Arquitecturas de Deep Learning

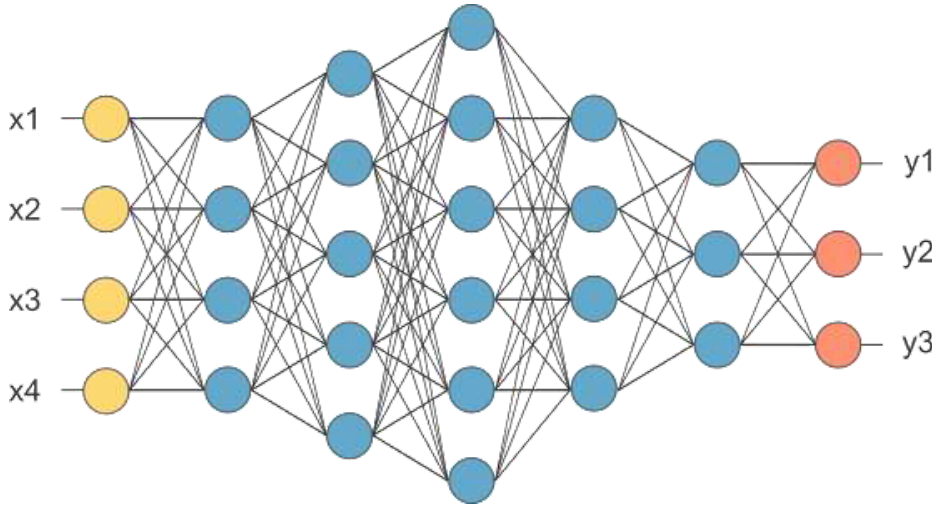
Roberto Muñoz, PhD
Astrónomo y Data Scientist
MetricArts

github.com/rpmunoz

@RobertoKPax

Temario

- Arquitecturas más populares
  - Red neuronal prealimentada o feed-forward
  - Redes neuronales convolucionales
  - Redes neuronales recurrentes
- Regularización

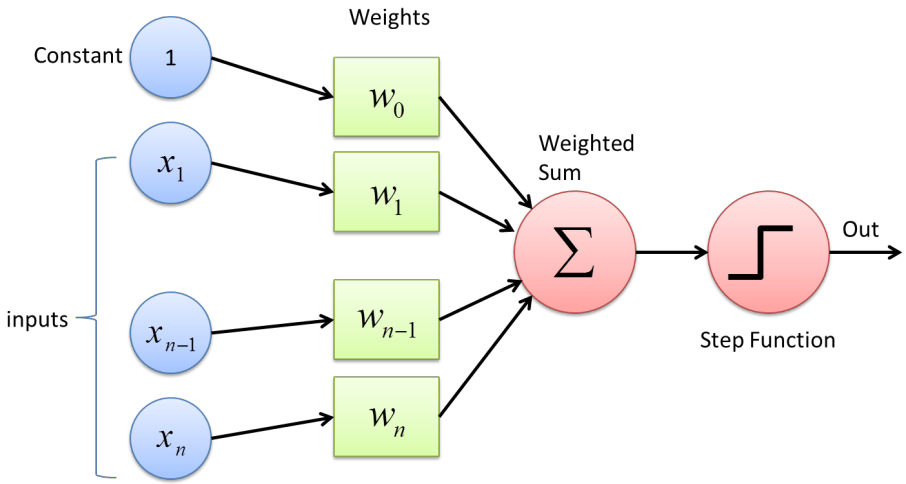


Feed-forward network

REDES NEURONALES PREALIMENTADAS

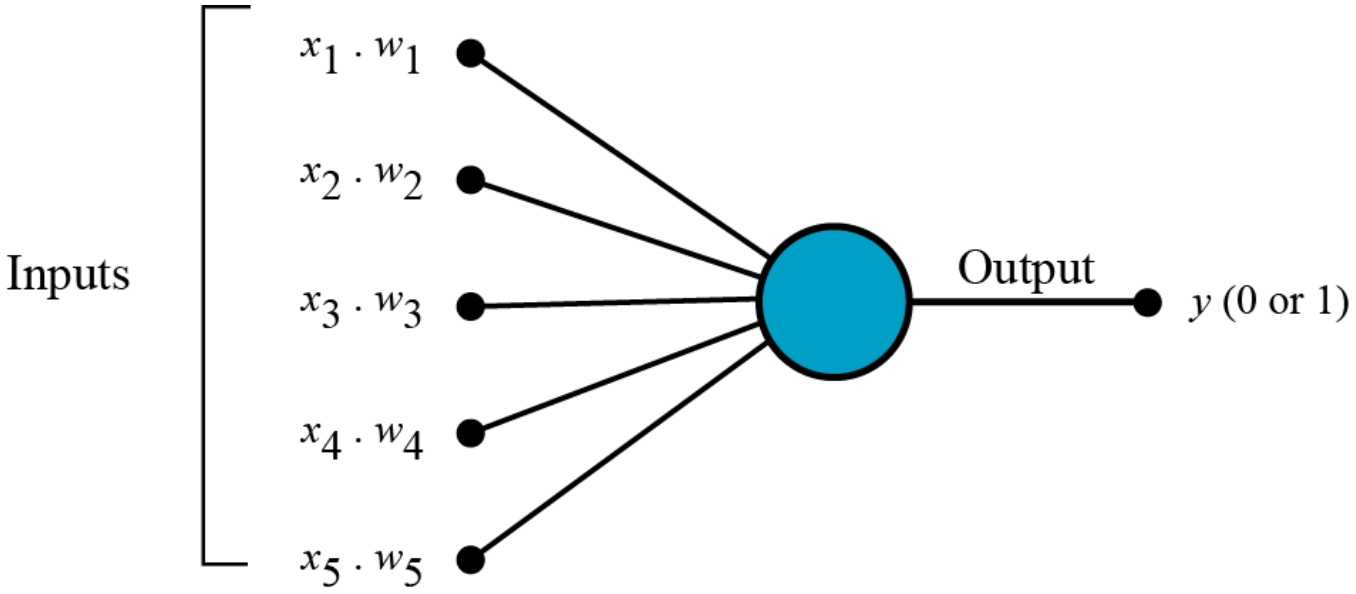
Perceptrón

- Perceptrón lo introduce Frank Rosenblatt (1958)
- Principales componentes
  - Valores de entrada o capa de input
  - Pesos (weights) y sesgo (bias)
  - Suma total
  - Función de activación



