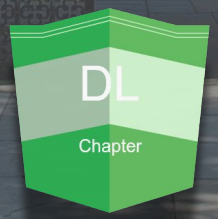
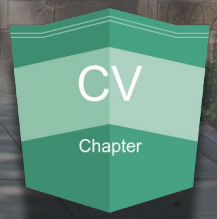
Masterclass

by IDATHA Academy





Contenido

AGENDA

REPASO HISTÓRICO

I REDES CONVOLUCIONALES

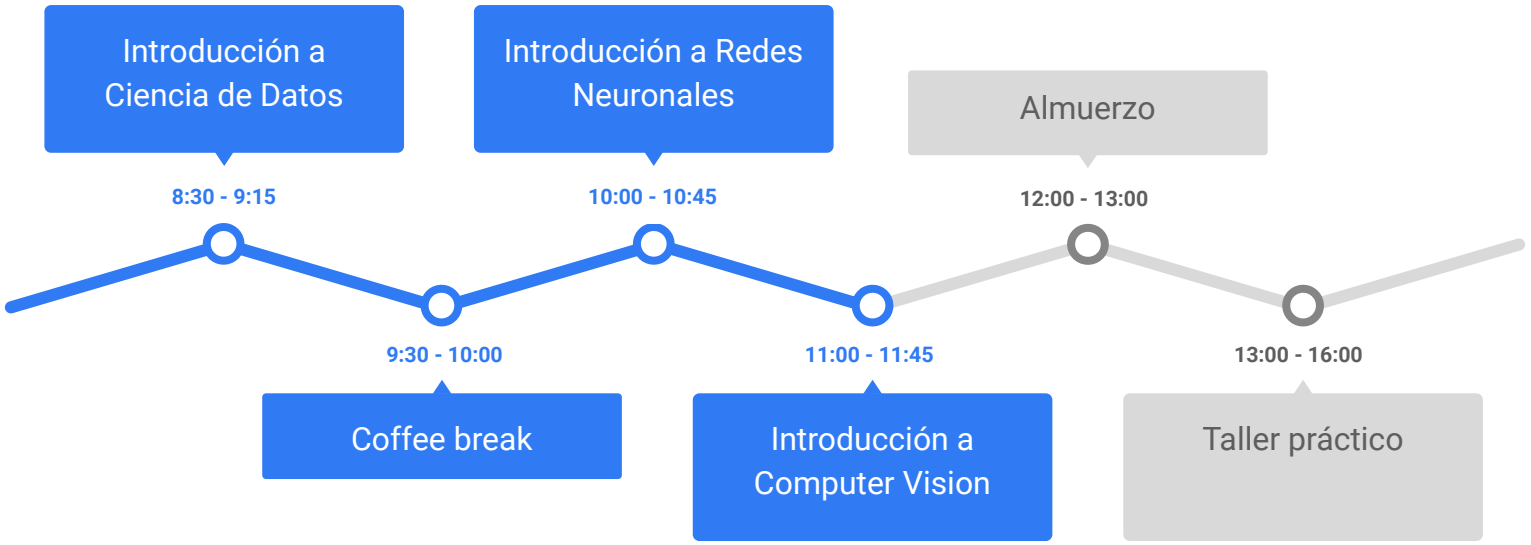
II BLOQUES DE MAX POOLING

III APLICACIONES

IV ARQUITECTURAS

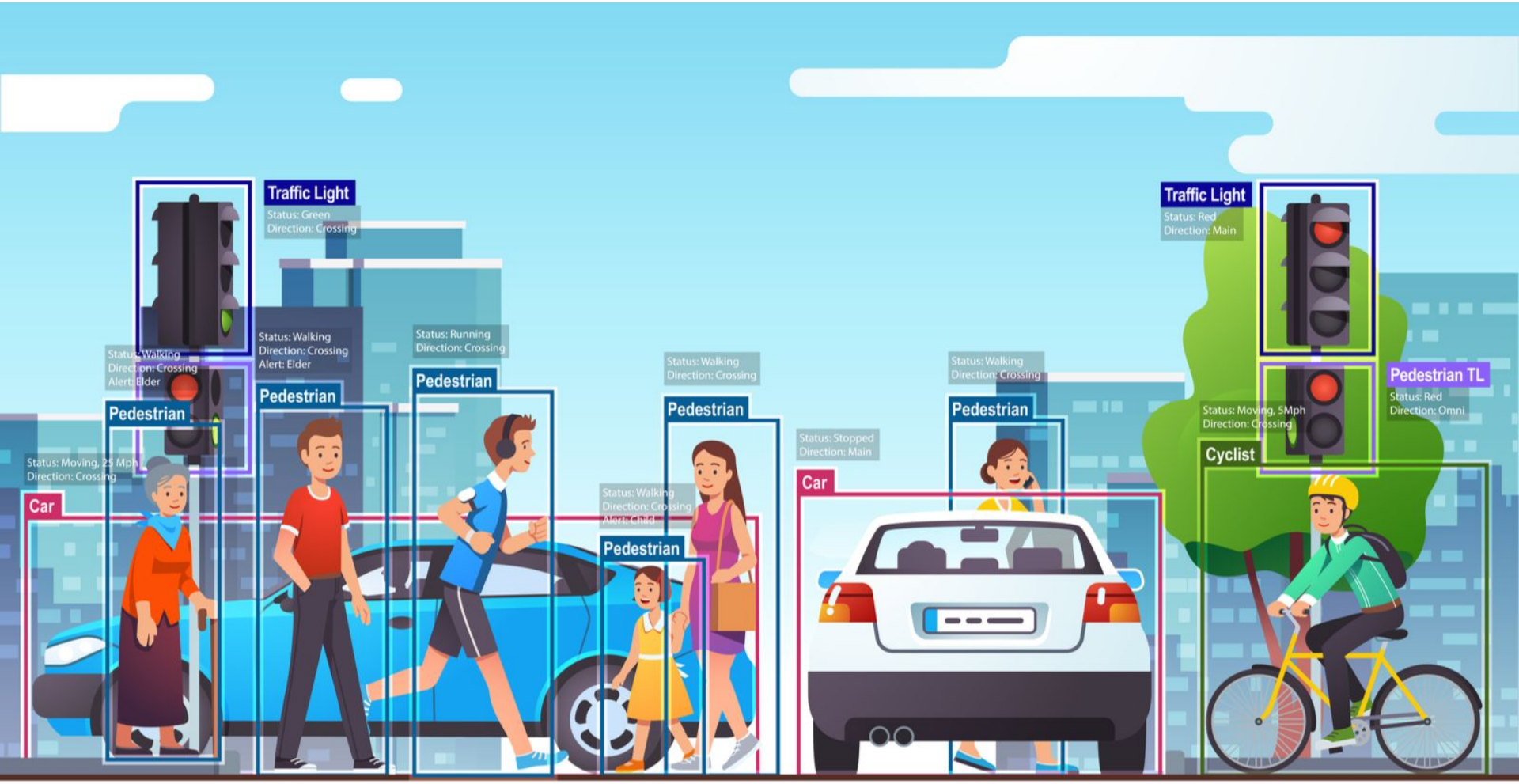
V CONCLUSIONES

AGENDA



IDATHA

¿Qué es la Visión Artificial? (computer vision)



(*) Imagen de 22-tech.com

IDATHA

¿Qué es la Visión Artificial? (computer vision)

“El objetivo de la visión artificial es el de usar datos observados en forma de imágenes para inferir algo sobre el [estado del] mundo.”

(Simon J.D. Prince, Computer Vision: Models, Learning and Inference)

(*) Imagen de towardsdatascience

HISTORIA

1958

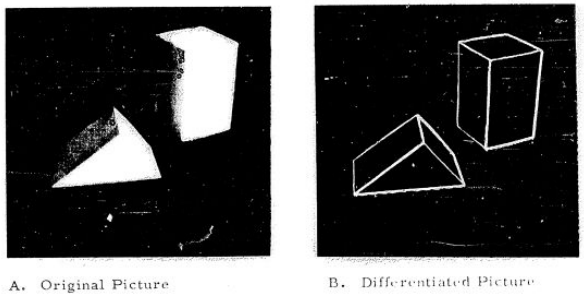
Perceptron de Frank Rosenblatt

Primera implementación de algoritmo basado en la idea de una neurona artificial. Originalmente simulado en una IBM 704

1959

Hubel and Wiesel

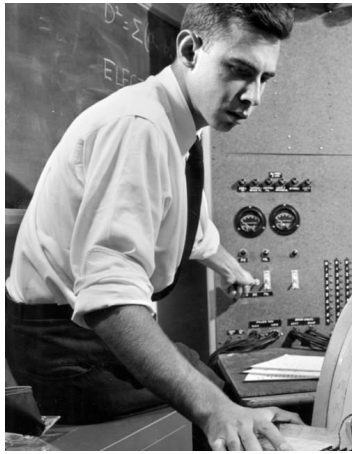
Experimentos sobre el campo de visión en el cerebro de felinos, utilizando aparatos de medición electrónicos.

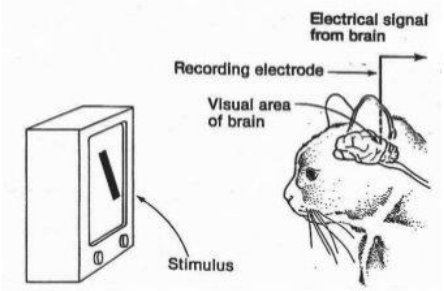


1959

Primera Fotografía Digital

Russell Kirsch digitalizo la primera imagen en 1957 utilizando un escáner que media la variación de intensidad en una imagen. La imagen de tan solo 5x5 cm y 176x176 píxeles de resolución contenía el rostro de su pequeño hijo.





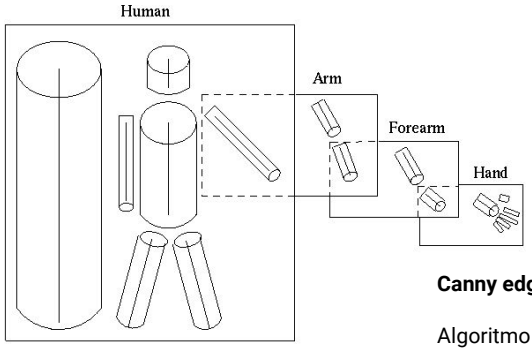
1963

L. Roberts Tesis Doctorado

Machine perception of three-dimensional solids. Programa capaz de detectar sólidos en tres dimensiones a partir de una imagen.

IDATHA

HISTORIA

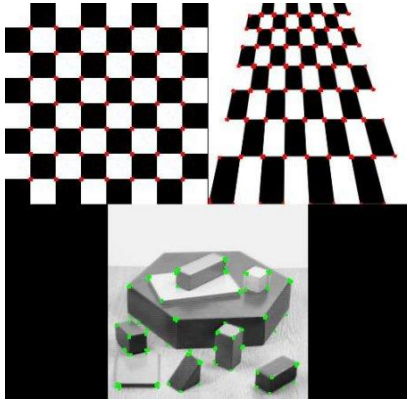


MIT “Summer vision project” de S. Papert

Workshop organizado por el MIT para realizar avances significativos en el campo de la Visión Artificial.

Canny edge detection

Algoritmo desarrollado por John F. Canny para la detección de bordes en imágenes a partir de procesar las variaciones de intensidades en los pixeles de las imágenes.



1966

1982

1986

1988

Bottom-up approach - “Vision” D. Marr

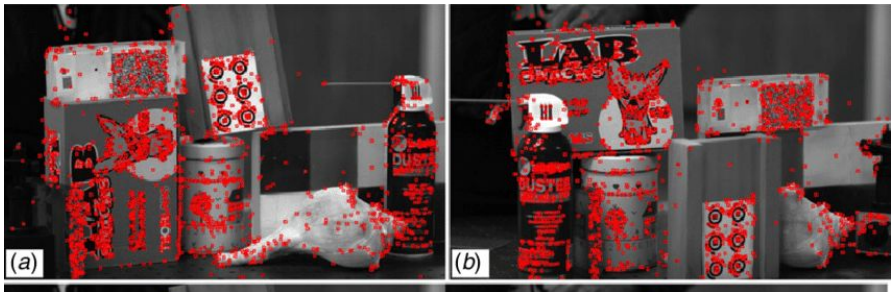
Propuesta de diseño para sistemas de visión artificial y representación tridimensional 3D.

Harris corner detection

Algoritmo propuesto por Chris Harris & Mike Stephen en artículo “A Combined Corner and Edge Detector “ para detectar esquinas. Se basa en la idea de que en una imagen, en una esquina, se produce un cambio abrupto de intensidad en todas las direcciones.

HISTORIA

Tomasi-Kanade Shape from motion



Primeras CNN: Y. Lecun

1992

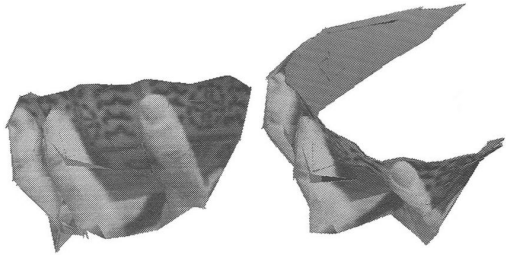
1997

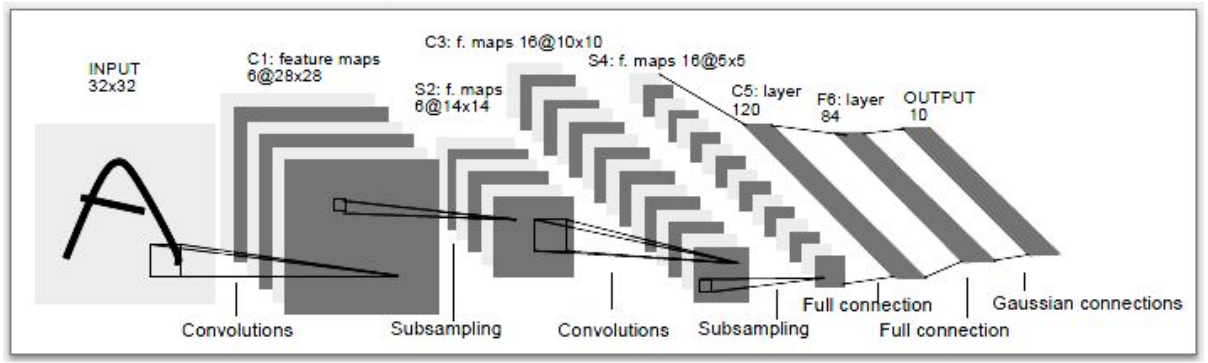
Shi-Malik Image segmentation

1998

1999

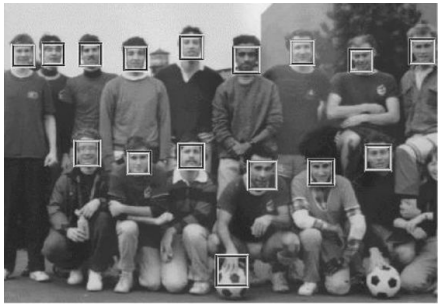
D. Lowe's Scale-Invariant Feature Transform (SIFT)



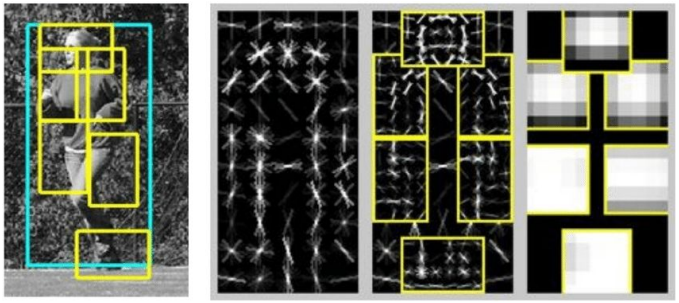


IDATHA

HISTORIA



Viola-Jones face detection

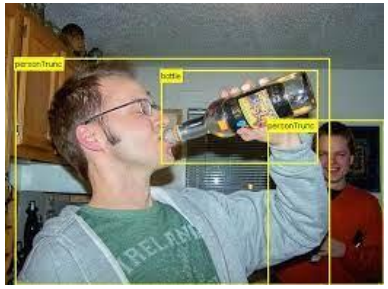


Part-based model - Felzenswab et al. (2009)

2001

2007

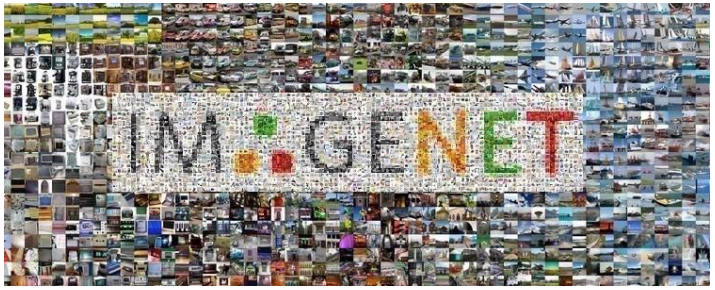
PASCAL Visual Object Classes Challenge (2007)



2009

2010

IMAGENET y competencias en Object Classification



IDATHA

Red Neuronal de múltiples capas

Red con L capas ocultas.

- Entrada:

  $x\ (= h^{(0)})$

- Pre-activación capa *(j)*:

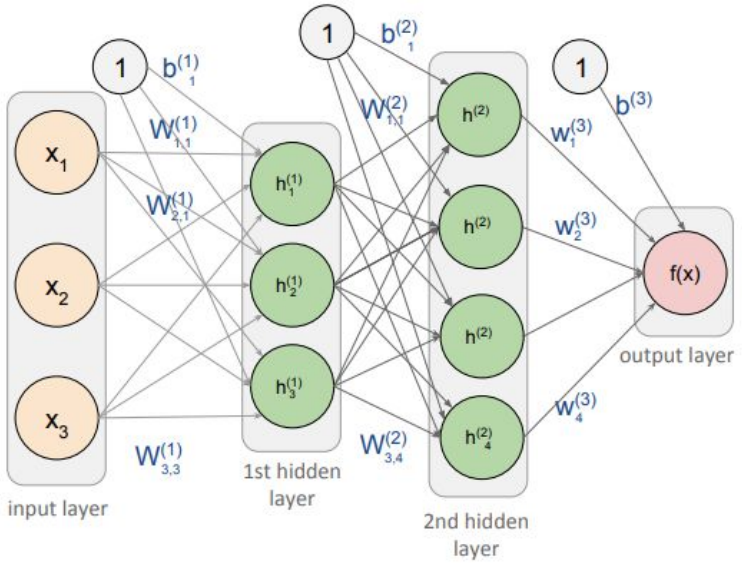
  $a^{(j)}(x) = W^{(j)T} h^{(j-1)} + b^{(j)}$

- Salida capa *(j):*

  $h^{(j)}(x) = g(a^{(j)}(x))$

- Salida de la red (capa $L + 1$):

  $f(x) = o\ (\ a^{(L+1)}\ )$



Nomenclatura:

- "3-layer neural net" o "2-hidden-layer neural net"
- Capas totalmente conectadas ("Fully-connected layers")
- También conocido como (Multi-layer Perceptron (MLP))

Aprendizaje mediante Optimización

- Dado un dataset (x_i, y_i), $i = 1, \ldots n$, con $y_i \in \{1, \ldots, c\}$
- Supongamos que tenemos puntajes de la forma: $s = f(x; W)$
- Mencionamos dos maneras de definir una función de ajuste (*loss*):

SVM (*hinge*)

$$L = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Logistic Regression (Softmax)

$$L = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

- Para encontrar $W = [W^{(1)}, \ldots, W^{(L+1)}]$, minimizar:

$$L(\mathbf{W}) = \sum_{i=1}^{n} L_i(\mathbf{x}_i, y_i; \mathbf{W}) + \lambda R(\mathbf{W})$$

- Descenso por gradiente

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \eta \nabla_{\mathbf{W}} L(\mathbf{W}_t) \quad \eta > 0 \text{ ("learning rate").}$$

Aprendizaje mediante Optimización

- Dado un dataset (x_i, y_i), $i = 1, \ldots n$, con $y_i \in \{1, \ldots, c\}$
- Supongamos que tenemos puntajes de la forma: $s = f(x; W)$
- Mencionamos dos maneras de definir una función de ajuste (*loss*):

SVM (*hinge*)

$$L = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Logistic Regression (Softmax)

$$L = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

- Para encontrar $W = [W^{(1)}, \ldots, W^{(L+1)}]$, minimizar:

$$L(\mathbf{W}) = \sum_{i=1}^{n} L_i(\mathbf{x}_i, y_i; \mathbf{W}) + \lambda R(\mathbf{W})$$

- Algoritmo de *Backpropagation* para calcular:

$$\nabla_{\mathbf{W}} L_i(\mathbf{x}_i, y_i; \mathbf{W}_t)$$

Redes neuronales de convolución (CNN)

Problemas de visión son muy difíciles: se requieren invarianzas a distintas transformaciones (punto de vista, iluminación,..).

