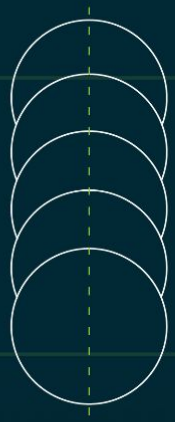
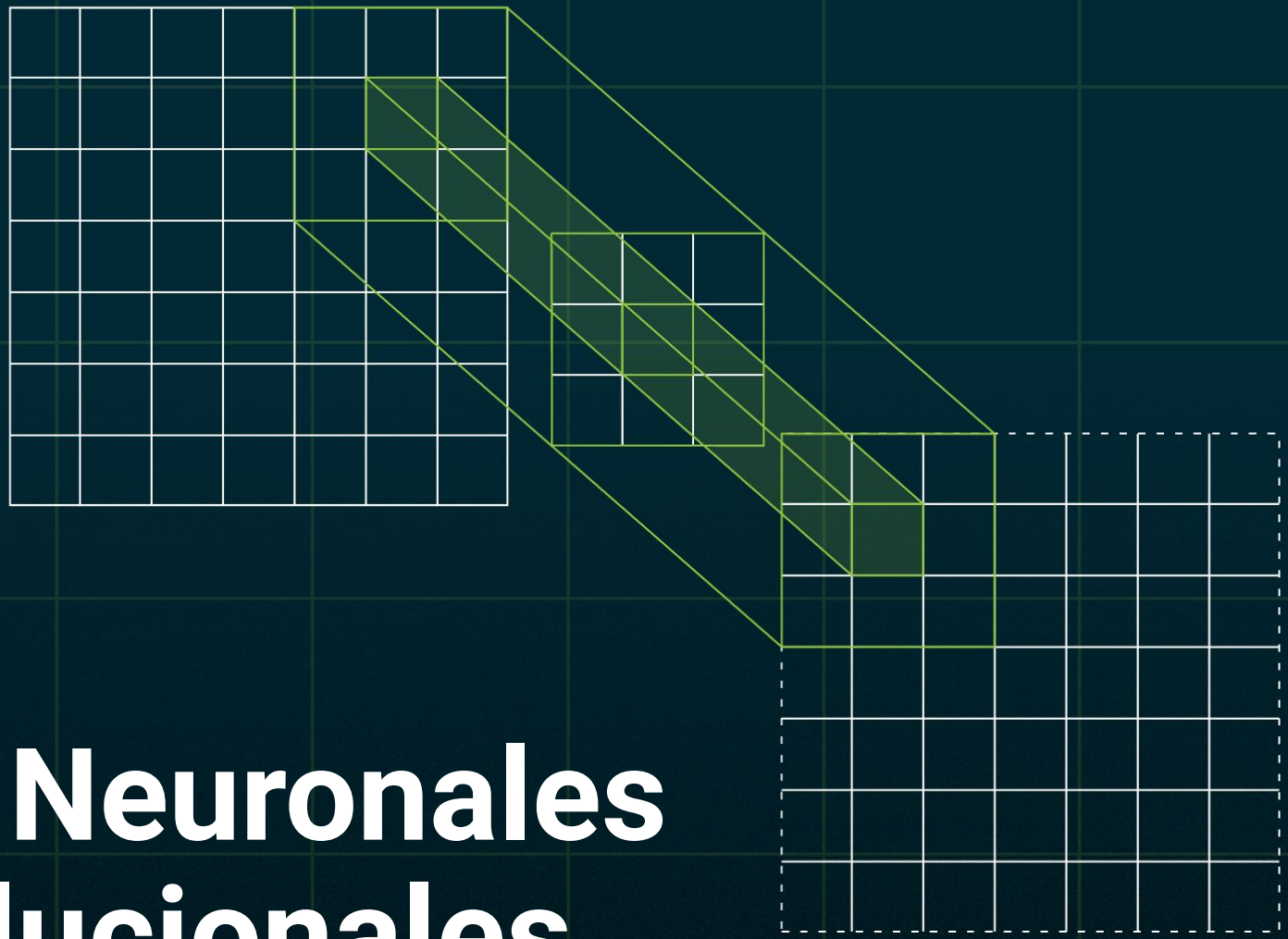
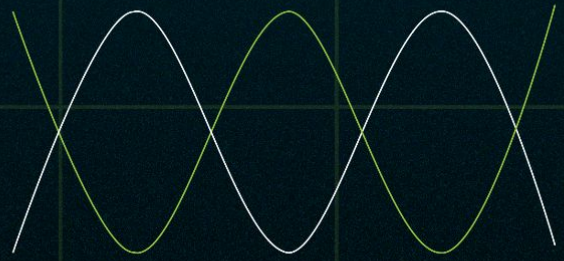


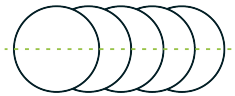


Curso de

Redes Neuronales Convolucionales

Carlos Alarcón



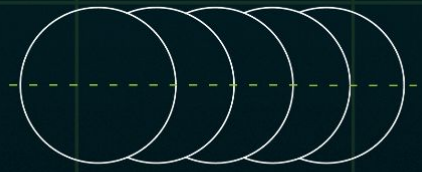


Un poco sobre mí

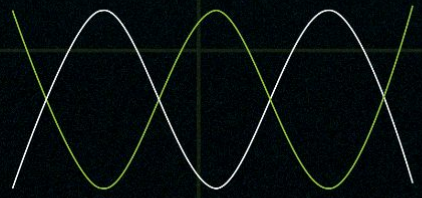
🧑‍💼 Data scientist en Platzi.

💻 Especialista en ciencia de datos / bases de datos / AI.

🧠 Profe de AWS Redshift y redes neuronales.



La importancia de computer vision

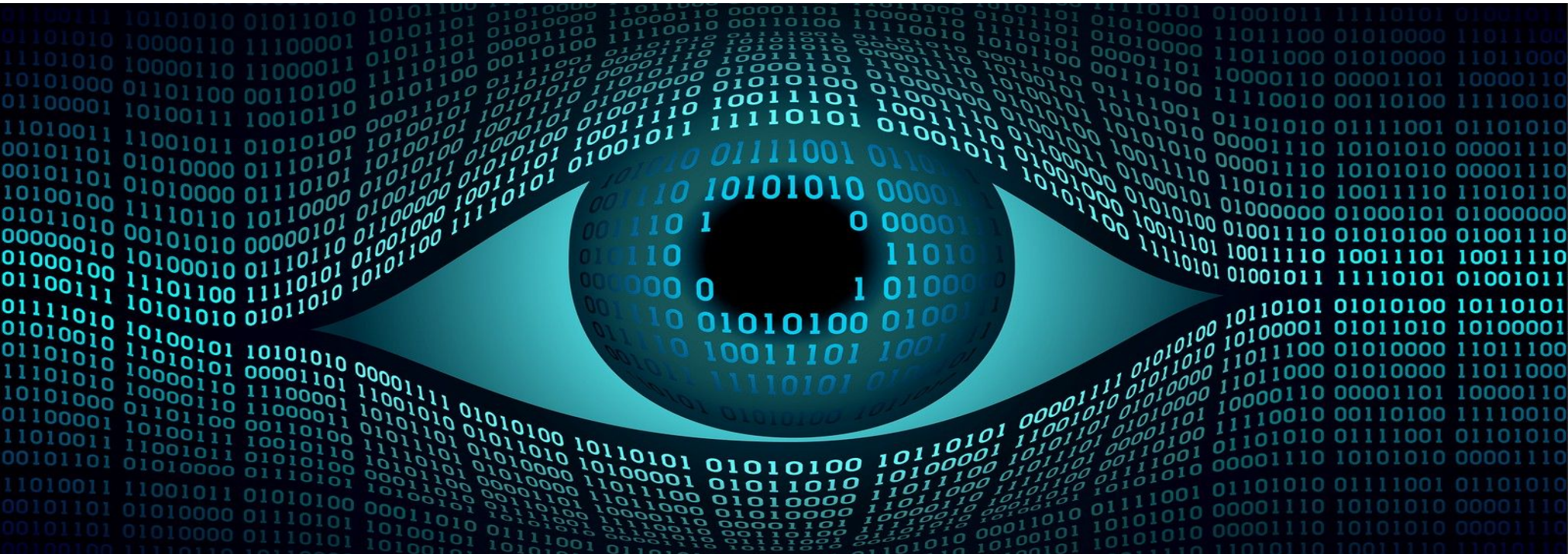


La importancia de la visión

¿Qué tan importante es la visión en tu día a día?

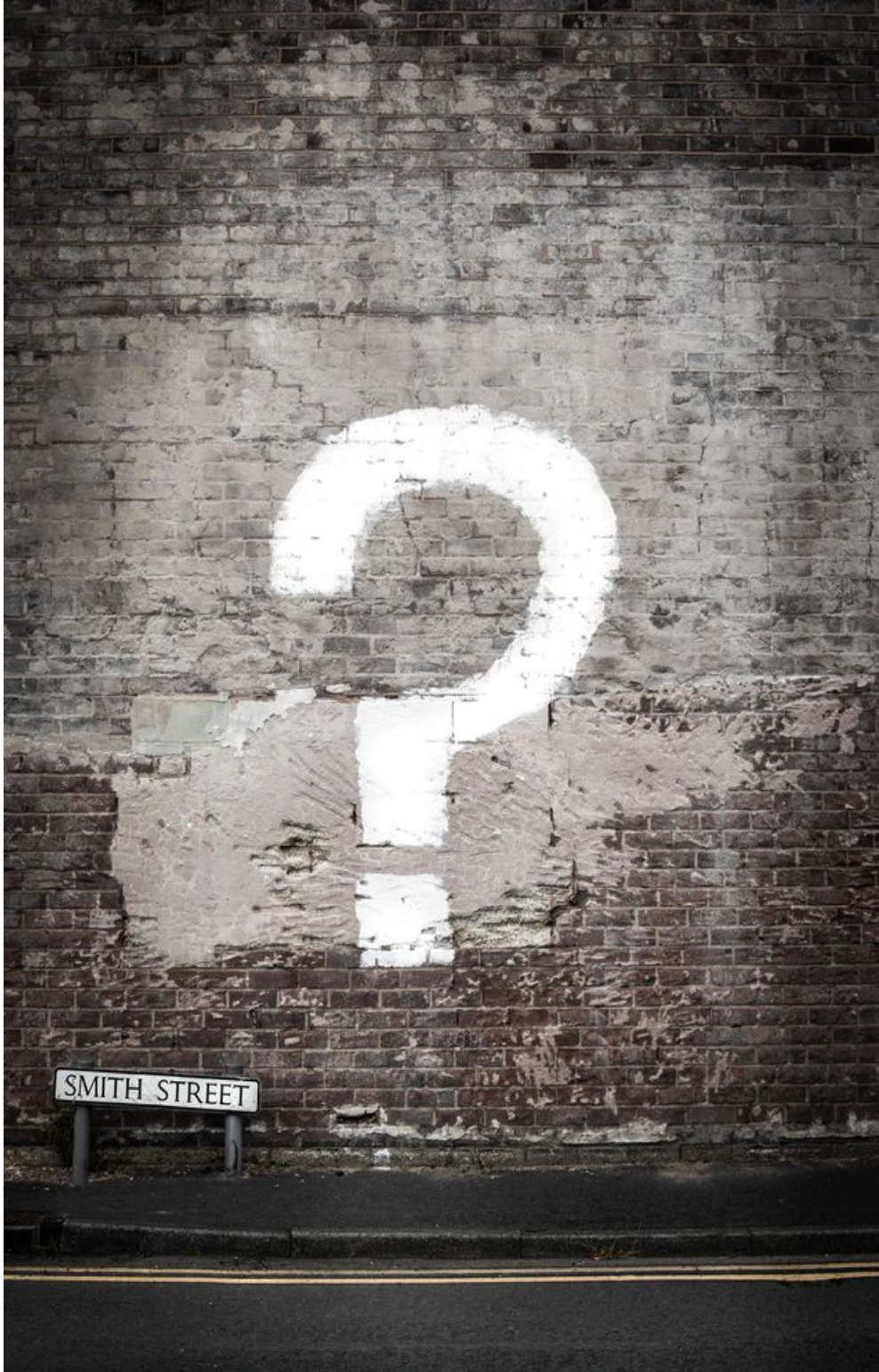
¿Qué es computer vision?

Campo de la inteligencia artificial encargado de proporcionar a las máquinas la capacidad de “ver”.

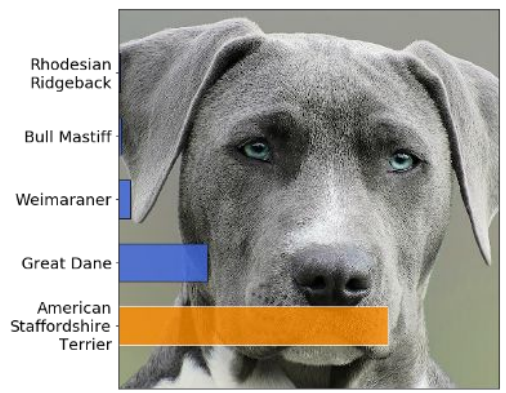


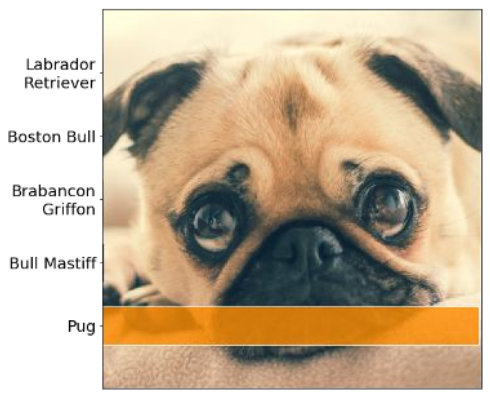
Computer vision

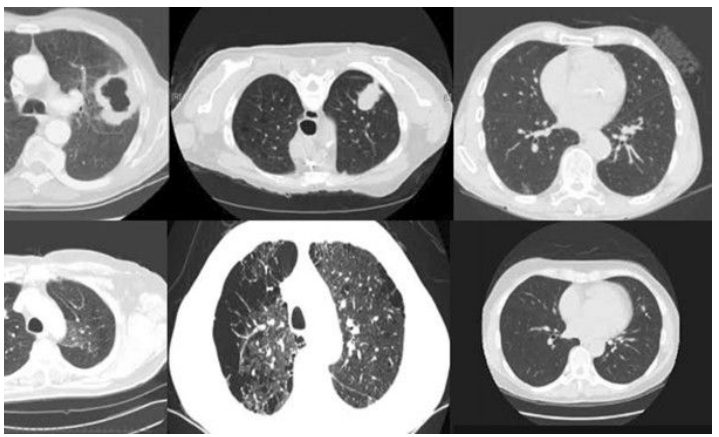
¿Pero qué puedo lograr con simular la visión en una máquina?

