

Indexación, búsqueda y análisis en repositorios multimedia: imagen-video

Juan Carlos San Miguel

Agenda: Multimedia (imagen, video)

- **Redes convolucionales: entrenamiento**
 - Introducción
 - Conceptos básicos
 - Conceptos avanzados
 - Caso de estudio

Agenda: Multimedia (imagen, video)

➤ **Redes convolucionales: entrenamiento**

- Introducción
- Conceptos básicos
- Conceptos avanzados
- Caso de estudio

INTRODUCCION

¿chihuahua o muffin?



https://retina.elpais.com/retina/2018/01/25/tendencias/1516886113_207256.html

INTRODUCCION

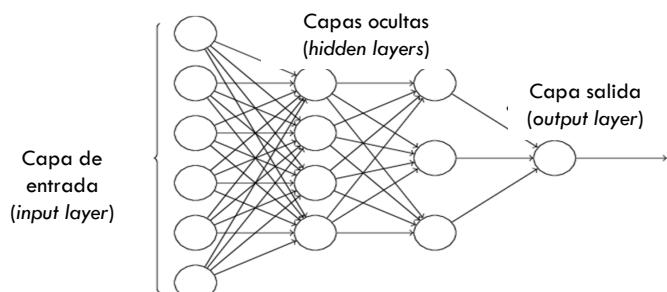
- El entrenamiento de redes es complejo...



INTRODUCCION

- Parámetros versus hiperparámetros:
 - Parámetros: variables cuyo valor se obtiene tras el entrenamiento
 - Hiperparámetros: variables cuyo valor se fija antes de entrenar

Ejemplo: perceptrón multicapa

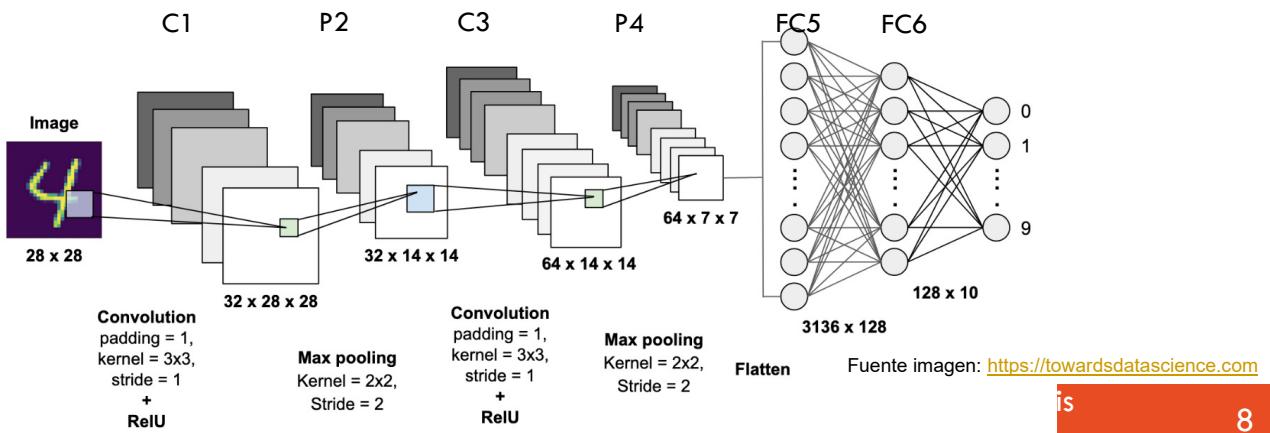


- Sea la red mostrada en la imagen, calcule el número de parámetros de la primera capa convolucional cuya salida es el volumen de datos C1 ($32 \times 28 \times 28$). Seleccione una opción:

a) 320

b) 18496

c) 18816



Agenda: Multimedia (imagen, video)

- **Redes convolucionales: entrenamiento**
 - Introducción
 - Conceptos básicos
 - Optimización:
 - Función de coste (loss function)
 - Descenso por gradiente (gradient descent)
 - Retropropagación (backpropagation)
 - Procesado por lotes (batch, iteration, epoch)
 - Regularización (regularization)
 - Evaluación con curvas de pérdidas (loss) y rendimiento (accuracy)
 - Conceptos avanzados
 - Caso de estudio

