

Introducción

Visión artificial o computer vision es la disciplina que estudia cómo procesar, analizar e interpretar imágenes de forma automática. Estas técnicas tienen aplicaciones en muchos ámbitos, como la seguridad, la medicina o la navegación automática.

Las imágenes digitales son matrices de números

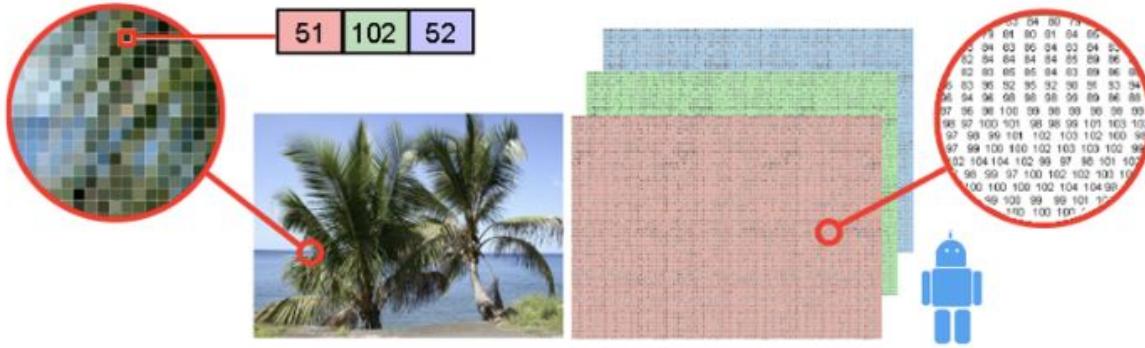
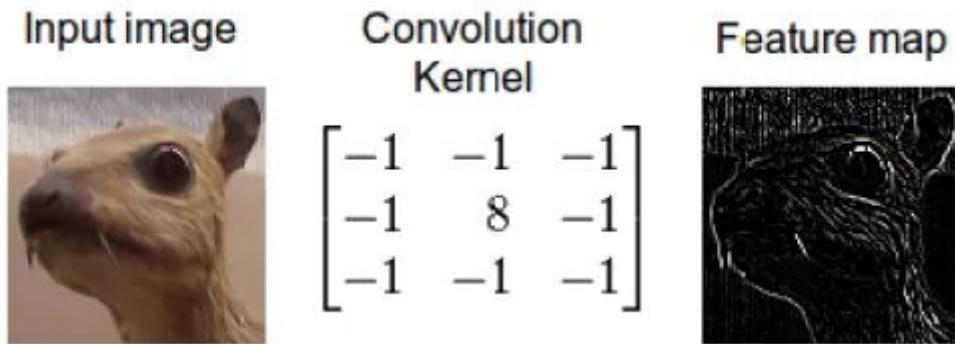


Imagen de [este artículo](#).

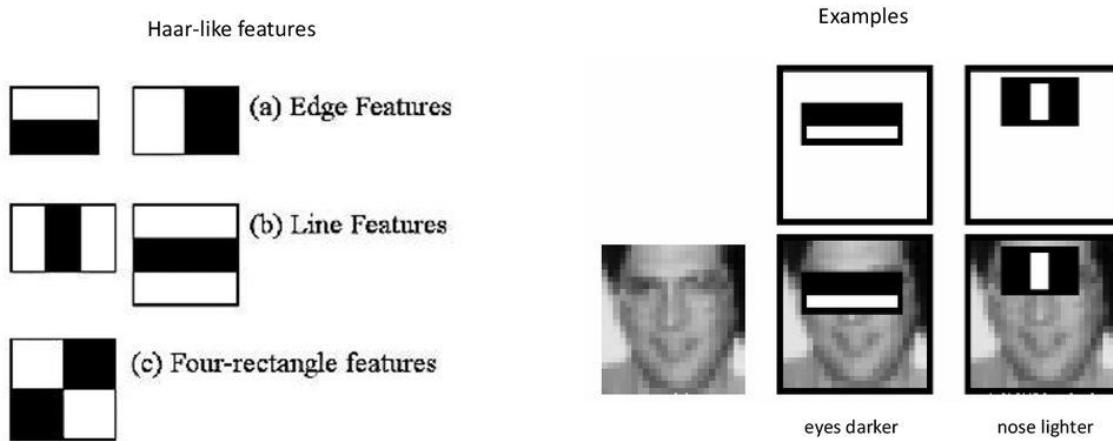
Estas técnicas de procesado de matrices extraen la información que necesitamos para luego concluir que hay en esa imagen. **Extraer información significa reconocer patrones** por lo que esas técnicas son formulaciones matemáticas, estadísticas y algoritmos que, en general, existen desde hace decenas de años. Básicamente reconocen contrastes, brillos, cambios de intensidad mediante el uso de filtros. Resaltar bordes, esquinas o formas particulares de los objetos.



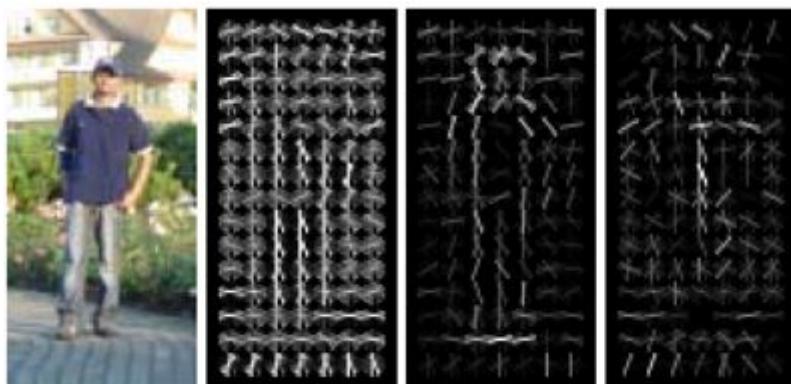
Los filtros también se conocen como kernels. Aplicarlo consiste en multiplicar la matriz de 3x3 a toda la imagen normalmente en escala de grises. El resultado se compara con otros previamente validados por un humano

Old school: Viola-Jones

Haar Feature-based Cascade Classifiers



SIFT-like Histogram of Gradients (HoG)



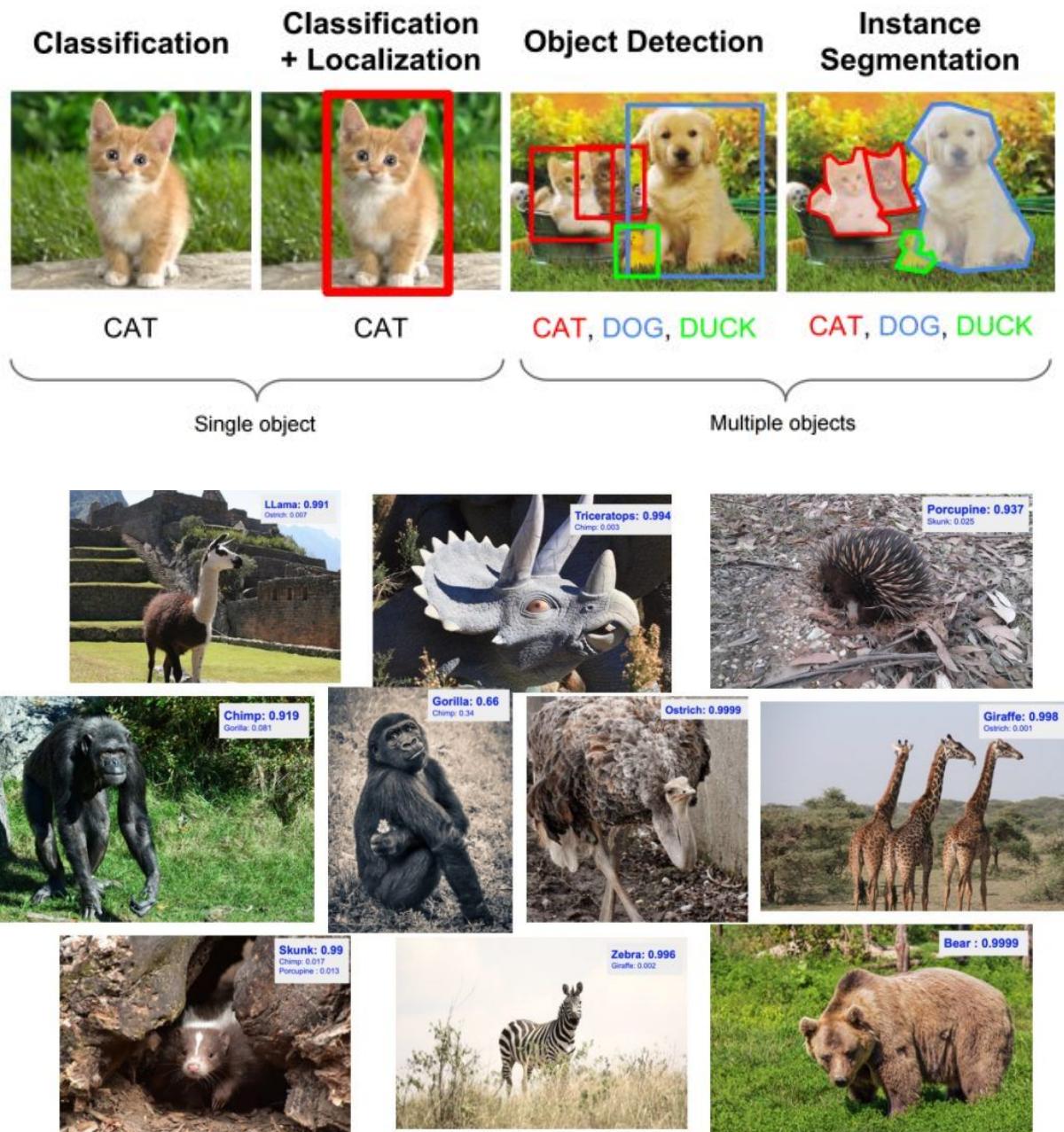
[Artículo](#) con la Imagen

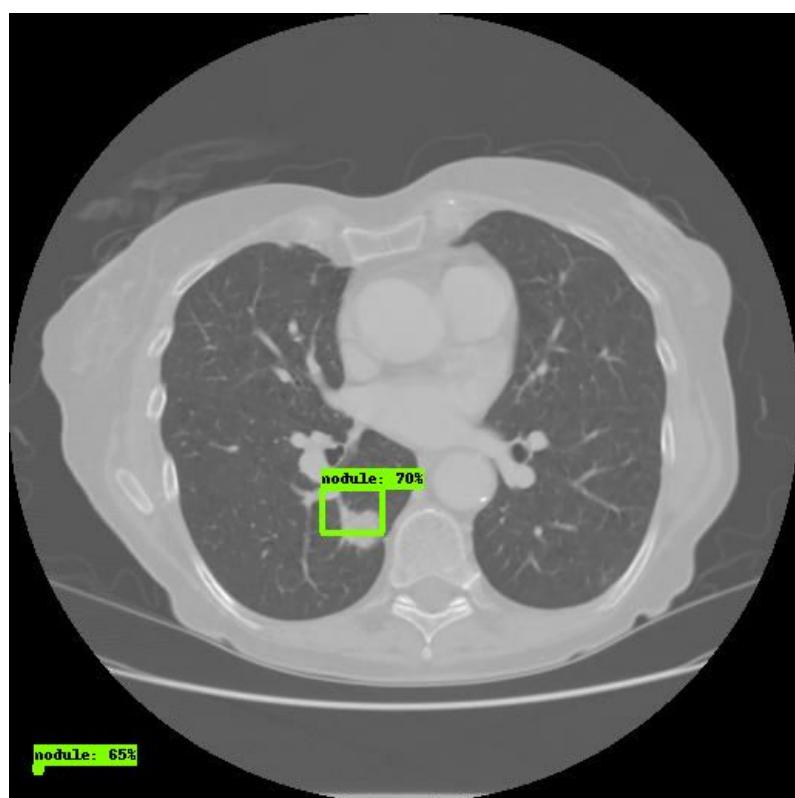
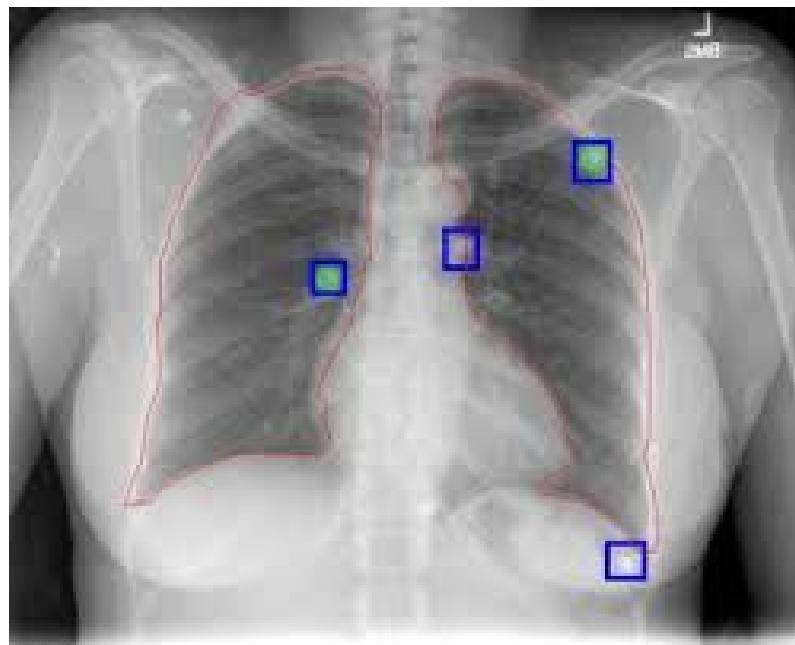
Podéis profundizar en estas técnicas [con este libro](#).

