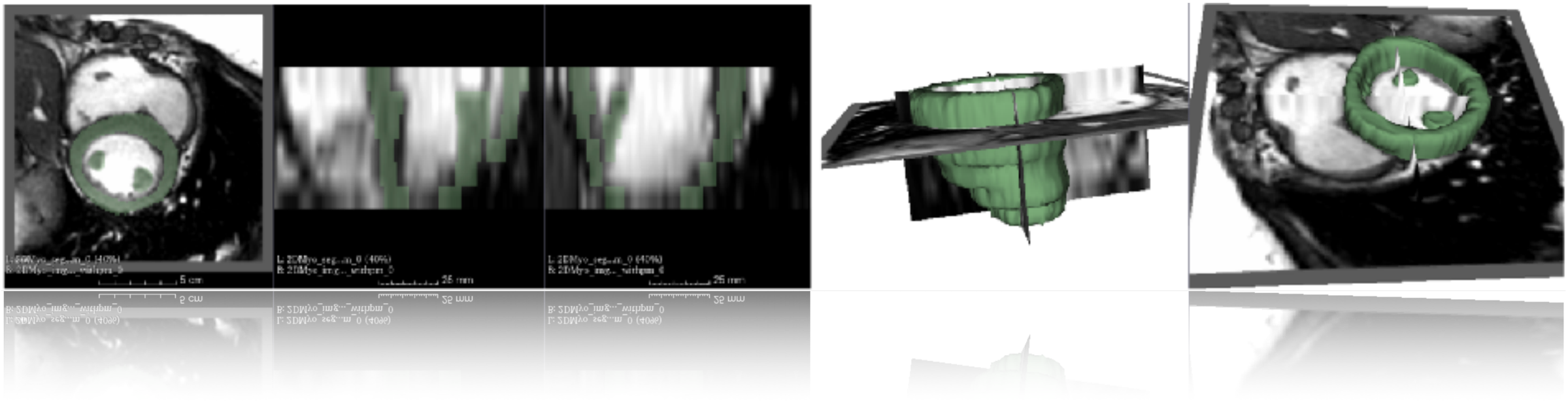


Departamento de Física Médica - Centro atómico Bariloche - IB

Redes Convolucionales

Ariel Hernán Curiale
ariel.curiale@cab.cnea.gov.ar

Algunos slides fueron adaptados de Fei Fei Li, J. Johnson y S. Yeung, cs231n, Stanford 2017.

CONICET

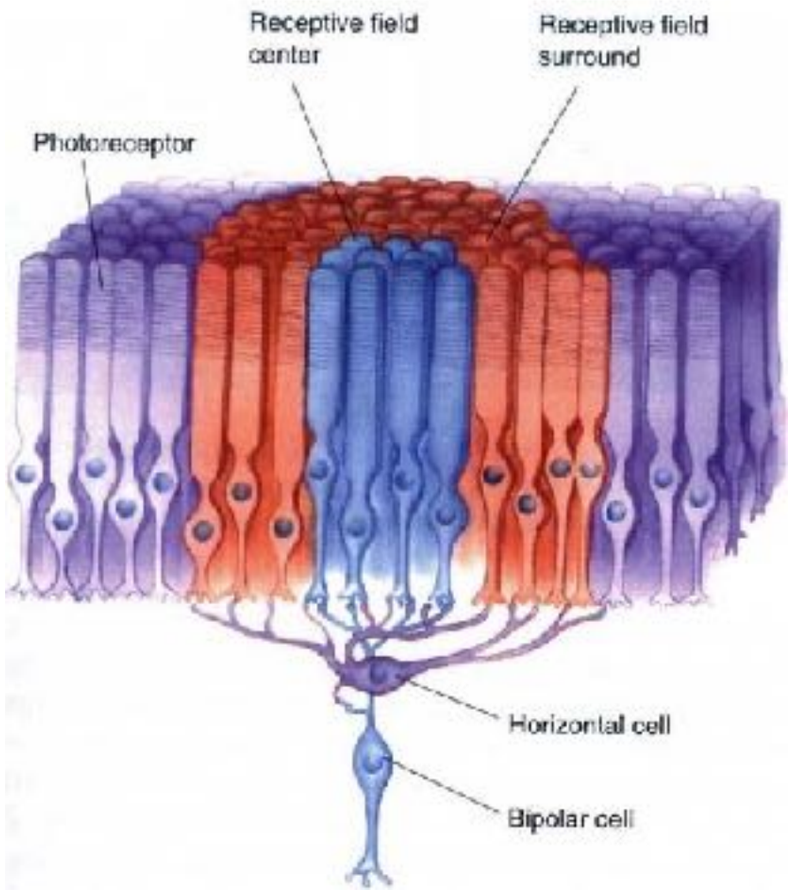
UNCUYO
UNIVERSIDAD NACIONAL DE CUYO

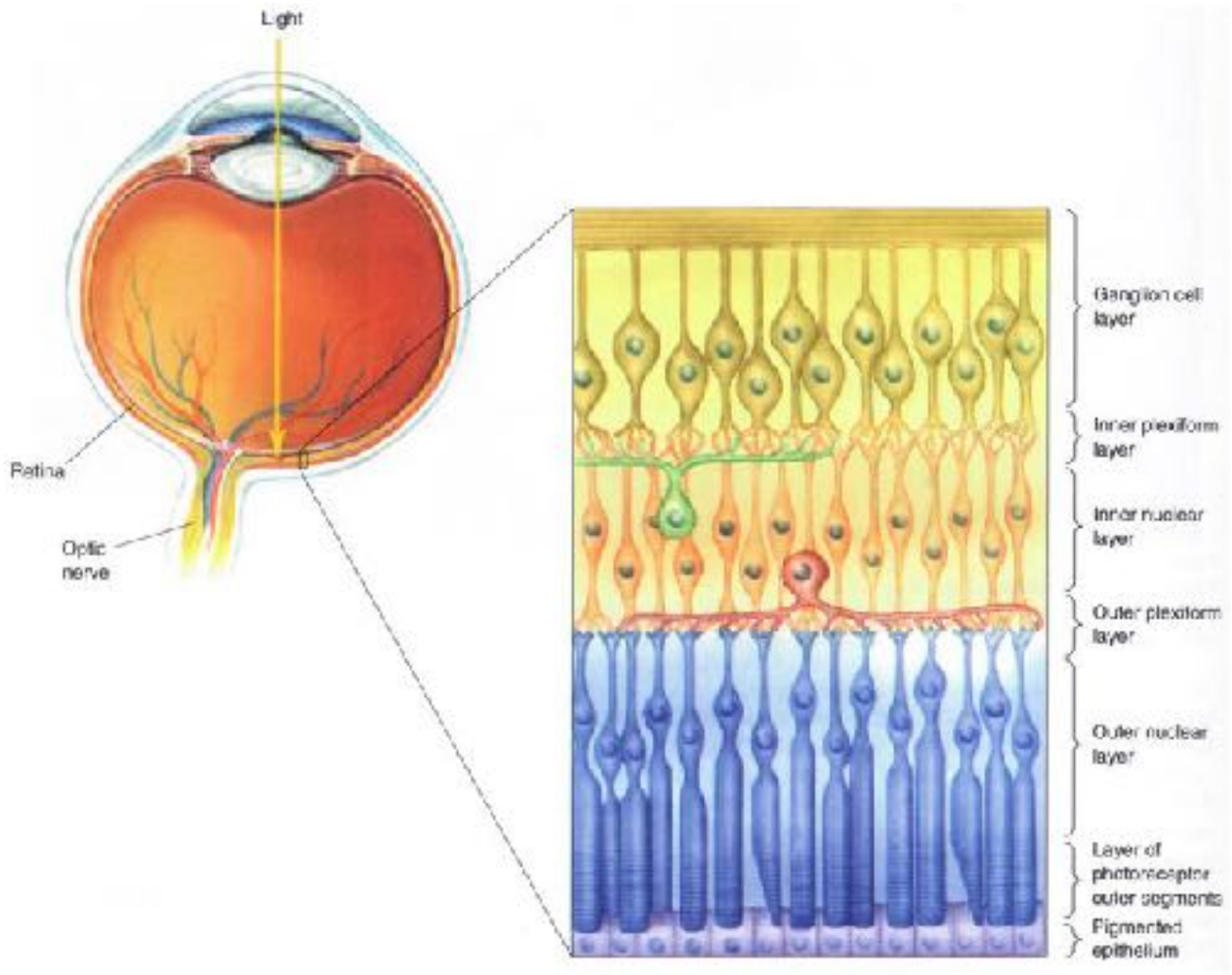
Instituto Balseiro

CNEA

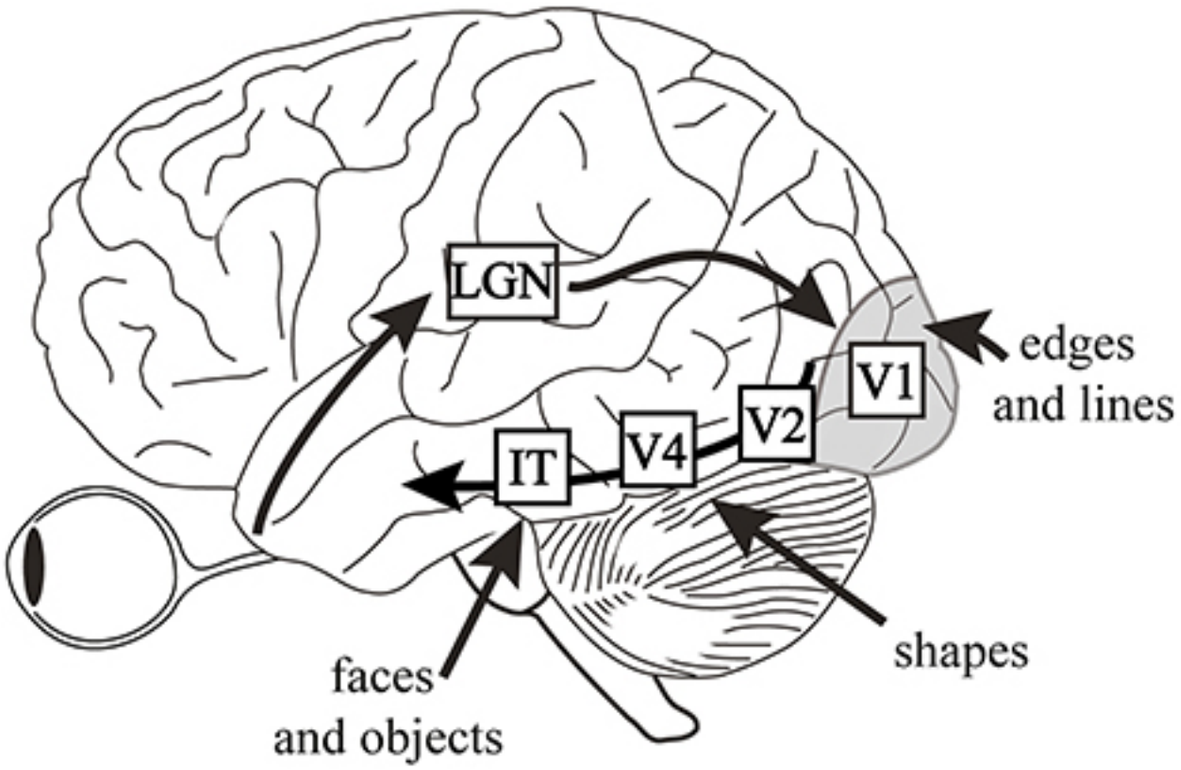
Redes Convolucionales

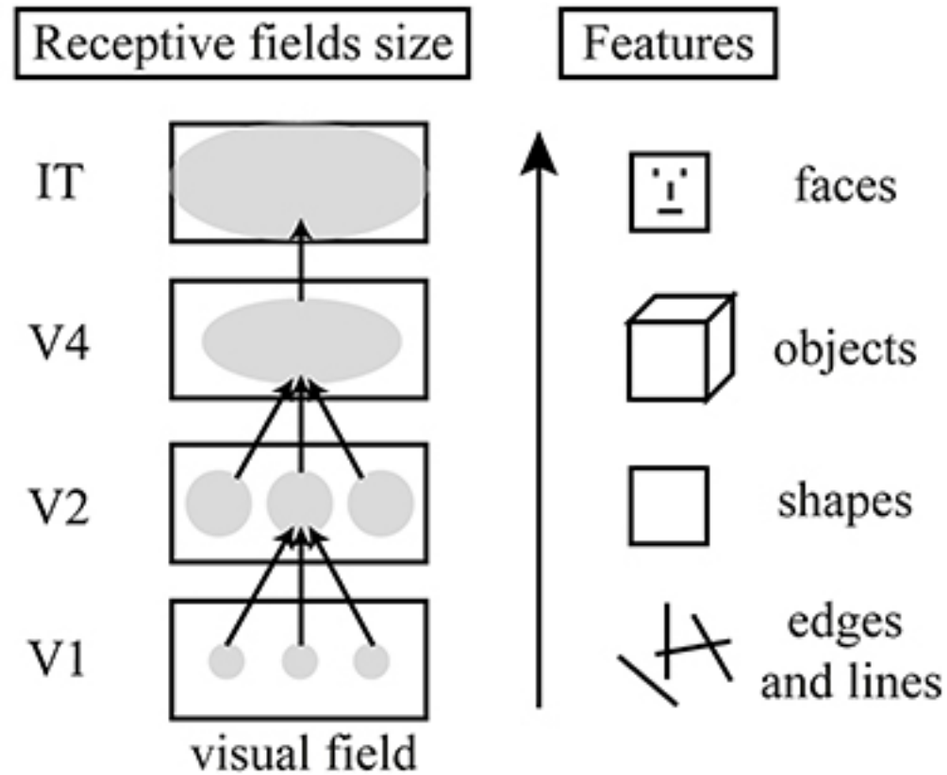
- Convolutional Neural Networks (Fukushima 1980, LeCun, 1989): Este tipo de redes neuronales esta inspirada en la idea de campo receptivo de los sistemas sensoriales.
  - Por ejemplo, las células ganglionares de la retina pueden ser excitadas o inhibidas dependiendo del lugar del campo receptivo donde pasa el estimulo visual.





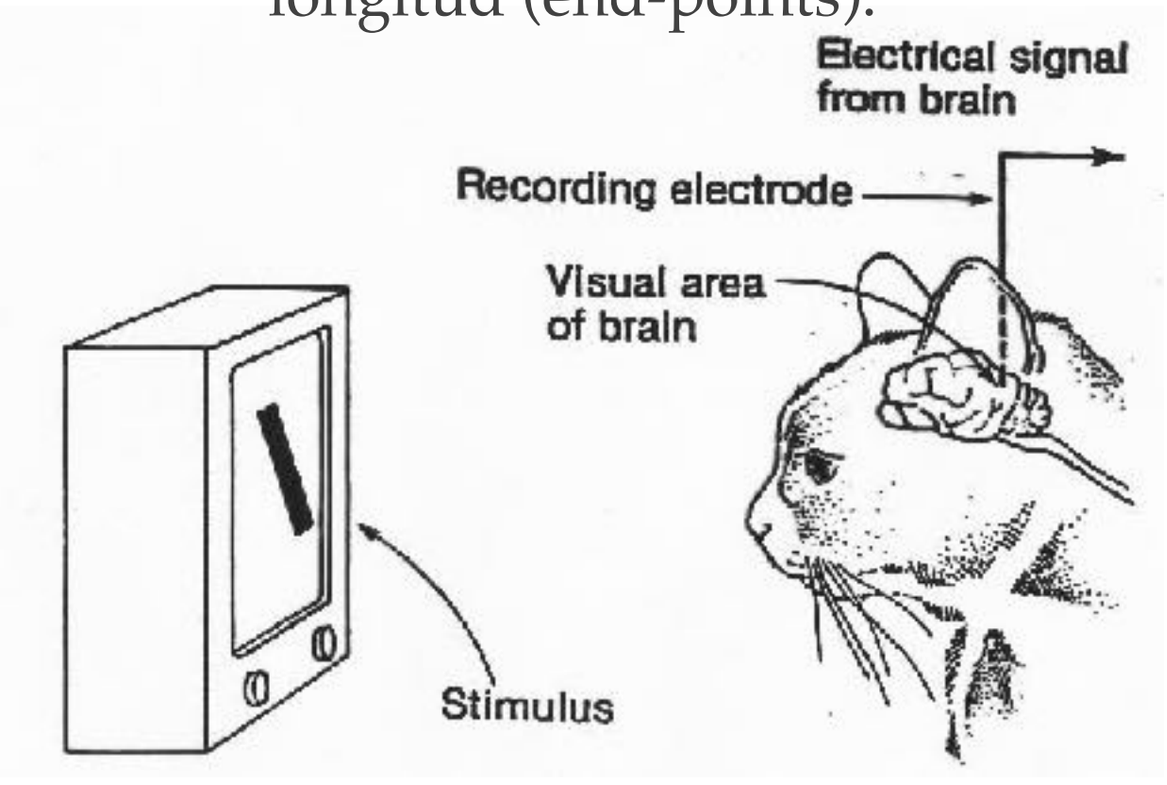
Introducción





Introducción

- Hubel & Wiesel, 1959 (Premio Nobel 1981): Descubren distintos tipos de células en la corteza visual primaria
  - Simples: responden a orientación y posición.
  - Complejas: responden orientación, movimiento y dirección
  - Hipercomplejas (1965): responden a orientación, movimiento, dirección y longitud (end-points).