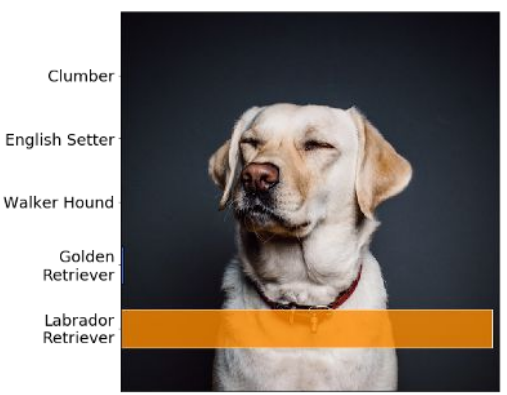


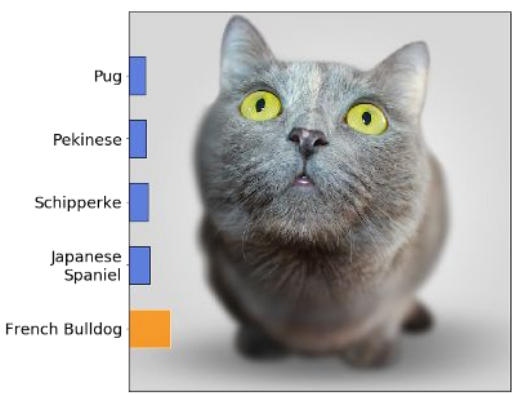
Clasificación de imágenes



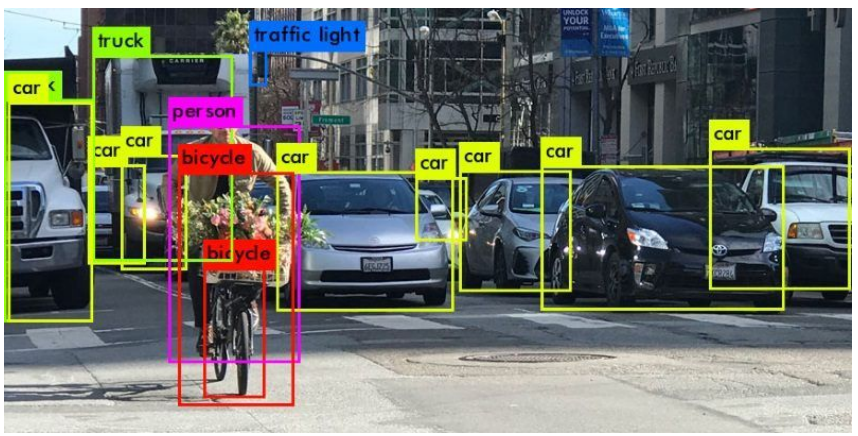


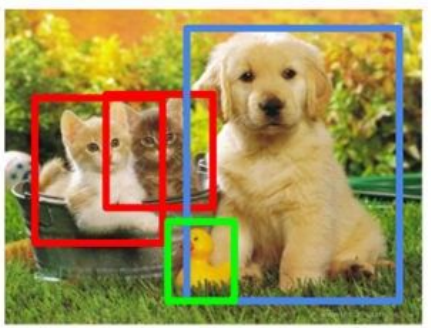




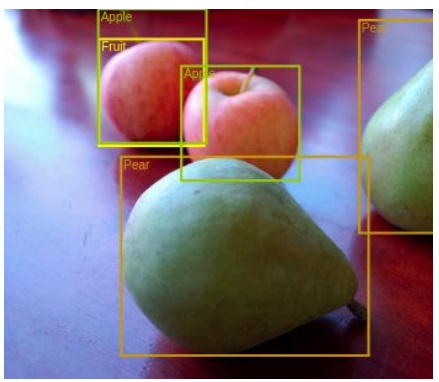


Detección de objetos

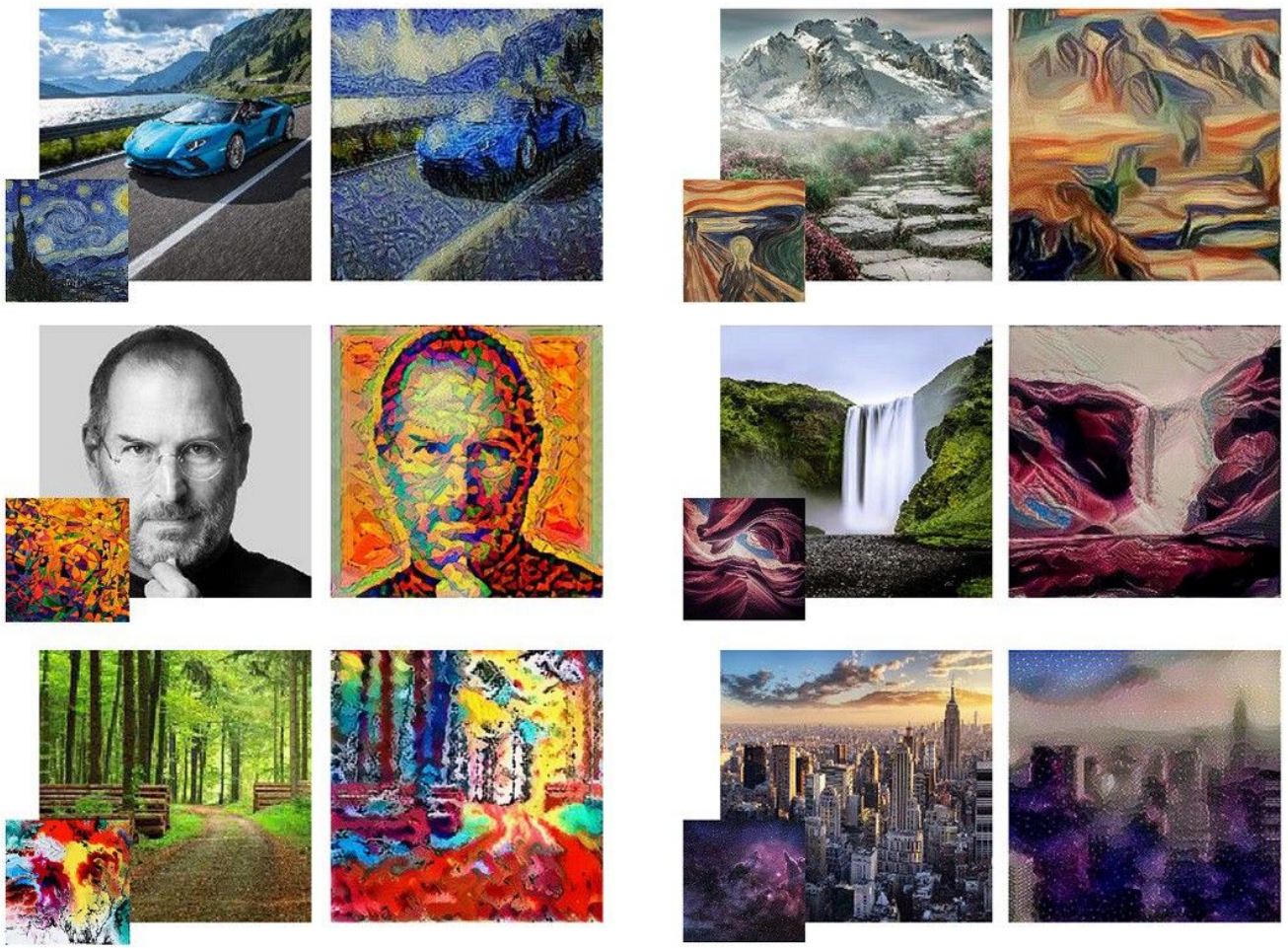




CAT, DOG, DUCK



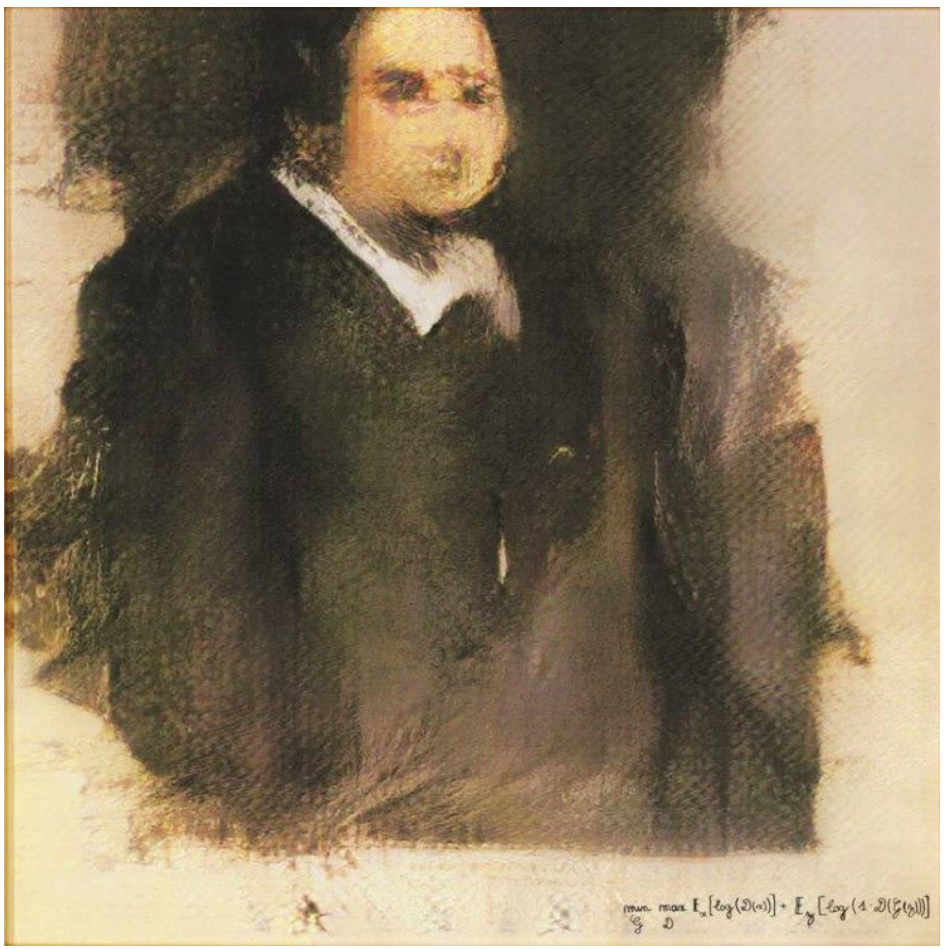
Transferencia de estilos



Transferencia de estilos

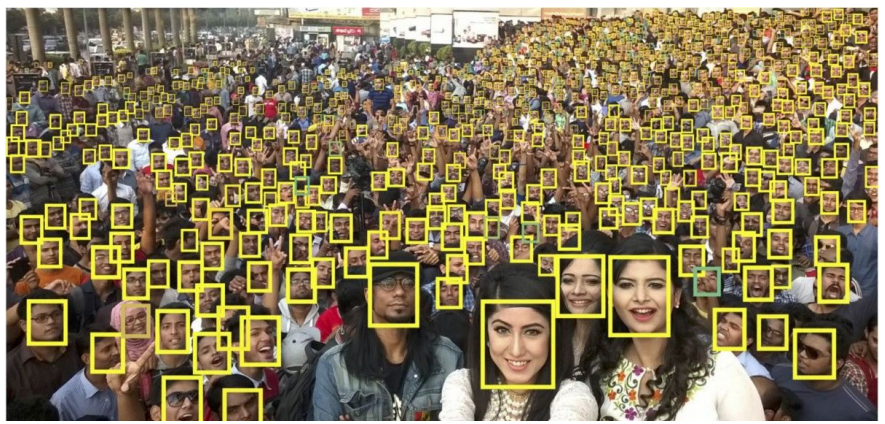
GANs

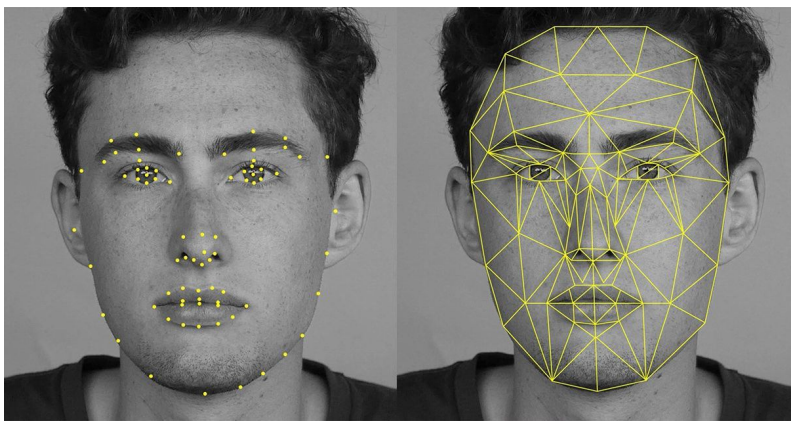
GANs

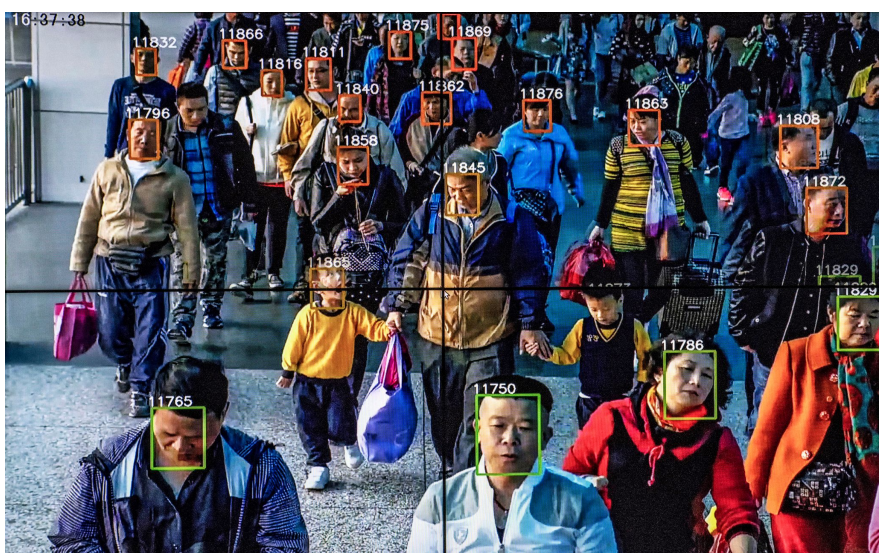


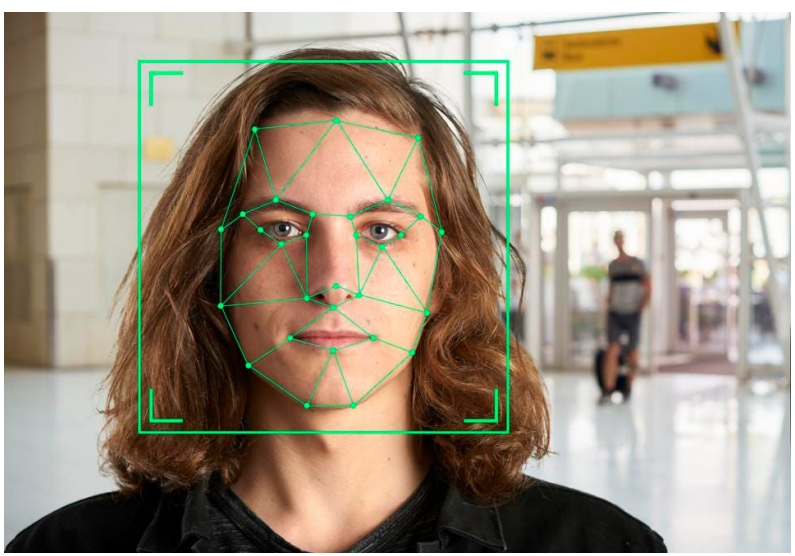
$432,500

Reconocimiento facial

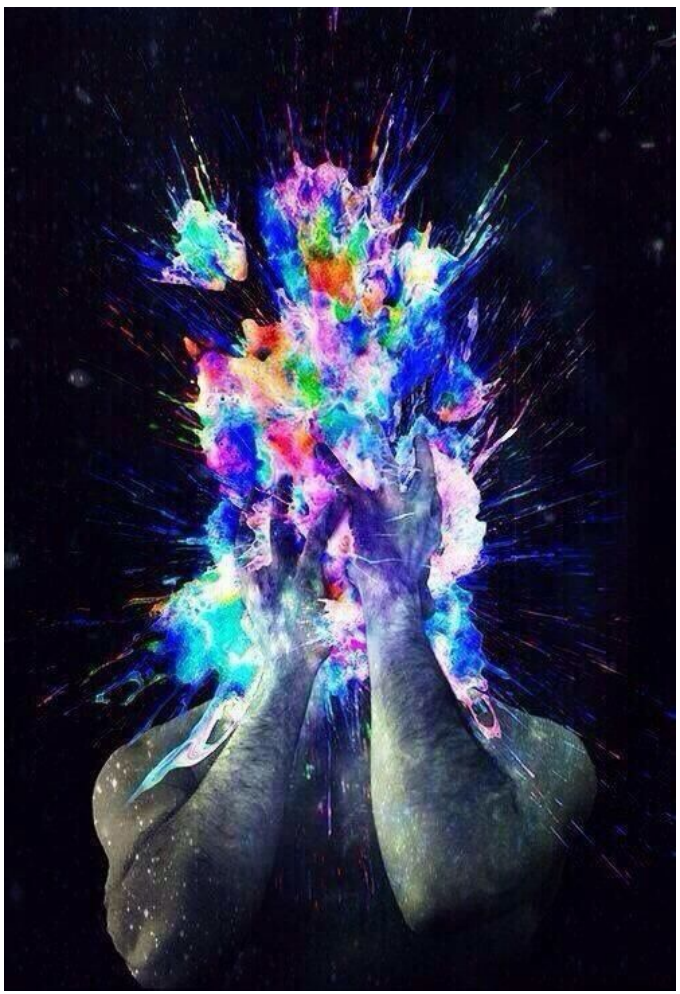


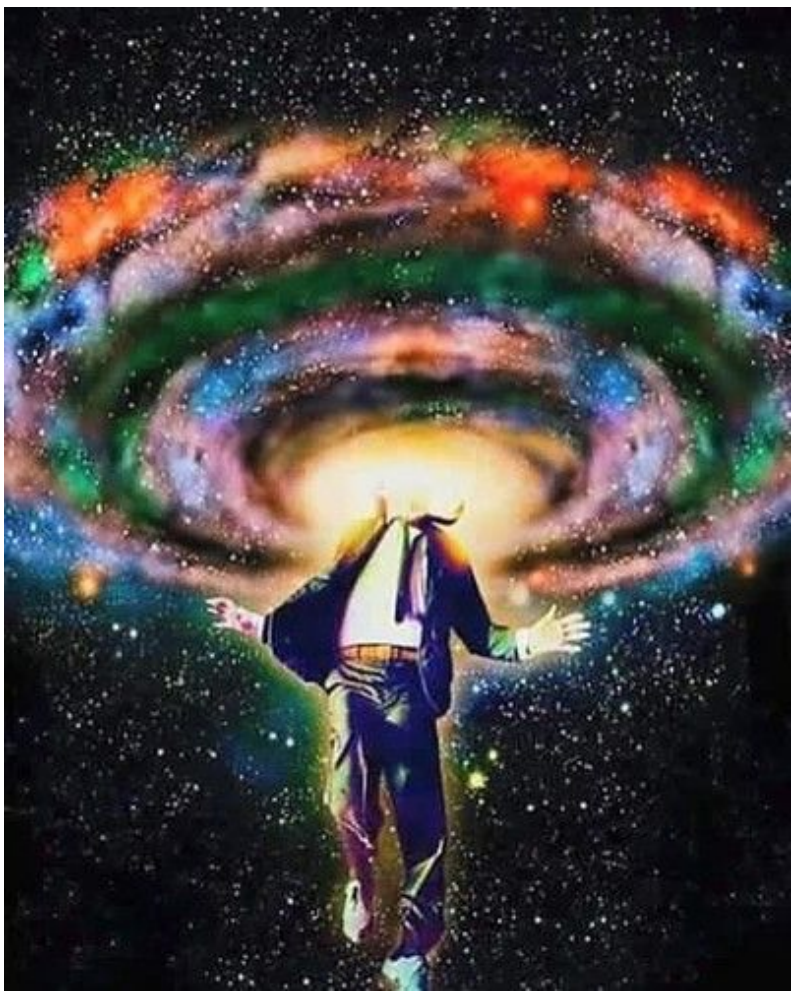






Un mundo de posibilidades

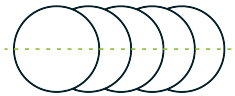




Clasificación con redes convolucionales

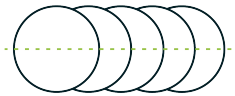
Acá empieza el asombroso viaje dentro del computer vision.

¿Qué herramientas usaremos?

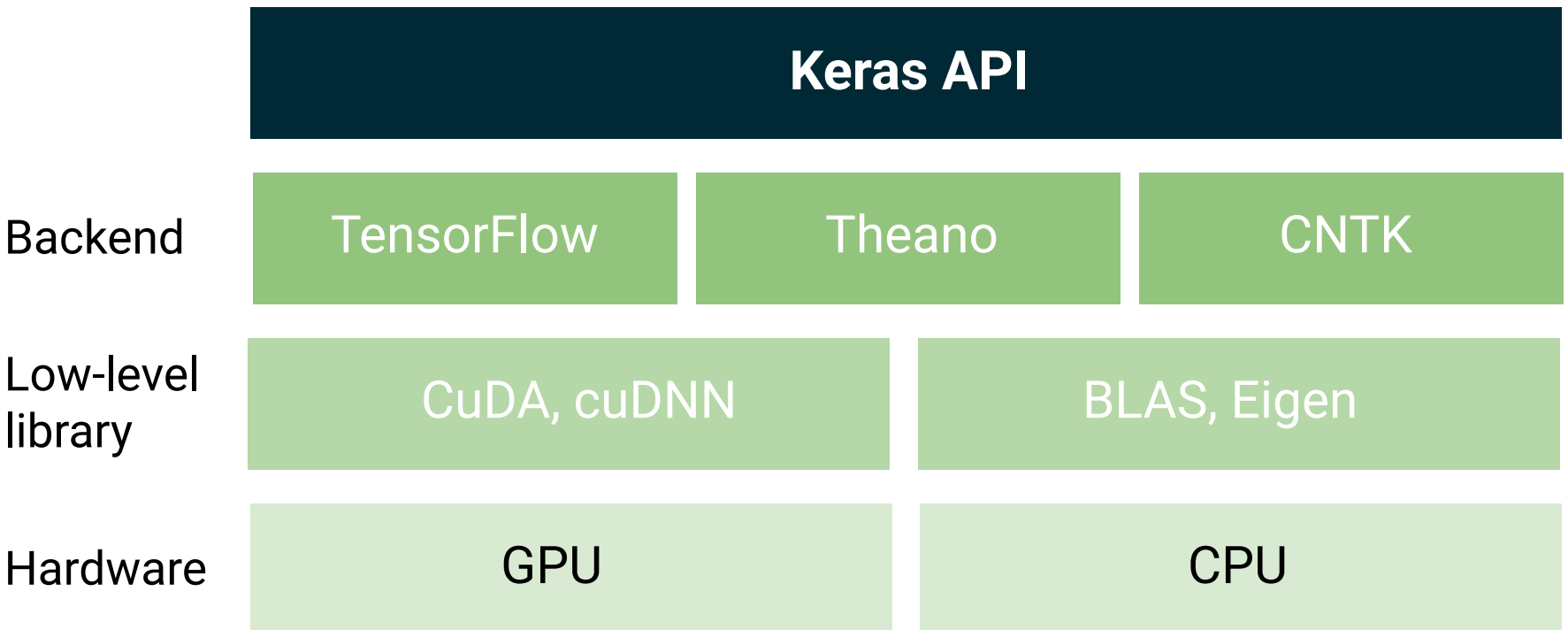


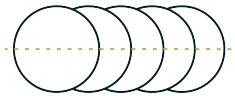
Frameworks

theano

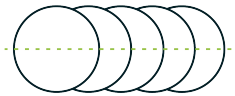


Frameworks


Keras API
Backend
TensorFlow
Theano
CNTK
Low-level library
CuDA, cuDNN
BLAS, Eigen
Hardware
GPU
CPU

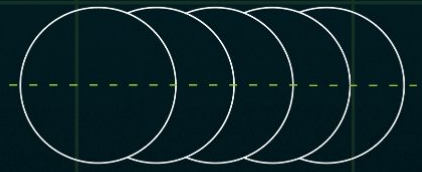


Notebooks

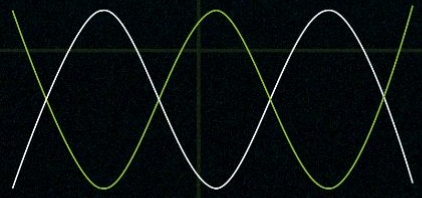


Notebooks

kaggle



¿Qué son las redes convolucionales?



Red neuronal convolucional (CNN)

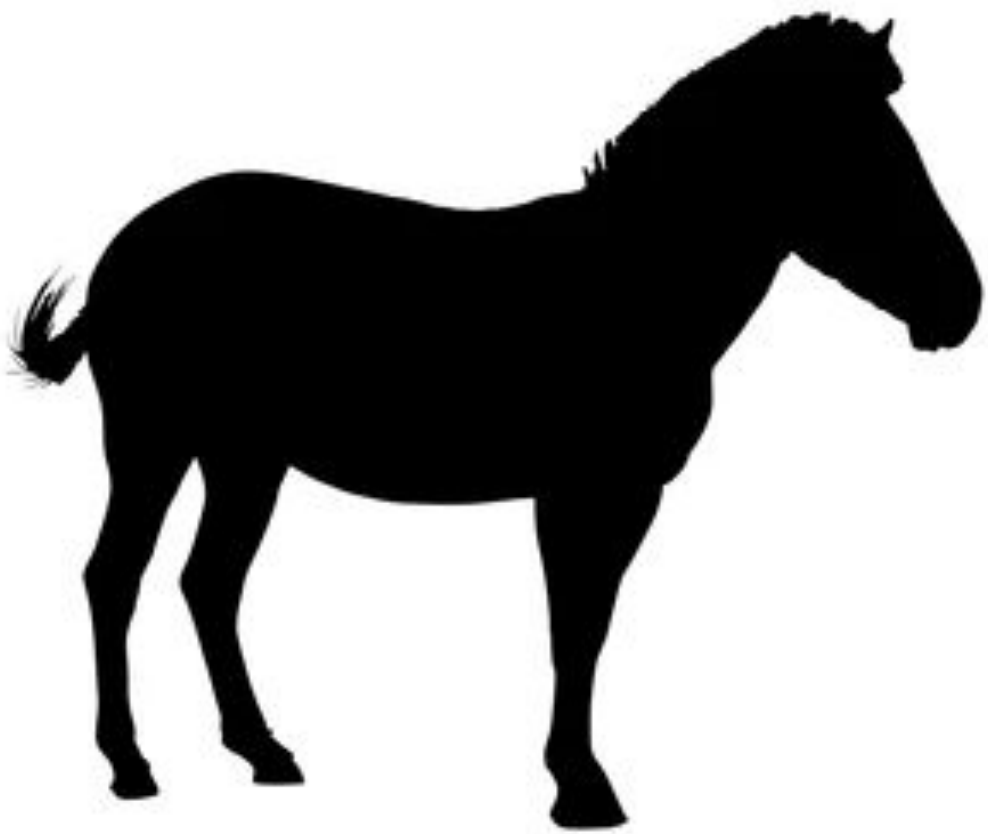
- Red neuronal especializada en el reconocimiento de imágenes.
- Intenta simular el comportamiento de la corteza visual de nuestro cerebro.

¿Cómo vemos los humanos?

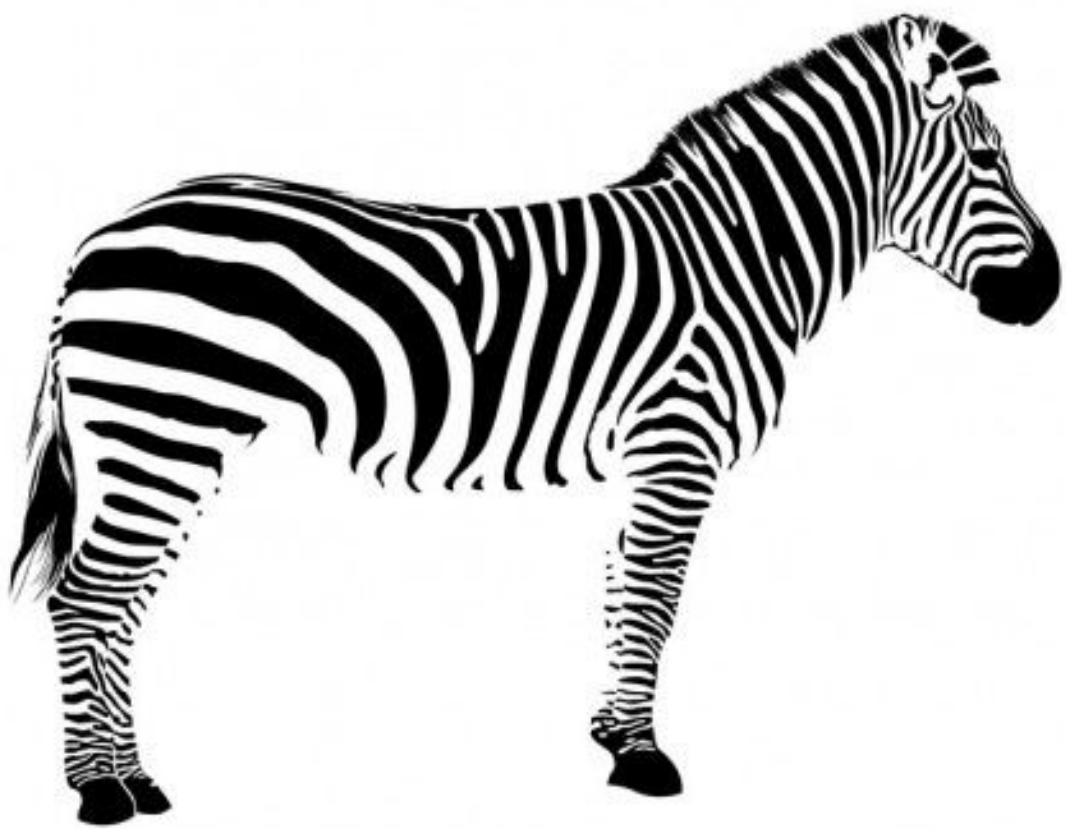
¿Qué es esto?

¿Otro auto?

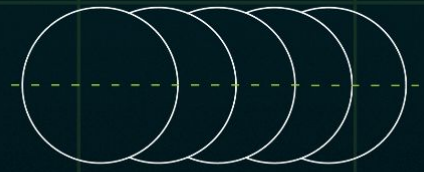
¿Y qué es esto?



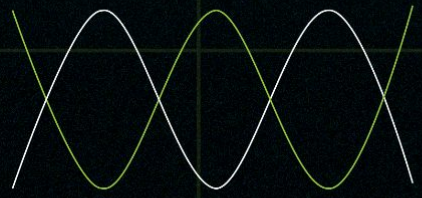
¿Y qué es esto?

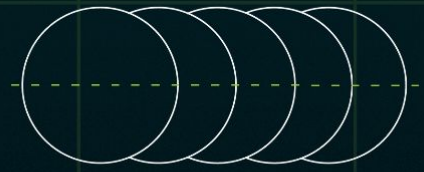


¿Cómo funciona?

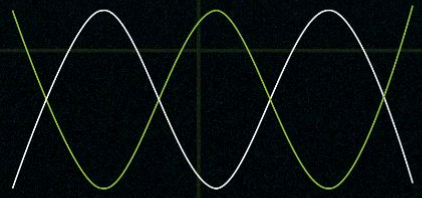


Creando nuestra primera red convolucional



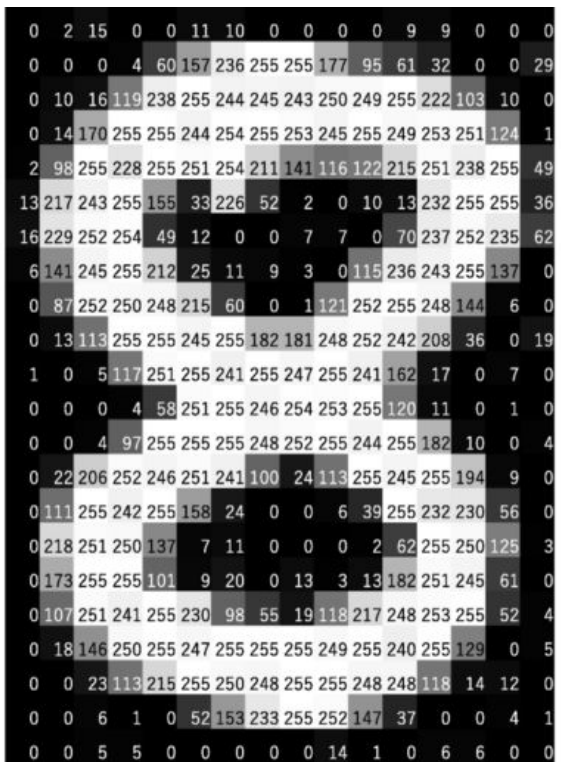


Entrenando nuestra primera red convolucional



Consejos para el manejo de imágenes

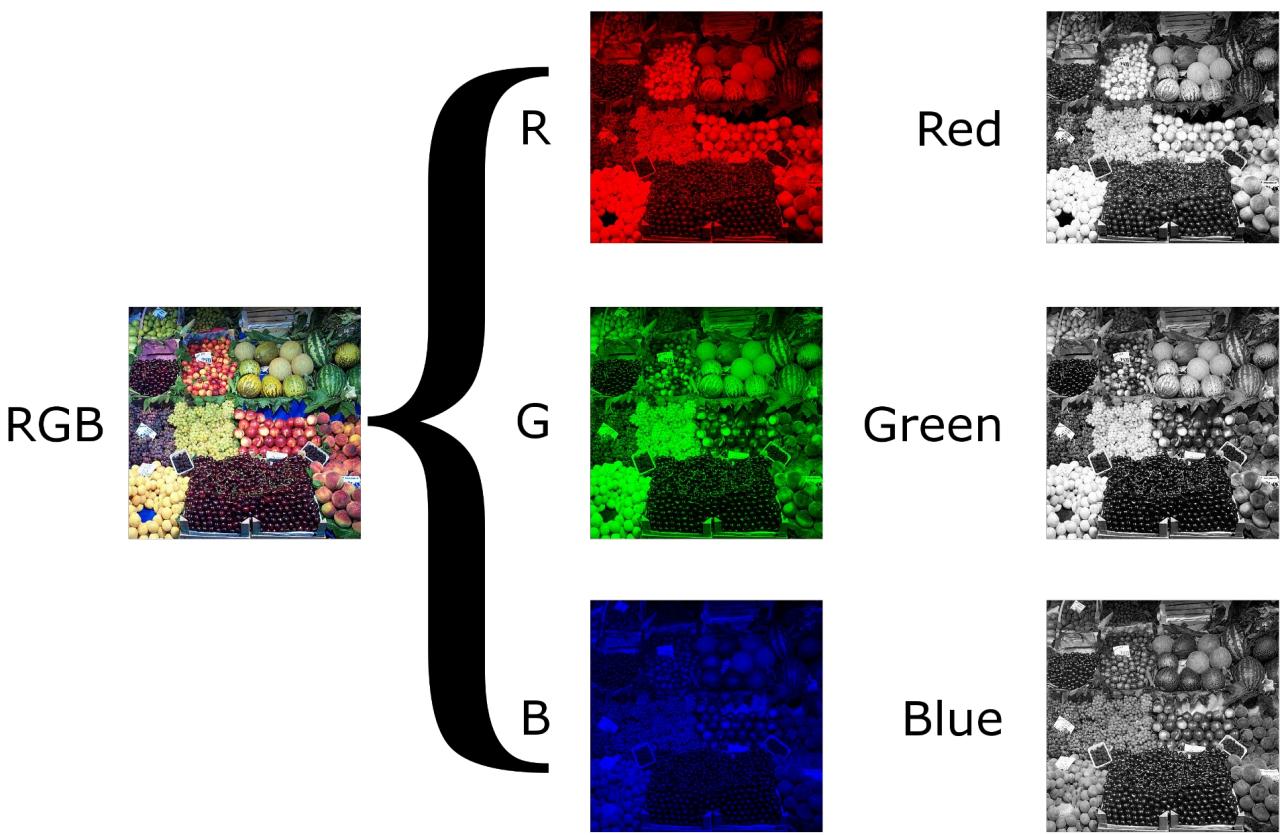
Lo que ve la máquina



0 2 15 0 0 11 10 0 0 0 0 9 9 0 0 0
0 0 0 4 60 157 236 255 255 177 95 61 32 0 0 29
0 10 16 119 238 255 244 245 243 250 249 255 222 103 10 0
0 14 170 255 255 244 254 255 253 245 255 249 253 251 124 1
2 98 255 228 255 251 254 211 141 116 122 215 251 238 255 49
13 217 243 255 155 33 226 52 2 0 10 13 232 255 255 36
16 229 252 254 49 12 0 0 7 7 0 70 237 252 235 62
6 141 245 255 212 25 11 9 3 0 115 236 243 255 137 0
0 87 252 250 248 215 60 0 1 121 252 255 248 144 6 0
0 13 113 255 255 245 255 182 181 248 252 242 208 36 0 19
1 0 5 117 251 255 241 255 247 255 241 162 17 0 7 0
0 0 0 4 58 251 255 246 254 253 255 120 11 0 1 0
0 0 4 97 255 255 255 248 252 255 244 255 182 10 0 4
0 22 206 252 246 251 241 100 24 113 255 245 255 194 9 0
0 111 255 242 255 158 24 0 0 6 39 255 232 230 56 0
0 218 251 250 137 7 11 0 0 0 2 62 255 250 125 3
0 173 255 255 101 9 20 0 13 3 13 182 251 245 61 0
0 107 251 241 255 230 98 55 19 118 217 248 253 255 52 4
0 18 146 250 255 247 255 255 255 249 255 240 255 129 0 5
0 0 23 113 215 255 250 248 255 255 248 248 118 14 12 0
0 0 6 1 0 52 153 233 255 252 147 37 0 0 4 1
0 0 5 5 0 0 0 0 0 14 1 0 6 6 0 0

