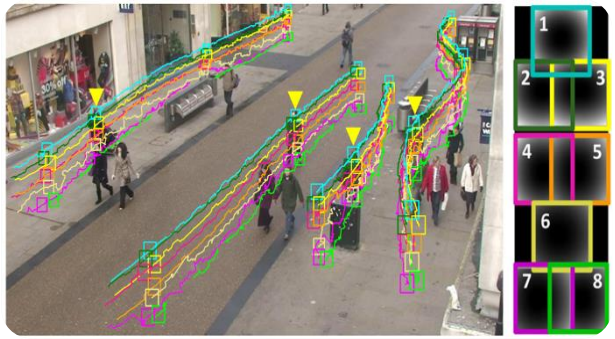




DEEP LEARNING

Aprendizaje Maquinal

Segundo Cuatrimestre - 2022 -

MOTIVACIÓN

- Algoritmos de ML tradicionales utilizan arquitecturas poco profundas (ANN con una sola capa oculta, máquinas de vectores soporte (SVM), regresiones con kernel logístico, etc.)

- Representaciones internas aprendidas simples e incapaces de extraer algunos tipos de estructuras complejas a partir de entradas de alta dimensionalidad

- Ser humano muestra procesamiento sensorial usando múltiples capas: procesamiento de datos en redes neuronales profundas

MOTIVACIÓN

- Hallazgos neurobiológicos sugieren que:
  - El cerebro no pre-procesa la información sensorial
  - Permite su propagación a través de una jerarquía de etapas
  - Aprenden a representar las observaciones basadas en las regularidades exhibidas

MOTIVACIÓN

- Primeros intentos de replicar artificialmente arquitecturas neuronales profundas poco exitosos

- Algoritmos de aprendizaje automático, bajo la ***"maldición de la dimensionalidad"***

- Estrategia usada para superar este inconveniente: pre-procesar los datos para reducir su dimensionalidad → pérdida de información

APRENDIZAJE PROFUNDO

- Última década: se proponen modelos profundos (jerárquicos) capaces de extraer representaciones útiles y estructuradas de alto nivel
- Pueden extraer dependencias estadísticas complejas a partir de entradas sensoriales de alta dimensión y aprender representaciones eficientes de alto nivel reutilizando y combinando conceptos intermedios
- Muestran buena generalización en una amplia variedad de tareas: clasificación de objetos, reconocimiento del habla, predicción de secuencias, etc.

DEEP LEARNING

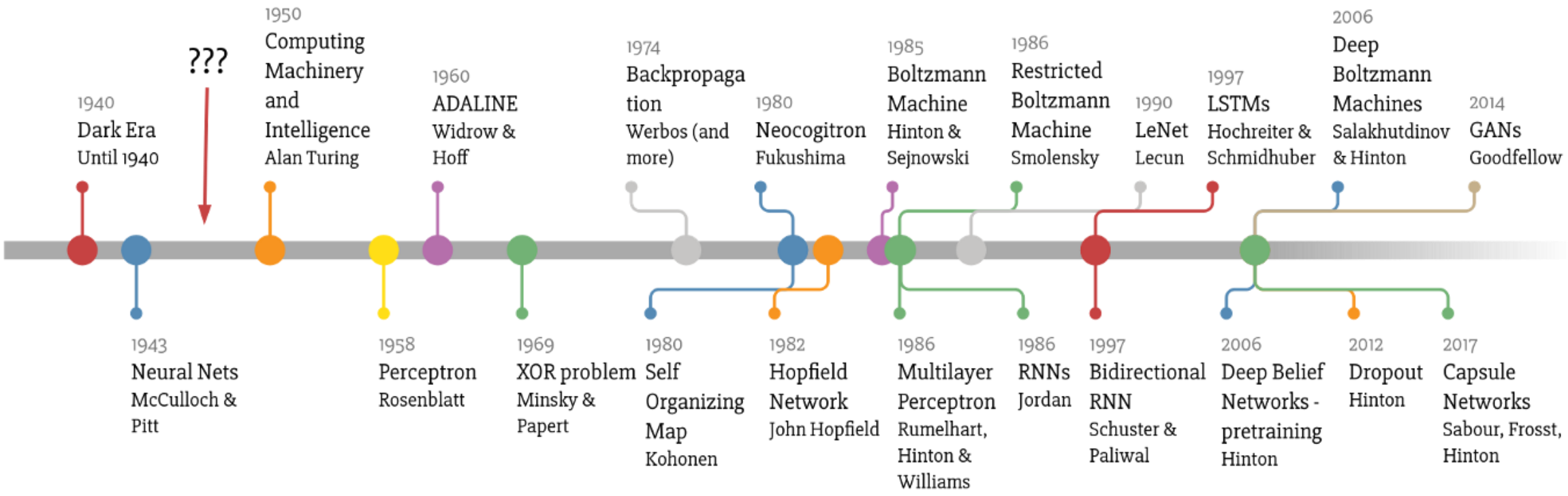
- Por qué ahora

Hardware	Datos	Software
• GPUs • TPUs • Posibilidad de paralelización masiva	• BD masivos (Imagenet) • Facilidad de captura de nuevos datos	• Mejores Modelos • Toolboxes (Tensorflow, PyTorch)

APRENDIZAJE PROFUNDO

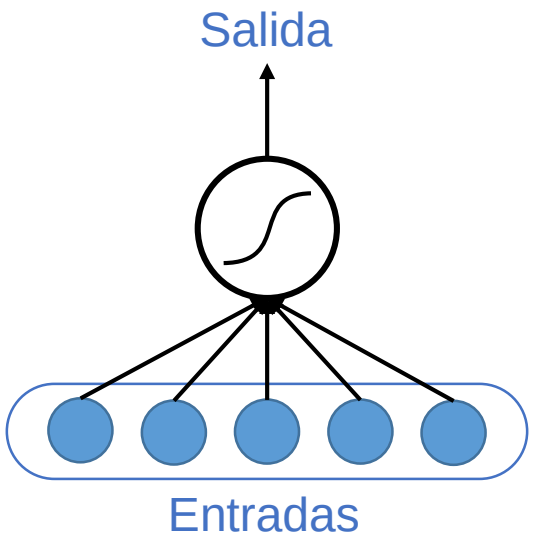
- Modelos computacionales para la representación de información inspirados en la neocorteza
- Empleo de abstracciones jerárquicas y construcción gradual de representaciones en niveles de abstracción incrementales
- Buscan capturar dependencia espacio-temporales en base a regularidades en las observaciones
- La **profundidad** de las arquitecturas se refiere al **número de niveles de operaciones no lineales** que componen la función aprendida

LÍNEA DE TIEMPO

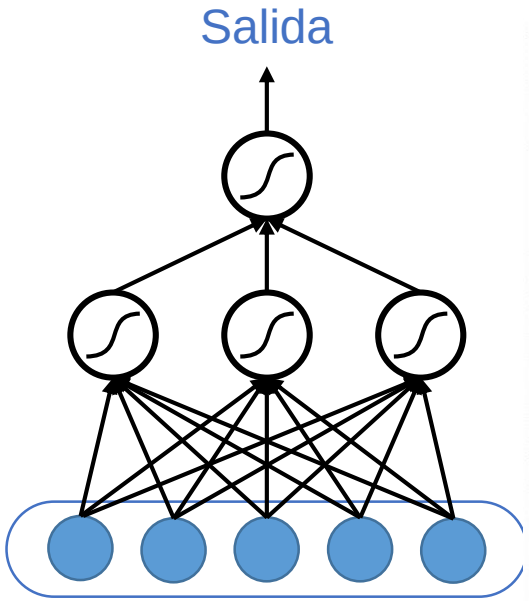


DEEP LEARNING

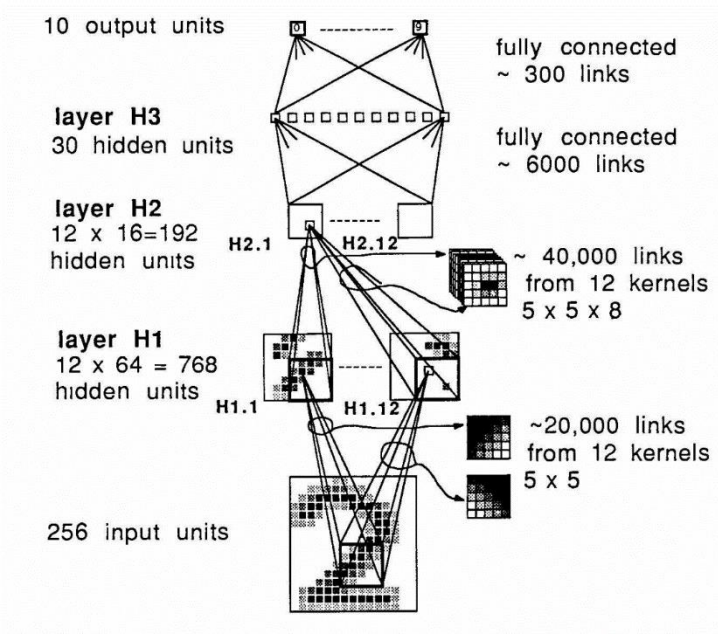
MUCHAS CAPAS OCULTAS…



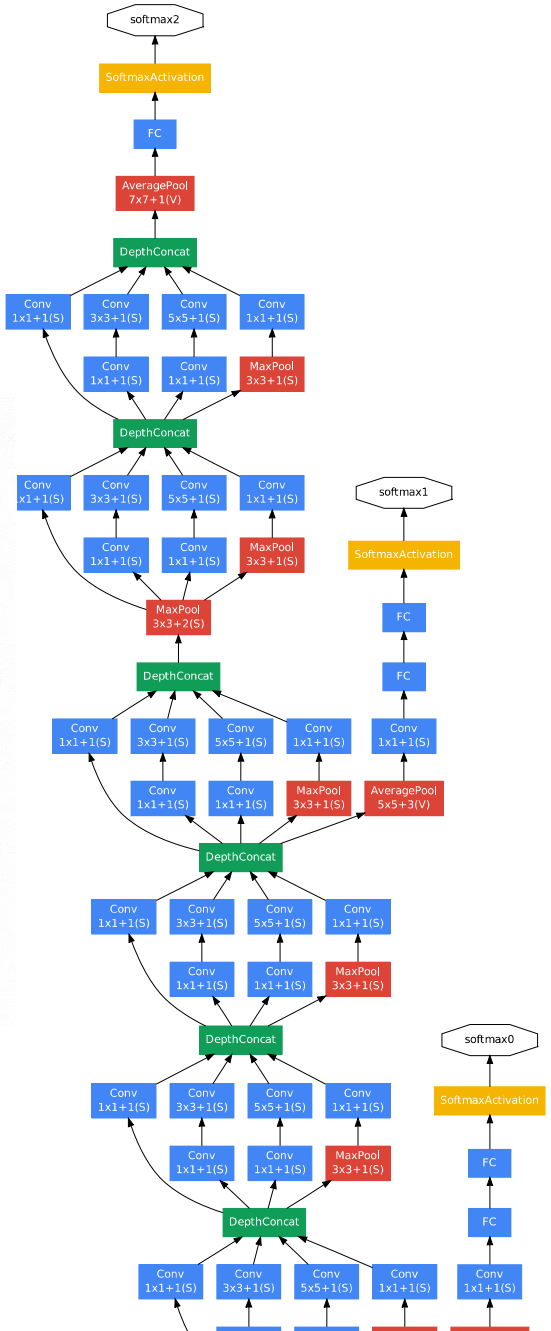
Regresión Logística



Perceptrón Multicapa



CNN LeNet



EVOLUCIÓN

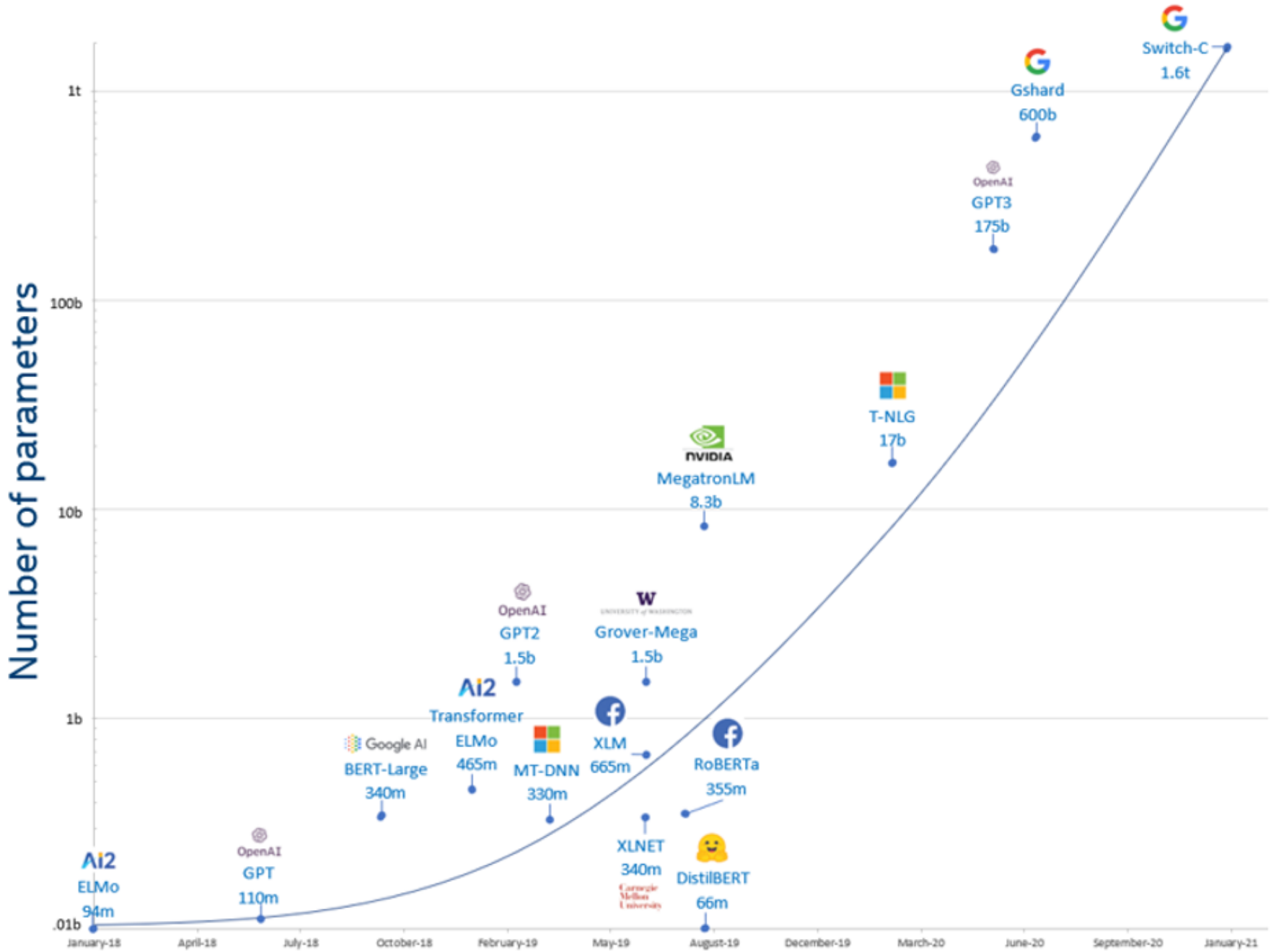
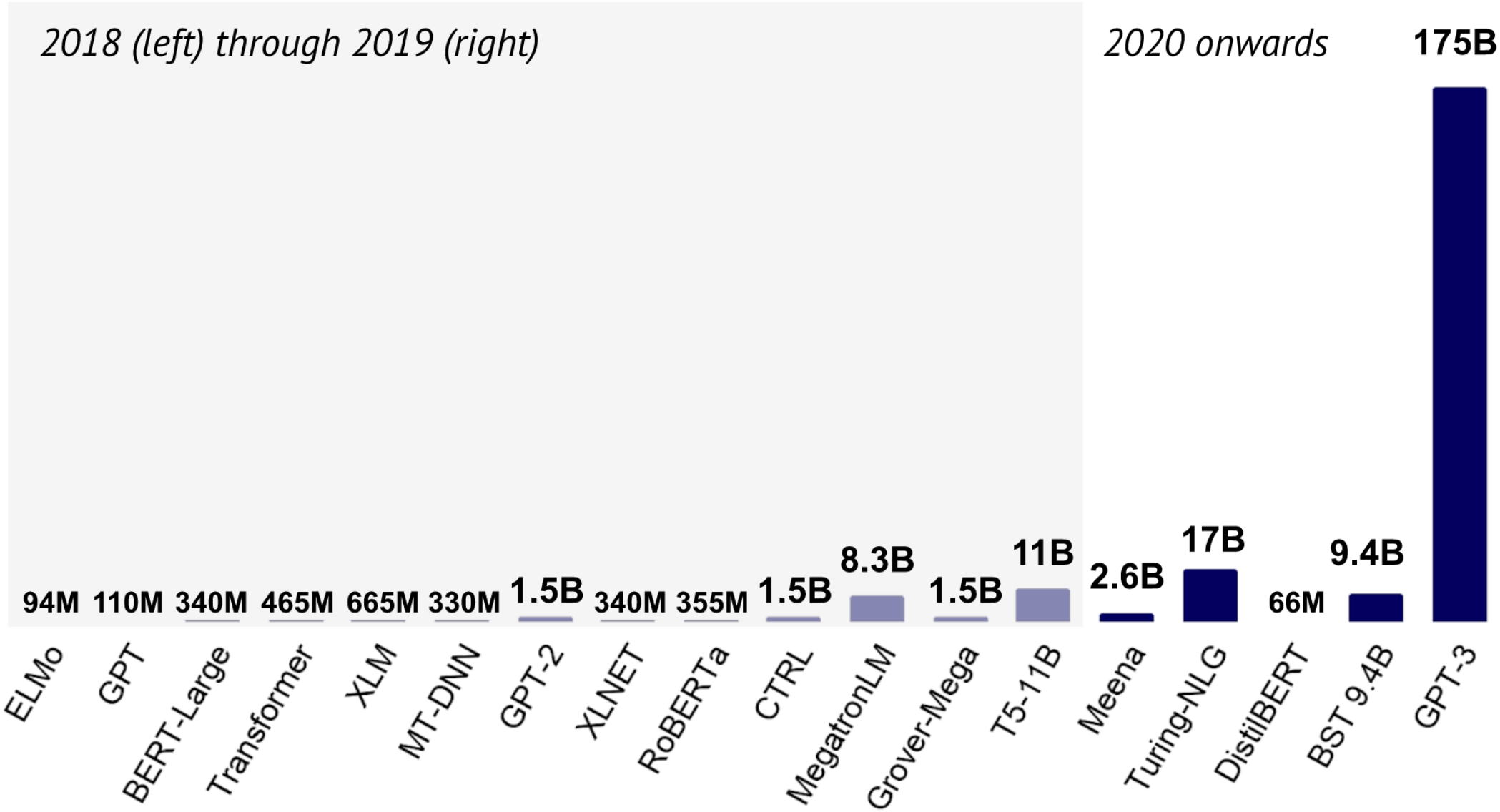
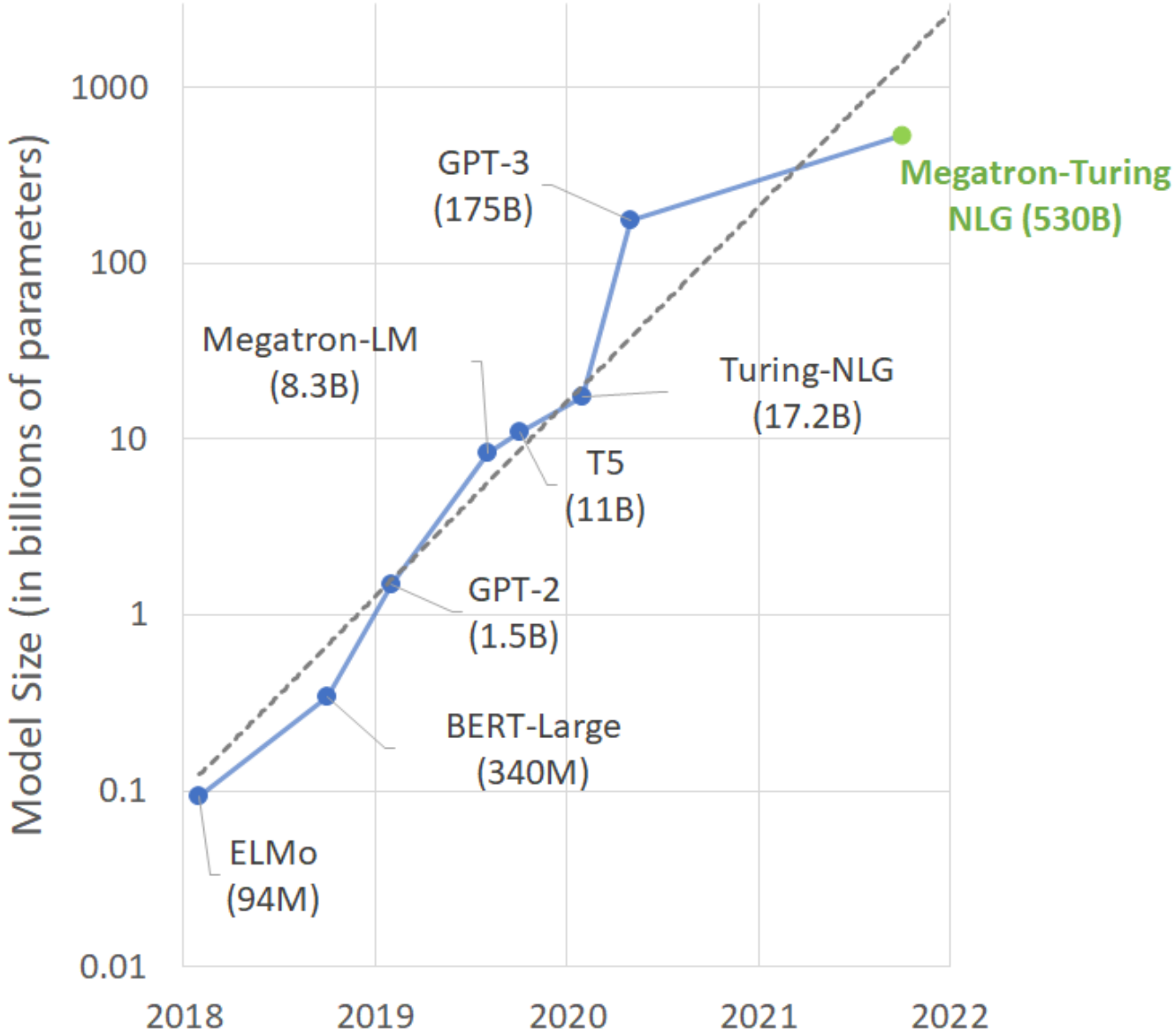


Figure 1: Exponential growth of number of parameters in DL models

EVOLUCIÓN DE MODELOS

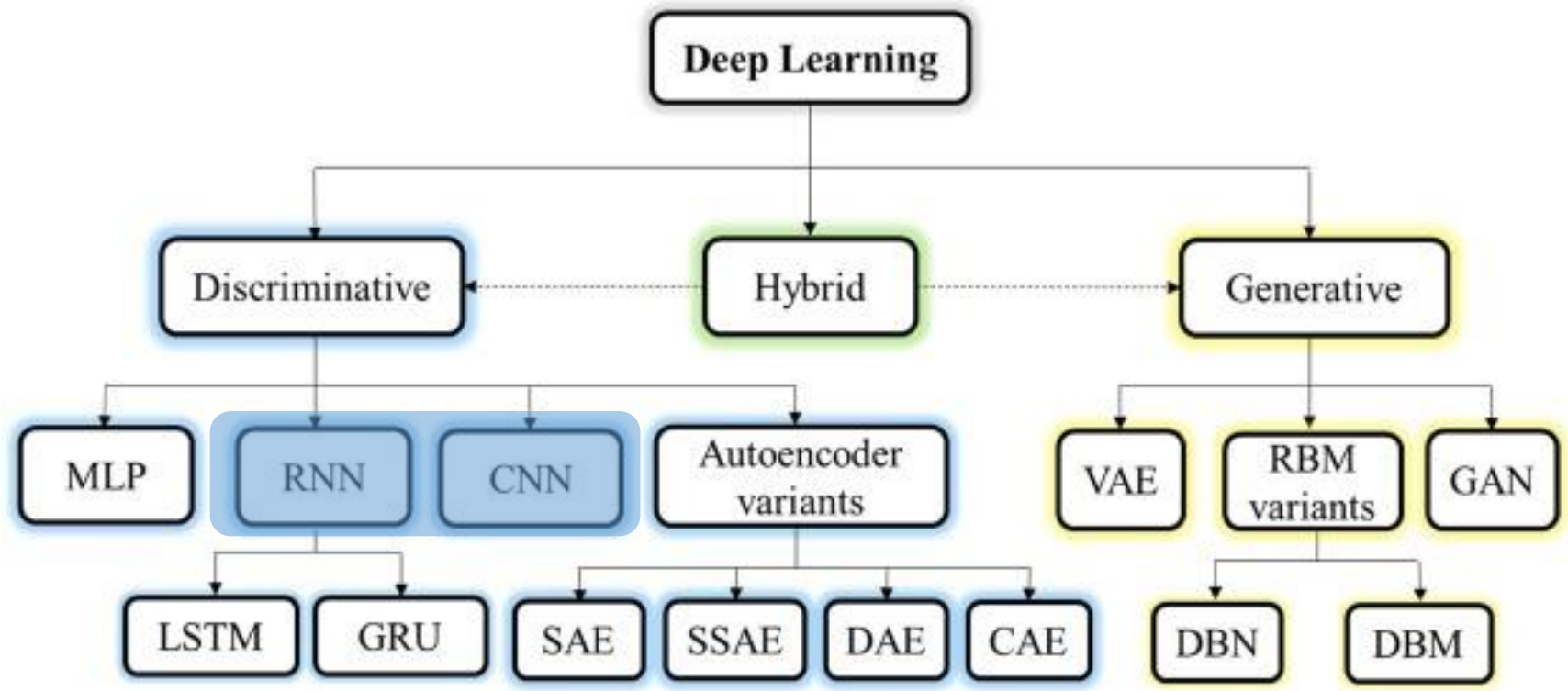


EVOLUCIÓN DE MODELOS



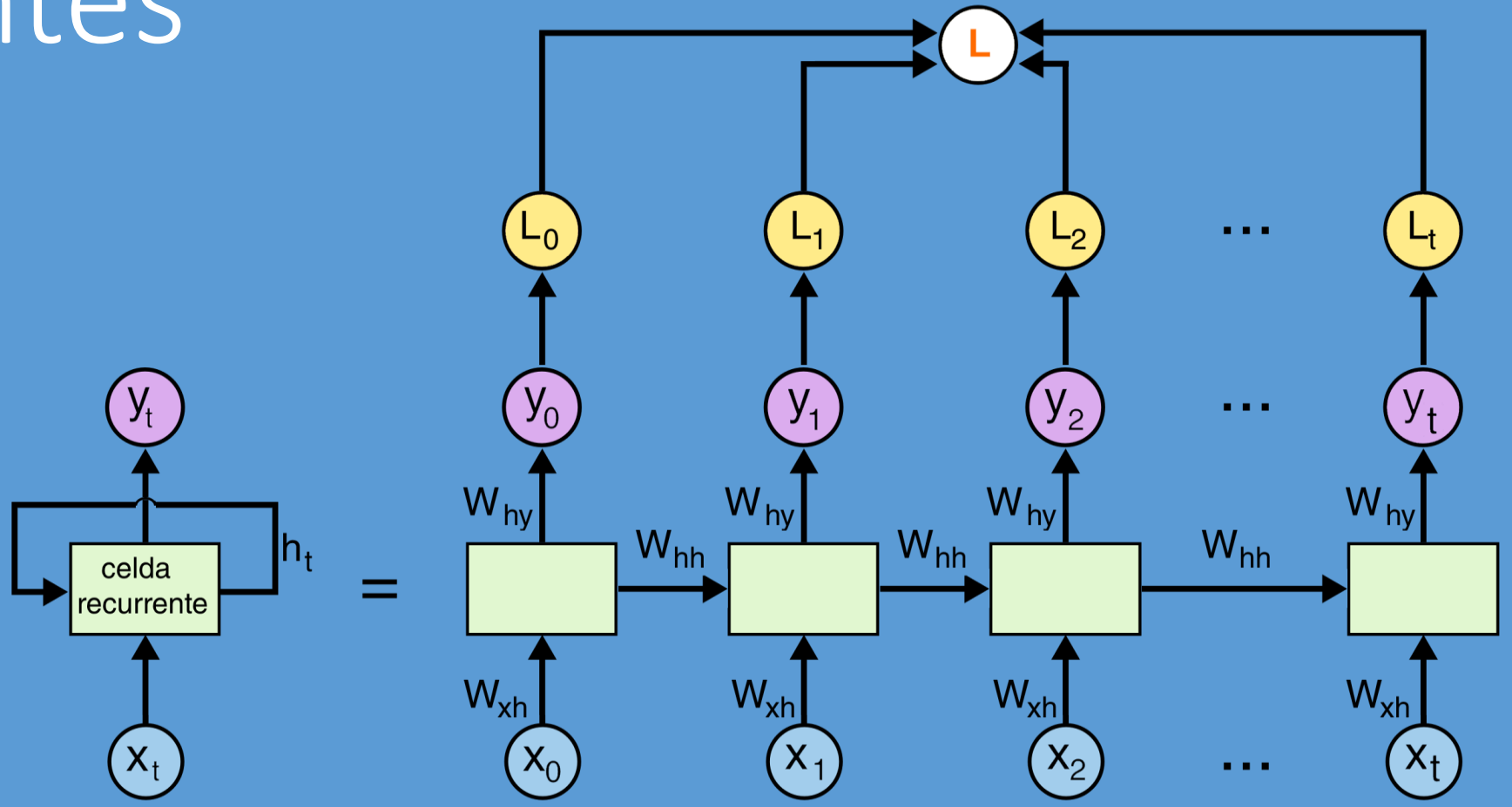
- 560 servidores Nvidia Selene - DGX A100
- Cada Servidor con 8 GPUs NVIDIA A100 80GB Tensor Core

TAXONOMÍA DE DEEP LEARNING

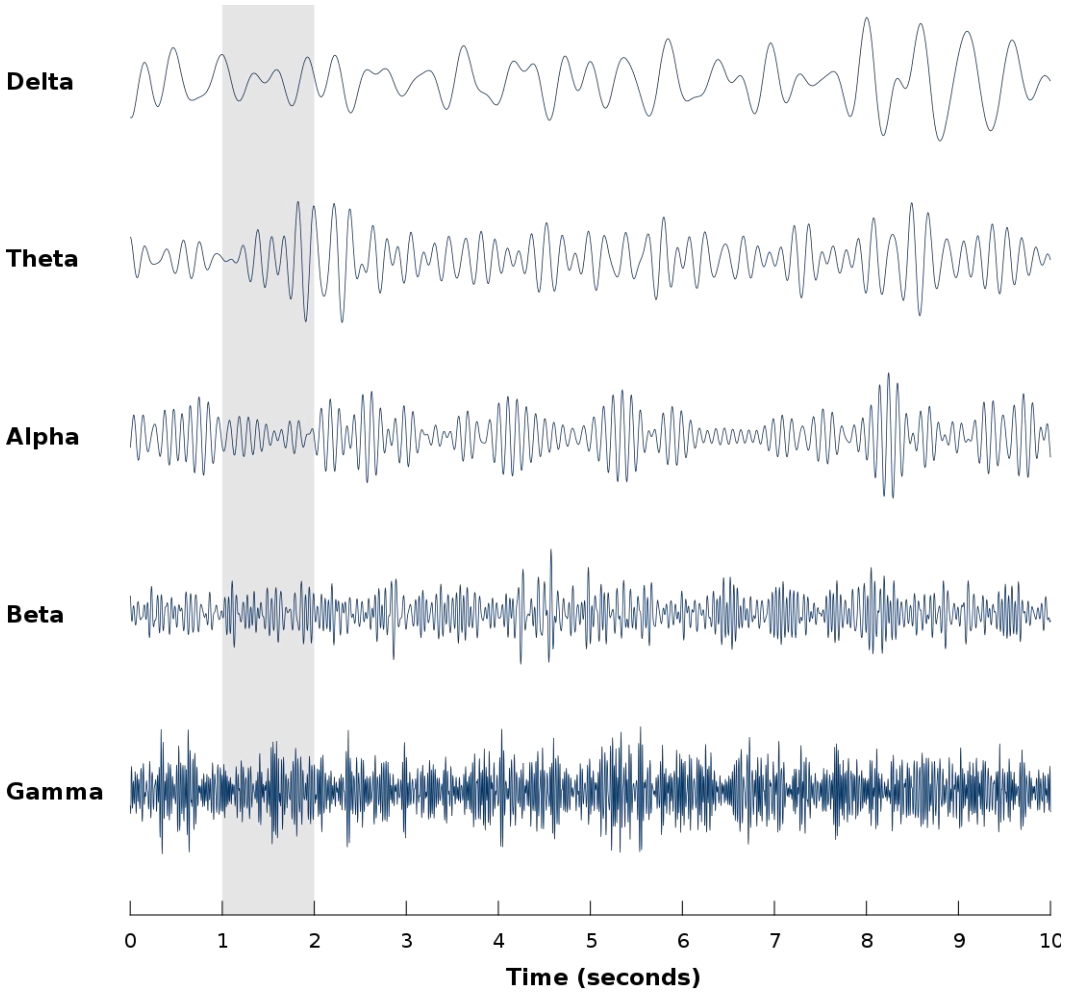


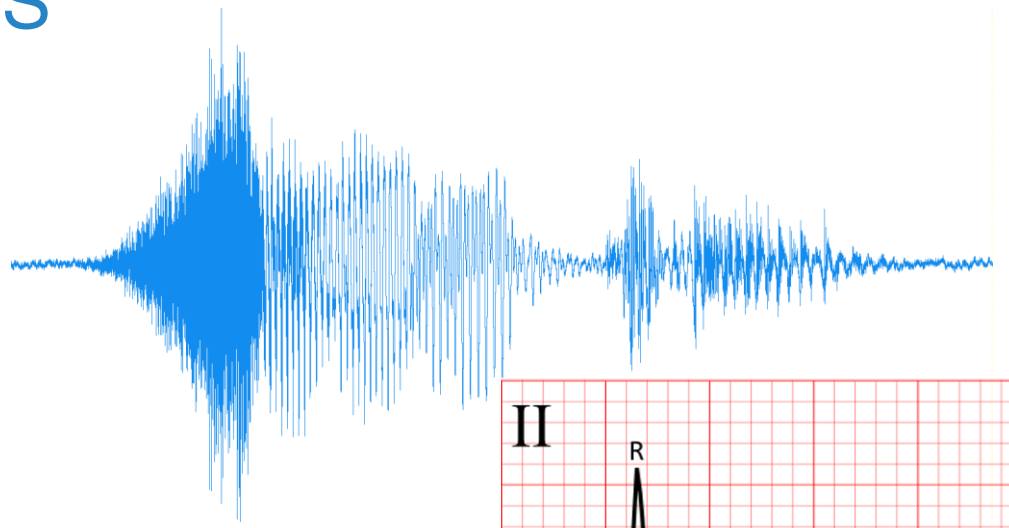
Transformers

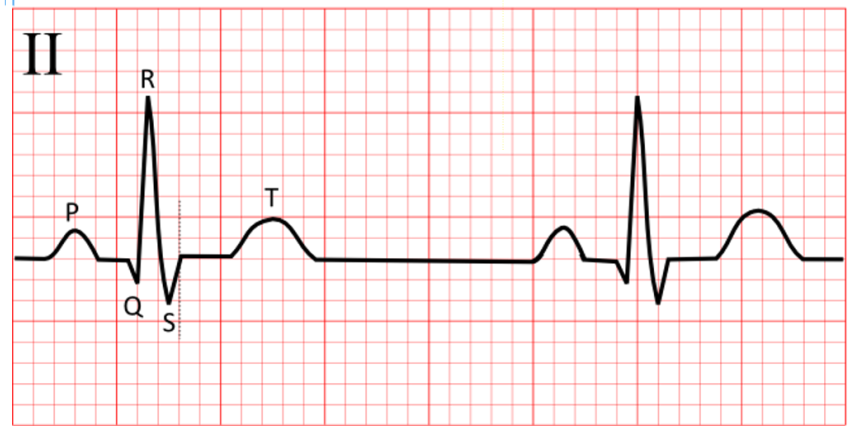
Redes Neuronales Recurrentes

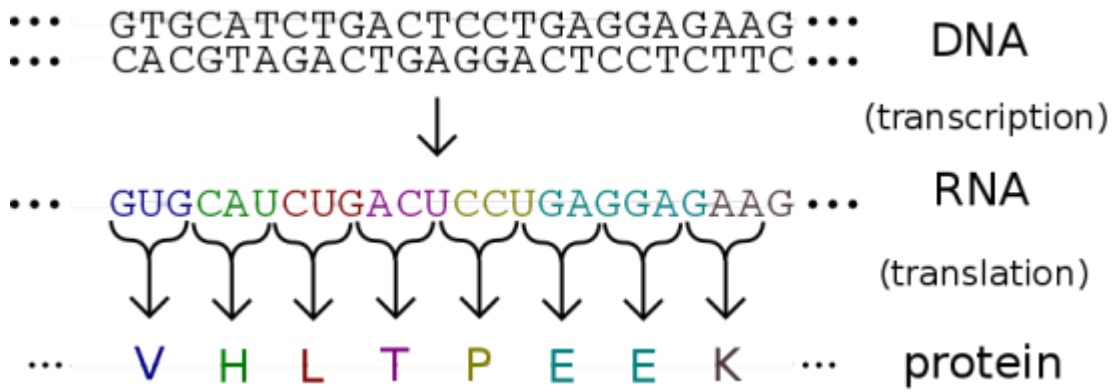


MODELADO DE SECUENCIAS







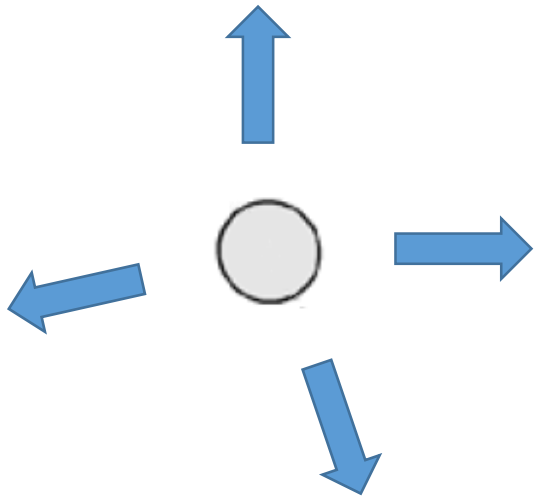


MODELADO DE SECUENCIAS

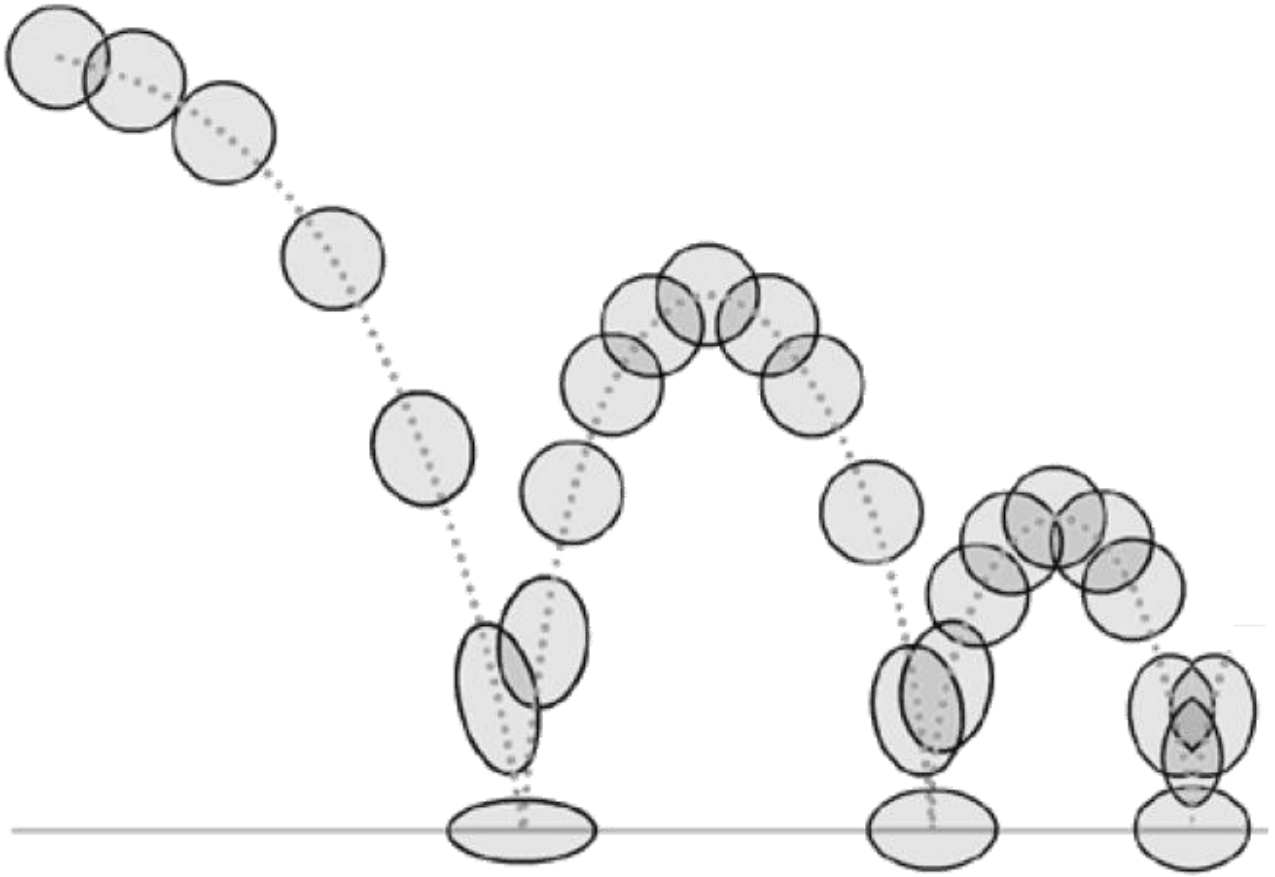
A young boy is playing basketball. 	Two dogs play in the grass. 	A dog swims in the water. 	A little girl in a pink shirt is swinging. 
A group of people walking down a street. 	A group of women dressed in formal attire. 	Two children play in the water. 	A dog jumps over a hurdle. 

