

# 4. Redes Convolucionales

Abril 2024

# ① INTRODUCCION

- objetivos : clasificación de imágenes, detección de objetos, transferencia de estilos (combinar dos estilos).
- Retos cada objeto puede ser muy grande.
- una estrategia consiste en considerar las imágenes como una composición de imágenes más simple: bordes verticales, horizontales, etc

# ① CONVOLUTIONES

## Vertical edge detection

$$3 \times 1 + 1 \times 1 + 2 \times 1 + 0 \times 0 + 5 \times 0 + 7 \times 0 + 1 \times -1 + 8 \times -1 + 2 \times -1 = -5$$

3	<u>0</u>	<u>1</u>	<u>2</u>	<u>7</u>	<u>4</u>
1	<u>5</u>	<u>8</u>	<u>9</u>	<u>3</u>	<u>1</u>
2	<u>7</u>	<u>2</u>	<u>5</u>	<u>1</u>	<u>3</u>
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

6x6

"convolution"  
\*

1	0	-1
1	0	-1
1	0	-1

3x3  
filter

0

KERNEL

=

<u>-5</u>	<u>-4</u>	0	<u>8</u>
<u>-10</u>	-2	2	3
0	-2	-4	-7
-3	-2	-3	<u>-16</u>

4x4

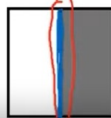
Andrew Ng

- ESTE FILTRO EN PARTICULAR AYUDA A DETECTAR BORDOS VERTICALES :

## Vertical edge detection

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	<u>10</u>	<u>10</u>	<u>0</u>	0	0
10	<u>10</u>	<u>10</u>	<u>0</u>	0	0
10	<u>10</u>	<u>10</u>	<u>0</u>	0	0

6x6



\*

1	0	-1
1	0	-1
1	0	-1

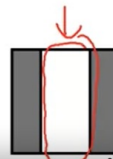
3x3

=

0	30	30	0
0	30	30	0
0	30	30	0
0	<u>30</u>	<u>30</u>	0

4x4

\*



Andrew Ng



- DIFERENCIA DE GRAY DE GRIS A BLANCO - BLANCO A GRIS

## Vertical edge detection examples

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0



0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10



\*

1	0	-1
1	0	-1
1	0	-1



=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0



\*

1	0	-1
1	0	-1
1	0	-1



=

0	-30	-30	0
0	-30	-30	0
0	-30	-30	0
0	-30	-30	0



## - CASOS MAS COMPLEJOS

### Vertical and Horizontal Edge Detection

1	0	-1
1	0	-1
1	0	-1

Vertical

1	1	1
0	0	0
-1	-1	-1

Horizontal

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

\*

1	1	1
0	0	0
-1	-1	-1

=

0	0	0	0
30	10	-10	-30
30	10	-10	-30
0	0	0	0

3:27

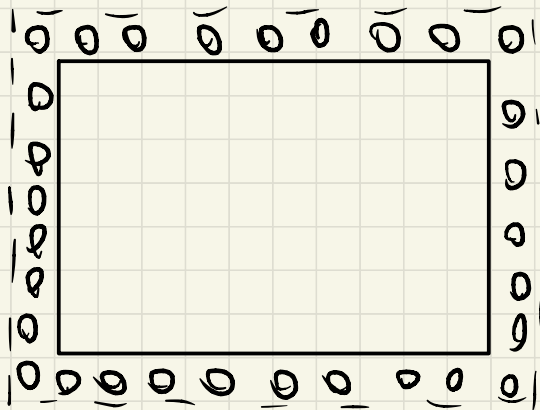
Andrew Ng

- EXISTEN OTRO TIPO DE FILTROS PARA DETECTAR BORDES VERTICALES / HORIZONTALES (VISION POR COMPUTADOR)
- LA APROXIMACION DE DC ES APROFUNDAR ESTOS FILTROS

## ② PADDING

- VARIOS FILTROS PUEDEN REDUCIR EL TAMAÑO DE LA IMAGEN
- SE PLENAN LOS BORDES CON INFORMACIÓN EN LOS BORDES
- SOLUCIÓN:

padding = p = 1



# - VALID AND SAME CONVOLUTIONS

## Valid and Same convolutions

→ no padding

$$\begin{array}{l} \text{"Valid": } n \times n \quad * \quad f \times f \quad \rightarrow \quad \underline{n-f+1} \times n-f+1 \\ 6 \times 6 \quad * \quad 3 \times 3 \quad \rightarrow \quad 4 \times 4 \end{array}$$

"Same": Pad so that output size is the same as the input size.

$$\begin{array}{l} n+2p-f+1 \times n+2p-f+1 \\ n+2p-f+1 = n \Rightarrow p = \frac{f-1}{2} \end{array}$$

### ③ STRIDED CONVOLUTIONS

## Strided convolution

2	3	7	4	6	2	9
6	6	9	8	7	4	3
3 <sup>3</sup>	4 <sup>4</sup>	8 <sup>4</sup>	3	8	9	7
7 <sup>1</sup>	8 <sup>0</sup>	3 <sup>2</sup>	6	6	3	4
4 <sup>-1</sup>	2 <sup>0</sup>	1 <sup>3</sup>	8	3	4	6
3	2	4	1	9	8	3
0	1	3	9	2	1	4

7x7

\*

3	4	4
1	0	2
-1	0	3

3x3

stride = 2

=

91	100	83

# Summary of convolutions

$n \times n$  image       $f \times f$  filter

padding  $p$       stride  $s$

$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$$



4:14 / 8:57

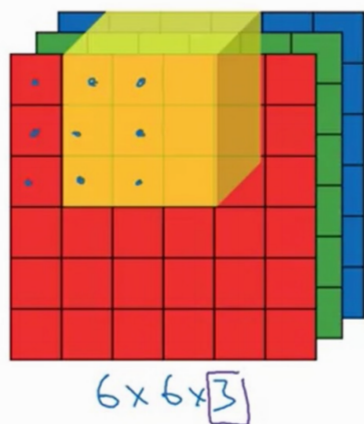


Andrew Ng

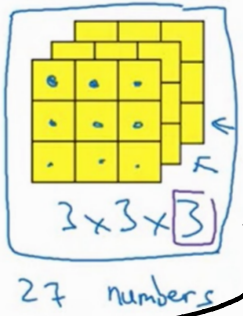
- EN MATEMÁTICAS ESTA OPERACIÓN ES MÁS  
PRECISAMENTE LLAMADA CORRELACION CANONICA,  
CORRELACION ES SIGUIENTEMENTE DISTINTO

# ④ CONVOLUTION DE IMÁGENES CON MÚLTIPLES CANALES VOLUMEN

## Convolutions on RGB image

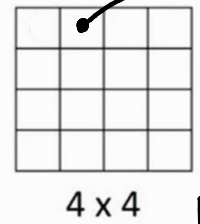


\*



Filter with volume

=



depth depth  
MUST BE EQUAL

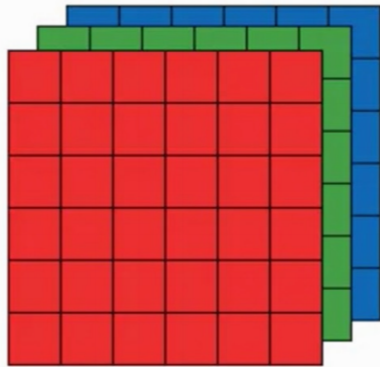
Se obtiene de multiplicar los 27 números como en el dibujo y después sumarlos.

