

CNNs Architecture: The Convolutional Layer

الشبكات العصبية الالتفافية – عناصر المعمارية



Eng. Mustafa Othman
Data Scientist & Analyst

Today's Outline:

- The Convolutional Layer Basics
- The Convolution Operation
- Convolutional Filter Hyperparameters

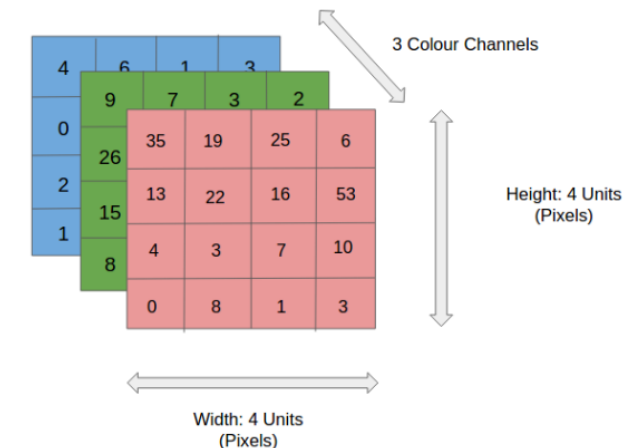
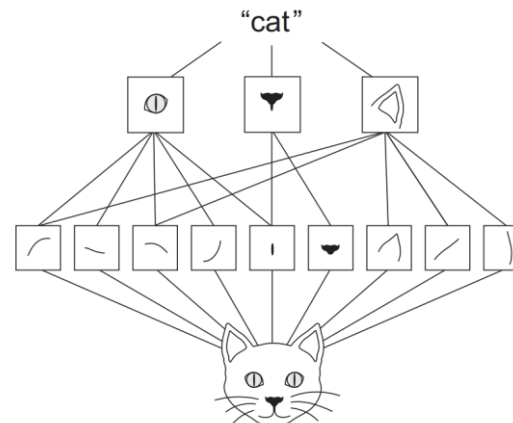
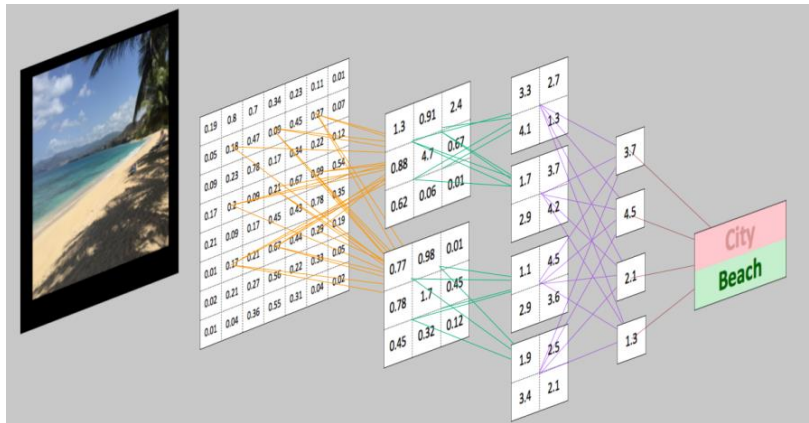


