

# Tema 5.2

# Redes Convolucionales

Deep Learning

Máster Oficial en Ingeniería Informática

Universidad de Sevilla

# Capas de una red convolucional

- Entrada
- Convolucional (Conv)
- ReLU
- Pooling
- Batch normalization (BatchNorm)
- Dropout
- Completamente conectada (Fully Connected, o simplemente, FC)

# Capas de una red convolucional

- ~~Entrada~~
- **Convolucional (Conv)**
- ~~ReLU~~
- **Pooling**
- **Batch normalization (BatchNorm)**
- ~~Dropout~~
- ~~Completamente conectada (Fully Connected, o simplemente, FC)~~

# Contenido

- Convolución
- Pooling
- Batch normalization
- Interpretando redes convolucionales

# Convolución

- Aspectos a considerar en una red multicapa:
  - **Parámetros:** topología, pesos conexiones...
  - **Optimización:** obtención de los pesos.
  - **Generalización:** conseguir que la red trabaje bien con ejemplos que no pertenezcan al conjunto de entrenamiento.
  - **Invarianza:** conseguir que la red sea robusta frente a transformaciones comunes en los datos
    - **Segmentación:** más de un objeto en la imagen.
    - **Ocultación:** partes de los objetos ocultas por otros.
    - **Iluminación, punto de vista, deformación, ...**

Translation Invariance



Rotation/Viewpoint Invariance



Size Invariance



Illumination Invariance



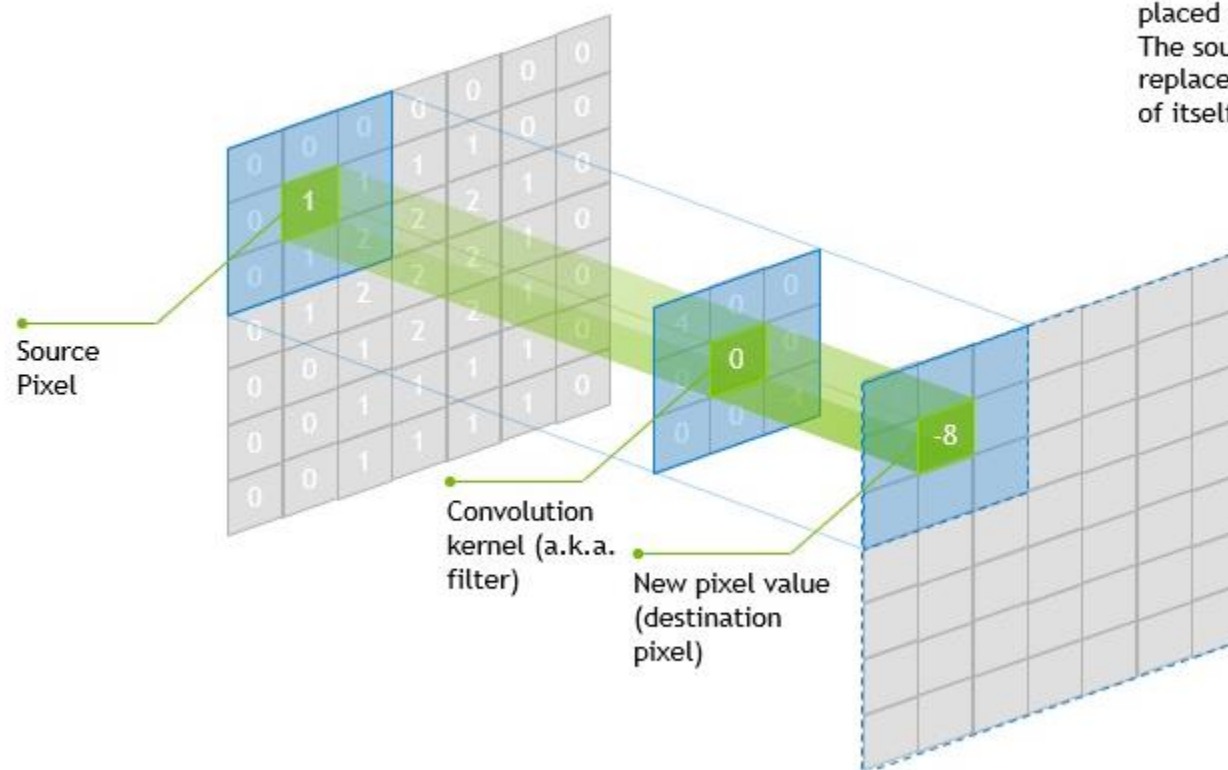
# Convolución

- **Convolución** es una operación muy utilizada en matemáticas, física y teoría de señales
  - Un tipo especial de operación lineal
- **Red convolucional (CNN, ConvNet)**: redes neuronales que usan la operación de convolución en lugar de regresión lineal en alguna de sus capas.
- **Elementos** de la operación:
  - Una matriz de **entrada**: datos de entrada o salida de capa anterior
  - Una pequeña **matriz**: kernel de convolución, filtro o máscara
  - **Devuelve** una **matriz** con un tamaño aproximado al de la entrada: mapa de activaciones, mapa de características o **feature map**.

# Convolución

- Operación básica:
  - Multiplicación elemento a elemento
  - Suma de las multiplicaciones

## CONVOLUTION



Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.

# Convolución

- Proceso sobre la matriz completa:
  - Para cada posible encaje del kernel en la matriz de entrada
    - Aplicar operación de convolución
    - Anotar el valor de salida en la matriz resultado

1 <sub>x1</sub>	1 <sub>x0</sub>	1 <sub>x1</sub>	0	0
0 <sub>x0</sub>	1 <sub>x1</sub>	1 <sub>x0</sub>	1	0
0 <sub>x1</sub>	0 <sub>x0</sub>	1 <sub>x1</sub>	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved  
Feature

# Convolución

- Ejemplos de convolución sobre imágenes

Input image

Convolution Kernel

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Feature map

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

Identidad

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Detección bordes

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Sharpen (enfoco)

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Difuminado (Blur)

$$\frac{1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Difuminado gaussiano 5x5

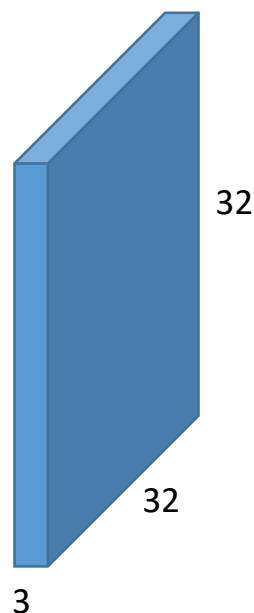
$$\begin{bmatrix} & & & & \\ & -2 & -1 & 0 & \\ & -1 & 1 & 1 & \\ & 0 & 1 & 2 & \\ & & & & \end{bmatrix}$$

Repujado

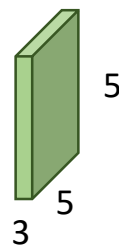
# Neuronas convolucionales

- Los filtros o kernels convolucionan sobre toda la entrada
- Los filtros siempre cubren toda la profundidad de la entrada

Imagen 32x32x3



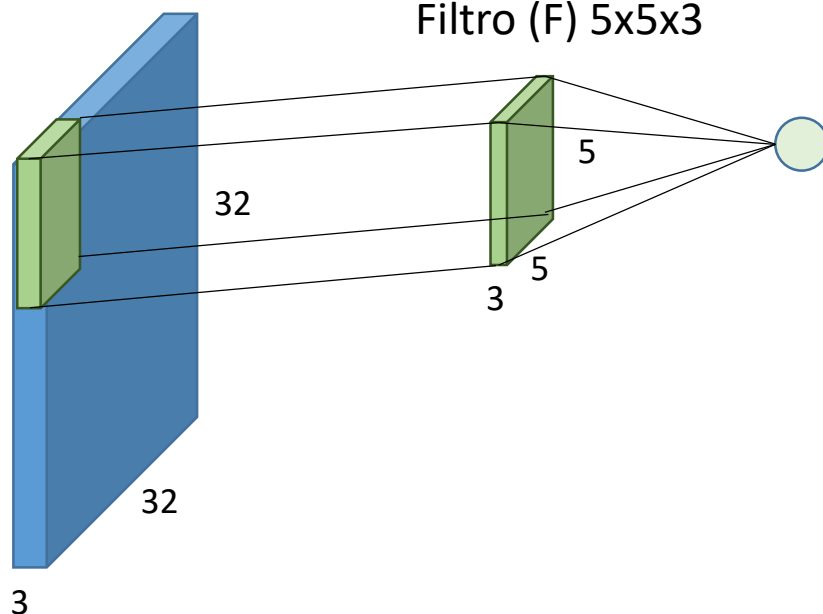
Filtro 5x5x3



# Neuronas convolucionales

- La operación de convolución aplicada a una sola porción de una imagen da un número

Imagen (I) 32x32x3



Un solo número:

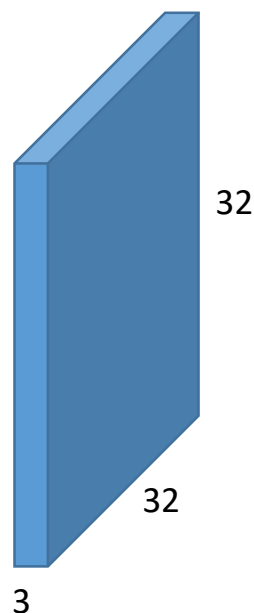
$$O_{x,y} = b + \sum_i^5 \sum_j^5 \sum_k^3 I_{x+i,y+j,z+k} * F_{i,j,k}$$

5\*5\*3 productos + 1 bias.

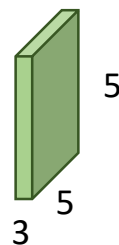
# Neuronas convolucionales

- Al aplicar el filtro a través de toda la imagen obtenemos un mapa de activación de la característica que el éste codifica.

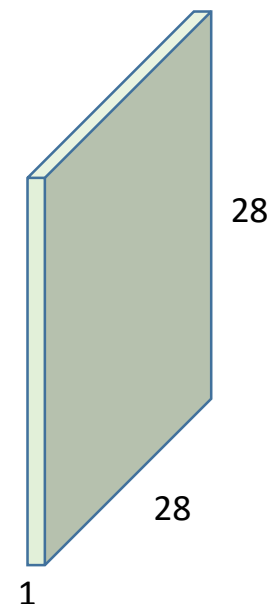
Imagen 32x32x3



Filtro 5x5x3



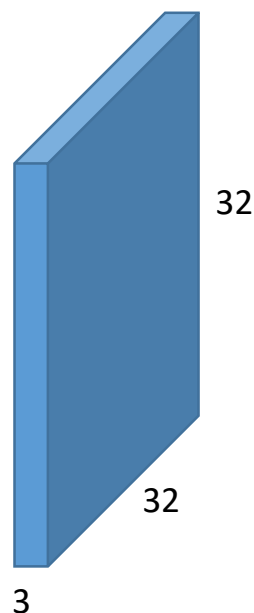
Mapa activación 28x28x1



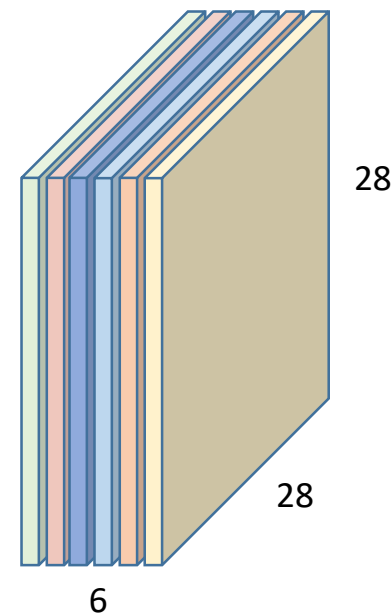
# Neuronas convolucionales

- Podemos tener más filtros, por ejemplo, 6, obteniendo una “nueva imagen” de  $28 \times 28 \times 6$ .