Andrew Ng



# ⑤ convolucion con multiples filtros

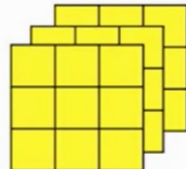
Multiple filters



$6 \times 6 \times 3$

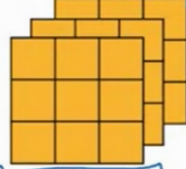
DOS FILTROS

vertical edge



$3 \times 3 \times 3$

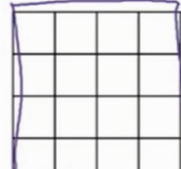
horizontal edge



$3 \times 3 \times 3$

\*

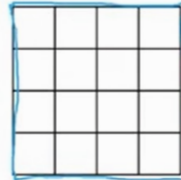
=



$4 \times 4$

\*

=

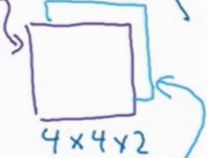


$4 \times 4$

RESULTADO  
ES un volumen



$4 \times 4 \times 2$



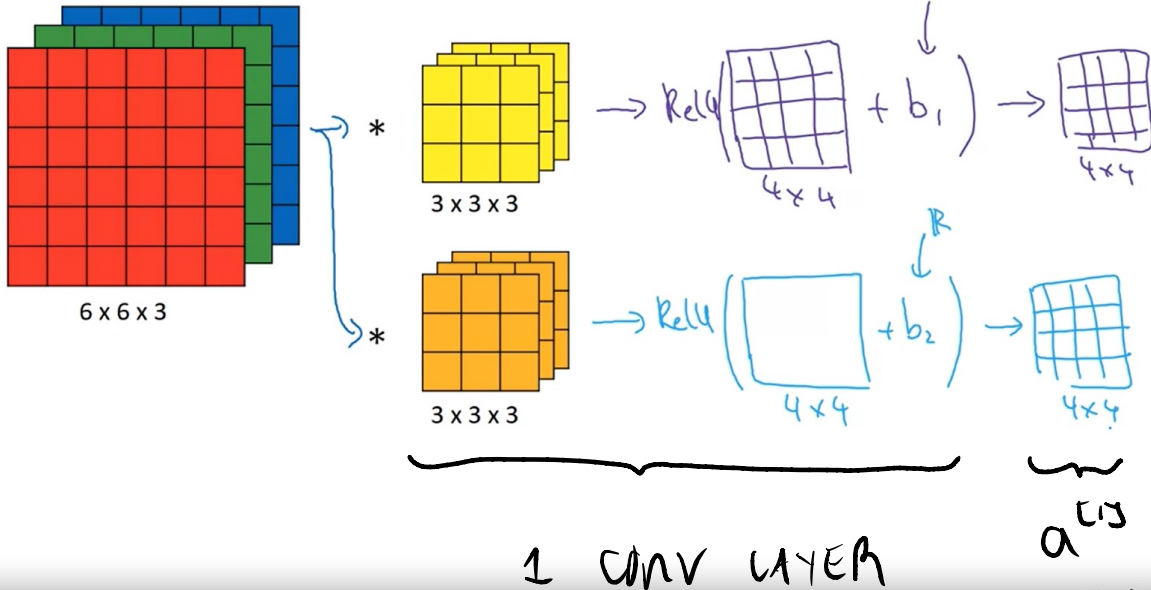
$4 \times 4 \times 2$

Summary:  $n \times n \times n_c$  \*  $f \times f \times n_c$   
 $6 \times 6 \times 3$   $3 \times 3 \times 3$

Andrew Ng

# ① RNN NEURONAL CON UNA CAPA CONVOLUCIONAL

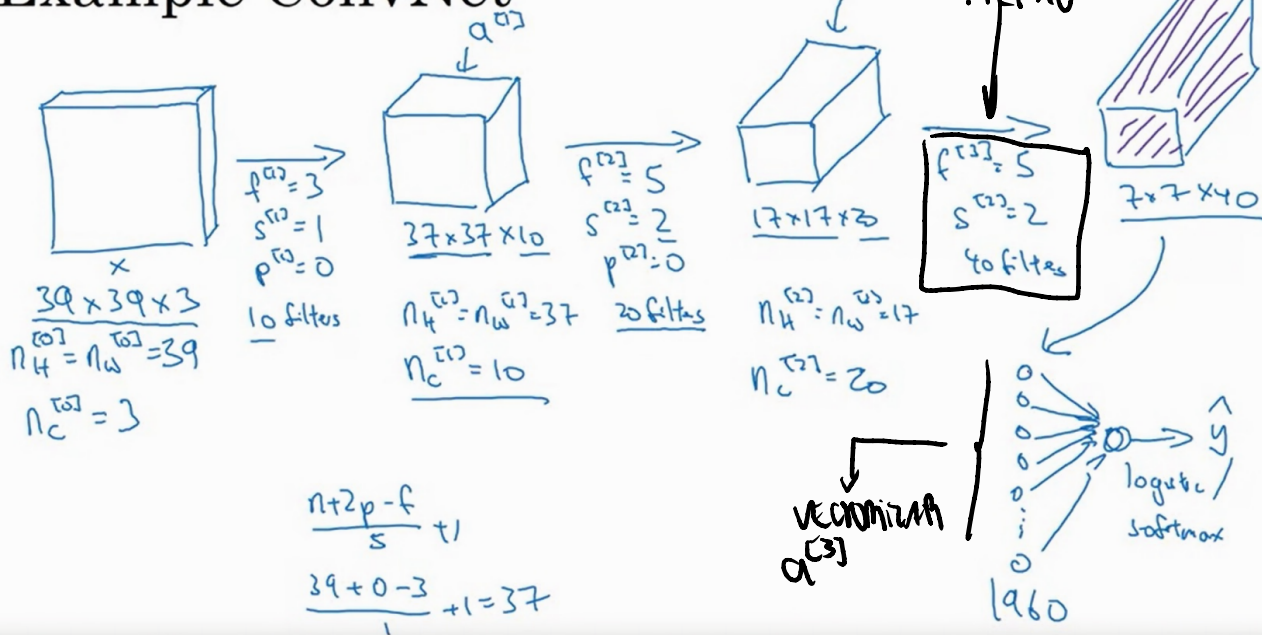
## Example of a layer



- observarse que el número de parámetros de una capa no depende del tamaño de la imagen. Solo de el número de filtros, su altura, ancho y profundidad + los sesgos que son tantos como filtros existen.