CNNs Architecture: The Convolutional Layer

“Our beauty lies in this extended capacity for convolution.” ~ Thomas Pynchon

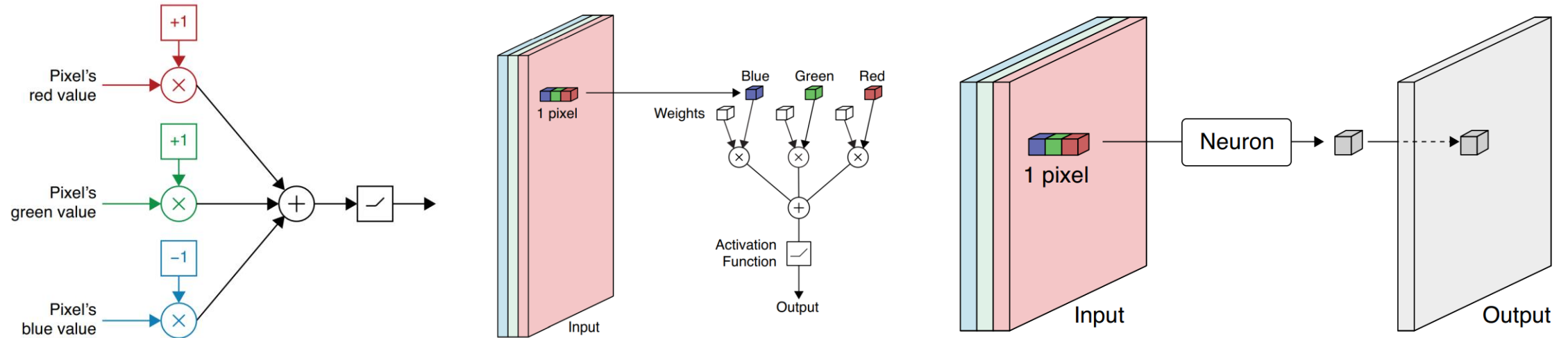
Convolutional Layer Basics (Overview)

- The **convolutional layer (CONV)** is the core building block of a CNN, and it is where most of the **computation** occurs. The primary **purpose** of Convolution in case of a ConvNet is to **extract features** from the input image.
- The **convolution layer** uses **filters (kernels)** that perform **convolution operations** as it is **scanning** the input II with respect to its dimensions. It requires a few components, which are **input data**, a **filter**, and a **feature map**.
- Let's assume that the input will be a **color image**, which is made up of a **matrix of pixels** in 3D. This means that the input will have three dimensions—a **height**, **width**, and **depth**—which correspond to RGB in an image.



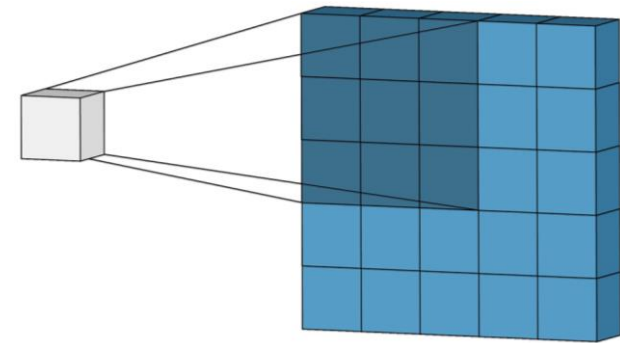
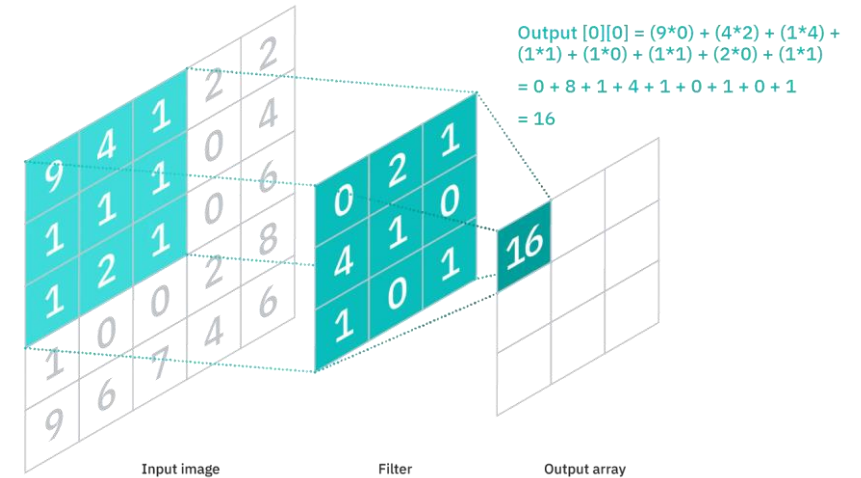
Convolutional Layer Basics

(A Simple Illustrated Example)