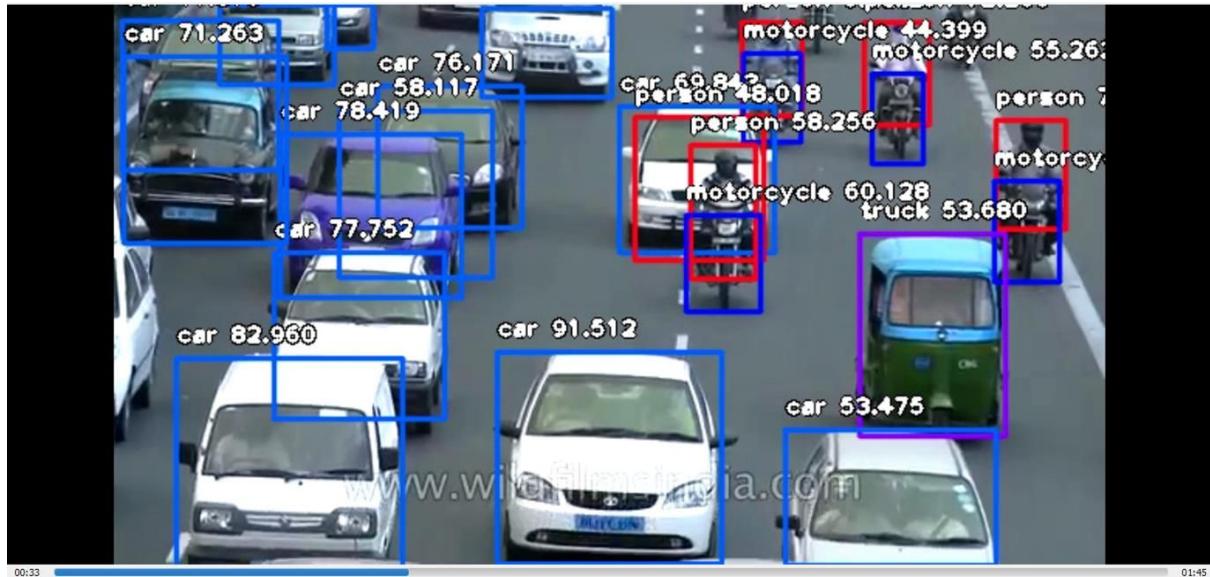
Esta disciplina cambió dramáticamente con la aparición del deeplearning allá por 2012 con la aparición del paper de Alexnet.

Categorizar nuestro problema a resolver

Enfrentándonos a proyectos en computer vision tenemos tres grandes tipos de problemas a resolver.







Es muy importante enmarcarse en uno de estos tipos cuando empezamos un proyecto. Los problemas son más complejos y difíciles de resolver según más clases añadas. Igualmente de clasificación a segmentación se incrementa la complejidad. Las empresas con recursos resuelven la complejidad con más y más imágenes etiquetadas.

Construyendo nuestro dataset

Problemas que se repiten constantemente

Multi-modal object recognition

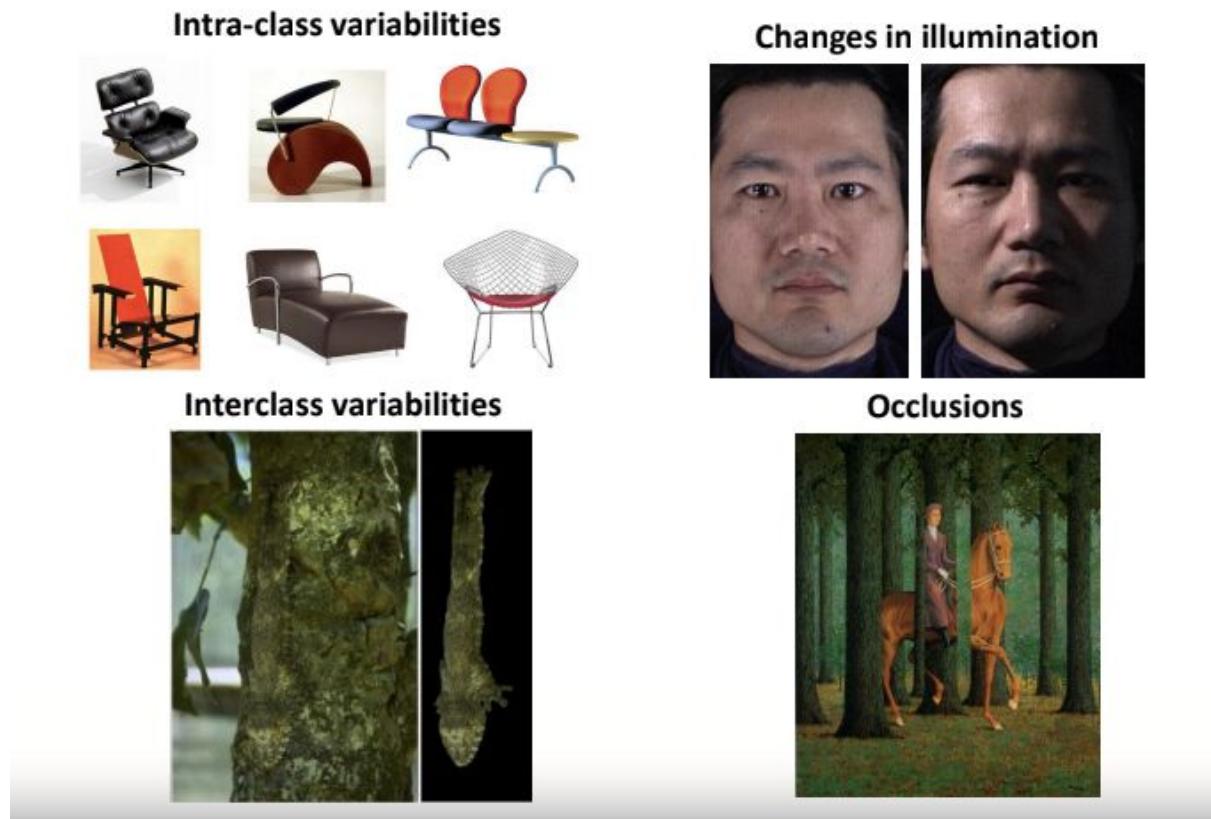


Imagen del curso sobre Object (and human) Recognition. Human Pose Recovery and Behavior Analysis Group.
Sergio Escalera Guerrero y Simone Balocco

Dataset: conjunto de imágenes con la información asociada a esas imágenes según el problema a resolver: clase, clase + coordenadas

Clases: output a extraer: (perros, gatos, personas, caras, tumores, emociones en una cara...)

Bounding boxes: 4 coordinadas con las que se construye un rectángulo que rodea el objeto (polígonos en el caso de segmentación)

¿Cuántas imágenes necesito?

Para poder reconocer automáticamente caras en cualquier situación, necesitamos que la máquina sea capaz de hacer una representación genérica de lo que es una cara, y esta representación tiene que ser invariante a todos estos cambios.

¿Qué ha ocurrido en los últimos 8 años?

Datasets

Son de acceso libre y producto de Android y google + Facebook + Microsoft (COCO) + grandes challenges de universidades (ImageNet) con el esfuerzo de trabajo mecánico, mal pagado y no reconocido (amazon turk y compañía). Conoced los tres datasets más importantes en [este artículo.](#)

