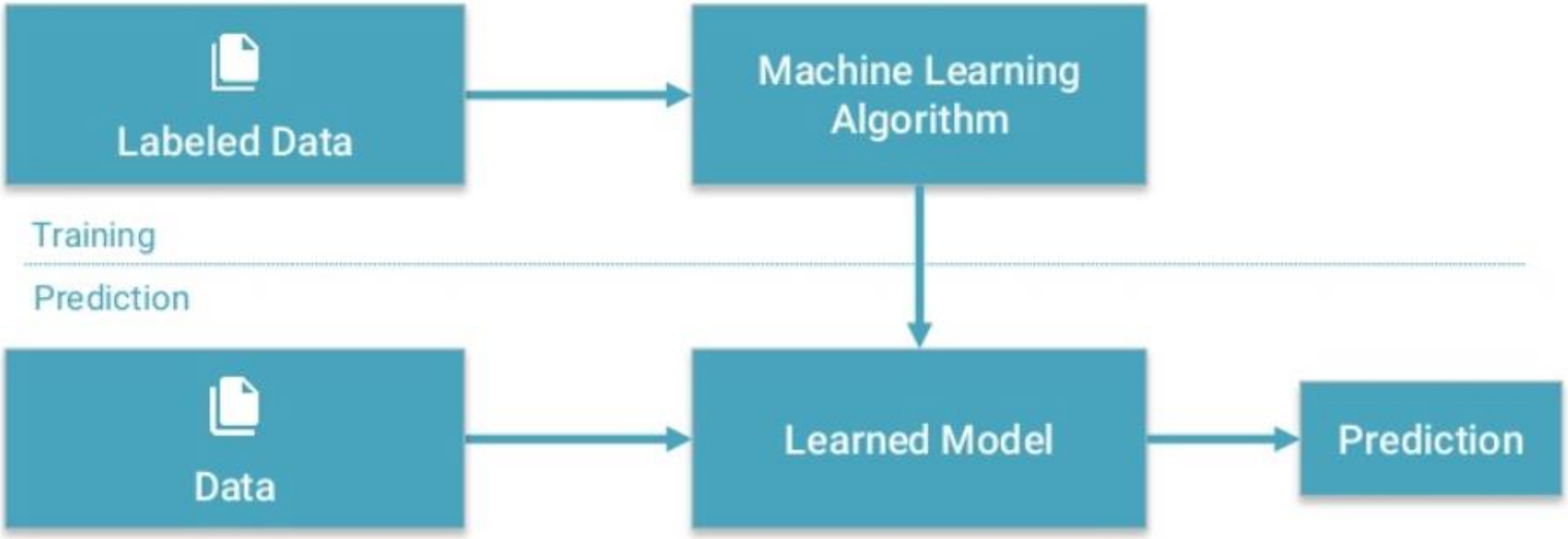
Deep Learning for Computer Vision

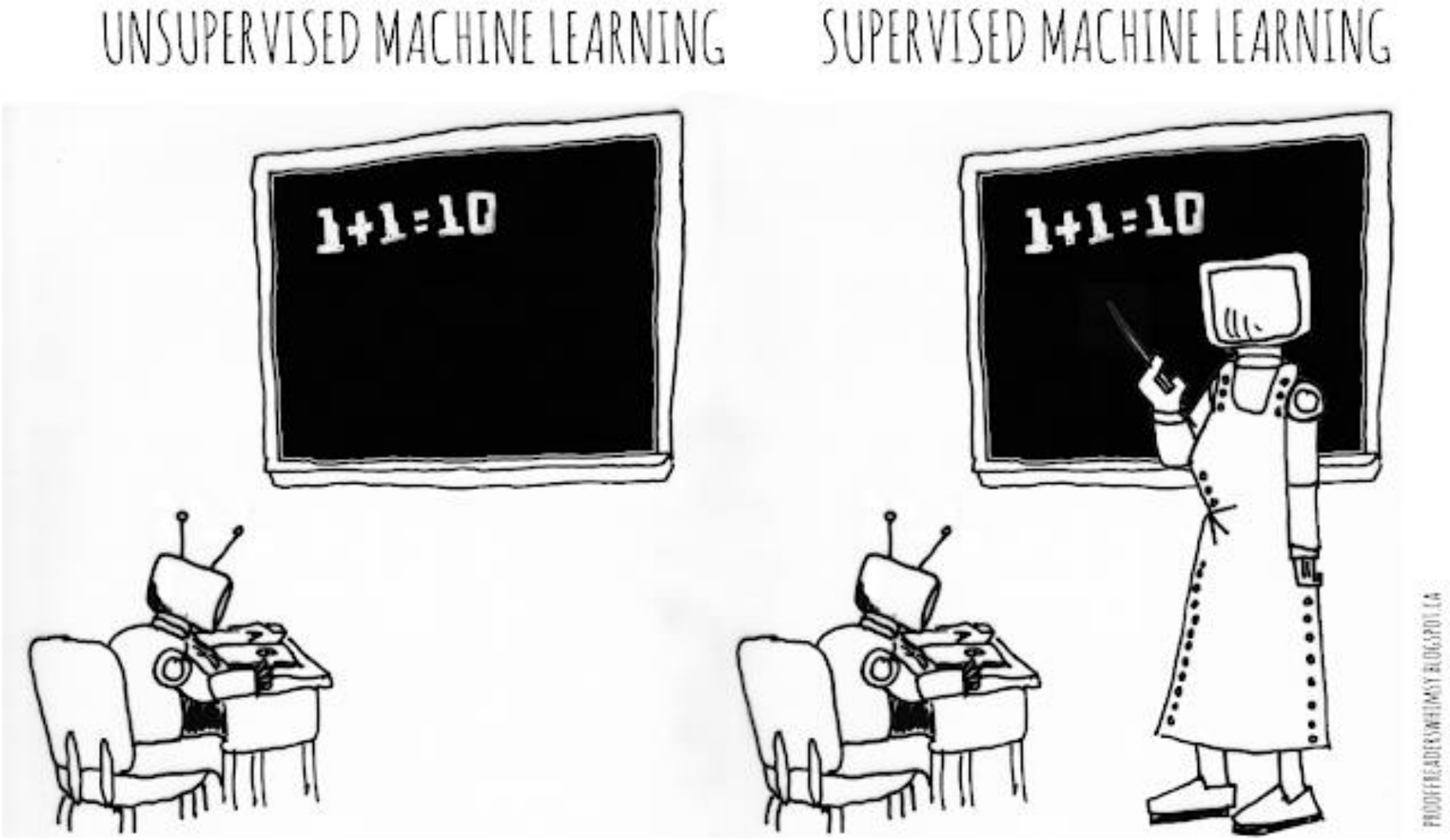
Santosh Chapaneri

Machine Learning

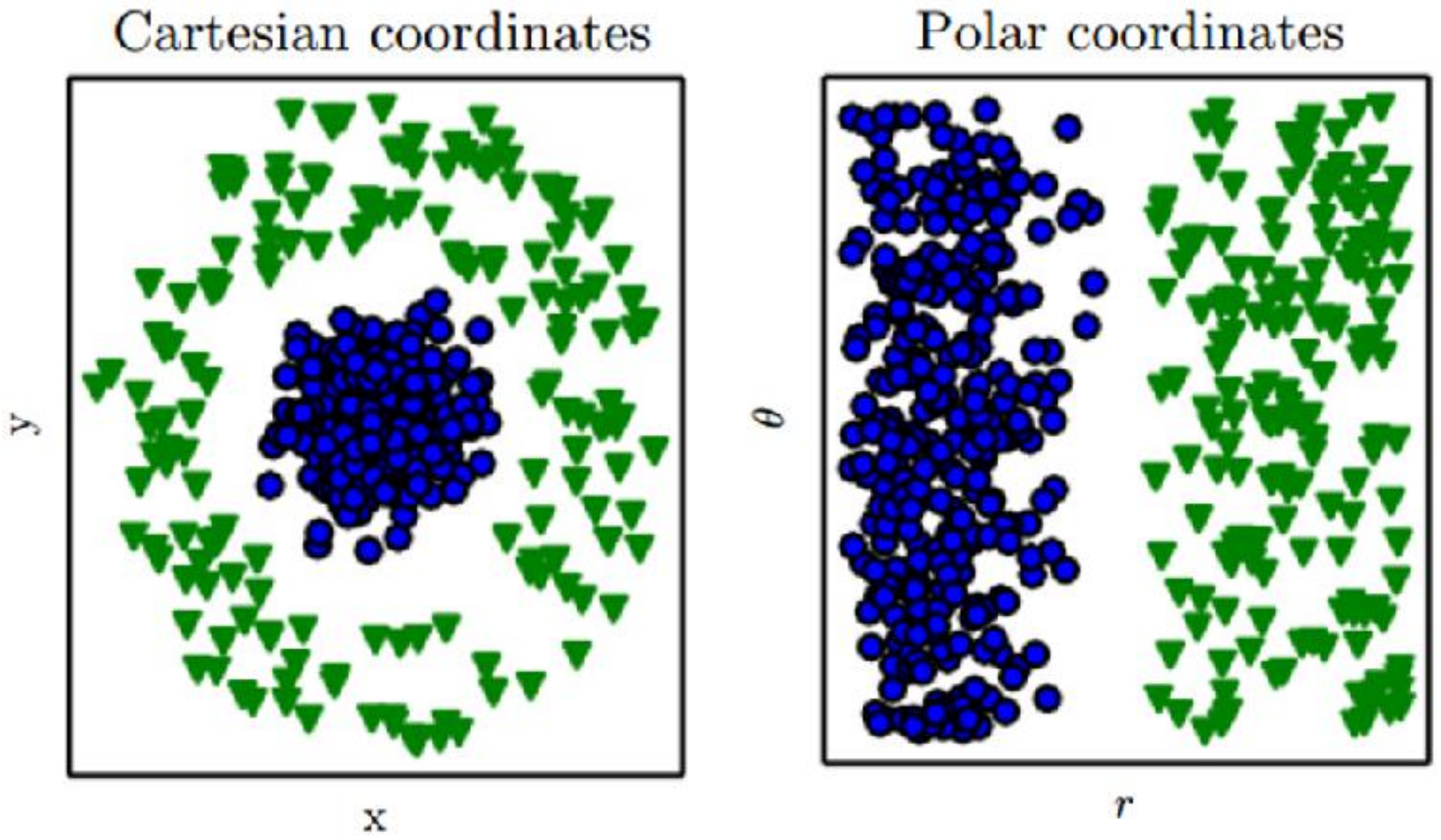


Various techniques that can learn from and make predictions on data

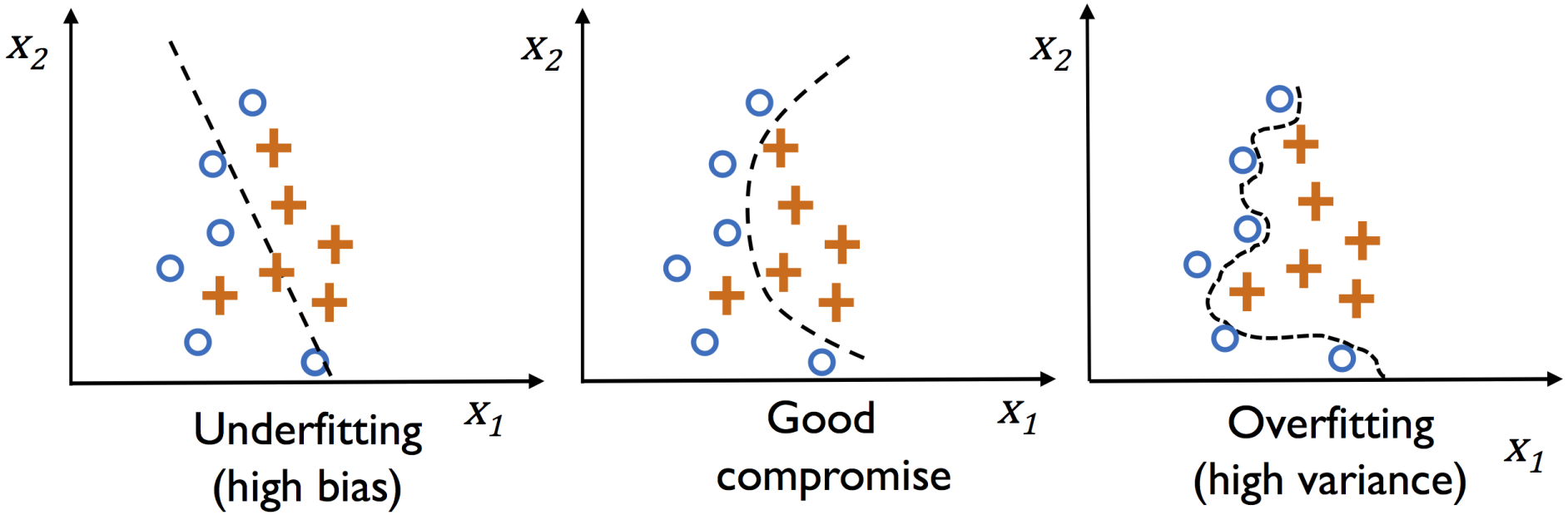
Machine Learning



Feature Space



Over-fitting



Traditional Engineering – ML

VISION

hand-crafted features SIFT/HOG

fixed

your favorite classifier

learned

"car"

SPEECH

hand-crafted features MFCC

fixed

your favorite classifier

learned

\'d ē p\

NLP

This burrito place is yummy and fun!

hand-crafted features Bag-of-words

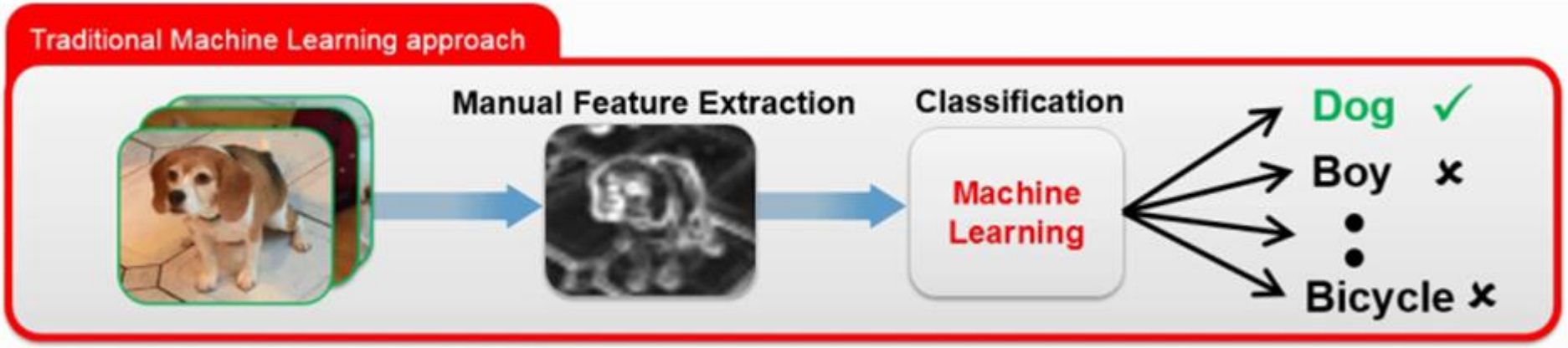
fixed

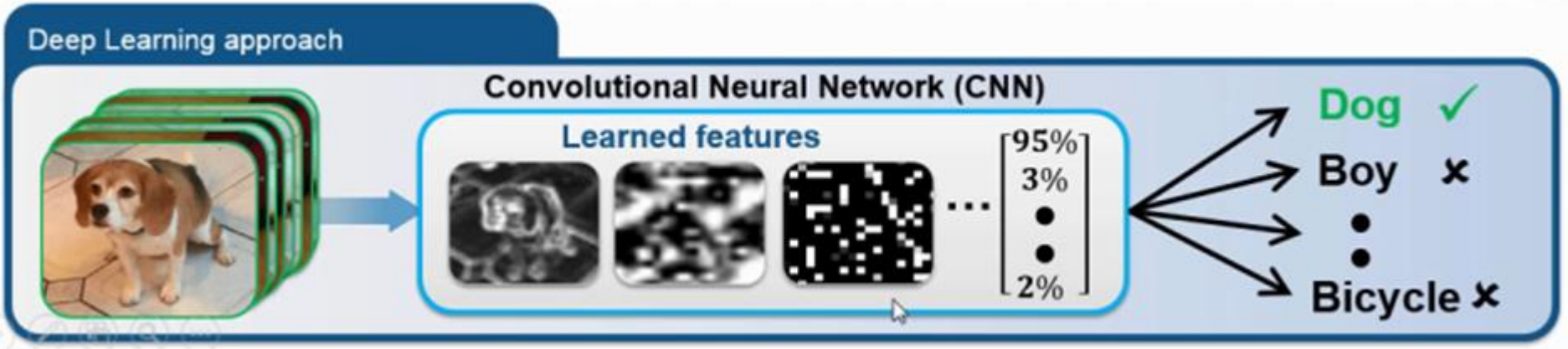
your favorite classifier

learned

"+"

Shallow v/s Deep Learning





DL – A Brief History of Time

1958 Perceptron

1969
Perceptron criticized

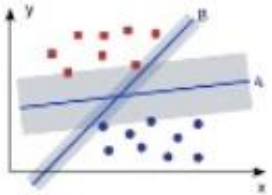
1974 Backpropagation

awkward silence (AI Winter)

Convolution Neural Networks for Handwritten Recognition

1998

1995
SVM reigns



2006
Restricted Boltzmann Machine

Google Brain Project on 16k Cores

2012

2012
AlexNet wins ImageNet

IMAGENET

DL – What changed?

