Taller de Aprendizaje Automático

Redes Neuronales Convolucionales Profundas

Instituto de Ingeniería Eléctrica
Facultad de Ingeniería

UNIVERSIDAD
DE LA REPÚBLICA
URUGUAY

1. Principales característcas de las redes convolucionales
2. Arquitecturas de redes convolucionales profundas
   - LeNet
   - AlexNet y ZFNet
   - VGG
   - GoogLeNet
   - ResNet
   - Xception
   - SENet
3. Utilización de modelos pre-entrenados y Transferencia de Aprendizaje
4. Aplicación: clasificación y localización

Principales característcas de las redes convolucionales

- Reducen la cantidad de parámetros en comparación con las redes densas
  - Los campos receptivos son locales
  - Se comparten pesos
- Originalmente compuestas por dos bloques
  - capas de convolución
    - Los filtros de aprenden durante el entrenamiento
    - La cantidad de parámetros aprendidos es: $(F \times F \times C_{in} + 1) \times C_{out}$
    - El tamaño del volumen de salida está dado por: $(N - F + 2P)/S + 1$
  - capa de pooling
    - permiten capturar dependencias más globales en forma eficiente

Lenet-5 - 1998 (poniendo todo junto)*

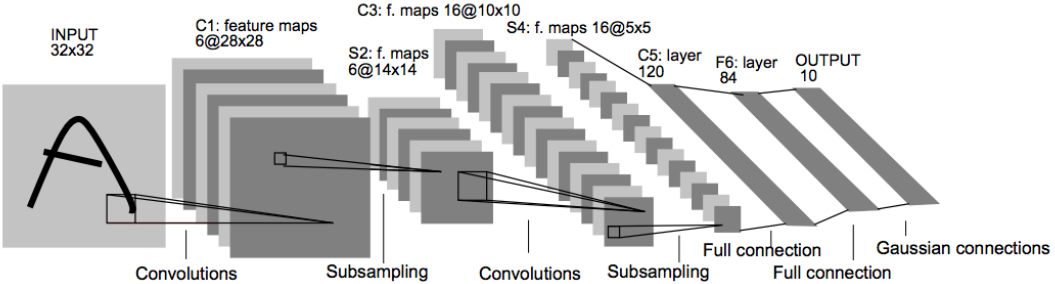


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

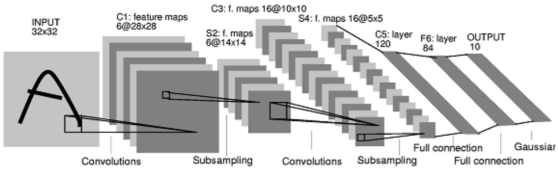
+info: https://yohanes.gultom.me/understanding-lenet-lecun-1998

*Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proc. of the IEEE*, vol. 86(11), pp. 2278–2324, 1998

Evolución de las arquitecturas

1998

LeCun et al.



# of transistors

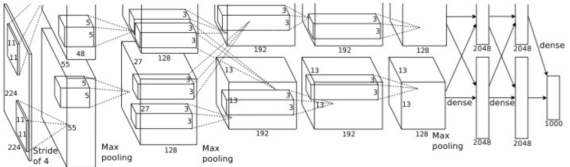
10^6

# of pixels used in training

10^7 NIST

2012

Krizhevsky et al.



# of transistors

10^9

GPUs

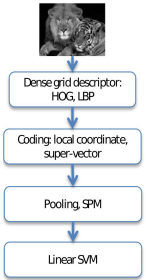
# of pixels used in training

10^{14} IMAGENET

Evolución de las arquitecturas

2011

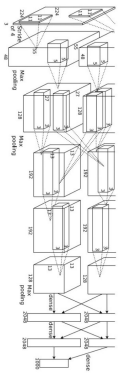
NEC-UIUC



[Lin 2011]

2012

SuperVision



[Krizhevsky 2012]

2014

GoogLeNet

[Szegedy 2014]

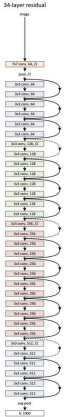
2014

VGG

[Simonyan 2014]

2015

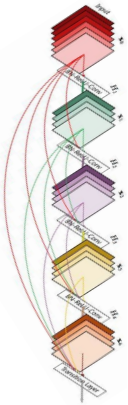
Res-nets



[He 2015]

2017

DenseNet



[Huang 2017]

ImageNet Large Scale Visual Recognition Competition

Competencia de clasificación de imágenes

- 1000 categorías de imágenes; 1.4 millones de imágenes de entrenamiento
- Respuesta correcta entre las primeras 5 imágenes.
- Krizhevsky, Sutskever, Hinton, 2012 [Supervision/AlexNet].

2012 Teams	%error
Supervision (Toronto)	15.3
ISI (Tokyo)	26.1
VGG (Oxford)	26.9
XRCE/INRIA	27.0
UvA (Amsterdam)	29.6
INRIA/LEAR	33.4

2013 Teams	%error
Clarifai (NYU spinoff)	11.7
NUS (singapore)	12.9
Zeiler-Fergus (NYU)	13.5
A. Howard	13.5
OverFeat (NYU)	14.1
UvA (Amsterdam)	14.2
Adobe	15.2
VGG (Oxford)	15.2
VGG (Oxford)	23.0

2014 Teams	%error
GoogLeNet	6.6
VGG (Oxford)	7.3
MSRA	8.0
A. Howard	8.1
DeeperVision	9.5
NUS-BST	9.7
TTIC-ECP	10.2
XYZ	11.2
UvA	12.1

Técnicas que usan CNN - Otros métodos

ImageNet Large Scale Visual Recognition Competition

Competencia de clasificación de imágenes

- 1000 categorías de imágenes; 1.4 millones de imágenes de entrenamiento
- Respuesta correcta entre las primeras 5 imágenes.
- Krizhevsky, Sutskever, Hinton, 2012 [Supervision/AlexNet].

Arquitecturas de redes convolucionales profundas

Veremos las siguientes arquitecturas:

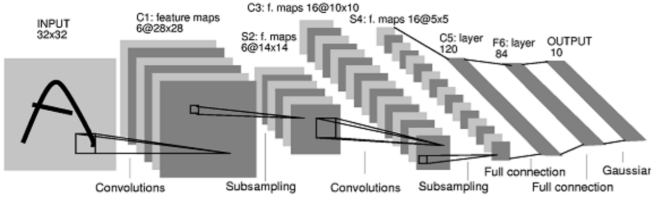
- LeNet
- AlexNet
- ZFNet
- VGG
- GoogLeNet
- ResNet
- SENet

Quedan por fuera otras arquitecturas relevantes como: DenseNet, WideNets, ResNeXt, Stochastic Depth, FractalNets, NASNet, ...

Antes del 2012...

1998

LeCun et al.



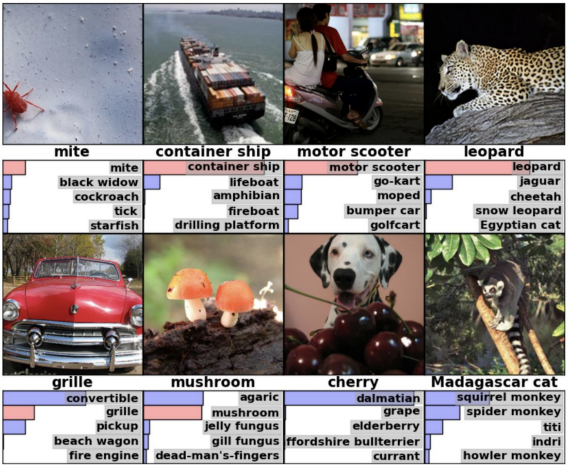
LeNet-5

- Capas de convolución
  - campos receptivos locales
  - compartir pesos
- Capas de pooling
- Capas *fully connected*

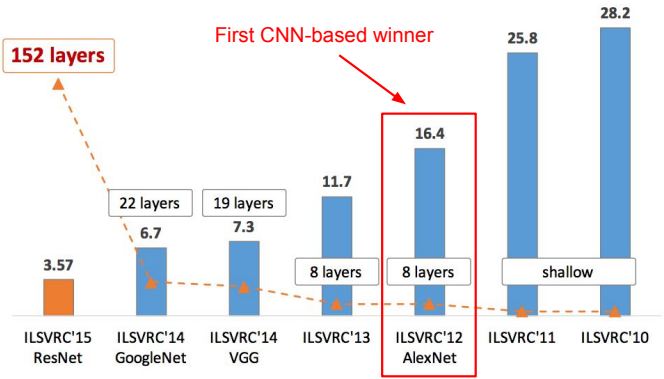
- CONV - POOL - CONV - POOL - FC - FC - FC
- Filtros de convolución 5x5, con paso 1
- Capas de submuestreo (Pooling) de 2x2, con paso 2
- Activaciones: tanh en capas ocultas, RBF en capa de salida

ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

- El objetivo es clasificar una imagen en una de 1000 categorías posibles
- El algoritmo debe devolver las cinco categorías más probables

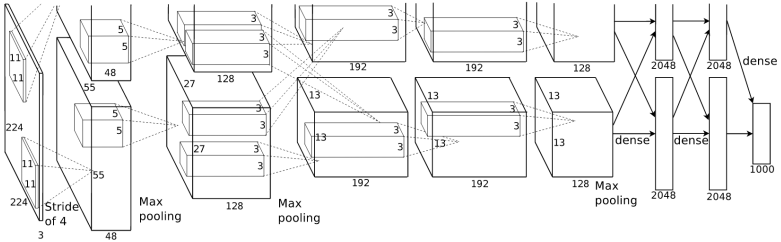


ImageNet Large Scale Visual Recognition Challenge (ILSVRC)



Slides tomadas de cs231n (Stanford) - Fei-Fei Li & Justin Johnson & Serena Yeung

AlexNet*



CONV1 - POOL1 - CONV2 - POOL2 - CONV3 - CONV4 - CONV5 - POOL3 - FC6 - FC7 - FC8

CONV1: 96 filtros de 11x11, paso = 4
Tamaño de entrada: 227x227x3
Tamaño de salida: 55x55x96
Número de parámetros: 11x11x3x96 = 34848
(sin contar bias)
FLOPS: (55x55x96)x(11x11x3 + 1) = 105Mflops

POOL1: regiones de 3x3, paso = 2
Tamaño de entrada: 55x55x96
Tamaño de salida: 27x27x96

*A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, 2012

AlexNet: arquitectura

Capa	Tamaño	Parámetros	Operaciones
INPUT	227x227x3		
CONV1	55x55x96	35K	105Mflops
POOL1	27x27x96		
CONV2	27x27x256	307K	223Mflops
POOL2	13x13x256		
CONV3	13x13x384	884K	149Mflops
CONV4	13x13x384	1.3M	224Mflops
CONV5	13x13x256	442K	74Mflops
POOL3	6x6x256		
FC	4096	37M	37Mflops
FC	4096	16M	16Mflops
FC	1000	4M	4Mflops

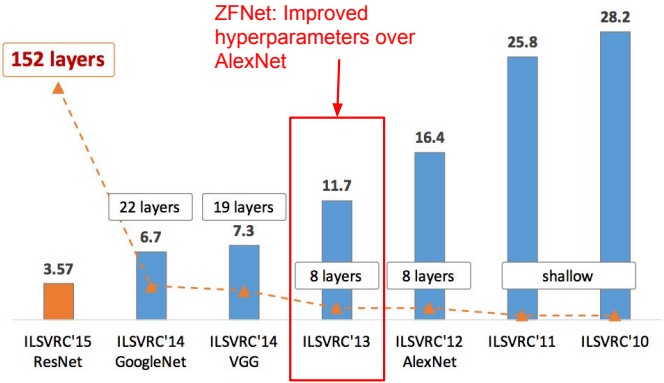
Novedades:

- **Activaciones ReLU** en capas ocultas, Softmax en capa de salida
- ***Competitive normalization***: Local Response Normalization (LRN) luego de los ReLU de CONV3 y CONV5

AlexNet: entrenamiento

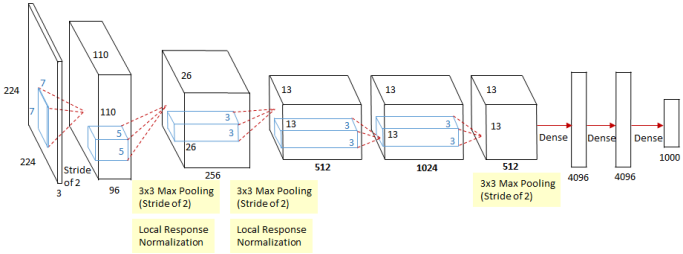
- *REvoLUción* en el campo de las redes neuronales
- Detalles de entrenamiento
  - Uso intensivo de técnicas de aumentado de datos
  - Dropout 0.5
  - Batch size 128
  - SGD Momentum 0.9
  - Learning rate: 1e-2 incial, luego dividido 10 cuando el error de validación se estanca
  - L2 weight decay 5e-4

ImageNet Large Scale Visual Recognition Challenge (ILSVRC)



Slides tomadas de cs231n (Stanford) - Fei-Fei Li & Justin Johnson & Serena Yeung

ZFNet*



Idéntica a AlexNet con las siguientes modificaciones:

- CONV1: filtros de 7x7, paso=2 en lugar de 11x11 y paso=4
- CONV2: paso de 2 en lugar de 4
- CONV3/4/5: 512/1024/512 filtros en lugar de 384/384/256

1.45M de parámetros menos que AlexNet

*M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision – ECCV 2014*, pp. 818–833, Springer, 2014

ZFNet

Más allá de bajar el error casi 5%, Zeiler y Fergus proponen:

- Una técnica de visualización de los mapas de activación que permite ganar en comprensión sobre cómo funcionan las redes profundas.

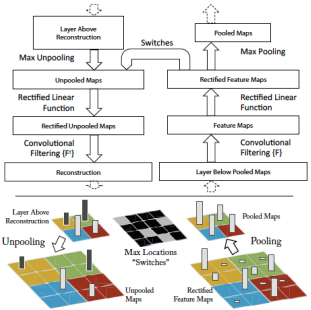
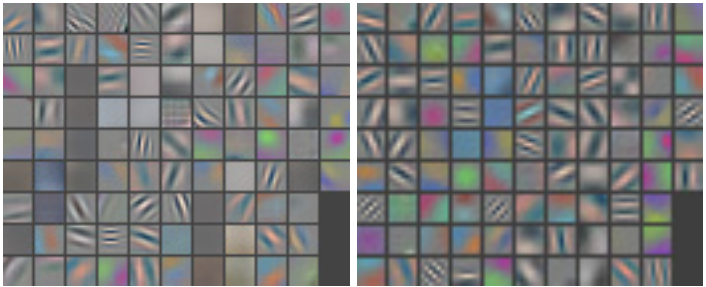


Fig. 1. Top: A deconvnet layer (left) attached to a convnet layer (right). The deconvnet will reconstruct an approximate version of the convnet features from the layer beneath. Bottom: An illustration of the unpooling operation in the deconvnet, using *switches* which record the location of the local max in each pooling region (colored zones) during pooling in the convnet. The black/white bars are negative/positive activations within the feature map.

ZFNet

Más allá de bajar el error casi 5%, Zeiler y Fergus proponen:

- Usar las visualizaciones para identificar los puntos débiles de AlexNet y obtener una mejor arquitectura.



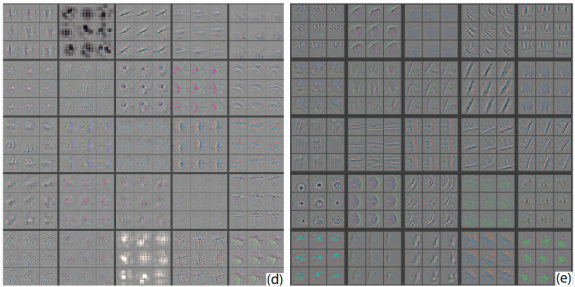
Features de la primera capa de AlexNet y ZFNet.

Reduciendo los filtros de 11×11 a 7×7 se reducen las neuronas muertas y se logran filtros más distintivos.

ZFNet

Más allá de bajar el error casi 5%, Zeiler y Fergus proponen:

- Usar las visualizaciones para identificar los puntos débiles de AlexNet y obtener una mejor arquitectura.

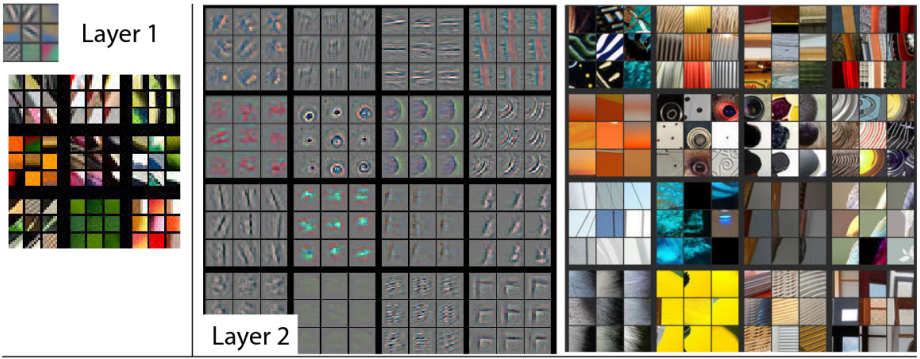


Features de la segunda capa de AlexNet y ZFNet.

Reduciendo el stride de la primera capa de 4 a 2, se reduce el submuestreo que causa *aliasing* en los features de la segunda capa de AlexNet.