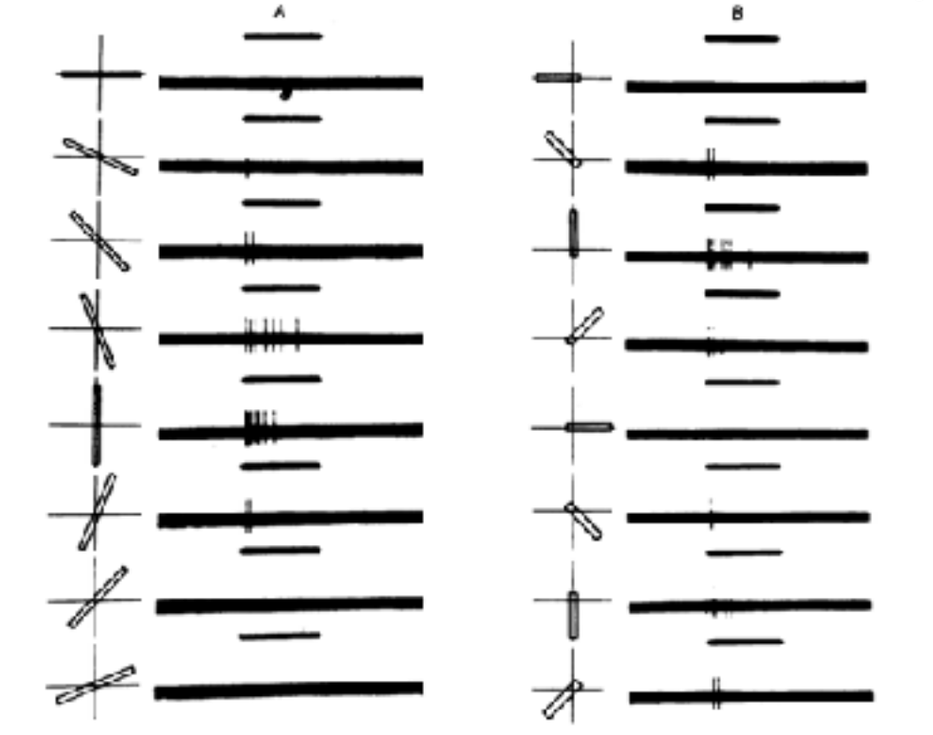
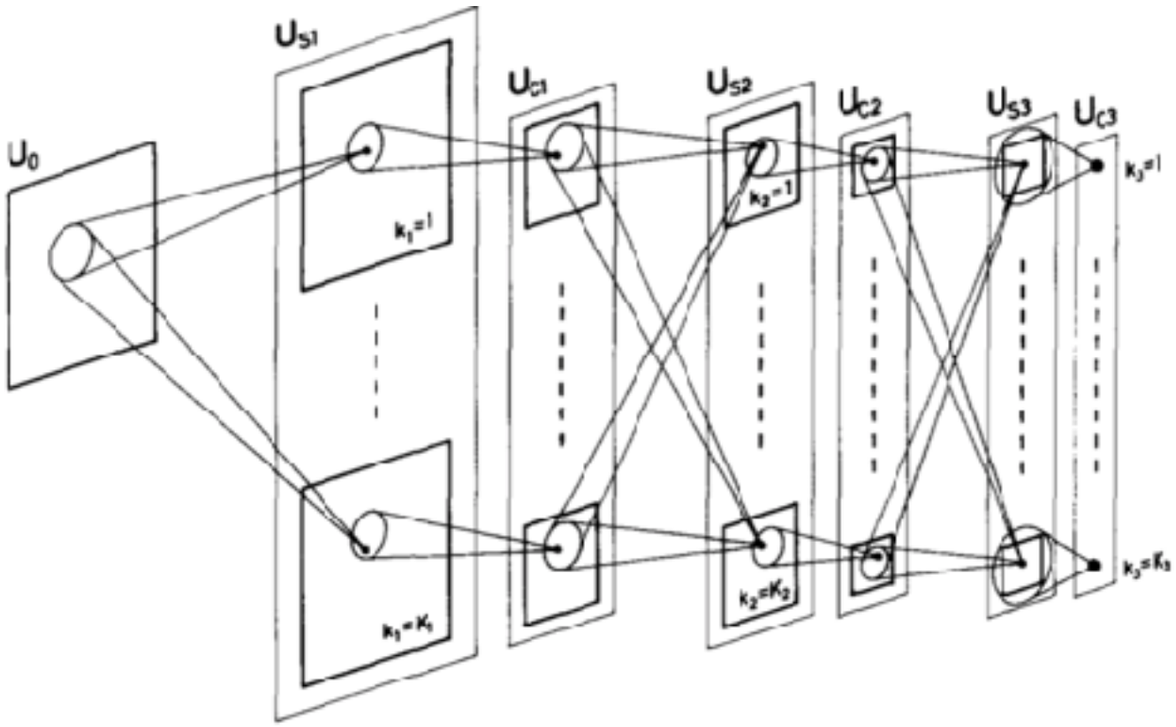


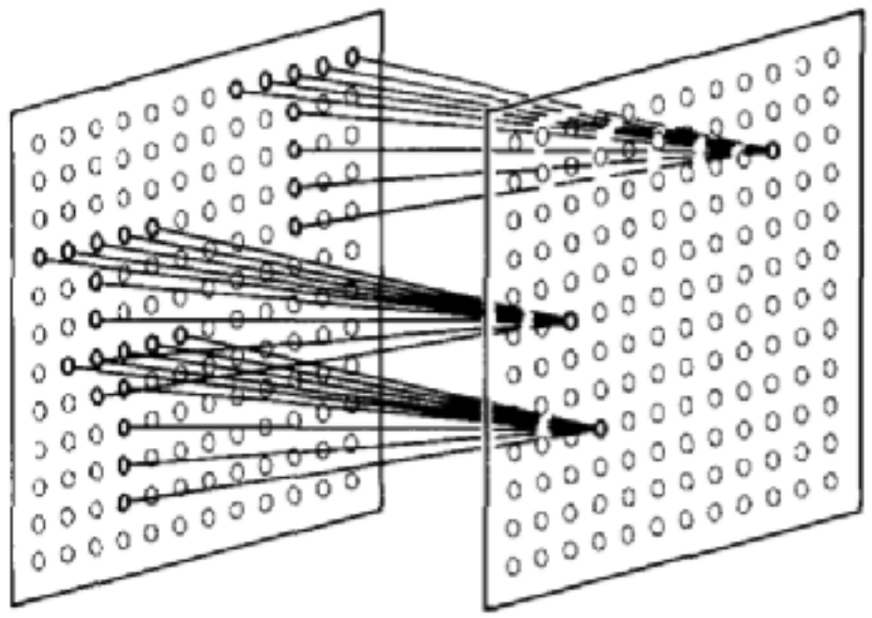
Fig. 3. Same unit as in Fig. 2. *A*, responses to shining a rectangular light spot, 1° × 8°; centre of slit superimposed on centre of receptive field; successive stimuli rotated clockwise, as shown to left of figure. *B*, responses to a 1° × 5° slit oriented in various directions, with one end always covering the centre of the receptive field: note that this central region evoked responses when stimulated alone (Fig. 2a). Stimulus and background intensities as in Fig. 1; stimulus duration 1 sec.

Hubel, David H., and Torsten N. Wiesel. "Receptive fields of single neurones in the cat's striate cortex." The Journal of physiology 148.3 (1959): 574-591.

Introducción

- Neocognitron [Fukushima 1980] fue la primer rede neuronal en implementar los descubrimientos de Hubel & Wiesel (campo receptivo, neuronas simple y complejas).

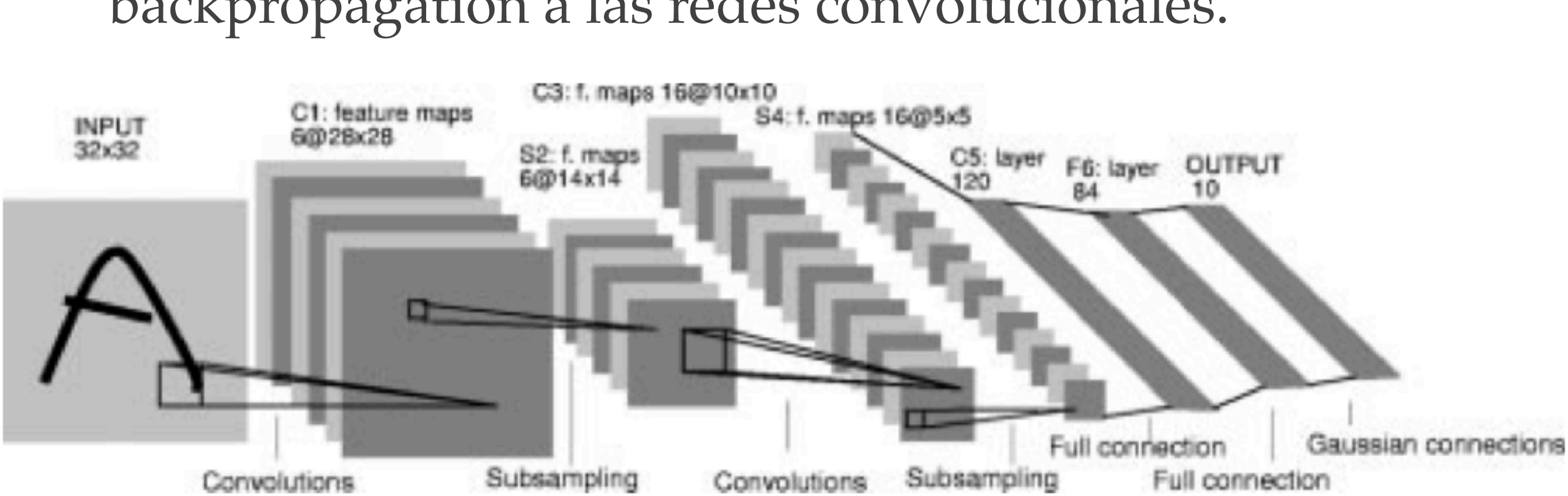




K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," Biological cybernetics, vol. 36, no. 4, pp. 193–202, 1980.

Introducción

- Gradient-based learning applied to document recognition [LeCun, Bottou, Bengio, Haffner 1998] muestran el primer ejemplo de como aplicar backpropagation a las redes convolucionales.



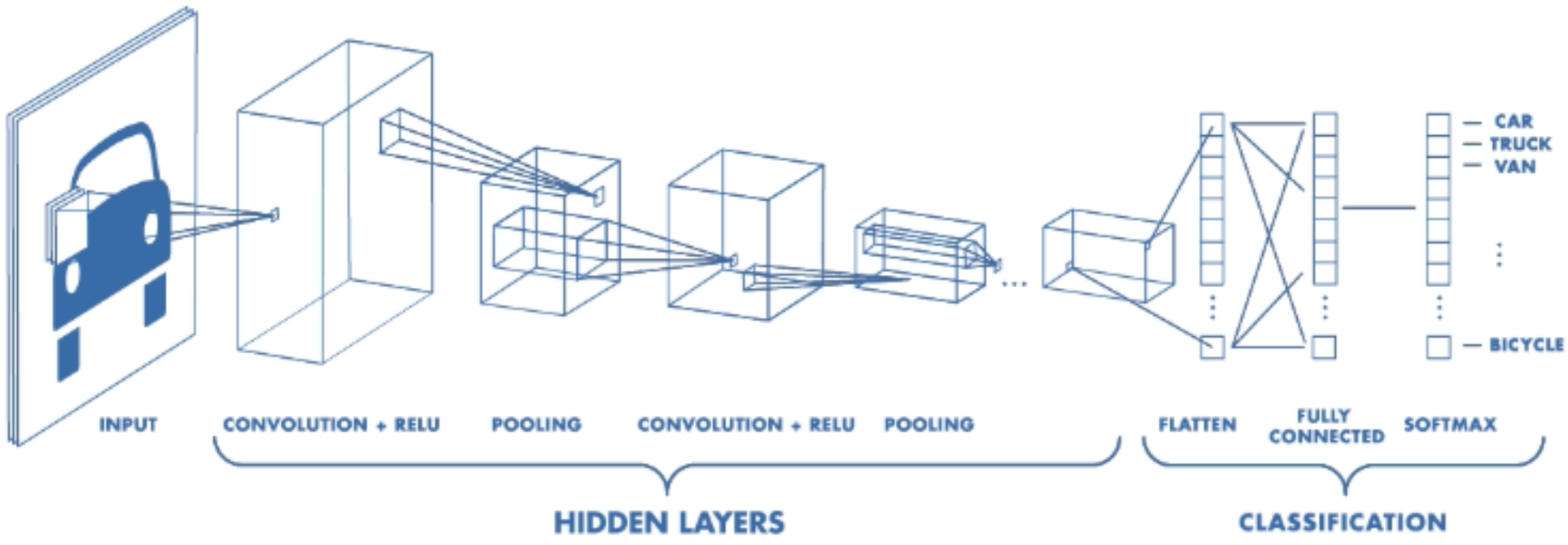
Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recogni- tion,” Proceedings of the IEEE, vol. 86, pp. 2278–2324, Nov. 1998.

Redes Convolucionales

- ¿Que ganamos?
  - Se logra reducir notablemente la cantidad de parámetros a entrenar, pero aún no se cuenta con el poder de cómputo adecuado para ser aplicarlo a problemas interesantes.
- *Deep learning* [2006s - hoy].

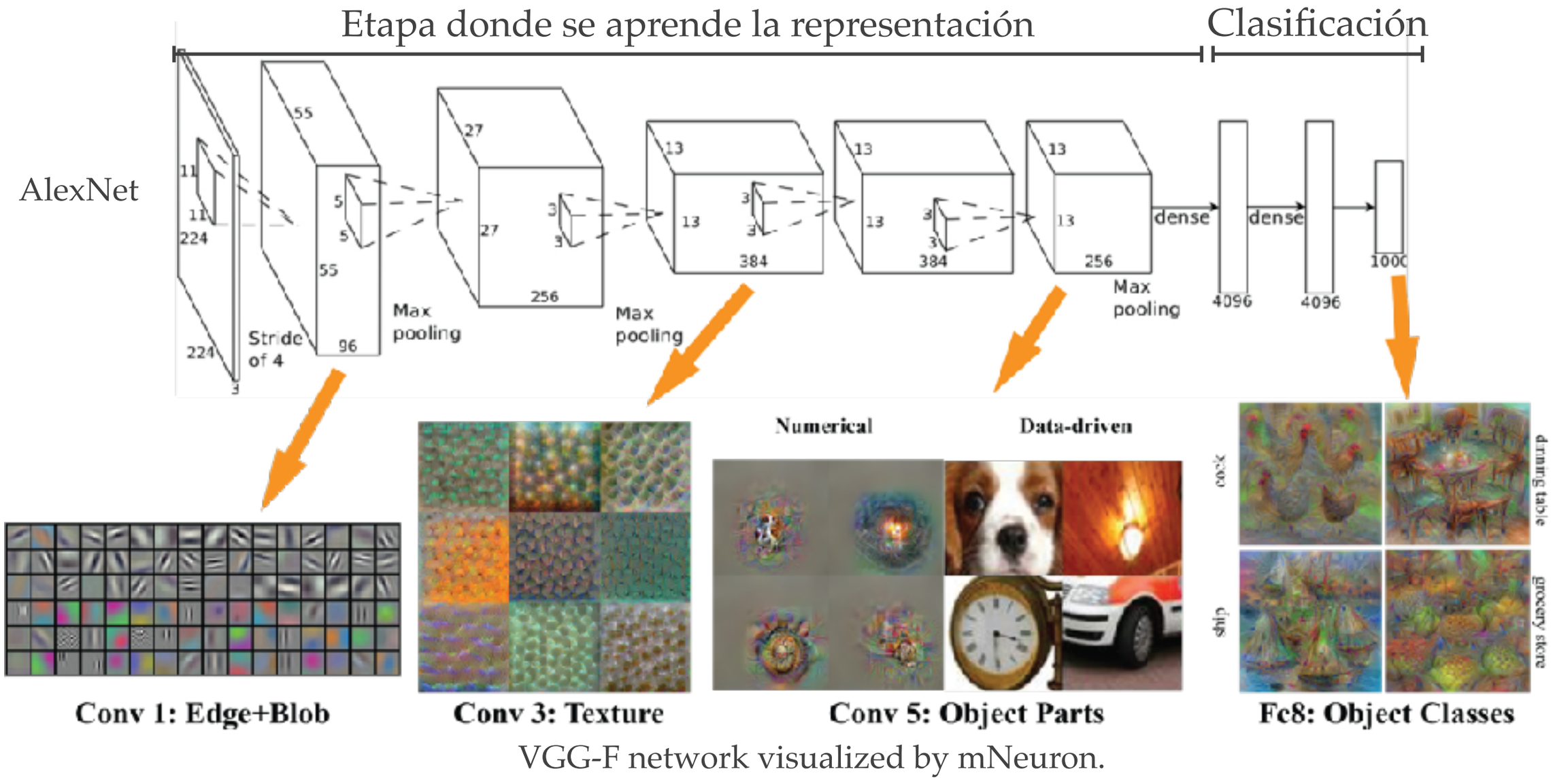
Introducción

- Convolutional Neural Networks
  - Cada neurona busca identificar en la capa anterior rasgos específicos.
  - Todas las neuronas de una capa buscan el mismo rasgo pero en una posición diferente.
- Representation learning: intentamos aprender la representación de los datos y además la relación entre esta representación y la salida (primeras clases).
  - Estas dos etapas se ven bien marcadas en CNN's