Dos caminos:

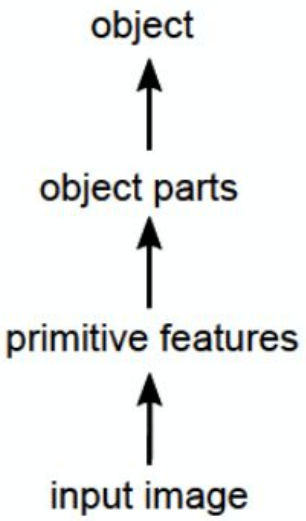
1. **Aprender** las invarianzas a partir de un (enorme) conjunto de un entrenamiento.

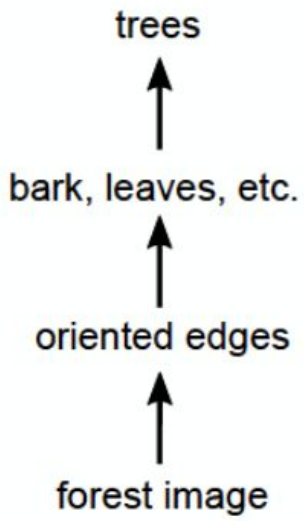
2. **Construir** las invarianzas imponiendo un modelo en la representación.

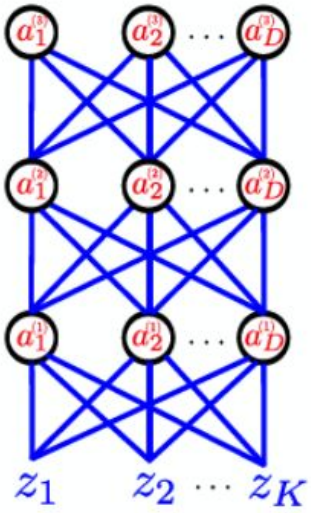



¿Por qué usar redes profundas?

Las imágenes (en general) tienen una organización jerárquica:



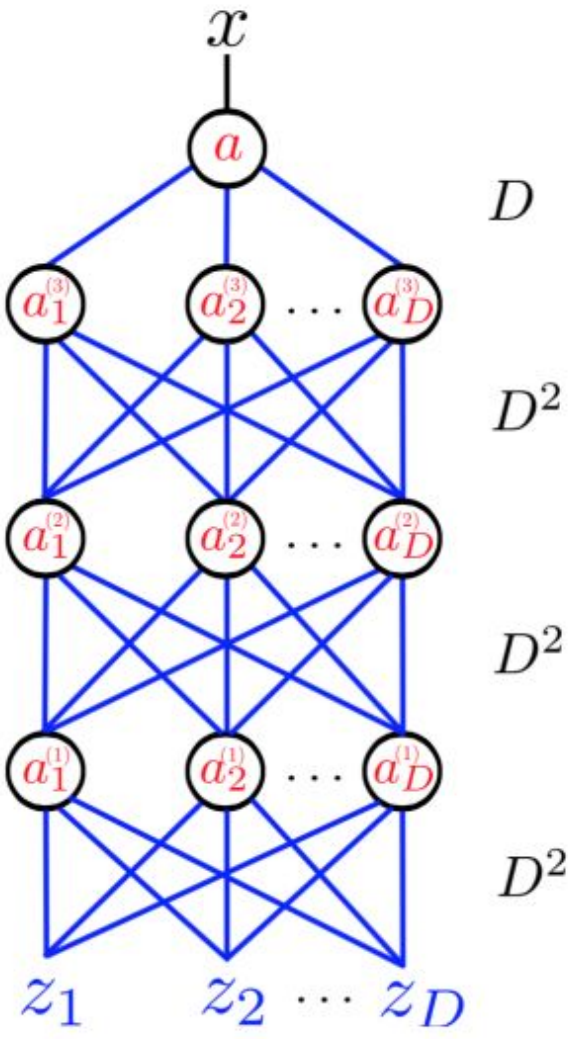




¿Cuál es el problema de las redes *fully connected*?

- ¿Cuántos parámetros tiene esta red?
  $3D^2 + D$
- Si tenemos una imagen de 32x32
  $3 \times (32^2)^2 + 32^2 \approx 3 \times 10^6$

- **Difícil de entrenar**: sobreajuste, inicialización
- **Redes de Convolución**: permiten disminuir número de parámetros forzando invarianzas



1. Redes Convolucionales

Convolución de imágenes

• **Convolución en imágenes**: Operación **lineal** entre imagen y filtro, se produce una nueva imagen.

• Cada píxel se calcula como suma ponderada píxeles de imagen entrada trasladada y núcleo de convolución:

$$(u * h)(i,j) = \sum_{k,l} u(i-k, j-l)h(k,l)$$

$$\underbrace{\begin{bmatrix} 130 & 136 & 53 & 44 & 231 & 67 & 108 \\ 130 & 89 & 77 & 58 & 250 & 154 & 130 \\ 208 & 239 & 120 & 111 & 112 & 181 & 22 \\ 203 & 223 & 59 & 79 & 28 & 57 & 67 \\ 164 & 140 & 215 & 235 & 66 & 30 & 204 \\ 97 & 159 & 50 & 110 & 104 & 76 & 7 \\ 207 & 150 & 58 & 47 & 152 & 81 & 237 \end{bmatrix}}_{u} * \underbrace{\frac{1}{5}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}}_{h} = \underbrace{\begin{bmatrix} 79.2 & 81.6 & 62.0 & 77.2 & 118.4 & 112.0 & 61.0 \\ 111.4 & 134.2 & 79.4 & 108.0 & 161.0 & 156.4 & 82.8 \\ 156.0 & 175.8 & 121.2 & 96.0 & 136.4 & 105.2 & 80.0 \\ 159.6 & 172.8 & 139.2 & 102.4 & 68.4 & 72.6 & 70.0 \\ 120.8 & 180.2 & 139.8 & 141.0 & 92.6 & 86.6 & 61.6 \\ 125.4 & 119.2 & 118.4 & 109.2 & 101.6 & 59.6 & 104.8 \\ 90.8 & 114.8 & 61.0 & 73.4 & 76.8 & 109.2 & 65.0 \end{bmatrix}}_{u*h}$$

Convolución de imágenes

Image Kernels, Explained Visually por Victor Powell
https://setosa.io/ev/image-kernels/

Image Kernels

Explained Visually

By Victor Powell

An image kernel is a small matrix used to apply effects like the ones you might find in Photoshop or Gimp, such as blurring, sharpening, outlining or embossing. They're also used in machine learning for 'feature extraction', a technique for determining the most important portions of an image. In this context the process is referred to more generally as "convolution" (see: convolutional neural networks.)

To see how they work, let's start by inspecting a black and white image. The matrix on the left contains numbers, between 0 and 255, which each correspond to the brightness of one pixel in a picture of a face. The large, granulated picture has been blown up to make it easier to see; the last image is the "real" size.

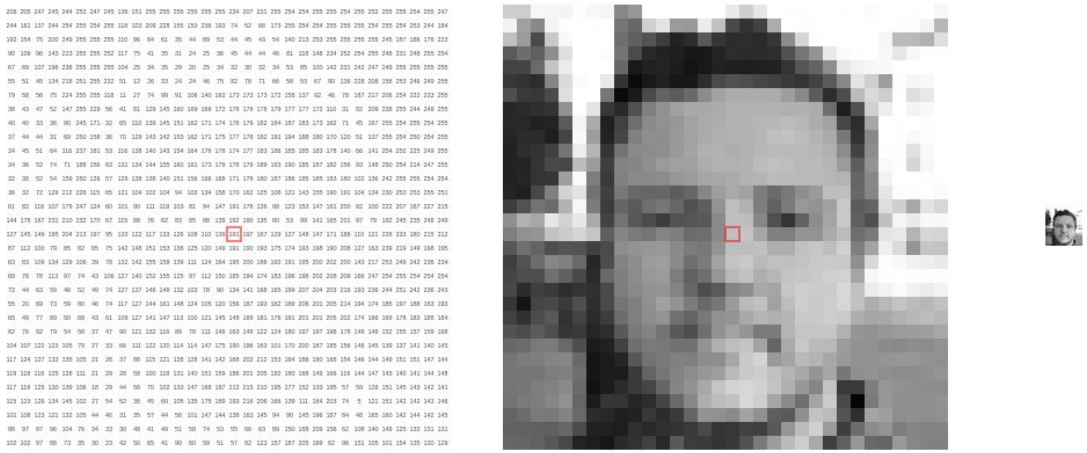
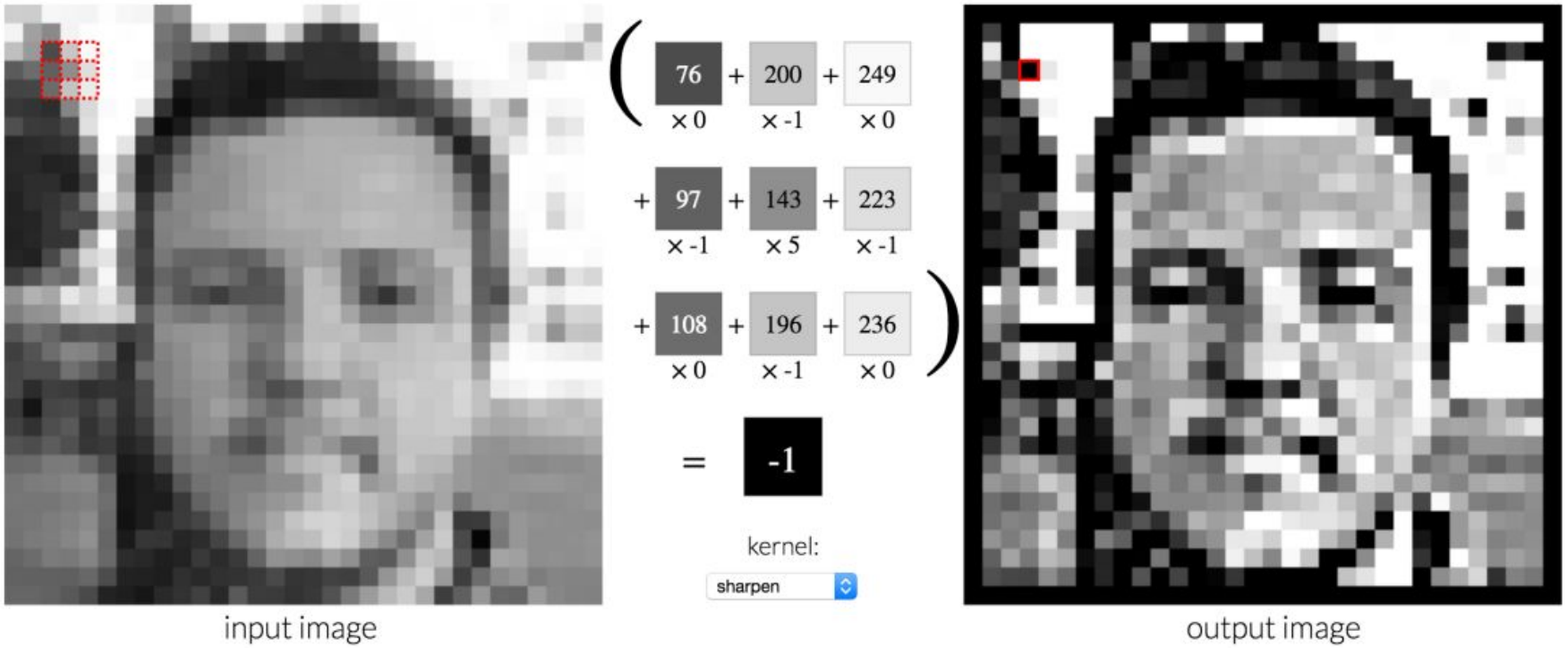


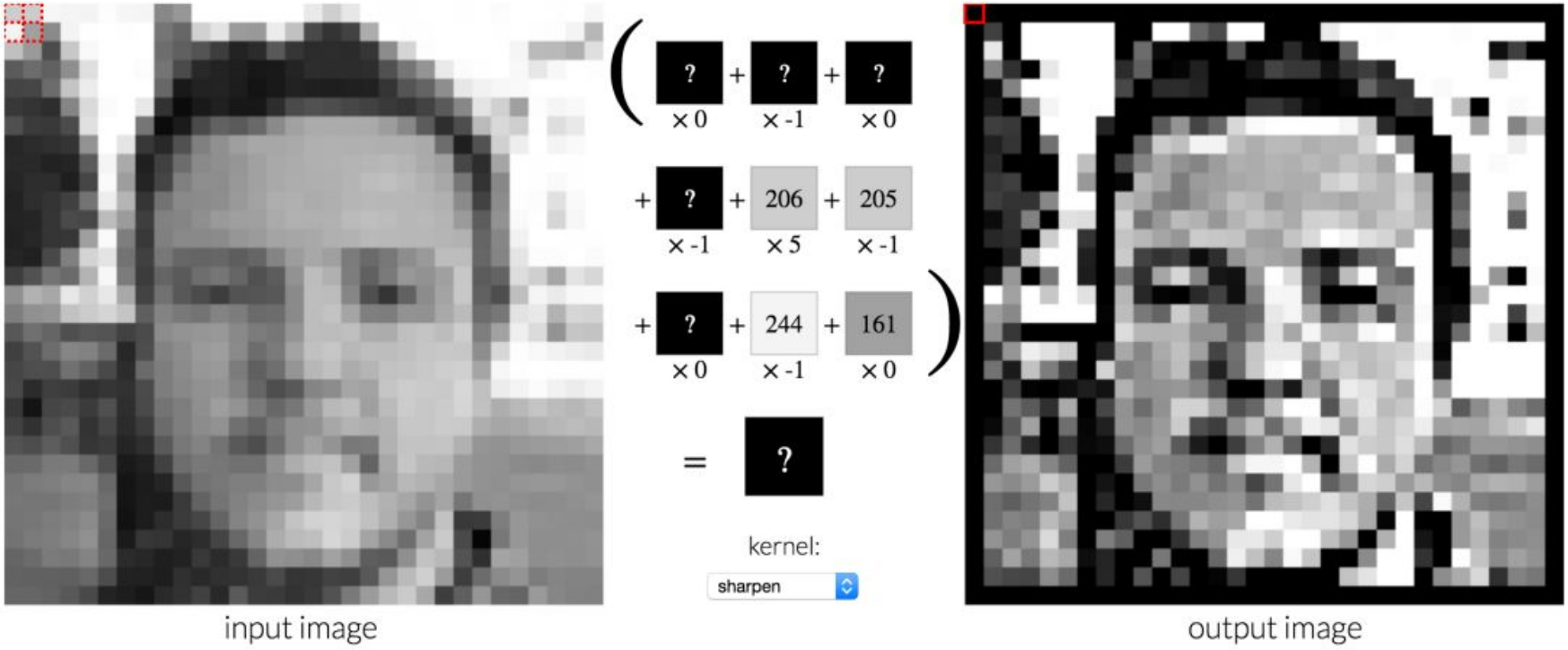
Image Kernels, Explained Visually por Victor Powell
https://setosa.io/ev/image-kernels/



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

Convolución de imágenes

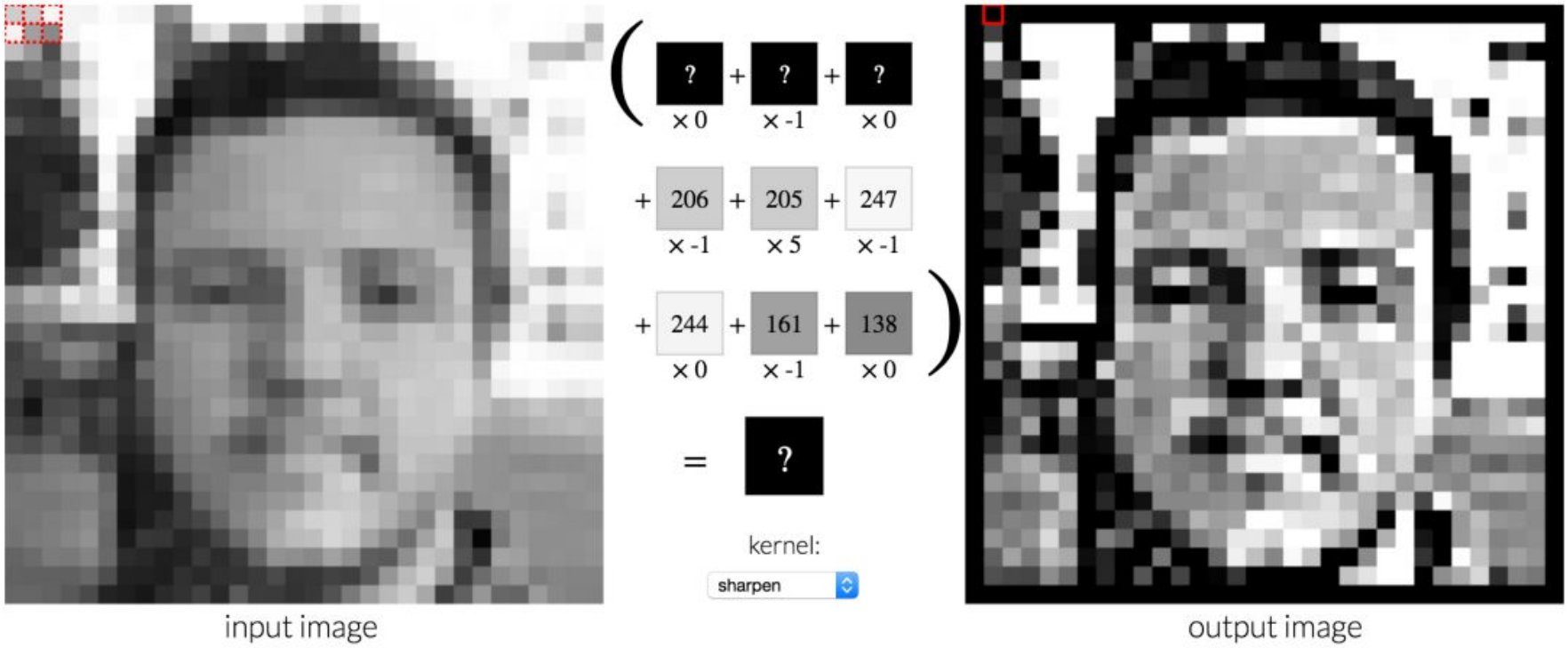
Image Kernels, Explained Visually por Victor Powell
https://setosa.io/ev/image-kernels/



