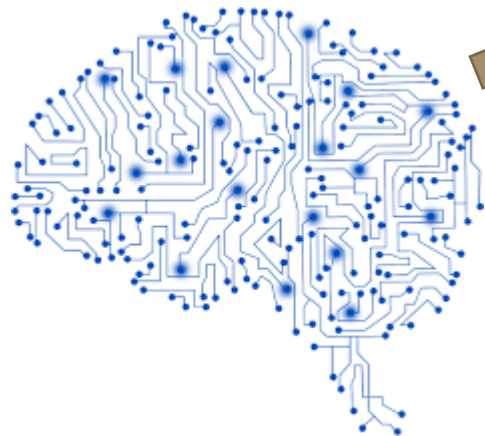




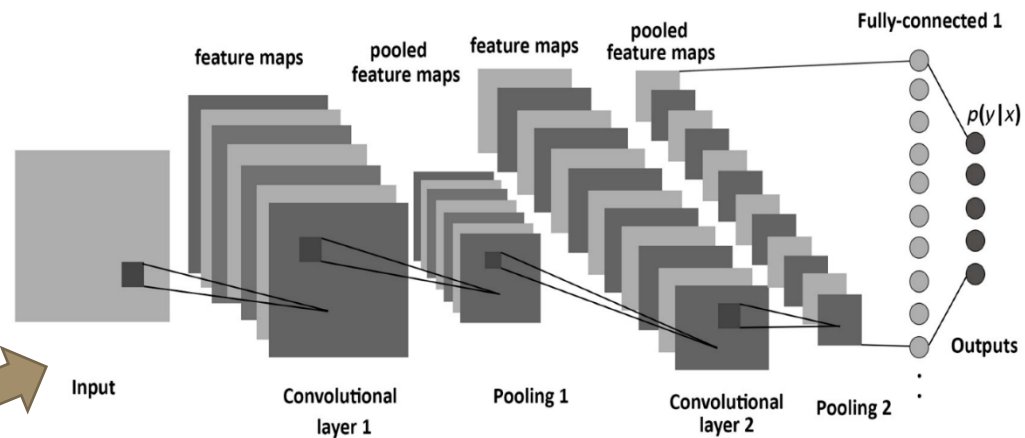
Profesor: Daniel Osorio



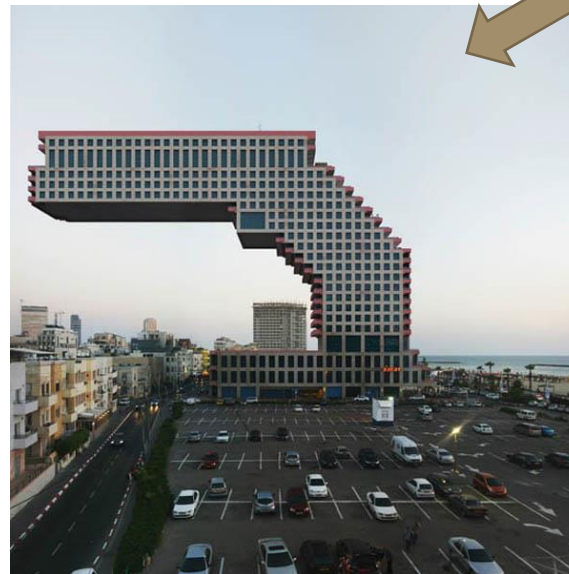
AGENDA



Deep Learning



Redes convolucionales

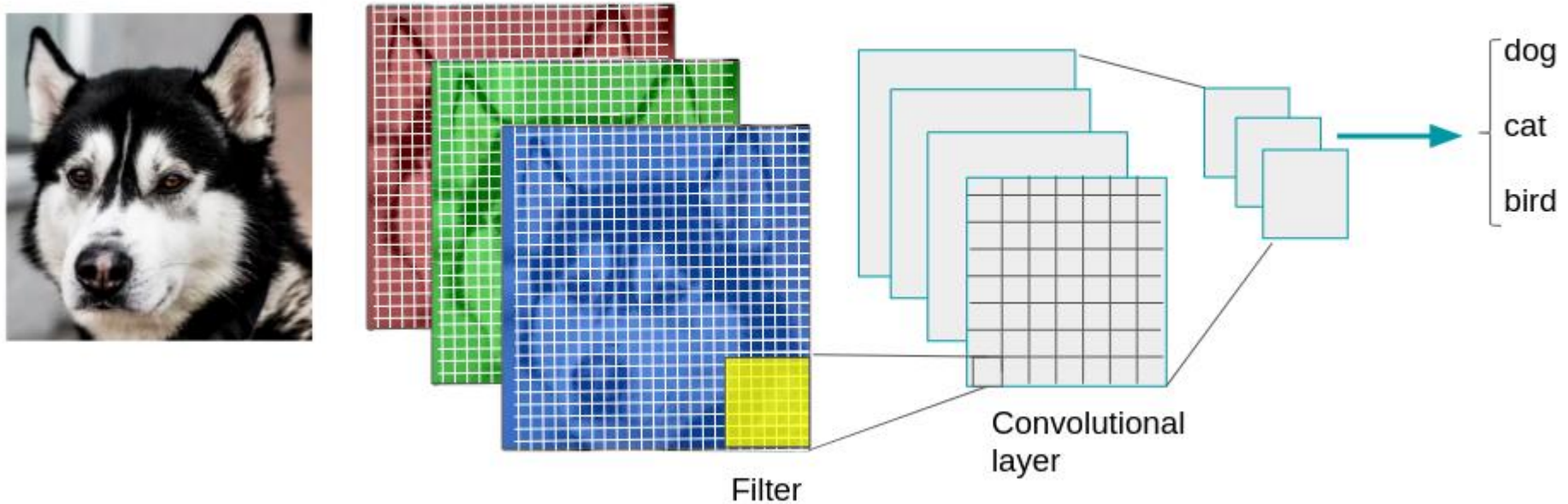


**Arquitecturas
convolucionales**

NED
means
Not Enough Data

**Limitaciones
de datos**





CNN — REDES CONVOLUCIONALES



REDES CONVOLUCIONALES

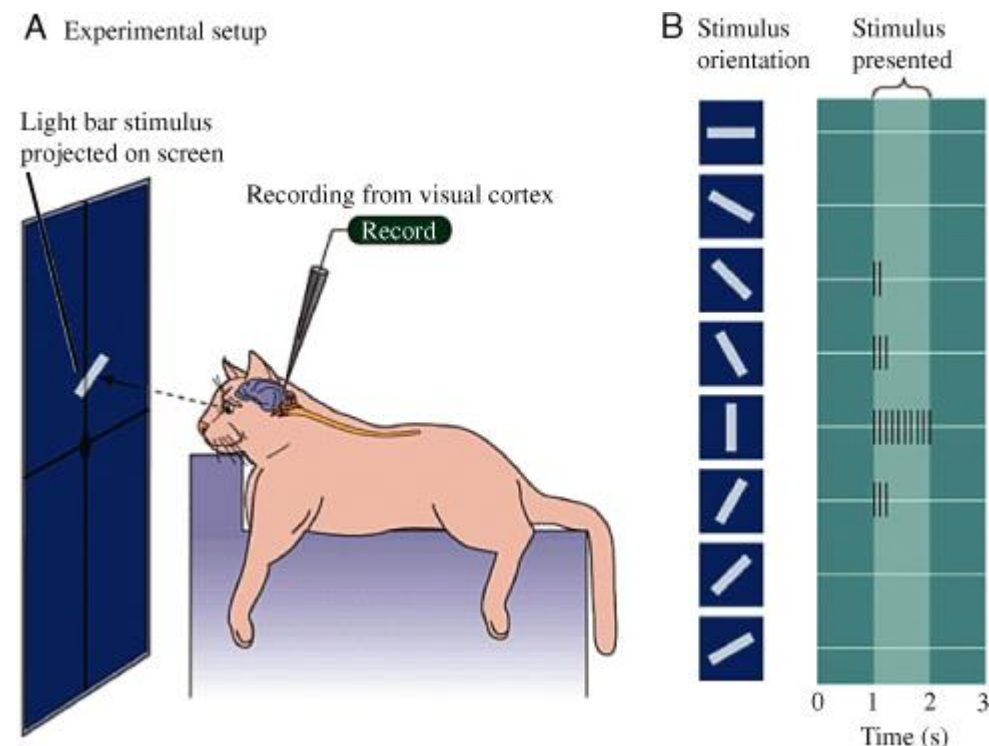
- La arquitectura de las redes neuronales convolucionales (**CNNs** – Convolutional Neural Networks) es muy diferente al de las redes basadas en capas **densas**.
- La arquitectura se basa en la idea de diferentes niveles de **abstracción jerárquicos** que permiten encontrar nuevas representaciones (**features**) de los datos de entrada orientados a responder al objetivo de entrenamiento del modelo
- Aunque su desarrollo surgió en el campo de **visión por computador**, tienen múltiples campos de aplicación
- Se inspiran de diferentes campos como la **Biología y neuro ciencia**, el **tratamiento de imágenes** y la **teoría del aprendizaje y el machine learning**.



REDES CONVOLUCIONALES

Motivaciones, inspiración:

- **Biología, neuro ciencia:** el proceso de la visión en el cerebro comienza por la aplicación de los mismos filtros básicos a todo el campo visual (Hubel & Wiesel, premio Nobel 1960)



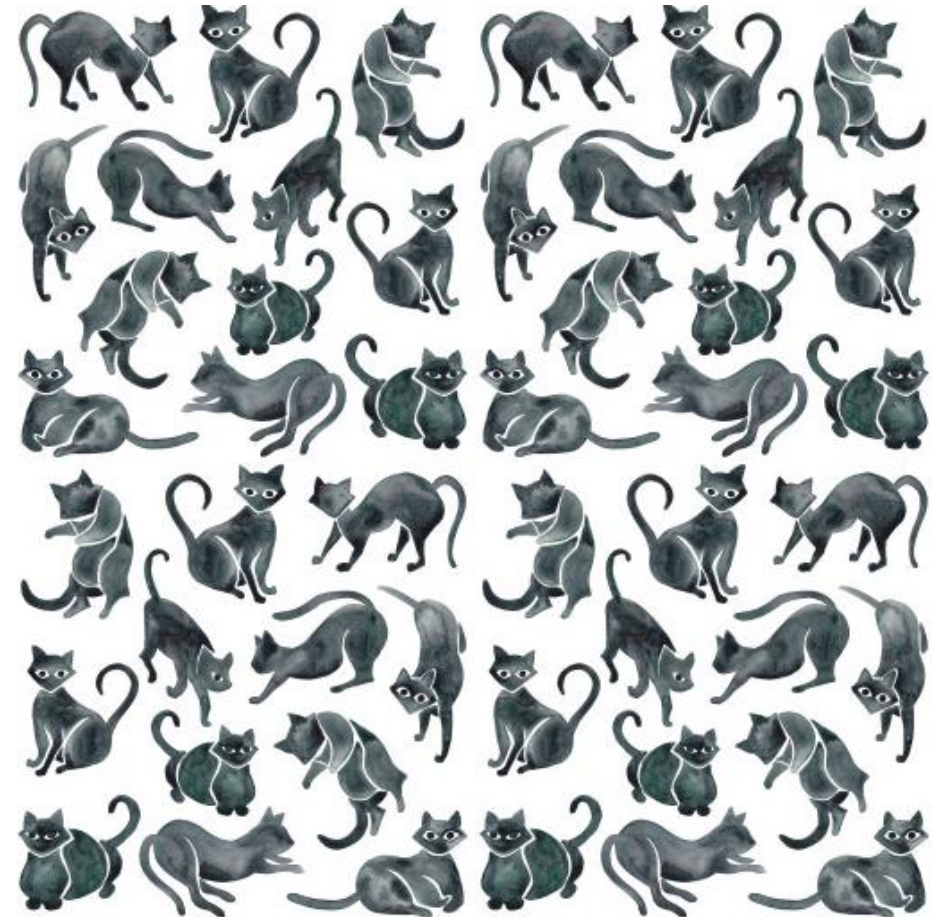
<http://www.informit.com/articles/article.aspx?p=1431818>



REDES CONVOLUCIONALES

Motivaciones, inspiración:

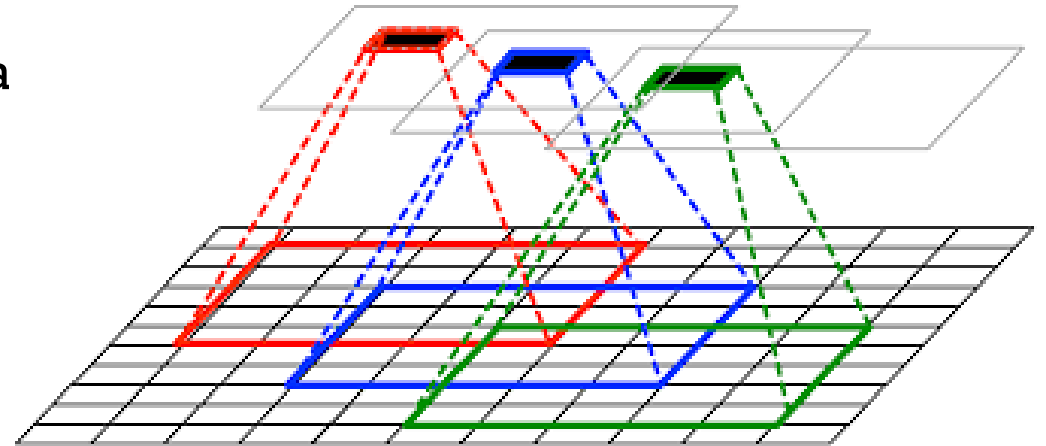
- **Tratamiento de imágenes:** el reconocimiento de un objeto en una imagen debe ser independiente de:
 - la posición (traslación) y ángulo (rotación)
 - diferencias de tamaño
 - diferencias de iluminación
 - leves deformaciones o distorsiones
- Propiedad de **invarianza**
- Repetir operaciones de detección locales en diferentes partes de la imagen



