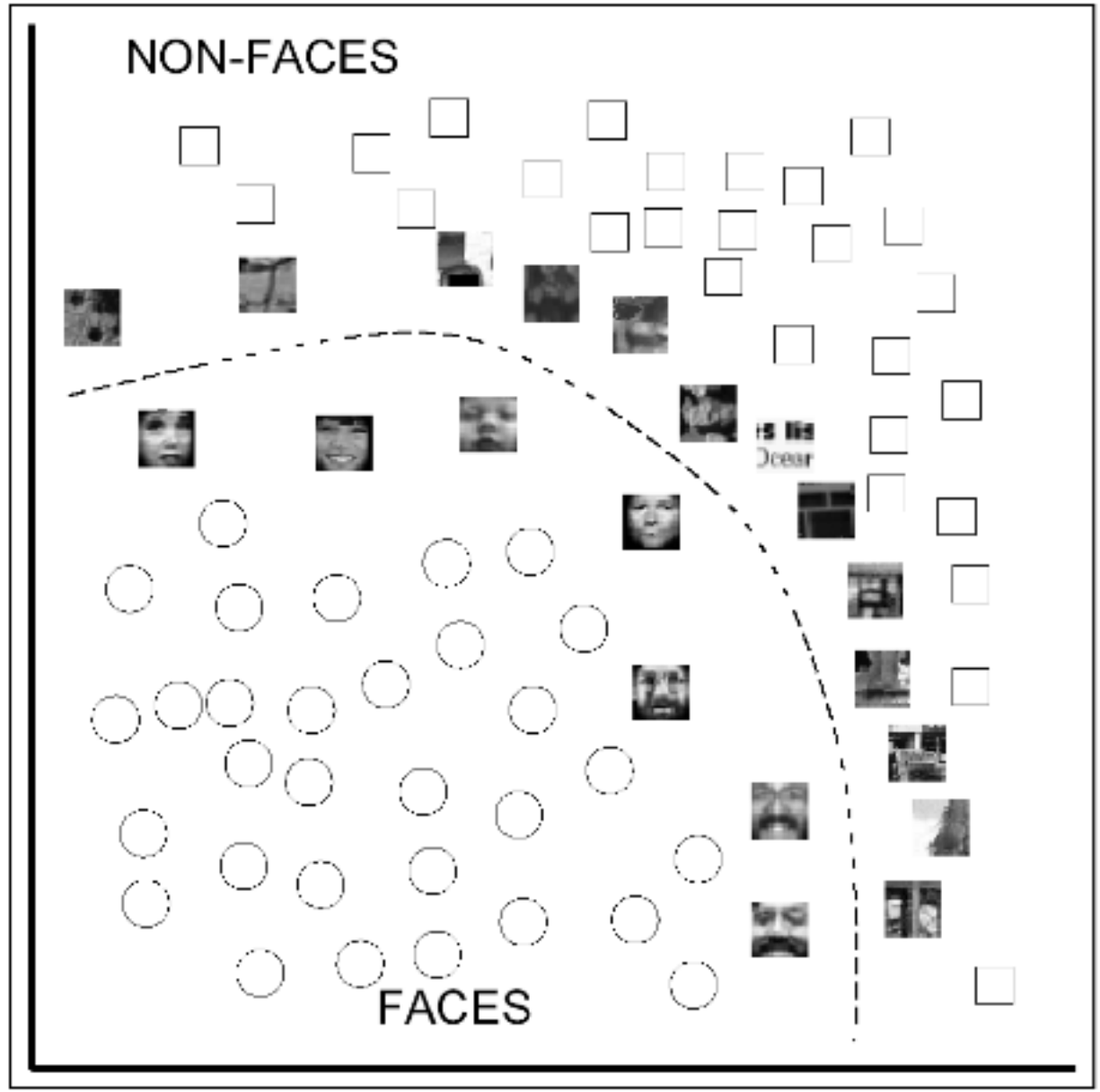
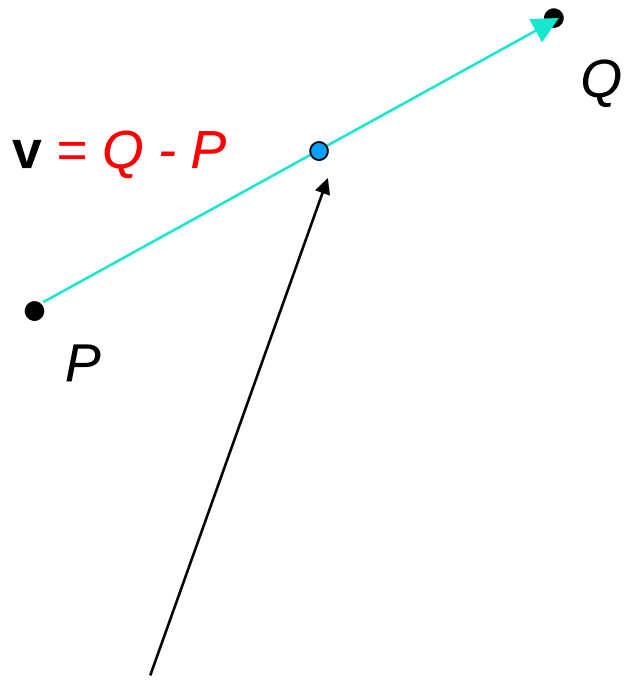
Linear Perceptron

"It says it's sick of doing things like inventories and payrolls, and it wants to make some breakthroughs in astrophysics."


NON-FACES
FACES

Averaging Points

What's the average of P and Q?



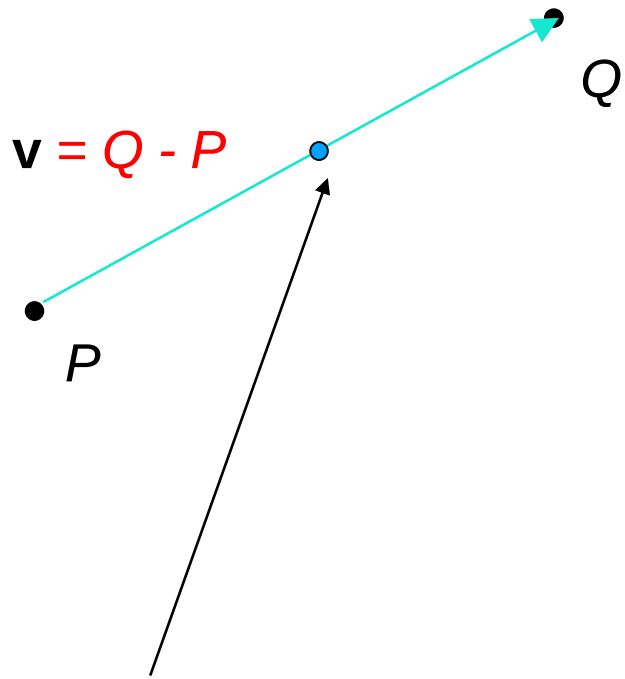
$P + 0.5\mathbf{v}$

$= P + 0.5(Q - P)$

$= 0.5P + 0.5\,Q$

Averaging Points

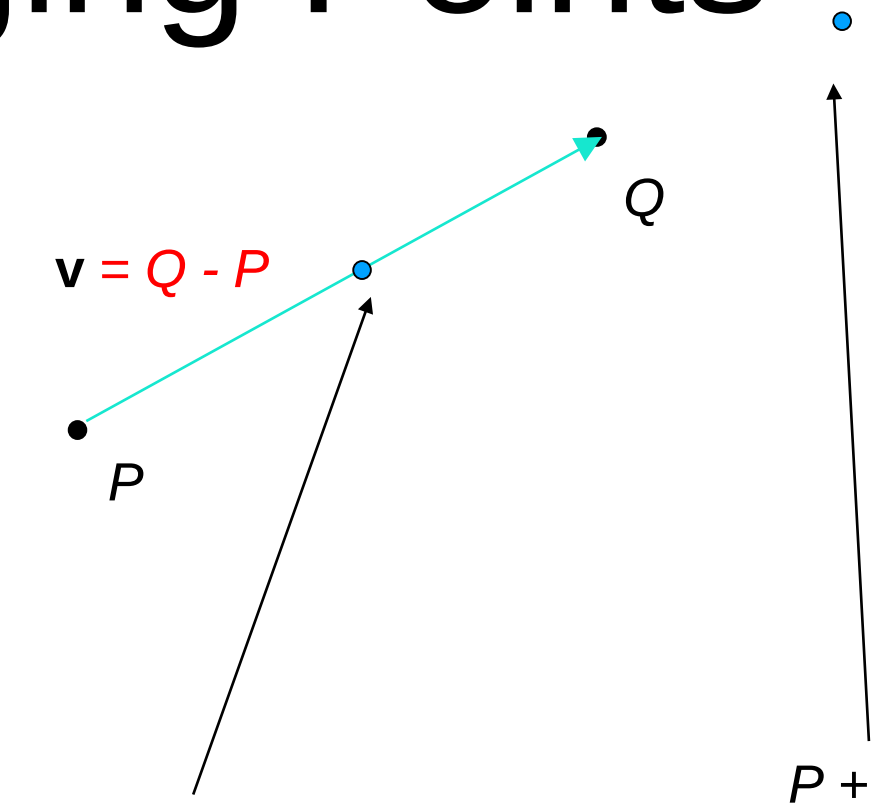
What's the average of P and Q?



$P + 0.5\mathbf{v}$
$= P + 0.5(Q - P)$
$= 0.5P + 0.5Q$

Linear Interpolation (Affine Combination):
New point $aP + bQ$,
defined only when $a+b = 1$
So aP+bQ = aP+(1-a)Q

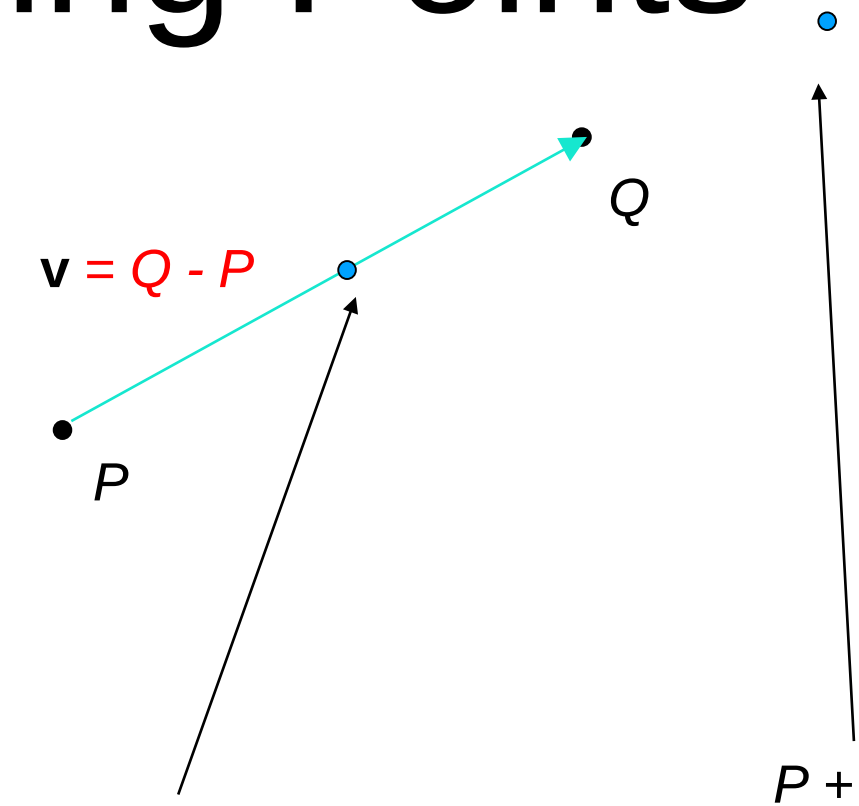
Averaging Points



$P + 0.5\mathbf{v}$
$= P + 0.5(Q - P)$
$= 0.5P + 0.5\,Q$

$P + 1.5\mathbf{v}$
$= P + 1.5(Q - P)$
$= -0.5P + 1.5\,Q$
(extrapolation)

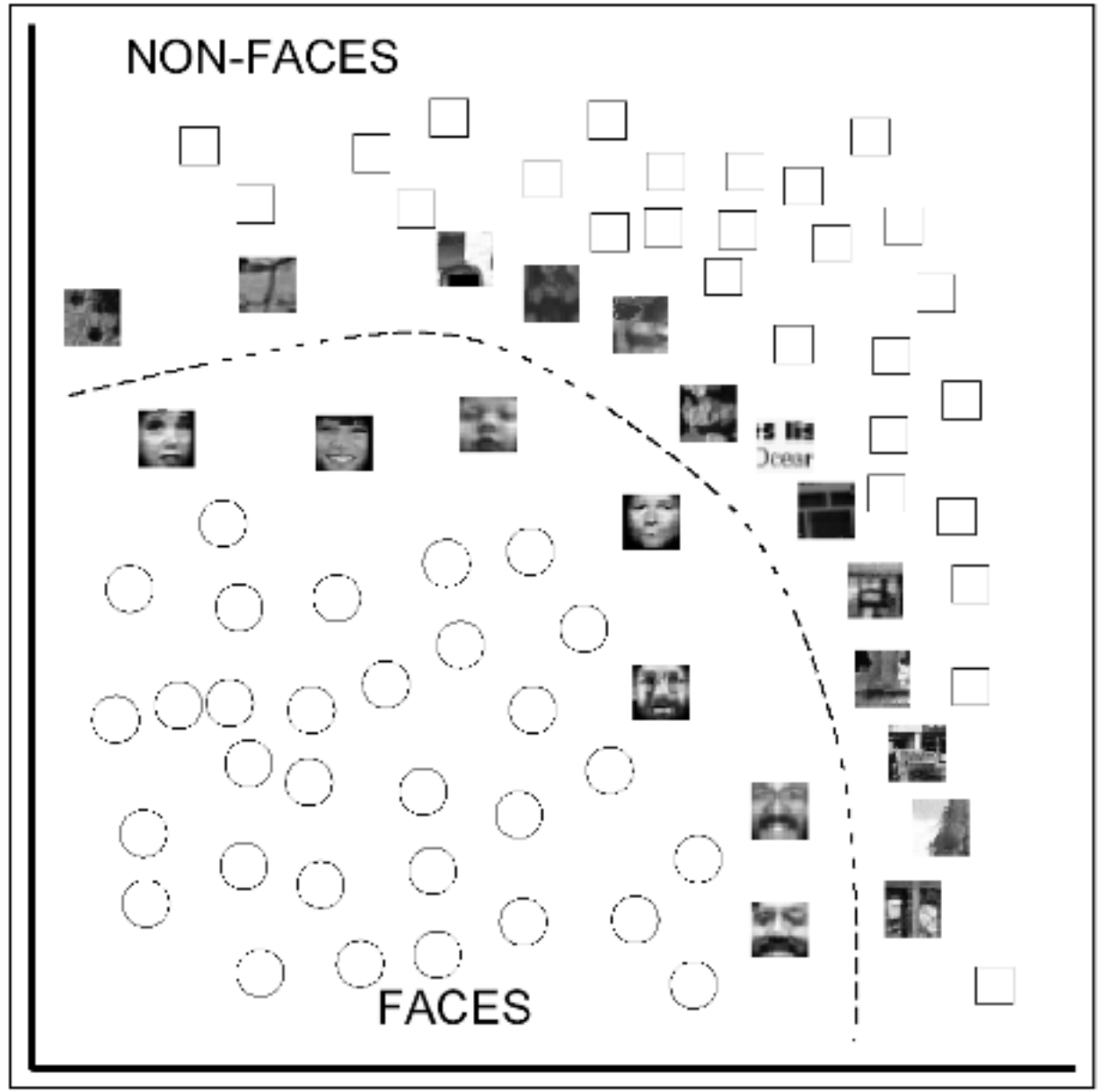
Averaging Points



$P + 0.5\mathbf{v}$
$= P + 0.5(Q - P)$
$= 0.5P + 0.5\,Q$

$P + 1.5\mathbf{v}$
$= P + 1.5(Q - P)$
$= -0.5P + 1.5\,Q$
(extrapolation)

- P and Q can be anything:
  - points on a plane (2D) or in space (3D)
  - Colors in RGB or HSV (3D)
  - Whole images (m-by-n D)… etc.


