Capítulo 5. Deep learning para visión por computadora

5.1 Introducción a los convnets

Objetivo

- Introducir redes convolucionales o `convnets`
  - Clasificación de imágenes
- *Data augmentation* para mitigar sobreajustes
- Uso de convets pre-entrenados
  - Ajuste fino
  - Visualización del proceso de aprendizaje

MNIST: dígitos 0-9

- Utilizando DNN alcanzamos un *accuracy* del 97.8%
- Usaremos una pila de capas:
  - Conv2D
  - MaxPooling2D
- Entrada:
  - Tensor imagen: (altura, anchura, canales)
  - MNIST (28, 28, 1)
    - Un solo canal de grises

Sumario del modelo

- La salida de cada capa es un tensor
  - (altura, anchura, canales)
  - Dimensiones altura/ anchura tienden a disminuir
  - El número de canales está controlado por el 1er. argumento de las capas

```
from keras import layers
from keras import models

model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), \
          activation='relu', \
          input_shape=(28, 28, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), \
          activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), \
          activation='relu'))
```

- Añadir capas densas
  - Aplanar el modelo 3D —> 1D: (3, 3, 64) —>(576, 1)
  - Añadir capas densas
  - `Softmax` para clasificación en última capa

```
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
```

- Cargar y preparar datos MNIST
  - Entrenamiento y test
  - Etiquetas
- Compilar y ajustar
  - Tarda 2 min.

```
from keras.datasets import mnist
from keras.utils import to_categorical

(train_images, train_labels), \
  (test_images, test_labels) = \
    mnist.load_data()

train_images = train_images.reshape((60000, 28, 28, 1))
train_images = train_images.astype('float32') / 255

test_images = test_images.reshape((10000, 28, 28, 1))
test_images = test_images.astype('float32') / 255

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)


model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=5, batch_size=64)
```

Evaluación

Modelo	**Accuracy**	**Err. rate (1-Acc)**	**Factor redución**
DNN	0.978	0.022	--
Conv	0.993	0.007	0.682[1]

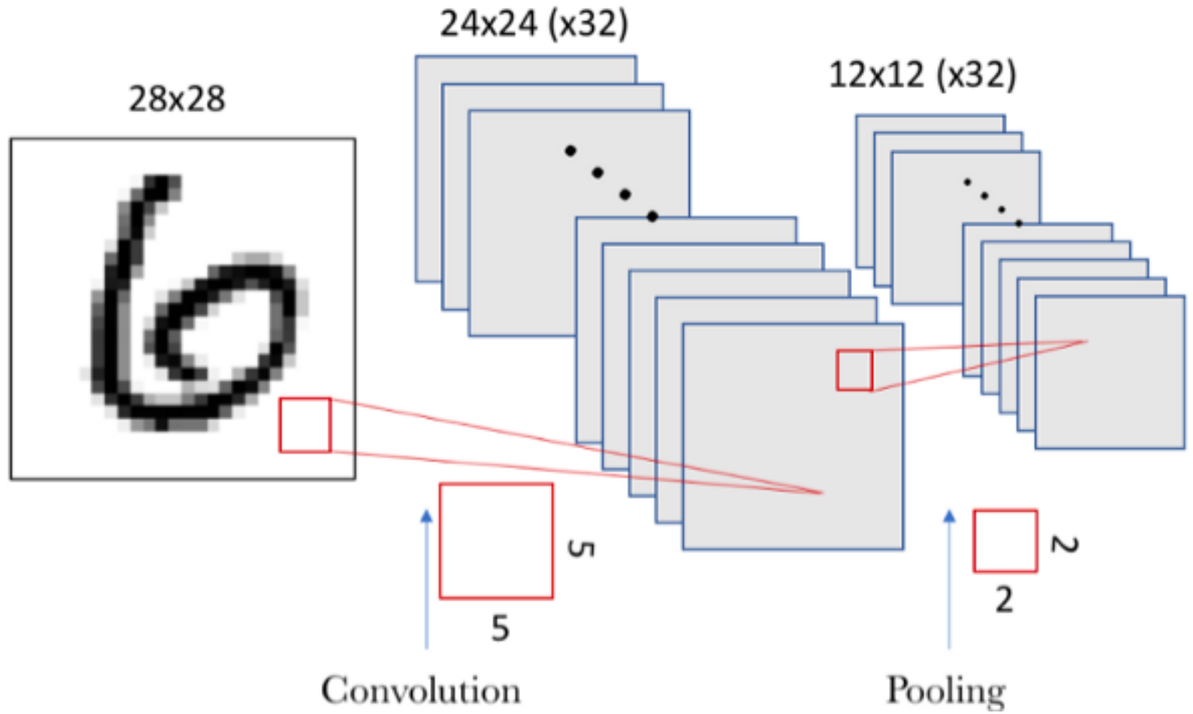
[1] Factor de reducción:

$$1 - \frac{Acc_{Conv}}{Acc_{DNN}} = 1 - \frac{0.007}{0.022} = 0.682$$

La operación de Convolución

Diferencias entre capas

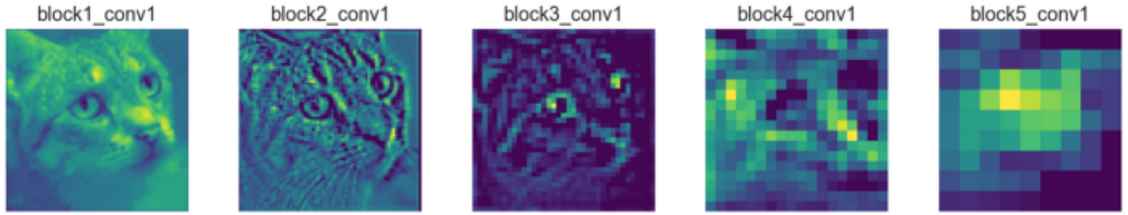
- Capas densas
  - Aprenden patrones globales
    - MINIST: todos los pixeles
- Capas convolucionales[2]
  - Aprenden patrones locales
    - En imágenes: ventanas pequeñas



[2] Jordi Torres 2019, *Convolutional Neural NEtworks for beginners - Practical Guide with Python and Keras*

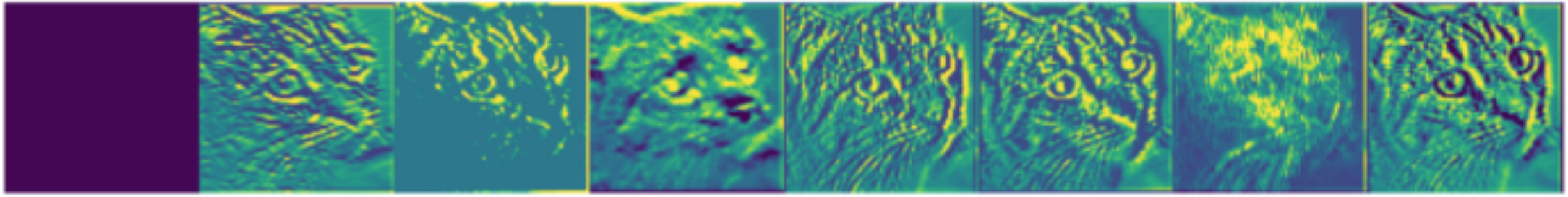
Propiedades de los convnets

- Los patrones son invariantes frente a traslaciones
- Aprenden jerarquías espaciales de patrones
  - Patrones mayores y más abstractos en cada capa[3]

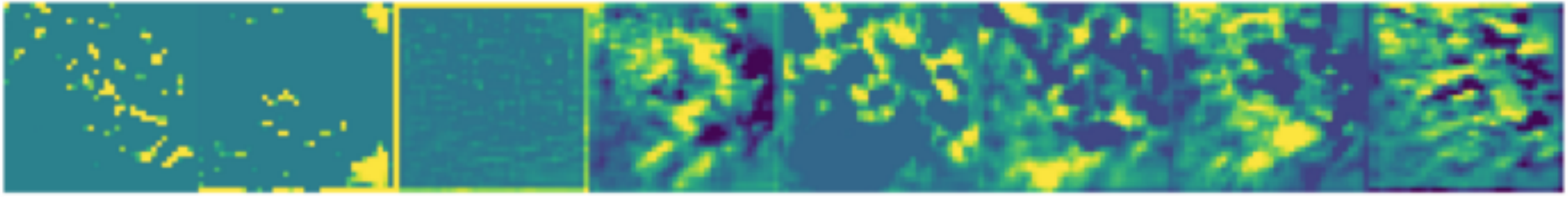


[3] Arden Dertat, 2017, *Applied Deep Learning - Part 4: Convolutional Neural Networks*

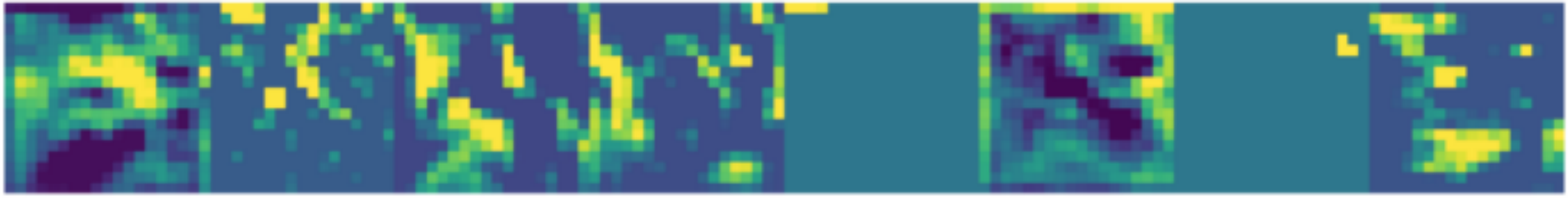
block2_conv1



block3_conv1



block4_conv1



Feature maps (FM)

