

LAB 4 - GOING DEEPER INTO CONVOLUTIONAL NEURAL NETWORKS

Dr. Pandiyaraju V
Shravan Venkatraman

August 9, 2024

TABLE OF CONTENTS

01

INTRODUCTION

02

MLP RECAP

03

MINI BATCH GRADIENT DESCENT

04

HYPERPARAMETER TUNING

05

TUNING TECHNIQUES

01

INTRODUCTION

02

MLP RECAP

03

MINI BATCH GRADIENT DESCENT

04

HYPERPARAMETER TUNING

05

TUNING TECHNIQUES

CNN RECAP

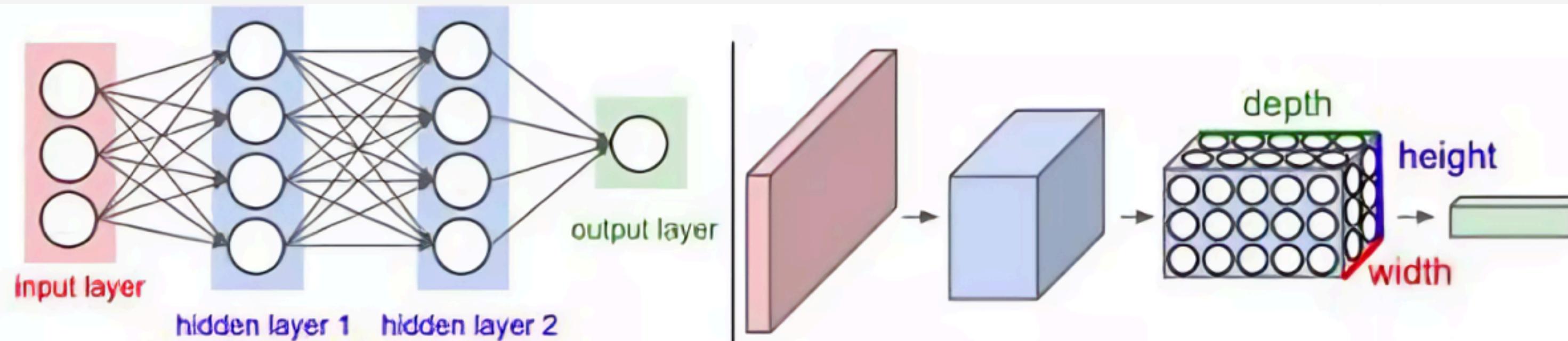
Human Vision



Computer Vision

58	24	134	93	114	50	38	178	163	127	211	176	39	12	88	191
23	10	140	196	193	223	29	240	25	53	234	232	125	88	70	141
54	7	126	101	194	118	64	108	155	58	53	170	173	72	219	234
224	89	234	149	185	106	252	0	222	118	41	70	193	25	10	86
6	89	54	236	46	55	207	162	198	76	71	18	41	96	136	13
20	131	173	254	166	198	148	44	80	56	126	63	118	52	216	81
143	171	194	205	197	132	125	208	127	29	179	232	109	210	50	10
86	49	90	220	162	41	28	153	96	240	191	186	179	38	57	51
138	90	179	39	42	34	100	246	215	134	39	40	253	167	201	93
116	250	142	106	139	5	222	39	200	150	110	60	125	118	201	18
182	144	96	42	152	216	166	248	176	243	224	76	242	52	224	58
244	117	62	183	80	34	117	125	81	203	77	224	201	167	30	141
142	148	161	241	131	159	188	232	73	134	199	45	109	74	27	250
66	158	244	9	253	149	152	64	108	57	61	192	22	111	73	10
206	19	90	68	185	138	228	107	143	114	10	31	8	238	68	47
29	43	186	2	214	174	33	253	183	181	202	139	173	102	5	72
170	1	170	64	110	247	244	118	163	203	137	2	63	208	64	131
98	34	92	145	14	122	35	111	85	255	55	43	99	198	143	254
226	88	133	62	140	212	235	45	238	83	100	32	46	63	104	151
219	46	170	76	58	213	126	66	61	154	96	122	29	9	205	164
71	106	191	194	78	147	224	190	179	39	103	61	238	108	95	148
98	145	27	125	53	38	189	224	23	239	68	145	245	249	160	161
25	200	201	88	21	160	159	237	82	183	14	236	68	153	250	140
126	182	37	225	118	180	195	118	143	123	189	247	116	181	24	50
167	194	188	101	96	194	143	140	103	2	204	87	88	207	11	36
205	127	184	108	241	66	10	174	128	13	5	185	66	248	147	36
163	37	97	52	70	222	30	14	79	75	81	193	148	106	144	68

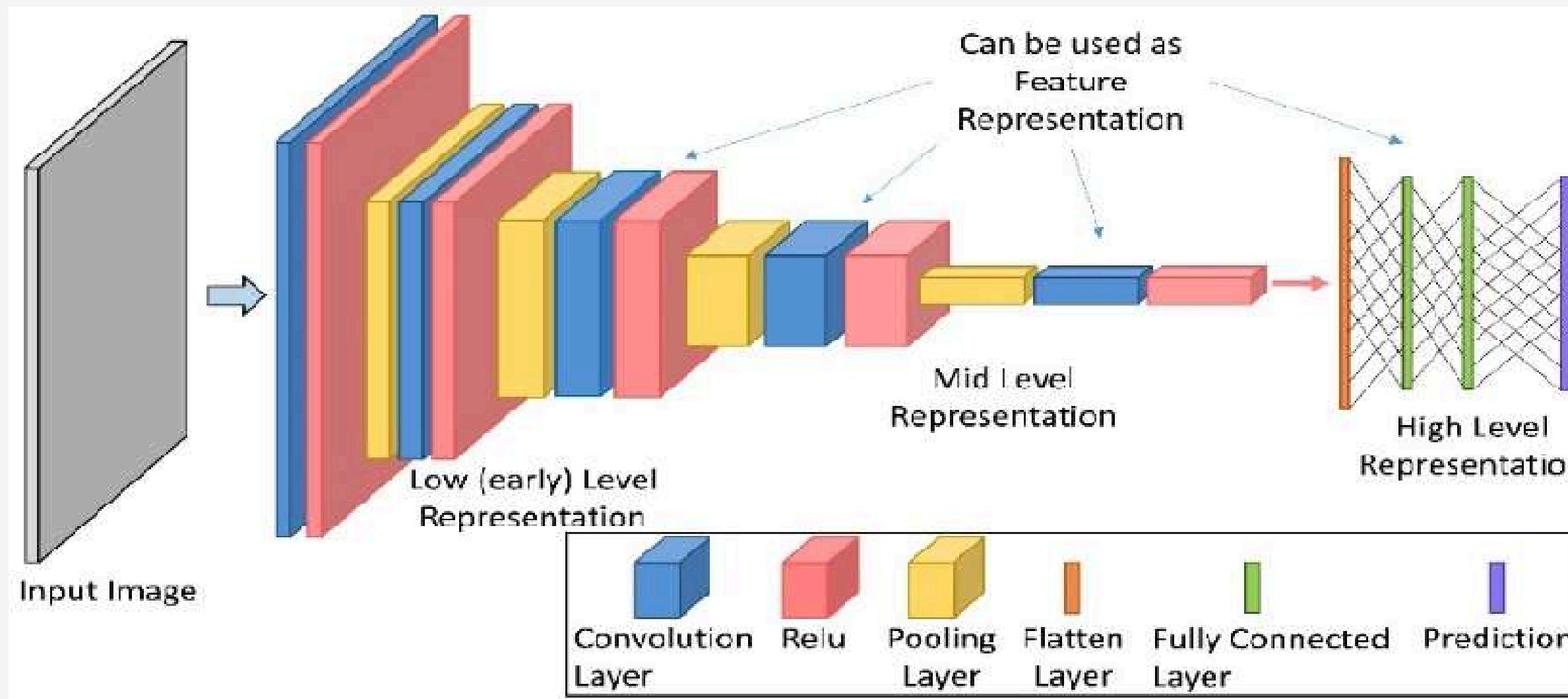
WHY CNNS?



