PIF
صندوق الاستثمارات العامة

LEVEL 2: Introduction to Computer Vision and Convolutional Neural Networks

Day 4

Course Outline

- Computer Vision
- Computer Vision Tasks
- Image representation
- Convolutional Neural Networks
- Padding
- Pooling Layer
- Activition
- SOTA CNN Architectures
- Data Handling

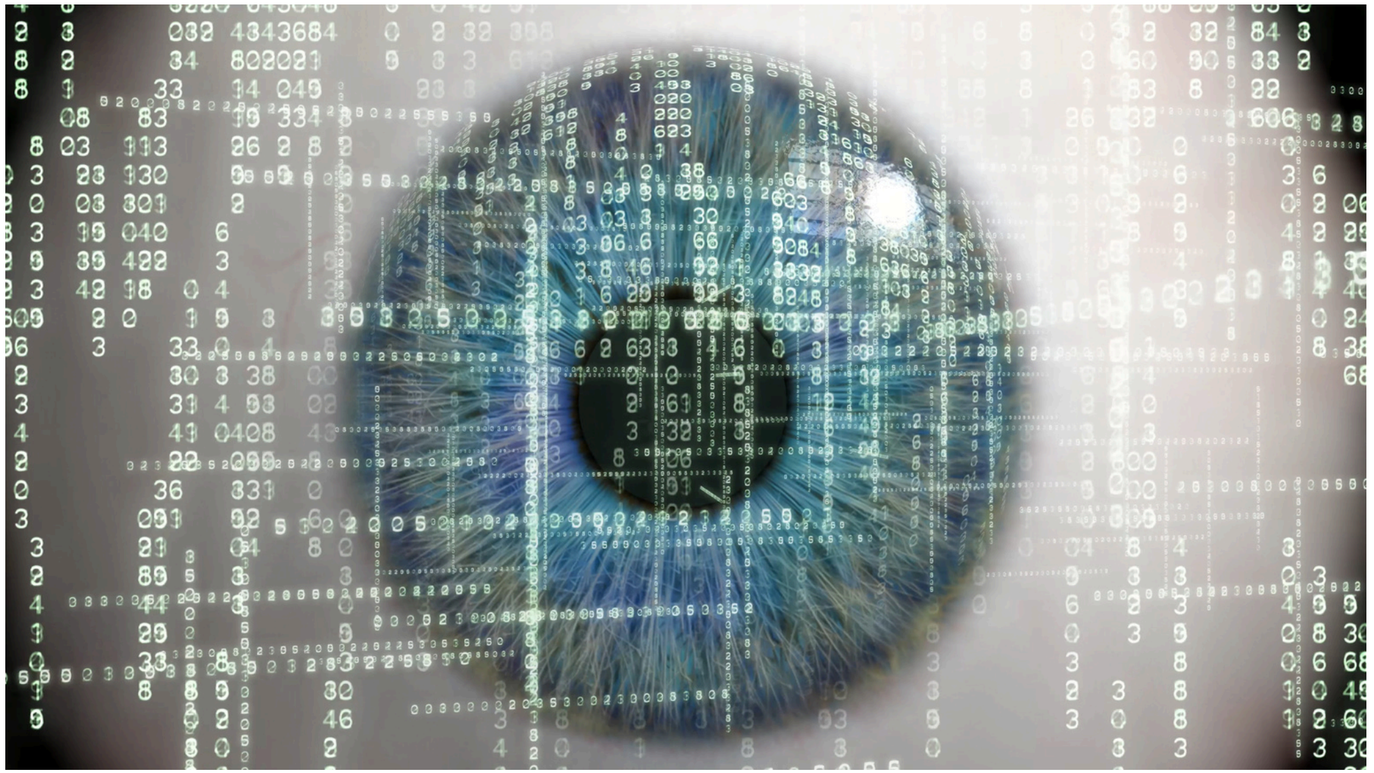
Learning Objectives

- Explain Computer Vision tasks such as image classification, object detection, and segmentation.
- Represent images numerically, including pixel values, channels, and feature maps.
- Describe and apply Convolutional Neural Networks (CNNs) for visual data processing.
- Analyze the role of padding and its impact on spatial dimensions and feature extraction.
- Differentiate pooling layers (max, average, global pooling) and explain their purpose.
- Select and apply activation functions and understand their effect on model learning.
- Compare SOTA CNN architectures (e.g., VGG, ResNet, EfficientNet) in terms of depth, efficiency, and performance.

Computer Vision

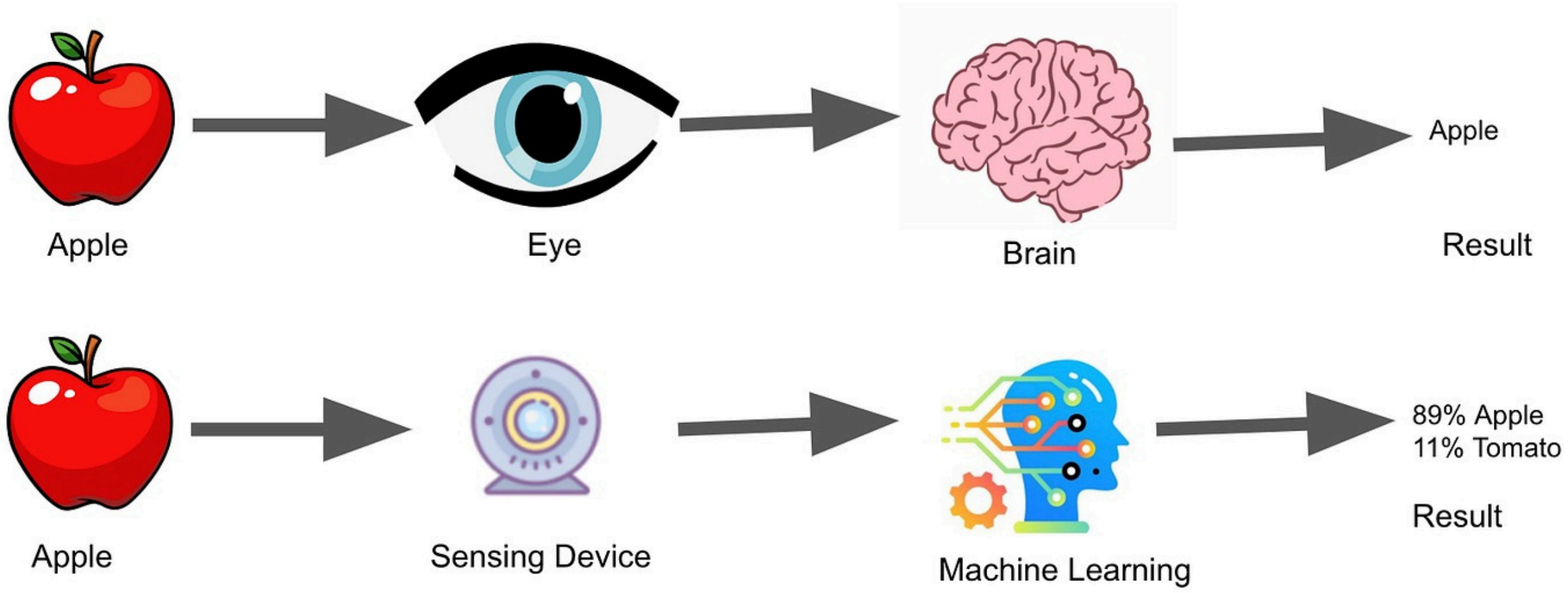
Computer Vision

- Computer Vision (CV) enables machines to see, understand, and interpret the visual world.



Computer Vision

- Computer Vision (CV) is inspired by how humans perceive and process visual information.



Computer Vision Tasks

Computer Vision Tasks

- Image Classification

The class in this case is a cat

The class in this case is a dog

Computer Vision Tasks

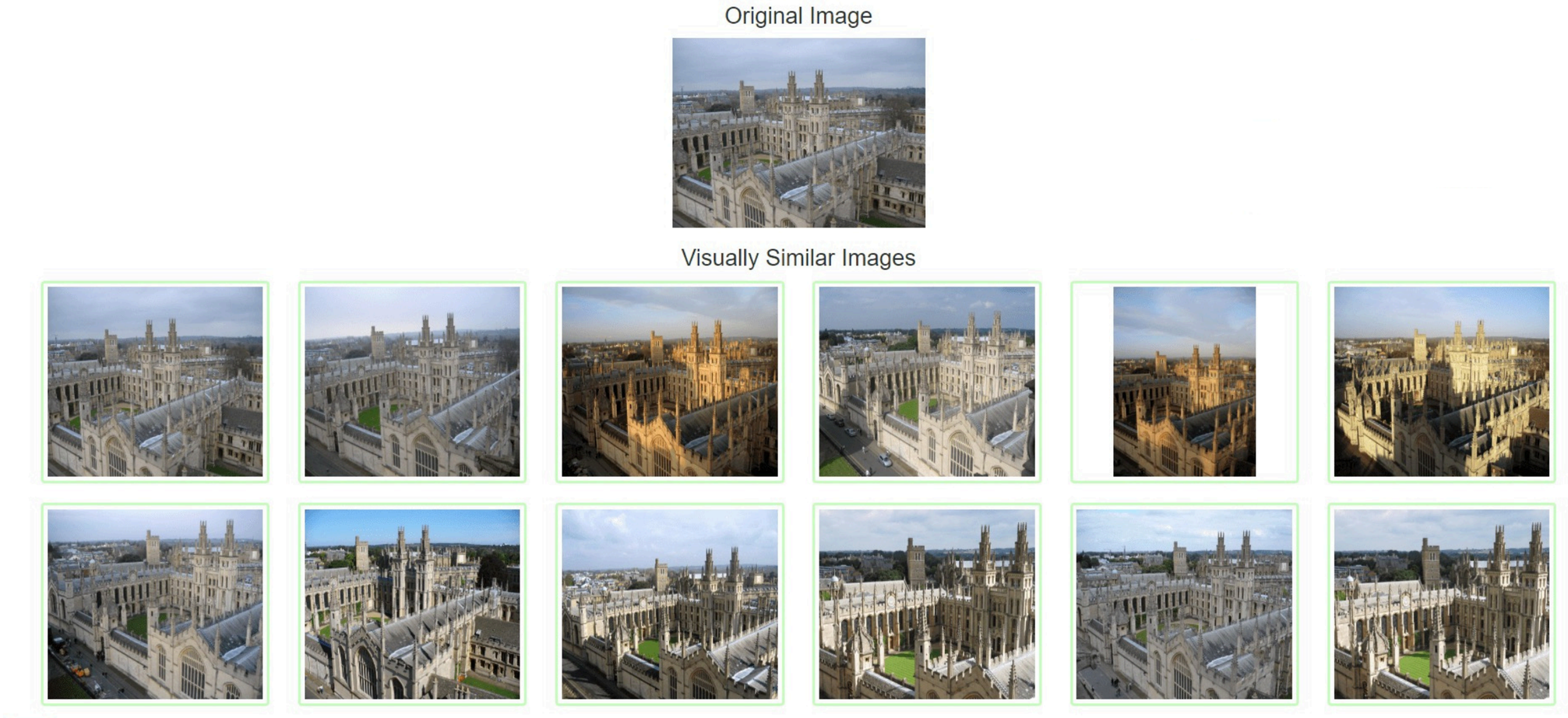
- Object Detection and Localization

Class: No mask detected

Class: A mask is detected in the image

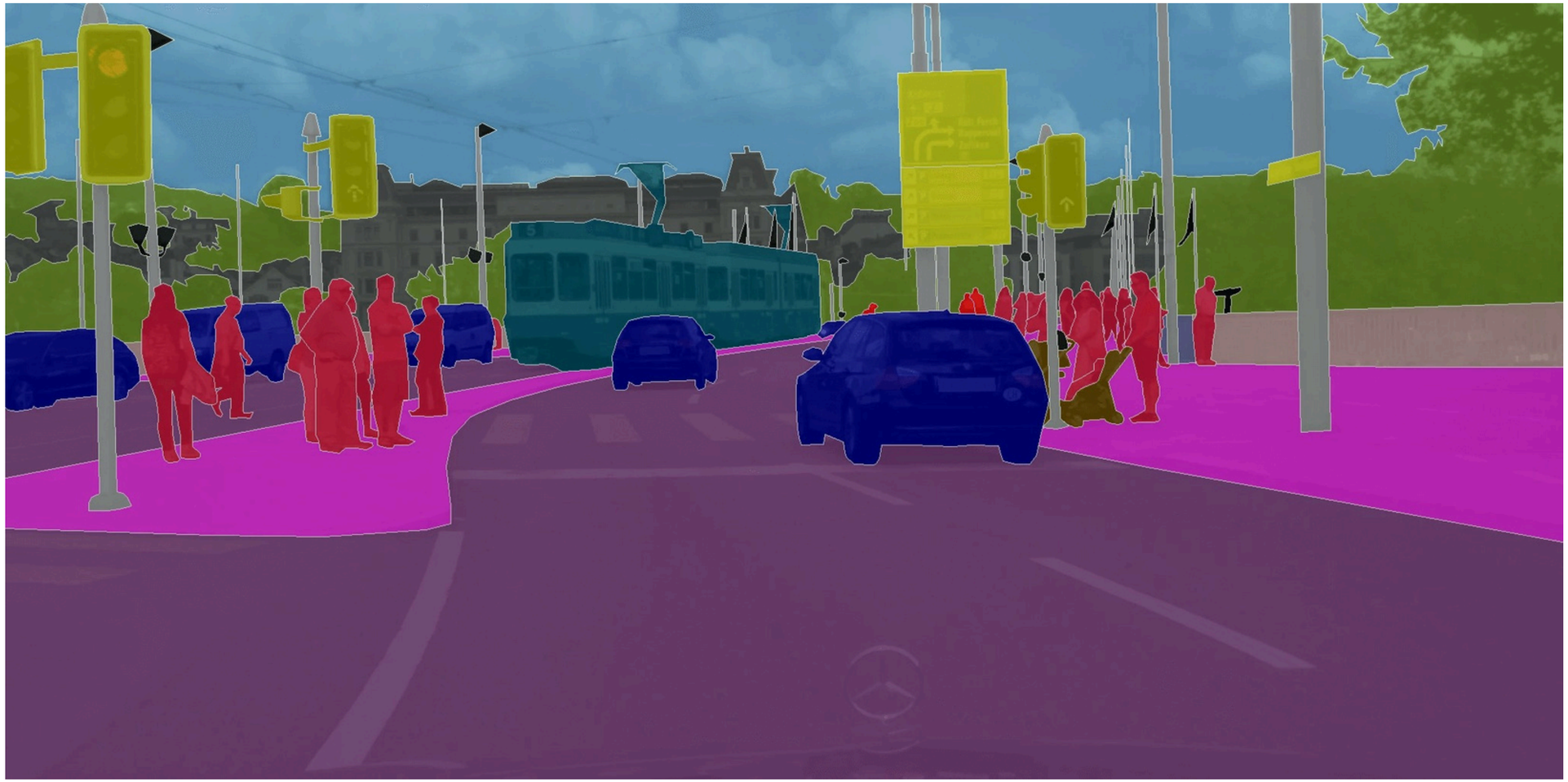
Computer Vision Tasks

Image Retrieval



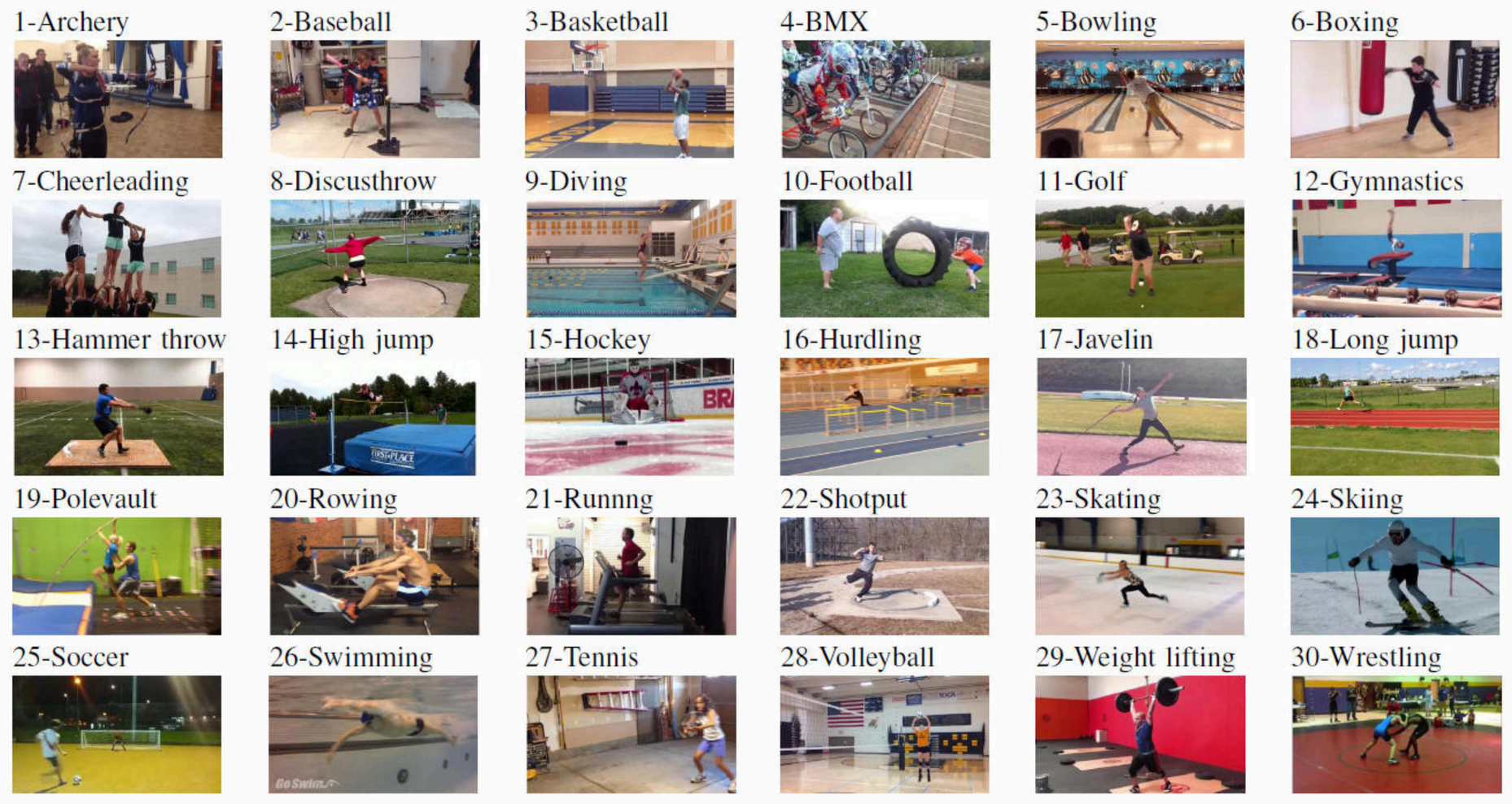
Computer Vision Tasks

- Image Segmentation



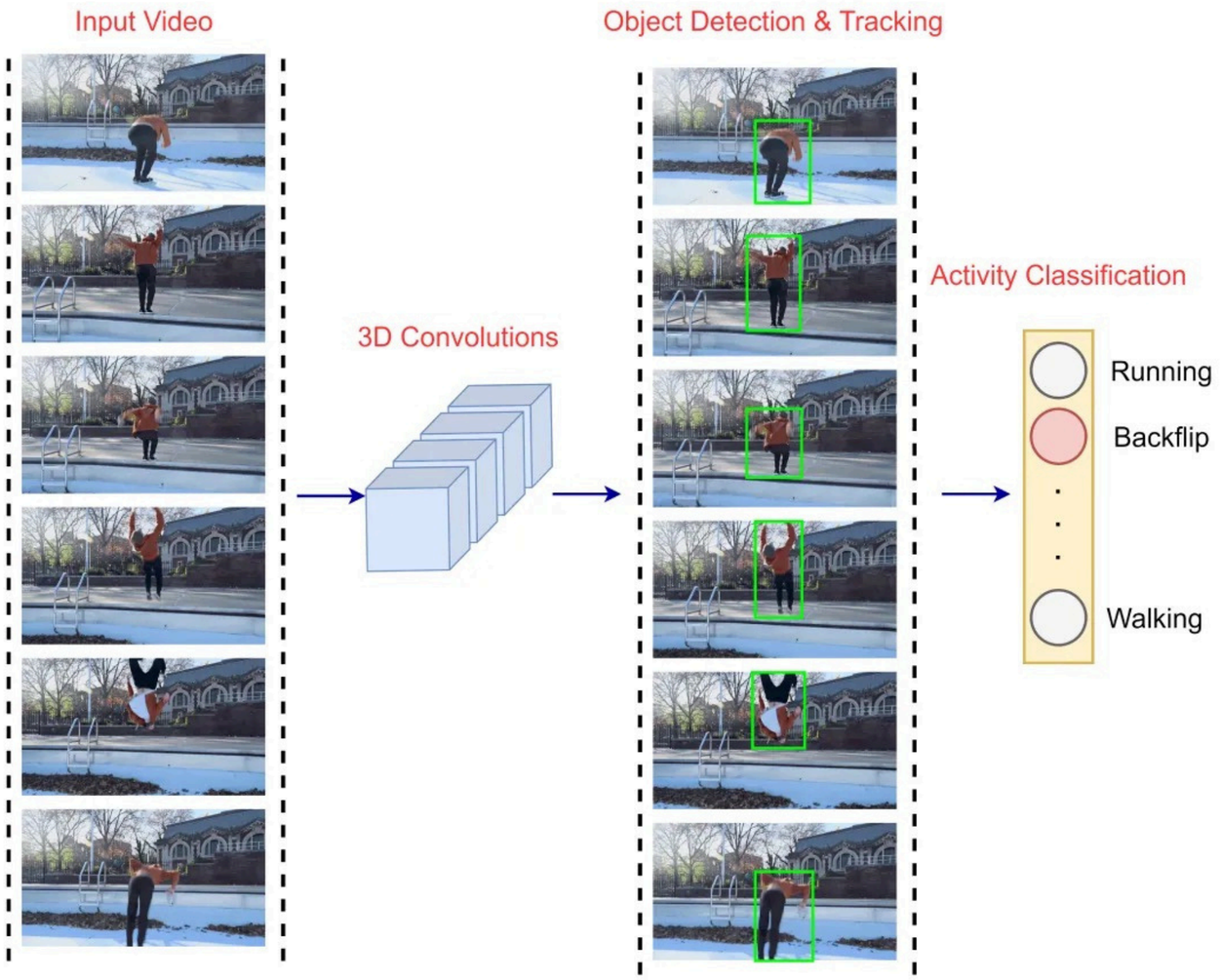
Computer Vision Tasks

Video Classification



Computer Vision Tasks

Activity Recognition



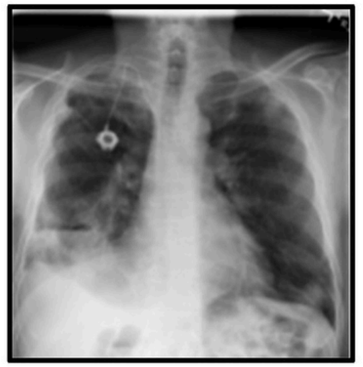
Computer Vision Tasks

- Medical Imaging

(a) Atelectasis

(b) Effusion

(c) Infiltration

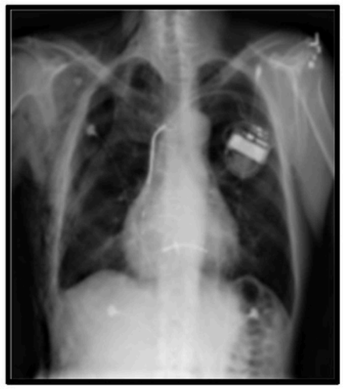


(d) Mass

(e) Nodule

(f) Normal

(g) Others



(h) Pneumothorax

Computer Vision Tasks

- Image Captioning

A young boy is playing basketball. 	Two dogs play in the grass. 	A dog swims in the water. 	A little girl in a pink shirt is swinging. 