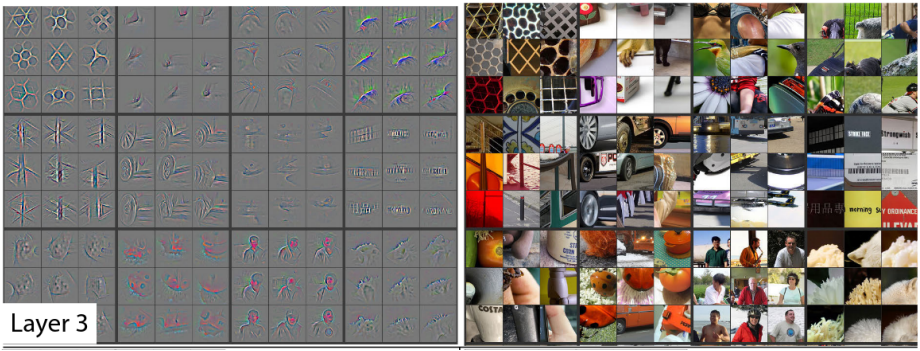
ZFNet: Interpretación de capas intermedias

- ¿Cuáles fueron los 9 *patches* que generaron la mayor activación de un determinado filtro?



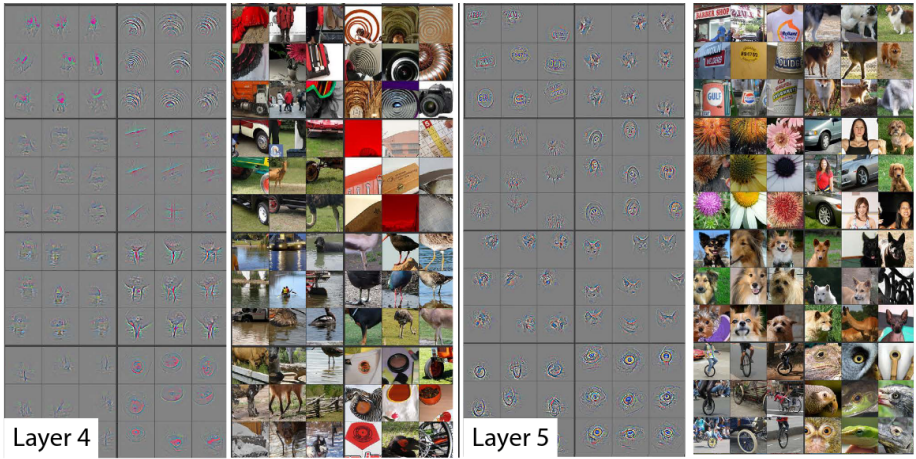
ZFNet: Interpretación de capas intermedias

- ¿Cuáles fueron los 9 *patches* que generaron la mayor activación de un determinado filtro?

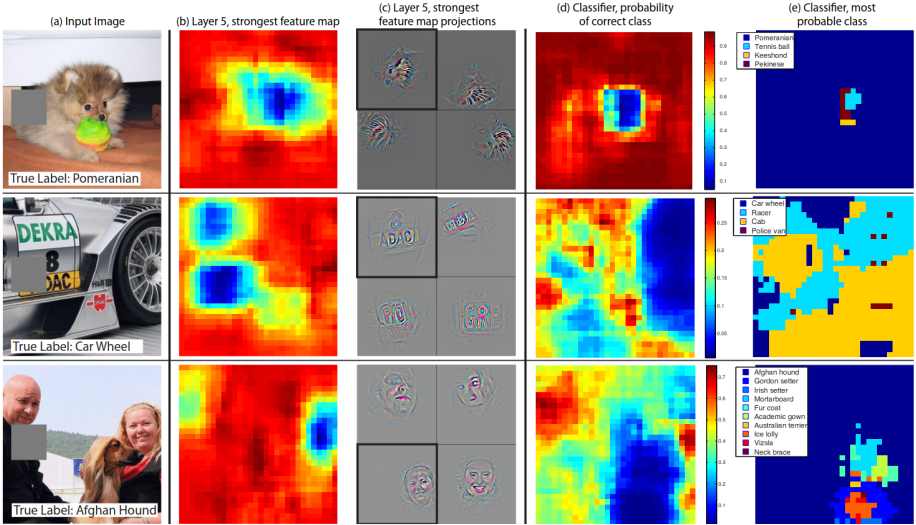


ZFNet: Interpretación de capas intermedias

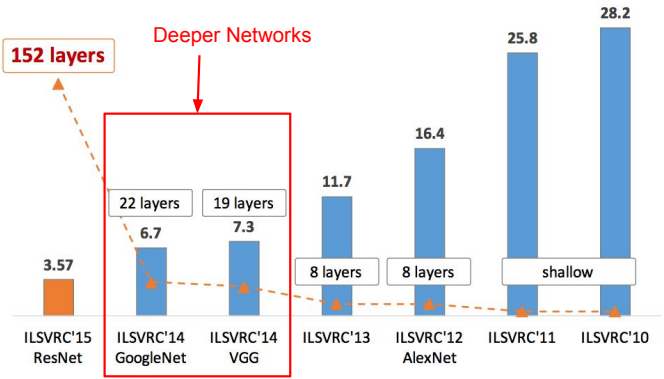
- ¿Cuáles fueron los 9 *patches* que generaron la mayor activación de un determinado filtro?



Sensibilidad a oclusiones



ImageNet Large Scale Visual Recognition Challenge (ILSVRC)



Slides tomadas de cs231n (Stanford) - Fei-Fei Li & Justin Johnson & Serena Yeung

VGG*†

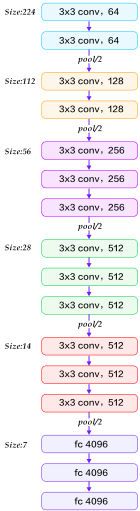
Se estudia la importancia de la profundidad de la red.

Red sensiblemente más profunda (16 o 19 capas de parámetros):

- Convoluciones: filtros 3×3, stride 1
- Maxpooling 2×2, stride 2

Filtros de convolución pequeños:

- Con tres convoluciones 3×3 se obtiene igual campo receptivo que con una convolución 7×7
- Requiere menos parámetros: $3 \times (3^2 \times C_{in} \times C_{out}) < 7^2 \times C_{in} \times C_{out}$
- Se introducen no linealidades intermedias (ReLU): mayor capacidad del modelo



*Visual Geometry Group, Oxford University

†K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015

VGG

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					

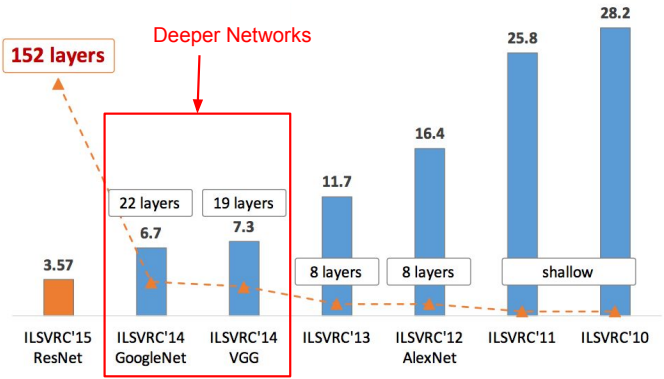
VGG-16

Capa	Tamaño	Memoria	Parámetros
INPUT	224x224x3	224*224*3=150K	0
CONV3-64	224x224x64	224*224*64 = 3,2M	(3*3*3)*64 = 1.728
CONV3-64	224x224x64	224*224*64 = 3,2M	(3*3*64)*64 = 36.864
POOL2	112x112x64	112*112*64 = 800K	0
CONV3-128	112x112x128	112*112*128 = 1,6M	(3*3*64)*128 = 73.728
CONV3-128	112x112x128	112*112*128 = 1,6M	(3*3*128)*128 = 147.456
POOL2	56x56x128	56*56*128 = 400K	0
CONV3-256	56x56x256	56*56*256 = 800K	(3*3*128)*256 = 294.912
CONV3-256	56x56x256	56*56*256 = 800K	(3*3*256)*256 = 589.824
CONV3-256	56x56x256	56*56*256 = 800K	(3*3*256)*256 = 589.824
POOL2	28x28x256	28*28*256 = 200K	0
CONV3-512	28x28x512	28*28*512 = 400K	(3*3*256)*512 = 1.179.648
CONV3-512	28x28x512	28*28*512 = 400K	(3*3*512)*512 = 2.359.296
CONV3-512	28x28x512	28*28*512 = 400K	(3*3*512)*512 = 2.359.296
POOL2	14x14x512	14*14*512 = 100K	0
CONV3-512	14x14x512	14*14*512 = 100K	(3*3*512)*512 = 2.359.296
CONV3-512	14x14x512	14*14*512 = 100K	(3*3*512)*512 = 2.359.296
CONV3-512	14x14x512	14*14*512 = 100K	(3*3*512)*512 = 2.359.296
POOL2	7x7x512	7*7*512 = 25K	0
FC	1x1x4096	4096	7*7*512*4096 = 102.760.448
FC	1x1x4096	4096	4096*4096 = 16.777.216
FC	1x1x1000	1000	4096*1000 = 4.096.000
		15,2M $\times$ 4 bytes $\simeq$ 60,8MB	138M $\times$ 4 bytes = 552MB