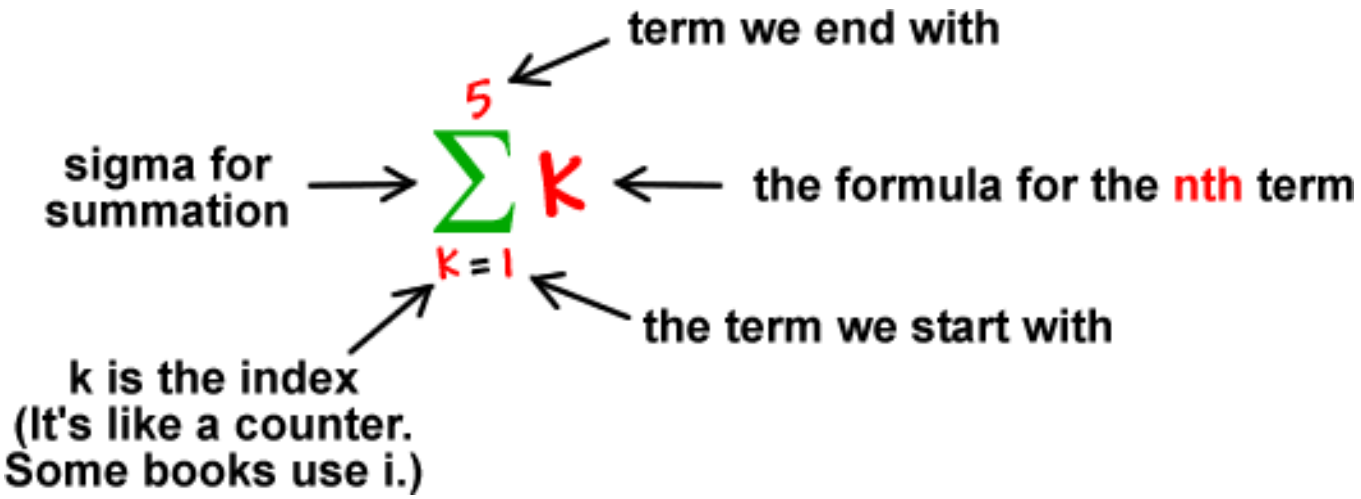
Paso 1

- Cada uno de los valores de entrada **x** es multiplicado por un valor de peso **w**.
- El resultado se llama **k = x × w**.



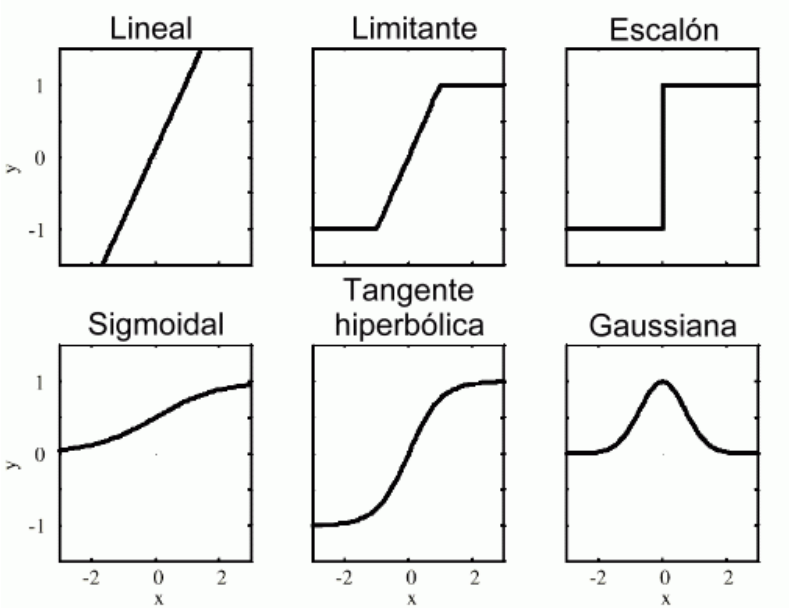
Paso 2

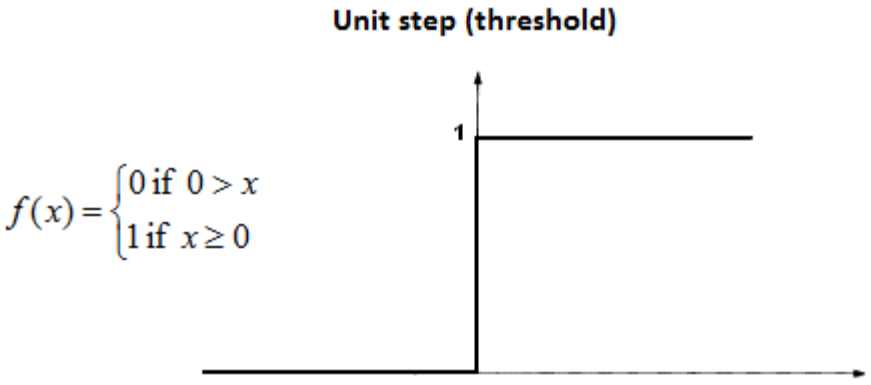
- Sumar todos los valores de **k**. Esa suma se llama suma pesada.



