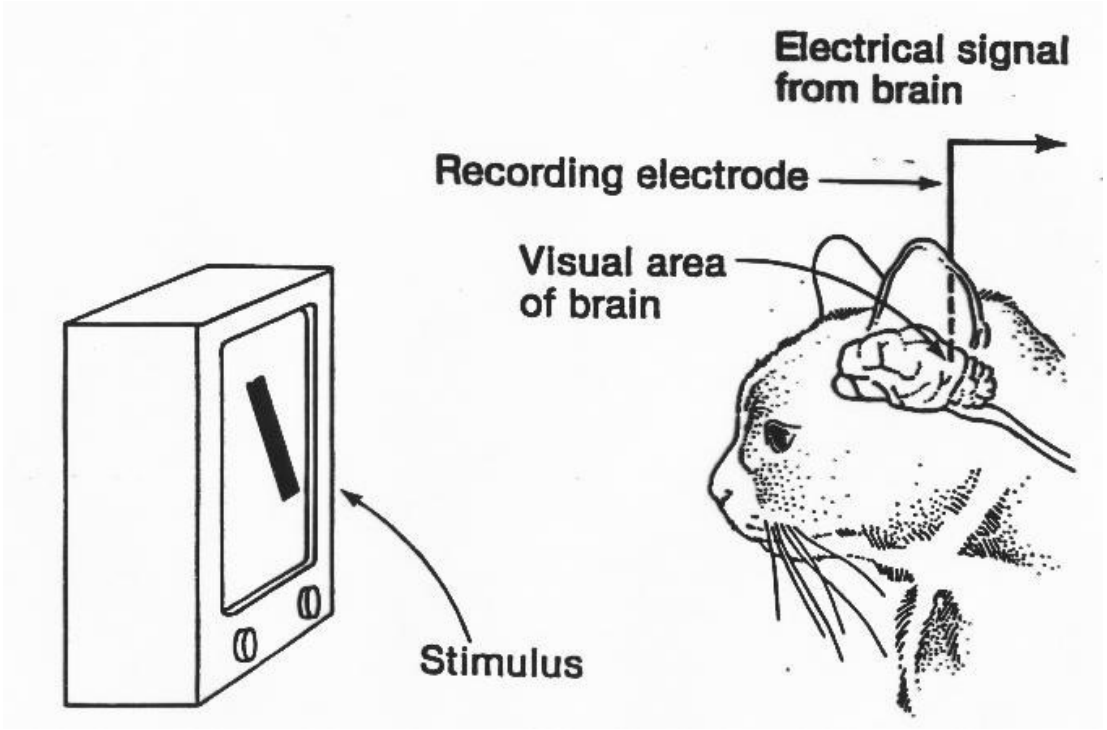
Visión Computacional

Ivan Sipiran

La revolución es la convolución

Pero no una nueva revolución

Hubel and Wiesel, 1959



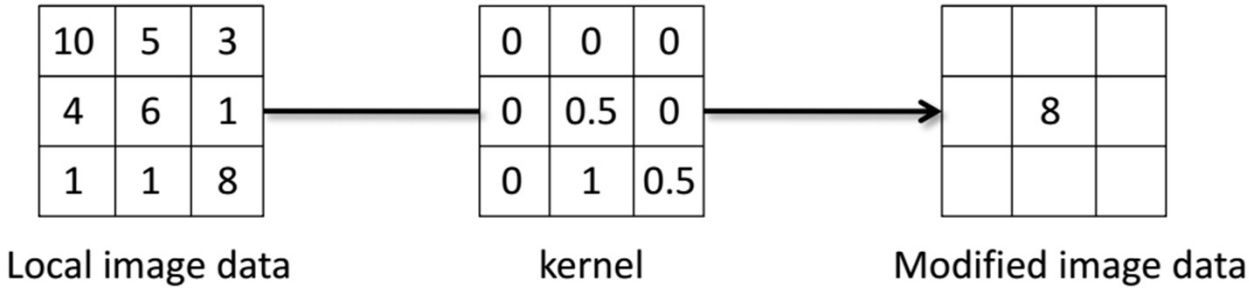
Los experimentos mostraron que ciertas células en el córtex visual responden a estímulos específicos. Formalmente, la respuesta es obtenida por una operación de convolución entre la señal de entrada y la funcionalidad de la célula.

Qué es convolución?

- Es una operación lineal que transforma el contenido de la imagen usando un elemento estructural llamado *kernel*.
- Técnicamente es una función lineal

Qué es convolución?

- En general, un píxel es reemplazado como la combinación de sus vecinos
- El kernel contiene los pesos para computar esa combinación



Convolución

- Sea F una imagen, H (size $2k+1, 2k+1$) un kernel, y G la imagen de salida

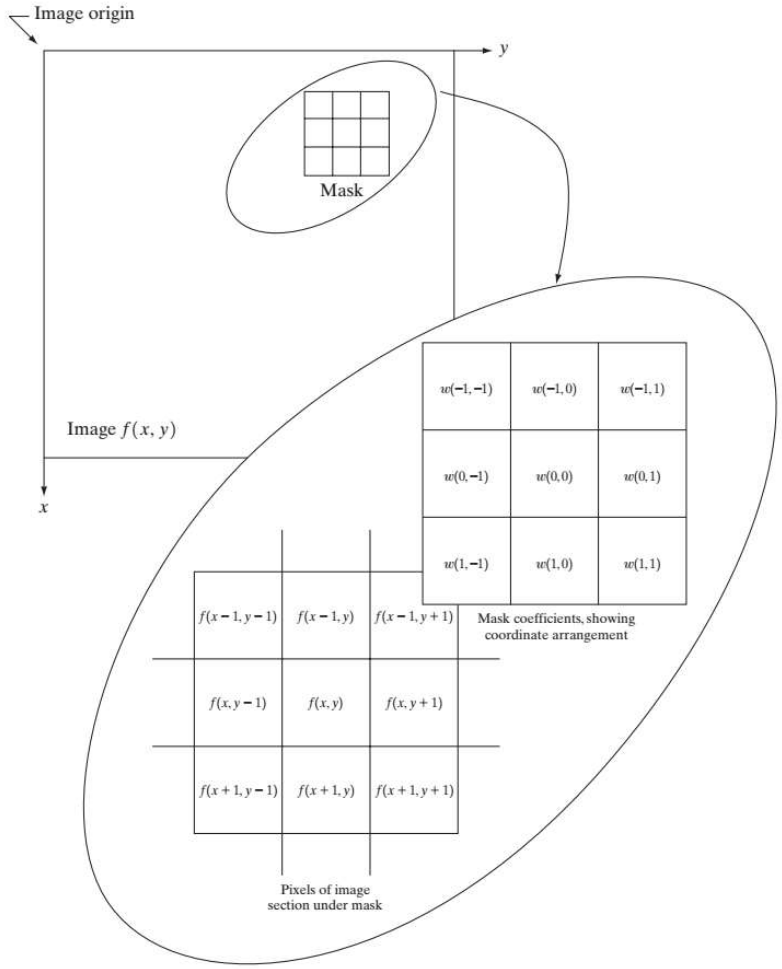
$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u, j-v]$$

- Convolución es asociativa y conmutativa

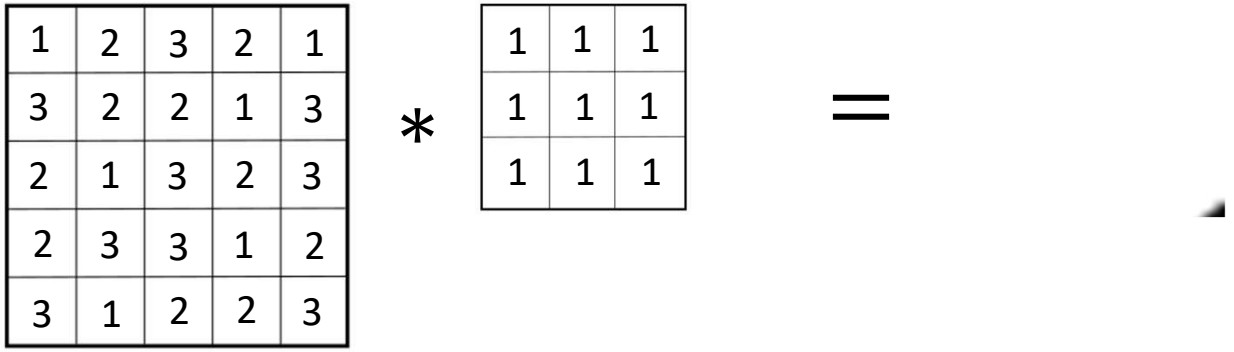
$$G = H * F$$

Convolución

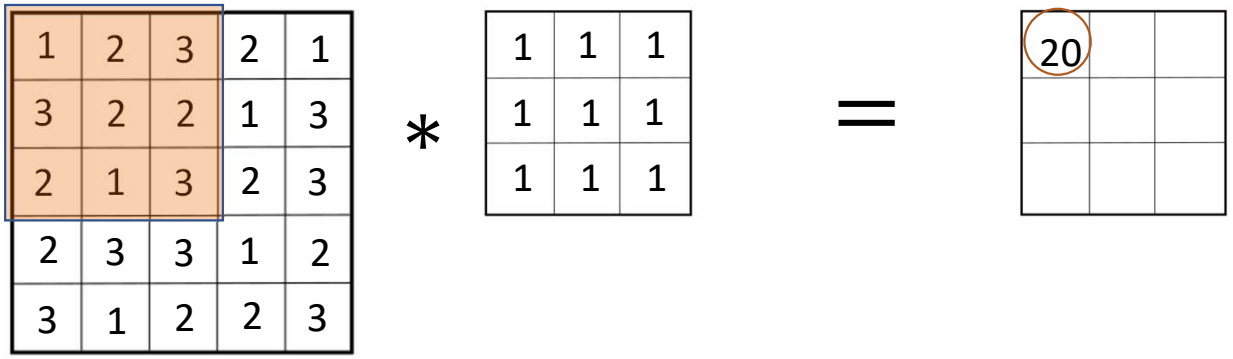
- Es una operación píxel a píxel
- El kernel se desliza en la imagen mientras computa la transformación lineal local.



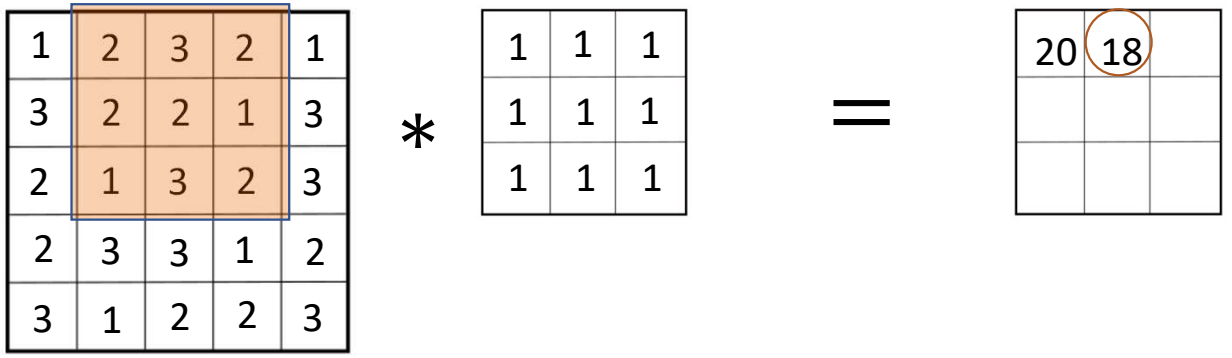
Convolución: Ejemplo



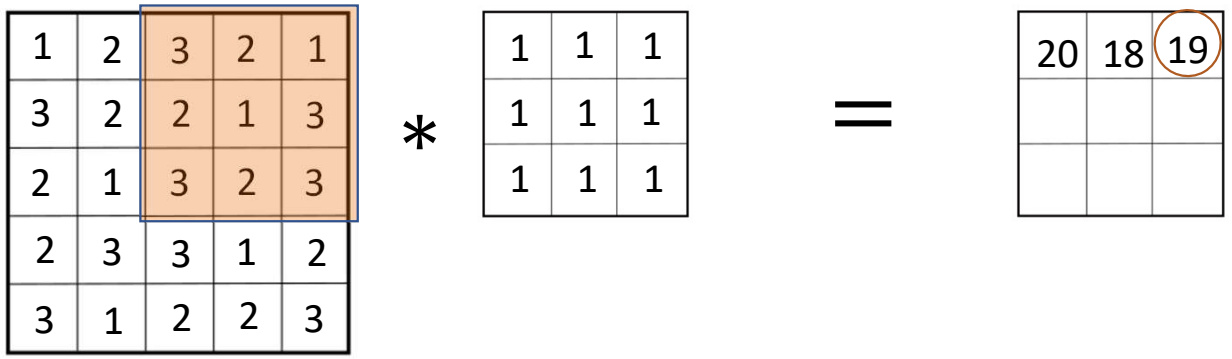
Convolución: Ejemplo



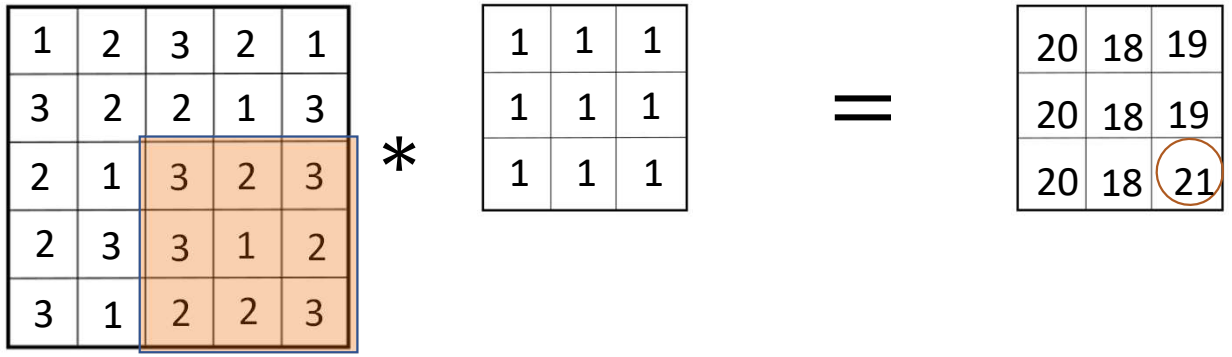
Convolución: Ejemplo



Convolución: Ejemplo

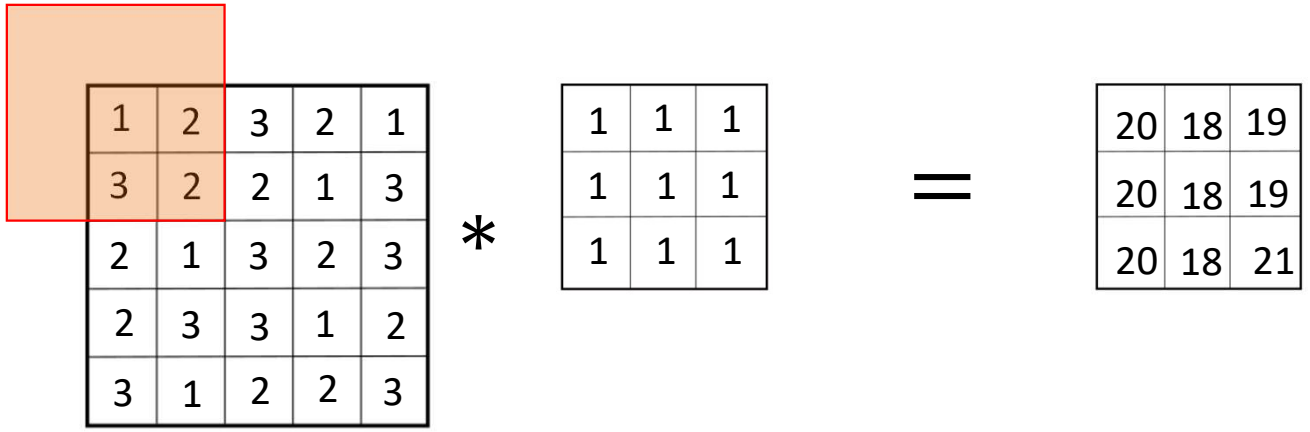


Convolución: Ejemplo



Convolución: Ejemplo

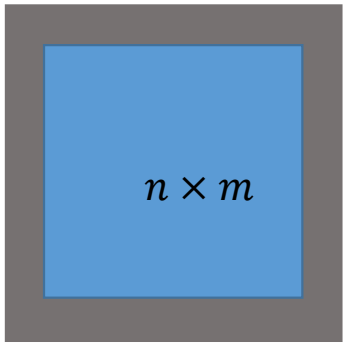
- Has notado que se pierde información?



- Imagen de entrada es de 5x5 y la salida es de 3x3
- No podemos aplicar el kernel en ciertas posiciones (bordes)

Convolución: padding

- Estrategias para el problema de los bordes:
  - Tomar la salida, sin cambios. Podríamos estar perdiendo información valiosa.
  - Padding: rellenar la imagen de entrada con ceros, de tal forma que la salida tenga el mismo tamaño que la entrada



$(n + k - 1) \times (m + k - 1)$

$k \times k$

Efecto de la convolución

- Veamos algunos ejemplos de convolución

$$\text{Kernel} = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Qué operación estamos realizando?

