Indexación, búsqueda y análisis en repositorios multimedia: imagen-video

Juan Carlos San Miguel

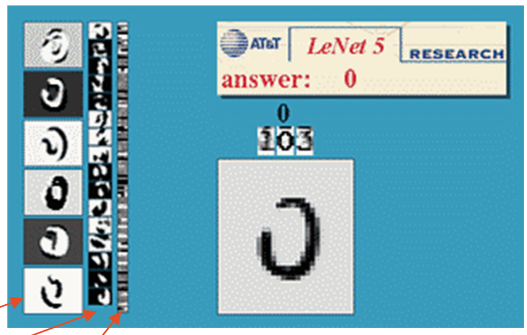
Agenda: Multimedia (imagen, video)

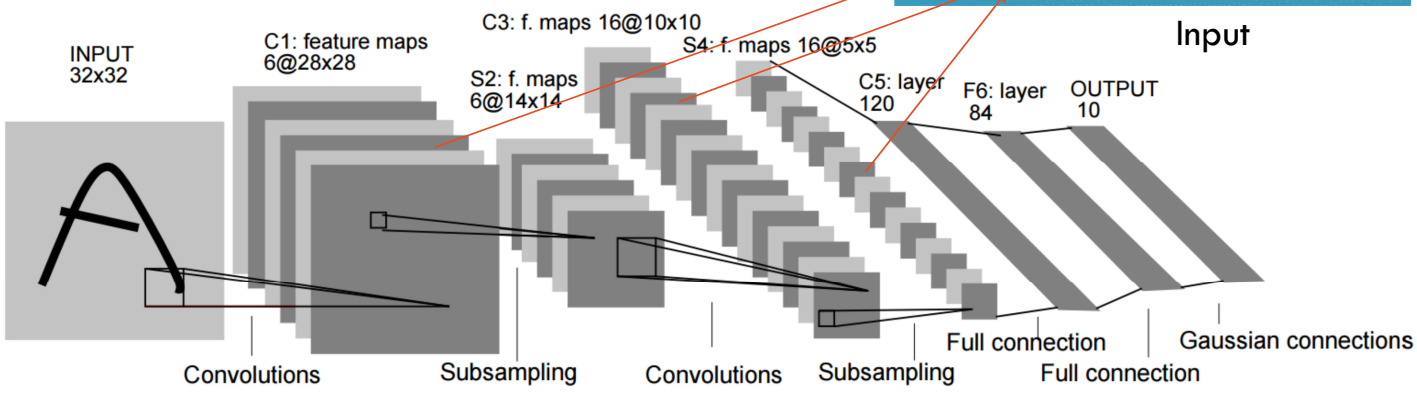
Temas avanzados

- Arquitecturas populares en clasificación
  - ZFNet
  - VGG
  - GoogleLeNet
  - ResNet
- Transfer Learning

Arquitecturas populares en clasificación

- LeNet: inicios
  - Conv filtros son 5x5, aplicados con stride 1
  - 2x2 submuestreo (Pooling) con stride 2

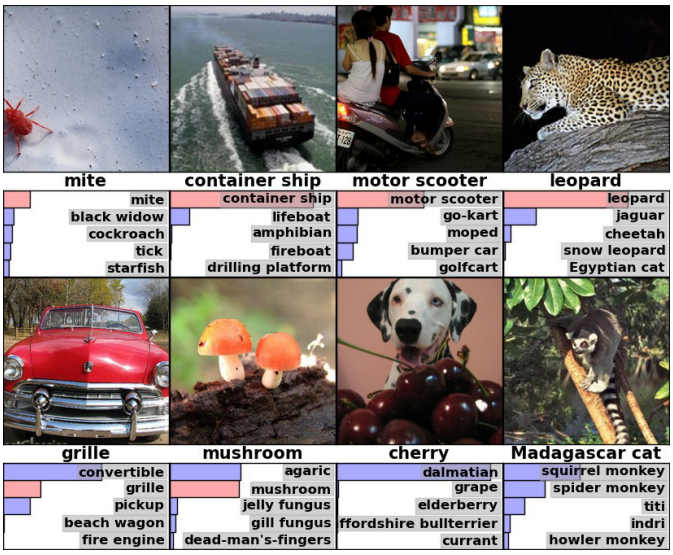




Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86(11): 2278–2324, 1998.

Arquitecturas populares en clasificación

- ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

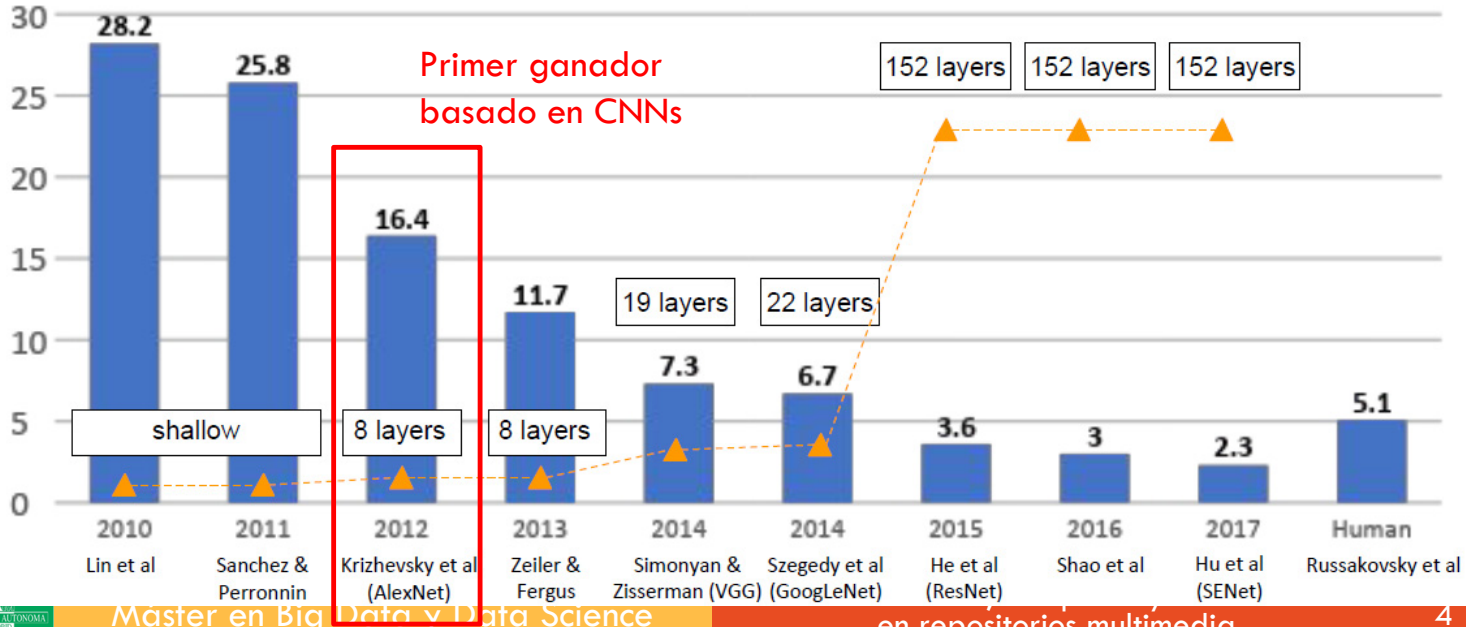


IMAGENET

- ~14M de imágenes obtenidas de internet
- Anotadas manualmente con 20k categorías
- Competición/Challenge:
  - 1k categorías
  - 1.2M imágenes train
  - 150K imágenes test

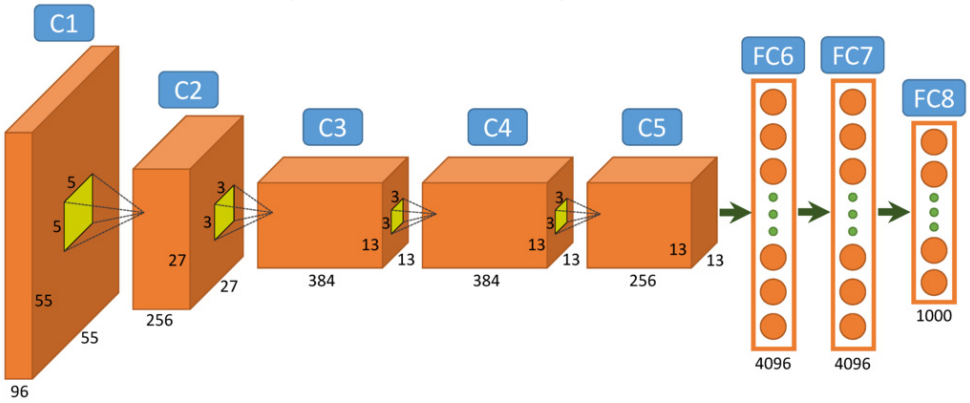
Arquitecturas populares en clasificación

- ImageNet Large Scale Visual Recognition Challenge (ILSVRC)



Arquitecturas populares en clasificación

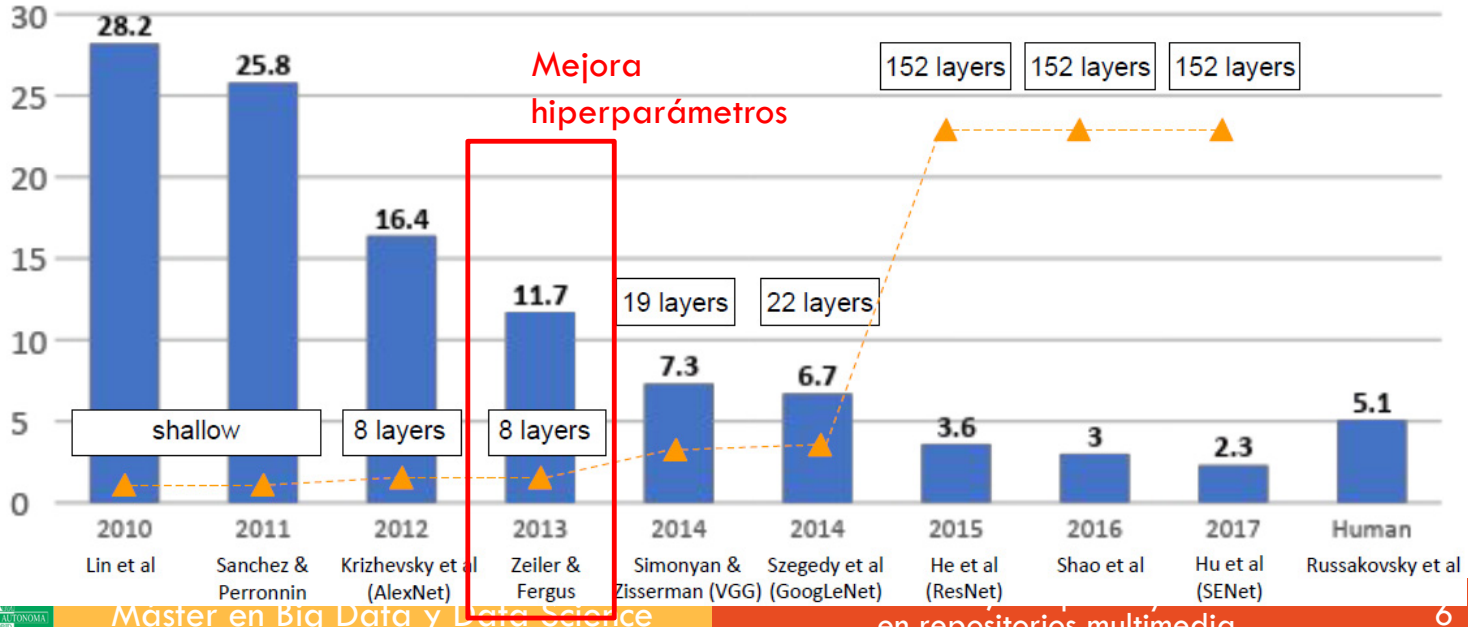
- AlexNet: ganador ILSVRC 2012
  - Max pooling, ReLU nonlinearity, regularización Dropout
  - Más datos y modelo más complejo (7 capas ocultas, ~650K unidades, ~60M params)
  - Aceleración con GPUs (~50x sobre CPU), entrenamiento ~1 semana con 2 GPUs