NON-FACES
FACES

Averaging Images: Cross-Dissolve

Interpolate whole images:

$$Image_{halfway} = (1-t)*Image_1 + t*image_2$$

This is called **cross-dissolve** in film industry

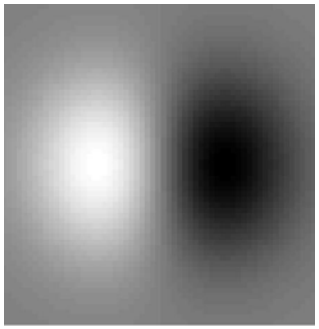
Averaging Images: Cross-Dissolve

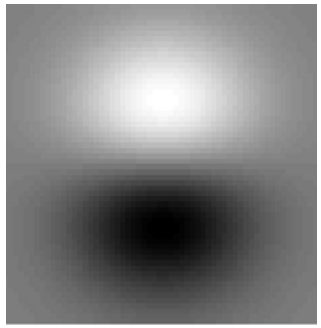
Interpolate whole images:

$$Image_{halfway} = (1-t)*Image_1 + t*image_2$$

This is called **cross-dissolve** in film industry

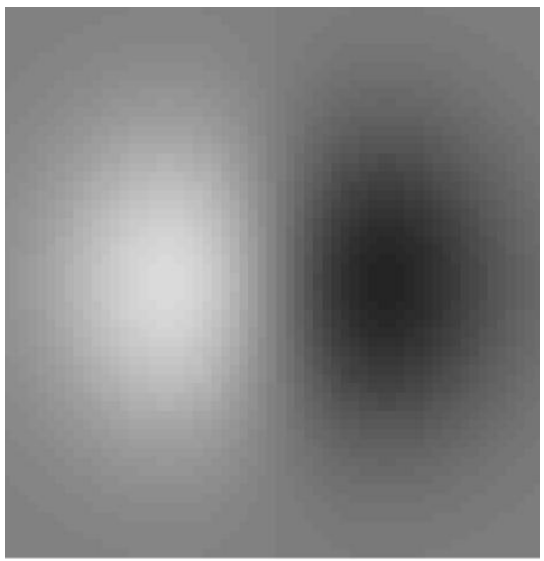
Averaging Images



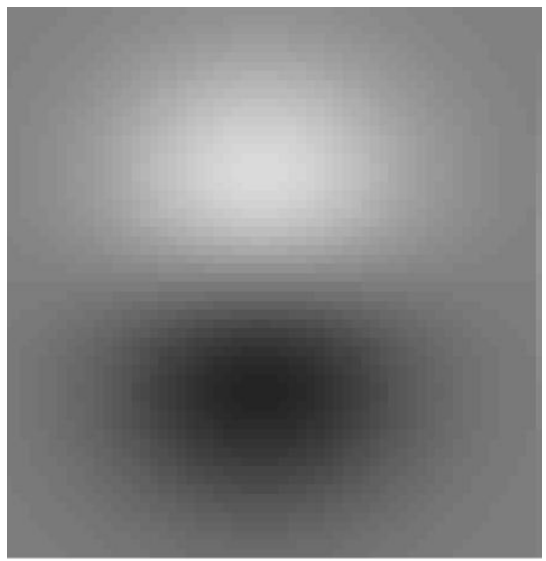


$Image_1$ $Image_2$

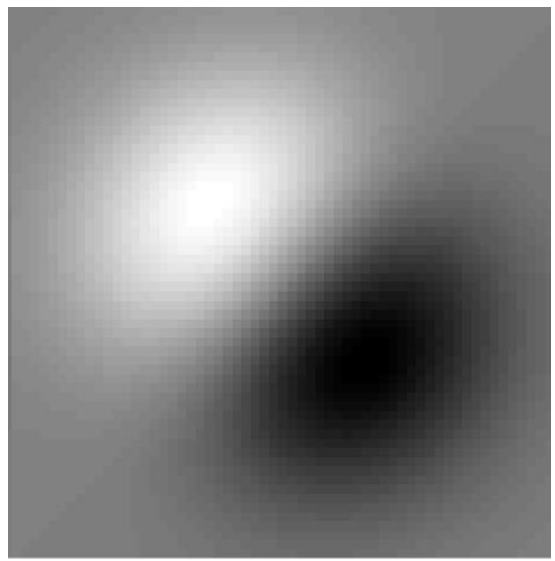
t = 0 t = 0.5 t = 1



+

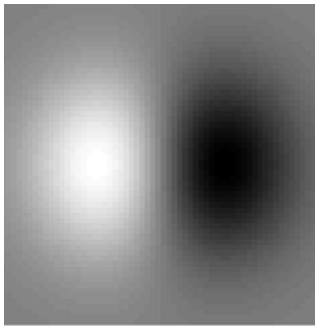


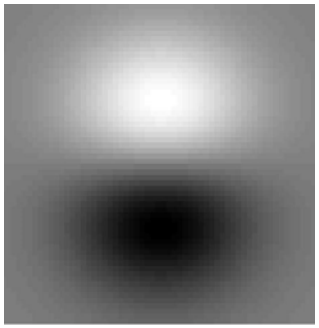
=



$(1-t)*Image_1$ + $t*Image_2$ = $Image_{halfway}$

Averaging Images

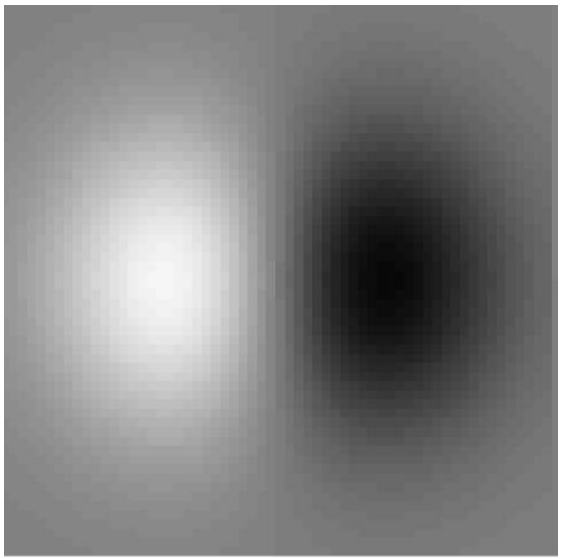




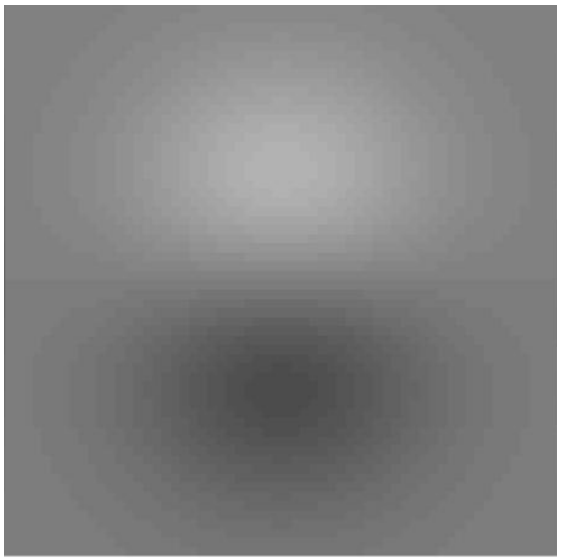
Averaging Images = Rotating Objects

$Image_1$ $Image_2$

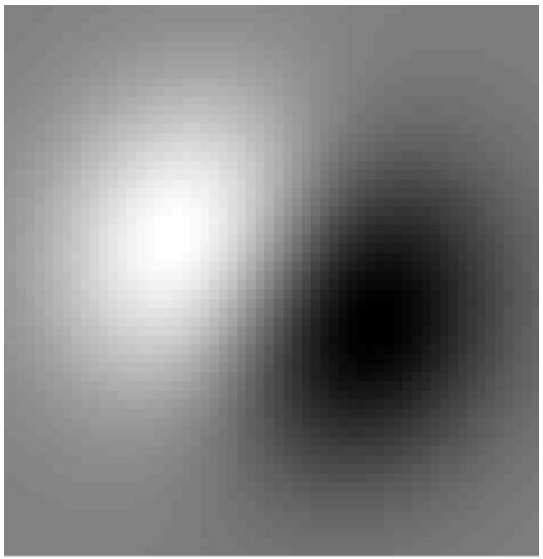
t = 0 t = 0.3 t = 1



+



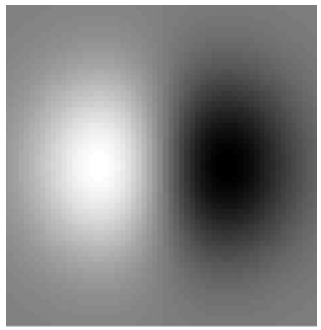
=



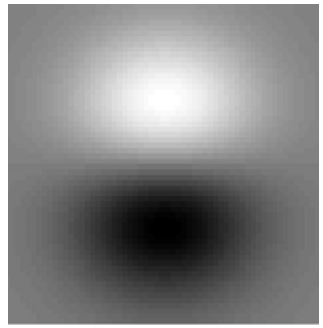
$(1-t)*Image_1$ + $t*Image_2$ = $Image_{halfway}$