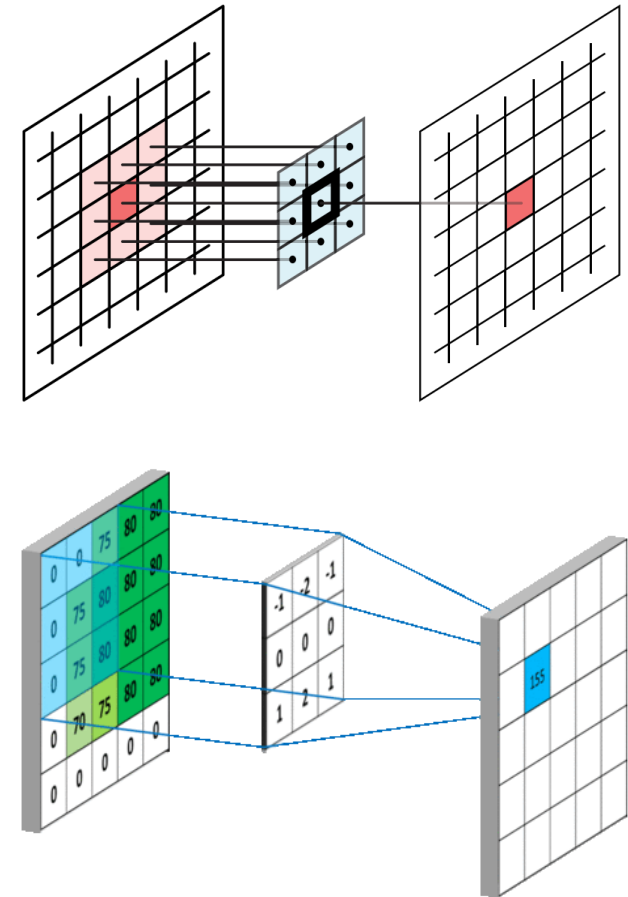
The Convolution Operation (Overview)

- Convolutional layers perform **transformations** on the input data volume that are a function of the **activations** in the input volume and the parameters (weight and biases of the neurons).
- We commonly refer to the sets of **weights** in a convolutional layer as a **filter** (or **kernel**). This filter is convolved with the input and the result is a **feature map** (or **activation map**).
- ConvNets need not be limited to only one Convolutional Layer. Conventionally, the first Conv Layer is responsible for capturing the **Low-Level features** such as **edges, color, gradient orientation**, etc. With added layers, the architecture adapts to the **High-Level features** as well, giving us a network, which has the wholesome understanding of images in the dataset, like how we would.
- Convolutional layers have **parameters** for the layer and additional **hyperparameters**.



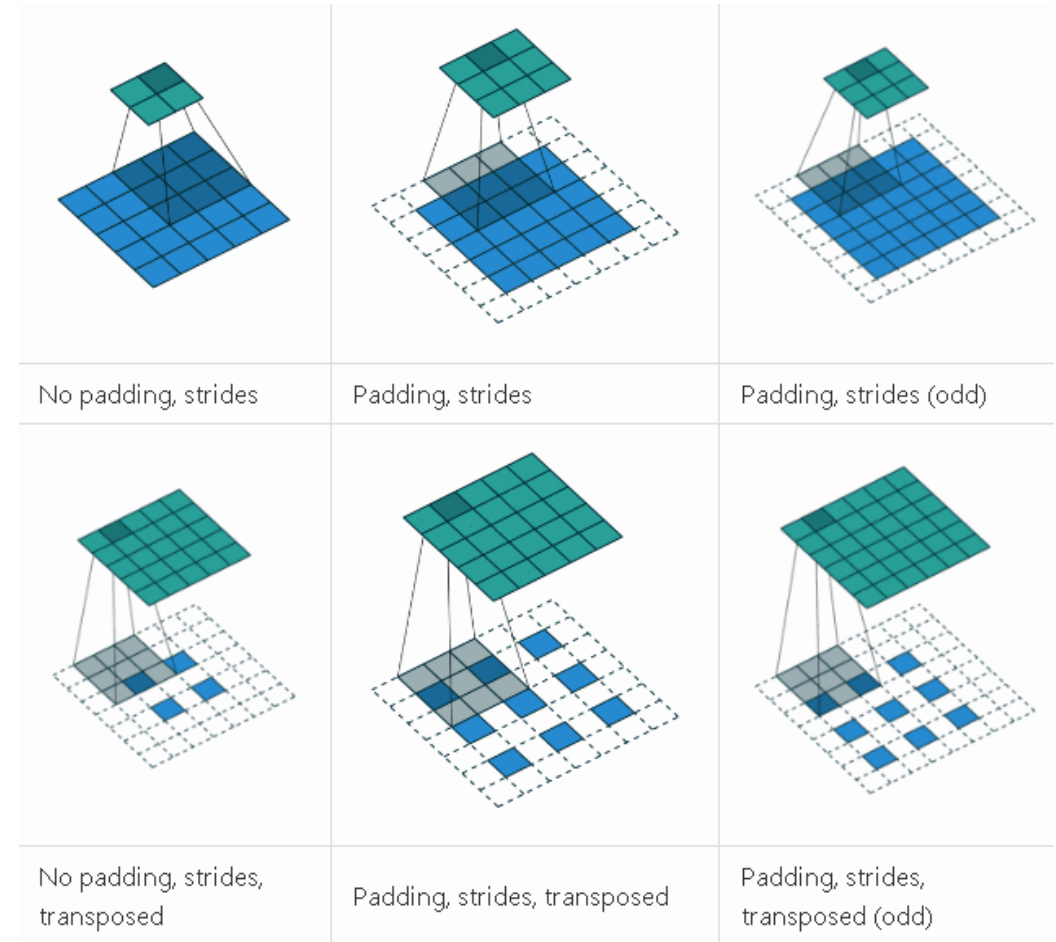
The Convolution Operation (Kernels / Filters)