CONCEPTOS BASICOS: OPTIMIZACION

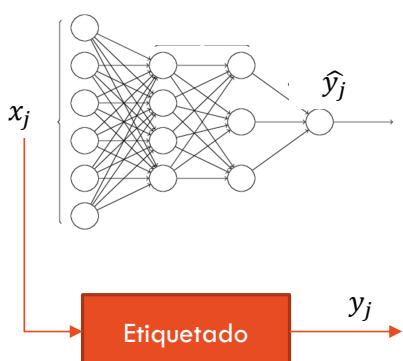
- Permite obtener el valor óptimo de los parámetros θ
- Consiste minimizar una función objetivo $L(x_j, y_j; \theta)$
- Emplea el conjunto de entrenamiento con datos etiquetados

$$\theta^* = \operatorname{argmin}_{\theta} \sum_{j=1}^N L(x_j, y_j; \theta)$$

- Elementos básicos:
 - Función de coste/pérdidas
 - Estrategia de búsqueda

CONCEPTOS BASICOS: OPTIMIZACION

- Función de coste $L(x_j, y_j; \theta)$
 - Comparación del resultado obtenido \hat{y}_j (e.g. por la red neuronal) con el resultado esperado y_j (i.e. etiqueta o *ground-truth*)
 - Algoritmos con salida \hat{y}_j única (i.e. escalar)



- Entropía cruzada
(*Cross-entropy error*)

$$L_{CE}(\theta) = -\frac{1}{N} \sum_{j=1}^N \hat{y}_j \log(y_j) + (1 - \hat{y}_j) \log(1 - y_j)$$

- Error cuadrático medio
(*Mean least-square error*)

$$L_{MSE}(\theta) = \frac{1}{N} \sum_{j=1}^N (\hat{y}_j - y_j)^2$$

CONCEPTOS BASICOS: OPTIMIZACION

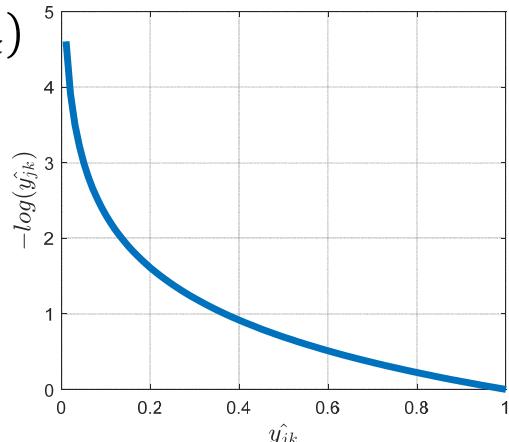
- Función de coste $L(x_j, y_j; \theta)$
- Algoritmos con salida \hat{y}_j múltiple (i.e. vector con k elementos \hat{y}_{jk})

$$L(x_j, y_j; \theta) = -\frac{1}{N} \sum_{j=1}^N \sum_{c=1}^P y_{jk} \log(\hat{y}_{jk})$$

y_{jk} : probabilidad etiquetada para clase k con el dato x_j

\hat{y}_{jk} : Probabilidad predicha para clase k con el dato x_j

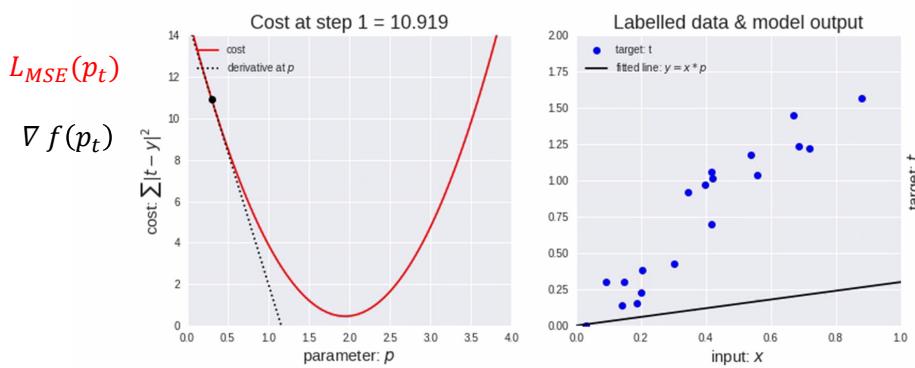
$\log(\hat{y}_{jk}) \rightarrow$ Función derivable y decrece monótonamente si aumenta \hat{y}_{jk}