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

Convolución de imágenes

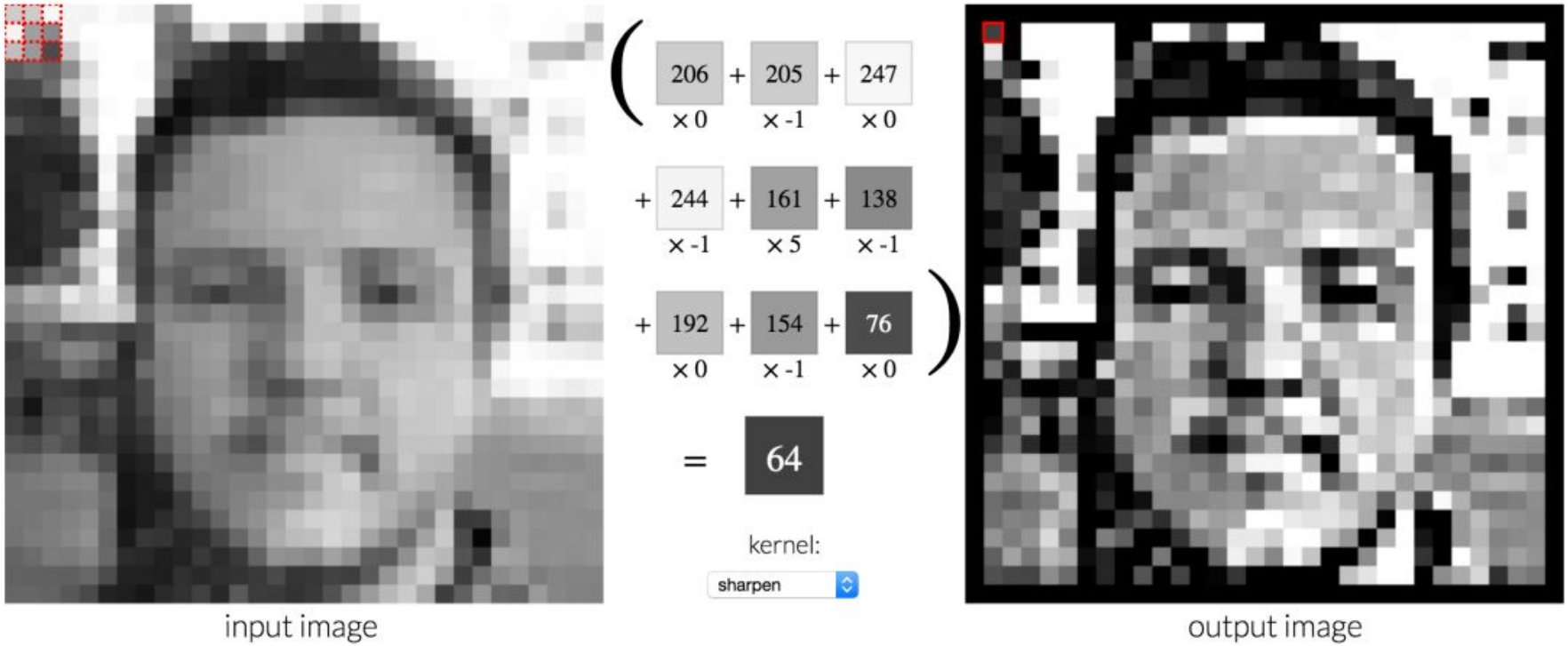
Image Kernels, Explained Visually por Victor Powell
https://setosa.io/ev/image-kernels/



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

Convolución de imágenes

Image Kernels, Explained Visually por Victor Powell
https://setosa.io/ev/image-kernels/



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

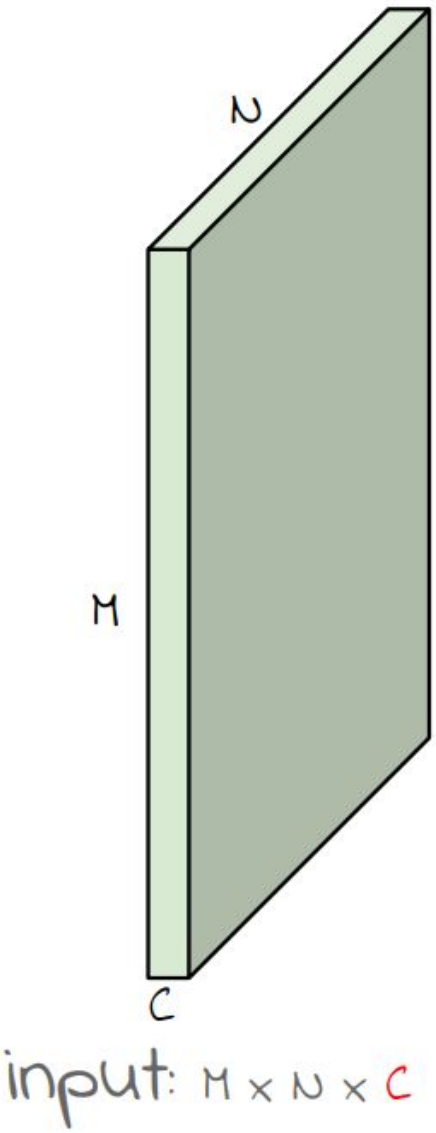
Redes Convolucionales: Motivación

1. **Estadísticas en imágenes son invariantes a traslaciones**
   a. Imponer invarianza a traslación en el modelo (en lugar de aprenderla)
   b. Baja el número de parámetros: Se comparten pesos

2. **Características de bajo nivel son locales (detector de blobs, bordes)**
   a. Imponer localidad en el modelo: conectividad local (soporte del filtro)
   b. Baja el número de parámetros: Núcleos pequeños

3. **Se espera que características de alto nivel sean gruesas (biología)**
   a. Se puede submuestrear a medida que aumenta la profundidad en la red
   b. Baja (aún más) el número de parámetros

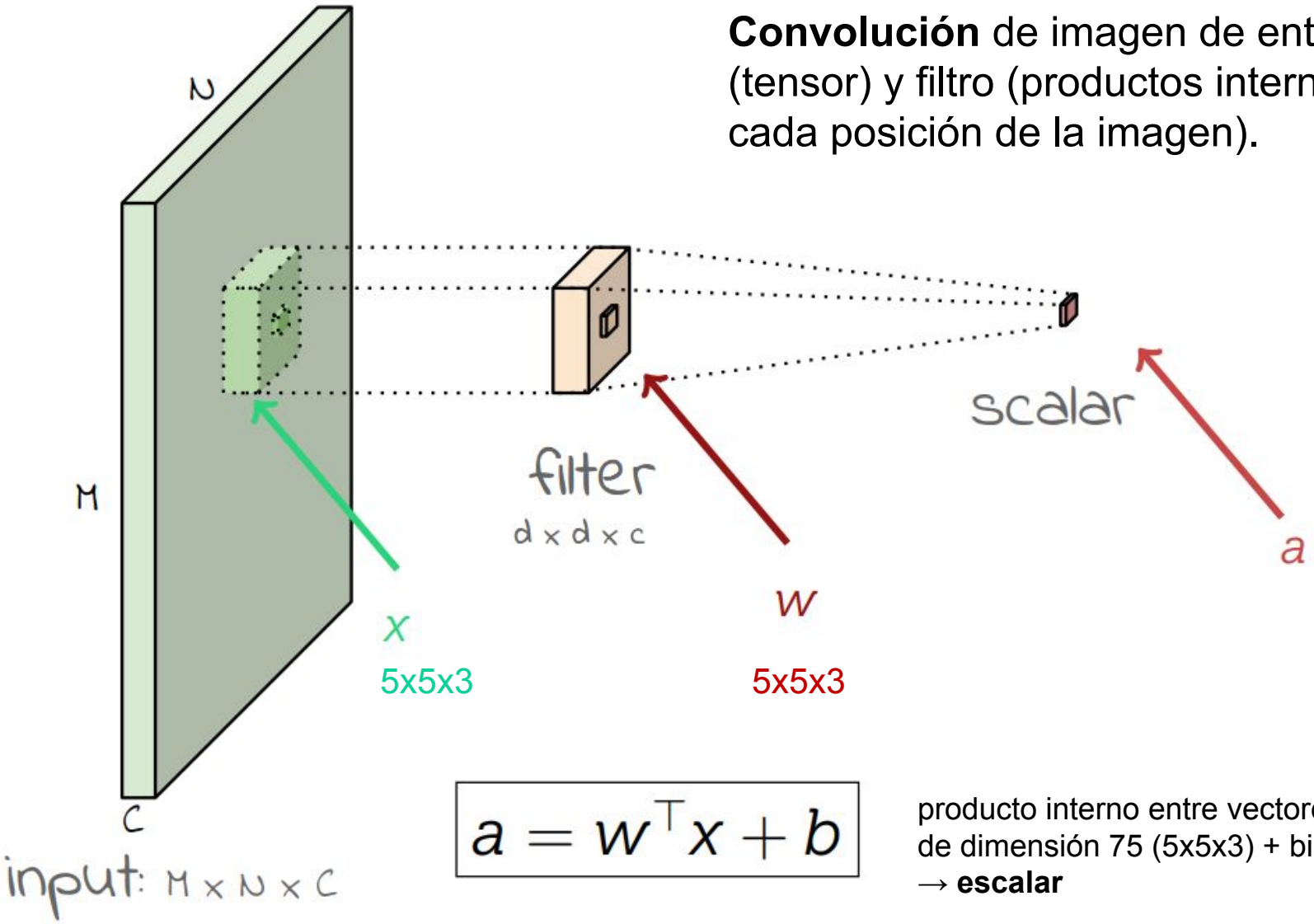
Capa de Convolución



Convolución de imagen de entrada (tensor) y filtro (productos internos cada posición de la imagen).

Capa de Convolución

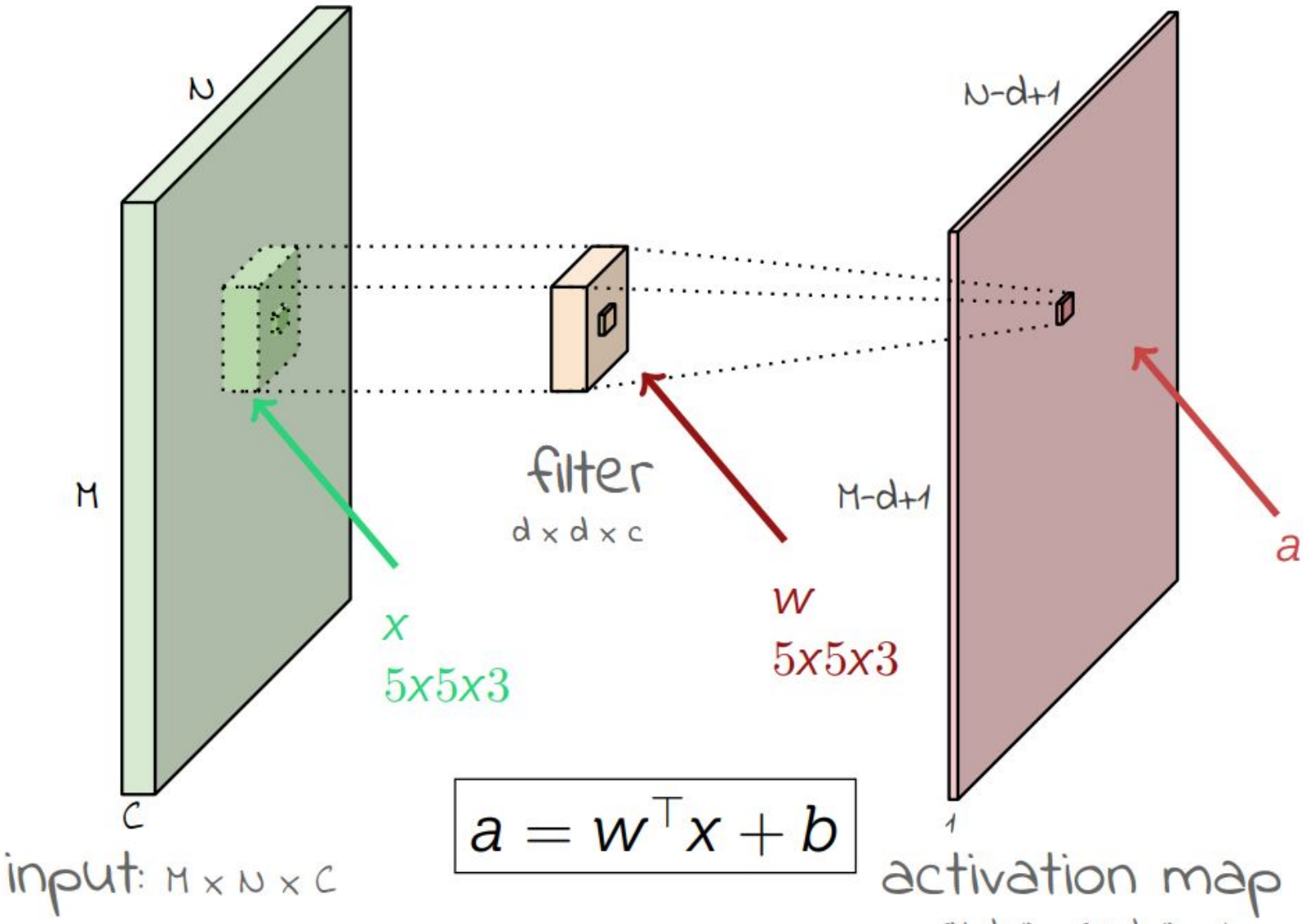
Convolución de imagen de entrada (tensor) y filtro (productos internos cada posición de la imagen).



$$a = w^\top x + b$$

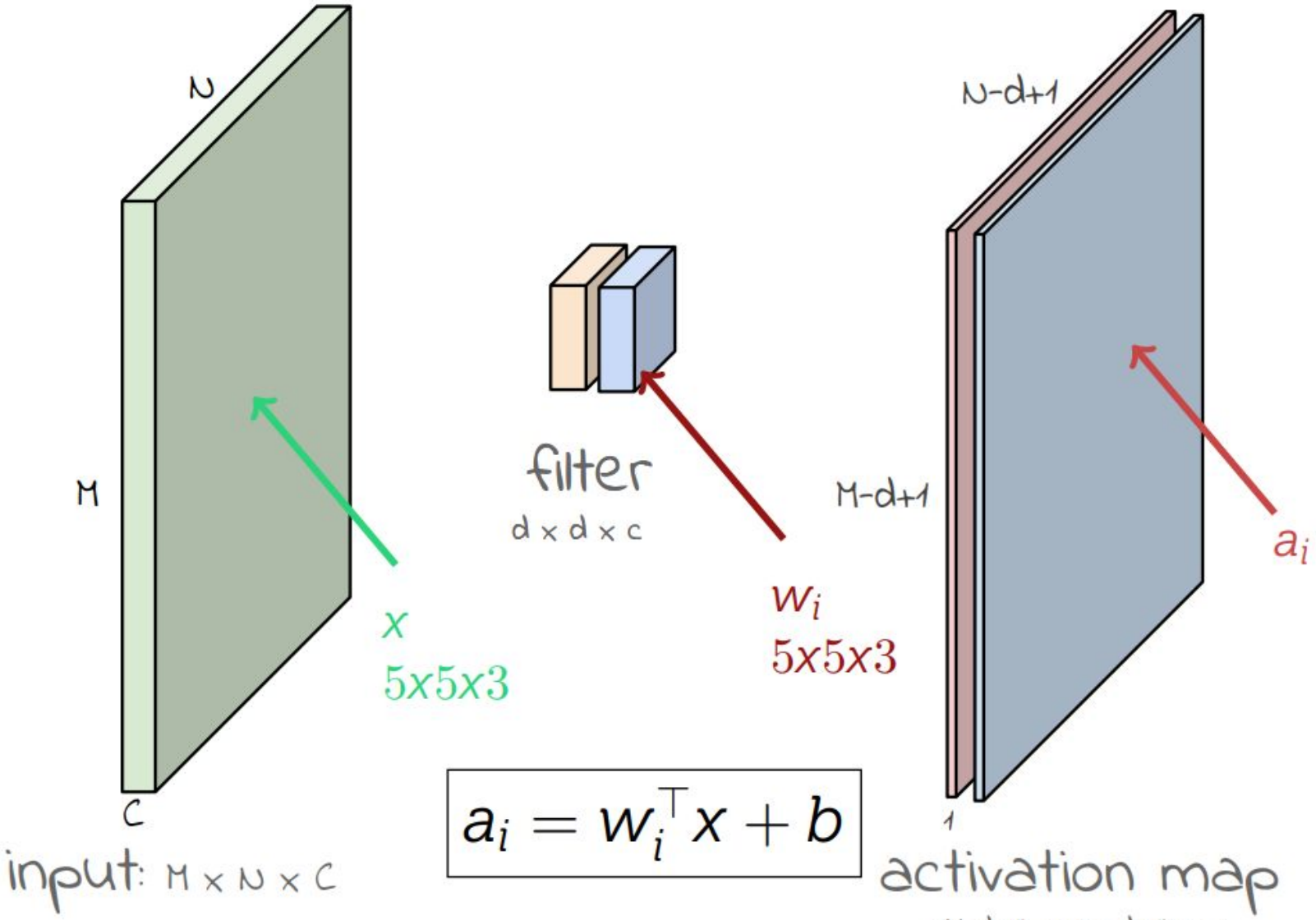
producto interno entre vectores de dimensión 75 (5x5x3) + bias → **escalar**