Introducción

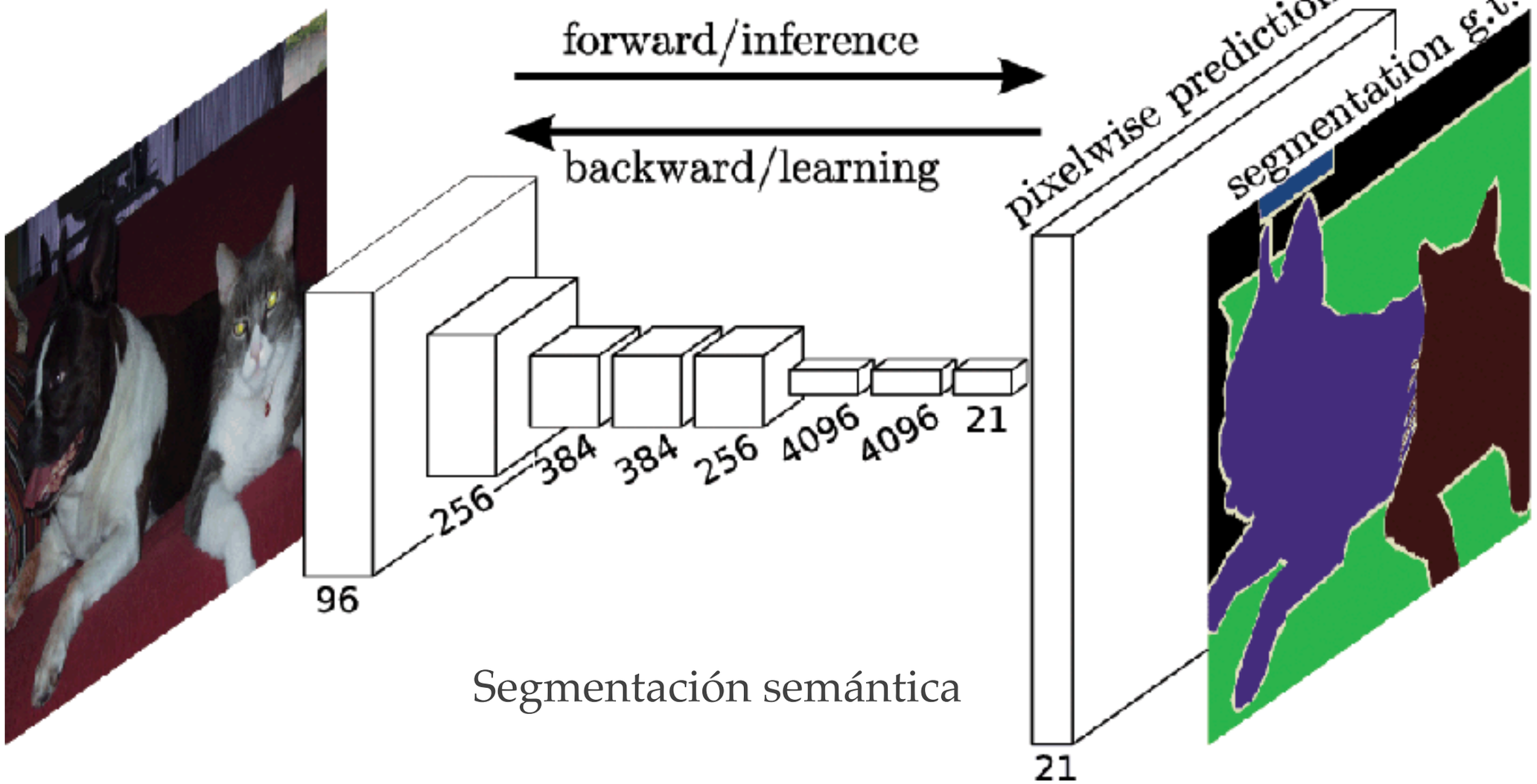
- Imagenet classification with deep convolutional neural networks [Krizhevsky, Sutskever, Hinton, 2012]



VGG-F network visualized by mNeuron.

Introducción

- Fully convolutional neural networks (FCN).

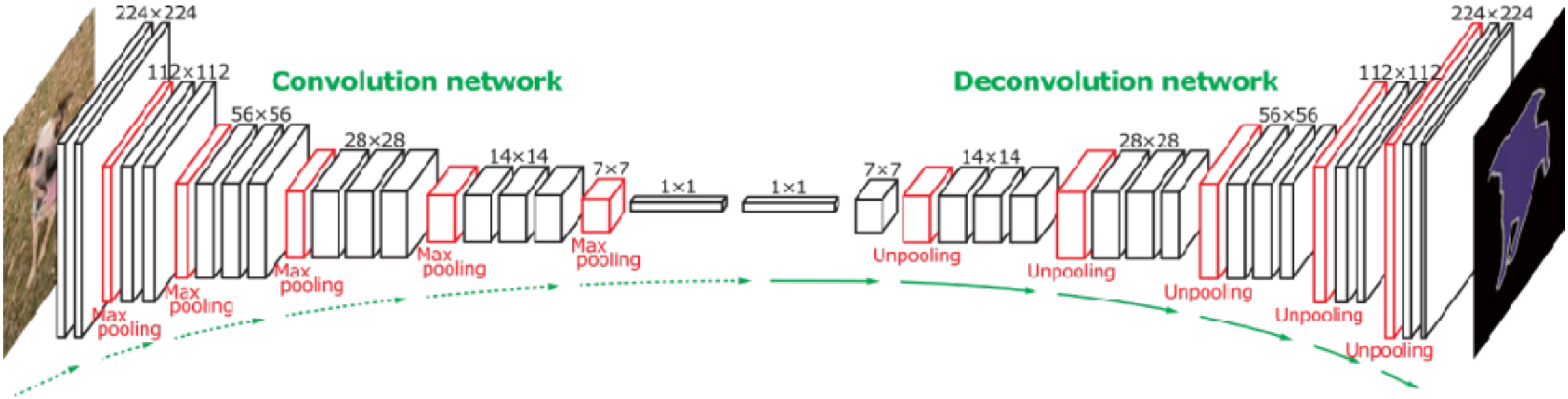


Segmentación semántica

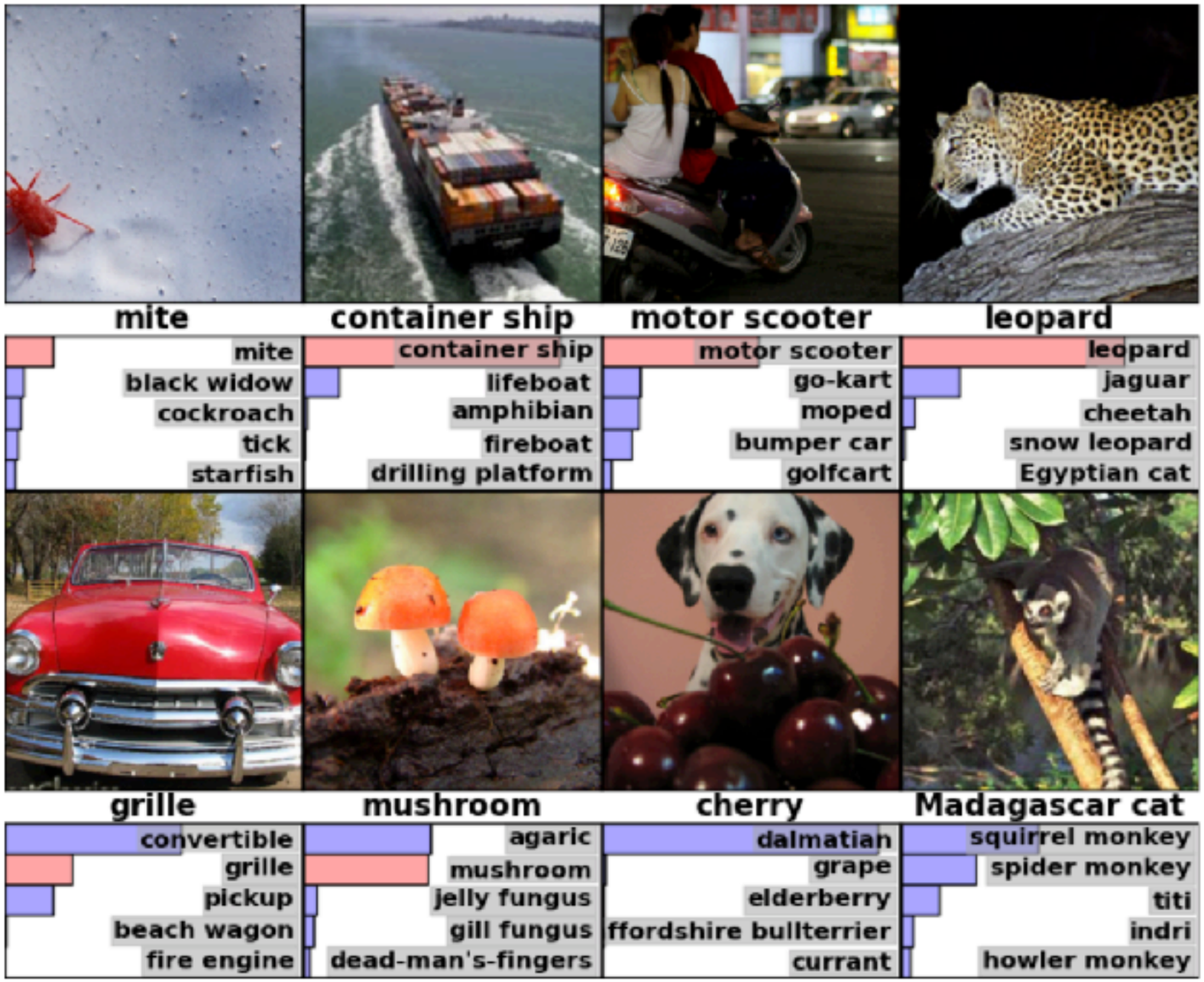
J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015.

Introducción

- Fully convolutional neural networks.

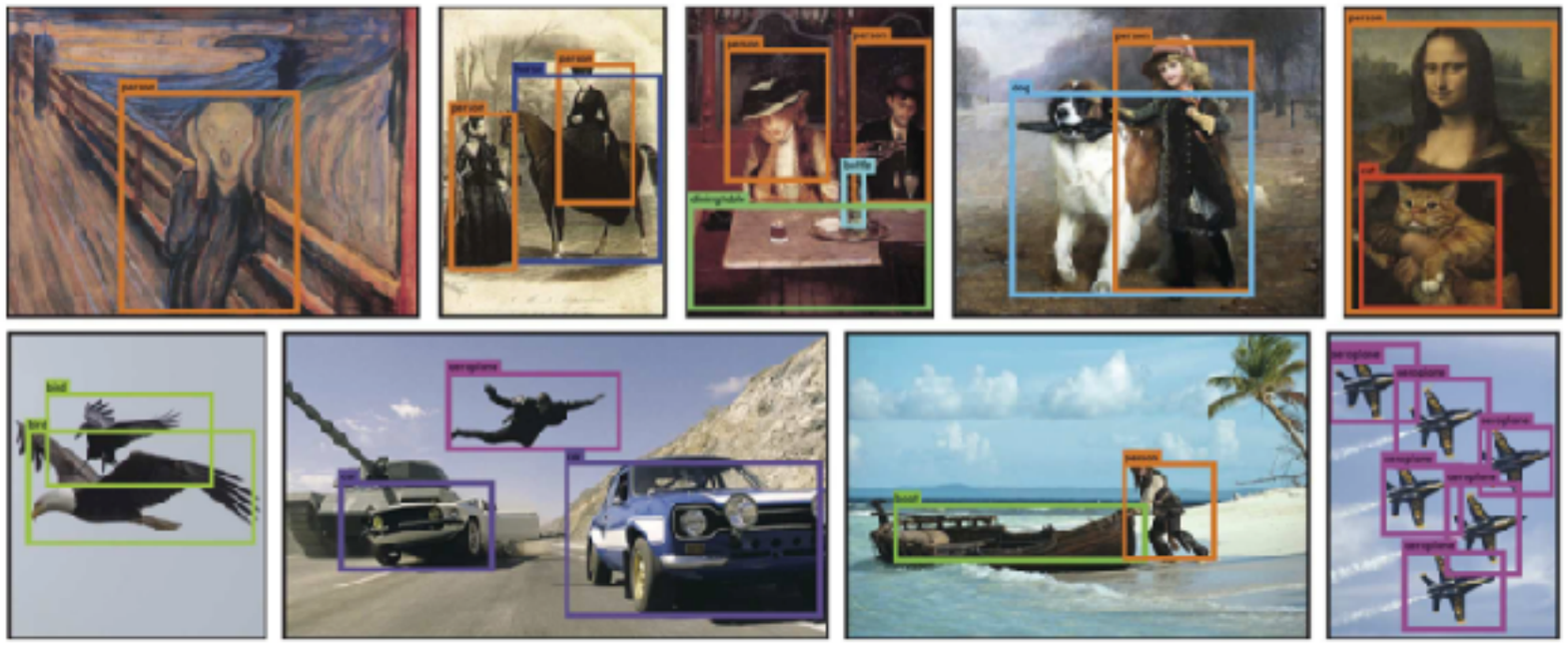


Clasificación

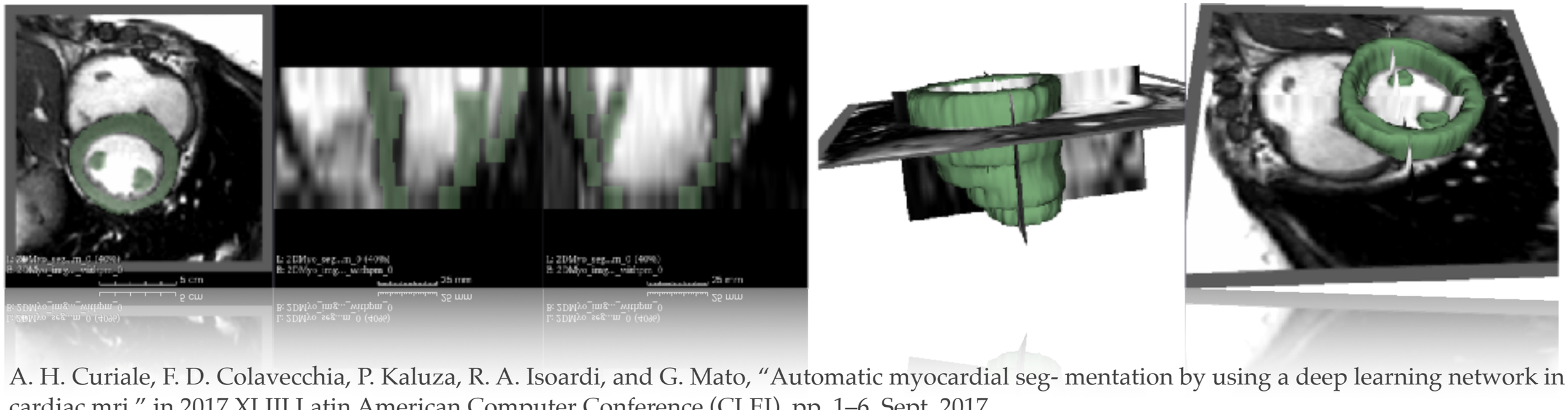


A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in neural information processing systems, pp. 1097–1105, 2012.

Detección y Segmentación

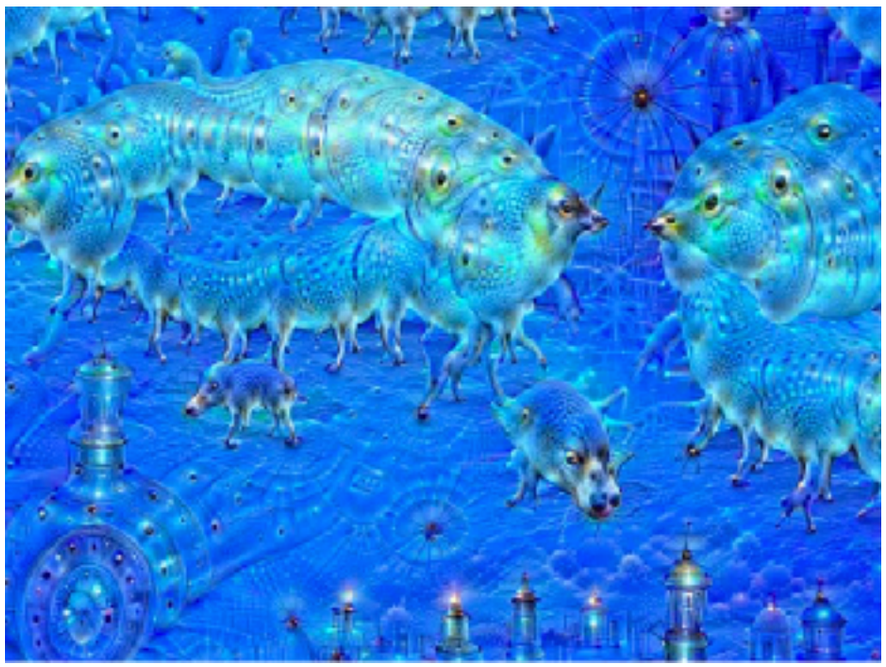


J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779– 788, 2016.



A. H. Curiale, F. D. Colavecchia, P. Kaluza, R. A. Isoardi, and G. Mato, "Automatic myocardial seg- mentation by using a deep learning network in cardiac mri," in 2017 XLIII Latin American Computer Conference (CLEI), pp. 1–6, Sept. 2017.