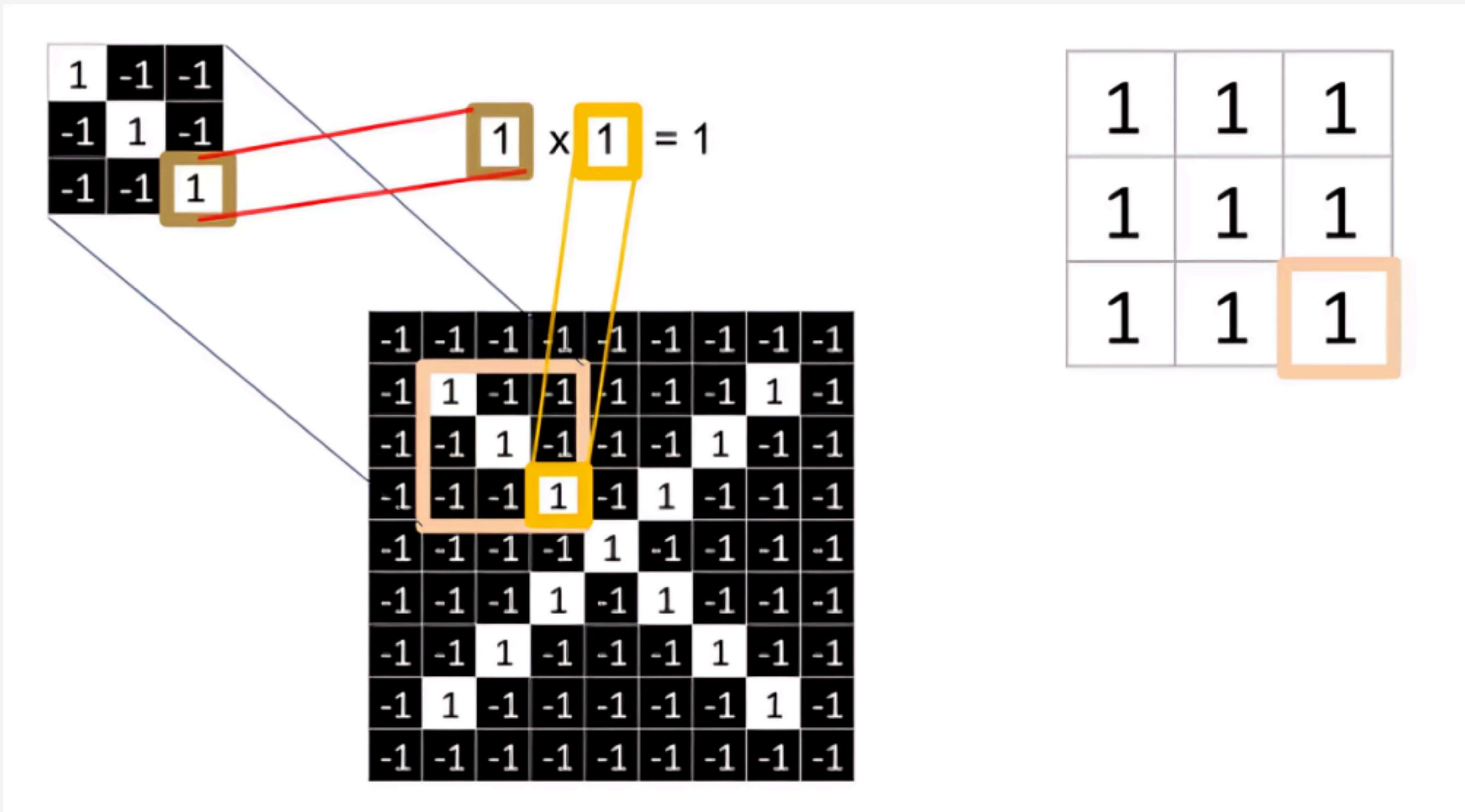
Left: A regular 3-layer Neural Network. Right: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels).