Promedio

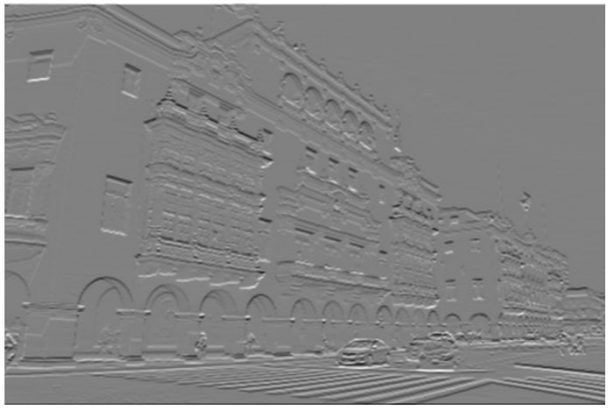
Efecto de la convolución

- Veamos algunos ejemplos de convolución

Kernel = kernel promedio de tamaño 21

Efecto de la convolución

- Veamos algunos ejemplos de convolución



Kernel =

-1	0	1
-2	0	2
-1	0	1

Sobel, vertical edges

Deep Learning

Capas Convolucionales

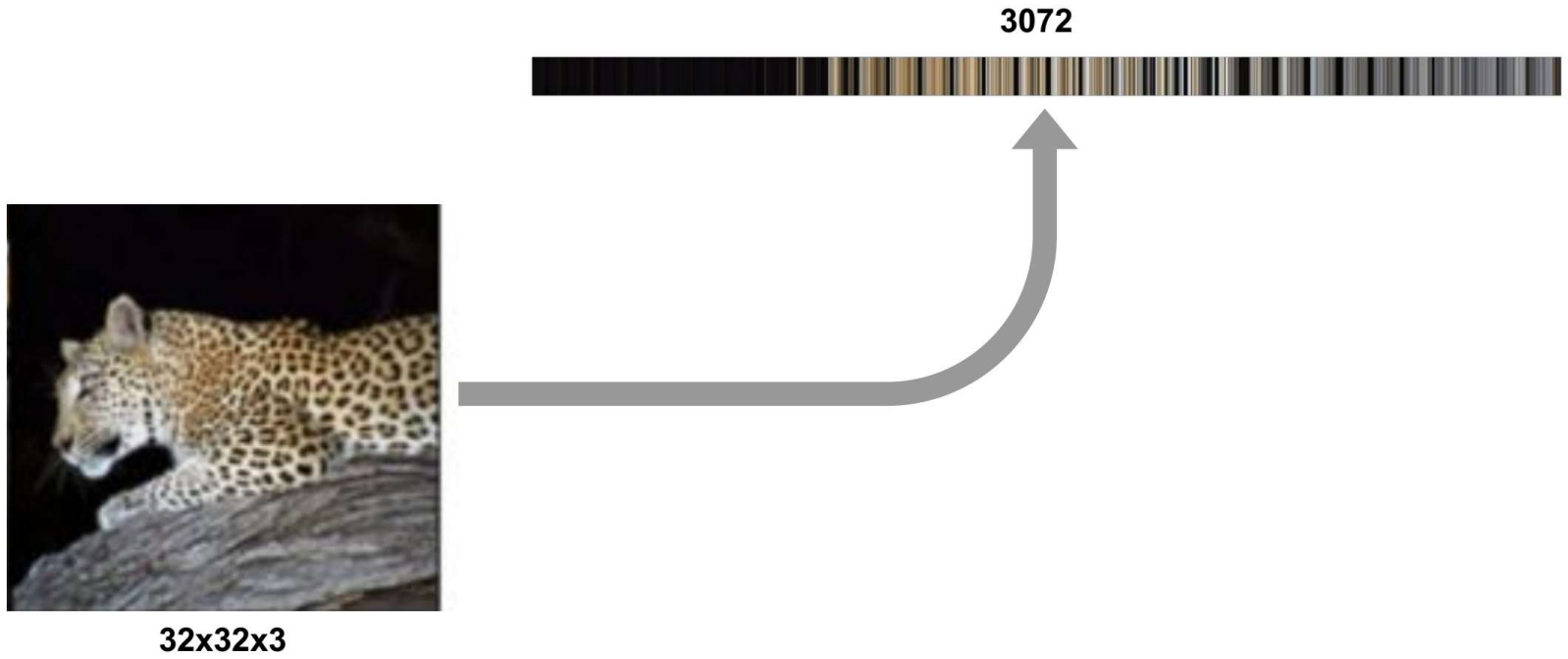
Imágenes son datos

- Si queremos implementar clasificación de imágenes con un MLP

32x32x3

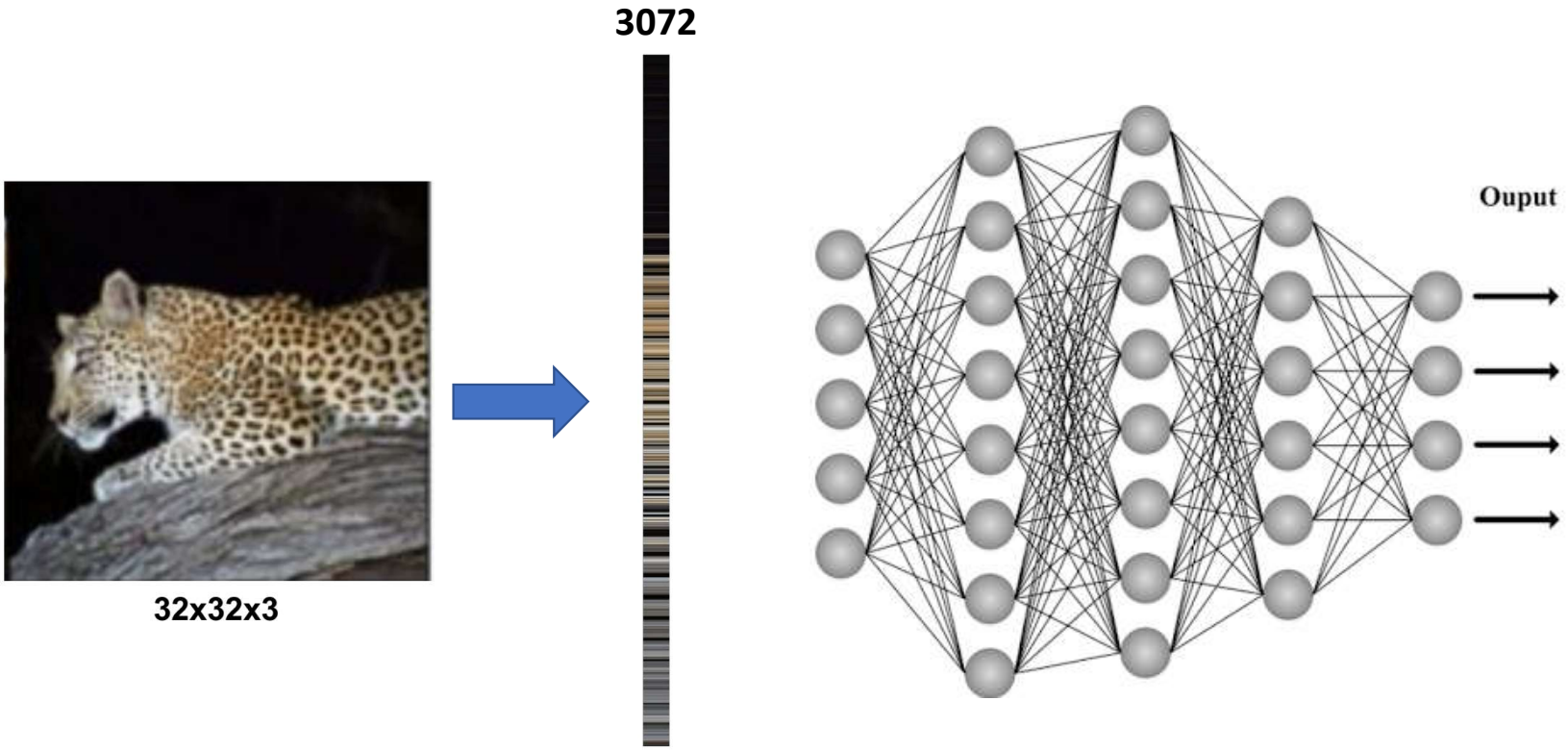
Imágenes son datos

- Si queremos implementar clasificación de imágenes con un MLP



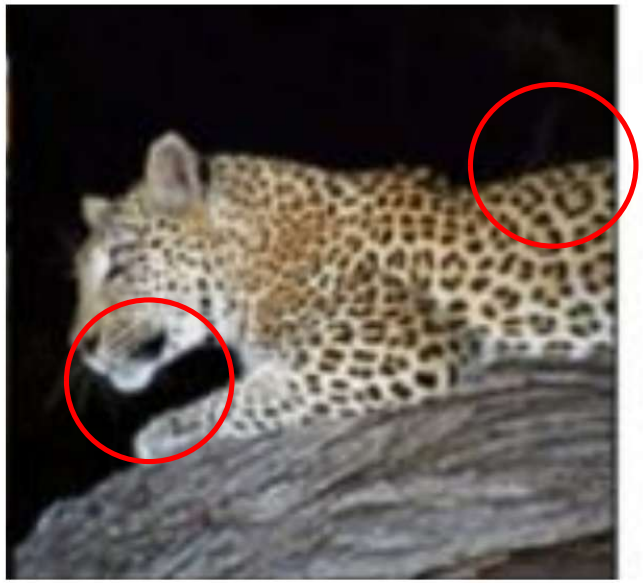
Imágenes son datos

- Si queremos implementar clasificación de imágenes con un MLP



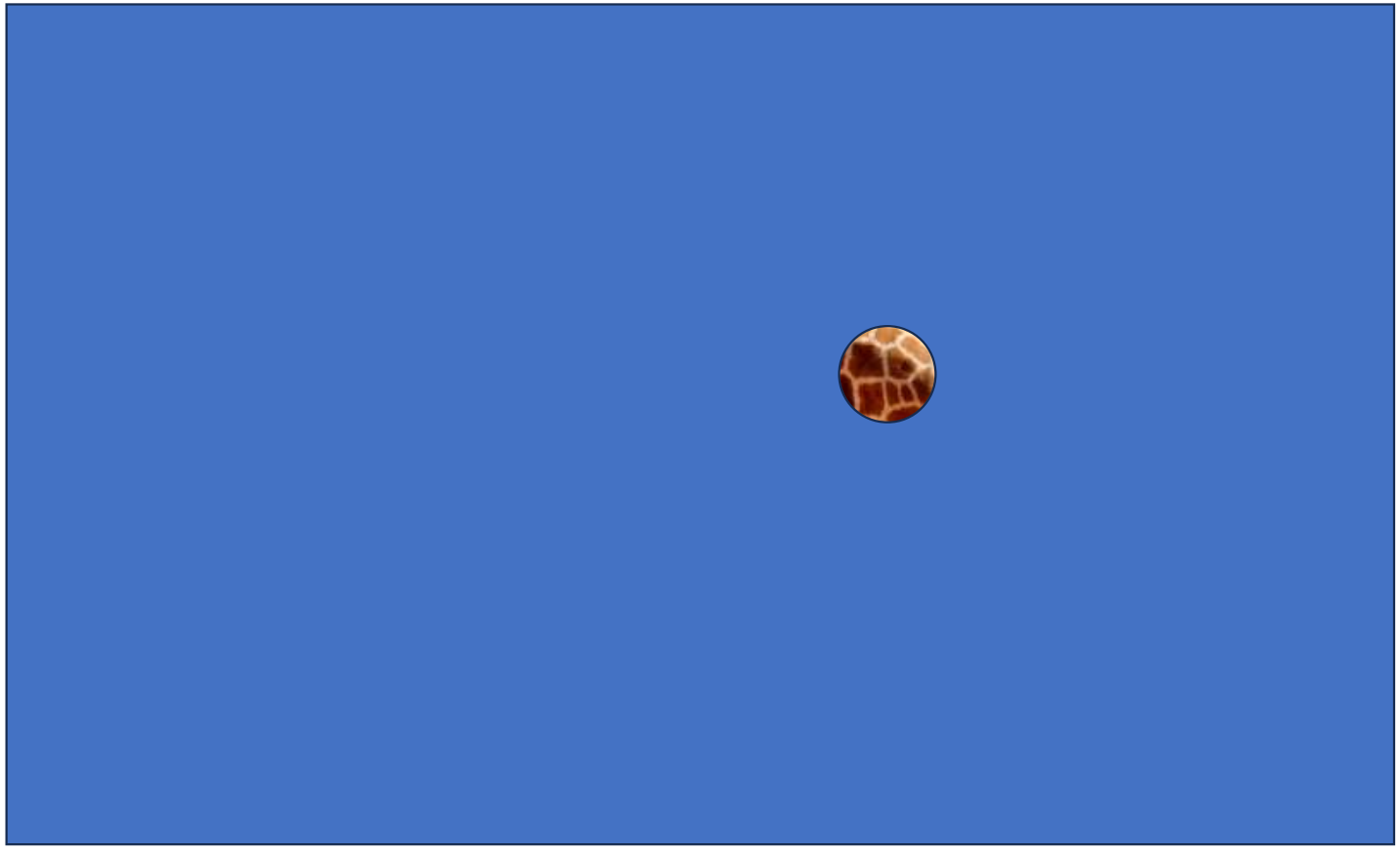
Imágenes son datos

- Si queremos implementar clasificación de imágenes con un MLP
- La cantidad de parámetros es enorme
- Un MLP intenta encontrar todas las posibles relaciones entre los píxeles de entrada. Eso está bien?