MODELADO DE SECUENCIAS

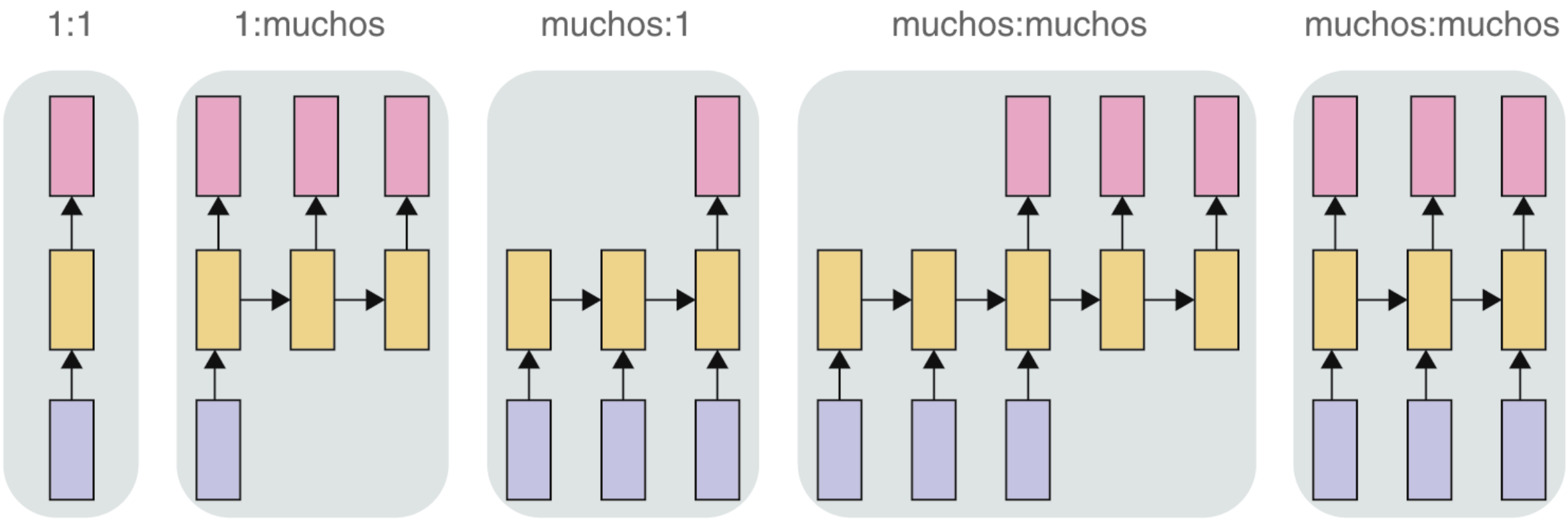
¿Cuál será la trayectoria de la pelota?



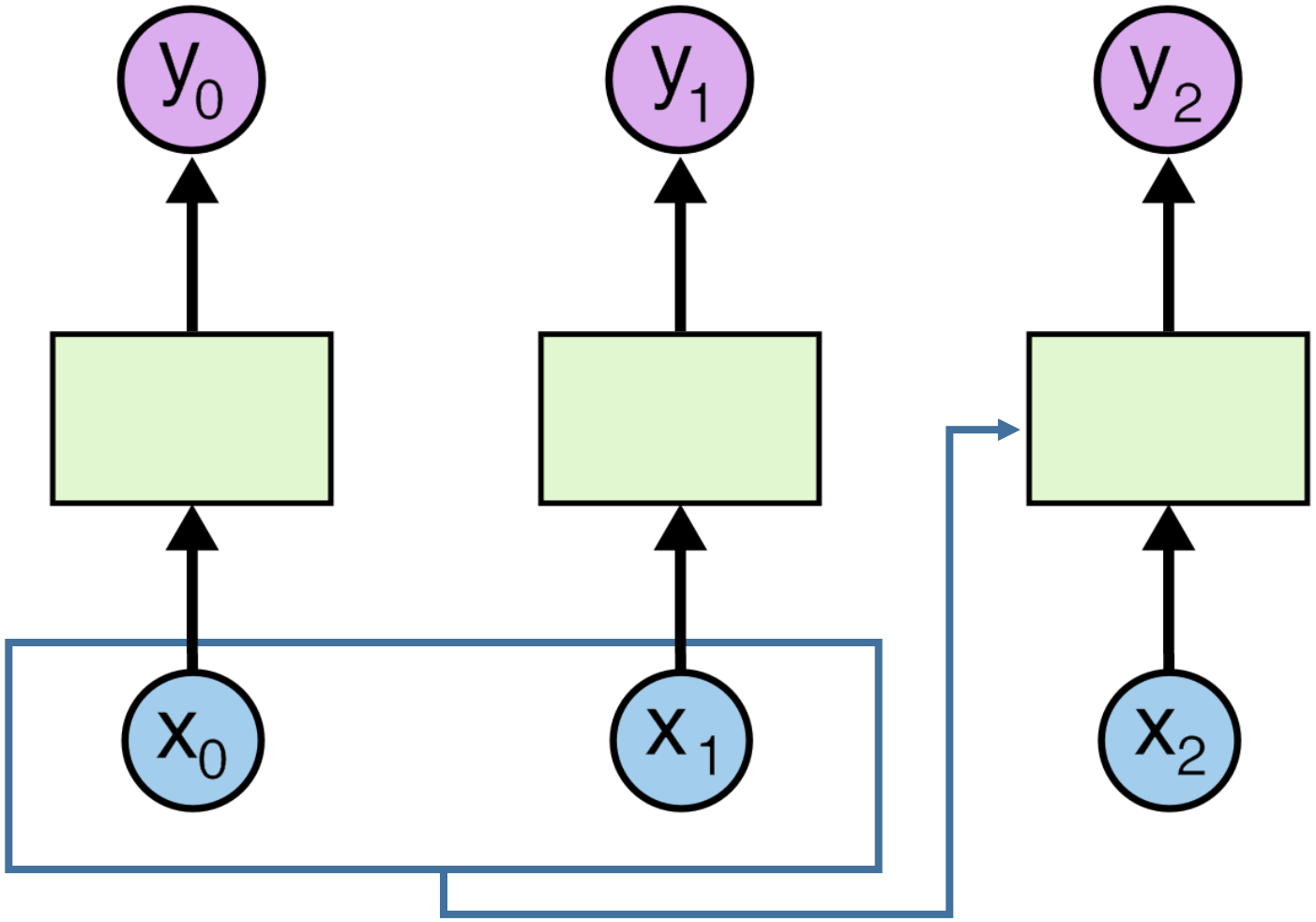
MODELADO DE SECUENCIAS



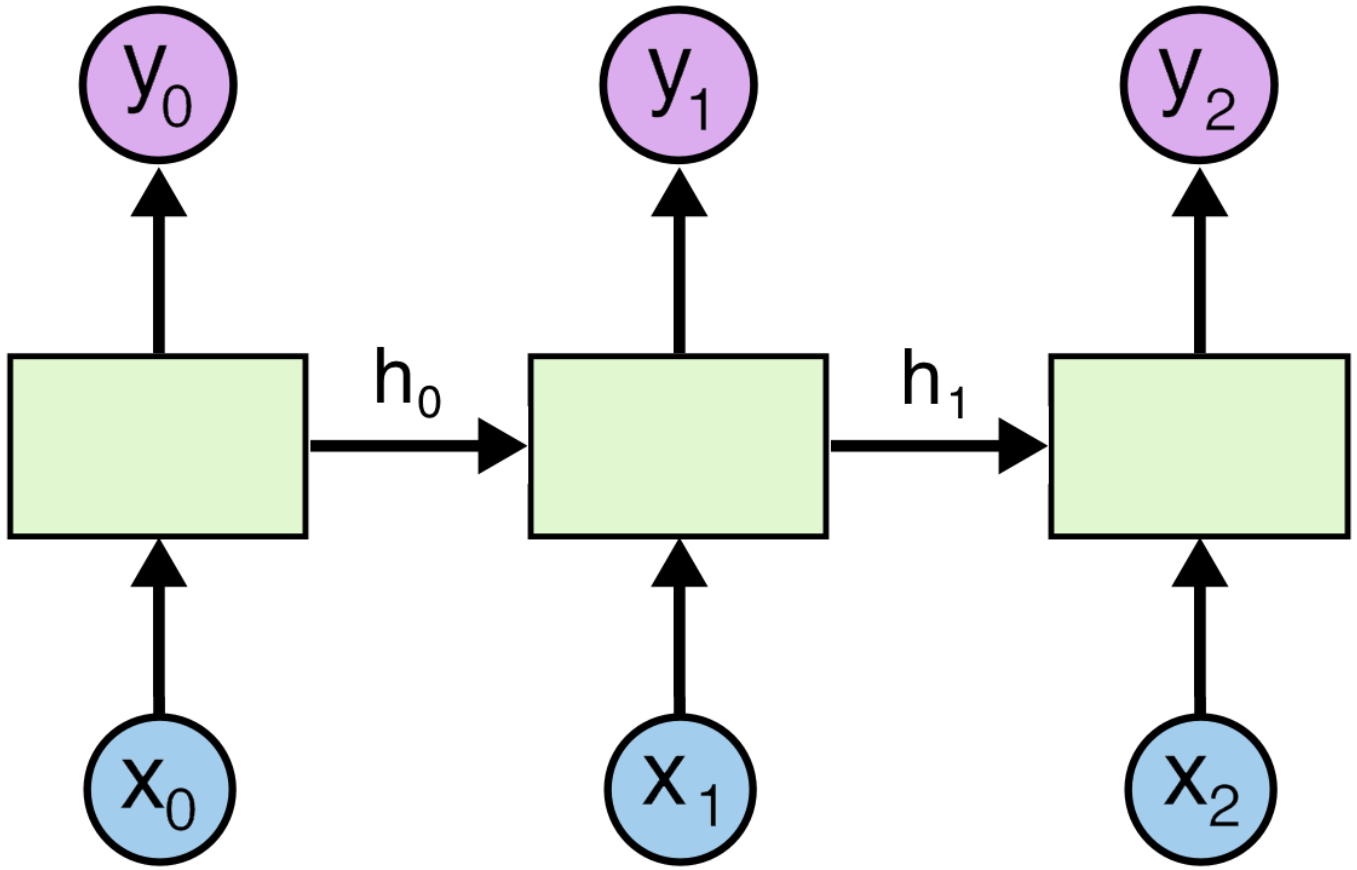
MODELADO DE SECUENCIAS Y DEEP LEARNING



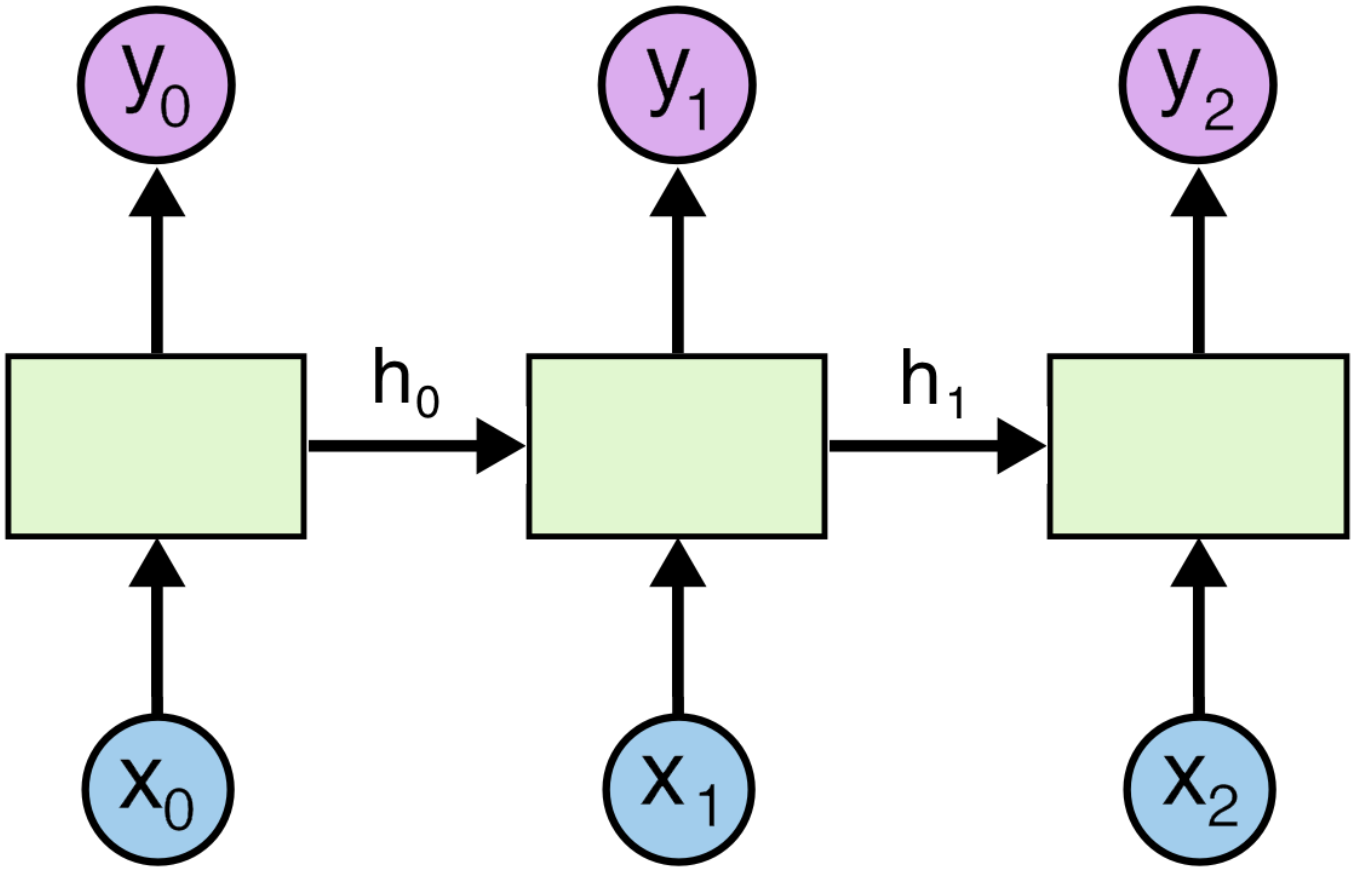
MODELADO DE SECUENCIAS Y DEEP LEARNING

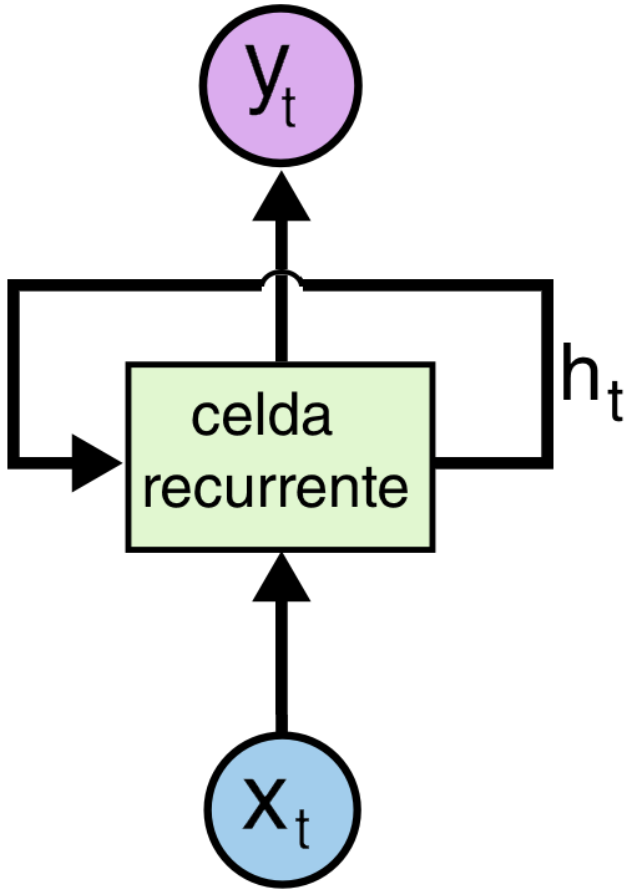


MODELADO DE SECUENCIAS Y DEEP LEARNING



MODELADO DE SECUENCIAS Y DEEP LEARNING

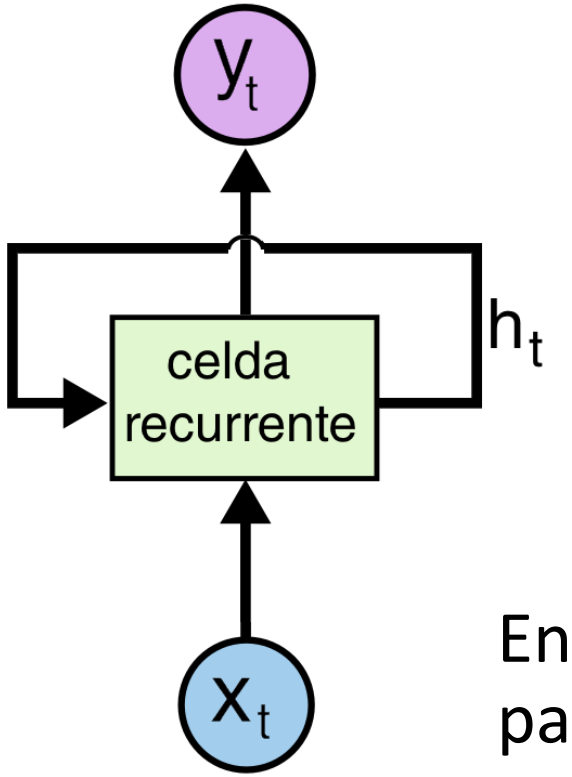




$$y_t = f(x_t, h_{t-1})$$

REDES NEURONALES RECURRENTES (RNN)

Las RNN tienen un estado h_t que se actualiza cada paso de tiempo a medida que se procesa la secuencia



Relación de recurrencia

Función con pesos *w*

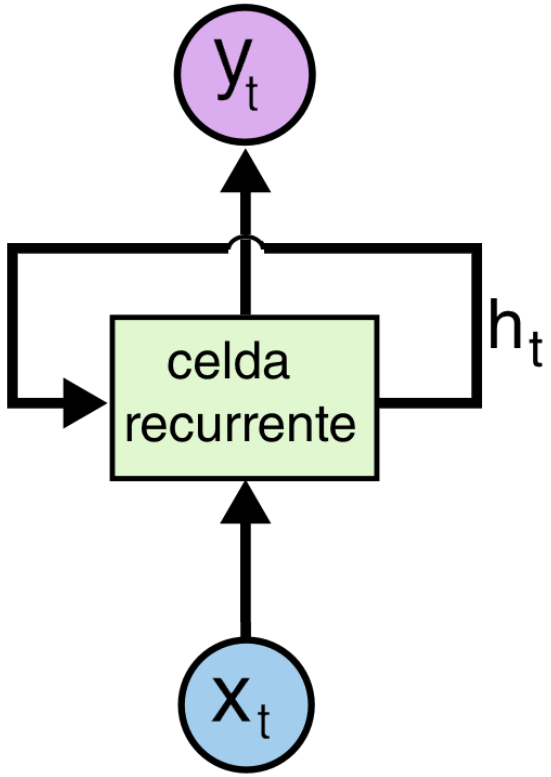
$$h_t = f_w(x_t, h_{t-1})$$

Entrada actual

Estado previo

En cada paso se usan la misma función y conjunto de parámetros

REDES NEURONALES RECURRENTES (RNN)



Actualización de estados y salidas

- Entradas: x_t

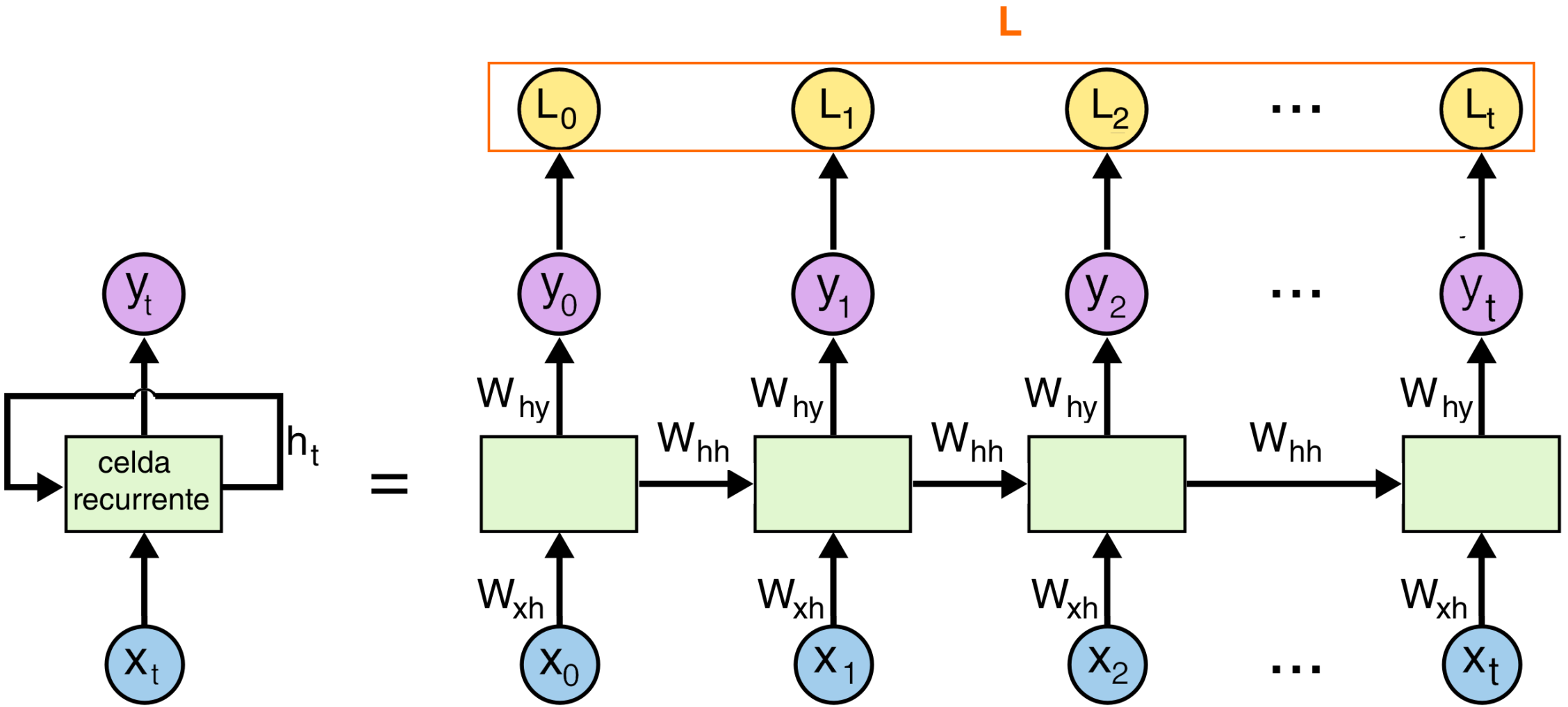
- Estados ocultos:

$$h_t = \tanh(W_{hh}^T \, h_{t-1} + W_{xh}^T \, x_t)$$

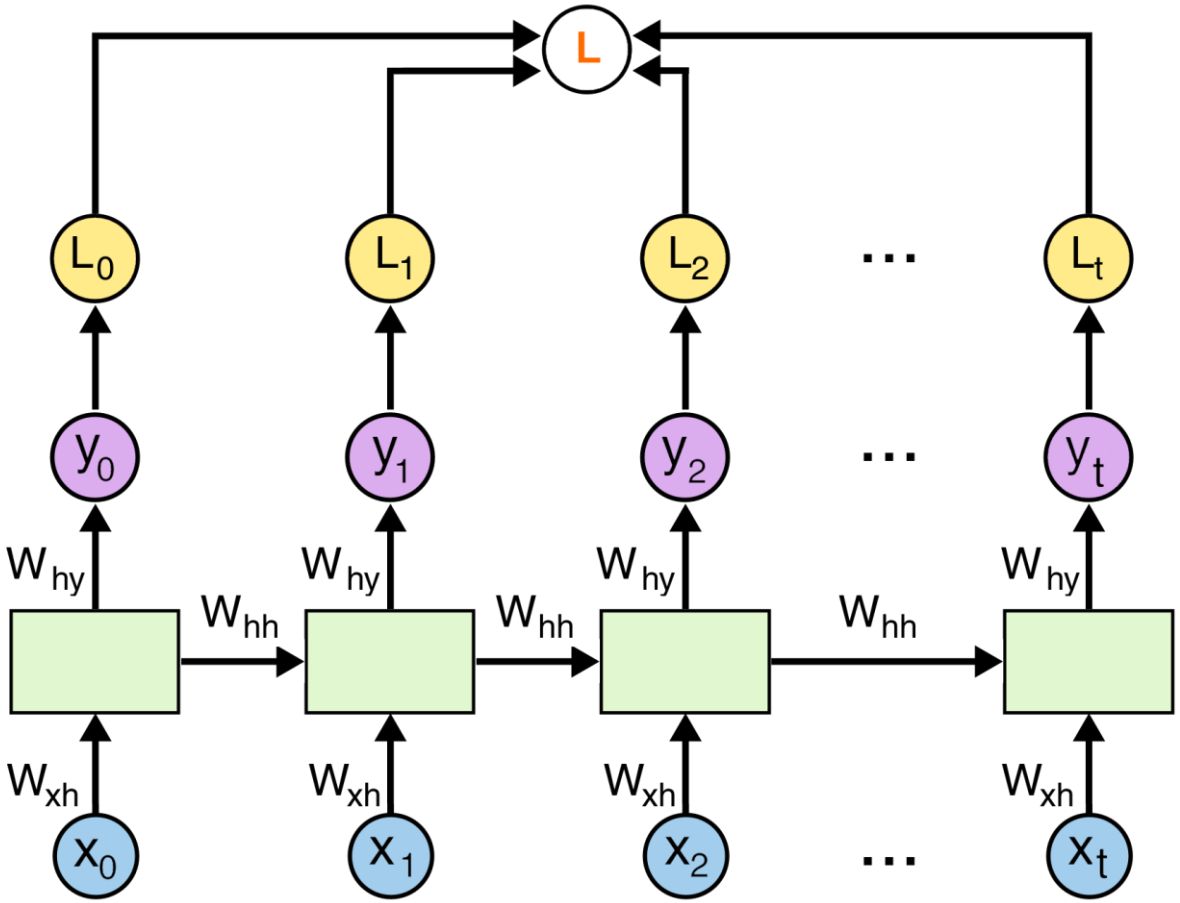
- Salidas:

$$y_t = W_{hy}^T \, h_t$$

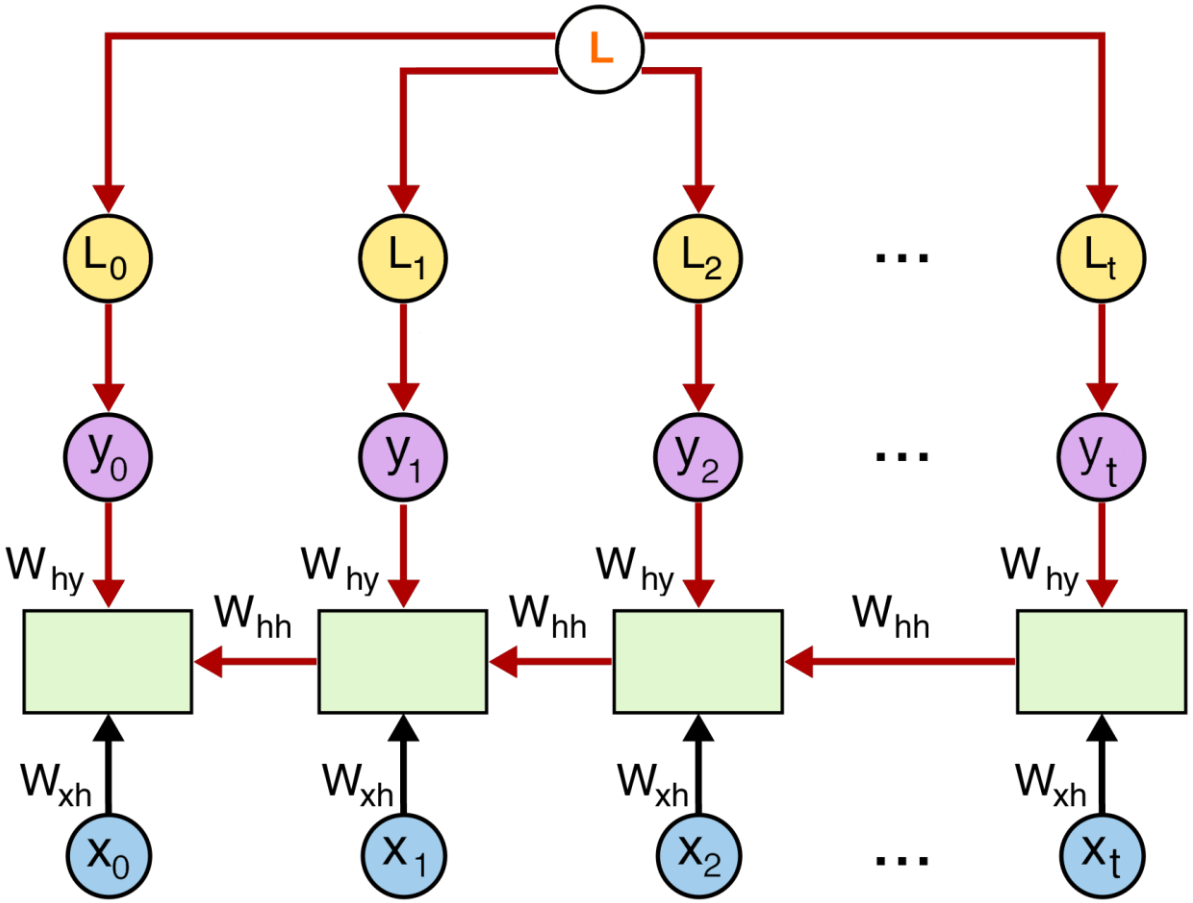
RNN – DESPLEGADO TEMPORAL



RNN – PASADA HACIA ADELANTE



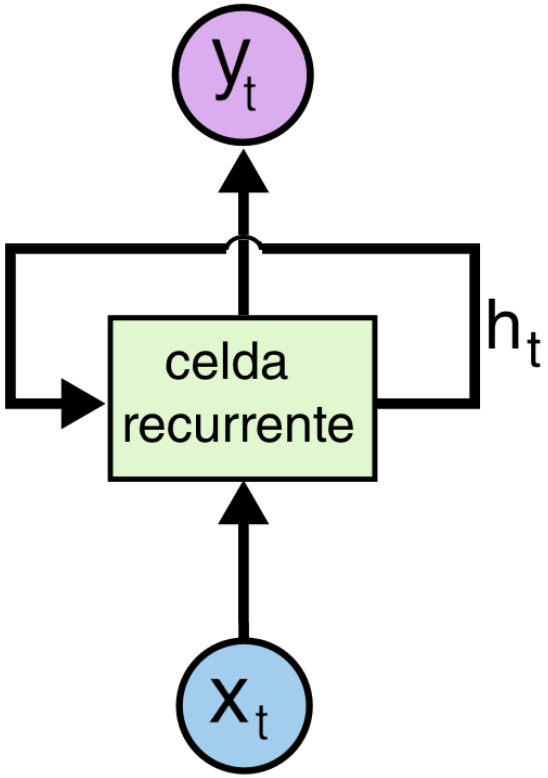
RNN – RETROPROPAGACIÓN EN EL TIEMPO (BPTT)



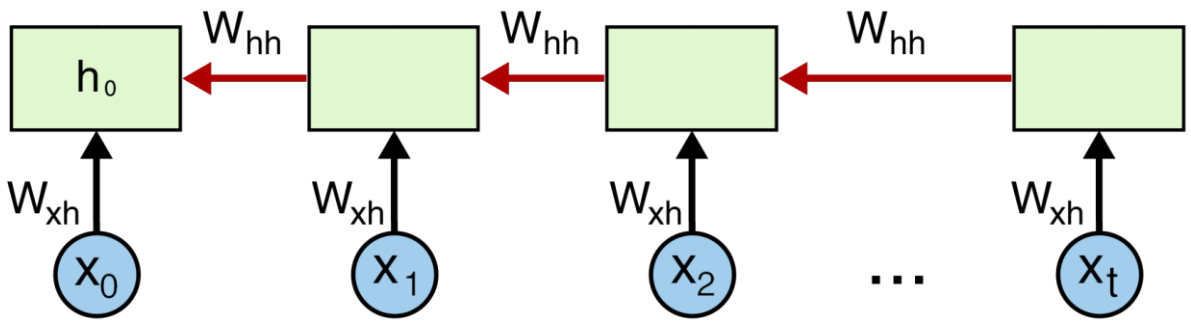
REDES NEURONALES RECURRENTES (RNN)

Las RNN satisfacen los siguientes requisitos:

- Admitir secuencias de longitud variable
- Admitir dependencias de largo plazo
- Conservar información del orden
- Compartir parámetros a lo largo de la secuencia