COMPONENTS OF A CNN

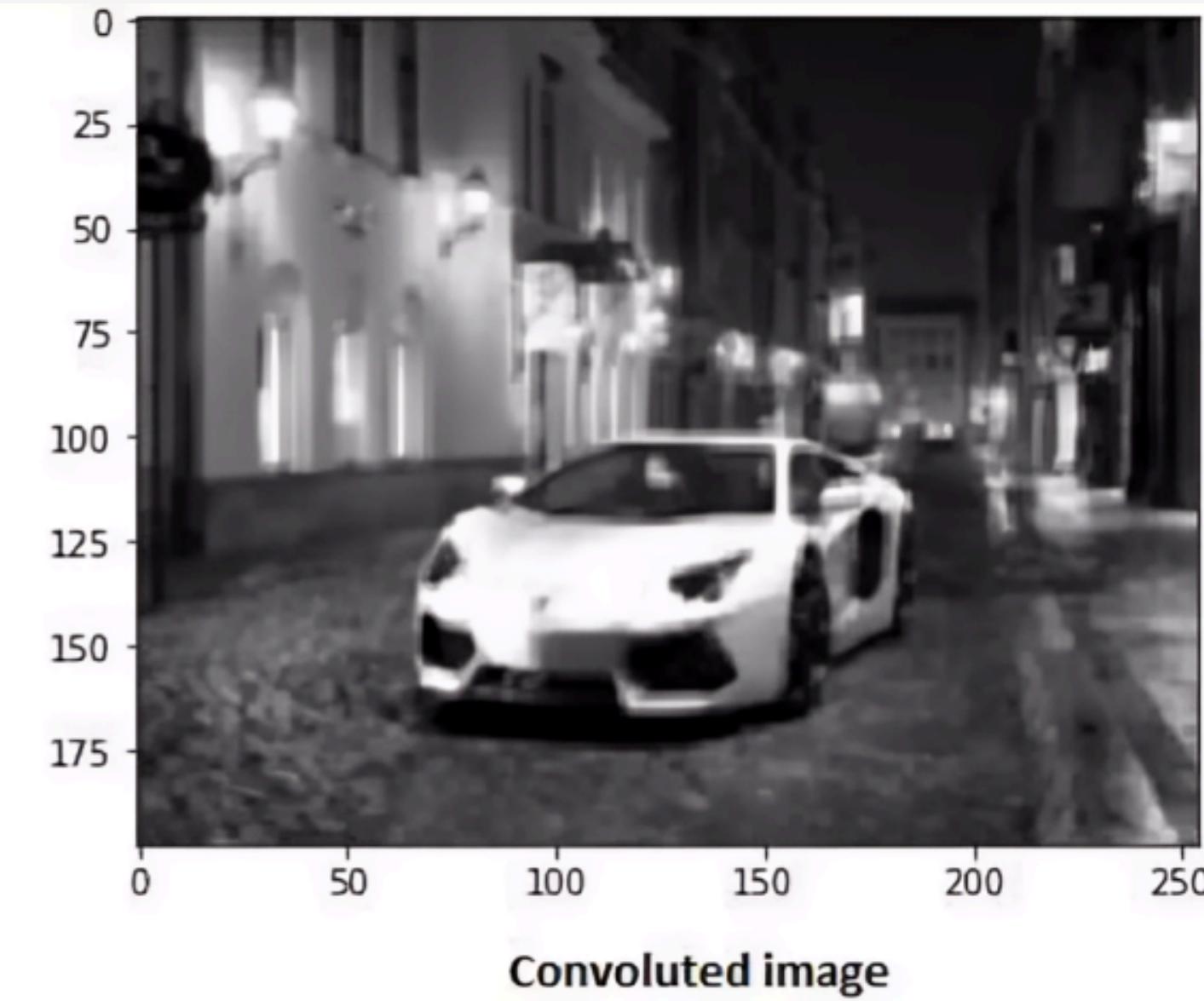
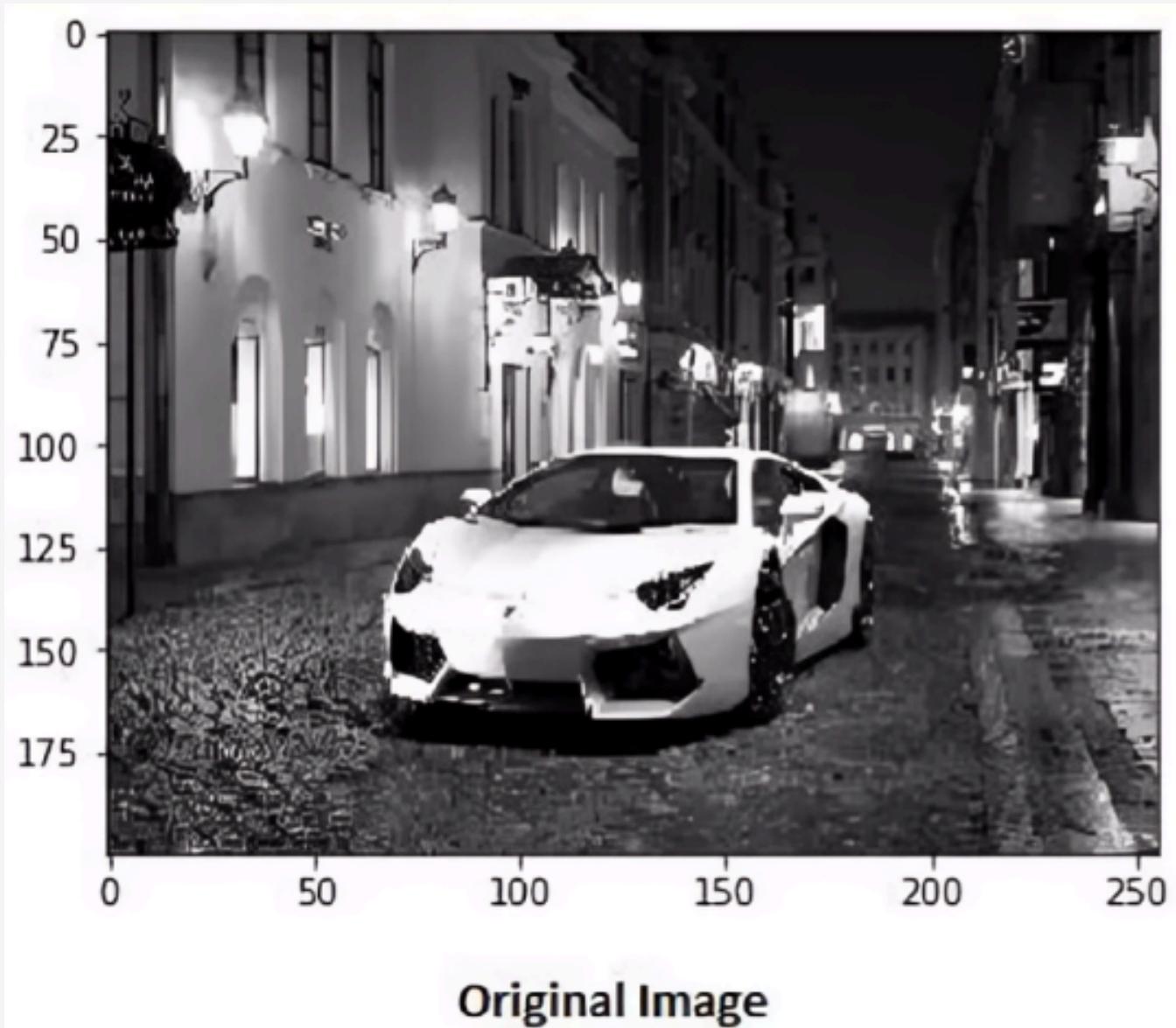
- Hidden Layers / Feature Extraction Part: the part where feature extraction happens, via a series of convolutions and pooling operations.
- The classification part: the fully connected layers will serve as the classifier on top of these extracted features.