# ② EJEMPLO DE RED NEURONAL CONV PROFUNDA (PUNTA)

## Example ConvNet



Andrew Ng

### ③ CAPA DE POOLING (HAS NO PARAMETERS TO LEARN)

- MAX POOLING / AVERAGE POOLING

Pooling layer: Max pooling NO HAY PARAMETROS

Aplicar función ↓

1	3	2	1	3
2	9	1	1	5
1	3	2	3	2
8	3	5	1	0
5	6	1	2	9

5x5

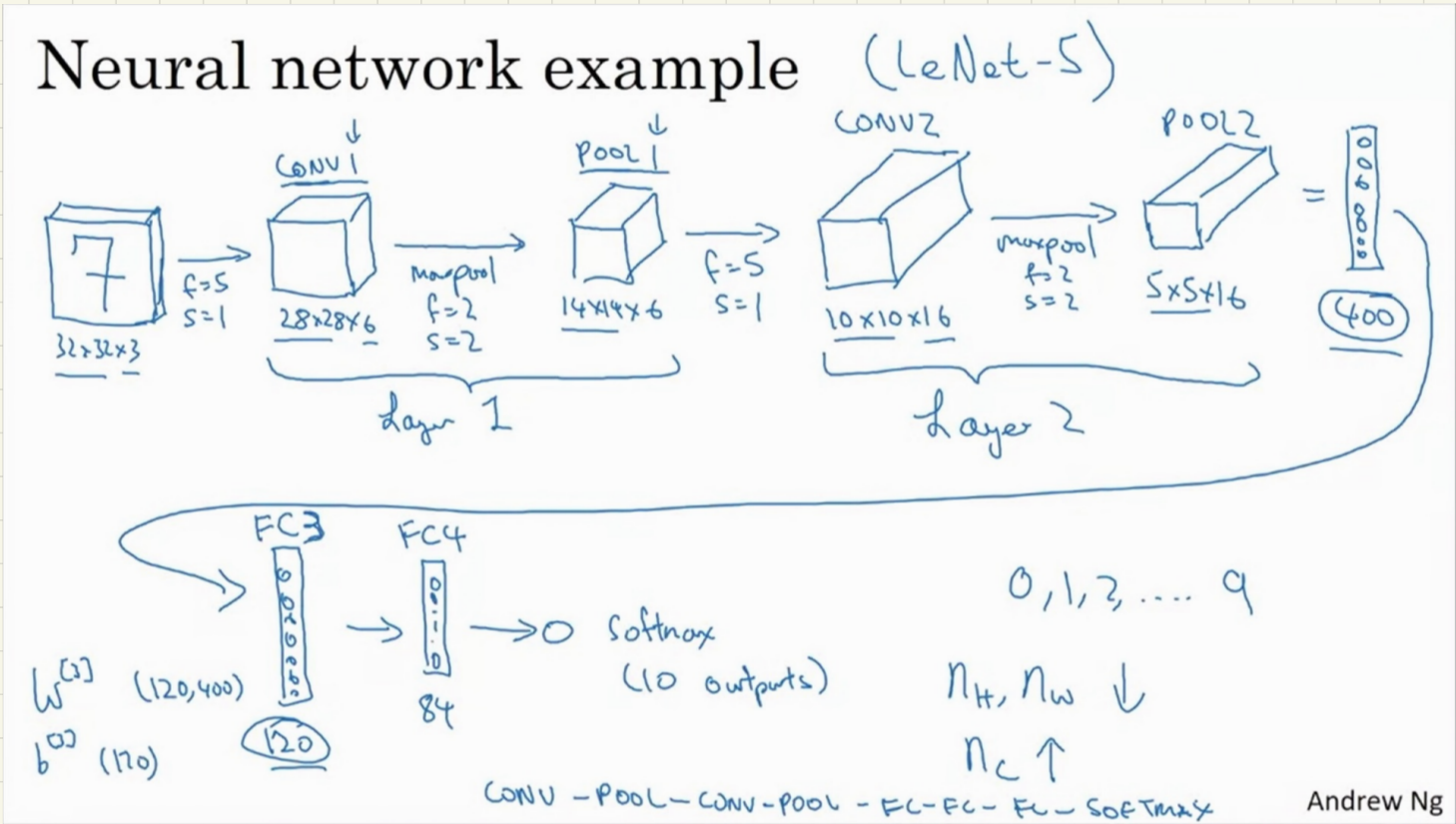

$f=3$   
 $s=1$

4:43 Andrew Ng

4:19 / 10:25

- SE HACE PARTE INDEPENDIENTE PARA CADA CANAL.
- USUALMENTE NO SE USA PADDING

# ④ EJEMPLO DE RED NEURONAL CONV PROFUNDA



Ahora la convencion es incluir la capa de pooling en la def. de CAPA CONVOLUCIONAL

# Neural network example

	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	— 3,072 $a^{[0]}$	0
CONV1 (f=5, s=1)	(28,28,6)	4,704	456 ←
POOL1	(14,14,6)	1,176	0 ←
CONV2 (f=5, s=1)	(10,10,16)	1,600	2,416 ←
POOL2	(5,5,16)	400	0 ←
FC3	(120,1)	120	48,120 } }
FC4	(84,1)	84	10,164 } }
Softmax	(10,1)	10	850

⑤ ¿Por que necesitamos convoluciones?

- Parámetros compartidos
- Sparsity of connections: en cada capa son pocas inputs importantes



# ① EJEMPLOS DE REDES

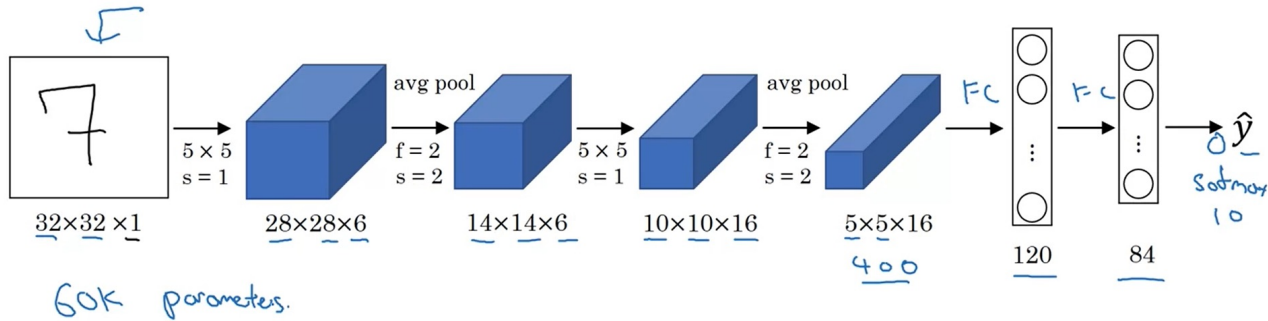
- LeNet-5
  - AlexNet
  - VGG
- } CLÁSICAS

- ResNet
- Inception

# LeNet - 5 (Le Cun 1988: Gradient based learning applied to document recognition)

- Historicamente una de las primeras redes en resolver exitosamente un problema: clasificación de dígitos.

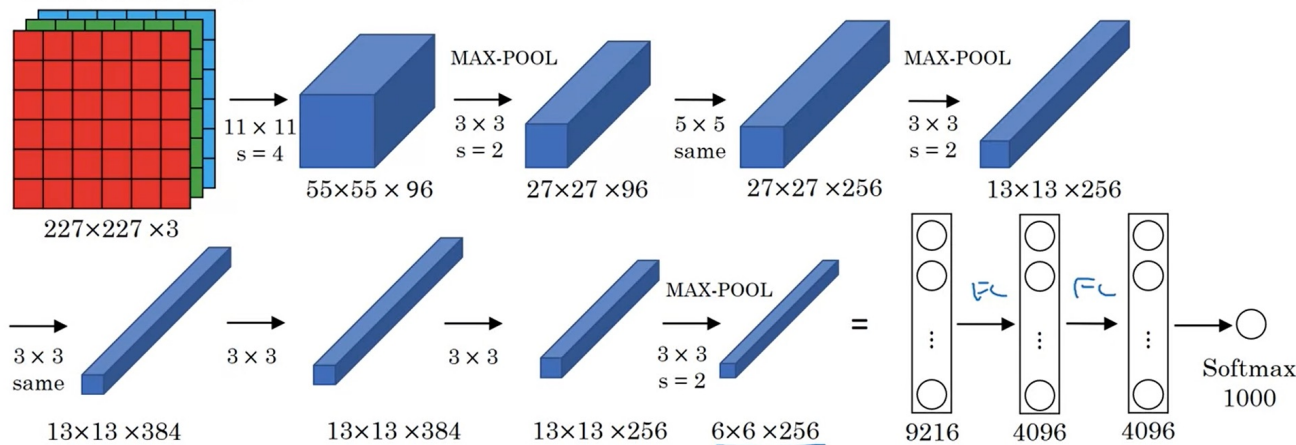
## LeNet - 5



# AlexNet : Krizhevsky 2012 : Image net classification with deep conv. neural network.

- Similar a LeNet-5 pero con 60 millones de parámetros.
- ESTE MODELO ES EL PUNTO DE INFLEXION EN CV USANDO DC

## AlexNet



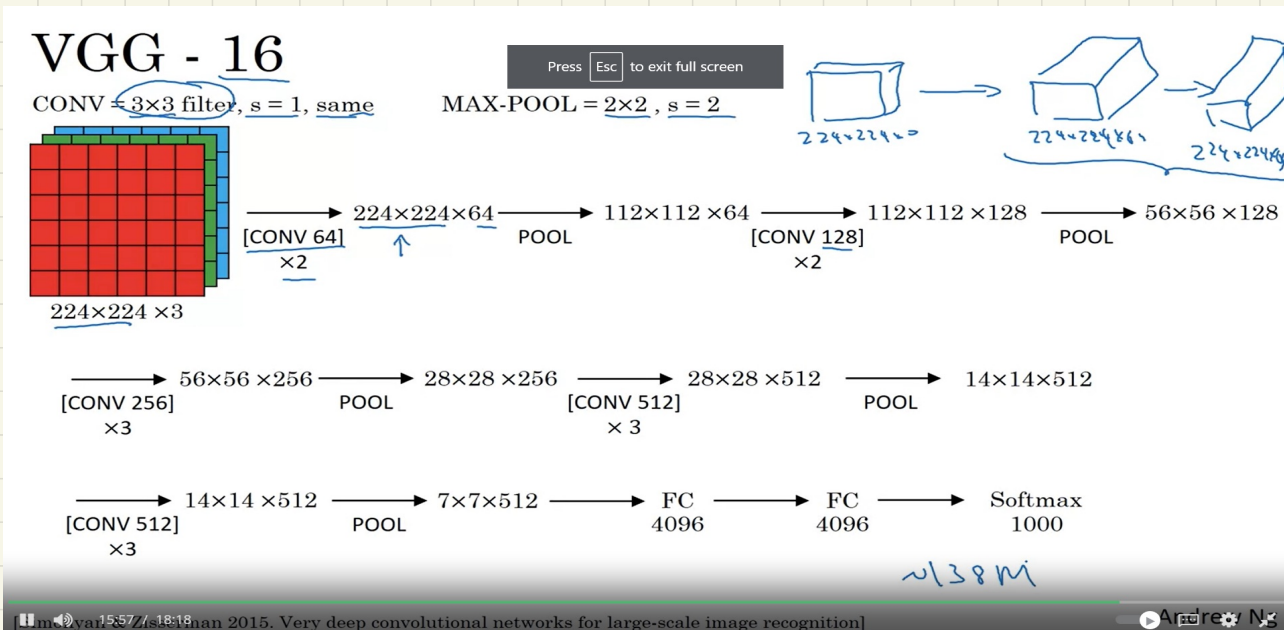
- Similar to LeNet, but much bigger.