Yamashita, R., Nishio, M., Do, R.K.G. *et al.* Convolutional neural networks: an overview and application in radiology. *Insights Imaging* **9,** 611–629 (2018). https://doi.org/10.1007/s13244-018-0639-9

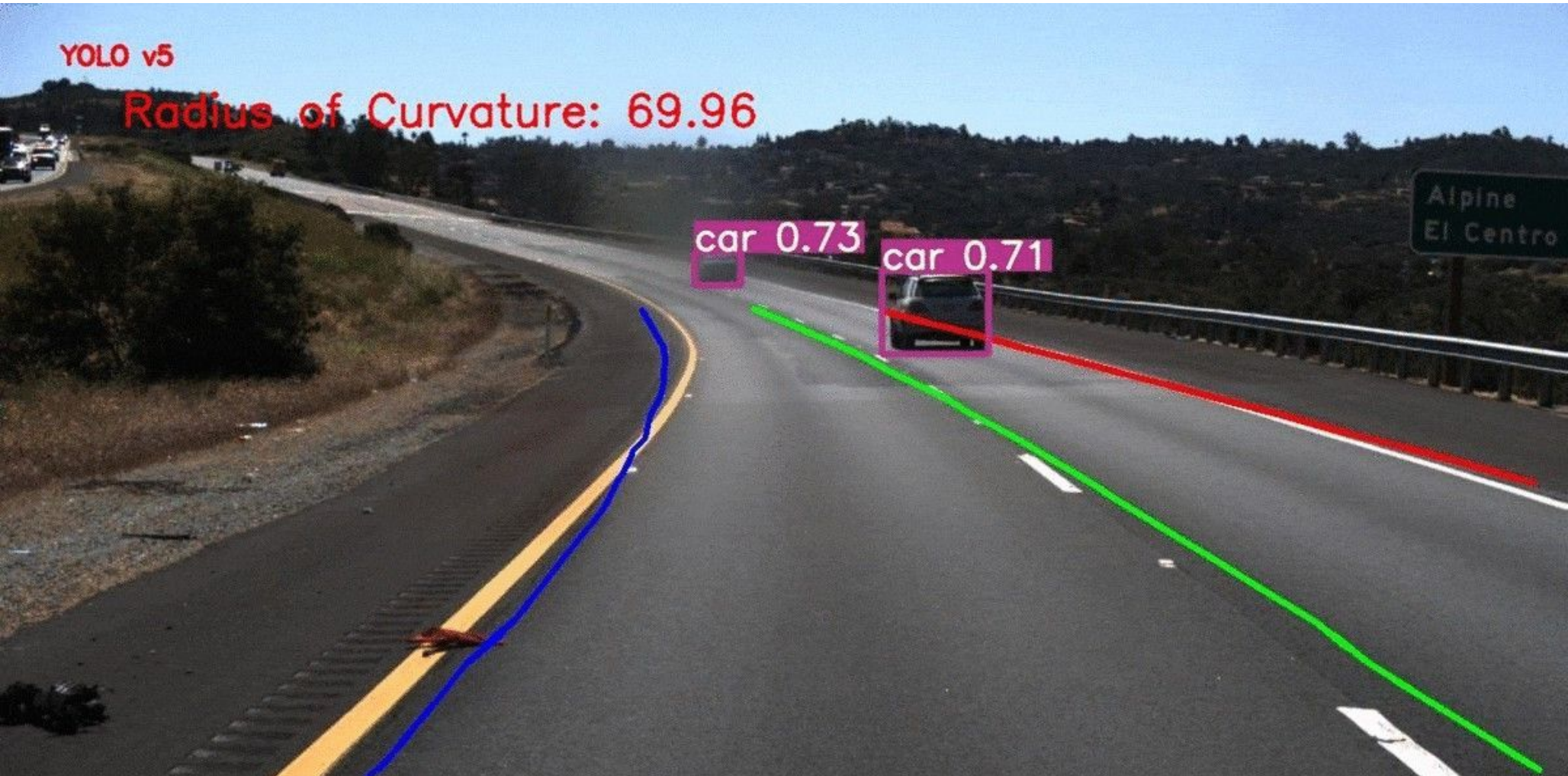
¿Y las imágenes a color?



Wikipedia contributors. (2021, September 10). Grayscale. In *Wikipedia, The Free Encyclopedia*. Retrieved 22:08, October 11, 2021, from https://en.wikipedia.org/w/index.php?title=Grayscale&oldid=1043483162

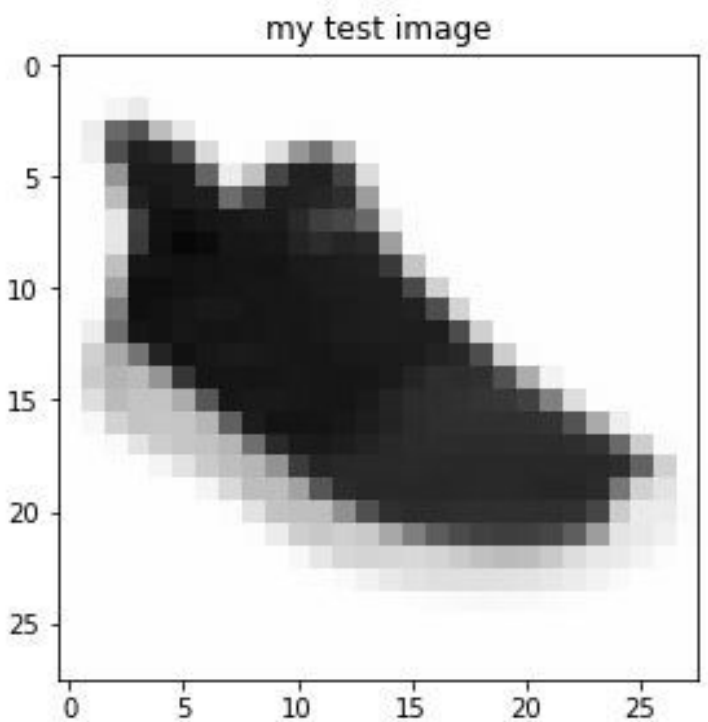
Trabajar con escala de grises

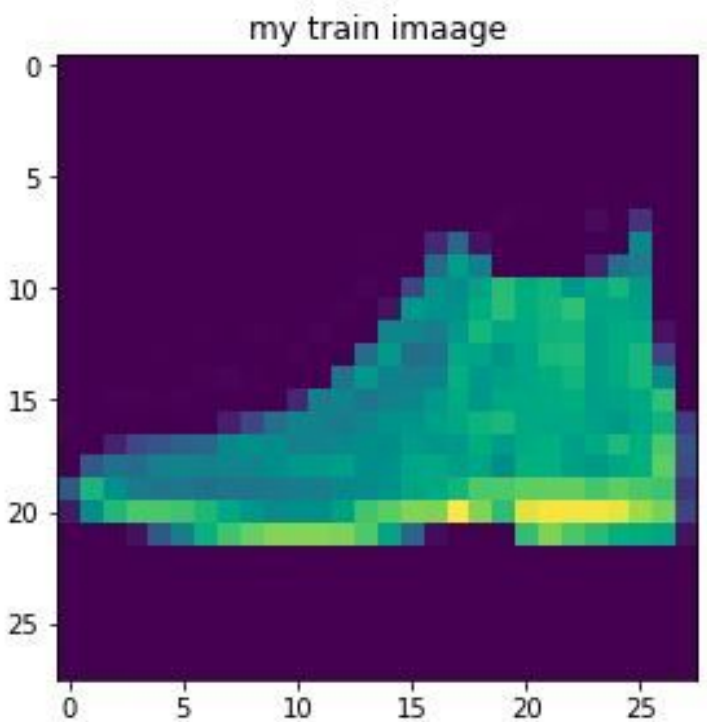
¿Cuándo importa el color?



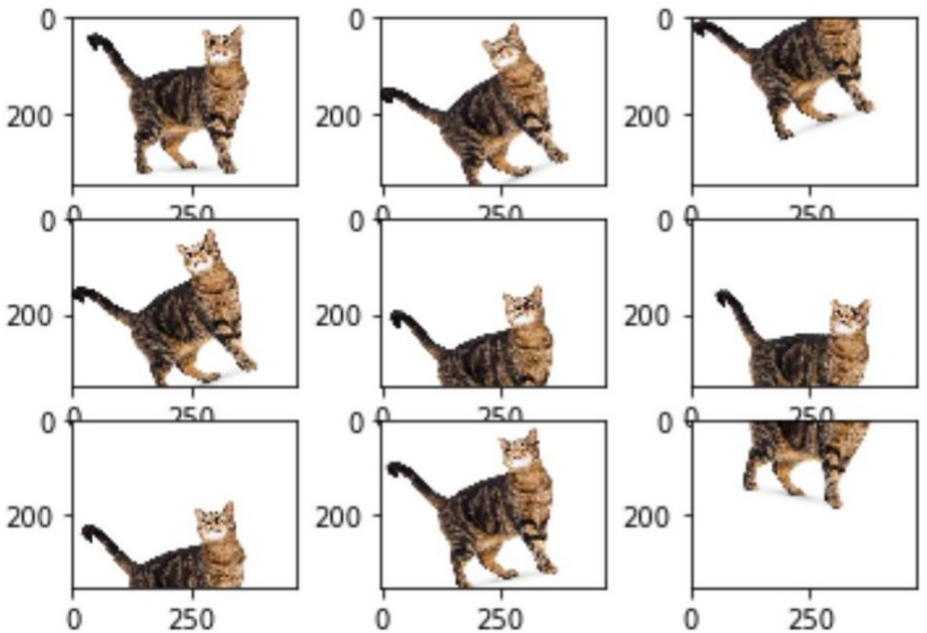
Dimensiones definidas

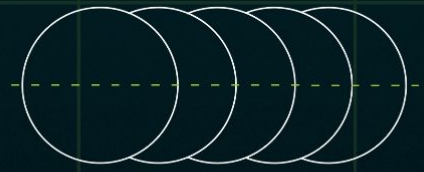
```
my test image size: (28, 28, 3)
train image size: (28, 28)
```



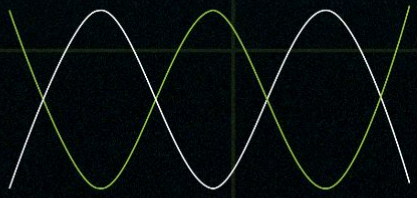


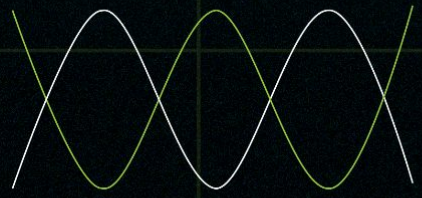
Data augmentation

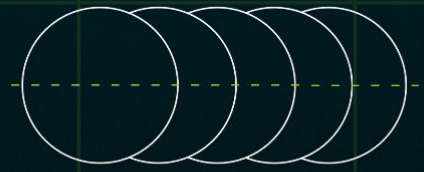




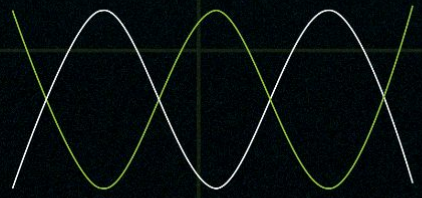
Manejo de imágenes con Python







Kernel en redes neuronales



El kernel

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	4	62	146	182	254	254	181	176	139	15	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	34	186	253	217	208	136	136	136	166	232	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	61	242	208	111	3	0	0	0	0	0	18	32	107	43	0	0	0	0	0	0
0	0	0	0	0	0	0	0	156	242	23	0	0	0	0	0	0	0	13	191	181	6	0	0	0	0	0	0
0	0	0	0	0	0	0	0	121	255	98	3	0	0	0	0	0	8	194	225	12	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	169	253	120	3	0	0	0	0	128	247	51	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	3	111	244	169	19	0	14	131	249	117	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	59	241	235	72	142	229	66	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	25	218	254	231	36	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	133	253	221	33	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	19	237	111	196	217	19	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	138	0	23	193	204	18	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	96	224	0	0	0	25	218	169	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	138	0	0	0	0	86	253	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	3	162	214	11	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	0	118	253	68	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	185	157	0	0	0	0	0	40	254	98	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	50	244	61	0	0	0	0	112	244	58	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	251	142	59	83	167	244	111	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	6	133	253	253	253	169	61	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

El kernel

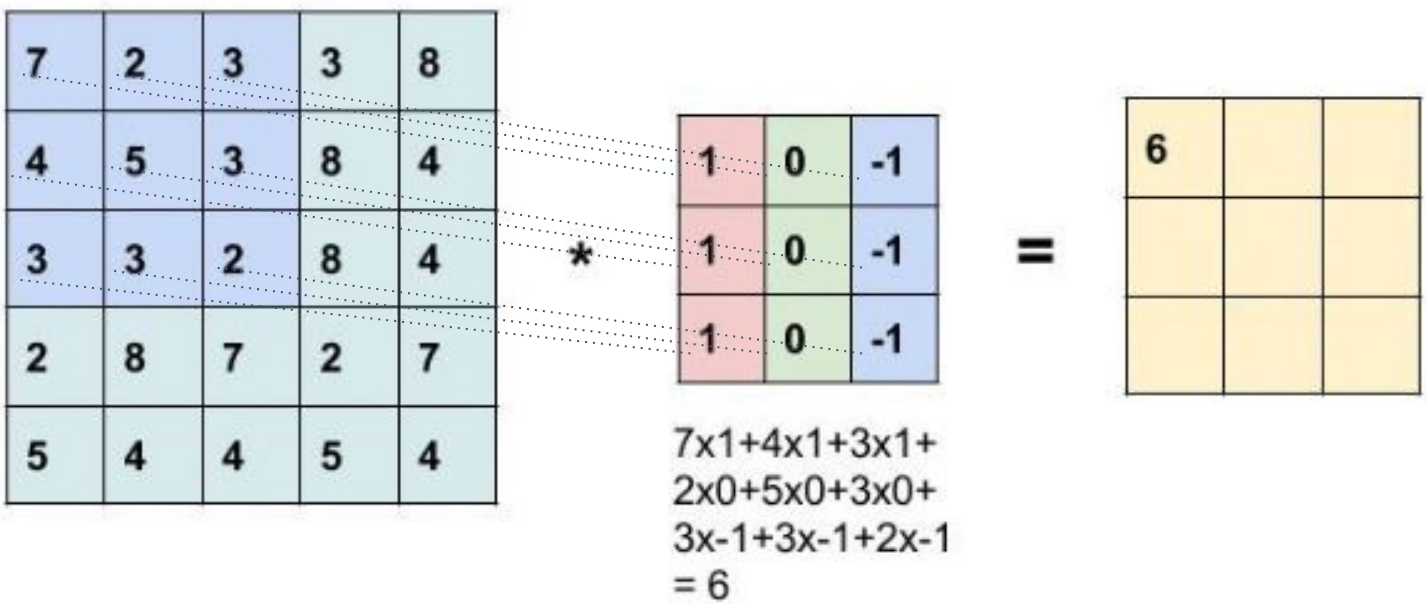
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	4	62	146	182	254	254	181	176	139	15	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	34	186	253	217	208	136	136	136	166	232	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	61	242	208	111	3	0	0	0	0	0	18	32	107	43	0	0	0	0	0	0
0	0	0	0	0	0	0	0	156	242	23	0	0	0	0	0	0	0	13	191	181	6	0	0	0	0	0	0
0	0	0	0	0	0	0	0	121	255	98	3	0	0	0	0	0	8	194	225	12	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	169	253	120	3	0	0	0	0	128	247	51	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	3	111	244	169	19	0	14	131	249	117	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	59	241	235	72	142	229	66	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	25	218	254	231	36	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	133	253	221	33	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	19	237	111	196	217	19	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	138	0	23	193	204	18	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	96	224	0	0	0	25	218	169	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	138	0	0	0	0	86	253	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	3	162	214	11	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	0	118	253	68	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	185	157	0	0	0	0	0	40	254	98	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	50	244	61	0	0	0	0	112	244	58	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	251	142	59	83	167	244	111	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	6	133	253	253	253	169	61	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Kernel / Filter

1	1	1
0	0	0
-1	-1	-1

¿Cómo funciona el kernel?

Convolución



Detectar bordes verticales

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

6 x 6

*

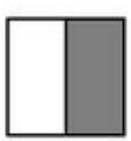
1	0	-1
1	0	-1
1	0	-1

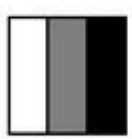
3 x 3

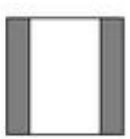
=

-0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

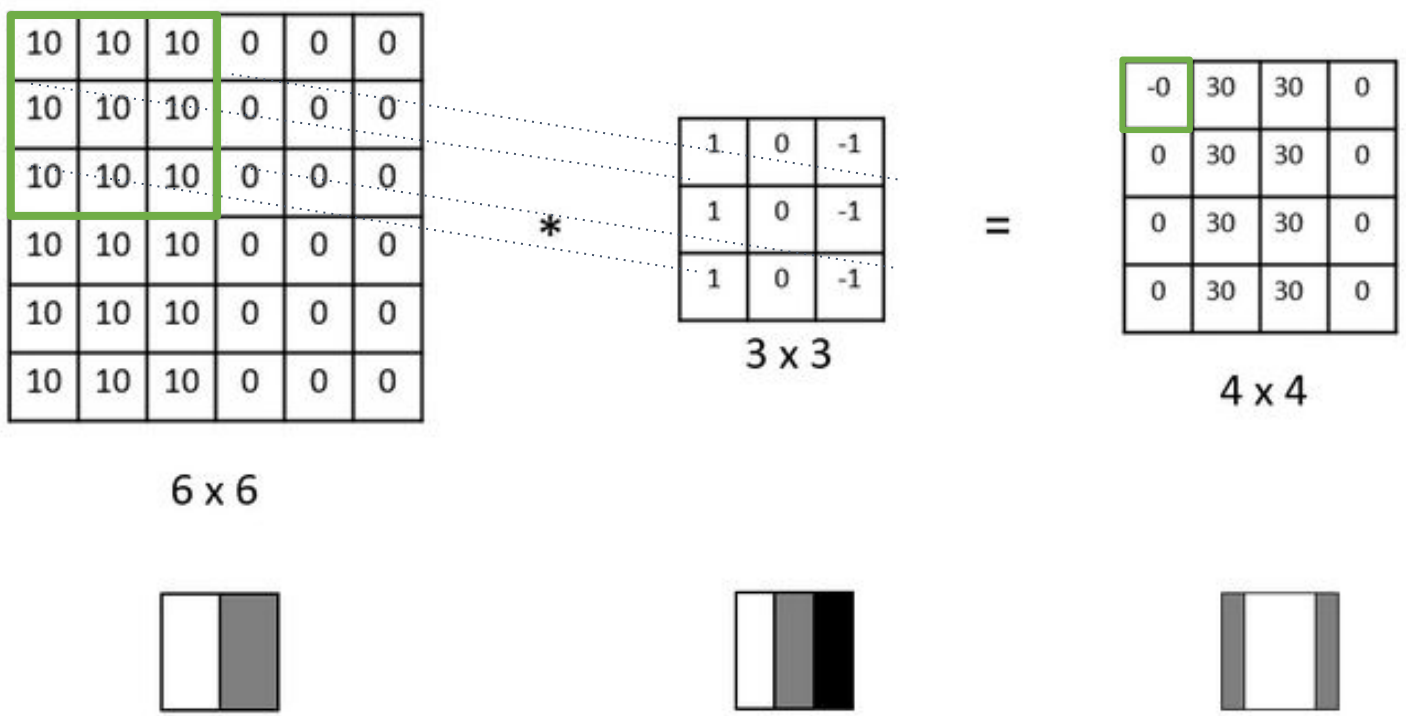
4 x 4





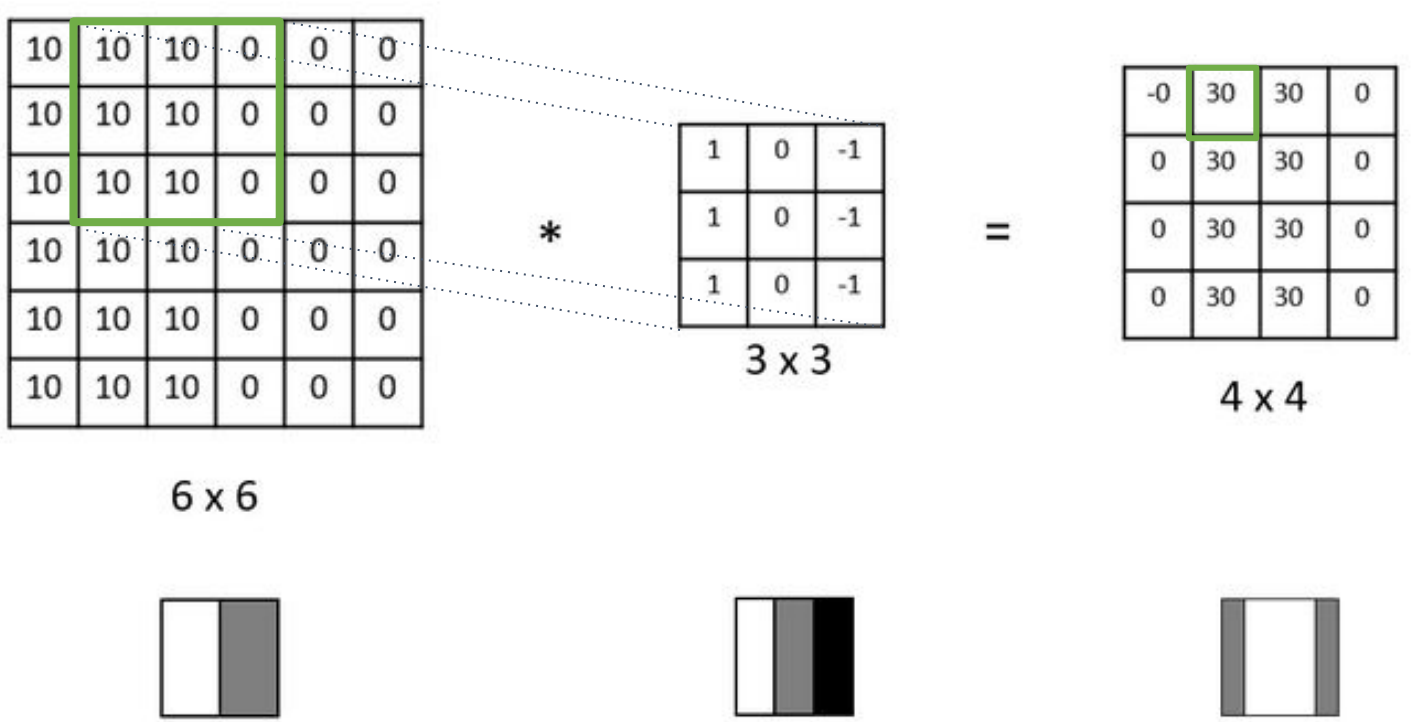


Detectar bordes verticales



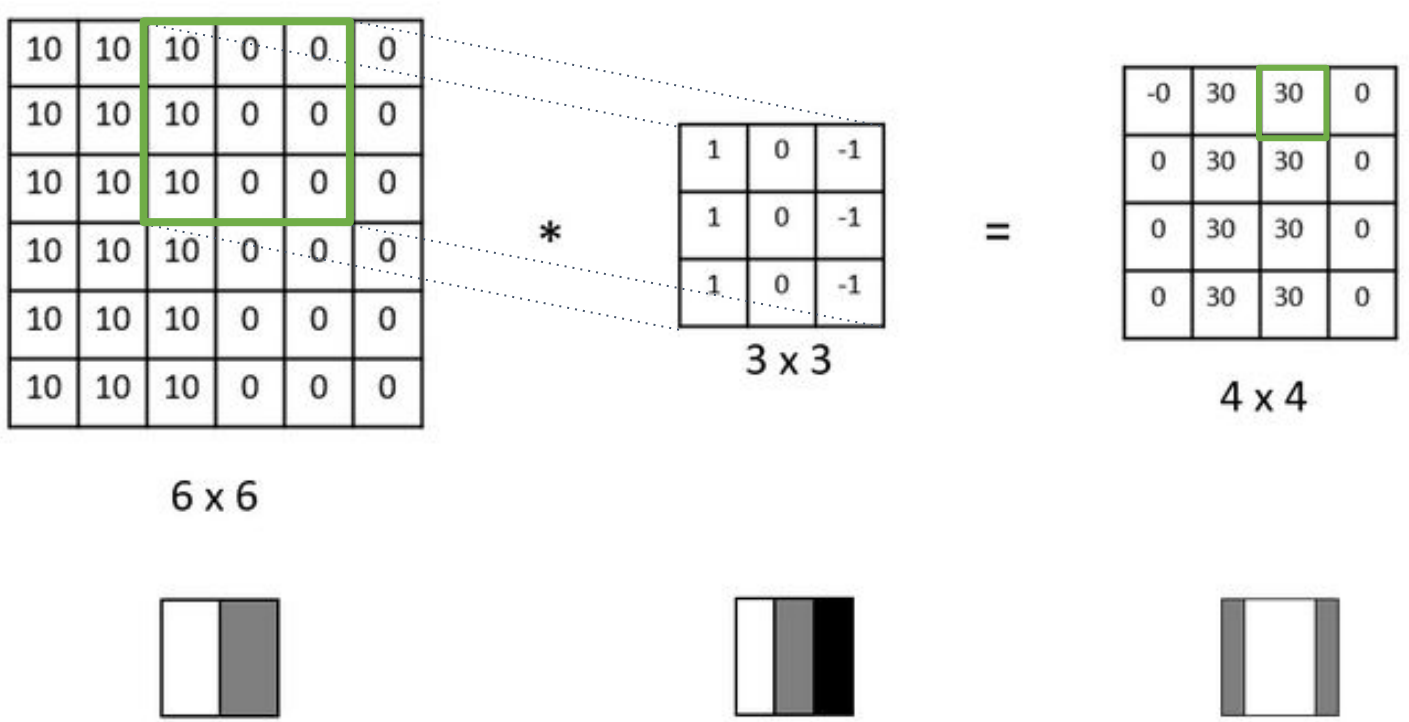
(10*1) + (10*0) + (10*-1) +(10*1) + (10*0) + (10*-1) + (10*1) + (10*0) + (10*-1) = 0