VGG: entrenamiento

- Aumentado de datos
- Capa de salida Softmax
- SGD Momentum 0.9
- Batch size 128
- Dropout 0.5
- L2 weight decay 5e-4
- Learning rate 10e-2, reducido 3 veces por un factor de 10
- 74 *epochs* (370000 iteraciones)
- Inicialización aleatoria con distribución normal de media cero y varianza 0.01
- Redes profundas difíciles de entrenar: las primeras cuatro capas CONV y las capas FC se inicializan con los pesos de la red de 11 capas (el resto, inicialización aleatoria)

ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

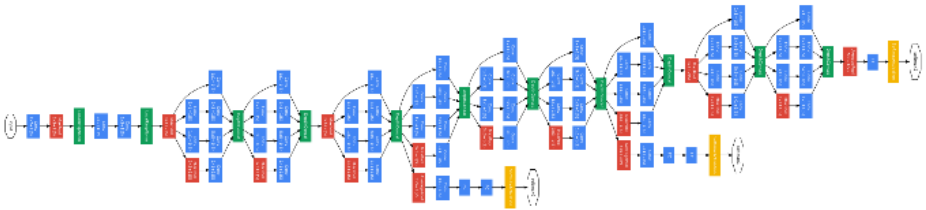


Slides tomadas de cs231n (Stanford) - Fei-Fei Li & Justin Johnson & Serena Yeung

GoogLeNet*

Red diseñada para ser eficiente en términos de memoria y cómputo

- 1500 millones de sumas y multiplicaciones
- 12 veces menos parámetros que AlexNet

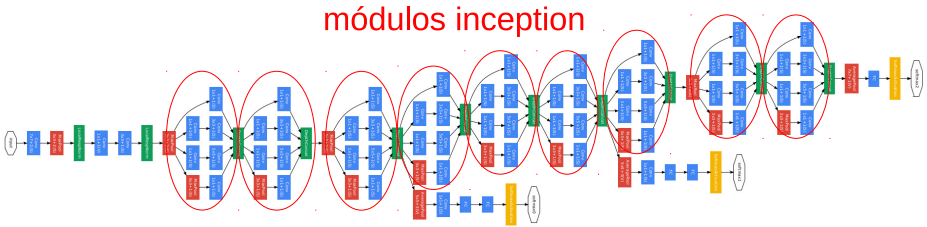


*C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, June 2015

GoogLeNet*

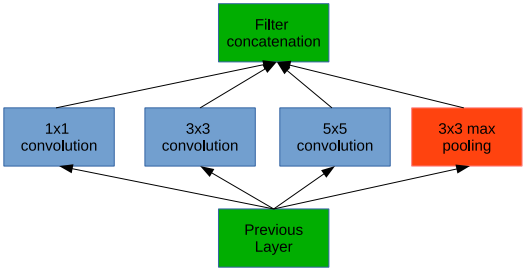
Red diseñada para ser eficiente en términos de memoria y cómputo

- 1500 millones de sumas y multiplicaciones
- 12 veces menos parámetros que AlexNet



*C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, June 2015

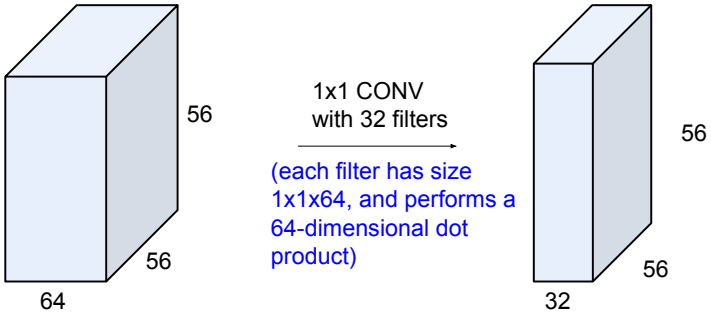
GoogLeNet



Módulo Inception: versión naive

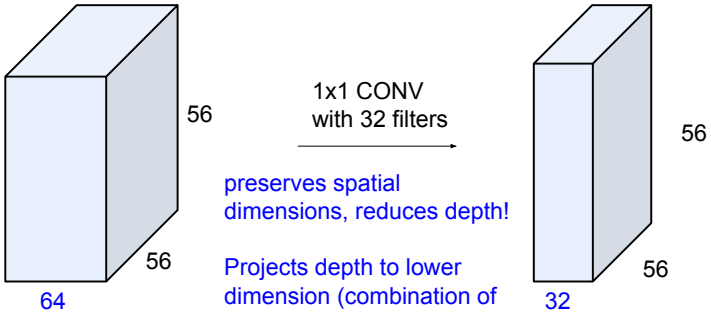
- Filtrado multi-escala
  - Convoluciones 1×1, 3×3 y 5×5
  - Max-pooling de 3×3
- Las salidas se concatenan en un solo volumen

GoogLeNet



Slides tomadas de cs231n (Stanford) - Fei-Fei Li & Justin Johnson & Serena Yeung

GoogLeNet

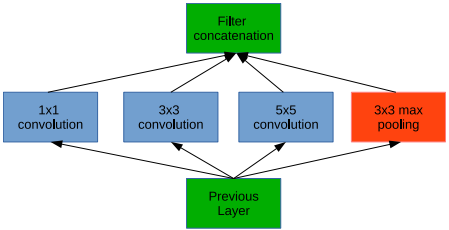


preserves spatial dimensions, reduces depth!

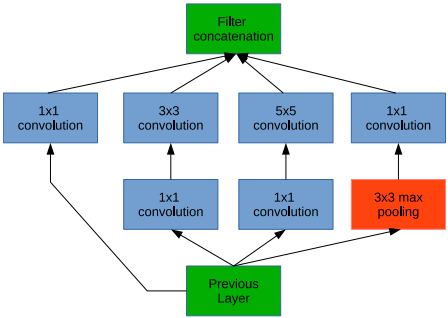
Projects depth to lower dimension (combination of feature maps)

Slides tomadas de cs231n (Stanford) - Fei-Fei Li & Justin Johnson & Serena Yeung

GoogLeNet

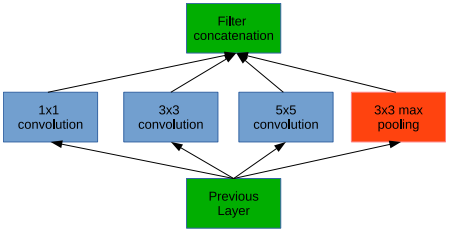


Módulo Inception: versión naive

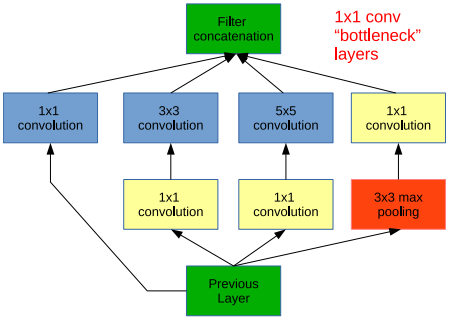


Módulo Inception con reducción de dimensionalidad

GoogLeNet

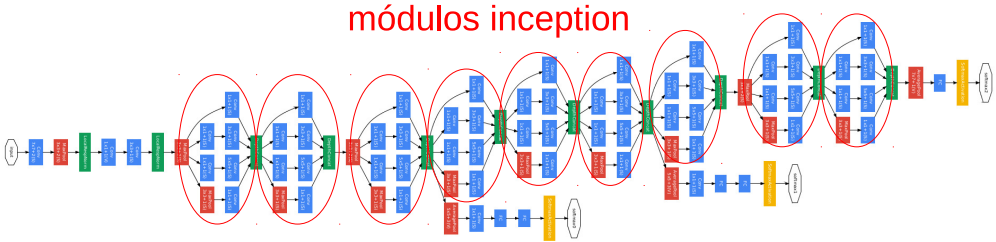


Módulo Inception: versión naive



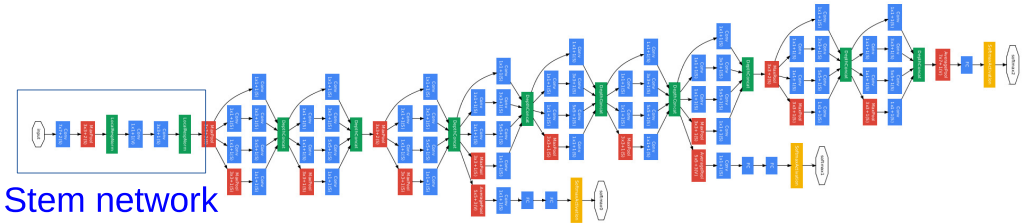
Módulo Inception con reducción de dimensionalidad