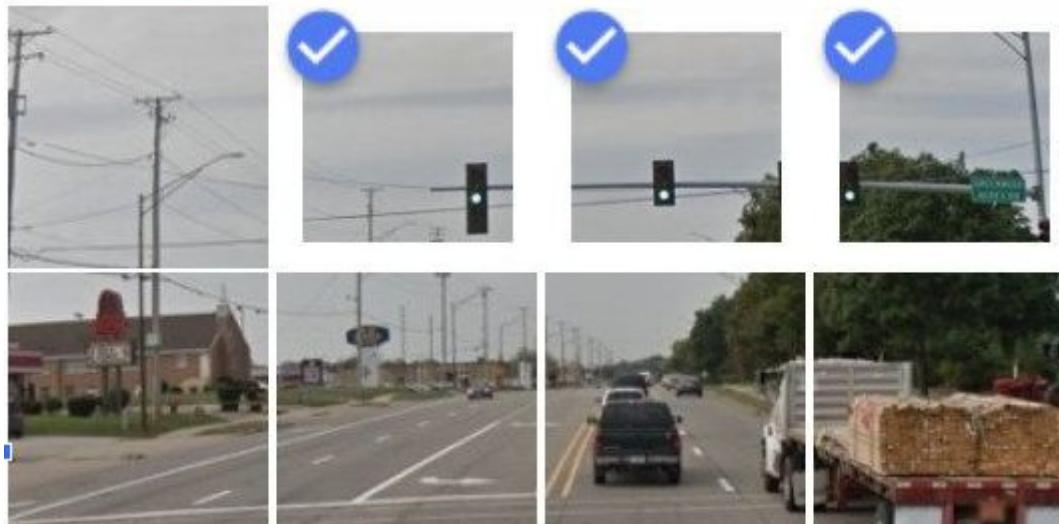




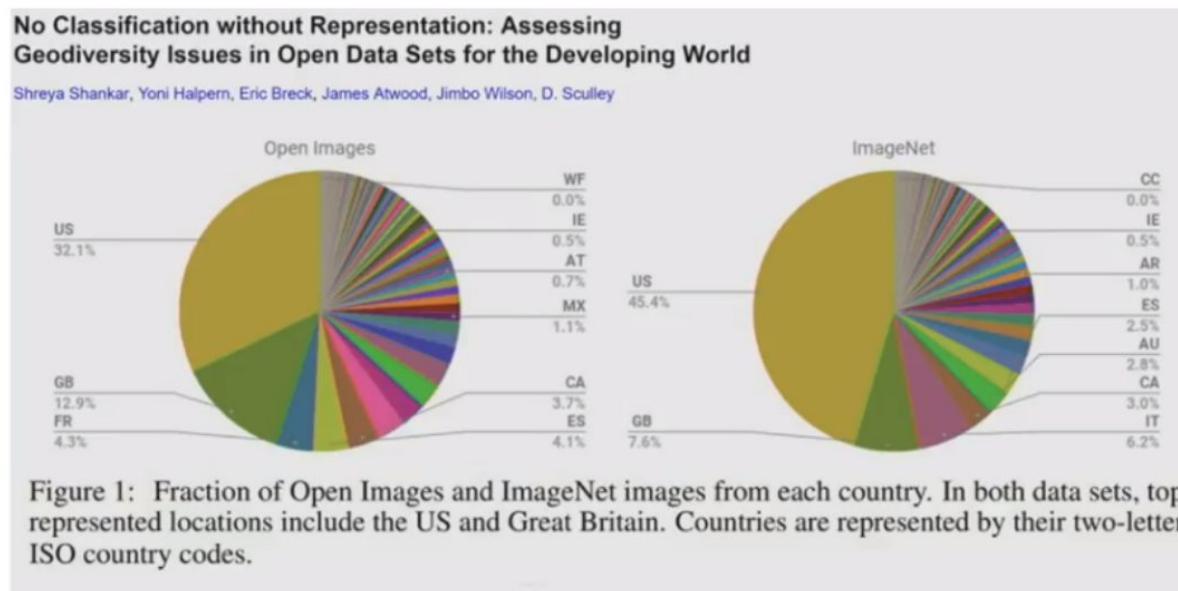
Maneras ingeniosas

Selecciona todos los cuadrados que contengan
semáforos

Si no hay ninguno, pulsa Saltar



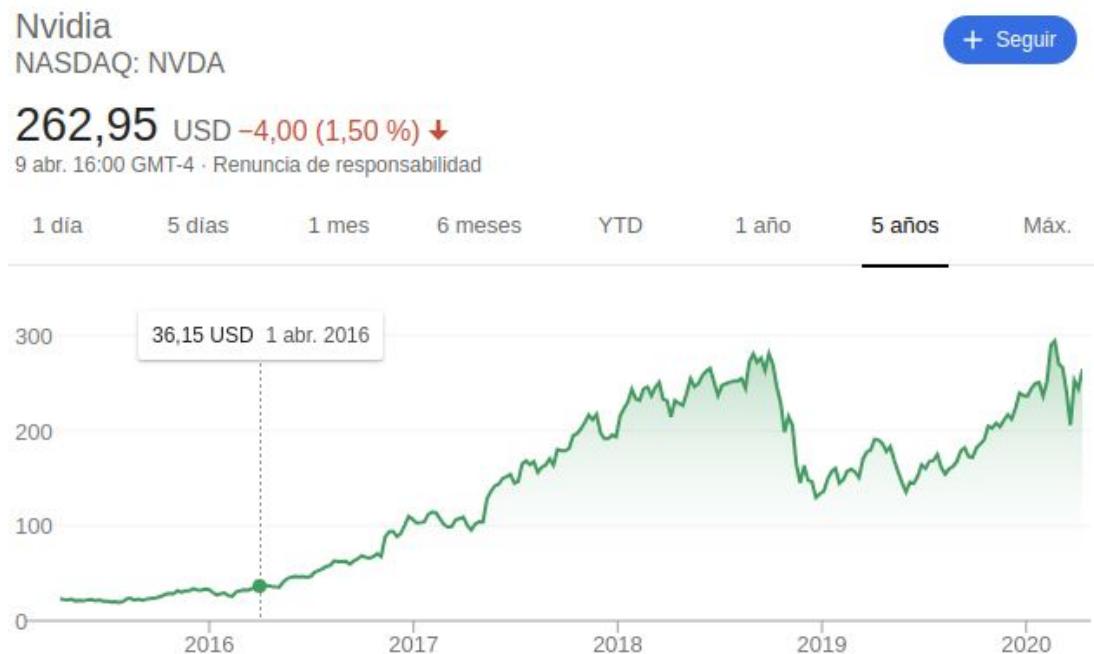
Problema grave: biased datasets o sesgados



El dataset es la parte más importante mucho más que arquitecturas, modelos, algoritmos... En general os llevará más tiempo, os costará más dinero y será la base de unos buenos resultados

Computación

Gpus de 16 gigas a disposición.



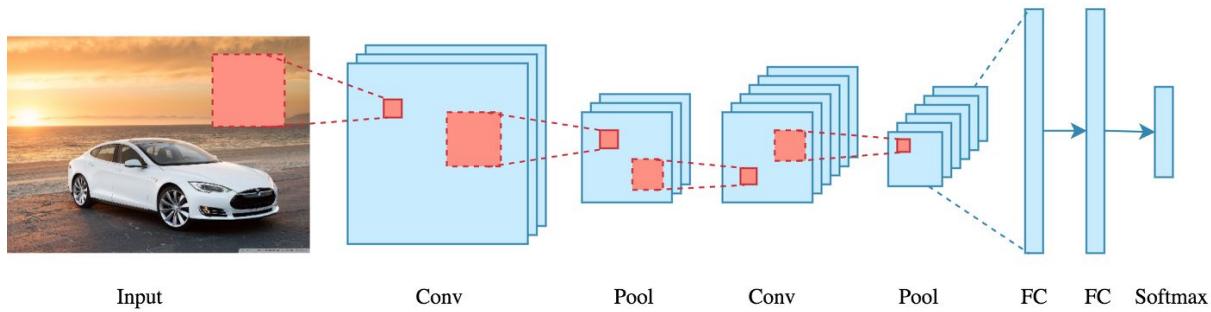
Deep Learning

Cambio radical que revolucionó toda la disciplina. Deep Learning (para generalizar redes convolucionales de decenas de capas) es un cambio de paradigma. Construimos nuestros desarrollos de manera totalmente diferente y además funcionan mejor: recogemos decenas de miles de ejemplos, los etiquetamos, los pasamos por una red neuronal y esta encuentra (ella sola) como reconocerlas. Pero que no nos engañen, **esto es fuerza bruta**. Suma de centenas de miles de imágenes etiquetadas y fuerza computacional (aka GPUs)

Arquitecturas y modelos pre-entrenados

Imágenes extraídas de [este artículo](#) que recomendamos leer para entender más.

Casi todos las arquitecturas CNN tienen este tipo de capas.