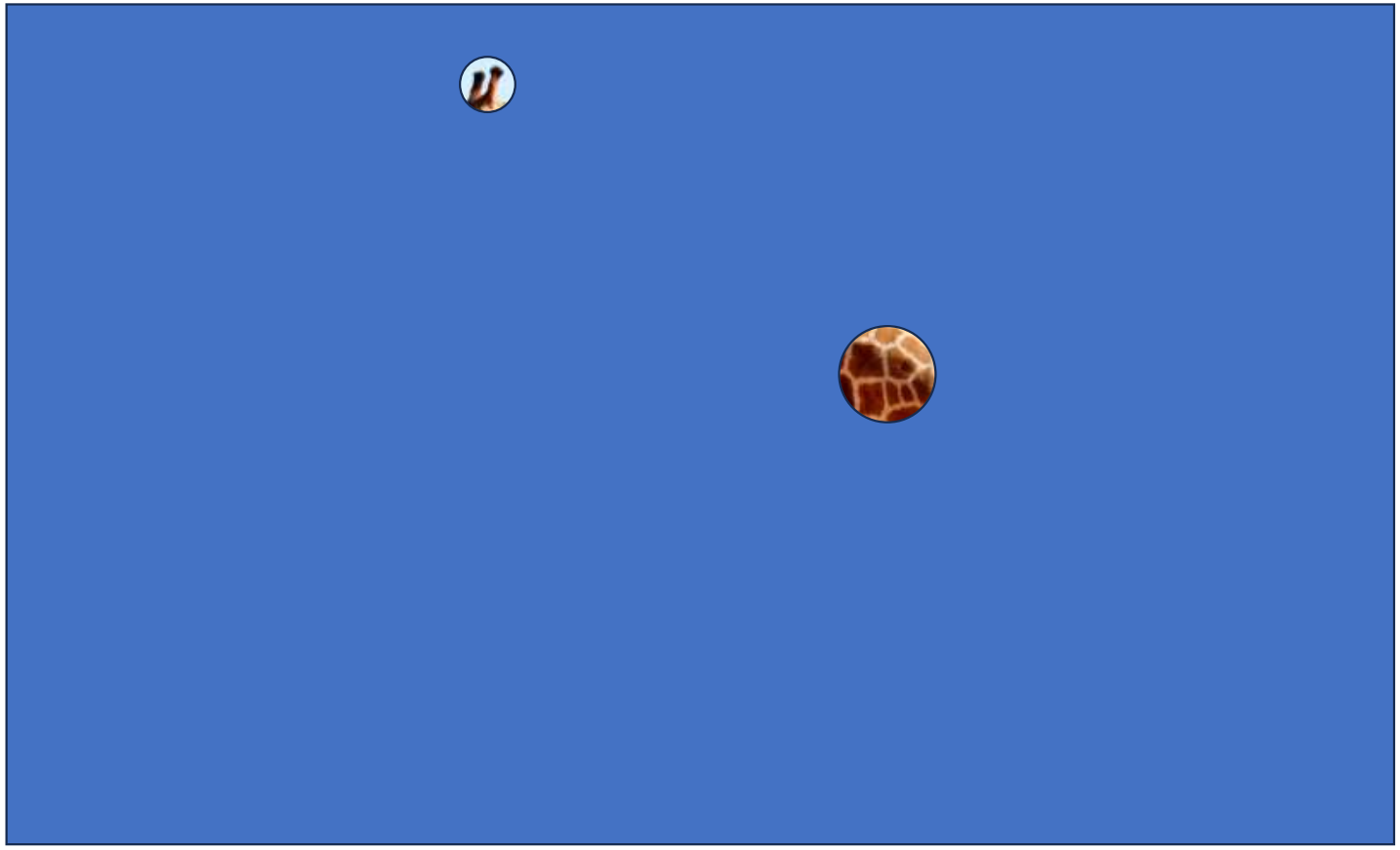


Imágenes son datos

Imágenes son datos



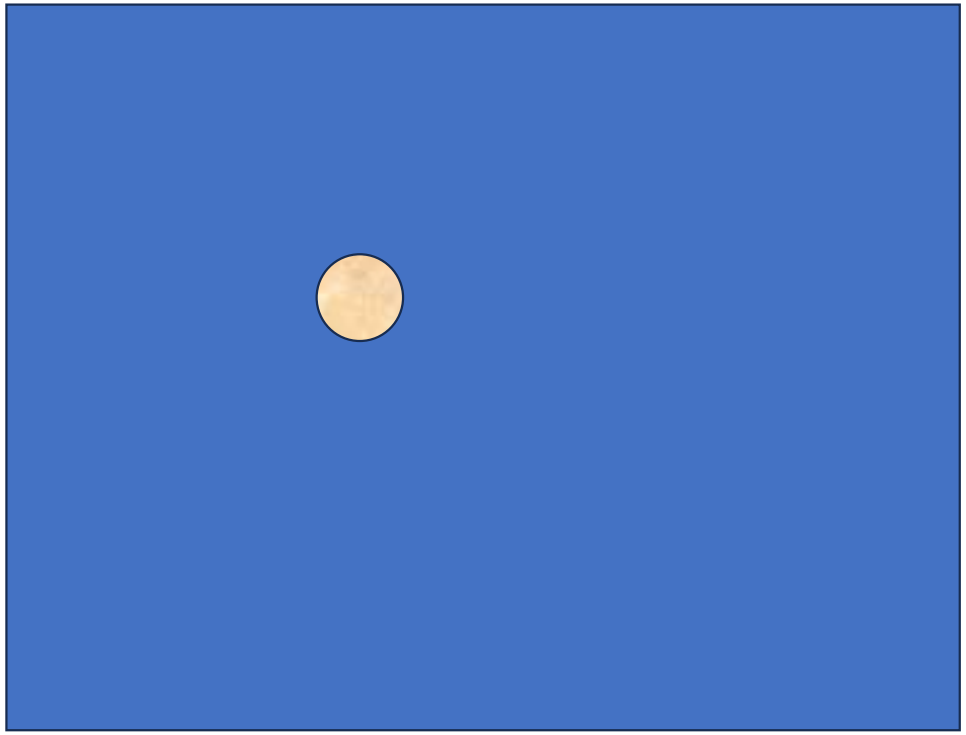
Imágenes son datos



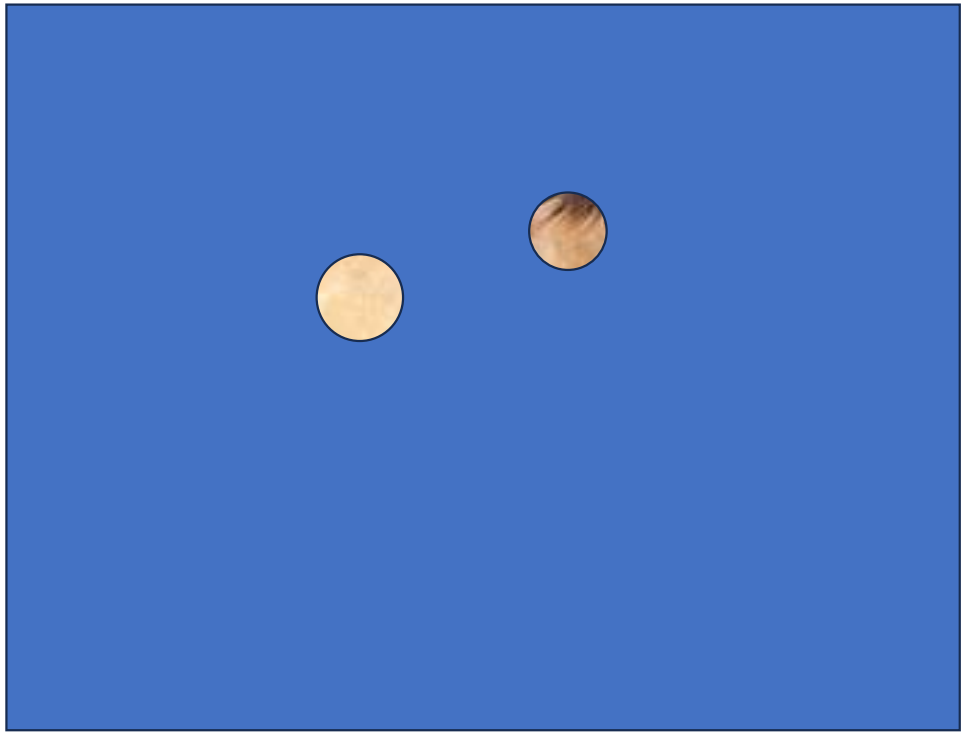
Imágenes son datos

Imágenes son datos

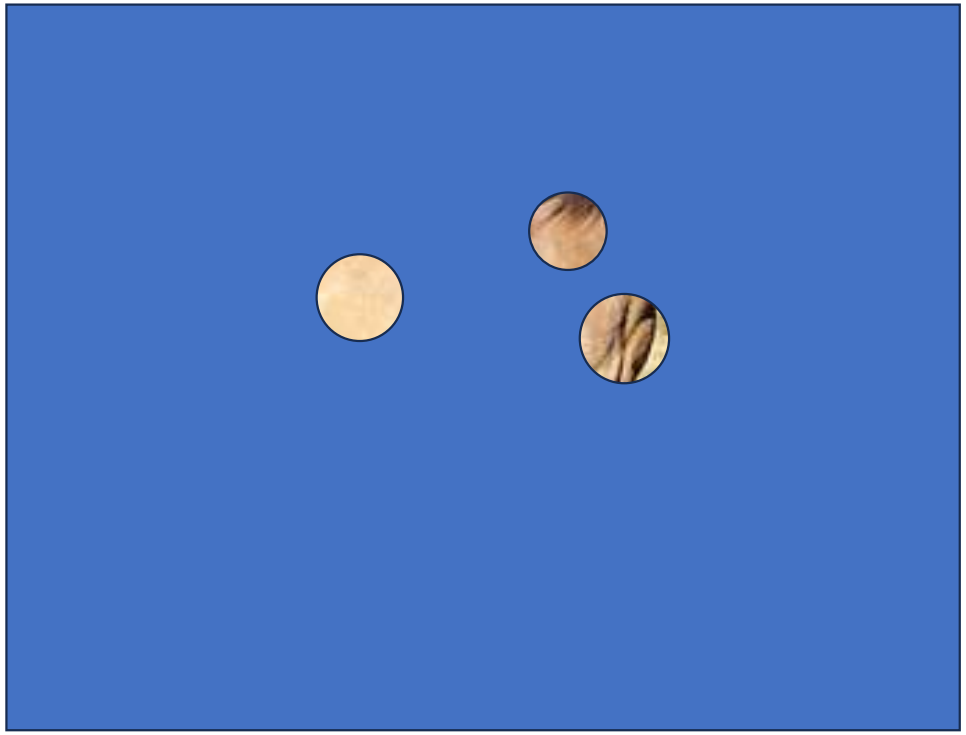
Imágenes son datos



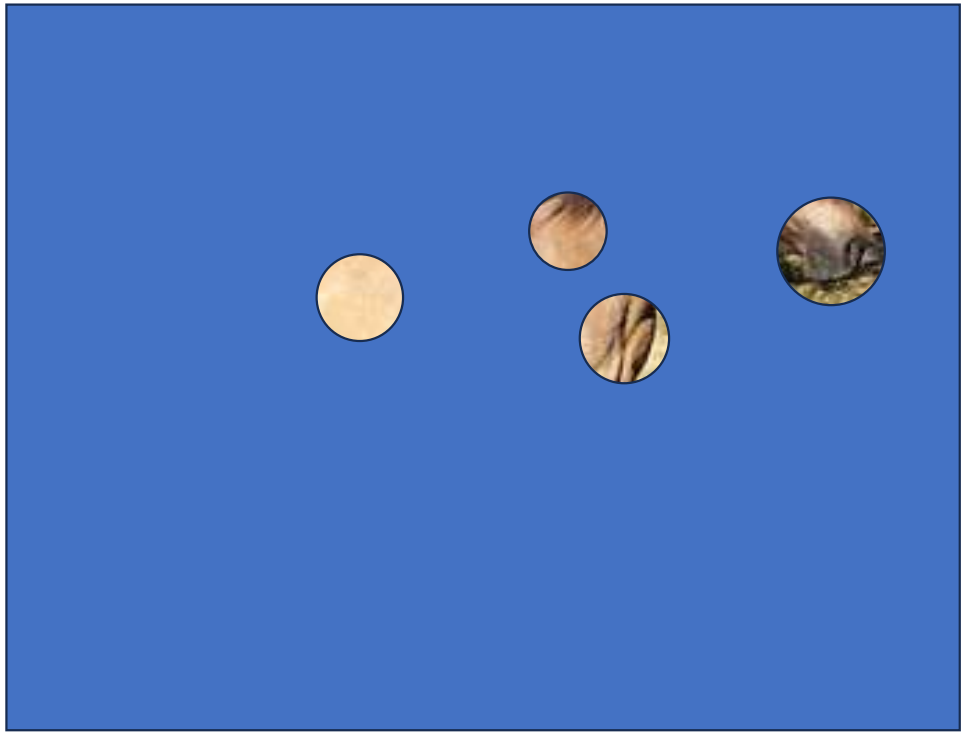
Imágenes son datos



Imágenes son datos



Imágenes son datos



Imágenes son datos

Imágenes son datos

Mucha información visual estructural es local.

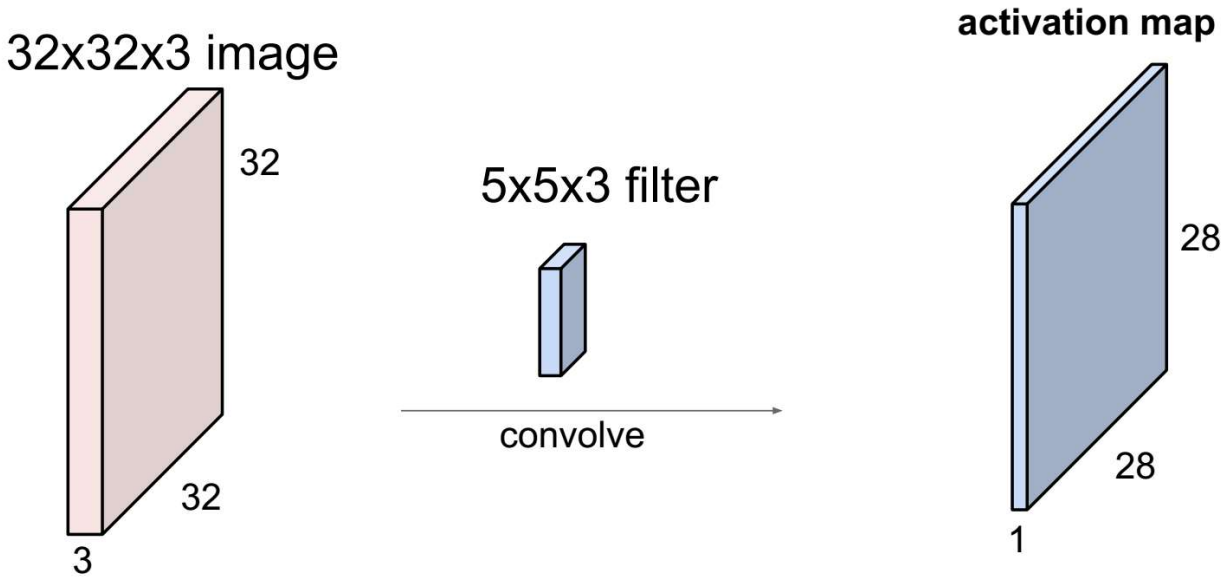
Píxeles en regiones lejanas podrían no relacionarse a través de contenido de interés

Ya conocemos una operación que computa información local

Convolución!

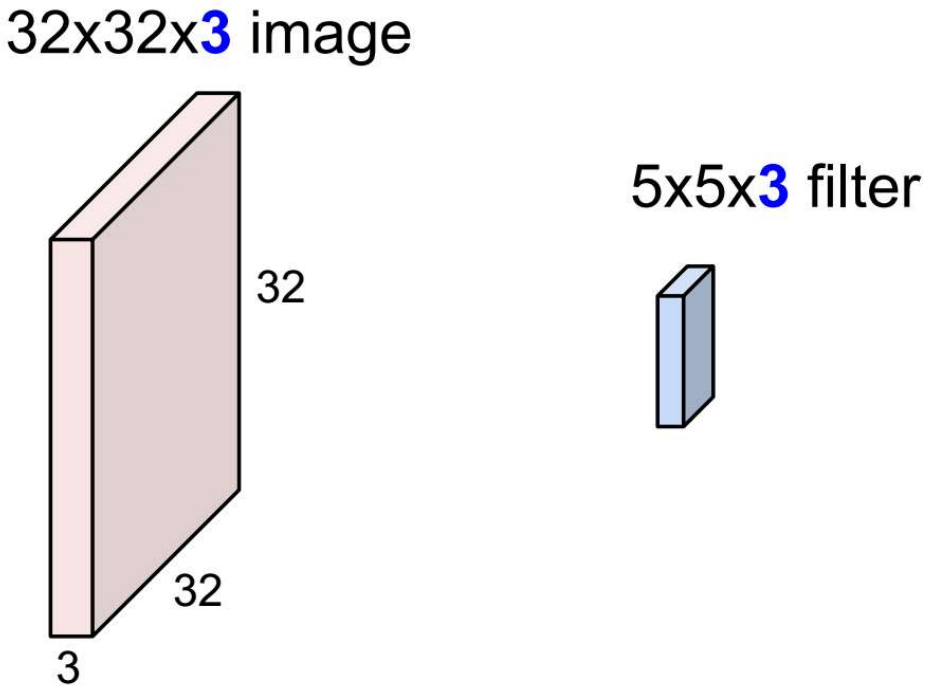
Capas convolucionales

- Una capa convolucional está compuesta de un conjunto de kernels
- Cada kernel se convoluciona con la entrada y genera un mapa de activación (también conocido como mapa característico)



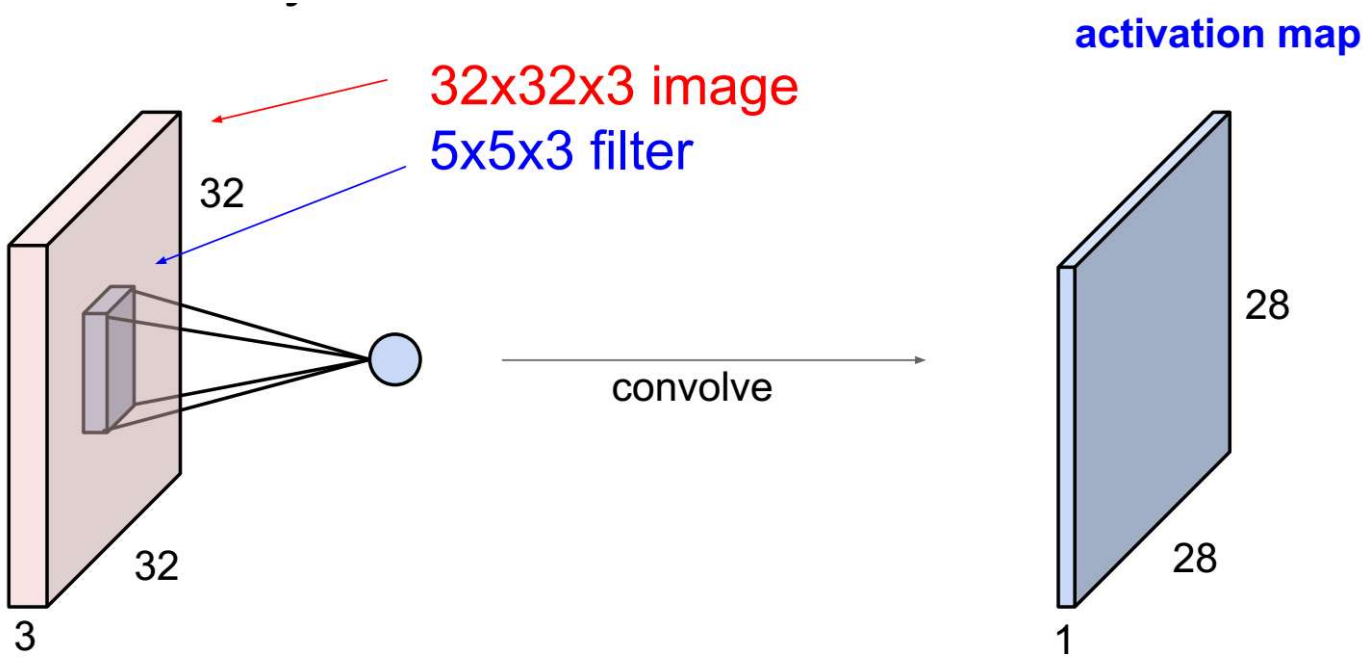
Capas convolucionales

- La profundidad del input es siempre la profundidad de los kernels



Capas convolucionales

- Podemos tener varios kernels en la misma capa
- Cada uno genera un mapa de activación



