Deep Learning para visión por Computador. Redes Neuronales de Convonlución CNN

Qué es una CNN

Arquitectura de una red neuronal CNN

Capas de una CNN. Eliminación de Ruido

Arquitecturas de Redes Conocidas

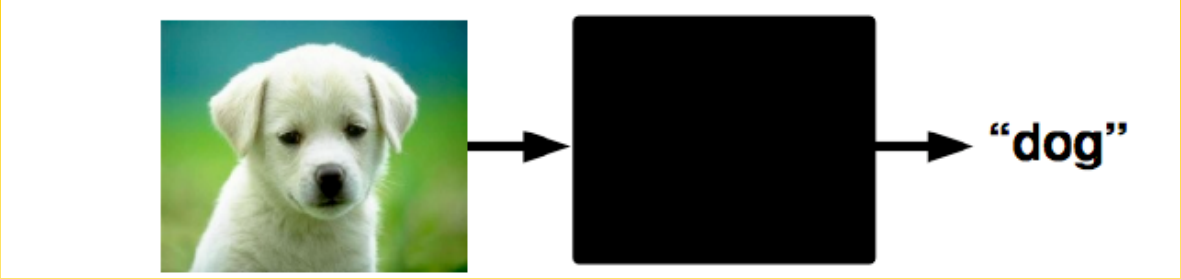
Transferencia de Aprendizaje

Qué es una CNN

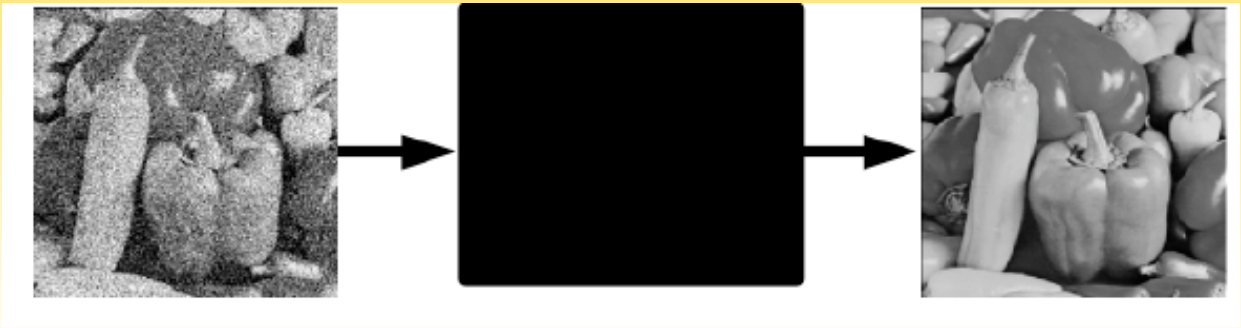
- **CNN**: Es un método de aprendizaje profundo (‘deep learning’) supervisado.
- Como en cualquier otro método de *‘deep learning’* es una composición de transformaciones no lineales sobre los datos
- Objetivo: Aprender una representación útil directamente de los datos.

Aprendizaje Supervisado Ejemplos

Clasificación

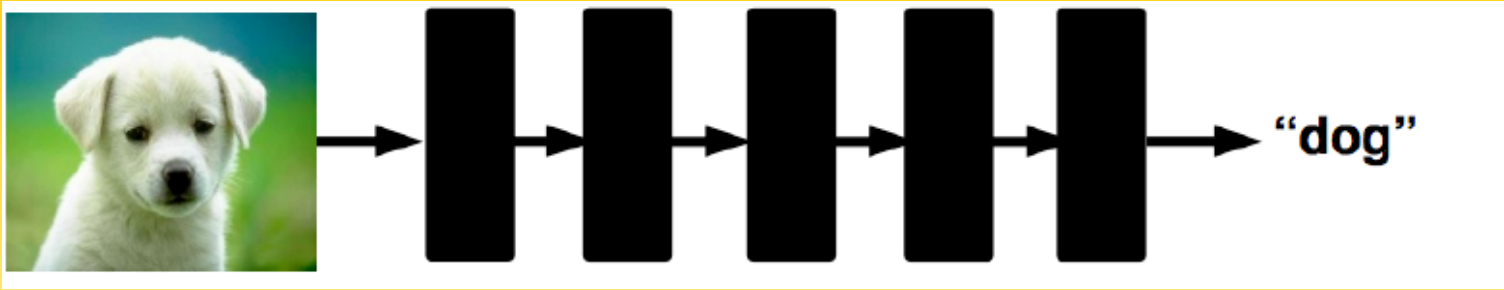


Regresión

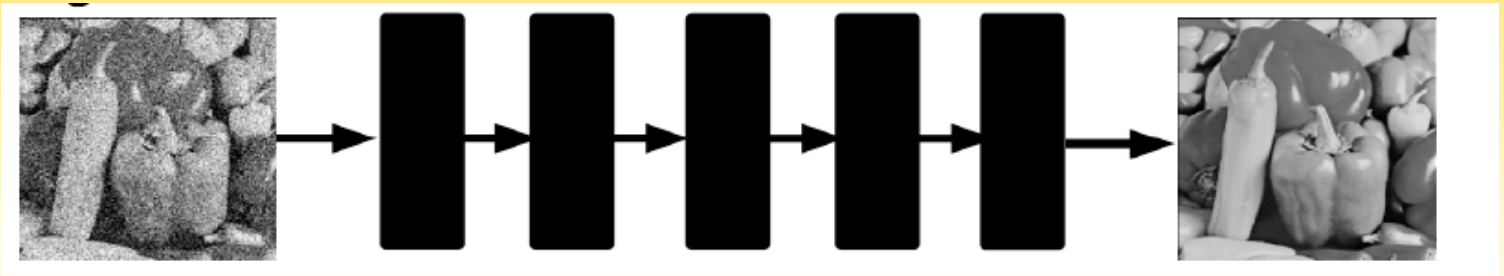


Deep Learning
Aprendizaje Supervisado
Ejemplos

Clasificación



Regresión



Deep Learning para visión por Computador. Redes Neuronales de Convonlución CNN

Qué es una CNN

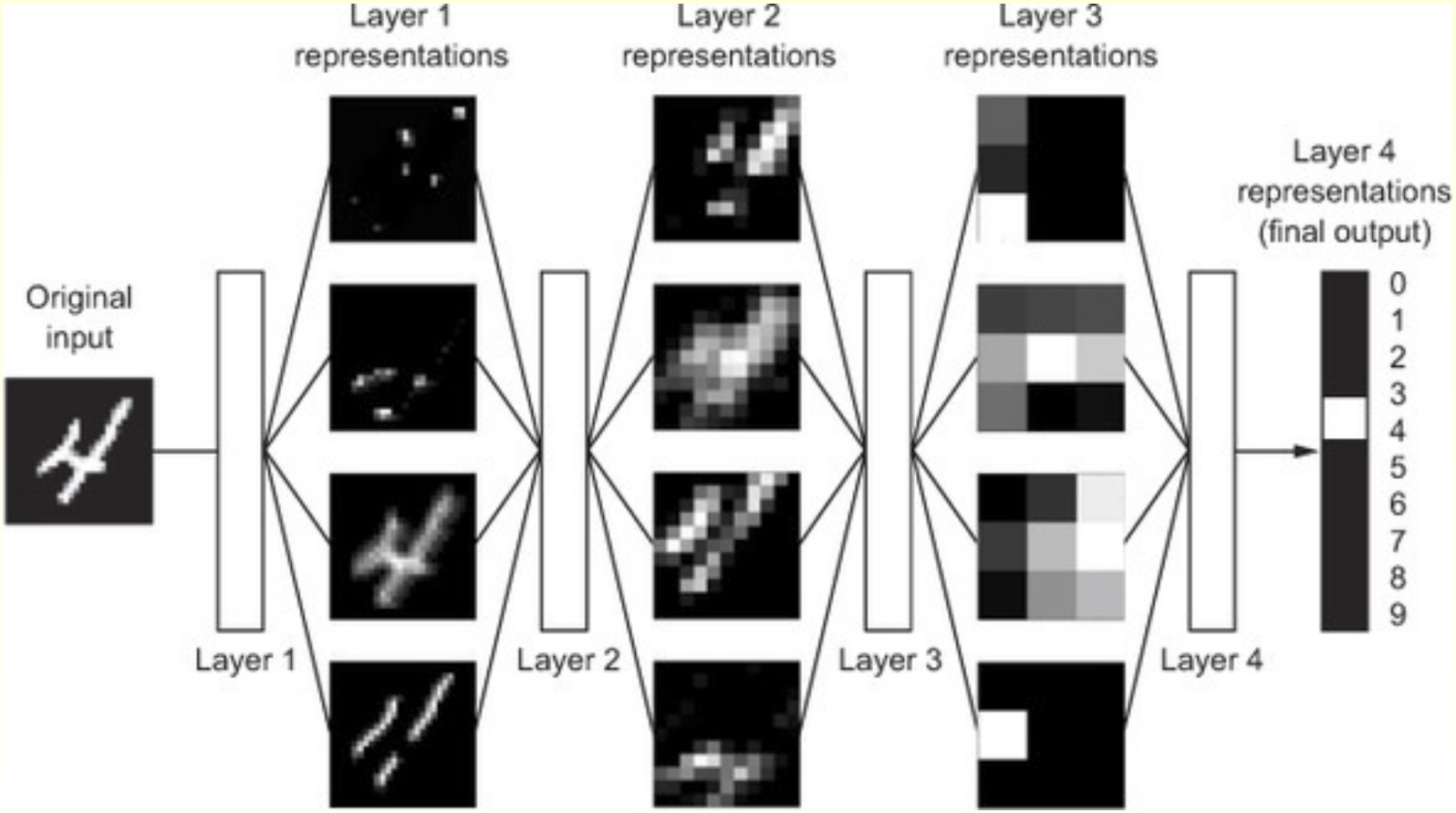
Arquitectura de una red neuronal CNN

Capas de una CNN. Eliminación de Ruido

Arquitecturas de Redes Conocidas

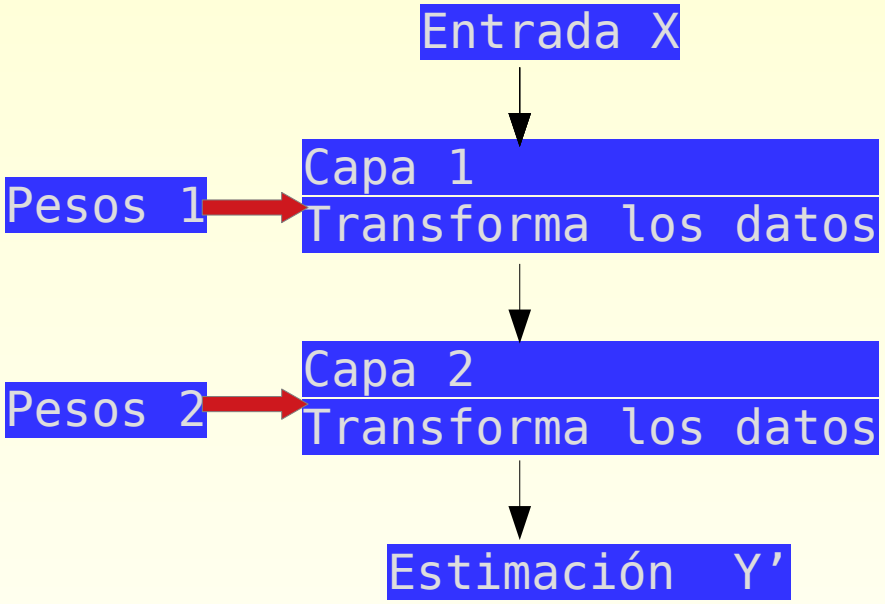
Transferencia de Aprendizaje

CNN: es una red neuronal donde en su arquitectura encontramos capas de convolución.