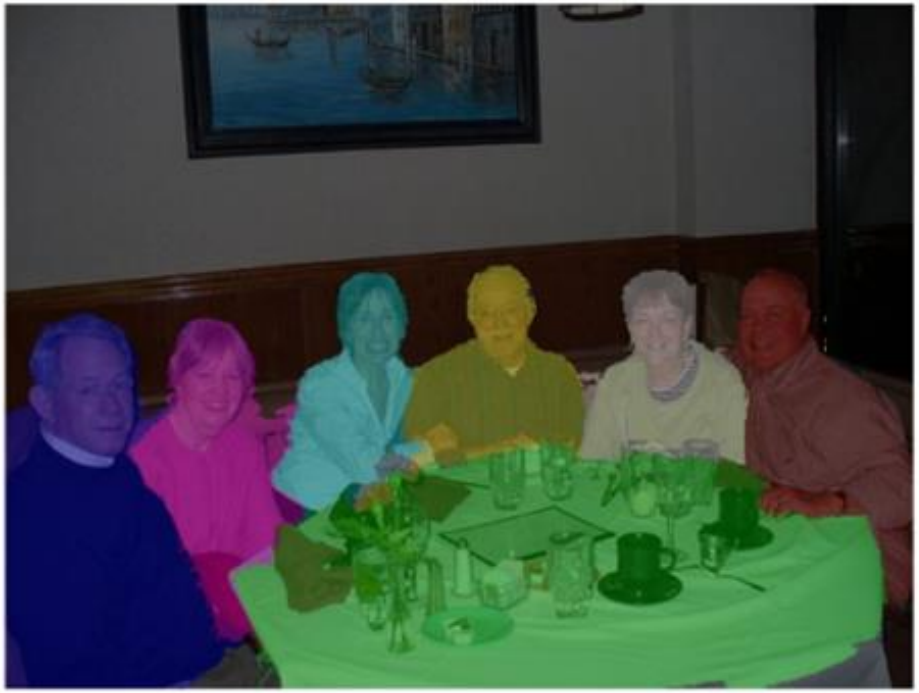
Big Data

Computation
(Moore's law, GPUs)

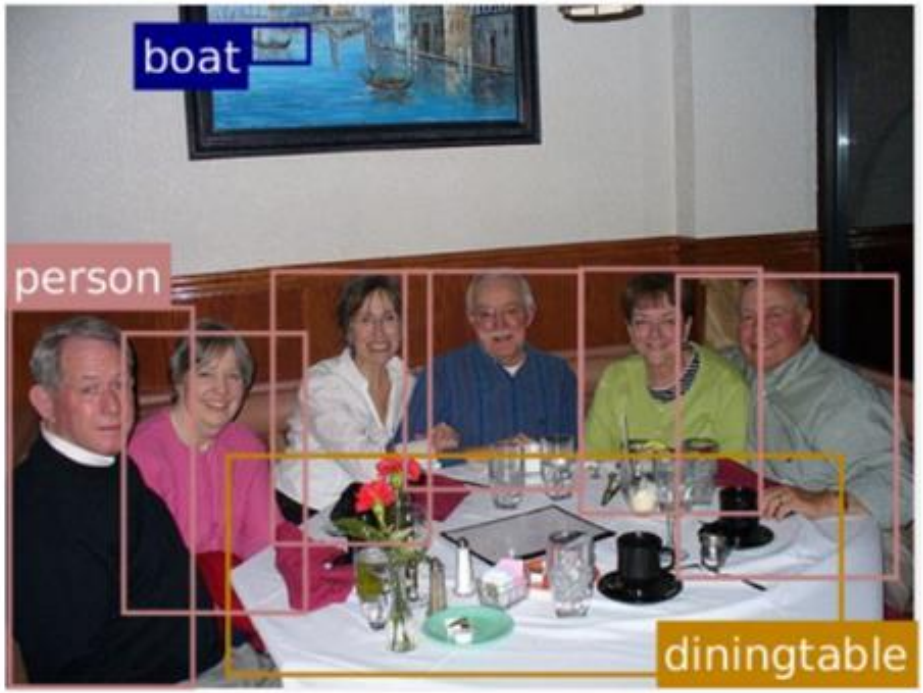
Algorithmic
progress

DL – Applications

Semantic image segmentation

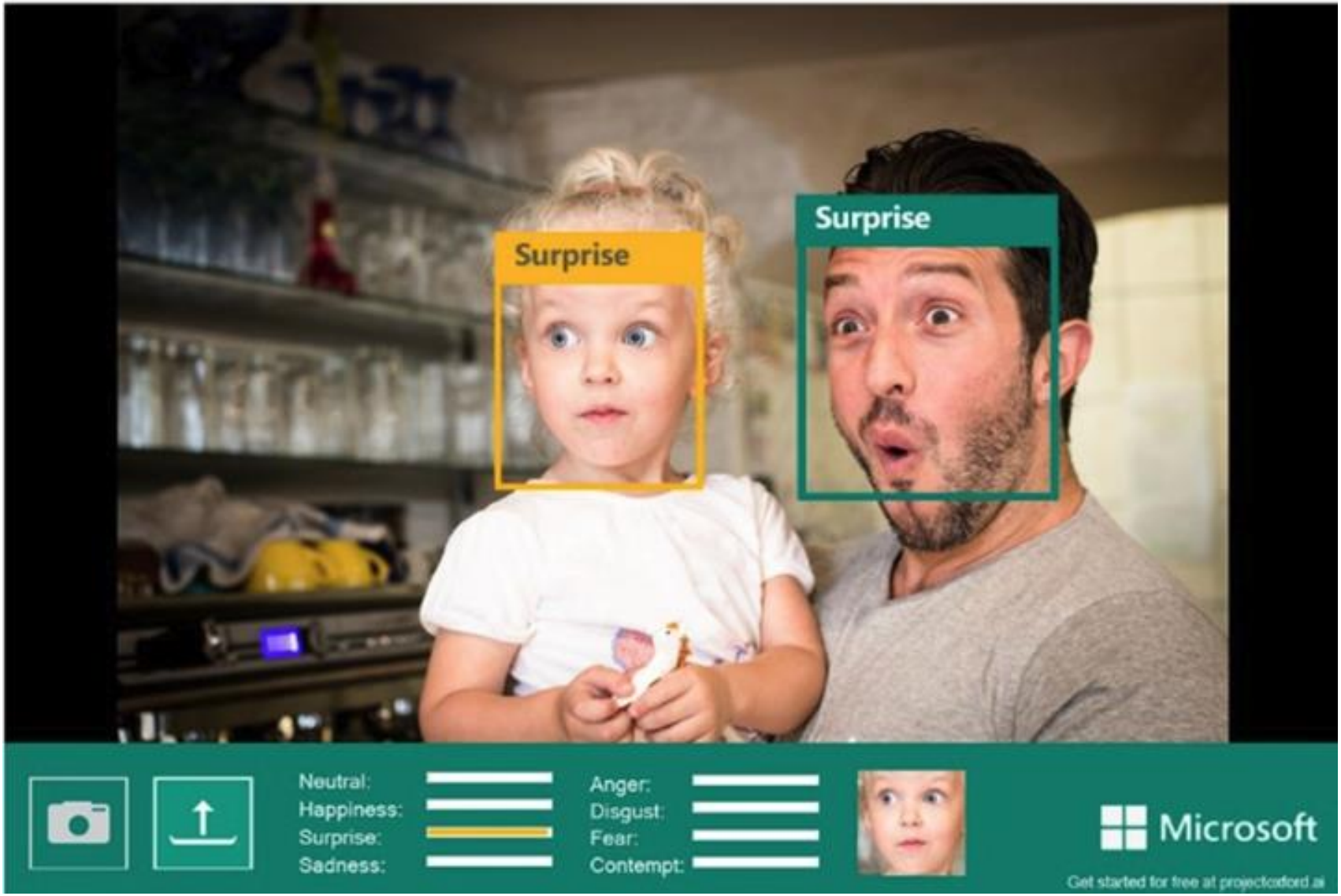


Object classification



DL – Applications

Face Detection + Emotion Classification (Microsoft Research)



DL – Applications

Food Recognition (Calorie Mama – Azumio, NutriNet)

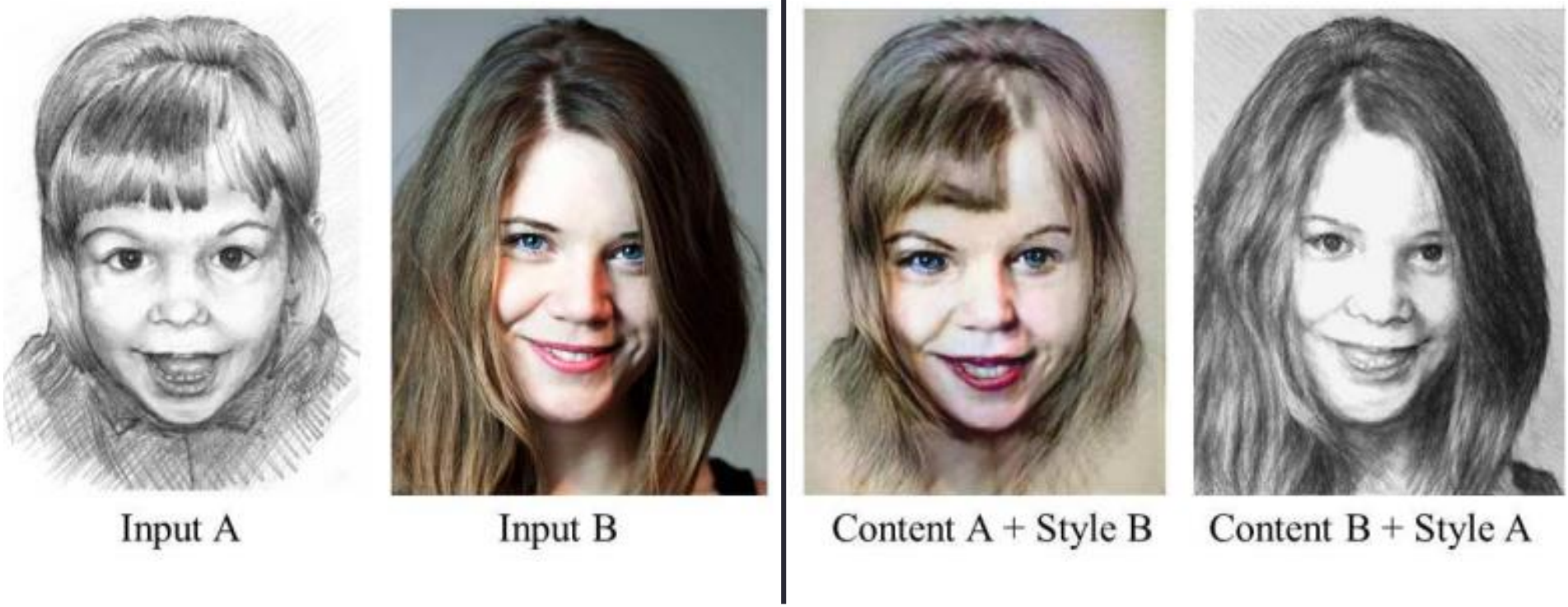
Segmentation:

Portion & Nutrition Estimation:

Sandwich: 146g
French Fries: 85g
Strawberries: 114g

DL – Applications

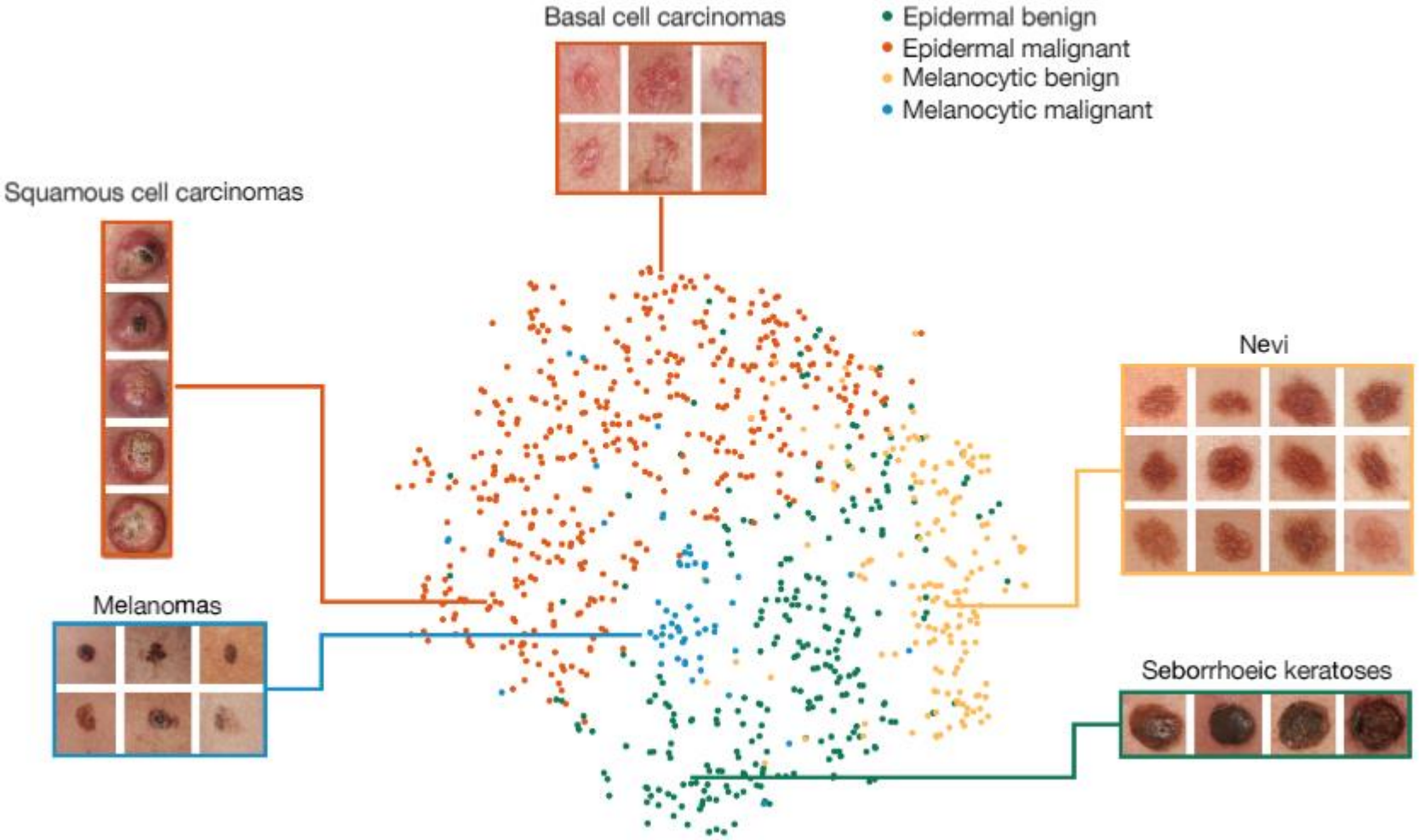
Photo-Realistic Synthesis (CVPR 2016)
(MRF and CNN)



DL – Applications

Skin Cancer Diagnosis (Nature 2017)

The **CNN** performed just as well as **two dozen dermatologists** in deciding whether lesion needed further medical attention.



DL – Applications

Show-and-Tell: Neural Image Caption Generator (CVPR 2015)

DL model: A group of young people playing a game of Frisbee.

DL – Applications

Visual Question Answering (CVPR 2017)

Who is wearing glasses?

man

woman

Where is the child sitting?

fridge

arms

Is the umbrella upside down?

yes

no

How many children are in the bed?

2

1

DL – Applications

Real-time Image Text Translation (Google, 2015)

MNIST Dataset

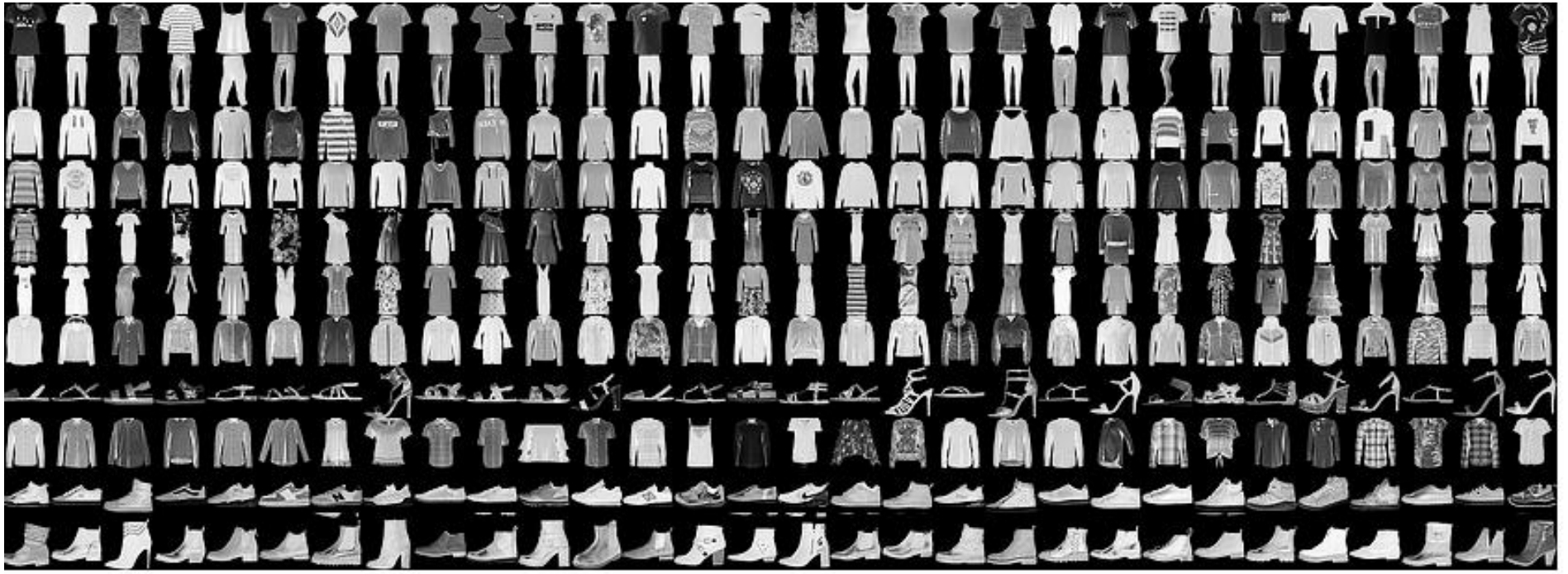
60,000 training images and 10,000 test images; 10 classes

Digits 0 to 9

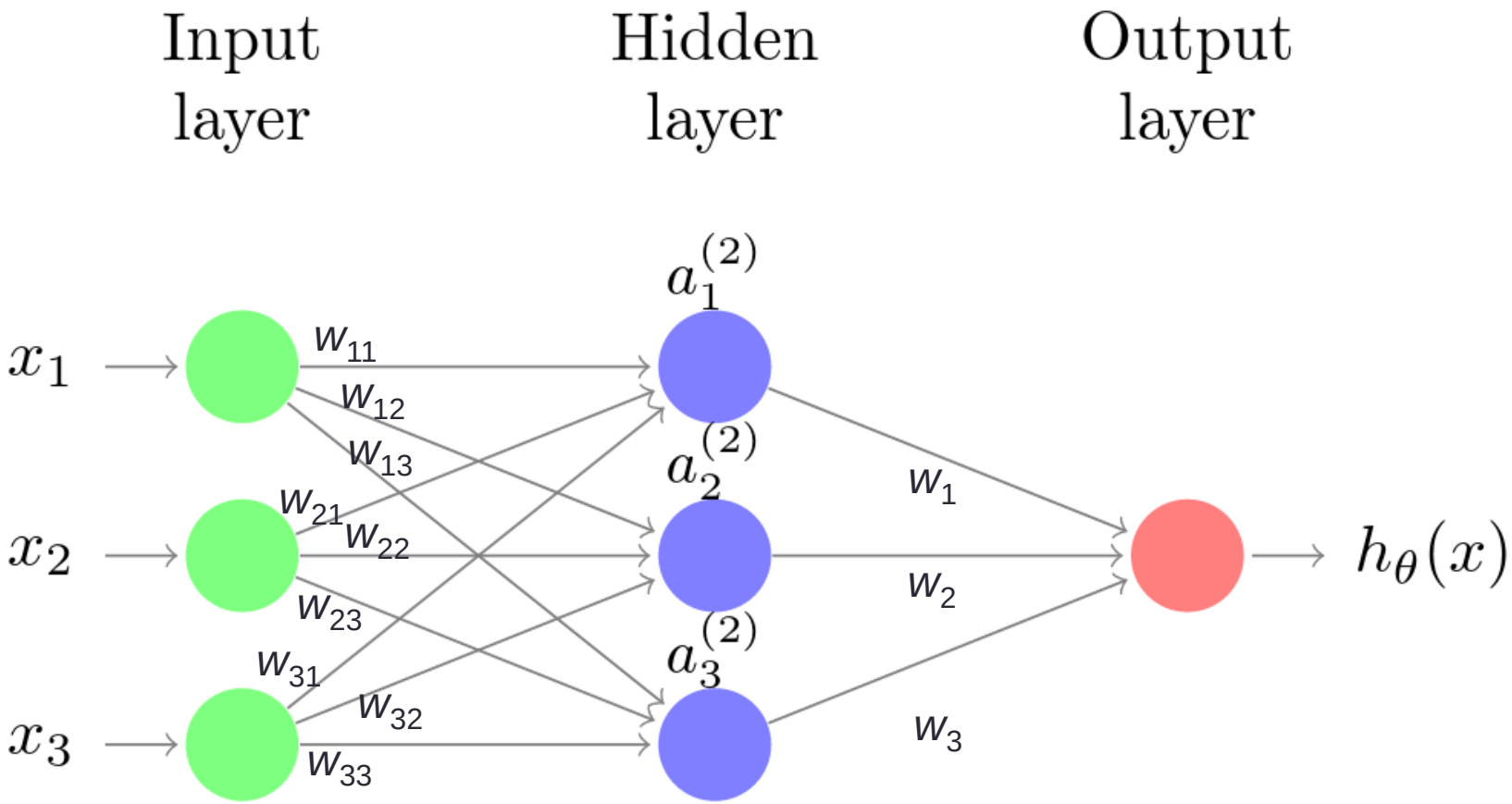
Fashion MNIST Dataset

60,000 training images and 10,000 test images; 10 classes

Clothes and Accessories: shirts, bags, shoes, etc.



Artificial Neural Network (ANN)



$$a_j^{(2)} = \sigma\left(\sum_i x_i w_{ij} + b_j\right) \qquad h_\theta = \sigma\left(\sum_j a_j^{(2)} w_j + b\right)$$

Artificial Neural Network (ANN)

$$L = \frac{1}{2}\sum_i \left(h_\theta(x_i) - y_i\right)^2$$

$$a_j^{(2)} = \sigma\left(\sum_i x_i w_{ij} + b_j\right)$$

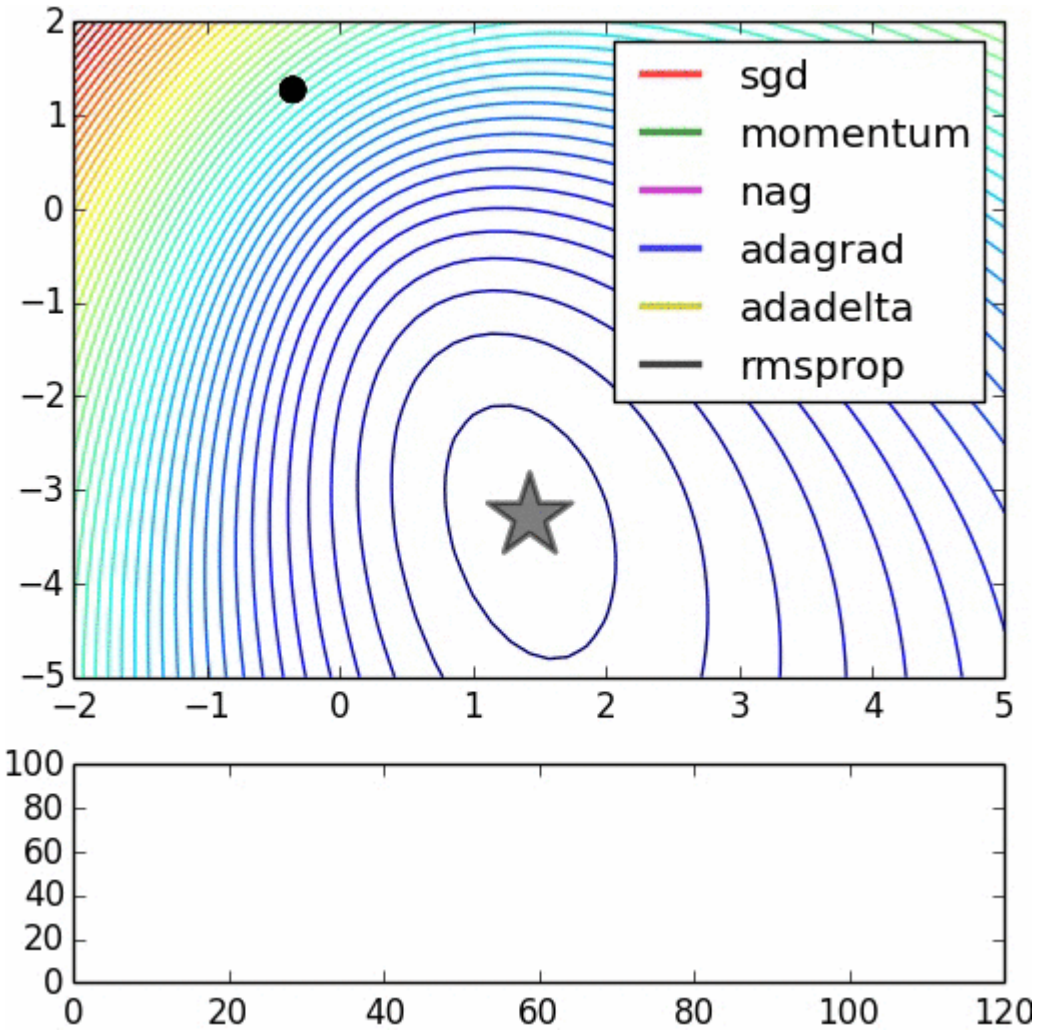
$$\frac{\partial L}{\partial \theta} = 0$$

$$h_\theta = \sigma\left(\sum_j a_j^{(2)} w_j + b\right)$$

$$\frac{\partial L}{\partial w_j} = \frac{\partial L}{\partial h_\theta}\frac{\partial h_\theta}{\partial w_j} \qquad \frac{\partial L}{\partial w_{ij}} = \frac{\partial L}{\partial h_\theta}\frac{\partial h_\theta}{\partial a_j^{(2)}}\frac{\partial a_j^{(2)}}{\partial w_{ij}}$$

$$\theta^{(n+1)} = \theta^{(n)} - \eta\frac{\partial L}{\partial \theta^{(n)}}$$

Model Optimization

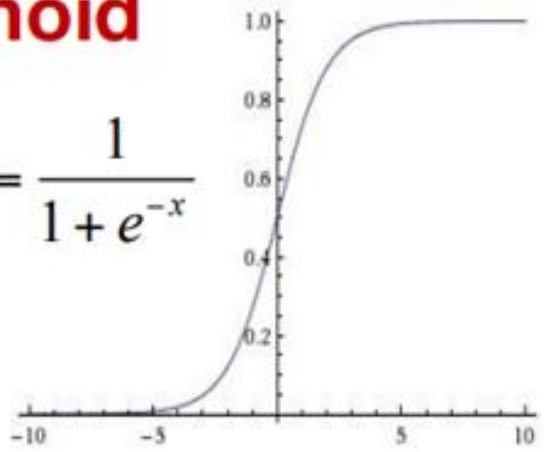


Gradient Descent

Activation Functions

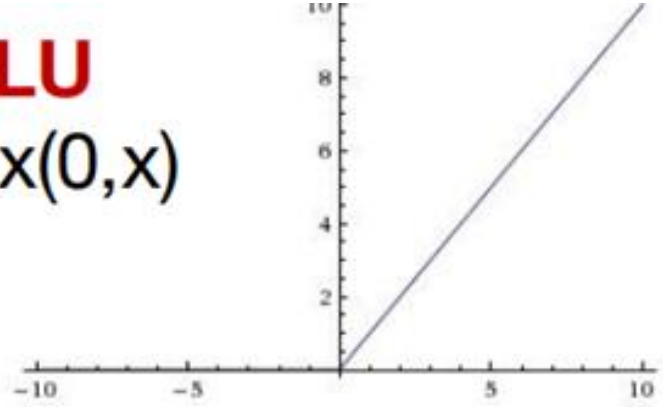
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



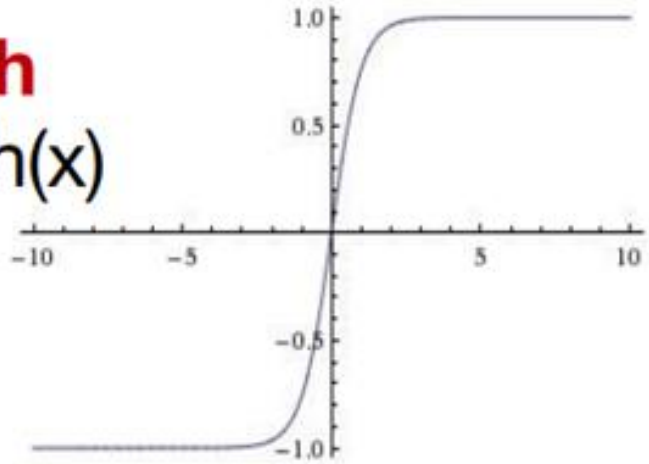
ReLU

max(0,x)



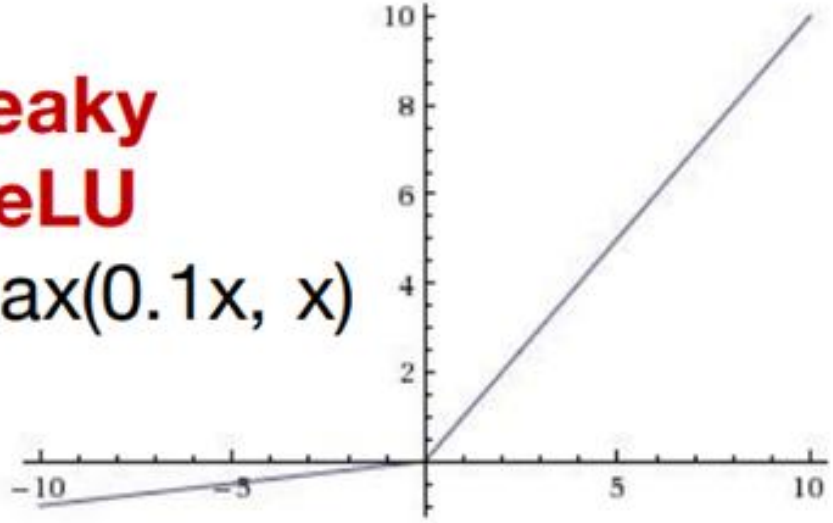
tanh

tanh(x)



Leaky ReLU

max(0.1x, x)



Softmax activation function

(Normalized exponential function)

$$\sigma(x_i) = \frac{\exp(x_i)}{\sum_k \exp(x_k)}$$

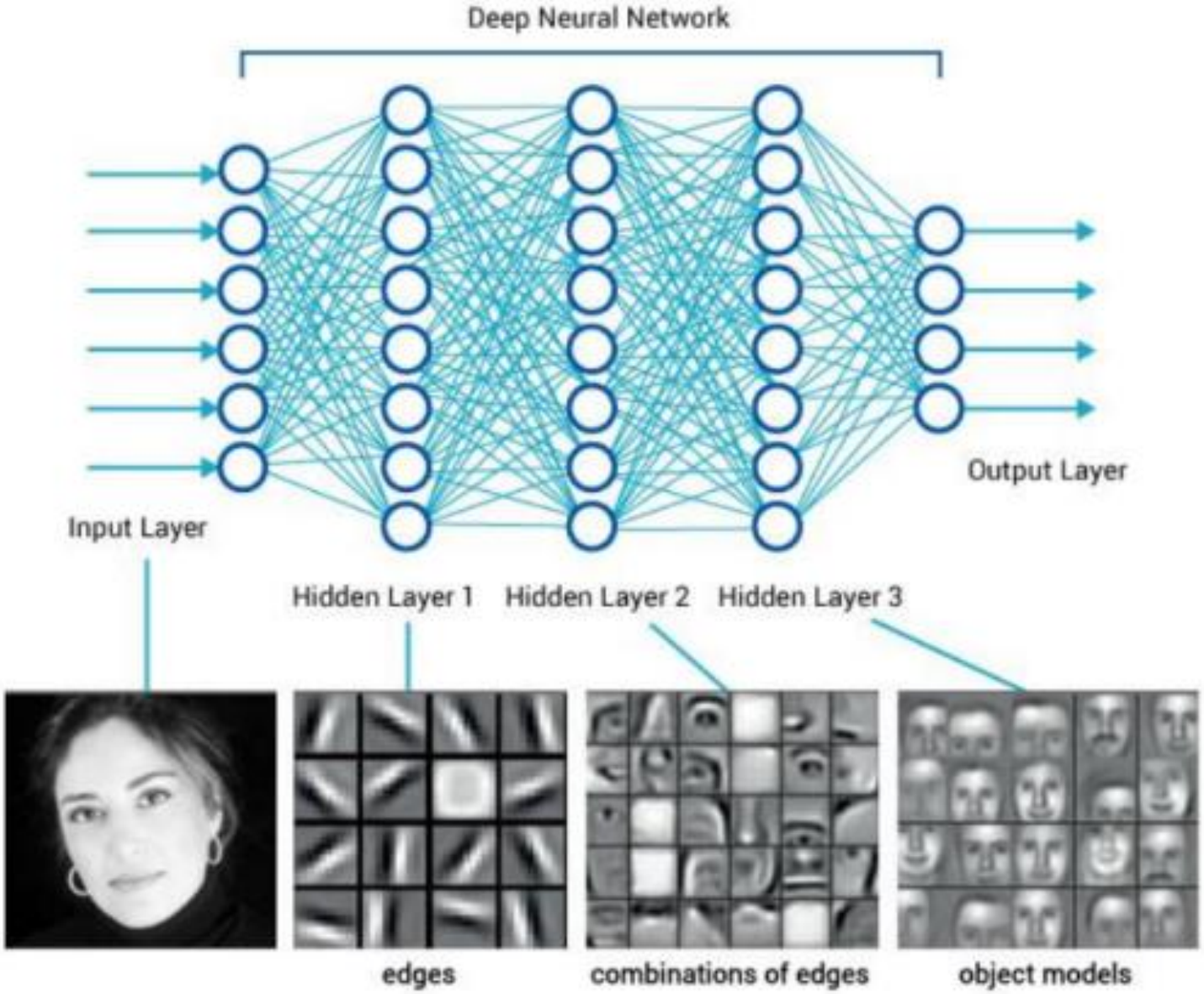
Consider input = **[1, 2, 3, 4, 1, 2, 3]**,

the corresponding softmax 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.

Deep Neural Network (DNN)



TensorFlow API

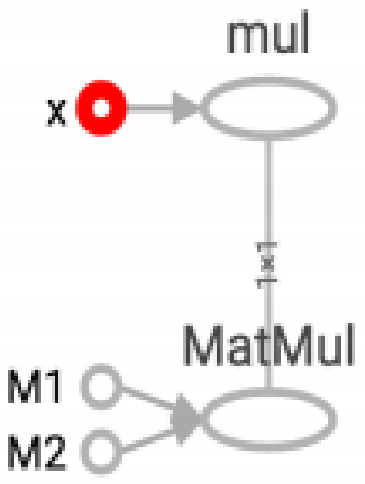
```
import tensorflow as tf
sess = tf.Session()
hello = tf.constant('Hello, TensorFlow')
sess.run(hello)
```


```
import tensorflow as tf
# We construct a graph (we write to the default graph)
m1 = tf.constant([[3., 3.]], name='M1')
m2 = tf.constant([[2.],[2.]], name='M2')
product = 10*tf.matmul(m1,m2)
```

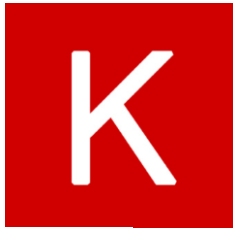
$$10\begin{pmatrix} 3 & 3 \end{pmatrix}\begin{pmatrix} 2 \\ 2 \end{pmatrix} = 120$$

```
sess = tf.Session()
res = sess.run(product)
print(res)
sess.close()
```

```
[[ 120.]]
```



Keras API



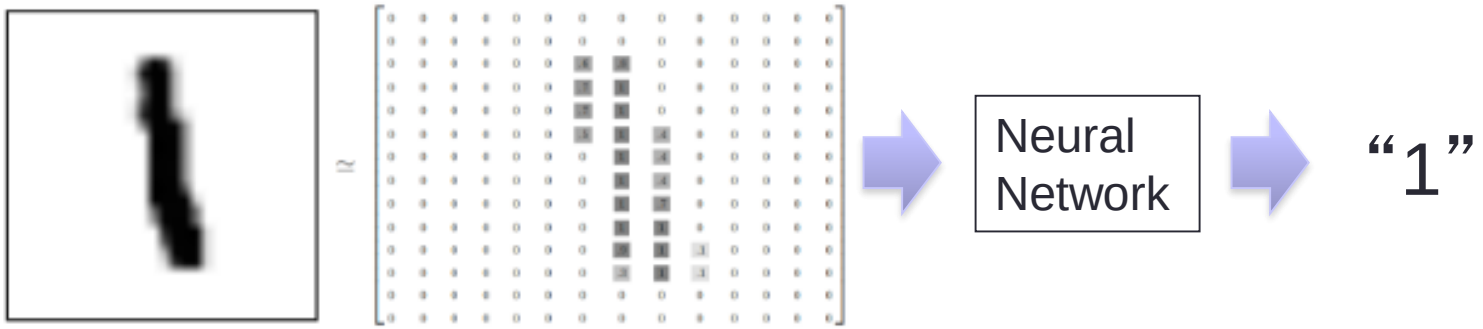
```
import keras
from keras import layers

model = keras.Sequential()
model.add(layers.Dense(20, activation='relu', input_shape=(10,)))
model.add(layers.Dense(20, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

model.fit(x, y, epochs=10, batch_size=32)
```

Fully connected NN

0–9 handwritten digit recognition:

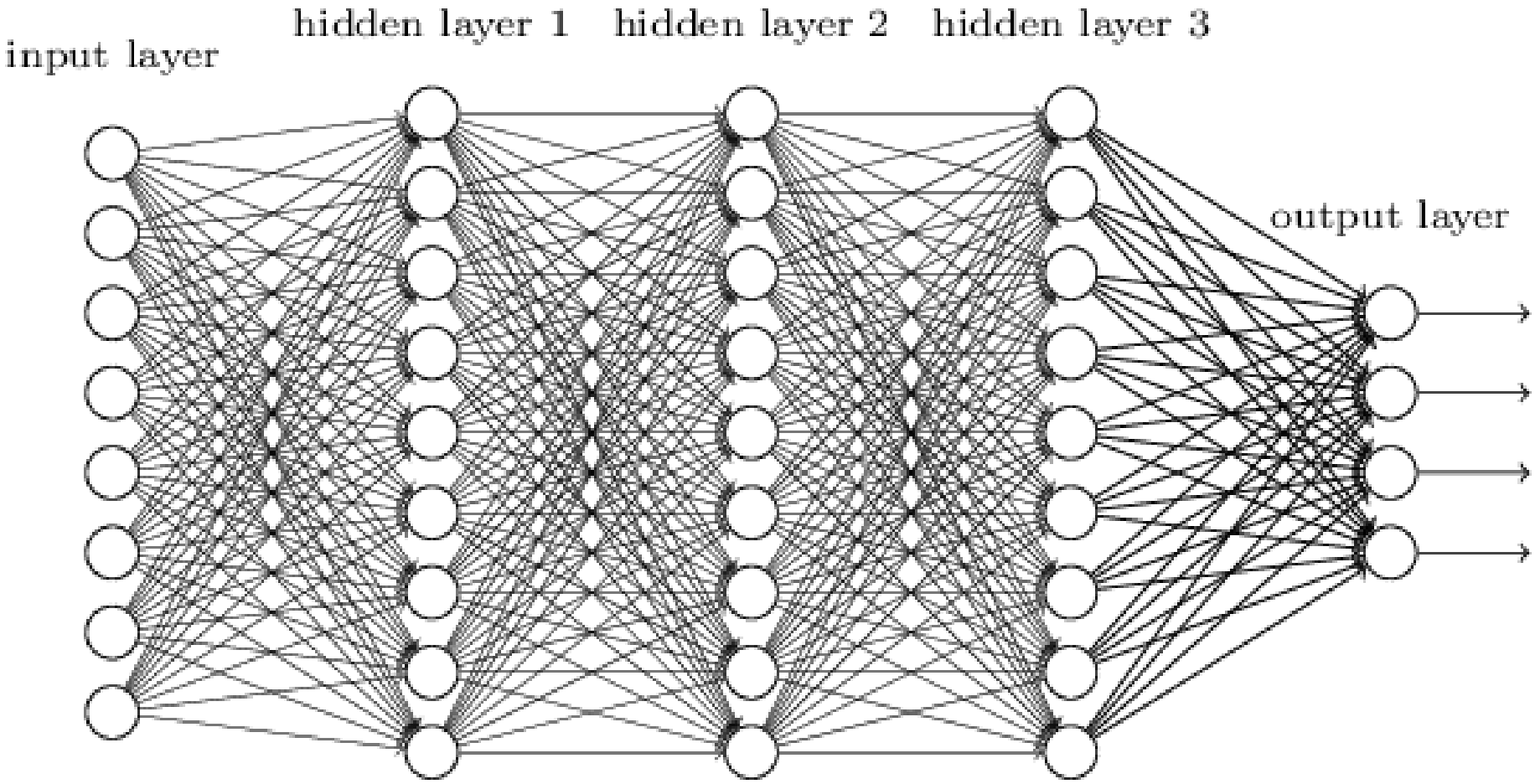


28 x 28

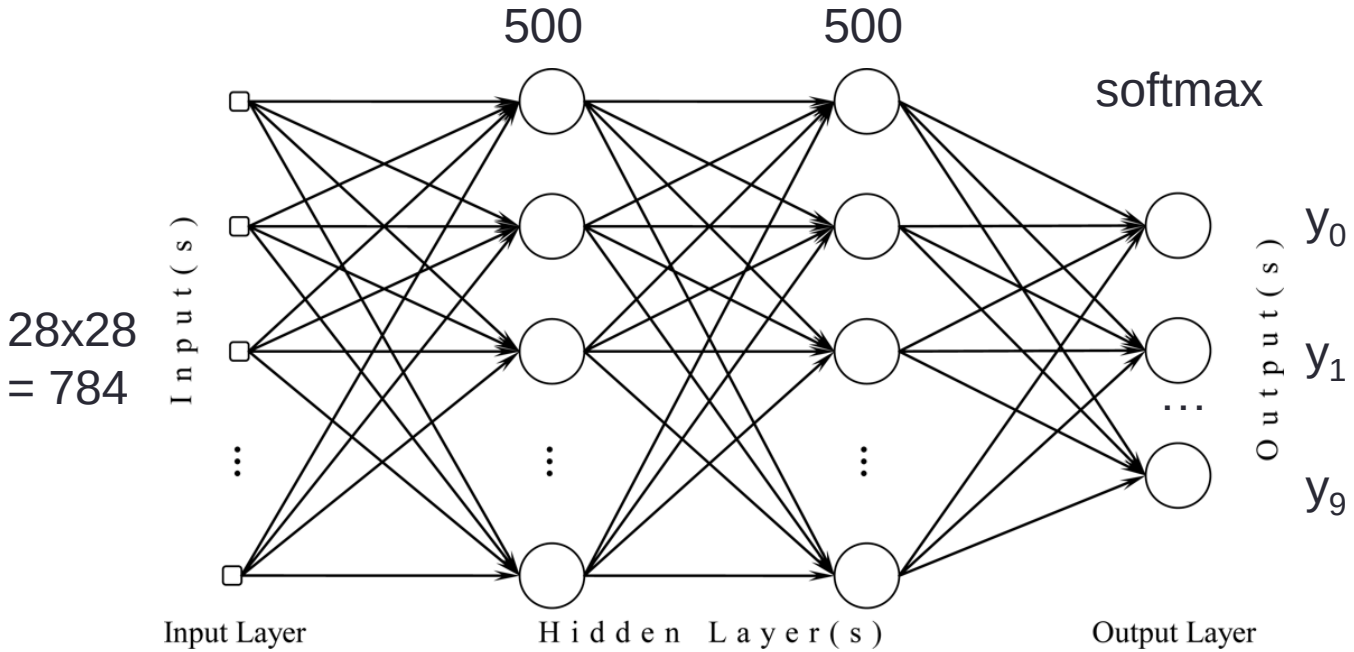
MNIST Data maintained by Yann LeCun: http://yann.lecun.com/exdb/mnist/

Keras provides data sets loading function at http://keras.io/datasets

Fully connected NN (FCNN)



FCNN in Keras

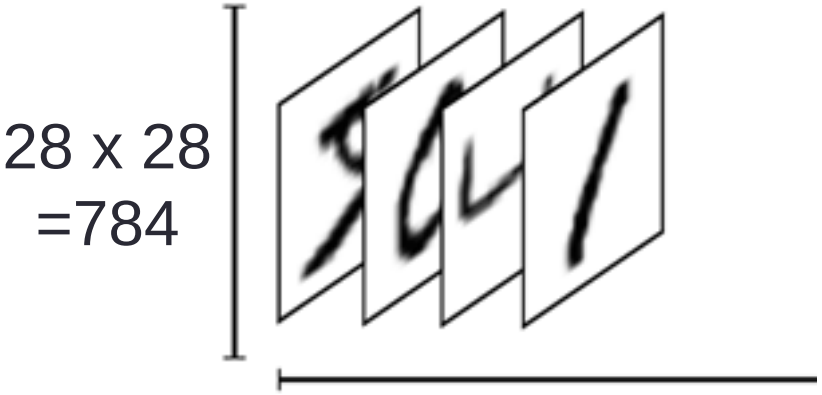


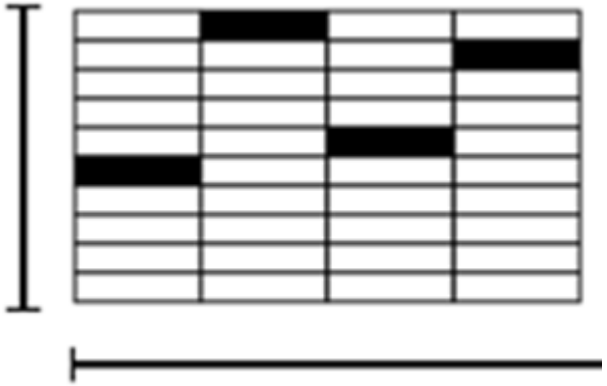
model = **Sequential**() # layers are sequentially added
model.add(**Dense**(input_dim=28*28, output_dim=500))
model.add(Activation('sigmoid')) #: softplus, softsign,relu,tanh, hard_sigmoid
model.add(**Dense**(output_dim = 500))
model.add (Activation('sigmoid'))
model.add(**Dense**(output_dim=10))
model.add(Activation('softmax'))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

FCNN – Training

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

numpy array

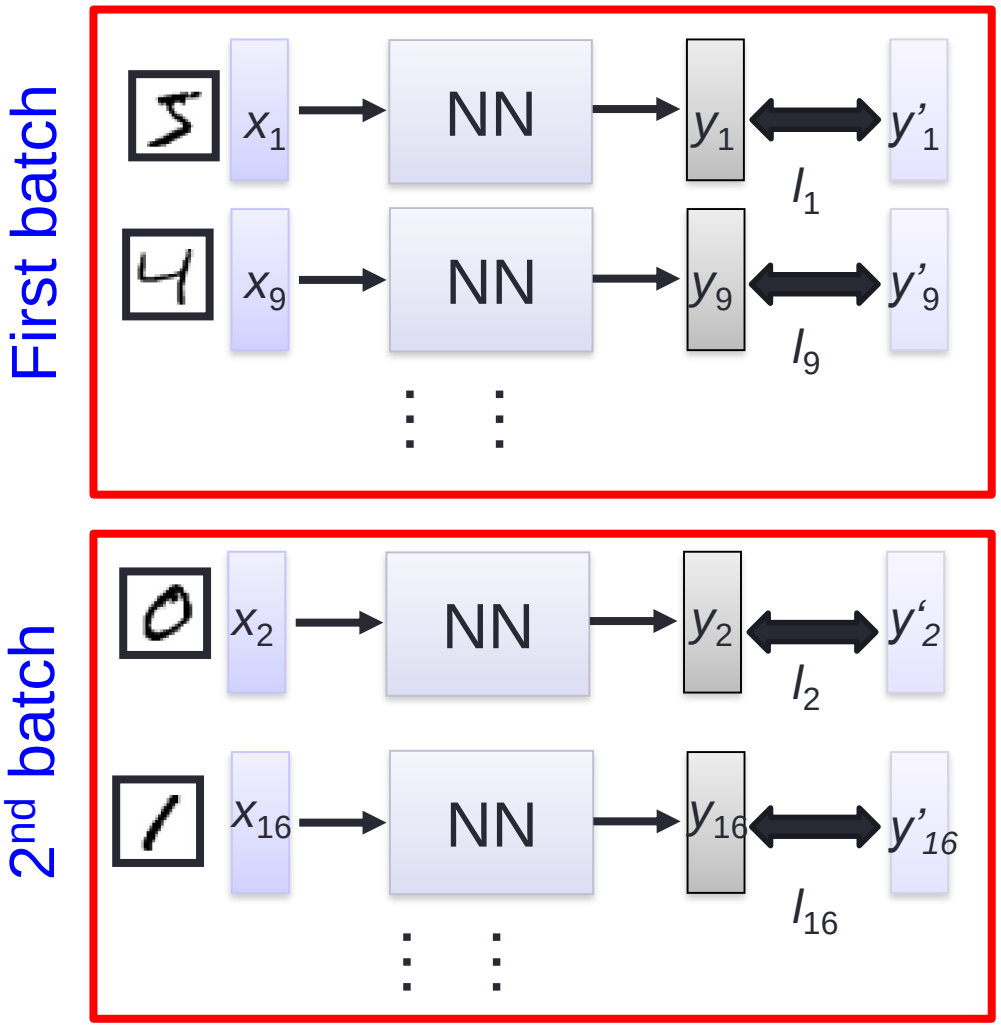




Batch Training

We do not really minimize total loss!

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



- 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

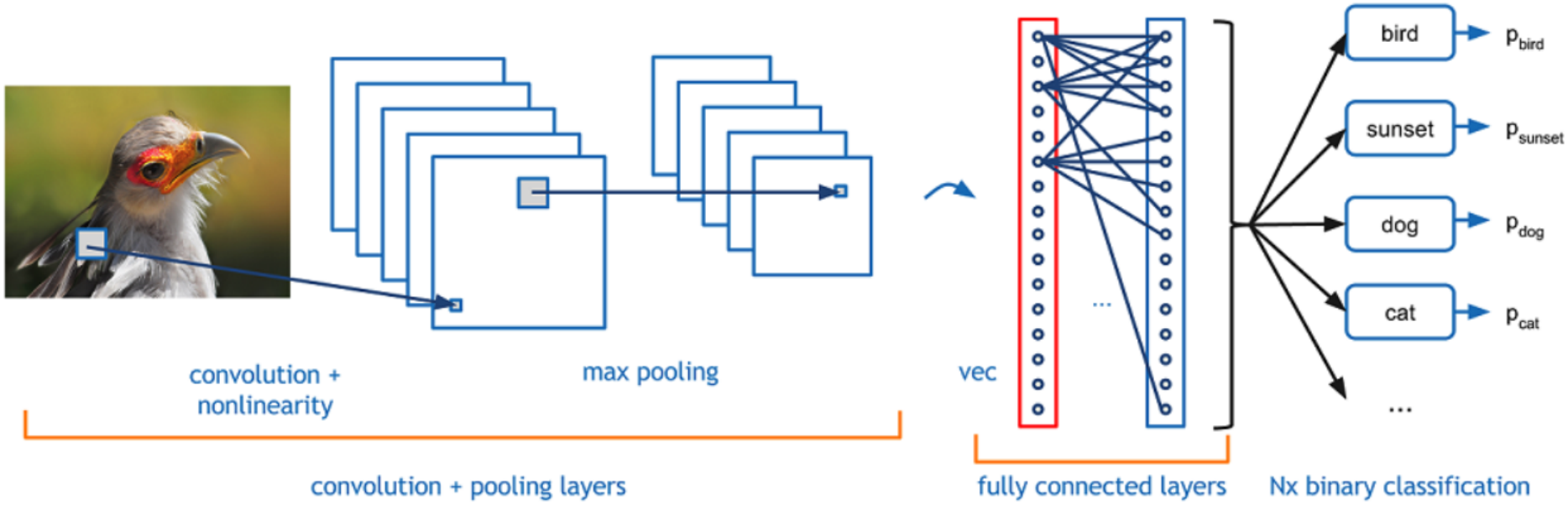
one epoch

Repeat the above process

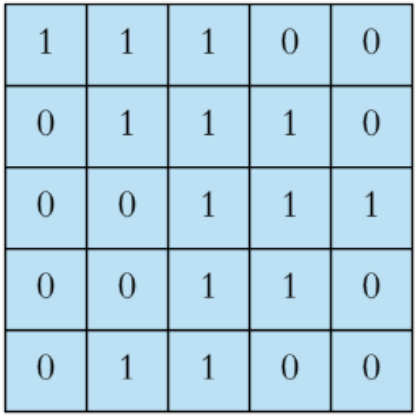
Hands-on
MLP & Deep MLP

1) drive.google.com (login)
2) download notebook from **http://bit.ly/DL_SFIT**
3) copy notebook to your Google Drive
4) Open notebook with Google Colab

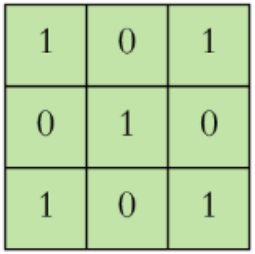
Convolution Neural Network (CNN)



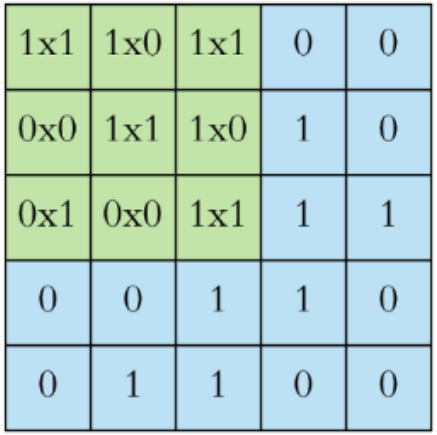
Convolutional Layer



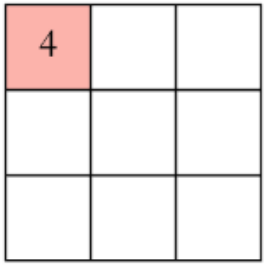
Input



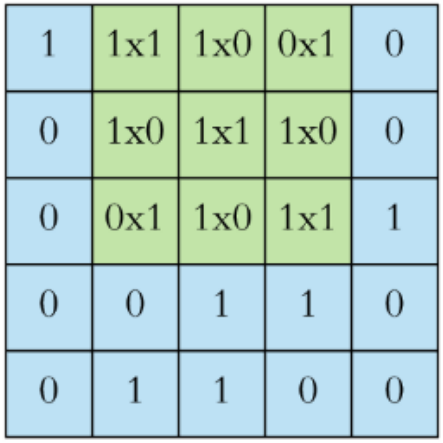
Filter / Kernel



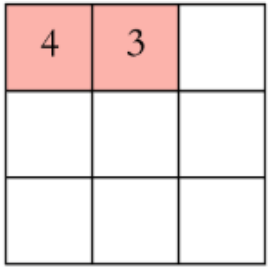
Input x Filter



Feature Map



Input x Filter



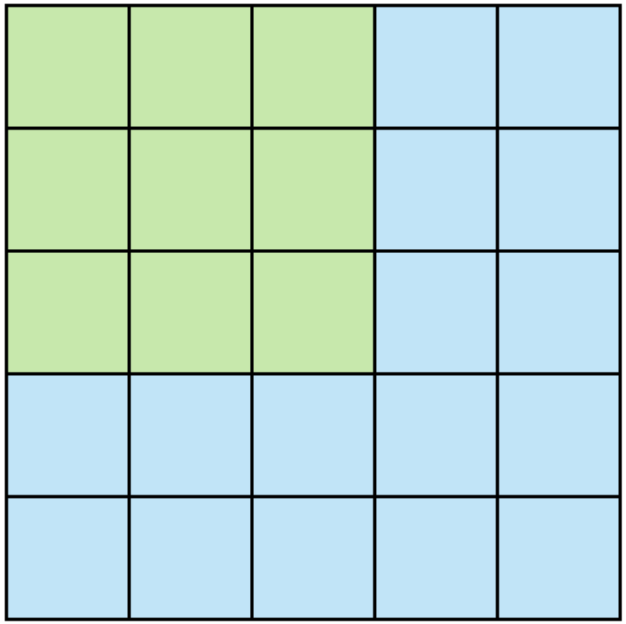
Feature Map

Convolutional Layer

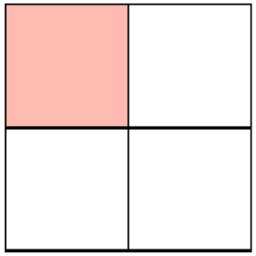
1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

4		

Convolutional Layer – Strides

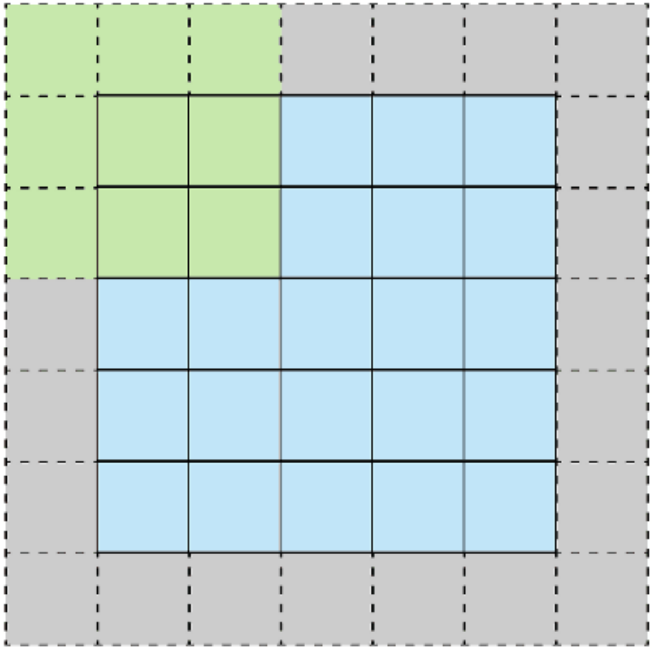


Stride 2

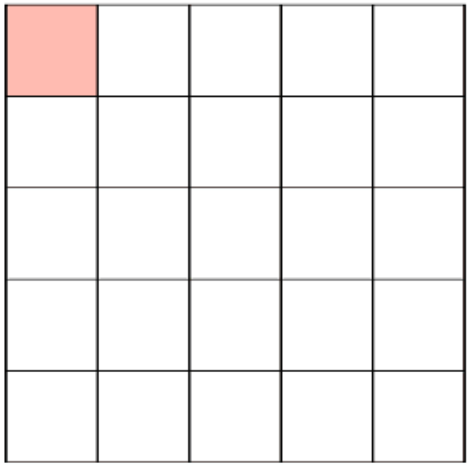


Feature Map

Convolutional Layer – Padding



Stride 1 with Padding



Feature Map

Parameters in Convolutional Layer

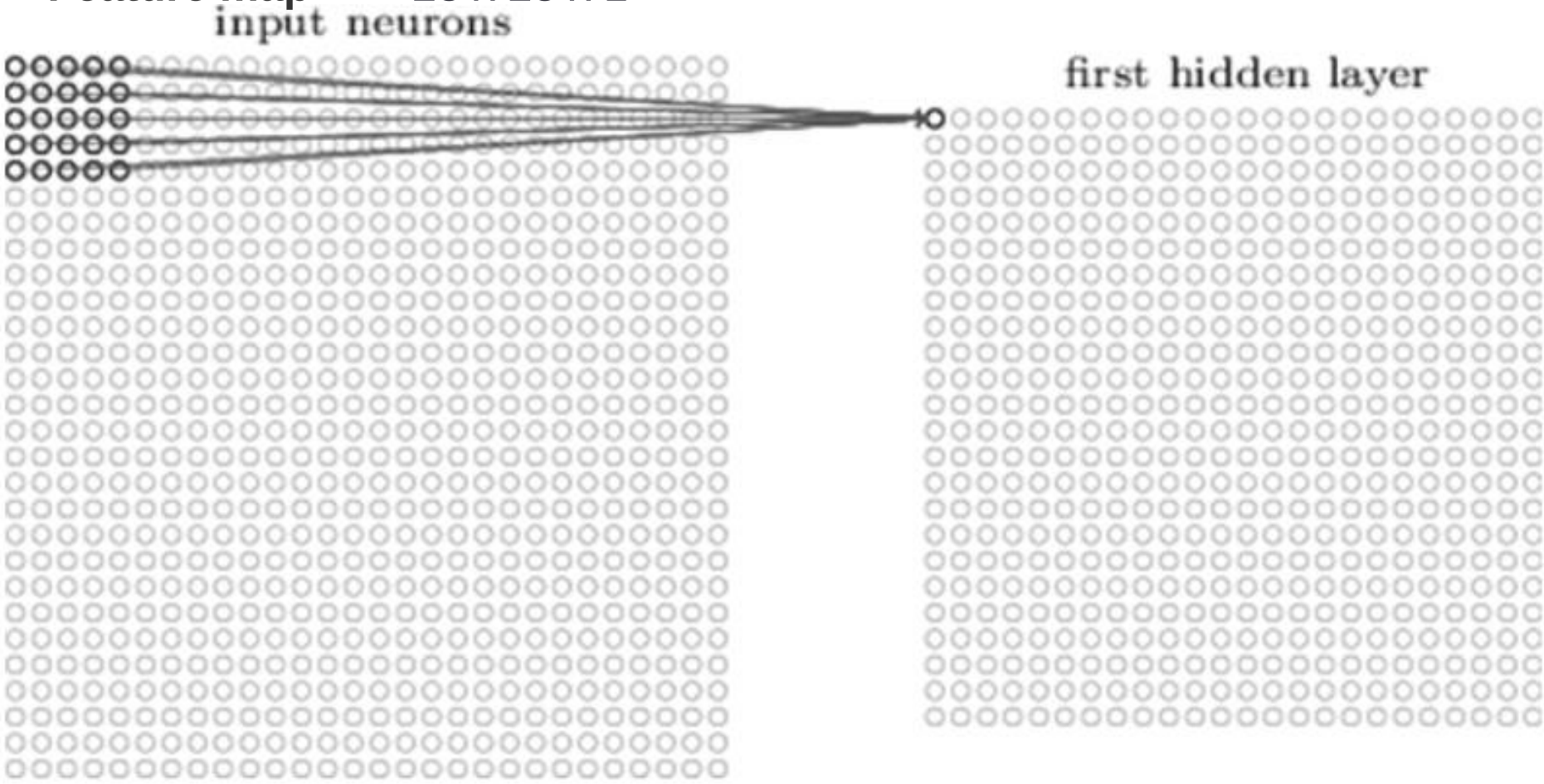
- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$

Convolutional Layer

Input ==> 32 x 32 x 1
Filter ==> 5 x 5 x 1

Feature map ==> 28 x 28 x 1

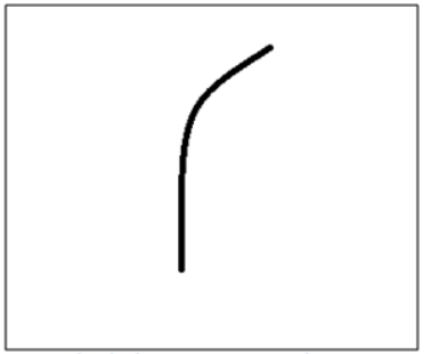
Consider stride of 1 and no padding



Convolutional Layer – Bird's eye view

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

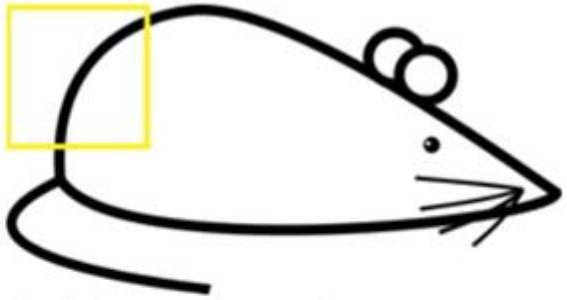


Visualization of a curve detector filter

$(50*30)+(50*30)+(50*30)+(20*30)+(50*30)$

$= 6600$ (A large number!)

Original image



Visualization of the filter on the image

Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

*

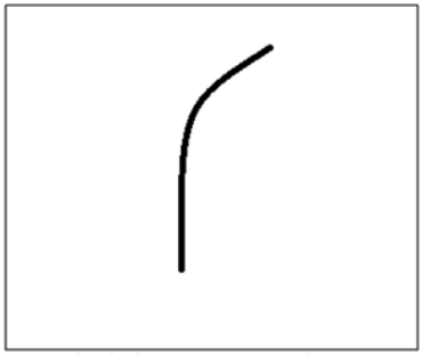
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Convolutional Layer – Bird's eye view

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter



Visualization of a curve detector filter

Visualization of the filter on the image

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

Pixel representation of receptive field

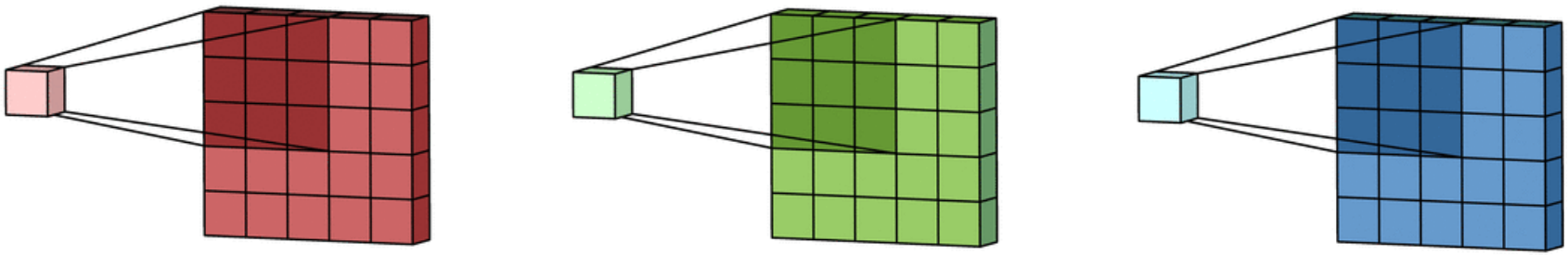
*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = 0

Convolutional layer – Color image



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 Kernels

$*\frac{1}{8}$

0	1	0
1	4	1
0	1	0

$*$

0	-1	0
-1	4	-1
0	-1	0

$*$

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

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

6 x 6 image

Convolution

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

Filter 1

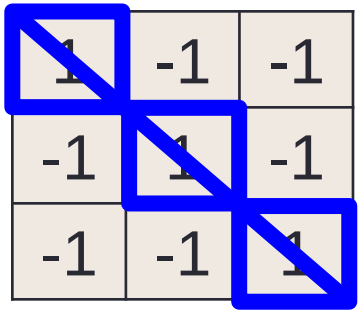
If stride=2

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

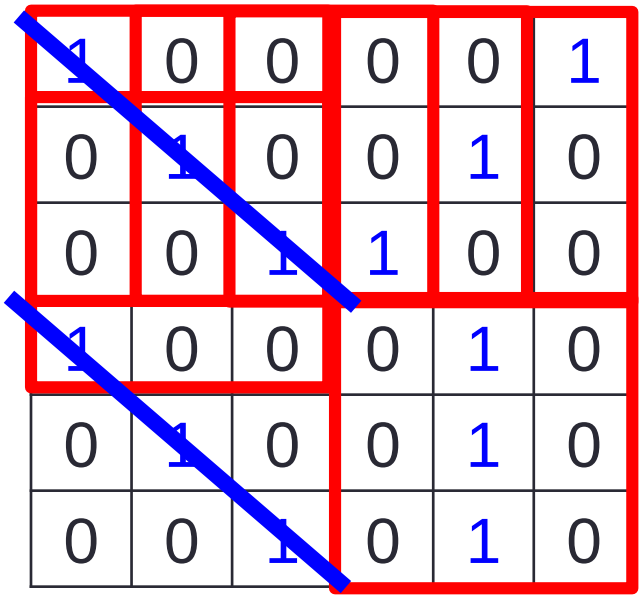
3 -3

Convolution

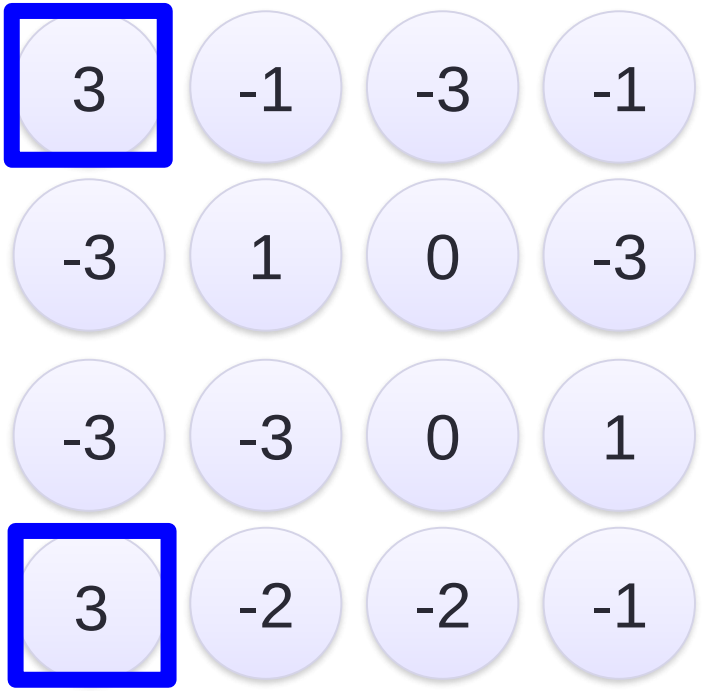


Filter 1

stride=1



6 x 6 image



Convolution

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

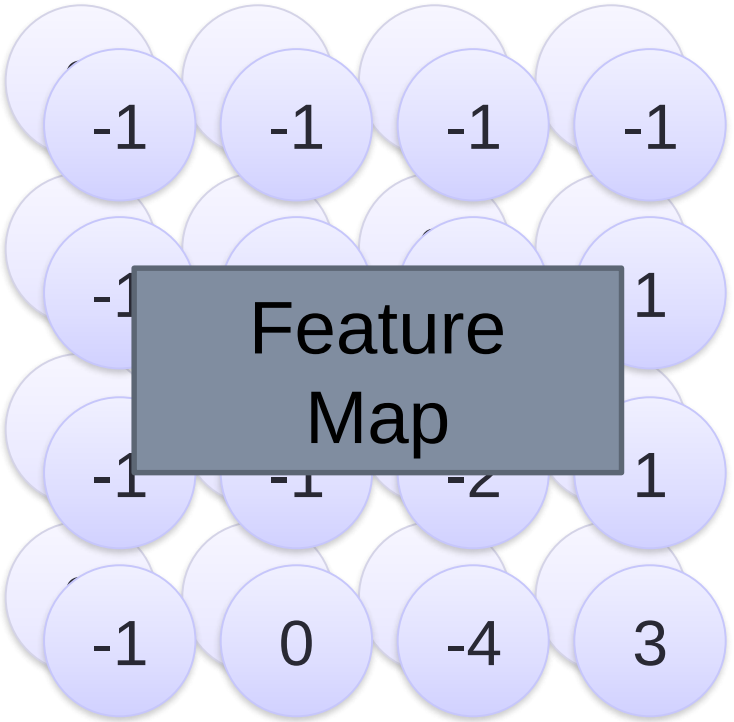
Filter 2

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 4 x 4 x 2 tensor

Color image: RGB 3 channels

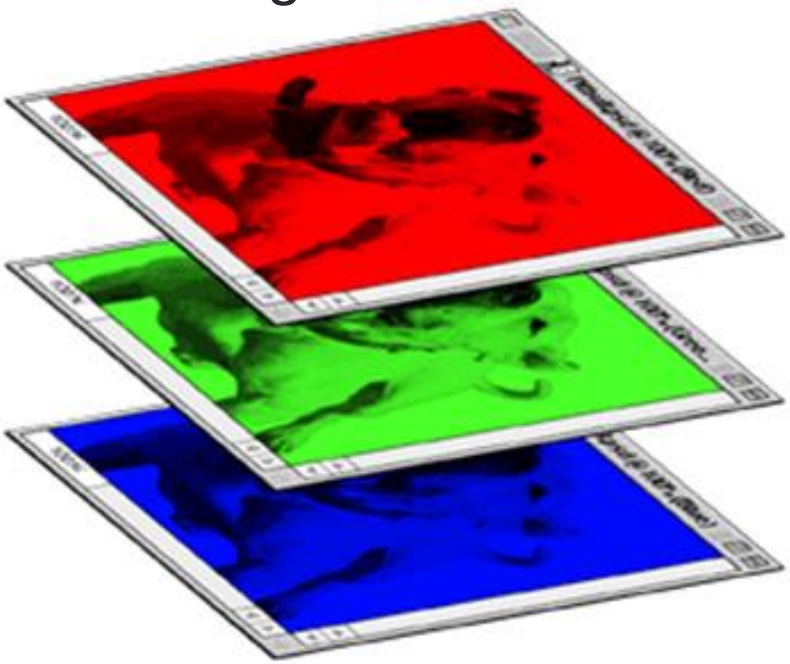
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

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

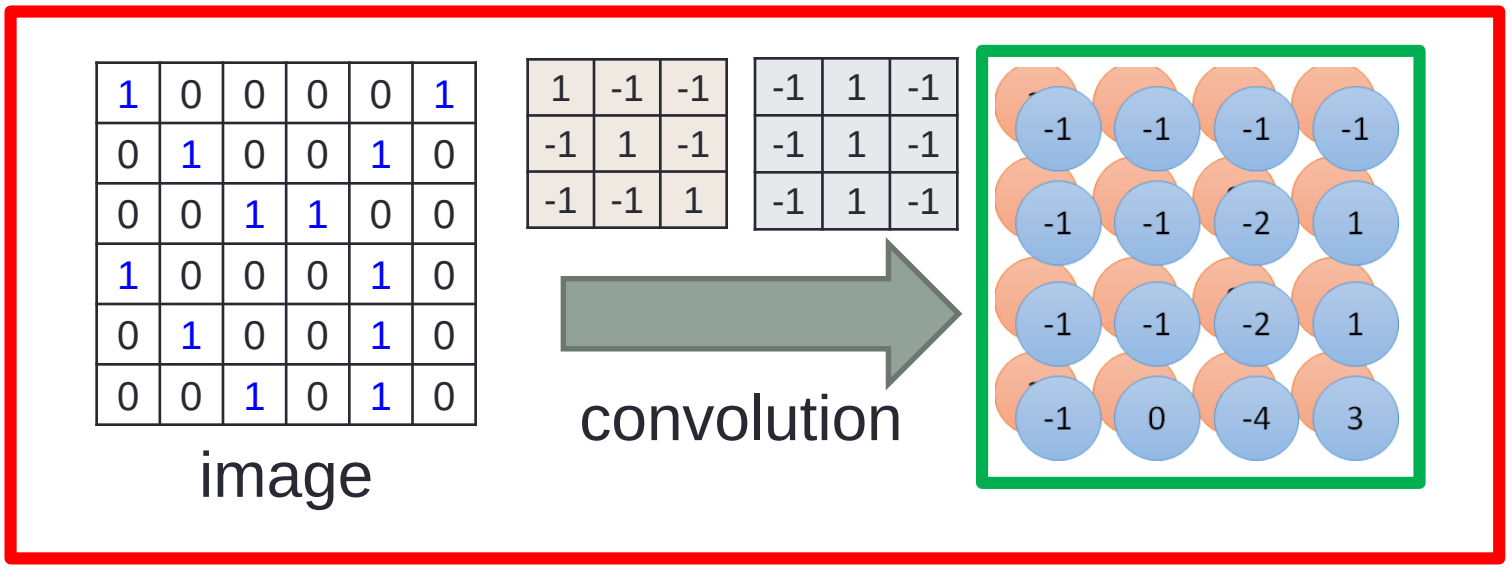
Filter 2

Color image

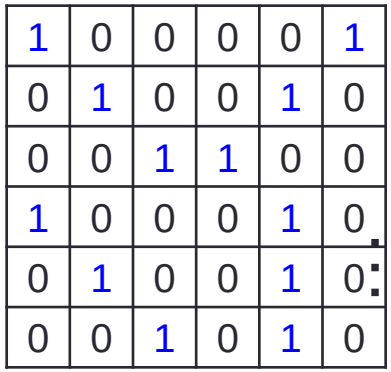


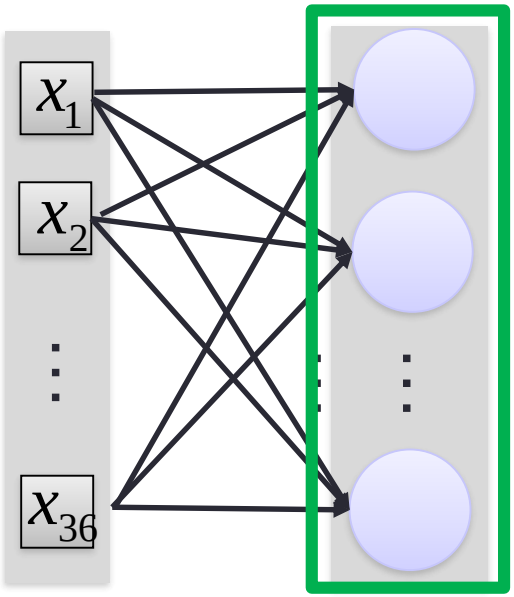
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

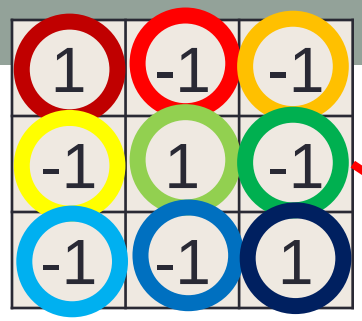
Convolution v/s Fully Connected



Fully-connected







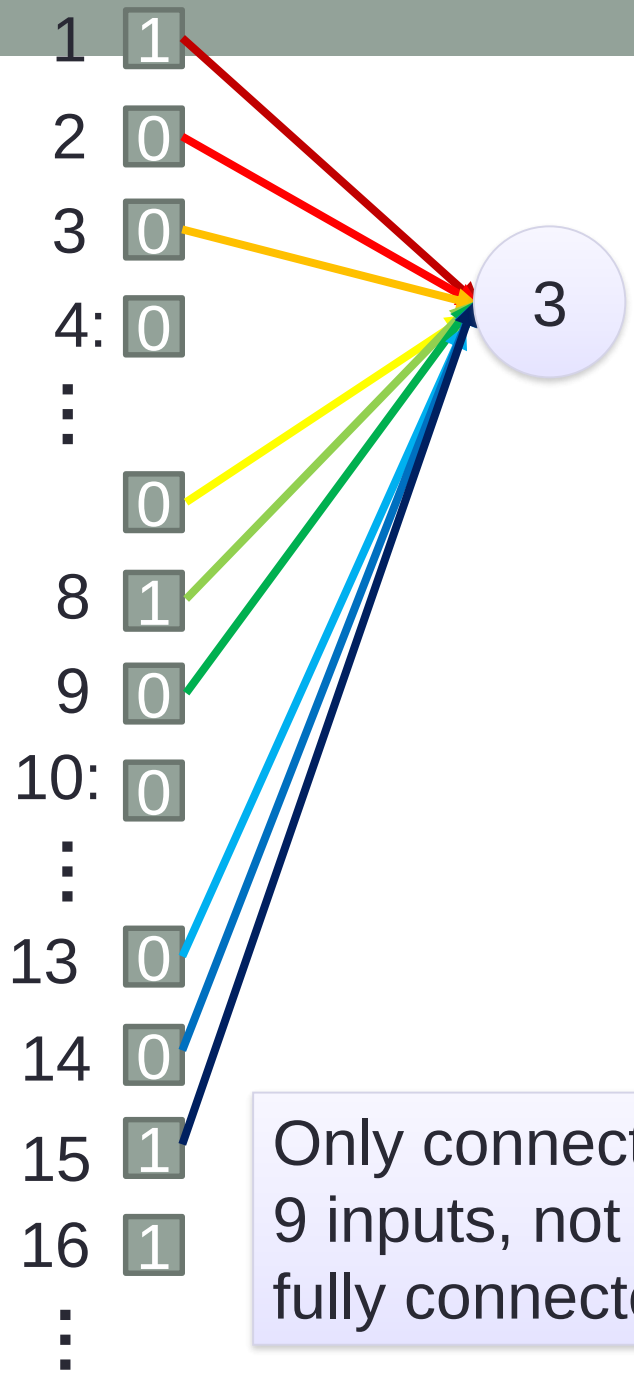
Filter 1

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

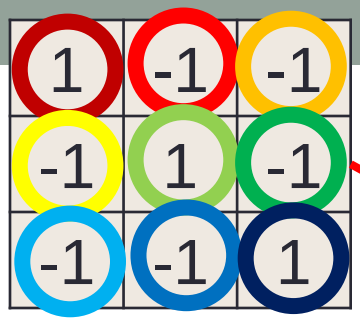
6 x 6 image

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

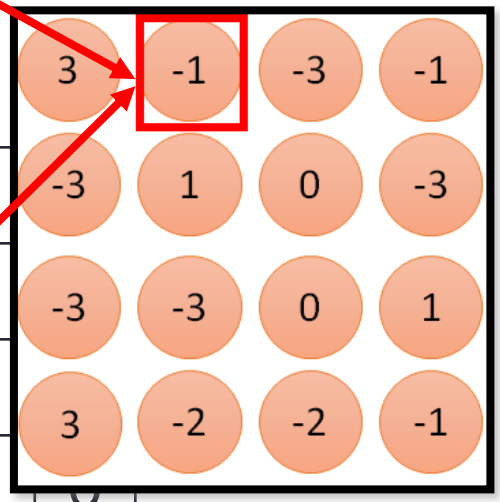
fewer parameters!



Only connect to 9 inputs, not fully connected



Filter 1

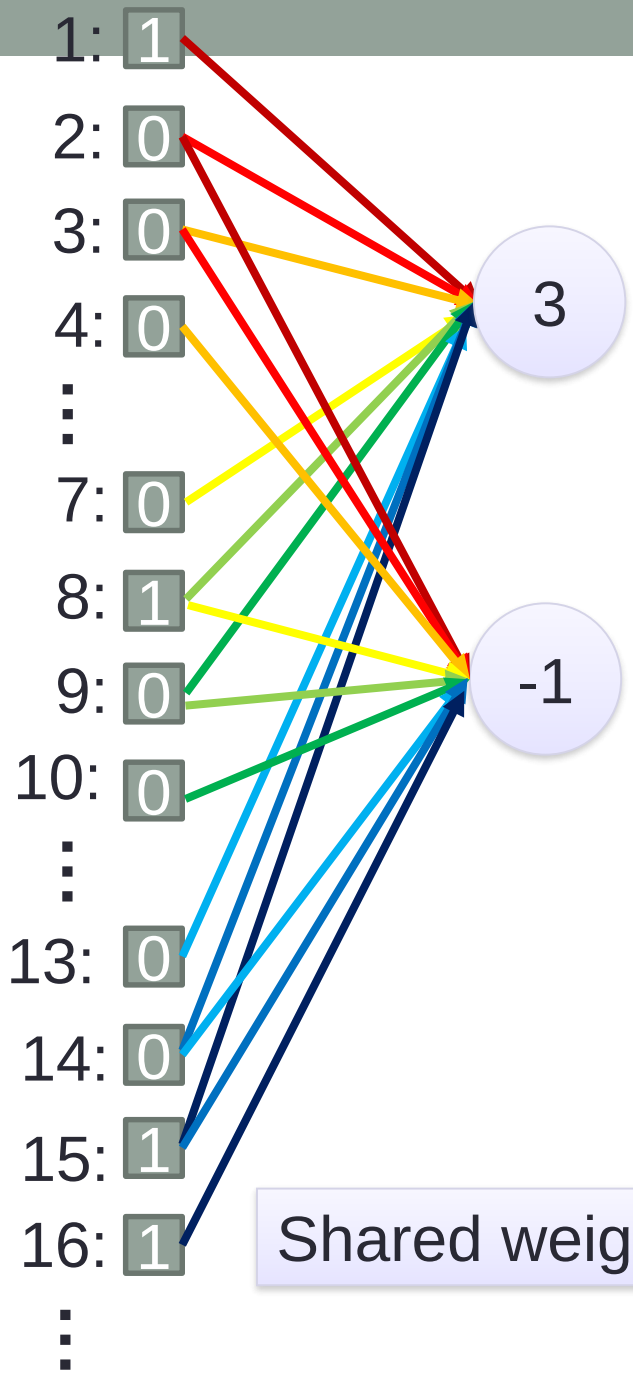


1	0	0	0	0	
0	1	0	0	1	
0	0	1	1	0	
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

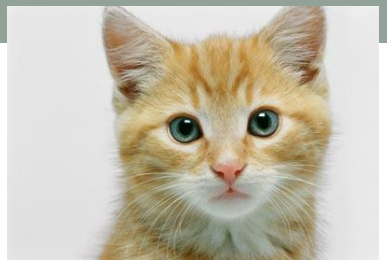
Fewer parameters

Even fewer parameters



Shared weights

The whole CNN



cat dog ……

Convolution

Max Pooling

Convolution

Max Pooling

Can repeat many times

Flattened

Fully Connected Feedforward network

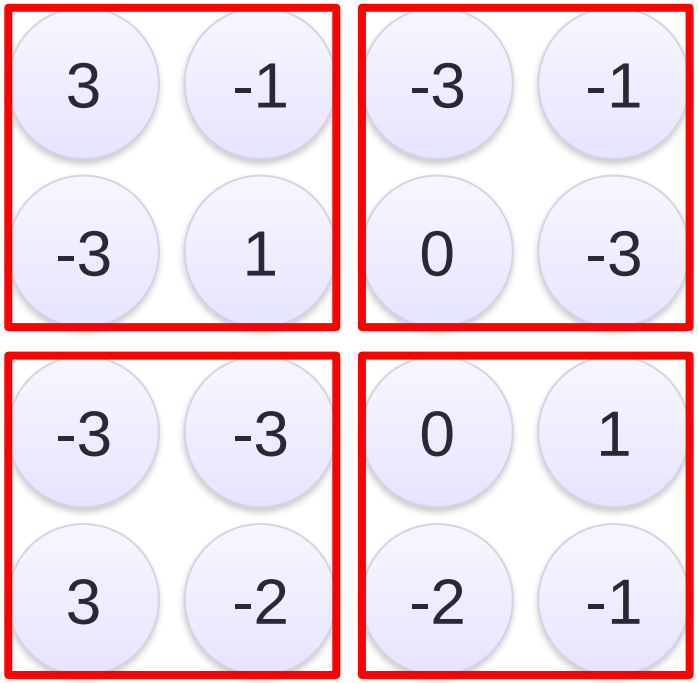
Max Pooling

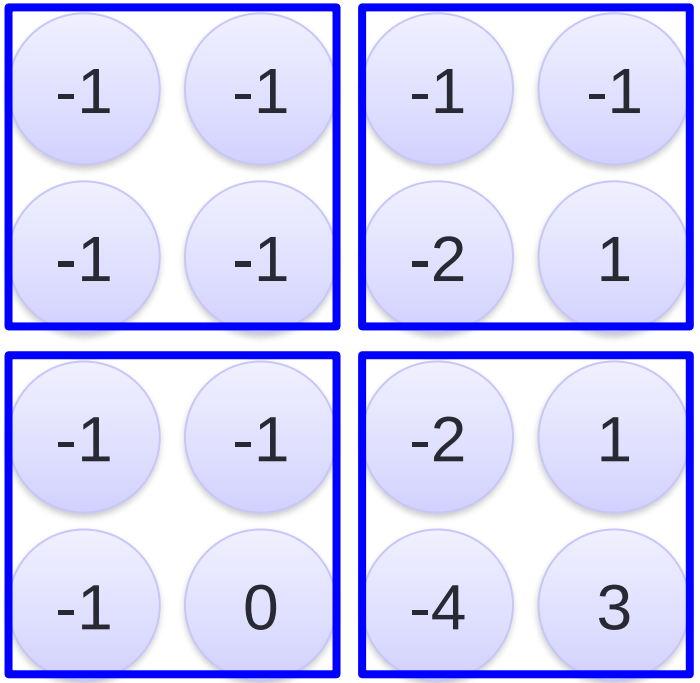
1	-1	-1
-1	1	-1
-1	-1	1

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

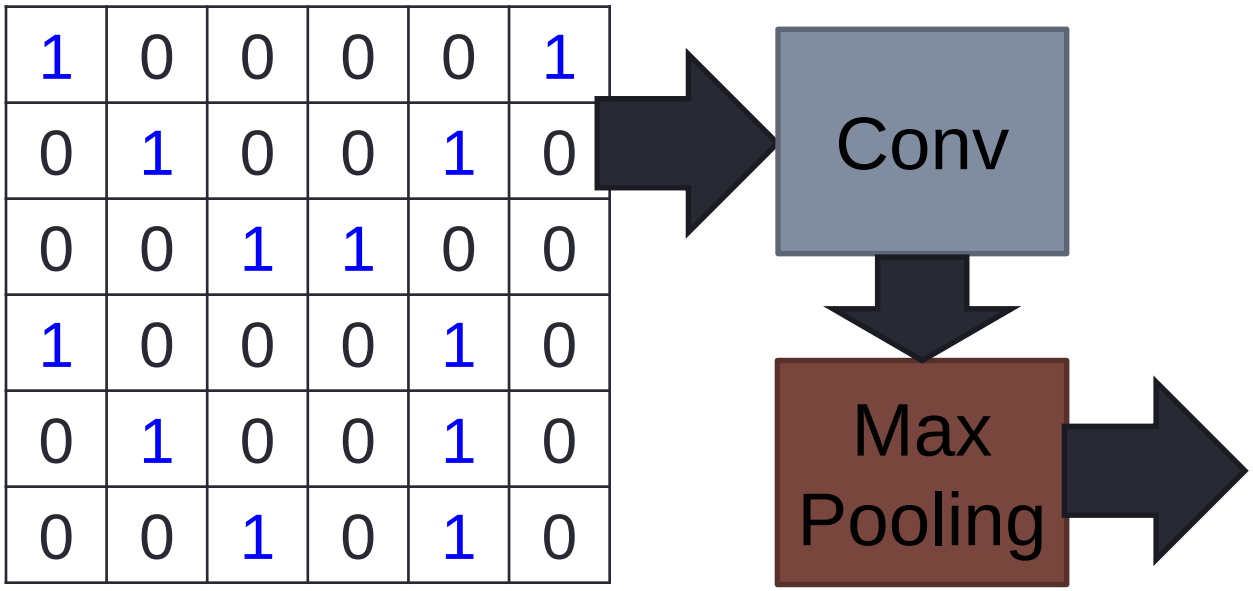
We can subsample the pixels to make image smaller

→ fewer parameters to characterize the image

A CNN compresses a fully connected network in two ways:

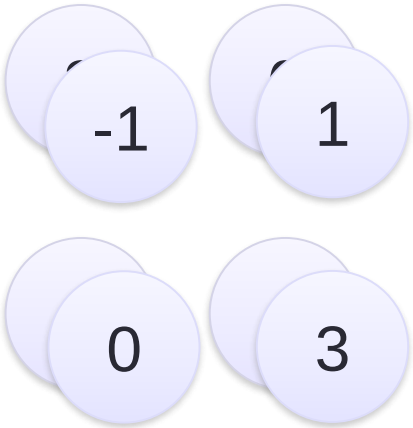
- **Reduced** number of **connections**
- **Shared weights** on the edges
- **Max pooling** further reduces the complexity

Max Pooling



6 x 6 image

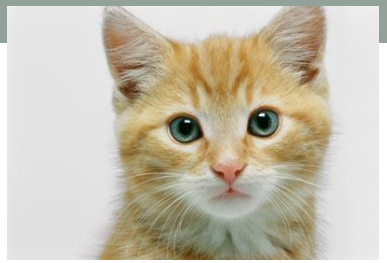
New image but smaller

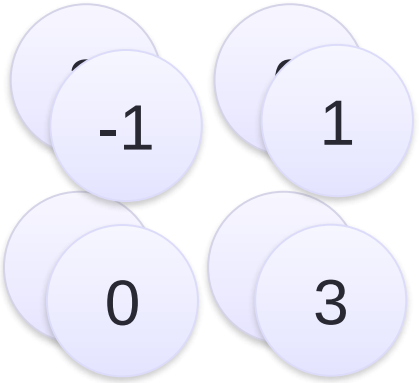


2 x 2 image

Each filter is a channel

The whole CNN





A new image

Convolution

Max Pooling

Convolution

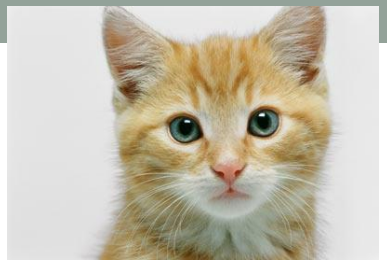
Max Pooling

Can repeat many times

Smaller than the original image

The number of channels is the number of filters

The whole CNN



cat dog ……

Convolution

Max Pooling

A new image

Convolution

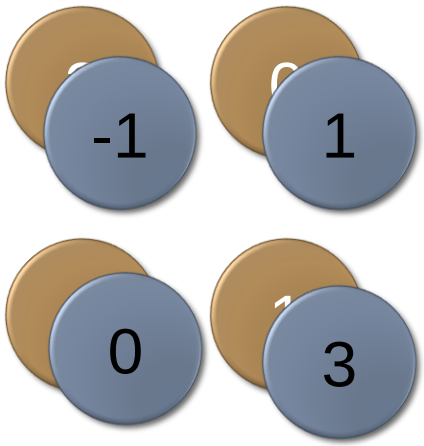
Max Pooling

A new image

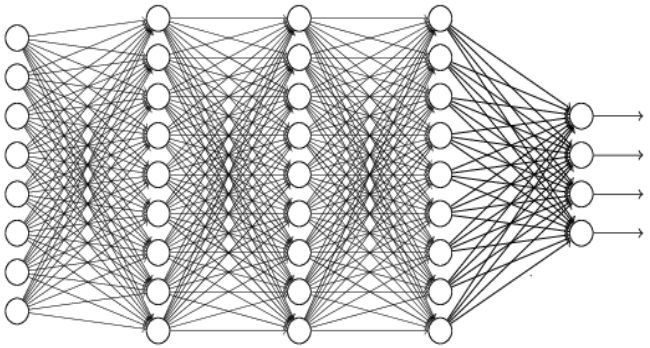
Flattened

Fully Connected Feedforward network

Flattening



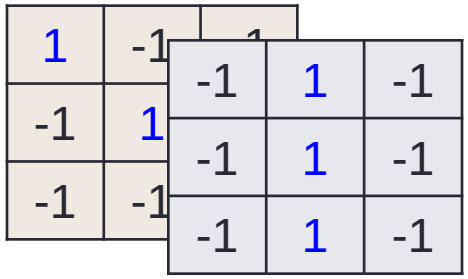
Flattened



Fully Connected
Feedforward network

CNN in Keras

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



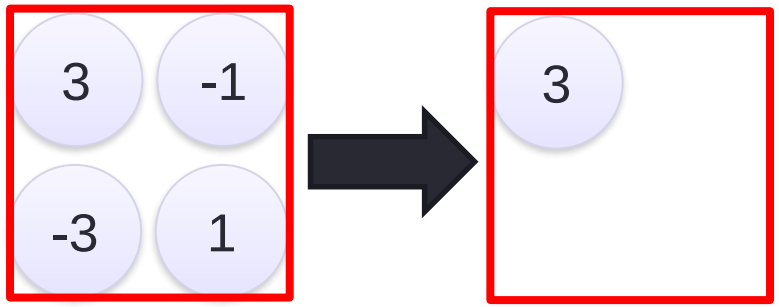
There are 25 3x3 filters.

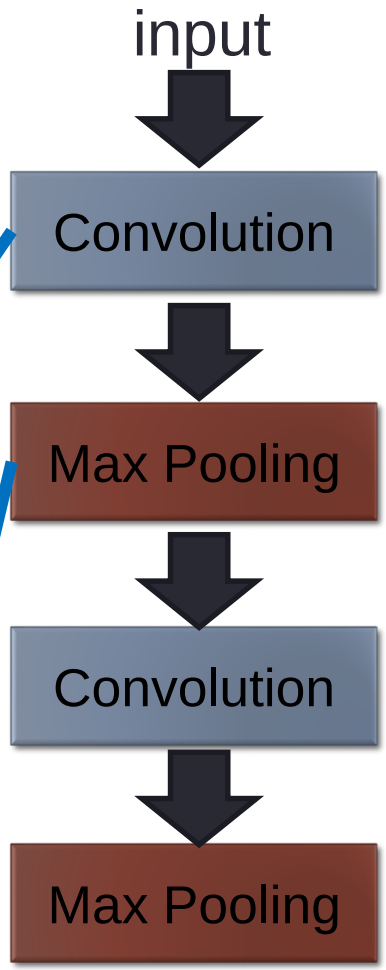
Input_shape = (28 , 28 , 1)

28 x 28 pixels

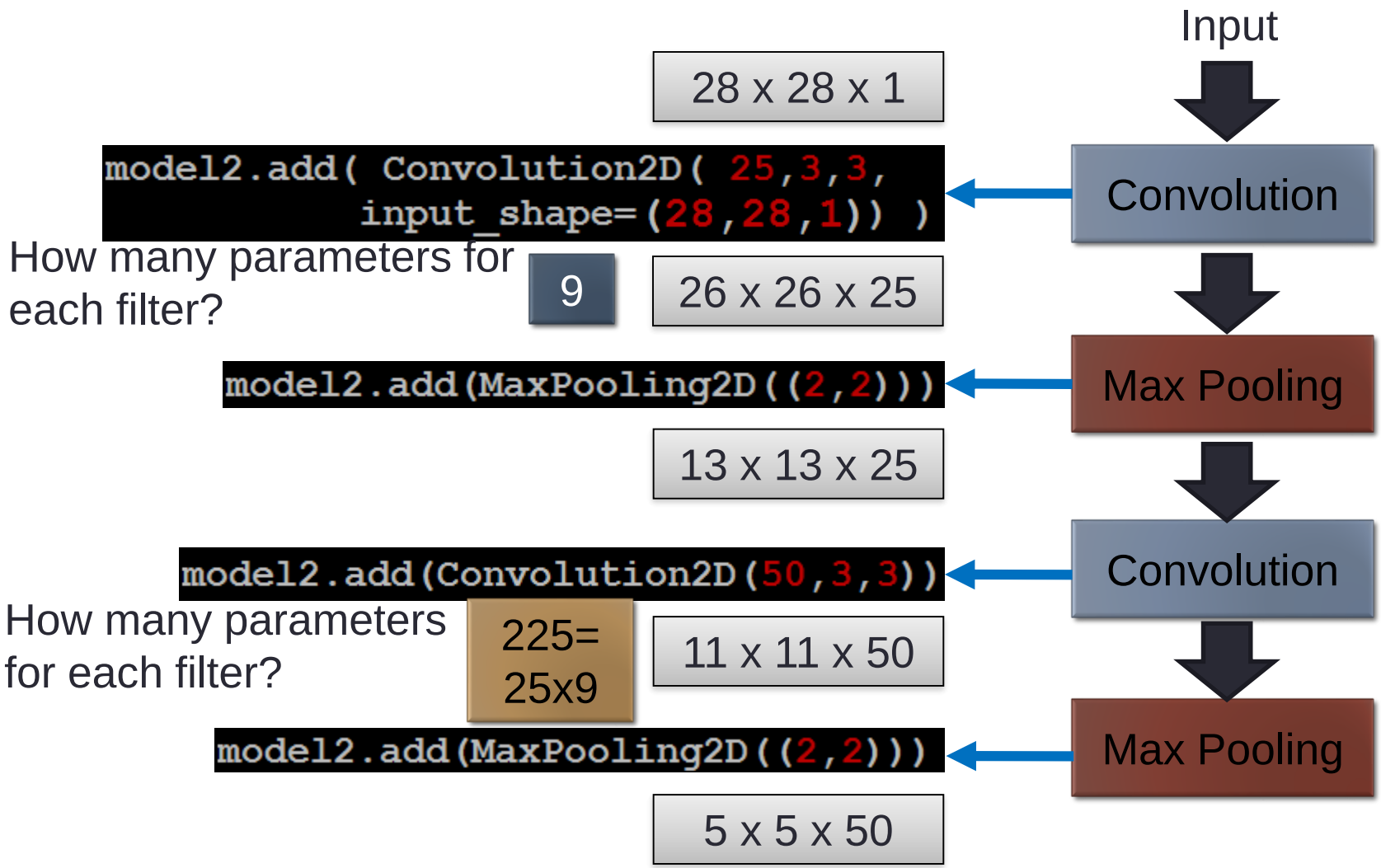
1: black/white, 3: RGB

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

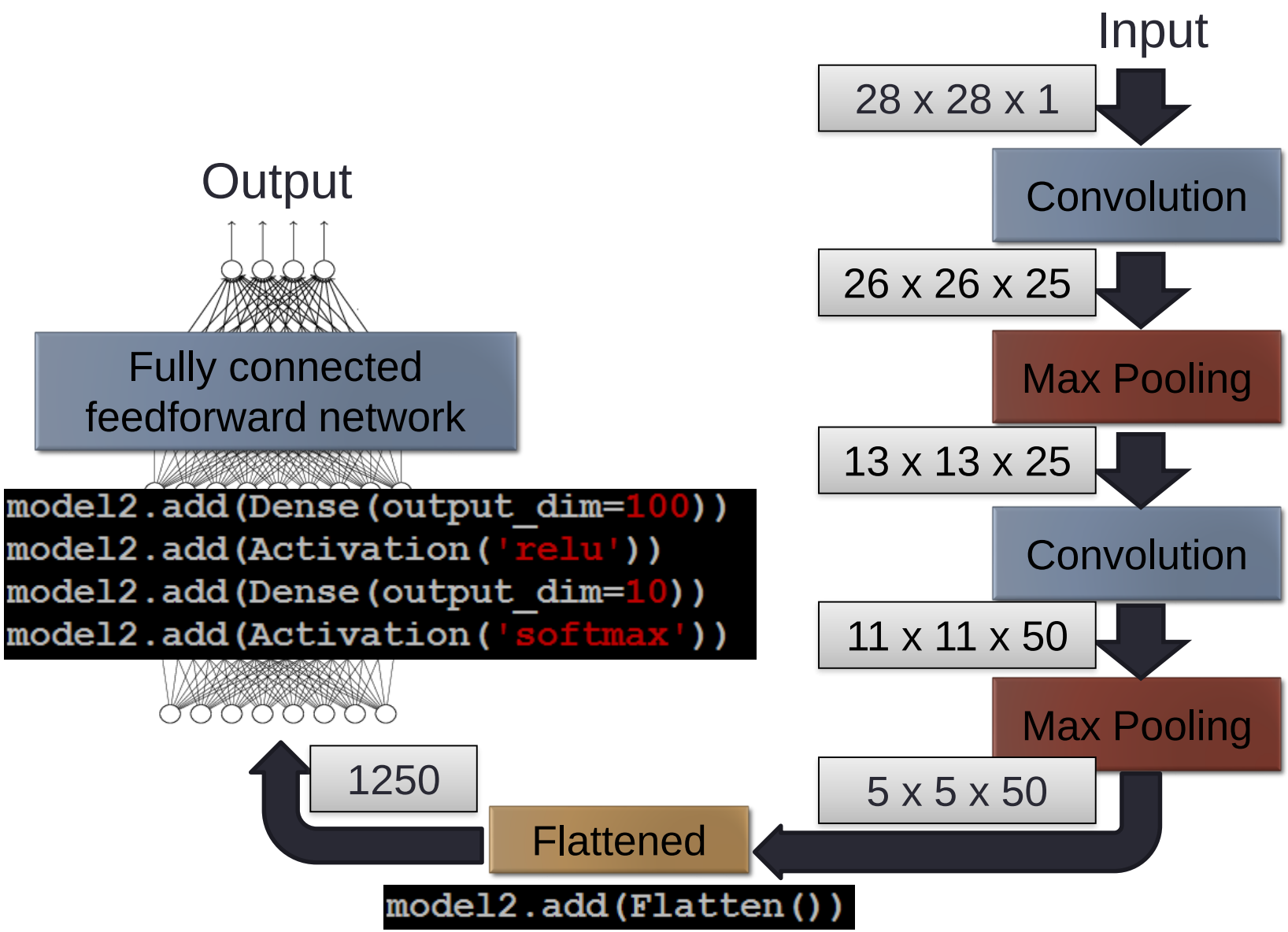




CNN in Keras


Input
28 x 28 x 1
model2.add(Convolution2D(25,3,3, input_shape=(28,28,1)))
Convolution
How many parameters for each filter?
9
26 x 26 x 25
model2.add(MaxPooling2D((2,2)))
Max Pooling
13 x 13 x 25
model2.add(Convolution2D(50,3,3))
Convolution
How many parameters for each filter?
225= 25x9
11 x 11 x 50
model2.add(MaxPooling2D((2,2)))
Max Pooling
5 x 5 x 50

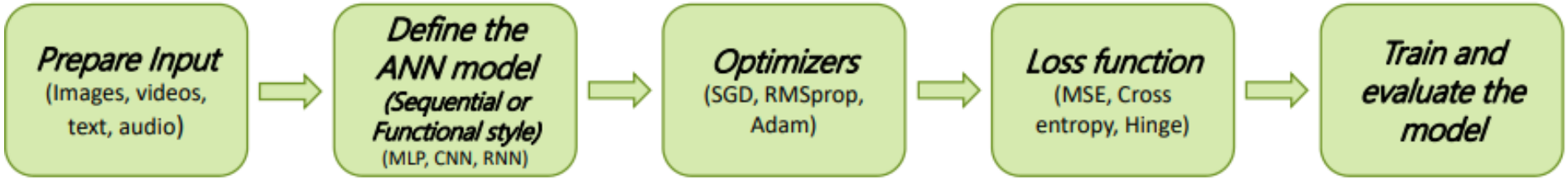
CNN in Keras


Input
28 x 28 x 1
Convolution
26 x 26 x 25
Max Pooling
13 x 13 x 25
Convolution
11 x 11 x 50
Max Pooling
5 x 5 x 50
Flattened
1250
Fully connected feedforward network
Output
model2.add(Dense(output_dim=100))
model2.add(Activation('relu'))
model2.add(Dense(output_dim=10))
model2.add(Activation('softmax'))
model2.add(Flatten())

Implementing CNN in Keras

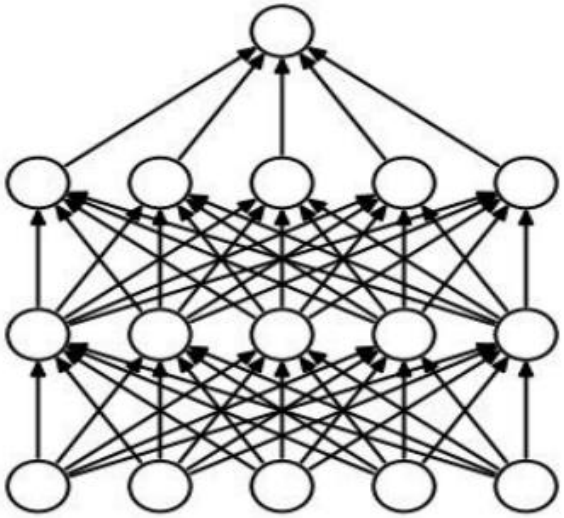
Five major steps

- Preparing the input and specify the **input dimension**
- Define the model **architecture**
- Specify the **optimizer** and **configure** the learning process
- Specify the **Inputs**, **Outputs** of the model and the **Loss function**
- **Train** and **test** the model on the dataset

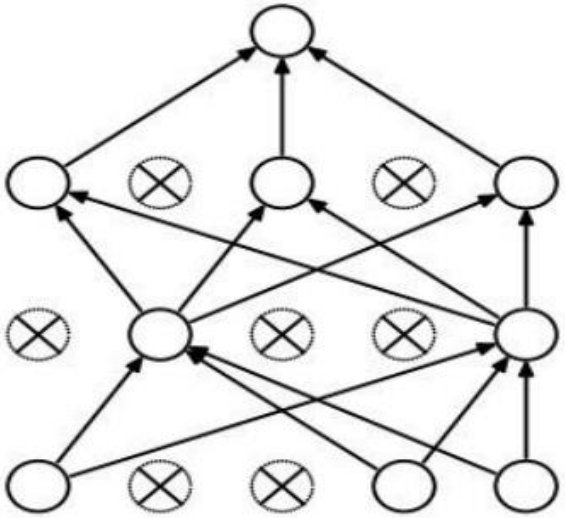


Regularization: Dropout

- Randomly set some neurons to zero in the forward pass
  - Multiply the output of the neuron by zero
  - So its gradient will be zero, so its weight will not get an update



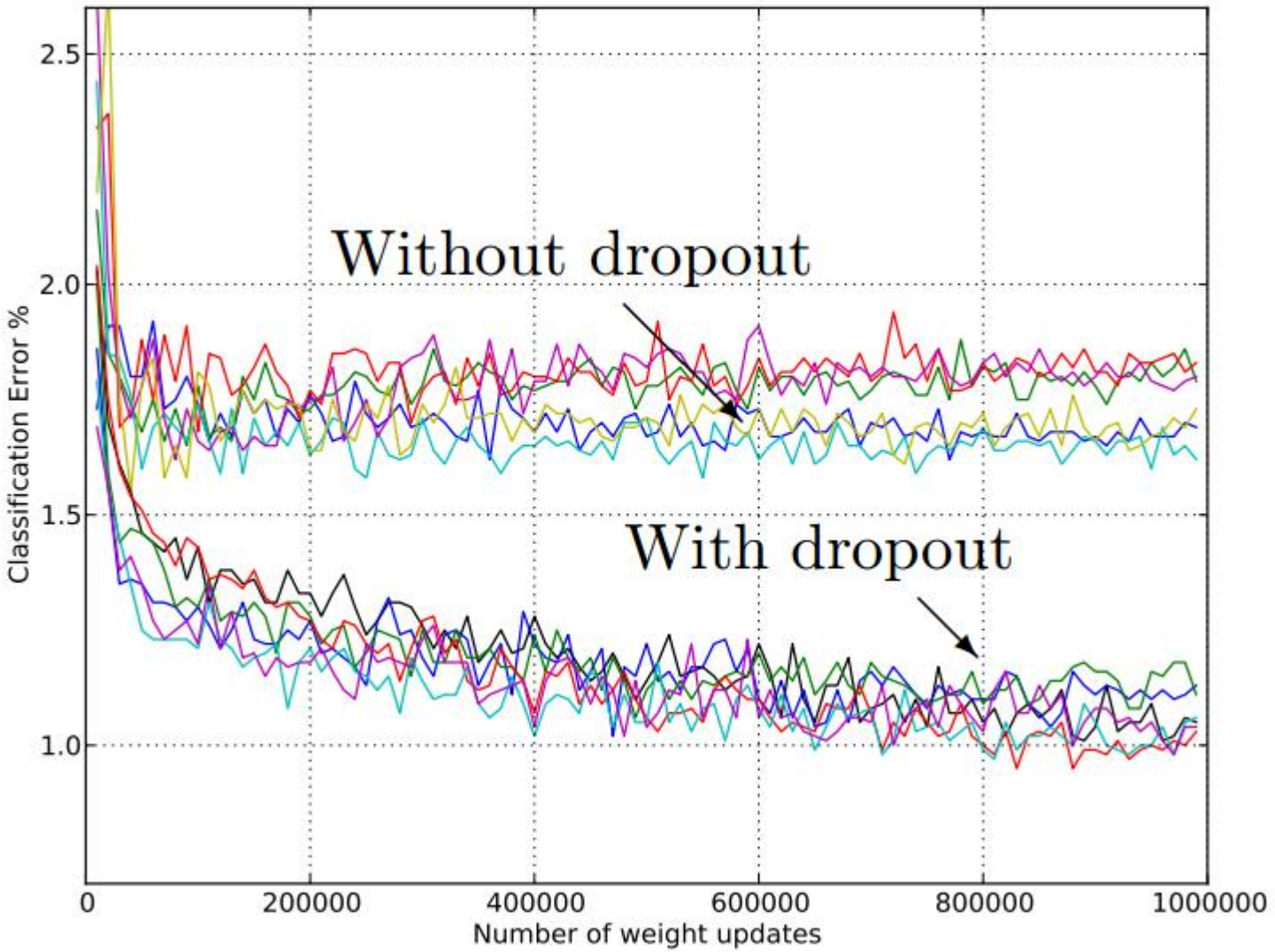
(a) Standard Neural Net



(b) After applying dropout

[Srivastava et al., 2014]

Regularization: Dropout



How deep is enough?

LeNet (1998)

- 2 convolutional layers
- 2 fully connected layers

How deep is enough?

LeNet (1998)

2 convolutional layers
2 fully connected layers

AlexNet (2012)

5 convolutional layers
3 fully connected layers

How deep is enough?

LeNet (1998)

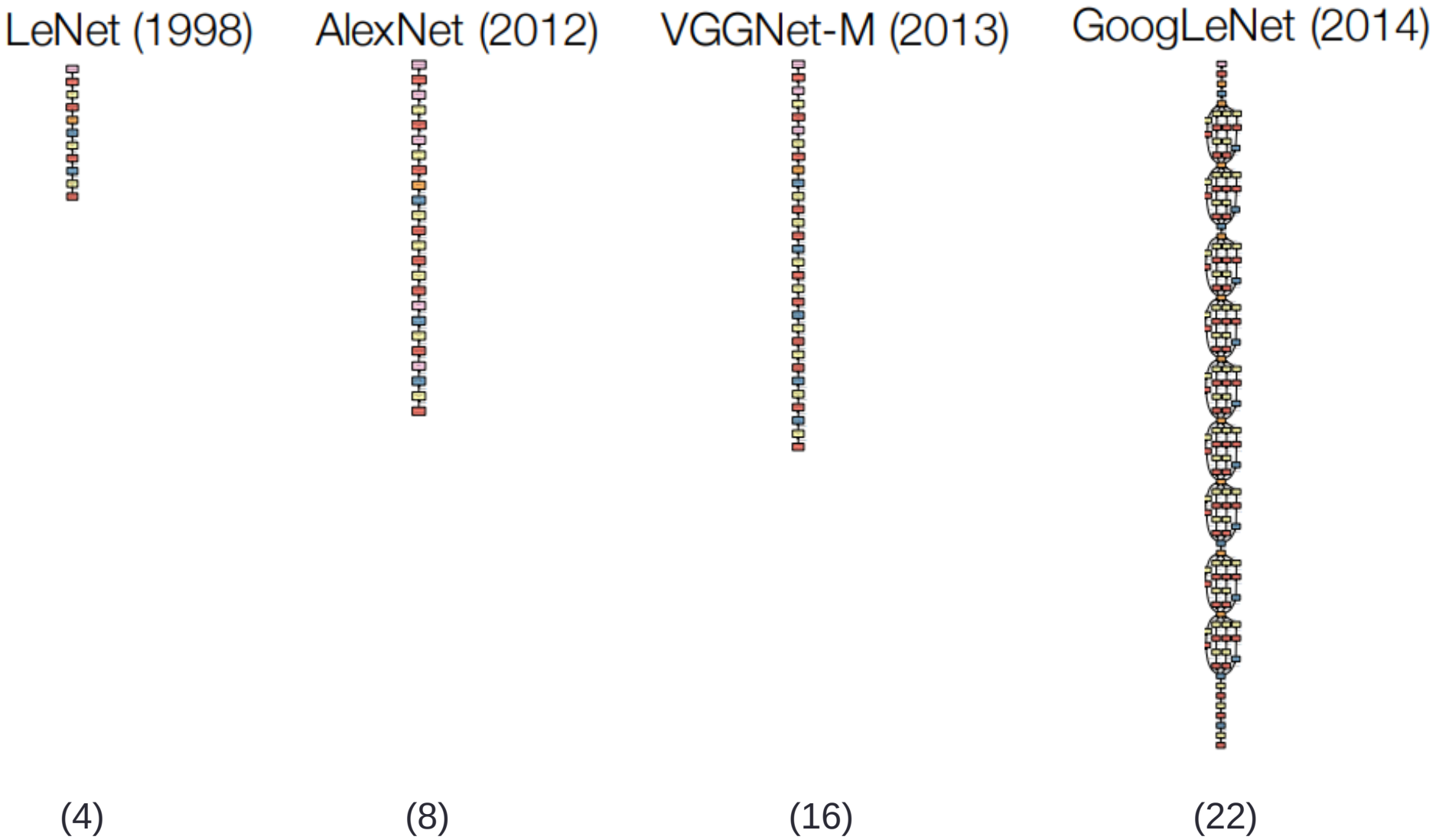
AlexNet (2012)

VGGNet-M (2013)

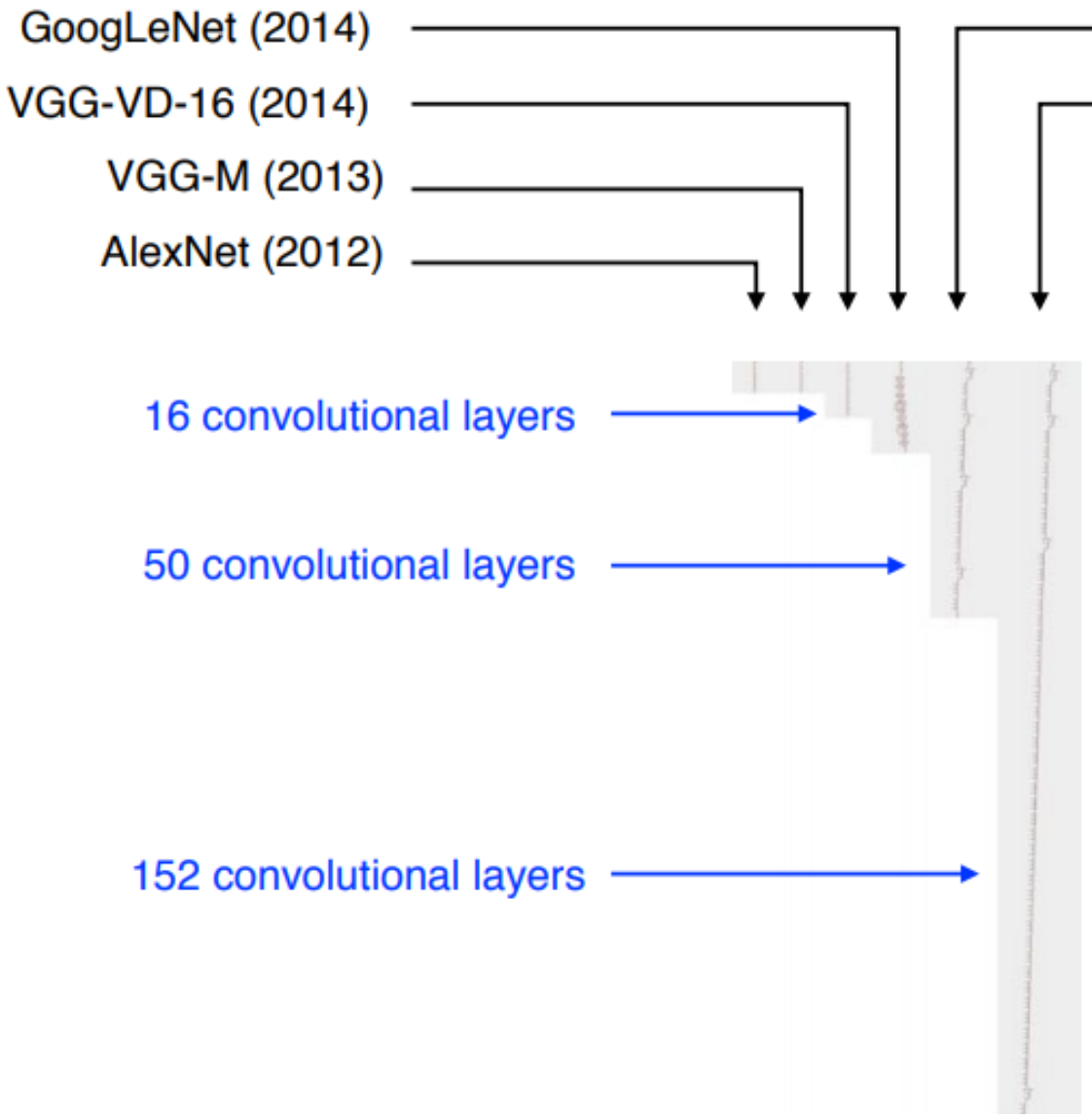
How deep is enough?

http://knowyourmeme.com/memes/we-need-to-go-deeper

How deep is enough?



How deep is enough?



Krizhevsky, I. Sutskever, and G. E. Hinton. *ImageNet classification with deep convolutional neural networks*. In Proc. NIPS, 2012.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. *Going deeper with convolutions*. In Proc. CVPR, 2015.

K. Simonyan and A. Zisserman. *Very deep convolutional networks for large-scale image recognition*. In Proc. ICLR, 2015.

K. He, X. Zhang, S. Ren, and J. Sun. *Deep residual learning for image recognition*. In Proc. CVPR, 2016.

DL Frameworks

Transfer Learning with CNNs

- Keep layers 1-7 of ImageNet-trained model fixed
- Train a new softmax classifier on top using the training images of the new dataset.