Capas convolucionales

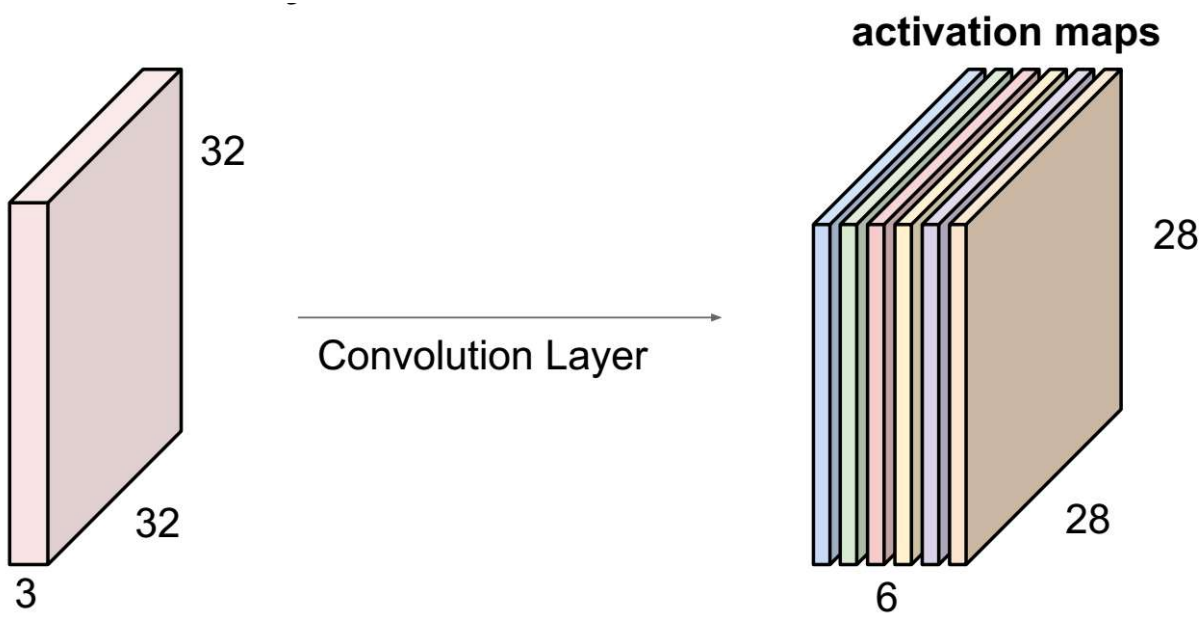
- Podemos tener varios kernels en la misma capa
- Cada uno genera un mapa de activación

Para el segundo kernel

Capas convolucionales

- Podemos tener varios kernels en la misma capa
- Cada uno genera un mapa de activación

Si usamos seis kernels, la salida tendrá
Profundidad seis



Capas convolucionales

- Parámetros
  - En una capa lineal, los parámetros son pesos y biases, y la operación de la capa es

$$z_L = X_{L-1} \cdot W_L + b_L$$

  - La operación lineal es el producto punto entre el input y los pesos
  - En la capa convolucional tenemos

$$z_L = X_{L-1} * W_L + b_L$$

en donde la operación lineal es la convolución, y la salida es el mapa de activación. W_L representa el kernel y el bias es un solo número.

Capas convolucionales

- Cuántos parámetros hay en una capa convolucional de m kernels de dimensión $k \times k \times d$?

$$m \times k \times k \times d + m$$

Parámetros de kernel

Biases

Deep Learning

Hyper-parámetros y Pooling

Parámetros

- Para construir una capa convolucional necesitamos
  - K: número de kernels en la capa
  - F: extensión espacial del kernel
  - S: el stride (número de píxeles para mover la ventana de convolución)
  - P: el número f de píxels para padding

Si el volumen de entrada tiene tamaño W, el tamaño del mapa de activación es

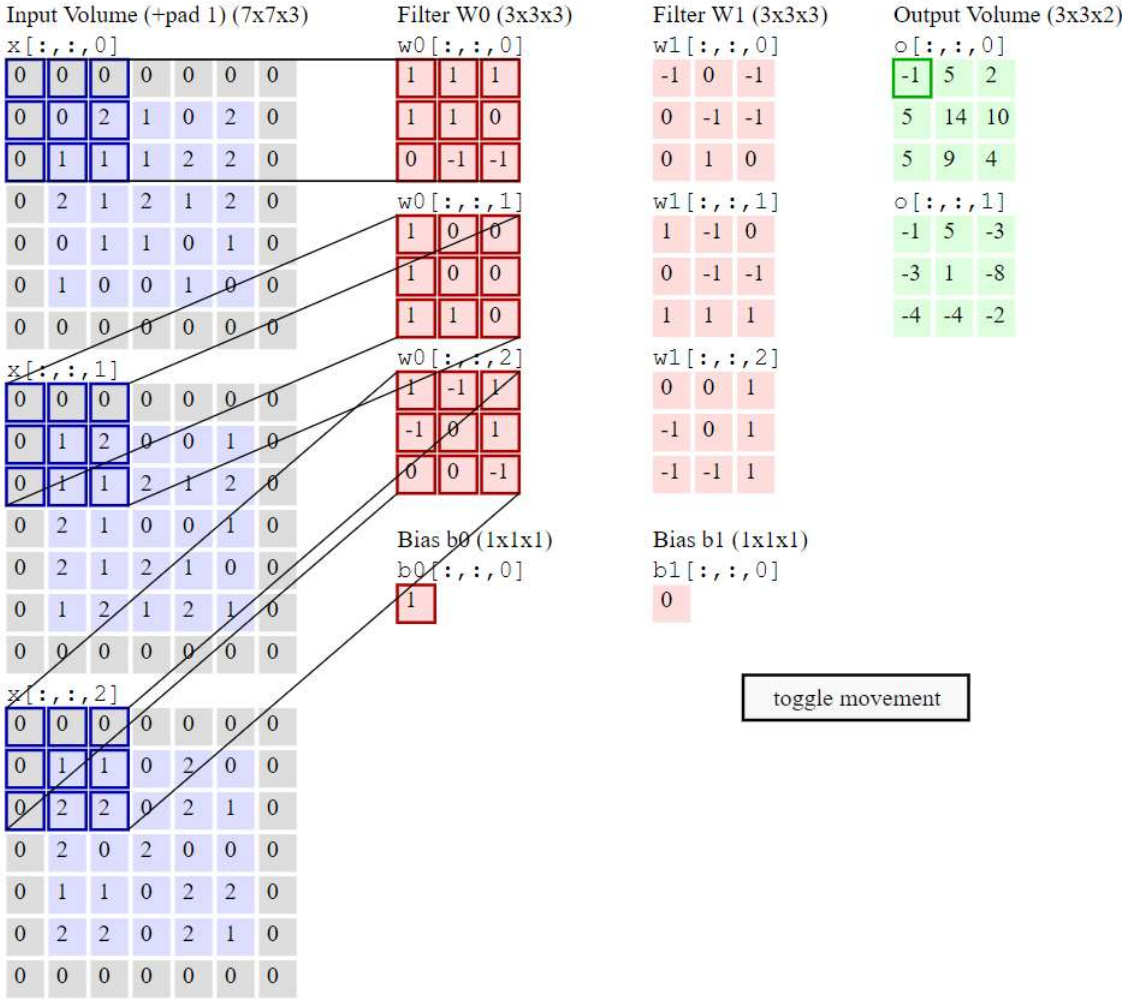
$$\frac{W - F + 2P}{S} + 1$$

Ejemplo

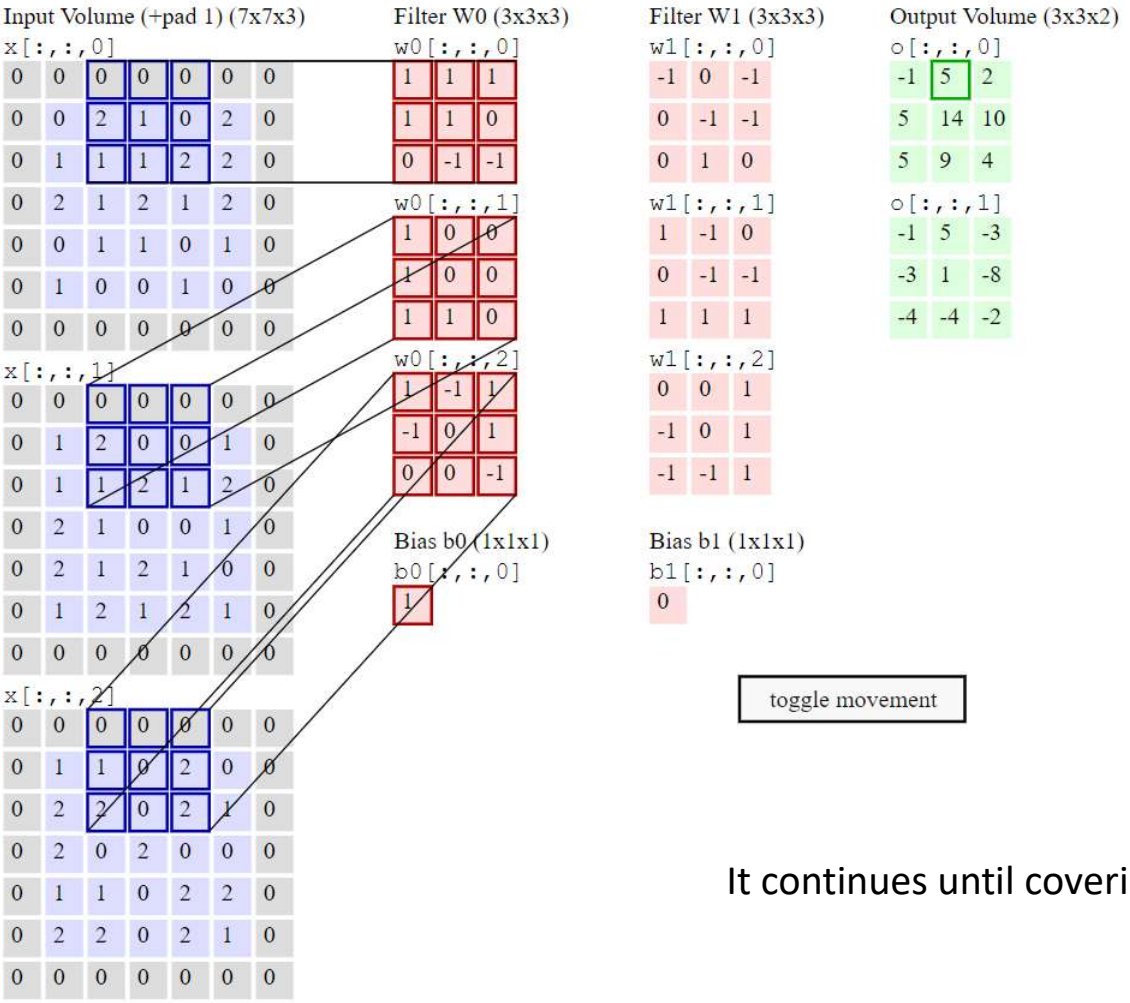
Red channel

Green channel

Blue channel

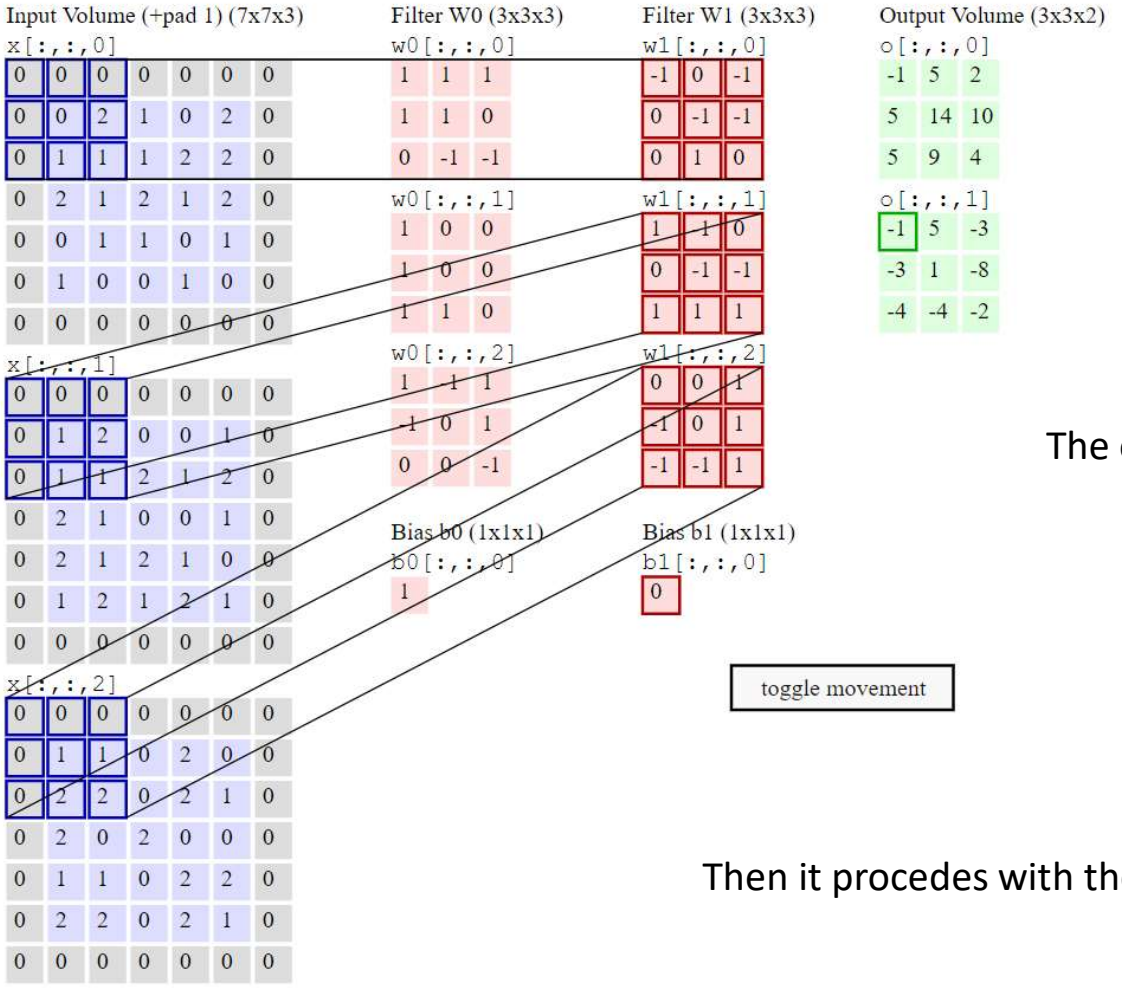


Ejemplo



It continues until covering the complete image

Ejemplo

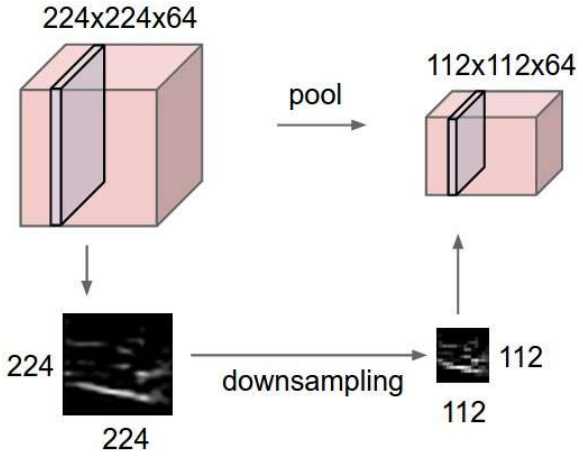


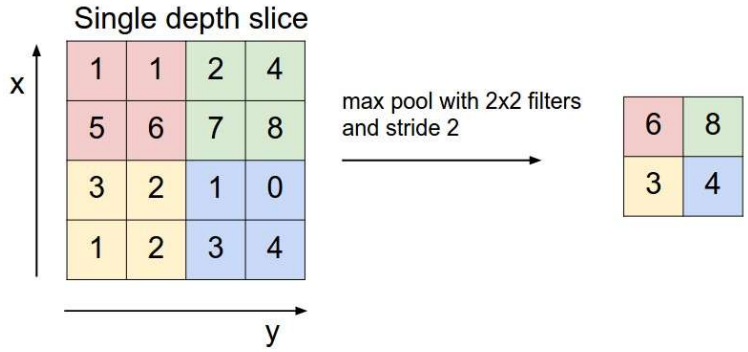
The output volumen is 3x3x2

Then it procedes with the second kernel

Capa de Pooling

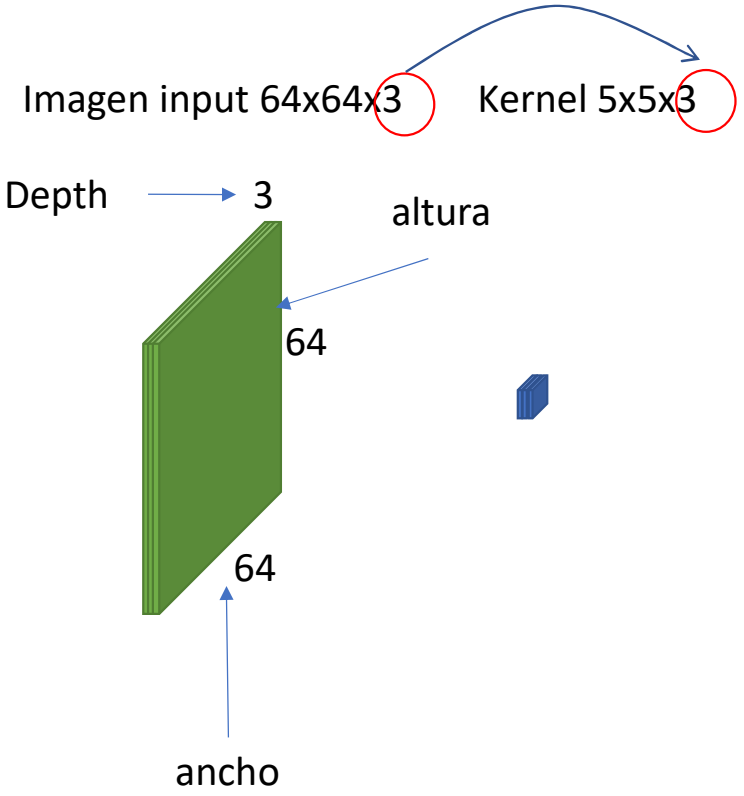
- Reduce el tamaño espacial de la representación
  - Reduce la cantidad de parámetros
  - Reduce el cómputo
  - Permite encontrar representaciones multi-escala en las imágenes
- Max pooling, tamaño 2, stride 2 es la elección típica