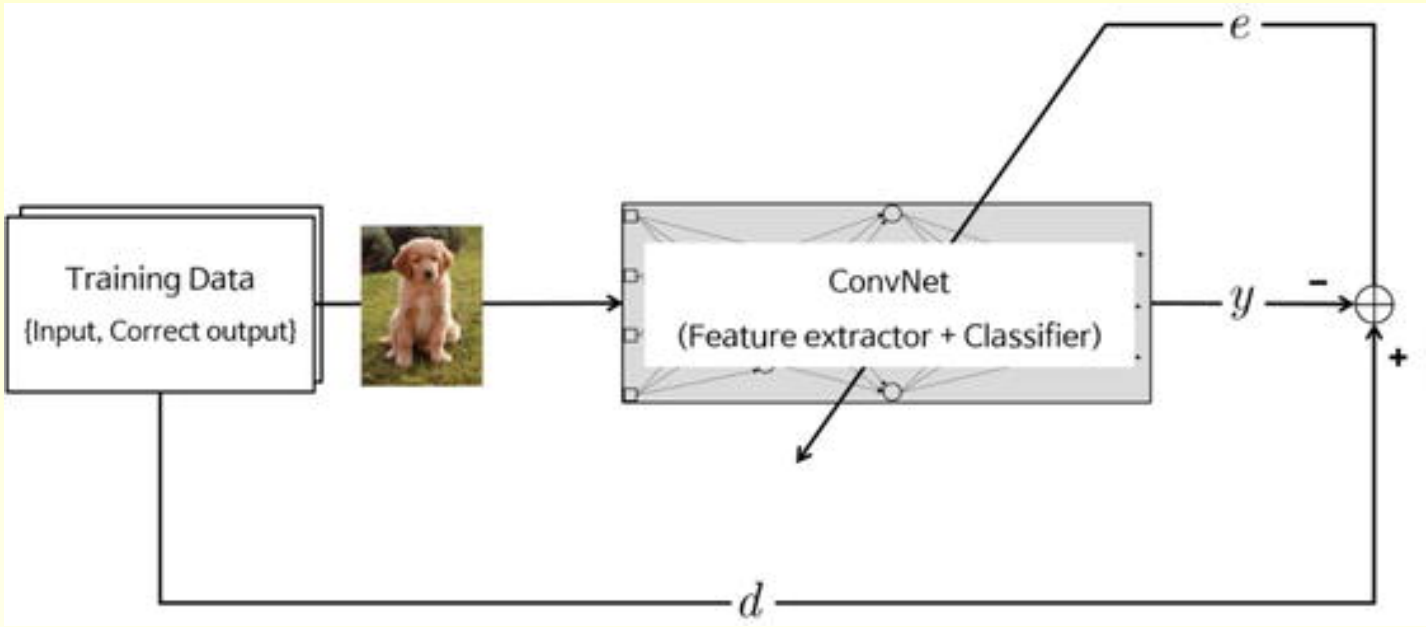


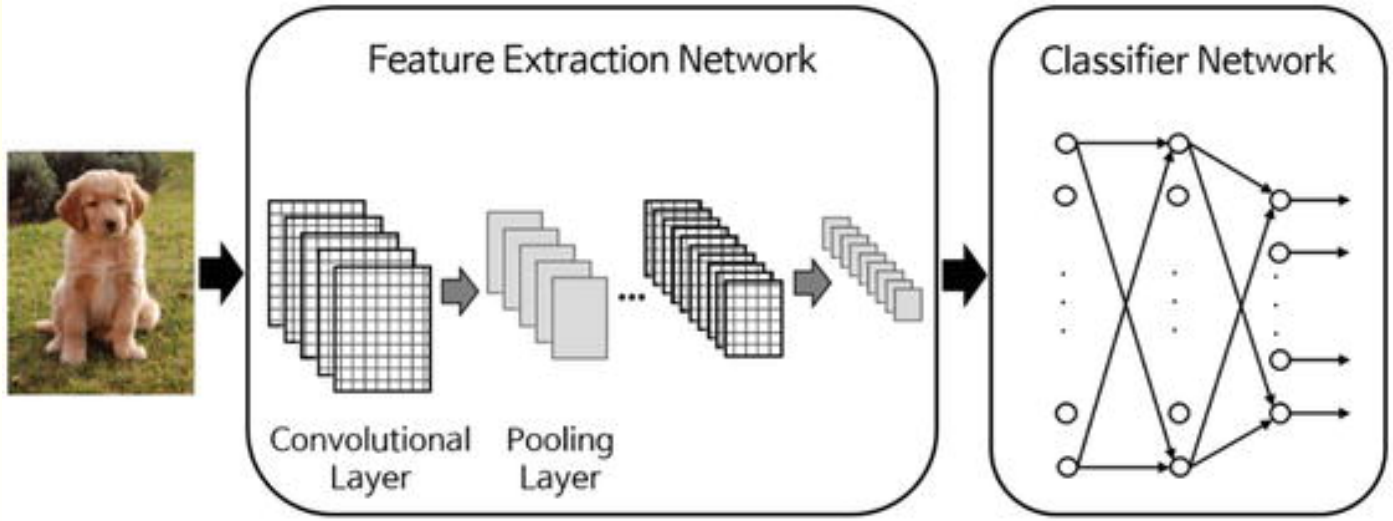
Una capa dentro de una red neuronal es una transformación compuesta por un conjunto de pesos.

Objetivo: Aprender los pesos de las capas

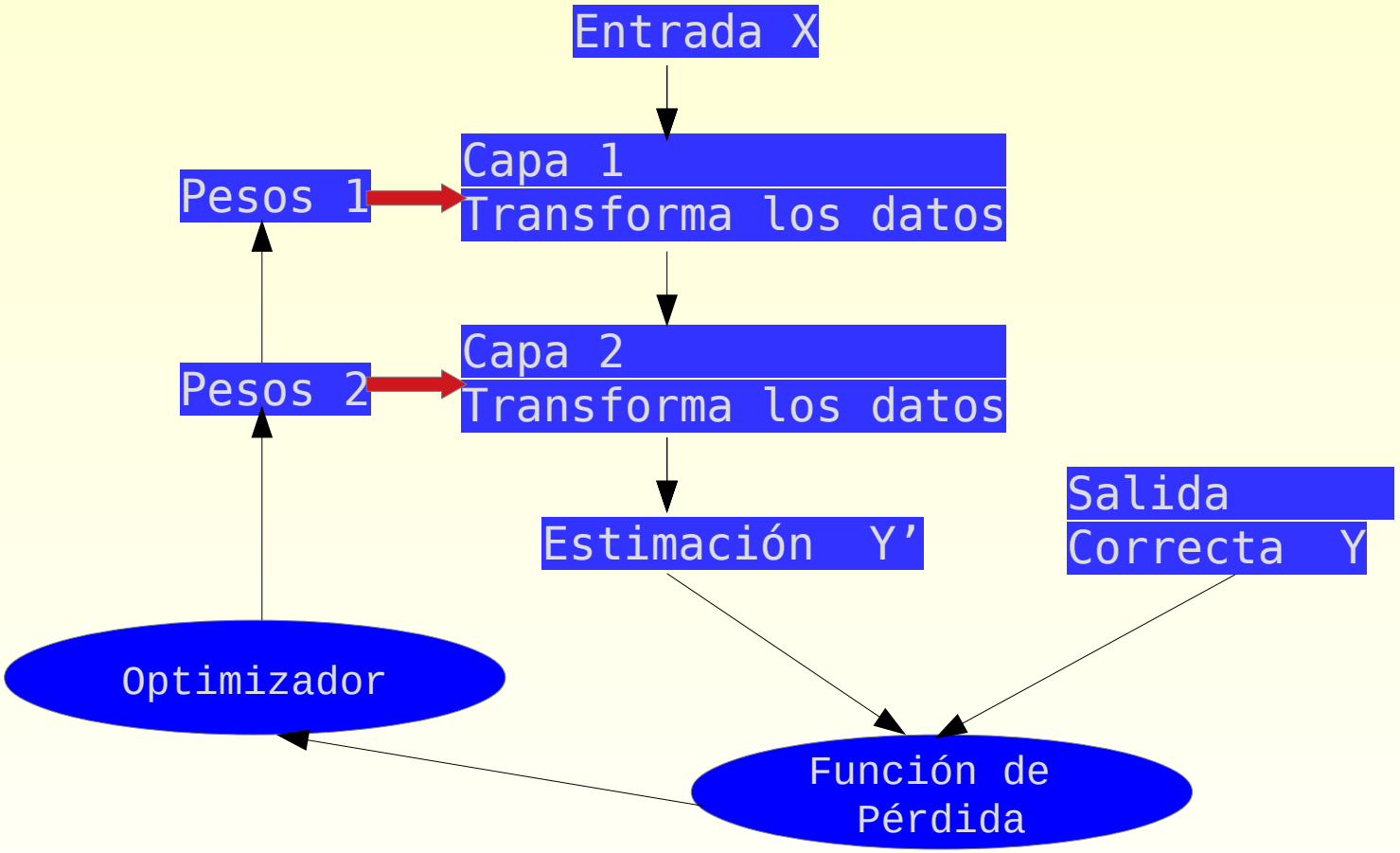


Arquitectura de una red neuronal CNN





Arquitectura de una red neuronal CNN



Deep Learning para visión por Computador. Redes Neuronales de Convonlución CNN

Qué es una CNN

Arquitectura de una red neuronal CNN

Capas de una CNN. Eliminación de Ruido

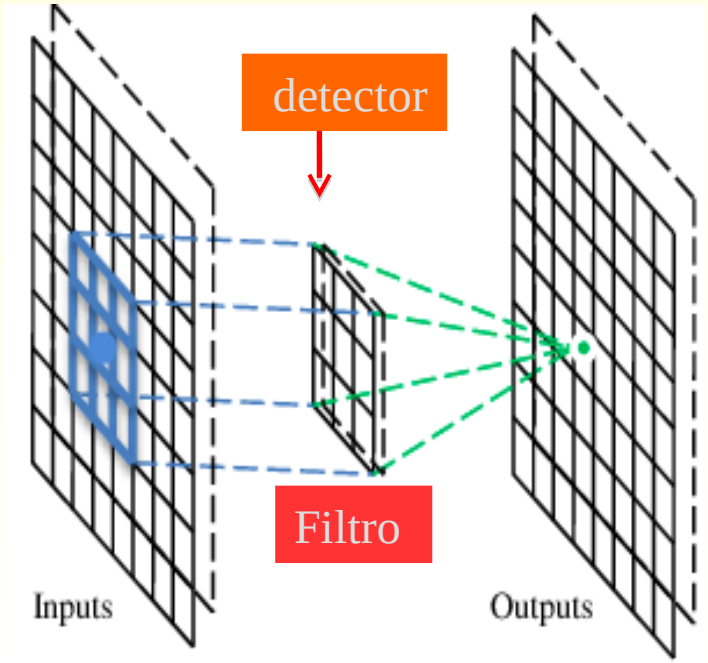
Arquitecturas de Redes Conocidas

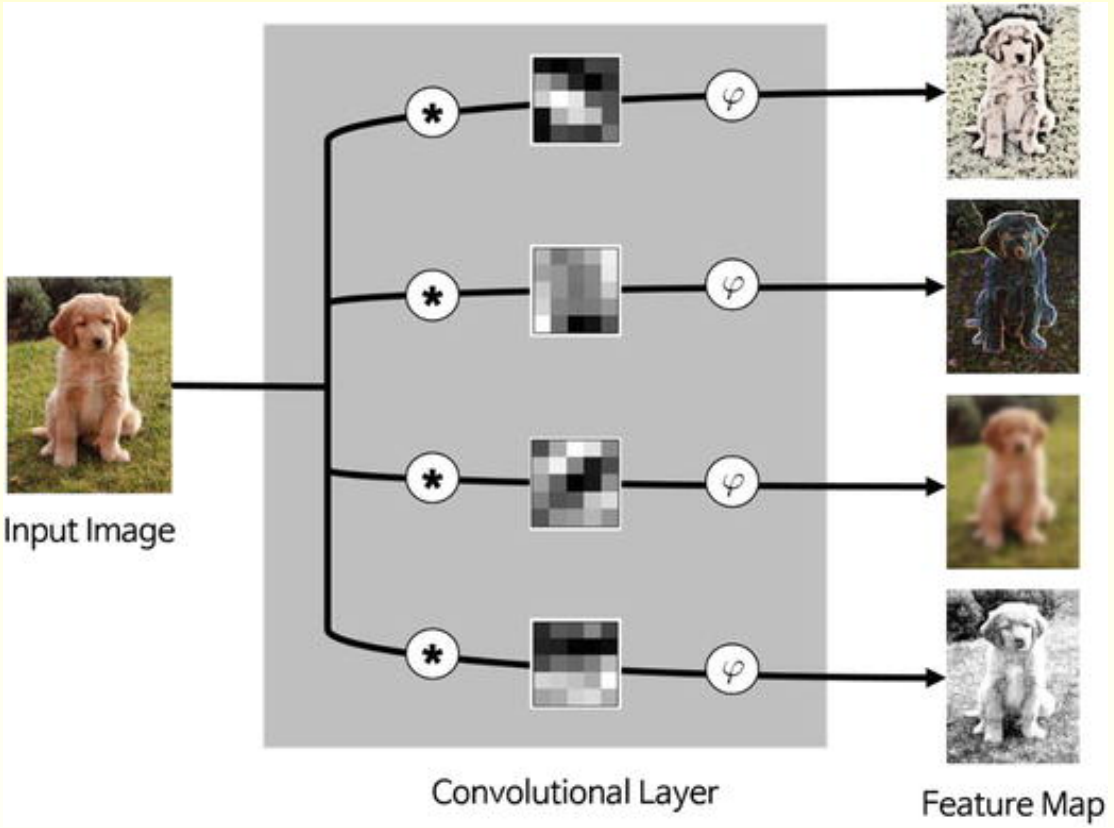
Transferencia de Aprendizaje

Capas de convolución:

- A partir de una entrada obtiene un mapa de rasgos
- Los mapas de rasgos acentúan características únicas de la entrada

Las capas de convolución se componen de un conjunto de filtros → Filtros de Convolución.





