Introduction to Profile Areas in Bioinformatics

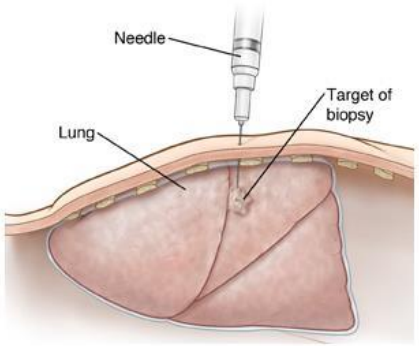
Lecturer: Kathleen Gallo

Session 3

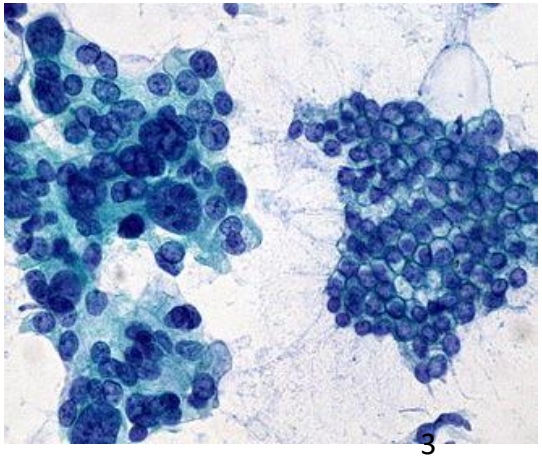
- kathleen.gallo@online.de
- Source code and report via mail until 13.11., 18h
- No peer review
- Tasks will be evaluated by me
- Short feedback at the beginning of the lecture on 8.11.

LAST WEEK's Project Overview

- Imagine, you are given data about patients having some suspicious tissue, e.g. in lung or breast.
- Features are computed from a digitized image of a fine needle aspirate (FNA) tissue. They describe characteristics of the cell nuclei present in the image.
- You want to build a tool that can help medical doctors to decide whether a sample is benign or malign.
- How would you do that?

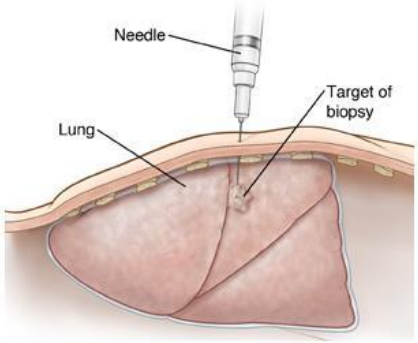


For the biopsy, a thin needle is used to remove samples of tissue from an abnormal area in the lung.

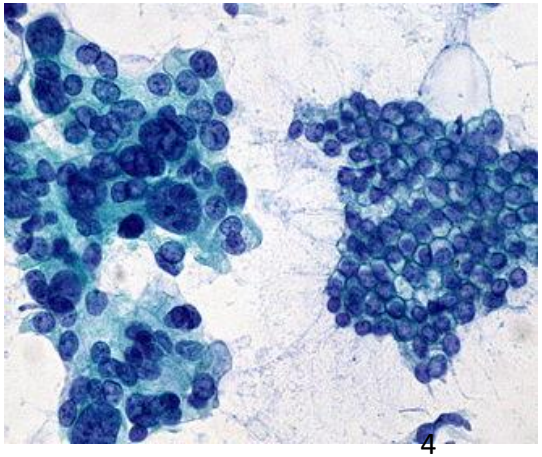


THIS WEEK's Project Overview

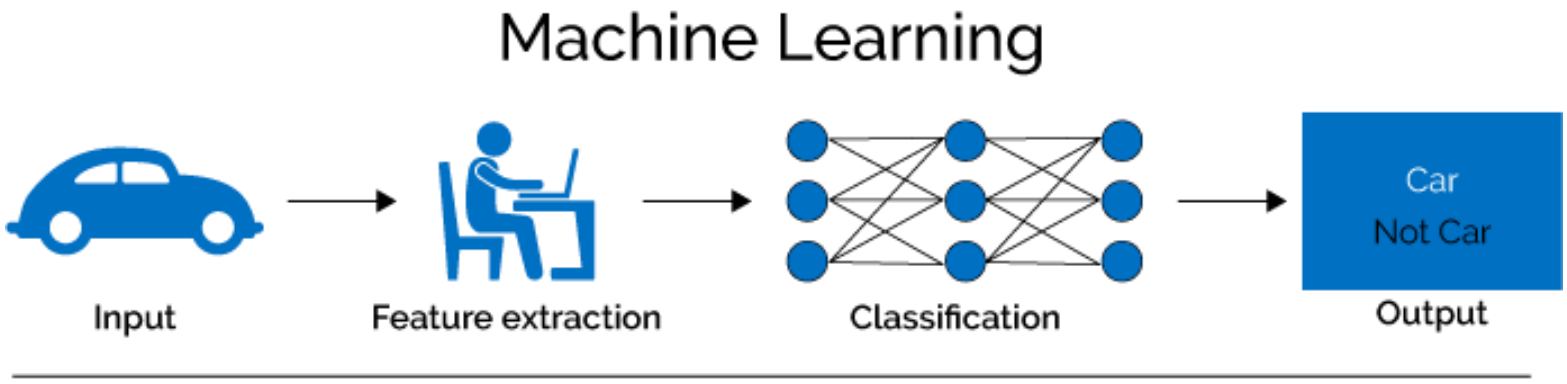
- Imagine, you are given data about patients having some suspicious tissue, e.g. in lung or breast.
- ~~Features are computed from~~ a digitized image of a fine needle aspirate (FNA) tissue. ~~They describe characteristics of the cell nuclei present in the image.~~
- You want to build a tool that can help medical doctors to decide whether a sample is benign or malign.
- How would you do that?



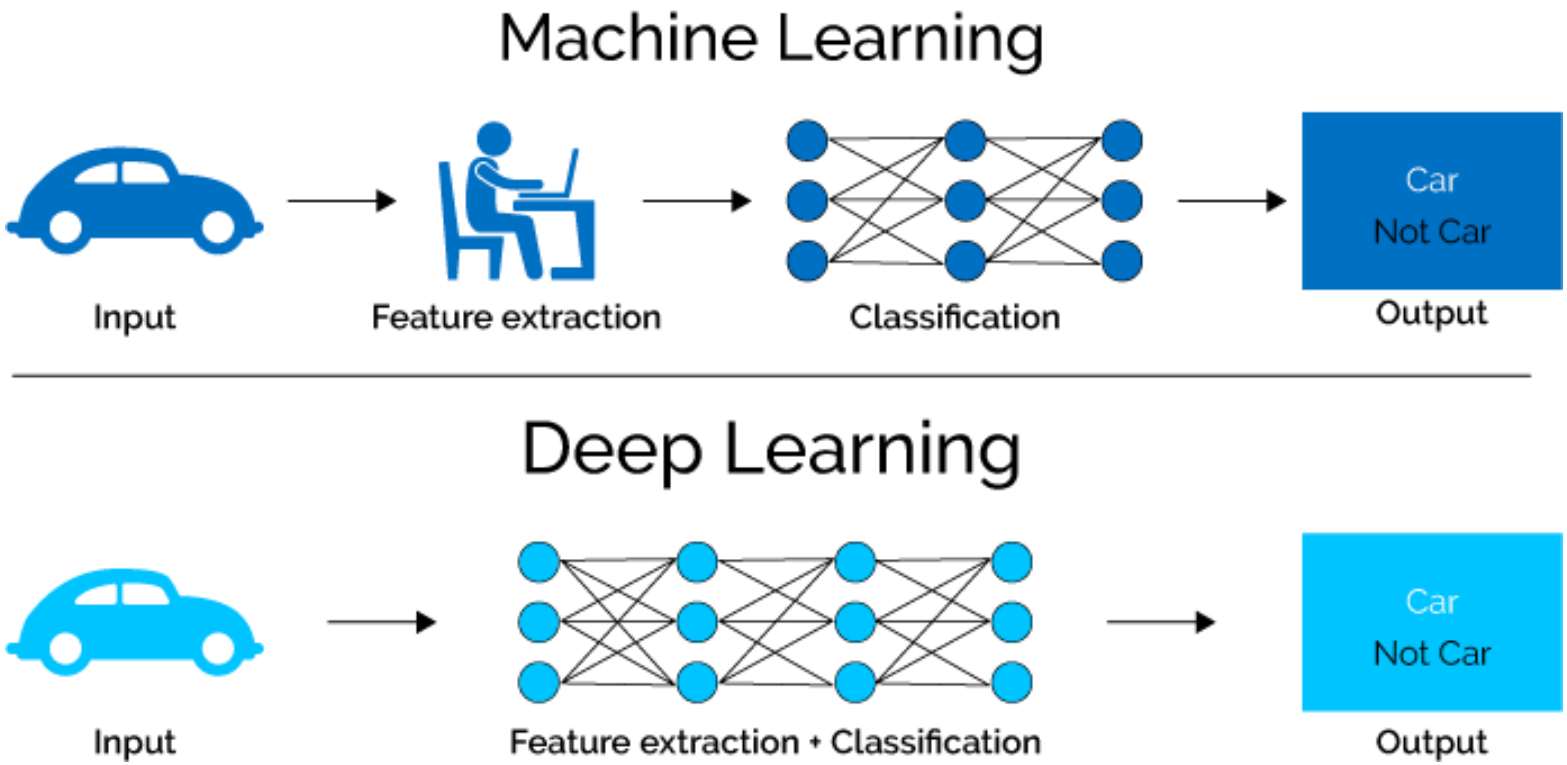
For the biopsy, a thin needle is used to remove samples of tissue from an abnormal area in the lung.



Two main approaches



Two main approaches



THIS WEEK's Project Overview

- The Data:
  - LAST WEEK: Breast Cancer Wisconsin (Diagnostic) Data Set
  - THIS WEEK: A data-set containing microscopy images
- The task: develop a classifier to diagnose the state of a sample.
  - LAST WEEK: Using extracted features
  - THIS WEEK: Using the images directly
- What to deliver: The source code and a report.

Deep learning

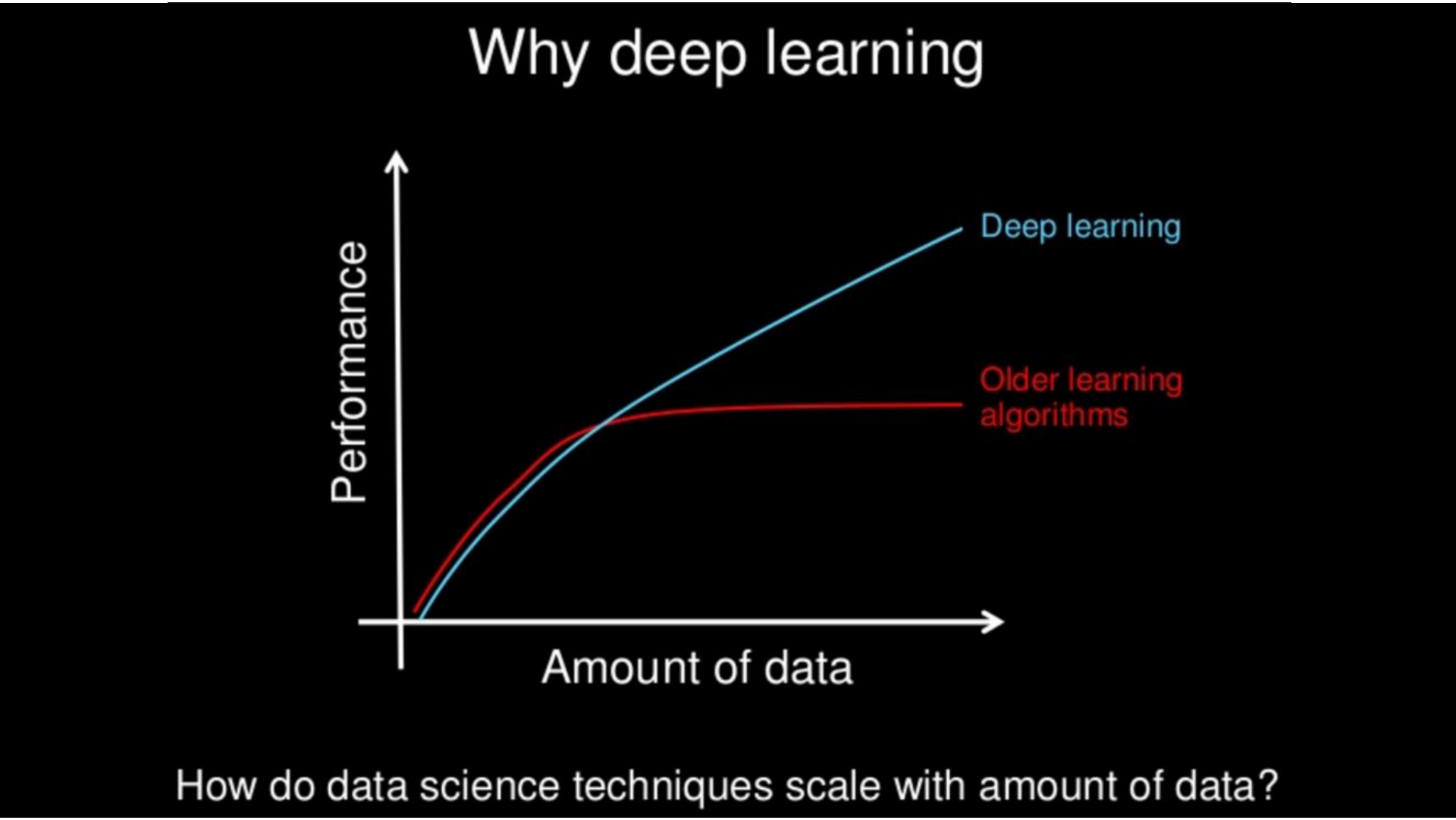
From Wikipedia, the free encyclopedia

„… is a branch of machine learning […]“

“… has been characterized as a buzzword, or a rebranding of neural networks.”

“… is often presented as a step towards realizing strong [generalized] AI […]”

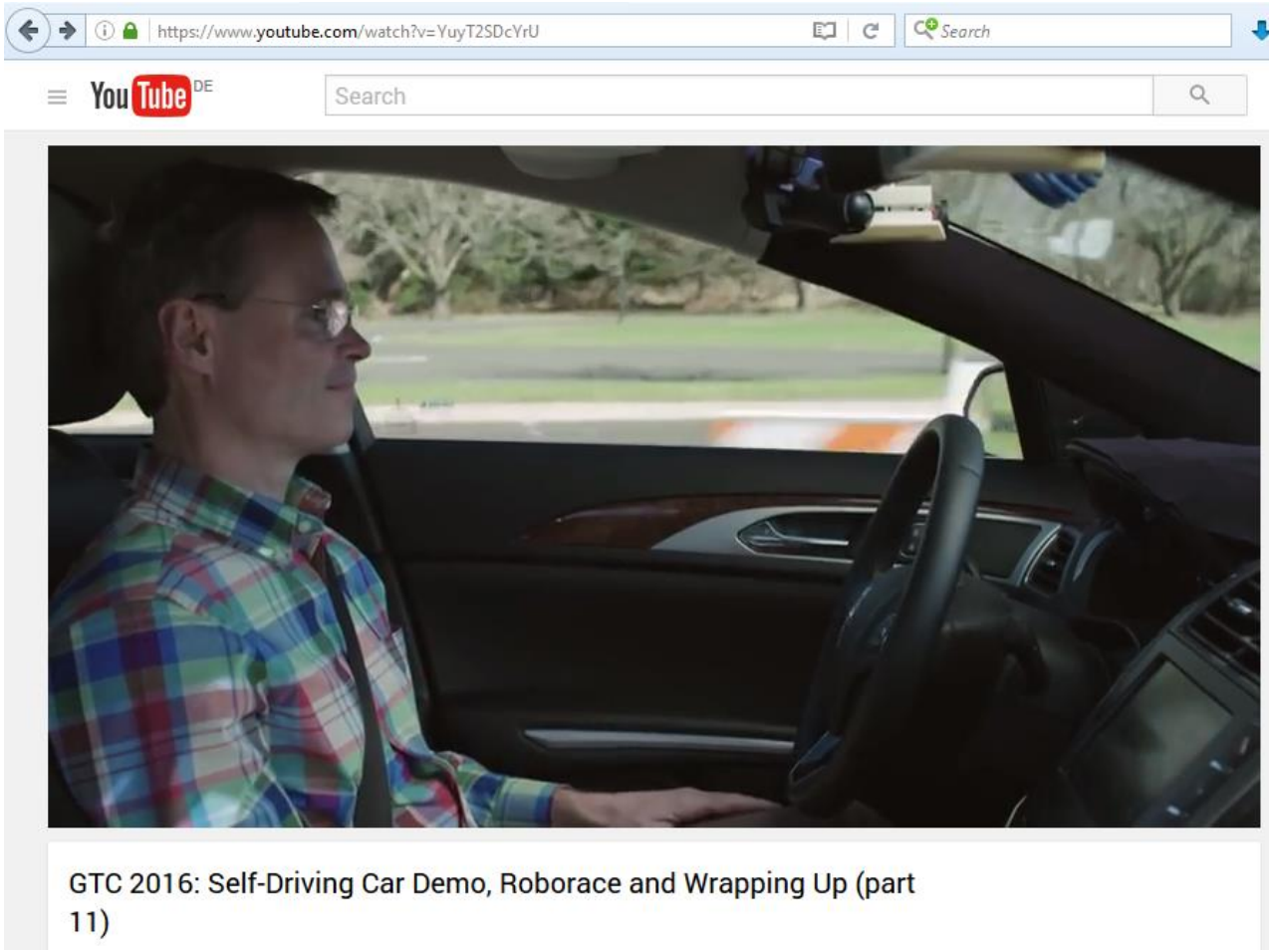
Deep Learning



- "There's a distinction […] between **generalized AI** and **specialized AI**.
- In science fiction, what you hear about is **generalized AI**, right? Computers start getting smarter than we are […]
- My impression, based on talking to my top science advisers, is that we're still a reasonably long way away from that."

https://www.wired.com/2016/10/president-obama-mit-joi-ito-interview/

Driving a Car – nothing but: **y=f(x)**?



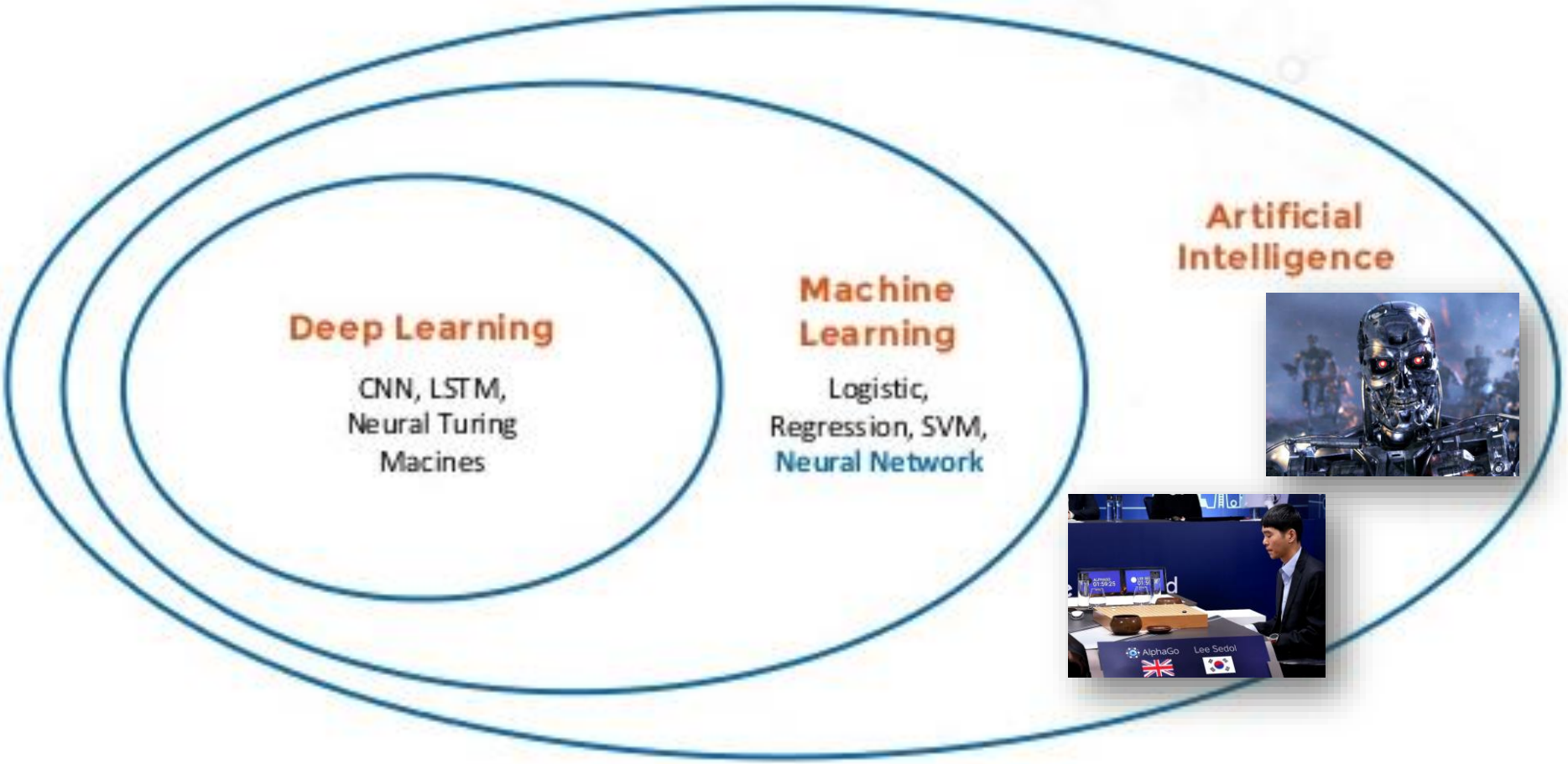
https://www.youtube.com/watch?v=YuyT2SDcYrU

- "[…] **specialized AI** […] is about using algorithms and computers to figure out increasingly complex tasks.

- We've been seeing specialized AI in every aspect of our lives, from medicine and transportation to how electricity is distributed, […]"

https://www.wired.com/2016/10/president-obama-mit-joi-ito-interview/

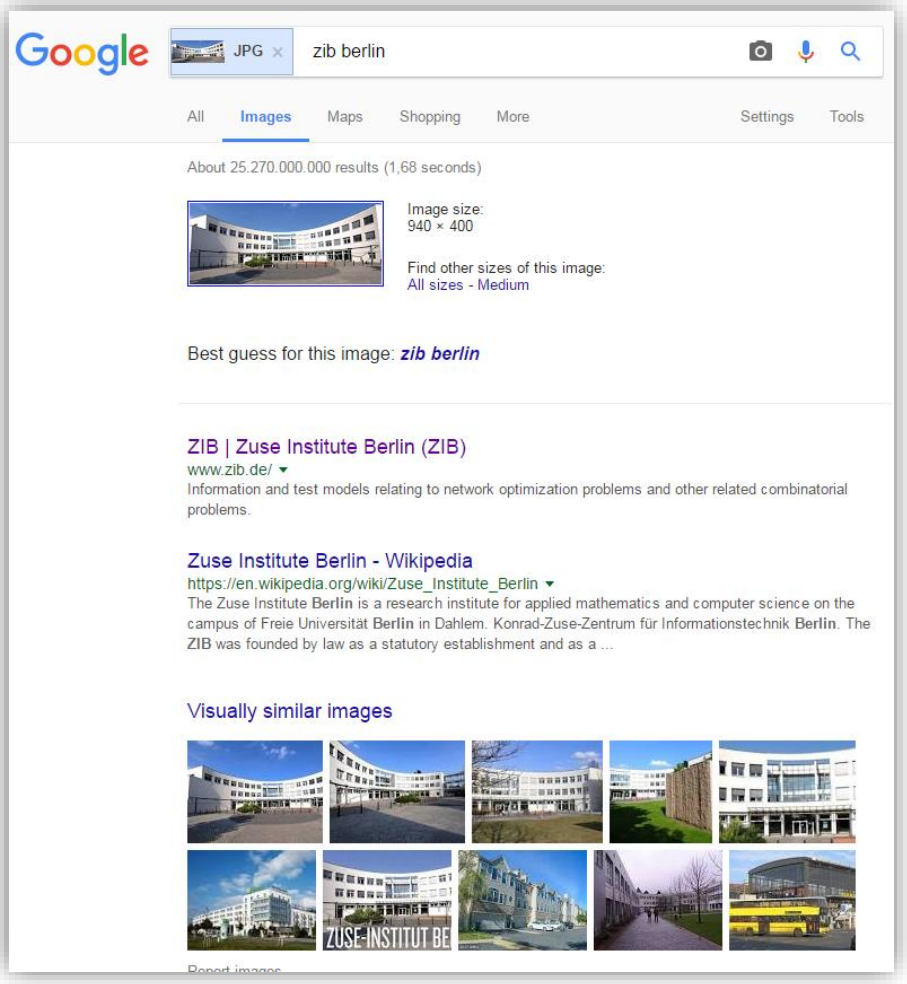
From AI to Deep Learning



Deep Learning for AI

The methods we are going to talk about today are used by several companies for a variety of applications, such as classification, retrieval, detection, etc.

It is especially well established in the area of computer vision.



Deep Learning for (specialized) AI

clarifai Demo Solutions Pricing Developer API Resources Get your free API key Talk to us

GENERAL MODEL

PREDICTED CONCEPT	PROBABILITY
architecture	0.997
contemporary	0.987
modern	0.984
facade	0.984
house	0.979
horizontal plane	0.979
luxury	0.977
window	0.976
no person	0.970
outdoors	0.960
sky	0.957
lease	0.945
building	0.942

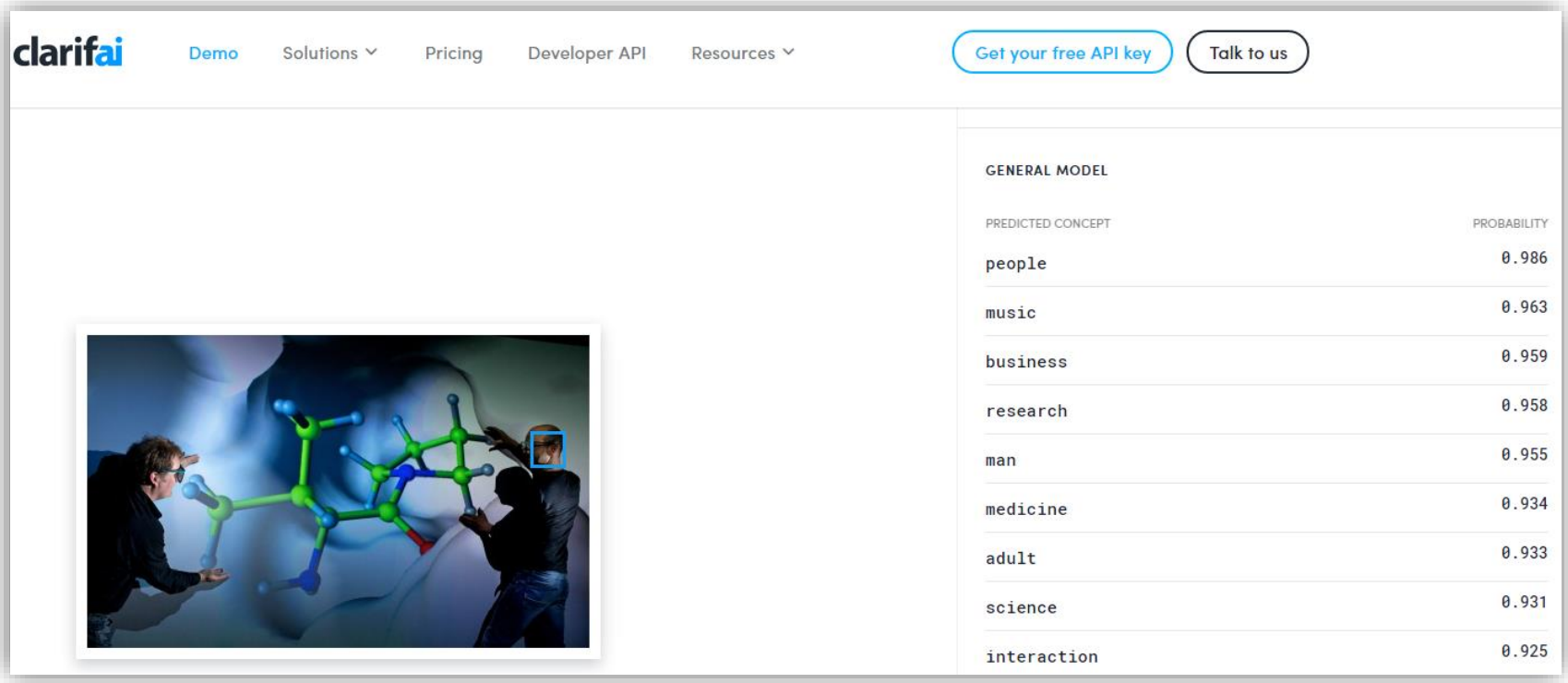
Deep Learning for (specialized) AI

clarifai Demo Solutions Pricing Developer API Resources Get your free API key Talk to us

GENERAL MODEL

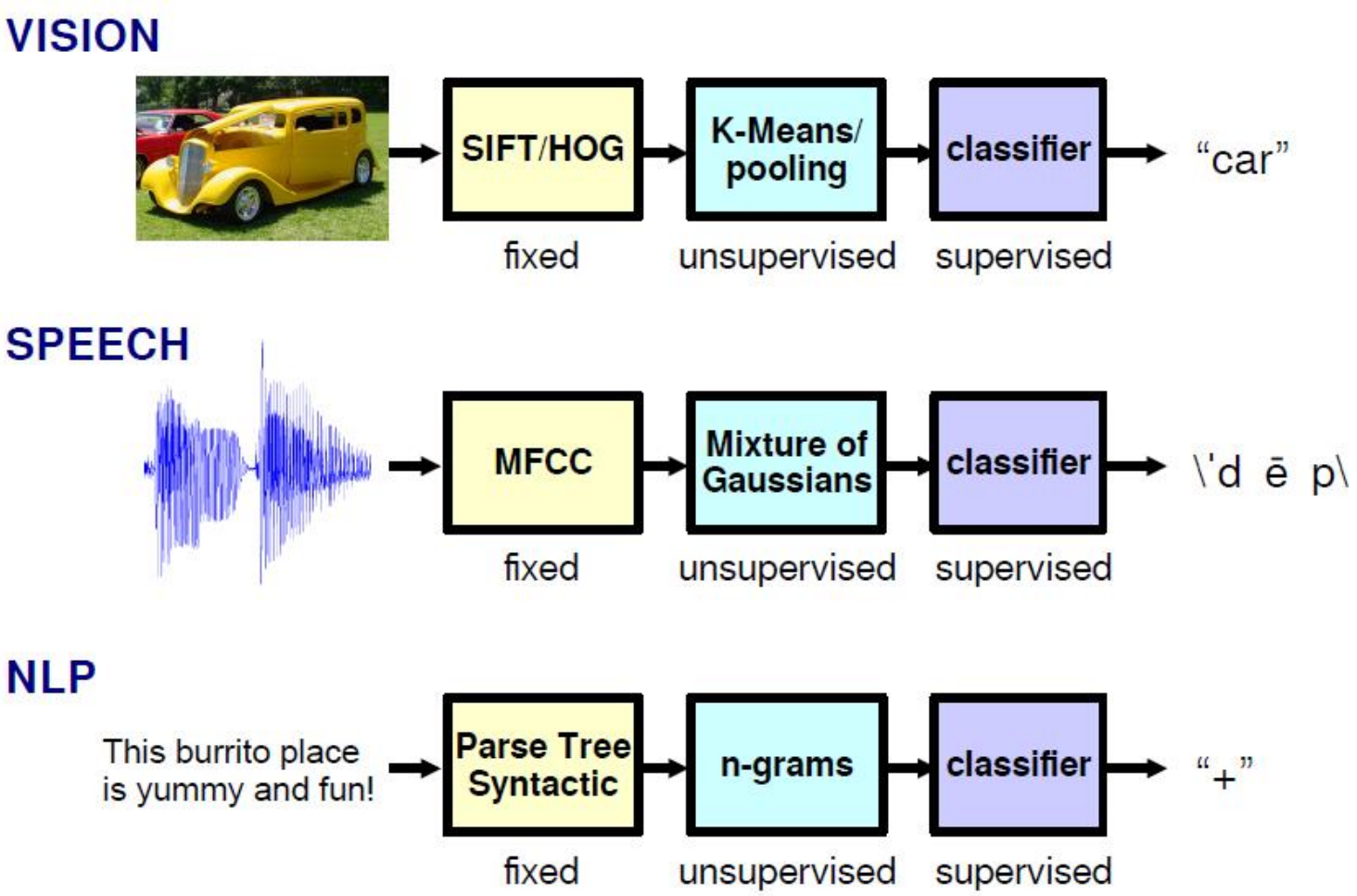
PREDICTED CONCEPT	PROBABILITY
adult	0.990
healthcare	0.986
hospital	0.982
man	0.980
woman	0.978
people	0.973
surgeon	0.973
nurse	0.962
medicine	0.959
portrait	0.953
medical practitioner	0.943
education	0.942
teamwork	0.941

