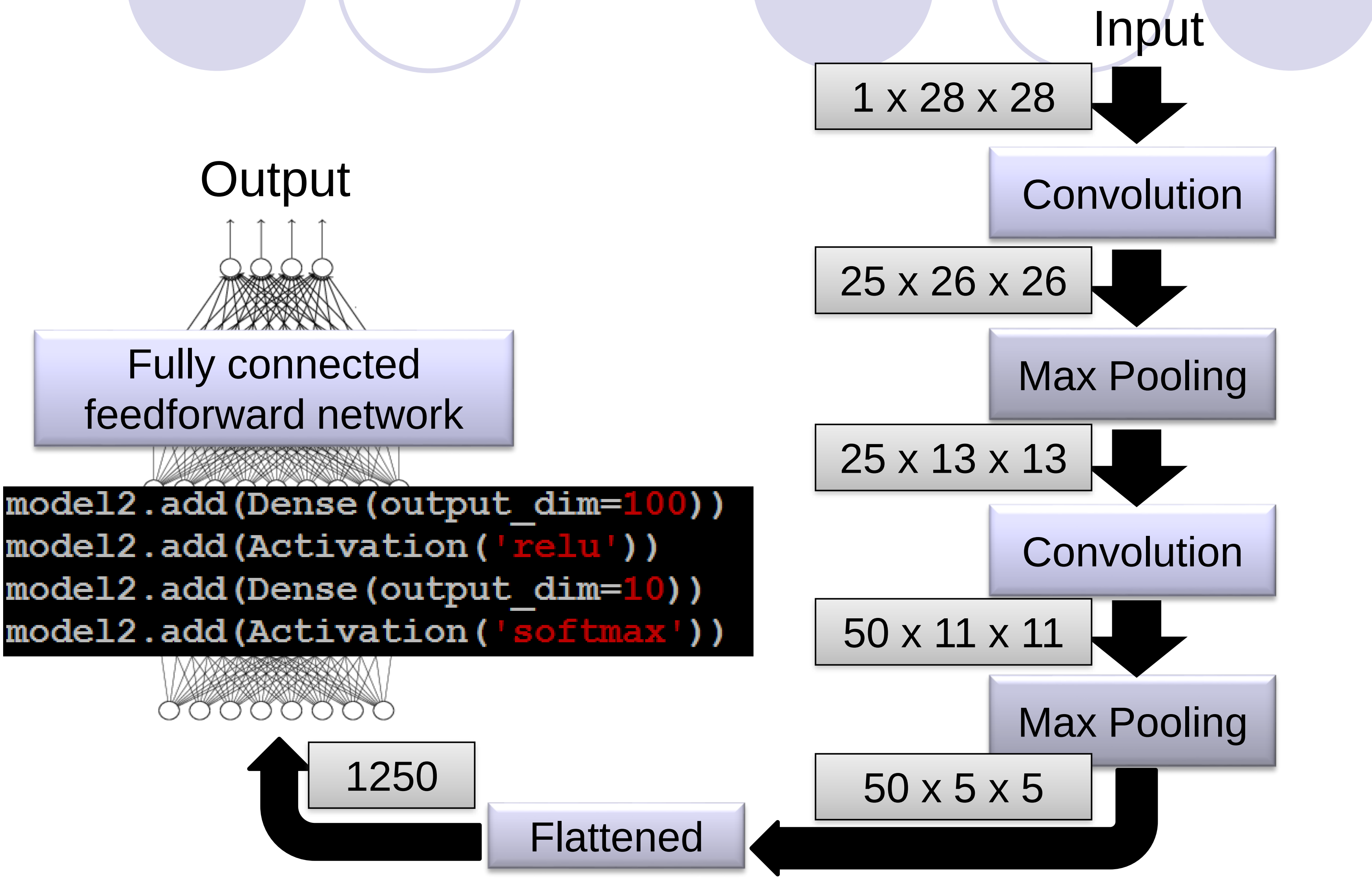
Max Pooling

```
model2.add(MaxPooling2D((2,2)))
```

50 x 5 x 5

CNN in Keras

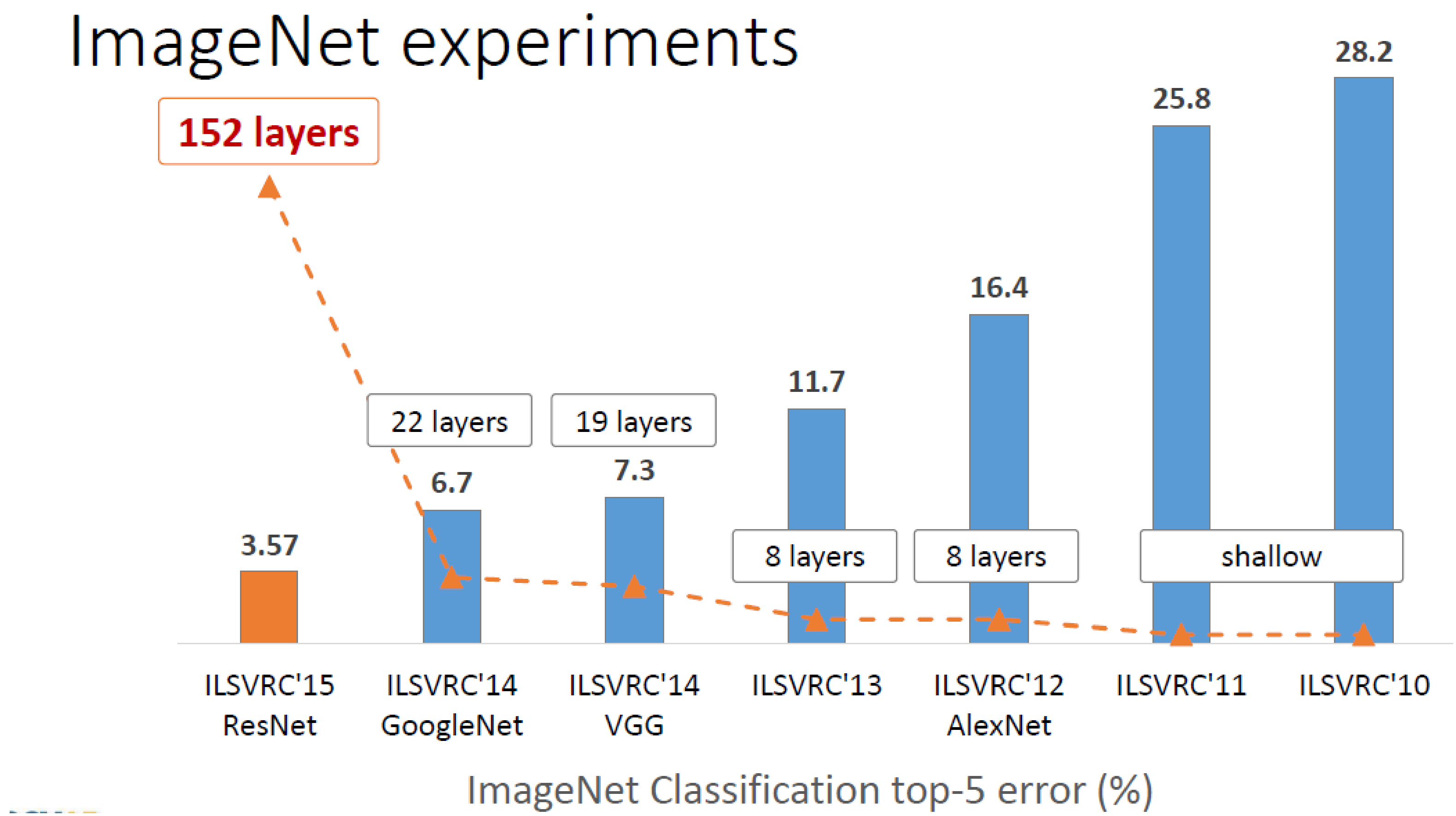
Only modified the ***network structure*** and ***input format (vector -> 3-D array)***




```
model2.add(Dense(output_dim=100))
model2.add(Activation('relu'))
model2.add(Dense(output_dim=10))
model2.add(Activation('softmax'))
```

```
model2.add(Flatten())
```

Data Driven Deep Learning



Deep Residual Nets with 152 layers best on ImageNet Challenge (2015)

Slide credit: Kai-Ming He, Microsoft Research

AlphaGo

19 x 19 matrix

Black: 1

white: -1

none: 0

Neural Network

Next move

(19 x 19 positions)

Fully-connected feedforward network can be used

But CNN performs much better

AlphaGo's policy network

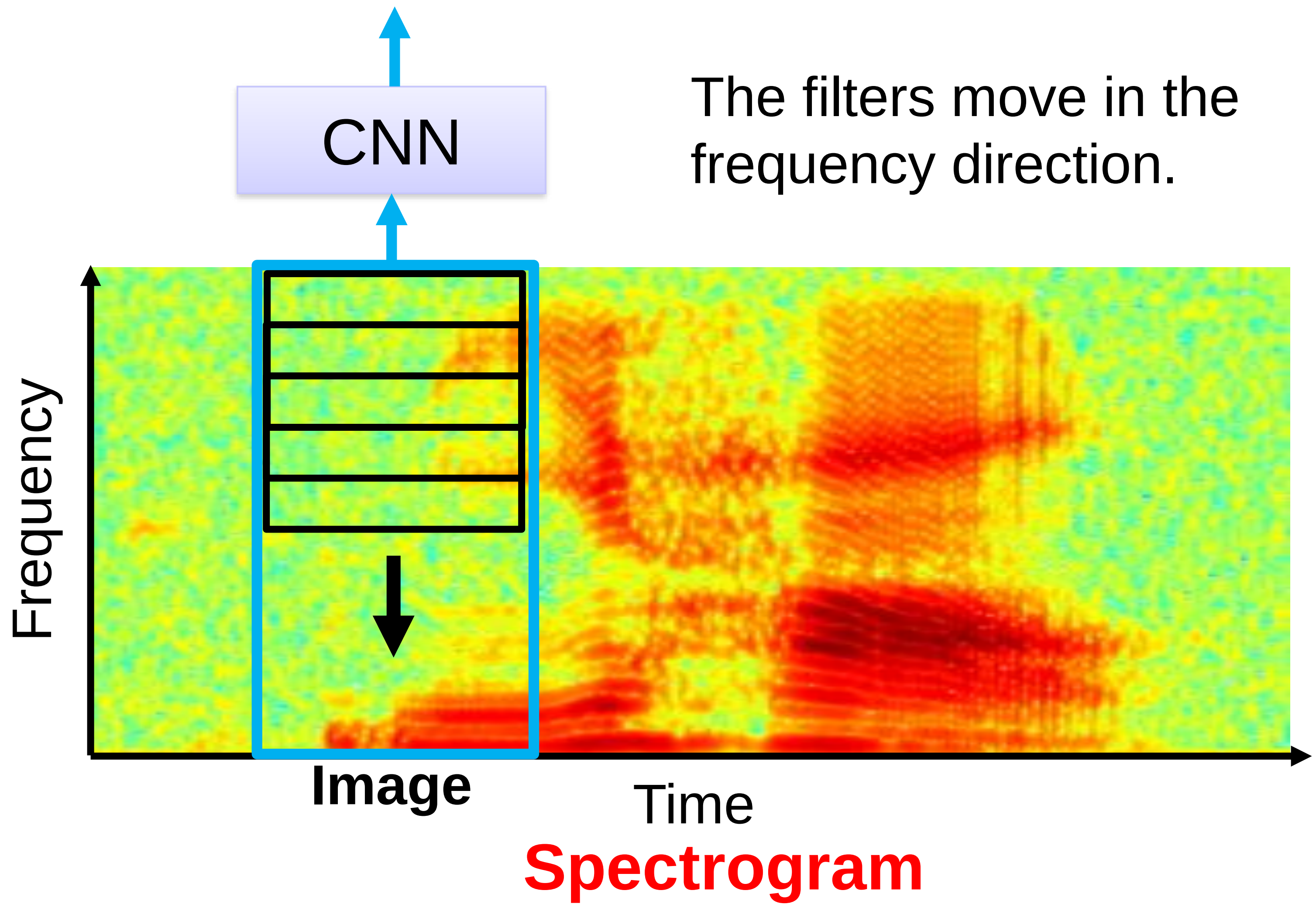
The following is quotation from their Nature article:

Note: AlphaGo does not use Max Pooling.

