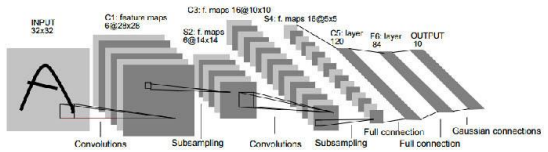
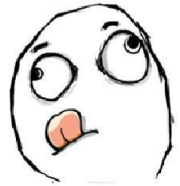
Tema 5.3
Arquitecturas de redes convolucionales actuales

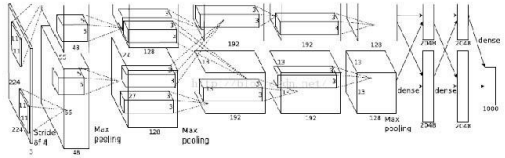
Miguel Ángel Martínez del Amor
Deep Learning
Departamento Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla



LeNet-5

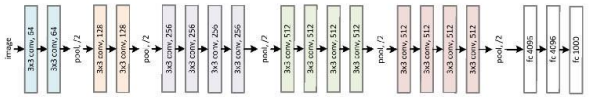


Convolution networks



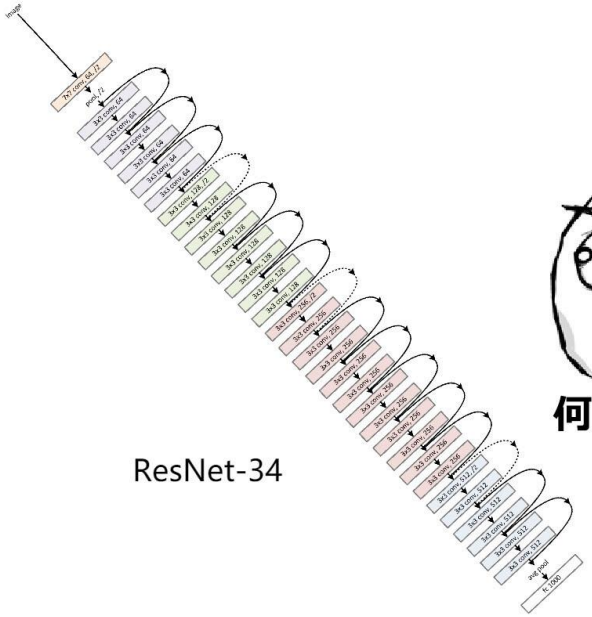
AlexNet

This is getting complicated

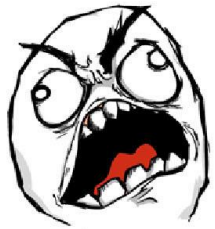


VGG-19

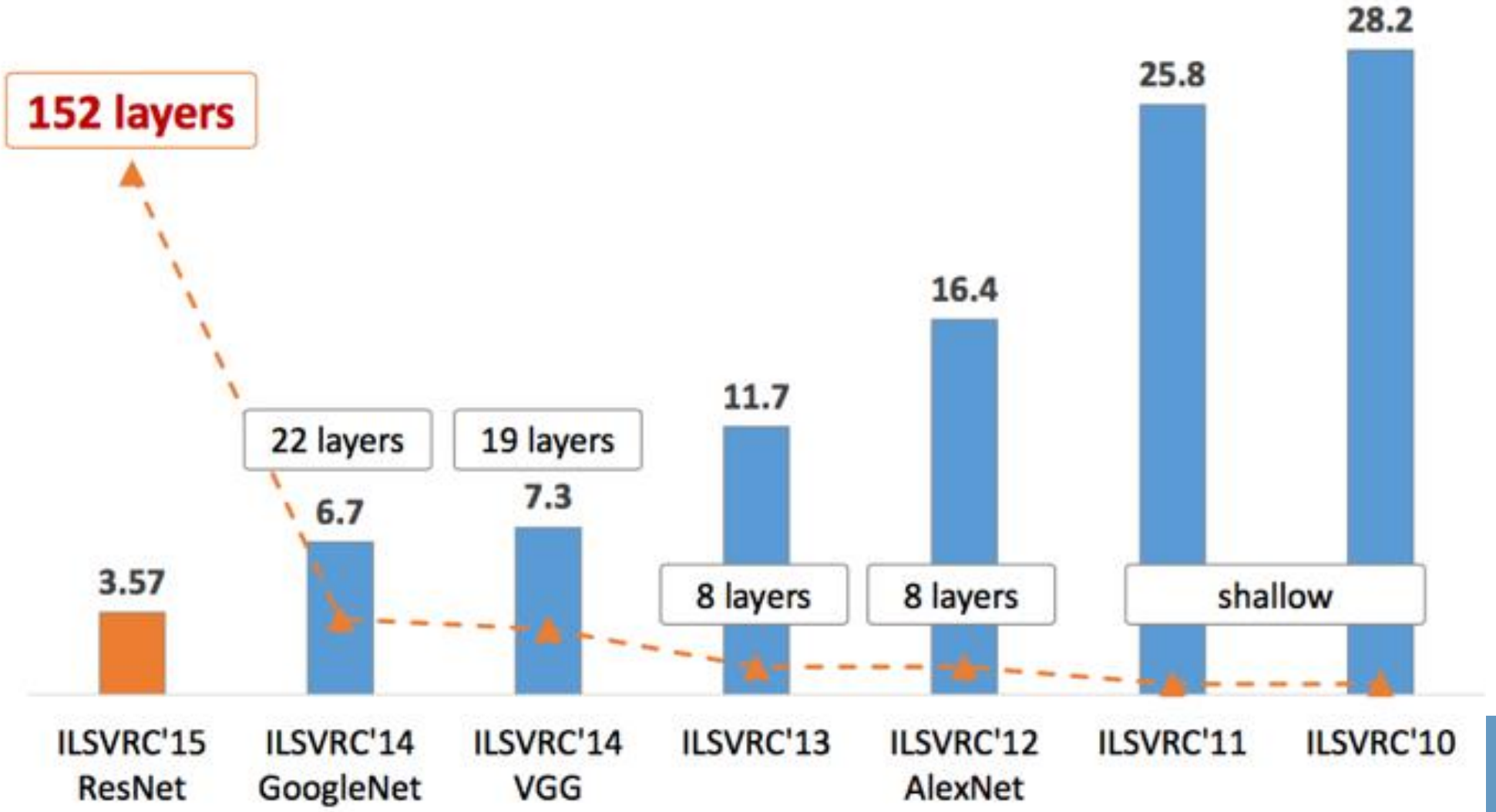
Deep learning



ResNet-34



何だこりゃ！


152 layers
22 layers
19 layers
8 layers
8 layers
shallow
3.57
6.7
7.3
11.7
16.4
25.8
28.2
ILSVRC'15 ResNet
ILSVRC'14 GoogleNet
ILSVRC'14 VGG
ILSVRC'13
ILSVRC'12 AlexNet
ILSVRC'11
ILSVRC'10

