



Visión por Computador II

CEAI, FIUBA

Profesores:

- Javier A. Kreiner, javkrei@gmail.com
- Andrés F. Brumovsky, abrumov@gmail.com

Segunda clase:

- Repasos de la clase pasada (y de algunas fórmulas)
- Experimento: Con Pooling vs. Sin Pooling
- Ejercicio:
 - ○ Primer intento de solución de un problema de Kaggle con redes convolucionales
 - Kahoot nº 1
- Data Augmentation
- Ejercicio:
 - ○ Segundo intento de solución del problema utilizando data augmentation
 - Algunas arquitecturas clásicas:
 - ○ LeNet-5
 - ○ AlexNet
- • Kahoot nº 2



Método para entregar los ejercicios

- Subir en su github las soluciones, y pasarnos el github
- El github puede contener también links a colabs, si eso es más cómodo
- Separar las clases en carpetas, o sea, todos los ejercicios de una clase en una carpeta, una carpeta para cada clase

Repaso

- 3 tipos de capas principales: CONV, POOL (AVG o MAX), DENSE/FC
- Parámetros de POOL: stride(s) y pool size(f)
- Parámetros de CONV: stride(s), filter size(f), padding(p), n_c (# de filtros)
- 'same' padding: p tal que la salida mide lo mismo que la entrada
- 'valid' padding: p=0
- tamaño de salida después de CONV:
 - $\text{floor}[(n + 2p - f) / s + 1] \times \text{floor}[(n + 2p - f) / s + 1] \times n_c$
- # de parámetros de CONV: $f^{[I]} \times f^{[I]} \times n_c^{[I-1]} \times n_c^{[I]} + n_c^{[I]}$ ← bias.
- tamaño de salida después de POOL:
 - ○ $\text{floor}[(n + 2p - f) / s + 1] \times \text{floor}[(n + 2p - f) / s + 1] \times n_c^{[I-1]}$
 - no tiene parámetros. ✗
- # de parámetros de DENSE/FC:
 - $n^{[I-1]} \times n^{[I]} + n^{[I]}$


Experimento - Con Pool vs Sin Pooling



- colab: <https://colab.research.google.com/drive/10fNVCqHkst5EYPFG2tCFV4-oVFJiUTc3?usp=sharing>

Ejercicio: entrenar una red para distinguir perros de gatos

(<https://www.kaggle.com/c/dogs-vs-cats>)

- hay que entrenar un clasificador que distinga entre perros y gatos
- vamos a usar un subconjunto de este dataset (4000 en vez de 25000 imágenes)
-  vamos a entrenar varias redes y usando diversas técnicas

Organización del dataset:

- Tenemos train, validation y test datasets
- Las imágenes están en directorios con los nombres respectivos
- En cada directorio (train, test, validation) hay un subdirectorio por clase (cat, dog)
- Las imágenes son importadas con valores de 0 a 255 (y tres canales, RGB)
- Usamos ImageDataGenerator para reescalarlas entre a valores en [0,1]
- Las imágenes tienen diferentes tamaños y por eso hay que preprocesarlas llevándolas todas a 150x150
- Para generar datasets usamos ImageDataGenerator.flow_from_directory con los parámetros correspondientes
- <https://keras.io/api/preprocessing/image/>

Ejercicio: entrenar una red para distinguir perros de gatos

(<https://www.kaggle.com/c/dogs-vs-cats>)

Estructura a diseñar:

- Capa convolucional de 32 filtros de 3x3, activación ReLU
- Max Pooling 2x2
- Capa convolucional de 64 filtros de 3x3, activación ReLU
- Max Pooling 2x2
- Capa convolucional de 128 filtros de 3x3, activación ReLU
- Max Pooling 2x2
- Capa convolucional de 128 filtros de 3x3, activación ReLU
- Max Pooling 2x2
- Capa Flatten
- Capa FC de 512 elementos, activación ReLU
- Capa FC de salida (1 elemento), activación sigmoidea
- Loss function: Binary Cross-entropy
- Optimizador; RMSprop (lr = 1e-4)

link al colab:

<https://colab.research.google.com/drive/1x2iV92KUCL2rr70p3Jwb9LMeMdnsmj7e?usp=sharing>

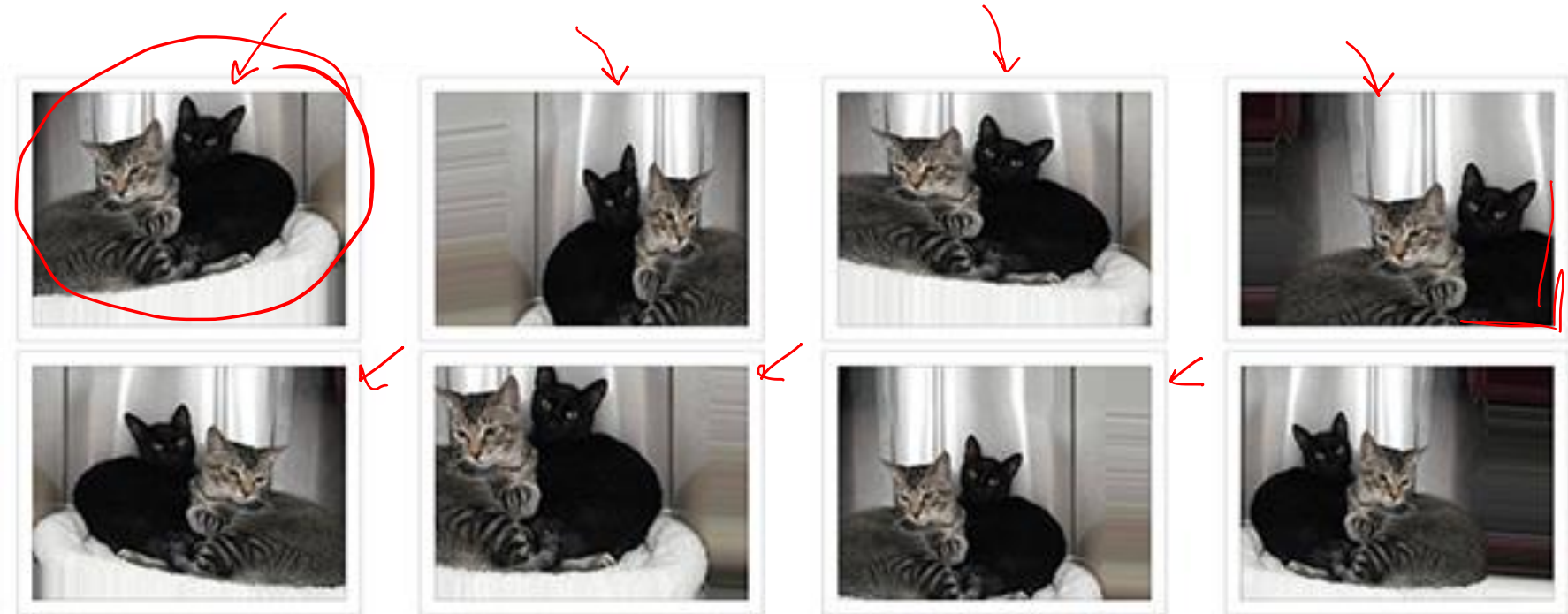


Kahoot de redes convolucionales 1

Data Augmentation

- Consiste en 'aumentar' la cantidad de datos que tenemos
- Se consigue con la generación de ejemplos modificando ejemplos ya existentes
- Las transformaciones dependen de la aplicación específica y las características de los datos
- En el caso de imagenes podemos:
 - rotar
 - escalar
 - distorsionar
 - invertir
 - agregar ruido
 - crop
 - zoom
 - deformaciones de luz, color, locales
 - etc.
- Si estas transformaciones son lógicas dados los tipos de datos, la red va a tener un conjunto de entrenamiento mayor















