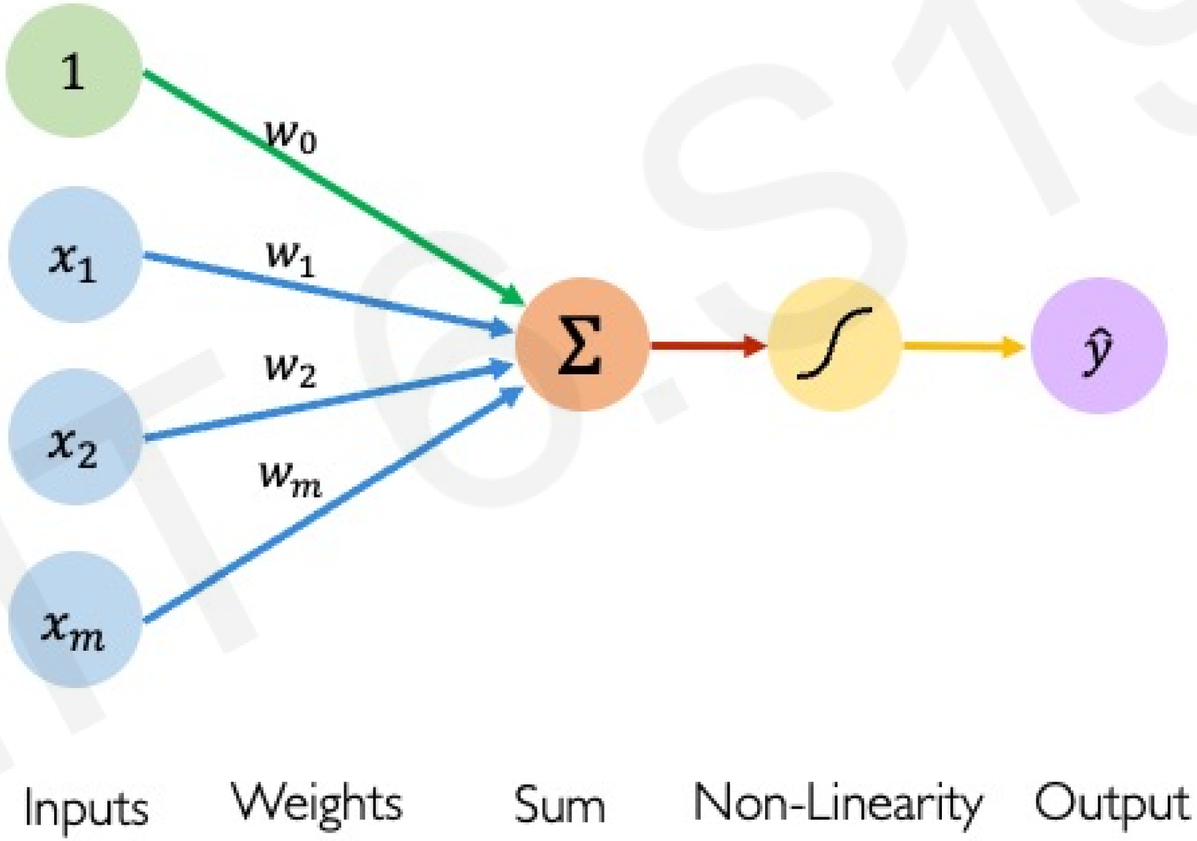
The Perceptron: Simplified

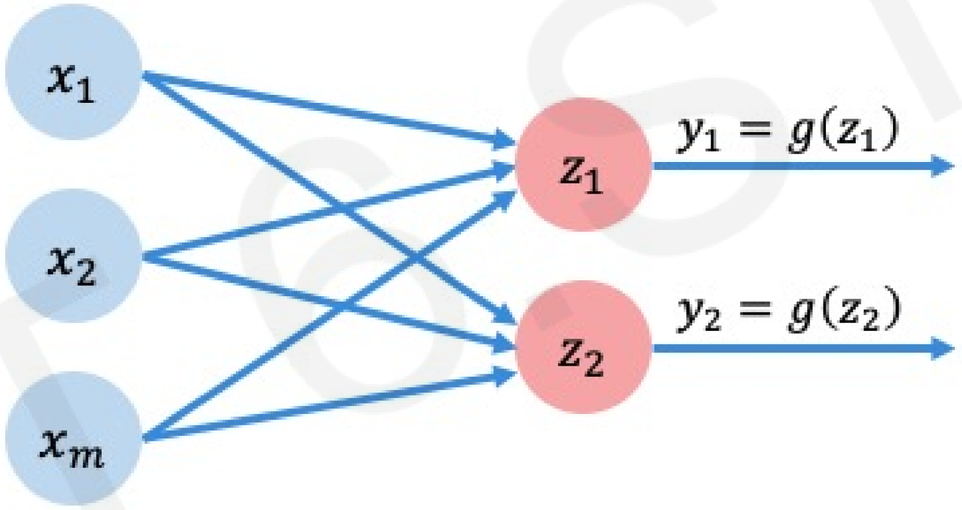
$$\hat{y} = g\left(w_0 + \boldsymbol{X}^T \boldsymbol{W} \right)$$



MIT 6.S191

Multi Output Perceptron

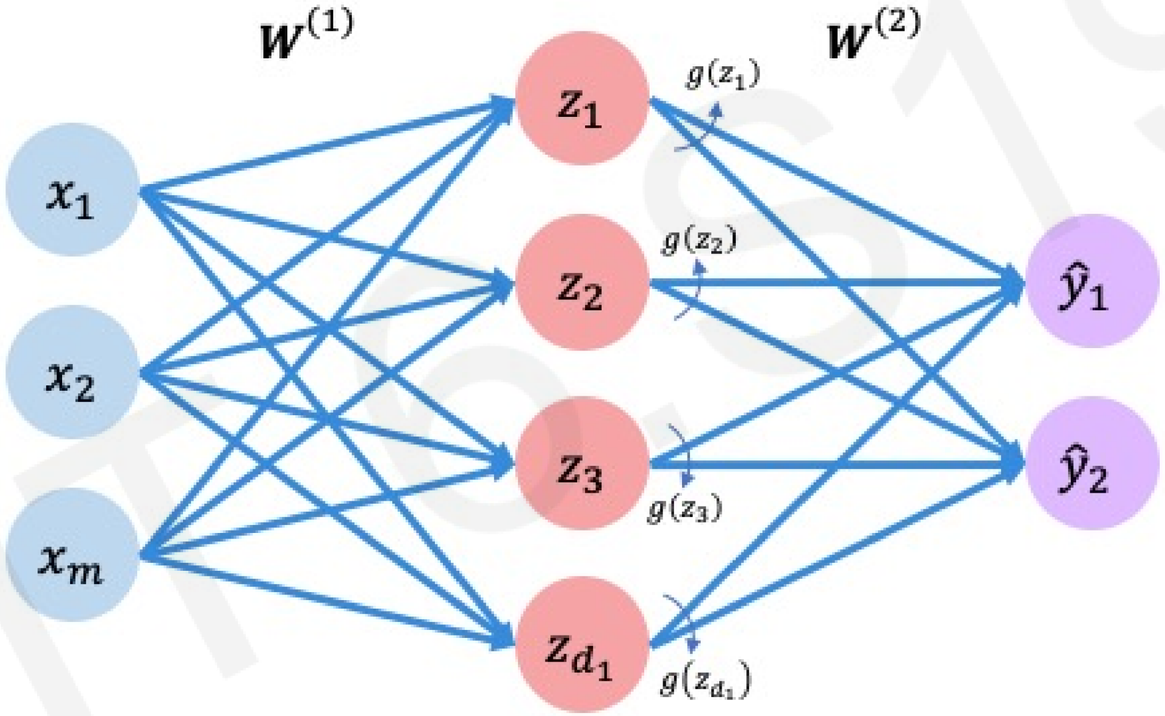
Because all inputs are densely connected to all outputs, these layers are called **Dense** layers



$$z_i = w_{0,i} + \sum_{j=1}^{m} x_j \, w_{j,i}$$

MIT 6S191

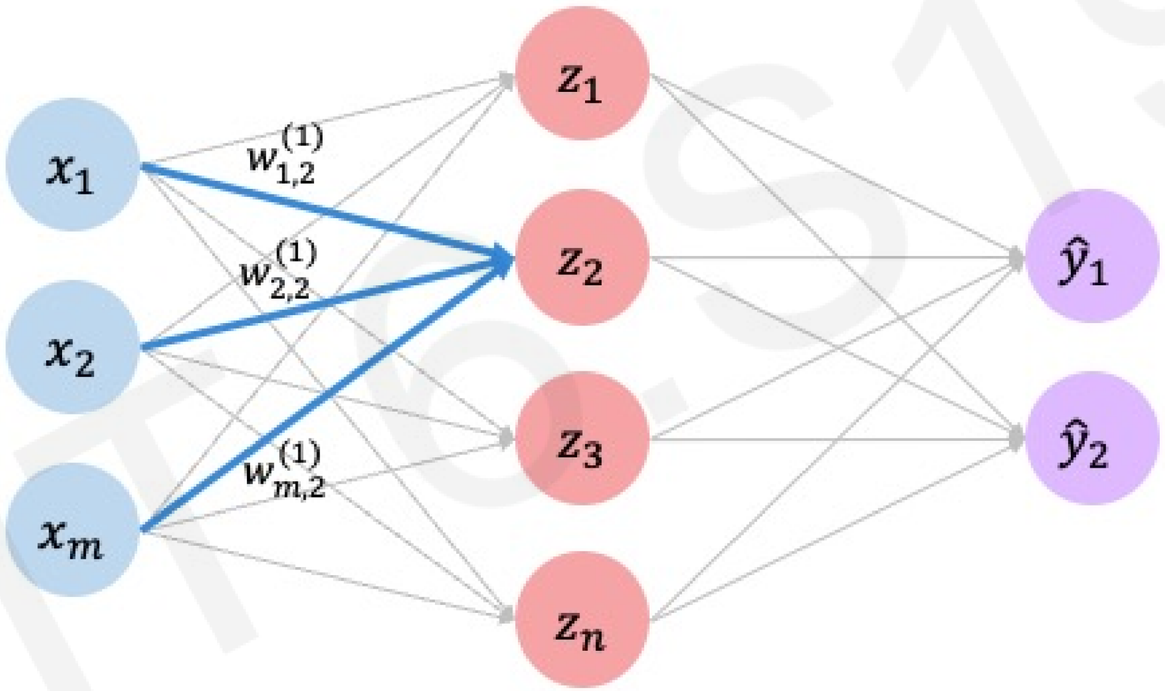
Single Layer Neural Network



$$z_i = w_{0,i}^{(1)} + \sum_{j=1}^{m} x_j \, w_{j,i}^{(1)} \qquad \hat{y}_i = g\left(w_{0,i}^{(2)} + \sum_{j=1}^{d_1} g(z_j) \, w_{j,i}^{(2)}\right)$$

MIT 6.S191

Single Layer Neural Network



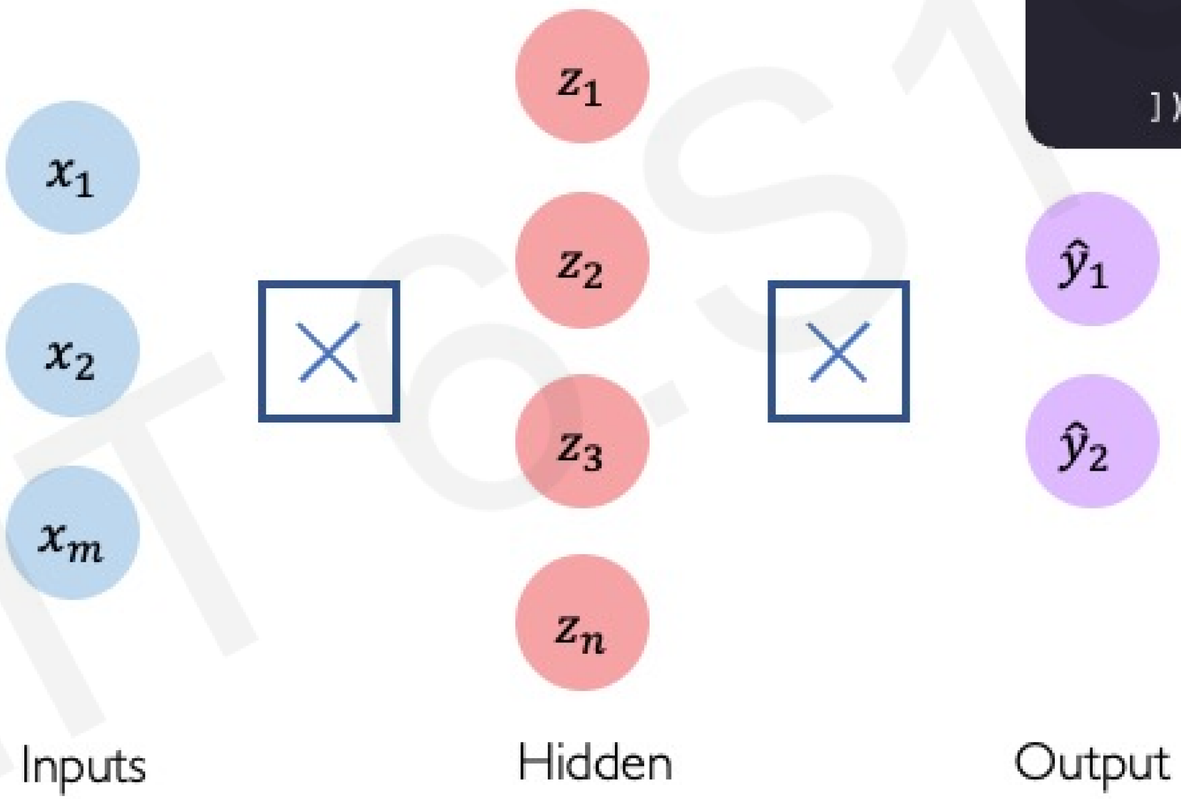
$$z_2 = w_{0,2}^{(1)} + \sum_{j=1}^{m} x_j \, w_{j,2}^{(1)}$$

$$= w_{0,2}^{(1)} + x_1 \, w_{1,2}^{(1)} + x_2 \, w_{2,2}^{(1)} + x_m \, w_{m,2}^{(1)}$$

MIT 6.S191

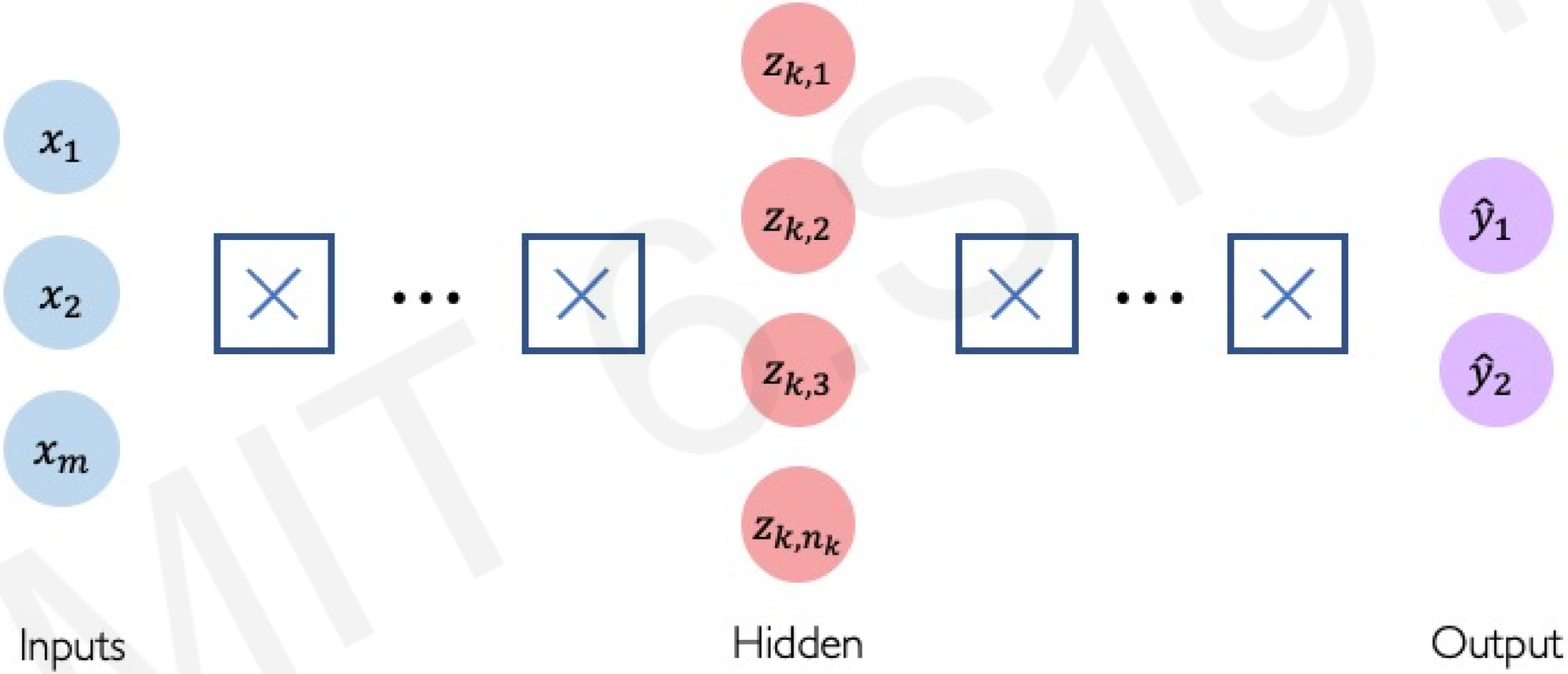
Multi Output Perceptron

```python
import tensorflow as tf

model = tf.keras.Sequential([
    tf.keras.layers.Dense(n),
    tf.keras.layers.Dense(2)
])
```



MIT 6.S191

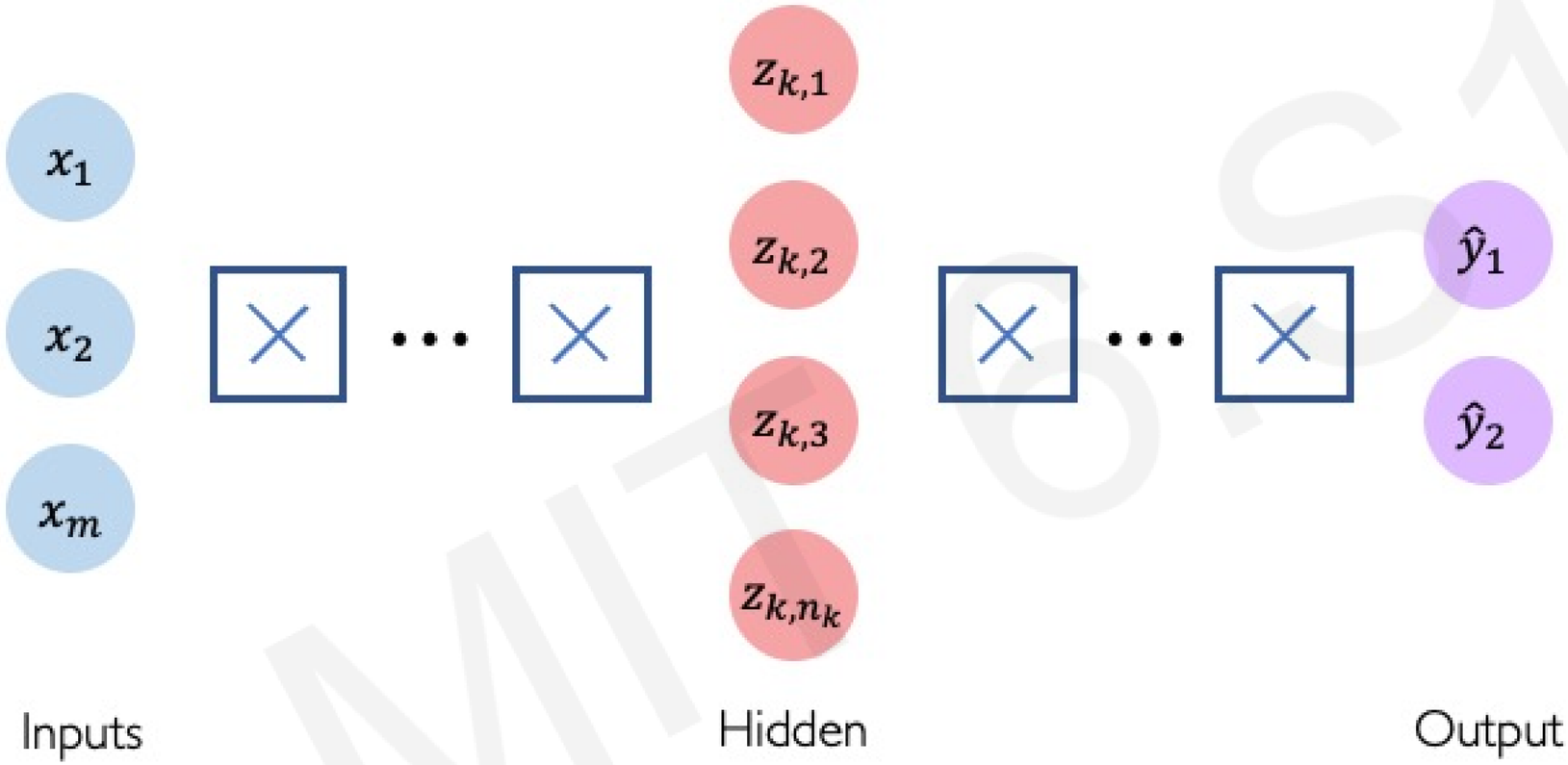
Deep Neural Network



$$z_{k,i} = w_{0,i}^{(k)} + \sum_{j=1}^{n_{k-1}} g(z_{k-1,j})\, w_{j,i}^{(k)}$$

MIT 6.S191

Deep Neural Network



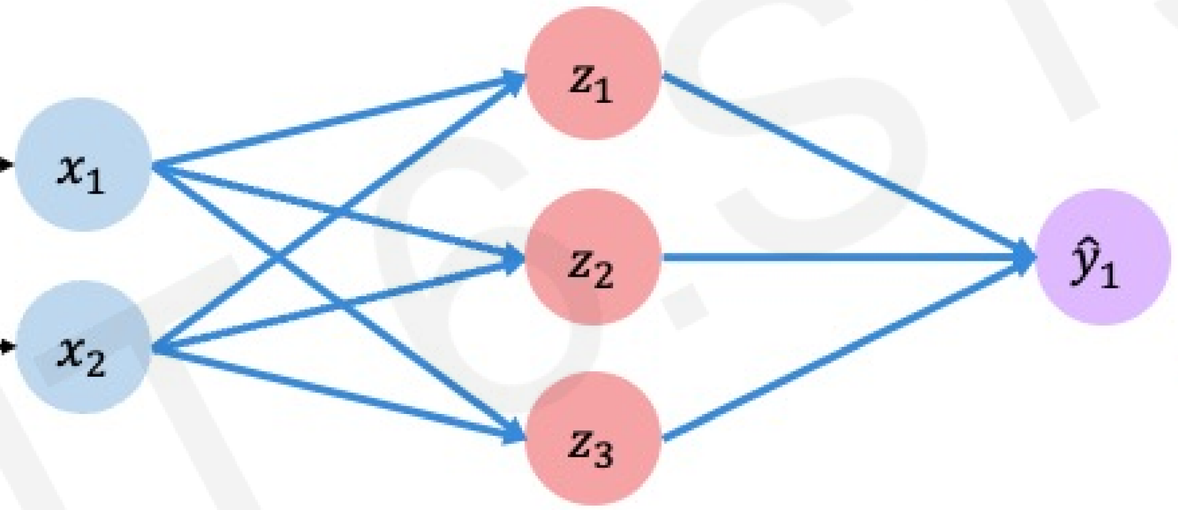

```
import tensorflow as tf

model = tf.keras.Sequential([
  tf.keras.layers.Dense(n1),
  tf.keras.layers.Dense(n2),
  ⋮
  tf.keras.layers.Dense(2)
])
```

$$z_{k,i} = w_{0,i}^{(k)} + \sum_{j=1}^{n_{k-1}} g(z_{k-1,j})\, w_{j,i}^{(k)}$$

MIT 6.S191

Example Problem: Will I pass this class?