REDES CONVOLUCIONALES

Motivaciones, inspiración:

- **Teoría del aprendizaje, machine learning:**
una capa densa otorga demasiada importancia a inputs particulares de la capa anterior, sometiendo los resultados a una dependencia de los mismos.
- Se debería buscar la generalización de los resultados con pequeñas capas densas aplicadas a subconjuntos de los inputs en busca de una **regularización** del modelo

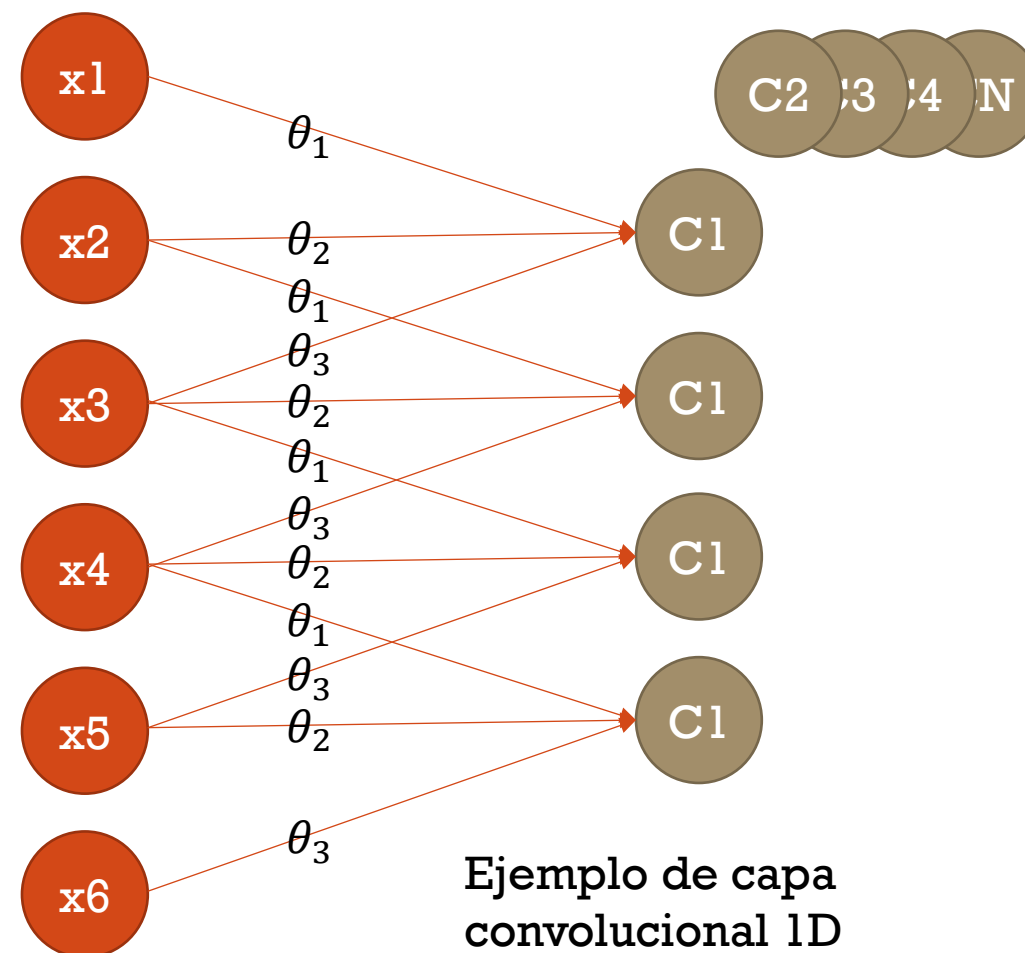


<https://arxiv.org/pdf/1512.07108.pdf>

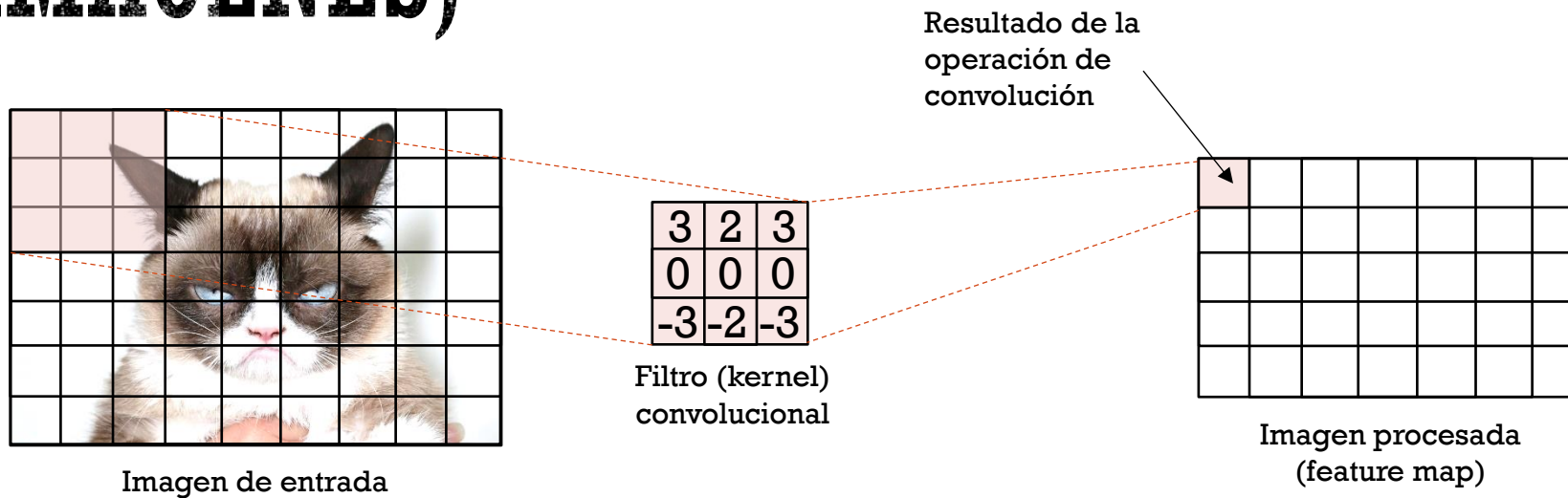


REDES CONVOLUCIONALES — 1D

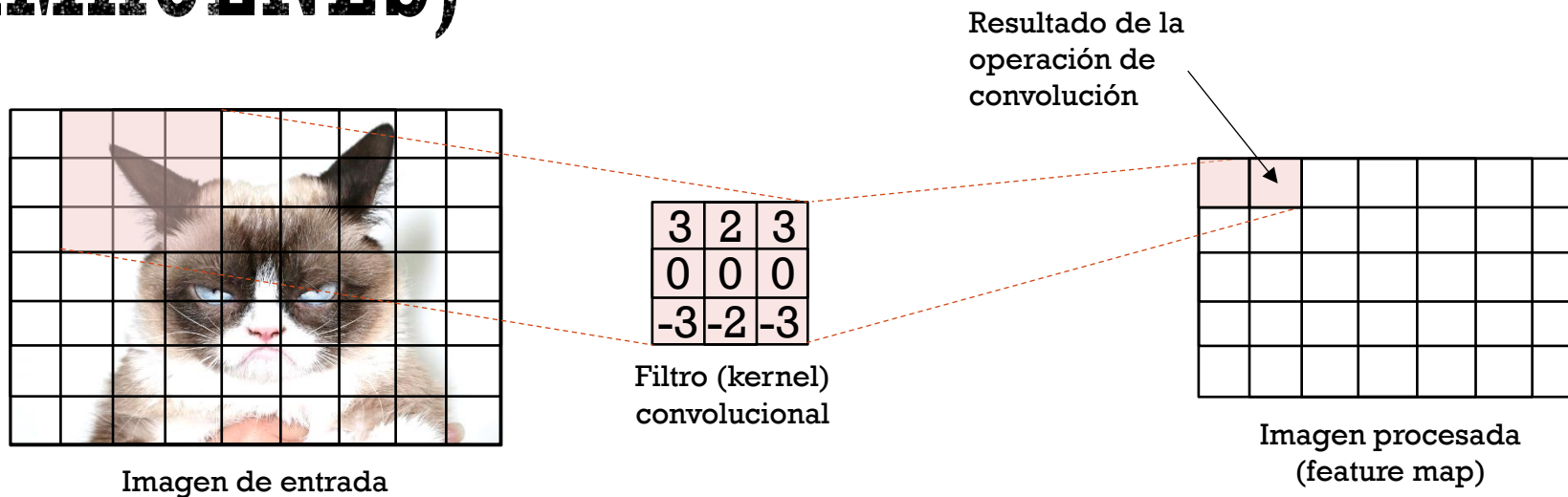
- Una **capa convolucional** consiste en **filtros** que realizan barridos en subregiones de la capa anterior (\rightarrow ventanas 1D, 2D, 3D, etc.), que se deslizan sobre las neuronas de la capa anterior). El entrenamiento consiste en determinar los pesos de los filtros mas adecuados para la tarea de aprendizaje.



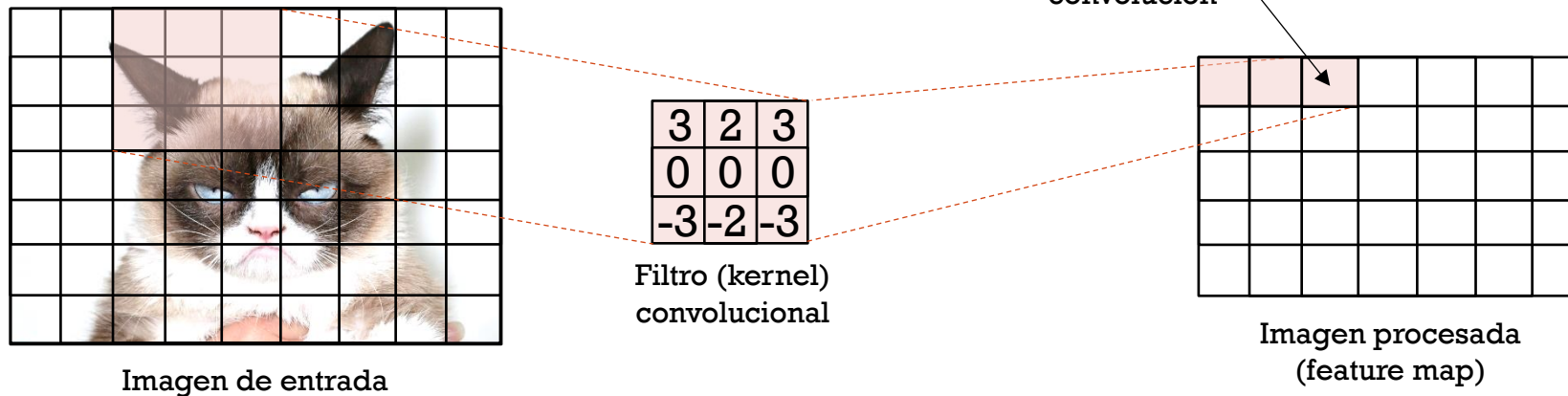
REDES CONVOLUCIONALES — 2D (IMÁGENES)



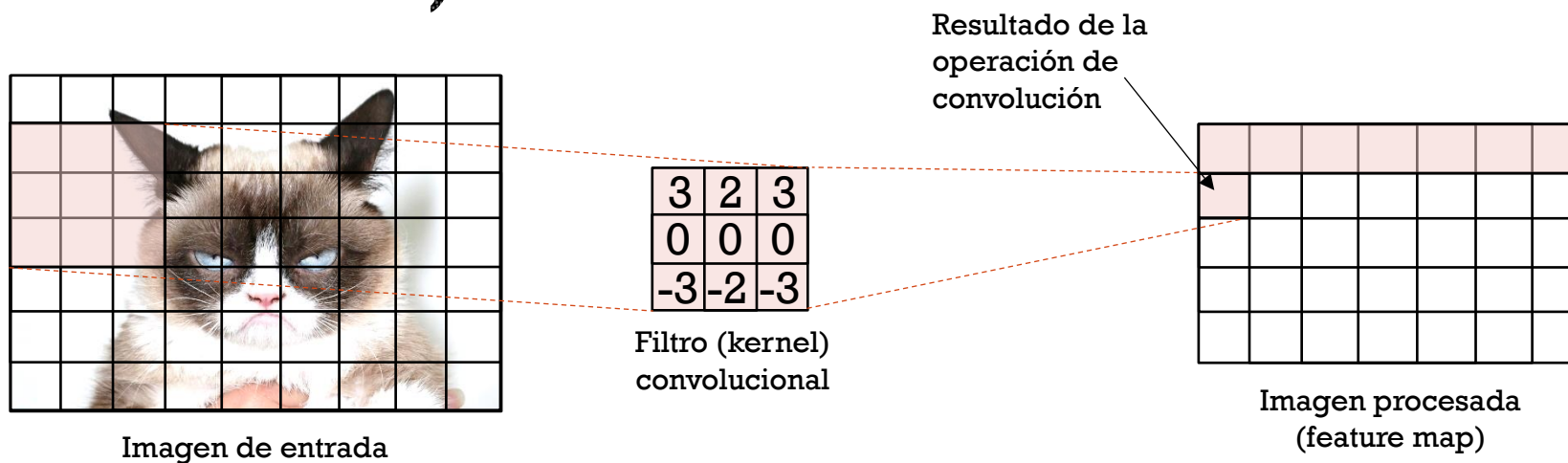
REDES CONVOLUCIONALES — 2D (IMÁGENES)