Capa de Convolución



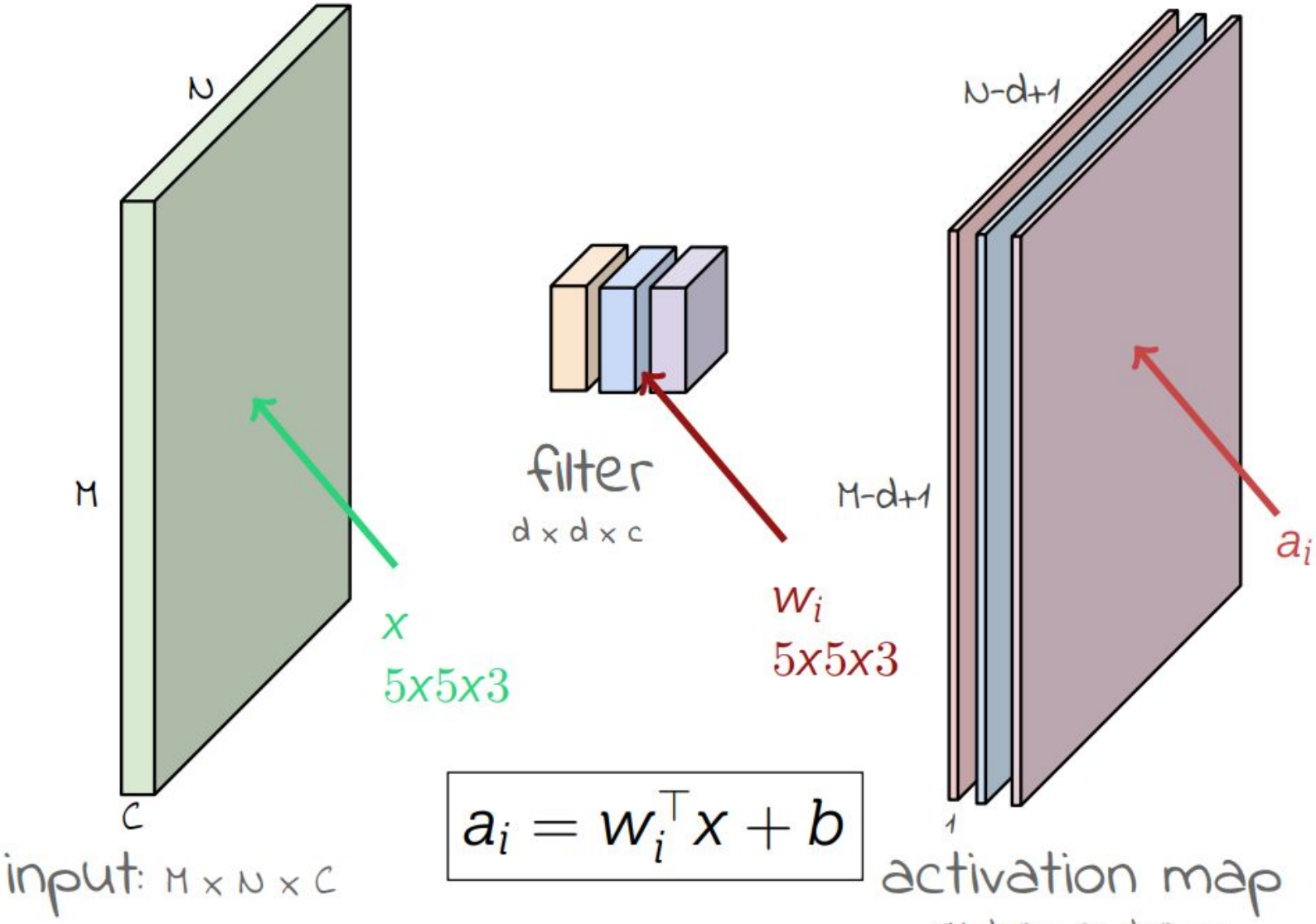
$$a = w^\top x + b$$

Capa de Convolución



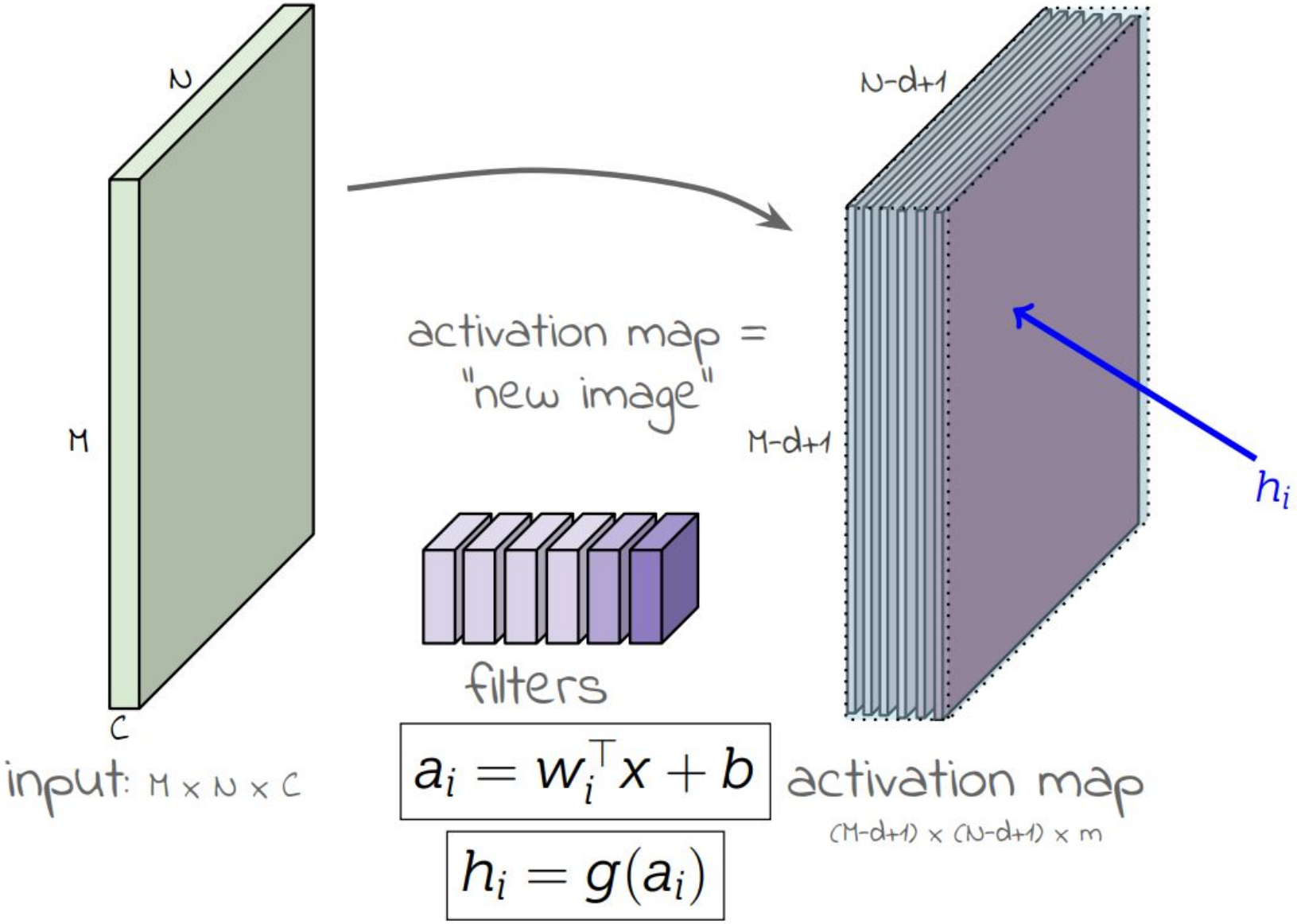
$$a_i = w_i^\top x + b$$

Capa de Convolución

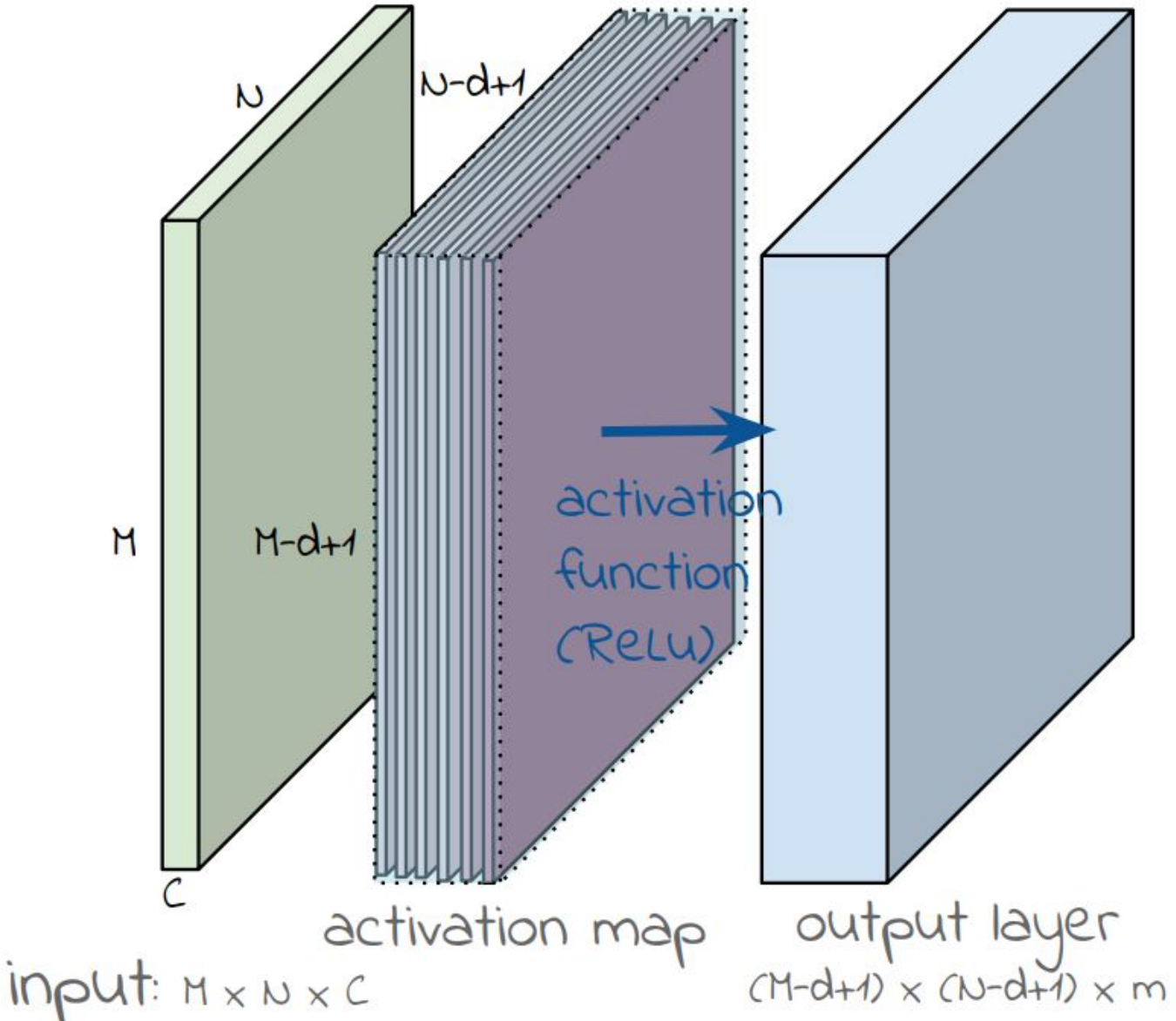


$$a_i = w_i^\top x + b$$

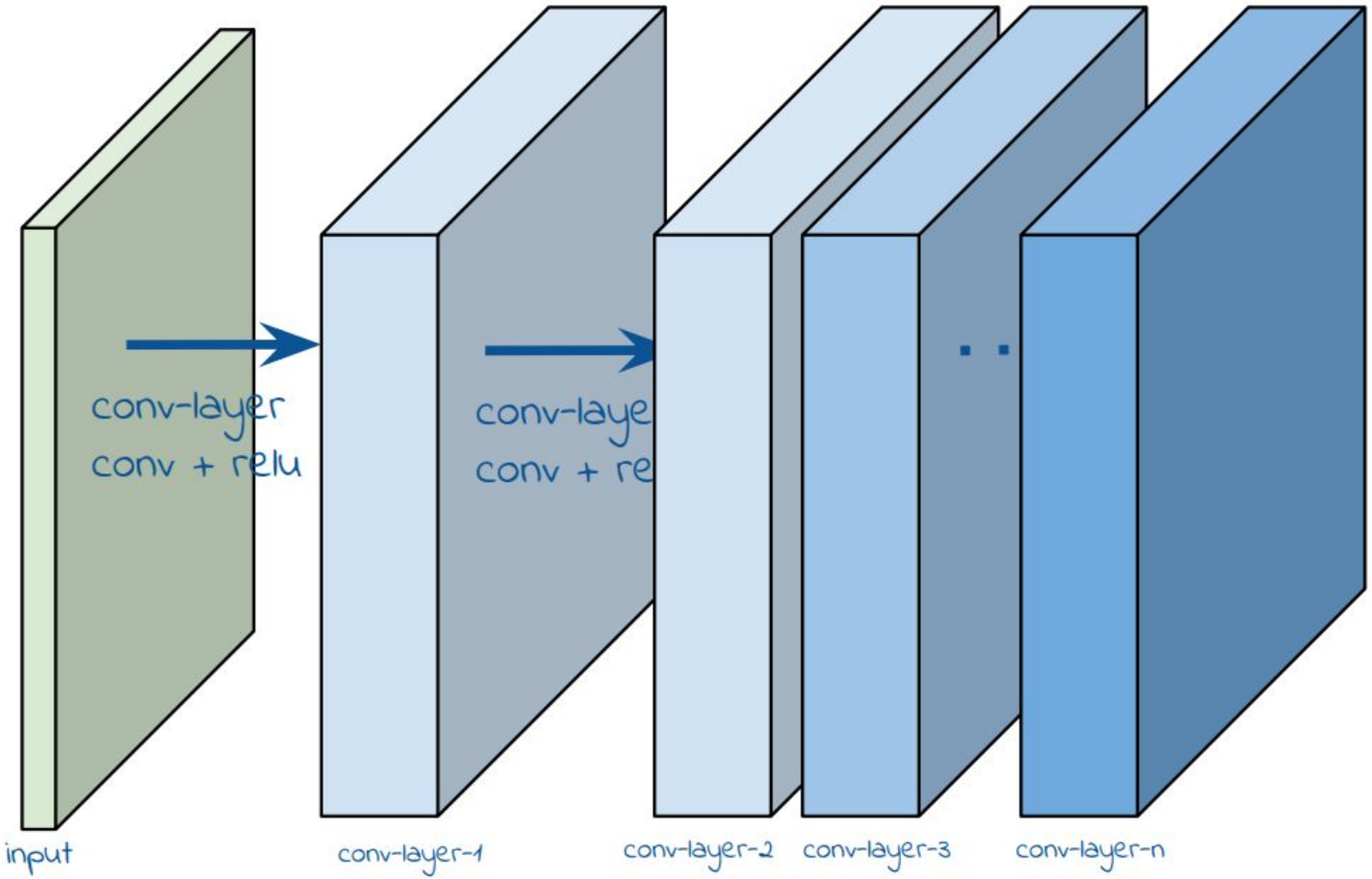
Capa de Convolución



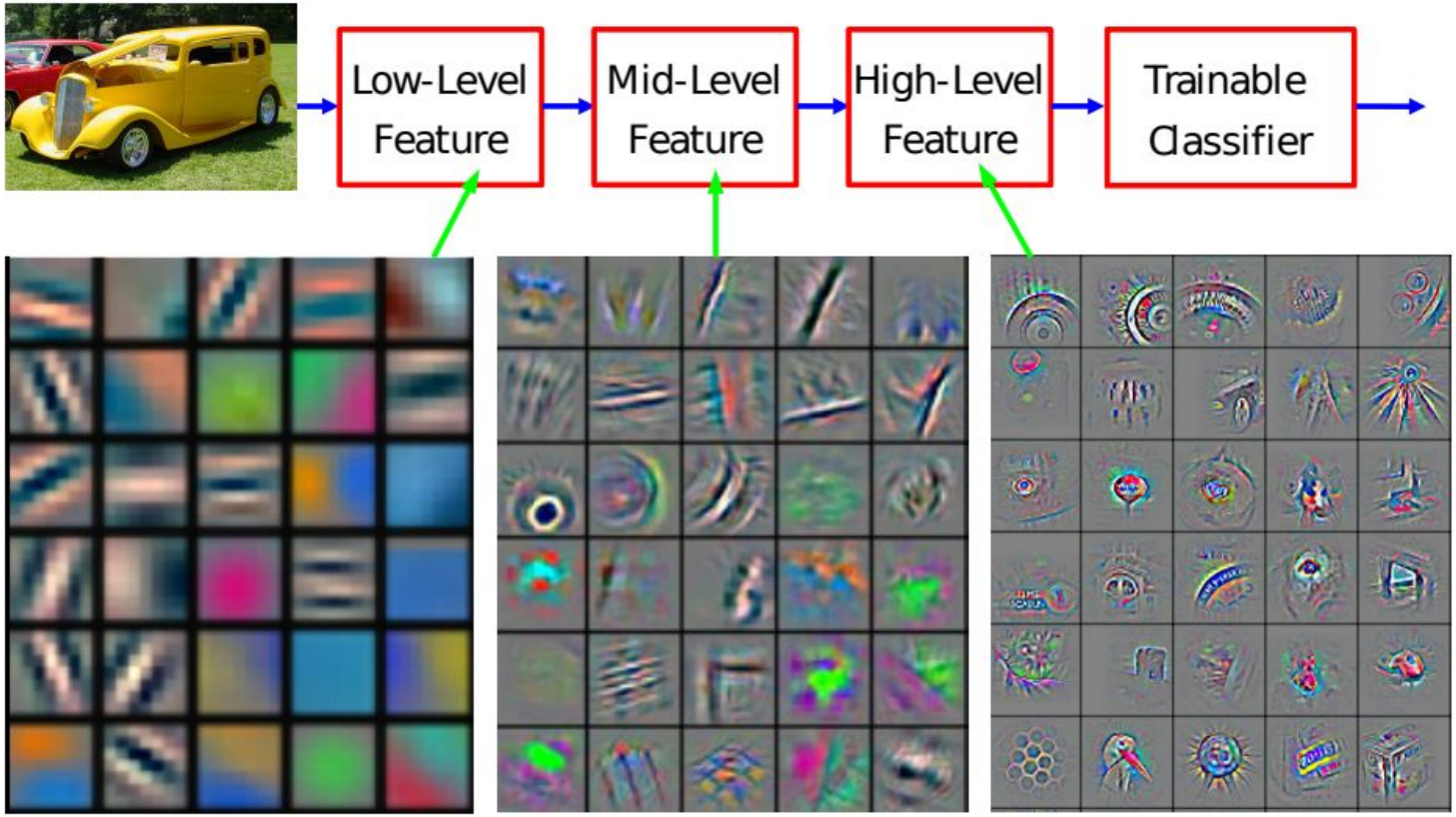
Capa de Convolución + Activación



Capa de Convolución + Activación



Representaciones jerárquicas

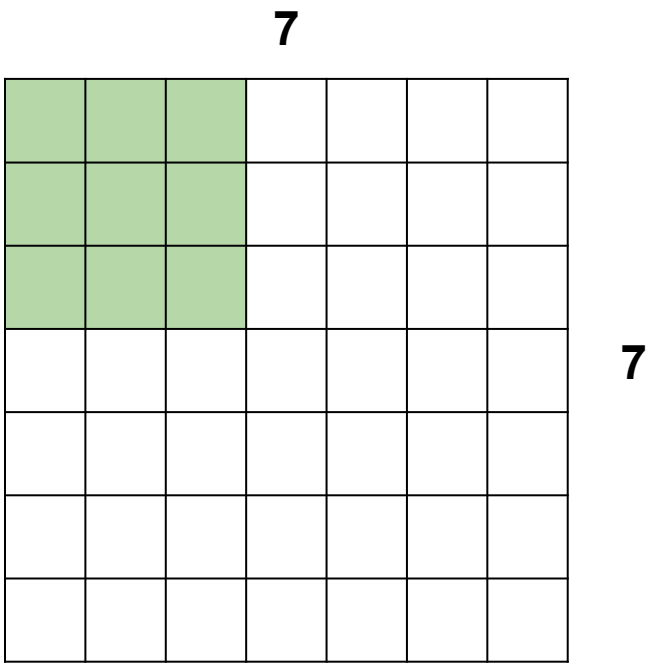


Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Ejemplo tomado de charlas de Y. Lecun

Capa de Convolución

Una mirada más de cerca sobre las dimensiones:

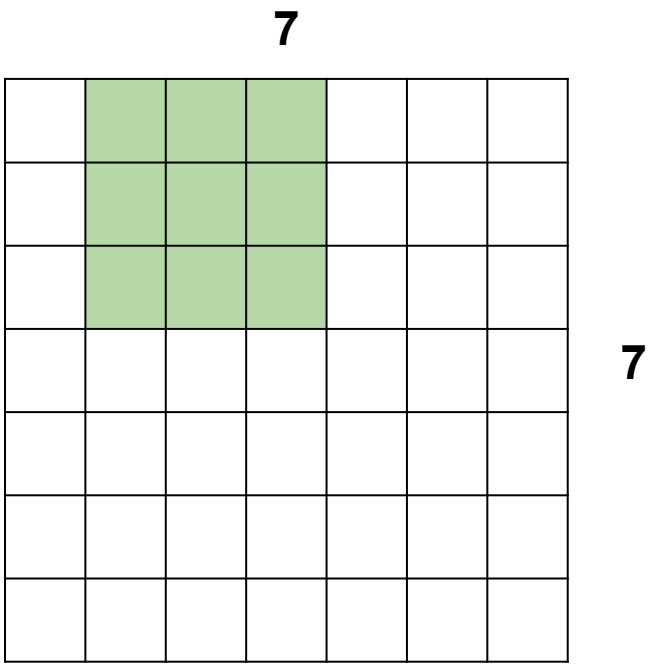


7x7 input (dimensiones)
asumiendo filtro 3x3

Slides tomadas de cs231n (stanford) - Fei-Fei Li & Justin Johnson & Serena Yeung

Capa de Convolución

Una mirada más de cerca sobre las dimensiones:

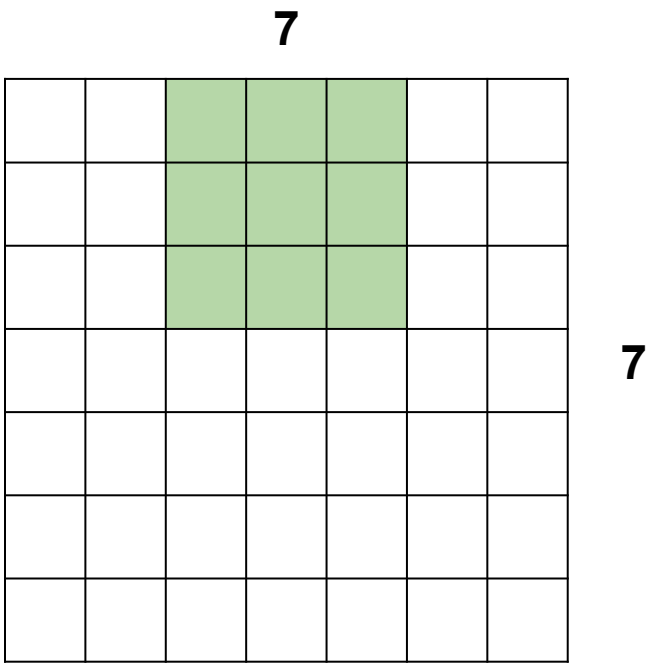


7x7 input (dimensiones)
asumiendo filtro 3x3

Slides tomadas de cs231n (stanford) - Fei-Fei Li & Justin Johnson & Serena Yeung

Capa de Convolución

Una mirada más de cerca sobre las dimensiones:

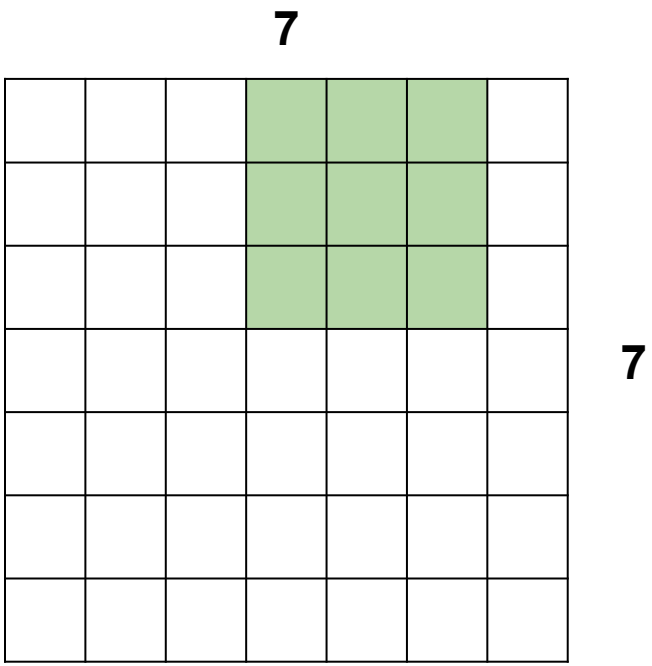


7x7 input (dimensiones)
asumiendo filtro 3x3

Slides tomadas de cs231n (stanford) - Fei-Fei Li & Justin Johnson & Serena Yeung

Capa de Convolución

Una mirada más de cerca sobre las dimensiones:

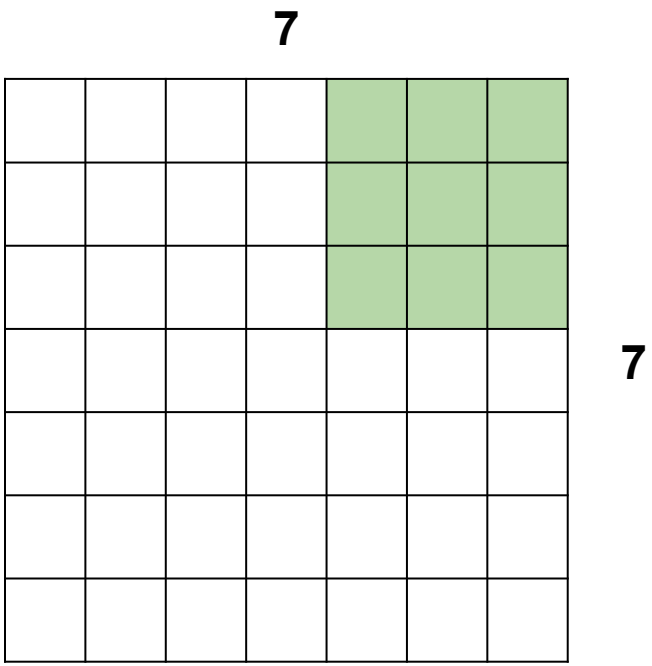


7x7 input (dimensiones)
asumiendo filtro 3x3

Slides tomadas de cs231n (stanford) - Fei-Fei Li & Justin Johnson & Serena Yeung

Capa de Convolución

Una mirada más de cerca sobre las dimensiones:



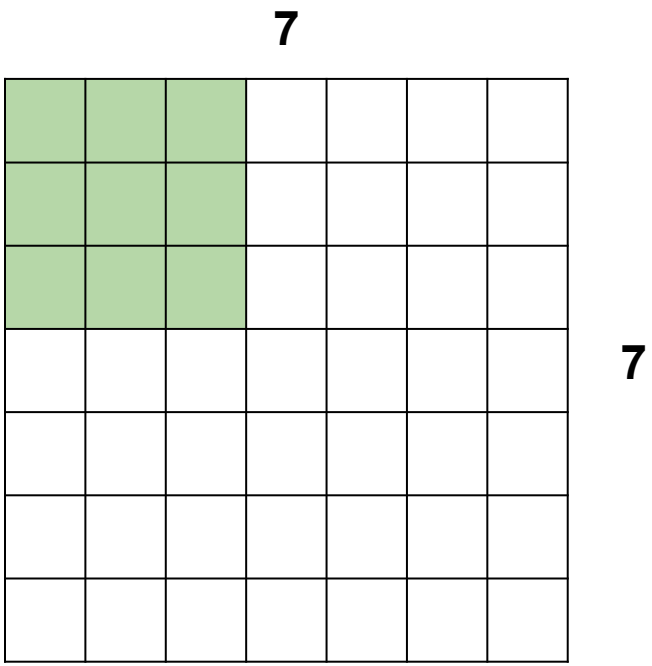
7x7 input (dimensiones)
asumiendo filtro 3x3

→ **5x5 output**

Slides tomadas de cs231n (stanford) - Fei-Fei Li & Justin Johnson & Serena Yeung

Capa de Convolución

Una mirada más de cerca sobre las dimensiones:

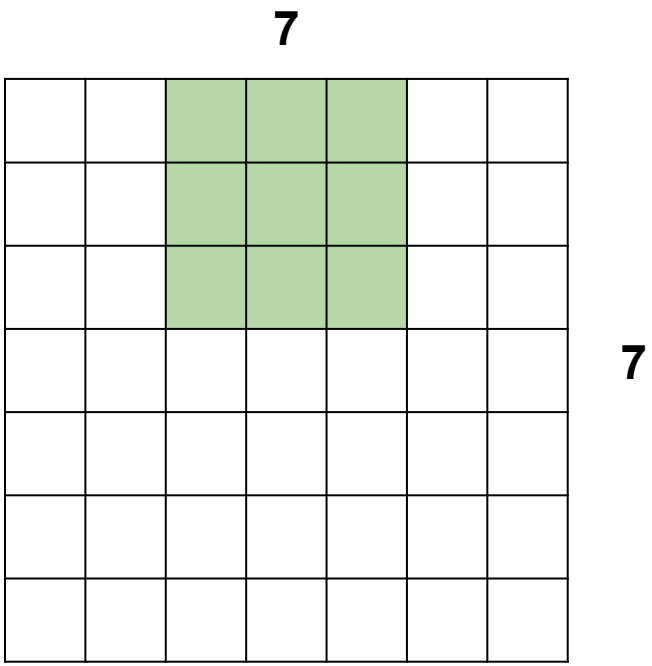


7x7 input (dimensiones) asumiendo filtro 3x3 **con stride 2**

Slides tomadas de cs231n (stanford) - Fei-Fei Li & Justin Johnson & Serena Yeung

Capa de Convolución

Una mirada más de cerca sobre las dimensiones:

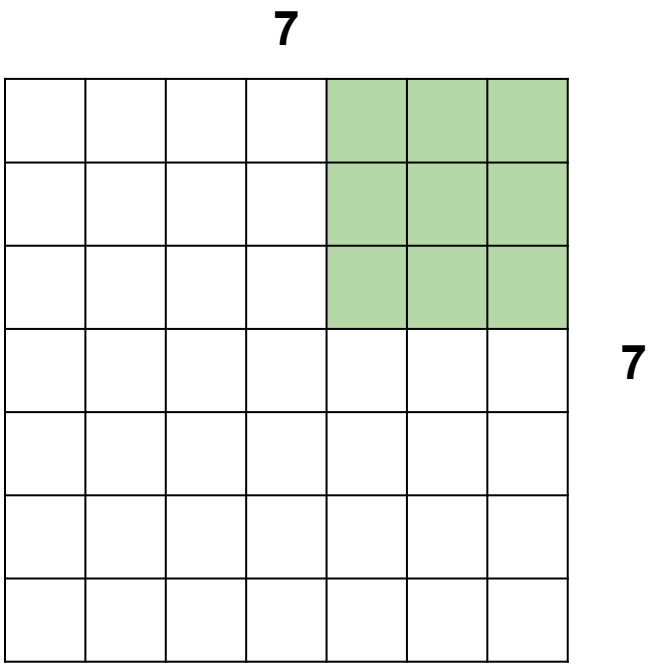


7x7 input (dimensiones) asumiendo filtro 3x3 **con stride 2**

Slides tomadas de cs231n (stanford) - Fei-Fei Li & Justin Johnson & Serena Yeung

Capa de Convolución

Una mirada más de cerca sobre las dimensiones:



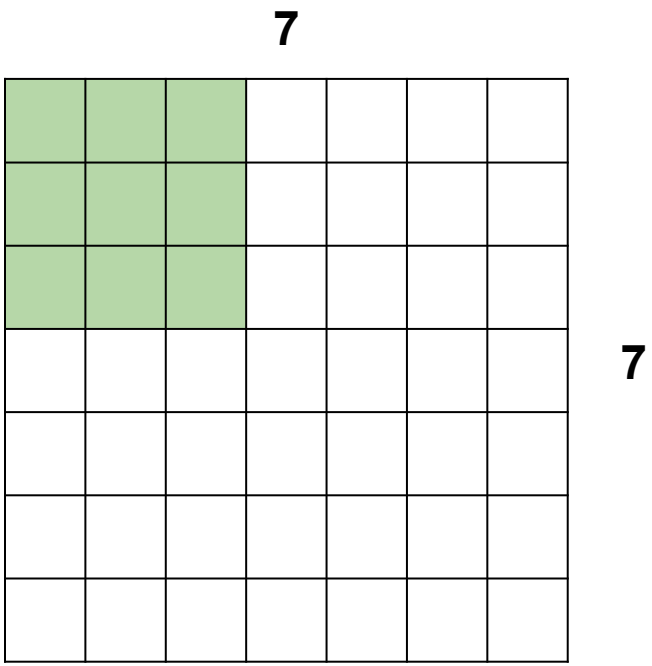
7x7 input (dimensiones) asumiendo filtro 3x3 **con stride 2**

→ **3x3 output**

Slides tomadas de cs231n (stanford) - Fei-Fei Li & Justin Johnson & Serena Yeung

Capa de Convolución

Una mirada más de cerca sobre las dimensiones:



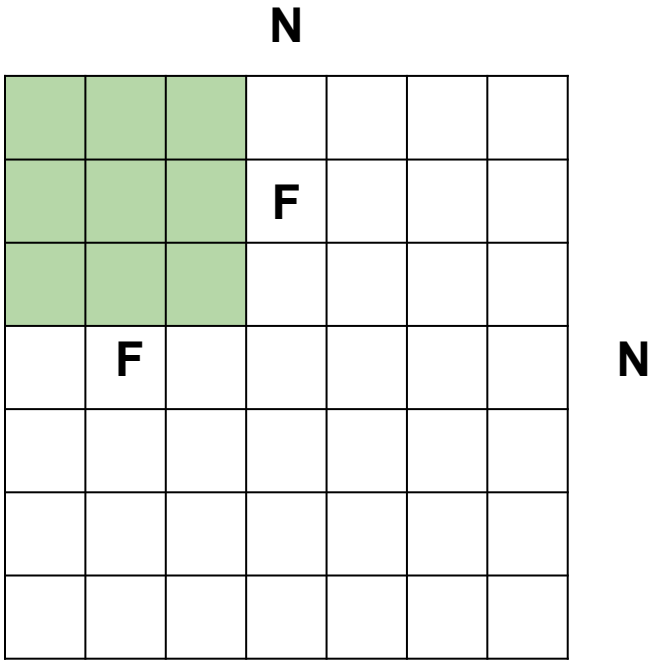
7x7 input (dimensiones) asumiendo filtro 3x3 **con stride 3?**

→ **No se puede!**
NO es posible aplicar filtro de 3x3 con stride de 3 en entrada de 7x7

Slides tomadas de cs231n (stanford) - Fei-Fei Li & Justin Johnson & Serena Yeung

Capa de Convolución

Una mirada más de cerca sobre las dimensiones:



Output size:
(N - F) / stride + 1

Ejemplos:
N = 7, F = 3

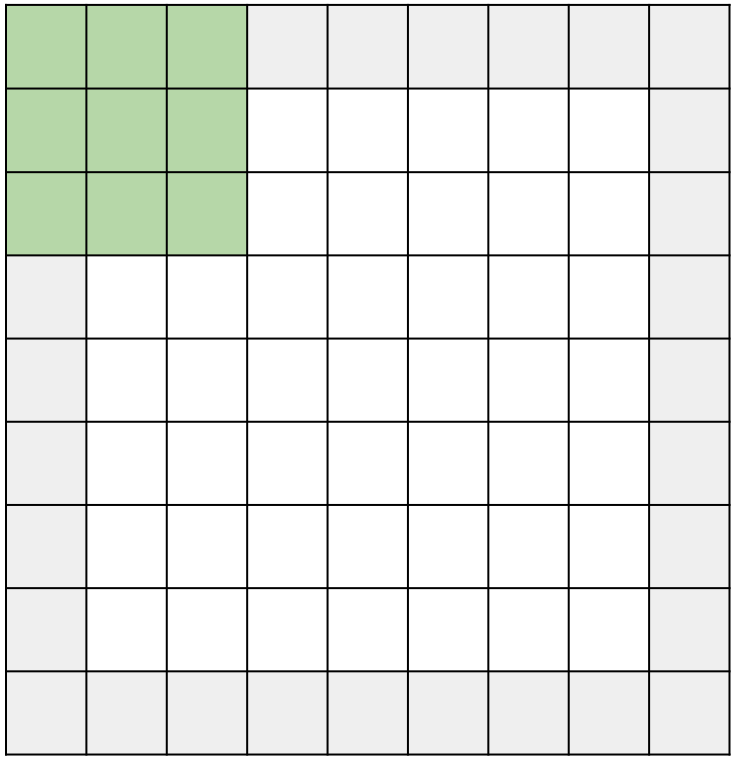
stride 1 → (7 - 3)/1 + 1 = 5
stride 2 → (7 - 3)/2 + 1 = 3
stride 3 → (7 - 3)/3 + 1 = 2.33 X

SOLUCIÓN: Rellenar con valores constantes, usualmente 0s (zero-padding)

Slides tomadas de cs231n (stanford) - Fei-Fei Li & Justin Johnson & Serena Yeung

Capa de Convolución

Una mirada más de cerca sobre las dimensiones:



Output size:
(N - F) / stride + 1

Ejemplos:
N = 7, F = 3

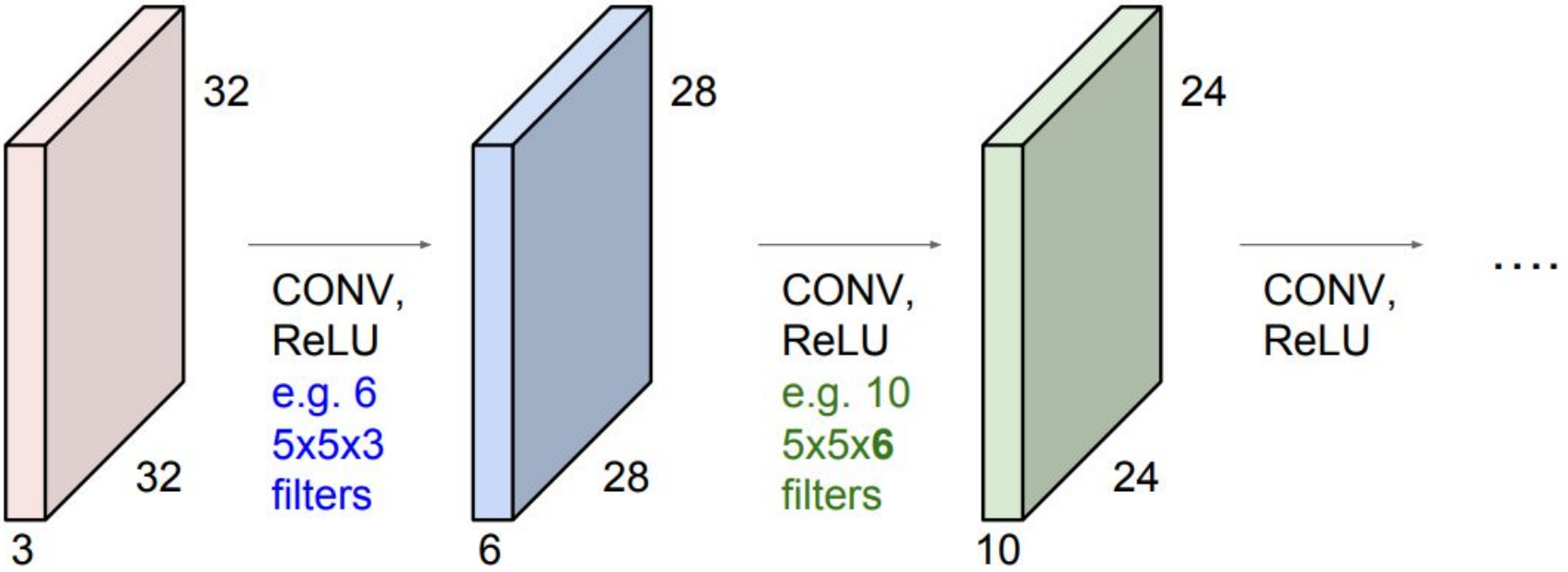
stride 1 → (7 - 3)/1 + 1 = 5
stride 2 → (7 - 3)/2 + 1 = 3
stride 3 → (7 - 3)/3 + 1 = 2.33 X

SOLUCIÓN: Rellenar con valores constantes, usualmente 0s (zero-padding)

Slides tomadas de cs231n (stanford) - Fei-Fei Li & Justin Johnson & Serena Yeung

Capa de Convolución

IMPORTANTE: Una entrada de 32x32 en convoluciones repetidas con filtro de 5x5 reduce super rápido la dimensionalidad! (32 → 28 → 24…)
Cuidado que reducir dimensionalidad muy rápido no suele funcionar bien!