LIMITACIONES DE LAS RNN



- Cálculo del gradiente respecto a h_0 demanda muchas matrices W_{hh} y cálculo de otros gradientes → alto costo computacional
- Otros dos problemas: explosión y desvanecimiento de gradientes

short term memory

- Explosión de gradientes (muchos valores > 1)
  - Truncado de gradiente
- Desvanecimiento de gradientes (muchos valores < 1)
  - Función de activación
  - Inicialización de pesos
  - Modificación de arquitectura

EJEMPLO – PREDICCIÓN DE TEXTO

Dado una cadena de texto, predecir la siguiente palabra

"Saqué a mi perro a ....."

pasear

Codificación de entradas:

a perro
dar
pasear
saqué
mi un

1. Lexicón

a 1
mi 2
pasear 3
perro 4
saqué 5

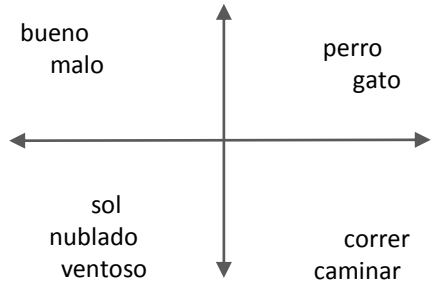
2. Indexado

One-hot

mi -> [01000]

índice 2

Aprendida



3. Embedding

EJEMPLO – PREDICCIÓN DE TEXTO

Desvanecimiento de gradiente

↓

Multiplicación sucesiva de valores < 0, el gradiente se anula para pasos distantes

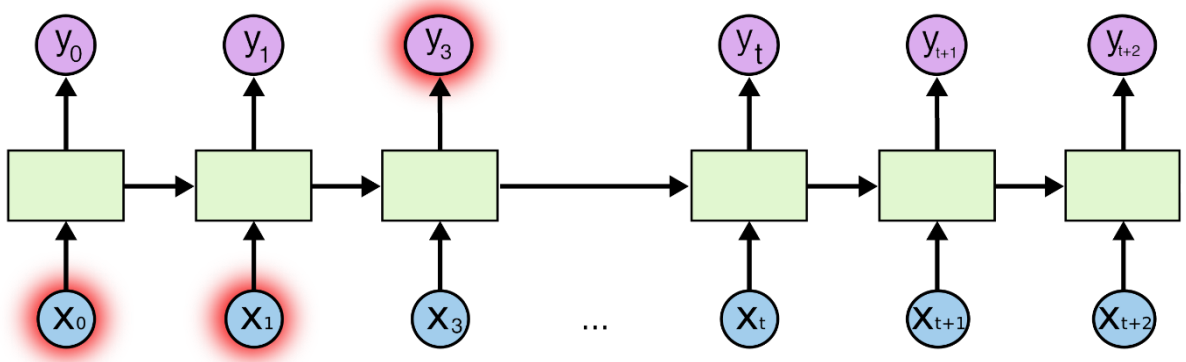
↓

Se genera sesgo hacia dependencias temporales recientes

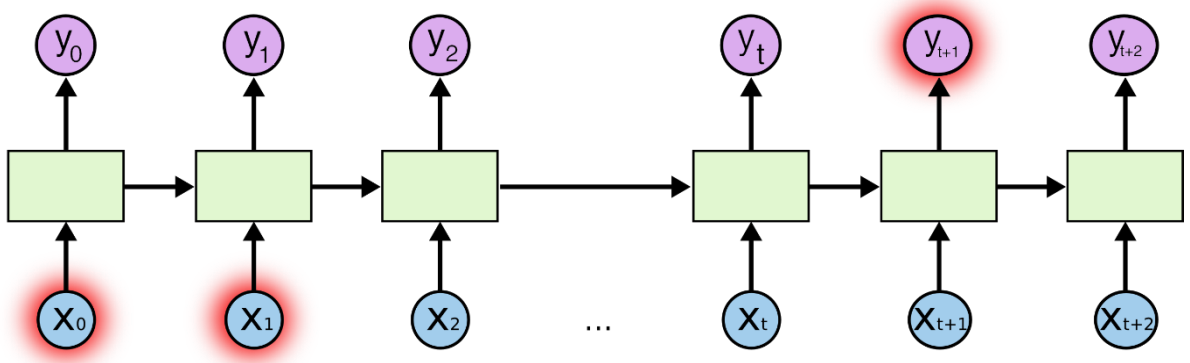
↓

Se pierden dependencias a largo plazo

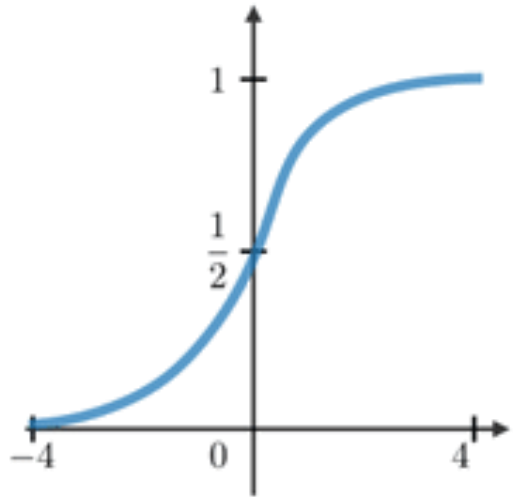
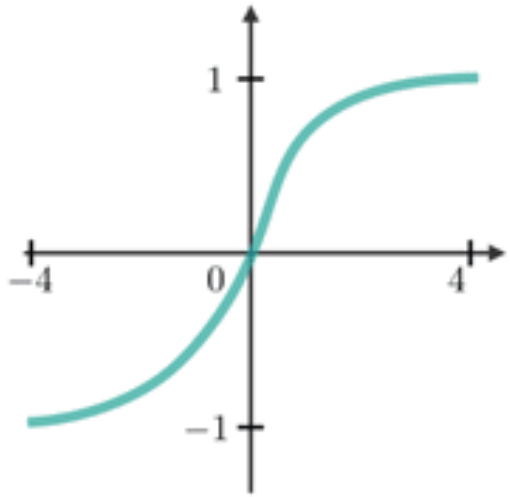
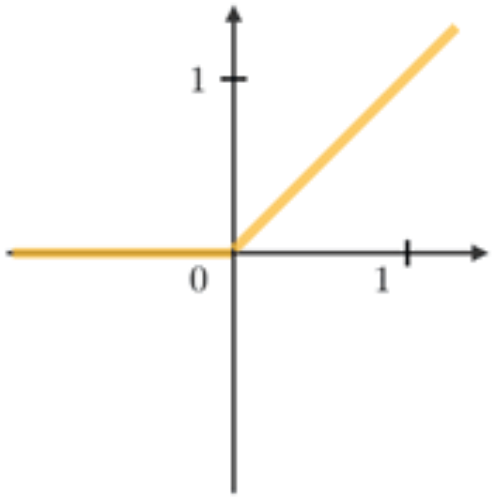
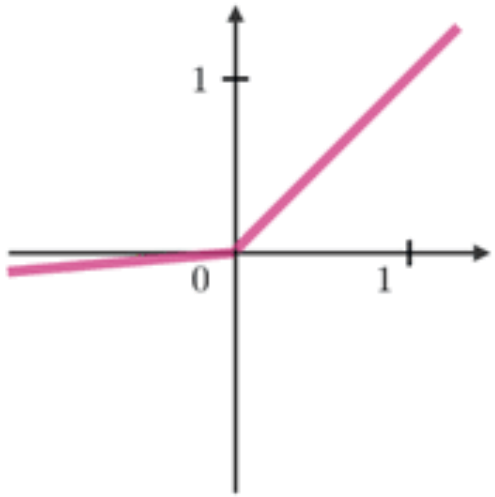
“El perro ..ladra..”



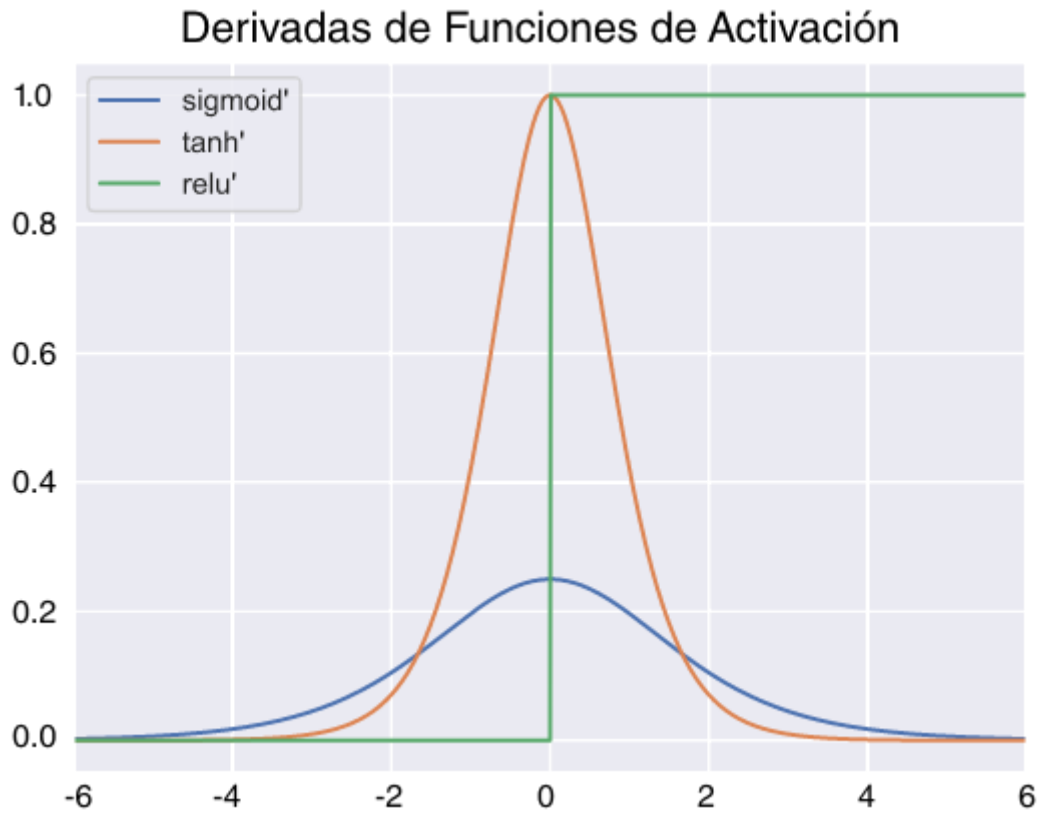
“Soy francés, crecí en…, hablo bien…..francés…..”



TRUCOS PARA RNN - FUNCIÓN DE ACTIVACIÓN

Sigmoid	Tanh	ReLU	Leaky ReLU
$g(z) = \frac{1}{1+e^{-z}}$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \max(0, z)$	$g(z) = \max(\epsilon z, z)$ with $\epsilon \ll 1$
			

TRUCOS PARA RNN - FUNCIÓN DE ACTIVACIÓN



ReLU evita que el gradiente disminuya cuando $x > 0$

TRUCOS PARA RNN - INICIALIZACIÓN DE PARÁMETROS

Se puede evitar que los pesos de RNNs con funciones de activación ReLU tiendan rápidamente a 0, inicializando:

- Matrices de pesos: matriz identidad
- Bias: vector 0

$$W_{hh} = \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix}$$

Le, Q. V., Jaitly, N., & Hinton, G. E. (2015). A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*.

TRUCOS PARA RNN – CELDAS DE COMPUERTA

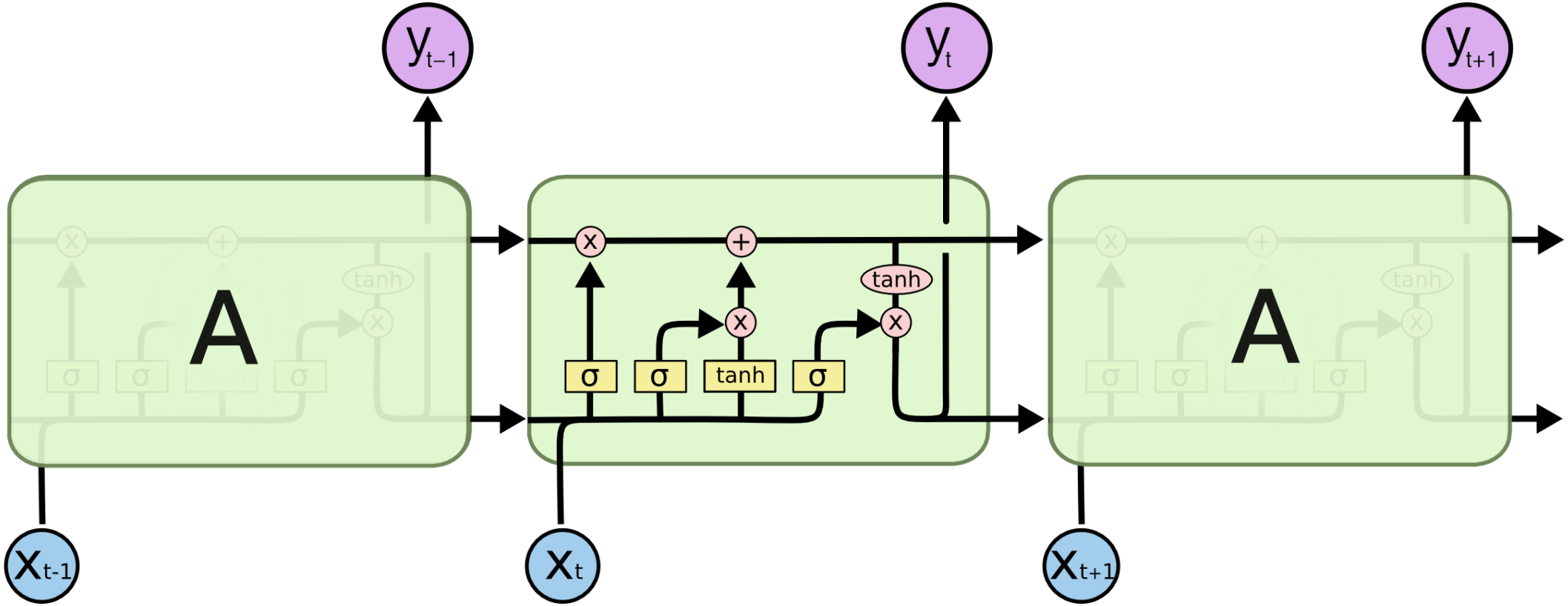
Idea:

- Diseñar una arquitectura para mitigar el desvanecimiento de gradientes
- Usar compuertas para agregar o eliminar selectivamente información dentro de cada unidad recurrente (**recordar sólo lo importante**)
- Se implementan con operaciones estándar para redes neuronales
- Ejemplos LSTM, GRU, etc.

LONG SHORT TERM MEMORY (LSTM)

- Estructura de cadena similar a una RNN estándar
- La celda de estado/memoria compuesta por celdas:
  - Olvido: para eliminar información irrelevante del pasado
  - Almacenamiento: para mantener información relevante
  - Actualización: para actualizar el estado interno de la celda
  - Salida: para generar como predicción de salida una versión filtrada del estado de la celda
- Las compuertas dan estabilidad al *backpropagation* (menos multiplicaciones y parámetros) reducen efecto de desvanecimiento

LONG SHORT TERM MEMORY (LSTM)



ALTERNATIVAS AL MODELADO DE SECUENCIAS

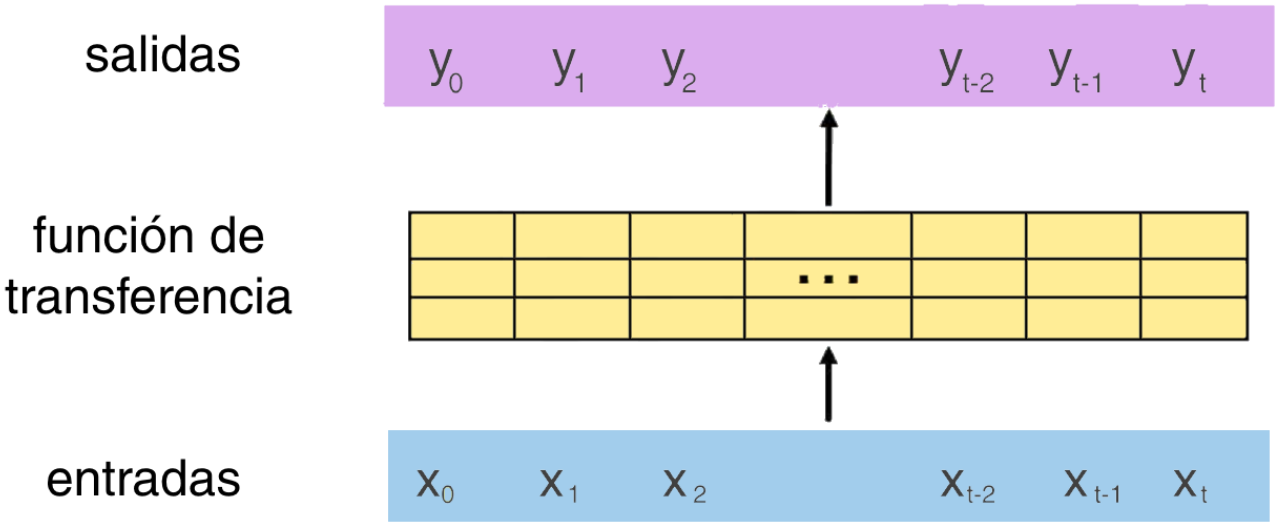
Limitaciones de redes recurrentes

- Cuello de botella de codificación: la necesidad de codificar las entradas implica pérdida de información
- Ineficientes: requieren procesamiento secuencial, limita la capacidad de paralelizar
- Limitación en la memoria de largo plazo: no escalan bien a secuencias con longitudes de miles o decenas de miles de pasos de tiempo

ALTERNATIVAS AL MODELADO DE SECUENCIAS

Idea:

- Concatenar los pasos de tiempo individuales y formar un vector de entrada con datos de todos los puntos de tiempo
- Usar una red densa para la clasificación



- No recurrente
- No escalable
- Pierde información de orden
- No tiene memoria a largo plazo

MECANISMO DE ATENCIÓN

Idea:

- En lugar de intentar prestar atención a todas las muestras, ser más selectivo
- Introducir noción de qué puntos de la secuencia son importantes
- Es la idea clave detrás del concepto de **atención**
- Capacidad de identificar y ***atender*** a las partes de una entrada relevantes es uno de los avance más valiosos de la última década
- Extremadamente potente y la pieza fundamental de modelos como el BERT de Google y de los ***Transformers***

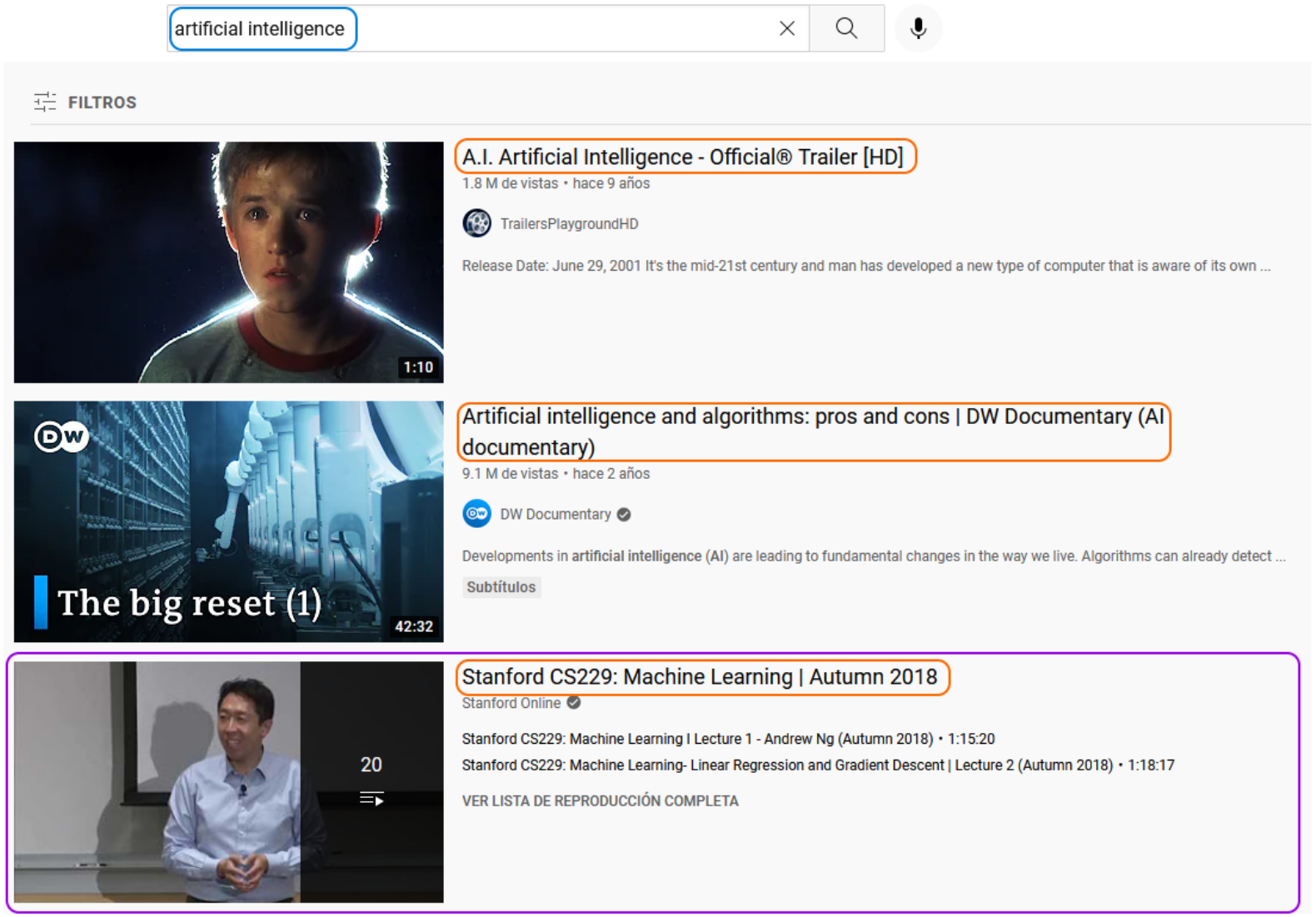
MECANISMO DE ATENCIÓN

MECANISMO DE ATENCIÓN

Mecanismo de búsqueda

- Identificar las regiones que se deben atender
- Obtener atributos que caracterizan esas regiones

MECANISMO DE ATENCIÓN



Query

Key (K_1)

Key (K_2)

Key (K_3)

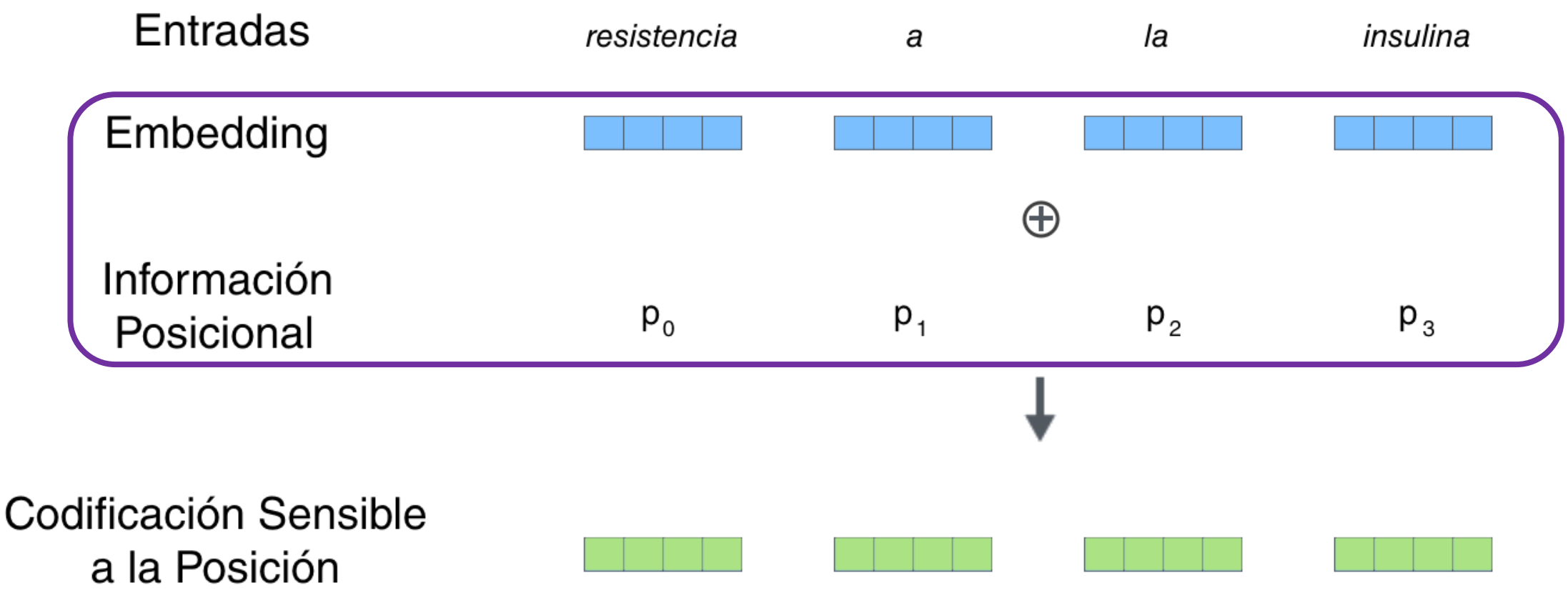
Value (V)

Similitud entre Key / Query (máscara de atención)

APRENDIZAJE DE AUTO-ATENCIÓN

Objetivo: identificar y atender a los atributos más importantes de la entrada

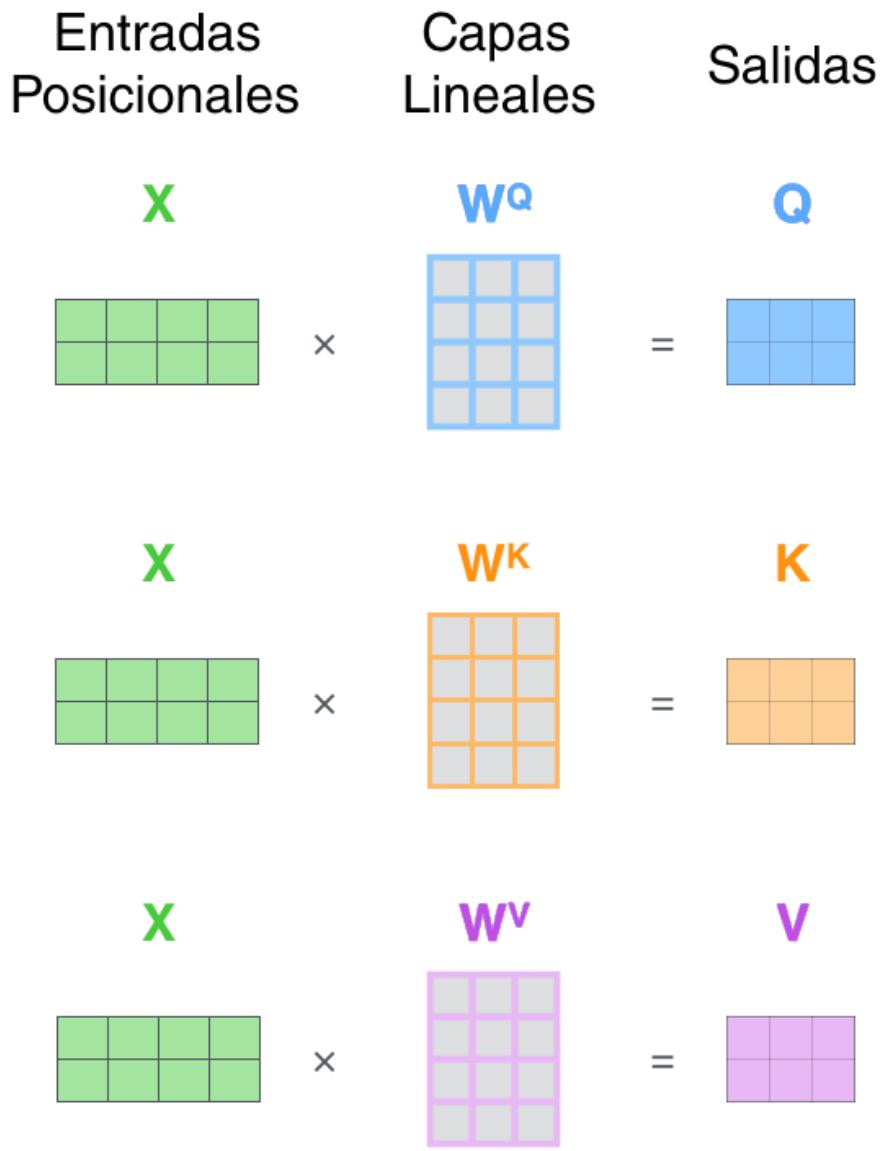
Paso 1: codificación



APRENDIZAJE DE AUTO-ATENCIÓN

Paso 2: extracción de Q , K y V

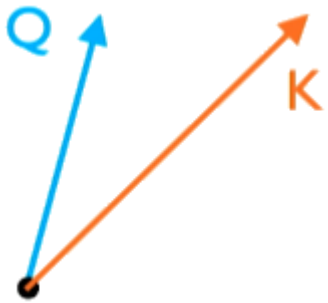
- Crear 3 transformaciones nuevas y únicas del embedding posicional
- Producto matricial entre el mismo embedding y una capa lineal diferente para cada elemento



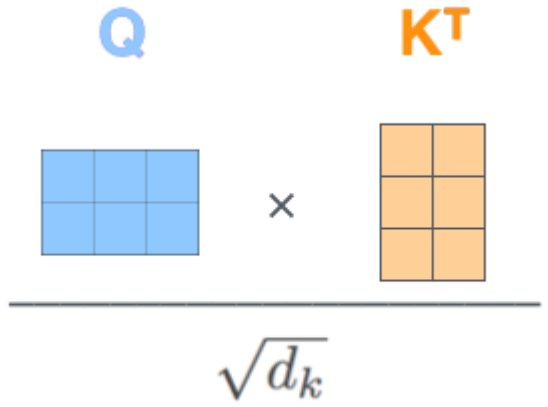
APRENDIZAJE DE AUTO-ATENCIÓN

Paso 3: calcular la ponderación de atención (cuánto y a dónde atender)

- Calcular similitud de a pares entre Q , K usando distancia coseno



$$\frac{Q \cdot K^T}{escala}$$

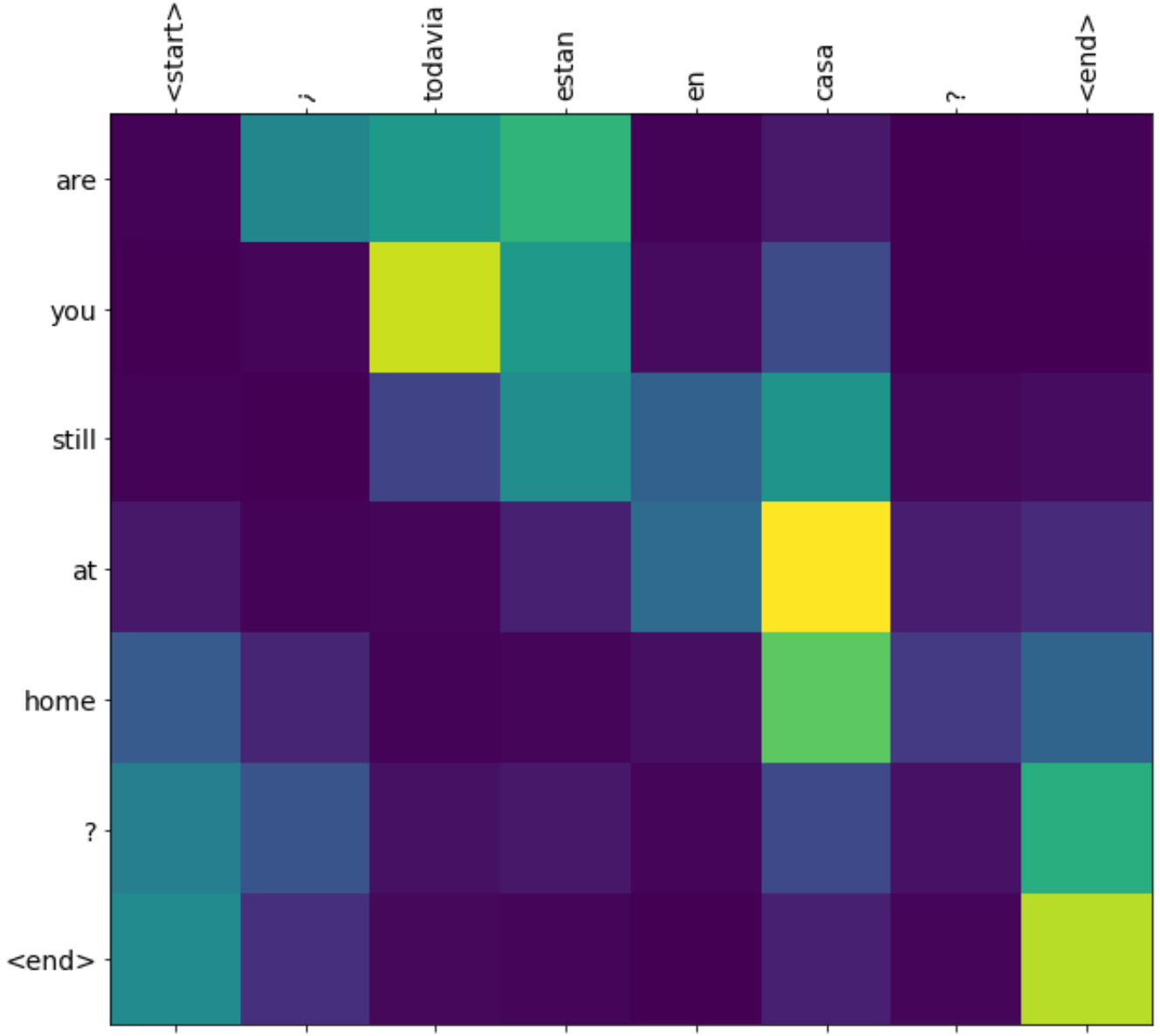


Resultado: matriz de atención

APRENDIZAJE DE AUTO-ATENCIÓN

Matriz de atención

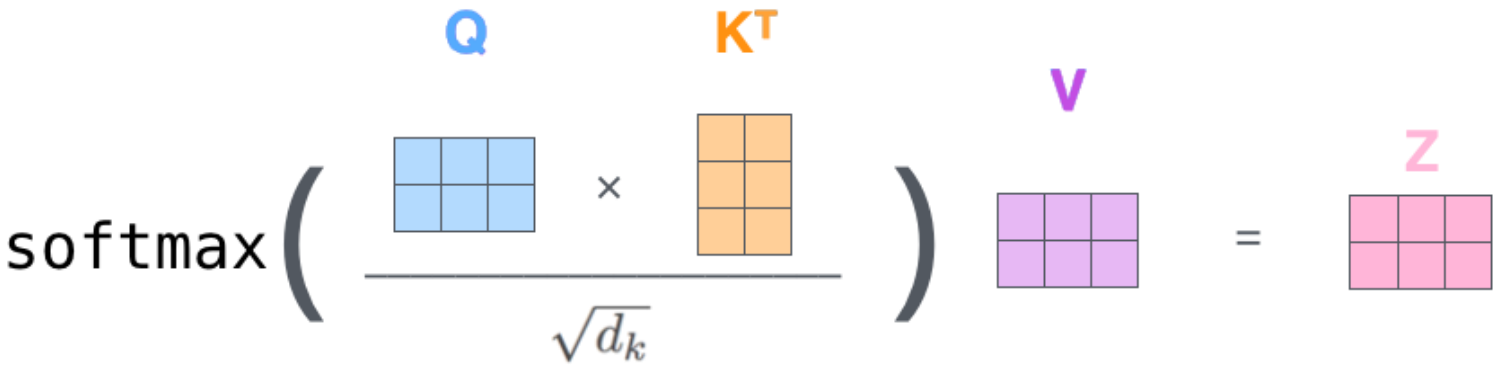
sus elementos reflejan las relaciones entre los componentes de las entradas