Imagen  $32 \times 32 \times 3$

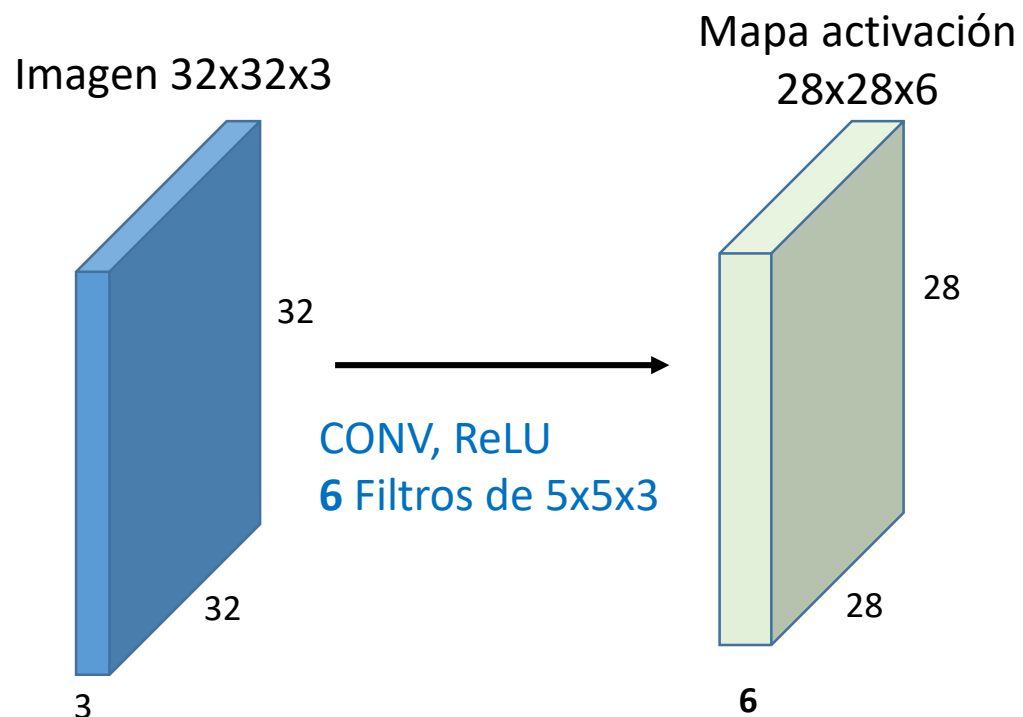


Mapa activación  $28 \times 28 \times 6$



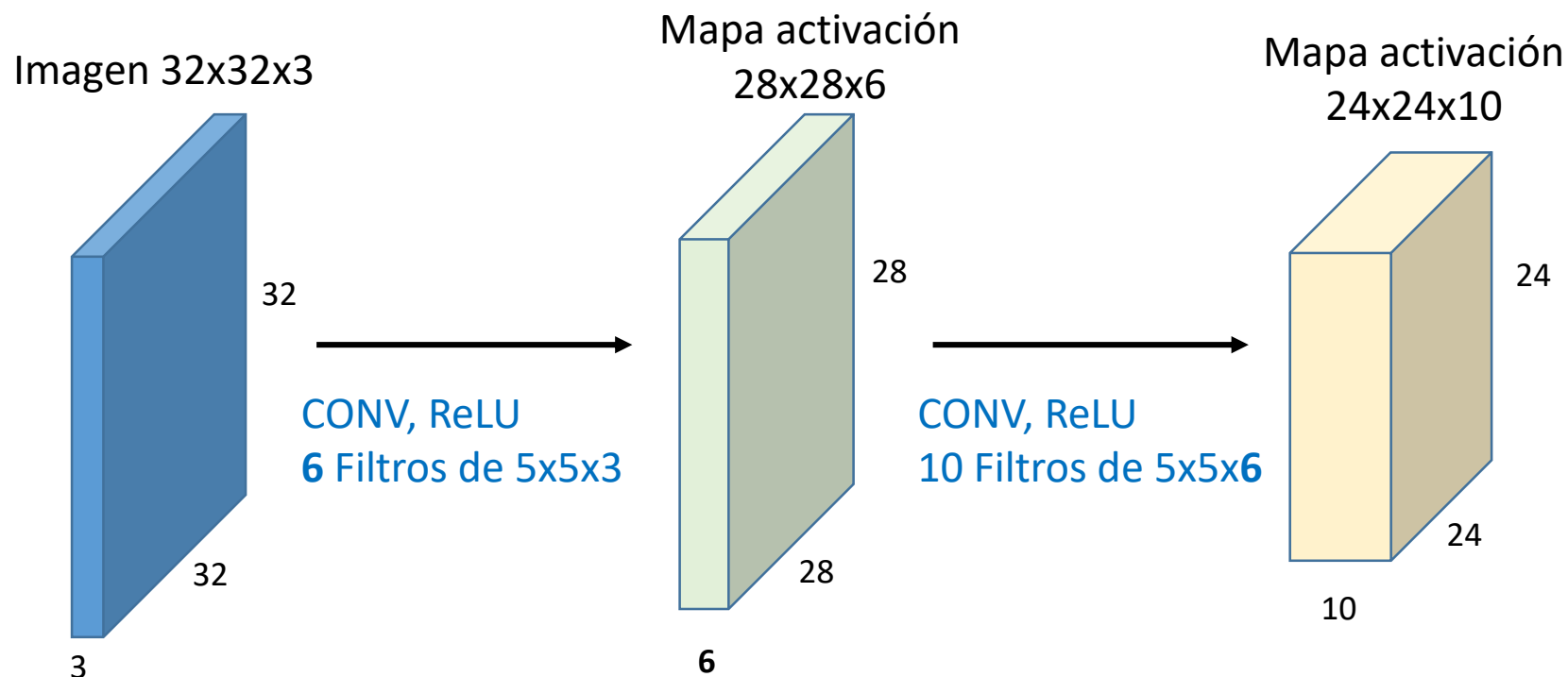
# Redes convolucionales

- Una ConvNet es una secuencia de capas convolucionales reguladas con función de activación.



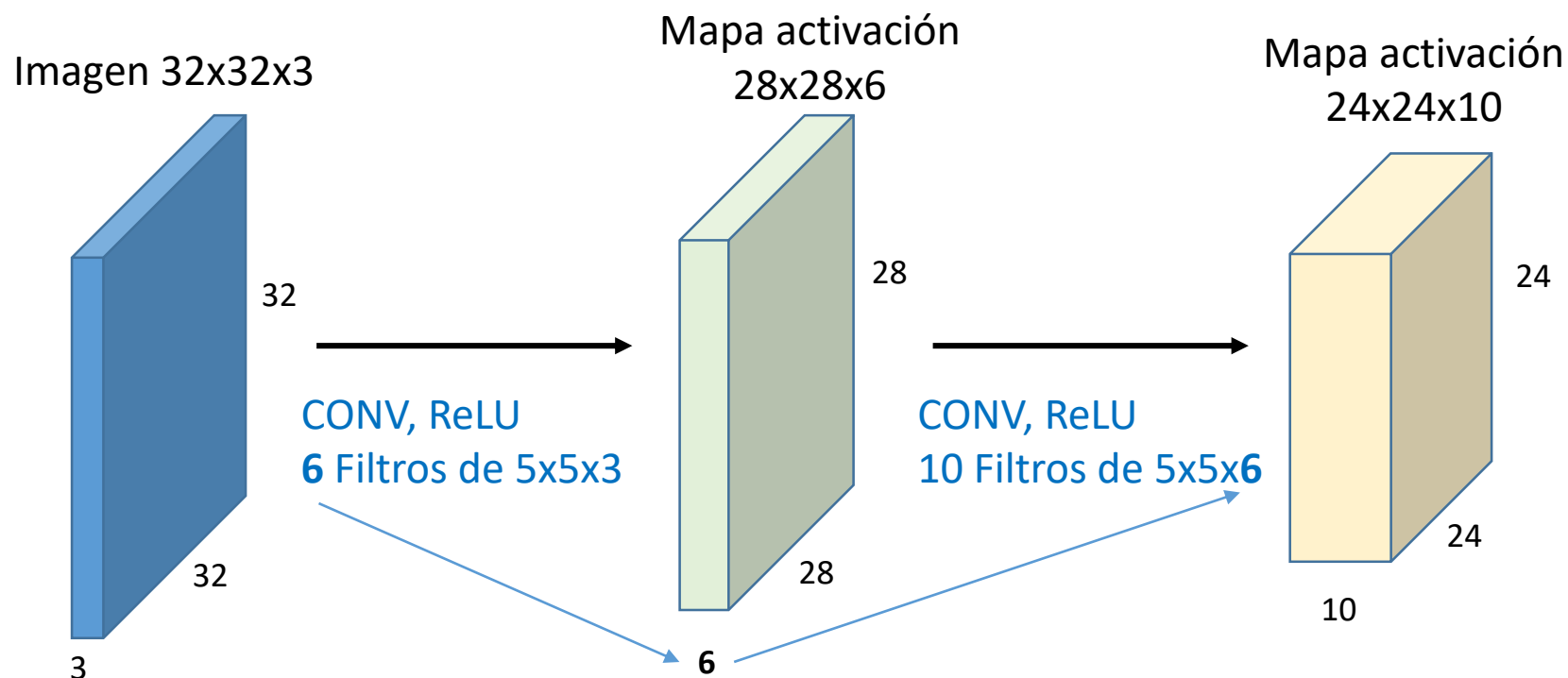
# Redes convolucionales

- Una ConvNet es una secuencia de capas convolucionales reguladas con función de activación.