A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

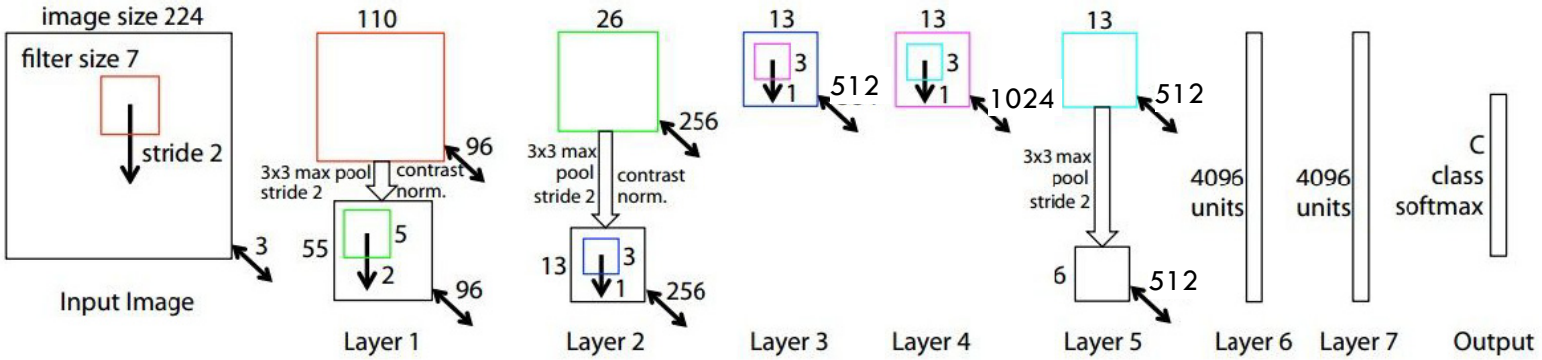
Arquitecturas populares en clasificación

- ImageNet Large Scale Visual Recognition Challenge (ILSVRC)



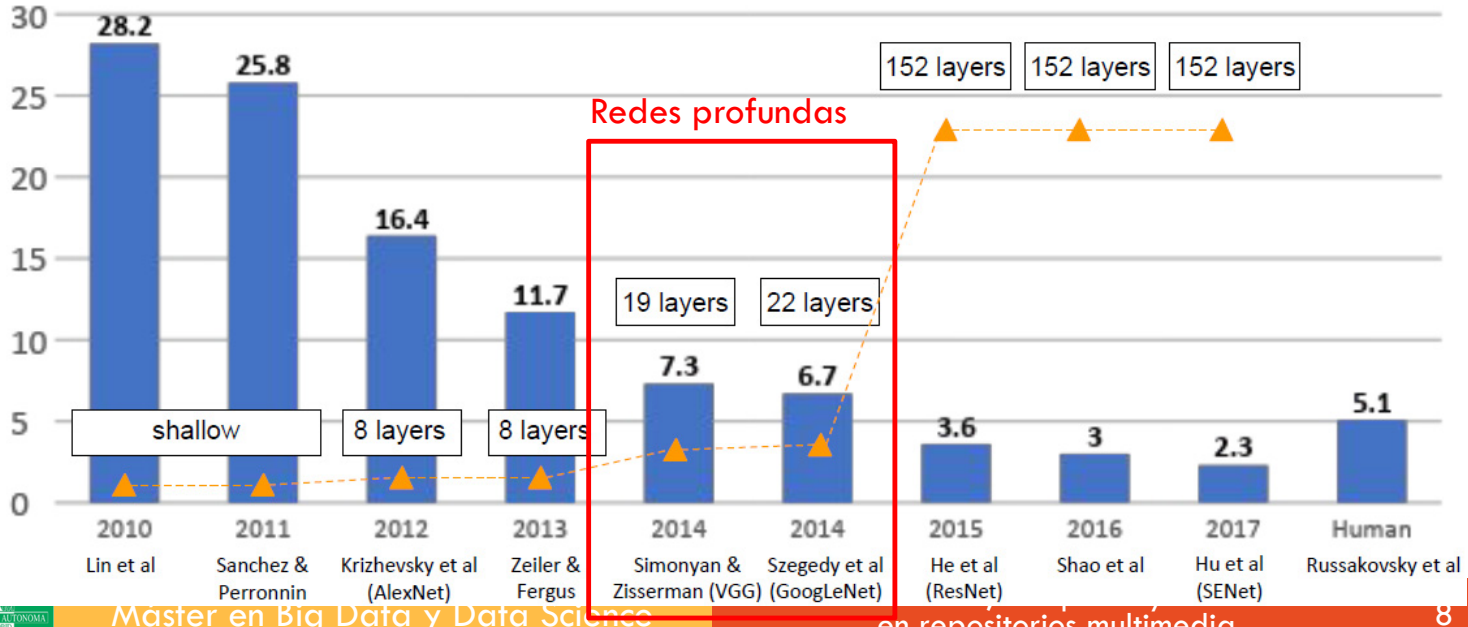
Arquitecturas populares en clasificación: ZFNet

- Cambios con respecto a AlexNet
  - CONV1: 11x11 stride 4 → 7x7 stride 2
  - CONV3,4,5: 384/384/256 filtros → 512/1024/512 filtros
  - Mejora error Top-5 16.4% → 11.7%



Arquitecturas populares en clasificación

- ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

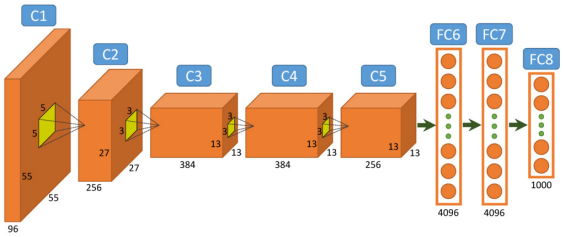


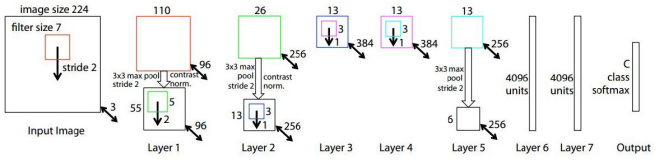
- Seleccione la afirmación correcta

a) AlexNet puede procesar una imagen de cualquier tamaño

b) ZFNet puede procesar una imagen de cualquier tamaño

c) Ambas AlexNet y ZFNet que las imágenes tengan tamaño fijo

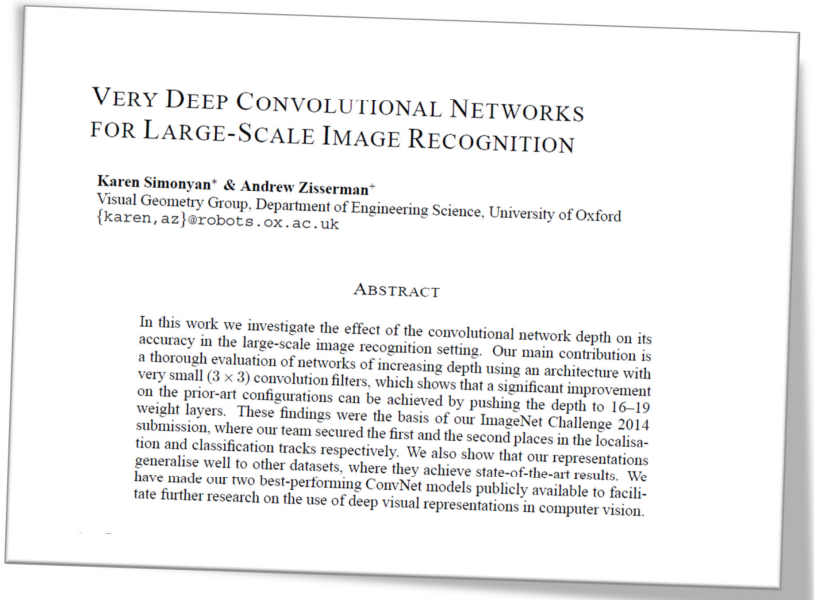




Arquitecturas populares en clasificación

- VGG Net
  - Oxford University (UK)
  - Publicado en 2015
  - International Conference on Learning Representation
  - +33K citas (Ene 2020)

Disponible en
http://www.robots.ox.ac.uk/~vgg/research/very_deep/



VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION

Karen Simonyan* **& Andrew Zisserman+**
Visual Geometry Group, Department of Engineering Science, University of Oxford
{karen,az}@robots.ox.ac.uk

ABSTRACT

In this work we investigate the effect of the convolutional network depth on its accuracy in the large-scale image recognition setting. Our main contribution is a thorough evaluation of networks of increasing depth using an architecture with very small (3×3) convolution filters, which shows that a significant improvement on the prior-art configurations can be achieved by pushing the depth to 16–19 weight layers. These findings were the basis of our ImageNet Challenge 2014 submission, where our team secured the first and the second places in the localisation and classification tracks respectively. We also show that our representations generalise well to other datasets, where they achieve state-of-the-art results. We have made our two best-performing ConvNet models publicly available to facilitate further research on the use of deep visual representations in computer vision.

Arquitecturas populares en clasificación: VGG

- 2ª posición en ILSVRC 2014
- Filtros pequeños, Redes más profundas
- 8 capas (AlexNet)
  → 16-19 capas en VGG
- Solo 3x3 CONV stride 1, pad 1 y 2x2 MAX POOL stride 2

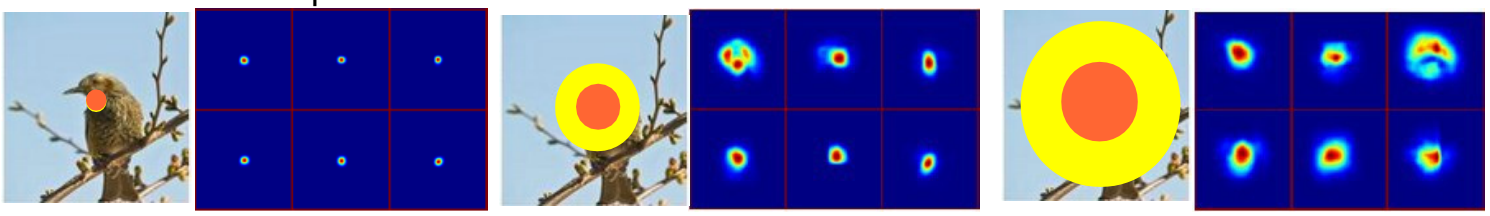
Arquitecturas populares en clasificación: VGG

- ¿Por qué utilizar filtros más pequeños? (3x3 conv)
  - Apilamiento de tres capas 3x3 conv (stride 1) tiene un efecto similar (**effective receptive field**) como una capa con filtros 7x7 conv
  - Tres capas 3x3 convs para obtener C mapas de características ($3x3x3xC^2=27C^2$ params)
    → una capa de filtros 7x7 convs ($1x7x7xC^2=49C^2$ params)