Paso 3

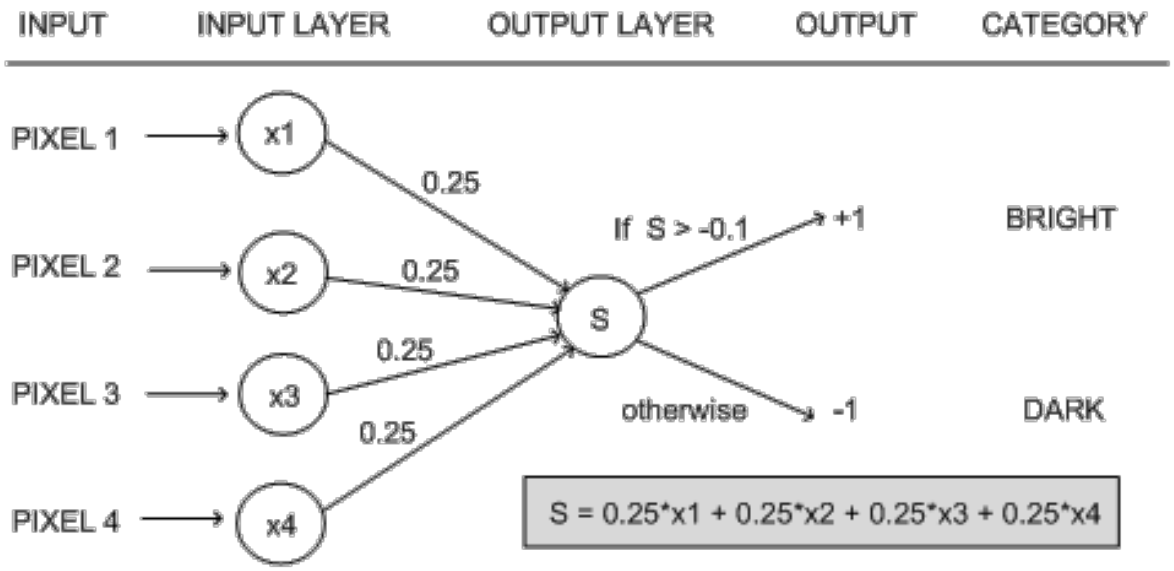
- Usar el valor de la suma pesada como valor de entrada para la función de activación.
- Funciones de activación: lineal, limitante, escalón, sigmoidal, tangente hiperbólica, Gaussiana, etc.





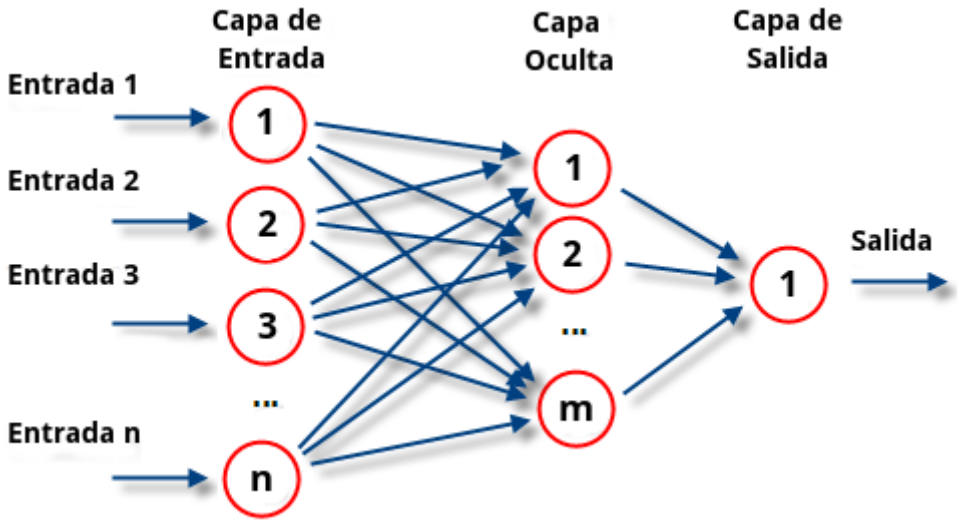
¿Porqué usar pesos y sesgo?

- Los pesos indican cuán importante es un nodo.
- Los valores de sesgo permiten desplazar el threshold de la función de activación ya sea hacia valores bajos o altos.



Feed-forward network

- Las Redes neuronales prealimentadas fueron las primeras que se desarrollaron y son el modelo más sencillo.
- En estas redes la información se mueve en una sola dirección: hacia adelante.
- Los principales exponentes de este tipo de arquitectura son el perceptrón y el perceptrón multicapa.