$$x^{(1)} = [4\ ,5\,]$$



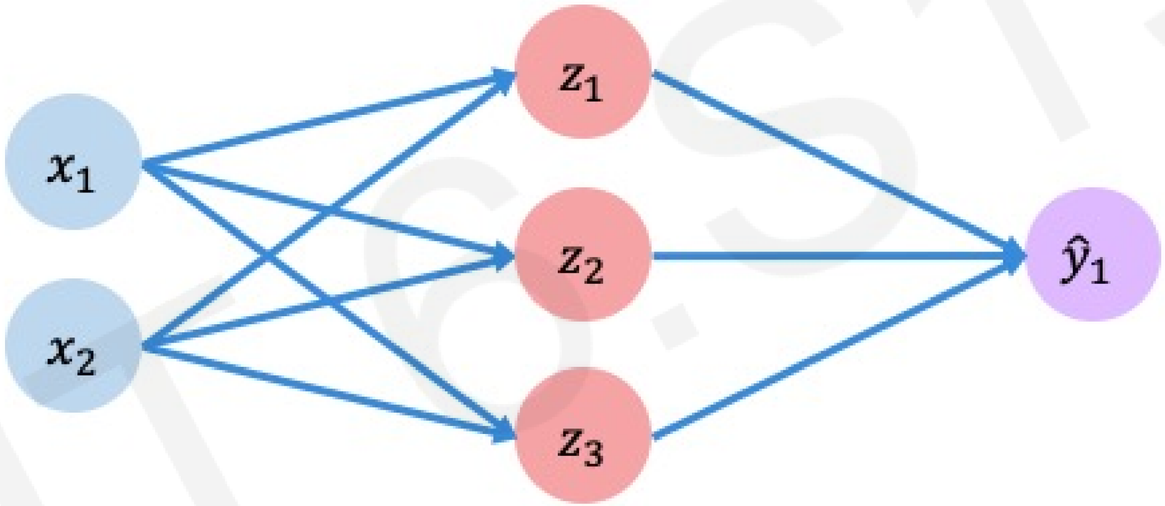
Predicted: 0.1
Actual: 1

MIT6.S191

Binary Cross Entropy Loss

Cross entropy loss *can be used with models that output a probability between 0 and 1*

$$\boldsymbol{X} = \begin{bmatrix} 4, & 5 \\ 2, & 1 \\ 5, & 8 \\ \vdots & \vdots \end{bmatrix}$$



$$\begin{array}{ccc} f(x) & & y \\ \begin{bmatrix} 0.1 \\ 0.8 \\ 0.6 \\ \vdots \end{bmatrix} & \begin{matrix} \times \\ \times \\ \checkmark \\ \end{matrix} & \begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \end{bmatrix} \end{array}$$

$$\boldsymbol{J}(\boldsymbol{W}) = -\frac{1}{n}\sum_{i=1}^{n} \underbrace{y^{(i)}}_{\text{Actual}} \log\left(\underbrace{f\left(x^{(i)};\boldsymbol{W}\right)}_{\text{Predicted}}\right) + (1 - \underbrace{y^{(i)}}_{\text{Actual}}) \log\left(1 - \underbrace{f\left(x^{(i)};\boldsymbol{W}\right)}_{\text{Predicted}}\right)$$

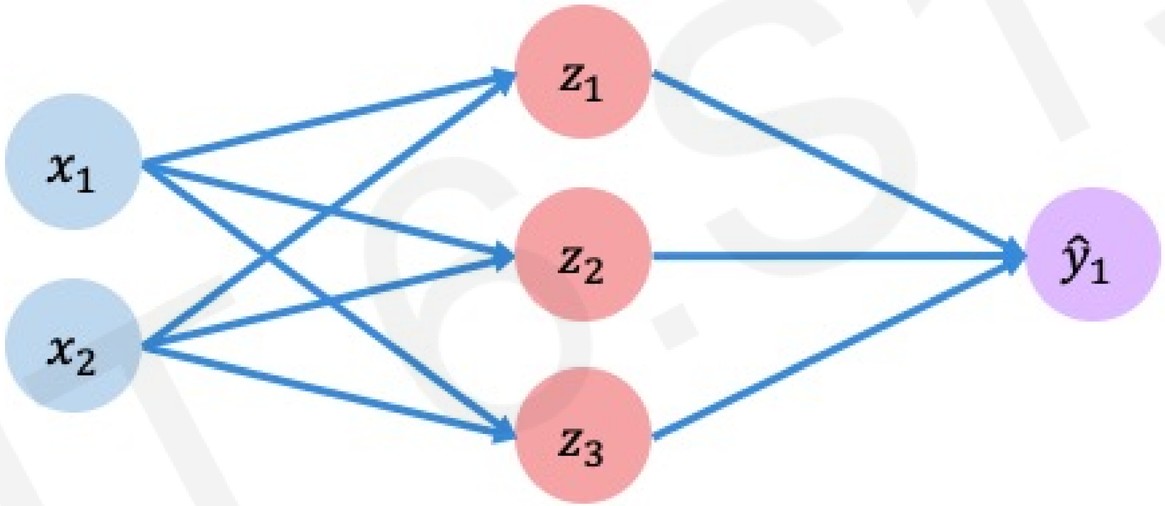
```
loss = tf.reduce_mean( tf.nn.softmax_cross_entropy_with_logits(y, predicted) )
```

MIT 6.S191

Mean Squared Error Loss

Mean squared error loss *can be used with regression models that output continuous real numbers*

$$\boldsymbol{X} = \begin{bmatrix} 4, & 5 \\ 2, & 1 \\ 5, & 8 \\ \vdots & \vdots \end{bmatrix}$$



$$\underbrace{\begin{array}{cc} f(x) & y \\ \begin{bmatrix} 30 \\ 80 \\ 85 \\ \vdots \end{bmatrix} & \begin{matrix} \times \\ \times \\ \checkmark \\ \end{matrix} \begin{bmatrix} 90 \\ 20 \\ 95 \\ \vdots \end{bmatrix} \end{array}}_{\text{Final Grades (percentage)}}$$

$$J(\boldsymbol{W}) = \frac{1}{n}\sum_{i=1}^{n}\left(\underbrace{y^{(i)}}_{\text{Actual}} - \underbrace{f\left(x^{(i)}; \boldsymbol{W}\right)}_{\text{Predicted}}\right)^2$$

```
loss = tf.reduce_mean( tf.square(tf.subtract(y, predicted)) )
loss = tf.keras.losses.MSE( y, predicted )
```

MIT 6.S191

Loss Optimization

We want to find the network weights that ***achieve the lowest loss***

$$\boldsymbol{W}^* = \underset{\boldsymbol{W}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}\left(f\left(x^{(i)}; \boldsymbol{W}\right), y^{(i)}\right)$$

$$\boldsymbol{W}^* = \underset{\boldsymbol{W}}{\operatorname{argmin}} J(\boldsymbol{W})$$

Remember:

$$\boldsymbol{W} = \left\{\boldsymbol{W}^{(0)}, \boldsymbol{W}^{(1)}, \dots\right\}$$

MIT 6.S191

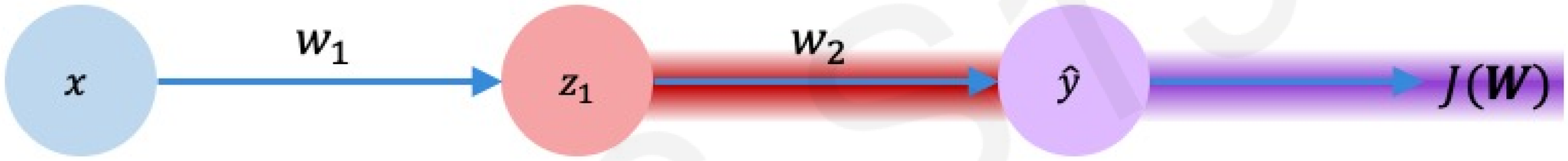
Computing Gradients: Backpropagation

$$\frac{\partial J(\boldsymbol{W})}{\partial w_2} =$$

Let's use the chain rule!

MIT 6.S191

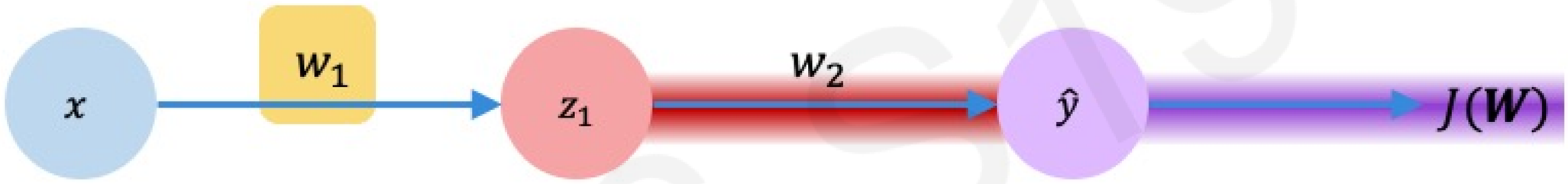
Computing Gradients: Backpropagation



$$\frac{\partial J(\boldsymbol{W})}{\partial w_2} = \frac{\partial J(\boldsymbol{W})}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial w_2}$$

MIT 6.S191

Computing Gradients: Backpropagation



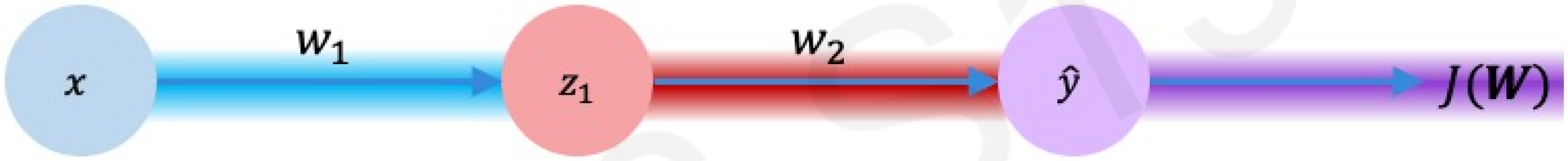
$$\frac{\partial J(\boldsymbol{W})}{\partial w_1} = \frac{\partial J(\boldsymbol{W})}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial w_1}$$

Apply chain rule!

Apply chain rule!

MIT 6.S191

Computing Gradients: Backpropagation

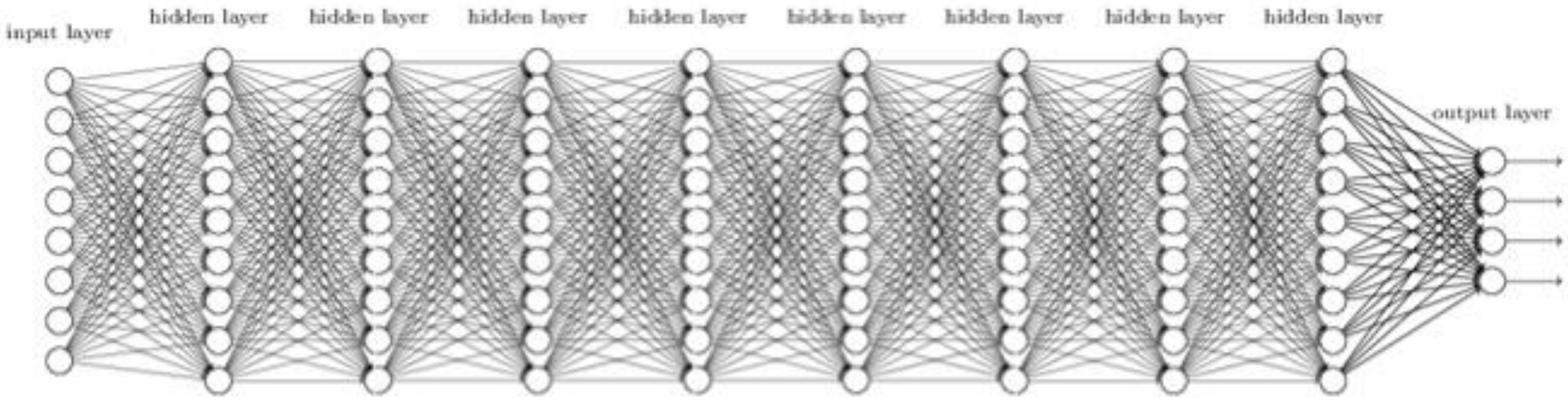


$$\frac{\partial J(\boldsymbol{W})}{\partial w_1} = \frac{\partial J(\boldsymbol{W})}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial z_1} * \frac{\partial z_1}{\partial w_1}$$

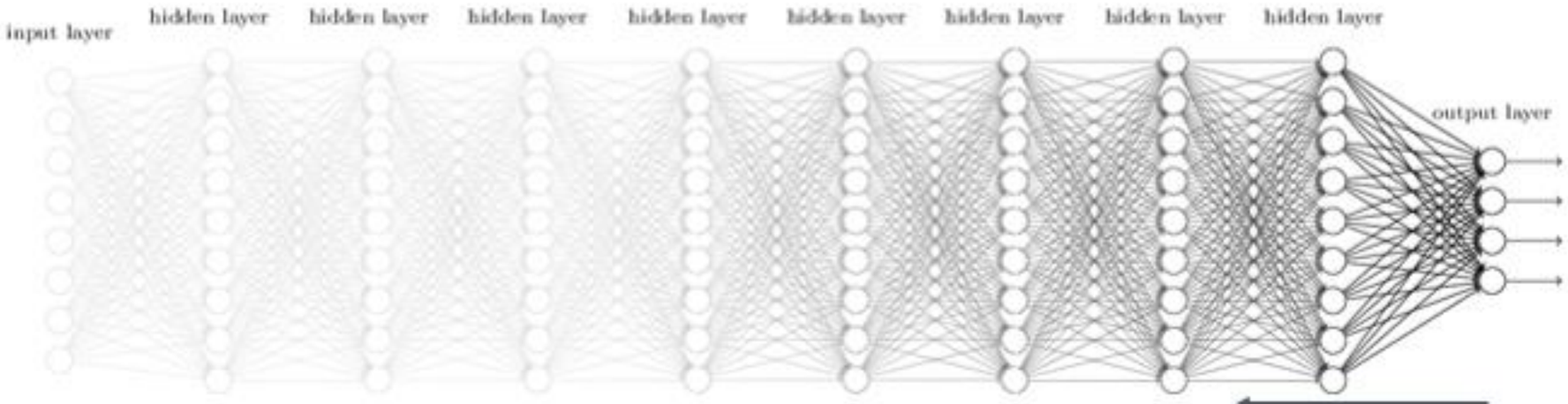
Repeat this for ***every weight in the network*** *using gradients from later layers*

MIT 6.S191

Fit하는 과정에서 학습이 잘 안되는 경우가 있다!

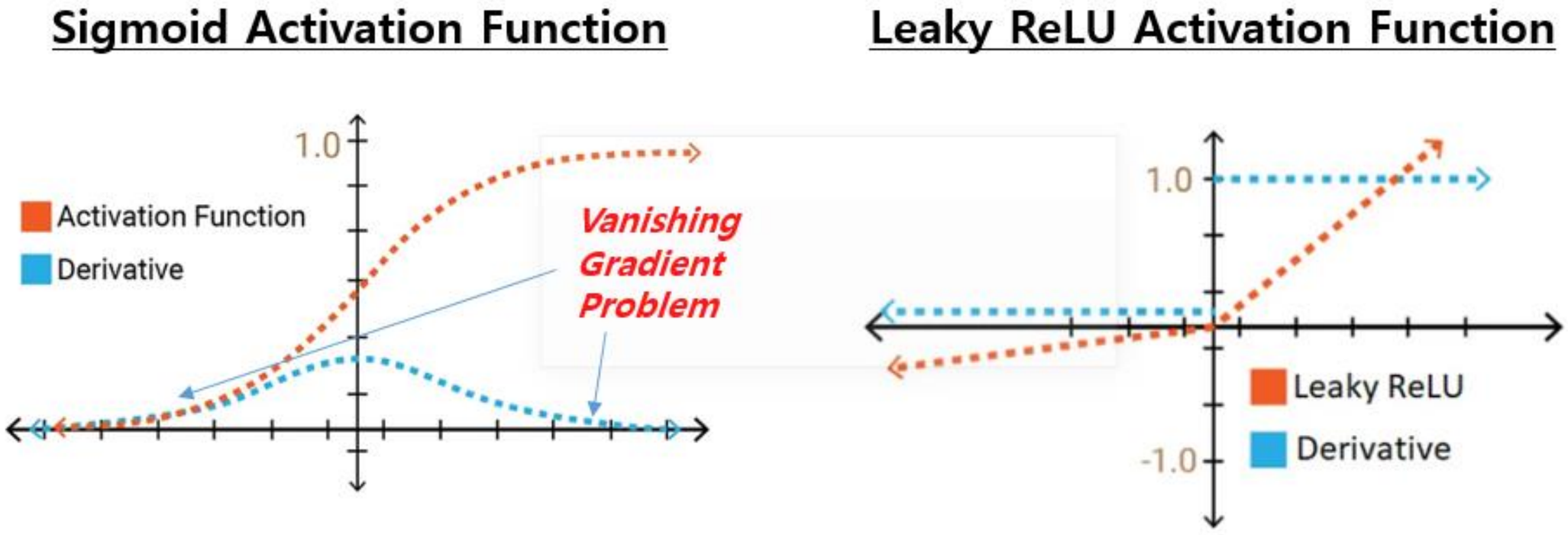


Deep Neural Network



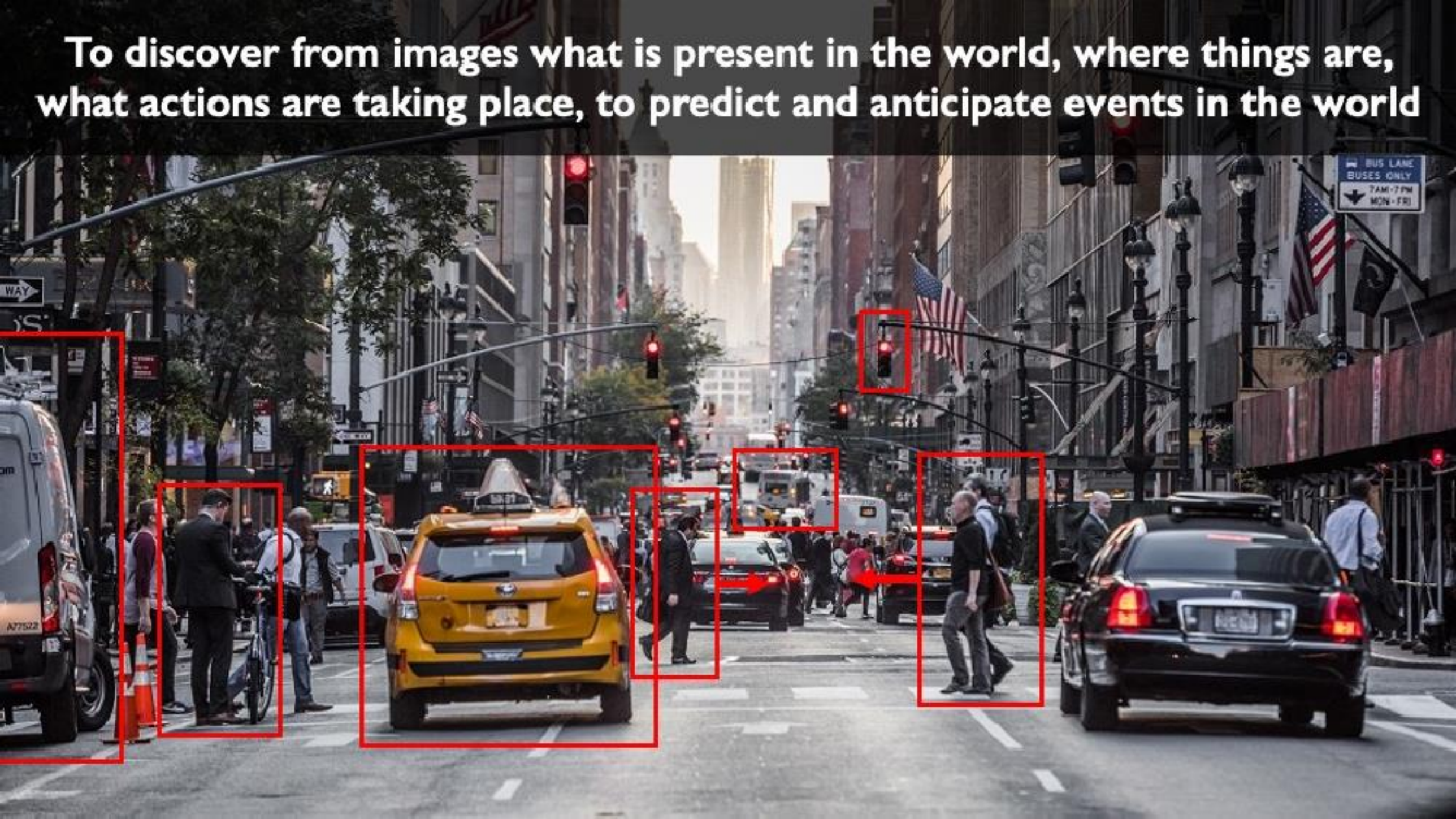
Vanishing Gradient

반영이 되는 미분의 값을 확인해보자!