Averaging Images

Averaging Images = Rotating Objects



$Image_1$

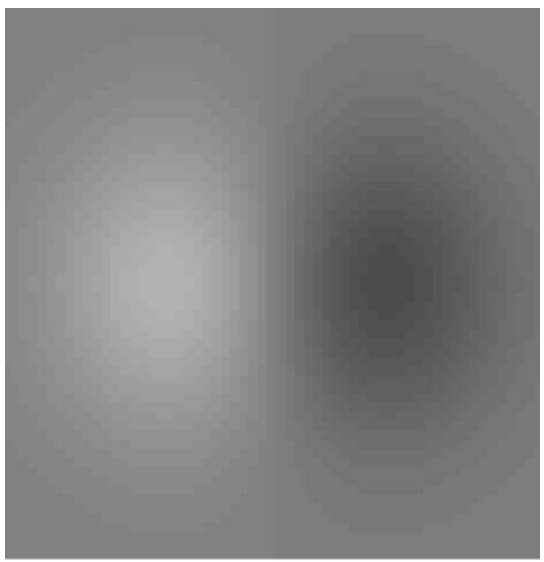


$Image_2$

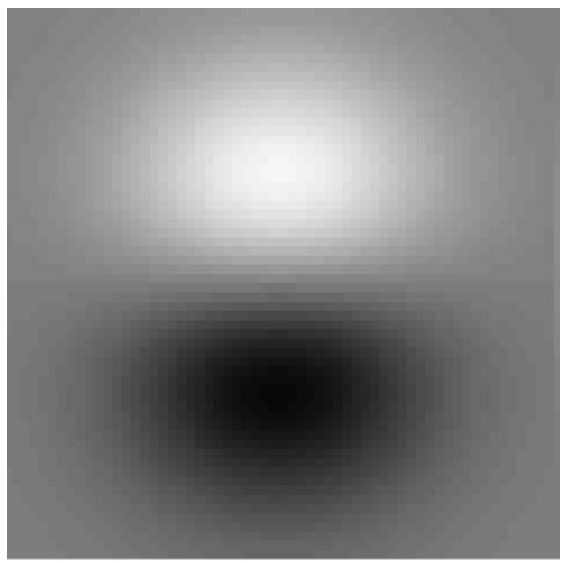
t = 0

t = 0.7

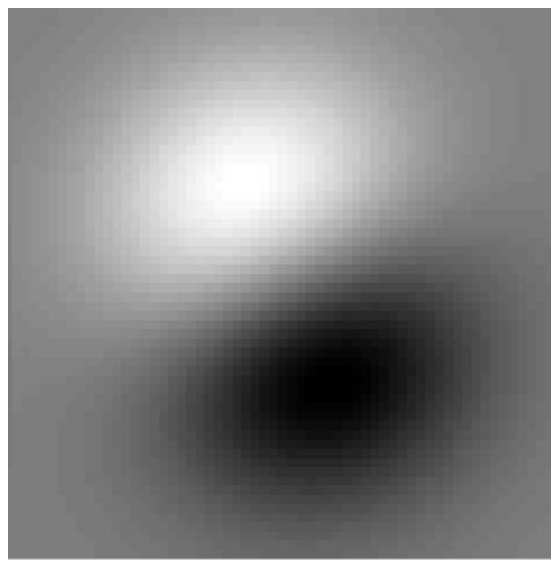
t = 1



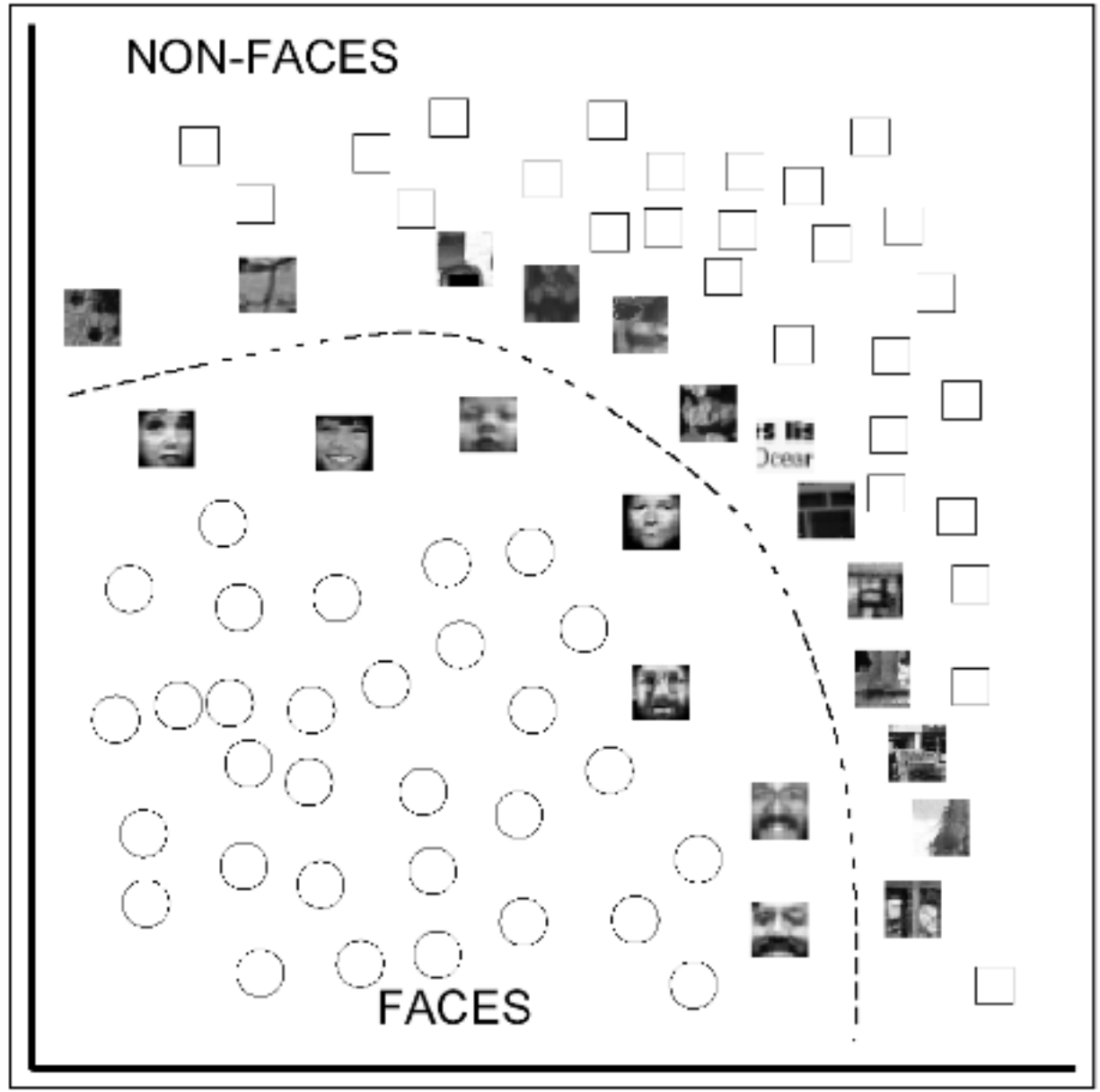
+

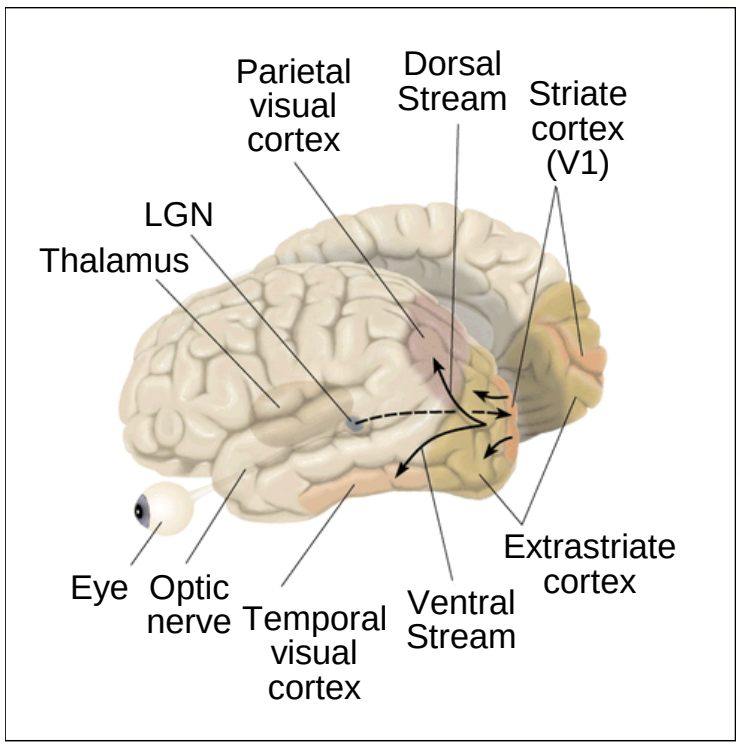


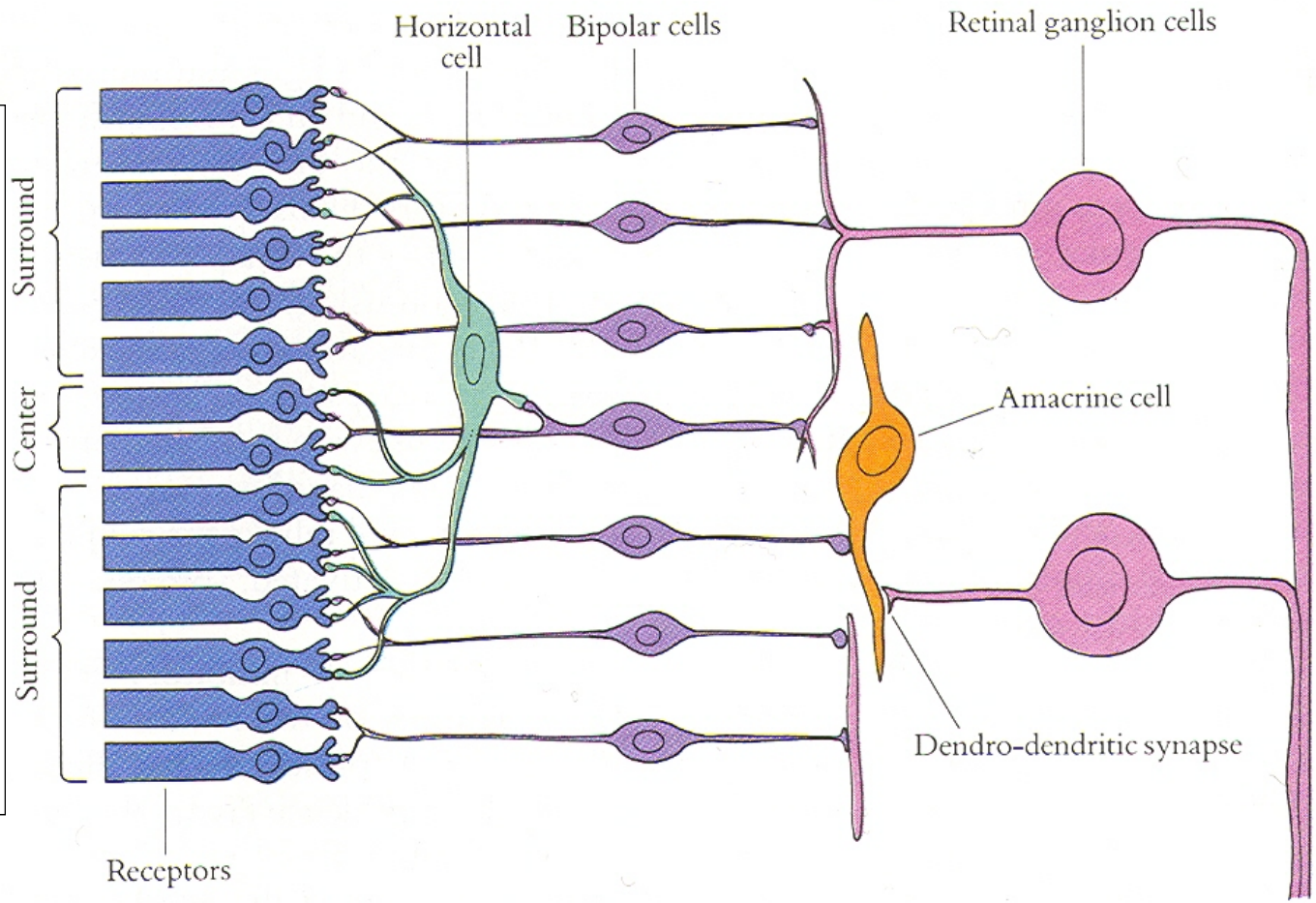
=



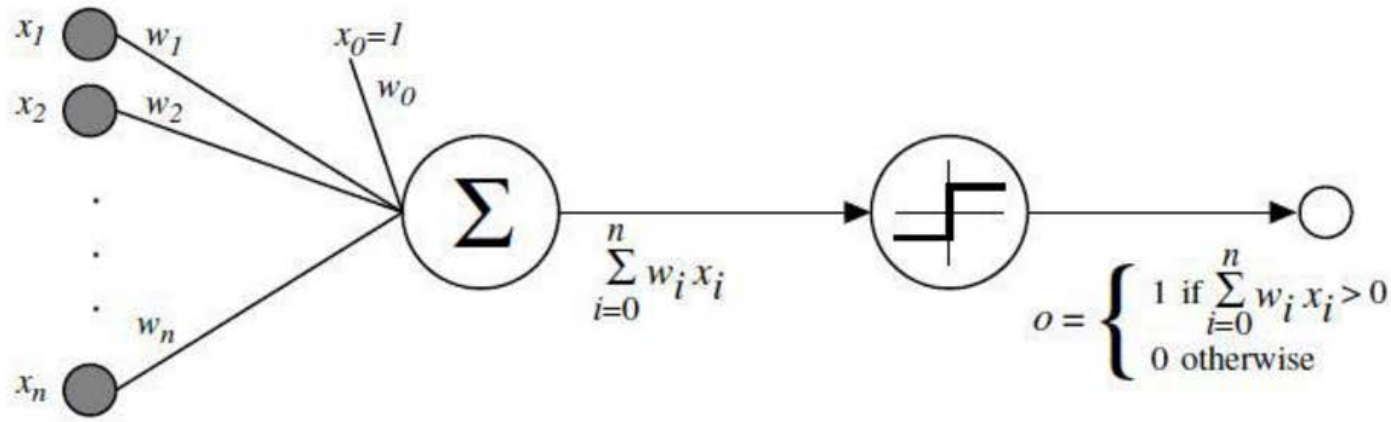
$(1-t)*Image_1$ + $t*Image_2$ = $Image_{halfway}$


NON-FACES
FACES


Parietal visual cortex
Dorsal Stream
Striate cortex (V1)
LGN
Thalamus
Eye
Optic nerve
Temporal visual cortex
Ventral Stream
Extrastriate cortex


Horizontal cell
Bipolar cells
Retinal ganglion cells
Surround
Center
Surround
Amacrine cell
Dendro-dendritic synapse
Receptors

Perceptron [Rosenblatt 1958]:



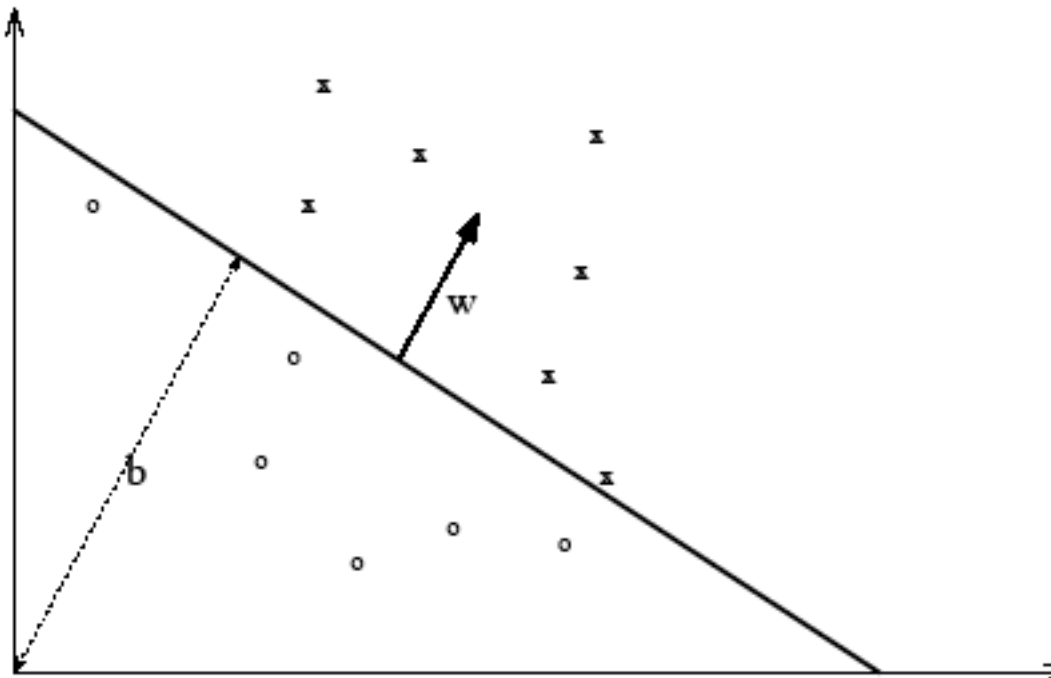
- **A weighted sum of the features with a non-linearity added on top.**
- **Weights can be learned by minimizing the error of the prediction using a gradient descent.**

- Inner product between vectors

$$\langle \bar{x}, \bar{z} \rangle = \sum_i x_i z_i$$

- Hyperplane:

$$\langle w, x \rangle + b = 0$$



- Input space $x \in X$
- Output space $y \in Y = \{-1, +1\}$
- Hypothesis $h \in H$
- Real-valued: $f : X \to \mathrm{R}$
- Training Set $S = \{(x_1, y_1), \ldots, (x_i, y_i), \ldots.\}$
- Test error ε
- Dot product $\langle x, z \rangle$

Walter Pitts was born in 1923, vanished from the scene in the late 1950s, and died at the end of the 1960s, having destroyed, as much as he could, any traces of his past existence. He is a peculiarly difficult subject for a biography because, although he remains a vividly haunting memory to those who knew him, he seems only a group delusion to others. At least that was the opinion of the neurologist Norman GESCHWIND.

Pitts appeared as a penniless 14-year-old at the University of Chicago in 1937, attended various classes, though unregistered, and was accepted by Rashevsky's coterie as a very talented but mysterious junior. All that was known of him was that he came from Detroit, and that would be all that was known thereafter.

Pitts is best known for his contribution to "A Logical Calculus of Ideas Immanent in Nervous Activity" (1943), which he co-authored with Warren McCulloch.

Walter Pitts (April 23 1923 – 1969) was a logician who worked in the field of cognitive psychology.