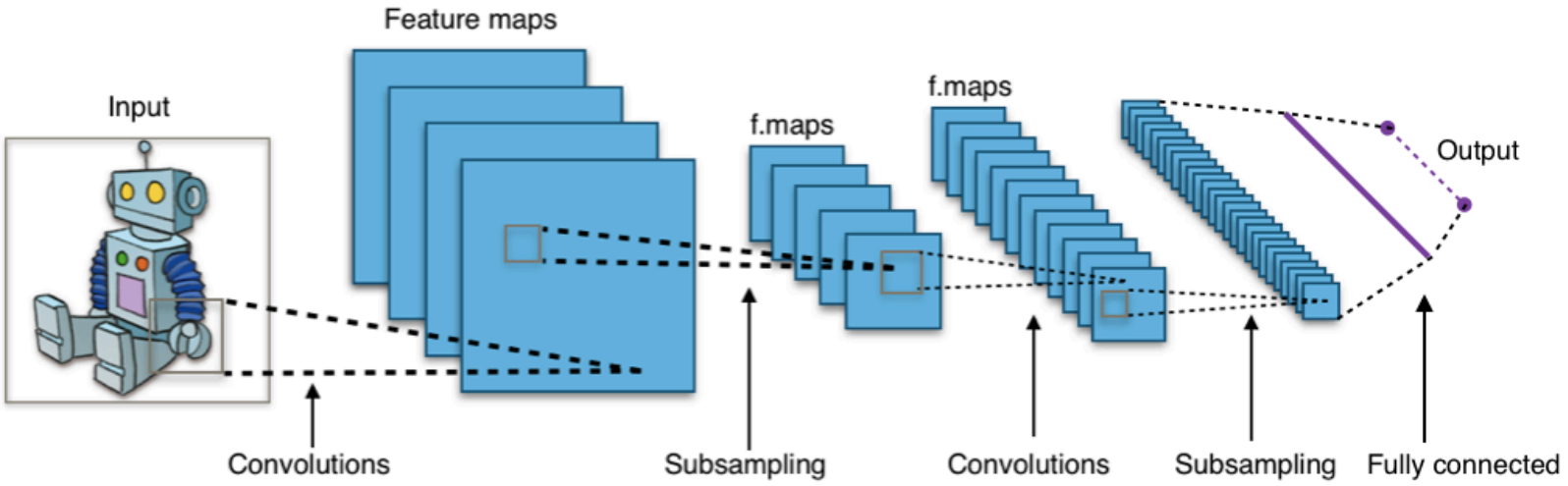


CNN

REDES CONVOLUCIONALES

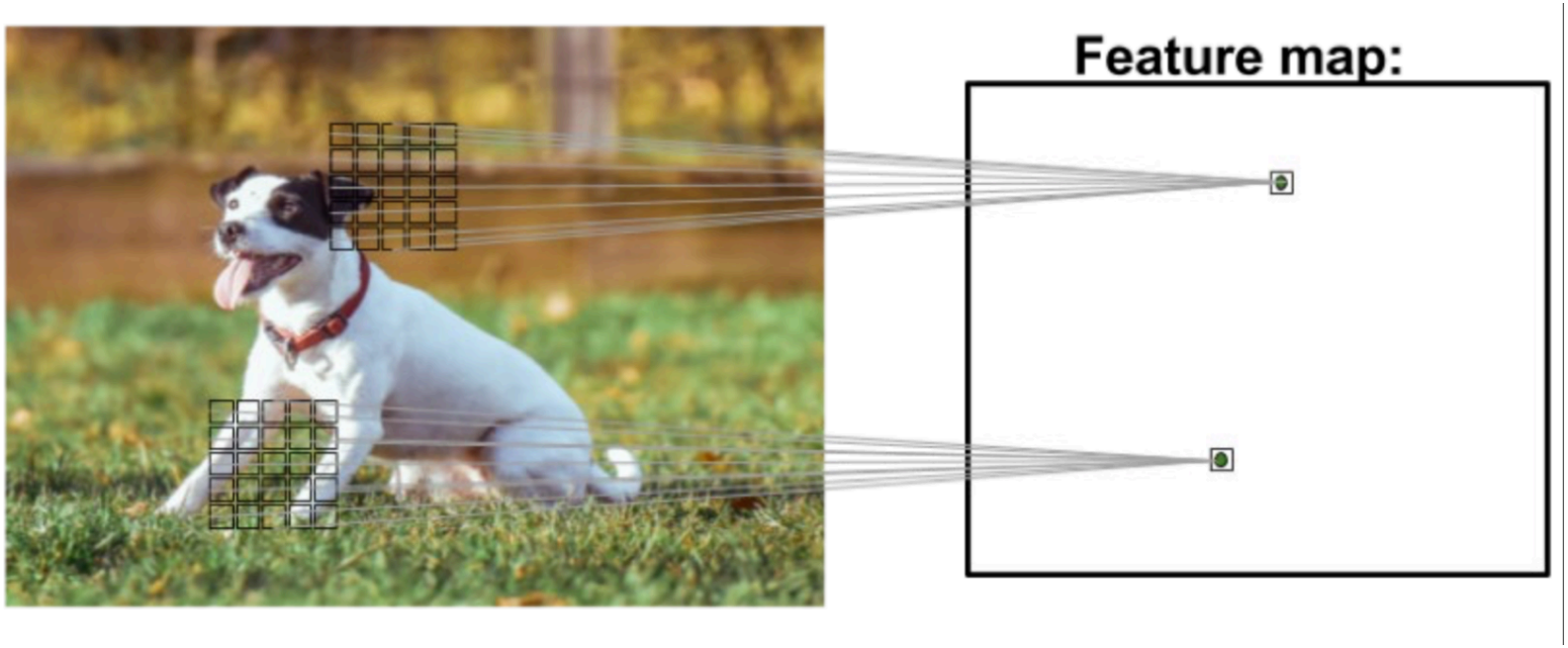
Convolutional neural network

- Las redes neuronales convolucionales son similares a las redes neuronales ordinarias.
- Se componen de neuronas que tienen pesos y sesgos que pueden aprender.
- Cada neurona recibe algunas entradas, realiza un producto escalar y luego aplica una función de activación.



Convolutional neural network

- CNN calcula mapas de features a partir de una imagen de entrada. Cada elemento proviene de un región local de píxels.
- La región local de píxels se denomina local receptive field o campos receptivos.



Convolutional neural network

- ¿Porqué funcionan tan bien en imágenes?
  - Conectividad local: Cada elemento del mapa de features está conectado con una pequeña región de píxels/patrones. Función de respuesta alta de los filtros aprendidos. Se parte con representaciones de pequeñas regiones y luego se escala a regiones de mayor tamaño.
  - Pesos compartidos: Cada filtro se replica a lo largo del campo de visión completo. Invariancia respecto a translación.
  - Volúmenes 3D de neuronas: Las capas de una CNN tiene