Deep Learning for (specialized) AI



TRADITIONAL VS HIERARCHICAL LEARNING

Traditional Pattern Recognition



Hierarchical Compositionality (DEEP)

$$f(x) \approx g_1(g_2(\ldots g_n(x)\ldots))$$

VISION

pixels → edge → texton → motif → part → object

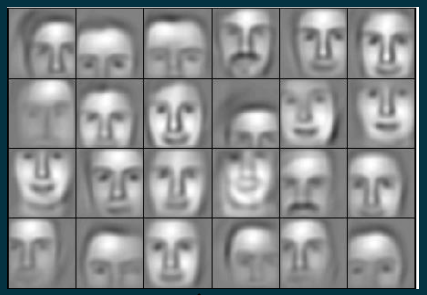
SPEECH

sample → spectral band → formant → motif → phone → word

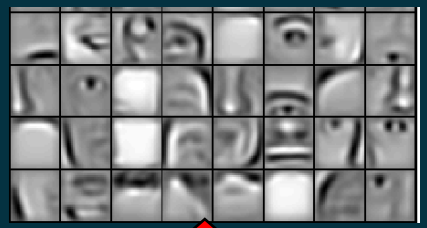
NLP

character → word → NP/VP/.. → clause → sentence → story

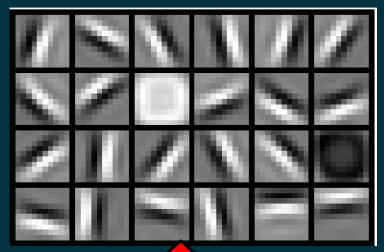
Deep Learning: learn hierarchical representations



object models



object parts (combination of edges)

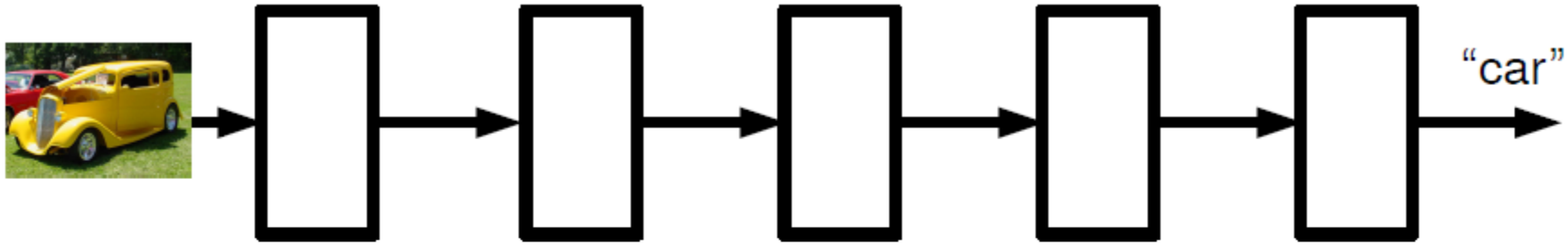


edges

pixels

$$f(x) \approx g_1(g_2(\ldots g_n(x)\ldots))$$

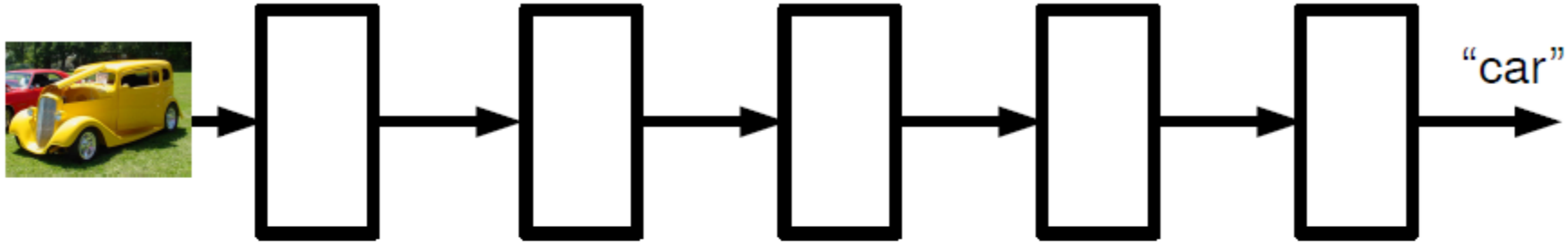
Deep Learning



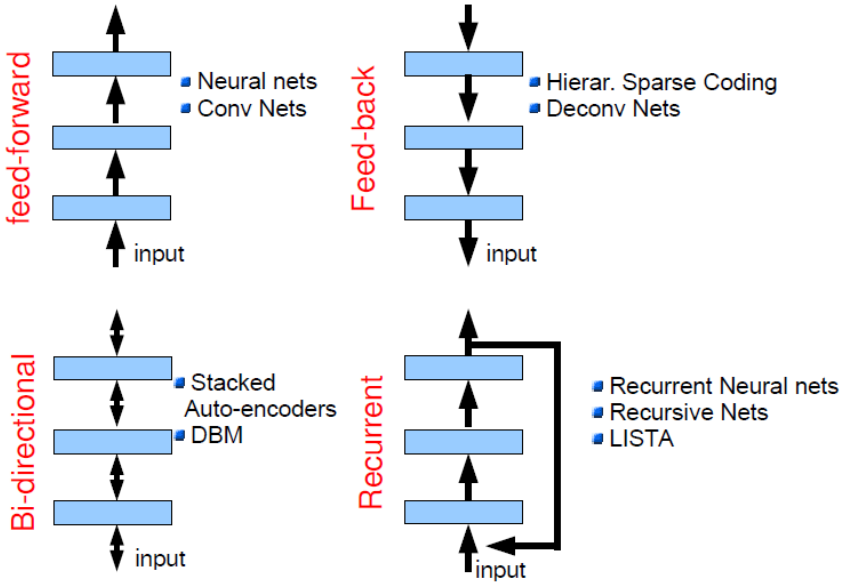
What is Deep Learning

- Cascade of non-linear transformations
- End to end learning
- General framework (any hierarchical model is deep)

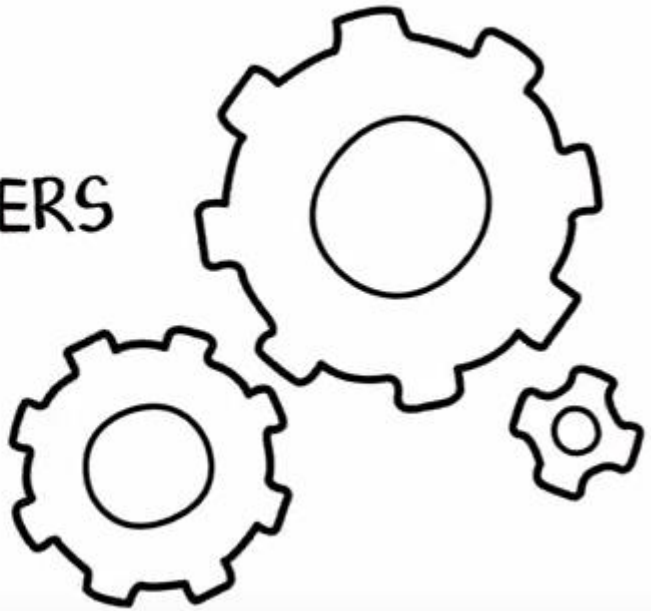
Deep Learning



There are many architectures and flavors.

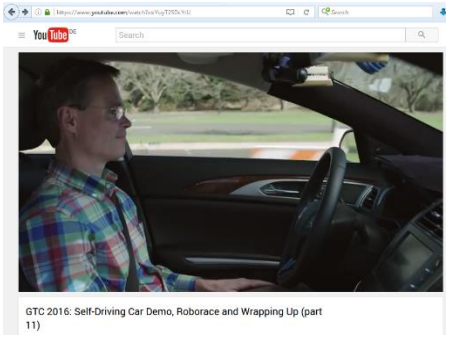


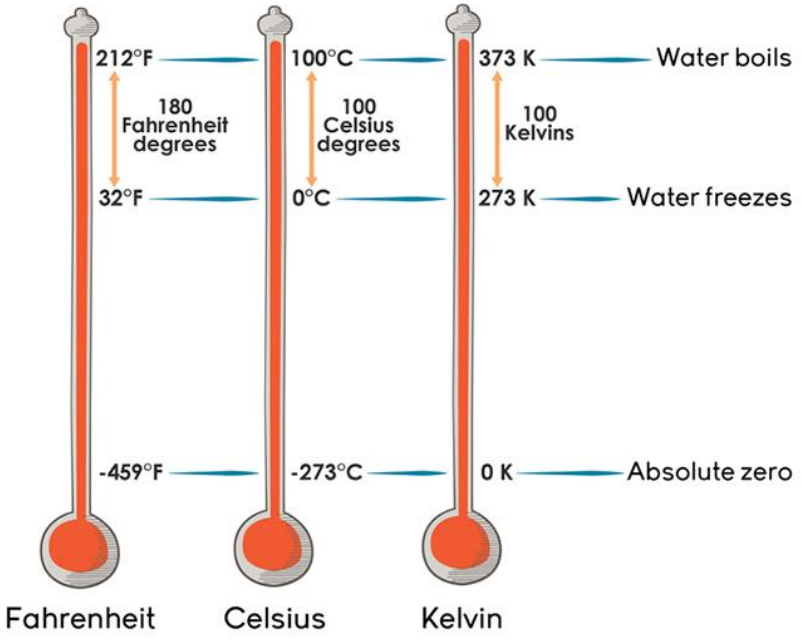
MACHINE LEARNING IS A METHOD OF TEACHING COMPUTERS TO MAKE PREDICTIONS BASED ON SOME DATA.



Machine Learning: Find the Function

X → Y = f(X) → Y





Model-based vs. Data-driven
(or „deductive“ vs „inductive“)

Classical (Deductive) Approach

Example: Converting Celsius to Fahrenheit

Requirements

Input: C
Output: F
Where F is Fahrenheit equivalent of C in Celsius

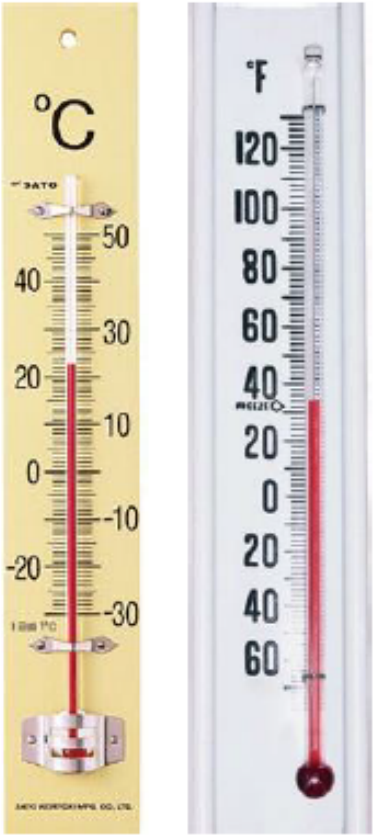
Model

F = 1.8 * C + 32

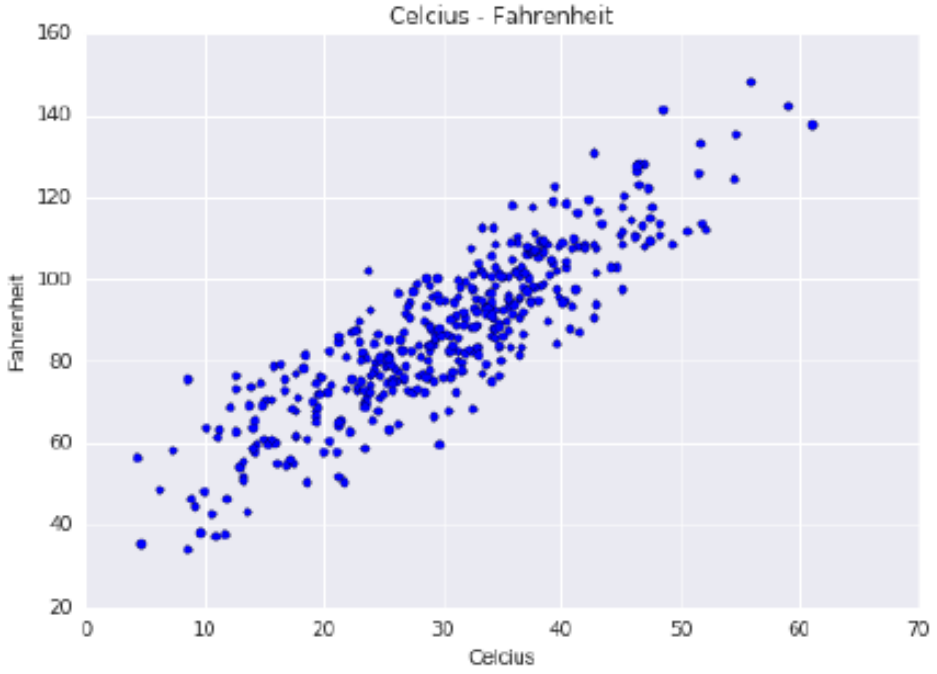
Implementation

```
public class C2F {
	public static void main(String[] args) {
		java.util.Scanner sc = new java.util.Scanner(System.in);
		double c = sc.nextDouble();
		double f = 1.8 * c + 32.0;
		System.out.println("F = " + f);
		sc.close();
	}
}
```

Alternative: Data-driven, Inductive Approach

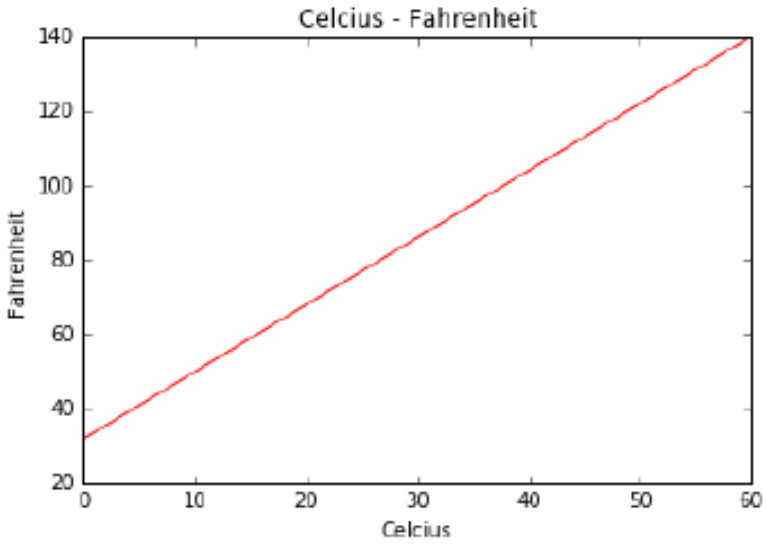


"Find function that represents this data"



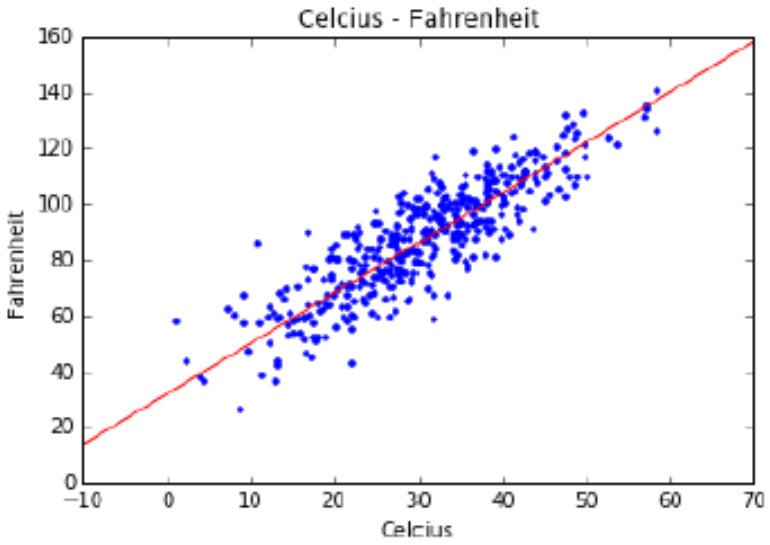
Machine Learning does exactly this

Definitive Function



$F = 1.8 * C + 32$

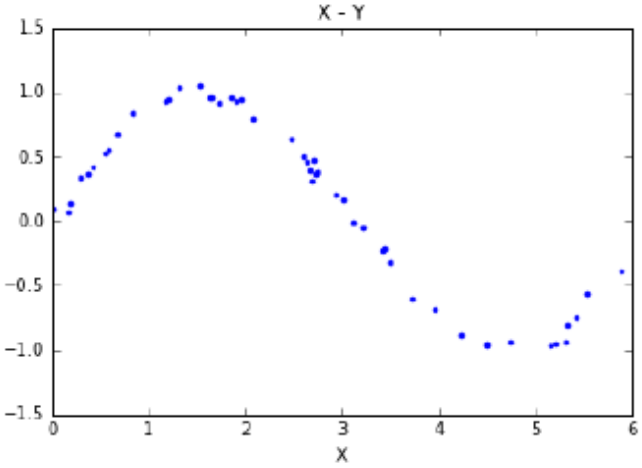
Stochastic Function



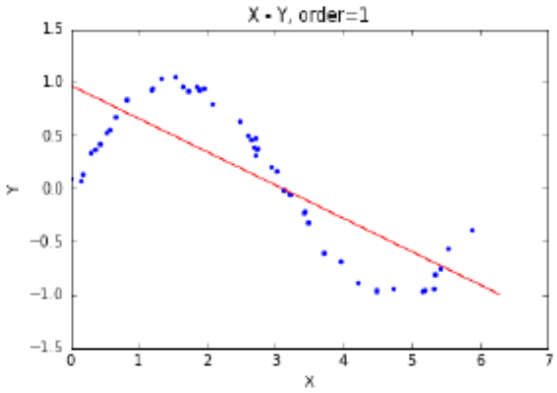
$F = 1.8 * C + 32 + e$
$e \sim N(0,10)$

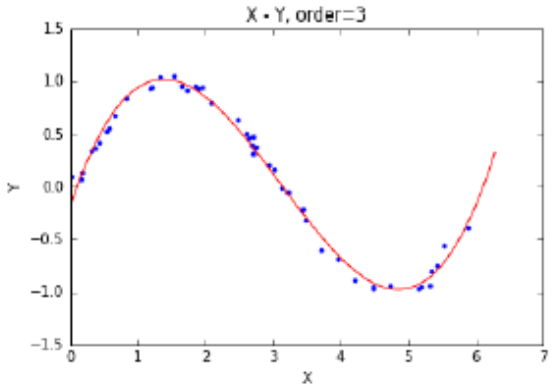
In (classical) Machine Learning, Model must be chosen a-priori

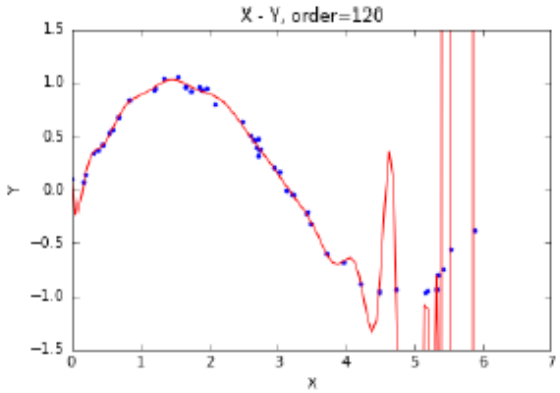
What is the function that represents this data?



Too many parameters result in overfitting!

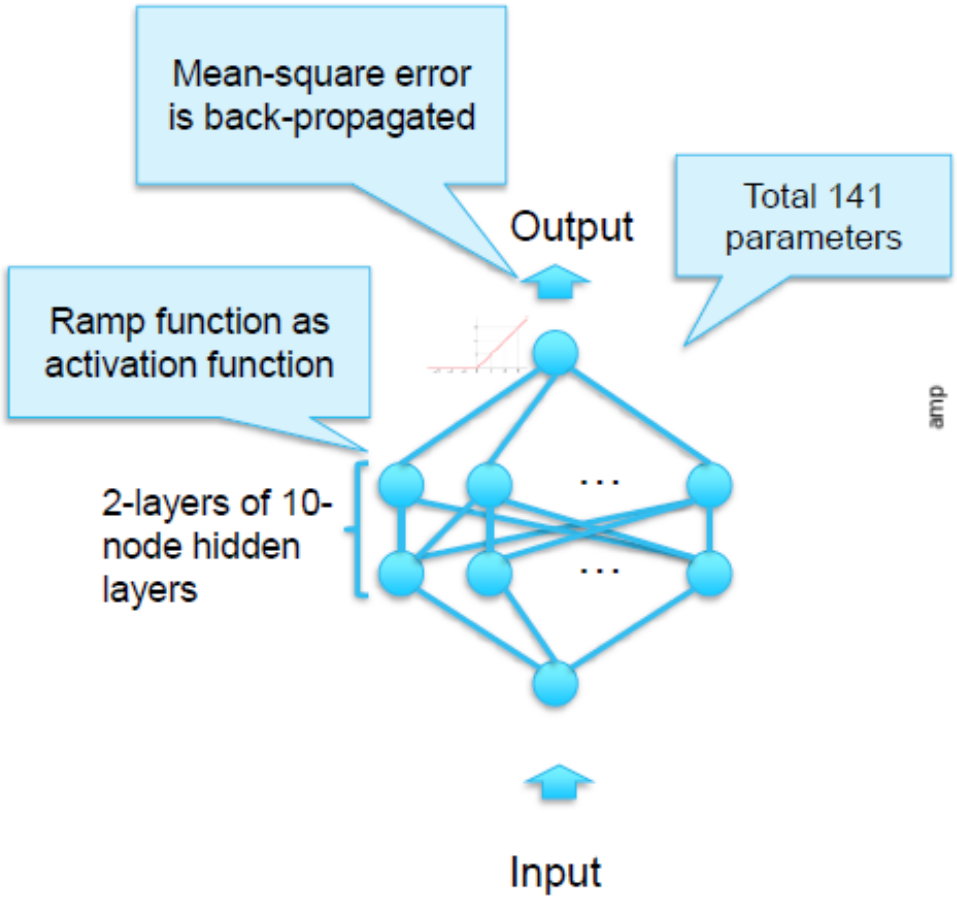


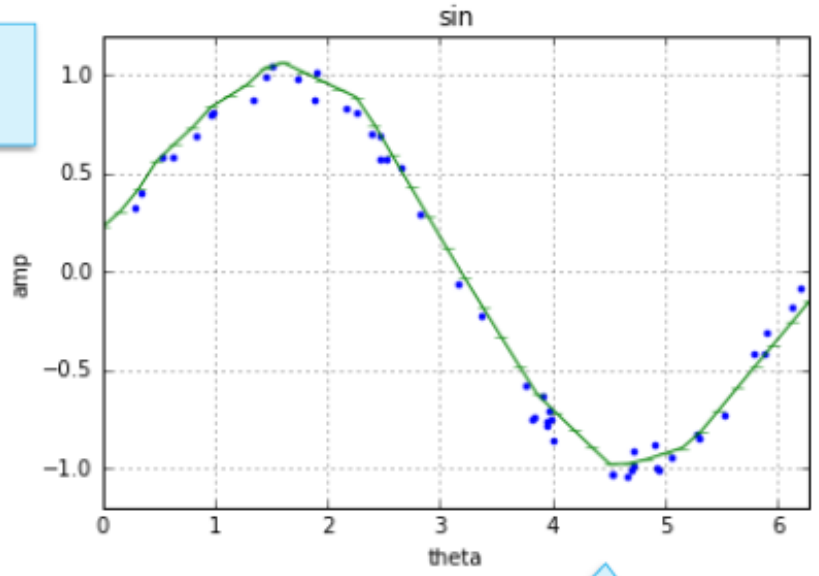




Choosing the right model is the key

Deep Learning can approximate any function without too much overfitting (mostly)





Good approximation without too much overfitting

So, 1. **what exactly is deep learning** ?

And, 2. **why is it generally better** than other methods on image, speech and certain other types of data?

The short answers

So, 1. **what exactly is deep learning** ?

'Deep Learning' means using a **neural network** with **several layers of nodes** between input and output

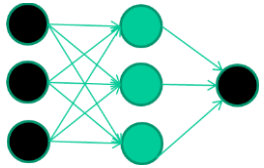
And, 2. **why is it generally better** than other methods on image, speech and certain other types of data?

The series of layers between input & output do feature identification and processing in a series of stages, just as our brains seem to.

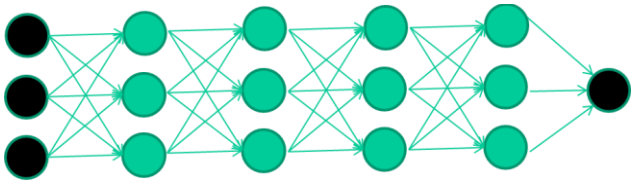
Multilayer neural networks have been around for 25 years. What's actually new?

We have always had good algorithms for learning the weights in networks with few (1) hidden layer

But these algorithms are not good at learning the weights for networks with more hidden layers

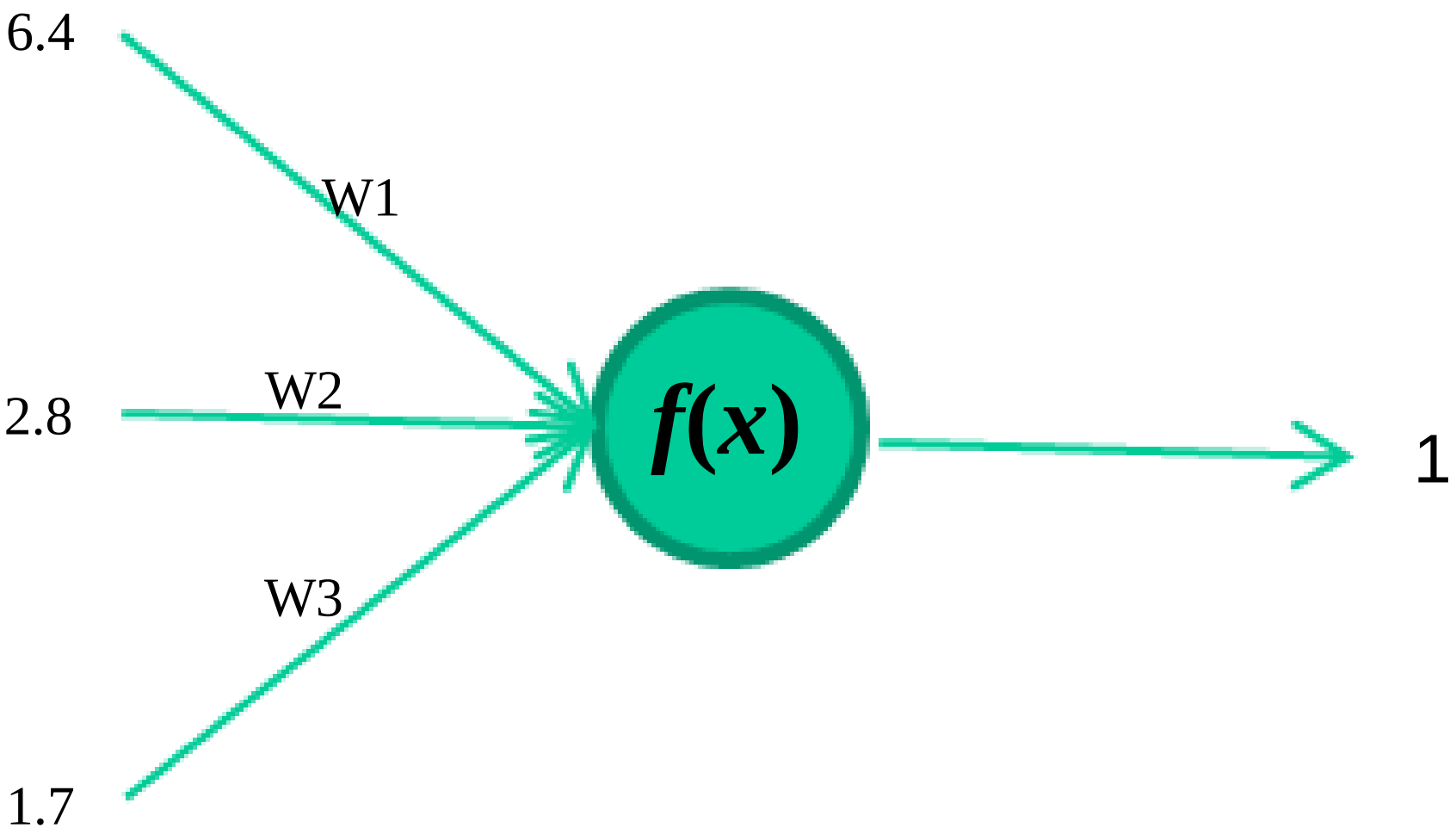


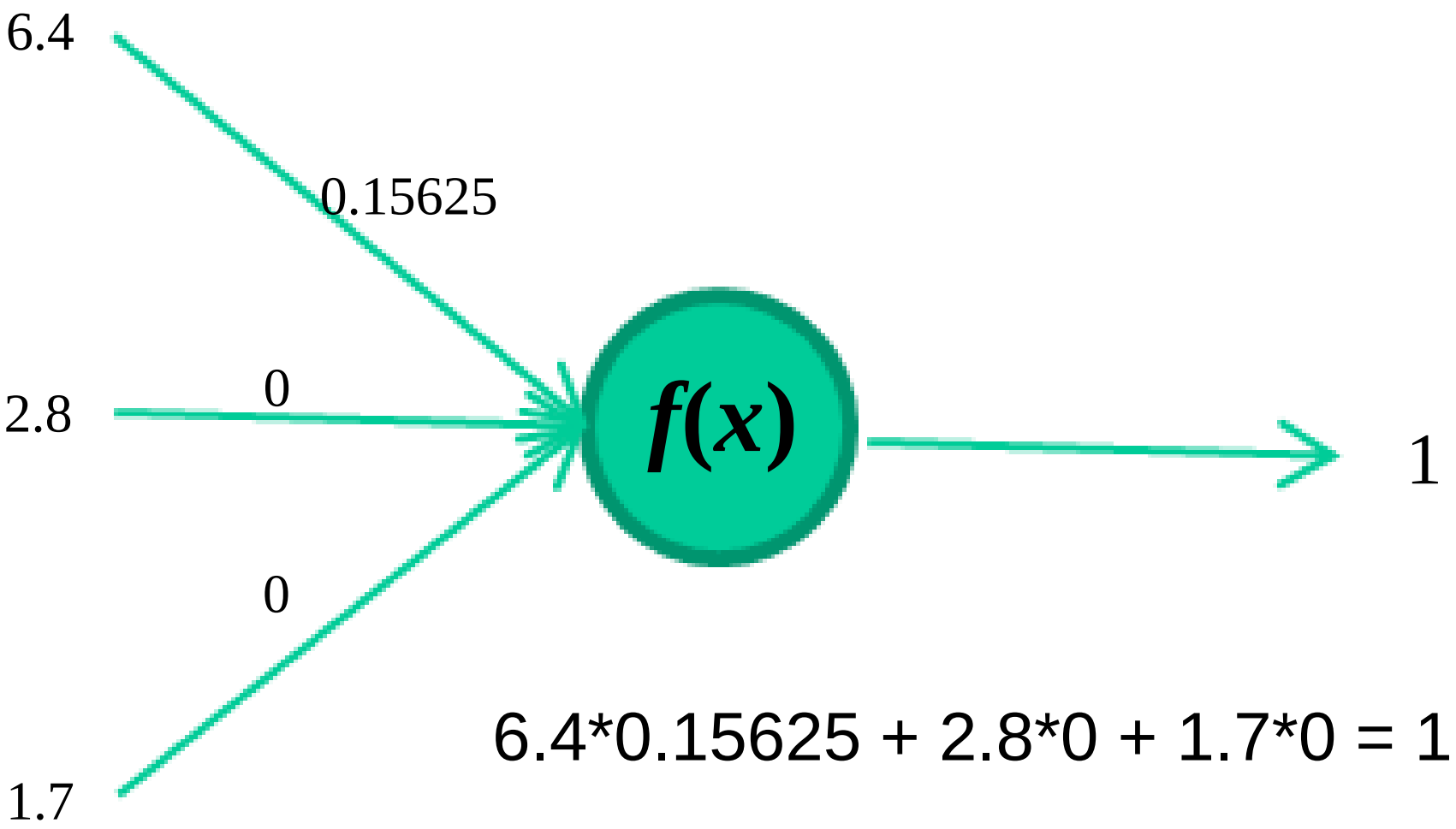
What's new is:
algorithms for training many-layer networks