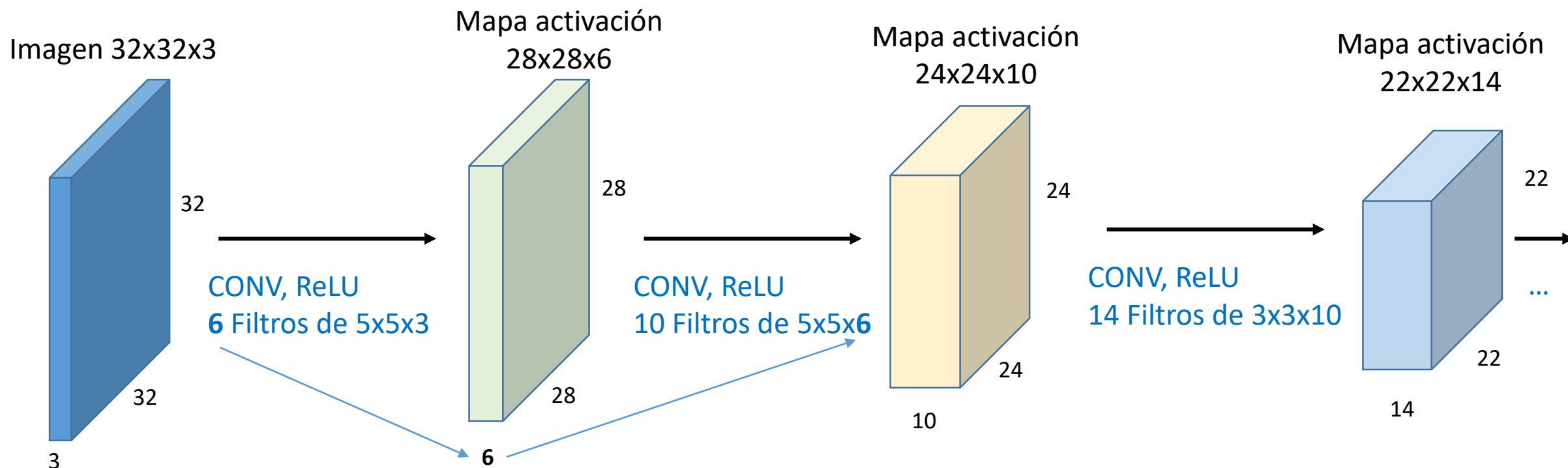
# Redes convolucionales

- Una ConvNet es una secuencia de capas convolucionales reguladas con función de activación.



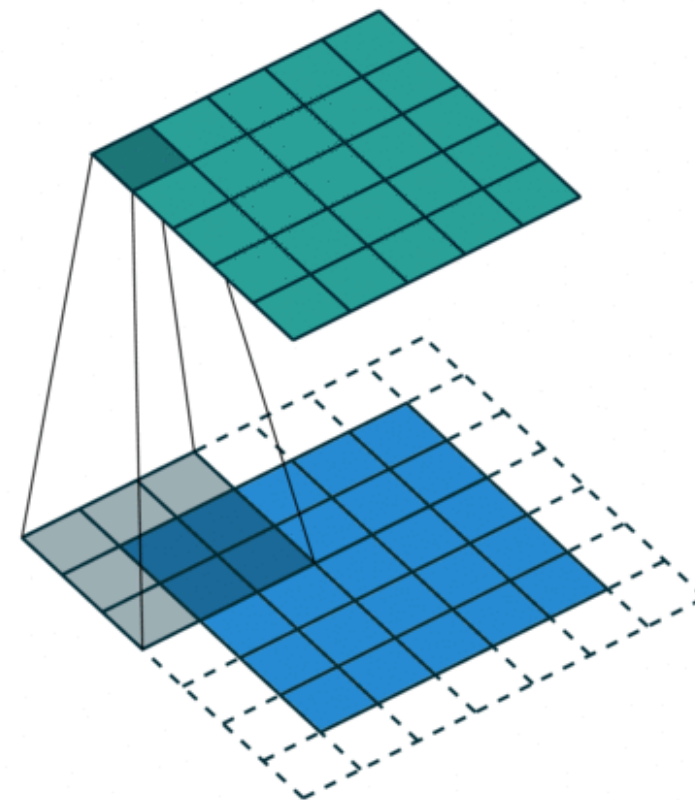
# Redes convolucionales

- Una ConvNet es una secuencia de capas convolucionales reguladas con función de activación.



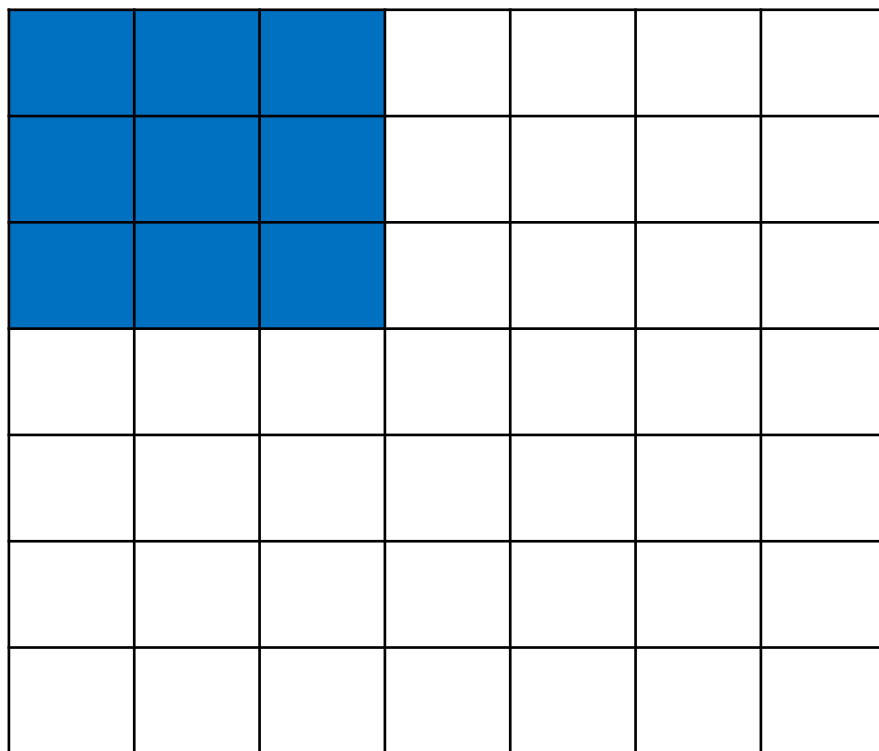
# Redes convolucionales

- Hiperparámetros para definir de capas convolucionales:
  - Tamaño de los filtros (kernel **size**):  $F \times F \times P$
  - Profundidad (**Depth**): número de filtros a aprender en una capa (número de neuronas en la capa).
  - Salto (**Stride**): suele ser 1 o 2. Si 2, salto de lo(s) filtro(s).
  - Relleno (**Padding**): relleno de bordes al aplicar convolución, para evitar pérdida de margen.



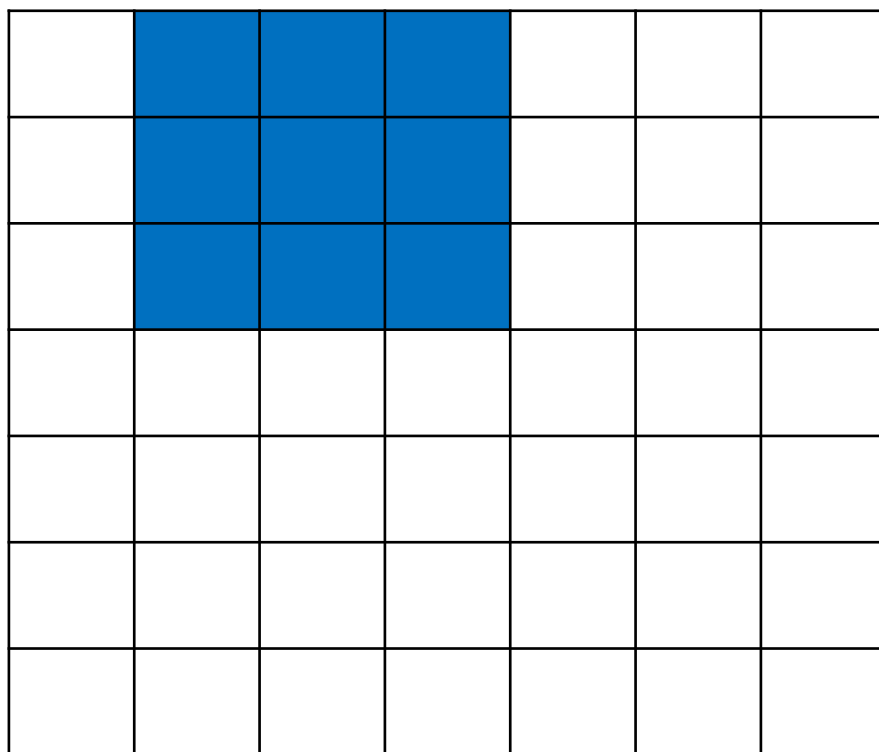
# Redes convolucionales

- Ejemplo sin padding (entrada 7x7, filtro 3x3, stride 1)



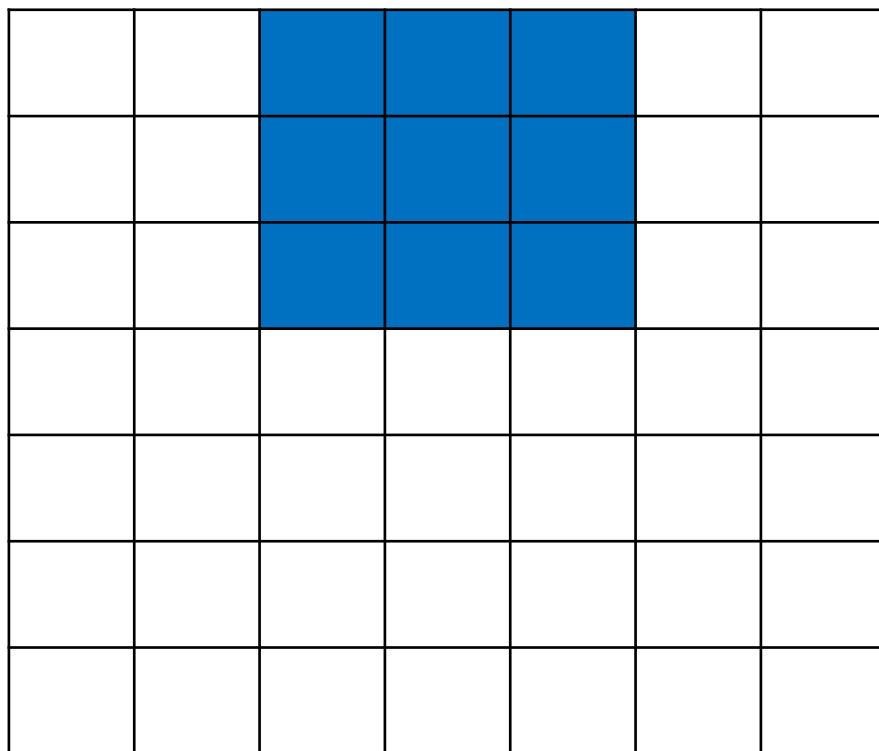
# Redes convolucionales

- Ejemplo sin padding (entrada 7x7, filtro 3x3, stride 1)



