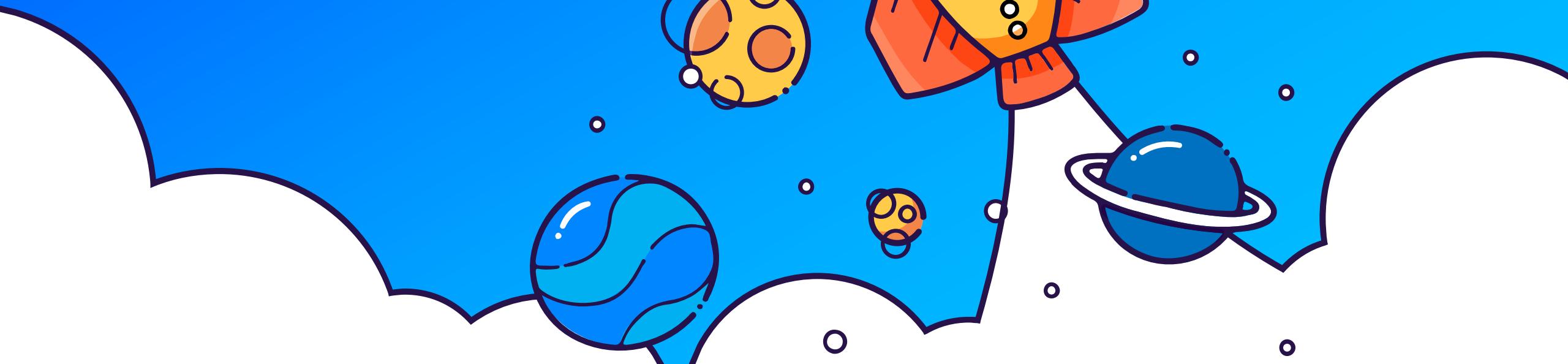


# Redes Neuronales Convolucionales con Pytorch

Christian Tutivén





# Ph.D. Christian Tutivén Gálvez

- Doctorado en Automática, Robótica y Visión de la Universidad Politécnica de Cataluña (UPC).
- Maestría en Administración de Empresas de la Universidad Santa María de Chile.
- Maestría en Telecomunicaciones de la Universidad Católica Santiago de Guayaquil.
- Ingeniería en Telecomunicaciones de la Universidad Católica Santiago de Guayaquil.
- Miembro del grupo de investigación CoDALab (Control, Dinámica y Aplicaciones) de la UPC.
- Investigador y catedrático en la Escuela Superior Politécnica del Litoral.
- Profesor en la materia de Deep Learning en la Universidad de Nariño (Pasto, Colombia.).
- Actualmente coordinador de proyectos:
  - Detección y aislamiento de fallos y daños en turbinas de viento (ESPOL - UPC).
  - Desarrollo de veleros autónomos (ESPOL -UPC).
- Experiencia en el sector industrial y tecnologías.

/01

## Introducción

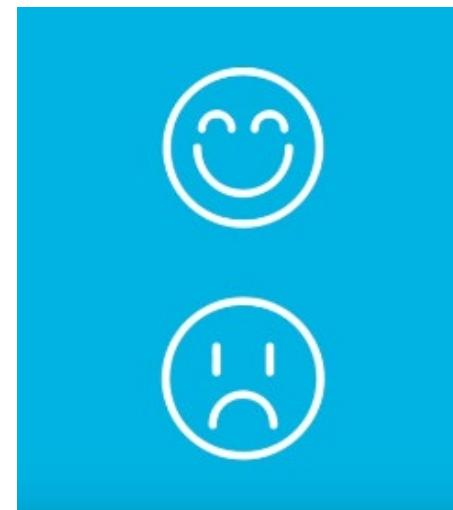


# Estado del arte en una variedad de problemas

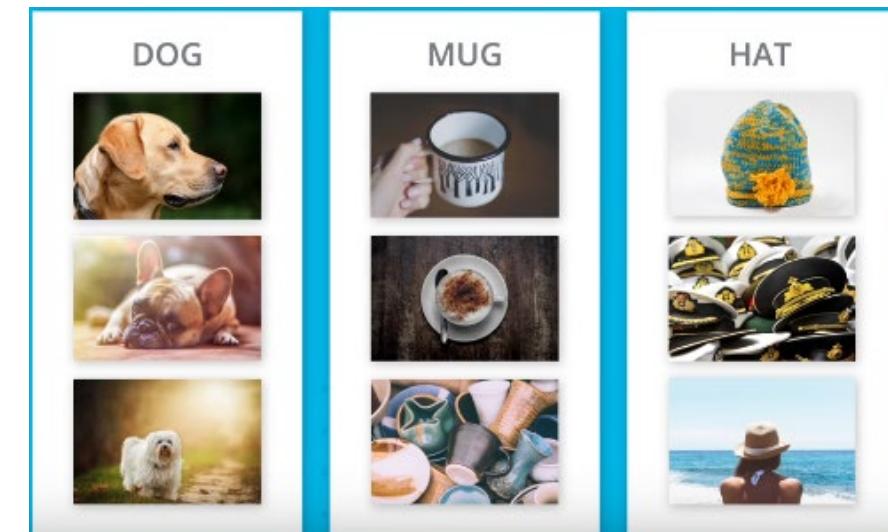
Interfaz de voz de usuario



Procesamiento de lenguaje natural



Visión por computadora



Wavenet

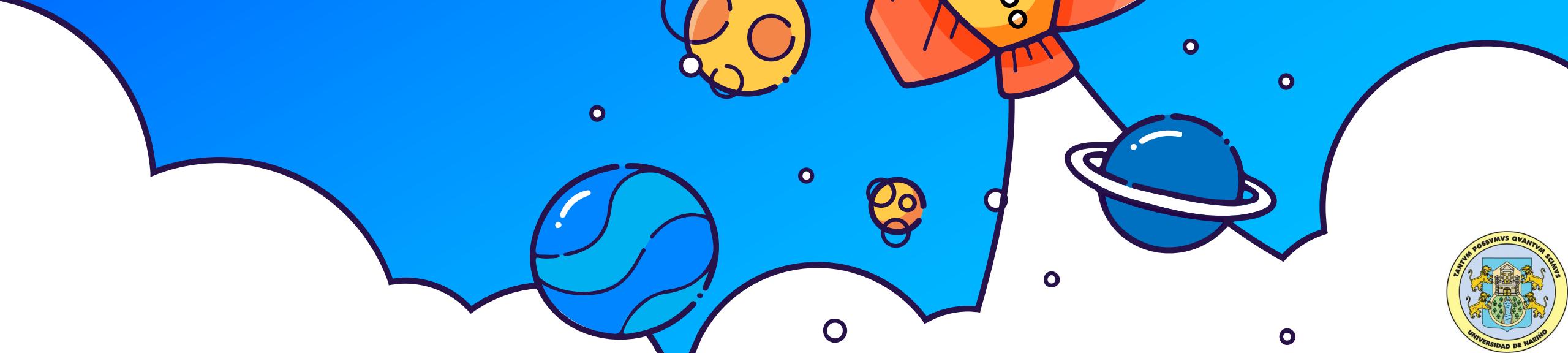
<https://deepmind.com/blog/article/wavenet-generative-model-raw-audio>

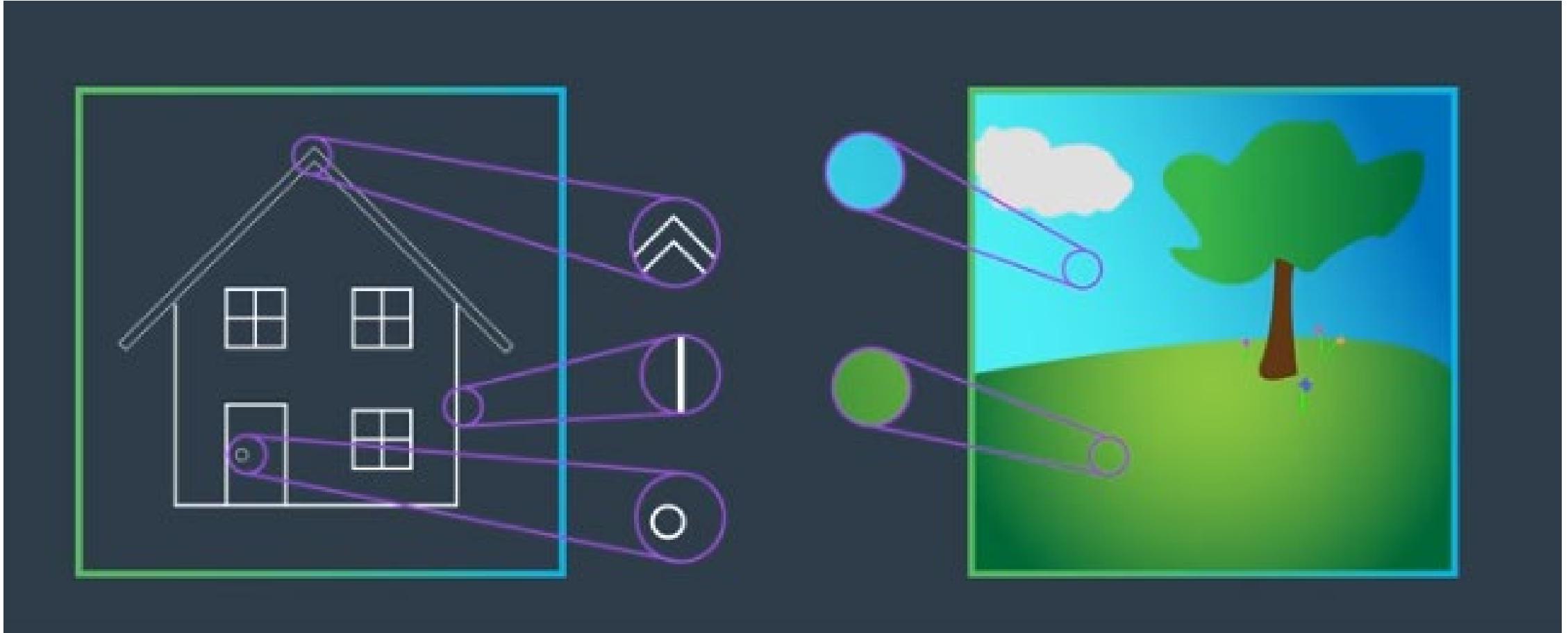
Análisis de sentimientos

Clasificación de imágenes

# /02

## Características (Features)





Las redes convolucionales ven las imágenes como un todo y aprenden a identificar patrones espaciales, como colores, sombras o donde una textura es borrosa o suave.

Los colores y sombras que definen a cualquier imagen y a cualquier objeto son usualmente llamados características y son estas las que una CNN debe aprender a identificar y usar para clasificar las imágenes.

A large, abstract graphic on the left side of the slide features a dark blue base layer with irregular white and light blue splatters and streaks, resembling liquid or smoke. It has a textured, organic appearance.

# ¿Qué es una característica?

Una forma útil de pensar acerca de qué es una característica es pensar en lo que nos atrae visualmente cuando vemos por primera vez un objeto y cuando identificamos diferentes objetos. Por ejemplo, ¿qué observamos para distinguir un gato de un perro? La forma de los ojos, el tamaño y cómo se mueven son solo algunos ejemplos de características visuales.

Como otro ejemplo, digamos que vemos a una persona caminando hacia nosotros y queremos ver si es alguien que conocemos; podemos mirar su cara, y aún más su forma general, ojos (e incluso el color de sus ojos). ¡La forma distintiva de una persona y su color de ojos son excelentes ejemplos de características distintivas!

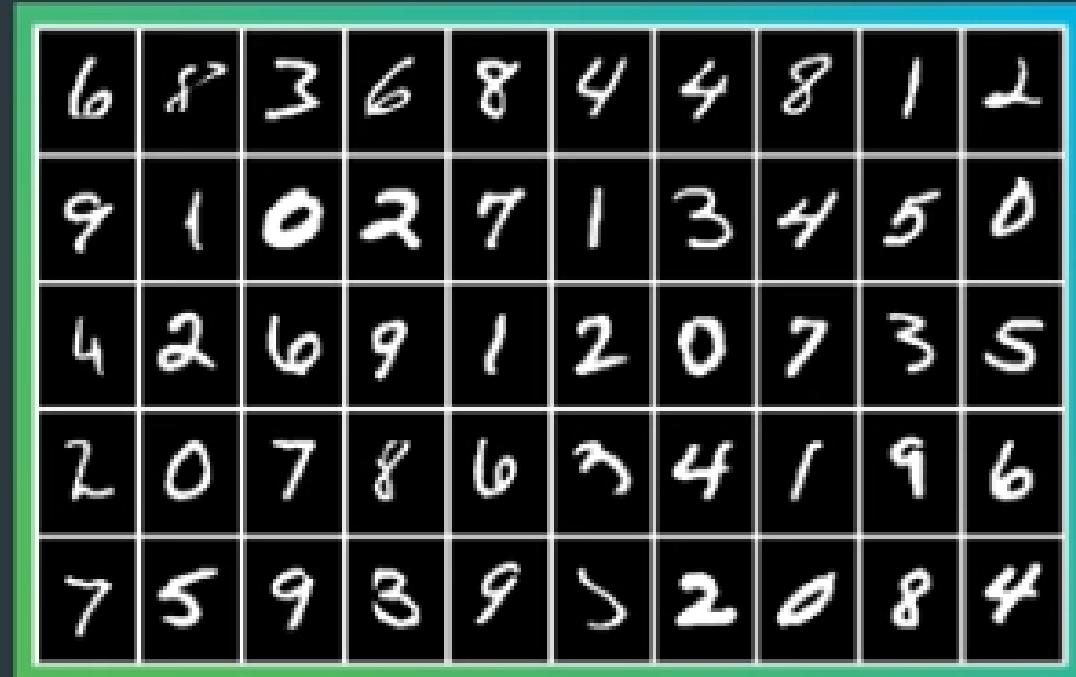
## MNIST Database:

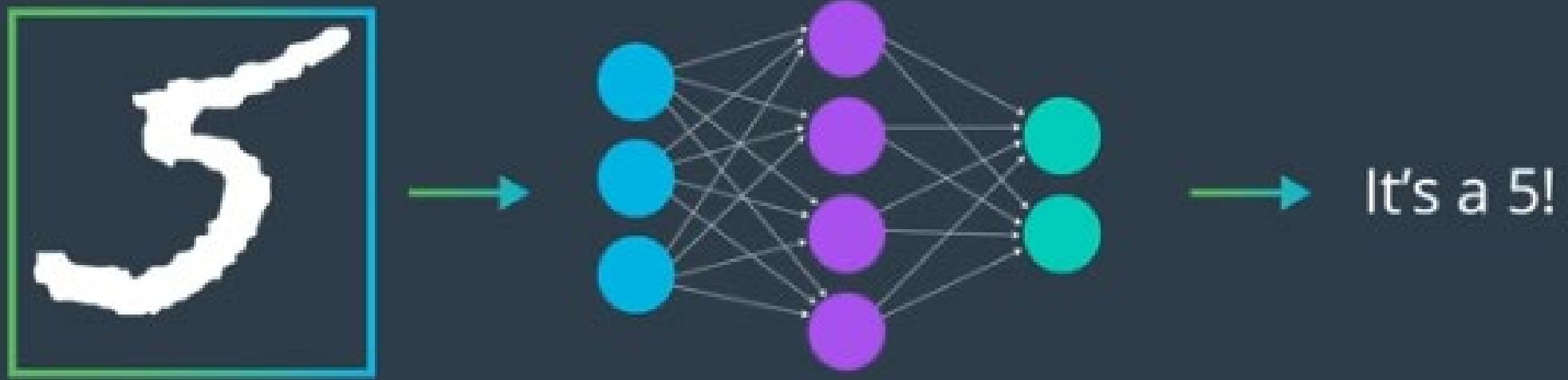
70,000 images of  
hand-written digits

Famous database in  
machine learning

¡La base de datos MNIST es posiblemente la base de datos más famosa en el campo del aprendizaje profundo!

[\(<https://www.kaggle.com/benhamner/popular-datasets-over-time>\)](https://www.kaggle.com/benhamner/popular-datasets-over-time)



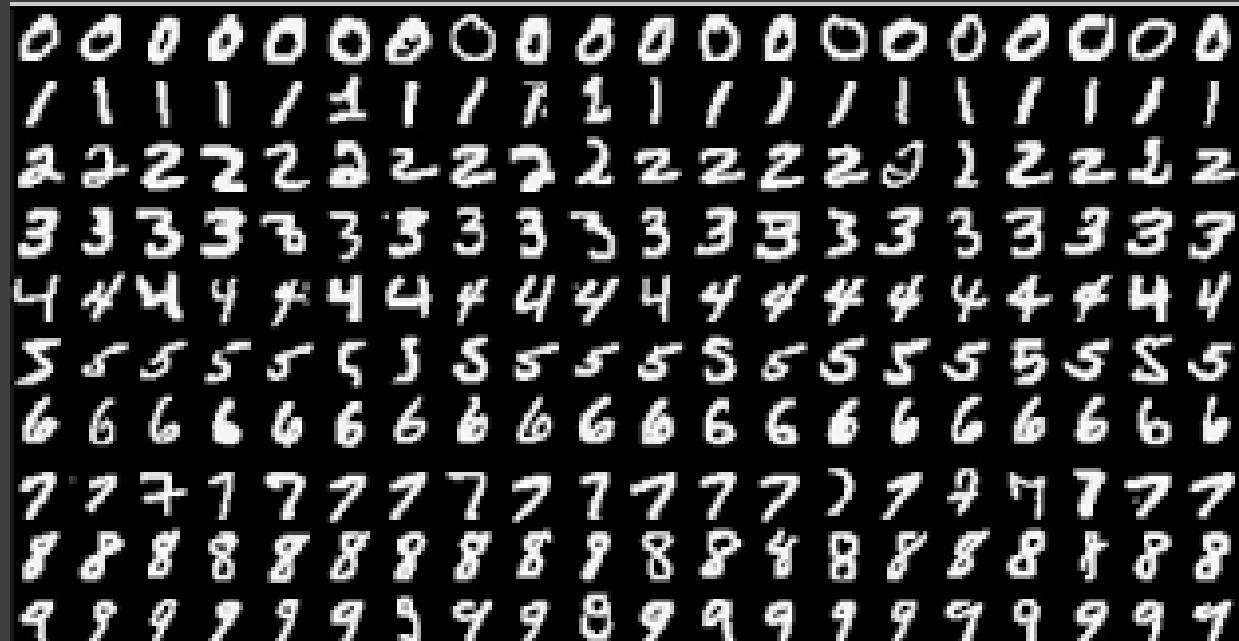


Queremos diseñar un clasificador de imágenes que tome una imagen de un número escrito a mano y produzca una clase predicha para dicho número

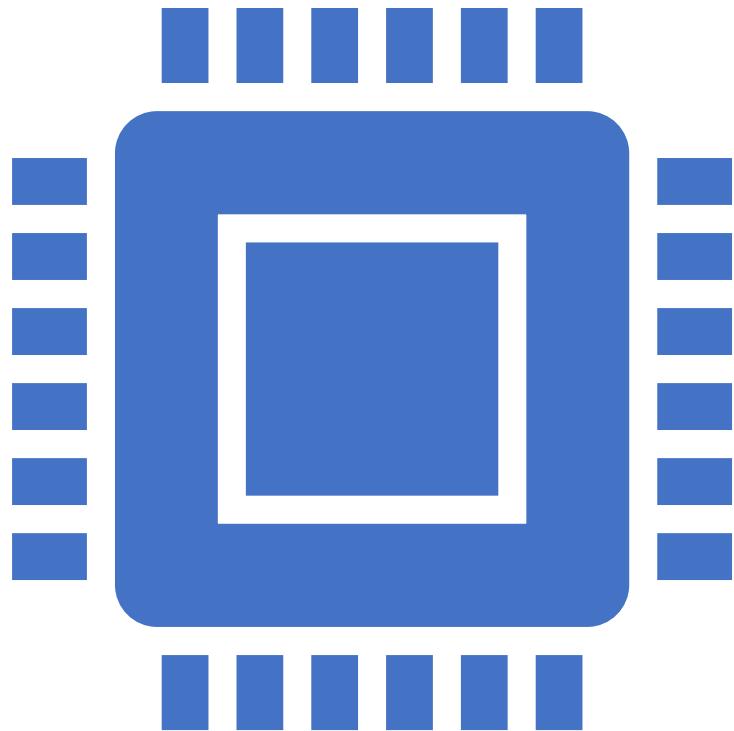
# MNIST Database

¡La base de datos MNIST es posiblemente la base de datos más famosa en el campo del aprendizaje profundo!

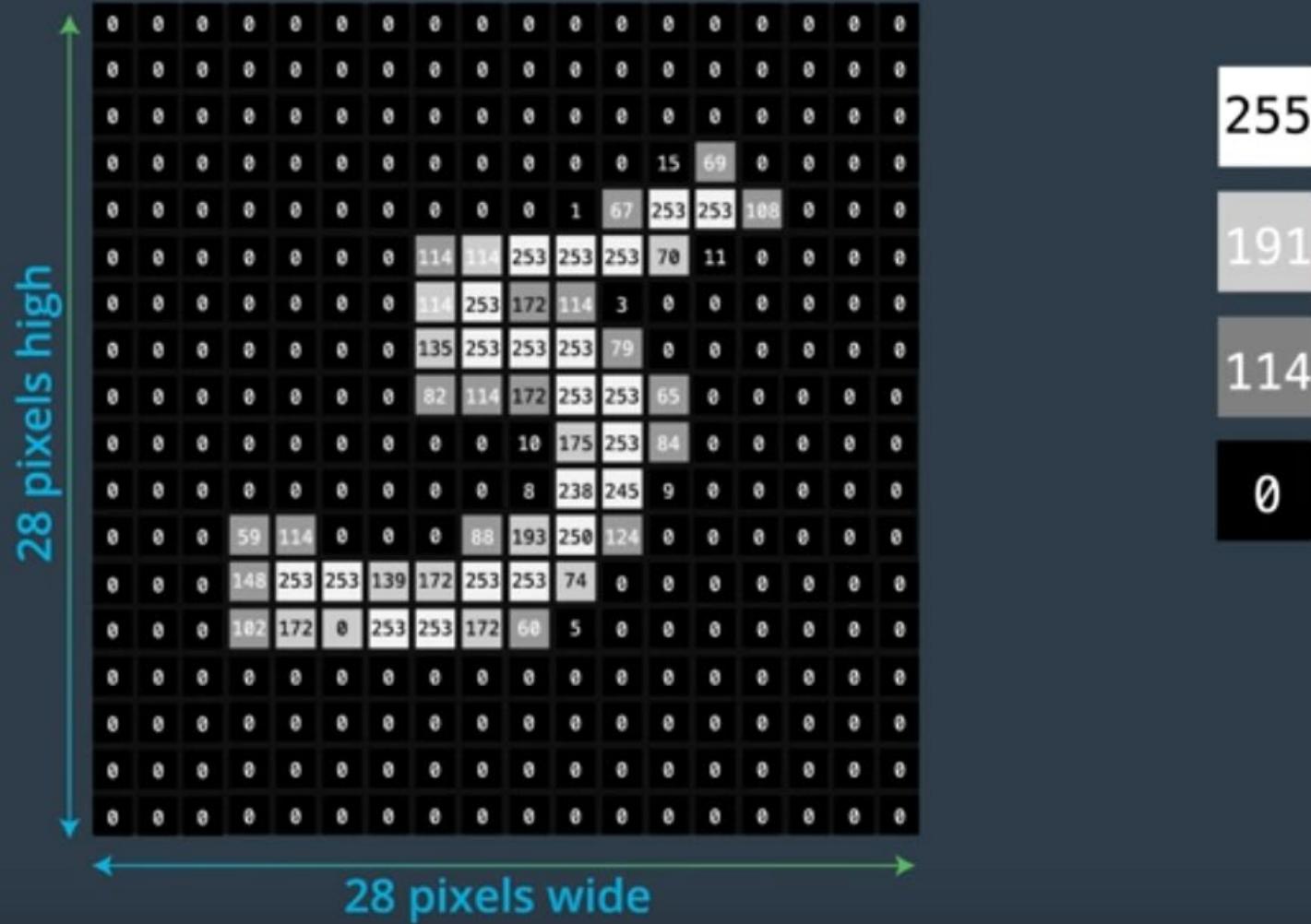
[\(https://www.kaggle.com/benhamner/popular-datasets-over-time\)](https://www.kaggle.com/benhamner/popular-datasets-over-time)



Cómo las  
computadoras ven  
a las imágenes?