DeepDream



Source: Chollet F., Deep Learning with Python

Style Transfer

Source: Chollet F., Deep Learning with Python

Capa densa o totalmente conectadas

0.2	1.0	0.0	…	0.0
1.1	2.4	-2	…	-2
0.3	4	-0.5	…	-0.5
2	-0.7	30	…	30
-0.8	0.1	1.0	…	1.0

W

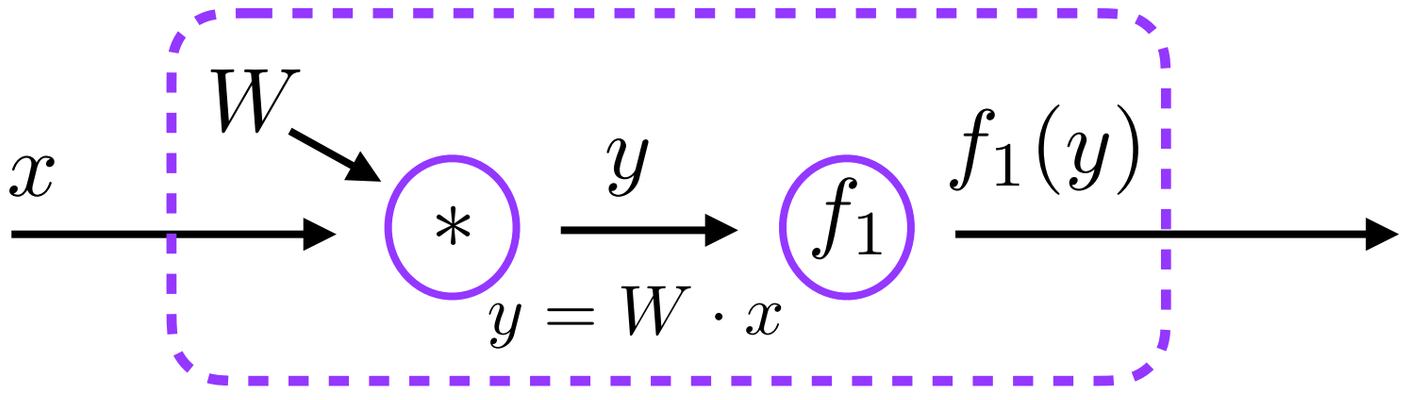
$\cdot$

1
112
123
…
45

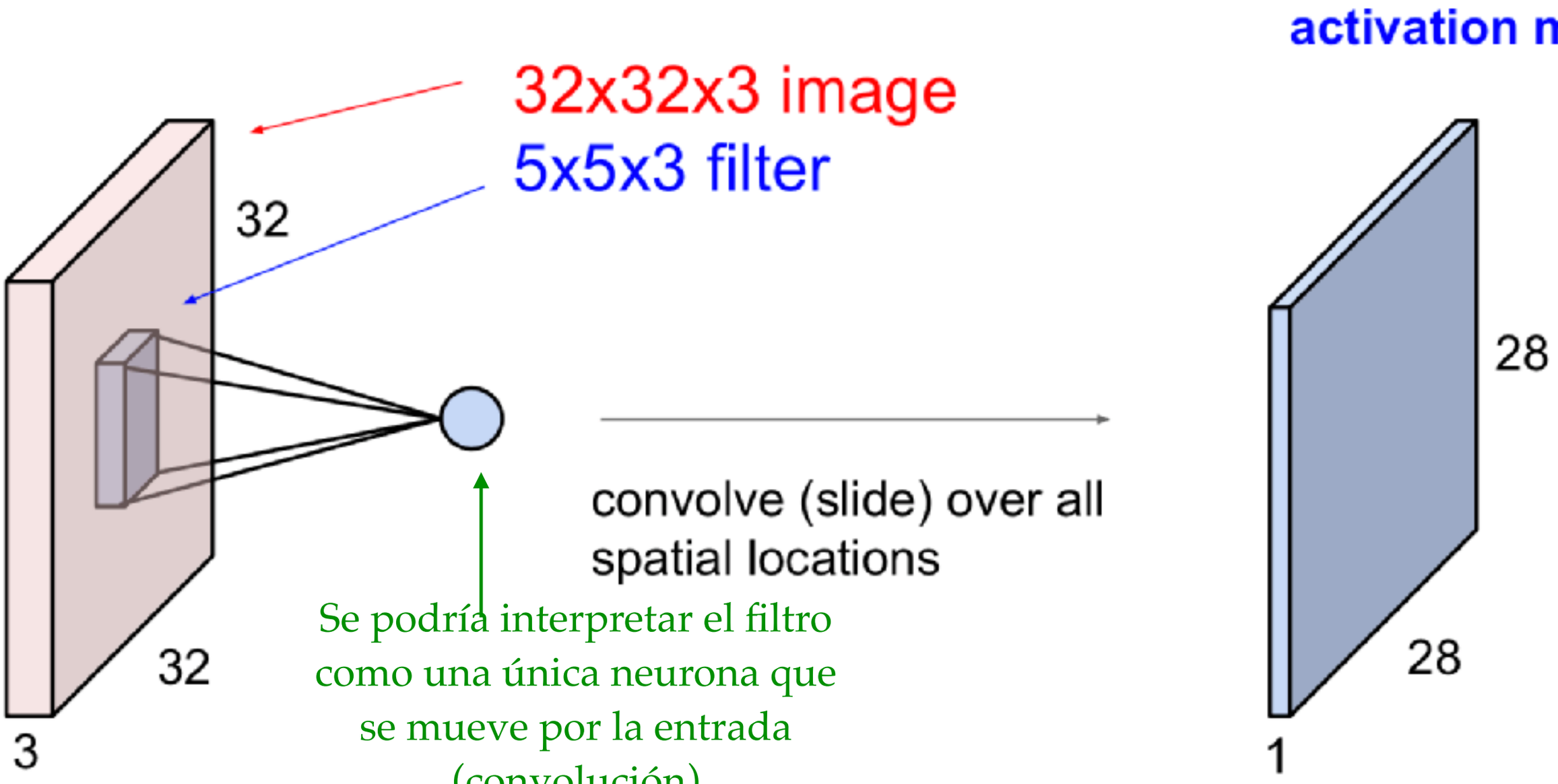
x

$=$

0.2	cat
1.1	deer
0.3	dog
2	frog
-0.8	horse



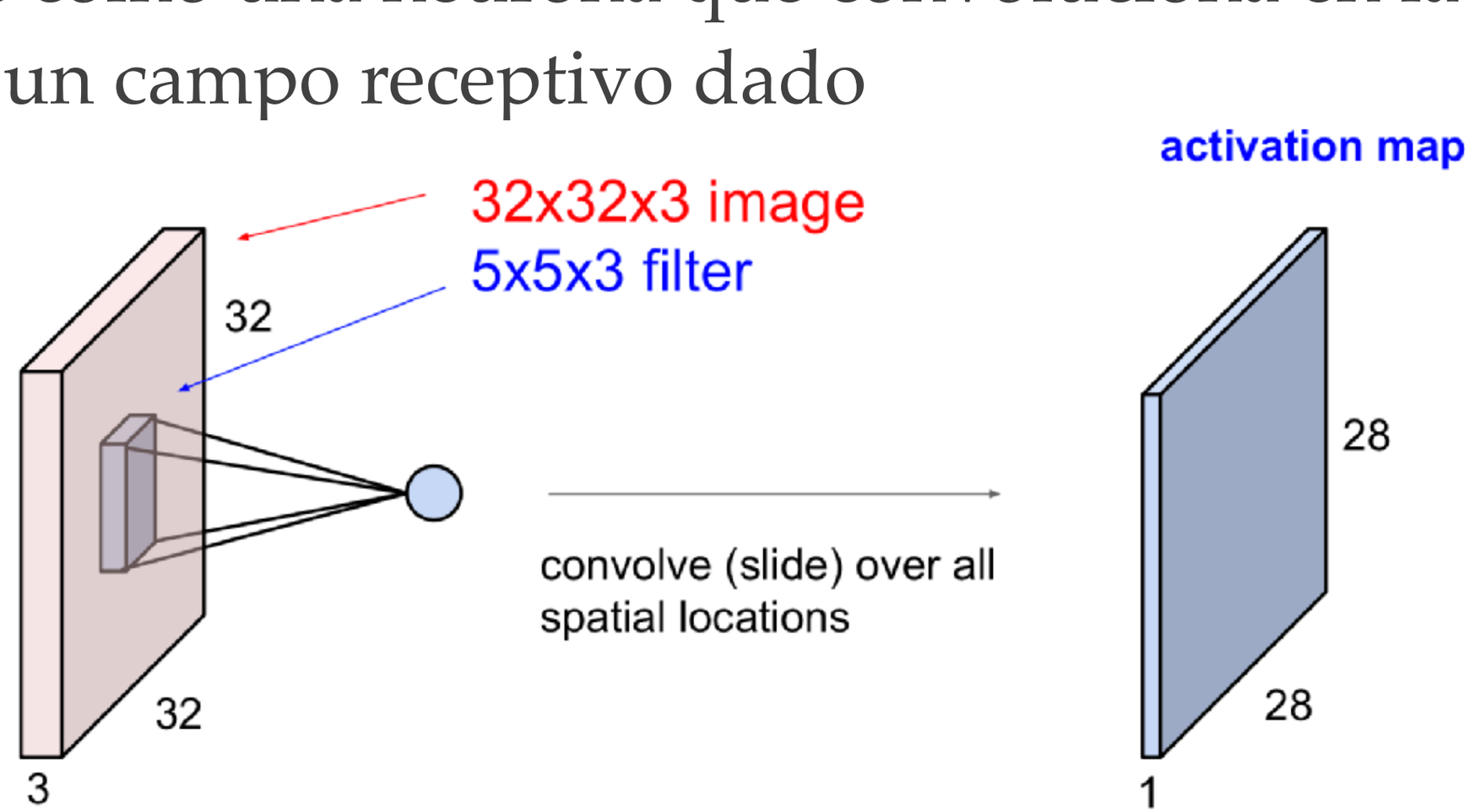
Capa convolucional



Source: Stanford CS231n 2017

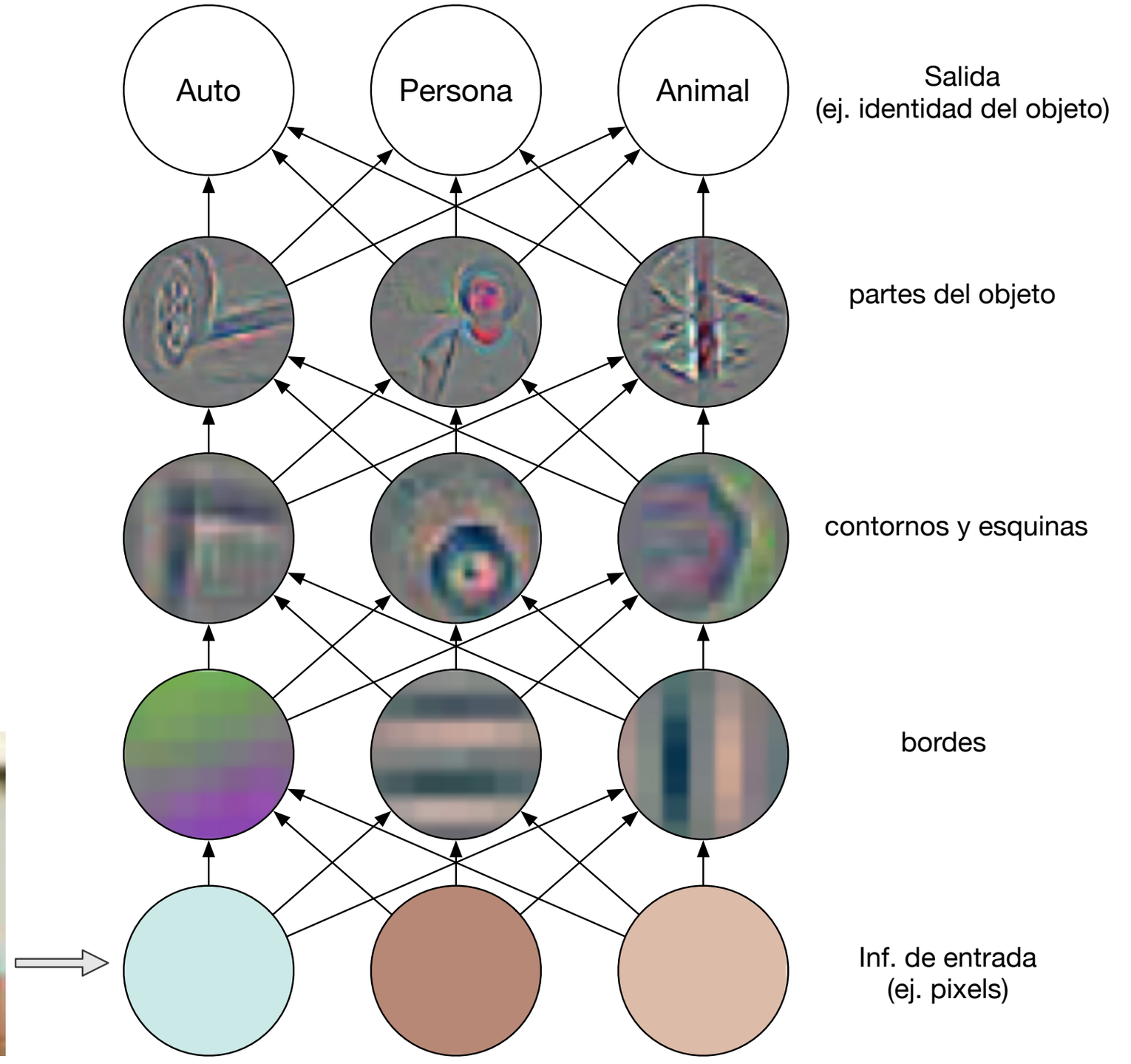
Mapa de activación de una capa convolucional

- Mapa de activación: representa la activación de las neuronas convolucional, recordar que cada filtro puede verse como una neurona que convoluciona en la entrada para un campo receptivo dado



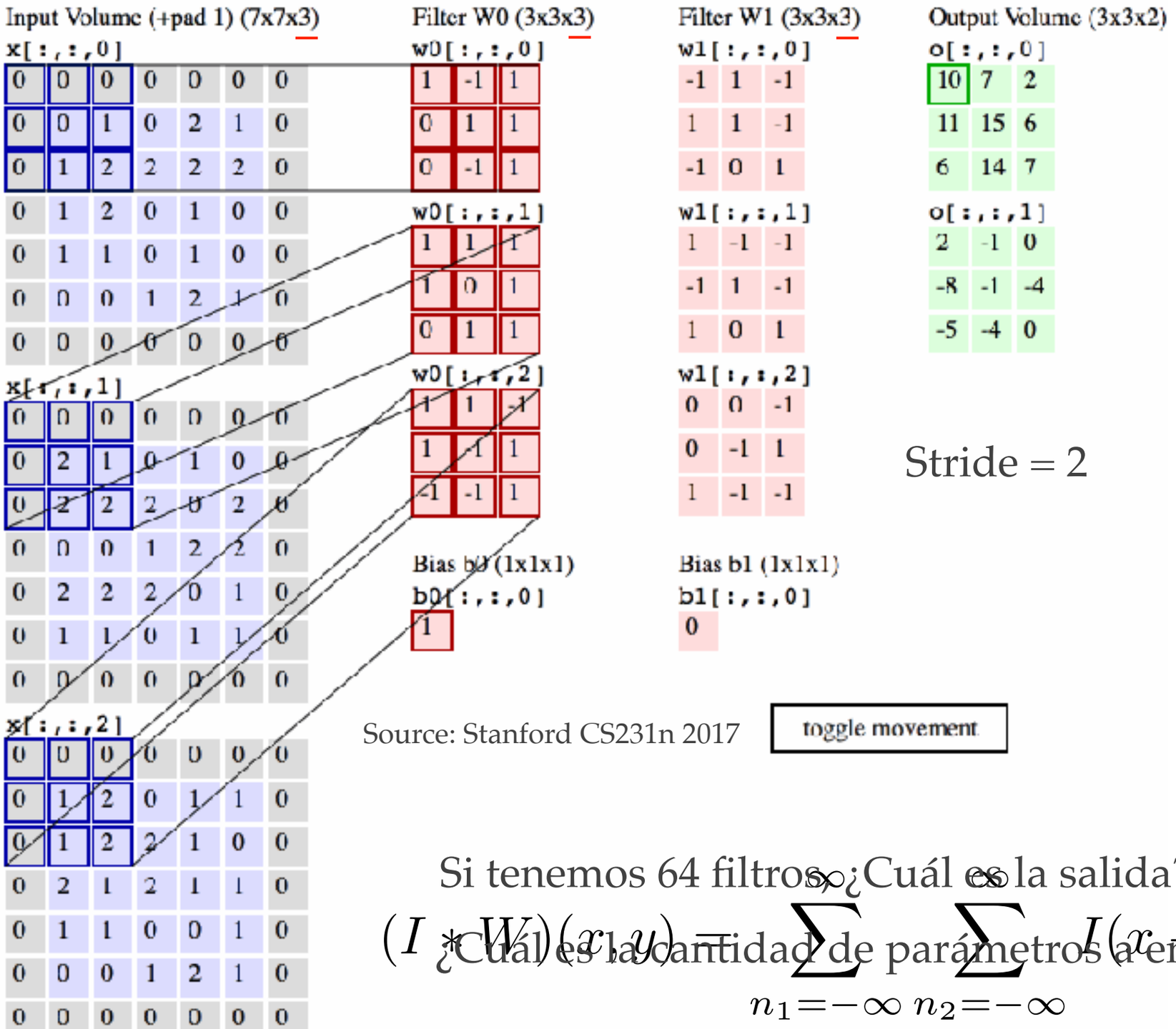
Img. Source: Stanford CS231n 2017

Características de las capas convolucionales



I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT, 2016.

Capa convolucional



Source: Stanford CS231n 2017

Tamaño de Salida:

$$\frac{N-K}{\text{stride}}+1$$

Stride = 2

Eje.:
Input: 50x50x3
K: 5x5x3
Stride: 3
Zero padding
Output: 16x16x1

Si tenemos 64 filtros, ¿Cuál es la salida?