Computer Vision Tasks

- Image Captioning

An image generated by DALL-E 2, from the prompt "Teddy bears working on new AI research underwater with 1990s technology"

Computer Vision Tasks

- 3D Vision

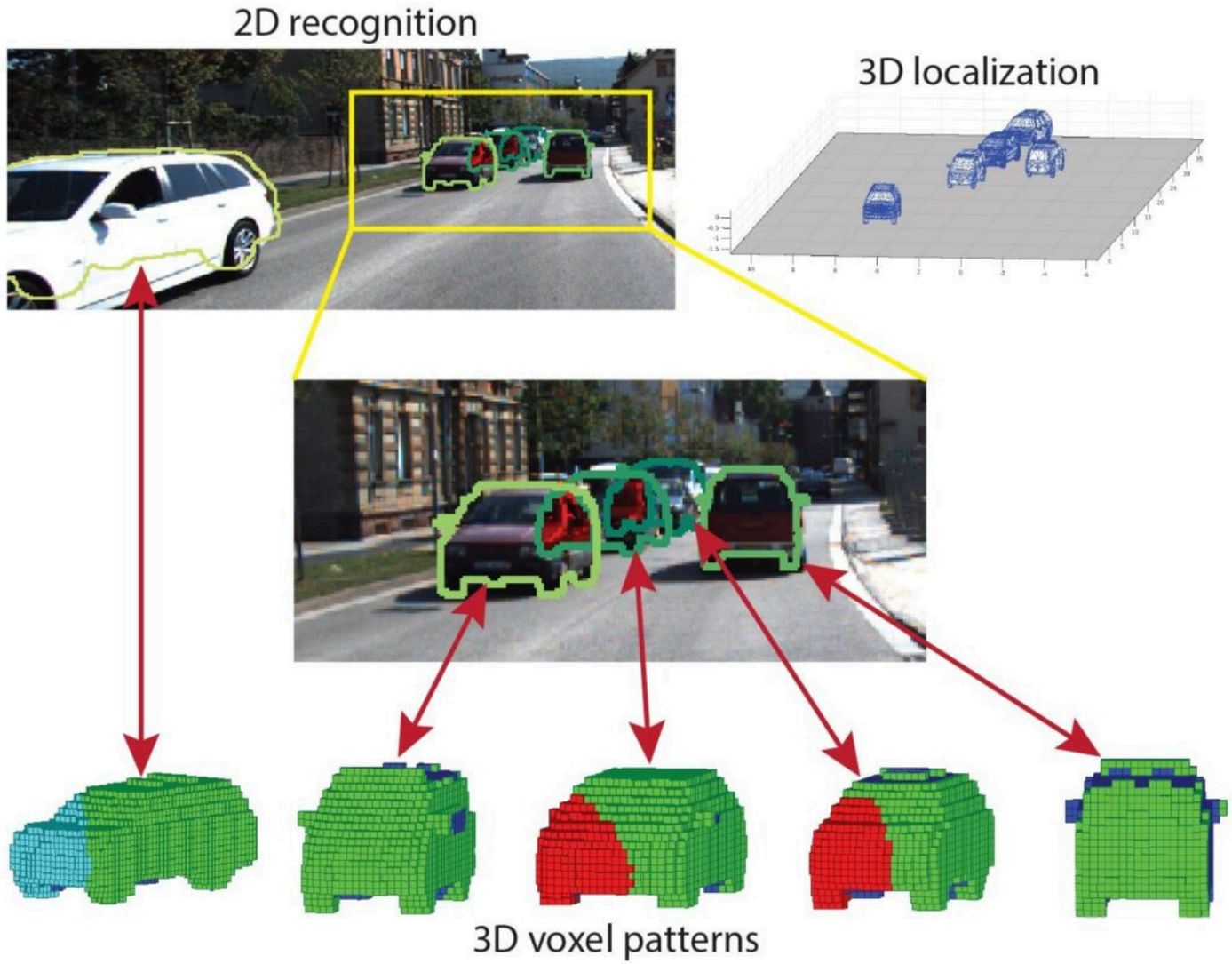


Image Representation

Image Representation

- Black\white
- Grey images

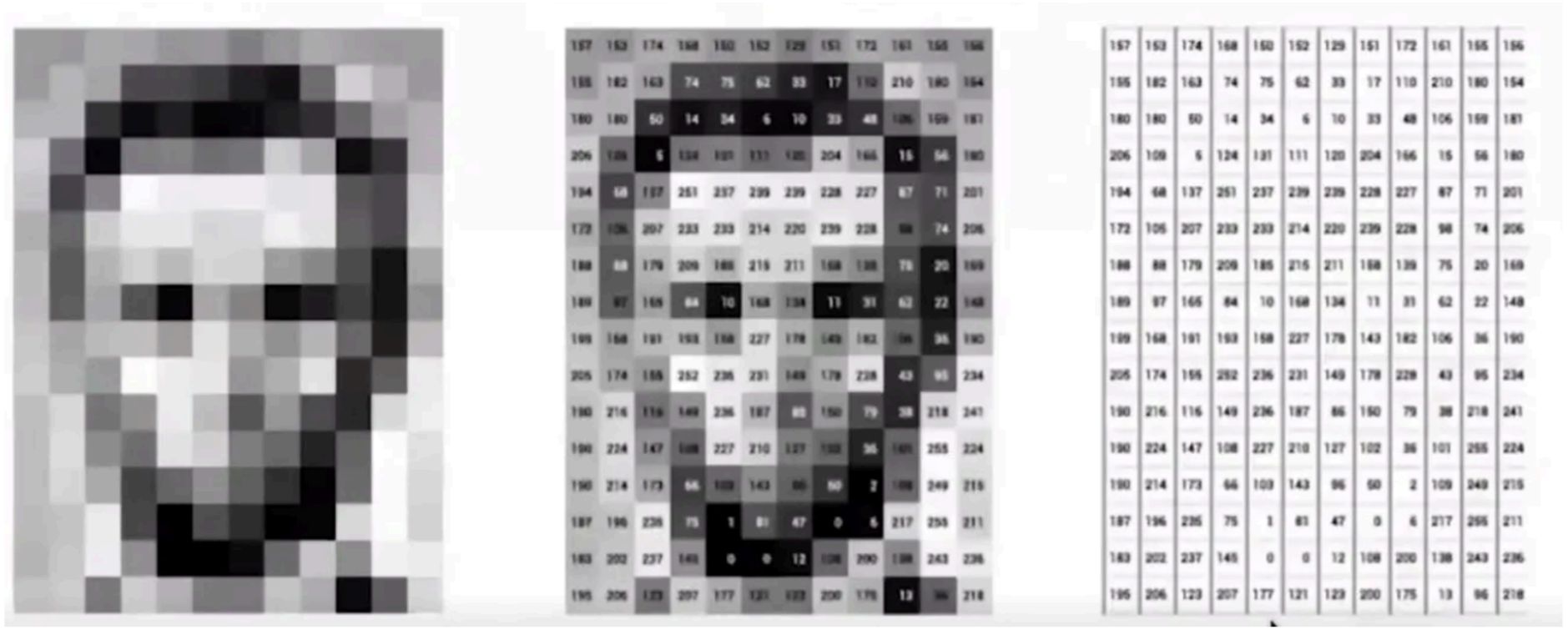
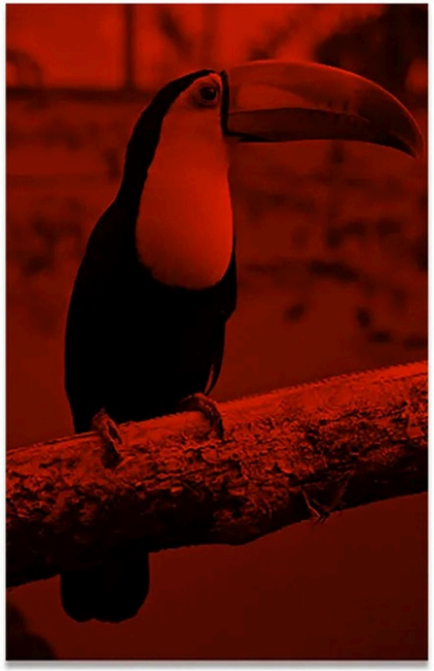


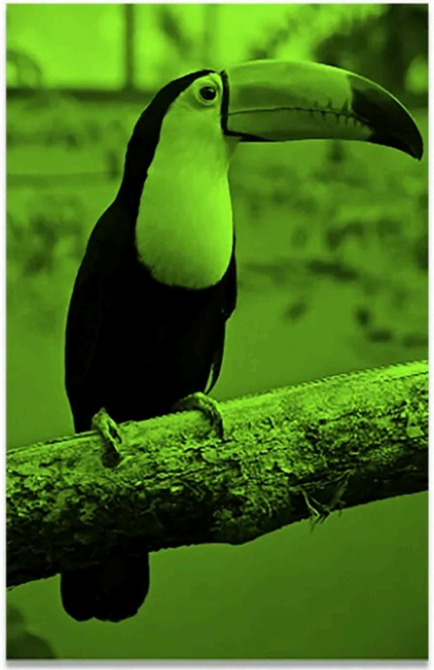
Image Representation

- Colored Images

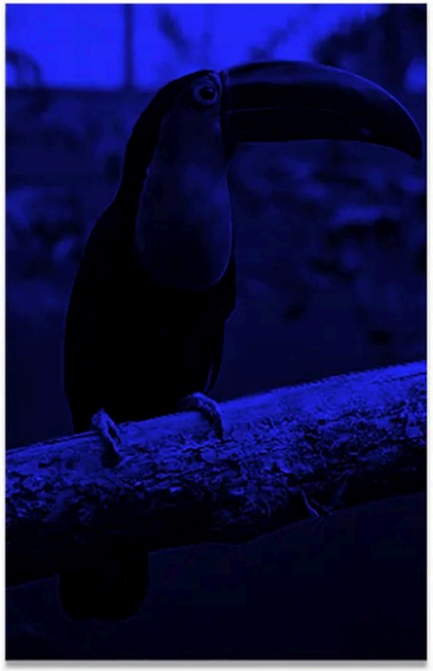
Original Image



Red Channel



Green Channel



Blue Channel

Image Representation

- Colored Images

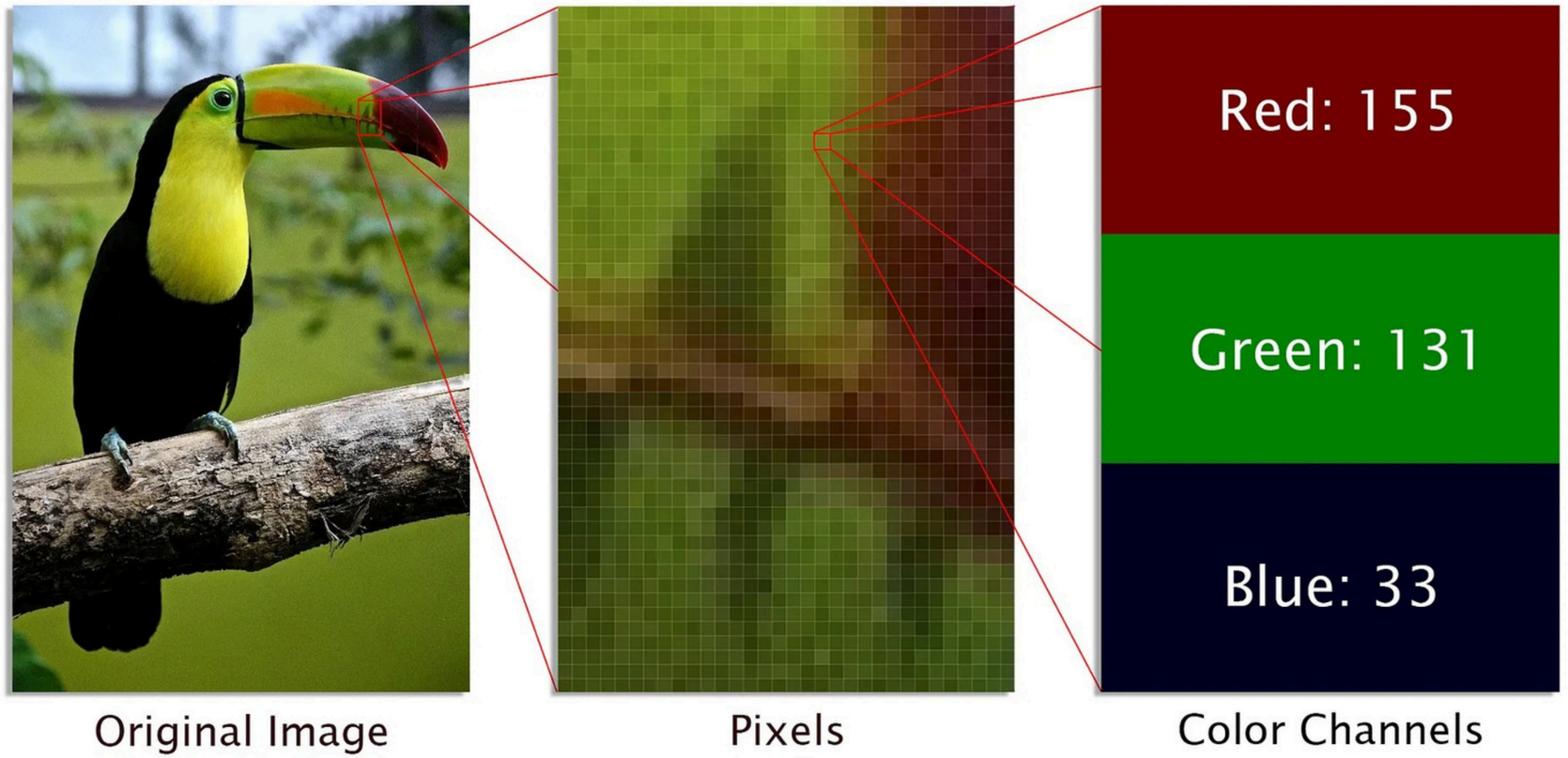
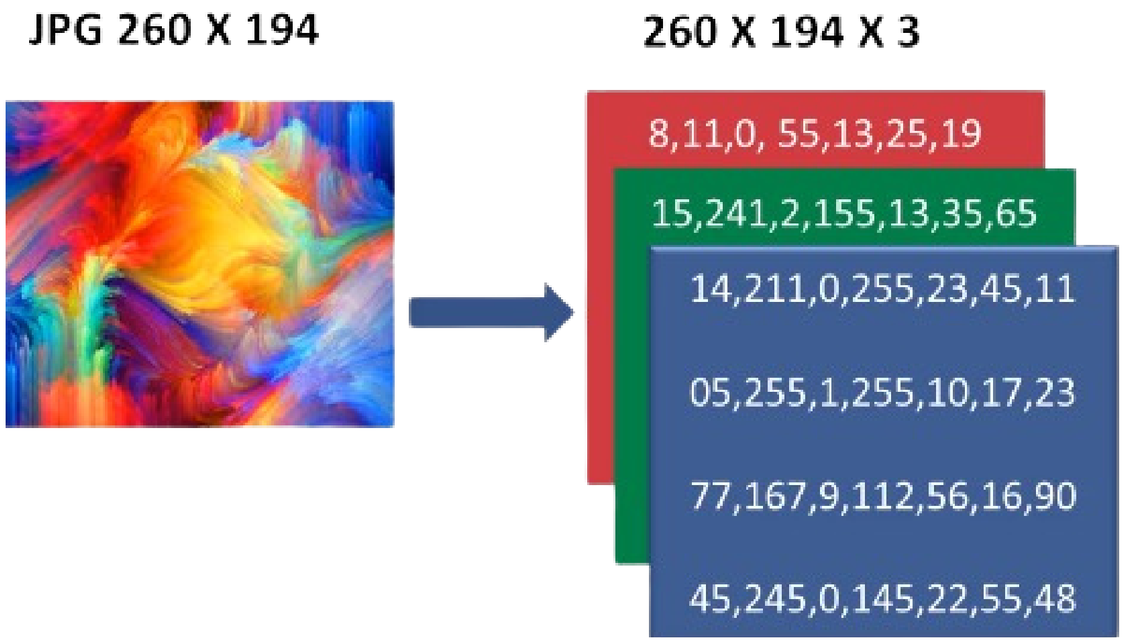


Image Representation

Tabular = structured rows & columns

Size (m^2)	Rooms	Type	Parking	Price (SAR)
120	3	Apartment	1	950,000
180	4	Villa	2	1,350,000
140	3	Apartment	1	980,000
220	5	Villa	3	1,850,000
160	4	Townhouse	2	1,200,000

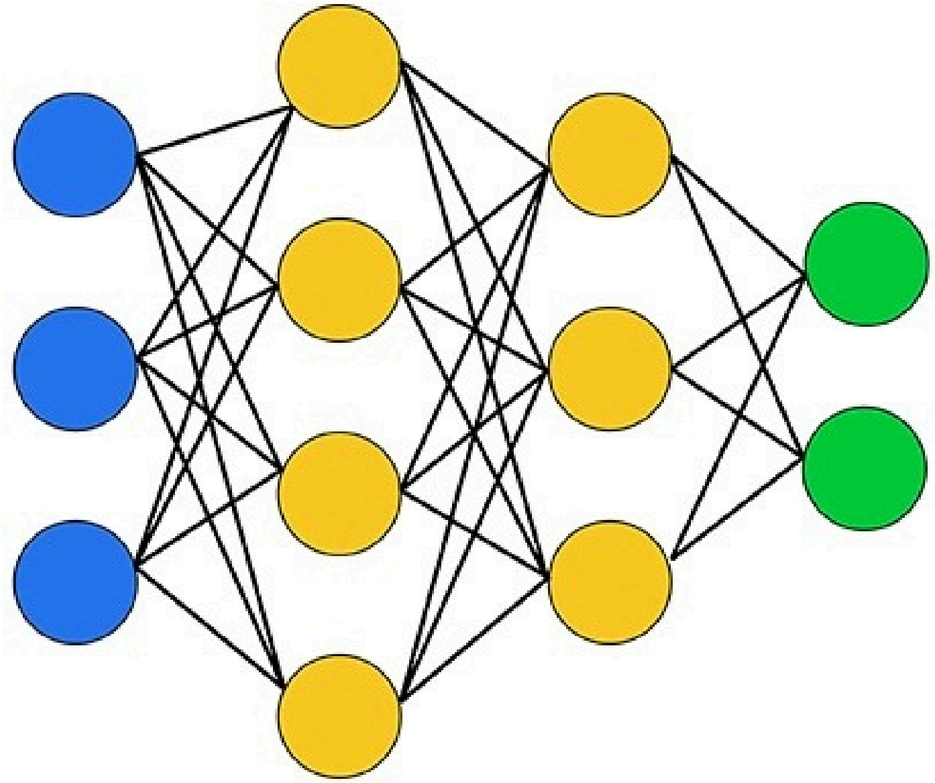
Image = unstructured pixel grid



Convolutional Neural Networks

Convolutional Neural Networks

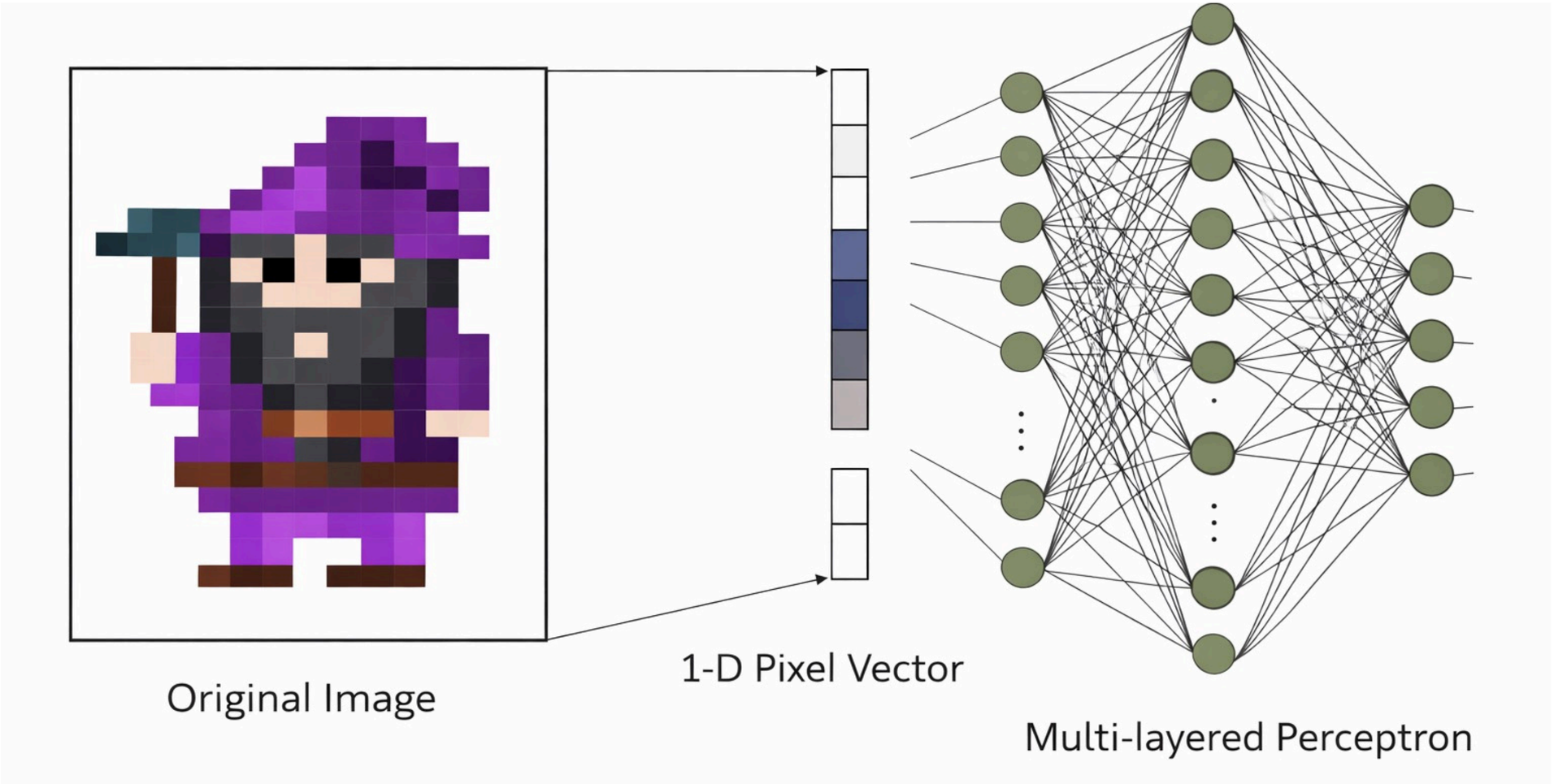
- How to build vision models?



- But can we apply the same approach to images?