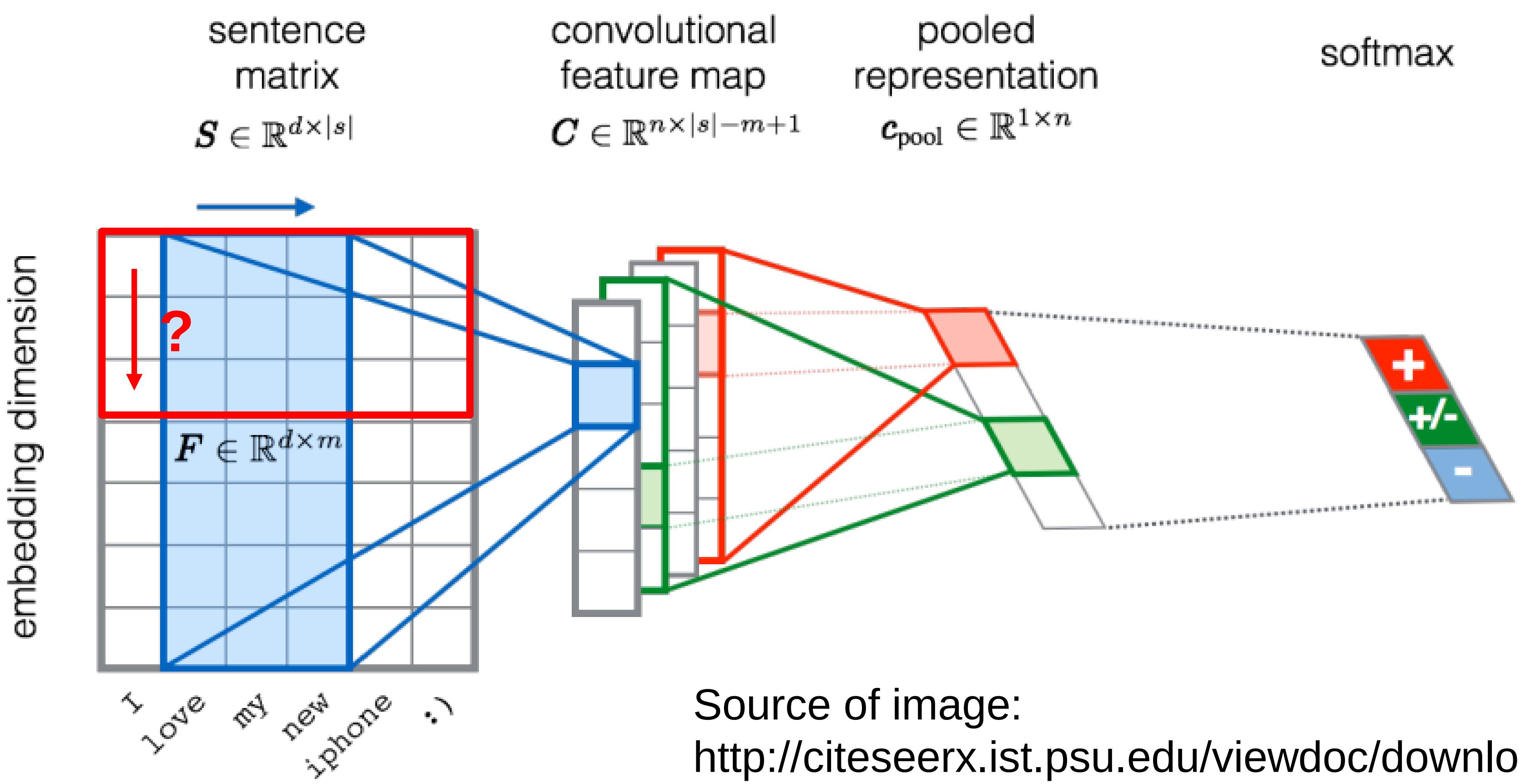
Neural network architecture. The input to the policy network is a $19 \times 19 \times 48$ image stack consisting of 48 feature planes. The first hidden layer zero pads the input into a 23×23 image, then convolves k filters of kernel size 5×5 with stride 1 with the input image and applies a rectifier nonlinearity. Each of the subsequent hidden layers 2 to 12 zero pads the respective previous hidden layer into a 21×21 image, then convolves k filters of kernel size 3×3 with stride 1, again followed by a rectifier nonlinearity. The final layer convolves 1 filter of kernel size 1×1 with stride 1, with a different bias for each position, and applies a softmax function. The match version of AlphaGo used $k = 192$ filters; Fig. 2b and Extended Data Table 3 additionally show the results of training with $k = 128$, 256 and 384 filters.

CNN in speech recognition

The filters move in the frequency direction.



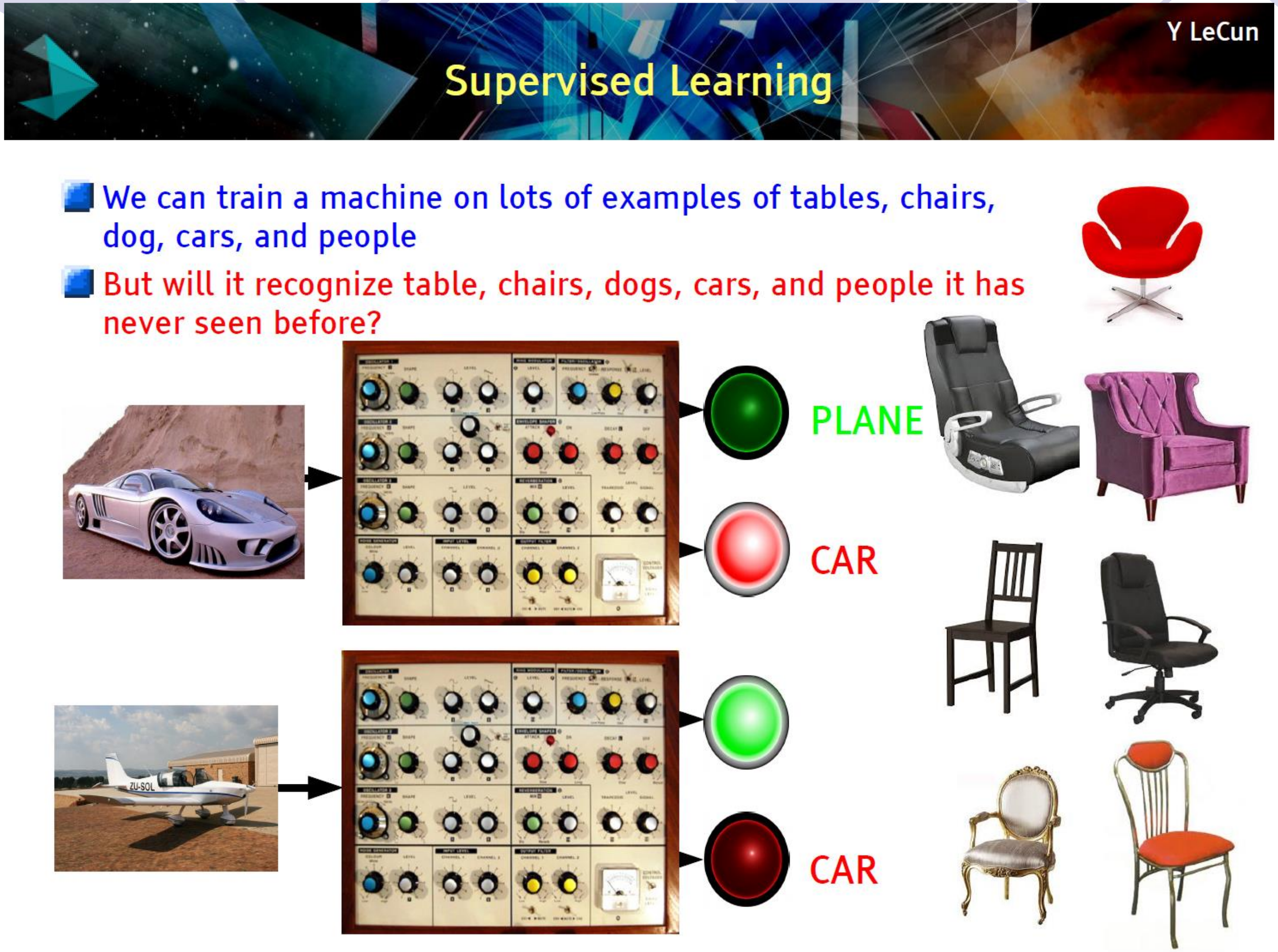
CNN in text classification



Source of image: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.703.6858&rep=rep1&type=pdf