¿Cuál es la cantidad de parámetros a entrenar?

$$(I * W)(x,y) = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} I(x-n_1, y-n_2) W(n_1, n_2)$$

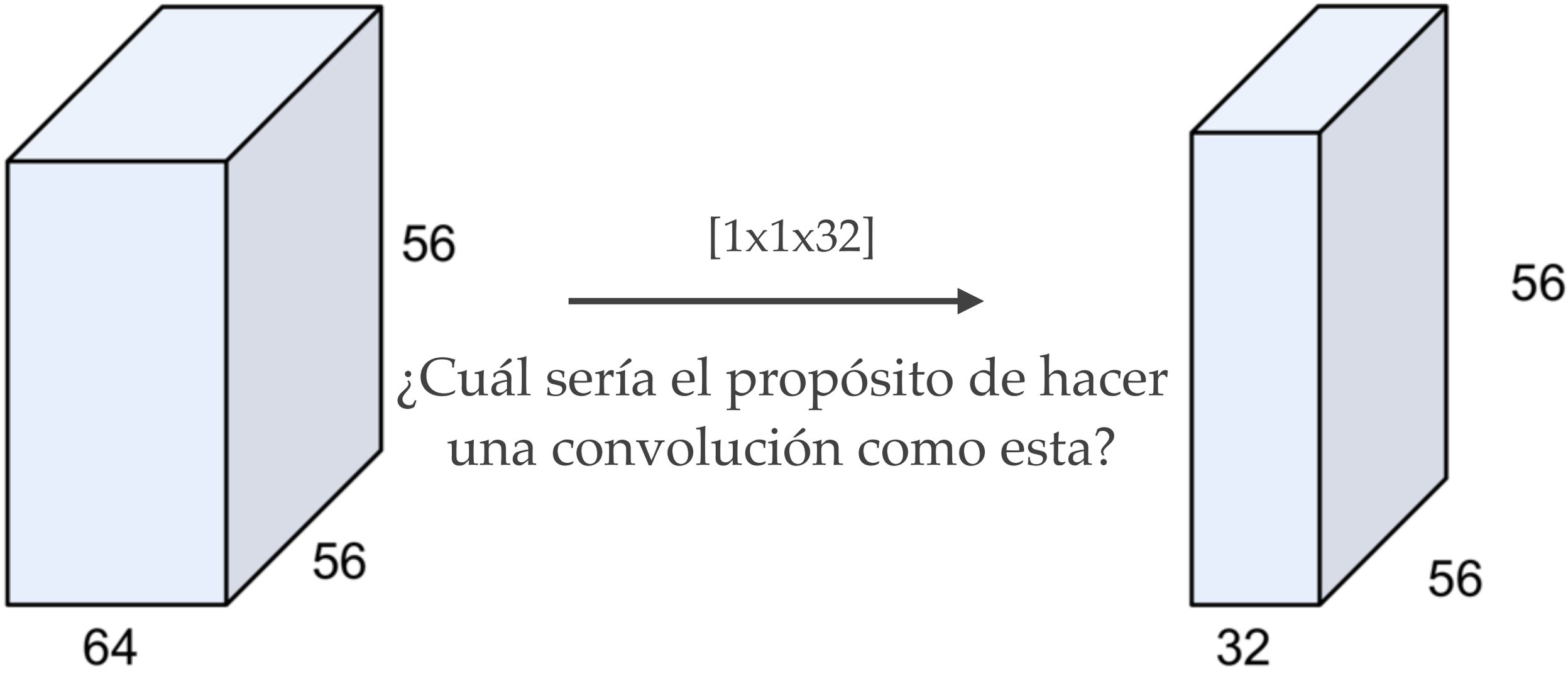
Capa convolucional

- Hiperparámetros: Tamaño del kernel, cantidad de filtros, si usa bias, la función de activación, el stride, si usa padding y cuanto, …
- Valores típicos:
  - La cantidad de filtros normal mente es potencia de 2: 32, 64, 128, 512, 1024, etc.
  - Tamaño del kernel: 1, 3, 5, 7 (normalmente impar)
  - Stride: 1, 2
  - Zero Padding: 0, 1, 2 (acorde con el tamaño del kernel)

Capa convolucional

- Cantidad de parámetros para una entrada N x M x D:
  - kernel size: K x K x D con K= 1, 3, 5, …
  - Cant. de filtros: W
  - Cant. de pesos total: (K * K * D) * W
  - Cant. de bias total: W El bias es único por filtro
- Salida de H filtros para una entrada de N x M x D con un kernel K x K x D con stride S (N y M ya contemplan el padding 2P):
  - [(N - K)/S + 1] x [(M - K)/S + 1] x H

Capa convolucional

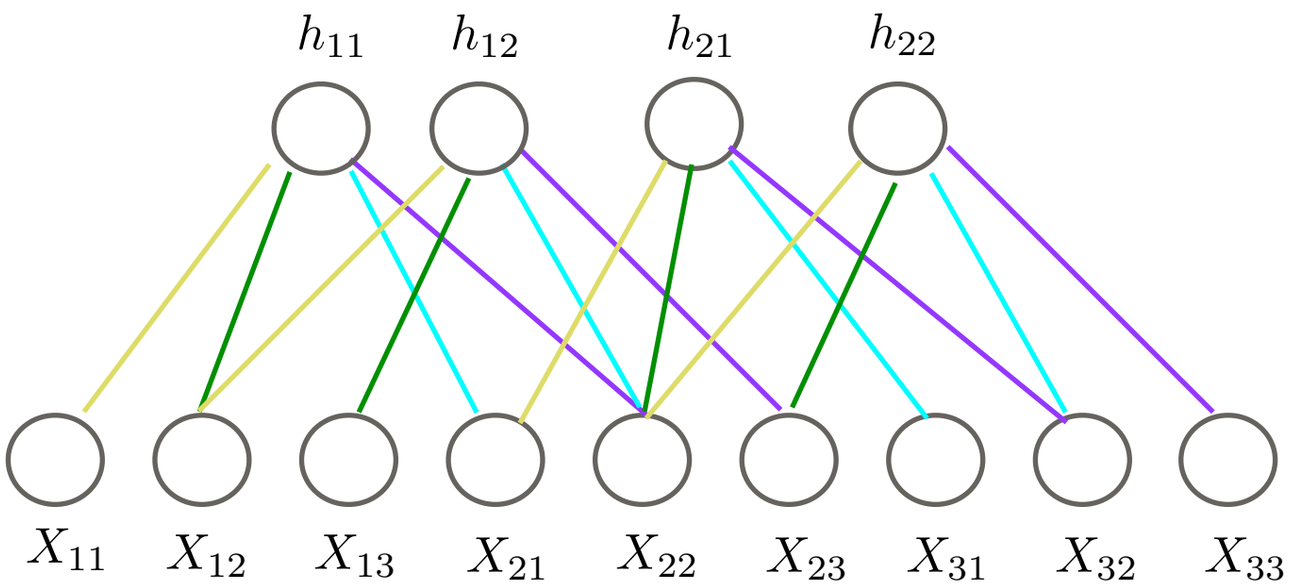


¿Cuál sería el propósito de hacer una convolución como esta?

Capa convolucional: Backpropagation

X_{11}	X_{12}	X_{13}
X_{21}	X_{22}	X_{23}
X_{31}	X_{32}	X_{33}

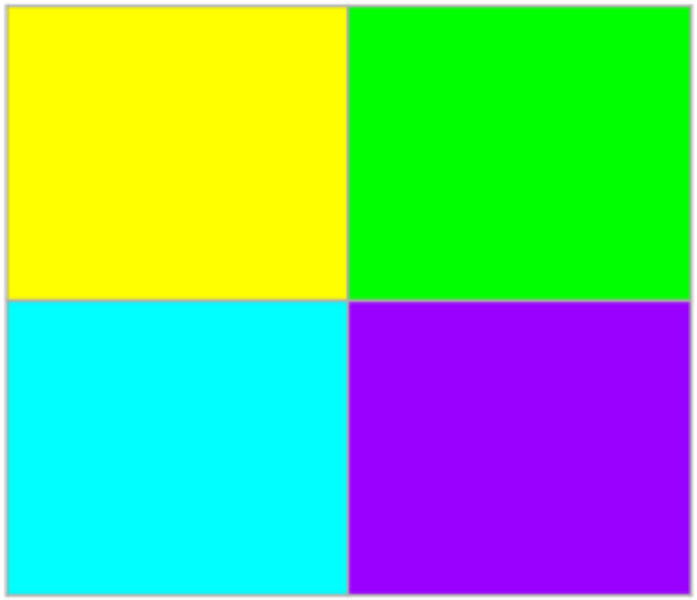
W_{11}	W_{12}
W_{21}	W_{22}



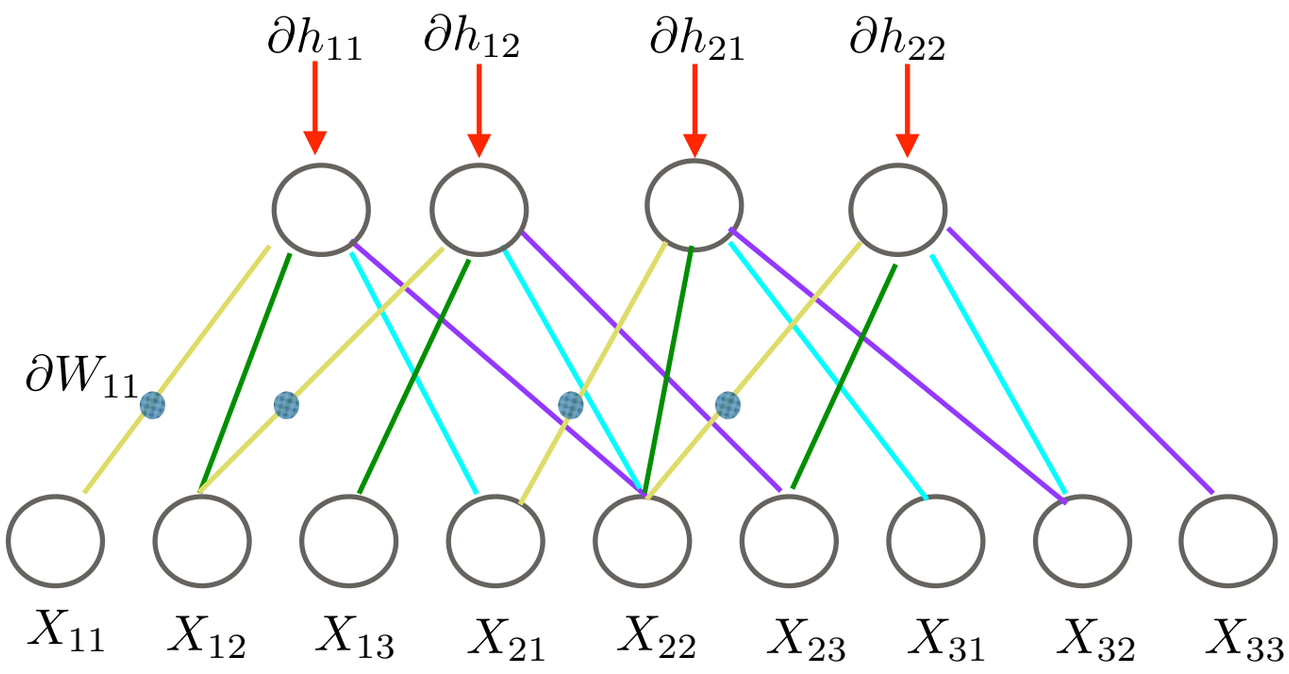
$$h_{11} = W_{11}X_{11} + W_{12}X_{12} + W_{21}X_{21} + W_{22}X_{22}$$
$$h_{12} = W_{11}X_{12} + W_{12}X_{13} + W_{21}X_{22} + W_{22}X_{23}$$
$$h_{21} = W_{11}X_{21} + W_{12}X_{22} + W_{21}X_{31} + W_{22}X_{32}$$
$$h_{22} = W_{11}X_{22} + W_{12}X_{23} + W_{21}X_{32} + W_{22}X_{33}$$

Capa convolucional: Backpropagation

X_{11}	X_{12}	X_{13}
X_{21}	X_{22}	X_{23}
X_{31}	X_{32}	X_{33}



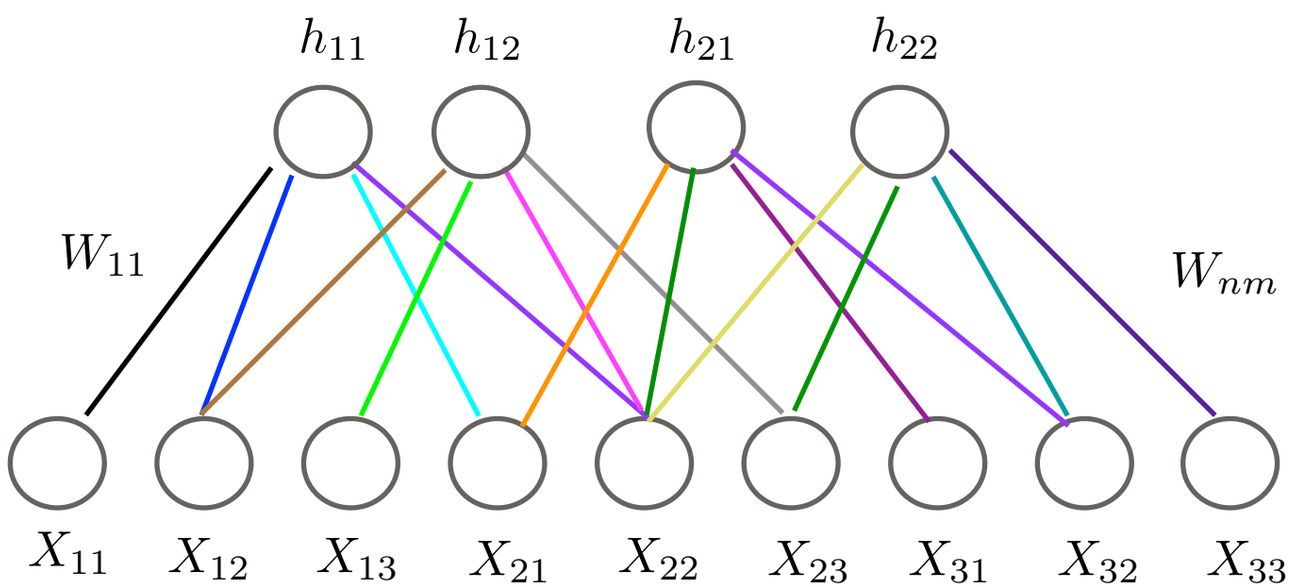
∂h_{11}	∂h_{12}
∂h_{21}	∂h_{22}



$$\partial h_{ij} = \frac{\partial L}{\partial h_{ij}}$$

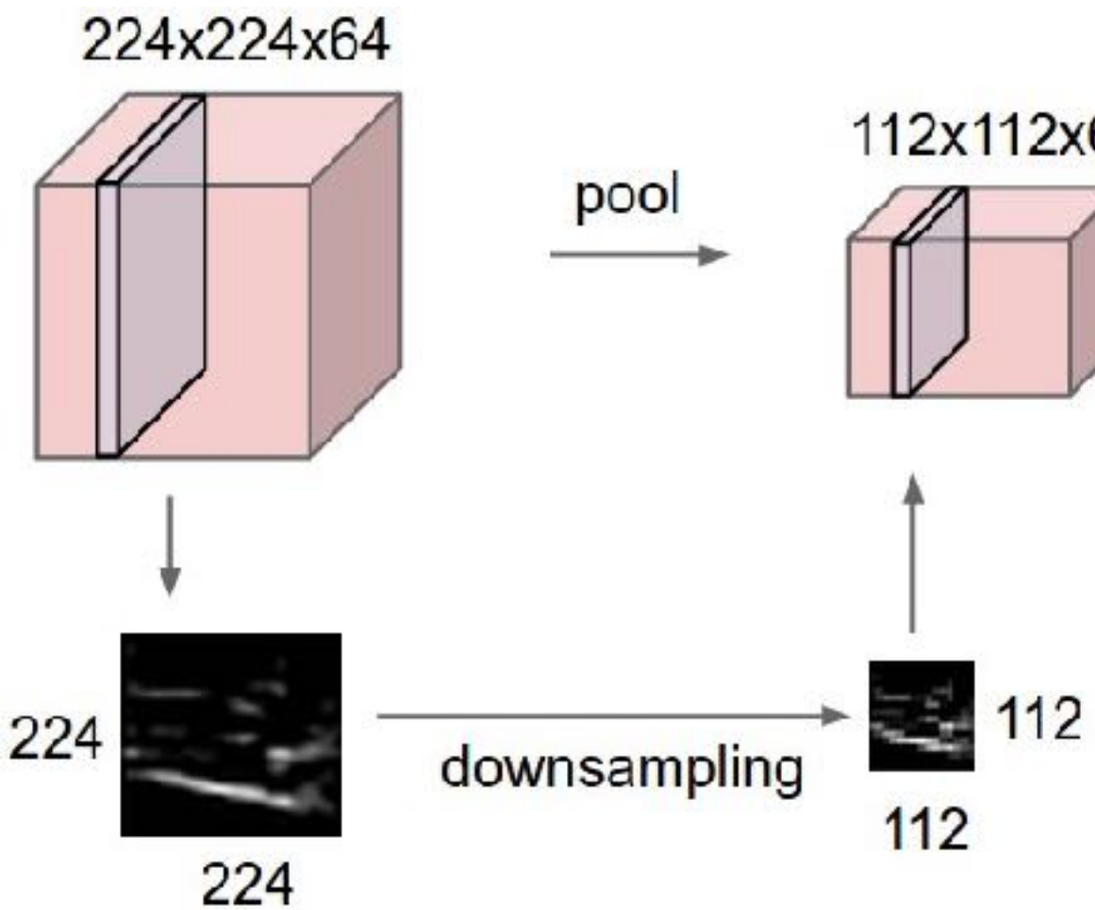
Capa Localmente conectada

No confundir esta capa con la capa convolucional. Las capas localmente conectadas no comparten los pesos como pasa en las convolucionales. Por lo tanto no se modifica el algoritmo de backpropagation

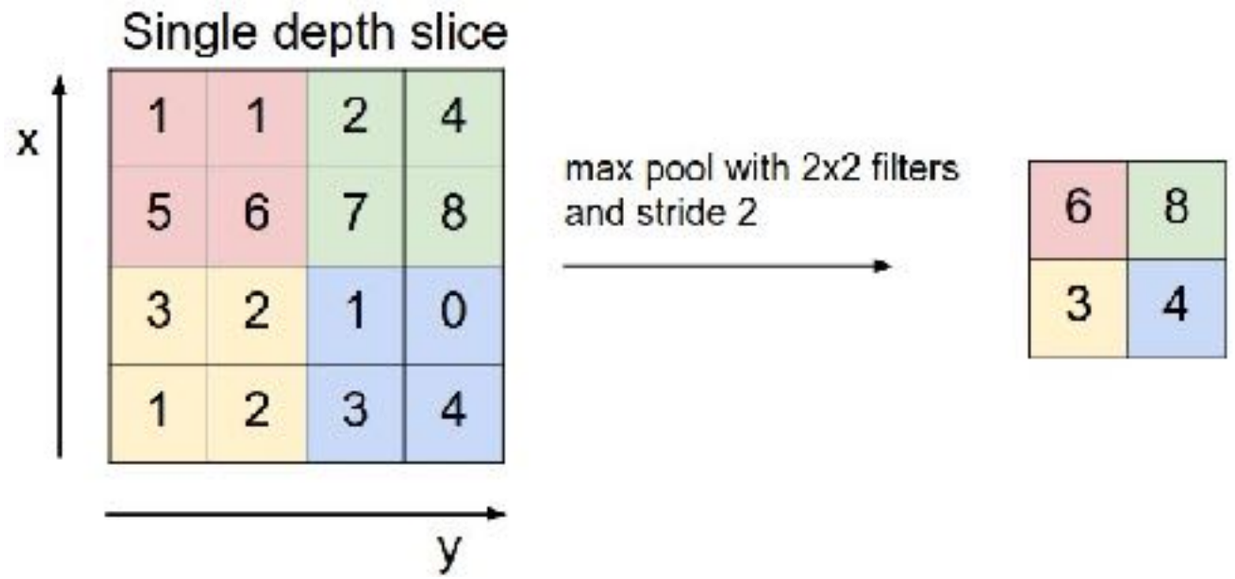


Submuestreo / Pooling

- Pool



- Max pooling



Hiperparámetros: Tamaño y Stride

Tamaño de Salida: $\frac{N-K}{\text{stride}}+1$

No es común usar Padding

¿Cantidad de parámetros a entrenar?