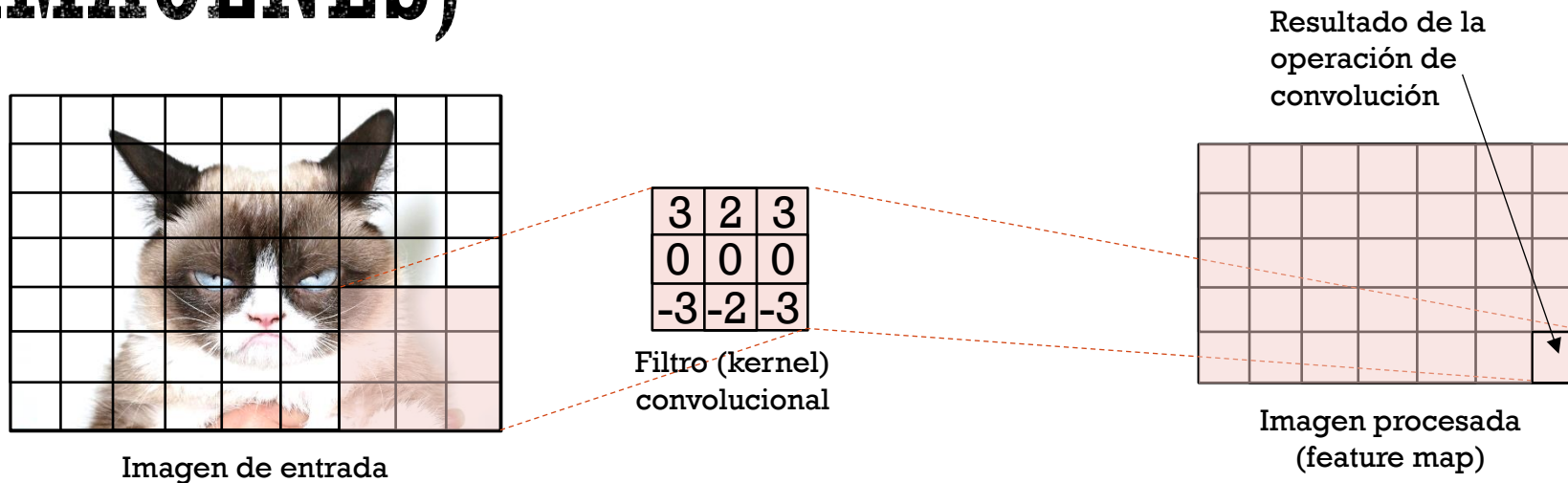
REDES CONVOLUCIONALES — 2D (IMÁGENES)



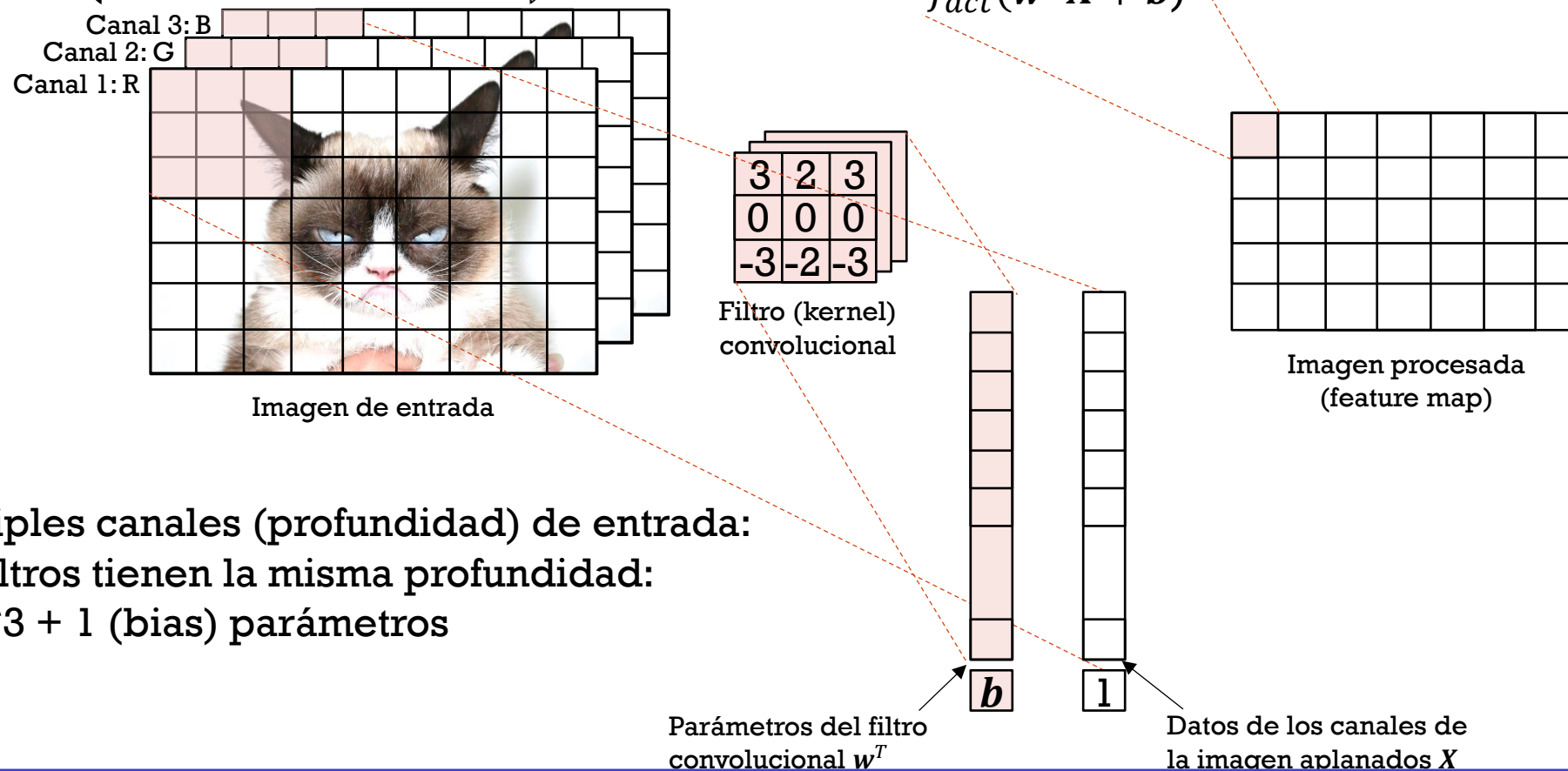
REDES CONVOLUCIONALES — 2D (IMÁGENES)



REDES CONVOLUCIONALES — 2D (IMÁGENES)



REDES CONVOLUCIONALES — 2D (IMÁGENES)



Múltiples canales (profundidad) de entrada:
los filtros tienen la misma profundidad:
 $K \times K \times 3 + 1$ (bias) parámetros

REDES CONVOLUCIONALES — 2D (IMÁGENES)

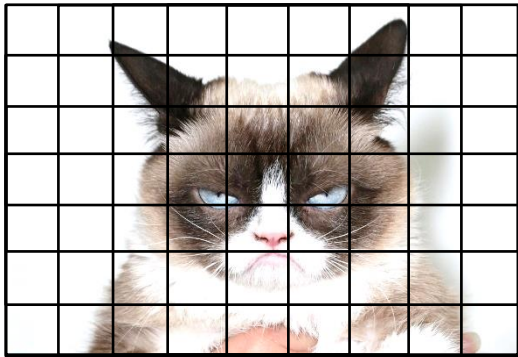
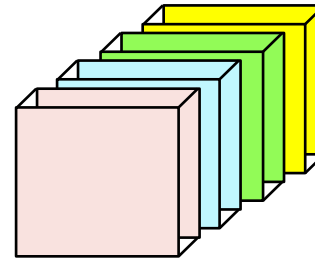


Imagen de entrada



Filtros (kernels)
convolucionales

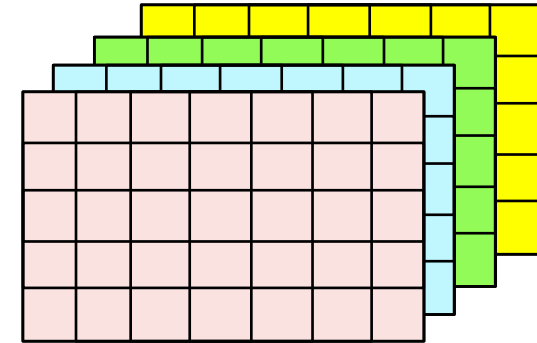


Imagen procesada
(feature map)

Múltiples filtros convolucionales en la misma capa



TALLER CONVOLUCIONES EN EXCEL

Desarrollar las 2 operaciones de convolución, encontrando los feature maps correspondientes.



REDES CONVOLUCIONALES

stride=2

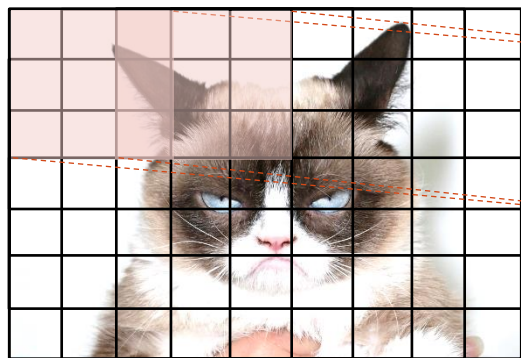


Imagen de entrada

Filtro (kernel)
convolucional

3	2	3
0	0	0
-3	-2	-3

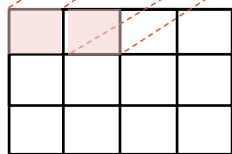


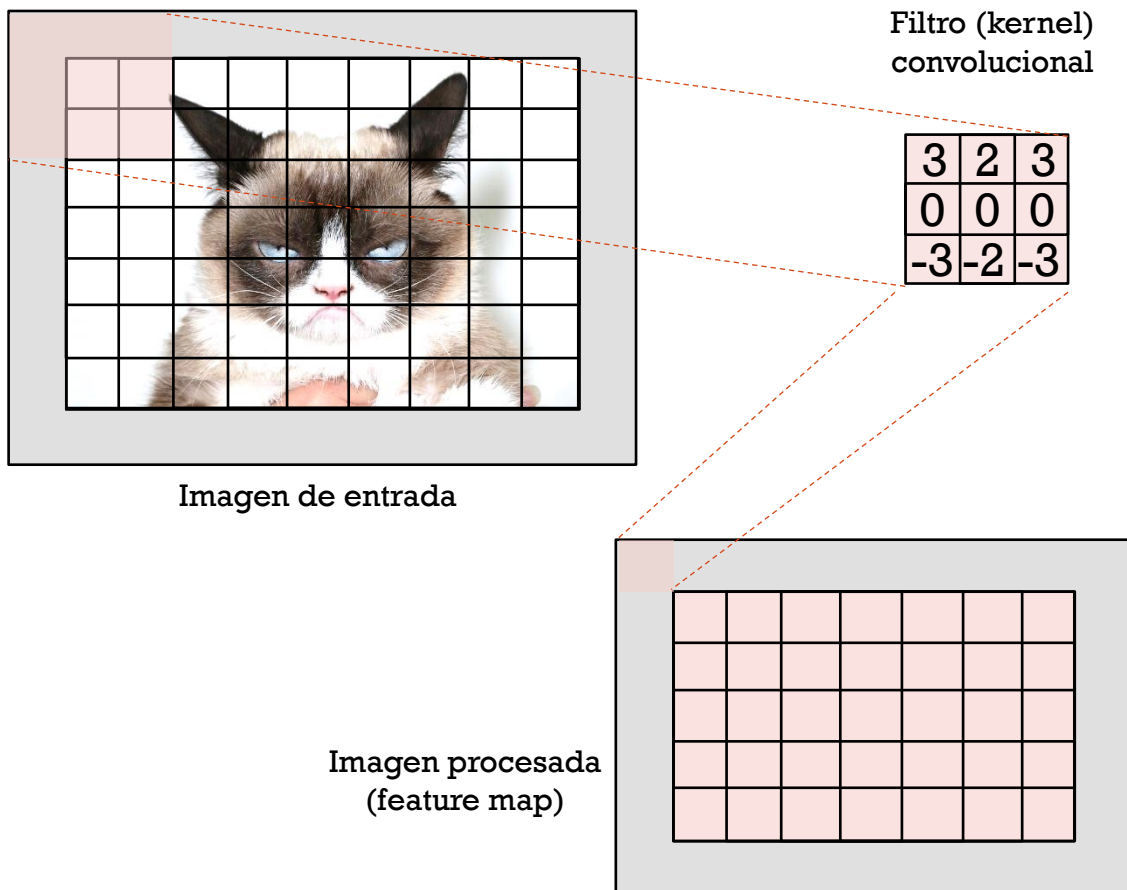
Imagen procesada
(feature map)

- **Stride:** Controla el barrido de los kernels convolucionales al atravesar los inputs.
 - Entre más grande el stride: más pequeño el feature map resultante
 - A un stride de 1 se le llama “FULL” stride
 - Se debe definir el stride para cada una de las dimensiones de los datos de entrada
 - ¿Cuál sería el tamaño de los feature maps con stride 1 para filtros de 3x3 y de 5x5?
- El tamaño del filtro influye en el tamaño del feature map



REDES CONVOLUCIONALES

Padding = SAME, stride = 1



- **Padding:** Para contrarrestar la reducción del tamaño de las entradas después de una convolución se puede “rellenar” los extremos de las mismas con datos ficticios que idealmente no influyan en los resultados
- Cuando no hay padding, se le llama “**VALID**”, los datos de los extremos superiores se ignoran (e.g. últimas columnas y filas en una imagen)
- A un padding que conserva el tamaño de las entradas cuando se tiene un stride de 1 se le llama “**SAME**”. Si el número de dimensiones es impar, se agregan datos en los extremos superiores
- Padding “**FULL**” consiste en $P = K - 1$, pero casi no se utiliza en Deep Learning. Tiene sentido en procesamiento de señales para mitigar efectos borde.