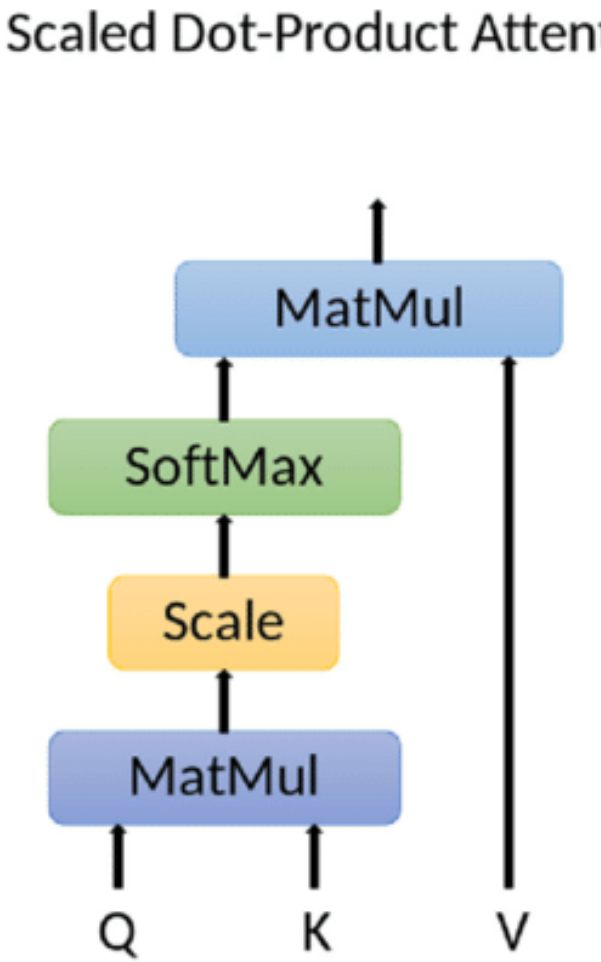
APRENDIZAJE DE AUTO-ATENCIÓN

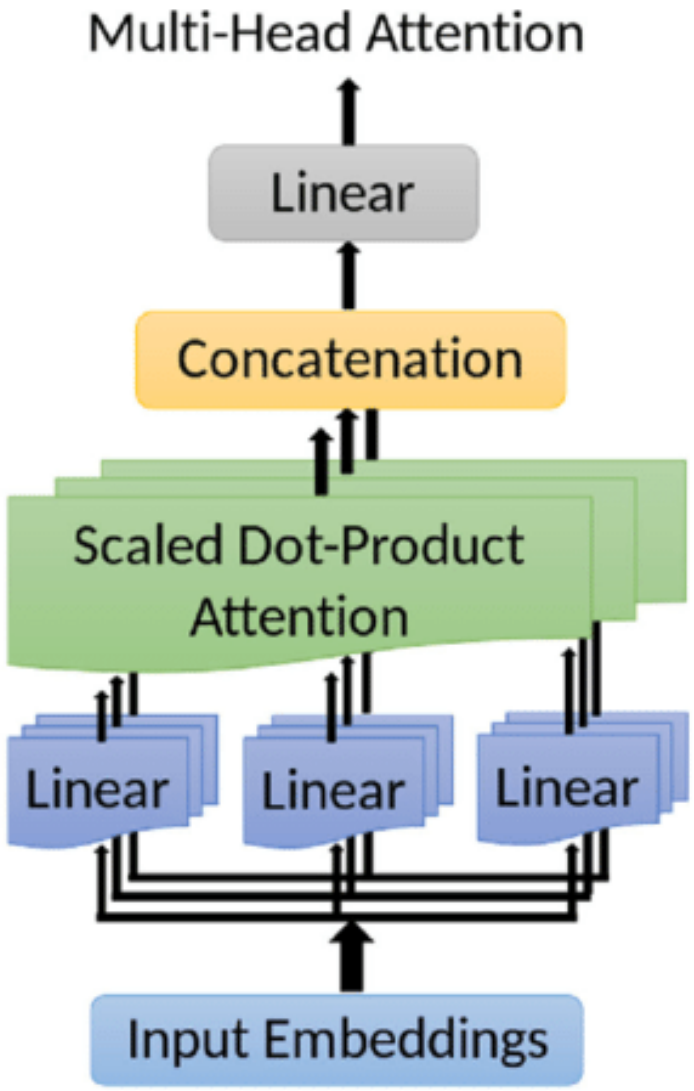
Paso 4: extraer atributos con mayor atención

- Aplicamos una función softmax a la matriz de ponderación de atención, para llevar sus valores al rango [0,1]
- La multiplicamos por la matriz de valores
- El resultado refleja las características que corresponden a los componentes de alta atención



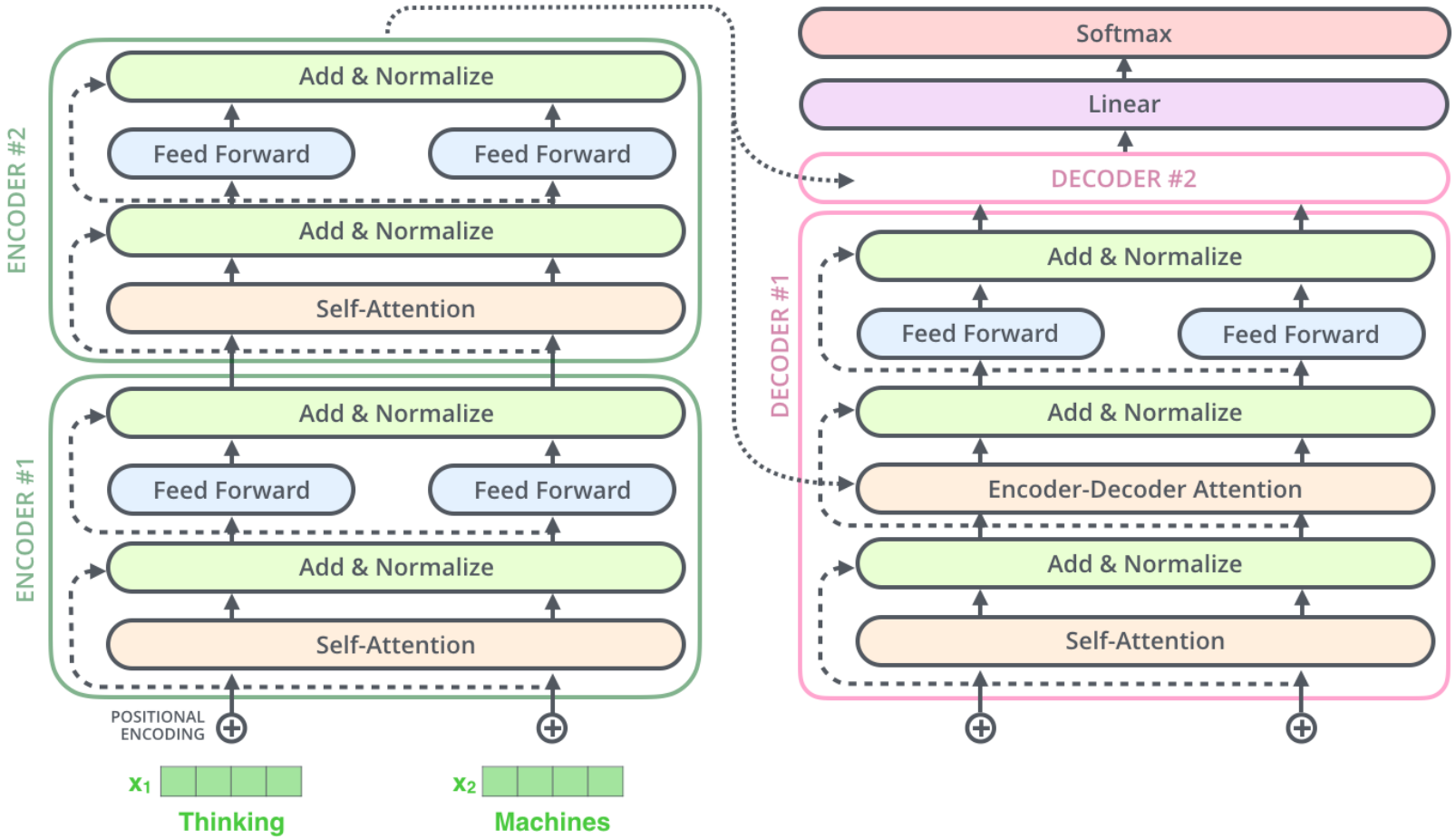
APRENDIZAJE DE AUTO-ATENCIÓN





TRANSFORMERS

- Los mecanismos de atención son el núcleo de las arquitecturas de Transformers. Ej.



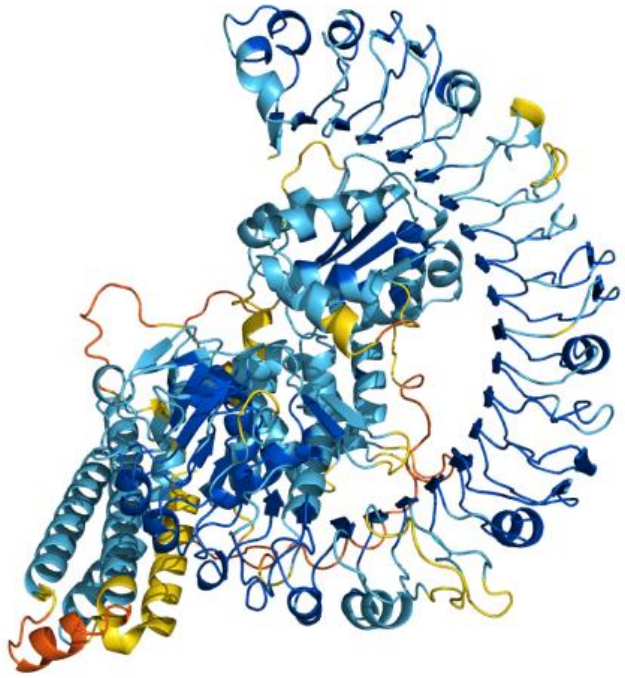
TRANSFORMERS EN NLP

- Síntesis de imágenes desde texto (BERT, GPT-3, T5, DALL-2)
- Copilot

“An armchair in the shape of an avocado”

TRANSFORMERS EN PROTEÓMICA

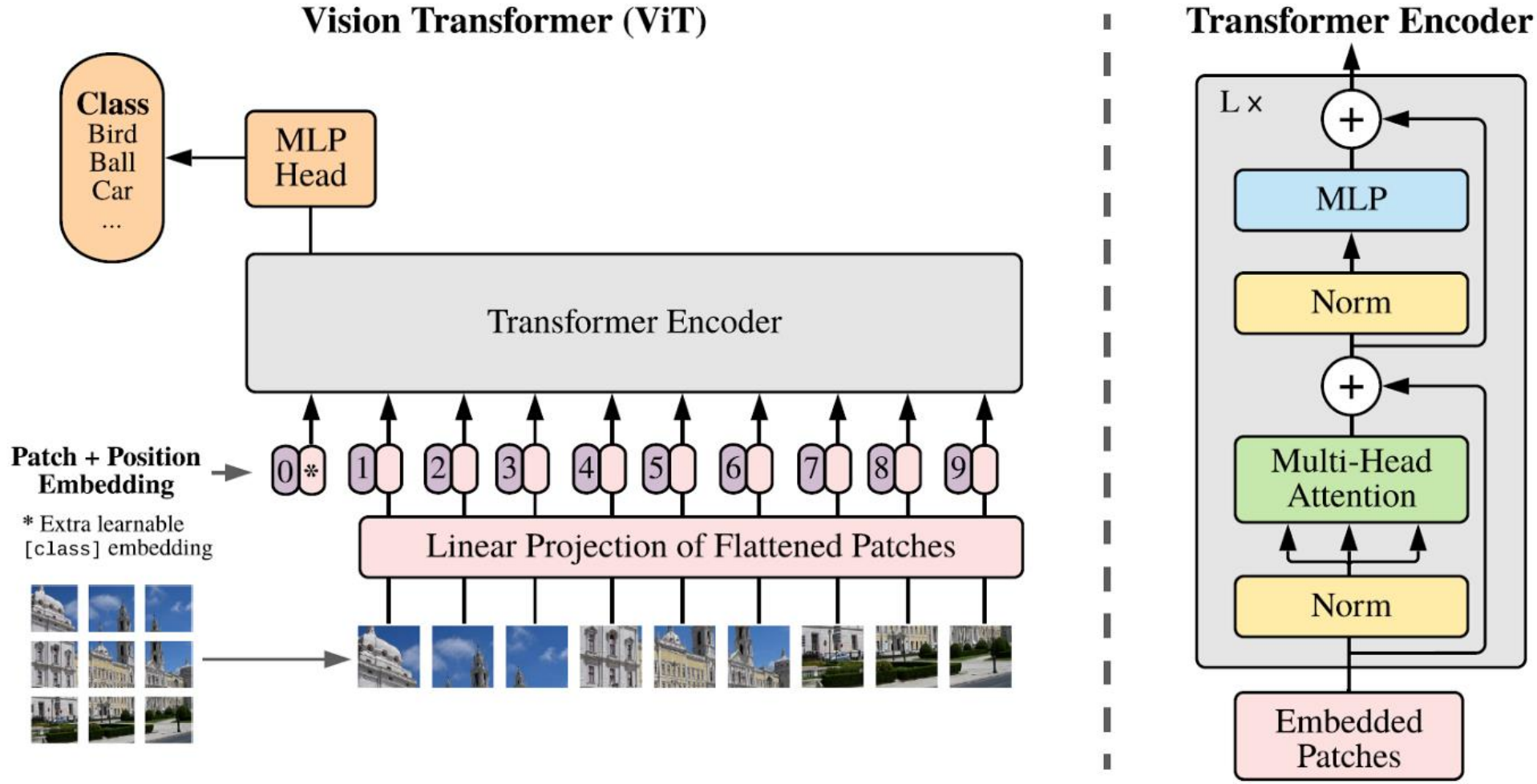
- Predicción de la estructura de las proteínas (AlphaFold2)



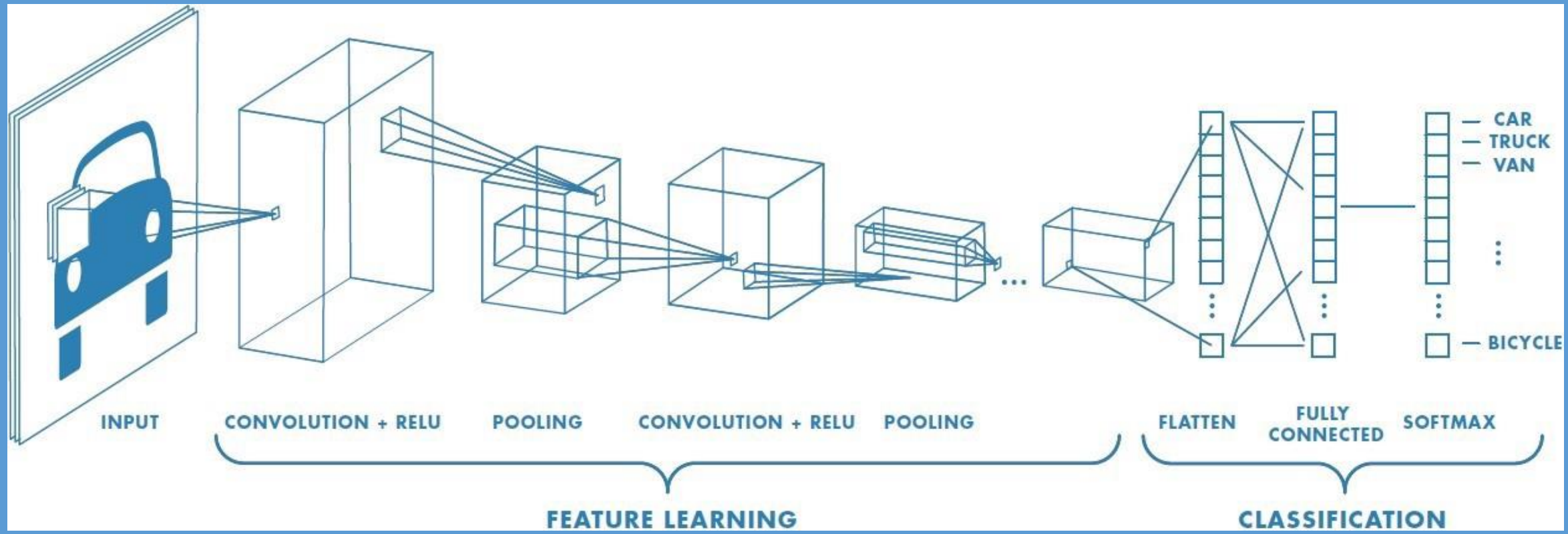
Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., ... & Hassabis, D. (2021). Highly accurate protein structure prediction with AlphaFold. Nature, 596(7873), 583-589.

TRANSFORMERS EN VISIÓN ARTIFICIAL

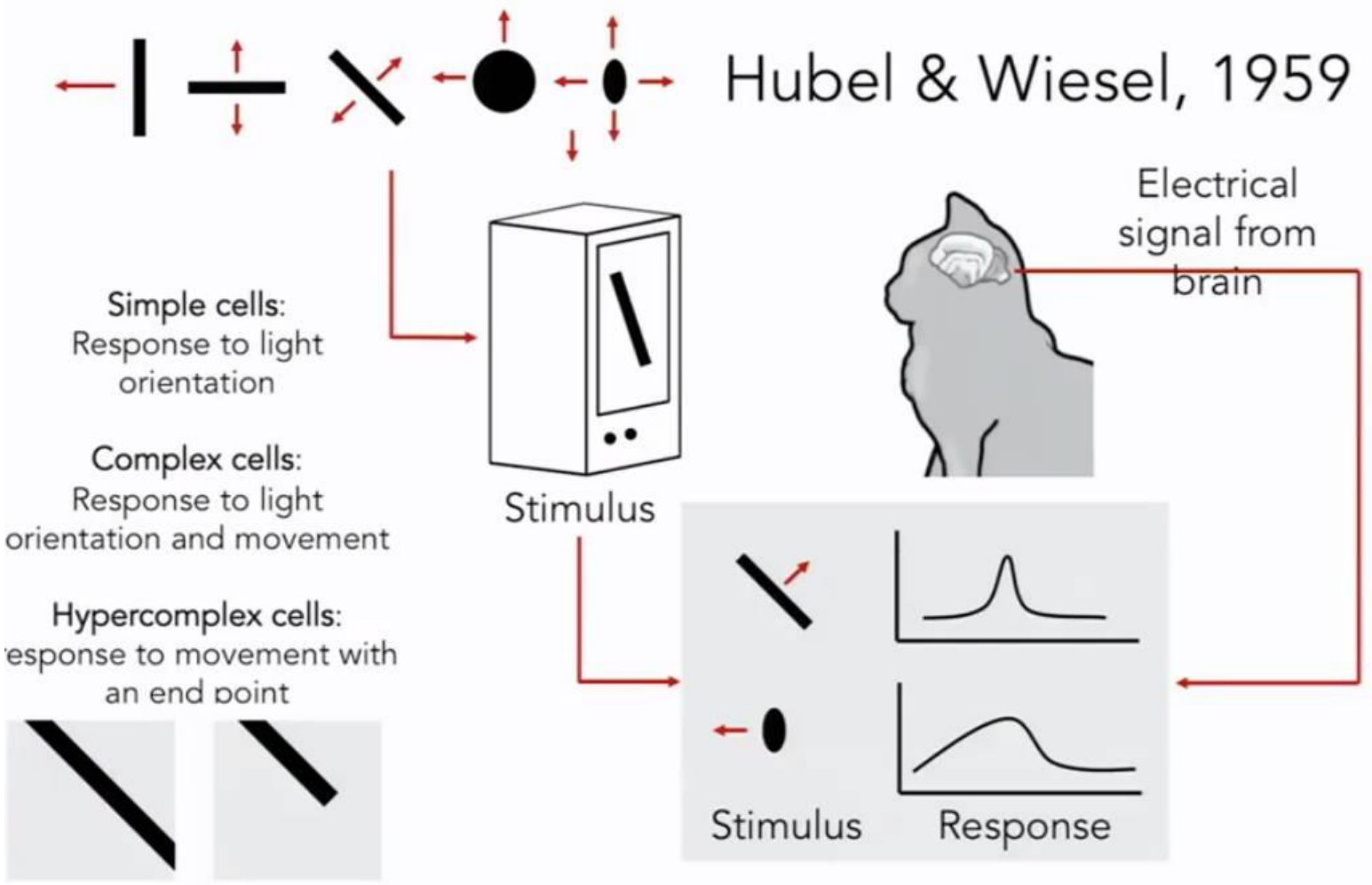
- Clasificación de imágenes (ViT – Vision Transformers)



Redes Neuronales Convolucionales



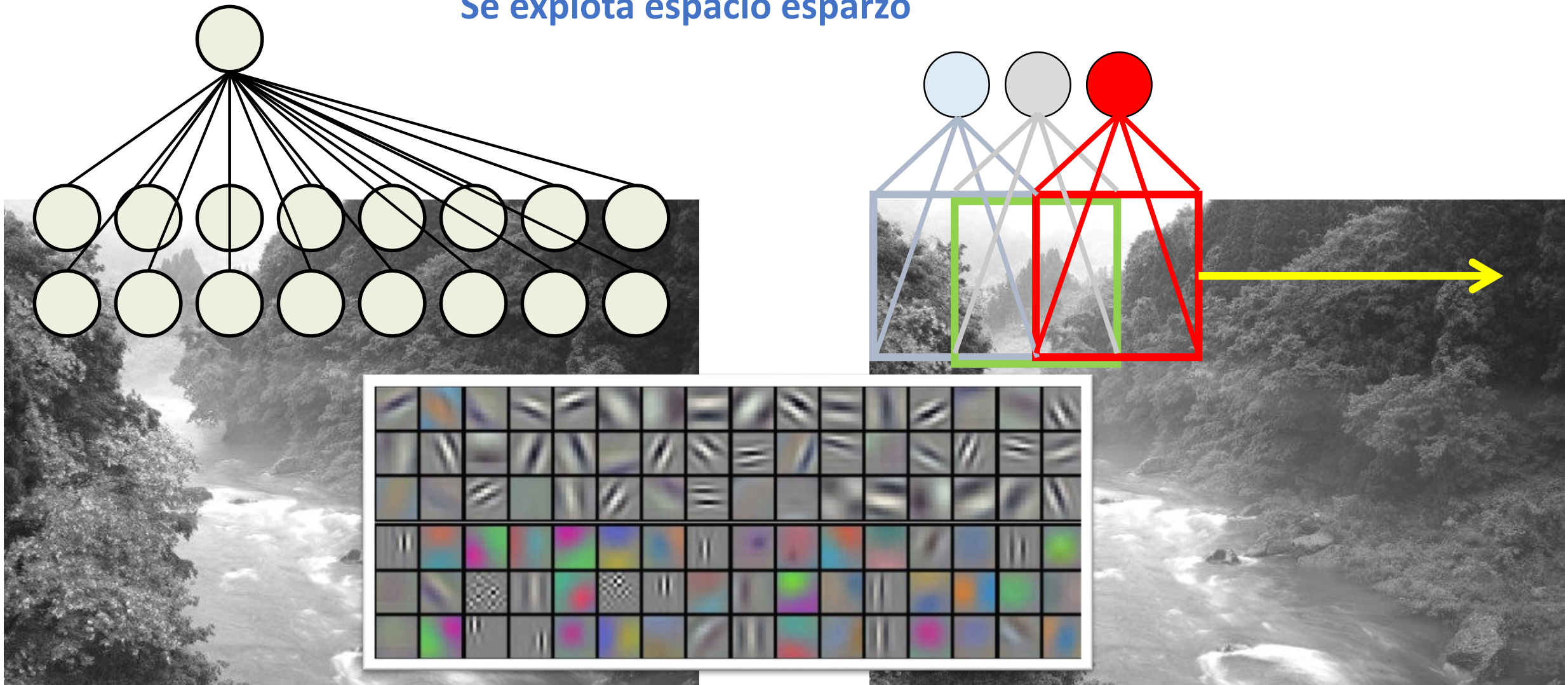
INSPIRADAS EN EL SISTEMA VISUAL HUMANO



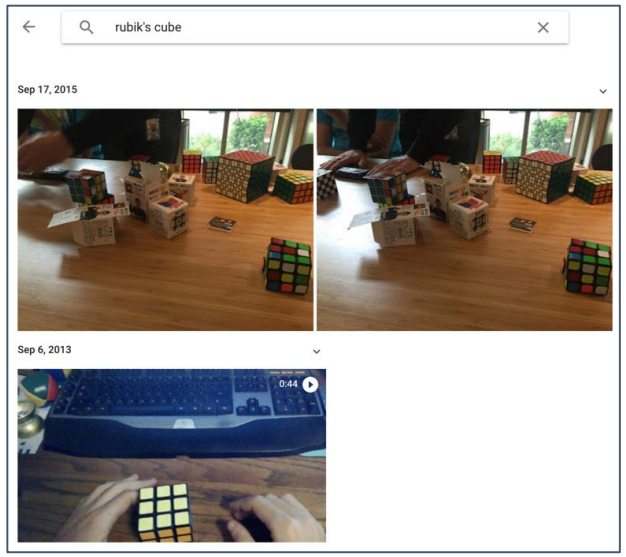
Hubel, D. H., & Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. The Journal of physiology, 148(3), 574.

REDES NEURONALES CONVOLUCIONALES (CNN)

Se explota espacio esparzo

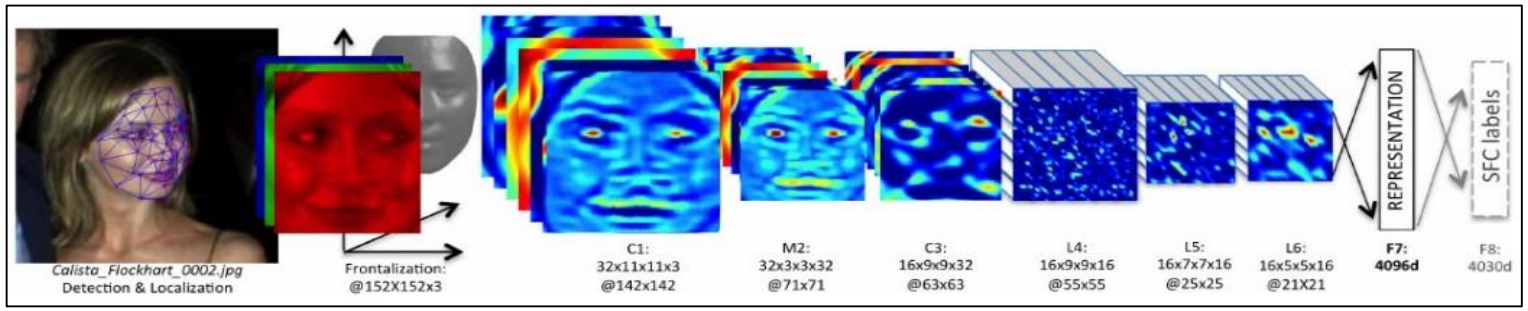


ACTUALMENTE CONVNETS EN TODAS PARTES…



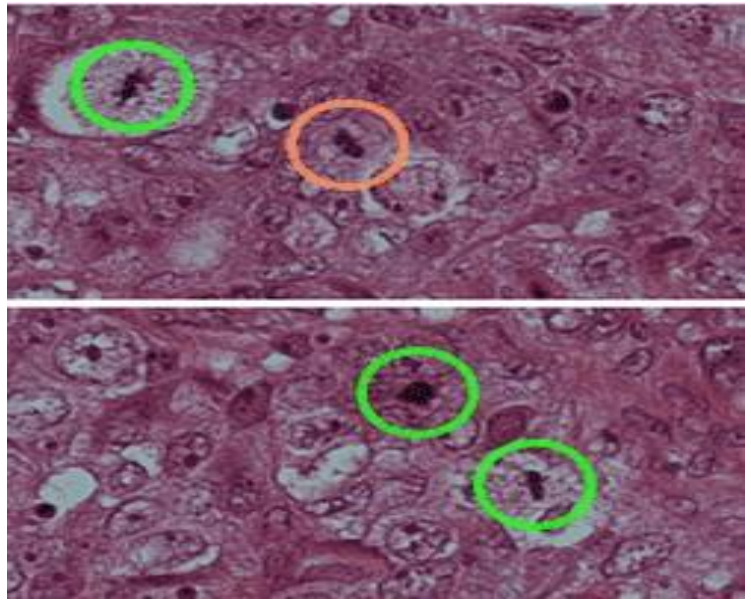
Google Photos search

Verificación Facial, Taigman et al. 2014 (FAIR)



[Goodfellow et al. 2014]

Vehículos autónomos



Ciresan et al. 2013

ACTUALMENTE CONVNETS EN TODAS PARTES…

Detección de Pose

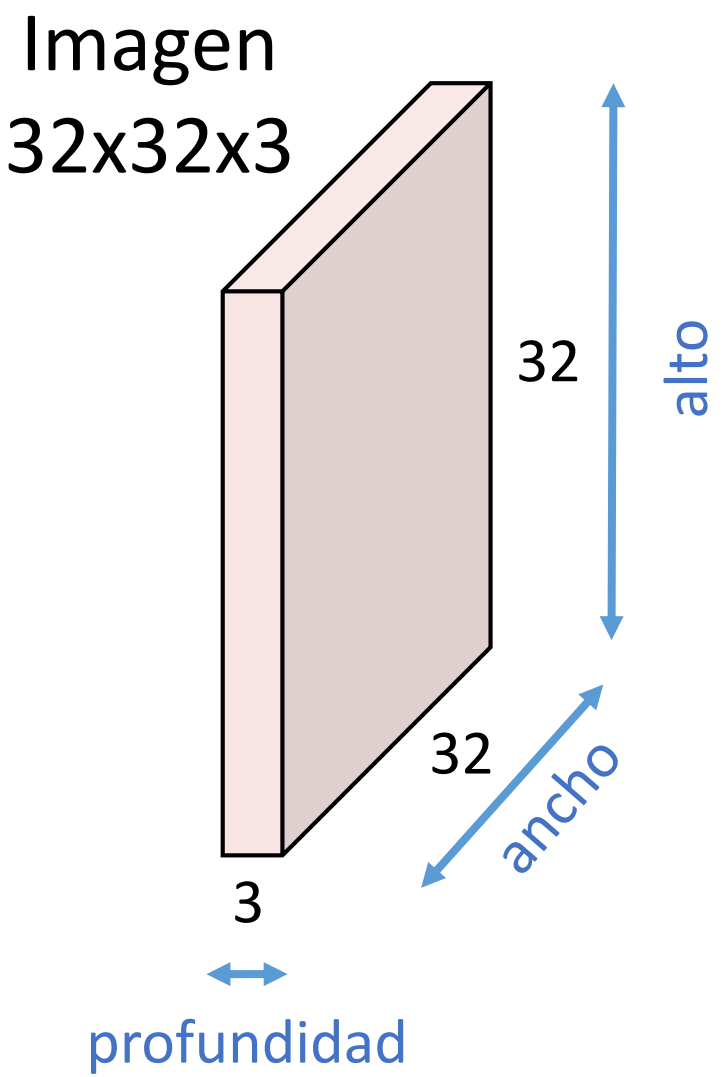
Toshev, A., & Szegedy, C. (2014). Deeppose: Human pose estimation via deep neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1653-1660)

ACTUALMENTE CONVNETS EN TODAS PARTES…

Subtitulado de imágenes

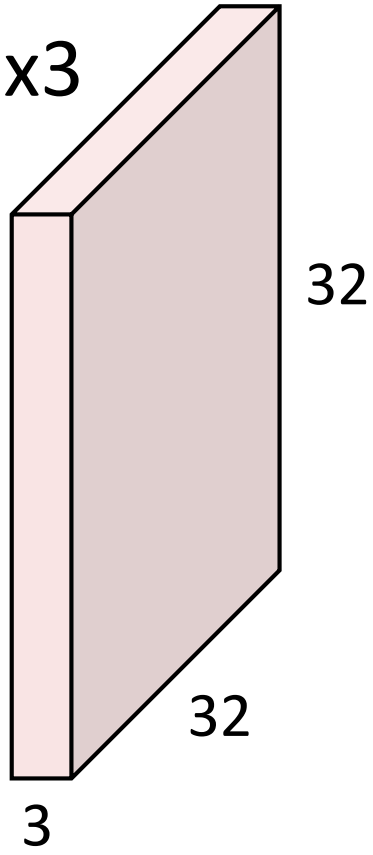
Karpathy, A., & Fei-Fei, L. (2015). Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3128-3137)

CAPA DE CONVOLUCIÓN

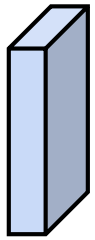


CAPA DE CONVOLUCIÓN

Imagen
32x32x3



Filtro
5x5x3

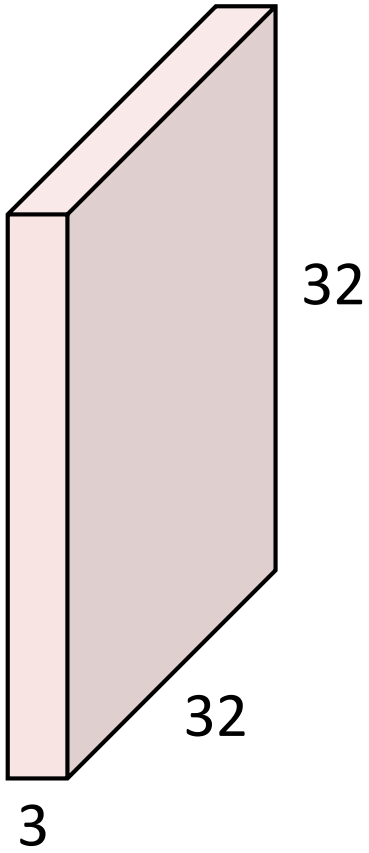


Convolucionar del filtro con la imagen: “recorrer espacialmente la imagen calculando producto punto con el filtro”

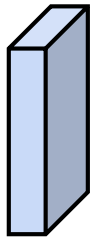
CAPA DE CONVOLUCIÓN

Filtros siempre extienden la profundidad del volumen de entrada

32x32x3

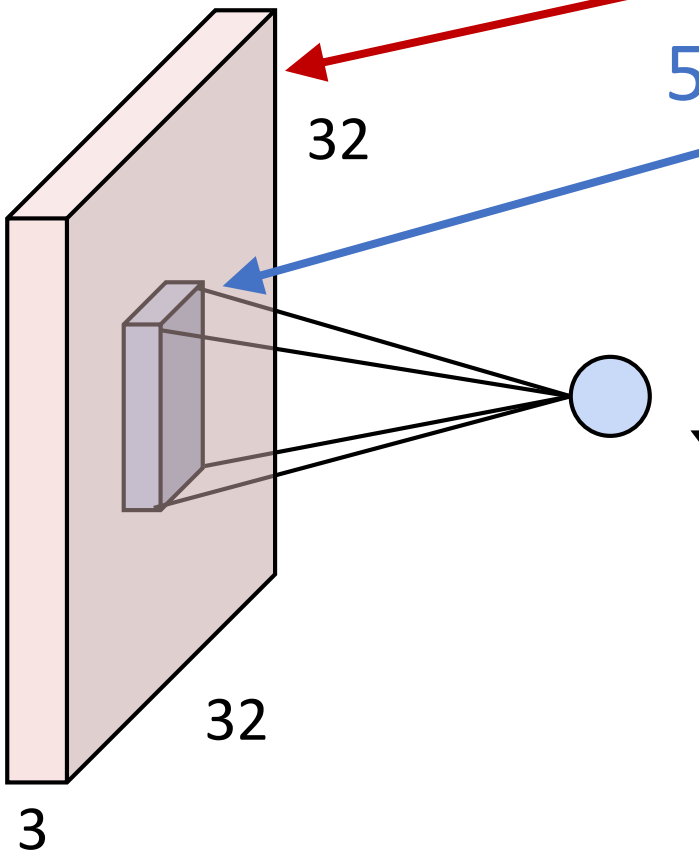


5x5x3



Convolucionar del filtro con la imagen: “recorrer espacialmente la imagen calculando producto punto con el filtro”

CAPA DE CONVOLUCIÓN



32x32x3 imagen x

5x5x3 filtro w

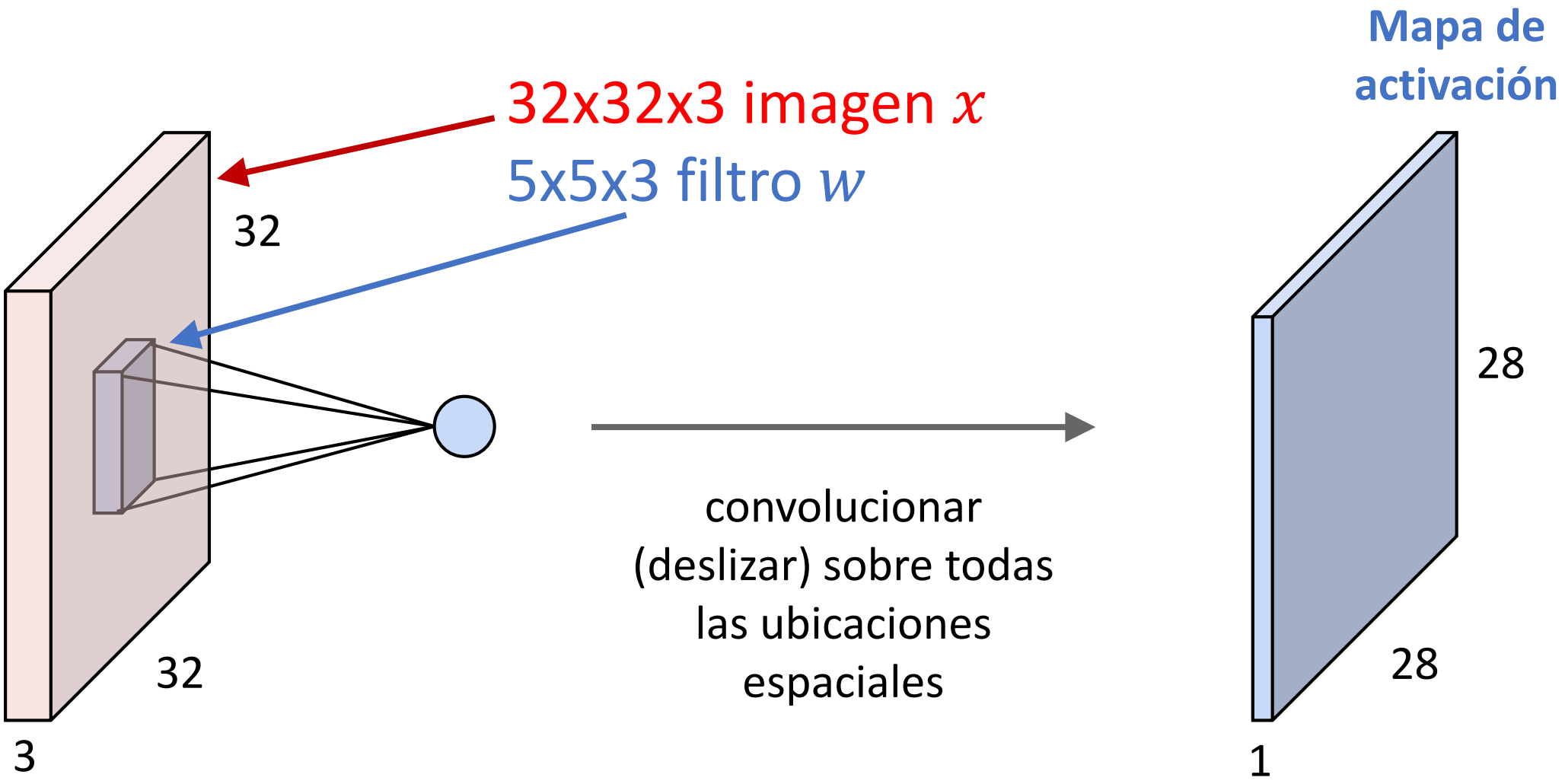
1 número:

Resultado de producto punto entre el filtro y una porción de imagen

(producto punto 5*5*3 = 75-dimensional + bias)

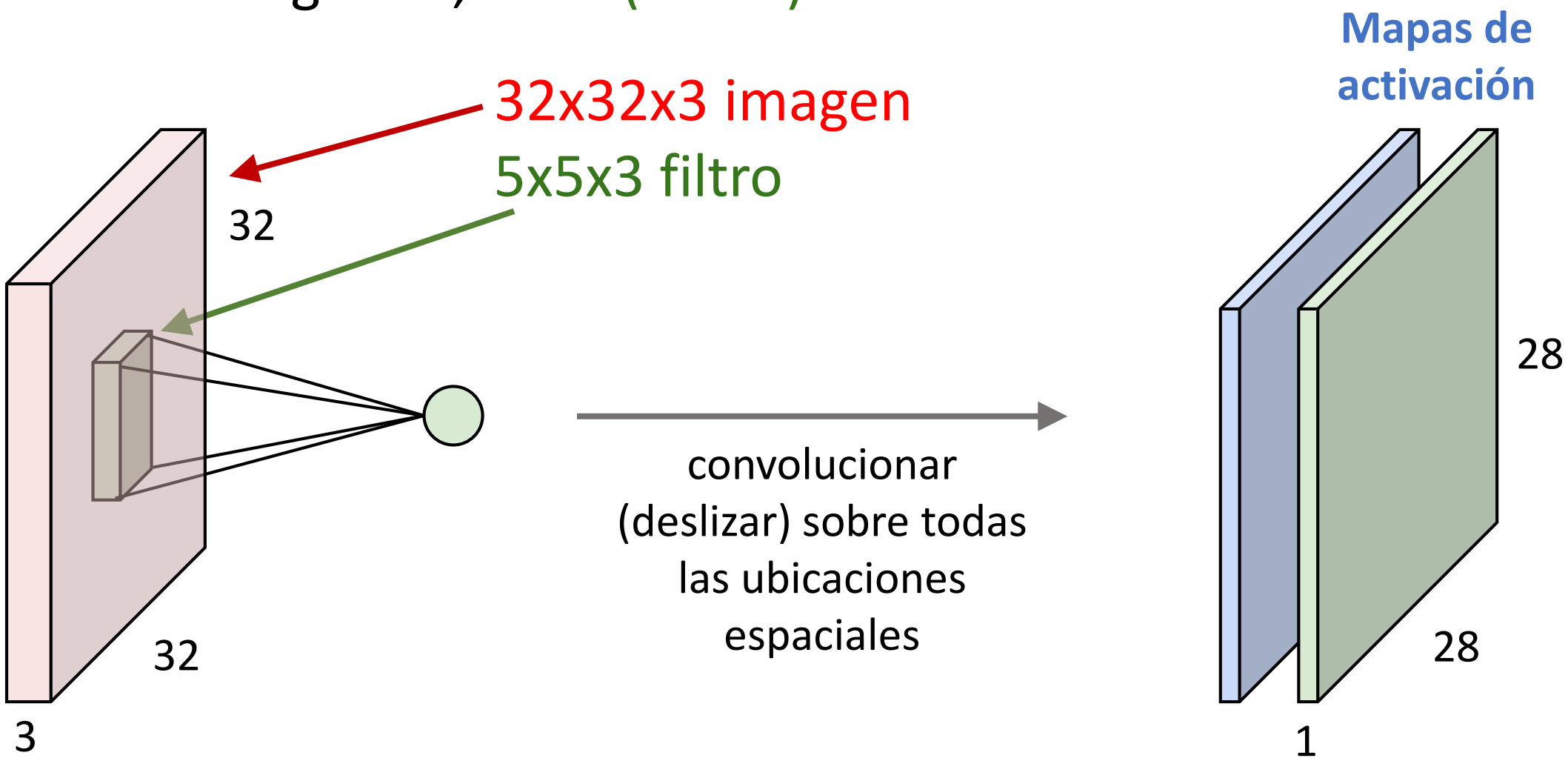
$$w^T x + b$$

CAPA DE CONVOLUCIÓN

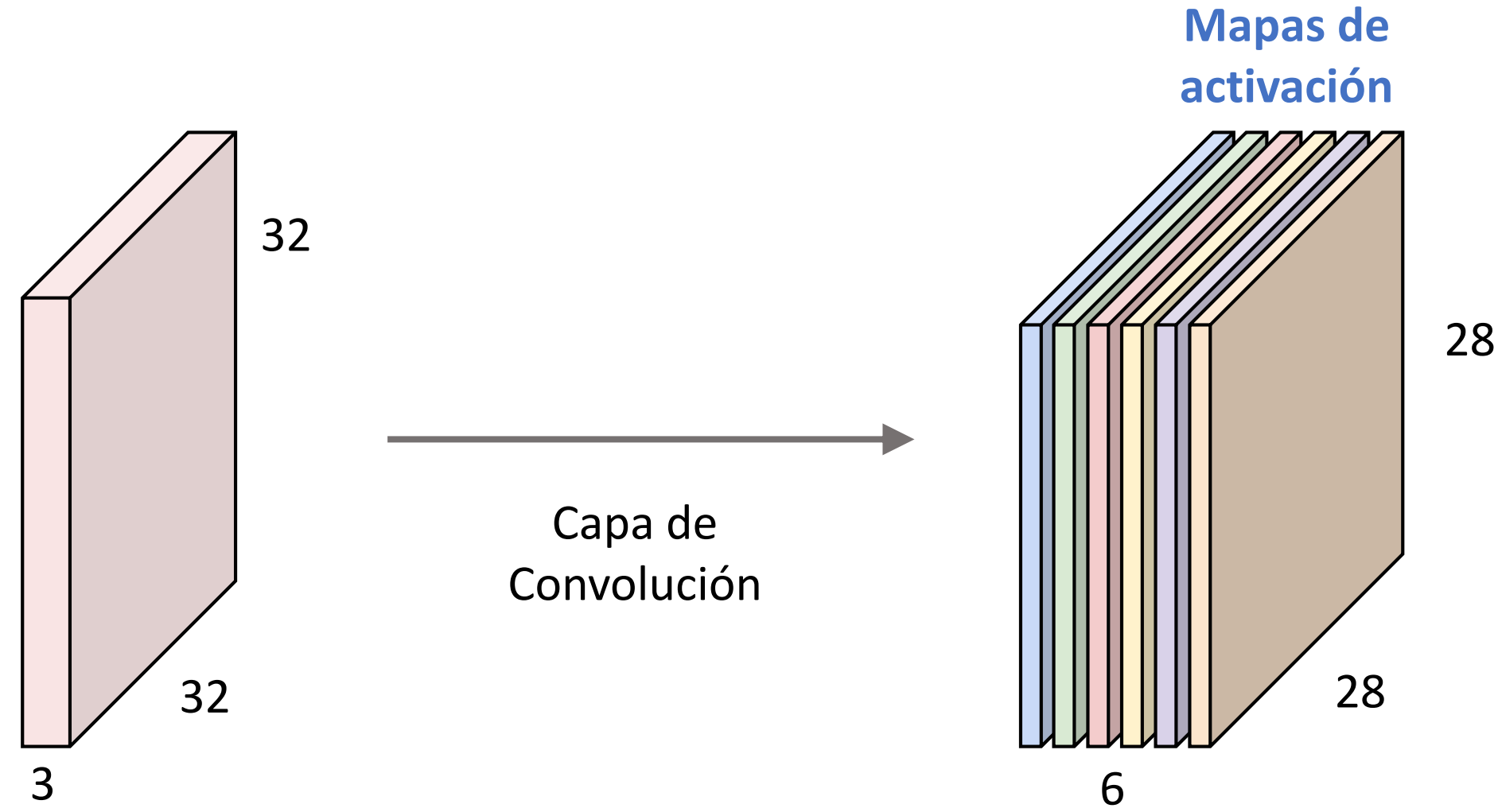


CAPA DE CONVOLUCIÓN

considere un segundo, filtro (verde)

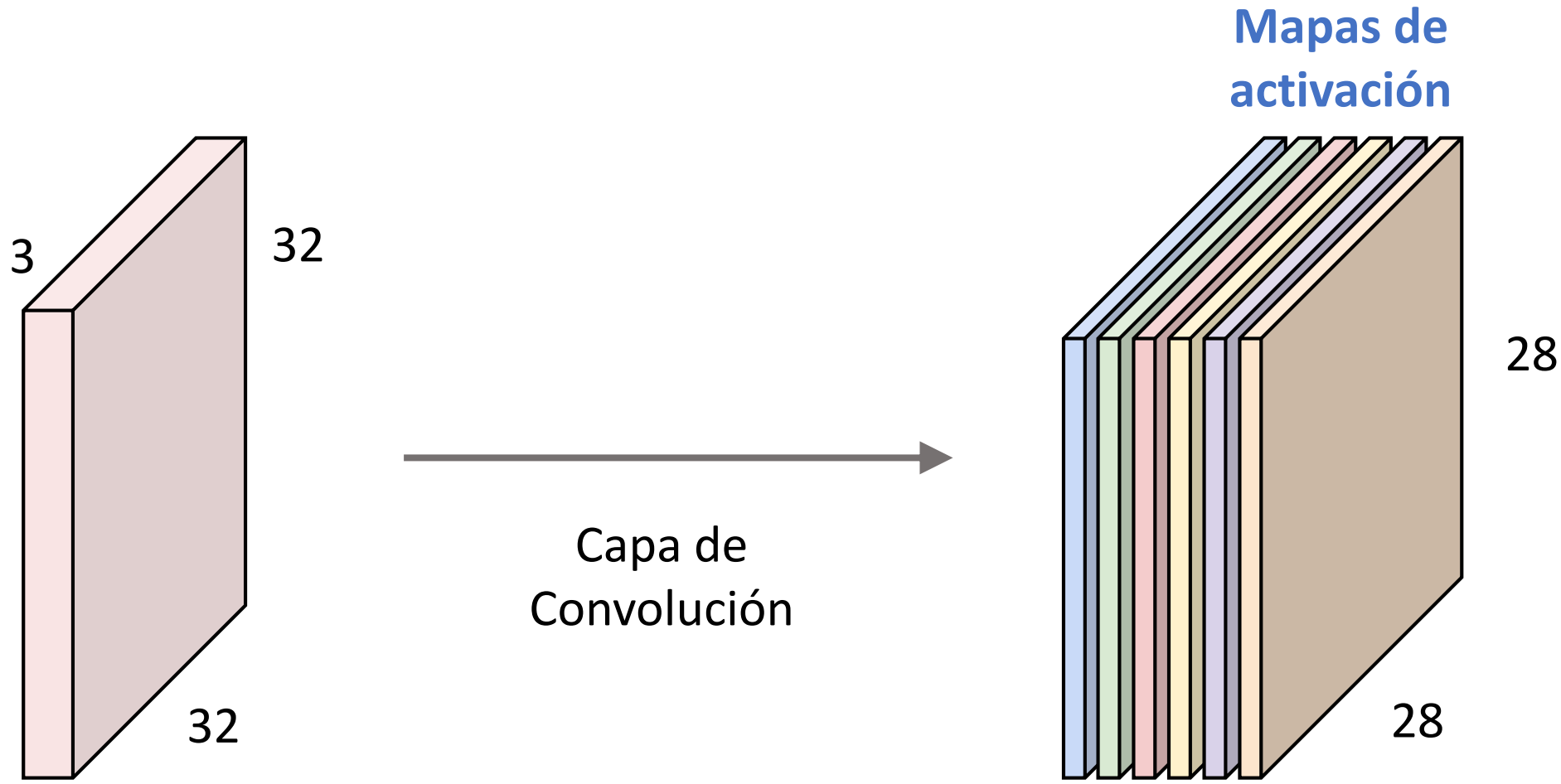


CAPA DE CONVOLUCIÓN



- Suponer 6 filtros de 5x5, se obtendrán 6 mapas de activación diferentes
- Los apilamos y se obtiene una "nueva imagen" de tamaño 28x28x6

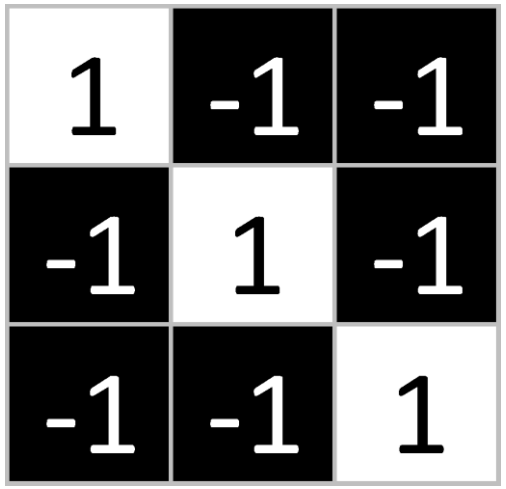
CAPA DE CONVOLUCIÓN

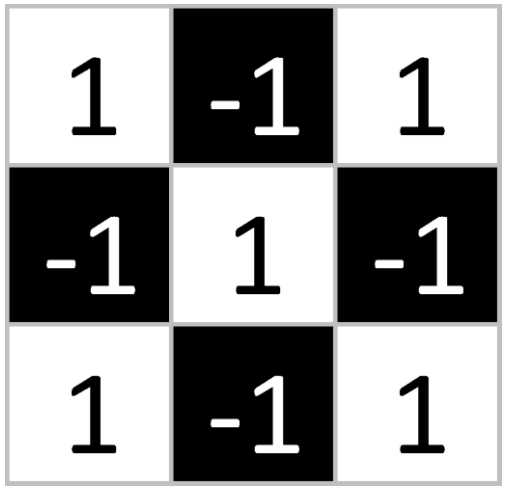


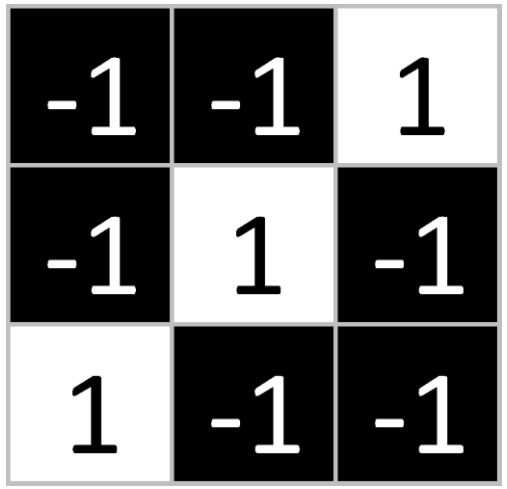
- Procesamos un volumen [32x32x3] en otro de [28x28x6]
- Si fuera completamente conectada, (32*32*3) ** (28*28*6) = 14,45M parámetros
- Enfoque convolutivo (5*5*3)*6 = **450 parámetros** y (5*5*3)*(28*28*6) = **~350K multip.**

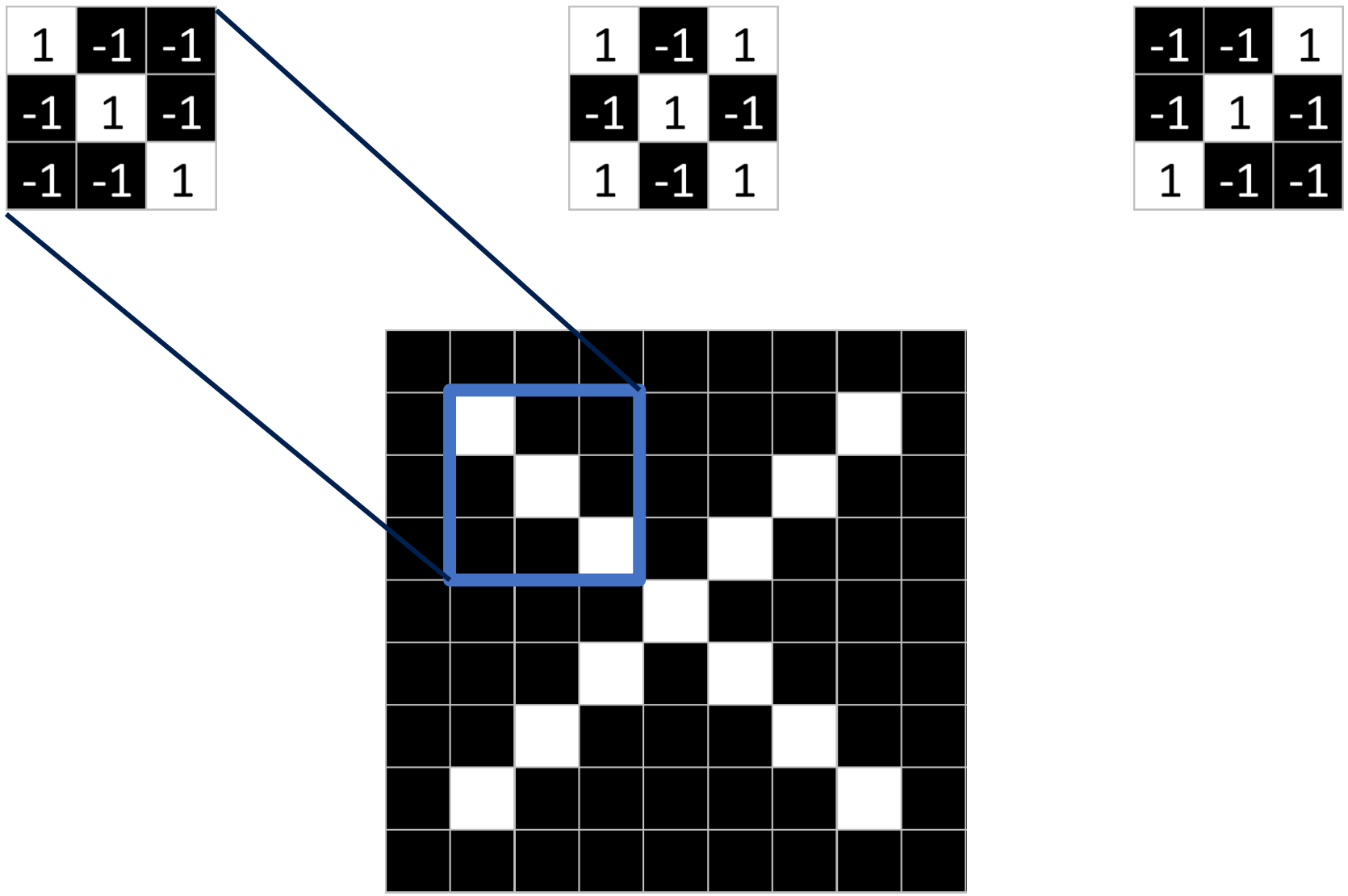
MATRICES DE CARACTERÍSTICAS

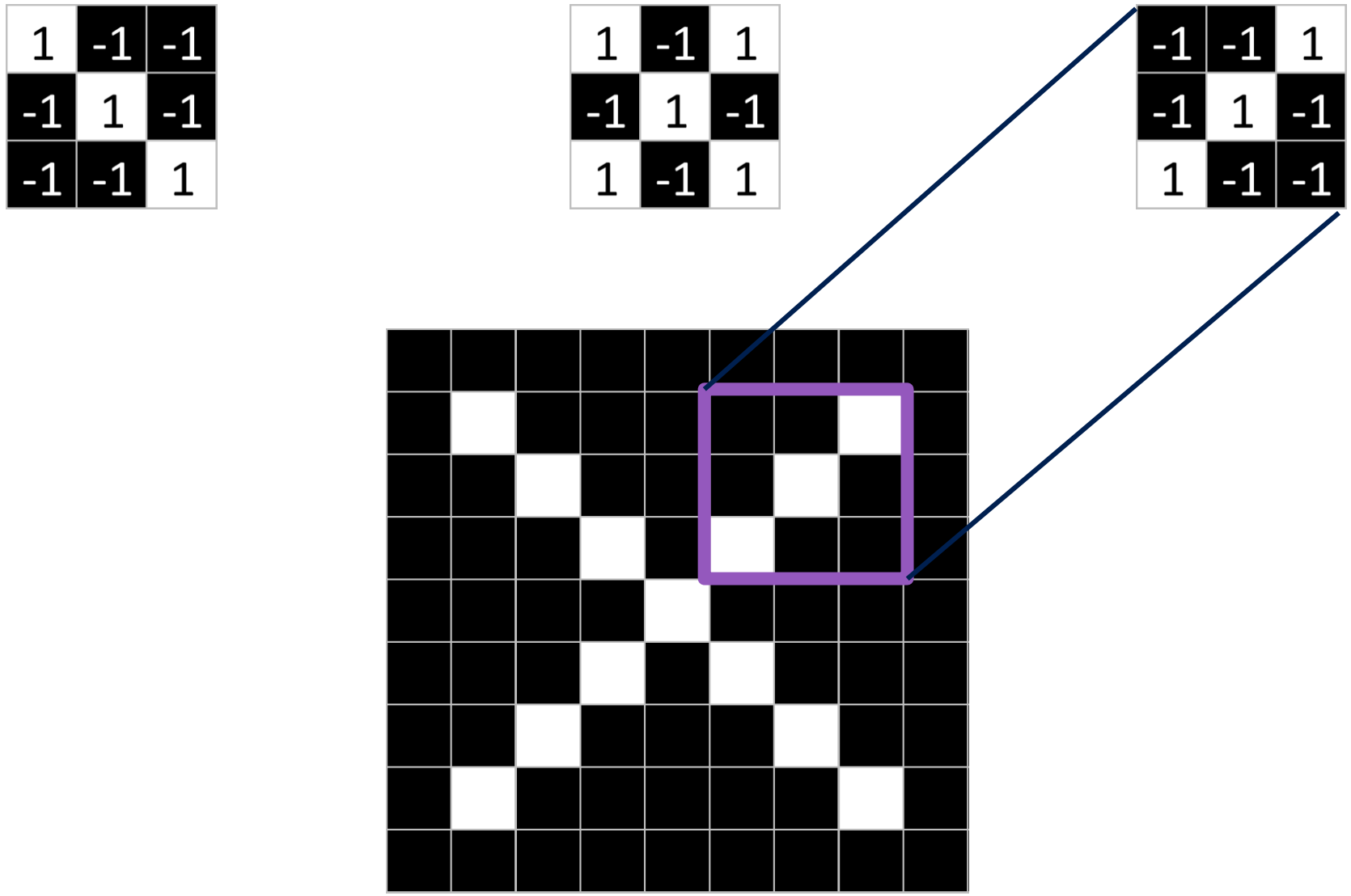
Patrones que se van a tratar de determinar si están presentes en la imagen y en qué parte