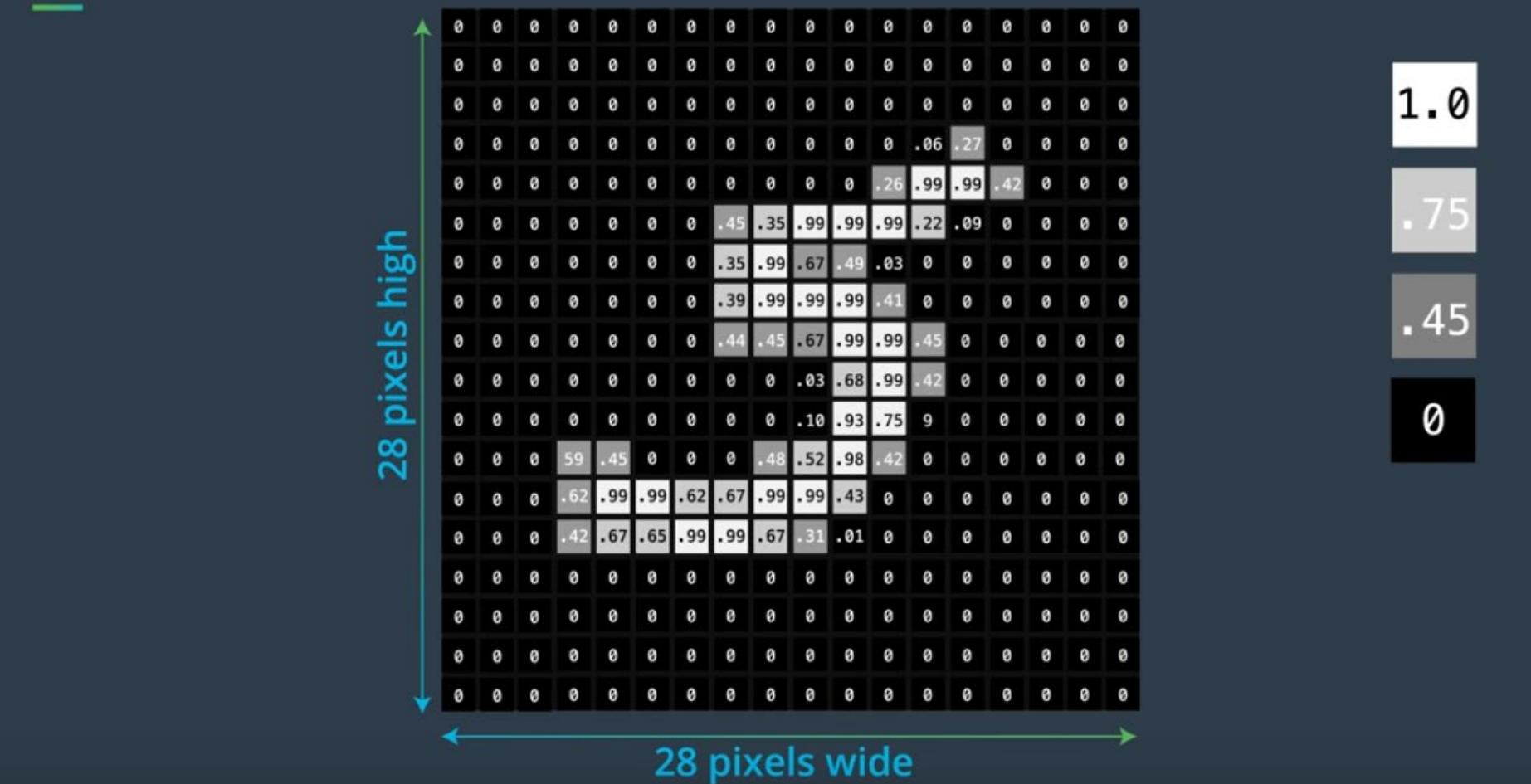


## Visualize the data:



- Una imagen en escala de grises es interpretada por una computadora como un arreglo.
- Cada celda es llamado pixel y tienen un valor numérico.

## Pre-process the data:



- Las imágenes del MNIST dataset han sido pre procesadas.
  - Se escalo los valores de 0 a 1.

# Normalización de entradas de imagen

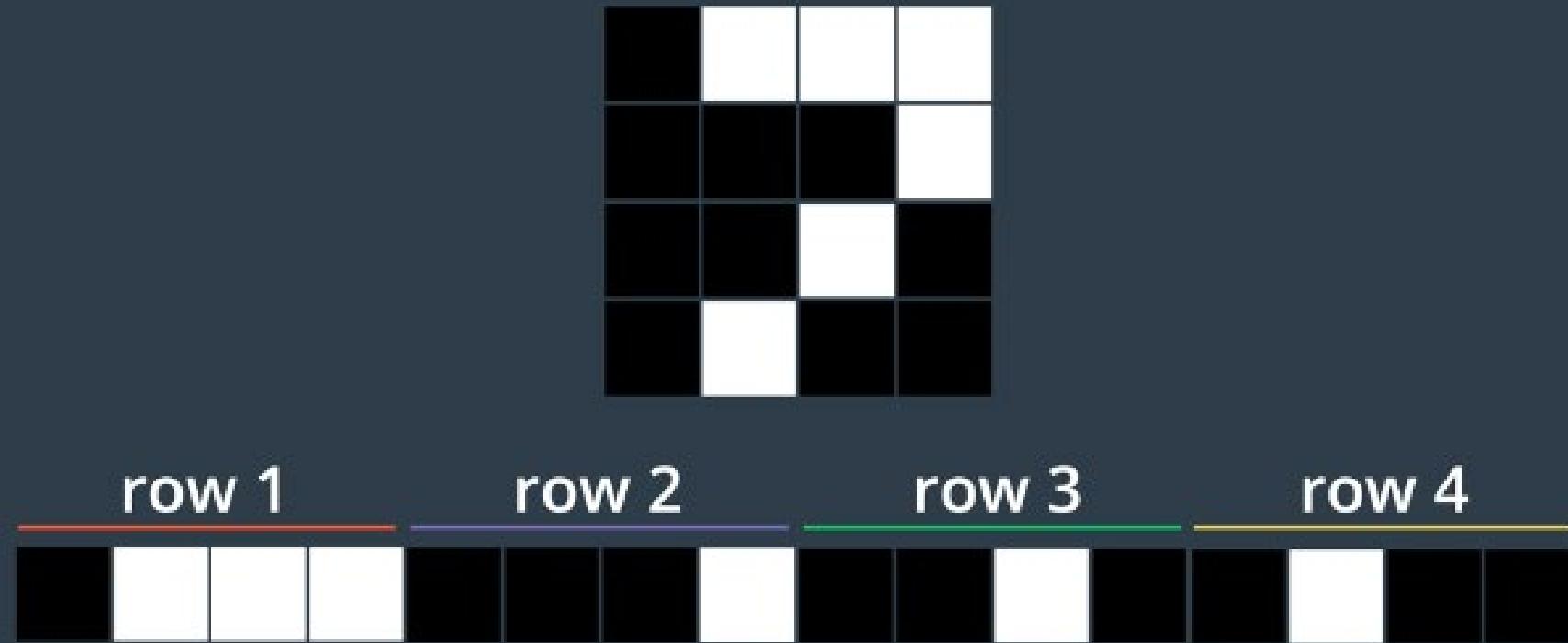
La normalización de datos es un paso importante que asegura que cada entrada (cada valor de píxel, en este caso) proviene de una distribución estándar. Es decir, el rango de valores de píxeles en una imagen de entrada es el mismo que el rango en otra imagen. ¡Esta estandarización hace que nuestro modelo entrene y alcance un error mínimo, más rápido!

La normalización de datos generalmente se realiza restando la media (el promedio de todos los valores de píxel) de cada píxel y luego dividiendo el resultado por la desviación estándar de todos los valores de píxel. A veces veremos una aproximación aquí, donde usamos una desviación estándar y media de 0.5 para centrar los valores de píxeles ([leer](#)).

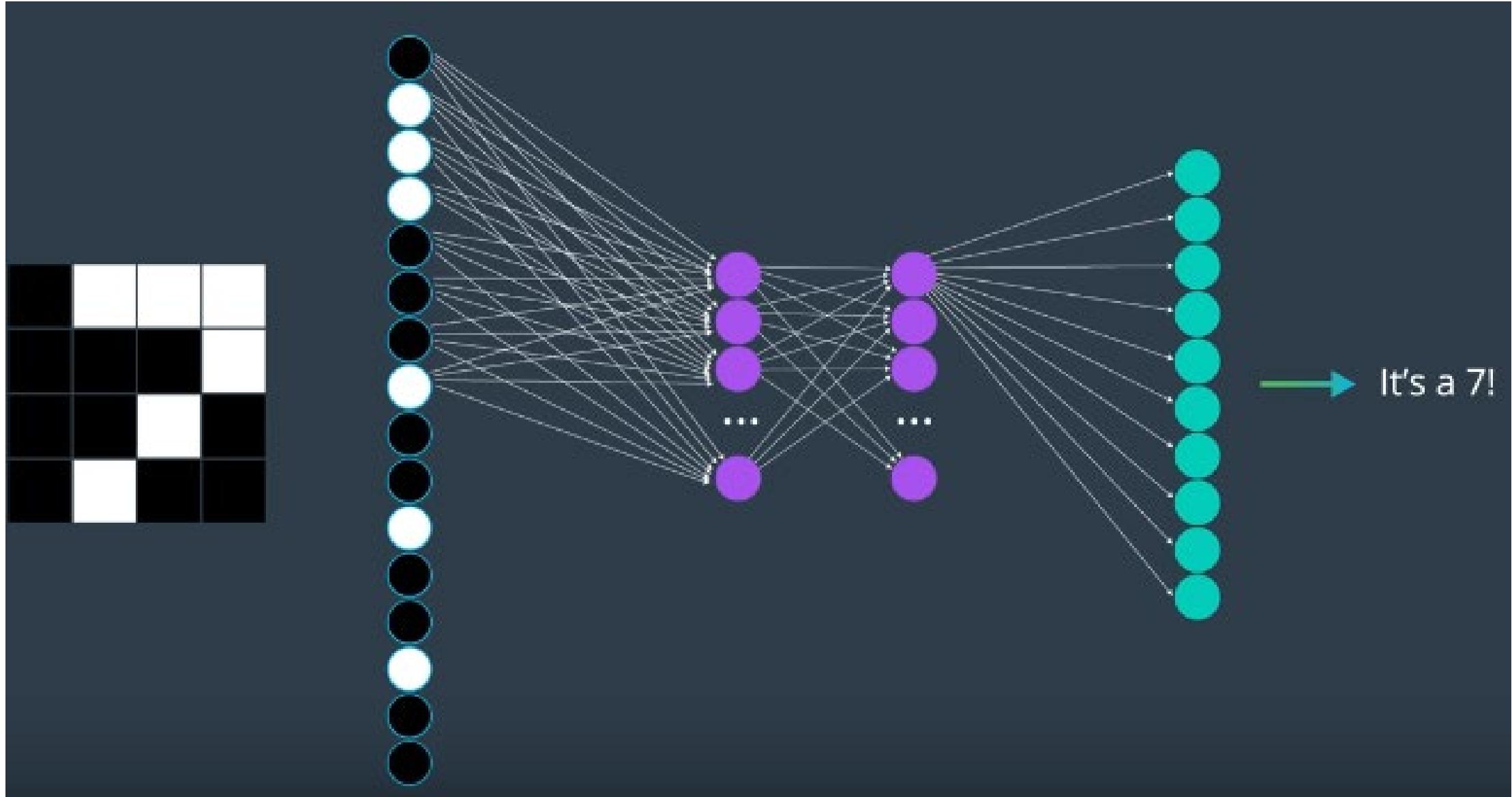
La distribución de tales datos debe parecerse a una función gaussiana centrada en cero. Para las entradas de imagen, necesitamos que los números de píxeles sean positivos, por lo que a menudo elegimos escalar los datos en un rango normalizado [0,1] ([revisar](#)).

# Cómo clasificar las imágenes?

Flattening



# Flattening

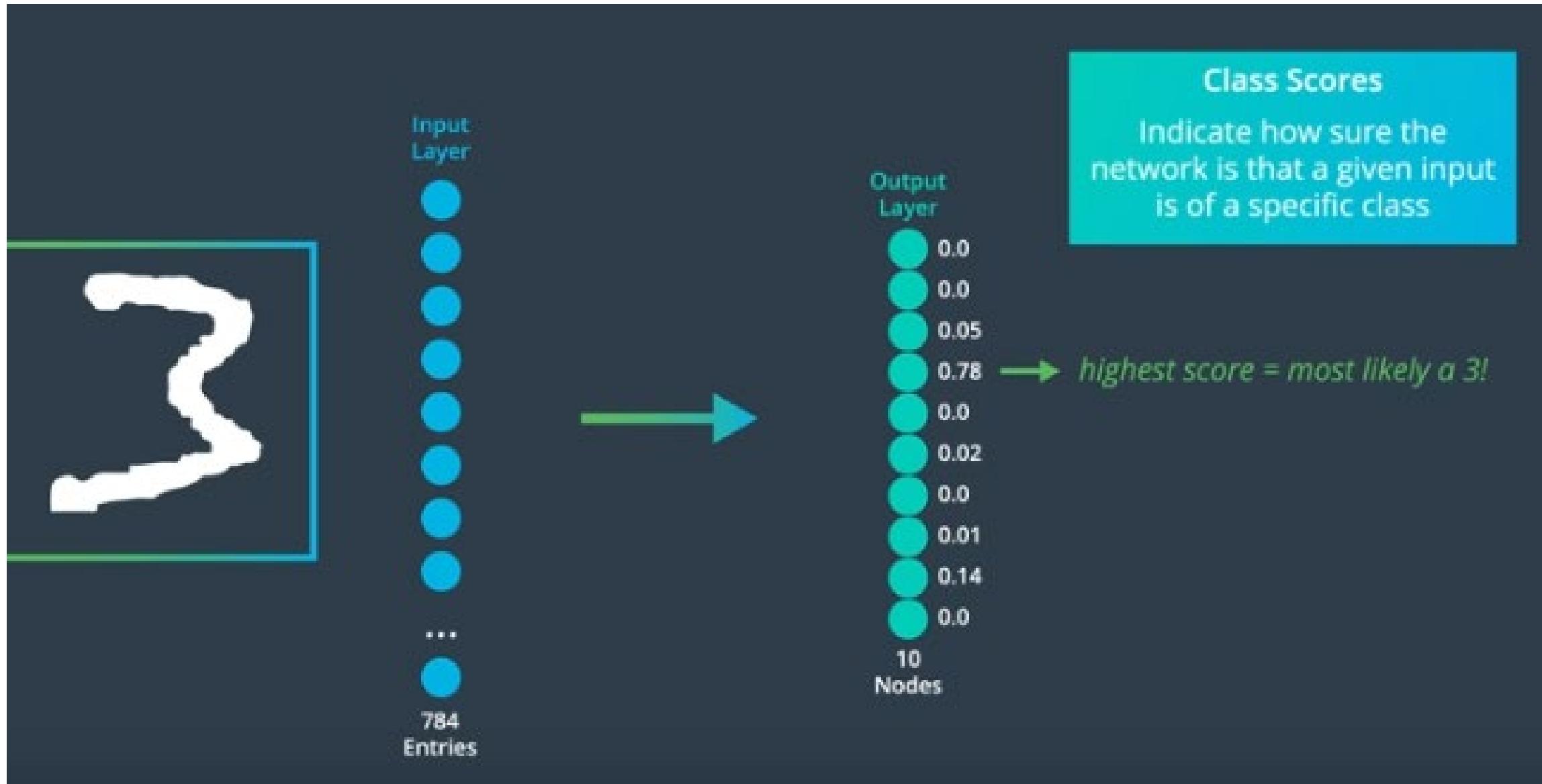




Crear un red neuronal para descubrir patrones en nuestros datos.



Luego ver como se comporta en datos de validación.



## Output Layer

0.0  
0.0  
0.05  
0.78  
0.0  
0.02  
0.0  
0.01  
0.14  
0.0

10  
Nodes

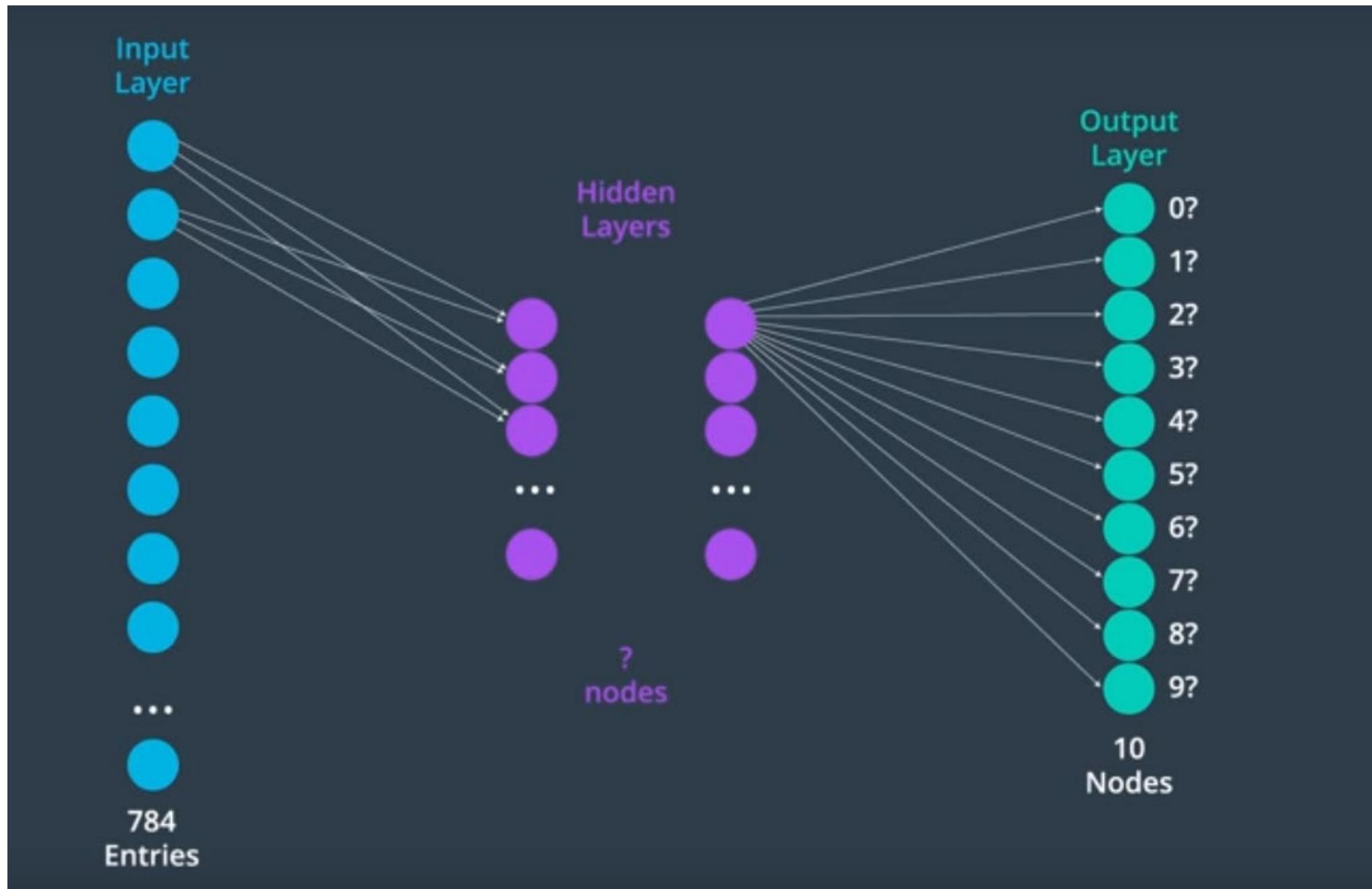
## Class Scores

Confidence in this Class

0 1 2 3 4 5 6 7 8 9

Class





# ¿Cómo definir la estructura de mi red?



Buscar trabajos  
relacionados al tuyo.

Leer artículos.

Buscar el estado del  
arte.

/03

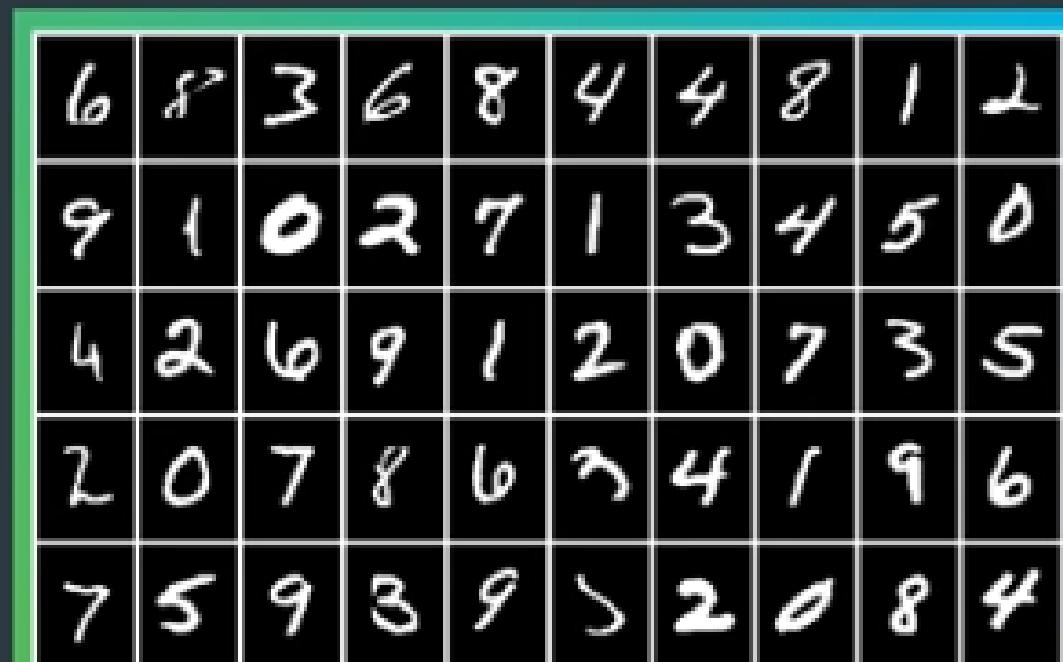
MLP vs CNN



## MNIST Database:

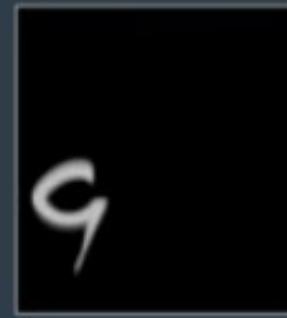
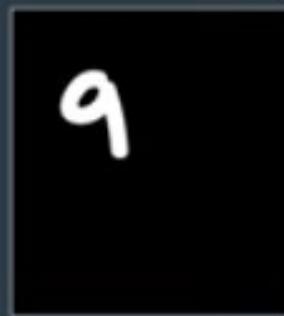
70,000 images of  
hand-written digits

