



# TP4

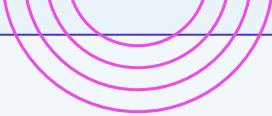
## Redes Neuronales Convolucionales

Problema: Identificar señales de tráfico



Alumno: Castellano, Valentin





# Tabla de contenidos

- |   |  |
|---|--|
| <p><b>01</b> Neuronas</p> <p><b>02</b> Redes Neuronales</p> <p><b>03</b> Aprendizaje en redes neuronales</p> <p><b>04</b> Computer Vision</p> | <p><b>05</b> Convolución de imágenes</p> <p><b>06</b> Redes Neuronales Convolucionales</p> <p><b>07</b> Aplicación en el problema</p> <p><b>08</b> Casos de aplicación adicionales</p> |
|---|--|





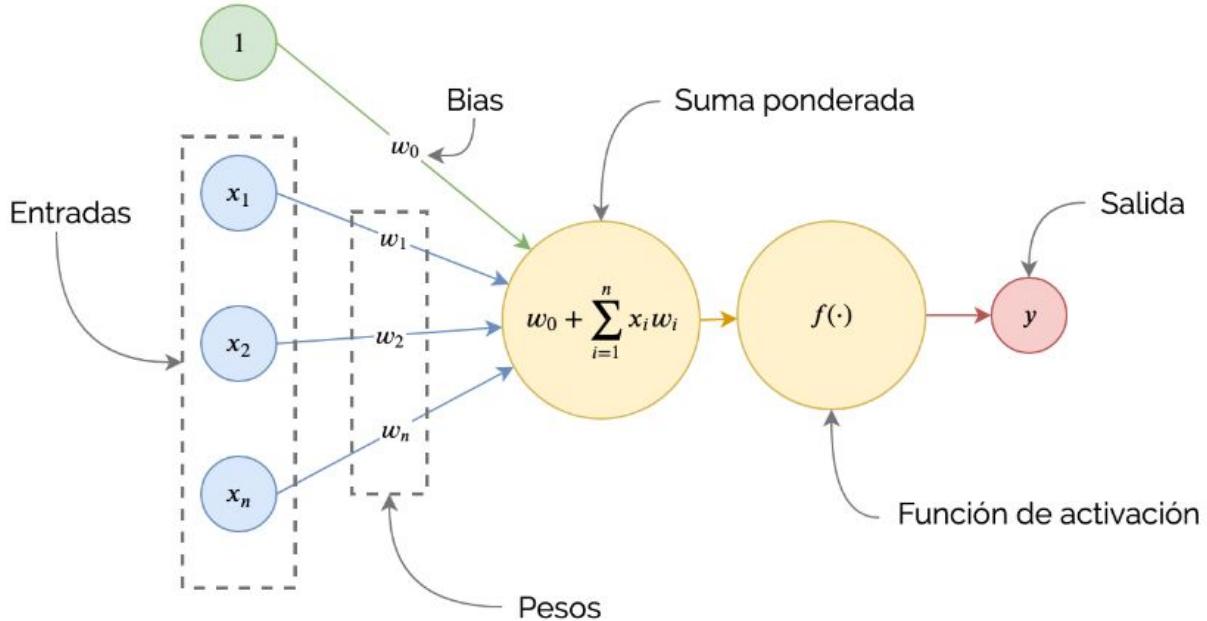
01



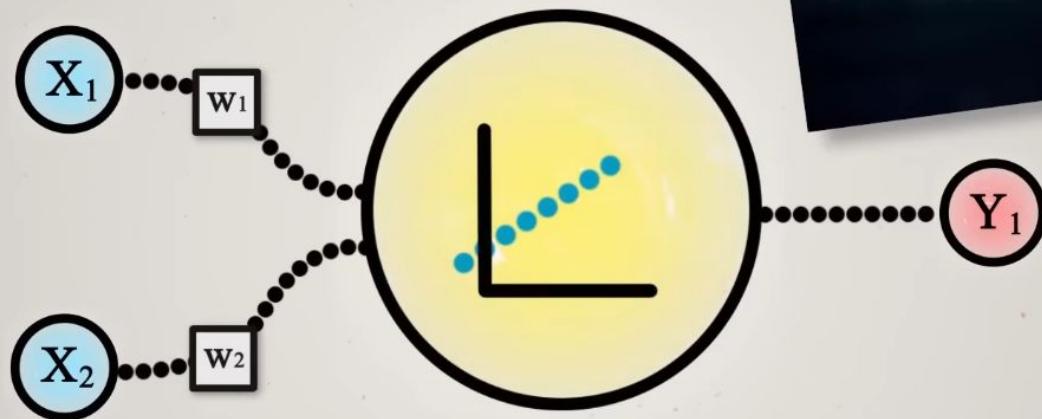
# Neuronas



# Las neuronas

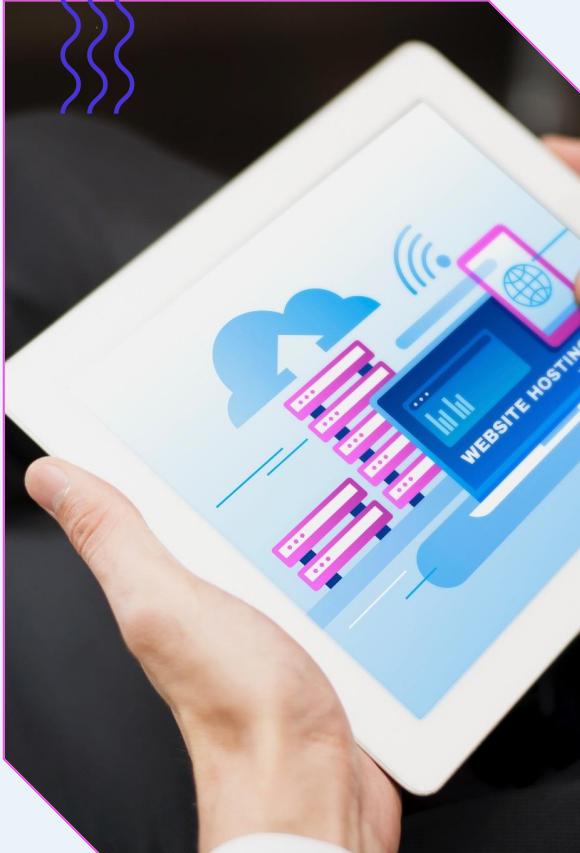


# Las neuronas



$$y = w_0 + w_1x_1 + w_2x_2$$

$$y = w_1x_1 + w_2x_2$$



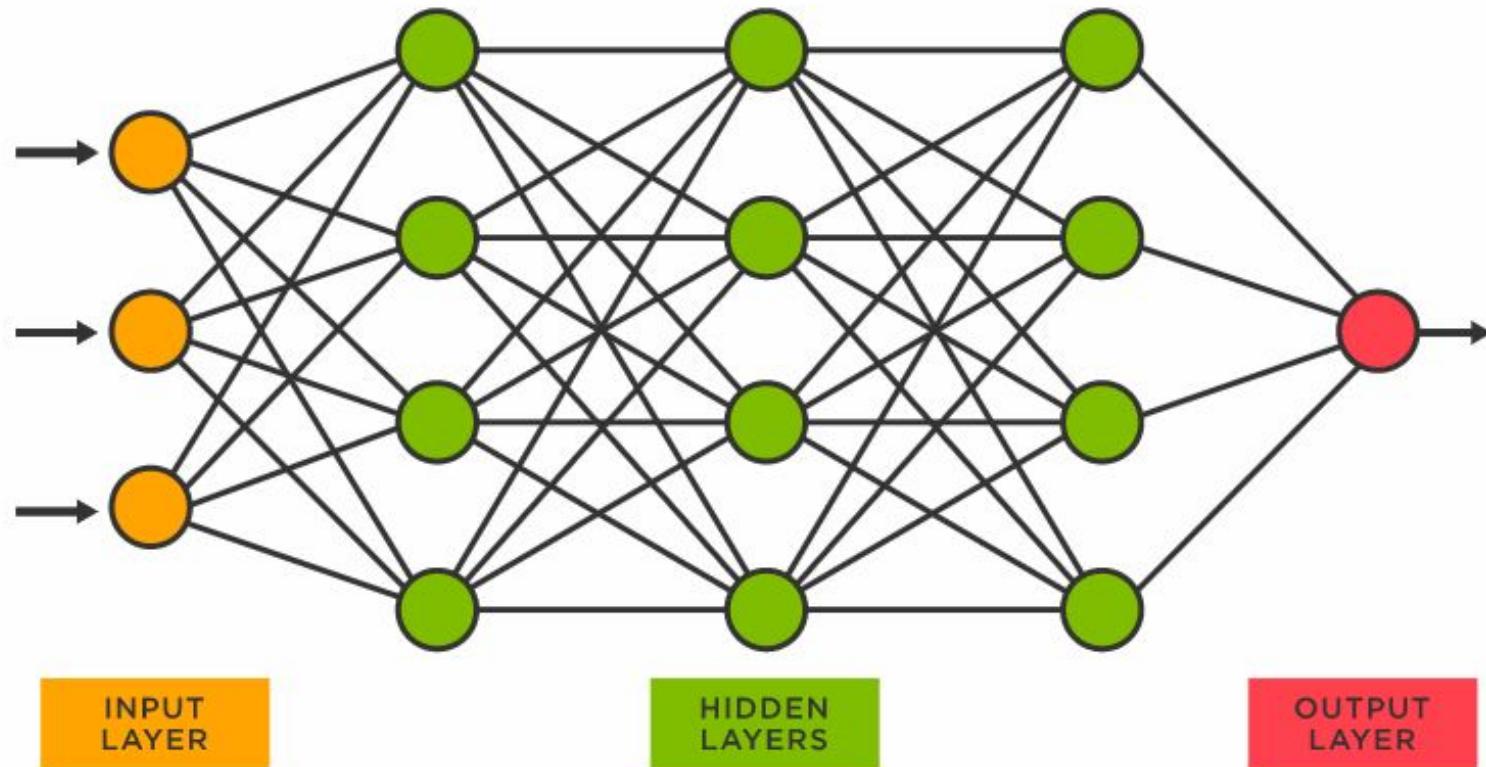
02



# Redes Neuronales



# Redes Neuronales





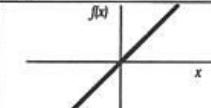
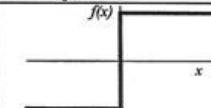
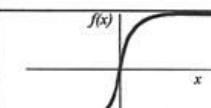