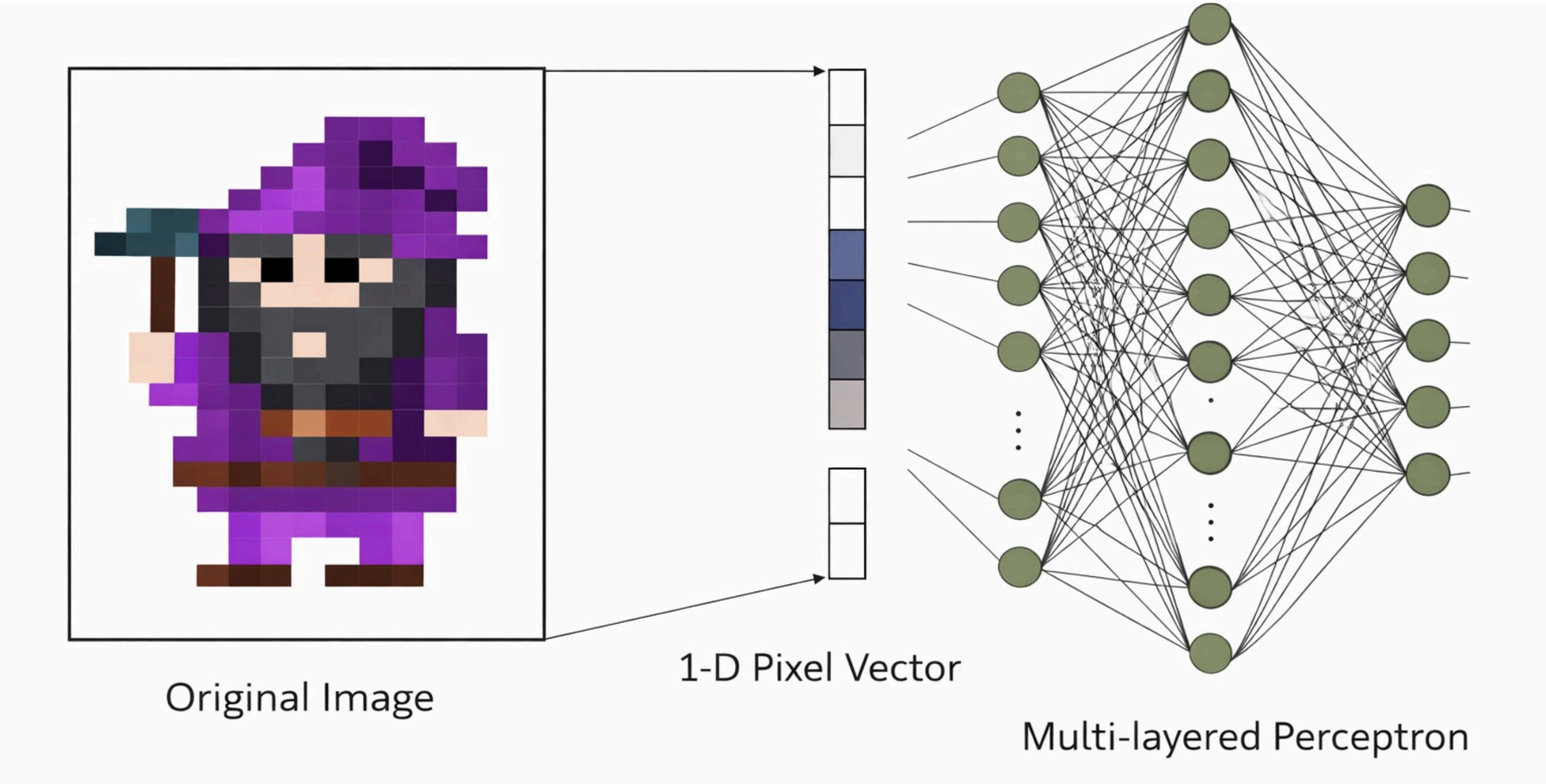
Convolutional Neural Networks

- How to build vision models?



Convolutional Neural Networks

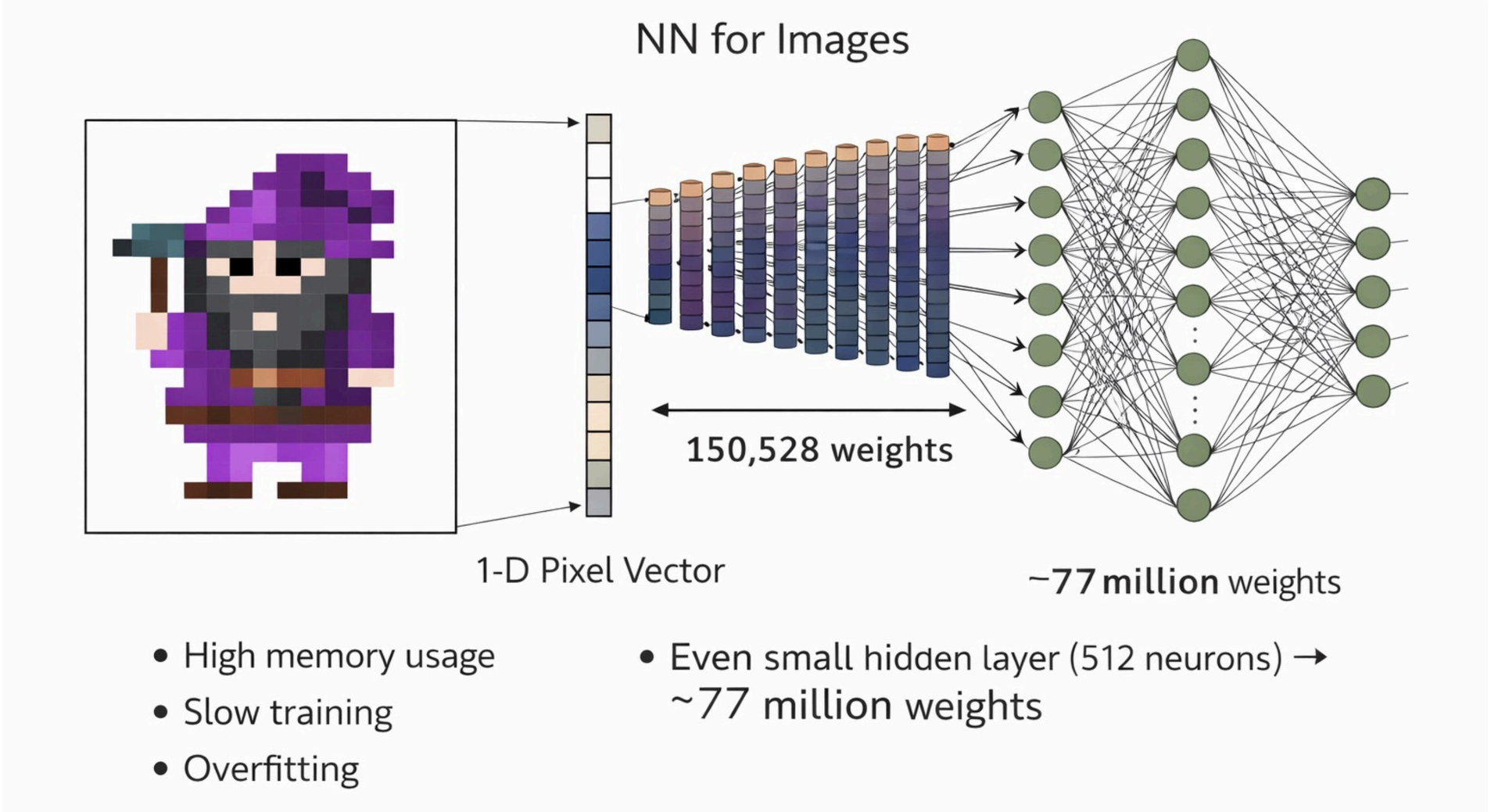
- How to build vision models?



- But!

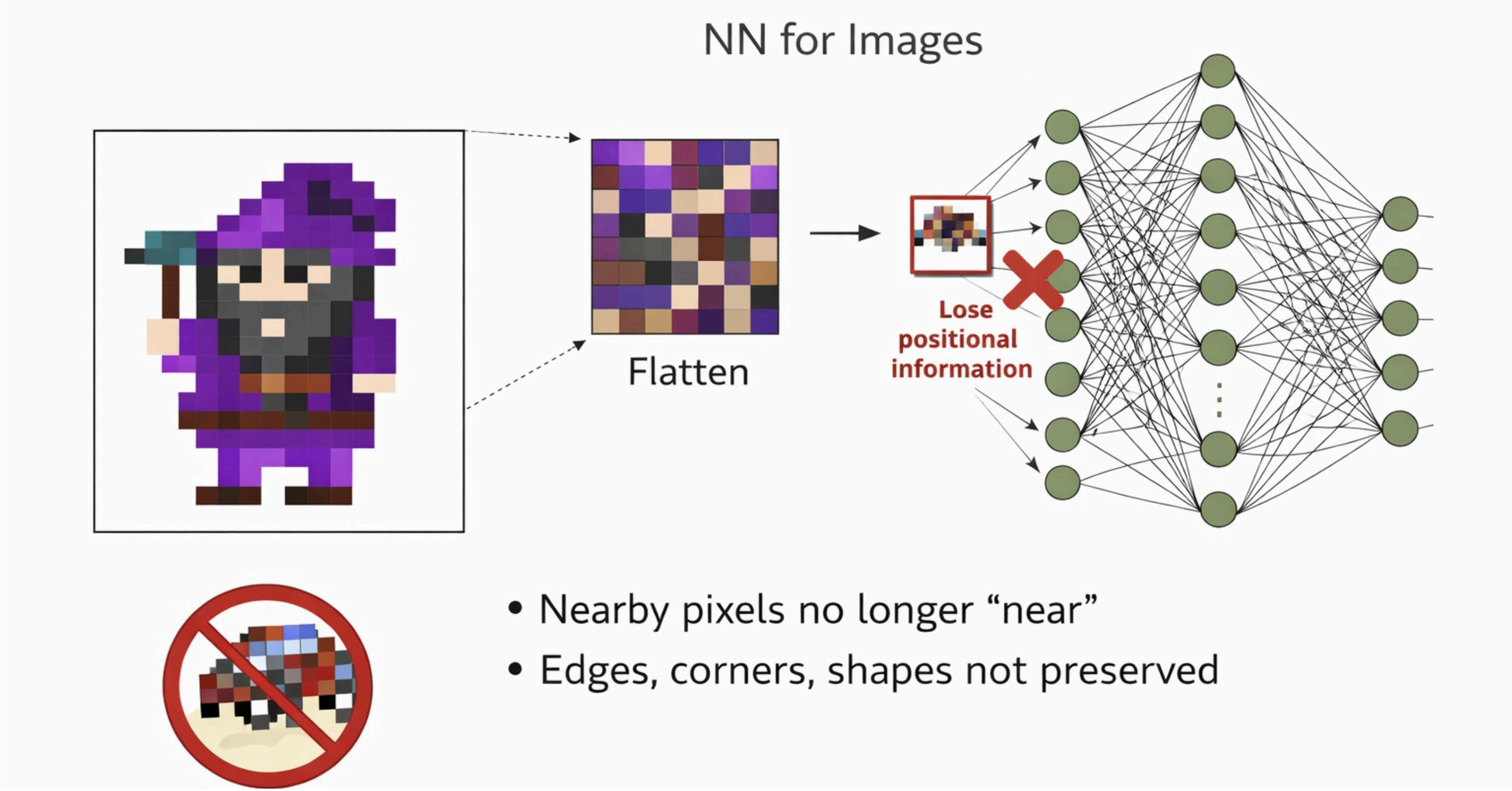
Convolutional Neural Networks

- Huge number of parameters



Convolutional Neural Networks

- Loss of spatial structure



Convolutional Neural Networks

- Little or no invariance to shifting, scaling, and other forms of distortion

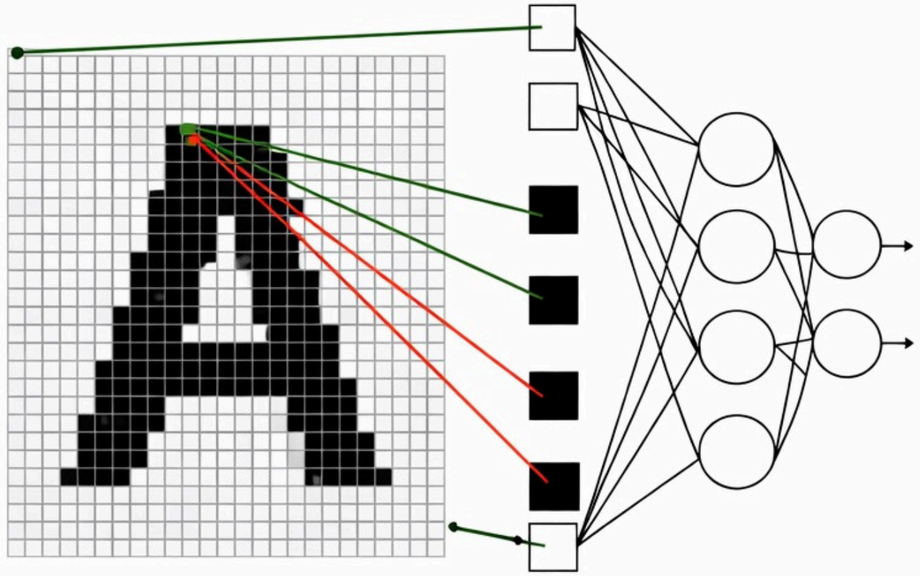


Figure 2: Original "A" character.

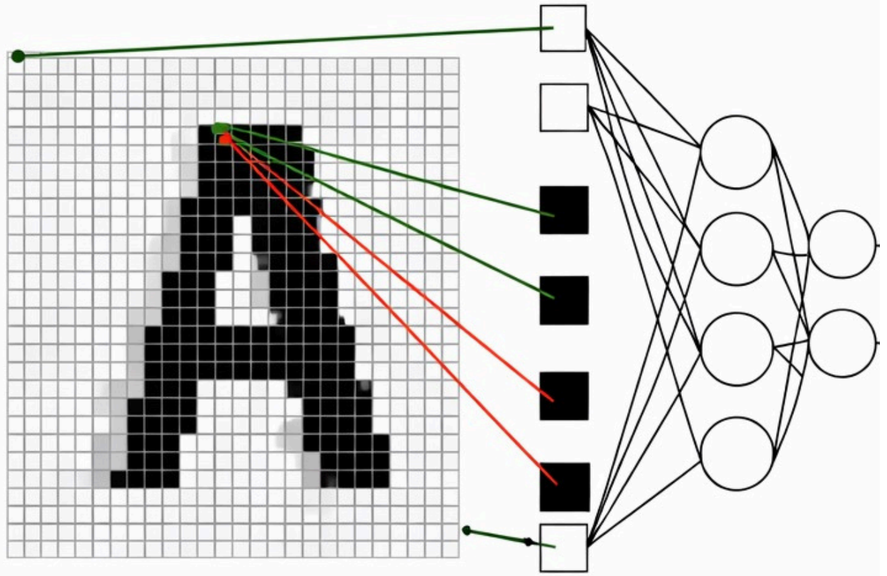
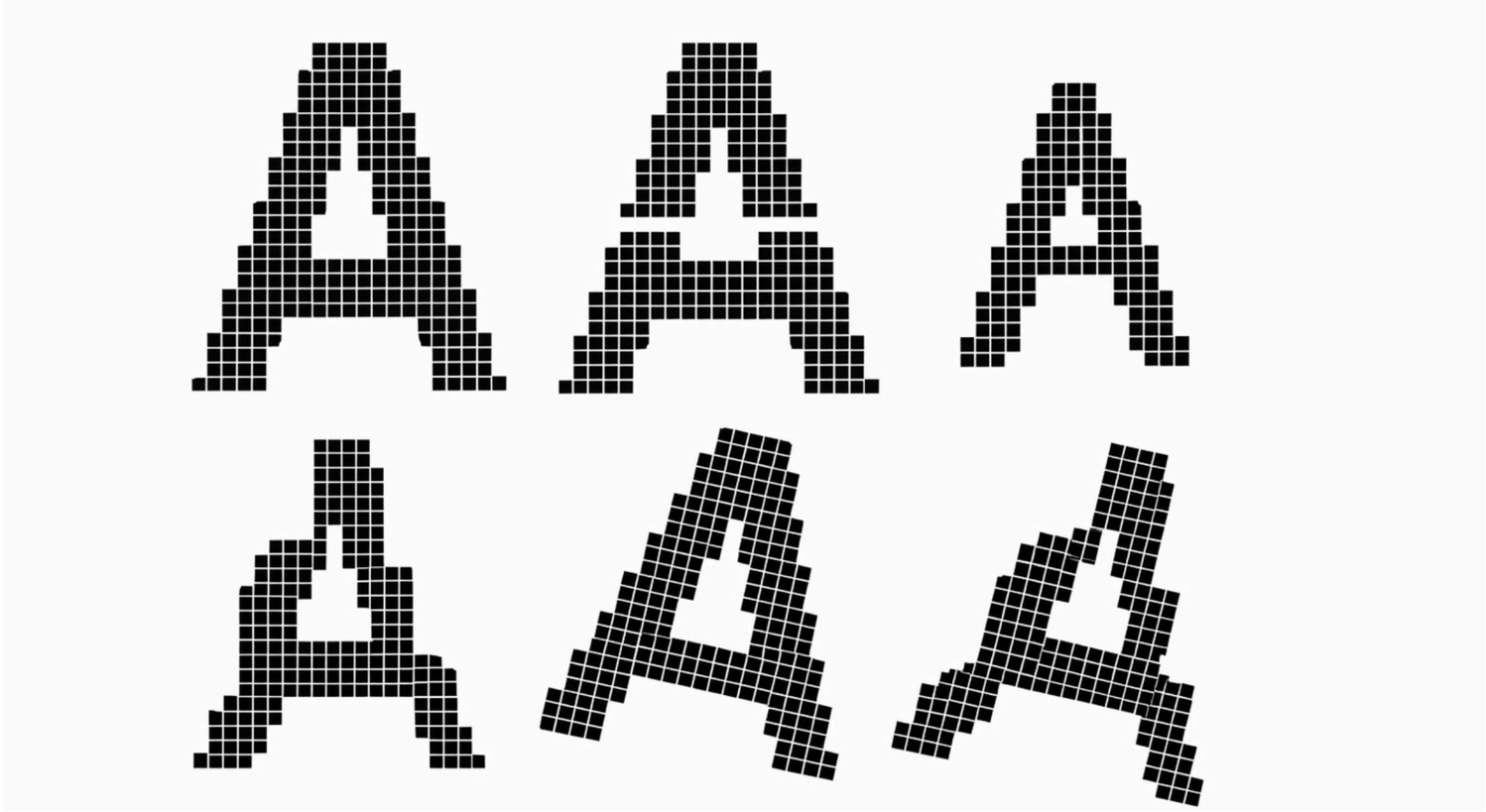


Figure 3: Shifted "A" character.

Convolutional Neural Networks

- It treats pixels as independent features rather than structured patterns.
  - Black and white patterns $2^{32\times32}$
  - Grayscale patterns $2^{32\times32}$



Convolutional Neural Networks

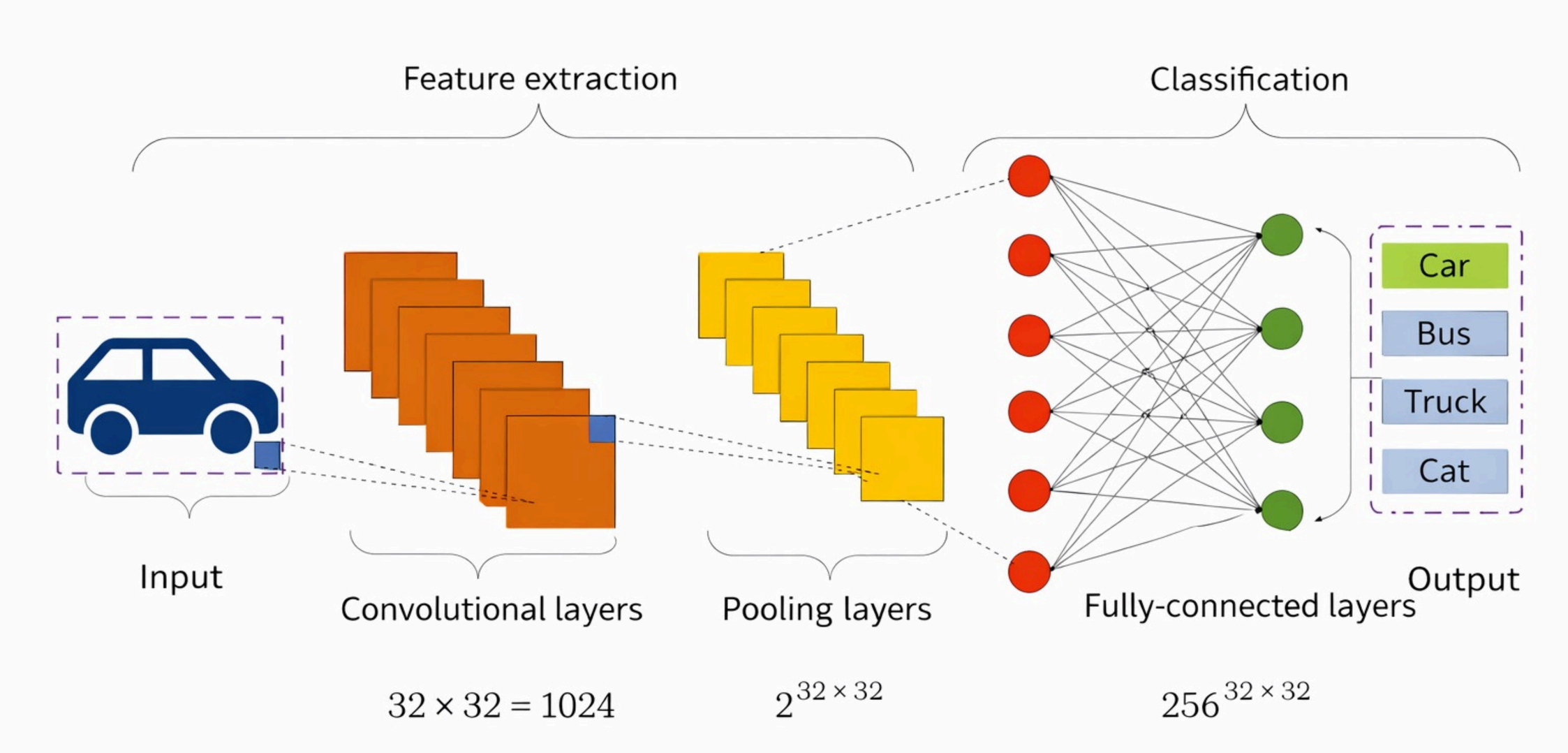
- We need a model that can:
  - **Learn visual patterns:** Automatically detect local features such as edges, corners, and shapes across the image.
  - **Operate efficiently:** Reduce computational cost by processing groups of pixels together instead of treating each pixel independently.
  - **Be robust to variations:** Correctly recognize objects despite changes in position, orientation, scale, or appearance.

Convolutional Neural Networks

- We need a model that can:
  - **Learn visual patterns:** Automatically detect local features such as edges, corners, and shapes across the image.
  - **Operate efficiently:** Reduce computational cost by processing groups of pixels together instead of treating each pixel independently.
  - **Be robust to variations:** Correctly recognize objects despite changes in position, orientation, scale, or appearance.

We need Convolutional Neural Networks (CNNs)!