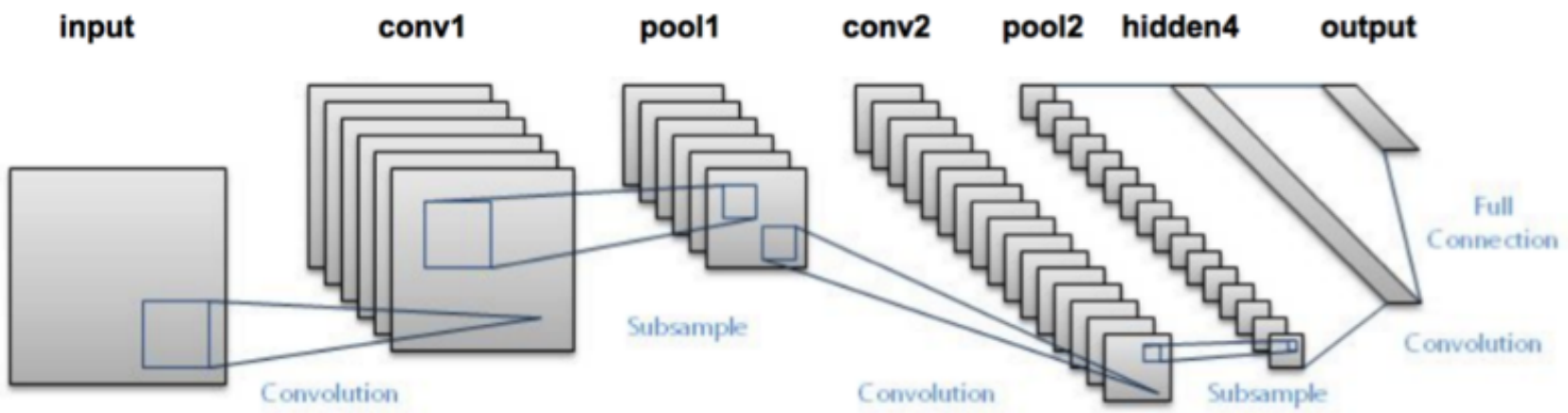
Convolutional neural network

- Las redes neuronales convolucionales van a estar construidas por 3 tipos de capas,
  - Capa convolucional
  - Capa de reducción o pooling. Reducir cantidad de parámetros, características más comunes.
  - Capa clasificadora totalmente conectada. Suelen ser las últimas capas de la red.

- LeNet, AlexNet, VGGNet, GoogleNet, ResNet, etc.

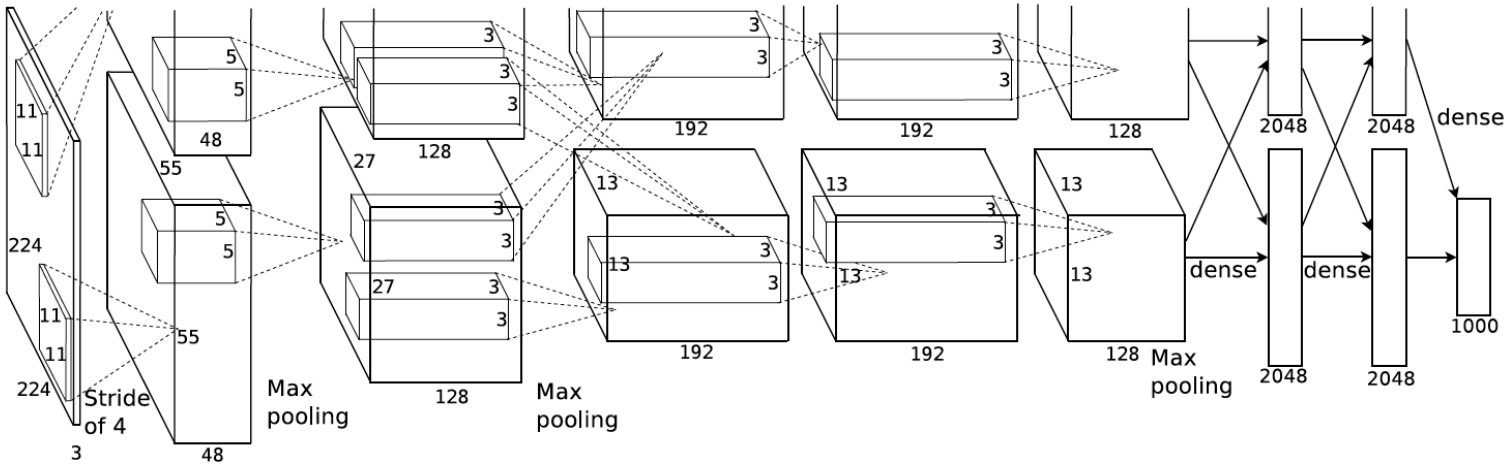
LeNet-5

- Propuesta por Yan LeCun et al. (1998)
- Usada por bancos para reconocer números escritos a mano en cheques digitalizados.



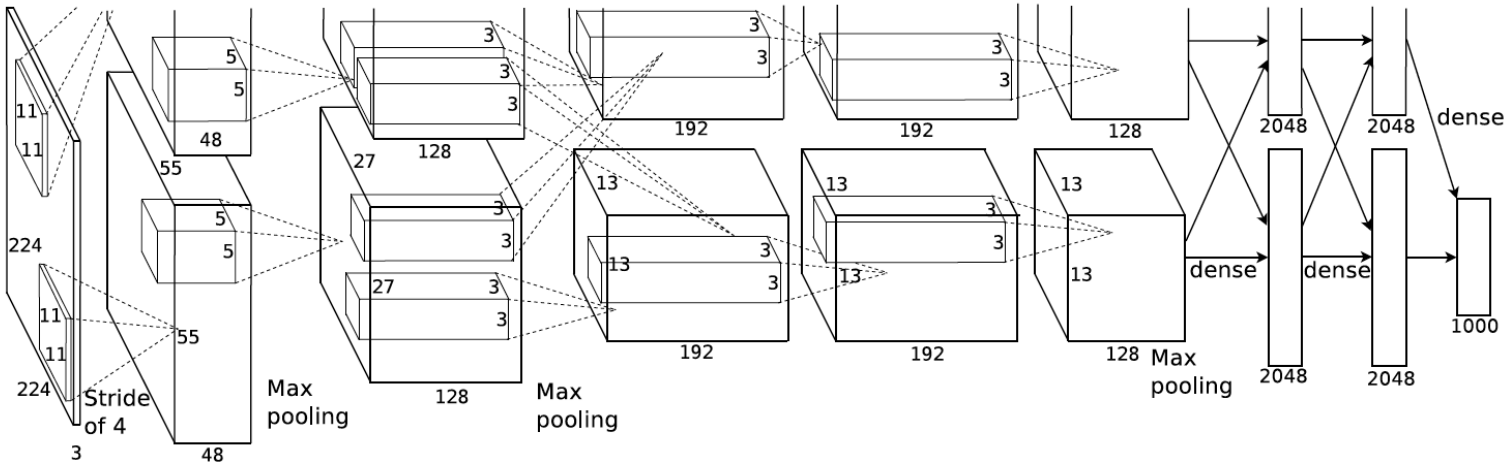
AlexNet

- Propuesta por Krizhevsky et al. (2012) como solución al challenge de ImageNet 2012.
- Clasificación de pequeñas imágenes.
- Top-5 score: Etiqueta anotada es una de las 5 predicciones con mayor probabilidad.
- Redujo error top-5 de 26,2% a 15,3%.