Detectar bordes verticales

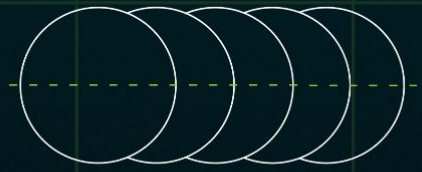


$(10^*1) + (10^*0) + (0^*-1) + (10^*1) + (10^*0) + (0^*-1) + (10^*1) + (10^*0) + (0-1) = \mathbf{30}$

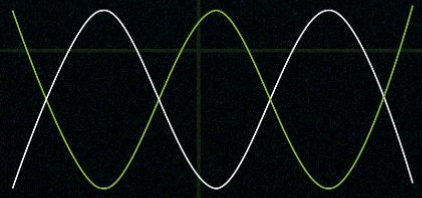
Detectar bordes verticales

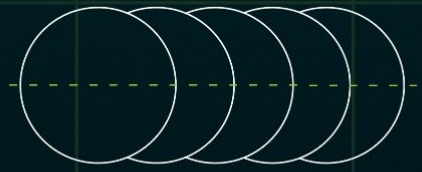


(10*1) + (0*0) + (0*-1) +(10*1) + (0*0) + (0*-1) + (10*1) + (0*0) + (0-1) = **30**

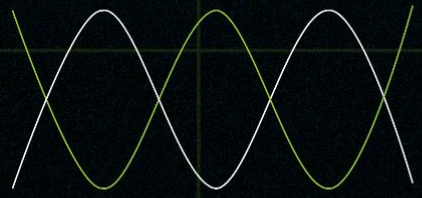


El kernel en acción





Padding y strides



Convolución

Padding

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

$6 \times 6 \rightarrow 8 \times 8$

*

1	0	-1
1	0	-1
1	0	-1

3×3

=

-10	-13	1			
-9	3	0			

6×6

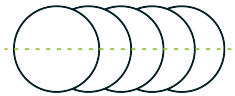
Padding

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

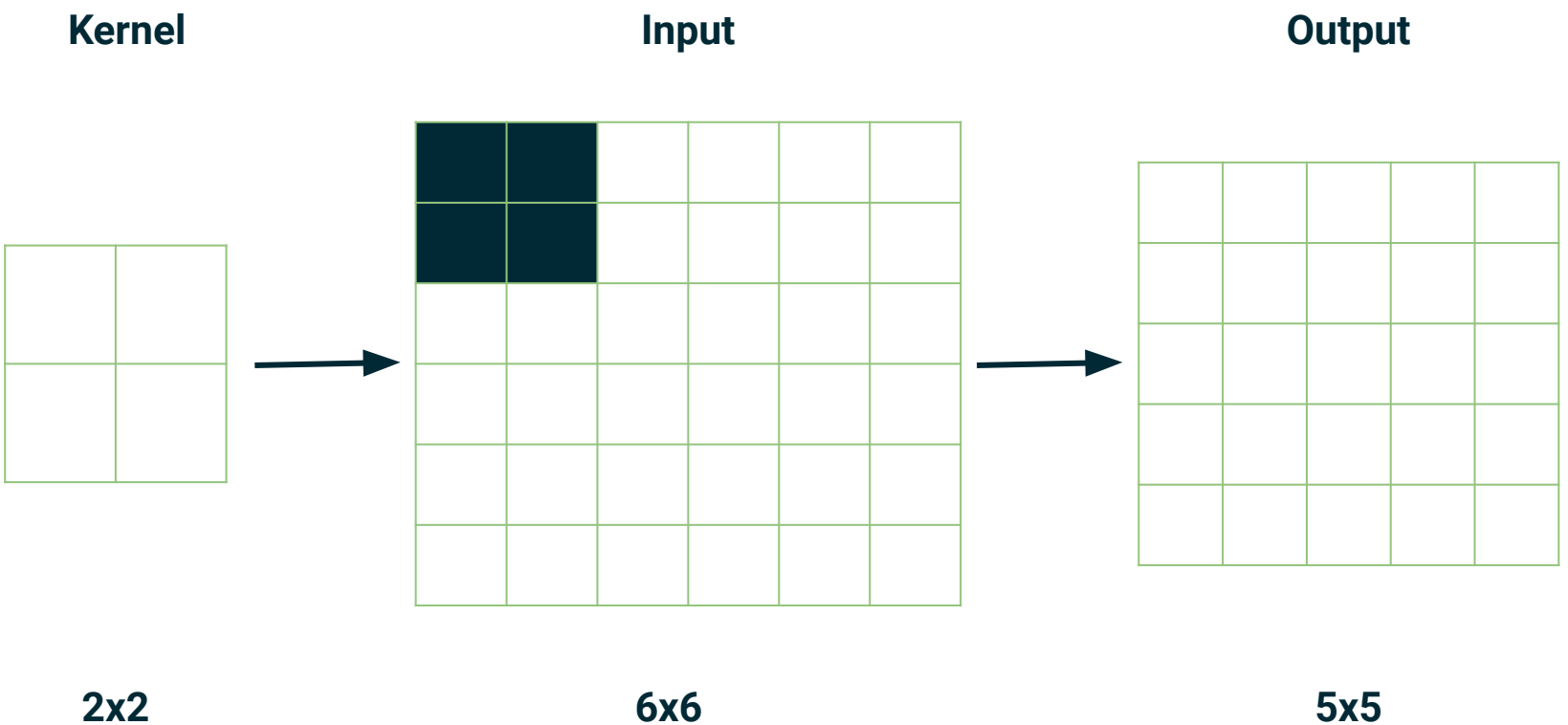
1

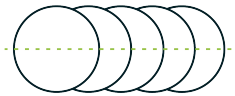
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