Famous database in  
machine learning



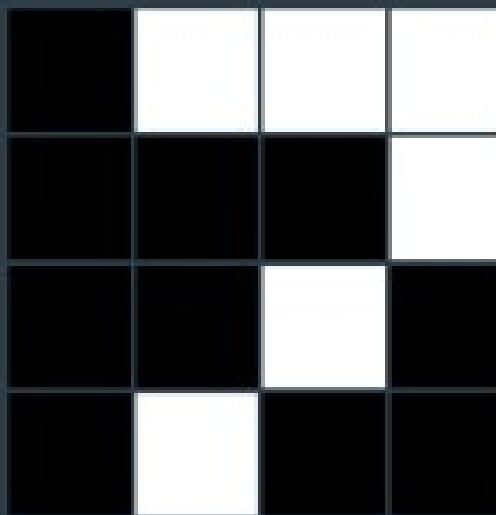


VS



# Flattening

---

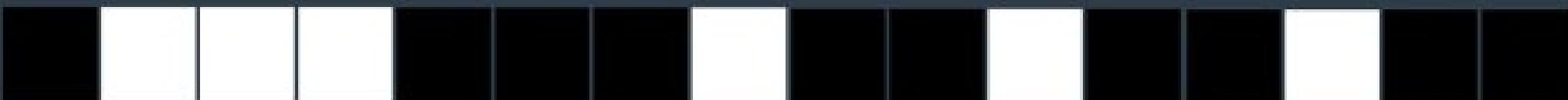


row 1

row 2

row 3

row 4

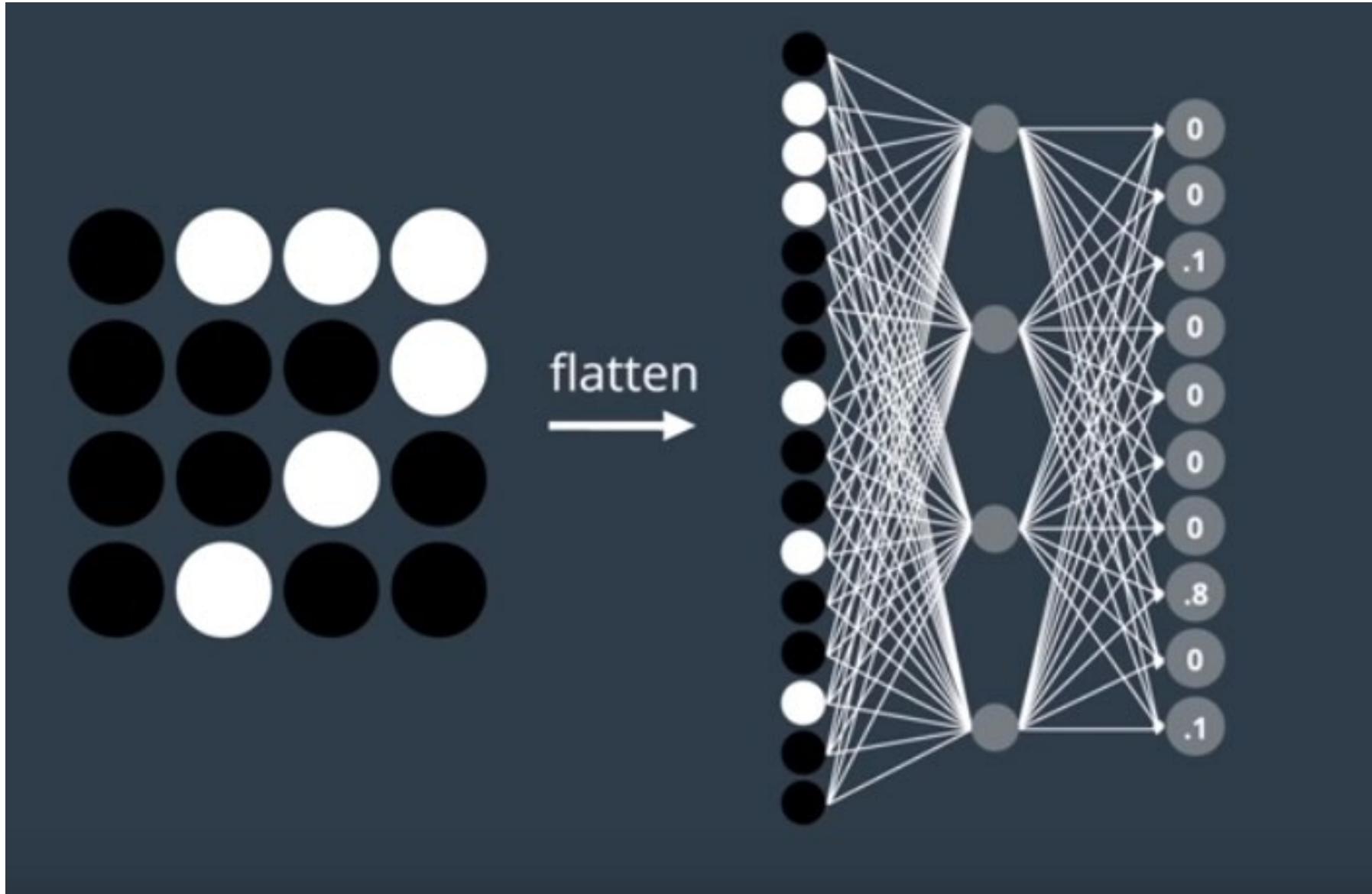


# MLPs

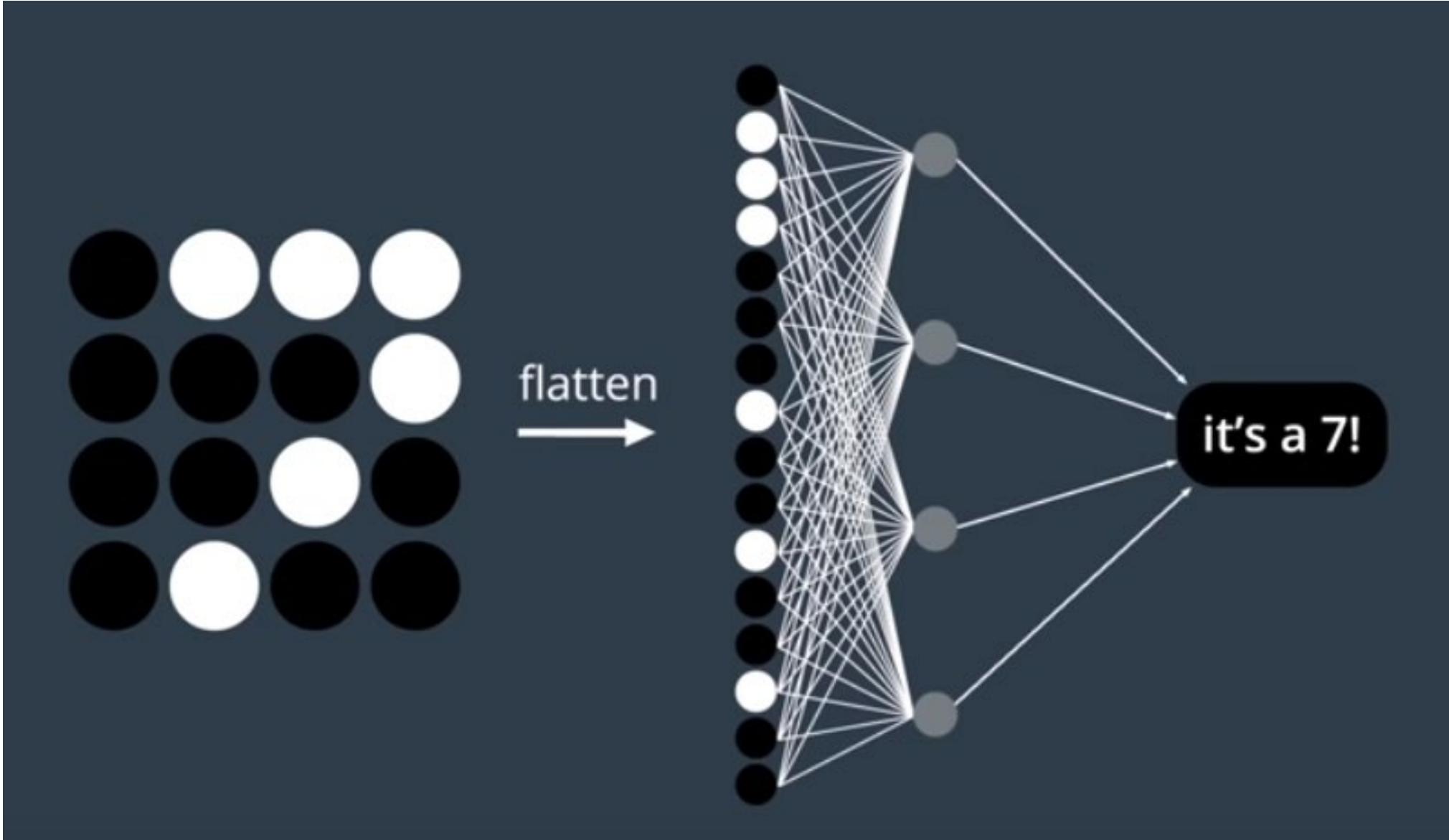
- Only use **fully** connected layers
- Only accept **vectors** as input

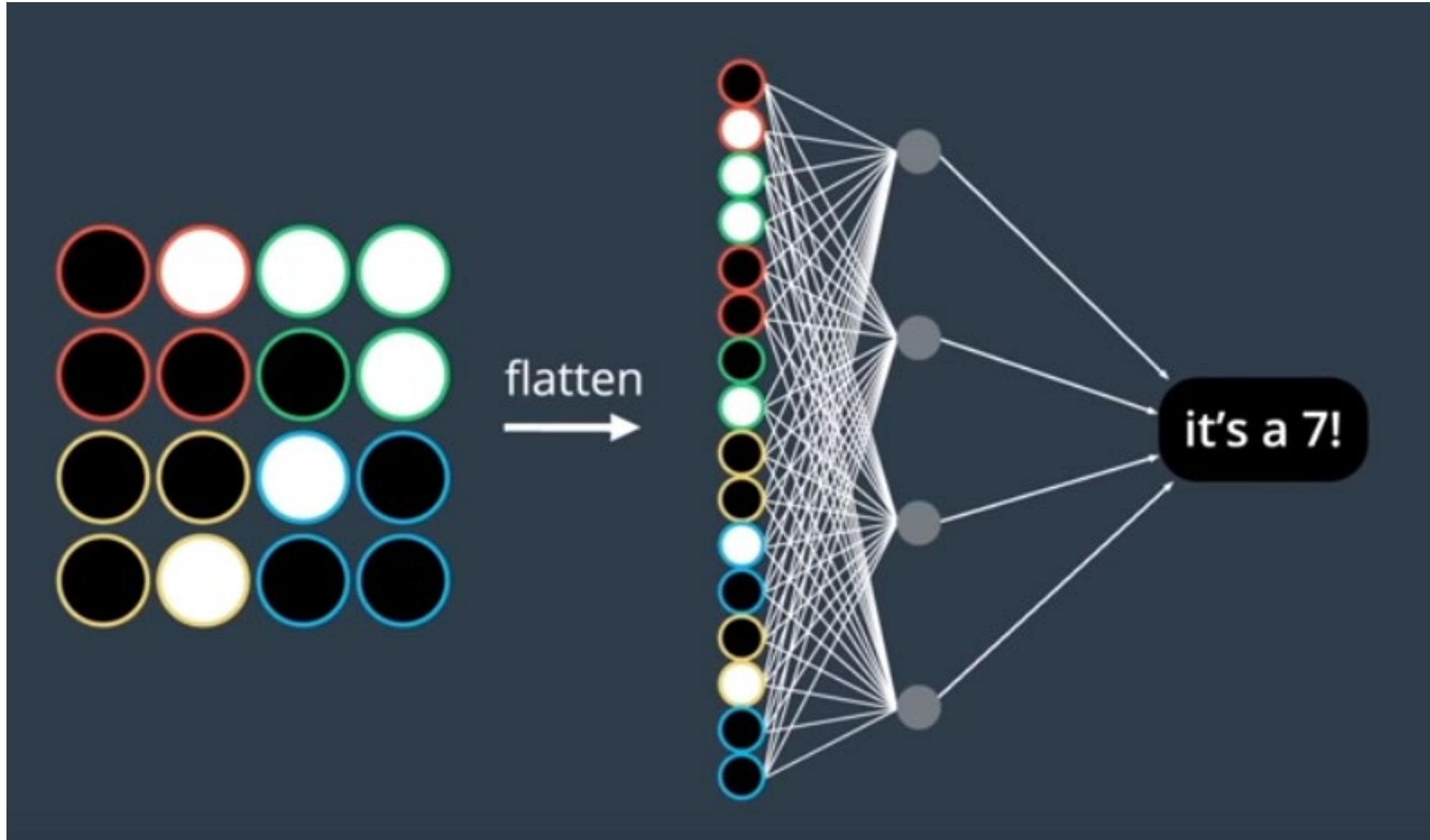
# CNNs

- Also use **sparsely** connected layers
- Also accept **matrices** as input

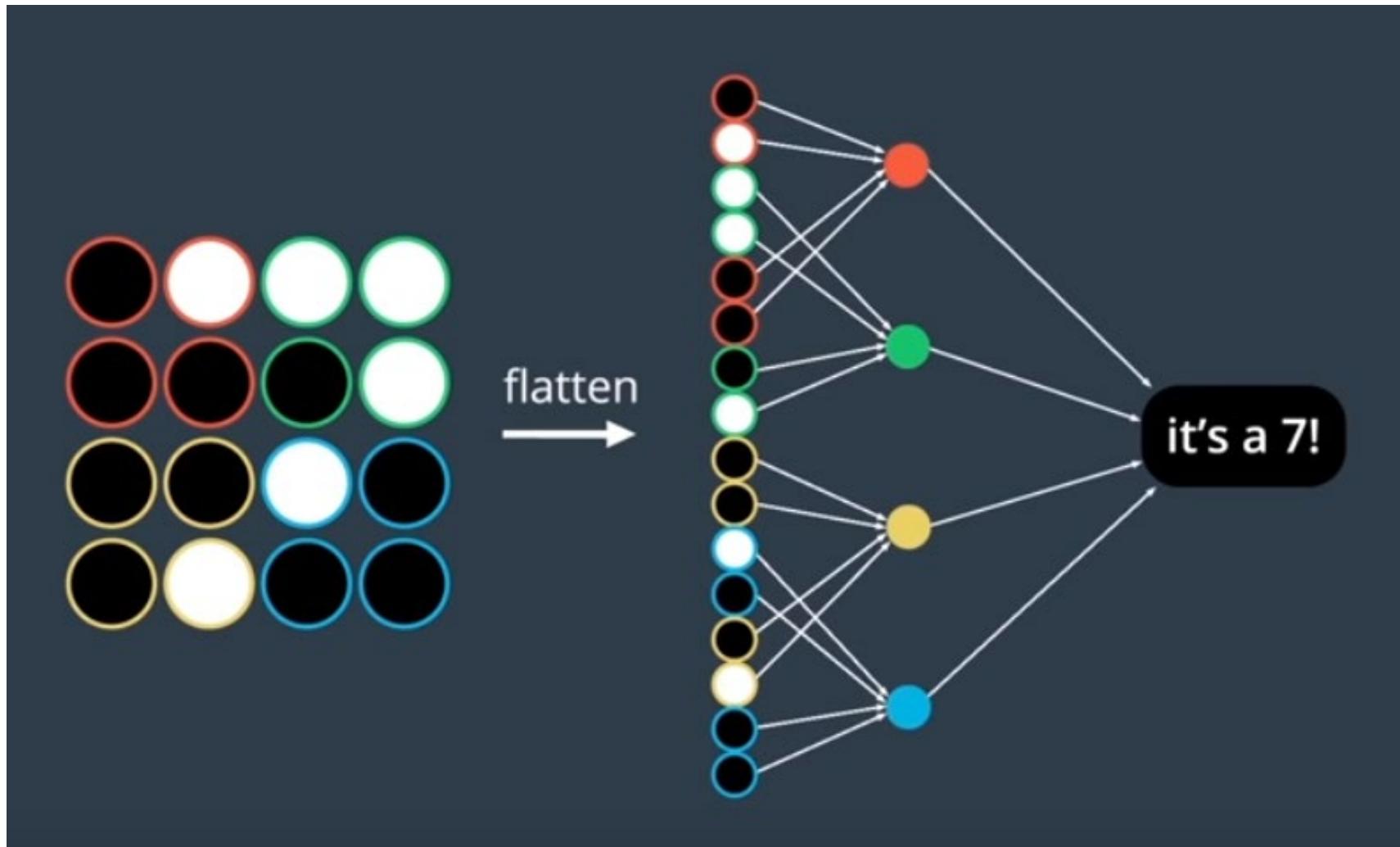


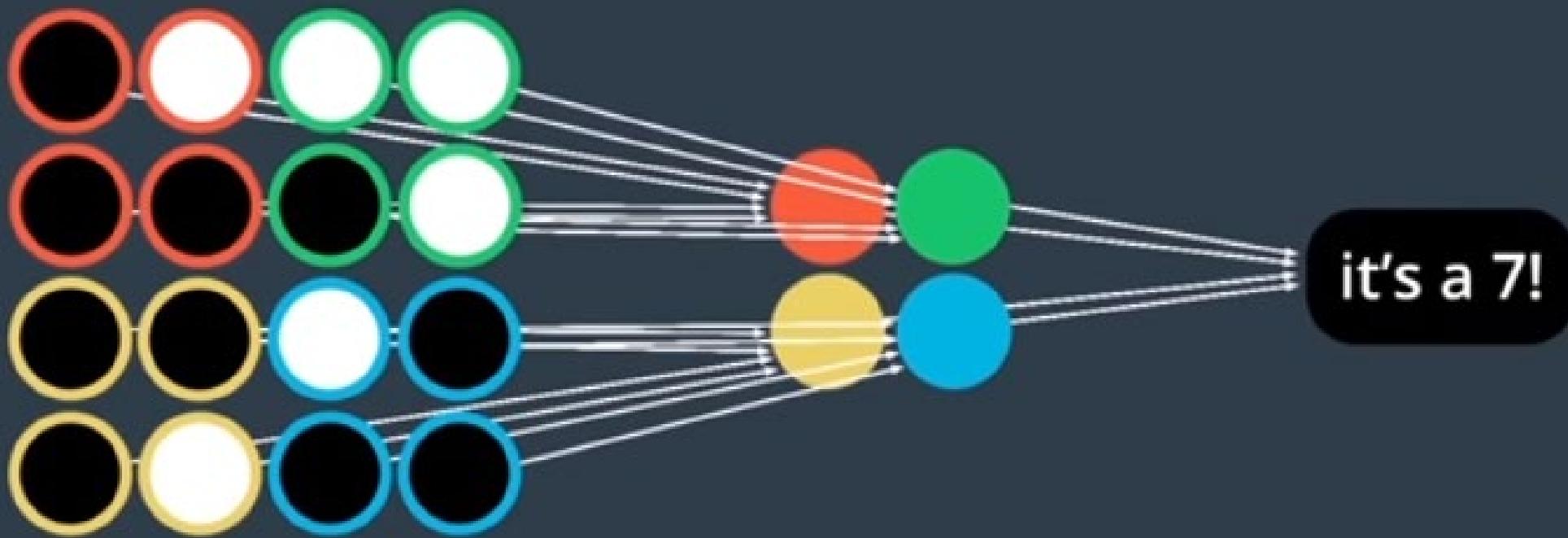
¿Todos los nodos intermedios deben estar conectados a todos los pixels de la imagen?

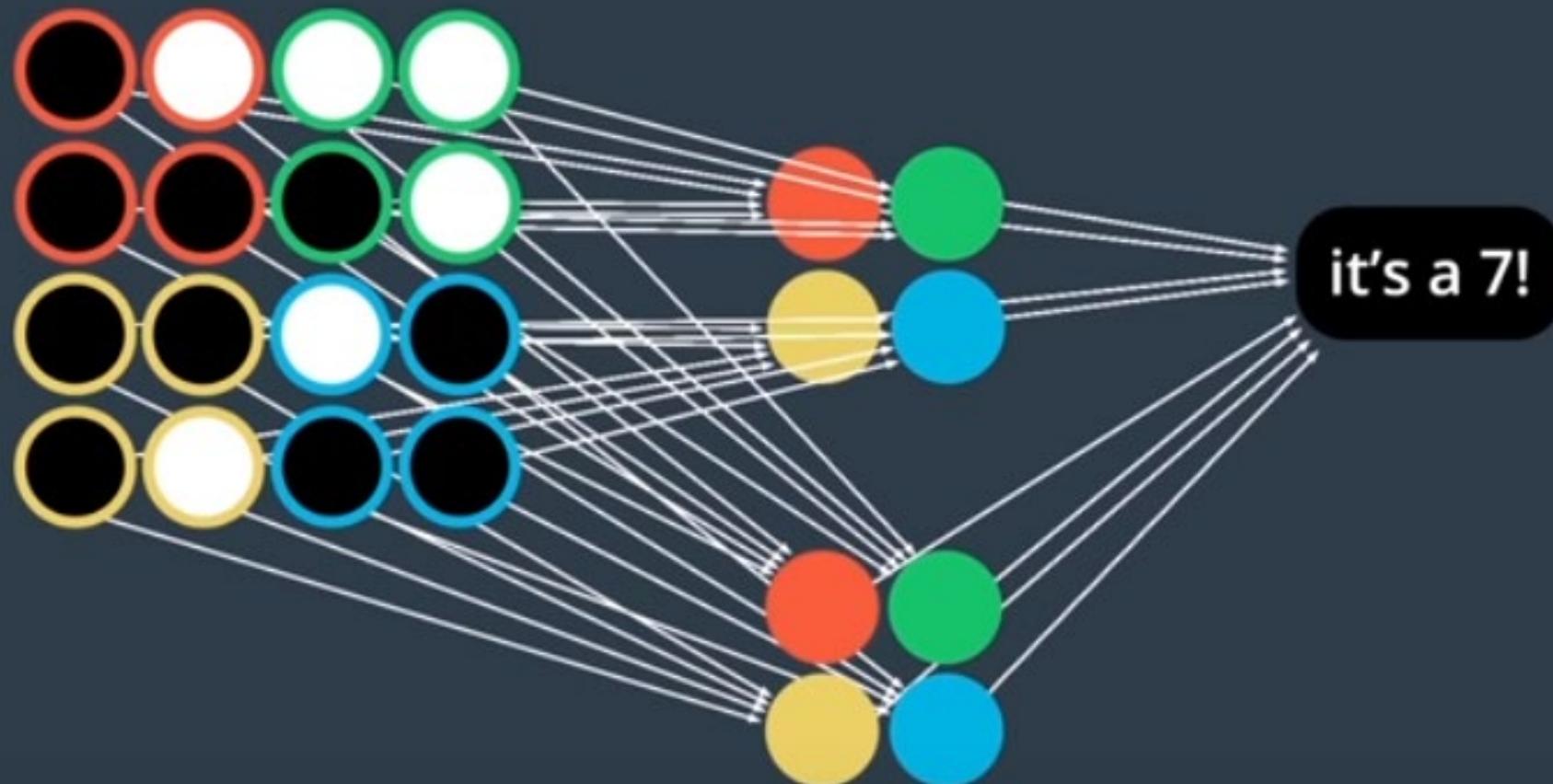




# Locally Connected Layer









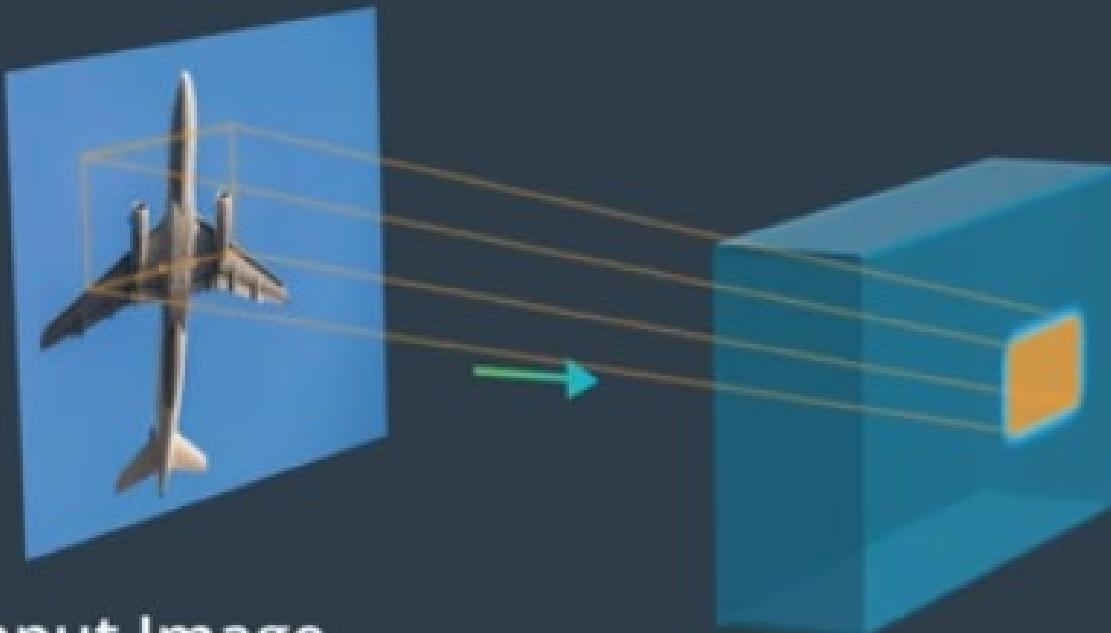
Cat

/04

## Filtros y la capa convolucional

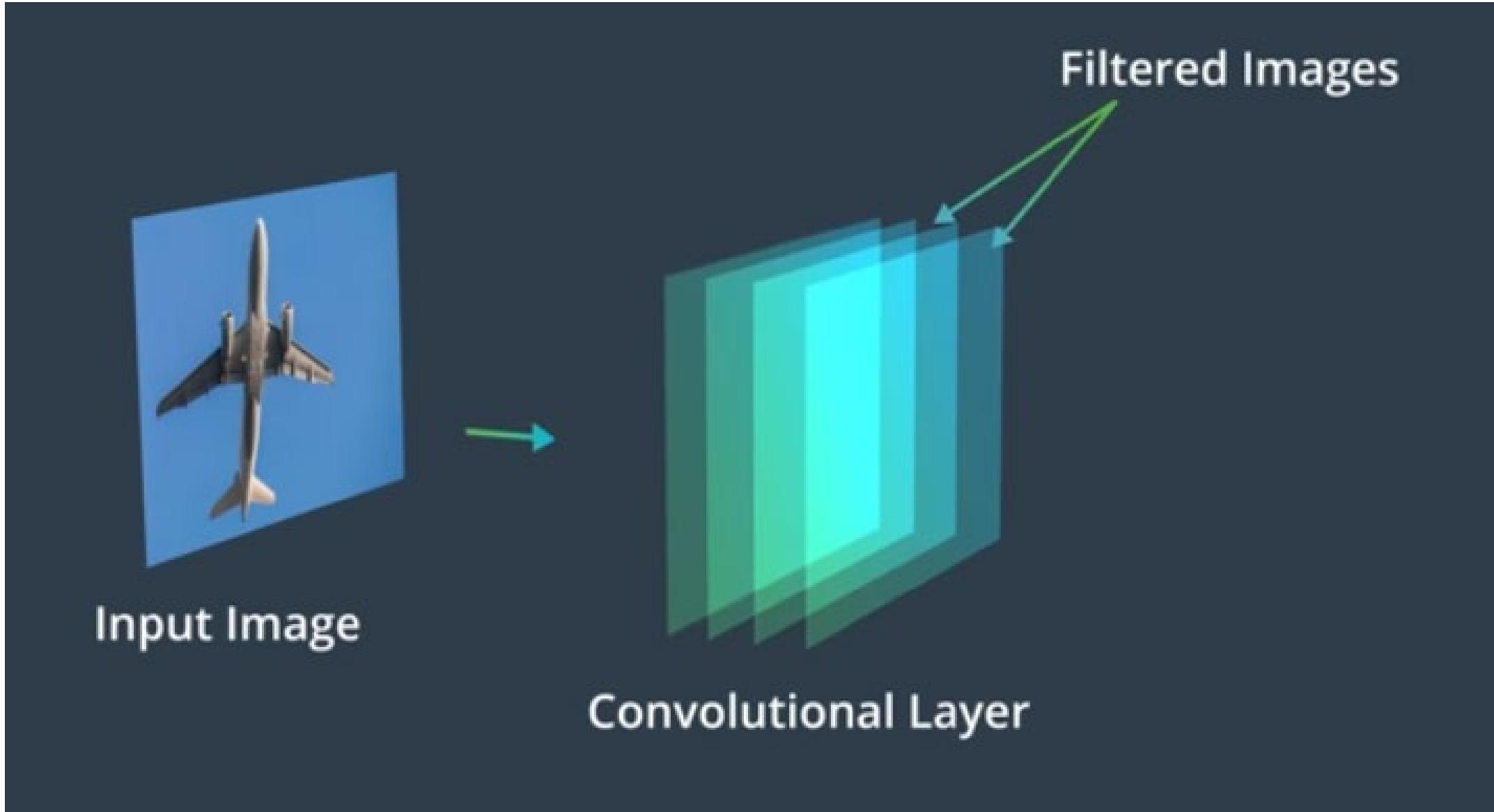


*Convolutional Kernels*



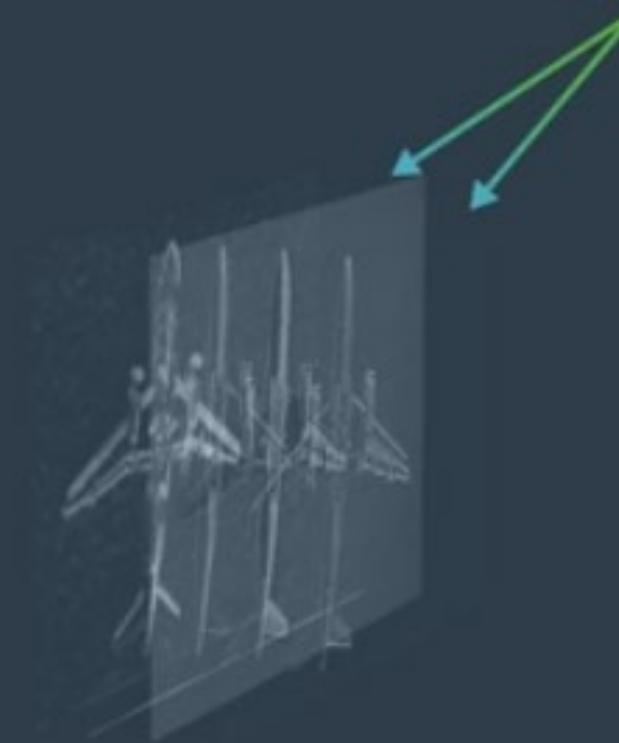
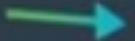
**Input Image**

**Convolutional Layer**





Input Image



Convolutional Layer

Filtered Images





Input

Neuron



Output

# Información Espacial



Cuando hablamos de información espacial hablamos de color o de forma.