CONCEPTOS BASICOS: OPTIMIZACION

- Estrategia de búsqueda: descenso por gradiente

$$p_{t+1} = p_t - \gamma \nabla f(p_t)$$



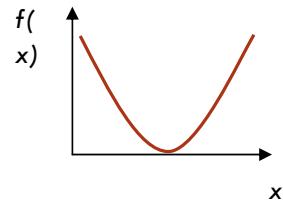
Función a optimizar
 $f(x, p) = x \cdot p$

Datos (x_j, y_j)

<https://youtu.be/noMAILVZqrE>

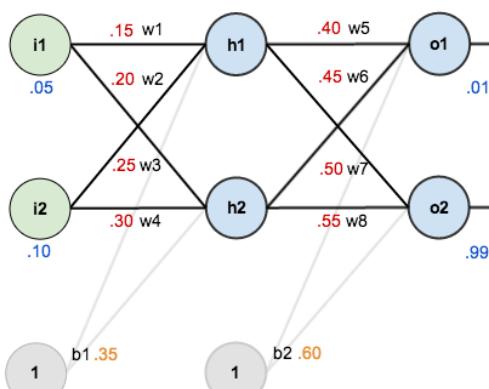
CONCEPTOS BASICOS: OPTIMIZACION

- Estrategia de búsqueda: descenso por gradiente
 - Optimizador por excelencia utilizado en aprendizaje automático
 - No es claramente aplicable para funciones no derivables
 - Menor coste computacional frente al ajuste por mínimos cuadrados
 - Convergencia
 - Garantizada para funciones convexas
 - No garantizada para funciones no-convexas
 - Si la función no es convexa, búsqueda de mínimos locales
- Utilizada también para funciones de múltiples variables
 - Necesario el cálculo de derivadas parciales (“Jacobiano”)



CONCEPTOS BASICOS: OPTIMIZACION

- Retropropagación (*backpropagation*)
 - Actualización de parámetros con descenso por gradiente
 - Dos etapas:
 - *Forward pass*: ejecución de la red para obtener la salida en “t”
 - *Backward pass*: calculo de derivadas parciales para actualizar θ_{t+1}



$$\omega_5^{t+1} = \omega_5^t - \eta \frac{\delta L}{\delta \omega_5}$$

Regla cadena

$$\frac{\delta L}{\delta \omega_5} = \frac{\delta L}{\delta out_{o_1}} \cdot \frac{\delta out_{o_1}}{\delta net_{o_1}} \cdot \frac{\delta net_{o_1}}{\delta \omega_5}$$

$$net_{o_1} = \omega_5 \cdot out_{h1} + \omega_6 \cdot out_{h2} + b_2$$

$$out_{o_1} = \frac{1}{1 + e^{-net_{o_1}}}$$

$$L = \sum_{j=1}^2 (out_{o_j} - true_j)^2$$

CONCEPTOS BASICOS: OPTIMIZACION

- Descenso por Gradiente y Retropropagación: resumen



<https://youtu.be/odlgtjXduVg>

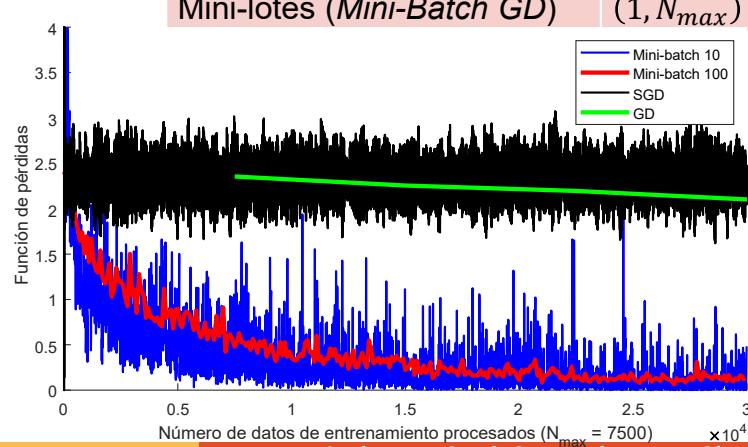
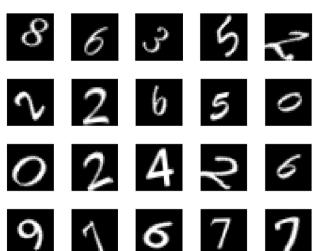
CONCEPTOS BASICOS: LOTES