Convolución

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

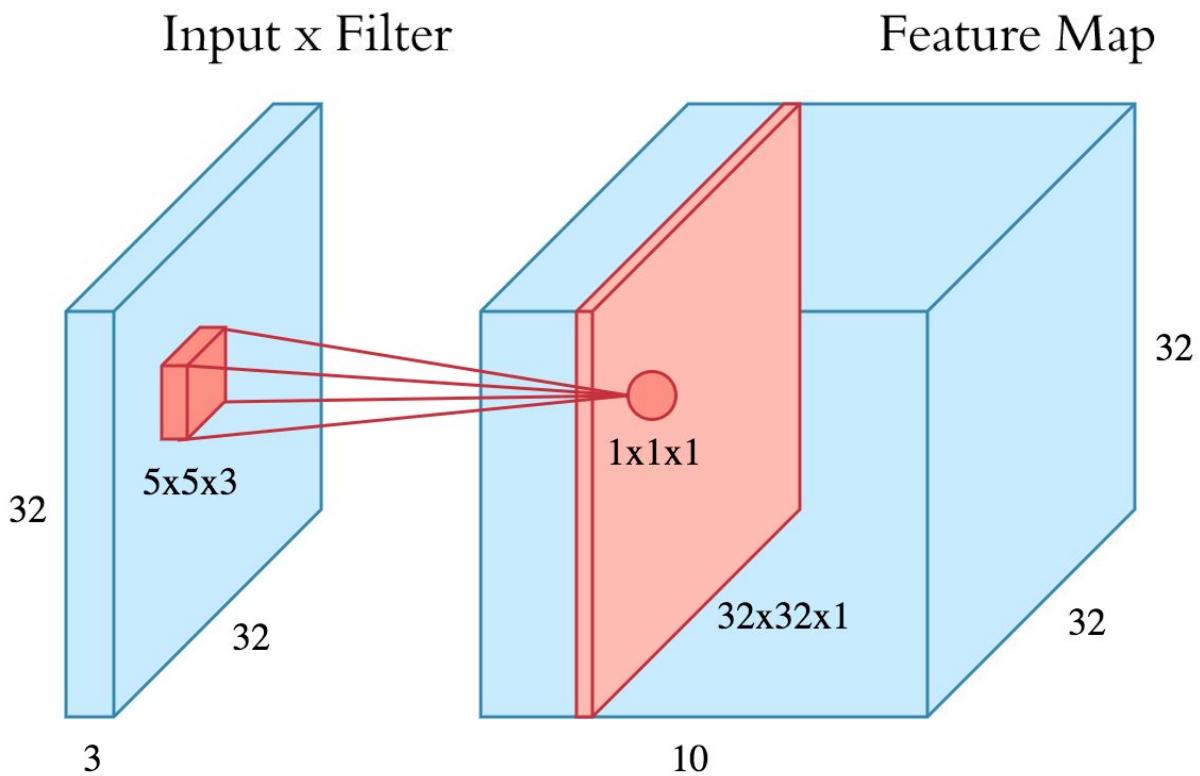
Input

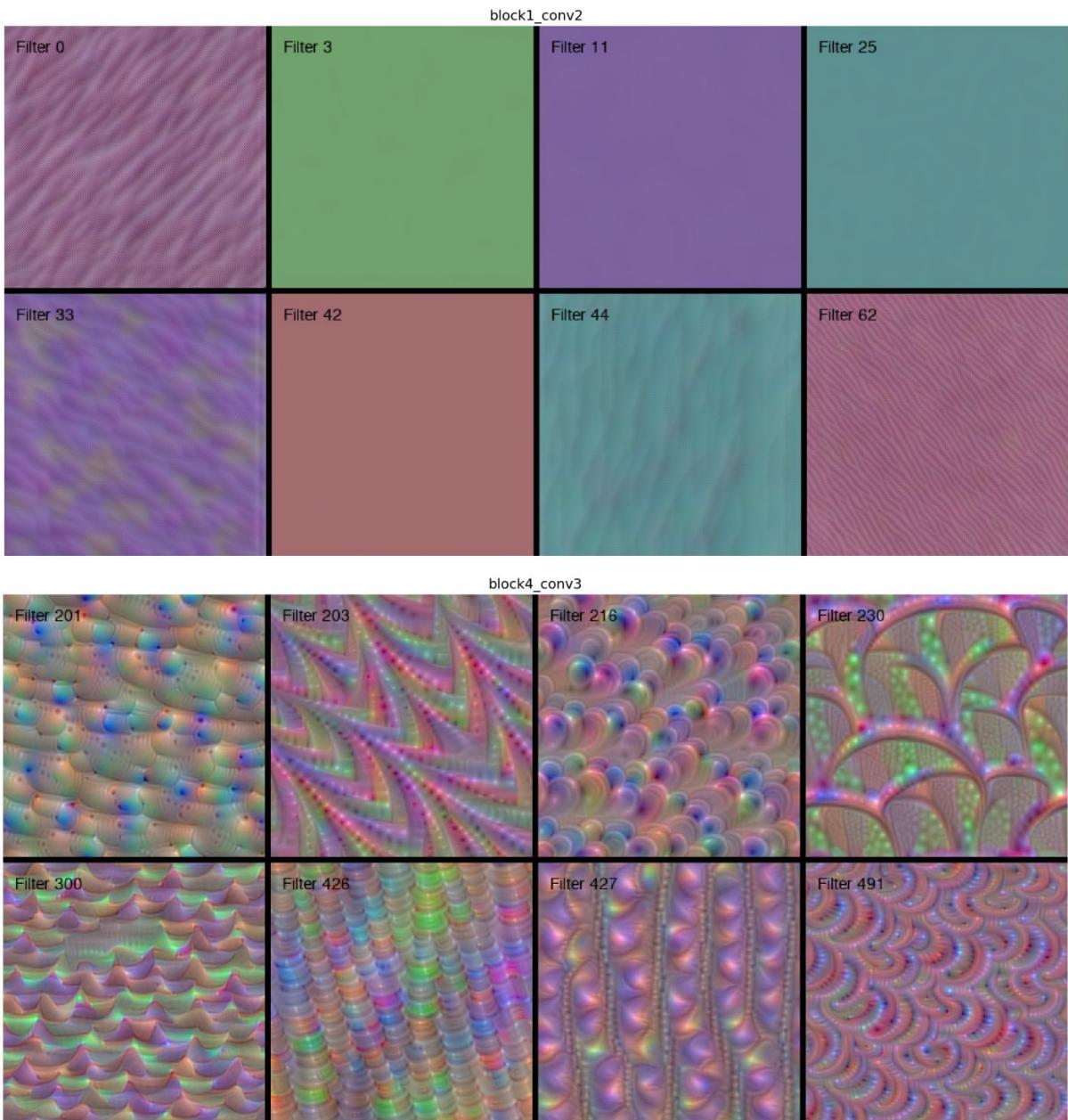
1	0	1
0	1	0
1	0	1

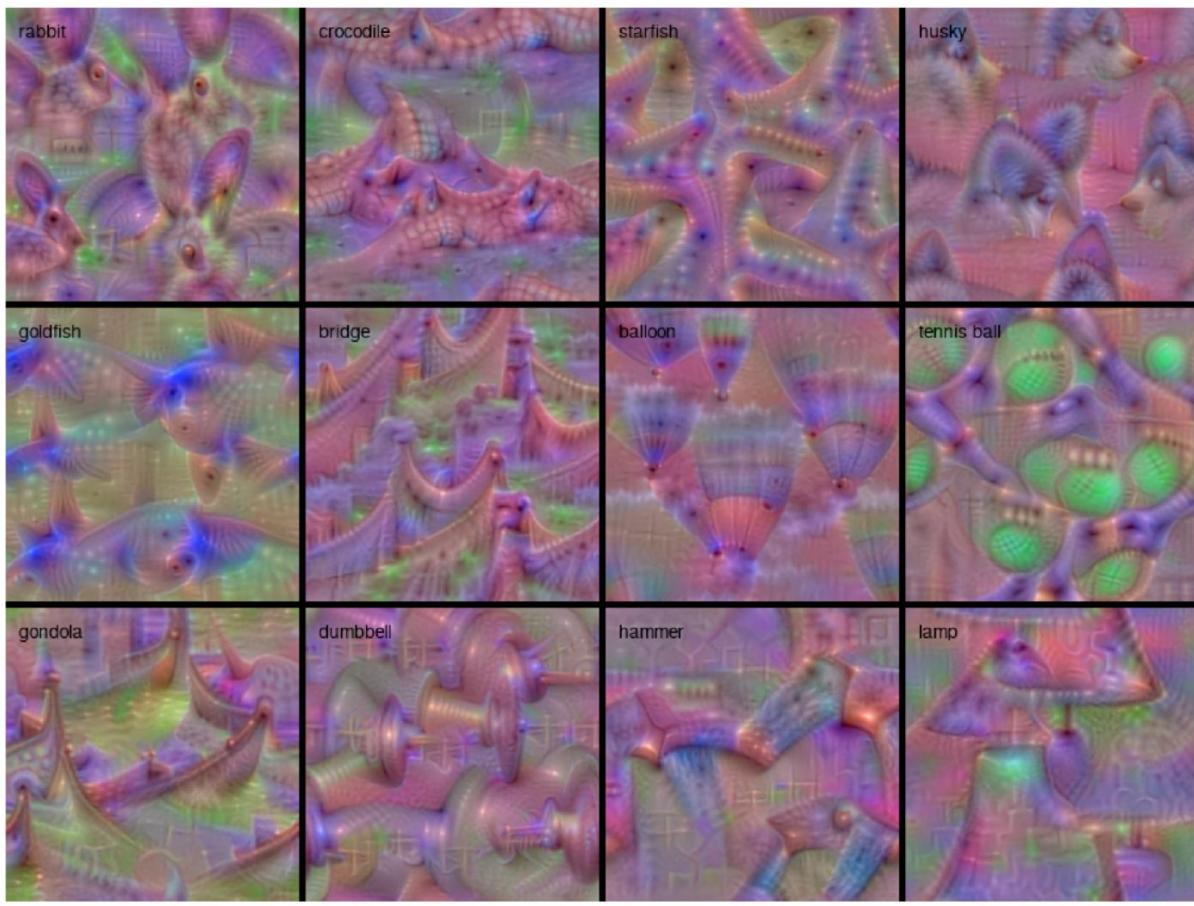
Filter / Kernel

1	1x1	1x0	0x1	0
0	1x0	1x1	1x0	0
0	0x1	1x0	1x1	1
0	0	1	1	0
0	1	1	0	0

4	3	



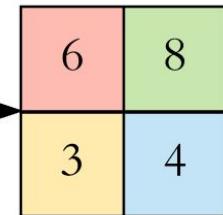




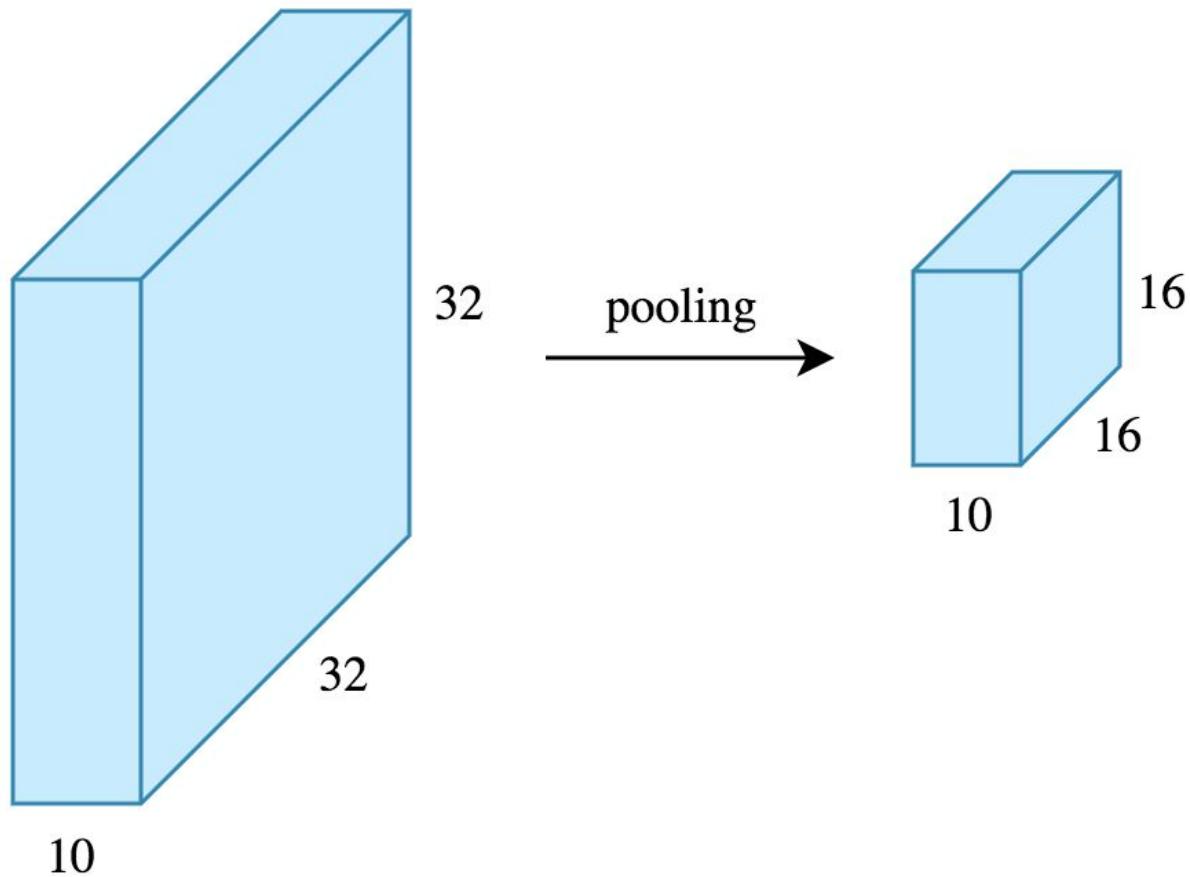
Pooling layers

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

max pool with 2x2
window and stride 2

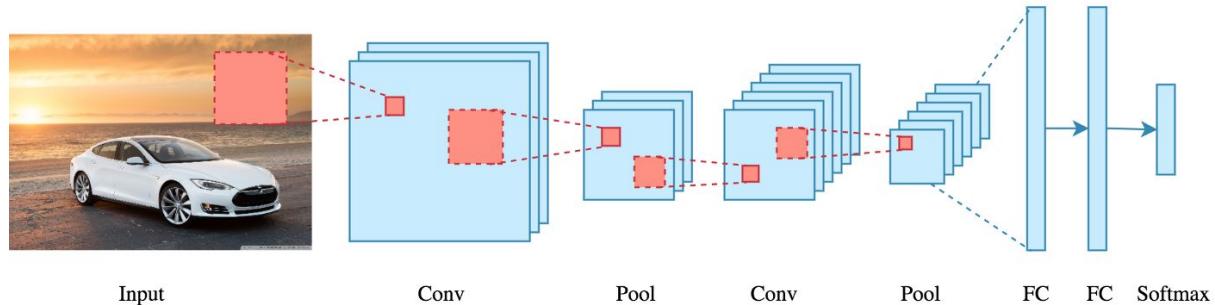


6	8
3	4



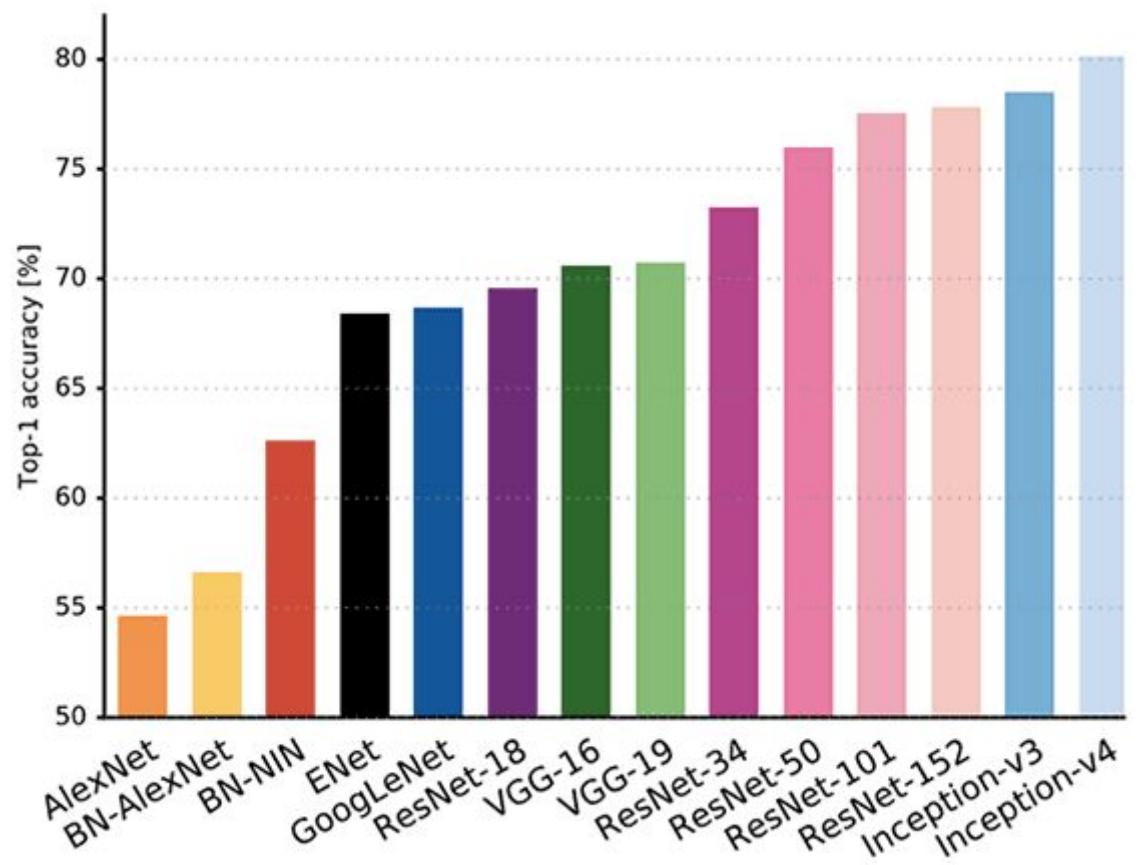
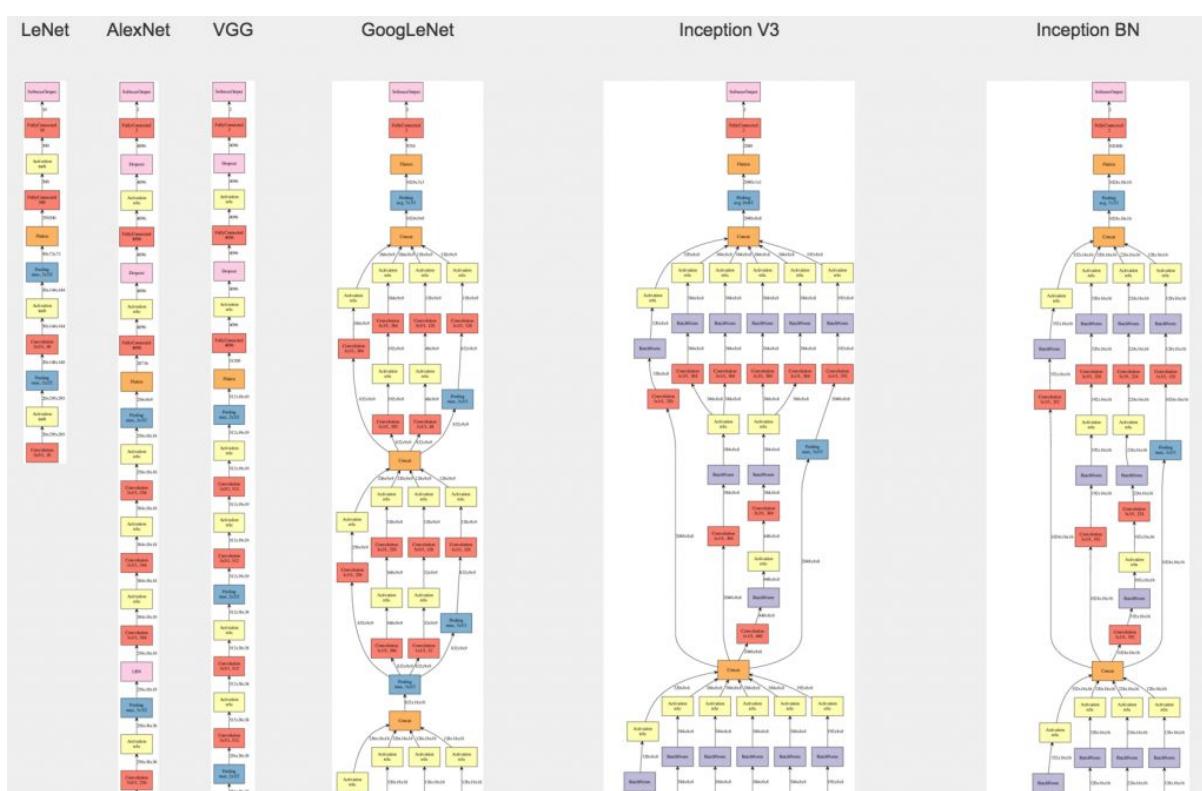
Output Layers

Una arquitectura CNN combina la extracción de features en las capas convolucionales y la parte clasificatoria en las capas FC.