Deep Learning

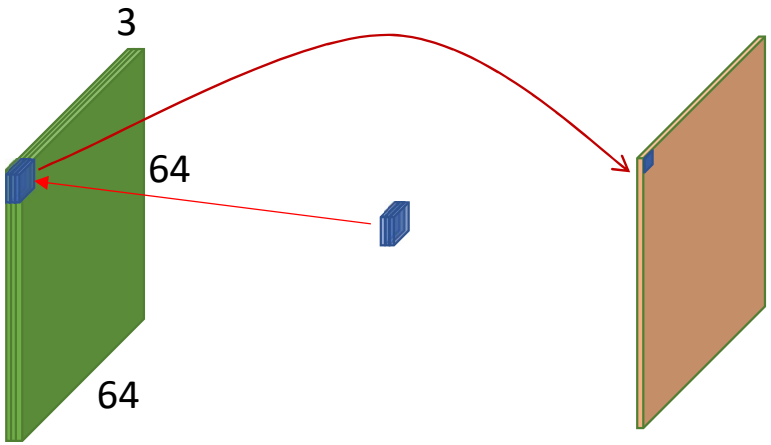
Ejemplo Convolución



Input depth = kernel depth

Imagen input 64x64x3 Kernel 5x5x3

Input depth = kernel depth

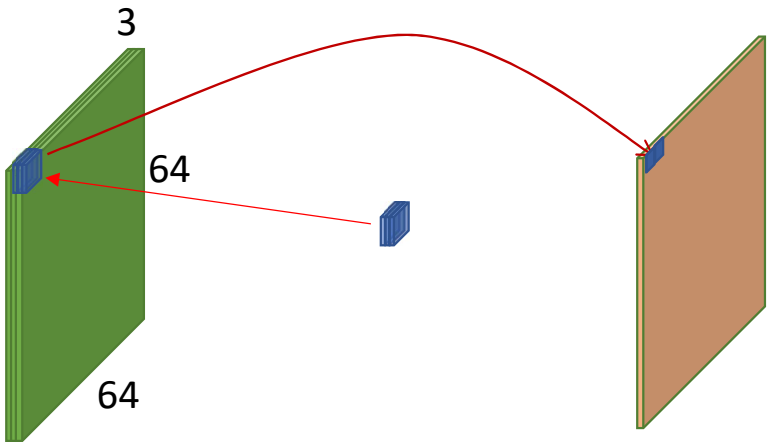


La convolución del kernel con el parche local
en la imagen genera un solo valor

Imagen input
64x64x3

Kernel 5x5x3

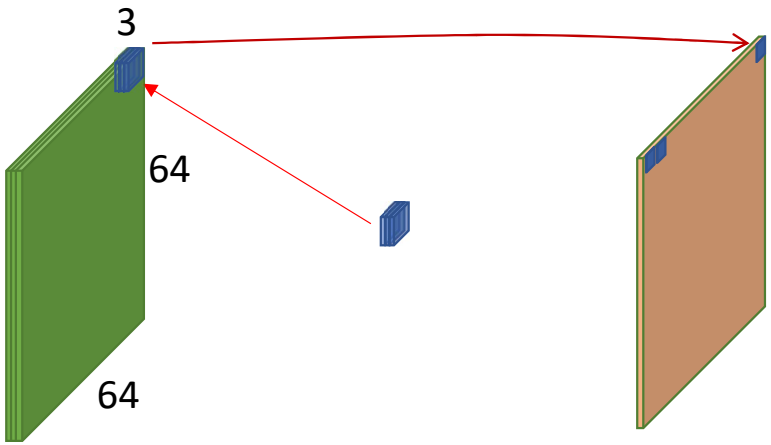
Input depth = kernel depth



La convolución del kernel con el parche local en la imagen genera un solo valor

Input image 64x64x3 Kernel 5x5x3

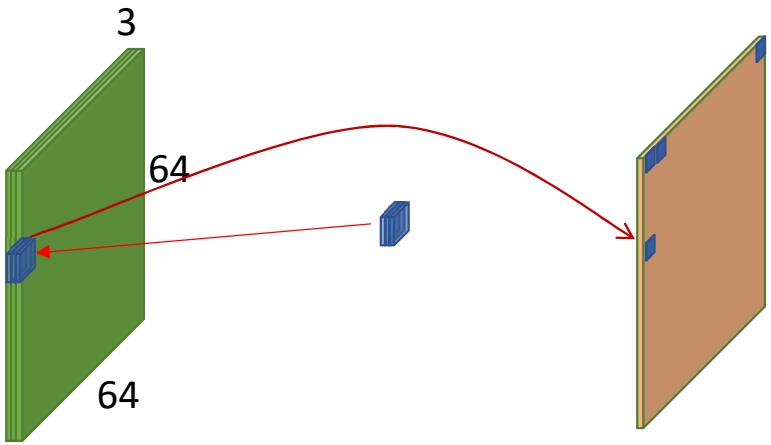
Input depth = kernel depth



La convolución del kernel con el parche local en la imagen genera un solo valor

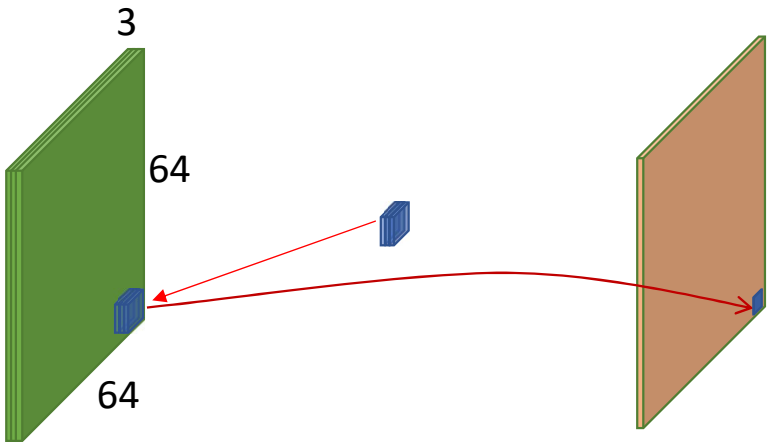
Input image 64x64x3 Kernel 5x5x3

Input depth = kernel depth



La convolución del kernel con el parche local en la imagen genera un solo valor

Input image 64x64x3 Kernel 5x5x3



La convolución del kernel con el parche local en la imagen genera un solo valor

Input depth = kernel depth

Resultado final: mapa de activación

Mapa de activación: 60x60x1 (sin padding)

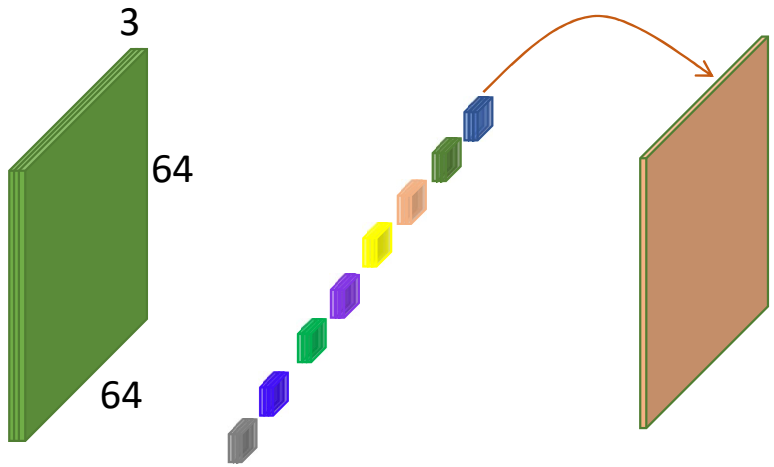
Mapa de activación: 64x64x1 (con padding=2)