* source: https://mc.ai/comparison-of-activation-functions-for-deep-neural-networks/

To discover from images what is present in the world, where things are, what actions are taking place, to predict and anticipate events in the world

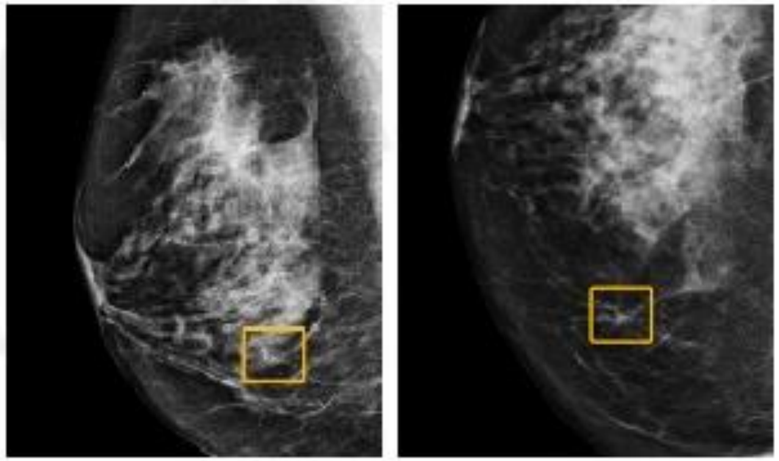


The rise and impact of computer vision

Robotics

Accessibility

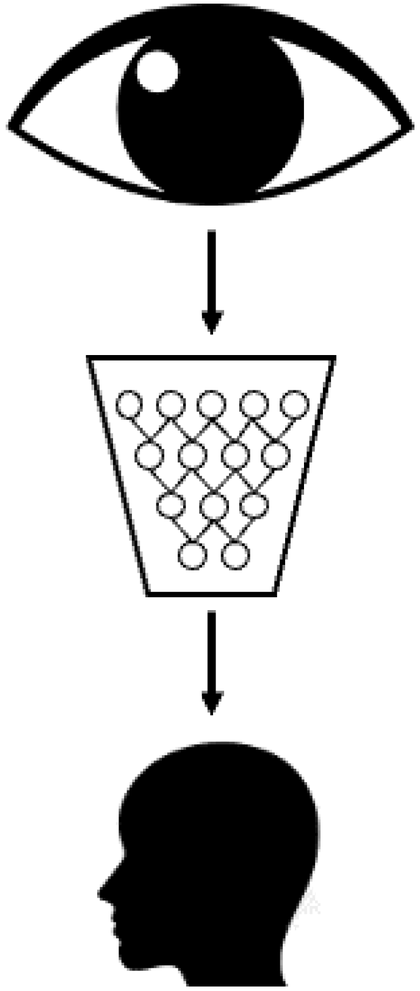
Biology & Medicine

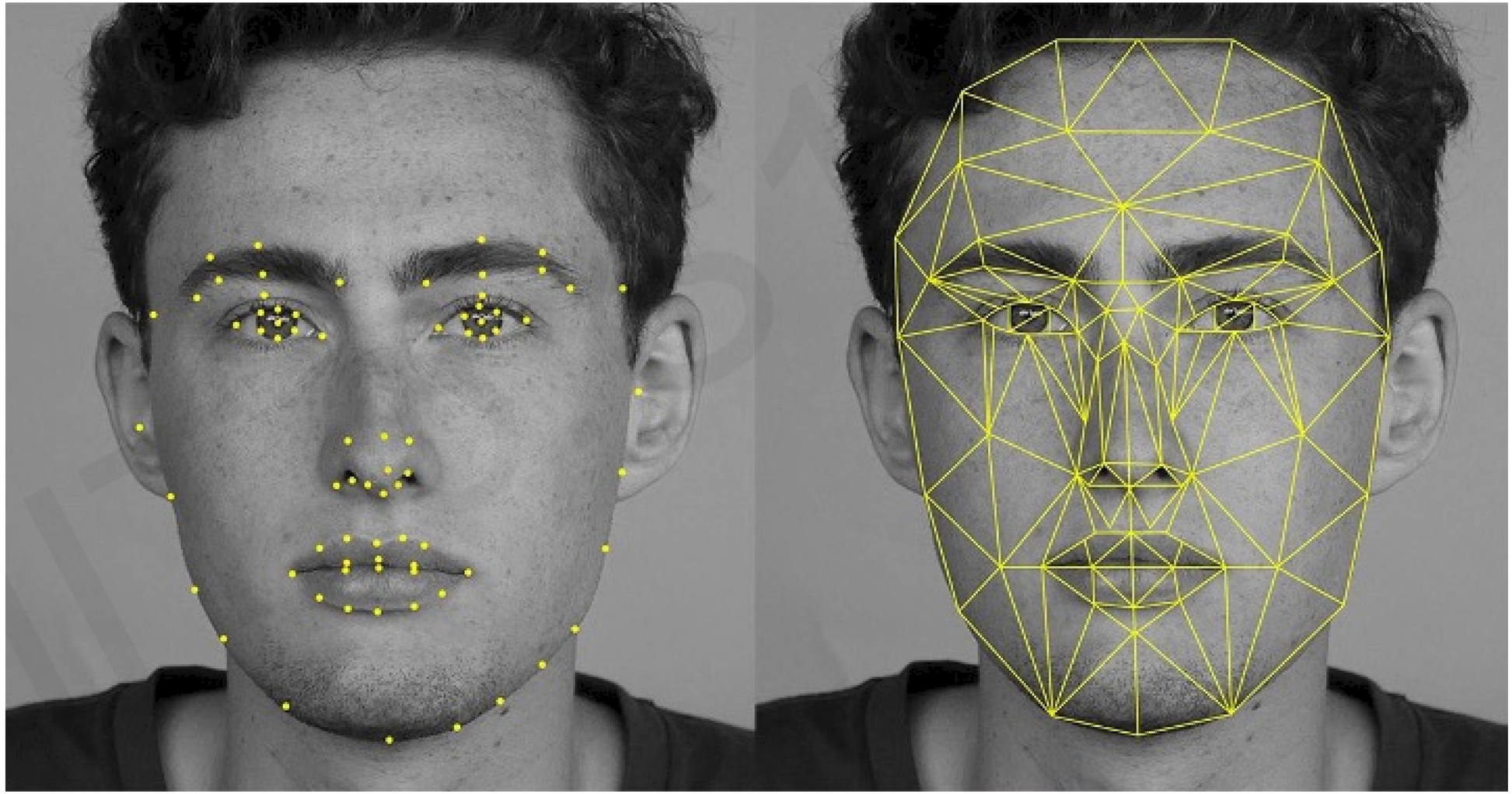


Autonomous driving

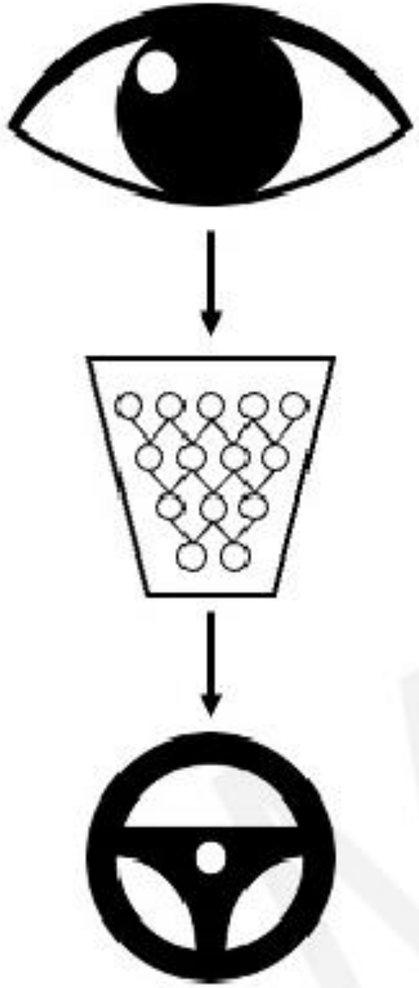
Mobile computing

Impact: Facial Detection & Recognition



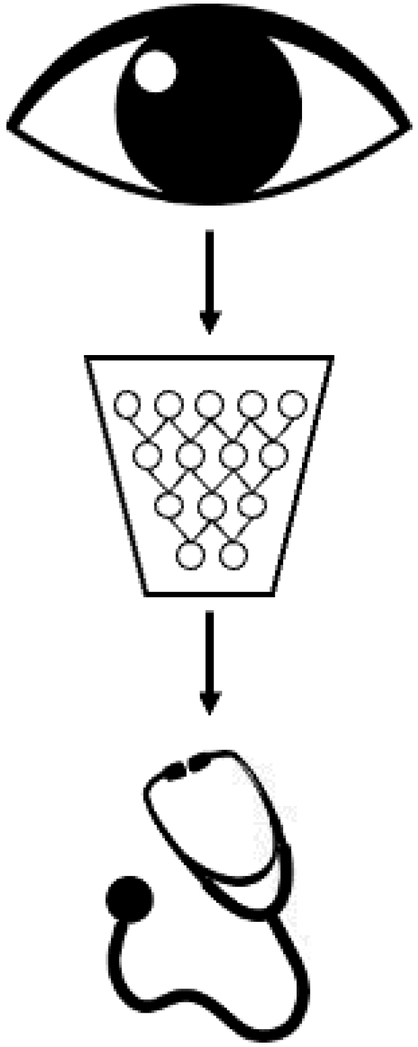


Impact: Self-Driving Cars

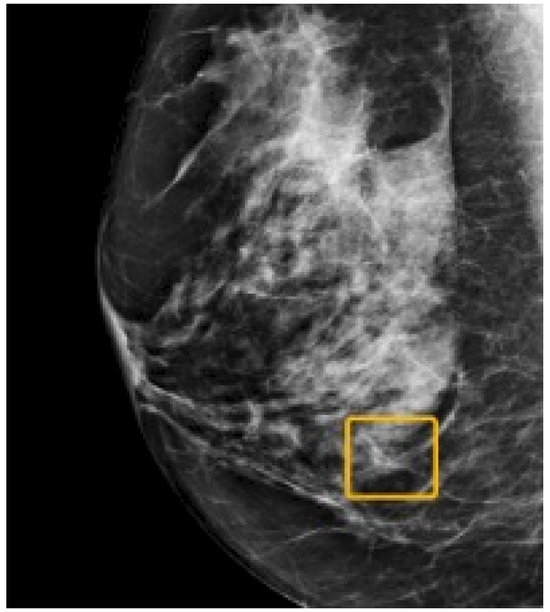


Amini+ *ICRA* 2019.

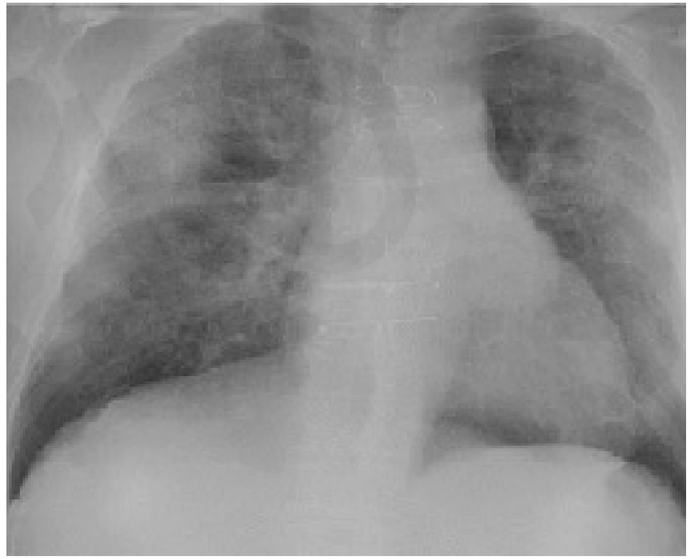
Impact Medicine, Biology, Healthcare



Breast cancer



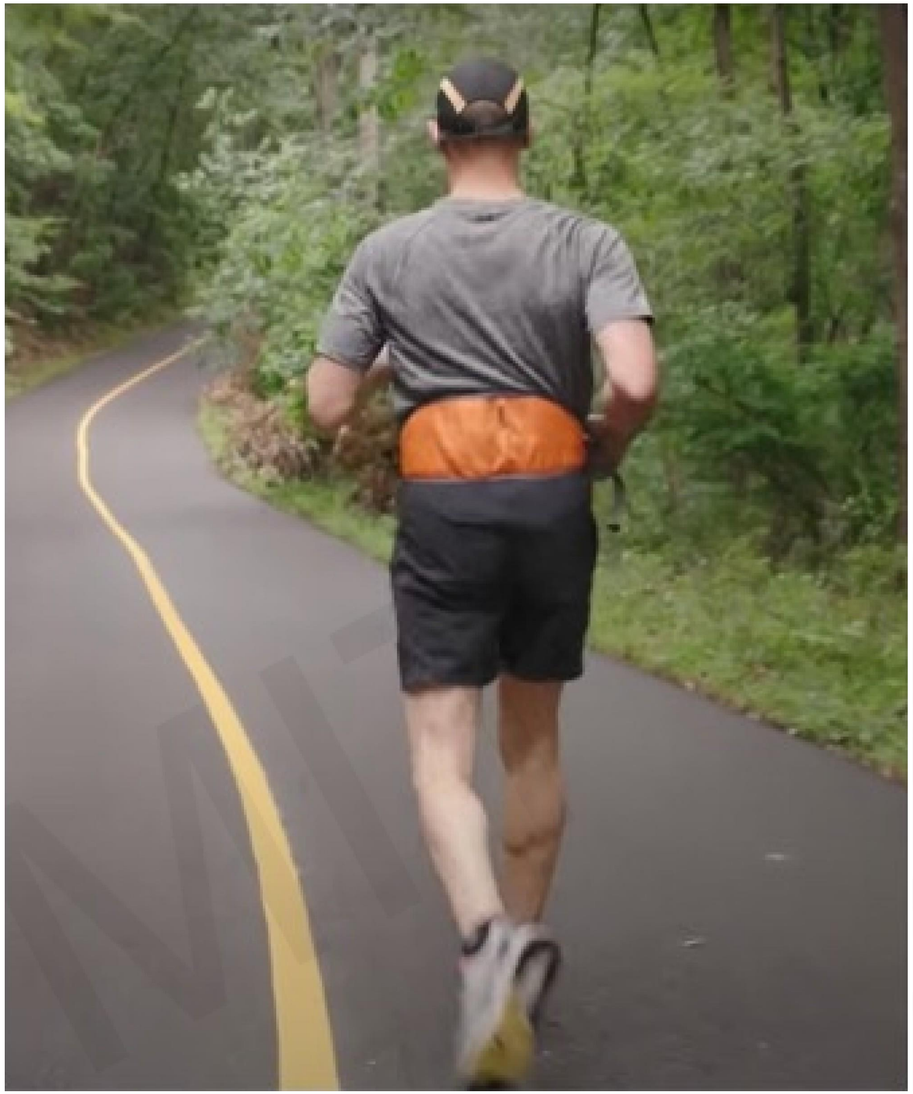
COVID-19

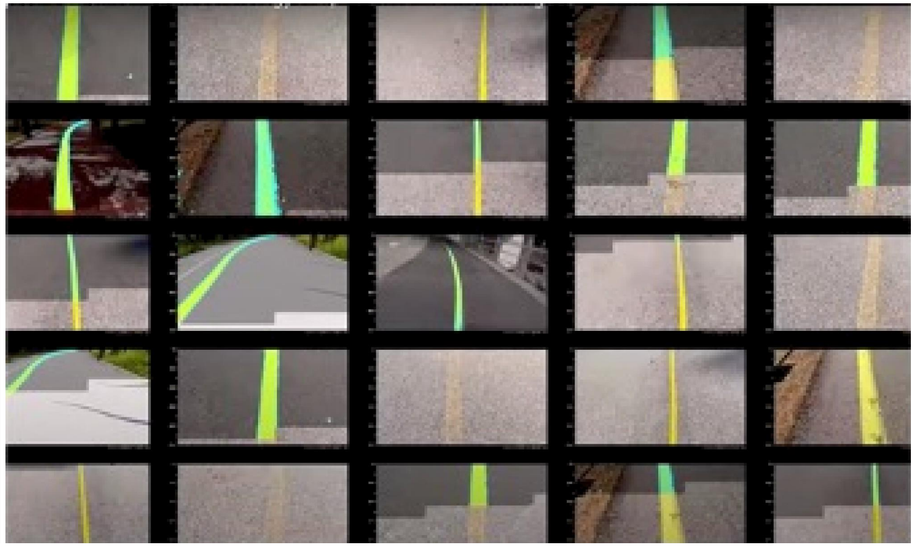


Skin cancer

Esteva+ 2017, McKinney+ *Nature* 2020, Wang+ Sci. 2020.

Impact: Accessibility

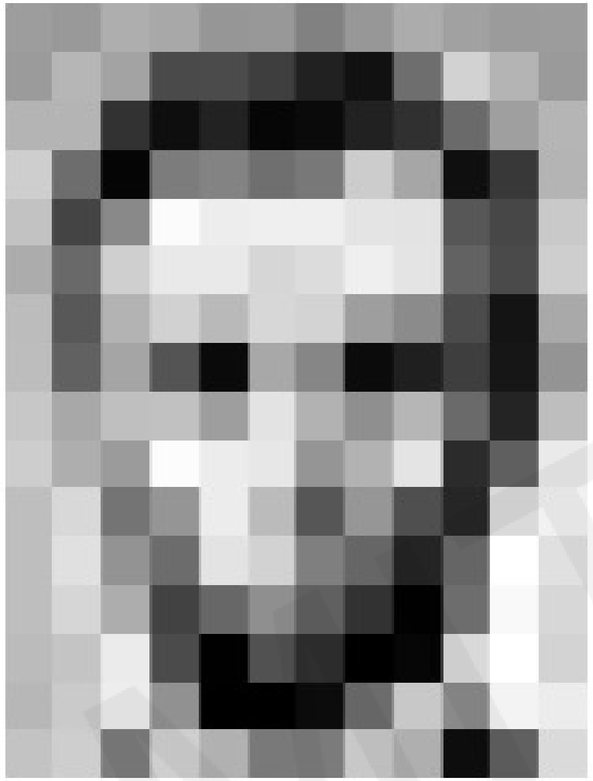




What Computers "See"

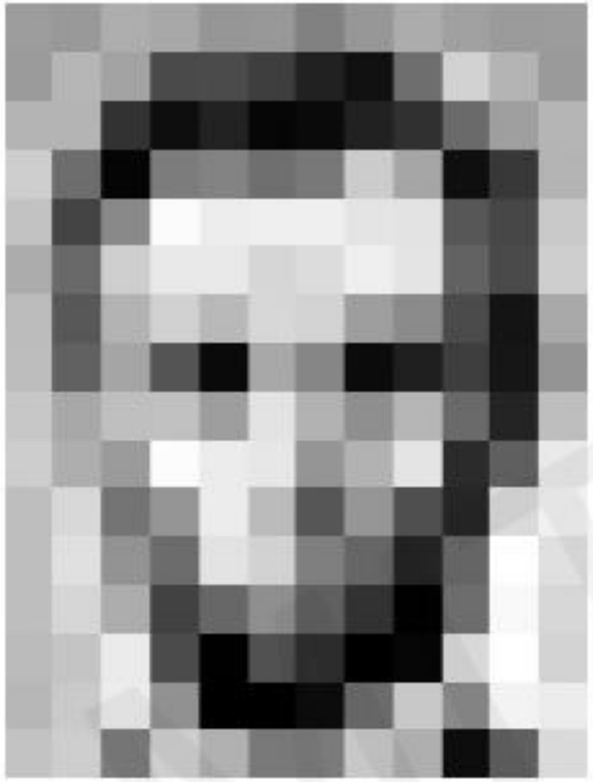
MIT 6.S191

Images are Numbers



MIT 6.S191

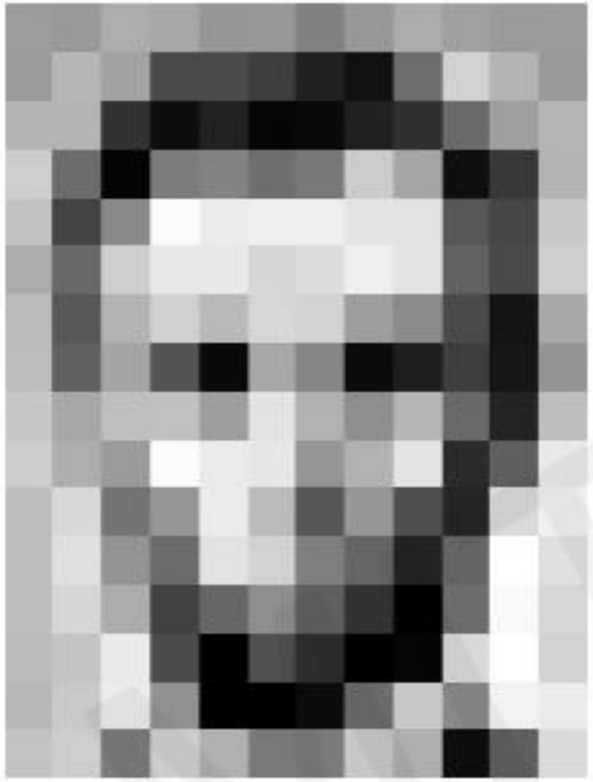
Images are Numbers



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	96	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

MIT 191

Images are Numbers



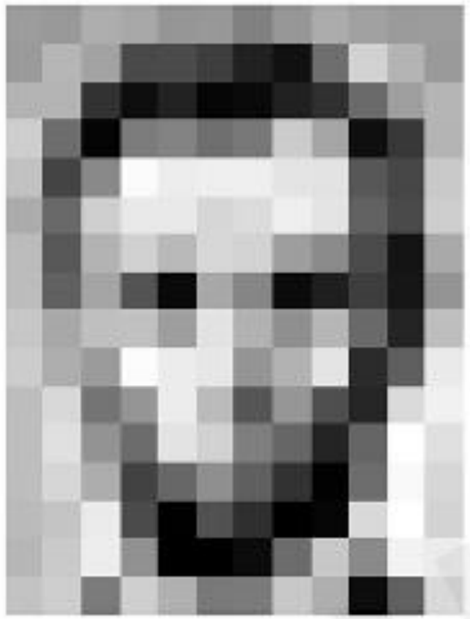
157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	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	96	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

What the computer sees

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	96	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

An image is just a matrix of numbers [0,255]!
i.e., 1080x1080x3 for an RGB image

Tasks in Computer Vision



Input Image

167	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	96	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

Pixel Representation

classification

Lincoln	0.8
Washington	0.1
Jefferson	0.05
Obama	0.05

- **Regression**: output variable takes continuous value
- **Classification**: output variable takes class label. Can produce probability of belonging to a particular class

MIT 6S191

High Level Feature Detection

Let's identify key features in each image category

Nose,
Eyes,
Mouth

Wheels

Door,
Windows
Steps

Manual Feature Extraction

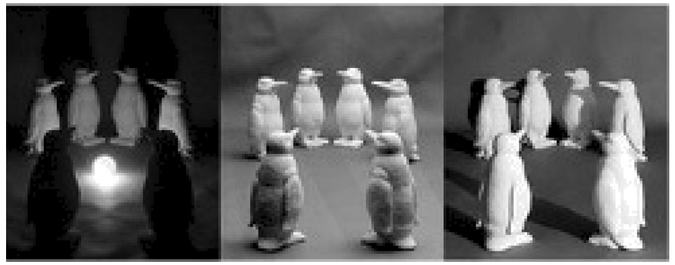
Problems?