Depende del problema a resolver en el caso de clasificación de clases tendremos **una capa softmax** que nos dará un porcentaje de cada clase {gato:0.98,perro:0.02}. El gran problema (que veremos en mayor detalle) es que si tienes una clase que no aparece en el dataset esta capa seguirá dando resultados que sumen 1.

Diferentes arquitecturas



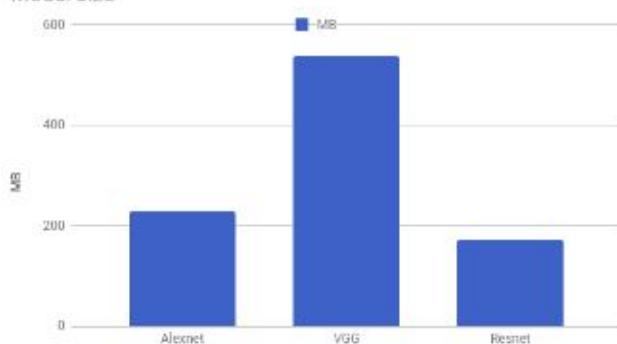
Pros y contras de cada modelo

El peso final de un modelo está en las últimas capas pero el tiempo de entrenamiento se va en las capas Conv.

P100 vs K80

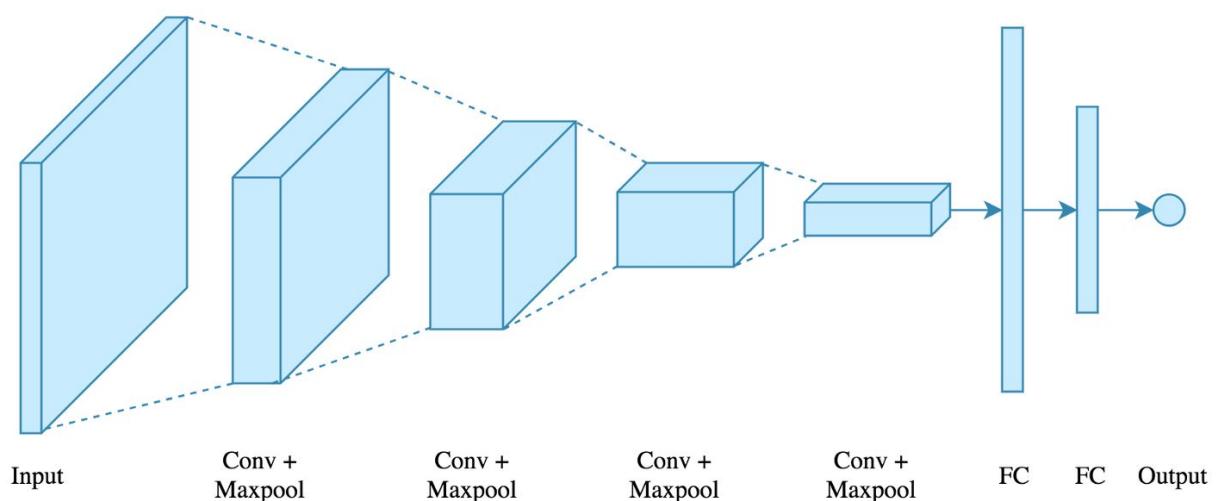


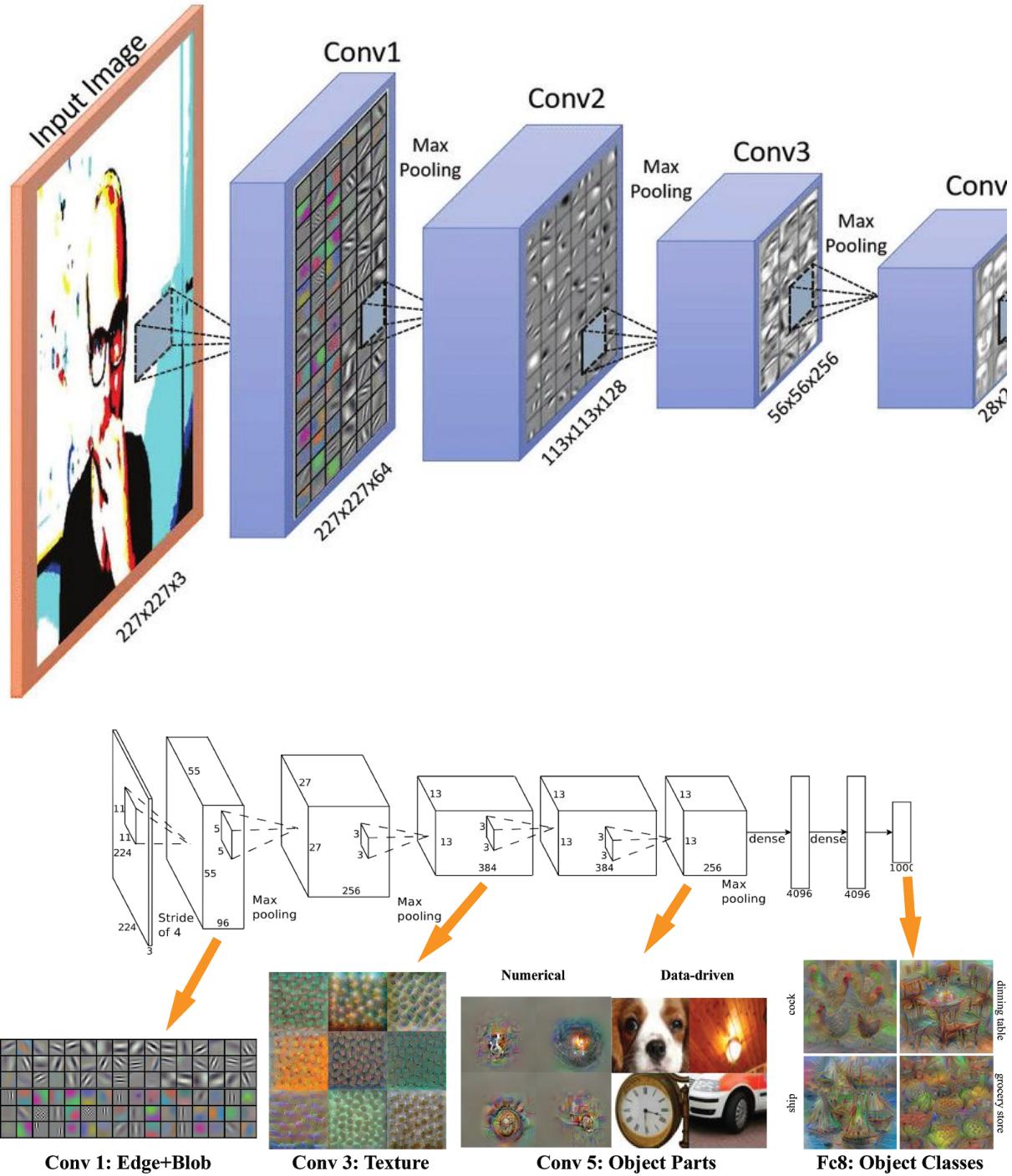
Model size

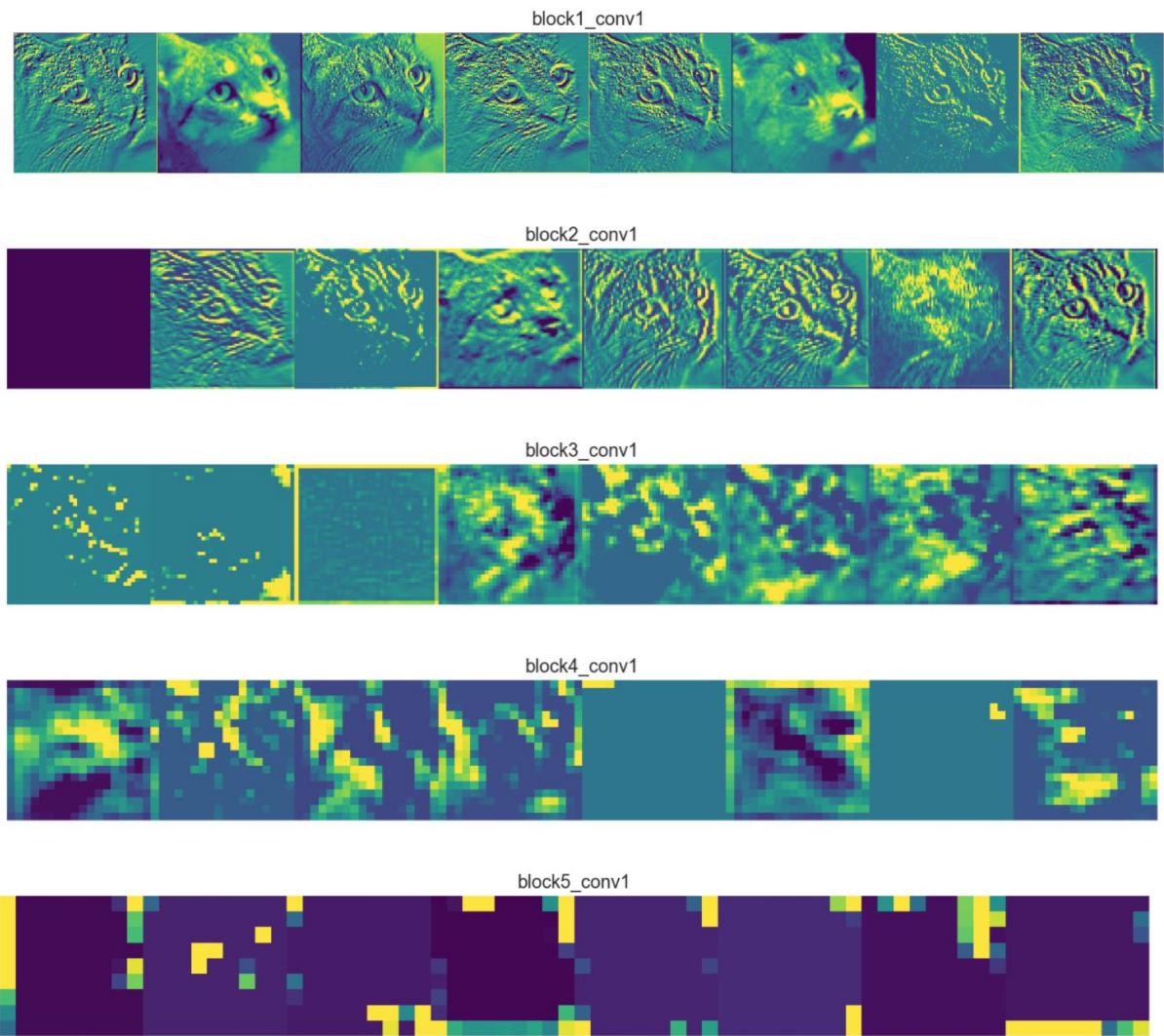


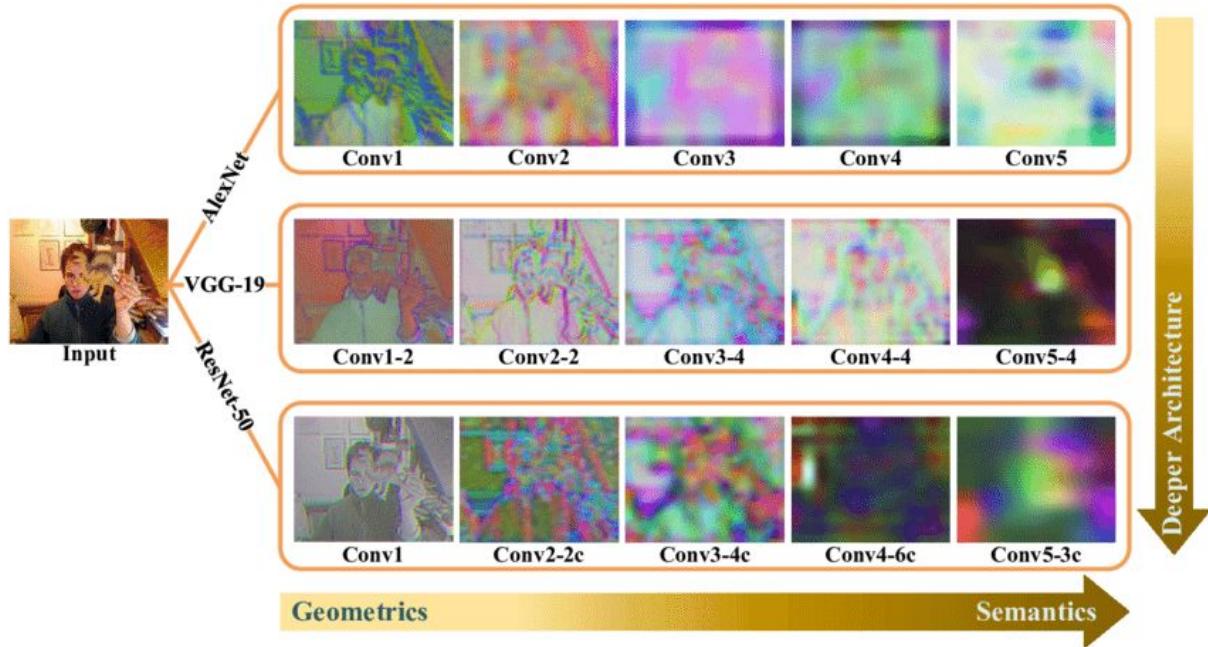
Model Size:
Alexnet: 228MB
VGG16: 537MB
Resnet 101: 171MB

Visualización de las capas







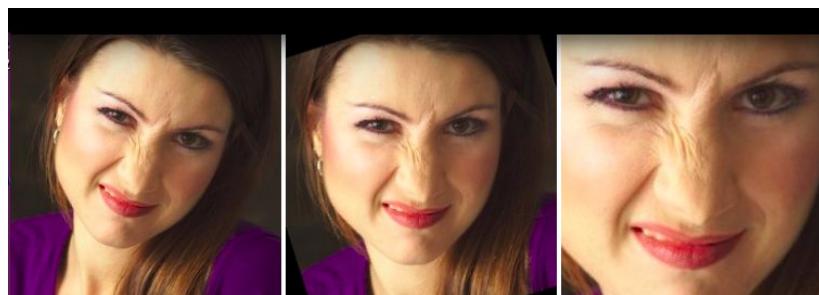


Entrene de CNN

Glosario:

Preproceso de inputs: trato de entrenar la red con el máximo de información posible.