216

160M param.

VGG - 16 : Simonyan & Zisserman, 2015. Very deep conv. NN for large scale image recognition

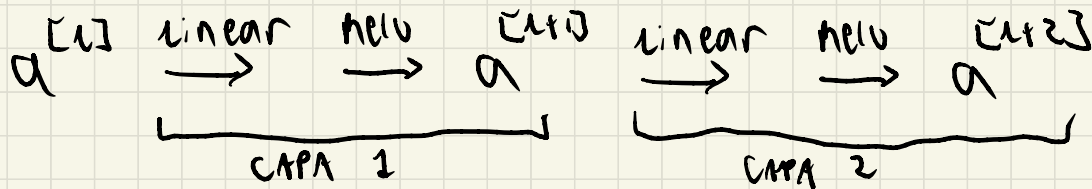
- simplifica las anteriores arquitecturas. ES MAS HOMOGÉNEA PERO MUCHO MAS GRANDE: 138 millones de parámetros.
- 16 CAPAS CON PESO

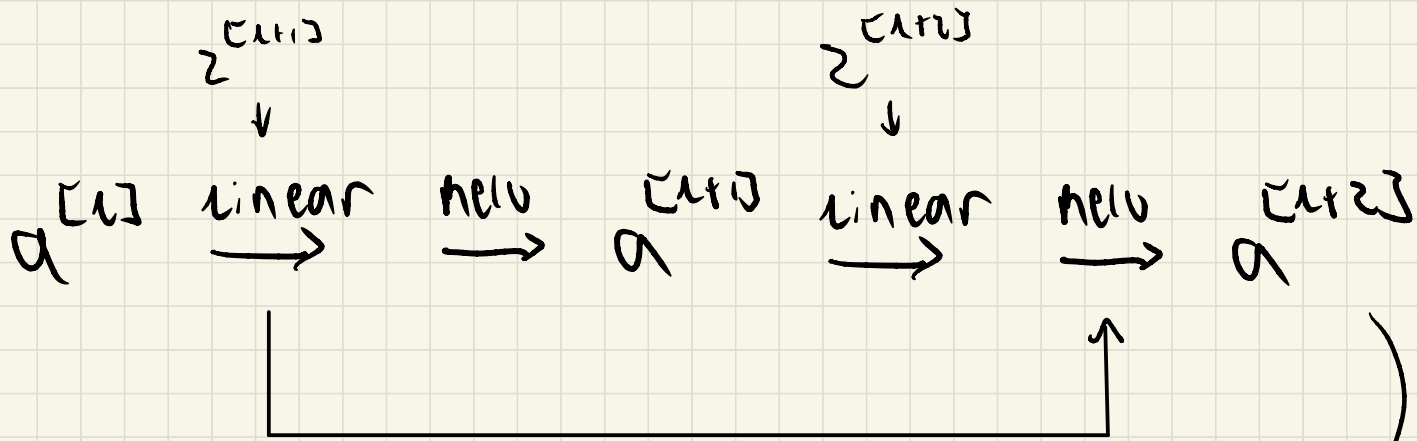


# RESNET IHe 2015. Deep residual networks for IMAGE RECOGNITION

- PERMITE ENTRENAR REDES MUY GRANDES
- USA UN "SHORT CUT" O SKIP CONNECTION
- MITIGA EL PROBLEMA DE GRADIENTES  $\rightarrow 0$  O  $\rightarrow \infty$

## DOS CAPAS DE UNA RED



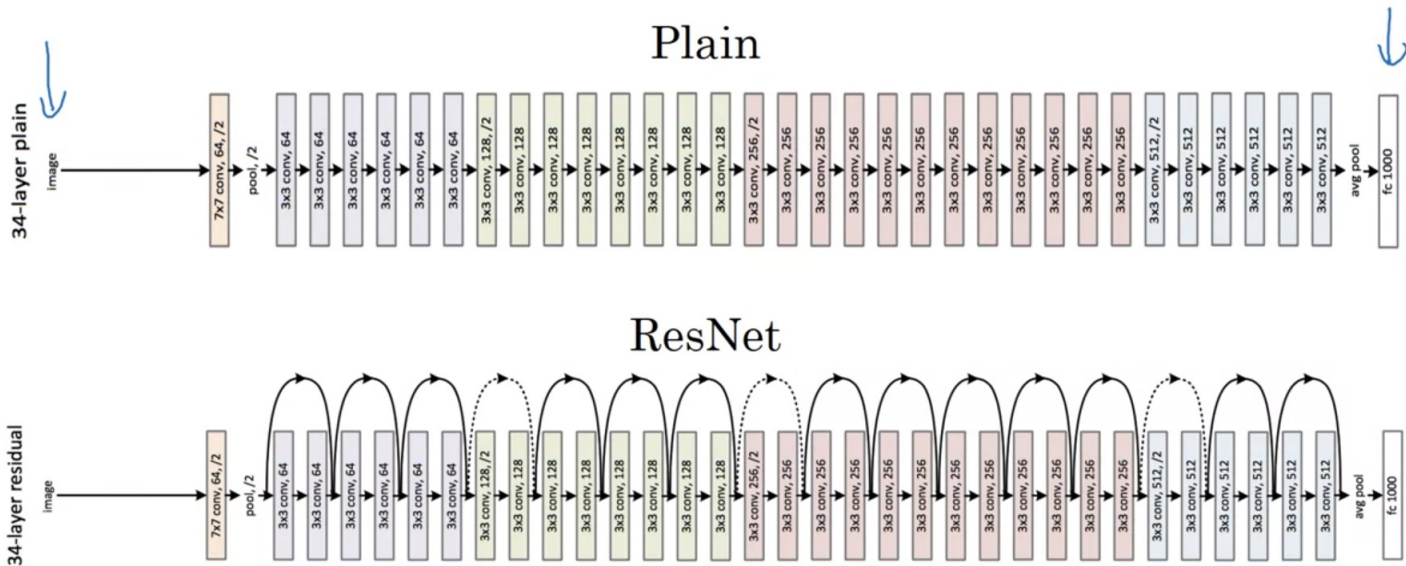


SHORT CUT  
(RESIDUAL BLOCK)

$$a^{[1+2]} = g(z^{[1+2]} + a^{[1]})$$

- HAY que tener cuidado con las dimensiones para que  
 $\alpha^{cat(x)} = g(z^{cat(x)} + a^{cat(x)}) \rightarrow$  HAGA SENTIDO