REDES CONVOLUCIONALES

- Si se tiene un stride de 1, y se quiere tener una salida que conserve las dimensiones de entrada, el padding a utilizar es: $ZeroPadding = \frac{K-1}{2}$
- Tanto el tamaño del kernel, como el stride y el padding influyen en el tamaño de los feature maps generados.
- La formula que los une es la siguiente:

$$O = \frac{(W - K + 2P)}{S} + 1 ,$$

donde:

O = Tamaño del feature map

W = Tamaño de los inputs

K = Tamaño del kernel

P = Tamaño del borde del padding

S = Stride



REDES CONVOLUCIONALES

Pooling (2, 2), stride=2

1	6	0	3	4	4
3	4	3	2	5	4
2	3	1	1	4	4
3	1	1	2	3	2
4	5	0	1	1	3
5	3	2	2	2	4

Feature map de entrada

Max pooling

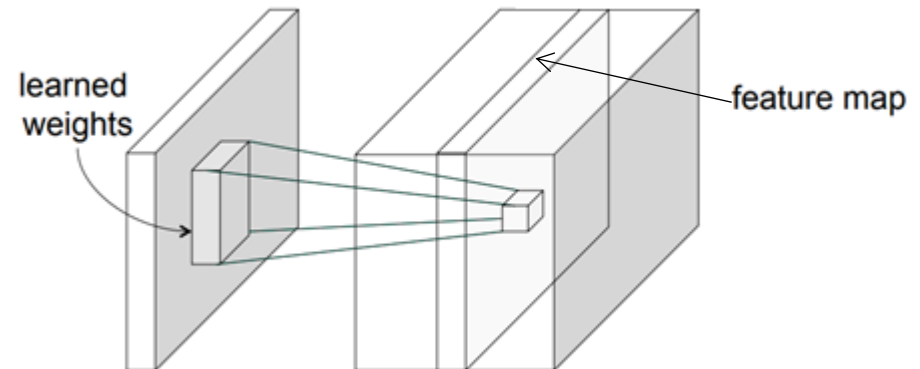
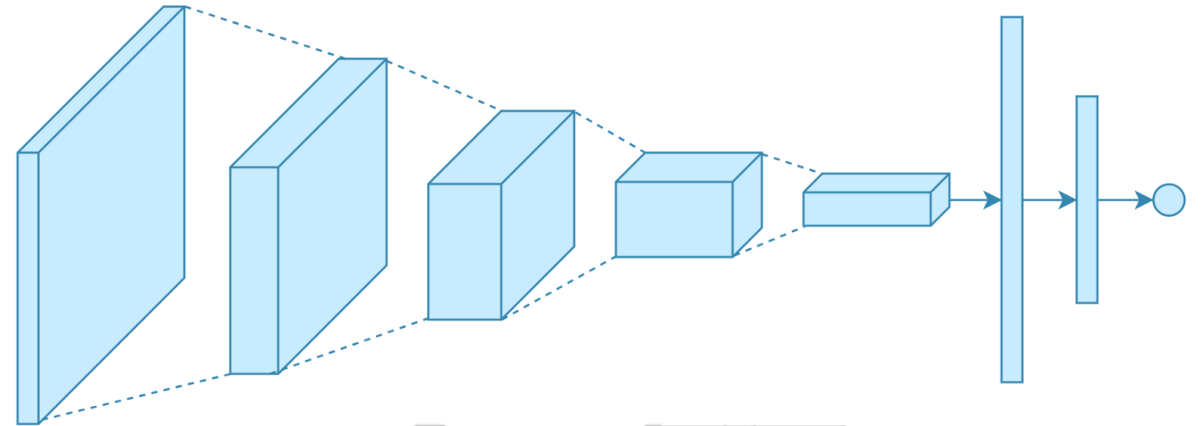
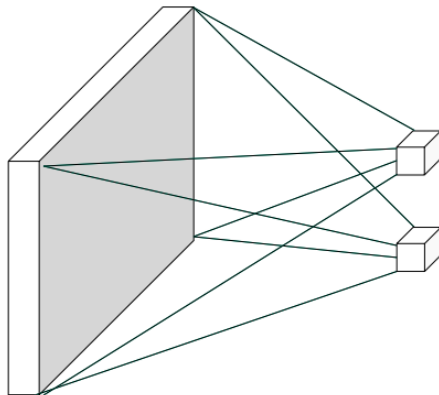
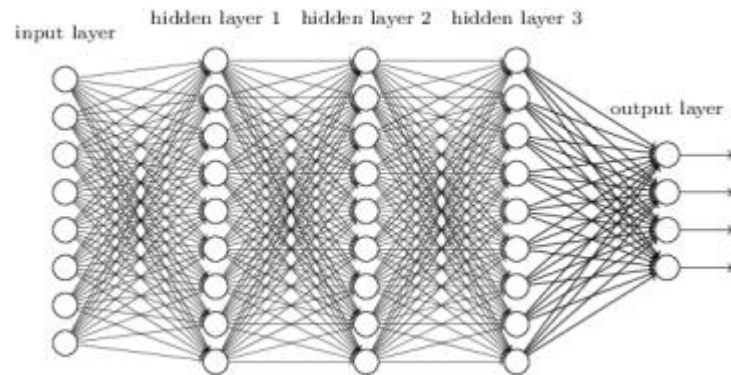
6	3	5
3	2	4
5	2	4

Feature map de salida

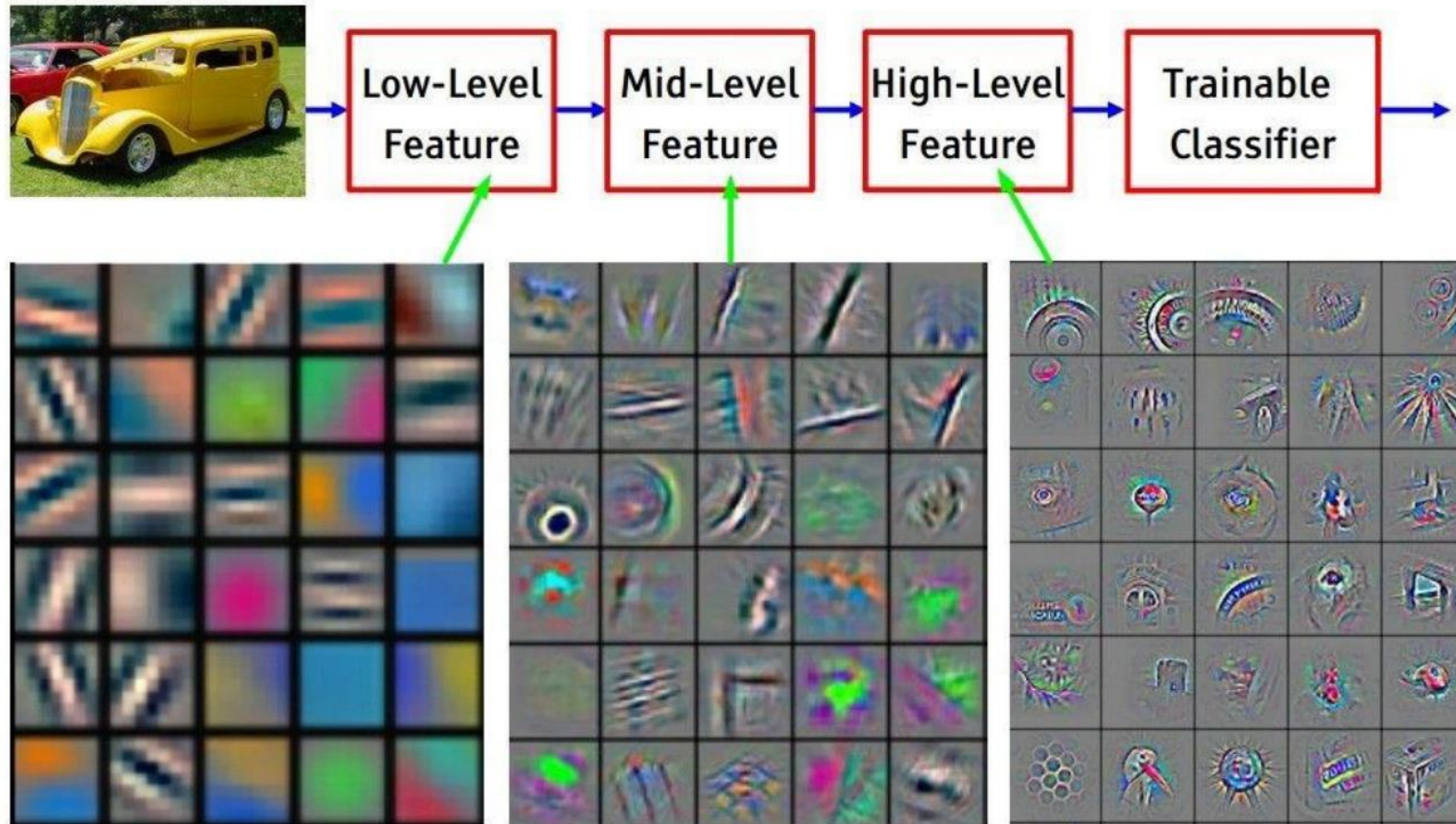
- **Pooling:** capas que permiten reducir la dimensionalidad de los feature maps y del número de parámetros de la red
 - Ventanas de agregación con **stride** y **padding**
 - Se aplican a menudo después de las capas convolucionales
 - Las funciones de agregación más usadas son Max, Average, L2-Norm
 - Simplifica la red (\rightarrow overfitting), reduciendo importancia a datos intermedios buscando robustez e invarianza de traslación a pequeños cambios de los inputs (e.g. efecto de mover a la derecha los pixeles de una imagen)
 - Permite reducir el número de operaciones, y otorgar a las capas posteriores una visión mas amplia de los datos originales



REDES CONVOLUCIONALES



REDES CONVOLUCIONALES

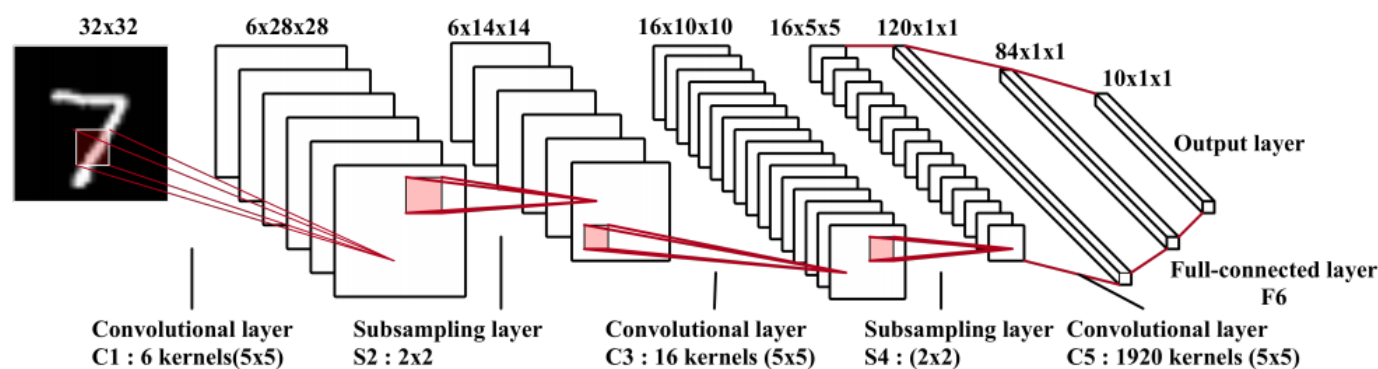


Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

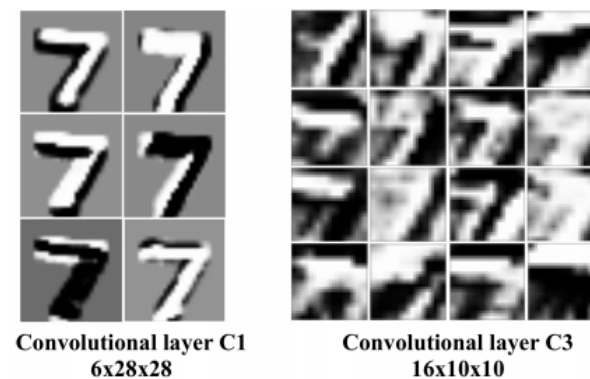


REDES CONVOLUCIONALES

- Las redes convolucionales están comúnmente compuestas de **capas convolucionales** que se utilizan como extractores de features, que al final son “aplanados” y utilizados como inputs de **capas densas** tradicionales que permiten realizar por ejemplo tareas de detección de objetos.