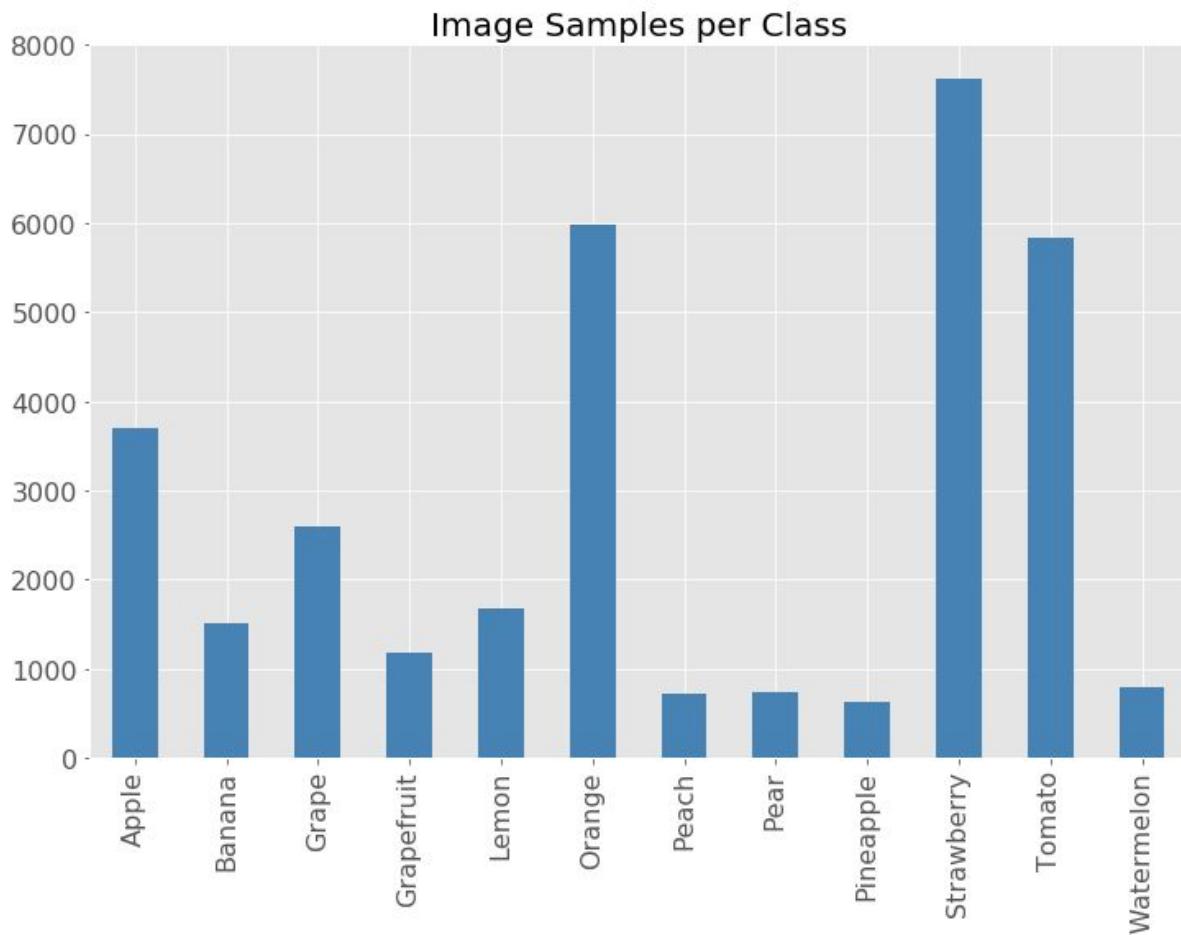
- Normalización Extraigo mediana de imágenes
- Transformaciones: roto y recorte
- Extraigo outliers y ejemplos excepcionales



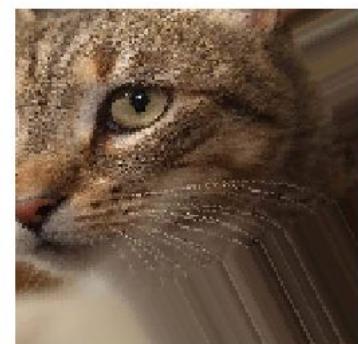
Os dejo [un ejemplo](#) muy bueno sobre un pipeline de detección de caras

Data augmentation:

Para datasets no balanceados o pocos datos



Generar ejemplos artificialmente



Evitamos el overfitting

Preparando el dataset

Dividir el dataset en 3 partes:

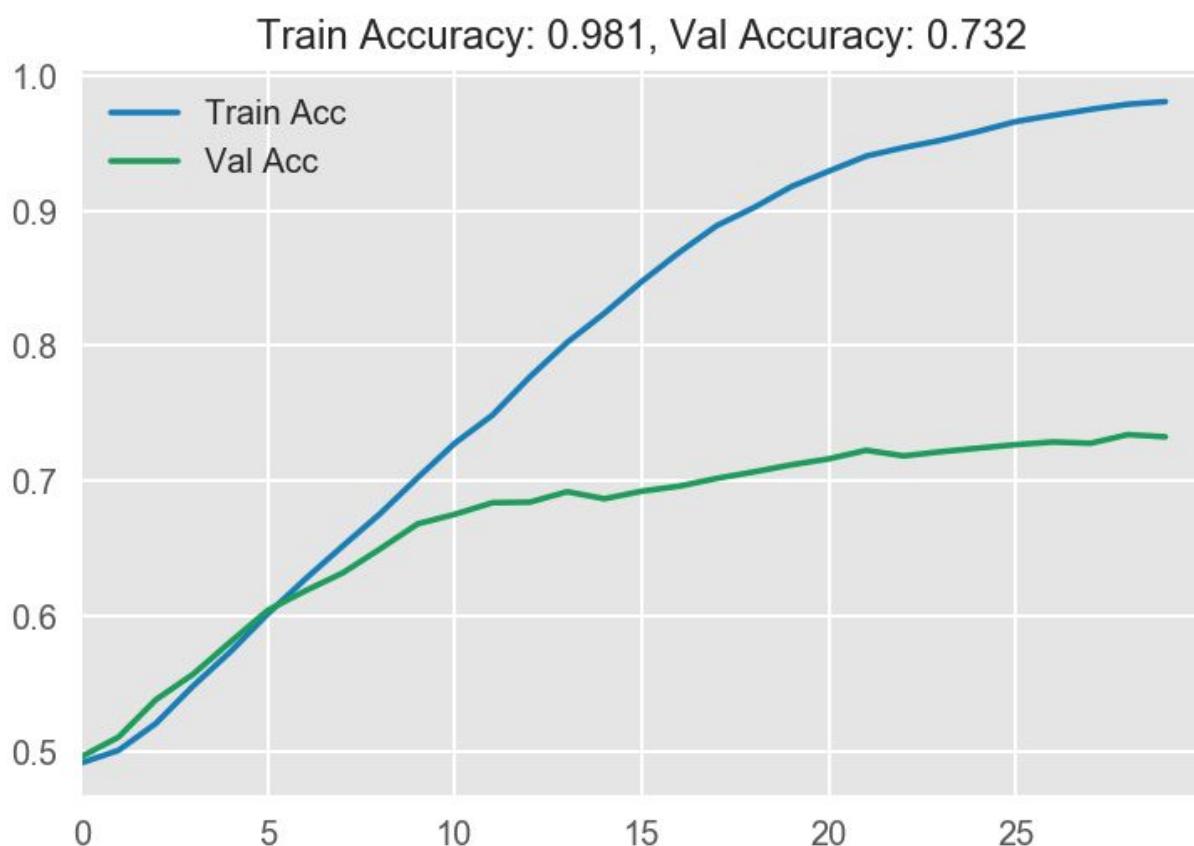
- Training set: 70% de las imágenes
- Validation set: 15% de las imágenes
- Test set: 15% de las imágenes

Asegurarse que sea representativo del dataset: mismo balanceo de clases, ejemplos de todo tipo en los 3 sets. Poner aparte el test set y no tocarlo nunca.



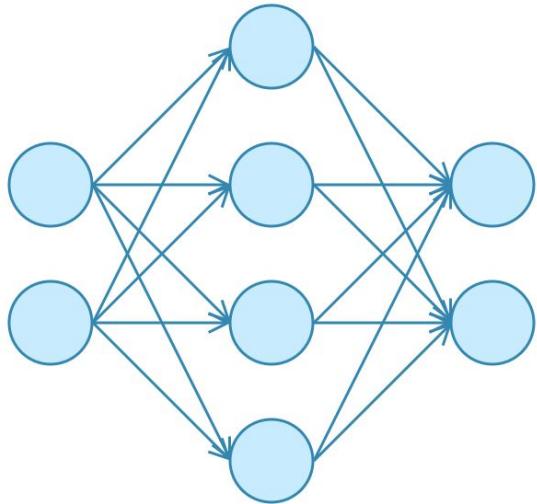
Overfitting

Es nuestro gran enemigo. Ocurre cuando nuestro modelo se está adaptando demasiado al training set y **no generaliza**.