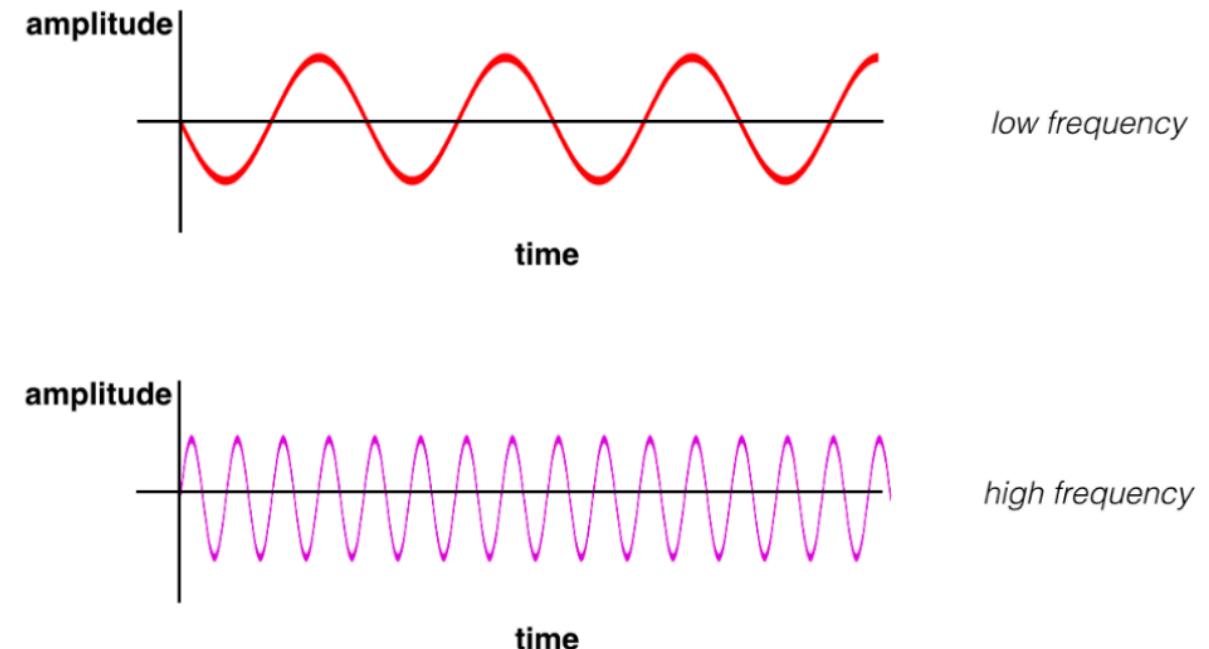
La forma puede también ser considerada como un patrón de intensidad en una imagen.



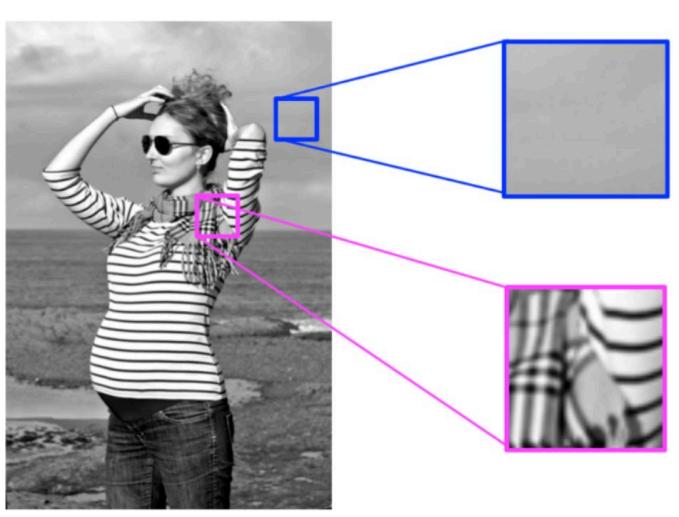
Intensidad es una medida de luz u oscuridad (igual que brillo) y podemos usar este conocimiento para detectar la forma de los objetos.

# Frecuencia en imágenes

- En sonidos, la alta frecuencia es un ruido agudo, como el canto de un pájaro o el violín. Y los sonidos de baja frecuencia son de tono bajo, como una voz grave o un bombo. Para el sonido, la frecuencia en realidad se refiere a qué tan rápido oscila una onda de sonido; Las oscilaciones generalmente se miden en ciclos/s (Hz).
- A continuación se muestran ejemplos de ondas sonoras de baja y alta frecuencia.



# Alta y baja frecuencia en imágenes



- La frecuencia en las imágenes es una tasa de cambio. Pero, ¿qué significa que una imagen cambie?
- Bueno, las imágenes cambian en el espacio, y una imagen de alta frecuencia es aquella en la que la intensidad cambia mucho. Y el nivel de brillo cambia rápidamente de un píxel al siguiente. Una imagen de baja frecuencia puede ser una que tenga un brillo relativamente uniforme o que cambie muy lentamente.
- La mayoría de las imágenes tienen componentes tanto de alta frecuencia como de baja frecuencia.
- Los componentes de alta frecuencia también corresponden a los bordes de los objetos en las imágenes, lo que puede ayudarnos a clasificar esos objetos.

/05

## Filtros Paso Alto



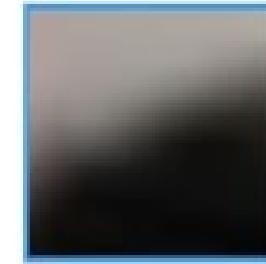
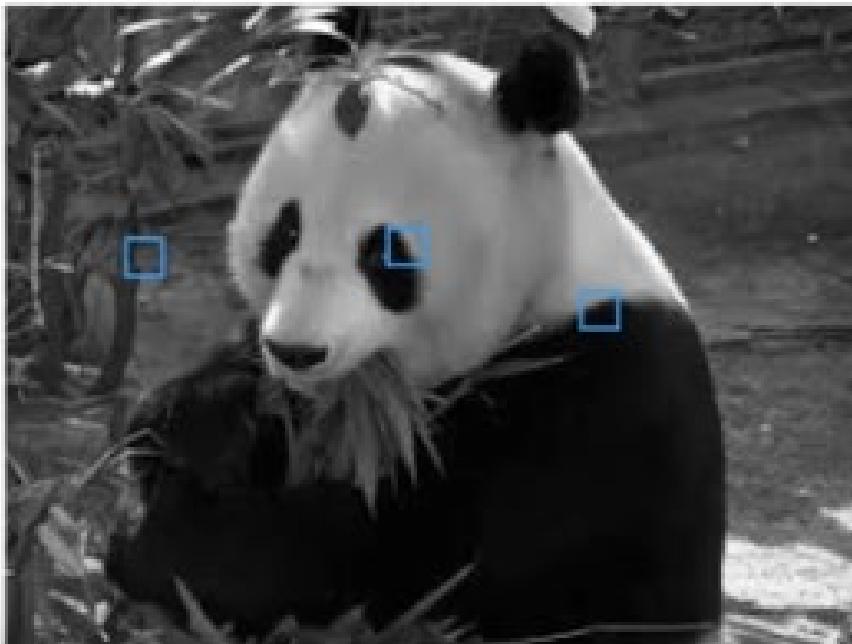
# FILTERS



1. Filter out unwanted information
2. Amplify features of interest

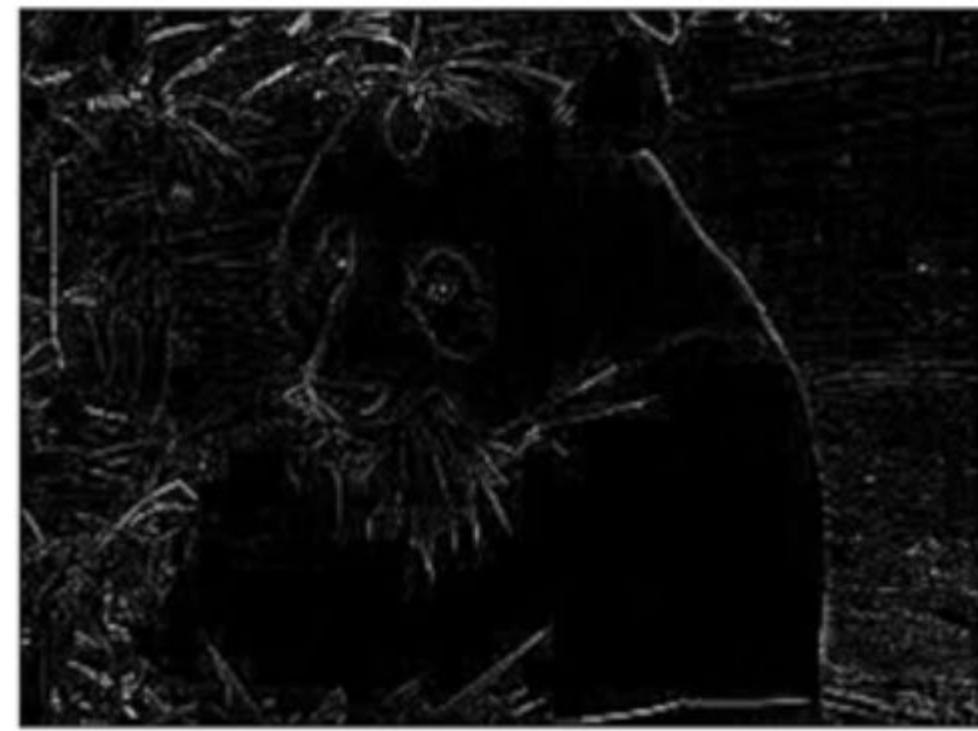
# HIGH-PASS FILTERS

- Sharpen an image
- Enhance ***high-frequency*** parts of an image



# EDGE DETECTION

Emphasize Edges



**Edges** are areas in an image where the intensity changes very quickly, and they often indicate object boundaries

# CONVOLUTION KERNELS

0	-1	0
-1	4	-1
0	-1	0

edge detection filter

$$0 + -1 + 0 + -1 + 4 + -1 + 0 + -1 + 0 = \mathbf{0}$$

# CONVOLUTION KERNELS

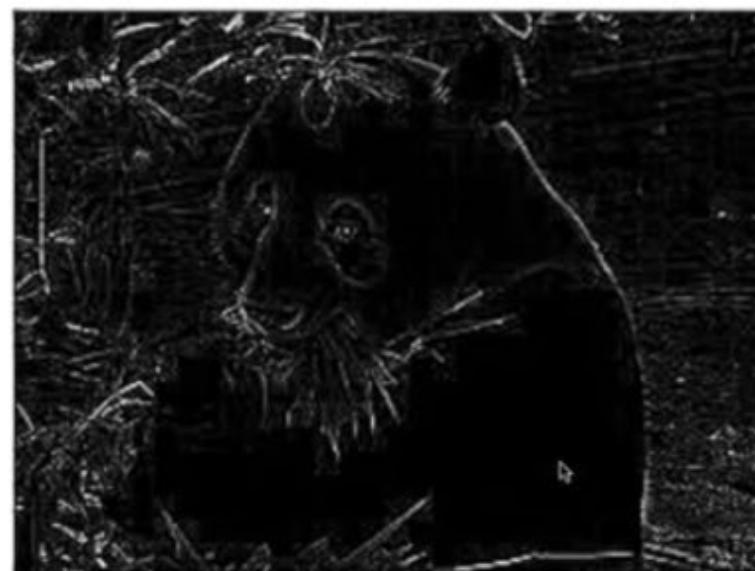
0	-1	0
-1	4	<b>-2</b>
0	-1	0



$$0 + -1 + 0 + -1 + \mathbf{8} + -1 + 0 + -1 + 0 = 1$$

# CONVOLUTION KERNELS

0	-1	0
-1	4	<b>-2</b>
0	-1	0



$$0 + -1 + 0 + -1 + 4 + \boxed{-2} + 0 + -1 + 0 = \boxed{-1}$$

# CONVOLUTION

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

\*



K

F(x,y)

K \* F(x,y) = output image

# CONVOLUTION



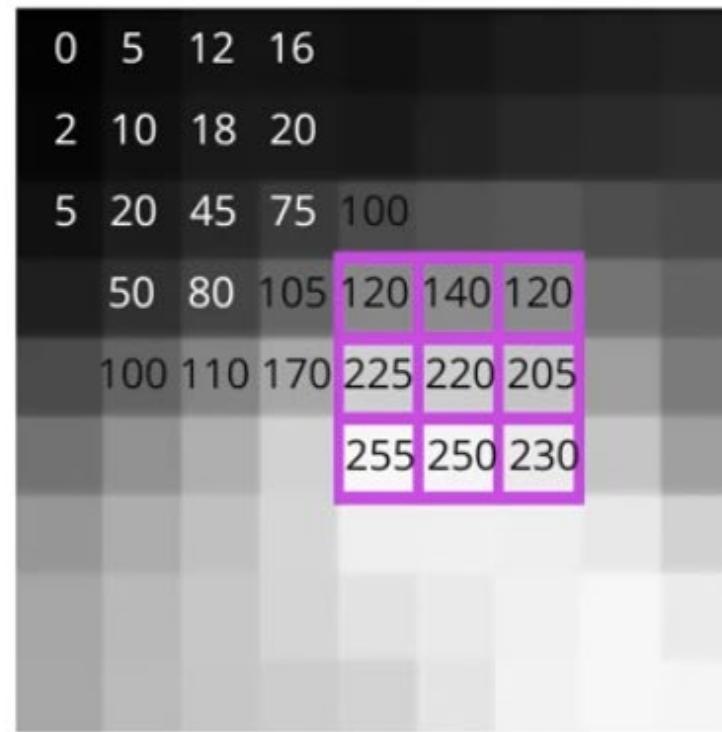
$K * F(x,y) = \text{output image}$

# CONVOLUTION

0	-1	0
-1	4	-1
0	-1	0

0	-140	0
-225	880	-205
0	-250	0

= 60



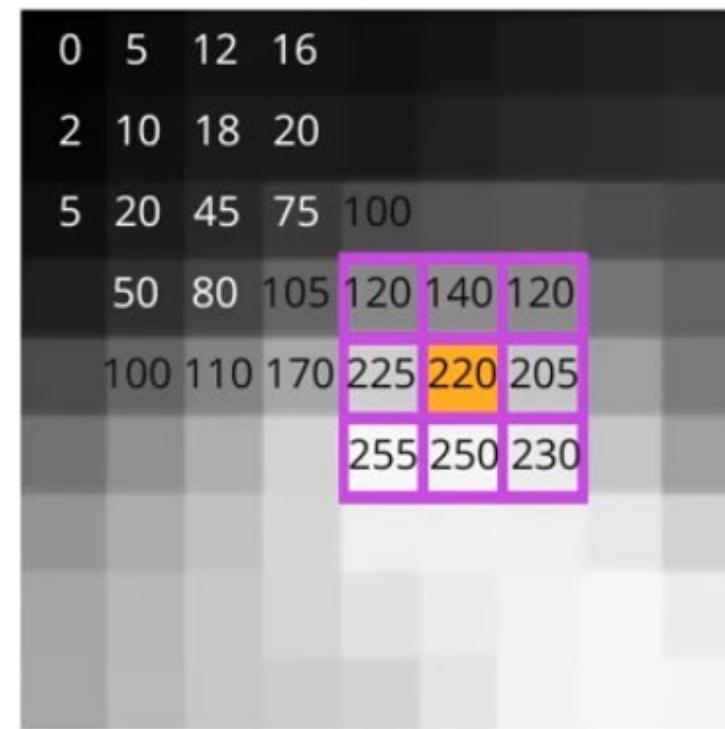
# CONVOLUTION

**Weights**

0	-1	0
-1	4	-1
0	-1	0

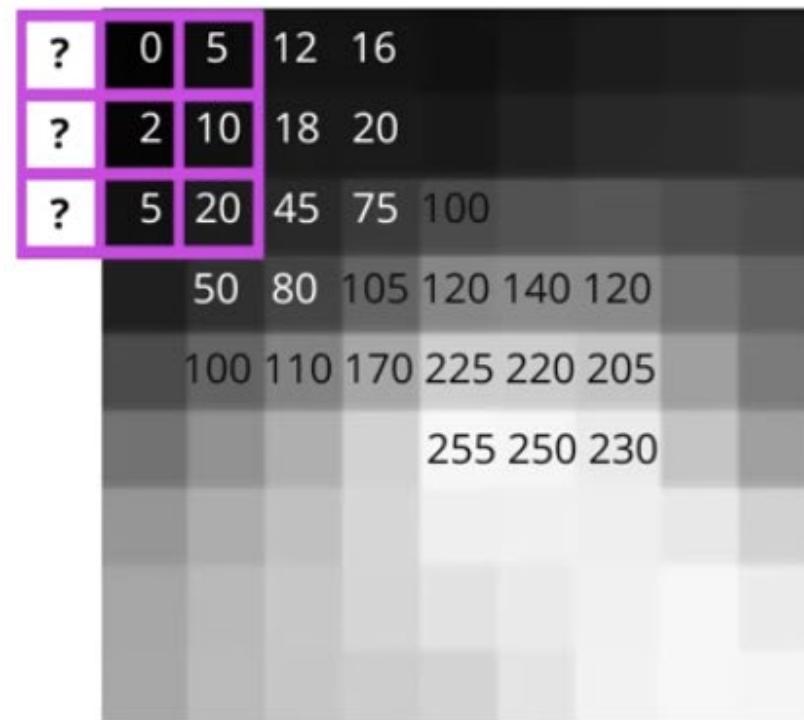
  

0	-140	0
-225	880	-205
0	-250	0

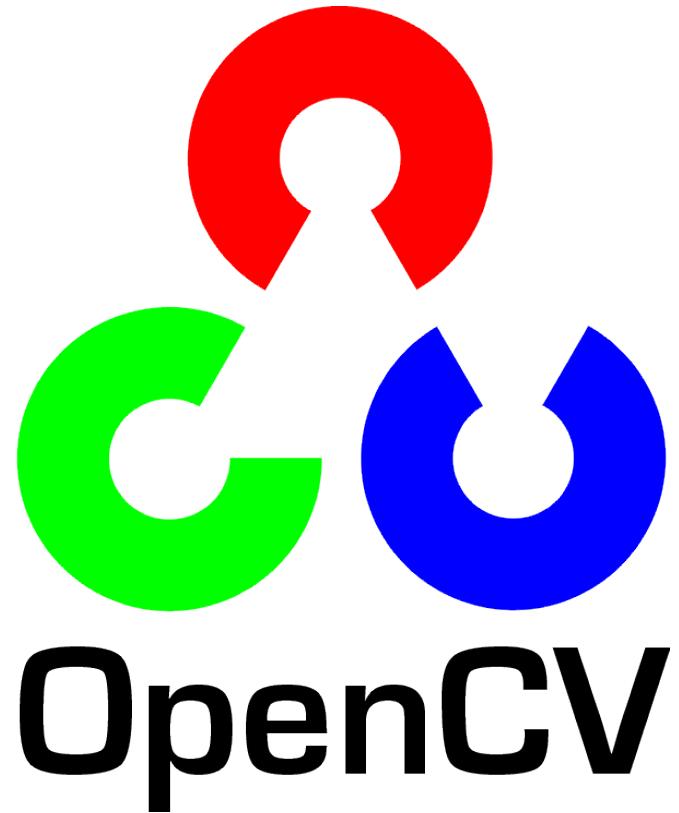


= **60** Pixel value in the output image

0	-1	0
-1	4	-1
0	-1	0



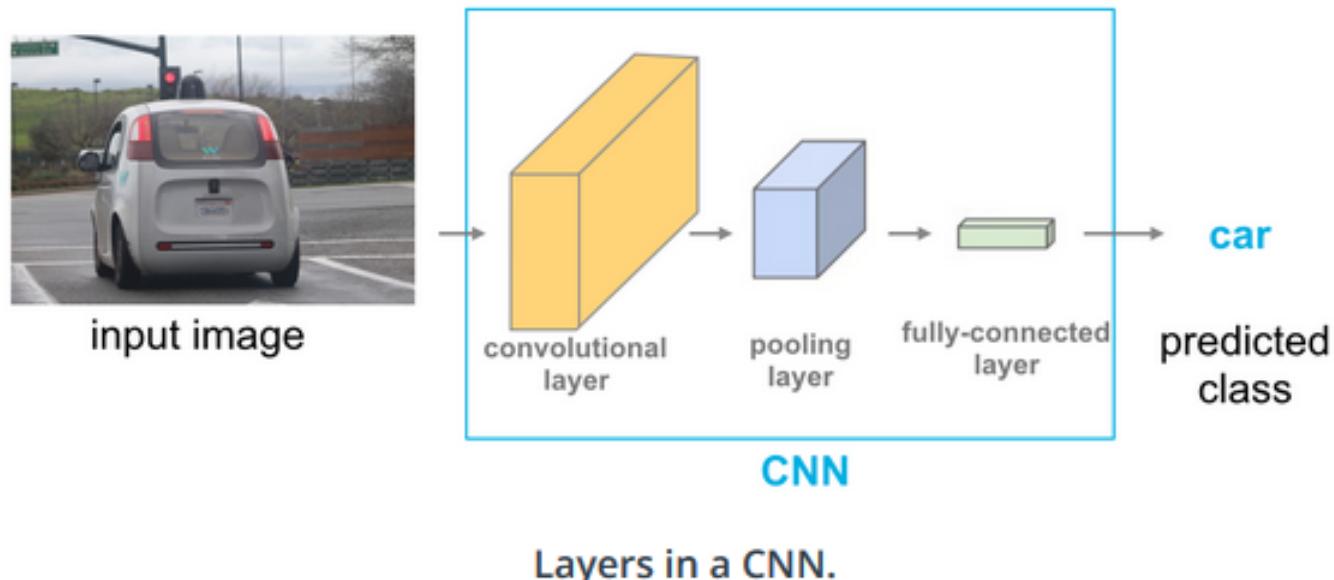
- Antes de comenzar a codificar nuestros propios kernels/filtros convolucionales, veremos una nueva biblioteca que será útil para manejar tareas de visión por computadora, como la clasificación de imágenes.
- OpenCV es una biblioteca de software de visión artificial y aprendizaje automático que incluye muchos algoritmos comunes de análisis de imágenes que nos ayudarán a crear aplicaciones de visión por computadora inteligentes y personalizadas. Para empezar, ¡esto incluye herramientas que nos ayudan a procesar imágenes y seleccionar áreas de interés! La biblioteca es ampliamente utilizada en aplicaciones académicas e industriales; desde su sitio, OpenCV incluye una impresionante lista de usuarios: "Junto con compañías bien establecidas como Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota que emplean la biblioteca, hay muchas nuevas empresas como Applied Minds, VideoSurf y Zeitera, que hacen un uso extensivo de OpenCV".