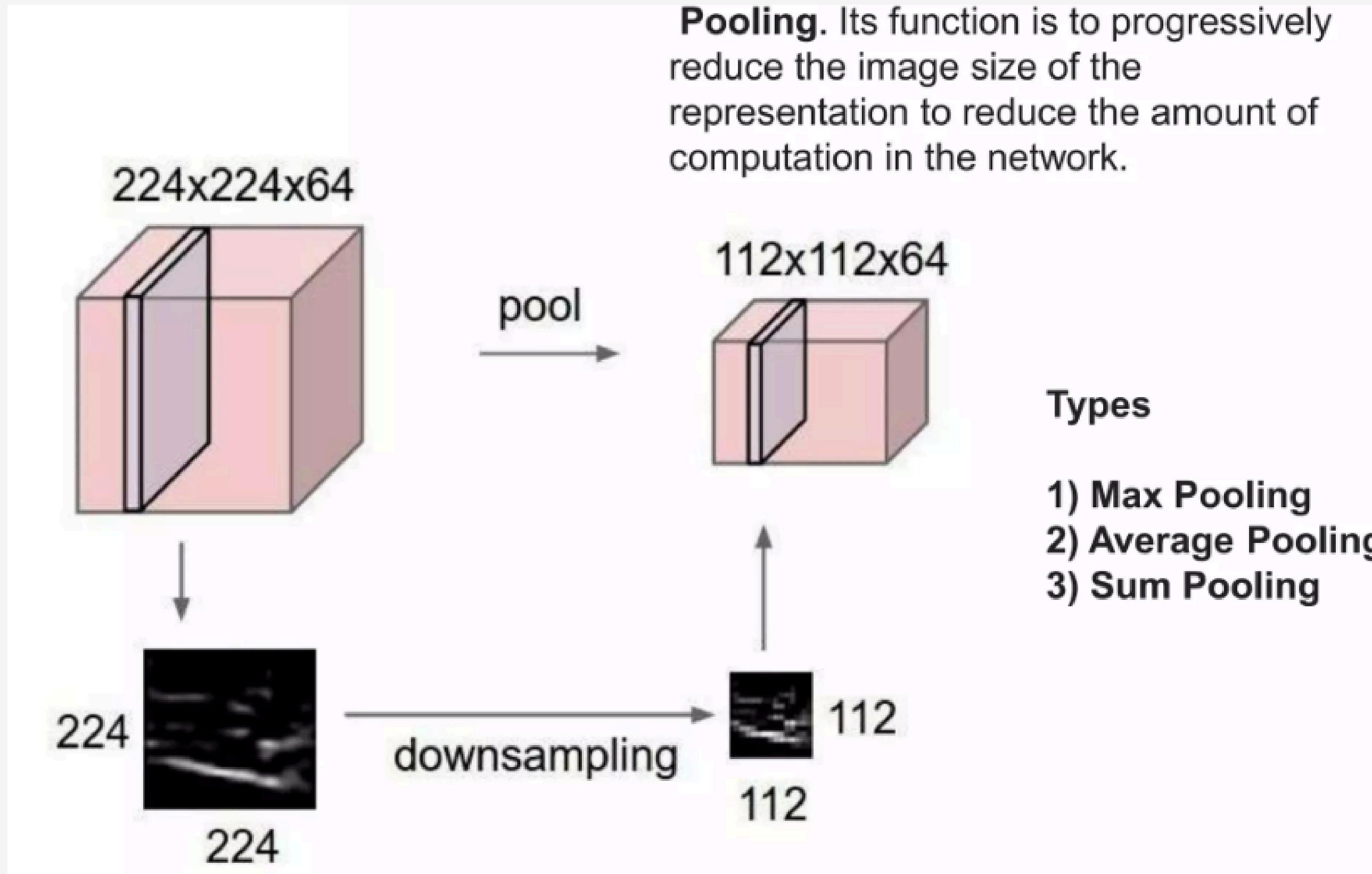
WHAT HAPPENS WHEN YOU DO A CONVOLUTION?



WHAT HAPPENS WHEN YOU DO A CONVOLUTION?



WHAT HAPPENS WHEN YOU MAXPOOL?



WHAT HAPPENS WHEN YOU MAXPOOL?