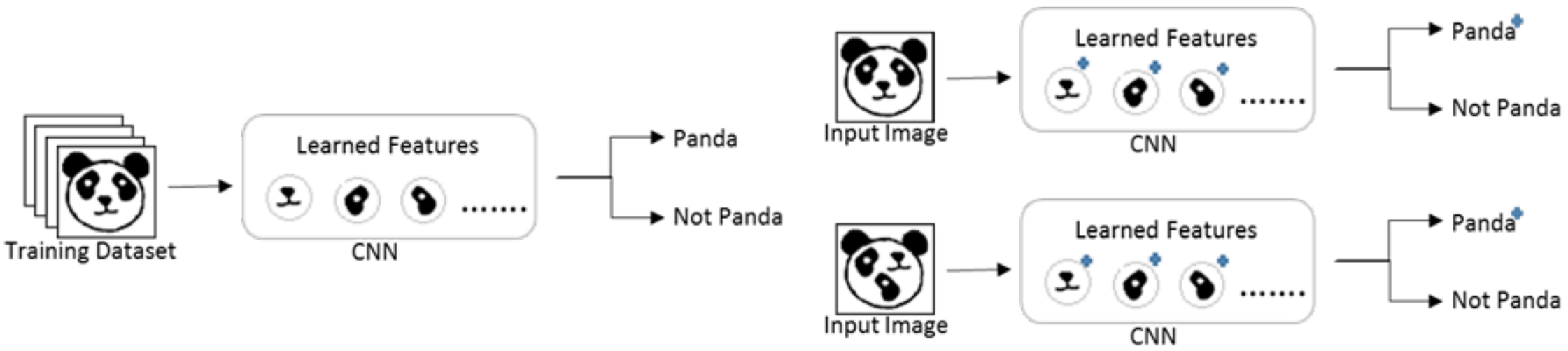
Img. source: Stanford CS231n 2017

Submuestreo / Pooling

- MaxPooling
- AveragePooling: ojo puede perder la activación al promediar
- GlobalMaxPooling: en secuencias temporales
- GlobalAveragePooling: en sec. temporales

Submuestreo / Pooling

- Pros
  - Reduce la representación de la entrada o imagen
  - Da invariancia espacial a las características
- Contras
  - Se pierde la noción espacial de las características y su relación entre ellas

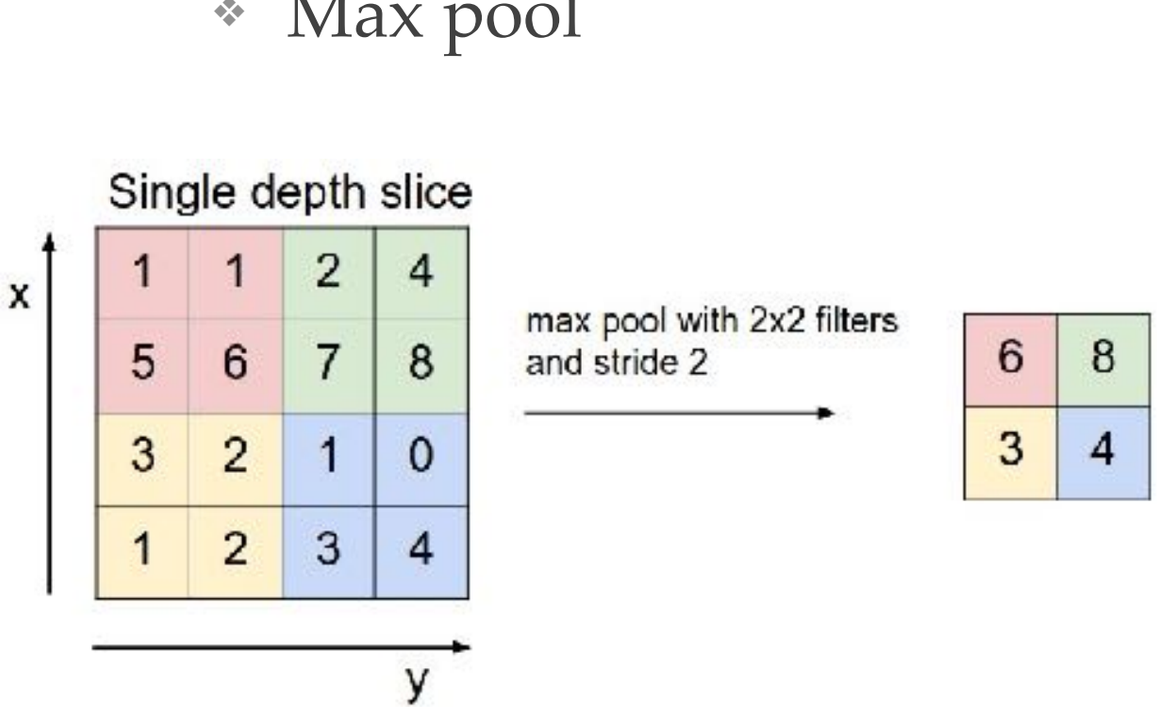


Img. source: https://becominghuman.ai/understanding-capsnet-part-1-e274943a018d

Submuestreo / Pooling

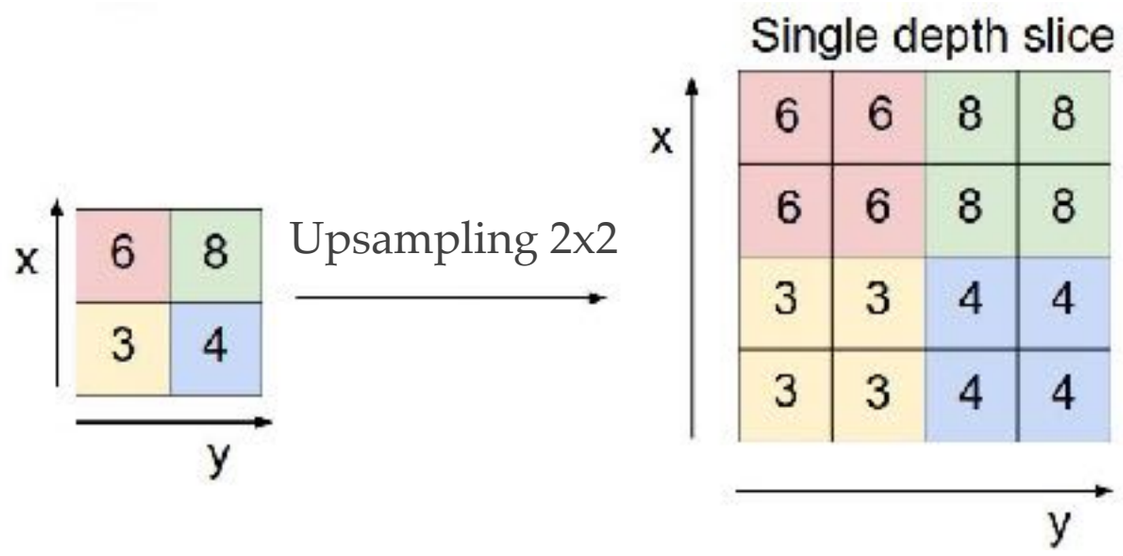
- UpSampling

  - Max pool



Img. source: Stanford CS231n 2017

  - Up Sampling



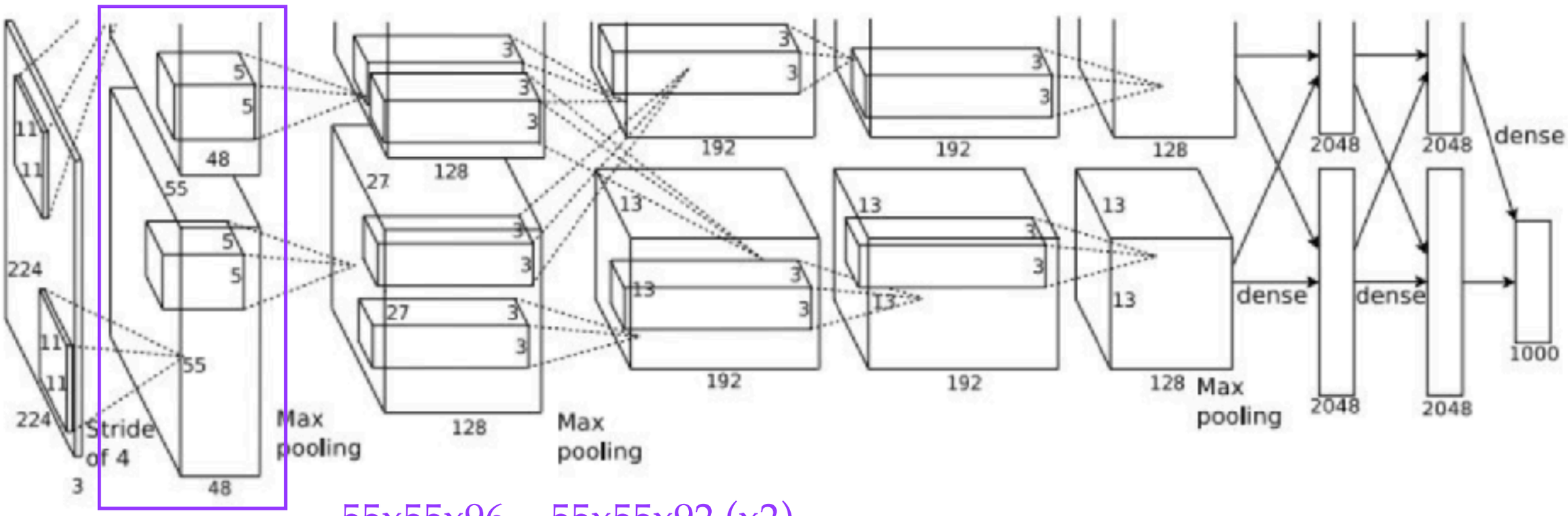
Hiperparámetros: Tamaño

Tamaño de Salida: Tamaño x Input

Casos de estudio

AlexNet

[Krizhevsky et al. 2012]

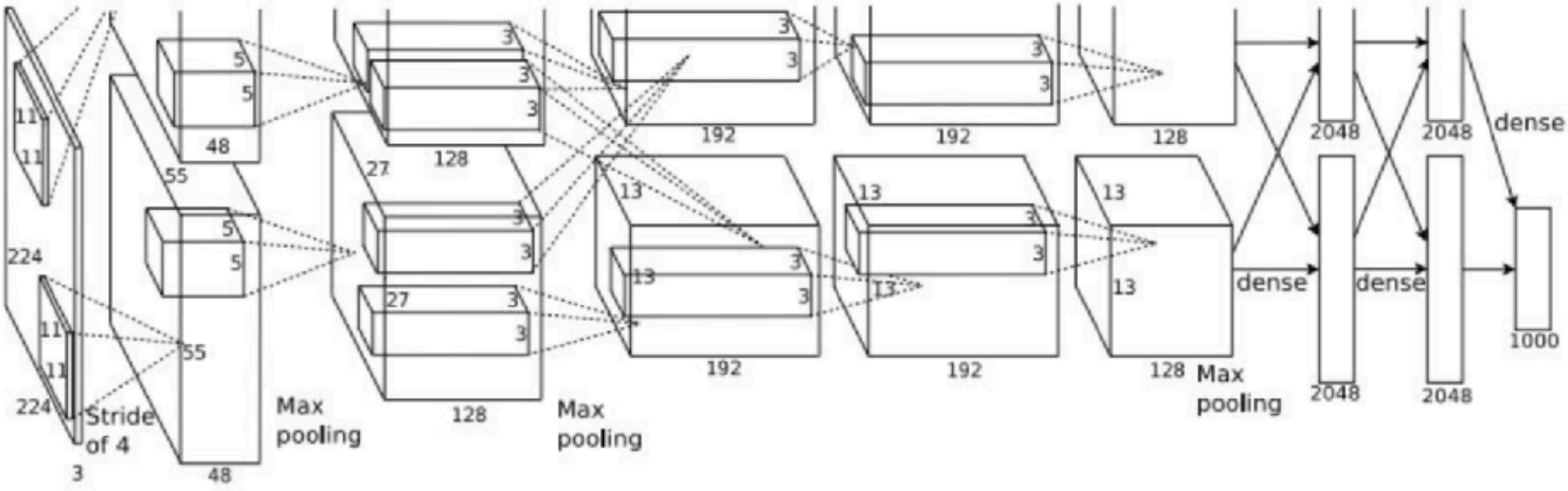


55x55x96 = 55x55x92 (x2)

- Convolución con filtros de 11x11, 5x5 y 3x3
- Primer uso de ReLU
- Fuerte utilización de aumentación de datos
- Uso de L2 como regularización

AlexNet

[Krizhevsky et al. 2012]



input: 224x224x3 y la capa de conv es de 96 filtros de 11x11 stride=4

¿Cuál es el tamaño del volumen de salida de la primer capa conv?

55x55x96

¿Cuántos parámetros a entrenar tiene? 11x11x3x96+96

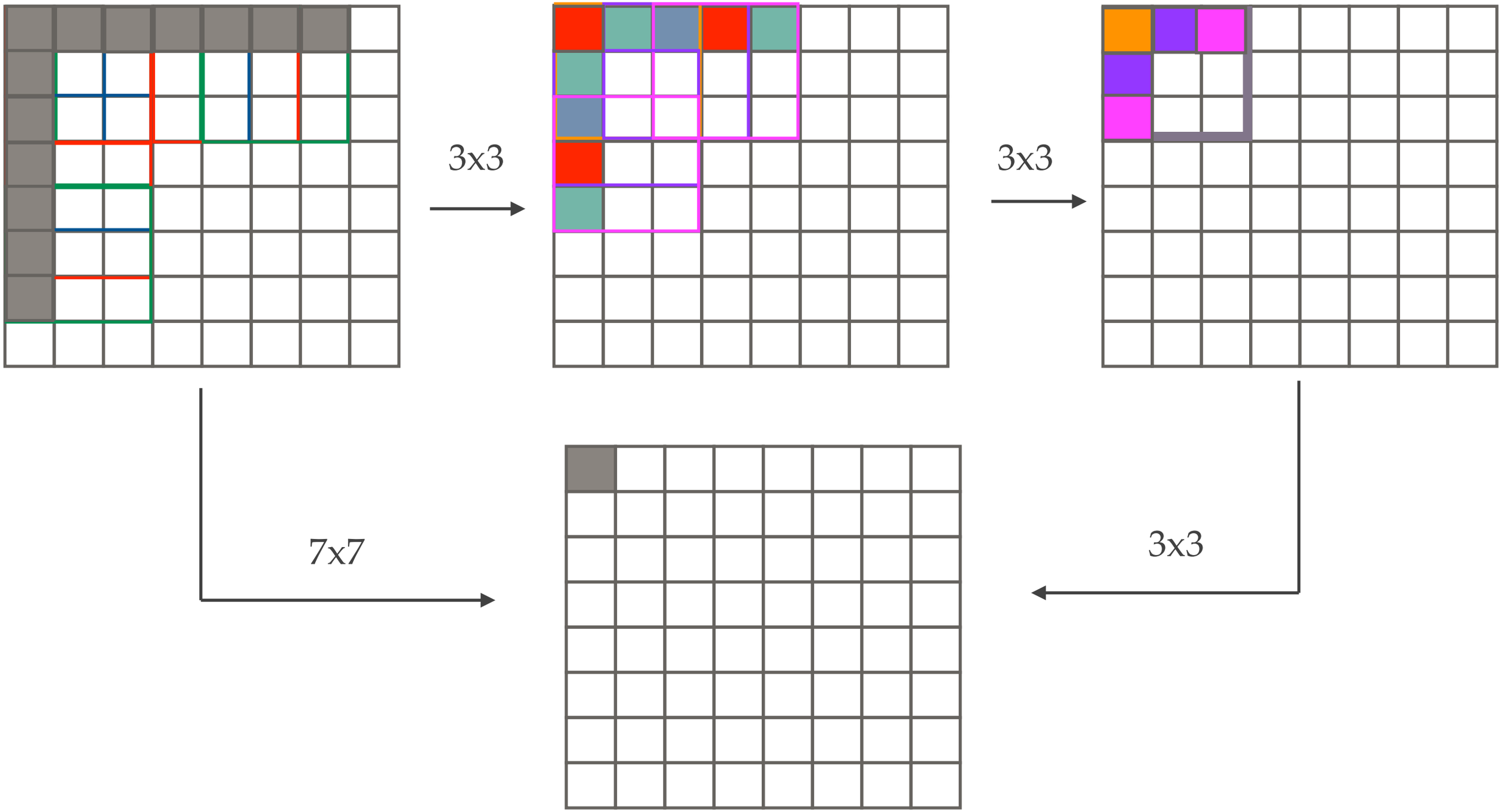
¿Y el polling de 3x3 s=2 (vol. salida y parámetros)? 27x27x96 y 0