# Custom\_filters notebook

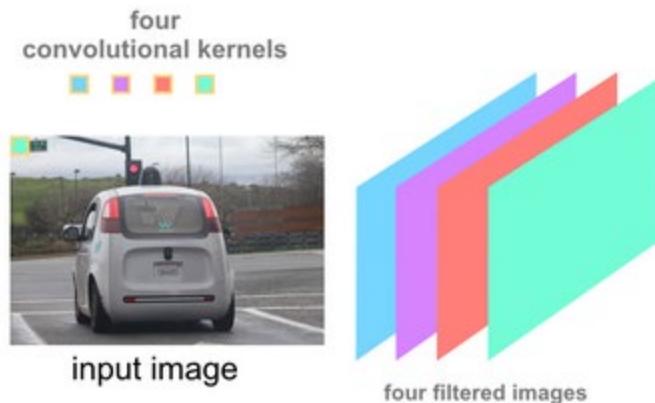
## The Importance of Filters

What you've just learned about different types of filters will be really important as you progress through this course, especially when you get to Convolutional Neural Networks (CNNs). CNNs are a kind of deep learning model that can learn to do things like image classification and object recognition. They keep track of spatial information and *learn* to extract features like the edges of objects in something called a **convolutional layer**. Below you'll see a simple CNN structure, made of multiple layers, below, including this "convolutional layer".



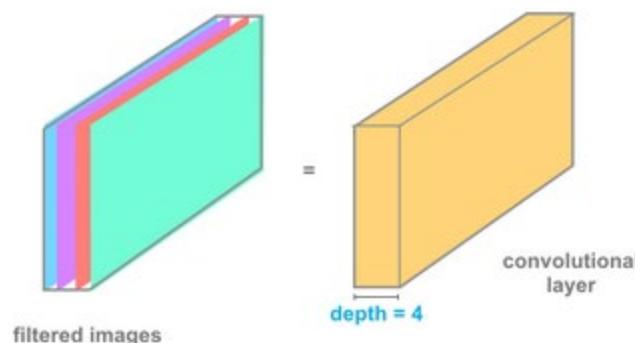
## Convolutional Layer

The convolutional layer is produced by applying a series of many different image filters, also known as convolutional kernels, to an input image.



**4 kernels = 4 filtered images.**

In the example shown, 4 different filters produce 4 differently filtered output images. When we stack these images, we form a complete convolutional layer with a depth of 4!



A convolutional layer.

/06

## Capa Convolucional



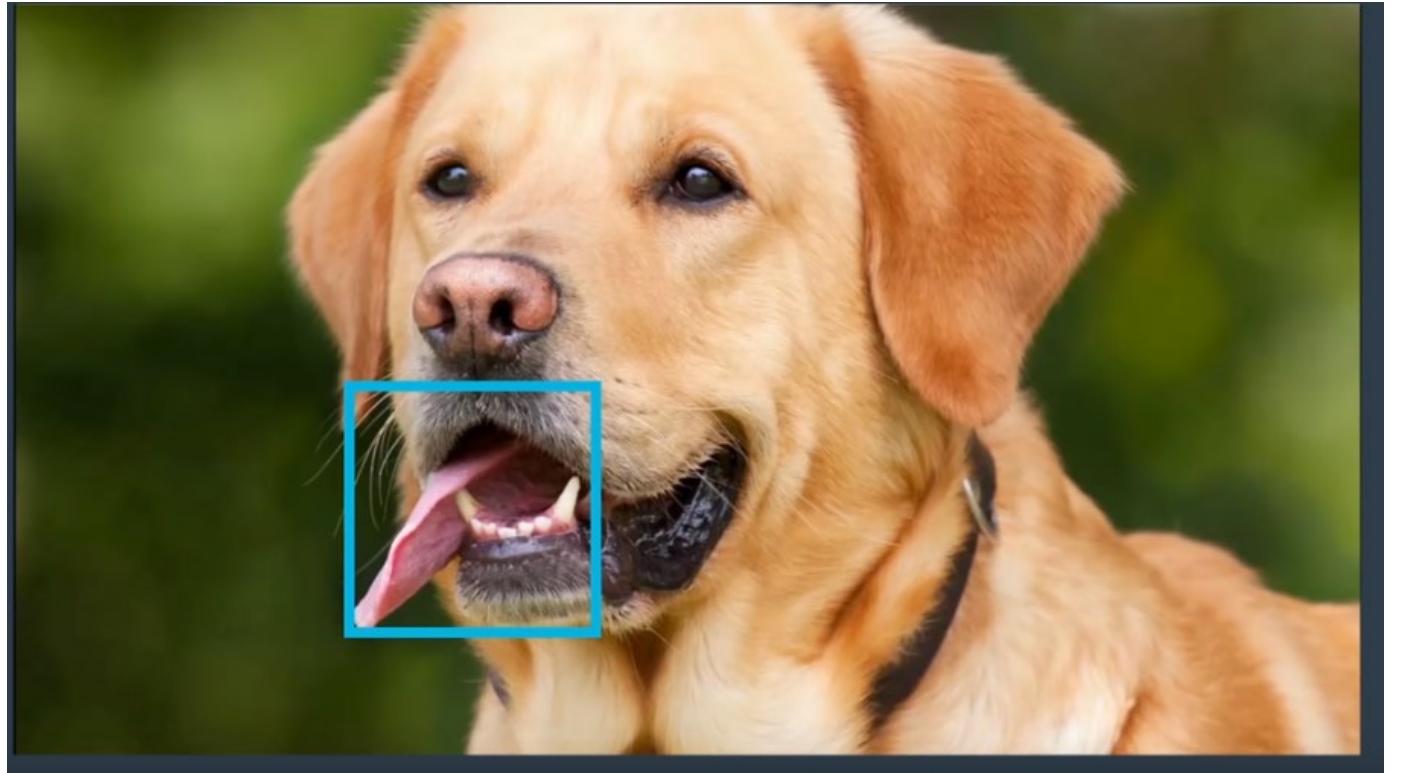
# Aprendizaje

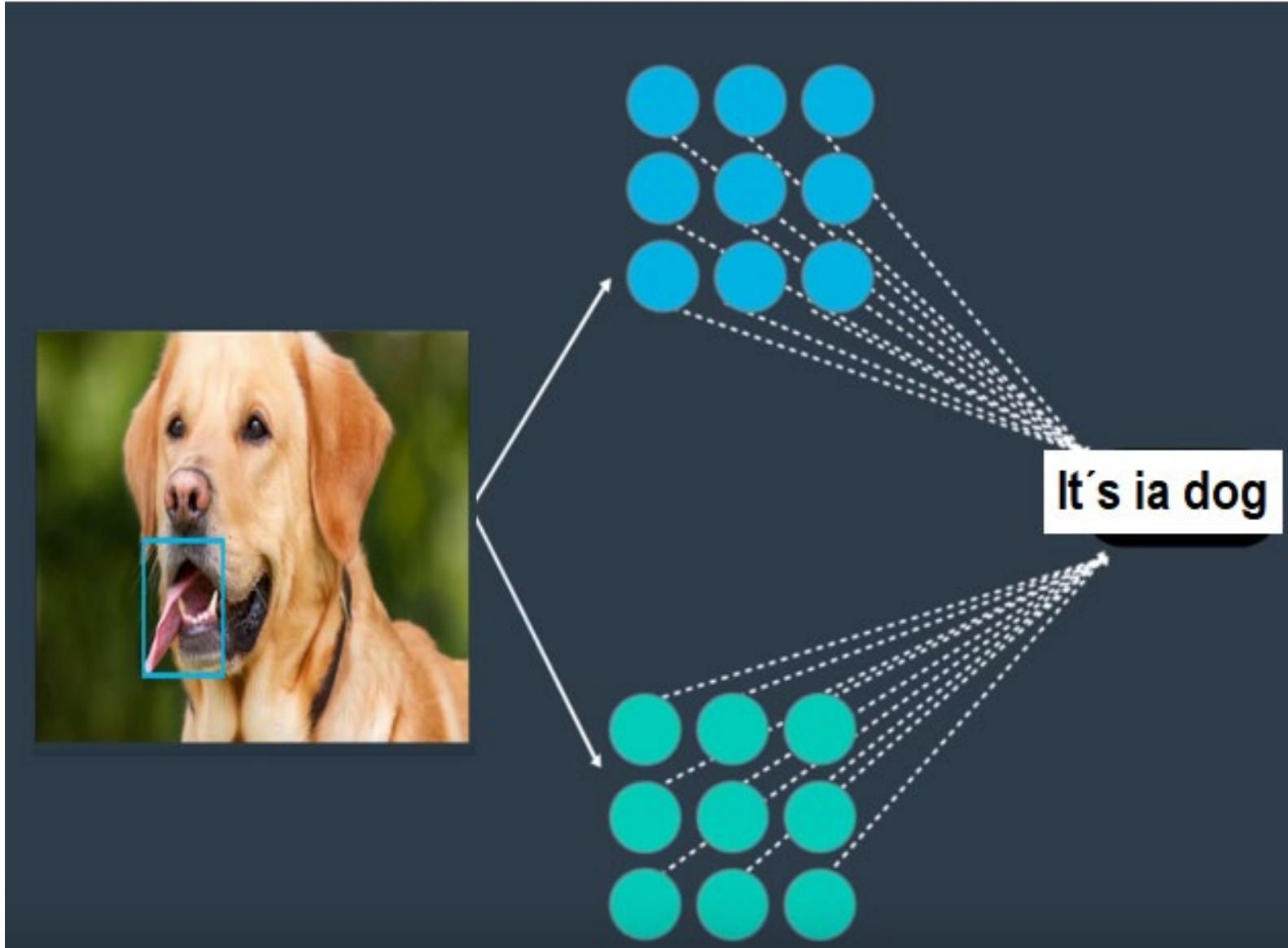
Hemos establecido los valores de los pesos de filtro explícitamente, pero las redes neuronales realmente aprenderán los mejores pesos de filtro a medida que entrenan en un conjunto de datos de imagen.

Recordar que los filtros de paso alto y paso bajo son los que definen el comportamiento de una red como esta, ¡y sabemos cómo codificarlos desde cero!

En la práctica, encontraremos que muchas redes neuronales aprenden a detectar los bordes de las imágenes porque los bordes del objeto contienen información valiosa sobre la forma de un objeto.

Esta imagen tiene muchas características que queremos detectar. Esta zona tiene dientes, lengua, labios, etc. Para entender la imagen, necesitamos filtros para detectar estas características.





## Input Layer



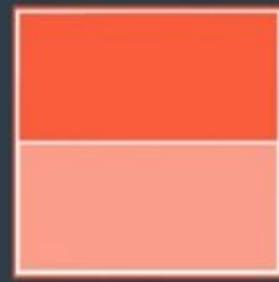
Filter 1



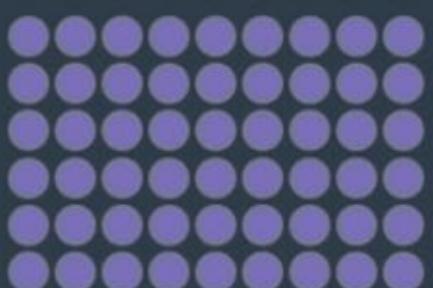
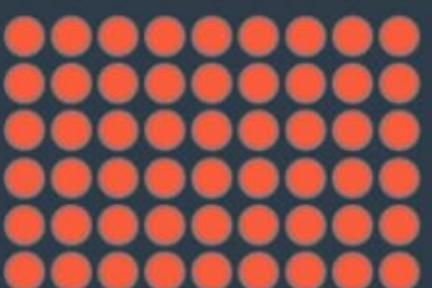
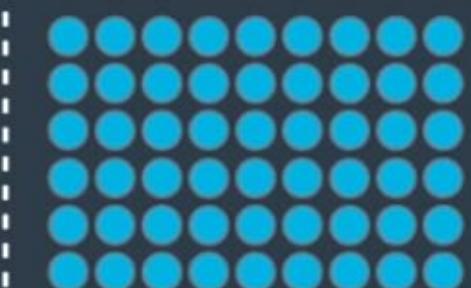
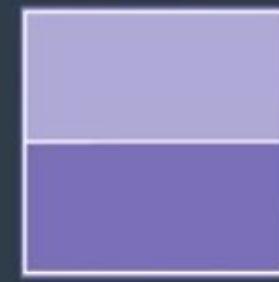
Filter 2