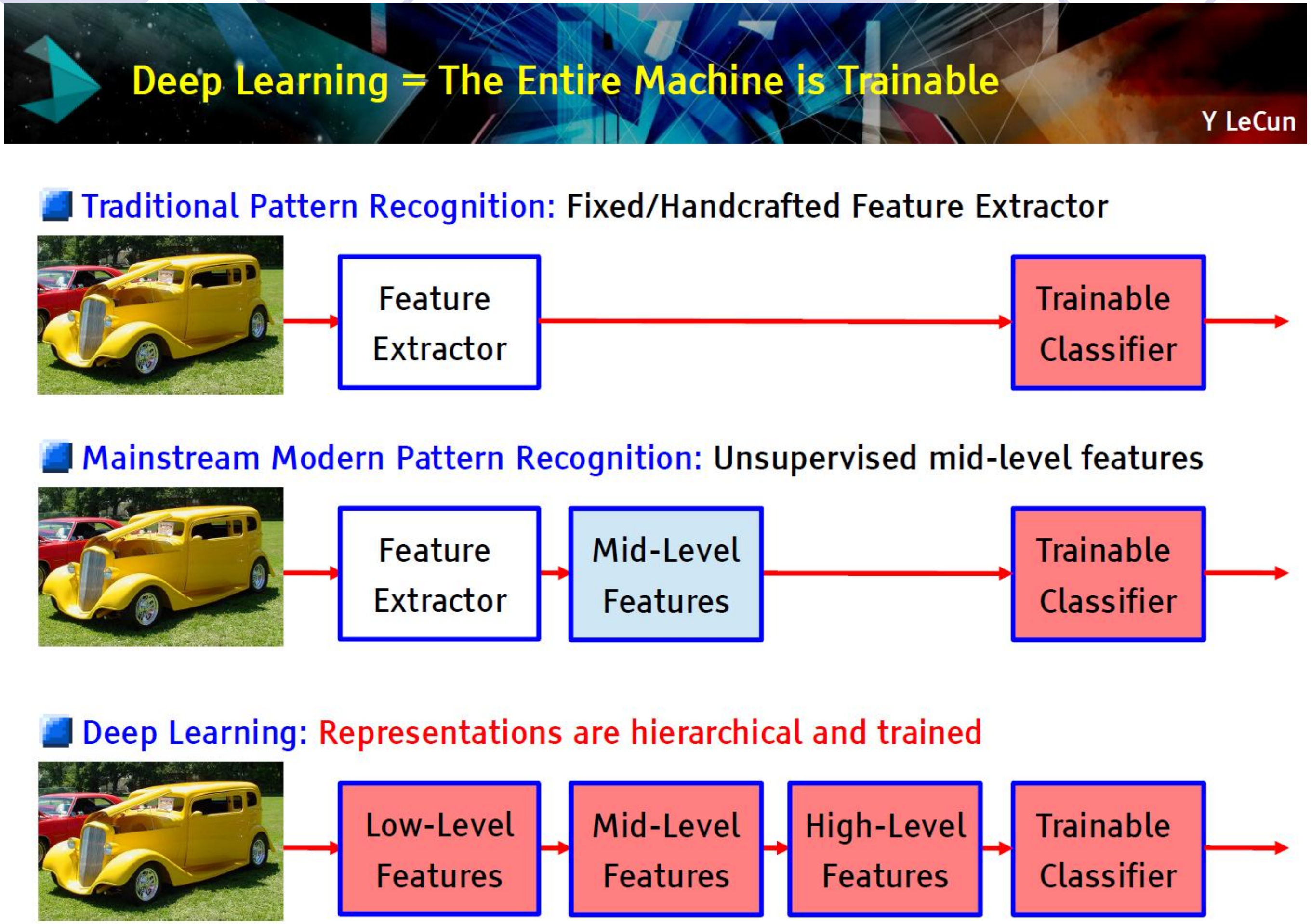
Lecun's viewpoints

Supervised Learning



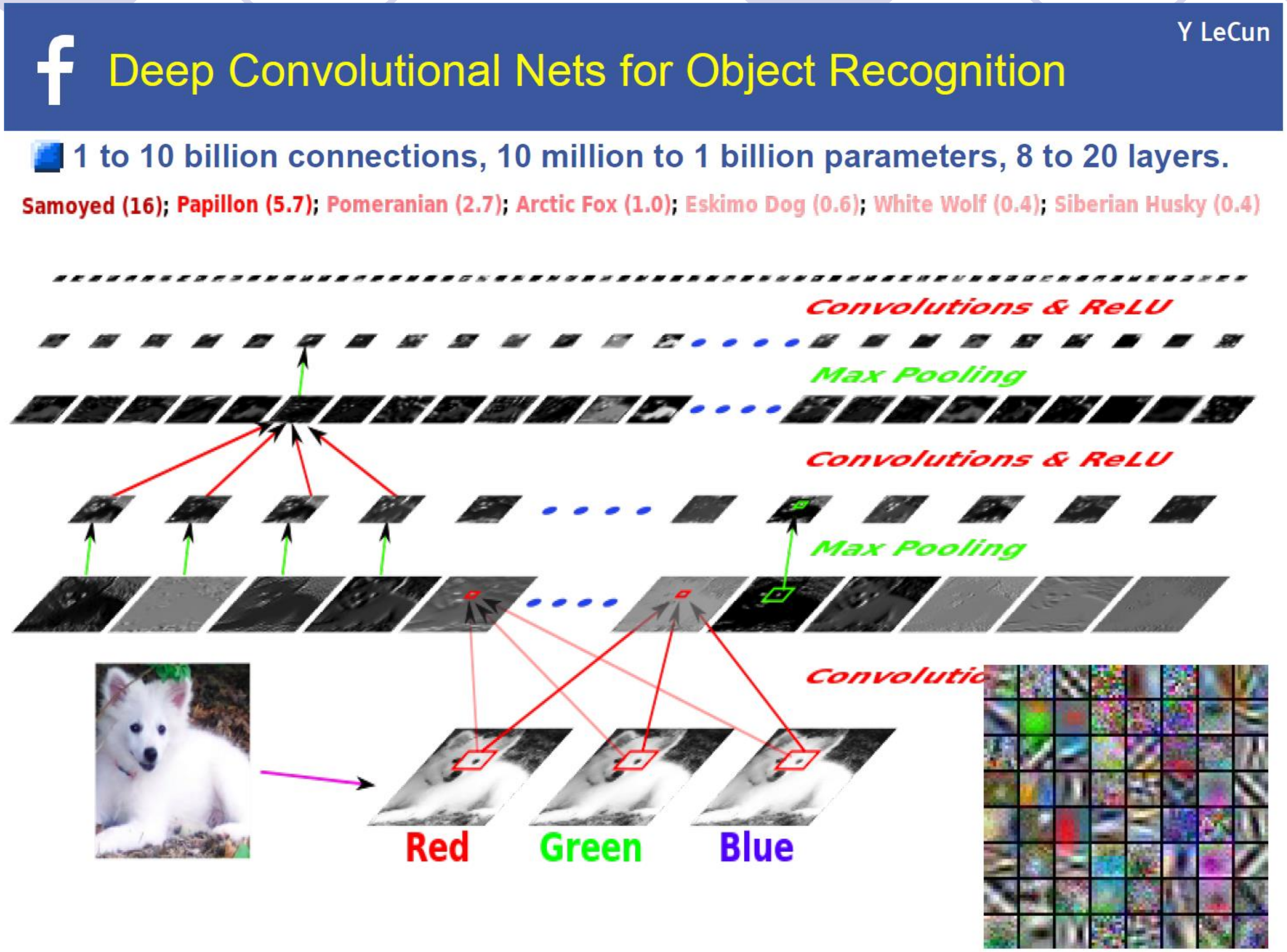
(VR)

Deep Learning



(VR)

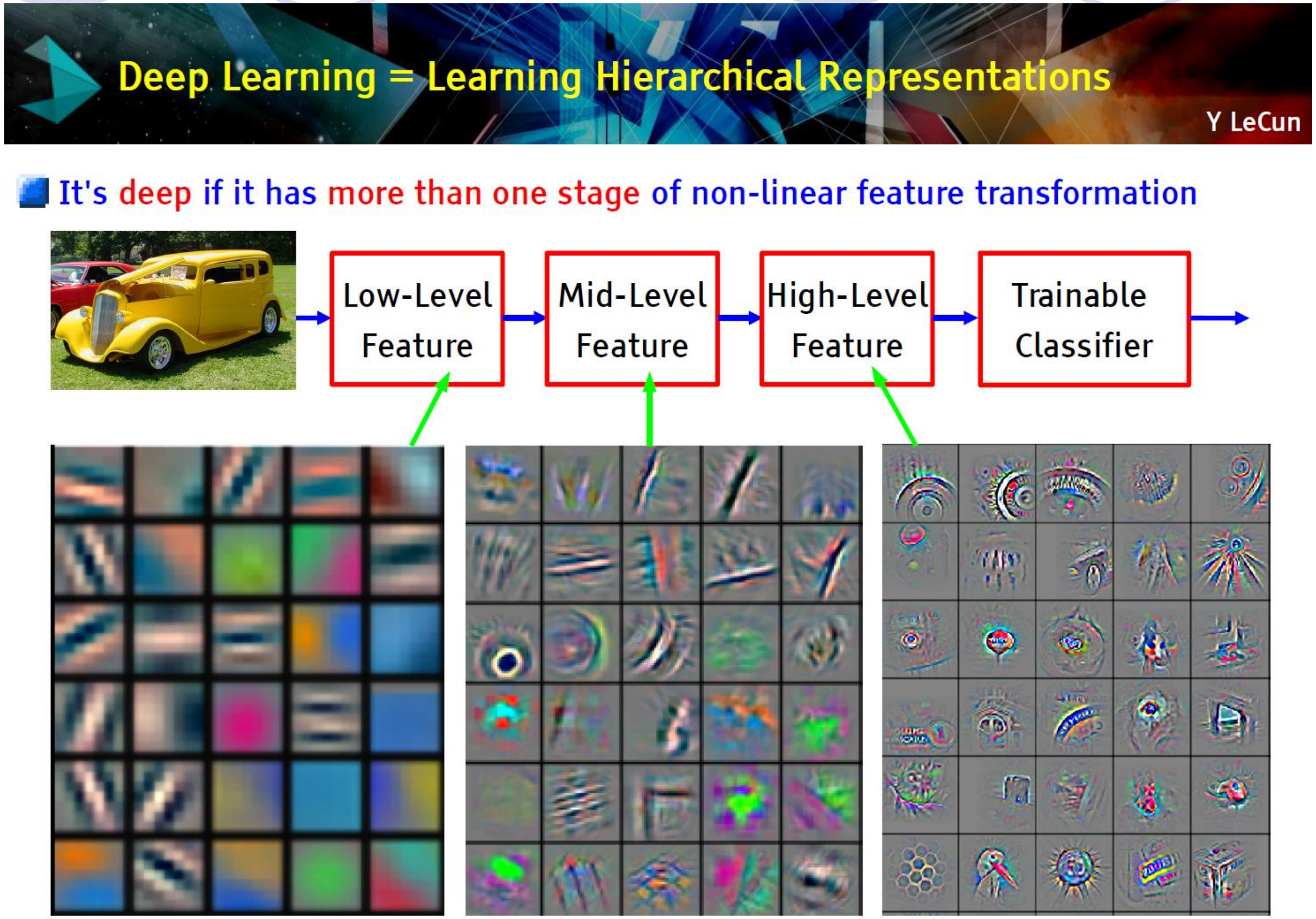
Deep CNN's



Y LeCun
Deep Convolutional Nets for Object Recognition
1 to 10 billion connections, 10 million to 1 billion parameters, 8 to 20 layers.
Samoyed (16); Papillon (5.7); Pomeranian (2.7); Arctic Fox (1.0); Eskimo Dog (0.6); White Wolf (0.4); Siberian Husky (0.4)
Convolutions & ReLU
Max Pooling
Convolutions & ReLU
Max Pooling
Red
Green
Blue

(VR)

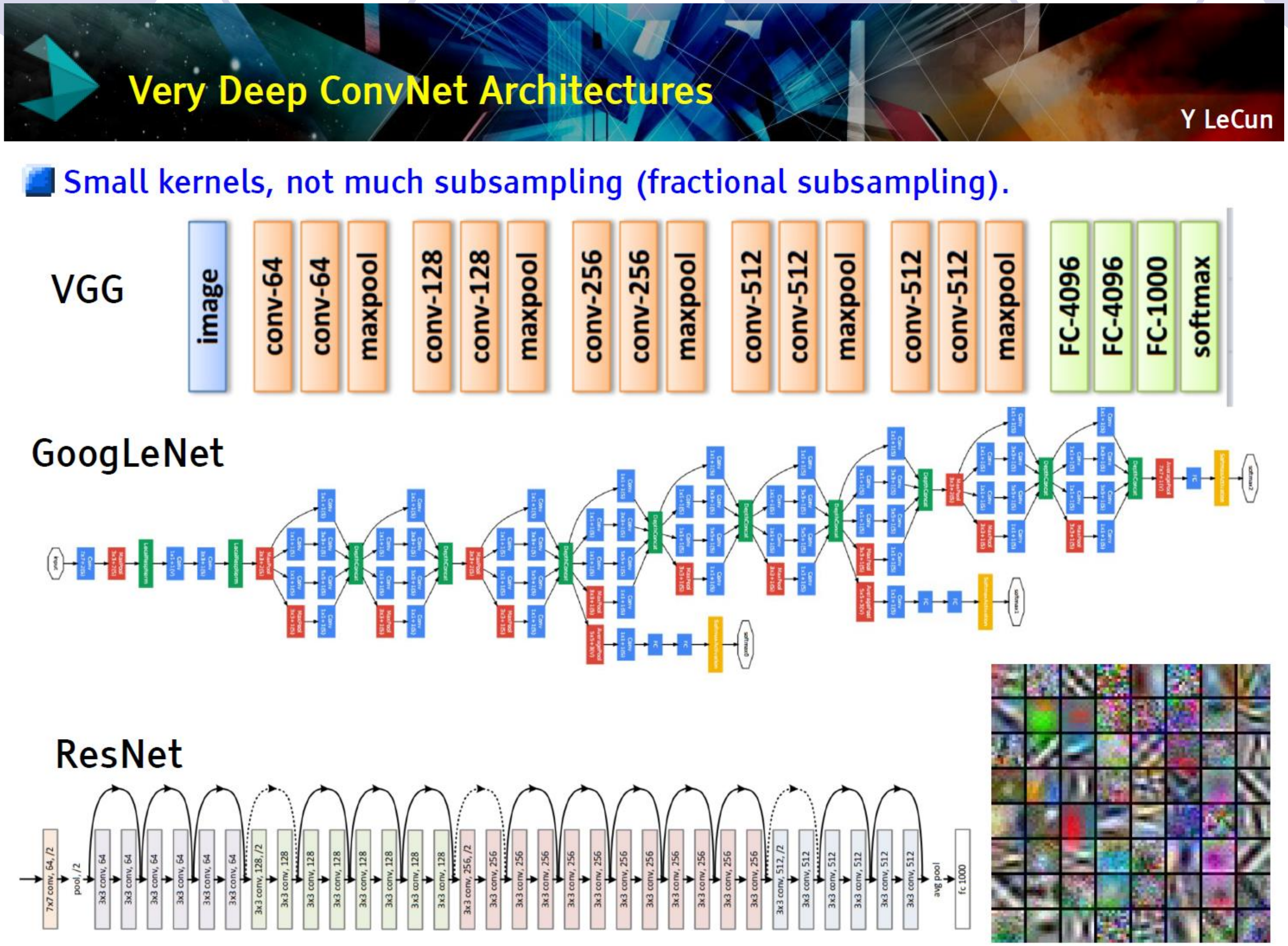
Deep Learning



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

(VR)

State of the art in Deep Learning



Autonomous Driving

(VR)

Obstacles to Progress in AI (Lecun's view)

Obstacles to Progress in AI

Y LeCun

- **Machines need to learn/understand how the world works**
  - Physical world, digital world, people,….
  - They need to acquire some level of common sense
- **They need to learn a very large amount of background knowledge**
  - Through observation and action
- **Machines need to perceive the state of the world**
  - So as to make accurate predictions and planning
- **Machines need to update and remember estimates of the state of the world**
  - Paying attention to important events. Remember relevant events
- **Machines neet to reason and plan**
  - Predict which sequence of actions will lead to a desired state of the world

- **Intelligence & Common Sense =**

  Perception + Predictive Model + Memory + Reasoning & Planning

Common Sense Knowledge

What is Common Sense?