Capas de convolución versus Capas Completamente Conectadas

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

imagen

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

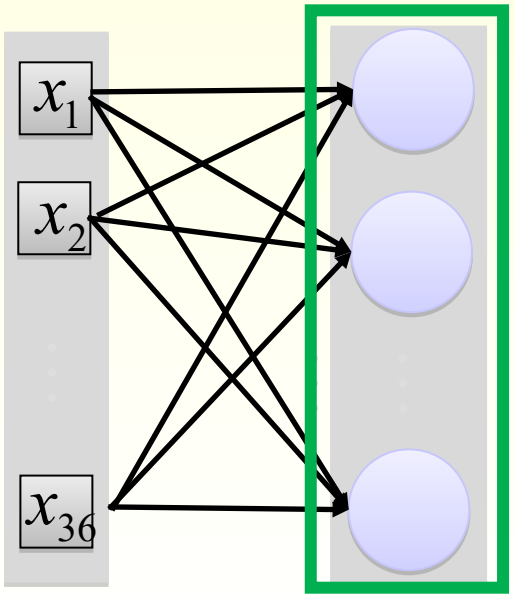
-1	1	-1
-1	1	-1
-1	1	-1

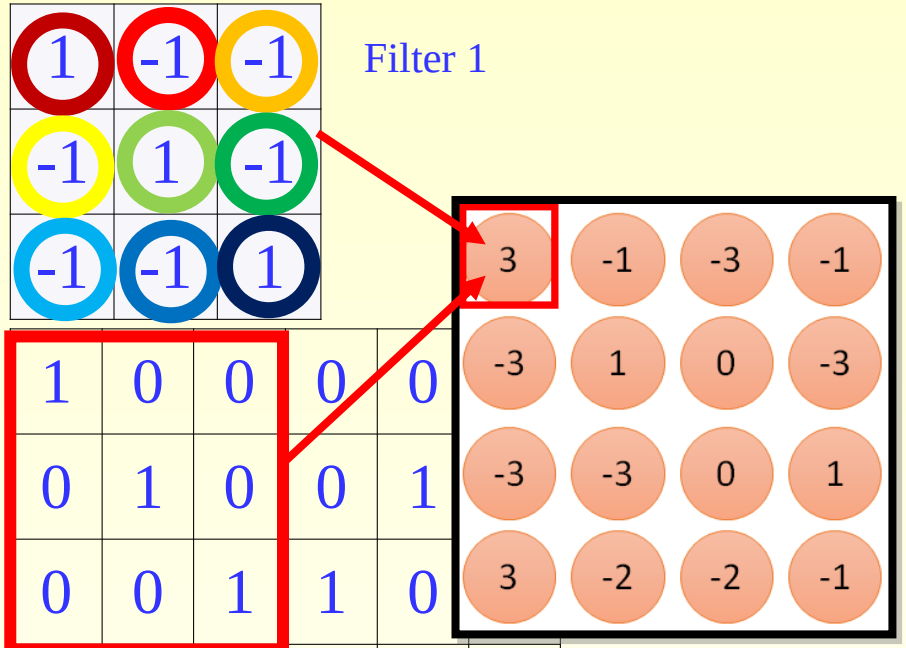
convolución

-1	-1	-1	-1
-1	-1	-2	1
-1	-1	-2	1
-1	0	-4	3

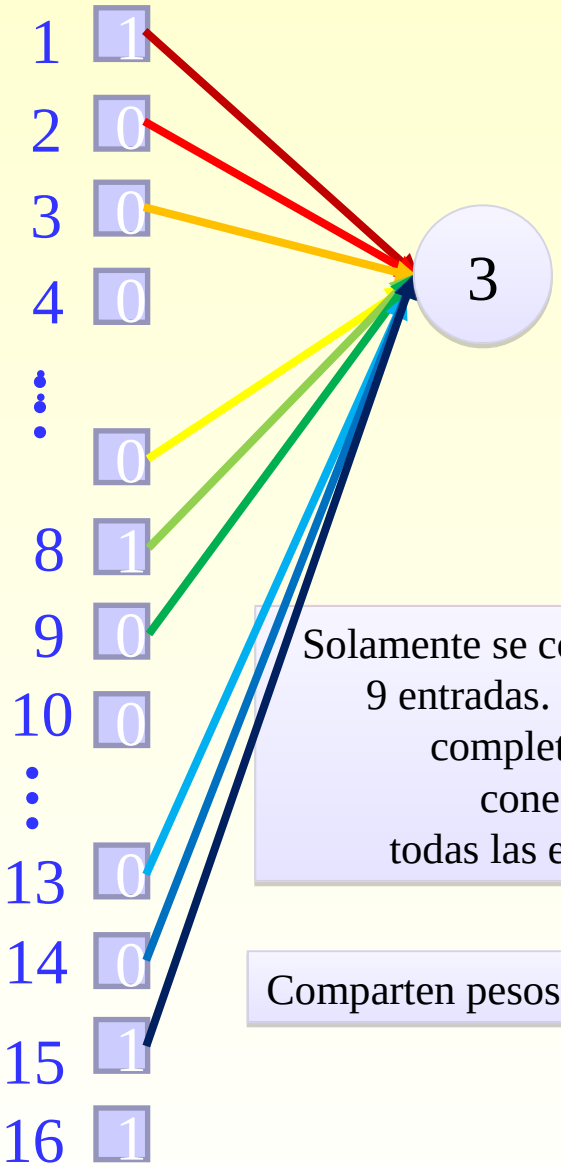
Capa completamente conectada

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0





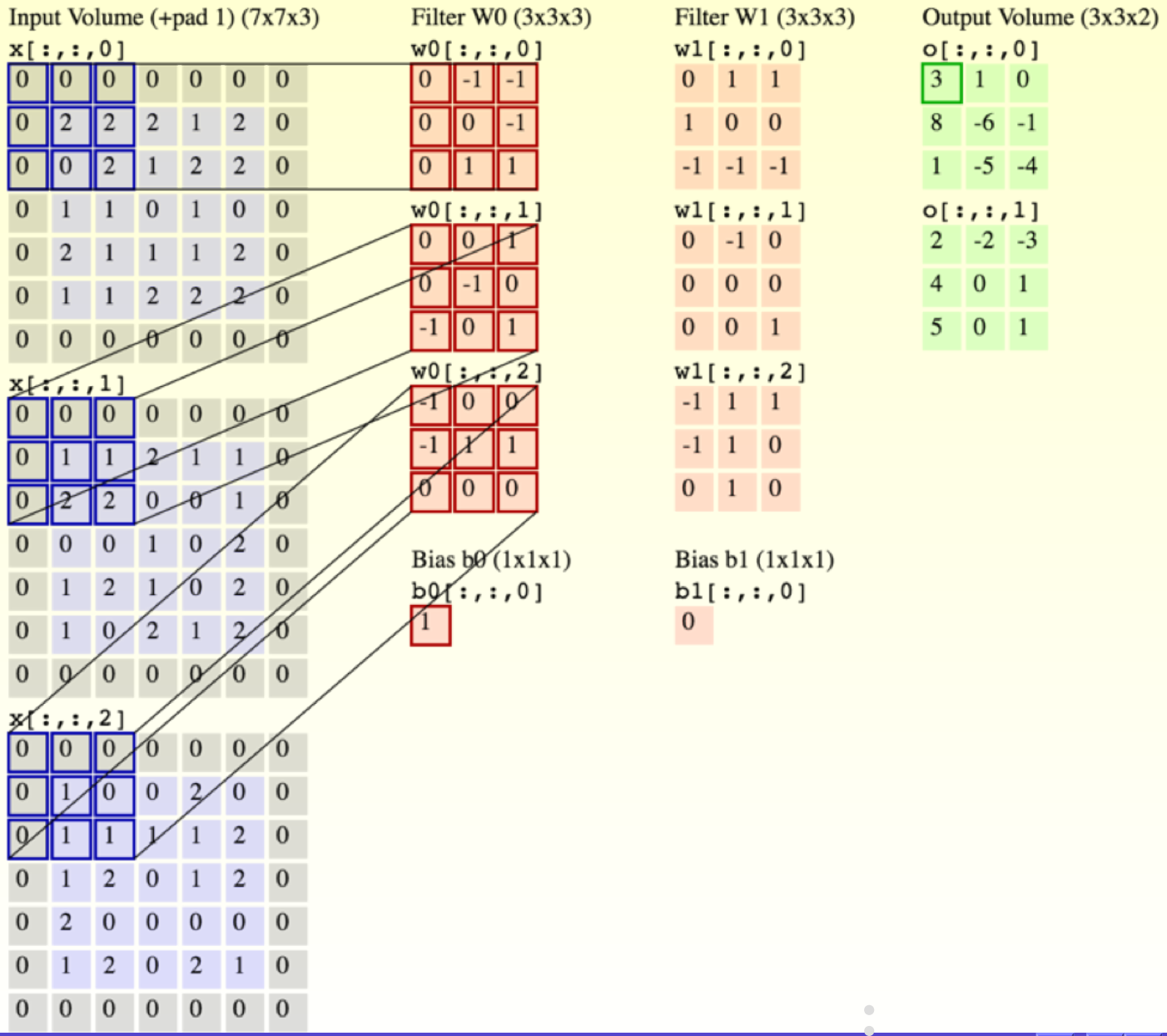
Muy pocos parámetros



Solamente se conectan 9 entradas. No está completamente conectada a todas las entradas

Comparten pesos

Capas de Convolución



Por qué usar agrupamientos (usando las capas Pooling):