Campo receptivo estimado

El tamaño real del campo receptivo es mucho menor que el tamaño teórico

pool1 conv3 pool5



Arquitecturas populares en clasificación: VGG

- Complejidad VS performance

Model	top-5 classification error on ILSVRC-2012 (%)	
	validation set	test set
16-layer	7.5%	7.4%
19-layer	7.5%	7.3%
model fusion	7.1%	7.0%

Model	VOC-2007 (mean AP, %)	VOC-2012 (mean AP, %)	Caltech-101 (mean class recall, %)	Caltech-256 (mean class recall, %)
16-layer	89.3	89.0	91.8±1.0	85.0±0.2
19-layer	89.3	89.0	92.3±0.5	85.1±0.3
model fusion	89.7	89.3	92.7±0.5	86.2±0.3

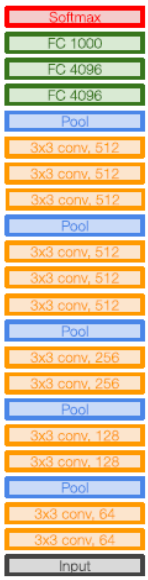
Mejor

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					

Arquitecturas populares en clasificación: VGG

(not counting biases)

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000



VGG16

TOTAL memory: 24M * 4 bytes ~= 96MB / image (for a forward pass)
TOTAL params: 138M parameters

Fuente: http://cs231n.stanford.edu/

Arquitecturas populares en clasificación: GoogleLeNet

- GoogleLeNet
  - Desarrollado por Google
  - Publicado en 2015
  - Int. Conf. on Computer Vision and Pattern Recognition
  - +19K citas (a Ene 2020)

Disponible en
https://github.com/zy314099208/CVPR2015-Going-Deeper-with-Convolutions

Going Deeper with Convolutions

Christian Szegedy[1], Wei Liu[2], Yangqing Jia[1], Pierre Sermanet[1], Scott Reed[3], Dragomir Anguelov[1], Dumitru Erhan[1], Vincent Vanhoucke[1], Andrew Rabinovich[4]
[1]Google Inc. [2]University of North Carolina, Chapel Hill
[3]University of Michigan, Ann Arbor [4]Magic Leap Inc.
[1]{szegedy,jiayq,sermanet,dragomir,dumitru,vanhoucke}@google.com
[2]wliu@cs.unc.edu, [3]reedscott@umich.edu, [4]arabinovich@magicleap.com

Abstract

We propose a deep convolutional neural network architecture codenamed Inception that achieves the new state of the art for classification and detection in the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14). The main hallmark of this architecture is the improved utilization of the computing resources inside the network. By a carefully crafted design, we increased the depth and width of the network while keeping the computational budget constant. To optimize quality, the architectural decisions were based on the Hebbian principle and the intuition of multi-scale processing. One particular incarnation used in our submission for ILSVRC14 is called GoogLeNet, a 22 layers deep network, the quality of which is assessed in the context of classification and detection.

ger and bigger deep networks, but from the synergy of deep architectures and classical computer vision, like the R-CNN algorithm by Girshick et al [6].

Another notable factor is that with the ongoing traction of mobile and embedded computing, the efficiency of our algorithms – especially their power and memory use – gains importance. It is noteworthy that the considerations leading to the design of the deep architecture presented in this paper included this factor rather than having a sheer fixation on accuracy numbers. For most of the experiments, the models were designed to keep a computational budget of 1.5 billion multiply-adds at inference time, so that the they do not end up to be a purely academic curiosity, but could be put to real world use, even on large datasets, at a reasonable cost.

In this paper, we will focus on an efficient deep neural network architecture for computer vision, codenamed Inception, which derives its name from the Network in net-

Arquitecturas populares en clasificación: Goo

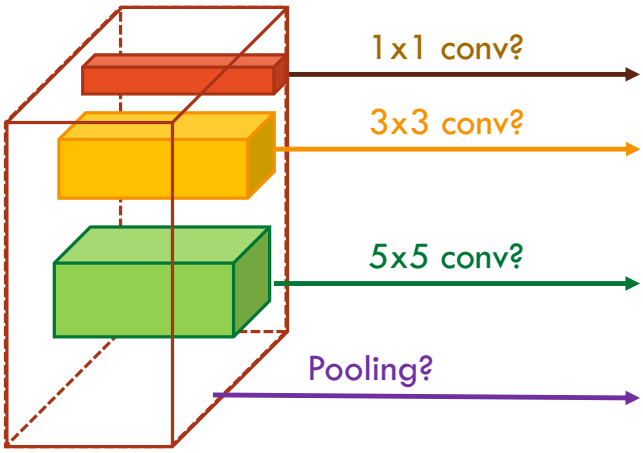
- Incremento sustancial de capas
- 22 capas en la red
- "Network-in-network"

Arquitecturas populares en clasificación: GoogleLeNet

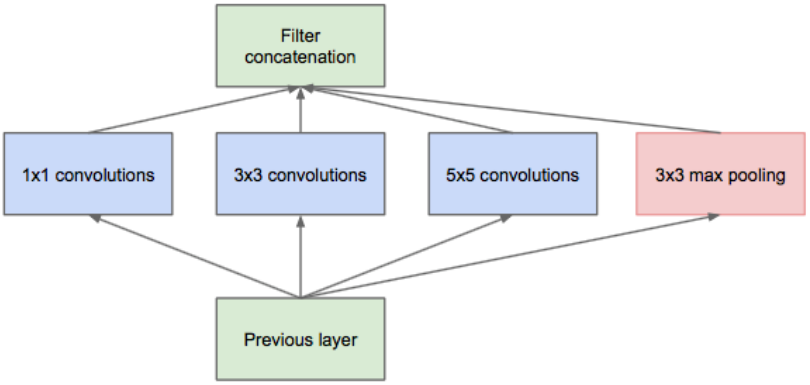
- Problemas derivados de incrementar la profundidad (*challenges of going deeper*)
  - Incremento del número de parámetros
  - Incremento del coste computacional

Arquitecturas populares en clasificación: GoogleLeNet

- Convoluciones?



Usar todas!!



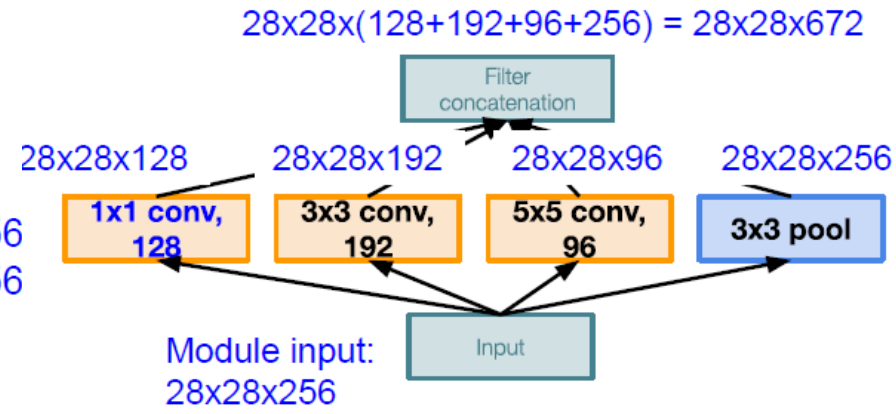
Módulo Inception (version simplificada)

(c) Mohsen Ghafoorian

Arquitecturas populares en clasificación: GoogleLeNet

- ¿Cómo es el volumen de salida en cada módulo?

Conv Ops:
[1x1 conv, 128] 28x28x128x1x1x256
[3x3 conv, 192] 28x28x192x3x3x256
[5x5 conv, 96] 28x28x96x5x5x256
Total: 854M ops

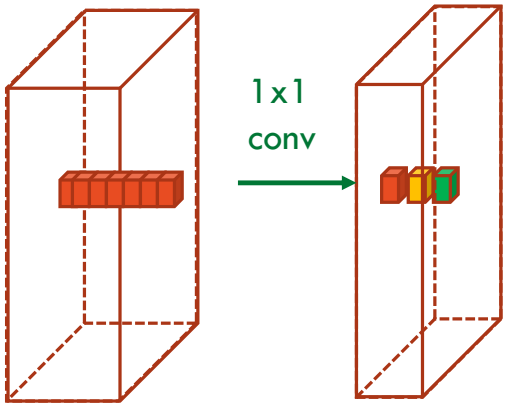


Naive Inception module

Fuente: http://cs231n.stanford.edu/

Arquitecturas populares en clasificación: GoogleLeNet

- Idea
  - ¿Tienen sentido convoluciones 1x1?
  - ¿Podemos reducir dimensionalidad (*depth*)?

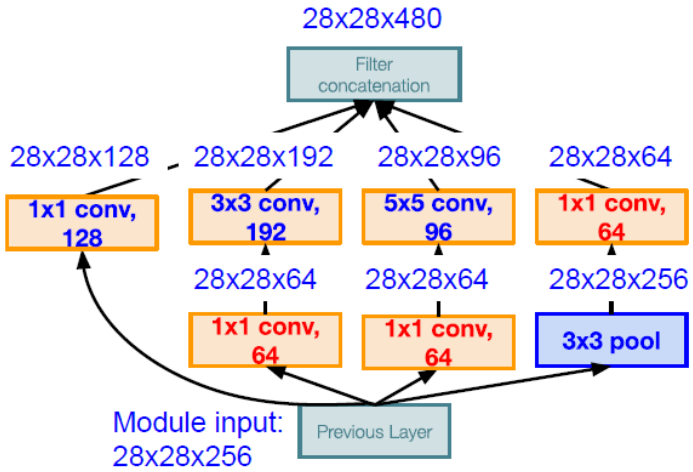


(c) Mohsen Ghafoorian

Arquitecturas populares en clasificación: GoogleLeNet

- Solución: convoluciones 1x1

Conv Ops:
[1x1 conv, 64] 28x28x64x1x1x256
[1x1 conv, 64] 28x28x64x1x1x256
[1x1 conv, 128] 28x28x128x1x1x256
[3x3 conv, 192] 28x28x192x3x3x64
[5x5 conv, 96] 28x28x96x5x5x64
[1x1 conv, 64] 28x28x64x1x1x256
Total: 358M ops

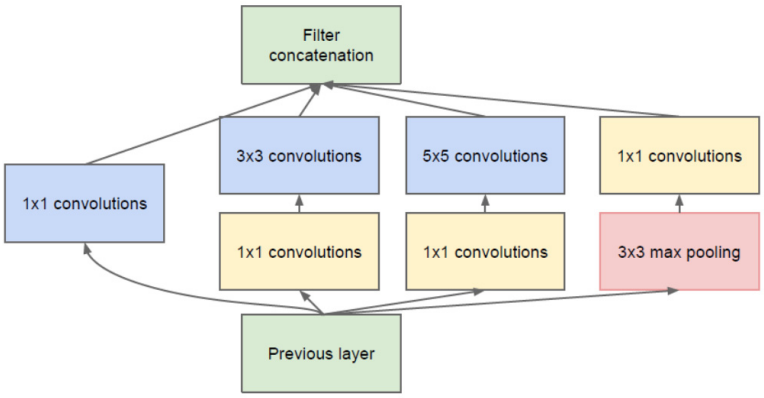


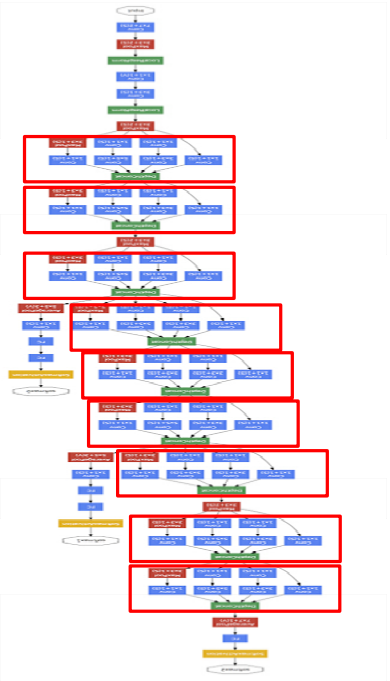
Inception module with dimension reduction