Filter 3



Filter 4



Convolutional Layer

## Input Layer



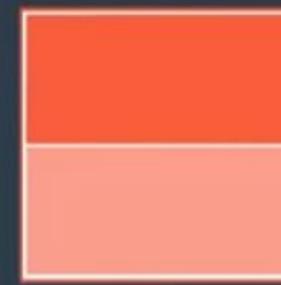
Filter 1



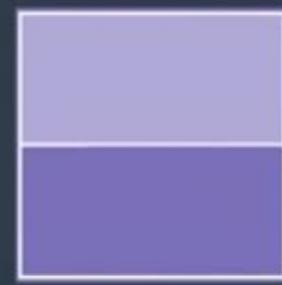
Filter 2



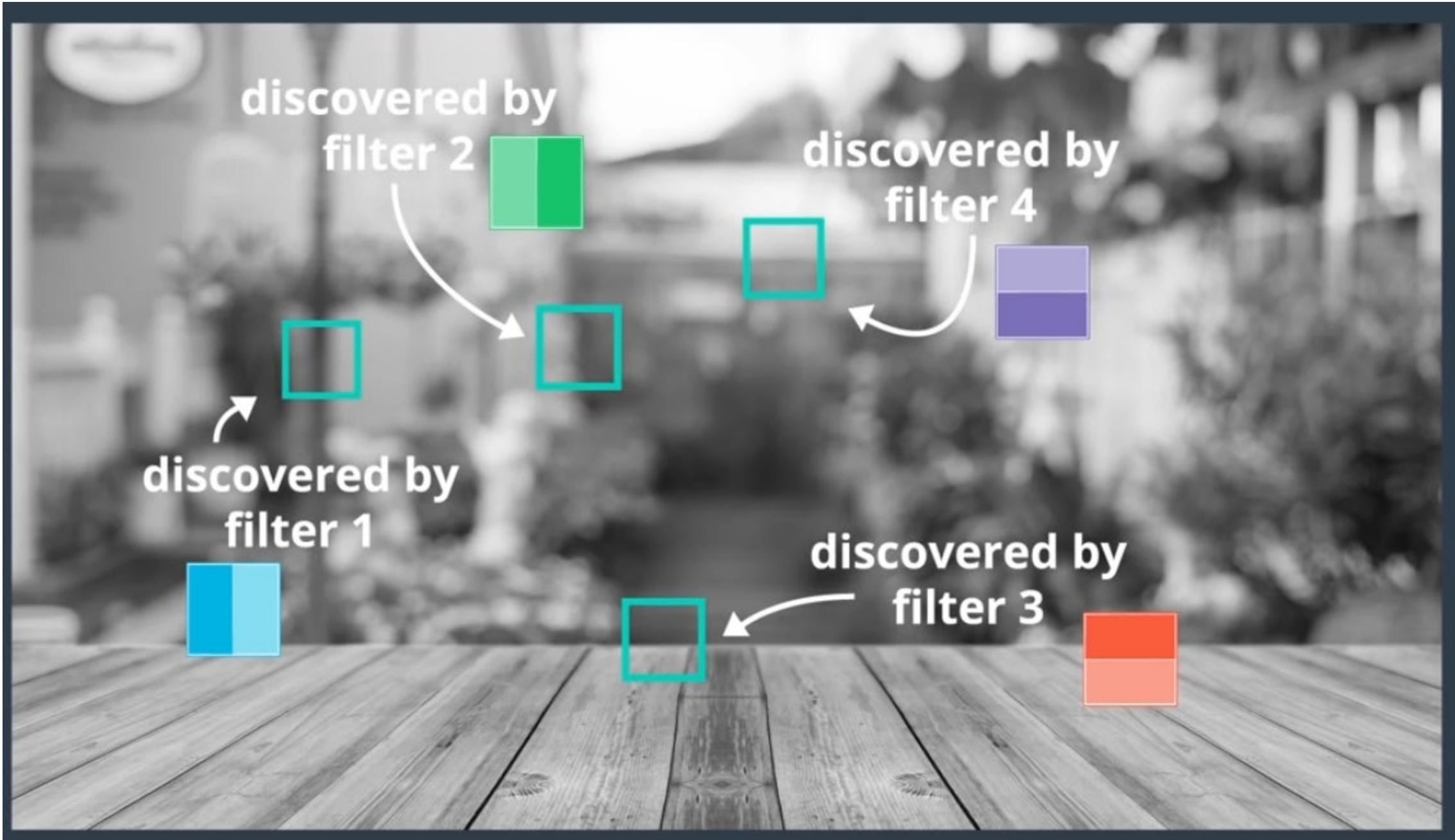
Filter 3



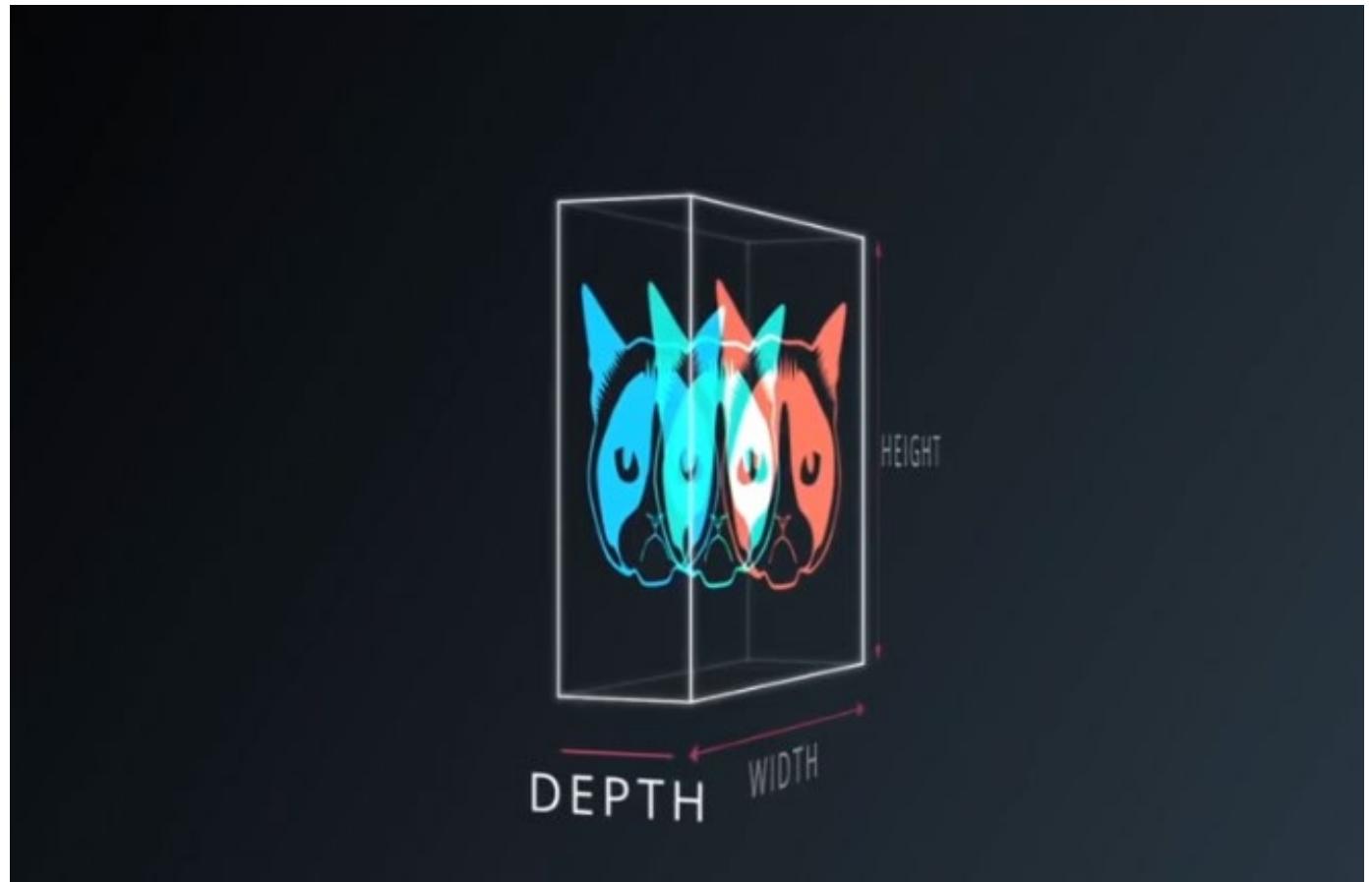
Filter 4

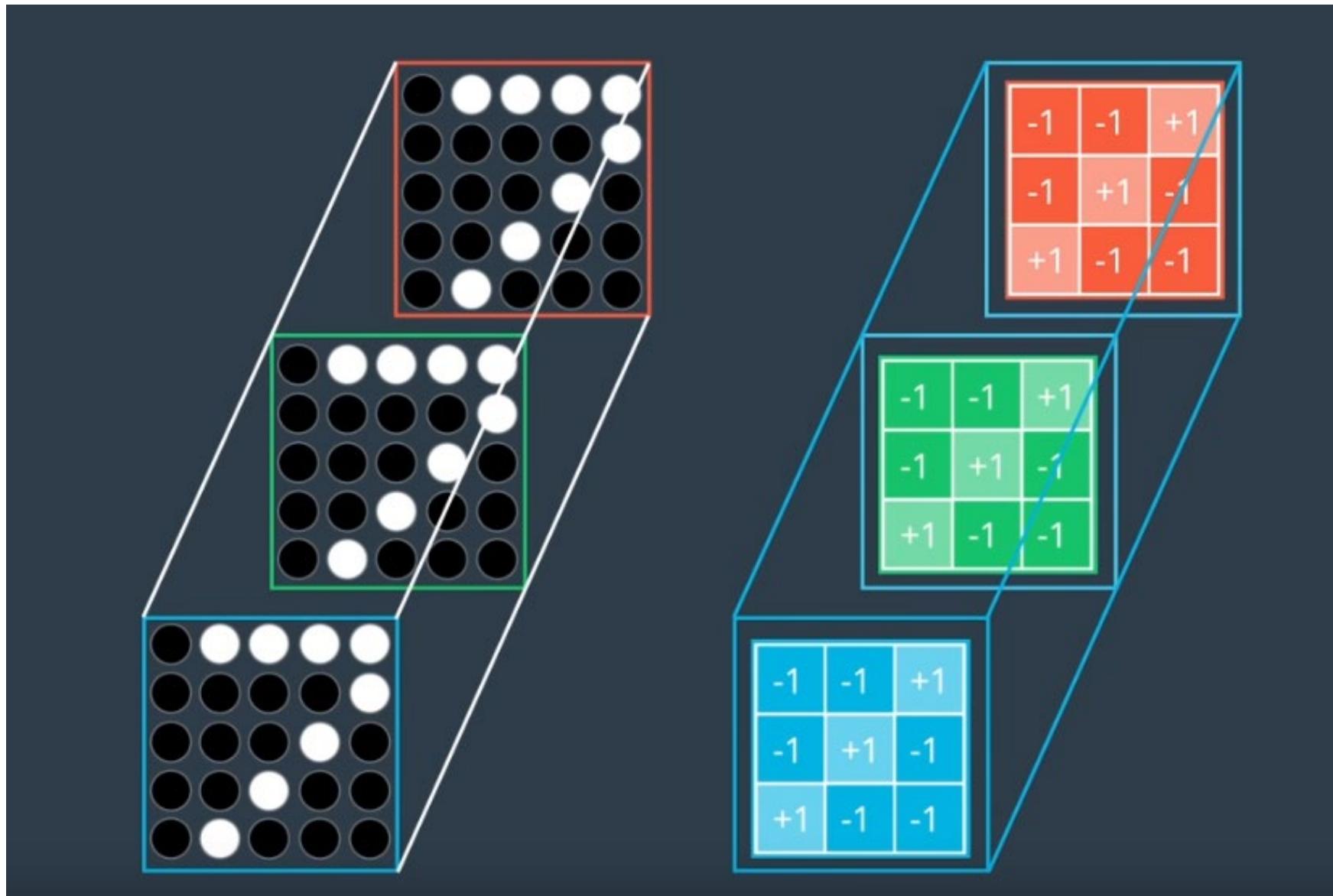


## Convolutional Layer



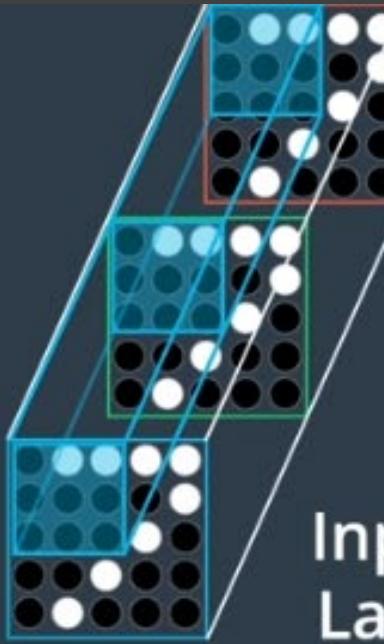
# Imágenes de color







Filter

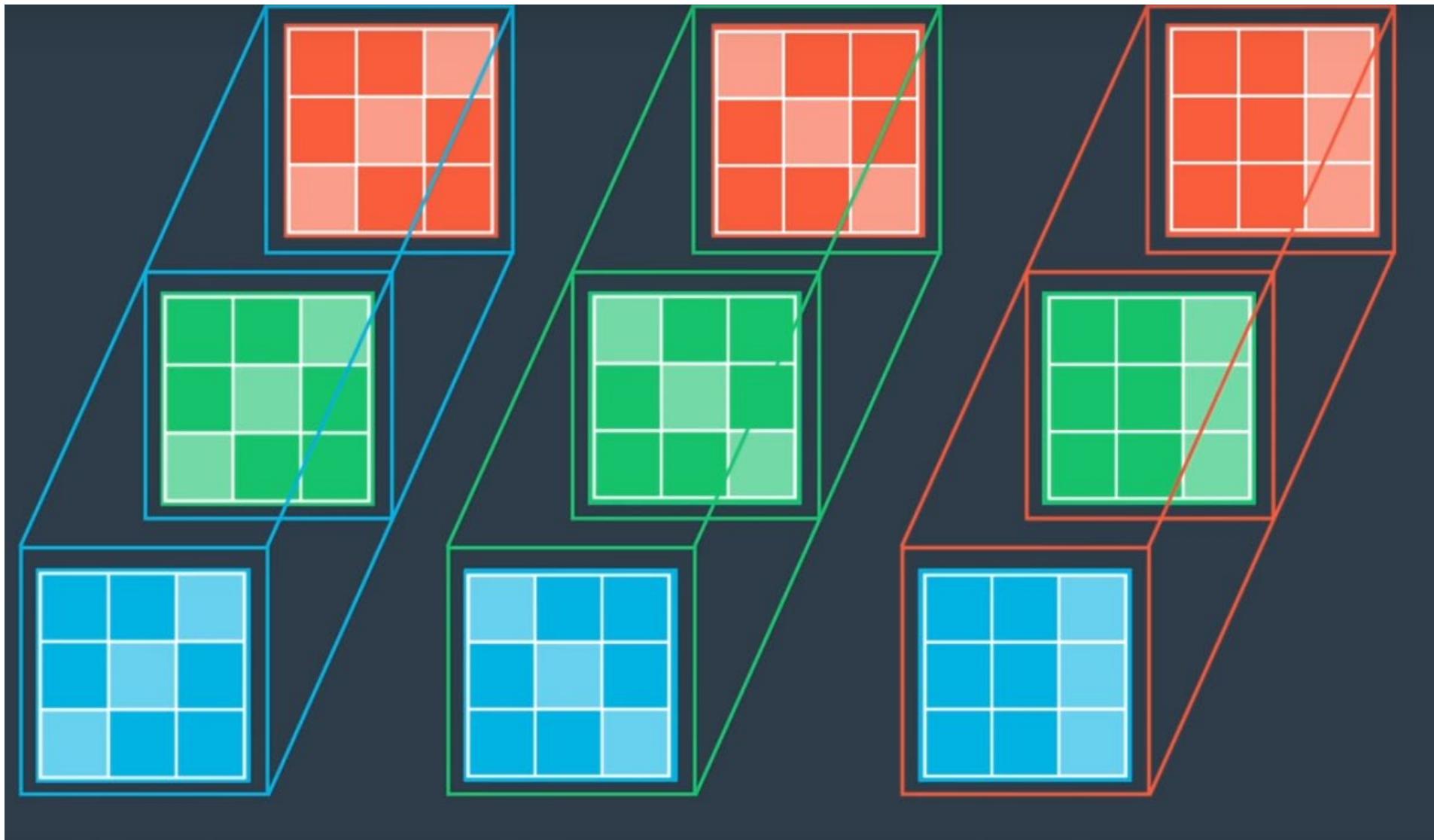


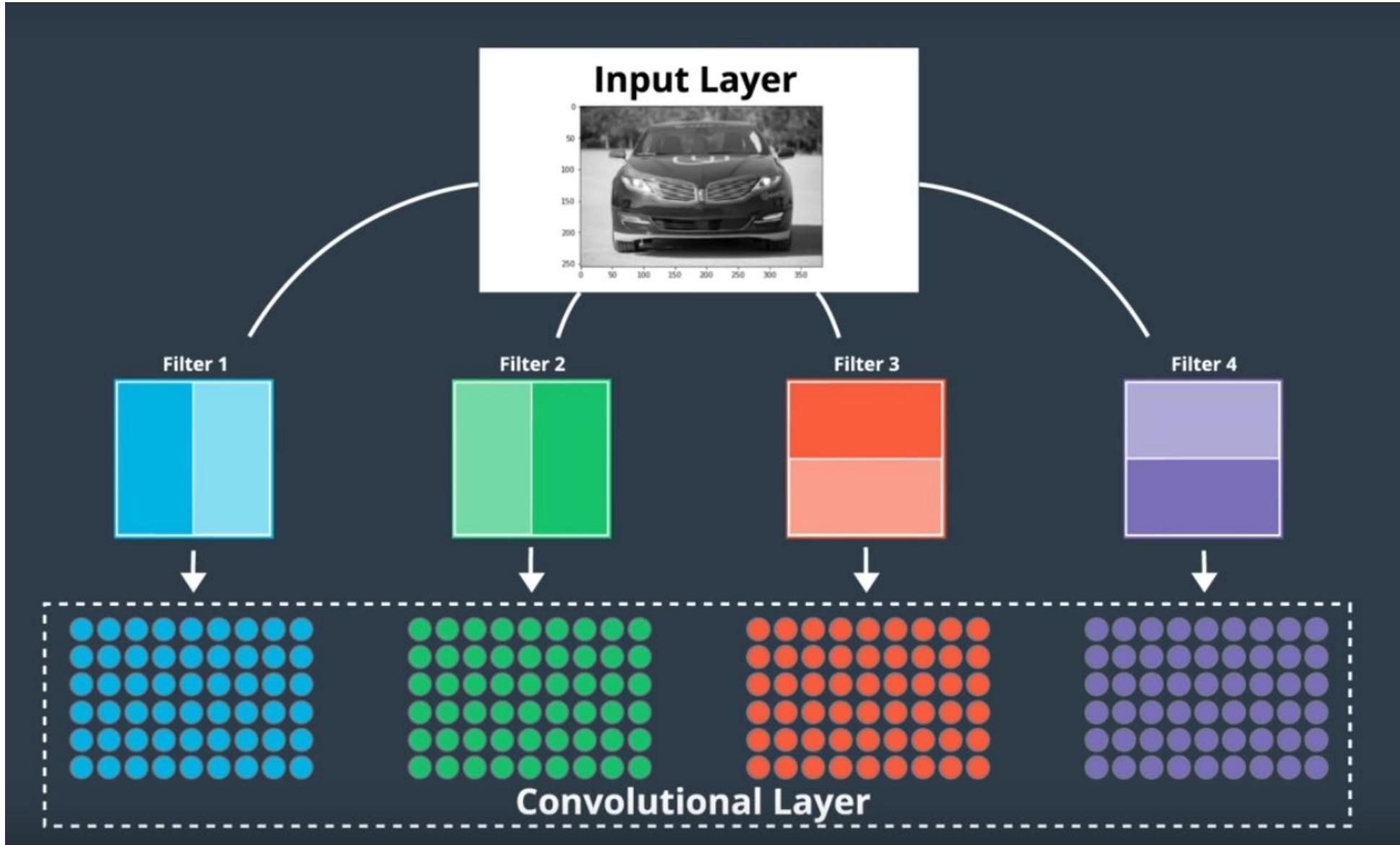
Input Layer

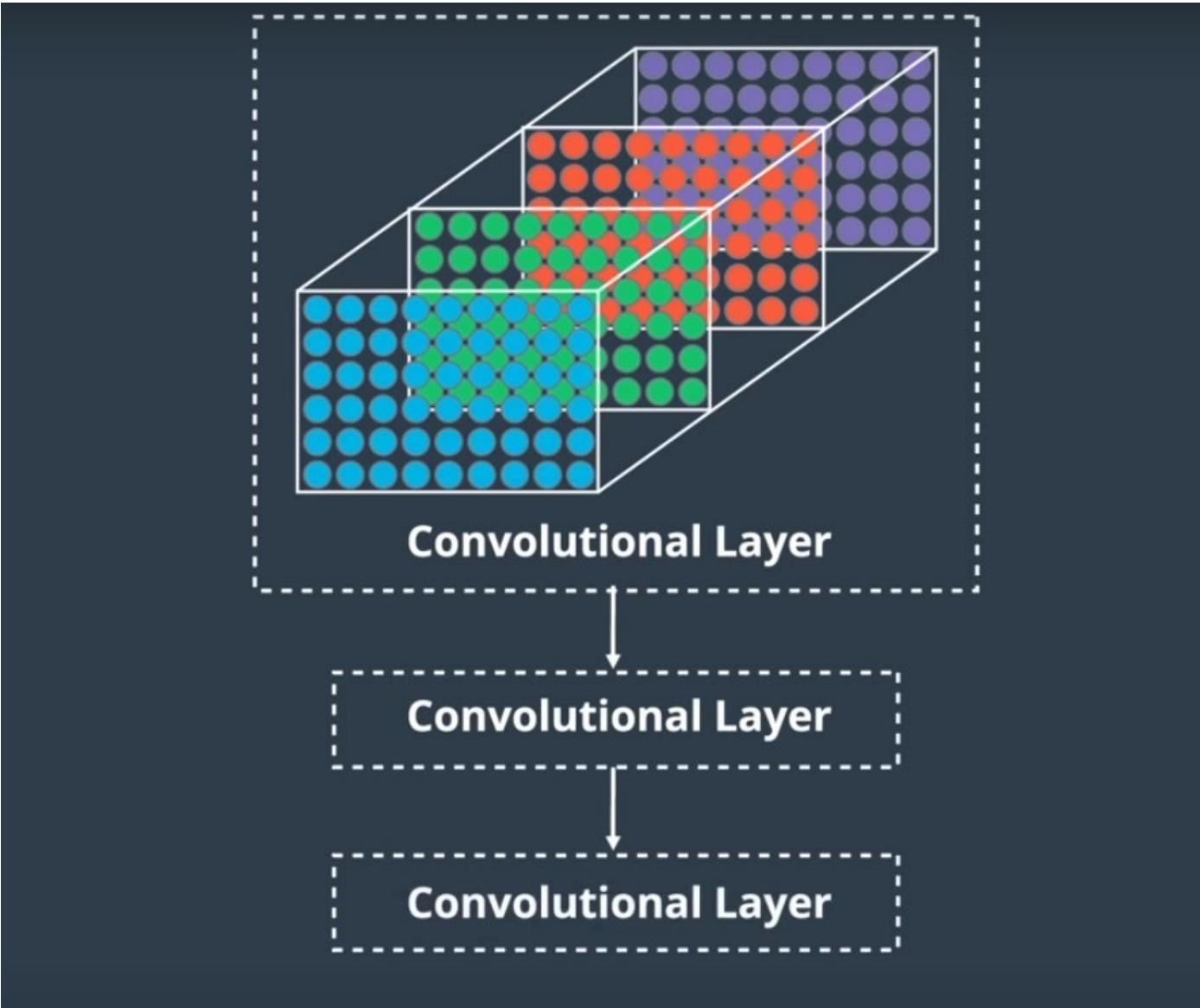


Convolutional Layer

$$\text{ReLU} \left( \text{SUM} \left( \begin{array}{ccccccccc} \text{blue} \times \text{black} & \text{blue} \times \text{white} & \text{light blue} \times \text{white} & \text{green} \times \text{black} & \text{green} \times \text{white} & \text{light green} \times \text{white} & \text{orange} \times \text{black} & \text{orange} \times \text{white} & \text{light orange} \times \text{white} \\ \text{blue} \times \text{black} & \text{light blue} \times \text{black} & \text{blue} \times \text{black} & \text{green} \times \text{black} & \text{light green} \times \text{black} & \text{green} \times \text{black} & \text{orange} \times \text{black} & \text{light orange} \times \text{black} & \text{orange} \times \text{black} \\ \text{light blue} \times \text{black} & \text{blue} \times \text{black} & \text{blue} \times \text{black} & \text{green} \times \text{black} & \text{light green} \times \text{black} & \text{green} \times \text{black} & \text{orange} \times \text{black} & \text{light orange} \times \text{black} & \text{orange} \times \text{black} \end{array} \right) \right)$$