Convolutional Neural Networks

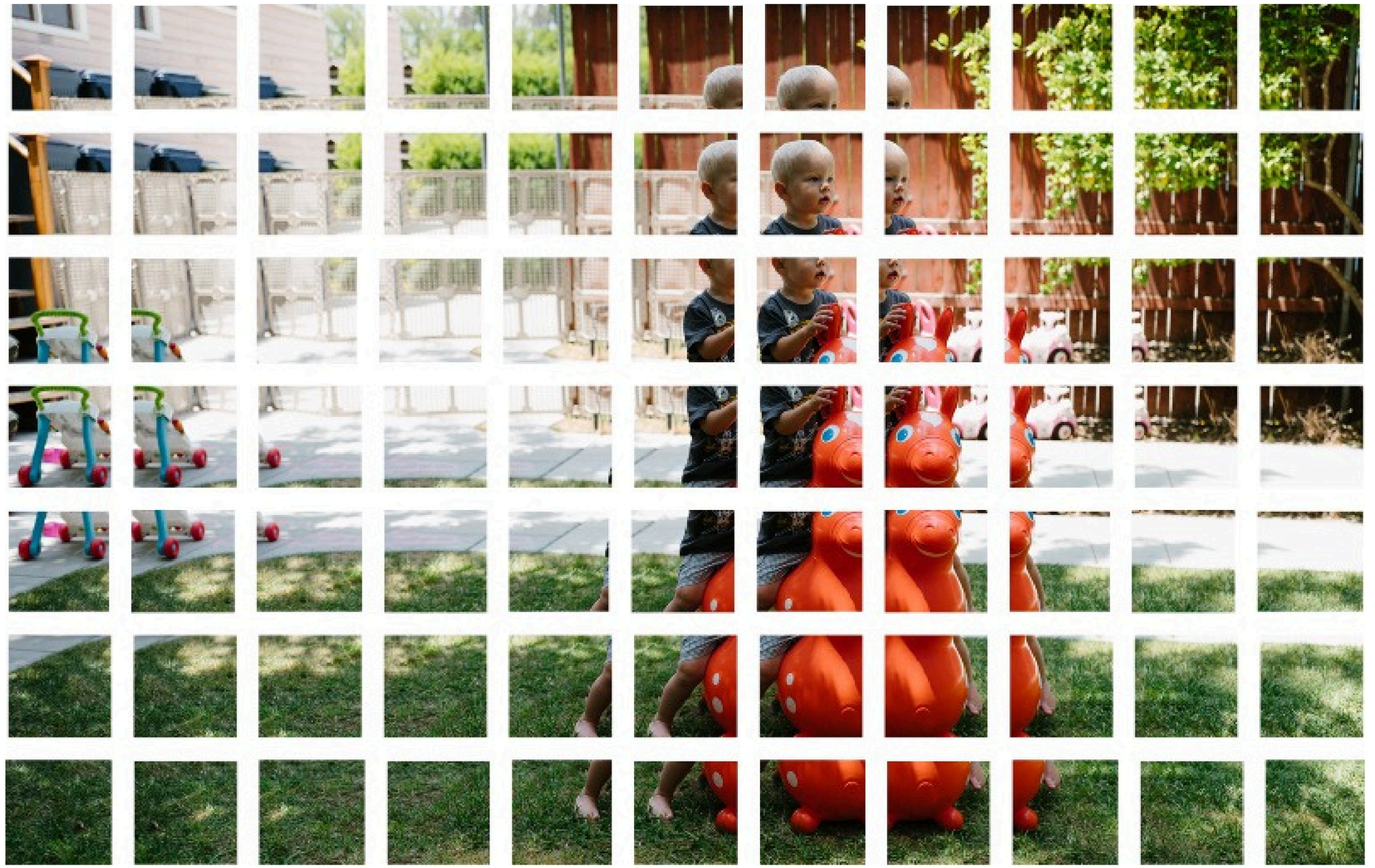


Convolutional Neural Networks

Convolution Layer

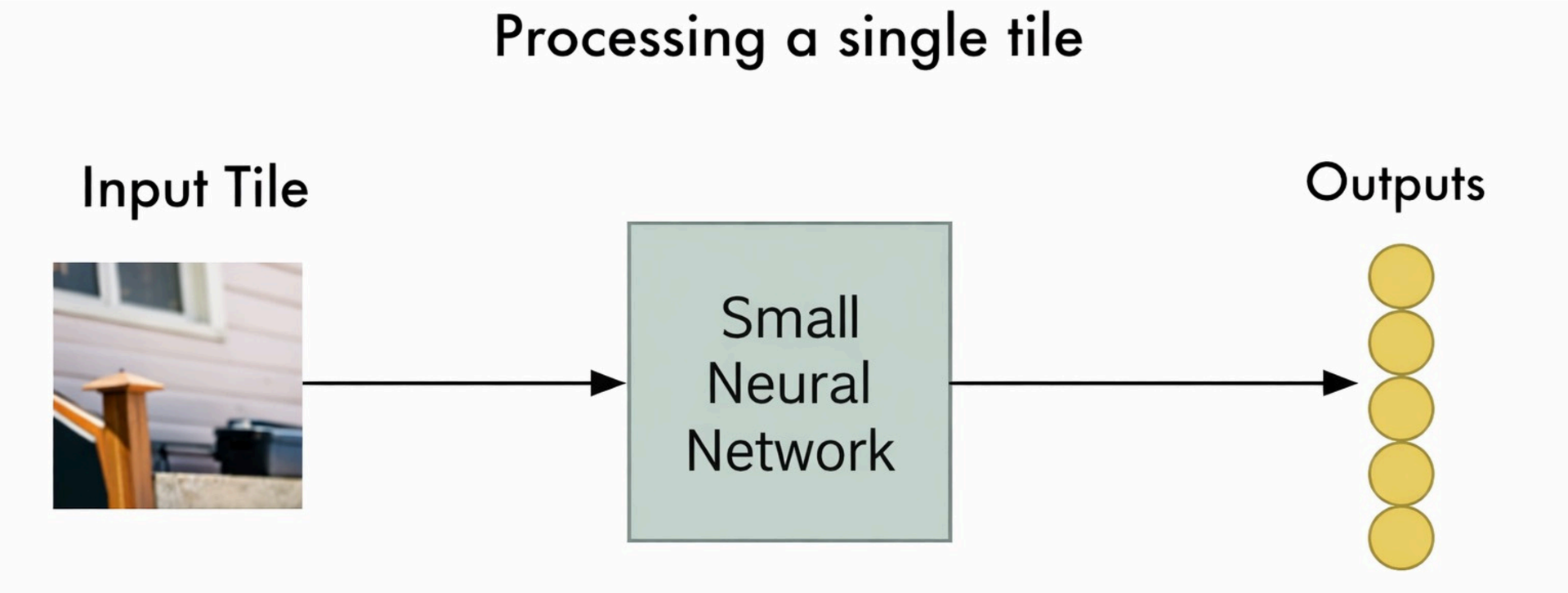
Convolutional Neural Networks

Convolution Layer



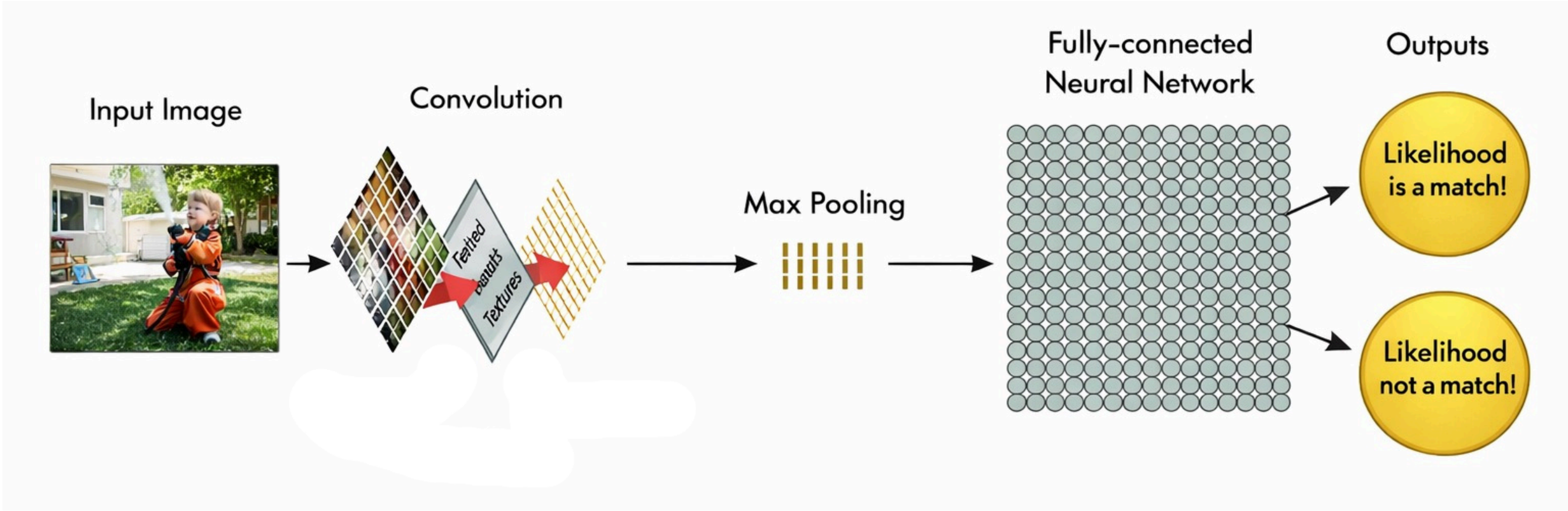
Convolutional Neural Networks

- We process each tile using the weights matrix of a small neural network.



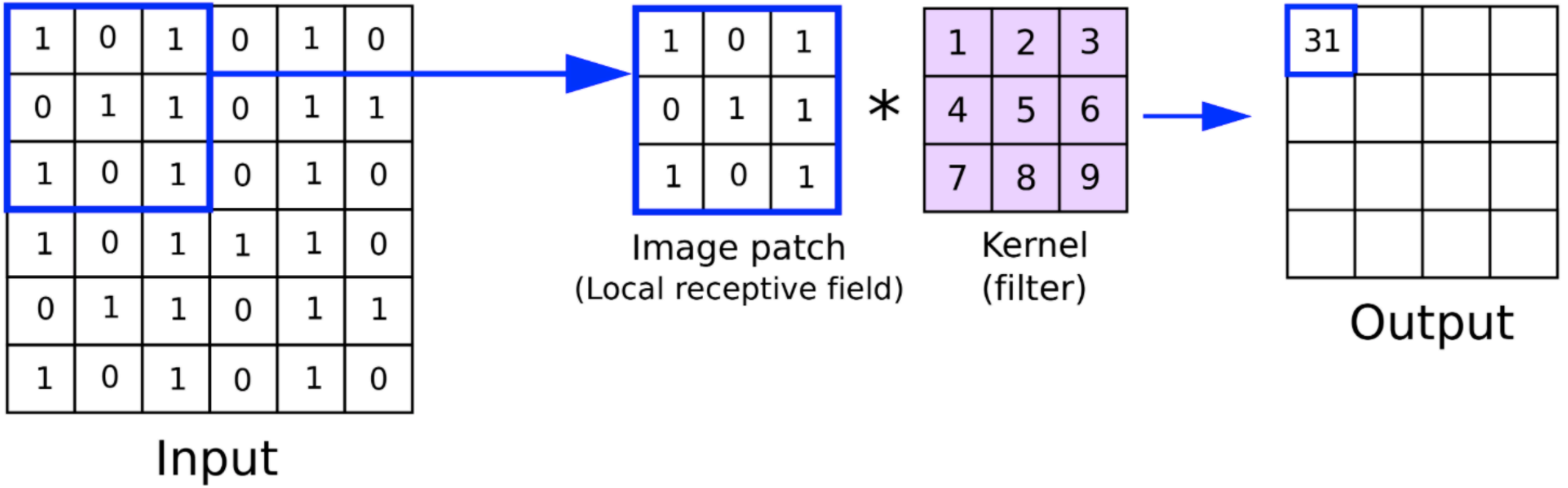
Convolutional Neural Networks

- We process each tile using the weights matrix of a small neural network.



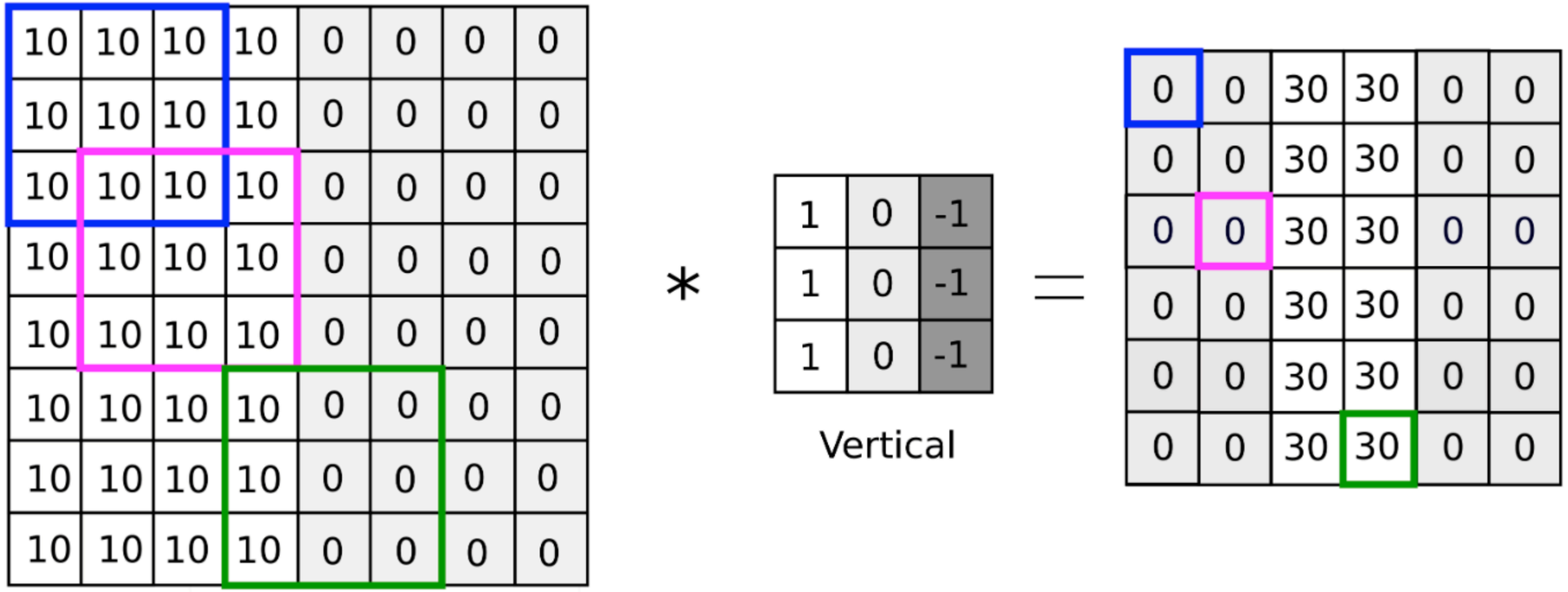
Convolutional Neural Networks

- Convolution involves applying a kernel (filter) over an image to extract features.



Convolutional Neural Networks

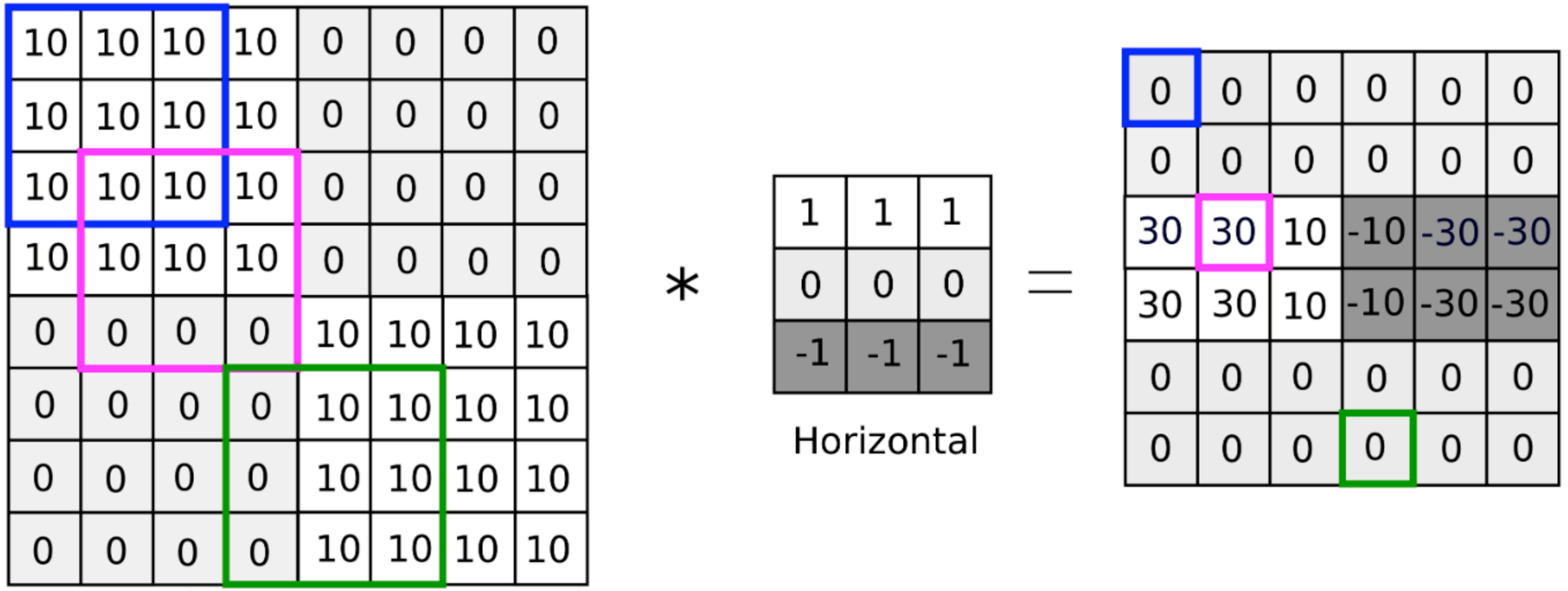
- Convolution involves applying a kernel (filter) over an image to extract features.



An example of vertical edge detection

Convolutional Neural Networks

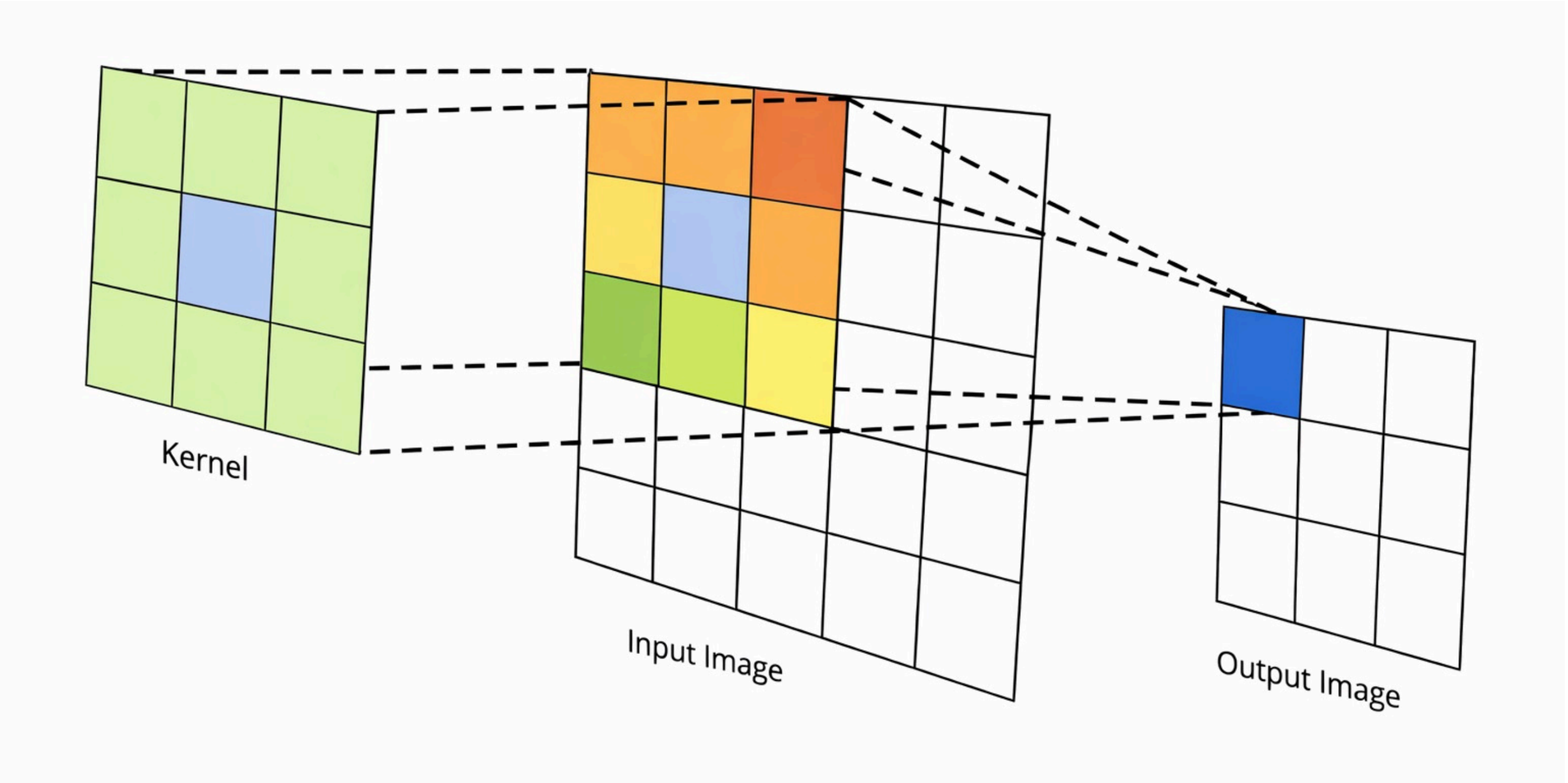
- Convolution involves applying a kernel (filter) over an image to extract features.



An example of horizontal edge detection

Convolutional Neural Networks

- The kernel is slid across the image, computing a dot product with local regions at each position.