Longer Answer

1. reminder/quick-explanation of how neural network weights are learned;
2. the idea of **unsupervised feature learning** (why 'intermediate features' are important for difficult classification tasks, and how NNs seem to naturally learn them)
3. The 'breakthrough' – the simple trick for training Deep neural networks

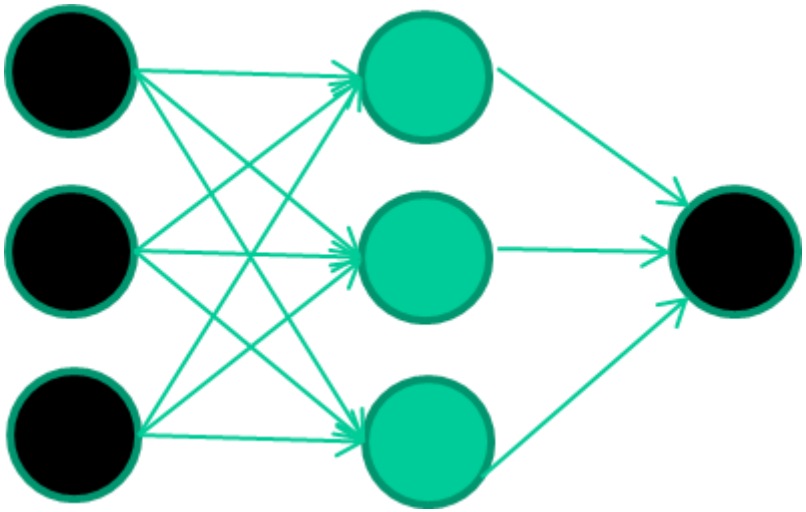

6.4
W1
2.8
W2
f(x)
1
W3
1.7



6.4*0.15625 + 2.8*0 + 1.7*0 = 1

A dataset

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

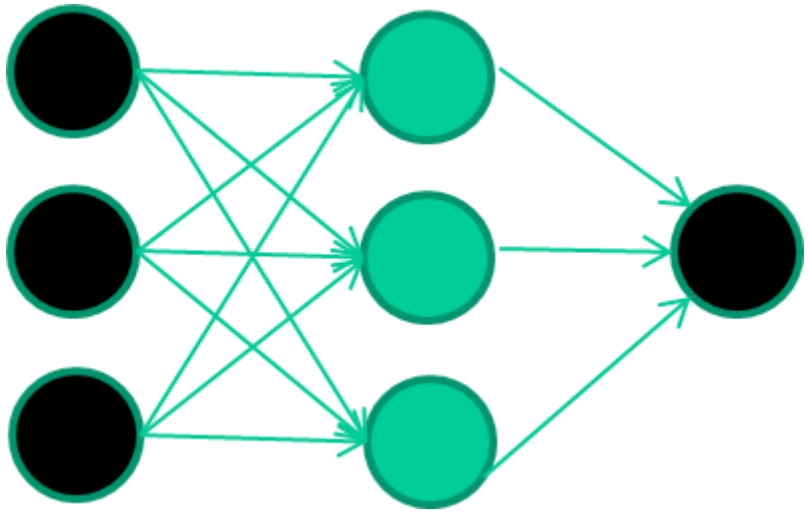


Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0

etc …

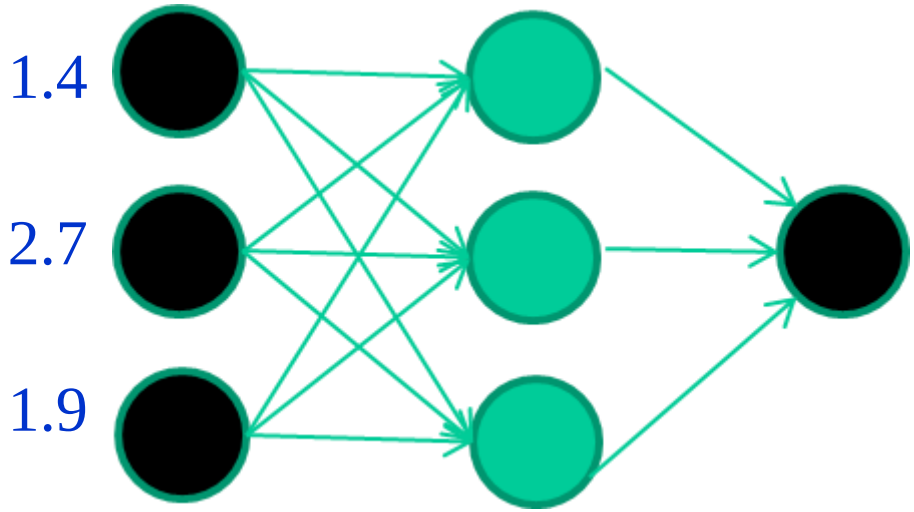
Initialise with random weights



Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

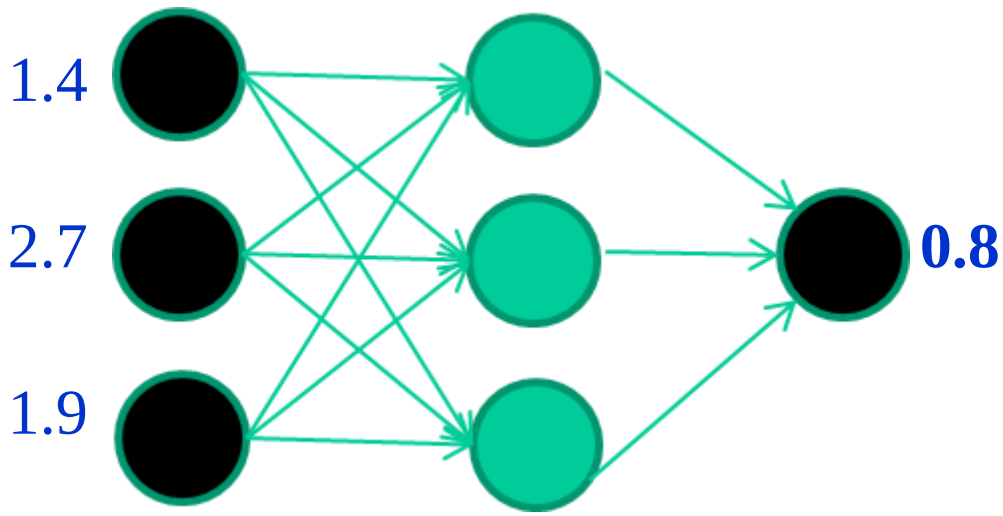
Present a training pattern



Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Feed it through to get output

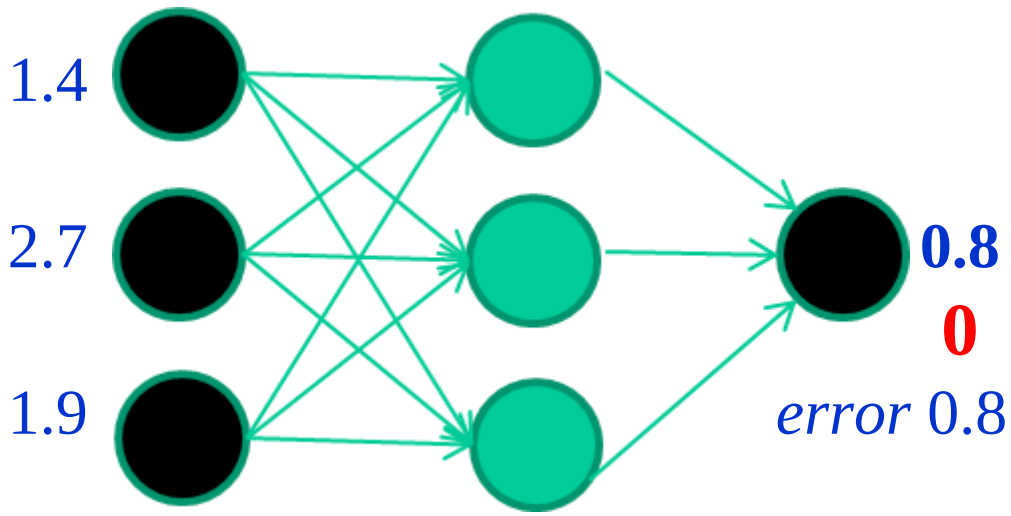


Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0

etc …

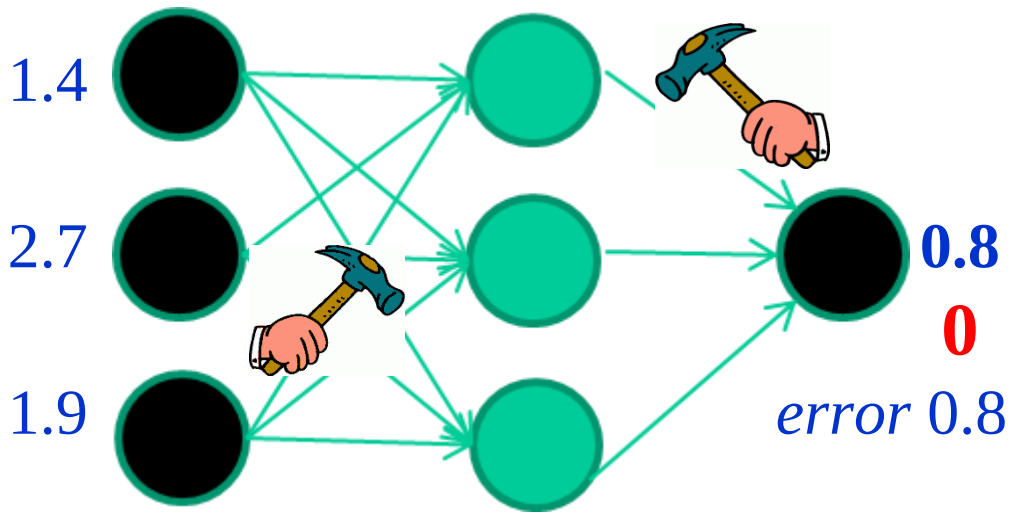
Compare with target output



Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

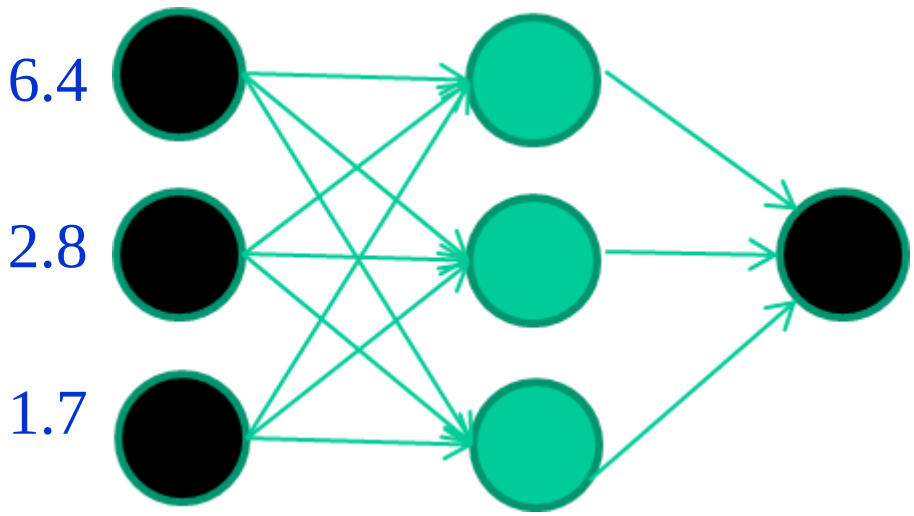
Adjust weights based on error



Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

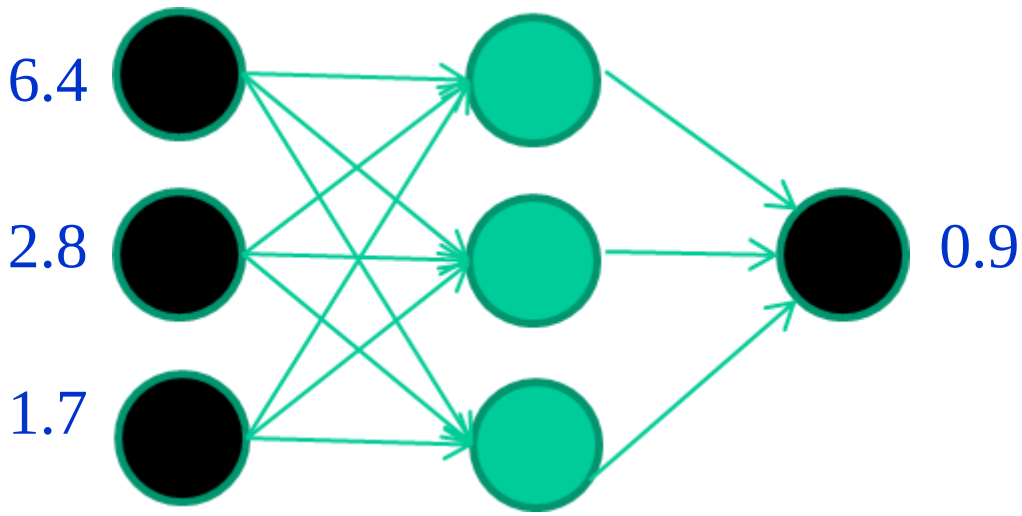
Present a training pattern



Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

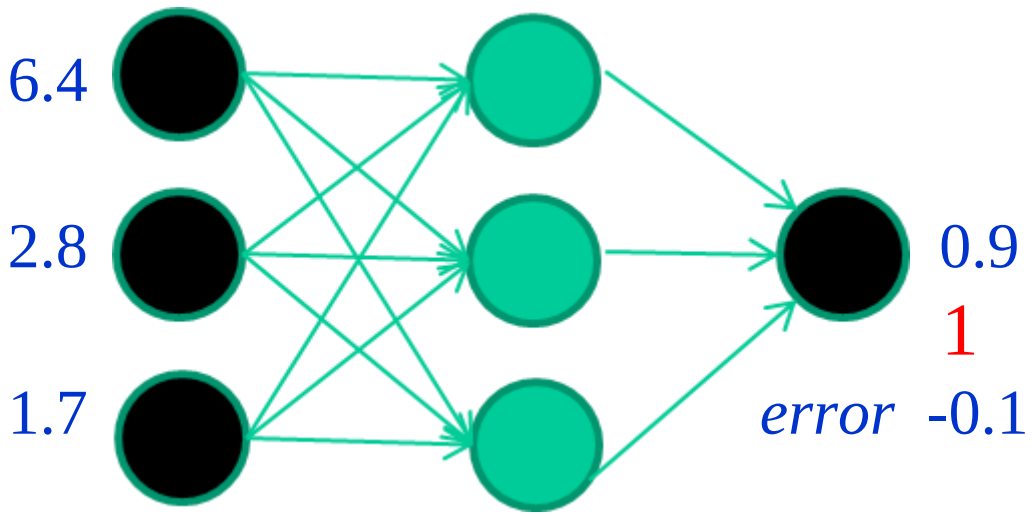
Feed it through to get output



Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

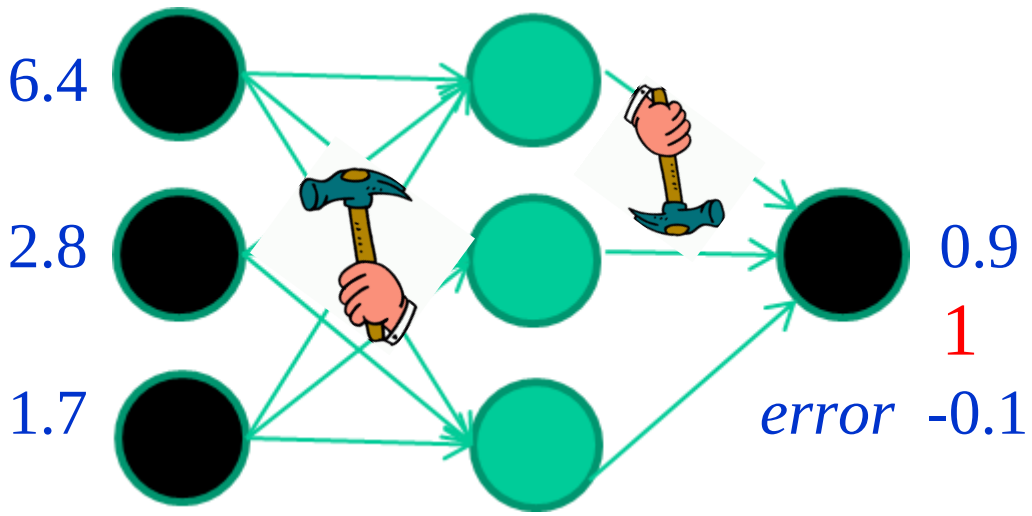
Compare with target output



Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Adjust weights based on error

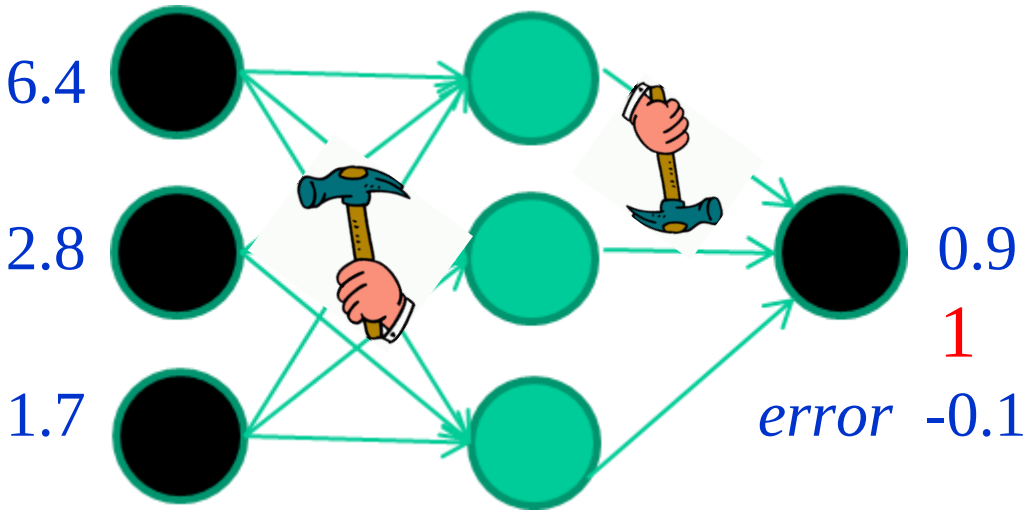


Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0

etc …

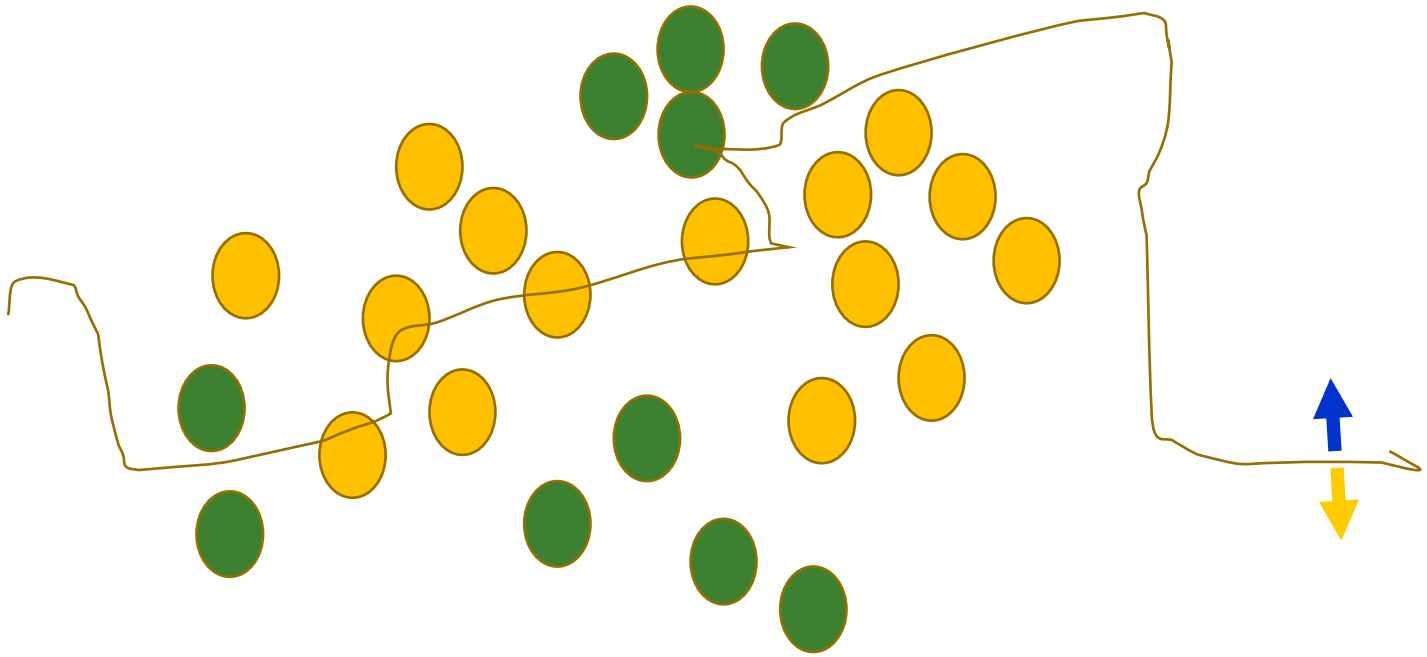
And so on ….



Repeat this thousands, maybe millions of times – each time taking a random training instance, and making slight weight adjustments.
Algorithms for weight adjustment are designed to make changes that will reduce the error.