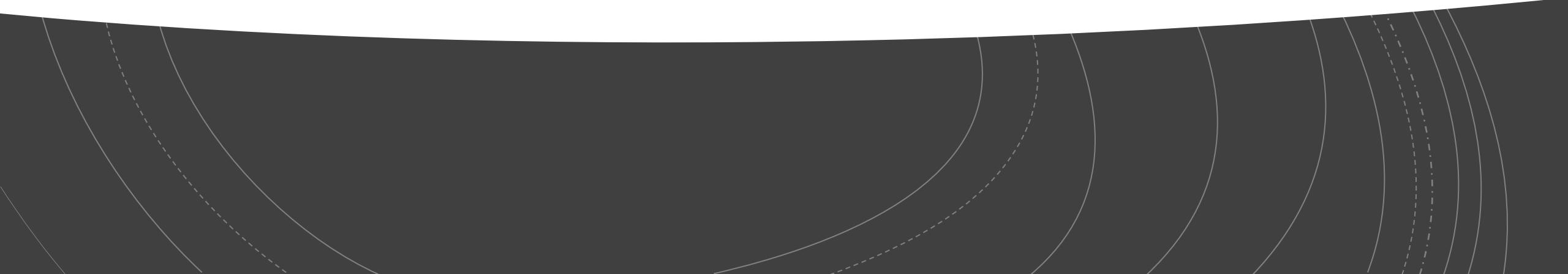






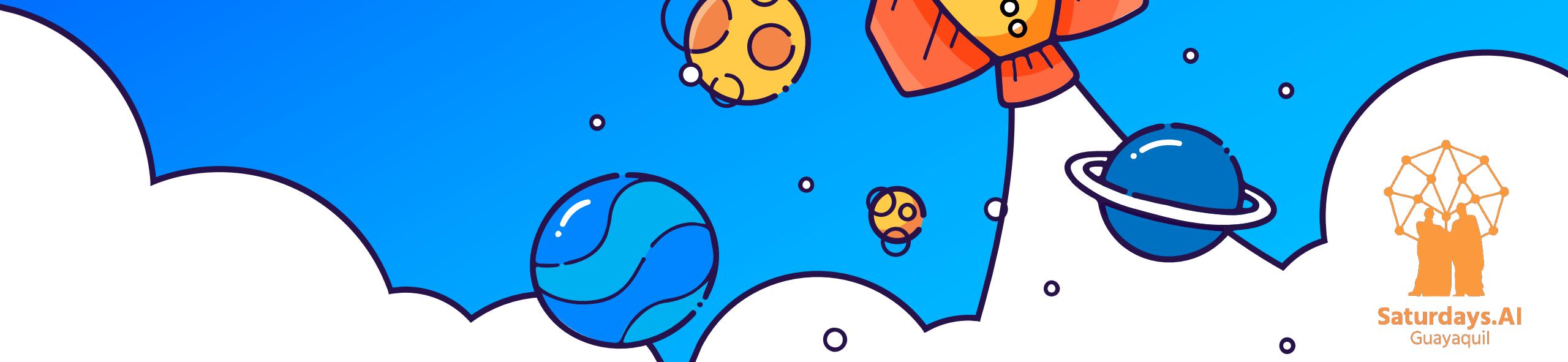
<http://setosa.io/ev/image-kernels/>

# Conv<sub>\_</sub>visualization notebook

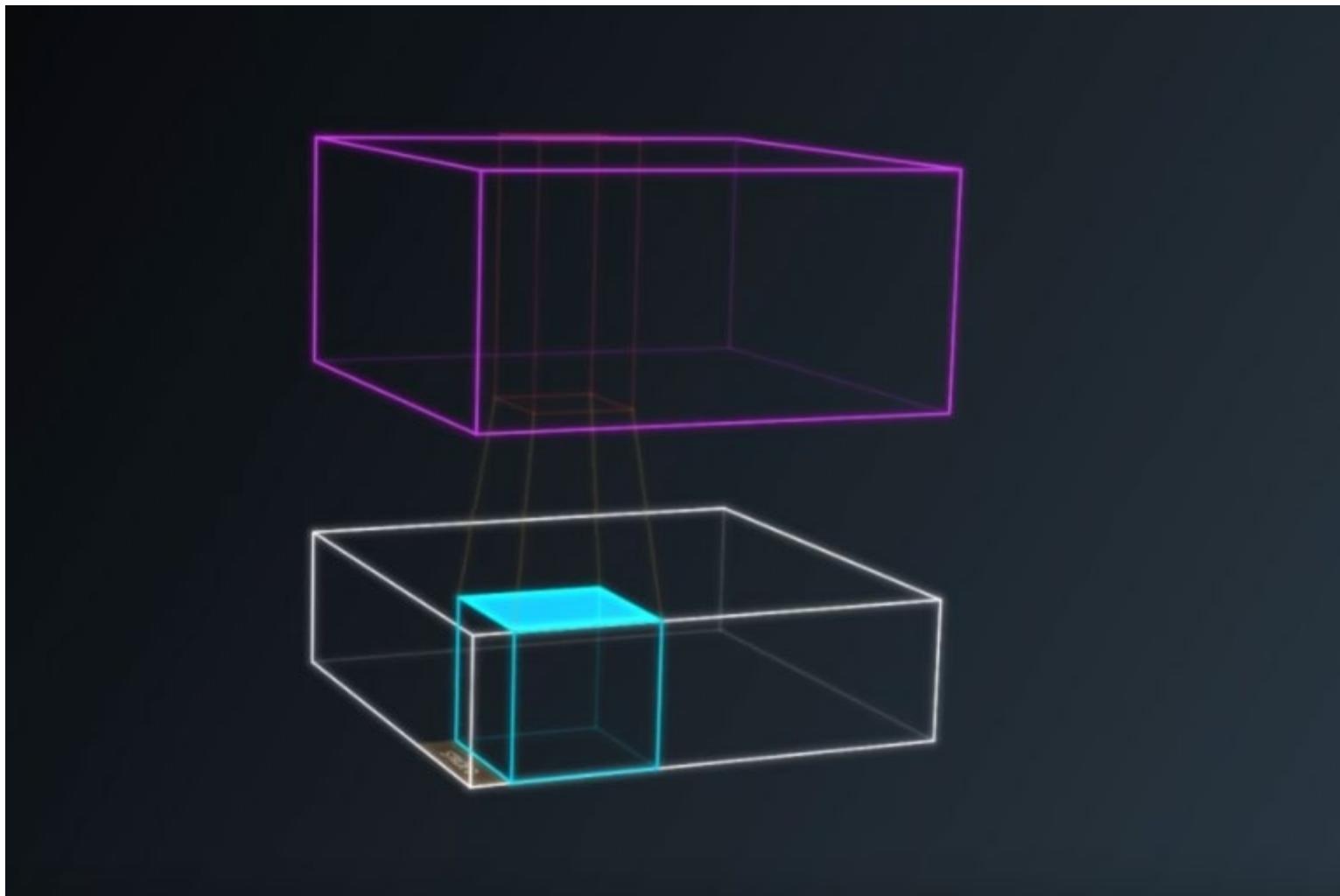


/07

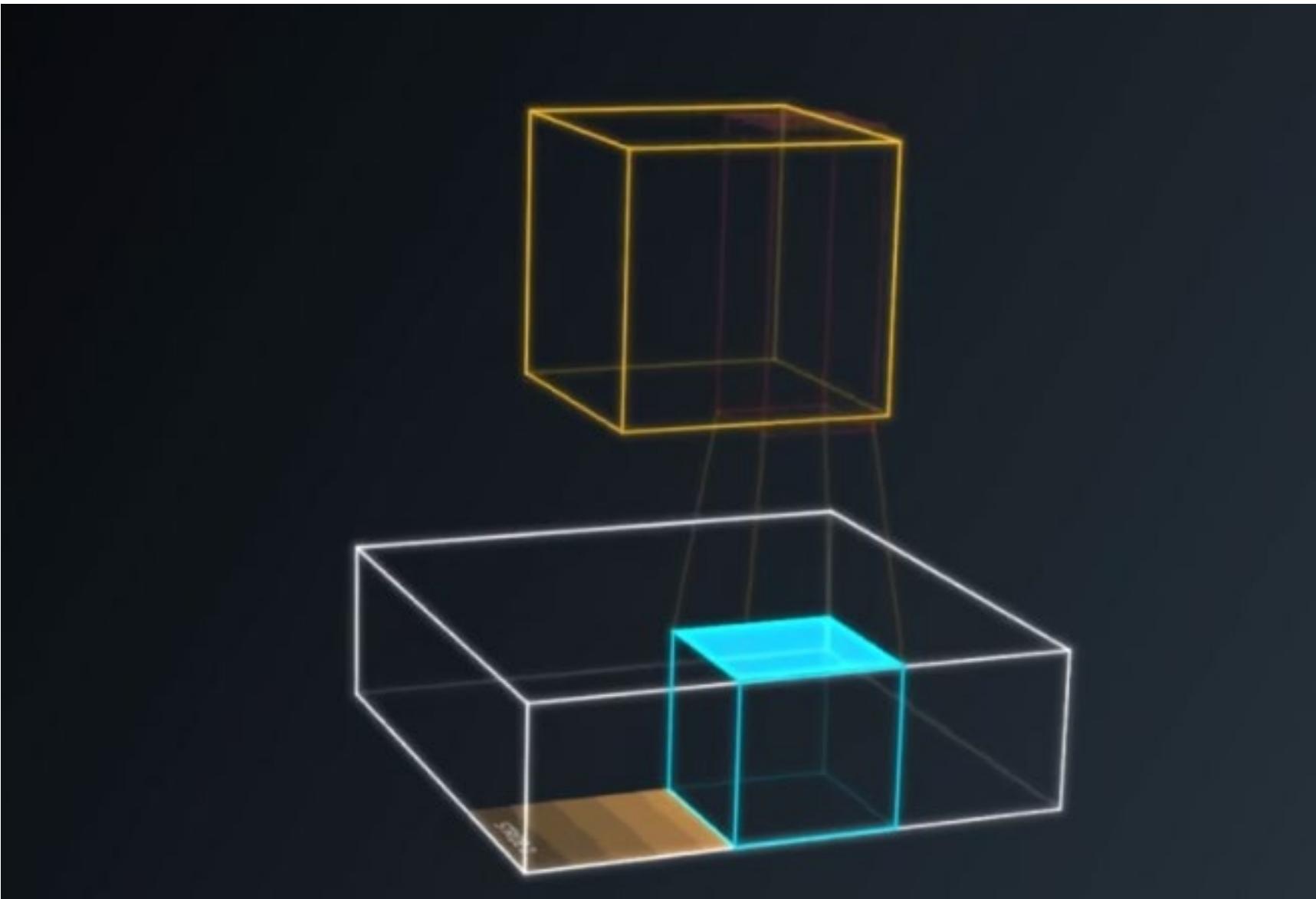
Stride y padding



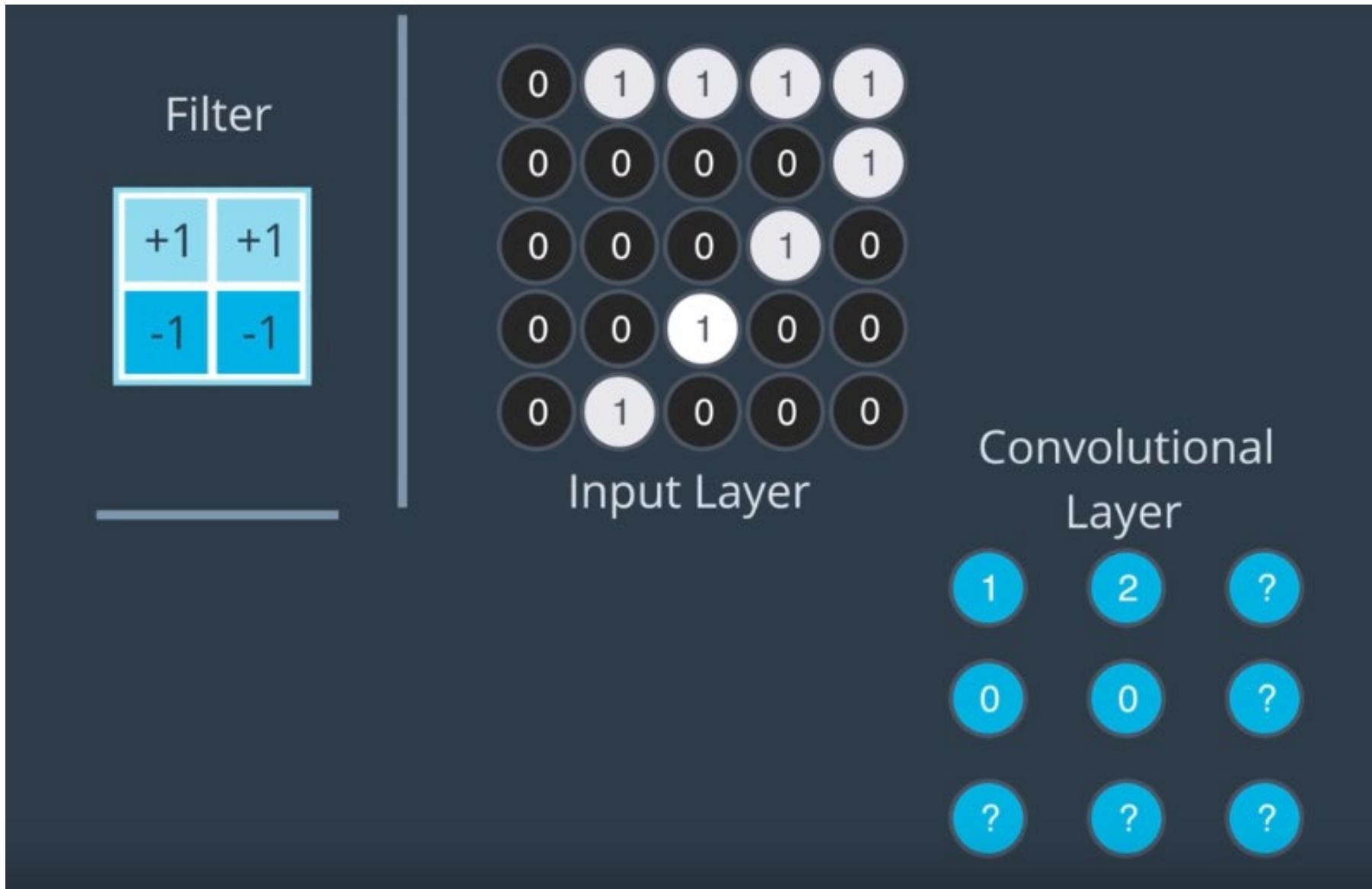
Saturdays.AI  
Guayaquil

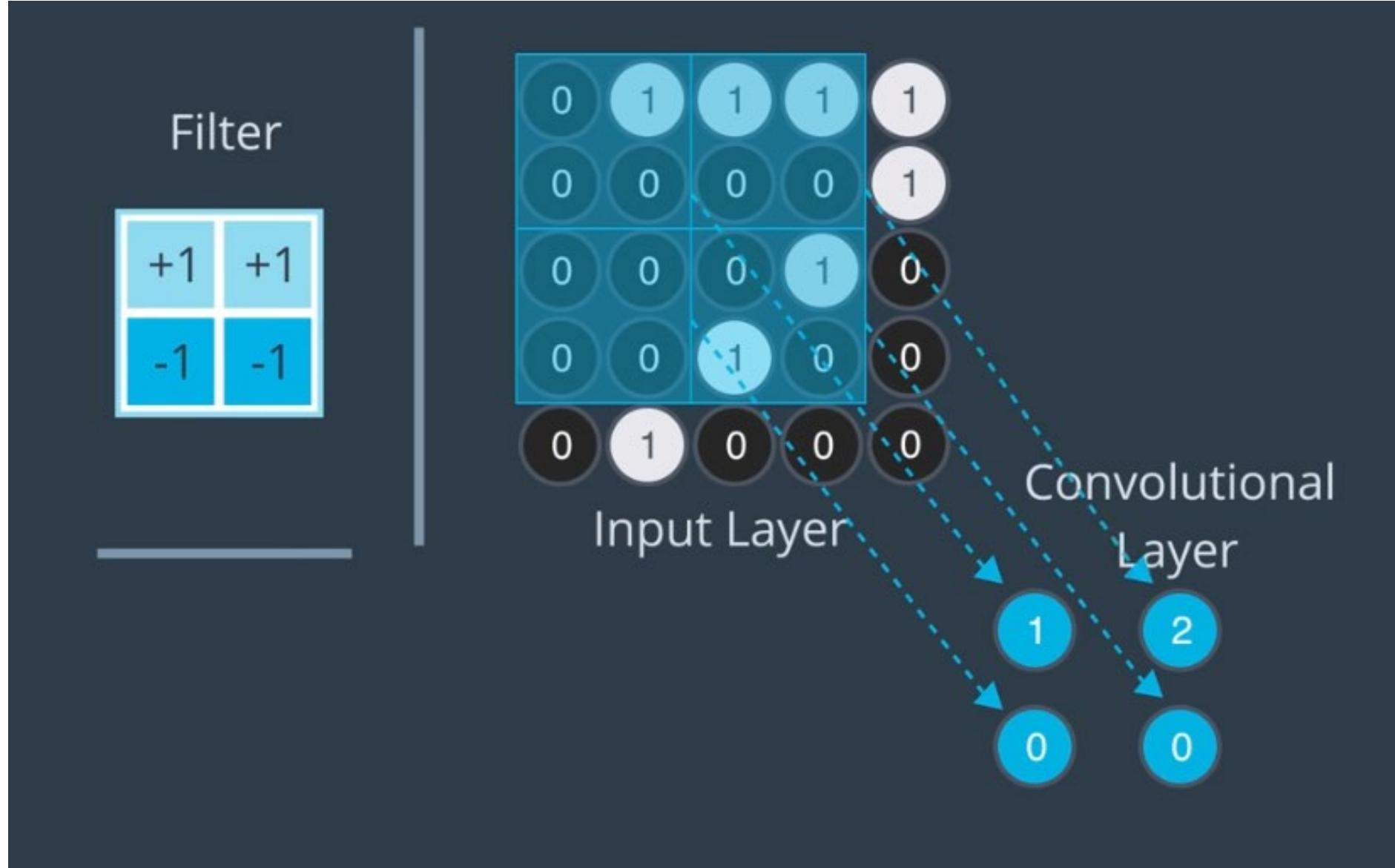


- Stride es la cantidad por la cual el filtro se desliza por la imagen.
- Stride casi igual a uno hará a la capa de convolución de la misma altura y ancho que la imagen de origen.

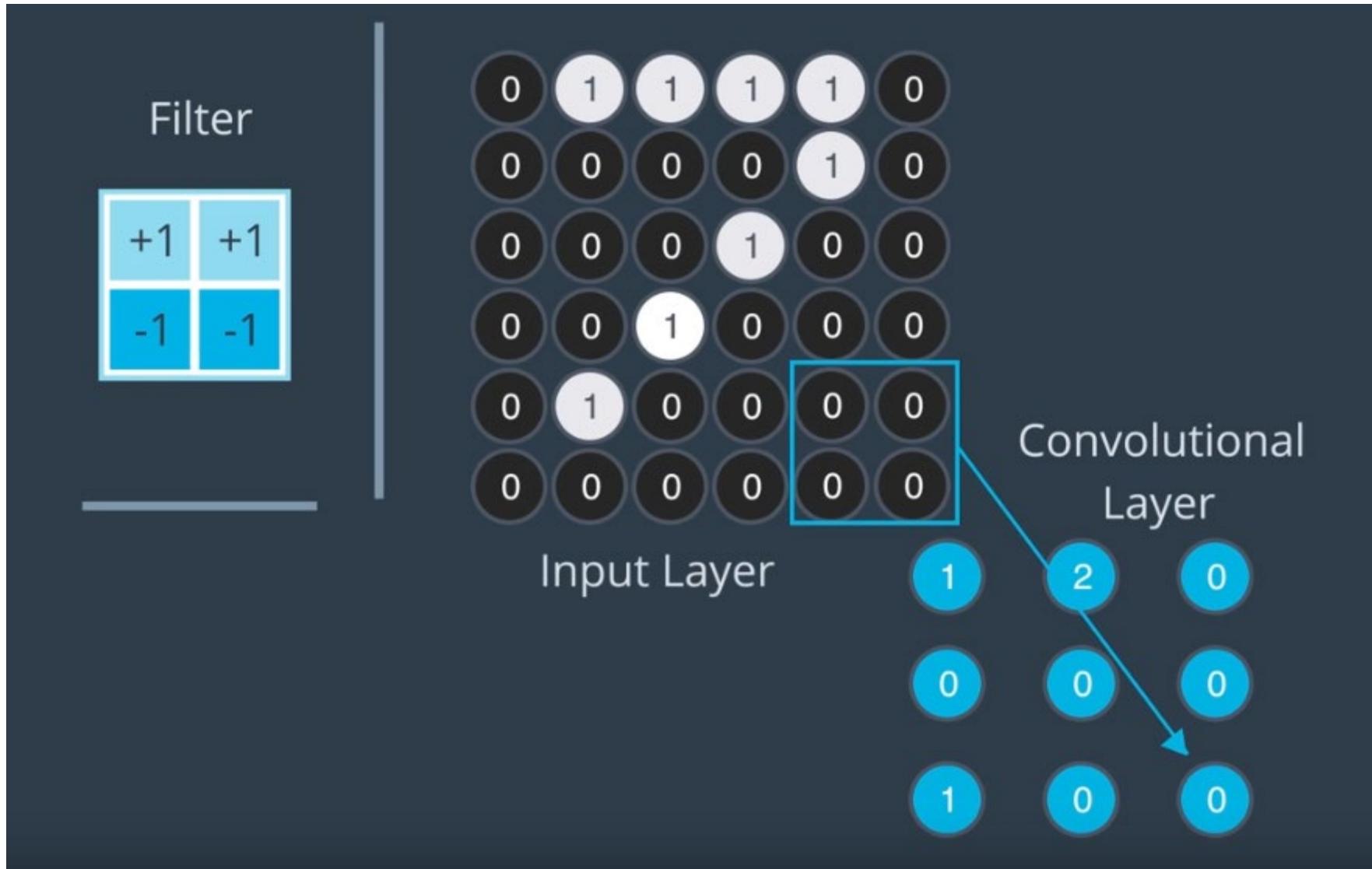


- Si el stride es casi igual a dos la capa convolucional será la mitad en altura y ancho.





Al eliminar nodos es posible que nuestra capa convolucional no tenga información de ciertas regiones de la imagen



Otra opción es rellenar la imagen con ceros.  
Tenemos contribución de todas las regiones de la imagen.

$$O = \frac{(W - K + 2P)}{S} + 1$$

Calcular salida

/08

## Pooling layers



INPUT

CONV

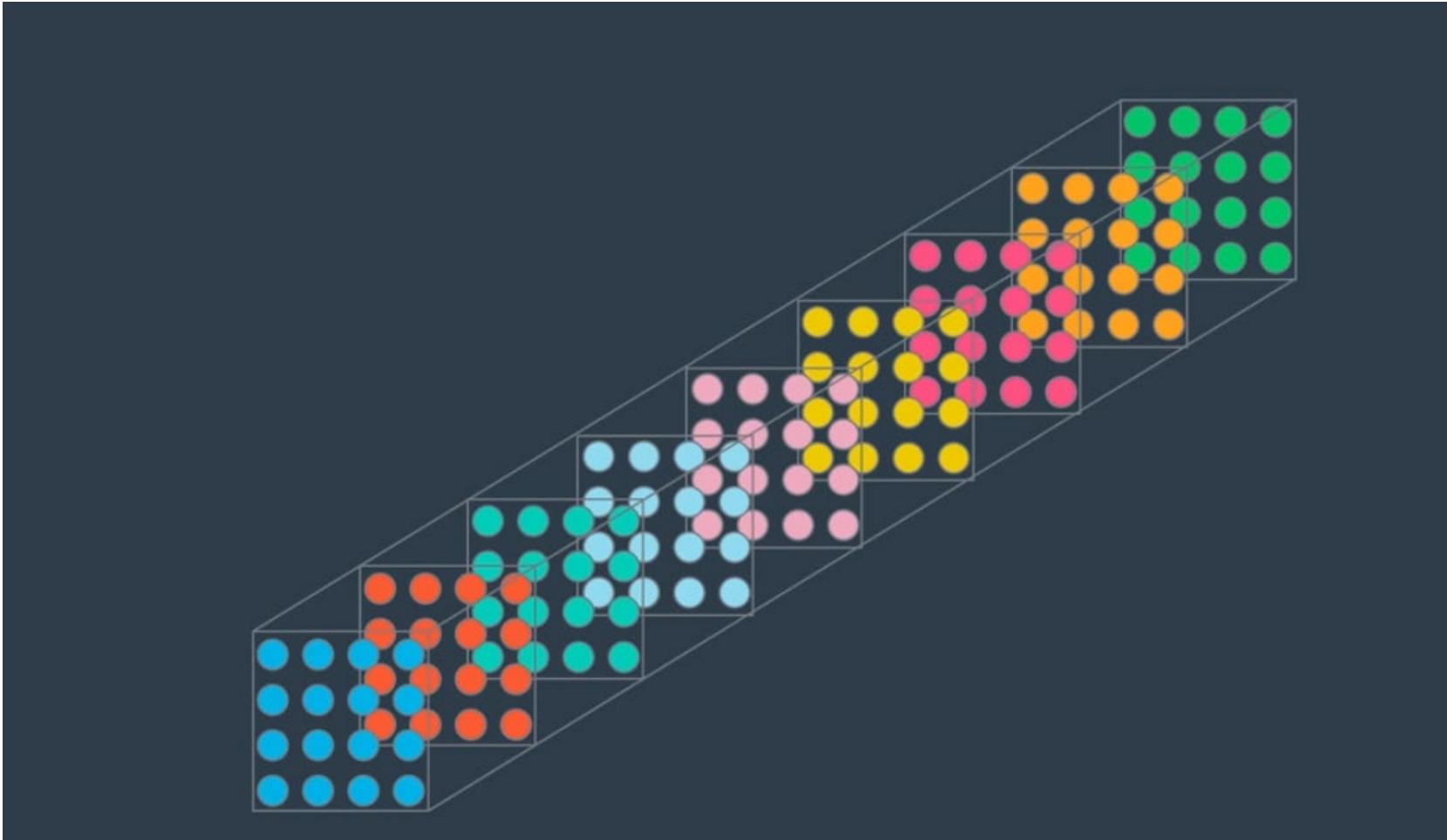
POOL

CONV

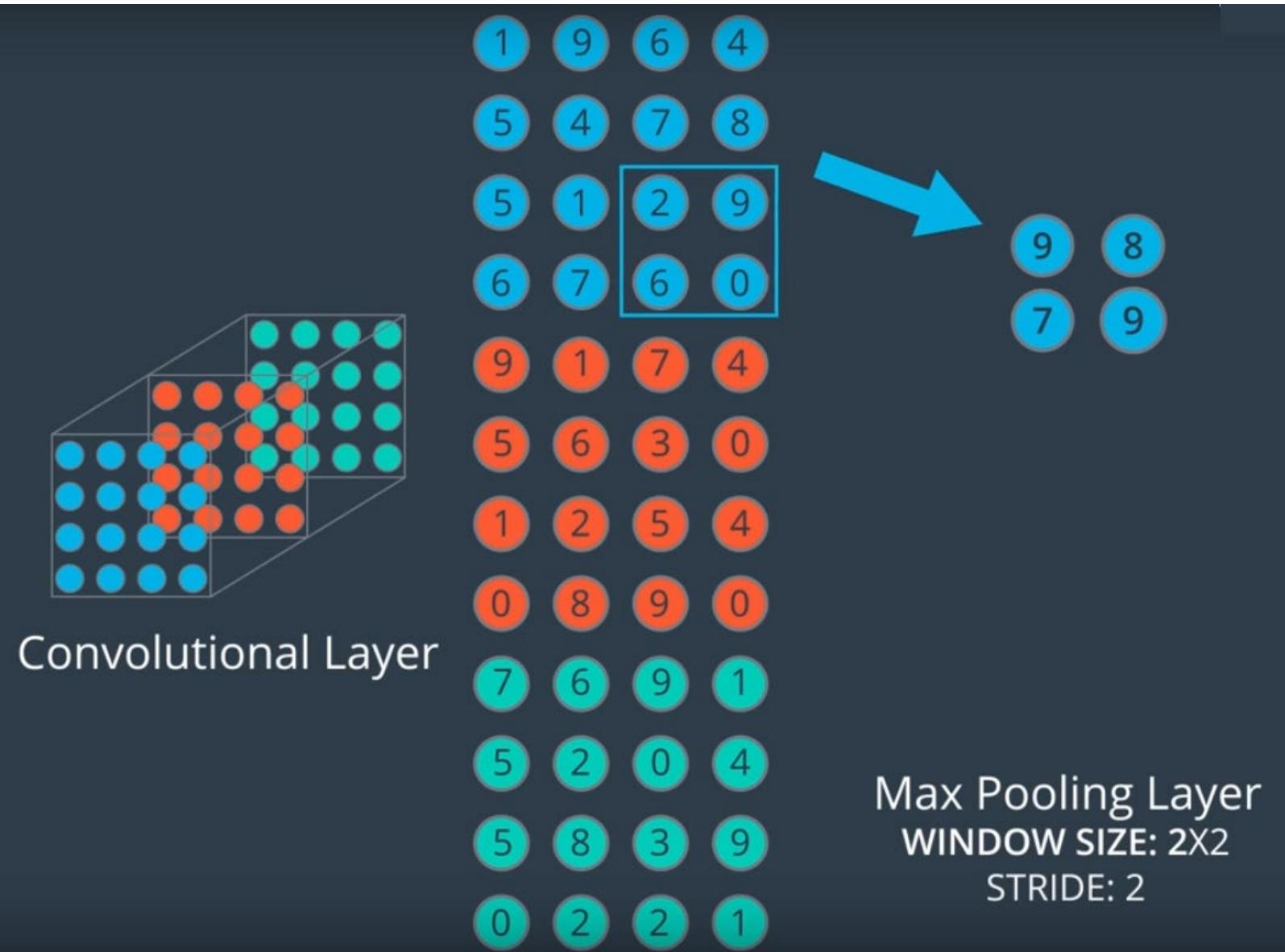
POOL

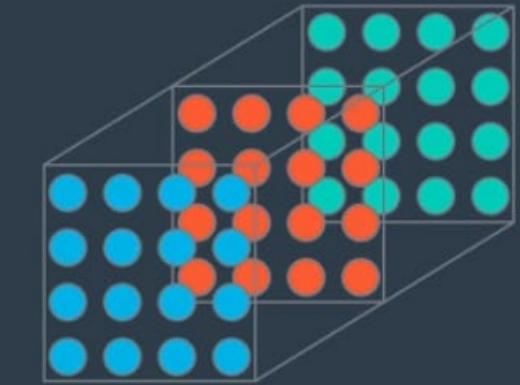
CONV

POOL



- Para varios objetos en una imagen necesitaremos más filtros.
- Más filtros significa un set más grande y mayor dimensionalidad.
- Mayor dimensionalidad significa que usaremos más parámetros (overfitting).
  - Para solucionar esto usaremos pooling layers (existen dos tipos)



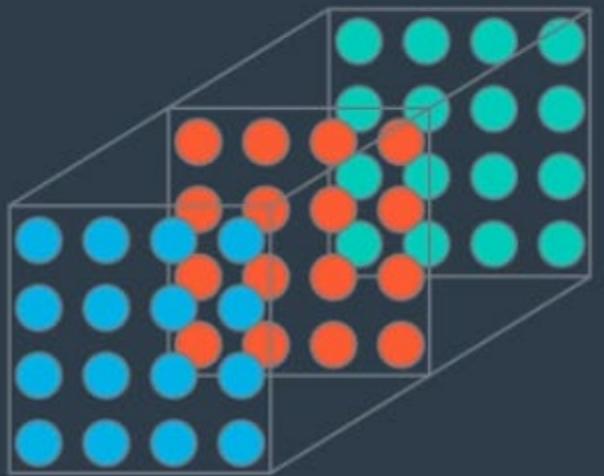


Convolutional Layer

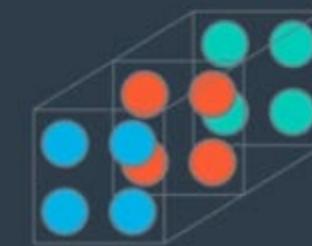
1	9	6	4
5	4	7	8
5	1	2	9
6	7	6	0
9	1	7	4
5	6	3	0
1	2	5	4
0	8	9	0
7	6	9	1
5	2	0	4
5	8	3	9
0	2	2	1

A large blue arrow points from the original 4x4 input grid to a 2x2 output grid. A red arrow points from the intermediate 2x2 grid to a final 2x2 output grid. The final output grid contains values 9, 8, 7, 9, 8, 9, 7, 9.

Max Pooling Layer  
WINDOW SIZE: 2x2  
STRIDE: 2



Convolutional Layer

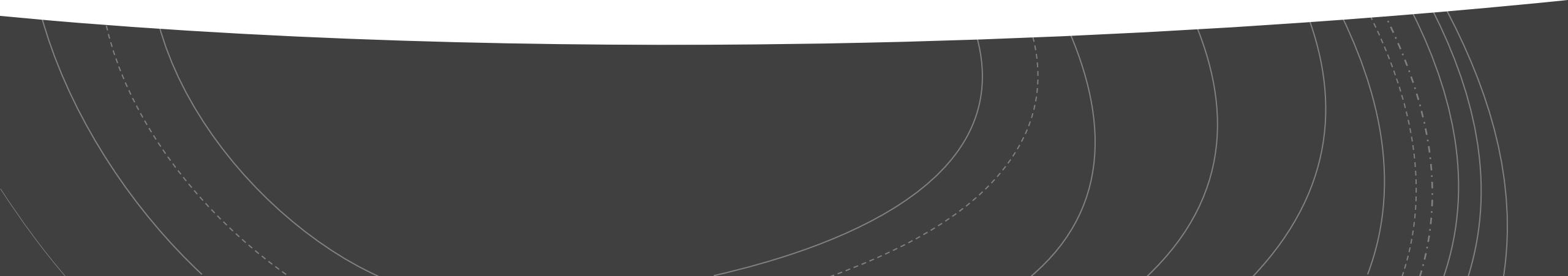


Max Pooling Layer  
**WINDOW SIZE: 2X2**  
**STRIDE: 2**

# Otros tipos de pooling

- Algunas arquitecturas eligen usar el average pooling ([promedio](#)), que elige valores promedio de píxeles en un tamaño de ventana determinado. Por lo tanto, en una ventana de 2x2, esta operación verá valores de 4 píxeles y devolverá un solo promedio de esos cuatro valores, como salida.
- Este tipo de pooling generalmente no se usa para problemas de clasificación de imágenes porque maxpooling es mejor para notar los detalles más importantes sobre los bordes y otras características en una imagen, pero puede ver esto utilizado en aplicaciones para las que es preferible suavizar una imagen.

