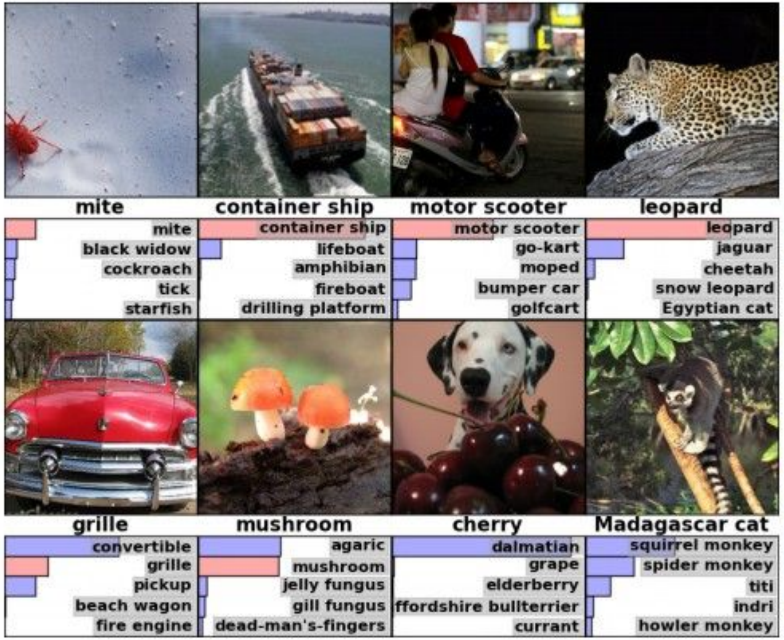
Deep Learning - CNN

ConvNets are everywhere

Classification

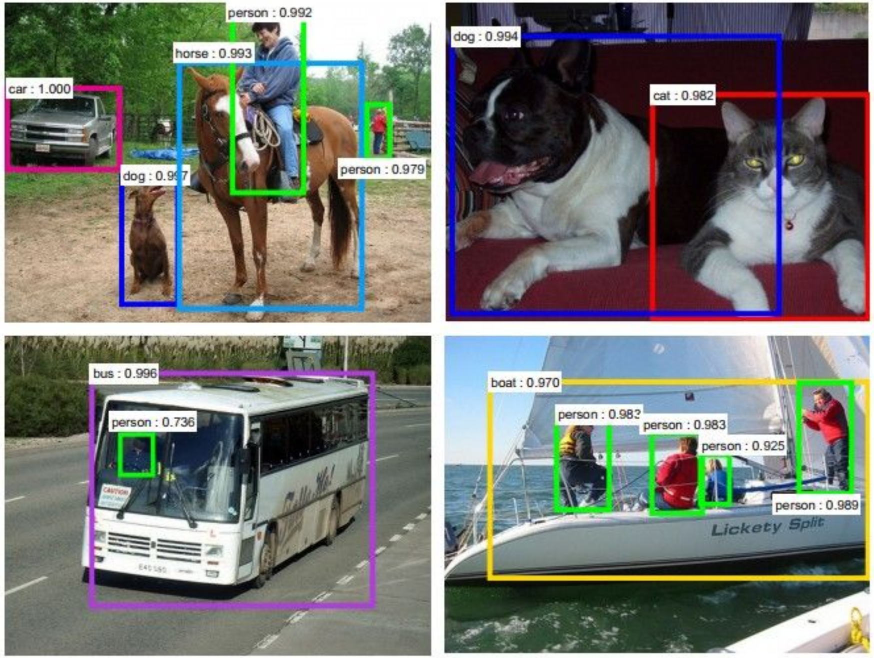


Retrieval

[Krizhevsky 2012]

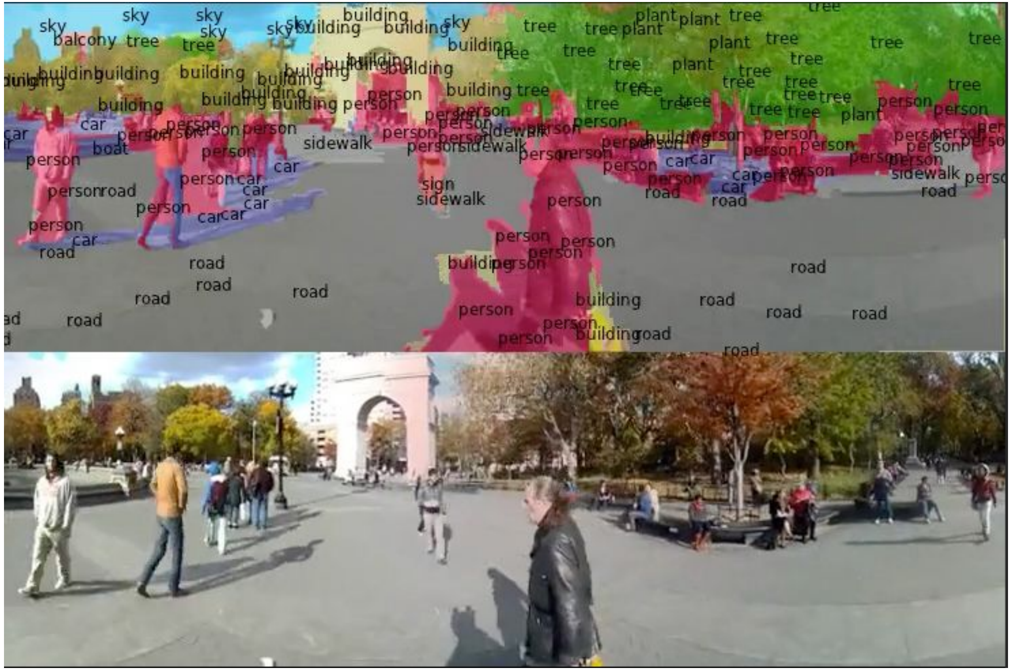
ConvNets are everywhere

Detection



[Faster R-CNN: Ren, He, Girshick, Sun 2015]

Segmentation



[Farabet et al., 2012]

ConvNets are everywhere

self-driving cars

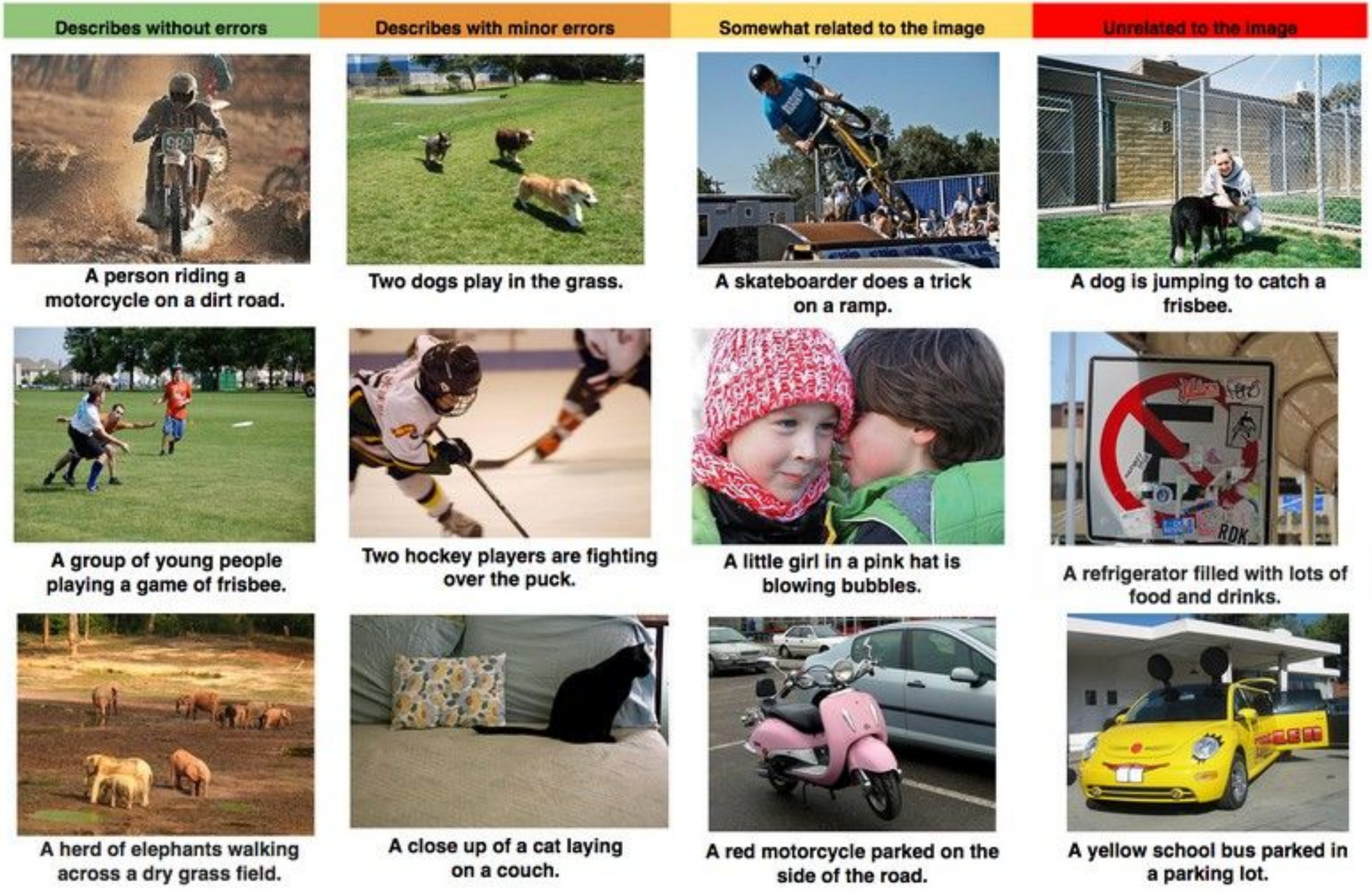
NVIDIA Tegra X1

ConvNets are everywhere

[Toshev, Szegedy 2014]

[Mnih 2013]