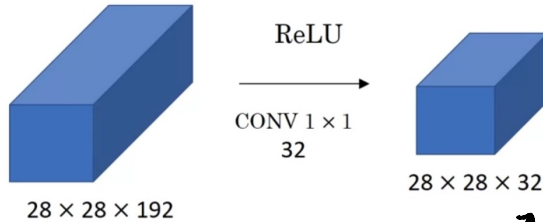
# ResNet



# 1x1 CONVOLUCIONES

- PERMITE CONTRAER & EXPANDIR EL VOLUMEN
- ESTO ES RELEVANTE EN INCEPTION NET

Using 1x1 convolutions

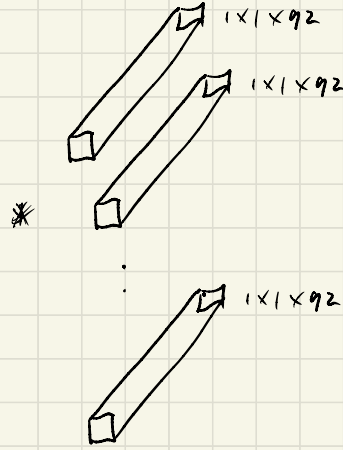


PERMITE HACER ESTA REDUCCION  
PARA FORMALIZAR ESO





$28 \times 28 \times 192$



32 filtros

Me lo  
⇒



$28 \times 28 \times 32$

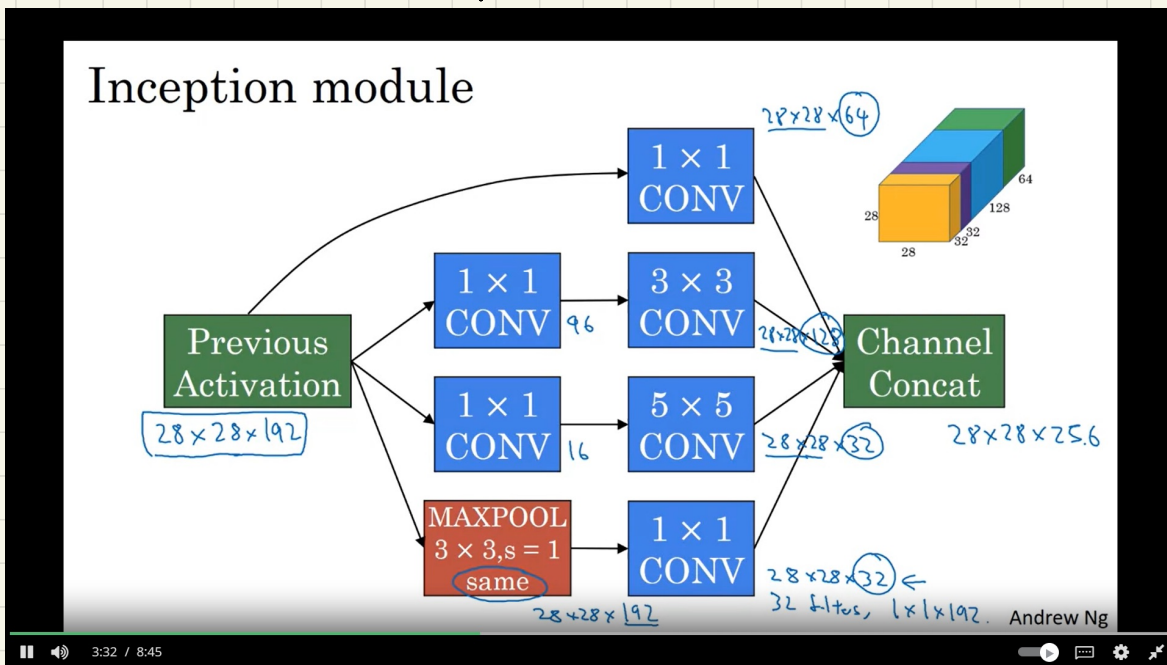
Observación: En general no siempre se va reportar el volumen de los filtros porque tienen que ser igual al volumen del input.

## EN RESUMEN

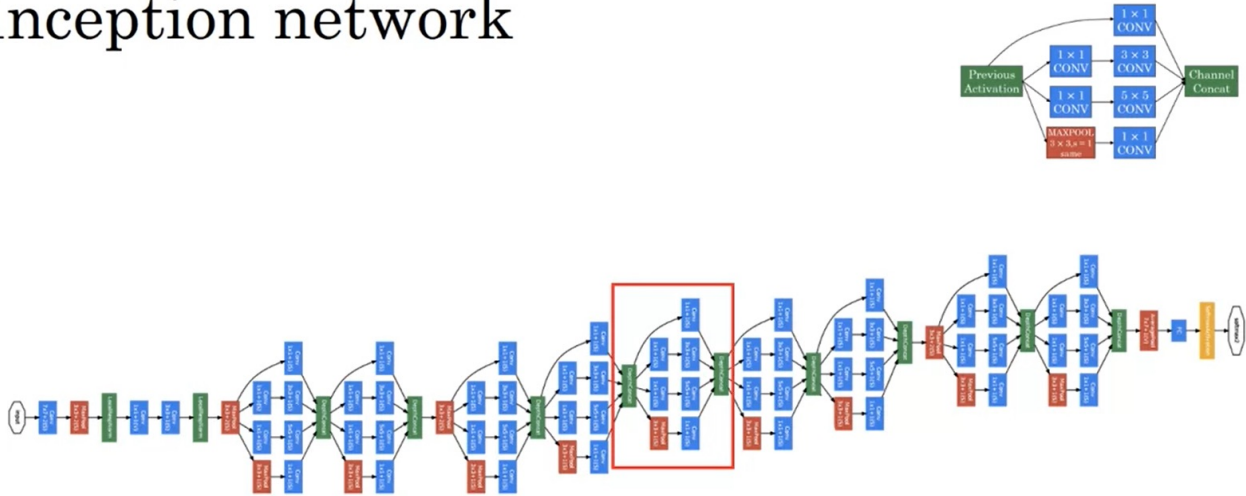
- LAS CAPAS DE pooling se pueden usar para Agrandar o reducir altura y ancho.
- LOS filtros  $1 \times 1$  para modificar el volumen.

# Inception Network: Szegedy et al 2014 Going deeper in conv. networks.

- NO es necesario decidir el tamaño de los filtros o capas de pooling. se ponen todas.



# Inception network



[Szegedy et al., 2014, Going Deeper with Convolutions]

Andrew Ng

## - Mobile Net

ES UNA ARQUITECTURA QUE PUEDE DESPLEGARSE CON POCAS CAPACIDADES DE COMPUTO.

ESTA BASADA EN UNA FORMA MAS EFICIENTE DE HACER LOS CALCULOS DE CONVOLUCION separando en dos etapas

- ① "depthwise" convolution
- ② "pointwise" convolution

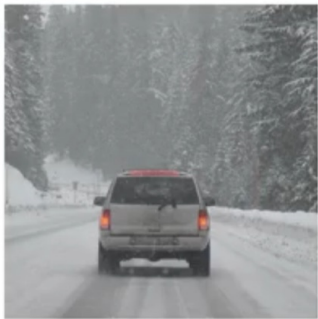
## - Efficient Net

ES UNA ARQUITECTURA QUE PERMITE Elegir ARQUITECTURAS DE MENOS A MAS RECURSOS COMPUTACIONALES.

# TAREA : CLASSIFICATION, LOCALIZATION, DETECTION

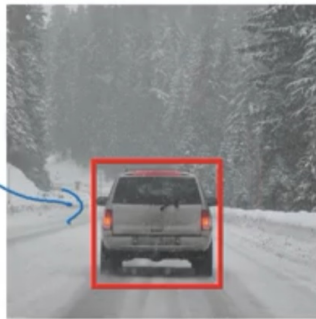
## What are localization and detection?

Image classification



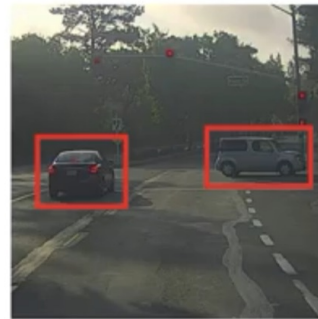
"Car"

Classification with localization



"Car"