(a) LeNet-5 network

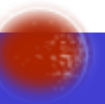


(b) Learned features

<https://arxiv.org/pdf/1512.07108.pdf>



DATOS LIMITADOS



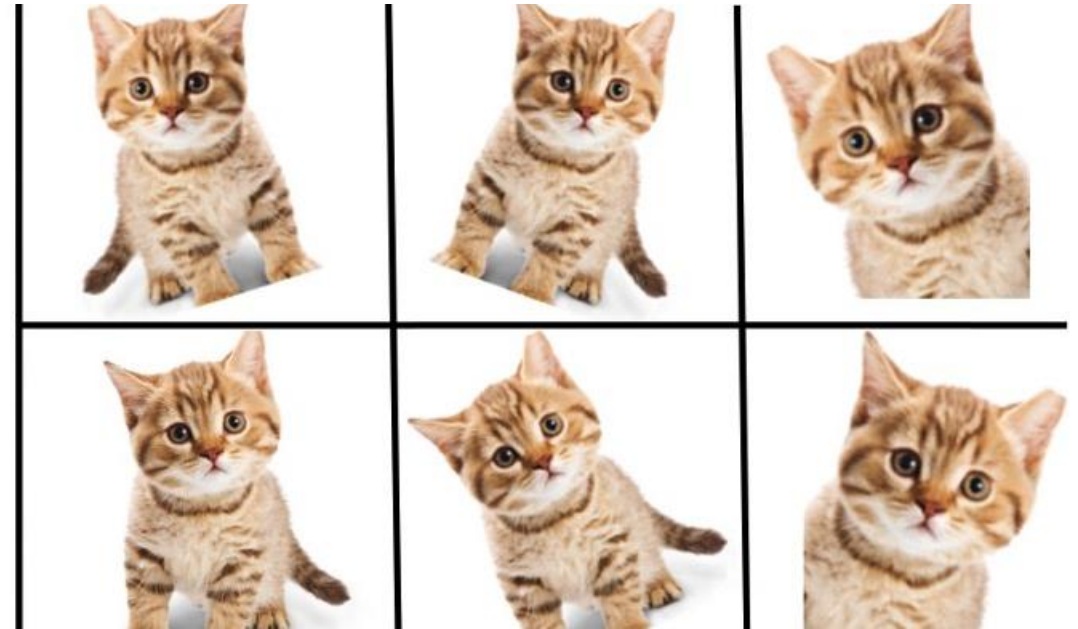
DATOS LIMITADOS

- Si la complejidad de la tarea a resolver es alta, el número de parámetros y la profundidad de la red debe ser elevado
- Necesidad de grandes conjuntos de datos para poder entrenar una red profunda
- Si no hay suficiente puede tender al **overfitting** o no converger
- A menudo, no hay datos suficientes...



DATA AUGMENTATION

- Agrandar el dataset a partir de transformaciones plausibles de los datos disponibles:
 - Espejos
 - Translaciones
 - Rotaciones
 - Deformaciones
 - Recorte
 - Coloraciones
 - Iluminación
 - Zoom
 - Cambio de fondo
 - Agregar ruido
- **Objetivo:** mejorar los modelos entrenados

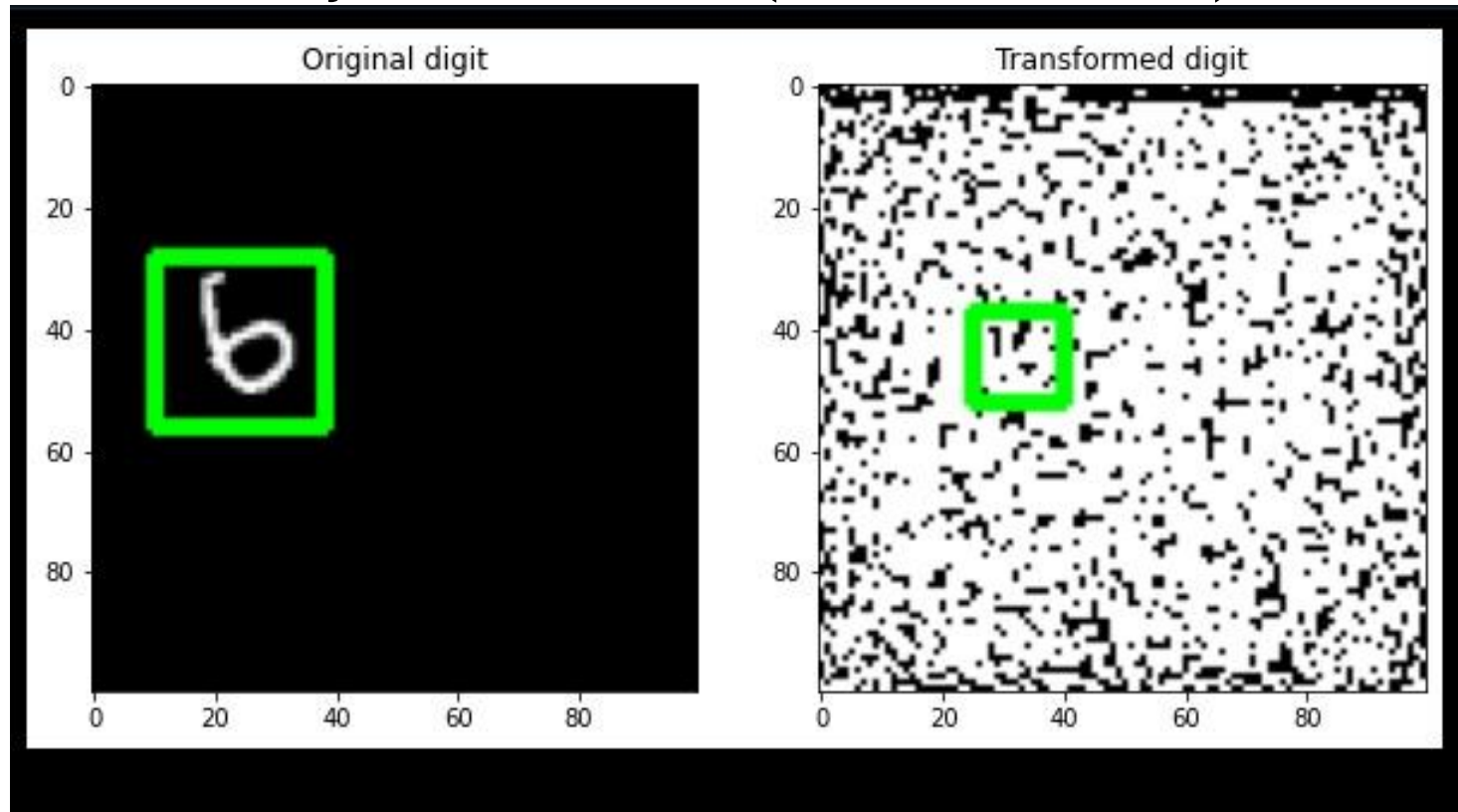


<https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced>



DATA AUGMENTATION

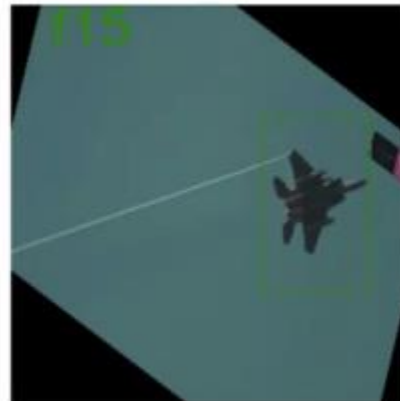
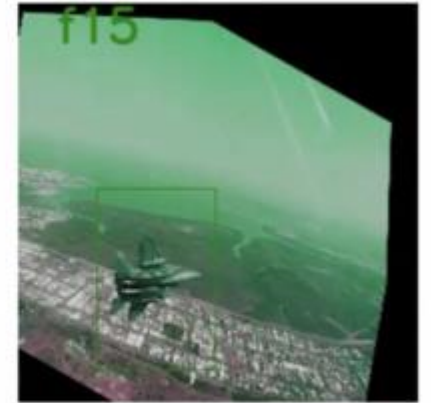
Ojo con los abusos (caso real de taller)



DATA AUGMENTATION

- No es lo mismo que tener datos nuevos: hay una correlación entre los datos originales y los aumentados
- Aún cuando se tiene un gran volumen de datos, el data augmentation puede ayudar a evitar sesgos de los datos (e.g. fotos tomadas siempre desde el mismo ángulo)
- **Offline augmentation:** creación y persistencia de los datos aumentados
- **Online augmentation:** los datos son distorsionados durante el entrenamiento después de ser cargados en memoria y antes de ser presentados al algoritmo de aprendizaje





TALLER RED CONVOLUCIONAL

Desarrollar taller introductorio de CNN



OVERFITTING

MACHINE LEARNING GENERALIZATION FINDING THE PERFECT FIT

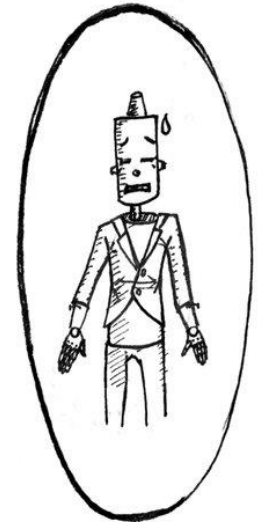
UNDERFIT



GOLDILOCKS ZONE

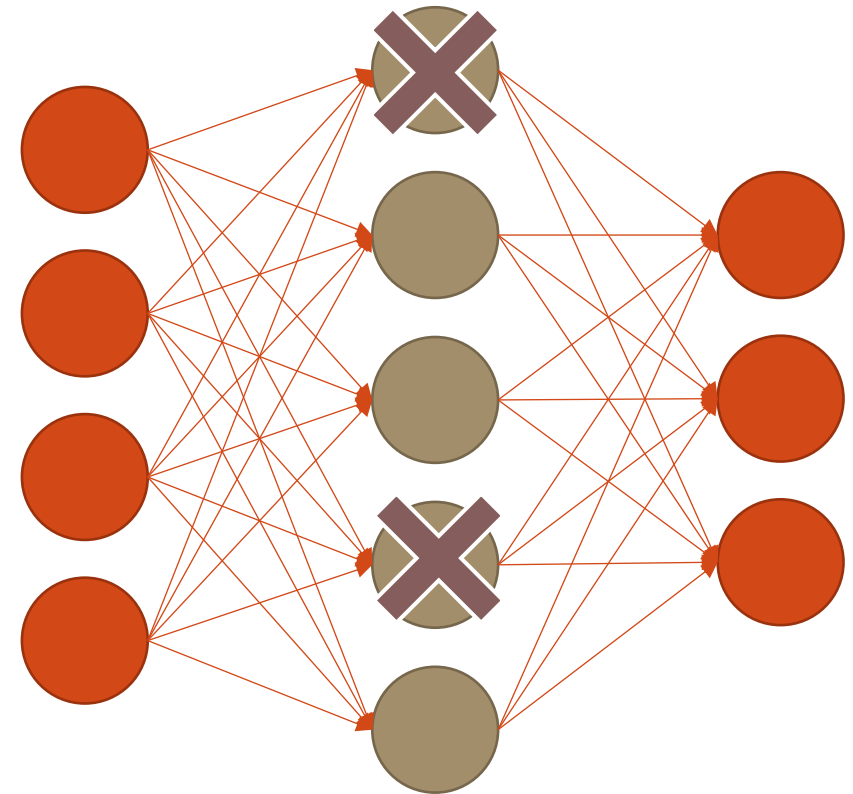


OVERFIT



OVERFITTING

- **Early Stopping:** detectar cuando no se han logrado mejoras en el los, parando el entrenamiento preventivamente
- **Dropout:** Técnica de regularización para luchar contra el overfitting
 - Forzar aleatoriamente un porcentaje de las neuronas de una capa a “apagarse” cambiando sus activaciones a 0.
 - Necesidad de hiper-parámetro de probabilidad de dropout (e.g. 50%)
 - Solo se debe aplicar durante el entrenamiento (nunca durante la predicción), y las neuronas a “apagar” se actualizan para cada instancia de aprendizaje
 - Se debe aplicar después de la función de activación, o de las capas del pooling (si las hay)
 - Se puede aplicar en conjunción con regularización L1 o L2.