Transformaciones básicas



Transformaciones de color



Más Transformaciones

Original Image	Basic	Light deformation	Extreme deformation	Color deformation	Image overlapping	Background swapping
						
						

Ejercicio: Usar data augmentation para mejorar la performance



- link al dataset:
https://drive.google.com/file/d/1WgbH_Xt421hNhD4gcfwsvtVsFheJKefm/view?usp=sharing
- link al colab:
<https://colab.research.google.com/drive/1x2iV92KUCL2rr70p3Jwb9LMeMdnsmj7e?usp=sharing>

Algunas arquitecturas clásicas: LeNet-5

- A fines de 1989 Yann LeCun experimentaba con redes convolucionales
- Una muy conocida es LeNet-5
- Una aplicación importante fue al reconocimiento de dígitos
- Su estructura es representativa de la organización actual
- Paper original: http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf
- Aproximadamente 60,000 parámetros

Diagrama

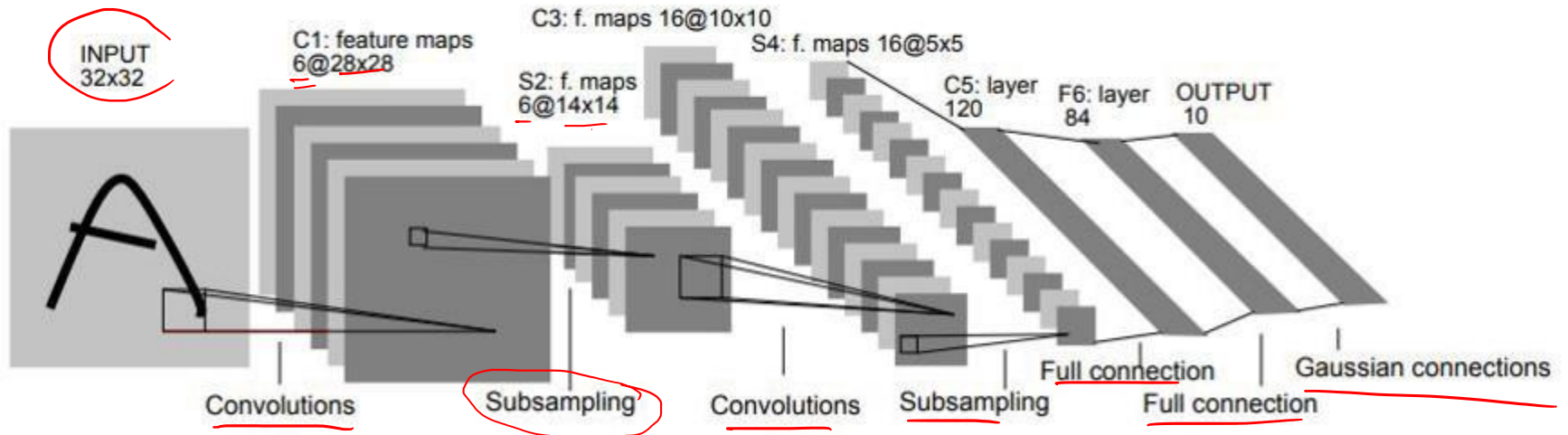
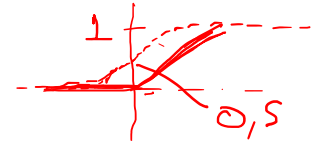


Tabla con la arquitectura



Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	32x32	-	-	-
1	Convolution	<u>6</u>	<u>28x28</u>	<u>5x5</u>	<u>1</u>	<u>tanh</u>
2	Average Pooling	6	14x14	2x2	2	<u>tanh</u>
3	Convolution	16	10x10	5x5	1	<u>tanh</u>
4	Average Pooling	16	5x5	2x2	2	<u>tanh</u>
5	Convolution	120	1x1	5x5	1	<u>tanh</u>
6	FC	-	84	-	-	tanh
Output	FC	-	<u>10</u>	-	-	<u>softmax</u>