Contenidos

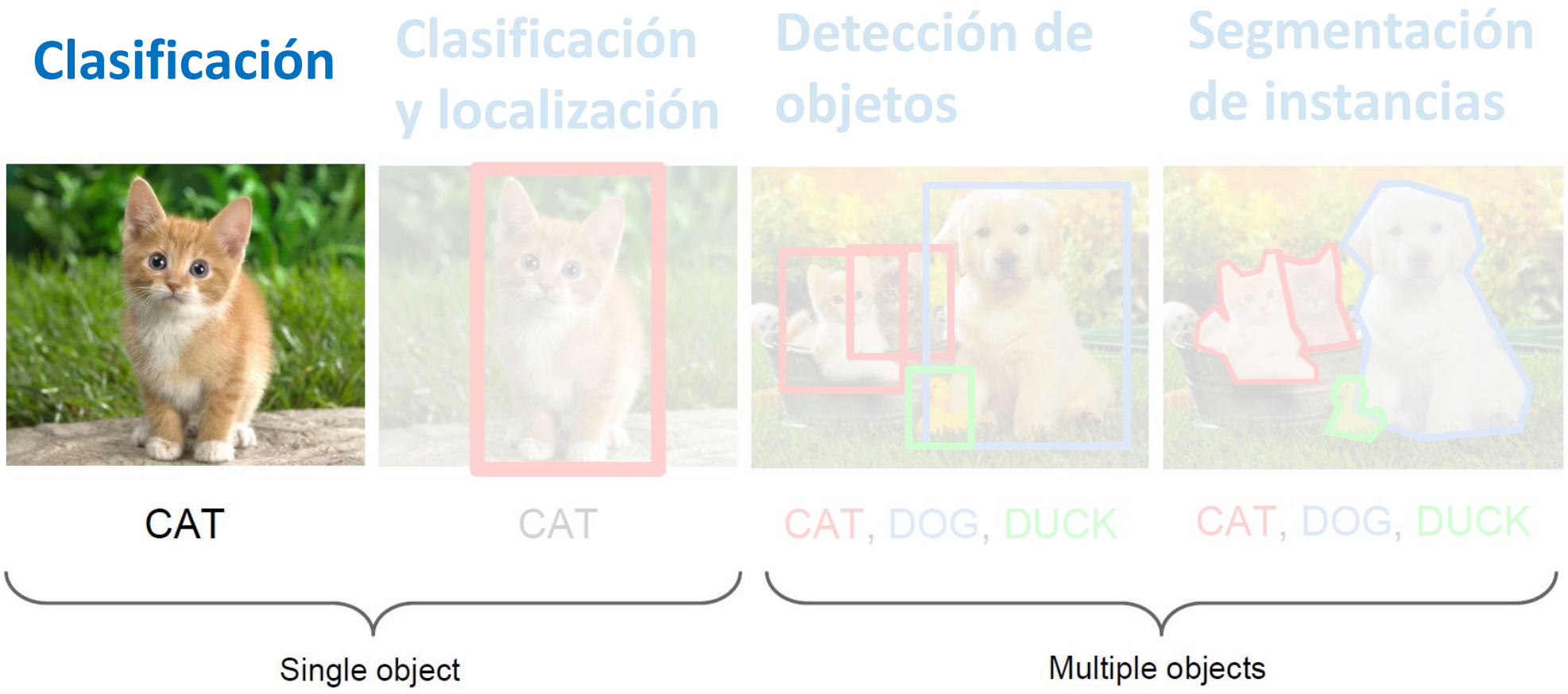
- Clasificación de objetos
  - Redes secuenciales (una sola secuencia de capas)
    - LeNet, AlexNet, VGG
  - Redes no secuenciales (ramas de secuencias de capas)
    - ResNet, GoogLeNet/Inception
  - Comparativa
- Detección de objetos
  - R-CNN
  - YOLO

Contenidos

- **Clasificación de objetos**
  - Redes secuenciales (una sola secuencia de capas)
    - LeNet, AlexNet, VGG
  - Redes no secuenciales (ramas de secuencias de capas)
    - ResNet, GoogLeNet/Inception
  - Comparativa
- Detección de objetos
  - R-CNN
  - YOLO