Fuente: http://cs231n.stanford.edu/

Arquitecturas populares en clasificación: GoogleLeNet

- Arquitectura completa

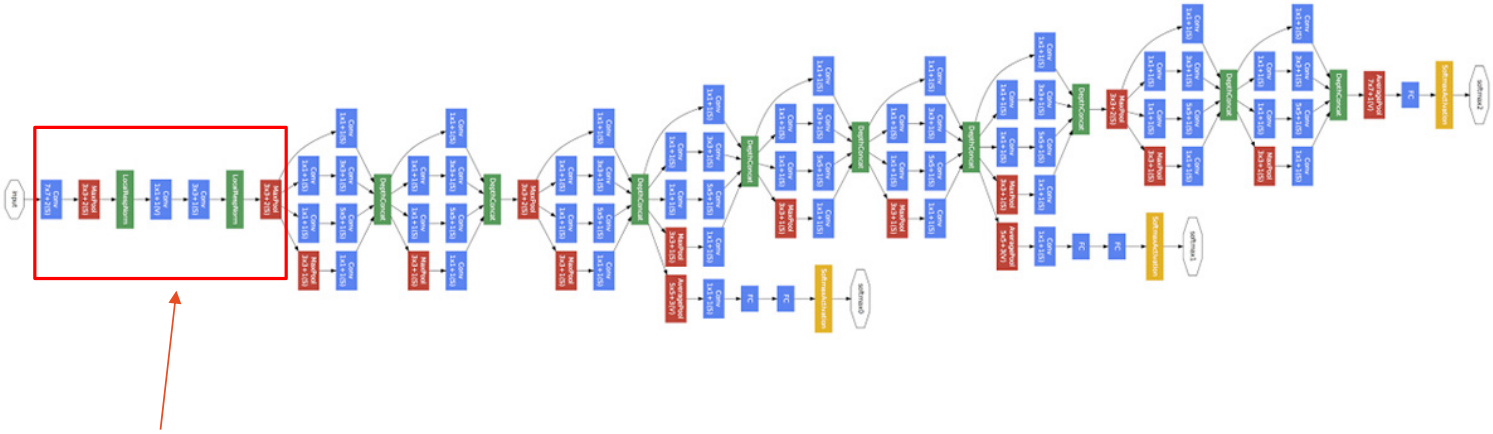




9 módulos *inception*

Arquitecturas populares en clasificación: GoogleLeNet

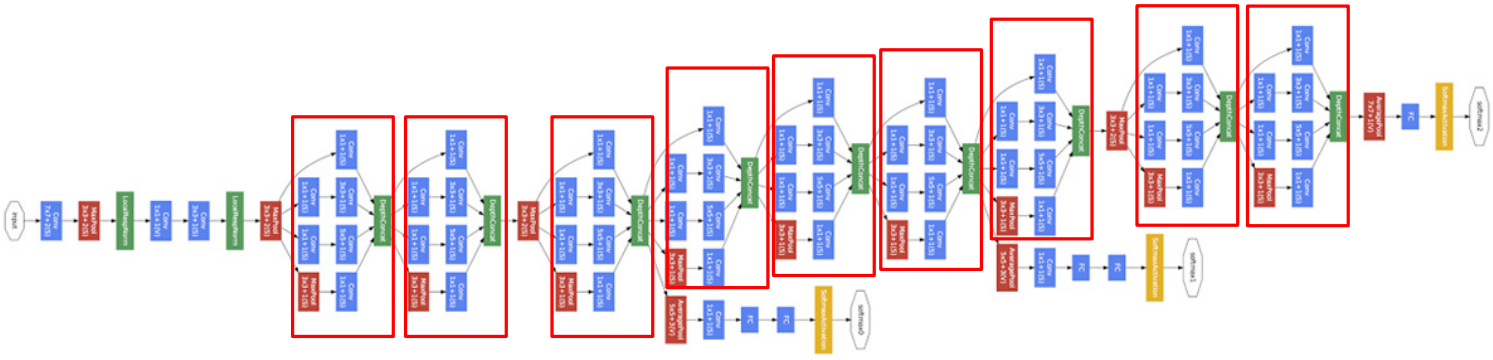
- Arquitectura completa



ENTRADA: Conv+MaxPool+Norm+2xConv+Norm+MaxPool

Arquitecturas populares en clasificación: GoogleLeNet

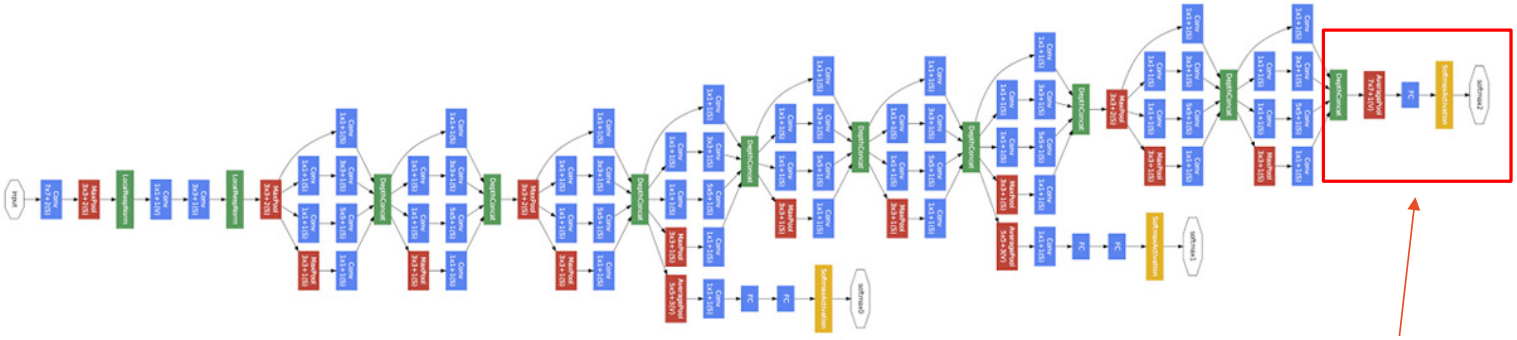
- Arquitectura completa



9 módulos *inception*

Arquitecturas populares en clasificación: GoogleLeNet

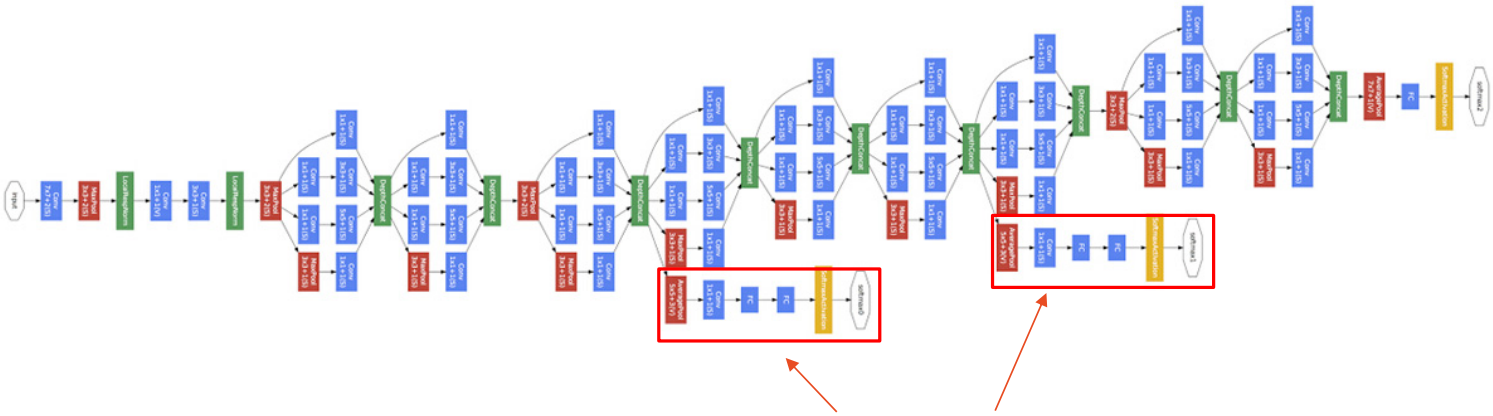
- Arquitectura completa



SALIDA: MaxPool + FC + Softmax

Arquitecturas populares en clasificación: GoogleLeNet

- Arquitectura completa



SALIDAS INTERMEDIAS: AvgPool + 1x1ConV + FC + FC + Softmax
(permiten un entrenamiento más eficiente)

Arquitecturas populares en clasificación: GoogleLeNet

- Parámetros

Solo 5M parámetros!
(12x menos que AlexNet,
27x menos que VGG!)
¿Cómo es posible?

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Arquitecturas populares en clasificación: GoogleLeNet

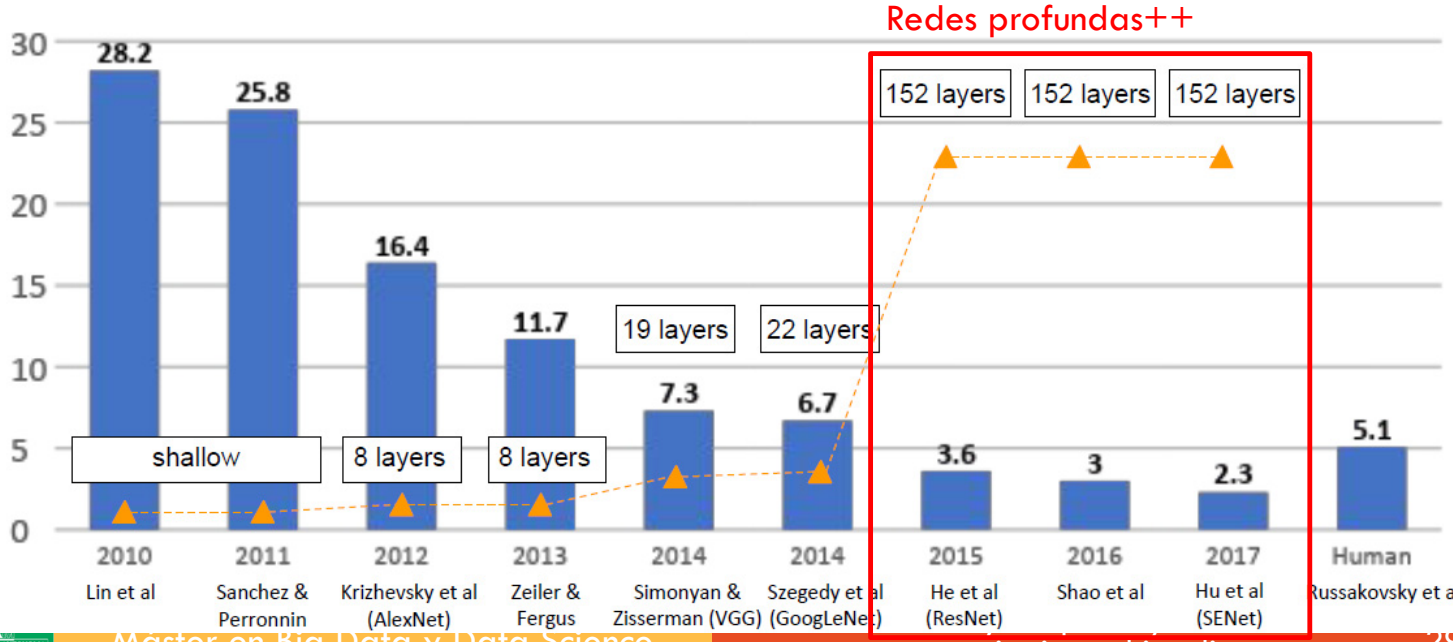
- Parámetros de VGG Net

INPUT: [224x224x3] memory: 224*224*3=150K params: 0 (not counting biases)
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