- Max Pooling
- Average Pooling

pájaro

Submuestrreo

pájaro

Caracterizamos la imagen o mapa de rasgos con menos parámetros

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

Filter 1

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

Filter 2

3	-1
-3	1

-3	-1
0	-3

-3	-3
3	-2

0	1
-2	-1

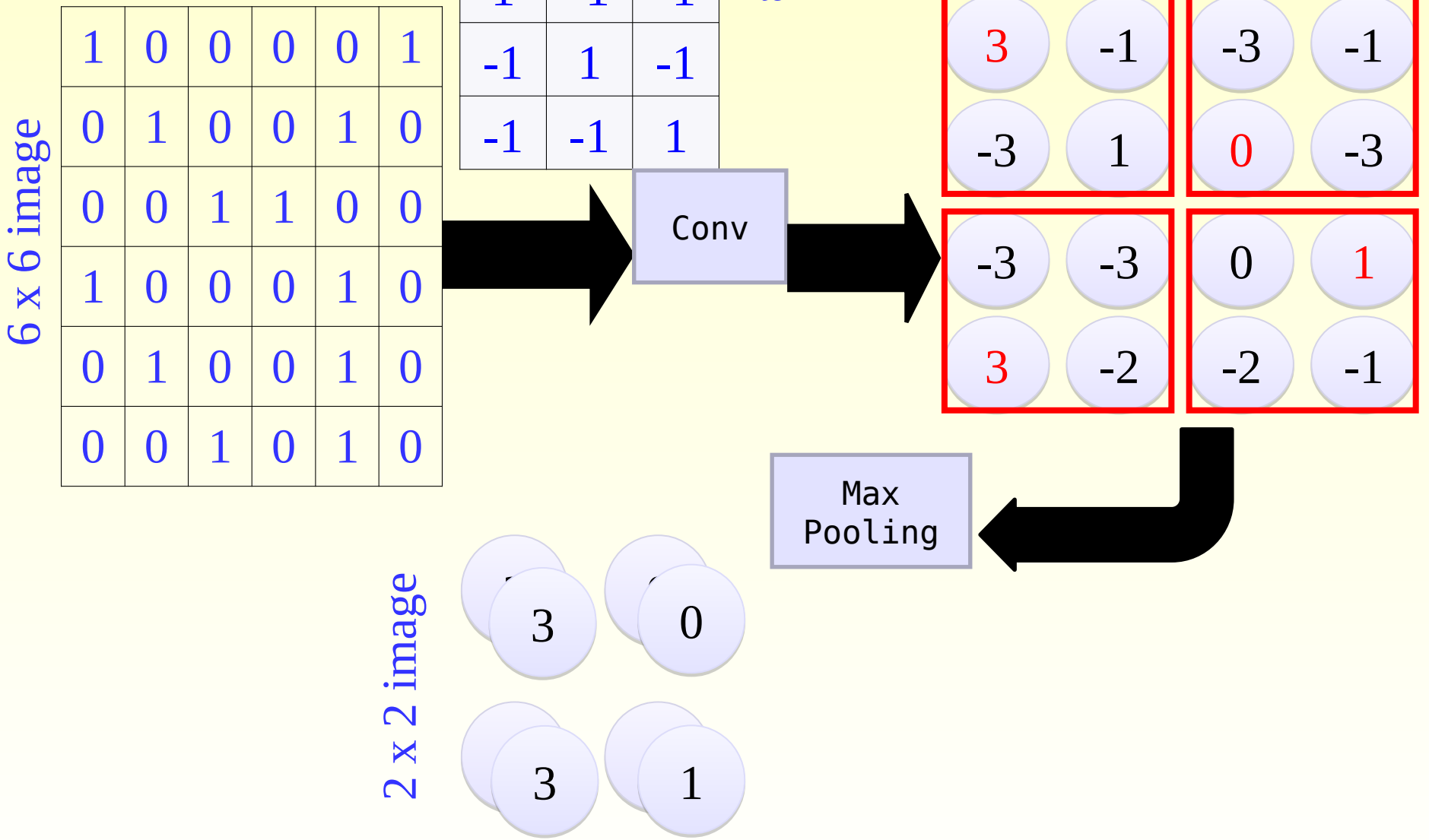
-1	-1
-1	-1

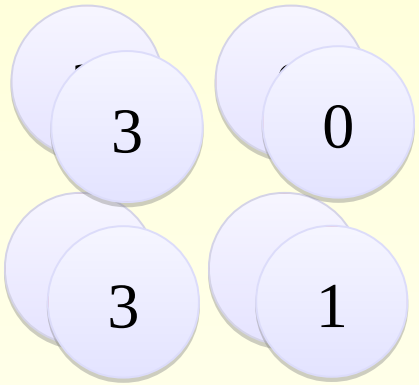
-1	-1
-2	1

-1	-1
-1	0

-2	1
-4	3

Capas Max Pooling

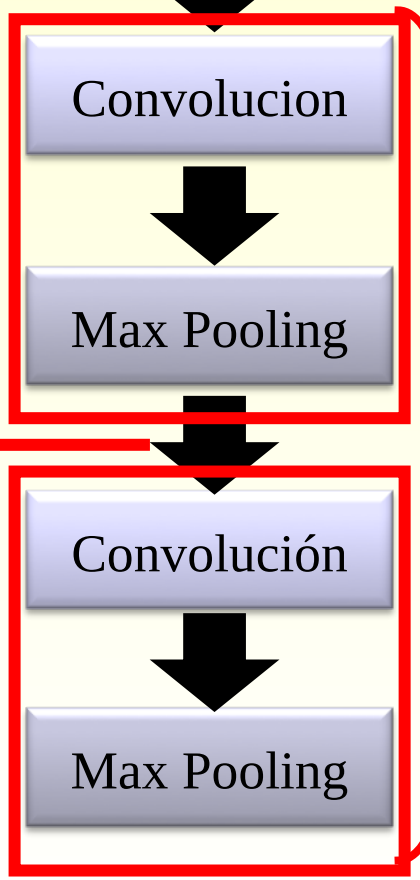




Una imagen nueva

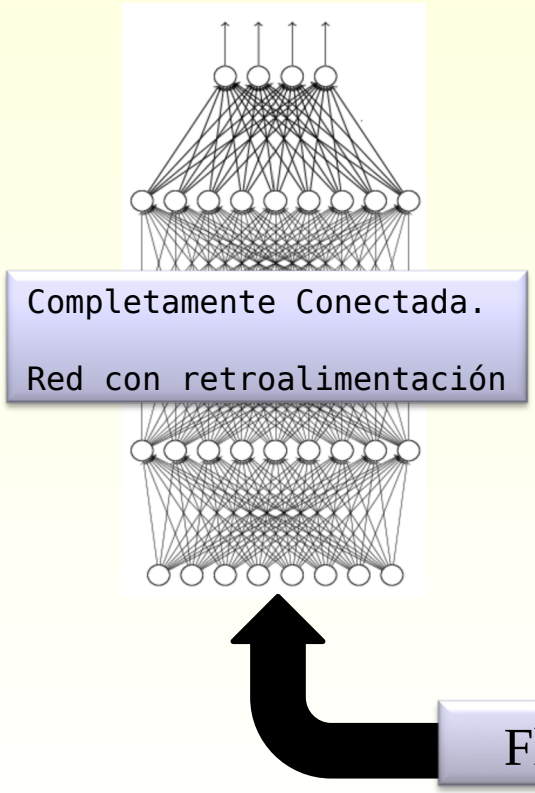
Una imagen más pequeña que la original

La salida de la capa de Convolución puede dar lugar a más de una imagen, tantas como filtros.



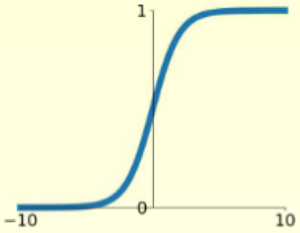
Se puede repetir muchas veces

Esquema genérico de una CNN



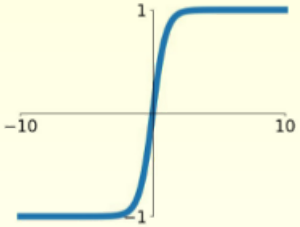
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



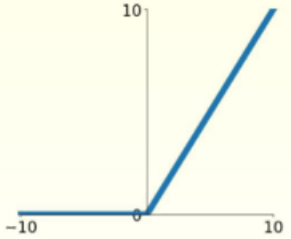
tanh

$$\tanh(x)$$



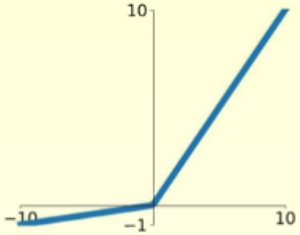
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

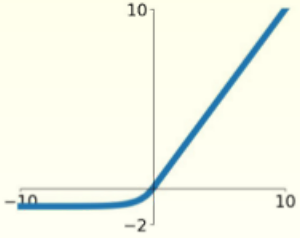


Maxout

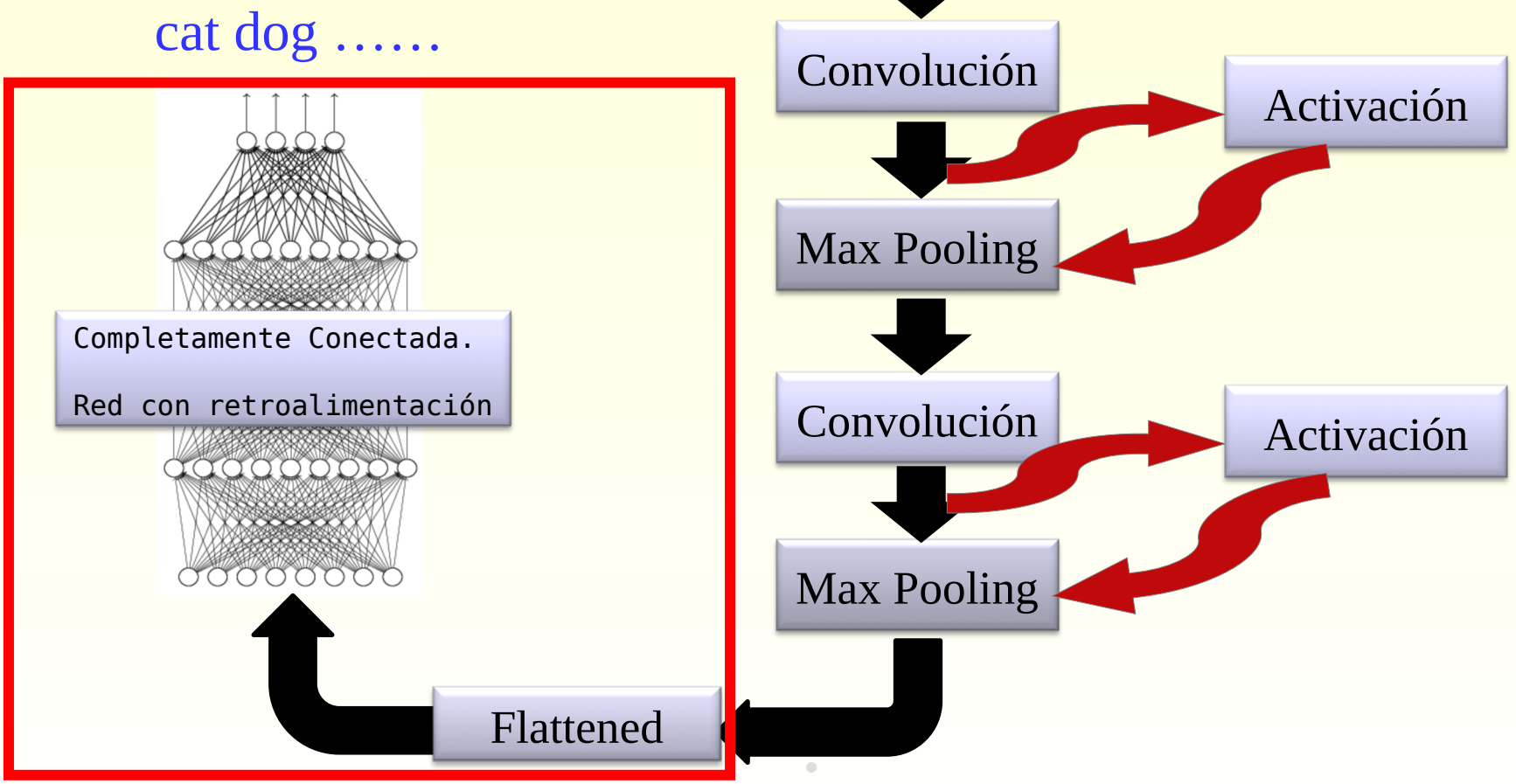
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

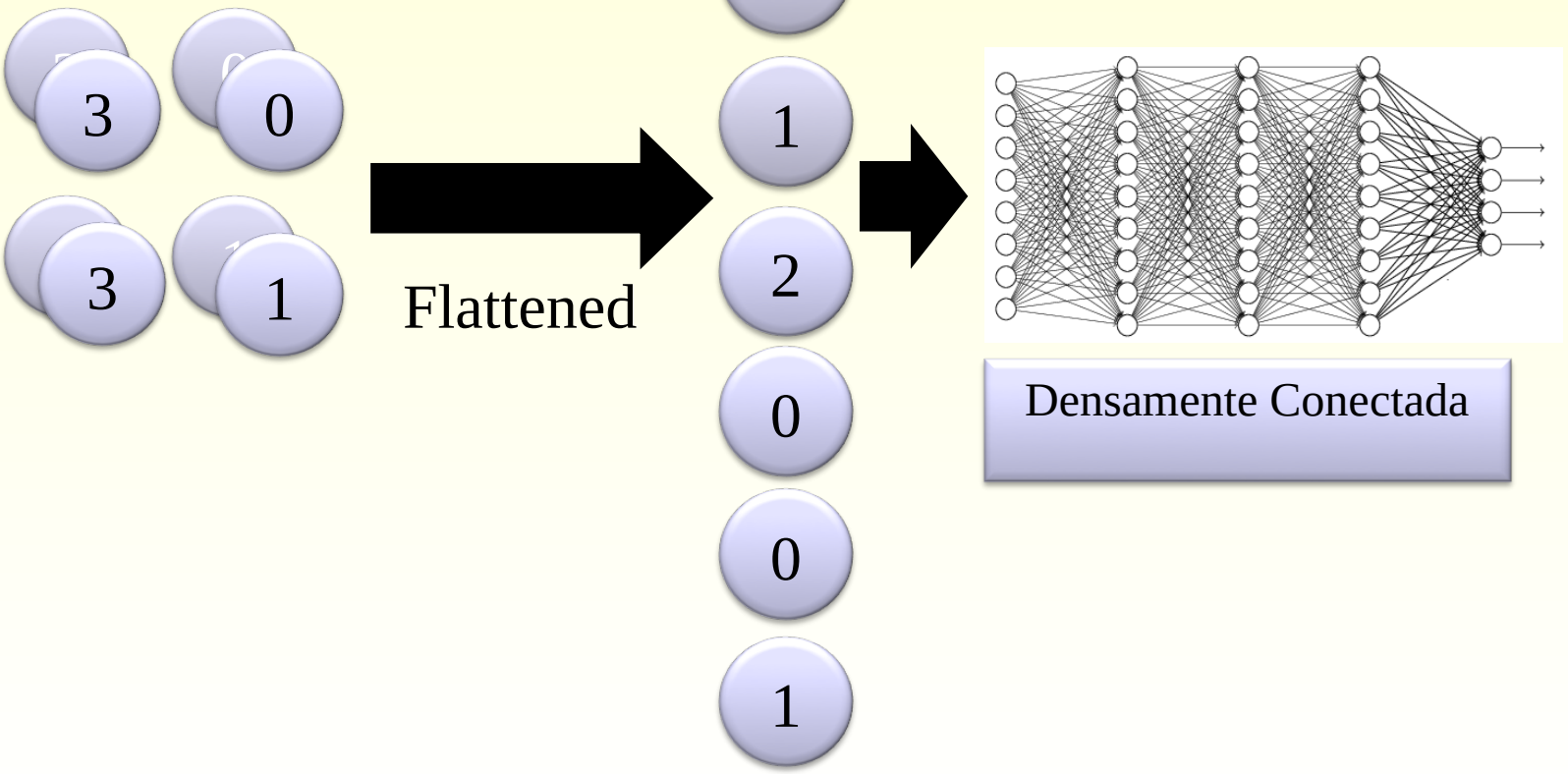
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Esquema genérico de una CNN