Pitts taught himself logic and mathematics and was able to read a number of languages including Greek and Latin. At the age of 12 he spent three days in a library reading *Principia Mathematica* and sent a letter to Bertrand Russell pointing out what he considered serious problems with the first half of the first volume. Russell was appreciative and invited him to study in the United Kingdom; although this offer was apparently not taken up, Pitts decided to become a logician.

He attended lectures at the University of Chicago, without registering as a student, and met Jerome Lettvin with whom he became good friends. In 1938 he met Rudolf Carnap by walking into his office and presenting him with an annotated version of Carnap's recent book on logic. Since Pitts did not introduce himself, Carnap spent months searching for him, and when he found him he obtained for him a menial job at the university. Pitts at the time was homeless and without income.

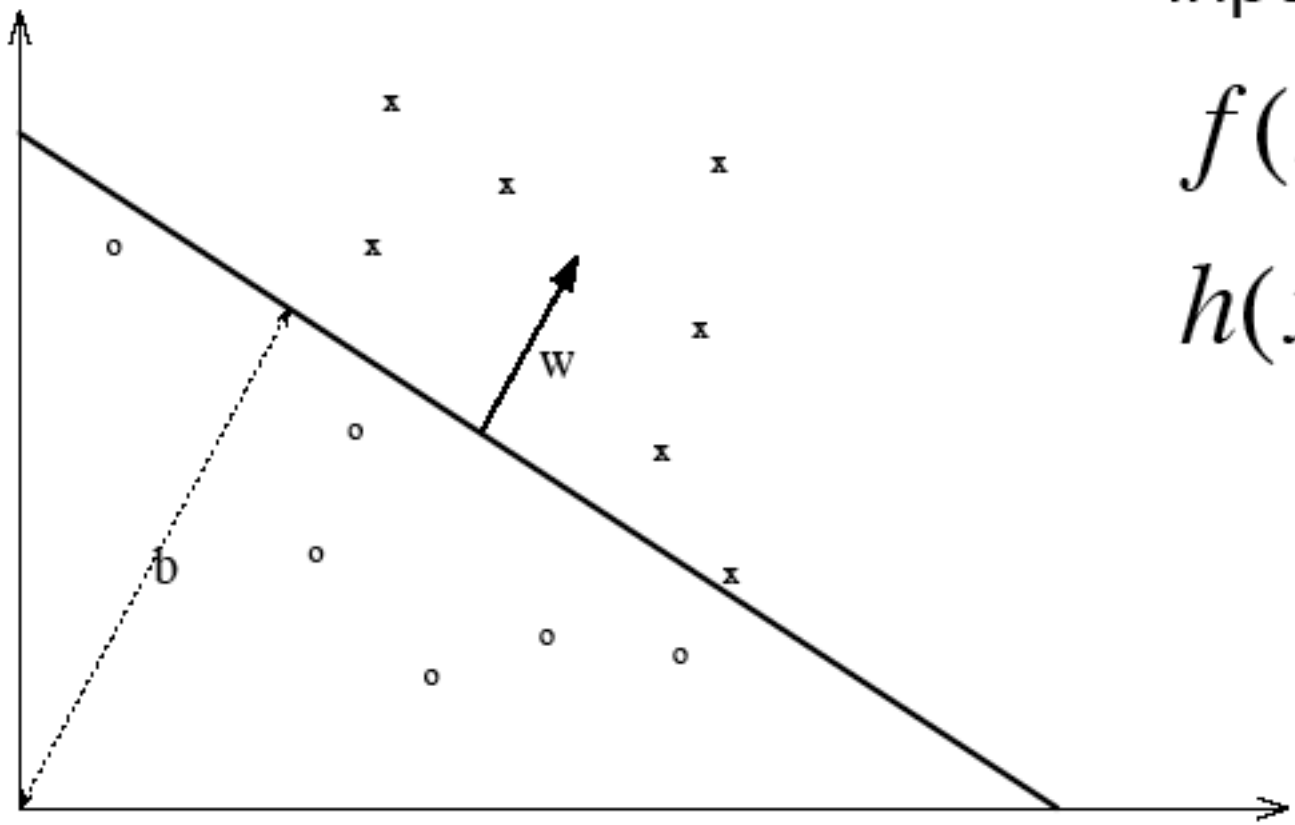
Later Warren McCulloch also arrived at the university, and in early 1942 invited Pitts, who was still homeless, together with Lettvin to live with his family. In the evenings McCulloch and Pitts collaborated. Pitts was familiar with the work of Gottfried Leibniz on computing and they considered the question of whether the nervous system could be considered a kind of universal computing device as described by Leibniz. This led to their seminal neural networks paper *A Logical Calculus of Ideas Immanent in Nervous Activity*.

In 1943 Lettvin introduced Pitts to Norbert Wiener at MIT, who had recently lost his "right-hand man". Their first meeting, where they discussed Wiener's proof of the ergodic theorem, went so well that Pitts moved to Boston to work with Wiener. In 1944 Pitts was hired by Kellex Corporation, part of the Atomic Energy Project.

In 1951 Wiener convinced Jerry Wiesner to hire some physiologists of the nervous system. A group was established with Pitts, Lettvin, McCulloch, and Pat Wall. Pitts wrote a large thesis on the properties of neural nets connected in three dimensions. Lettvin described him as "in no uncertain sense the genius of the group … when you asked him a question, you would get back a whole textbook". Pitts was also described as an eccentric, refusing to allow his name to be made publicly available. He refused all offers of advanced degrees or official positions at MIT as he would have to sign his name.

Wiener suddenly turned against McCulloch, on account of his wife Margaret Wiener who hated McCulloch, and broke off relations with anyone connected to him including Pitts. This sent Walter Pitts into 'cognitive suicide', a gradual but steep decline into social isolation, from which he never recovered. He burnt the manuscript on three dimensional networks and took little further interest in work. The only exception was a collaboration with Robert Gesteland which produced a paper on olfaction. Pitts died in 1969. The mathematical model of a neuron is today called as McCulloch–Pitts neuron. The theoretical formulation of the neural activity of the brain remains as the lasting legacy of Walter Pitts and Warren McCulloch to the Cognitive sciences.

Perceptron



- Linear Separation of the input space

$$f(x) = \langle w, x \rangle + b$$

$$h(x) = sign(f(x))$$

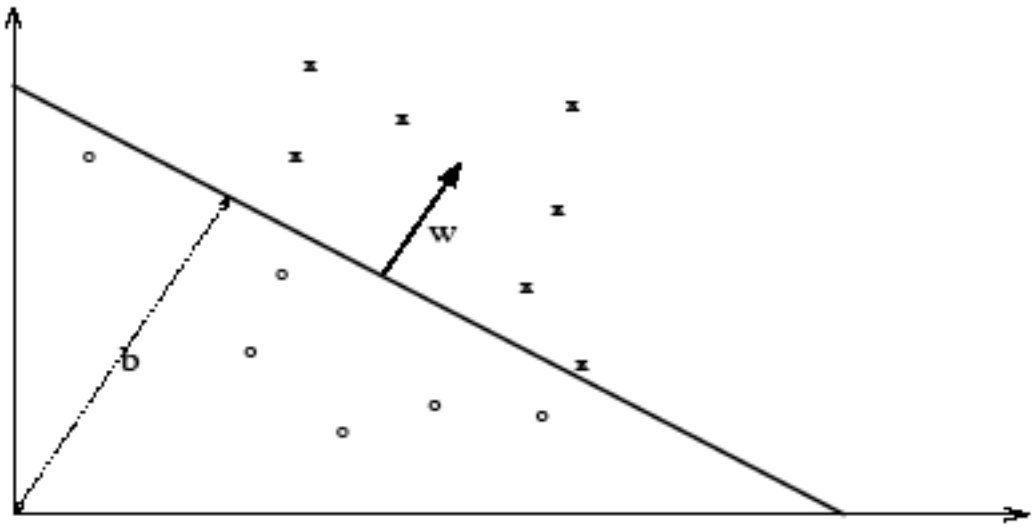
McCulloch–Pitts neuron

Update rule
(ignoring threshold):

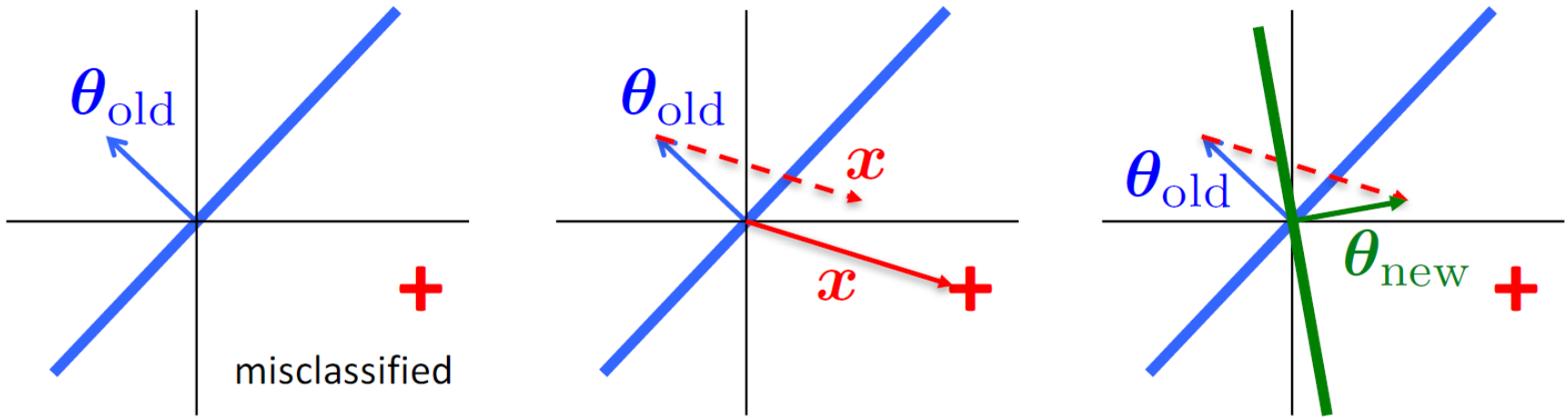
- if $y_i(\langle w_k, x_i \rangle) \leq 0$ then