ConvNet Configuration			
B	C	D	
13 weight layers	16 weight layers	16 weight layers	19
put (224 × 224 RGB image)			
conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	cc cc
maxpool			
conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	co co
maxpool			
conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	co co co **co**
maxpool			
conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	co co co **co**
maxpool			
conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	co co co **co**
maxpool			
FC-4096			
FC-4096			
FC-1000			
soft-max			

Arquitecturas populares en clasificación

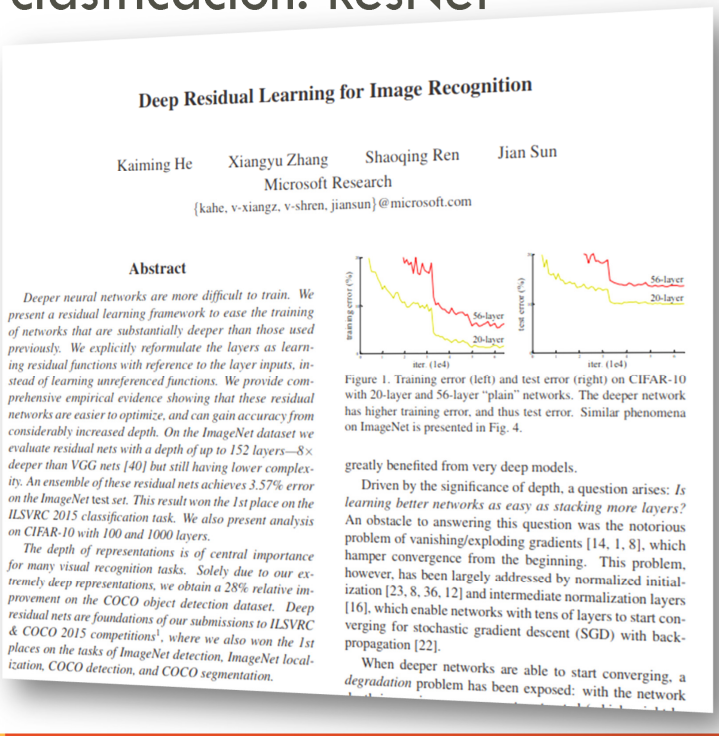
- ImageNet Large Scale Visual Recognition Challenge (ILSVRC)



Arquitecturas populares en clasificación: ResNet

- ResNet
  - Desarrollado por Microsoft
  - Publicado en 2016
  - International Conference on Computer Vision and Pattern Recognition
  - +40K citas (a Ene 2020)

Disponible at https://github.com/KaimingHe/deep-residual-networks



Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

Abstract

Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset we evaluate residual nets with a depth of up to 152 layers—8× deeper than VGG nets [40] but still having lower complexity. An ensemble of these residual nets achieves 3.57% error on the ImageNet test set. This result won the 1st place on the ILSVRC 2015 classification task. We also present analysis on CIFAR-10 with 100 and 1000 layers.

The depth of representations is of central importance for many visual recognition tasks. Solely due to our extremely deep representations, we obtain a 28% relative improvement on the COCO object detection dataset. Deep residual nets are foundations of our submissions to ILSVRC & COCO 2015 competitions[1], where we also won the 1st places on the tasks of ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation.

Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

greatly benefited from very deep models.

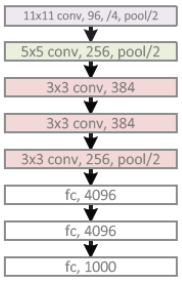
Driven by the significance of depth, a question arises: *Is learning better networks as easy as stacking more layers?* An obstacle to answering this question was the notorious problem of vanishing/exploding gradients [14, 1, 8], which hamper convergence from the beginning. This problem, however, has been largely addressed by normalized initialization [23, 8, 36, 12] and intermediate normalization layers [16], which enable networks with tens of layers to start converging for stochastic gradient descent (SGD) with back-propagation [22].

When deeper networks are able to start converging, a *degradation* problem has been exposed: with the network

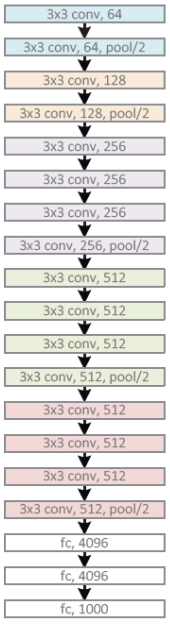
Arquitecturas populares en clasificación: ResNet

- Revolución de profundidad (*going deeper again*)

AlexNet, 8 layers (ILSVRC 2012)



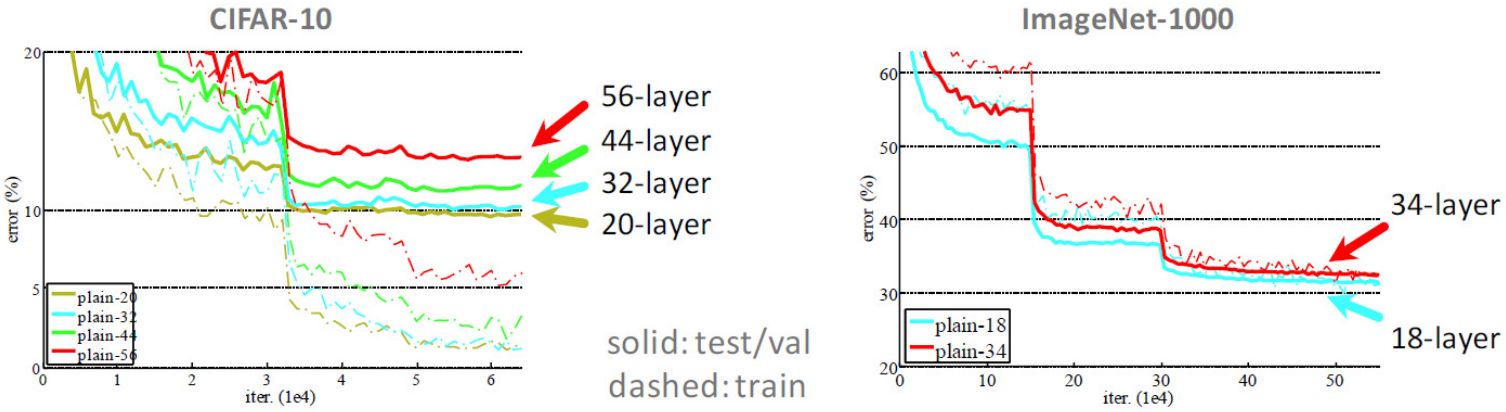
VGG, 19 layers (ILSVRC 2014)



ResNet, 152 layers (ILSVRC 2015)

Arquitecturas populares en clasificación: ResNet

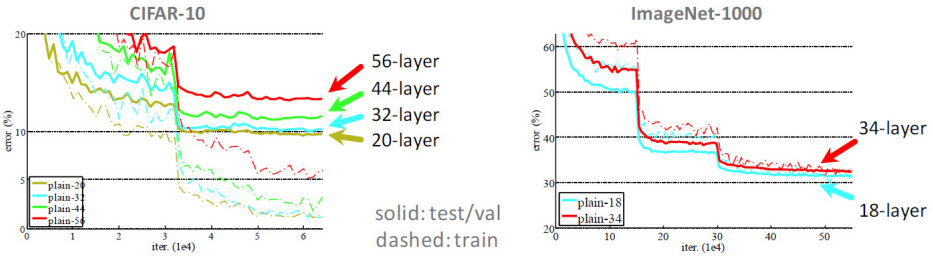
- Revolución de profundidad (*curse of dimensionality*)



¿Mejor aprendizaje simplemente añadiendo más capas?

K. He, et al. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385, CVPR 2016

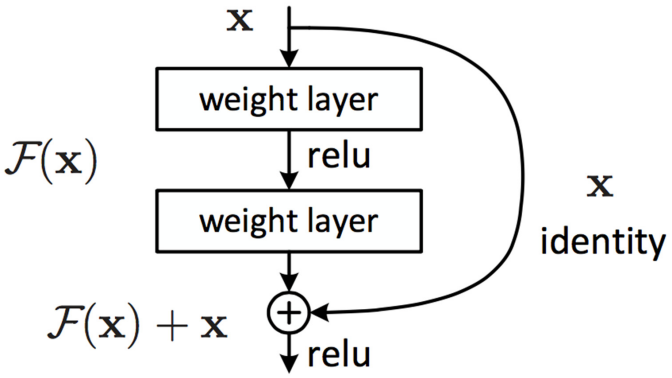
Arquitecturas populares en clasificación: ResNet



¿Por qué la diferencia entre el error de entrenamiento/test es más alto para CIFAR-10 que para el dataset de imagenet-1000?

Arquitecturas populares en clasificación: ResNet

- Ganador ILSVRC 2015
- Bloques residuales (*Residual blocks*)
  - Adelantar el input
  - Permite entrenar redes con muchas capas

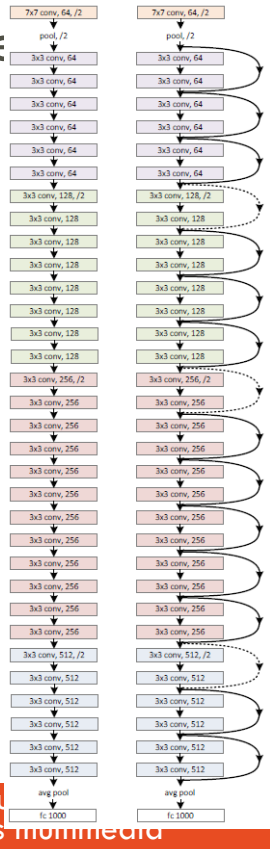


K. He, et al. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385, CVPR 2016

Arquitecturas populares en clasificación: Re

- Ganador ILSVRC 2015
- Bloques residuales (*Residual blocks*)
- Diseño simple
  - Casi todo son 3x3 conv
  - Sin *Fully Connected* ni *dropout*

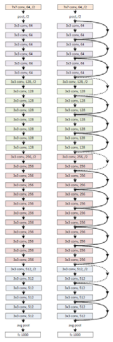
Red "plana"

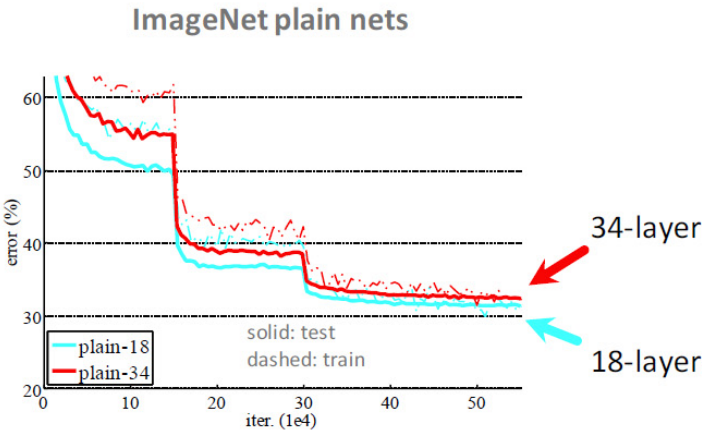


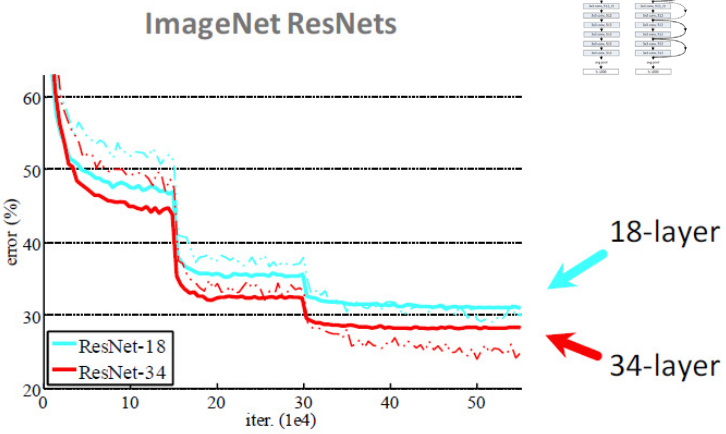
ResNet

Arquitecturas populares en clasificación: ResNet

- Deep ResNets pueden entrenarse mejor







K. He, et al. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385, CVPR 2016

Agenda: Multimedia (imagen, video)