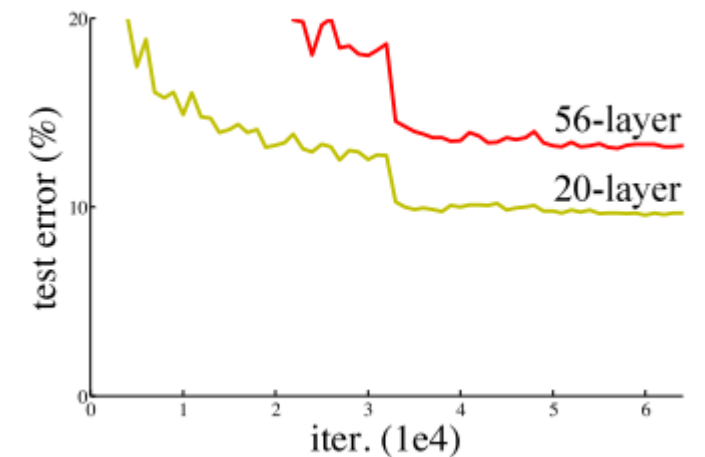
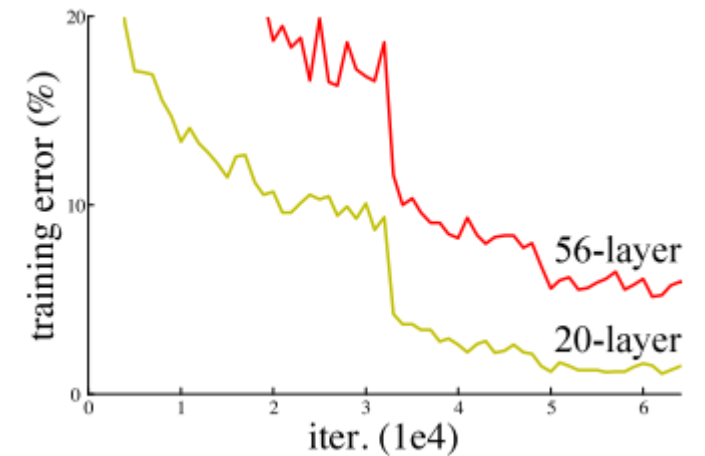


Dropout ($p=40\%$)



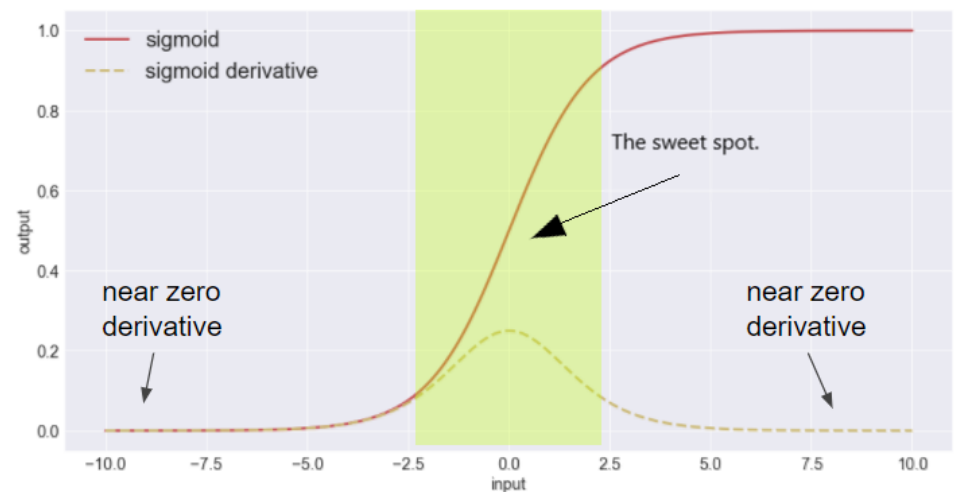
DESVANECIMIENTO / EXPLOSIÓN DEL GRADIENTE

- **Problema del desvanecimiento del gradiente:** A medida que las redes son más profundas, la multiplicación sucesiva del back propagation hace que los gradientes se acerquen cada vez a valores infinitesimalmente pequeños, lo que conlleva a un empeoramiento en los resultados. Se encuentra con funciones de activación tradicionales (logit, tanh)
- **Problema de la explosión del gradiente:** problema análogo con funciones de activación que pueden tener gradientes importantes muy importantes (sobre todo en modelos recurrentes)
- Las primeras capas son las más importantes ya que aprenden los patrones de base, pero son entonces las más difíciles de entrenar → Tiempo de entrenamiento alto y degradación de los resultados del modelo



DESVANECIMIENTO / EXPLOSIÓN DEL GRADIENTE

- Técnicas para combatirlo:
 - Pre entrenar capa por capa (años 90s)
 - RBMs (Restricted Boltzman Machines – 2006, Hinton) y Deep Belief Networks
 - Dejar de usar sigmoide y tanh: usar ReLUs que solo saturan en un sentido
 - Agregar una capa de salida en la mitad del modelo para agregar una **supervisión adicional**
 - Inicialización de los pesos (**Xavier / Glorot**): usar factor multiplicativo inversamente proporcional al numero de neuronas y de entradas de la capa en cuestión.
 - **Batch normalization**
 - Conexiones **residuales** (ResNet)



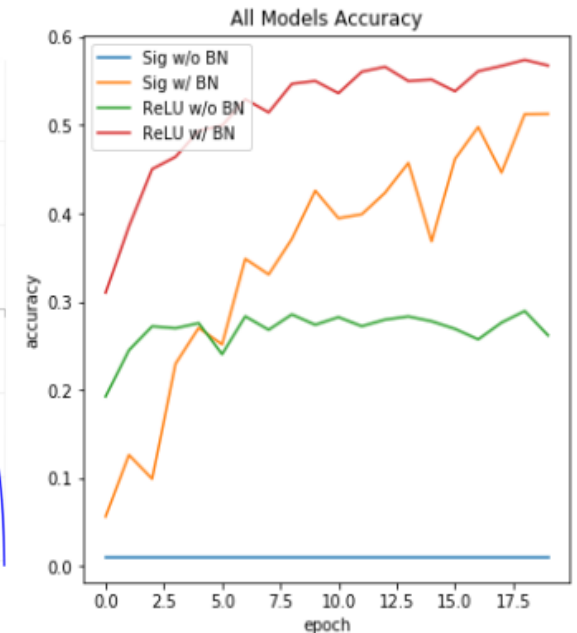
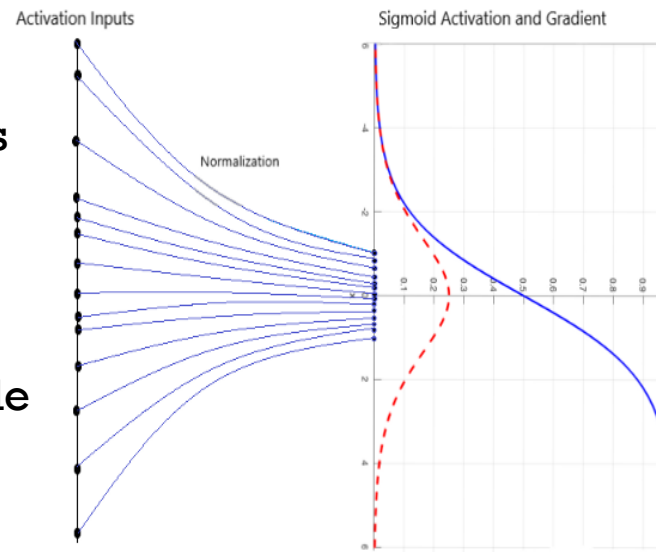
<https://towardsdatascience.com/intuit-and-implement-batch-normalization-c05480333c5b>



OVERFITTING & MEJORAS

■ Batch normalization:

- La distribución de los inputs a las capas cambian dado cada mini batch de entrenamiento (internal covariant shift)
- Difícil de entrenar cuando se saturan las no linealidades de las funciones de activación.
- Se aplica antes de la función de activación
- Estandarizar los inputs de cada mini batch antes de pasarlos por la función de activación (Z's) de cada capa ("whitening"), logrando:
 - un efecto de regularización
 - reducir el tiempo de entrenamiento
 - utilizar mayores learning rates
 - reducir sensibilidad de pesos iniciales



TALLER

Desarrollar los puntos del taller dedicados a la utilización de capas de dropout y de batch normalization



TALLER

Desarrollar la parte del taller dedicada al data augmentation para reducir el overfitting



TRANSFER LEARNING

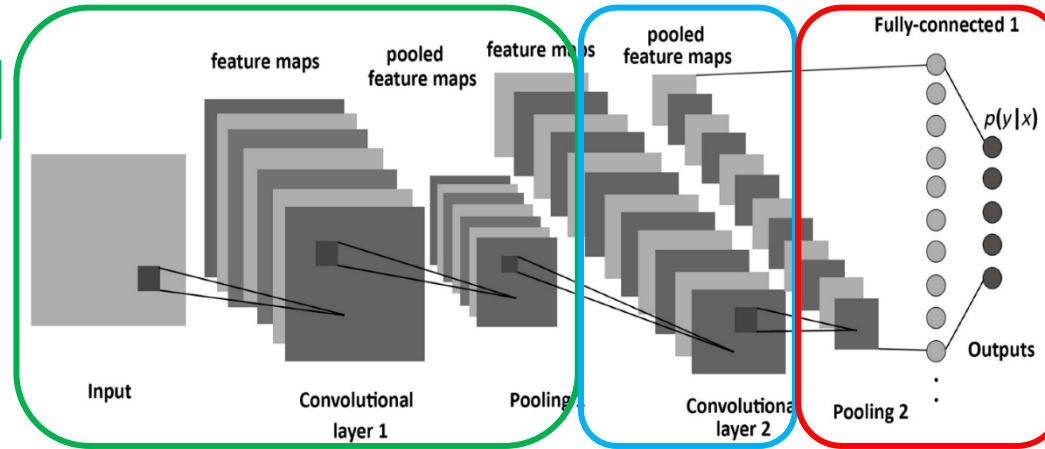
- El entrenamiento de una red convolucional importante requiere de:
 - Un conjunto grande de datos (e.g. 15 millones de imágenes de ImageNet)
 - Una infraestructura computacional importante en cuanto a recursos de procesamiento (GPUs, memoria)
 - Tiempo de entrenamiento
- Idea: poder reutilizar trabajos previos de entrenamiento de redes



TRANSFER LEARNING

Congelar

Re-entrenar
(fine tuning)



Remplazar

- En las CNNs, se tienen diferentes niveles de abstracción en cada capa, dada su profundidad
- Los filtros aprendidos pueden ser lo suficientemente generales como para poder ser aplicados a otras tareas con otros datasets:
 - Reutilizar una gran parte de la arquitectura de una red original (se congelan las capas correspondientes para que no sean reentrenadas)
 - Eliminar capas densas finales de clasificación, crear nuevas capas para la tarea en cuestión.
 - Reentrenar las últimas capas convolucionales que hayan aprendido filtros mas específicos a la tarea original

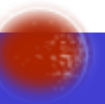


TRANSFER LEARNING

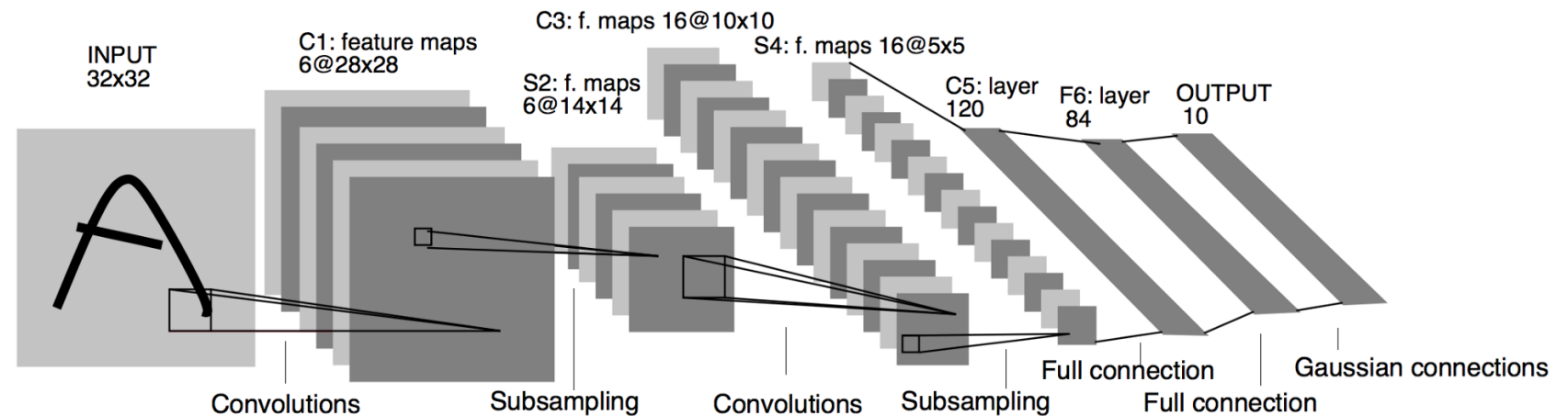
- Con redes convolucionales, una manera de ahorrar en tiempo de cómputo, en el caso de no tener que reentrenar las capas convolucionales, consiste en pasar el training set por la red importada, aplanar la salida y grabar cada feature map aplanado en archivo. Luego se crea una red “shallow” que consista en unas cuantas capas densas, que se entrenará sobre los archivos previamente procesados.
- El número de capas a reentrenar depende de la cantidad de datos disponibles. Entre mas datos, se vuelve posible reentrenar mas capas finales
- Es posible reentrenar toda la red, simplemente utilizando los valores de los parámetros transferidos como inicialización de los pesos de la red.