Convolutional Neural Networks

- This can be represented mathematically by:

$$z_{i,j} = (W * X)_{i,j}$$

$$z_{i,j} = \sum_{a=0}^{m-1} \sum_{b=0}^{n-1} W_{a,b}\, X_{i+a,\ j+b}$$

$z_{i,j}$: output of the convolution at position (i, j)

W : filter (kernel) matrix of size $m \times n$

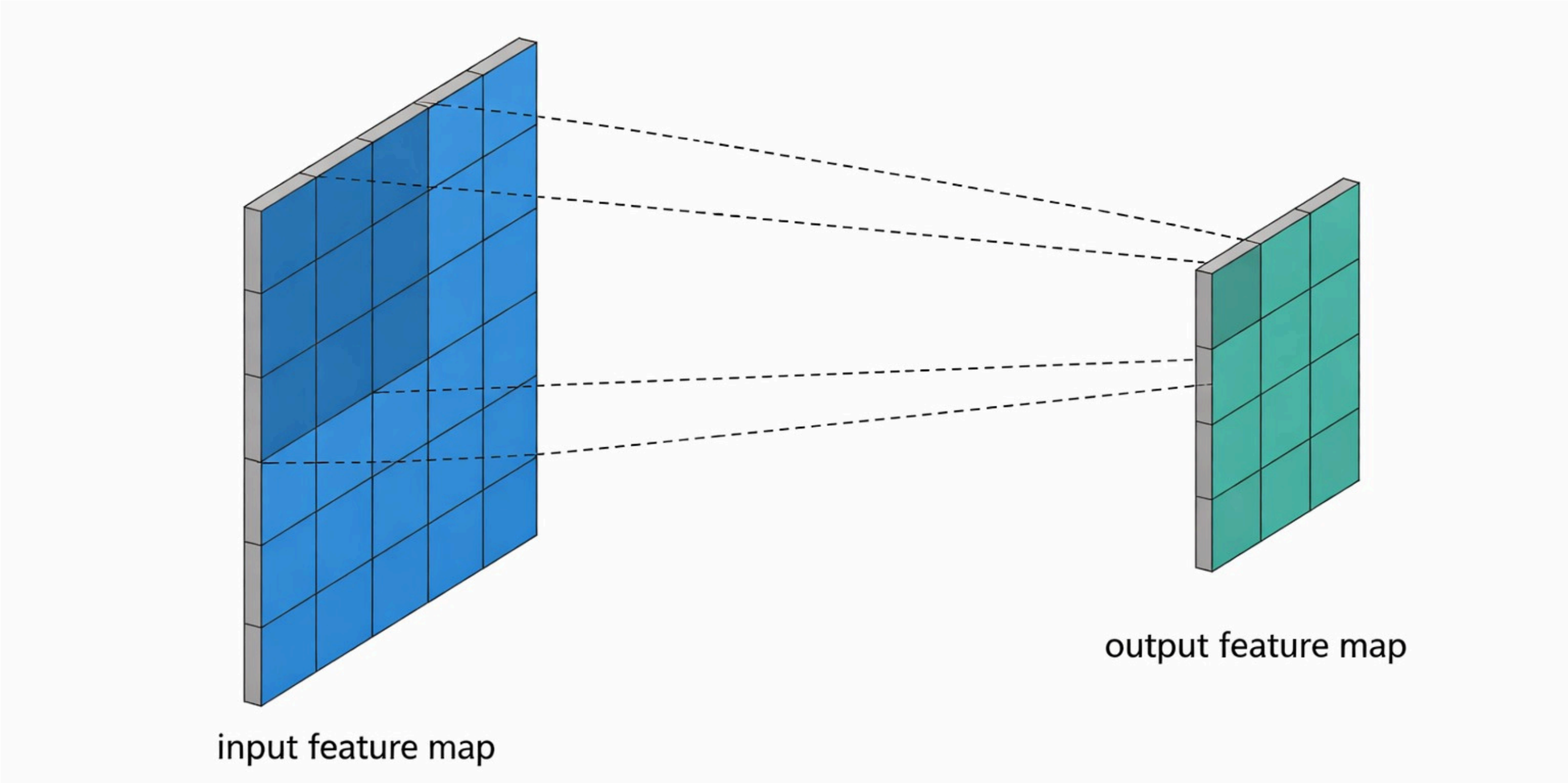
$X_{i,j}$: input value at position (i, j)

$W_{a,b}$: weight of the filter at position (a, b)

$X_{i+a,\ j+b}$: input value in the local region at $(i + a,\ j + b)$

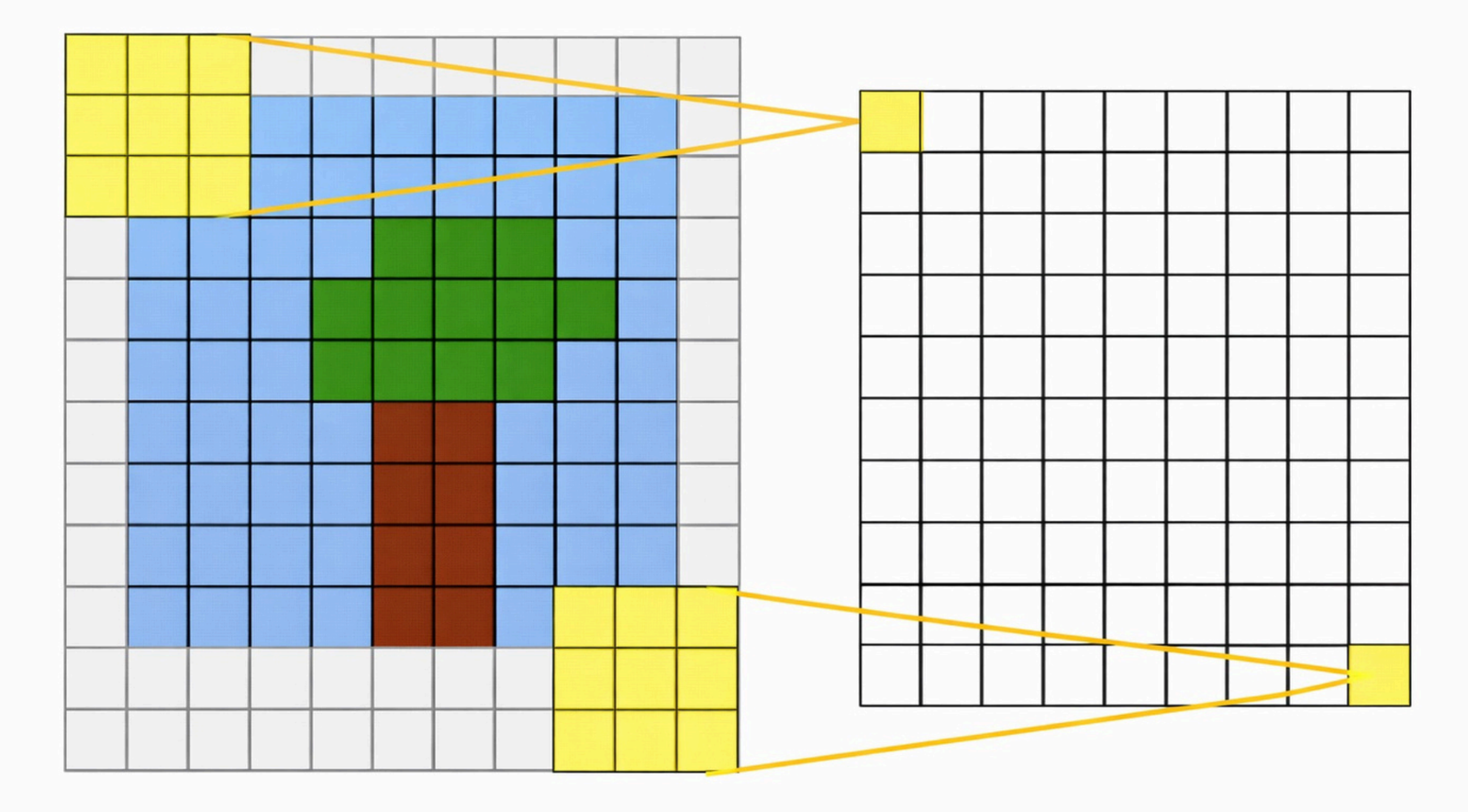
Convolutional Neural Networks

- Applying convolution without padding reduces the feature map size because the kernel cannot fully overlap the input at the borders.



Convolutional Neural Networks

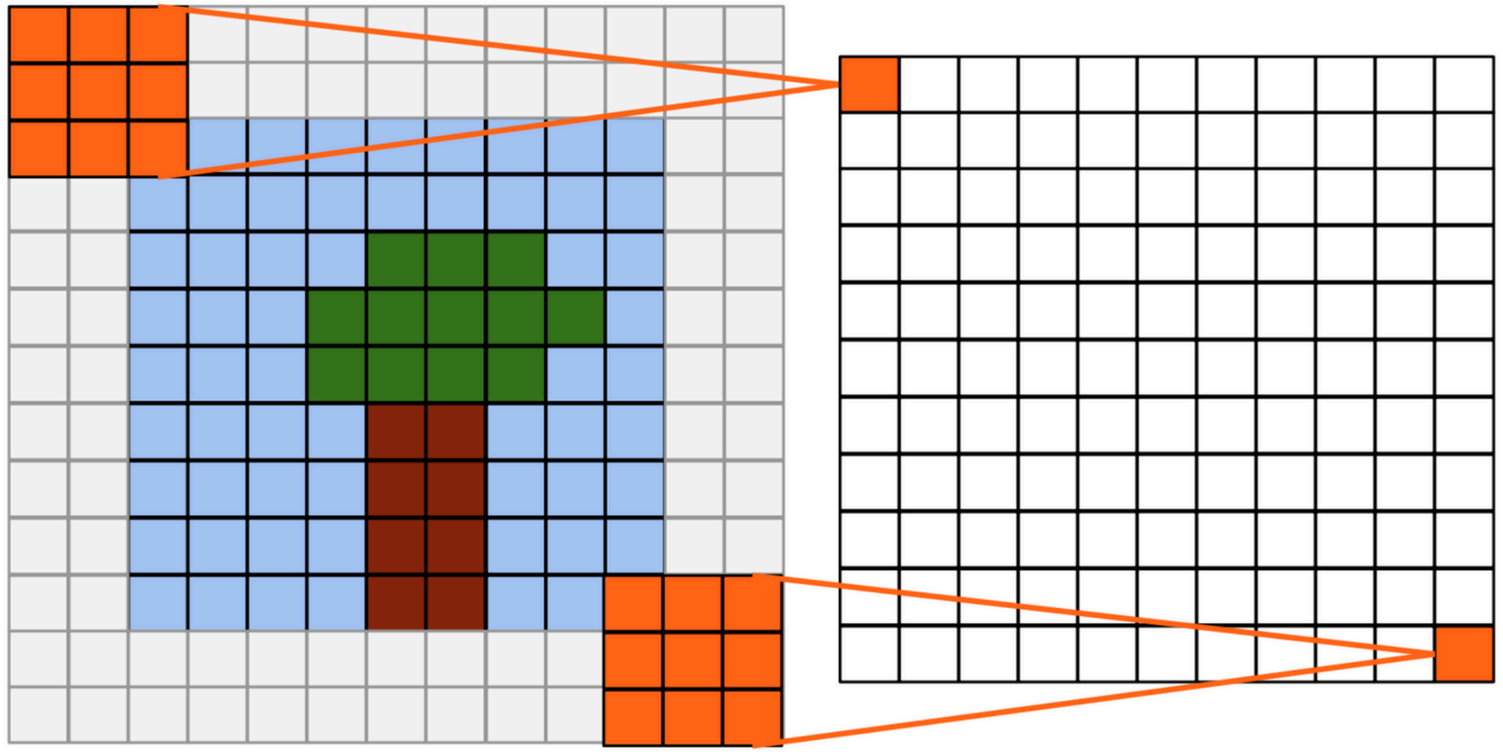
- **Padding** is employed by extending the input image with zeros along its boundaries to control the spatial dimensions of the convolution output.
  - *Same Convolution:* Padding is added so the output size equals the input size.



Convolutional Neural Networks

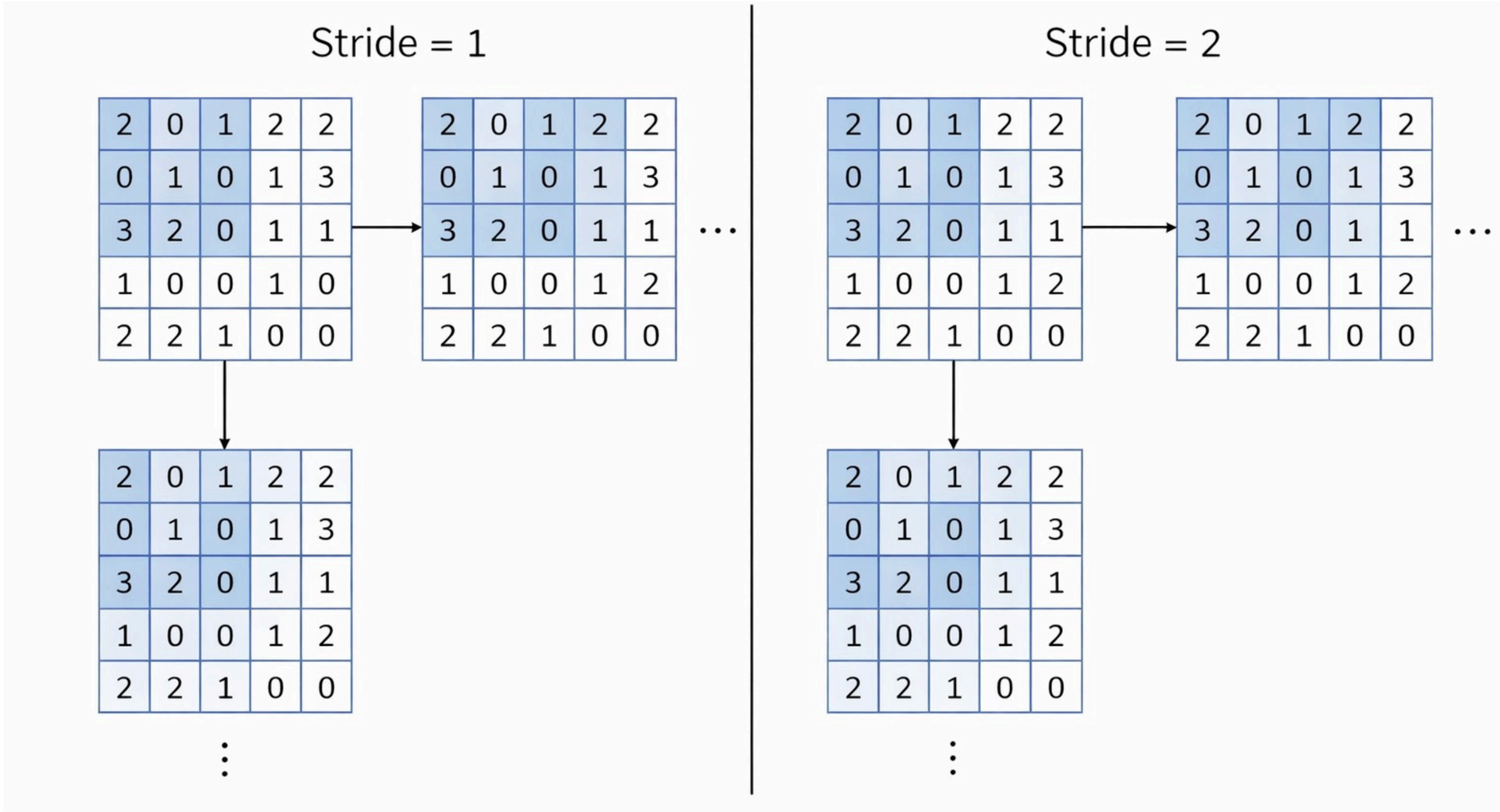
- *Full convolution* adds padding so the kernel can slide over the entire input, including the edges.

$$\text{output size} = \text{input size} + \text{kernel size} - 1$$



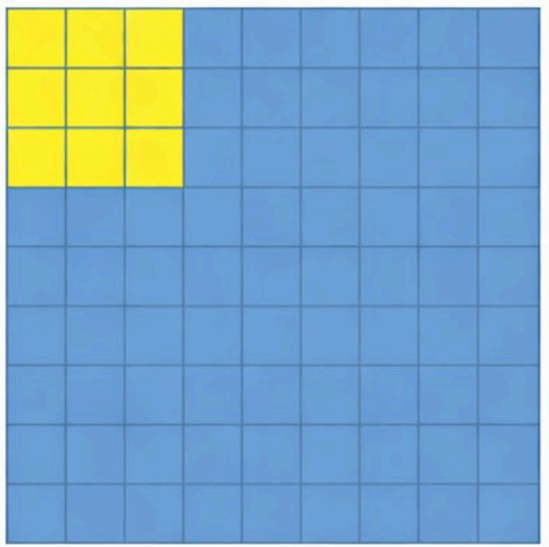
Convolutional Neural Networks

- **Strided Convolution:** Kernel slides along the image with a step >1
  - Larger strides reduce output size.

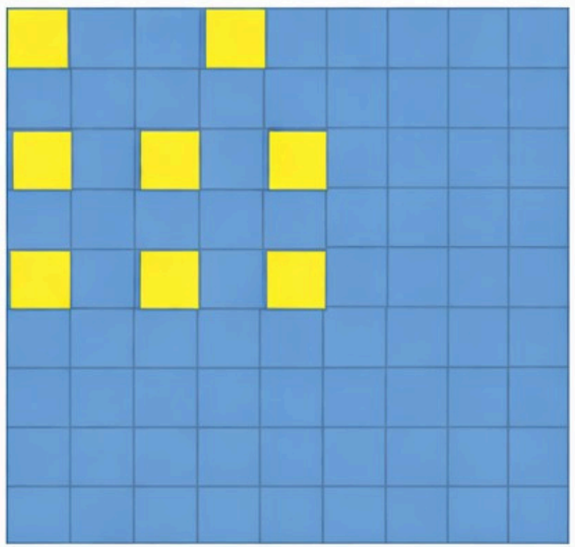


Convolutional Neural Networks

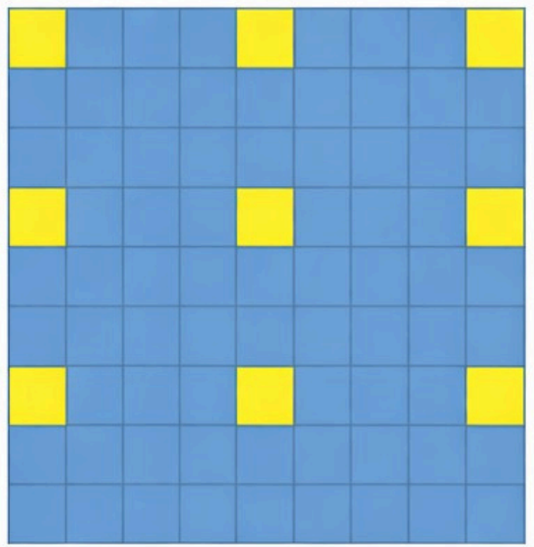
- Dilated Convolution: Kernel is spread out, step > 1 between kernel elements.



Kernel size: 3 × 3
Dilation rate: 1



Kernel size: 3 × 3
Dilation rate: 2



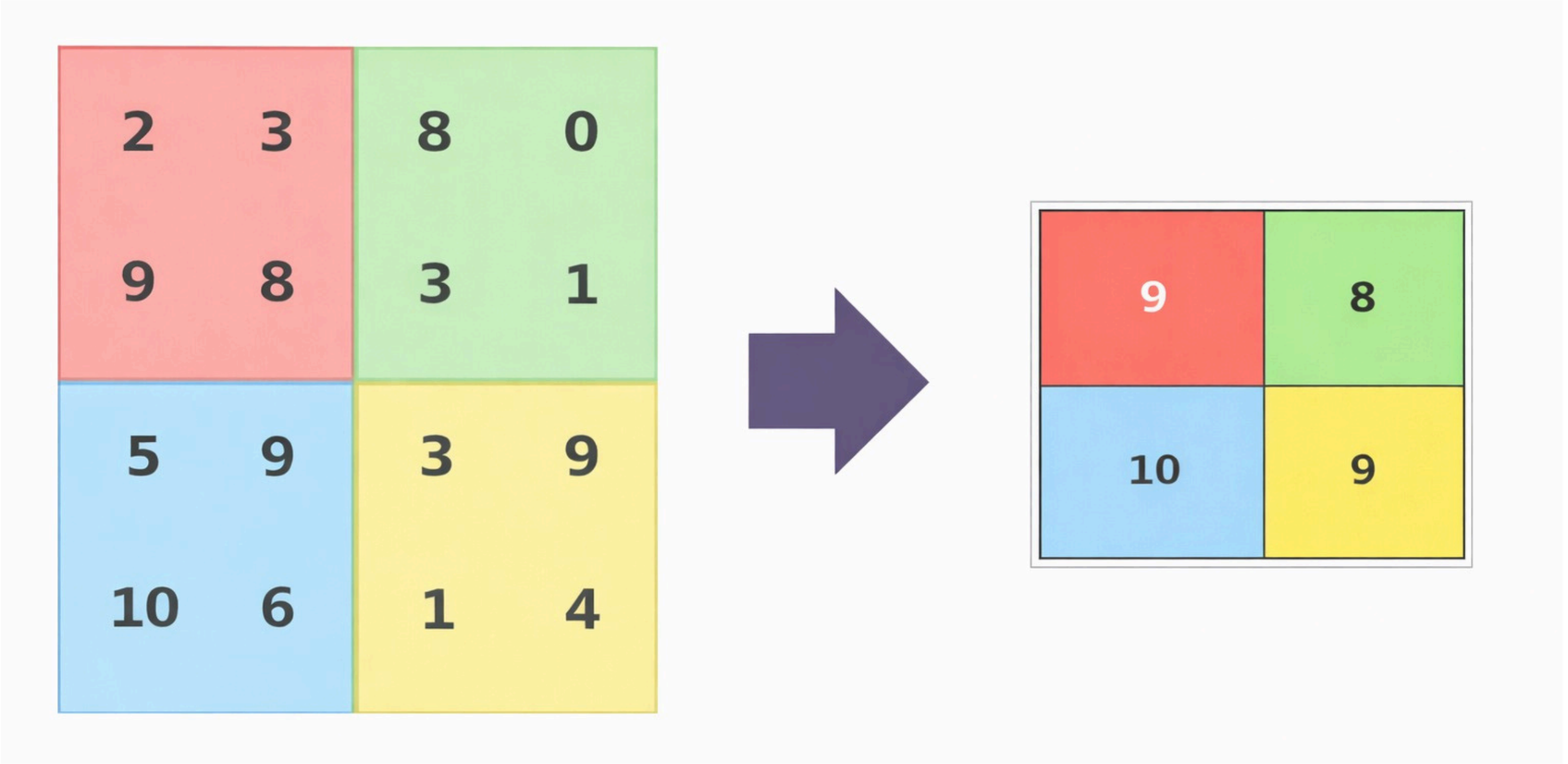
Kernel size: 3 × 3
Dilation rate: 3

Pooling Layer

Pooling Layer

- Pooling is used for dimensionality reduction in CNNs.
  It reduces the spatial size of feature maps while keeping important information.

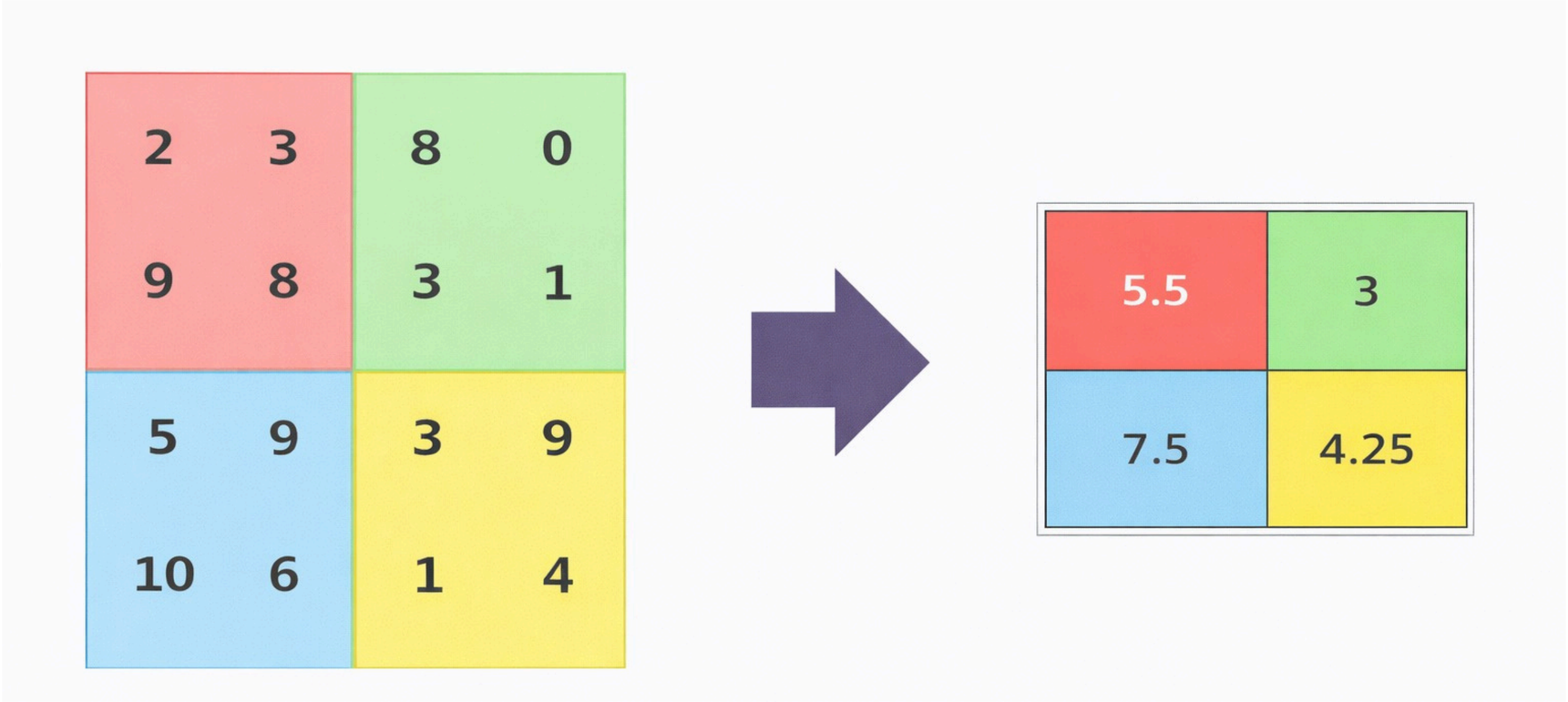
- Max Pooling



Pooling Layer

- Pooling is used for dimensionality reduction in CNNs.
  It reduces the spatial size of feature maps while keeping important information.