Detection

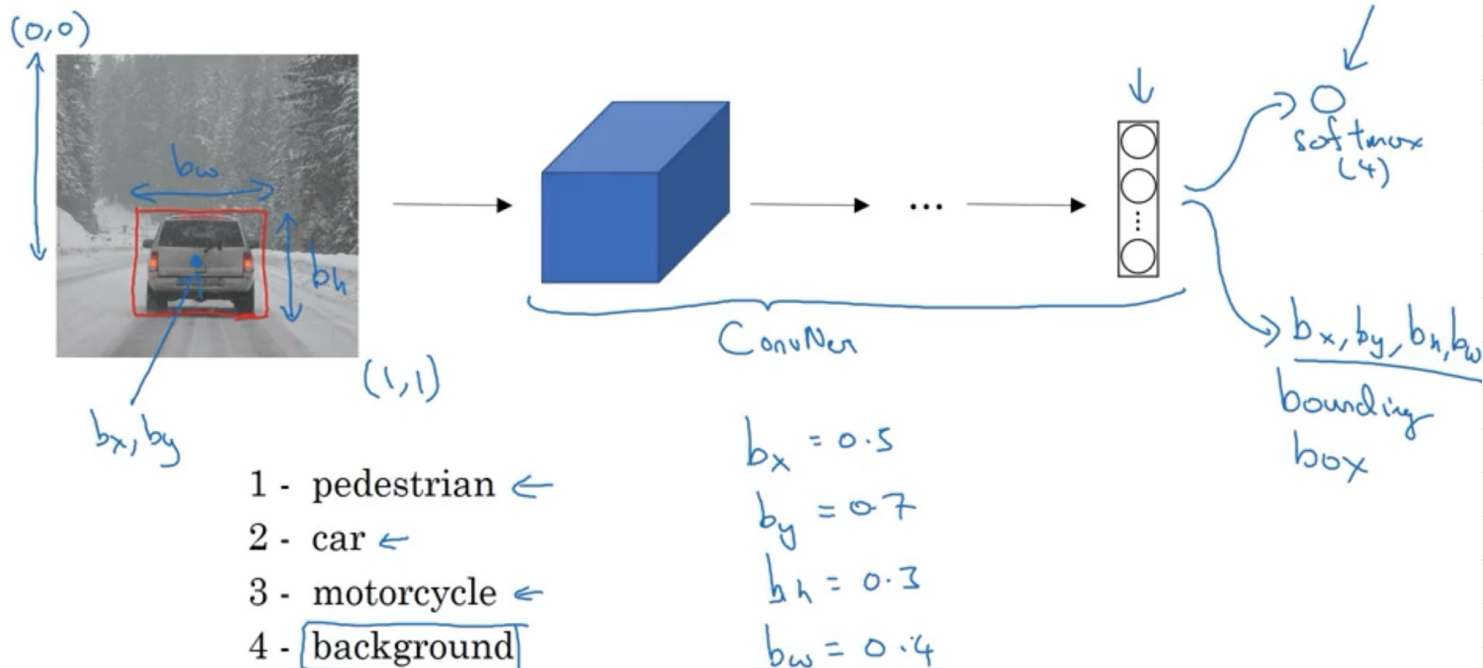


multiple objects

1 object

# TAREA : CLASSIFICACION, LOCALIZACION

## Classification with localization

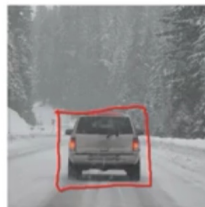


# TAREA : CLASSIFICATION, LOCALIZATION

## Defining the target label $y$

- 1 - pedestrian
- 2 - car ←
- 3 - motorcycle
- 4 - background ←

Need to output  $b_x, b_y, b_h, b_w$ , class label (1-4)



$$L(\hat{y}, y) = \begin{cases} (\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 \\ + \dots + (\hat{y}_n - y_n)^2 & \text{if } y_i = 1 \\ (\hat{y}_1 - y_1)^2 & \text{if } y_i = 0 \end{cases}$$

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

is there any object?

$(x, y)$

$$\begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ \dots \\ \dots \\ \dots \\ \dots \\ \dots \\ \dots \\ \dots \end{bmatrix}$$

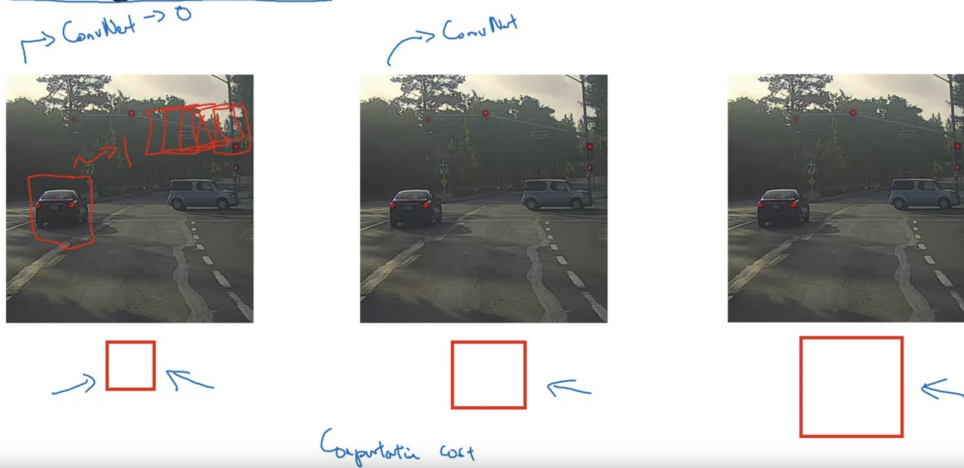
← "don't care"



# TAREA : DETECCIÓN

- ENTRENAR UN ALGORITMO DE CLASIFICACION
- USAR "SLIDING WINDOW DETECTION" PUEDE IMPLEMENTARSE DE FORMA CONVENCIONAL: TODAS LAS PREDICCIONES SE HACEN DE FORMA SIMULTÁNEA.

## Sliding windows detection

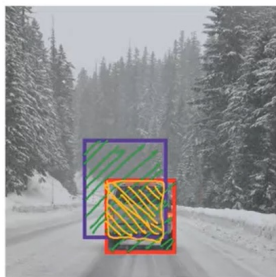


TAREA : BOUNDING BOX PREDICTIONS

TAREA = EVALUACION DE QUE TAN BUENA ES LA LOCALIZACION DE UN OBJETO

Métrica: INTERSECCION SOBRE UNION

## Evaluating object localization



Intersection over Union (IoU)

$$= \frac{\text{size of } \text{[yellow box]}}{\text{size of } \text{[green box]}}$$

"Correct" if  $\text{IoU} \geq 0.5$  ←

0.6 ←

More generally, IoU is a measure of the overlap between two bounding boxes.

Andrew Ng

- un problema con las técnicas hasta este punto es que se detecta varias veces un objeto.

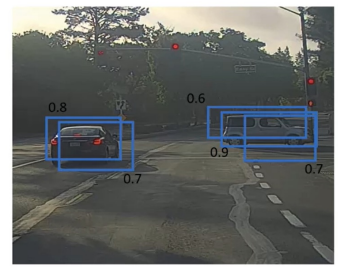
- NON MAX SUPPRESSION : consiste en :

1 Descartar todas las cajas con prob. baja  $\leq 0.6$

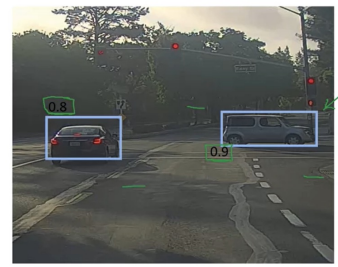
2 Elegir la que tenga prob. alta

3 Descartar todas las que tengan IoU  $\geq 0.5$

Non-max suppression example

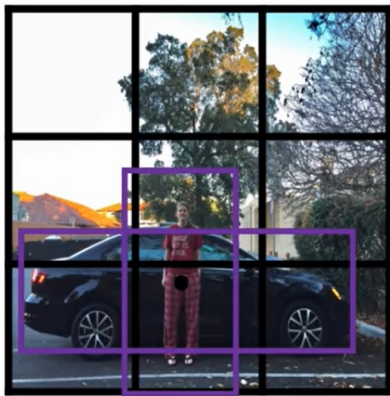


Non-max suppression example

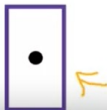


# TAREA : DETECCIÓN DE MÚLTIPLES OBJETOS

## Anchor box example



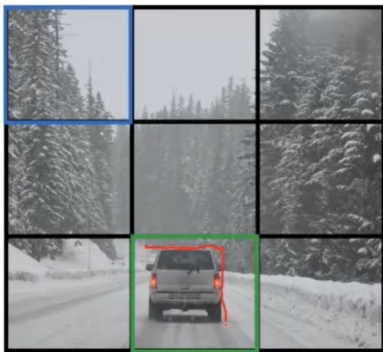
Anchor box 1:      Anchor box 2:



$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \begin{array}{l} | \\ b_x \\ b_y \\ b_h \\ b_w \\ | \\ 0 \\ 0 \\ 0 \\ | \\ b_x \\ b_y \\ b_h \\ b_w \\ | \\ 0 \\ 0 \\ 0 \end{array}$$