Flattening



Ejemplo de CNN para eliminar ruido

59 capas

```
  1   'InputLayer'              Image Input             50x50x1 images
  2   'Conv1'                   Convolution             64 3x3x1 convolutions with stride [1  1]
                                                        and padding [1  1  1  1]
  3   'ReLU1'                   ReLU                    ReLU
  4   'Conv2'                   Convolution             64 3x3x64 convolutions with stride [1  1]
                                                        and padding [1  1  1  1]
  5   'BNorm2'                  Batch Normalization     Batch normalization with 64 channels
  6   'ReLU2'                   ReLU                    ReLU
  7   'Conv3'                   Convolution             64 3x3x64 convolutions with stride [1  1] &
                                                        padding [1  1  1  1]
  8   'BNorm3'                  Batch Normalization     Batch normalization with 64 channels
  9   'ReLU3'                   ReLU                    ReLU
 10   'Conv4'                   Convolution             64 3x3x64 convolutions with stride [1  1]
                                                        and padding [1  1  1  1]
 11   'BNorm4'                  Batch Normalization     Batch normalization with 64 channels
 12   'ReLU4'                   ReLU                    ReLU


...

 57   'ReLU19'                  ReLU                    ReLU
 58   'Conv20'                  Convolution             1 3x3x64 convolutions with stride [1  1]
                                                        and padding [1  1  1  1]
 59   'FinalRegressionLayer'    Regression Output       mean-squared-error
```

BatchNormalization

Normaliza las entradas x_i de la siguiente forma:

$$\frac{(x_i-\mu_B)}{\sigma_B{}^2+\varepsilon}$$

μ_B y σ_B Siendo la media y varianza sobre un lote de entradas.

A continuación aplica una transformación de escala y desplazamiento:

$$y_i = y * x_i + \beta$$

Además cuando la red termina del entrenamiento, esta capa calcula la media y varianza sobre el conjunto total de entrenamiento. Así sobre una imagen de tipo test usa esta media y varianza para transformar los datos.

ReLU: Una capa ReLU aplica una operación de umbralización a cada elemento de la entrada, donde cualquier valor menor que cero se le asigna cero.

$$f(x)=\begin{cases} x & si \quad x \geq 0 \\ 0 & en \quad otro \quad caso \end{cases}$$

Para problemas de regresión esta capa obtiene el error absoluto promedio (MAE) . Siendo Y la predicción y T los valores reales. El valor MAE se obtiene como:

$$L=\frac{1}{N}\sum_{n=1}^{N}\frac{1}{R}\sum_{i=1}^{R}|Y_{n,i}-T_{n,i}|$$

Siendo N el número de observaciones y R el número de respuestas

```
net = denoisingNetwork('DnCNN');
I = imread('cameraman.tif');
noisyI = imnoise(I,'gaussian',0,0.01);
denoisedI = denoiseImage(noisyI, net);
```

Original Image (left) and Noisy Image (right)

Denoised Image

```
net = denoisingNetwork('DnCNN');
net.Layers(2)
```

Convolution2DLayer with properties:

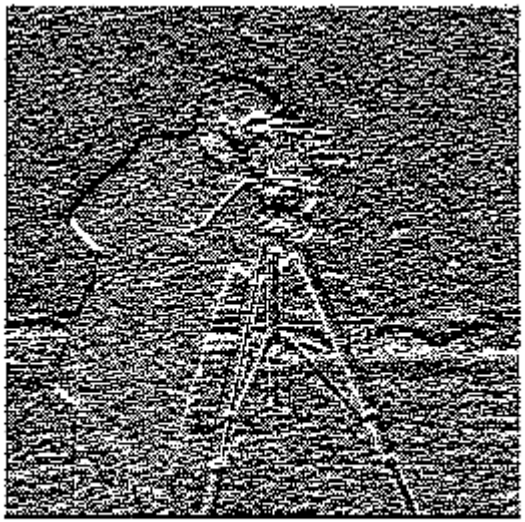
Name: 'Conv1'

Hyperparameters
FilterSize: [3 3]
NumChannels: 1
NumFilters: 64
Stride: [1 1]
PaddingMode: 'manual'
PaddingSize: [1 1 1 1]

Learnable Parameters
Weights: [3×3×1×64 single]
Bias: [1×1×64 single]

```
net.Layers(2).Weights(:,:,1,1)

    0.1201    0.0118    0.1693
   -0.0516   -0.0485   -0.1265
   -0.0307   -0.0603    0.0029
out=conv2(noiseyI,net.Layers(2).Weights(:,:,1,1))
```



Deep Learning para visión por Computador. Redes Neuronales de Convonlución CNN

Qué es una CNN

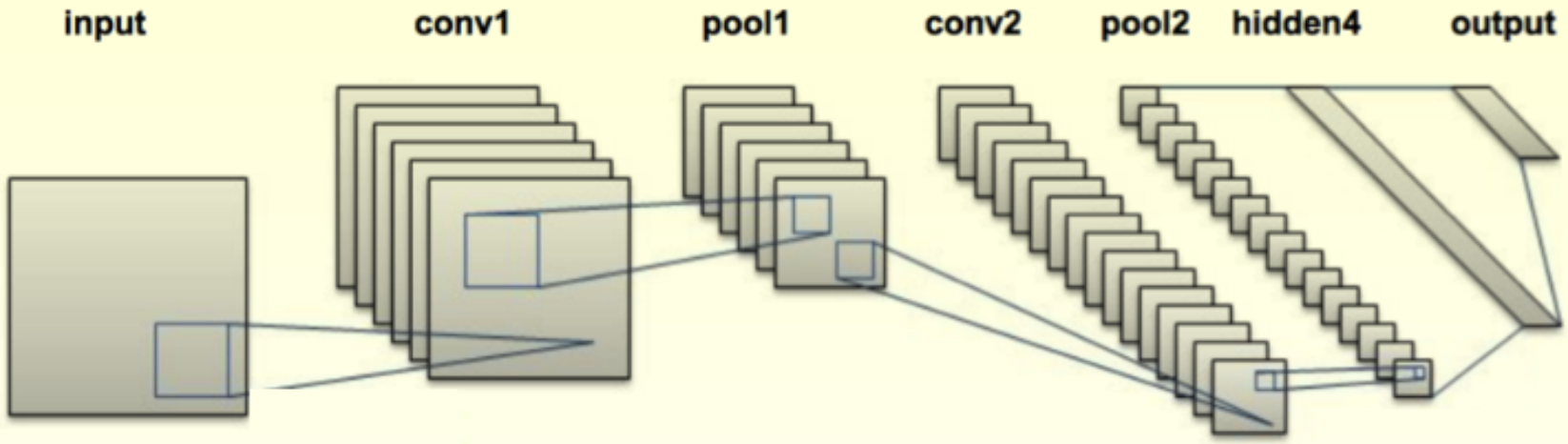
Arquitectura de una red neuronal CNN

Capas de una CNN. Eliminación de Ruido

Arquitecturas de Redes Conocidas

Transferencia de Aprendizaje

LetNet



Capas:

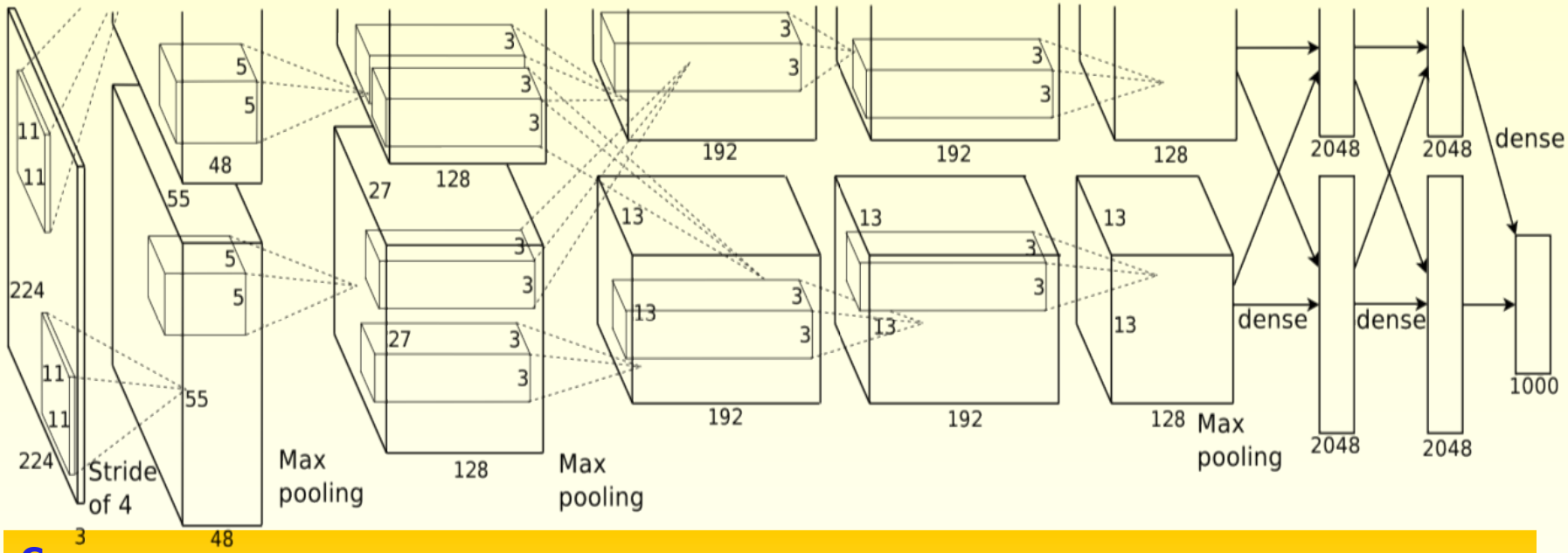
1.- Dos repeticiones: convolucion-activación-pooling

2.- Dos repeticiones: una capa completamente conectada + activación

3.-Clasificación Softmax

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{i=1}^{K} e^{z_i}}$$

AlexNet



Capas:

Capa 1.- 96 filtros de 11x11x3, padding=1,stride=4. Respuesta normalizada y max-pooling. Las salida tiene dimension 55x55x96

Capa 2.- 256 filtros de 5x5x48,stride=1. Respuesta normalizada y max-pooling. Las salida tiene dimension 27x27x256