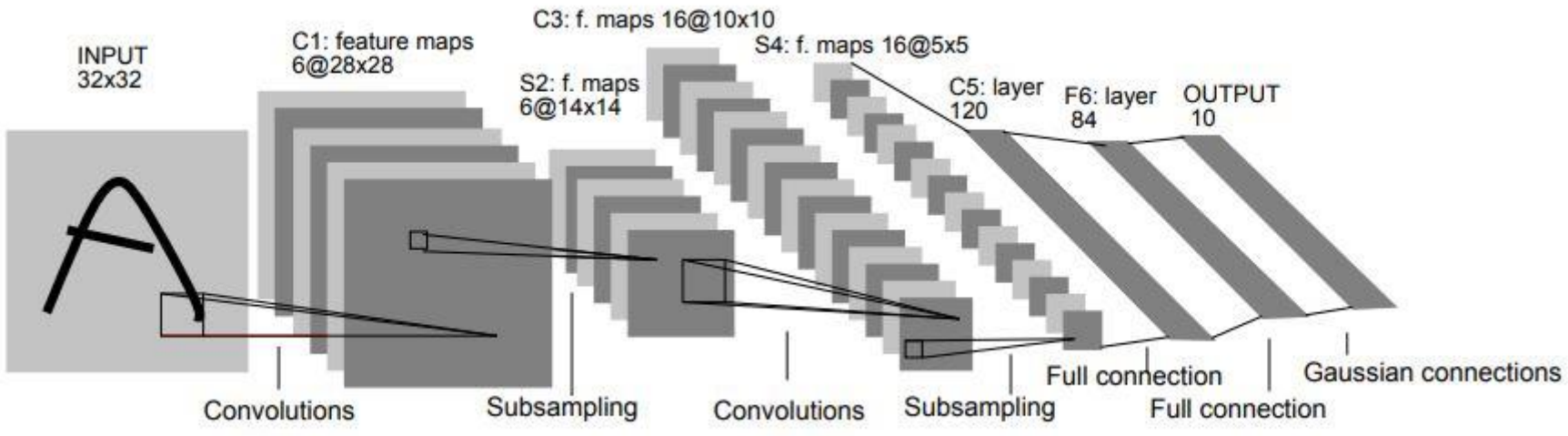
Clasificación de objetos

- Tareas en **visión por computador**



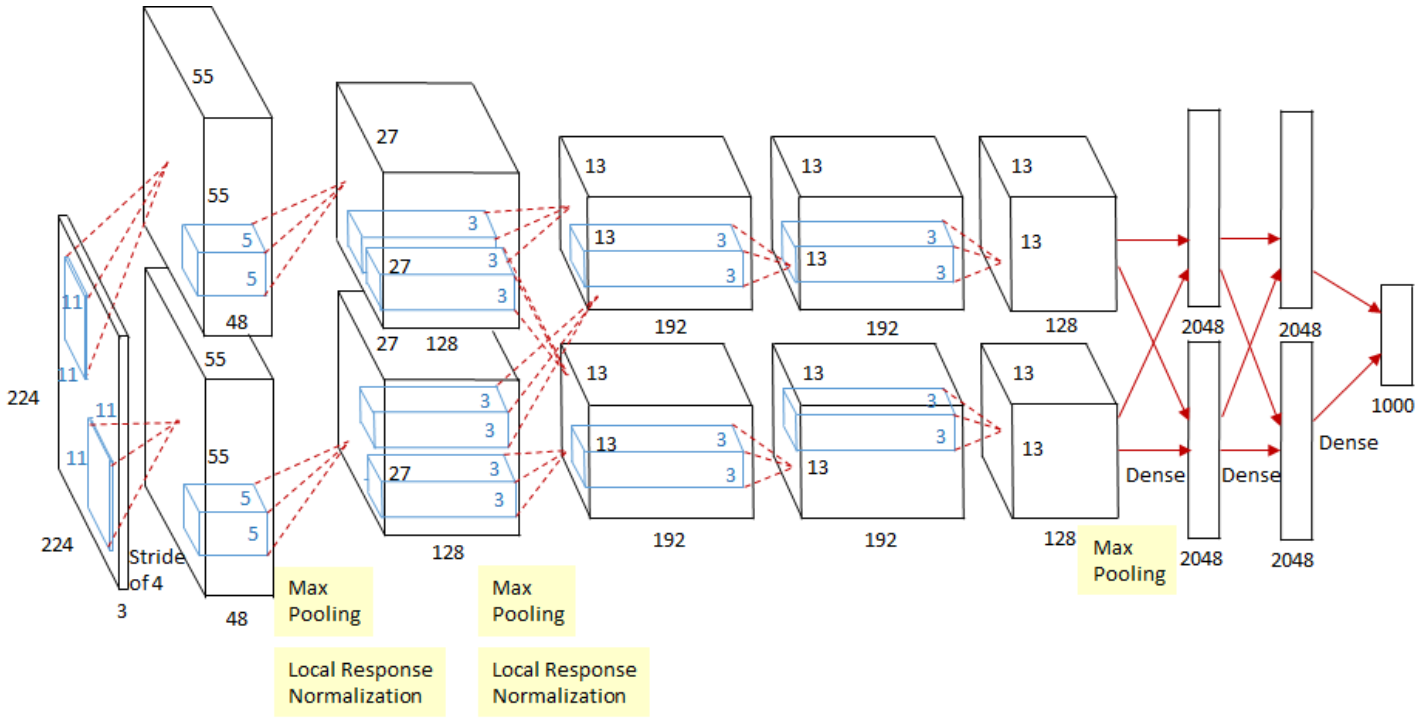
Redes secuenciales

- **LeNet** (1990)
  - Primera red convolucional, para clasificar dígitos (MNIST).
  - Dos bloques de conv y pooling, y 2 FC.



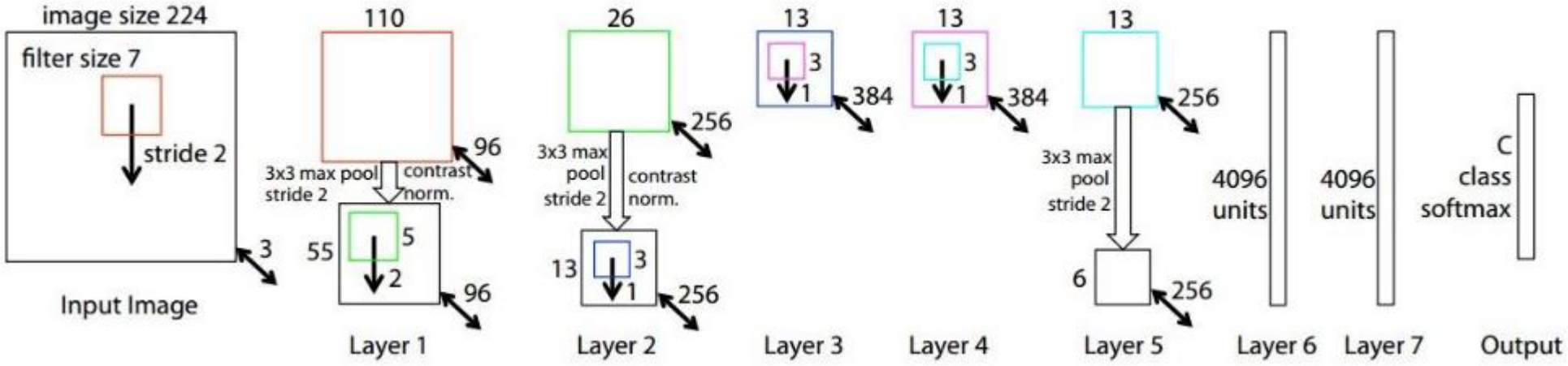
Redes secuenciales

- **AlexNet** (2012)
  - Éxito en ILSRVC con ImageNet.
  - 2 capas conv, pooling
  - 3 capas conv, una pooling
  - Dropout
  - FC con 2 capas



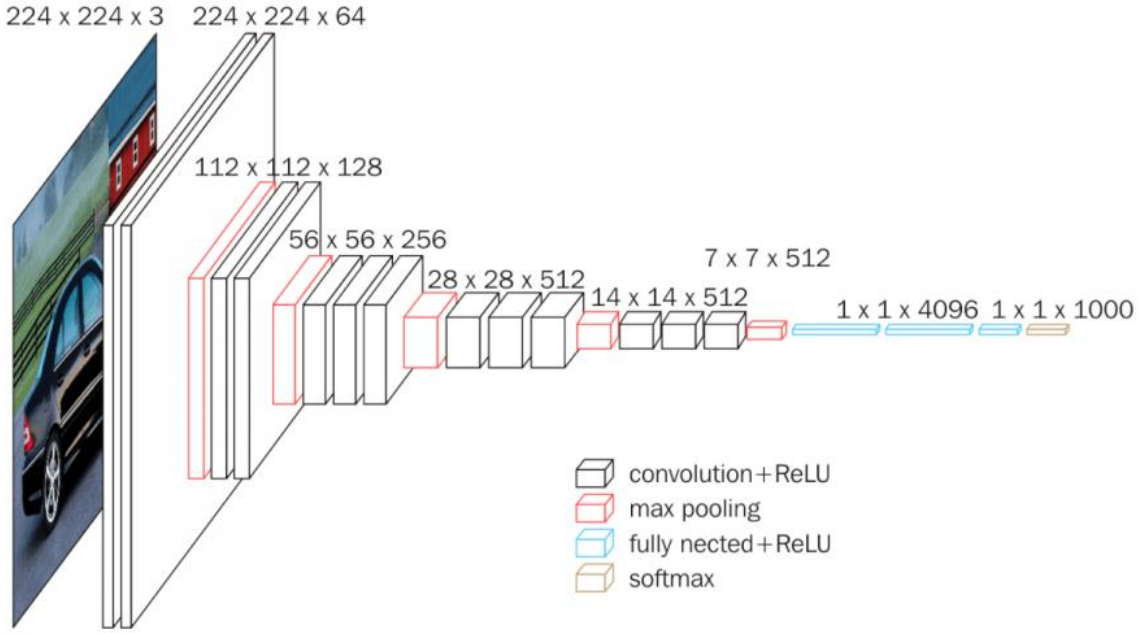
Redes secuenciales

- **ZFNet** (2013):
  - Ganó ILSRVC2013.
  - Estructura similar a AlexNet pero con más elementos.
  - Cambio de hiperparámetros para entrenarlo.