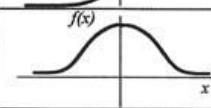
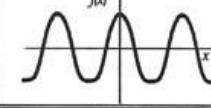
# Matemáticamente hablando...

$$\begin{array}{c} \text{L-shaped segments with blue dots} \\ + \\ \text{L-shaped segments with blue dots} \\ + \\ \text{L-shaped segments with blue dots} \\ = \\ \text{L-shaped segment with blue dots} \end{array}$$



# Funciones de activación

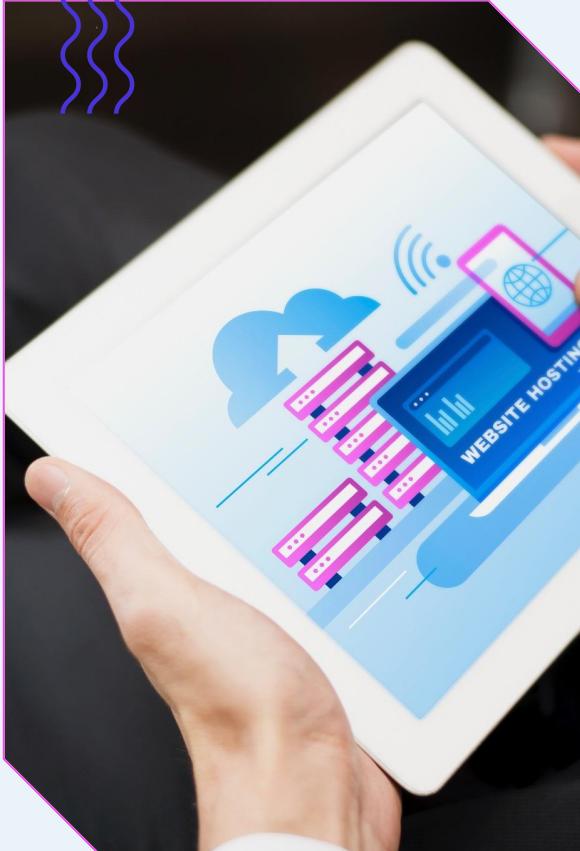
Filtro, función limitadora o umbral, que modifica el valor resultado o impone un límite que se debe sobrepasar para poder proseguir a otra neurona.

	Función	Rango	Gráfica
Identidad	$y = x$	$[-\infty, +\infty]$	
Escalón	$y = sign(x)$ $y = H(x)$	$\{-1, +1\}$ $\{0, +1\}$	
Lineal a tramos	$y = \begin{cases} -1, & \text{si } x < -l \\ x, & \text{si } -l \leq x \leq l \\ +1, & \text{si } x > +l \end{cases}$	$[-1, +1]$	
Sigmoidea	$y = \frac{1}{1+e^{-x}}$ $y = tgh(x)$	$[0, +1]$ $[-1, +1]$	
Gaussiana	$y = Ae^{-Bx^2}$	$[0, +1]$	
Sinusoidal	$y = A \sen(\omega x + \varphi)$	$[-1, +1]$	