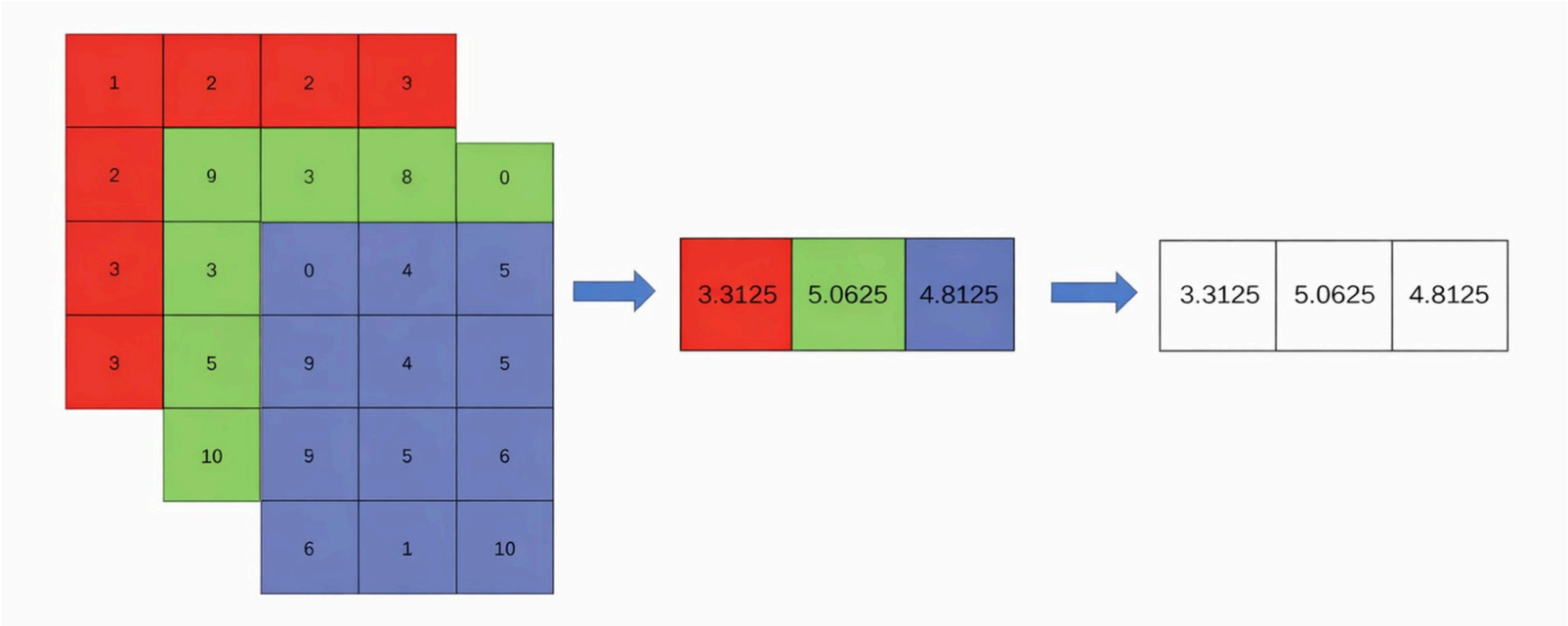
- Average Pooling



Pooling Layer

- Pooling is used for dimensionality reduction in CNNs.
  It reduces the spatial size of feature maps while keeping important information.

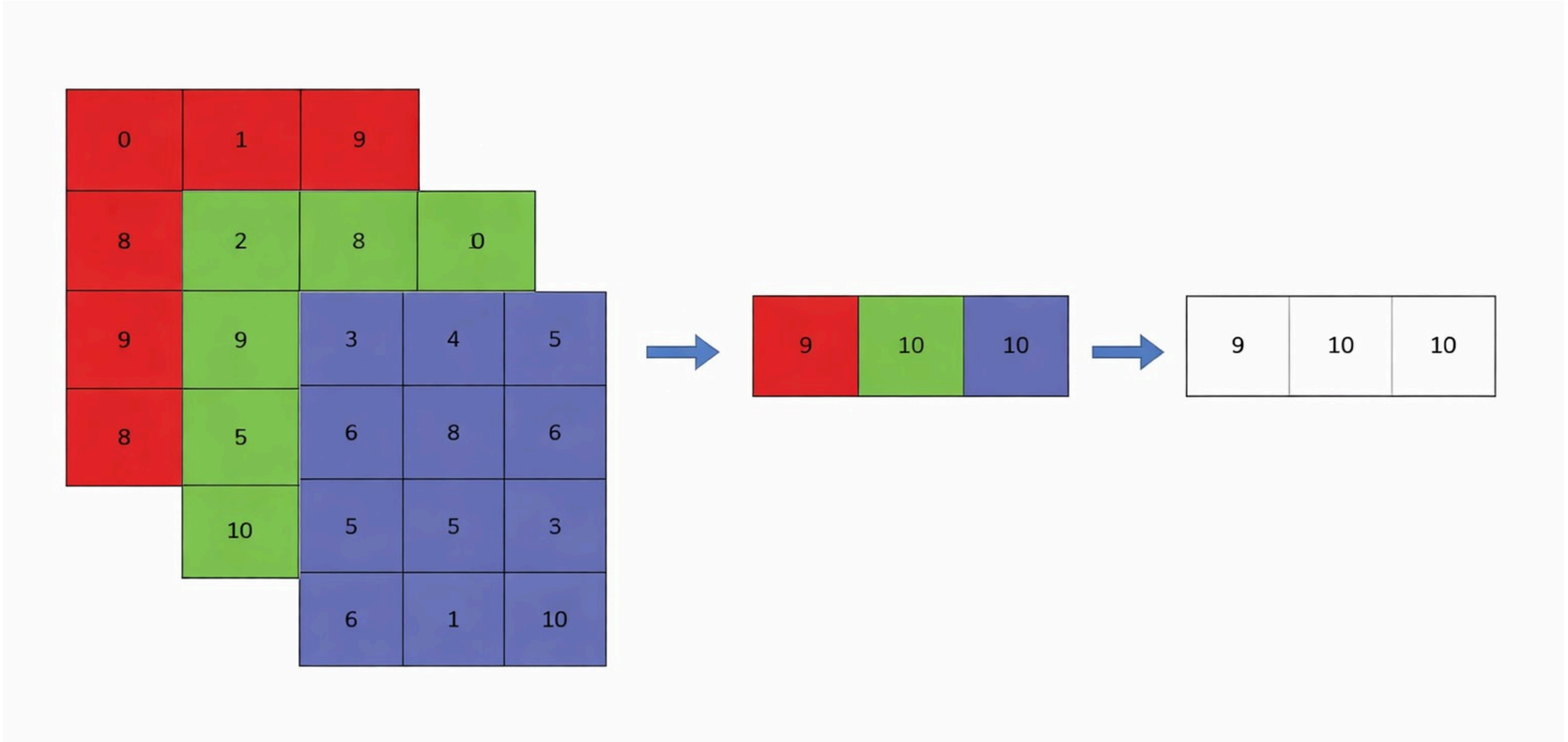
- Global Average Pooling



Pooling Layer

- Pooling is used for dimensionality reduction in CNNs.
  It reduces the spatial size of feature maps while keeping important information.

- Global Max Pooling



CNN Output Size

$$n_{\text{out}} = \left\lfloor \frac{n_{\text{in}} + 2p - k}{s} \right\rfloor + 1$$

n_{in} : number of input features

n_{out} : number of output features

k : convolution kernel size

p : convolution padding size

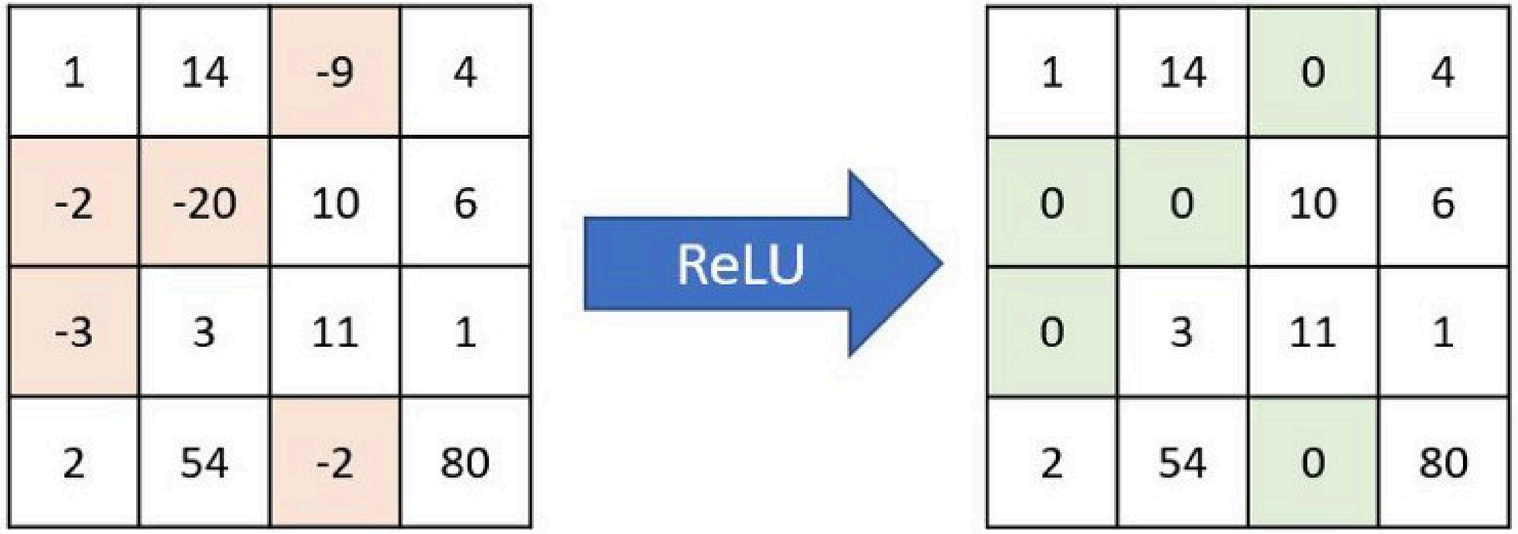
s : convolution stride

Activation

Just like Fully-Connected Neural Networks, we can apply anactivation over the convolutional layer outputs

For example, the Rectified Linear Unit (ReLU) activation function is defined as

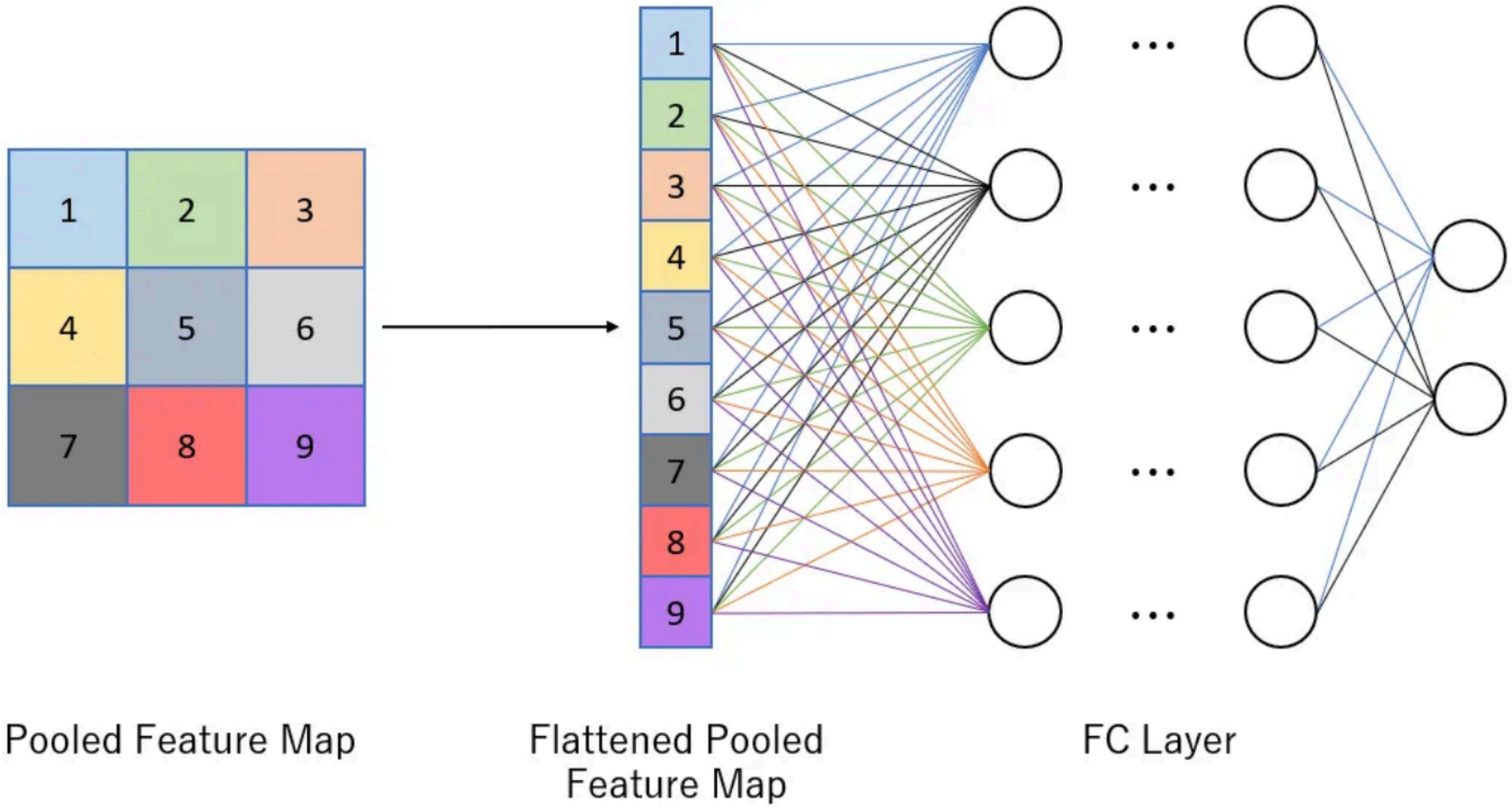
$$\sigma(x) = \max(0, x)$$



Fully Connected layer

A Fully Connected layer

- It combines the features learned by earlier convolutional and pooling layers to perform high-level reasoning or classification.



SOTA CNN Architectures

SOTA CNN Architectures

- The most extensive data for Image Classification
- 3 RGB channels from 0 to 255
- 14,197,122 images
- 1000 classes

SOTA CNN Architectures

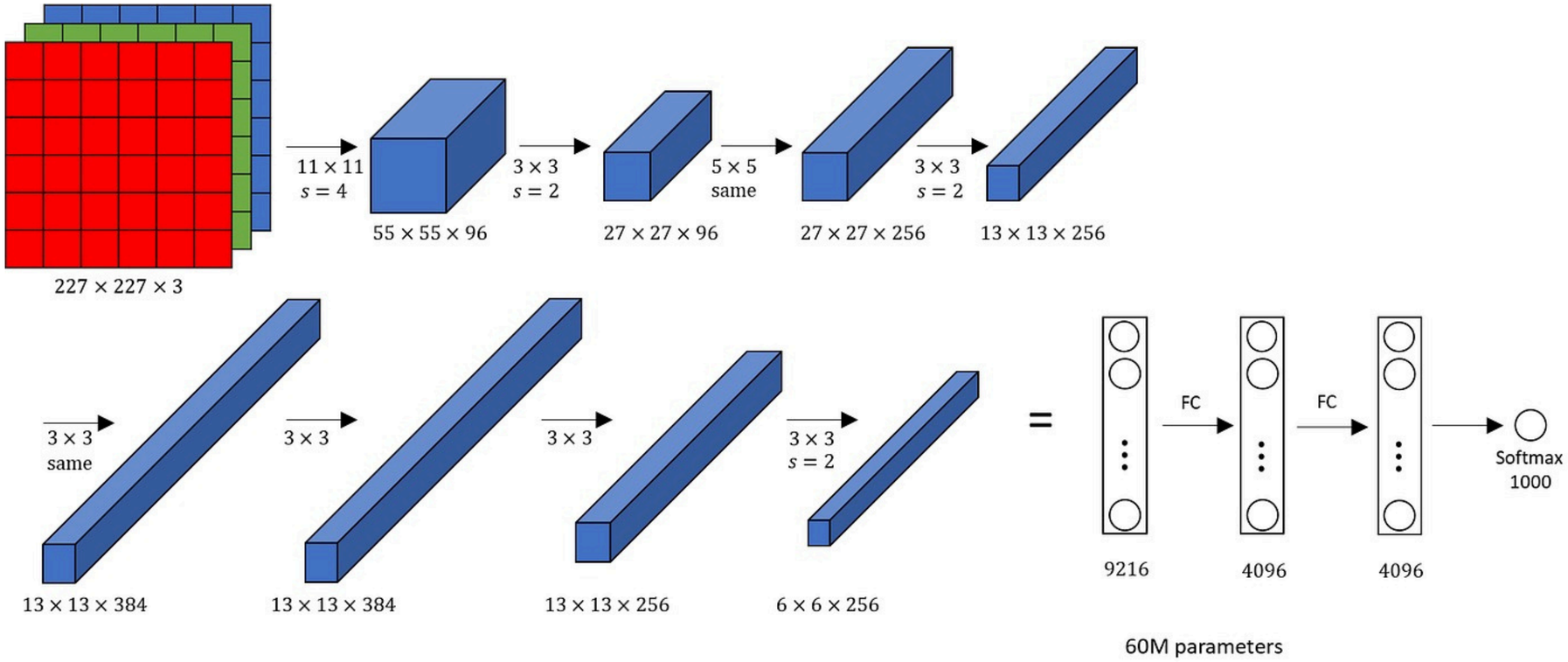
- Over time, researchers built advanced CNN architectures to improve performance and efficiency. These architectures introduced key innovations:

  • **AlexNet** [Krizhevsky et al. 2012]: The first CNN to achieve breakthrough performance on image classification.
  • **VGGNet** [Simonyan and Zisserman, 2014]: Used very deep networks (up to 19 layers).
  • **InceptionNet** (GoogLeNet) [Szegedy et al., 2014]: Used multiple filter sizes per layer (Inception modules).
  • **ResNet** [He et al., 2015]: Introduced skip connections for training very deep networks.
  • **EfficientNet** [Tan and Le, 2019]: Found a scaling method that simultaneously scales a CNN's depth, width, and resolution optimally using a single scaling coefficient.

SOTA CNN Architectures

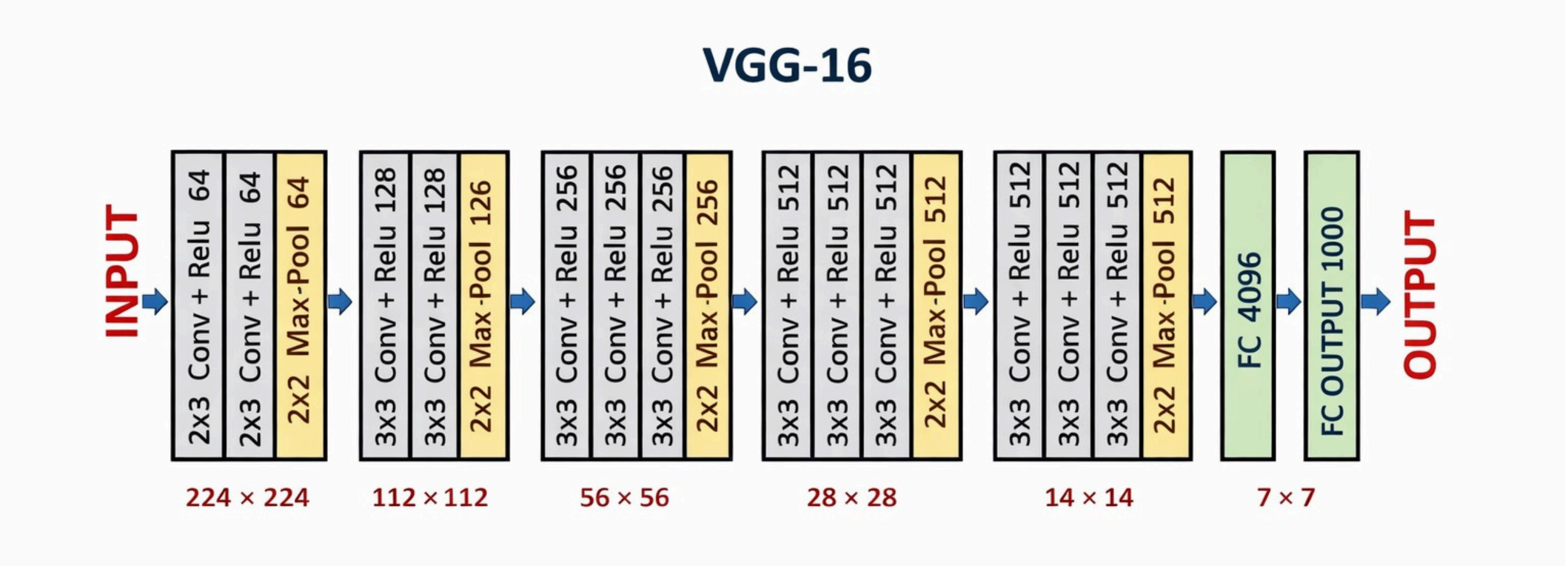
- **AlexNet** is a deep convolutional neural network that revolutionized computer vision by winning the ImageNet 2012competition with a large margin.