2

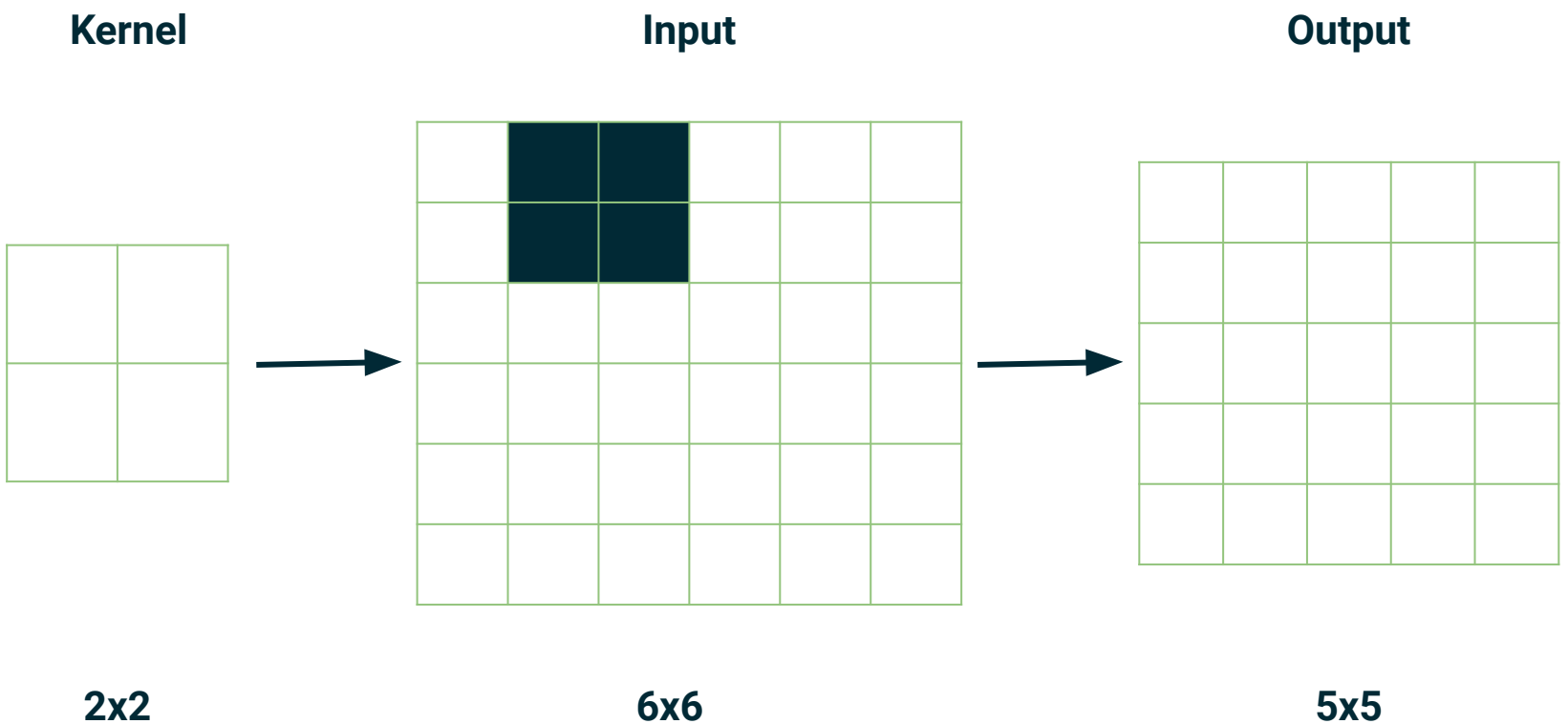


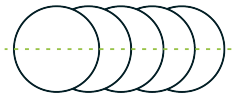
Strides = 1



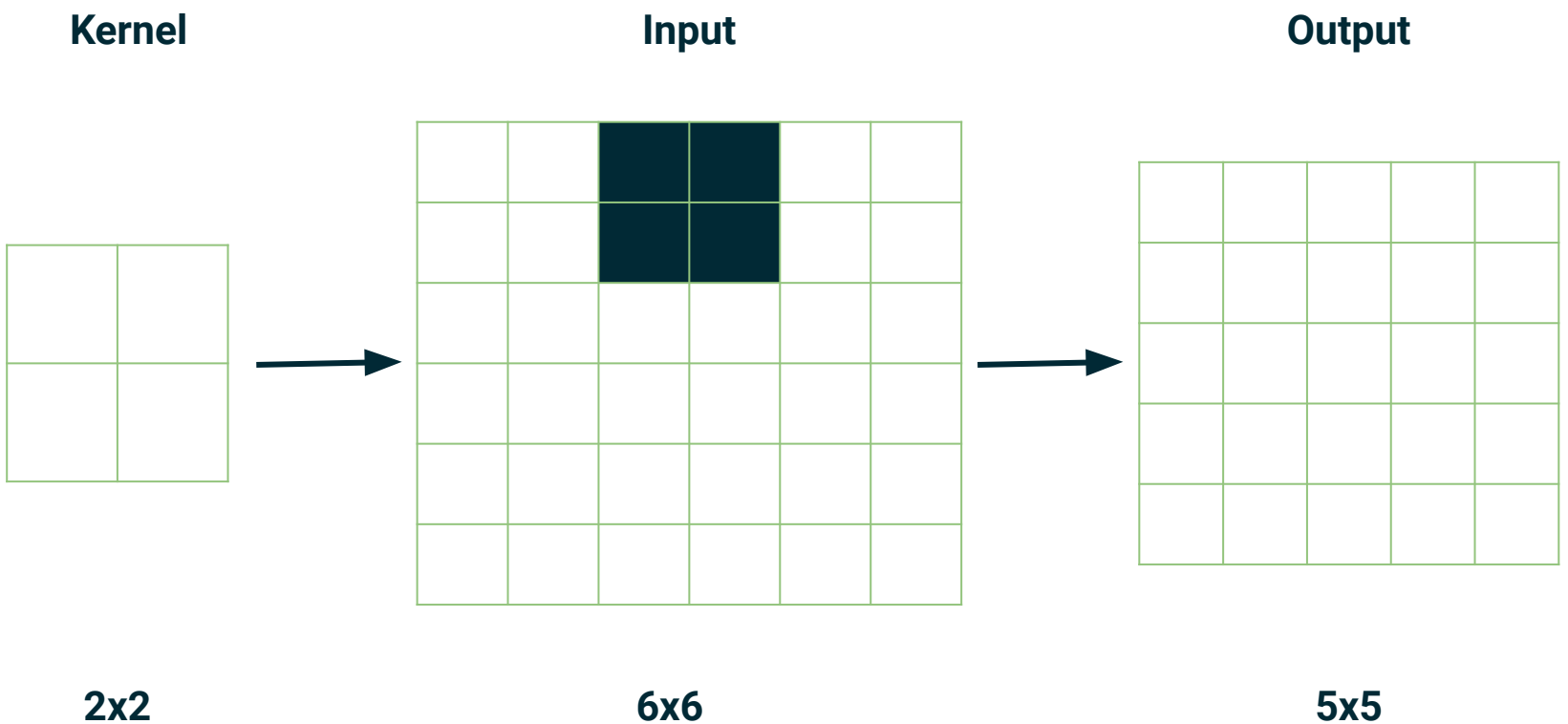


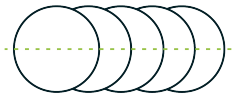
Strides = 1



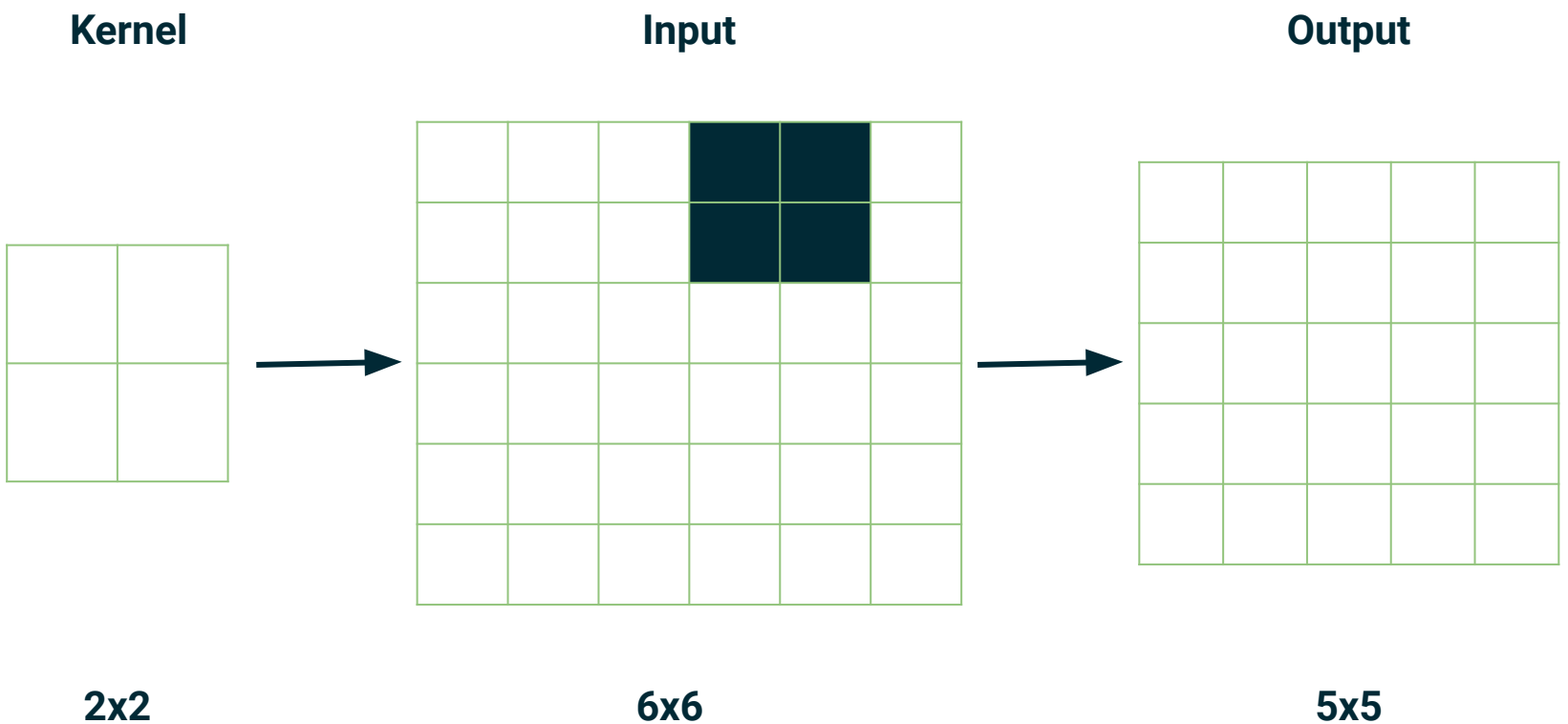


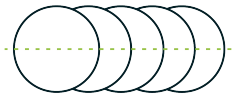
Strides = 1



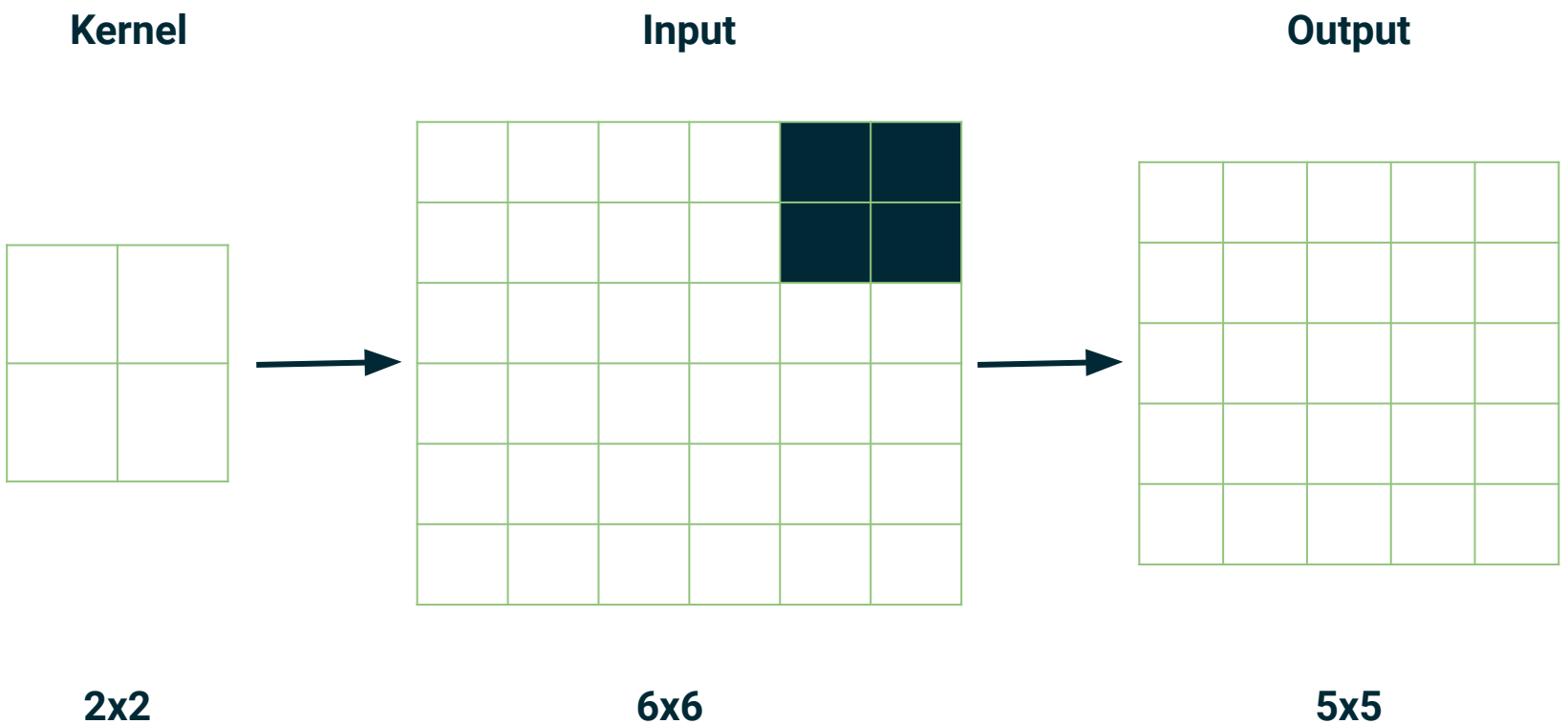


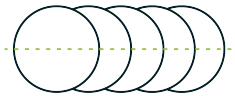
Strides = 1



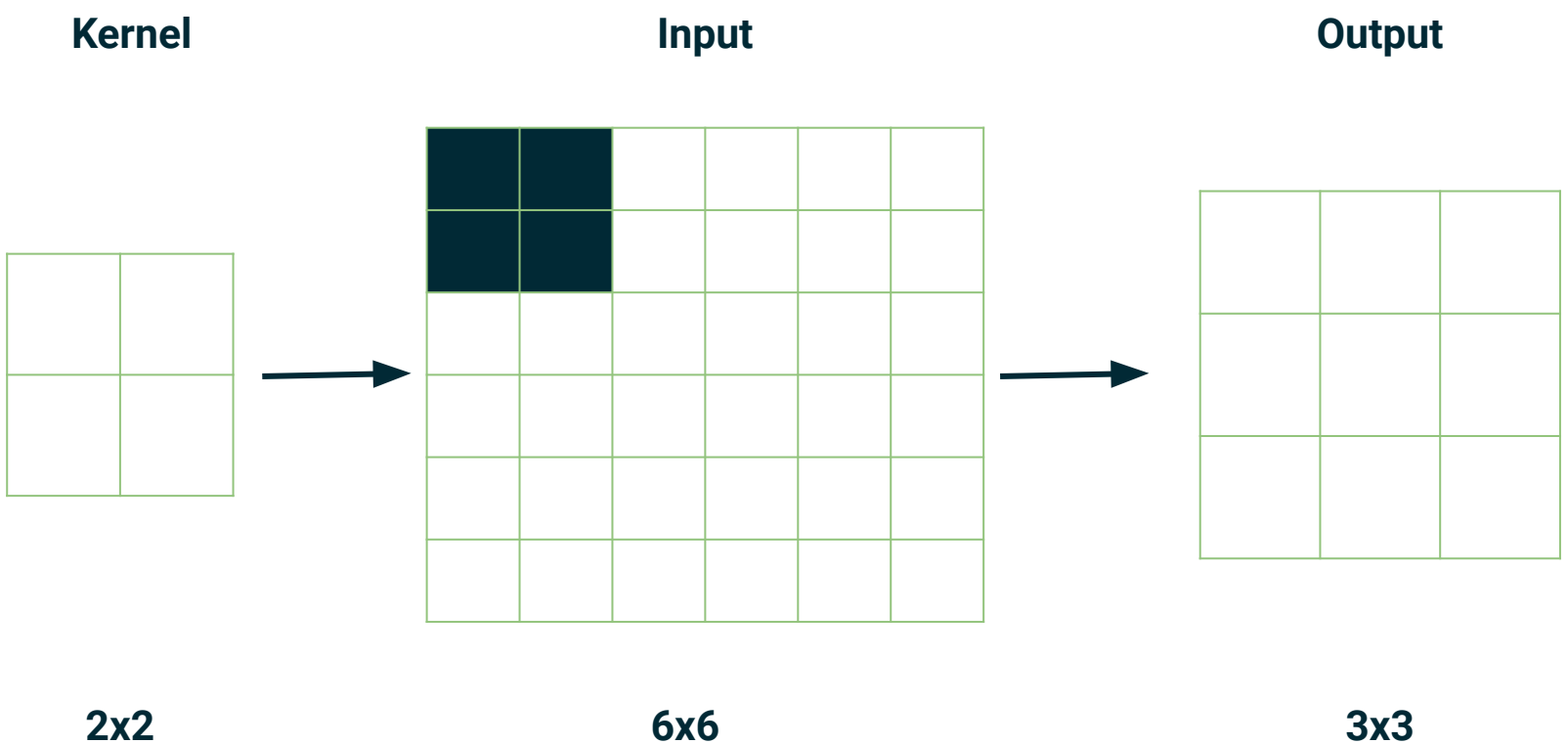


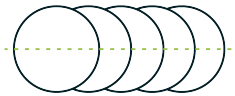
Strides = 1



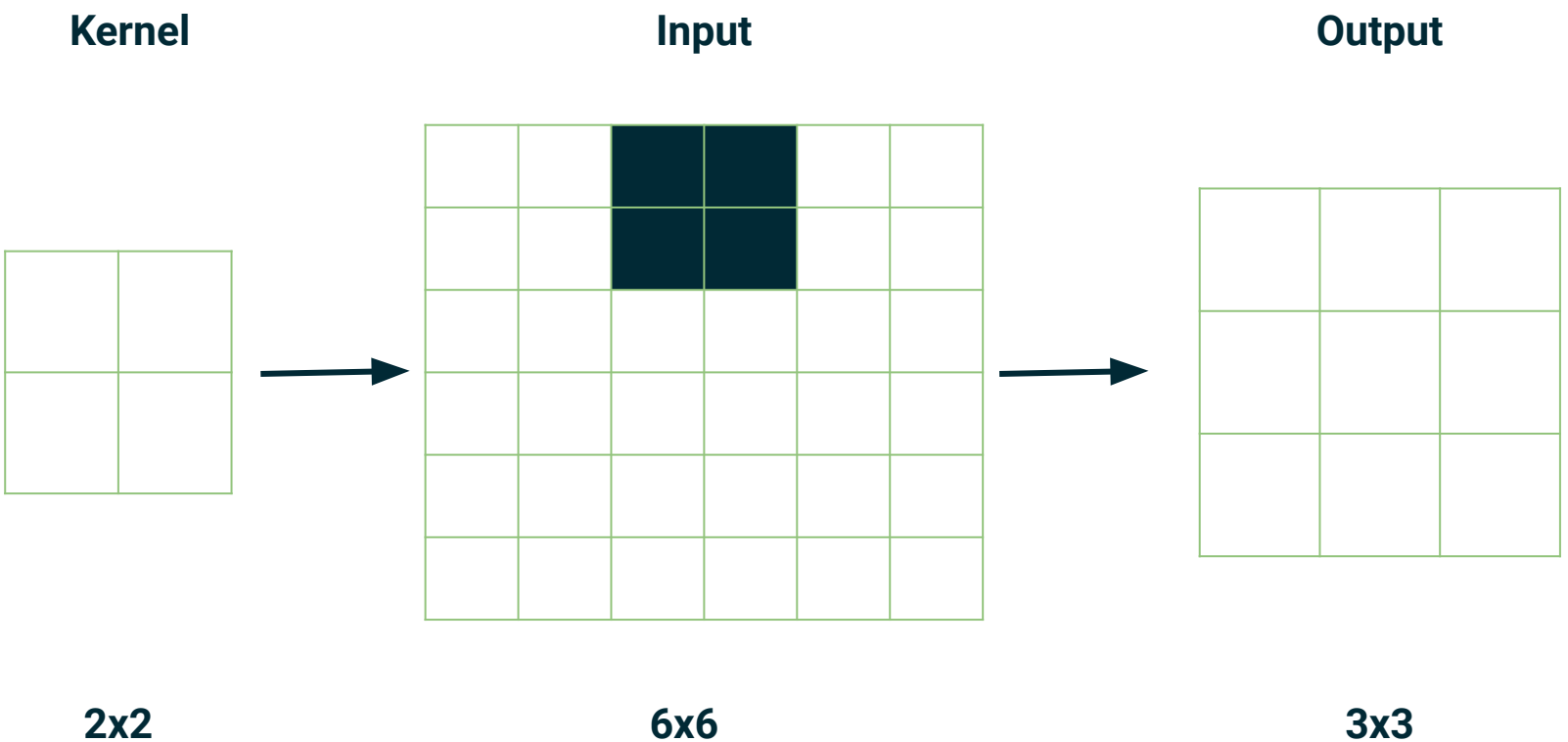


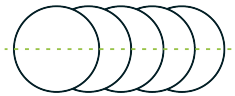
Strides = 2



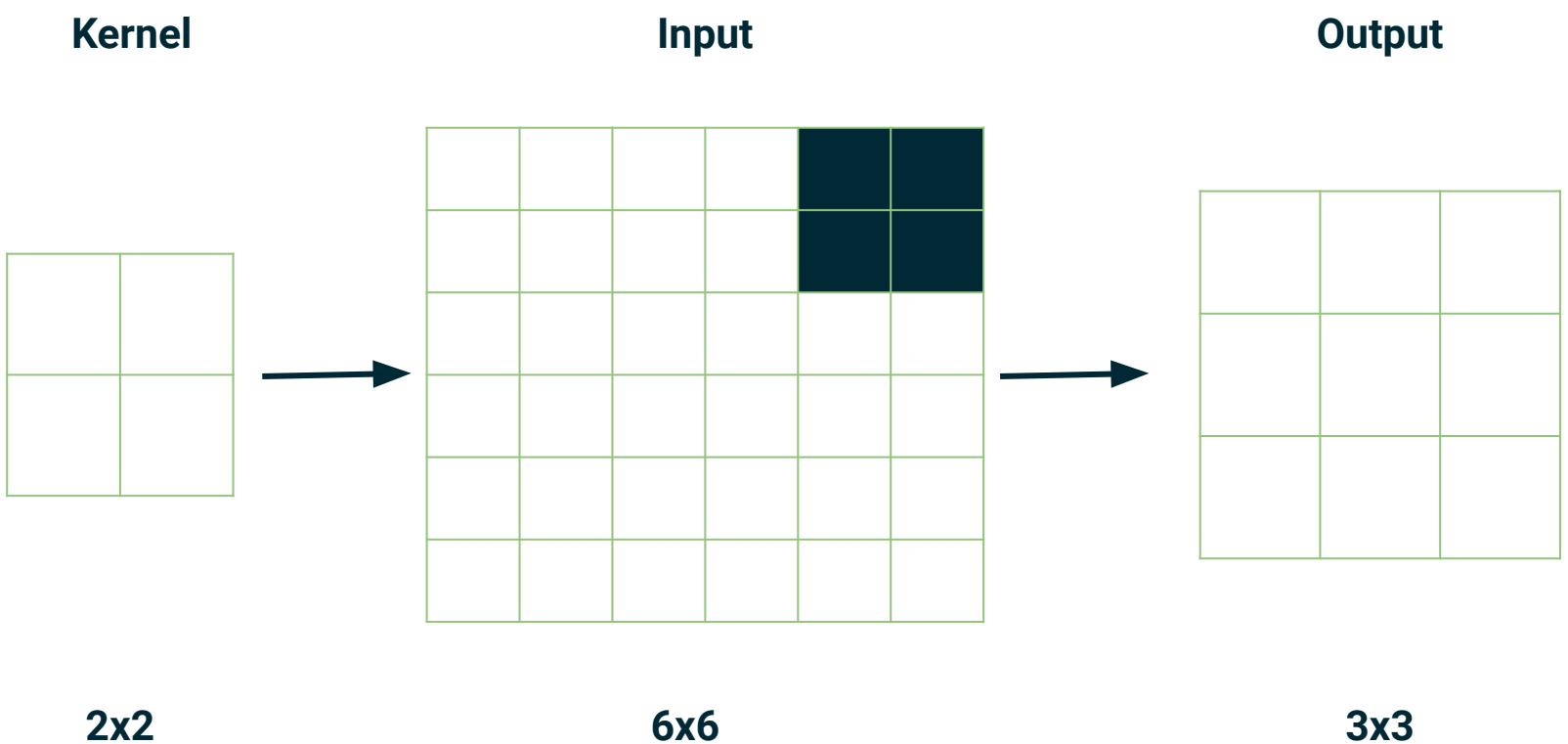


Strides = 2

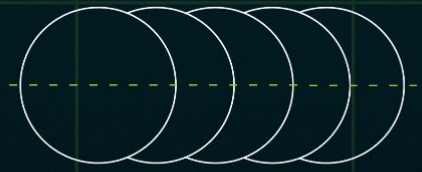




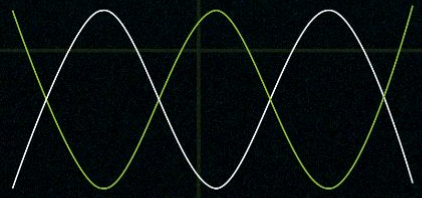
Strides = 2



```
tf.keras.layers.Conv2D(
    filters,
    kernel_size,
    strides=(1, 1),
    padding=["valid","same"])
```



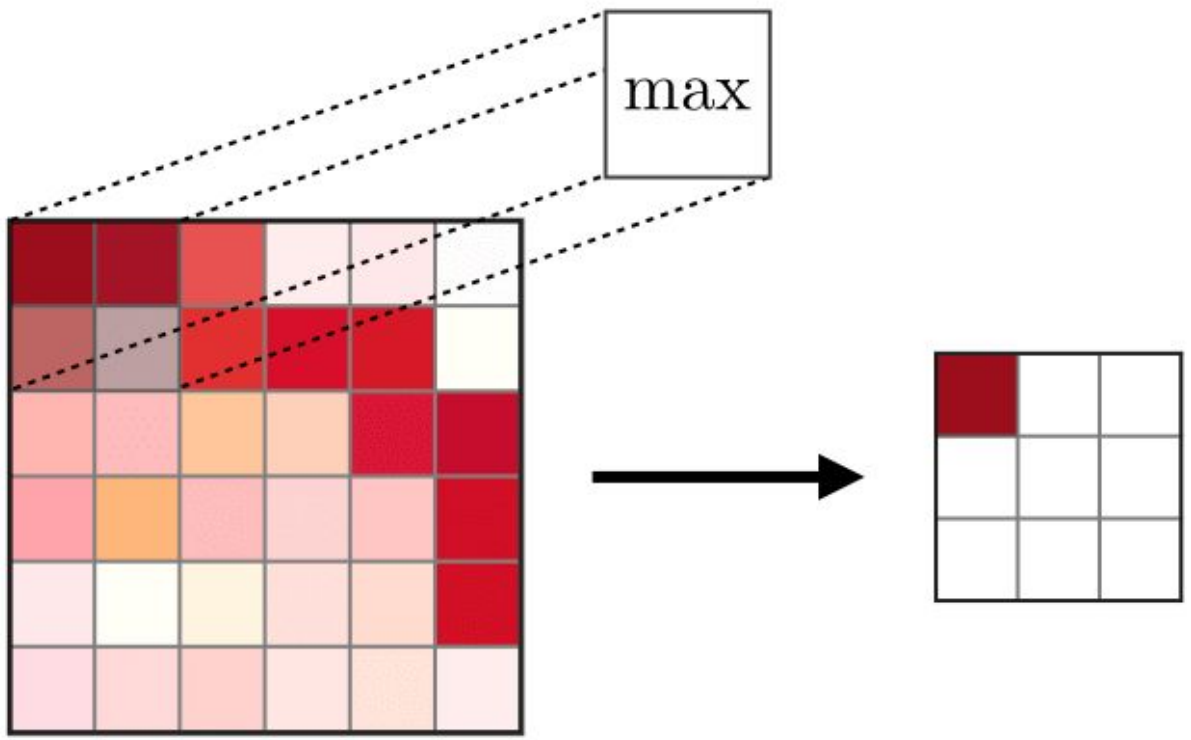
Capa de pooling

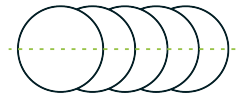


El problema

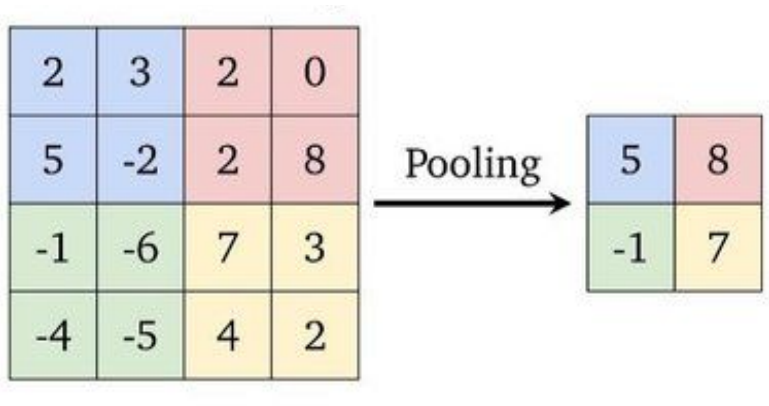
Múltiples filtros en cada capa hacen una red compleja con muchos parámetros.

Pooling

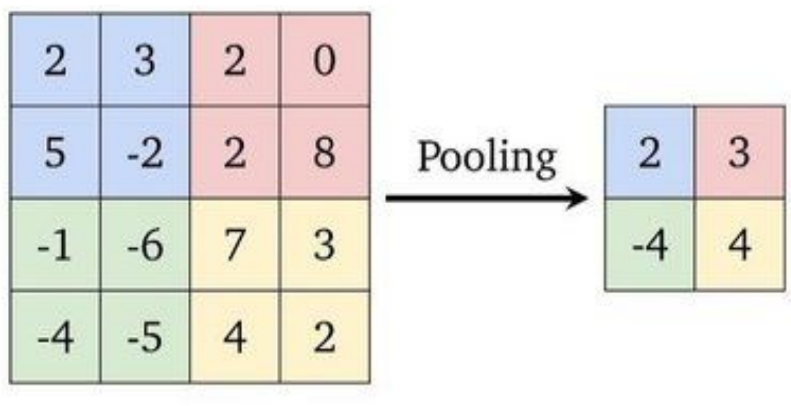




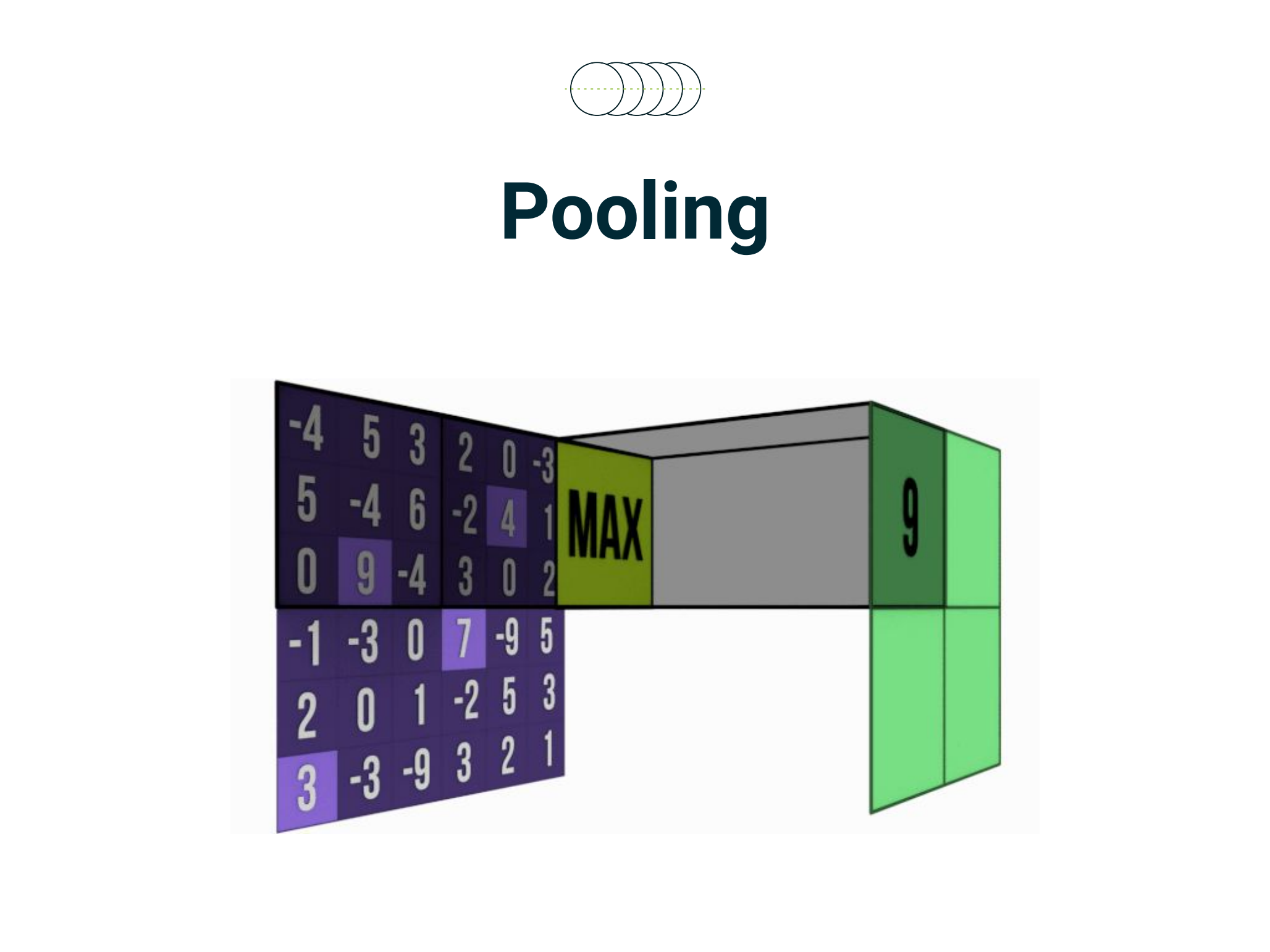
Pooling



Max

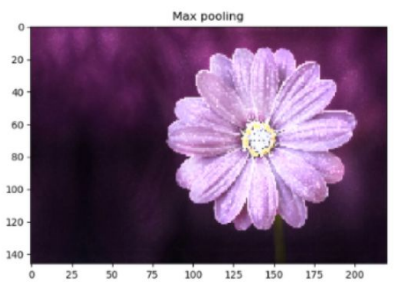


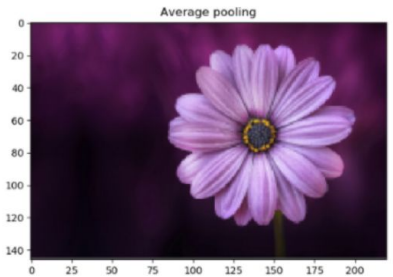
Average

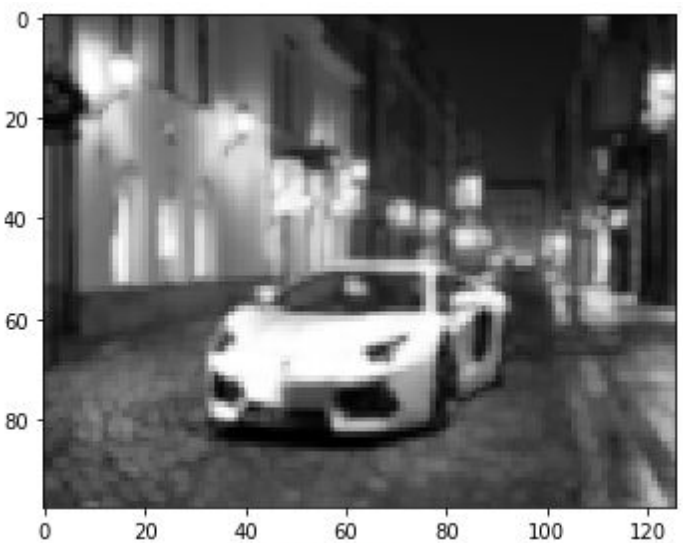


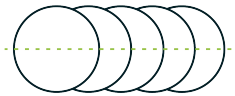
Pooling

Pooling







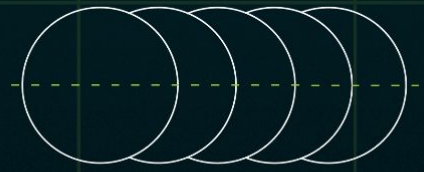


Pooling

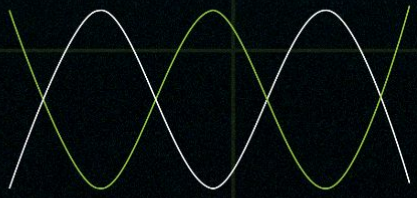
Parámetros = Filtros * tamaño del kernel * profundidad de la capa anterior + Filtros (Bias)1