ARQUITECTURAS CONVOLUCIONALES



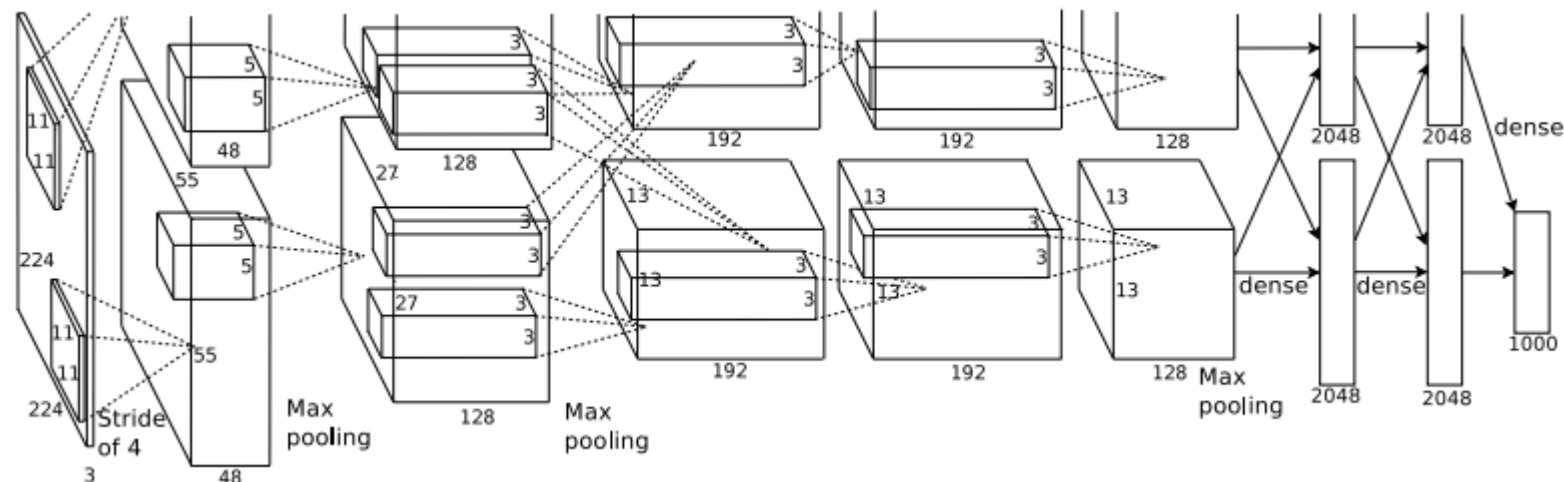
LENET-5



- Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86(11): 2278–2324, 1998.
 - MNIST 60000 ejemplos en escala de grises.
 - Utilizaba funciones de activación sigmoides y tanh. Aún no se usaba la ReLU.
 - En la época no se usaba Padding ni Dropout, y se aplicaba una función no lineal después de las capas de Pooling.
 - La capa final utilizaba una técnica que ya no se usa aplicando distribuciones gaussianas para cada dígito.
- Arquitectura:
 - 2 capas convolucionales con 6 y 16 filtros 5x5 con stride de 1. $(5*5*1*+1)*6=156$ params + $(5*5*6*+1)=272$ params → 428
 - Se aplanan la salida convolucional ($5*5*16 = 400$ neuronas) y se conecta con capas densas sucesivas de 120, 84 y 10 neuronas. $((400*120)=48000)+((120*84)=10080)+((84*10=840)$ params → 58920
 - Casi 60K parámetros, la mayoría en la parte densa



ALEXNET



- A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012
 - Ganador del challenge ILSVRC 2012 (15,4% de error vs 26,2% del segundo). Desató la moda del DL
 - Usó ReLU, dropout, SGD con momentum, weight decay del learning rate, Softmax y data augmentation
 - 1,2 millones de imágenes de entrenamiento, 1000 clases. 6 días de entrenamiento en 2 GPUs GTX 580 de manera paralela (cada una se ocupa de la mitad de la red)
- Arquitectura:
 - Conv 11x11x96 con stride 4, seguida de MaxPooling 3x3 con stride 2 → 55x55x96
 - Conv 5x5x256 SAME padding, seguida de MaxPooling 3x3 con stride 2 → 27x27x256
 - Conv 3x3x384 SAME padding, seguida de otra Conv 3x3x384 SAME padding → 13x13x384
 - Conv 3x3x256 SAME padding, seguida de MaxPooling 3x3 con stride 2 → 13x13x256
 - Salida convolucional aplanada en 9216 neuronas que entran a dos capas densas sucesivas de 4096 y 4096 neuronas
 - Salida de la red con una capa Softmax de 1000 neuronas
 - > 60 millones de parámetros (solo posible al tener un dataset de entrenamiento tan grande)



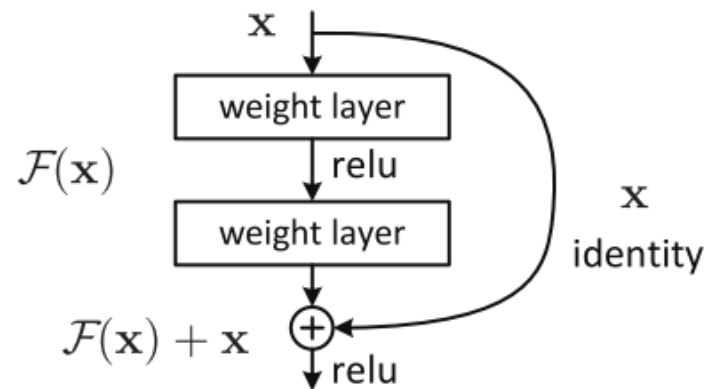
VGG-16



- K. Simonyan and A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR 2015
 - 2o puesto ILSVRC 2014, 7,3%
 - Usó Convs 1x1 para reducción de profundidad de los feature maps, Softmax
- Arquitectura: Simplicidad y profundidad, todas las capas se basan en Convs con filtros 3x3 con stride 1 y SAME padding, seguidos de MaxPooling 2x2 con stride 2)
 - 1 bloque de 2 Conv 3x3x64 seguido de MaxPooling → 112x112x64
 - 1 bloque de 2 Conv 3x3x128 seguido de MaxPooling → 56x56x128
 - 1 bloque de 3 Conv 3x3x256 seguido de MaxPooling → 28x28x256
 - 1 bloque de 3 Conv 3x3x512 seguido de MaxPooling → 14x14x512
 - 1 bloque de 3 Conv 3x3x512 seguido de MaxPooling → 7x7x512
 - Salida convolucional aplanada en 25088 neuronas que entran a dos capas densas sucesivas de 4096 y 4096 neuronas
 - Salida de la red con una capa Softmax de 1000 neuronas
 - > 138 millones de parámetros (solo posible al tener un dataset de entrenamiento tan grande)



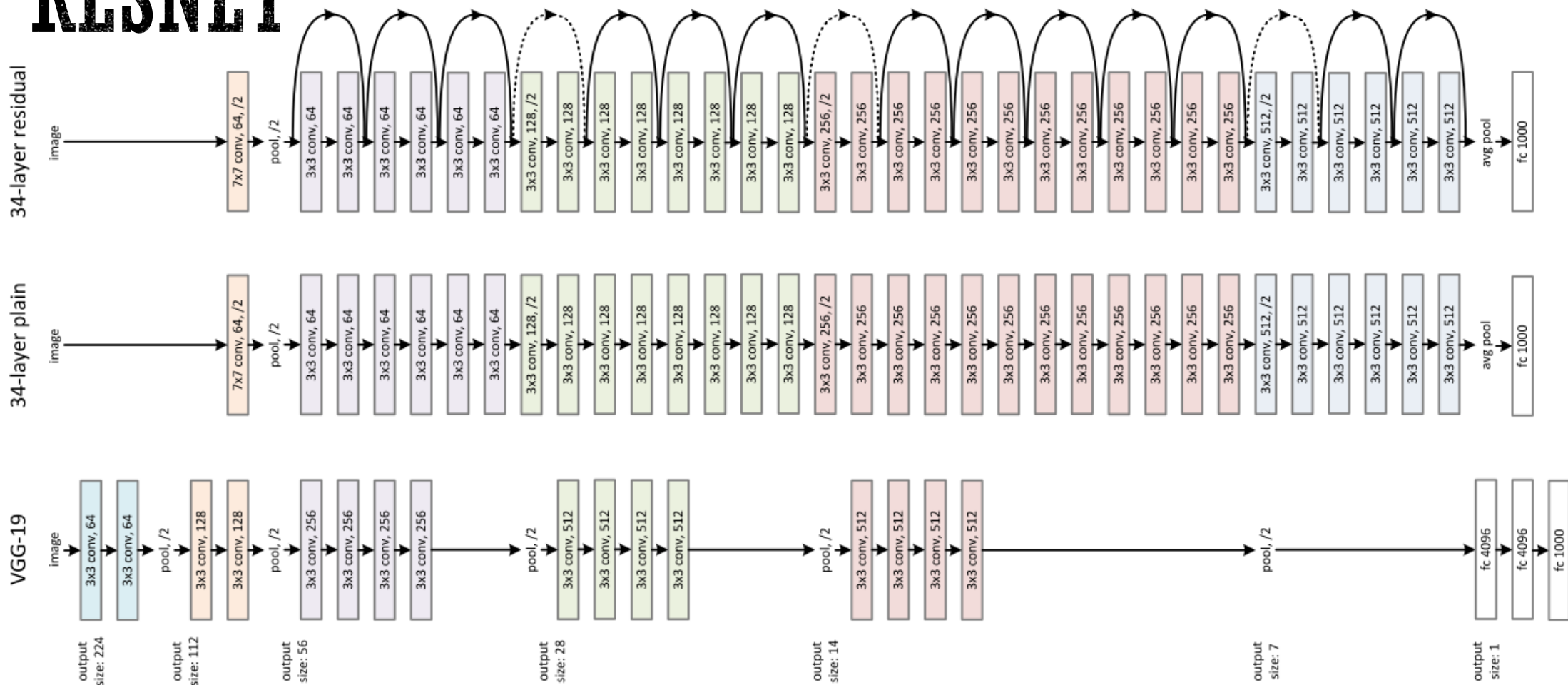
RESNET



- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, Deep Residual Learning for Image Recognition, CVPR 2016
 - Ganador ILSVRC 2015, 3,6% error (menor al error humano – entre 5% y 10%) y COCO 2015: ambas en las categorías de detección y localización,
 - Agregar una nueva capa no debería degradar los resultados (e.g. capa identidad)
 - Combate el problema del **vanishing gradient** efectivamente
 - Converge más rápidamente que los modelos correspondientes sin conexiones residuales
- Arquitectura
 - **Bloque residual:** proyecta la salida de una capa convolucional como parte de la entrada de la capa que se encuentra dos pasos mas adelante y se combina con la agregación lineal de la misma antes de la función de activación correspondiente (ver imagen).
 - Los bloques residuales se ponen uno después de otro, tantos como se quieran (se entrenó un ResNet con 1001 bloques), seguidos por capas de MaxPooling para reducir las dimensiones del filtro convolucional mientras se aumenta el número de filtros

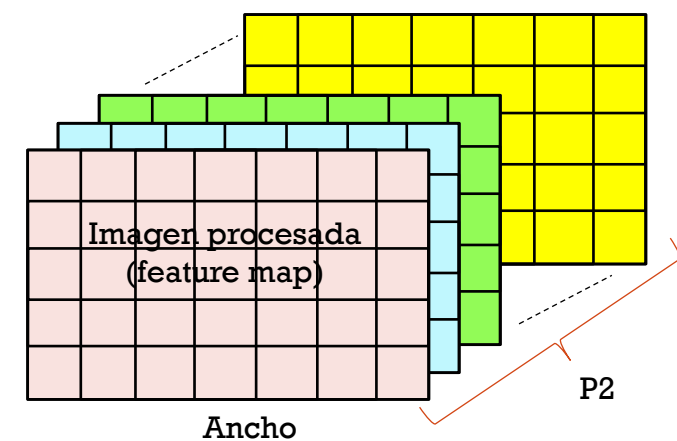
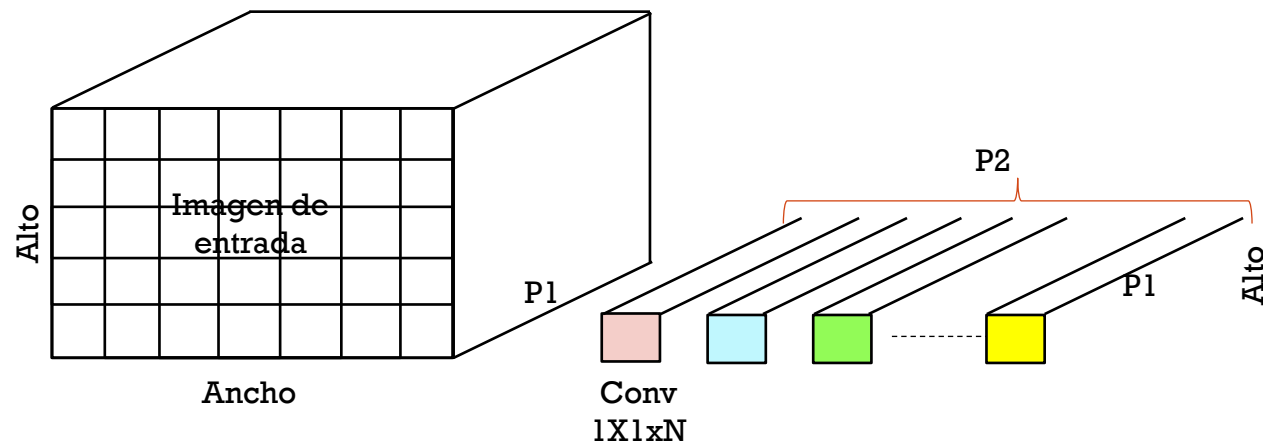