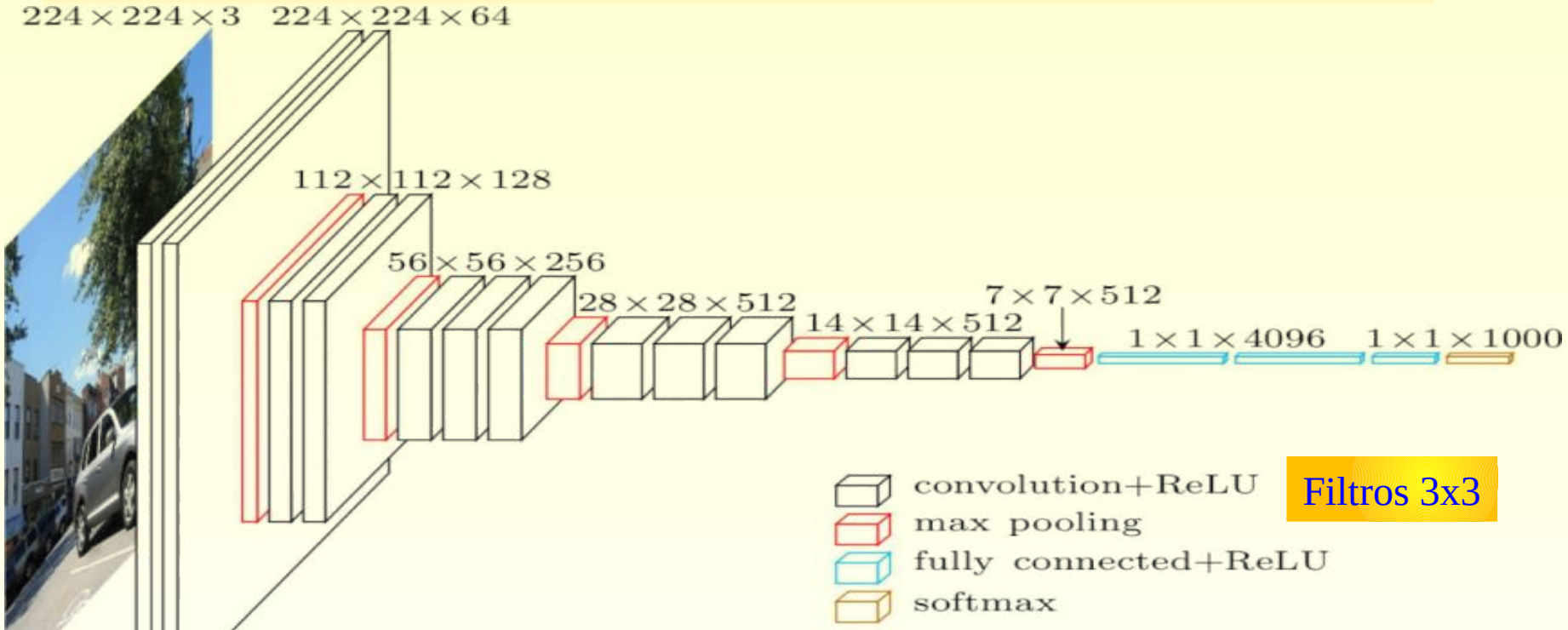
Capa 3.- 384 filtros de 3x3x128. La salida tiene dimensión 13x13x384 Capa 4.- 384 filtros de 3x3x192.La salida tiene dimensión 13x13x384

Capa 5.- 256 filtros de 3x3x192.La salida tiene dimensión 13x13x256

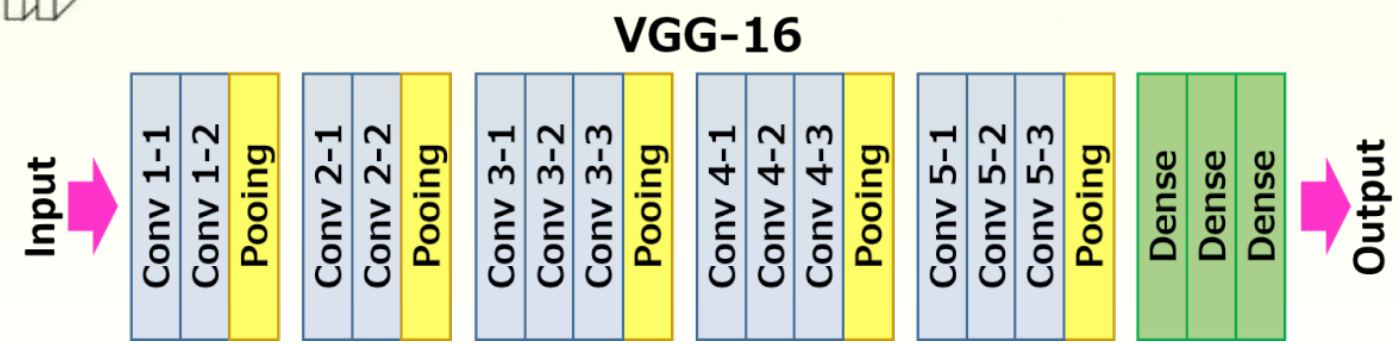
Capa 6,7: Completamente conectada de 4096 neuronas.

Capa 8: Softmax con 1000 clases.

VGG-16



Filtros 3x3



Inception.-
Premisas.

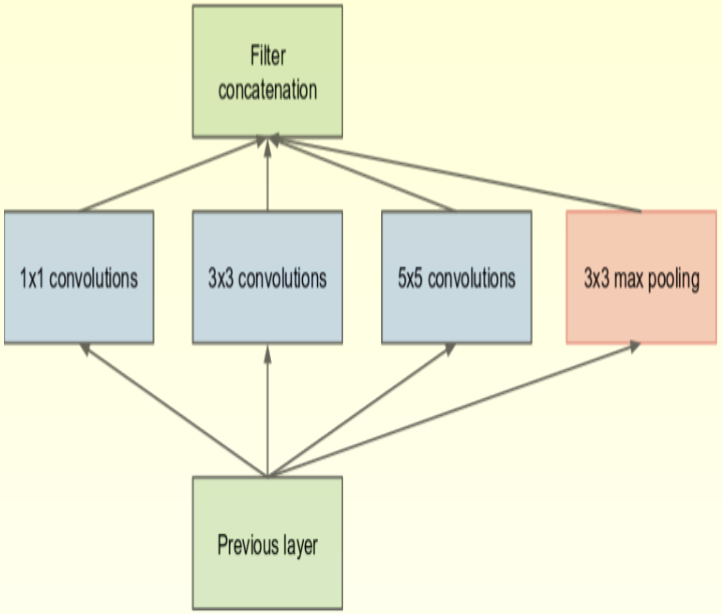
- Características importantes en la imagen puede ser de diferente tamaño.
- Elegir un tamaño fijo y correcto de los filtros es muy difícil. Un tamaño grande sirve para información distribuida más globalmente. Y tamaños pequeños del filtro se usa para información más local.
- Redes con muchas capas produce sobre ajuste. A la vez que la modificación del gradiente a través de la red es difícil (llegando a anularse en puntos intermedios).

Inception.-
Solución.

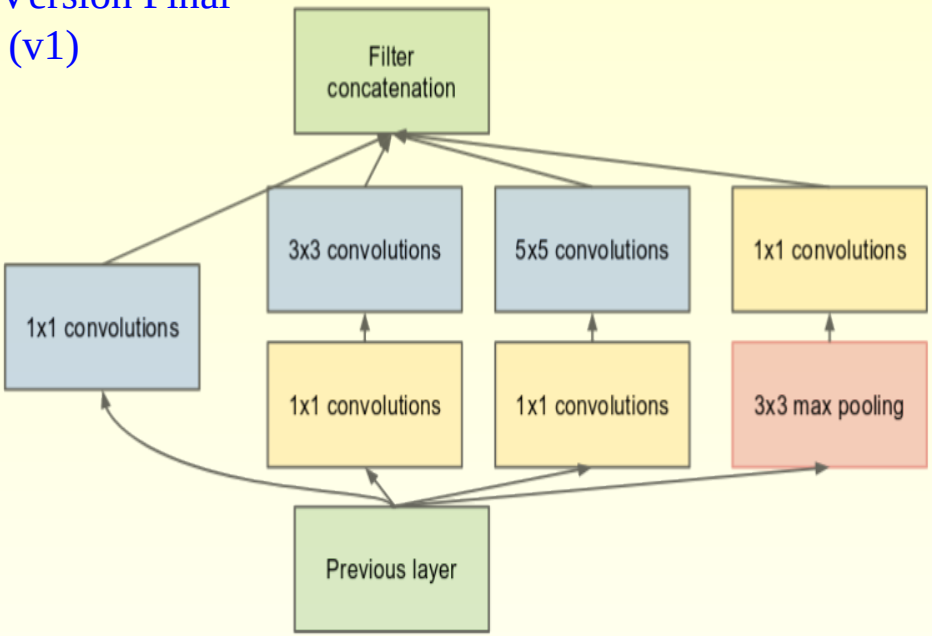
- Tener filtros con diferentes tamaños que operan sobre los mismos datos de entrada.
- La red en vez de ser muy profunda es más ancha .

Inception.-

Versión Inicial



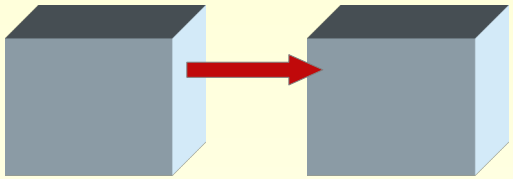
Versión Final (v1)



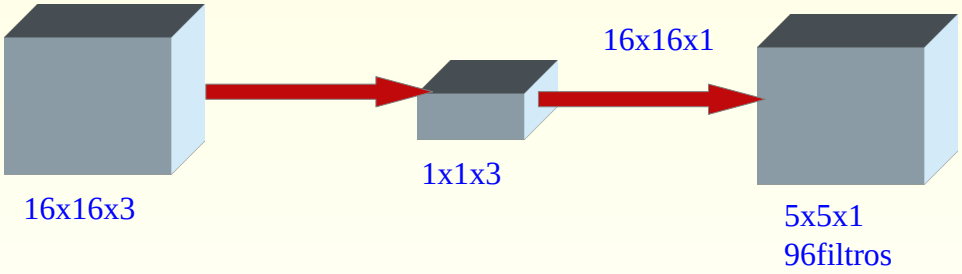
Objetivo con los filtros 1x1: Limitar el número de canales de entrada a las convoluciones 3x3 y 5x5.

Inception.-

Objetivo con los filtros 1x1: Limitar el número de canales de entrada a las convoluciones 3x3 y 5x5.

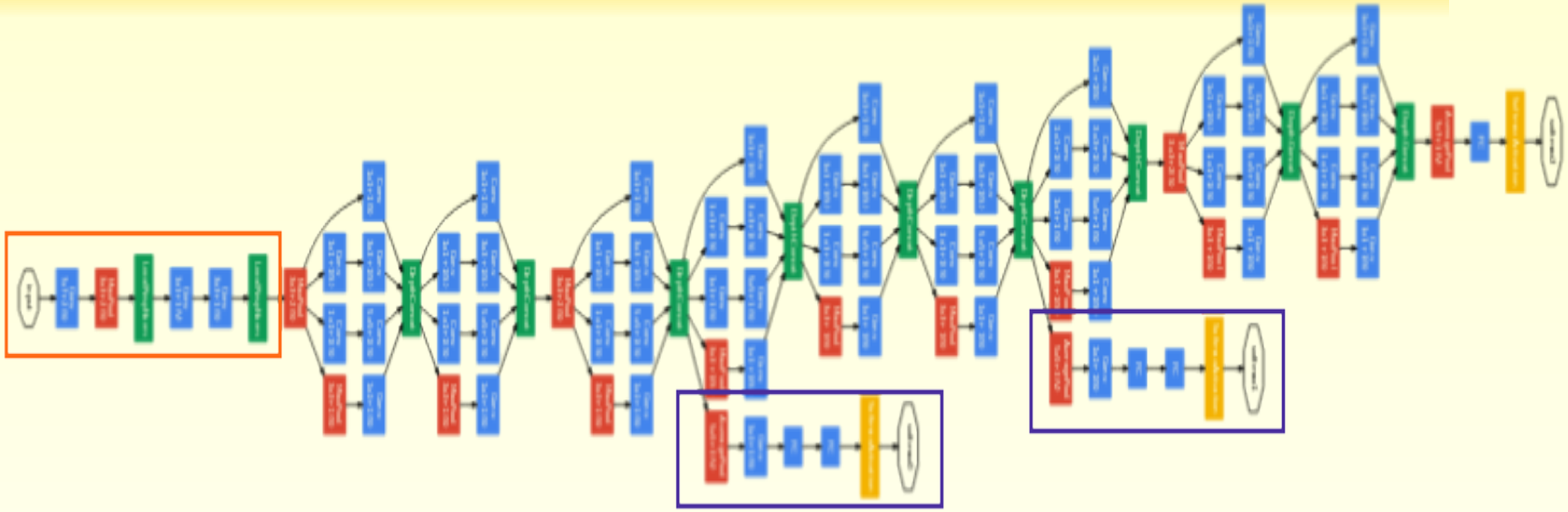


Operaciones realizadas:
96x5x5x3x16x16=1.843.200



Operaciones realizadas:
16x16x3+
16x16x5x5x96=615.168

GoogleNet (Inception v1)



Comentarios:

9 módulos inception. 27 capas de profundidad. Usa average-pooling al final de cada modulo inception.