Slides tomadas de cs231n (stanford) - Fei-Fei Li & Justin Johnson & Serena Yeung

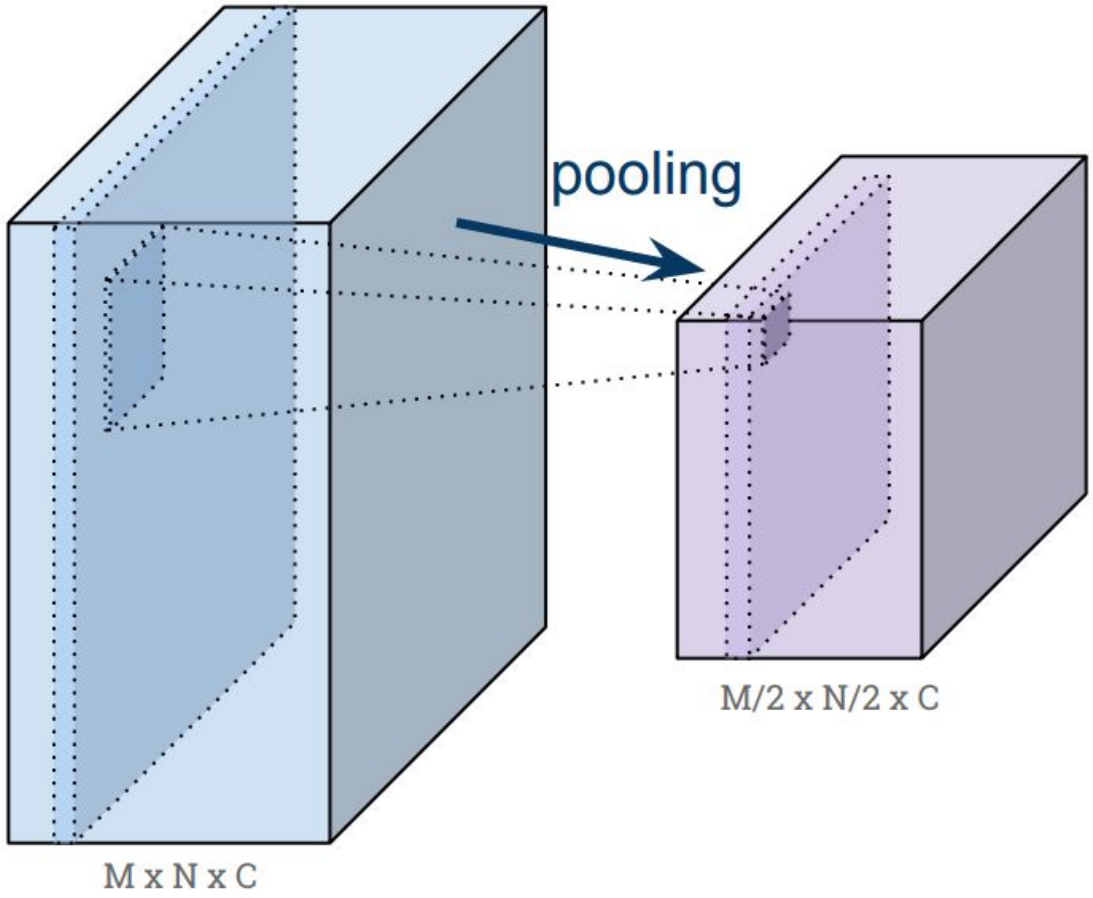
Capa de Convolución: Observaciones

- **Conexiones Esparsas:** Conectividad de un elemento en la salida está dada por el soporte del filtro (en general pequeño: 3×3 o 5×5).
- **Pesos compartidos:** A diferencia de capas totalmente conectadas, los pesos de las capas de convolución (filtros) se reutilizan en varios elementos de la entrada.
- **Equivarianza:** Si se traslada la entrada, se traslada la salida.
- **Representaciones:** Capas de convolución permiten manejar datos de diferente tamaño (e.g., imágenes), sin necesidad de cambiar la arquitectura.

2. Max Pooling

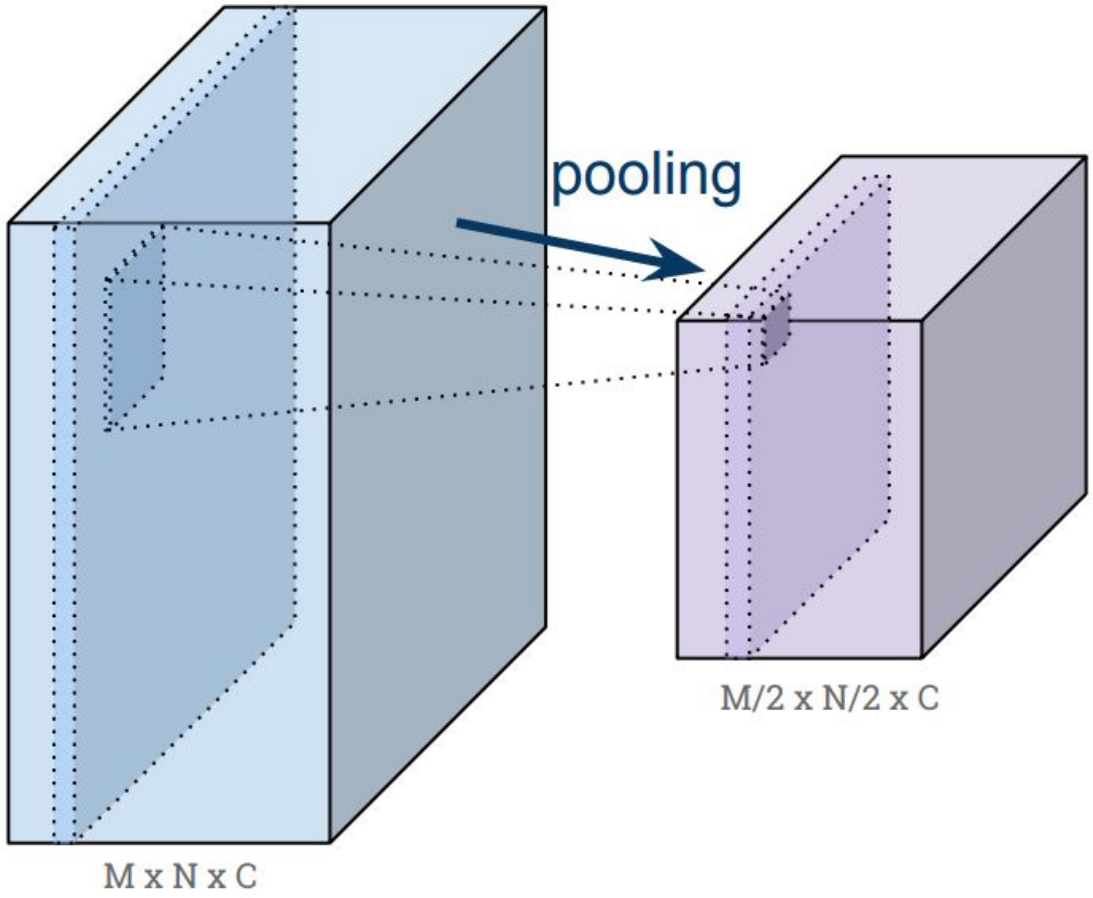
Capa de Pooling

- Comprime (sub-muestreo) de la representación
- Opera en cada mapa de activación (canal) por separado



Capa de Pooling

- Comprime (sub-muestreo) de la representación
- Opera en cada mapa de activación (canal) por separado



Capa de Pooling

- Comprime (sub-muestreo) de la representación
- Opera en cada mapa de activación (canal) por separado

Capa de Pooling

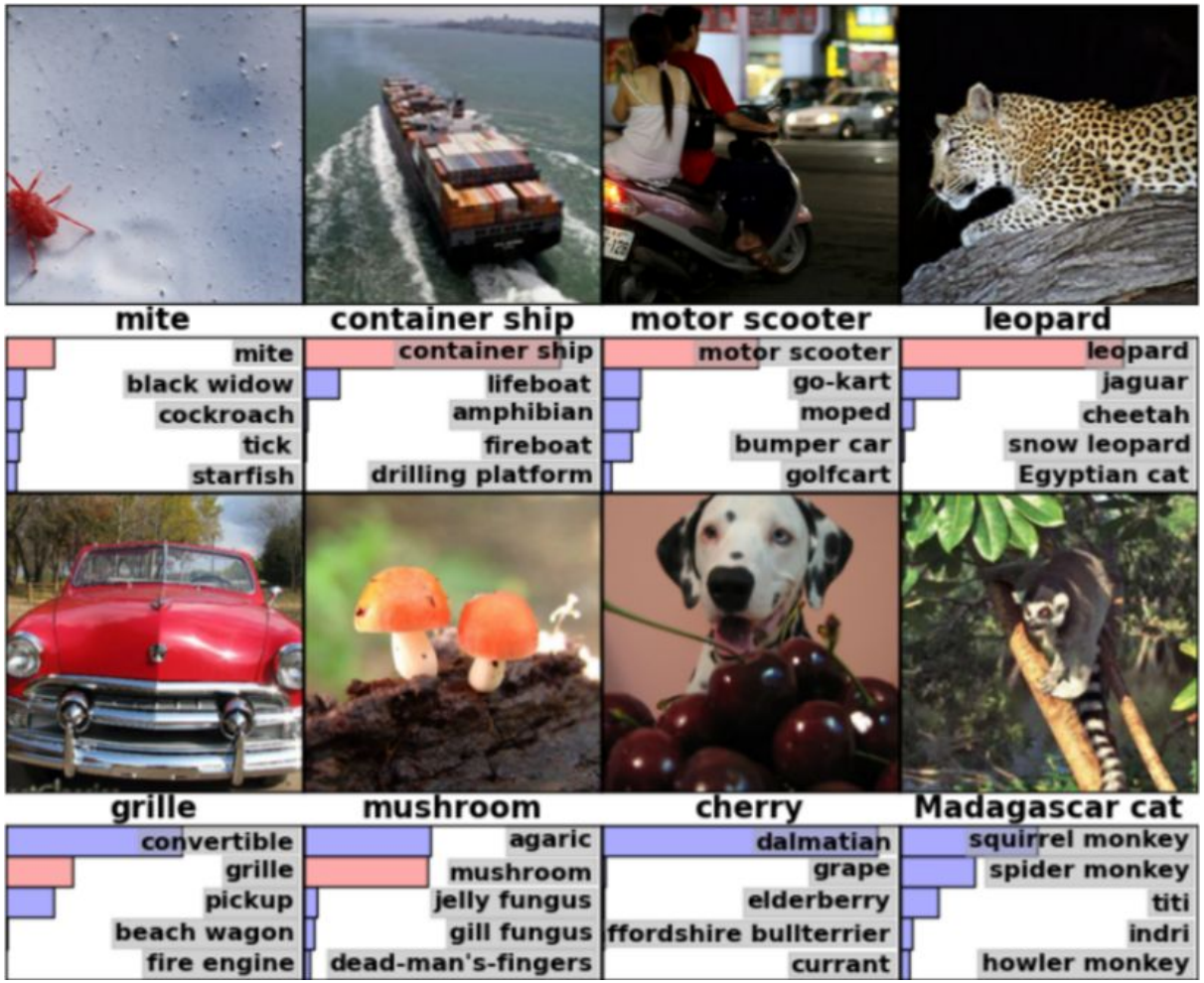
- Comprime (sub-muestreo) de la representación
- Opera en cada mapa de activación (canal) por separado

Convolución + Pooling: observaciones

- **Convolución** puede verse como un caso particular de una capa totalmente conectada pero con muchos menos parámetros
- **Pooling** aporta invariancia a traslación, no tiene parámetros
- **Atención**: Convolución + Pooling puede generar subajuste (se puede perder información importante)
- Genera bloques re-utilizables y re-ensamblables para formar arquitecturas más complejas.
- Introduce nuevos parámetros para hiper-tunear (hiper parámetros): kernel size y stride.

3. Aplicaciones

Clasificación



Krizhevsky, Alex, Ilya Sutskever, Geoffrey E. Hinton.
”Imagenet classification with deep convolutional neural networks.”, NIPS 2012. (28k citas)

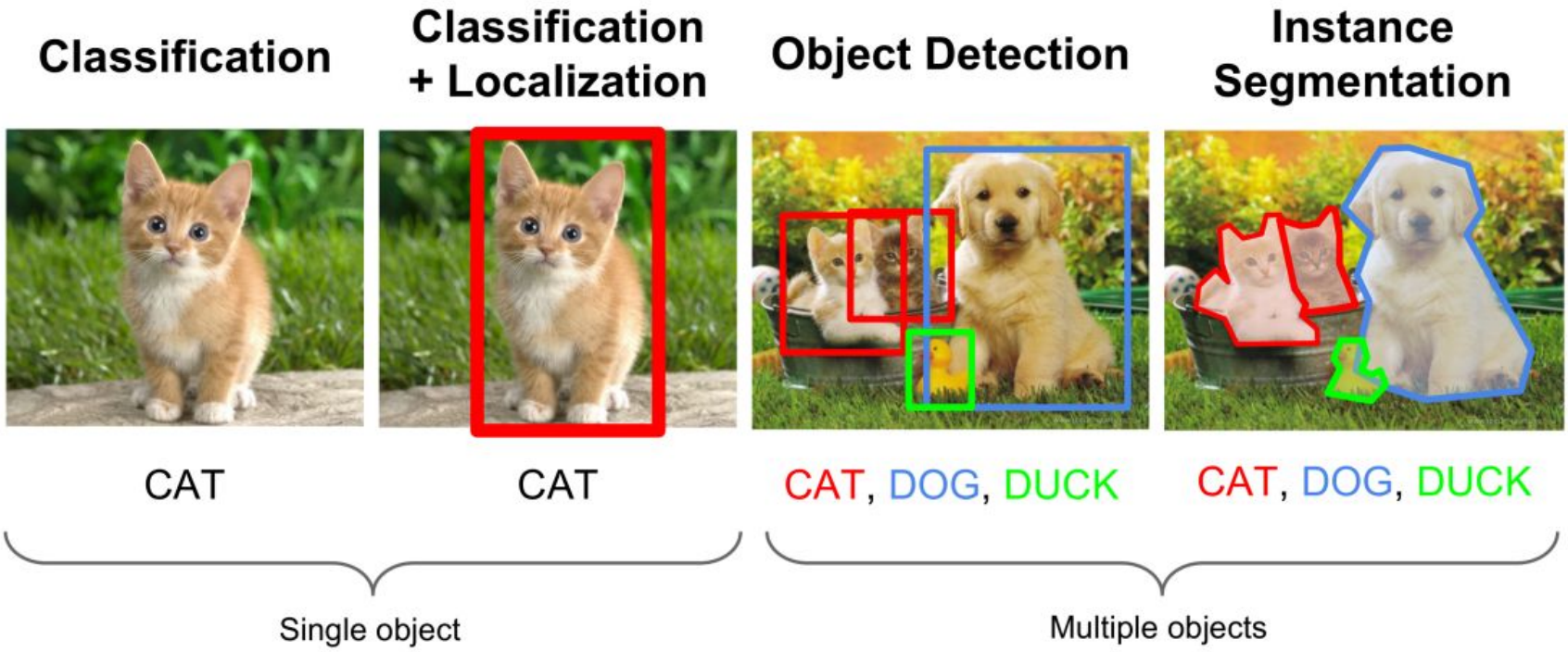
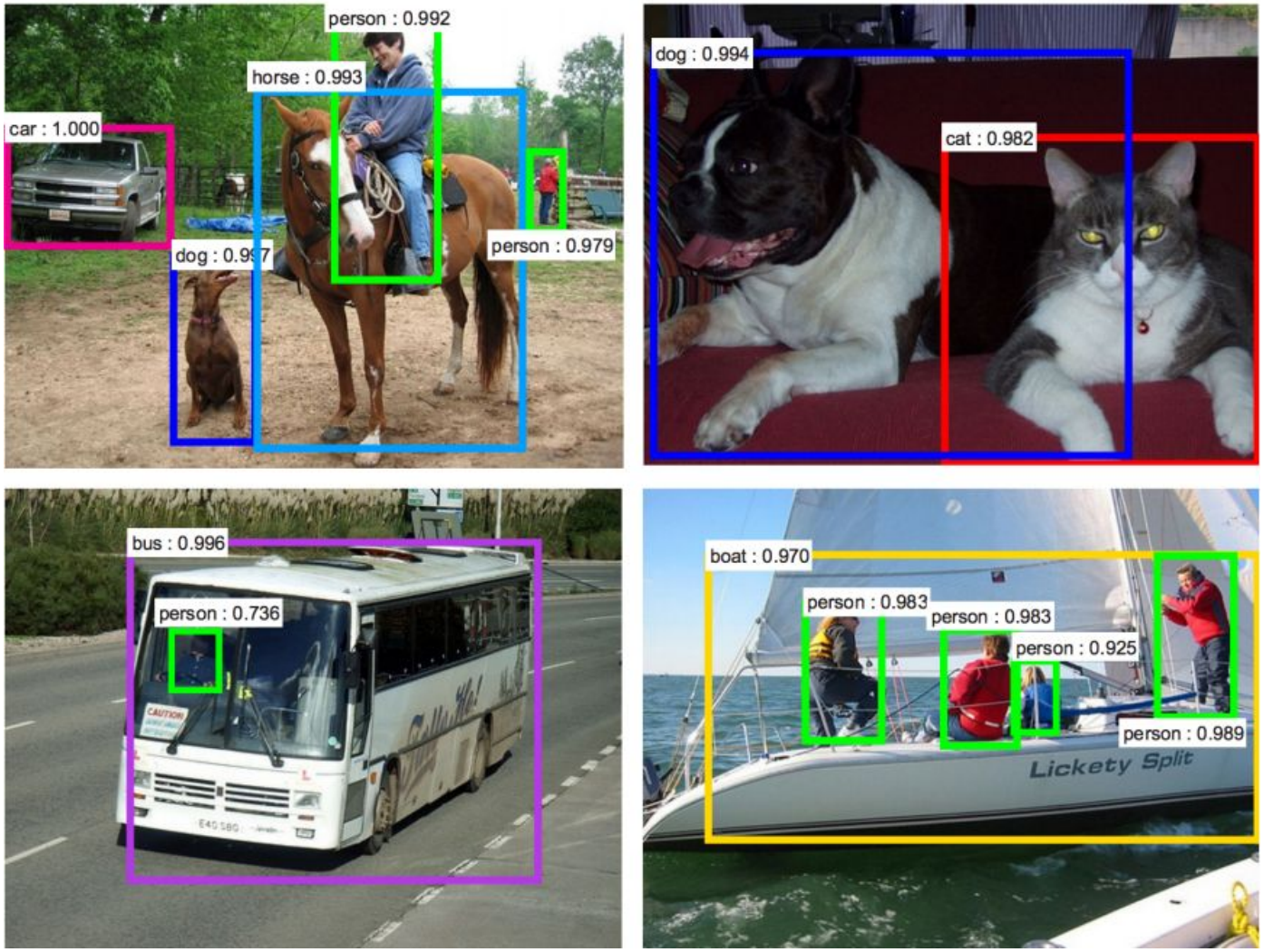


Imagen tomada de cs231n (Stanford) - Fei-Fei Li & Justin Johnson & Serena Yeung

Detección



Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun.
"Faster R-CNN: Towards real-time object detection with region proposal networks." NIPS 2015 (4.6k citas)

Segmentación

Figure 4. More results of **Mask R-CNN** on COCO test images, using ResNet-101-FPN and running at 5 fps, with 35.7 mask AP (Table 1).