Feature Map

6	6	6	6
4	5	5	4
2	4	4	2
2	4	4	2

Max
Pooling

6	6
4	4

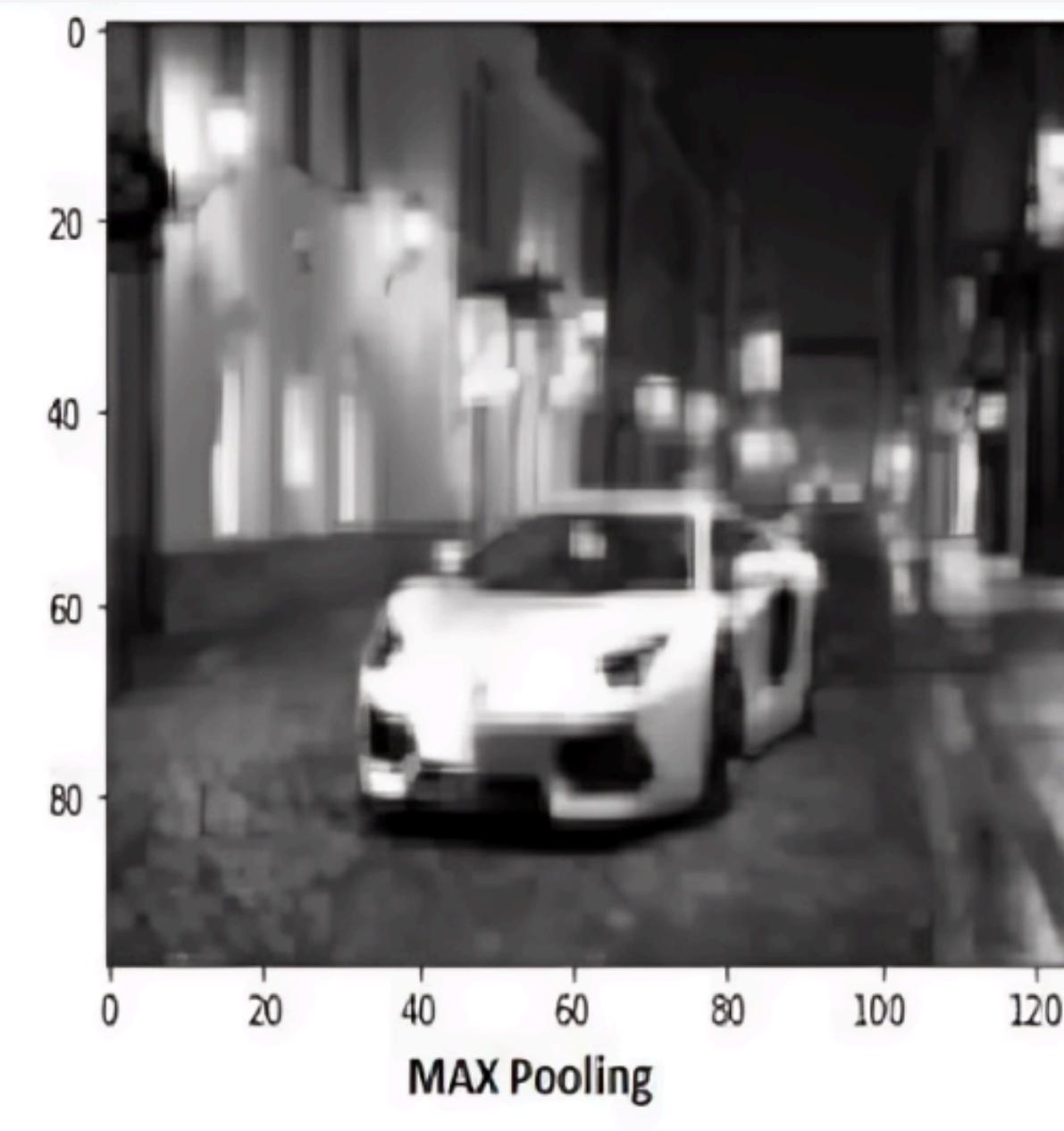
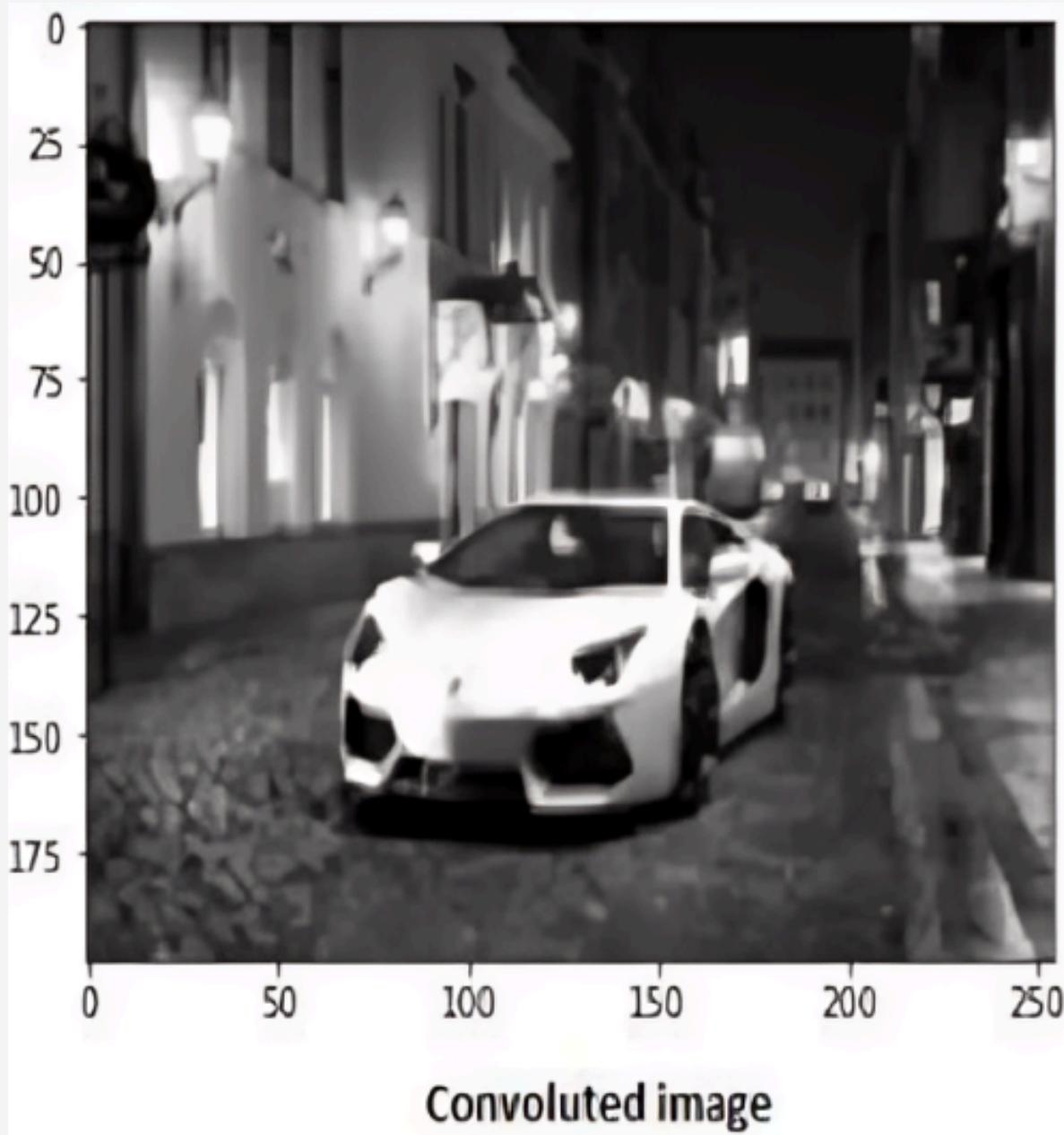
Average
Pooling

5.25	5.25
3	3

Sum
Pooling

21	21
12	12

WHAT HAPPENS WHEN YOU MAXPOOL?



WAYS TO CONVERT FEATURE MAP TO 1D

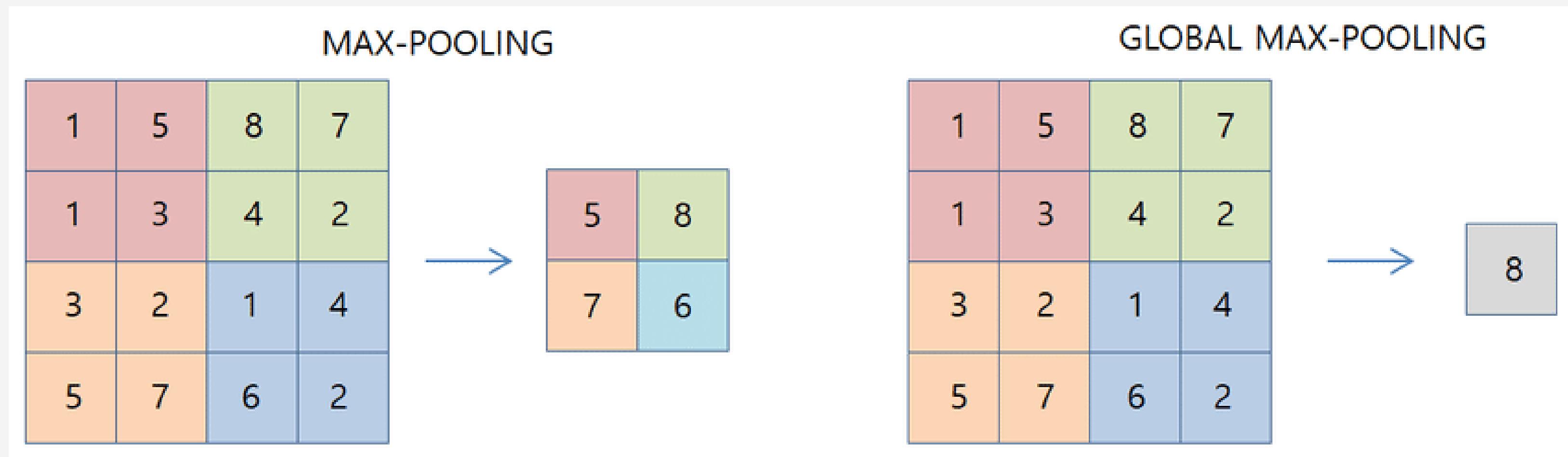
1	1	0
4	2	1
0	2	1

Pooled Feature Map

Flattening

1
1
0
4
2
1
0
2
1

WAYS TO CONVERT FEATURE MAP TO 1D



WAYS TO CONVERT FEATURE MAP TO 1D

Global Average Pooling

2	3	1	4
5	6	7	8
9	10	11	12
13	14	15	16



8.5

WHERE ARE CNNS PRIMARILY USED

- Image Classification (Disease Detection, Face Recognition, etc.)
- Object Detection (Spot Lesions from Medical Images, Potholes from live surveillance cameras, etc.)
- Semantic Segmentation (Tumor Segmentation from Medical Images)
- Instance Segmentation (Segment individual objects in crowded scenes, Identify Overlapping Nuclei, etc.)
- Image Generation (GANs, Style Transfer, etc.)