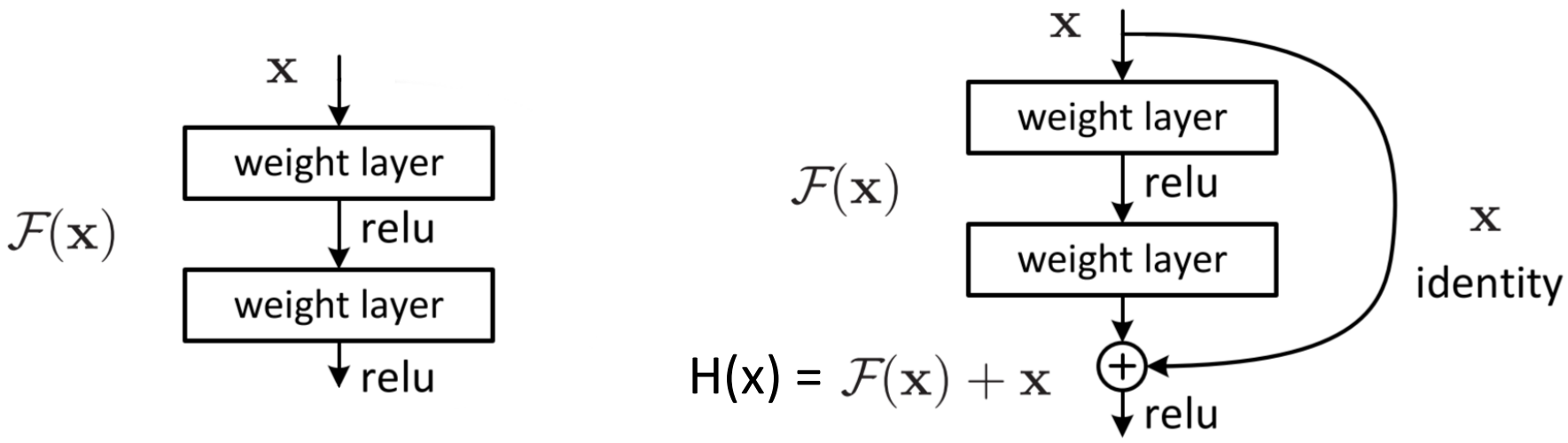
Redes secuenciales

- **VGG** (2014):
  - Versión **VGG16** y **VGG19** (con 16 y 19 capas)
  - Ganó ILSRVC2014 en clasificación y localización de objetos.
  - Convolución 3x3 (con efectividad de 5x5)
  - Pooling 2x2



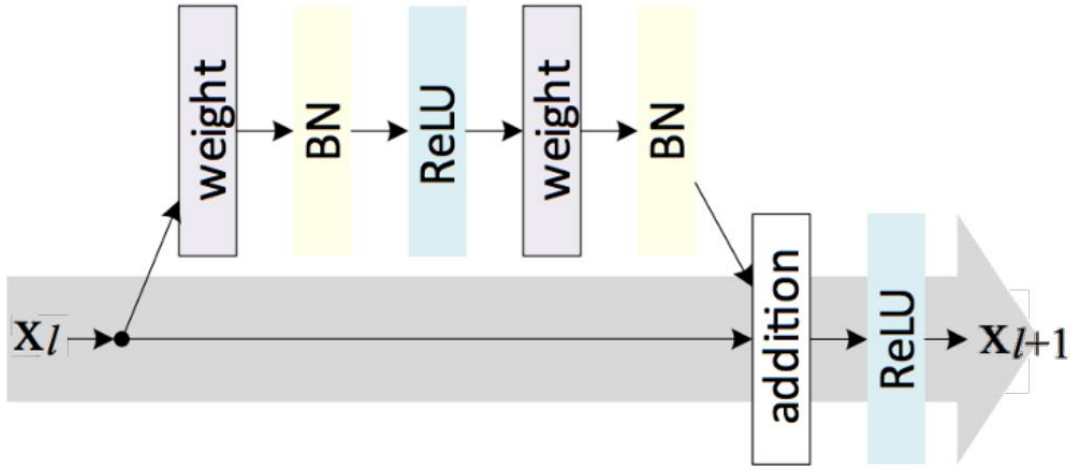
Redes no secuenciales

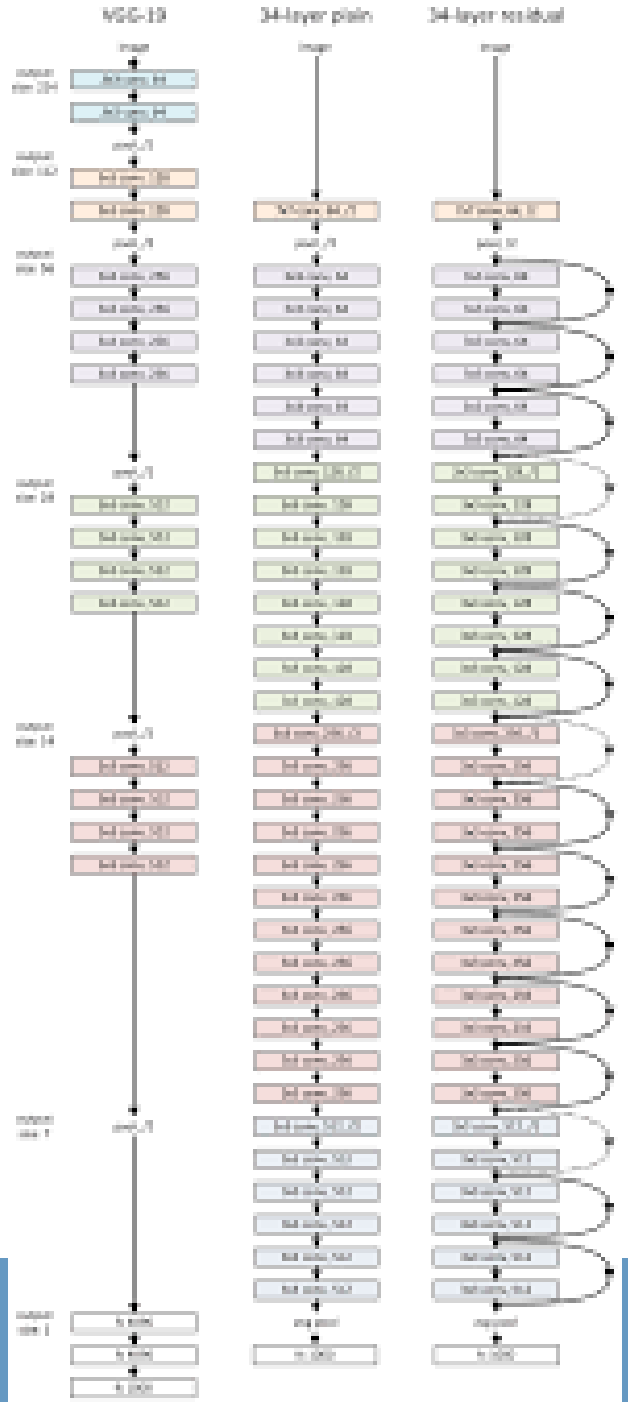
- **ResNet** (2015):
  - Con la profundidad de la red se producen problemas de desvanecimiento o explosión de los gradientes.
  - Idea: **bloques residuales**, propagando la entrada del bloque a su salida