$$w_{k+1} \leftarrow w_k + \eta y_i x_i$$

$$k \leftarrow k+1$$



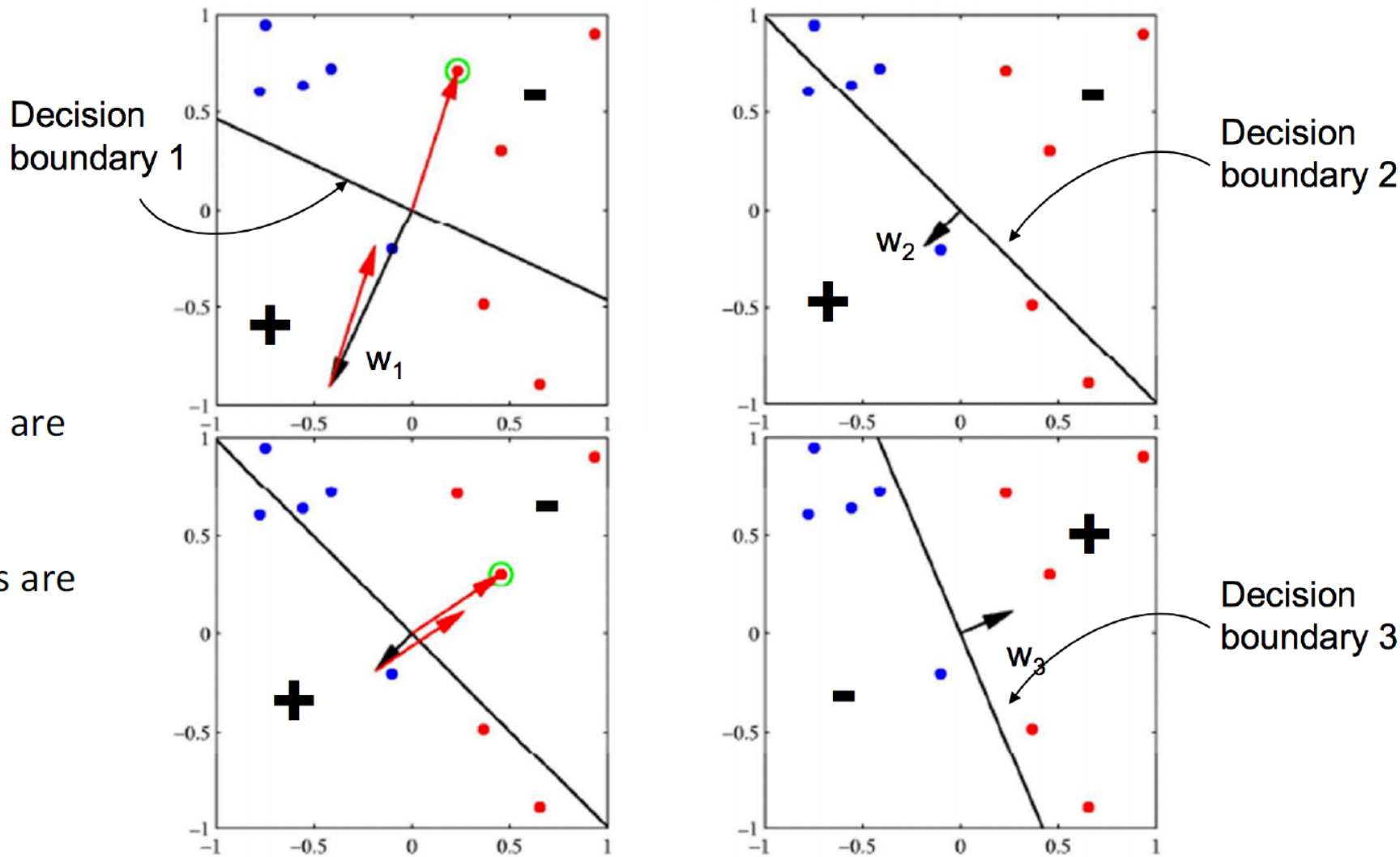
Why the Perceptron Update Works



Based on slide by Piyush Rai

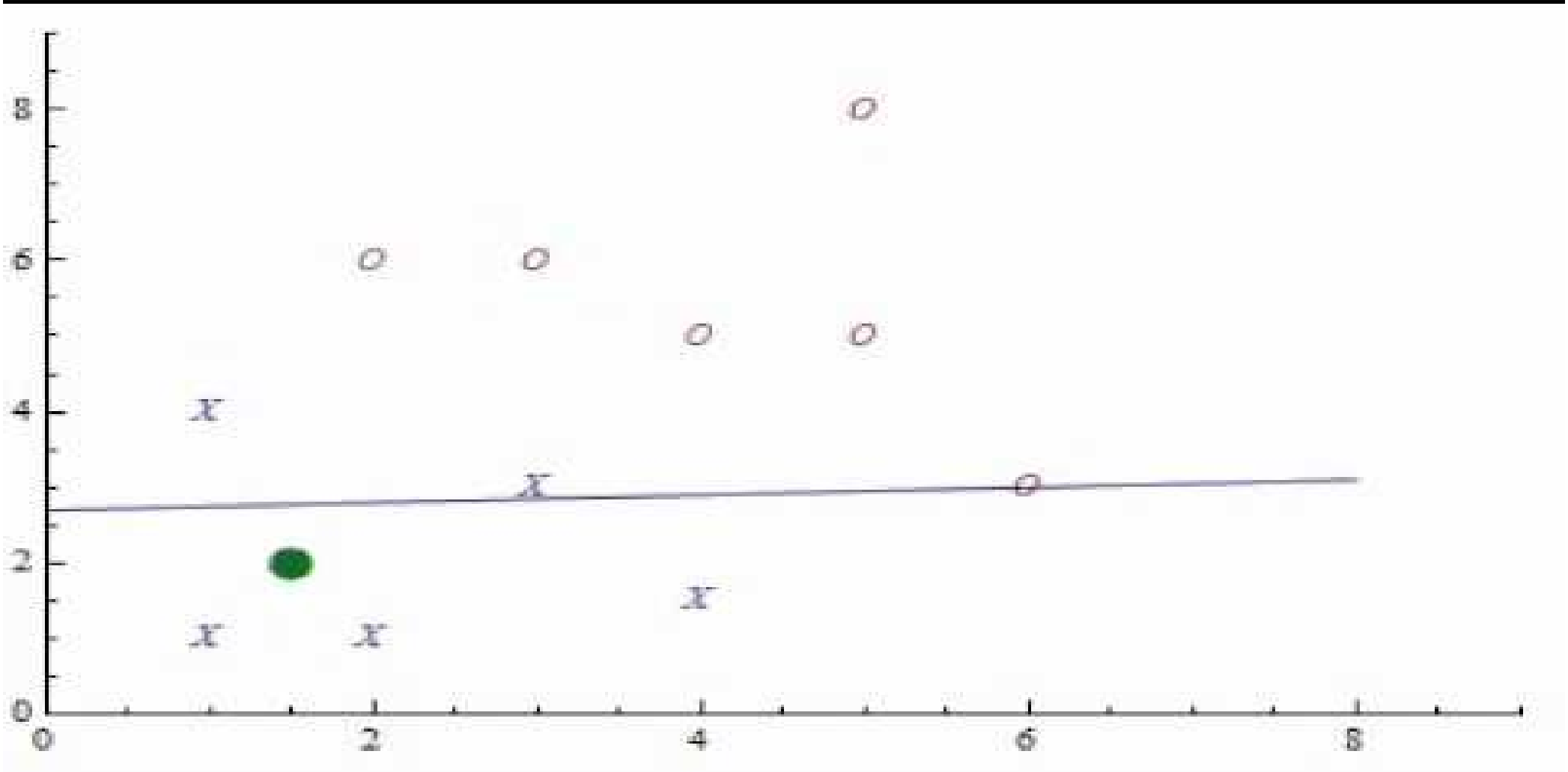
Online Perceptron Algorithm

When an error is made, moves the weight in a direction that corrects the error



Red points are labeled +

Blue points are labeled -


8
6
4
2
0
0
2
4
6
8

- Solution is a linear combination of training points $$w = \sum \alpha_i y_i x_i$$ $$\alpha_i \geq 0$$
- Only used informative points (mistake driven)
- The coefficient of a point in combination reflects its ‘difficulty’

The decision function can be re-written as follows:

$$f(x) = \langle w, x \rangle + b = \sum \alpha_i y_i \langle x_i, x \rangle + b$$

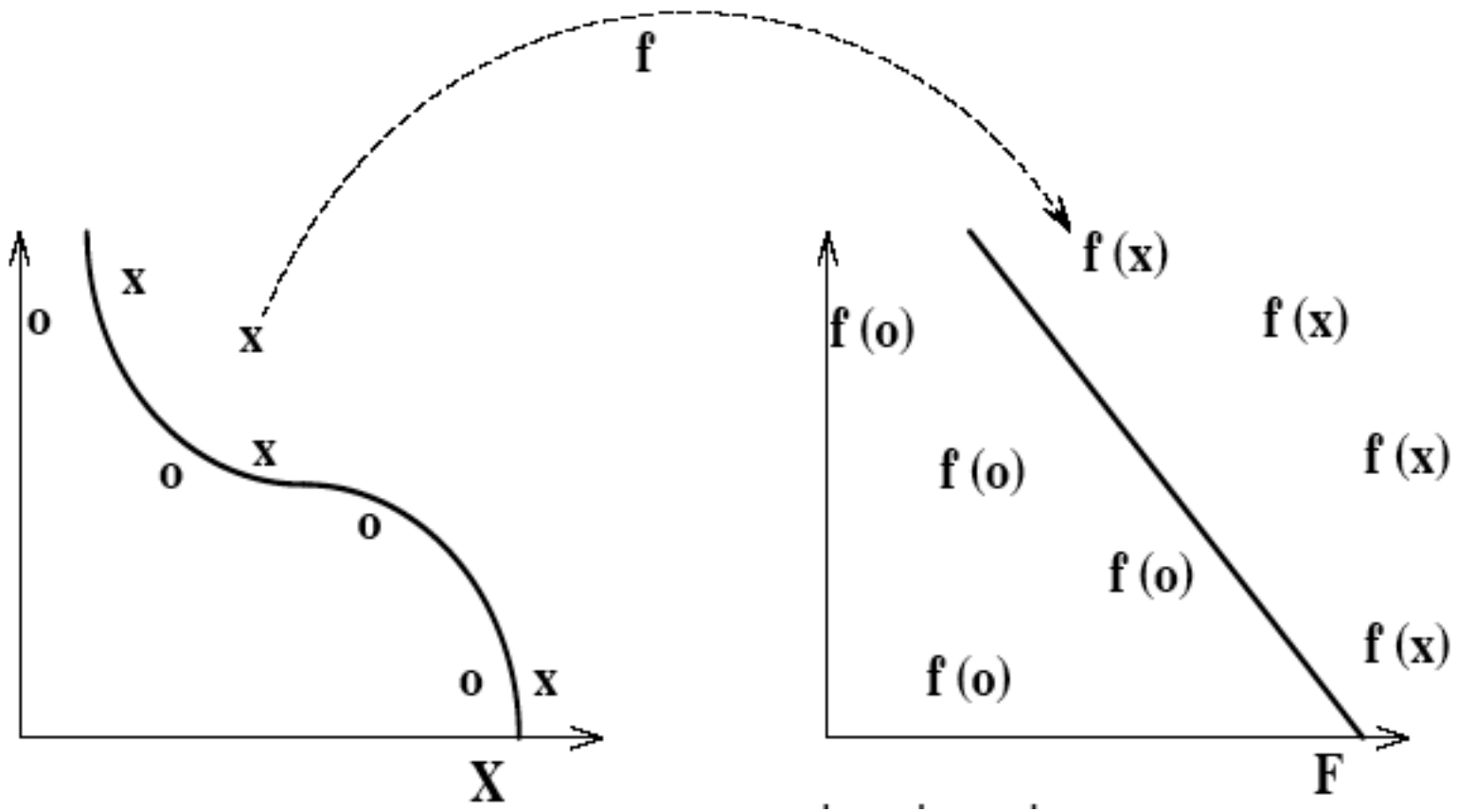
$$w = \sum \alpha_i y_i x_i$$

- One solution: creating a net of simple linear classifiers (neurons): a Neural Network

  (problems: local minima; many parameters; heuristics needed to train; etc)

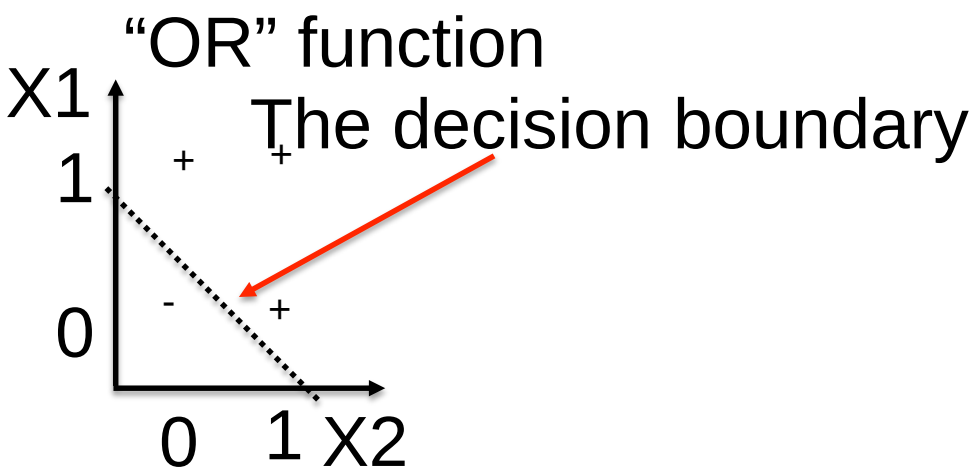
- Other solution: map data into a richer feature space including non-linear features, then use a linear classifier

- Map data into a feature space where they are linearly separable $x \rightarrow \phi(x)$



What is the problem with the Perceptron Algorithm?

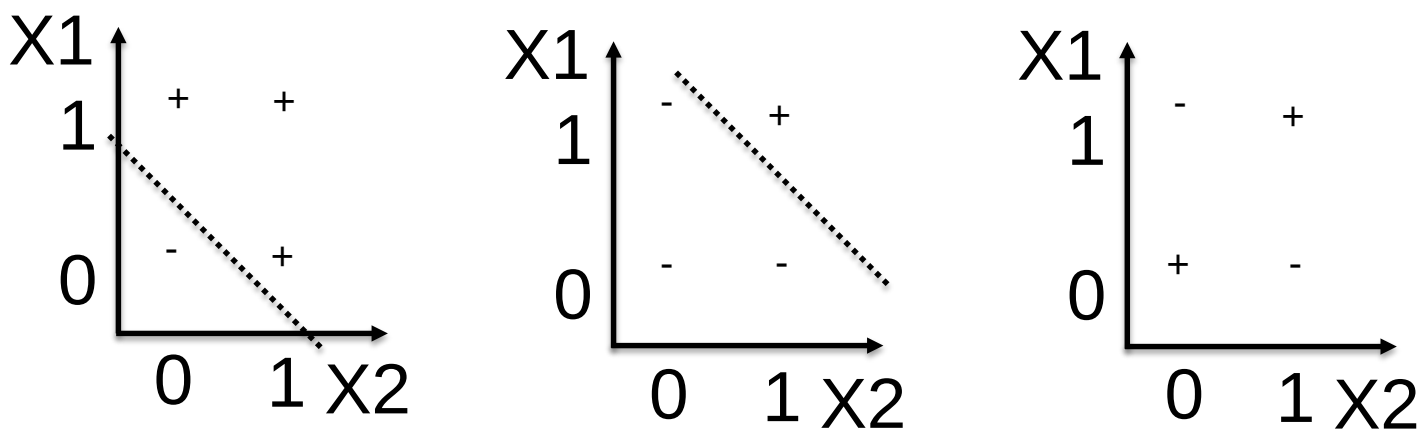
- To obtain correct predictions, the perceptron will learn weights (w1, w2 in this case)



What is the problem with the Perceptron Algorithm?

- To obtain correct predictions, the perceptron will learn weights (w1, w2 in this case)

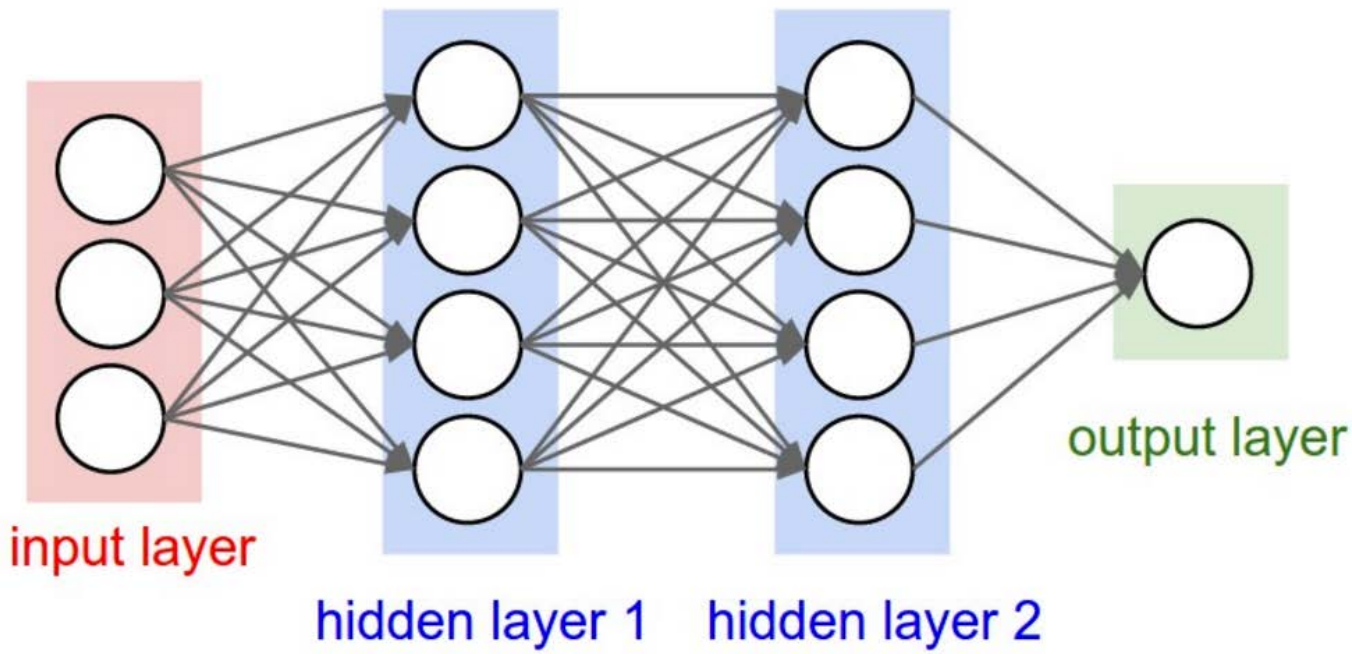
"OR" function "AND" function "XOR" function



- **Perceptron's decision boundary is linear**
- **Most of the problems exhibit non-linear relationships**