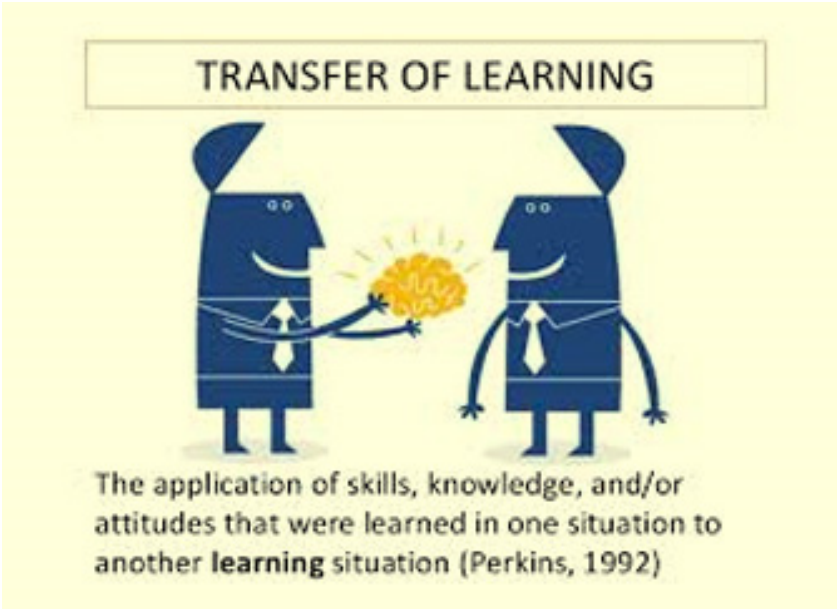
Temas avanzados

- Arquitecturas populares en clasificación
- *Transfer Learning*
  - Introducción
  - Impacto
  - Concepto
  - *Transfer learning* en CNNs
  - Estrategias
  - Casos de estudio
  - Ejemplos

INTRODUCCION

https://youtu.be/0YhJxJZOWBw

INTRODUCCION

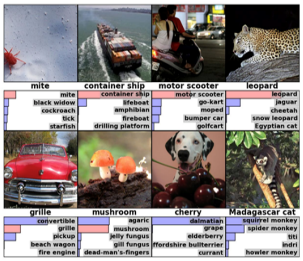


INTRODUCCION

ImageNet Challenge

IMAGENET

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



5000 object classes and concepts
16M images
https://www.vision.ee.ethz.ch/webvision/

	VGGNet	DeepVideo	GNMT
Used For	Identifying Image Category	Identifying Video Category	Translation
Input	Image	Video	English Text
Output	1000 Categories	47 Categories	French Text
Parameters	140M	~100M	380M
Data Size	1.2M Images with assigned Category	1.1M Videos with assigned Category	6M Sentence Pairs, 340M Words
Dataset	ILSVRC-2012	Sports-1M	WMT'14

INTRODUCCION

Se necesitan grandes cantidades de datos para entrenar/utilizar redes neuronales convolucionales (CNNs)

¿Datasets pequeños?

1500 imágenes entrenamiento / 1500 imágenes test

INTRODUCCION

Se necesitan grandes cantidades de datos para entrenar/utilizar redes neuronales convolucionales (CNNs)

¿Datasets pequeños?

↓

¿Existe alternativa?

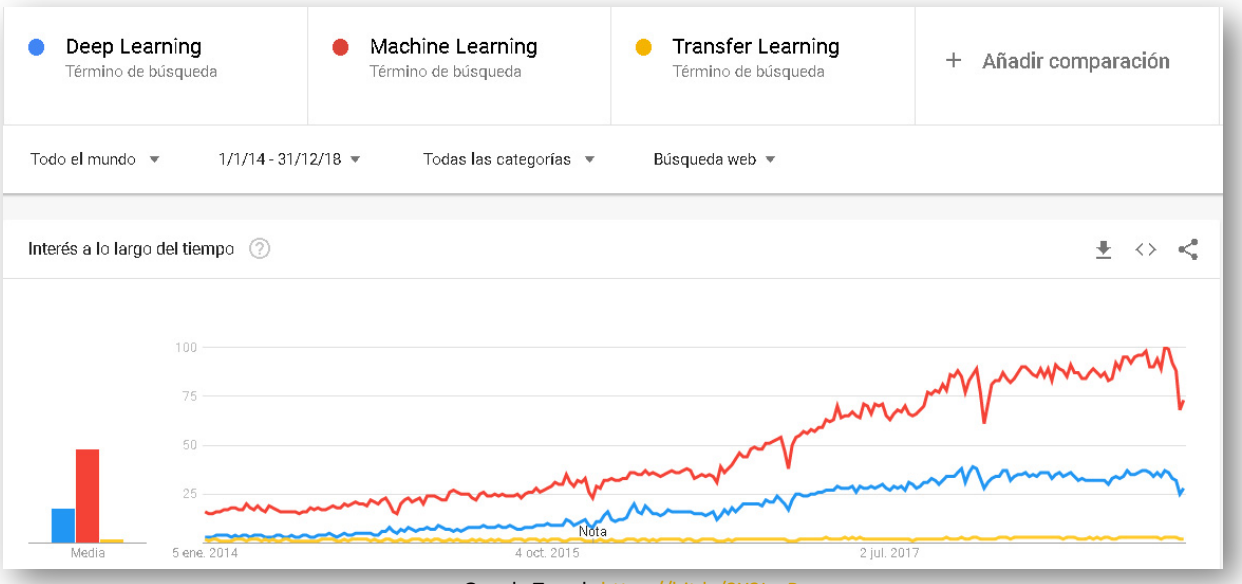
↓

Si, transferencia del conocimiento (*transfer learning*)

↓

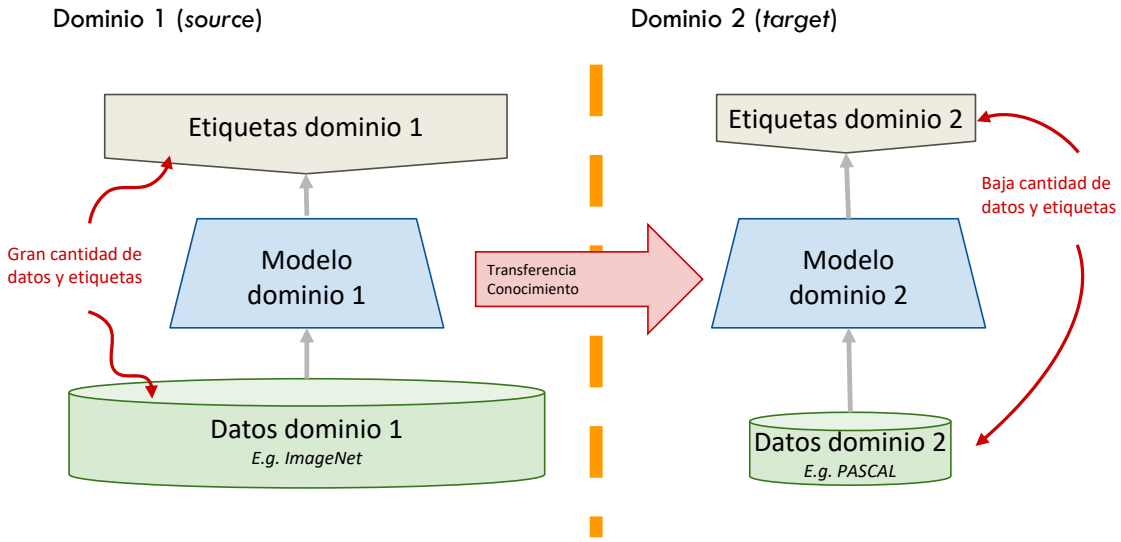
Utilizar CNNs como extractores de características

IMPACTO



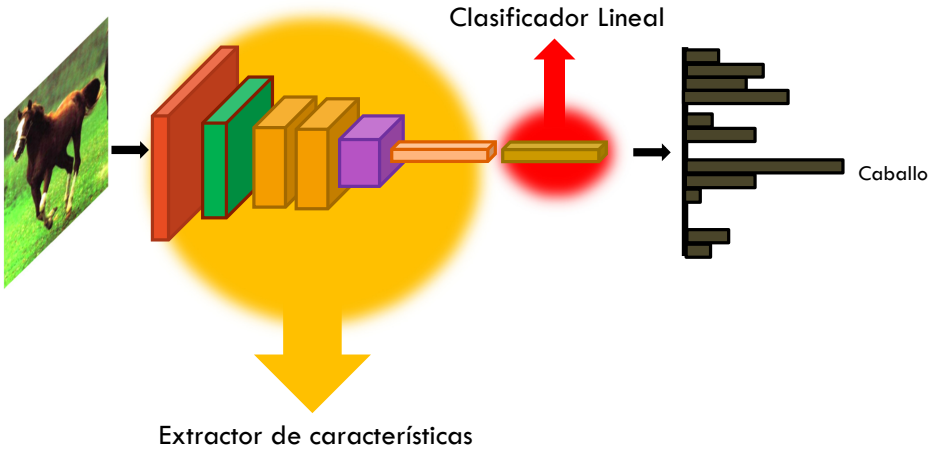
Google Trends https://bit.ly/2X2IspR

CONCEPTO



TRANSFER LEARNING EN CNNs

- Descomposición funcional de una red CNN

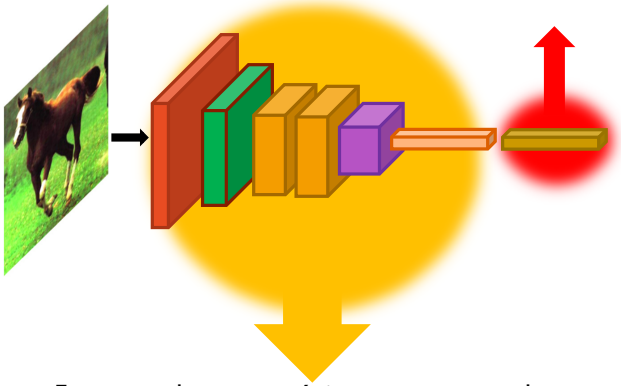


Fuente: http://www.cs.cornell.edu/courses/cs4670/2019sp/

TRANSFER LEARNING EN CNNs

¿Qué transferir a un nuevo problema de clasificación?