GoogLeNet



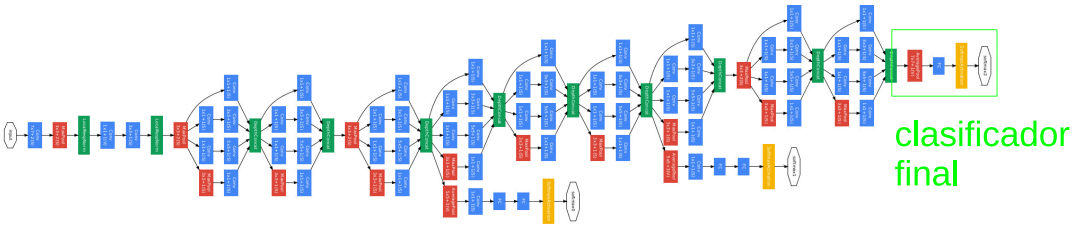
- 9 módulos inception en cascada

GoogLeNet



- La primera etapa es una arquitectura clásica
  - CONV - POOL - LRN - CONV - CONV - LRN - POOL
- Fuerte reducción de resolución espacial por razones de eficiencia computacional

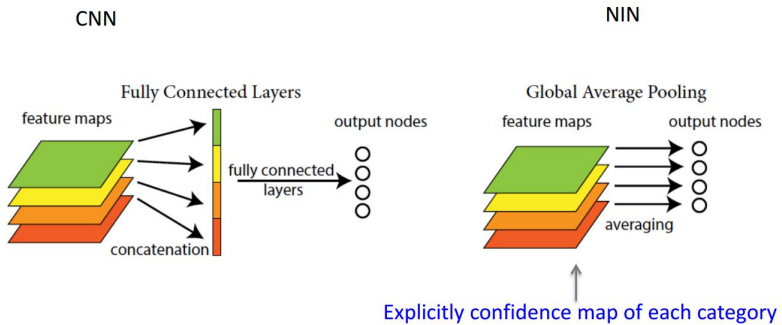
GoogLeNet



- La etapa de clasificación utiliza *Global Average Pooling*

GoogLeNet

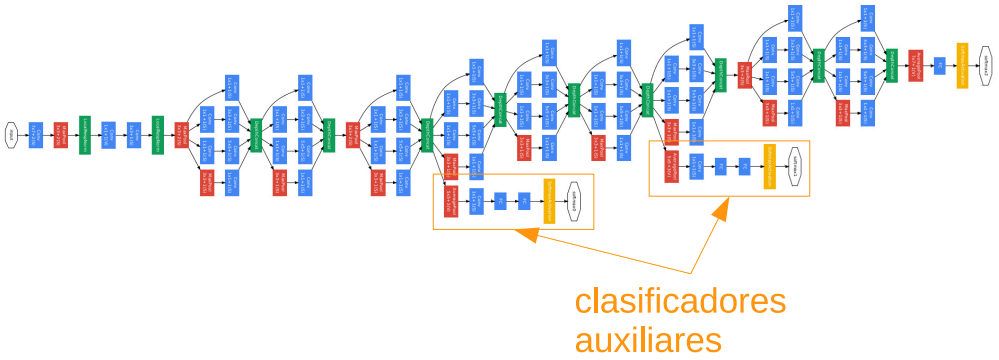
- Global Average Pooling*



- Características:
  - Menos parámetros. No requiere de varias capas FC a la salida.
  - Permite variar el tamaño de la imagen de entrada
  - Descarta información espacial (sin importancia aquí, porque los mapas de activación al final son apenas 7×7, y la tarea es clasificación y no detección).

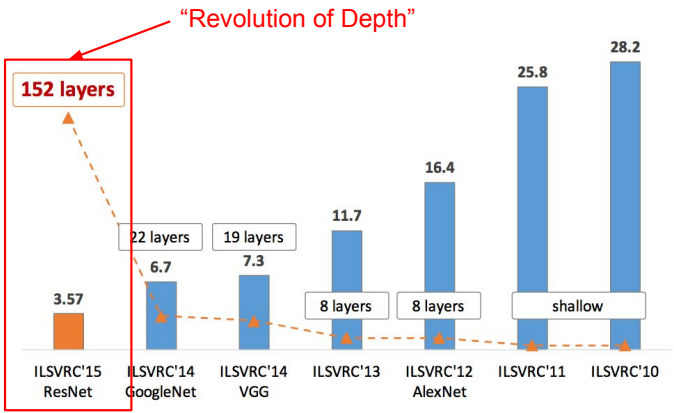
*M. Lin, Q. Chen, and S. Yan, "Network in network," in *2nd International Conference on Learning Representations, ICLR*, 2014

GoogLeNet



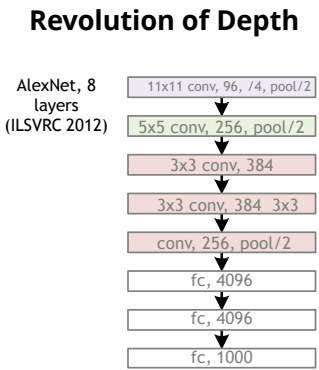
- Clasificadores auxiliares diseñados para combatir el *vanishing gradient* durante el entrenamiento. Son descartados luego del entrenamiento.
- Este truco dejó de ser necesario con el surgimiento de Batch Normalization, que soluciona problemas de entrenamiento de redes con esta profundidad.

ImageNet Large Scale Visual Recognition Challenge (ILSVRC)



Slides tomadas de cs231n (Stanford) - Fei-Fei Li & Justin Johnson & Serena Yeung

Residual Networks (ResNet) *



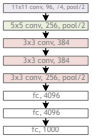
Slide Credit: He et al. (MSRA)

*K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015

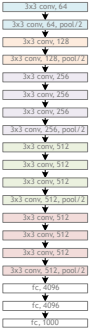
Residual Networks (ResNet) *

Revolution of Depth

AlexNet, 8 layers (ILSVRC 2012)



VGG, 19 layers (ILSVRC 2014)

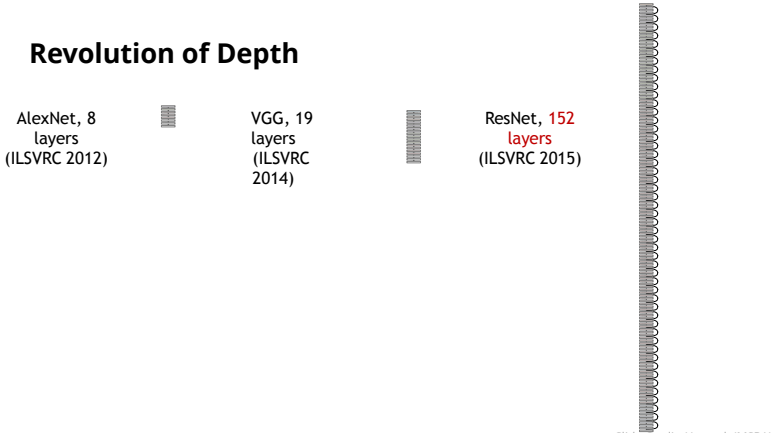


GoogleNet, 22 layers (ILSVRC 2014)

Slide Credit: He et al. (MSRA)

*K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015

Residual Networks (ResNet) *



*K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015

ResNet

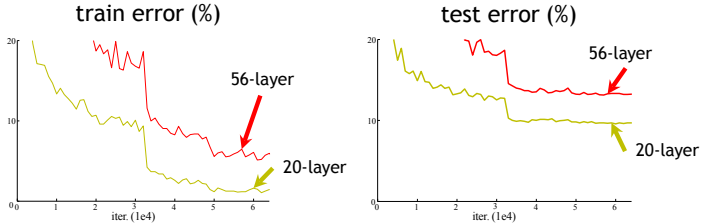
Is learning better networks

as simple as stacking more layers?

Slide Credit: He et al. (MSRA)

Simply stacking layers?