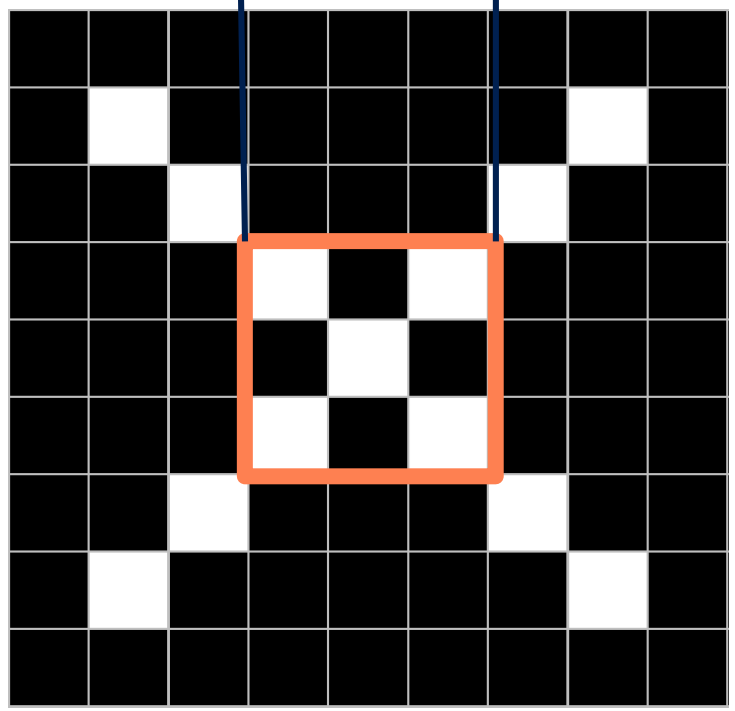



1 -1 -1
-1 1 -1
-1 -1 1
1 -1 1
-1 1 -1
1 -1 1
-1 -1 1
-1 1 -1
1 -1 -1

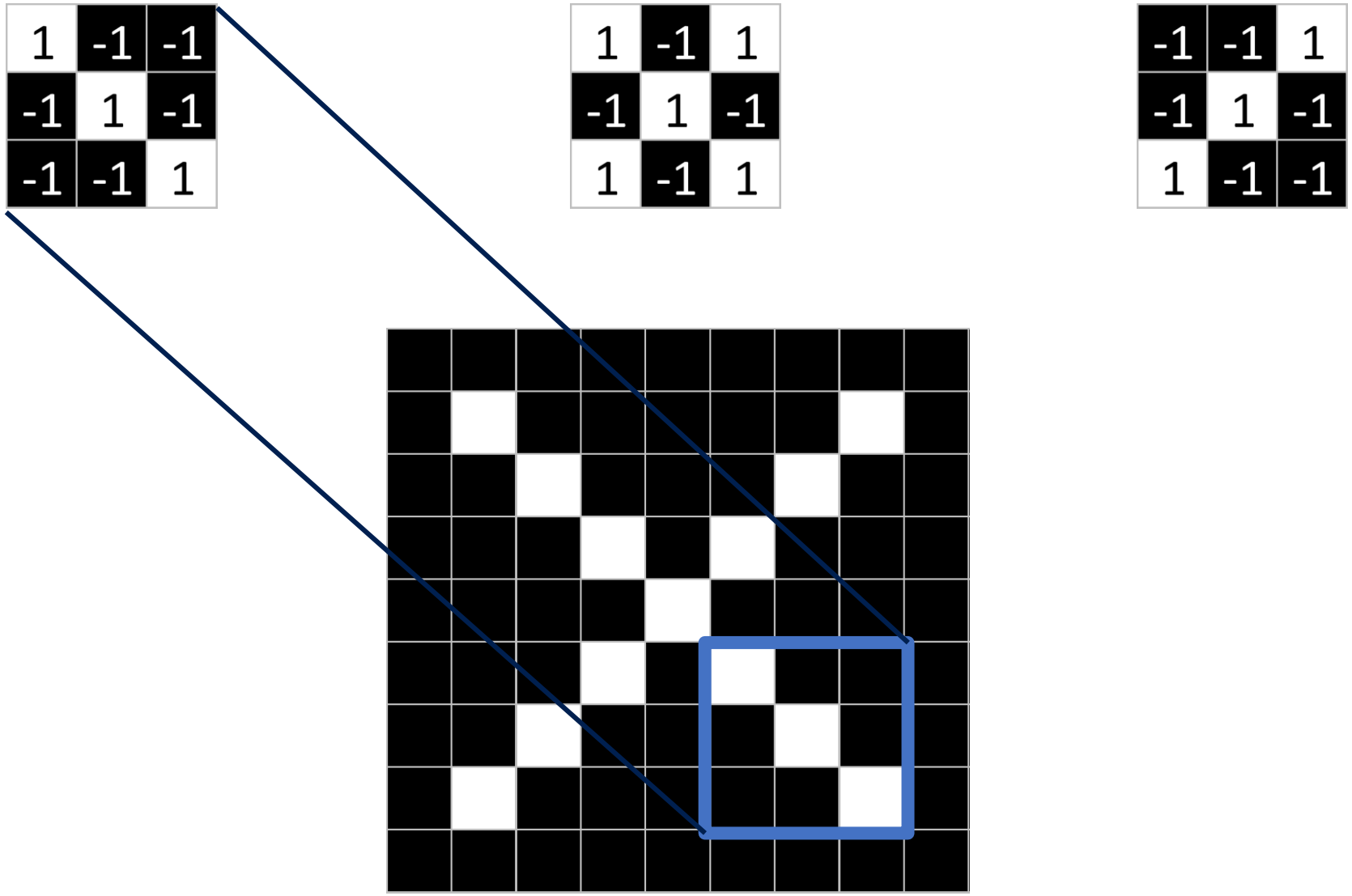

1 -1 -1
-1 1 -1
-1 -1 1
1 -1 1
-1 1 -1
1 -1 1
-1 -1 1
-1 1 -1
1 -1 -1

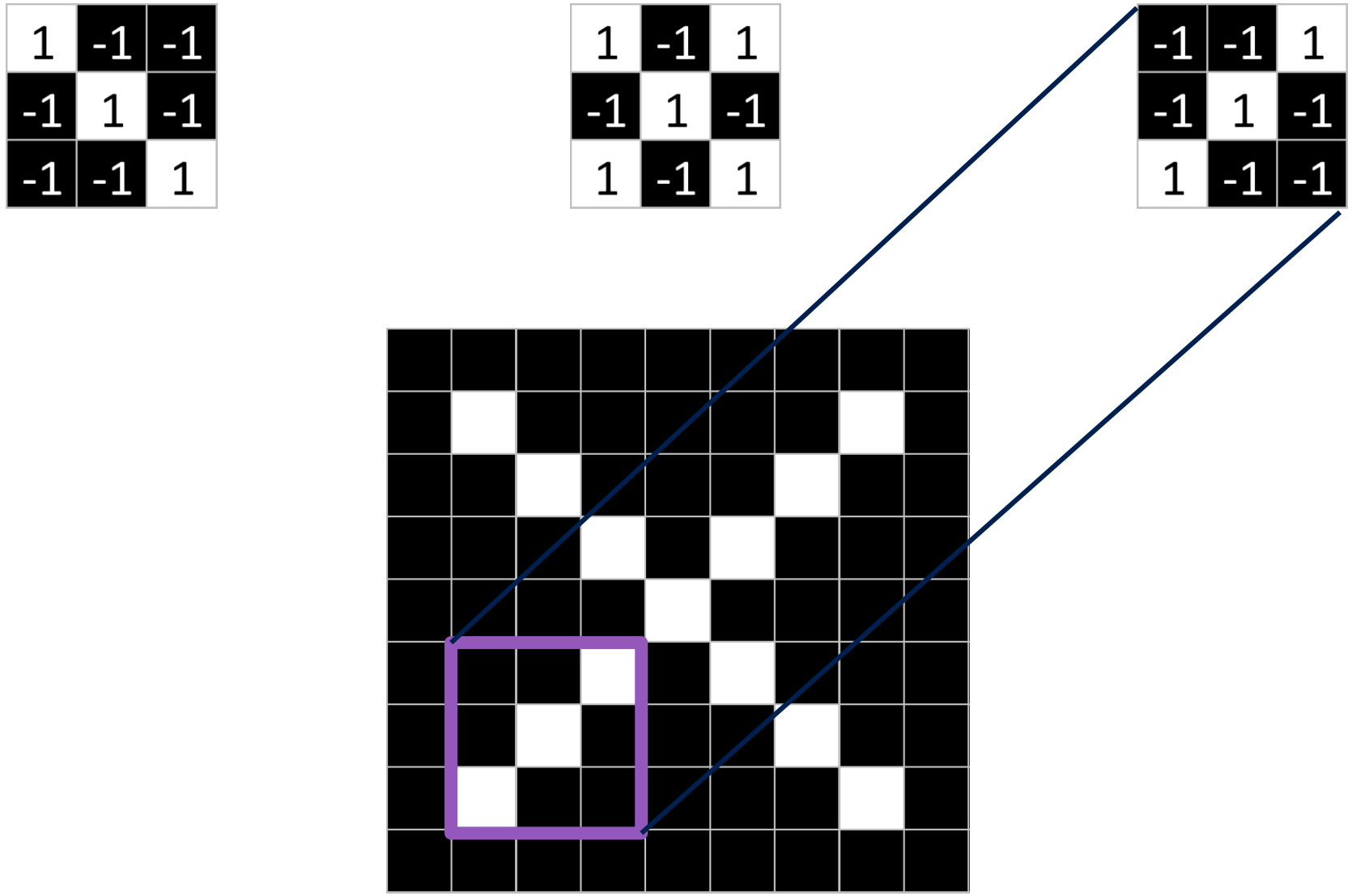
1	-1	-1
-1	1	-1
-1	-1	1

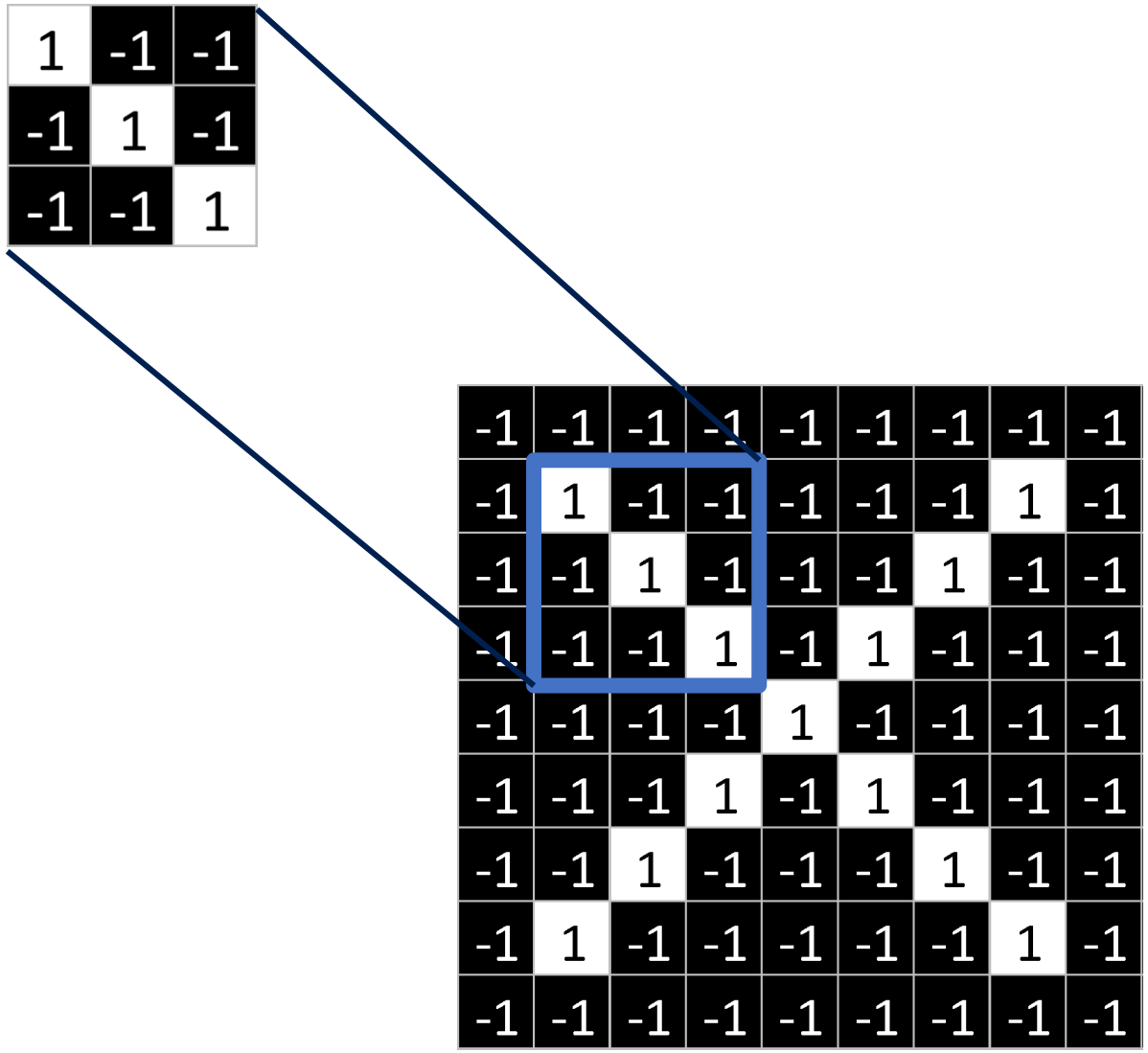
1	-1	1
-1	1	-1
1	-1	1



-1	-1	1
-1	1	-1
1	-1	-1


1 -1 -1
-1 1 -1
-1 -1 1
1 -1 1
-1 1 -1
1 -1 1
-1 -1 1
-1 1 -1
1 -1 -1

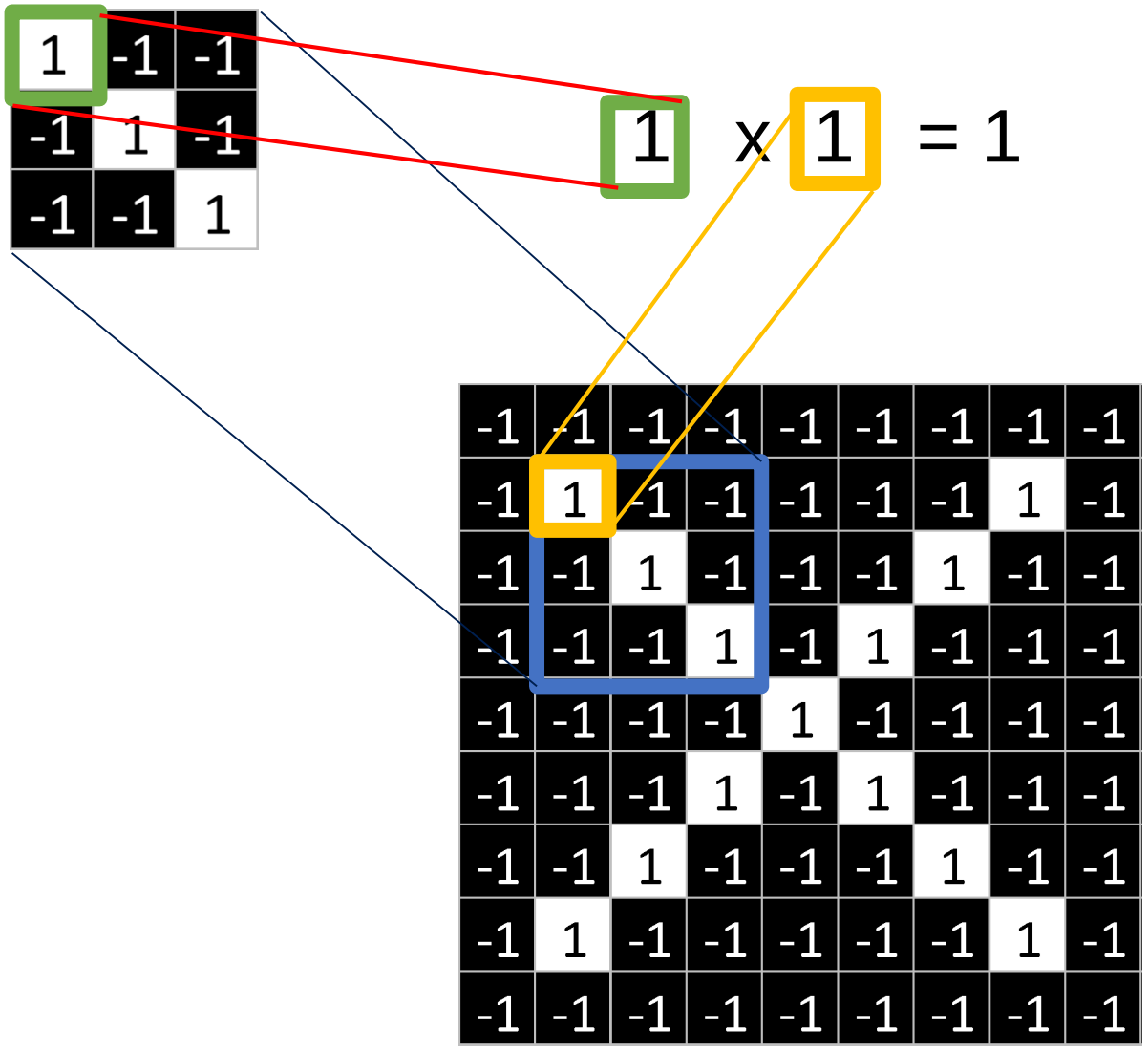

1 -1 -1
-1 1 -1
-1 -1 1
1 -1 1
-1 1 -1
1 -1 1
-1 -1 1
-1 1 -1
1 -1 -1


1 -1 -1
-1 1 -1
-1 -1 1
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1

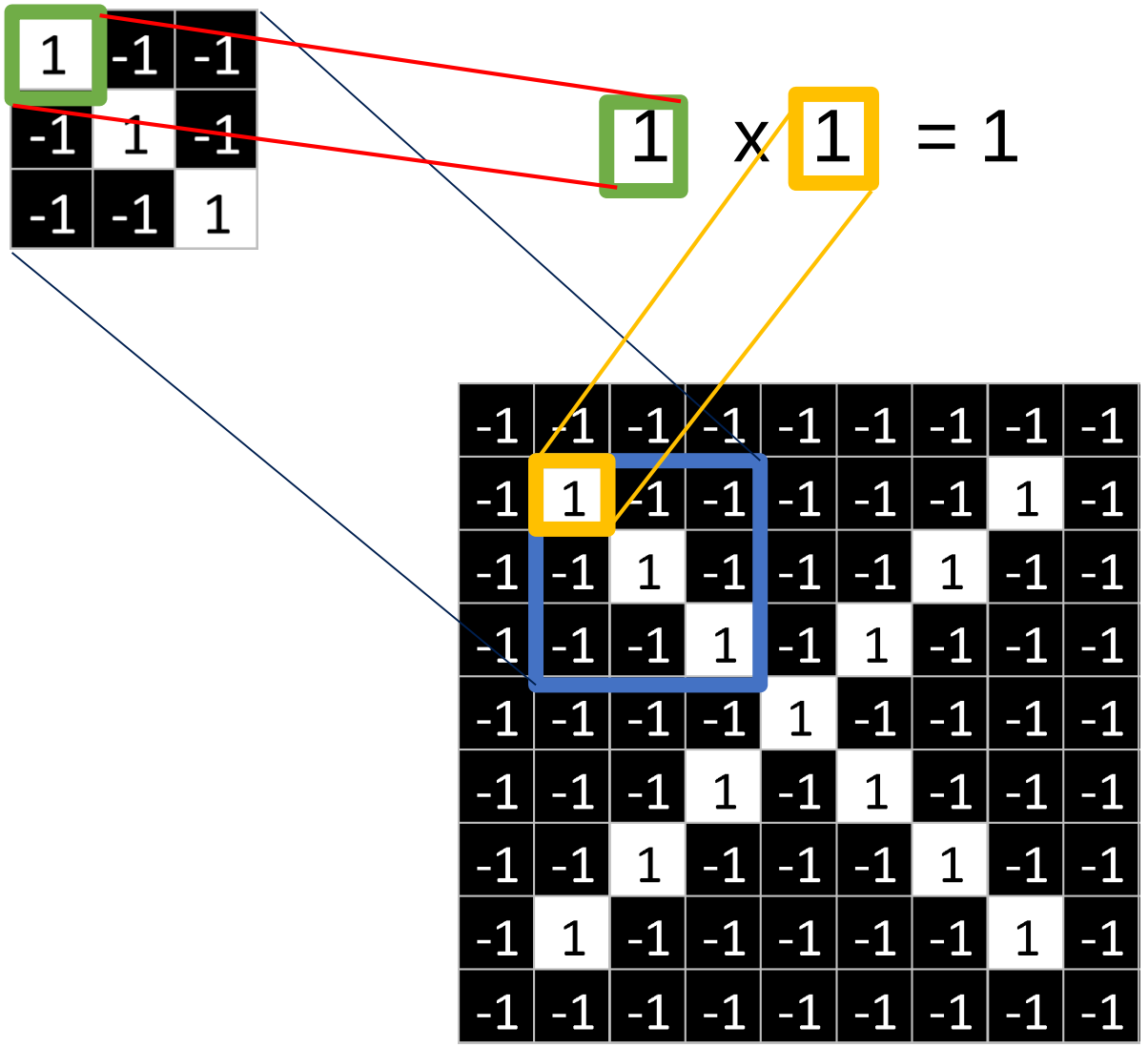
OPERACIÓN DE FILTRADO

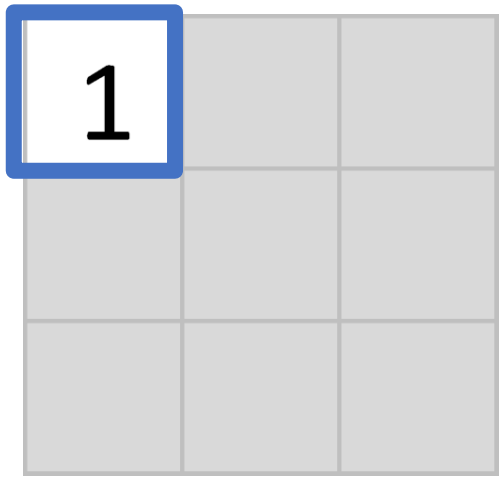
- Alinear la matriz de características con el parche de la imagen
- Multiplicar píxel a pixel ambas matrices
- Sumar los productos
- Dividir por el número total de píxeles de la matriz de características

FILTRADO

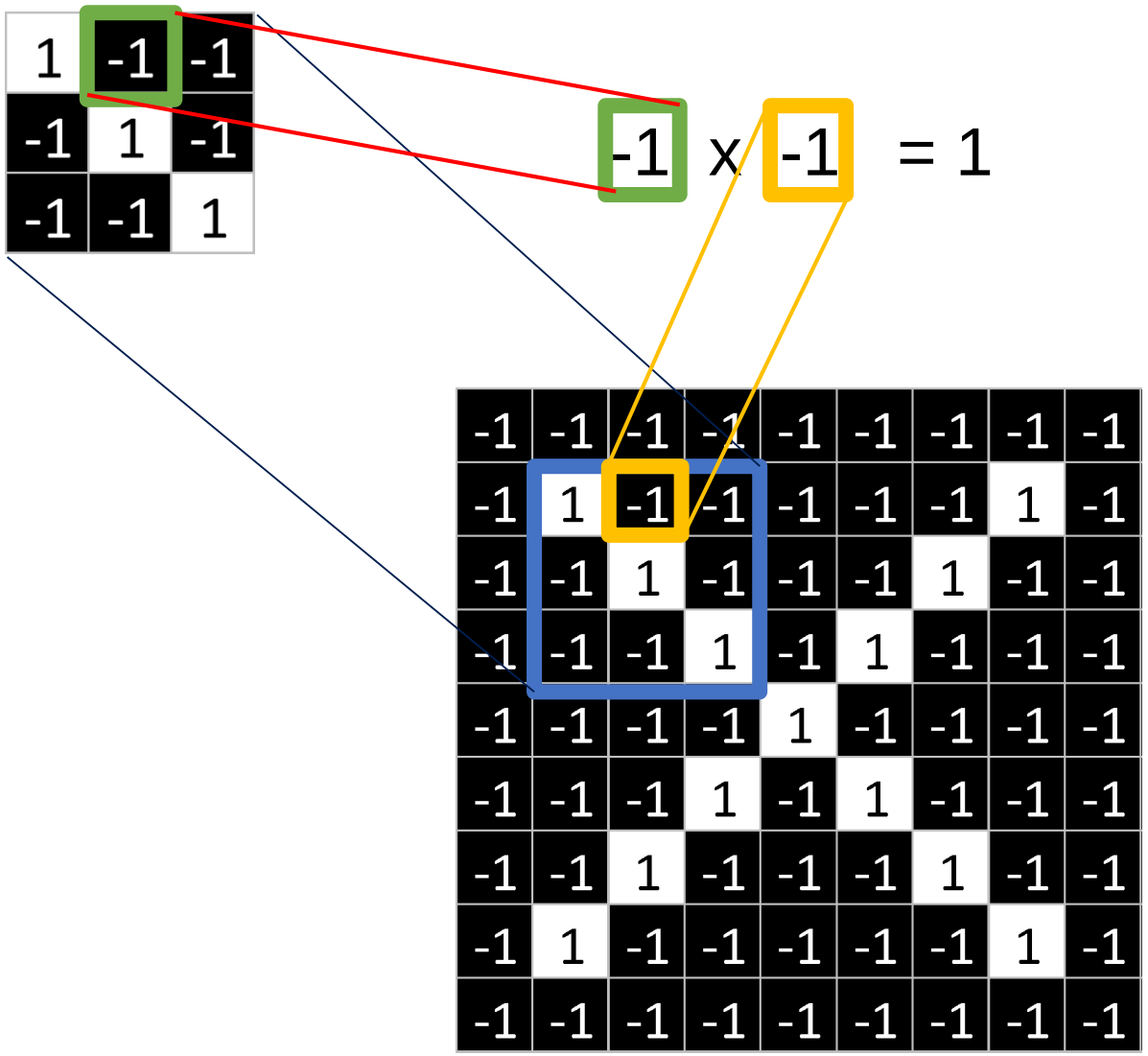

1 -1 -1
-1 1 -1
-1 -1 1
1 x 1 = 1
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1

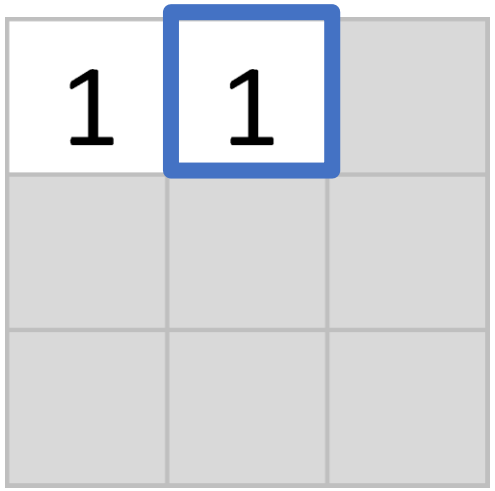
FILTRADO


1 -1 -1
-1 1 -1
-1 -1 1
1 x 1 = 1
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1

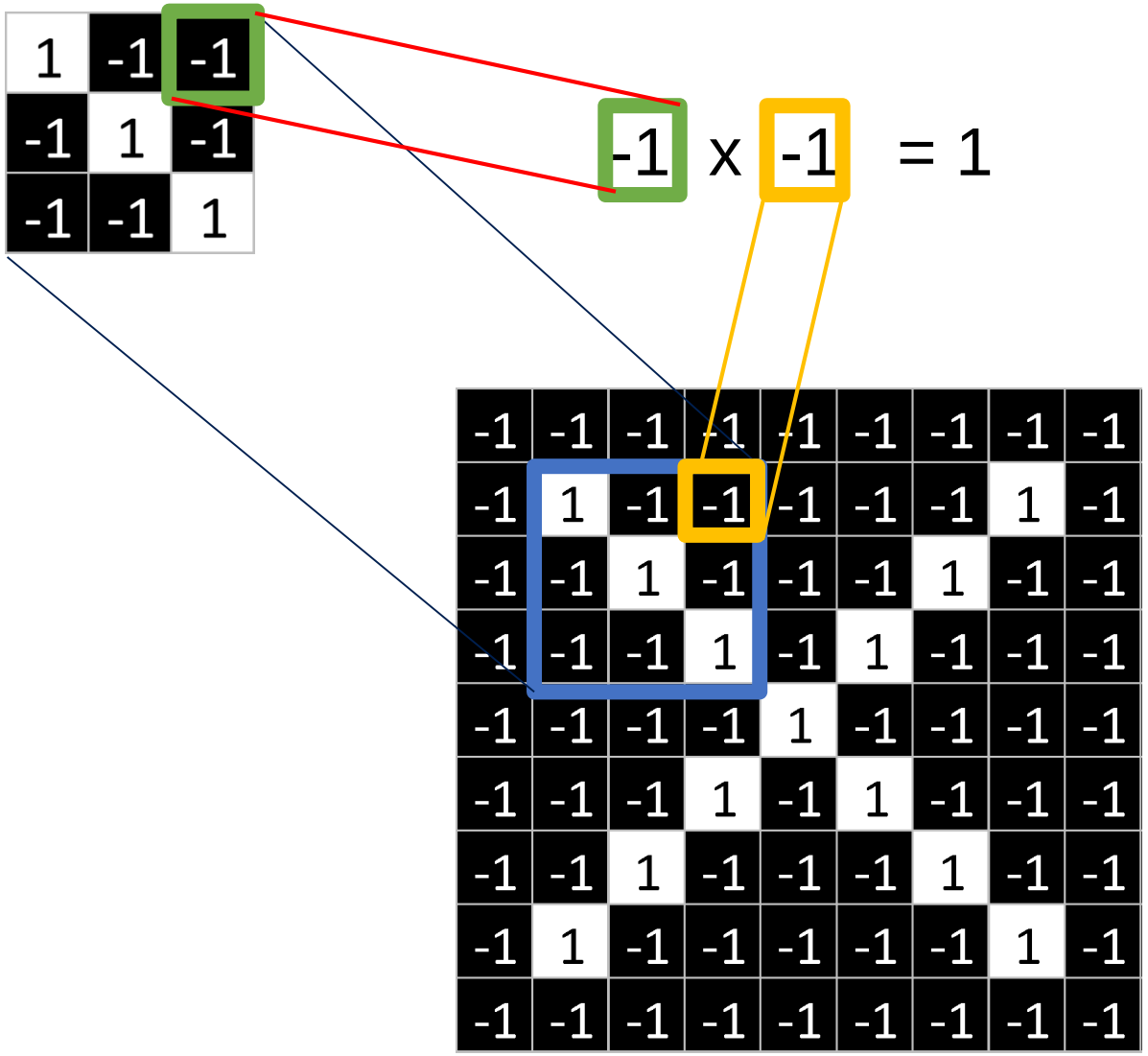

1

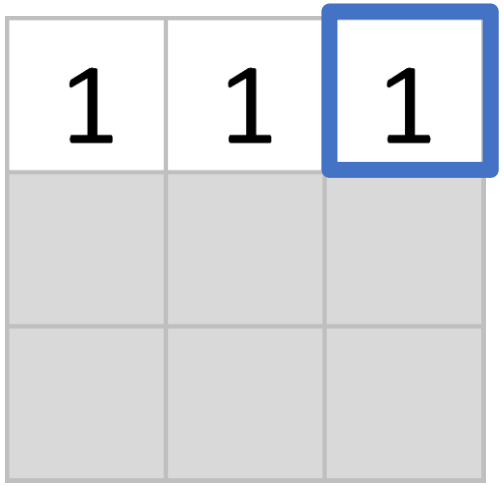
FILTRADO


1 -1 -1
-1 1 -1
-1 -1 1
-1 x -1 = 1
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1

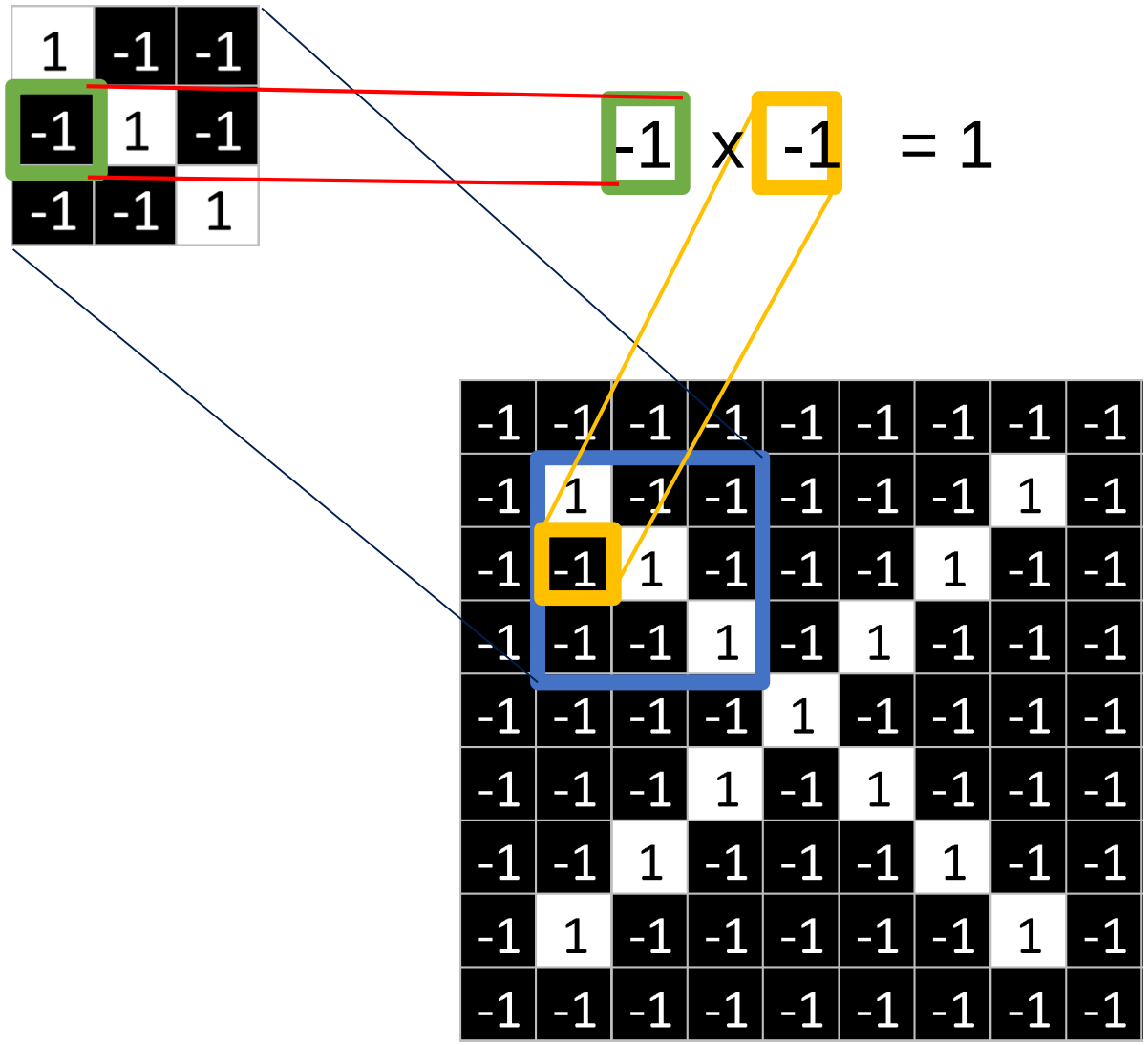

1 1

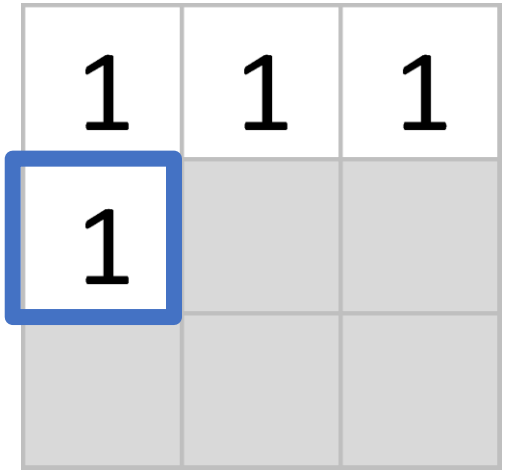
FILTRADO


1 -1 -1
-1 1 -1
-1 -1 1
-1 x -1 = 1
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1

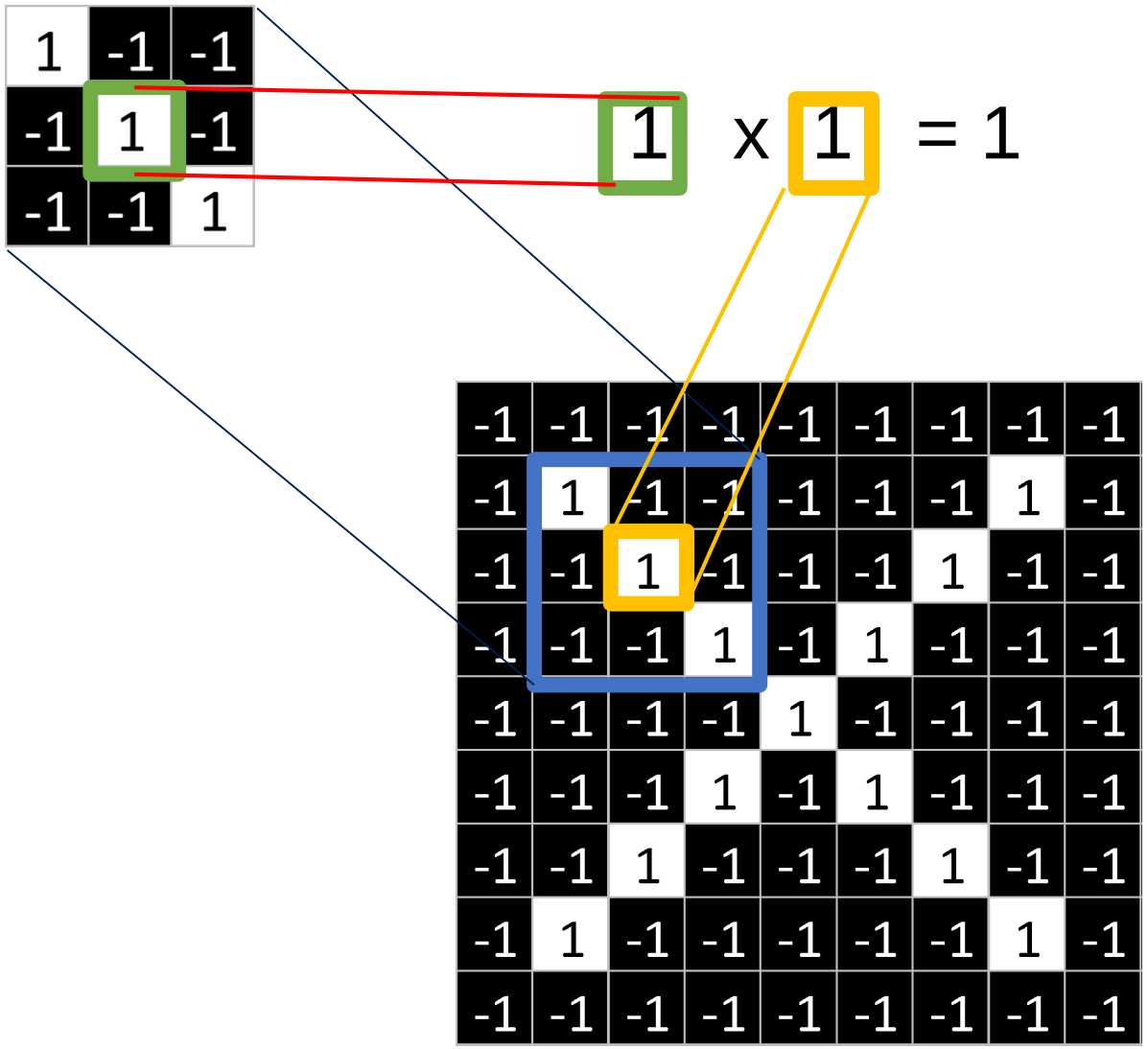

1 1 1

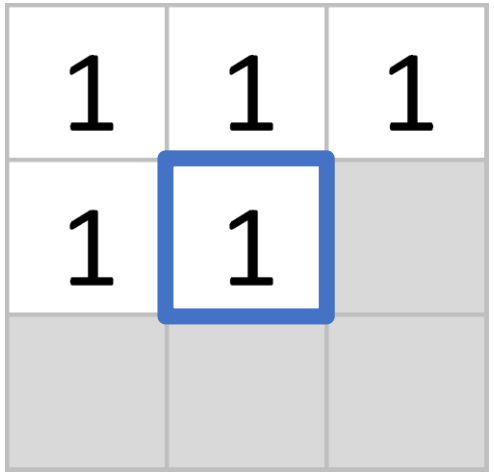
FILTRADO


1 -1 -1
-1 1 -1
-1 -1 1
-1 x -1 = 1
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1

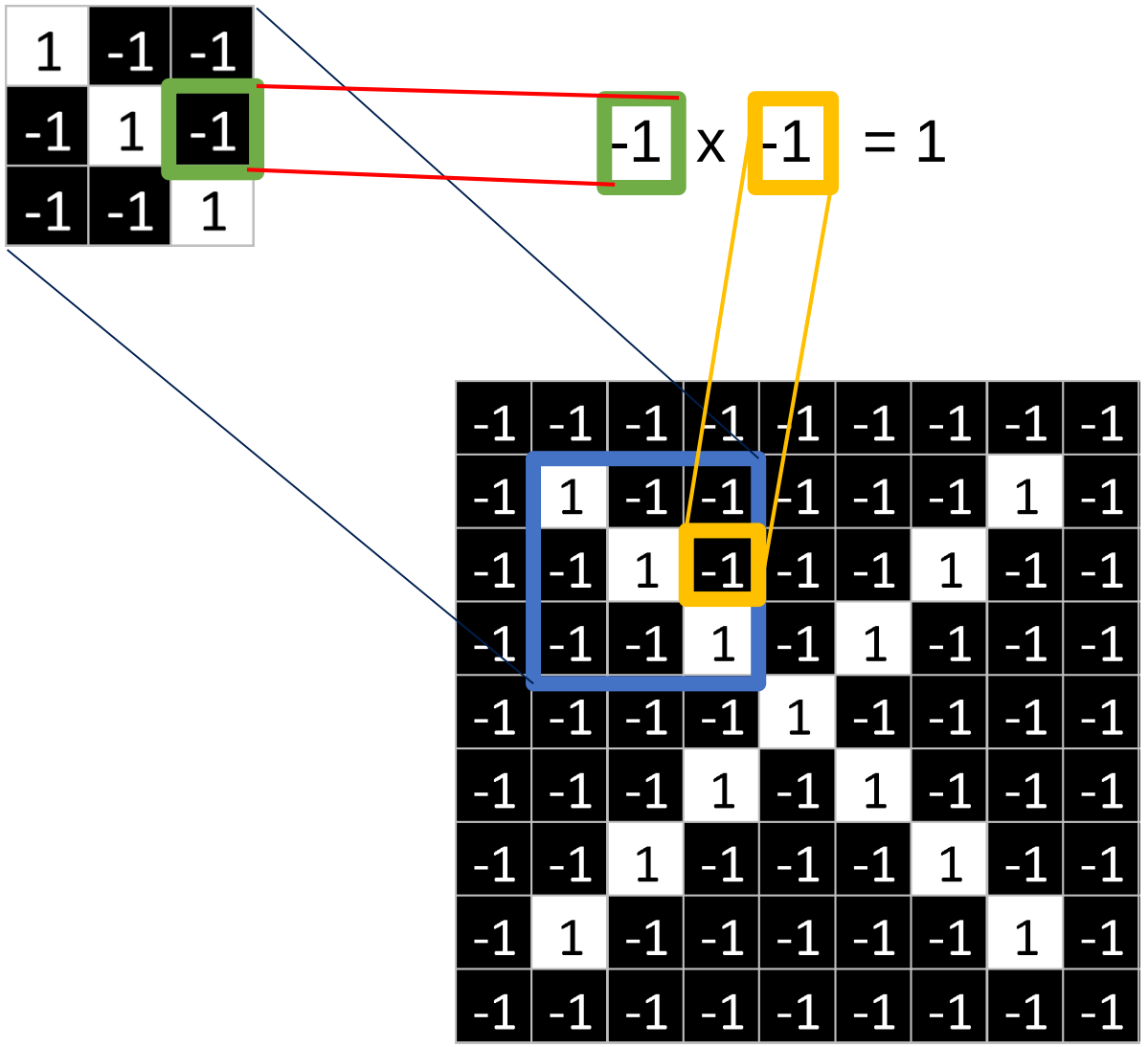

1 1 1
1

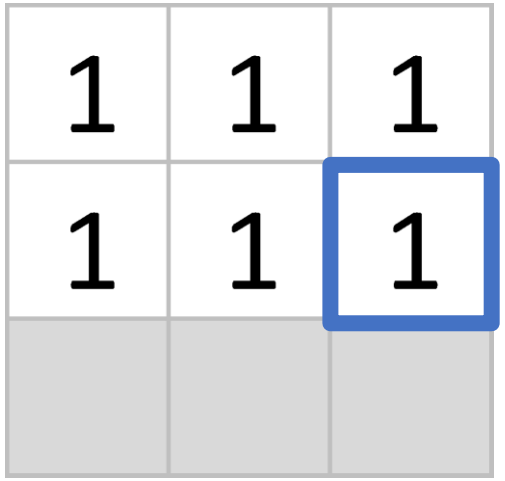
FILTRADO


1 -1 -1
-1 1 -1
-1 -1 1
1 x 1 = 1
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1

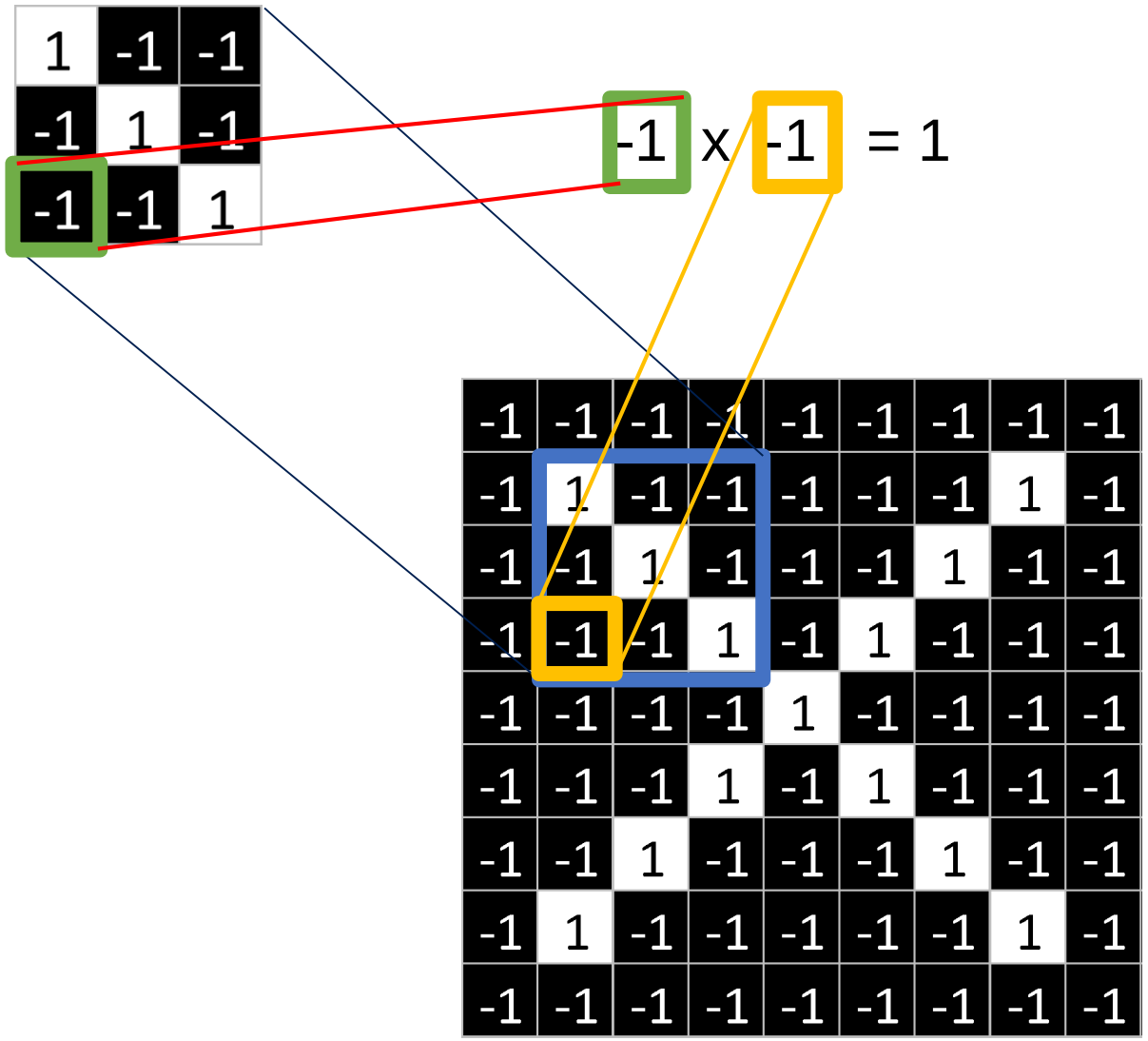

1 1 1
1 1

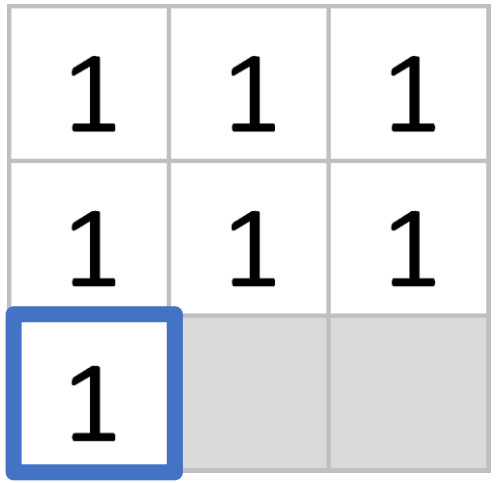
FILTRADO


1 -1 -1
-1 1 -1
-1 -1 1
-1 x -1 = 1
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1