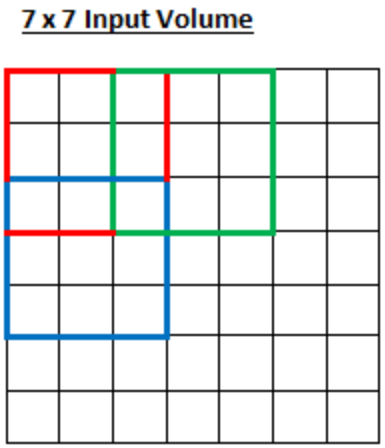


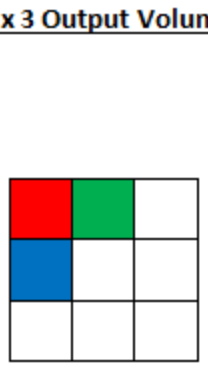
AlexNet

- Contiene 5 capas convolucionales y 3 capas densas (fully-connected).
- Se aplica ReLU después de cada capa convolucional y capa densa.
- Se aplica dropout antes de la primera y segunda capa densa.

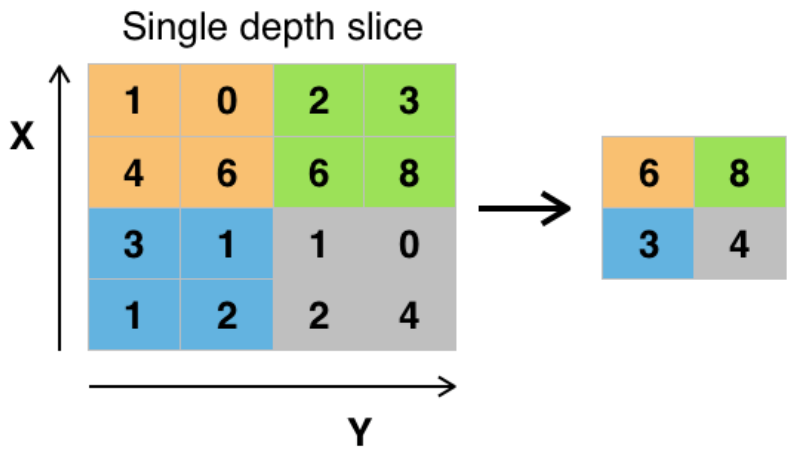


Stride, padding, pooling





0	0	0	0	0	0
0	35	19	25	6	0
0	13	22	16	53	0
0	4	3	7	10	0
0	9	8	1	3	0
0	0	0	0	0	0



Example of Maxpool with a 2x2 filter and a stride of 2

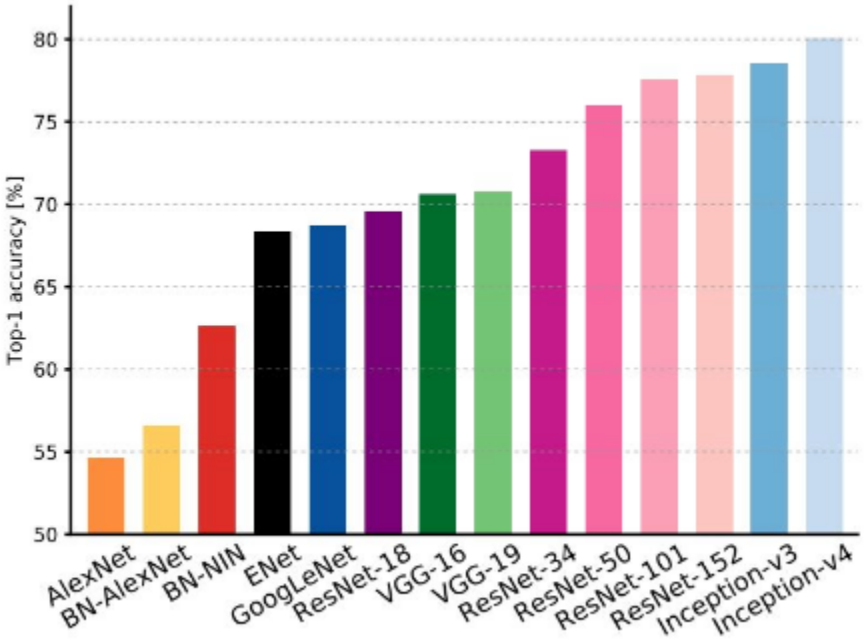
AlexNet – Parámetros

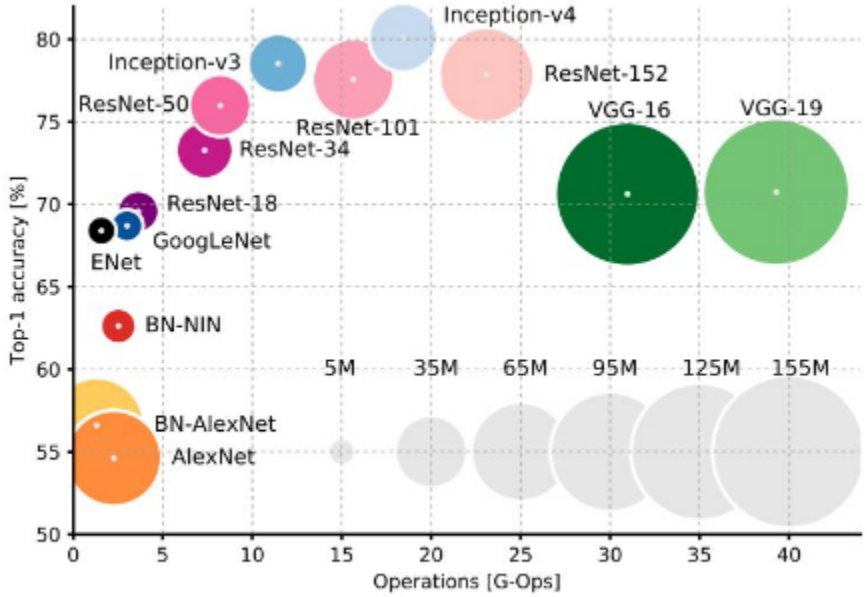
Size / Operation	Filter	Depth	Stride	Padding	Number of Parameters	Forward Computation
3 * 227 * 227						
Conv1 + Relu	11 * 11	96	4		(11∗11∗3 + 1) ∗ 96=34944	(11∗11∗3 + 1) ∗ 96 ∗ 55 ∗ 55=105705600
96 ∗ 55 ∗ 55						
Max Pooling	3 ∗ 3		2			
96 * 27 * 27						
Norm						
Conv2 + Relu	5 ∗ 5	256	1	2	(5 ∗ 5 ∗ 96 + 1) ∗ 256=614656	(5 ∗ 5 ∗ 96 + 1) ∗ 256 ∗ 27 ∗ 27=448084224
256 * 27 * 27						
Max Pooling	3 ∗ 3		2			
256 * 13 * 13						
Norm						
Conv3 + Relu	3 ∗ 3	384	1	1	(3 ∗ 3 ∗ 256 + 1) ∗ 384=885120	(3 ∗ 3 ∗ 256 + 1) ∗ 384 ∗ 13 ∗ 13=149585280
384 ∗ 13 ∗ 13						
Conv4 + Relu	3 ∗ 3	384	1	1	(3 ∗ 3 ∗ 384 + 1) ∗ 384=1327488	(3 ∗ 3 ∗ 384 + 1) ∗ 384 ∗ 13 ∗ 13=224345472