Y LeCun

- "The trophy doesn't fit in the suitcase because it's too large/small"
  - (winograd schema)

- "Tom picked up his bag and left the room"

- We have common sense because we know how the world works
- How do we get machines to learn that?

(VR)

Common Sense

Common Sense is the ability to fill in the blanks

Y LeCun

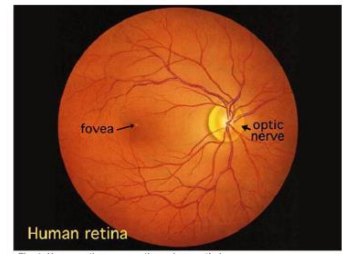


Fig. 1. Human retina as seen through an opthalmoscope.

- Infer the state of the world from partial information
- Infer the future from the past and present
- Infer past events from the present state

- Filling in the visual field at the retinal blind spot
- Filling in occluded images
- Fillling in missing segments in text, missing words in speech.
- Predicting the consequences of our actions
- Predicting the sequence of actions leading to a result

- Predicting any part of the past, present or future percepts from whatever information is available.

- That's what predictive learning is
- But really, that's what many people mean by unsupervised learning

Unsupervised/Predictive Learning

The Necessity of Unsupervised Learning / Predictive Learning

Y LeCun

- The number of samples required to train a large learning machine (for any task) depends on the amount of information that we ask it to predict.
  - The more you ask of the machine, the larger it can be.
- "The brain has about 10^14 synapses and we only live for about 10^9 seconds. So we have a lot more parameters than data. This motivates the idea that we must do a lot of unsupervised learning since the perceptual input (including proprioception) is the only place we can get 10^5 dimensions of constraint per second."
  - Geoffrey Hinton (in his 2014 AMA on Reddit)
  - (but he has been saying that since the late 1970s)
- Predicting human-provided labels is not enough
- Predicting a value function is not enough

(VR)

Predictive Learning

How Much Information Does the Machine Need to Predict?

Y LeCun

- "Pure" Reinforcement Learning (cherry)
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- Supervised Learning (icing)
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- Unsupervised/Predictive Learning (cake)
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

- (Yes, I know, this picture is slightly offensive to RL folks. But I'll make it up)

(VR)

Reinforcement Learning

Sutton's Dyna Architecture: "try things in your head before acting"

Y LeCun

- Dyna: an Integrated Architecture for Learning, Planning and Reacting
  - [Rich Sutton, ACM SIGART 1991]

The main idea of Dyna is the old, commonsense idea that planning is 'trying things in your head,' using an internal model of the world (Craik, 1943; Dennett, 1978; Sutton & Barto, 1981). This suggests the existence of a more primitive process for trying things *not* in your head, but through direct interaction with the world. *Reinforcement learning* is the name we use for this more primitive, direct kind of trying, and Dyna is the extension of reinforcement learning to include a learned world model.

REPEAT FOREVER:

1. Observe the world's state and reactively choose an action based on it;

2. Observe resultant reward and new state;

3. Apply reinforcement learning to this experience;

4. Update action model based on this experience;

5. Repeat K times:

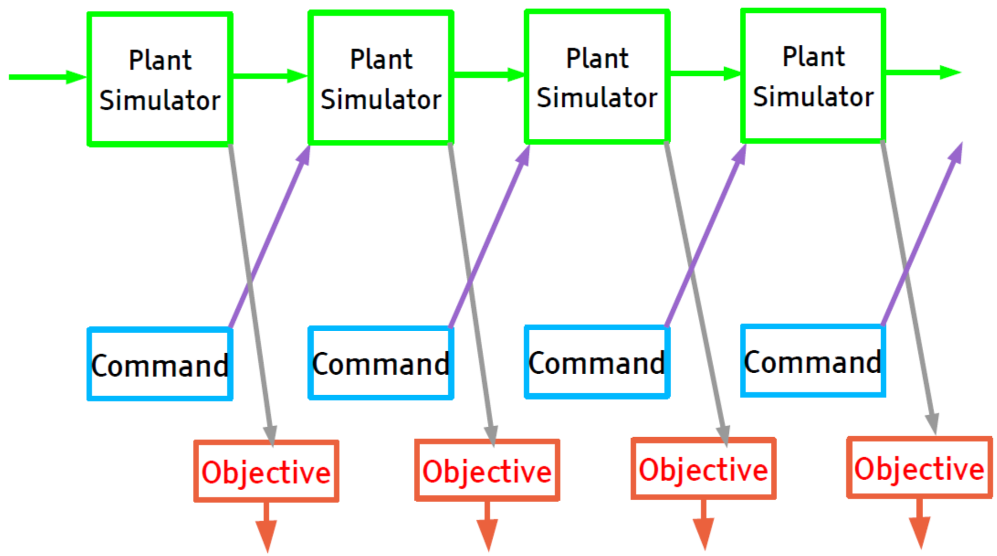
- **5.1** Choose a hypothetical world state and action;
- **5.2** Predict resultant reward and new state using action model;
- **5.3** Apply reinforcement learning to this hypothetical experience.

Classical Model-based Optimal Control

Y LeCun

Classical model-based optimal control

- **Simulate the world (the plant) with an initial control sequence**
- **Adjust the control sequence to optimize the objective through gradient descent**
- **Backprop through time was invented by control theorists in the late 1950s**
  - it's sometimes called the adjoint state method
  - [Athans & Falb 1966, Bryson & Ho 1969]



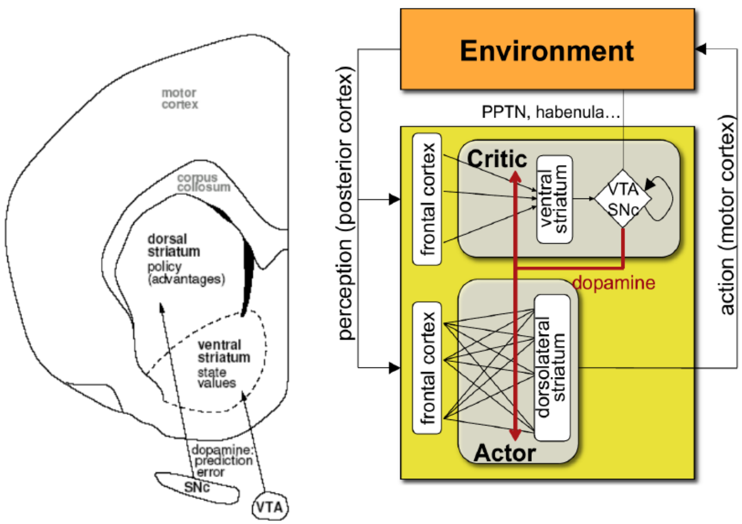
(VR)

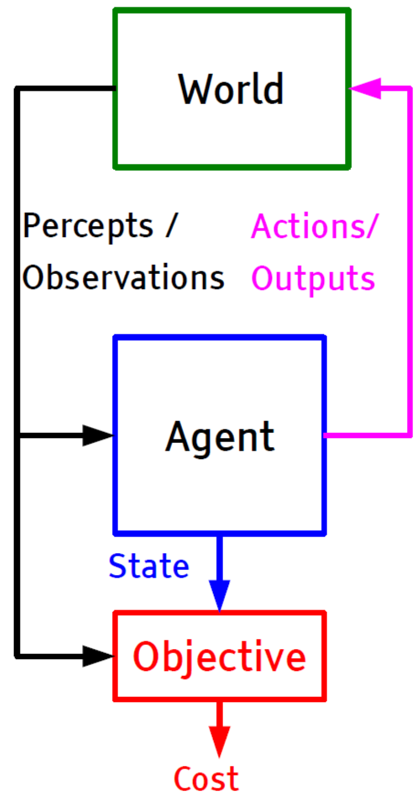
AI system

Y LeCun

AI System: Learning Agent + Immutable Objective

- The agent gets percepts from the world
- The agent acts on the world
- The agents tries to minimize the long-term expected cost.





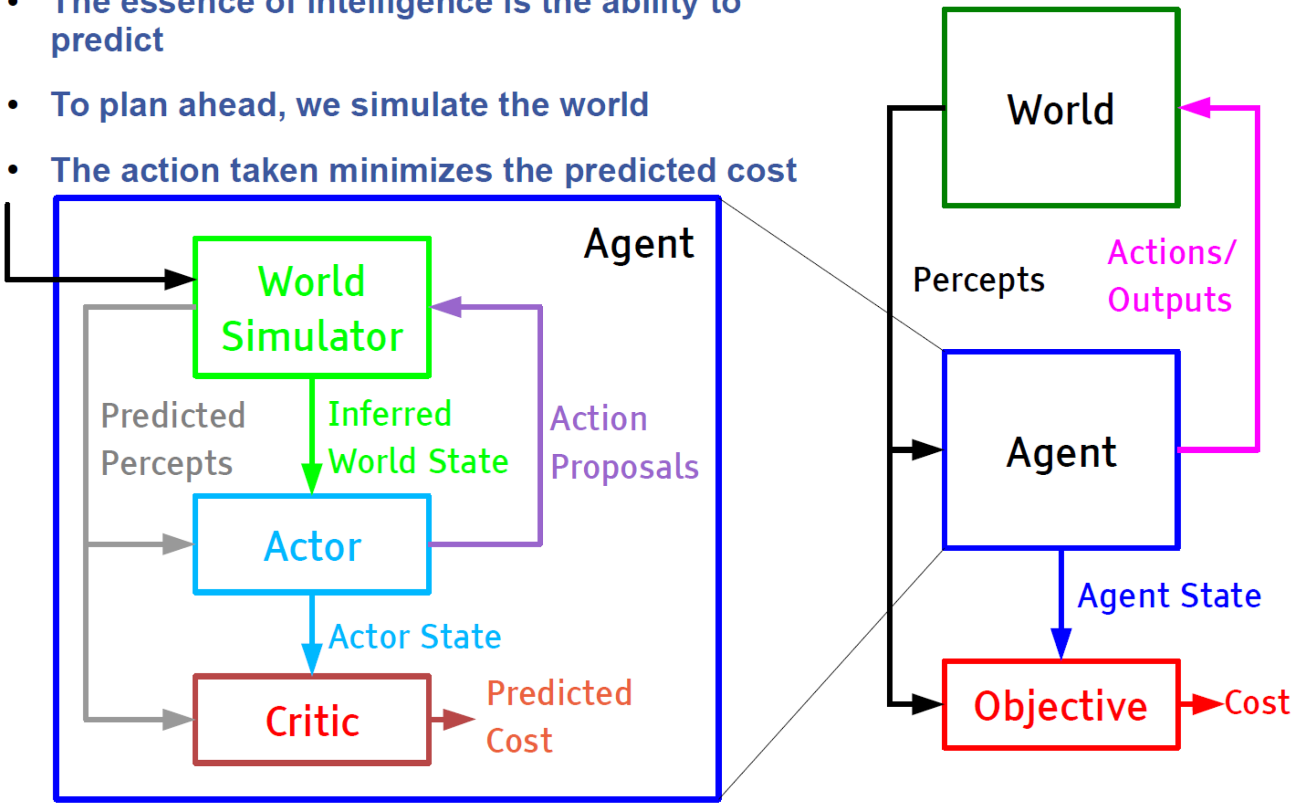
(VR)

Predicting + Planning = Reasoning

Y LeCun

AI System: Predicting + Planning = Reasoning

- The essence of intelligence is the ability to predict
- To plan ahead, we simulate the world
- The action taken minimizes the predicted cost