- **Convolutional layers** consist of sets of **kernels**, which are also known as **filters**. Each of these kernels is a small window (called a **patch**) that **scans** across the image (in more technical terms, the filter **convolves**), from top left to bottom right, and computing the dot product is called the '**Convolved Feature**' or 'Activation Map' or the 'Feature Map'.
- Kernels are made up of **weights**, which—as in dense layers—are learned through **backpropagation**. Kernels can range in size, but a typical size is 3×3.
- Typically, we have **multiple filters** in each convolutional layer. Each filter enables the network to learn a representation of the data at a given layer in a unique way.
- It is important to note that filters acts as **feature detectors** from the original input image.



Convolutional Filter Hyperparameters (Overview)

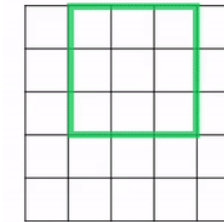
- In contrast with dense layers, convolutional layers are inherently **not fully connected**.
- That is, there isn't a weight mapping every single pixel to every single neuron in the first hidden layer.
- Instead, there are a handful of **hyperparameters** that dictate the number of weights and biases associated with a given convolutional layer. These include:
 - **Kernel size**
 - **Stride length**
 - **Padding**



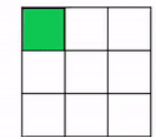
Convolutional Filter Hyperparameters (Padding)

- **Zero-padding:** Sometimes, it is convenient to pad the input matrix with **zeros** around the border, so that we can apply the filter to bordering elements of our input image matrix. It allows us to control the size of the feature maps.
- **Valid padding:**
 - This is also known as no padding. In this case, the last convolution is dropped if dimensions do not align.
- **Same padding:**
 - This padding ensures that the output layer has the same size as the input layer
- **Full padding:**
 - This type of padding increases the size of the output by adding zeros to the border of the input.

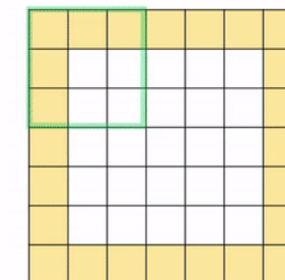
Original Image, 5x5



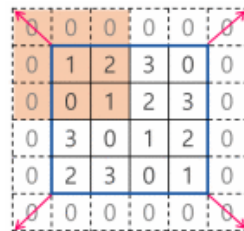
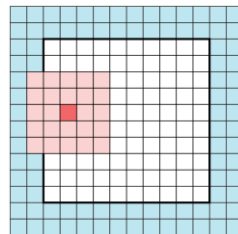
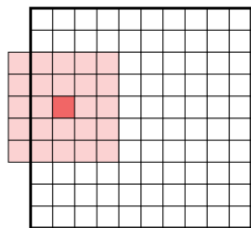
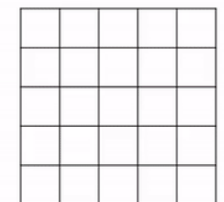
Integral Image, 3x3