- 5 Convolutional layers
- 3 Fully Connected layers
- ReLU after every layer
- Max Pooling
- Dropout



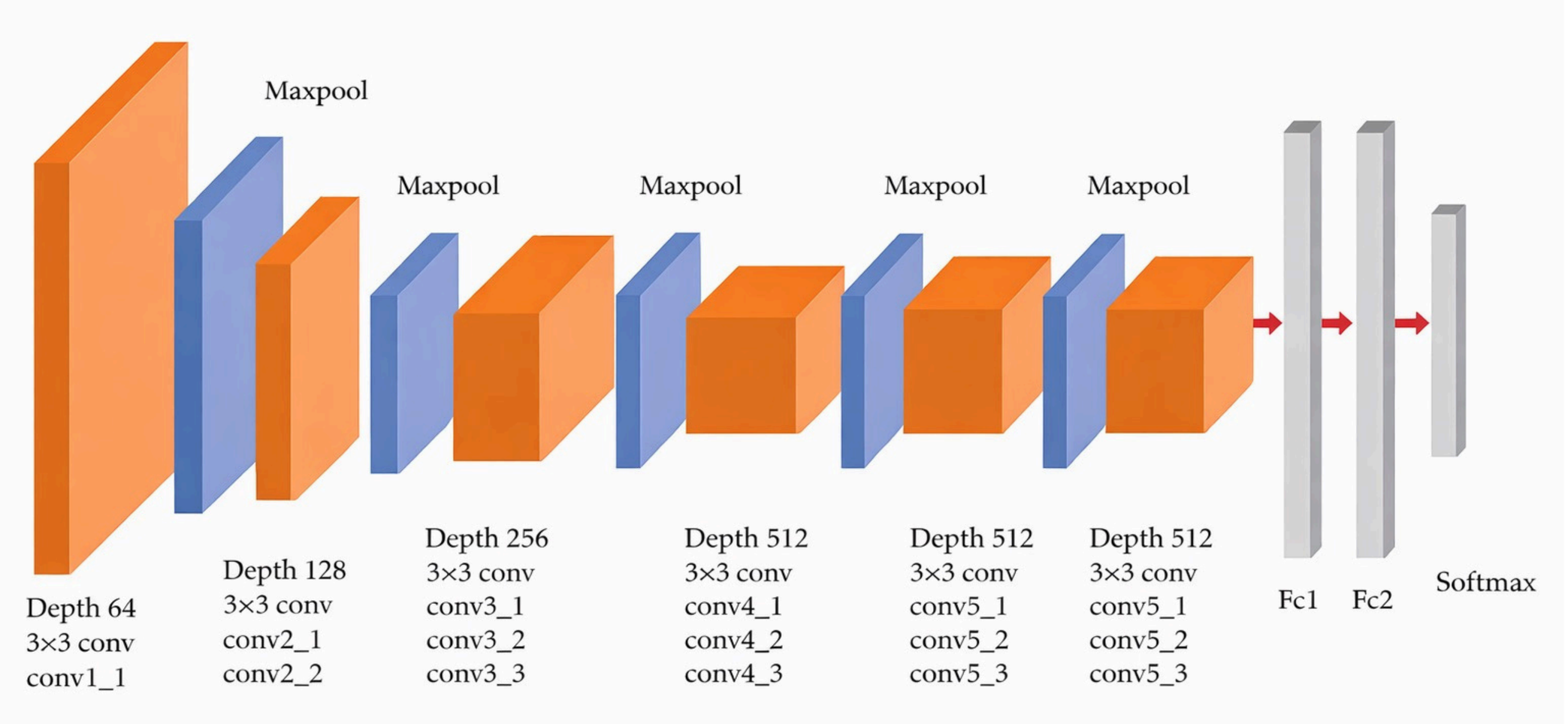
SOTA CNN Architectures

- **VGG16** image highlight 5 convolutional blocks and 16 weight layers



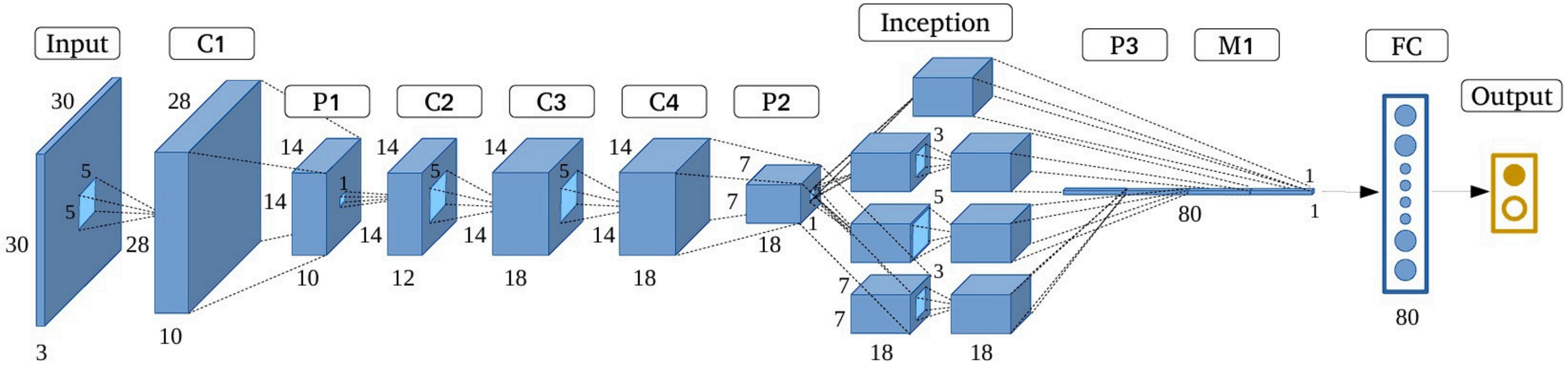
SOTA CNN Architectures

- **VGG19** image highlight deeper blocks with extra 3×3 conv layers



SOTA CNN Architectures

- **InceptionNet** achieves a depth of 22 layers while maintaining efficient computation.

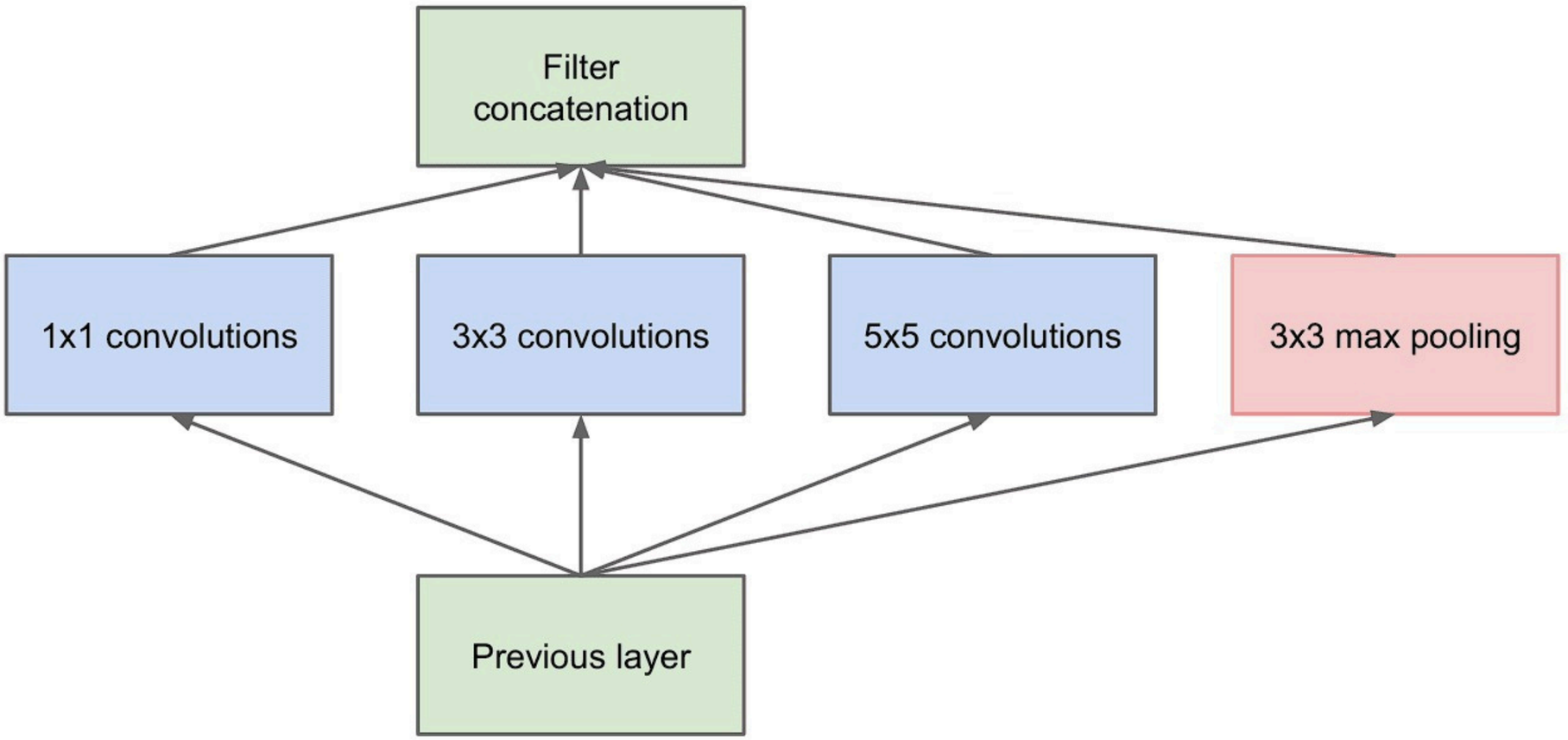


SOTA CNN Architectures

type	patch size/stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1x1x1024	0								
dropout (40%)		1x1x1024	0								
linear		1x1x1000	1							1000K	1M
softmax		1x1x1000	0								

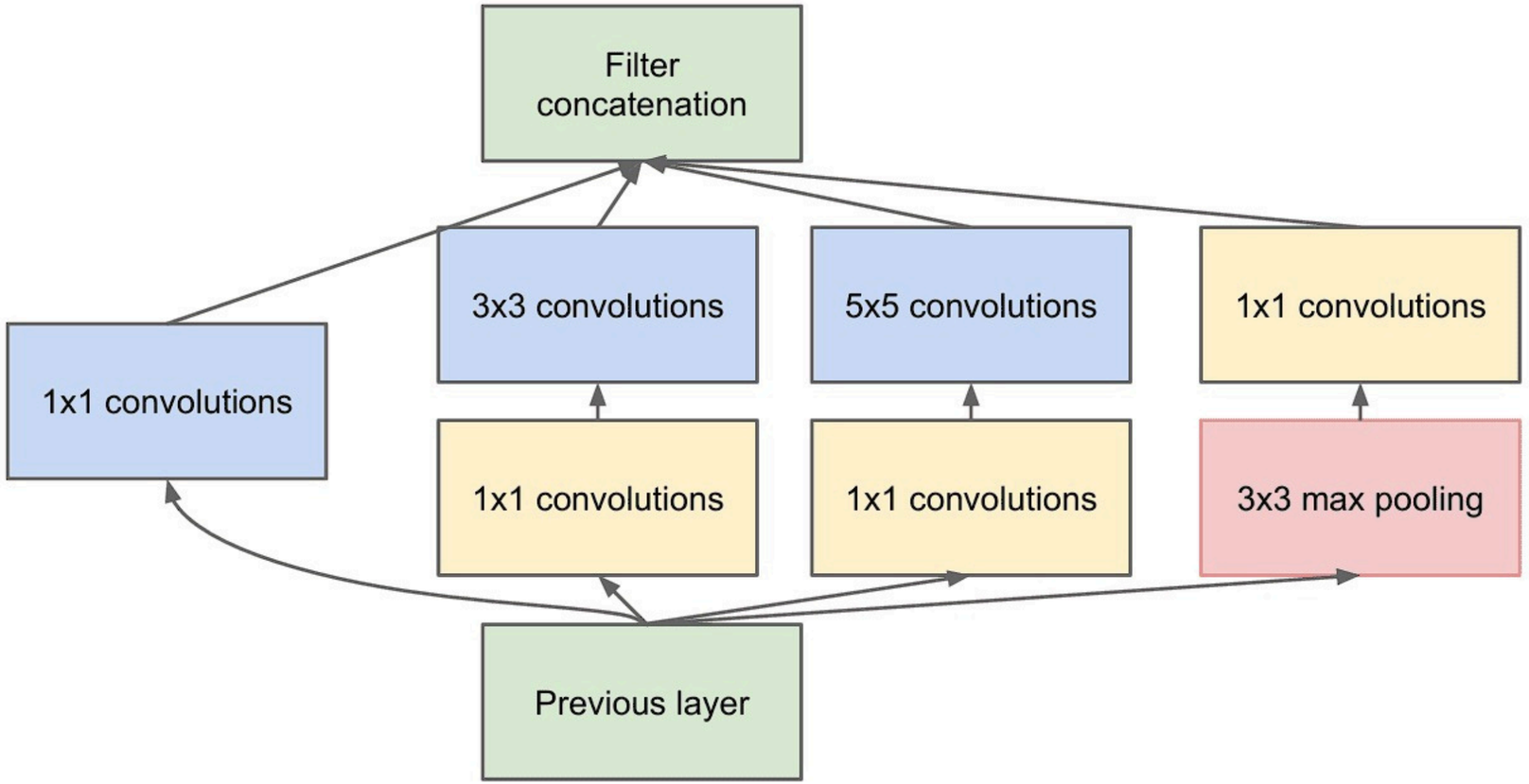
SOTA CNN Architectures

- **Inception module:** Uses multiple filter sizes (1×1, 3×3, 5×5), in parallel, to capture different features, then combines their outputs.



SOTA CNN Architectures

- **Inception module:** Uses multiple filter sizes (1×1, 3×3, 5×5), in parallel, to capture different features, then combines their outputs.



SOTA CNN Architectures

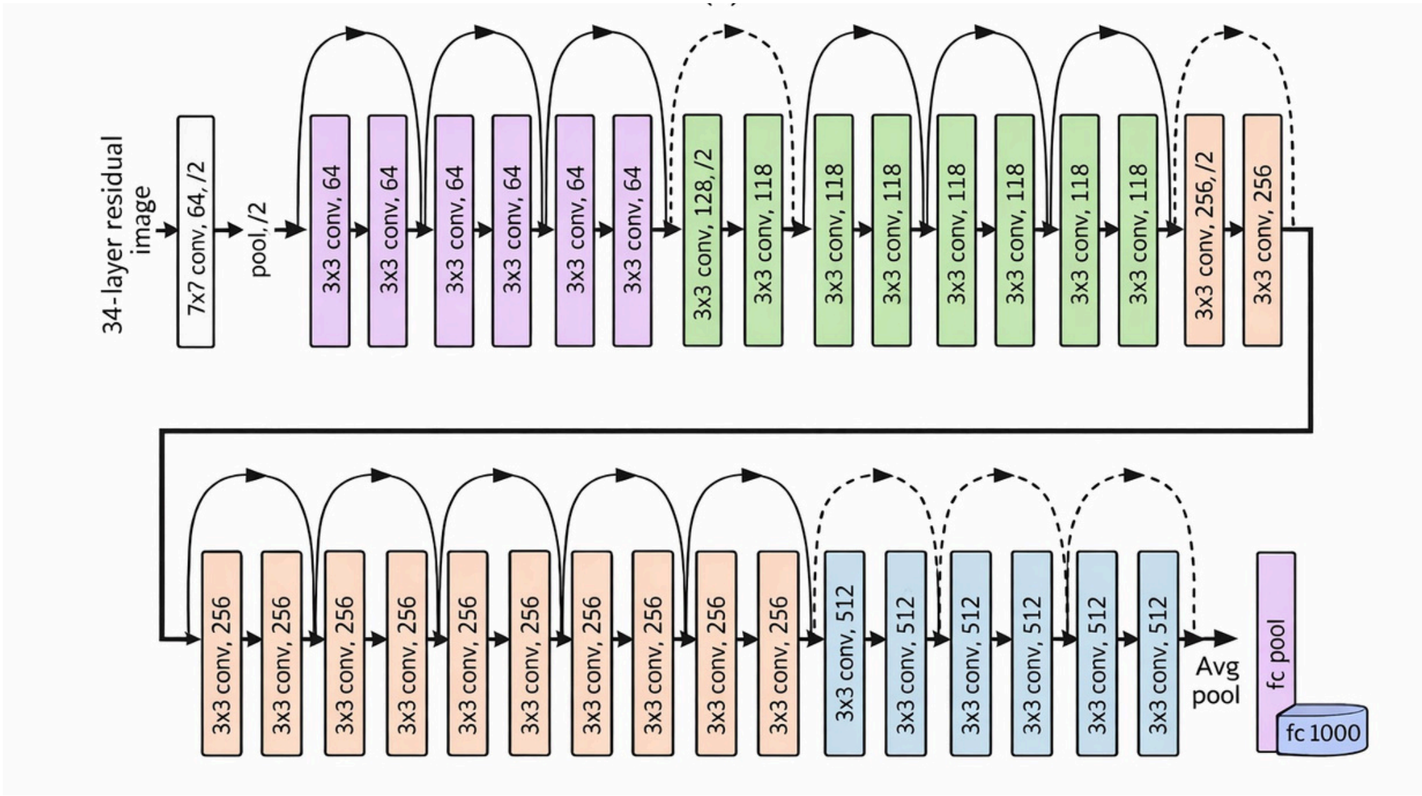
Deeper Is Not Always Better

- Simply increasing network depth does not guarantee higher accuracy and can even degrade performance.
- In very deep networks, gradients can become extremely small during backpropagation. This slows learning or completely prevents effective weight updates in early layers.

SOTA CNN Architectures

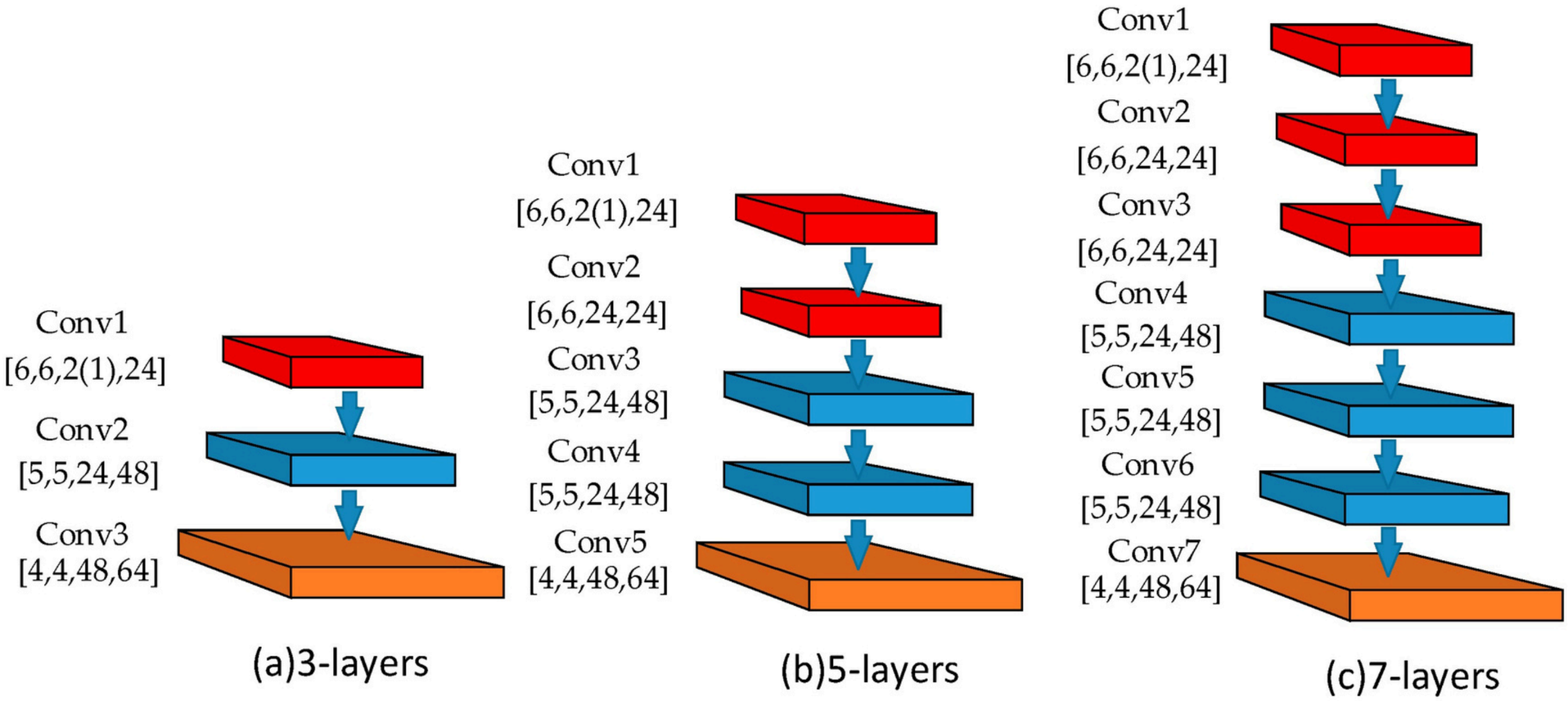
ResNet

- Very deep networks using residual connections
- 152-layer model for ImageNet
- Residual: A shortcut connection that helps the network pass
- information through layers more easily.



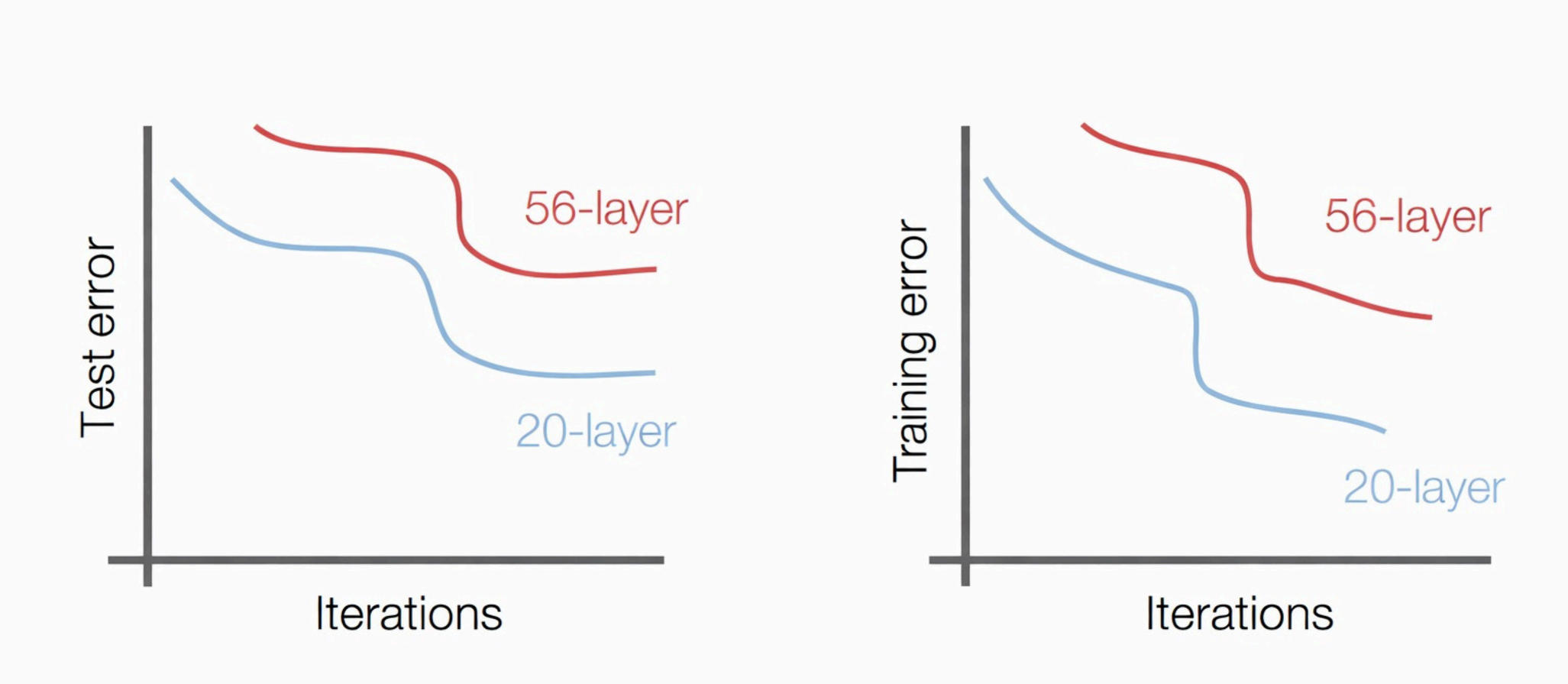
SOTA CNN Architectures

- What happens when we continue stacking deeper layers on a ”plain” Convolutional neural network?