- Son los tensores 3D:
  - (Altura, Anchura, Profundidad / Canales)
  - Imagen RGB —> 3 canales
  - Imagen B&W —> 1 canal (niveles de grises)

Convolución:

- Extrae cuadros de los FM de entrada
- Transformaciones y FM de salida con tamaño FM arbitrarios
  - Primera capa: Profundidad son Canales de colores
  - Capas sucesivas: Profundidad son *filtros* que codifican aspectos de los datos[4]

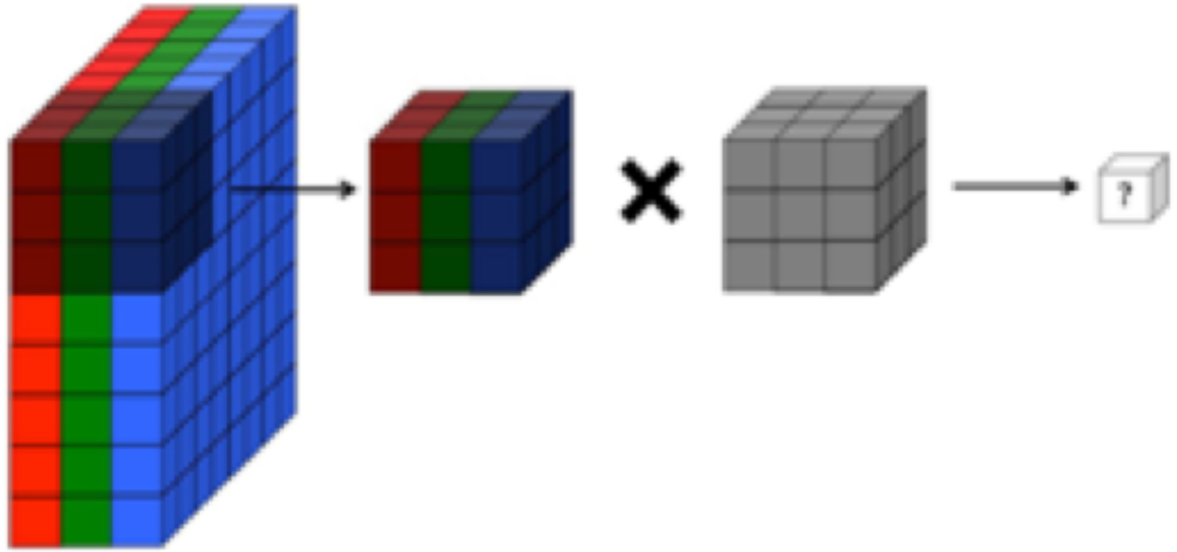


Figure 5-8. Representing a full-color RGB image as a volume and applying a volumetric convolutional filter

[4] StackExchange, *In a CNN, does each new filter have different weights for each input channel, or are the same weights of each filter used across input channels?*

Efectos de borde

- Ejemplo[5]
  - FM 5x5
  - Ventana (filtro) 3x3
  - Sólo se pueden formar 9 cuadros completos

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

Input

1	0	1
0	1	0
1	0	1

Filter / Kernel

[5] Arden Dertat, 2017, *Applied Deep Learning - Part 4: Convolutional Neural Networks*