Image Captioning



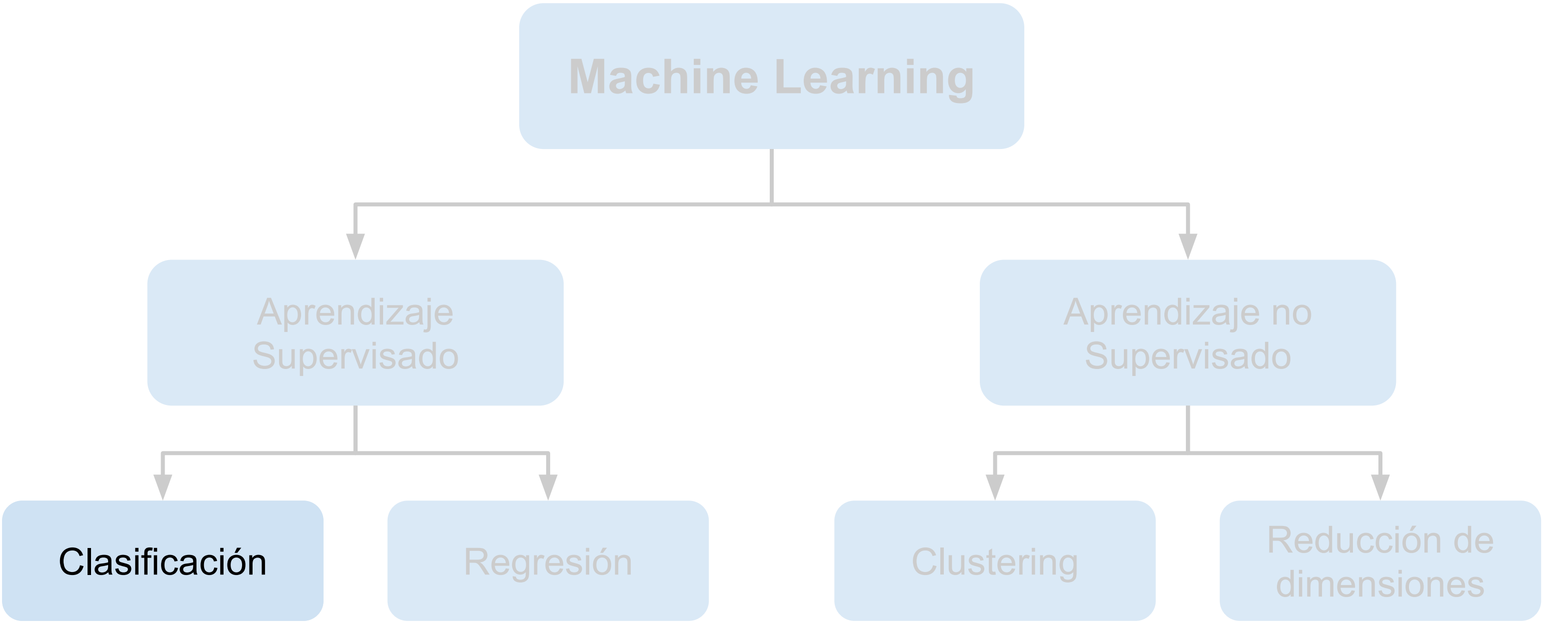
[Vinyals et al., 2015]

Redes Neuronales Convolucionales

1. Clasificando imágenes
2. Capas Densas (Fully connected)
3. Convoluciones
4. Convolutional Neural Network (CNN)
5. Regularización
6. Transfer Learning

Parte 1:

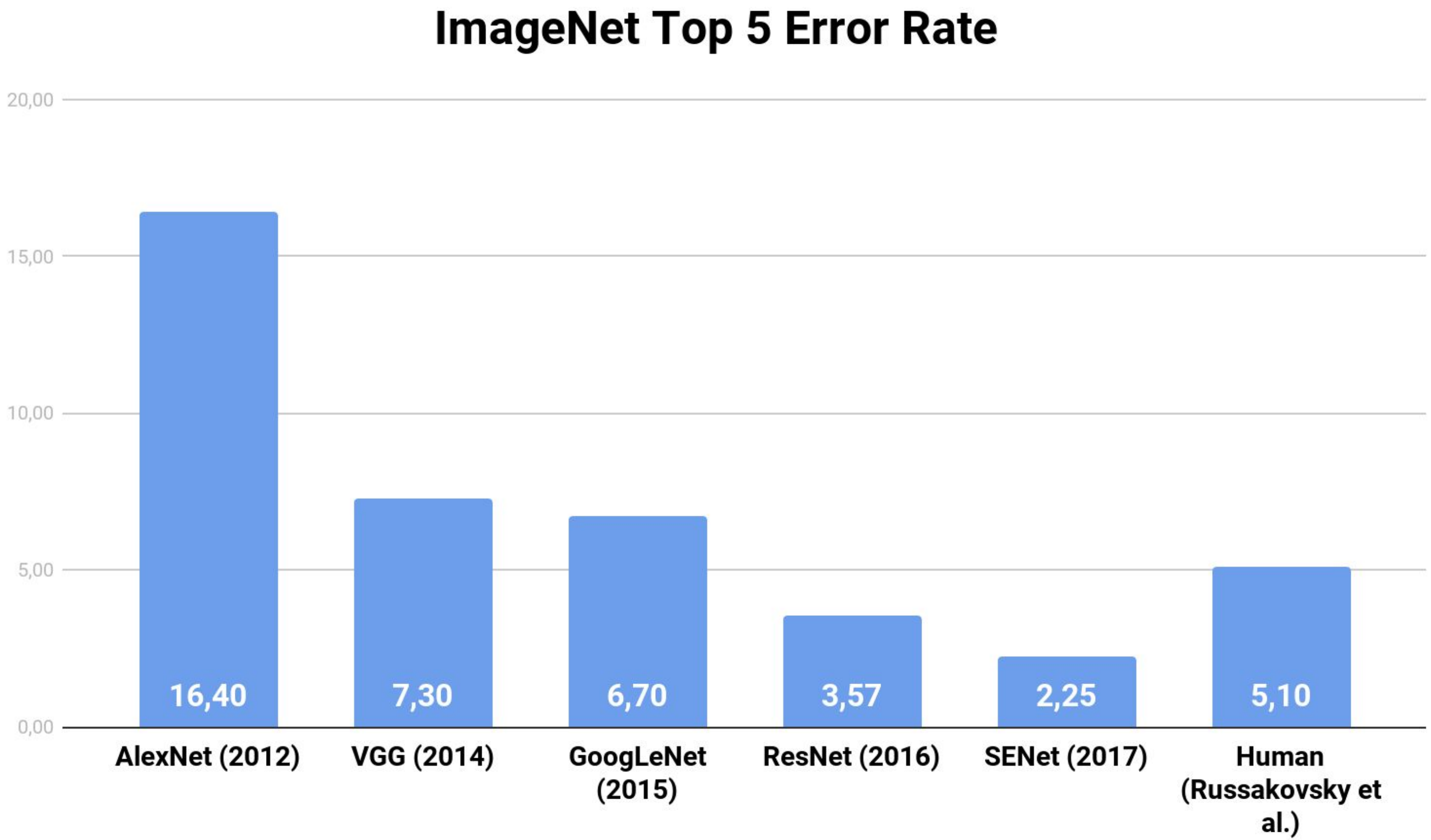
Clasificando imágenes


Machine Learning
Aprendizaje Supervisado
Aprendizaje no Supervisado
Clasificación
Regresión
Clustering
Reducción de dimensiones

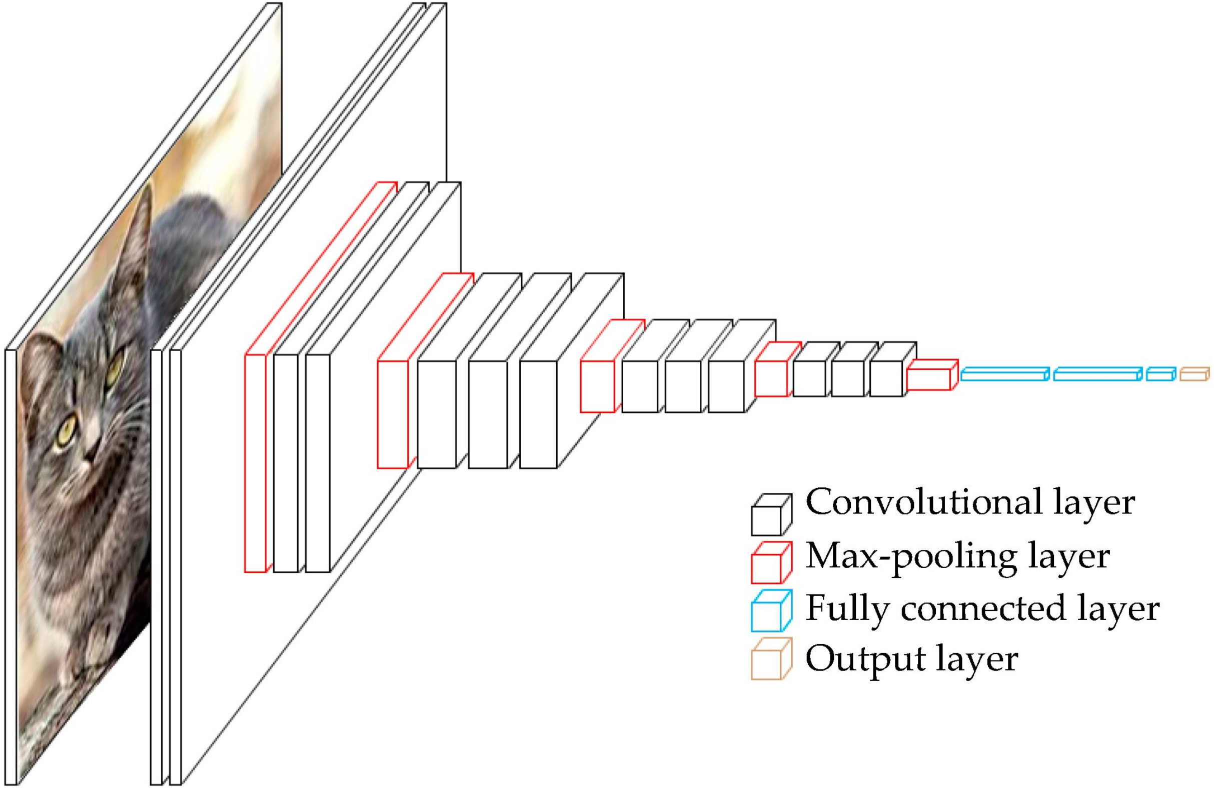
Clasificando imágenes

IMAGENET

- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.