- Procesado por lotes basado en descenso por gradiente

$$\theta^* = \operatorname{argmin}_{\theta} \sum_{j=1}^N L(x_j, y_j; \theta)$$

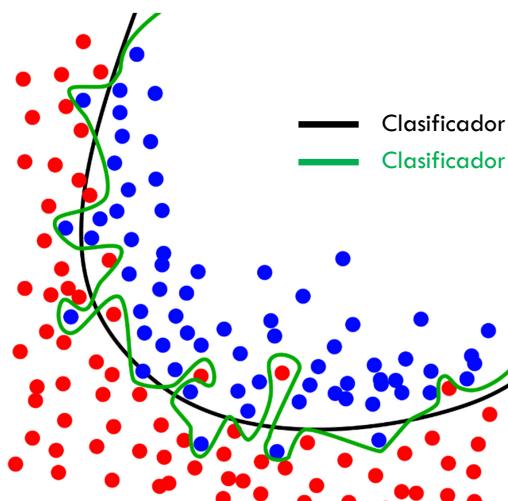
Aproximaciones	N
En lote (Batch o GD)	N_{max}
Estocástico (SGD)	1
Mini-lotes (Mini-Batch GD)	$(1, N_{max})$

Ejemplo: reconocimiento de dígitos escritos (0-9)



CONCEPTOS BASICOS: REGULARIZACION

- El objetivo es prevenir sobreajuste (*overfitting*)



Solución posible:
evitar aprender parámetros
con valores altos



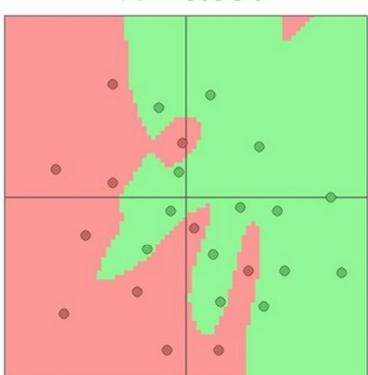
Modificar la función de pérdidas

$$L(\theta) = \frac{1}{N} \sum_{i=1}^N (\hat{y}_i - y_i)^2 + \frac{\lambda}{2} \sum_{\omega_j} |\omega_j|^2$$

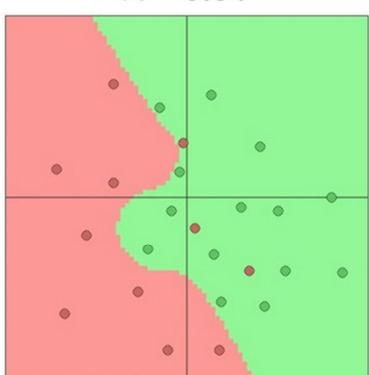
CONCEPTOS BASICOS: REGULARIZACION

$$L(\theta) = \frac{1}{N} \sum_{i=1}^N (\hat{y}_i - y_i)^2 + \frac{\lambda}{2} \sum_{\omega_j} |\omega_j|^2$$