CIFAR-10

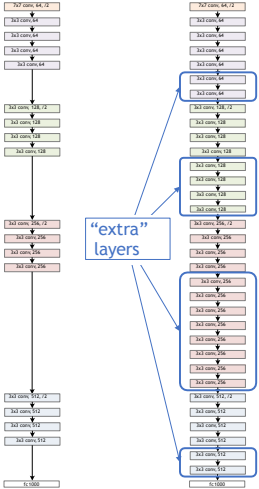


- *Plain* nets: stacking 3x3 conv layers...
- 56-layer net has **higher training error** and test error than 20-layer net

Slide Credit: He et al. (MSRA)

ResNet

a shallower model (18 layers)



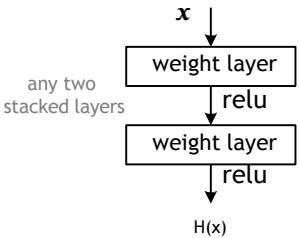
a deeper counterpart (34 layers)

- A deeper model should not have **higher training error**
- A solution *by construction*:
  - original layers: copied from a learned shallower model
  - extra layers: set as identity
  - at least the same training error
- Optimization difficulties: solvers cannot find the solution when going deeper…

Slide Credit: He et al. (MSRA)

Deep Residual Learning

- Plain net

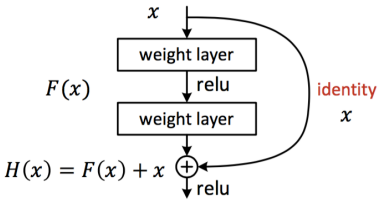


H(x) is any desired mapping,
hope the 2 weight layers fit F(x)

Slide Credit: He et al. (MSRA)

Deep Residual Learning

- Residual net



$H(x)$ is any desired mapping,

~~hope the 2 weight layers fit $H(x)$~~

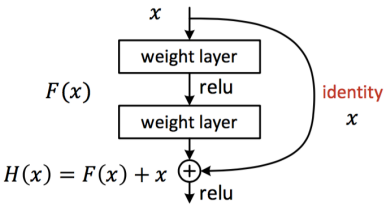
hope the 2 weight layers fit $F(x)$

let $H(x) = F(x) + x$

Slide Credit: He et al. (MSRA)

Deep Residual Learning

- $F(x)$ is a residual mapping w.r.t. identity



- If identity were optimal, easy to set weights as 0
- If optimal mapping is closer to identity, easier to find small fluctuations

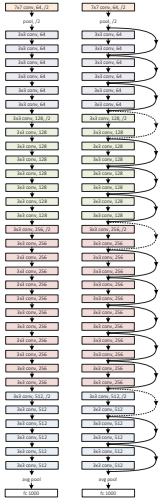
Slide Credit: He et al. (MSRA)

ResNet

Network “Design”

- Keep it simple
- Our basic design (VGG-style)
  - all 3x3 conv (almost)
  - spatial size /2 => # filters x2 (~same complexity per layer)
  - Simple design; just deep!
- Other remarks:
  - no hidden fc
  - no dropout

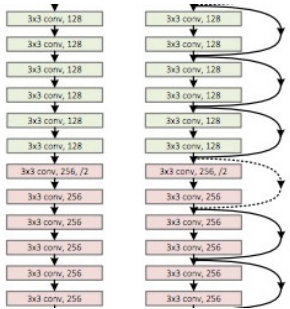
plain net ResNet



Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. CVPR 2016.

Resnet

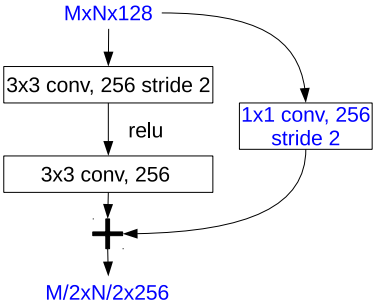
¿Qué hacer cuando las dimensiones de los bloques no coinciden?

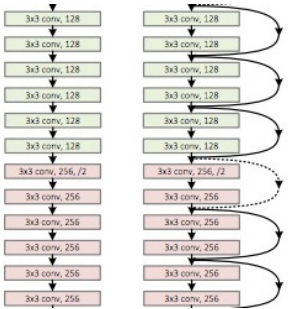


Resnet

¿Qué hacer cuando las dimensiones de los bloques no coinciden?

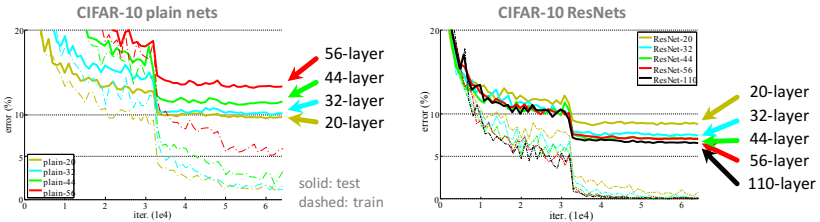
Se sustituye la conexión identidad por un bloque de convolución que ajuste las dimensiones





ResNet

CIFAR-10 experiments

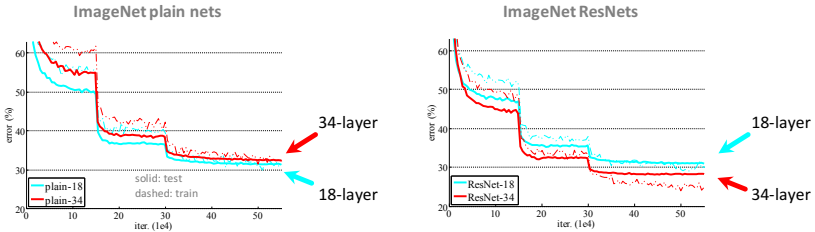


- Deep ResNets can be trained without difficulties
- Deeper ResNets have **lower training error**, and also lower test error

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

ResNet

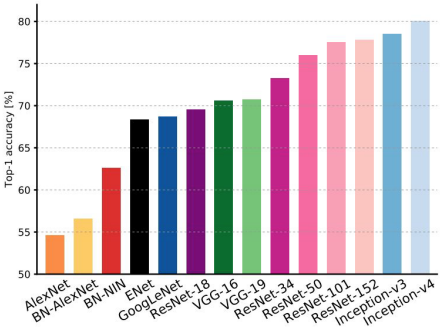
ImageNet experiments

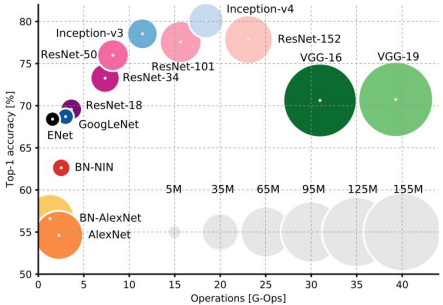


- Deep ResNets can be trained without difficulties
- Deeper ResNets have **lower training error**, and also lower test error

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

Comparación de las arquitecturas*

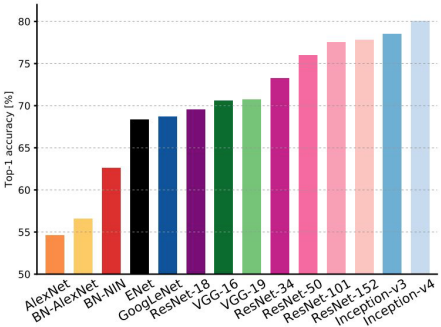


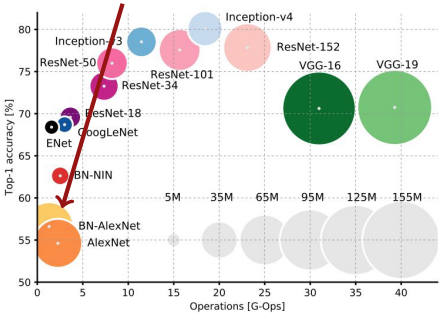


*A. Canziani, A. Paszke, and E. Culurciello, “An analysis of deep neural network models for practical applications,” *CoRR*, vol. abs/1605.07678, 2016

Comparación de las arquitecturas*

AlexNet: "Pequeña" pero mucha memoria, baja precisión

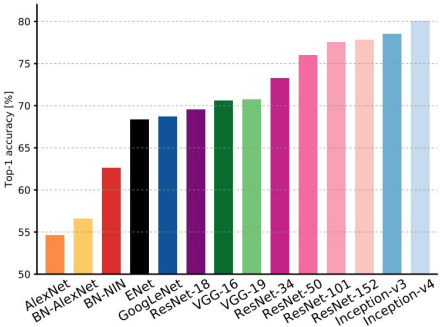


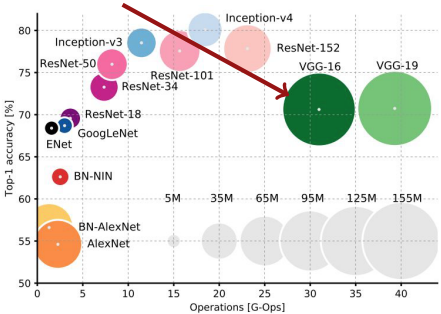


*A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," *CoRR*, vol. abs/1605.07678, 2016