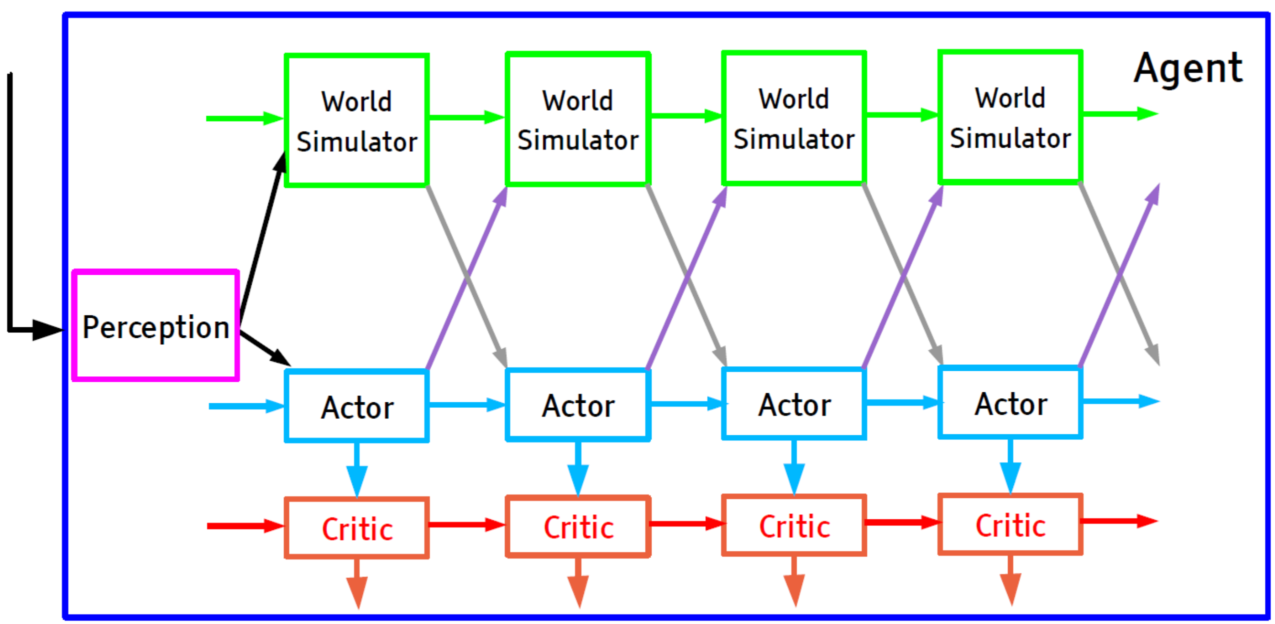
(VR)

Model-based Reinforcement Learning

What we need is Model-Based Reinforcement Learning

Y LeCun

- The essence of intelligence is the ability to predict
- To plan ahead, we must simulate the world, so as to minimizes the predicted value of some objective function.



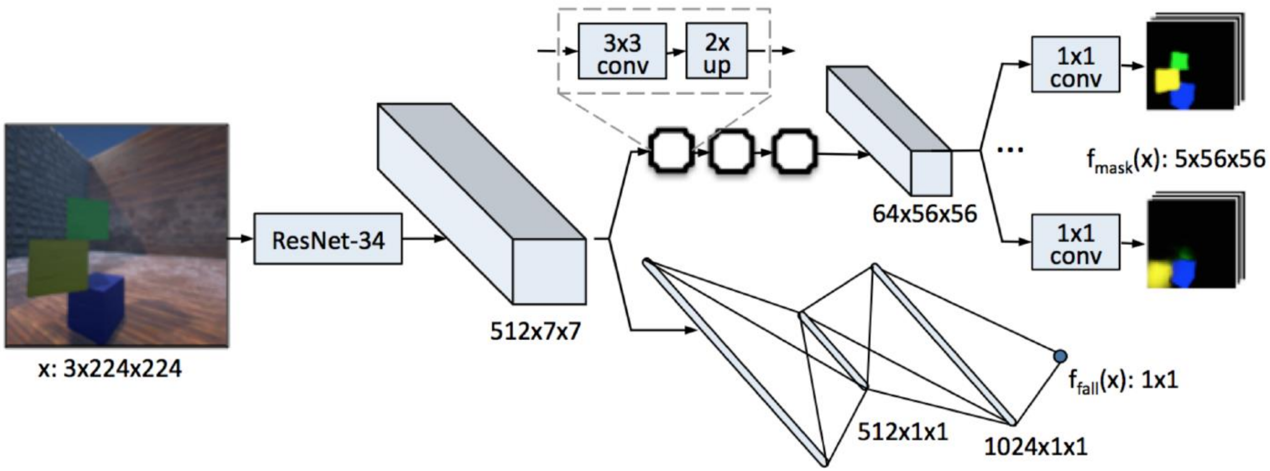
(VR)

Example

Learning Physics (PhysNet)

Y LeCun

- [Lerer, Gross, Fergus arxiv:1603.01312]
  - ConvNet produces object masks that predict the trajectories of falling blocks
  - Uses the Unreal game engine.



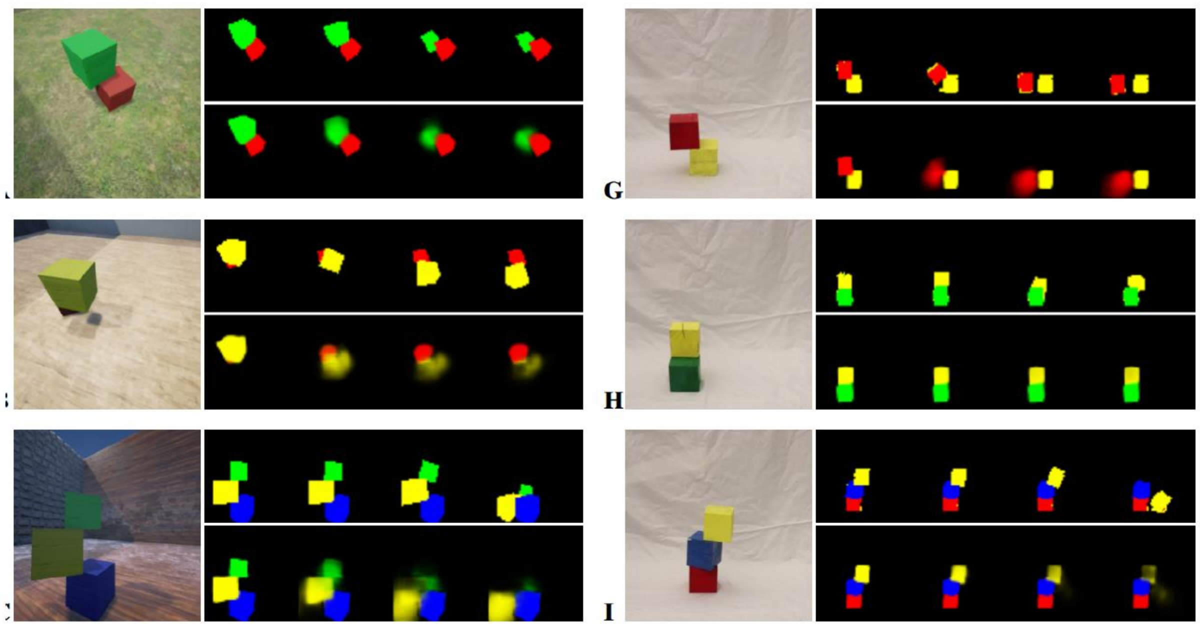
(VR)

Learning Physics

Learning Physics (PhysNet)

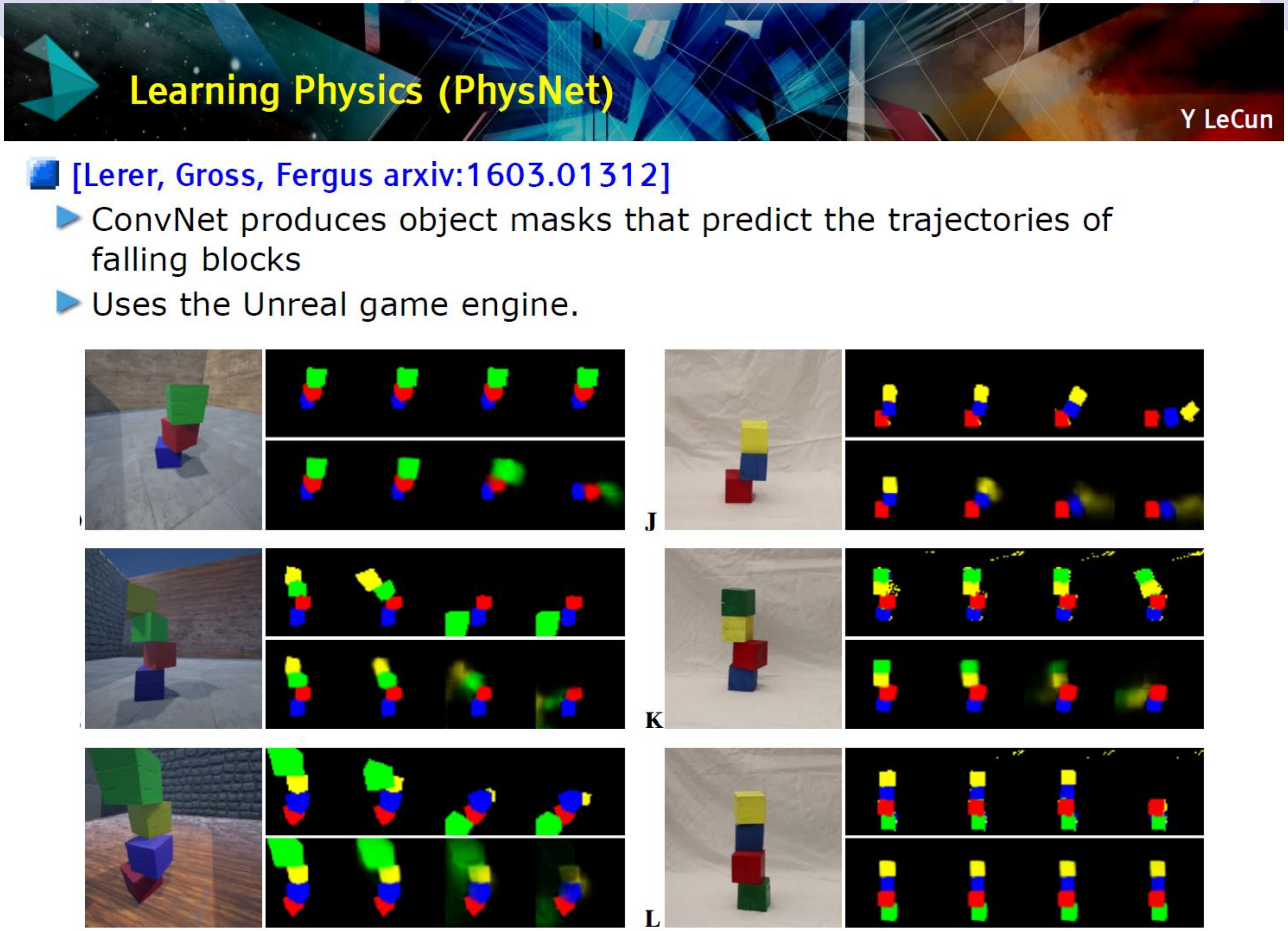
Y LeCun

- [Lerer, Gross, Fergus arxiv:1603.01312]
  - ConvNet produces object masks that predict the trajectories of falling blocks
  - Uses the Unreal game engine.