RESNET



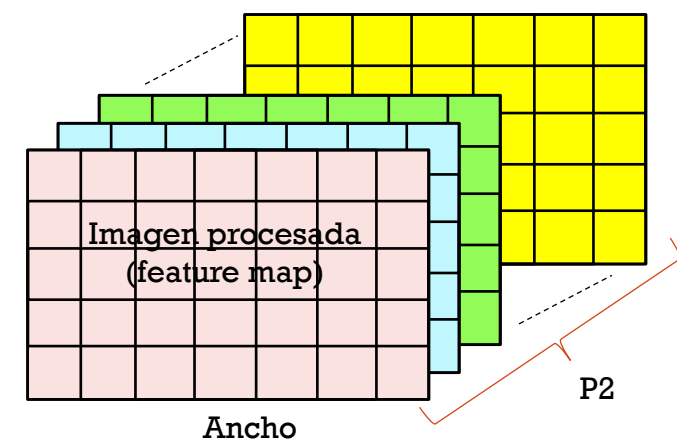
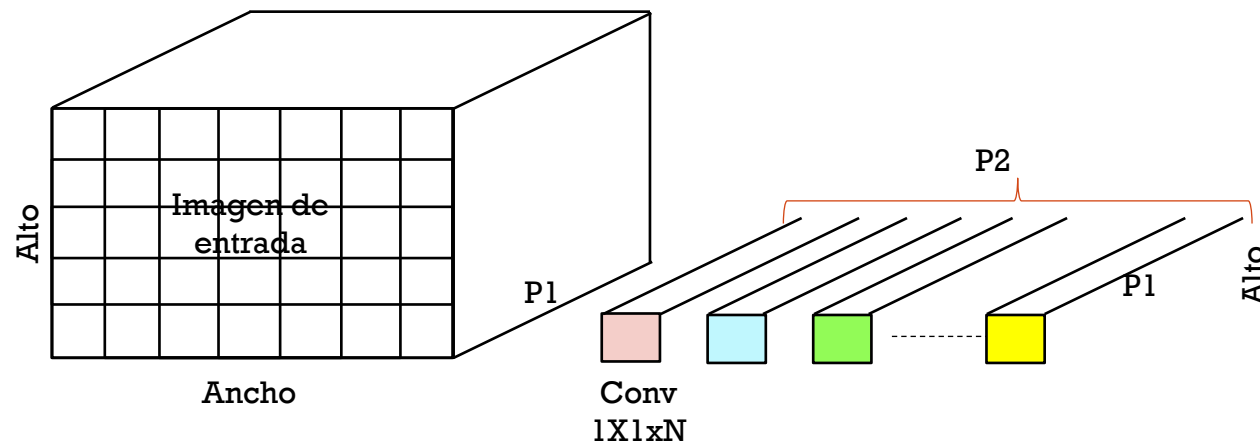
CONVOLUCIONES 1X1

- En el caso de un feature map de entrada de profundidad 1 (e.g., imagen en escala de grises) → Multiplicación por escalar
- En el caso de múltiples canales de entrada → producto punto de los $P1$ canales de entrada por los filtros $1 \times 1 \times P1$
- La función de activación se encarga de transformar el valor escalar final
- Se pueden aplicar $P2$ filtros convolucionales 1×1 , con $P2 < P1$, de tal manera que se pasa de $P1$ canales a $P2$ canales.
- Para reducir la representación tenemos entonces las capas Pooling que reducen el alto y ancho, y las convoluciones 1×1 , que reducen la profundidad



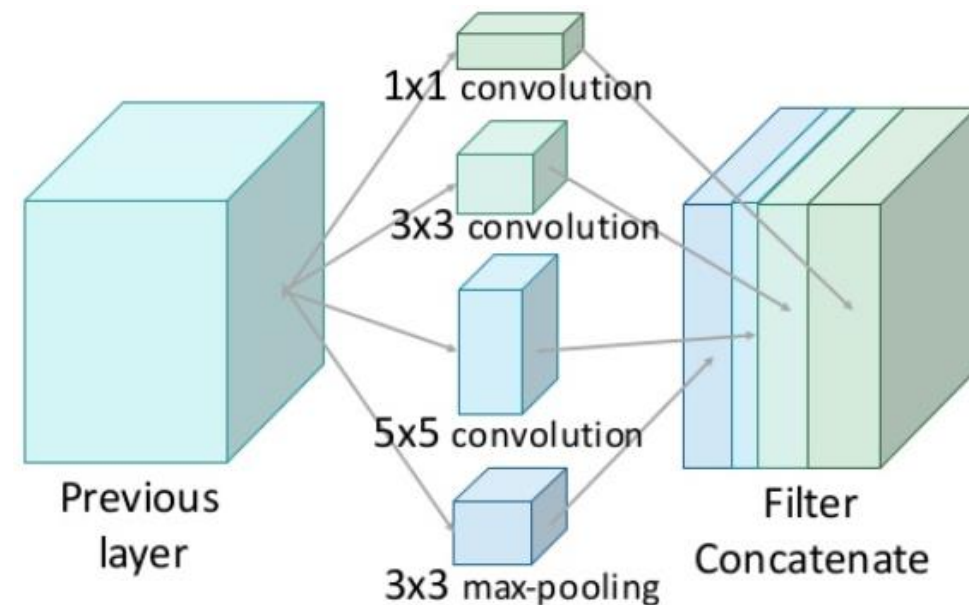
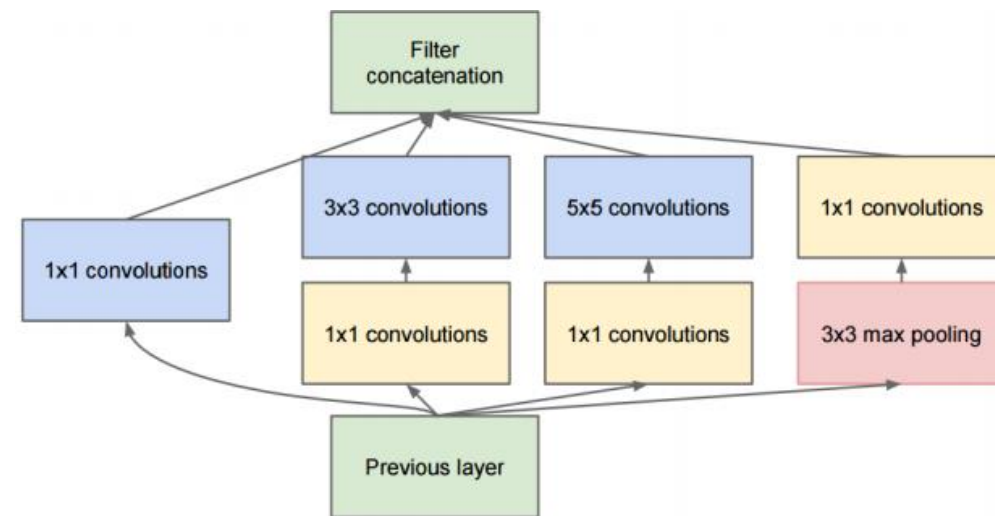
CONVOLUCIONES 1X1

- Si vamos a aplicar una convolución SAME de 32 filtros 5x5 a un feature map de entrada de 28x28 con 192 canales de profundidad,
 - ¿Cuáles van a ser las dimensiones del tensor de salida?
 - ¿Cuántos cálculos se necesitan?
- Si utilizamos una convolución 16 filtros 1x1 intermedia (“bottleneck”), seguida de una convolución SAME con 32 filtros 5x5, ¿qué tanto mejora la situación?
- Se puede reducir el tamaño de representación y de cálculos sin perjudicar el performance de la clasificación



INCEPTION

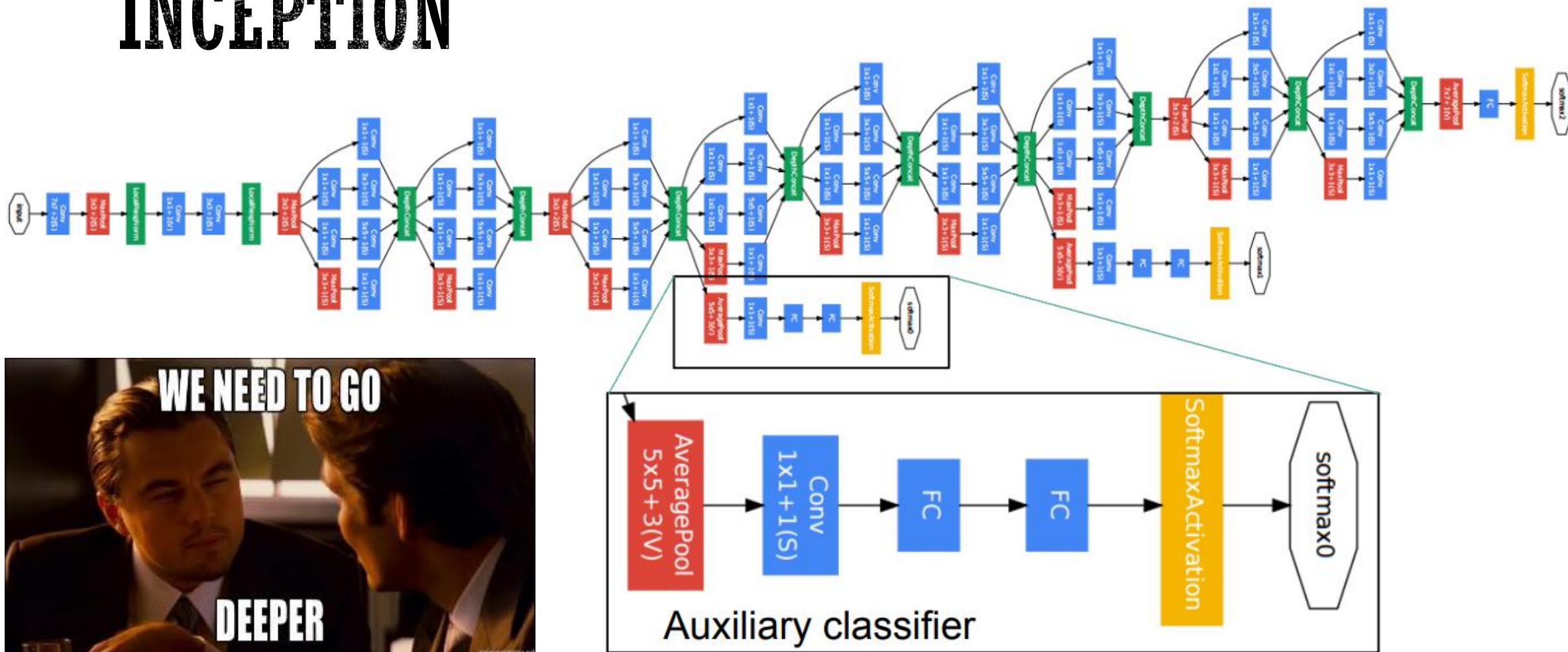
- C. Szegedy et al., Going deeper with convolutions, CVPR 2014
 - Idea de base: por qué hacer una selección entre una conv 1×1 , 3×3 , 5×5 o un pooling (same), si podemos hacerlas todas a la vez?
 - Definición de módulos que se reutilizan a lo largo de la red (se actualizan en Inception v2, v3, v4)
 - Cada modulo combina los resultados de 4 tipos de secuencias de tratamiento convolucional, cada uno debe producir feature maps de las mismas dimensiones, que se concatenan en profundidad de canales
 - Se utilizan adicionalmente clasificadores auxiliares temprano en la red para asegurarse que los features en las capas intermedias no sean tan alejados de lo que se busca.



Szegedy et al, 2014



INCEPTION



GENERALIDADES ARQUITECTURALES

- Globalmente las arquitecturas tienen dos partes: una convolucional que sirve para extraer features, y una densa que sirve para predicción
- A medida que se adquiere profundidad en la capa, la altura y el ancho de los feature maps disminuyen, mientras que el número de canales aumenta
- Es común encontrar capas de pooling después de 1, 2, o 3 capas convolucionales como método de reducción del tamaño de los feature maps
- Los filtros no deben ser muy grandes ya que pueden ignorar información relevante para las capas subsiguientes (e.g. filtro 11×11 de Alexnet en la primera Conv)
- Es preferible utilizar padding SAME en las capas convolucionales, preservando el tamaño de los feature maps, y reducirlos solo a partir de capas pooling.



TALLER

Desarrollar la parte del taller dedicada a la aplicación del transfer Learning de modelos previamente entrenados



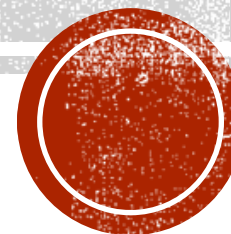
HIPERPARÁMETROS

- Topología de la red:
 - Número de capas de la red, conexiones recursivas
 - Número de neuronas, número de filtros convoluciones por cada capa
 - Dimensiones de las convoluciones y poolings (kernels, strides, paddings)
 - Regularización: probabilidades de dropout, nivel de L2, batch norm
 - Inicialización: factor multiplicativo para los pesos
 - Transfer Learning: modelo de base, fine tuning
- Optimizador / entrenamiento:
 - Learning rate, momentum decay, learning rate decay, ...
 - Número de épocas de entrenamiento, tamaño del batch
- Feature engineering:
 - Tamaño de imágenes, ventanas de secuencias
 - Data augmentation: ángulos de rotación, niveles de deformación, color shifting, zoom, recorte

Encontrar los buenos ajustes de los hiperparámetros hacen que se diga que el “Deep Learning es una arte oscura”



GRACIAS



REFERENCIAS

- *Learning TensorFlow*, Tom Hope, Yehezkel S. Resheff & Itay Lieder, O'Reilly 2017
- *Introduction to TensorFlow*, Chris Manning & Richard Socher, Lecture 7 of the CS224n course at Stanford University, 2017
- *Hands-On Machine Learning with Scikit-Learn and TensorFlow*, Aurélien Géron, 2017
- *Machine learning with TensorFlow*, Nishant Shukla, Manning, 2018
- *Python Machine Learning (2nd ed.)*, Sebastian Raschka & Vahid Mirjalili, Packt, 2017
- *TensorFlow for Deep Learning*, Charath Ramsundar & Reza Bosagh Zadeh, O'Reilly, 2018
- *Deep Learning with Python*, Francois Chollet, Manning 2018
- *Neural Networks and Deep Learning*, Andrew Ng, Coursera, 2017