384 ∗ 13 ∗ 13						
Conv5 + Relu	3 ∗ 3	256	1	1	(3 ∗ 3 ∗ 384 + 1) ∗ 256=884992	(3 ∗ 3 ∗ 384 + 1) ∗ 256 ∗ 13 ∗ 13=149563648
256 * 13 * 13						
Max Pooling	3 ∗ 3		2			
256 ∗ 6 ∗ 6						
Dropout (rate 0.5)						
FC6 + Relu					256 ∗ 6 ∗ 6 ∗ 4096=37748736	256 ∗ 6 ∗ 6 ∗ 4096=37748736
4096						
Dropout (rate 0.5)						
FC7 + Relu					4096 ∗ 4096=16777216	4096 ∗ 4096=16777216
4096						
FC8 + Relu					4096 * 1000=4096000	4096 * 1000=4096000
1000 classes						
Overall					62369152=62.3 million	1135906176=1.1 billion
Conv VS FC					Conv:3.7million (6%) , FC: 58.6 million (94%)	Conv: 1.08 billion (95%) , FC: 58.6 million (5%)

AlexNet

- La red tiene 62,3 millones de parámetros
- Necesita de 1.100 millones de unidades cómputo en un solo forward pass.
- Las capas convolucionales representan el 6% de todos los parámetros. Consumen cerca del 95% del cómputo.
- Capas convolucionales tienen pocos parámetros y requieren un elevado número de cálculos.
- Capas densas o fully-connected tienen muchos parámetros y requiere pocos cálculos.

CNN architectures

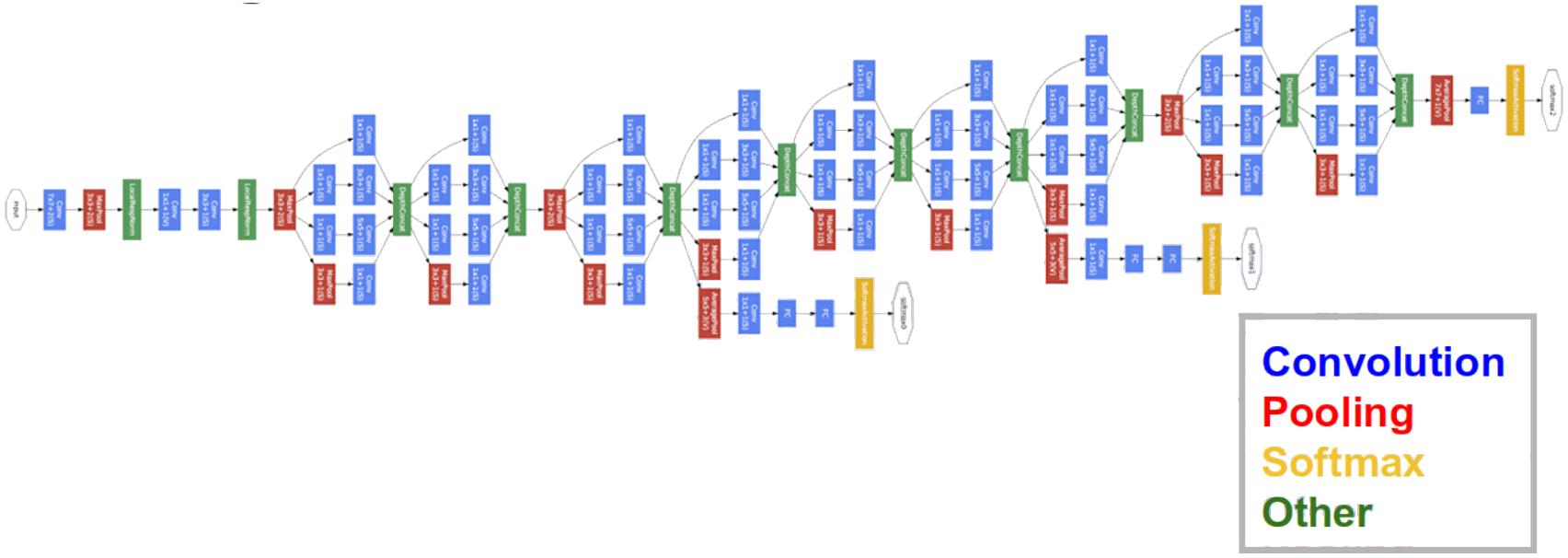




An Analysis of Deep Neural Network Models for Practical Applications, 2017.