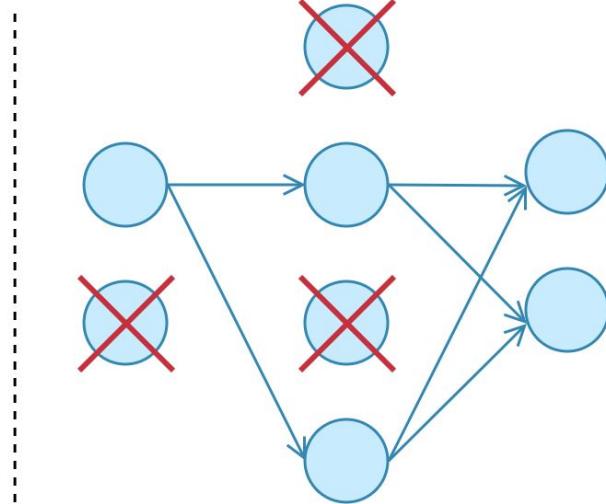


Hay técnicas para tratar de evitarlo pero al final consiste (como siempre) en tener más datos

y más variados por eso jugar al máximo con el data augmentation. Si ya no podemos conseguir más datos tenemos que pasar a aumentar el dropout



No Dropout



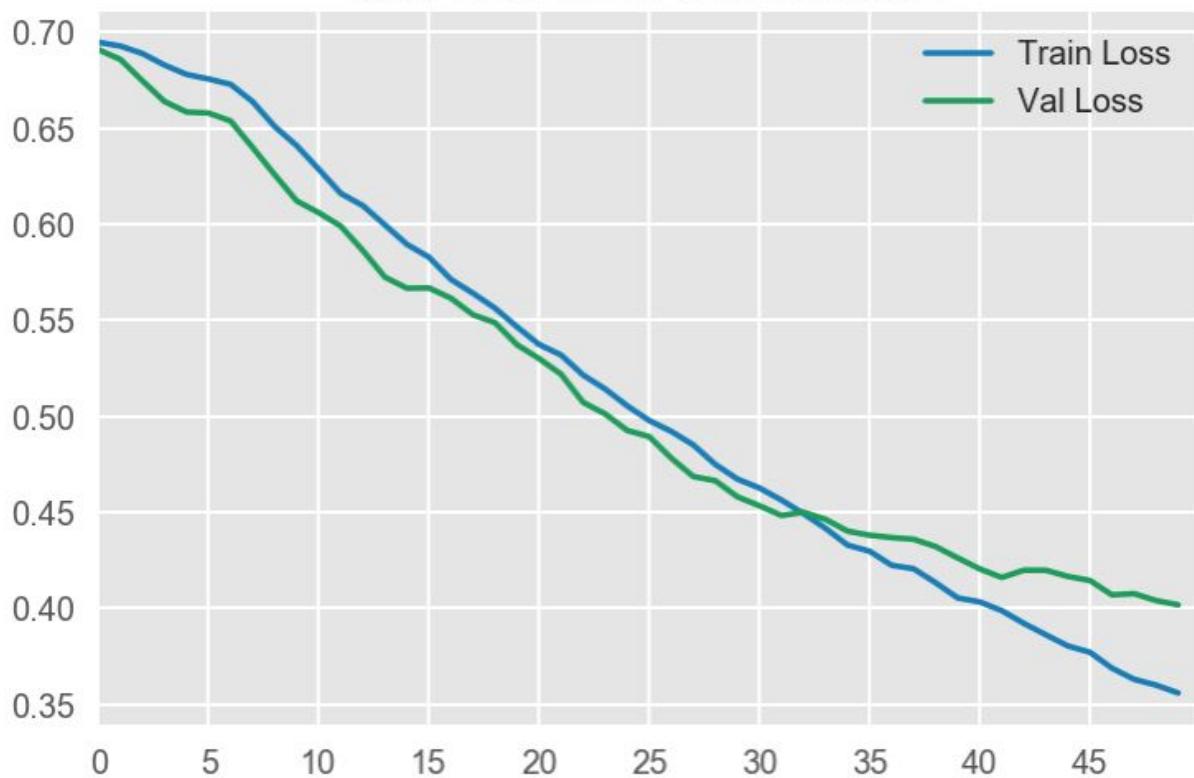
With Dropout

¿Cómo sabemos que un modelo está entrenando correctamente?

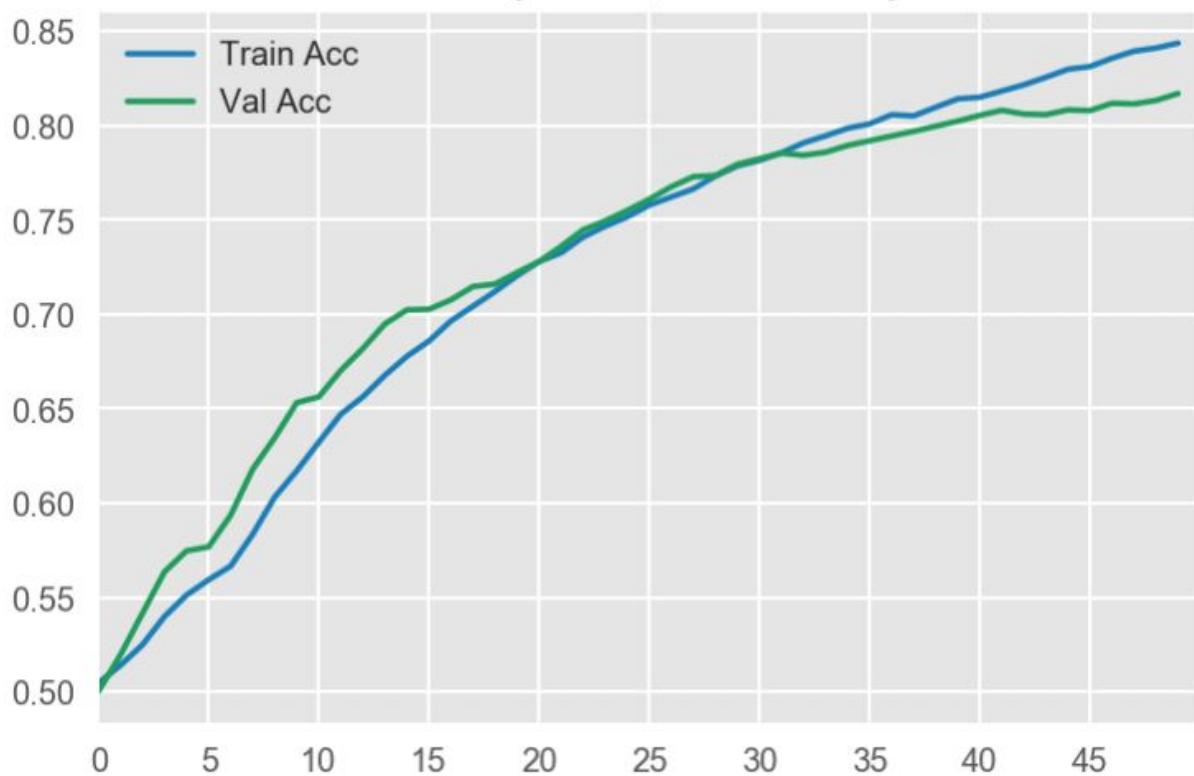
Hemos revisado que cumplamos el checklist más fundamental:

- Data preprocessing (tamaño imagen y regularización)
- Dataset balanceado
- Splits equilibrados
- Métricas elegidas

Train Loss: 0.356, Val Loss: 0.402

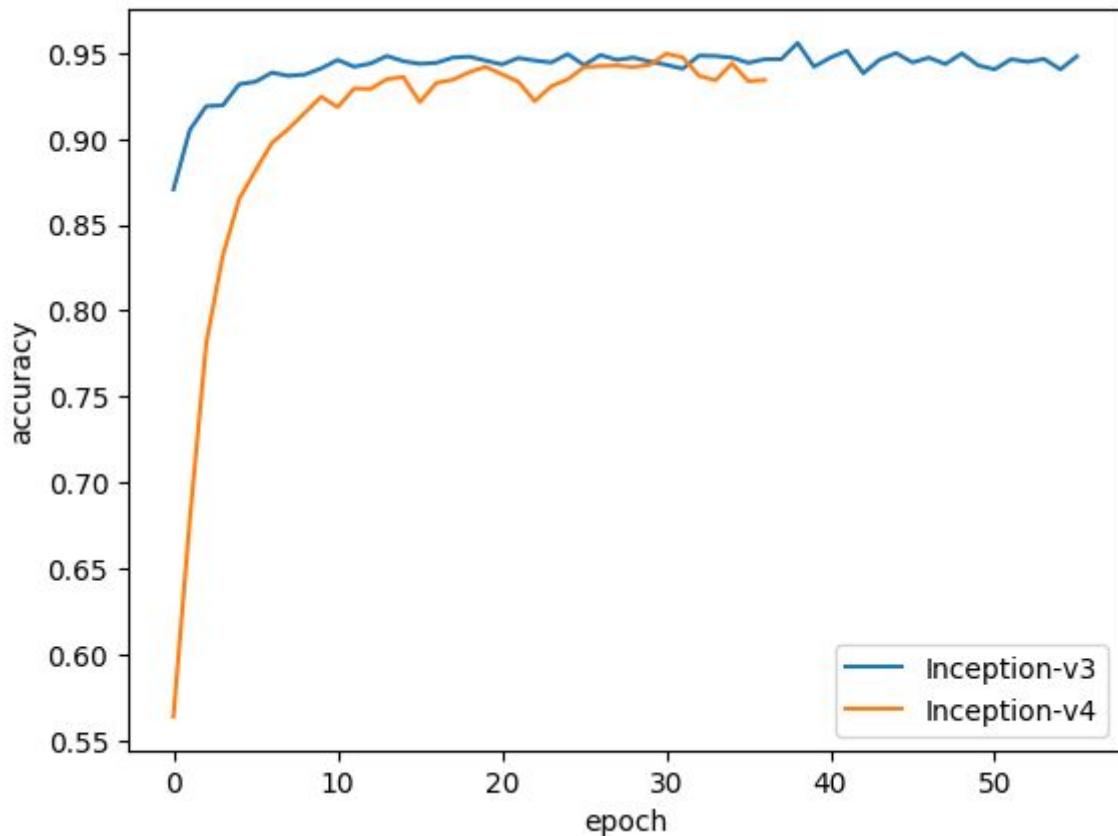


Train Accuracy: 0.844, Val Accuracy: 0.817



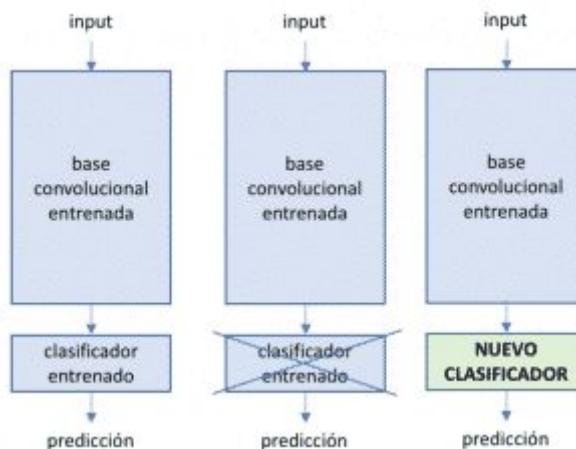
¿Cuándo parar de entrenar?

Entrenamos un número de **epochs** y el framework va haciendo **snapshots** de cada modelo entonces paramos cuando las curvas alcanzan el plateau (se allanan)



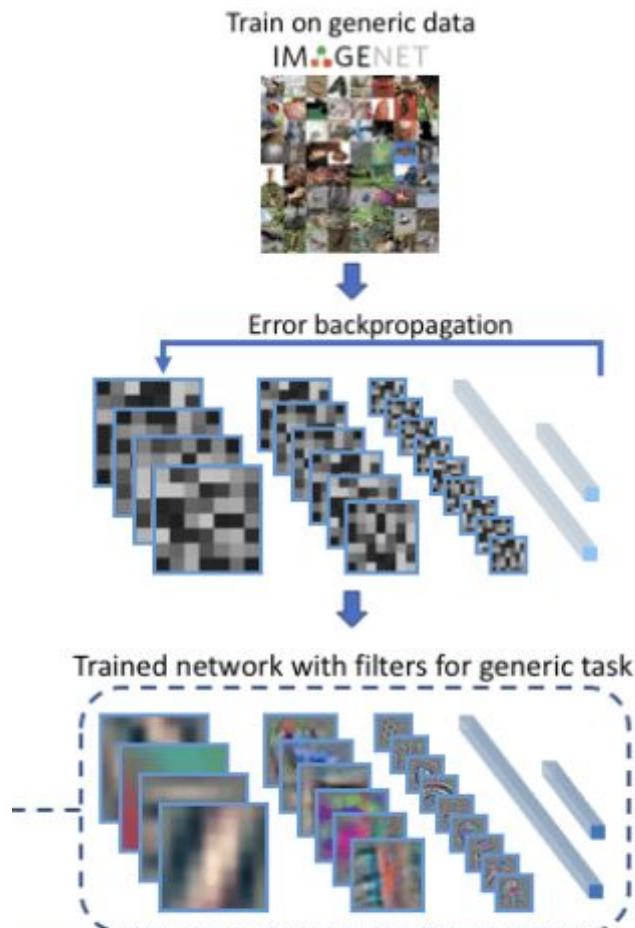
Transfer learning

Es nuestro gran amigo. Consiste en usar un modelo ya entrenado (con millones de fotos y semanas entrenando) y solo cambiar las últimas capas para adaptarlo a nuestro problema. Las ventajas son en tiempo de entrenamiento y resultados pero puede no funcionar si el problema a resolver es muy diferente del dataset que se usó para entrenar el modelo.