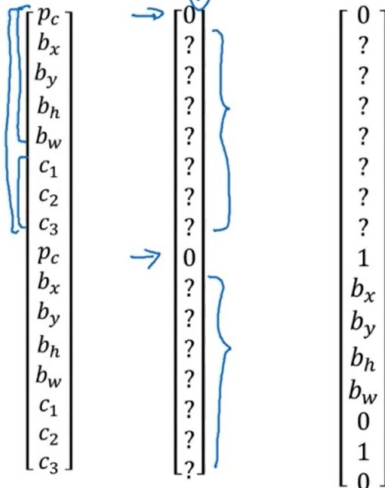
# YOLO ALGORITHM PONE TODAS TÉCNICAS EN UN ALGORITMO

## Training



- 1 - pedestrian
- 2 - car
- 3 - motorcycle

$y =$



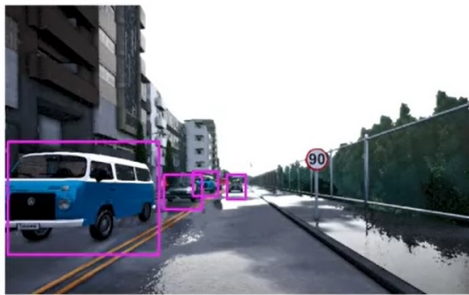
$3 \times 3 \times 16$   
 $y$  is  $3 \times 3 \times 2 \times 8$   
↑ #anchors    ↑  $5 + \#classes$

# SEMANTIC SEGMENTATION

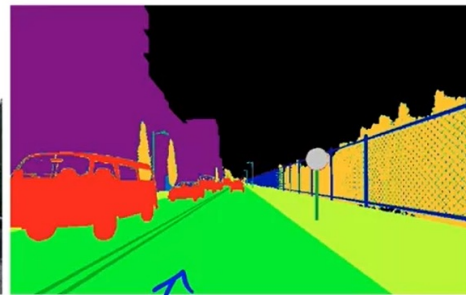
## Object Detection vs. Semantic Segmentation



Input image

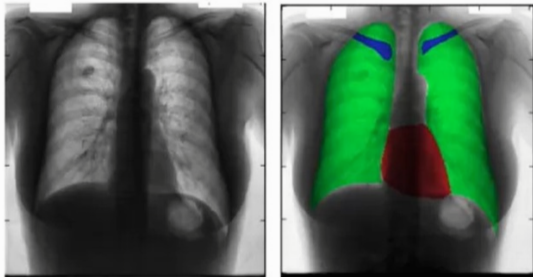


Object Detection

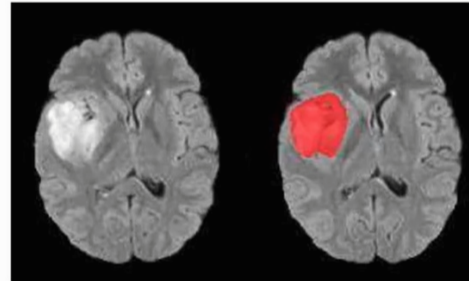


Semantic Segmentation

# Motivation for U-Net



Chest X-Ray



Brain MRI

[Novikov et al., 2017, Fully Convolutional Architectures for Multi-Class Segmentation in Chest Radiographs]

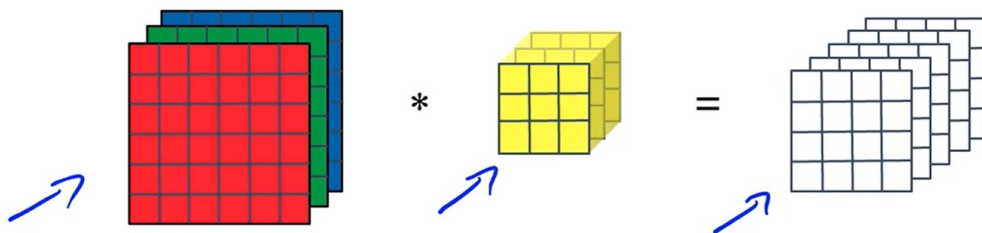
[Donat et al., 2017, Automatic Brain Tumor Detection and Segmentation Using U-Net Based Fully Convolutional Networks]



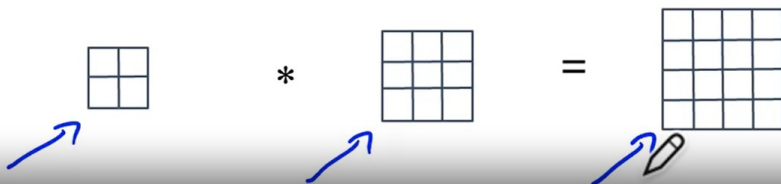
# TRANSPOSE CONVOLUTION

## Transpose Convolution

Normal Convolution



Transpose Convolution



0:56

# Transpose Convolution

2	1
3	2

2x2

1 <sup>1</sup>	2 <sup>1</sup>	1 <sup>1</sup>
2 <sup>1</sup>	0 <sup>1</sup>	1 <sup>1</sup>
0 <sup>1</sup>	2 <sup>1</sup>	1 <sup>1</sup>




4x4

filter  $f \times f = 3 \times 3$

padding  $p=1$

stride  $s=2$

# Transpose Convolution

2	1
3	2

$2 \times 2$

$1^1$	$2^1$	$1^1$
$2^1$	$0^1$	$1^1$
$0^1$	$2^1$	$1^1$




$4 \times 4$

filter  $f \times f = 3 \times 3$

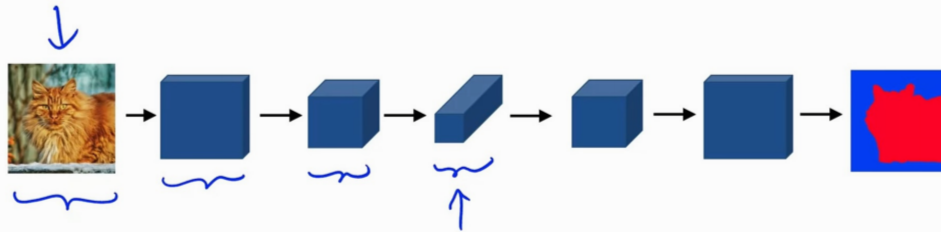
padding  $P=1$

stride  $s=2$

# U-Net

- ESTA ARQUITECTURA SE APLICA EN TRANSPOSE CONVOLUTION PARA RESOLVER EL PROBLEMA DE SEMANTIC SEGMENTATION.

## Deep Learning for Semantic Segmentation



## RECONOCIMIENTO DE CARAS: VERIFICACION VS RECONOCIMIENTO

- one-shot learning: con un solo ejemplo reconocer si la persona está en lista
- estrategia aprender una función de similitud:  $d(\text{img}_1, \text{img}_2)$

### Face verification vs. face recognition

#### → Verification

1:1

- Input image, name/ID
- Output whether the input image is that of the claimed person

#### → Recognition

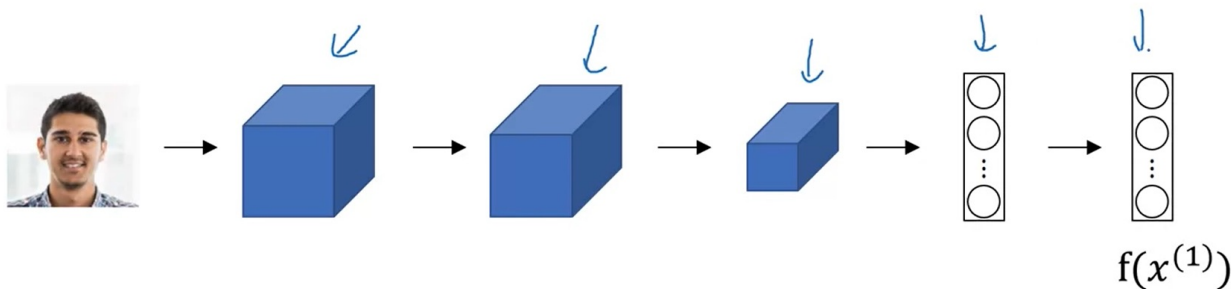
1:K

- Has a database of K persons
- Get an input image
- Output ID if the image is any of the K persons (or “not recognized”)

Andrew Ng

# Siamese Network

## Goal of learning



Parameters of NN define an encoding  $f(x^{(i)})$  128

Learn parameters so that:

If  $x^{(i)}, x^{(j)}$  are the same person,  $\|f(x^{(i)}) - f(x^{(j)})\|^2$  is small.