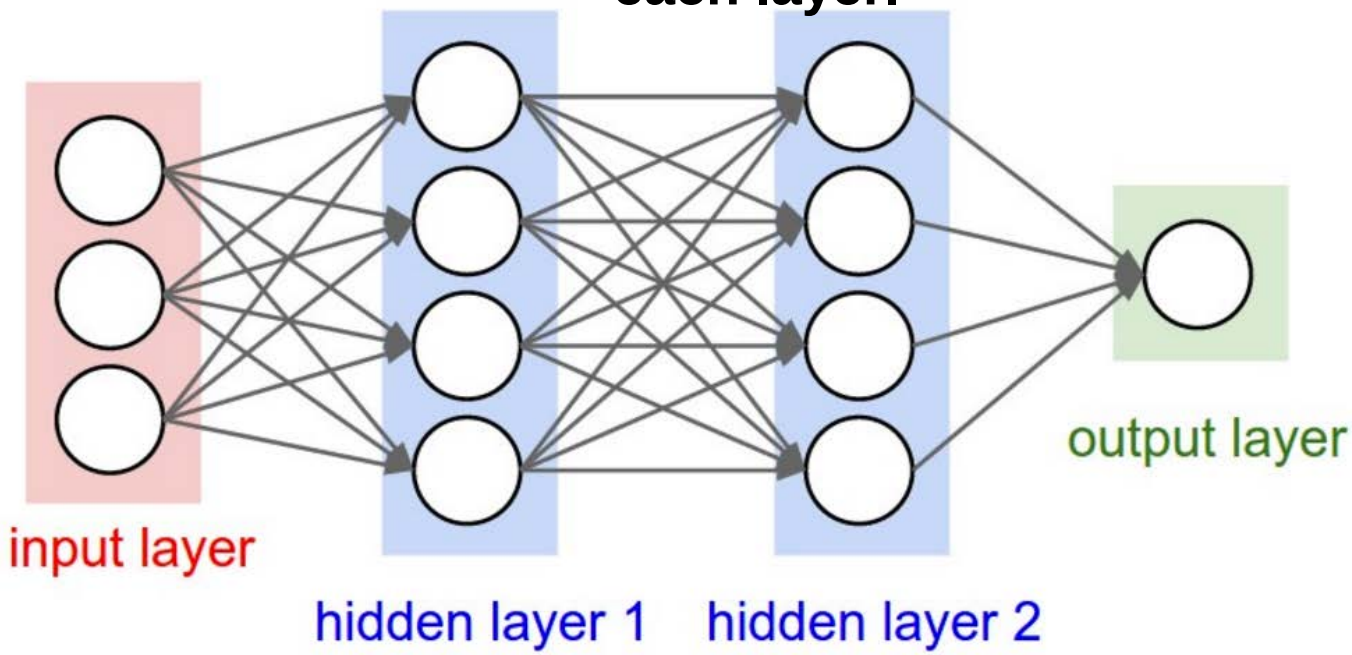
How can we address this problem?

- **A multi-layer perceptron then becomes a non-linear classifier (can solve XOR problem)**

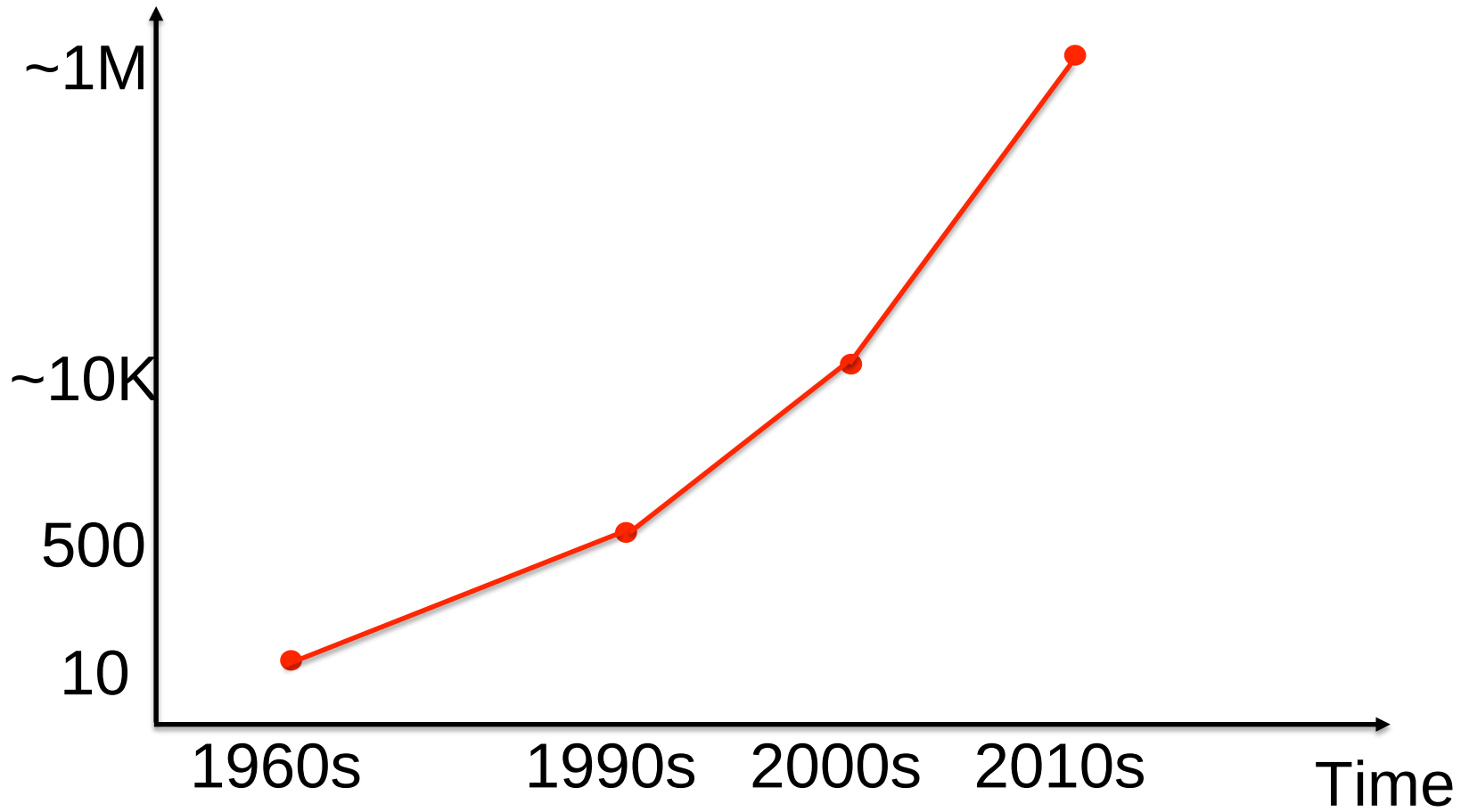


What's the problem with the multi-layer perceptron?

- **For a long time, researchers did not know how to propagate the error to every layer.**
- **If we cannot do so, there is no way to learn the weights in each layer.**

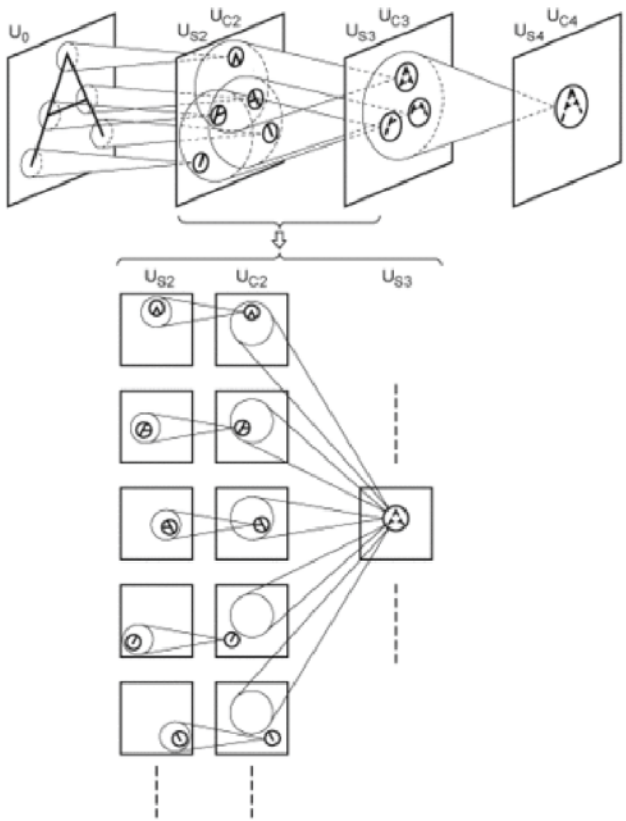


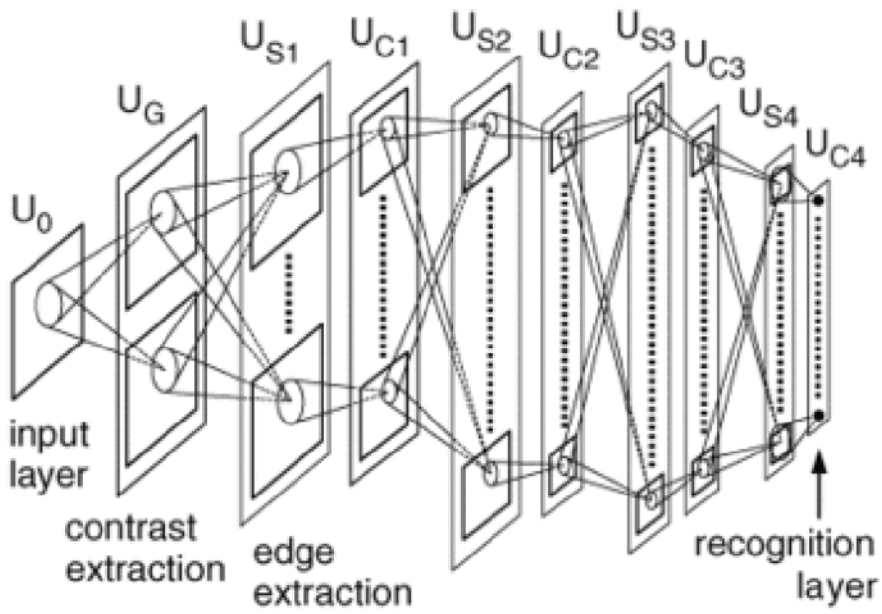
Labeled Dataset Size



Earliest “deep” architecture

Neocognitron

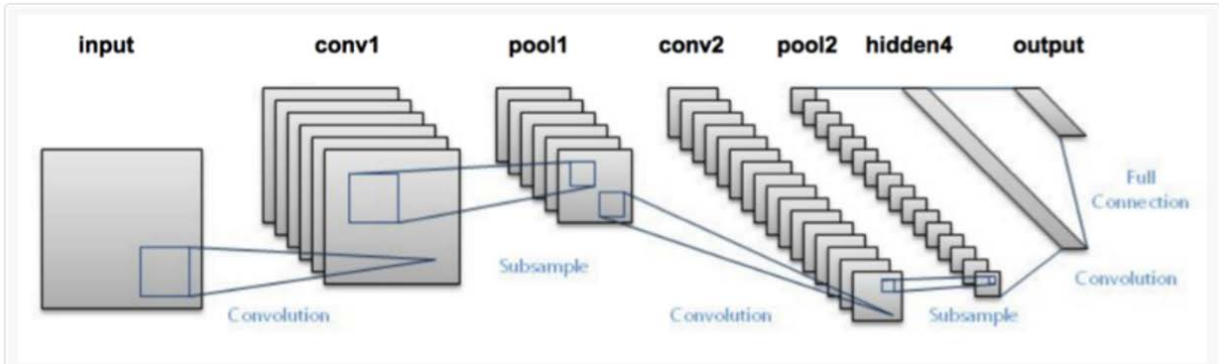




(Fukushima 1974-1982)

LeNet-5 (~1998):

- **Convolutional Neural Network used for hand-written digit recognition**
  - **~50K parameters**
- **Given a binary hand-written 28x28 digit image, the network predicts one of 10 digit categories.**



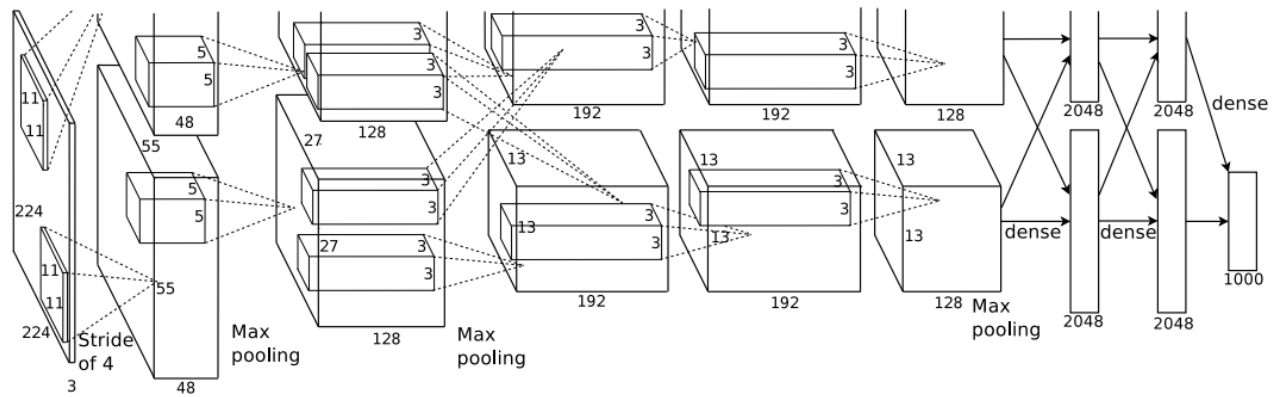
LeNet-5 (~1998):

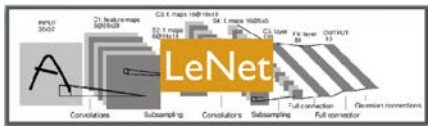
- **Convolutional Neural Network used for hand-written digit recognition**
- **~50K parameters**
- **Given a binary hand-written 28x28 digit image, the network predicts one of 10 digit categories.**

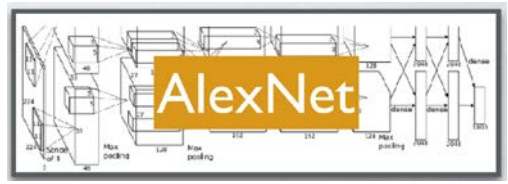
Figure 1: MNIST digit recognition dataset.

