Visión por Computador II

CEAI, FIUBA

Profesor: Javier Kreiner, javkrei@gmail.com

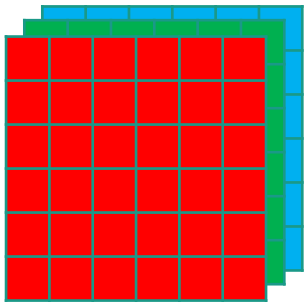
Tercera clase:

- VGG Network
- Programación
  - Implementación de una VGG Network
- Residual networks, bloque residual
- Ejemplo: Red residual vs red ‘normal’
- Transfer Learning
- Programación
  - Utilizar VGG Network pre-entrenada para el problema de gatos/perros
  - Utilizar ResNet-50 pre-entrenada para el mismo problema

VGG - 16

~ 138 M parámetros

CONV = 3×3 filter, s = 1, same MAX-POOL = 2×2 , s = 2



224×224 ×3

→ [CONV 64] ×2 → 224×224×64 → POOL → 112×112 ×64 → [CONV 128] ×2 → 112×112 ×128 → POOL → 56×56 ×128

→ [CONV 256] ×3 → 56×56 ×256 → POOL → 28×28 ×256 → [CONV 512] ×3 → 28×28 ×512 → POOL → 14×14×512

→ [CONV 512] ×3 → 14×14 ×512 → POOL → 7×7×512 → FC 4096 → FC 4096 → Softmax 1000

[Simonyan & Zisserman 2015. Very deep convolutional networks for large-scale image recognition]

VGG-16

- ILSVRC 2014
- La misma estructura se repite
- Receptive fields pequeños, de 3x3
- 16 capas
- ReLU como no-linearidad
- 3 FC layers al final
- Paper original: Very deep convolutional networks for large-scale image recognition -> https://arxiv.org/pdf/1409.1556
- 'Filosofía'/metodología de VGG:
  - si la salida de una capa tiene la misma resolución que la capa anterior, mantenemos la cantidad de canales
  - si la salida de una capa divide por dos cada dimensión, duplicamos la cantidad de canales

Keras, método funcional de armar redes

- Definir input: `inputs = Input(shape=(4,))`
- Definir capas ocultas:
  - `x = Dense(5, activation='relu')(inputs)`
  - `x = Dense(10, activation='relu')(x)`
- Definir outputs: `outputs = Dense(3, activation='softmax')(x)`
- Instanciar el modelo: `model = Model(inputs=inputs, outputs=outputs)`

Ejercicio de programación, implementar VGG con método funcional

- colab: https://colab.research.google.com/drive/1A3bP-1NMi-WnxzVySzUf0UfAFut2cE1R?usp=sharing

Problemas al entrenar redes muy profundas

- Gradientes que explotan o tienden a cero
  - Para este problema es común usar Batch Normalization
- Degradación de la performance de entrenamiento
  - Vamos a ver Redes Residuales para esto

Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

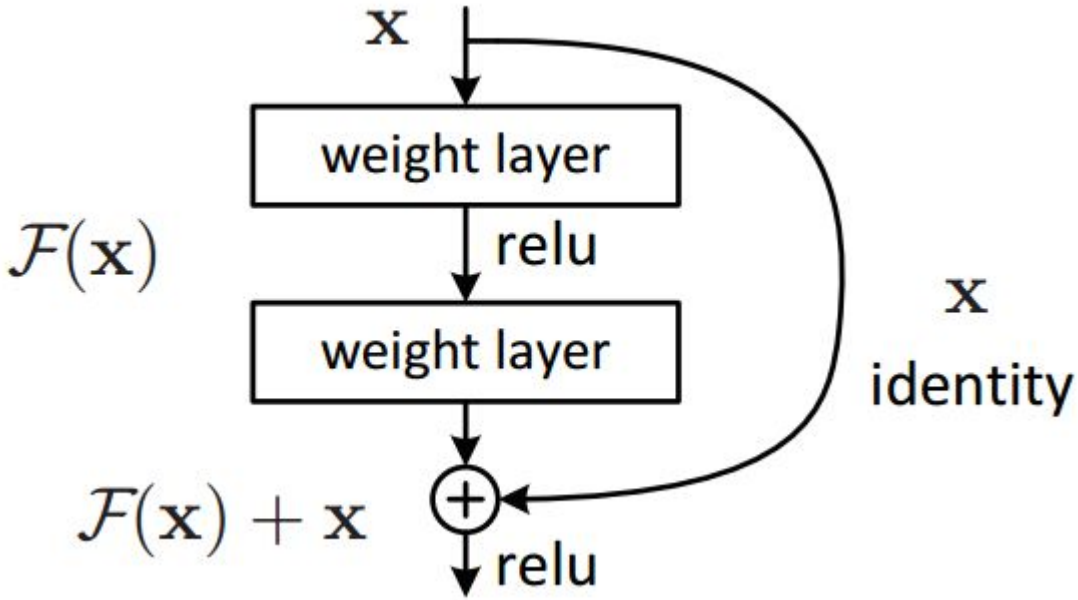
Redes residuales

- A medida que las redes se hacen más profundas (aumentan las capas) se vuelven más difíciles de entrenar, hay una degradación de la performance en el training set
- Esto indica que no todas las redes son igual de fácil de optimizar
- Si tomáramos una red y le agregamos una capa que sea la identidad, por construcción la red expandida tendría la misma performance
- Este argumento implica que la red extendida bien entrenada debe tener al menos la misma performance que la red original
- Los experimentos muestran que en la práctica se entrenan más fácil que una red ‘común’ para la misma complejidad

Bloque residual

- Supongamos que queremos aproximar $H(x)$
- Y supongamos que ya tenemos una aproximación con x
- Podemos en vez aproximar el residuo: $R(x) = H(x) - x$
- O sea $H(x) = R(x) + x$
- Esto es lo que se quiere lograr con las conexiones residuales o ‘skip connections’