If  $x^{(i)}, x^{(j)}$  are different persons,  $\|f(x^{(i)}) - f(x^{(j)})\|^2$  is large.

- función objetivo: TRIPLET LOSS function

- los datos se organizan en tripletes:  $(A, P, N)$

HAY que tener cuidado porque si se hace aleatorio en lugar de minimizar la pérdida.

Anchor

positive (same)

Negative (different)

-  $q(A, P, N)$

$$= \max \{ \|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + d, 0 \}$$

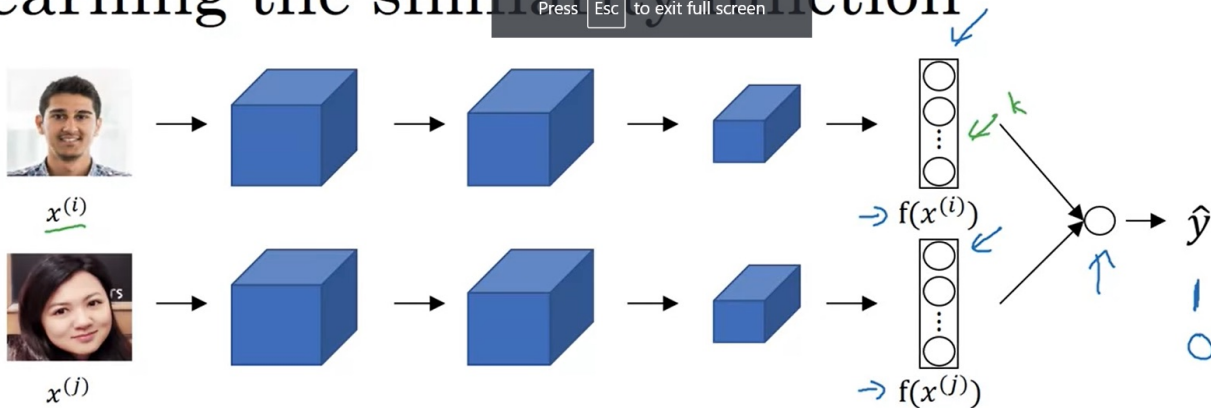
↓  
n-encoding de siamese architecture

$$J = \sum_{i=1}^m q(A^{(i)}, P^{(i)}, N^{(i)})$$

- una vez entrenada la red se puede usar con ejemplos nuevos (one shot learning)

- como un problema de clasificación con estimación:

# Learning the similarity function



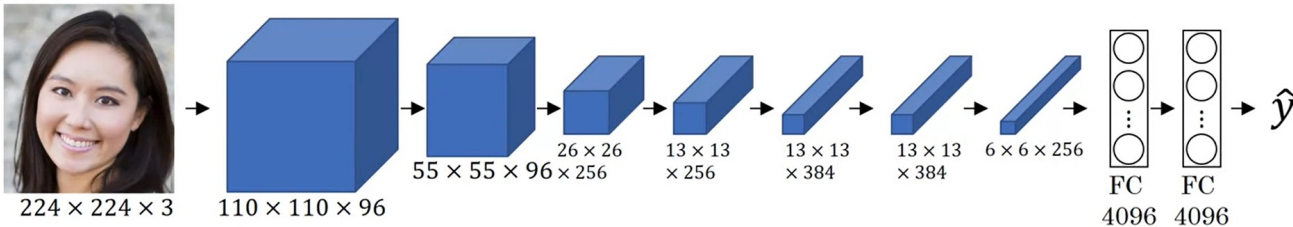
$$\hat{y} = \sigma \left( \sum_{k=1}^{128} \underset{\uparrow}{w_k} \underbrace{|f(x^{(i)})_k - f(x^{(j)})_k|}_{\text{similarity}} + \underset{\uparrow}{b} \right)$$



- ¿Que' HACE una CONV NN?

VISUALIZANDO Que HACE una CONV NN : visualizing and understanding CONV NN. 2013.

## Visualizing what a deep network is learning

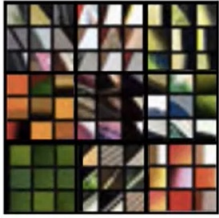


Pick a unit in layer 1. Find the nine image patches that maximize the unit's activation.

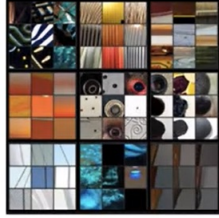
Repeat for other units.



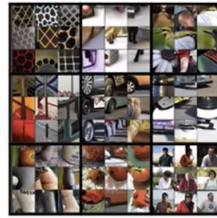
# Visualizing deep layers: Layer 3



Layer 1



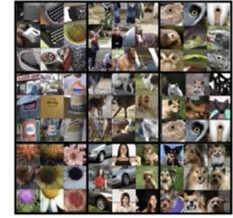
Layer 2



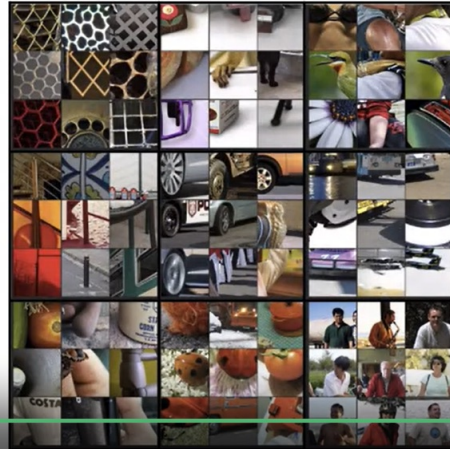
Layer 3



Layer 4



Layer 5



# Neural style transfer

## Neural style transfer cost function



Content C

Style S



Generated image G ←

$$J(G) = \alpha J_{\text{Content}}(C, G) + \beta J_{\text{Style}}(S, G)$$

[Gatys et al., 2015. A neural algorithm of artistic style. Images on slide generated by Justin Johnson] Andrew Ng



2:10 / 3:59



# Find the generated image $G$

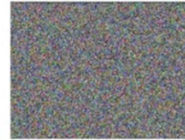
1. Initiate  $G$  randomly

$$G: \underline{100} \times \underline{100} \times \underline{3}$$

↑  
RGB

2. Use gradient descent to minimize  $J(G)$

$$G := G - \frac{\partial}{\partial G} J(G)$$



[Gatys et al., 2015. A neural algorithm of artistic style]

3:30

Andrew Ng



3:35 / 3:59



## FUNCIONES DE COSTO : contenido

- $J_{\text{content}}(C, b)$  se define comparando la representación de  $C$  y  $b$  en la capa  $n$  de una red preentrenada (e.g., VCG)
- si las funciones de activación son:

$$a^{c \rightarrow c} \quad \text{y} \quad a^{c \rightarrow b}$$
$$J_{\text{content}}(C, b) = \frac{1}{2} \| a^{c \rightarrow c} - a^{c \rightarrow b} \|^2$$

# FUNCIONES DE COSTO : ESTILO

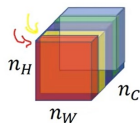
- primero definimos la correlacion entre canales:

$$C_{G,KK'} = \sum_i^{n_H} \sum_j^{n_W} a_{i,j,K} C_{ij}(G) a_{i,j,K'}$$

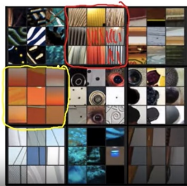
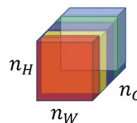
ESTO CAPTURA LA MODA DE ESTILO CARACTERISTICAS  
QUE OCURREN EN DIFERENTES PARTES DE LA IMAGEN

Intuition about style of an image

Style image



Generated Image



- FUNCIÓN DE COSTOS DE ESTRUC EN CAPA  $l$ :

$$J_{\text{struc}}^{(l)}(s, b) = \frac{1}{2 n_H^{(l)} n_W^{(l)} n_C^{(l)}} \sum_k \sum_{k'} (G_{kk'}^{(l)} - G_{kk'}^{(l')})^2$$

- FUNCIÓN DE COSTOS DE MÚLTIPLES CAPAS:

$$J_{\text{struc}}(s, b) = \sum_l J_{\text{struc}}^{(l)}(s, b)$$

↓  
hiperparámetros