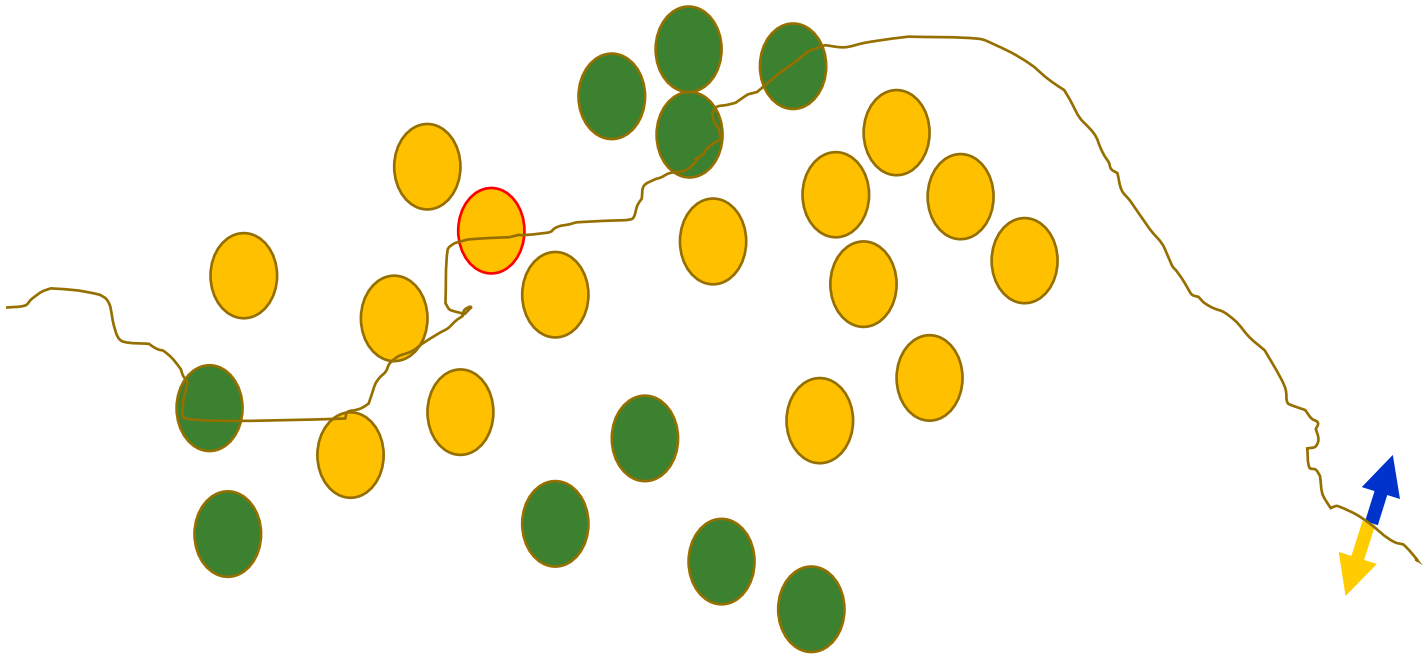
The decision boundary perspective…

Initial random weights



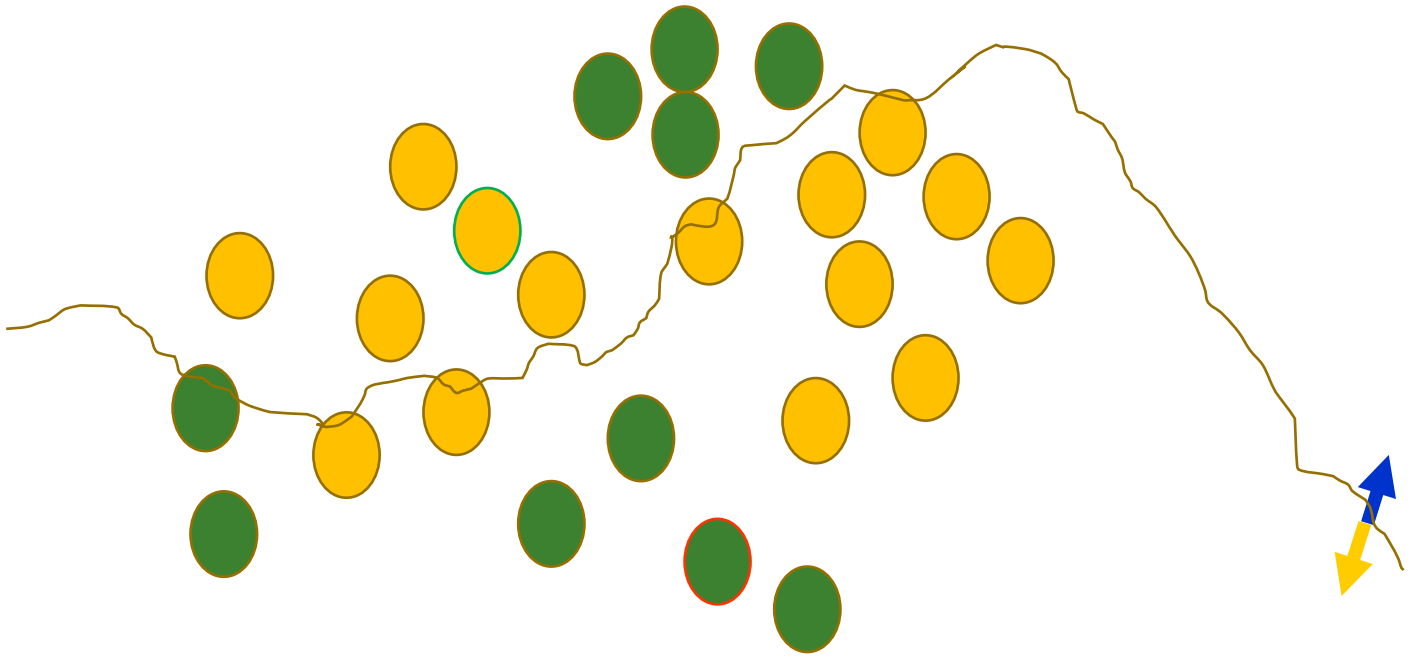
The decision boundary perspective…

Present a training instance / adjust the weights



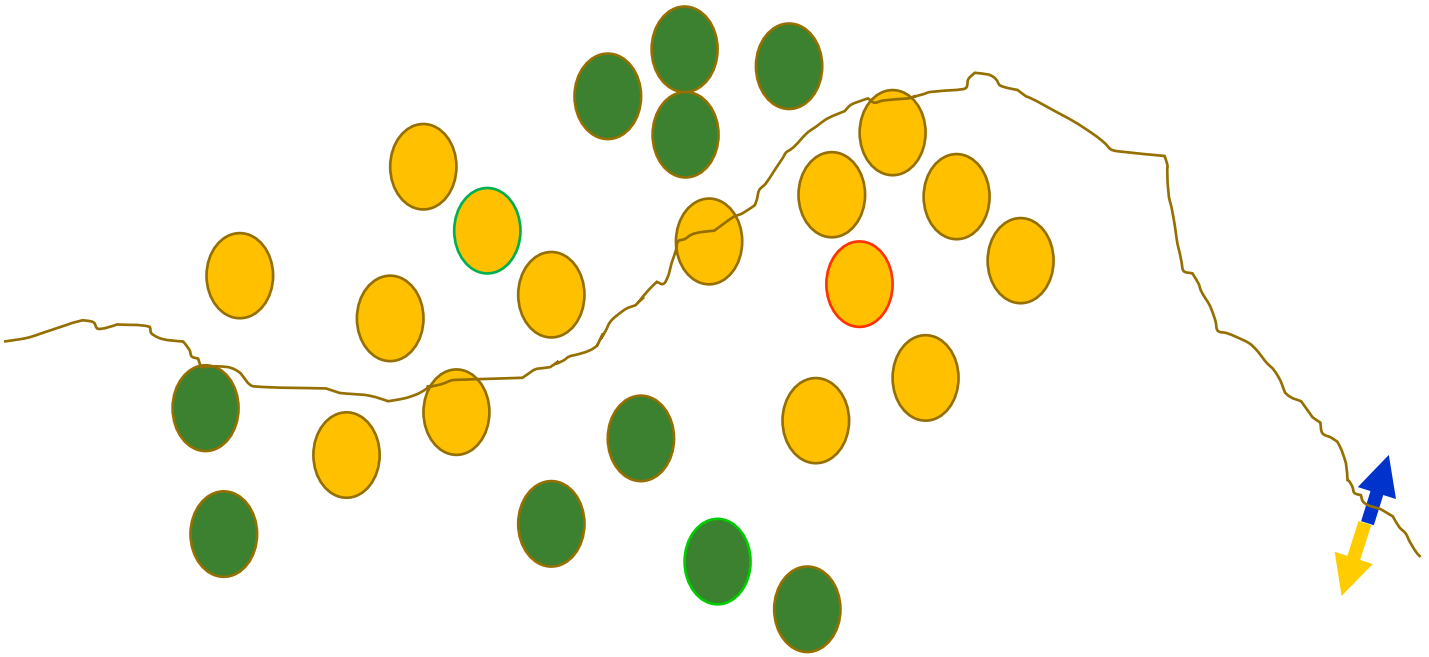
The decision boundary perspective…

Present a training instance / adjust the weights



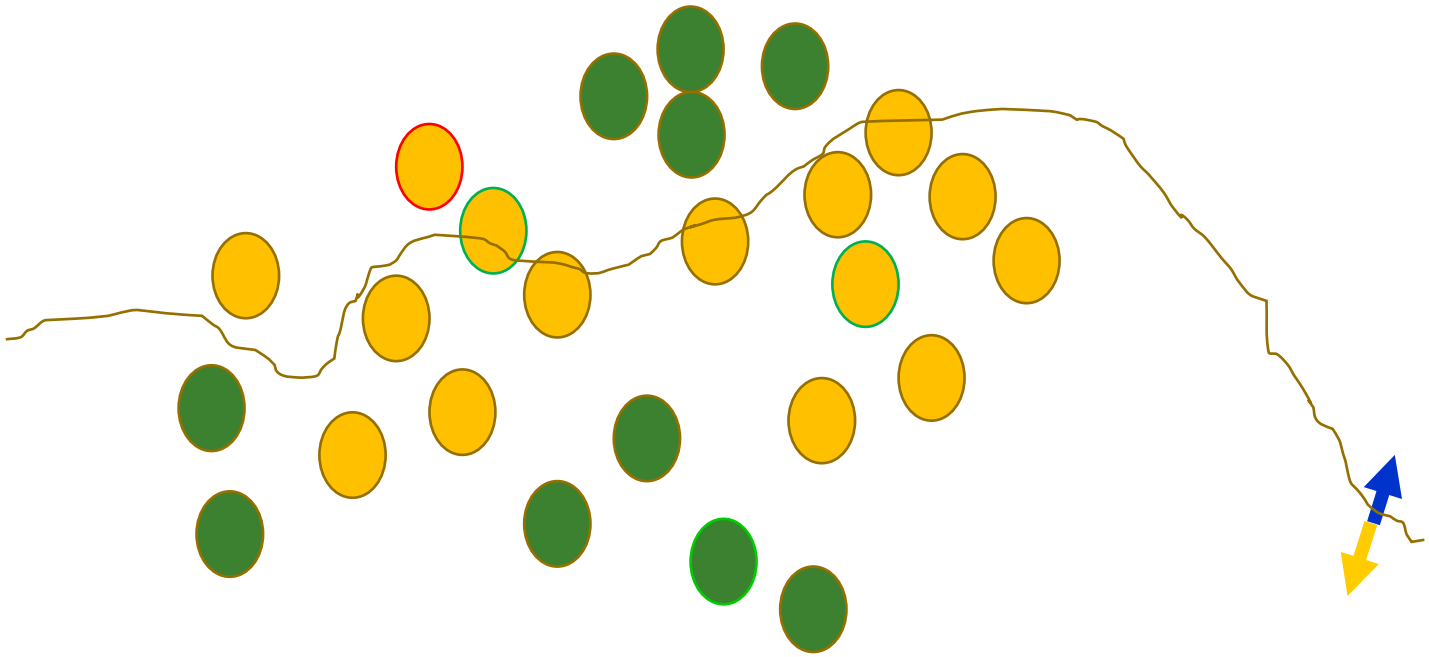
The decision boundary perspective…

Present a training instance / adjust the weights



The decision boundary perspective…

Present a training instance / adjust the weights

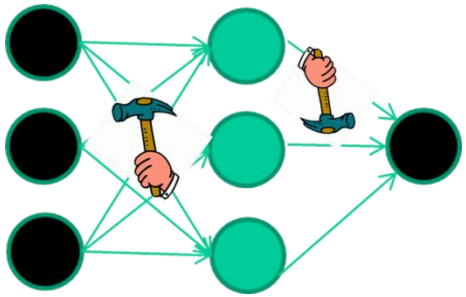


The decision boundary perspective…

Eventually ….

Point

- Weight-learning algorithms for NNs are dumb
- They work by making thousands and thousands of tiny adjustments, each making the network do better at the most recent pattern, but perhaps a little worse on many others
- But, by dumb luck, eventually this tends to be good enough to learn effective classifiers for many real applications



Some other points

If *f*(x) is non-linear, a network with 1 hidden layer can, in theory, learn perfectly any classification problem. A set of weights exists that can produce the targets from the inputs.

The problem is finding them.

Feature detectors

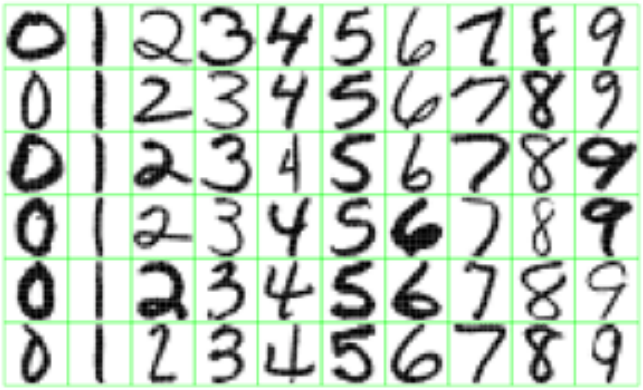
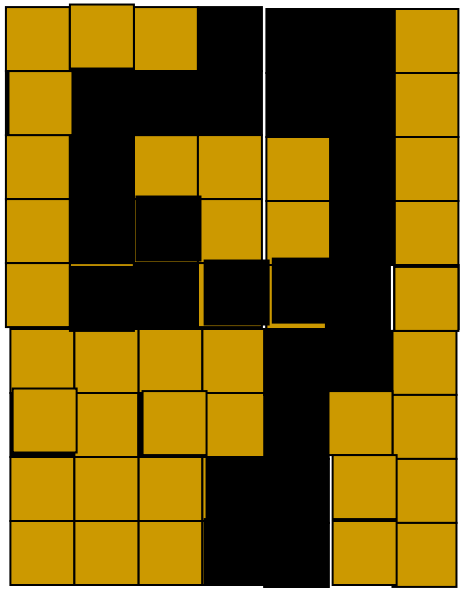
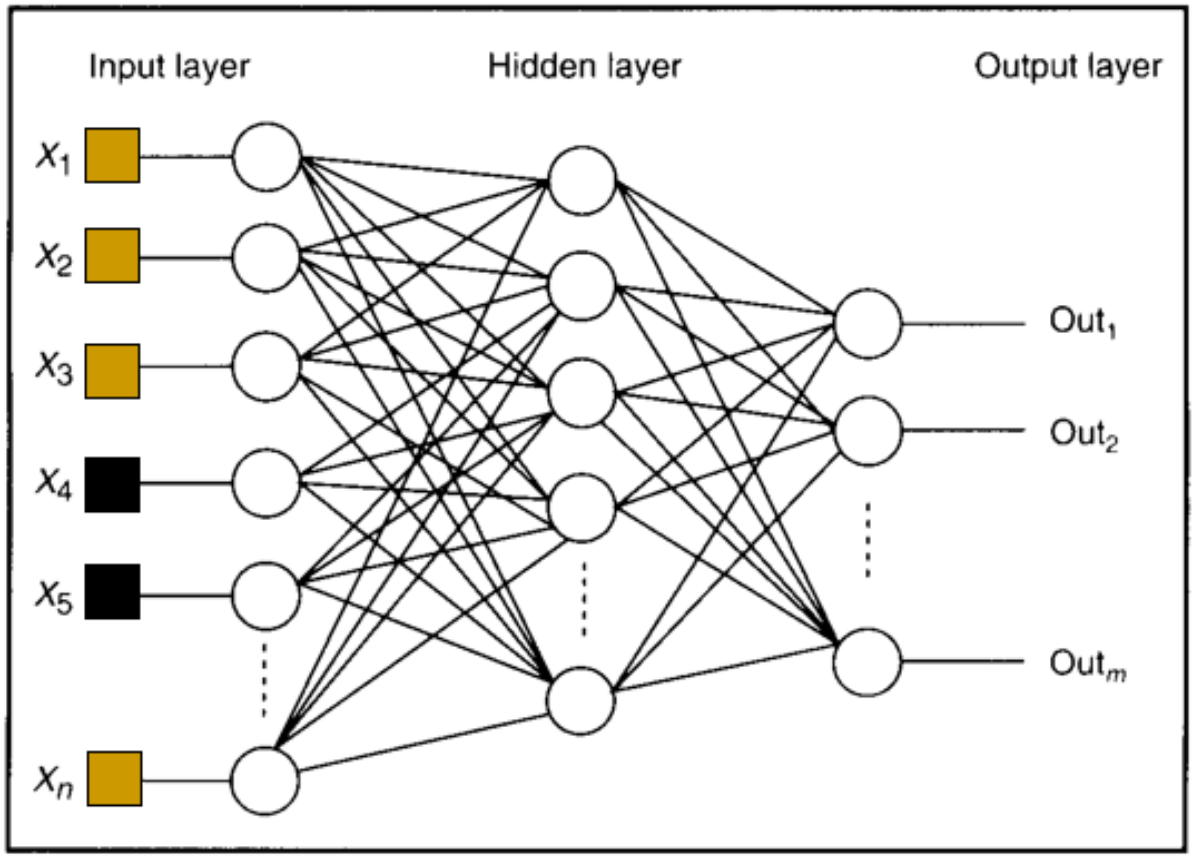


Figure 1.2: *Examples of handwritten digits from U.S. postal envelopes.*





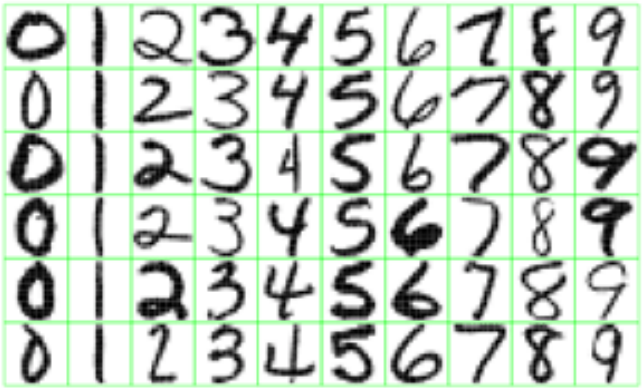
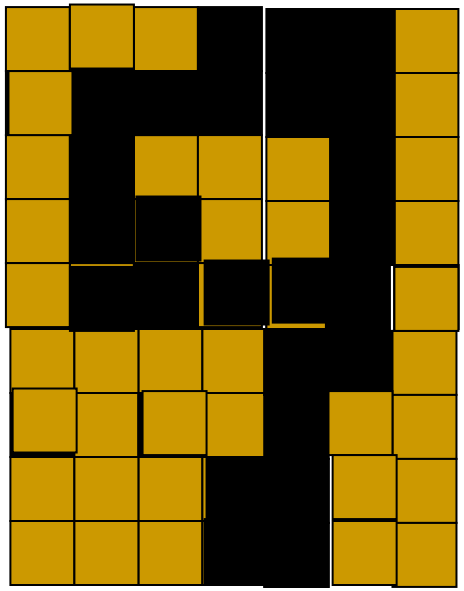
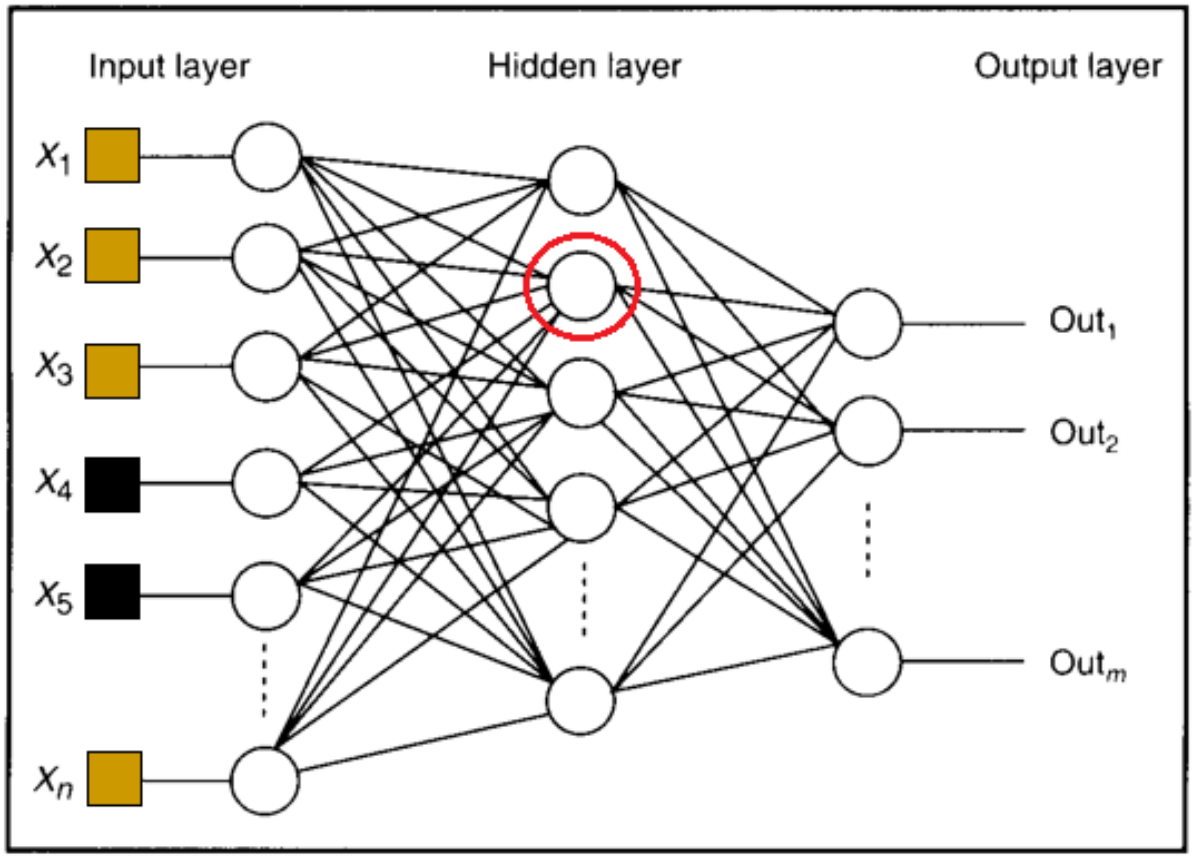


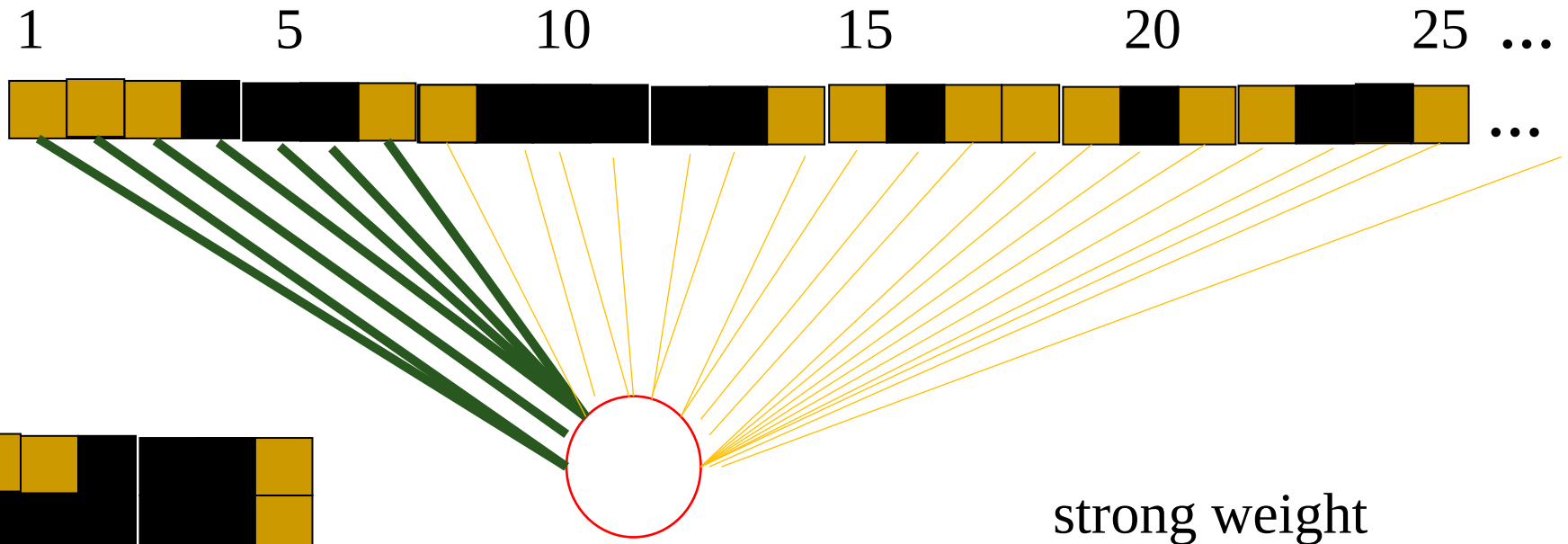
Figure 1.2: *Examples of handwritten digits from U.S. postal envelopes.*

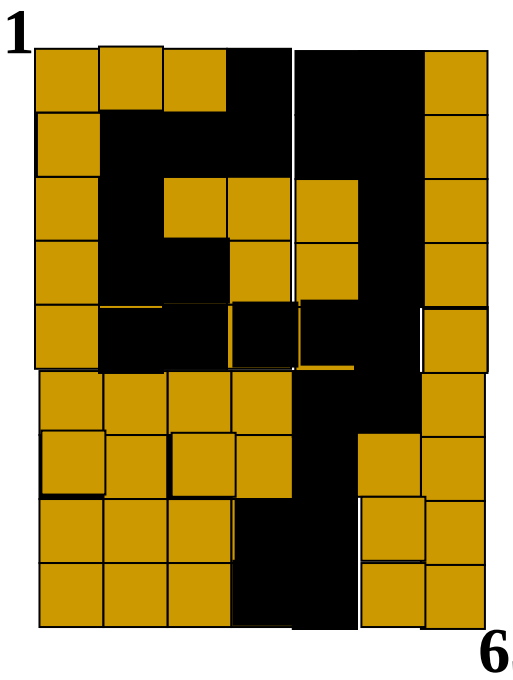


What is this unit doing?



Hidden layer units become *self-organised feature detectors*

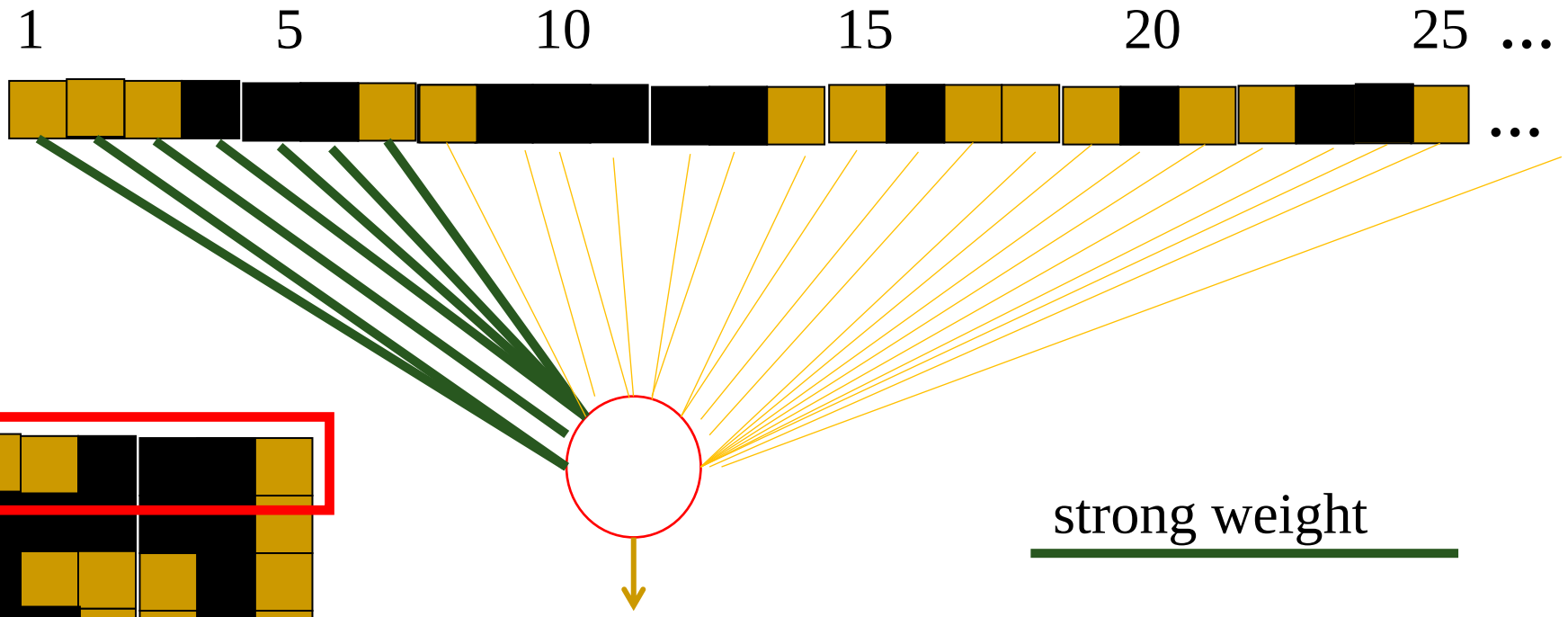


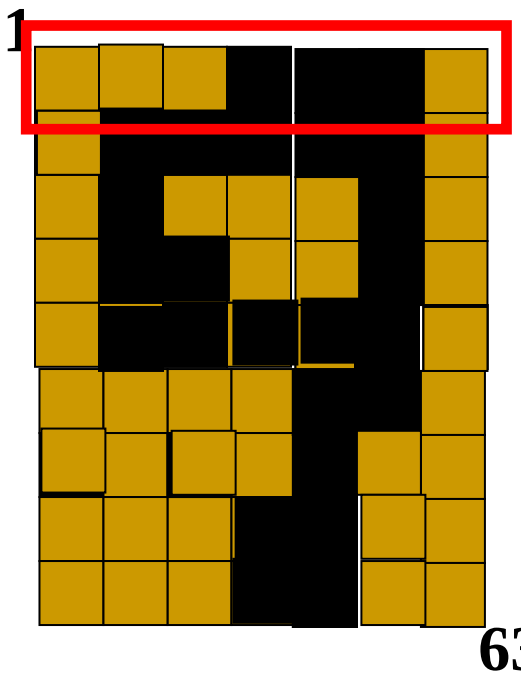


strong weight

low/zero weight

What does this unit detect?



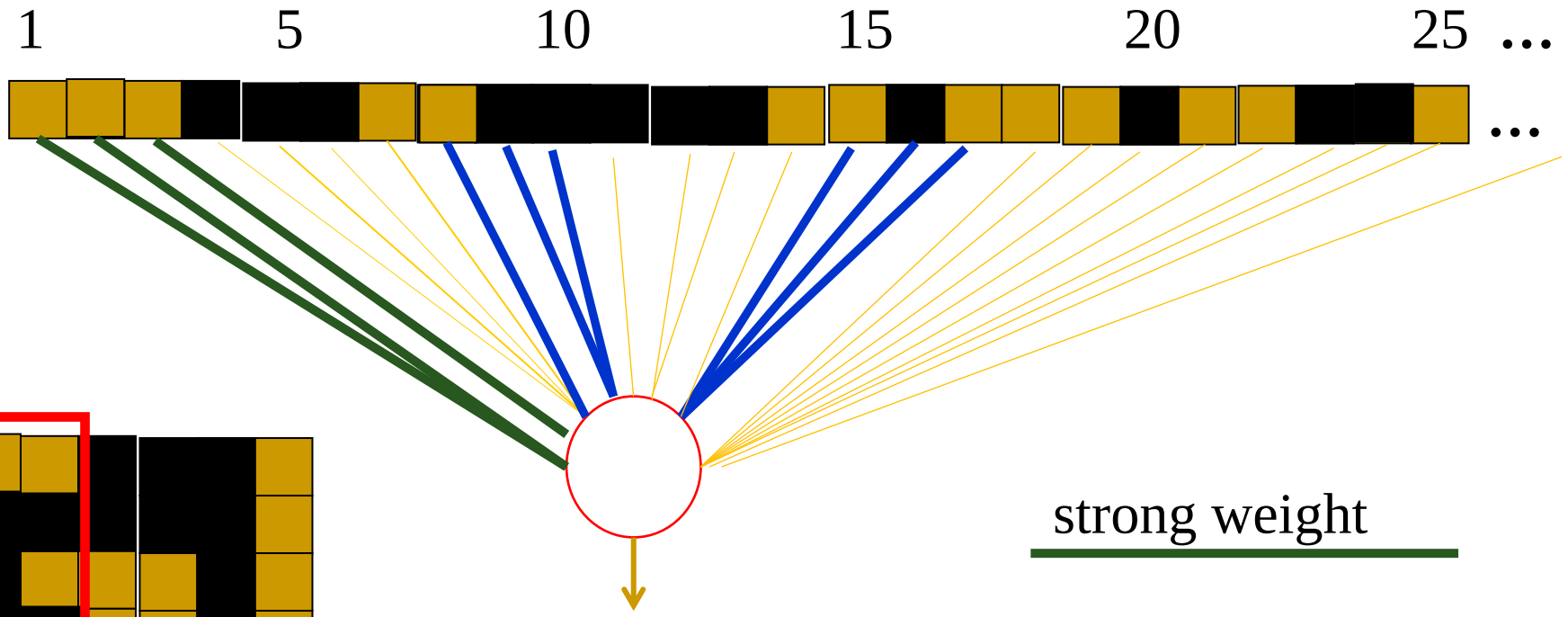


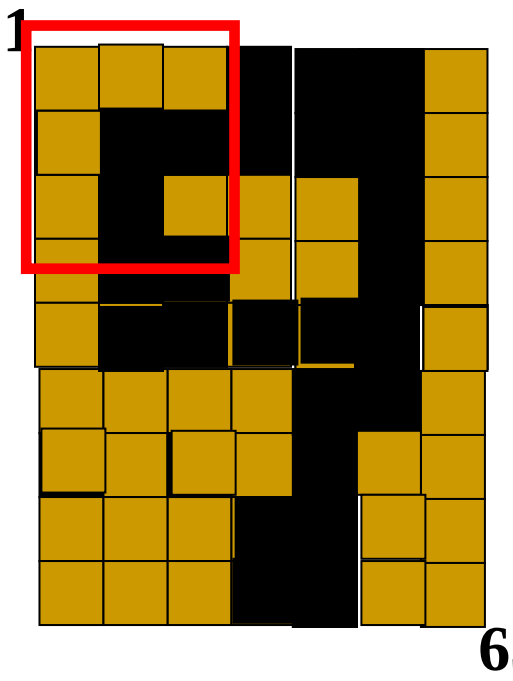
strong weight

low/zero weight

it will send strong signal for a horizontal line in the top row, ignoring everywhere else

What does this unit detect?