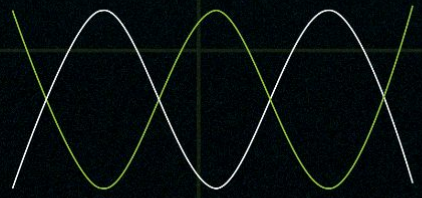
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 16)        208
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 16, 16, 32)        2080
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 32)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 8, 8, 64)          8256
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
dropout (Dropout)            (None, 4, 4, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 500)               512500
_________________________________________________________________
dropout_1 (Dropout)          (None, 500)               0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                5010
=================================================================
Total params: 528,054
Trainable params: 528,054
Non-trainable params: 0
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 32, 32, 16)        208
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 32, 32, 32)        2080
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 32, 32, 64)        8256
_________________________________________________________________
dropout_2 (Dropout)          (None, 32, 32, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 65536)             0
_________________________________________________________________
dense_2 (Dense)              (None, 500)               32768500
_________________________________________________________________
dropout_3 (Dropout)          (None, 500)               0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                5010
=================================================================
Total params: 32,784,054
Trainable params: 32,784,054
Non-trainable params: 0
```

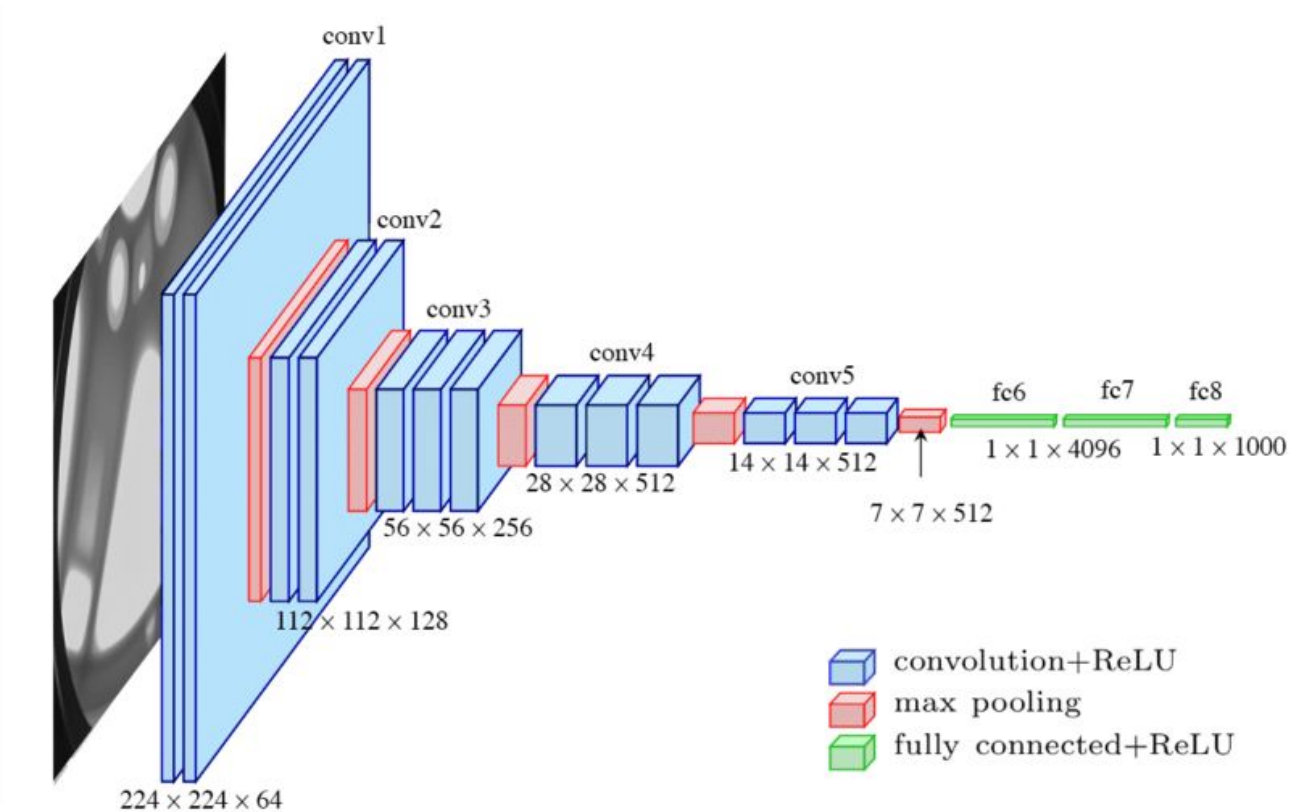


Arquitectura de redes convolucionales





CNN

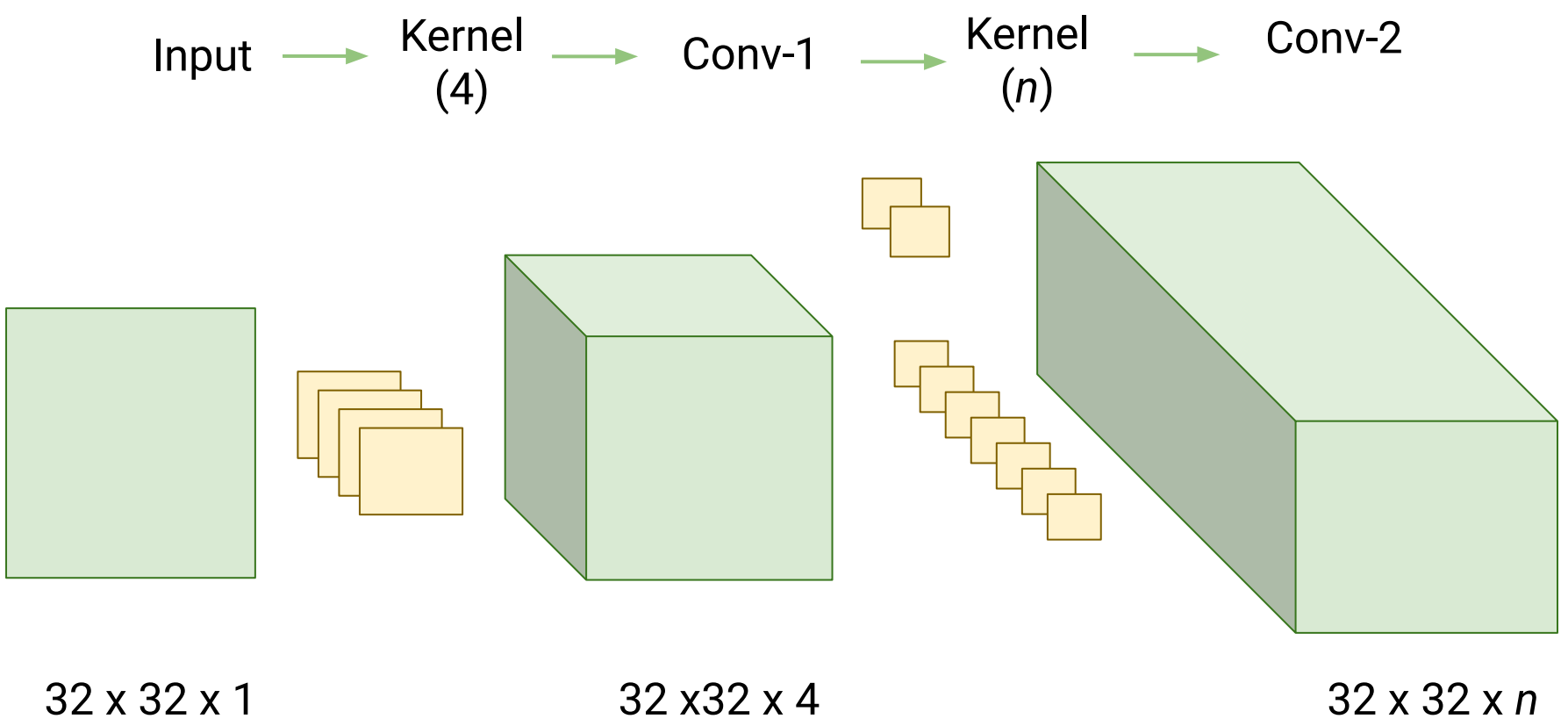


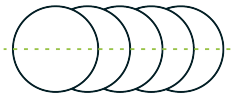
Referencias. Ferguson, Max & ak, Ronay & Lee, Yung-Tsun & Law, Kincho. (2017). Automatic localization of casting defects with convolutional neural networks. 1726-1735. 10.1109/BigData.2017.8258115.

¿Complicado?

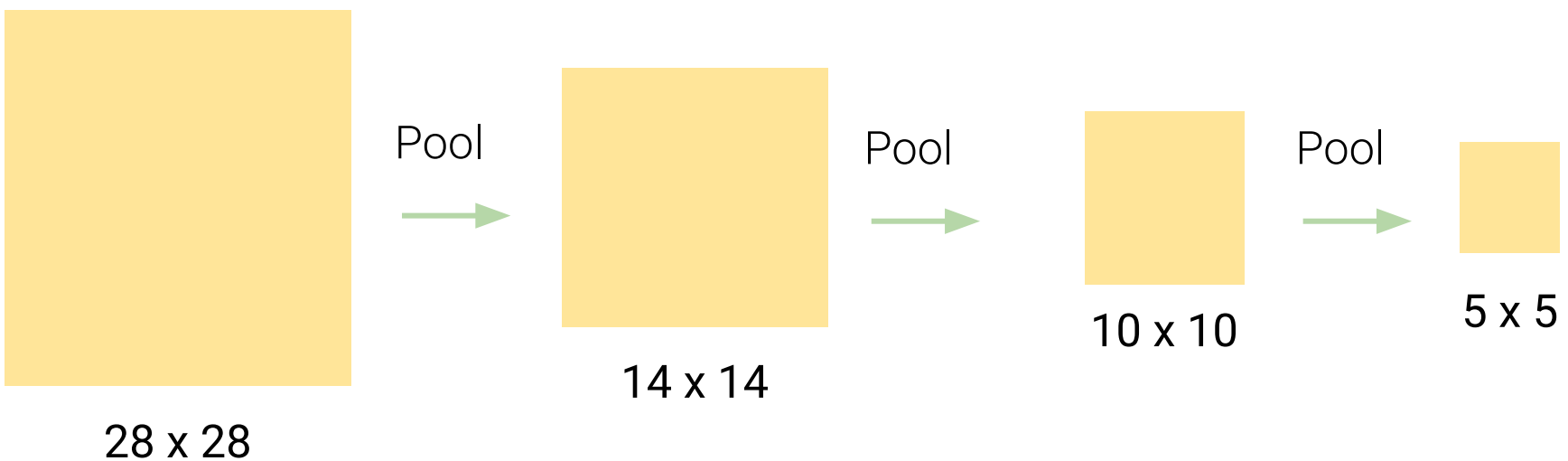
Veámoslo un poco más en detalle.

CNN - Convolución

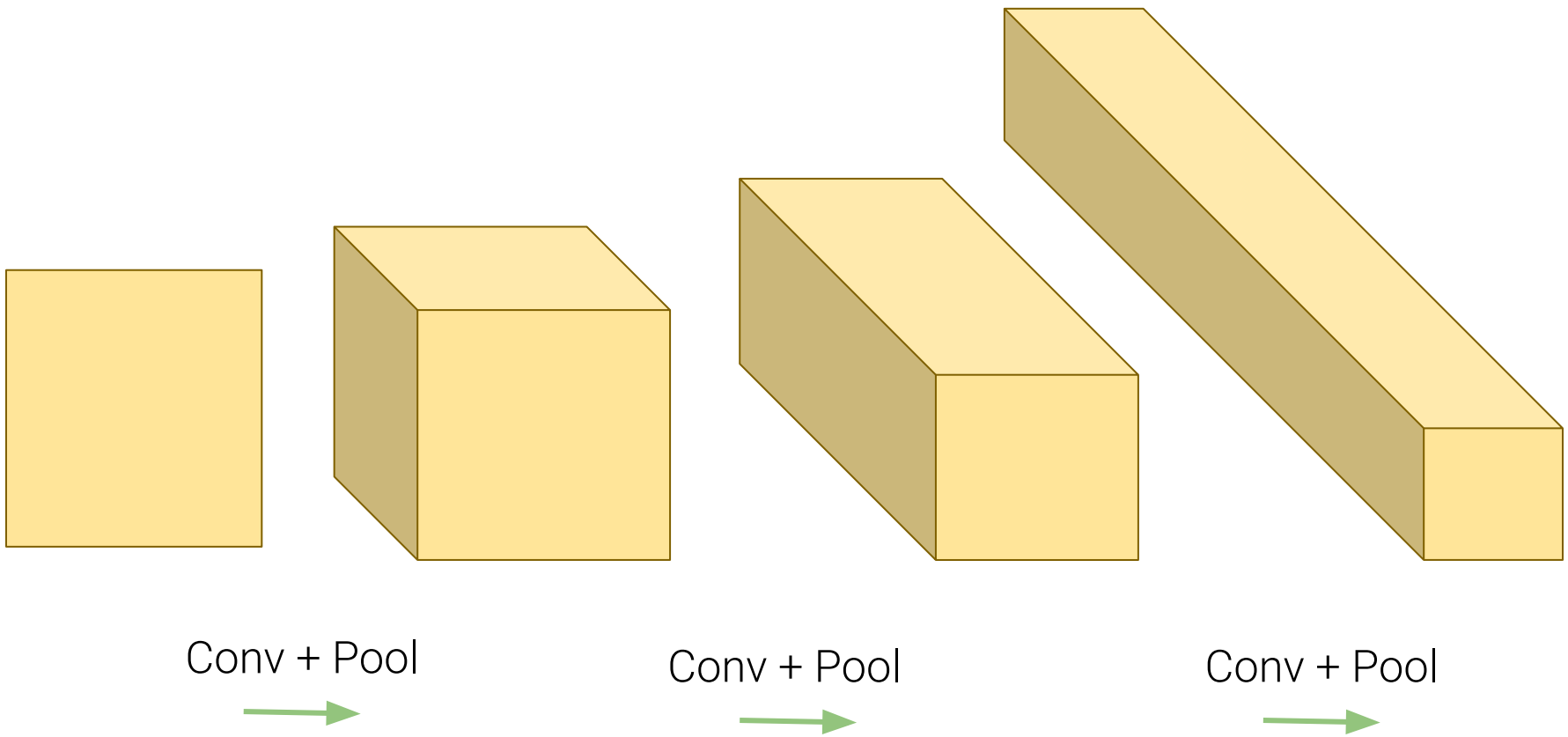


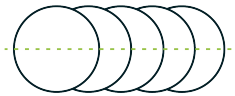


CNN - Pooling

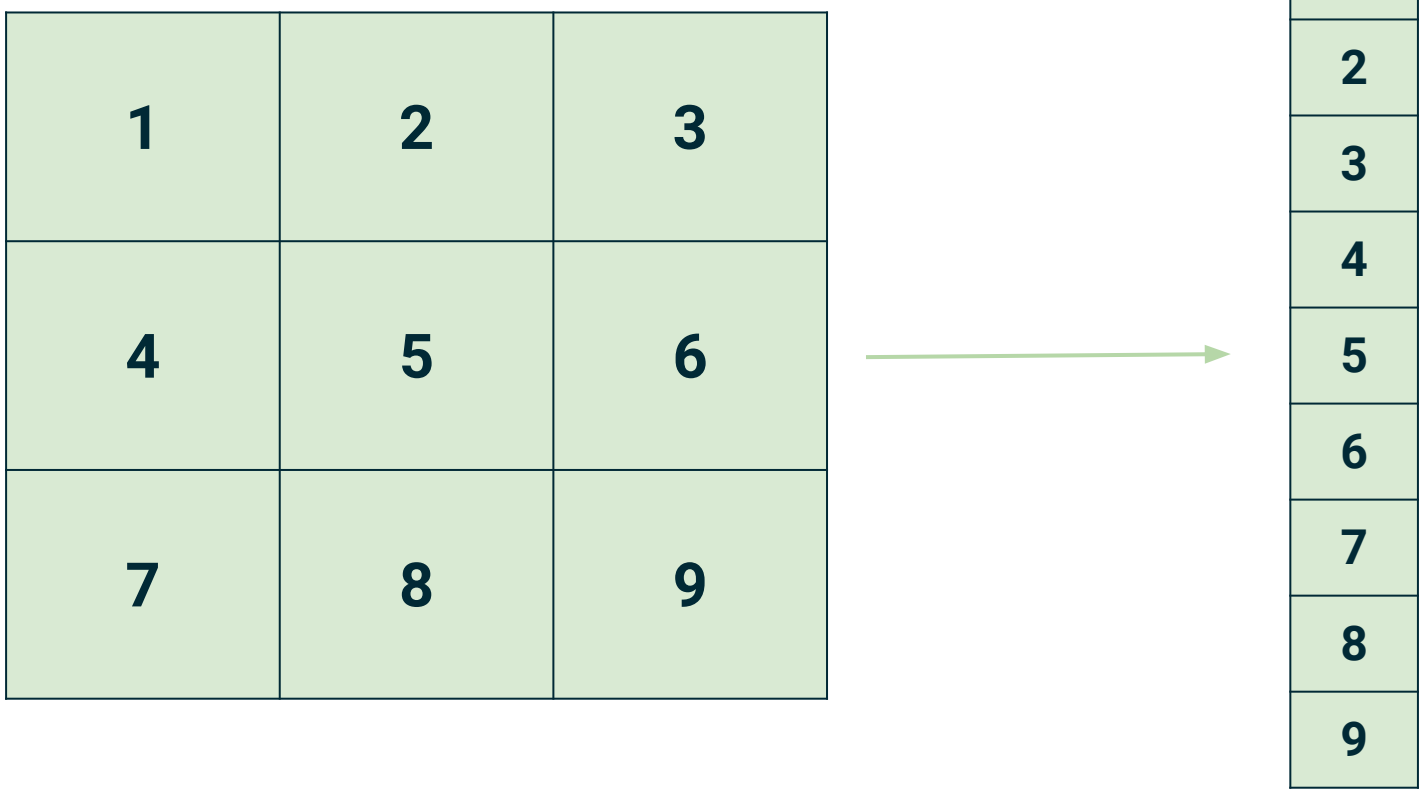


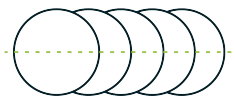
CNN - Todo junto





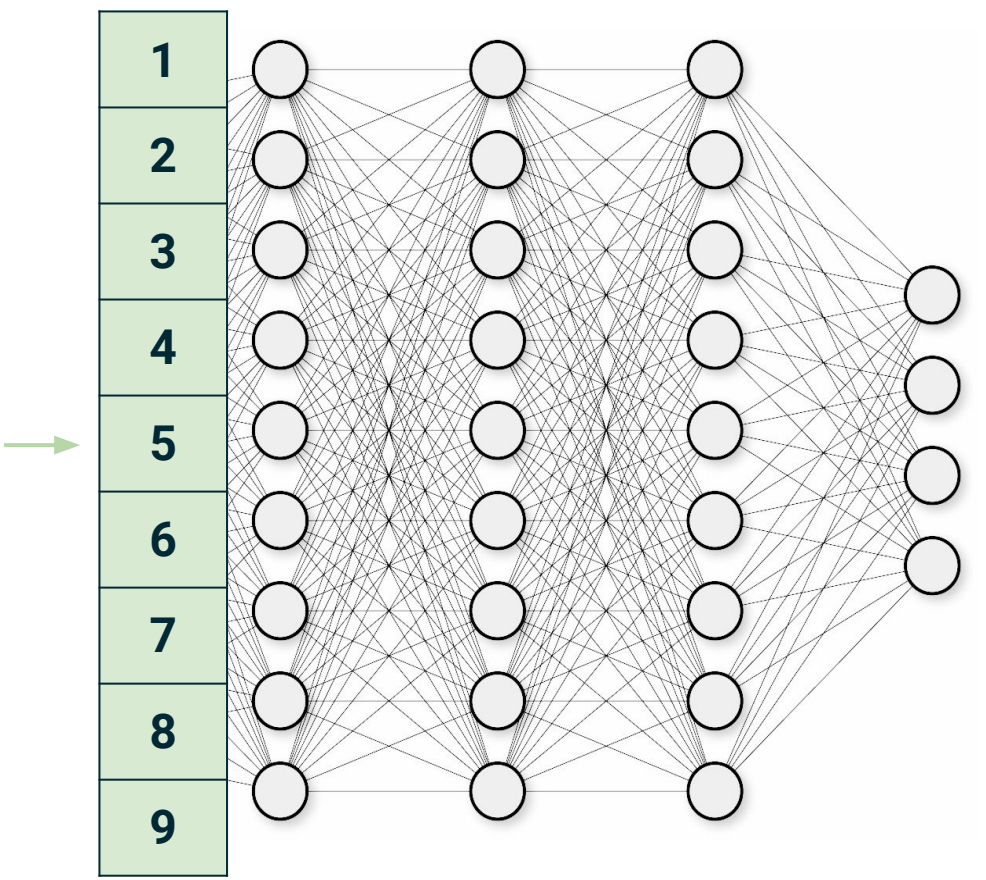
Flatten



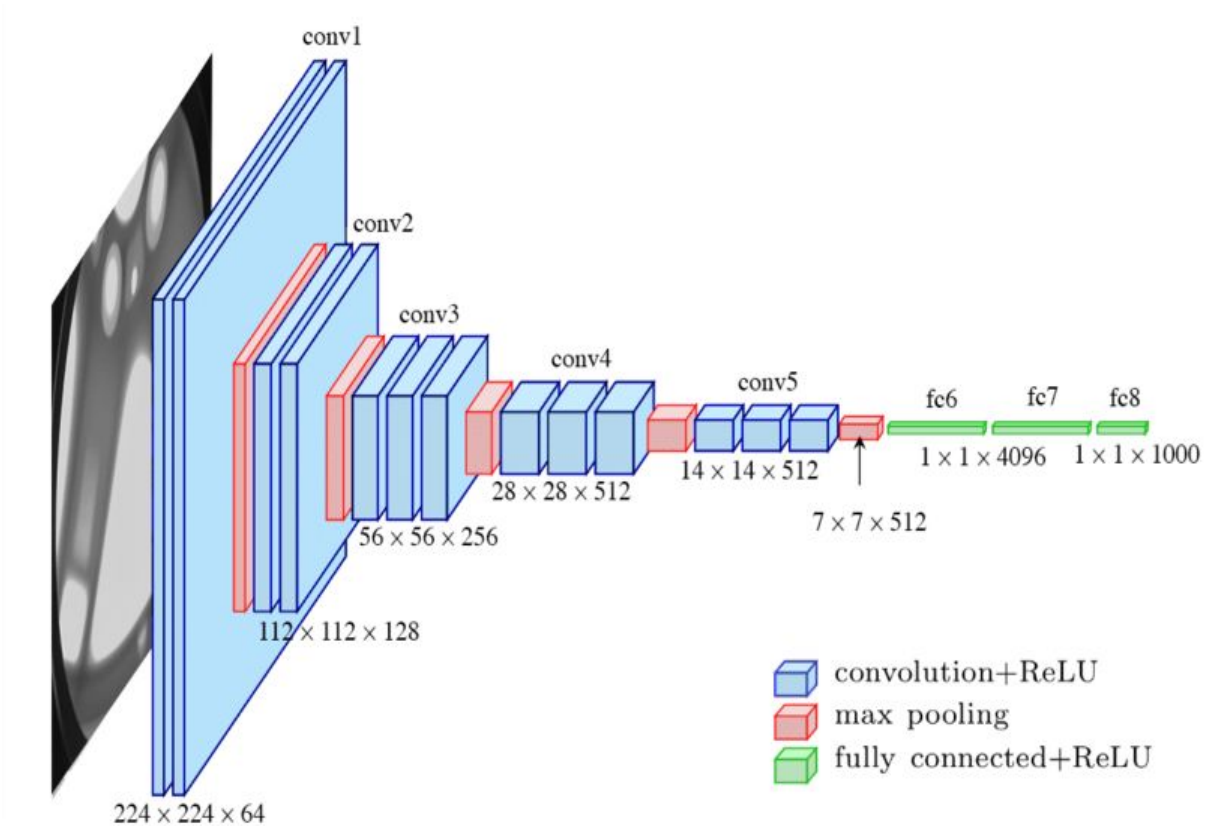


Flatten

1	2	3
4	5	6
7	8	9

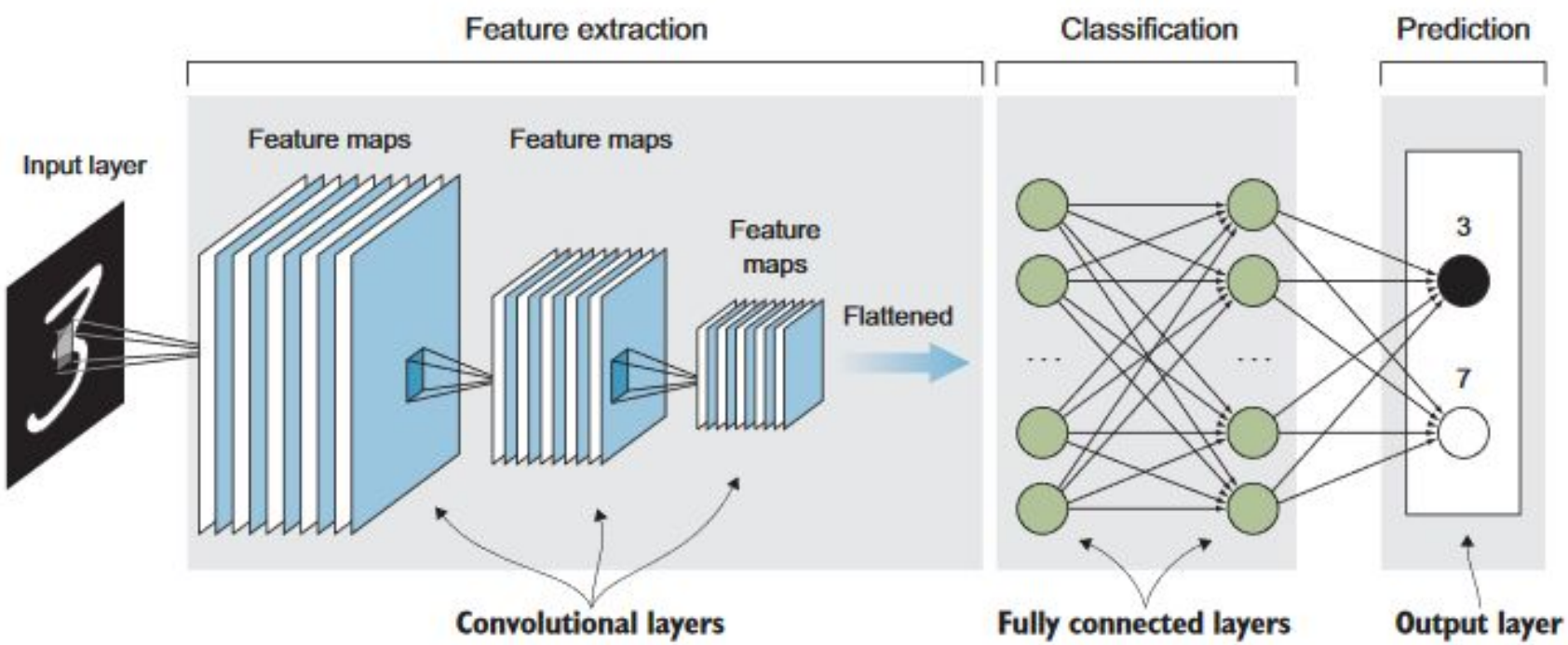


CNN

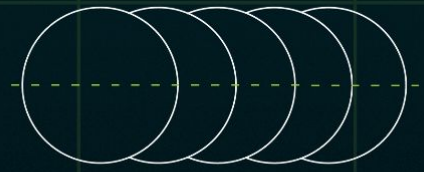


Referencias. Ferguson, Max & ak, Ronay & Lee, Yung-Tsun & Law, Kincho. (2017). Automatic localization of casting defects with convolutional neural networks. 1726-1735. 10.1109/BigData.2017.8258115.

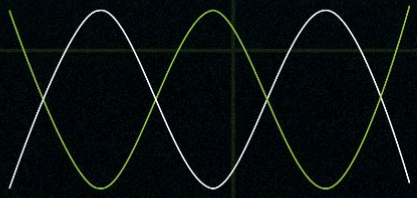
CNN

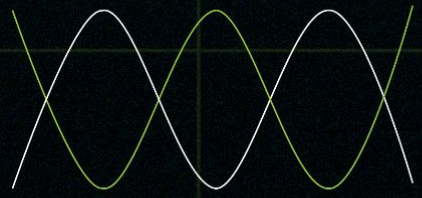