Manual Feature Extraction

Domain knowledge → Define features → Detect features to classify

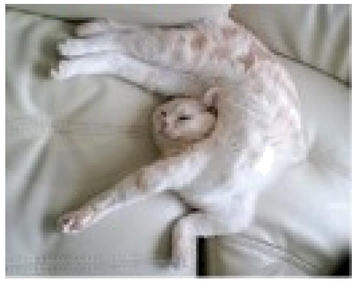
lew point variation

lumination œnditions

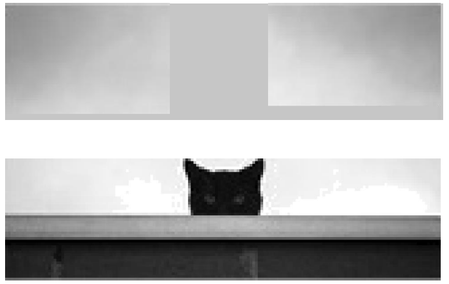


Smle variation

Oefowation

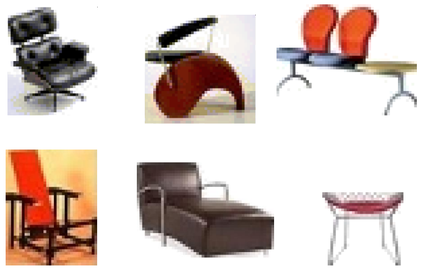


Occlusion



Bac@round clutter

Intra-class variation



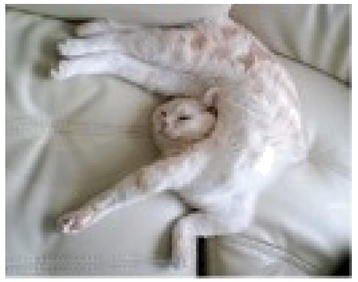
Manual Feature Extraction

Detect features to classify

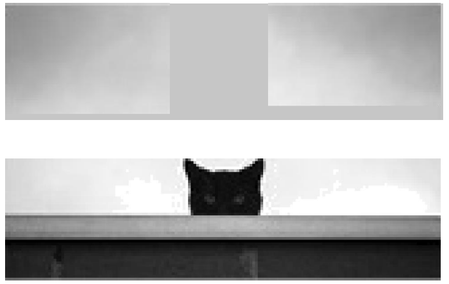
lew point variation

Smle variation

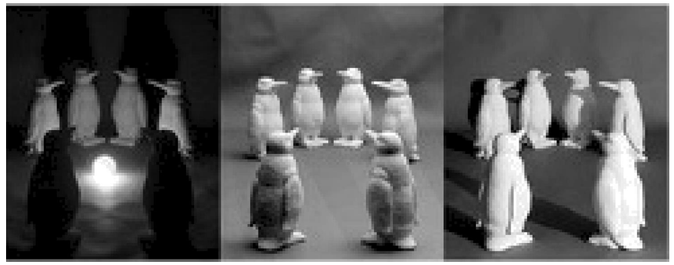
Oefowation



Occlusion

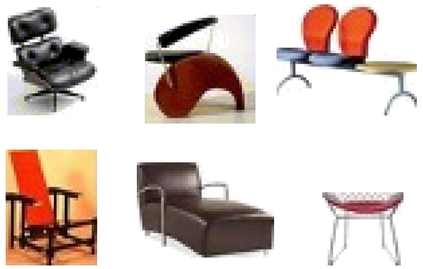


lumination œnditions



Bac@round clutter

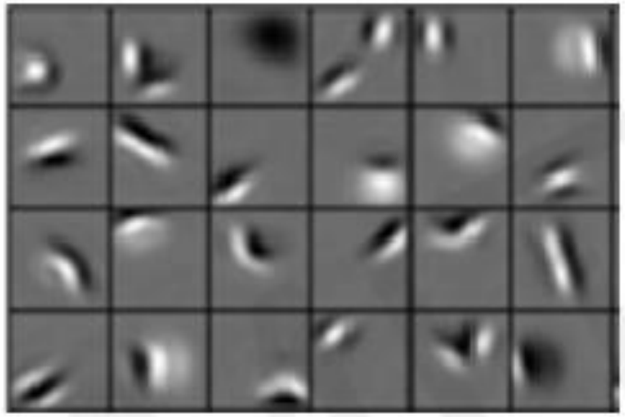
Intra-class variation



Learning Feature Representations

Can we learn a **hierarchy of features** directly from the data instead of hand engineering?

Low level features



Edges, dark spots

Mid level features

Eyes, ears, nose

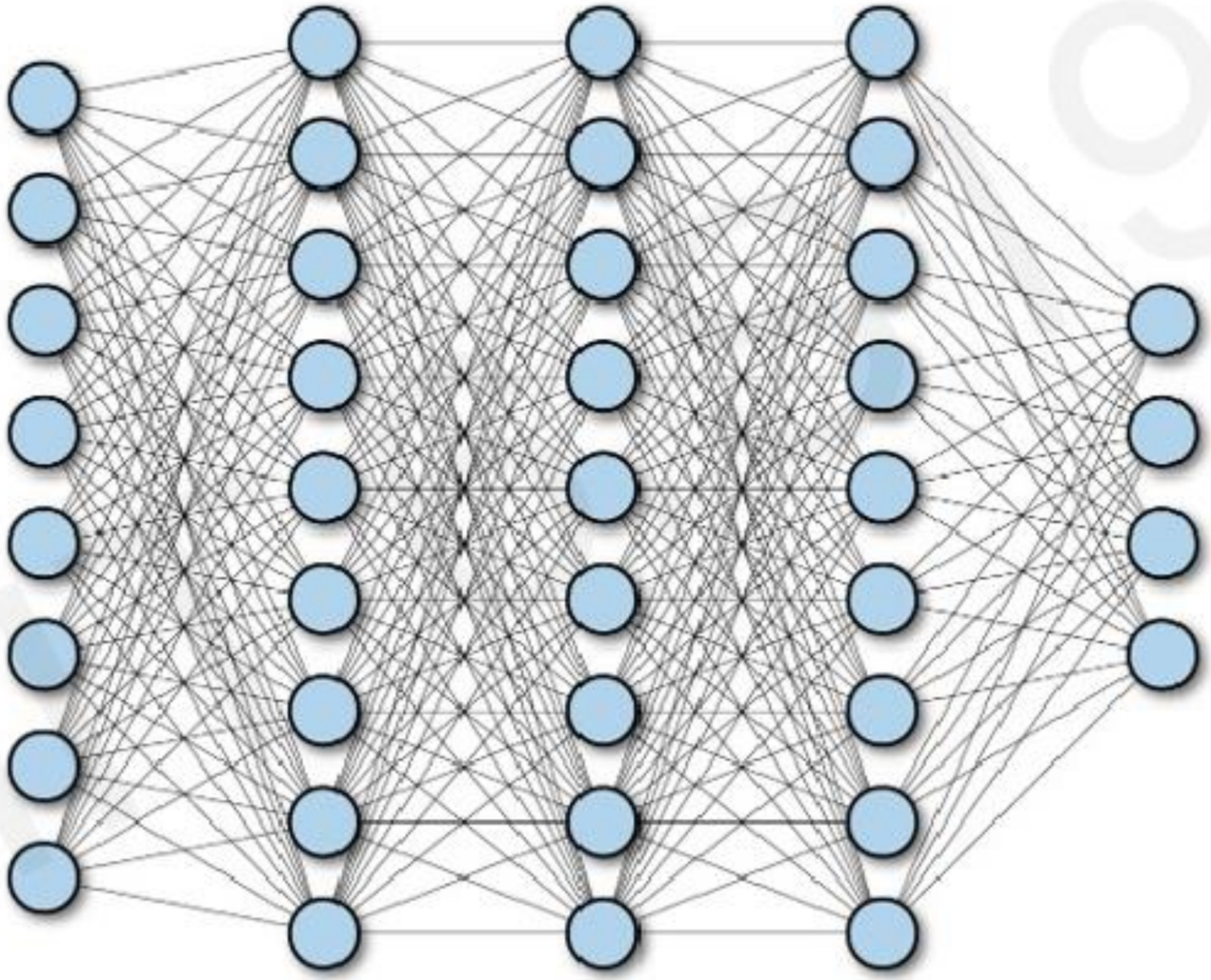
High level features

Facial structure

Learning Visual Features

MIT 6.S191

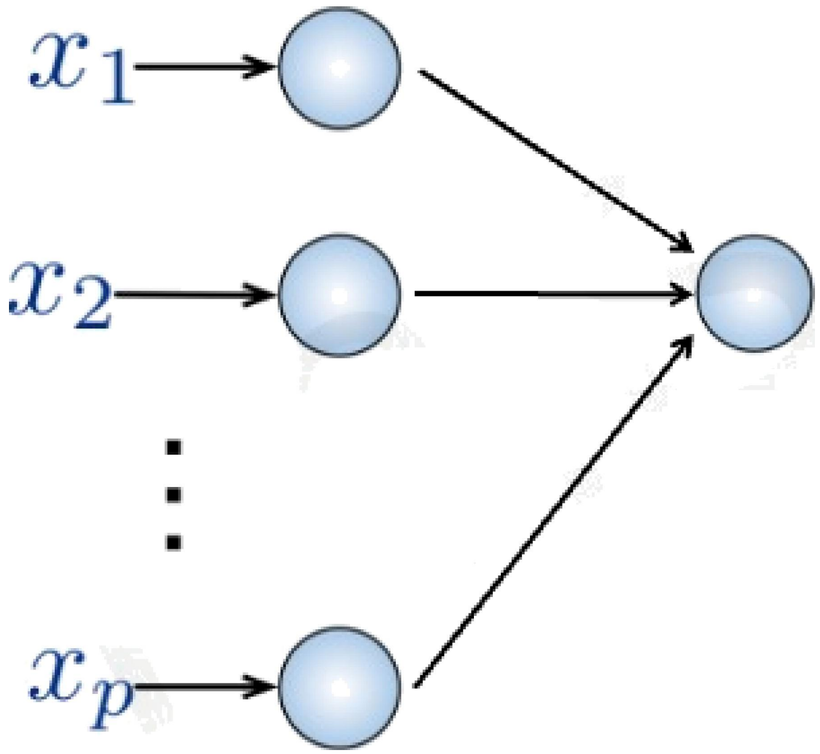
Fully Connected Neural Network



Fully Connected Neural Network

Input:

- 2D image
- Vector of pixel values



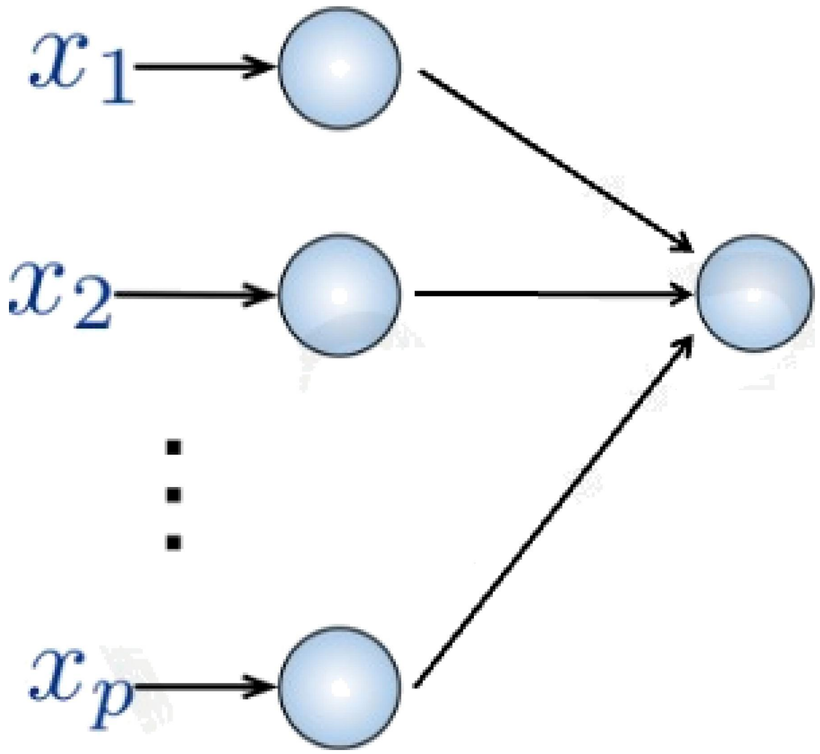
Fully Connecced:

- Connect neuron in hidden layer to all neurons in input layer
- No spahal informa‘tion!
- And many, many parameters!

Fully Connected Neural Network

Input:

- 2D image
- Vector of pixel values



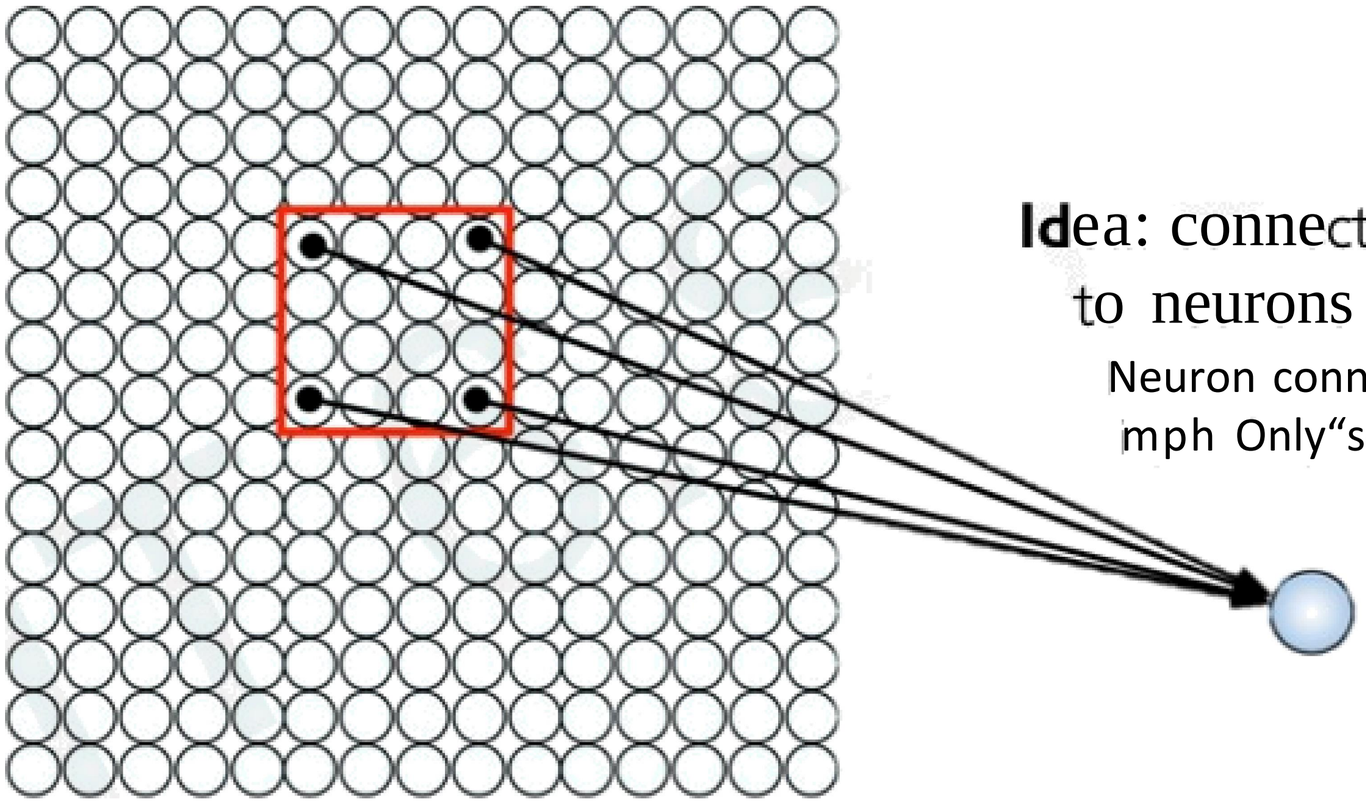
Fully Connected:

- Connect neuron in hidden layer to all neurons in input layer
- No spatial information!
- And many, many parameters!

How can we use **spatial structure** in the input to inform the architecture of the network?

Using Spatial Structure

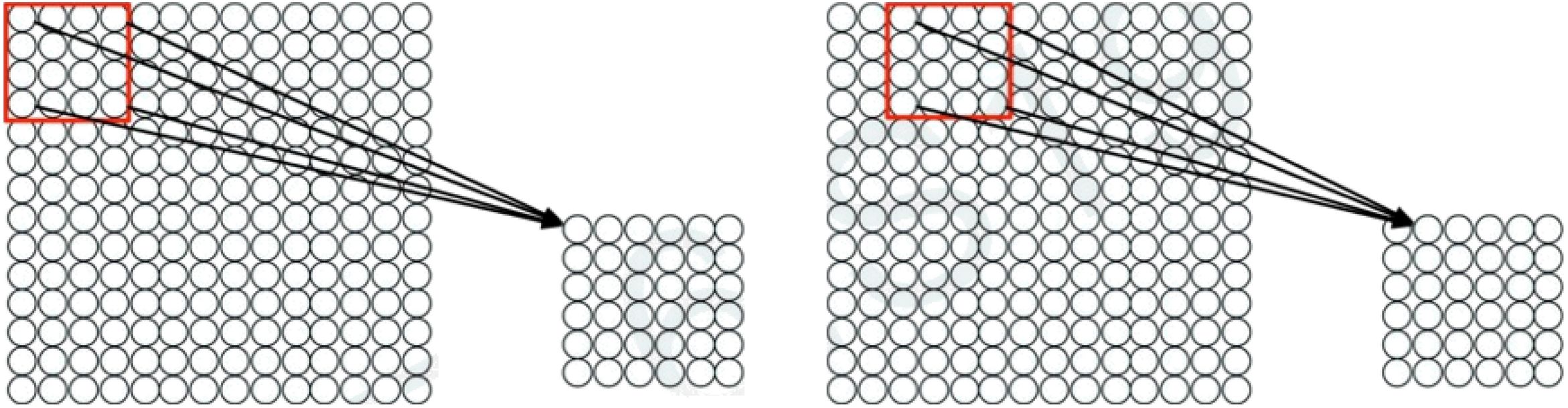
Input: 2D image.
Array of pixel values



Idea: connect patches of input to neurons in hidden layer

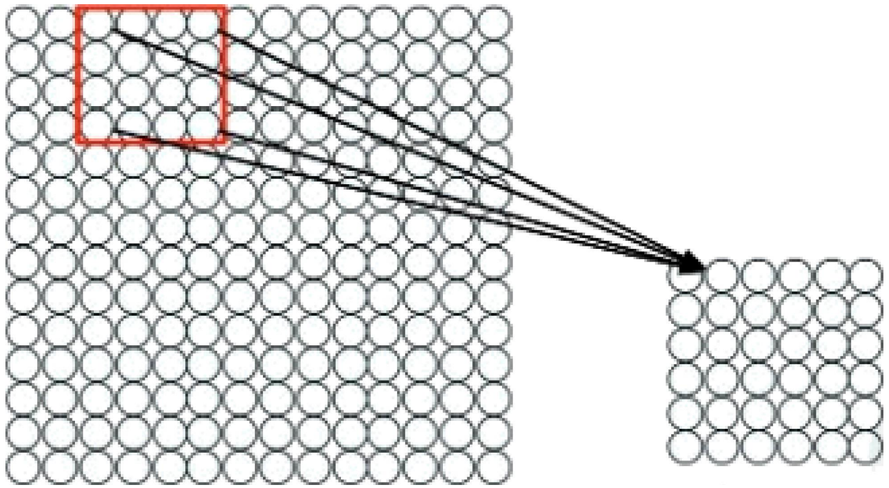
Neuron conneked to Æğ¡OÜ 0(mph Only“sees”these ælues.

Using Spatial Structure



Connect palch in input layer to a single neuron in subsequent layer.
Use a sliding window to defme connections.
How con we wei**ˆght** *the patch to dete*O *particular f*•eoturesŽ

Feature Extraction with Convolution



- Filter of size 4x4 : i 6 diffemnt weights
- Apply this same filter to 4x4 patches in input
- Shift by 2 pixe'is for next paich

This "patchy" opera'tion is **convolution**

1) Apply a set of weights – a filter – to extract **local** featu **res**

2) Use mu**ltiple filt**⊖○ to extract different fea'tures

3) **Spatially share** parameters of each filter

Feature Extraction and Convolution

A Case Study

MIT 6.S191

X or X?

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

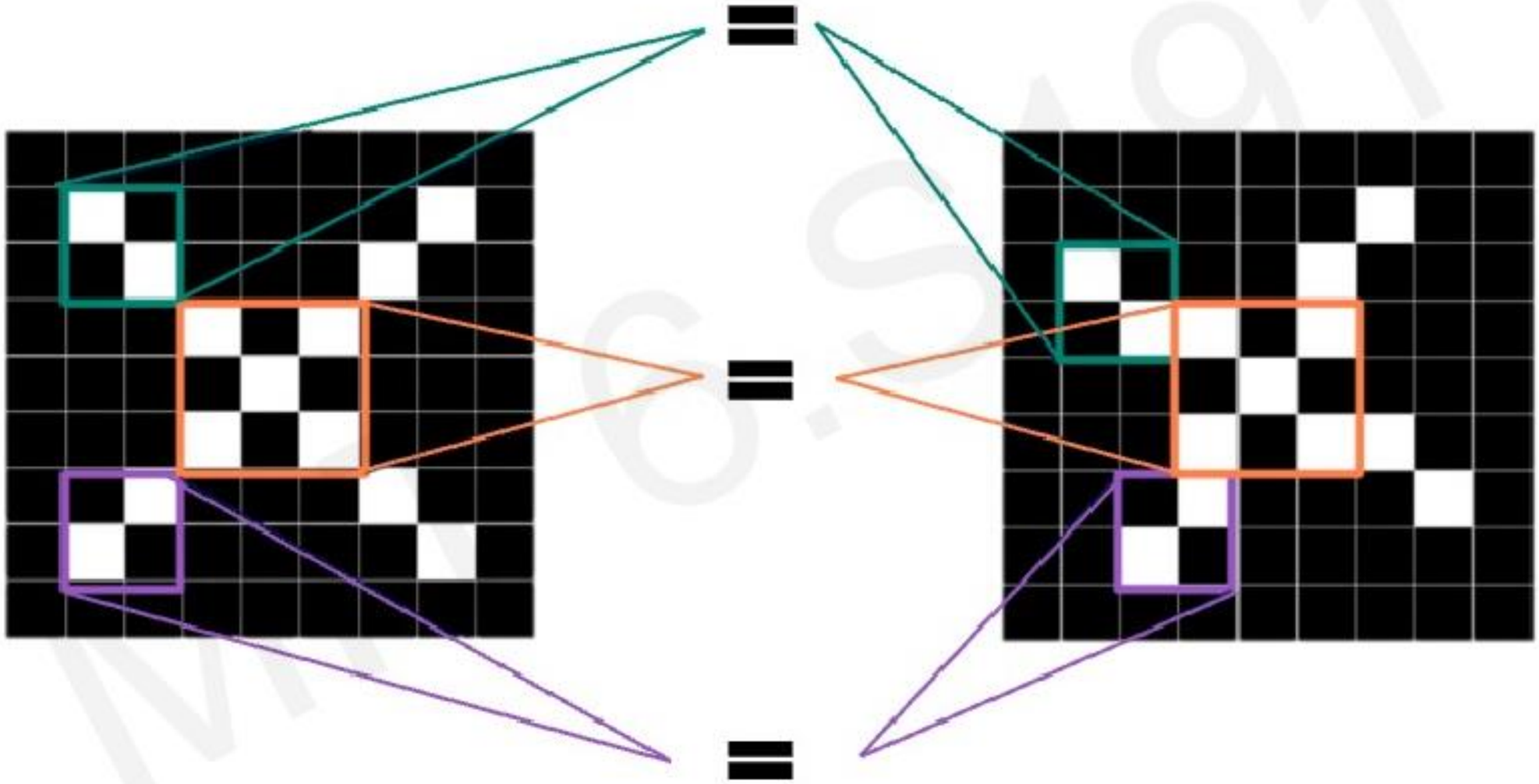
?