ImageNet Top 5 Error Rate
20,00
15,00
10,00
5,00
0,00
16,40
7,30
6,70
3,57
2,25
5,10
AlexNet (2012)
VGG (2014)
GoogLeNet (2015)
ResNet (2016)
SENet (2017)
Human (Russakovsky et al.)

Clasificando imágenes



Parte 2:

Capas Densas (Fully connected)

???

Cat!

???

Cat!

[32x32x3]
array of numbers 0...1
(3072 numbers total)

10 numbers,
indicating class
score

[0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
dog cat ship .... (one hot encoding)

Parametric approach

[32x32x3]
array of numbers 0...1
(3072 numbers total)
[3072x1]

image weights

f(**x**,**W**)

10 numbers, indicating class scores

[10x1]

Parametric approach: **Linear classifier**

[32x32x3]
array of numbers 0...1
(3072 numbers total)
[3072x1]

image weights

f(**x**,**W**)

$$f(x, W) = Wx$$

10 numbers,
indicating class
scores

[10x1]

Parametric approach: **Linear classifier**

$$\boxed{f(x, W)} = \boxed{W}\boxed{x}$$

3072x1

10x1

?

[32x32x3]
array of numbers 0...1
(3072 numbers total)
[3072x1]

10 numbers,
indicating class
scores

[10x1]

parameters, or “weights”

Parametric approach: **Linear classifier**

$$\boxed{f(x,W)} = \boxed{W}\boxed{x}$$

3072x1

10x1 **10x3072**

[32x32x3]
array of numbers 0...1

10 numbers, indicating class scores

weights

Parametric approach: **Linear classifier**

$$\boxed{f(x,W)} = \boxed{W}\boxed{x} \quad (+b)$$

3072x1

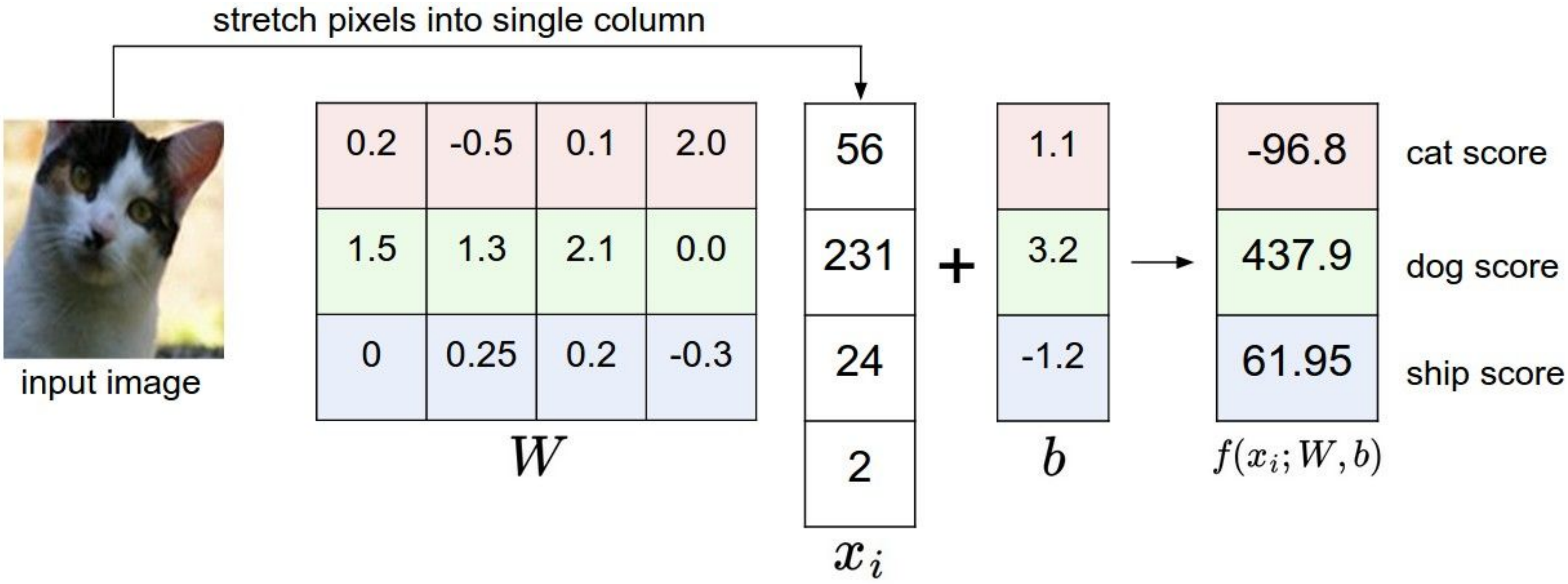
10x1 **10x3072** **10x1**

[32x32x3]
array of numbers 0...1

10 numbers, indicating class scores

weights

Example with an image with 4 pixels, and 3 classes (cat/dog/ship)



Softmax Classifier (Multinomial Logistic Regression)

$$\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

cat	**3.2**
car	5.1
frog	-1.7

unnormalized log probabilities

Softmax Classifier (Multinomial Logistic Regression)

$$\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

unnormalized probabilities

cat	**3.2**		**24.5**
car	5.1	$\xrightarrow{\text{exp}}$	164.0
frog	-1.7		0.18

unnormalized log probabilities

Softmax Classifier (Multinomial Logistic Regression)

$$\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

unnormalized probabilities

cat	**3.2**		**24.5**		**0.13**
car	5.1	$\xrightarrow{\text{exp}}$	164.0	$\xrightarrow{\text{normalize}}$	0.87
frog	-1.7		0.18		0.00

Softmax!

unnormalized log probabilities

probabilities

¿Cómo evaluamos el resultado?

		exp →		normalize →		
cat	**3.2**		**24.5**		**0.13**	**1**
dog	5.1		164.0		0.87	0
ship	-1.7		0.18		0.00	0
					probabilities $q(x)$	real labels $p(x)$

Cross Entropy (Loss Function)

$$L = -\sum p(x) \log q(x)$$

cat	**3.2**	**24.5**	**0.13**	**1**
dog	5.1	164.0	0.87	0
ship	-1.7	0.18	0.00	0

exp

normalize

probabilities
q(x)

real labels
p(x)

Cross Entropy (Loss Function)

$$L = -\sum p(x) \log q(x)$$

		exp		normalize	probabilities $q(x)$	real labels $p(x)$
cat	**3.2**		**24.5**		**0.13**	**1**
dog	5.1		164.0		0.87	0
ship	-1.7		0.18		0.00	0

Cross Entropy (Loss Function)

$$L = -\sum p(x) \log q(x)$$

L = -log(0.13)

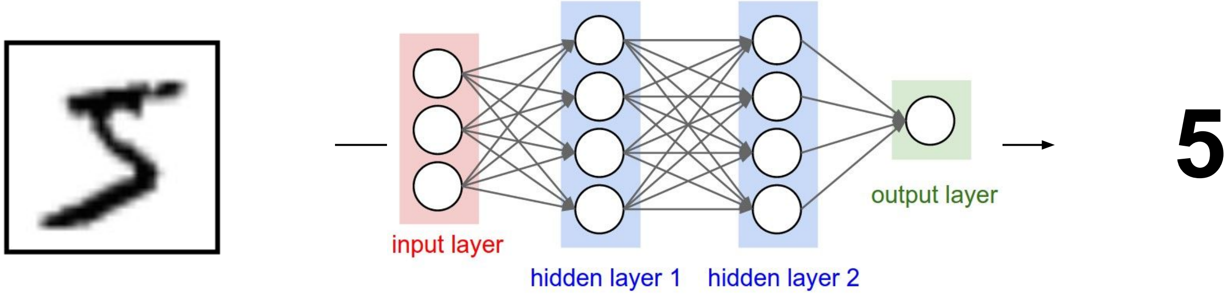
L = **0.89**

0.13	**1**
0.87	0
0.00	0

probabilities q(x)

real labels p(x)

Problem: **MNIST**



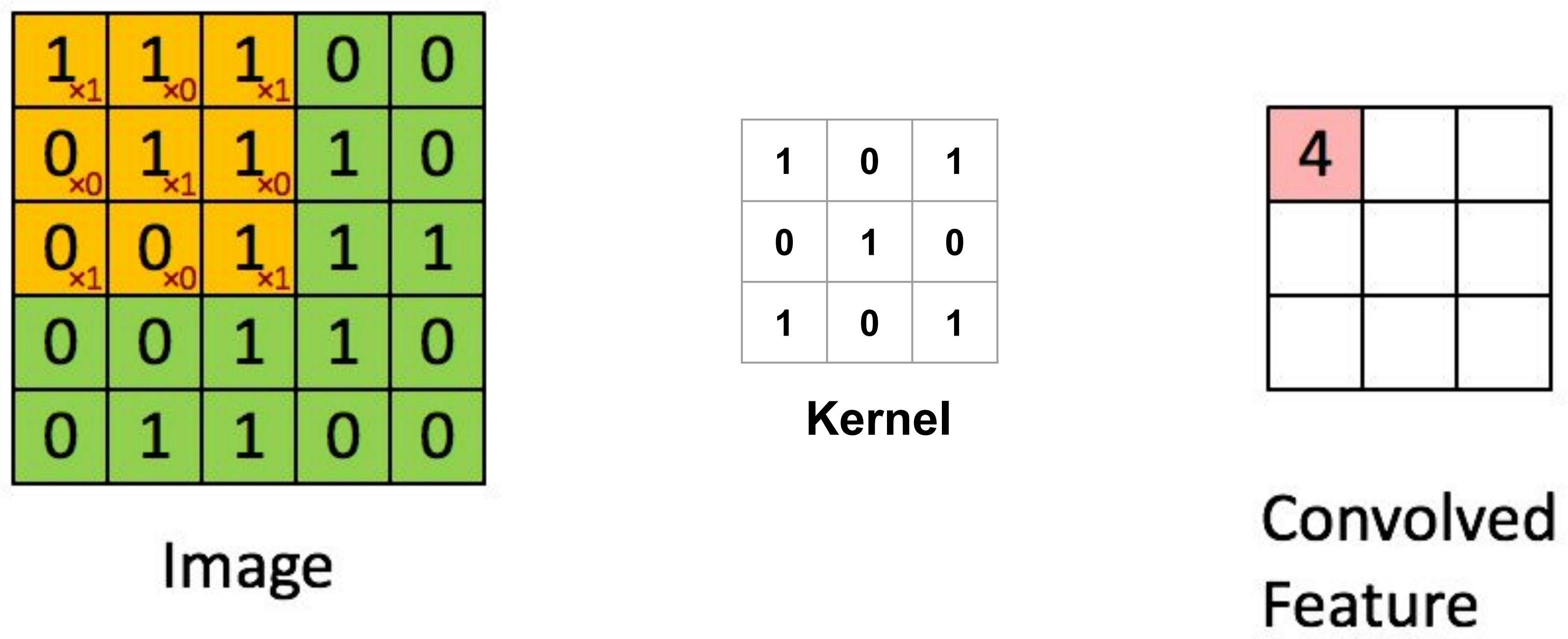

```
model = Sequential()
model.add(Dense(200, activation='relu', input_shape=(784,)))
model.add(Dense(10, activation='softmax'))
```

Solo la 1era capa necesita input shape

Parte 3:

Convoluciones

¿Qué es una convolución?