VGGNet

[Simonyan and Zisserman, 2014]

- Filtros más pequeños
- Redes más profundas
- Juntar 3 filtros de 3x3 con un stride de 1 dan un campo receptivo efectivo similar al de una conv. de 7x7

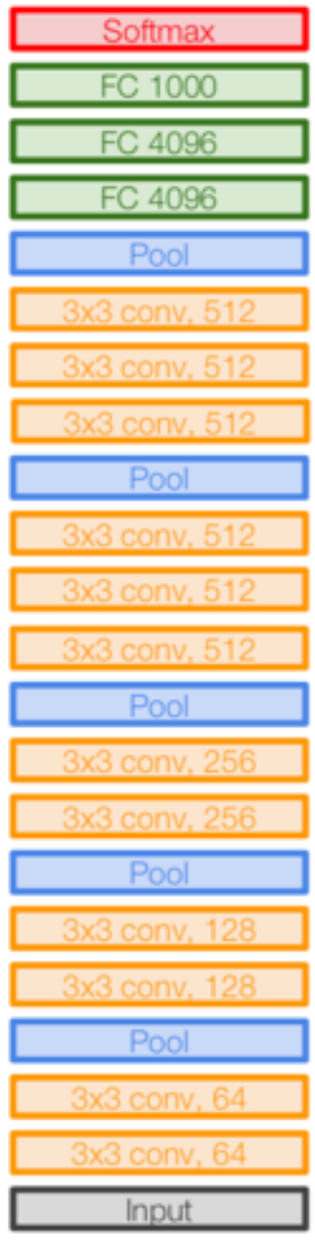
VGGNet



VGG Net: Memoria

INPUT: [224x224x3] memory: 224*224*3=150K params: 0 (not counting biases)
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K
CONV3-512: [14x14x512] memory: 14*14*512=100K
CONV3-512: [14x14x512] memory: 14*14*512=100K
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

TOTAL memory: 24M * 4 bytes ~= 96MB / image
TOTAL params: 138M parameters



VGG16

VGG Net: Memoria

INPUT: [224x224x3] memory: 224*224*3=150K params: 0 (not counting biases)
CONV3-64: [224x224x64] memory: **224*224*64=3.2M** params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: **224*224*64=3.2M** params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K
CONV3-512: [14x14x512] memory: 14*14*512=100K
CONV3-512: [14x14x512] memory: 14*14*512=100K
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = **102,760,448**
FC: [1x1x4096] memory: 4096 params: 4096*4096 = **16,777,216**
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

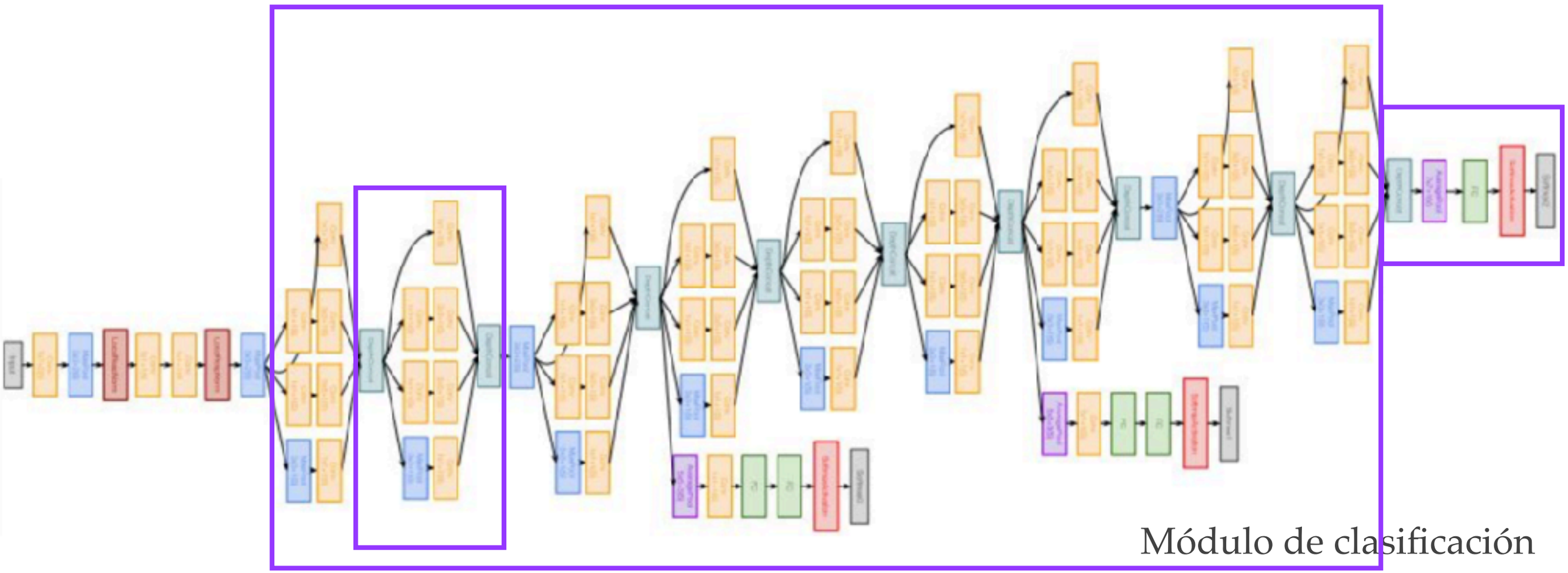
TOTAL memory: 24M * 4 bytes ~= 96MB / image
TOTAL params: 138M parameters

Mayor cantidad de memoria en las capas iniciales de conv.

Mayor cantidad de parámetros en las capas FC

GoogLeNet (Inception)

[Szegedy et al., 2014]



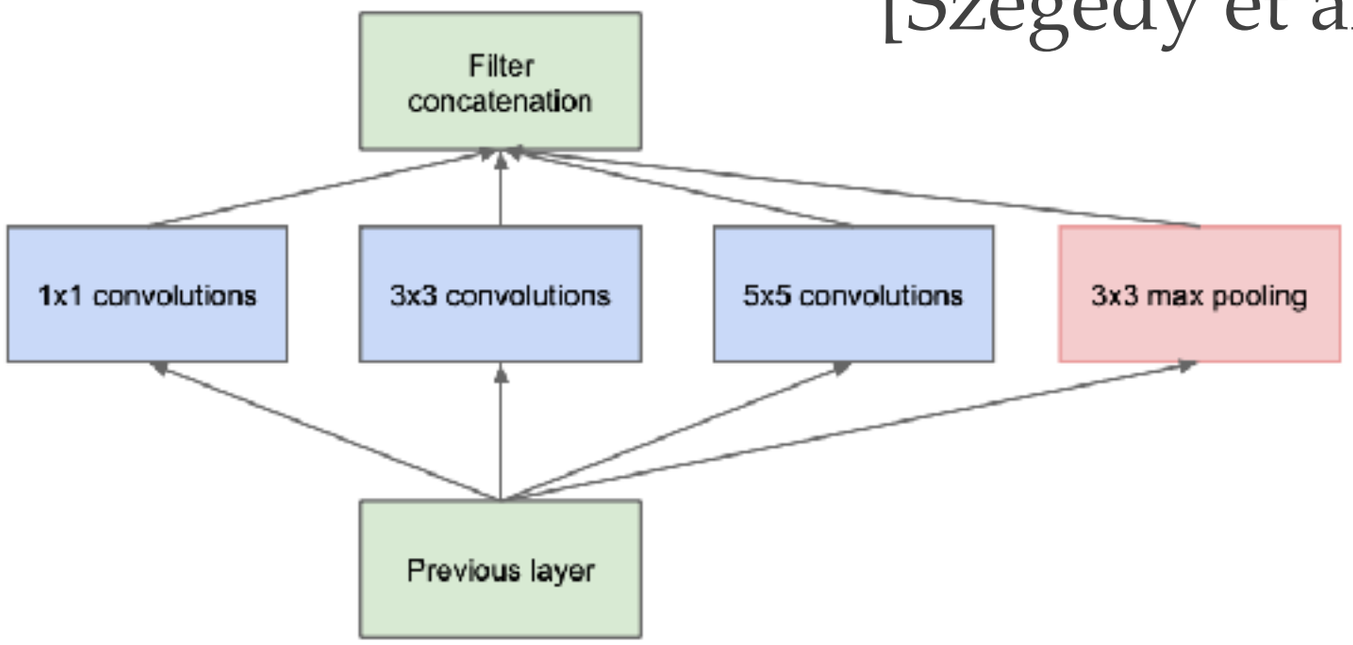
Módulo Inception

9 módulos Inception

Image Source: Stanford CS231n 2017

GoogLeNet (Inception)

[Szegedy et al., 2014]

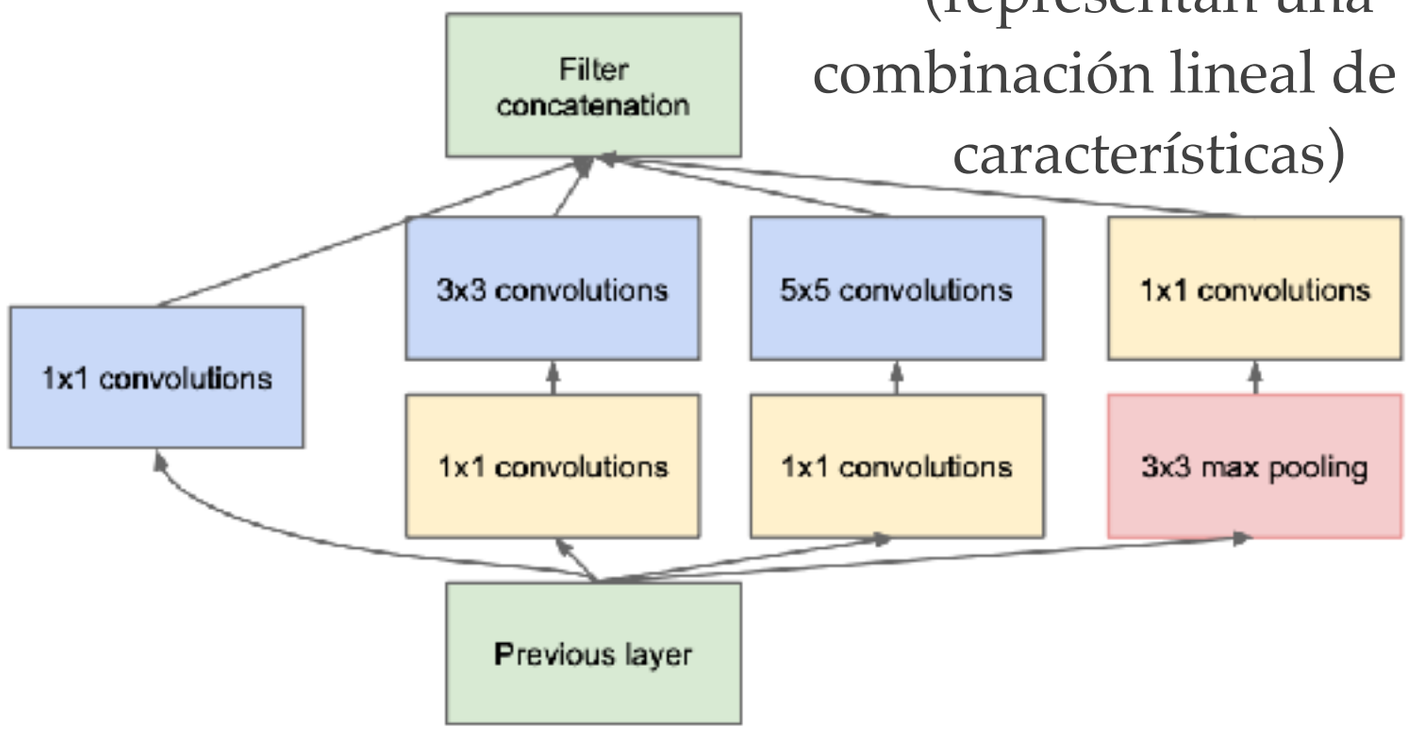


Módulo Inception (naive)

Computación muy costosa (considerar una entrada de 28x28x256 y calcular cuantas operaciones son necesarias)

Conv y Max pooling preservan el input (strides = 1)

1x1 conv. sin activación (representan una combinación lineal de las características)

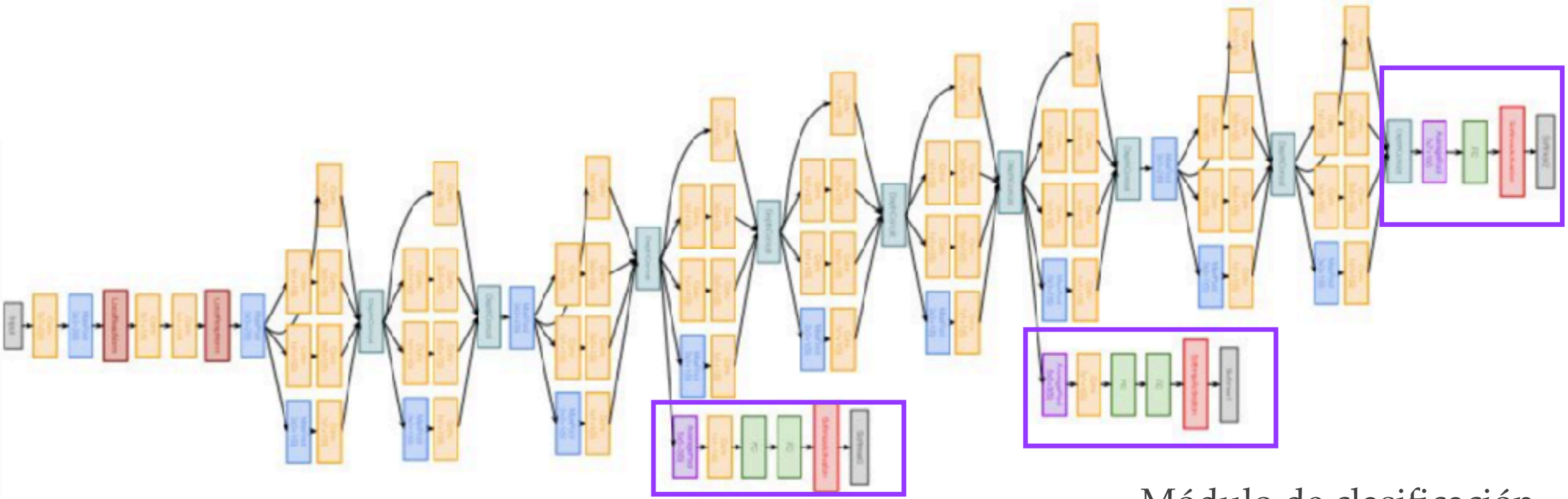


Módulo Inception con reducción de la dimensionalidad

Image Source: [Szegedy et al., 2014]

GoogLeNet (Inception)

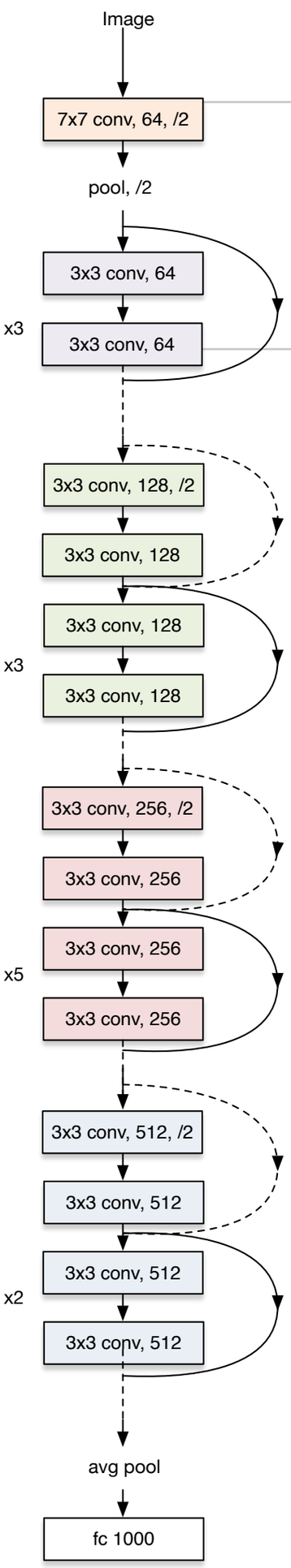
[Szegedy et al., 2014]



Módulo de clasificación

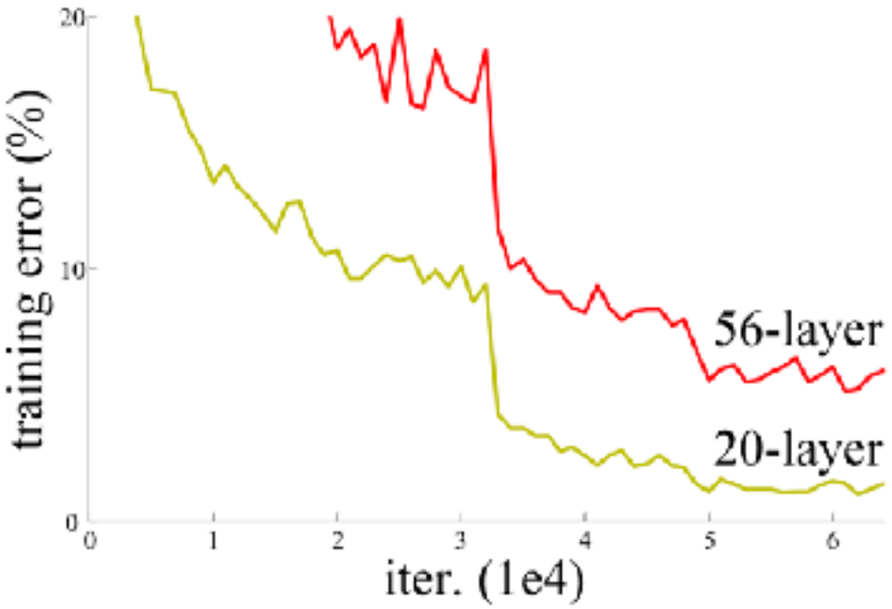
Estrategia de reinyección del gradiente mediante una clasificación adicional