strong weight

low/zero weight

Strong signal for an area in the top left corner

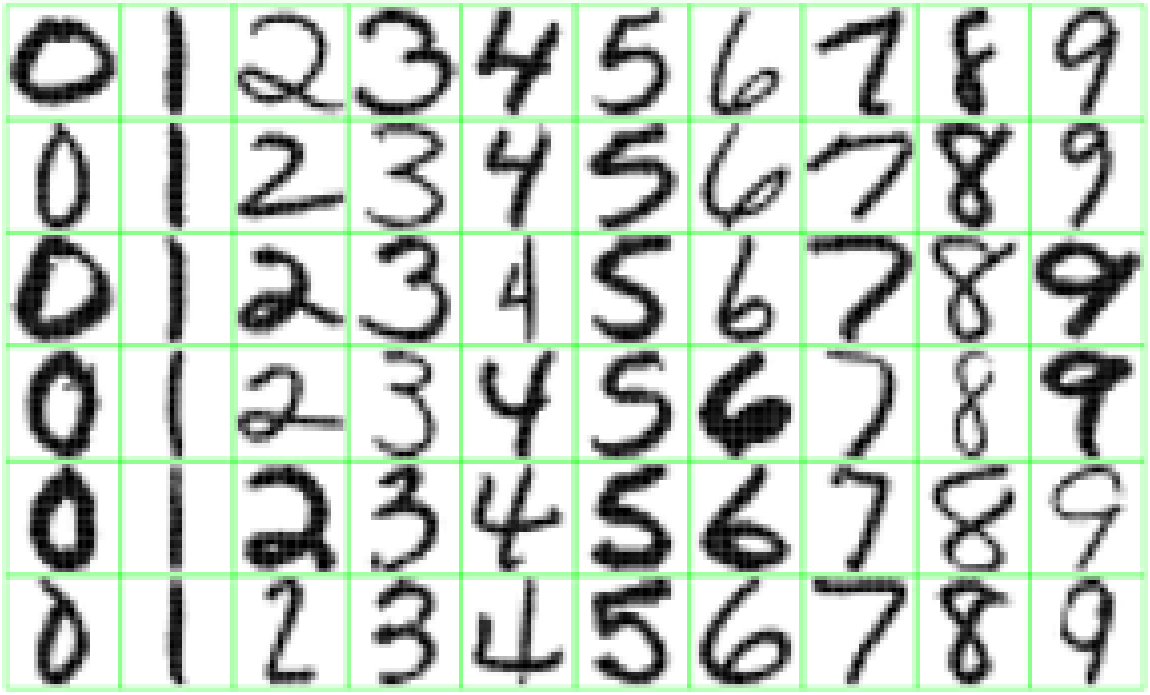
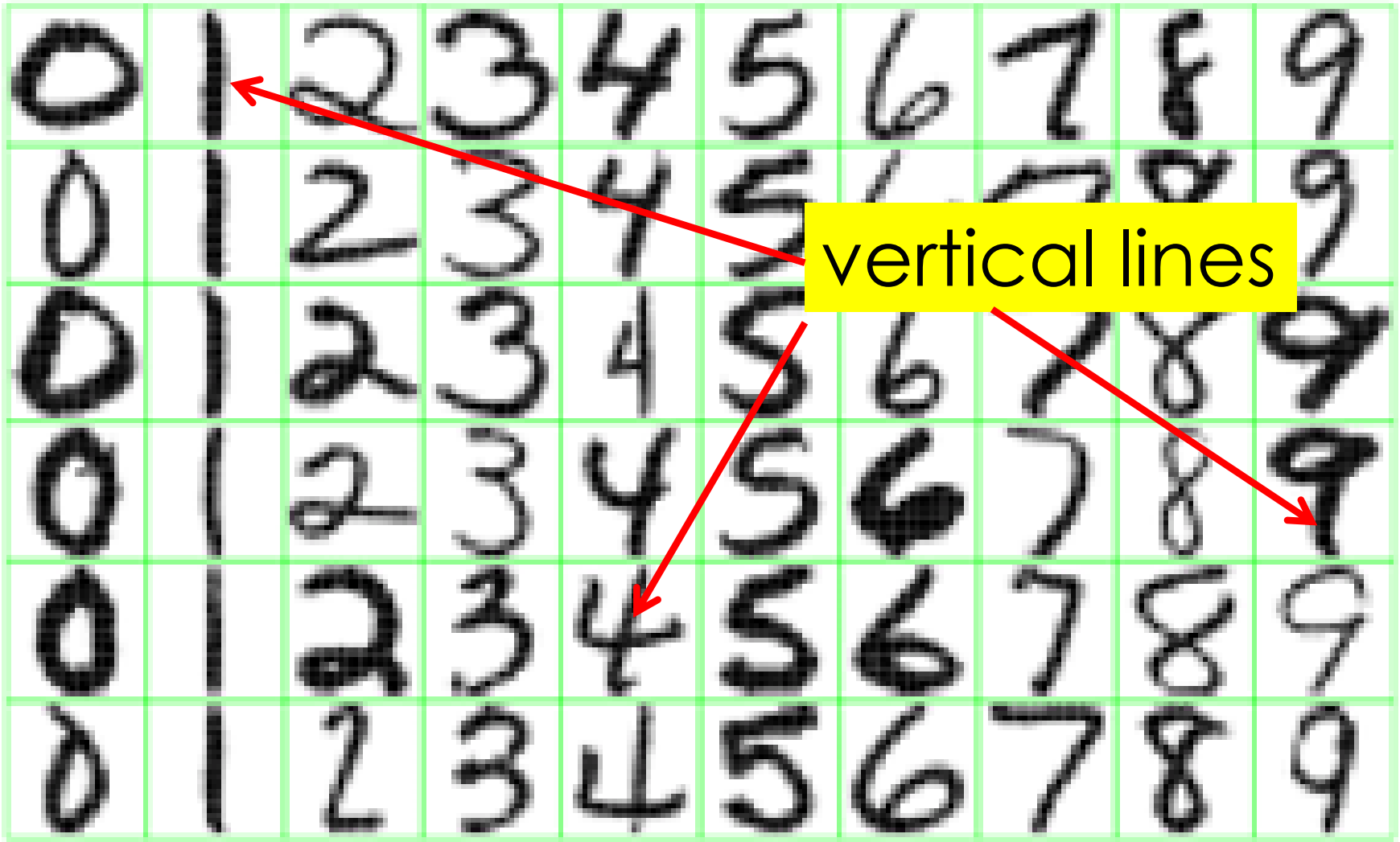


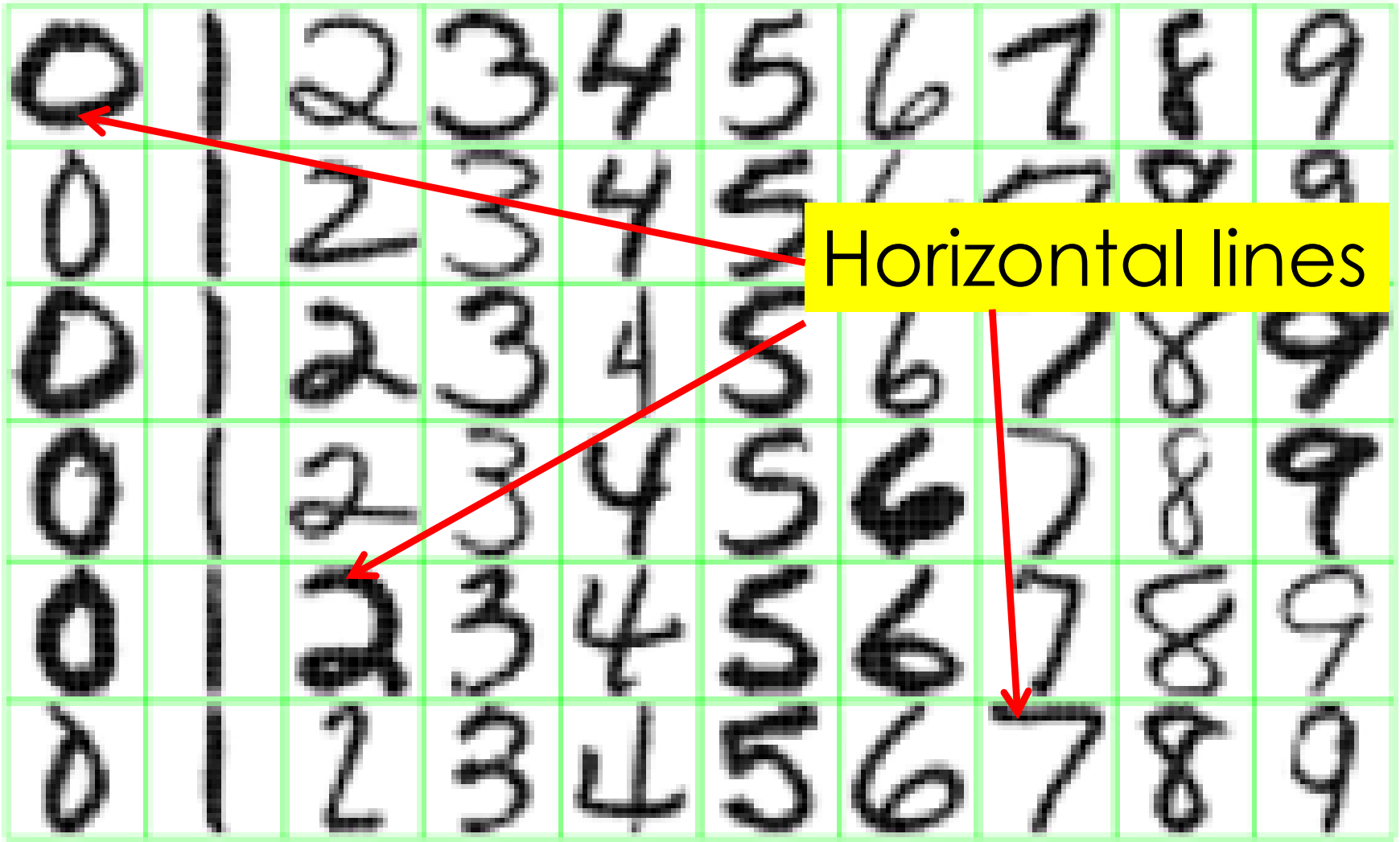
Figure 1.2: *Examples of handwritten digits from U.S. postal envelopes.*

What features might you expect a good NN to learn, when trained with data like this?



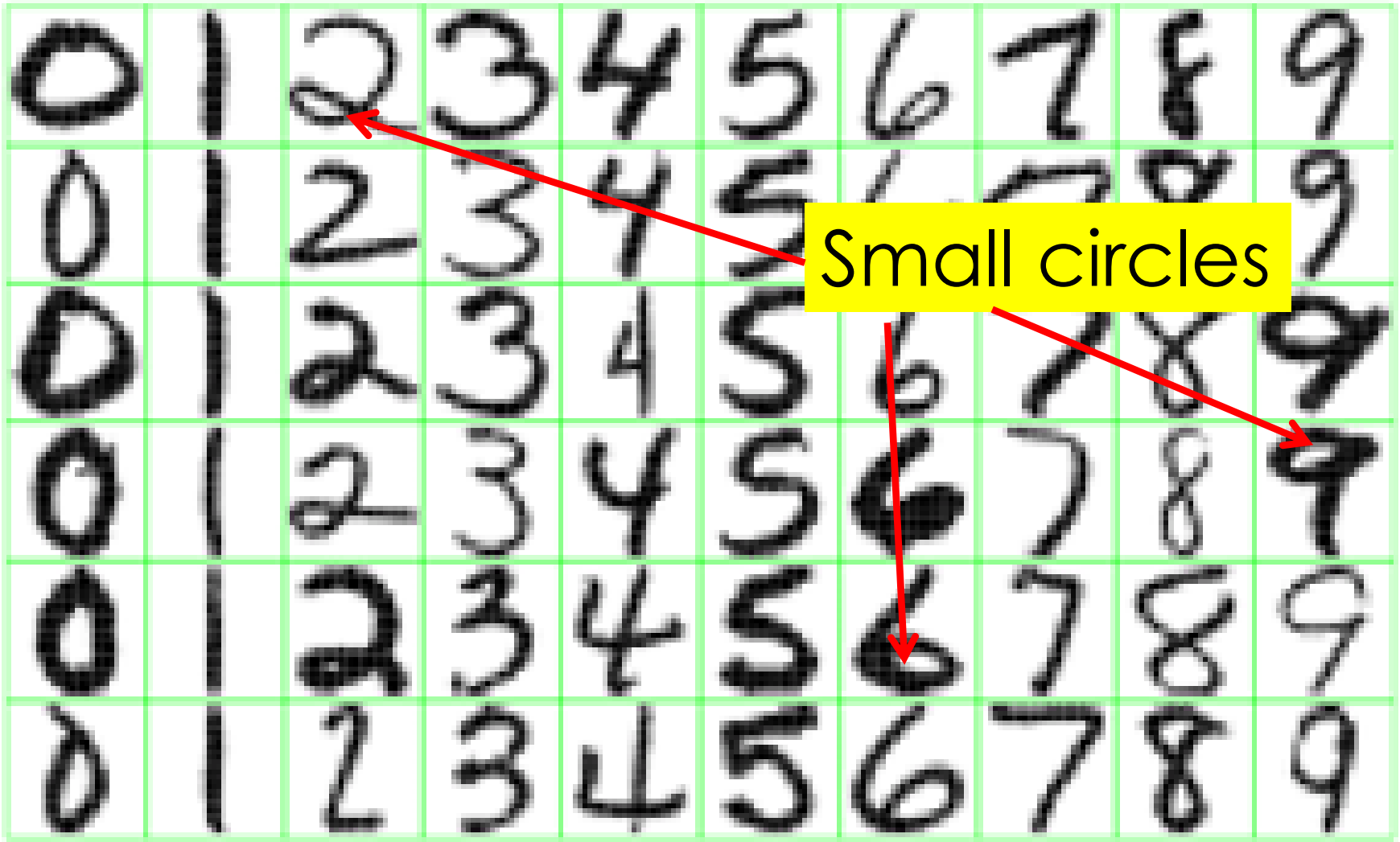
1

Figure 1.2: *Examples of handwritten digits from U.S. postal envelopes.*



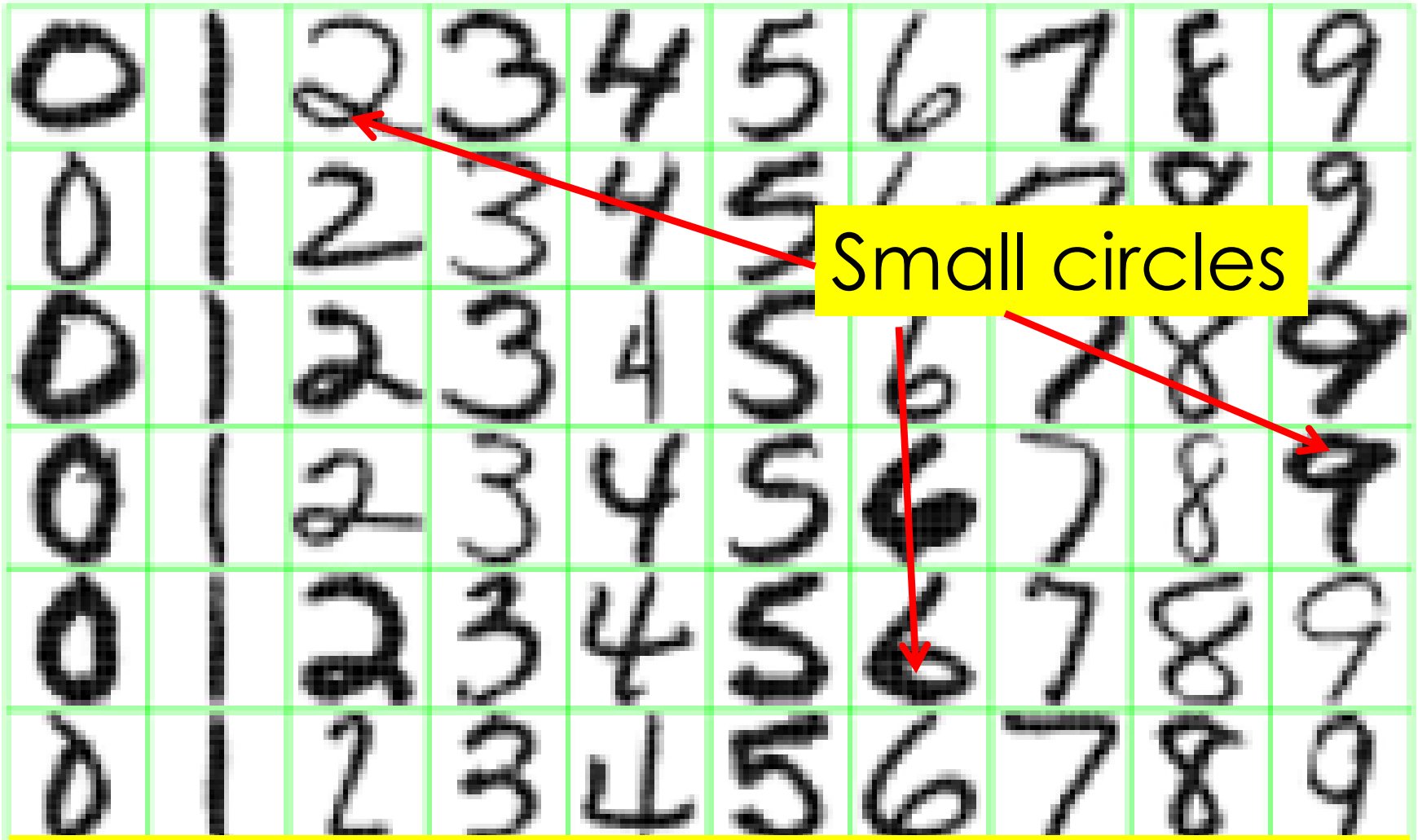
1

Figure 1.2: *Examples of handwritten digits from U.S. postal envelopes.*



1

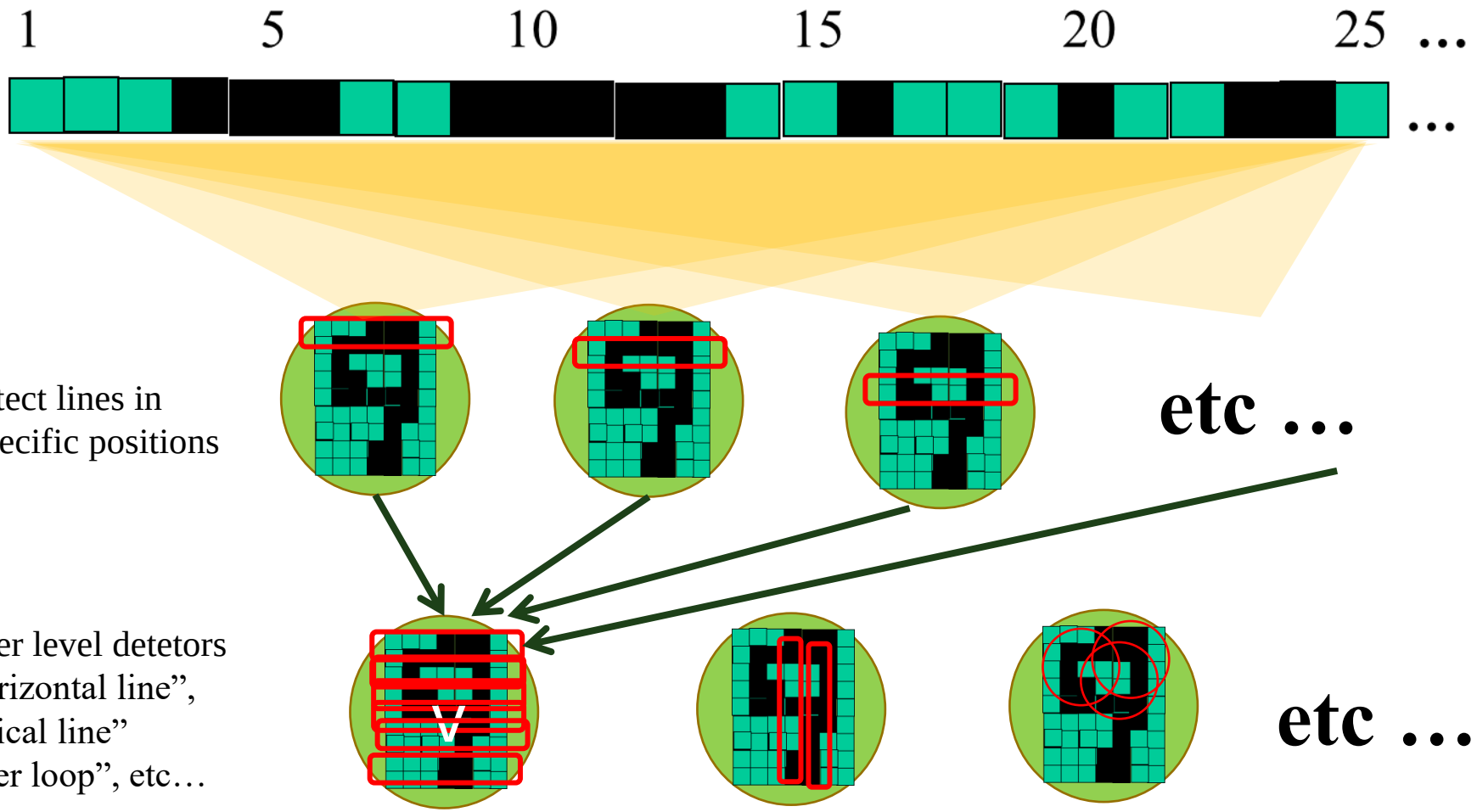
Figure 1.2: *Examples of handwritten digits from U.S. postal envelopes.*



1

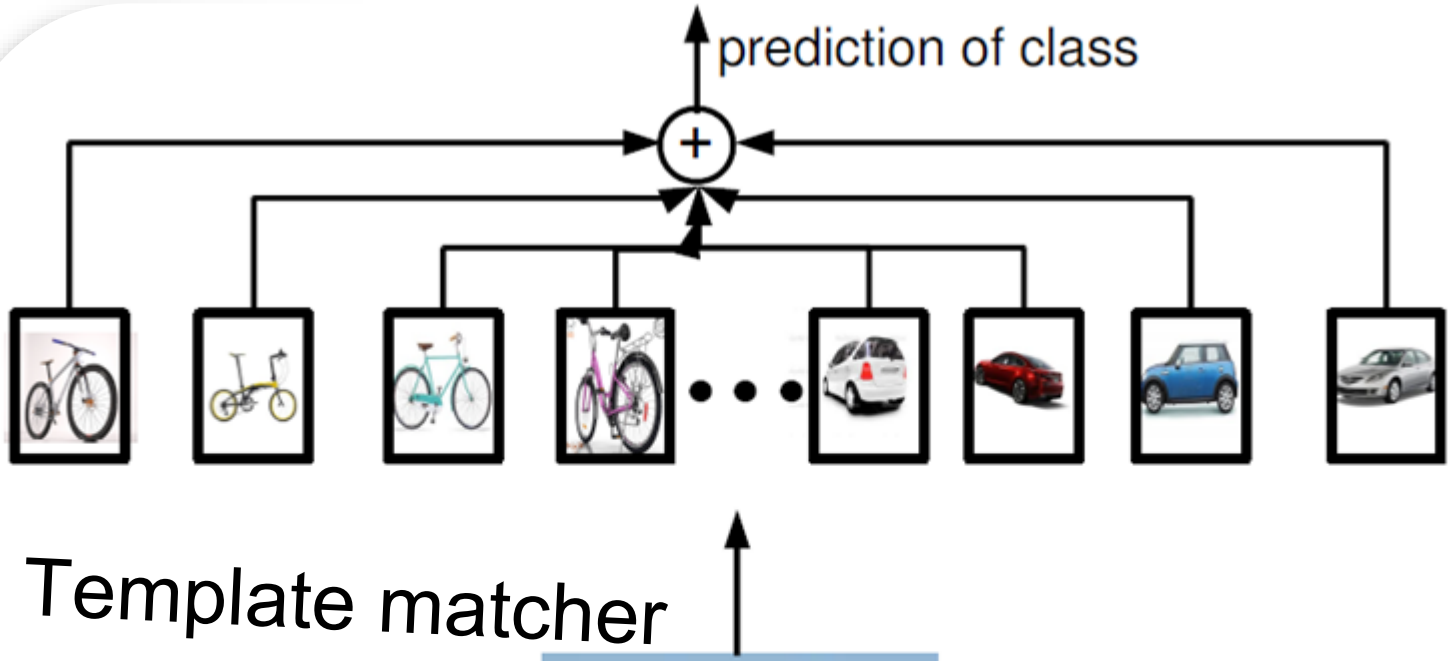
But what about position invariance ??? our example unit detectors were tied to specific parts of the image

successive layers can learn higher-level features …



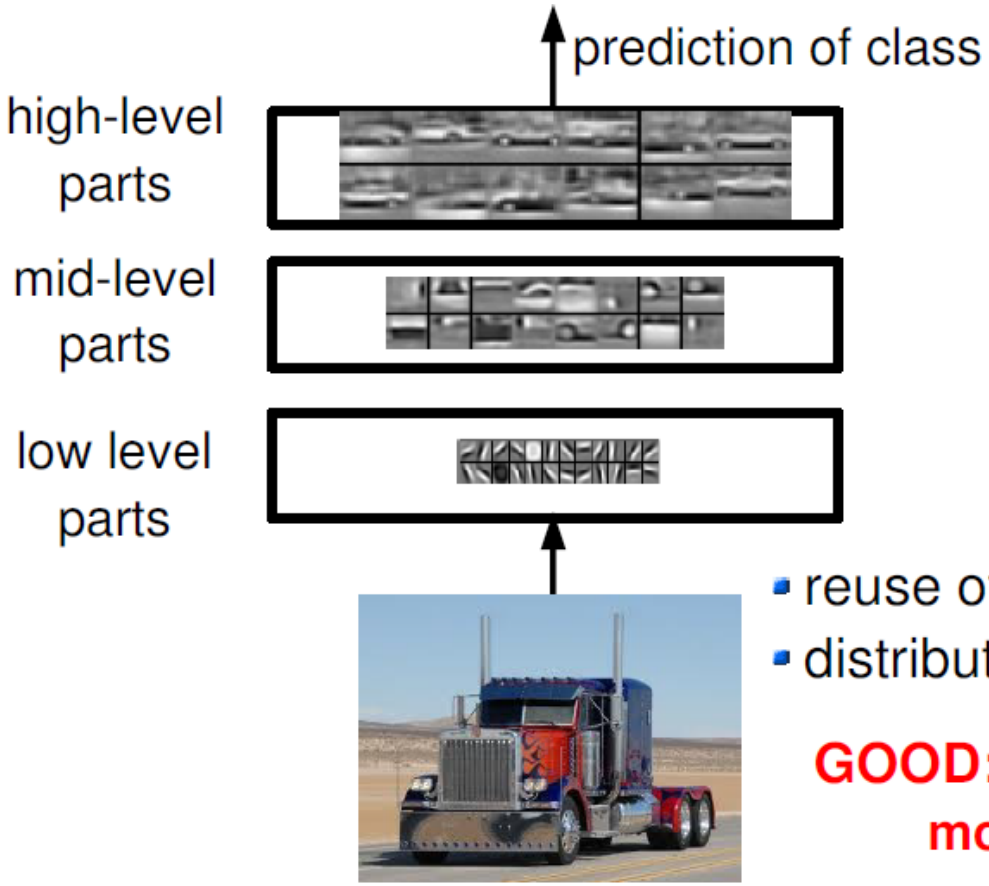
LEARNING

RECALL: the traditional approach



BAD: it may require an exponential nr. of templates!!!

Composition



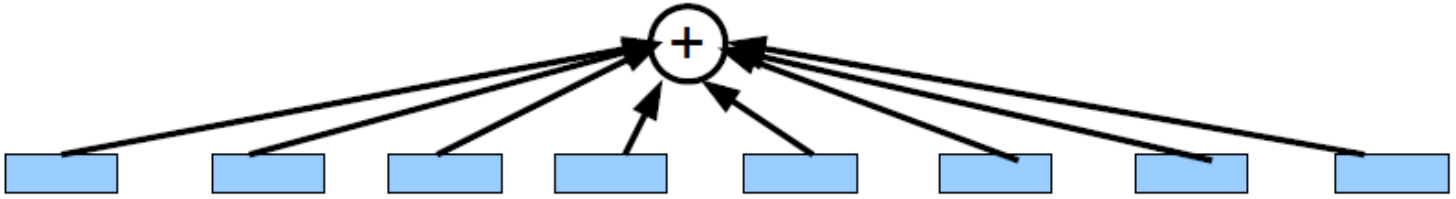
- reuse of intermediate parts
- distributed representations

GOOD: (exponentially) more efficient

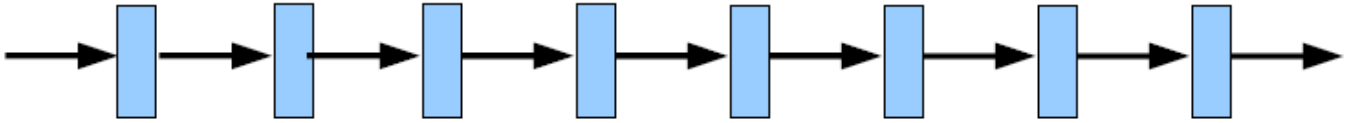
Learning

Given a dictionary of simple non-linear functions: $g_1, \ldots, g_n$

Proposal #1: linear combination $f(x) \approx \sum_j g_j$



Proposal #2: composition $f(x) \approx g_1(g_2(\ldots g_n(x)\ldots))$



Learning

Given a dictionary of simple non-linear functions: $g_1, \ldots, g_n$

Proposal #1: linear combination $f(x) \approx \sum_j g_j$

- Kernel learning
- Boosting
- ...