Ejercicio: implementar LeNet-5 para reconocer dígitos

Algunas arquitecturas clásicas: AlexNet

- En 2012 causó un gran impacto por obtener un puntaje significativamente mayor que el segundo puesto en ImageNet Large-Scale Visual Recognition Challenge (ILSVRC), partir de ese momento todos los ganadores comenzaron a ser redes convolucionales profundas
- En gran parte desató el furor por este tipo de redes en el campo de visión por computador
- Paper original: <https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>
- Aproximadamente 60,000,000 de parámetros

Características de AlexNet

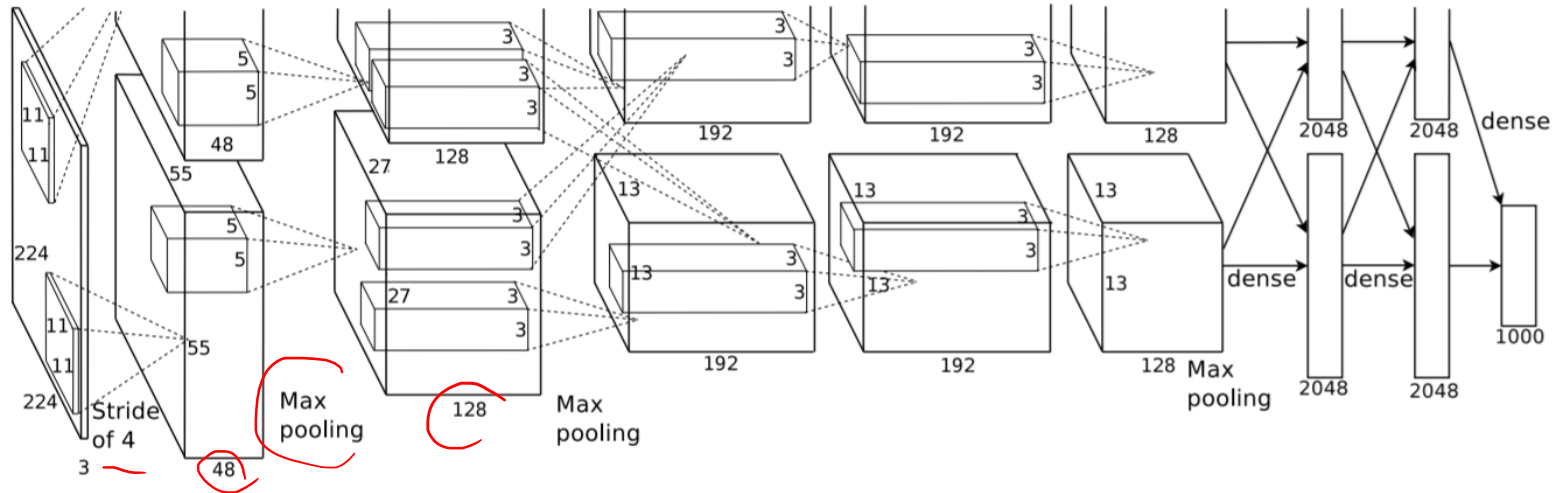
- Rectified Linear Units (relu)
- Uso de múltiples GPUs para implementar el modelo
- Dropout
- Local Response Normalization, no tan usado hoy en día
- Capas Pool con ventanas superpuestas
- Data Augmentation



Dropout

- En cada etapa de entrenamiento, nodos individuales son 'dropped' o 'caídos' o ignorados con probabilidad $1-p$ y retenidos con una probabilidad p
- De esa manera queda una red con menos nodos/conexiones entre nodos
- Ayuda a evitar overfitting
- En la etapa de testing todos los pesos se ajustan por el factor p