Bloque residual

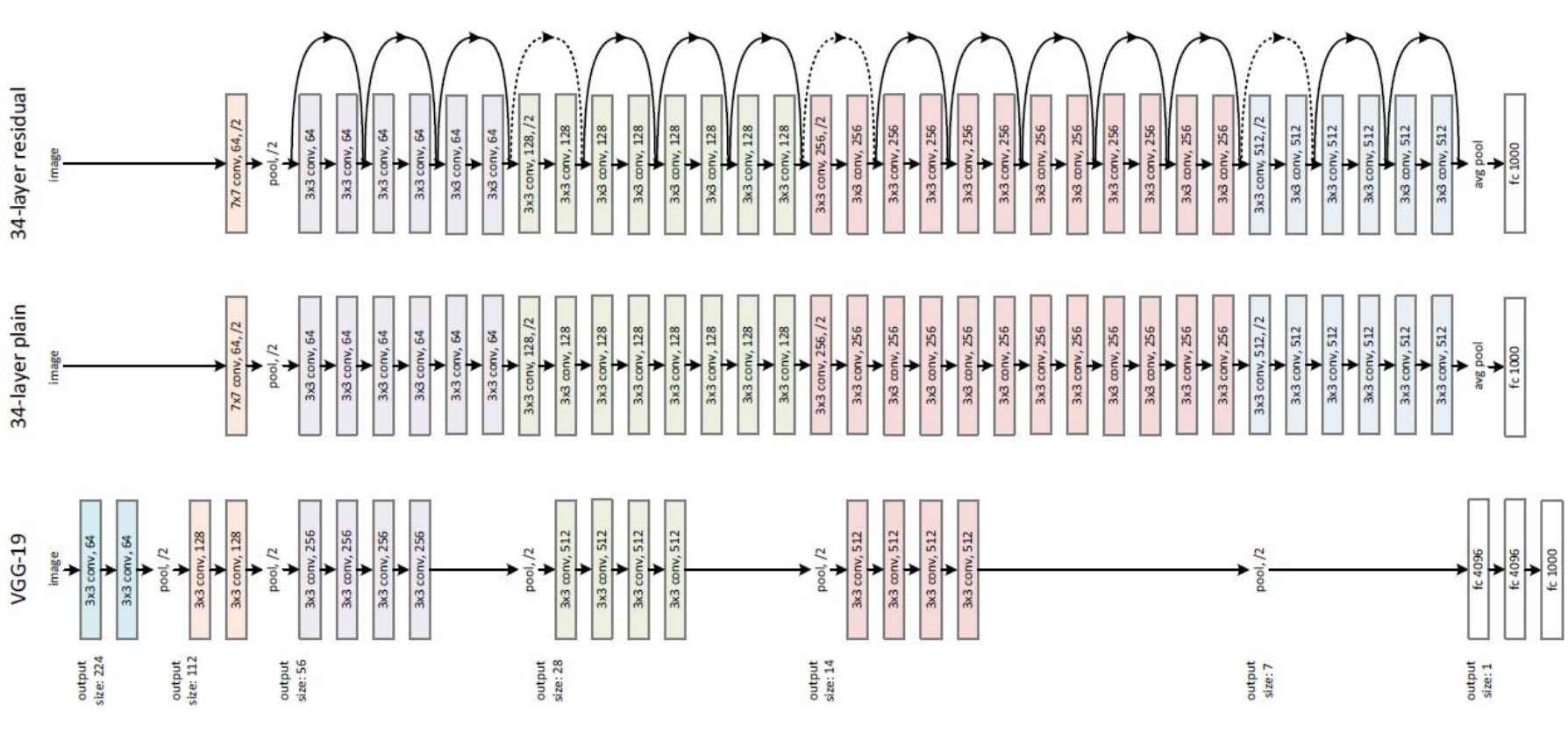


- paper: Deep Residual Learning for Image Recognition, https://arxiv.org/pdf/1512.03385.pdf

Comparación entre red residual y ‘normal’

- colab: https://colab.research.google.com/drive/1HOG1cwWKBkWj_LwDfsCPi5j544UP_88W?usp=sharing

Resnet



paper original: Deep Residual Learning for Image Recognition, https://arxiv.org/pdf/1512.03385.pdf