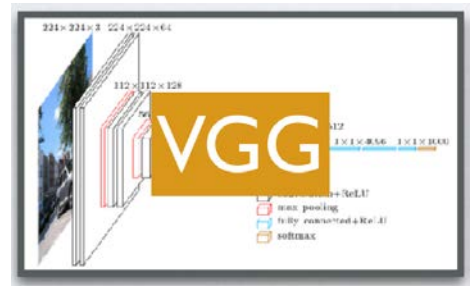
AlexNet (2011):

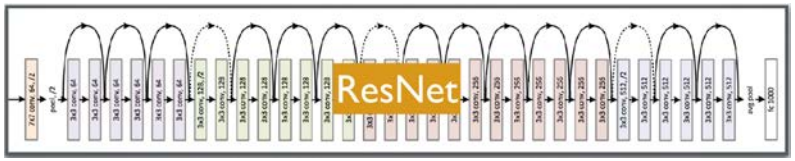
- **Convolutional Neural Network used for image categorization**
- **~60M parameters (8 layers)**
- **Given a 224x224 RGB image, the network predicts one of 1000 image categories.**



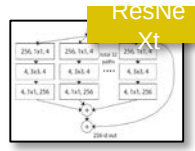




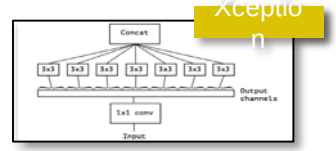




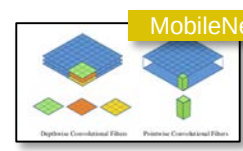
ResNeXt



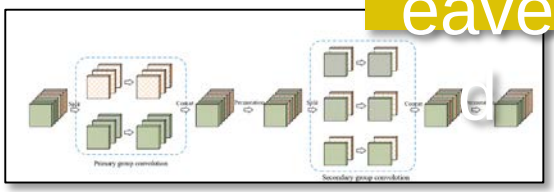
Xception



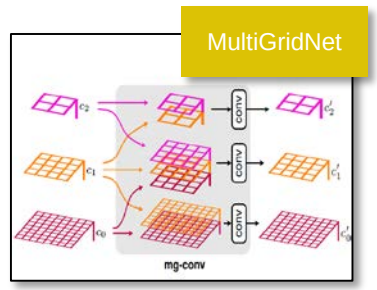
MobileNet



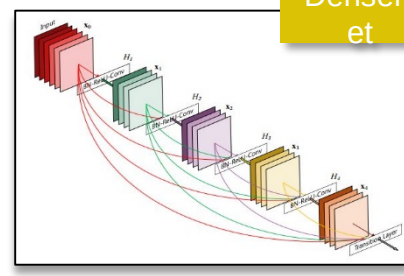
InterLeaved



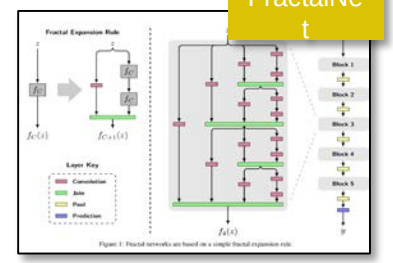
MultiGridNet



DenseNet



FractalNet



Supervised training

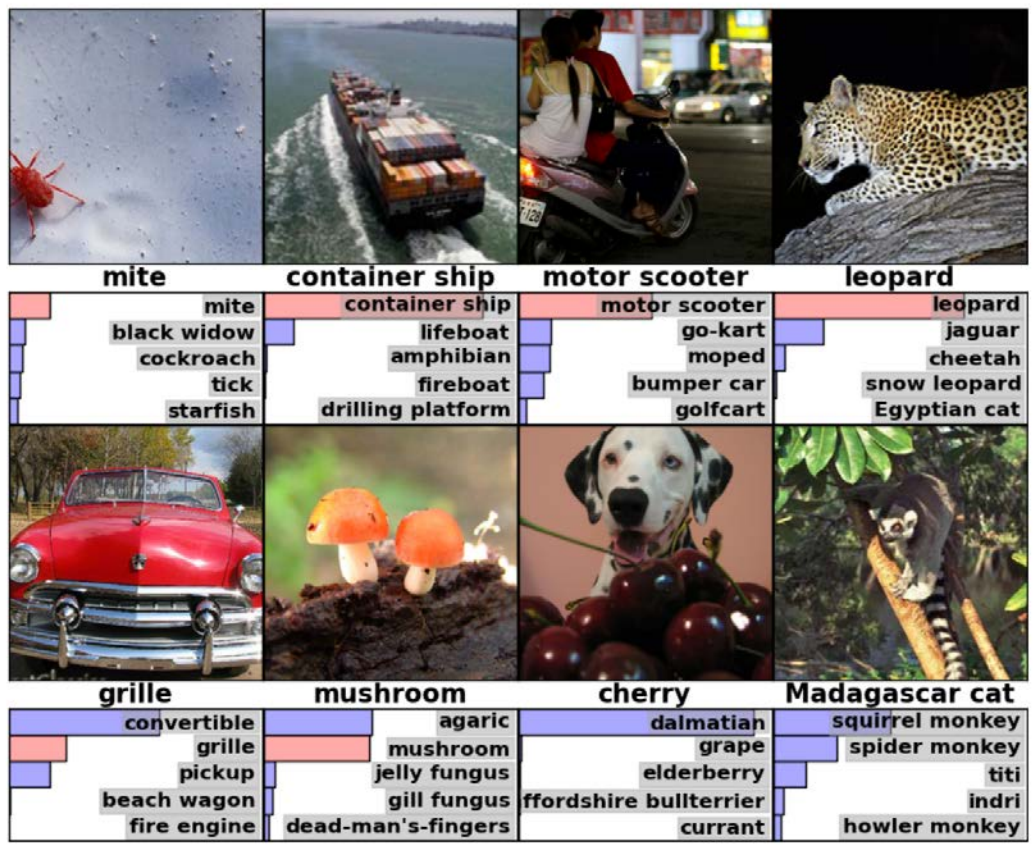
$$\{x_i, y_i\}$$

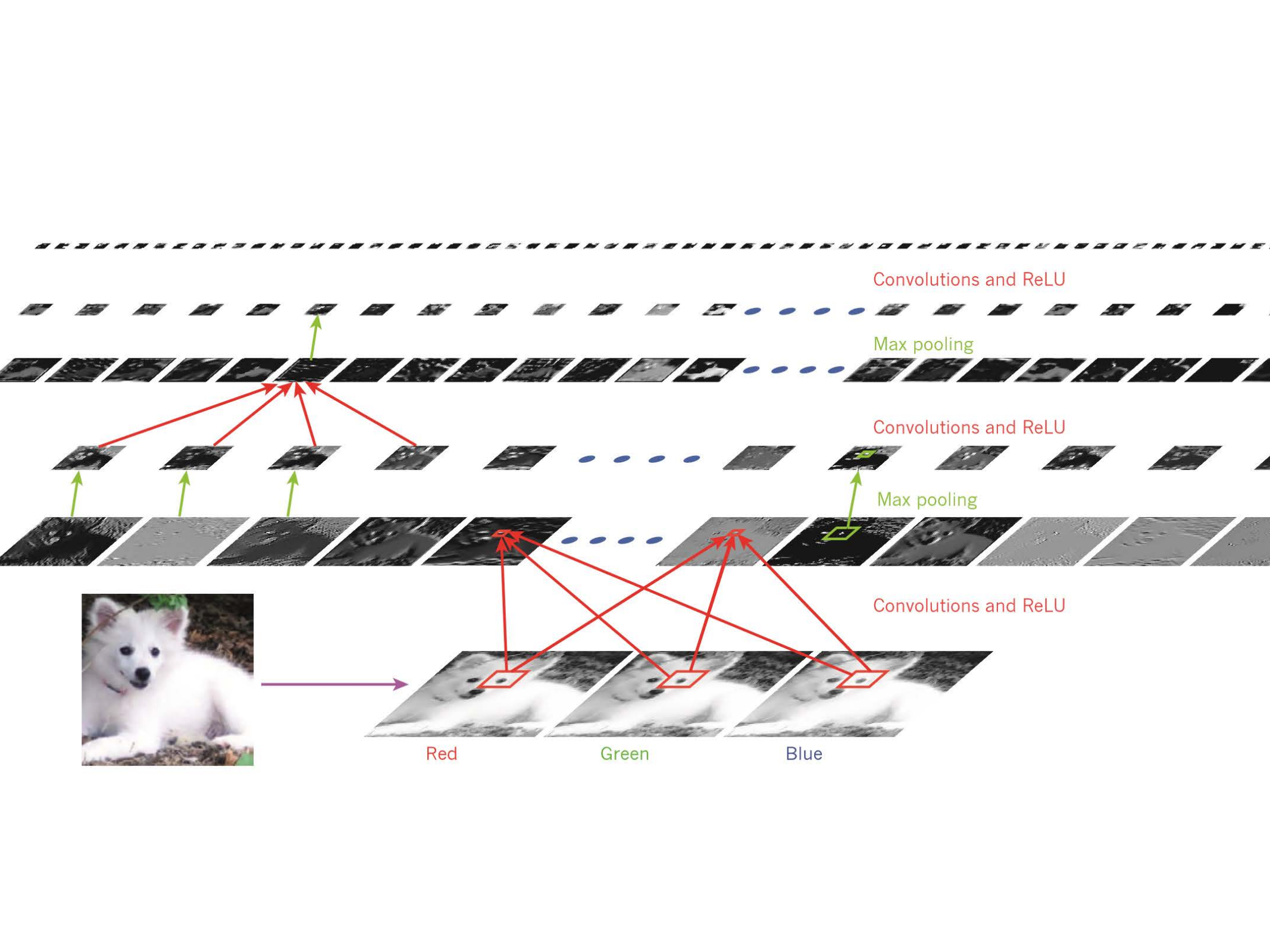
$$\min_w \frac{1}{2}||w||^2 + \sum_i (f_w(x_i) - y_i)^2$$

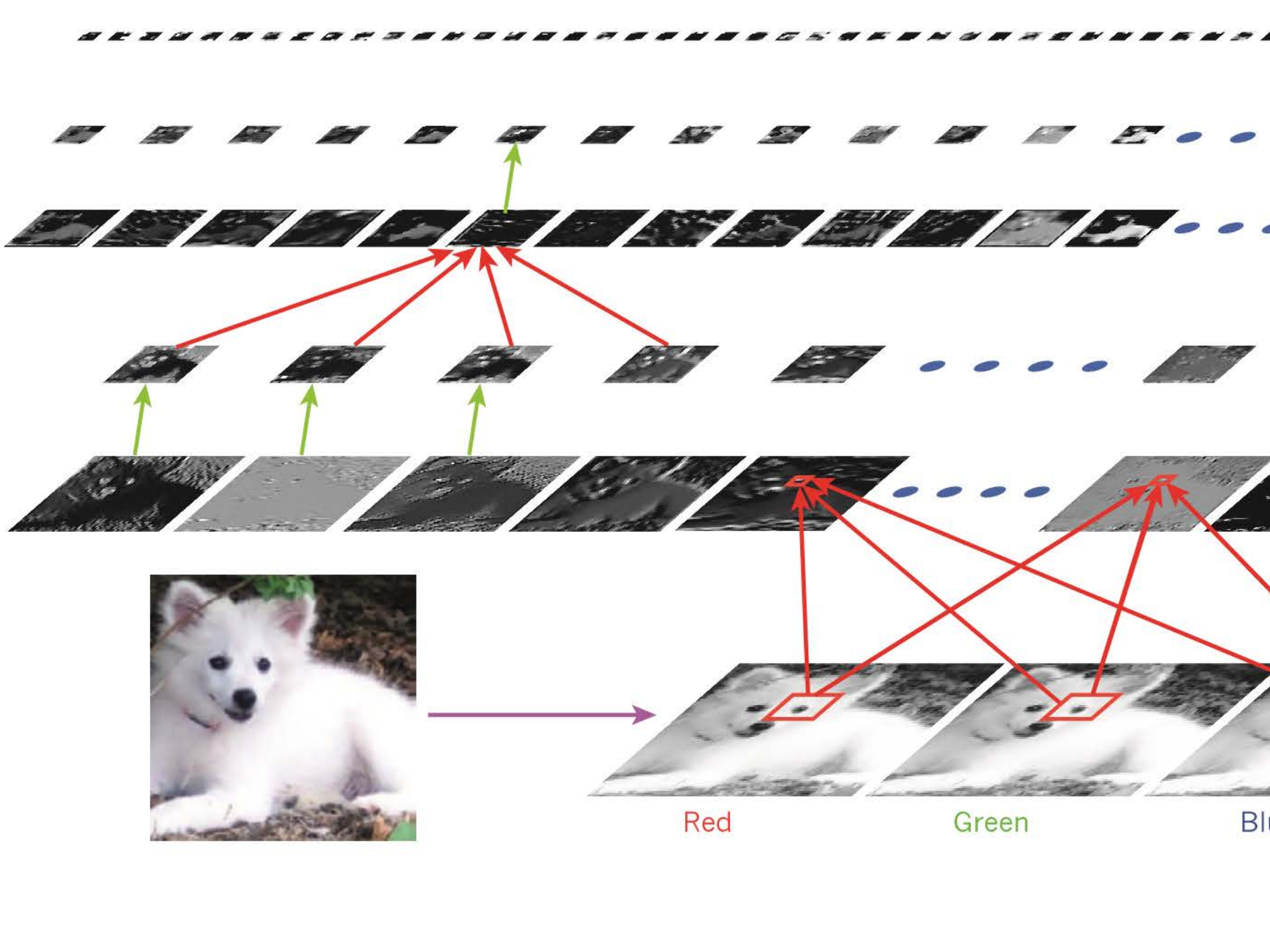
Image Categorization:

AlexNet (2011):

- **Qualitative results:**




Convolutions and ReLU
Max pooling
Convolutions and ReLU
Max pooling
Convolutions and ReLU
Red
Green
Blue


Red
Green

AlexNet (2011):

- **Quantitative results:**

Model	Top-1	Top-5
Sparse coding [2]	*47.1%*	*28.2%*
SIFT + FVs [24]	*45.7%*	*25.7%*
CNN	**37.5%**	**17.0%**

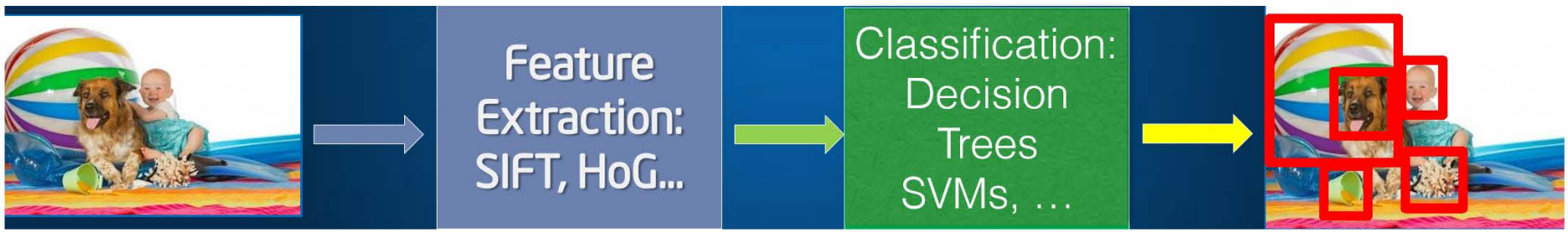
Table 1: Comparison of results on ILSVRC-2010 test set. In *italics* are best results achieved by others.