Comparación de las arquitecturas*

VGG: Mucha memoria, baja eficiencia, media precisión

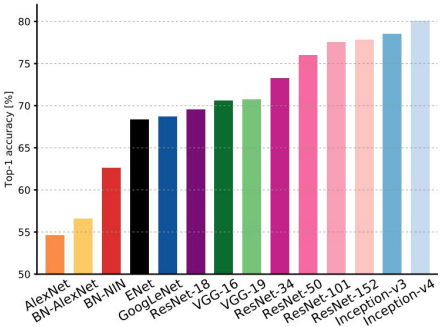


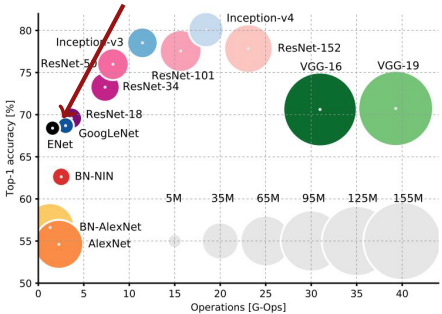


*A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," *CoRR*, vol. abs/1605.07678, 2016

Comparación de las arquitecturas*

GoogLeNet: Media precisión, excelente eficiencia

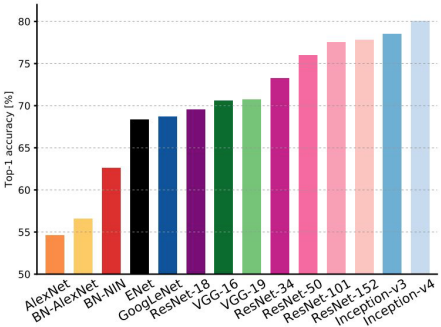


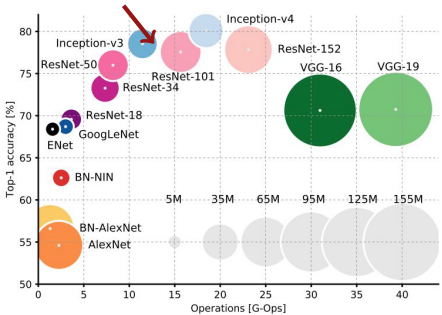


*A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," *CoRR*, vol. abs/1605.07678, 2016

Comparación de las arquitecturas*

ResNet: Alta precisión, eficiencia media/buena

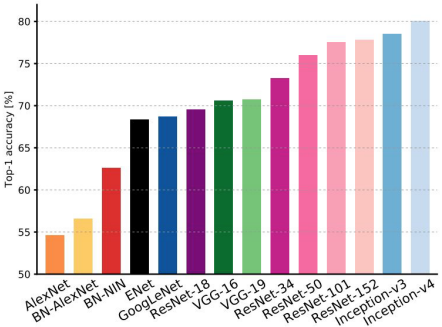


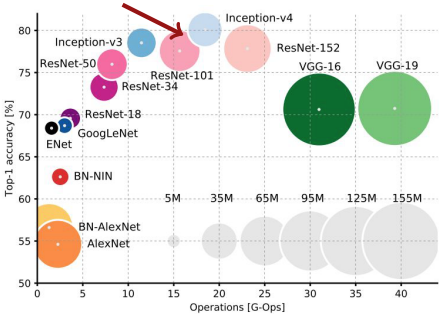


*A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," *CoRR*, vol. abs/1605.07678, 2016

Comparación de las arquitecturas*

Inception-v4:: Excelente precisión, media eficiencia

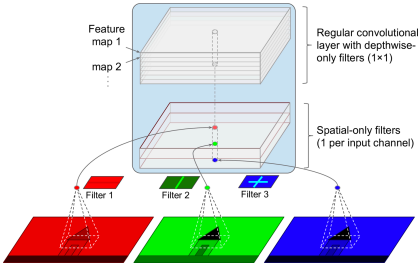




*A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," *CoRR*, vol. abs/1605.07678, 2016

Extreme Inception (Xception)*

- Como Inception-v4, combina ideas de GoogLeNet y ResNet, pero remplaza los módulos inception con una capa llamada *depthwise separable convoluton layer*.
- Se basa en el supuesto que los patrones espaciales e inter-canales se pueden modelar por separado.
- Se compone de dos partes:
  1. Un filtro espacial por cada feature map
  2. A continuación, un filtro inter-canal (capa convolucional con filtros 1×1).
- Ya que se trata de un solo filtro por feature map, no es recomendable para entradas con pocos canales (e.g., no se usa a la entrada).

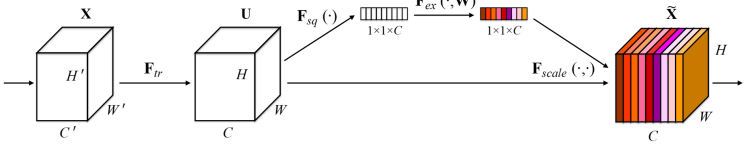


- Xception comienza con dos capas de convolución estándar, y luego usa solamente convoluciones separables (34 capas) con algunos max-pooling, y capas finales usuales (global average pooling y capa de salida fully connected).

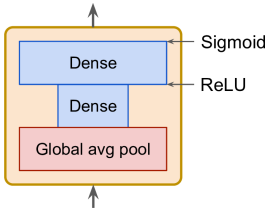
*F. Chollet, "Xception: Deep learning with depthwise separable convolutions," 2017

Squeeze-and-Excitation Networks (SENet)*

- Arquitectura ganadora de ILSCRV 2017 (2.25% Top-5 error)

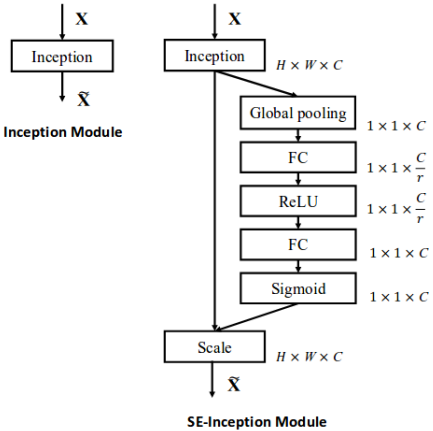


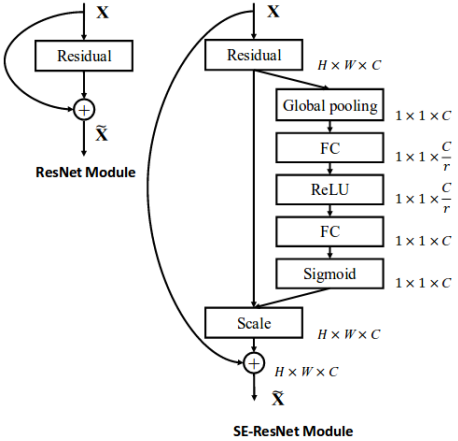
- Basado en observar que la dependencia entre los canales es:
  - Implícita: mezclada con información espacial
  - Local: no se explota la información contextual
- Propone aprender relaciones de inter-dependencia entre los canales
  - Fsq(*) - Global Average Pooling
  - Fex(*) - Busca capturar dependencias no lineales entre los canales
  - Fs(*) - reescalado de los canales dependiente de la entrada



*J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7132–7141, 2018

Squeeze-and-Excitation Networks (SENet)





Uso de modelos pre-entrenados

- Levantar un modelo pre-entrenado

```
model = keras.applications.resnet50.ResNet50(weights="imagenet")
```

- A tener en cuenta:
  - Tamaño de imagen con que fue entrenado
  - Preprocesamiento realizado

```
images_resized = tf.image.resize(images, [224, 224])
inputs = keras.applications.resnet50.preprocess_input(images_resized * 255)
```

- Se hacen las predicciones

```
Y_proba = model.predict(inputs)
```

Modelos pre-entrenados para *transfer learning*

- Queremos construir un **clasificador de imágenes de flores** pero tenemos pocos datos $\Rightarrow$ transfer learning: reutilizamos las capas inferiores de un modelo pre-entrenado para un problema similar.
- Usaremos un modelo Xception pre-entrenado.

```
import tensorflow_datasets as tfds

dataset, info = tfds.load("tf_flowers", as_supervised=True, with_info=True)
dataset_size = info.splits["train"].num_examples # 3670
class_names = info.features["label"].names # ["dandelion", "daisy", ...]
n_classes = info.features["label"].num_classes # 5

test_split, valid_split, train_split = tfds.Split.TRAIN.subsplit([10, 15, 75])

test_set = tfds.load("tf_flowers", split=test_split, as_supervised=True)
valid_set = tfds.load("tf_flowers", split=valid_split, as_supervised=True)
train_set = tfds.load("tf_flowers", split=train_split, as_supervised=True)
```

Importamos la base de flores, y definimos los conjuntos de entrenamiento, validación y test.