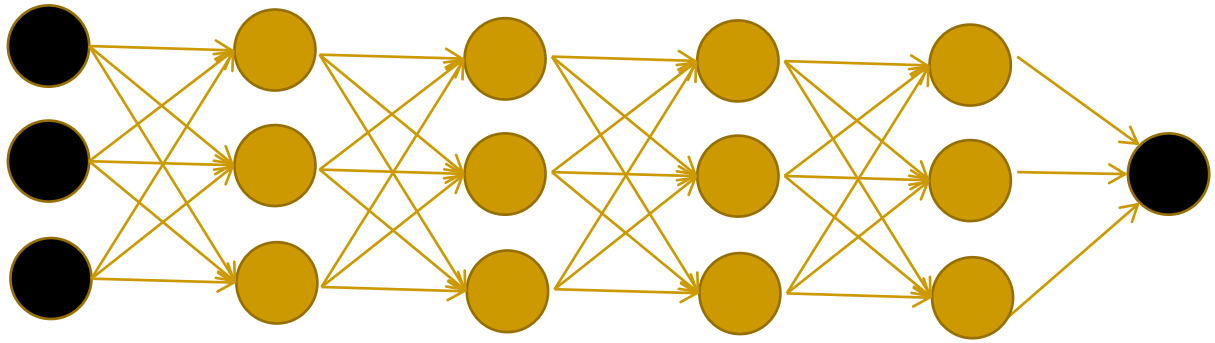
Shallow

Proposal #2: composition $f(x) \approx g_1(g_2(\ldots g_n(x)\ldots))$

- Deep learning
- Scattering networks (wavelet cascade)
- S.C. Zhou & D. Mumford "grammar"

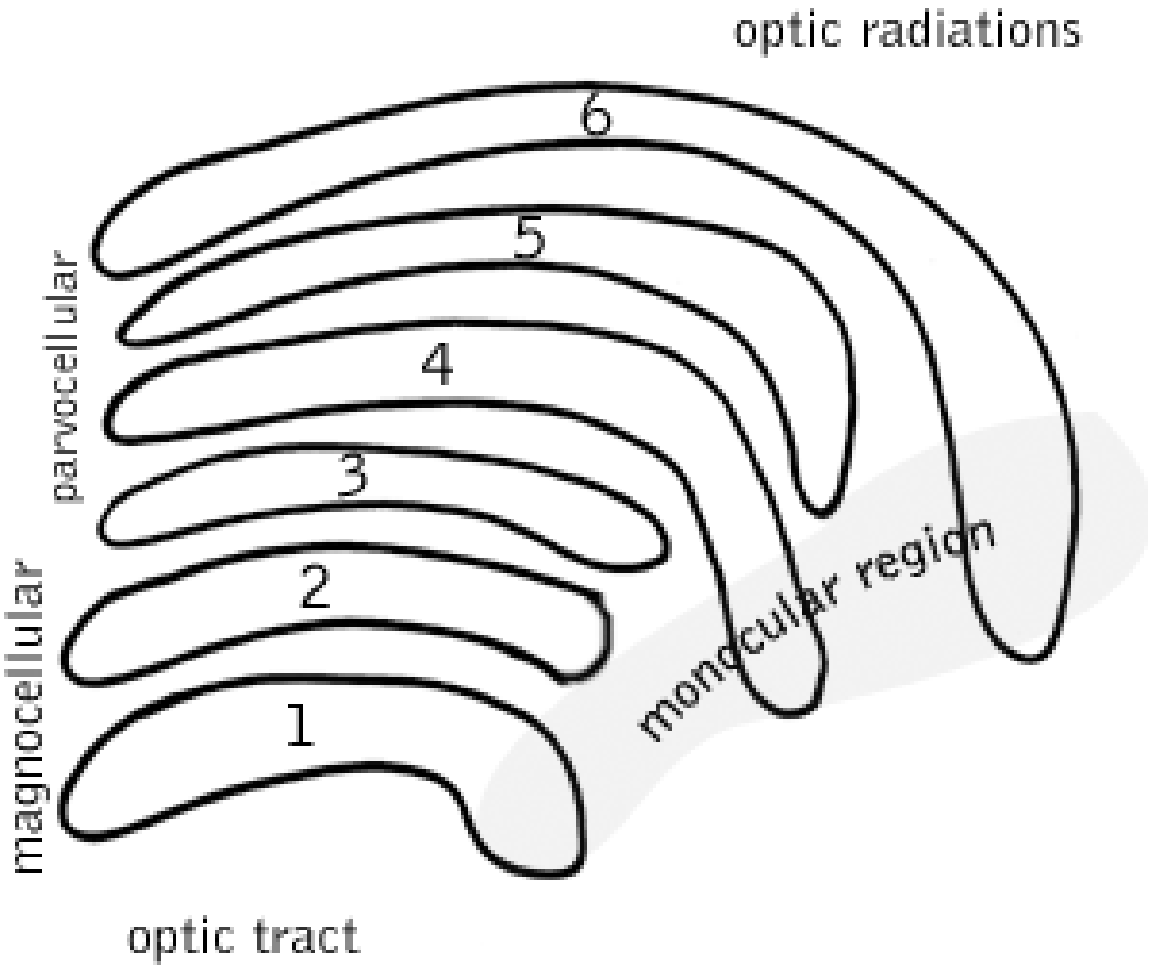
Deep

-> multiple layers



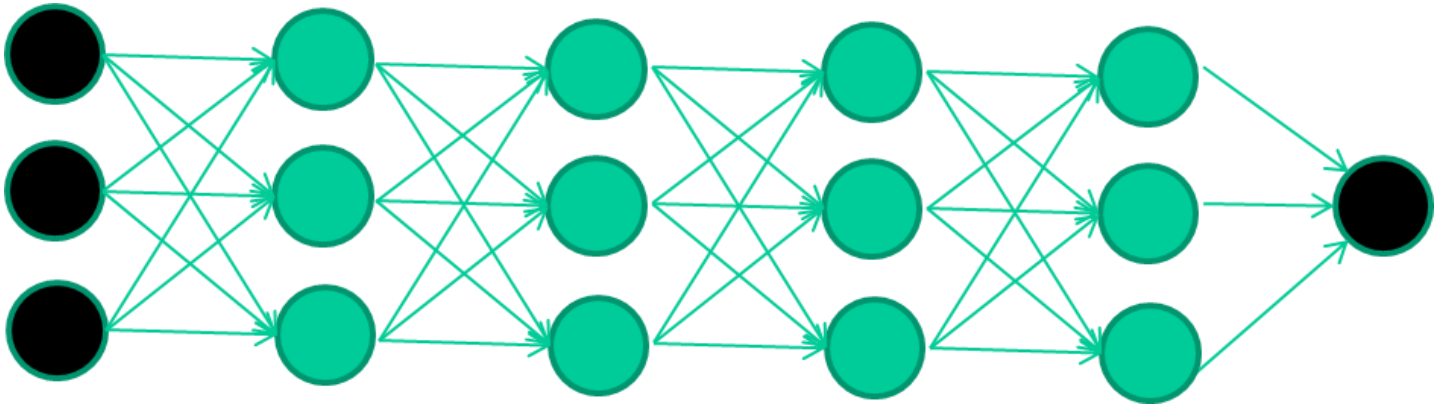
-> *multiple layers*

Your brain works that way

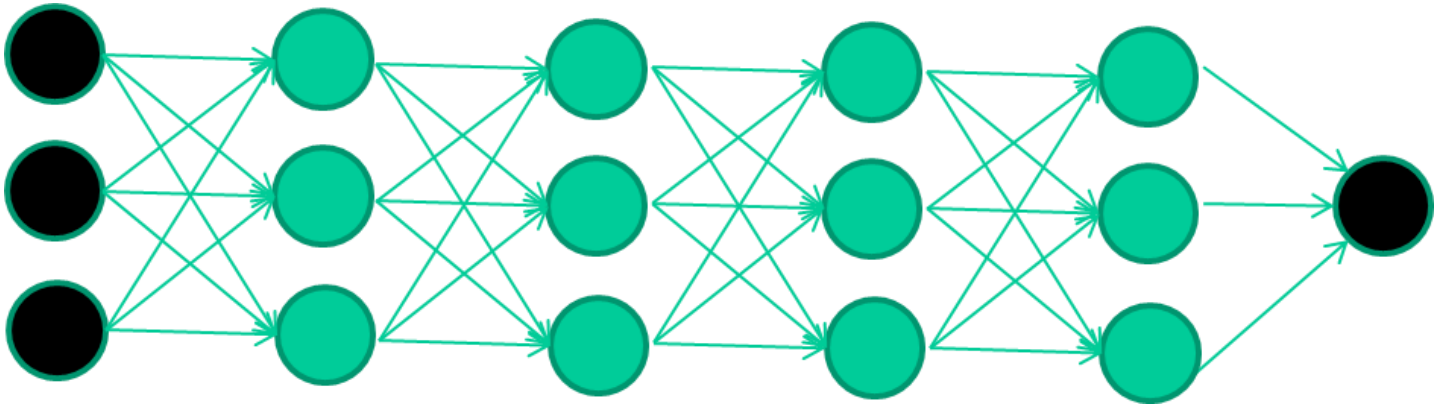


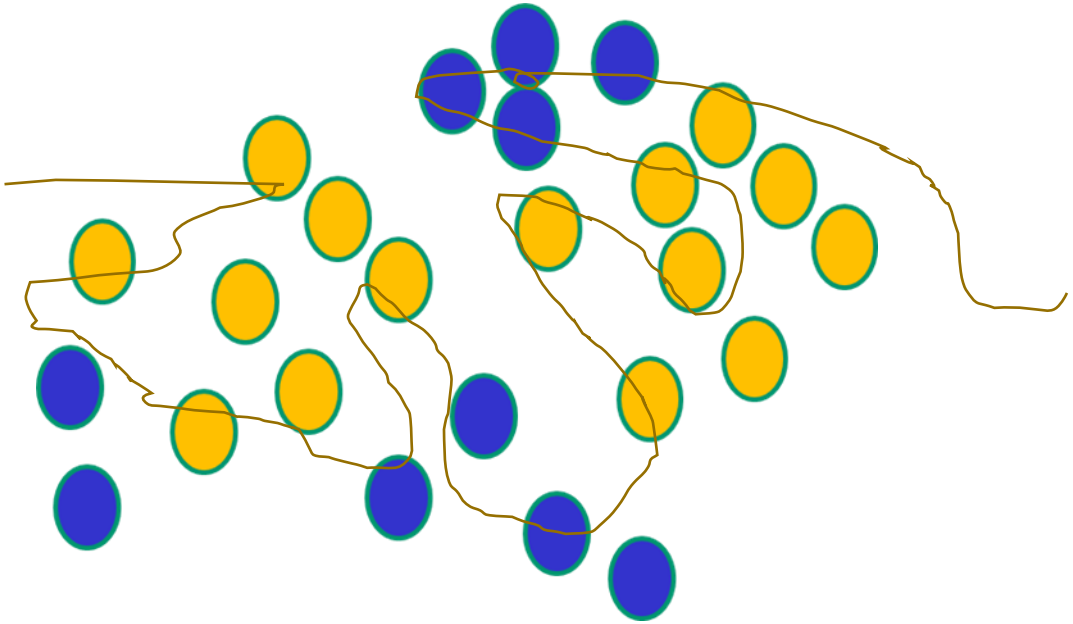
So: *multiple layers make sense*

Many-layer neural network architectures should be capable of learning the true underlying features and 'feature logic', and therefore generalise very well …



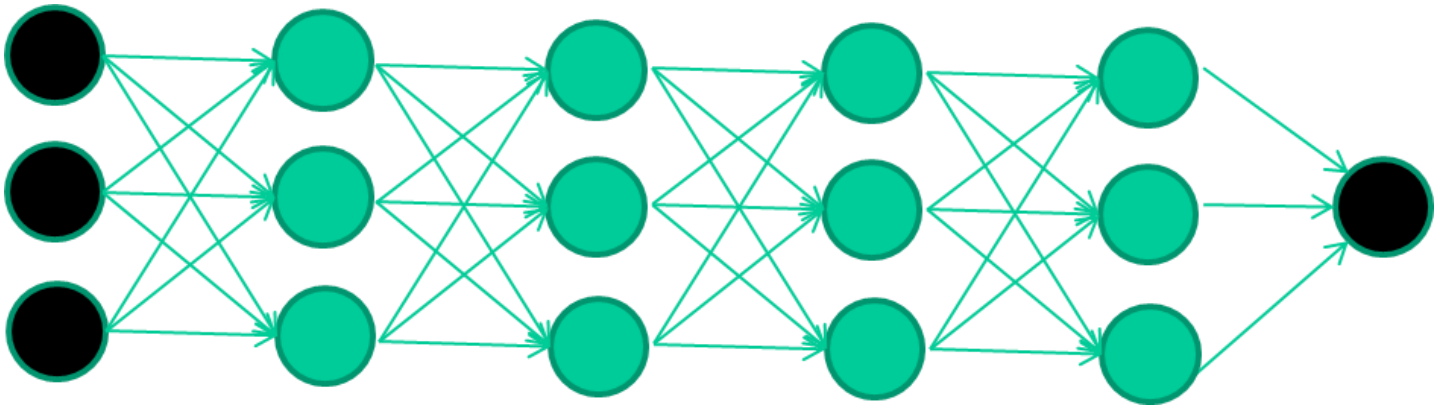
But, until very recently, our weight-learning algorithms simply did not work on multi-layer architectures



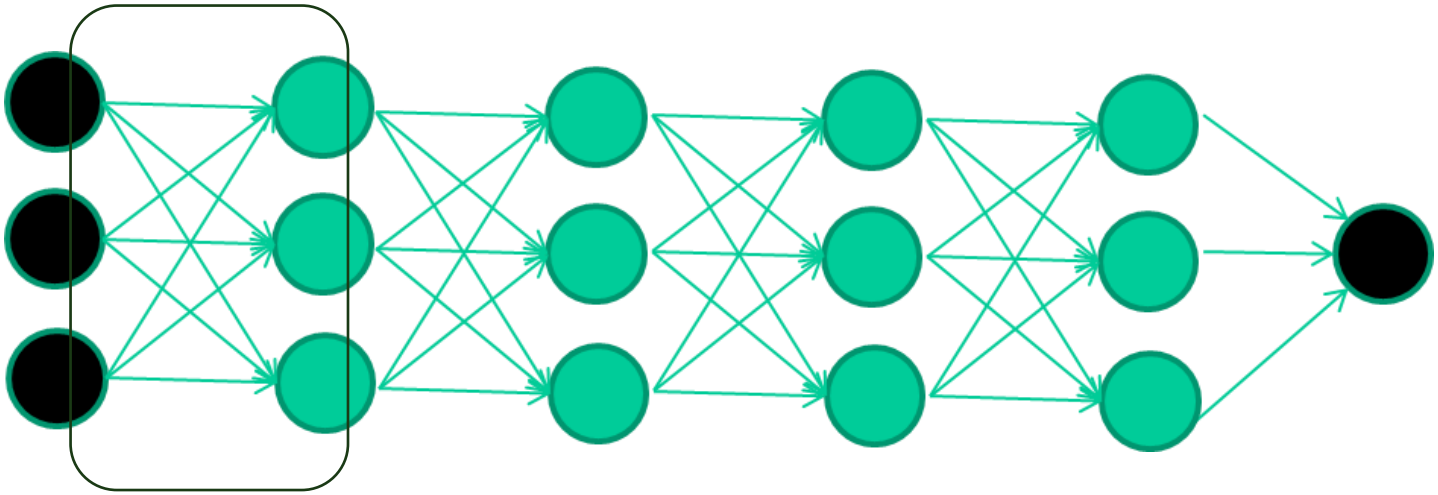



GREAT NEW
IDEA

A new way to train multi-layer NNs…

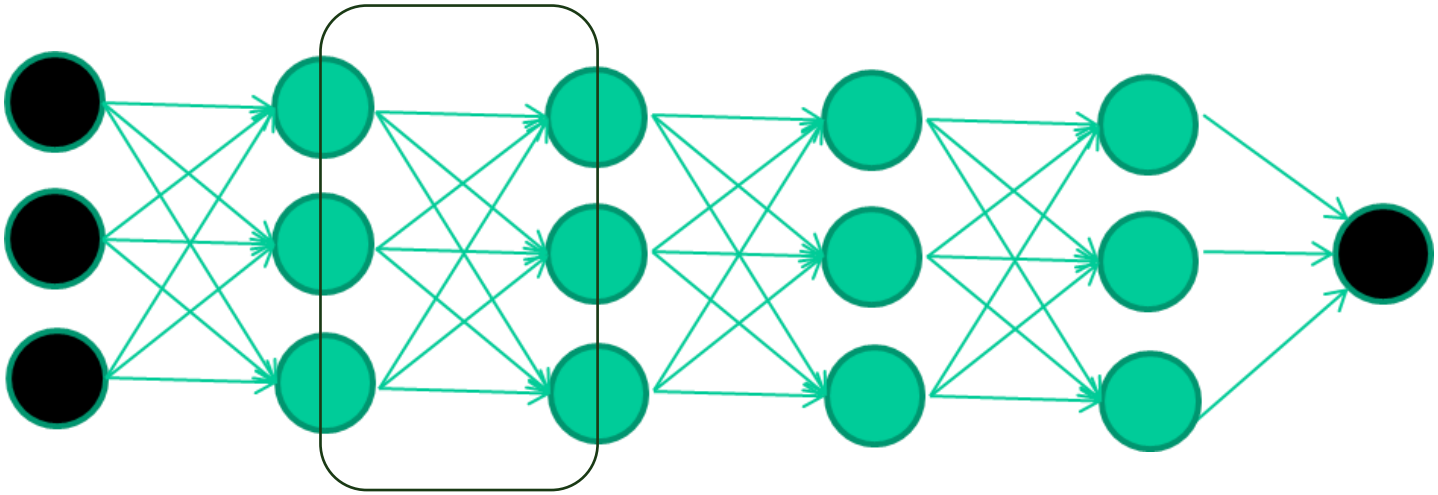


The new way to train multi-layer NNs…



Train **this** layer first

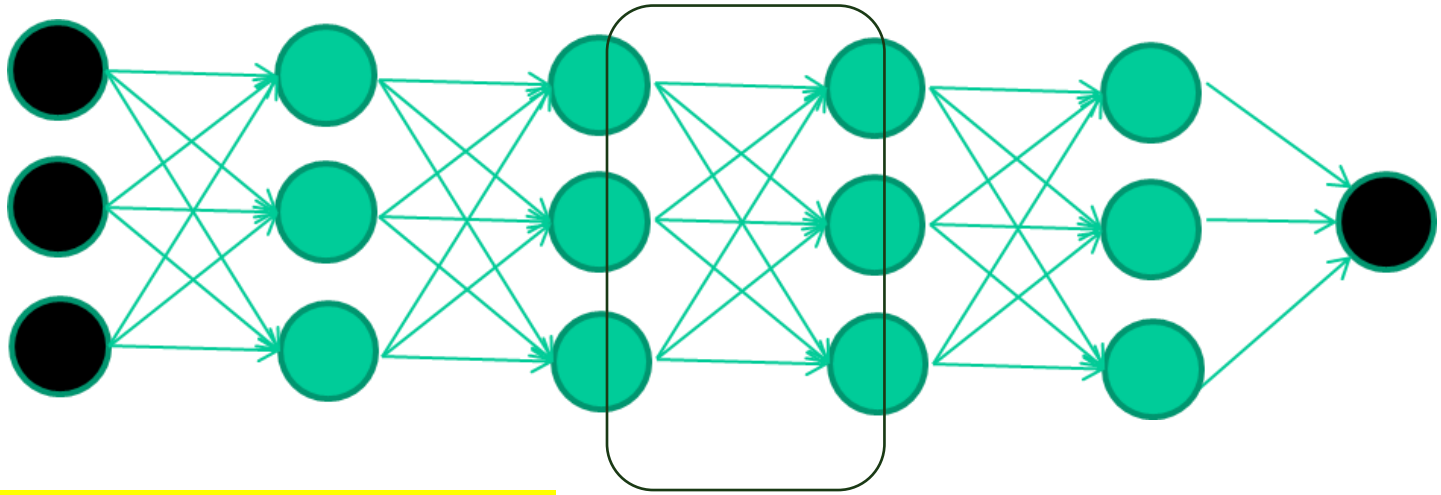
The new way to train multi-layer NNs…



Train **this** layer first

then **this** layer

The new way to train multi-layer NNs…

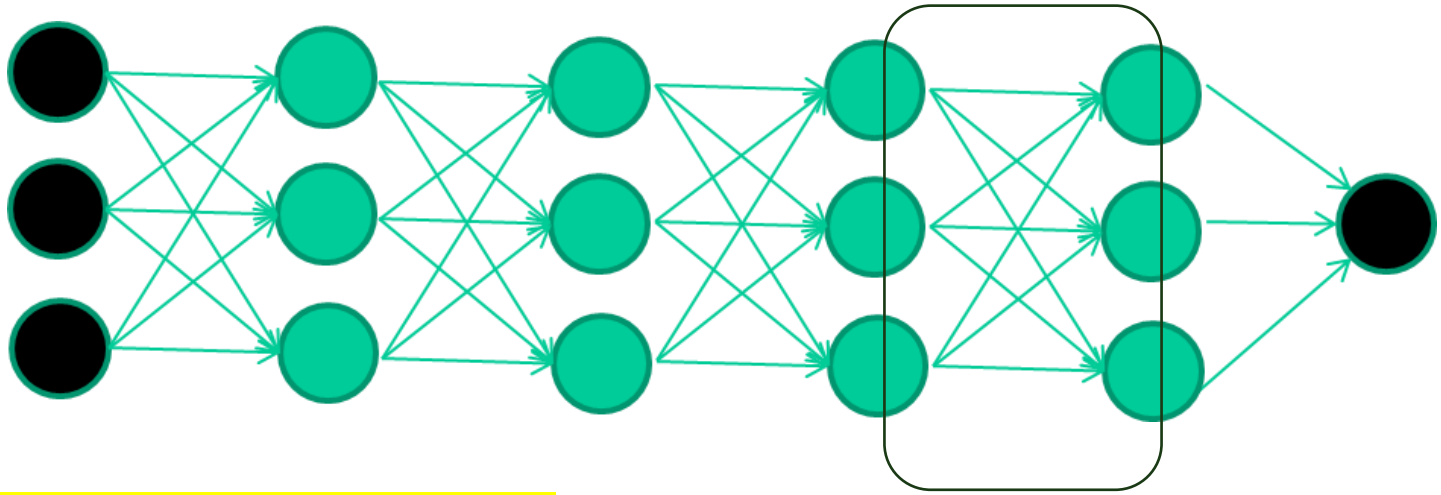


Train **this** layer first

then **this** layer

then **this** layer

The new way to train multi-layer NNs…



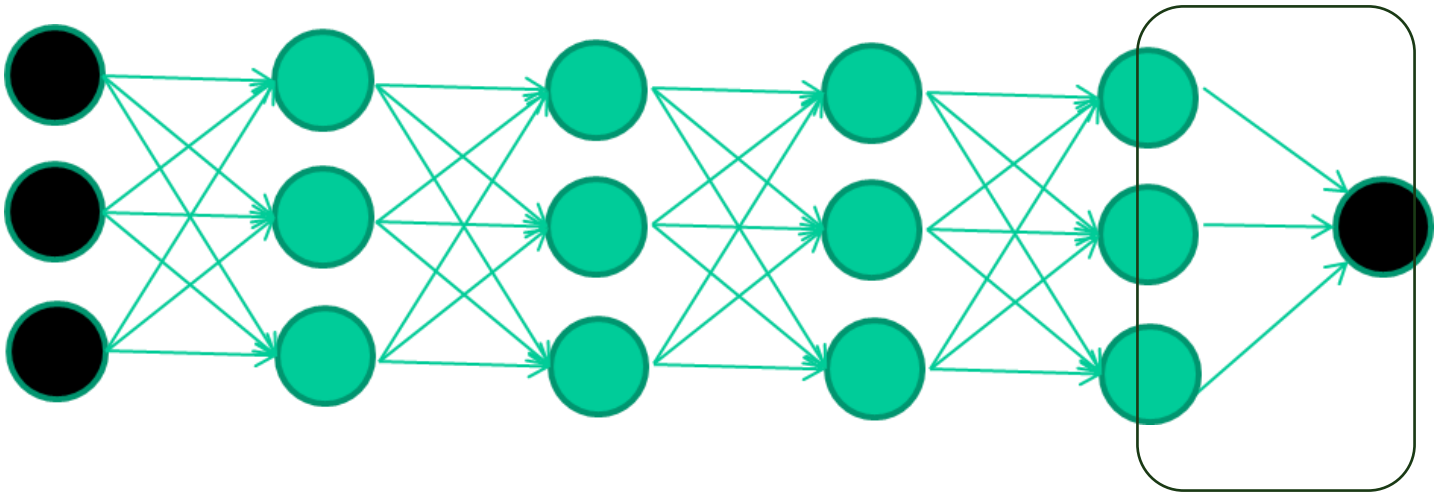
Train **this** layer first

then **this** layer

then **this** layer

then **this** layer

The new way to train multi-layer NNs…



Train **this** layer first

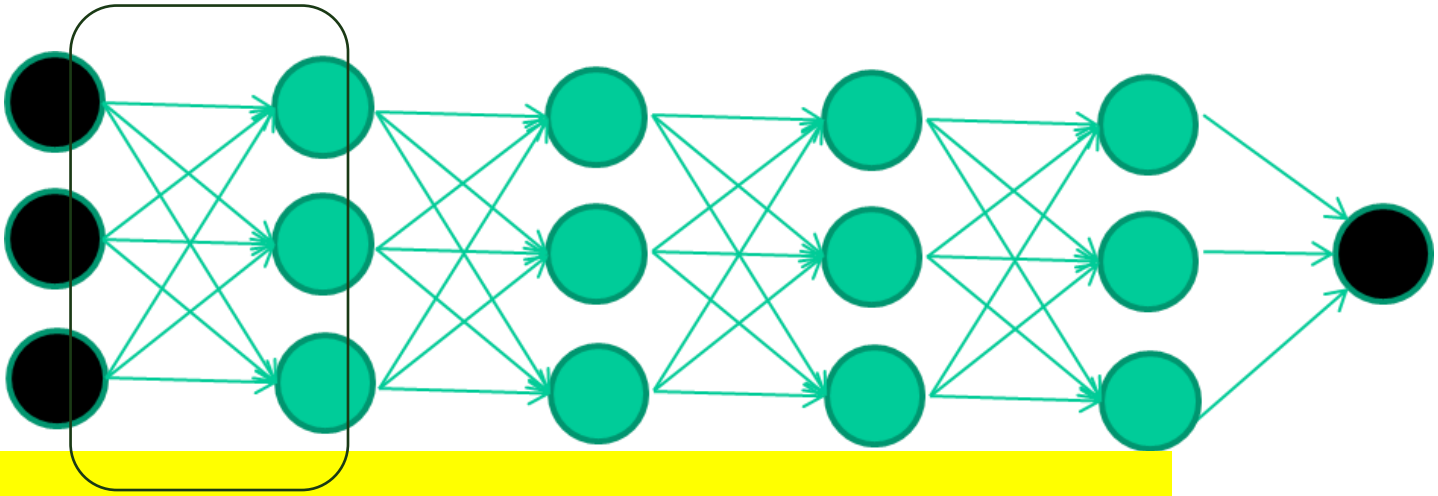
then **this** layer

then **this** layer

then **this** layer

finally **this** layer

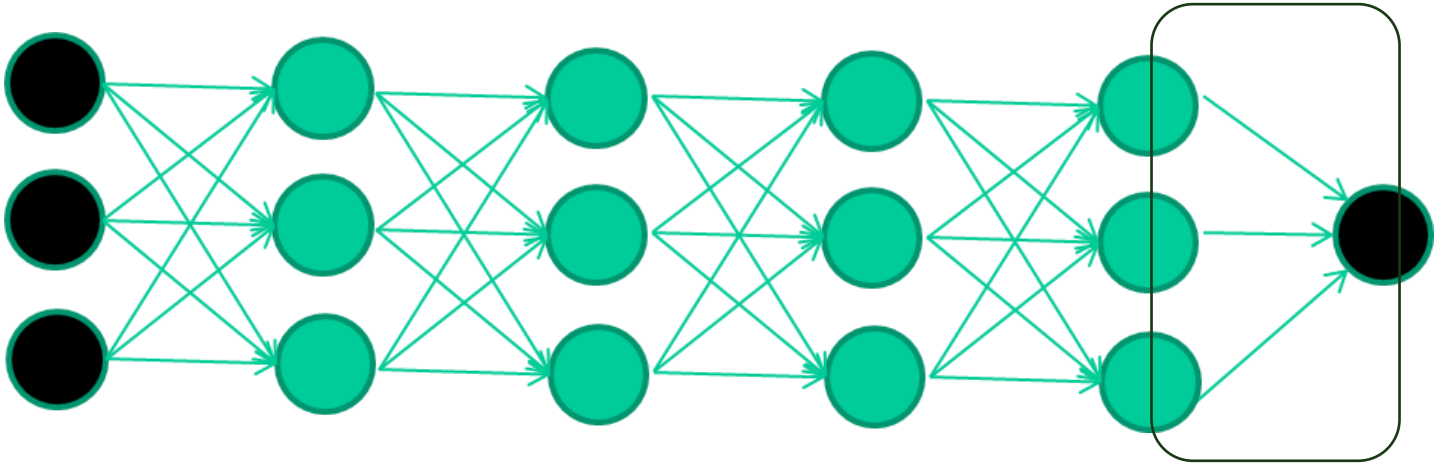
The new way to train multi-layer NNs…



EACH of the (non-output) layers is trained

to be an ***auto-encoder***

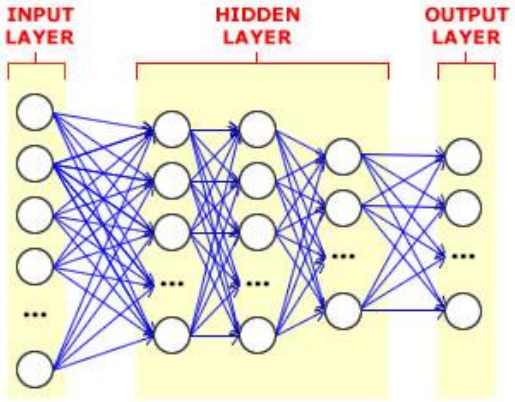
Basically, it is forced to learn good features that describe what comes from the previous layer



Final layer trained to predict class based on outputs from previous layers

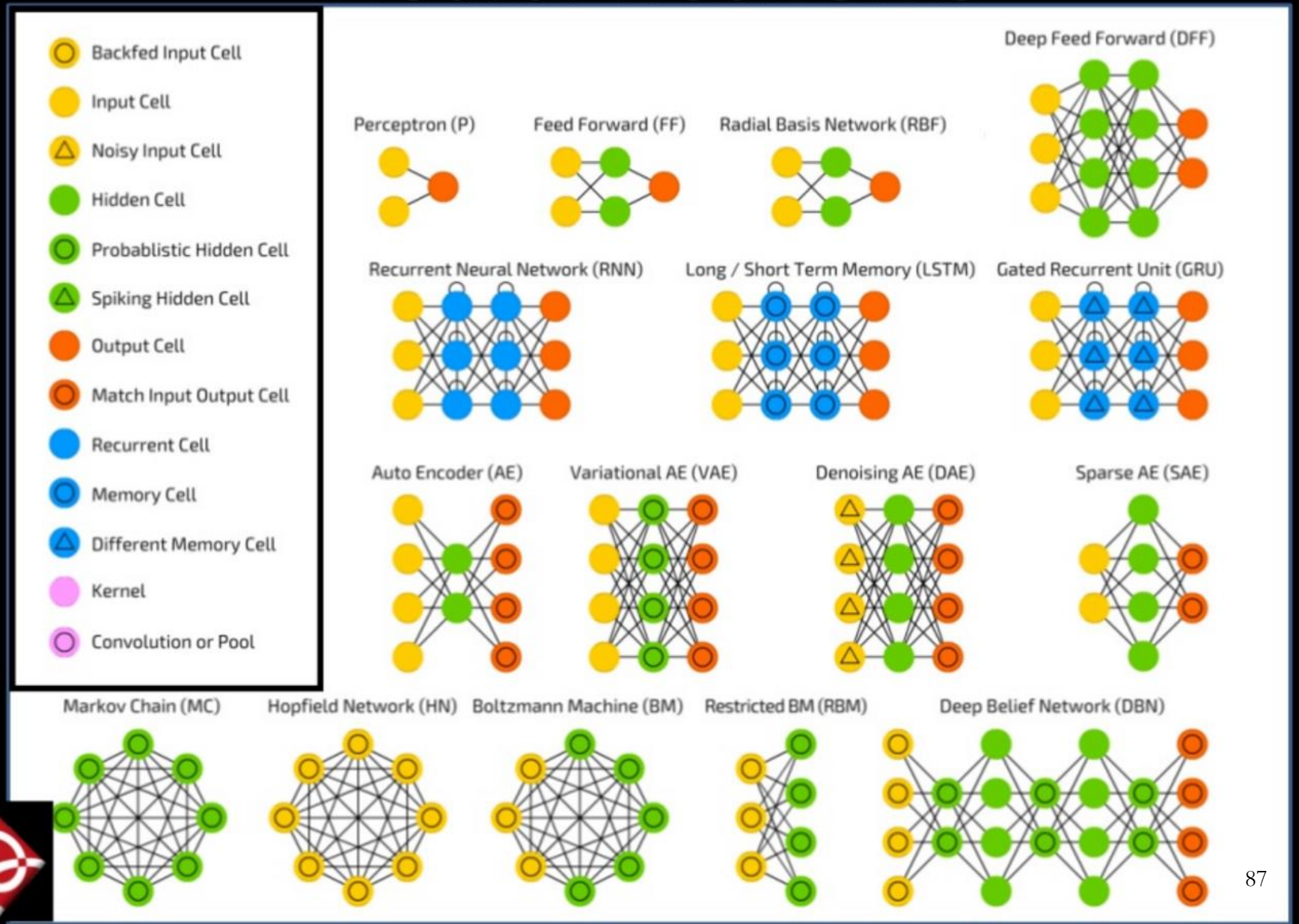
Basic Idea

- There are many, many types of training (learning) a deep learning network.
- Different kinds of autoencoder, variations on architectures and training algorithms, etc…
- Very fast growing area …

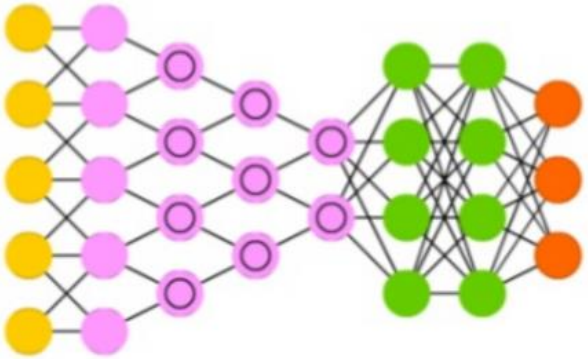


A Mostly Complete Chart of Neural Networks

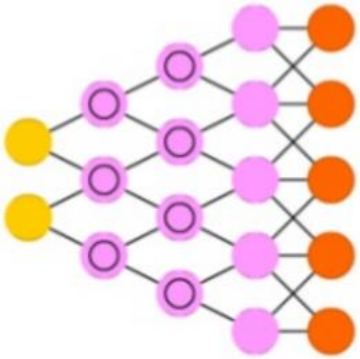
(1 of 2)



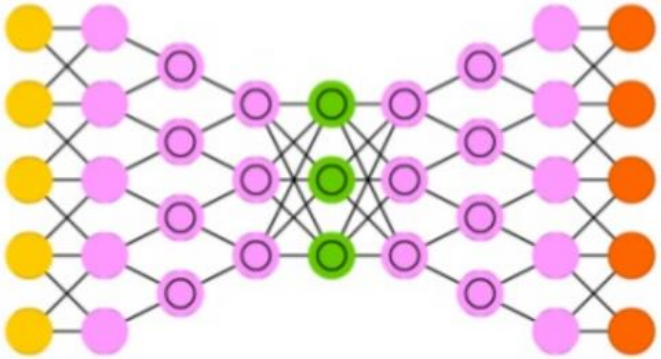
Deep Convolutional Network (DCN)