**La función de activación nos permite distorsionar nuestra salida, añadiendo deformaciones no lineales**

xx



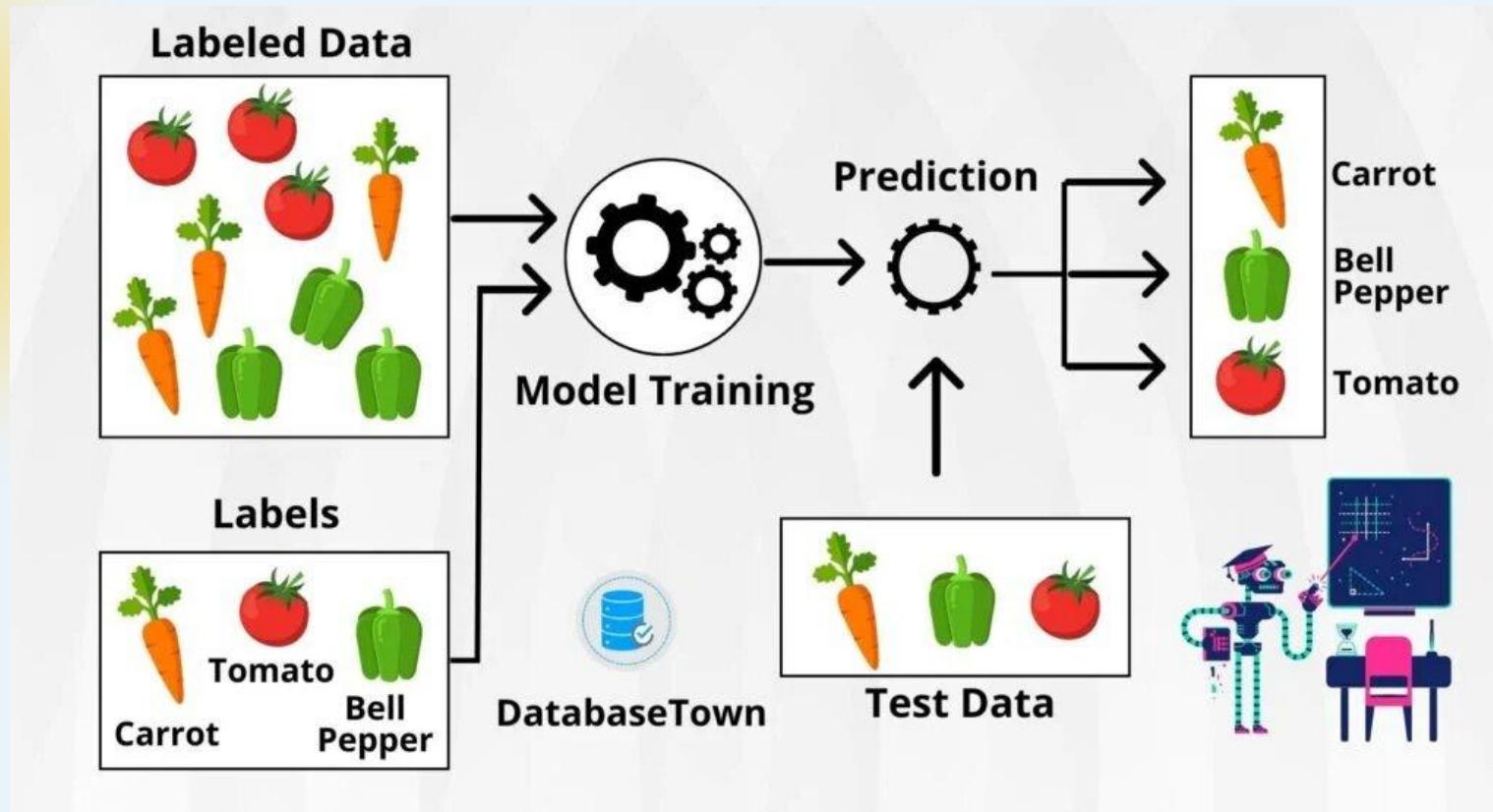
# 03



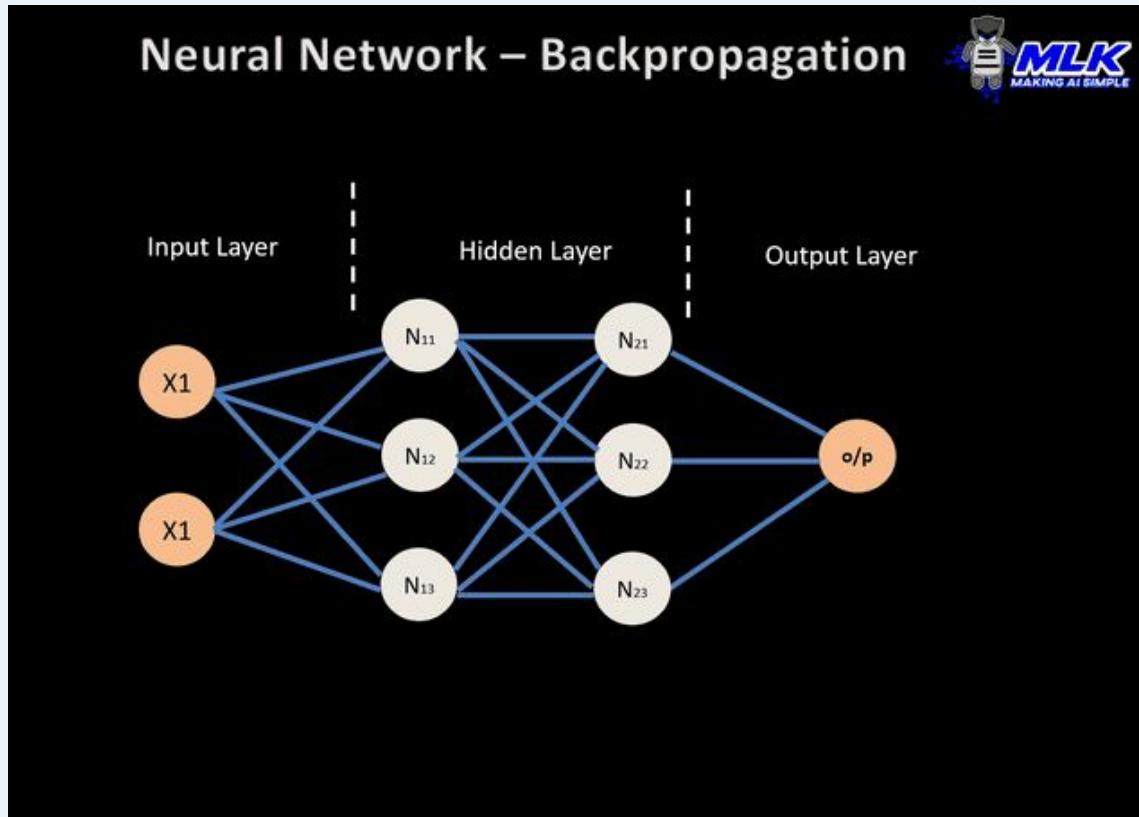
## Aprendizaje en redes neuronales



# Aprendizaje supervisado



# Backpropagation

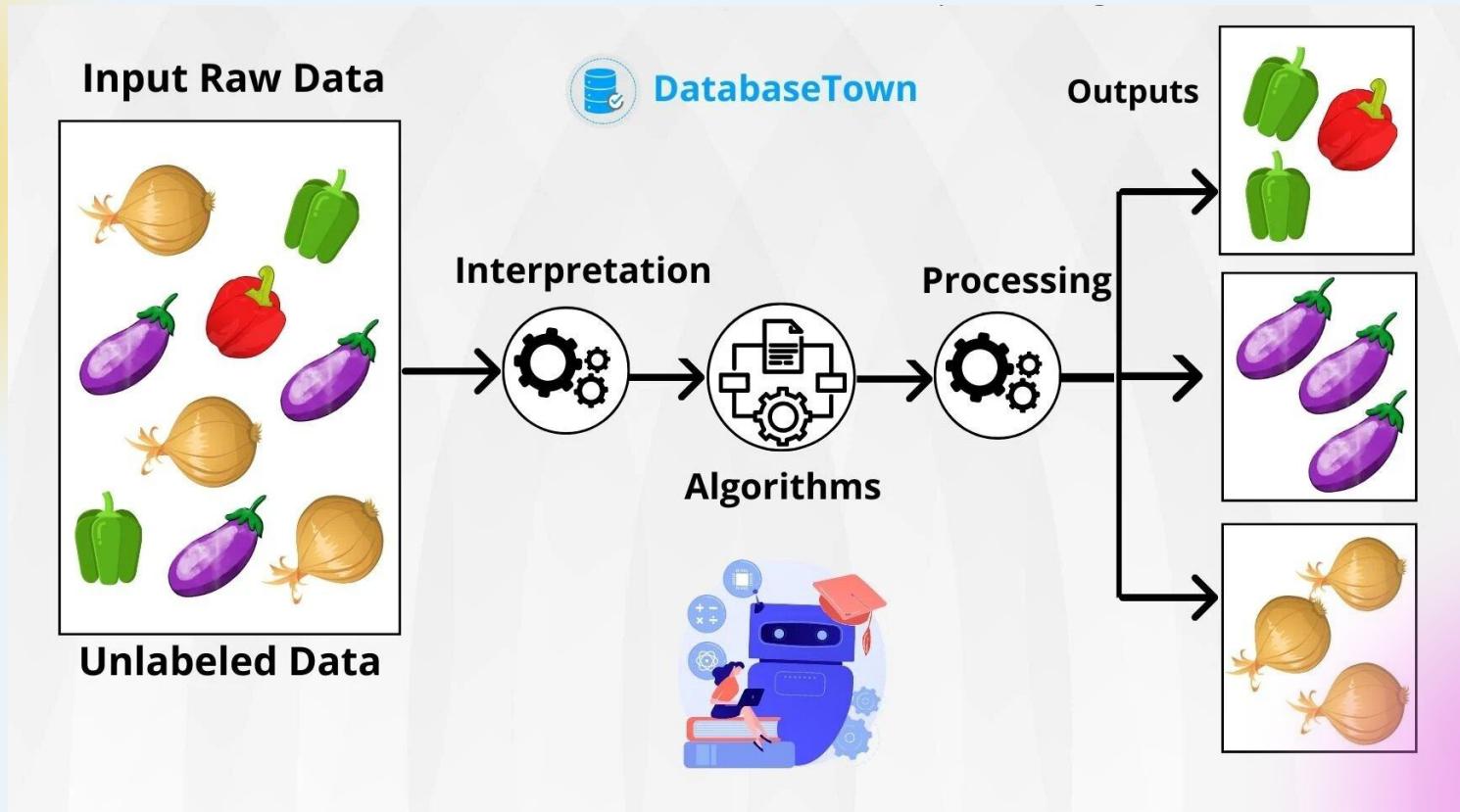




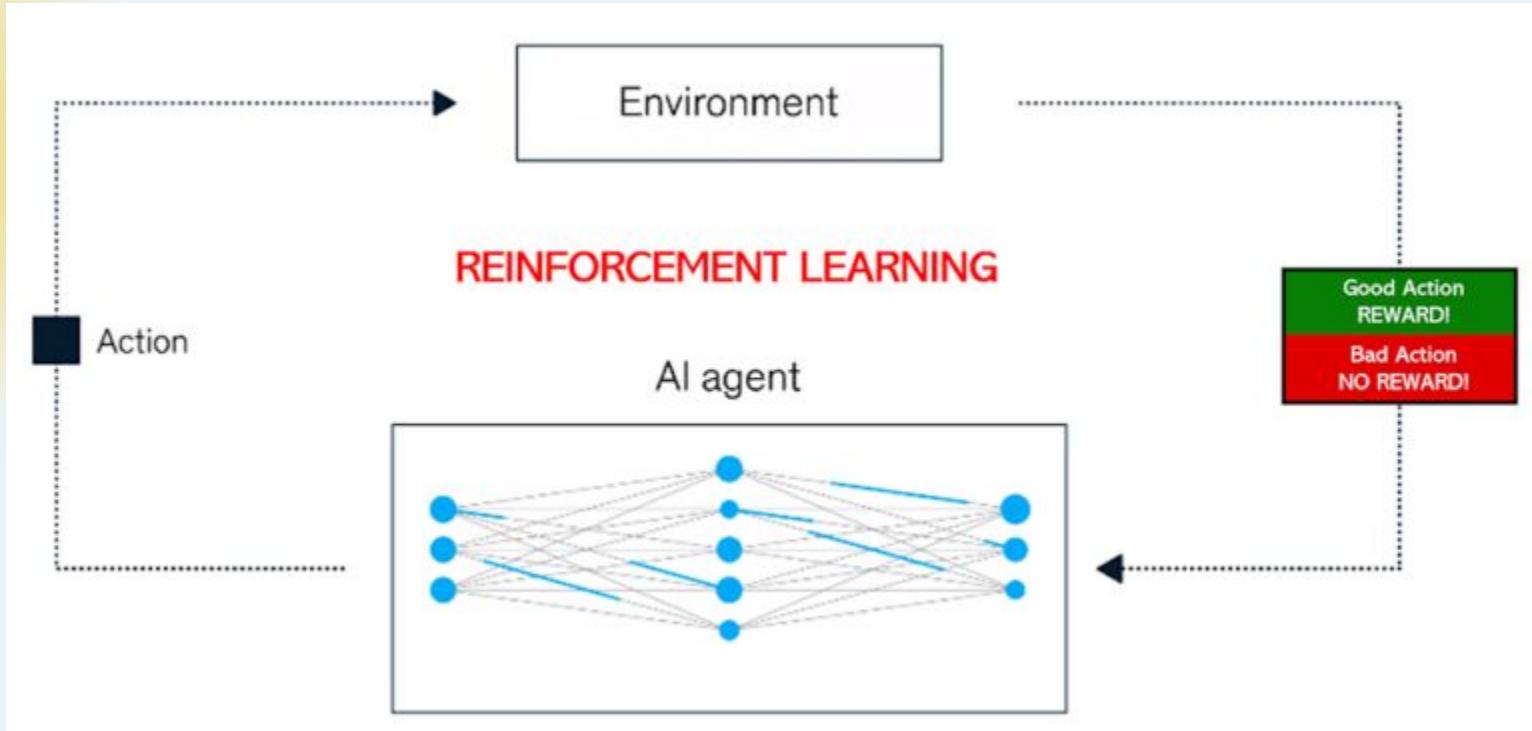
Otros aprendizajes...



# Aprendizaje no supervisado



# Aprendizaje reforzado





# 04



## Computer Vision

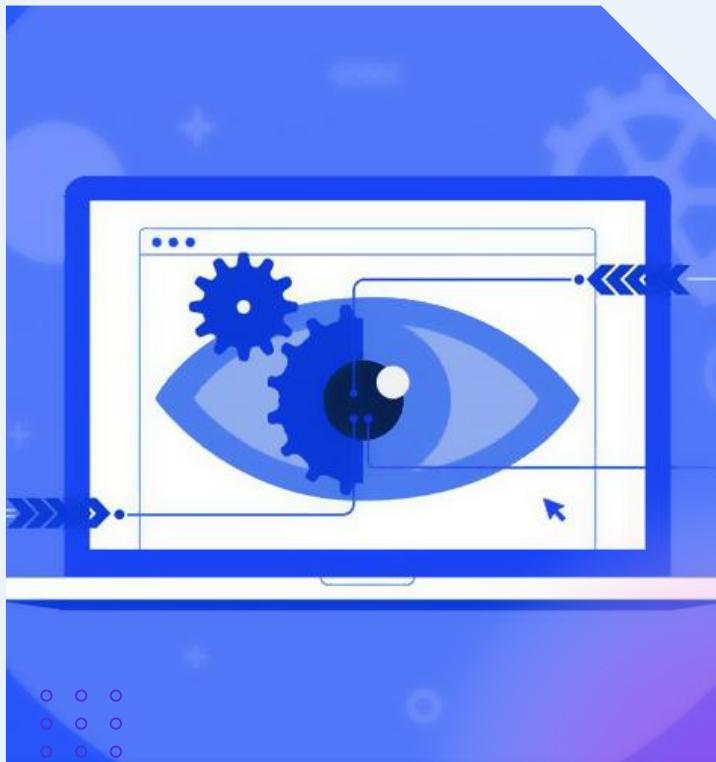


xx



# Computer Vision

Campo de Inteligencia Artificial e Informática cuyo objetivo es permitir a las máquinas interpretar y comprender el mundo visual que les rodea de la misma manera que lo hacen los seres humanos.





# 05



## Convolución de imágenes





**La convolución es una operación matemática fundamental en el procesamiento de imágenes que se utiliza para aplicar filtros a una imagen.**

xx



**Una imagen se representa como una matriz bidimensional de píxeles, donde cada píxel contiene información sobre el color o intensidad en ese punto específico de la imagen.**