1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

Input x Filter

4		

Feature Map

1	1x1	1x0	0x1	0
0	1x0	1x1	1x0	0
0	0x1	1x0	1x1	1
0	0	1	1	0
0	1	1	0	0

Input x Filter

4	3	

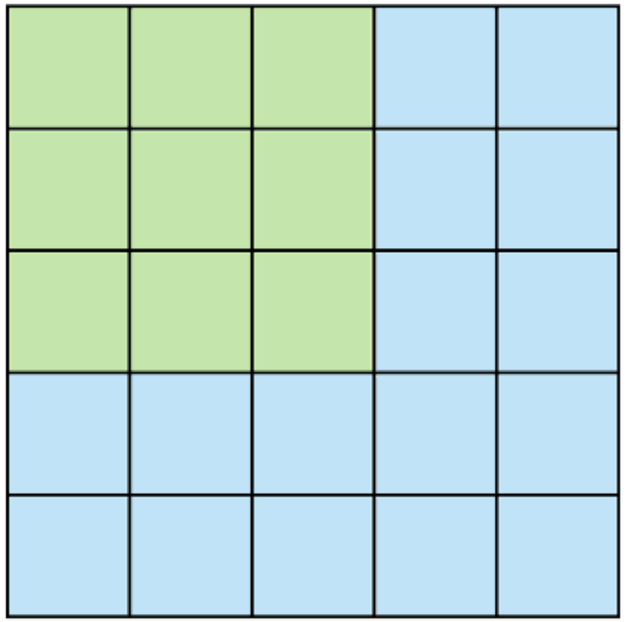
Feature Map

1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

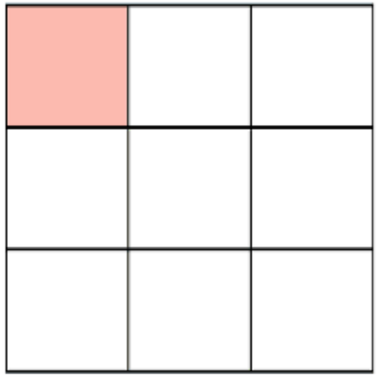
4		

Stride

- Pasos de convolución



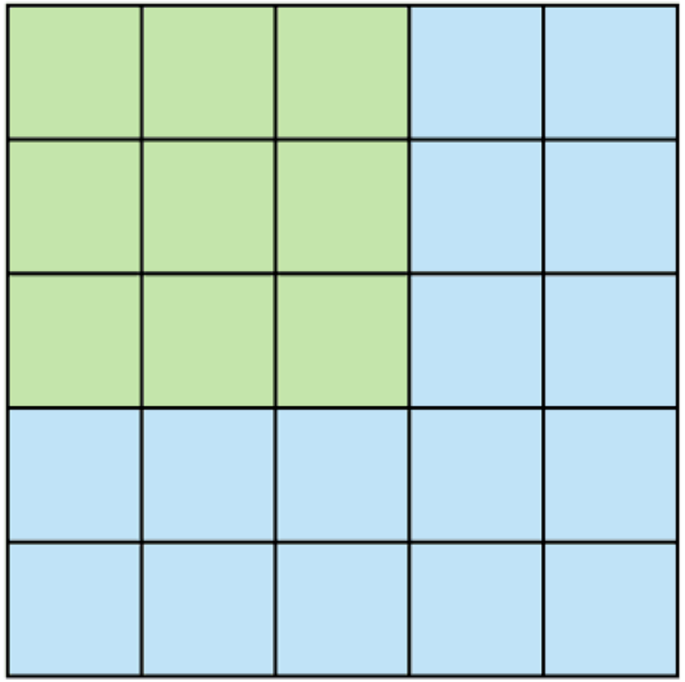
Stride 1



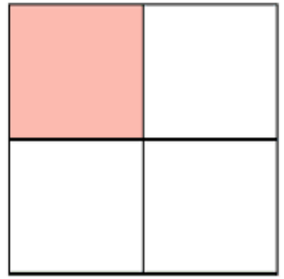
Feature Map

Stride

- Pasos de convolución
- Stride = 2 => Altura y anchura disminuyen por un factor 2
  - Además de efectos de borde



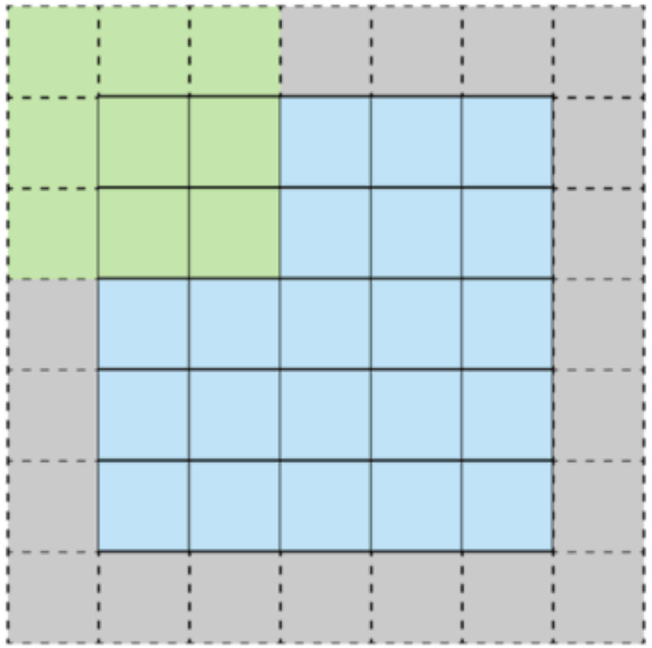
Stride 2



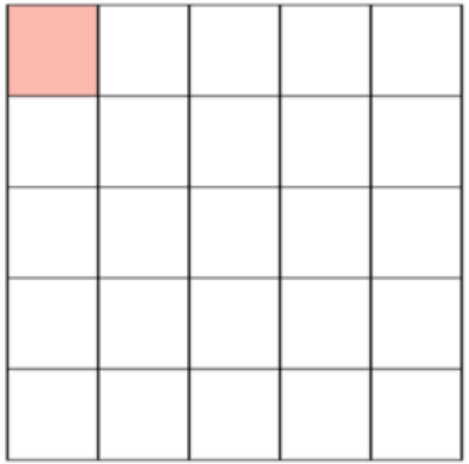
Feature Map

Padding

- Añadir artificialmente filas y columnas para mantener el tamaño de los FM entre capas
- Argumentos:
  - `valid` - **No** (sólo ventanas válidas)
  - `same` - **Sí**