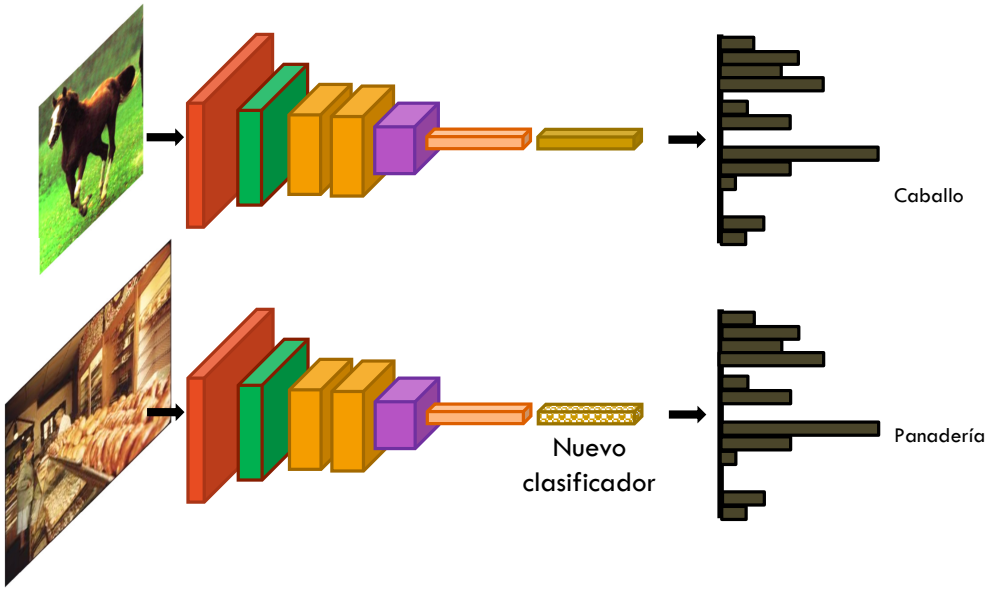
- Idea:
  - Congelar (la mayoría de) parámetros del extractor de características
  - Reentrenar el clasificador



Extractor de características pre-entrenado

Fuente: http://www.cs.cornell.edu/courses/cs4670/2019sp/

TRANSFER LEARNING EN CNNs



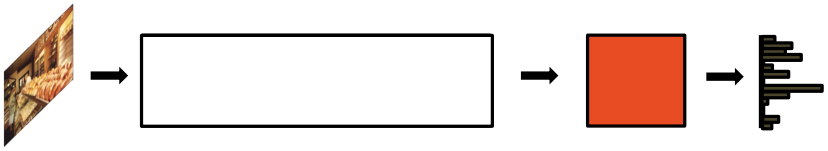
Fuente: http://www.cs.cornell.edu/courses/cs4670/2019sp/

ESTRATEGIAS

Estrategia 1
Entrenar la red completa

Estrategia 2
Entrenar clasificador y parte del extractor de características

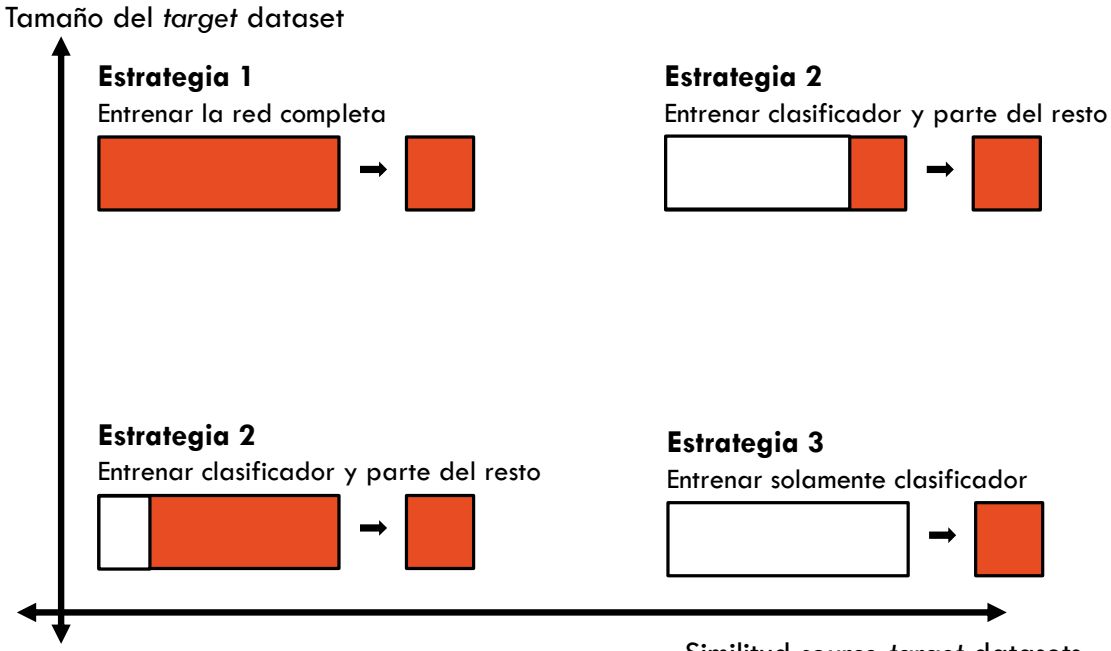
Estrategia 3
Entrenar solamente el clasificador



Parámetros fijos

Parámetros a entrenar

ESTRATEGIAS

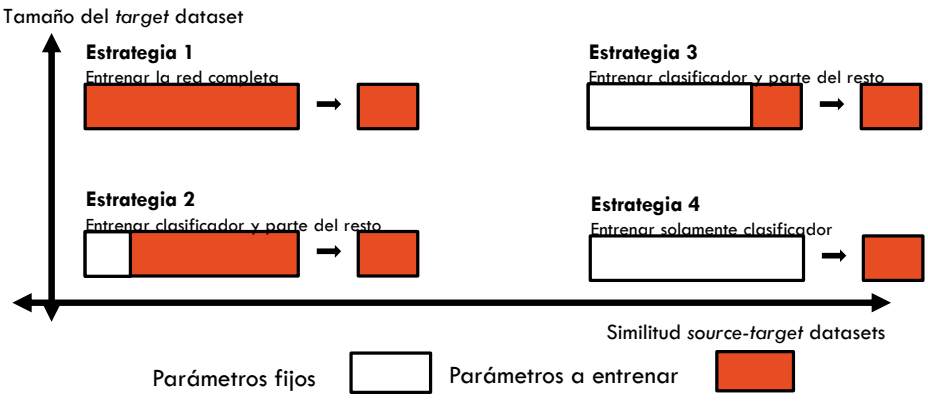


CASO DE ESTUDIO

- Reconocimiento de escenas:
  - 15 categorías
  - 1500 imágenes entrenamiento / 1500 imágenes test

- Sea el dataset de escenas mencionado con la partición mencionada en la diapositiva anterior, que estrategia de *Transfer Learning* sería más adecuada aplicar:

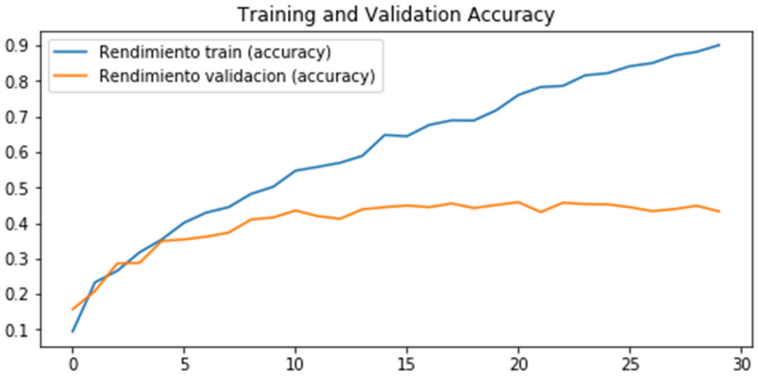
a) Estrategia 1 b) Estrategia 2 c) Estrategia 3 d) Estrategia 4

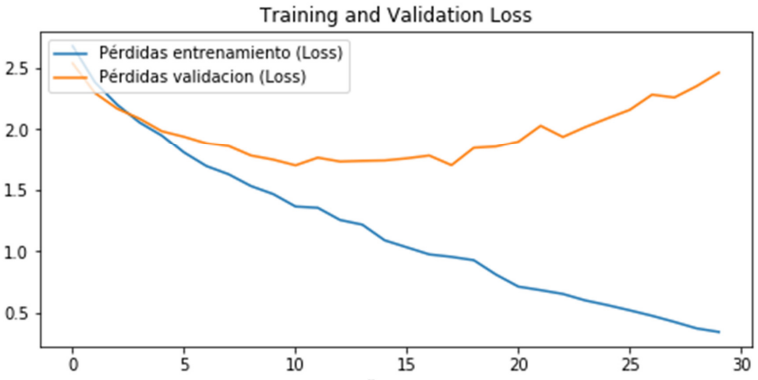


CASO ESTUDIO

- Resultados iniciales

```
Model: "Red ejemplo"
_________________________________________________________________
Layer          Output_shape Param #
=================================================================
conv2d           (None, 30, 30, 6) 168
_________________________________________________________________
avg_pooling2d (None, 15, 15, 6) 0
_________________________________________________________________
conv2d_1       (None, 13, 13, 16) 880
_________________________________________________________________
avg_pooling2d (None, 6, 6, 16) 0
_________________________________________________________________
flatten          (None, 576) 0
_________________________________________________________________
dense            (None, 120) 69240
_________________________________________________________________
dense_1          (None, 84) 10164
_________________________________________________________________
dense_2        (None, 15) 1275
=================================================================
Total params: 81,727
Trainable params: 81,727
Non-trainable params: 0
_________________________________________________________________
```





Resultado final (test): 0.4328

CASO ESTUDIO

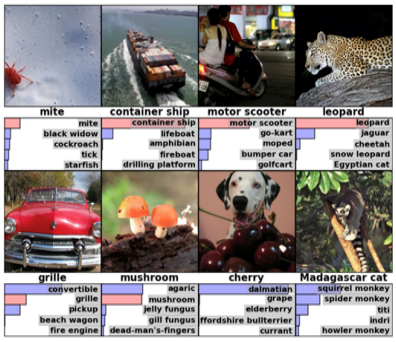
- Arquitectura VGG16 pre-entrenada en Imagenet

Imagen
Conv-64
Conv-64
MaxPool
Conv-
Conv-
MaxPool
Conv-
Conv-
Conv-
MaxPool
Conv-
Conv-
Conv-
MaxPool
Conv-
Conv-
Conv-
MaxPool
FC-4096
FC-4096
FC-1000 +softmax