# Convolución de imágenes

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

Imagen: 6x6

\*

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

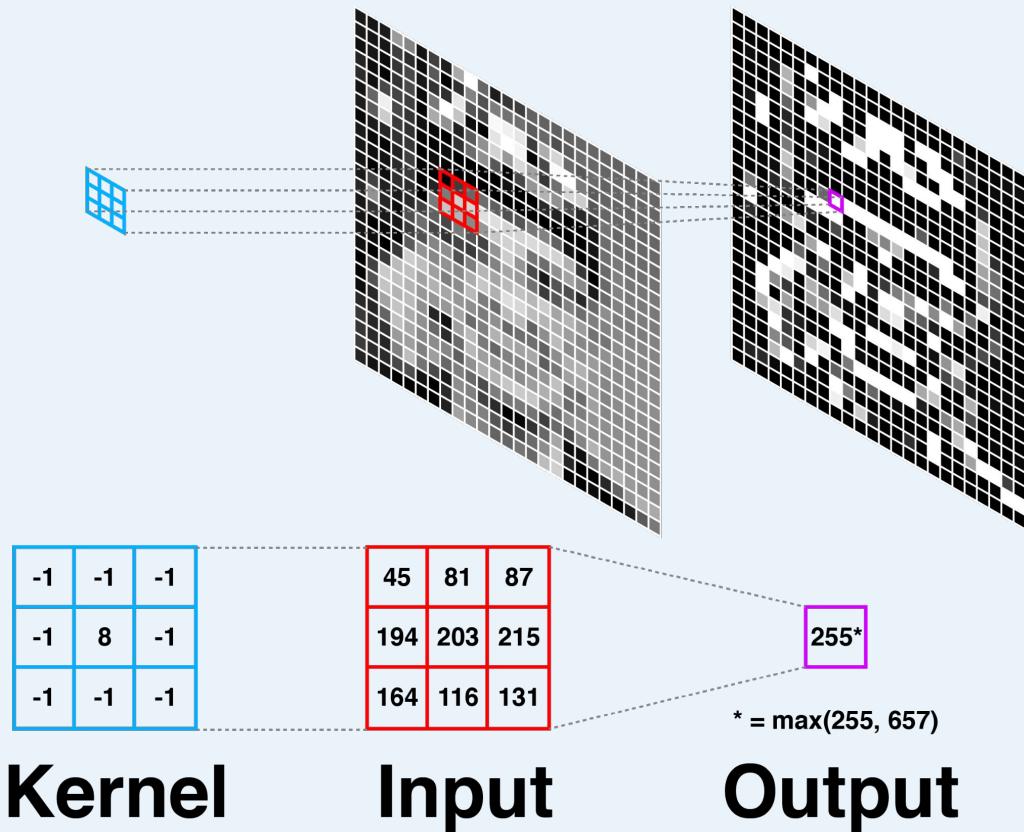
Filtro (kernel): 3x3

=

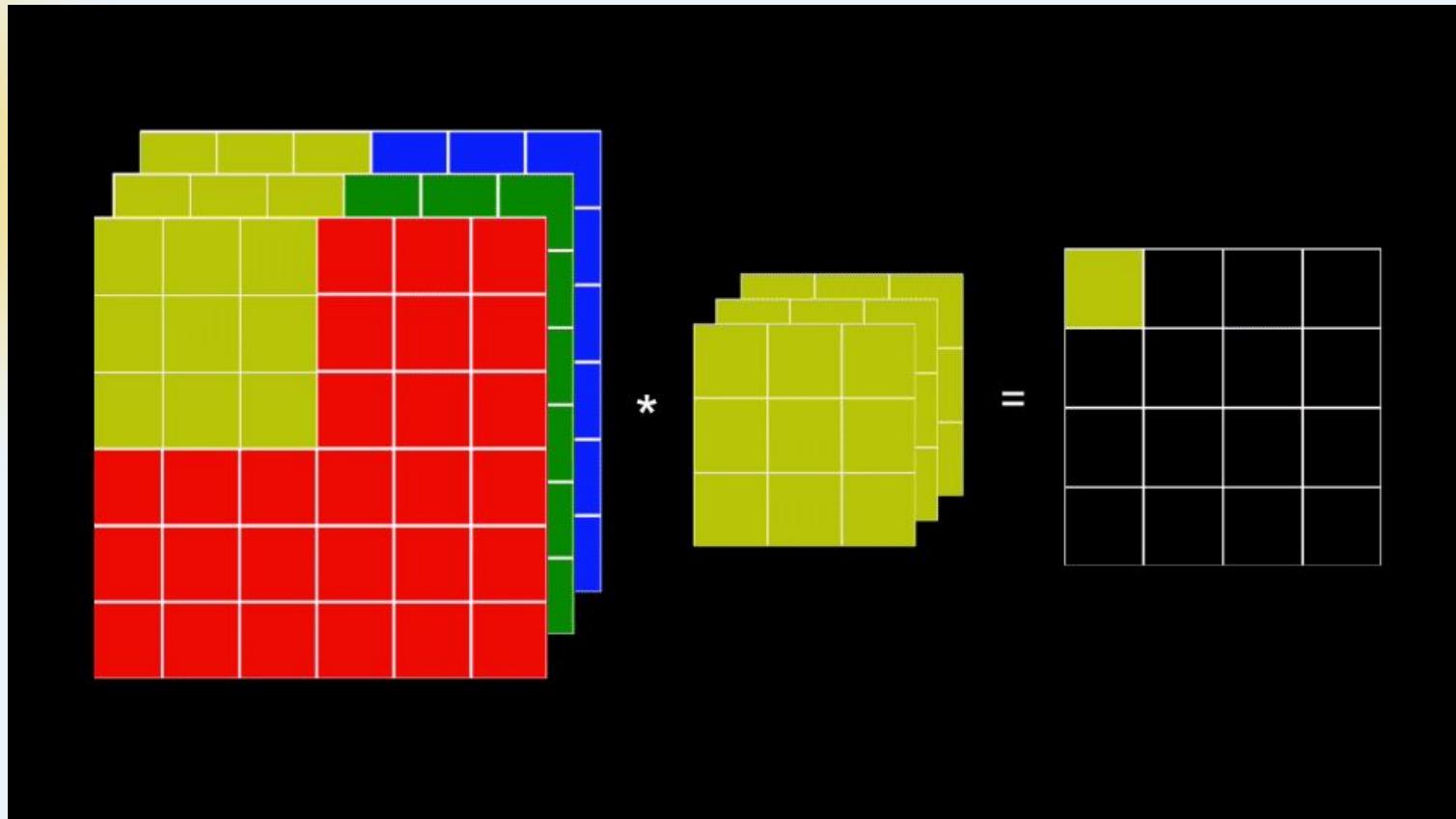
-5			

Resultado: 4x4

# Convolución de imágenes



# Convolución de imágenes



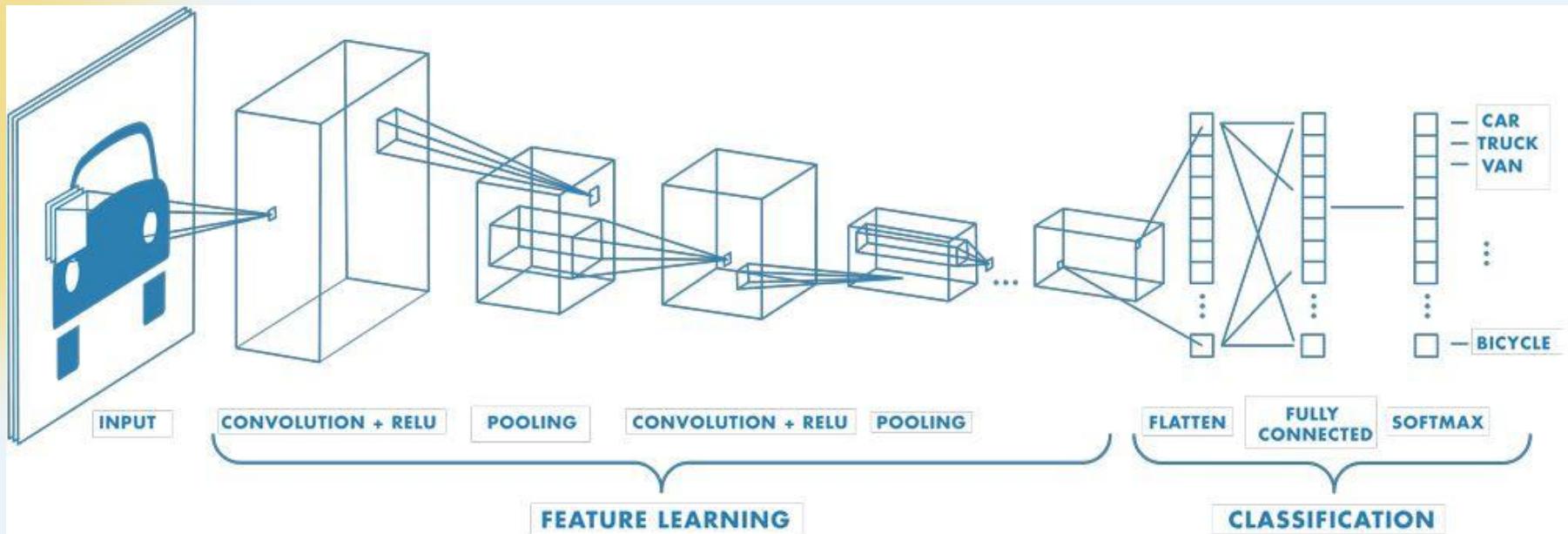


06



# Redes Neuronales Convolucionales

# Redes Neuronales Convolucionales



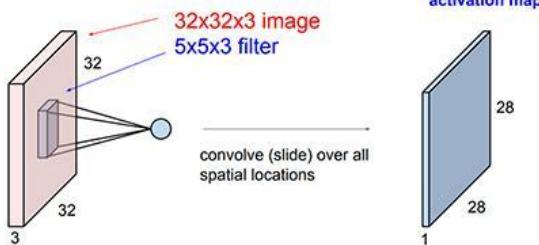


# Capa de convolucion

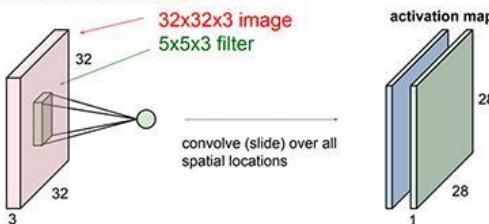
Filtros de pequeño tamaño, que van desplazándose por la imagen obteniendo las salidas de la capa. El filtro va recorriendo la imagen y obteniendo features relevantes reduciendo al mismo tiempo el tamaño de la imagen resultante.