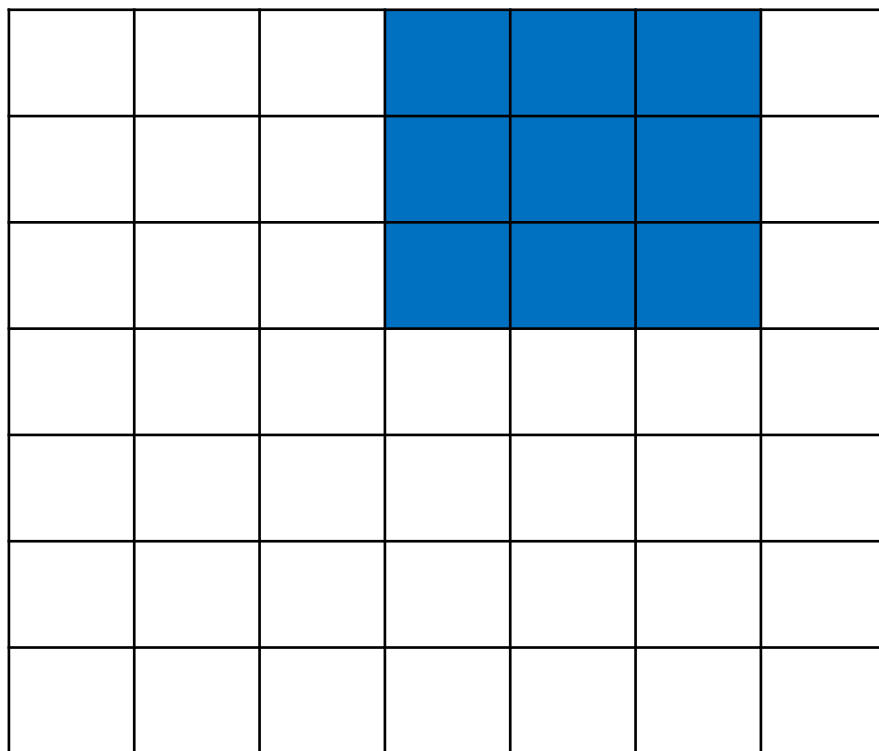
# Redes convolucionales

- Ejemplo sin padding (entrada 7x7, filtro 3x3, stride 1)



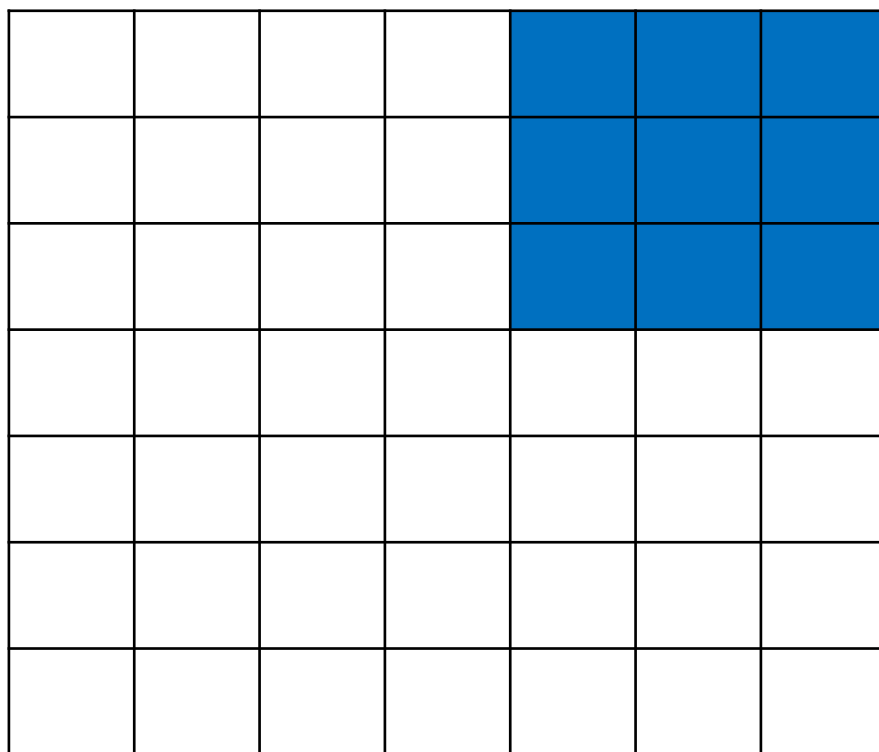
# Redes convolucionales

- Ejemplo sin padding (entrada 7x7, filtro 3x3, stride 1)



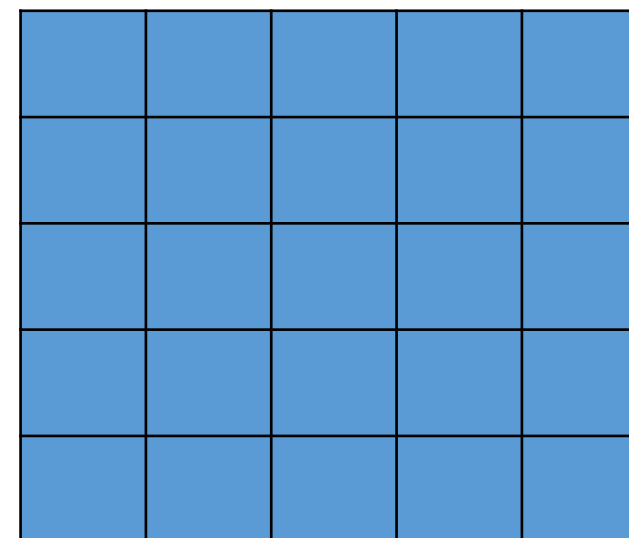
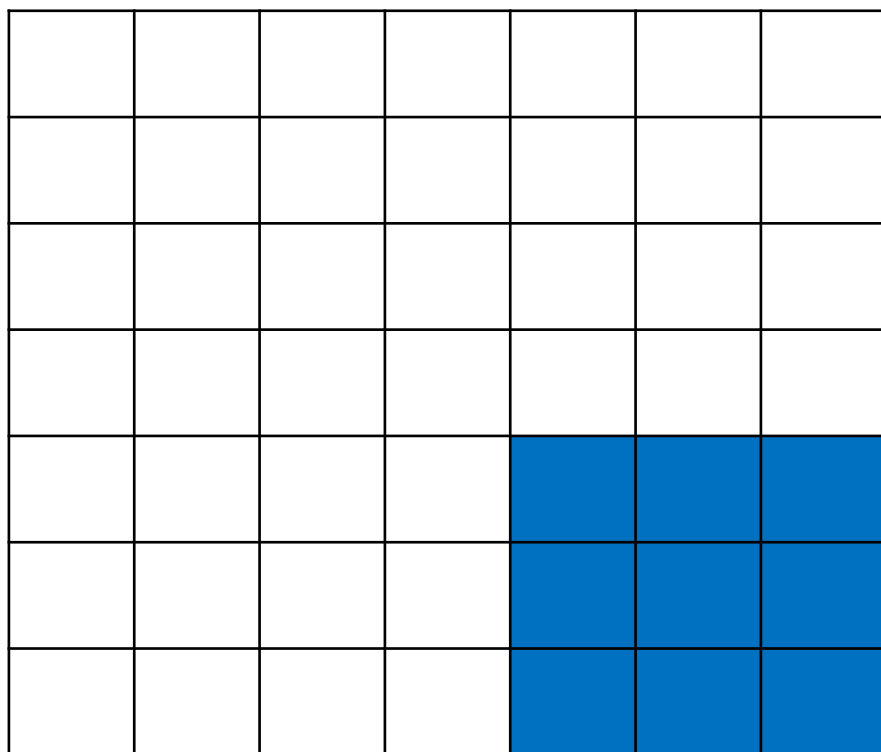
# Redes convolucionales

- Ejemplo sin padding (entrada 7x7, filtro 3x3, stride 1)



# Redes convolucionales

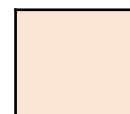
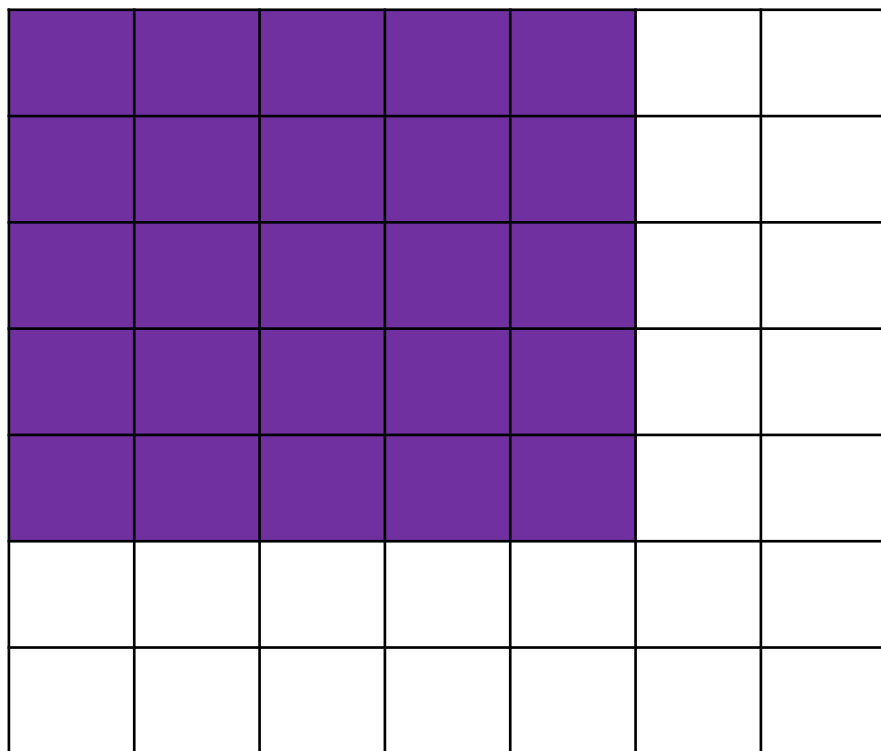
- Ejemplo sin padding (entrada 7x7, filtro 3x3, stride 1)