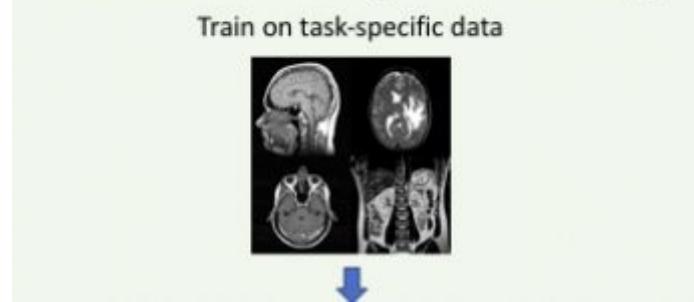


Ejemplo que no me parece correcto:

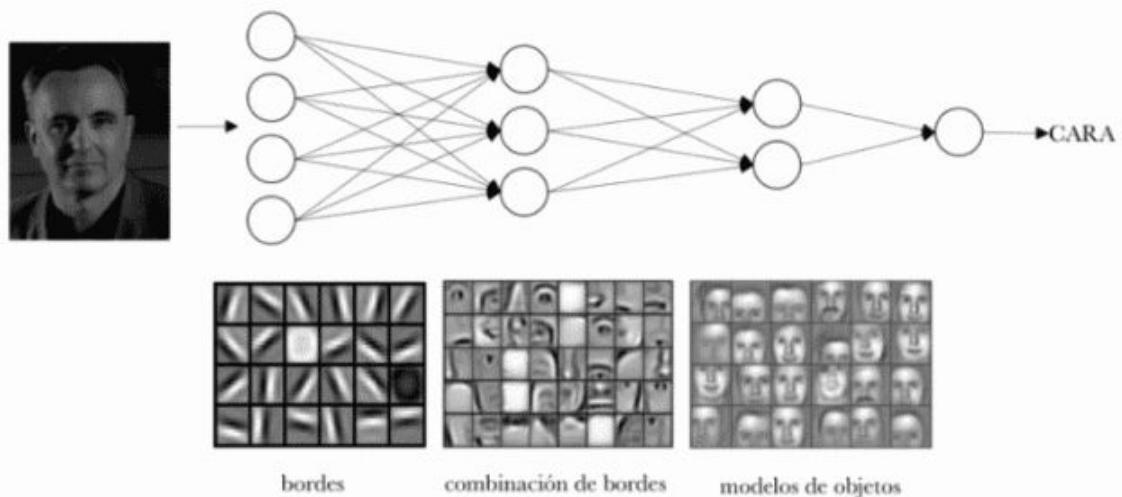
Pre-training for transfer learning



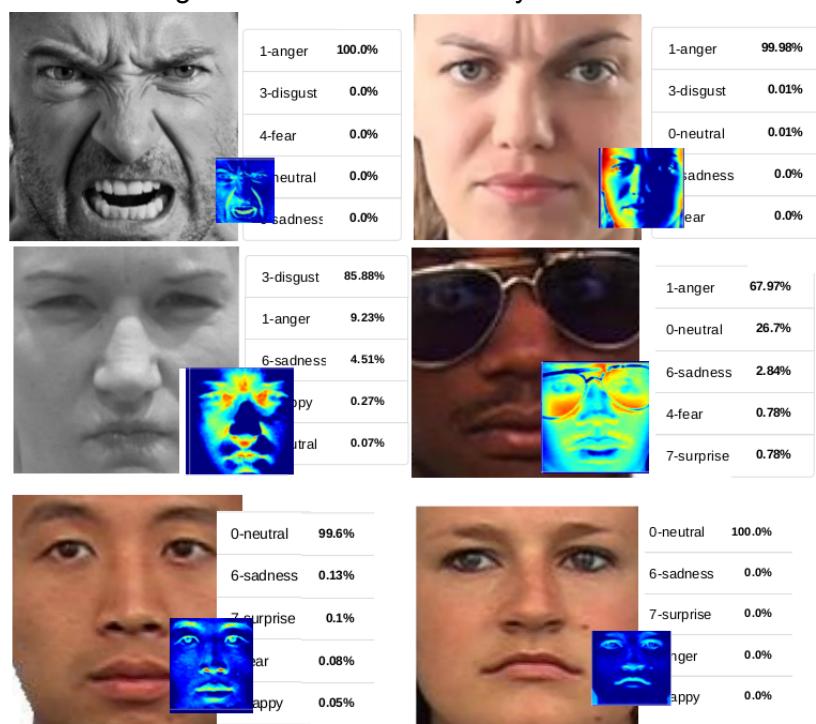
Transfer learning - fine-tuning



Tienes que cambiar las FC capas y adaptarlas al número de clases de tu problema.

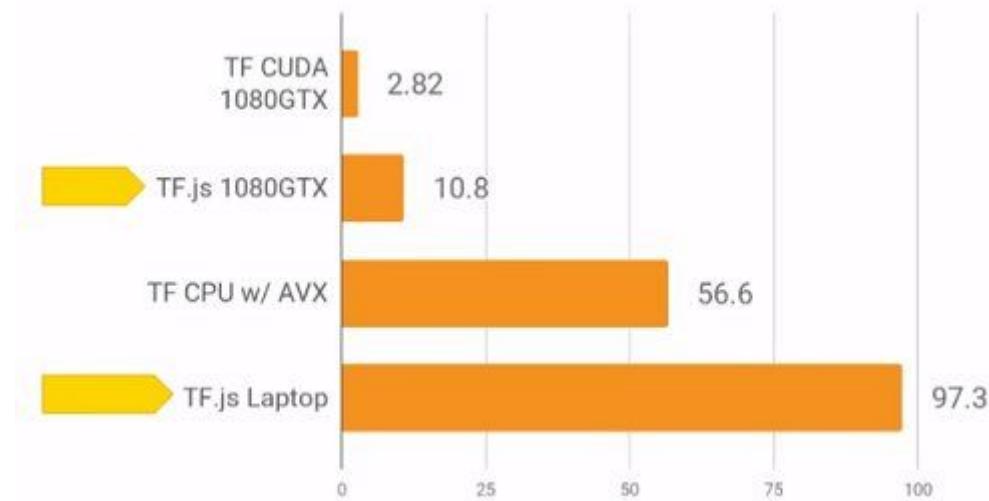


Este uso de transfer learning funciona correctamente y tiene sentido.

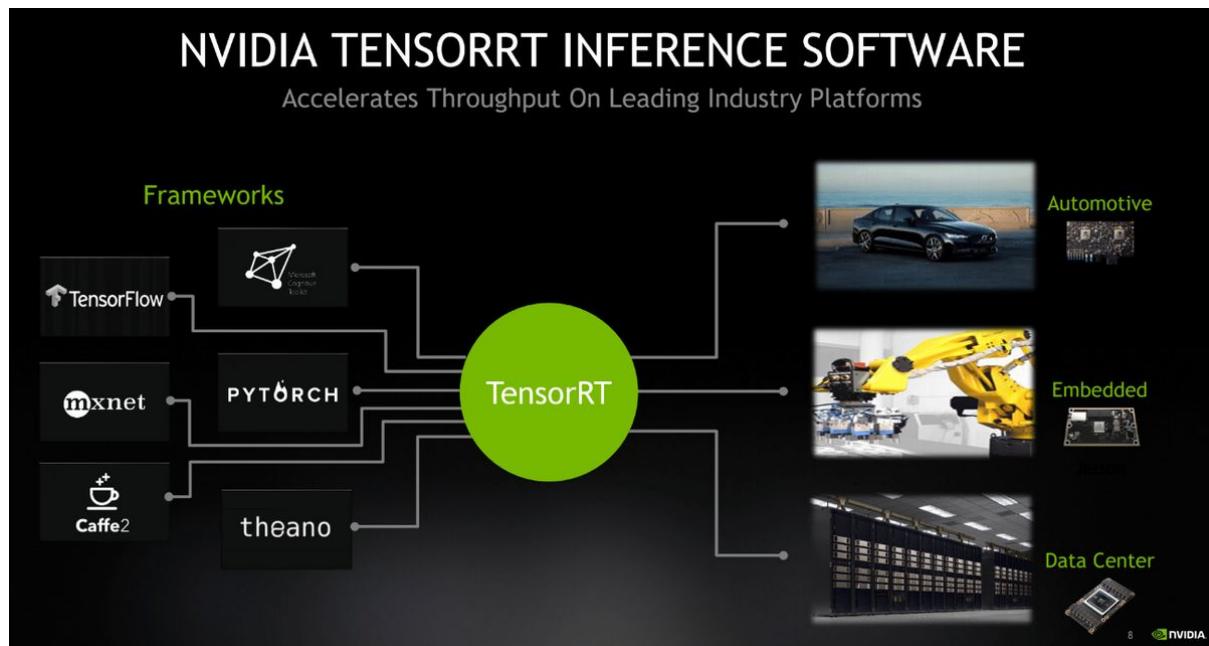


Puesta en producción

Se llama inference y TensorFlow tiene las herramientas más avanzadas para ello si haces inference en el cloud.



Pero Nvidia tiene herramientas mas avanzadas si haces inference on the edge



NEW JETSON FAMILY

Top-to-Bottom Embedded AI Computer Lineup

<p>JETSON NANO</p>  <p>0.5 TFLOPS (FP16) 5-10 W 45 mm x 70 mm \$129</p>	<p>JETSON TX2 SERIES (TX2, TX2 4GB, TX2i*)</p>  <p>1.3 TFLOPS (FP16) 7.5-15 W* 50 mm x 87 mm Starting at \$249</p>	<p>JETSON XAVIER NX</p>  <p>6 TFLOPS (FP16) 21 TOPS (INT8) 10-15 W 45 mm x 70 mm \$399</p>	<p>JETSON AGX XAVIER SERIES (AGX Xavier 8GB, AGX Xavier)</p>  <p>20-32 TOPS (INT8) 5.5-11 TFLOPS (FP16) 10-30 W 100 mm x 87 mm Starting at \$599</p>
One Software Architecture			

Problemas en deep learning computer vision.

A parte de la gran cantidad de datos necesarios y recursos computacionales (medio ambiente) tenemos otros graves problemas:

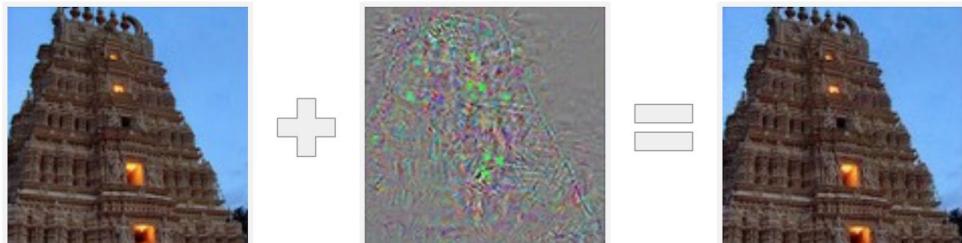
Dataset sesgados

Machine Learning can amplify bias.



- Data set: 67% of people cooking are women
- Algorithm predicts: 84% of people cooking are women

Adversarial attacks



Original image

Temple (97%)

Perturbations

Adversarial example

Ostrich (98%)

(a) Image



(b) Prediction



(c) Adversarial Example



(d) Prediction



Falta capacidad de explicación de los resultados

Data
Data
Data



Black box:
Statistical
procedures or
data science
algorithms



Results

[Segunda parte](#)