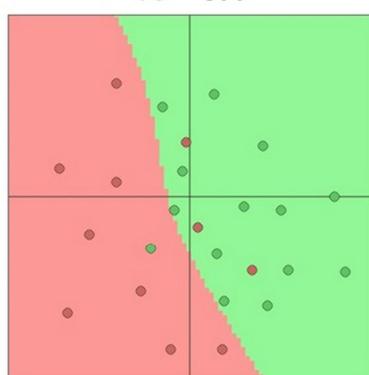
$$\lambda = 0.001$$



$$\lambda = 0.01$$



$$\lambda = 0.1$$



Ejemplos obtenidos con la herramienta

<https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html>

CONCEPTOS BASICOS: EVALUACION

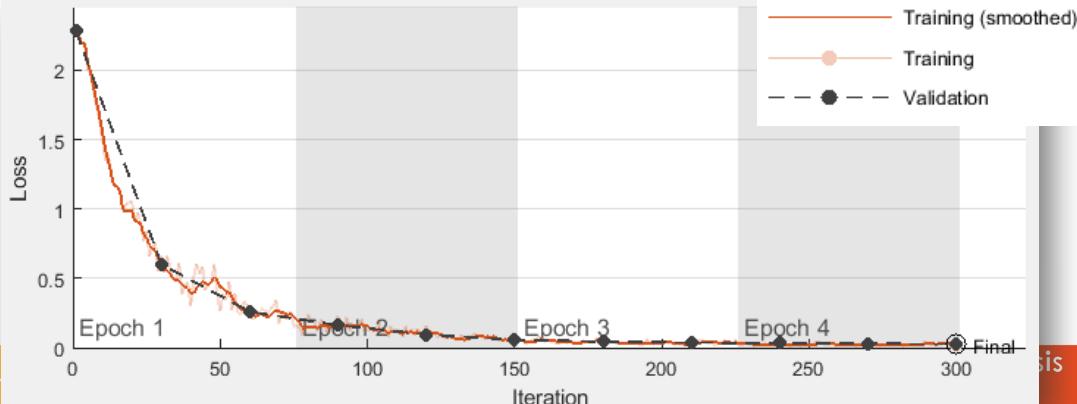
➤ Curvas de pérdidas (loss)

➤ Evolución de la función de pérdidas en el proceso de entrenamiento

➤ Época (epoch): análisis de todos los datos de entrenamiento

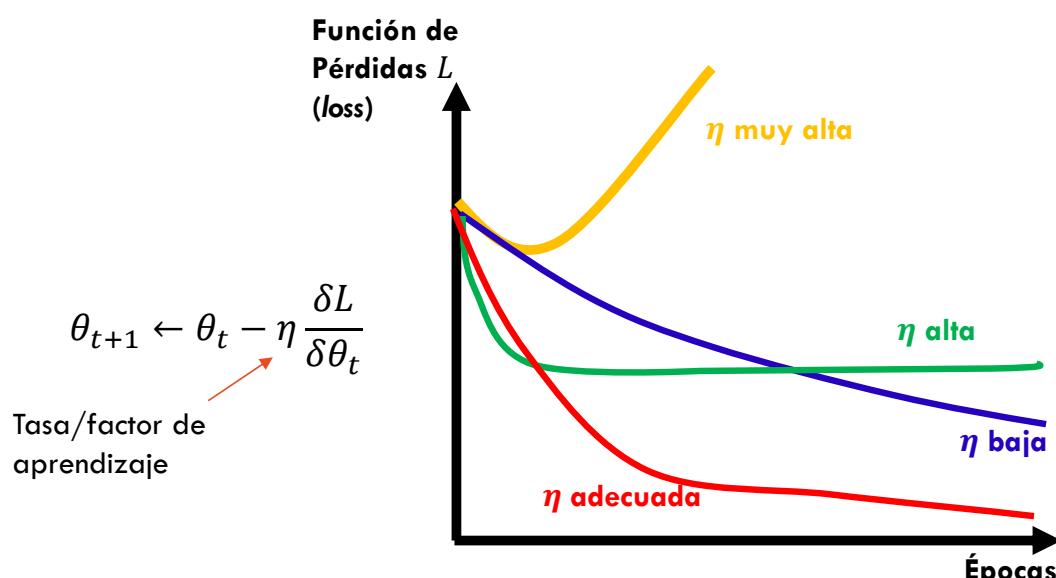
➤ Tamaño del lote (Batch size): número de datos procesados a la vez

➤ Iteraciones (iteration): número de pasadas sobre los datos, cada una con un número de datos de entrenamiento igual a batch size