(VR)

Learning Physics



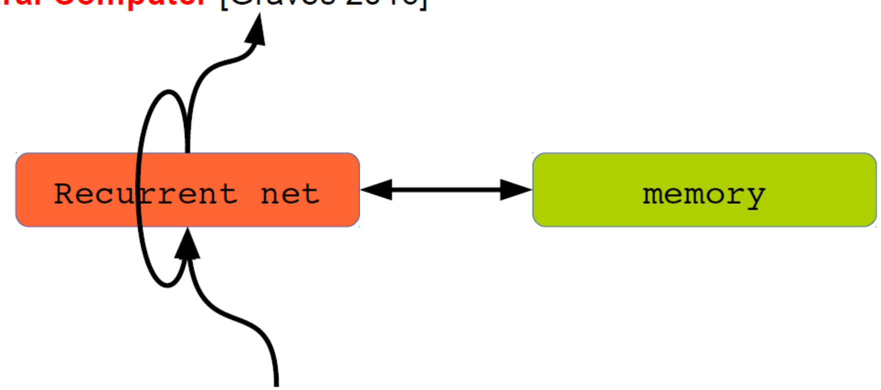
(VR)

Augmenting Neural Nets with Memory

Y LeCun

Augmenting Neural Nets with a Memory Module

- **Recurrent networks cannot remember things for very long**
  - The cortex only remember things for 20 seconds
- **We need a "hippocampus" (a separate memory module)**
  - LSTM [Hochreiter 1997], registers
  - **Memory networks** [Weston et 2014] (FAIR), associative memory
  - **Stacked-Augmented Recurrent Neural Net** [Joulin & Mikolov 2014] (FAIR)
  - **Neural Turing Machine** [Graves 2014],
  - **Differentiable Neural Computer** [Graves 2016]



(VR)

Link between CNN's and Model-based Network Designs

- Bayesian Model Based Vision (Binford)
- Systems Analysis of Deep Chains (Ramesh, various)
- Scattering Transform (Mallat, 2011)
- Modern perspectives – Patel & Baranuik (2015), others.

Radford Neal (90's)

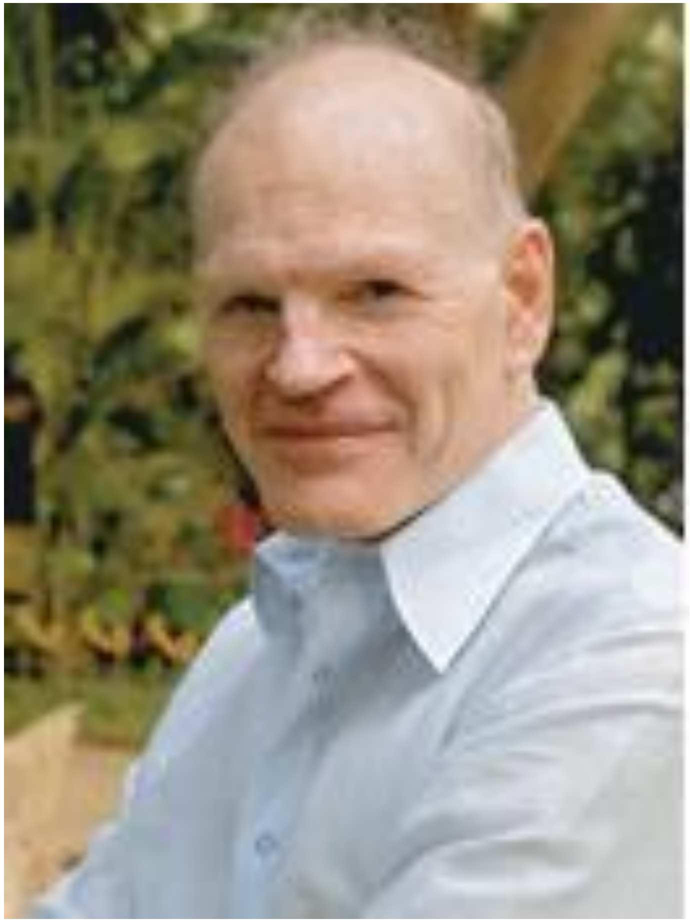
Infinite neural networks

Neural networks (one hidden layer) with random weights converge to a Gaussian process:

$$
\begin{aligned}
\mathbb{E}[f(x)] &= \mathbb{E}[\sum_{i=1}^{H} w_i^{(2)} h_i(x) + w_0^{(2)}] \\
&= \underbrace{\mathbb{E}[w_0^{(2)}]}_{=0} + \sum_{i=1}^{H} \underbrace{\mathbb{E}[w_i^{(2)}]}_{=0} \mathbb{E}[h_i(x)] = 0 \\
\mathbb{E}[f(x)f(x')] &= \mathbb{E}[\left(\sum_{i=1}^{H} w_i^{(2)} h_i(x) + w_0^{(2)}\right)\left(\sum_{i=1}^{H} w_i^{(2)} h_i(x') + w_0^{(2)}\right)] \\
&= \sum_{i=1}^{H} \mathbb{E}[(w_i^{(2)})^2]\mathbb{E}[h_i(x)h_i(x')] + \mathbb{E}[(w_0^{(2)})^2] \\
&= \frac{\sigma_{21}^2}{H} \sum_{i=1}^{H} \mathbb{E}[h_i(x)h_i(x')] + \sigma_{20}^2 \\
&= \sigma_{21}^2 \mathbb{E}[h(x)h(x')] + \sigma_{20}^2
\end{aligned}
$$

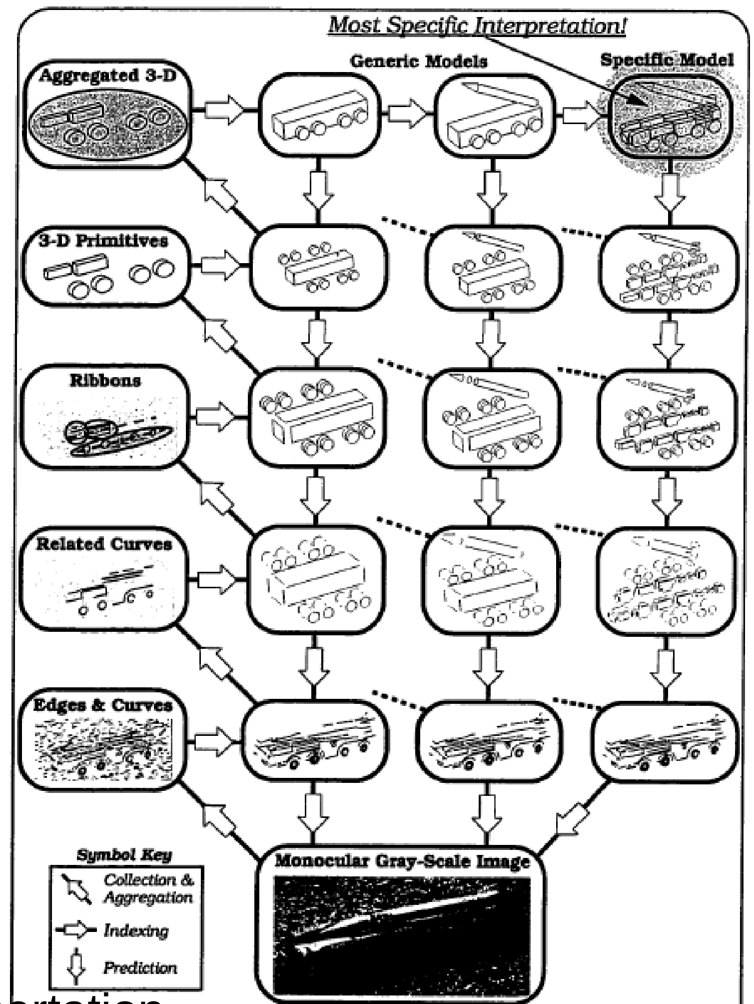
Bayesian Networks for Model-based Vision: Mann, Binford (1990's)

- **Early use of Hierarchical Bayesian Network representations for model-based recognition**
- **Illustration of 'quasi-invariant based indexing' followed by extrapolation (prediction) and verification**



Bayesian Networks in Vision (Mann, 1996)

- **Automated and dynamic generation of Bayesian networks**
- **Early Illustration of how to derive meaningful probabilities for Bayesian Networks**
- **Addressed problem of Articulated Model recognition in a given image using Bayesian networks**



*Source: W. Mann (1996), Stanford U., Phd. Dissertation

Interpretation Cycle: (Mann, 1996)

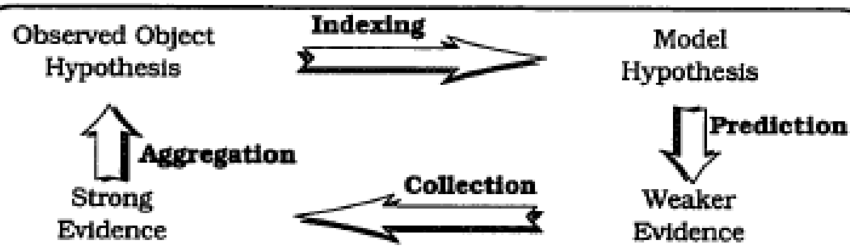


Figure 2-3: Fundamental Cycle of Interpretation

This is the basic cycle of the algorithm in this work.

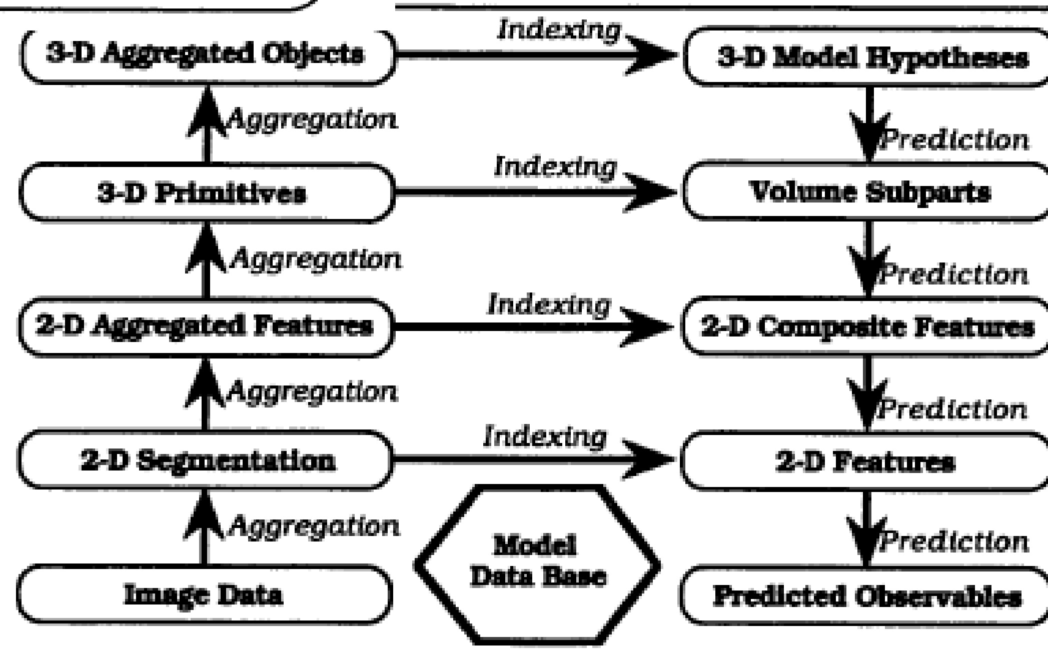
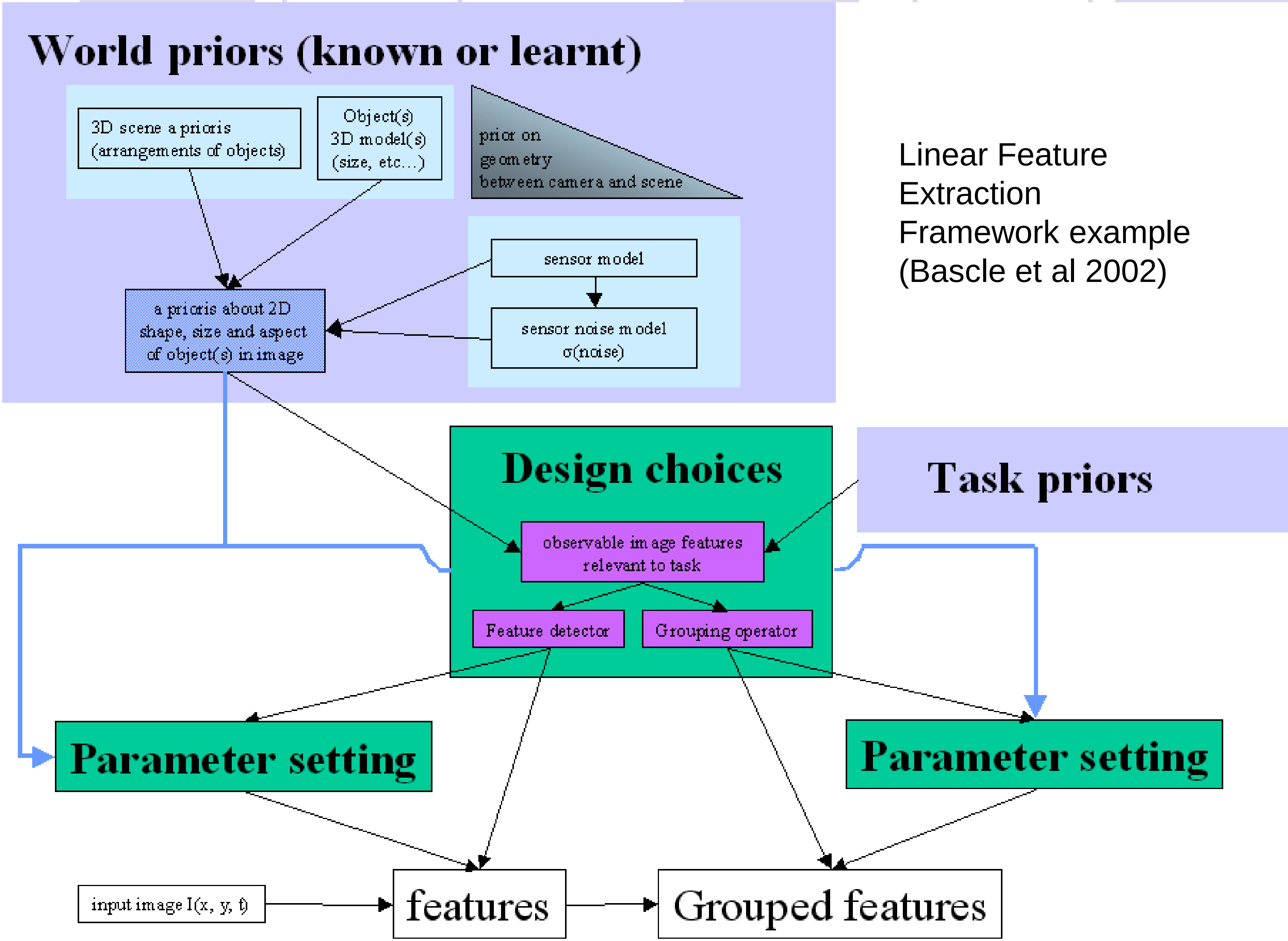