Stride 1 with Padding

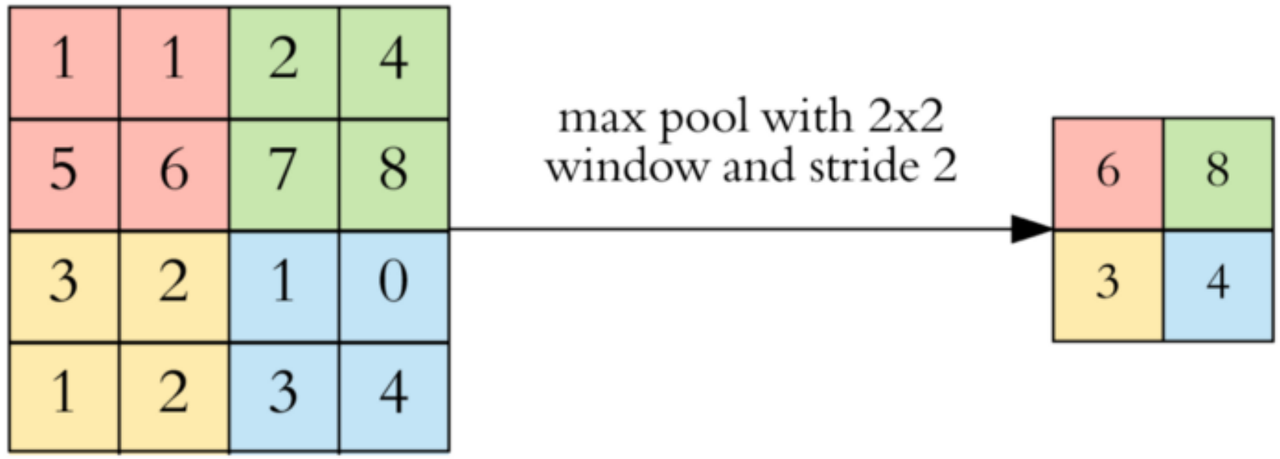


Feature Map

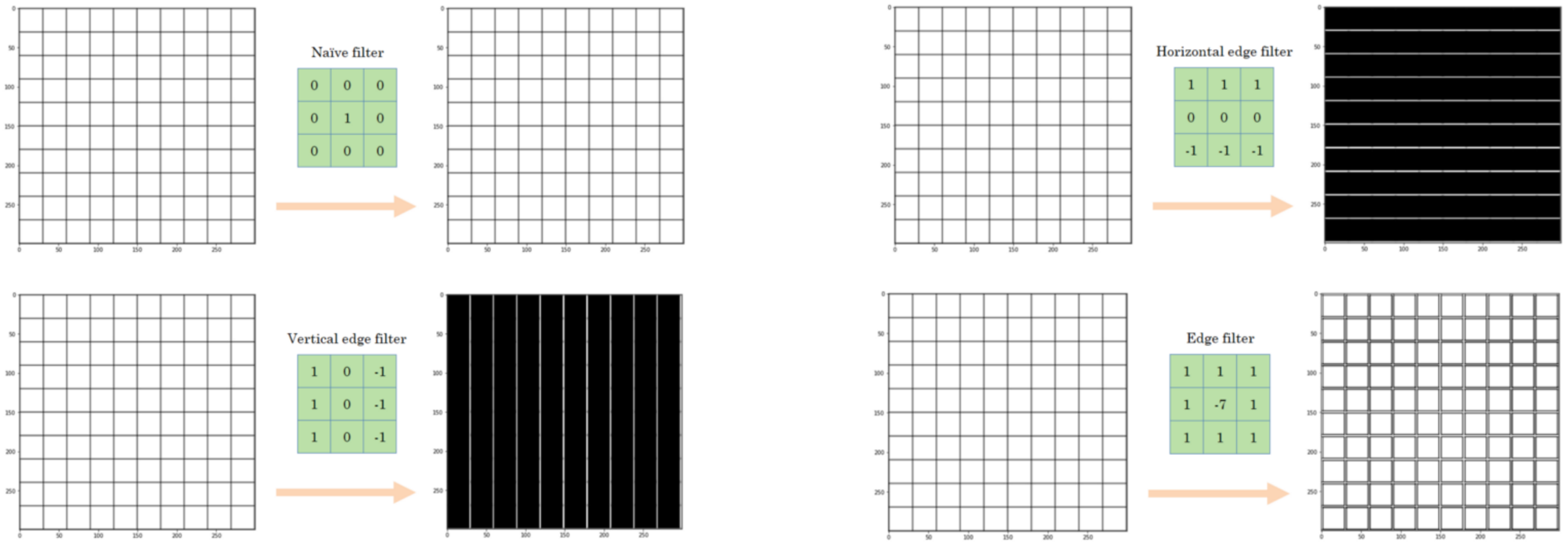
Max-pooling

- Necesarios para aprender la jerarquía y controlar el tamaño
- Disminuye el tamaño de los FM
- Extrae ventanas del FM con el valor máximo de cada canal
  - Aparte del *Max-pooling*, puede haber otros que extraigan promedio, mediana, etc.

En arquitecturas de redes convolutivas, *pooling* (`MaxPooling2D`) se lleva a cabo con ventanas 