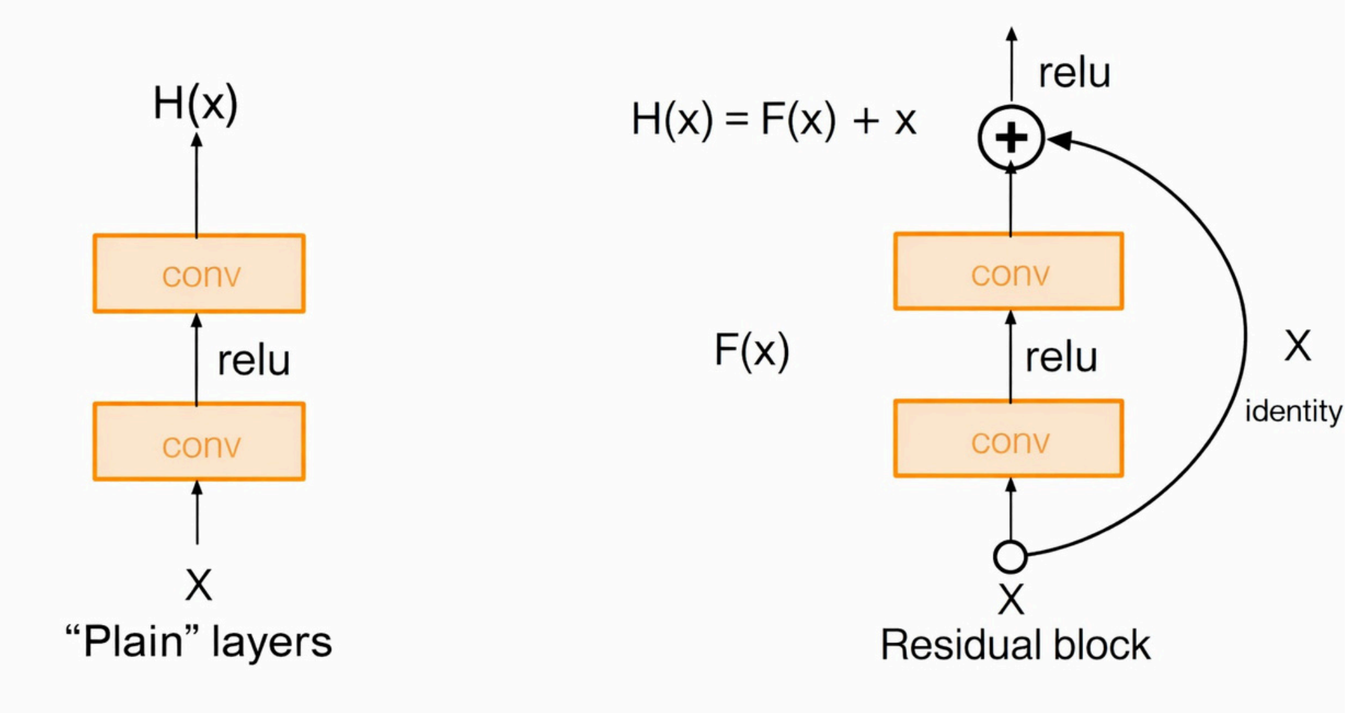
SOTA CNN Architectures

What happens when we continue stacking deeper layers on a ”plain” Convolutional neural network?



SOTA CNN Architectures

- Deep models have more representation power (more parameters) than shallower models.
- The problem is an optimization problem, deeper models are harder to optimize
- Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping



SOTA CNN Architectures

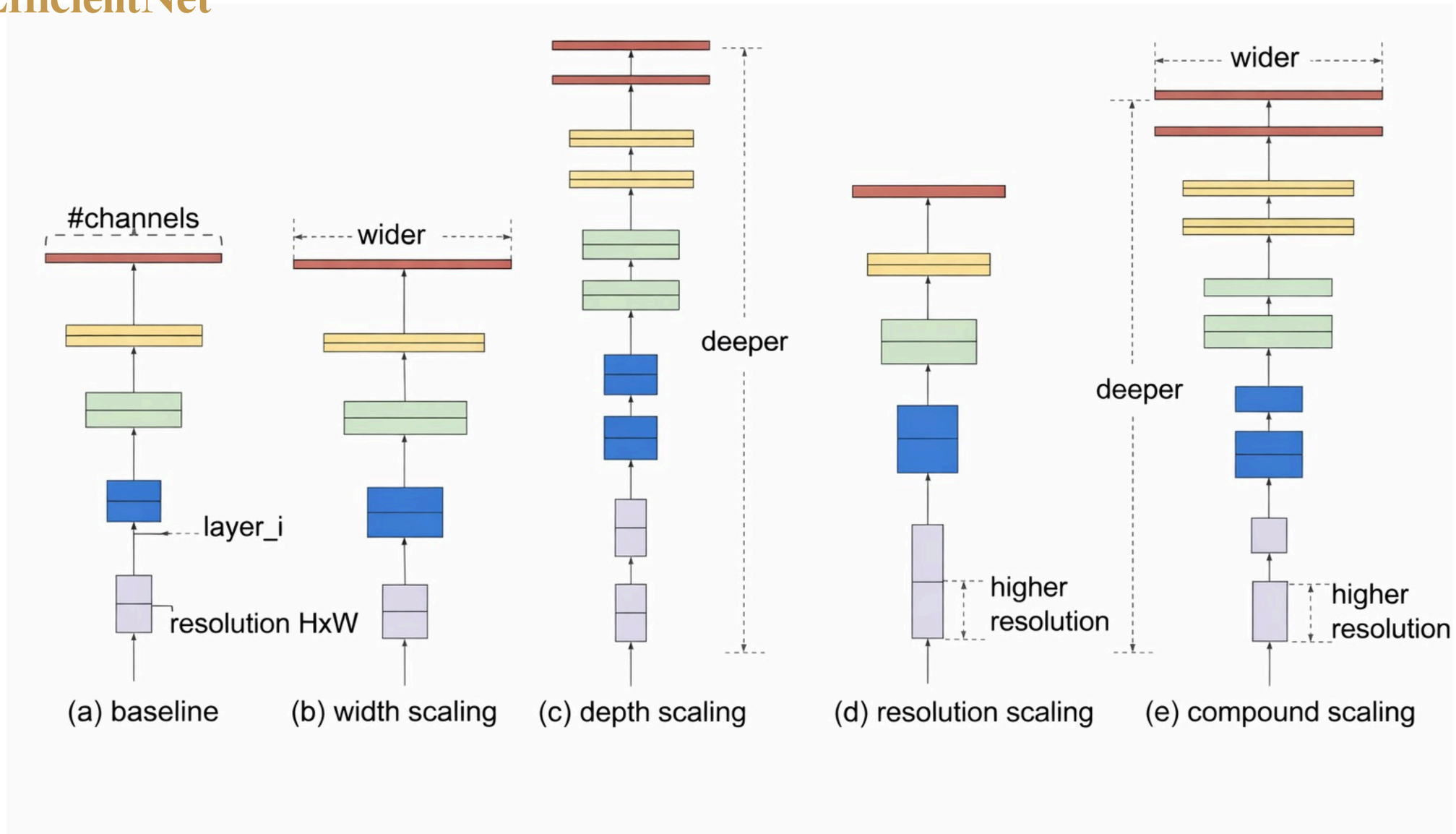
- **EfficientNet** introduced a compound scaling method to systematically find the optimal balance among depth, width, and resolution.

  - Increase the number of layers (depth)
  - Increase the number of neurons in each layer (width)
  - Use higher-resolution images (resolution)

  But finding the right balance of these three was largely based on trial and error.

SOTA CNN Architectures

EfficientNet



SOTA CNN Architectures

EfficientNet

Compound Scaling uses a single scaling coefficient (ϕ) to control:

$$\alpha^{\phi}: \text{ Depth}, \quad \beta^{\phi}: \text{ Width}, \quad \gamma^{\phi}: \text{ Resolution}$$

Goal: find α, β, γ that balance accuracy and efficiency, then scale up optimally by increasing ϕ

SOTA CNN Architectures

EfficientNet

Optimizes depth, width, and resolution using this constraint:

$$\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$

Increasing depth (α) increases FLOPs linearly

Increasing width (β) increases FLOPs quadratically (β^2)

Increasing resolution (γ) increases FLOPs quadratically (γ^2)

To double total FLOPs, the three factors must be balanced together.

SOTA CNN Architectures

EfficientNet

- The authors of EfficientNet searched for the best scaling factors on a small baseline model.
- They found:

$$\alpha = 1.2, \quad \beta = 1.1, \quad \gamma = 1.15$$

SOTA CNN Architectures

$$\alpha = 1.2, \quad \beta = 1.1, \quad \gamma = 1.15$$

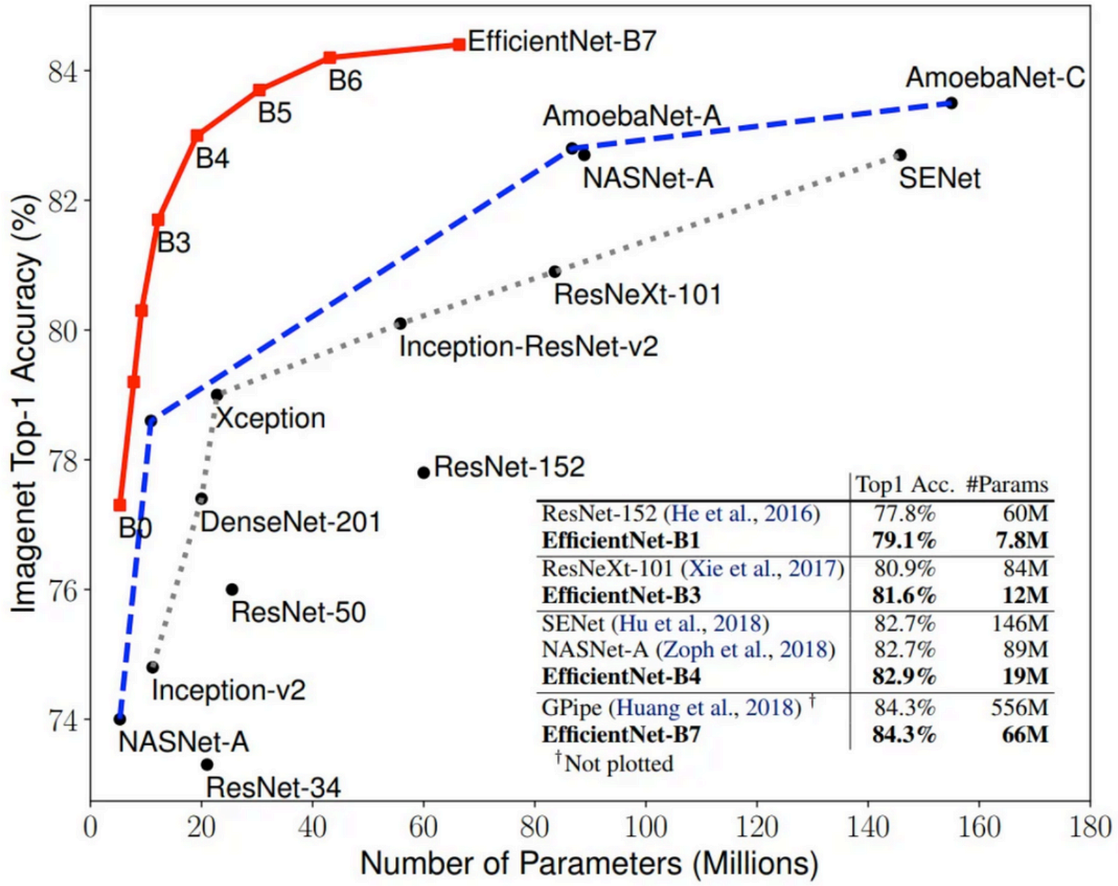
Model	ϕ	**Depth** (α^ϕ)	**Width** (β^ϕ)
B0	0	$1.2^0 = 1.0$	$1.1^0 = 1.0$
B1	1	$1.2^1 = 1.2$	$1.1^1 = 1.1$
B2	2	$1.2^2 = 1.44$	$1.1^2 = 1.21$
B3	3	$1.2^3 = 1.73$	$1.1^3 = 1.33$
B4	4	$1.2^4 = 2.07$	$1.1^4 = 1.46$
B5	5	$1.2^5 = 2.49$	$1.1^5 = 1.61$
B6	6	$1.2^6 = 2.99$	$1.1^6 = 1.77$
B7	7	$1.2^7 = 3.58$	$1.1^7 = 1.94$

Table 1: Scaling EfficientNet from B0 to B7

SOTA CNN Architectures

EfficientNet

- Achieve state-of-the-art accuracy with significantly fewer parameters and FLOPs.



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.3%**	**66M**

†Not plotted

SOTA CNN Architectures

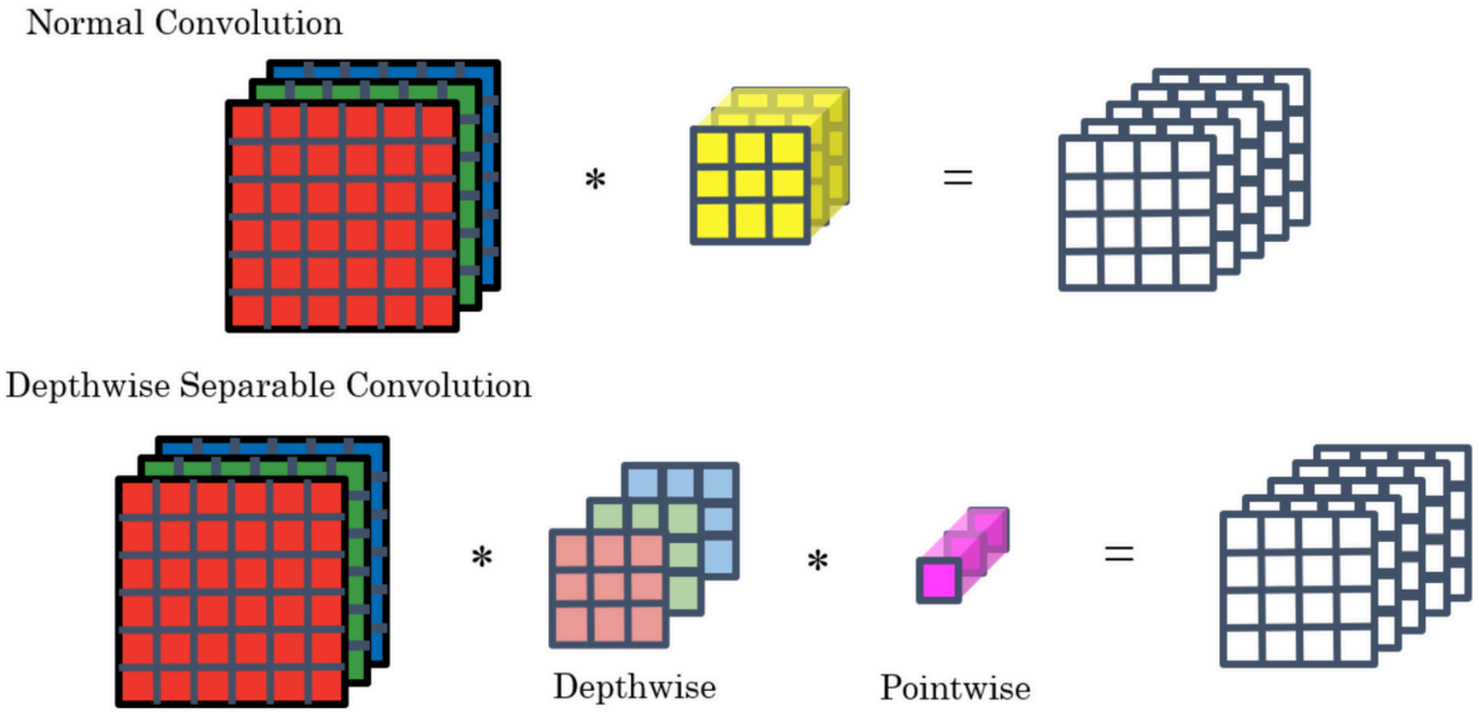
MobileNets

- Small-sized models are crucial for mobile and embedded devices.
- MobileNets reduce computational cost and memory usage while maintaining good accuracy.
- Use depthwise-separable convolutions to significantly reduce computation compared to standard convolutions.

SOTA CNN Architectures

MobileNets

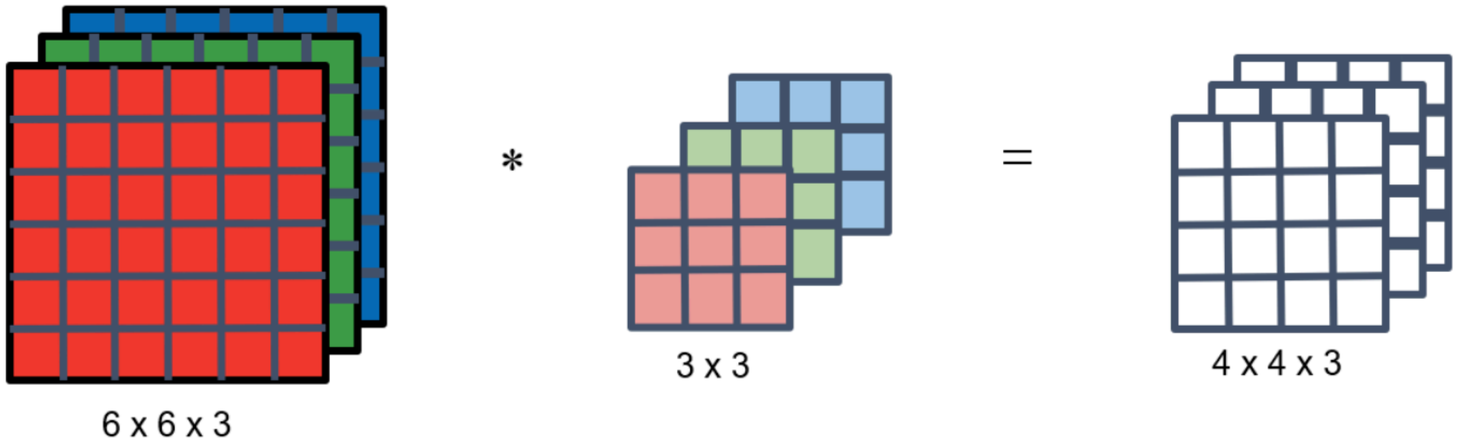
- Computational cost of standard convolution:
- Cost = # filter params × # filter positions × # filters
- Filters operate on all input channels, increasing computation significantly.



SOTA CNN Architectures

MobileNets

- Depthwise-Separable Convolutions
- Split standard convolution into two steps:
- Depthwise Convolution: Applies a single filter per input channel.
- Pointwise Convolution: Combines outputs from depthwise convolution.
- Key Benefit: Reduces computational cost significantly compared to standard convolution.

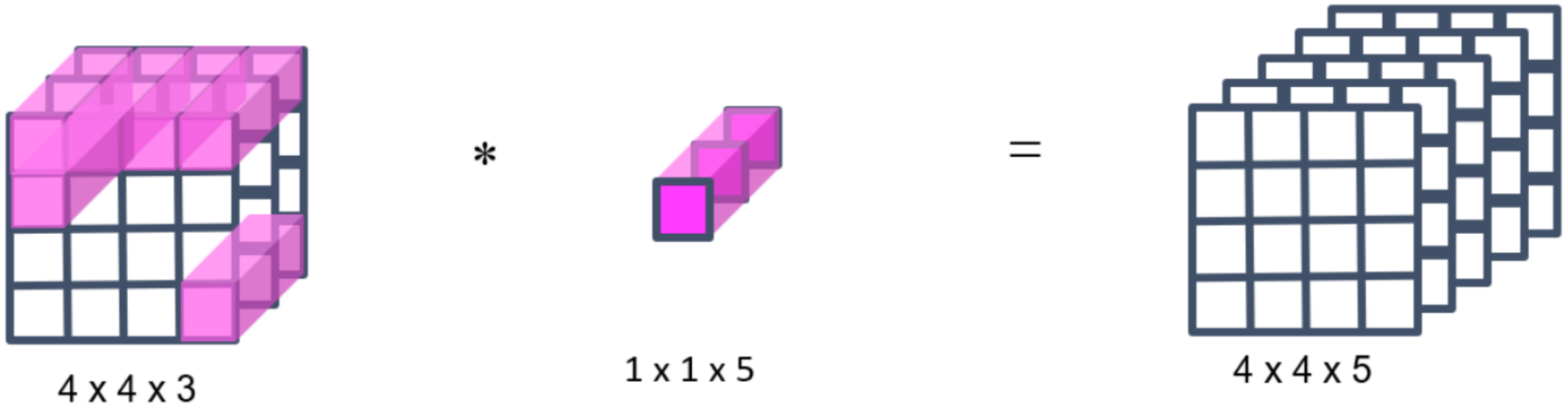


Computational cost = #filter params x # filter positions x # of filters

SOTA CNN Architectures

MobileNets

- Combines outputs from depthwise convolution using 1 × 1 convolutions (mixes channels).

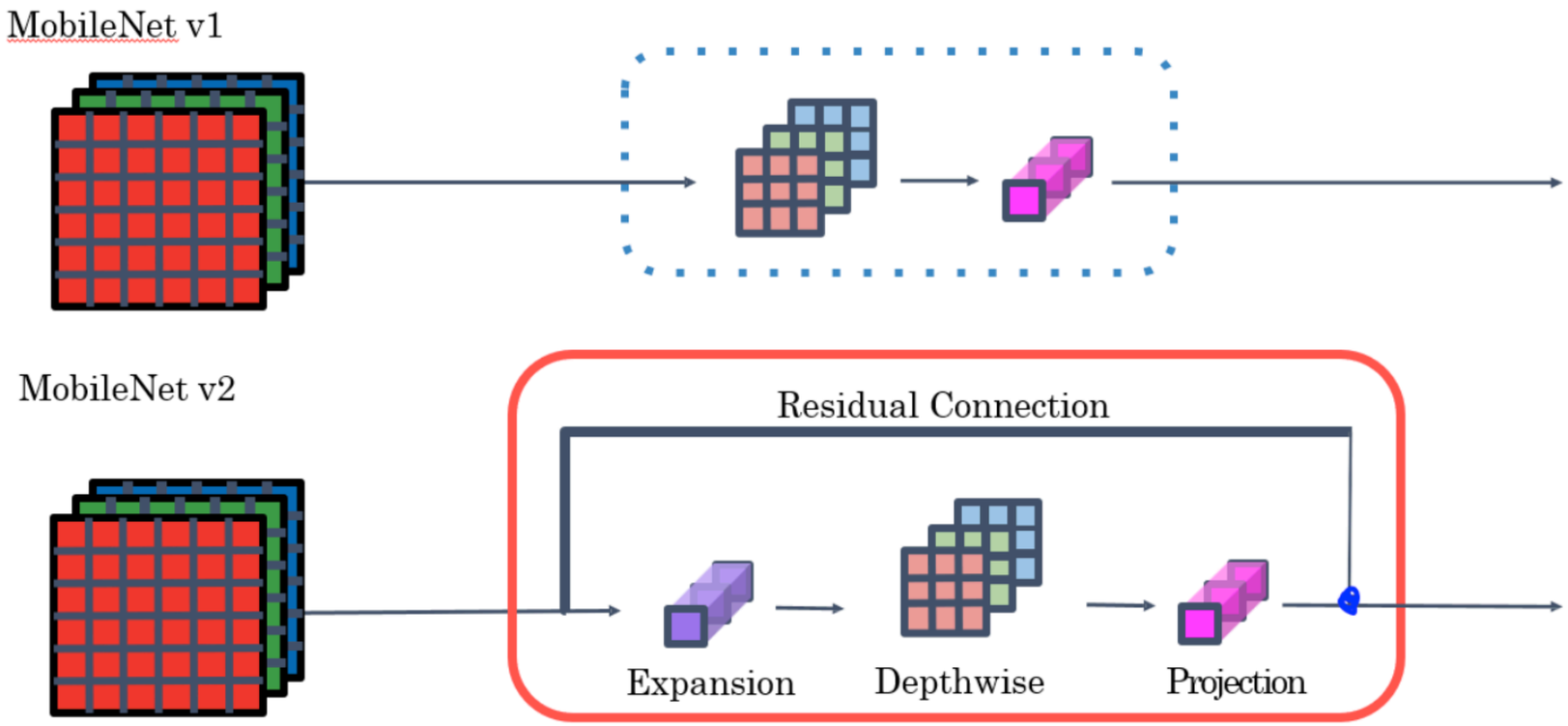


Computational cost = #filter params x # filter positions x # of filters

SOTA CNN Architectures

MobileNet v2:

- Adds residual connections.
- Expansion step: Expands input dimensions before depthwise convolution.
- Projection step: Reduces dimensions after processing.




Q
A

Thank You