Feature map de 5x5

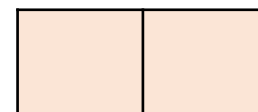
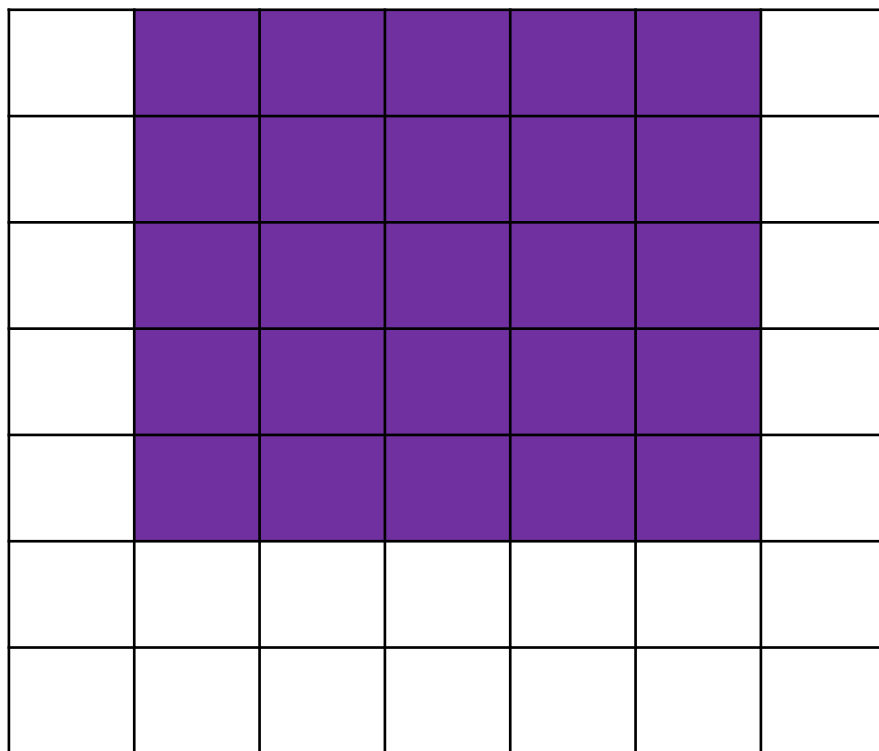
# Redes convolucionales

- Ejemplo sin padding (entrada 7x7, filtro 5x5, stride 1)



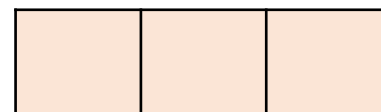
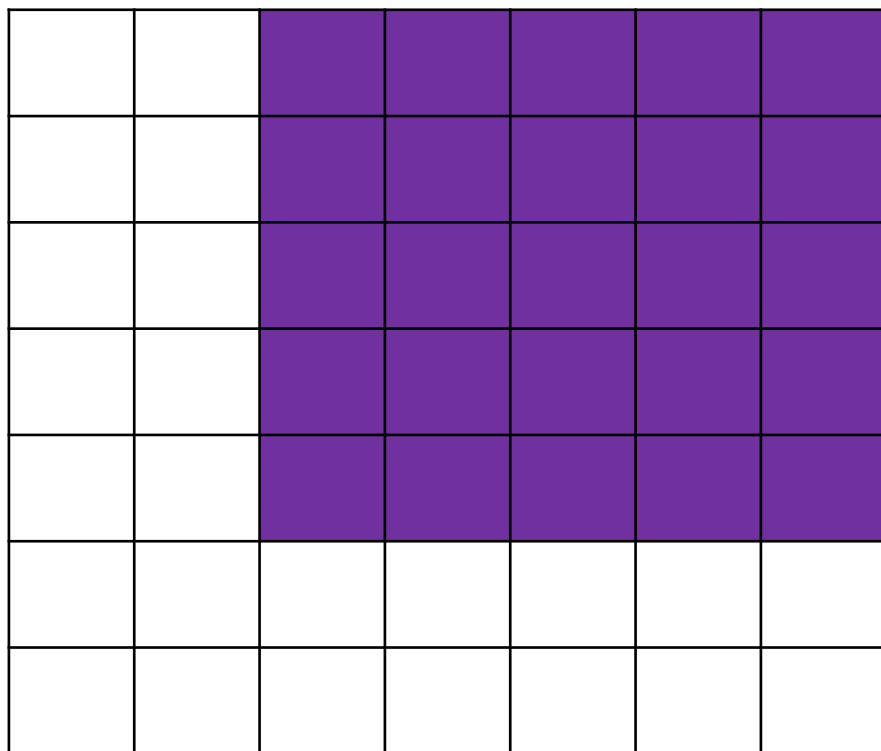
# Redes convolucionales

- Ejemplo sin padding (entrada 7x7, filtro 5x5, stride 1)



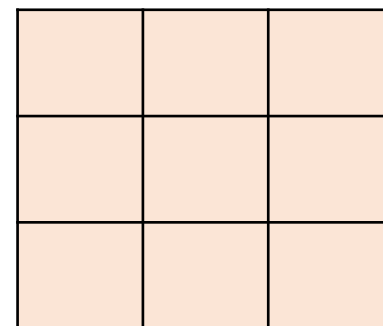
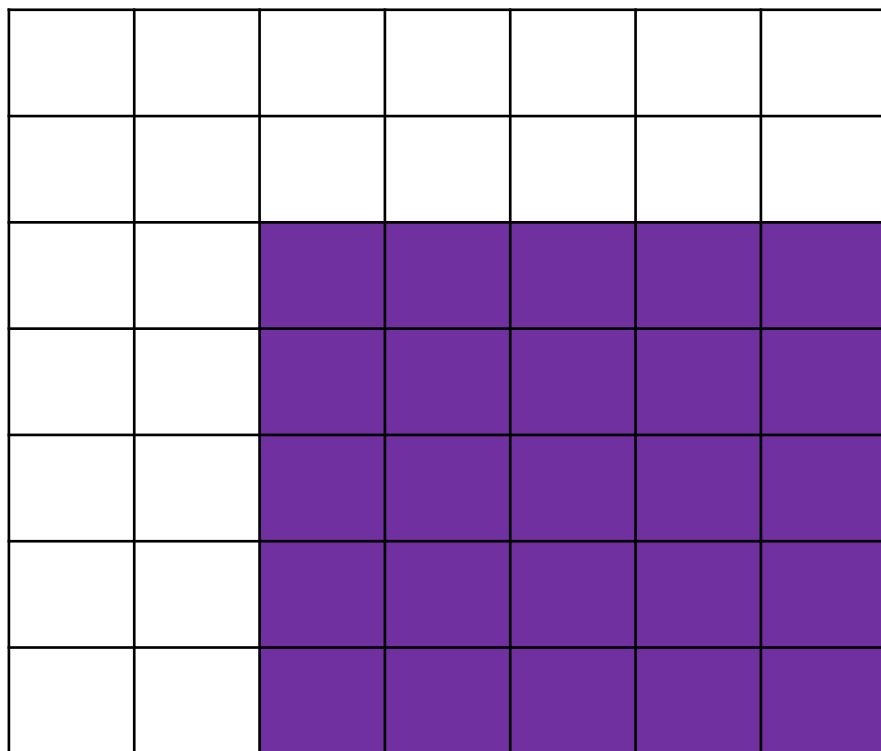
# Redes convolucionales

- Ejemplo sin padding (entrada 7x7, filtro 5x5, stride 1)



# Redes convolucionales

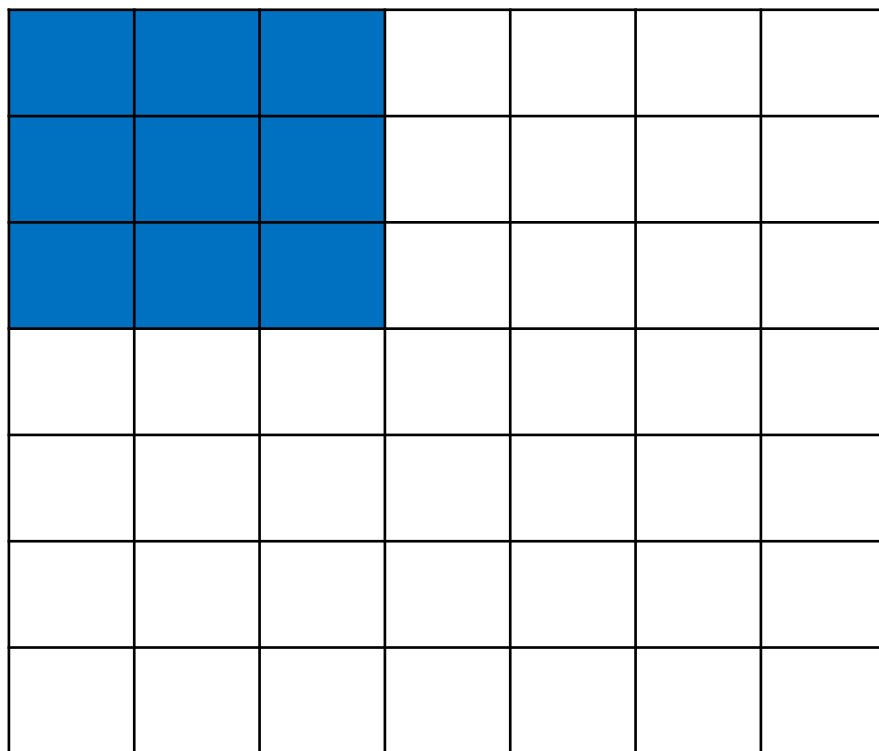
- Ejemplo sin padding (entrada 7x7, filtro 5x5, stride 1)



Feature map de 3x3

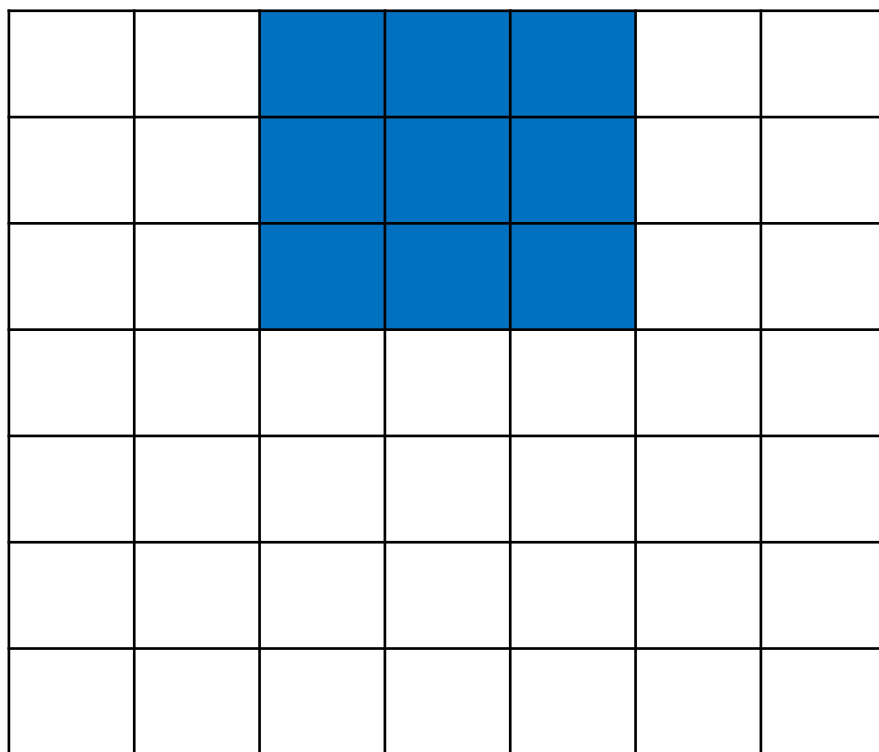
# Redes convolucionales

- Ejemplo sin padding (entrada 7x7, filtro 3x3, stride 2)