¿Qué es una convolución?

original

filter (3 x 3)

0	0	0
0	1	0
0	0	0

identity

¿Qué es una convolución?

original

filter (5 x 5)

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

blur

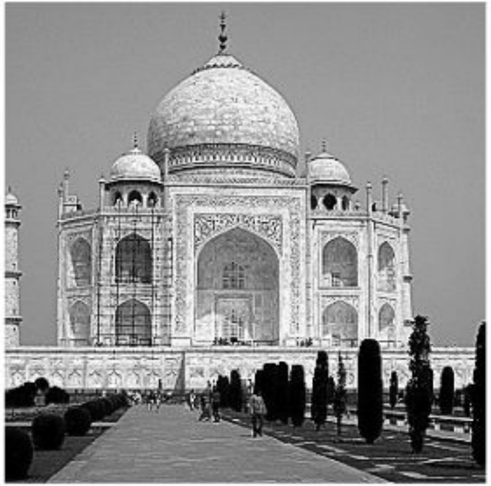
¿Qué es una convolución?

original

filter (5 x 5)

0	0	0	0	0
0	0	-1	0	0
0	-1	5	-1	0
0	0	-1	0	0
0	0	0	0	0

sharpen



¿Qué es una convolución?

original

filter (3 x 3)

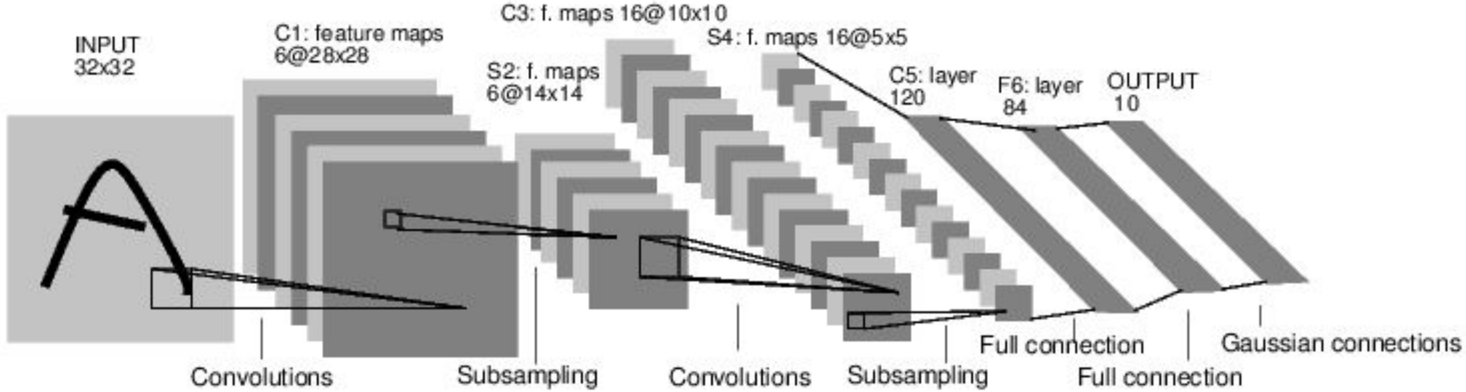
	0	0	0	
	-1	1	0	
	0	0	0	

vertical edge detector

Parte 4:

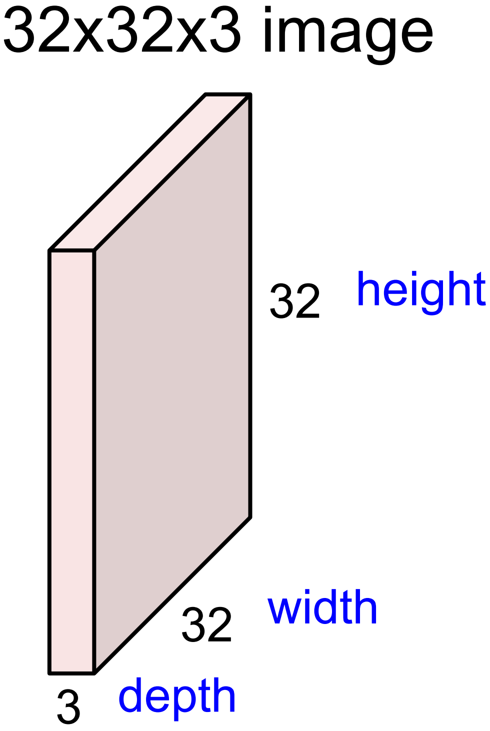
Convolutional Neural Network (CNN)

Convolutional Neural Networks

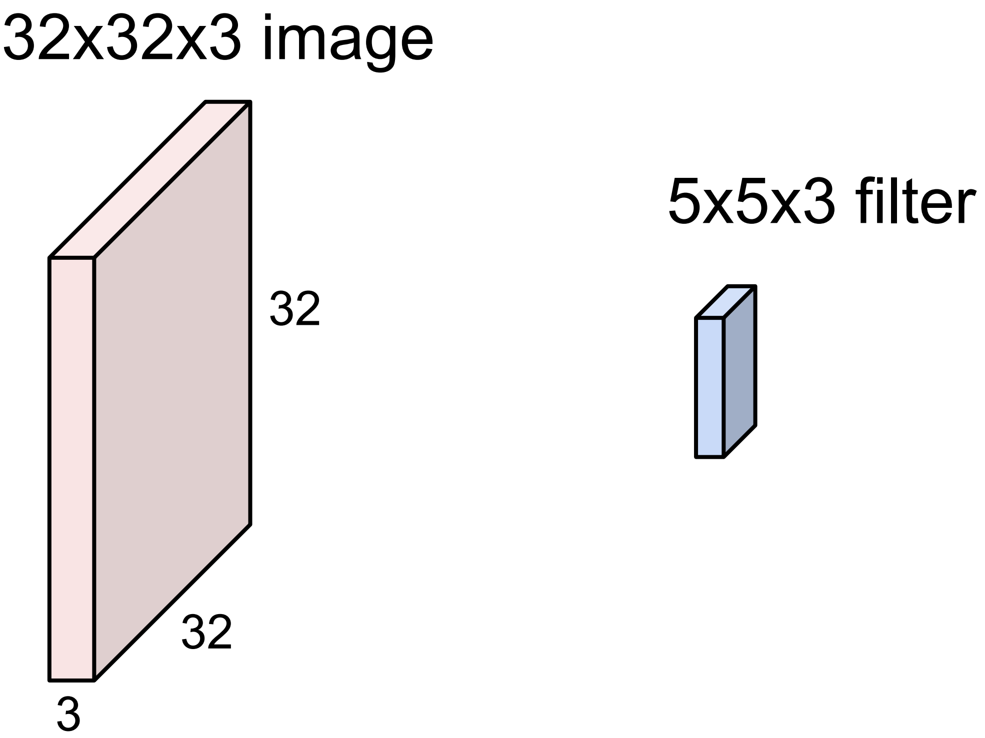


[LeNet-5, LeCun 1980]