=

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

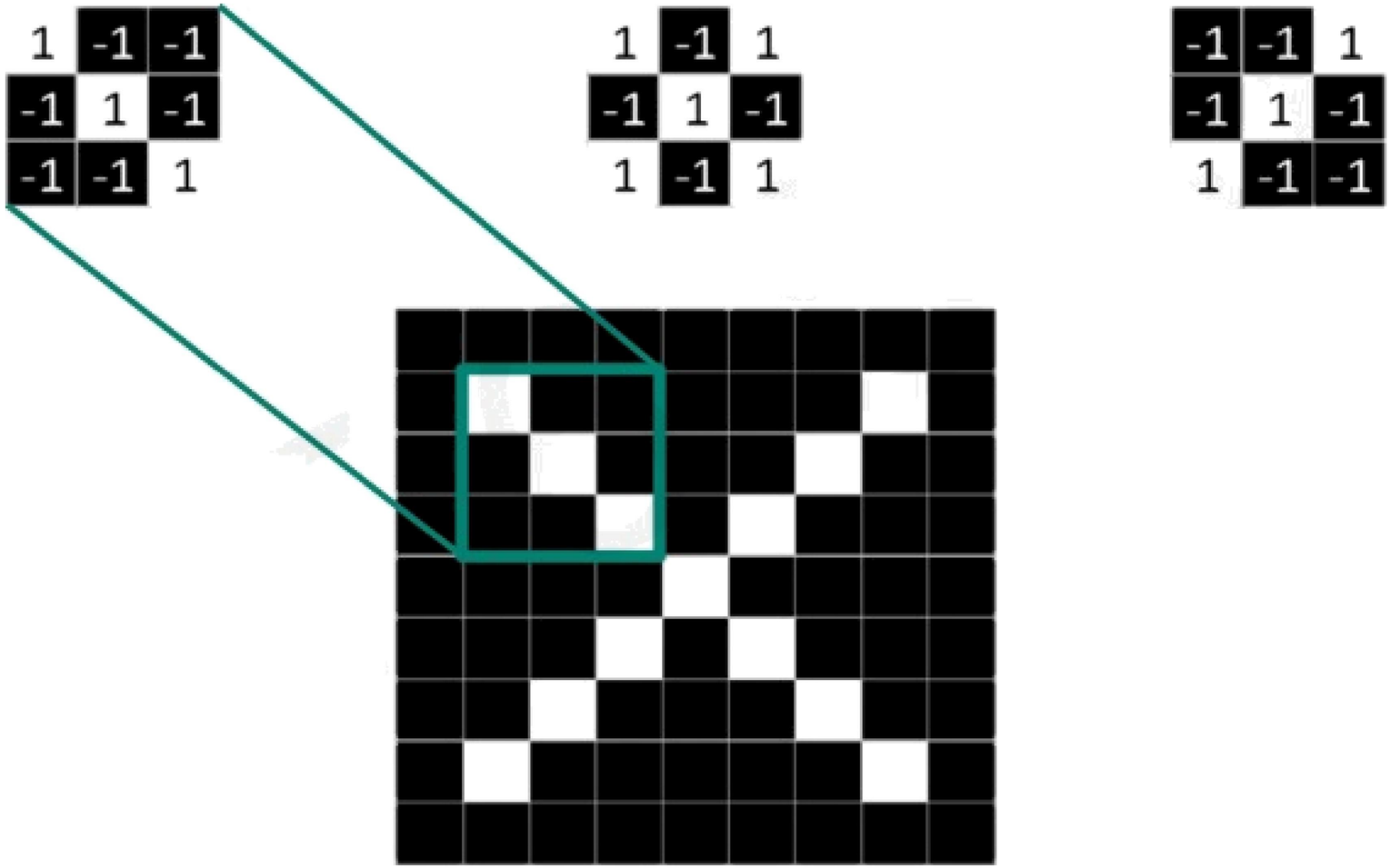
Image Is repmsented as matrix of pixel values . and computers are lùeral!
We want to be able to classify an X as an X even if it's shifted, shrunk, rotated, deformed.

Features of X

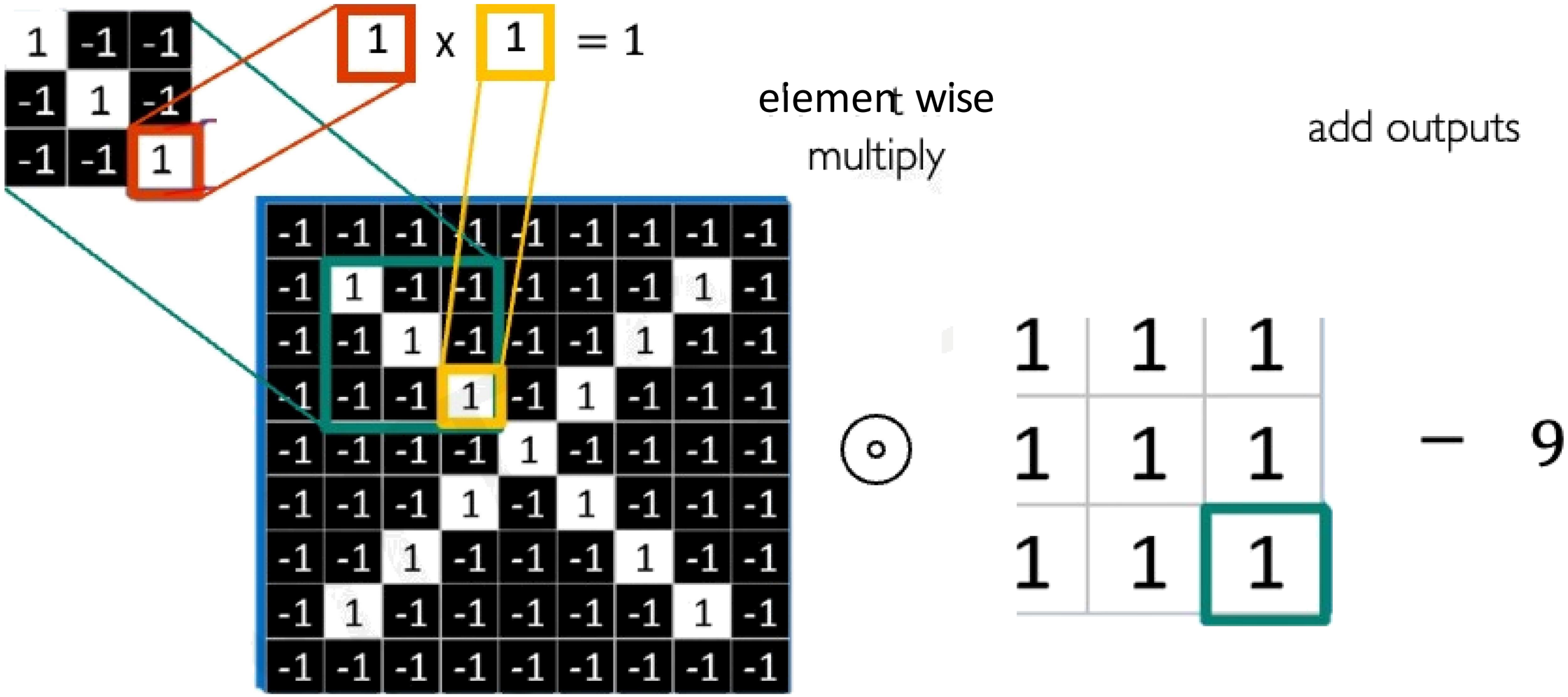


Filters to Detect X Features

filters



The Convolution Operation



The Convolution Operation

Suppose we want to compute the convolution of a 5x5 image and a 3x3 filter:

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

image

$\otimes$

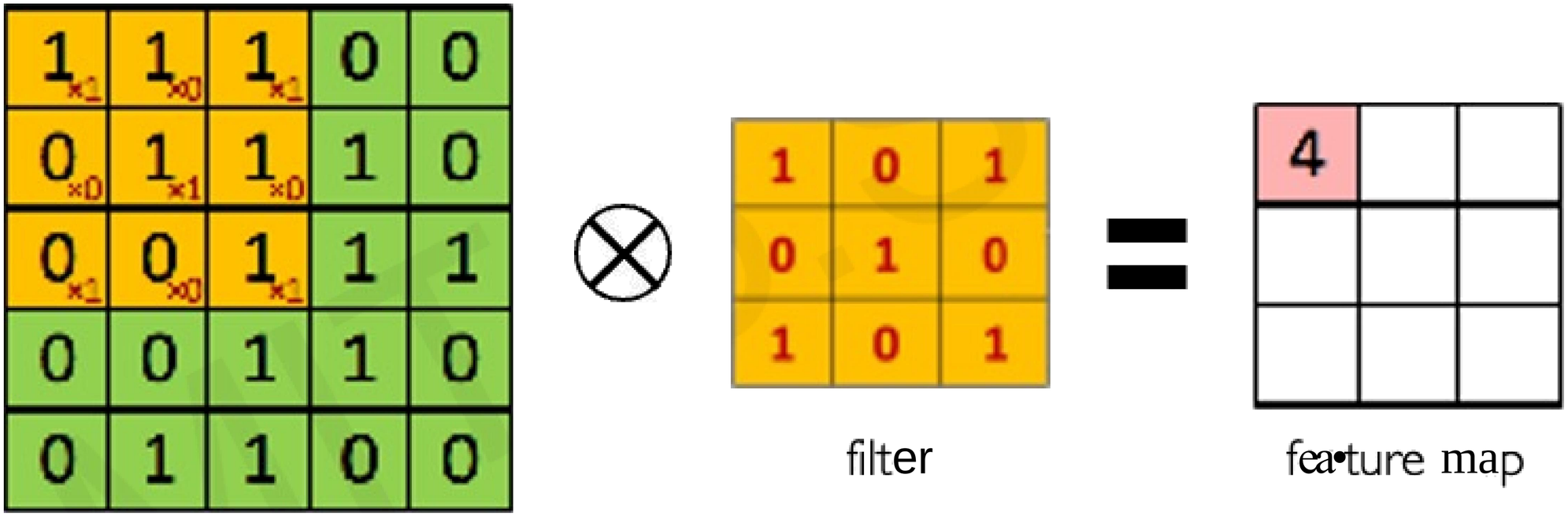
1	0	1
0	1	0
1	0	1

filter

We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs

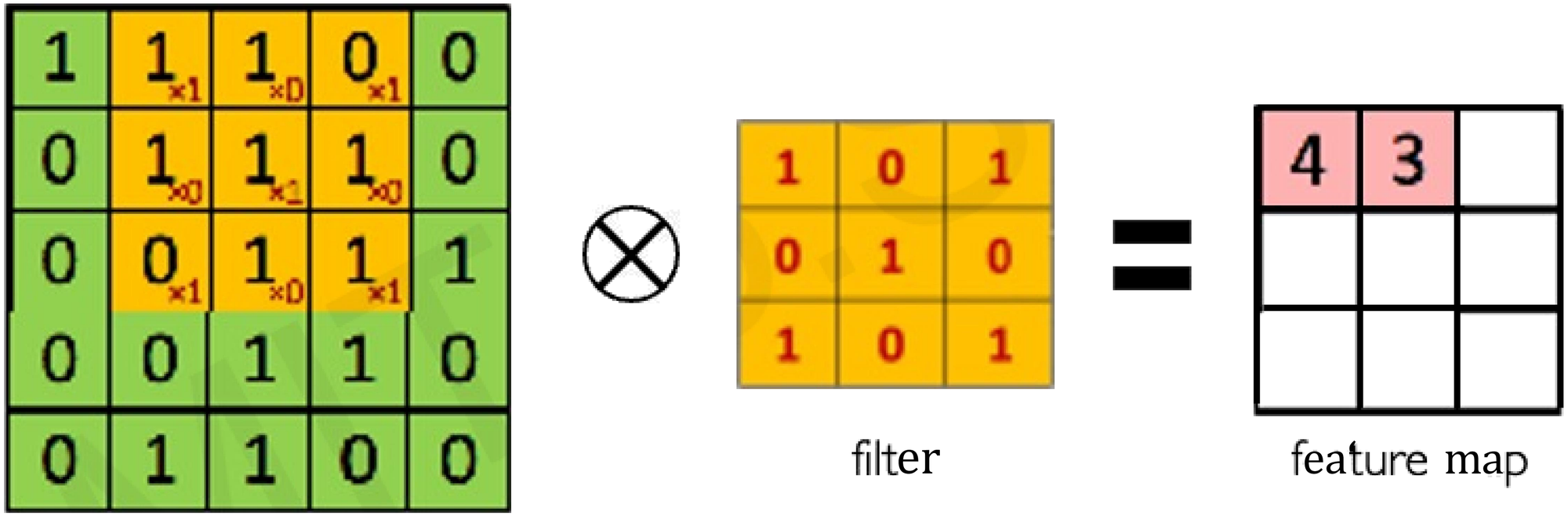
The Convolution Operation

We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs



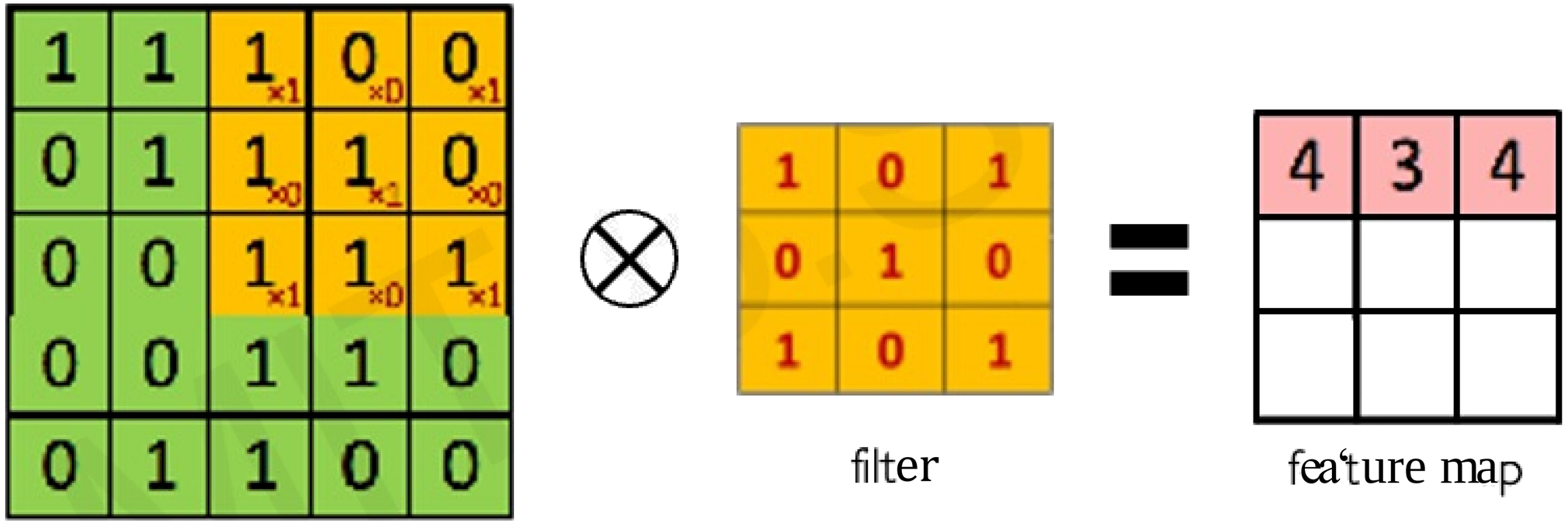
The Convolution Operation

We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs



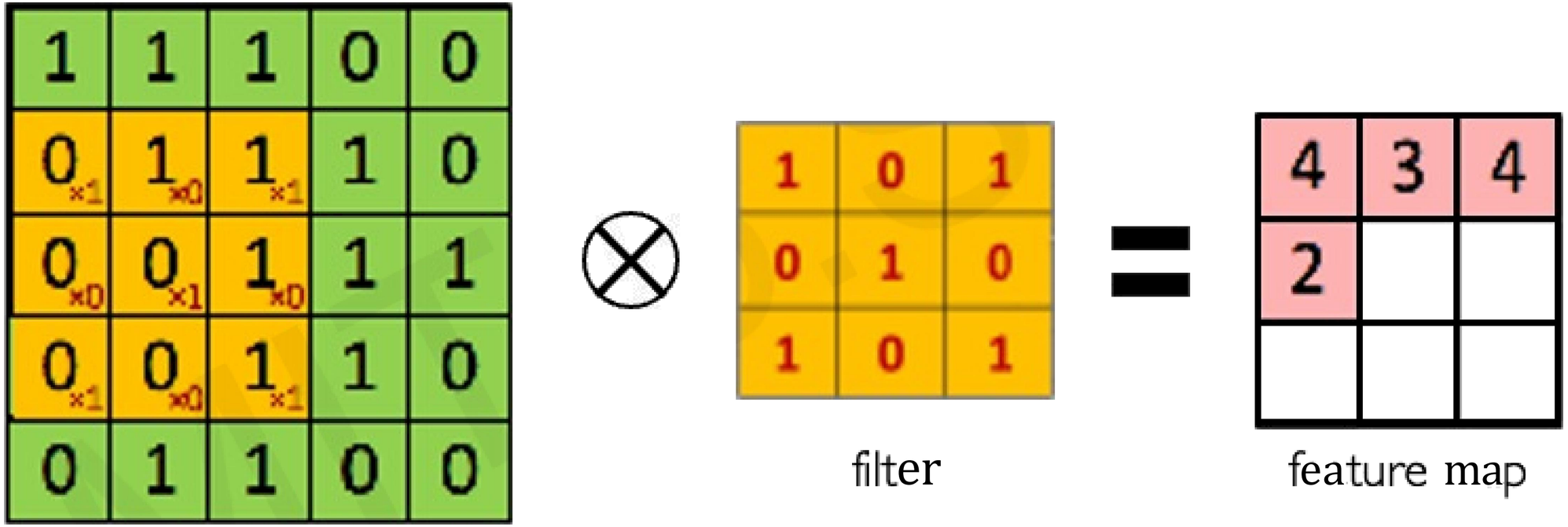
The Convolution Operation

We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs



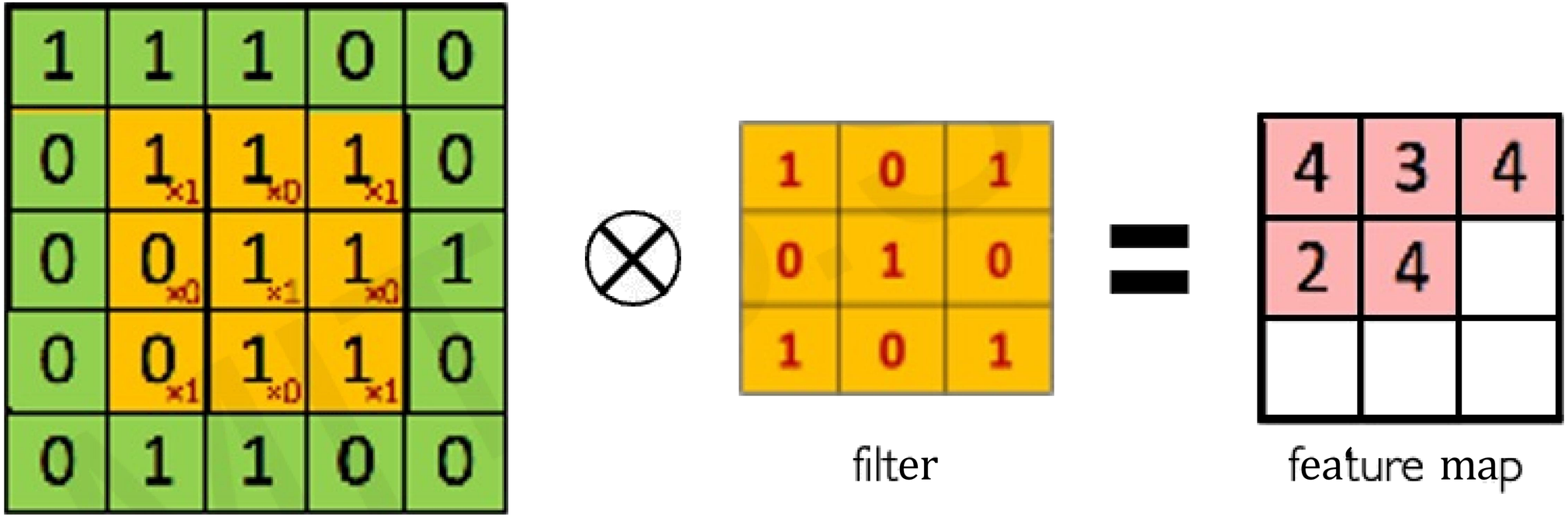
The Convolution Operation

We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs



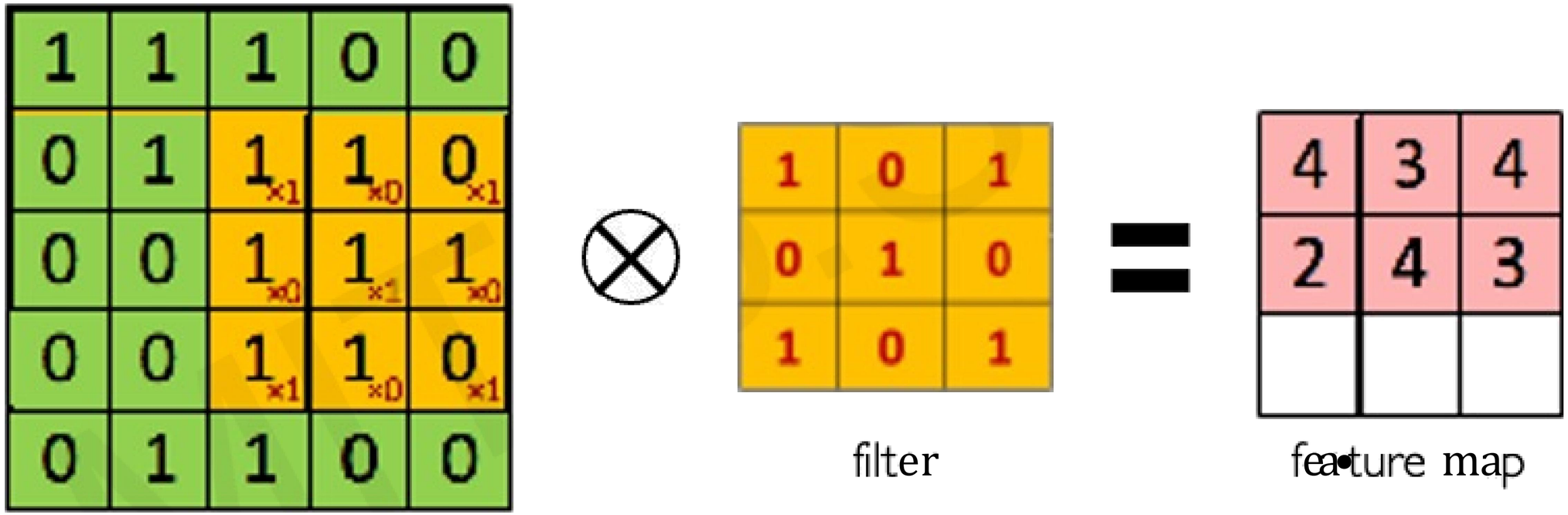
The Convolution Operation

We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs



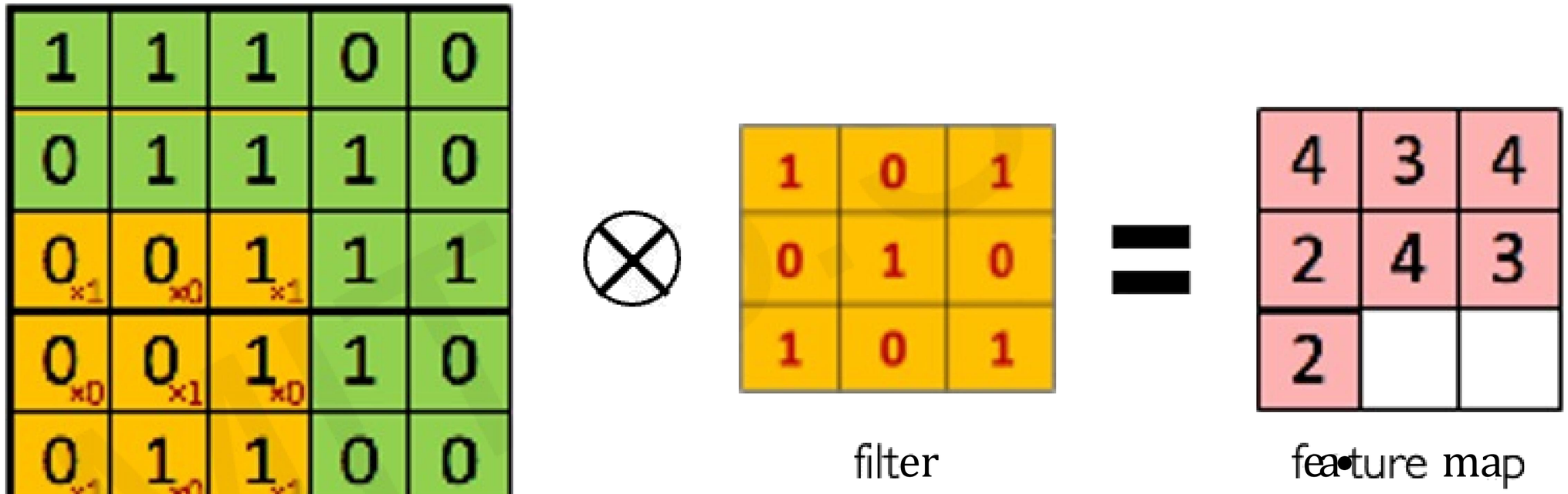
The Convolution Operation

We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs



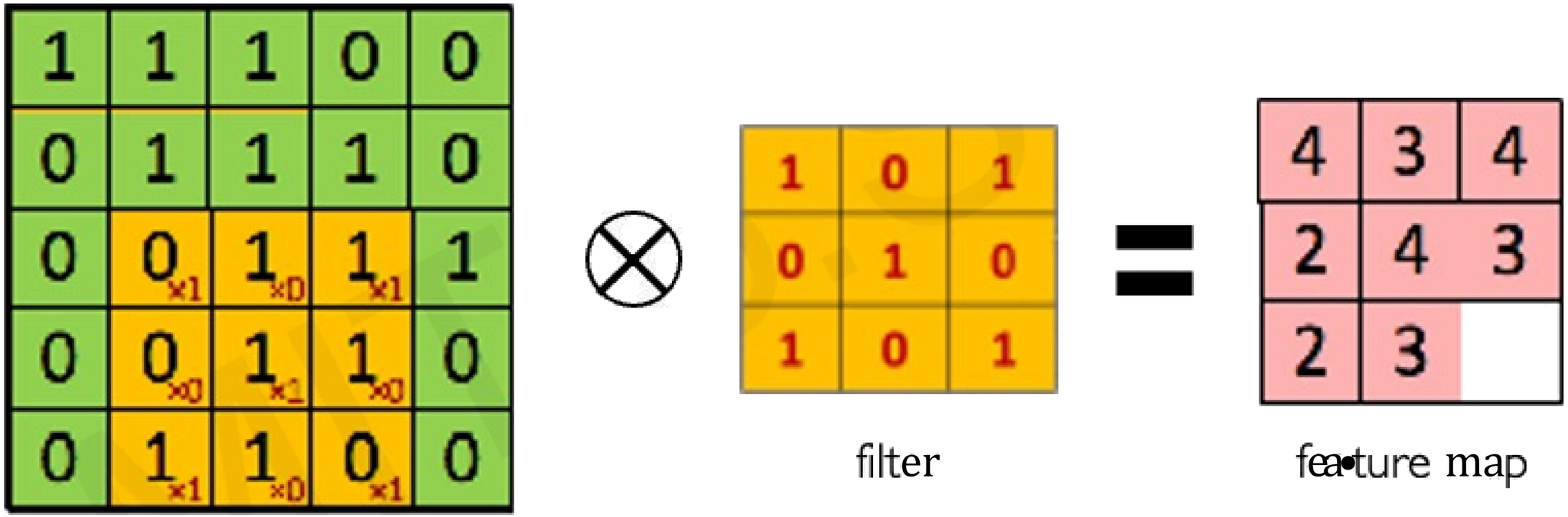
The Convolution Operation

We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs



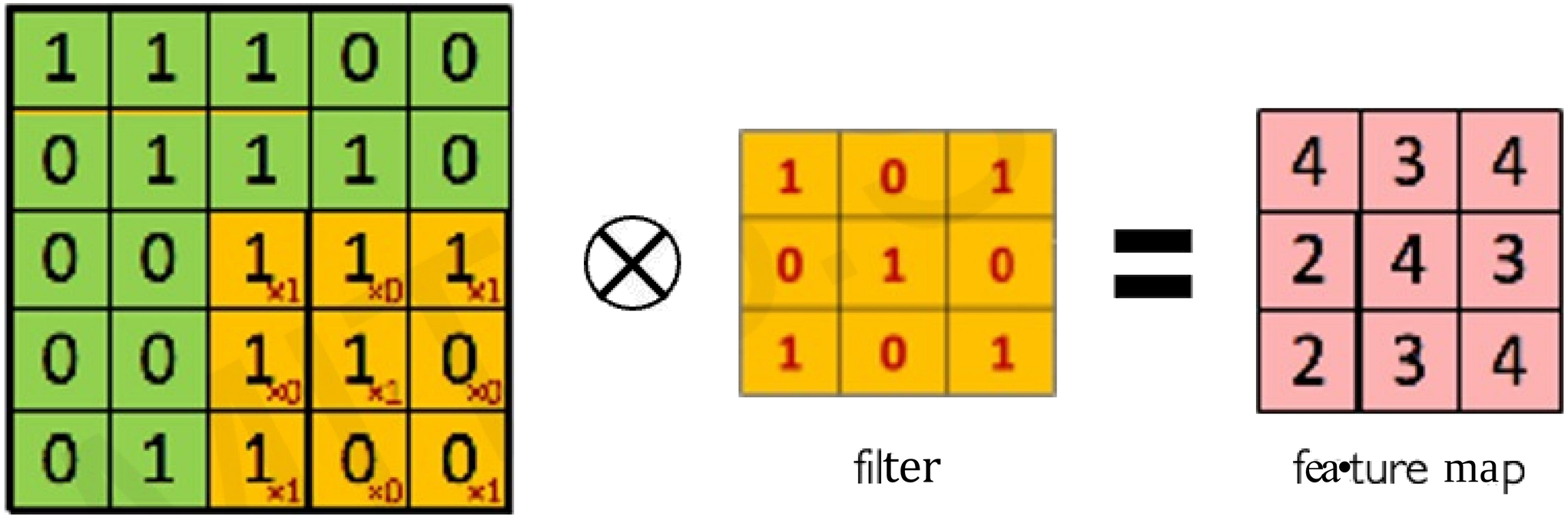
The Convolution Operation

We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs



The Convolution Operation

We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs



Producıng Feature Maps

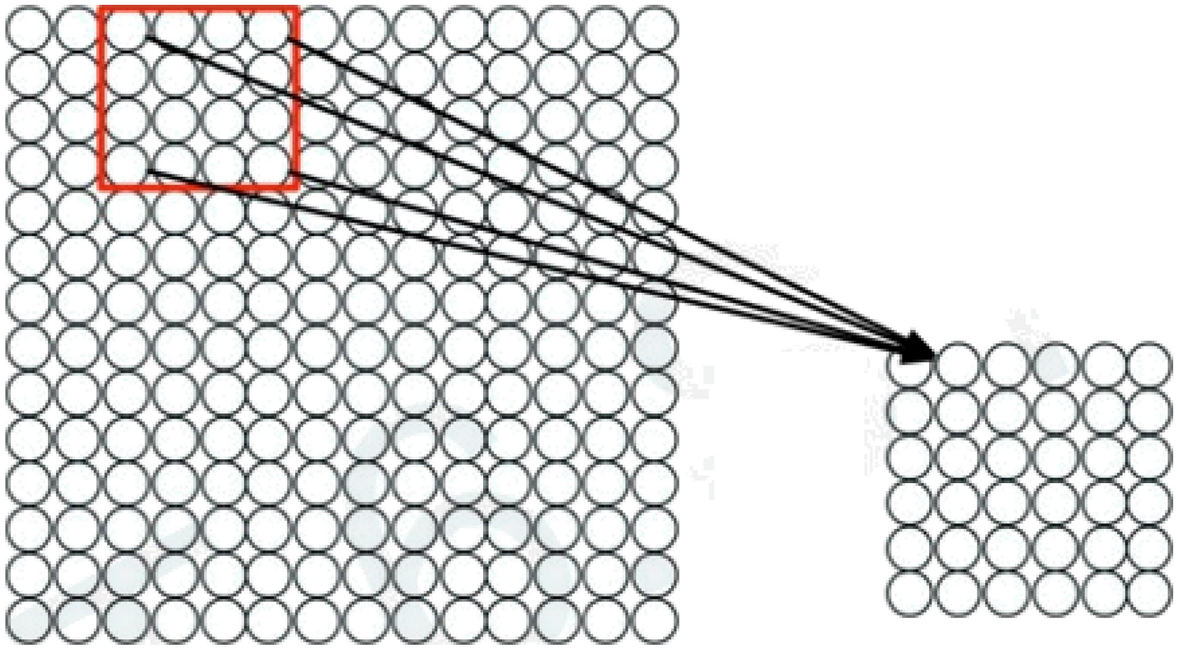
Orıgınal

Sharpen

Edge Deıect

"Strong' Edge
DeıJ ct

Feature Extraction with Convolution



1) Apply a set of weights – a filter — to extract **local features**

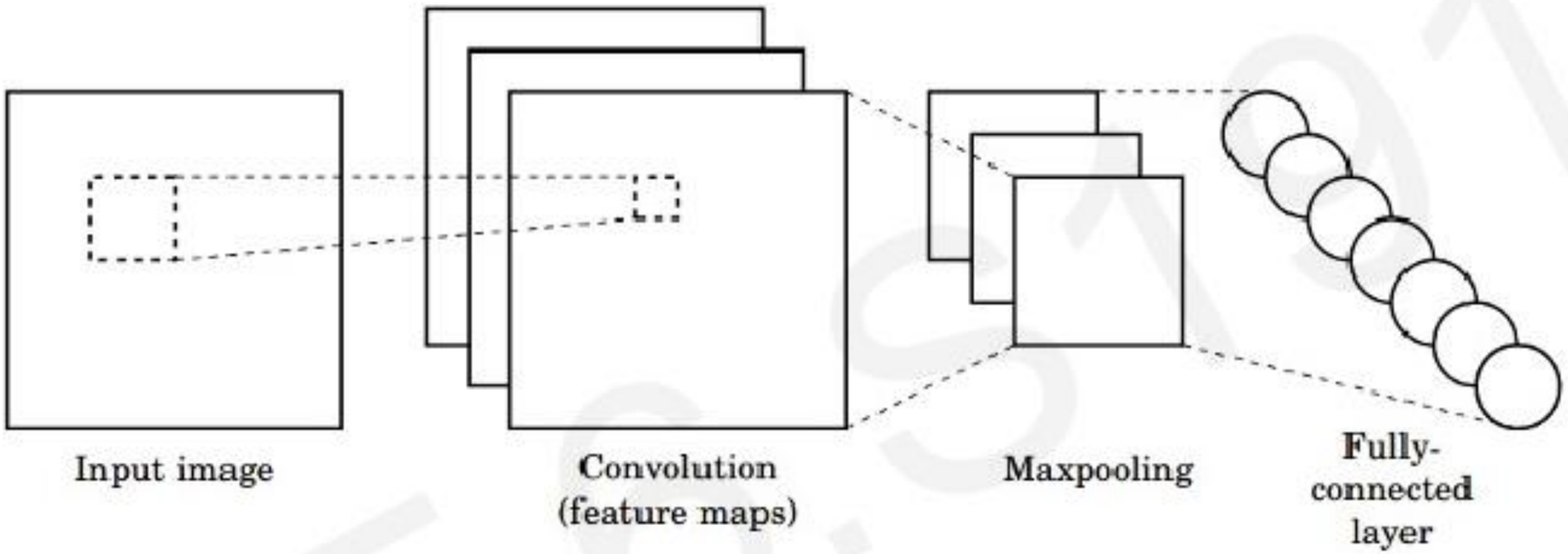
2) Use **multiple filters** to extract different features

3) **Spatially share** parameters of each filter

Convolutional Neural Networks (CNNs)

MIT 6.S191

CNNs for Classification



1. **Convolution**: Apply filters to generate feature maps.
2. **Non-linearity**: Often ReLU.
3. **Pooling**: Downsampling operation on each feature map.

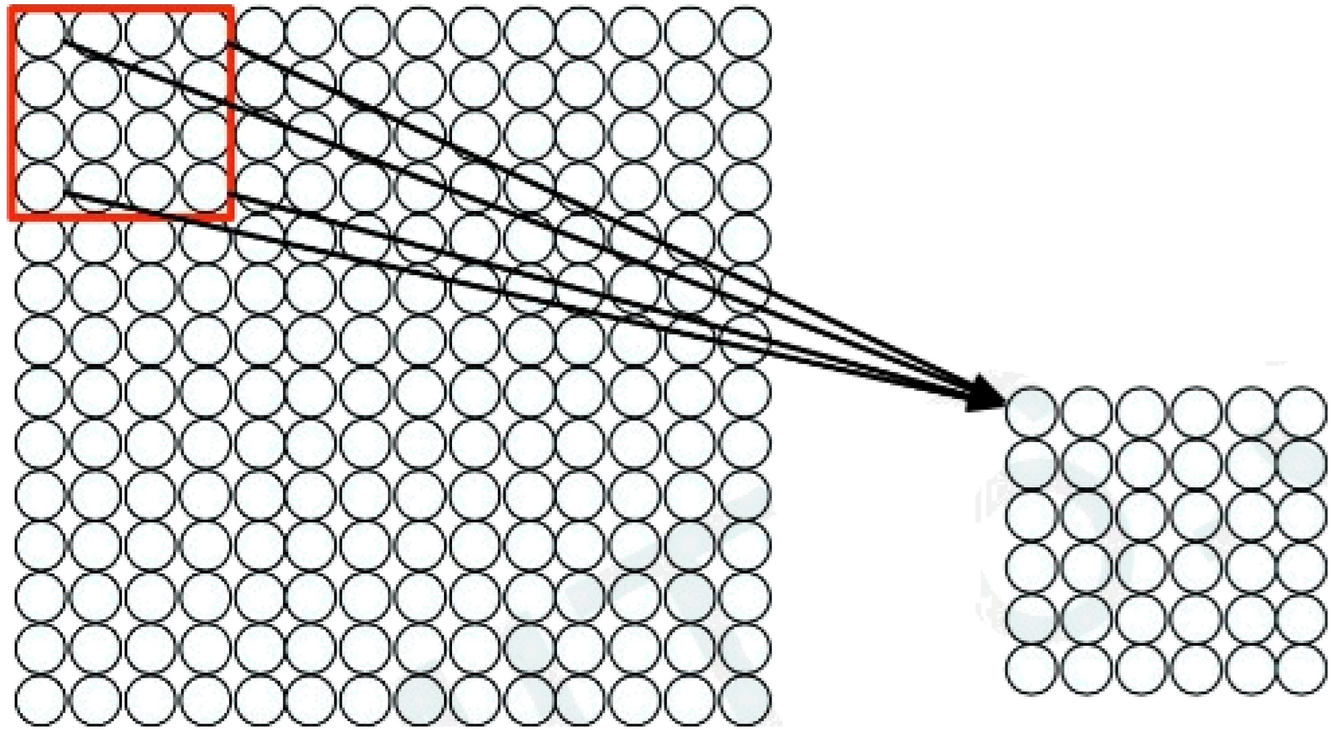
`tf.keras.layers.Conv2D`

`tf.keras.activations.*`

`tf.keras.layers.MaxPool2D`

Train model with image data.
Learn weights of filters in convolutional layers.

Convolutional Layers Local Connectivity



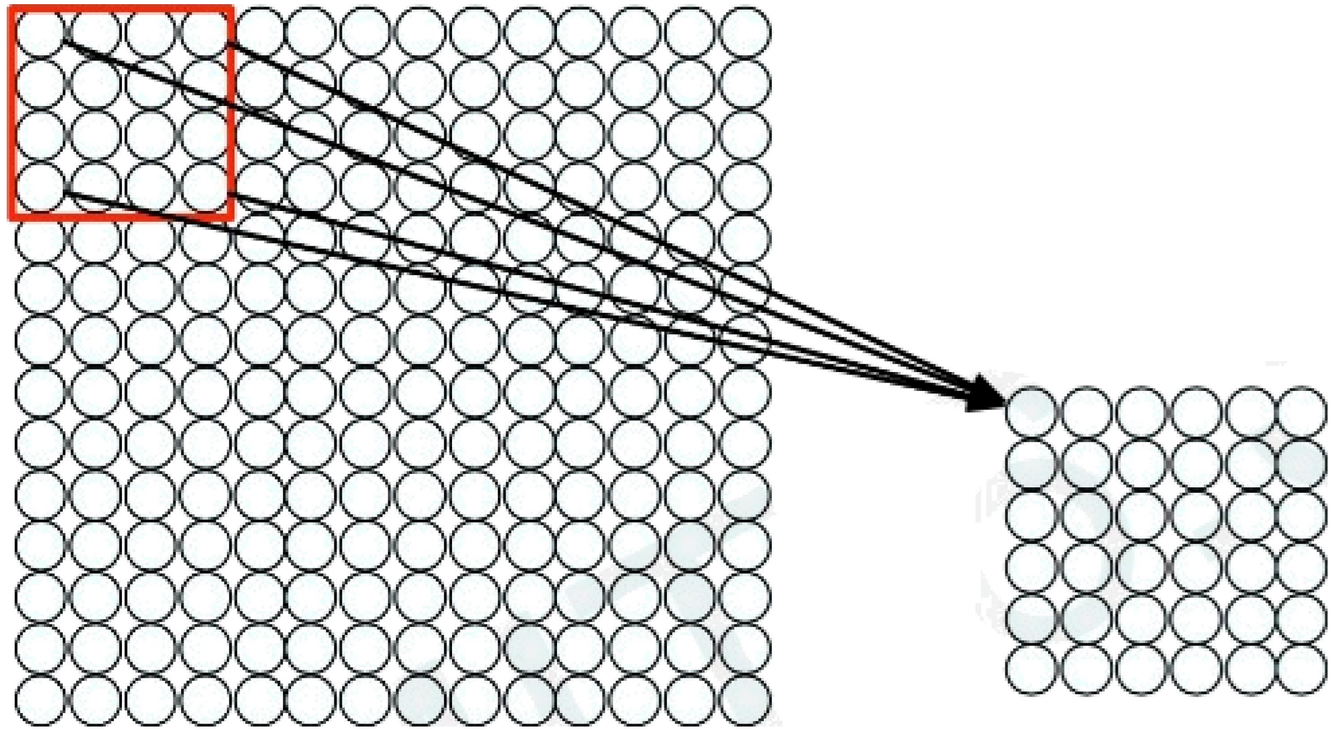
tf.keras.layers.Conv2D

For a neuron in hidden layer

- Take inputs from patch
- Compute weighted sum
- Appiy bias

Convolutional Layers Local Connectivity

tf.keras.layers.Conv2D



For a neuron in hidden layer:

- Take inputs from paich
- Compute weighted sum
- Apply bias

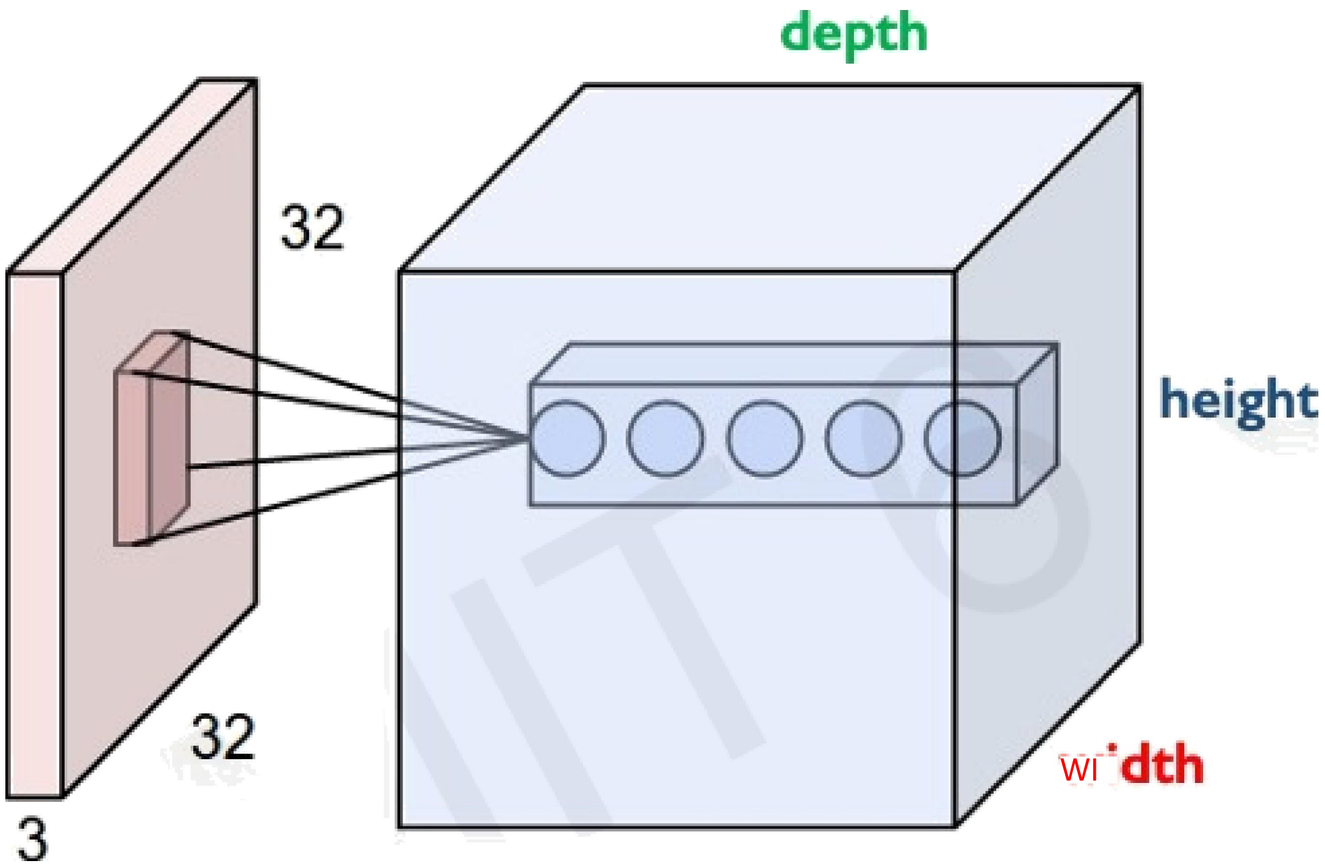
4x4 filter: matrix of weights $w_{t}g$

$$\sum_{i=1}^{4}\sum_{j=1}^{4} t/ \quad ‹4p„i4ę + b$$

for neuron (p,q) in hidden layer

I) applying a window of weighits
2) computing linear combinations
3) activating with non-linear function

CNNs Spatial Arrangement of Output Volume



Layer Dimensions:

$h \times w \times d$

where h and w are spatial dimensions
d (depth) = number of filters

Stride:

Filter step size

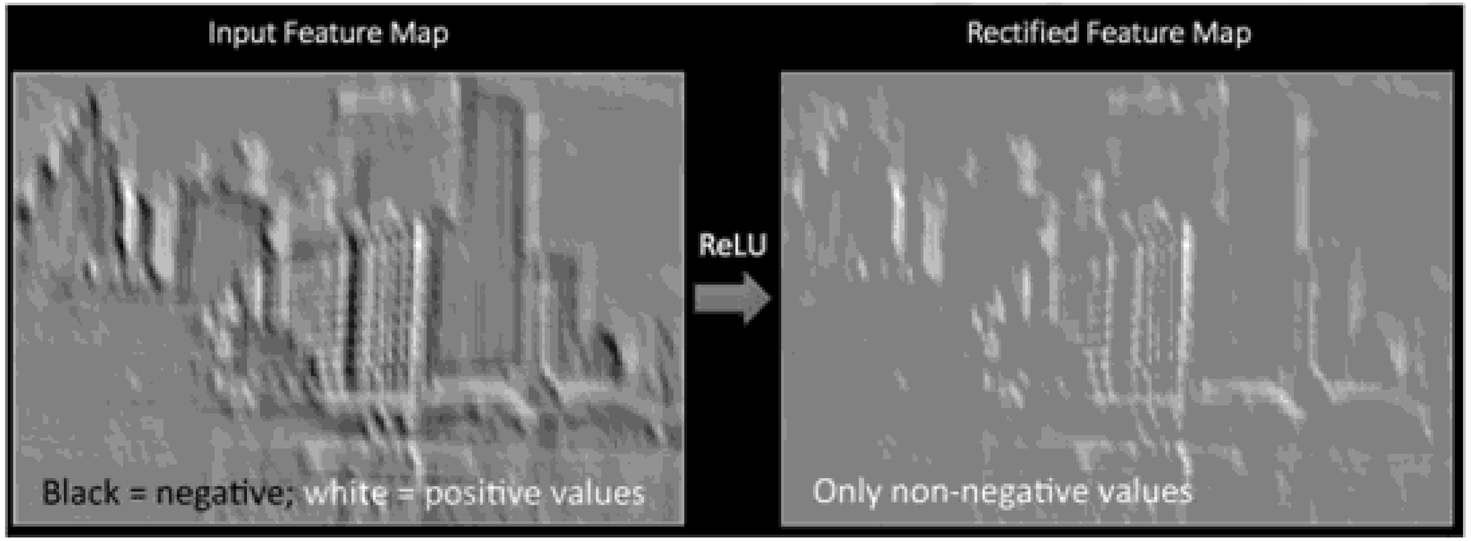
Receptive Field:

Locations in input image that a node is path connected to

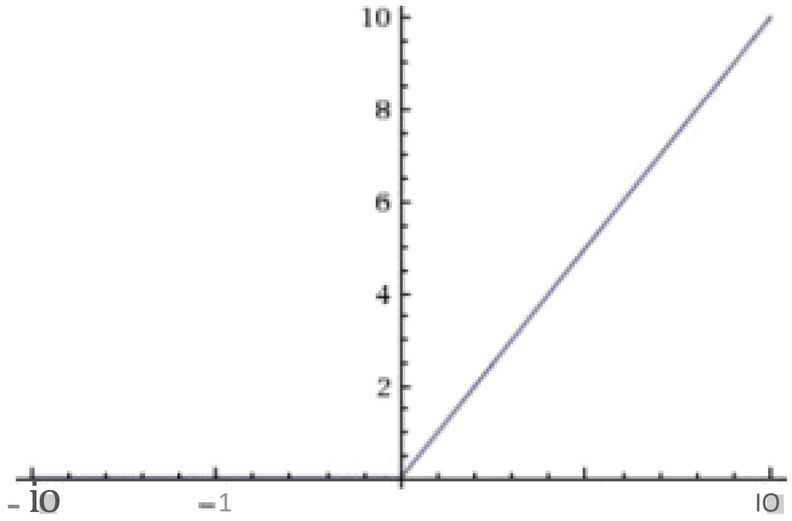
```
tf.keras.layers.Conv2D( filters=d, kernel_size=(h,w), strides=s )
```

Introducing Non Linearity

- Apply after every convolution operation (i.e- after convolutional layers)
- ReLU pixel-by-pixel operation that replaces all negative values by zero. **Non-linear operation!**



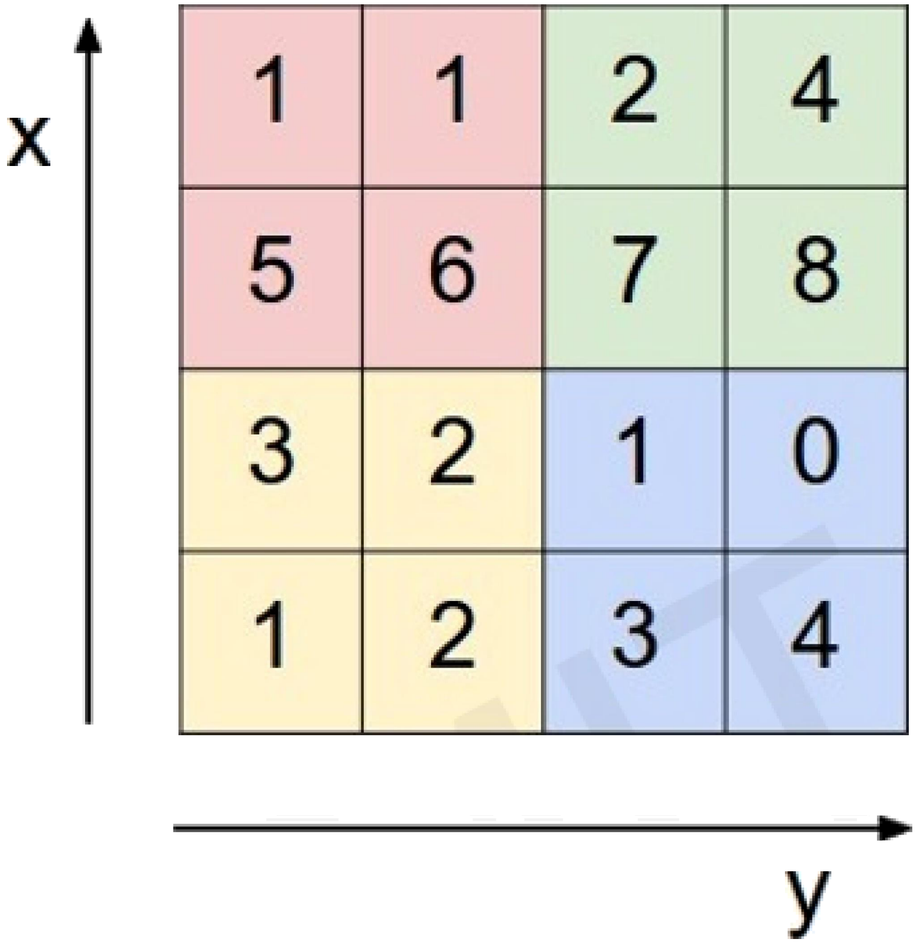
Rectified Linear Unit (ReLU)



$$g(z) = \max(0, z)$$

`tf.keras.layers.ReLU`

Pooling



max pool with 2x2 filters and stride 2

```
tf.keras.layers.MaxPool2D(
    pool_size=(2,2),
    strides=2
)
```

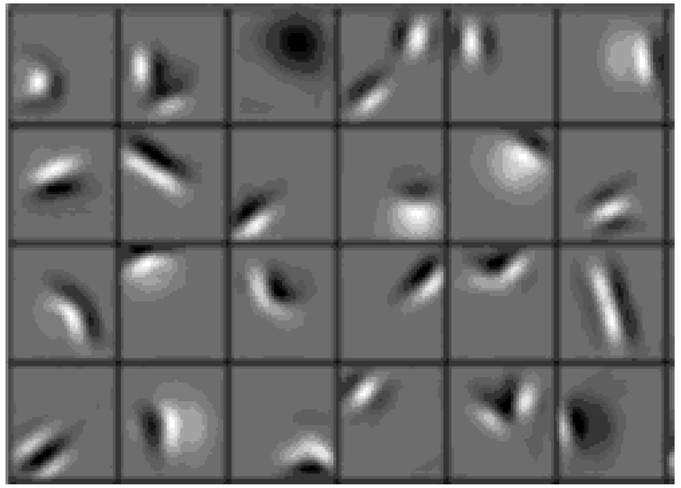
6	8
3	4

i) Reduced dimensionality
2) Spatial invariance

How else can we downsample and preserve spatial invariance?

Representation Learning in Deep CNNs

Low level fea‘tures



Edges dark spots

Conv Layer i

Mid level fea‘tures

Eyes, ears, nose

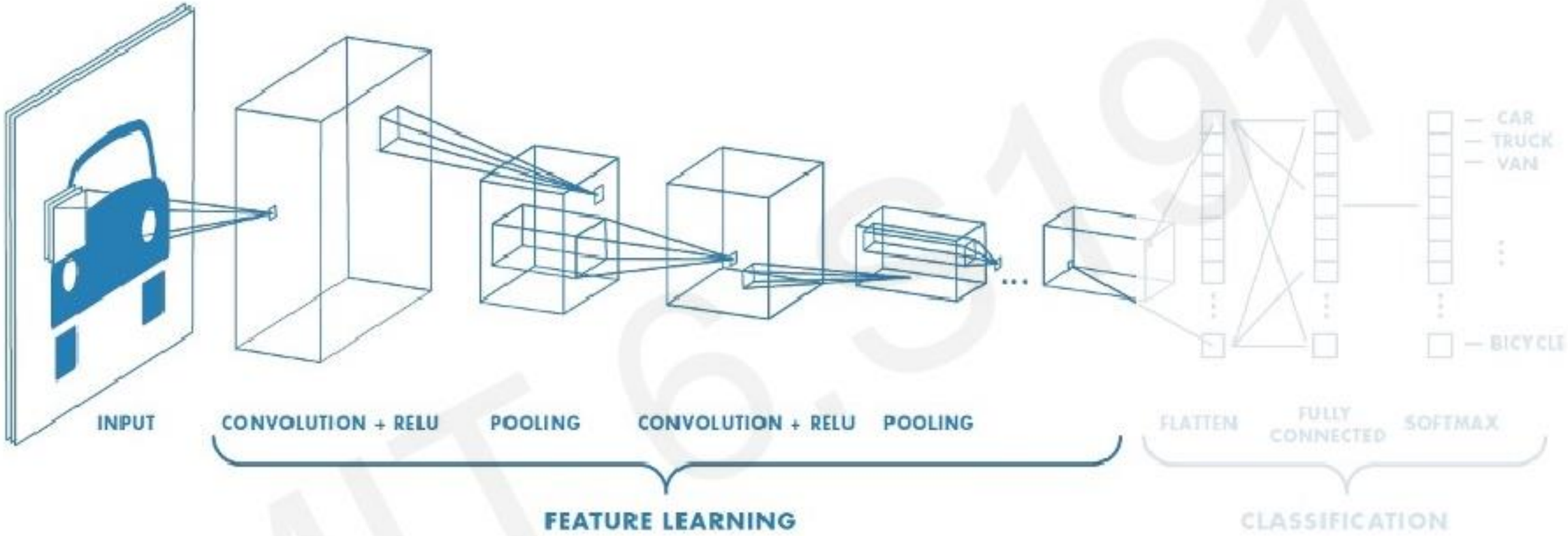
Conv Layer 2

High level features

Facial structure

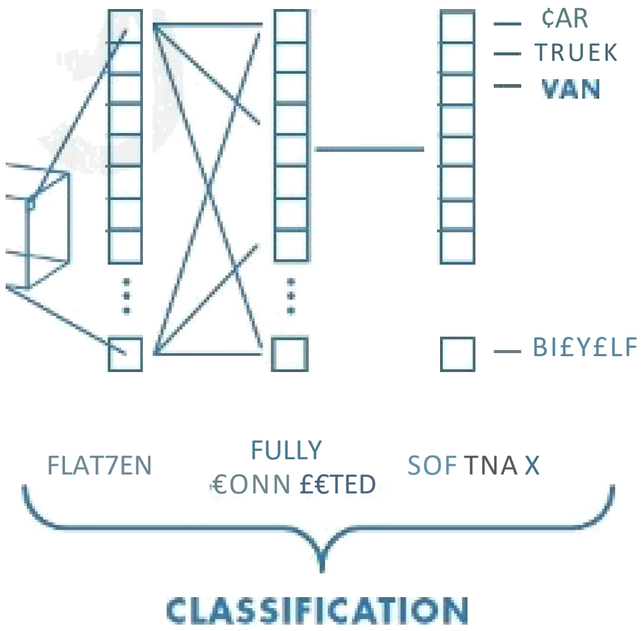
Conv Layer 3

CNNs for Classification: Feature Learning



1. Learn features in input image through **convolution**
2. Introduce **non-linearity** through activation function (real-world data is non-linear!)
3. Reduce dimensionality and preserve spatial invariance with **pooling**

CNNs for Classification Class Probabilities

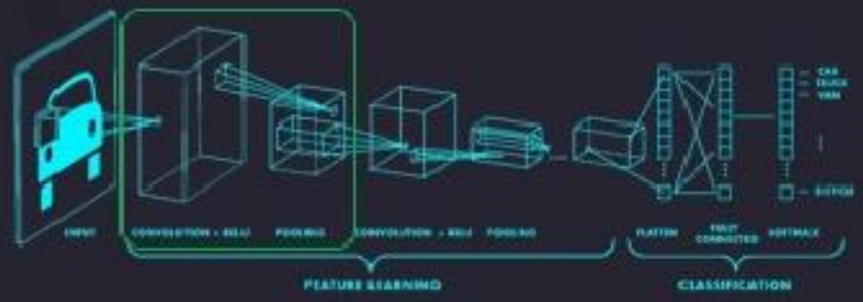


- CONV and POOL layers output high-level feŒums of input
- Fully connecże d layer uses these featums for classifying input image
- Expæss output as **probability** of image belonging to a particular class

$$\text{softmax}(y_i) = \frac{e^{*‹}}{\sum_j e^{+i}}$$

Putting it all together

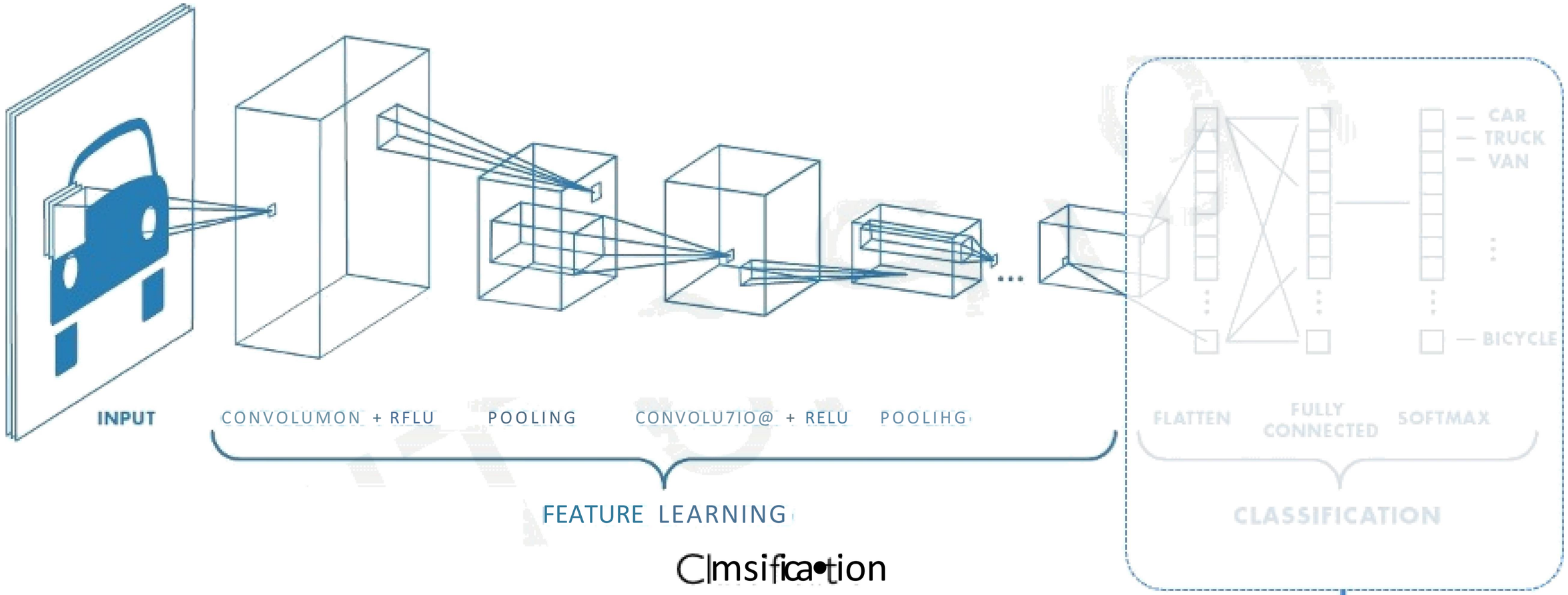
```python
import tensorflow as tf

def generate_model():
    model = tf.keras.Sequential([
        # first convolutional layer
        tf.keras.layers.Conv2D(32, filter_size=3, activation='relu'),
        tf.keras.layers.MaxPool2D(pool_size=2, strides=2),

        # second convolutional layer
        tf.keras.layers.Conv2D(64, filter_size=3, activation='relu'),
        tf.keras.layers.MaxPool2D(pool_size=2, strides=2),

        # fully connected classifier
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(1024, activation='relu'),
        tf.keras.layers.Dense(10, activation='softmax')  # 10 outputs
    ])
    return model
```



An Architecture for Many Applications

MIT 6.S191

An Architecture for Many Applications

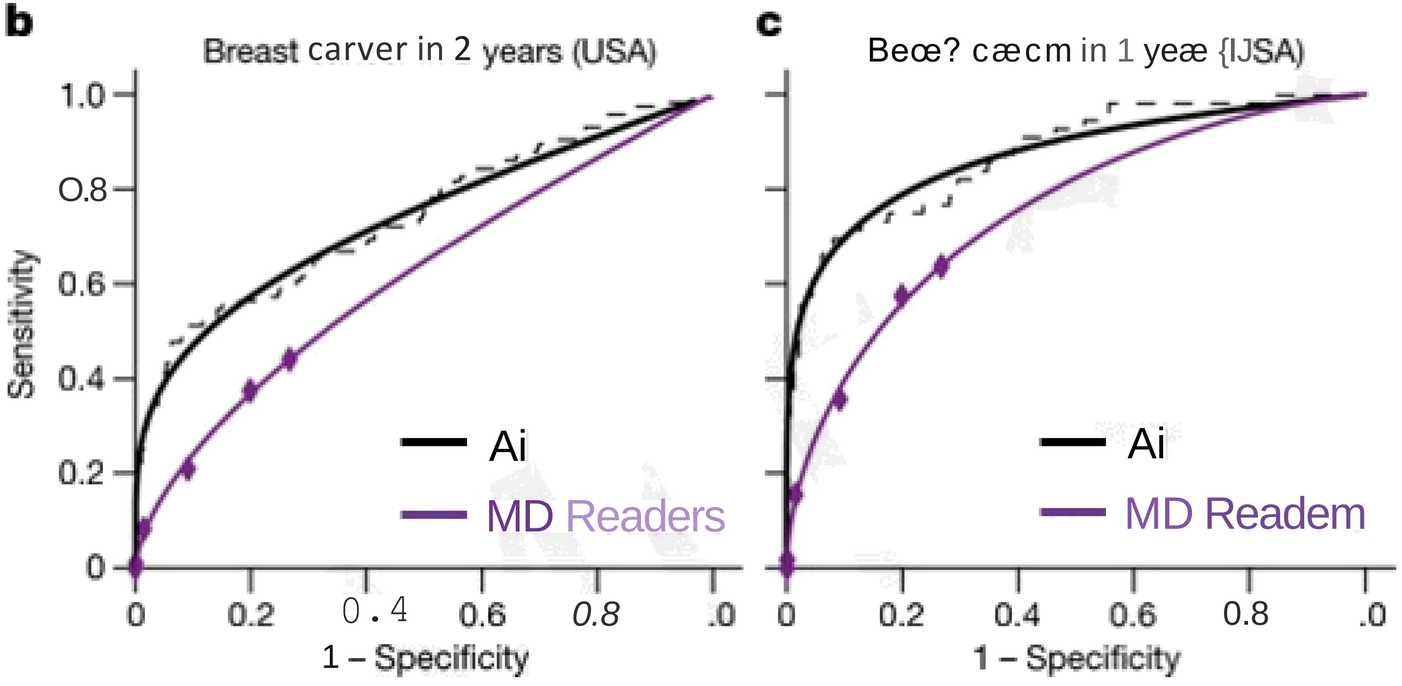


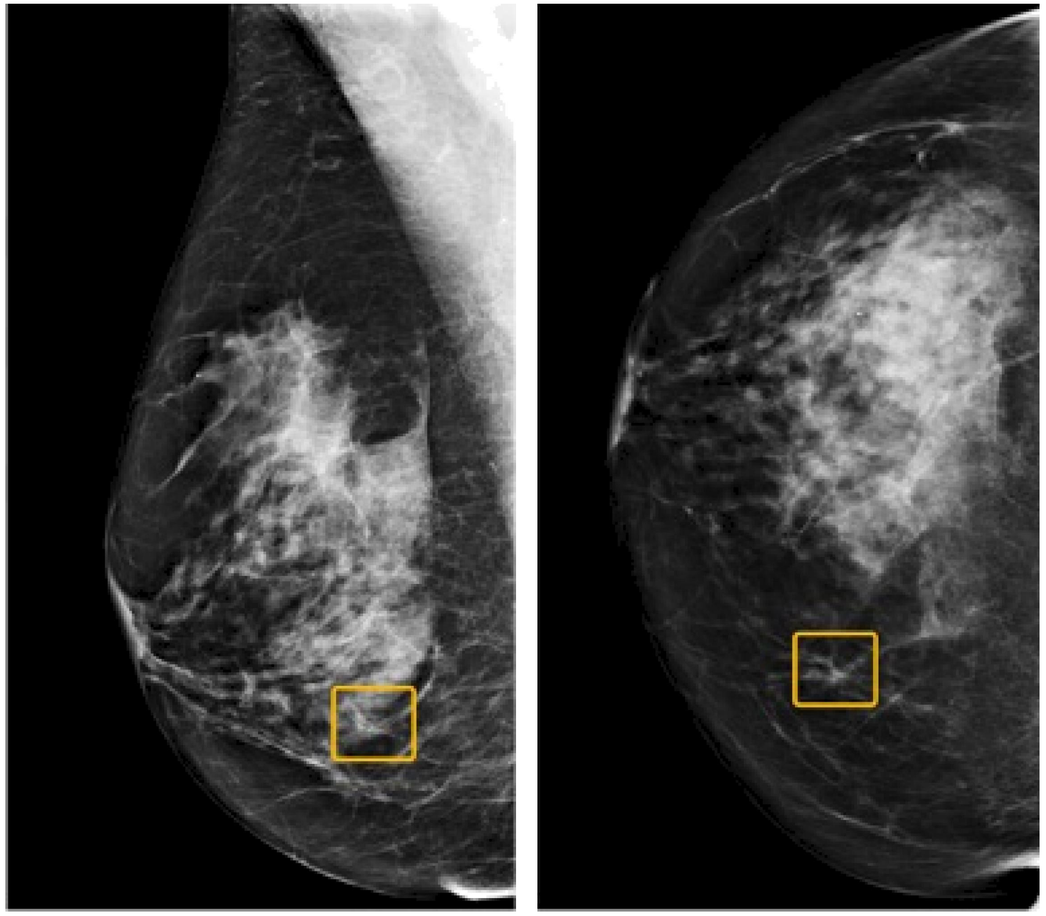
- Clmsifica•tion
- Object detection
- Segmentation
- Probabiiistic controi