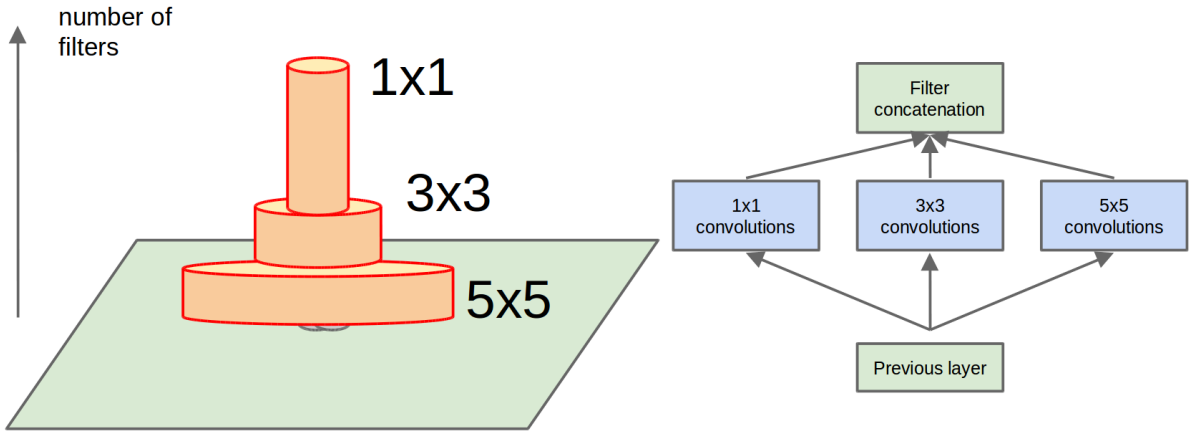
GoogleNet/Inception

- Fue propuesta por el equipo de Google el año 2014
- Alcanzó un error top-5 del 6,6%
- Consiste de 22 capas de CNN
- Se emplea un módulo llamado inception, el cual consiste en usar varias redes convolucionales pequeñas para así reducir el número de parámetros.

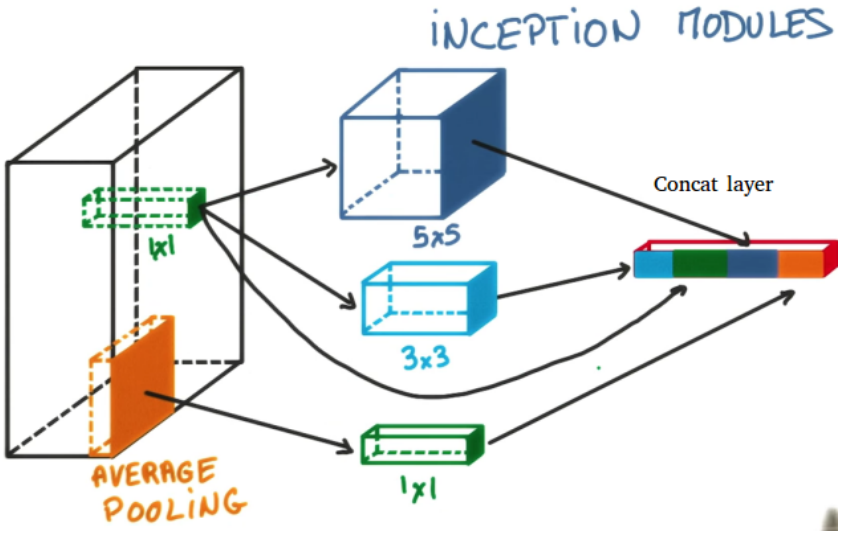


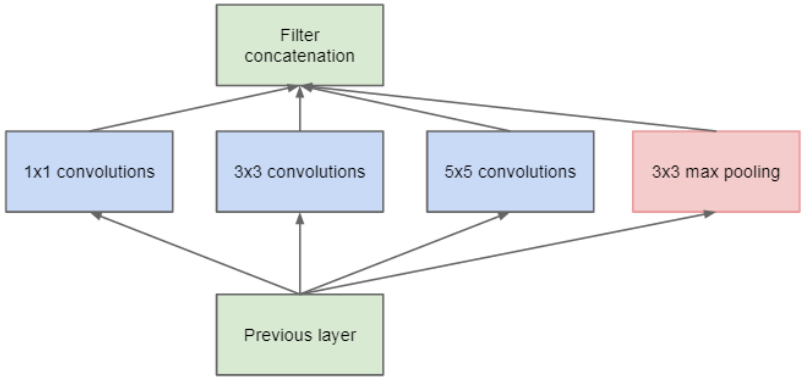
Módulo Inception

- La idea es cubrir una gran área de la imagen, pero seguir manteniendo una resolución mas fina para extraer información de regiones pequeñas.
- Un mayor número de parámetros aumenta la probabilidad de hacer sobre ajuste.

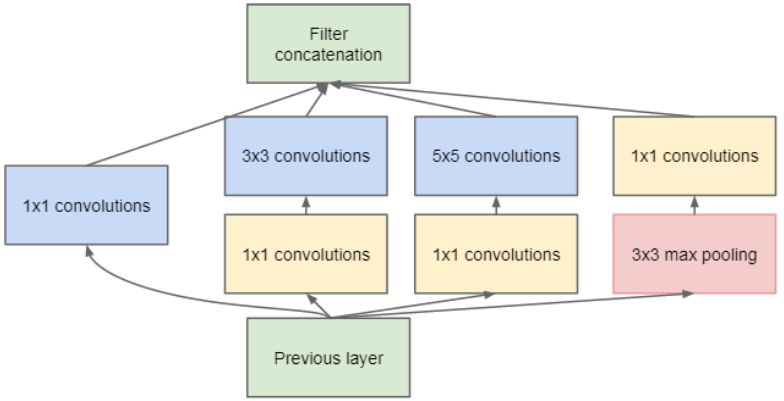


Módulo Inception





(a) Inception module, naïve version



(b) Inception module with dimension reductions

Capa bottleneck

- Digamos que tenemos 256 features de entrada y 256 de salida. Configuramos un módulo de inception que aplique convoluciones 3x3.

  Número de cálculos: 256x256x3x3 = 589.000 MAC
  MAC=multiply-accumulate

- Otra alternativa es configurar un módulo inception que reduzca la dimensionalidad de los features, digamos de 256 a 64.
  - 256x64x1x1 = 16.000 MAC
  - 64x64x3x3 = 36.000 MAC
  - 64x256x1x1 = 16.0000 MAC TOTAL=70.000 MAC