Un kernel produce un mapa de profundidad 1

Input image 64x64x3

kernel 5x5x3

Input depth = kernel depth

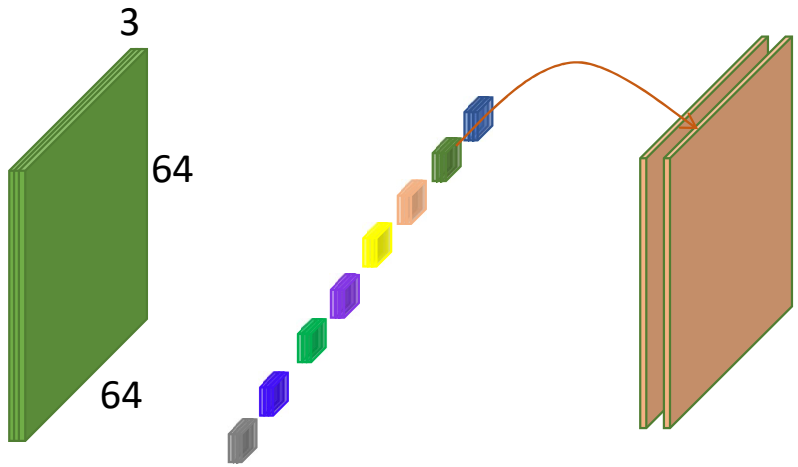


8 kernels of 5x5x3

Input image 64x64x3

kernel 5x5x3

Input depth = kernel depth

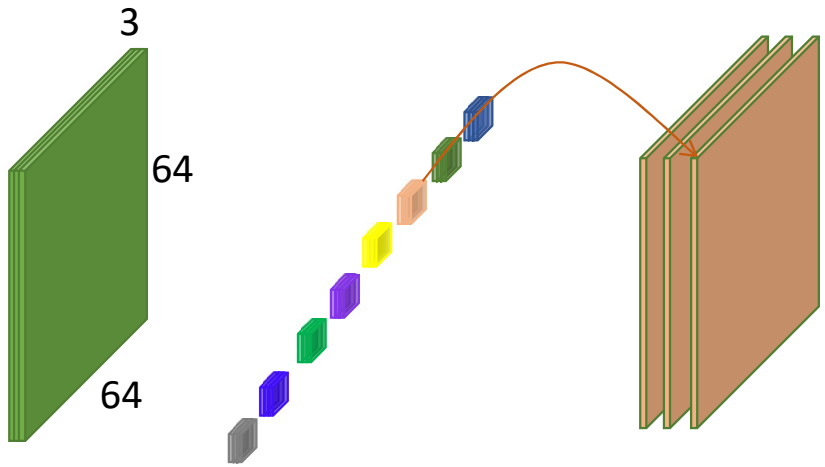


8 kernels of 5x5x3

Input image 64x64x3

kernel 5x5x3

Input depth = kernel depth

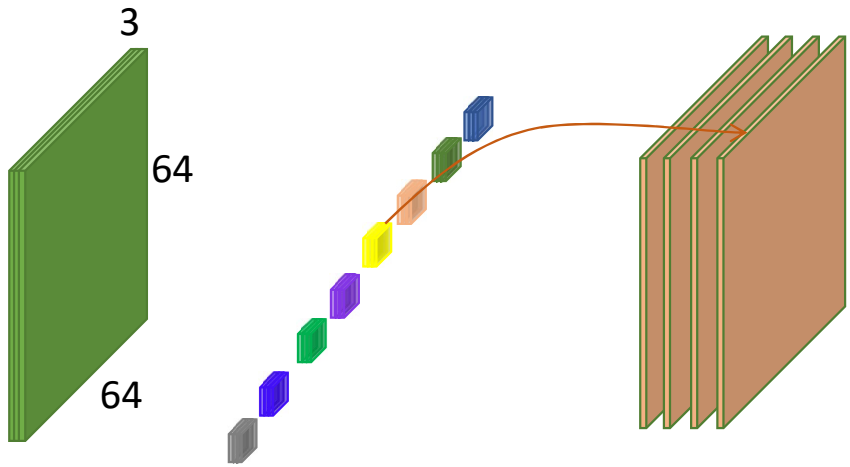


8 kernels of 5x5x3

Input image 64x64x3

kernel 5x5x3

Input depth = kernel depth

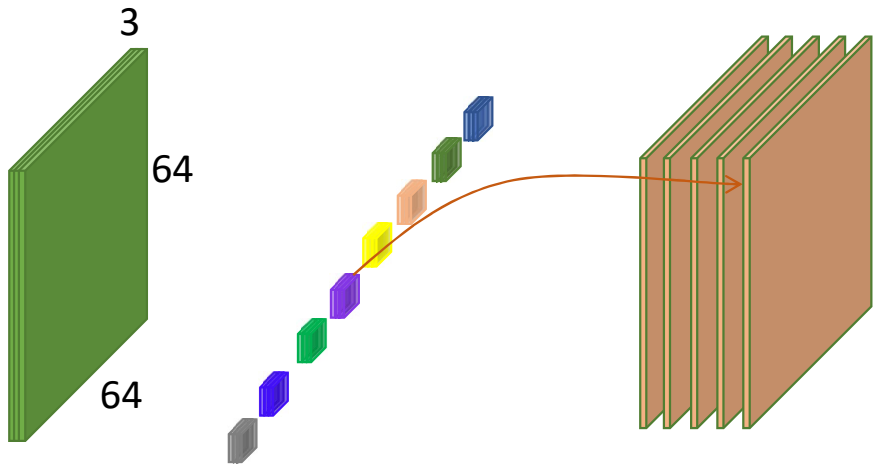


8 kernels of 5x5x3

Input image 64x64x3

kernel 5x5x3

Input depth = kernel depth

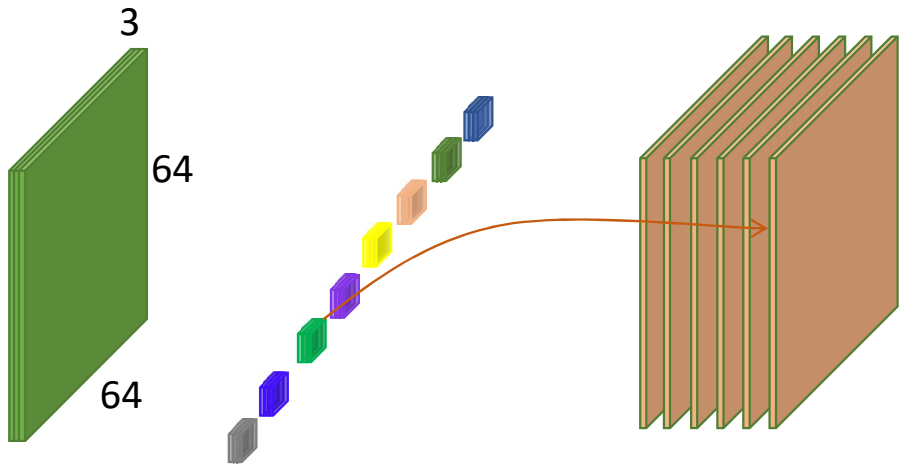


8 kernels of 5x5x3

Input image 64x64x3

kernel 5x5x3

Input depth = kernel depth

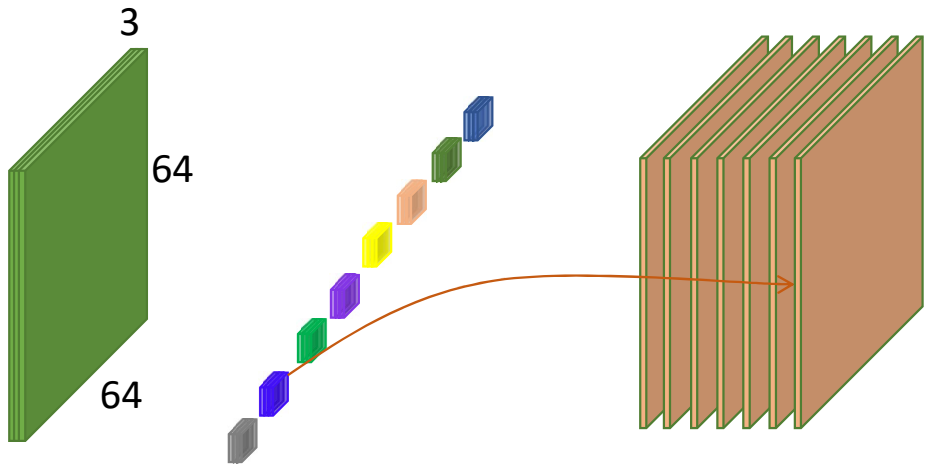


8 kernels of 5x5x3

Input image 64x64x3

kernel 5x5x3

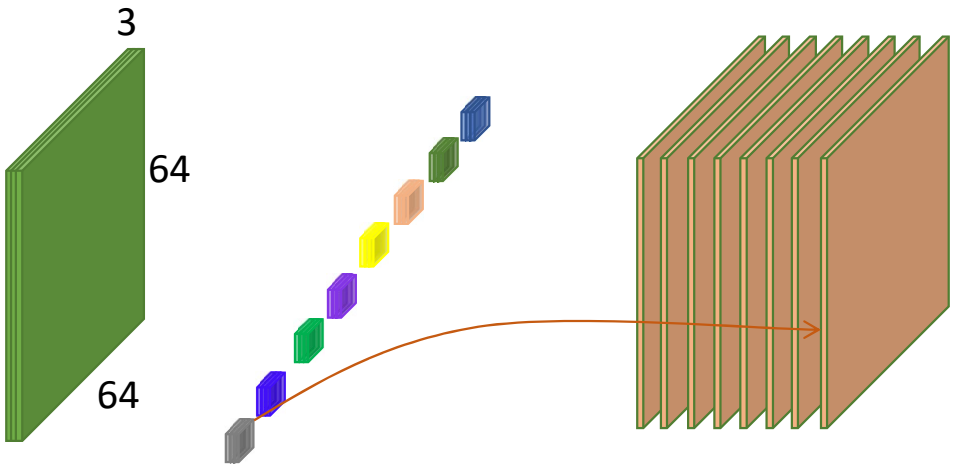
Input depth = kernel depth



8 kernels of 5x5x3

Input image 64x64x3 kernel 5x5x3

Input depth = kernel depth



Salida es *volumen* de 64x64x8

8 kernels of 5x5x3

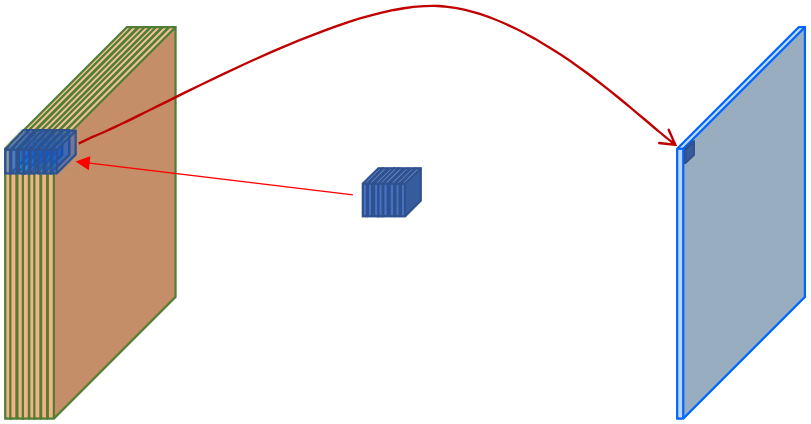
Primera capa convolucional

- Entrada: volumen de 64x64x3
- Kernels: 8 de 5x5x3 con padding=1
- Salida: volumen de 64x64x8

Segunda capa convolucional

Input: volumen 64x64x8

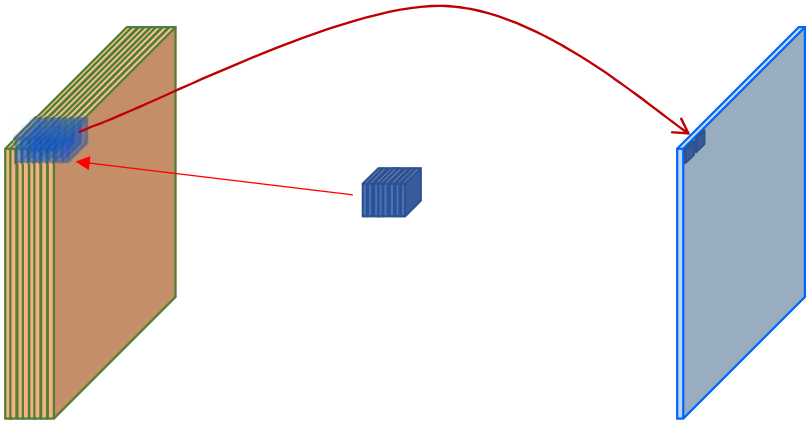
Kernel: volumen 5x5x8



Segunda capa convolucional

Input: volume 64x64x8

Kernel: volume 5x5x8



Deep Learning

Clasificación de imágenes

Introducción

- Problema de asignar una etiqueta a una imagen

Classification

CAT

Introducción

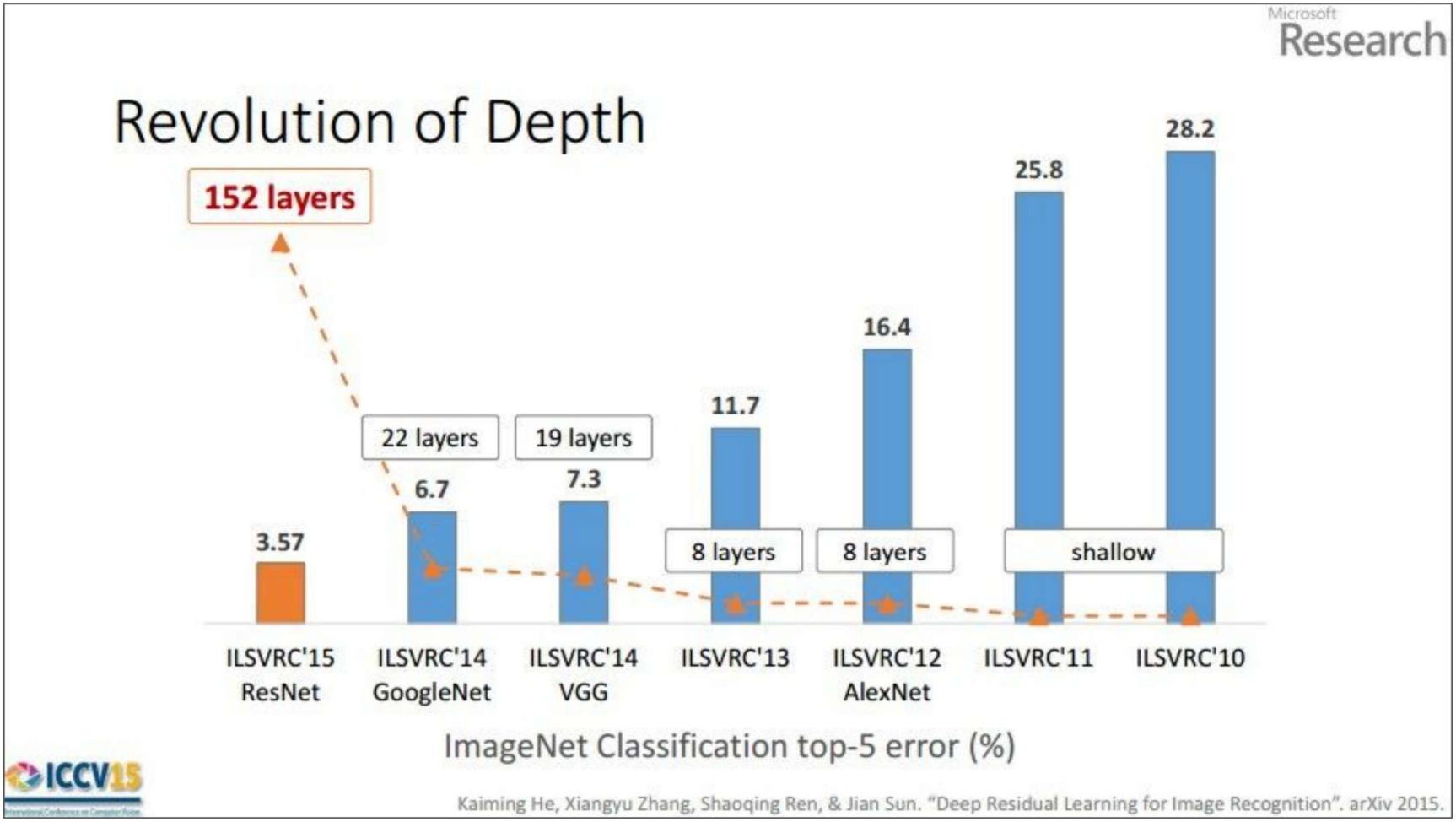
- Para entrenar modelos profundos, necesitamos datasets grandes: ImageNet
- +14M imágenes, +21,000 clases
- No todo el dataset es útil. Se usa el Imagenet Large Scale Visual Recognition Challenge dataset (ILSVR)

Introducción

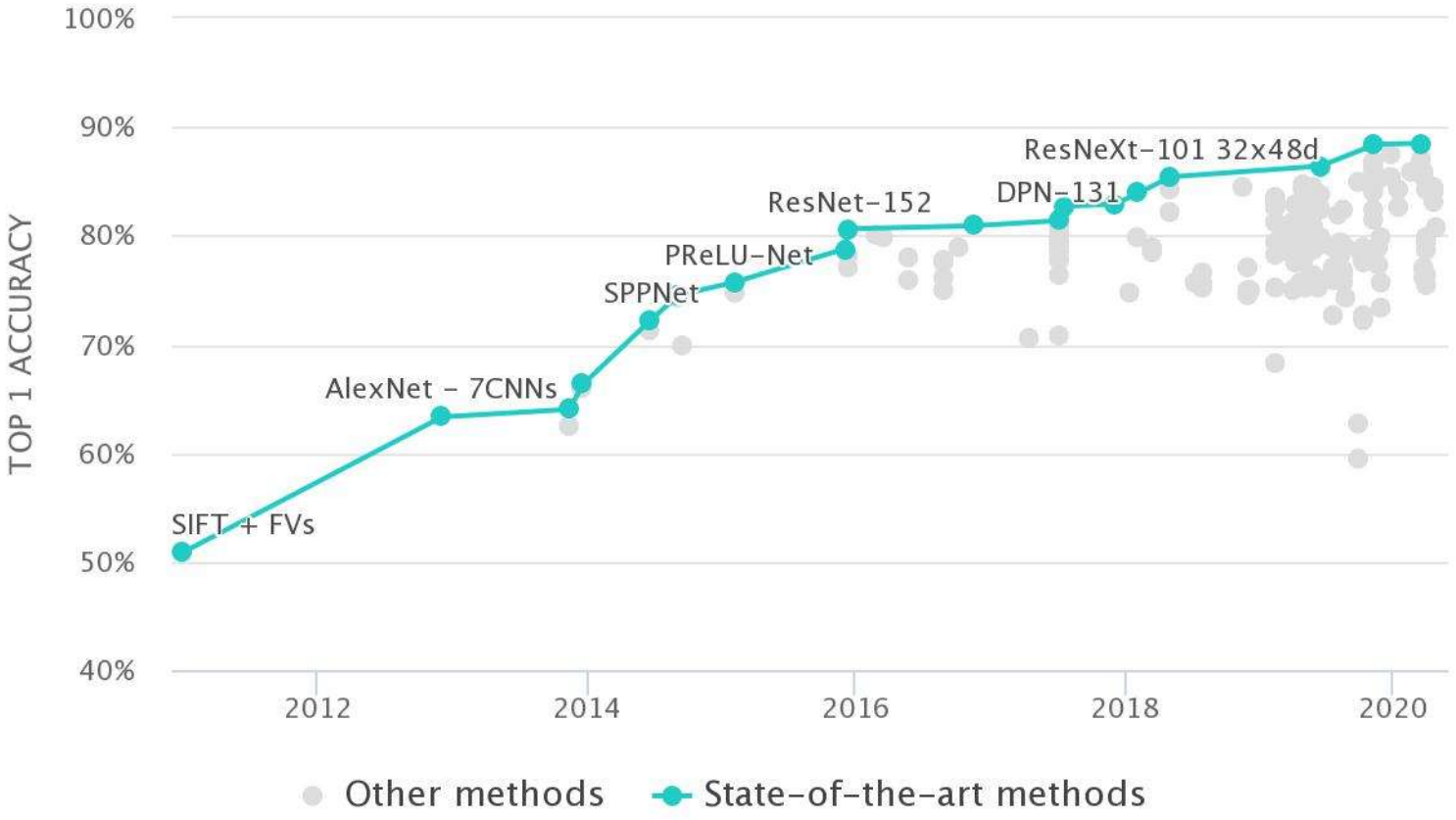
- ILSVR dataset
  - Usado para competiciones desde 2010
  - Num. imágenes: 1.2 Millones (200K test)
  - Num. clases: 1000
- Métricas de evlauación
  - Top-1 Accuracy: Ratio de imágenes de test con clase correcta con mayor probabilidad.
  - Top-5 Accuracy: Ratio de imágenes de test con clase correcta entre las 5 predicciones con mayor probabilidad.

http://www.image-net.org/challenges/LSVRC/

La competición



Estado reciente

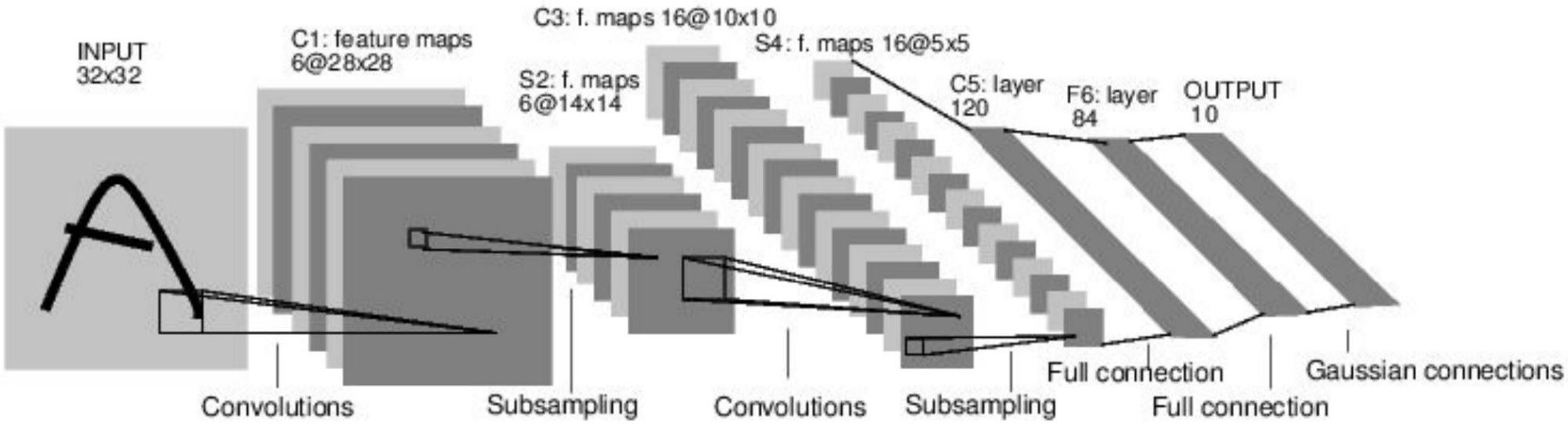


https://paperswithcode.com/sota/image-classification-on-imagenet

Deep Learning

Arquitecturas comunes

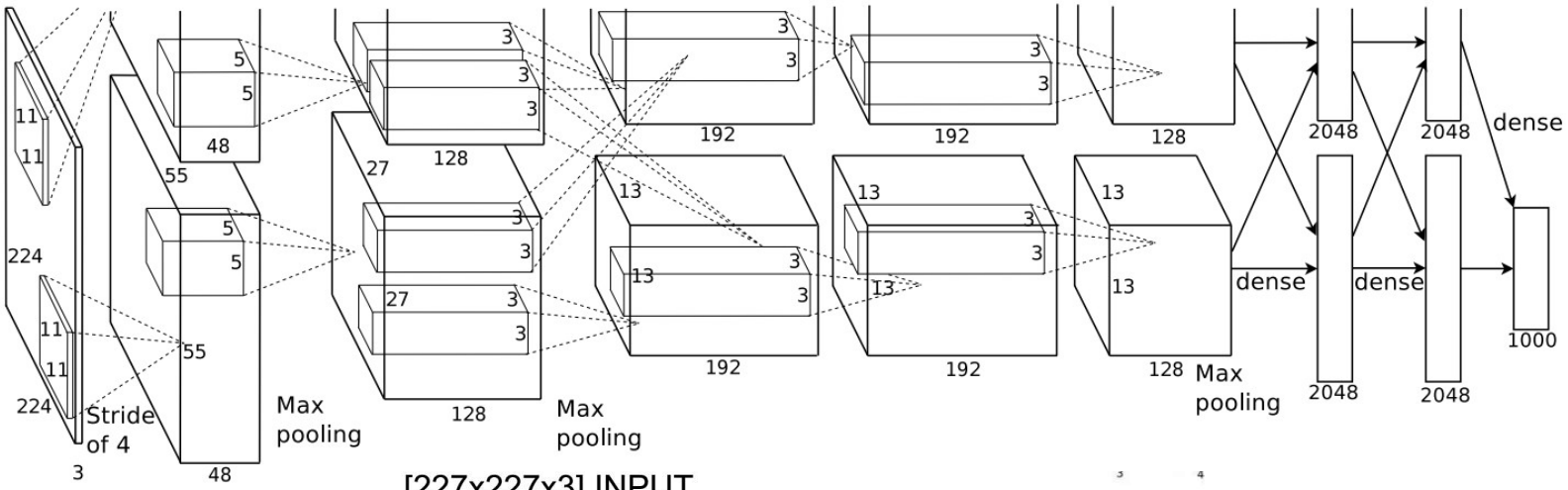
LeNet-5



Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-CONV-FC]

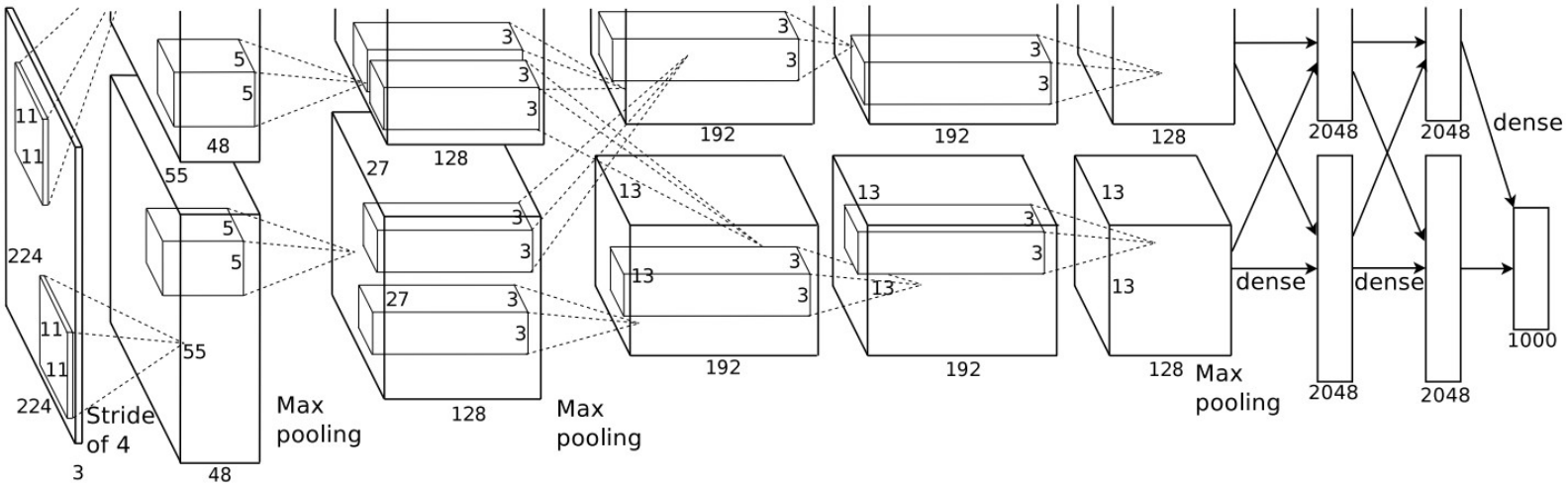
Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, november 1998.