Convolution Layer

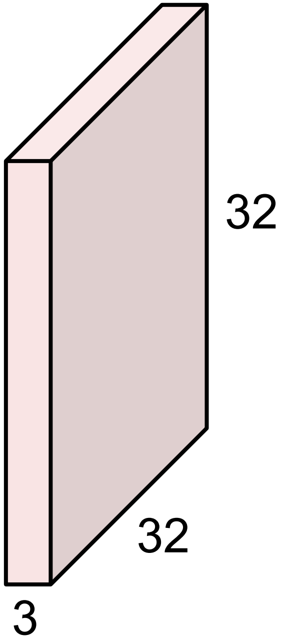


Convolution Layer

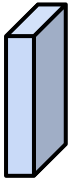


Convolution Layer

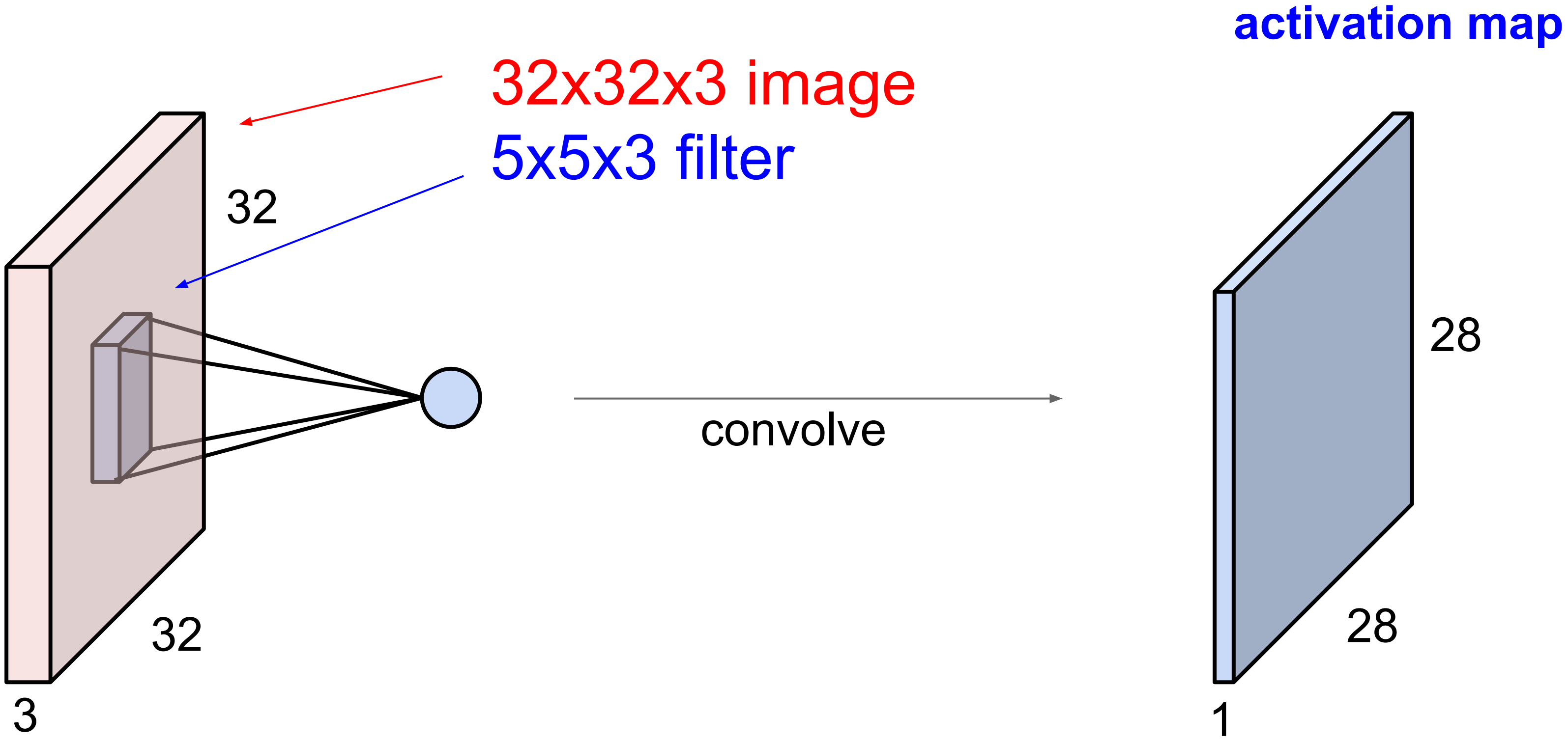
32x32x**3** image



5x5x**3** filter



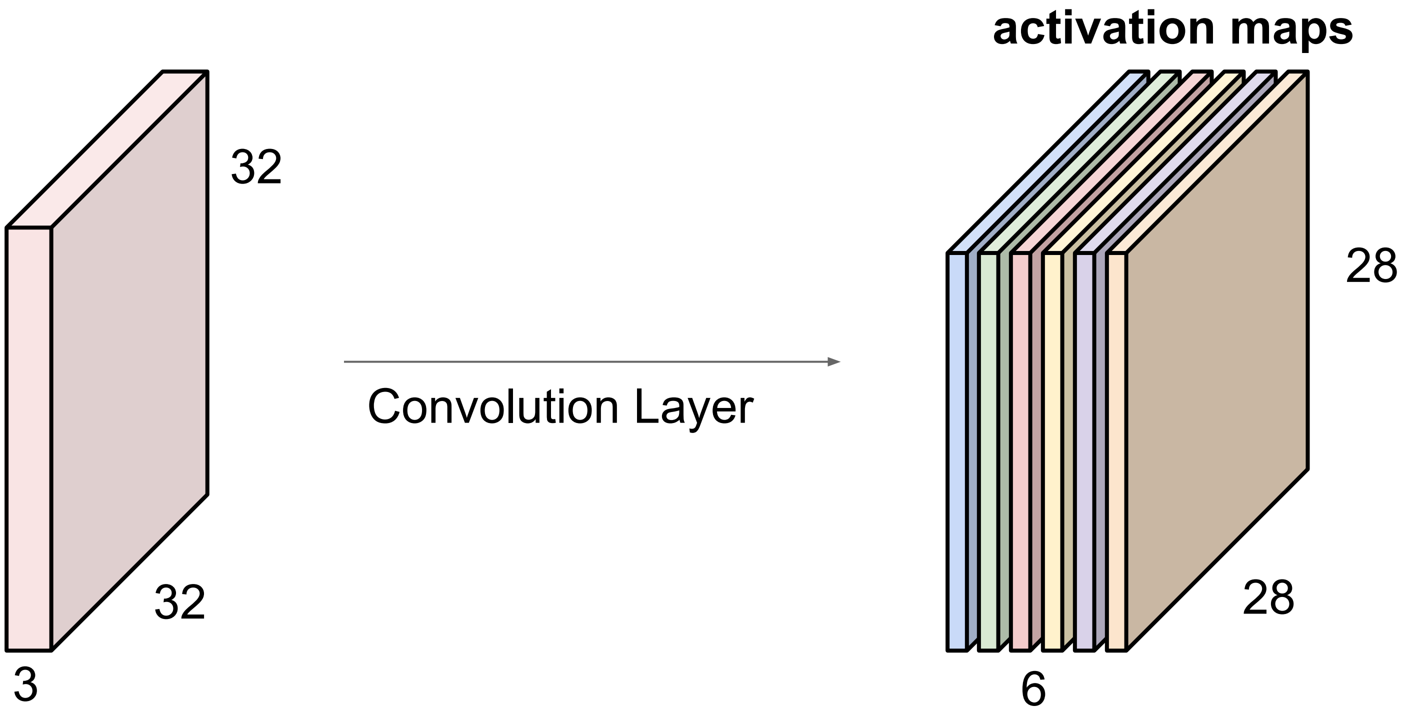
Convolution Layer



Convolution Layer

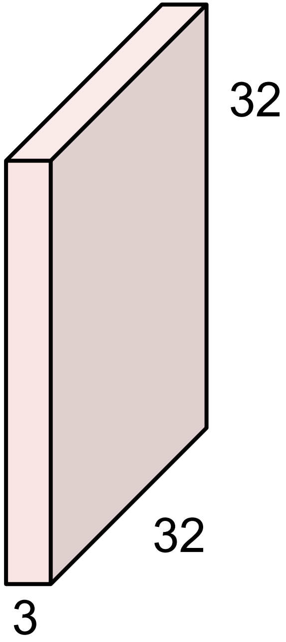
Un segundo kernel

Convolution Layer

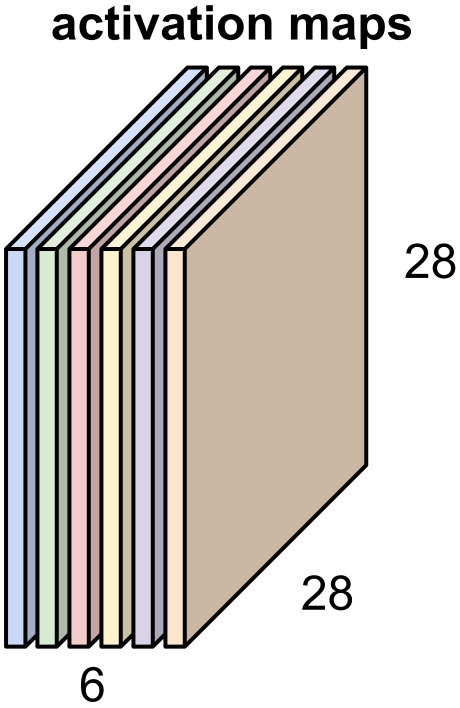


Si tenemos 6 filtros, el resultado tendría la forma: 28x28x6

Convolution Layer

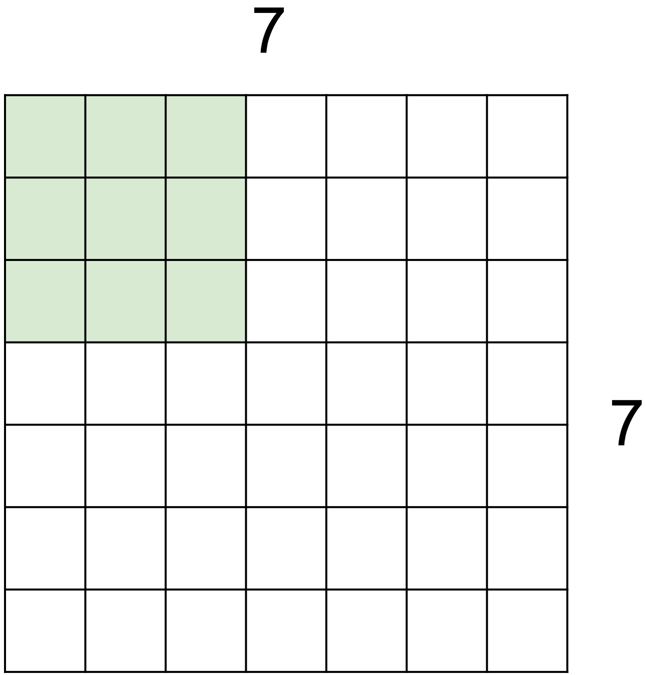


Convolution Layer

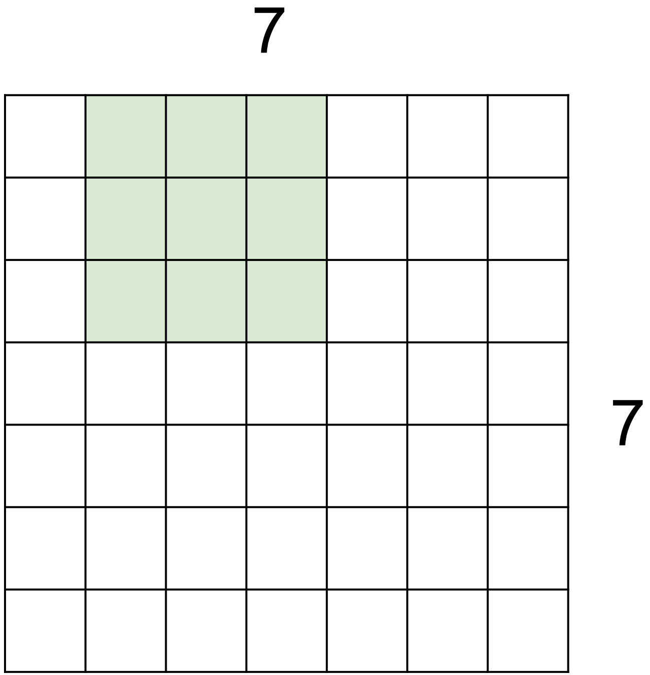


- Kernel size = 5
- # kernels = 6
- padding = 0

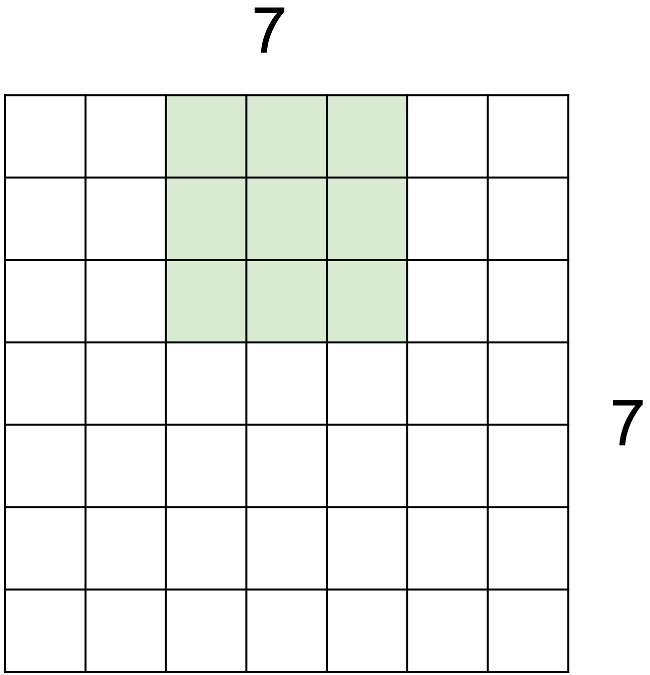
[convolution visualizer]