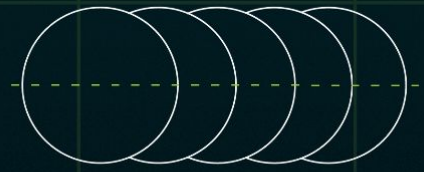
Referencias. Mohamed Elgendy. (2020). Deep Learning for Vision Systems



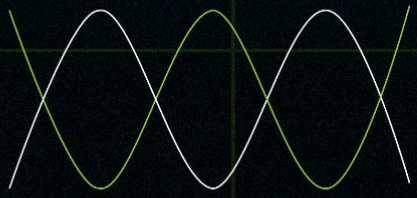
Clasificación con redes convolucionales

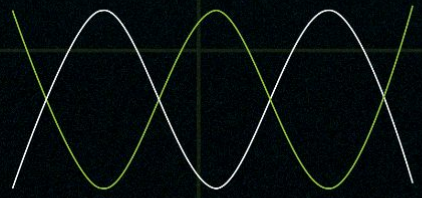


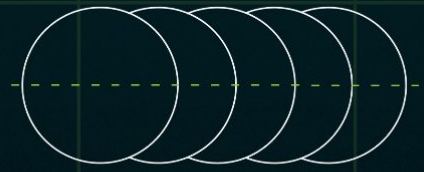




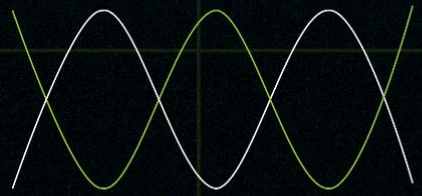
Creación de red convolucional para clasificación

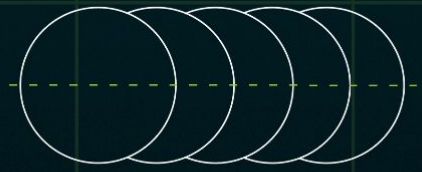




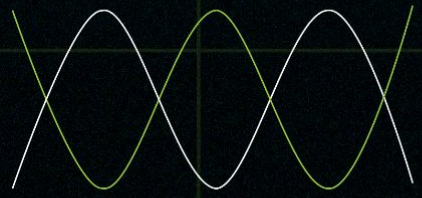


Entrenamiento de un modelo de clasificación con redes convolucionales



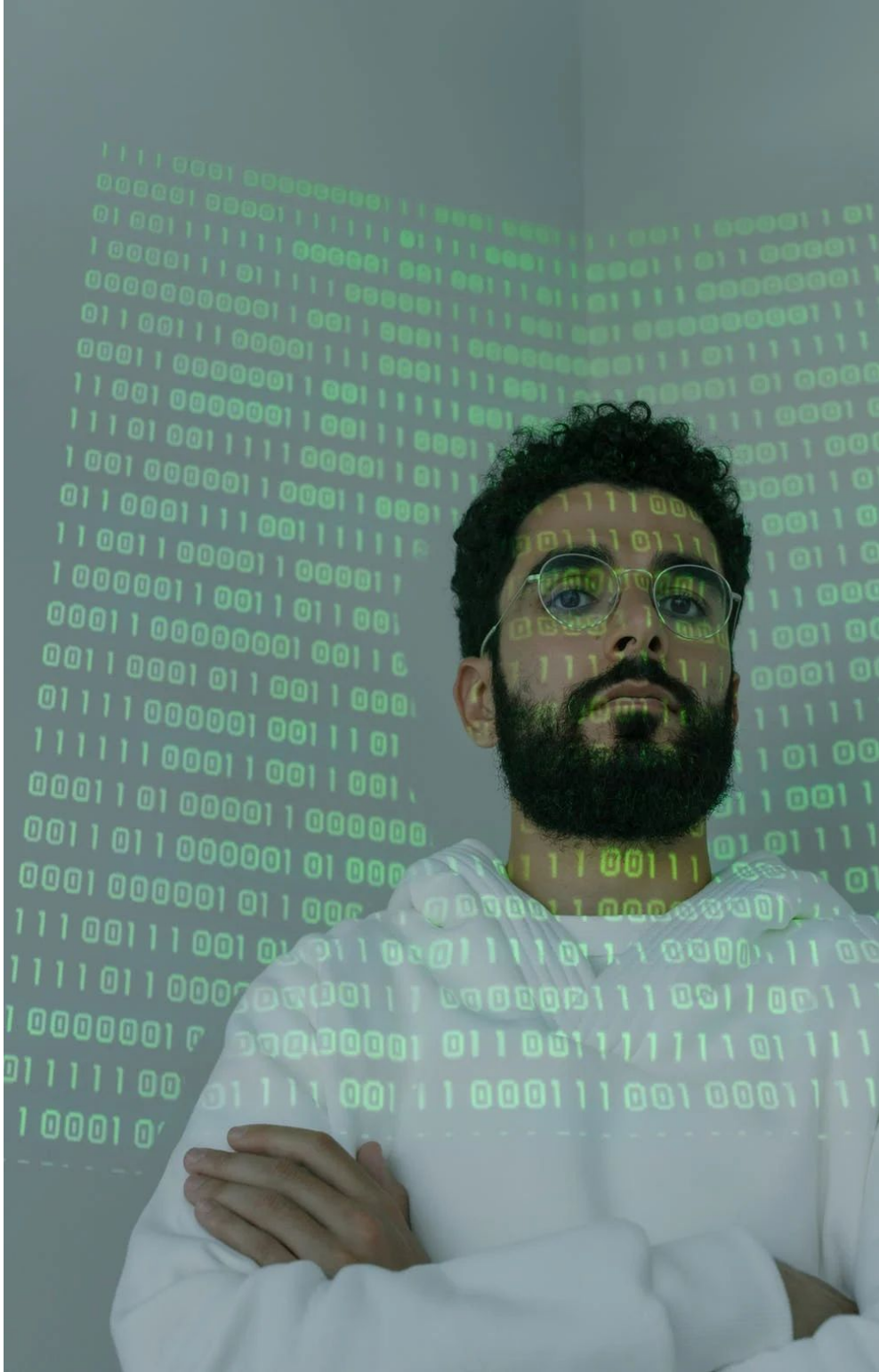


Data augmentation



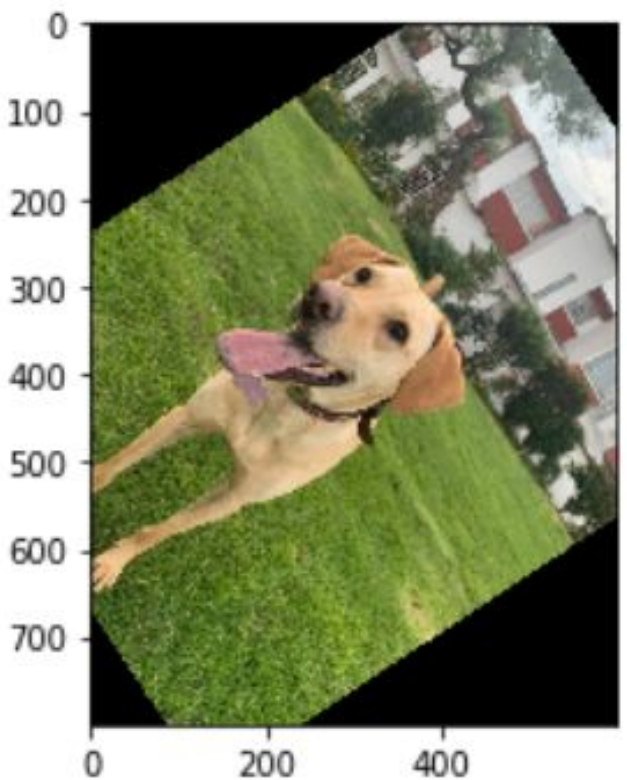
¿El problema?

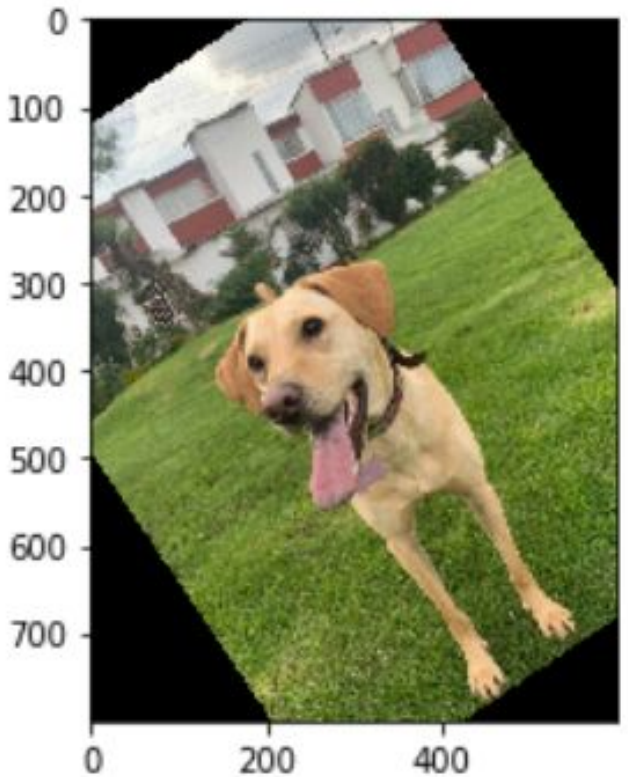
Es difícil obtener y clasificar las imágenes para entrenar nuestro modelo.



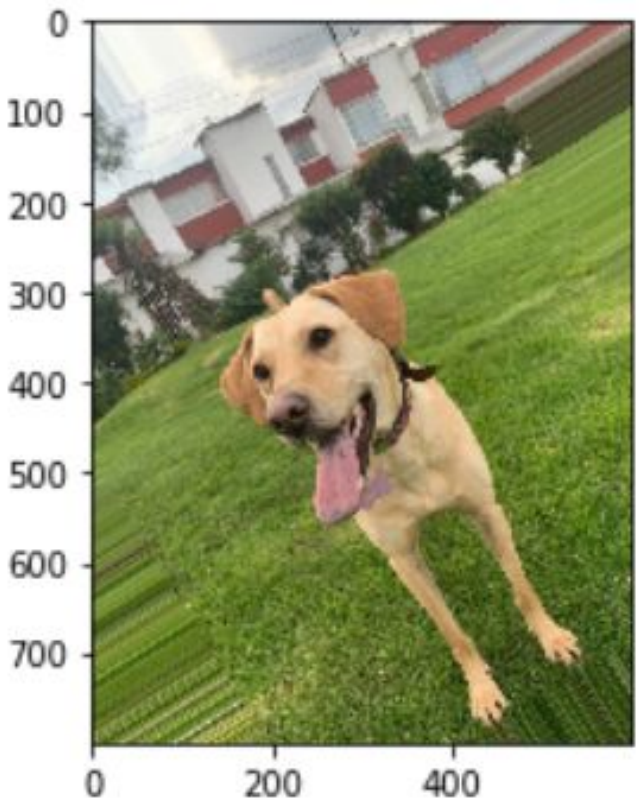
Data augmentation

Rotation


0
100
200
300
400
500
600
700
0
200
400

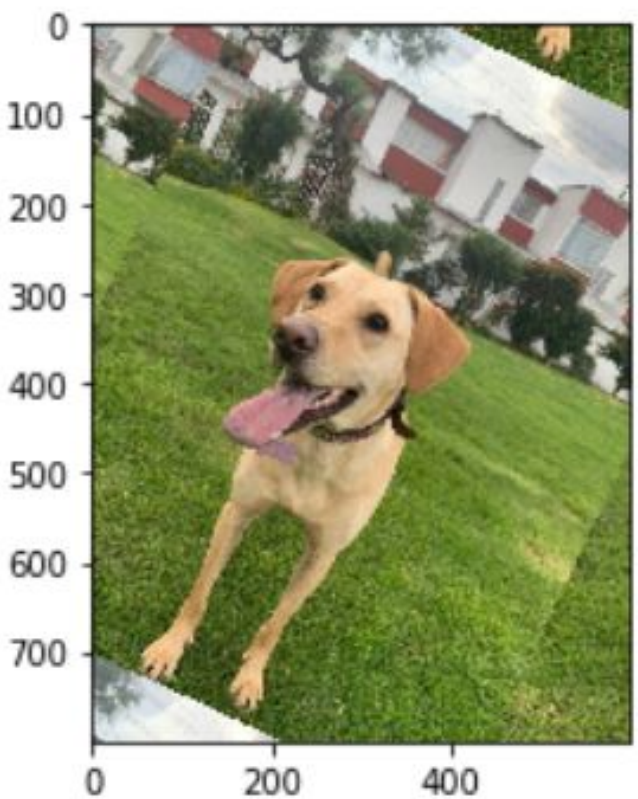

0
100
200
300
400
500
600
700
0
200
400

fill_mode = nearest

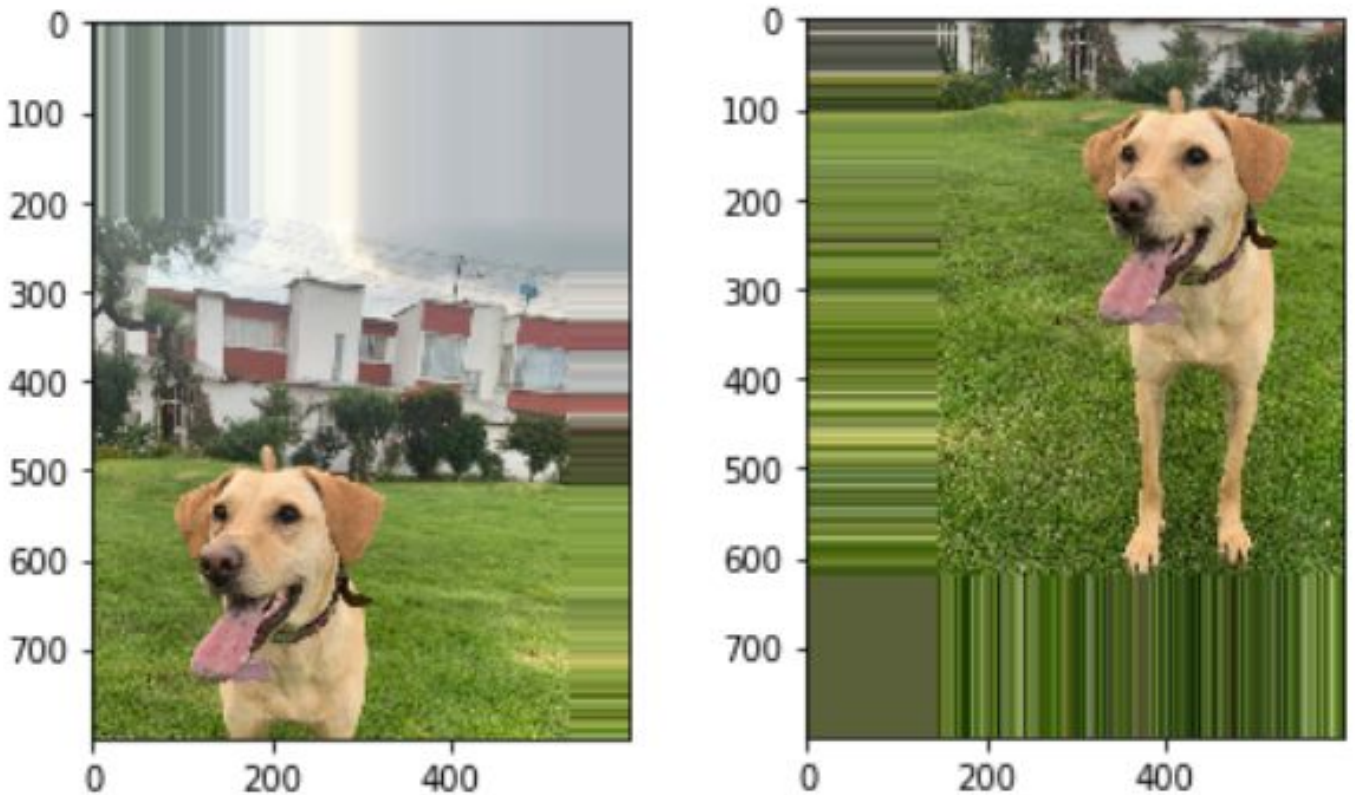


fill_mode = reflect

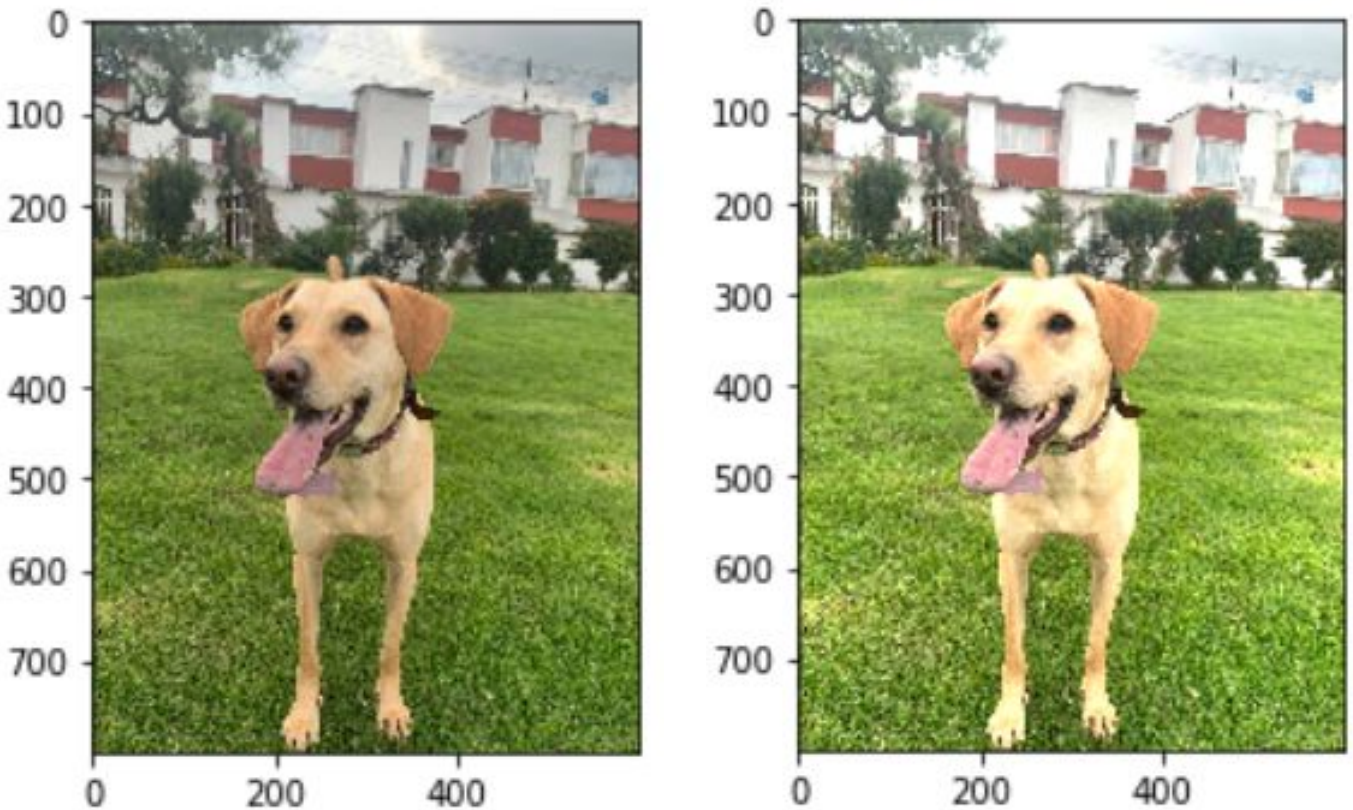
fill_mode = wrap



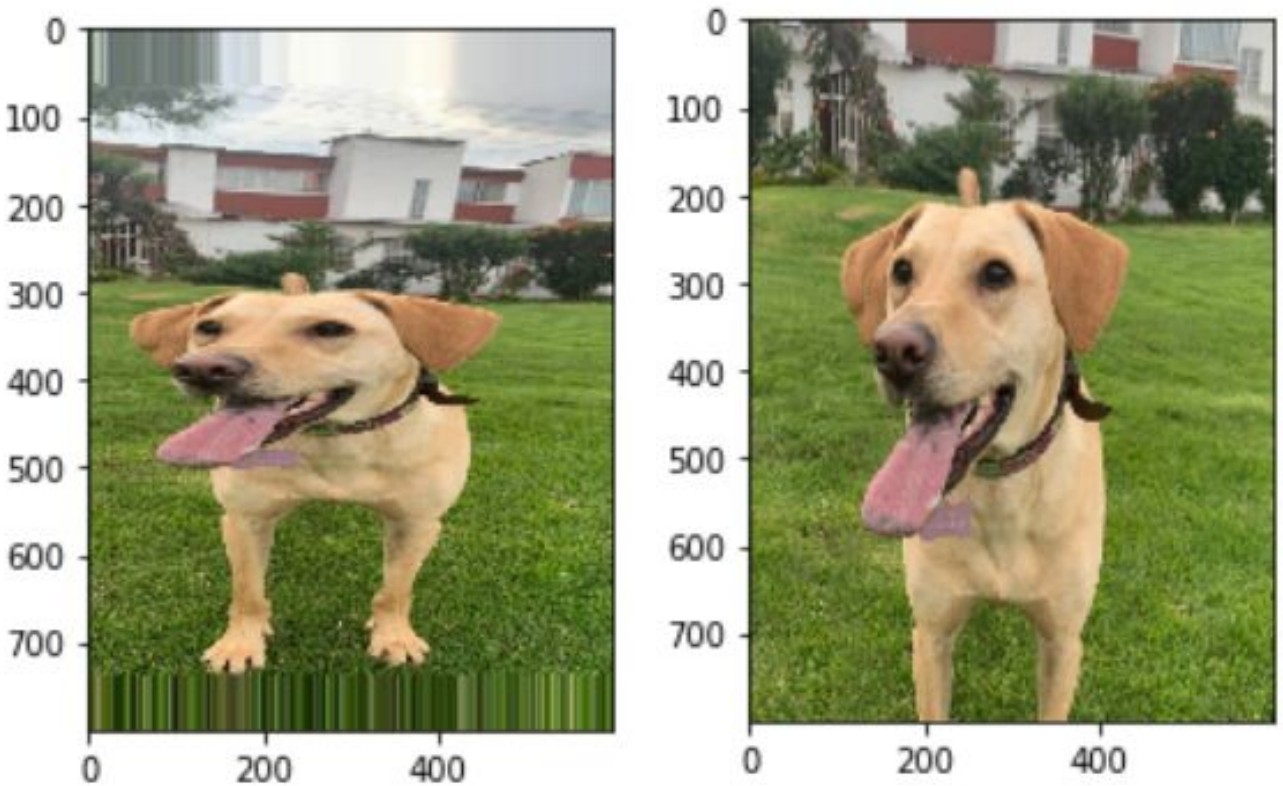
width_shift_range
height_shift_range

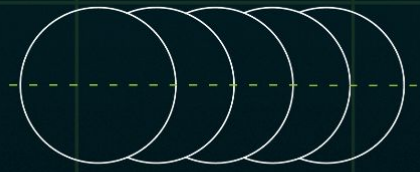


brightness_range

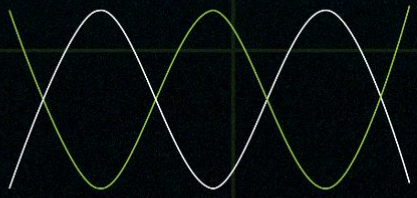


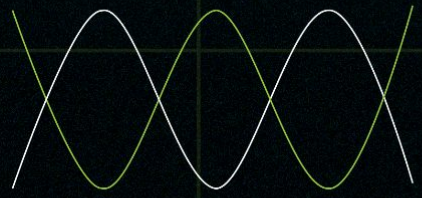
Zoom_range

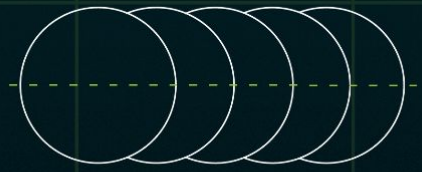

0
100
200
300
400
500
600
700
0
200
400
0
100
200
300
400
500
600
700
0
200
400



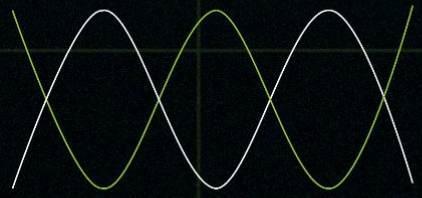
Aplicando data augmentation

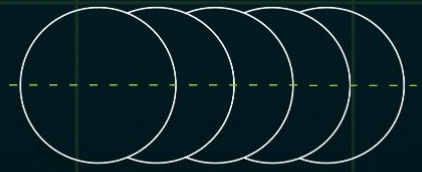




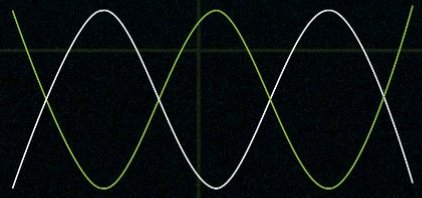


Early stopping y checkpoints



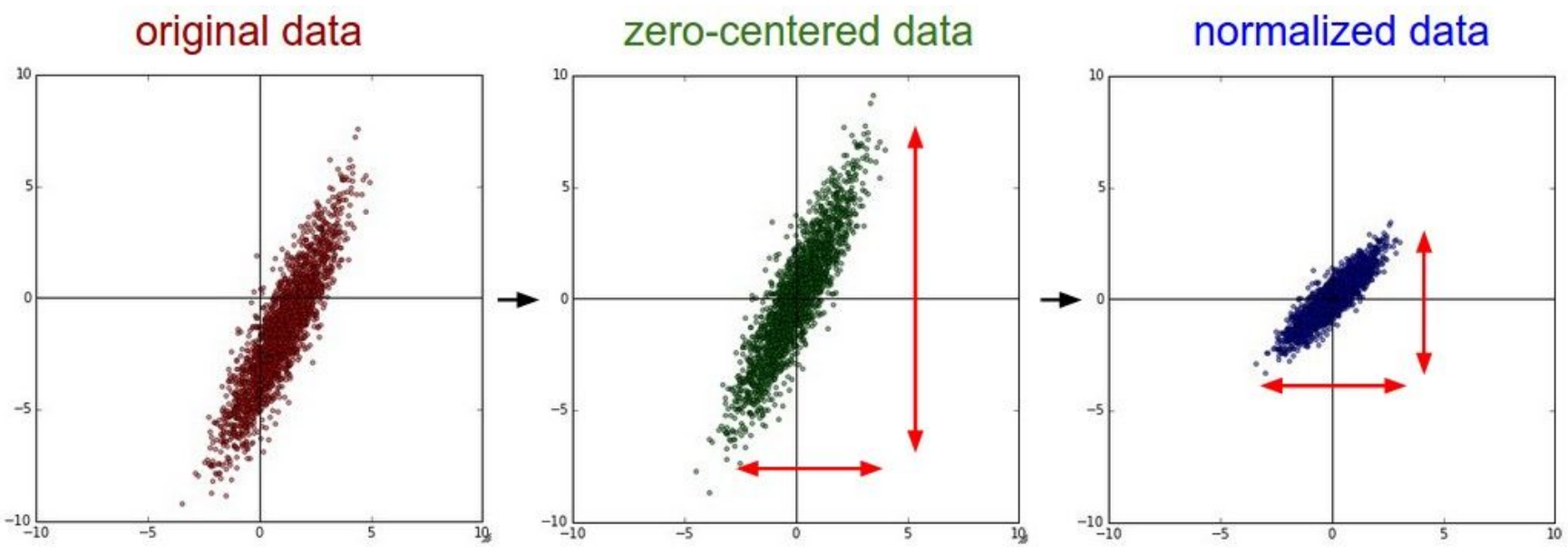


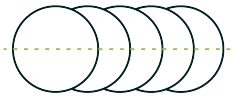
Batch normalization



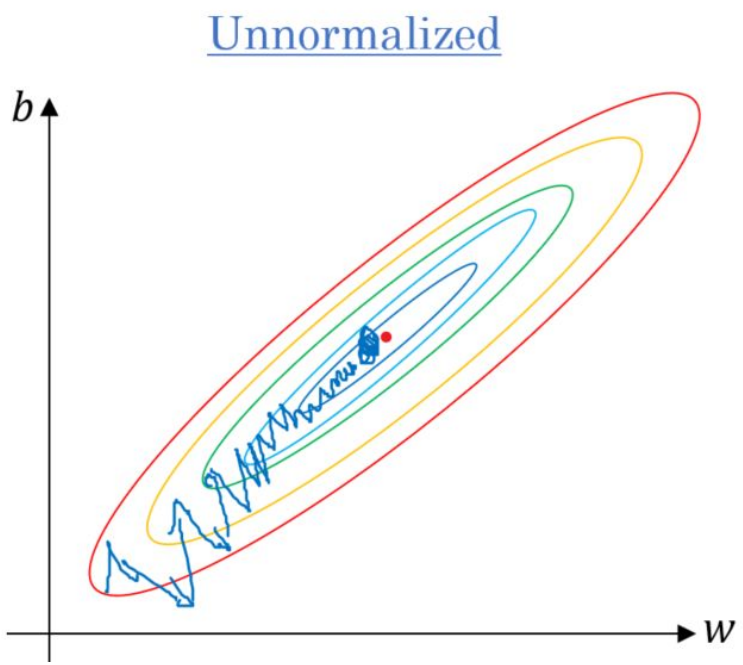
Normalization

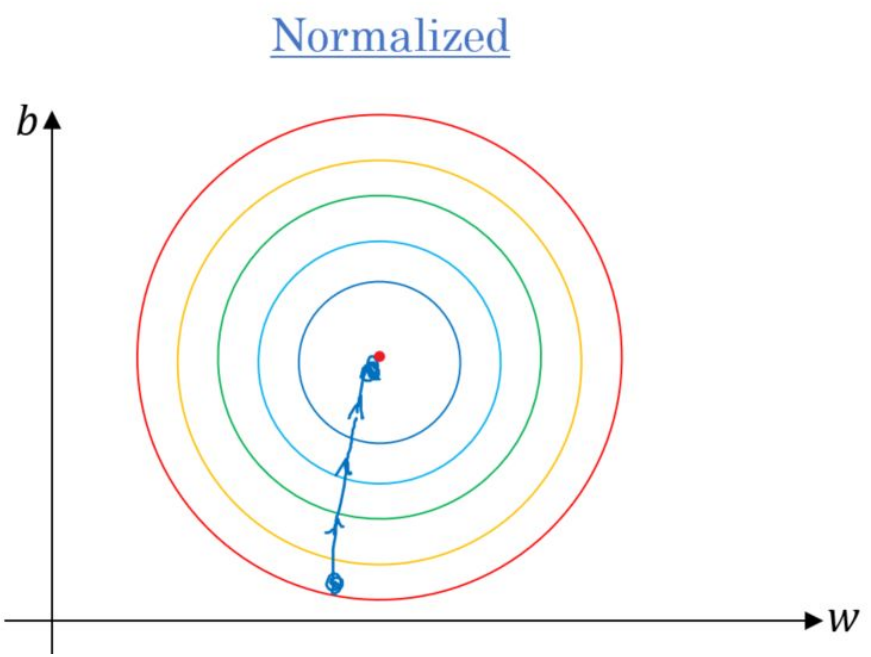
- **Valores pequeños**: típicamente entre 0 y 1.
- **Data homogénea**: todos los píxeles tienen datos en el mismo rango.



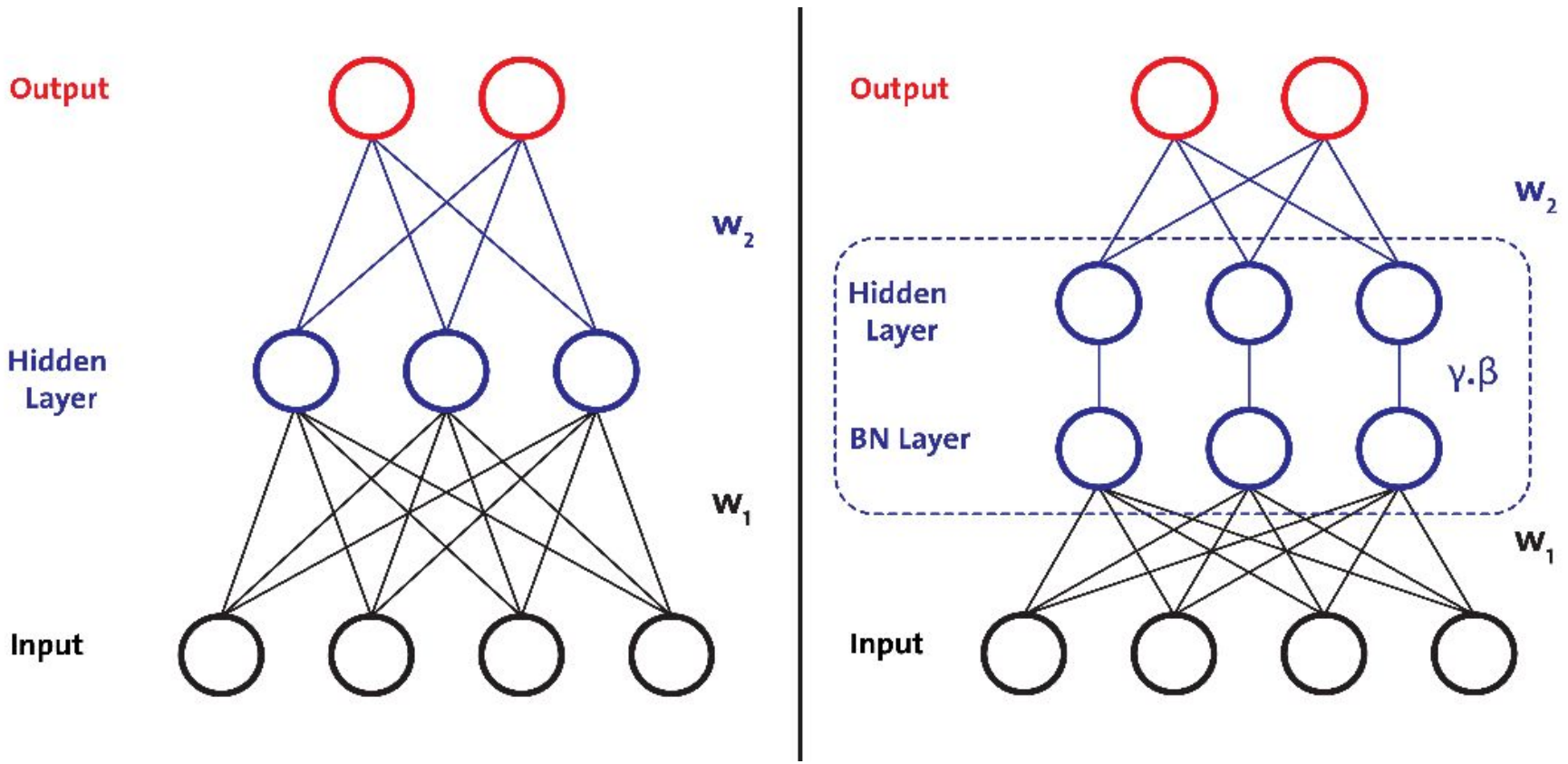


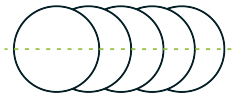
Normalization





Batch normalization





Matemática detrás

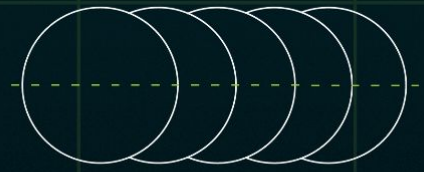
$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \quad \rightarrow \quad \text{mini-batch mean}$$

$$\sigma^2{}_B = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 \quad \rightarrow \quad \text{mini-batch variance}$$

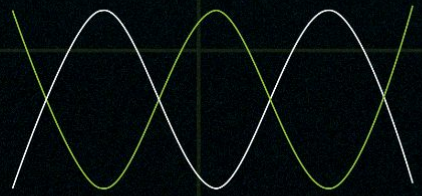
$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma^2{}_B + \epsilon}} \quad \rightarrow \quad \text{normalize}$$

$$y_i = \gamma\hat{x}_i + \beta \equiv BN_{y,b}(x_i) \quad \rightarrow \quad \text{scale and shift}$$

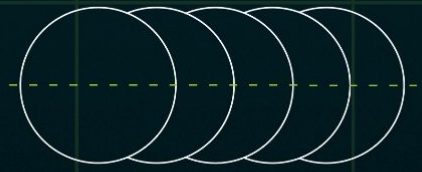
```
model.add(Conv2D(base_hidden_units, (3,3),
padding='same', input_shape=x_train.shape[1:]))
model.add(Activation('relu'))
model.add(BatchNormalization())