2x2, *stride* 2 y sin *padding*, mientras que las convoluciones (`Conv2D`) se llevan a cabo en ventanas 3x3, *stride* 1 y con *padding*.

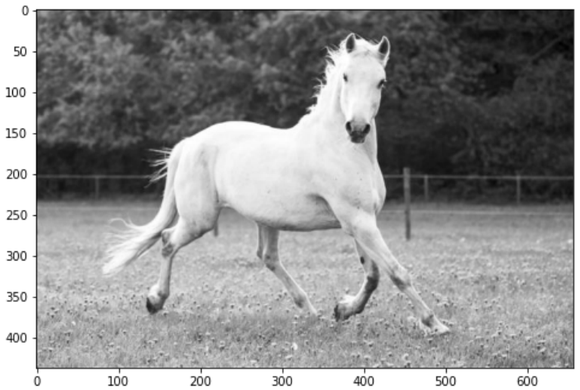


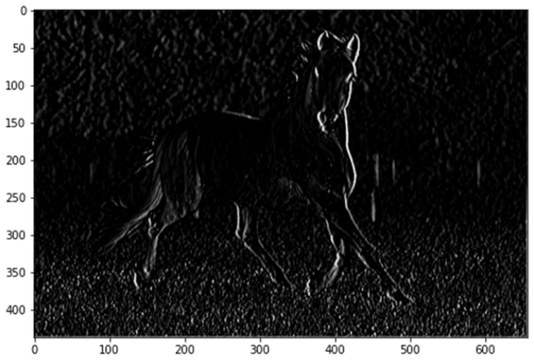
Ejemplos de filtros[6]

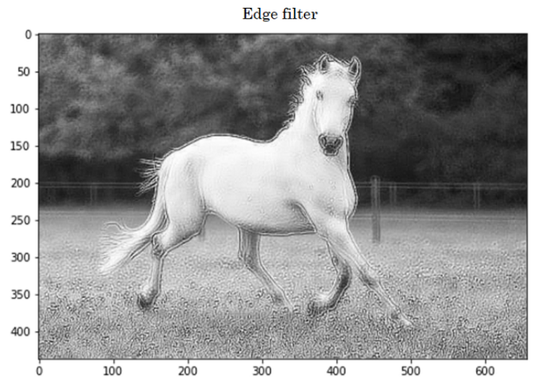


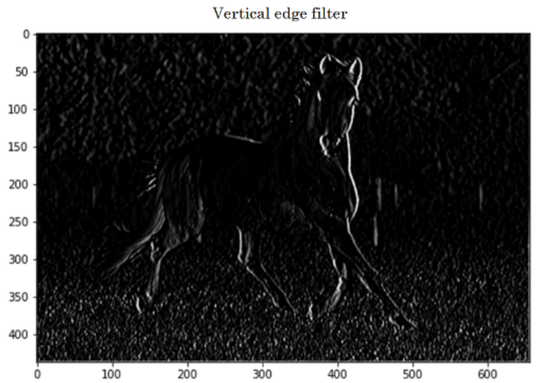
[6] Fuente: Miguel Fernández Zafra, 2020, *Towards Data Science*

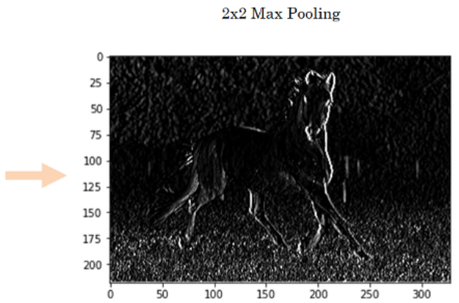
Ejemplos con fotografías reales[6]