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick , “Mask R-CNN.”, ICCV 2017

Detección de Pose (Pose estimation)

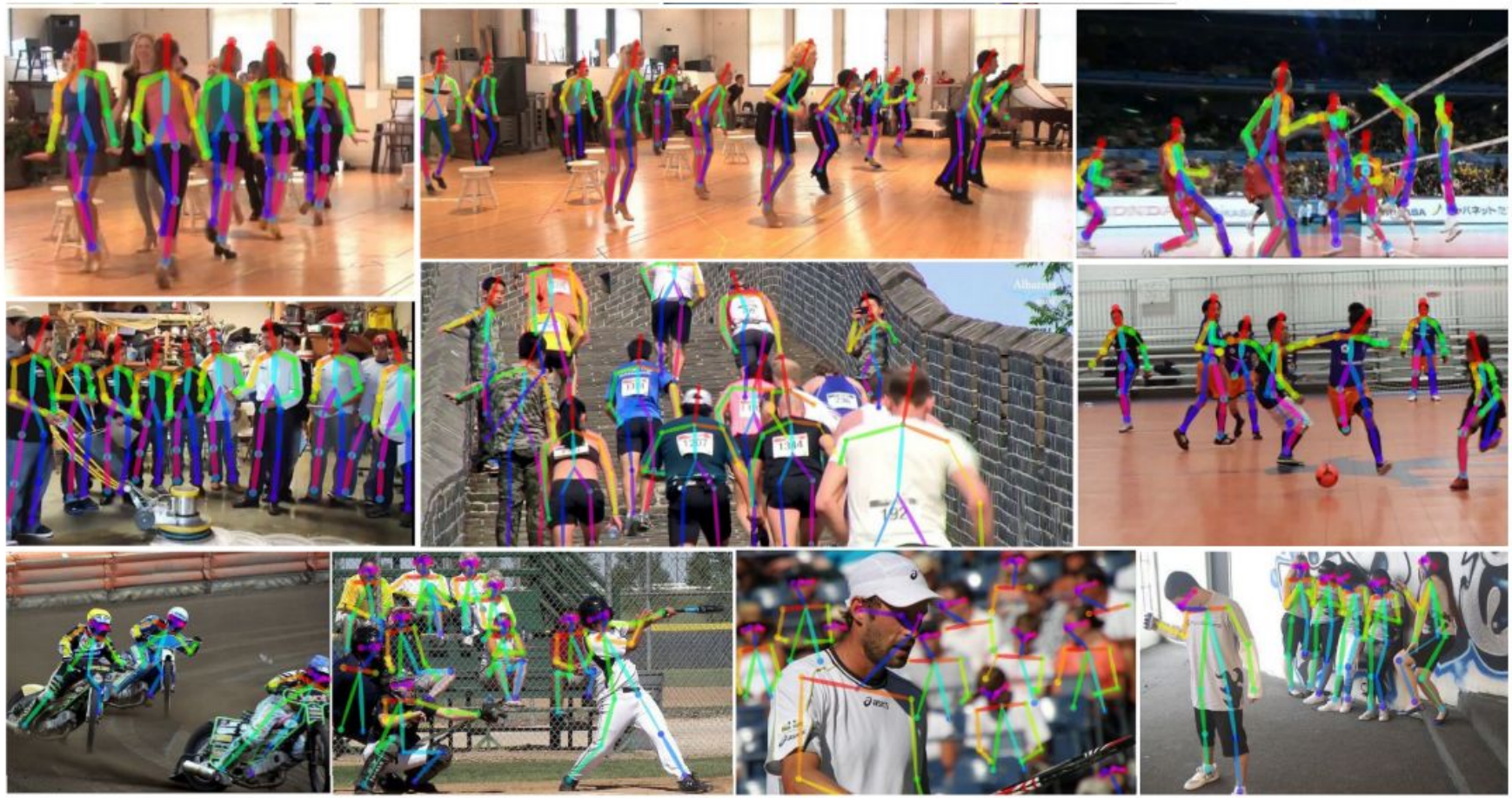


Figure 10. Results containing viewpoint and appearance variation, occlusion, crowding, contact, and other common imaging artifacts.

Zhe Cao and Tomas Simon and Shih-En Wei and Yaser Sheikh
"Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields.", CVPR 2017

Image Captioning

No errors

A white teddy bear sitting in the grass

A man riding a wave on top of a surfboard

Minor errors

A man in a baseball uniform throwing a ball

A cat sitting on a suitcase on the floor

Somewhat related

A woman is holding a cat in her hand

A woman standing on a beach holding a surfboard

Image Captioning

[Vinyals et al., 2015]
[Karpathy and Fei-Fei, 2015]

All images are CC0 Public domain:
https://pixabay.com/en/luggage-antique-cat-1643010/
https://pixabay.com/en/teddy-plush-bears-cute-teddy-bear-1623436/
https://pixabay.com/en/surf-wave-summer-sport-litoral-1668716/
https://pixabay.com/en/woman-female-model-portrait-adult-983967/
https://pixabay.com/en/handstand-lake-meditation-496008/
https://pixabay.com/en/baseball-player-shortstop-infield-1045263/

Captions generated by Justin Johnson using Neuraltalk2

Slide tomada de cs231n (Stanford) - Fei-Fei Li & Justin Johnson & Serena Yeung

4. Arquitecturas

Lenet-5 1998

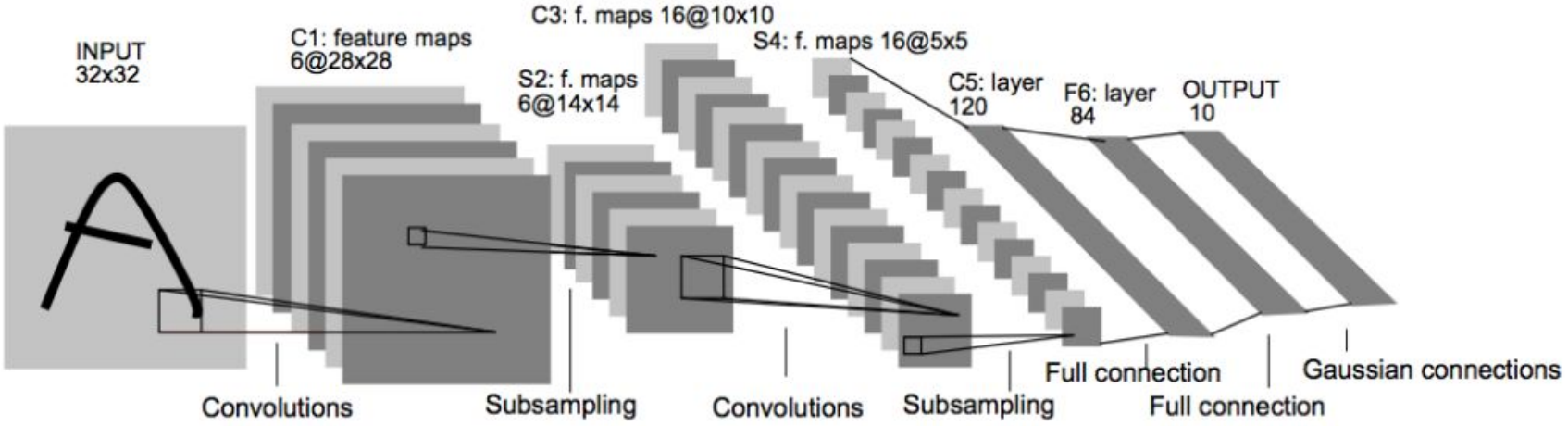


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. “Gradient-based learning applied to document recognition.” Proc. IEEE (1998). +info: https://yohanes.gultom.me/understanding-lenet-lecun-1998

Large Scale Visual Recognition Competition

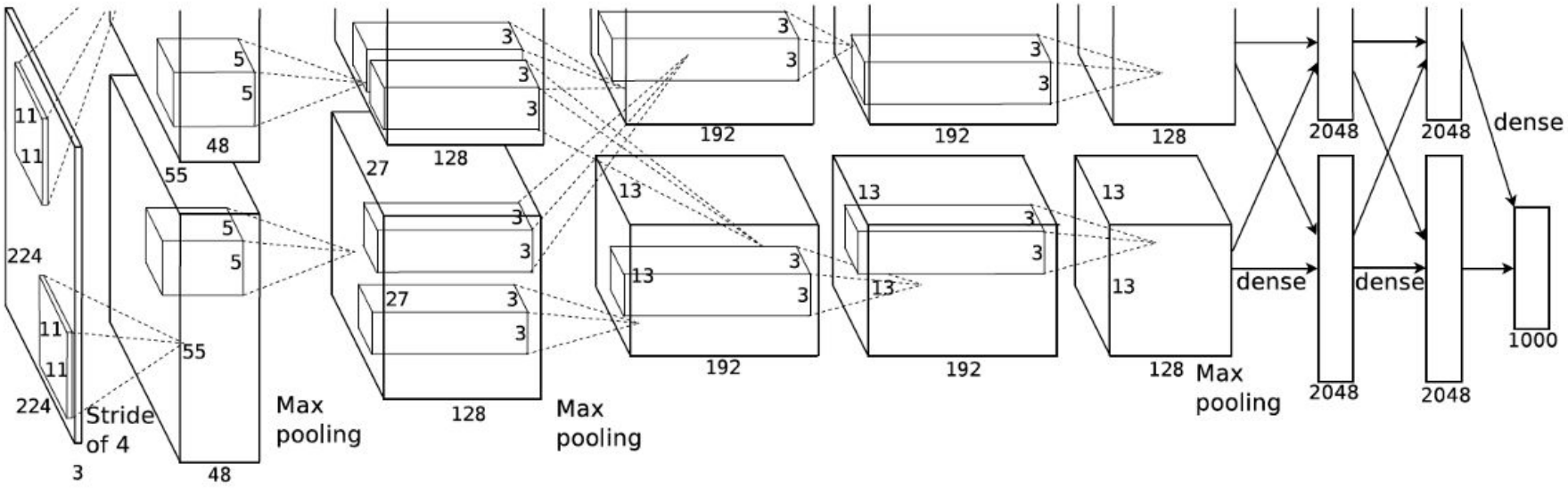
- 1000 categorías de imágenes, 1.4 millones de imágenes de entrenamiento
- Respuesta correcta entre las primeras 5 imágenes
- Krizhevsky, Sutskever, Hinton, 2012 [Supervision/AlexNet].

2012 Teams	%error
Supervision (Toronto)	15.3
ISI (Tokyo)	26.1
VGG (Oxford)	26.9
XRCE/INRIA	27.0
UvA (Amsterdam)	29.6
INRIA/LEAR	33.4

2013 Teams	%error
Clarifai (NYU spinoff)	11.7
NUS (singapore)	12.9
Zeiler-Fergus (NYU)	13.5
A. Howard	13.5
OverFeat (NYU)	14.1
UvA (Amsterdam)	14.2
Adobe	15.2
VGG (Oxford)	15.2
VGG (Oxford)	23.0

2014 Teams	%error
GoogLeNet	6.6
VGG (Oxford)	7.3
MSRA	8.0
A. Howard	8.1
DeeperVision	9.5
NUS-BST	9.7
TTIC-ECP	10.2
XYZ	11.2
UvA	12.1

AlexNet - 2012



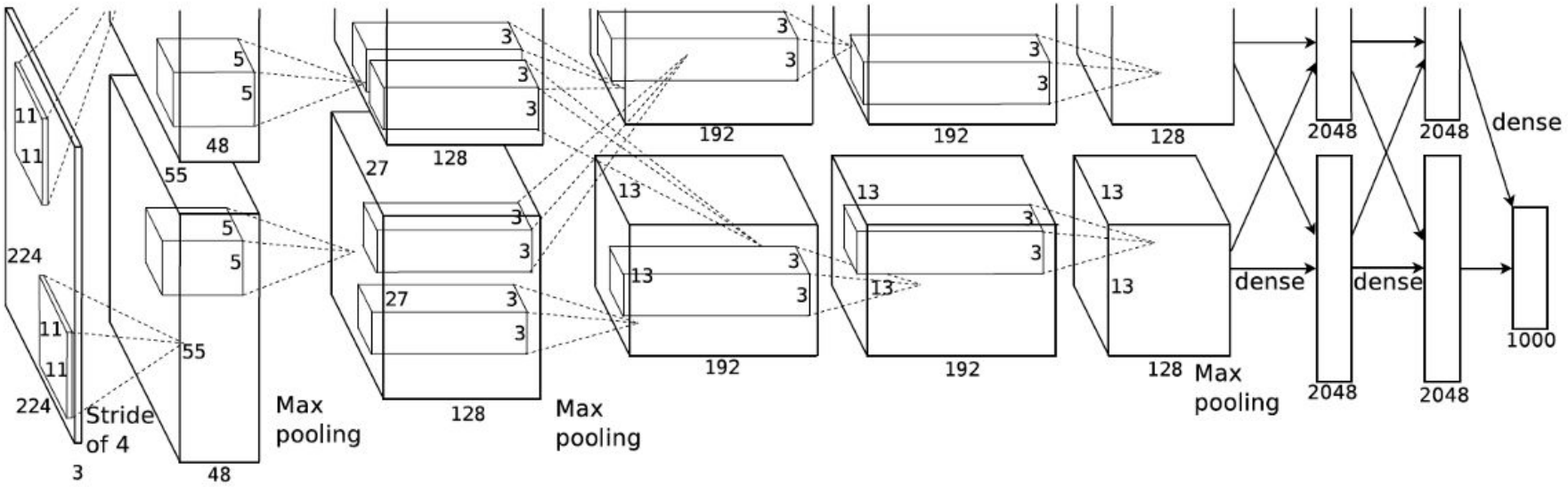
La foto de la arquitectura parece “rota”, es porque la arquitectura original incluye otro bloque idéntico al expuesto para paralelizar en 2 GPUs todo el procesamiento.

Krizhevsky, Alex, Ilya Sutskever, Geoffrey E. Hinton.
”Imagenet classification with deep convolutional neural networks.”, NIPS 2012. (28k citas)

AlexNet - 2012

Capa	Tamaño	Parámetros	Operaciones
INPUT	227x227x3		
CONV1	55x55x96	35K	105Mflops
POOL1	27x27x96		
CONV2	27x27x256	307K	223Mflops
POOL2	13x13x256		
CONV3	13x13x384	884 K	149Mflops
CONV4	13x13x384	1.3 M	224MFlops
CONV5	13x13x256	442 K	74Mflops
POOL3	6x6x256		
FC	4096	37M	37MFlops
FC	4096	16M	16Mflops
FC	1000	4M	4MFlops

AlexNet - 2012



- Revolución en el campo de las redes neuronales
- Detalles
  - Uso intensivo de data augmentation
  - Dropout (0.5)
  - Batch Size 128
  - SGD Momentum 0.9
  - LR 1e-2 con learning rate decay
  - L2 weight decay 5e-4

VGG-16 - 2014

- Introduce estudio de profundidad de la red
- Convoluciones de 3x3 y stride de 1 + Pooling de 2x2 con stride 2
- Utiliza menos parámetros pero igualmente es super poderosa

Krizhevsky, Alex, Ilya Sutskever, Geoffrey E. Hinton.
”Imagenet classification with deep convolutional neural networks.”, NIPS 2012. (28k citas)

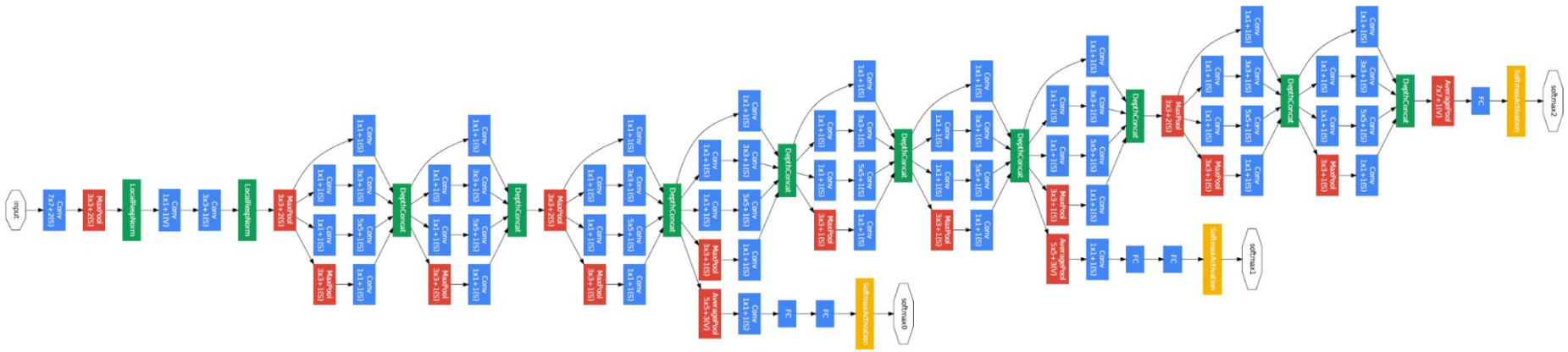
AlexNet - 2012

Capa	Tamaño	Memoria	Parámetros
INPUT	224x224x3	224*224*3=150K	0
CONV3-64	224x224x64	224*224*64 = 3,2M	(3*3*3)*64 = 1.728
CONV3-64	224x224x64	224*224*64 = 3,2M	(3*3*64)*64 = 36.864
POOL2	112x112x64	112*112*64 = 800K	0
CONV3-128	112x112x128	112*112*128 = 1,6M	(3*3*64)*128 = 73.728
CONV3-128	112x112x128	112*112*128 = 1,6M	(3*3*128)*128 = 147.456
POOL2	56x56x128	56*56*128 = 400K	0
CONV3-256	56x56x256	56*56*256 = 800K	(3*3*128)*256 = 294.912
CONV3-256	56x56x256	56*56*256 = 800K	(3*3*256)*256 = 589.824
CONV3-256	56x56x256	56*56*256 = 800K	(3*3*256)*256 = 589.824
POOL2	28x28x256	28*28*256 = 200K	0
CONV3-512	28x28x512	28*28*512 = 400K	(3*3*256)*512 = 1.179.648
CONV3-512	28x28x512	28*28*512 = 400K	(3*3*512)*512 = 2.359.296
CONV3-512	28x28x512	28*28*512 = 400K	(3*3*512)*512 = 2.359.296
POOL2	14x14x512	14*14*512 = 100K	0
CONV3-512	14x14x512	14*14*512 = 100K	(3*3*512)*512 = 2.359.296
CONV3-512	14x14x512	14*14*512 = 100K	(3*3*512)*512 = 2.359.296
CONV3-512	14x14x512	14*14*512 = 100K	(3*3*512)*512 = 2.359.296
POOL2	7x7x512	7*7*512 = 25K	0
FC	1x1x4096	4096	7*7*512*4096 = 102.760.448
FC	1x1x4096	4096	4096*4096 = 16.777.216
FC	1x1x1000	1000	4096*1000 = 4.096.000
		15,2M ×4 bytes = 60,8MB	138M ×4 bytes = 552MB

VGG-16 - 2014

- Redes profundas difíciles de entrenar
- Detalles
  - Data augmentation
  - Softmax en capa de salida
  - SGD Momentum 0.9
  - Batch Size 128
  - Dropout 0.5
  - L2 weight decay 5e-4
  - LR 10e-2 con LR decay
  - 74 epochs