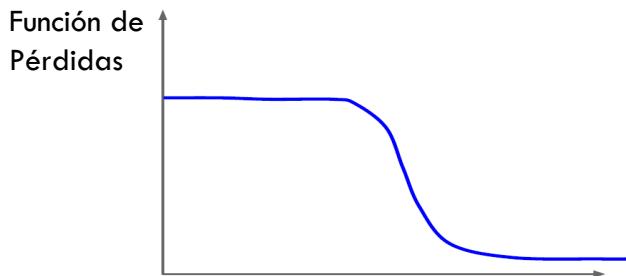


CONCEPTOS BASICOS: EVALUACION

➤ Curva de pérdidas (loss)



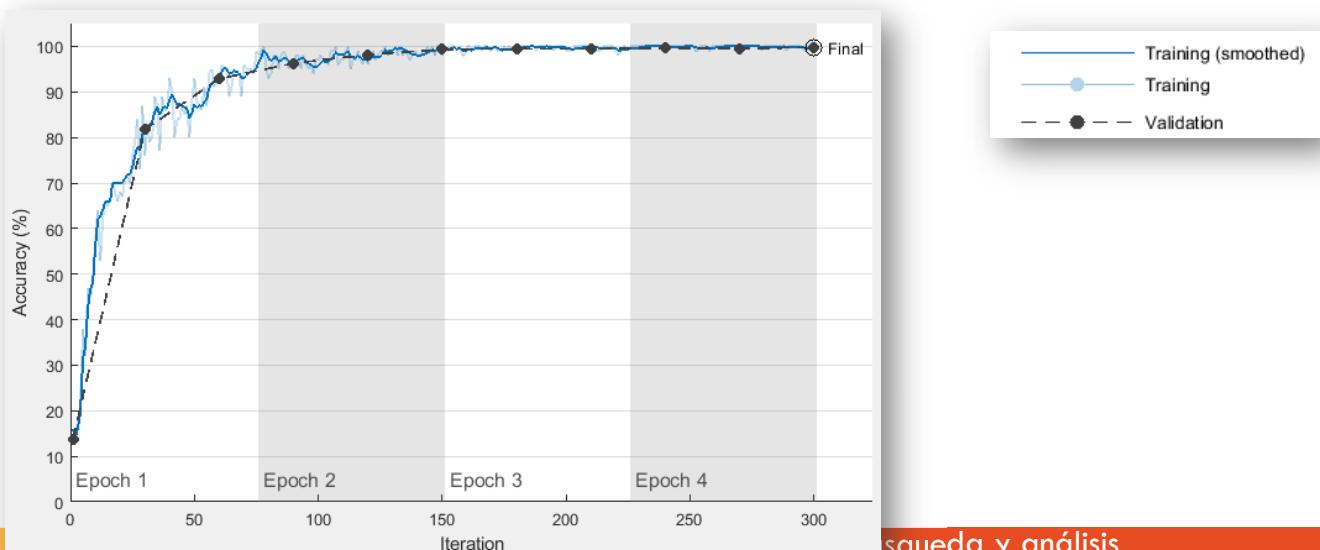
- Tras entrenar una red neuronal, se obtiene lo siguiente:



- Indique cuales de los siguientes motivos pueden ser las razones de la observación en la gráfica:
- La tasa de aprendizaje es muy baja
 - Se está produciendo un sobreajuste (*overfitting*)
 - La red no está aprendiendo nada sobre los datos entrenamiento
 - El proceso de optimización está en un mínimo global

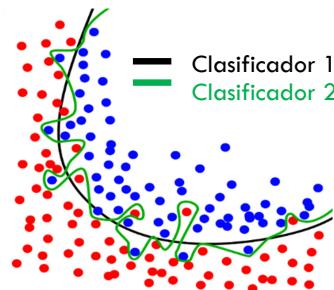
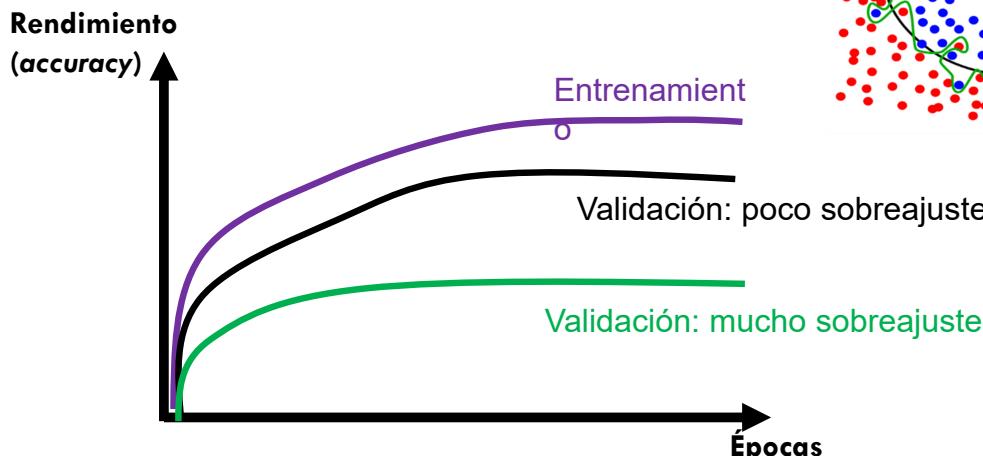
CONCEPTOS BASICOS: EVALUACION