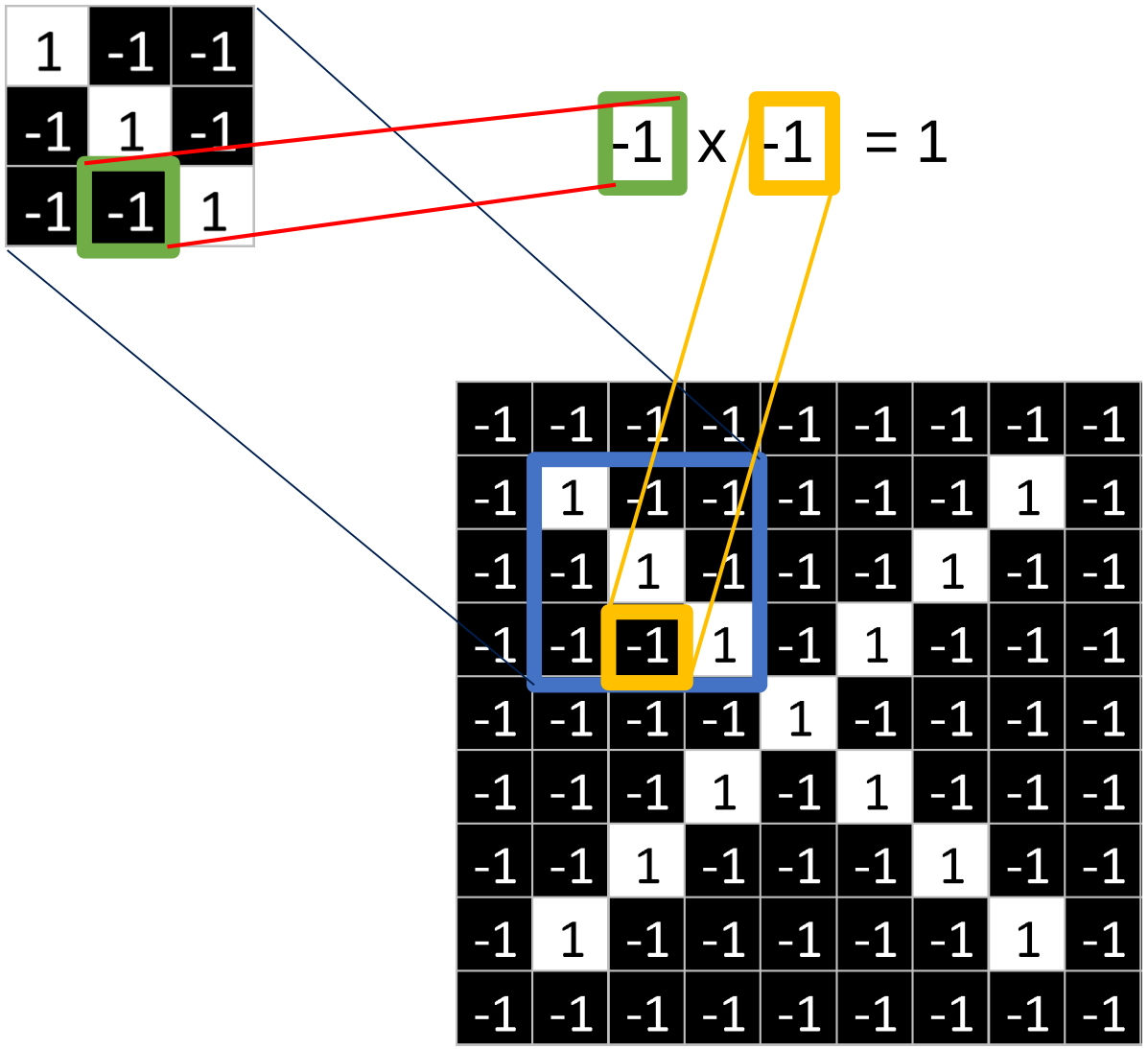

1 1 1
1 1 1

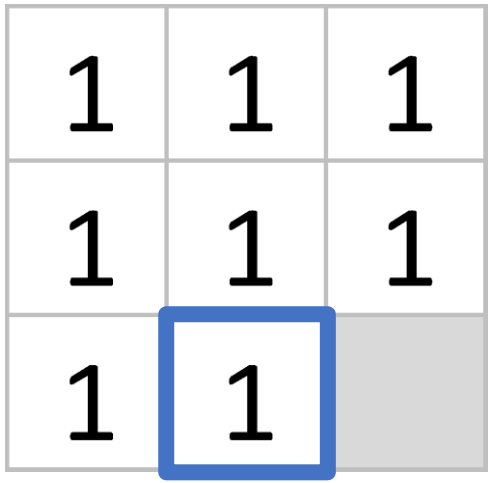
FILTRADO


1 -1 -1
-1 1 -1
-1 -1 1
-1 x -1 = 1
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1

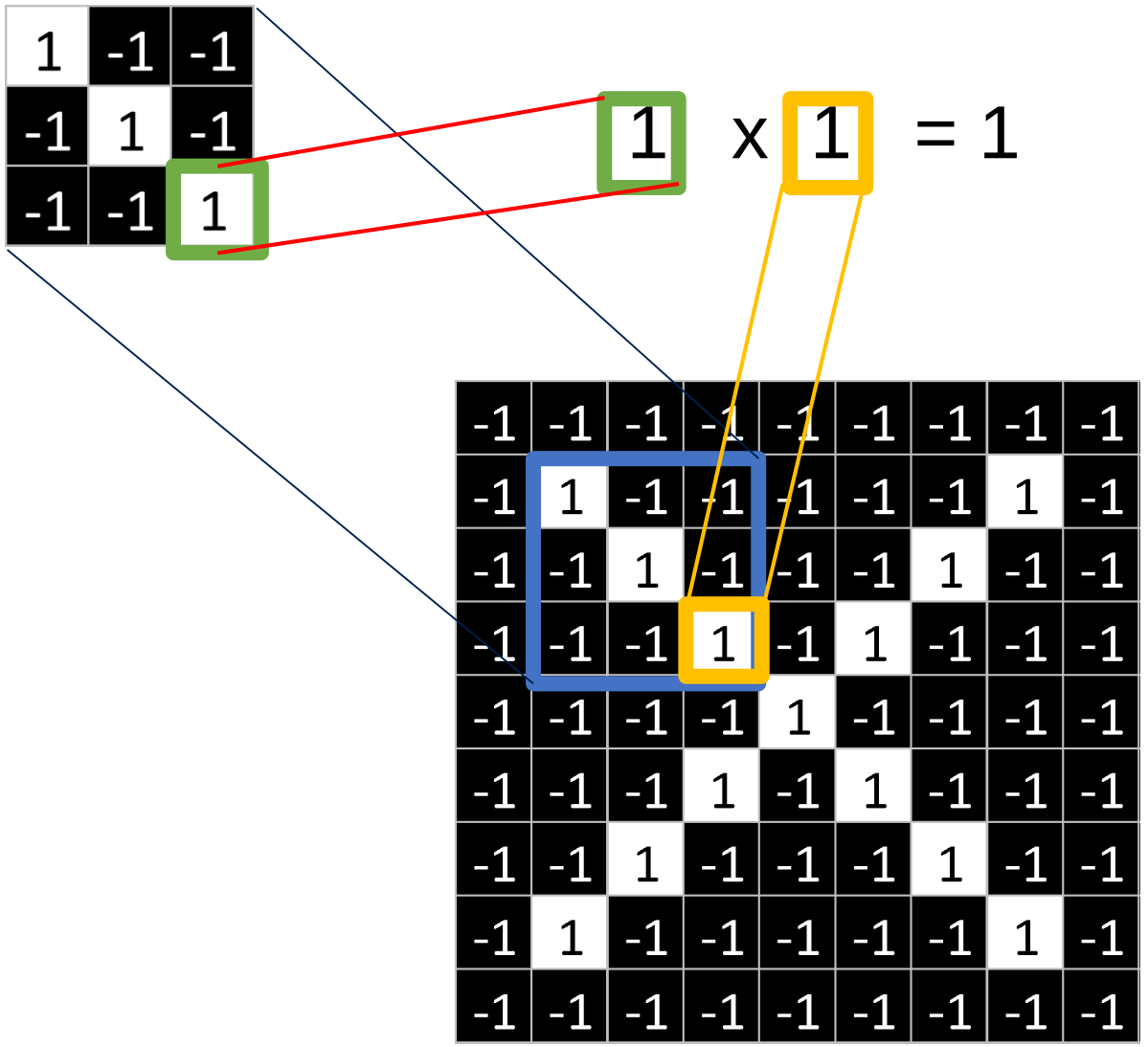

1 1 1
1 1 1
1

FILTRADO


1 -1 -1
-1 1 -1
-1 -1 1
-1 x -1 = 1
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1

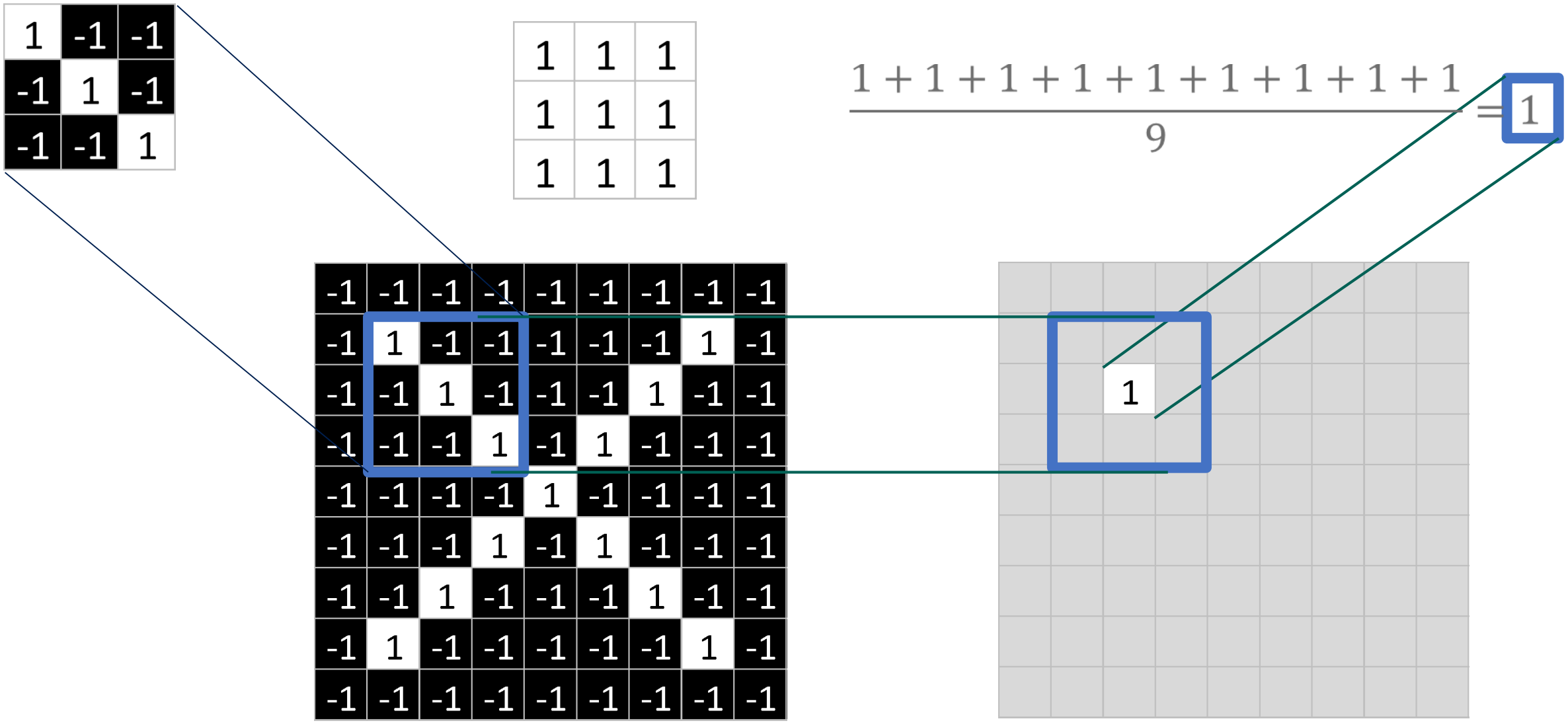

1 1 1
1 1 1
1 1

FILTRADO

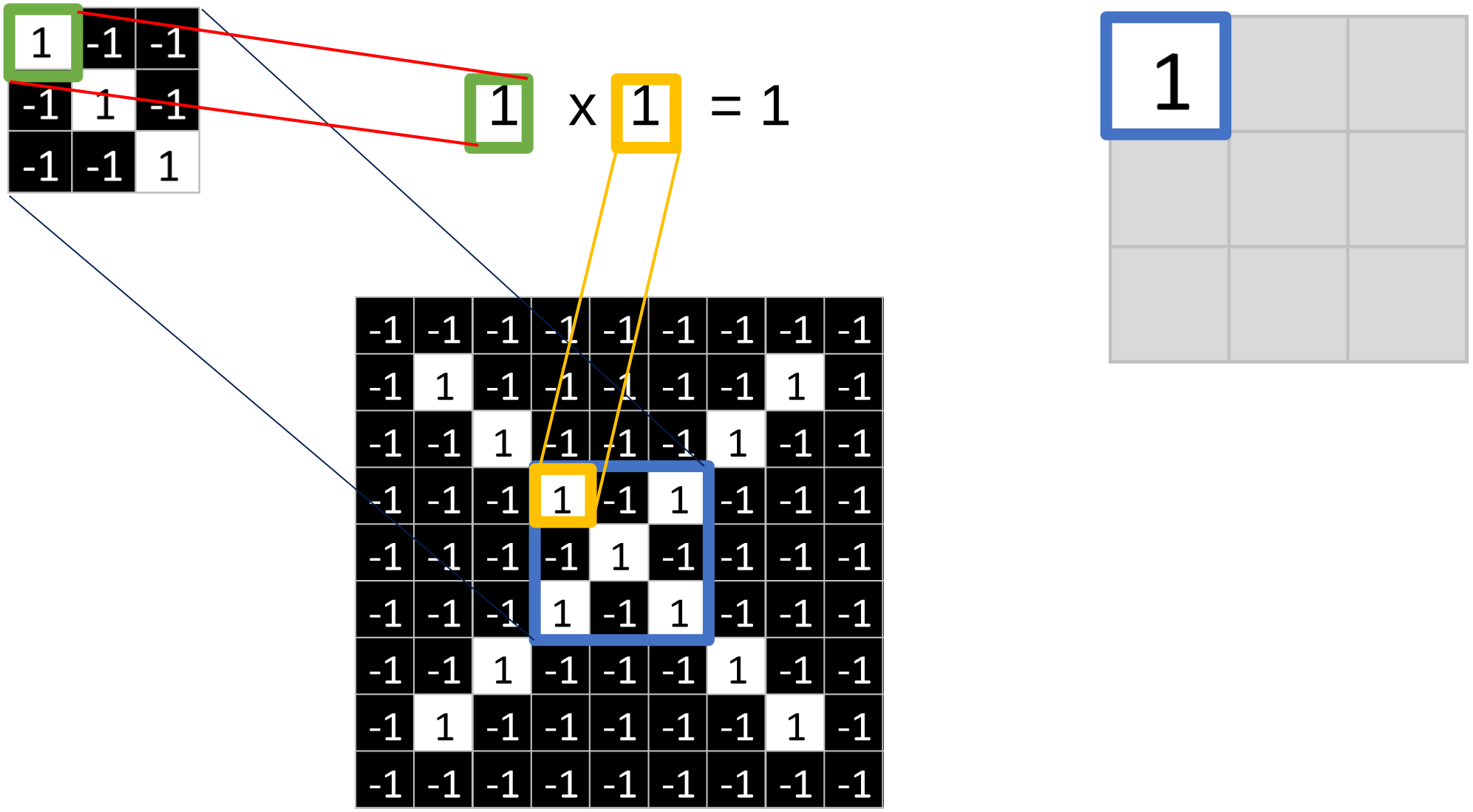


1	1	1
1	1	1
1	1	1

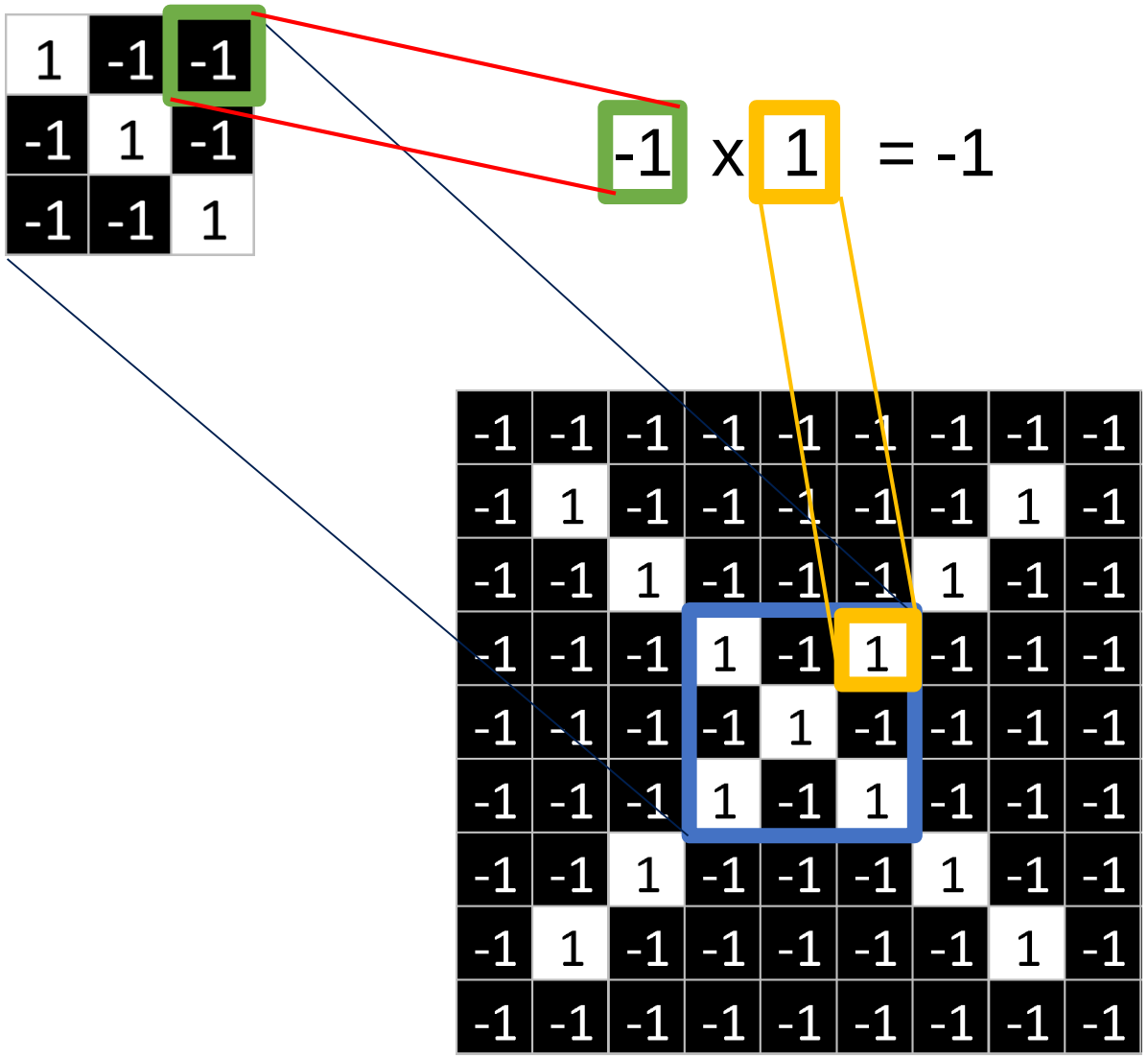
FILTRADO

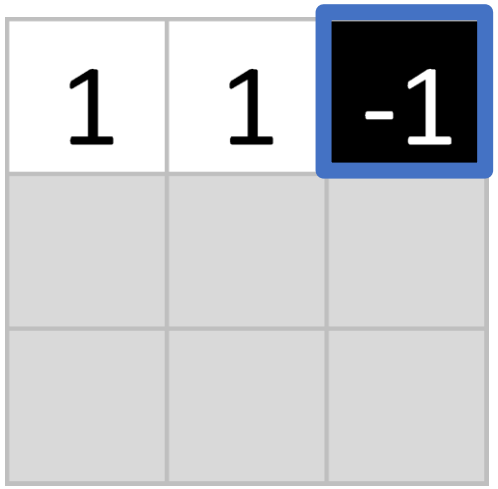

1 -1 -1
-1 1 -1
-1 -1 1
1 1 1
1 1 1
1 1 1
$\frac{1+1+1+1+1+1+1+1+1}{9} = 1$
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1
1

FILTRADO

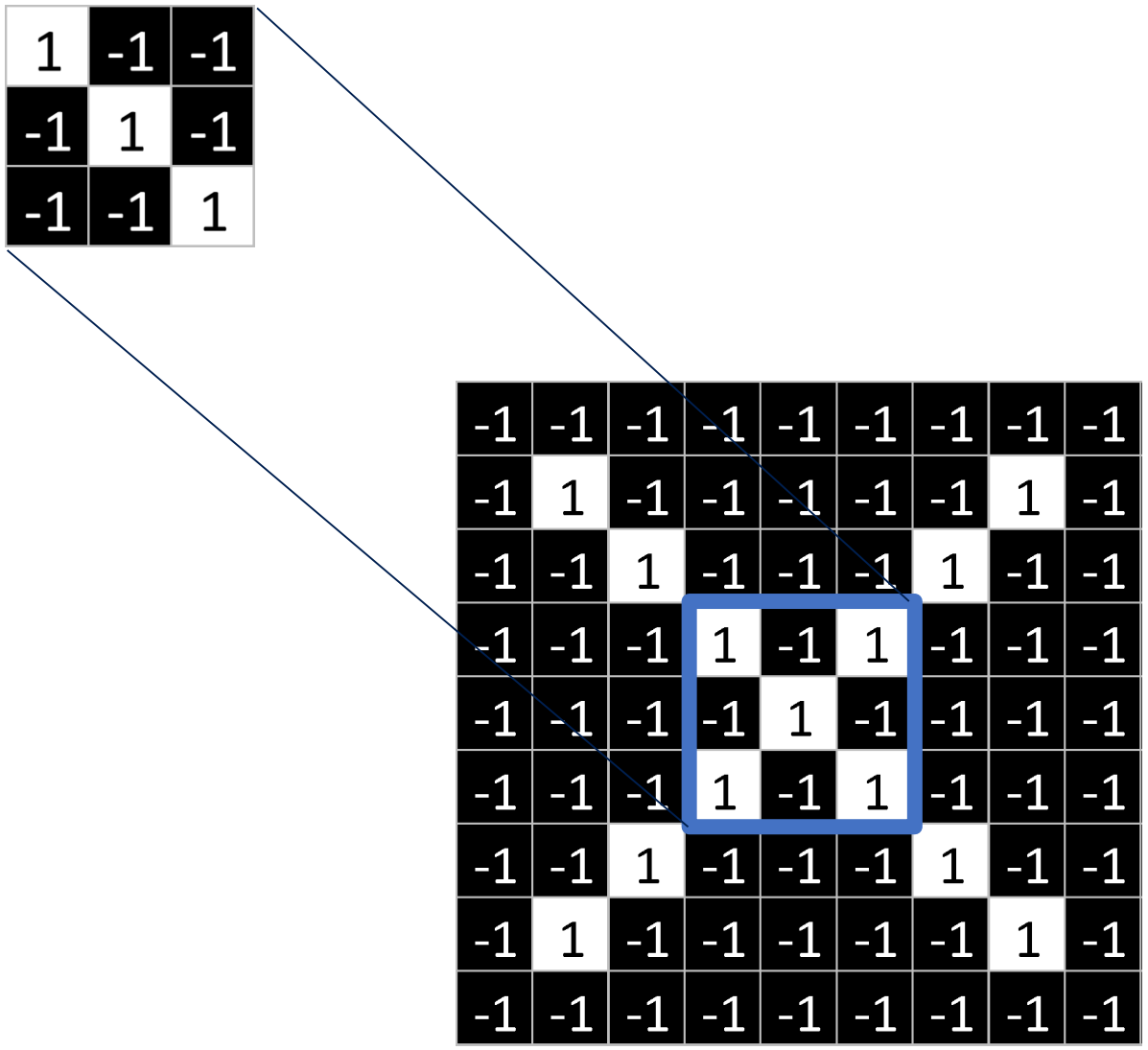

1 -1 -1
-1 1 -1
-1 -1 1
1 x 1 = 1
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1
1

FILTRADO


1 -1 -1
-1 1 -1
-1 -1 1
-1 x 1 = -1
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1

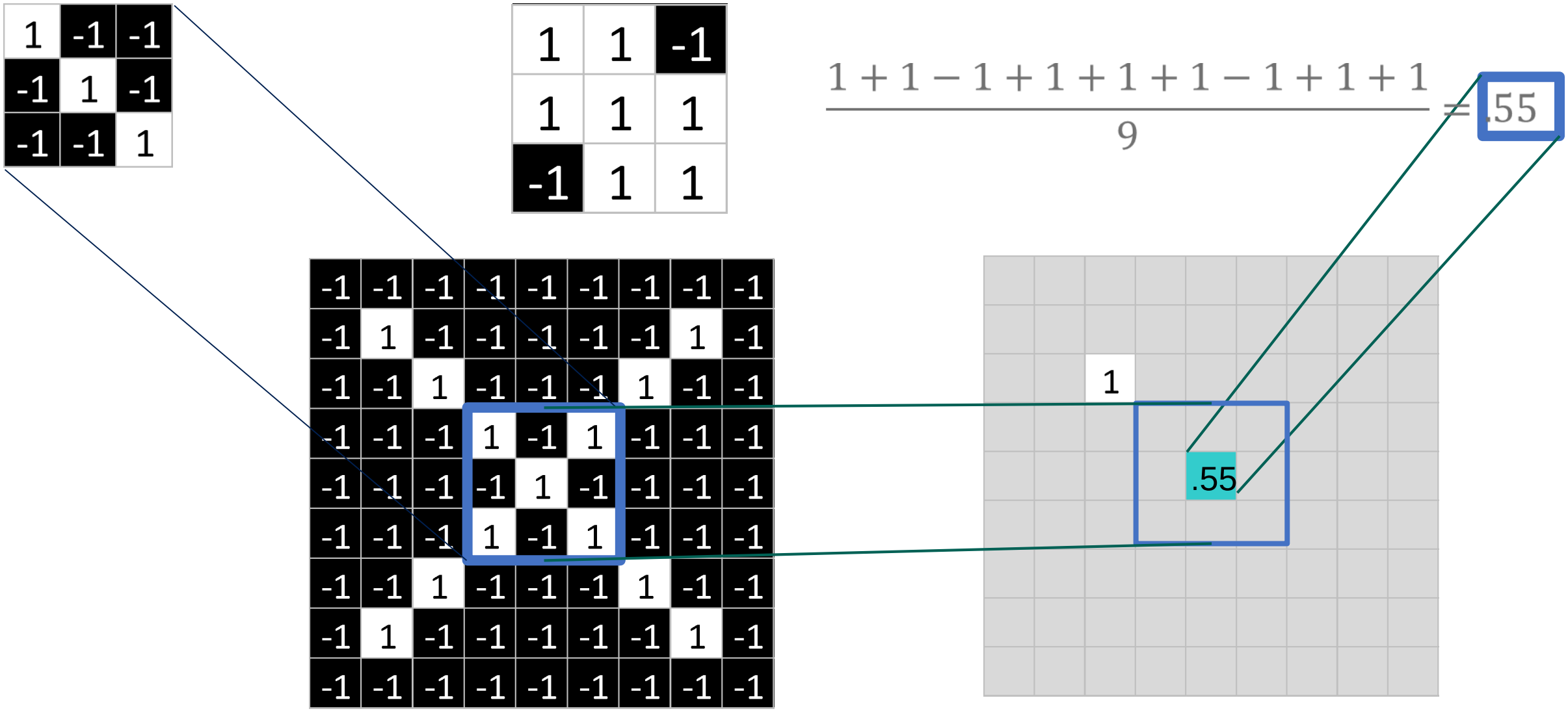

1 1 -1

FILTRADO



1	1	-1
1	1	1
-1	1	1

FILTRADO


1 -1 -1
-1 1 -1
-1 -1 1
1 1 -1
1 1 1
-1 1 1
$\frac{1+1-1+1+1+1-1+1+1}{9} = .55$
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1
1
.55

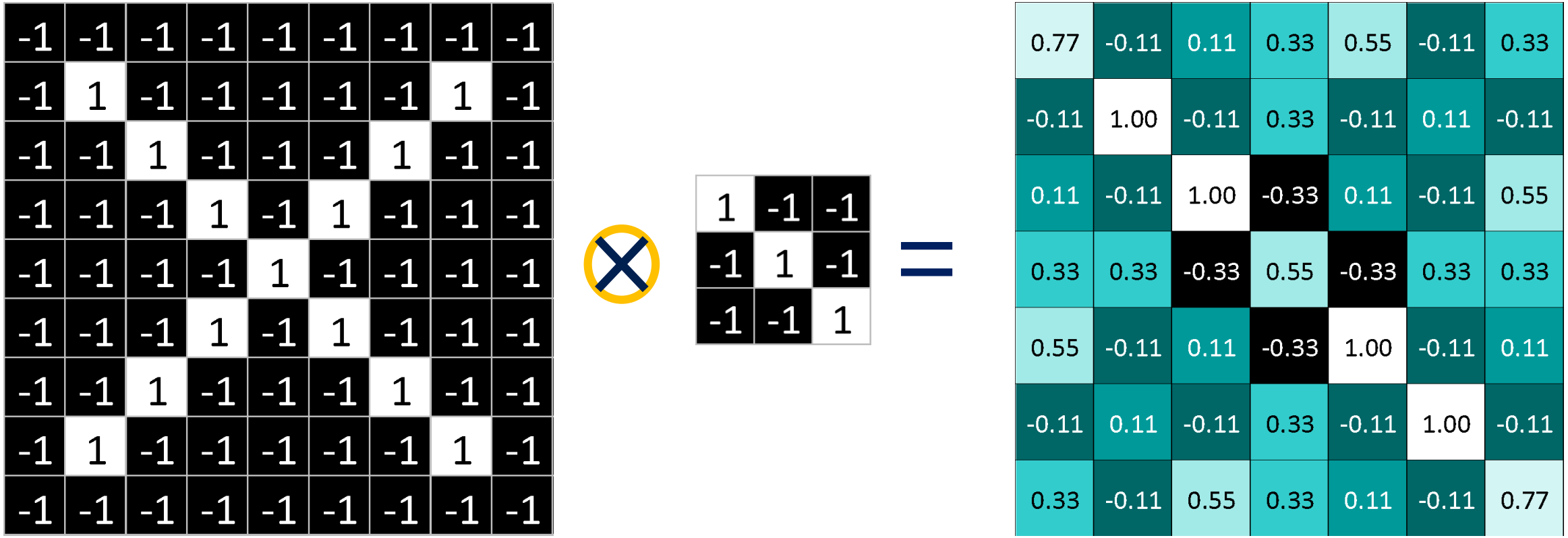
OPERACIÓN DE CONVOLUCIÓN

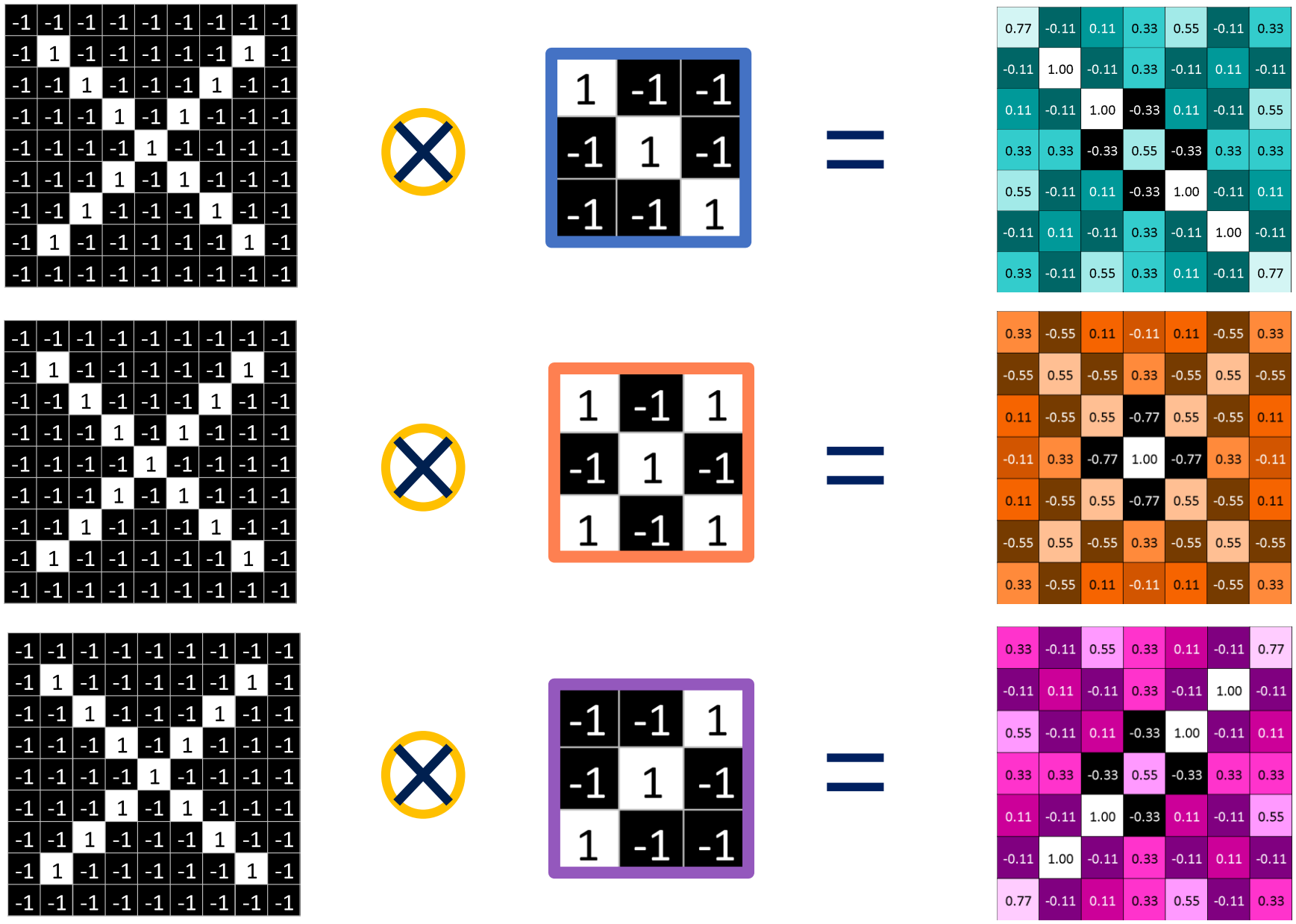
1	-1	-1
-1	1	-1
-1	-1	1

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

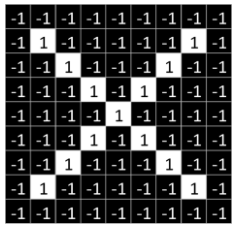
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

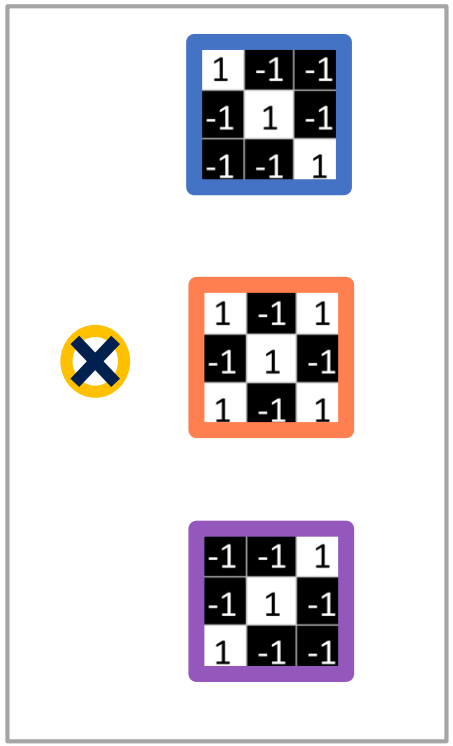
OPERACIÓN DE CONVOLUCIÓN

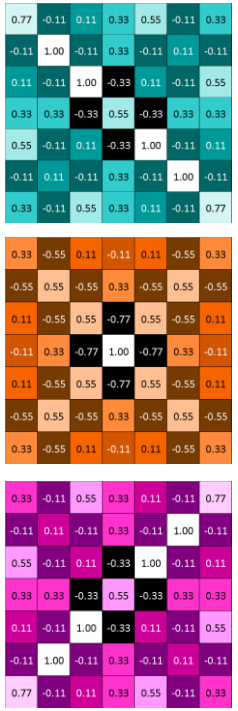



-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1
1 -1 -1
-1 1 -1
-1 -1 1
=
0.77 -0.11 0.11 0.33 0.55 -0.11 0.33
-0.11 1.00 -0.11 0.33 -0.11 0.11 -0.11
0.11 -0.11 1.00 -0.33 0.11 -0.11 0.55
0.33 0.33 -0.33 0.55 -0.33 0.33 0.33
0.55 -0.11 0.11 -0.33 1.00 -0.11 0.11
-0.11 0.11 -0.11 0.33 -0.11 1.00 -0.11
0.33 -0.11 0.55 0.33 0.11 -0.11 0.77
1 -1 1
-1 1 -1
1 -1 1
=
0.33 -0.55 0.11 -0.11 0.11 -0.55 0.33
-0.55 0.55 -0.55 0.33 -0.55 0.55 -0.55
0.11 -0.55 0.55 -0.77 0.55 -0.55 0.11
-0.11 0.33 -0.77 1.00 -0.77 0.33 -0.11
0.11 -0.55 0.55 -0.77 0.55 -0.55 0.11
-0.55 0.55 -0.55 0.33 -0.55 0.55 -0.55
0.33 -0.55 0.11 -0.11 0.11 -0.55 0.33
-1 -1 1
-1 1 -1
1 -1 -1
=
0.33 -0.11 0.55 0.33 0.11 -0.11 0.77
-0.11 0.11 -0.11 0.33 -0.11 1.00 -0.11
0.55 -0.11 0.11 -0.33 1.00 -0.11 0.11
0.33 0.33 -0.33 0.55 -0.33 0.33 0.33
0.11 -0.11 1.00 -0.33 0.11 -0.11 0.55
-0.11 1.00 -0.11 0.33 -0.11 0.11 -0.11
0.77 -0.11 0.11 0.33 0.55 -0.11 0.33