Figure 2-4: Interpretation in Successor

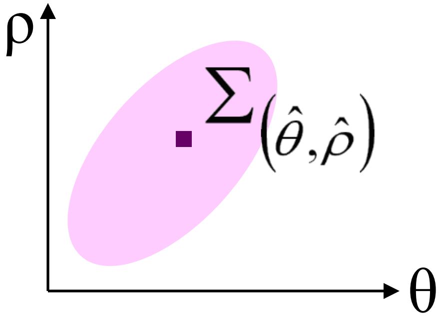
*Source: W. Mann (1996), Stanford U., Phd. Dissertation

Linear Feature Extraction Framework example (Bascle et al 2002)



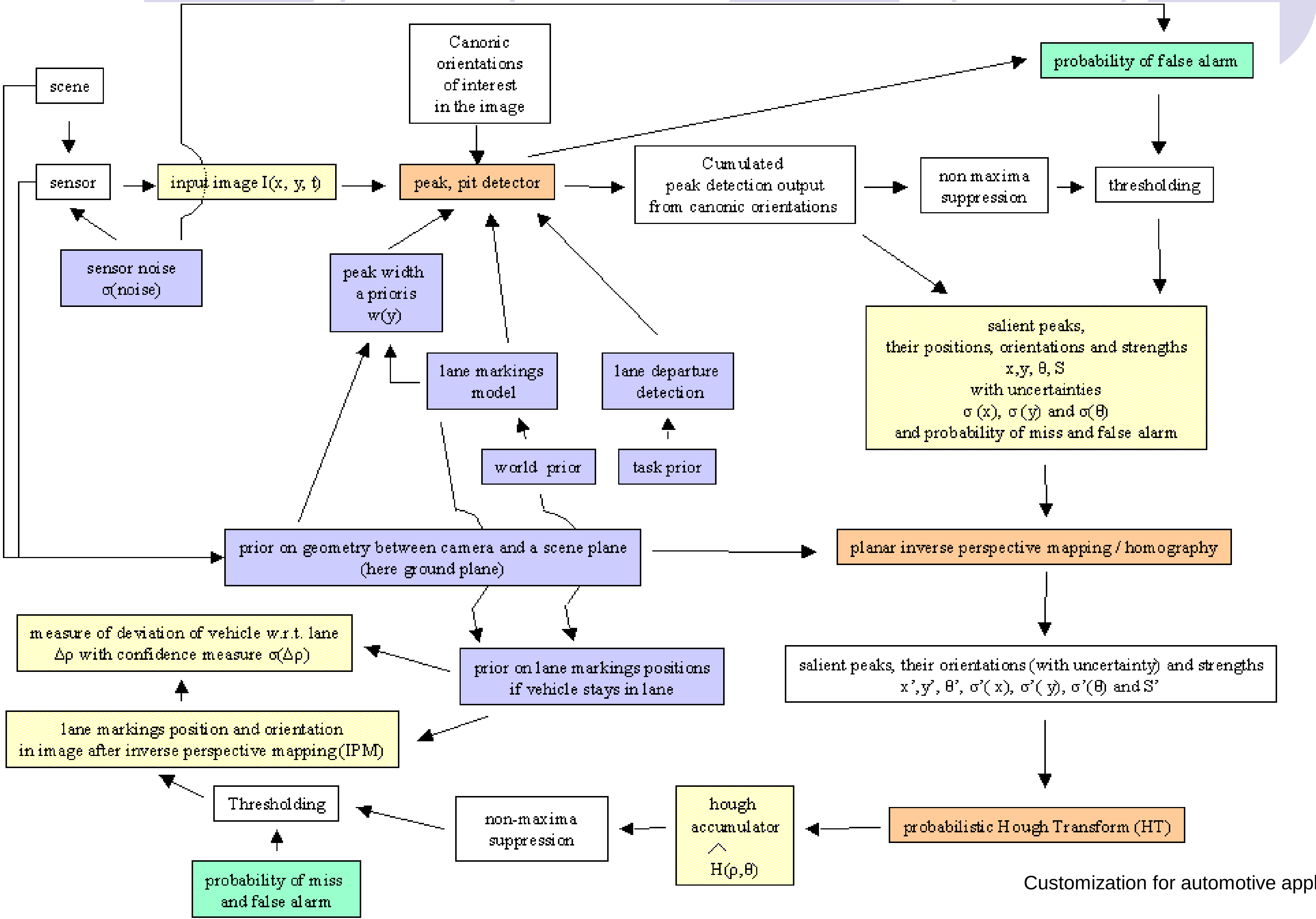
Lane Detection via Hough Transform

- Priors on position and orientation
- Wide line features
- Covariance propagation
- Automatic thresholding
- Fusion of line hypotheses or Variable Bandwidth Mean-Shift



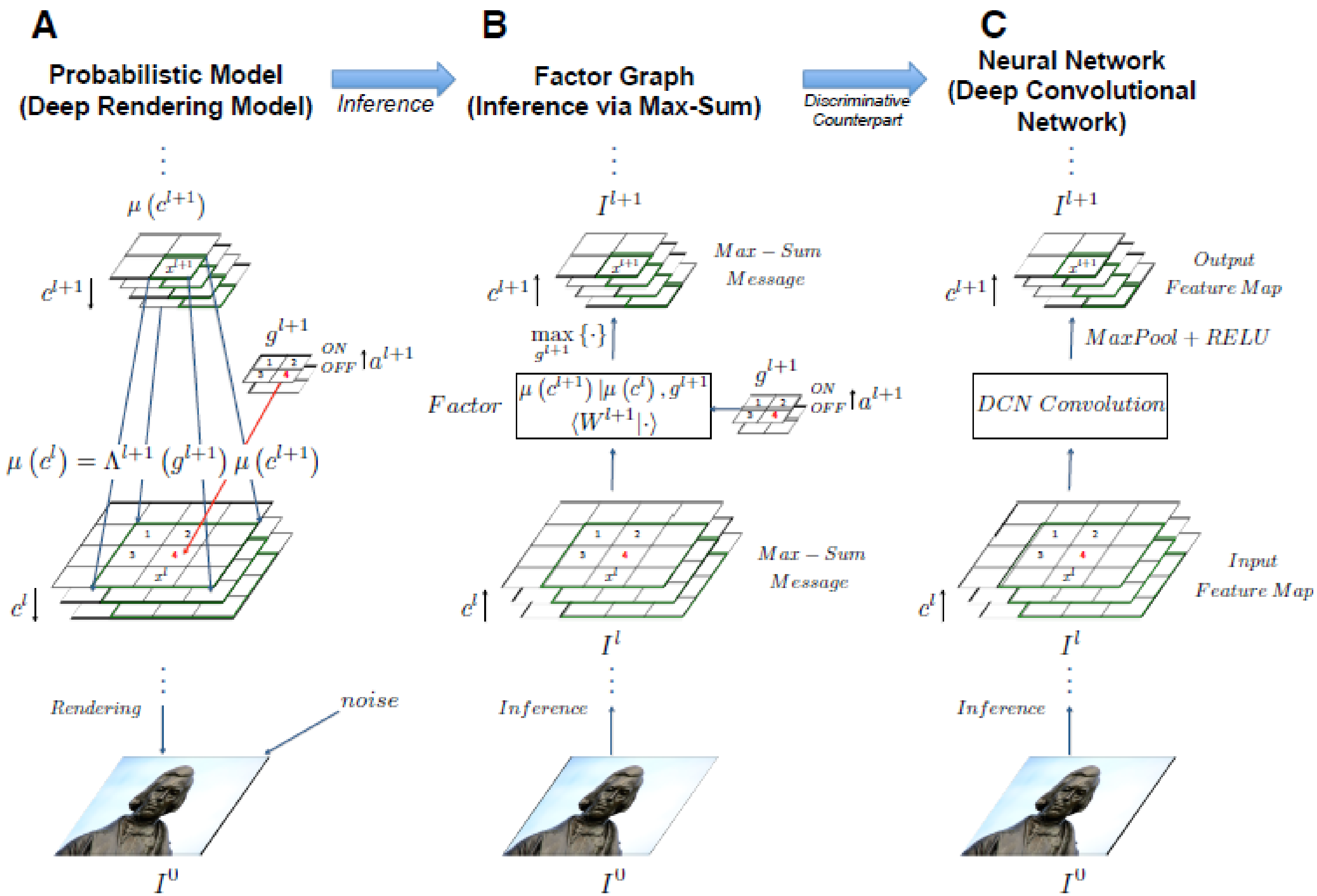
Published in SMVP'04

Systems Analysis:



Customization for automotive application

Deep Rendering Model (Patel et al, 2015)



Probabilistic Theory of Deep Learning (Patel et al, 2015)

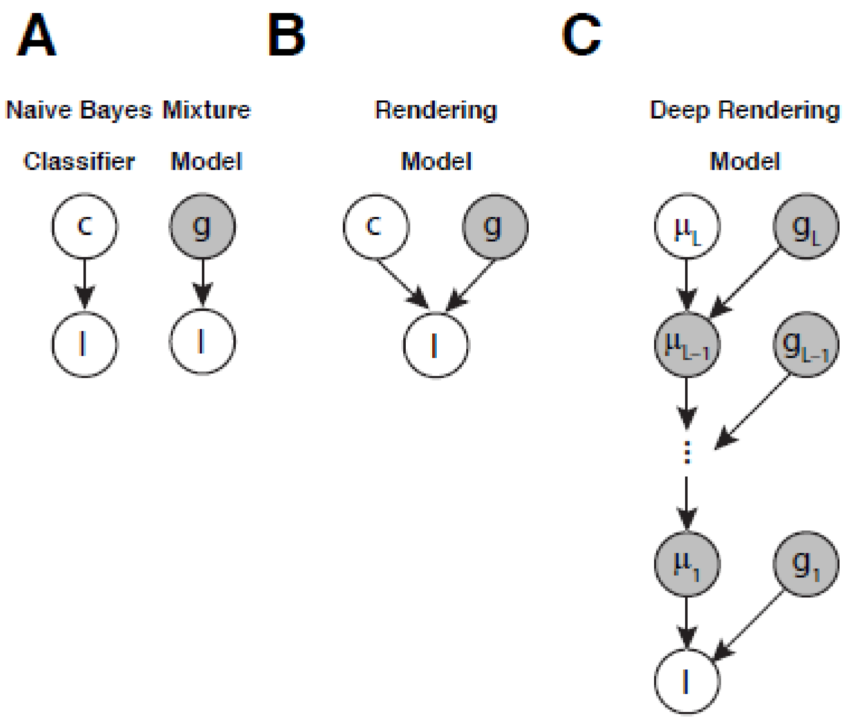


Figure 1. Graphical depiction of the Naive Bayes Classifier (A, left), Gaussian Mixture Model (A, right), the shallow Rendering Model (B) and the Deep Rendering Model (C). All dependence on pixel location x has been suppressed for clarity.