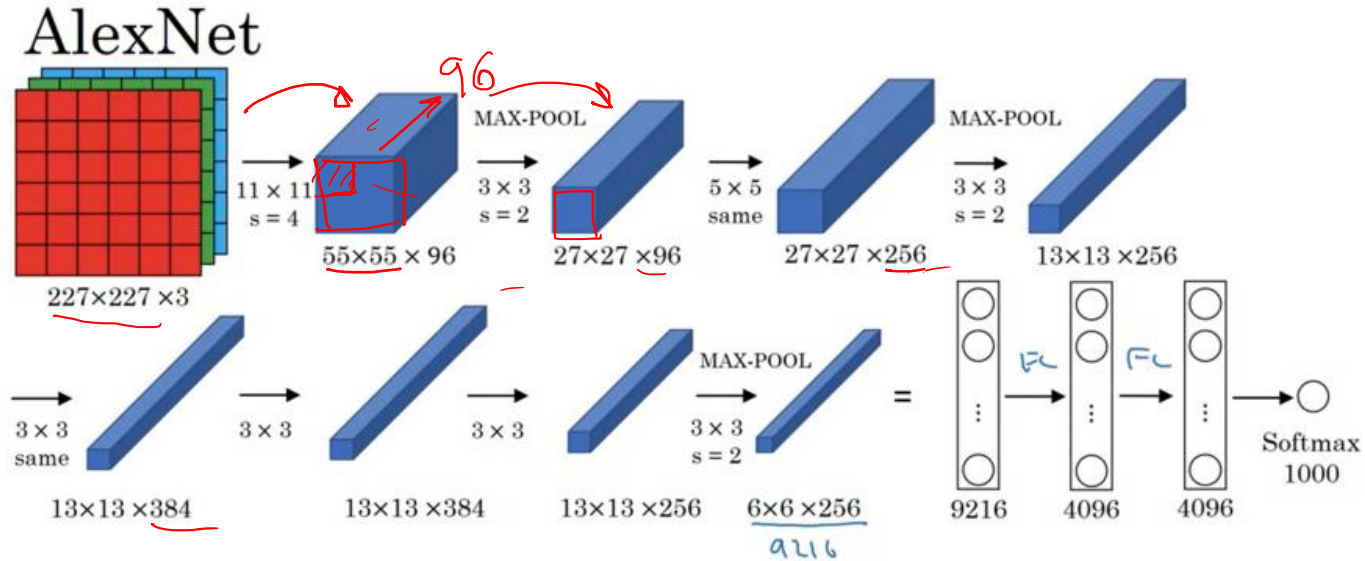
Estructura en el paper original



Estructura:

Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	227x227x3	-	-	-
1	Convolution	96	55 x 55 x 96	11x11	4	relu
	Max Pooling	96	27 x 27 x 96	3x3	2	relu
2	Convolution	256	27 x 27 x 256	5x5	1	relu
	Max Pooling	256	13 x 13 x 256	3x3	2	relu
3	Convolution	384	13 x 13 x 384	3x3	1	relu
4	Convolution	384	13 x 13 x 384	3x3	1	relu
5	Convolution	256	13 x 13 x 256	3x3	1	relu
	Max Pooling	256	6 x 6 x 256	3x3	2	relu
6	FC	-	9216	-	-	relu
7	FC	-	4096	-	-	relu
8	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax

Esquema





Kahoot de redes convolucionales 2



Tarea

- En pool vs no pool agregar uno o dos bloques más con y sin pool para ver si hay diferencia.
- Entrenar la red para perros y gatos y ver qué accuracy pueden obtener, *con data augmentation*
- Implementar LeNet-5 y utilizarla en MNIST *dígitos*



Bonus

- Leer el capítulo de redes convolucionales del libro de Goodfellow et al.
<https://www.deeplearningbook.org/contents/convnets.html>
- otro machete: <https://github.com/afshinea/stanford-cs-230-deep-learning/blob/master/en/cheatsheet-convolutional-neural-networks.pdf>