Cuadro comparativo

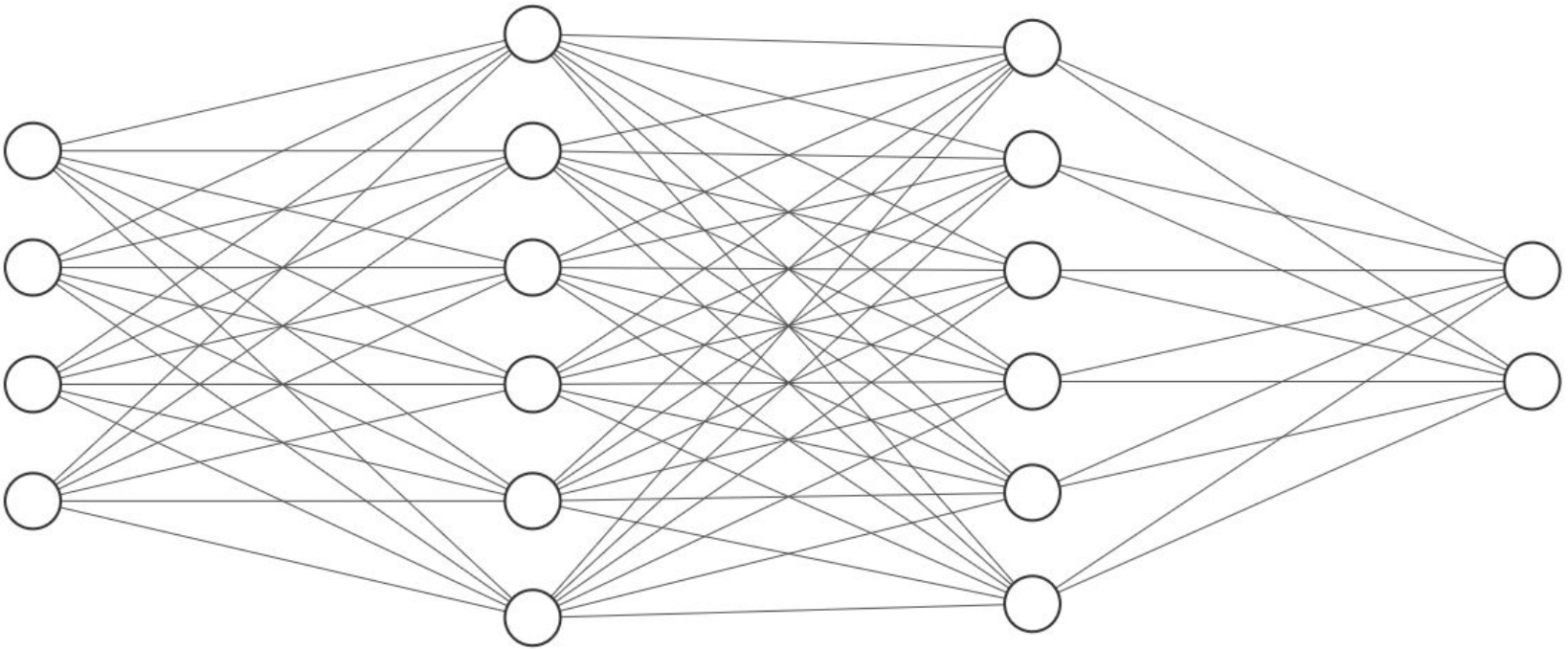
Models	Batch-size / Epochs	Baseline		# de parámetros
		Top-1	Top-5	
ResNet-50	1024 / 90	75.70%	92.78%	23 millones
GoogLeNet-v1	1024/ 80	69.26%	89.31%	4 millones
VGG-16	256 / 60	68.23%	88.47%	138 millones
AlexNet	1024 / 88	57.43%	80.65%	61 millones

Transfer learning

- Las redes actuales tienen muchos millones de parámetros
- Los datasets son también enormes y posiblemente privados
- Se requieren un gran poder de cómputo (tiempo/dinero) para entrenarlas
- Lo que podemos hacer es utilizar redes que ya fueron entrenadas (por otros, en internet) y utilizarlas para inicializar una red
- Posibilidades:
    - Congelar todos los layers excepto la última capa, reemplazar entrenar esa, en este caso se puede pre-computar y guardar a disco
    - Entrenar las las últimas capas, usando los pesos existentes para inicializar, o entrenarlas de cero
    - Reemplazar las última capas con otra arquitectura y entrenar
    - O usar los pesos como inicialización y entrenar toda la red

Depende de cuán grande sea nuestro dataset y de cuánto tiempo de cómputo tengamos

Transfer learning



Workflow de transfer learning con Keras

A. Opción uno

1. Instanciar un modelo base y cargarlo con sus pesos pre-entrenados
2. Congelar todos los layers del modelo base seteando `trainable=false`
3. Crear un nuevo modelo agregando capas luego de la salida del modelo base
4. Entrenar el modelo en el dataset que tenemos

B. Otra opción (más liviana):

1. Instanciar el modelo base y cargar los pesos pre-entrenados
2. Correr el modelo sobre nuevo dataset y guardar la salida. Con esto extraemos las features del modelo ya entrenado
3. Usar estas features como entrada para un modelo más pequeño

En esta segunda opción solo corremos el modelo pre-entrenado una vez en los datos.

Transfer Learning con Keras:

- colab: https://colab.research.google.com/drive/177_2H4LATJPmgqTg5xDOLkxdEo0G3xCX?usp=sharing

Tercer intento de resolver el problema con transfer learning

- Ejemplo con VGG pre-entrenada
- Ejemplo con ResNet pre-entrenada