- Curva de rendimiento (accuracy)
- $$accuracy = \frac{\#aciertos}{\#total\ datos}$$
- Evolución del rendimiento en el proceso de entrenamiento



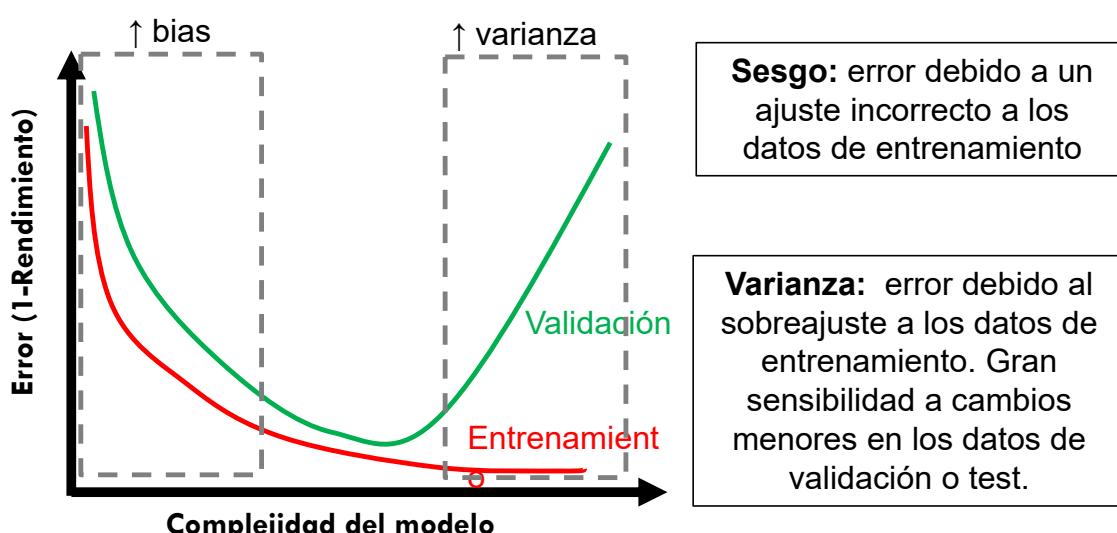
CONCEPTOS BASICOS: EVALUACION

➤ Curva de rendimiento (accuracy)

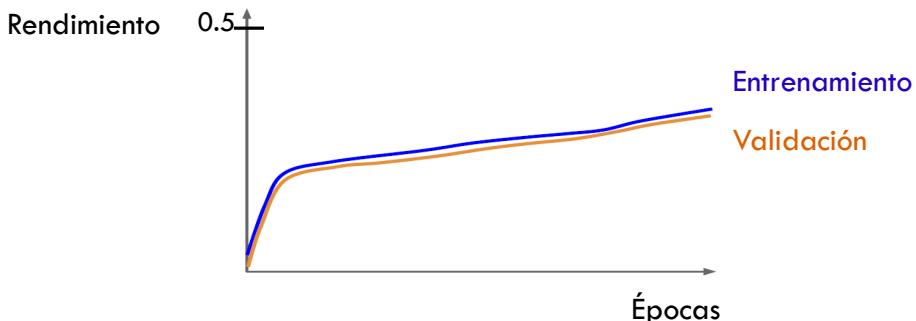


CONCEPTOS BASICOS: EVALUACION

➤ Balance entre sesgo (bias) y varianza (variance)