# Capa de convolucion

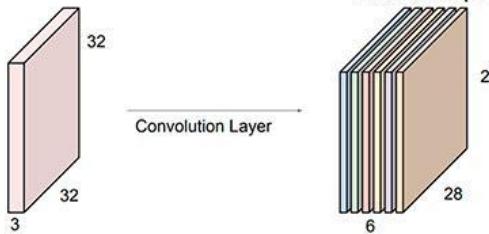
**PASO 1**



**PASO 2**



**PASO 3**



Fuente: Github.io

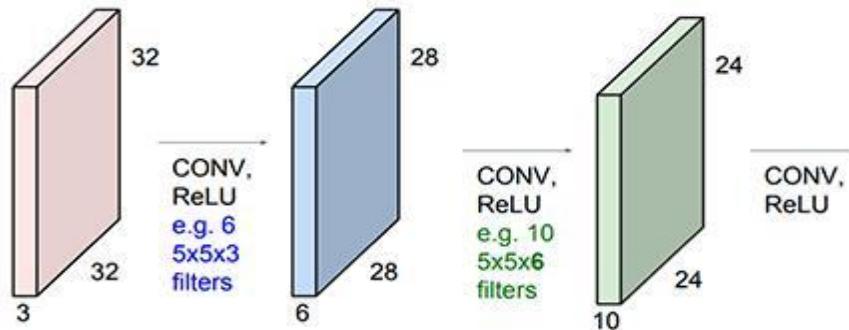


# Capa de convolucion

Lo más común es aplicar capas convolucionales seguidas de funciones de activación llamadas ReLU (Rectificador).

## PASO 4

Fuente: Github.io

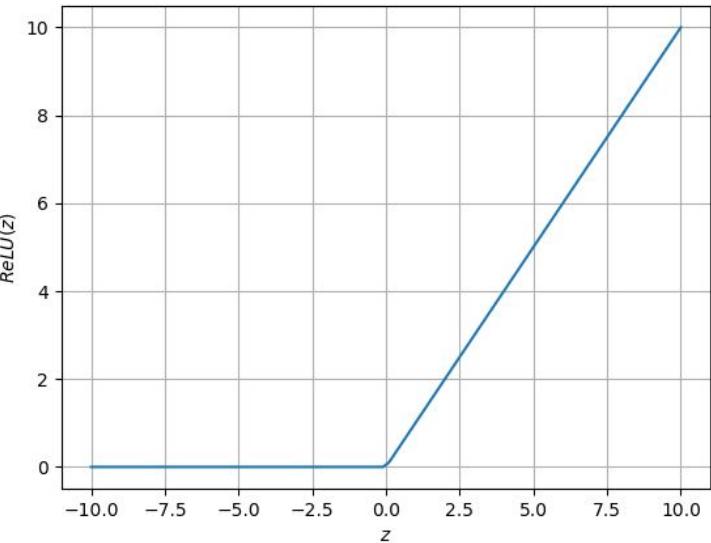


xx



# ReLU (Rectified Linear Unit)

Esta función generará una salida igual a cero cuando la entrada ( $z$ ) sea negativa, y una salida igual a la entrada cuando dicha esta última sea positiva.





# ReLU (Rectified Linear Unit)

Se ha convertido en una de las más usadas debido principalmente a:

- La no existencia de **saturación**, como sí ocurre en otras funciones. Lo anterior hace que el algoritmo del gradiente descendente converja mucho más rápidamente, facilitando así el entrenamiento.
- No es tan costosa computacionalmente.



# Saturación de funciones de activación

A la salida de las neuronas, y durante ciertas fases del proceso de entrenamiento de la red, los valores no cambian significativamente, lo que a su vez hace que el gradiente no cambie y que por tanto los parámetros (pesos) de la red (que se actualizan usando el gradiente descendente) no sufran cambios apreciables, lo que genera un "estancamiento" en el proceso de entrenamiento.



# Capa de convolucion

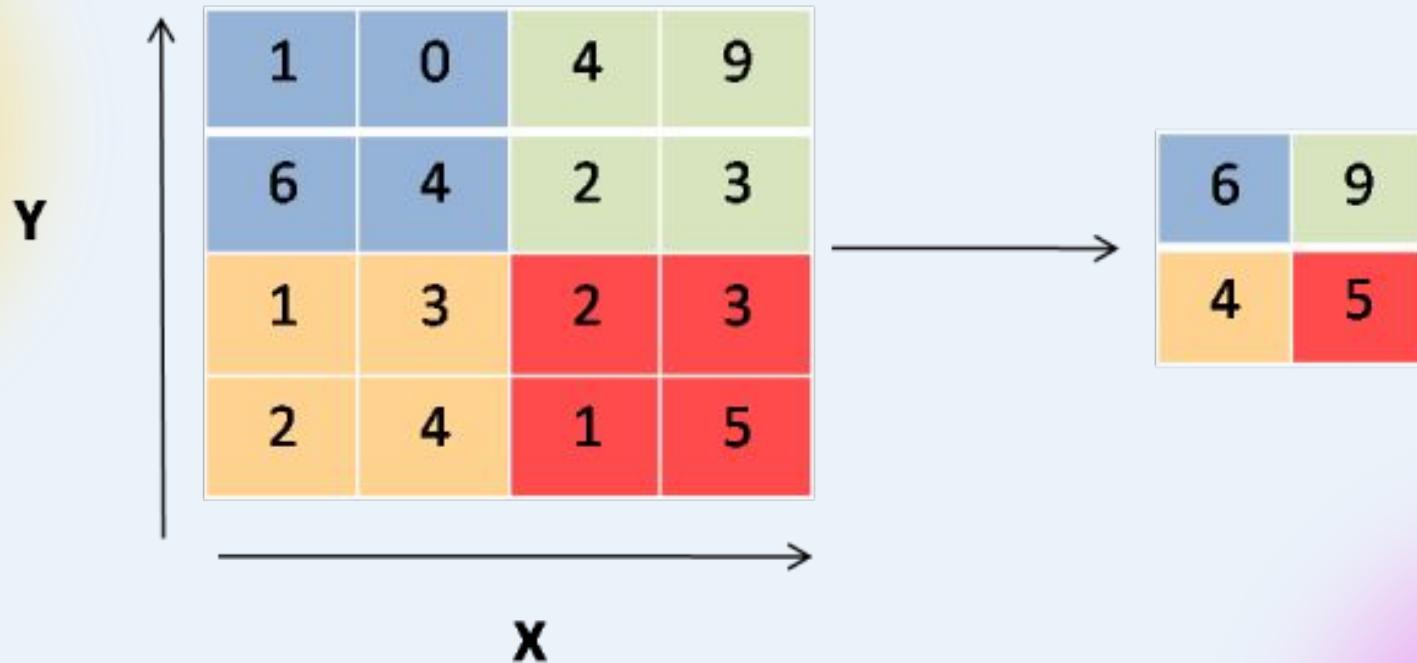
Las distintas capas van obteniendo una representación jerárquica de las features, con las primeras capas reconociendo elementos más simples en una imagen y las siguientes obteniendo representaciones de más alto nivel



# Capa de pooling

Su utilidad consiste en reducir las representaciones obtenidas de manera que estas se hagan más pequeñas y sean más manejables computacionalmente, reduciendo el número de parámetros necesarios.