Redes no secuenciales

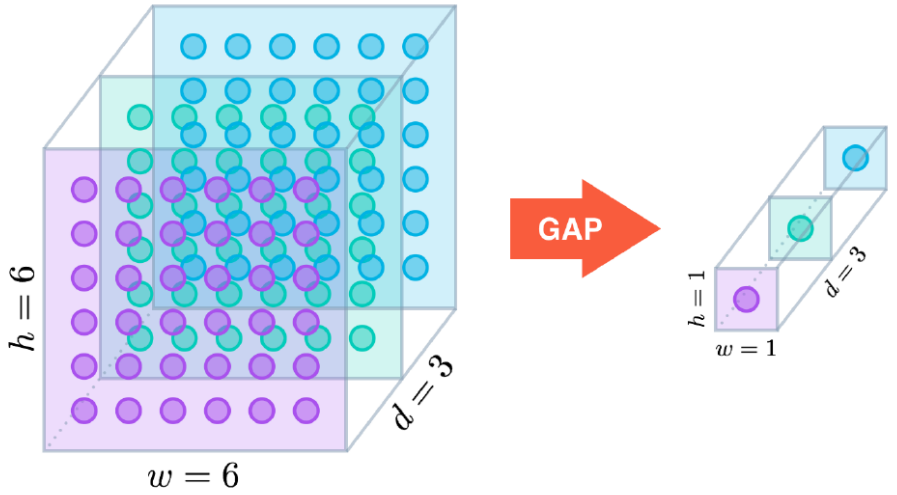
- **ResNet:**
  - Batch Normalization después de cada capa convolucional
  - Inicialización He et al (Xavier/2)
  - SGD + Momentum (0.9), learning rate: 0,1
  - Tamaño Mini-batch 256
  - Regularización L2 10-5
  - No dropout





Redes no secuenciales

- **ResNet**:
  - Uso de **Global Average Pooling (GAP)** en sustitucion de las capas totalmente conectadas.
    - Funcionamiento: toman el **promedio de cada mapa de características**.
    - Con cada promedio se crea un vector que pasa a la capa de clasificación.
  - Al tener una naturaleza más cercana a las estruc
    la red a crear una correspondencia entre su resp
    y la clasificación.
  - Ventajas:
    - Son más robustas a la traslación.
    - No se necesitan parámetros.

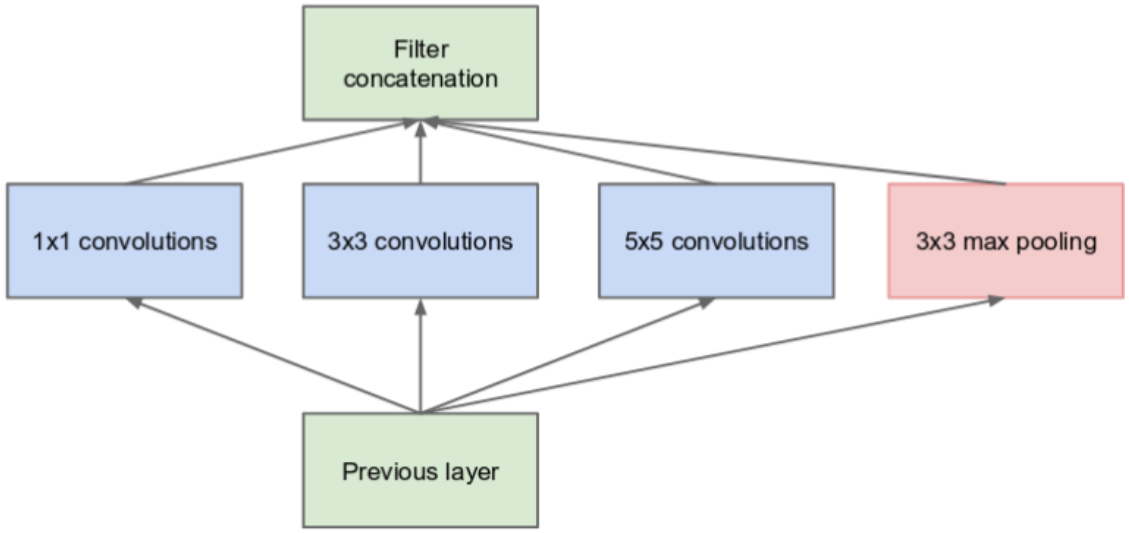


Redes no secuenciales

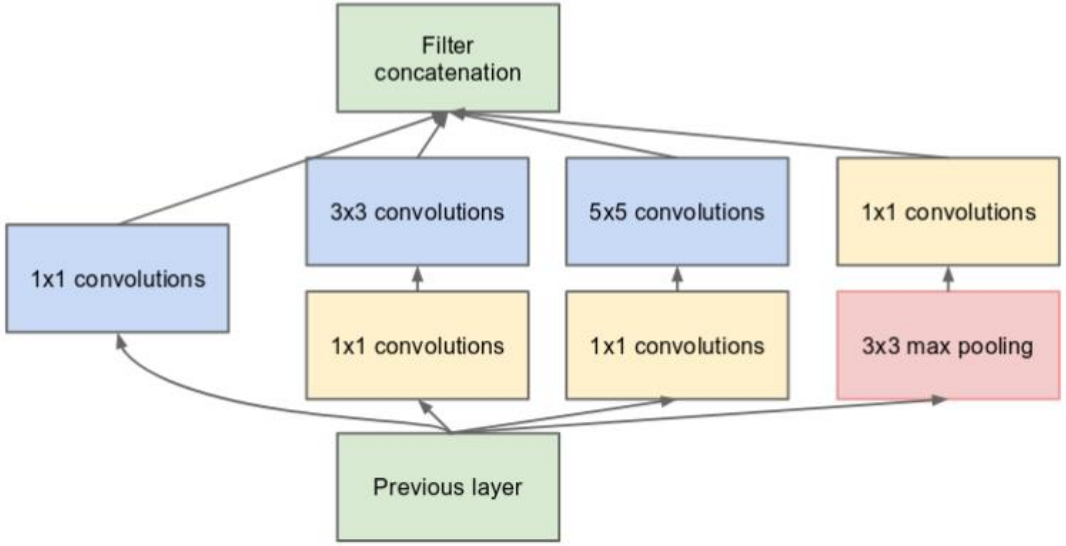
- **Inception** V1/**GoogLeNet** (2014):
  - Problema:
    - Elegir el tamaño de kernel es dificil:
      - tamaño grande (información global),
      - tamaño pequeño (información local).
    - Redes muy profundas son propensas al overfitting y al decaimiento del gradiente.
    - Aplicar capas convolucionales es costoso computacionalmente.
  - Solución: bloques inception:
    - Filtros de distinto tamaño en el mismo nivel
    - Extracción de características multinivel

Redes no secuenciales

- **Inception V1**:



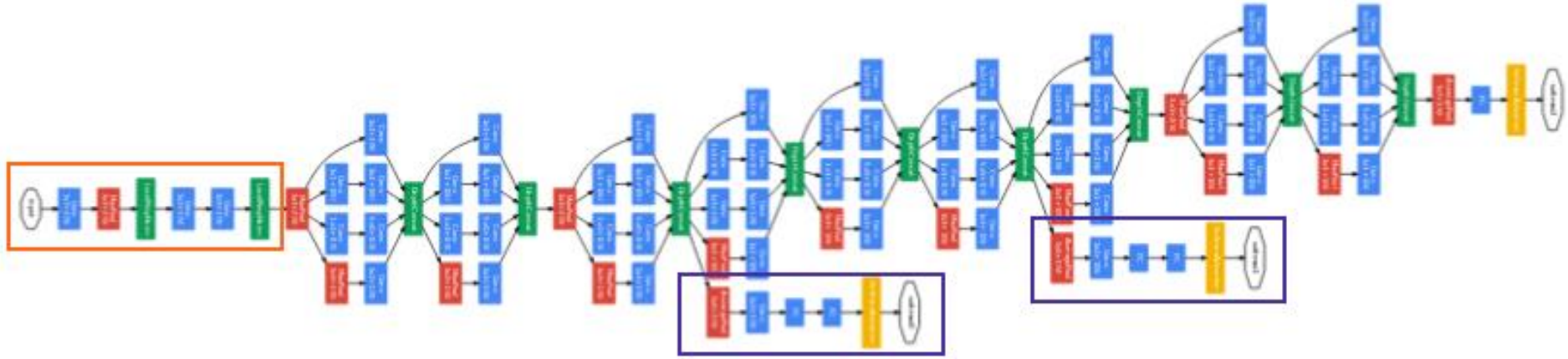
(a) Inception module, naïve version



(b) Inception module with dimension reductions

Redes no secuenciales

- **Inception V1**:
  - 9 módulos inception (total 27 capas), tras una fase stem (inicial)
  - Uso de GAP en vez de FC.
  - Para prevenir el vanishing gradient, incluyen dos clasificadores auxiliares.



Redes no secuenciales

- **Inception V2 (2015)**: aligeramiento de los módulos inception factorizando convoluciones 5x5 a 2 de 3x3.
- **Inception V3 (2015)**: incluye las mejoras del Inception V2 más:
  - Optimizador RMSProp.
  - Factorización de las convoluciones 7x7.
  - BatchNorm en los clasificacores auxiliares.
  - Label smoothing: regularizador añadido a la función de pérdida.
- **Inception V4 (2016)**: modificación de operaciones en stem (inicio antes de módulos inception), y nuevos bloques de reducción.
- **Inception-ResNet (2016)**: módulos inception con residuos, al estilo del bloque residual en ResNet.
- **Xception (2016)**: extensión de la arquitectura inception

Comparativa

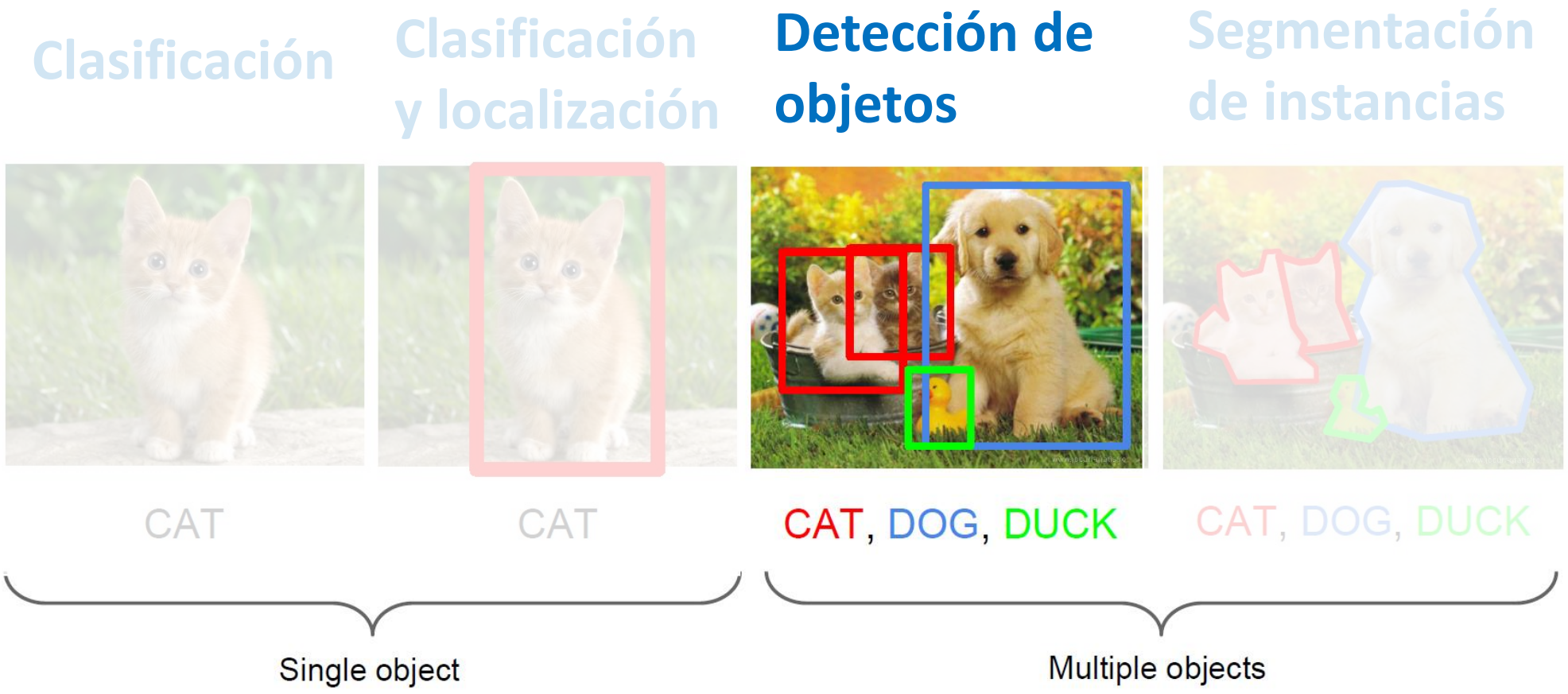
Year	CNN	Developed by	Place	Top-5 error rate	No. of parameters
1998	LeNet(8)	Yann LeCun et al			60 thousand
2012	AlexNet(7)	Alex Krizhevsky, Geoffrey Hinton, Ilya Sutskever	1st	15.3%	60 million
2013	ZFNet()	Matthew Zeiler and Rob Fergus	1st	14.8%	
2014	GoogLeNet(19)	Google	1st	6.67%	4 million
2014	VGG Net(16)	Simonyan, Zisserman	2nd	7.3%	138 million
2015	ResNet(152)	Kaiming He	1st	3.6%	

Contenidos

- Clasificación de objetos
  - Redes secuenciales (una sola secuencia de capas)
    - LeNet, AlexNet, VGG
  - Redes no secuenciales (ramas de secuencias de capas)
    - ResNet, GoogLeNet/Inception
  - Comparativa
- **Detección de objetos**
  - R-CNN
  - YOLO