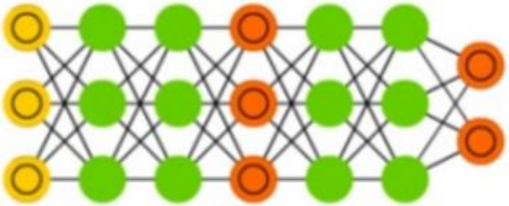
Deconvolutional Network (DN)



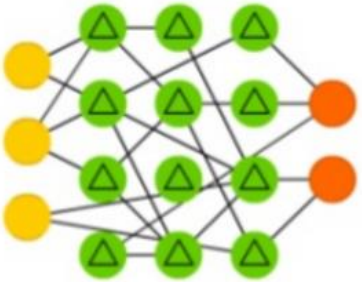
Deep Convolutional Inverse Graphics Network (DCIGN)



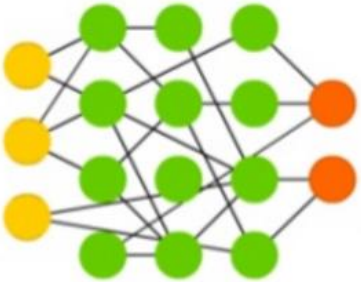
Generative Adversarial Network (GAN)



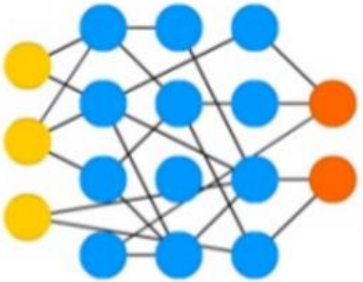
Liquid State Machine (LSM)



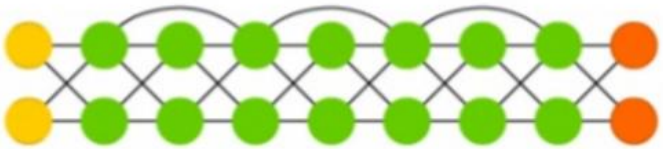
Extreme Learning Machine (ELM)



Echo State Network (ESN)



Deep Residual Network (DRN)



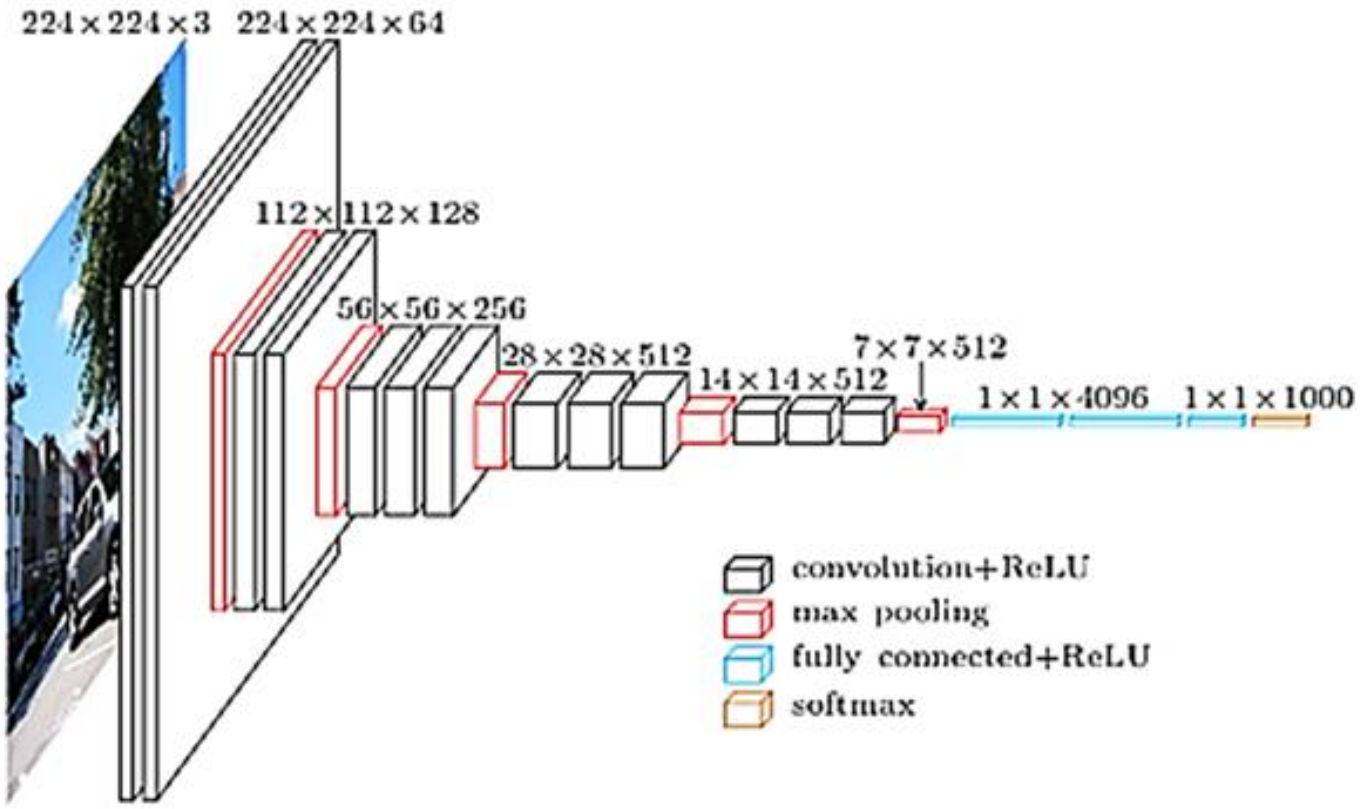
Kohonen Network (KN)

Support Vector Machine (SVM)

Neural Turing Machine (NTM)

CNN

CNNs

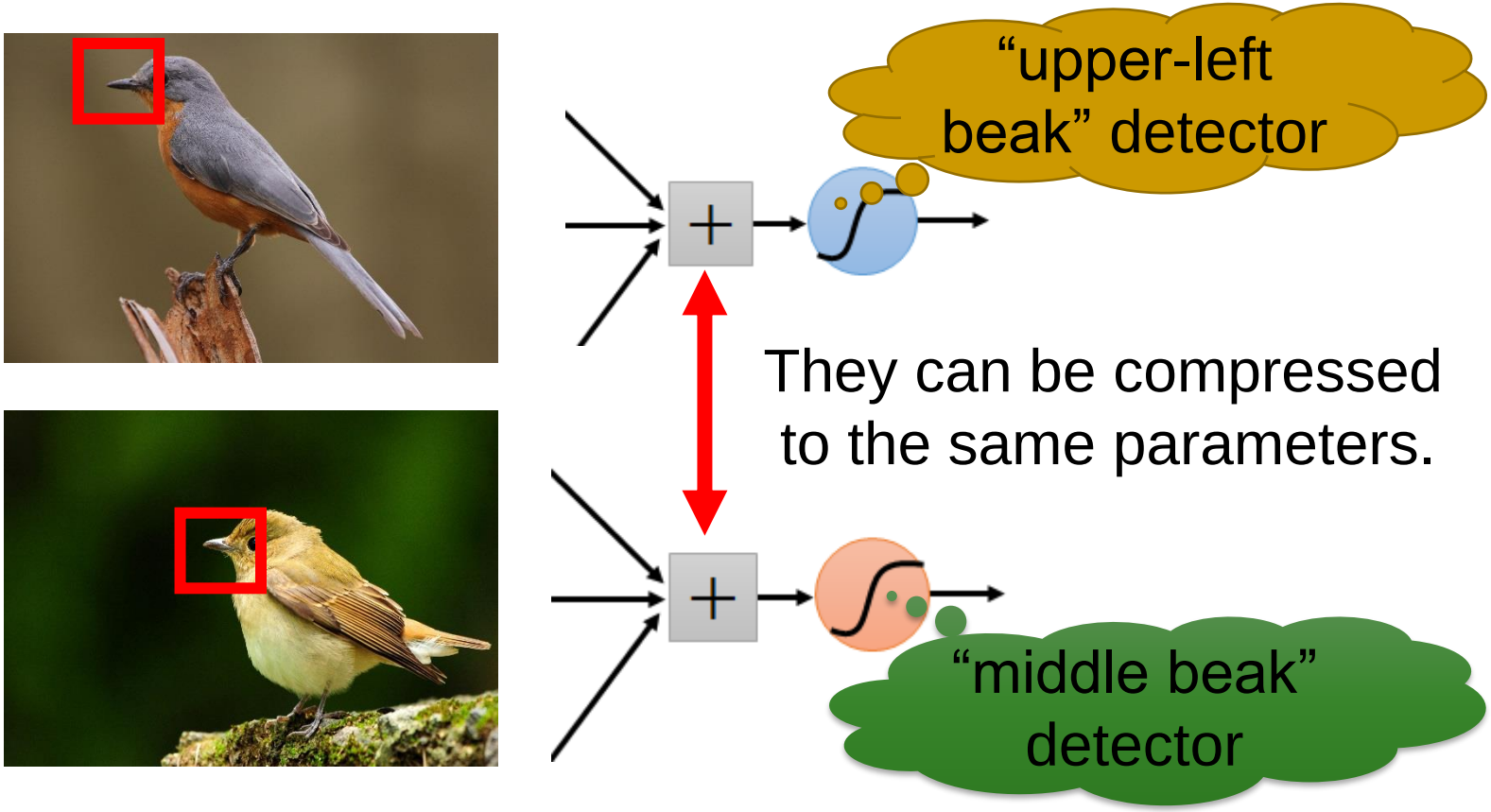

224×224×3
224×224×64
112×112×128
56×56×256
28×28×512
14×14×512
7×7×512
1×1×4096
1×1×1000
convolution+ReLU
max pooling
fully connected+ReLU
softmax

Consider learning an image:

- Some patterns are much smaller than the whole image

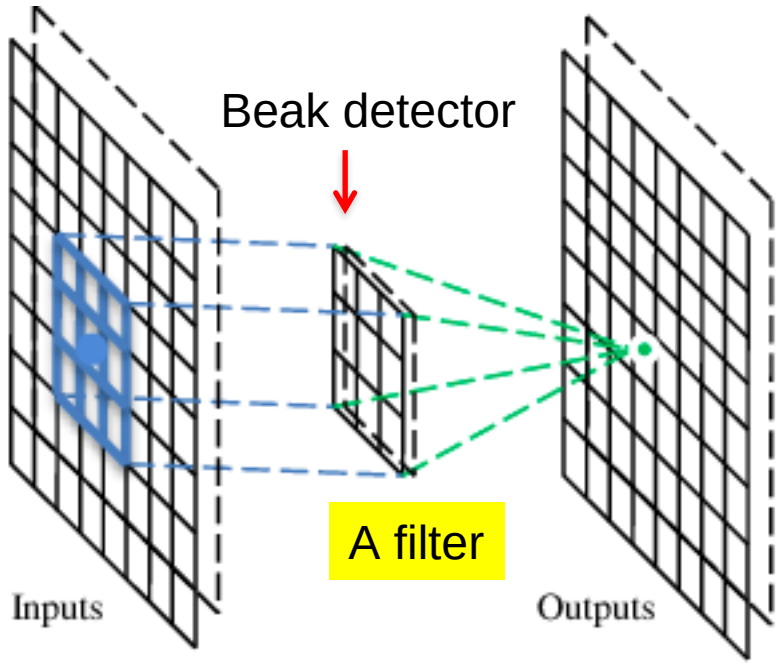
Can represent a small region with fewer parameters

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 do 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

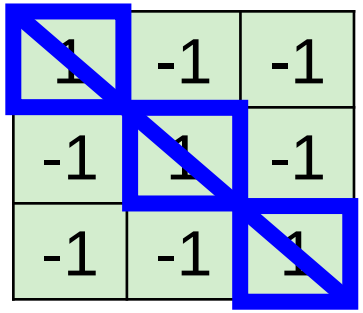
If stride=2

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

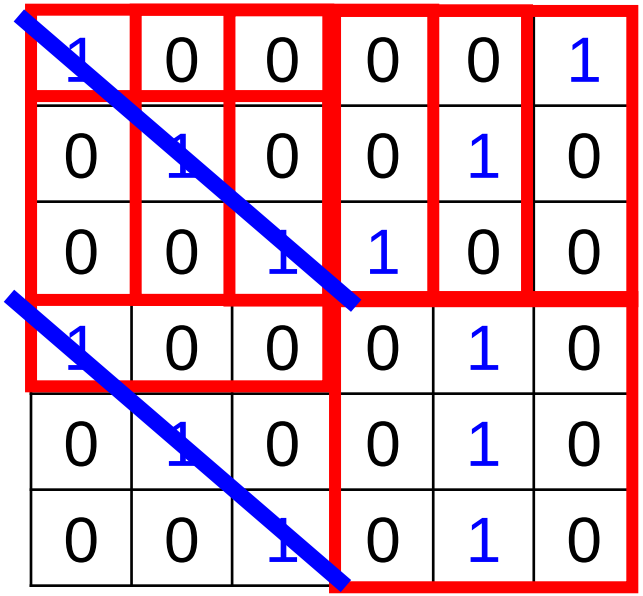
3 -3

Convolution

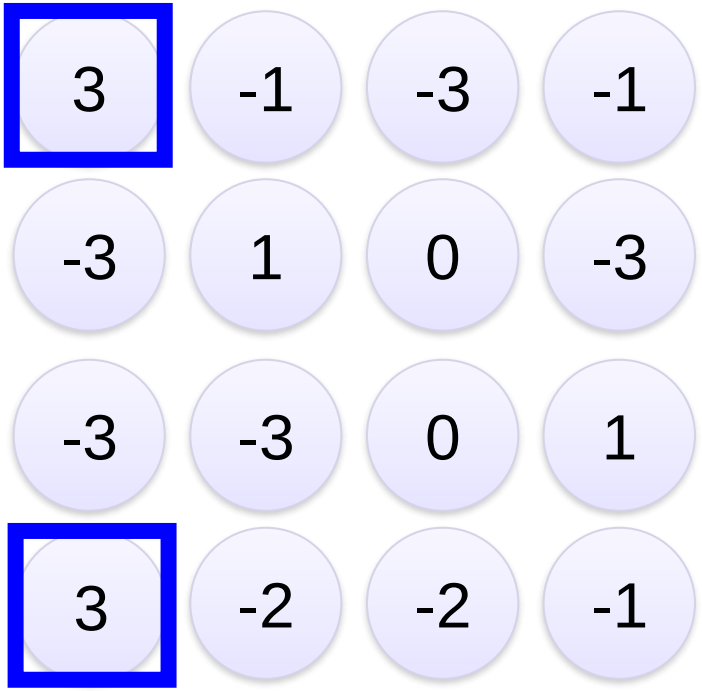


Filter 1

stride=1



6 x 6 image



Convolution

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

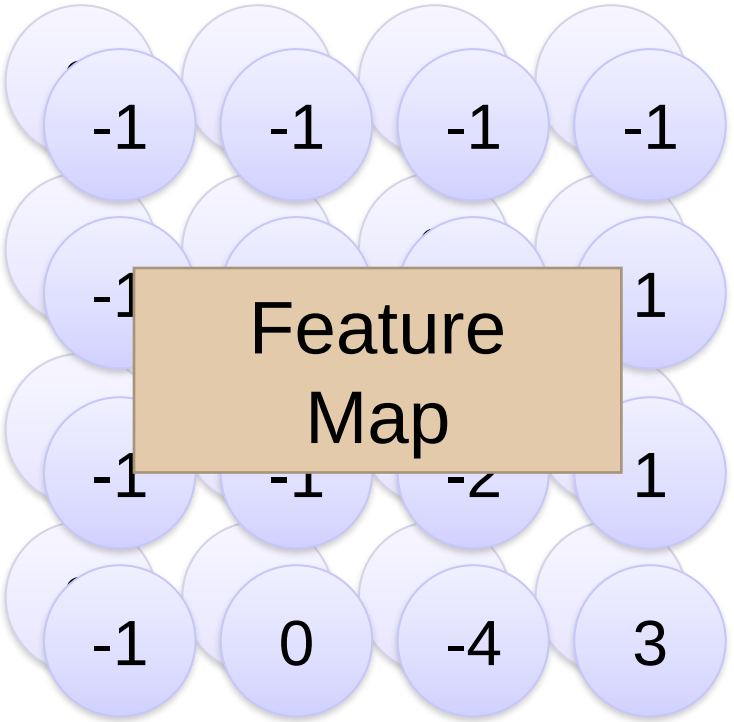
Filter 2

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

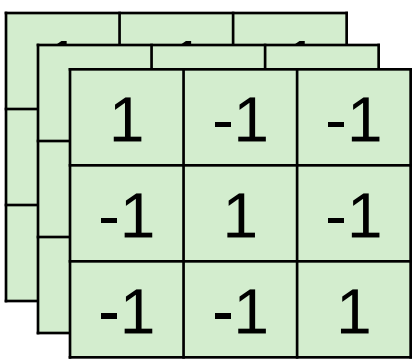
6 x 6 image

Repeat this for each filter

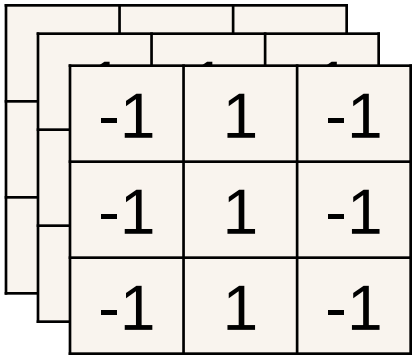


Two 4 x 4 images
Forming 2 x 4 x 4 matrix

Color image: RGB 3 channels

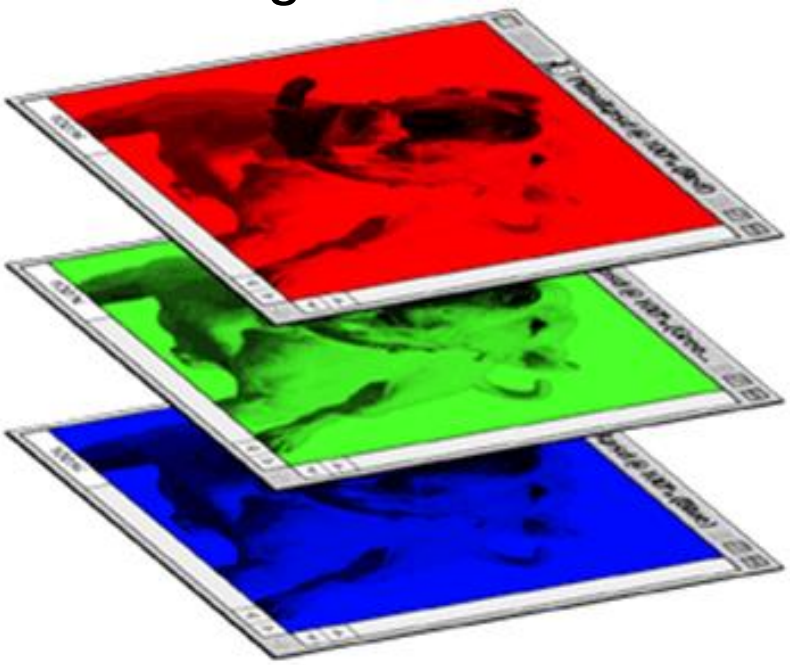


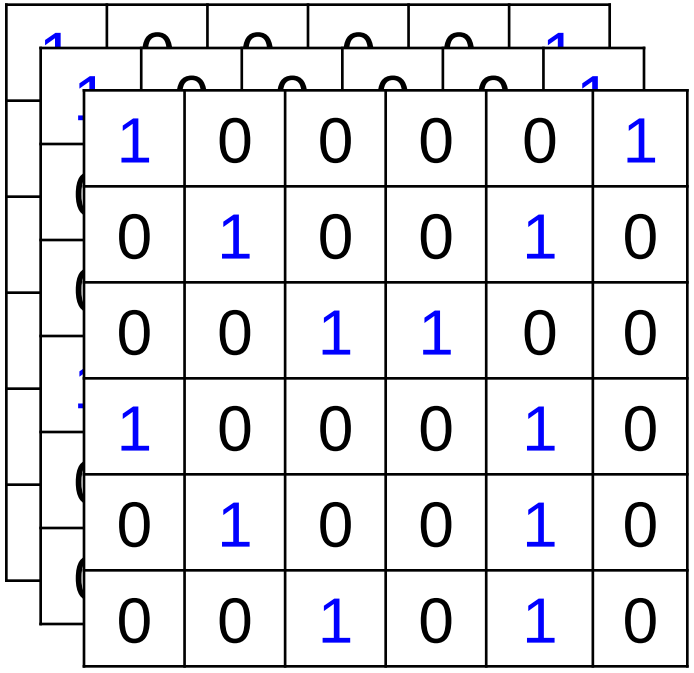
Filter 1



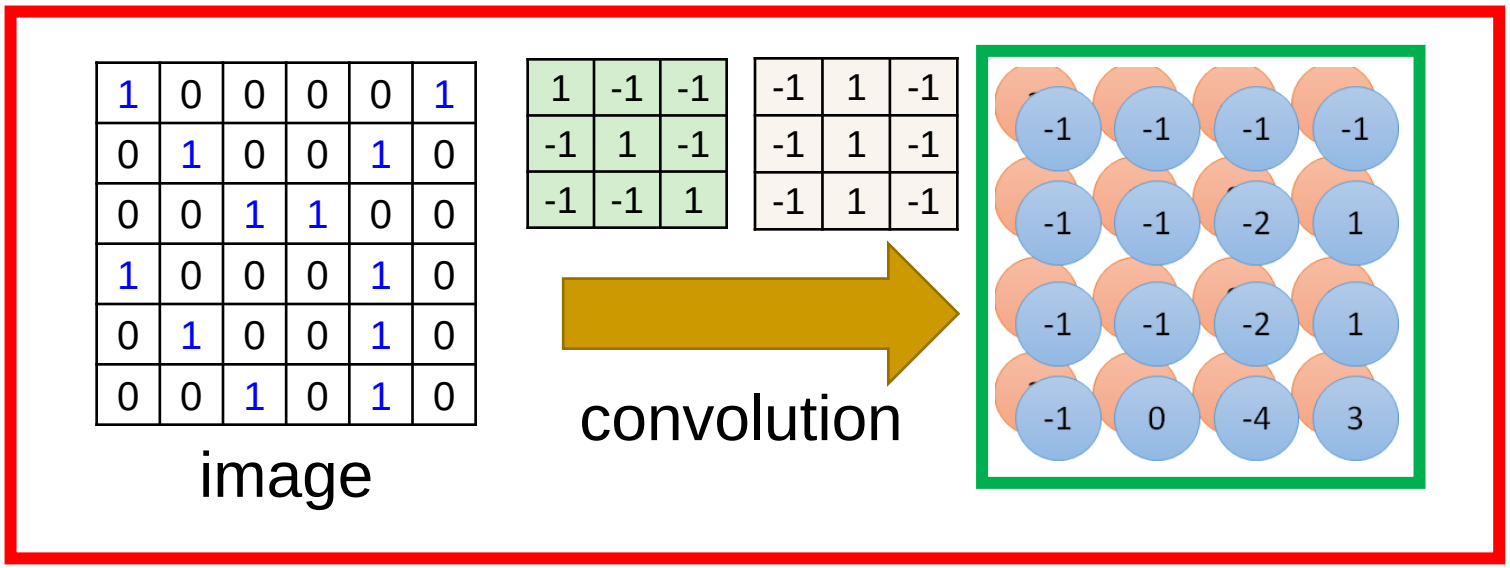
Filter 2

Color image



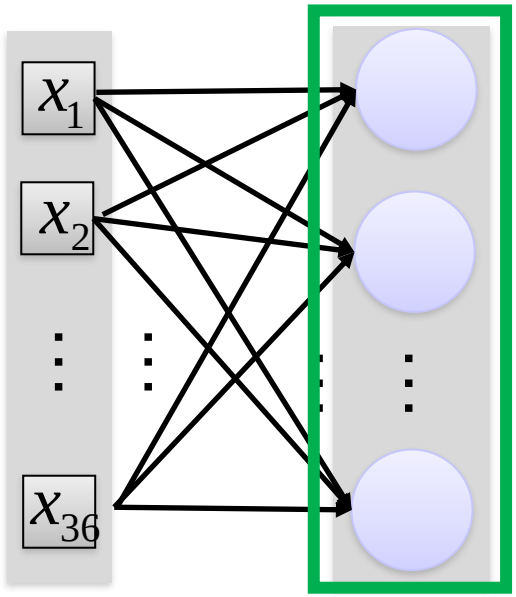


Convolution v.s. Fully Connected

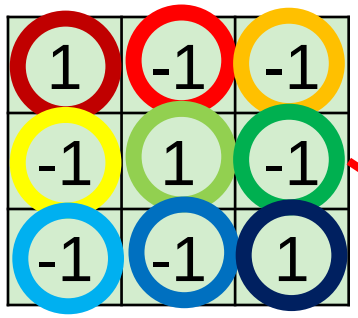


Fully-connected

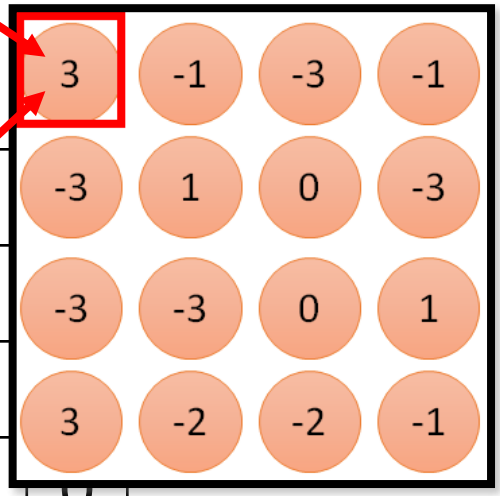
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0



How to reduce parameters?



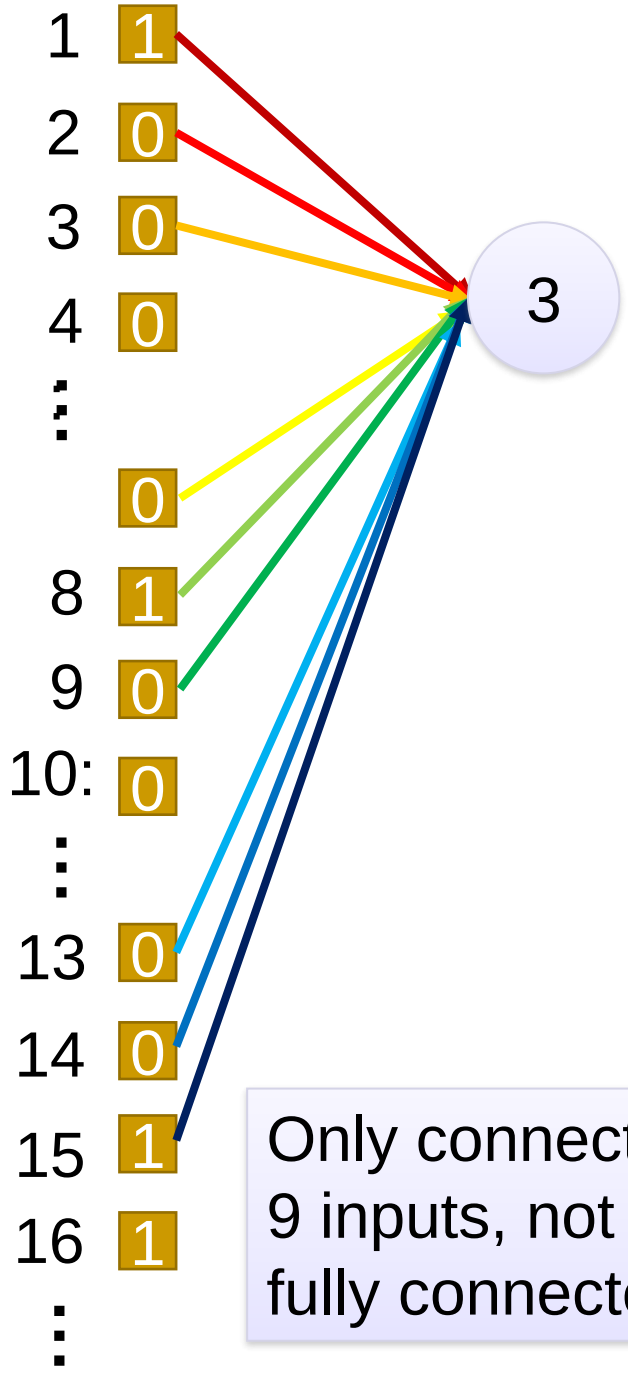
Filter 1



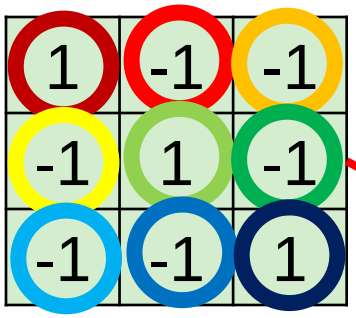
1	0	0	0	0	
0	1	0	0	1	
0	0	1	1	0	
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

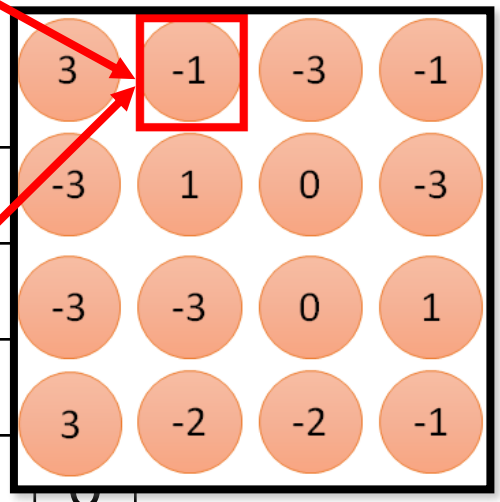
fewer parameters!