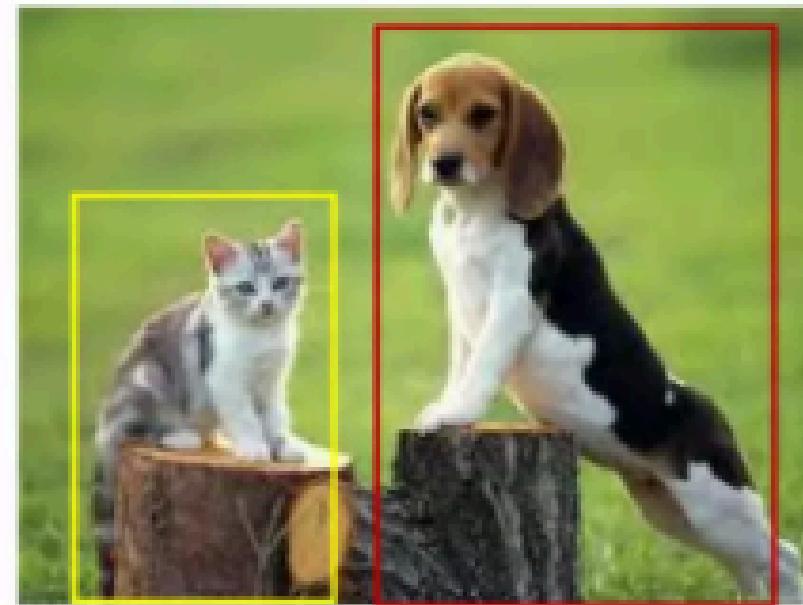
WHERE ARE CNNS PRIMARILY USED

Image Classification



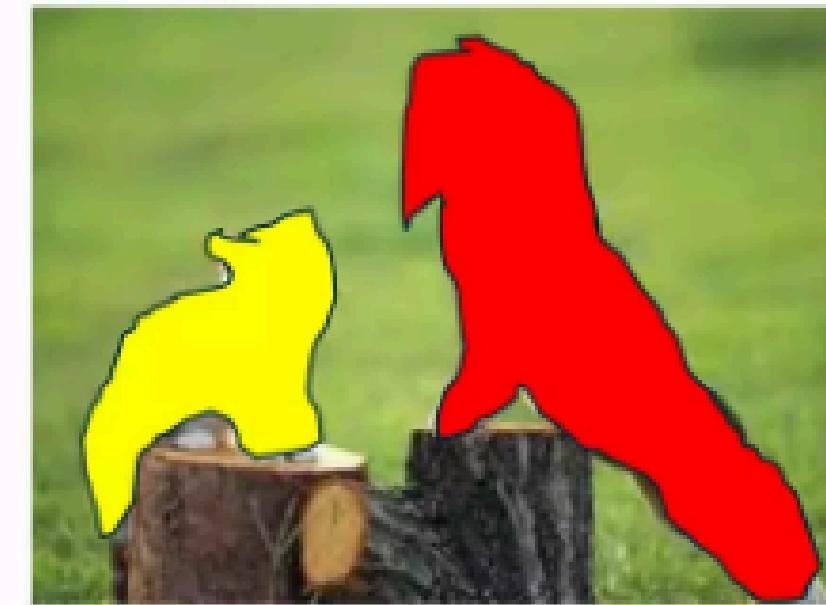
Cat or Dog? Entire Image

Object Detection



Classify Objects with Location

Segmentation

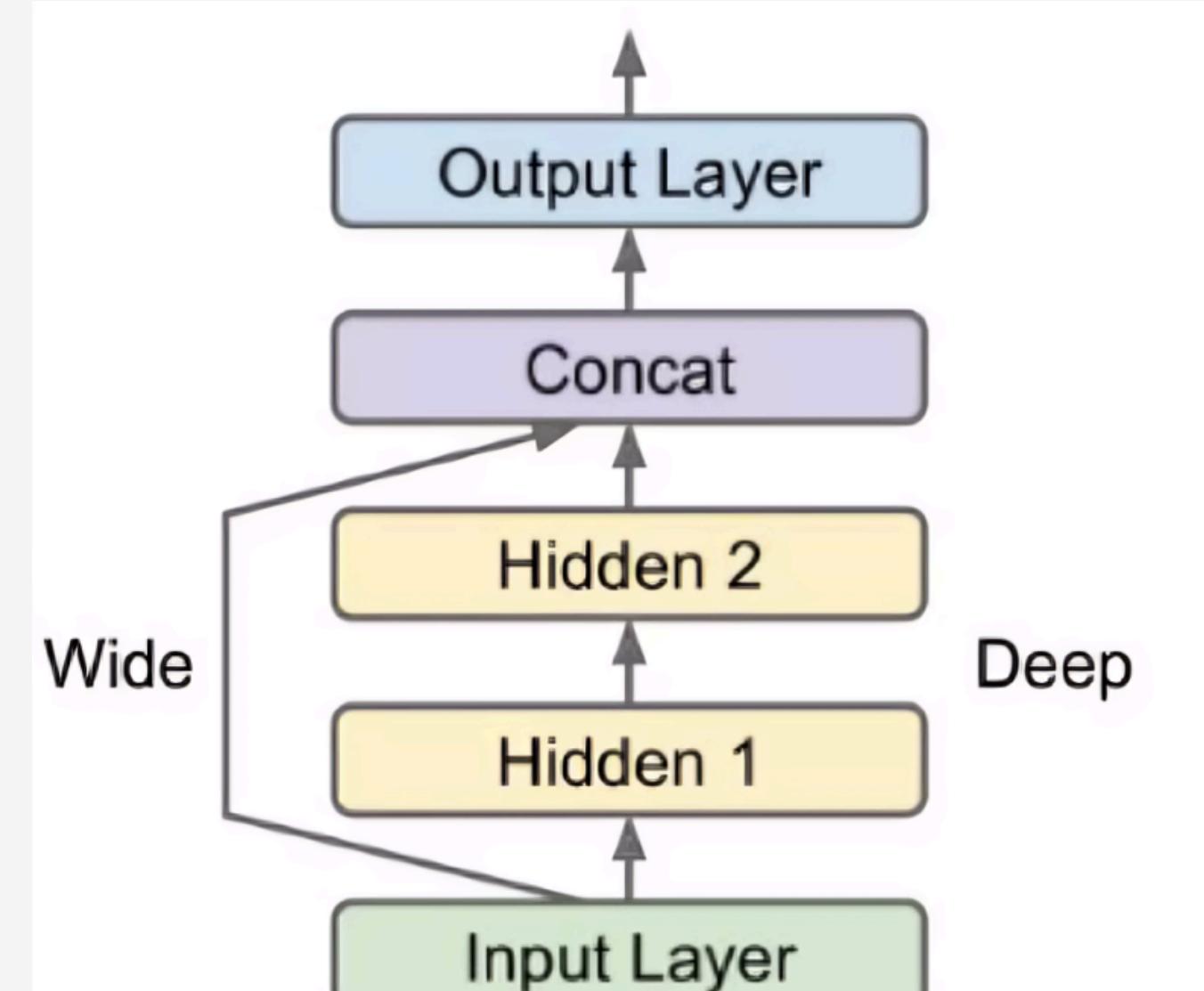


Classify each pixel to a class

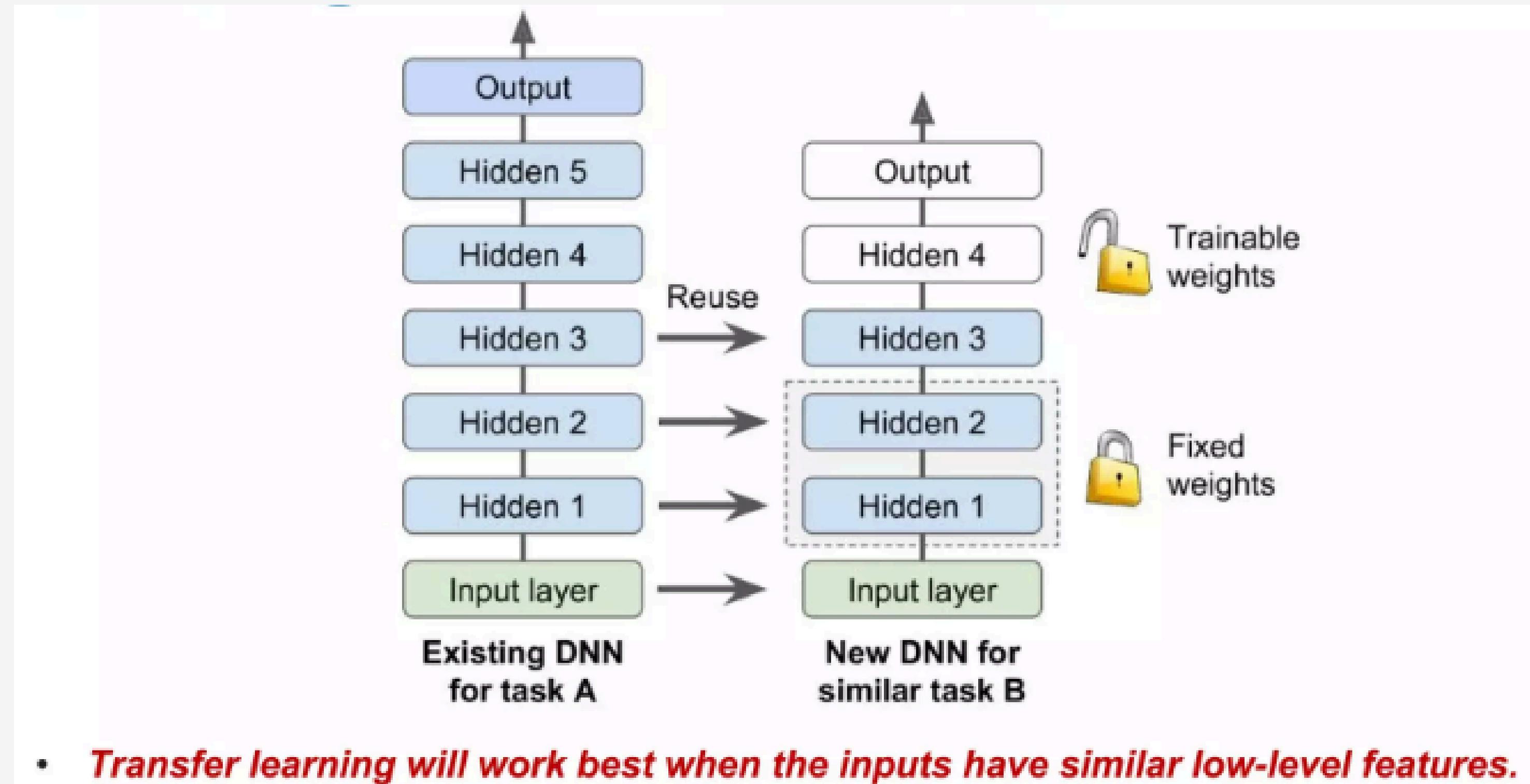
BUILDING COMPLEX MODELS - KERAS FUNCTIONAL API

FUNCTIONAL API

- The functional API is a way to create models that are more flexible than the Sequential API (`tf.keras.Sequential`)
- It can handle models with **non-linear topology**, models with **shared layers**, and models with **multiple inputs or outputs**.
- Enables us to build neural networks to learn both **deep patterns** (using the deep path) and **simple rules** (through the short path).