Image Source: Stanford CS231n 2017



ResNet

[He et al., 2016]



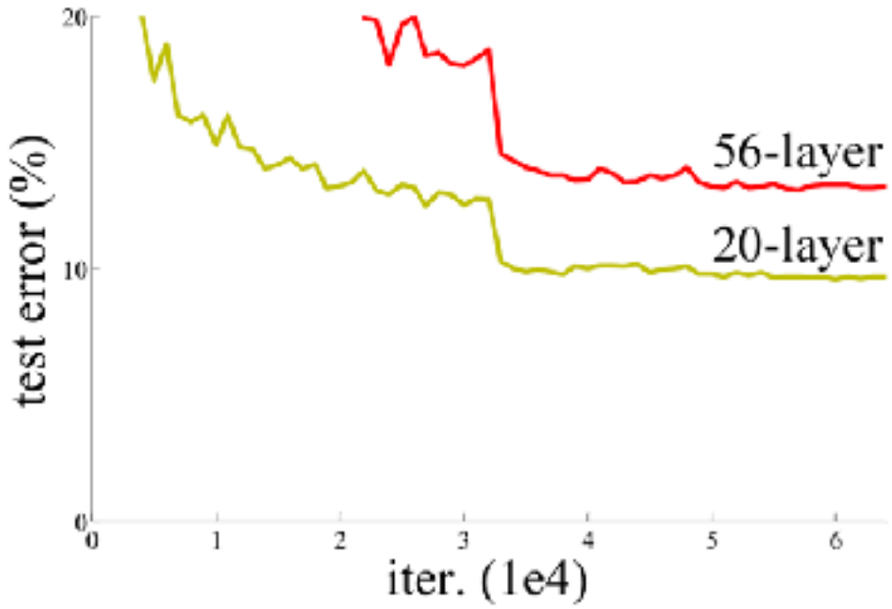
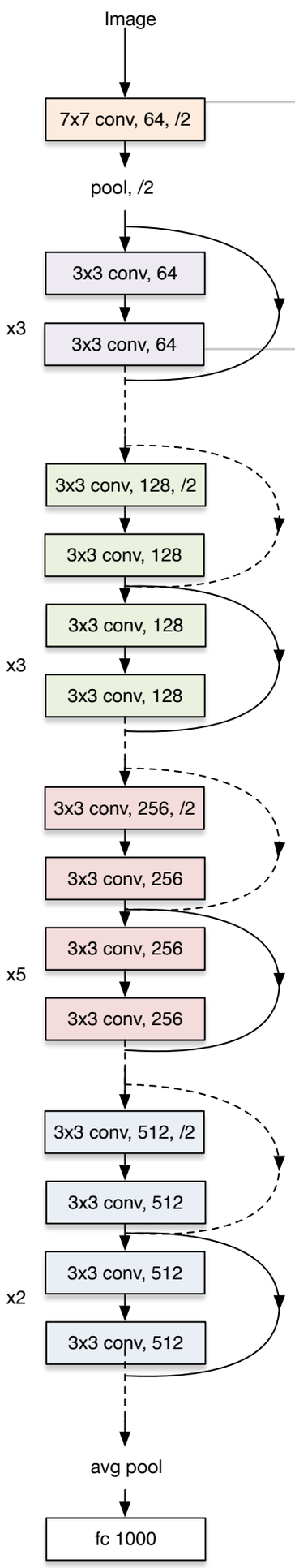


Image Source: He et al., 2016

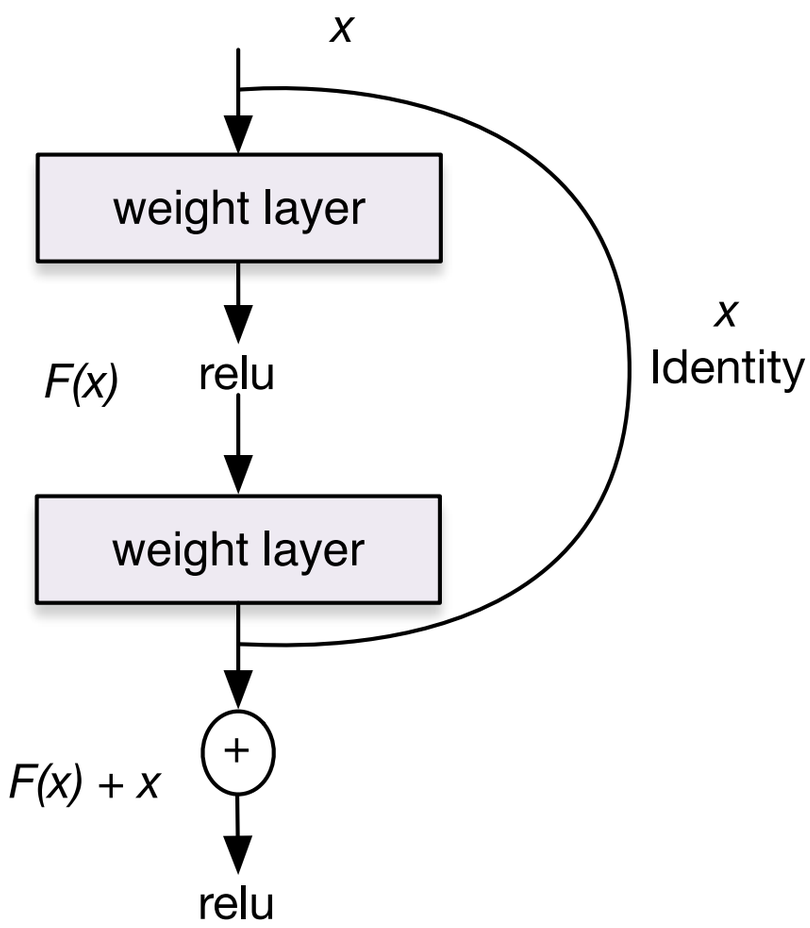
Al tener modelos más complejos se esperaría que la degradación provenga por un overfitting. Sin embargo, no es lo que realmente esta sucediendo (training error es peor)

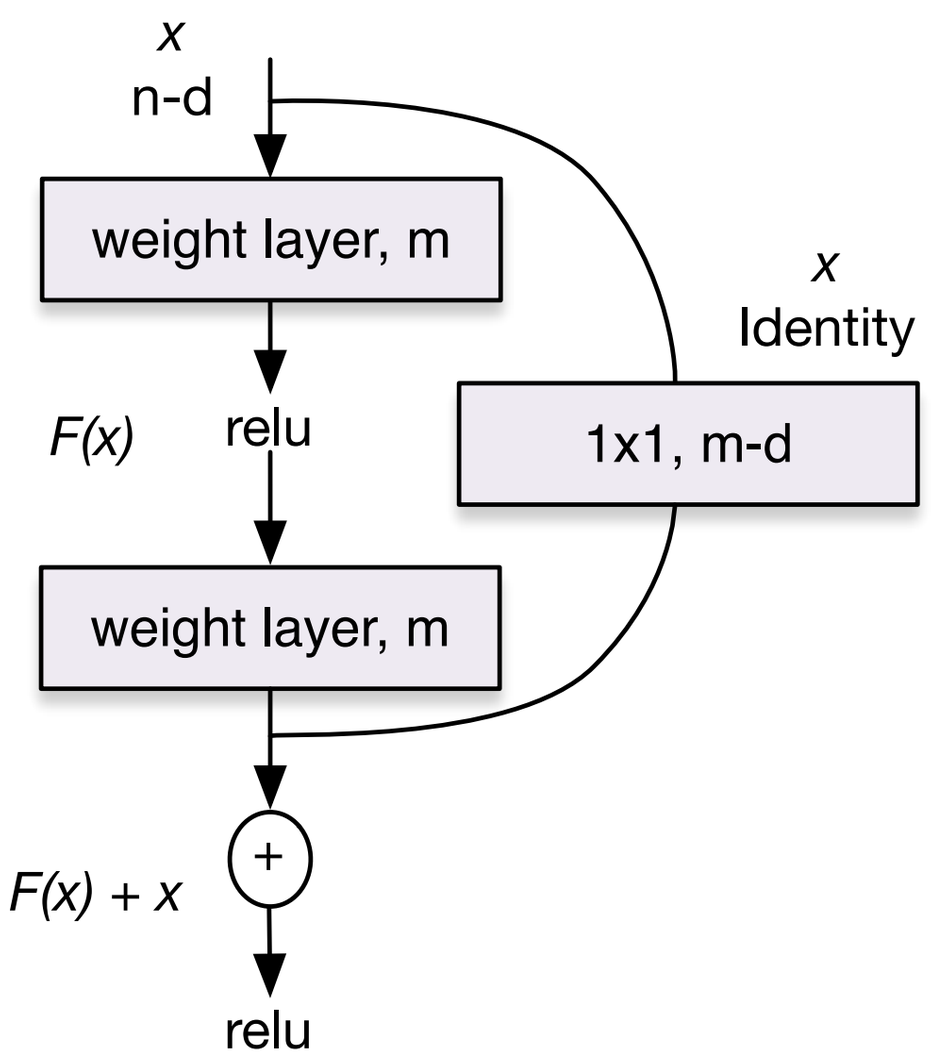
El problema viene por la degradación del gradiente, por eso plantean como atacar el problema de la degradación

ResNet

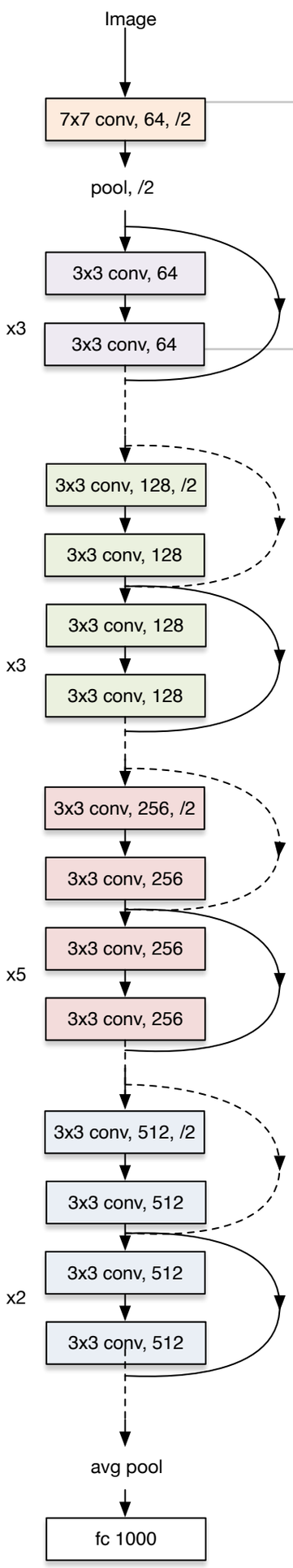


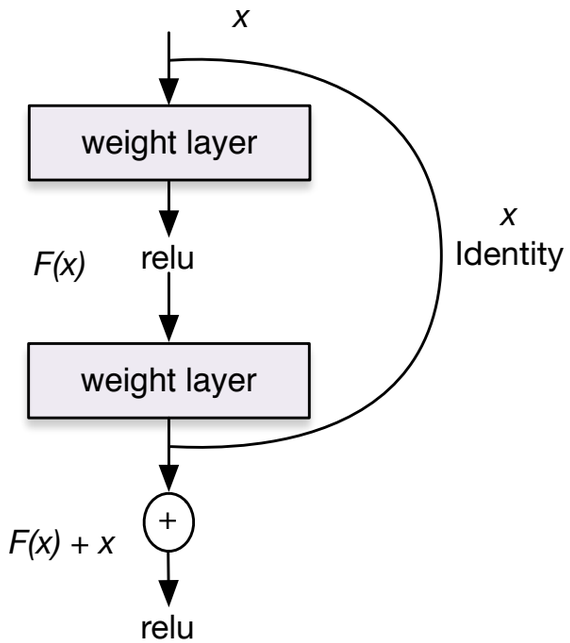
Hipótesis [He et al., 2016]: es más fácil optimizar un mapeo residual que el mapeo original. Sea $H(x)$ el mapeo original, proponen optimizar $F(x) = H(x) - x$





ResNet



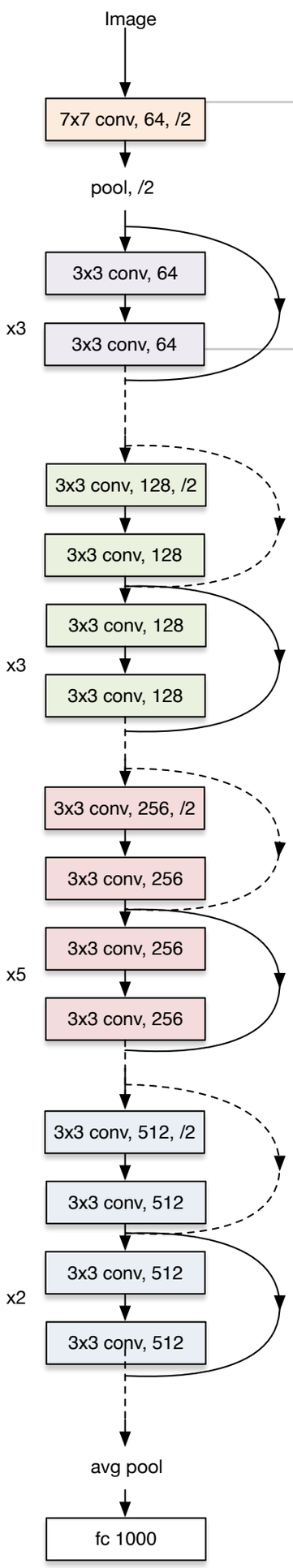


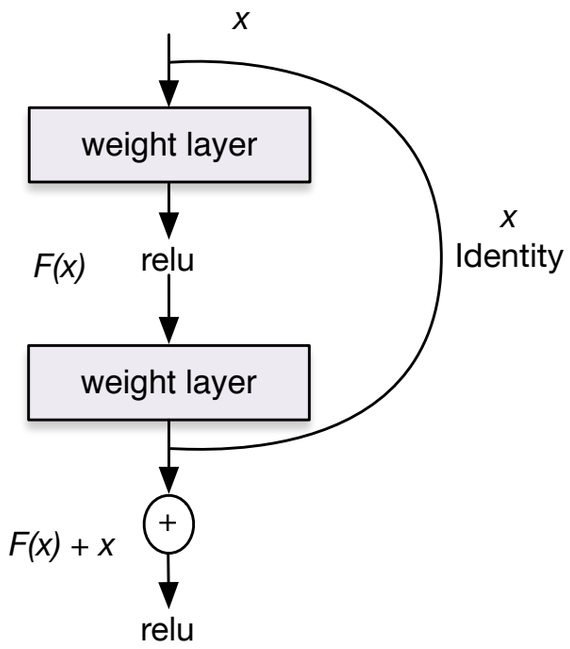
El módulo ResNet tiene dos propiedades muy interesantes:

1) Si los pesos del módulo residual son cero entonces se aprende la identidad, esto implica que hay capas que no se necesitan. El módulo ResNet puede aprender fácilmente a descartar capas que no necesita

2) Mejora el flujo del gradiente ya que reinyecta el gradiente mediante el camino de la identidad

ResNet





El módulo ResNet tiene dos propiedades muy interesantes:

Se puede utilizar una reducción de la dimensionalidad (conv. 1x1) antes del primer conv.

ResNet

[He et al., 2016]

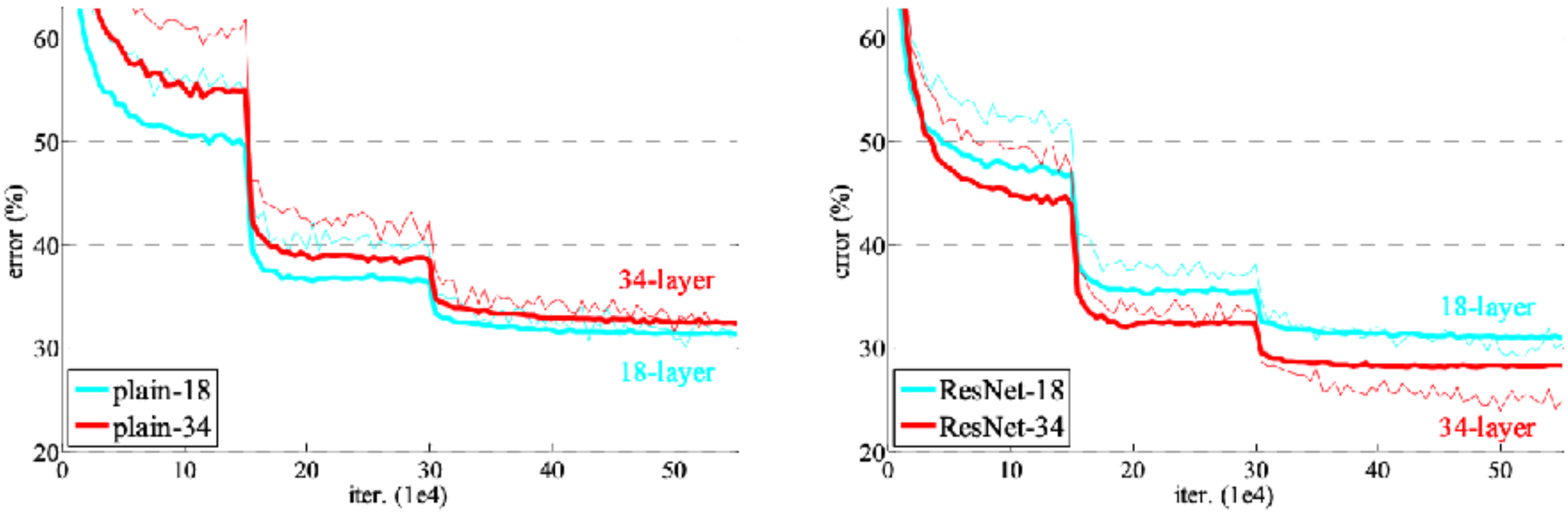


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

Image Source: He et al., 2016