Original Image, 5x5



Integral Image, 5x5



\odot

2	0	1
0	1	2
1	0	2

\rightarrow

7	12	10	2
4	15	16	10
10	6	15	6
8	10	4	3



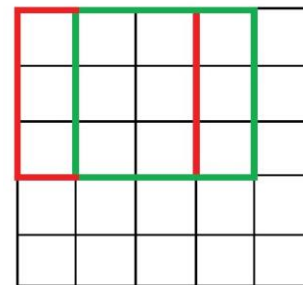
Convolutional Filter Hyperparameters (Strides)

- For a convolutional or a pooling operation, the **stride** S denotes the number of pixels by which the window **moves** after each operation.
- If the stride is **one**, we move the filters one pixel at a time. **Higher** the stride, **smaller** output volumes will be produced spatially.
- While stride values of **two** or greater is rare, a larger stride yields a smaller output. Having a larger stride will produce smaller feature maps.
- When the stride is 2, then the filters jump 2 pixels at a time as we slide them around.

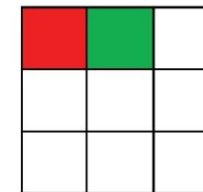
0 ₂	0 ₀	0 ₁	0	0	0	0
0 ₁	2 ₀	2 ₀	3	3	3	0
0 ₀	0 ₁	1 ₁	3	0	3	0
0	2	3	0	1	3	0
0	3	3	2	1	2	0
0	3	3	0	2	3	0
0	0	0	0	0	0	0