# CONV2
model.add(Conv2D(base_hidden_units, (3,3),
padding='same', input_shape=x_train.shape[1:]))
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))
```



Optimización de modelo de clasificación

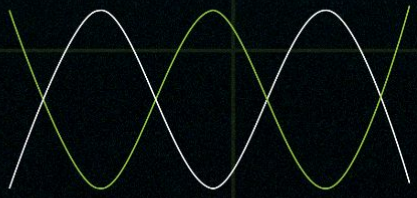


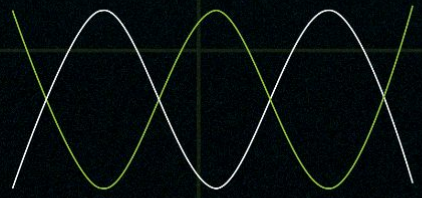
Entrenamiento de nuestro modelo de clasificación optimizado

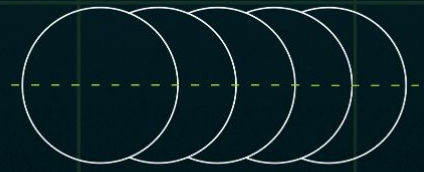


Clasificando entre gatos y perros

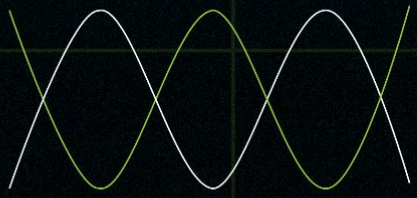
Resolviendo una competencia de Kaggle

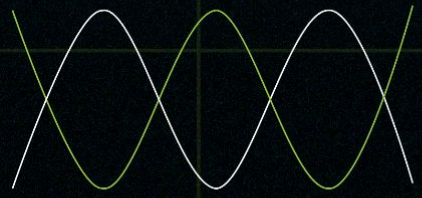






Siguientes pasos con redes neuronales convolucionales

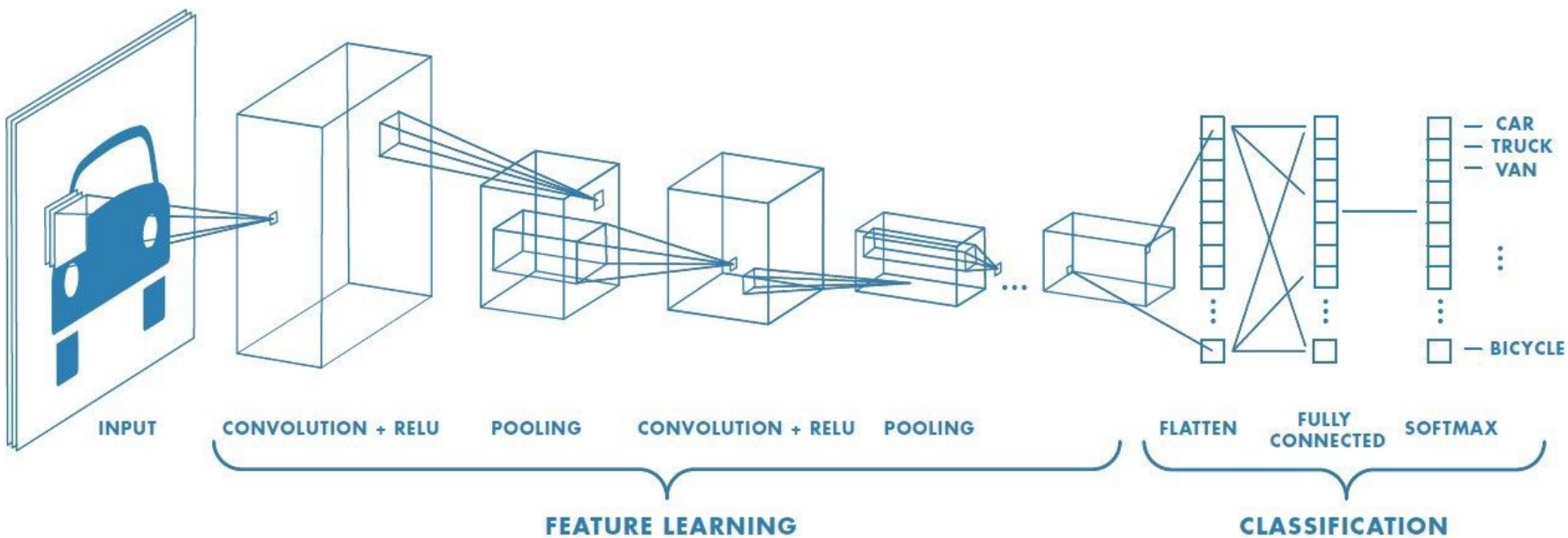




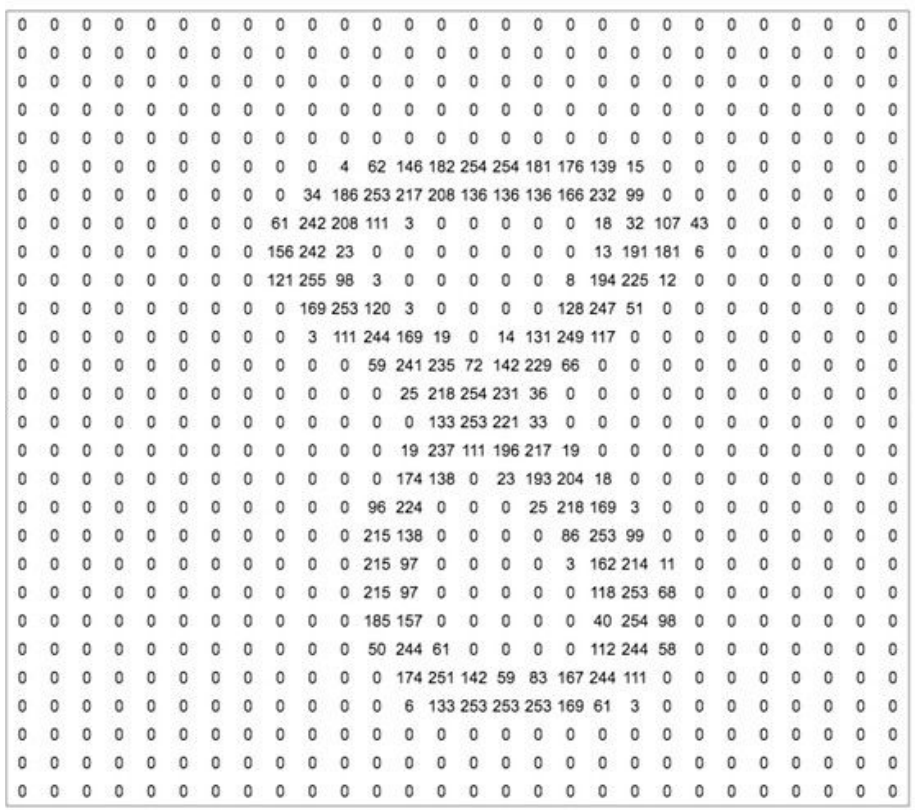
La importancia de computer vision

Clasificación de imágenes

Redes convolucionales



El kernel



Kernel / Filter

1	1	1
0	0	0
-1	-1	-1

Padding

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

$6 \times 6 \rightarrow 8 \times 8$

*

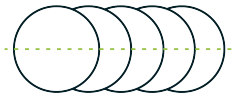
1	0	-1
1	0	-1
1	0	-1

3×3

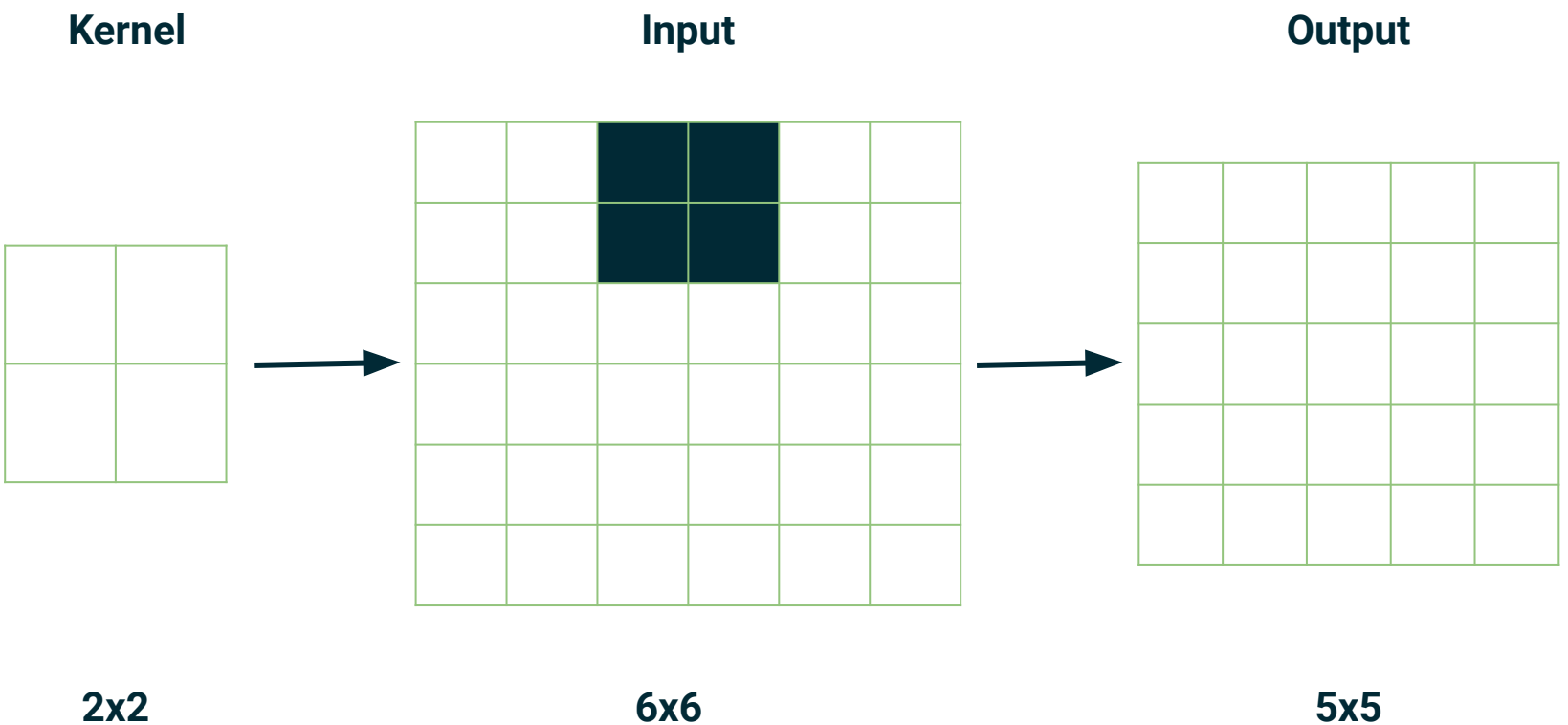
=

-10	-13	1			
-9	3	0			

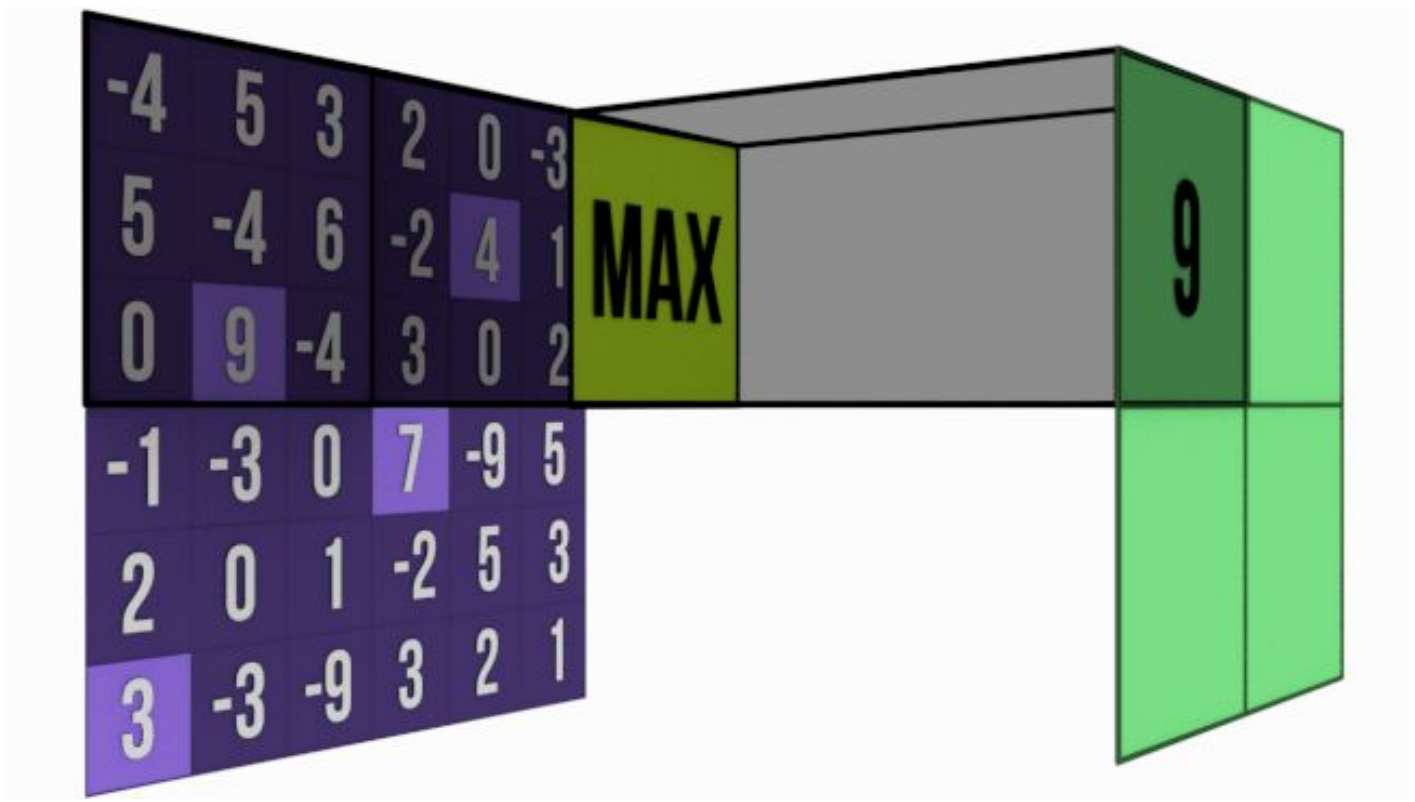
6×6

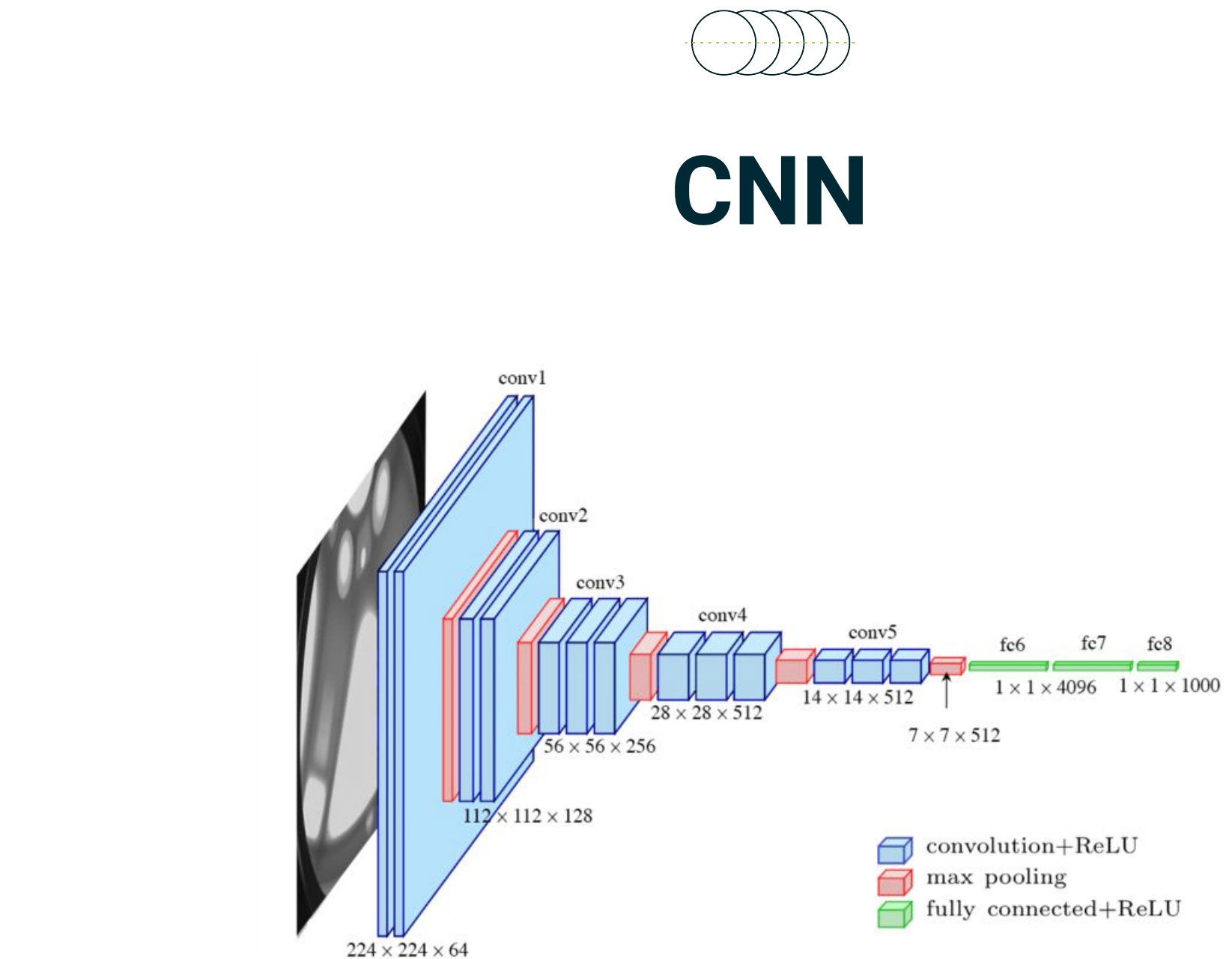


Strides



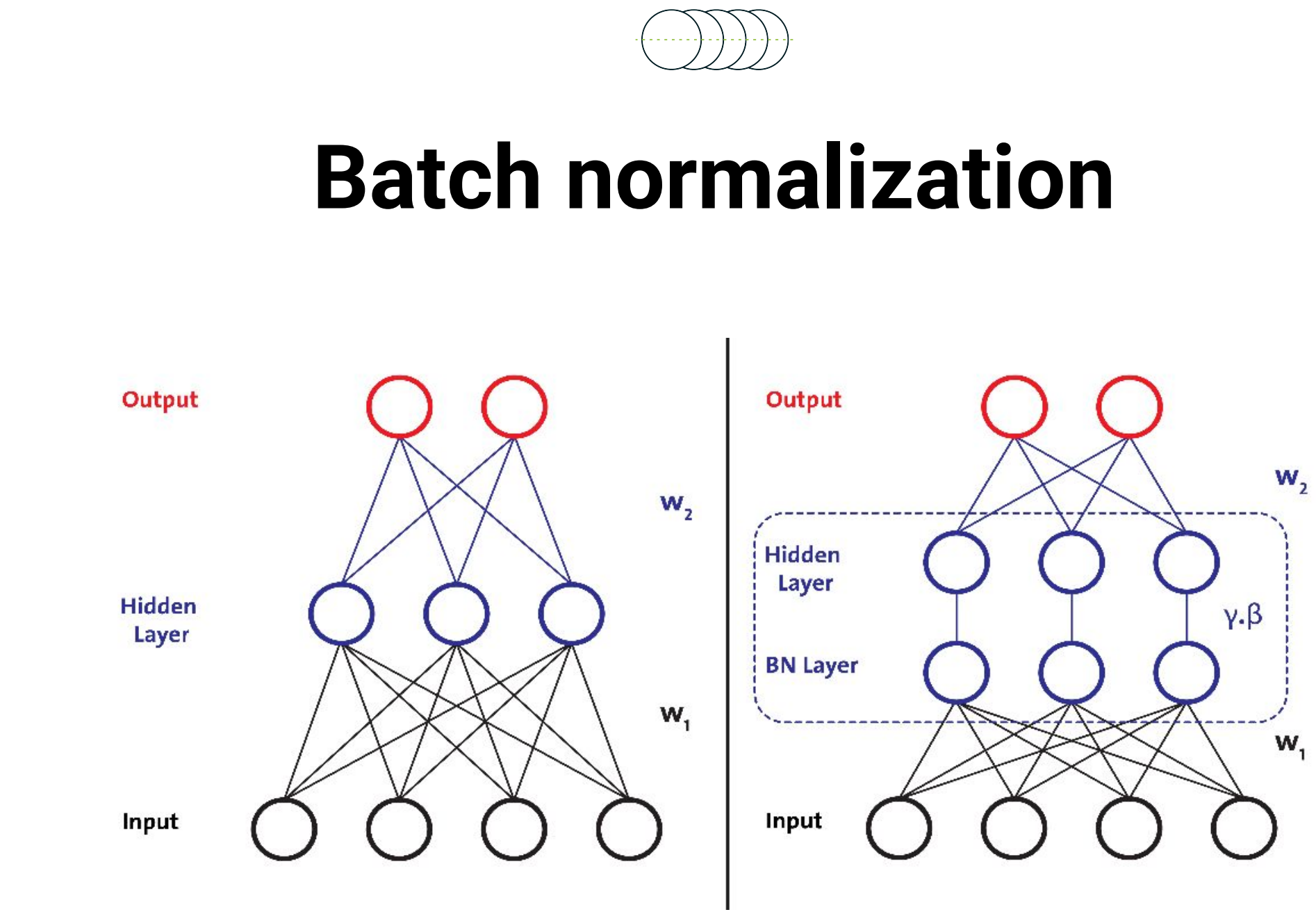
Pooling



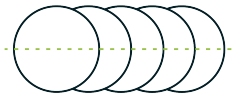

conv1
conv2
conv3
conv4
conv5
fc6
fc7
fc8
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
convolution+ReLU
max pooling
fully connected+ReLU

CNN

Data augmentation

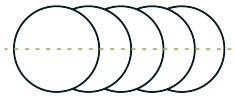


Batch normalization



Otras mejoras

- Normalización de datos
- Capa de Dropout
- Regularización
- Checkpoints
- Early stopping



Siguientes pasos

- Arquitecturas más complejas de CNN (Alexnet, VGGNet, LeNet, ResNet, etc).
- Uso de redes preentrenadas.
- TensorBoard.
- ¿Qué ve una red convolucional?
- Inception.