AlexNet



[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

AlexNet



Details/Retrospectives:

- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% -> **15.4%**

VGGNet

[Simonyan and Zisserman, 2014]

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

best model

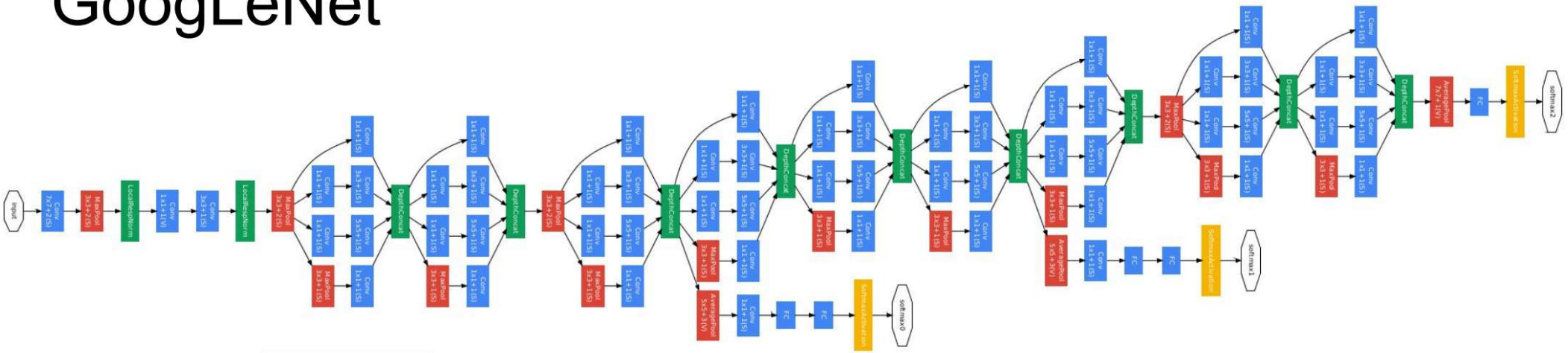
7.3% top 5 error

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

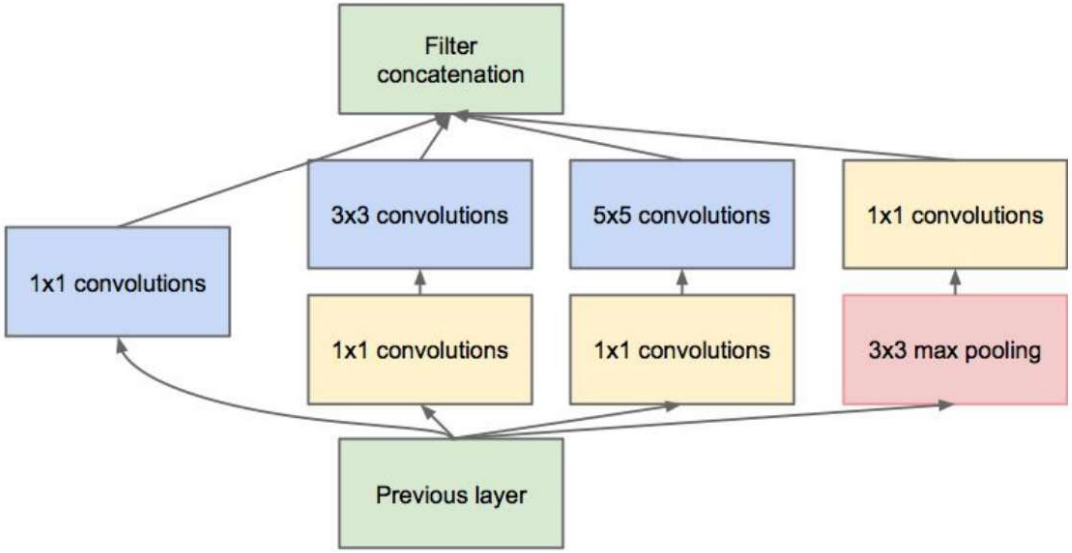
Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

GoogLeNet



[Szegedy et al., 2014]

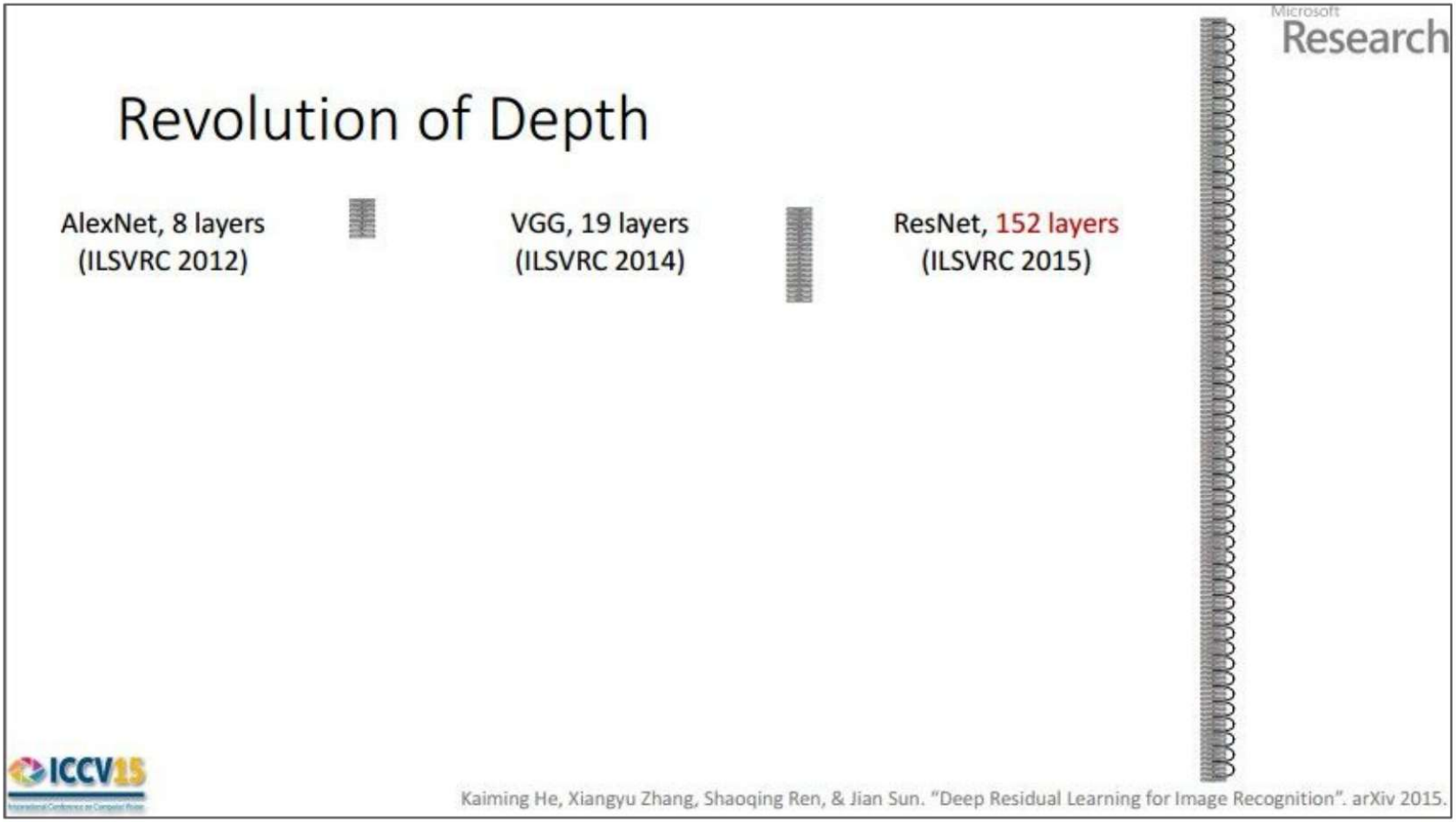


Inception module

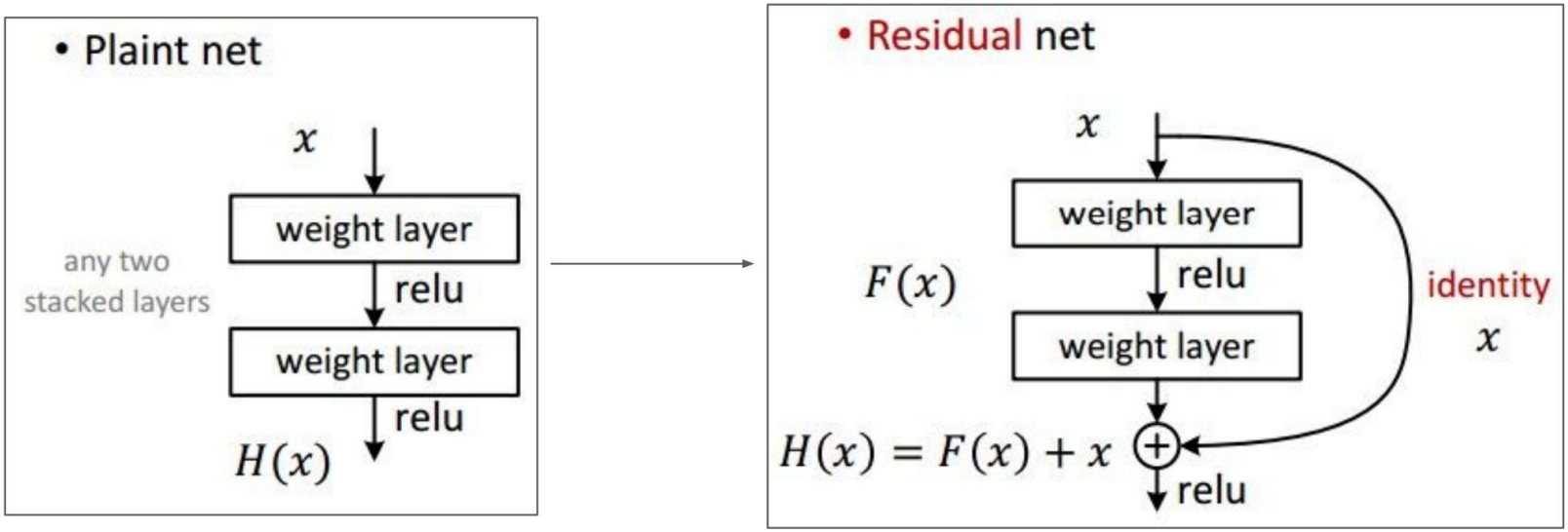
ILSVRC 2014 winner (6.7% top 5 error)

ResNet *[He et al., 2015]*

ILSVRC 2015 winner (3.6% top 5 error)



ResNet

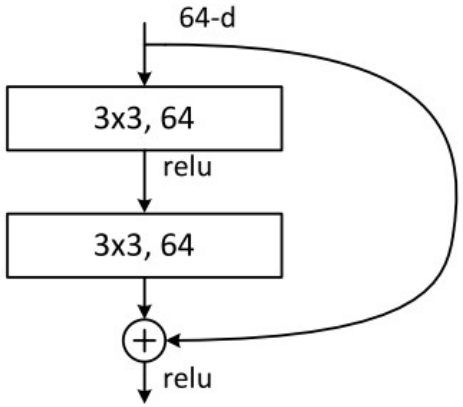

• Plaint net
x
weight layer
any two
stacked layers
relu
weight layer
relu
$H(x)$
• Residual net
x
weight layer
$F(x)$
relu
identity
x
weight layer
$H(x) = F(x) + x$
relu

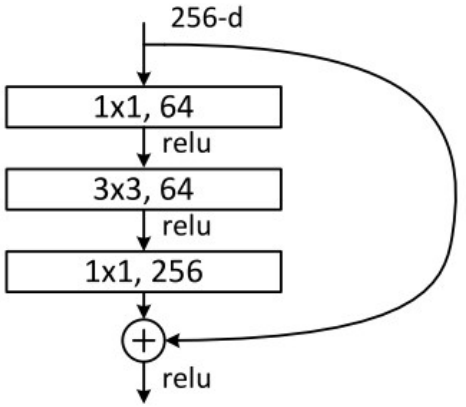
ResNet *[He et al., 2015]*

- Batch Normalization after every CONV layer
- Xavier/2 initialization from He et al.
- SGD + Momentum (0.9)
- Learning rate: 0.1, divided by 10 when validation error plateaus
- Mini-batch size 256
- Weight decay of 1e-5
- No dropout used

Resnet

- Para redes más profundas, es necesario reducir las computaciones.
- Conexiones en cuello de botella