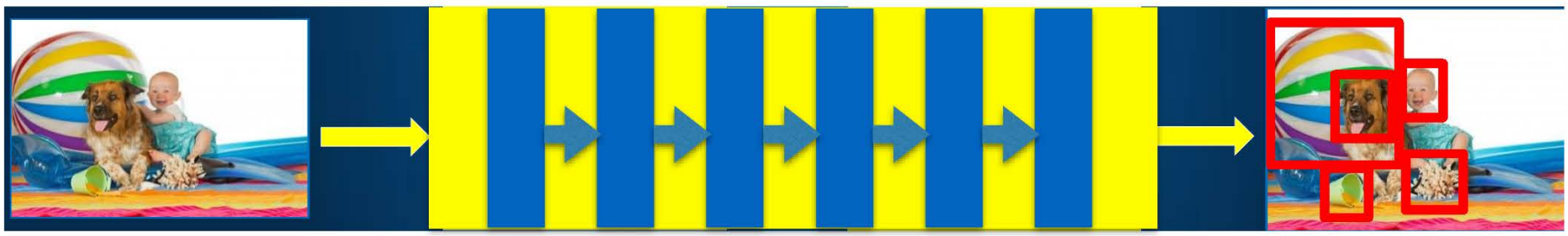
- **Huge Improvement over the previous baselines!**

Imagenet large-scale visual recognition challenge

1000 classes, ~1000 examples per class

Convolutional Neural Network for Image Recognition

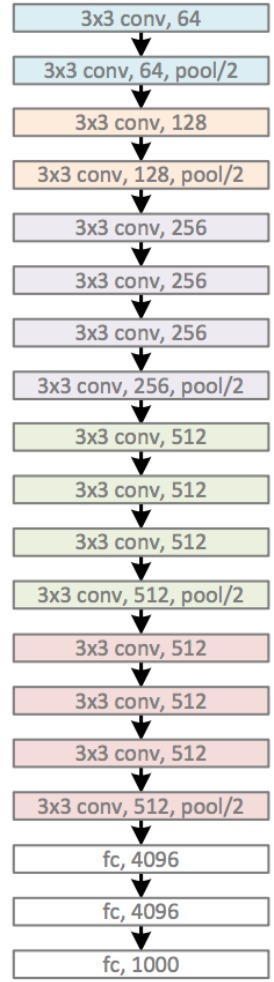




Deep network

VGG Net (2015):

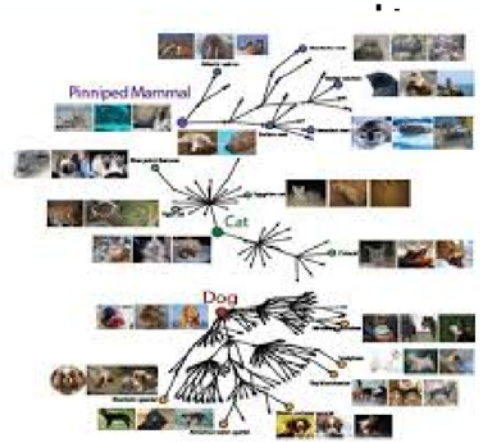
- **Convolutional Neural Network used for image categorization**
- **~138M parameters (19 layers)**
- **Given a 224x224 RGB image, the network predicts one of 1000 image categories.**



ResNet (2016):

- Convolutional Neural Network used for image categorization
- ~160M Parameters (~150 layers)
- Given a 224x224 RGB image, the network predicts one of 1000 image categories.

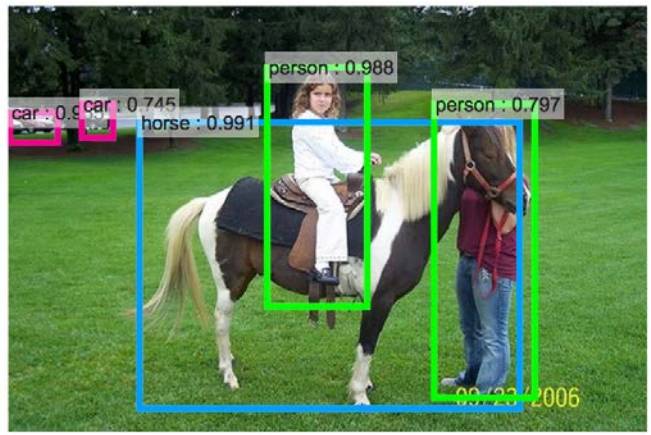
Imagenet 2014 image classification

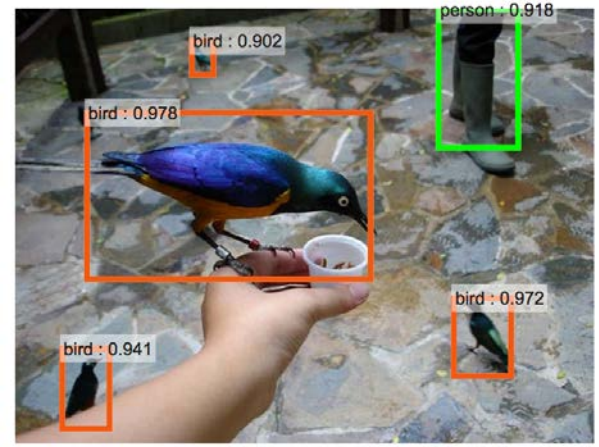


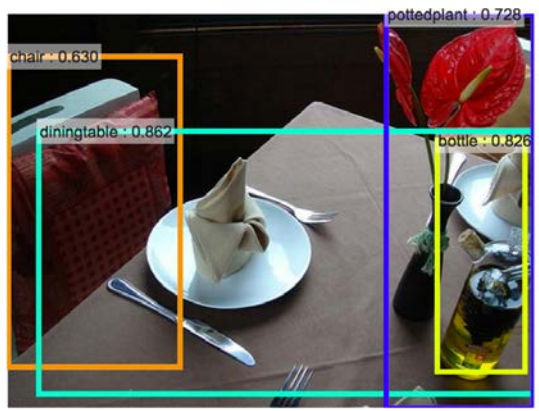
Model	Resolution	Crops	Models	Top-1 error	Top-5 error
GoogLeNet ensemble	224	144	7	-	6.67%
Deep Image low-res	256	-	1	-	7.96%
Deep Image high-res	512	-	1	24.88	7.42%
Deep Image ensemble	variable	-	-	-	5.98%
BN-Inception single crop	224	1	1	25.2%	7.82%
BN-Inception multicrop	224	144	1	21.99%	5.82%
BN-Inception ensemble	224	144	6	20.1%	**4.9%***

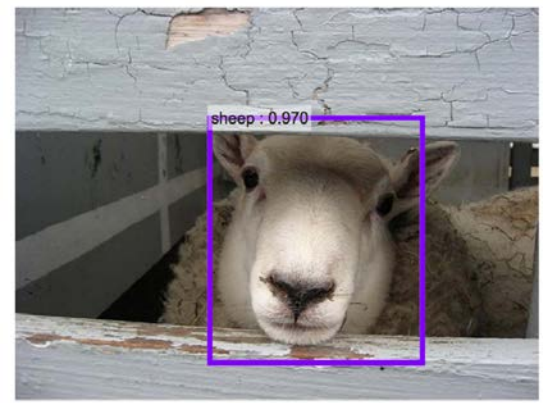
Human top-5 error: 5.1 %

Object Detection:



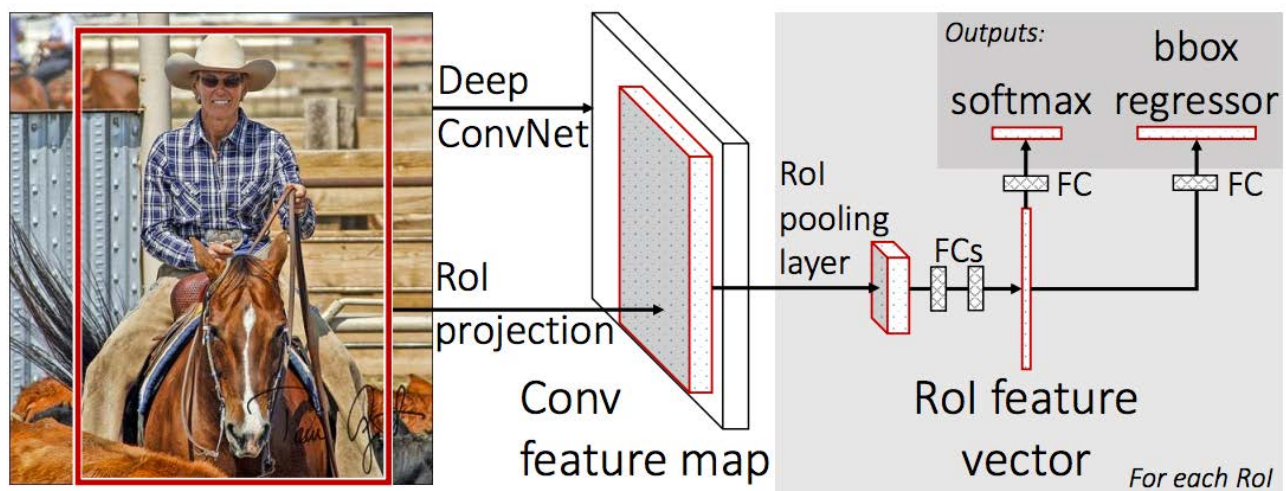






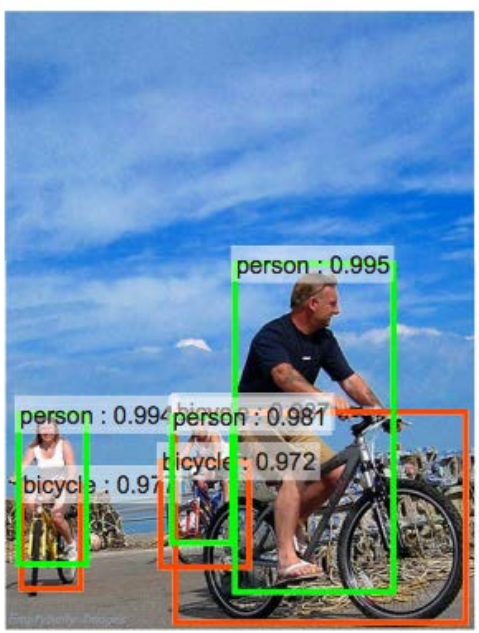
R-CNN (2015):

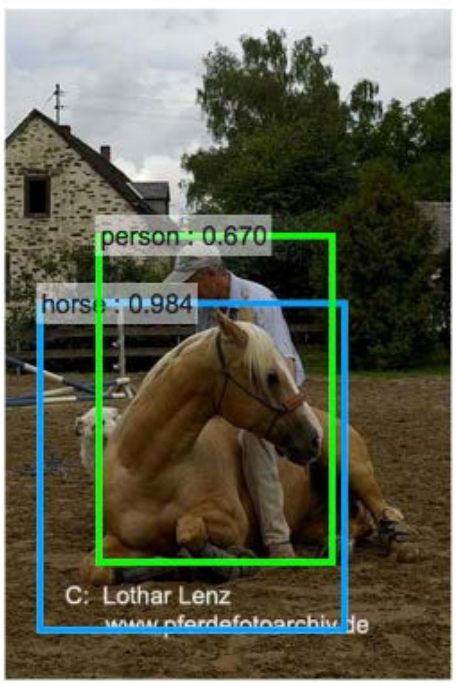
- **Convolutional Neural Network used for object localization**
- **Given an image, the network (1) classifies and (2) localizes objects in the image.**



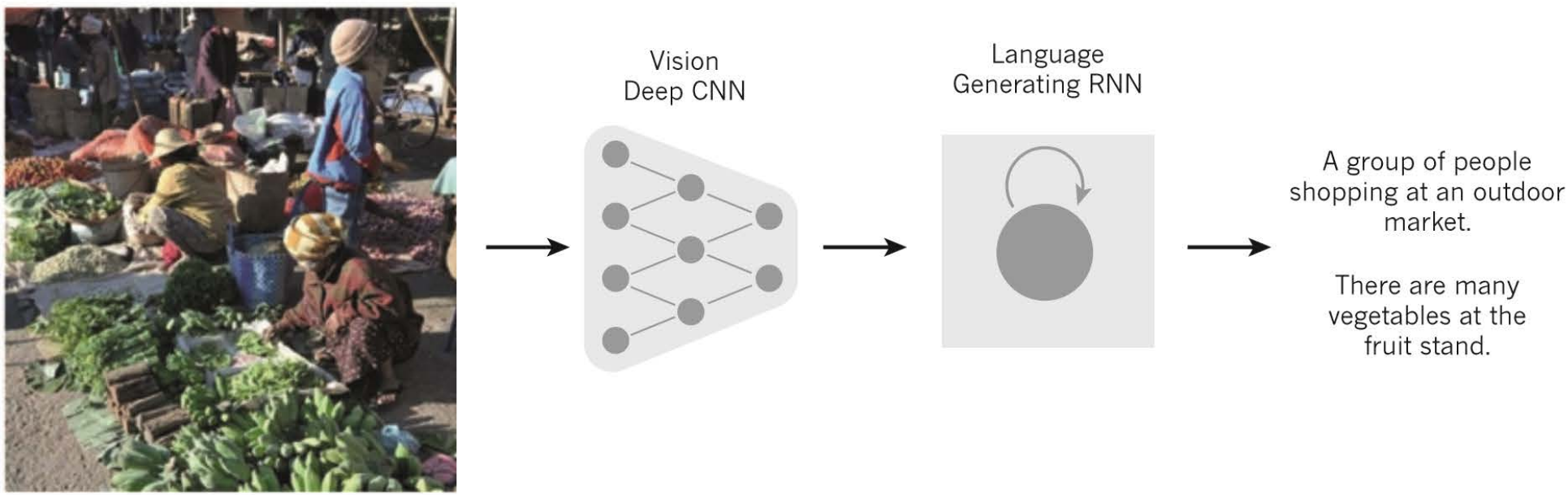
R-CNN (2015):

- **Example results:**





Recurrent Network



A woman is throwing a **frisbee** in a park.

A **dog** is standing on a hardwood floor.

A **stop** sign is on a road with a mountain in the background

A little **girl** sitting on a bed with a teddy bear.

A group of **people** sitting on a boat in the water.

A giraffe standing in a forest with **trees** in the background.

Why now? Open Source

- Deep learning tools are allowing everyone to be experts in analog computing. Lower barrier to entry.
- There are more precise measurements and benchmarks which tell us what are working. (compute)
- **Using standard off-the shelf networks is not that difficult (check out Caffe, Torch, Theano, and other libraries).**

Applications:

- Automatic driving: we can recognize better what is happening
- Shops without checkout lines: we can recognize faces
- Automated delivery, coupled with shopping
- Autograder: for mass online teaching
- Financial: text analysis
- Medical: