Introduction to Convolutional Neural Networks / Deep Learning

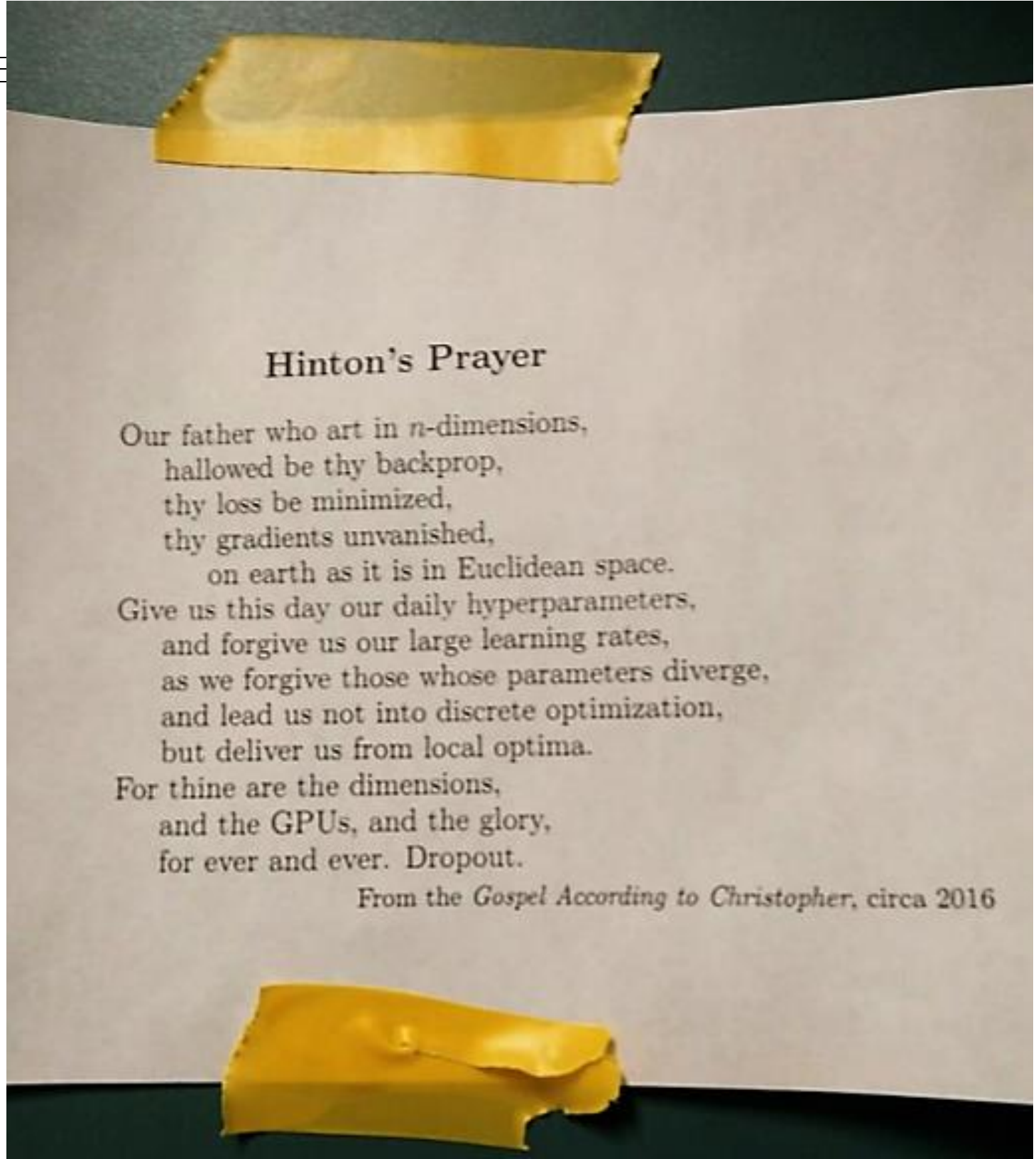
Ravi Kiran Sarvadevabhatla

https://ravika.github.io/

G. Hinton (U.Toronto, Google DeepMind)

Yann LeCun (Facebook AI)

Y. Bengio (MILA, Montreal)

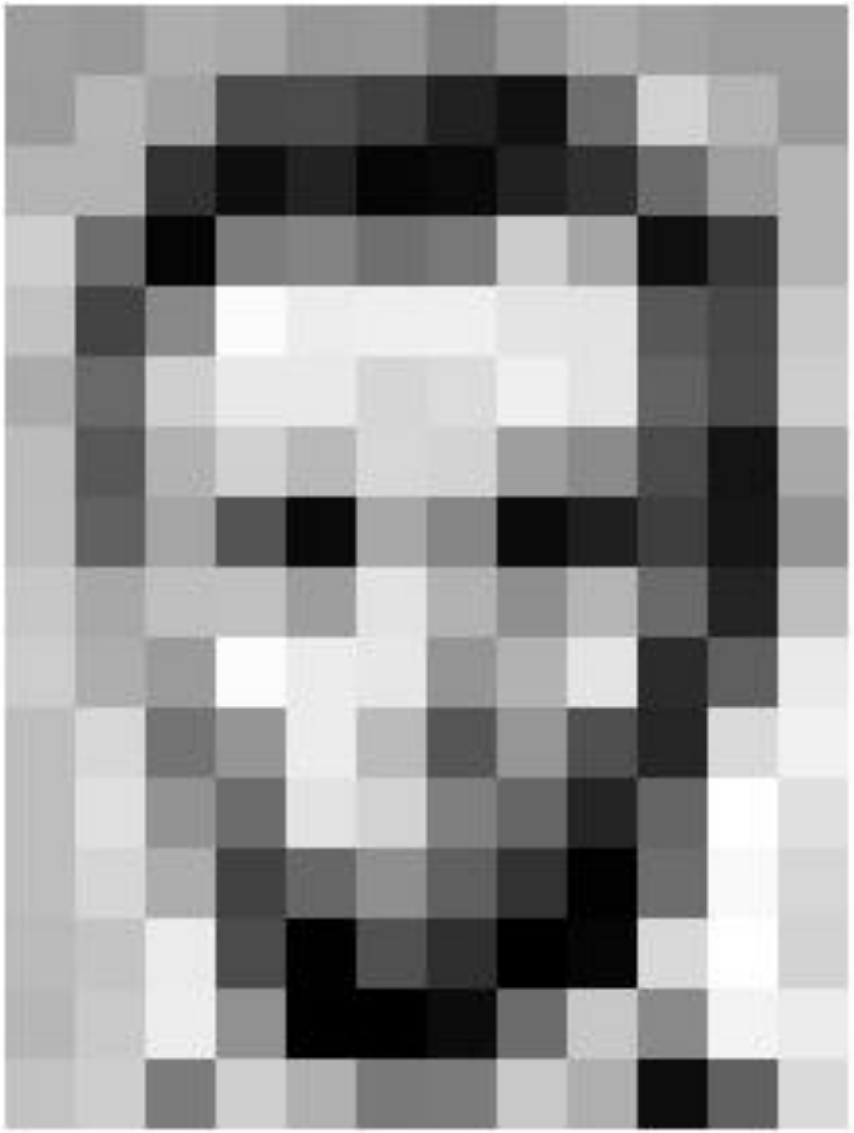


Hinton's Prayer

Our father who art in *n*-dimensions,
hallowed be thy backprop,
thy loss be minimized,
thy gradients unvanished,
on earth as it is in Euclidean space.
Give us this day our daily hyperparameters,
and forgive us our large learning rates,
as we forgive those whose parameters diverge,
and lead us not into discrete optimization,
but deliver us from local optima.
For thine are the dimensions,
and the GPUs, and the glory,
for ever and ever. Dropout.

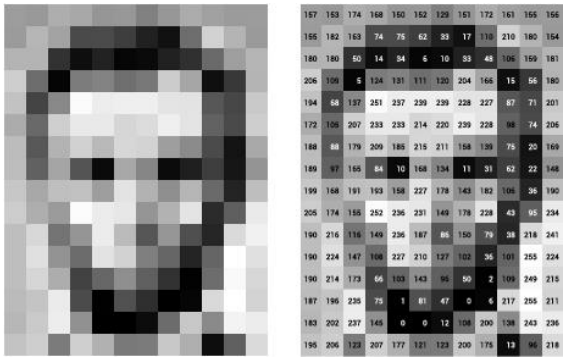
From the *Gospel According to Christopher*, circa 2016

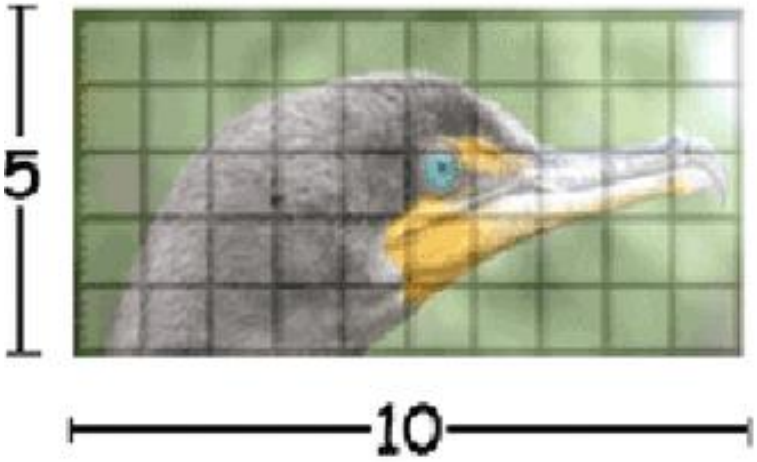
Image Representation

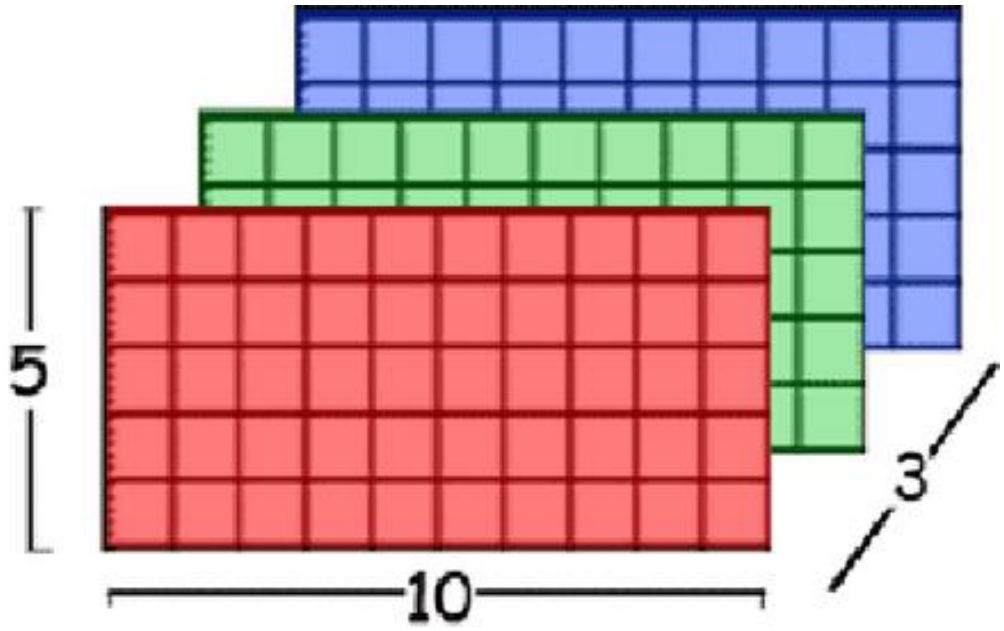


157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	95	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

Image Representation







A Case Study: Fish Classification

- Problem:
  - sort incoming fish on a conveyor belt according to species
  - Assume only two classes exist:
    - Sea Bass and Salmon

Source: https://www.slideshare.net/lgustavomartins/introduction-to-pattern-recognition

Features

- Features are properties of an object:
  - Ideally representative of a specific type (i.e. class) of objects
  - Compact (memory efficient)
  - Computationally simple (CPU efficient)
  - Perceptual relevant (if trying to implement a human inspired classifier)
- **=> Should pave the way to a good discrimination of different classes of objects!**

Source: https://www.slideshare.net/lgustavomartins/introduction-to-pattern-recognition

Features

- Take a group of graphical objects
  - Possible features:
    - Shape
    - Color
    - Size
    - ...
  - Allows to group them into different classes:

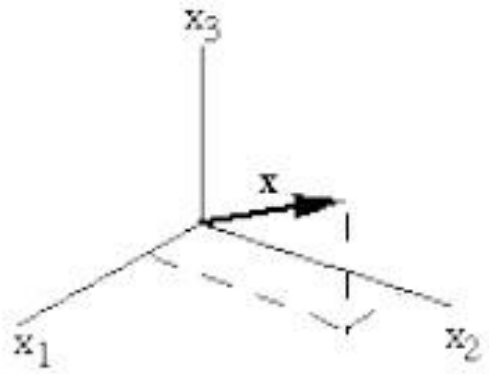
Source: https://www.slideshare.net/lgustavomartins/introduction-to-pattern-recognition

Feature Vectors

- Usually a single object can be represented using several features, e.g.
  - **x1** = shape (e.g. nr of sides)
  - **x2** = size (e.g. some numeric value)
  - **x3** = color (e.g. rgb values)
  - ...
  - **xd** = some other (numeric) feature.

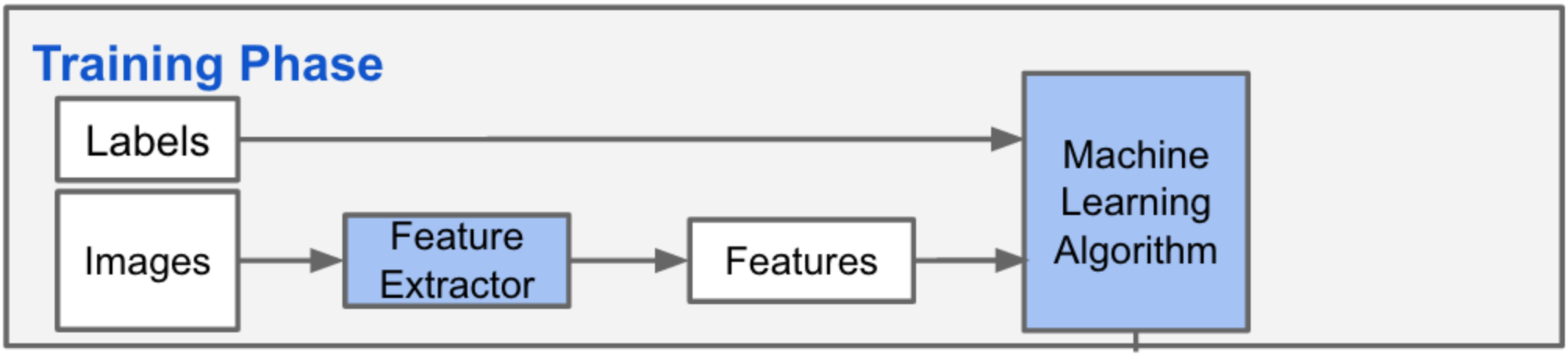
- **X** becomes a feature vector
  - x is a point in a d-dimensional **feature space**.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_d \end{bmatrix}$$



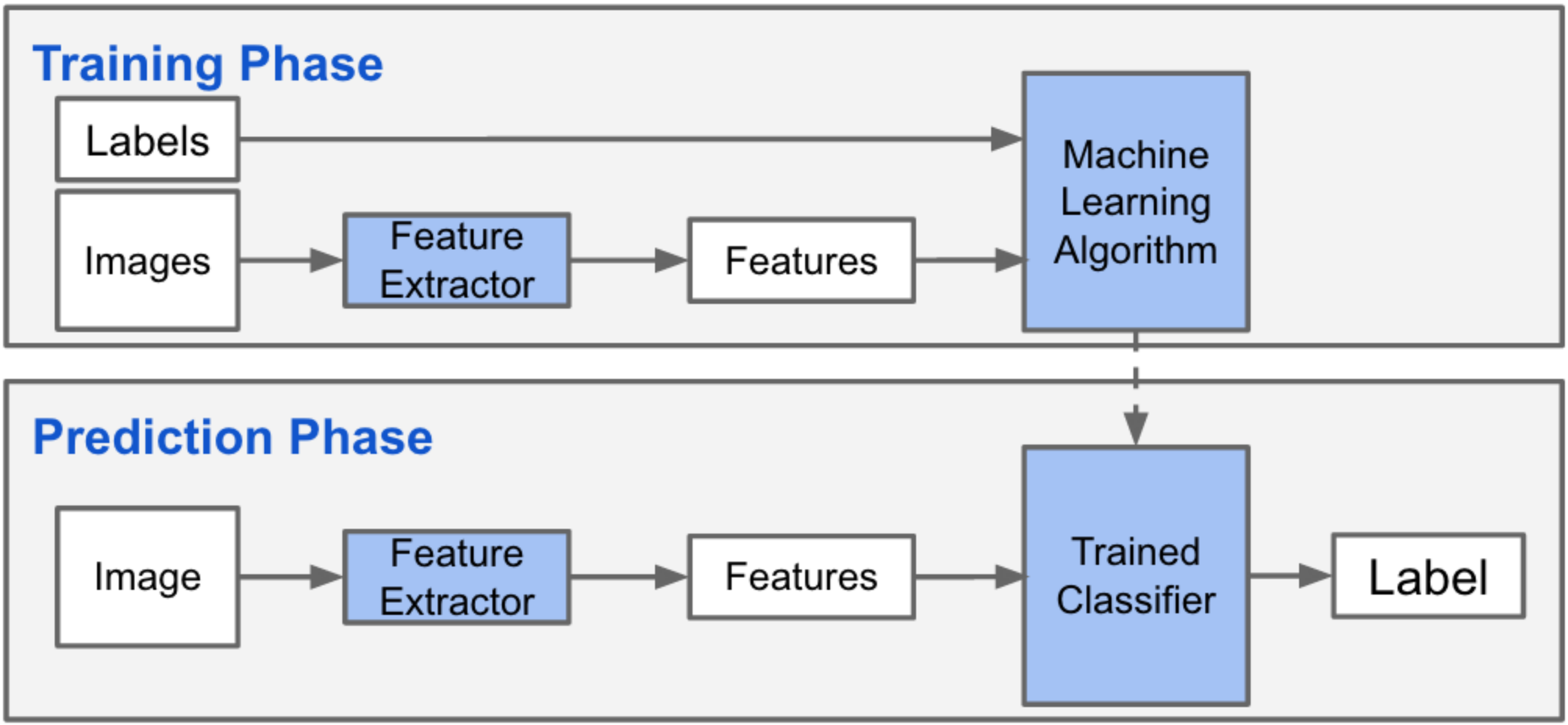
Source: https://www.slideshare.net/lgustavomartins/introduction-to-pattern-recognition

Classification



Source: http://adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/

Classification



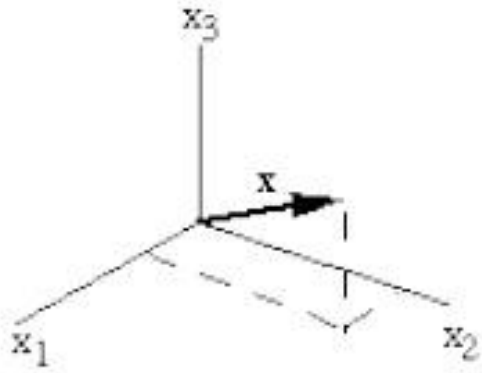
Source: http://adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/

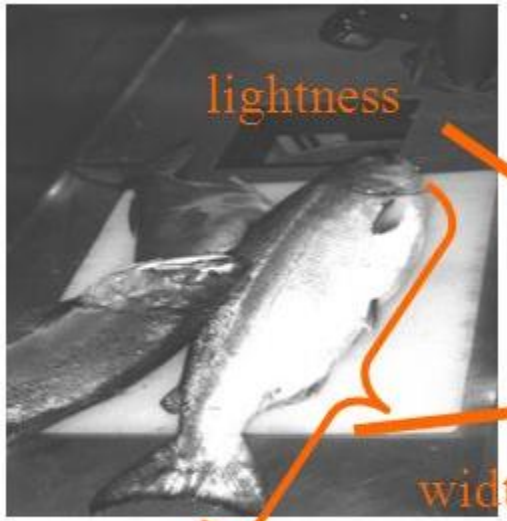
Feature Vectors

- Usually a single object can be represented using several features, e.g.
  - **x1** = shape (e.g. nr of sides)
  - **x2** = size (e.g. some numeric value)
  - **x3** = color (e.g. rgb values)
  - ...
  - **xd** = some other (numeric) feature.

- **X** becomes a feature vector
  - x is a point in a d-dimensional **feature space**.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_d \end{bmatrix}$$

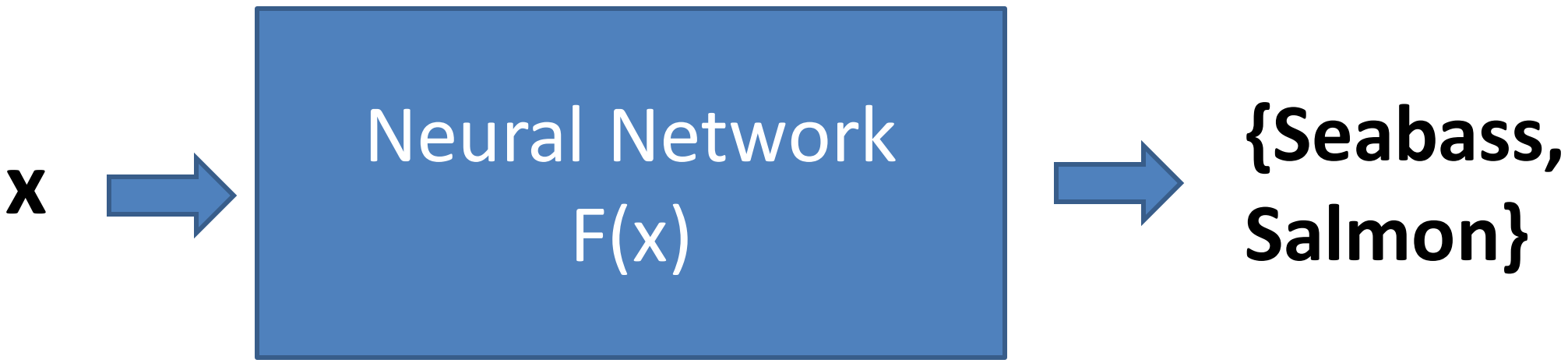




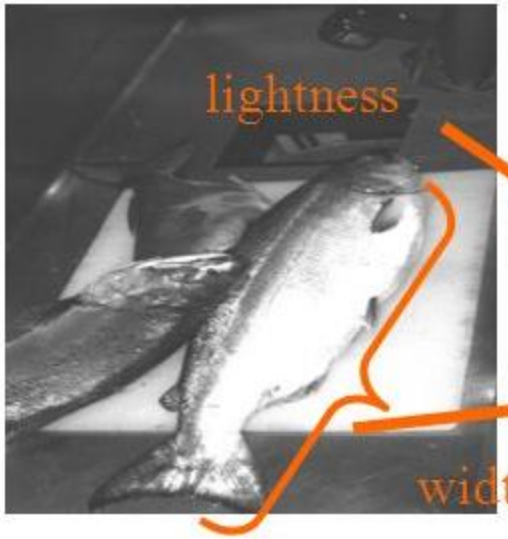
$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \mathbf{x}$$

Source: https://w

Neural Network = A function approximator



The simplest neural network

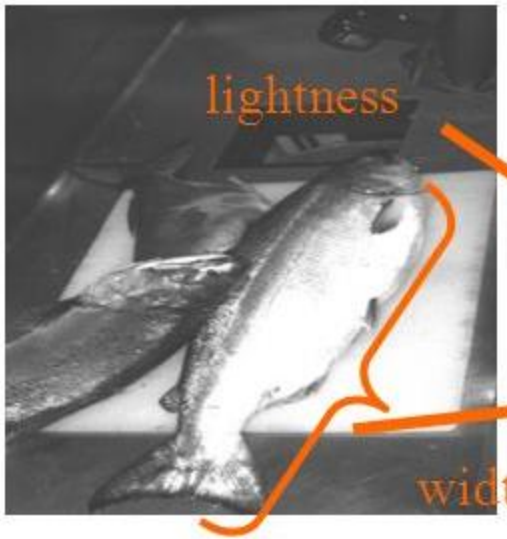


$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \mathbf{x}$$

Neuron

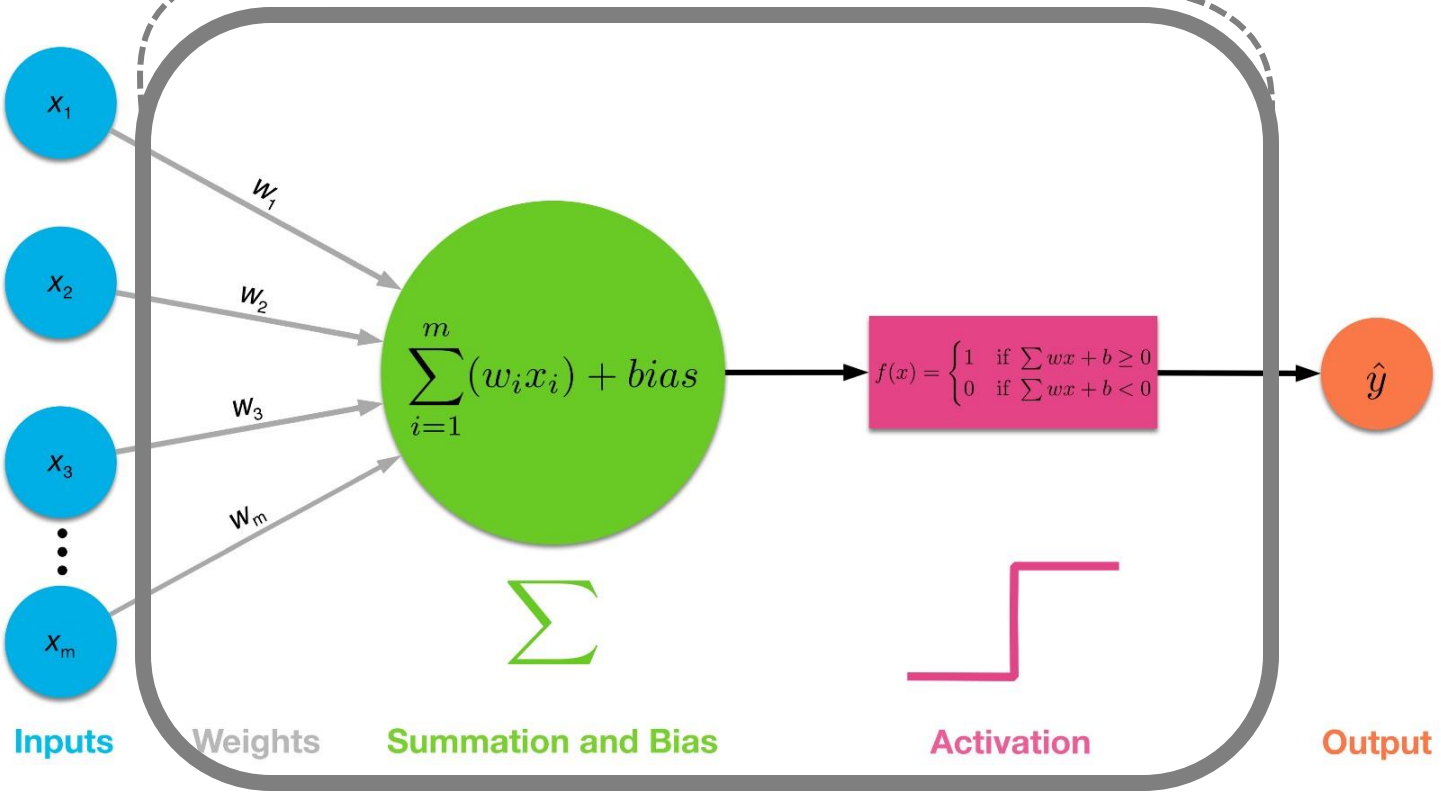
{Seabass, Salmon}

The simplest neural network

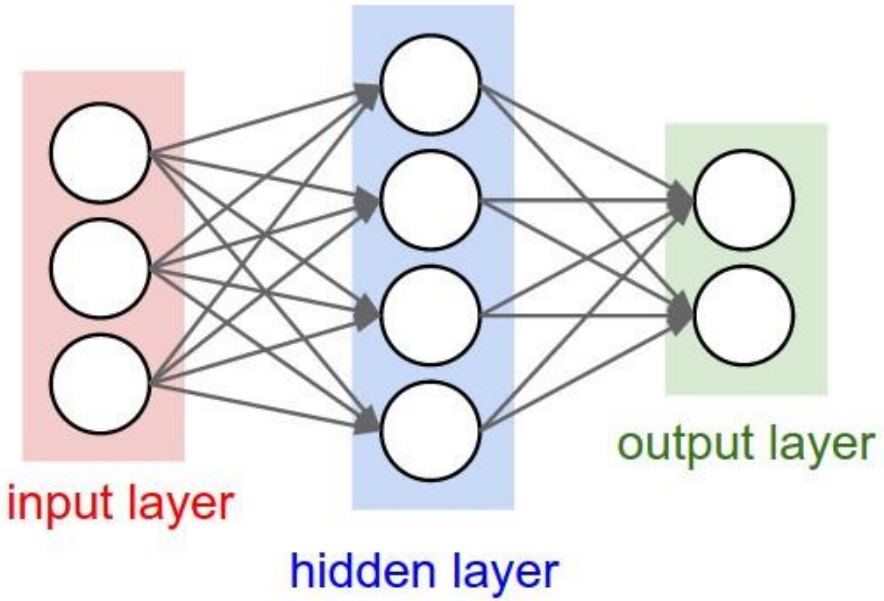


$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \mathbf{x}$$

{Seabass, Salmon}

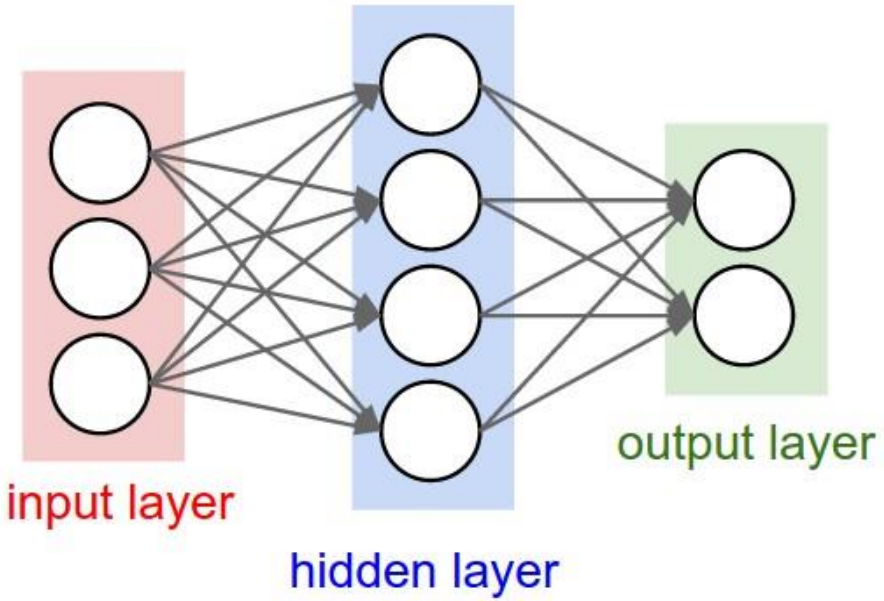


Writing “code” (Keras) …



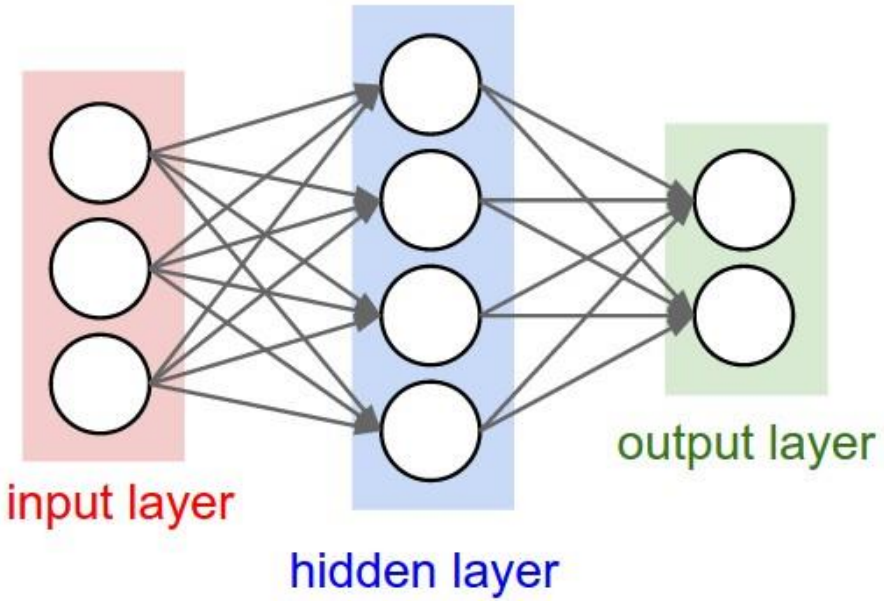

```
model = Sequential()
model.add(Dense(4,input_dim=3)
model.add(Activation('tanh'))
model.add(Dense(2))
model.compile(loss='mean_squared_error', optimizer=adam)
```

Training the model



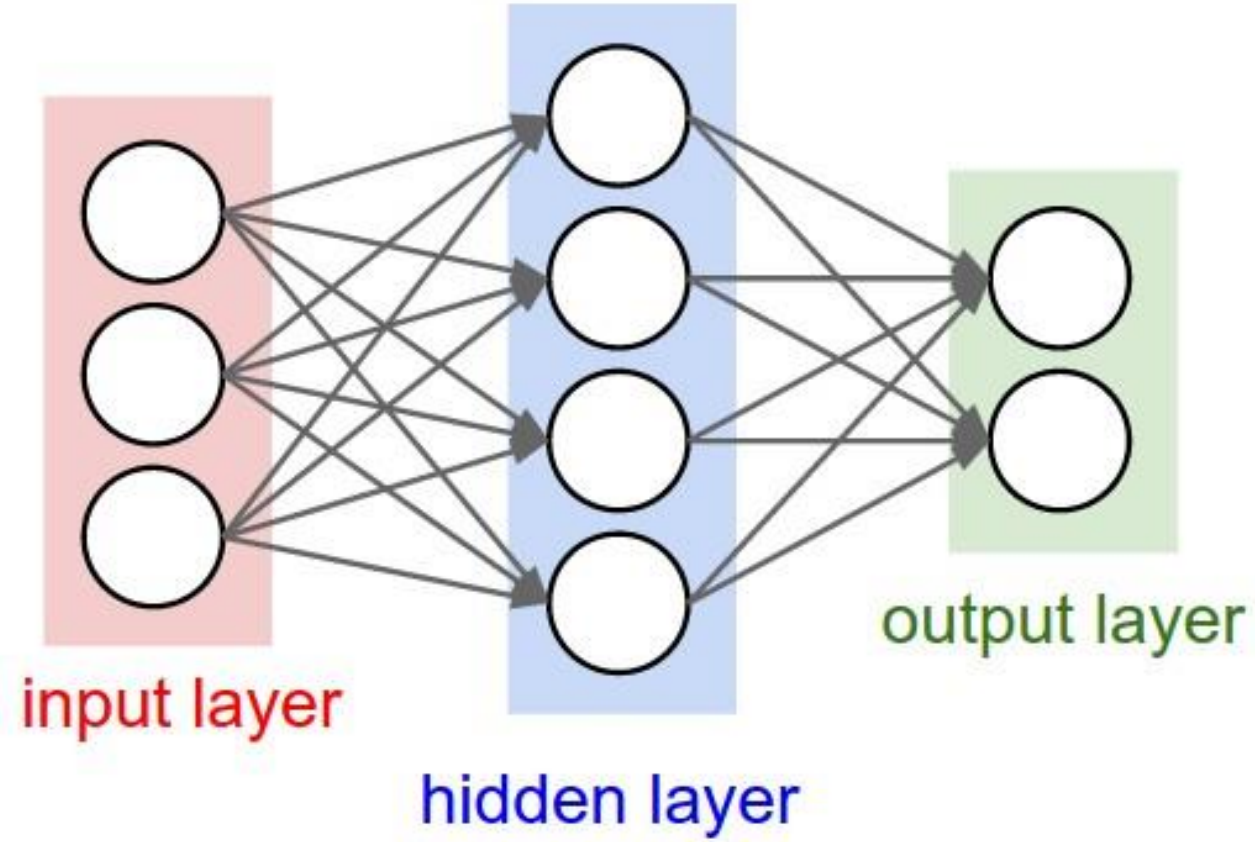

```
# Training
# X_train : [num-train-samples x 3]
# Y_train : [num-train-samples x 2]
model.fit(X_train, y_train)
```

Model evaluation / Prediction




```
# Prediction
# X_test : [num-test-samples x 3]
# Y_test : [num-test-samples x 2]
score = model.evaluate(X_test, y_test)
```

Traditional approach: Extract features, feed to NN

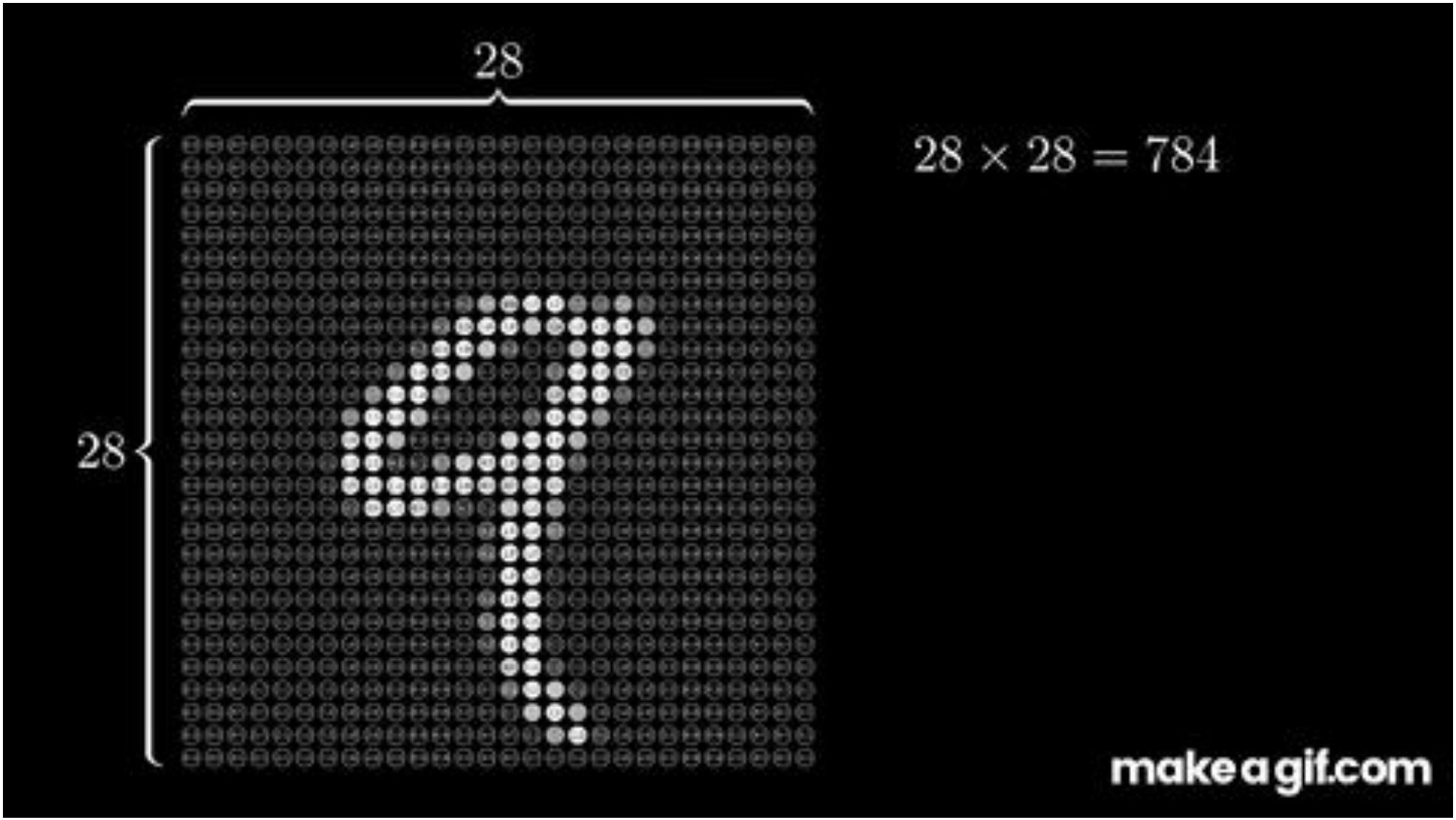


Wouldn't it be nice if we could feed the image directly ?

… and let the “learning process” figure out which features to extract ?

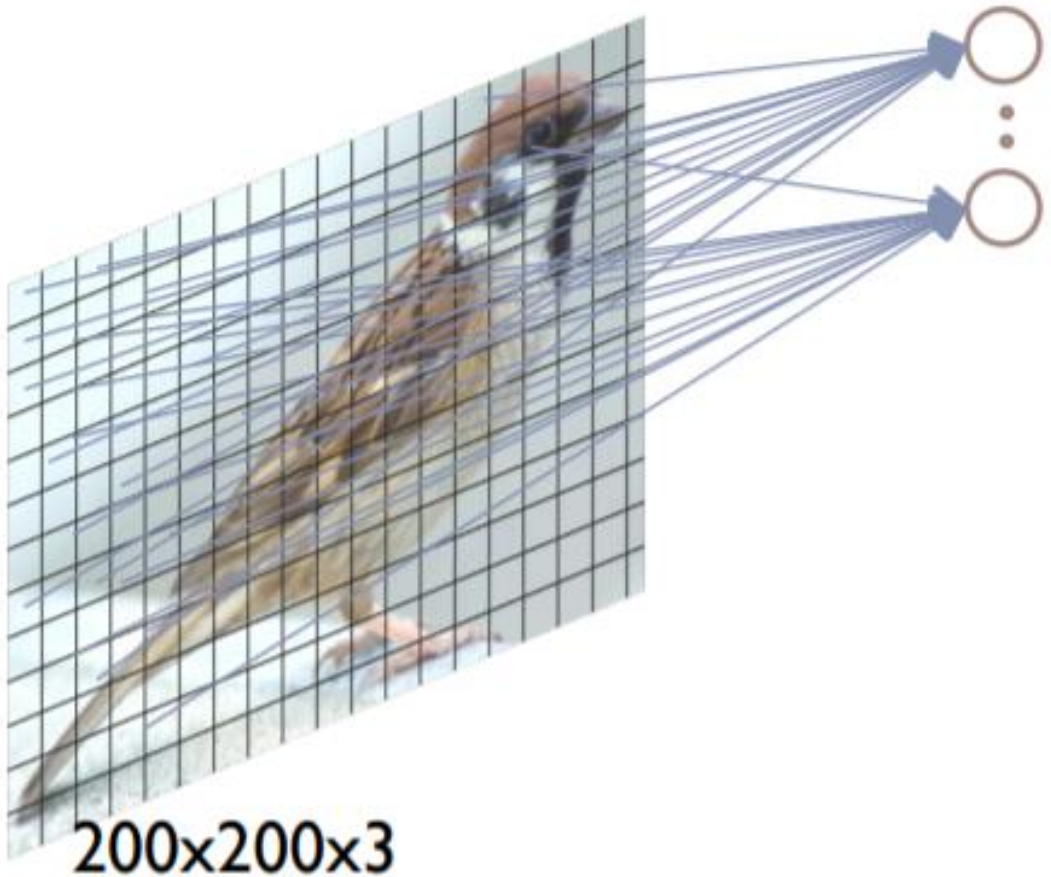
MNIST Handwritten Digits Dataset

Flatten image , Feed to NN



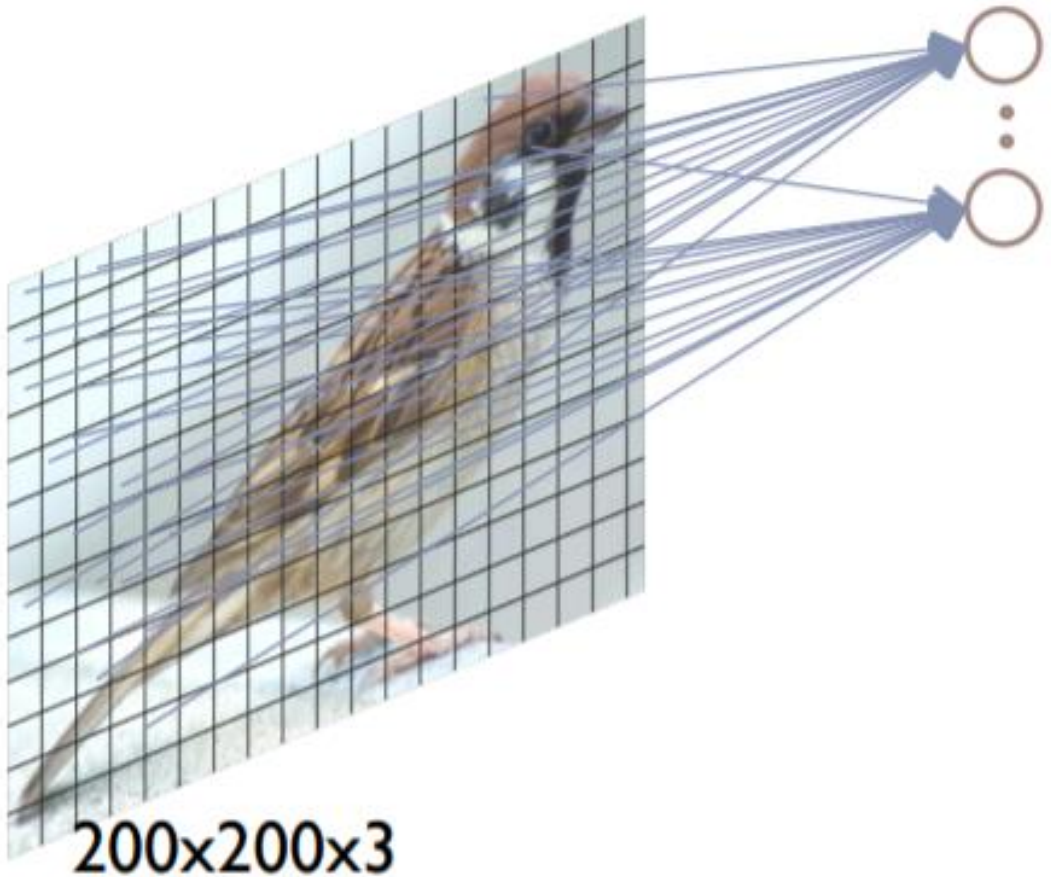
https://www.youtube.com/watch?v=aircAruvnKk

For color images ?



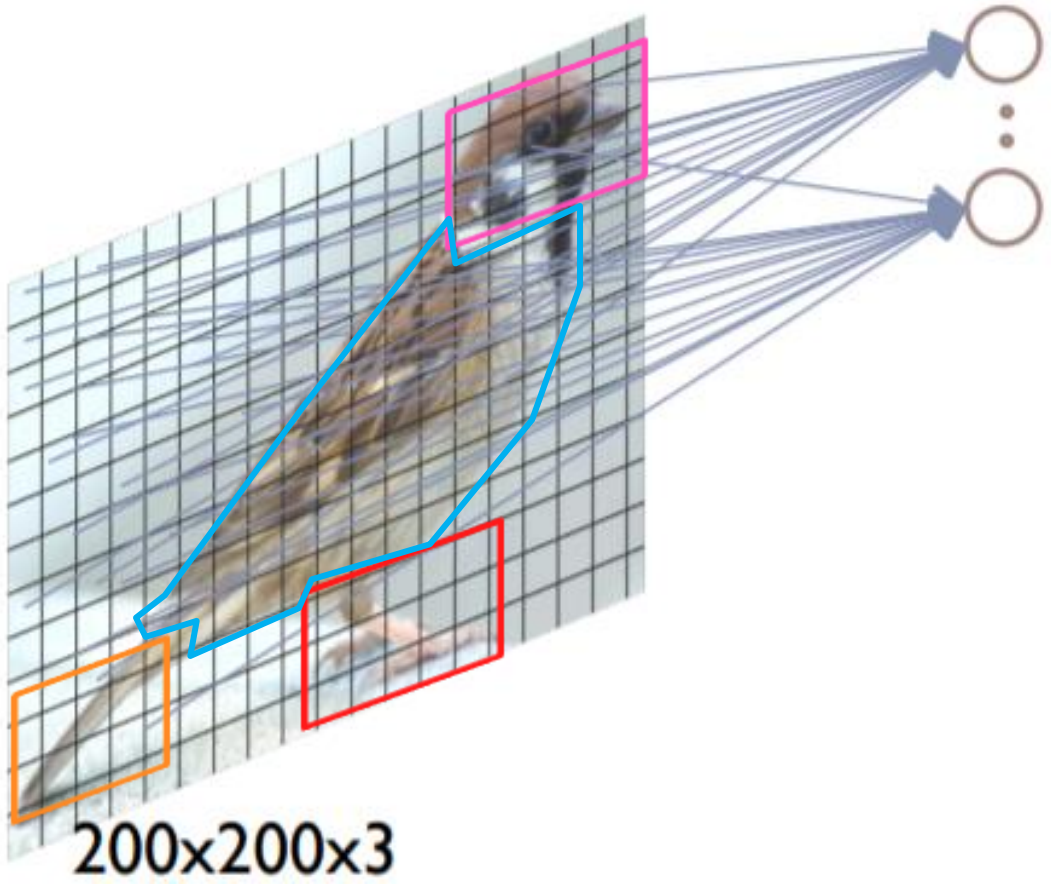
- #Hidden Units: 120,000
- #Params: 14.4 billion
- Need huge training data to prevent over-fitting!

Do we really need full connectivity ?



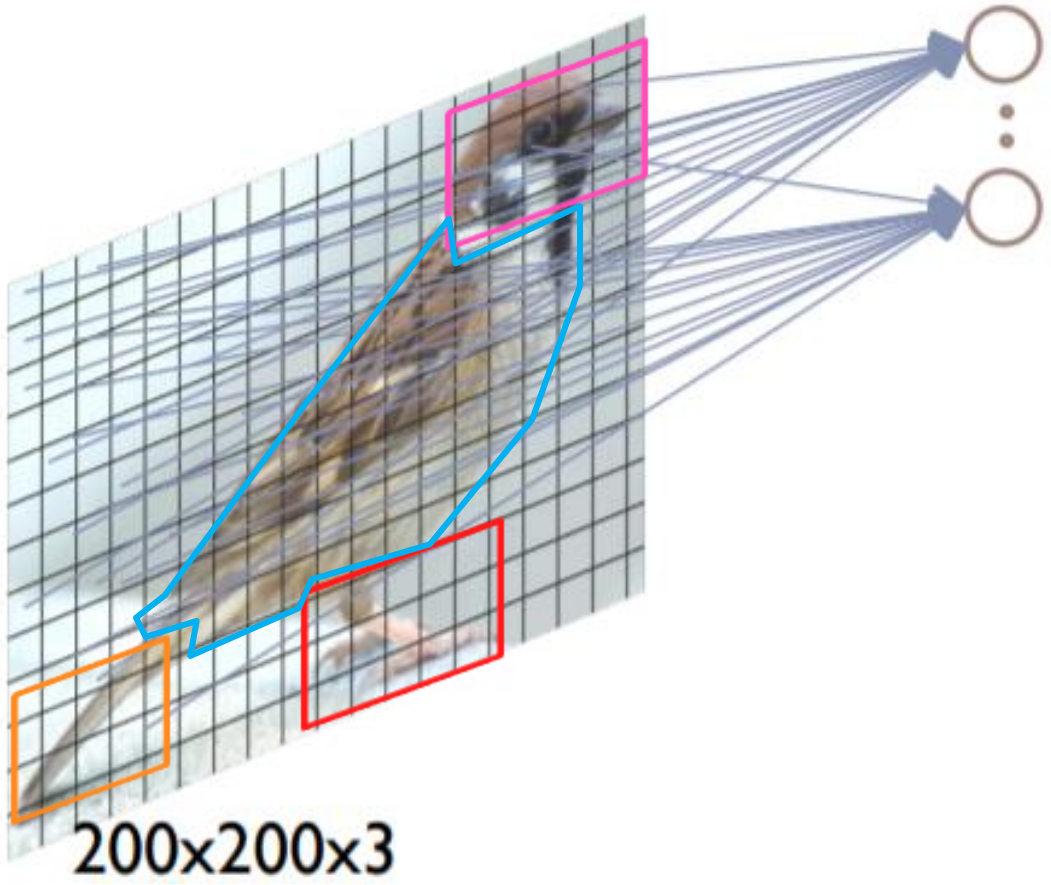
- #Hidden Units: 120,000
- #Params: 14.4 billion
- Need huge training data to prevent over-fitting!

Do we really need full connectivity ?



- Head + Body + Tail + Legs = sparrow

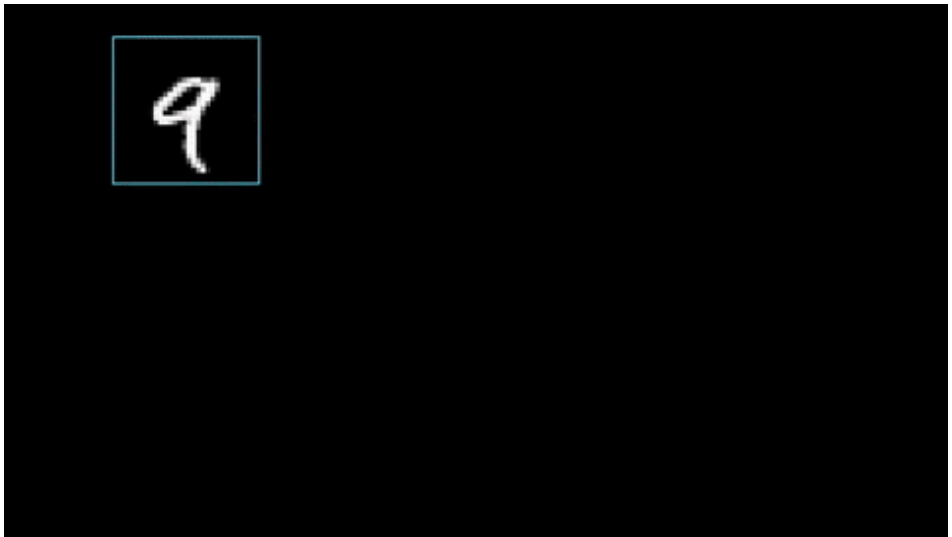
Do we really need full connectivity ?



- Head + Body + Tail + Legs = sparrow

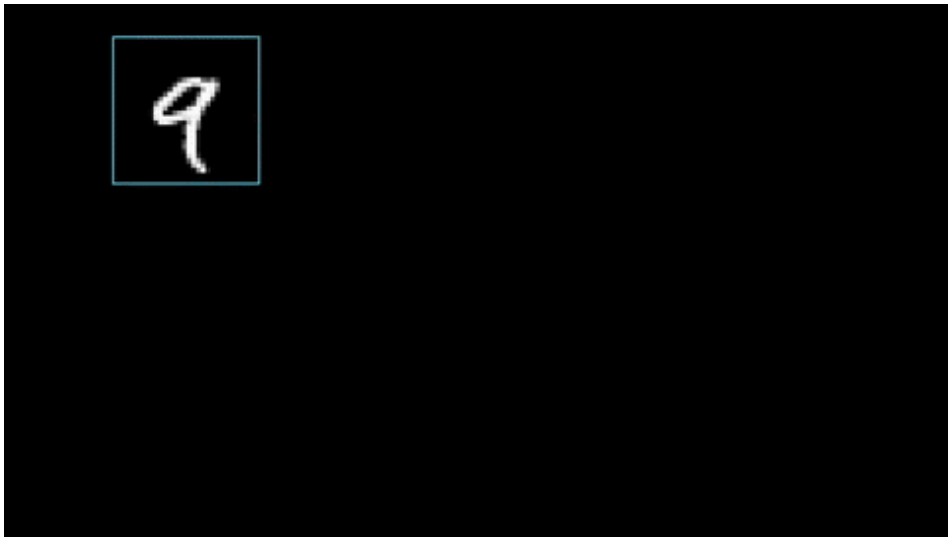
https://www.youtube.com/watch?v=aircAruvnKk

- Images are 2D.
- Assumption: Object image = combination of **2D image patterns**



- Images are 2D.
- Assumption: Object image = combination of **2D image patterns**

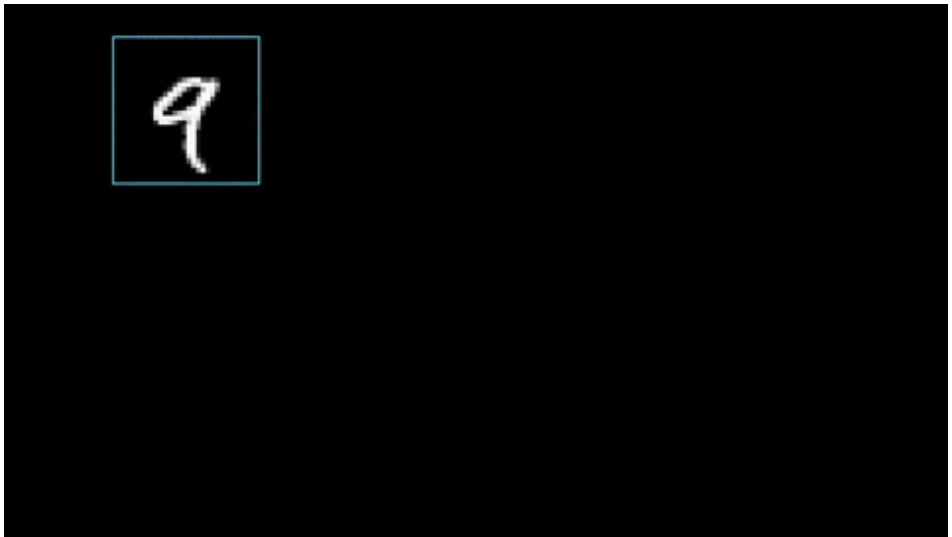
- Machine Learning Strategy:
  a) [Pre-final layer] Determine 2D image patterns
  b) Map 2D image patterns → Target label



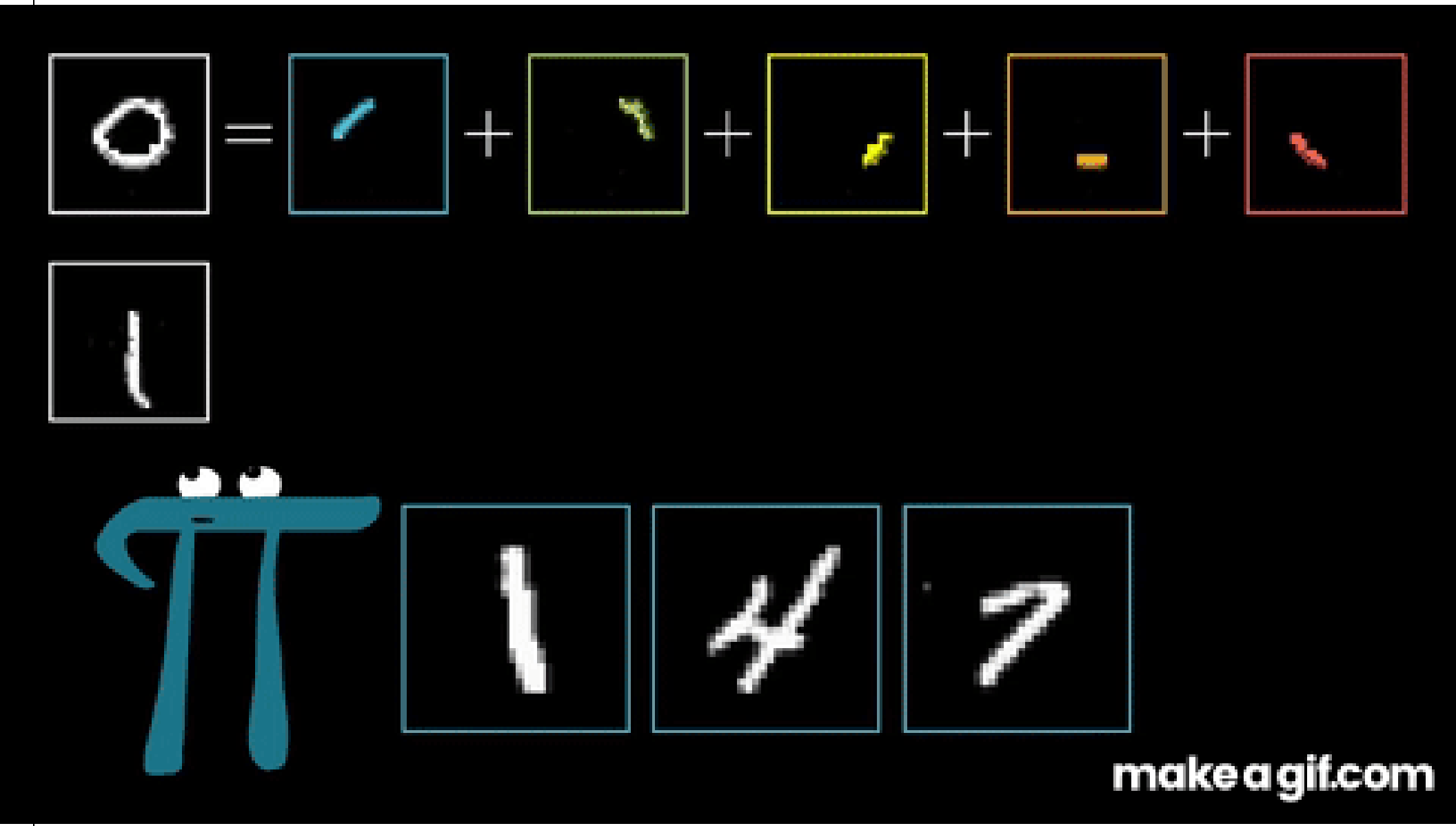
- Images are 2D.
- Assumption: Object image = combination of **2D image patterns**

- Machine Learning Strategy:
  - [Pre-final layer] Determine 2D image patterns
  - Map 2D image patterns → Target label

- NOTE: 2D image patterns are smaller than image



Hierarchical decomposition of input/features

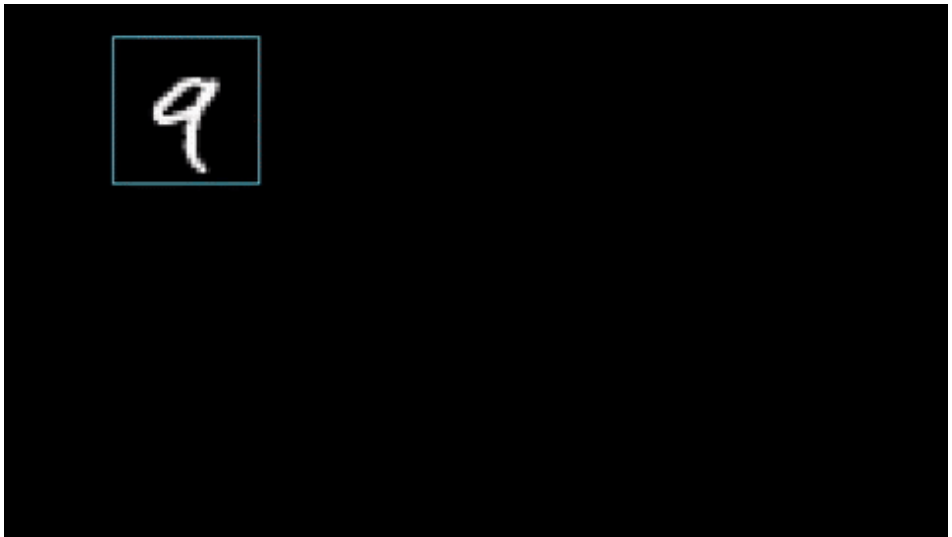


https://www.youtube.com/watch?v=aircAruvnKk

- Images are 2D.
- Assumption: Object image = **hierarchical** combination of **2D image patterns**

- Machine Learning Strategy:
  - [All except pre-final] Determine 2D image patterns
  - [Pre-final layer] Map 2D image patterns → Target label

- NOTE: 2D image patterns are smaller than image

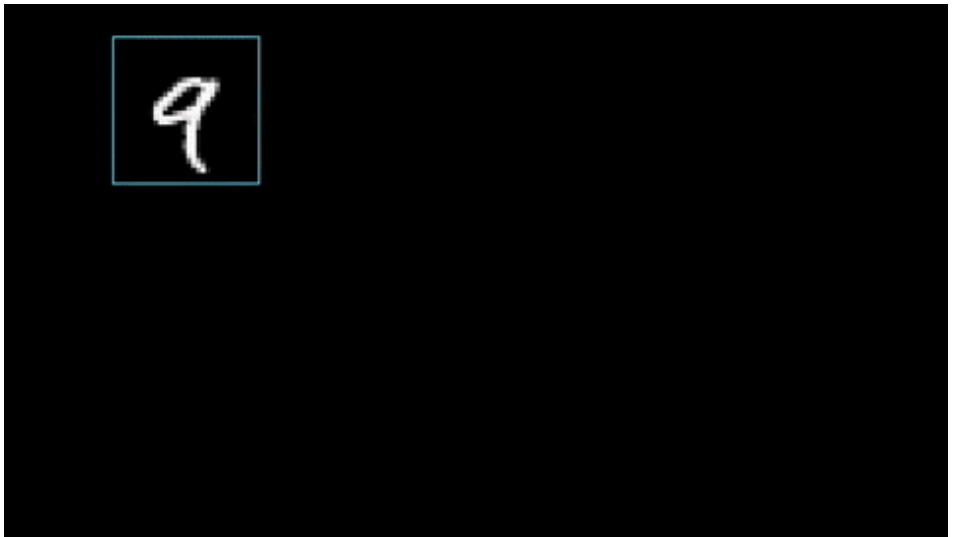


Efficient !

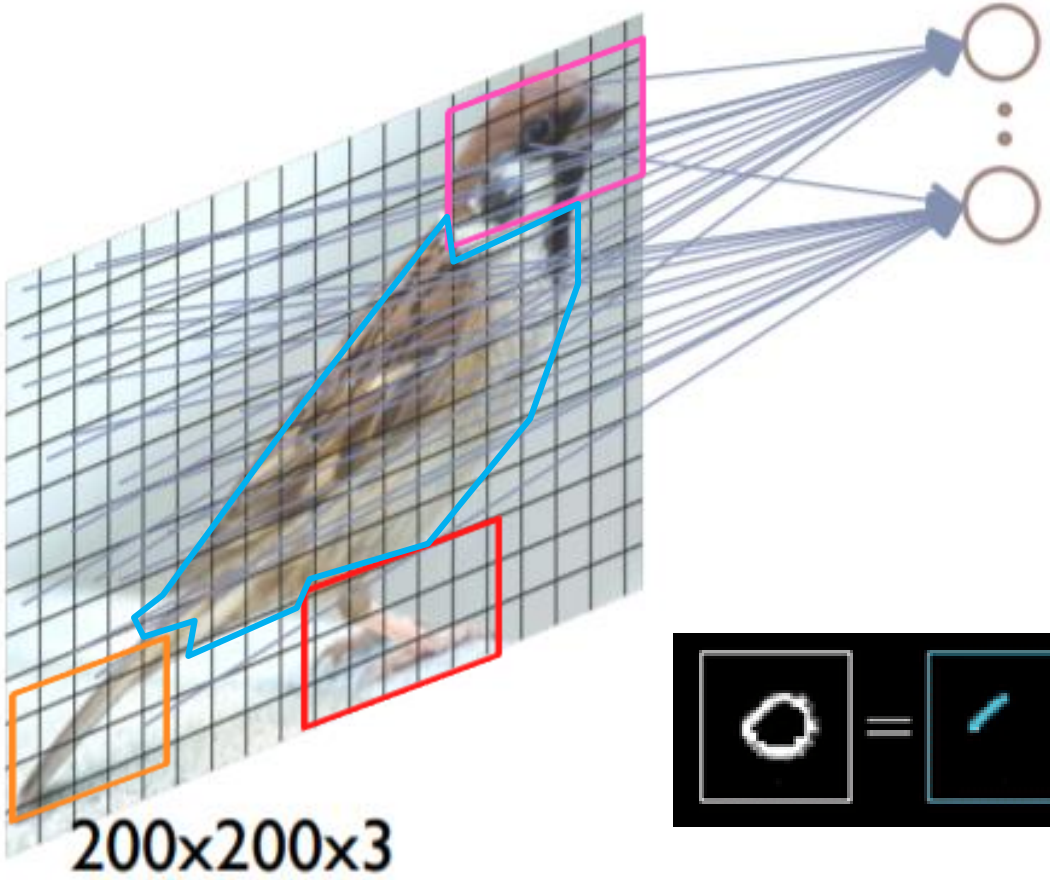
- Images are 2D.
- Assumption: Object image = **hierarchical** combination of **2D image patterns**

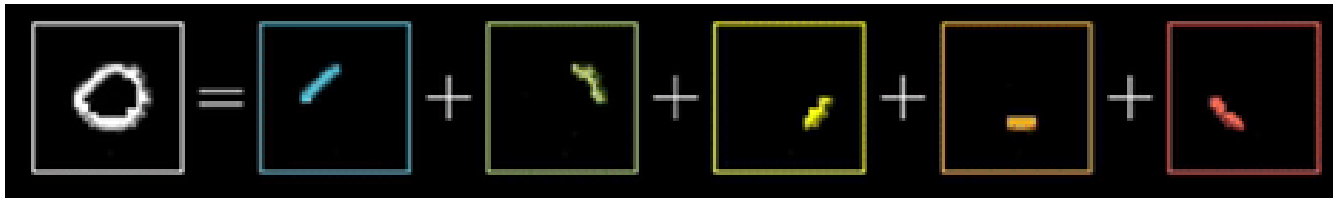
- Machine Learning Strategy:
  - [All except pre-final] Determine 2D image patterns
  - [Pre-final layer] Map 2D image patterns → Target label

- NOTE: 2D image patterns are smaller than image

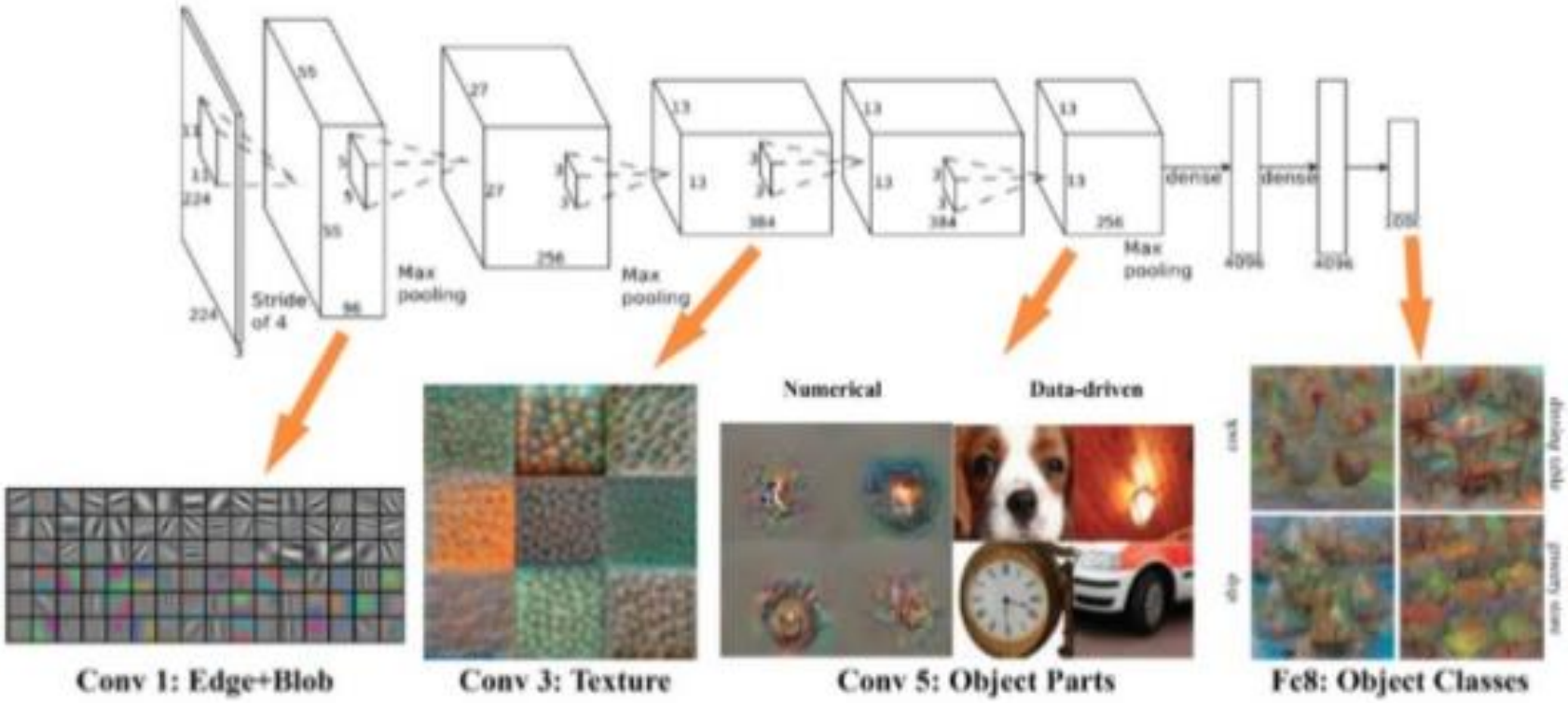



SPOILER ALERT!


200x200x3



SPOILER ALERT!

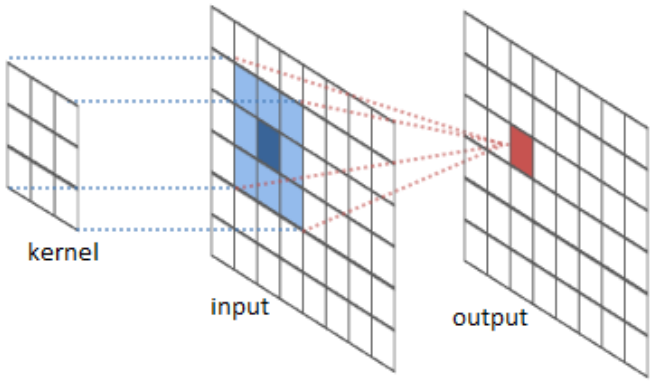


From: mNeuron: A Matlab Plugin to Visualize Neurons from Deep Models, Donglai Wei et. al.

- Pattern detectors = Filters
- What should be the size of filters ?
- How many filters do we need at each level ?
- How many levels ?

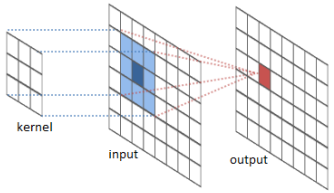
How do filters work ?

- Determine the filter
- ‘Scan’ the filter on input image / representation
- Output = A record of all input locations that ‘match’ the filter



$$\begin{pmatrix} 0 & 1 & 2 \\ 2 & 2 & 0 \\ 0 & 1 & 2 \end{pmatrix}$$

How do filters work ?



0	0	0
0	-1	1
0	0	0

Input

Kernel Convolution

-1	-1	-1
-1	8	-1
-1	-1	-1

Feature Map

Features

Wouldn't it be nice if we could feed the image directly ?

… and let the “learning process” figure out which features to extract ?

Wouldn't it be nice if we could feed the image directly ?

… and let the "learning process" figure out which features to extract ?

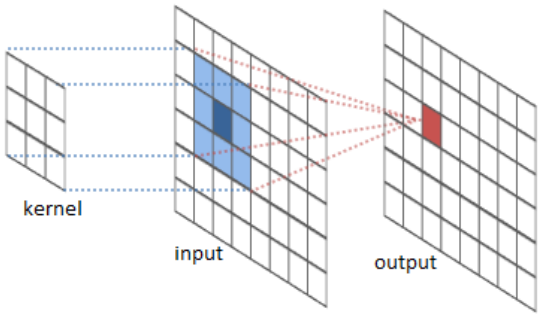
(which filters to construct)

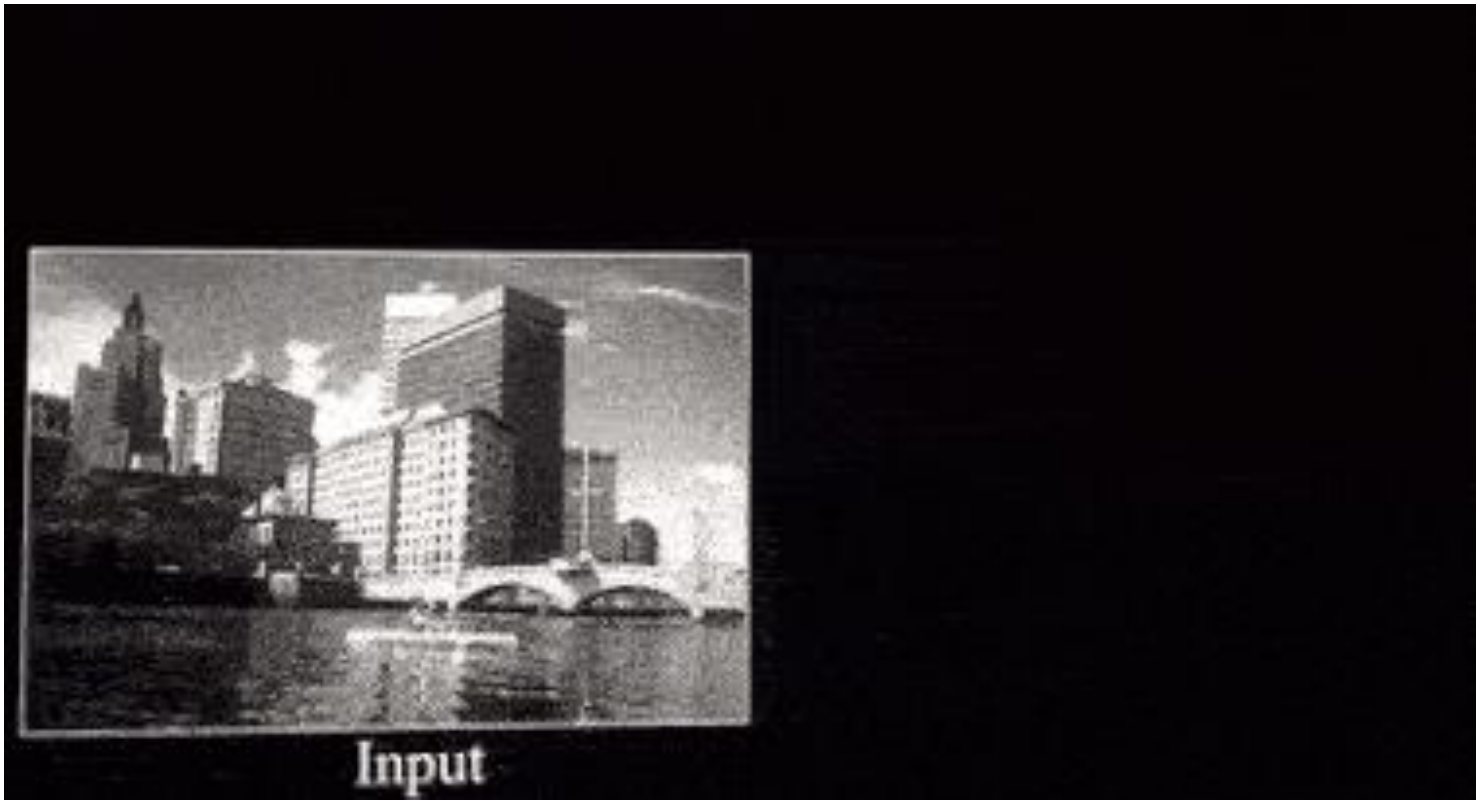
Convolution (in general)

3_0	3_1	2_2	1	0
0_2	0_2	1_0	3	1
3_0	1_1	2_2	2	3
2	0	0	2	2
2	0	0	0	1

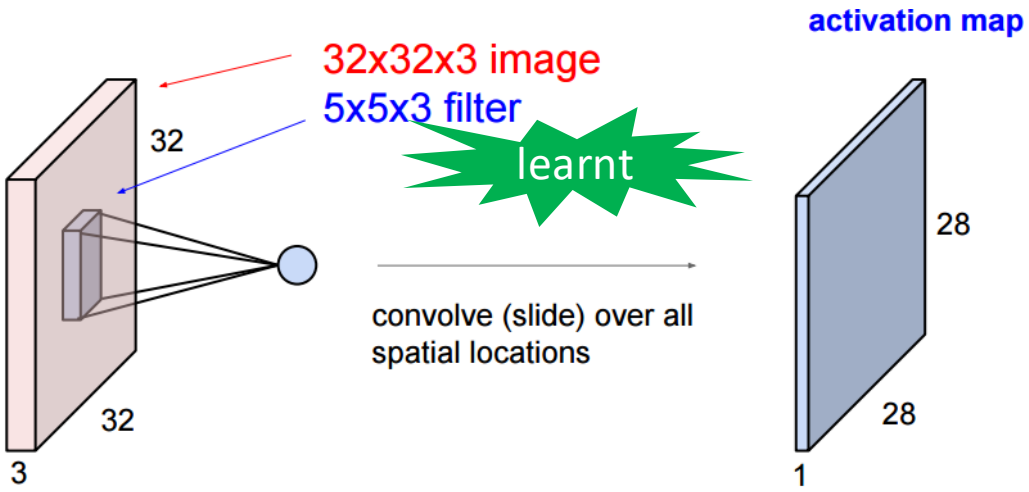
12	12	17
10	17	19
9	6	14

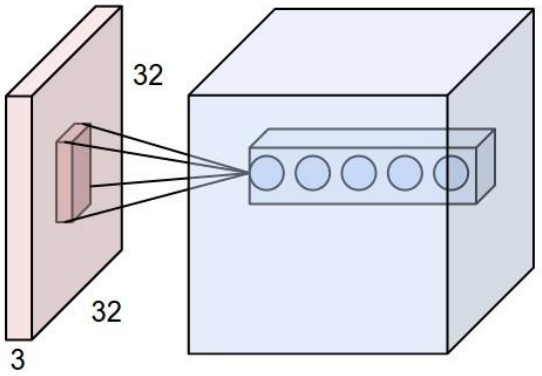
Multiple filters, multiple ‘feature maps’



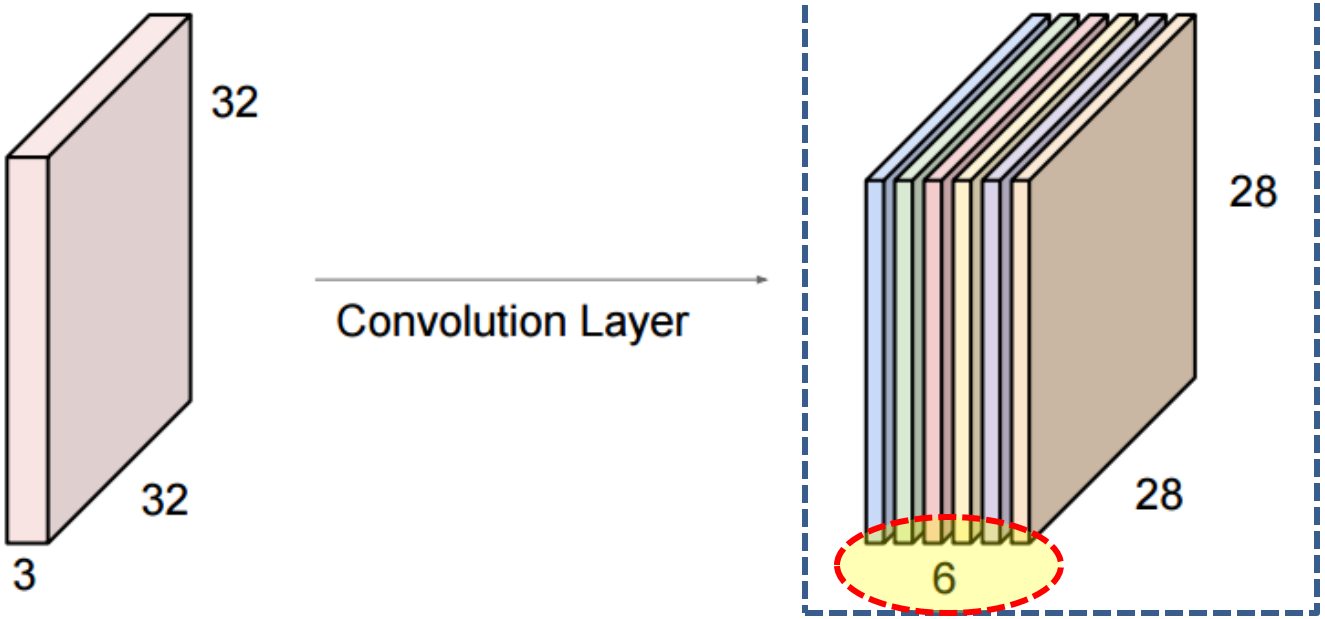


Convolution Layer





Input to next layer



Design choices : Filter size, # of filters

3x3

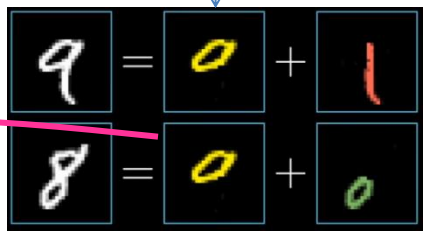
1	2	1
2	4	2
1	2	1

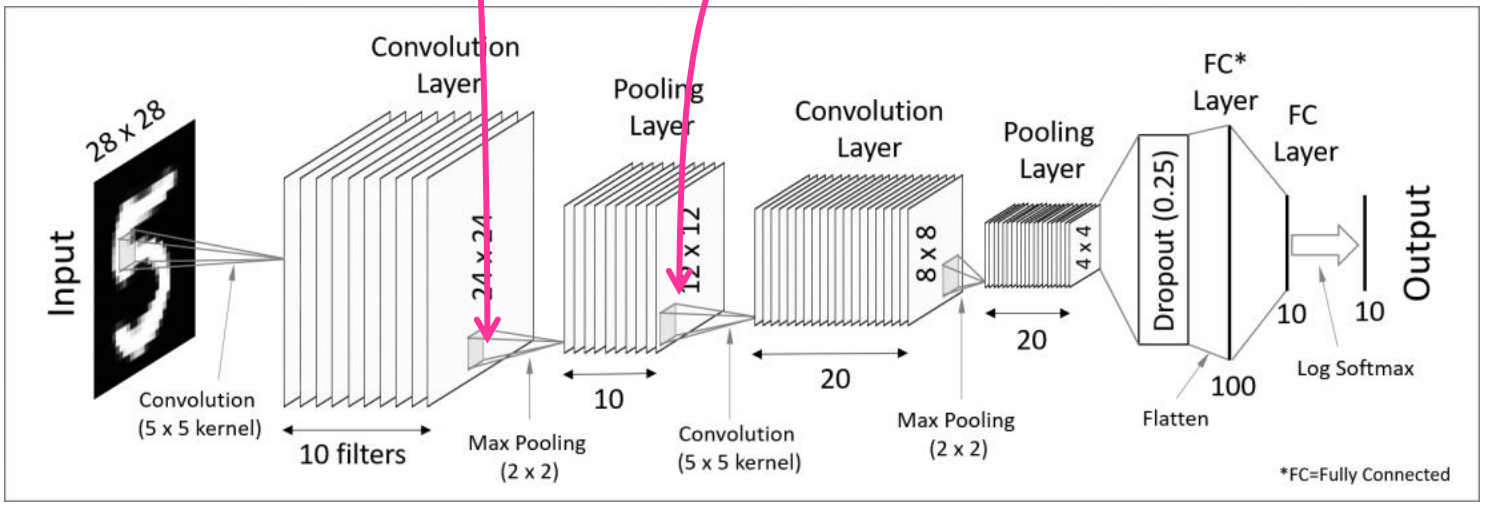
5x5

2	7	12	7	2
7	31	52	31	7
12	52	127	52	12
7	31	52	31	7
2	7	12	7	2

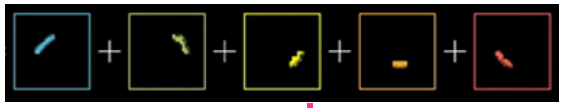
7x7

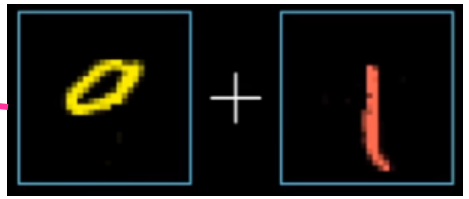
1	1	2	2	2	1	1
1	3	4	5	4	3	1
2	4	7	8	7	4	2
2	5	8	10	8	5	2
2	4	7	8	7	4	2
1	3	4	5	4	3	1
1	1	2	2	2	1	1

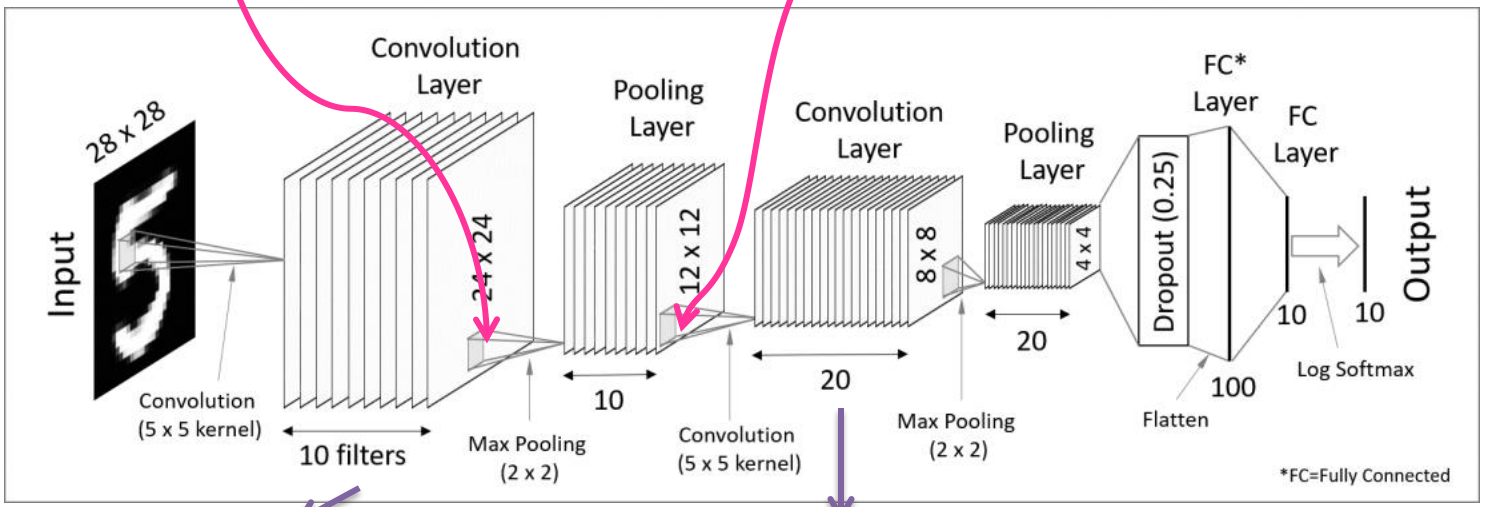




Design choices : Filter size, # of filters



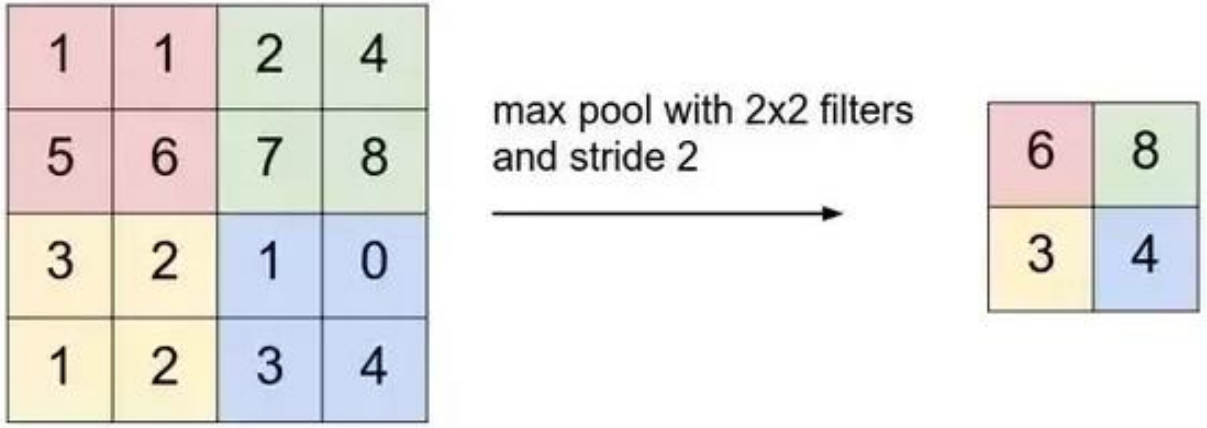




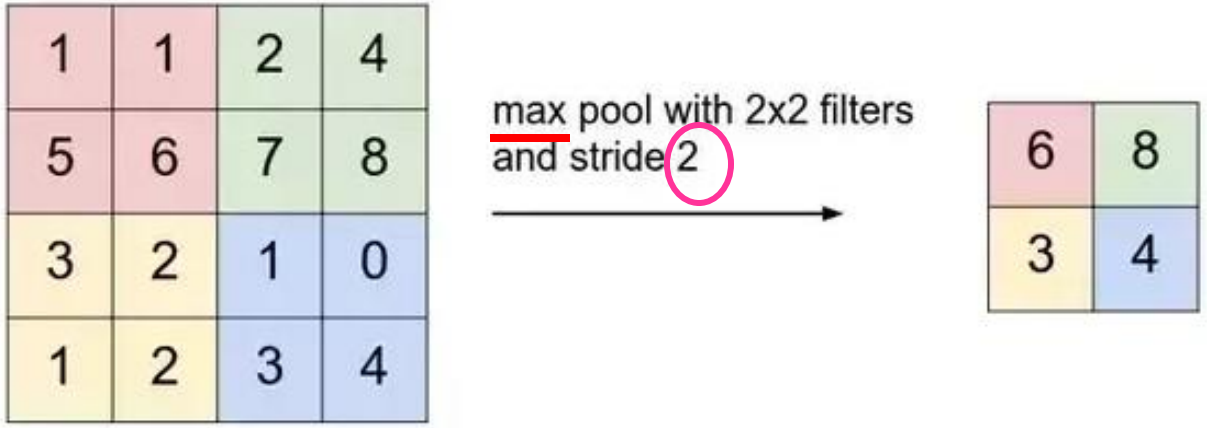
~ Alphabets

~ Words

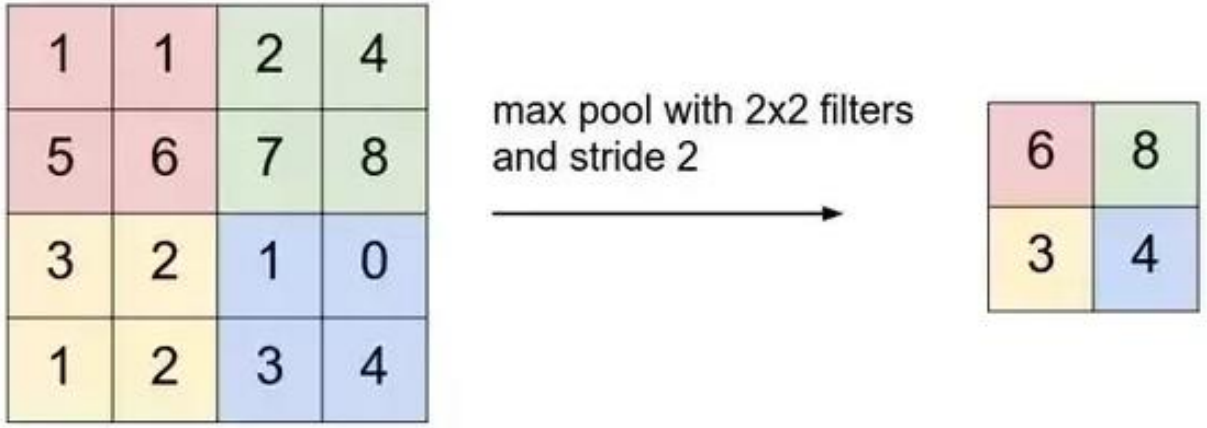
Pooling Layer



Pooling Layer

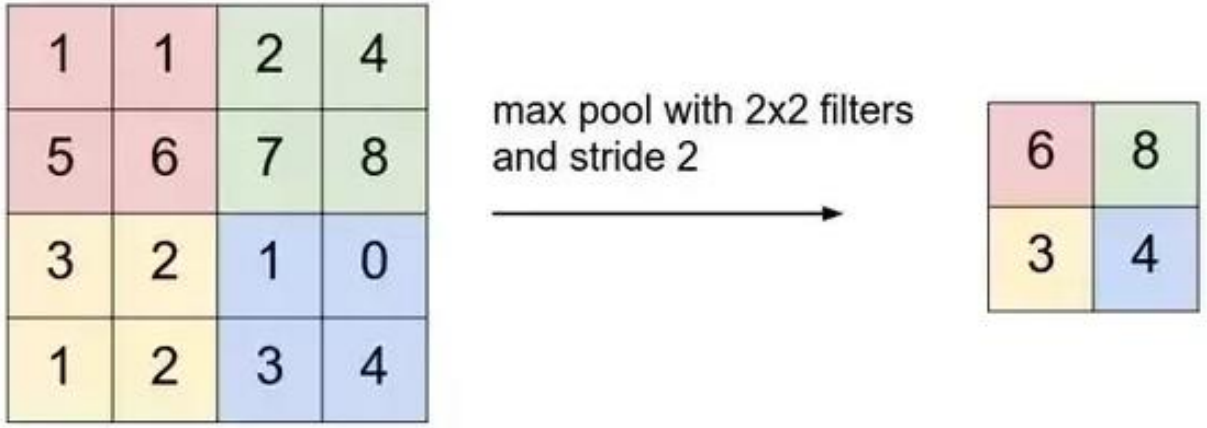


Pooling Layer

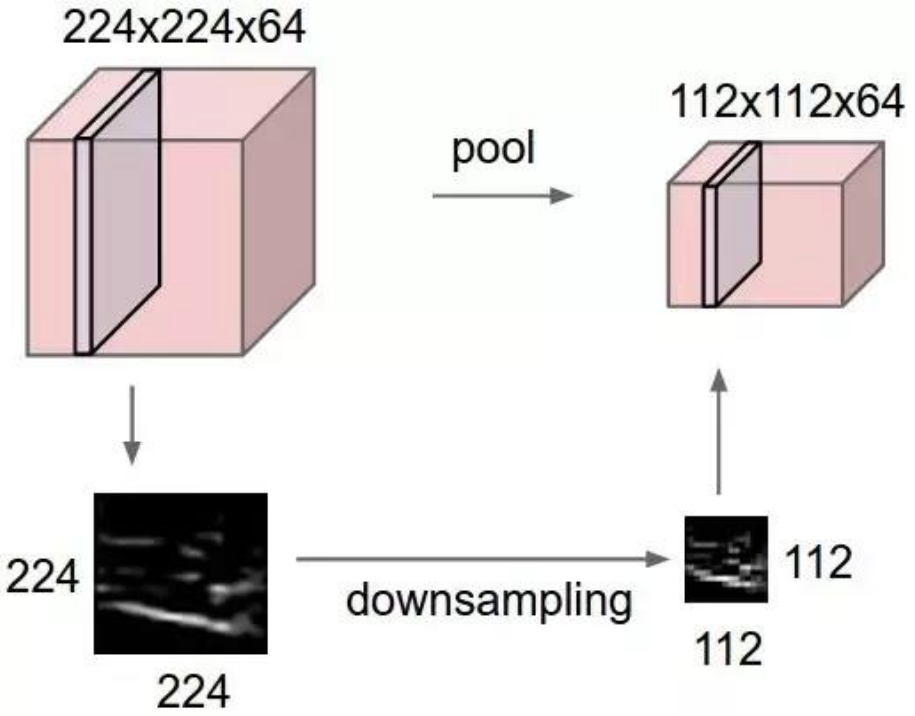


- **Motivation: We care about presence of features, not their exact location !**
- Dimensionality Reduction
- Prevents overfitting

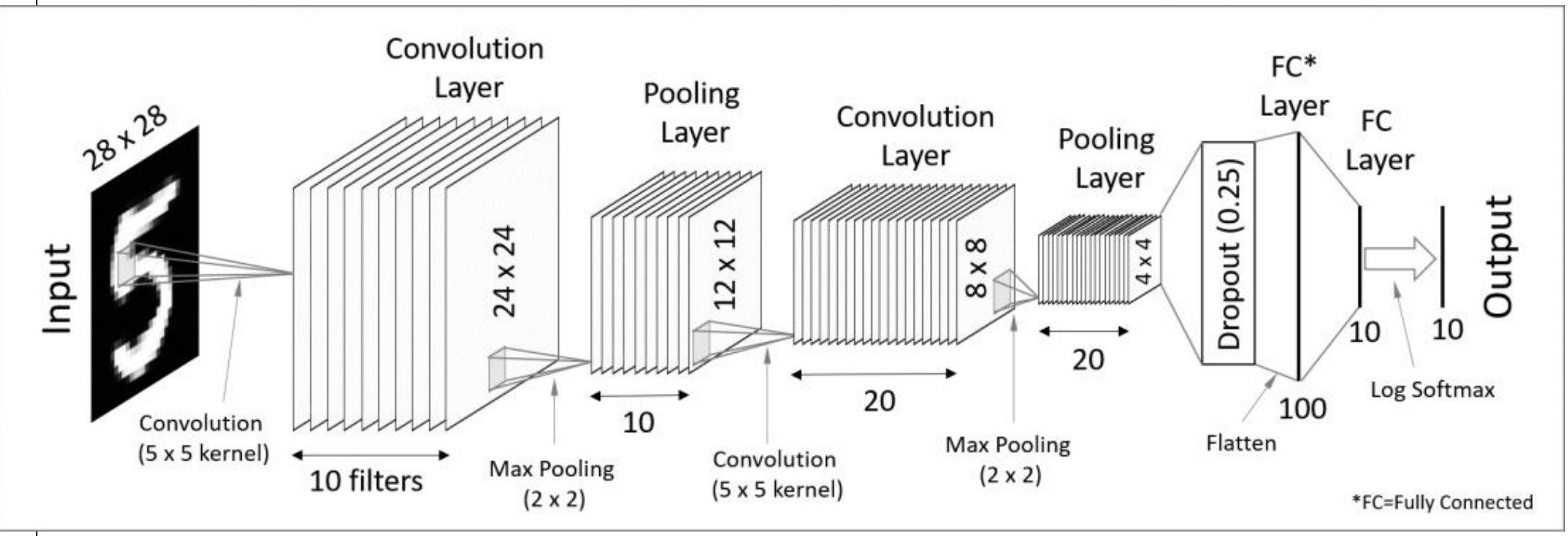
Pooling Layer



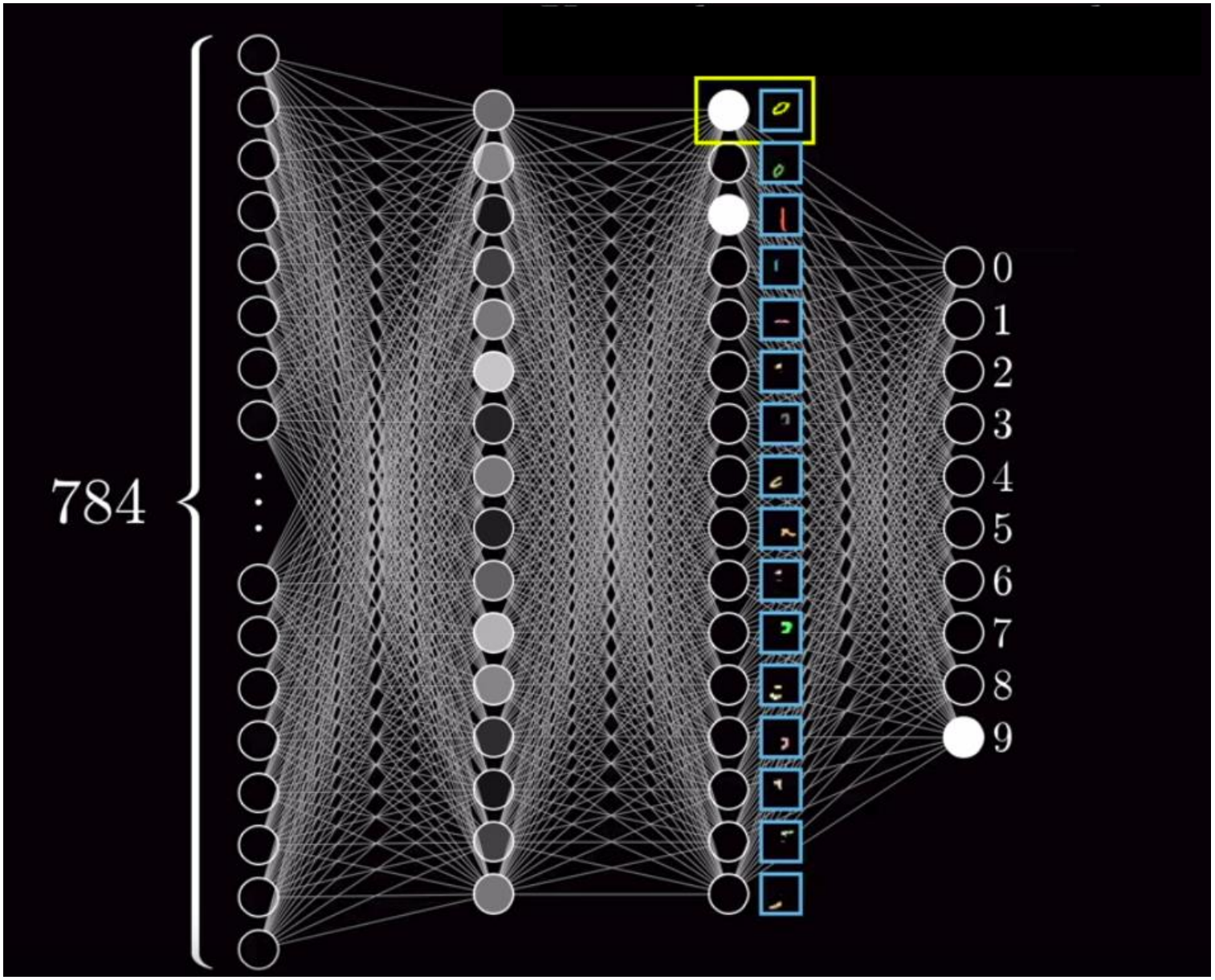
- **Motivation: We care about presence of features, not their exact location !**
- Dimensionality Reduction
- Prevents overfitting

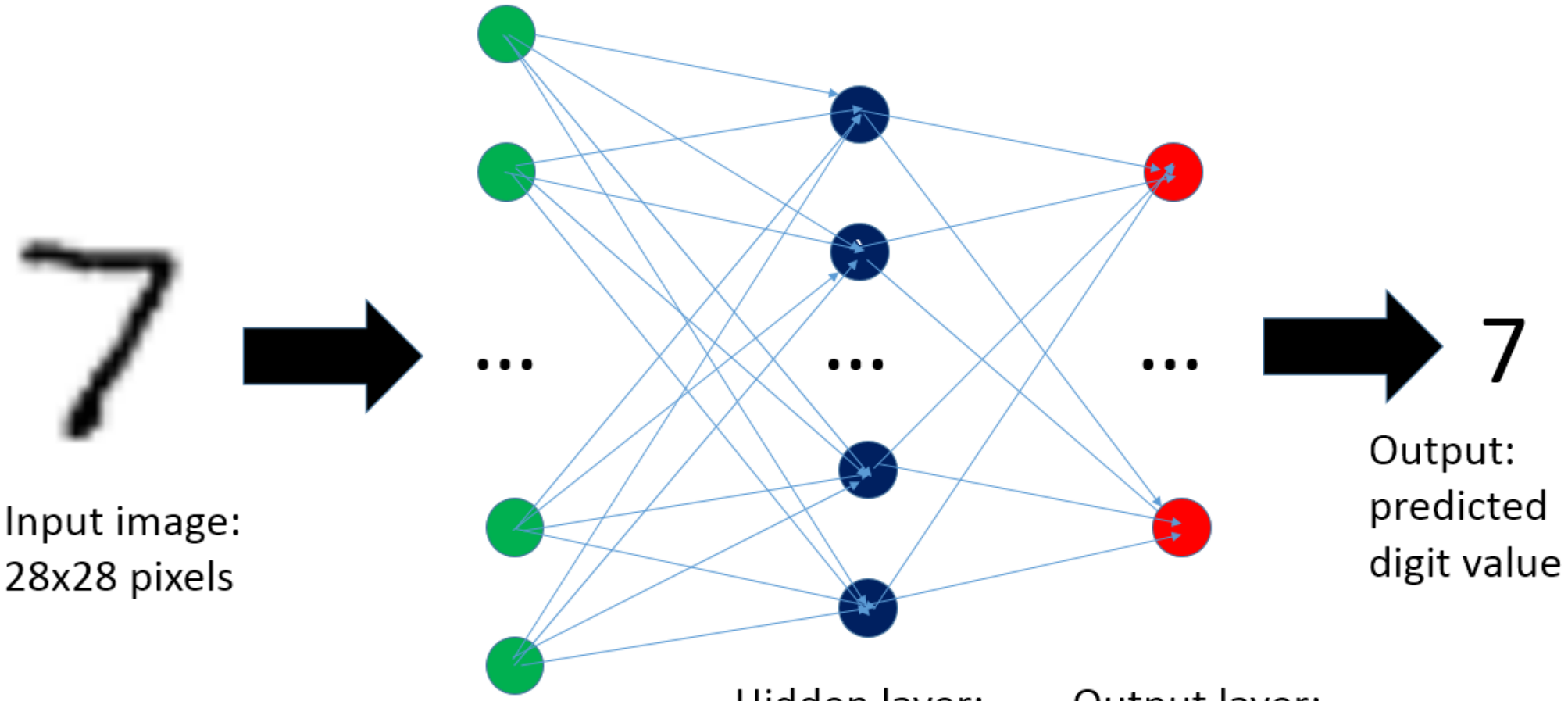


Fully-connected layers

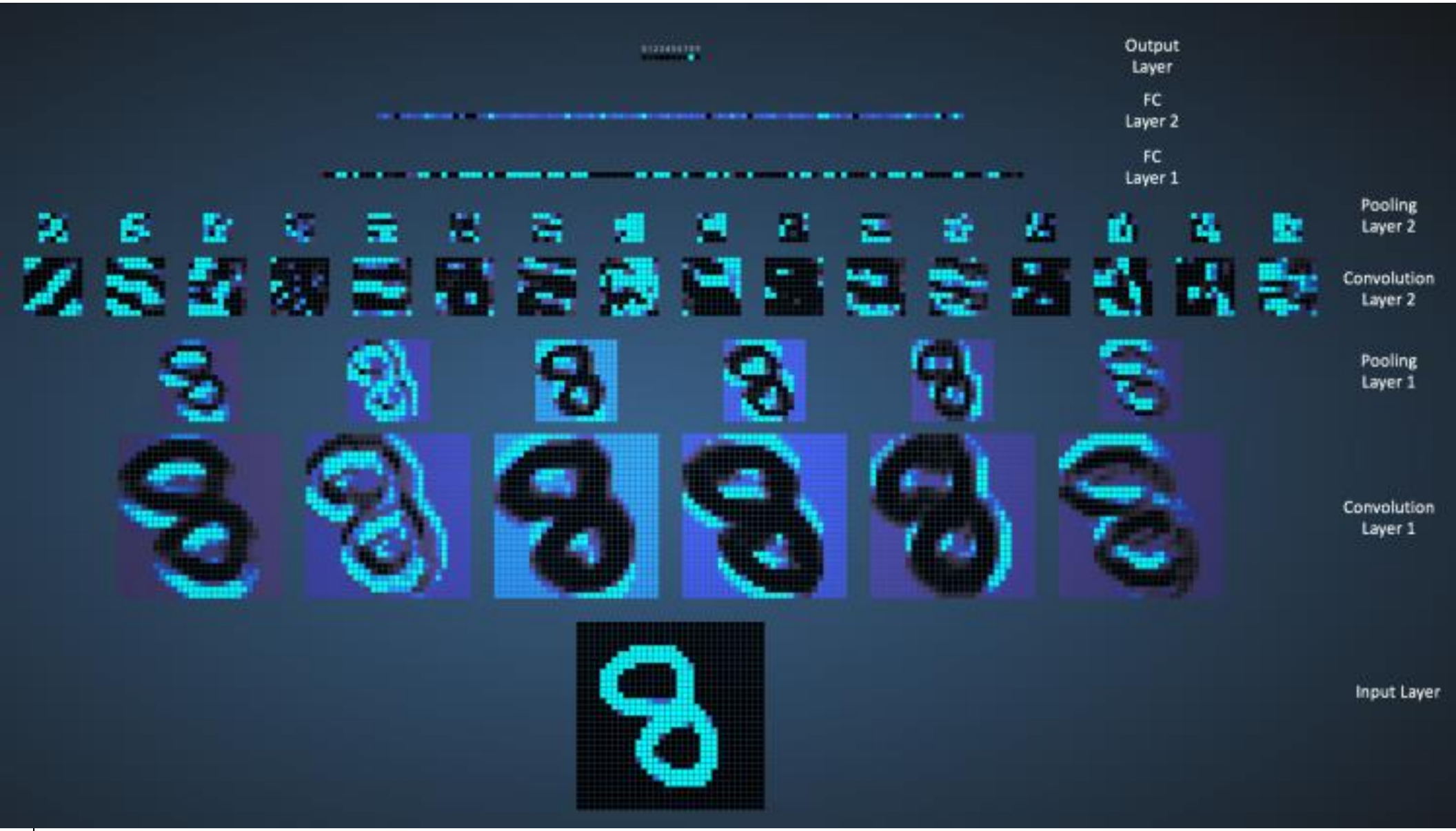


Fully Connected Layers



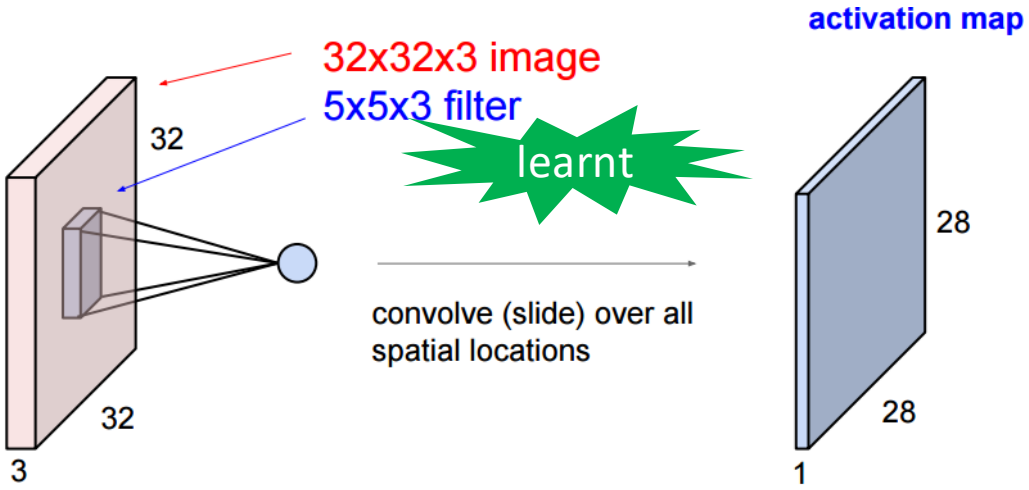

7
7
Input image:
28x28 pixels
Input layer:
784 neurons,
one per pixel
Hidden layer:
100 neurons
Output layer:
10 neurons
Output:
predicted
digit value

MNIST features visualized

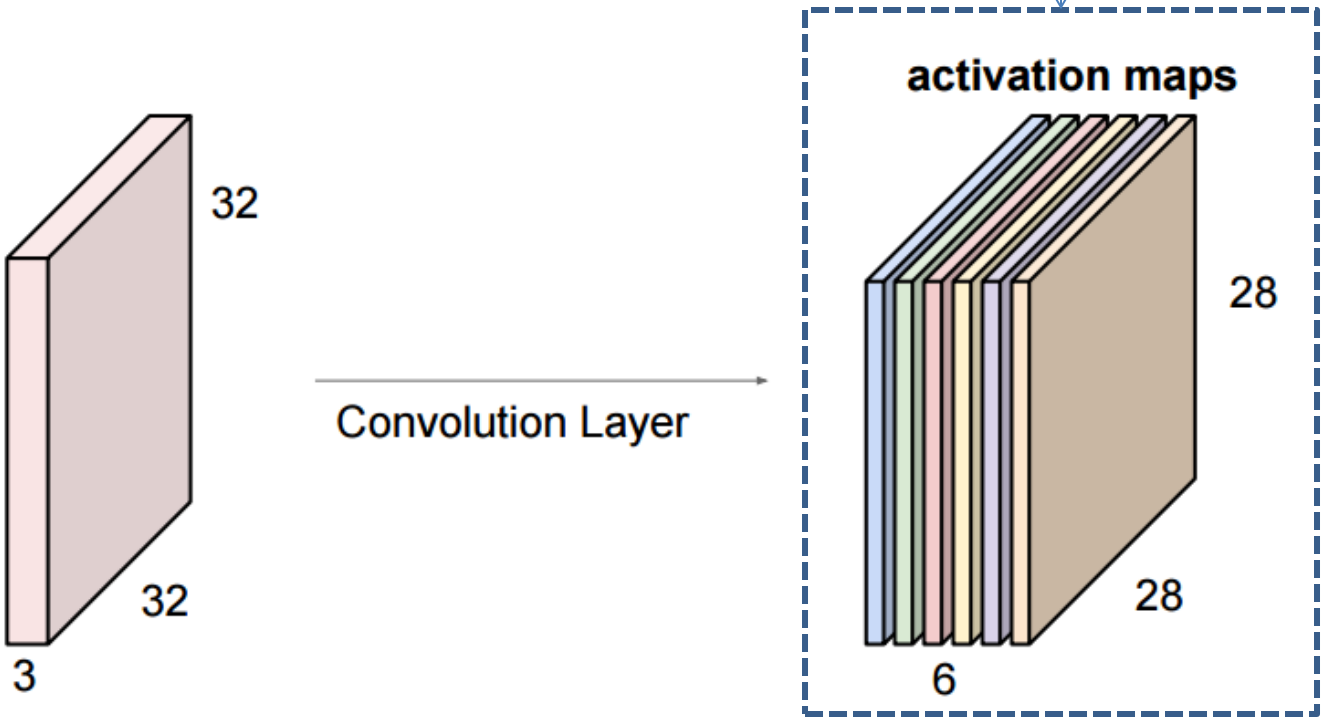


http://blog.arimaresearch.com/wp-content/uploads/2017/01/conv_all.png

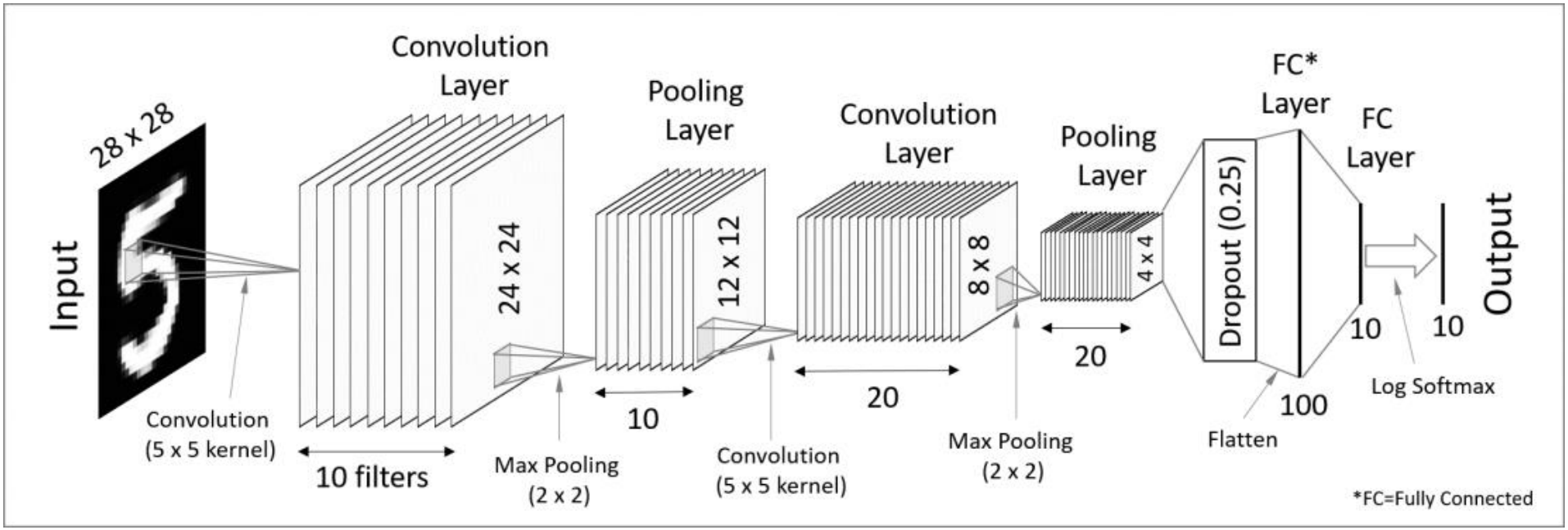
Convolution Layer



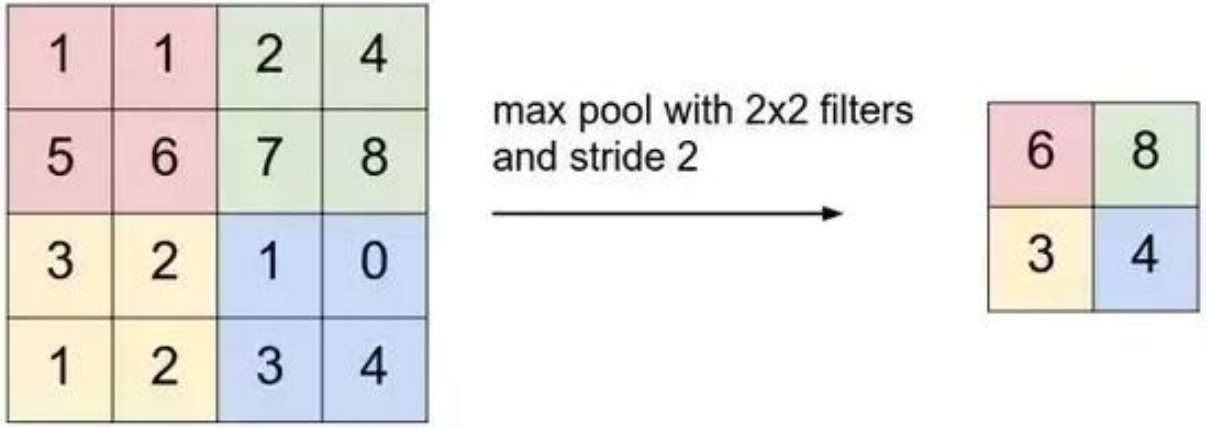
Input to next layer



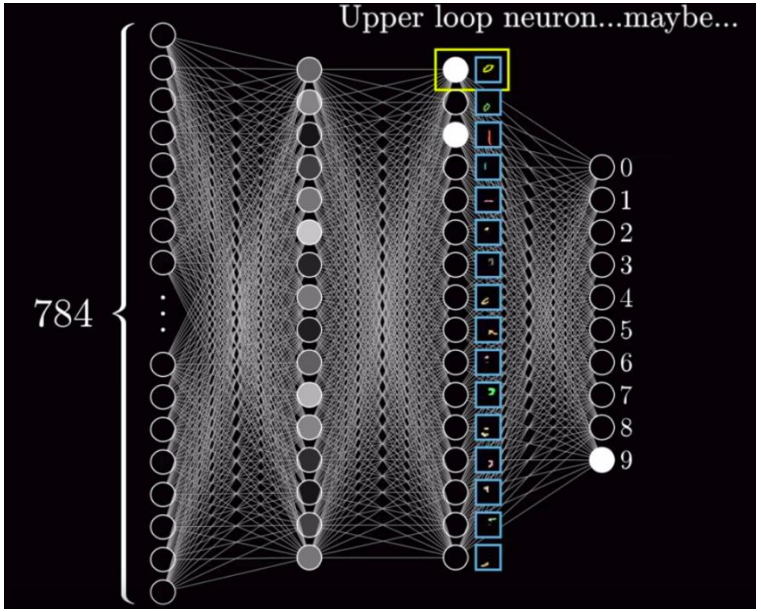
Convolutional Neural Network



Pooling Layer



Fully-connected Layer

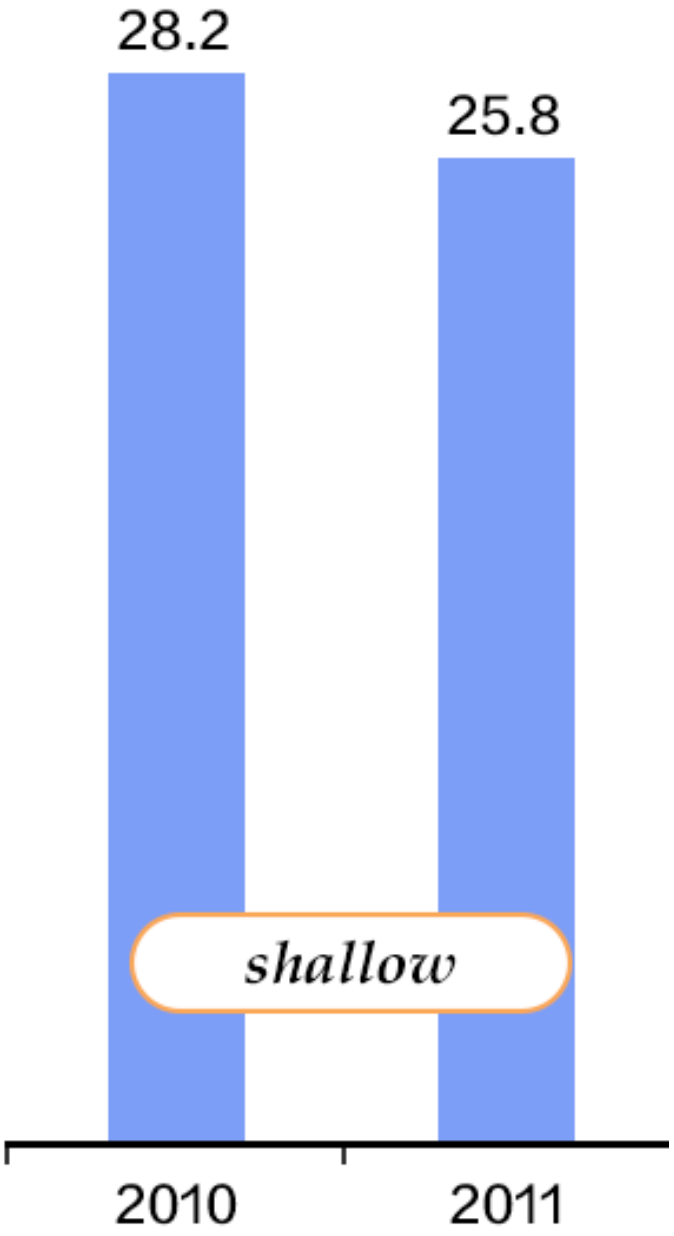


ImageNet Challenge

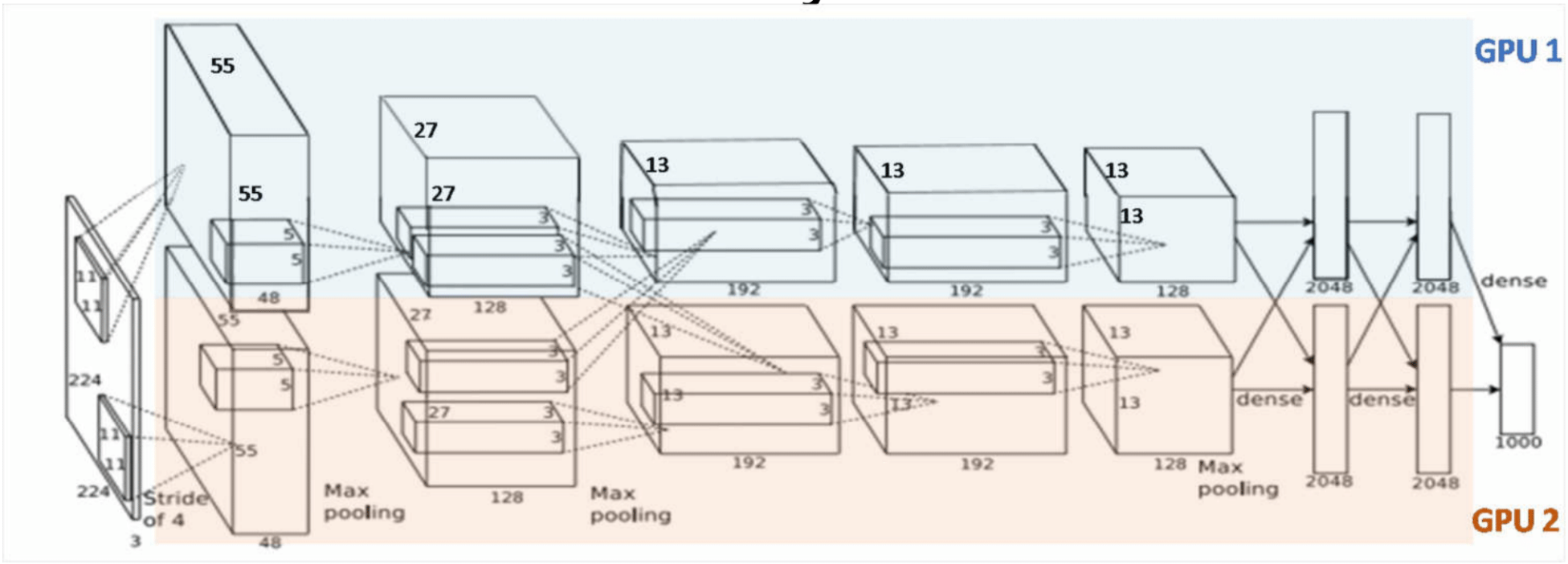
IMAGENET

- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.

Top-5 misclassification error rate



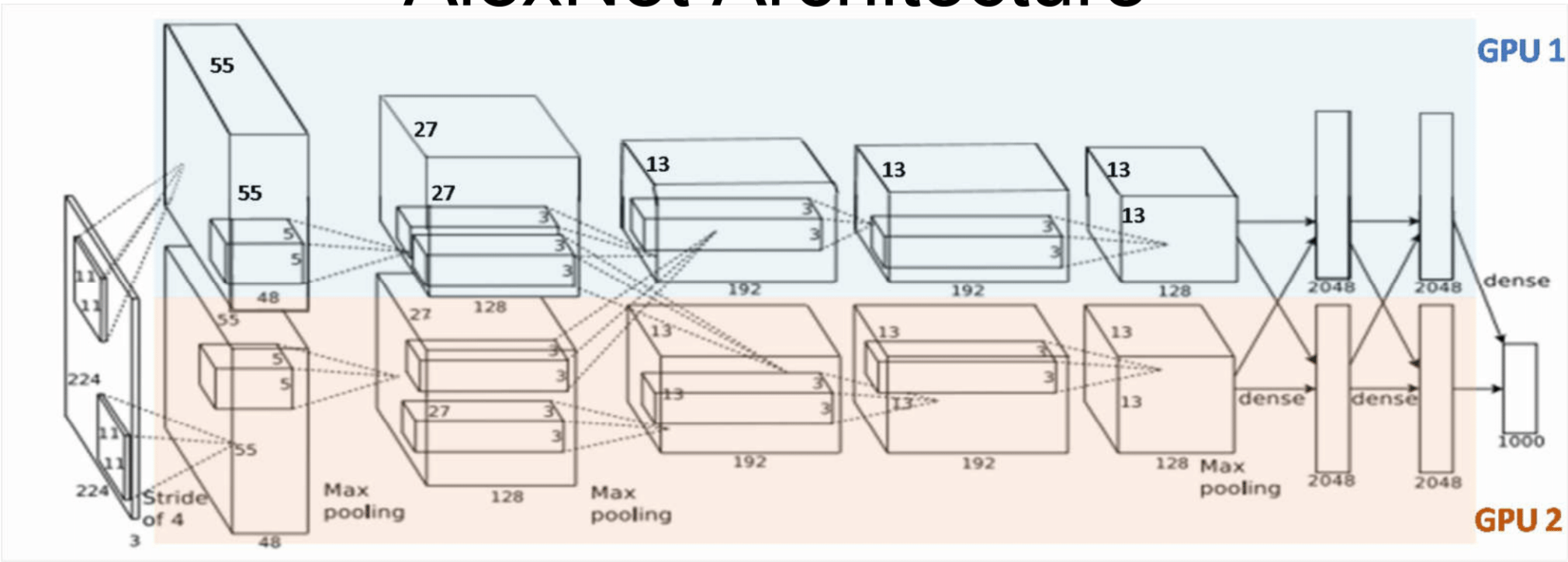
Case Study: AlexNet

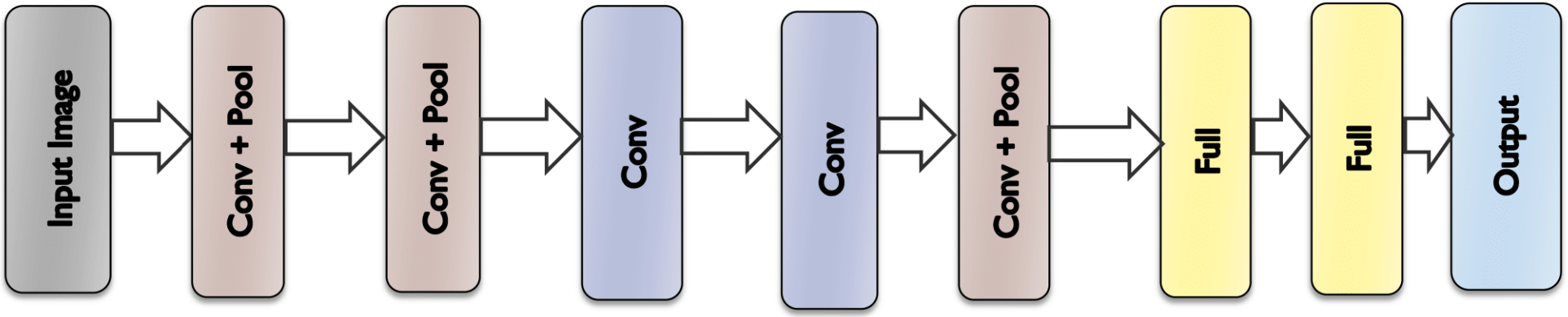


- Winner of ImageNet LSVRC-2010.
- Trained over 1.2M images using SGD with regularization.
- Deep architecture (60M parameters.)
- Optimized GPU implementation (cuda-convnet)

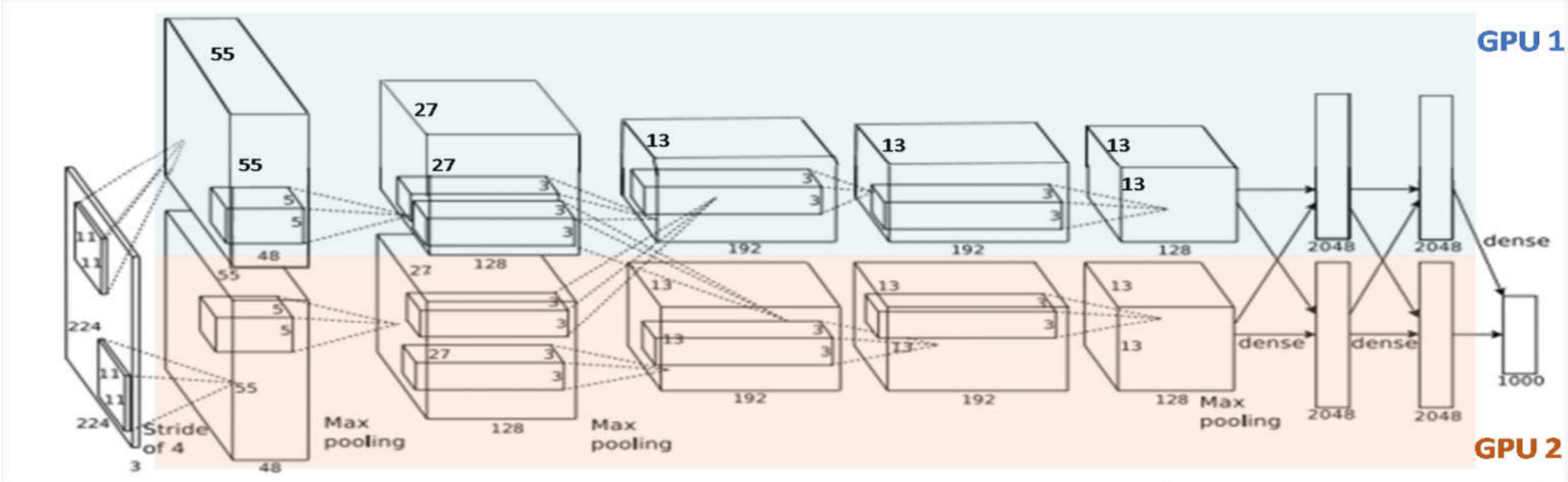
Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *NIPS* 2012.

AlexNet Architecture





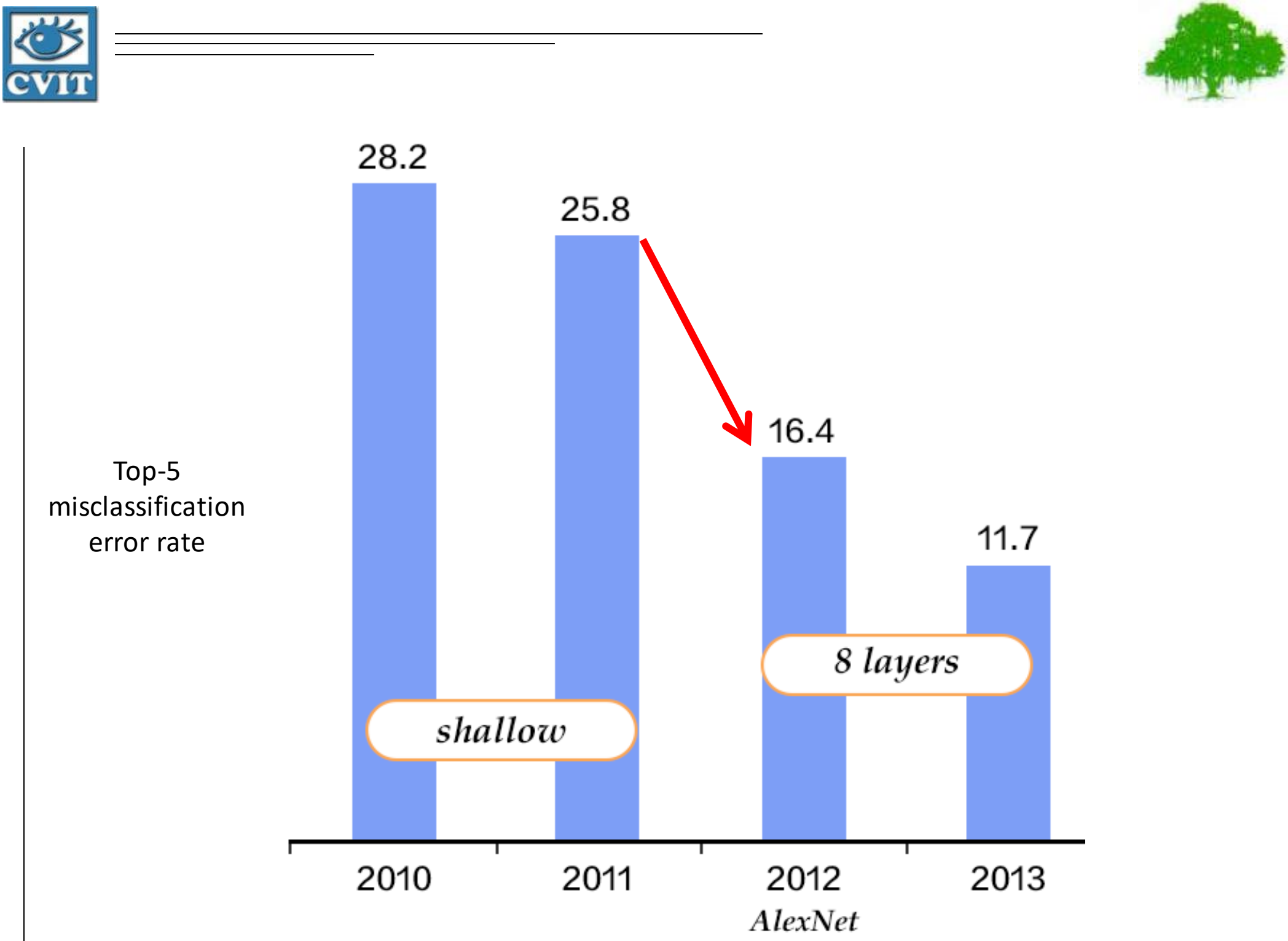
AlexNet Architecture



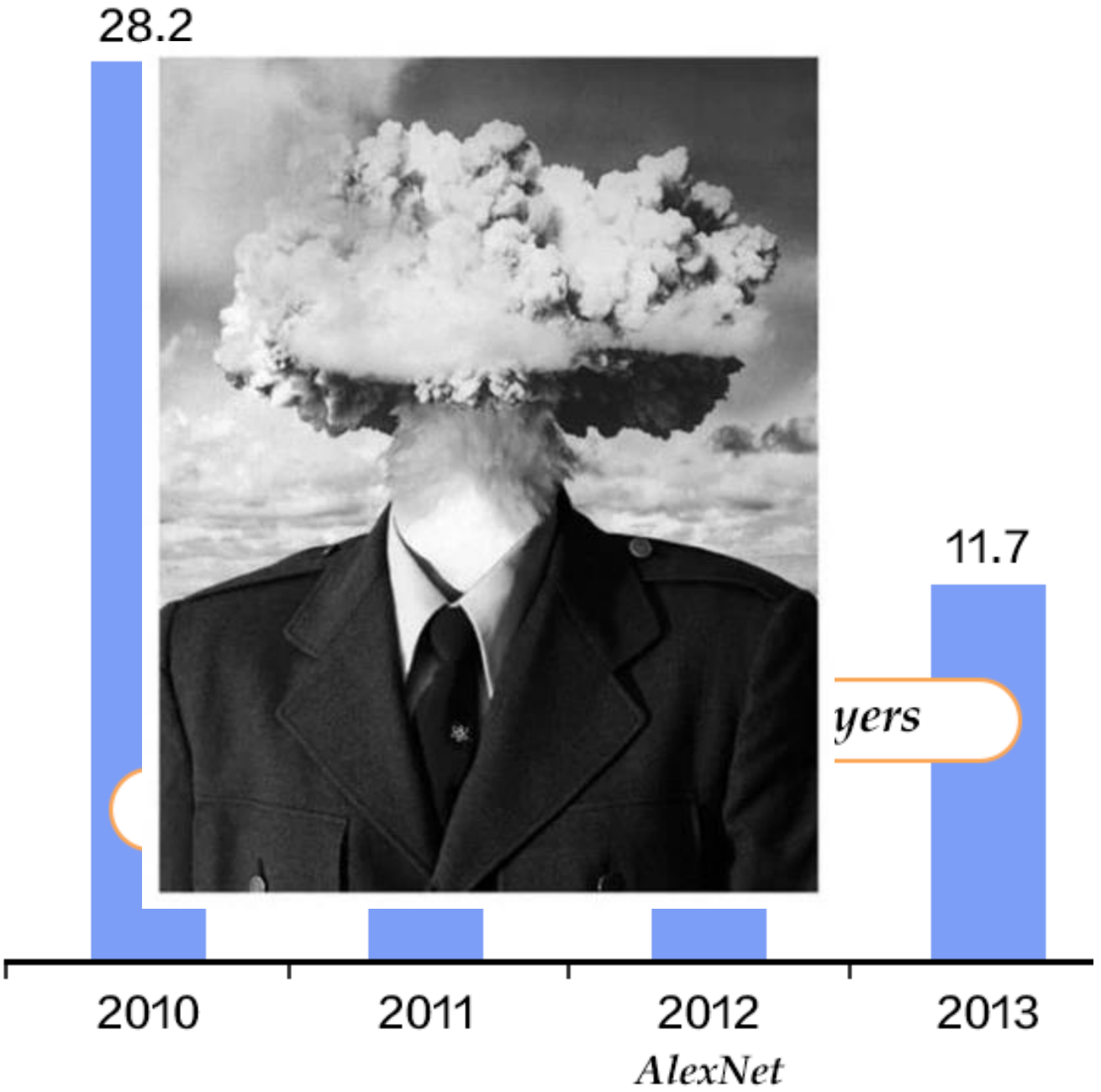
About 57 M parameters are in the fully connected layers

									Total
Parameters :	[(11 x 11 x 3) + 1] x 96 = 35 K	[5 x 5 x 48] x 256 = 307 K	[3 x 3 x 256] x 384 = 884 K	663 K	442 K	37 M	16 M	4 M	60 M
Neurons :	253,440	27 x 27 x 256 = 186,624	13 x 13 x 384 = 64,896	13 x 13 x 384 = 64,896	13 x 13 x 256 = 43,264	4096	4096	1000	0.63 M

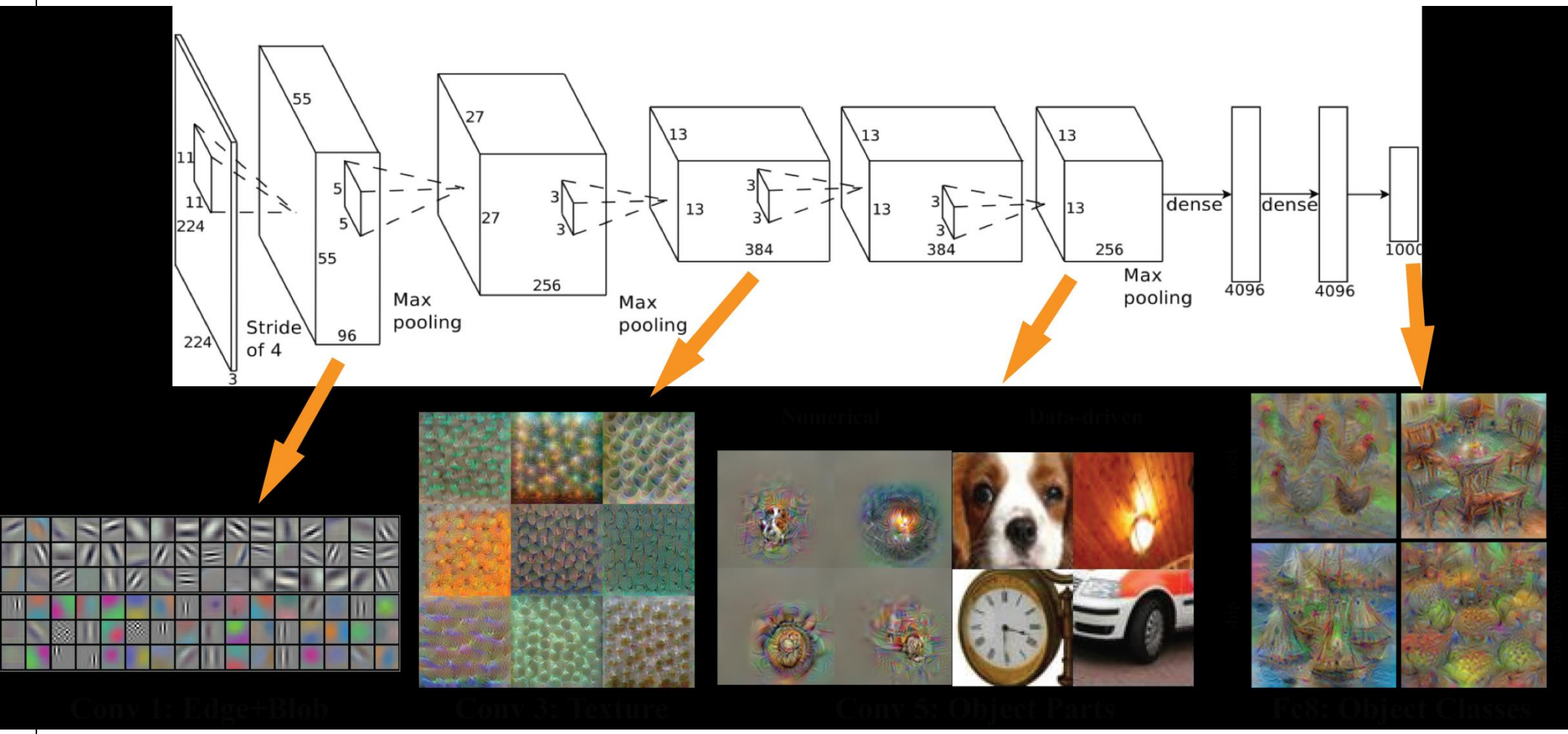
- Convolutional layers cumulatively contain about 90-95% of computation, only about 5% of the parameters
- Fully-connected layers contain about 95% of parameters.


Top-5 misclassification error rate
28.2
25.8
16.4
11.7
shallow
8 layers
2010
2011
2012
2013
AlexNet

Top-5 misclassification error rate



Filters learnt by AlexNet



https://www.cc.gatech.edu/~hays/compvision/proj6/deepNetVis.png

Committing to a single filter size at each layer seems somewhat rigid !

3x3

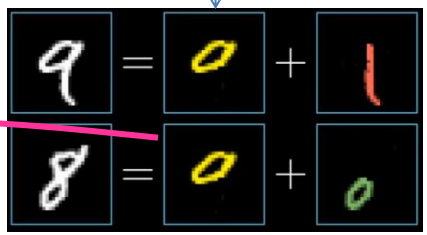
1	2	1
2	4	2
1	2	1

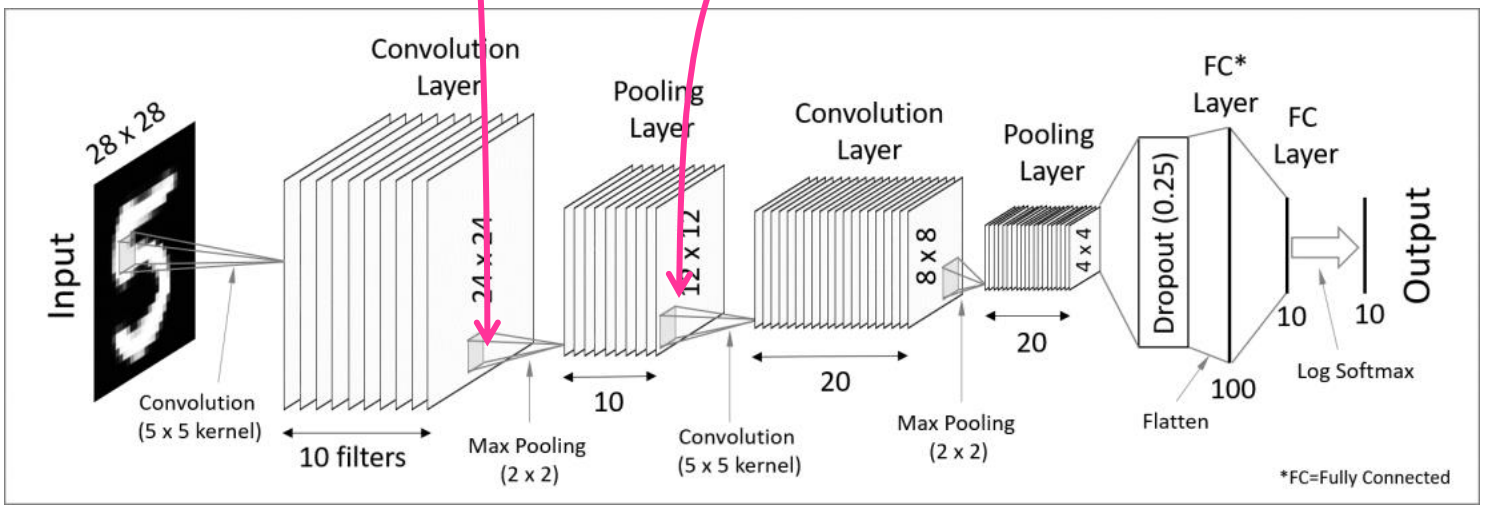
5x5

2	7	12	7	2
7	31	52	31	7
12	52	127	52	12
7	31	52	31	7
2	7	12	7	2

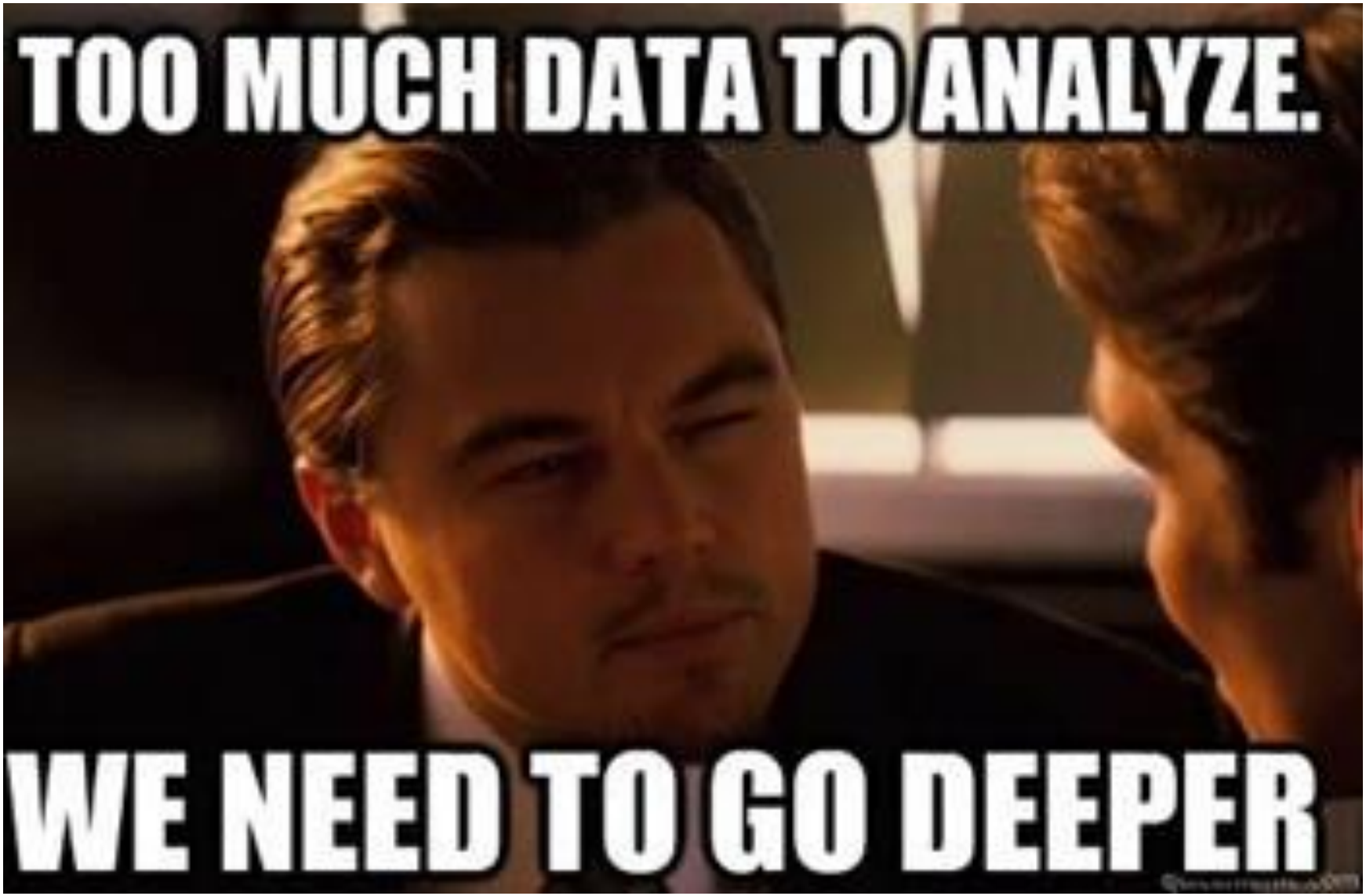
7x7

1	1	2	2	2	1	1
1	3	4	5	4	3	1
2	4	7	8	7	4	2
2	5	8	10	8	5	2
2	4	7	8	7	4	2
1	3	4	5	4	3	1
1	1	2	2	2	1	1

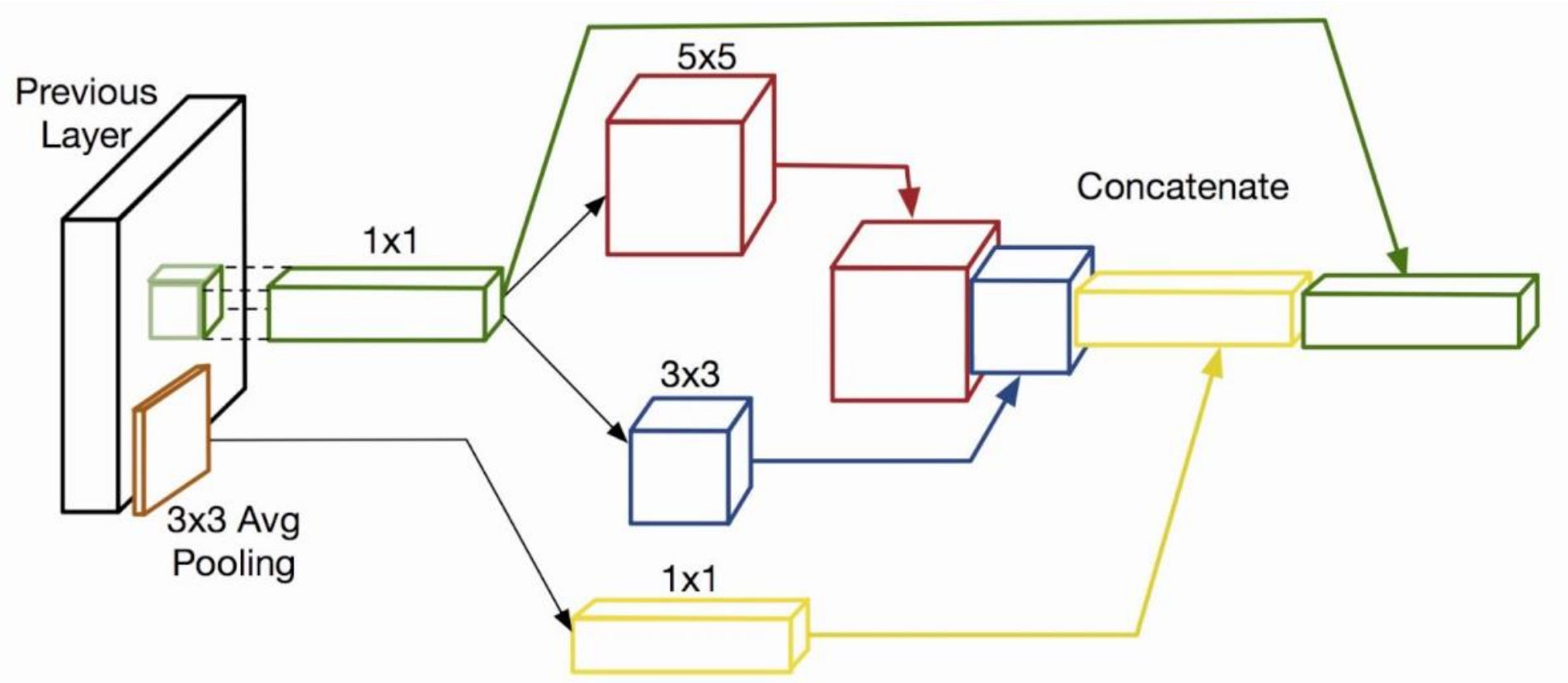




Why not have multiple levels (scales) of filtering ?



Why not have multiple levels (scales) of filtering ?

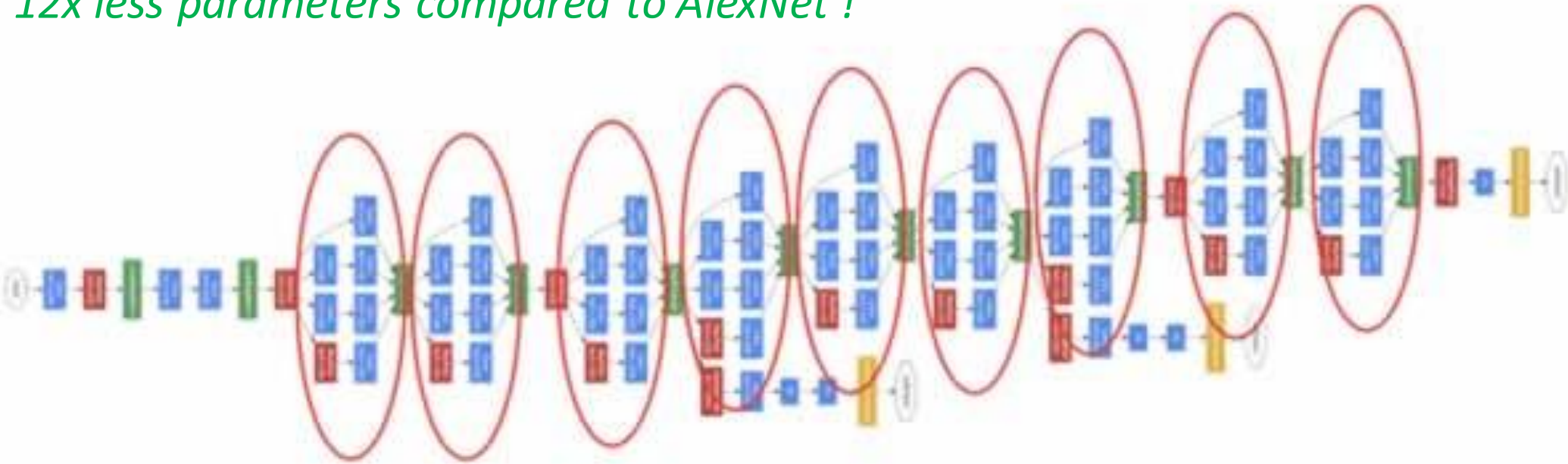


Inception Module

https://i.ytimg.com/vi/_XF7N6rp9Jw/maxresdefault.jpg

GoogLeNet

12x less parameters compared to AlexNet !

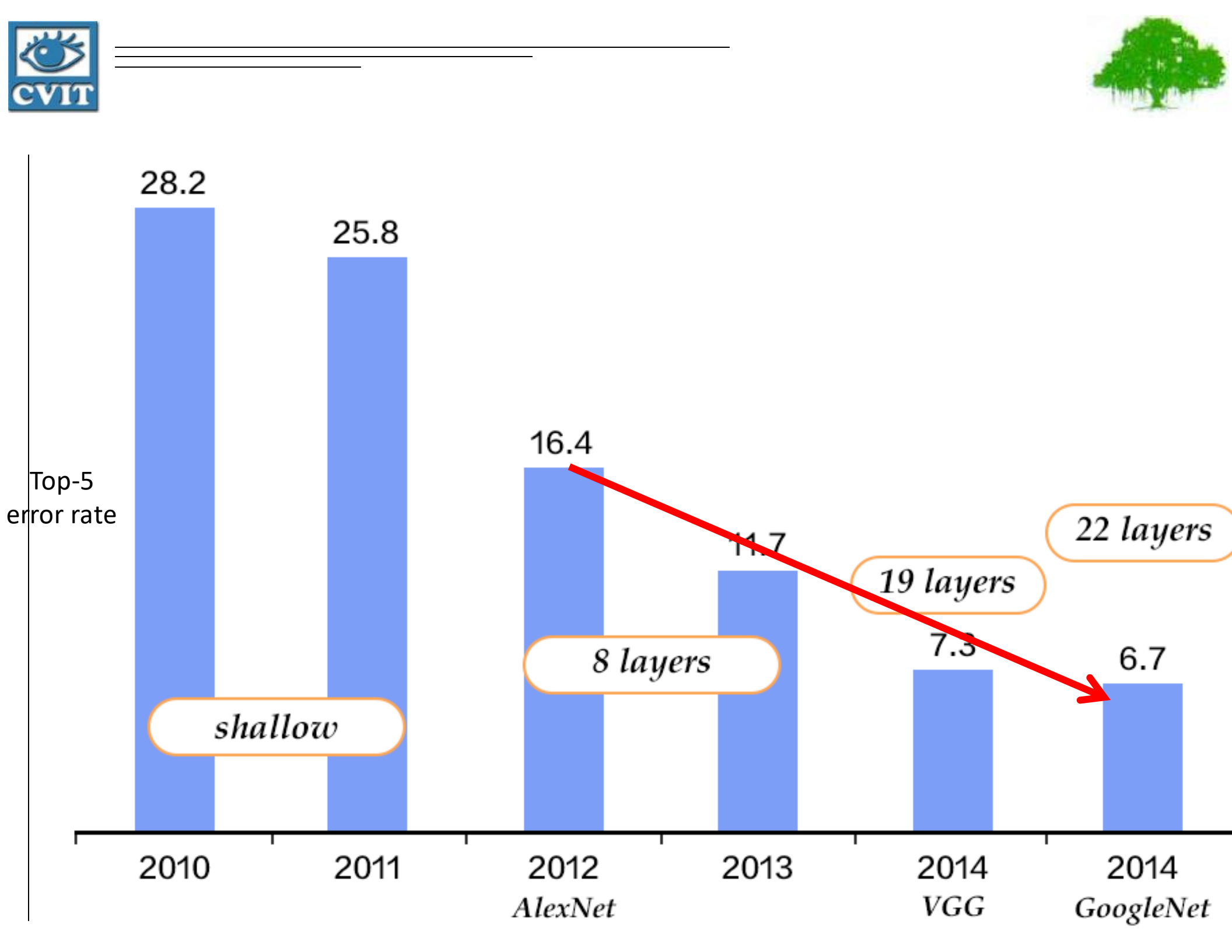


9 **Inception** modules

Network in a network in a network...

Convolution
Pooling
Softmax
Other

https://www.youtube.com/watch?v=LUaRa65J_Ms


Top-5
error rate
28.2
25.8
16.4
11.7
7.3
6.7
shallow
8 layers
19 layers
22 layers
2010
2011
2012
AlexNet
2013
2014
VGG
2014
GoogleNet

How deep can the network get ?

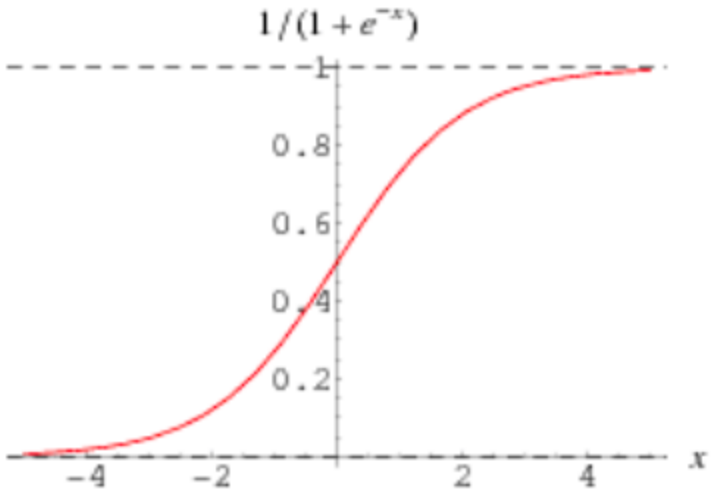
- 50 layers ? 100 ? 1000 ?

How deep can the network get ?

- 50 layers ? 100 ? 1000 ?
- Issue: Vanishing gradients

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

Sigmoid



Plot of Sigmoid function

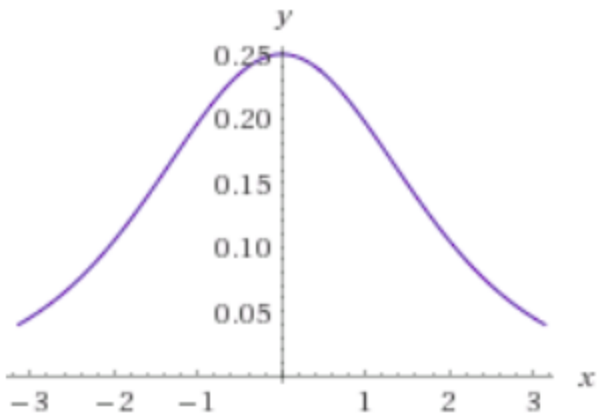
https://medium.com/@deepnarainsingh/vanishing-gradients-while-training-neural-network-42d6d5d83cdf

Vanishing gradients

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

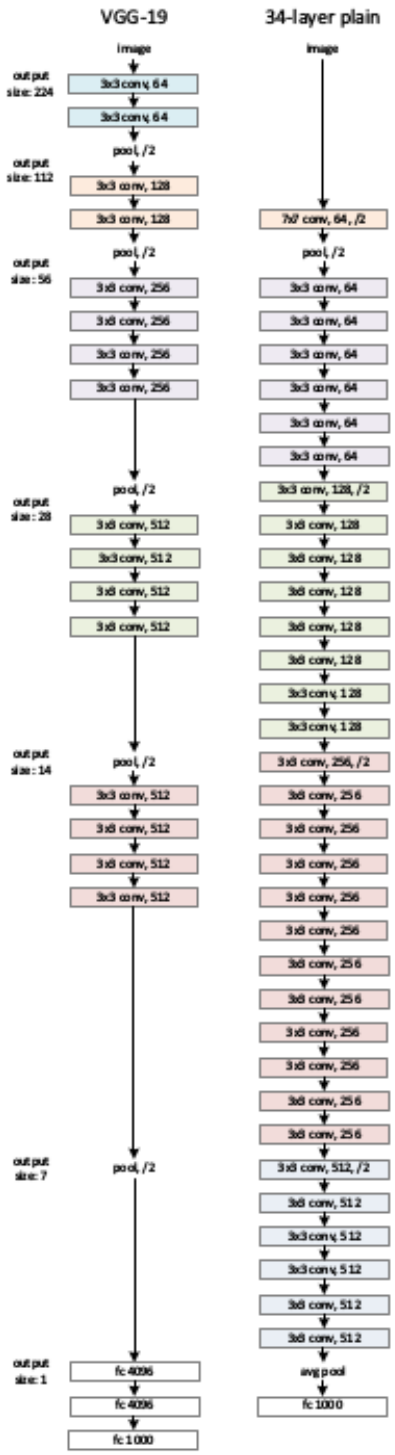
$$\sigma'(x) = \sigma(1-\sigma)$$

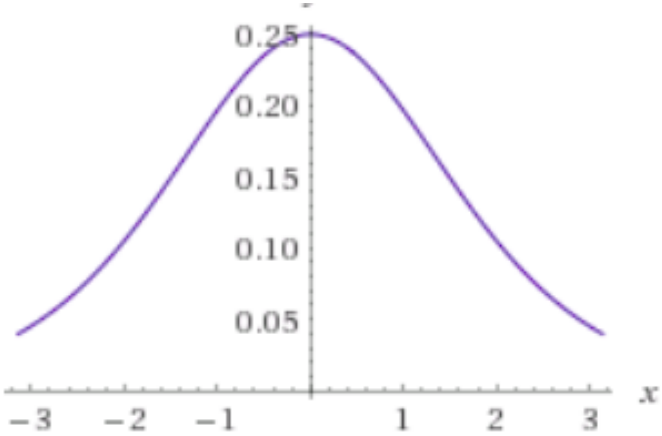
Derivative of sigmoid can be represented as



(x from −3.1 to 3.1)

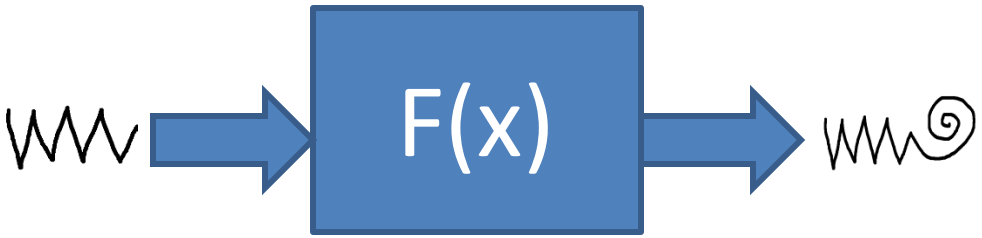
https://medium.com/@deepnarainsingh/vanishing-gradients-while-training-neural-network-42d6d5d83cdf

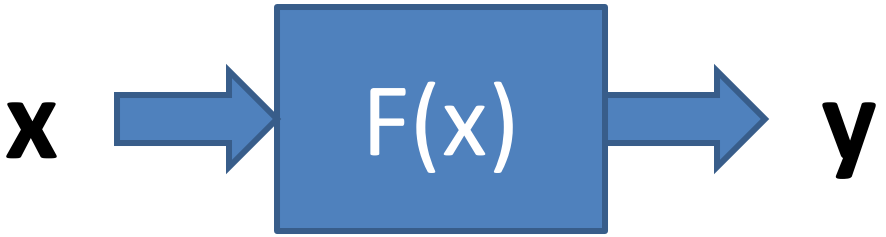




https://medium.com/@14prakash/understanding-and-implementing-architectures-of-resnet-and-resnext-for-state-of-the-art-image-cf51669e1624

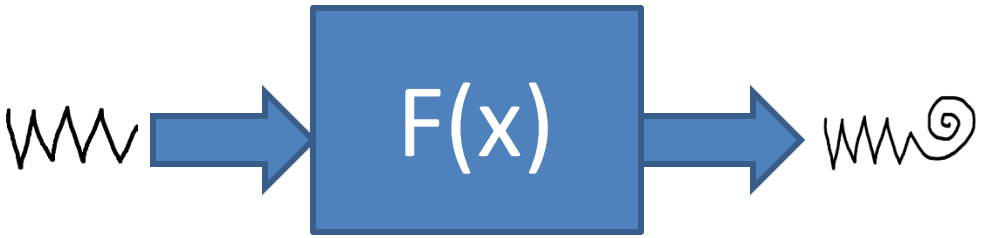
Residual Networks – A crayon analogy

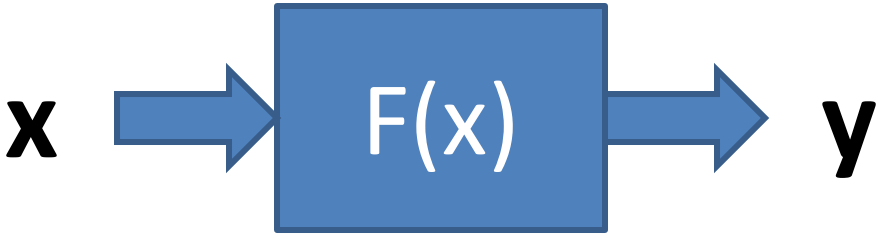


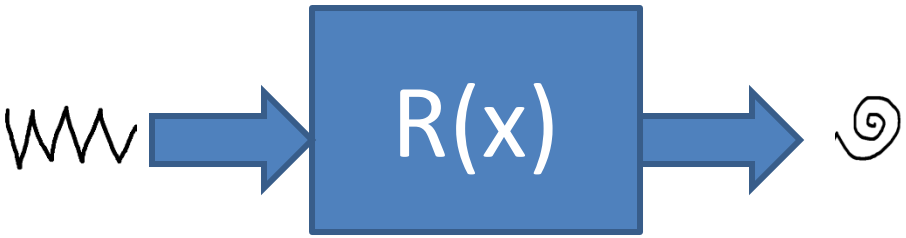


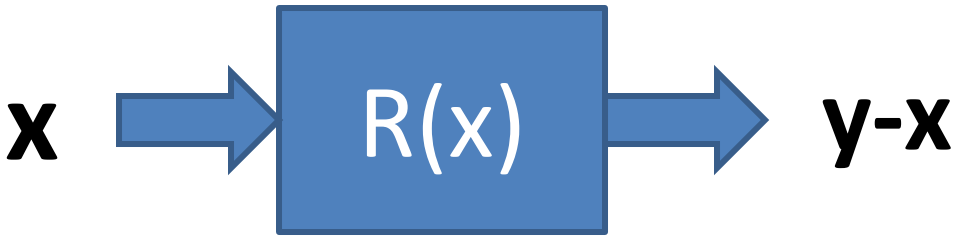
https://mc.ai/explaining-resnet-with-crayons/

Residual Networks – A crayon analogy



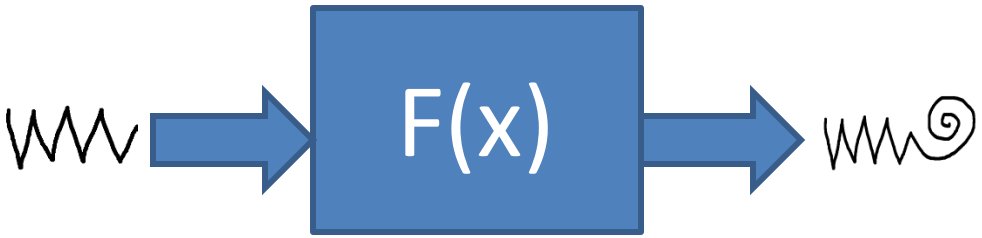


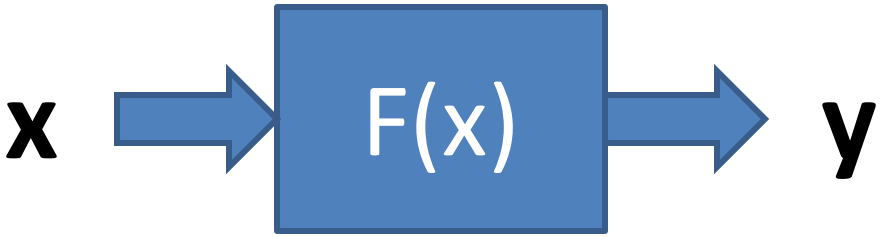


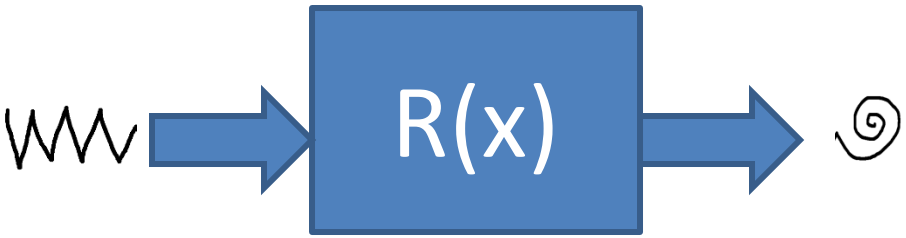


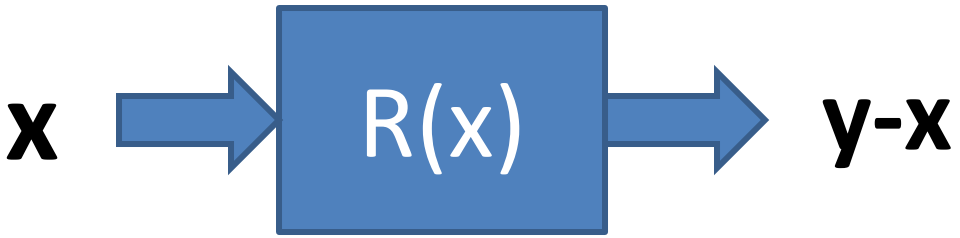
https://mc.ai/explaining-resnet-with-crayons/

Residual Networks – A crayon analogy





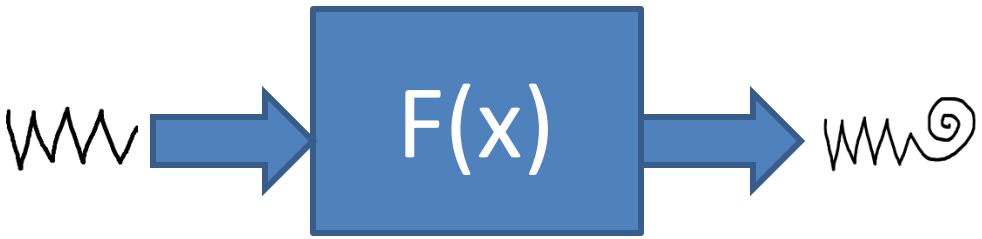


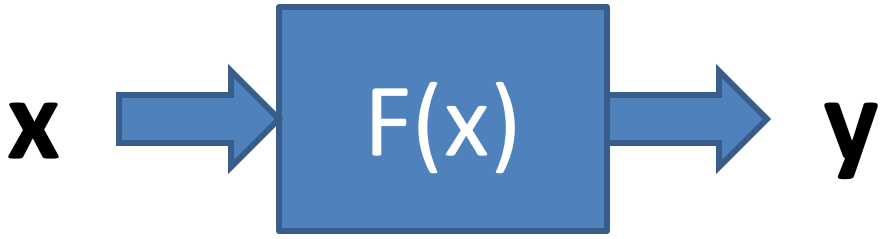


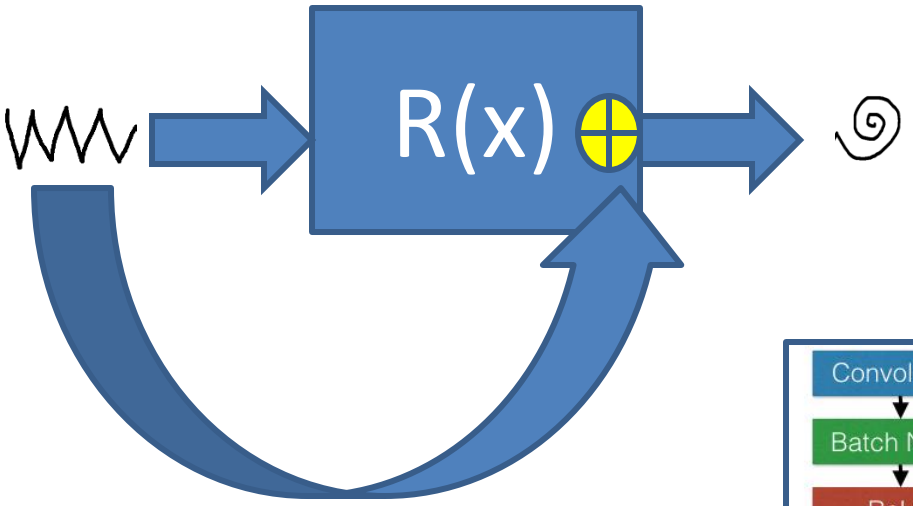
$$y = F(x) = x + R(x)$$

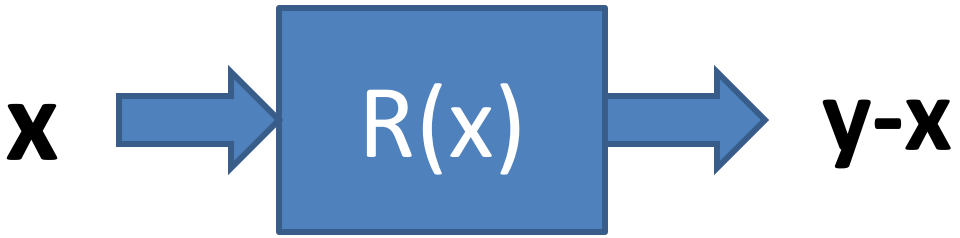
https://mc.ai/explaining-resnet-with-crayons/

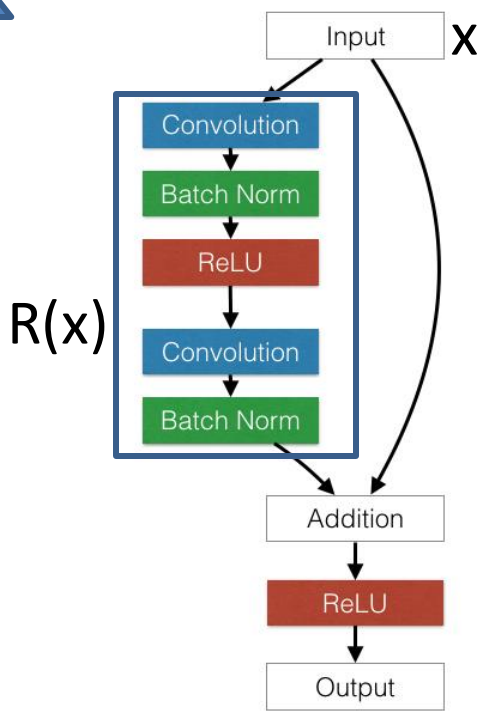
Residual Networks – A crayon analogy









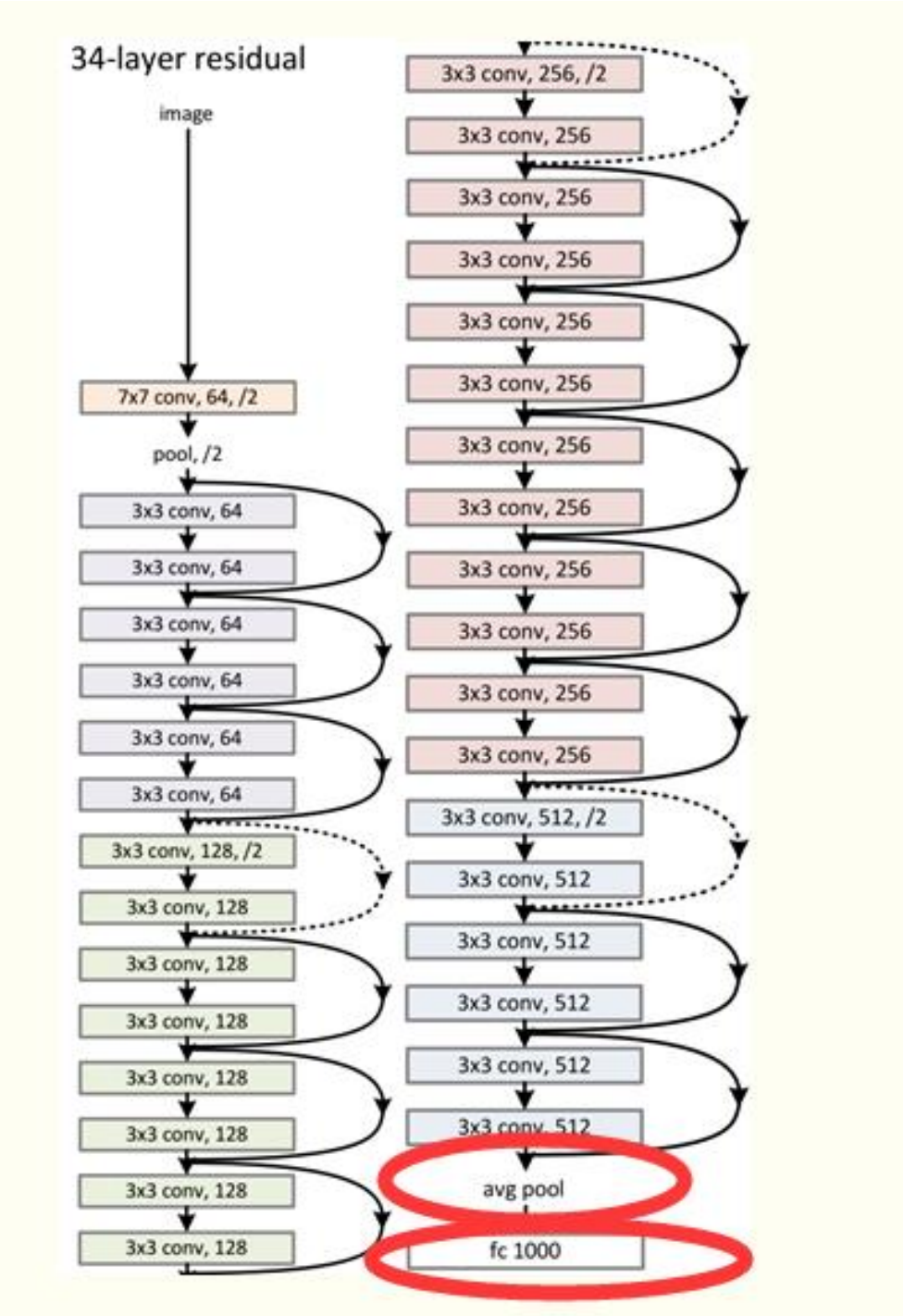


Architecture of a residual block

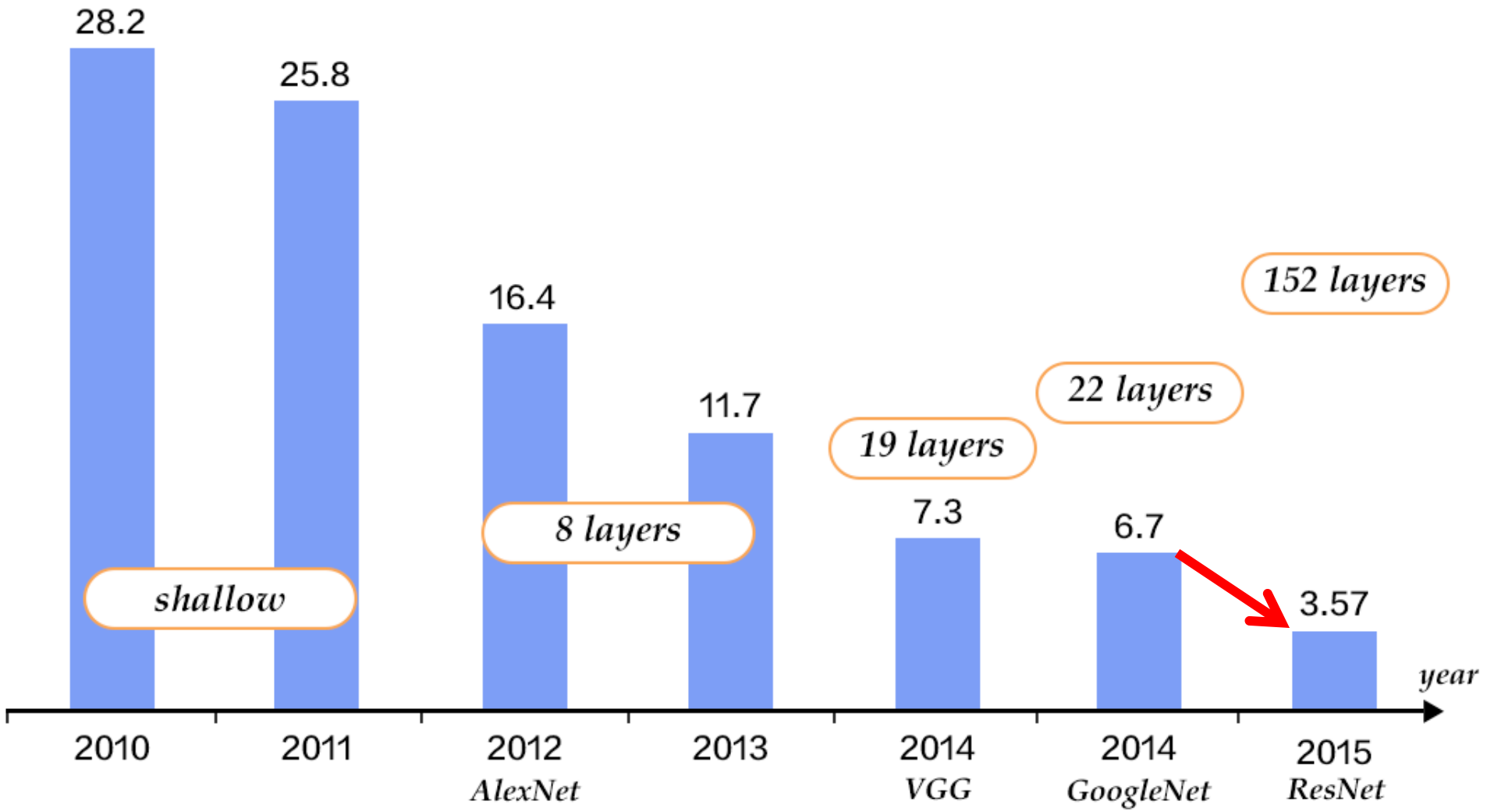
$F(x) = x + R(x)$

https://mc.ai/explaining-resnet-with-crayons/

Residual Networks

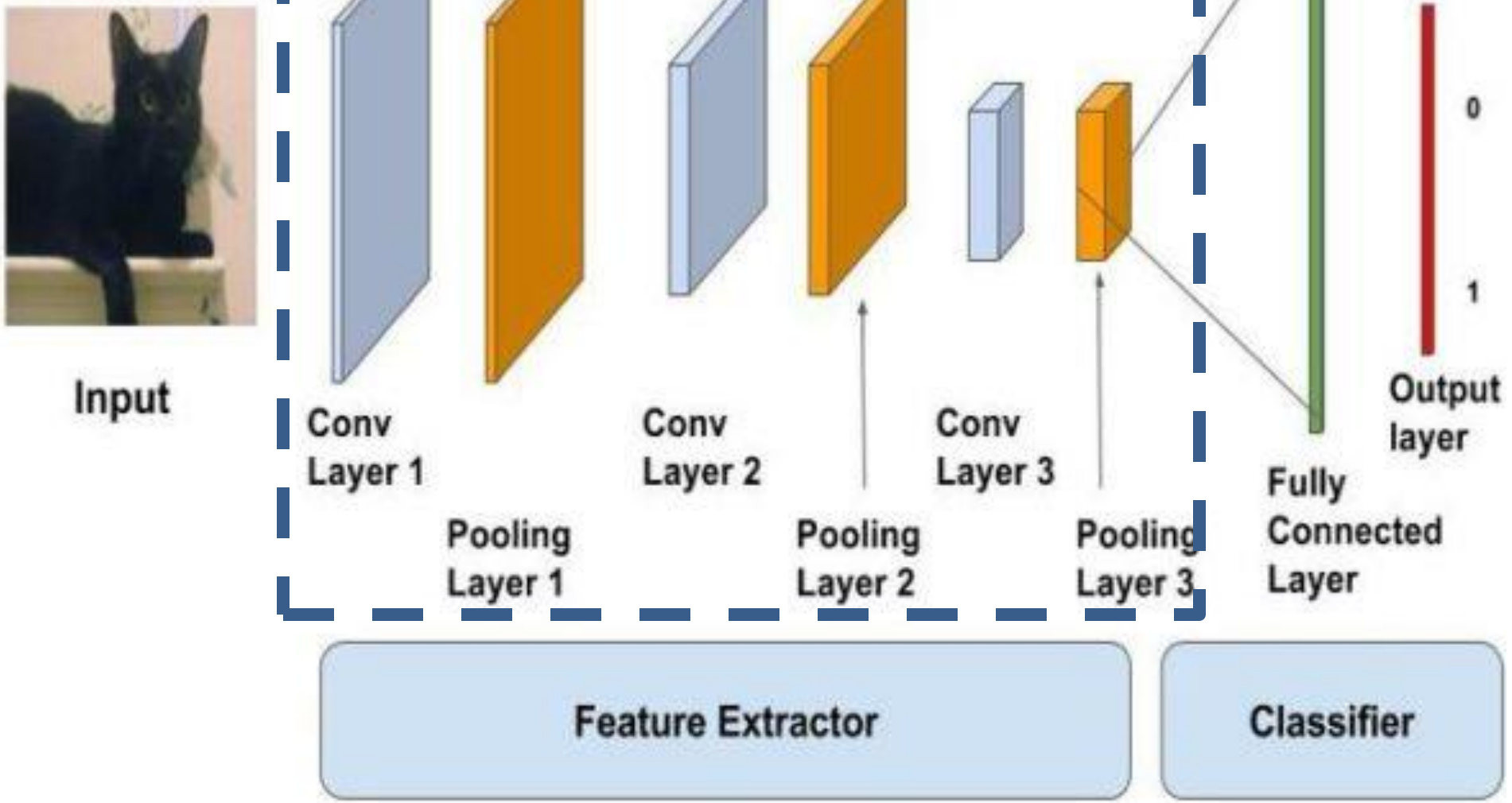


Gradients can be supplied to shallower layers directly if needed (via skips)

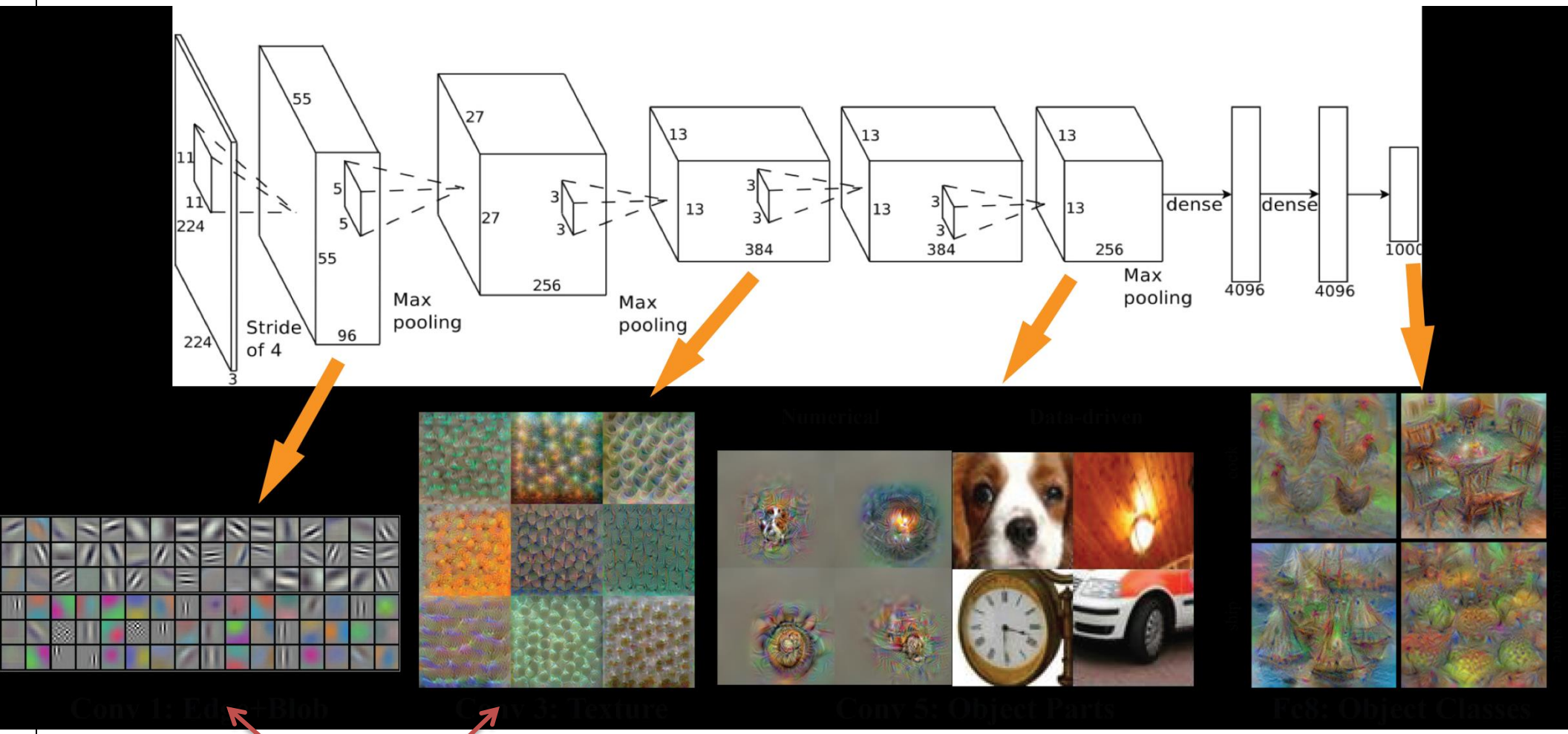

28.2
25.8
16.4
11.7
7.3
6.7
3.57
shallow
8 layers
19 layers
22 layers
152 layers
year
2010
2011
2012
AlexNet
2013
2014
VGG
2014
GoogleNet
2015
ResNet

Landscape of CNNs : Applications and Architectures

Transfer Learning



Filters learnt by AlexNet

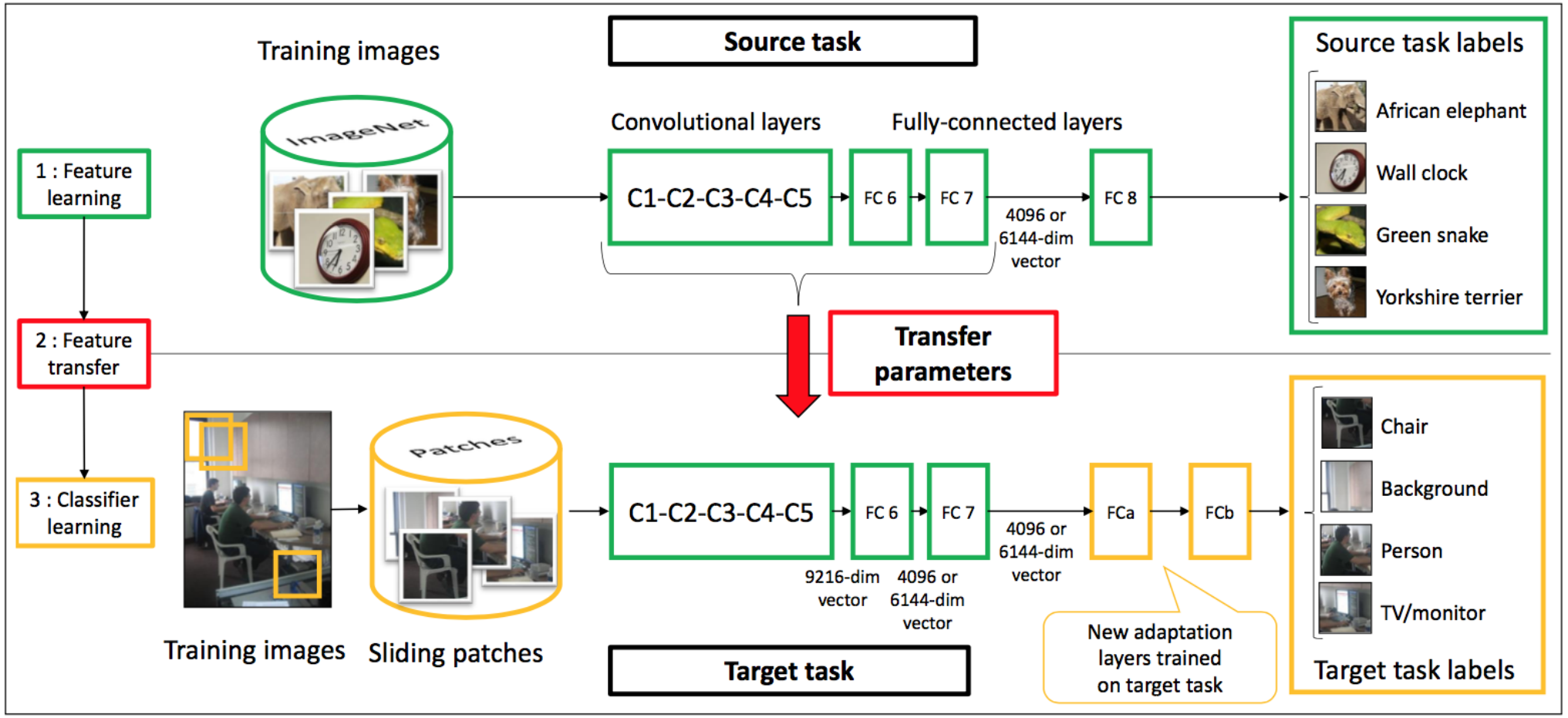


These features seem generic enough. Can we re-use them ?

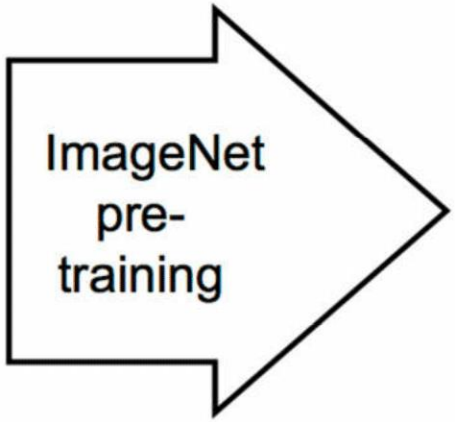
https://www.cc.gatech.edu/~hays/compvision/proj6/deepNetVis.png

Transfer Learning

Transfer Learning



Examples



Pre-trained features are quite general

Object classification (PASCAL VOC)

- [Chatifield et al. 2014, Razavian et al. 2014, Zeiler et al. 2014]

Object detection (PASCAL VOC)

- R-CNN [Girshick et al. 2014]

Fine-grained classification (UCSD birds)

- Part-R-CNN [Zhang et al. 2014]

MIT 67 scene classification

- [Razavin et al. 2014]

…

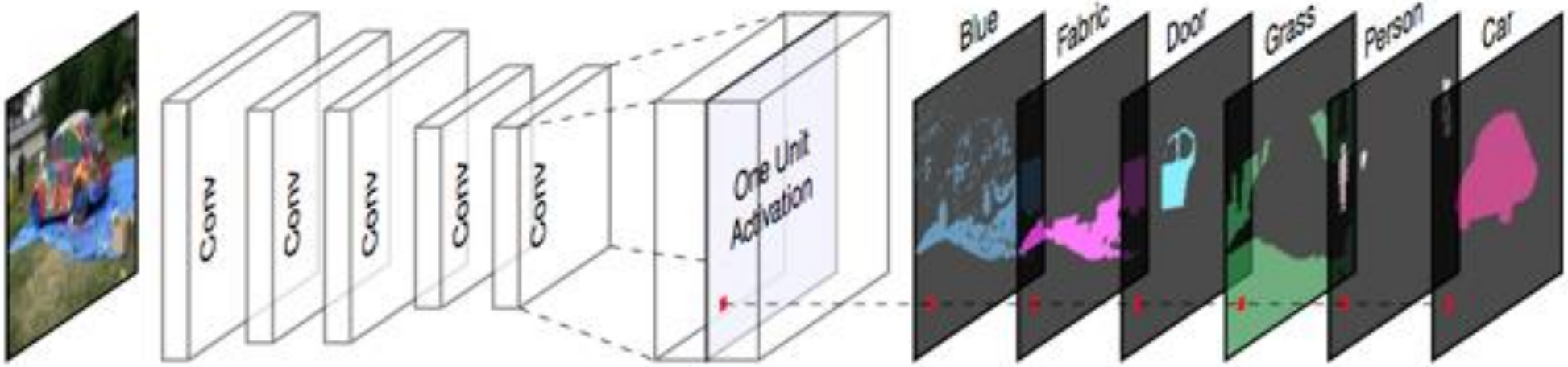
Semantic segmentation

Input

Segmentation [9]

Semantic segmentation

Semantic segmentation

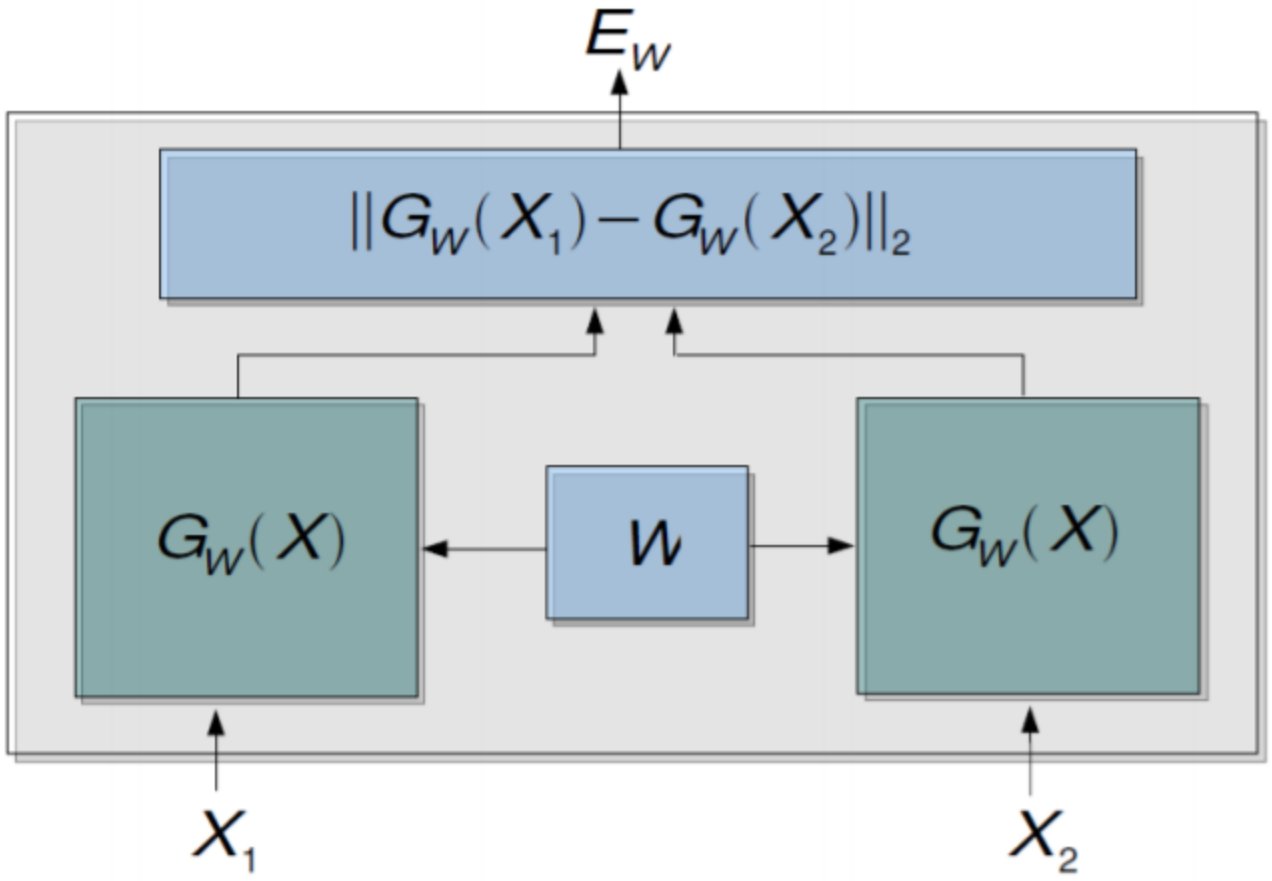


http://netdissect.csail.mit.edu/thumb/slides-thumbnail.png

https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/image_segmentation.html

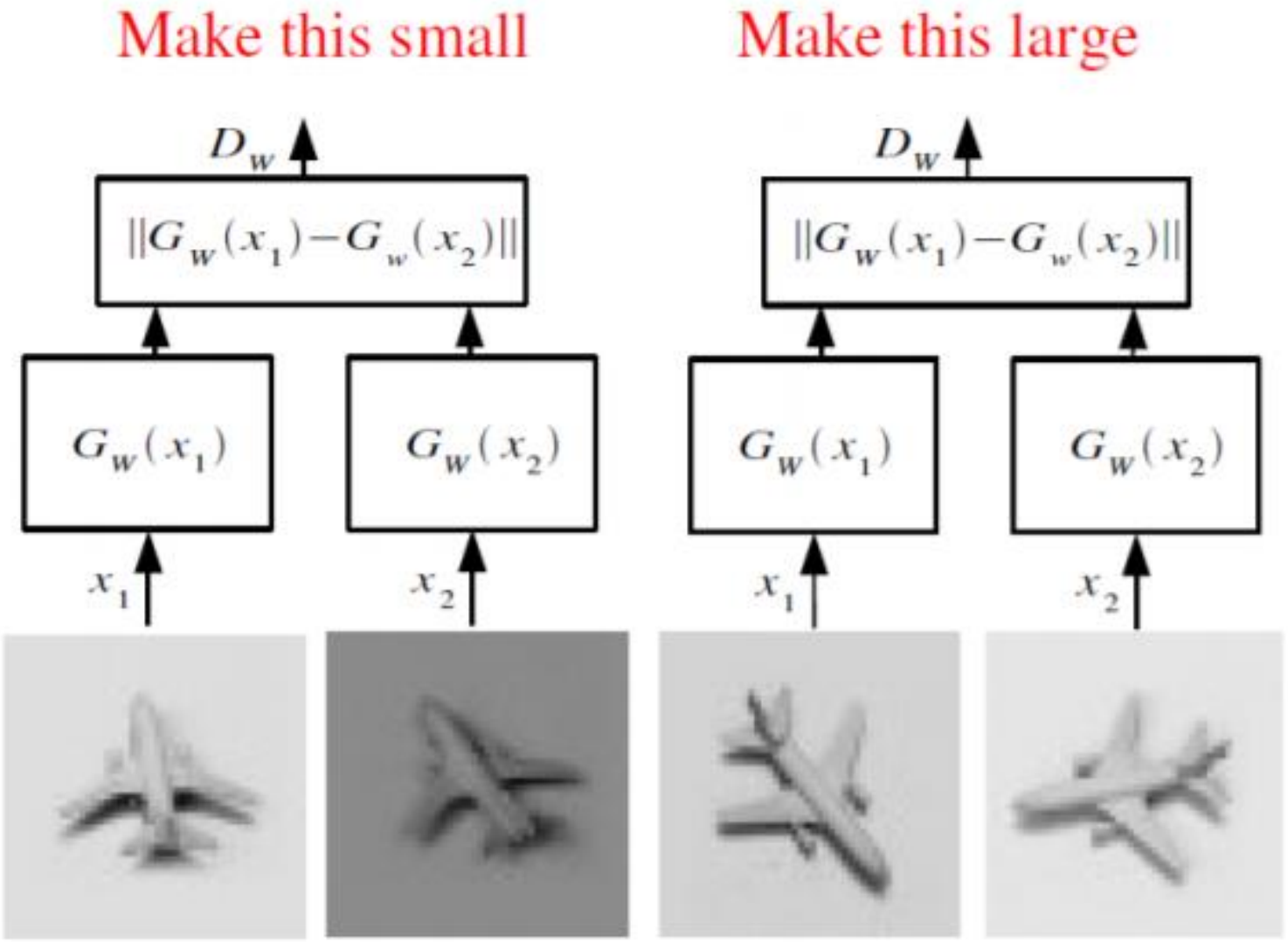
Siamese Network

Siamese Network



Learning Hierarchies of Invariant Features. Yann LeCun

Siamese Network



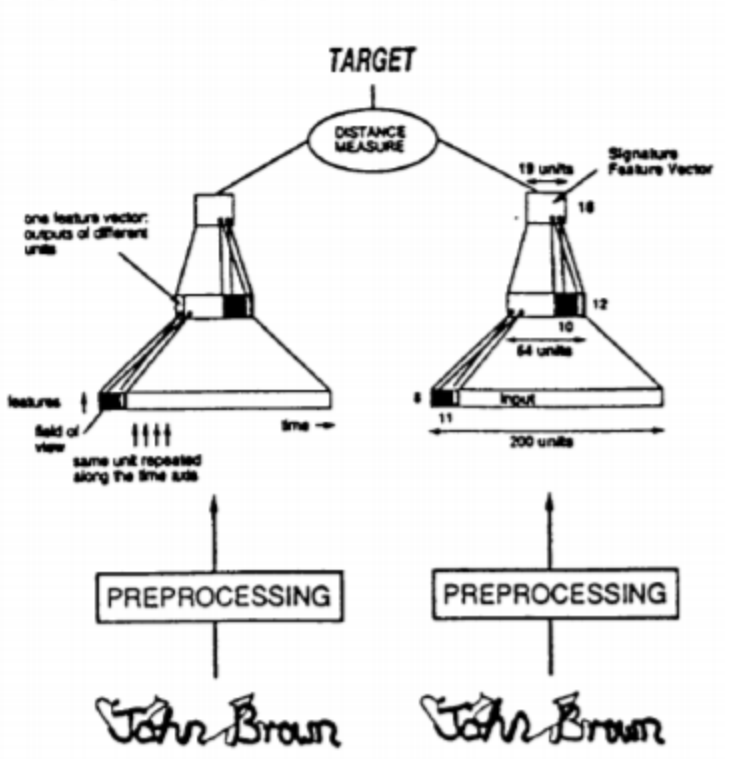
Similar Images **Dissimilar Images**

Learning Hierarchies of Invariant Features. Yann LeCun

Siamese Network

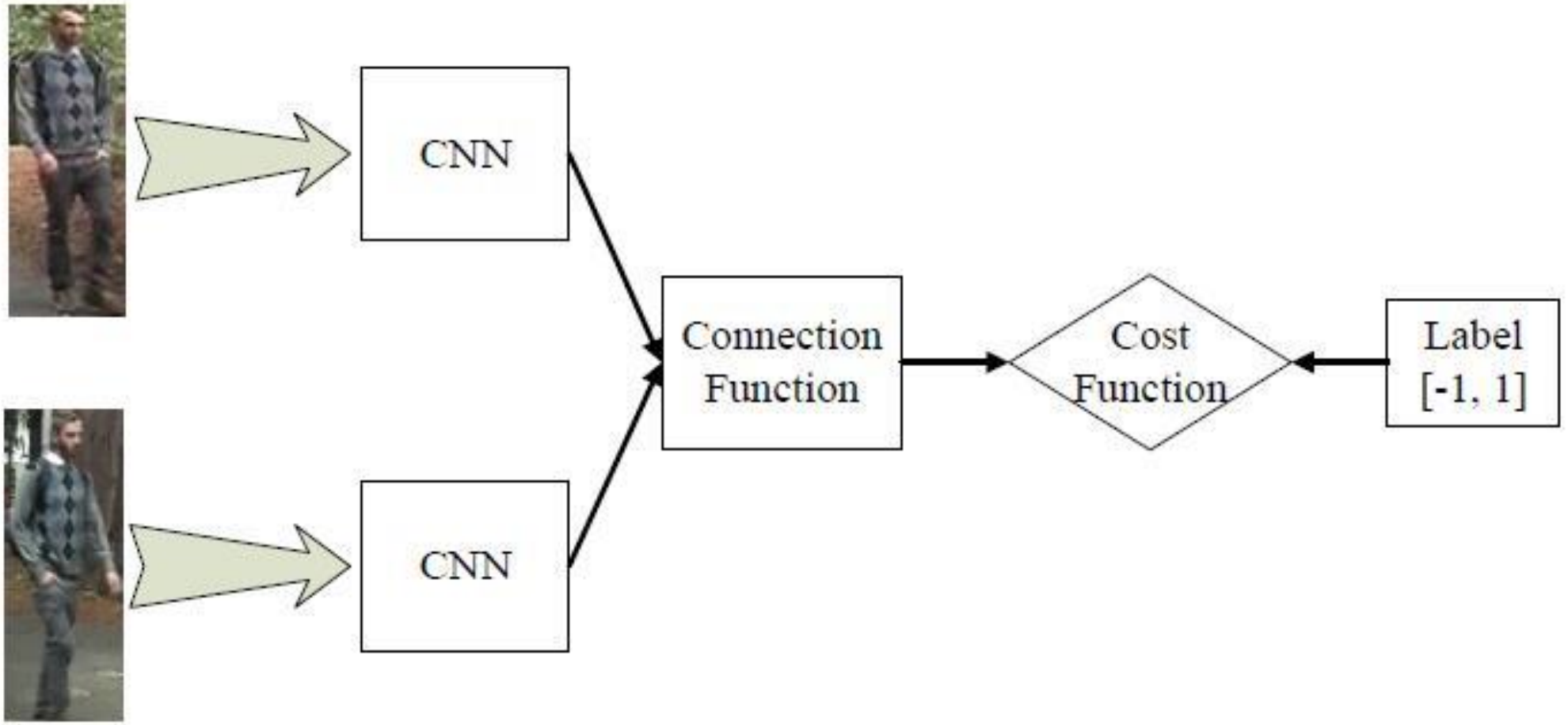
Application in Signature Verification

- The input is 8(feature) x 200(time) units.
- The cosine distance was used, (1 for genuine pairs, -1 for forgery pairs)



Bromley J, Guyon I, Lecun Y, et al. Signature Verification using a" Siamese" Time Delay Neural Network, NIPS Proc. 1994

Siamese Network (Person re-id)

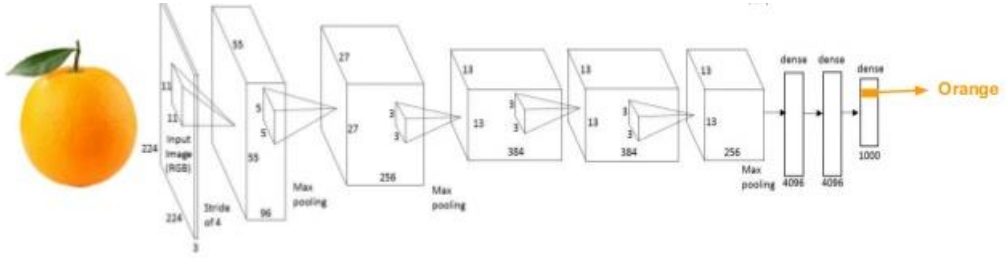


http://www.fubin.org/research/Person_ReID/Person_ReID.html

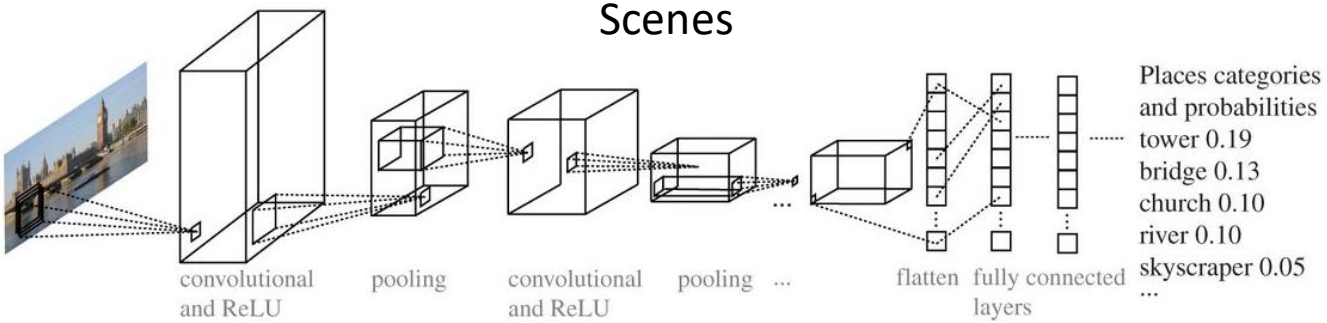
Landscape of CNNs : Architectures

Image → Label

Objects



Scenes



Faces

DeepFace Architecture

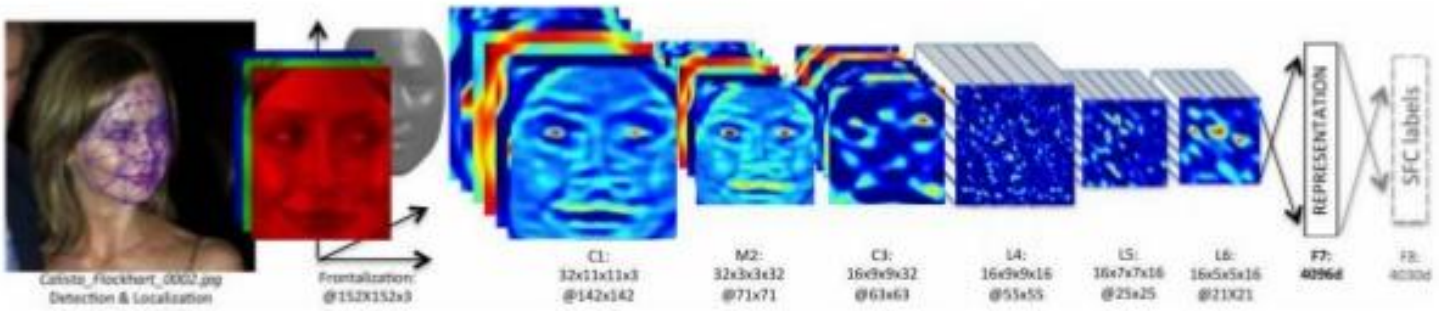
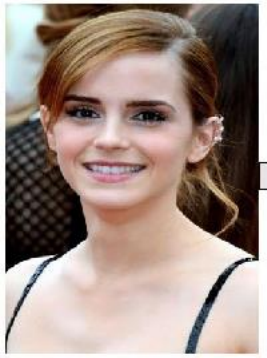
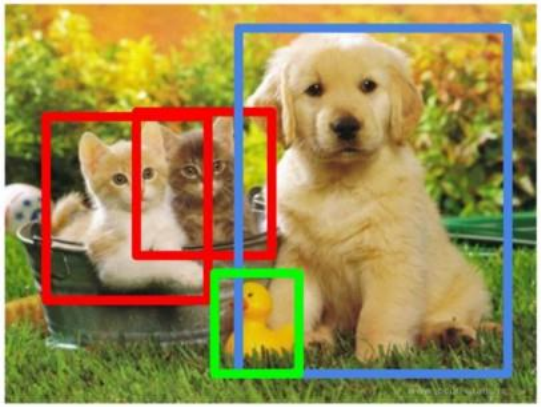


Image → Number



Face Detection → Face Alignment → Deep Feature Representation (from face-identification task) → 3-layer Neural Network → Age

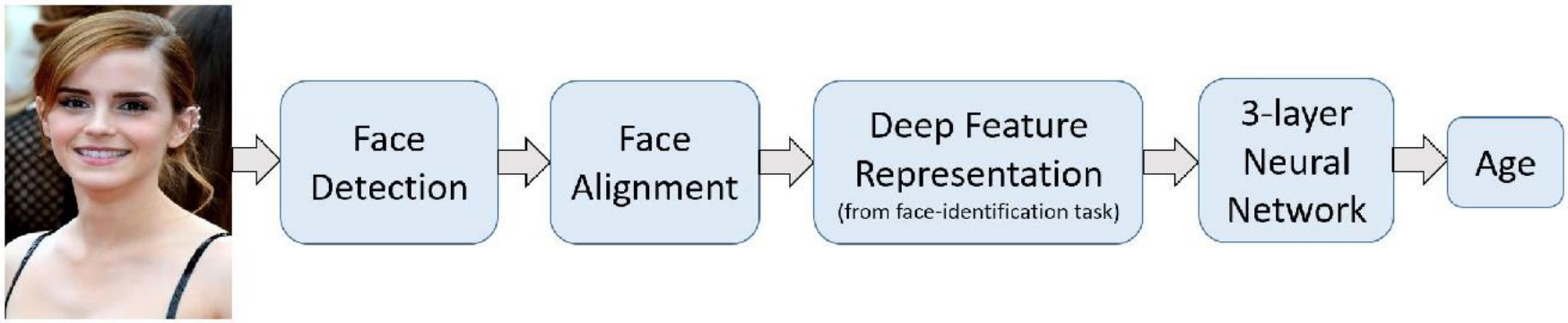
Object Detection



CAT, DOG, DUCK

Image → Number

Age Estimation



Object detection

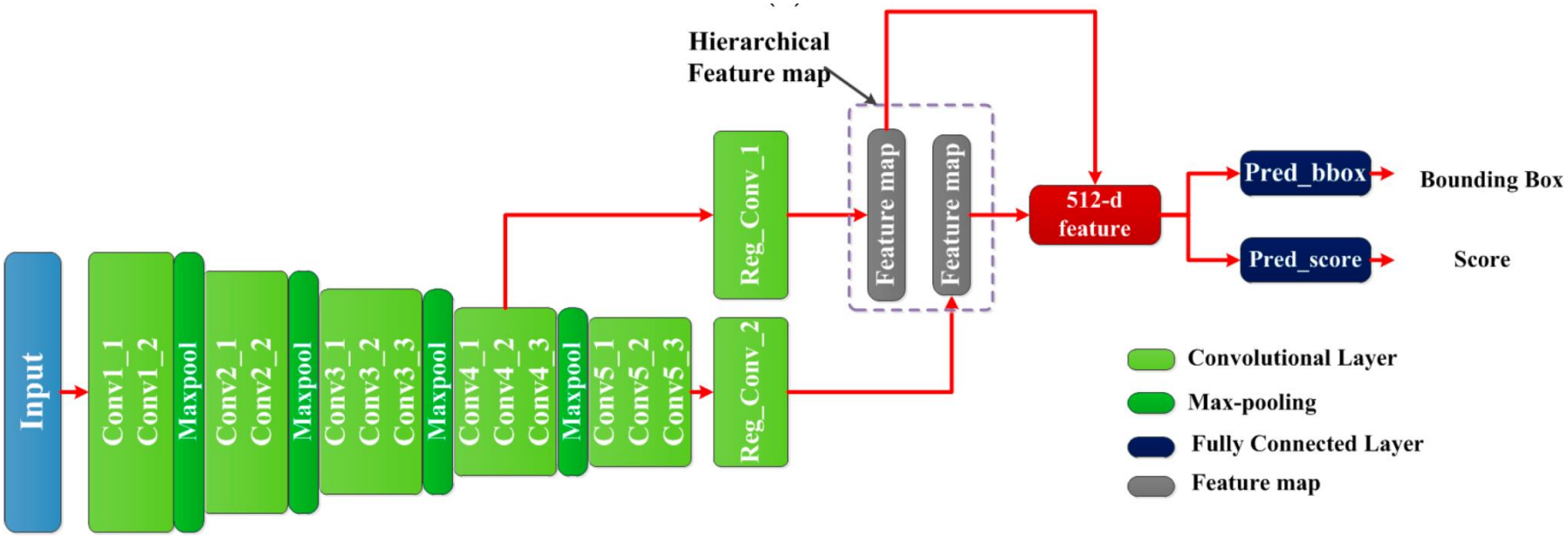
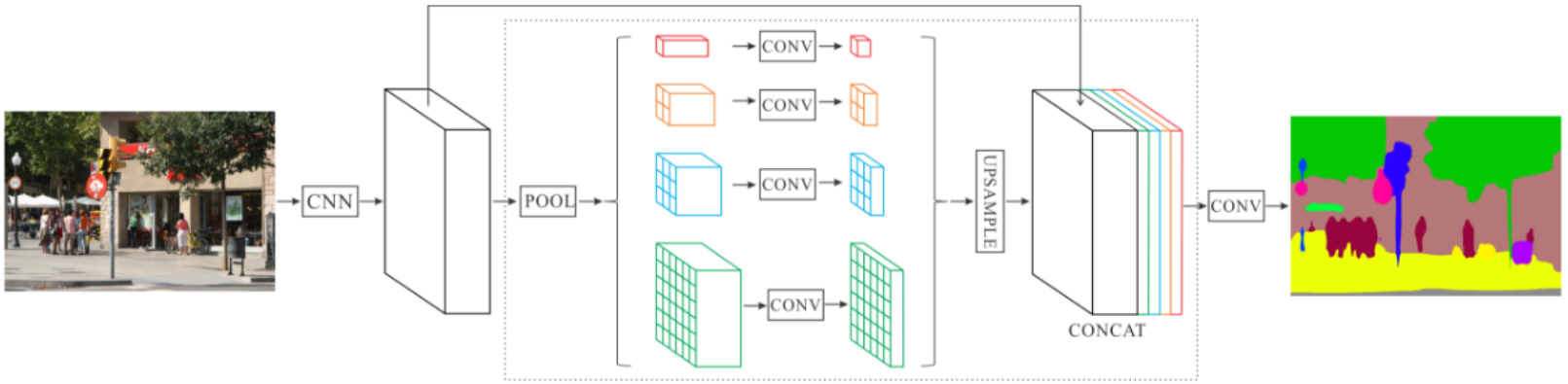


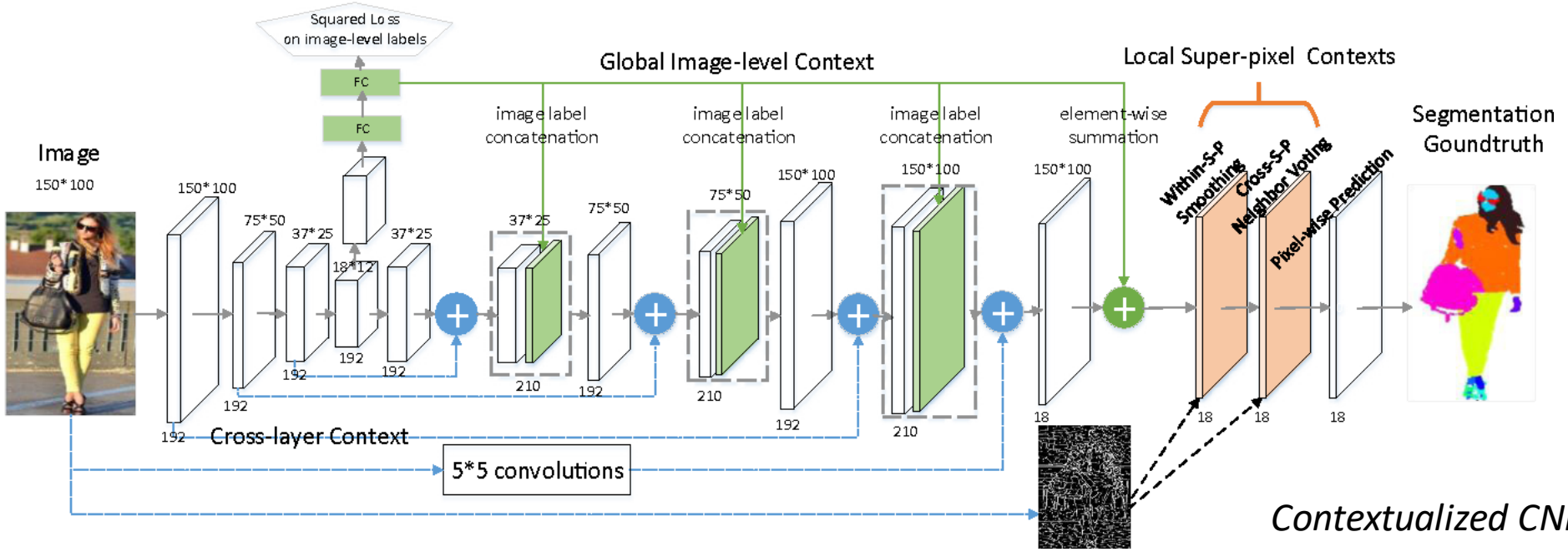
Image → Label Image

Scene Parsing



PSPNet

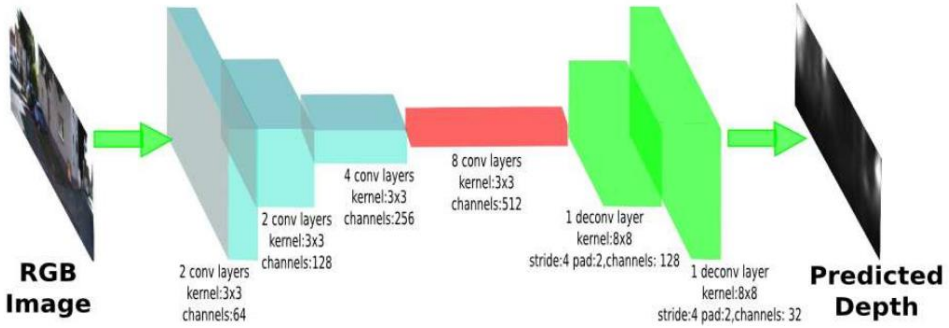
Object Parsing



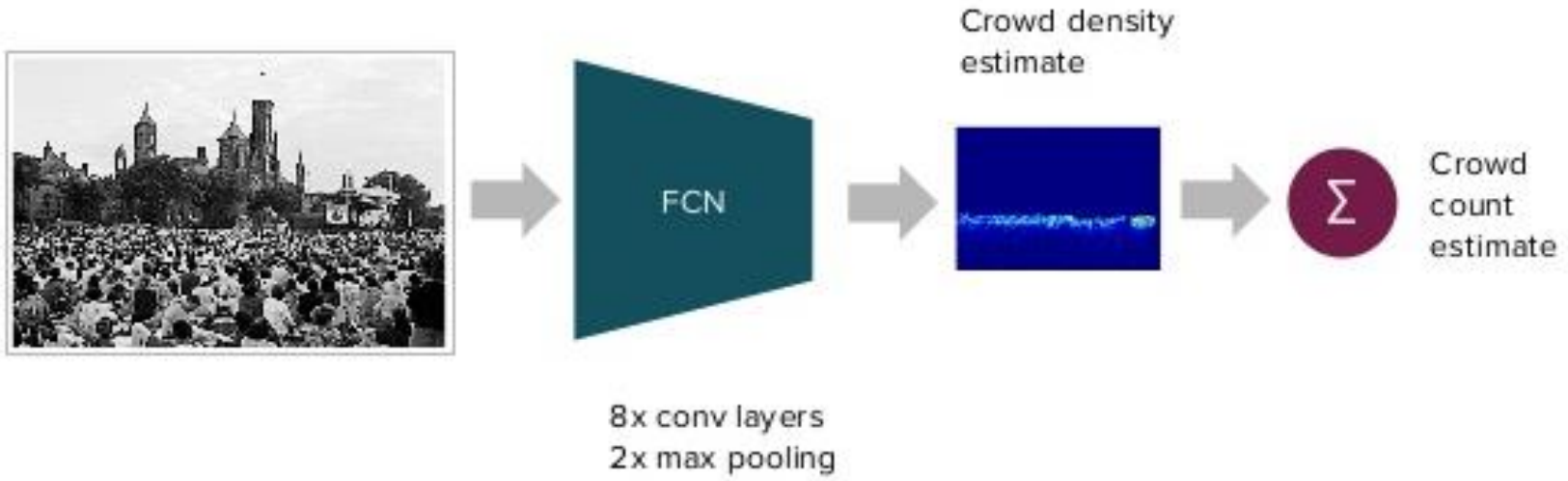
Contextualized CNN

Image → Image

Depth Estimation



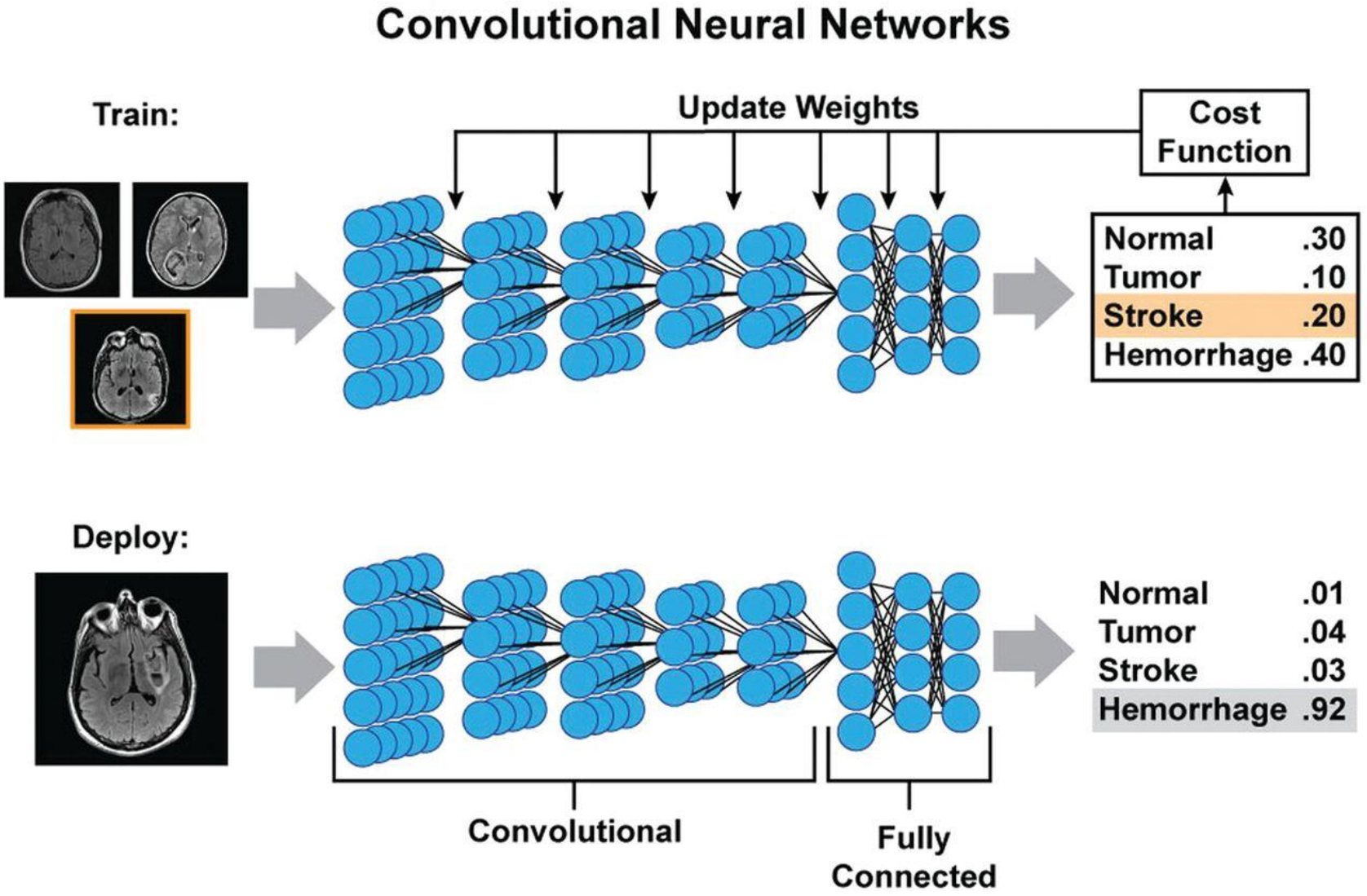
Crowd Counting



Marsden et al. **Fully convolutional crowd counting on highly congested scenes.** VISAPP 2017
https://arxiv.org/abs/1612.00220

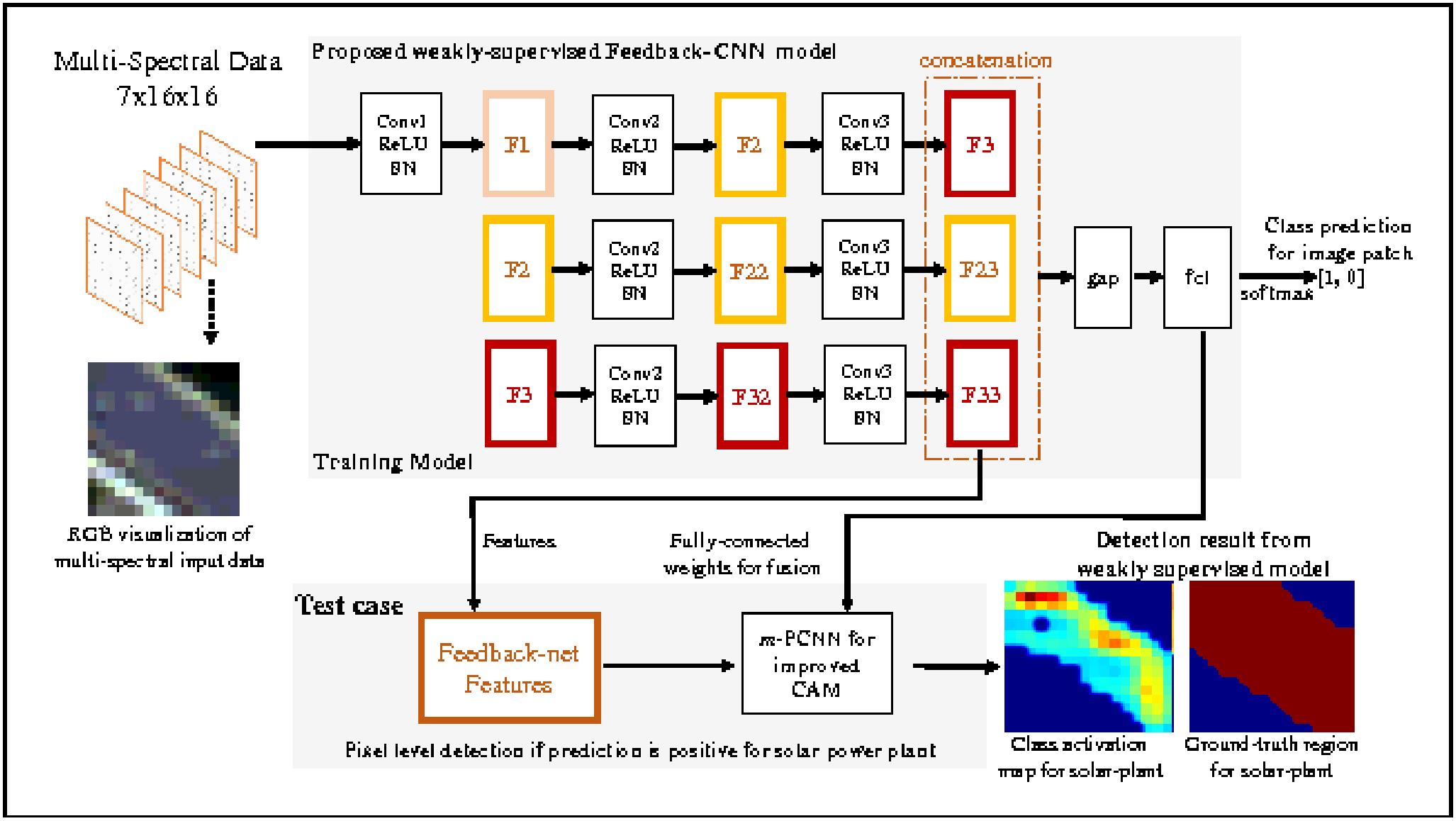
Multi-channel input

NeuroRadiology (fMRI)



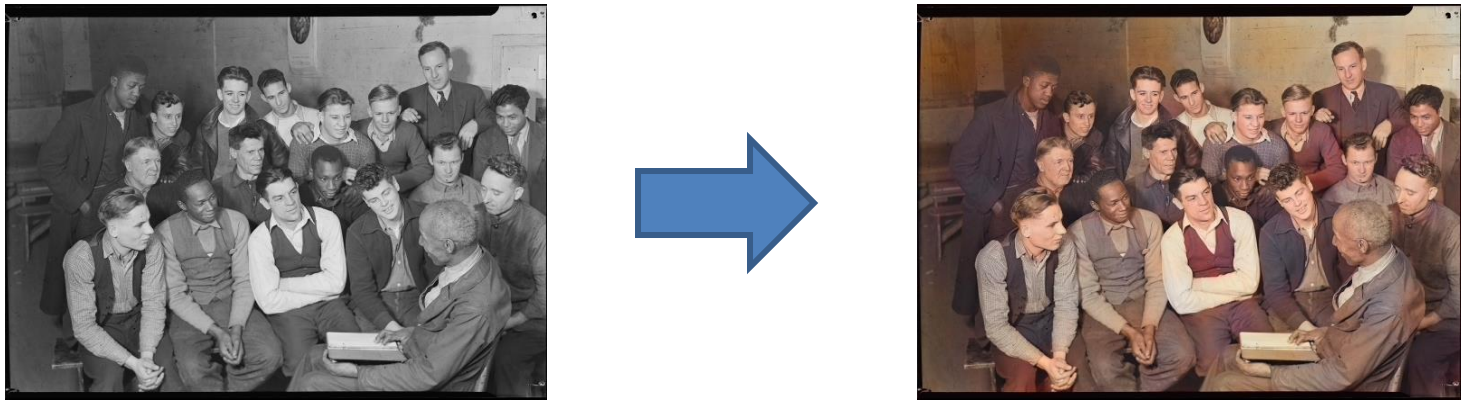
Multi-channel input

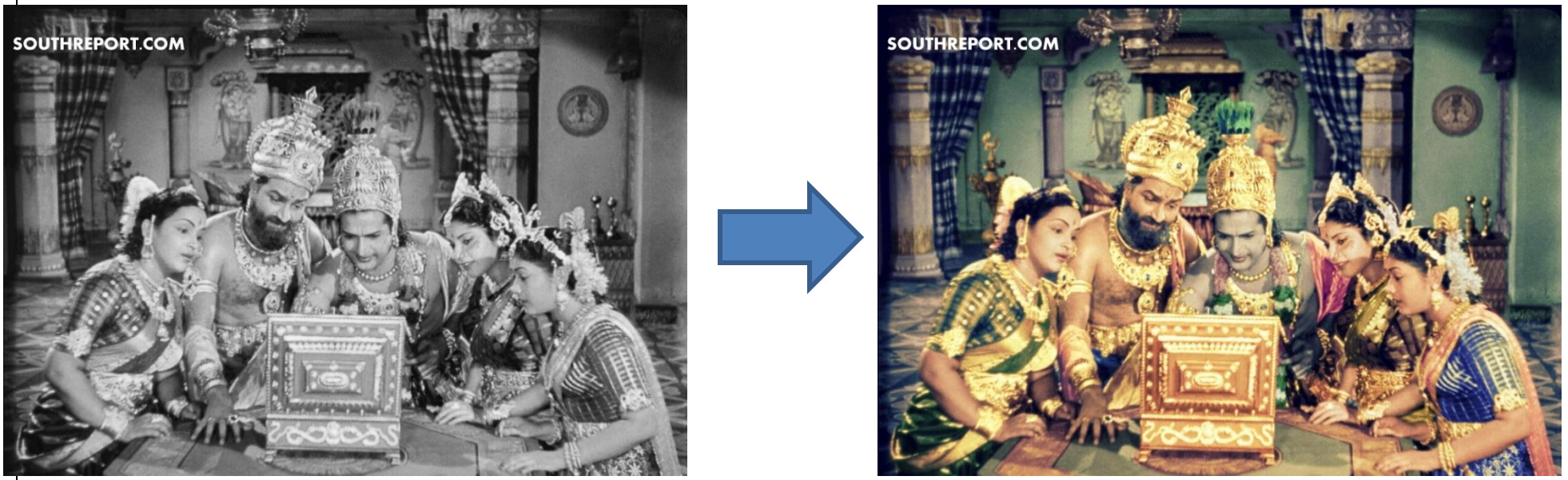
Solar-power plant detection from multi-spectral data



Multi-branch input

Image Colorization

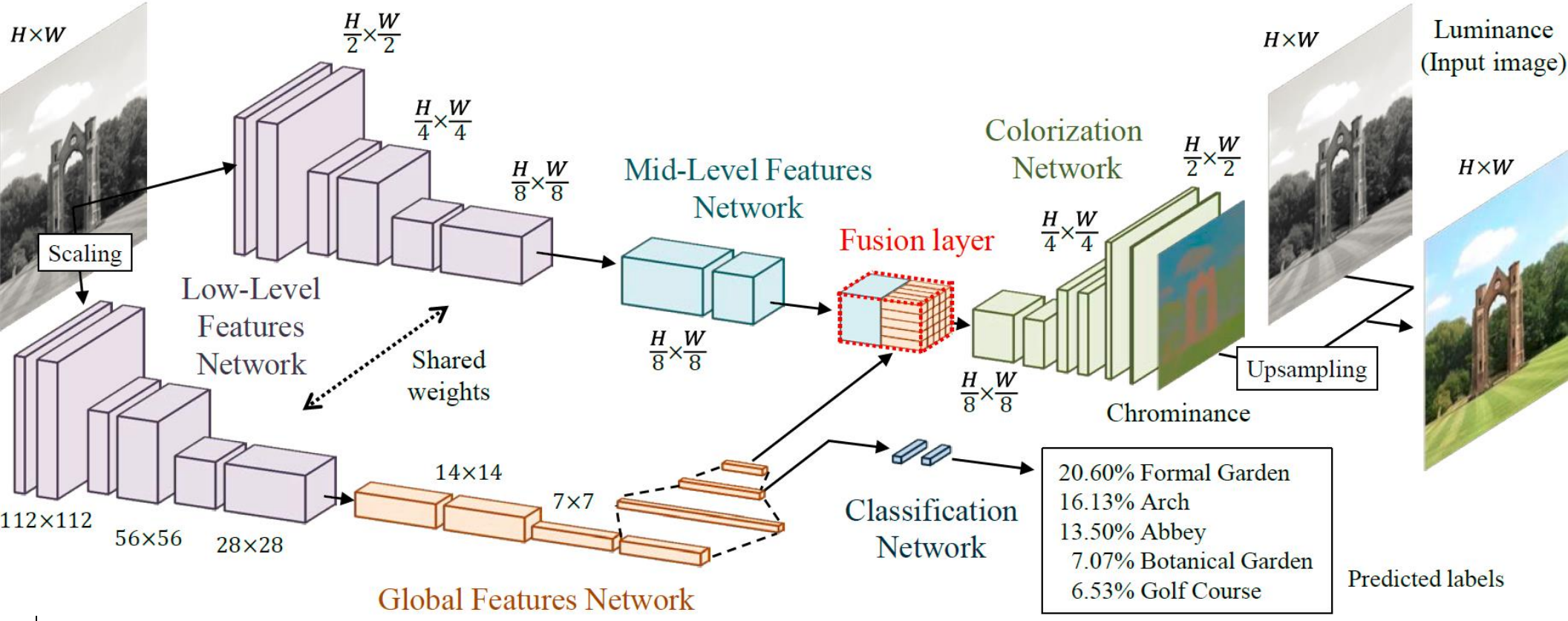




https://algorithmia.com/algorithms/deeplearning/ColorfulImageColorization

Multi-branch input

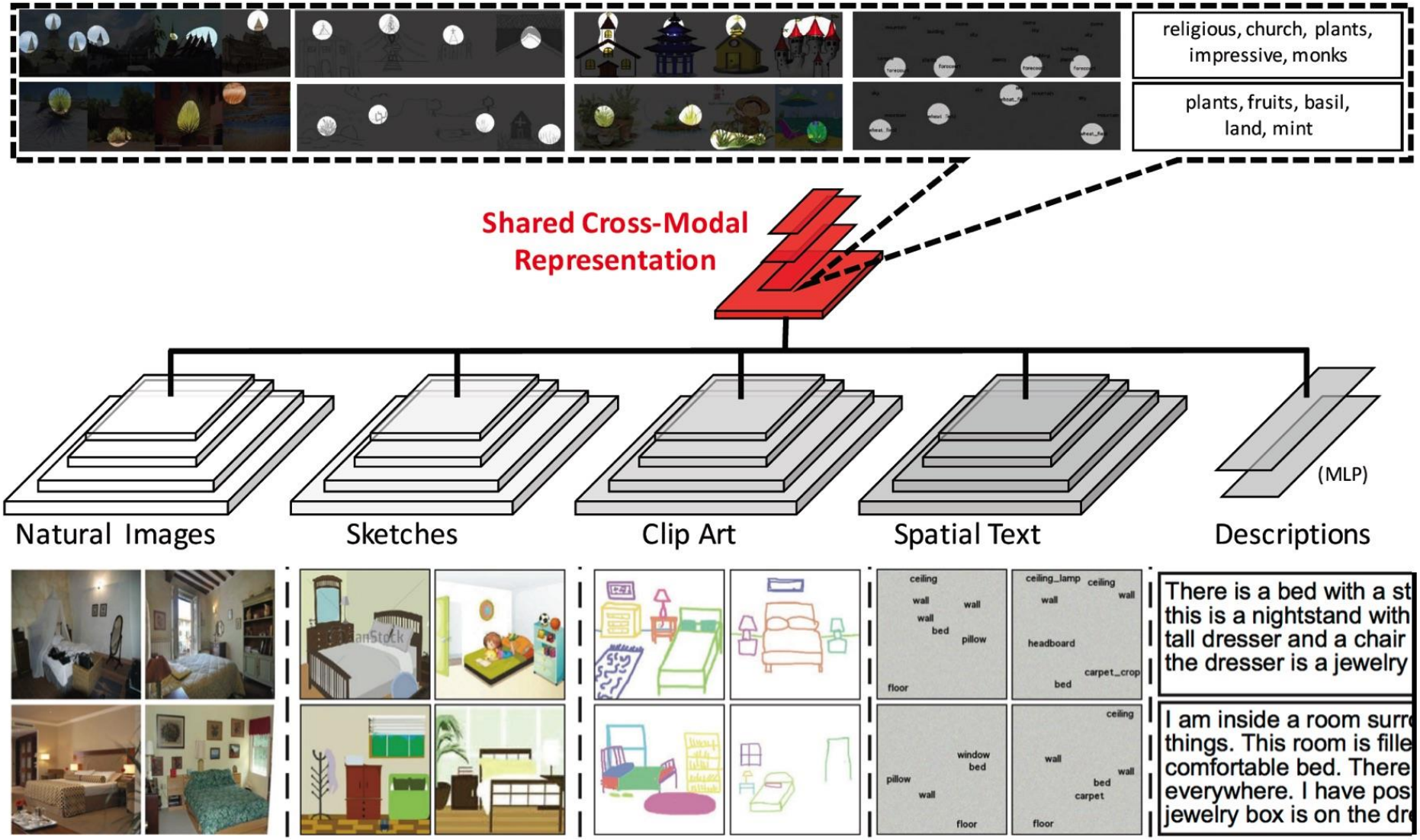
Image Colorization



http://hi.cs.waseda.ac.jp/~iizuka/projects/colorization/en/

Multi-branch input

Cross-modal Scene Classification



http://projects.csail.mit.edu/cmplaces/

Multi-branch input

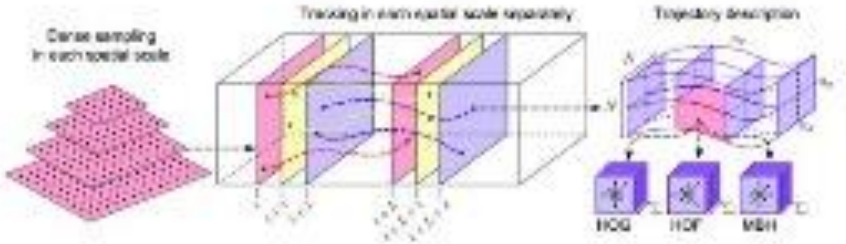
Activity Recognition

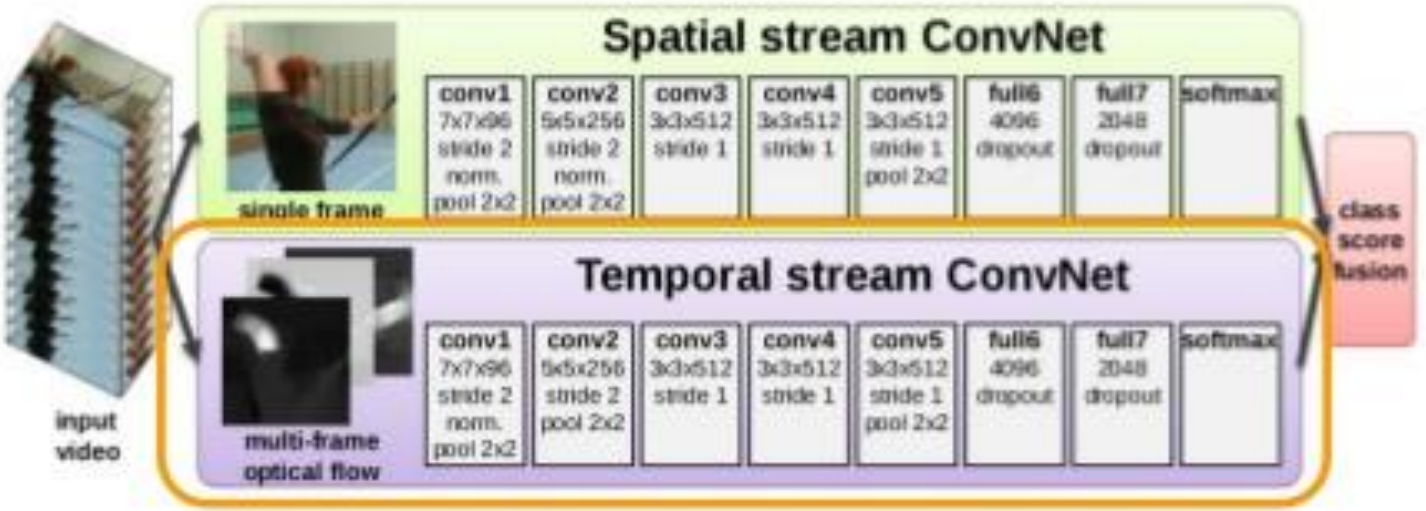
Recognition: Two stream

Two CNNs in paralel:

- One for RGB images
- One for Optical flow (hand-crafted features)

Fusion after the softmax layer

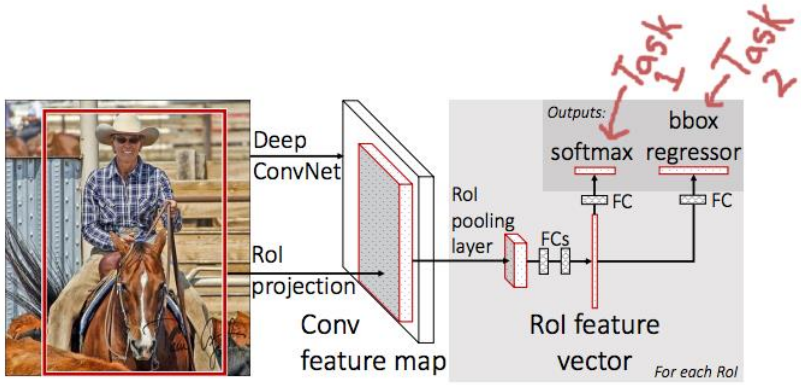




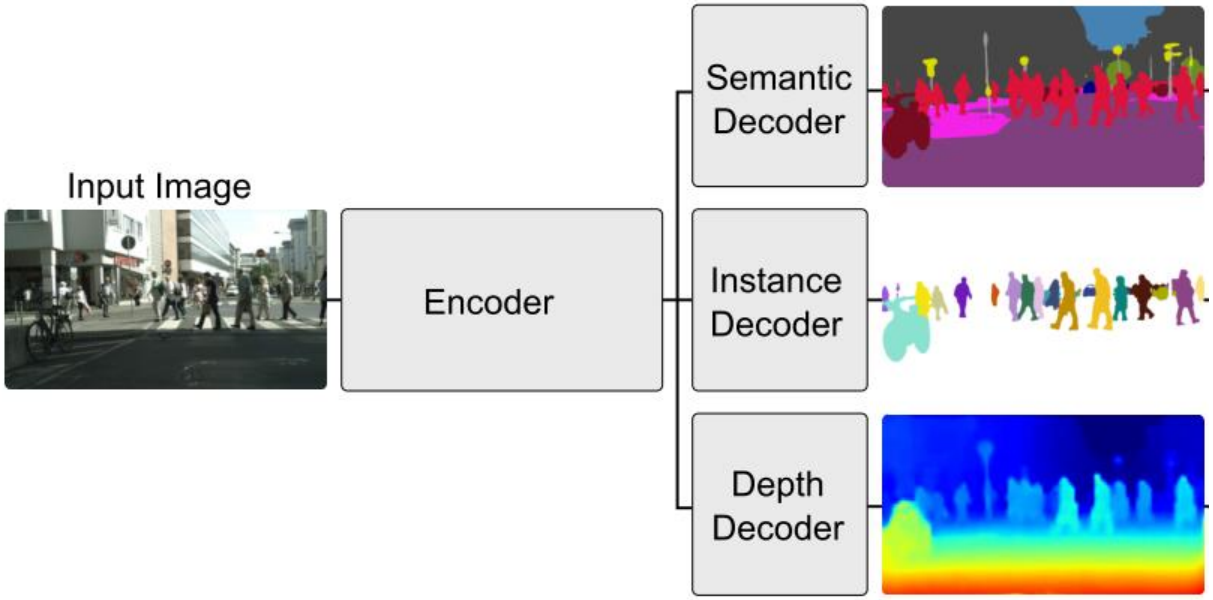
Simonyan, Karen, and Andrew Zisserman. "Two-stream convolutional networks for action recognition in videos." NIPS 2014.

Multi-branch output

Object Detection, Classification



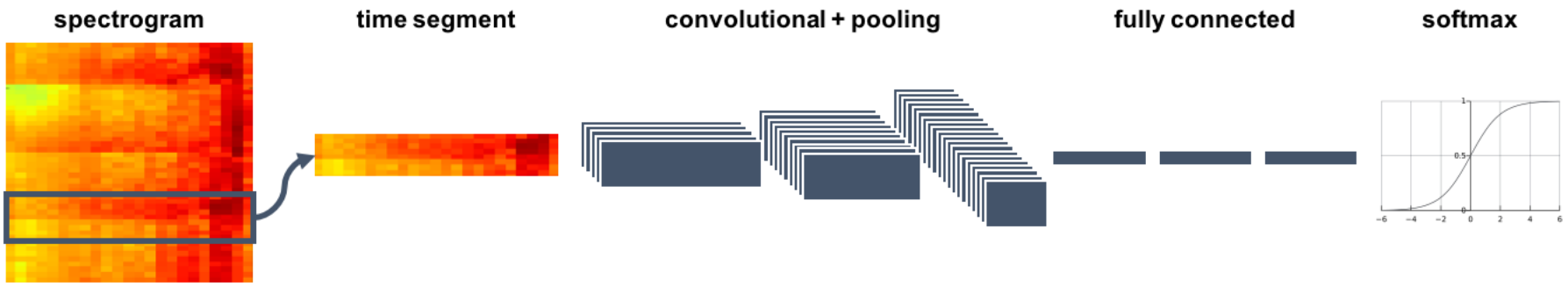
Scene Parsing



http://ruder.io/multi-task/

Image CNNs for non-image data

Audio Beat Detection



http://tommymullaney.com/projects/rhythm-games-neural-networks