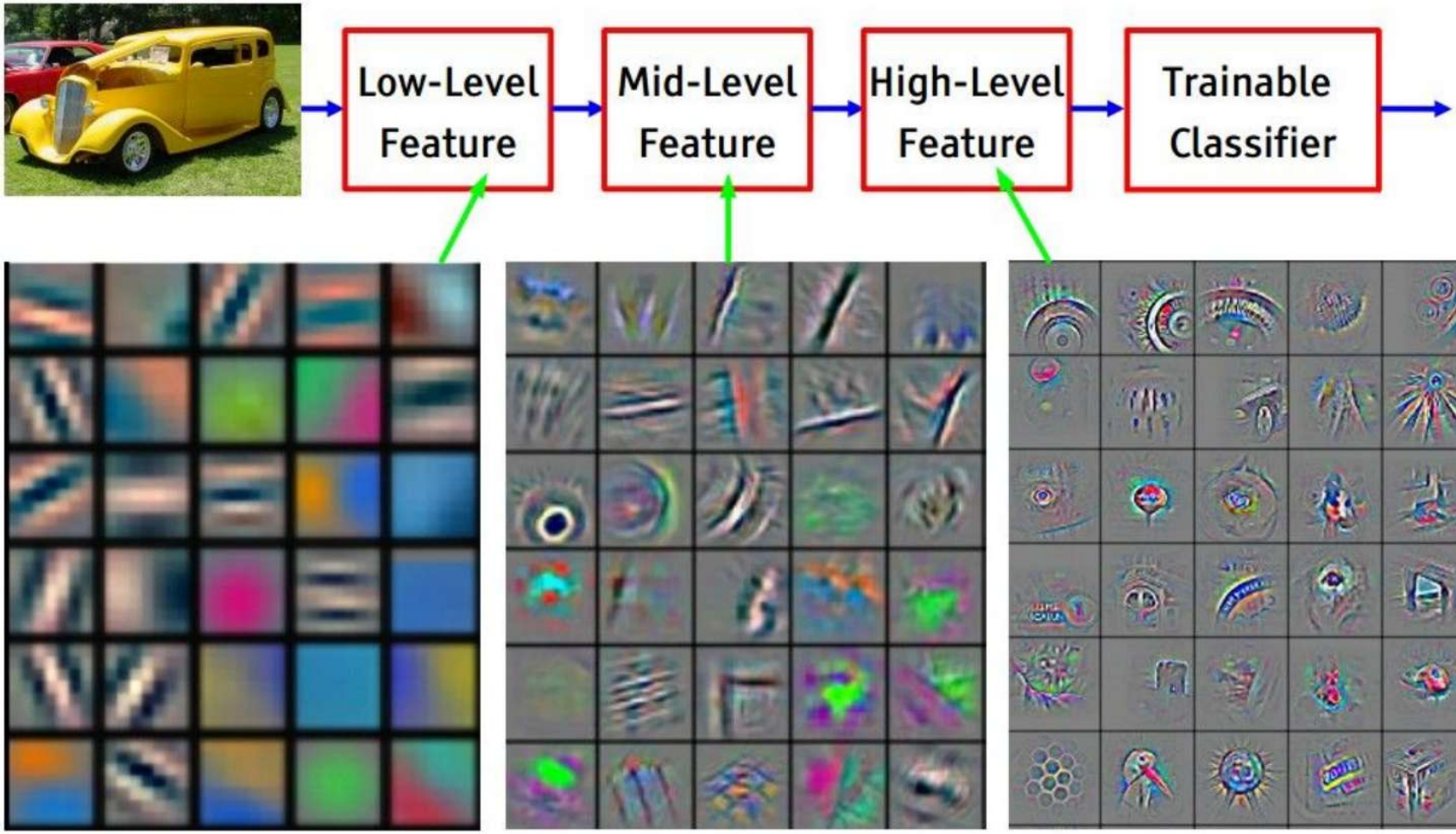


ResNet *[He et al., 2015]*

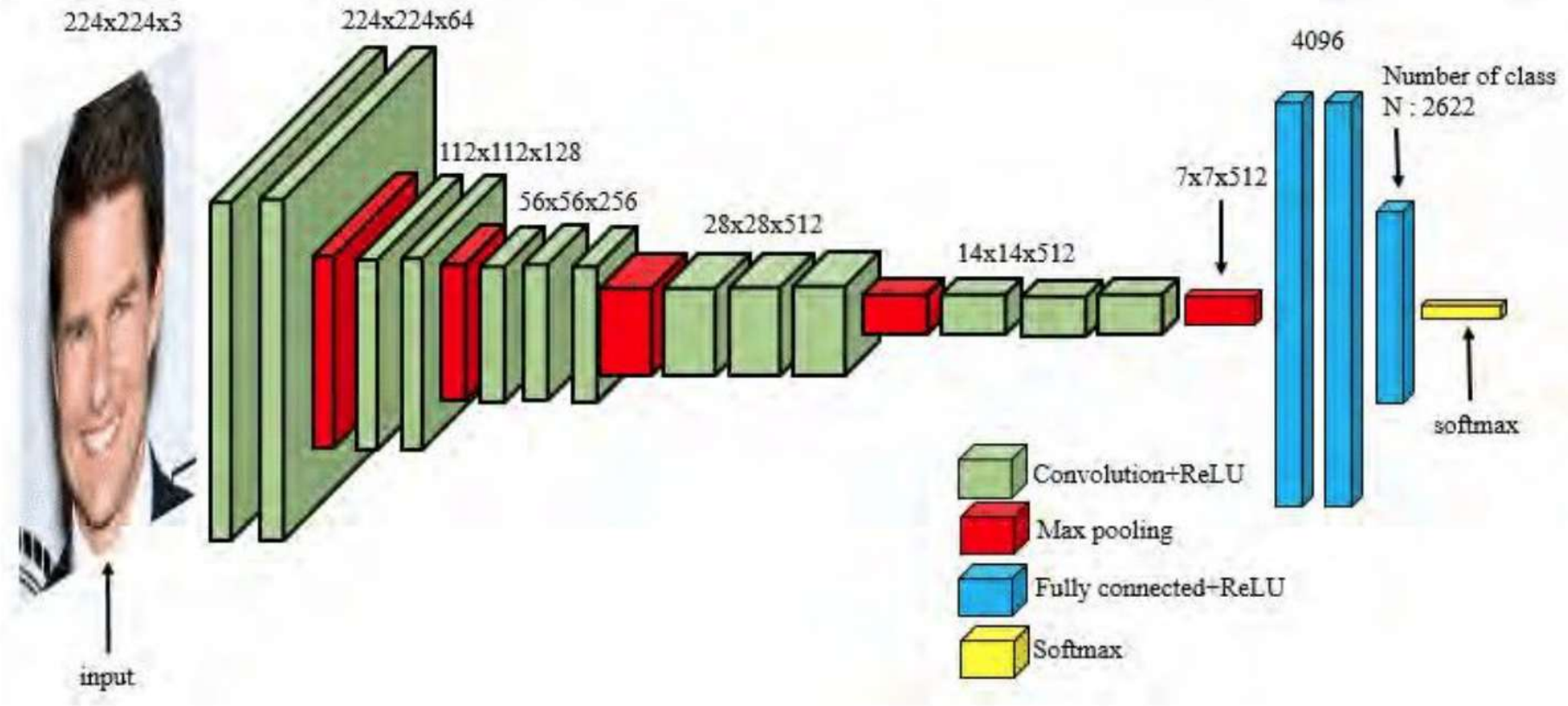
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

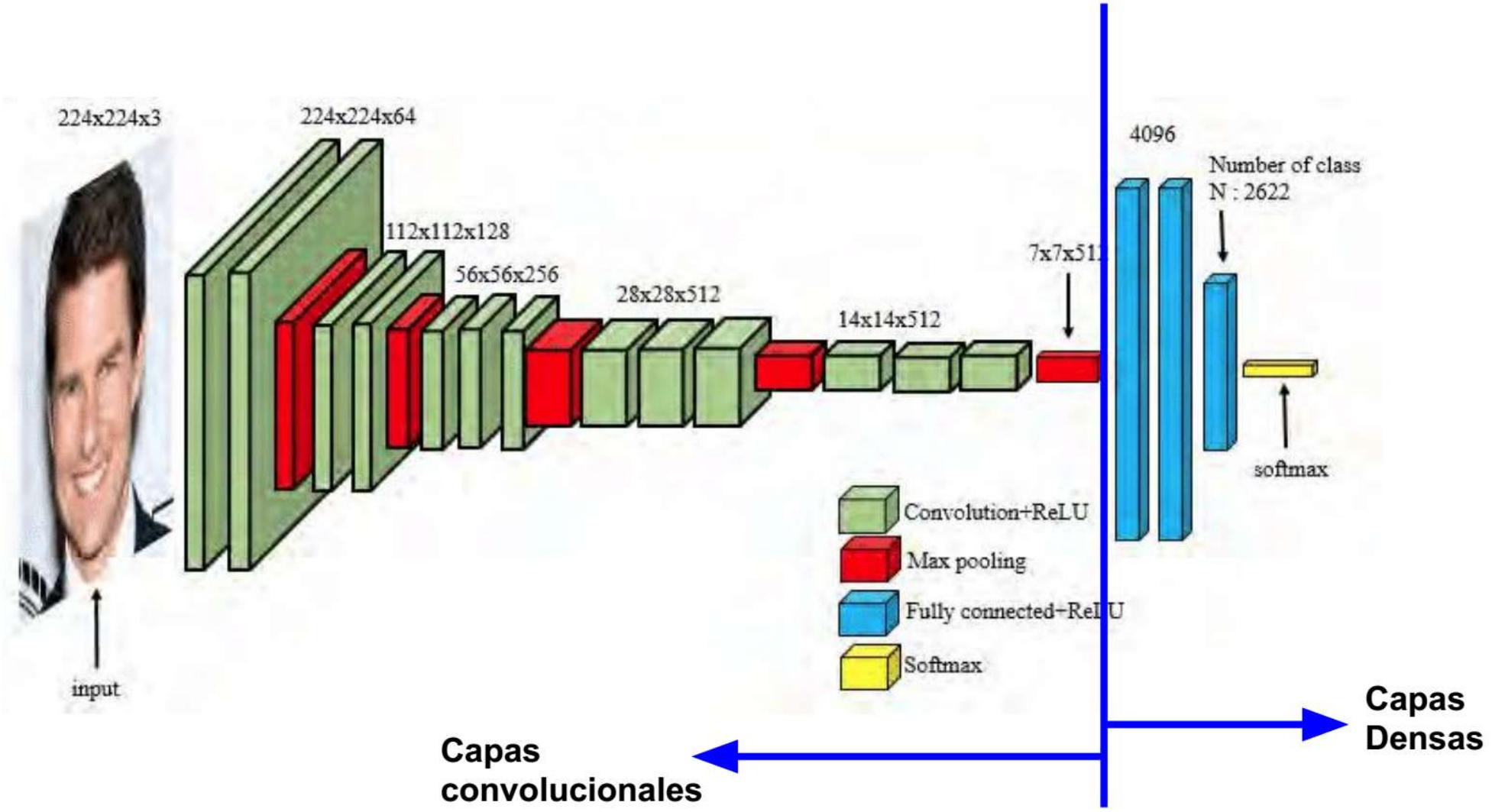
Deep Learning

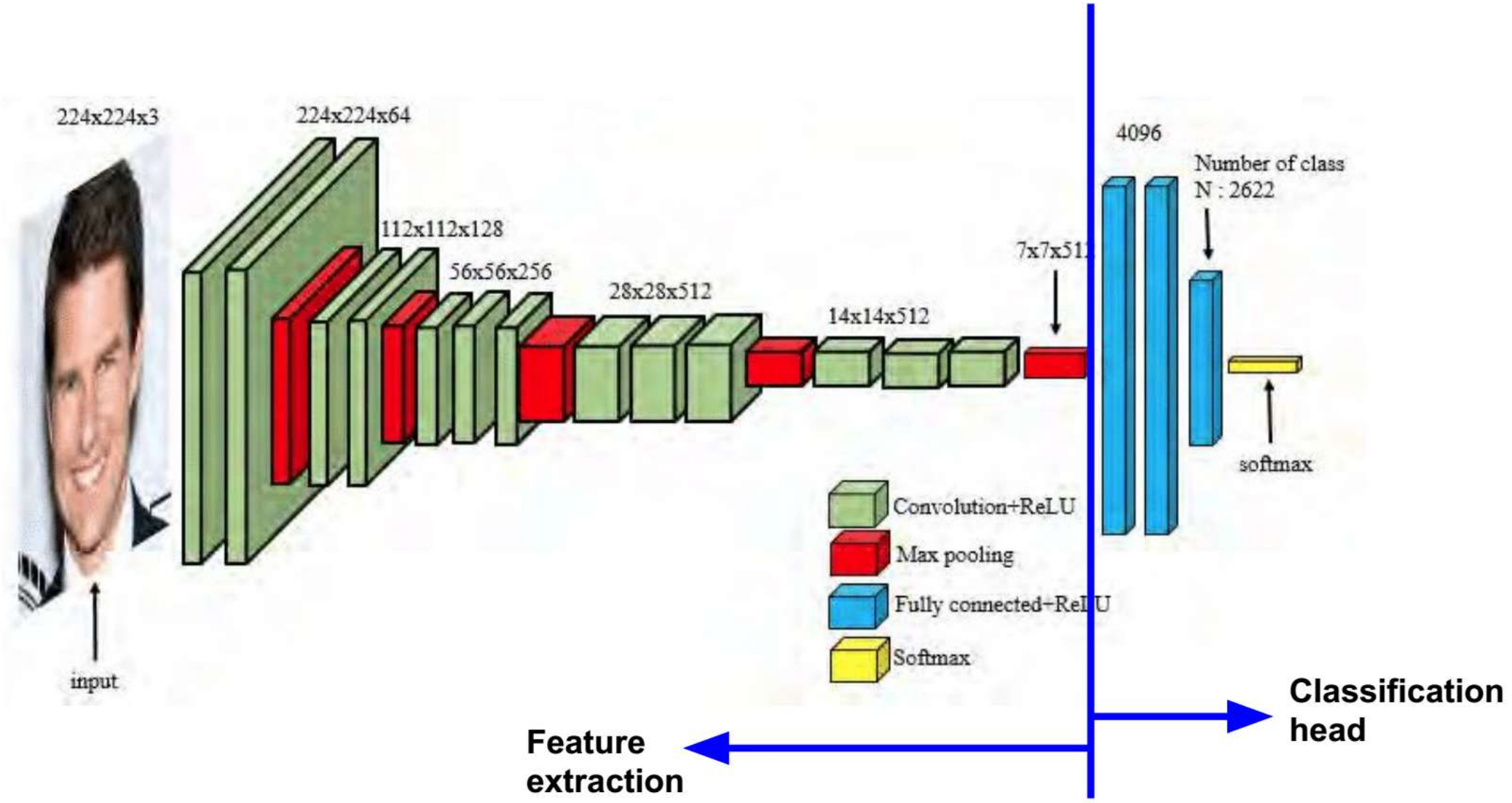
Transfer Learning



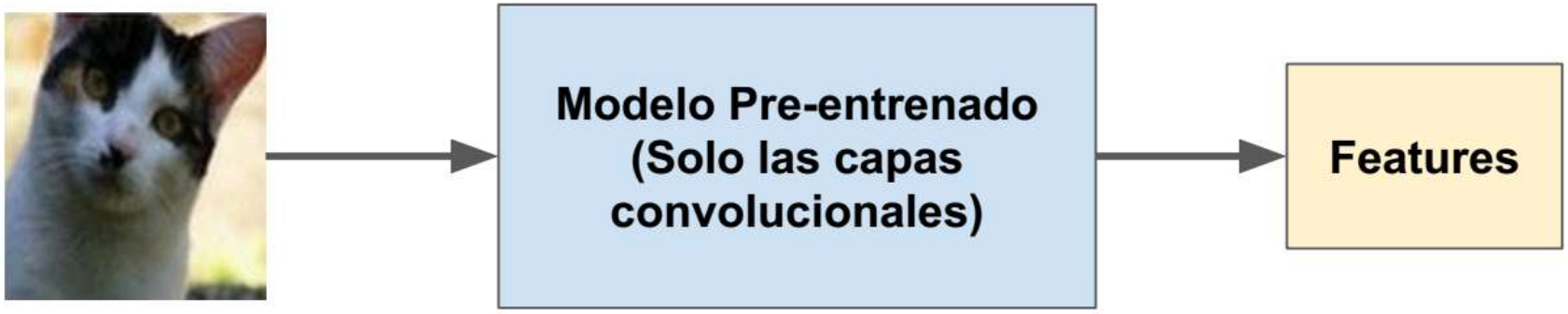
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]


224x224x3
224x224x64
112x112x128
56x56x256
28x28x512
14x14x512
7x7x512
4096
Number of class
N : 2622
softmax
input
Convolution+ReLU
Max pooling
Fully connected+ReLU
Softmax

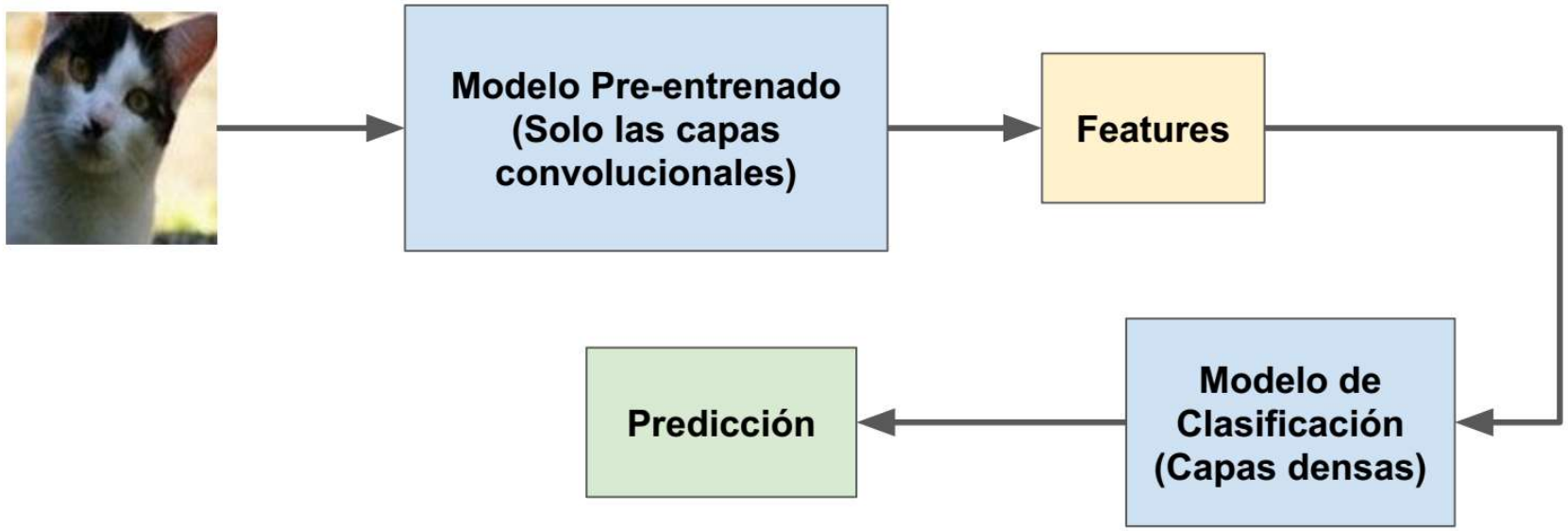

224x224x3
224x224x64
112x112x128
56x56x256
28x28x512
14x14x512
7x7x51
4096
Number of class
N : 2622
softmax
input
Convolution+ReLU
Max pooling
Fully connected+ReLU
Softmax
Capas convolucionales
Capas Densas


224x224x3
224x224x64
112x112x128
56x56x256
28x28x512
14x14x512
7x7x512
4096
Number of class
N : 2622
softmax
input
Convolution+ReLU
Max pooling
Fully connected+ReLU
Softmax
Feature extraction
Classification head

Transfer Learning



Transfer Learning



Transfer Learning - entrenamiento