Classification Breast Cancer Screening

International evaluation of an AI system for breast cancer screening

nature





CNN-bmed synem outperformed expen wdiologists at dRecting breast cmcer from mammograms

Breast cmcer cme mismd by radiologist bH dRected by AI

Object Detection

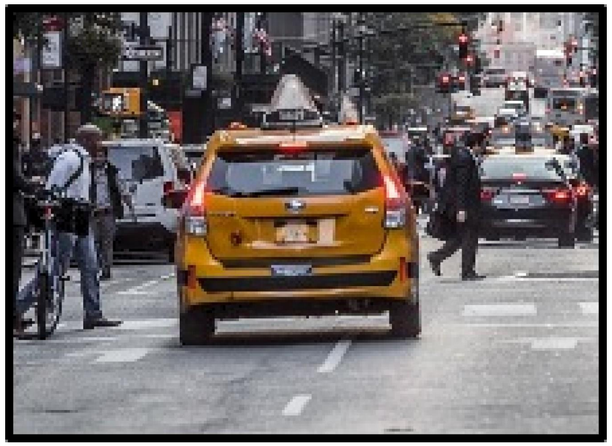


Image x

CNN

Taxi

Class label y

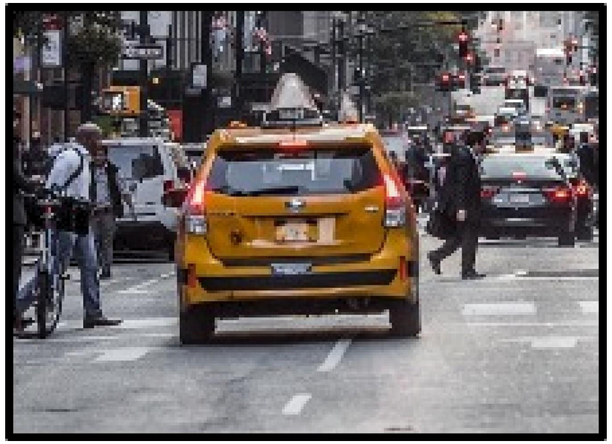
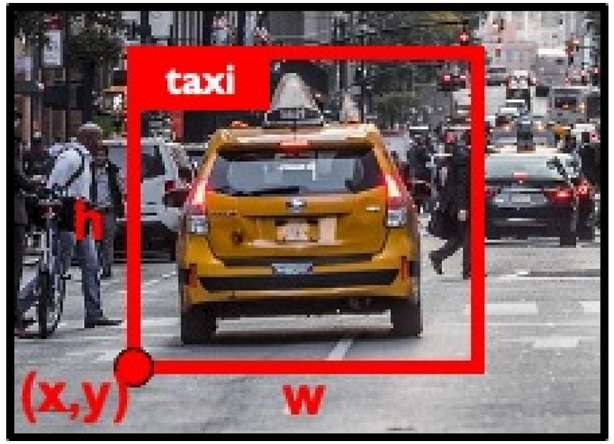


Image x

CNN



Label (x, y, w, h)

Object Detection

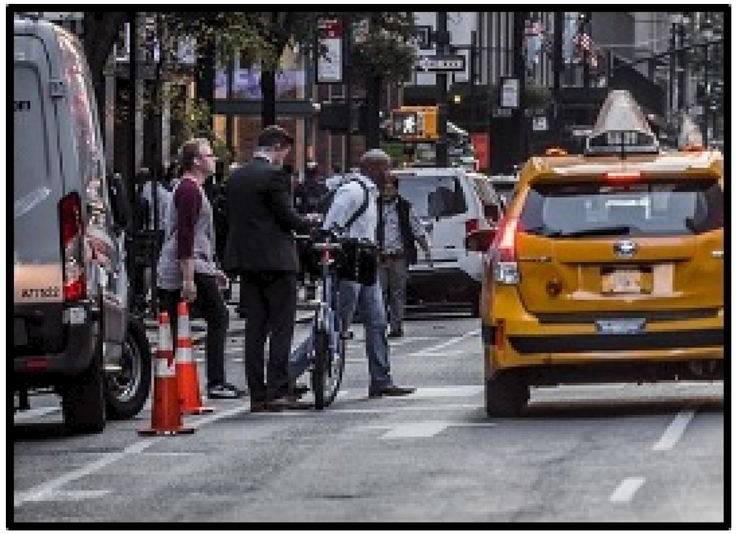
Image x

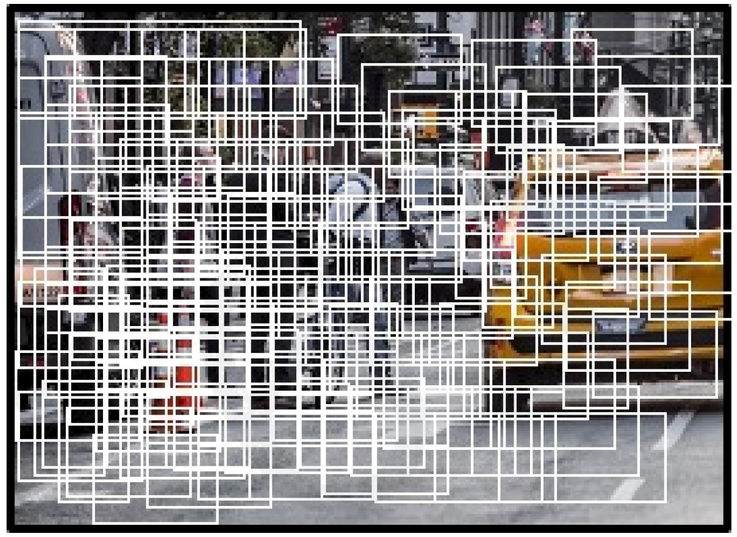
Output:
taxi: (x1, y1, w1, h1)

Image x

Output:
taxi: (x1, y1, w1, h1)
person: (x2, y2, w2, h2)
person: (x3, y3, w3, h3)
....

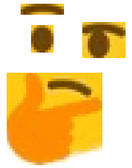
Naive Solution to Object Detection





CNN

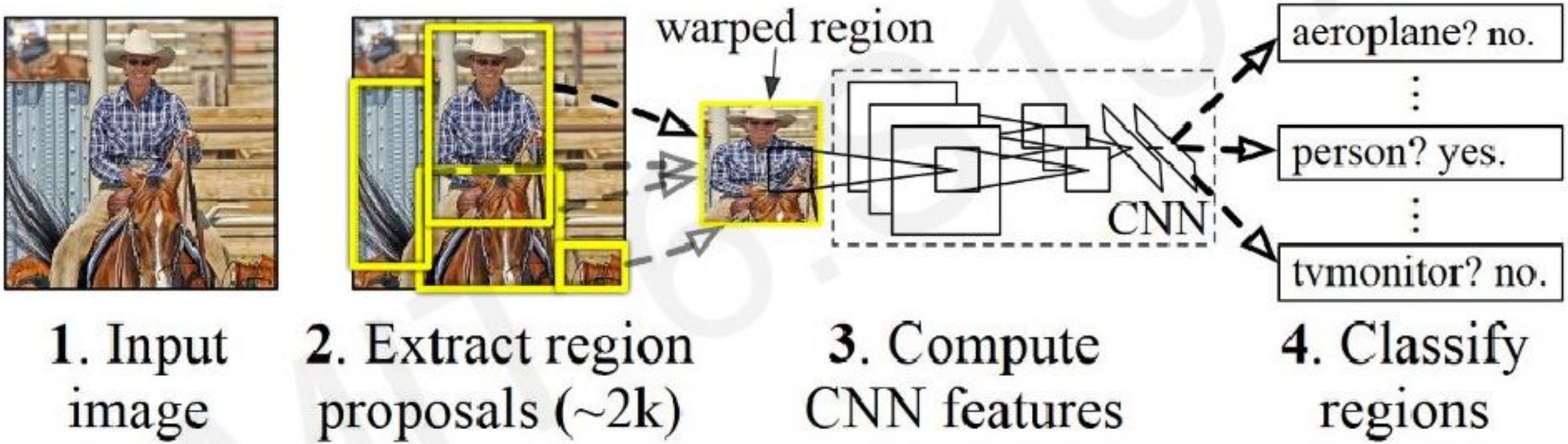
Class?



Problem Way too many inputs! This results in too many scales, positions, sizes!

Object Detection with R-CNNs

R-CNN algorithm: Find regions that we think have objects. Use CNN to classify.



Problems: 1) Slow! Many regions; time intensive inference.
2) Brittle! Manually defined region proposals.

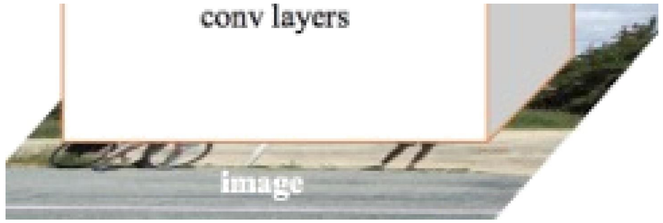
Faster R-CNN Learns Region Proposals

Clmsificaúon of regions →
object detecúon

Feature extraæion over
proposed regIOfiS

Region proposal network
to learn candidate regtone
Learned, data-dnven

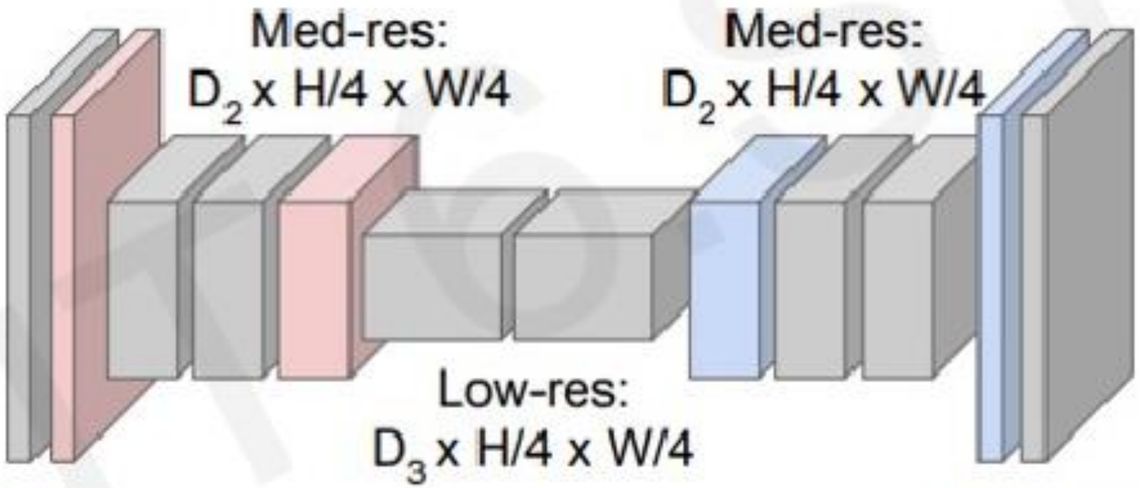
Image input directly into
nvolutional fœture extractor
Fast! Only input image once!



Semantic Segmentation: Fully Convolutional Networks

FCN: Fully Convolutional Network.
Network designed with all convolutional layers,
with **downsampling** and **upsampling** operations

Input:
3 x H x W



High-res:
D_1 x H/2 x W/2

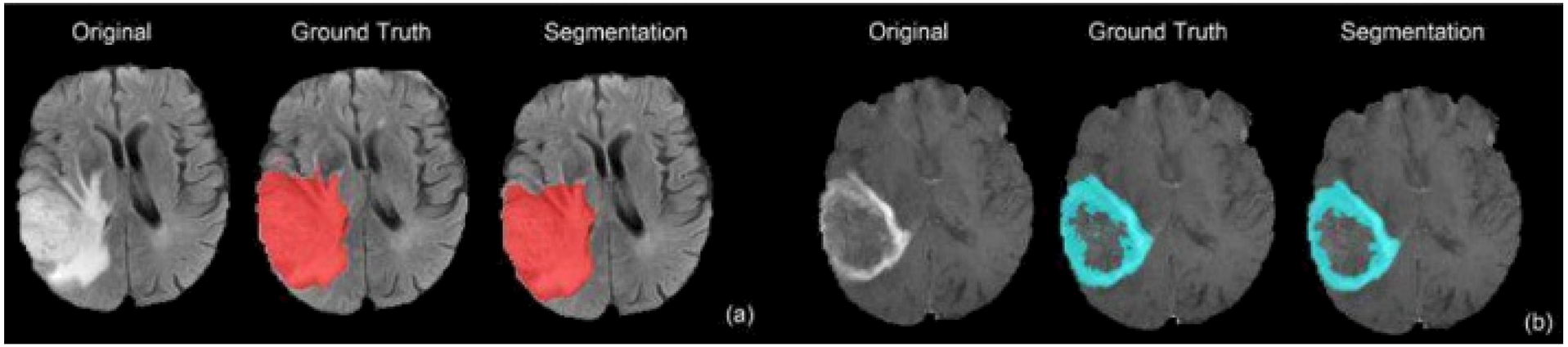
High-res:
D_1 x H/2 x W/2

Predictions:
H x W

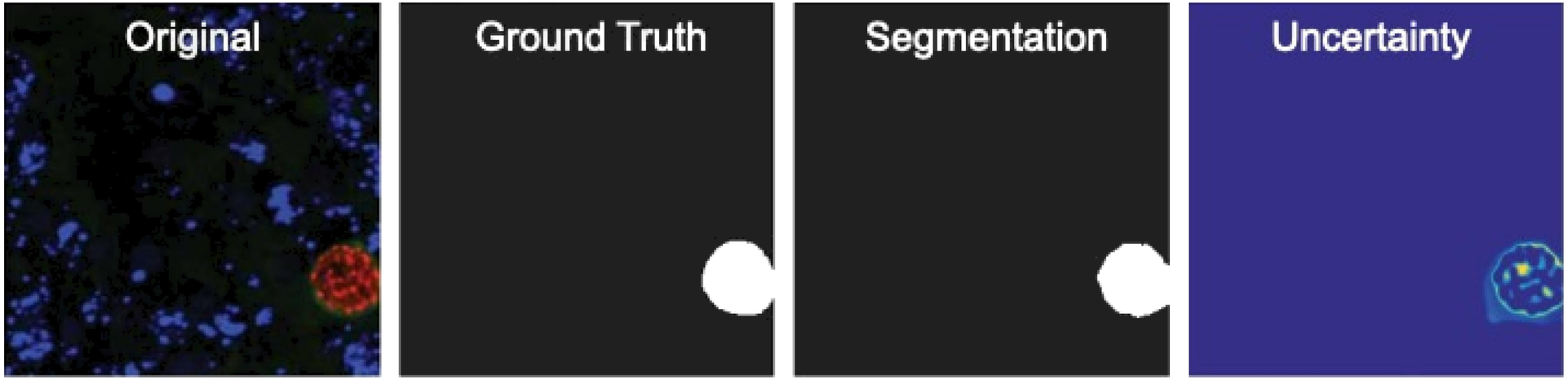
`tf.keras.layers.Conv2DTranspose`

Semantic Segmentation: Biomedical Image Analysis

Brain Tu
Dong+ *MIUA* 2017.



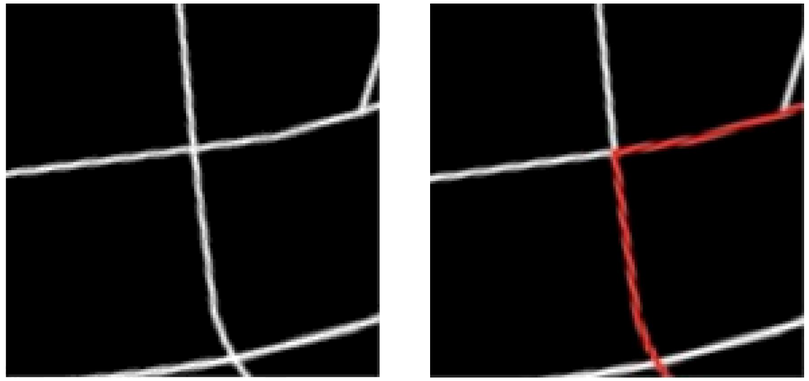
Malaria Infection
Soleimany+ *arXiv* 2019.



Continuous Control Navigation from Vision

Ram Peme ption
I
(ex. camera)

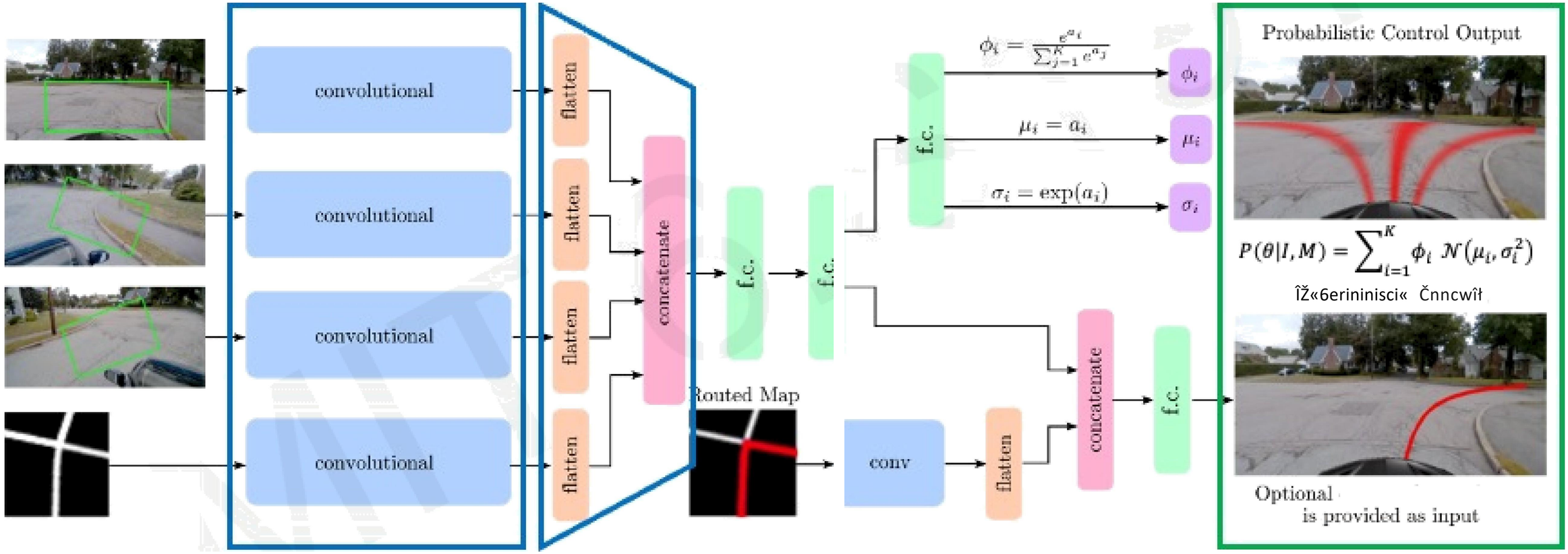
Coarse Maps
M
(ex. GPS)



Possible Control Commands

End-to End Framework for Autonomous Navigation

Entire modei is trained end-to-end **without any human labelling or annotations**



$$L = -\log(P(\theta|I,M))$$

Auto ON
Navigation and Localization
Amini+ ICRA 2019.

Deep Learning for Computer Vision: Impact

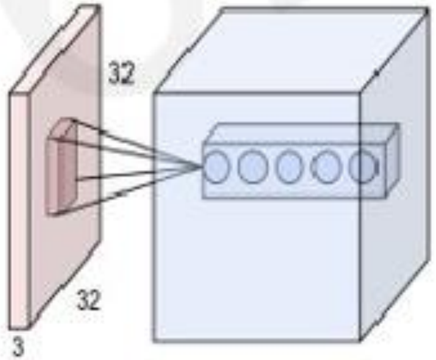
Deep Learning for Computer Vision: Summary

Foundations

- Why computer vision?
- Representing images
- Convolutions for feature extraction

CNNs

- CNN architecture
- Application to classification
- ImageNet



Applications

- Segmentation, image captioning, control
- Security, medicine, robotics

MIT 6.S191