# Capa de pooling





# Capas totalmente conectadas: Flattening

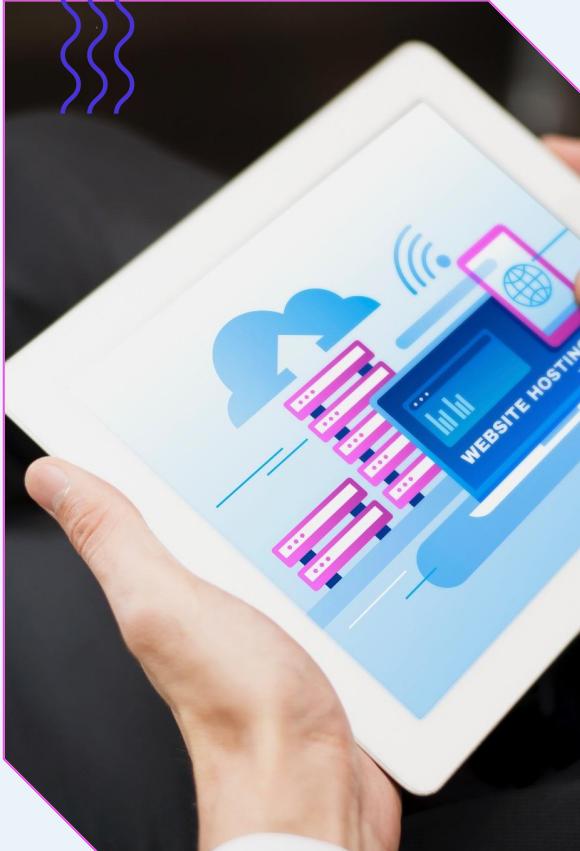
La capa de aplanado convierte la salida de las capas convolucionales en un vector unidimensional.

xx



# Capas totalmente conectadas: Densas

Las capas densas se utilizan comúnmente en la parte final de una red neuronal para realizar operaciones de aprendizaje basadas en las características extraídas en las capas anteriores. La última de estas capas contiene una función de activación softmax, la cual calcula la distribución de probabilidades.



07



## Aplicación en el problema





# Problematica

Construir una red neuronal para clasificar las señales de tráfico basadas en imágenes de las mismas.

xx



## Datos a tener en cuenta

- Se alimenta a la red con imágenes a color de un tamaño de 30x30
- Hay en total 43 categorías de señales
- Se tomarán 10 epochs para el entrenamiento

xx



# Metricas Utilizadas

## Precisión

La precisión se refiere a la fracción de predicciones correctas en el conjunto de datos de prueba. Valores cercanos a 1.0 (100%) indicaría que las predicciones son correctas, mientras que un valor de precisión cercano a 0.0 (0%) indicaría que las predicciones son incorrectas.

XX

## Perdida: Entropía cruzada categórica

La entropía cruzada categórica es una función de pérdida utilizada en problemas de clasificación. Mide cuán bien el modelo está asignando probabilidades a las clases correctas. Valores cercanos a 0 indican que el modelo tiene una alta confianza en sus predicciones, mientras que valores más altos indican mayor incertidumbre en las predicciones.



# Demostración de librería TensorFlow Keras

```
# Crear un modelo secuencial
model = models.Sequential()
# Capa convolucional de entrada
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(IMG_WIDTH,
IMG_HEIGHT, 3)))
# Capa de max-pooling
model.add(layers.MaxPooling2D((2, 2)))
# Capa convolucional adicional
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
# Otra capa de max-pooling
model.add(layers.MaxPooling2D((2, 2)))
# Capa de aplanado
model.add(layers.Flatten())
# Capas densas (totalmente conectadas)
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(NUM_CATEGORIES, activation='softmax'))
```



## Distribuciones y resultados





# Distribucion 1

- Capa convolucional de entrada con 32 filtros de  $3 \times 3$
- Capa de max pooling de  $2 \times 2$
- Capa convolucional con 64 filtros de  $3 \times 3$
- Capa de max pooling de  $2 \times 2$
- Capa de aplanado
- Capa densa de 128 neuronas
- Capa densa con softmax

xx



# Resultados 1

```
Epoch 1/10
500/500 [=====] - 12s 23ms/step - loss: 1.6425 - accuracy: 0.6713
Epoch 2/10
500/500 [=====] - 12s 25ms/step - loss: 0.3059 - accuracy: 0.9204
Epoch 3/10
500/500 [=====] - 11s 23ms/step - loss: 0.1504 - accuracy: 0.9604
Epoch 4/10
500/500 [=====] - 11s 23ms/step - loss: 0.1398 - accuracy: 0.9620
Epoch 5/10
500/500 [=====] - 12s 24ms/step - loss: 0.1194 - accuracy: 0.9680
Epoch 6/10
500/500 [=====] - 13s 26ms/step - loss: 0.1093 - accuracy: 0.9700
Epoch 7/10
500/500 [=====] - 13s 25ms/step - loss: 0.1006 - accuracy: 0.9759
Epoch 8/10
500/500 [=====] - 13s 27ms/step - loss: 0.0593 - accuracy: 0.9837
Epoch 9/10
500/500 [=====] - 14s 27ms/step - loss: 0.1100 - accuracy: 0.9741
Epoch 10/10
500/500 [=====] - 13s 25ms/step - loss: 0.0968 - accuracy: 0.9775
333/333 - 2s - loss: 0.2462 - accuracy: 0.9608 - 2s/epoch - 7ms/step
```



## Distribucion 2: Variación de filtros

- Capa convolucional de entrada con 16 filtros de  $3 \times 3$
- Capa de max pooling de  $2 \times 2$
- Capa convolucional con 32 filtros de  $3 \times 3$
- Capa de max pooling de  $2 \times 2$
- Capa de aplanado
- Capa densa de 128 neuronas
- Capa densa con softmax

xx



## Resultados 2

```
Epoch 1/10
500/500 [=====] - 7s 13ms/step - loss: 1.7992 - accuracy: 0.6129
Epoch 2/10
500/500 [=====] - 7s 13ms/step - loss: 0.3275 - accuracy: 0.9152
Epoch 3/10
500/500 [=====] - 7s 13ms/step - loss: 0.1675 - accuracy: 0.9560
Epoch 4/10
500/500 [=====] - 7s 14ms/step - loss: 0.1203 - accuracy: 0.9693
Epoch 5/10
500/500 [=====] - 7s 14ms/step - loss: 0.1029 - accuracy: 0.9738
Epoch 6/10
500/500 [=====] - 7s 14ms/step - loss: 0.0874 - accuracy: 0.9771
Epoch 7/10
500/500 [=====] - 7s 14ms/step - loss: 0.0719 - accuracy: 0.9815
Epoch 8/10
500/500 [=====] - 7s 14ms/step - loss: 0.0620 - accuracy: 0.9843
Epoch 9/10
500/500 [=====] - 7s 14ms/step - loss: 0.0661 - accuracy: 0.9842
Epoch 10/10
500/500 [=====] - 7s 14ms/step - loss: 0.0598 - accuracy: 0.9837
333/333 - 2s - loss: 0.2736 - accuracy: 0.9459 - 2s/epoch - 5ms/step
```



## Distribucion 3: Más capas de convolución

- Capa convolucional de entrada con 32 filtros de  $3 \times 3$
- Capa de max pooling de  $2 \times 2$
- 2 Capas convolucionales con 64 filtros de  $3 \times 3$
- 2 Capas de max pooling de  $2 \times 2$
- Capa de aplanado
- Capa densa de 128 neuronas
- Capa densa con softmax



# Resultados 3

```
Epoch 1/10
500/500 [=====] - 12s 22ms/step - loss: 1.8200 - accuracy: 0.5745
Epoch 2/10
500/500 [=====] - 11s 23ms/step - loss: 0.4156 - accuracy: 0.8902
Epoch 3/10
500/500 [=====] - 11s 22ms/step - loss: 0.2105 - accuracy: 0.9445
Epoch 4/10
500/500 [=====] - 11s 23ms/step - loss: 0.1521 - accuracy: 0.9609
Epoch 5/10
500/500 [=====] - 11s 22ms/step - loss: 0.1254 - accuracy: 0.9658
Epoch 6/10
500/500 [=====] - 11s 23ms/step - loss: 0.1297 - accuracy: 0.9648
Epoch 7/10
500/500 [=====] - 11s 22ms/step - loss: 0.1073 - accuracy: 0.9715
Epoch 8/10
500/500 [=====] - 12s 23ms/step - loss: 0.0824 - accuracy: 0.9787
Epoch 9/10
500/500 [=====] - 11s 23ms/step - loss: 0.1122 - accuracy: 0.9715
Epoch 10/10
500/500 [=====] - 11s 22ms/step - loss: 0.0857 - accuracy: 0.9796
333/333 - 2s - loss: 0.2123 - accuracy: 0.9545 - 2s/epoch - 7ms/step
```