Illustration of DRM

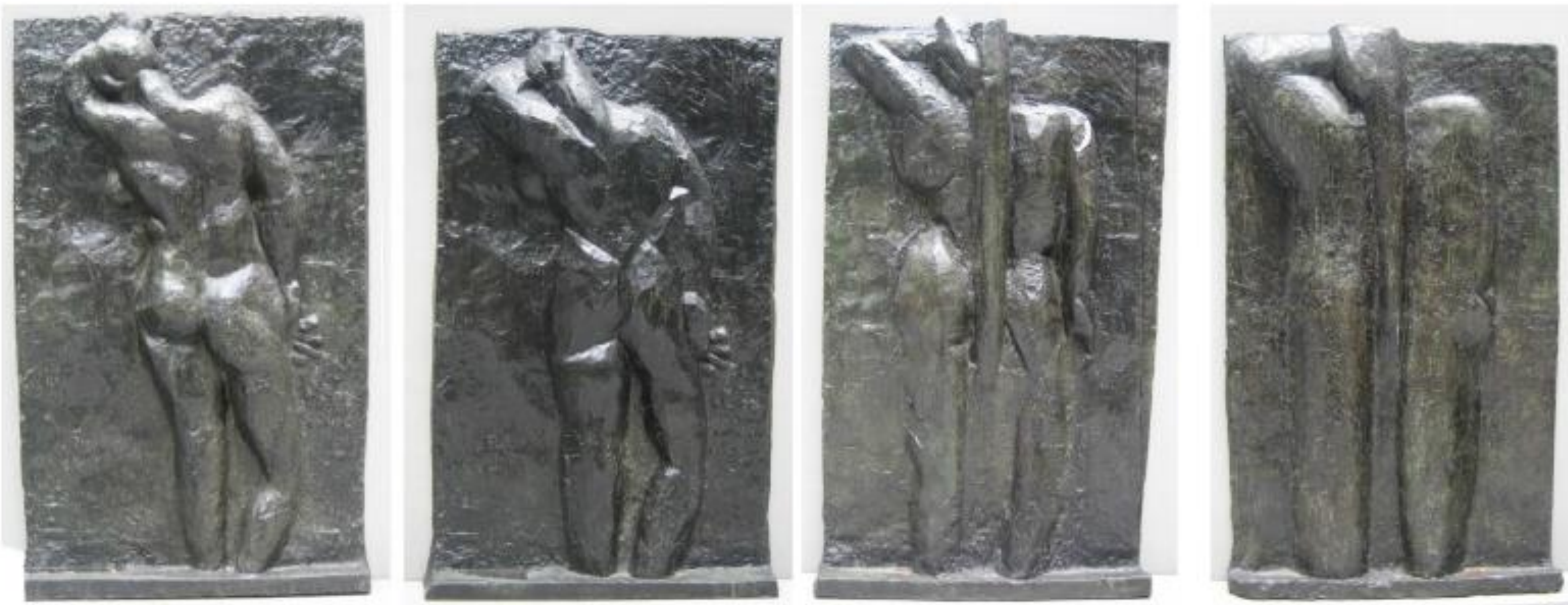


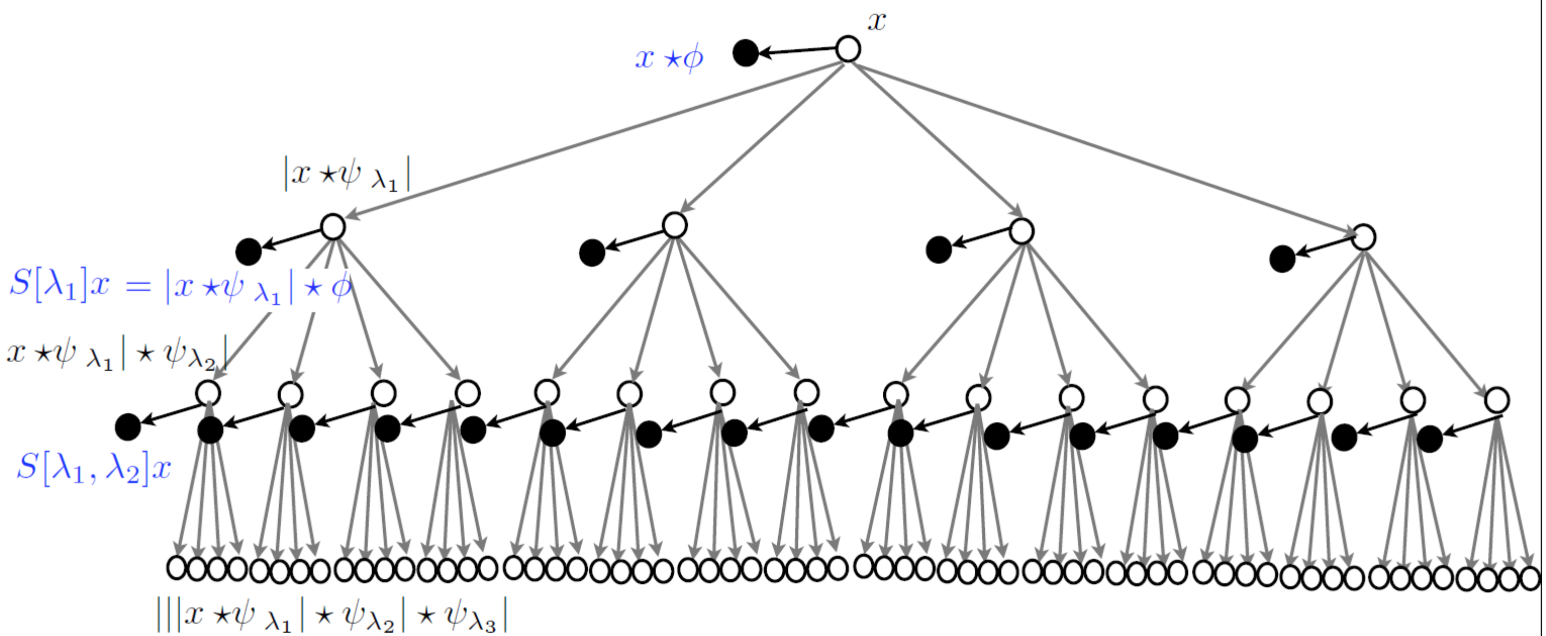
Figure 3. This sculpture by Henri Matisse illustrates the Deep Rendering Model (DRM). The sculpture in the leftmost panel is analogous to a fully rendered image at the lowest abstraction level $\ell = 0$. Moving from left to right, the sculptures become progressively more abstract, until the in the rightmost panel we reach the highest abstraction level $\ell = 3$. The finer-scale details in the first three panels that are lost in the fourth are the nuisance parameters g, whereas the coarser-scale details in the last panel that are preserved are the target c.

Scattering Transform (Mallat, 2011)

- Invariance and deformation stability
  – Fourier failure
  – Wavelet stability to deformations
  – Scattering invariants and deep convolution networks
  – Mathematical properties of deep scattering networks
  – Classification of images

Conv Net using Scattering Transform

- Iteration on $Ux = \{x \star \phi, |x \star \psi_\lambda|\}_\lambda$, contracting.

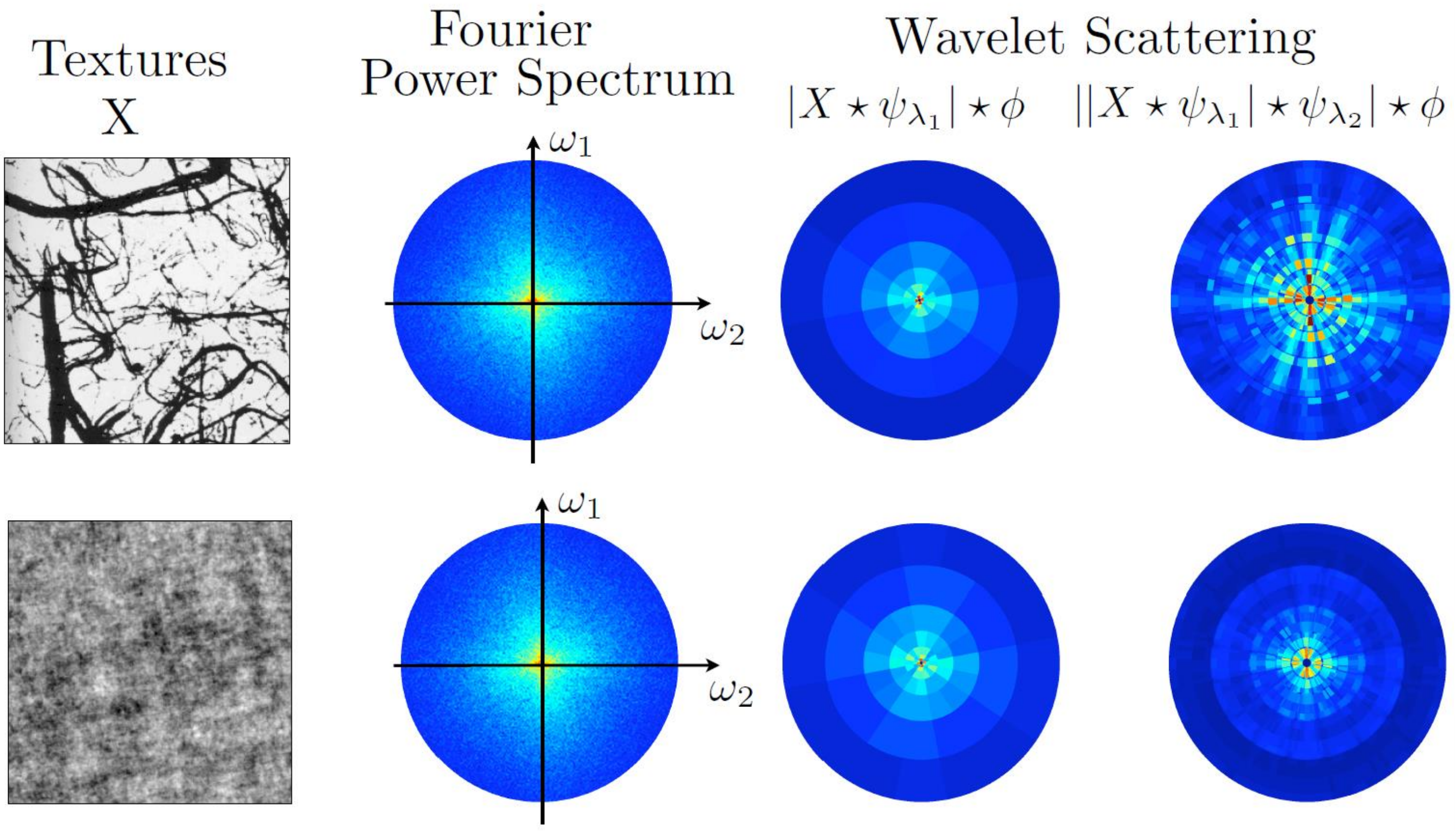


- Output at **all layers**: $\{S[p]x\}_{p \in \mathcal{P}}$.

  MFSC and SIFT are 1st layer outputs: $S[\lambda_1]x$

Textures with same spectrum

X: stationary process



window size = image size