Detección de objetos

- Tareas en **visión por computador**



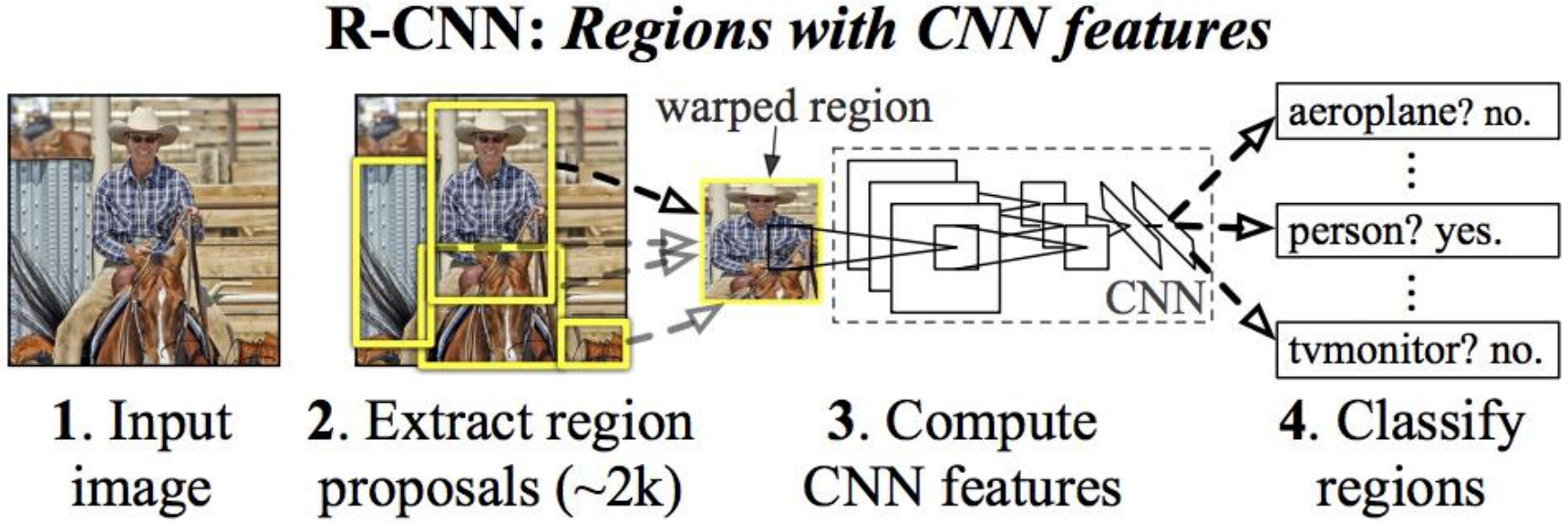
Detección de objetos

- Datasets

	PASCAL VOC (2010)	ImageNet Detection (ILSVRC 2017)	MS-COCO (2019)
Número de clases	20	200	80
Número de imágenes (train + val)	20.000	470.000	330.000
Media objetos por imagen	2,4	1,1	5

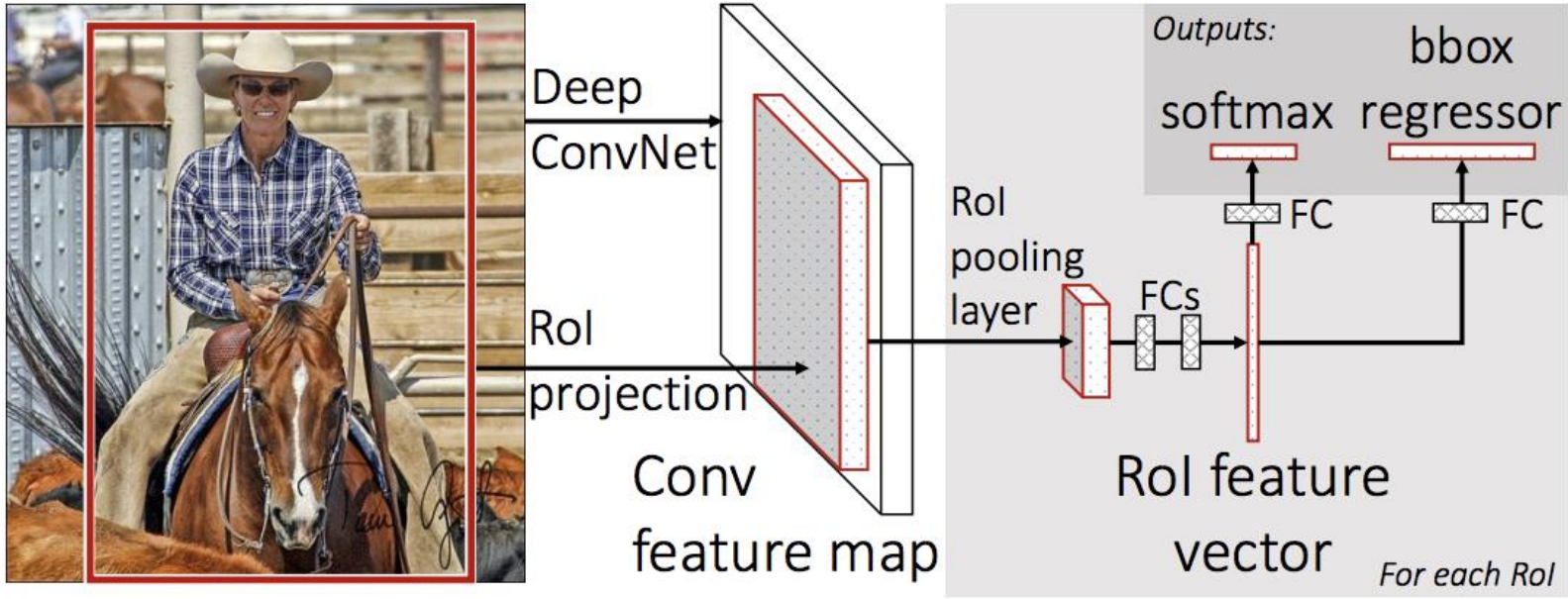
Detección de objetos

- **R-CNN** (2014): familia de modelos creados en Microsoft Research
  - **Region proposal**: selección de bounding boxes candidatos (*selective search*)
  - **Feature extractor**: extrae las características de cada región candidato usando una CNN (AlexNet con capa FC re-entrenada)
  - **Classifier**: clasifica las regiones en una clase