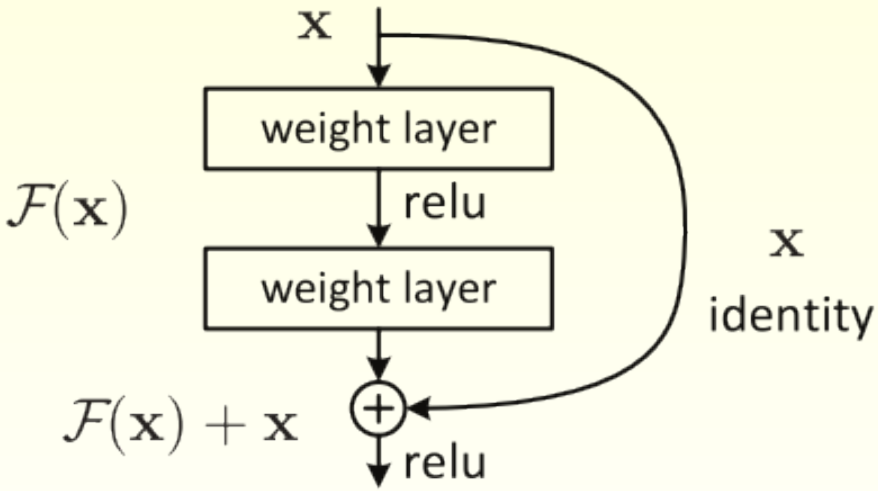
Para prevenir que el gradiente se anule, se introduce dos clasificadores auxiliares (las cajas mradas). Estos clasificadores auxiliares aplican softmax a las salidas de los correspondientes módulos inception y obtienen un error auxiliar (con el mismo conjunto de salida).

La función de pérdida total es una suma ponderada de los errores auxiliares y la pérdida real (al final).

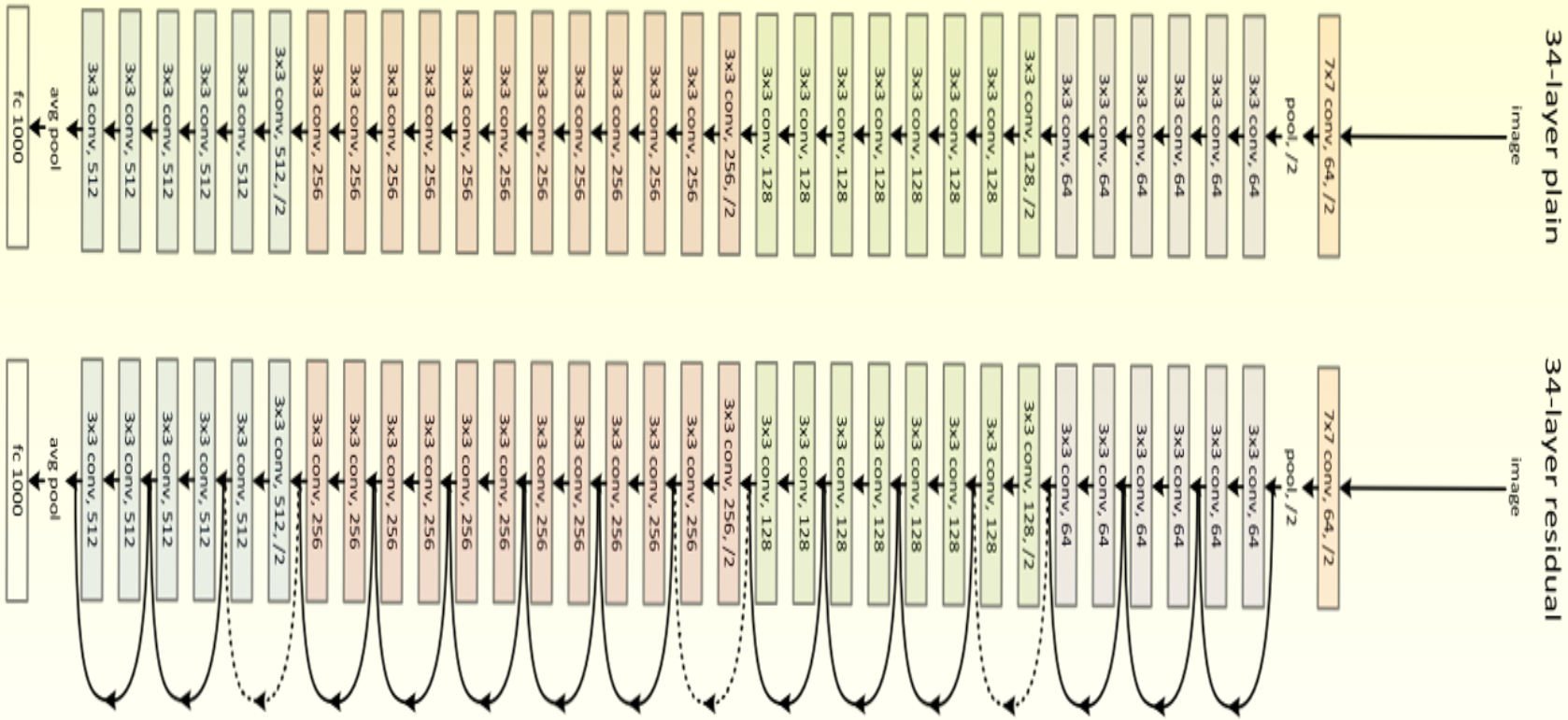
ResNet

Objetivo: Evitar que el gradiente se anule cuando se realiza propagación hacia atrás.

Solución: Bloque residual



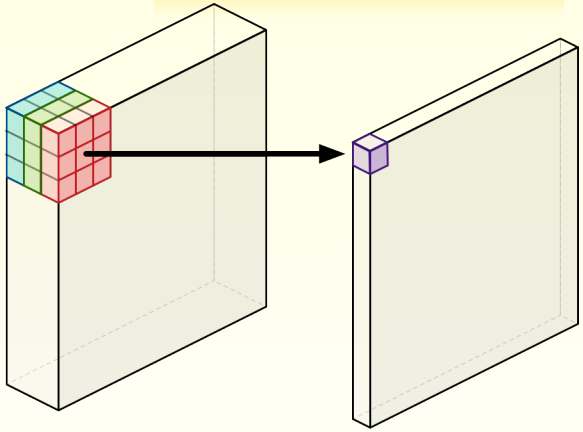
ResNet


34-layer plain
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128
3x3 conv, 256
3x3 conv, 512
3x3 conv, 128, /2
3x3 conv, 256, /2
3x3 conv, 512, /2
avg pool
fc 1000
34-layer residual
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128
3x3 conv, 256
3x3 conv, 512
3x3 conv, 128, /2
3x3 conv, 256, /2
3x3 conv, 512, /2
avg pool
fc 1000

MobileNet. Es una arquitectura muy rápida.

Se basa en realizar Convoluciones Separables de Profundidad

Convolución Normal

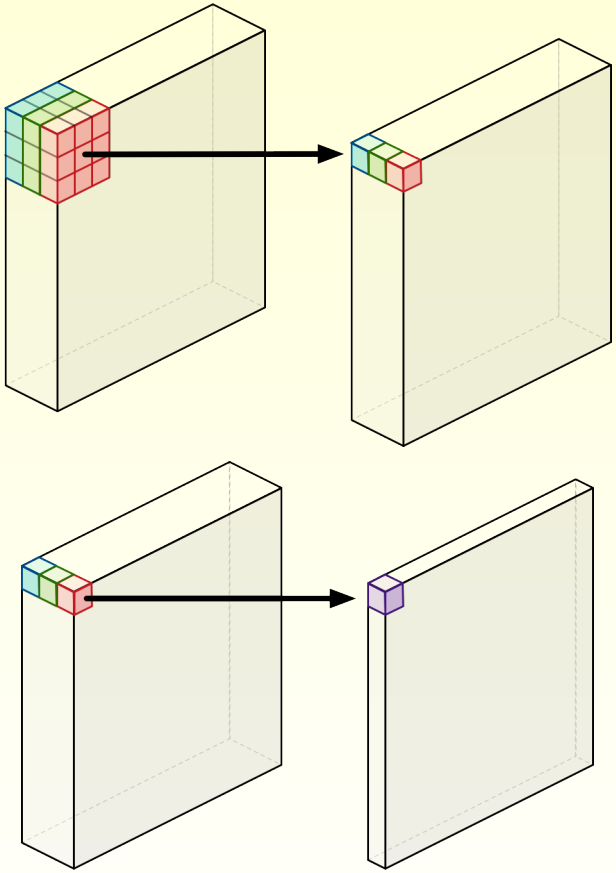


Convolución en Profundidad

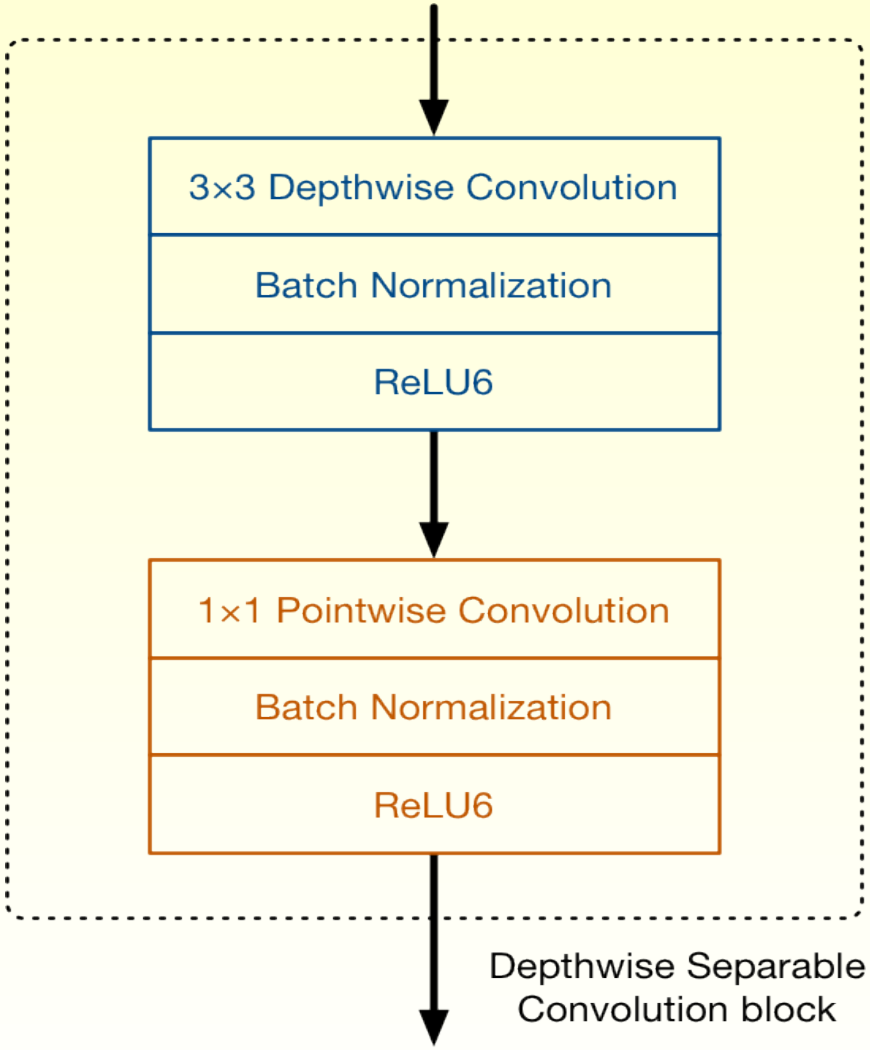
+

Convolución Puntual

Convolución Separable en Profundidad



MobileNet.