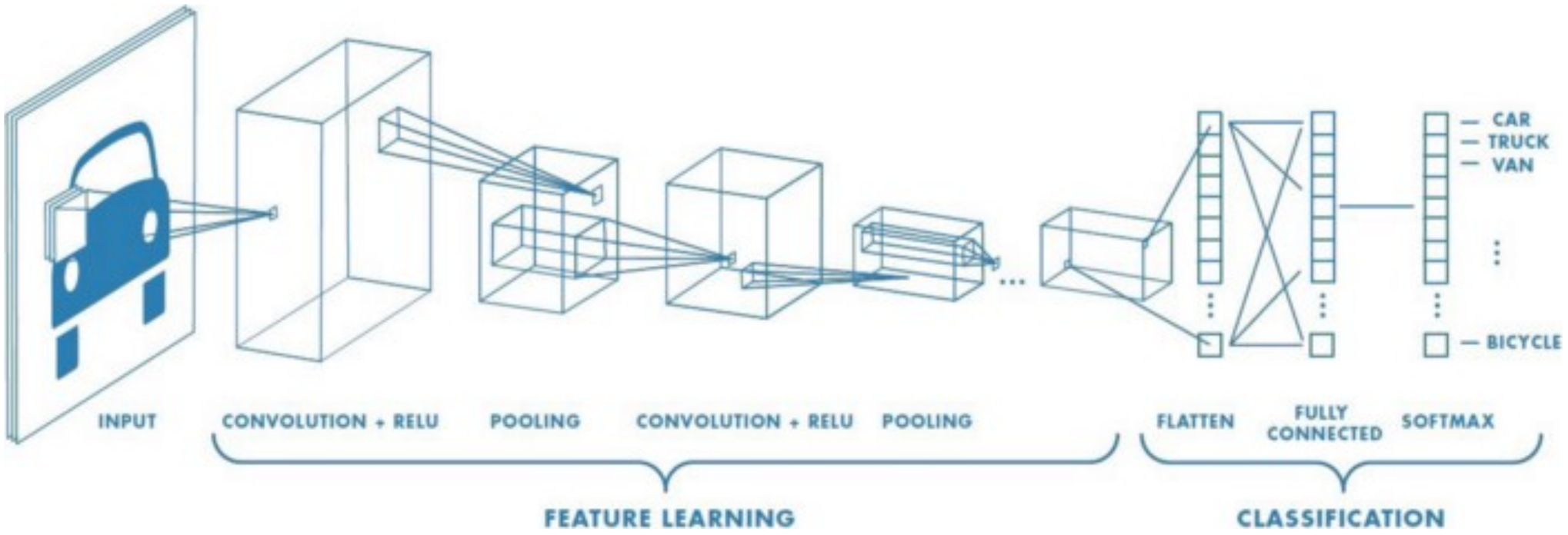






[6] Fuente: Miguel Fernández Zafra, 2020, *Towards Data Science*

Representación de red convolutiva

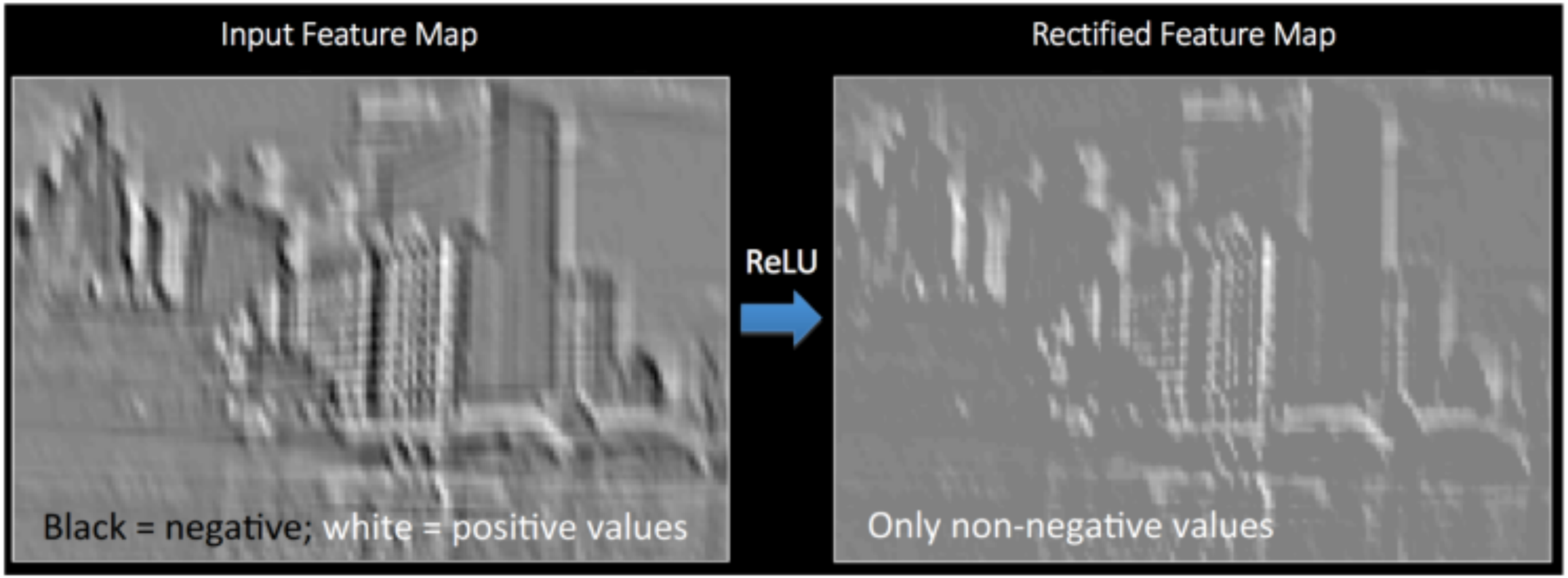


© Sumit Saha 2018, Towards Data Science: A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way

Filtros

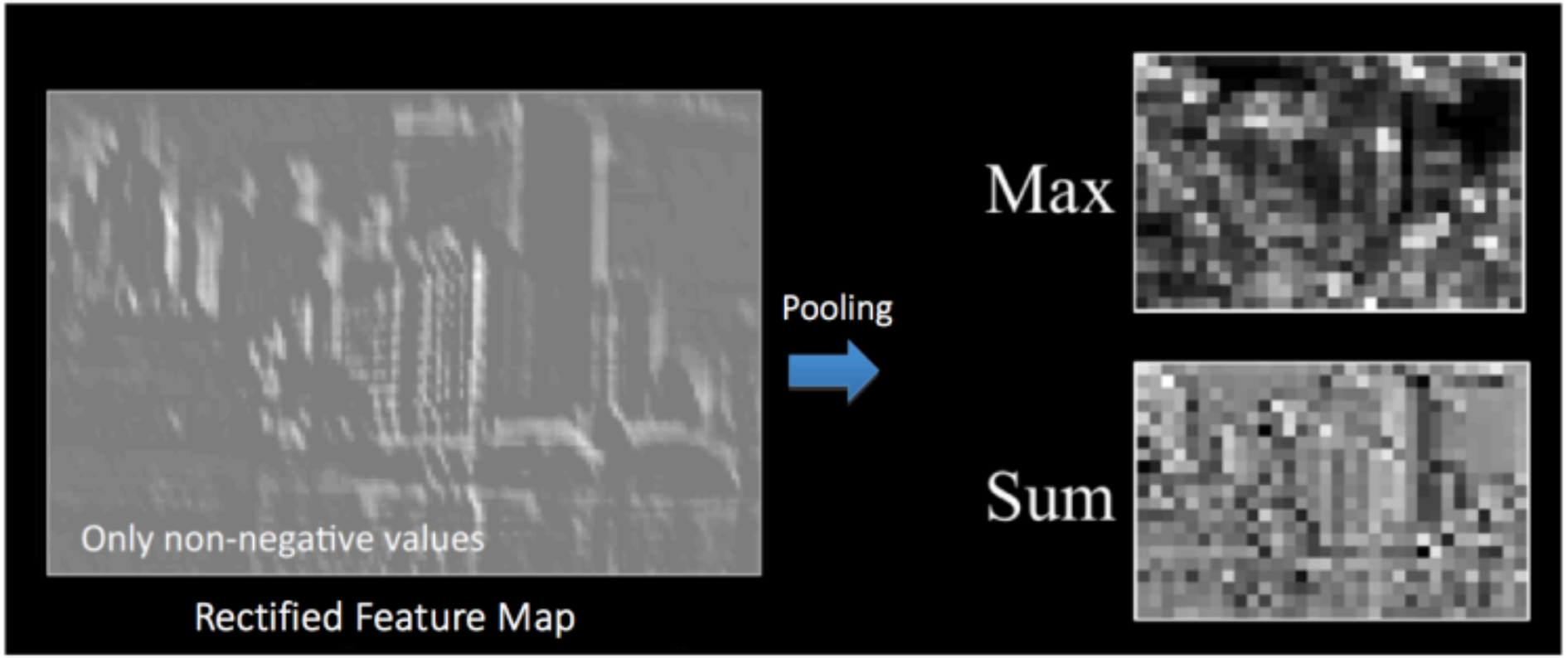
jcgonzalez, 2017, *aps*: Use of convolutional neural network for image classification

Activación ReLu



jcgonzalez, 2017, *aps*: Use of convolutional neural network for image classification

Max y Sum Pooling



jcgonzalez, 2017, *aps*: Use of convolutional neural network for image classification

Convnet con base de datos pequeña

El próximo día

KEEP
CALM
AND
SEE YOU
NEXT CLASS