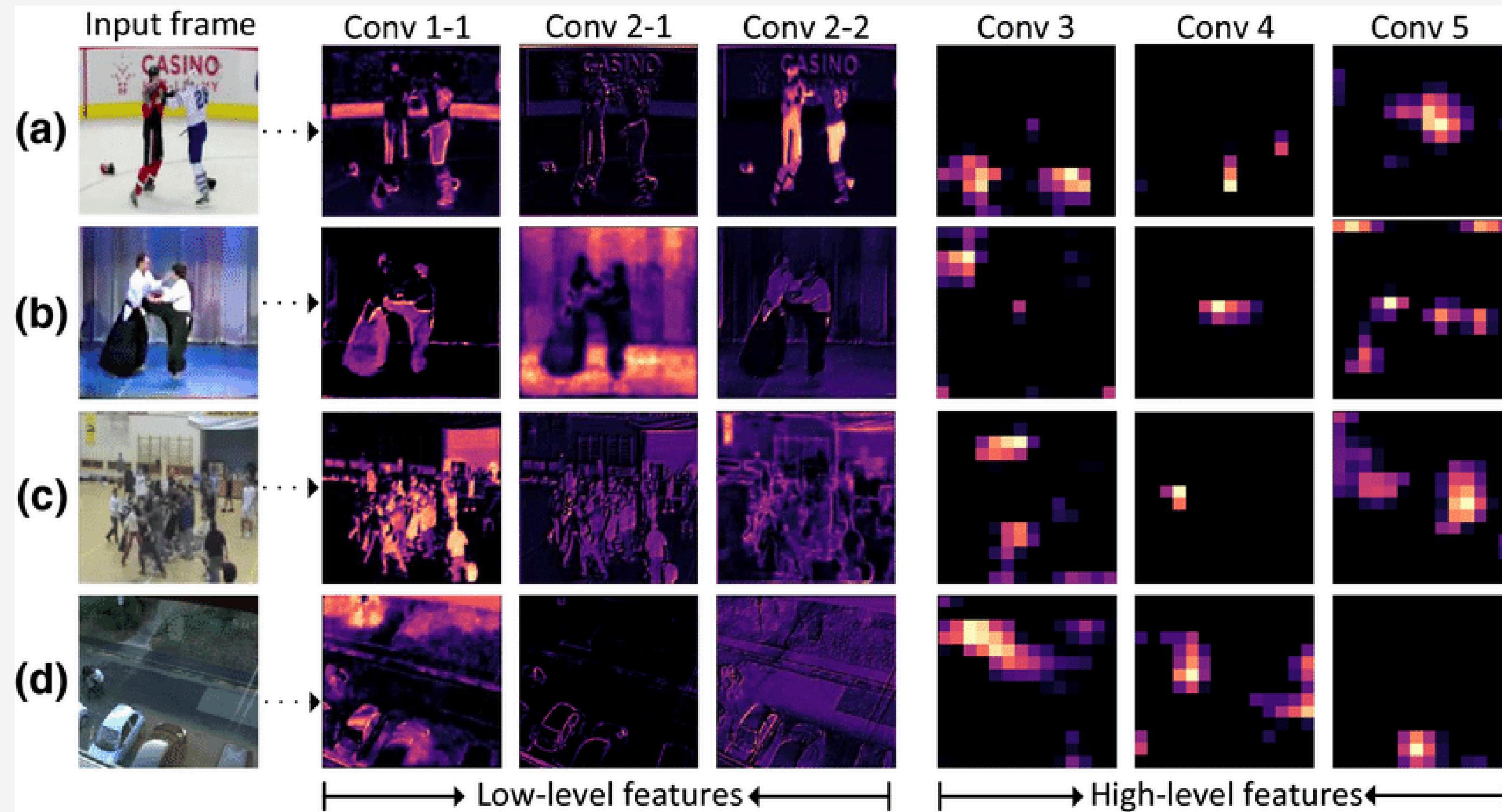
TRANSFER LEARNING



LOW-LEVEL AND HIGH-LEVEL FEATURES

- **Low-Level Features**
 - These features are typically extracted by the **early layers** of a CNN and include:
 - Edges
 - Textures
 - Basic Geometric Shapes
- **High-Level Features**
 - These features are extracted by **deeper layers** of the CNN and represent more complex patterns or objects:
 - Object Parts (eyes, nose)
 - Complex Patterns (combination of certain features)

LOW-LEVEL AND HIGH-LEVEL FEATURES



TRANSFER LEARNING MODELS

When To Use Which Model?

(not a rule of thumb, but its good to know the reason behind why each model was designed)

Xception

- Suitable for high-resolution image datasets where the model needs to handle a **large number of parameters** efficiently while capturing complex features.

Inception

- Ideal when **depth perception** is crucial, such as when understanding the distance between the lens and the object.

VGG

- Effective when fine-grained details are important and where **computational resources** are **less of a concern** due to its deeper architecture and large number of parameters.

TRANSFER LEARNING MODELS

When To Use Which Model?

(not a rule of thumb, but it's good to know the reason behind why each model was designed)

ResNet

- Beneficial for tasks where vanishing gradients are an issue:
 - ResNet - **residual connections** that help gradients flow through the network.

DenseNet

- DenseNet - each layer's output is concatenated with the outputs of **all preceding layers**.

EfficientNet

- Optimal for scenarios where **computational efficiency** and **accuracy** need to be balanced.

MobileNet

You can find the list of all the available transfer learning models under the TensorFlow Applications website: https://www.tensorflow.org/api_docs/python/tf/keras/applications

DATA AUGMENTATION

- Enhance model robustness and generalization by artificially expanding the training dataset.
- Benefits:
 - Increase Dataset Size
 - Prevent Overfitting
 - Improve Model Performance

HOW IS IT DONE?

Color and Illumination Examples

Original



Brightness



Contrast



Saturation



Color Temp-



Vignette



HOW IS IT DONE?

Geometric and Displacement Distortion Examples

Original



Horizontal Flip



Crop



Resize



Rotate



Vertical Flip



Warp



Fish-Eye Effect



HOW IS IT DONE?

Mixing Augmentations - Disruptive

Original



Blend



ColorTwist



Erase



Nonlinear
Blend



Crop And Patch



Glitch



Water

WHEN SHOULD YOU USE WHICH AUGMENTATION TECHNIQUE?

Not all augmentations apply to all datasets!

Original	Rotate 90	Rotate 180	Flip (mirror)
0 5	0 5	0 5	0 2
1 6	1 6	1 9	1 9
2 7	9 2	2 7	4 5
3 8	6 0	3 8	3 8
4 9	4 7	4 9	4 9

HANDS ON

Face Recognition using CNNs:

<https://colab.research.google.com/drive/1YbOcNwU1IzLLaDA2zlcgd-u-sitrzBGj8?usp=sharing>

Ensembling using CNNs:

https://colab.research.google.com/drive/1mjUmNeoPJuiBVc5z_Wr5rHLKNUIi0QaS?usp=sharing

CHALLENGE TASK

Task 1: Image Classification with CIFAR-10 Dataset

Train two artificial neural networks (ANNs) on the CIFAR-10 dataset using mini-batch gradient descent. Apply hyperparameter tuning to both models, using different regularization techniques for each. Evaluate the model's performance with visualizations of loss and accuracy, and display with reasoning as to which model performed better.

Task 2: Predicting House Prices with the Boston Housing Dataset

Implement an artificial neural network (ANN) for regression on the Boston Housing dataset, applying minibatch gradient descent, hyperparameter tuning, and various regularization techniques. Assess the model using Mean Squared Error and visualize training progress.

Dataset link: <https://www.kaggle.com/code/prasadperera/the-boston-housing-dataset>

You can load the datasets via TensorFlow/Kaggle API.

REFERENCES

- <https://www.analyticsvidhya.com/blog/2020/10/how-does-the-gradient-descent-algorithm-work-in-machine-learning/>
- <https://www.geeksforgeeks.org/ml-mini-batch-gradient-descent-with-python/>
- <https://machinelearningmastery.com/gentle-introduction-mini-batch-gradient-descent-configure-batch-size/>
- <https://www.geeksforgeeks.org/regularization-in-machine-learning/>
- <https://theaisummer.com/regularization/>
- <https://medium.com/data-science-365/how-to-apply-l1-and-l2-regularization-techniques-to-keras-models-da6249d8a469>