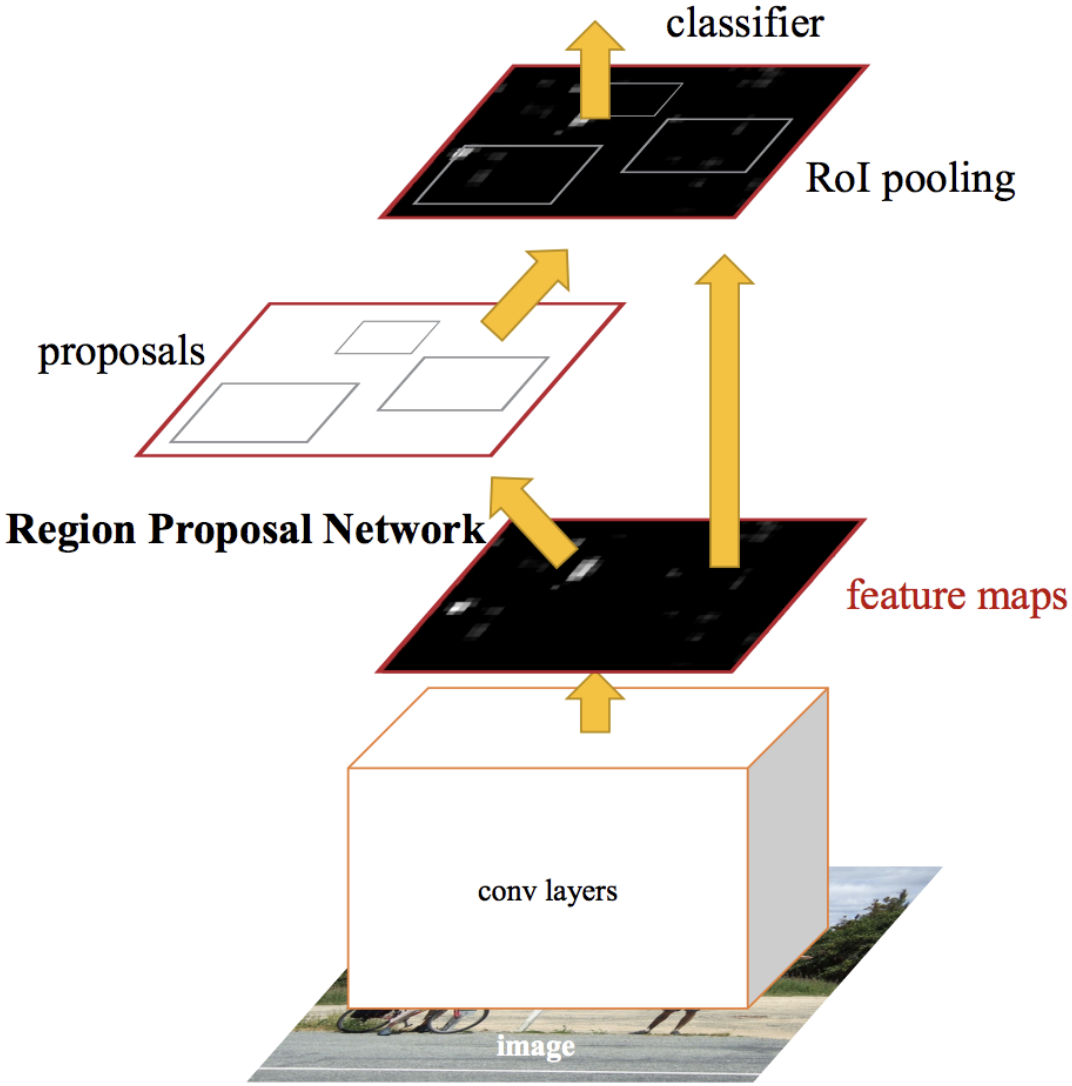
Detección de objetos

- **Fast R-CNN** (2015):
  - Un solo modelo en vez de un pipeline de módulos
  - La imagen va directa a una red convolucional (VGG-16), donde las capas FC se reemplazan por capa RoI Pooling, que extrae las características para una región candidata.



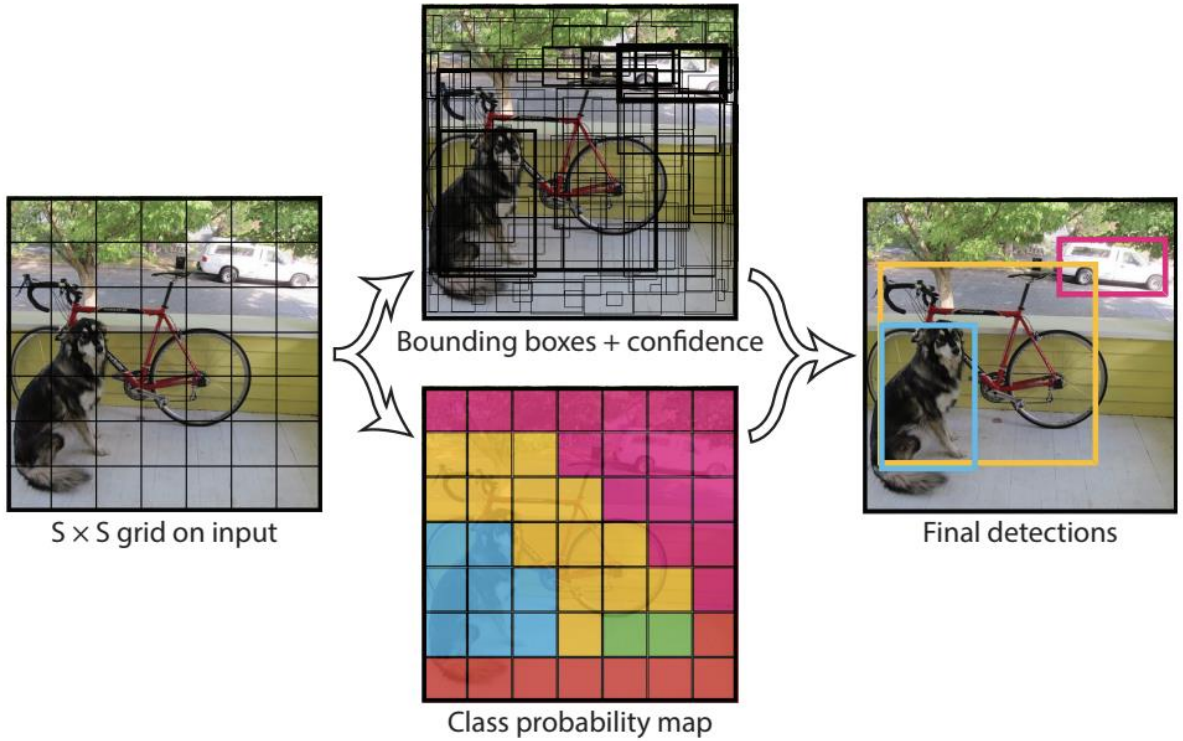
Detección de objetos

- **Faster R-CNN** (2016):
  - Region Proposal Network: CNN para proponer regiones y el tipo de objeto a considerar.
  - Fast R-CNN: extractor de características de las regiones propuestas, devolviendo el bounding box y las clases



Detección de objetos

- YOLO (*You Only Look Once,* 2015*):*
  - Más rápido que los R-CNN, pero menos preciso
  - Divide la imagen en un grid
  - Por cada celda del grid, predice:
    - Boxes B
    - Clases
  - Aplica regresión para completar la imagen



Detección de objetos

- YOLO (*You Only Look Once,* 2015*):*
  - Resultados

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [30]	2007	16.0	100
30Hz DPM [30]	2007	26.1	30
~~Fast YOLO~~	~~2007+2012~~	~~52.7~~	~~**155**~~
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [37]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[27]	2007+2012	73.2	7
Faster R-CNN ZF [27]	2007+2012	62.1	18

Detección de objetos

- YOLOv2 (2016)*:*
  - Entrenado con más objetos (9000).
  - Mejoras en la red (uso de BatchNorm).
  - Pre-calculado de bounding boxes usando un análisis de k-medias
- YOLOv3 (2018):
  - Mejoras menores: red más profunda, etc.

Recapitulación

- CNNs para **clasificación de objetos**
  - **LeNet**, **AlexNet**, **VGG**, **ResNet**, **Inception**, …
  - Las hemos clasificado en redes **secuenciales** y **no secuenciales**, según si en el pipeline de capas hay ramificaciones
- CNNs para **detección de objetos**
  - Familias **R-CNN** vs **YOLO**