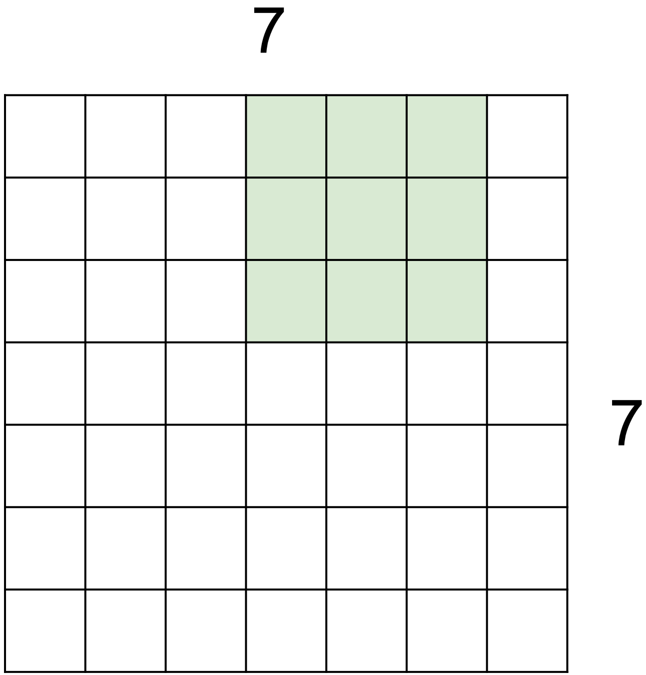
7x7 input
3x3 filter



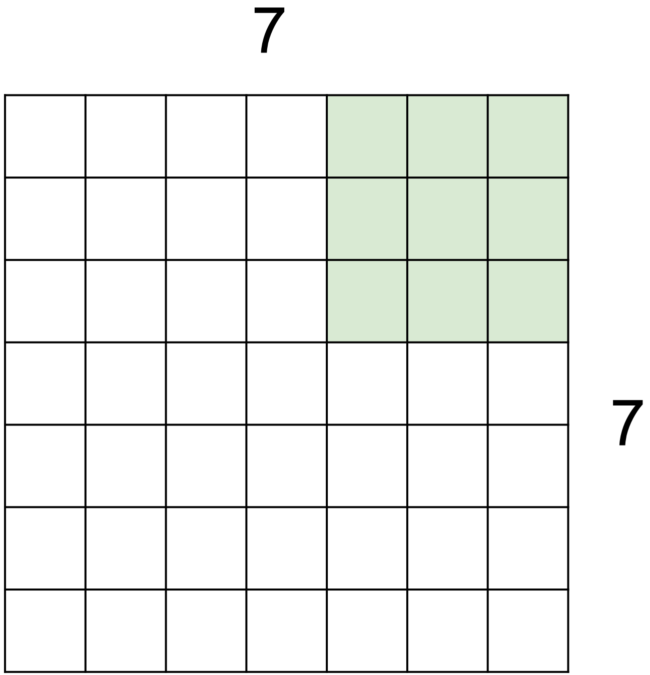
7x7 input
3x3 filter



7x7 input
3x3 filter



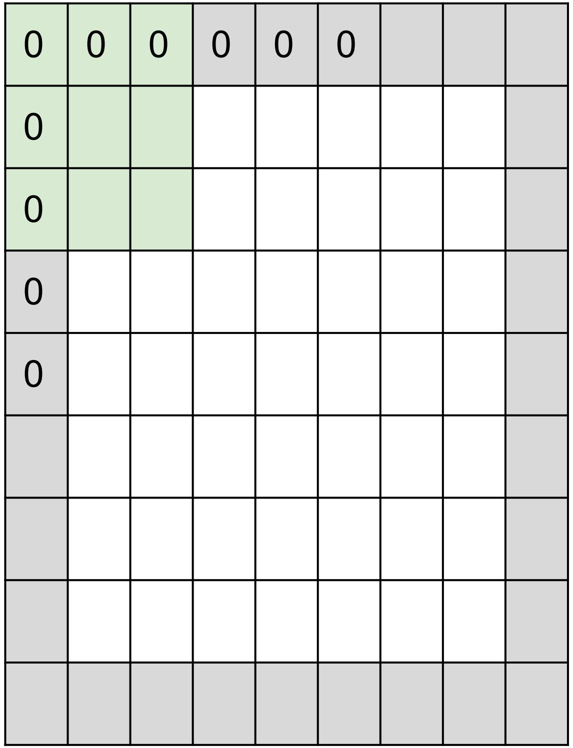
7x7 input
3x3 filter



7x7 input
3x3 filter

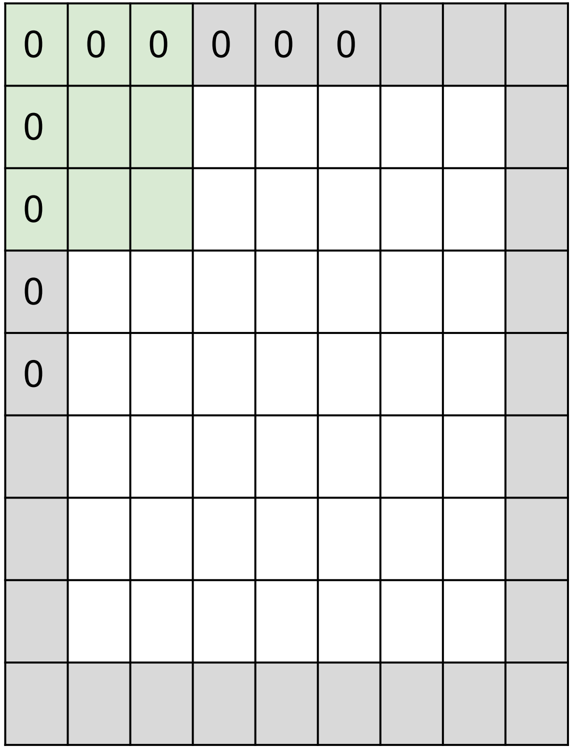
=> 5x5 output

Padding



input 7x7
3x3 filter
padding 1

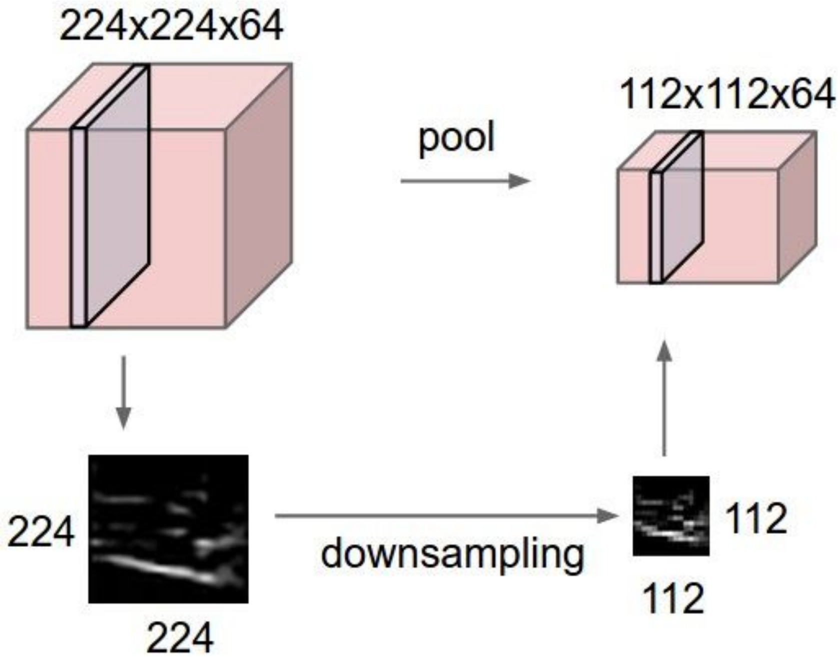
Padding



input 7x7
3x3 filter
padding 1

7x7 output!