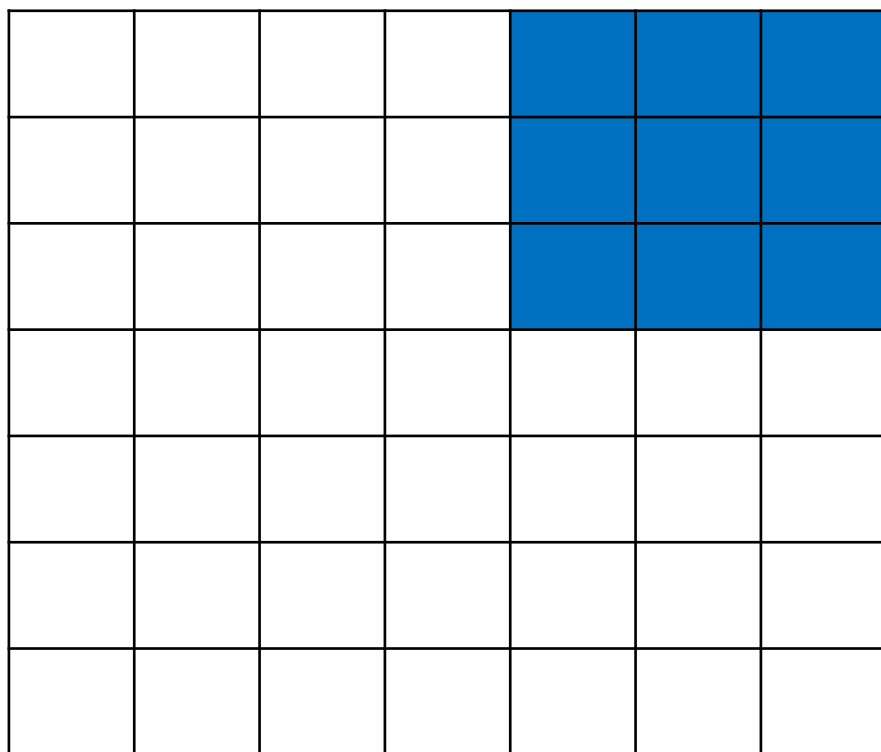
# Redes convolucionales

- Ejemplo sin padding (entrada 7x7, filtro 3x3, stride 2)



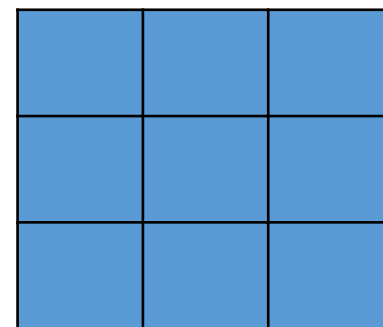
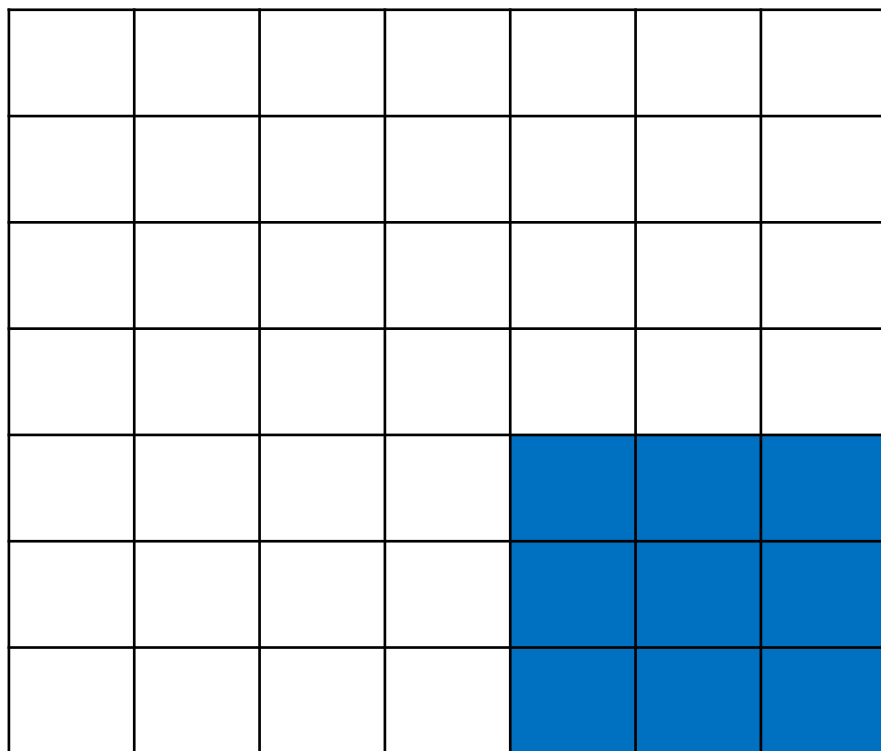
# Redes convolucionales

- Ejemplo sin padding (entrada 7x7, filtro 3x3, stride 2)



# Redes convolucionales

- Ejemplo sin padding (entrada 7x7, filtro 3x3, stride 2)



Feature map de 3x3

# Redes convolucionales

- Ejemplo con zero-padding (entrada 7x7, filtro 3x3, stride 1)

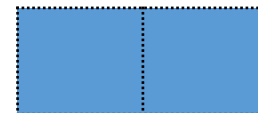
0	0	0	0	0	0	0	0	0
0								0
0								0
0								0
0								0
0								0
0								0
0	0	0	0	0	0	0	0	0



# Redes convolucionales

- Ejemplo con zero-padding (entrada 7x7, filtro 3x3, stride 1)

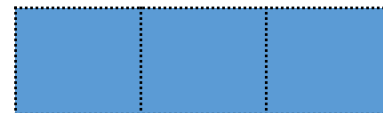
0	0	0	0	0	0	0	0	0
0								0
0								0
0								0
0								0
0								0
0								0
0	0	0	0	0	0	0	0	0



# Redes convolucionales

- Ejemplo con zero-padding (entrada 7x7, filtro 3x3, stride 1)

0	0	0	0	0	0	0	0	0
0								0
0								0
0								0
0								0
0								0
0								0
0	0	0	0	0	0	0	0	0



# Redes convolucionales

- Ejemplo con zero-padding (entrada 7x7, filtro 3x3, stride 1)

0	0	0	0	0	0	0	0	0
0								0
0								0
0								0
0								0
0								0
0								0
0	0	0	0	0	0	0	0	0



# Redes convolucionales

- Ejemplo con zero-padding (entrada 7x7, filtro 3x3, stride 1)

0	0	0	0	0	0	0	0	0
0								0
0								0
0								0
0								0
0								0
0								0
0	0	0	0	0	0	0	0	0



# Redes convolucionales

- Ejemplo con zero-padding (entrada 7x7, filtro 3x3, stride 1)

0	0	0	0	0	0	0	0	0
0								0
0								0
0								0
0								0
0								0
0								0
0	0	0	0	0	0	0	0	0



# Redes convolucionales

- Ejemplo con zero-padding (entrada 7x7, filtro 3x3, stride 1)

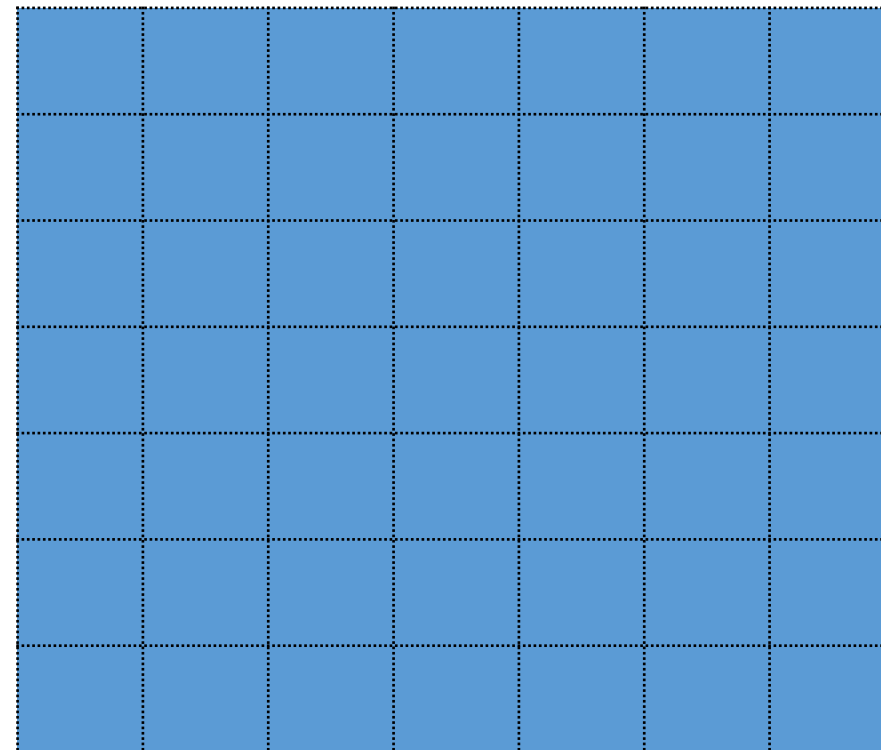
0	0	0	0	0	0	0	0	0
0								0
0								0
0								0
0								0
0								0
0								0
0	0	0	0	0	0	0	0	0



# Redes convolucionales

- Ejemplo con zero-padding (entrada 7x7, filtro 3x3, stride 1)

0	0	0	0	0	0	0	0	0
0								0
0								0
0								0
0								0
0								0
0								0
0								0
0	0	0	0	0	0	0	0	0



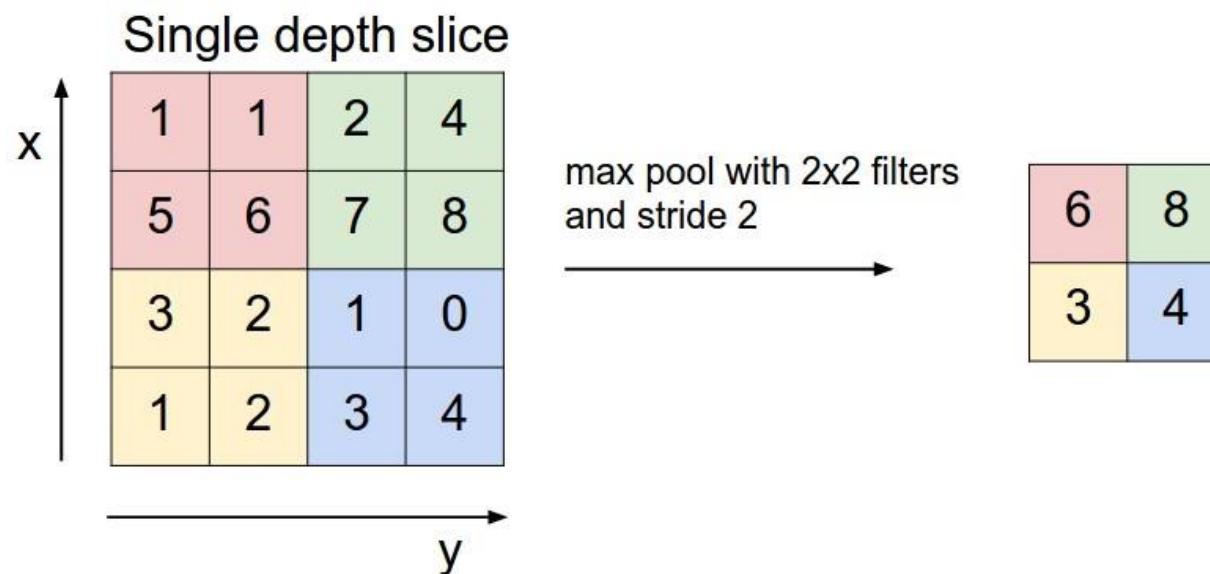
Feature map de 7x7

# Redes convolucionales

- Tamaño entrada ( $N \times N$ ), tamaño filtro ( $F \times F$ ), stride ( $S$ ), profundidad capa ( $P$ ), profundidad entrada ( $D$ )
  - **Tamaño de salida:**  $M \times M$ , donde  $M = (N - F) / S + 1$
  - **Número de parámetros:**  $(F * F * D + 1) * P$
- **Convolución  $1 \times 1$** , usado para
  - Adaptar profundidades, mezclándolas.
  - Reemplazar capas FC, permitiendo introducir imágenes más grandes y ganar en eficiencia.
- Convolución es [derivable](#) para usarse backpropagation
  - La derivada es otra convolución de los errores sobre la entrada.
- Ver [ejemplo](#).