## Distribucion 4: Variacion de pooling

- Capa convolucional de entrada con 32 filtros de  $3 \times 3$
- Capa de max pooling de  $3 \times 3$
- Capa convolucional con 64 filtros de  $3 \times 3$
- Capa de max pooling de  $3 \times 3$
- Capa de aplanado
- Capa densa de 128 neuronas
- Capa densa con softmax

xx



# Resultados 4

```
Epoch 1/10
500/500 [=====] - 8s 14ms/step - loss: 3.3293 - accuracy: 0.3388
Epoch 2/10
500/500 [=====] - 7s 14ms/step - loss: 0.9746 - accuracy: 0.7237
Epoch 3/10
500/500 [=====] - 7s 14ms/step - loss: 0.5223 - accuracy: 0.8547
Epoch 4/10
500/500 [=====] - 7s 14ms/step - loss: 0.3462 - accuracy: 0.9060
Epoch 5/10
500/500 [=====] - 7s 15ms/step - loss: 0.2557 - accuracy: 0.9311
Epoch 6/10
500/500 [=====] - 8s 16ms/step - loss: 0.2475 - accuracy: 0.9338
Epoch 7/10
500/500 [=====] - 8s 16ms/step - loss: 0.1906 - accuracy: 0.9466
Epoch 8/10
500/500 [=====] - 8s 16ms/step - loss: 0.1593 - accuracy: 0.9581
Epoch 9/10
500/500 [=====] - 8s 17ms/step - loss: 0.1876 - accuracy: 0.9513
Epoch 10/10
500/500 [=====] - 7s 15ms/step - loss: 0.1764 - accuracy: 0.9541
333/333 - 2s - loss: 0.2557 - accuracy: 0.9443 - 2s/epoch - 5ms/step
```



## Distribucion 5: Dropout

- Capa convolucional de entrada con 32 filtros de  $3 \times 3$
- Capa de max pooling de  $2 \times 2$
- Capa convolucional con 64 filtros de  $3 \times 3$
- Capa de max pooling de  $2 \times 2$
- Capa de aplanado
- Capa densa de 128 neuronas
- Capa de dropout con tasa de 0,5
- Capa densa con softmax

xx



# Resultados 5

```
Epoch 1/10
500/500 [=====] - 13s 24ms/step - loss: 3.9429 - accuracy: 0.0798
Epoch 2/10
500/500 [=====] - 12s 24ms/step - loss: 2.2188 - accuracy: 0.3814
Epoch 3/10
500/500 [=====] - 12s 24ms/step - loss: 1.1800 - accuracy: 0.6427
Epoch 4/10
500/500 [=====] - 12s 24ms/step - loss: 0.7811 - accuracy: 0.7586
Epoch 5/10
500/500 [=====] - 13s 27ms/step - loss: 0.5910 - accuracy: 0.8213
Epoch 6/10
500/500 [=====] - 12s 24ms/step - loss: 0.4472 - accuracy: 0.8625
Epoch 7/10
500/500 [=====] - 12s 23ms/step - loss: 0.4237 - accuracy: 0.8759
Epoch 8/10
500/500 [=====] - 13s 26ms/step - loss: 0.3463 - accuracy: 0.8993
Epoch 9/10
500/500 [=====] - 14s 28ms/step - loss: 0.2985 - accuracy: 0.9125
Epoch 10/10
500/500 [=====] - 18s 37ms/step - loss: 0.2714 - accuracy: 0.9185
333/333 - 2s - loss: 0.1392 - accuracy: 0.9663 - 2s/epoch - 7ms/step
```



## Conclusiones





# Conclusiones

- Distribución 1:
  - Obtiene una alta precisión en el conjunto de prueba (accuracy: 0.9608) y una pérdida baja (loss: 0.2462). Esto sugiere que la red es capaz de realizar una buena clasificación en el conjunto de datos de prueba.



# Conclusiones

- Distribución 2 (Variación de filtros):
  - Aunque la precisión en el conjunto de prueba es un poco menor que en la Distribución 1 (accuracy: 0.9459), sigue siendo bastante buena, y la pérdida es también razonable (loss: 0.2736).



# Conclusiones

- Distribución 3 (Más capas de convolución):
  - La precisión en el conjunto de prueba es más baja que en las dos distribuciones anteriores (accuracy: 0.9545). La pérdida también es un poco más alta (loss: 0.2123).
  - Esto puede indicar que la red está sobreajustando los datos de entrenamiento y no generaliza tan bien en los datos de prueba.



# Conclusiones

- Distribución 4 (Variacion de pooling):
  - La precisión en el conjunto de prueba (accuracy: 0.9443) es bastante buena, pero la pérdida (loss: 0.2557) es un poco más alta en comparación con las Distribuciones 1 y 2.



# Conclusiones

- Distribución 5 (Dropout):
  - En épocas tempranas la pérdida fue una de las más altas (3.9429) y la precisión de las más bajas (0.0798), pero al final la precisión en el conjunto de prueba (accuracy: 0.9663) es la más alta entre las distribuciones que proporcionaste, y la pérdida (loss: 0.1392) es la más baja.



# Conclusiones

- En general, la Distribución 1 parece ser una elección sólida, ya que ofrece una buena precisión y una pérdida razonable en el conjunto de prueba y un equilibrio entre resultados y costo computacional.
- Si la precisión en el conjunto de prueba es crítica y estás dispuesto a asumir un mayor costo computacional, la Distribución 5 también es una opción sólida.



07



## Casos de aplicación adicionales





## Diagnóstico médico

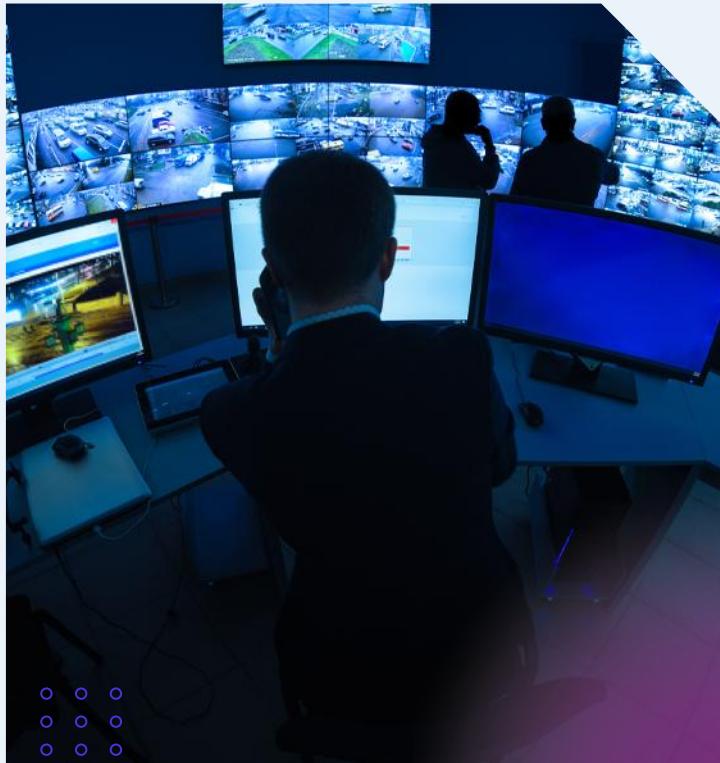
Las CNN se aplican en el análisis de imágenes médicas, como radiografías, tomografías computarizadas y resonancias magnéticas, para detectar enfermedades, tumores, fracturas óseas, etc.





## Análisis de videos

Las CNN se aplican para la detección de eventos en videos, seguimiento de objetos, clasificación de acciones humanas y más para obtener información relevante en investigaciones criminales, juicios legales o para otros fines relacionados con la seguridad y la aplicación de la ley.





**Gracias por  
su  
atencion!**