3×3 Depthwise Convolution
Batch Normalization
ReLU6
1×1 Pointwise Convolution
Batch Normalization
ReLU6
Depthwise Separable
Convolution block

Deep Learning para visión por Computador. Redes Neuronales de Convonlución CNN

Qué es una CNN

Arquitectura de una red neuronal CNN

Capas de una CNN. Eliminación de Ruido

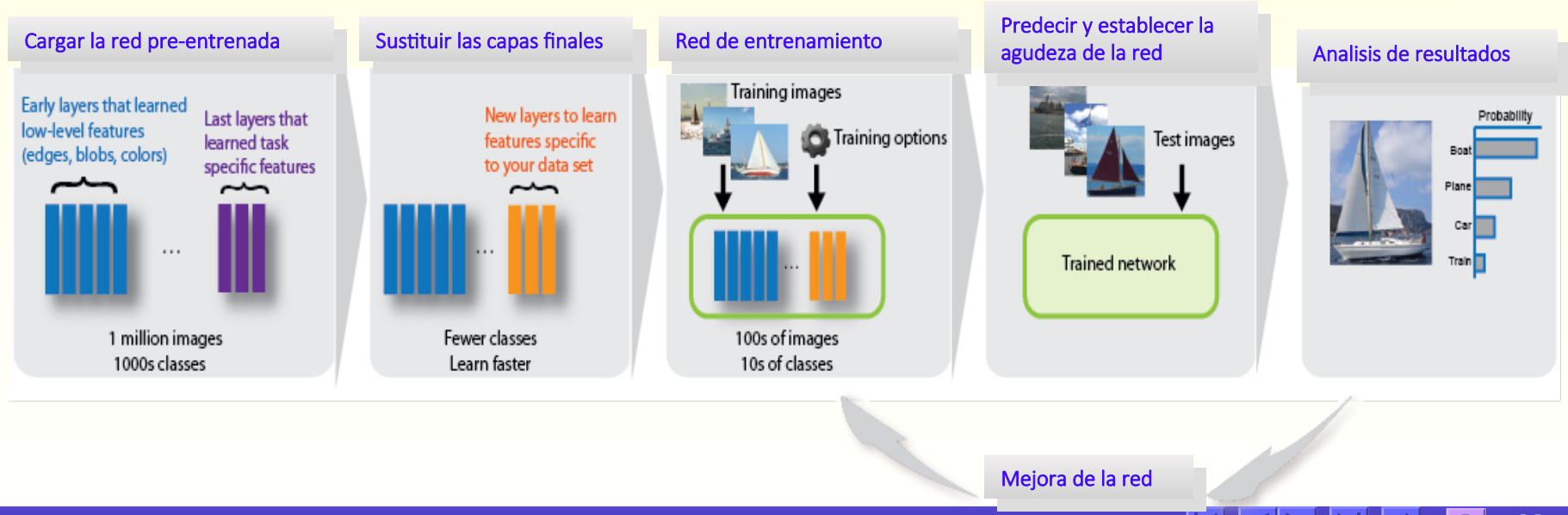
Arquitecturas de Redes Conocidas

Transferencia de Aprendizaje

Objetivo: Usar una red pre-entrenada como punto de inicio para aprender otra tarea.

Como: Modificando una arquitectura de red para la que ya ha sido entrenada. Los pesos que fueron aprendidos para esa arquitectura son usados como punto de inicio.

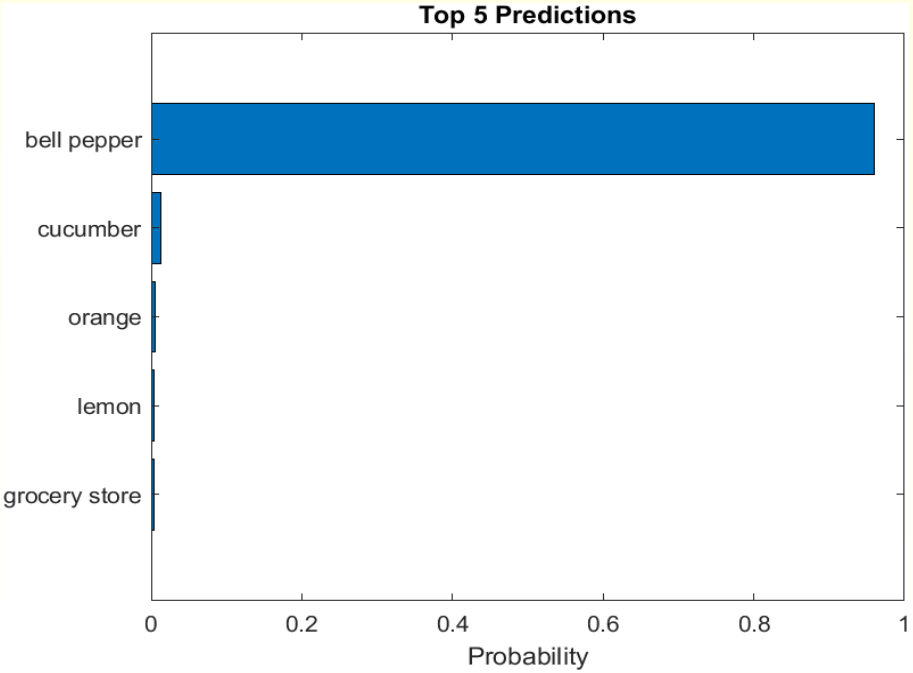
Ventajas:
Aprendizaje mas rápido
El conjunto de entrenamiento es más pequeño.



ImageNet: es una base de datos de imágenes con más de un millón de imágenes, en las que se han clasificado sobre 1000 clases.

Ejemplo en Matlab:

```
net =resnet18();
inputSize = net.Layers(1).InputSize
I=imread('peppers.png');
I = imresize(I,inputSize(1:2));
[label,scores] = classify(net,I);
[~,idx] = sort(scores,'descend');
idx = idx(5:-1:1);
classNamesTop =
net.Layers(end).ClassNames(idx);
scoresTop = scores(idx);
figure
barh(scoresTop)
xlim([0 1])
title('Top 5 Predictions')
xlabel('Probability')
yticklabels(classNamesTop)
```

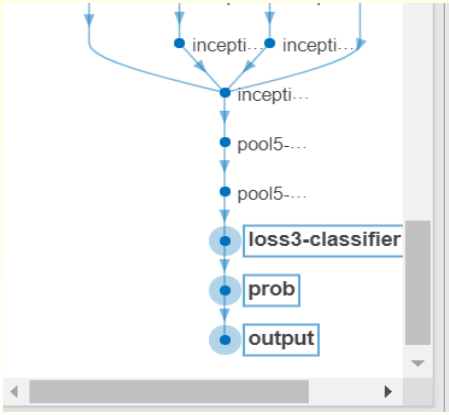


Aspectos que se deben considerar.

1.- La red que se escoge de partida.

```
net =googlenet();
lgraph=layerGraph(net);
analyzeNetwork(lgraph);
```

2.- Cuales son las capas que vais a modificar y como se van a modificar



138	inception_5b-relu... ReLU	ReLU	7×7×128	-
139	inception_5b-output Depth concatenation...	Depth concatenation	7×7×1024	-
140	pool5-7x7_s1 Global average pooling	Global Average Po...	1×1×1024	-
141	pool5-drop_7x7_s1 40% dropout	Dropout	1×1×1024	-
142	loss3-classifier 1000 fully connected...	Fully Connected	1×1×1000	Weights 1000×1024 Bias 1000×1
143	prob softmax	Softmax	1×1×1000	-
144	output crossentropyex with '...	Classification Output	-	-

```
nlayers=size(lgraph.Layers,1);
learnableLayer=lgraph.Layers(nlayers-2);
classLayer=lgraph.Layers(nlayers)
```

Suponiendo que nuestro problema tiene solamente 2 clases de salida

```
newLearnableLayer = fullyConnectedLayer(2, ...
        'Name','new_fc', ...
        'WeightLearnRateFactor',10, ...
        'BiasLearnRateFactor',10);

lgraph=replaceLayer(lgraph,learnableLayer.Name,newLearnableLayer);
%sustituimos la capa de clasificacion
newClassLayer=classificationLayer('Name','new_classoutput');
lgraph= replaceLayer(lgraph,classLayer.Name,newClassLayer);
```

3.- Congelamos determinadas capas poniendo la razón de aprendizaje a 0.

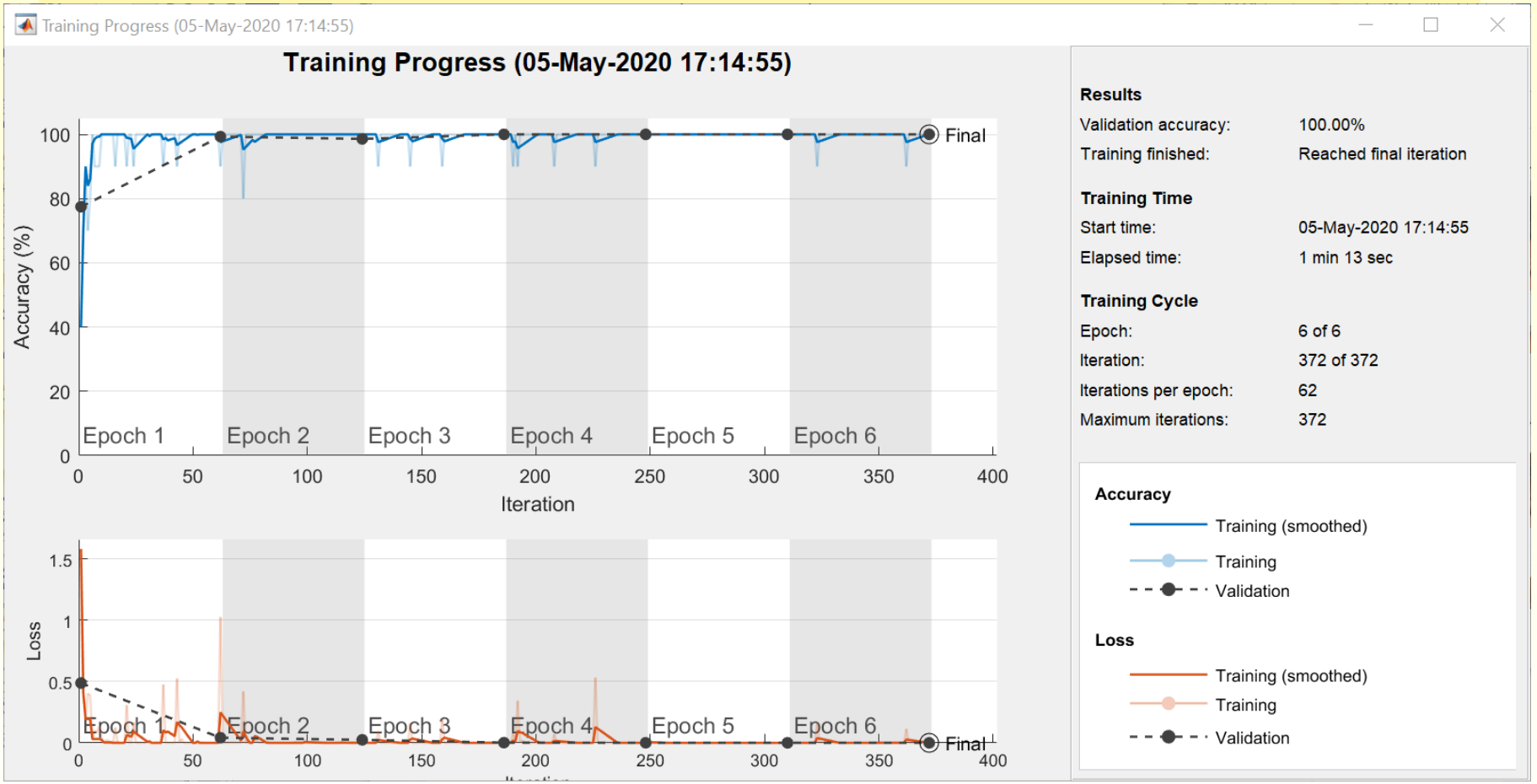
```
layers=lgraph.Layers;
connections=lgraph.Connections;
layers(1:10)=freezeWeights(layers(1:10))
lgraph=createLgraphUsingConnections(layers,connections)
```

4.- Datos sobre los que aplicar la nueva red

5.- Entrenamos la red

```
miniBatchSize=10;
valFrequency=floor(numel(augimdsTrain.Files)/
miniBatchSize)
options = trainingOptions('sgdm', ...
    'MiniBatchSize',miniBatchSize, ...
    'MaxEpochs',6, ...
    'InitialLearnRate',3e-4, ...
    'Shuffle','every-epoch', ...
    'ValidationData',augimdsValidation, ...
    'ValidationFrequency',valFrequency, ...
    'Verbose',false, ...
    'Plots','training-progress');
net = trainNetwork(augimdsTrain,lgraph,options);
```

5.- Entrenamos la red



6.- Aplicamos la red aprendida sobre el conjunto test