# Pooling (subsampling/downsampling)

- Capa que **redimensiona** espacialmente la representación
- Es común insertarla periódicamente entre capas convolutivas
- Se aplica a cada feature map en la profundidad
- **Operaciones** típicas: MAX, AVG, SUM, L2, ...
- Es **derivable** para propagación de gradientes.

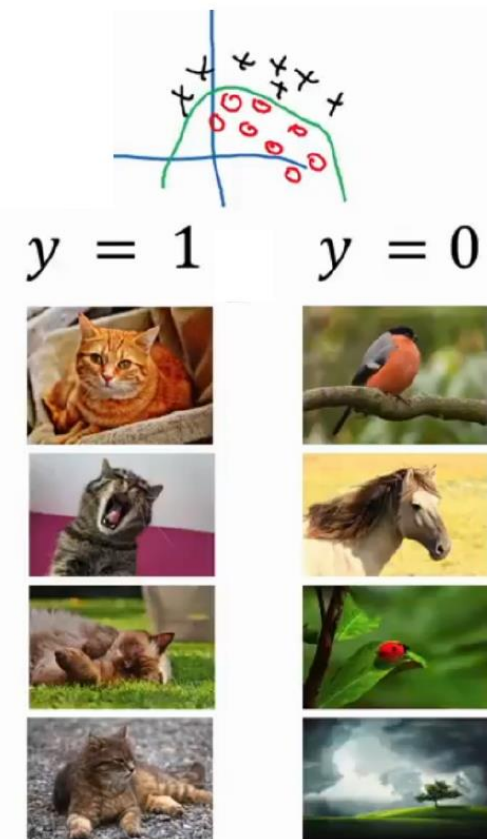
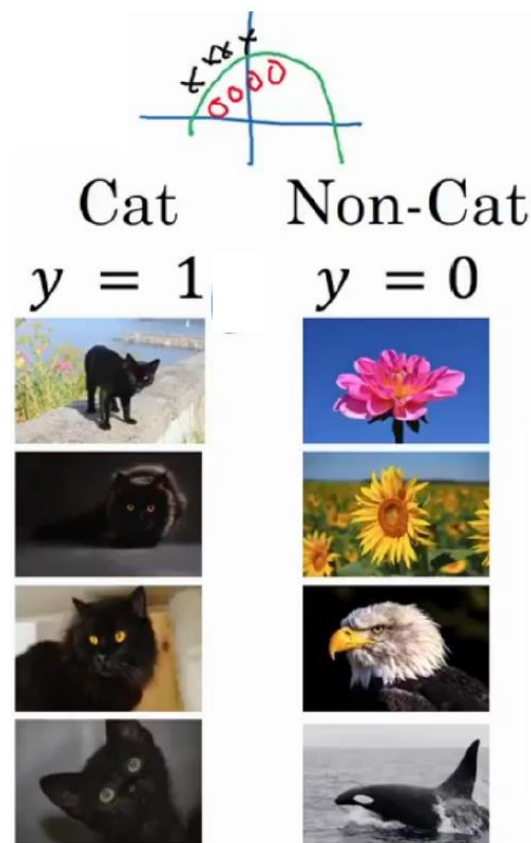


# Pooling (subsampling/downsampling)

- **Ayuda a:**
  - **Reducir** el número de parámetros: evitando el sobreajuste
  - Representación **invariante** a pequeñas traslaciones: detectar características sin importar ubicación
- **No** se puede **abusar** de ella:
  - Algunas propiedades requieren preservar la posición.
  - Algunas redes no intercalan siempre una de pooling después de convolución.
- Hiperparámetros:
  - Extensión espacial del nivel (F), stride (S).
  - Valores típicos  $F=2$ ,  $S=2$ , o  $F=3$ ,  $S=2$ .

# Batch normalization

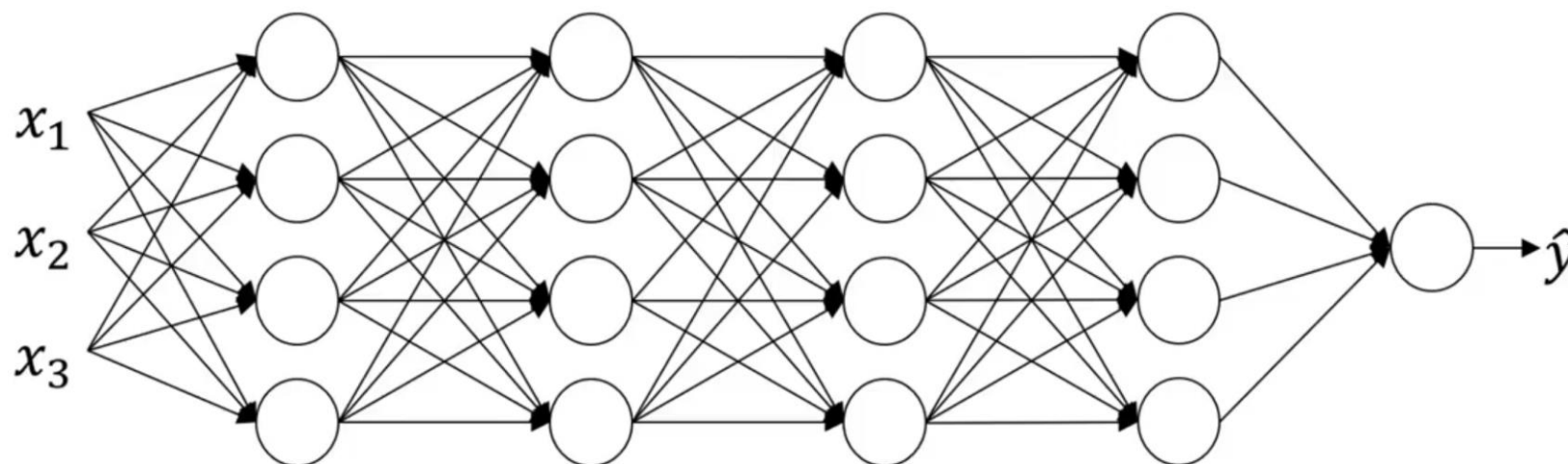
- Hemos visto la importancia de normalizar los datos:
  - La actualización de los pesos depende de las entradas!
- **Covariate shift problem:**  
Cambio de distribución de datos puede hacer que nuestro modelo no generalice bien.



# Batch normalization

- Motivación

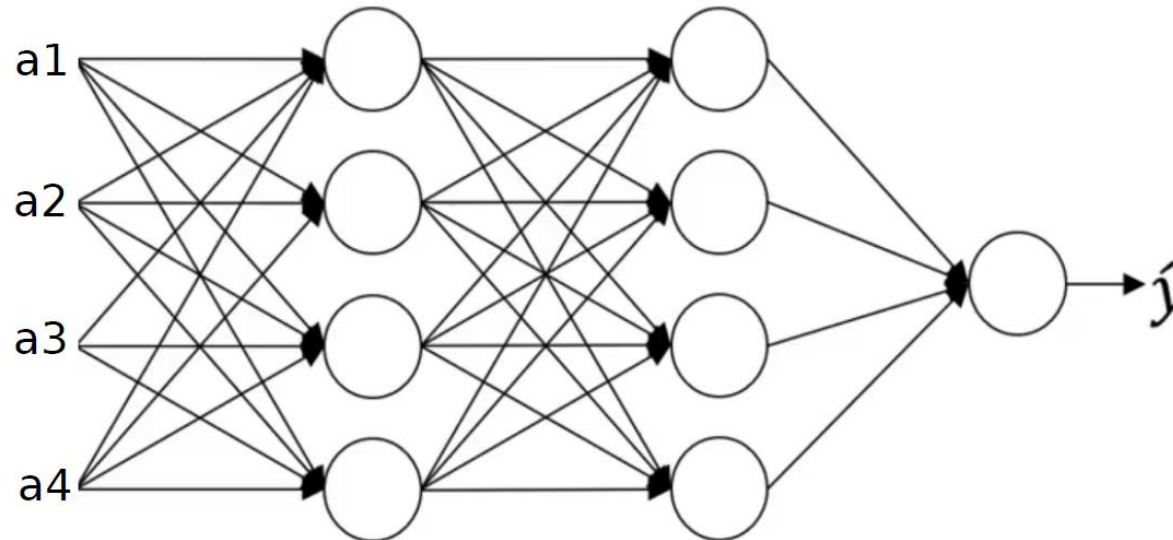
- Supongamos una red perceptrón multicapa,
- Centrémonos en la tercera capa.



# Batch normalization

- **Motivación**

- La salida de la segunda capa (entrada de la tercera) cambiará constantemente durante el entrenamiento.
- **Idea:** reducir la cantidad de cambios normalizando los valores.



# Batch normalization

- Introduce nuevos parámetros a aprender:
  - $\gamma$  factor de escalado
  - $\beta$  factor de desplazamiento
  - Pueden determinar si es necesario normalizar o no.
- Se suele aplicar antes de función de activación:
  - Evitar problema *vanishing/exploding gradient*.