Only connect to 9 inputs, not fully connected



Filter 1

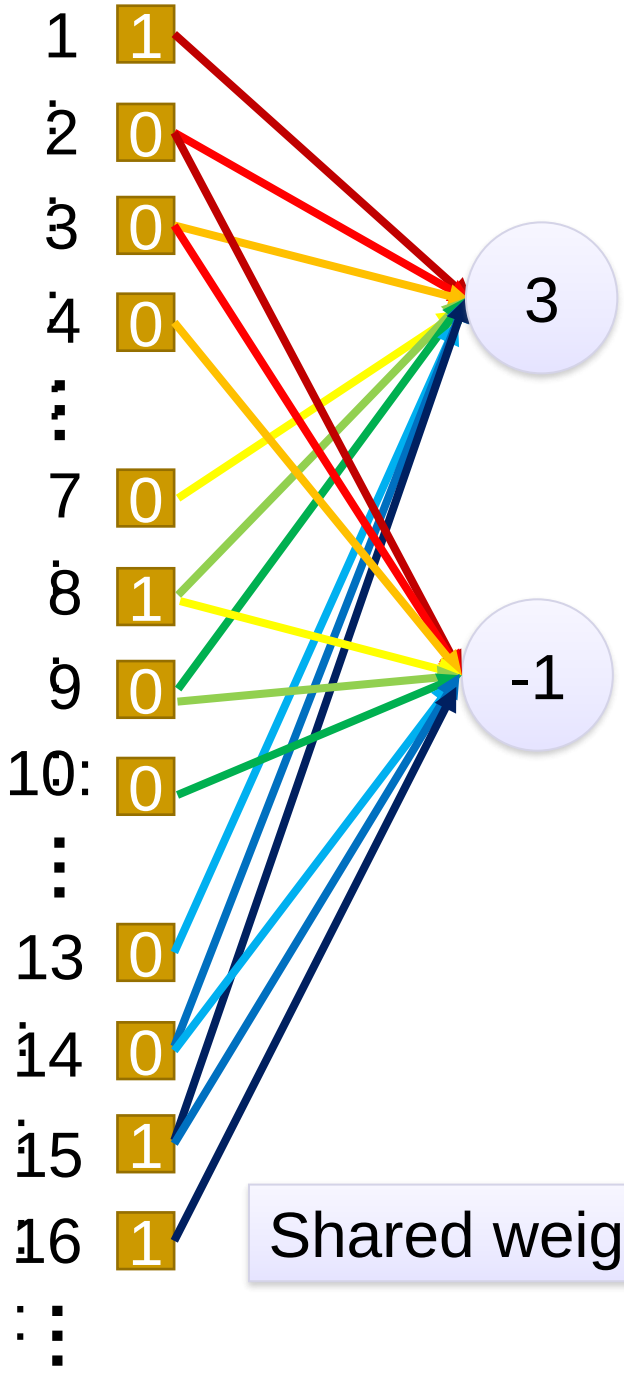


1	0	0	0	0	
0	1	0	0	1	
0	0	1	1	0	
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

Fewer parameters

Even fewer parameters



Shared weights

CNN – so far

Convolution

The whole CNN

Convolution

Max Pooling

Convolution

Max Pooling

Can repeat many times

Flattened

Fully Connected Feedforward network

cat dog ……

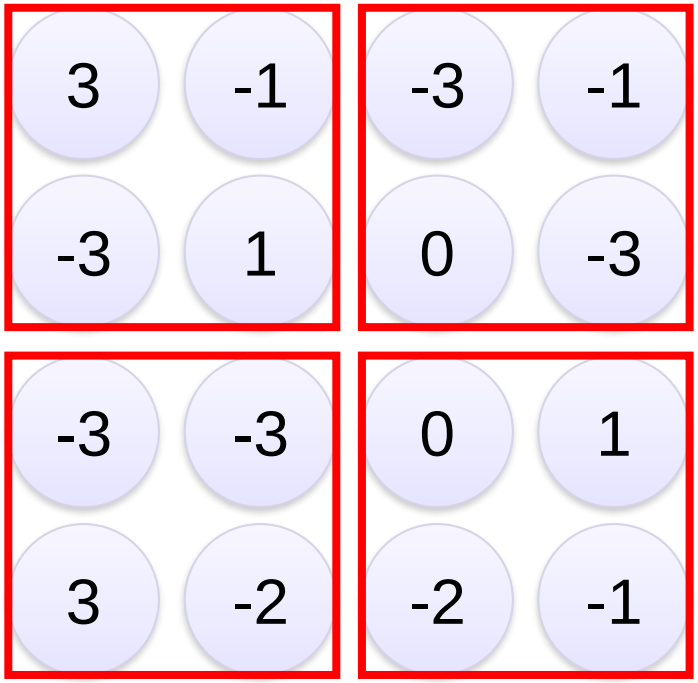
Max Pooling

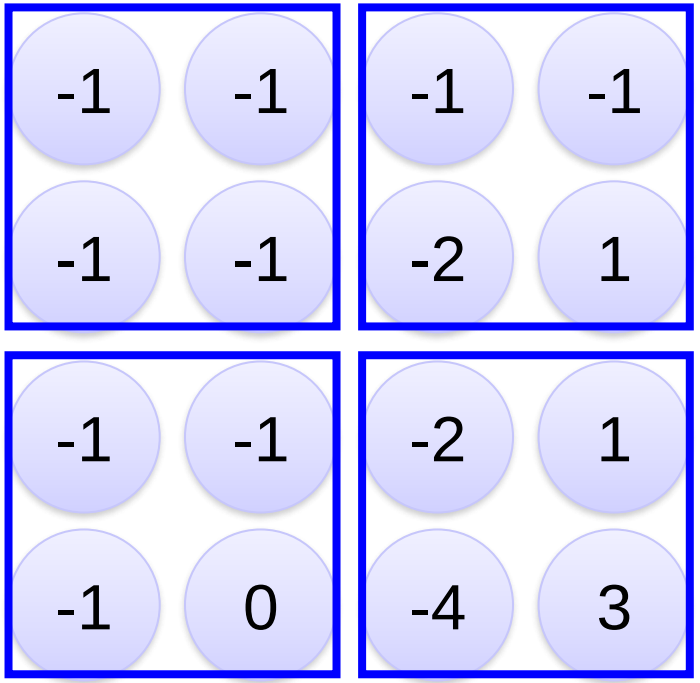
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

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

Filter 2





Why Pooling

- Subsampling pixels will not change the object

bird

Subsampling

bird

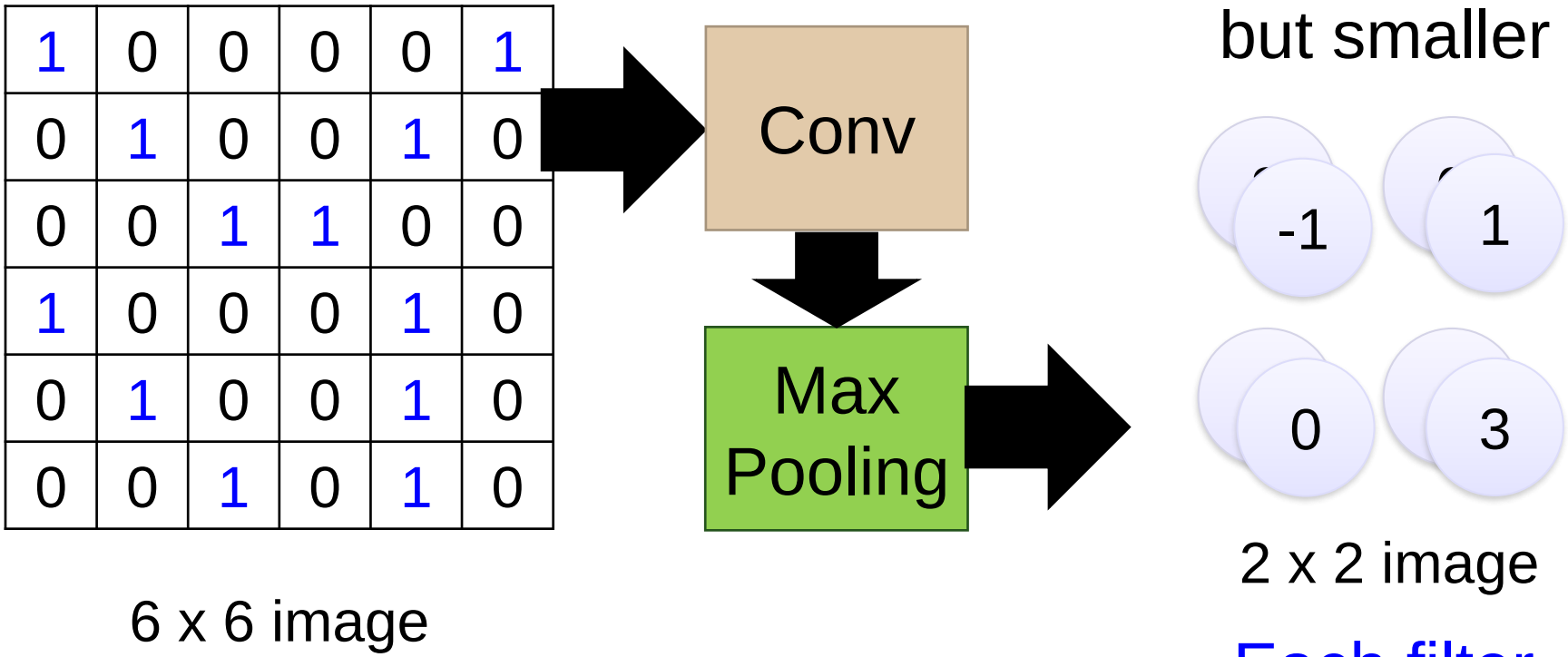
We can subsample the pixels to make image smaller

→ fewer parameters to characterize the image

A CNN compresses a fully connected network in two ways:

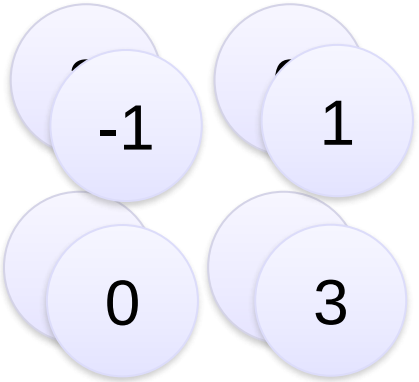
- Reducing number of connections
- Shared weights on the edges
- Max pooling further reduces the complexity

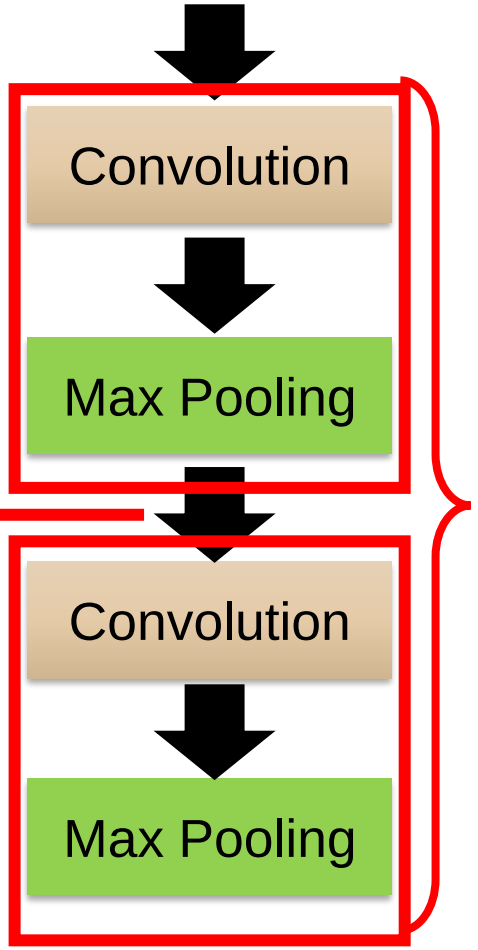
Max Pooling



Each filter is a channel

The whole CNN

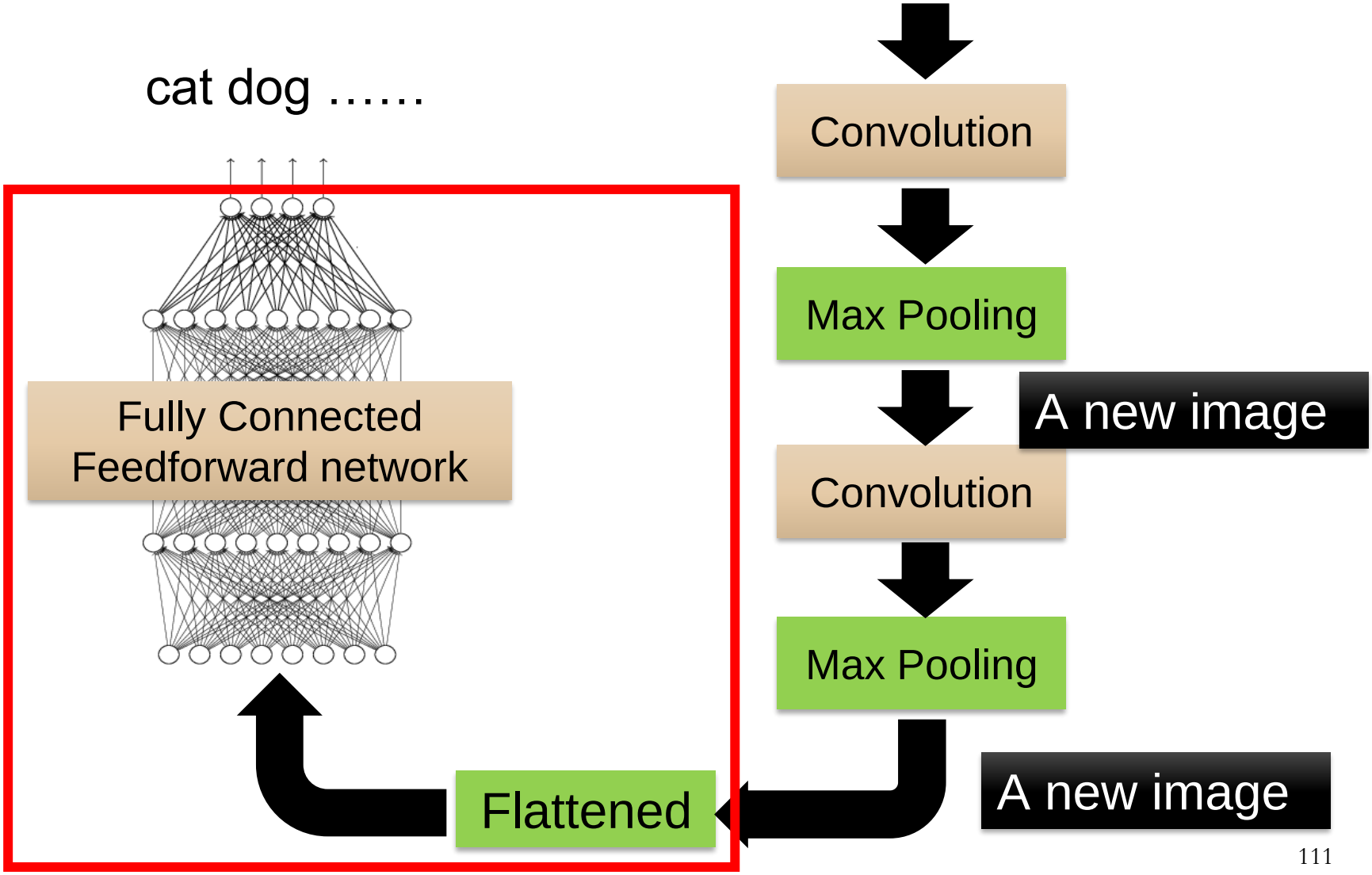




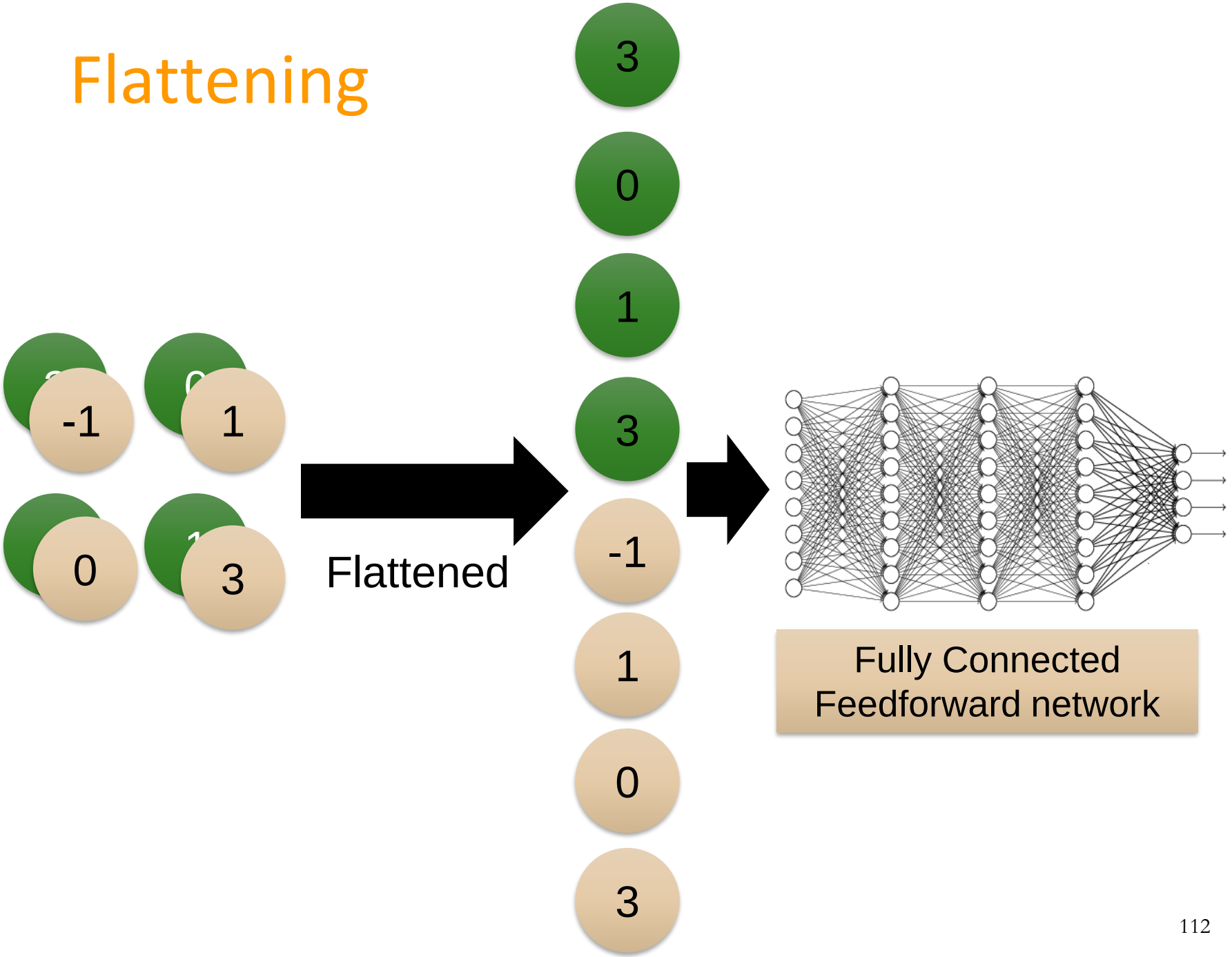
Smaller than the original image

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

Keras: The Python Deep Learning library

You have just found Keras.

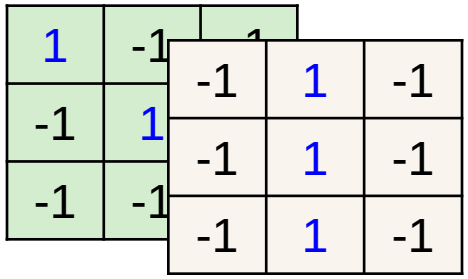
Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation.
Being able to go from idea to result with the least possible delay is key to doing good research.

https://keras.io/

CNN in Keras

Only need to modify the ***network structure*** and ***input format***

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



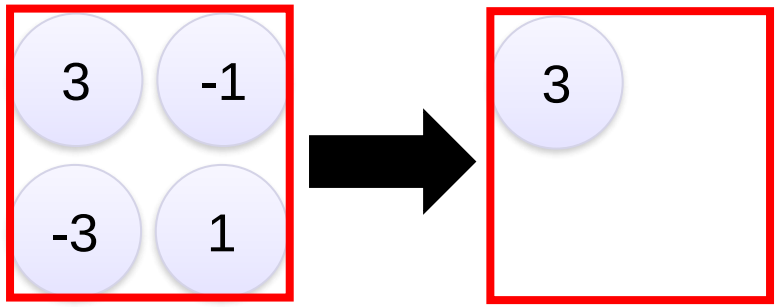
There are 25 3x3 filters.

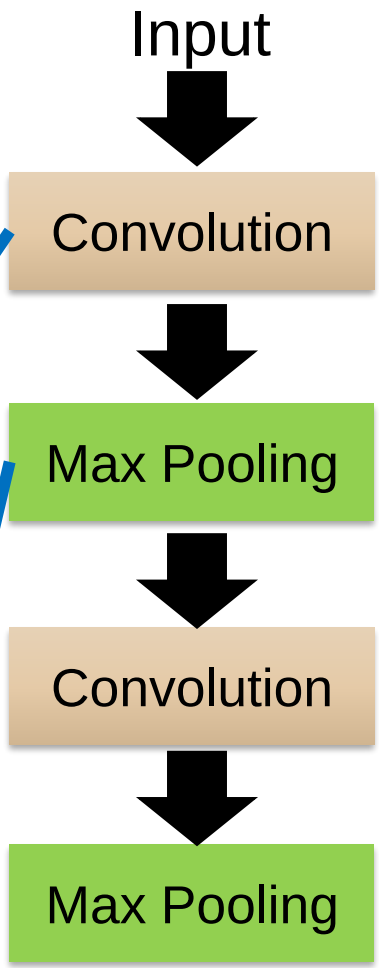
Input_shape = (28 , 28 , 1)

28 x 28 pixels

1: black/white, 3: RGB

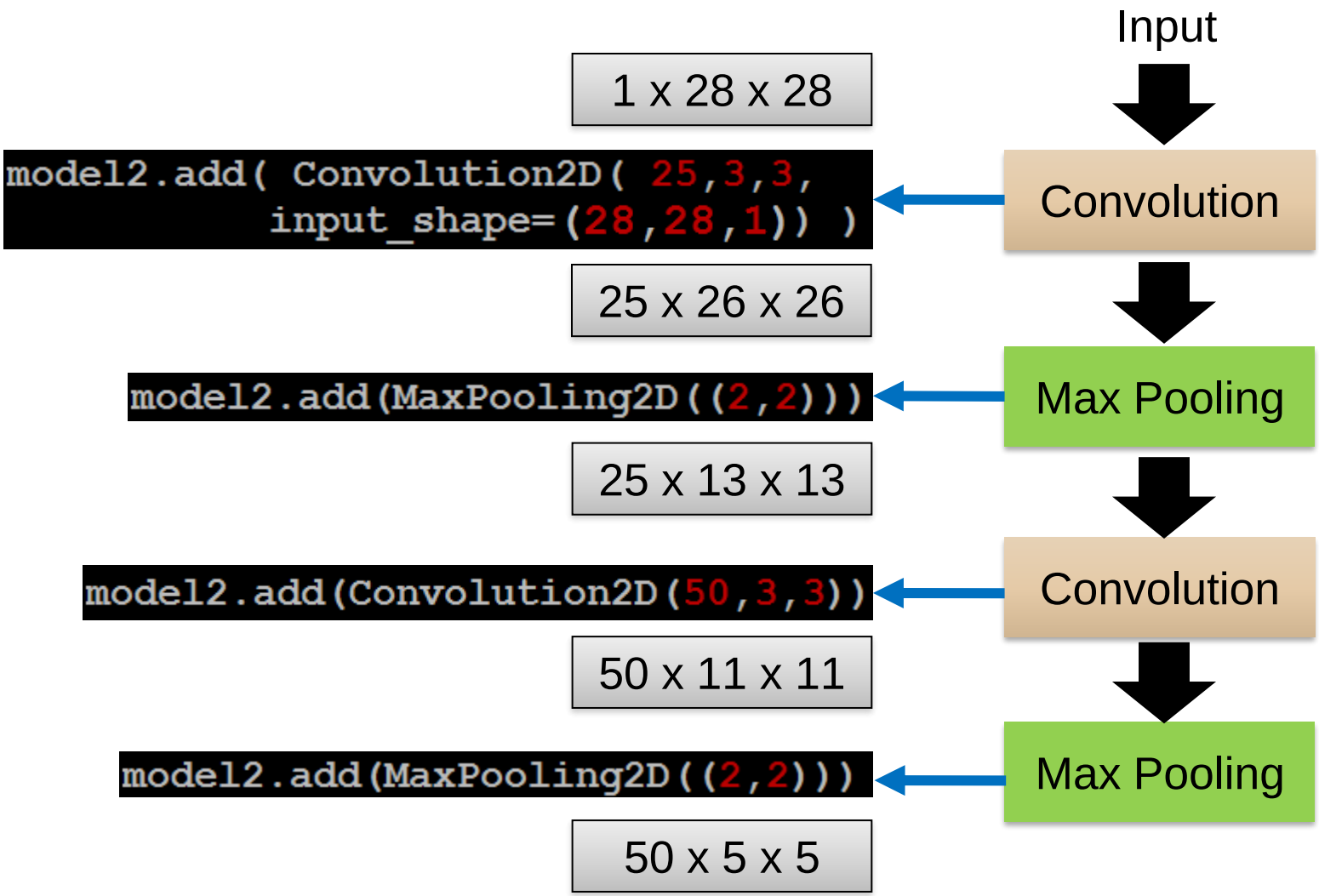
```
model2.add(MaxPooling2D((2,2)))
```





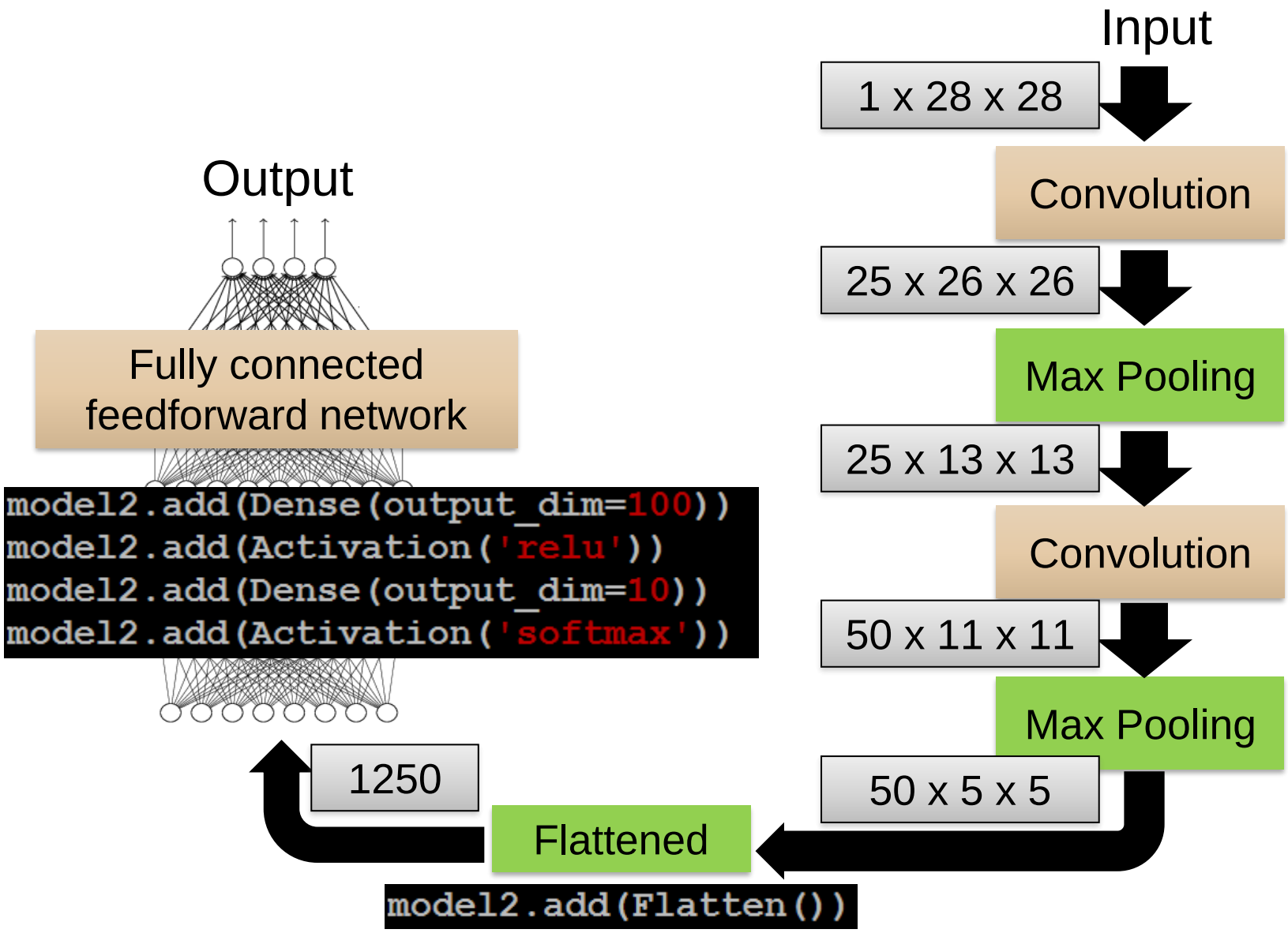
CNN in Keras

Only need to modify the ***network structure*** and ***input format***



CNN in Keras

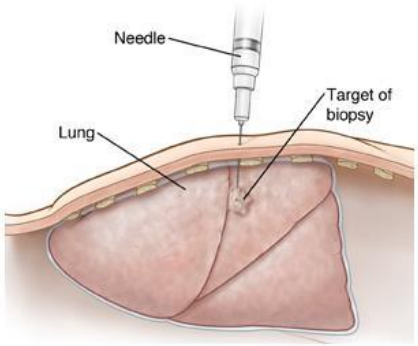
Only need to modify the ***network structure*** and ***input format***



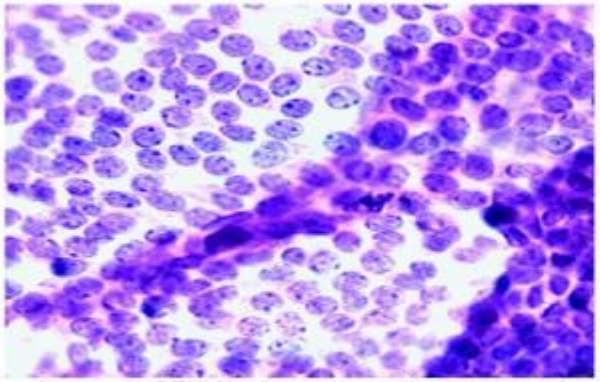
THIS WEEK: PROJECT 3

Project Overview

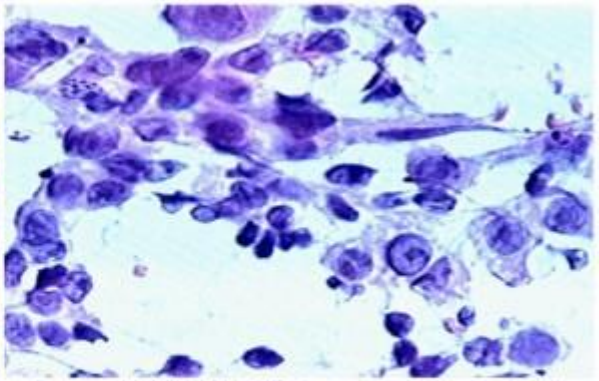
- Imagine, you are given data about patients having some suspicious tissue, e.g. in lung or breast.
- Digitized images are given and have labels “benign” or “malignant”.



For the biopsy, a thin needle is used to remove samples of tissue from an abnormal area in the lung.



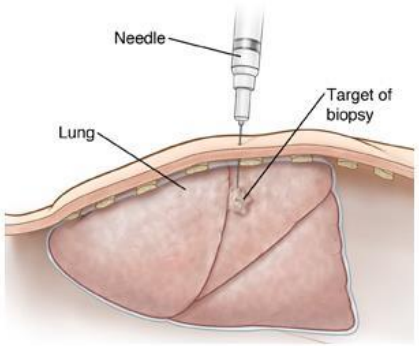
Smear with BENIGN diagnosis – uniform nucleus of cells, symmetrical, homogeneous, with areas within normal size



Smear with MALIGNANT diagnosis – nucleus of cells without uniformity, asymmetrical, not homogeneous (multiple sizes) and with areas above normal size

Project Overview

- Imagine, you are given data about patients having some suspicious tissue, e.g. in lung or breast.
- Digitized images are given and have labels “benign” or “malignant”.
- You want to build a tool that can help medical doctors to decide whether a sample is benign or malign.
- How would you do that?



For the biopsy, a thin needle is used to remove samples of tissue from an abnormal area in the lung.