GoogLeNet 2014



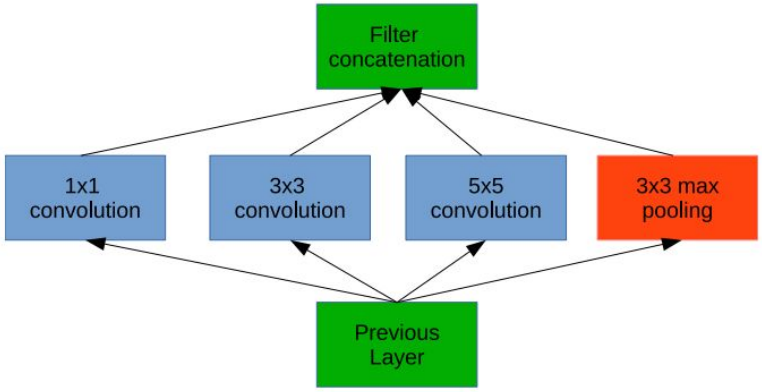
Red diseñada para ser eficiente en términos de cómputo y memoria:

- 1500 millones de sumas y multiplicaciones
- 12 veces menos parámetros que AlexNet
- 9 módulos de inception en cascada

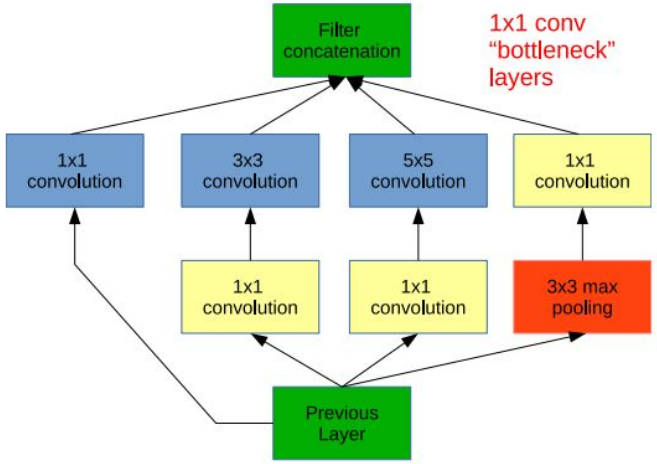
Otros:

- Global Average Pooling
- Bloques inception y clasificadores auxiliares para combatir vanishing gradient

GoogLeNet 2014



Módulo Inception: versión naive



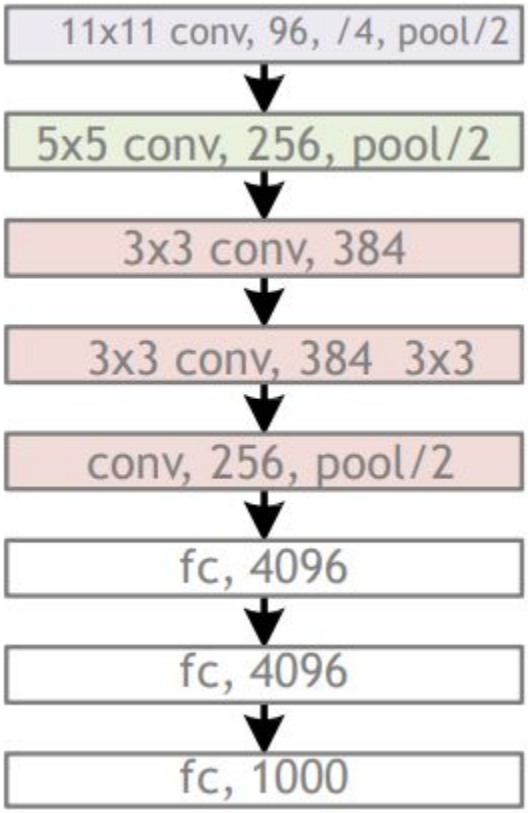
Módulo Inception con reducción de dimensionalidad

- Filtrado multi-escala:
  - convoluciones de 1x1, 3x3 y 5x5
  - pooling de 3x3
- Las salidas se concatenan a un solo volúmen

ResNet - 2015

Revolución de la profundidad:

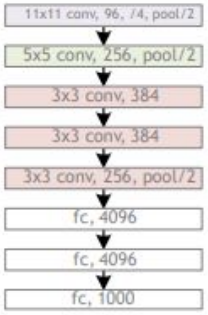
AlexNet, 8 layers (ILSVRC 2012)



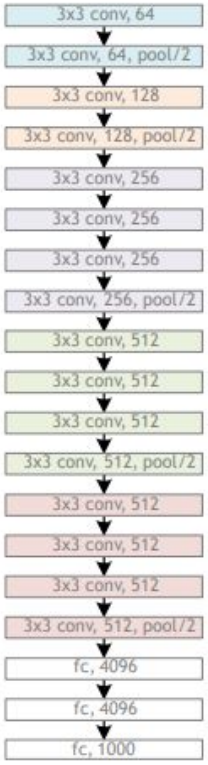
ResNet - 2015

Revolución de la profundidad:

AlexNet, 8 layers (ILSVRC 2012)



VGG, 19 layers (ILSVRC 2014)

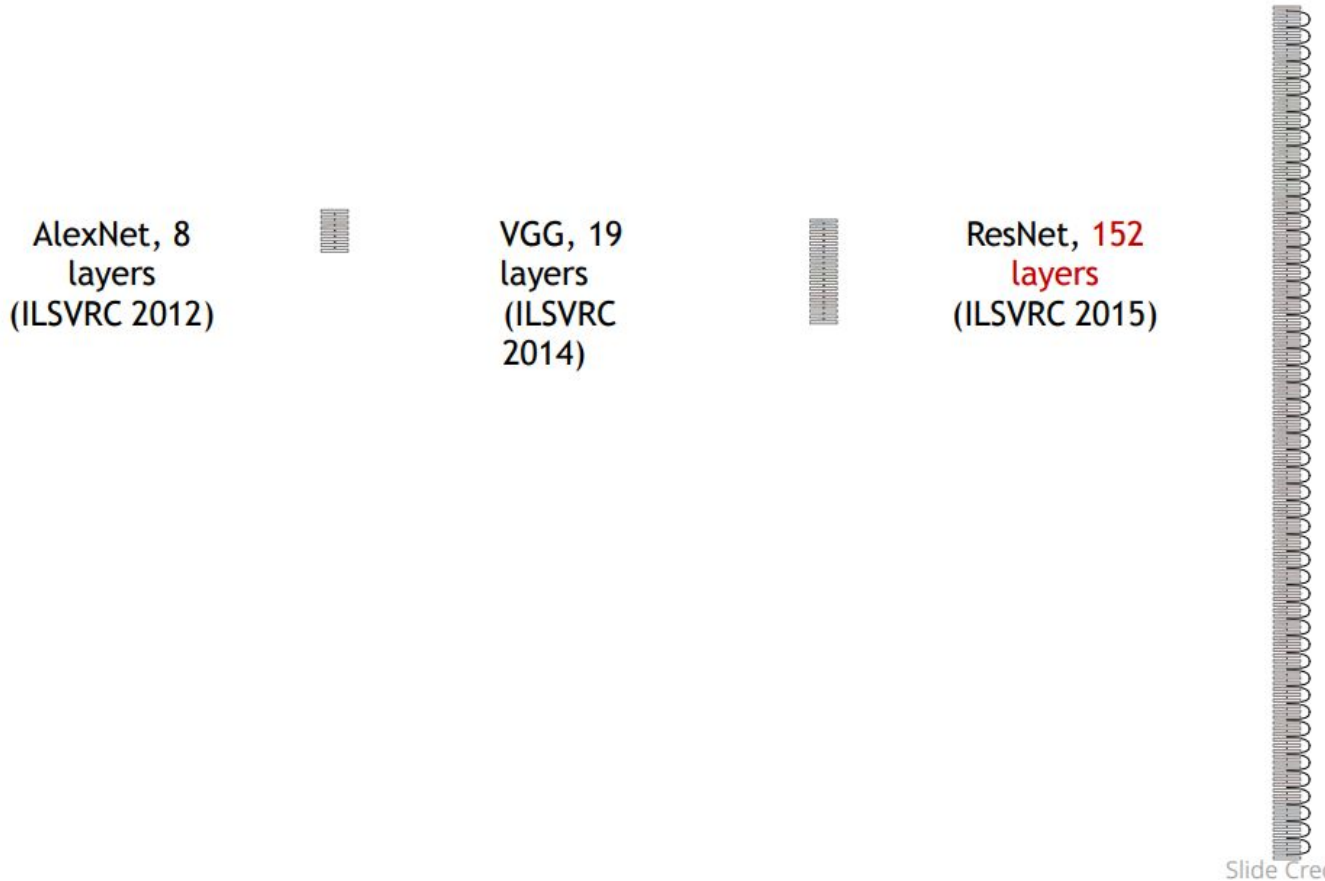


GoogleNet, 22 layers (ILSVRC 2014)

Slide Credit: He et al. (MSRA)

ResNet - 2015

Revolución de la profundidad:

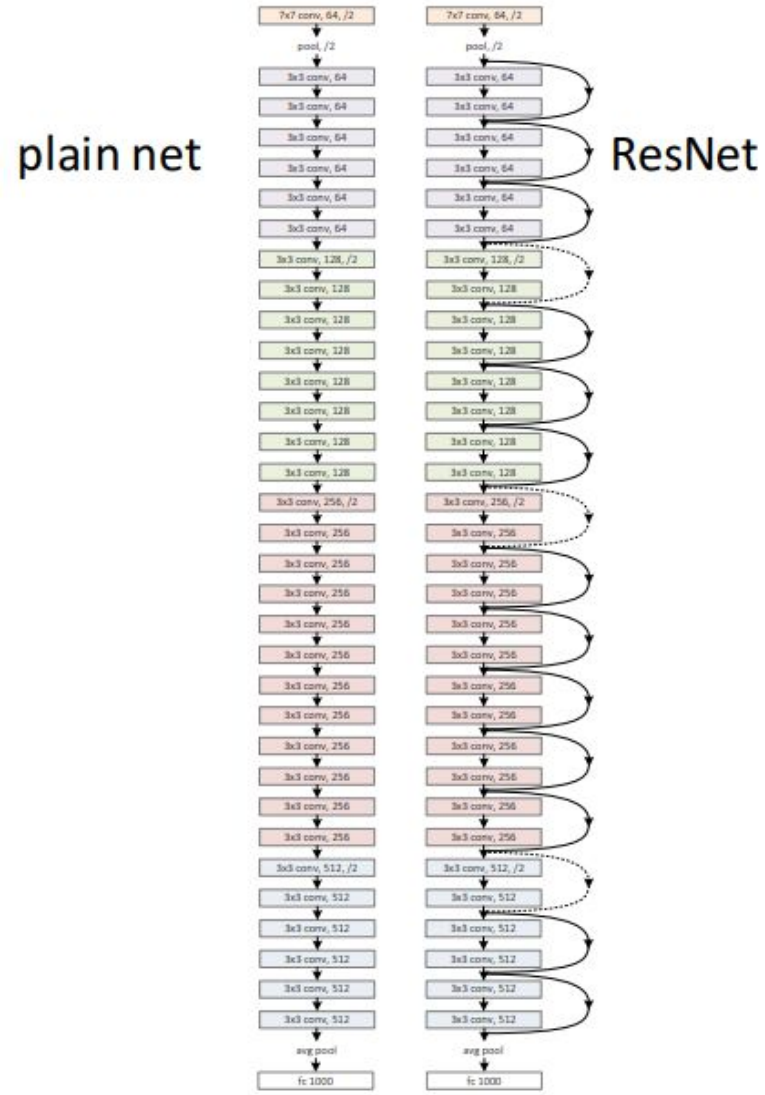


Slide Credit: He et al. (MSRA)

ResNet - 2015

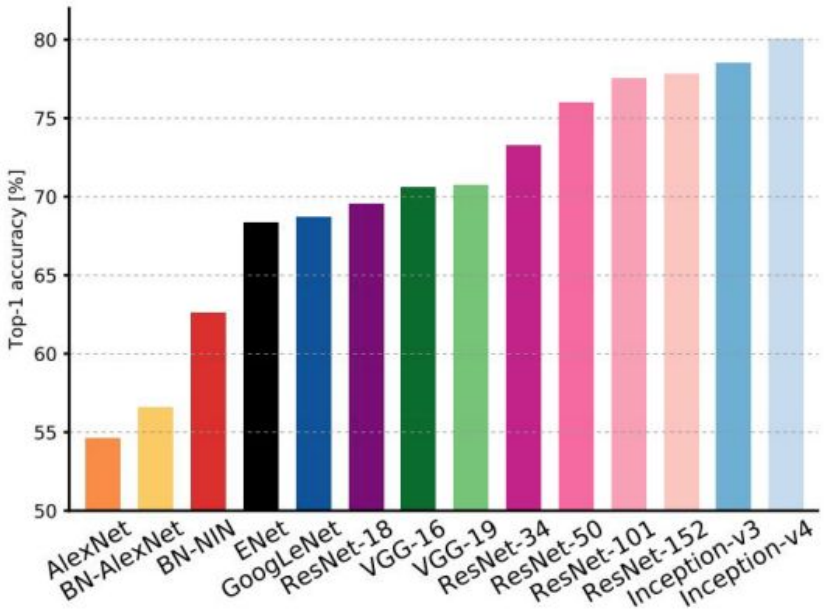
Introduce varias mejoras:

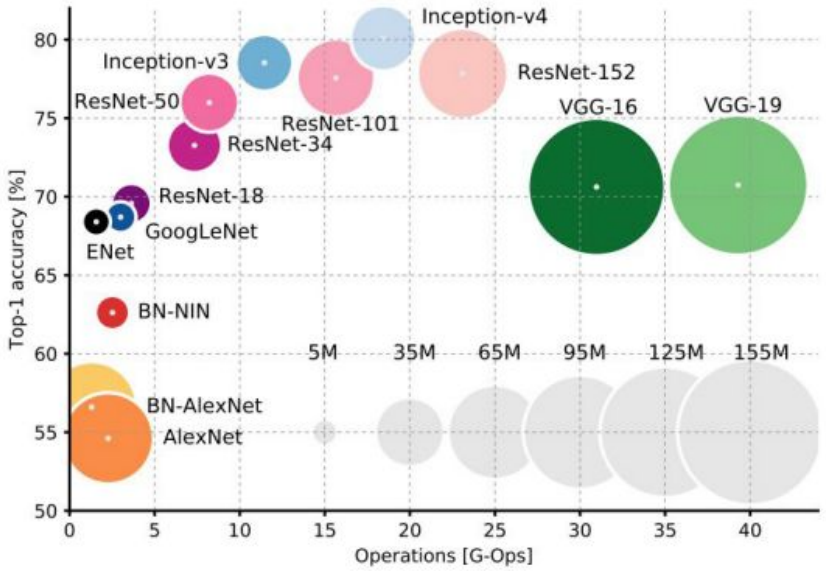
- Arquitectura profunda
- Convoluciones 3x3 todas iguales
- Sin dropout
- Bloques residuales para pasar información de capas tempranas a capas avanzadas en la red



IDATHA

Comparación de las arquitecturas





Canziani, A., Paszke, A. and Culurciello, E.,
An analysis of deep neural network models for practical applications. arXiv preprint, 2016

5. Conclusiones

Conclusiones

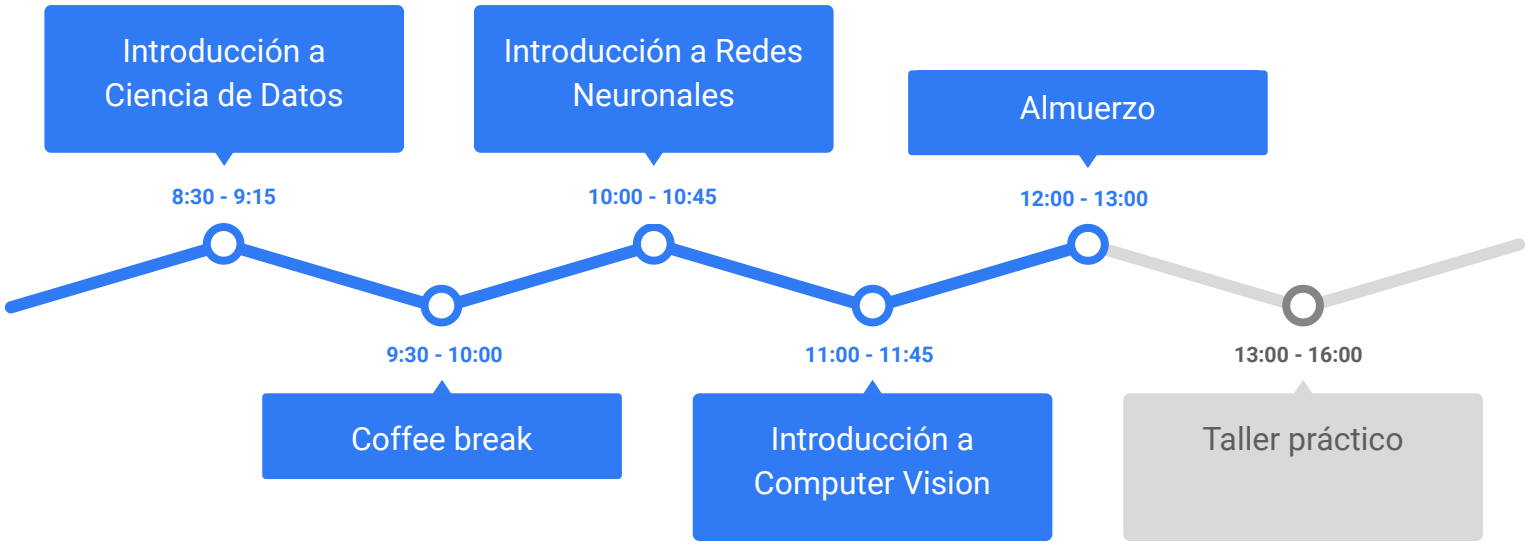
Algunas conclusiones:

- En Computer Vision se combina Deep Learning (presentación anterior) con técnicas particulares para procesar imágenes de forma eficiente, como convoluciones.
- Las redes convolucionales permiten extraer características de las imágenes de menor a mayor nivel, reduciendo la complejidad del modelo en cantidad de parámetros y permitiendo generalizar el aprendizaje.
- Las arquitecturas convolucionales son robustas a cambios en el dominio del problema.
- Arquitecturas actuales como Inception, ResNet logran una performance más que aceptable en tareas de reconocimiento de objetos.
- La cantidad de parámetros en modelos actuales es inmensa. Implica cientos de miles de datos para entrenamiento y una capacidad de cómputo solo disponible en la nube.

REFERENCIAS

- CS231n: Deep Learning for Computer Vision - Stanford
  http://cs231n.stanford.edu/
- CS231n: Convolutional Neural Networks: Architectures, Convolution / Pooling Layers
  https://cs231n.github.io/convolutional-networks/
- CS231n: Understanding and Visualizing Convolutional Neural Networks
  https://cs231n.github.io/understanding-cnn/
- CS231n: Transfer Learning and Fine-tuning Convolutional Neural Networks
  https://cs231n.github.io/transfer-learning/
- Fing - DLvis: Clase 6 - Redes Convolucionales
  https://iie.fing.edu.uy/~pmuse/DLVIS2020/DLVIS2020_clase6.pdf
- Fing - DLvis: Clase 11 - Arquitecturas de redes para Computer Vision
  https://eva.fing.edu.uy/pluginfile.php/318326/mod_resource/content/2/c11.pdf
- Fing - DLvis: Clase 12 - Clasificación de Objetos, Segmentación y otros
  https://iie.fing.edu.uy/~jlezama/DLVIS2020/slides/c12_light.pdf

AGENDA



IDATHA