Modelos pre-entrenados para *transfer learning*

- Queremos construir un **clasificador de imágenes de flores** pero tenemos pocos datos $\Rightarrow$ transfer learning: reutilizamos las capas inferiores de un modelo pre-entrenado para un problema similar.
- Usaremos un modelo Xception pre-entrenado.

```
def preprocess(image, label):
    resized_image = tf.image.resize(image, [224, 224])
    final_image = keras.applications.xception.preprocess_input(resized_image)
    return final_image, label

batch_size = 32
train_set = train_set.shuffle(1000)
train_set = train_set.map(preprocess).batch(batch_size).prefetch(1)
valid_set = valid_set.map(preprocess).batch(batch_size).prefetch(1)
test_set = test_set.map(preprocess).batch(batch_size).prefetch(1)
```

Adaptamos los datos al modelo Xception, que requiere entradas 224×224

Modelos pre-entrenados para *transfer learning*

- Queremos construir un **clasificador de imágenes de flores** pero tenemos pocos datos $\Rightarrow$ transfer learning: reutilizamos las capas inferiores de un modelo pre-entrenado para un problema similar.
- Usaremos un modelo Xception pre-entrenado.

```
base_model = keras.applications.xception.Xception(weights="imagenet",
                                                  include_top=False)
avg = keras.layers.GlobalAveragePooling2D()(base_model.output)
output = keras.layers.Dense(n_classes, activation="softmax")(avg)
model = keras.Model(inputs=base_model.input, outputs=output)


for layer in base_model.layers:
    layer.trainable = False
```

Finally, we can compile the model and start training:

```
optimizer = keras.optimizers.SGD(lr=0.2, momentum=0.9, decay=0.01)
model.compile(loss="sparse_categorical_crossentropy", optimizer=optimizer,
              metrics=["accuracy"])
history = model.fit(train_set, epochs=5, validation_data=valid_set)
```

- Cargamos el modelo Xception entrenado en ImageNet
- Excluimos las capas superiores de la red (global average pooling y capa densa de salida)
- Creamos nuestras propias capas de global average pooling y de salida
- Congelamos los pesos de las capas pre-entrenadas
- Compilamos y entrenamos

Modelos pre-entrenados para *transfer learning*

- Queremos construir un **clasificador de imágenes de flores** pero tenemos pocos datos $\Rightarrow$ transfer learning: reutilizamos las capas inferiores de un modelo pre-entrenado para un problema similar.
- Usaremos un modelo Xception pre-entrenado.

```
for layer in base_model.layers:
    layer.trainable = True

optimizer = keras.optimizers.SGD(lr=0.01, momentum=0.9, decay=0.001)
model.compile(...)
history = model.fit(...)
```

Refinamos el entrenamiento, descongelando los pesos de las capas pre-entrenadas (con un learning rate menor). Se alcanza un 95% de precisión en test.

Clasificación y localización

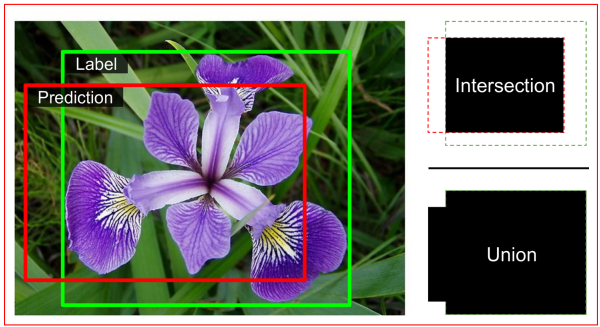
- La localización de un objeto se puede formular como un problema de regresión.
- El objetivo es predecir un *bounding box* que englobe al objeto: usualmente se busca predecir las coordenadas del centro del bounding box, y su alto y ancho.

```
base_model = keras.applications.xception.Xception(weights="imagenet",
                                                  include_top=False)
avg = keras.layers.GlobalAveragePooling2D()(base_model.output)
class_output = keras.layers.Dense(n_classes, activation="softmax")(avg)
loc_output = keras.layers.Dense(4)(avg)
model = keras.Model(inputs=base_model.input,
                    outputs=[class_output, loc_output])
model.compile(loss=["sparse_categorical_crossentropy", "mse"],
              loss_weights=[0.8, 0.2], # depends on what you care most about
              optimizer=optimizer, metrics=["accuracy"])
```

Clasificación y localización

- La localización de un objeto se puede formular como un problema de regresión.
- El objetivo es predecir un *bounding box* que englobe al objeto: usualmente se busca predecir las coordenadas del centro del bounding box, y su alto y ancho.

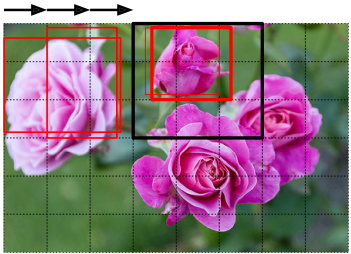
En lugar del MSE, conviene utilizar la medida *Intersection over Union*. En tf.keras, esta métrica está implementada en la clase `tf.keras.metrics.MeanIoU`.



Detección de objetos

Esta tarea consiste en localizar y clasificar múltiples objetos en una imagen. El enfoque utilizado hasta hace poco consistía en:

- Tomar una CNN entrenada para clasificar y localizar un único objeto
- Considerar una grilla de (e.g.) 6×8 de bloques, y deslizar una ventana de (e.g.) 3×3 bloques, que se procesa con la CNN. Repetir considerando ventanas de distinto tamaño.

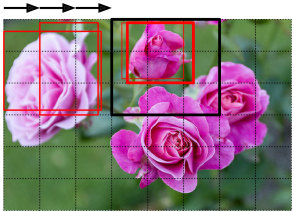


Detección de objetos

Este enfoque funciona bien pero es lento ya que requiere correr la CNN varias veces

Además, como naturalmente el mismo objeto se detectará replicado en varias ventanas solapadas, es necesario realizar un post-procesamiento llamado *non-max supression* para descartar bounding boxes espúreos, que consiste en:

1. Agregar a la CNN una salida extra de medida de *objectness*, que entrenado con un costo de entropía cruzada binario, permita descartar bounding boxes que no contengan objetos con un umbral sobre su *score*.
2. Encontrar el bounding box con el mayor *objectness score*, eliminar los demás bounding boxes detectados con un solapamiento considerable (e.g. 60%).
3. Repetir hasta que no se descarten más bounding boxes.



Veremos cómo detectar objetos con una sola pasada por la red: fully convolutional networks.

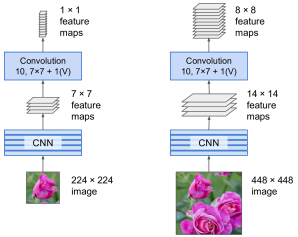
Fully Convolutional Networks (FCN)*

Es posible remplazar las capas densas finales de una CNN por capas convolucionales:

Ejemplo:

- Capa convolucional que devuelve 100 mapas de activación 7×7
- A continuación, una capa densa de 10 neuronas, que computa una suma ponderada de las $100 \times 7 \times 7$ activaciones (más un *bias*)
- ¿Qué sucede si remplazamos la capa densa por una capa convolucional de 10 filtros, cada uno de tamaño 7×7, con *padding* `valid`?

  $\Rightarrow$ La salida de esta capa son 10 mapas de activación, de tamaño 1×1. Esto es, 10 números, al igual que la capa densa.



Cuál es la principal ventaja de esto? Permite generar predicciones para 8x8 regiones de la imagen en una sola pasada

*J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440, 2015

You Only Look Once (YOLO)*

Arquitectura rápida y precisa para detección de objetos, que corre en tiempo real. Construida sobre la base de lo que hemos discutido, con las siguientes diferencias (entre otras):

- Por cada celda de la grilla, devuelve 45 números: 5 bounding boxes con sus 4 coordenadas correspondientes, su objectness score, y sus 20 probabilidades de clase (fue entrenada sobre la base PASCAL VOC que contiene 20 clases).
- Antes de entrenar la red, YOLO encuentra 5 tamaños representativos de bounding box, aplicando k-means a las alturas y anchos de los bounding boxes del conjunto de entrenamiento. Luego la red se entrena para predecir el factor de escala de cada uno de estos 5 bounding boxes representativos.
- Se entrena usando imágenes de distintas escalas, tomando al azar imágenes del batch y re-escalándolas a un nuevo tamaño.

* J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016

Referencias I

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” in *Proc. of the IEEE*, vol. 86(11), pp. 2278–2324, 1998.

A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems*, vol. 25, 2012.

M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *Computer Vision – ECCV 2014*, pp. 818–833, Springer, 2014.

K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *International Conference on Learning Representations*, 2015.

C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, June 2015.

M. Lin, Q. Chen, and S. Yan, “Network in network,” in *2nd International Conference on Learning Representations, ICLR*, 2014.

K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *CoRR*, vol. abs/1512.03385, 2015.

A. Canziani, A. Paszke, and E. Culurciello, “An analysis of deep neural network models for practical applications,” *CoRR*, vol. abs/1605.07678, 2016.

F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” 2017.

J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7132–7141, 2018.

J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440, 2015.

Referencias II

J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition*.
O'Reilly Media, Inc., 2019.