[[Ioffe y Szegedy 2015](#)]

**Input:** Values of  $x$  over a mini-batch:  $\mathcal{B} = \{x_{1\dots m}\}$ ;

Parameters to be learned:  $\gamma, \beta$

**Output:**  $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{ mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{ mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{ normalize}$$

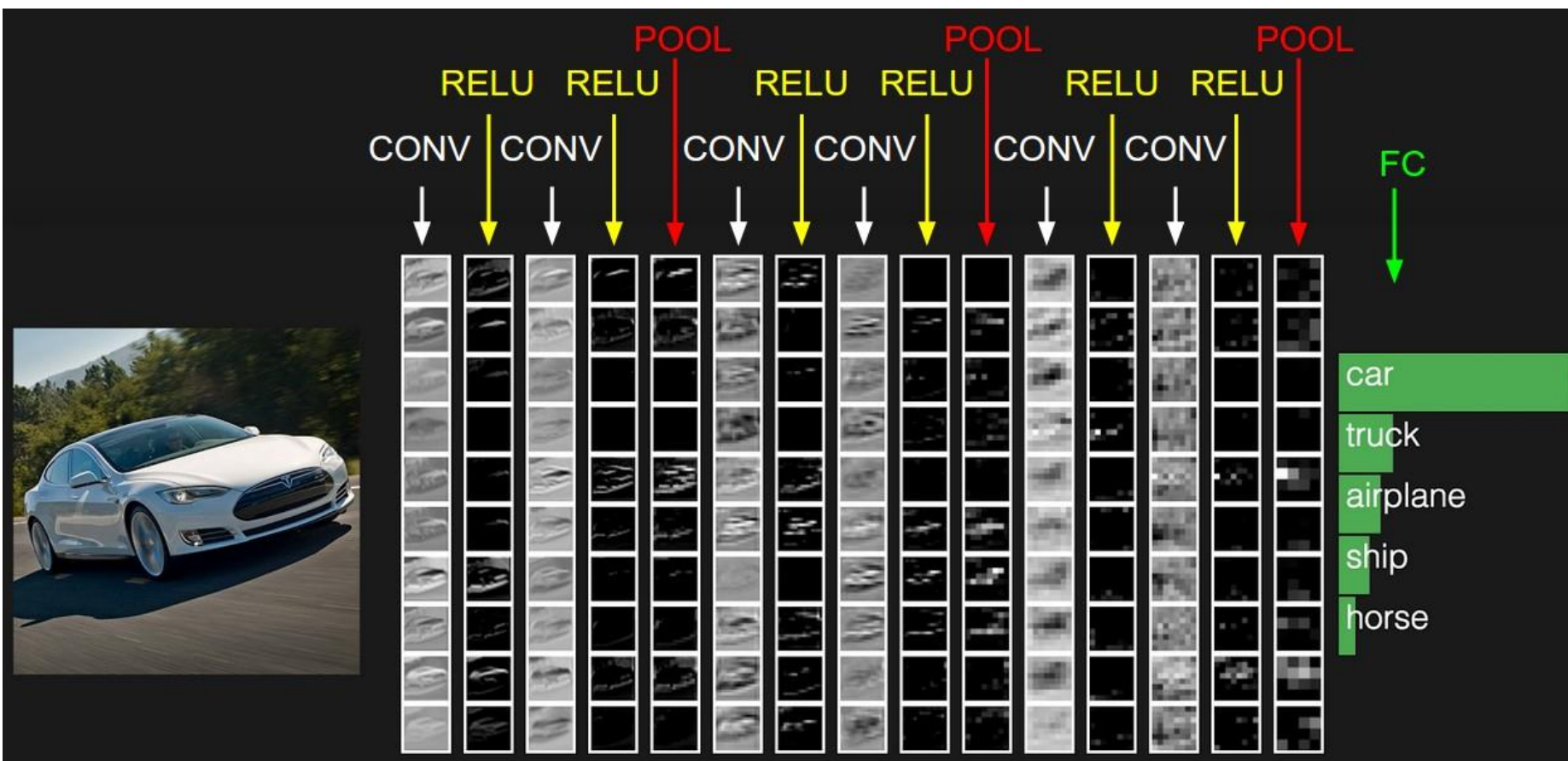
$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{ scale and shift}$$

**Algorithm 1:** Batch Normalizing Transform, applied to activation  $x$  over a mini-batch.

# Batch normalization

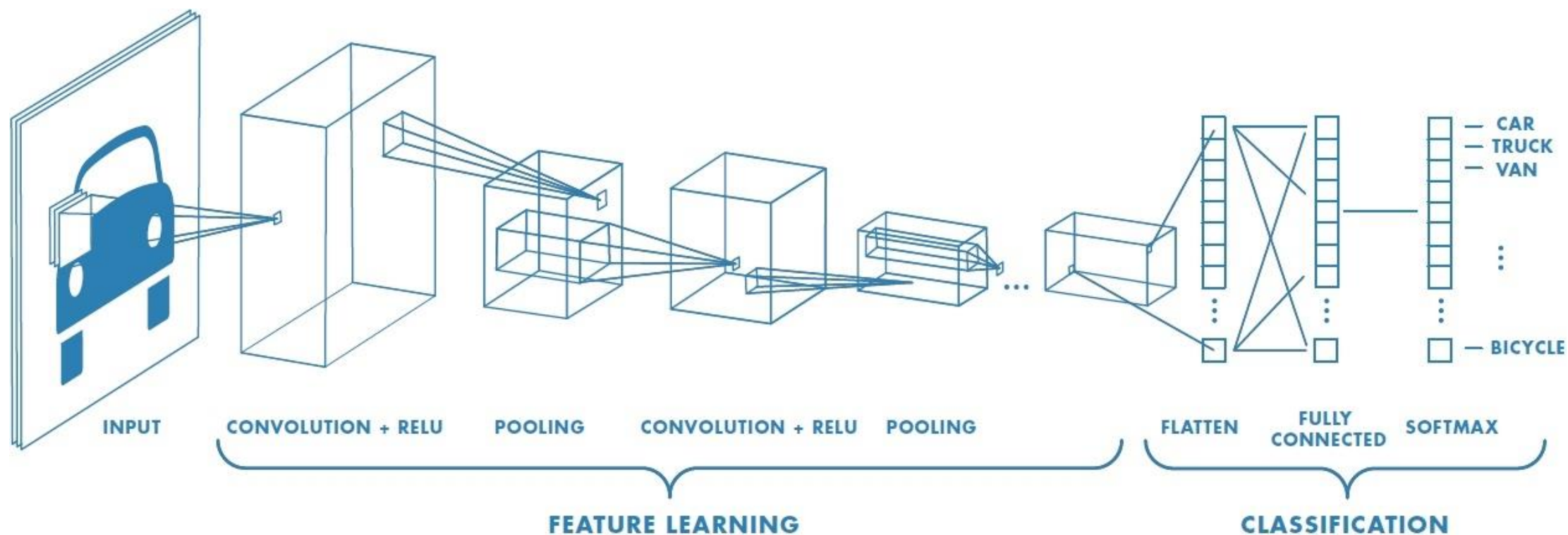
- Ventajas:
  - Las redes son menos sensibles a los valores iniciales.
  - Se pueden usar un learning rate mayor.
  - Reduce el tiempo de entrenamiento, convergencia más rápida.
  - Funciona como un regularizador, reduciendo la necesidad de usar otras técnicas de regularización.
- Desventaja: la predicciones son mas lentas.

# Interpretando redes convolucionales



# Interpretando redes convolucionales

- Esquema de una red convolucional



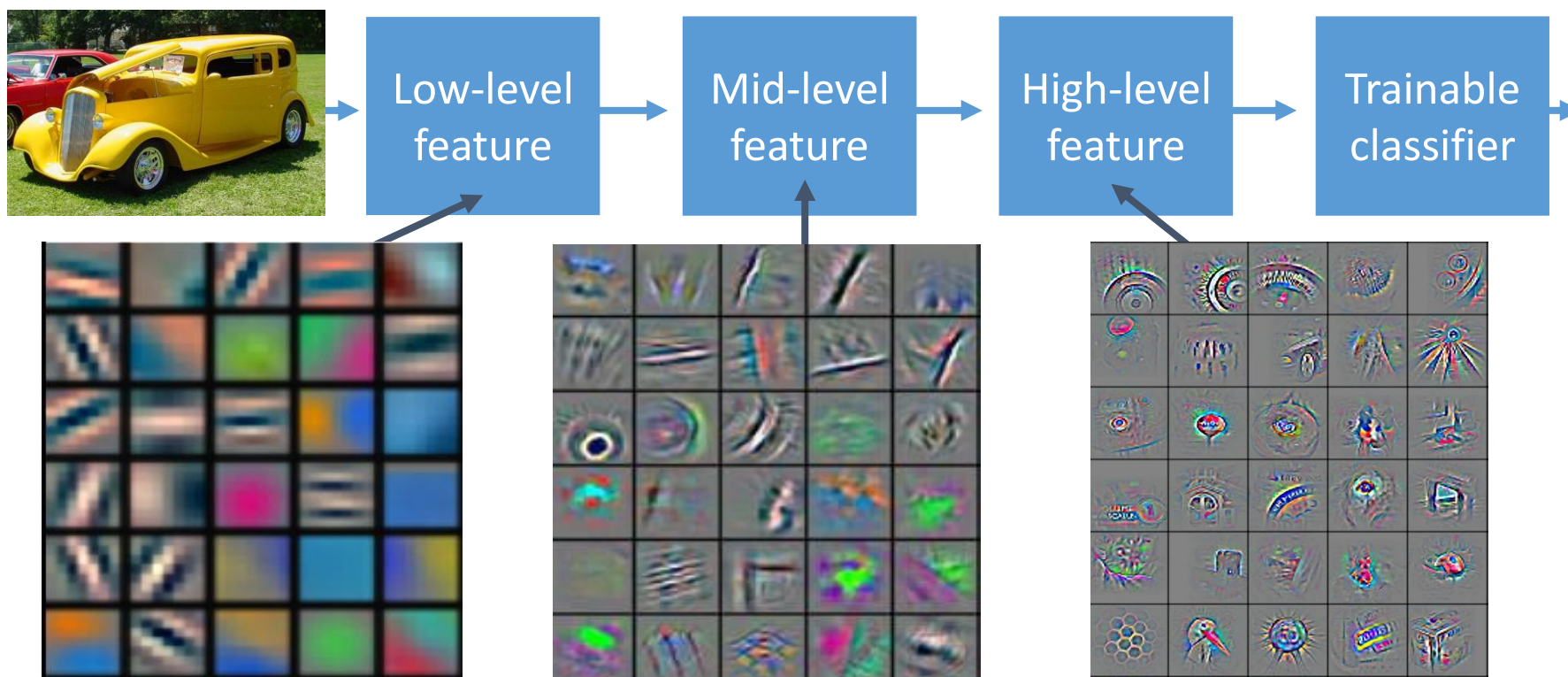
# Interpretando redes convolucionales

- Ver demo:

<https://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html>

# Interpretando redes convolucionales

- Visualización de los filtros aprendidos por una red ([ver aquí](#)).



# Recapitulando

- La operación de **convolución** permite representaciones invariantes.
  - Requiere definir una serie de **filtros** por capa, que se aplicarán a lo largo de la imagen o capa anterior, dando como resultado un mapa de activación (**feature map**).
- **Pooling** permite reducir los parámetros y la representación.
- **Batch normalization** reduce el efecto del covariate shift.