# Maxpooling\_visualization notebook



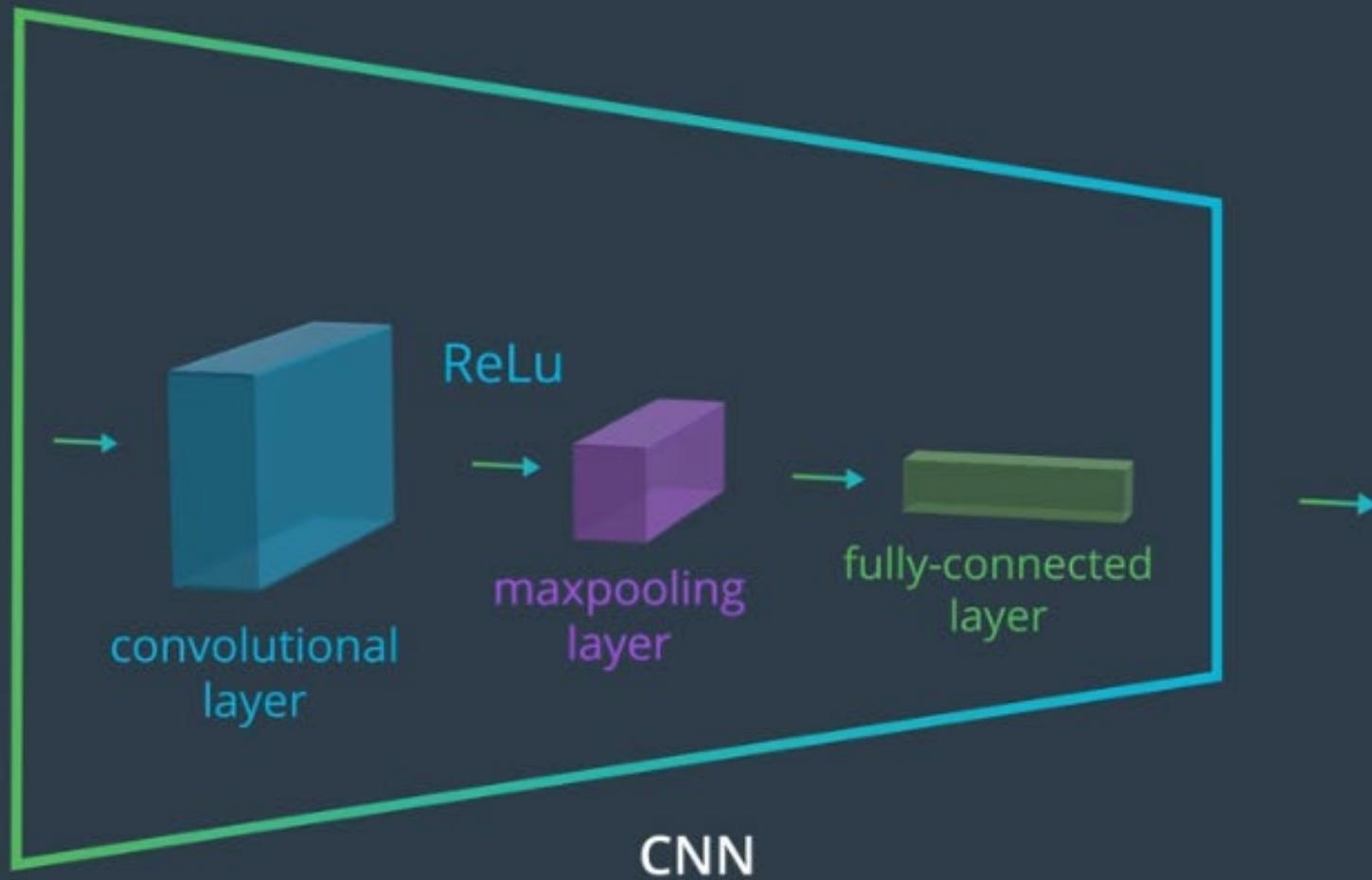
/09

Aumentando la profundidad



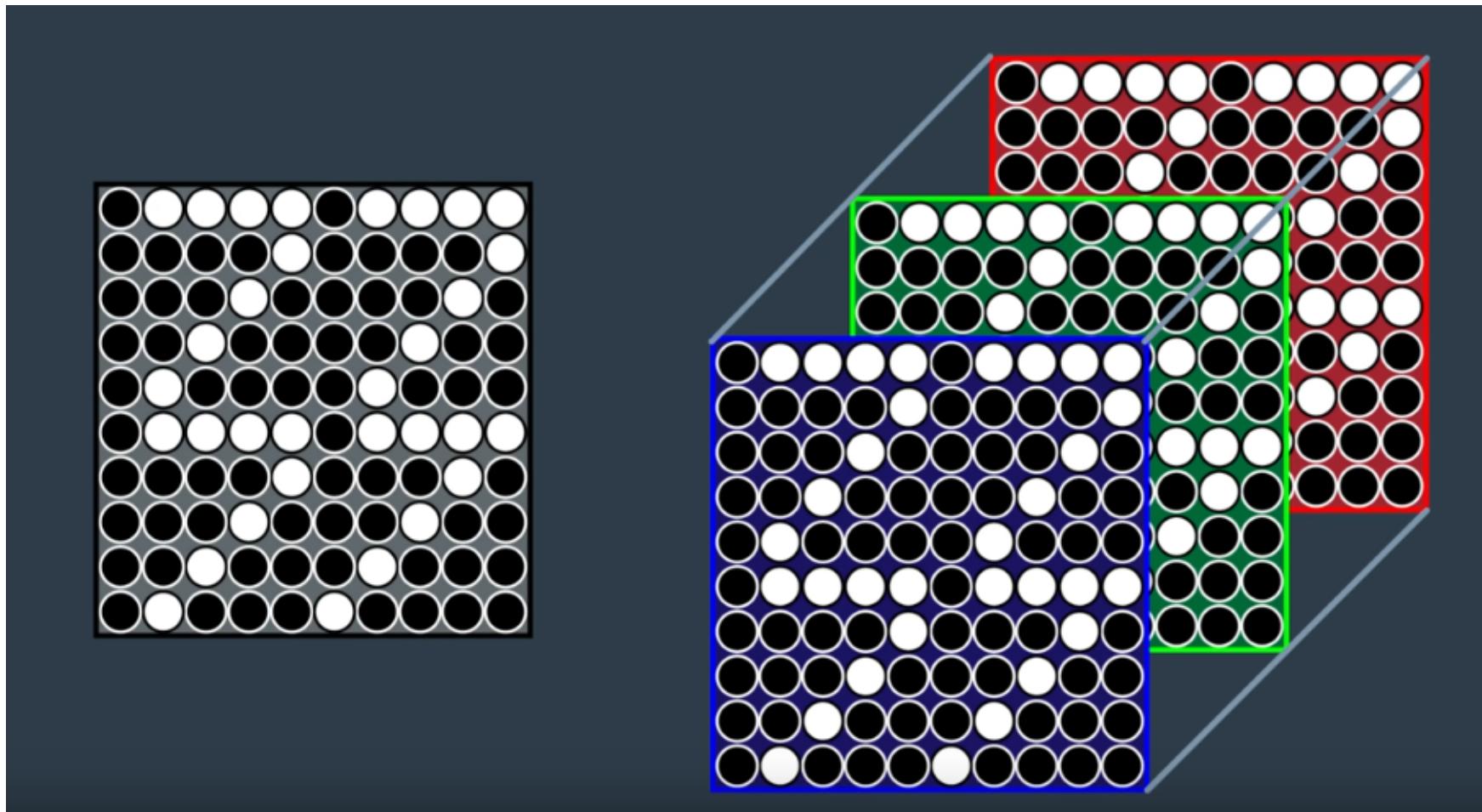


Input Image

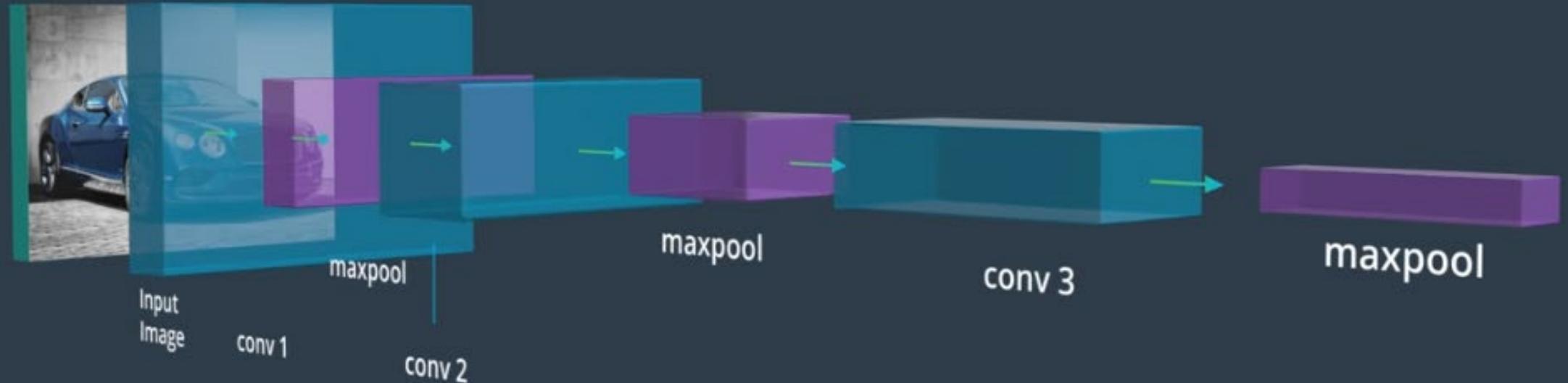








Las imágenes son arreglos. En color tiene profundidad de 3 y en escala de grises profundidad de 1.  
La altura y el ancho son de mayor dimensión que la profundidad.



- Nuestra arquitectura CNN busca tomar el arreglo de entrada y hacer que su profundidad sea mayor que la Dimensión de su altura y ancho.
- Al irse haciendo más profunda extrae más y más características y ayuda a identificar contenido y objetos en la imagen.
- Capas convolucionales la hacen más profunda y las capaz pooling reducen las dimensiones X y Y.
- Además descarga información espacial que no da información como fondos nublosos.

## Define a convolutional layer in PyTorch

```
self.conv1 = nn.Conv2d(3, 16,  
3, stride = 1, padding = 1)
```

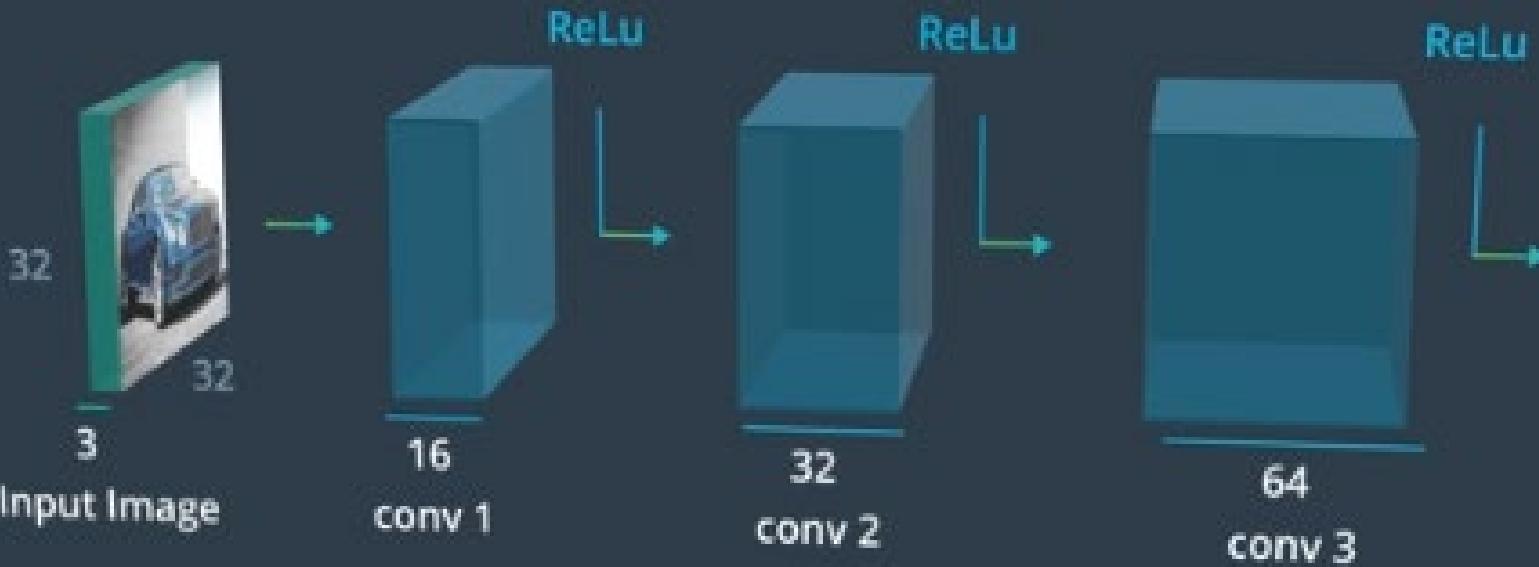


Input Image

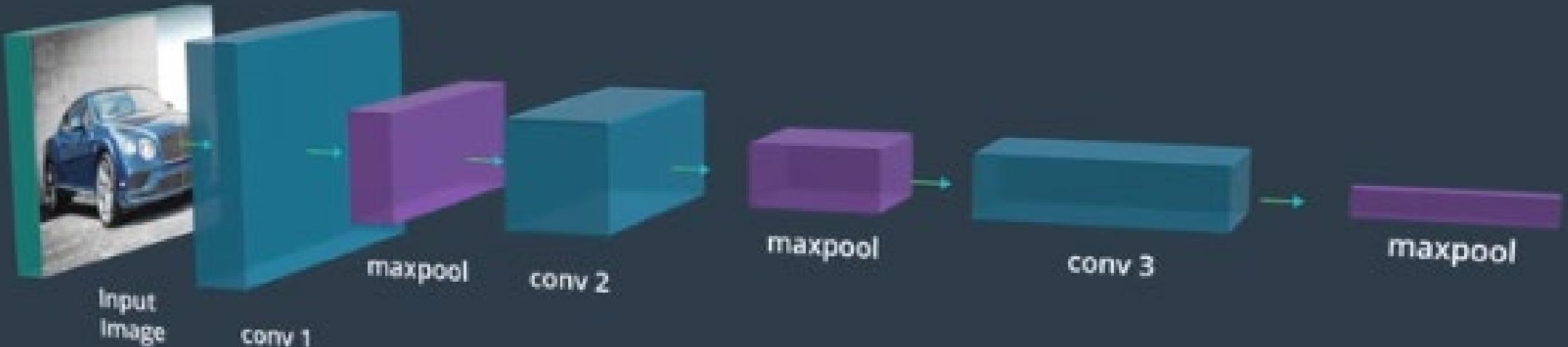


convolutional  
layer





```
self.conv1 = nn.Conv2d(3, 16, 3, padding = 1)  
self.conv2 = nn.Conv2d(16, 32, 3, padding = 1)  
self.conv3= nn.Conv2d(32, 64, 3, padding = 1)
```



```
self.conv1 = nn.Conv2d(3, 16, 3, padding = 1)
self.conv2 = nn.Conv2d(16, 32, 3, padding = 1)
self.conv3 = nn.Conv2d(32, 64, 3, padding = 1)

self.maxpool = nn.MaxPool2d(kernel_size, stride)
```

# Padding



Padding solo agrega un borde de píxeles alrededor de una imagen. En PyTorch, especifica el tamaño de este borde.



¿Por qué necesitamos relleno? Cuando creamos una capa convolucional, movemos un filtro cuadrado alrededor de una imagen, usando un píxel central como ancla. Entonces, este núcleo no puede superponer perfectamente los bordes/esquinas de las imágenes. La buena característica del relleno es que nos permitirá controlar el tamaño espacial de los volúmenes de salida (más comúnmente, lo usaremos para preservar exactamente el tamaño espacial del volumen de entrada para que el ancho y la altura de entrada y salida sean iguales).

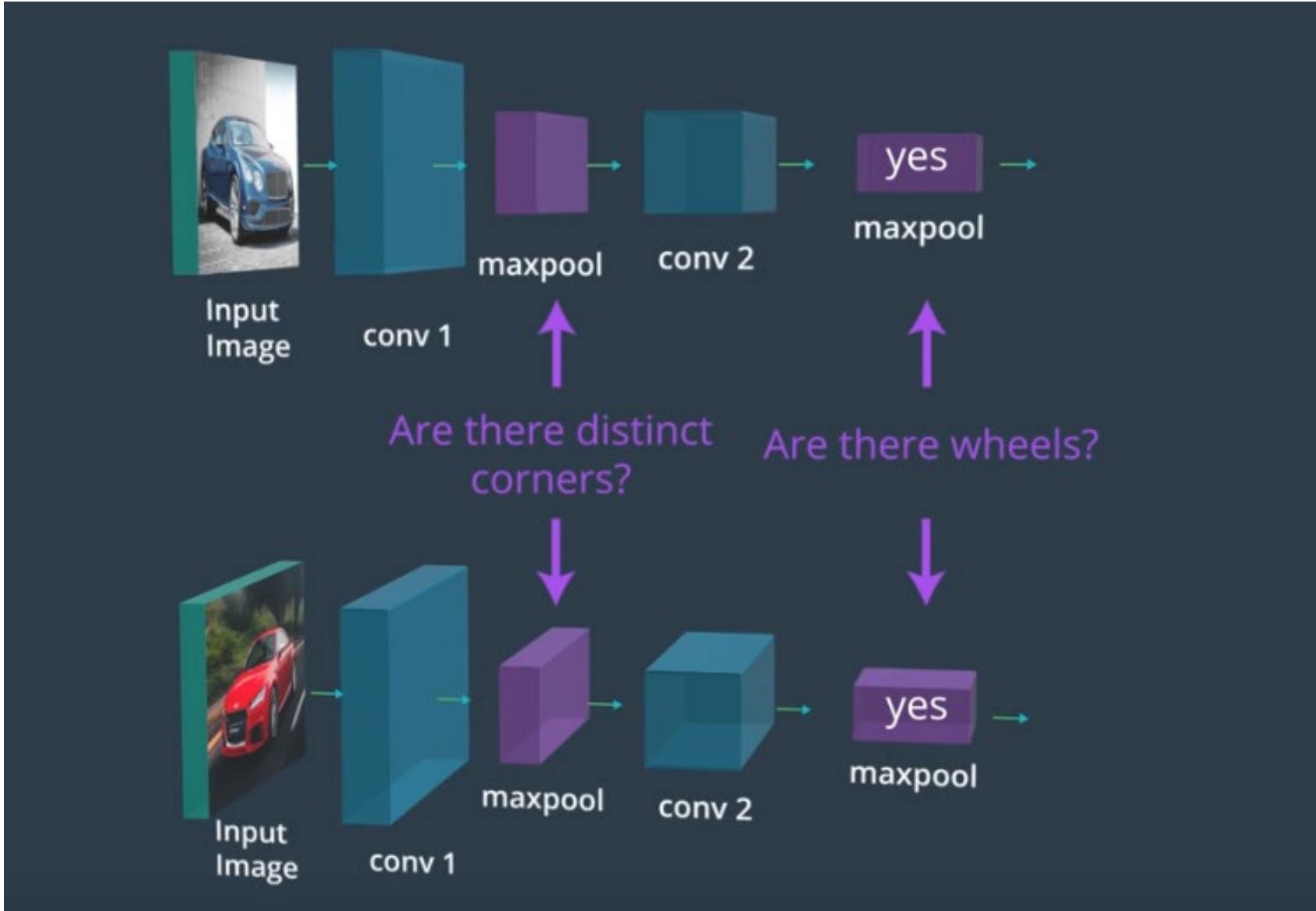


Puedes leer más sobre el cálculo de la cantidad de relleno, dado un `kernel_size`, [aquí](#).

/10

Vector de característica





Codifican solo el contenido de las imágenes. Esto es llamado representación a nivel de características de una imagen o vector de características.



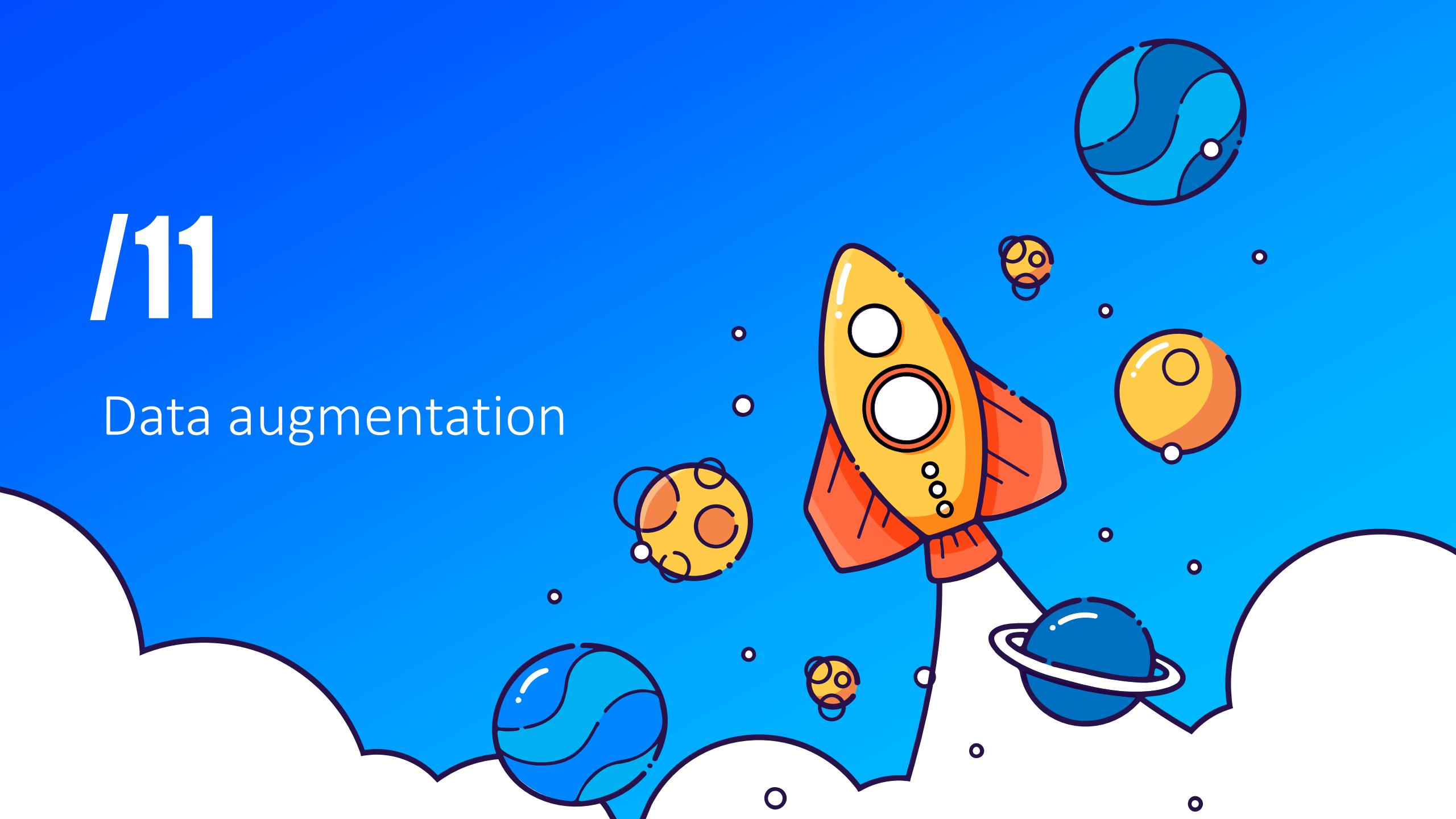
Feature-level  
representation



# Cifar\_exercise notebook

/11

## Data augmentation



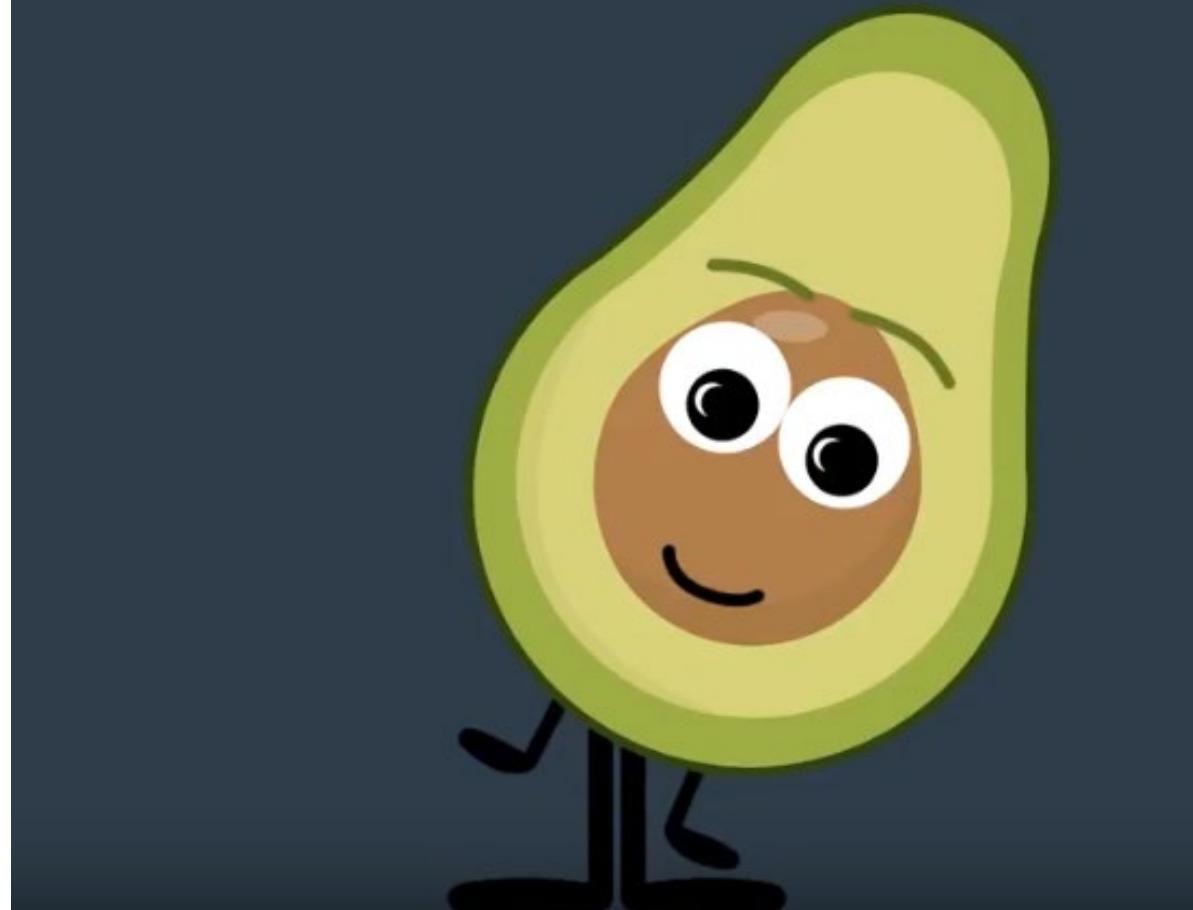
# Invariant representation



# Scale Invariance



# Rotation Invariance



# Translation Invariance





What we see

08 02 22 97 38 15 00 41 00 78 04 05 07 78 82 12 50 77 91 08  
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 18 04 56 62 00  
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65  
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91  
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80  
24 47 32 60 99 03 45 02 44 75 33 53 78 36 04 20 35 17 12 50  
08 02 22 97 38 15 00 41 00 78 04 05 07 78 82 12 50 77 91 08  
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 18 04 56 62 00  
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65  
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91  
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80  
24 47 32 60 99 03 45 02 44 75 33 53 78 36 04 20 35 17 12 50  
08 02 22 97 38 15 00 41 00 78 04 05 07 78 82 12 50 77 91 08  
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 18 04 56 62 00

What computers see



Rotation invariance



Translation invariance

# Data Augmentation



Cifar10\_cnn\_augmentationexercise  
notebook