ImageNet Challenge

IMAGENET

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

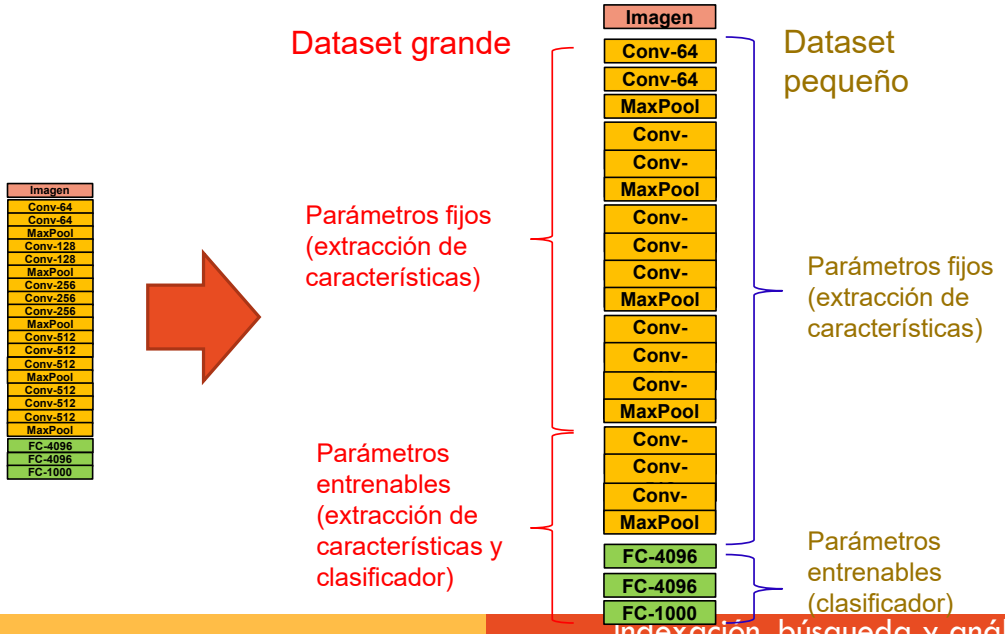


92.7% top-5 rendimiento (accuracy) en test

143,667,240 parámetros

CASO ESTUDIO

- Opciones para transferencia con datasets similares

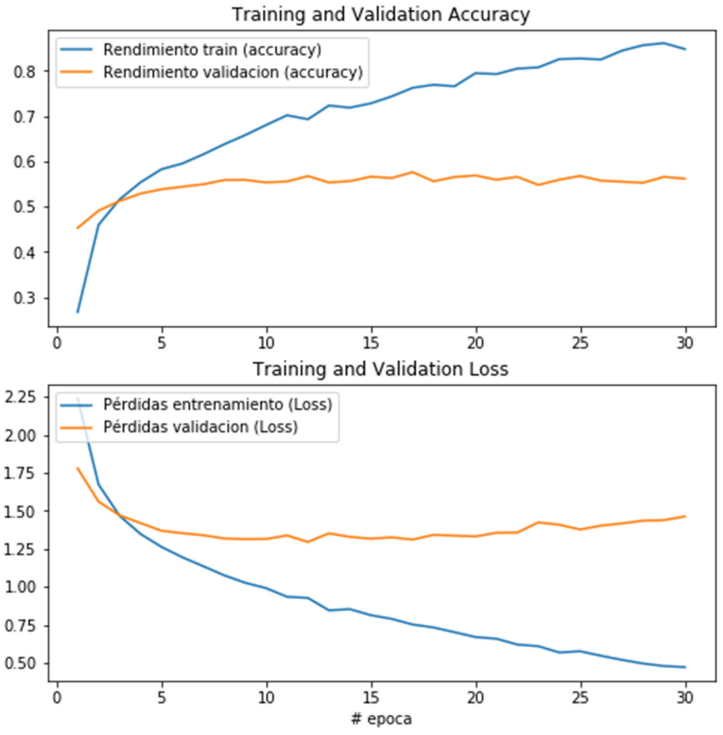


CASO ESTUDIO

- Resultados finales 1

```
Model: "Adaptada VGG19 a Scenes15"
_________________________________________________________________
Layer     Output_shape Param #
=================================================================
vgg16*        (None, 1, 1, 512)
14714688
_________________________________________________________________
dense_1       (None, 120) 61560
_________________________________________________________________
dense_2    (None, 15) 1815
=================================================================
Total params: 14,778,063
Trainable params: 63,375
Non-trainable params: 14,714,688
_________________________________________________________________

*vgg16 sin FC-4096, FC-4096 y FC-1000
```

+30% rendimiento con Transfer Learning

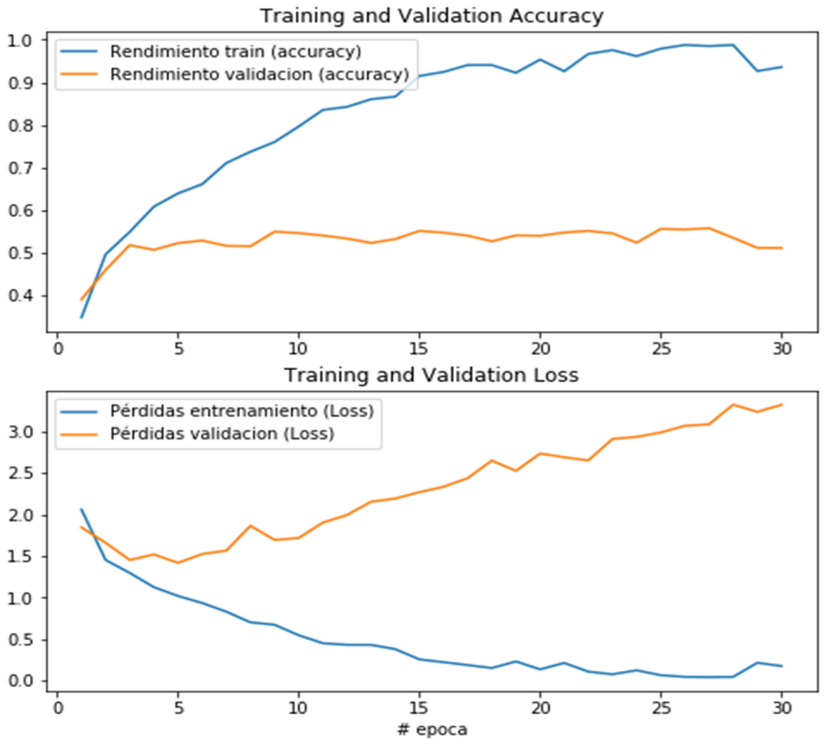


Resultado final (test): 0.5615

CASO ESTUDIO

- Resultados finales 2

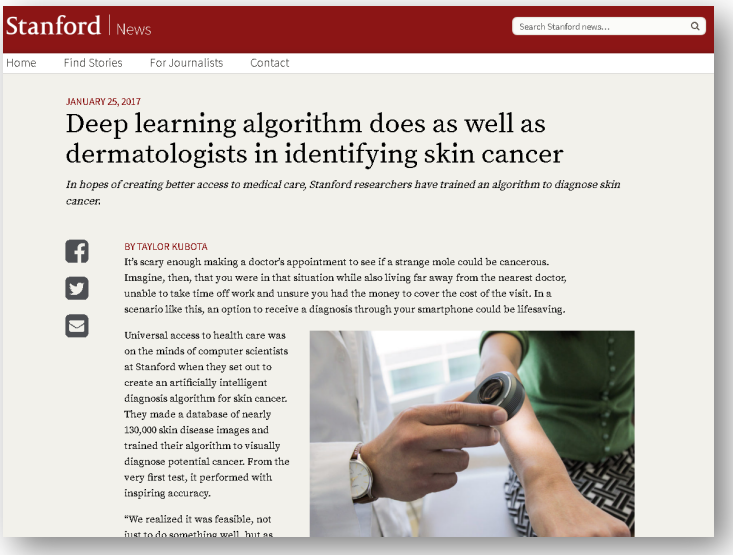
```
Model: "Adaptada VGG19 a Scenes15"
_________________________________________________________________
Layer      Output_shape Param #
=================================================================
vgg16*         (None, 1, 1, 512)
14714688
_________________________________________________________________
dense_1        (None, 4096) 2101248
_________________________________________________________________
dense_2    (None, 4096) 16781312
_________________________________________________________________
dense_3    (None, 15) 61455
=================================================================
Total params: 33,658,703
Trainable params: 18,944,015
Non-trainable params: 14,714,688
_________________________________________________________________

*vgg16 sin FC-4096, FC-4096 y FC-1000
```



Resultado final (test): 0.5114

EJEMPLOS

- Transfer Learning para detección de cáncer de piel:



https://news.stanford.edu/2017/01/25/artificial-intelligence-used-identify-skin-cancer/

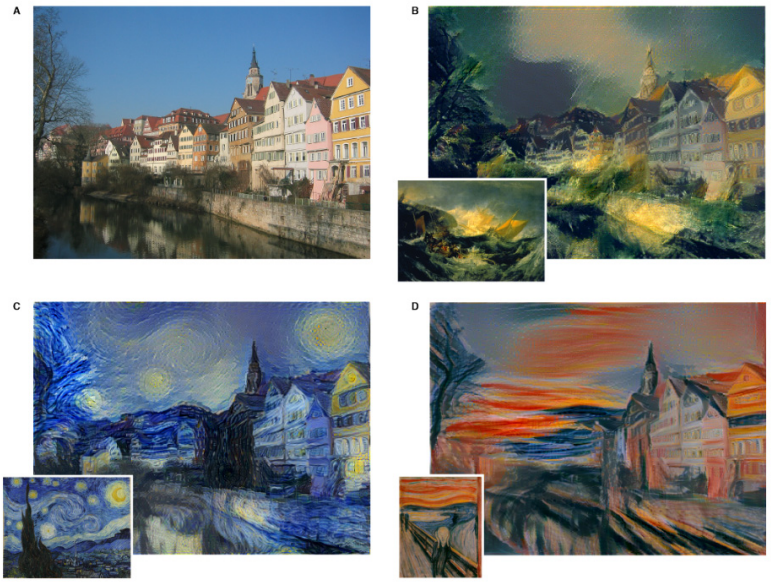
EJEMPLOS

- Transfer Learning para mejora de imágenes

https://github.com/alexjc/neural-enhance

EJEMPLOS

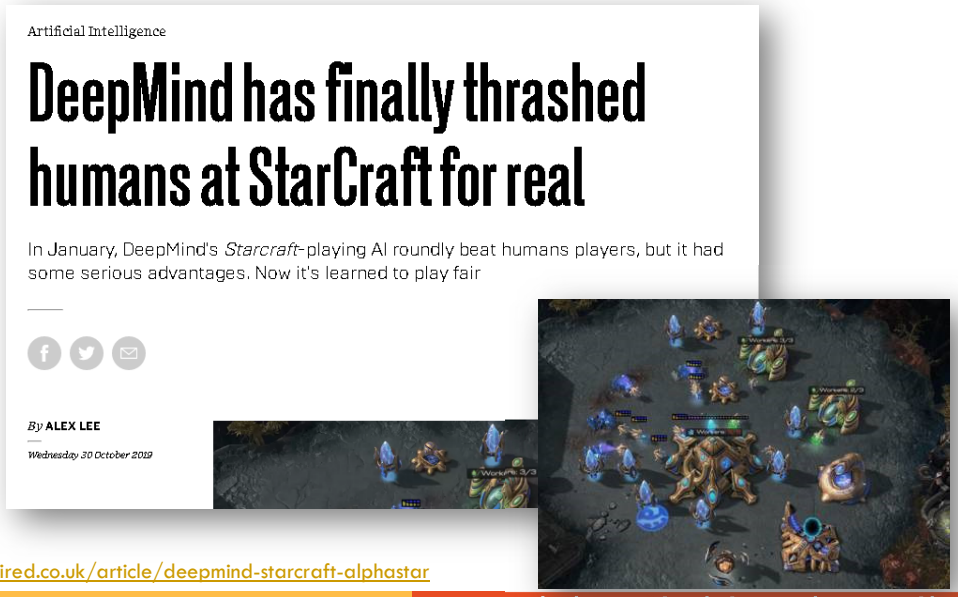
- Transfer Learning para copiar el "estilo artístico"



https://bit.ly/2N7xtqh

EJEMPLOS

- Transfer Learning para emular "jugadores" de videojuegos



https://www.wired.co.uk/article/deepmind-starcraft-alphastar

REFERENCIAS

- Básico:
  - Tamoghna Ghosh, Raghav Bali, Dipanjan Sarkar, "Hands-On Transfer Learning with Python", Packt 2018 (accessible desde el campus UAM en https://learning.oreilly.com/library/view/hands-on-transfer-learning)