Pooling layer



MAX POOLING

Single depth slice

x

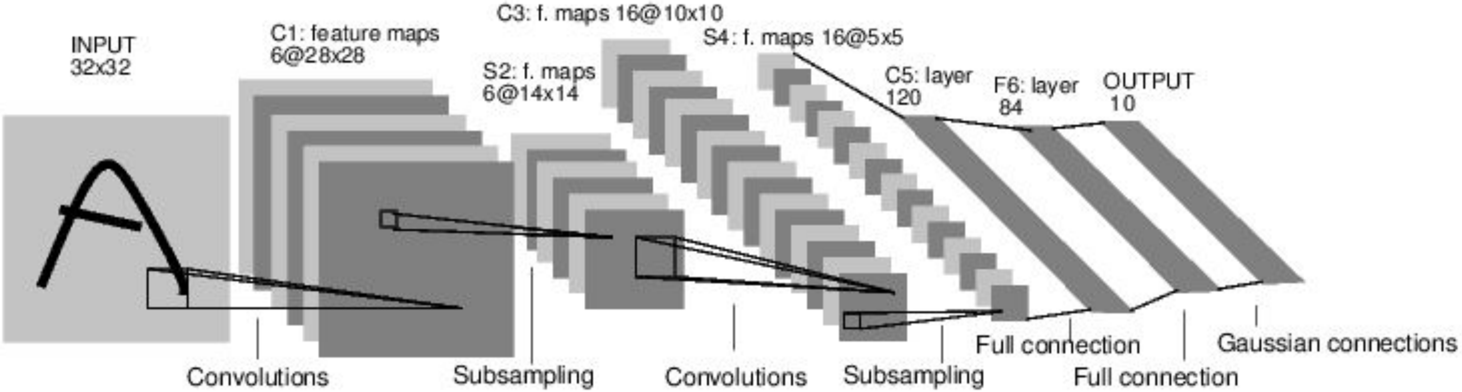
1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

y

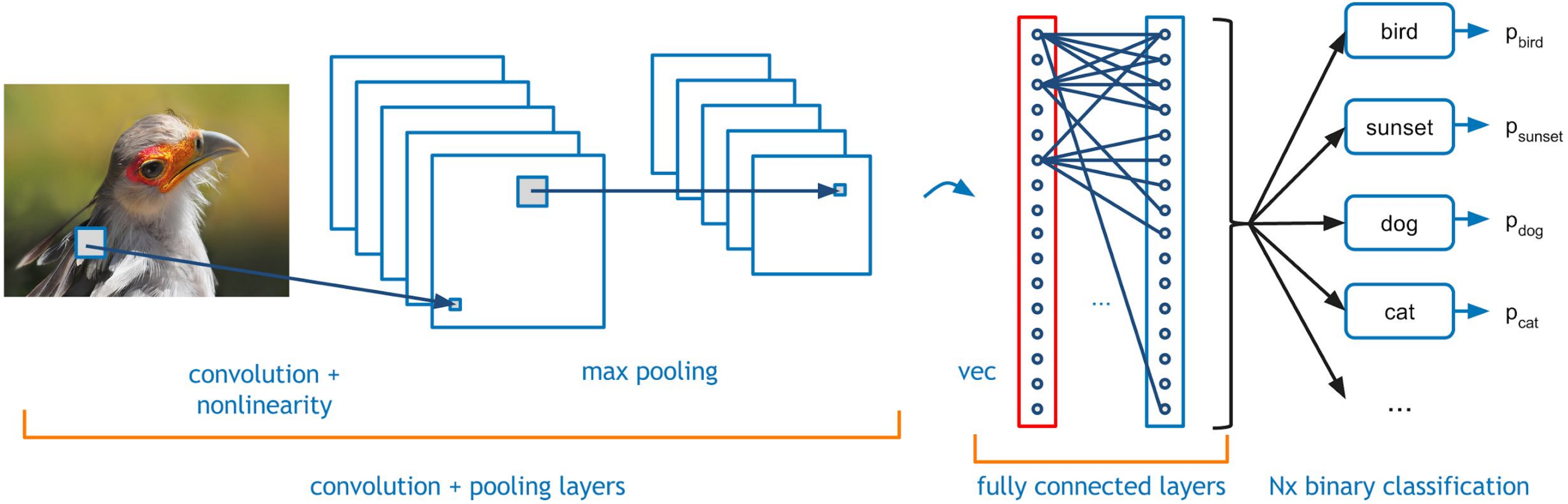
max pool with 2x2 filters
and stride 2

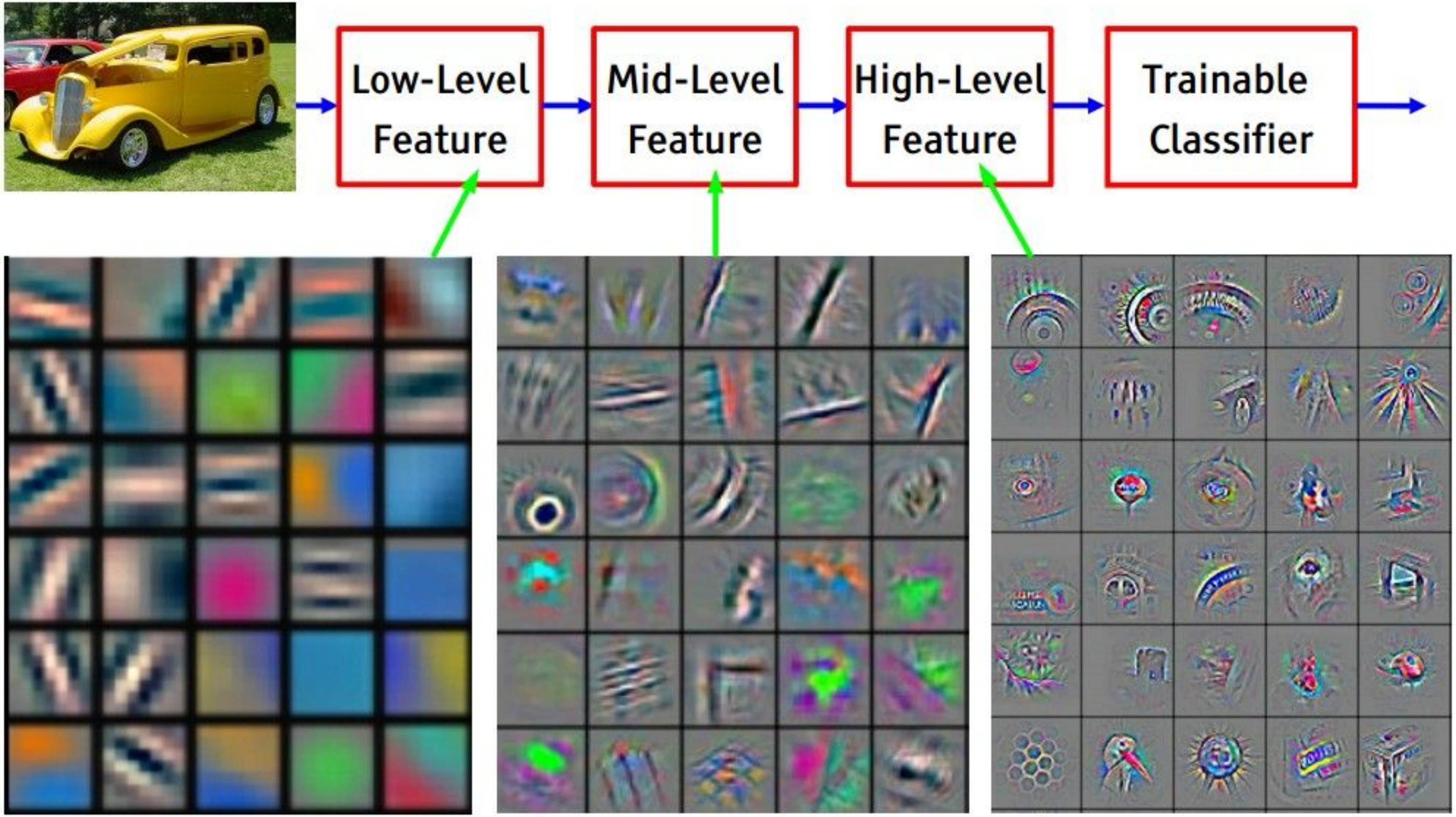
6	8
3	4

Convolutional Neural Networks



[LeNet-5, LeCun 1980]


convolution +
nonlinearity
max pooling
convolution + pooling layers
vec
...
fully connected layers
bird
sunset
dog
cat
...
p_{bird}
p_{sunset}
p_{dog}
p_{cat}
Nx binary classification

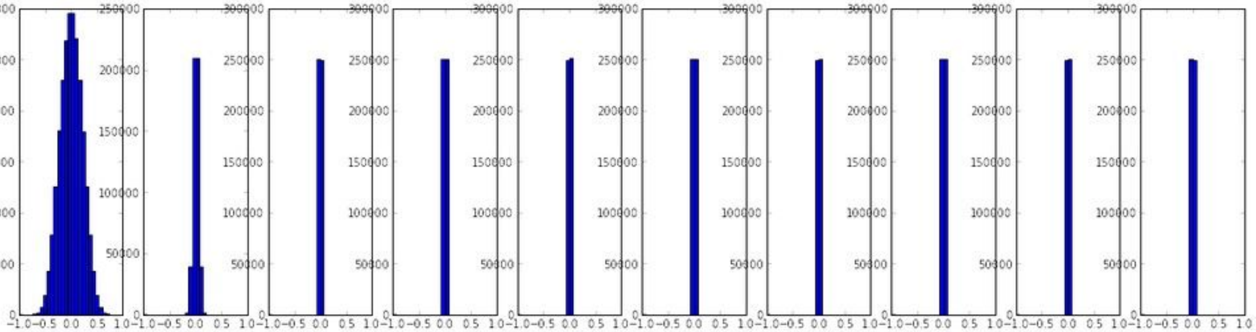


Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Parte 5:

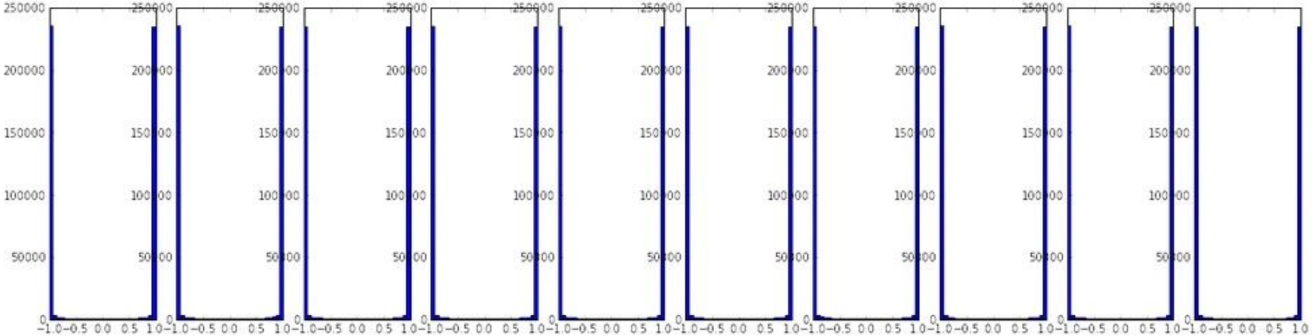
Regularización

Batch Normalization

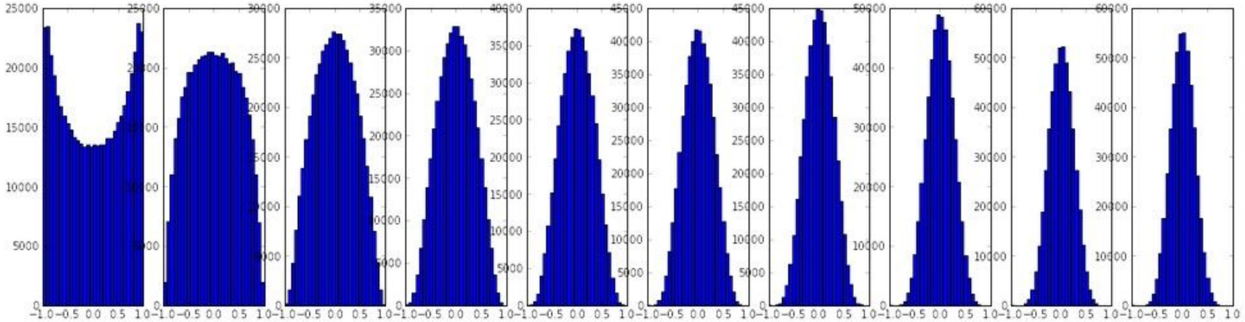


Todas las activaciones = 0

Todas las activaciones = -1 o +1



Batch Normalization

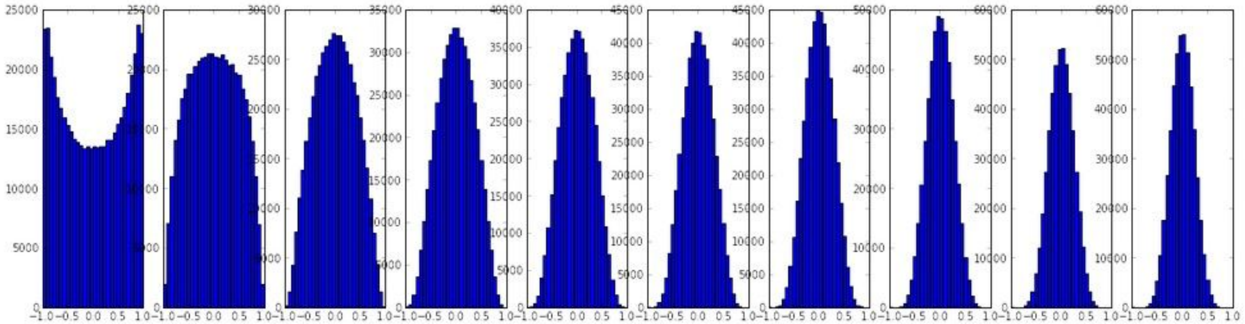


Situación ideal

¿Cómo?

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift
by Sergey Ioffe, Christian Szegedy 2015

Batch Normalization



Situación ideal

¿Cómo? Normalizamos las activaciones de cada capa.

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift
by Sergey Ioffe, Christian Szegedy 2015

Batch Normalization

1. Normalizamos las activaciones:

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

Batch Normalization

1. Normalizamos las activaciones:

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

2. Mitigamos el efecto de la normalización:

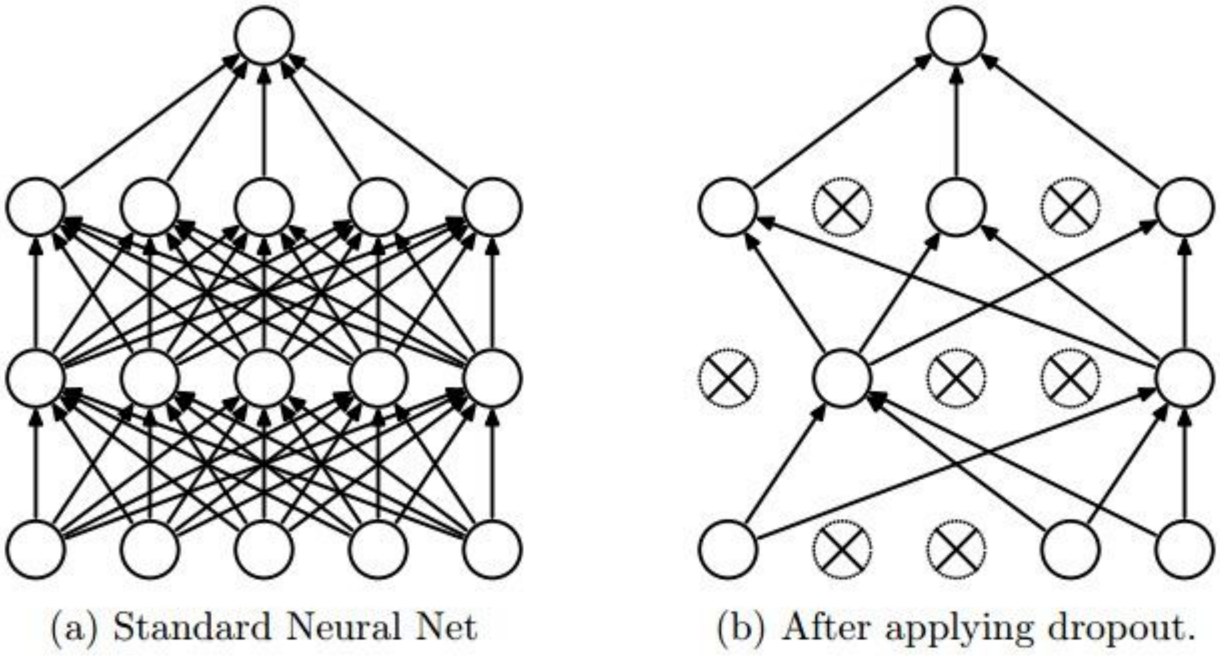
$$y^{(k)} = \gamma^{(k)}\widehat{x}^{(k)} + \beta^{(k)}$$

Es posible que la red aprenda los valores::

$$\gamma^{(k)} = \sqrt{\mathrm{Var}[x^{(k)}]}$$

$$\beta^{(k)} = \mathrm{E}[x^{(k)}]$$

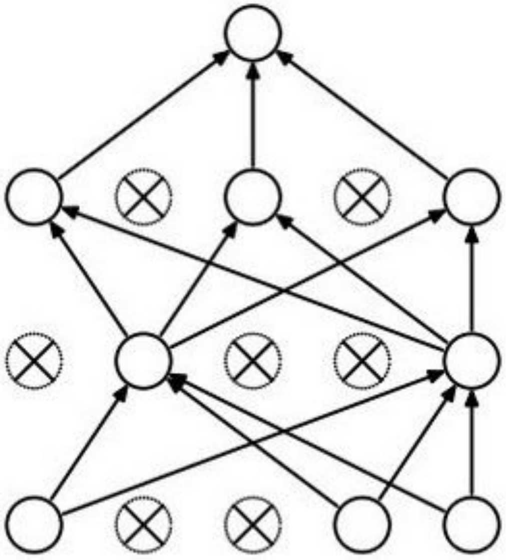
Dropout

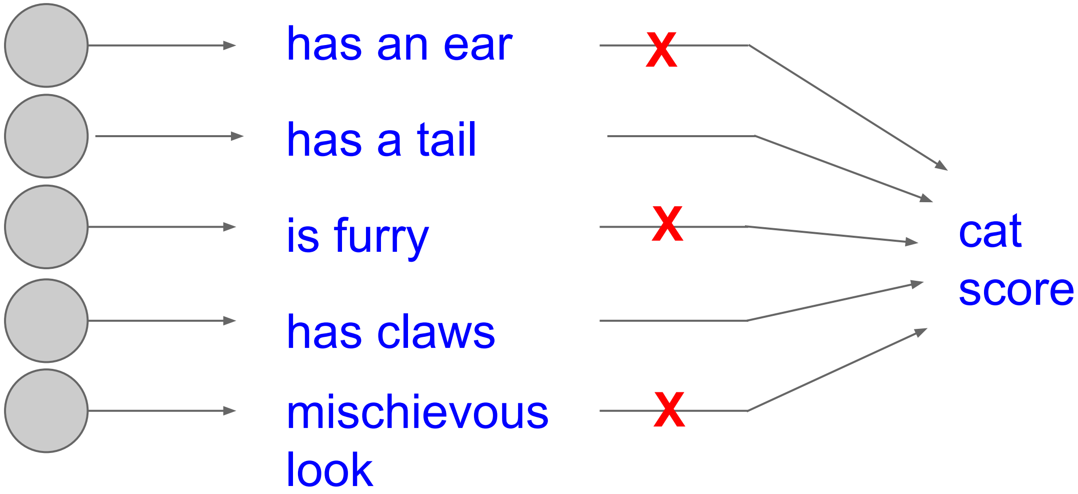


[Srivastava et al., 2014]

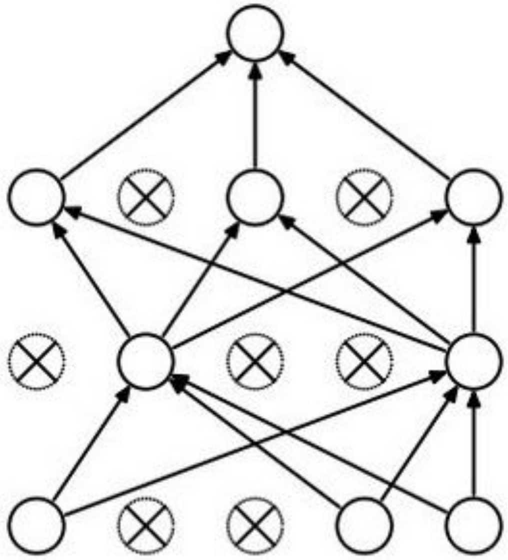
Dropout: A Simple Way to Prevent Neural Networks from Overfitting
by Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov 2014

Dropout




has an ear
X
has a tail
is furry
X
has claws
mischievous
look
X
cat
score

Dropout



Se puede interpretar dropout como un ensemble.

La red va a observar un conjunto distinto de características en cada batch de entrenamiento.

Y para hacer las predicciones se usan todas las características.