CAPA DE CONVOLUCIÓN

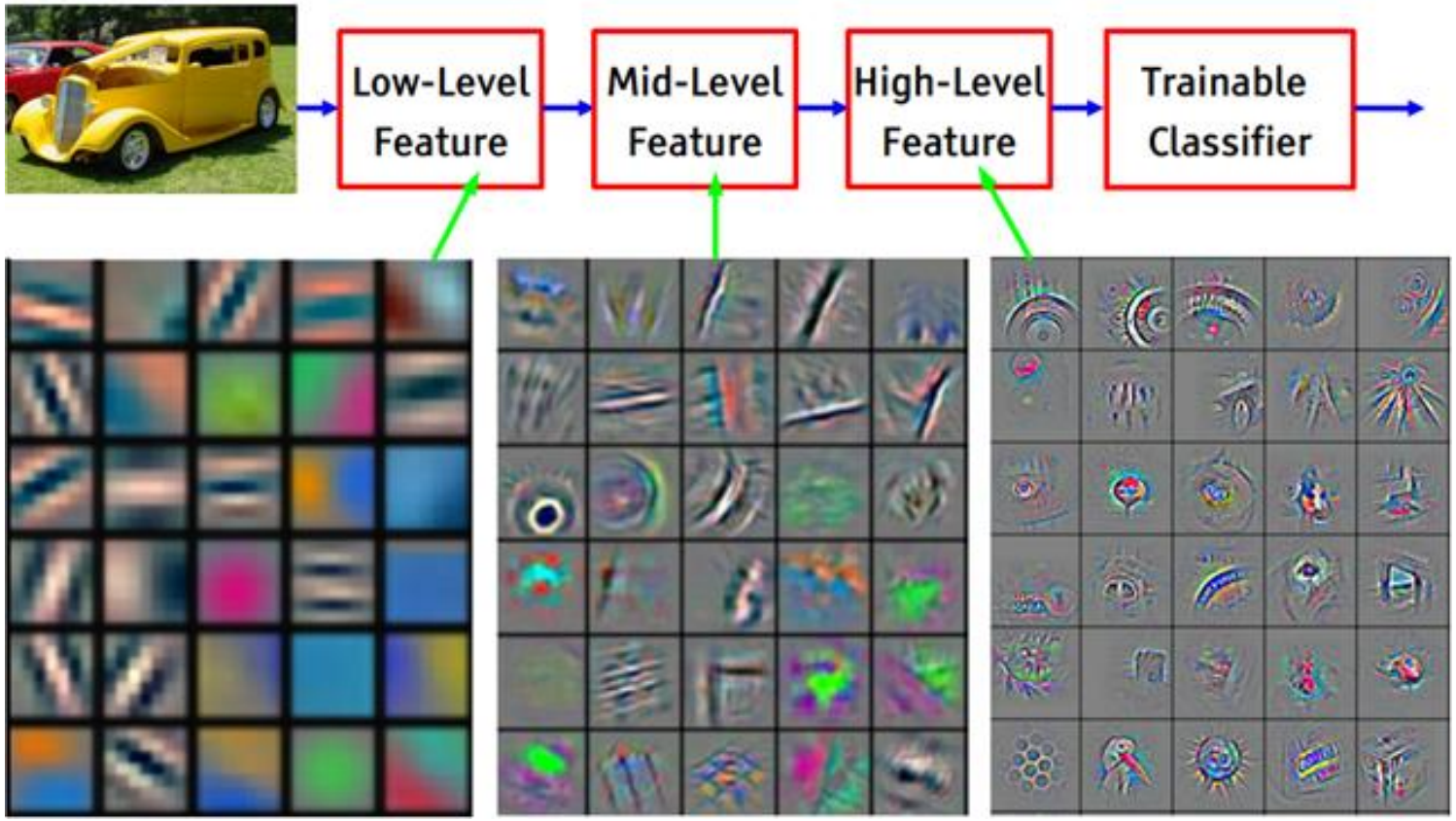






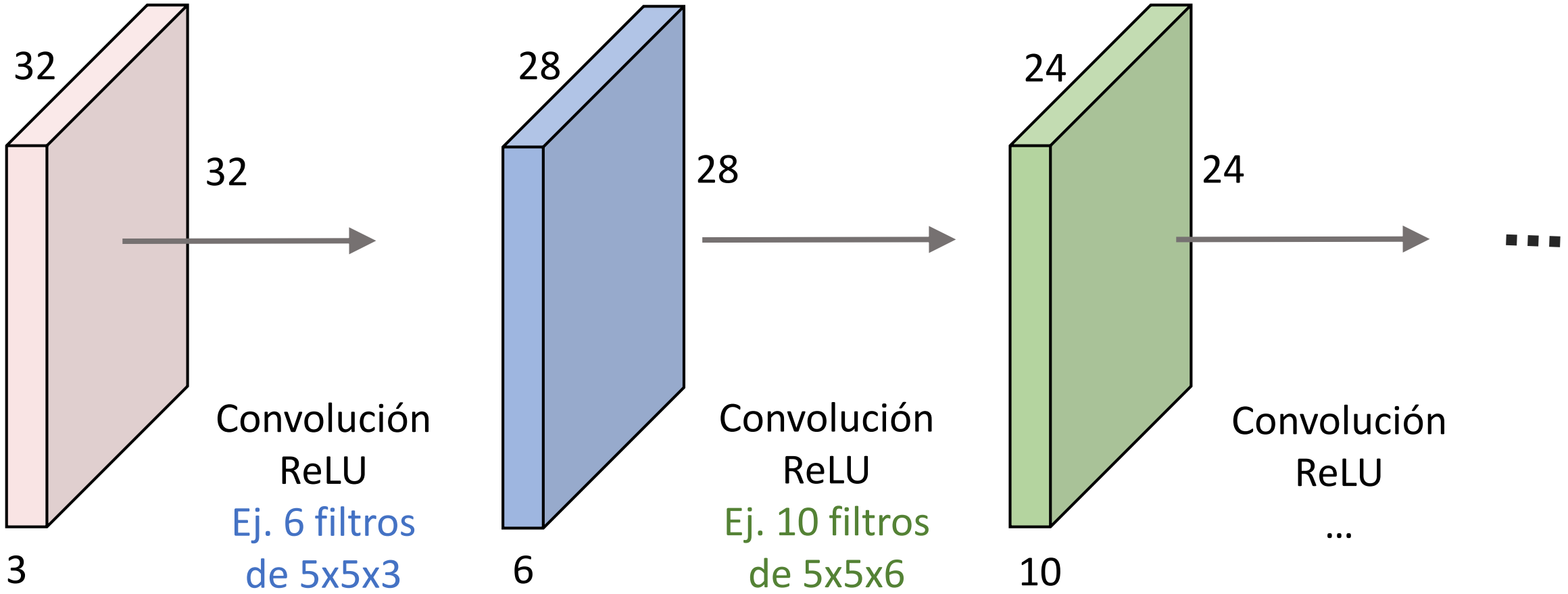
FILTROS COMO ATRIBUTOS

Atributos aprendidos por CNN



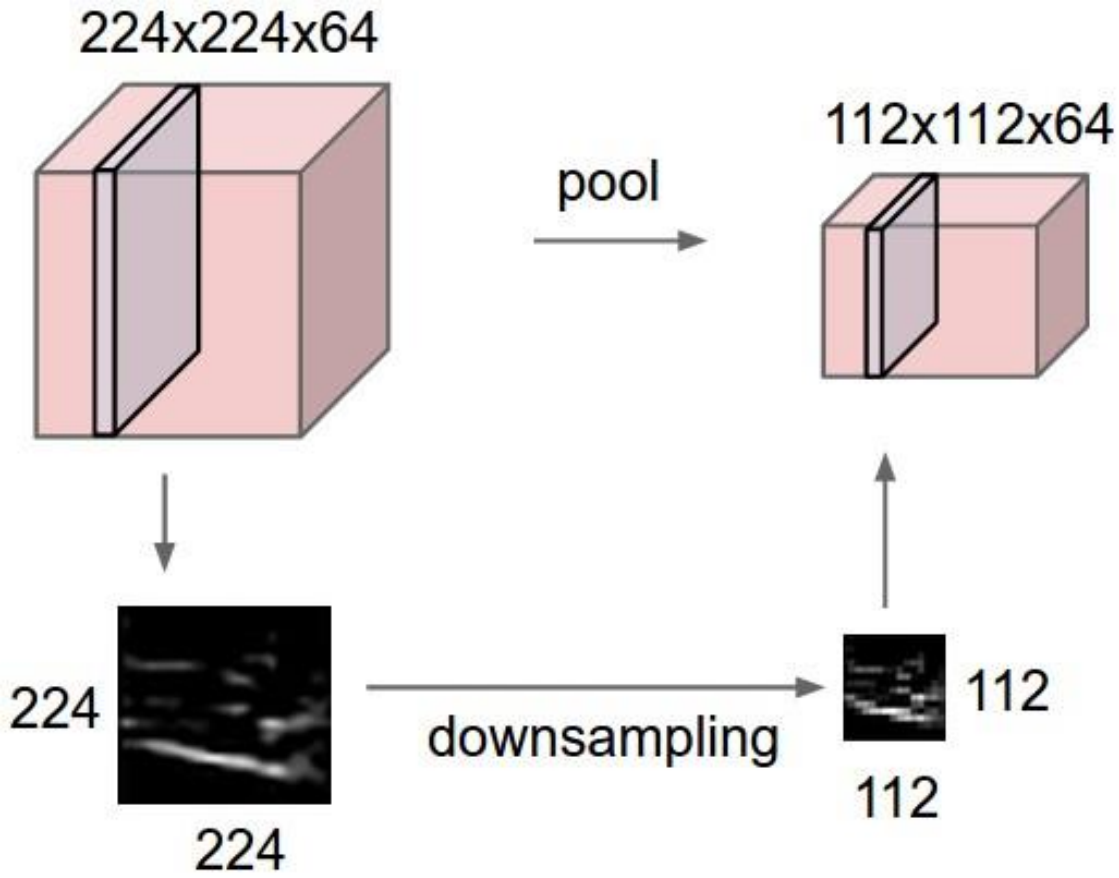
CONVNETS

- Secuencia de capas de convolución, intercalada con funciones de activación



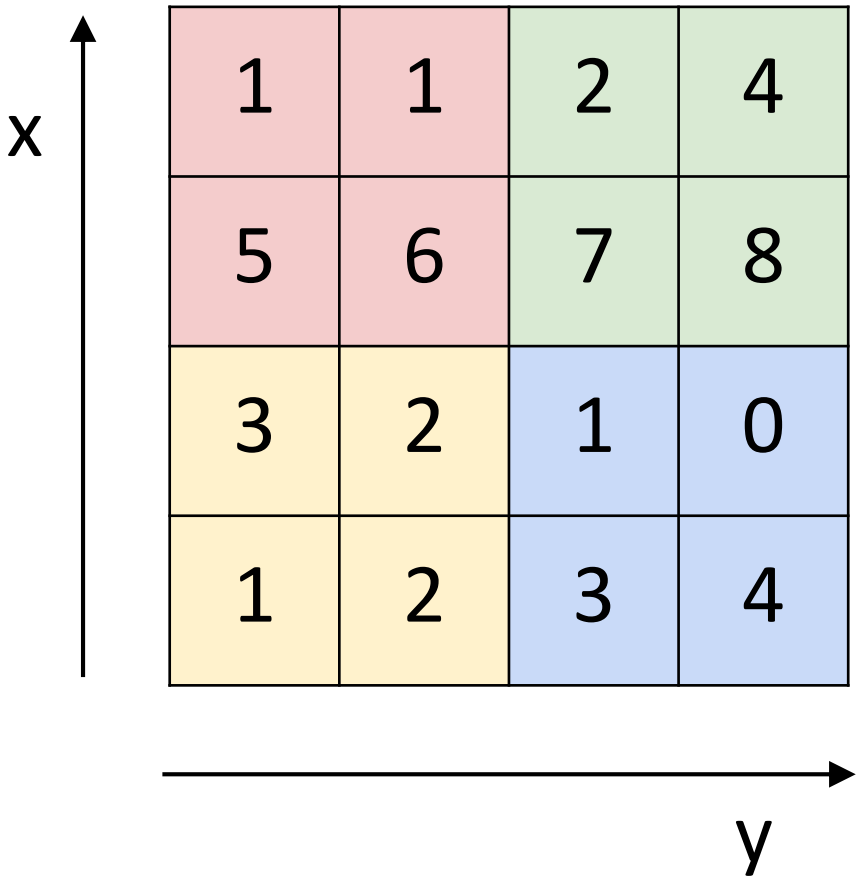
CAPA DE AGRUPAMIENTO (POOLING)

- Hace que las representaciones sean más pequeñas y manejables
- Opera sobre cada mapa de activación independientemente



MAX POOLING

Corte de una sola profundidad

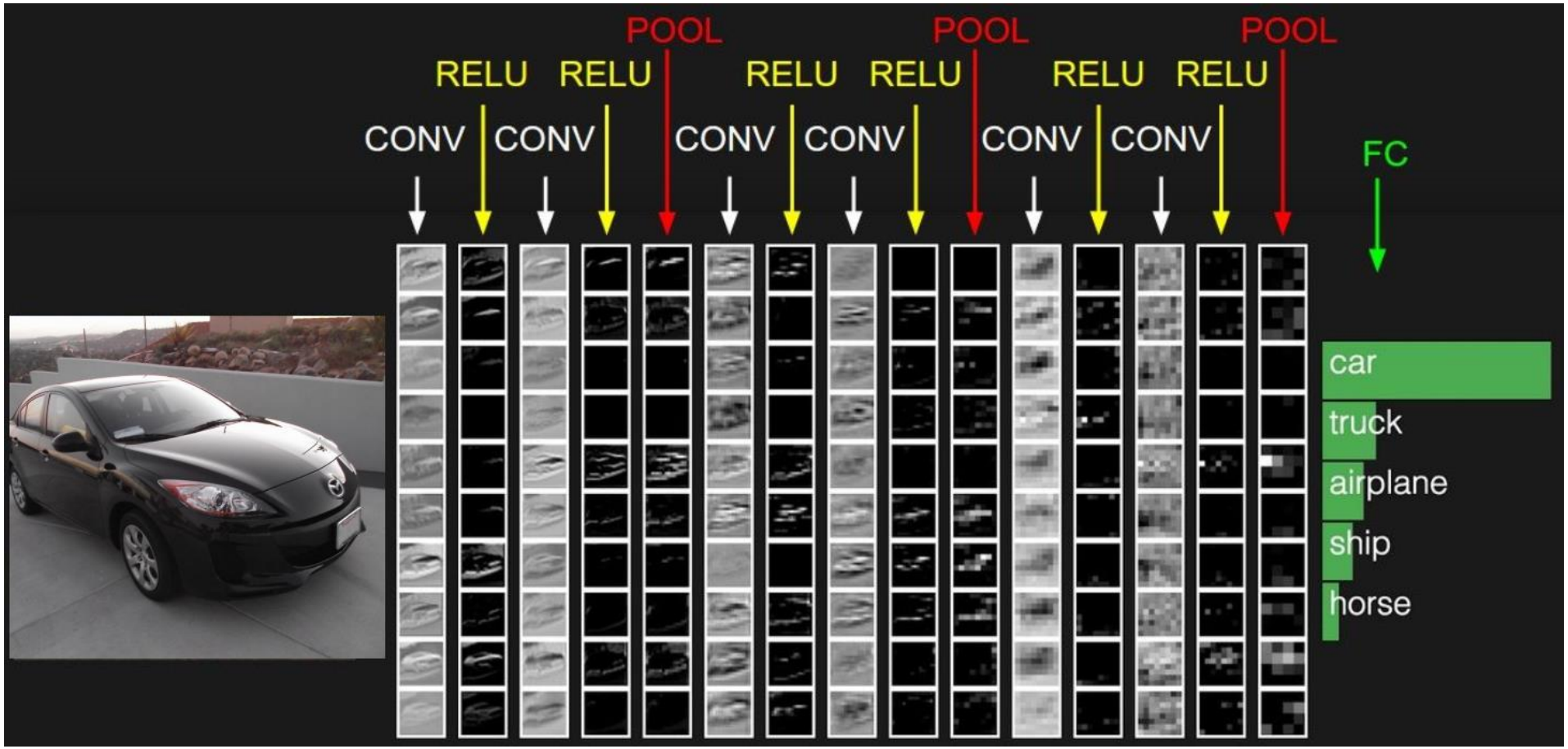


max pool con filtros
2x2 y paso de 2

6	8
3	4

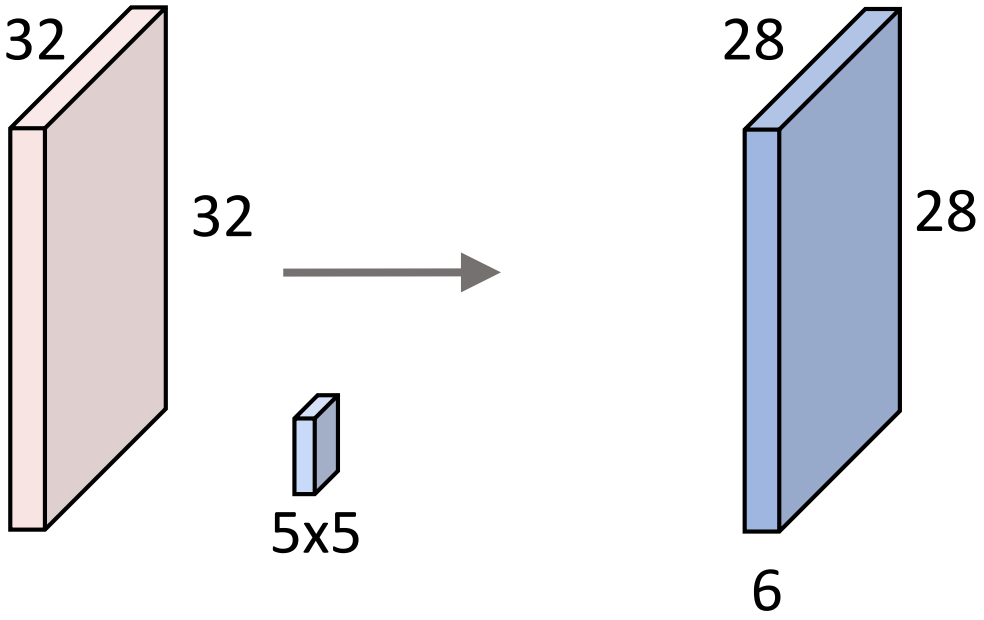
CAPA COMPLETAMENTE CONECTADA (FC)

- Contiene neuronas que se conectan al volumen de entradas completo, como en ANN estándar



APRENDIZAJE

- Supervisado (hay una salida deseada de referencia)
- Como redes convencionales se usa *Backpropagation*
- Pero **se ajusta el valor de los pesos de los distintos kernels**
- Mucho más económico que una red completamente conectada



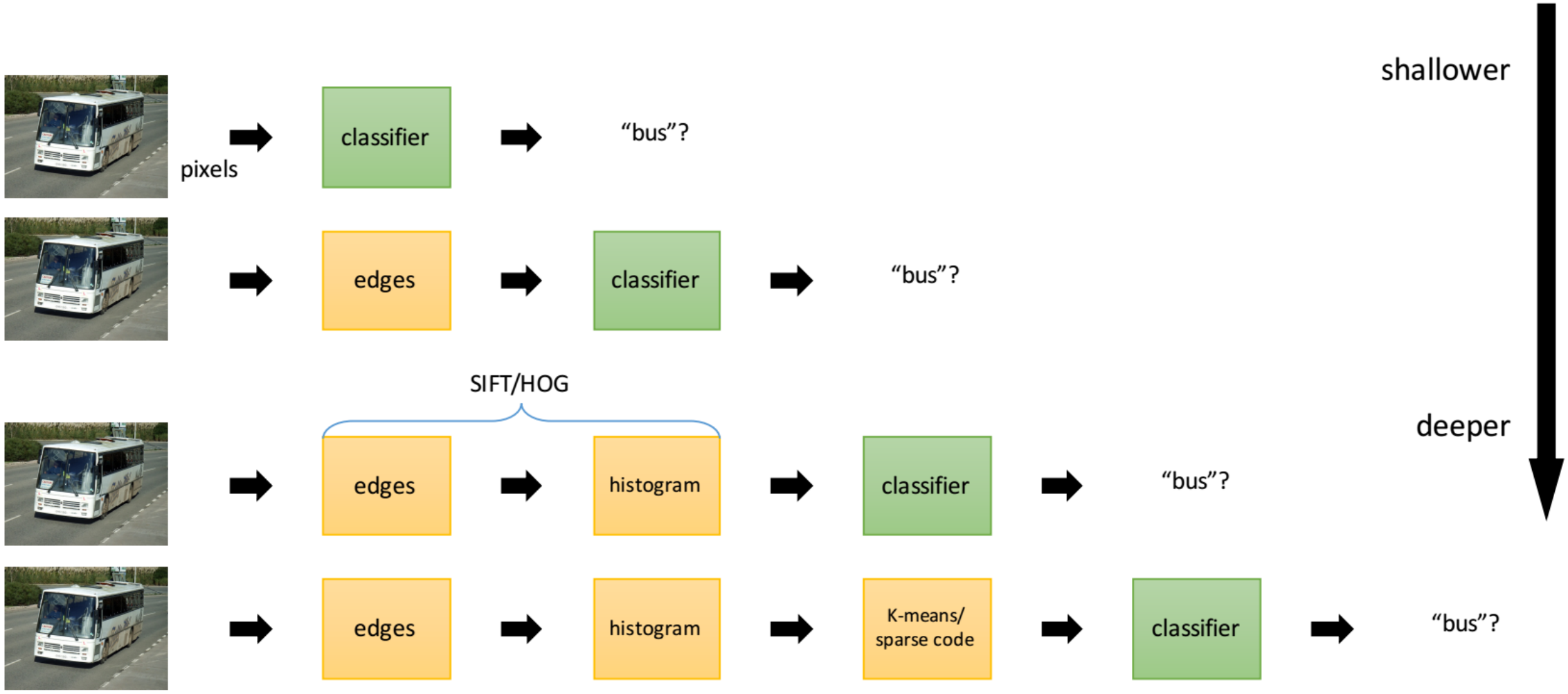
Ej. Cantidad de parámetros a ajustar en 1 capa:

MLP (32 x 32) x (28 x 28 x 6) ~ 4.8M Parámetros

Vs.

CNN (5 x 5 x 6) = 150 Parámetros

RECONOCIMIENTO DE OBJETOS Y DL



RECONOCIMIENTO DE OBJETOS Y DL

Specialized components, domain knowledge required

edges → histogram → K-means/ sparse code → classifier → "bus"?

Generic components ("layers"), less domain knowledge

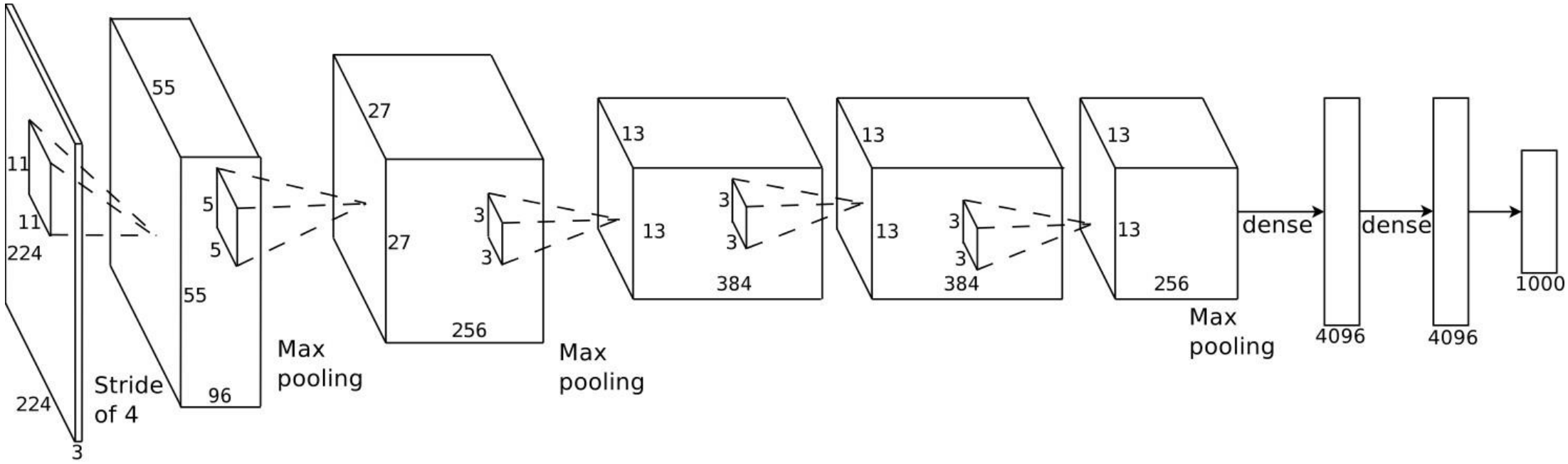
→ □ → □ → □ → □ → "bus"?

Repeat elementary layers => Going deeper

→ □ → □ → □ → □ → □ → □ → "bus"?

- Aprendizaje automático de principio a fin
- Espacio de soluciones más rico

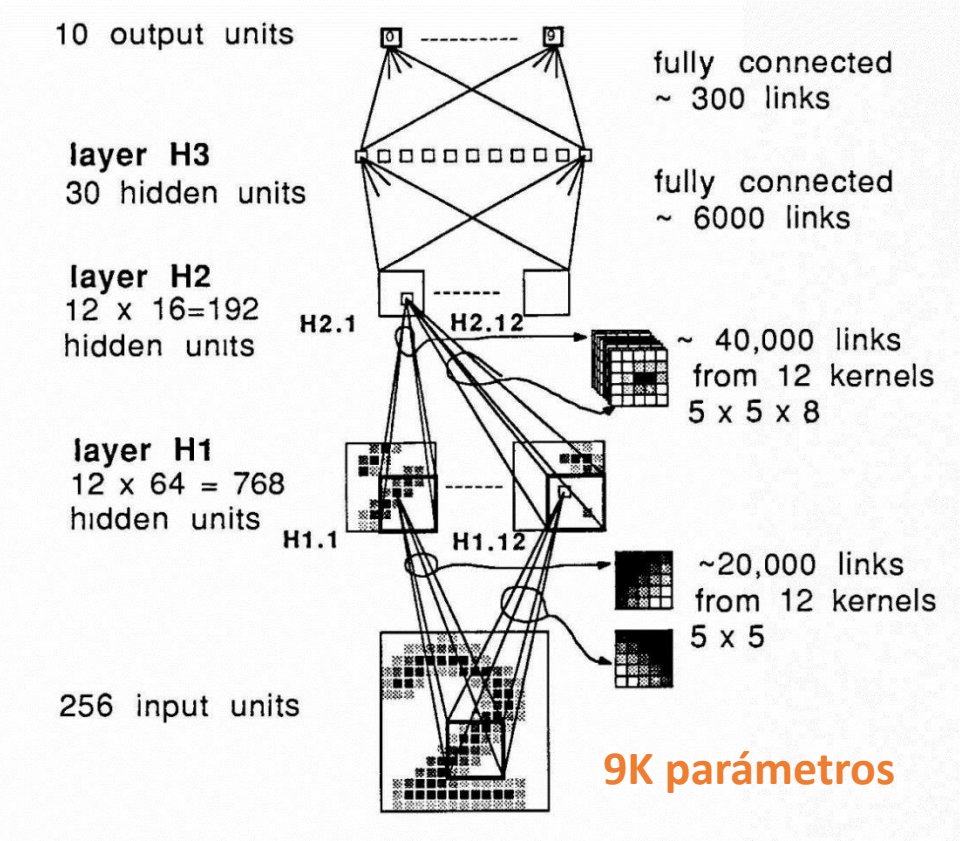
ALEXNET (KRIZHEVSKY ET AL., NIPS 2012)



- Presentadas en 2012, mejoraron significativamente el estado del arte
- Error top-5 de 16% vs. 26% del algoritmo en la 2º ubicación

CRECIMIENTO EN COMPLEJIDAD

LeCun et al, *"Backpropagation Applied to Handwritten Zip Code Recognition"*. 1989



- Szegedy et al, *"Going Deeper with Convolution"*. 2014 Ganador de ImageNet Large-Scale Visual Recognition Challenge 2014

- **GoogLeNet (7.89% error)**
  - **22 capas**
  - **6.8M parámetros**
  - **1.5*B* flops**
  - **Ensamble de 7 modelos**

- **Mejor Actual: ResNet (3.57% error)**
  - **152 capas**
  - **2.3M parámetros**
  - **11.3*B* flops**
  - **Ensamble de 6 modelos**

COSTO DE CÓMPUTO

- Cientos de millones de parámetros + convoluciones & desplegado
- Requiere aceleración hardware

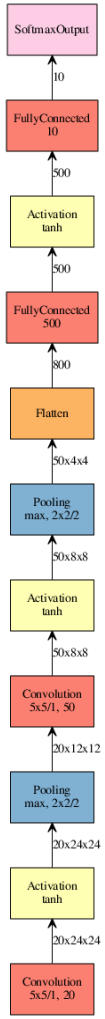
Network: GoogLeNet	Batch Size	Titan X (FP32)	Tegra X1 (FP32)	Tegra X1 (FP16)
Inference Performance	1	138 img/sec	33 img/sec	33 img/sec
Power		119.0 W	5.0 W	4.0 W
Performance/Watt		1.2 img/sec/W	6.5 img/sec/W	8.3 img/sec/W
Inference Performance	128 (Titan X) 64 (Tegra X1)	863 img/sec	52 img/sec	75 img/sec
Power		225.0 W	5.9 W	5.8 W
Performance/Watt		3.8 img/sec/W	8.8 img/sec/W	12.8 img/sec/W

Table 3 GoogLeNet inference results on Tegra X1 and Titan X. Tegra X1's total memory capacity is not sufficient to run batch size 128 inference.

http://www.nvidia.com/content/tegra/embedded-systems/pdf/jetson_tx1_whitepaper.pdf

ARQUITECTURAS DE CNN

"LeNet" 1998:



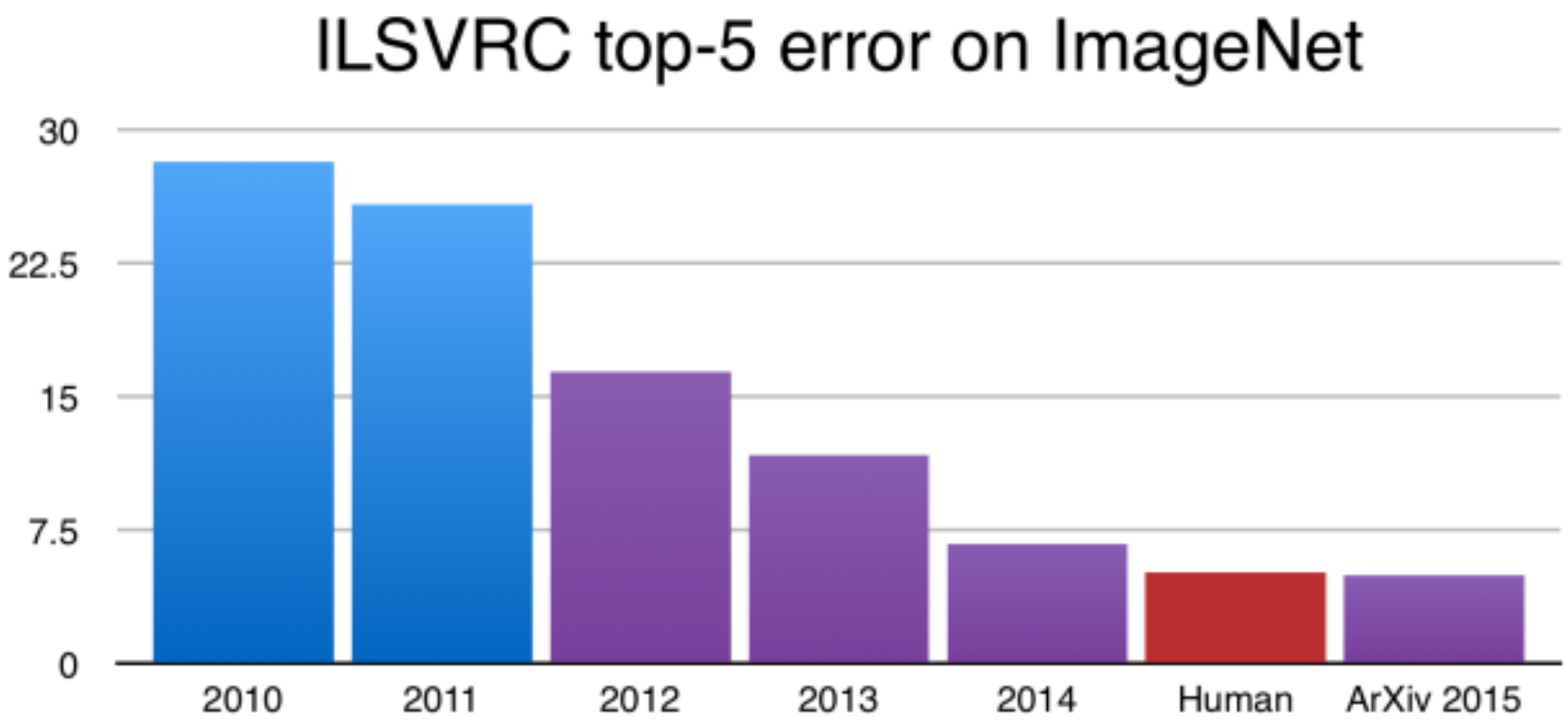
"AlexNet" 2012:

"GoogLeNet" 2014:

"Inception v3" 2015:

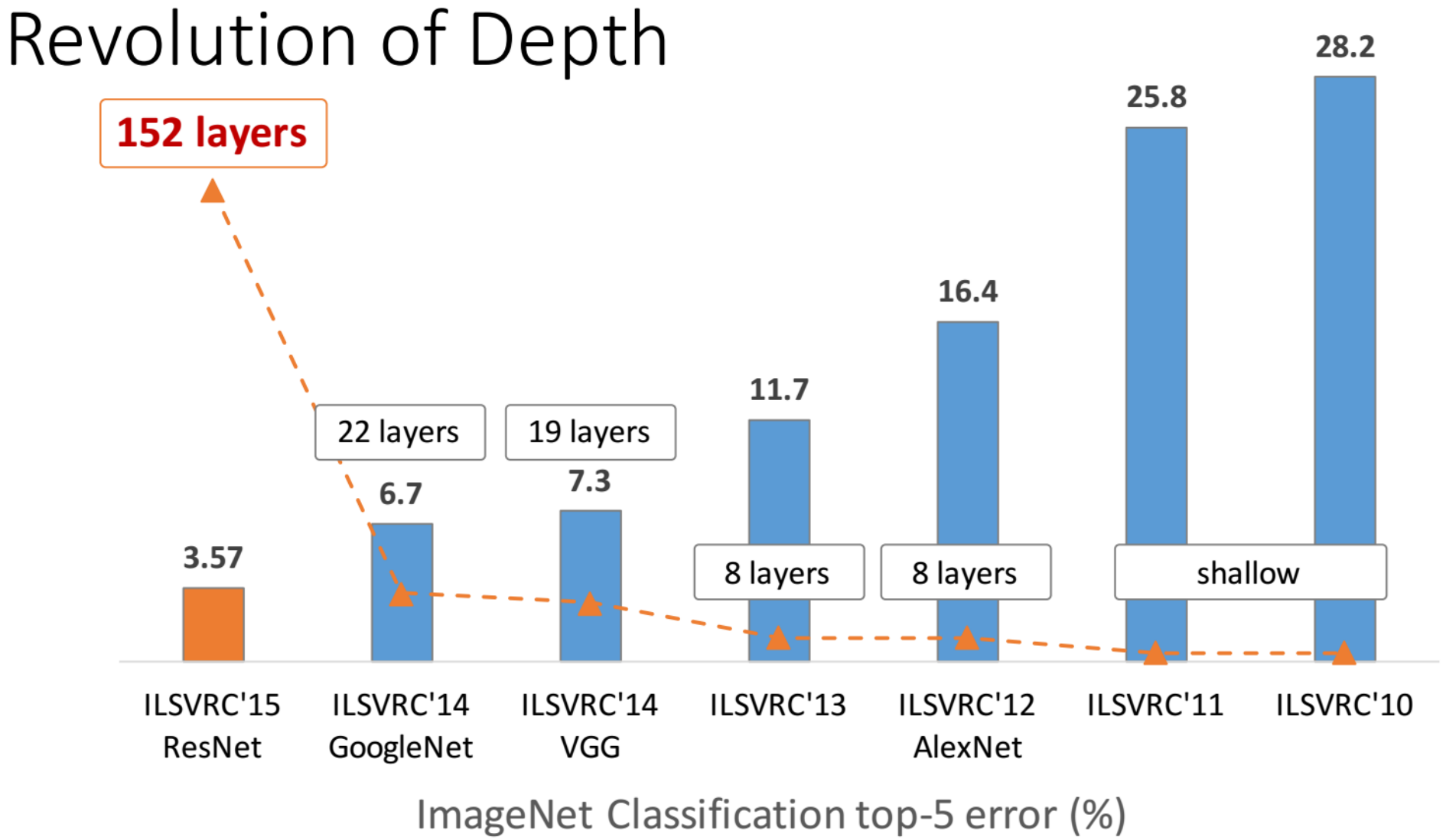
Source: http://josephpcohen.com/w/visualizing-cnn-architectures-side-by-side-with-mxnet/

MEJORAS EN RECONOCIMIENTO SOBRE IMAGENET



IMAGENET Large Scale Visual Recognition Challenge

AUMENTO DE LA COMPLEJIDAD SOBRE IMAGNET



He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778)

RESUMEN - COMPARACIÓN DE REDES

	MLP	RNN	CNN
Tipo de Datos	Tabular	Secuencias (series temporales, audio, texto, …)	Imágenes
Conexiones recurrentes	✗	✓	✗
Parámetros compartidos	✗	✓	✓
Relaciones espaciales	✗	✗	✓
Desvanecimiento y explosión de parámetros	✓	✓	✓

BIBLIOGRAFÍA

- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105)
- Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. In Neural networks: Tricks of the trade (pp. 437-478). Springer Berlin Heidelberg
- LeCun, Y. A., Bottou, L., Orr, G. B., & Müller, K. R. (2012). Efficient backprop. In *Neural networks: Tricks of the trade* (pp. 9-48). Springer Berlin Heidelberg
- Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems (pp. 91-99)
- Taigman, Y., Yang, M., Ranzato, M. A., & Wolf, L. (2014). Deepface: Closing the gap to human-level performance in face verification. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1701-1708)
- Karpathy, A., & Fei-Fei, L. (2015). Deep visual-semantic alignments for generating image descriptions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 3128-3137)
- Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In AAAI (pp. 4278-4284)