1	6	5
7	10	9
7	10	8

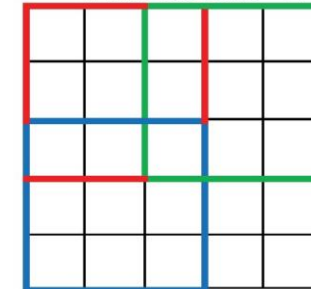
Convolution with Stride=1



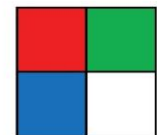
Output



Convolution with Stride=2

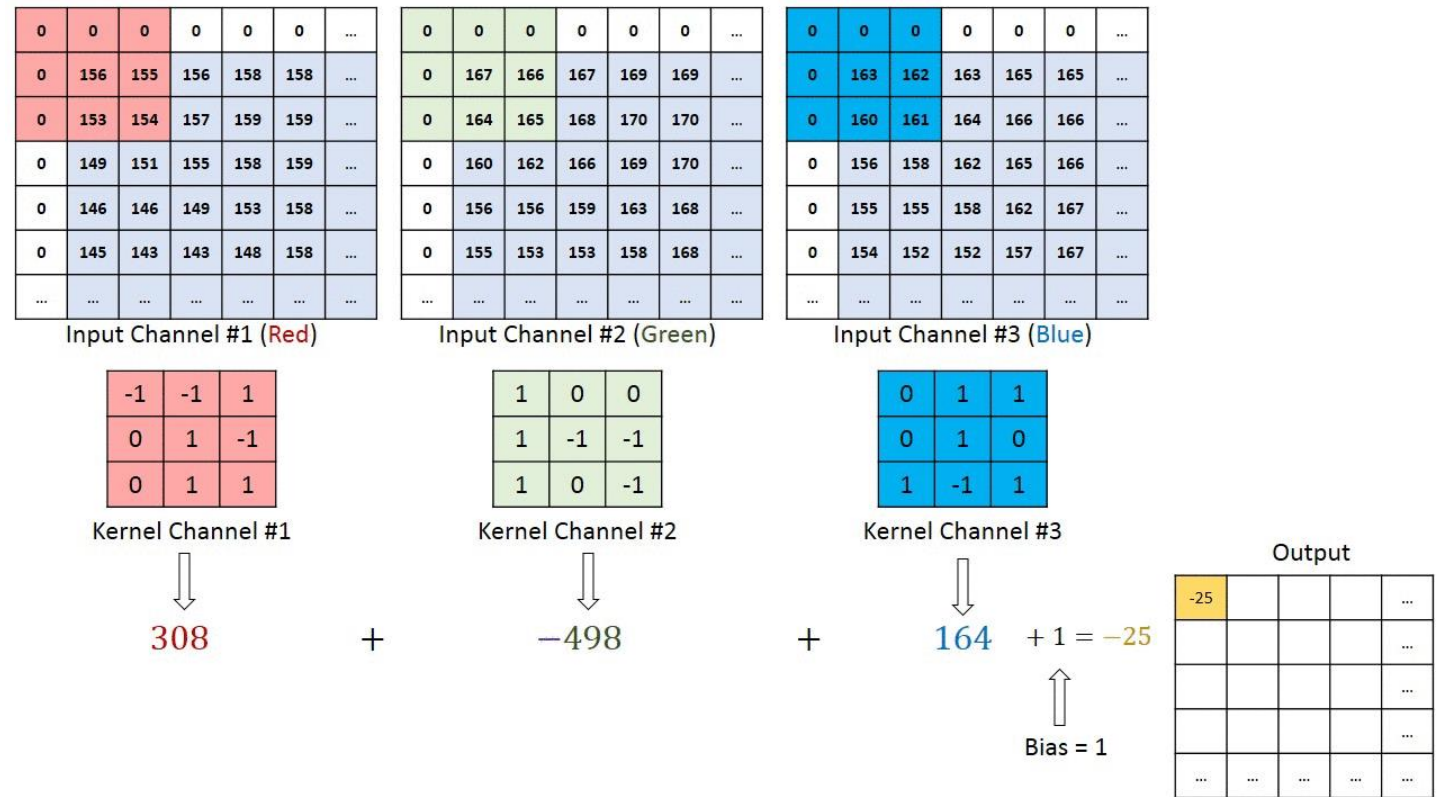


Output



Final Thoughts (Full Illustrated Example)

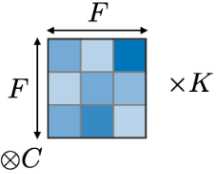
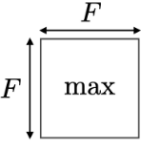
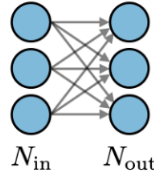
- The filter moves to the right with a certain Stride Value till it parses the complete width.
- Moving on, it hops down to the beginning (left) of the image with the same Stride Value and repeats the process until the entire image is traversed.
- In the case of images with multiple channels (e.g. RGB), the Kernel has the same depth as that of the input image.



Final Thoughts

(Understanding Model Complexity)

- In order to assess the **complexity** of a model, it is often useful to determine the **number of parameters** that its architecture will have.
- Note that; **I** is the length of the input volume size, **F** is the length of the filter, **K** is the number of kernels, **C** is for channels, **S** is the stride, and the output size **O** of the feature map along that dimension.

	CONV	POOL	FC
Illustration			
Input size	$I \times I \times C$	$I \times I \times C$	N_{in}
Output size	$O \times O \times K$	$O \times O \times C$	N_{out}
Number of parameters	$(F \times F \times C + 1) \cdot K$	0	$(N_{in} + 1) \times N_{out}$
Remarks	<ul style="list-style-type: none"> One bias parameter per filter In most cases, $S < F$ A common choice for K is $2C$ 	<ul style="list-style-type: none"> Pooling operation done channel-wise In most cases, $S = F$ 	<ul style="list-style-type: none"> Input is flattened One bias parameter per neuron The number of FC neurons is free of structural constraints

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_1 (Conv2D)	(None, 11, 11, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 64)	0
conv2d_2 (Conv2D)	(None, 3, 3, 64)	36928
flatten (Flatten)	(None, 576)	0
dense (Dense)	(None, 64)	36928
dense_1 (Dense)	(None, 10)	650

Further Readings

- Deep Learning Illustrated, Jon Krohn
 - Chapter 10
- Deep Learning with Python, François Chollet
 - Chapter 5
- Deep Learning: A Visual Approach, Andrew Glassner
 - Chapter 16



THANKS

Keep Moving Forward! 😊



Eng. Mustafa Othman
Data Scientist & Analyst