- Tras entrenar una red neuronal, se obtiene lo siguiente:

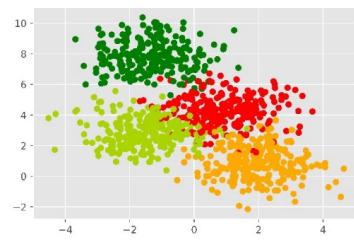


- Indique cuales de los siguientes motivos pueden ser las razones de la observación en la gráfica:
 - a) La tasa de aprendizaje es muy baja
 - b) Se está produciendo un sobreajuste (*overfitting*)
 - c) El factor de regularización es muy alto

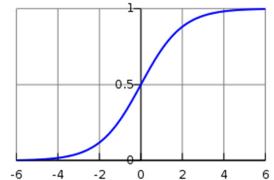
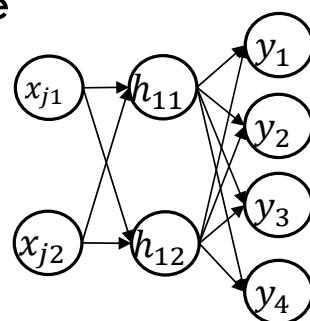
Agenda: Multimedia (imagen, video)

- **Redes convolucionales: entrenamiento**
 - Introducción
 - Conceptos básicos
 - Conceptos avanzados
 - Inicialización de parámetros (*weight initialization*)
 - Normalización del lote (*batch normalization*)
 - Aumentación de datos (*data augmentation*)
 - *Dropout*
 - Búsqueda de Hiperparámetros
 - Caso de estudio

- Sea el problema de clasificación multiclase con K=4 mostrado en la imagen de la derecha

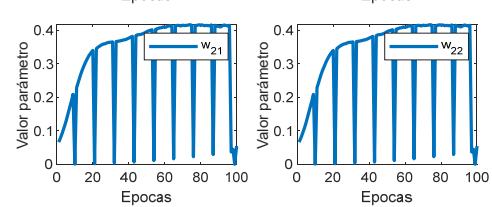
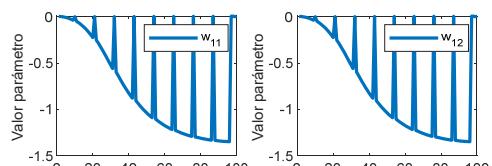
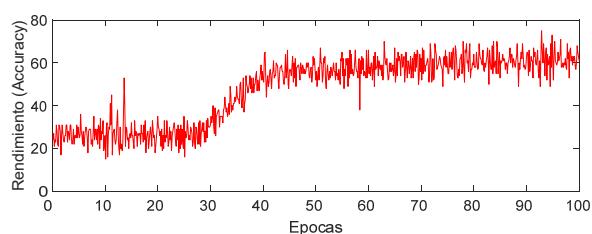
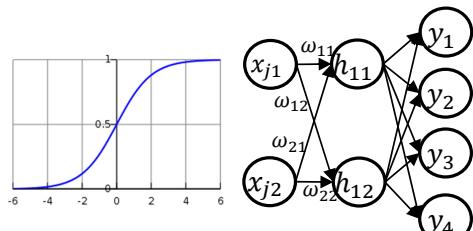
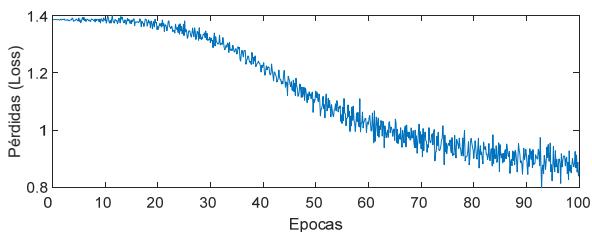


- A continuación se define la siguiente red para resolver este problema
¿Qué ocurre si iniciamos todos los parámetros de la red con un valor constante?
(asuma activación sigmoide)



INICIALIZACION PARAMETROS

- Ejemplo para activación sigmoide
 - Inicialización parámetros con $\omega_i = 0$



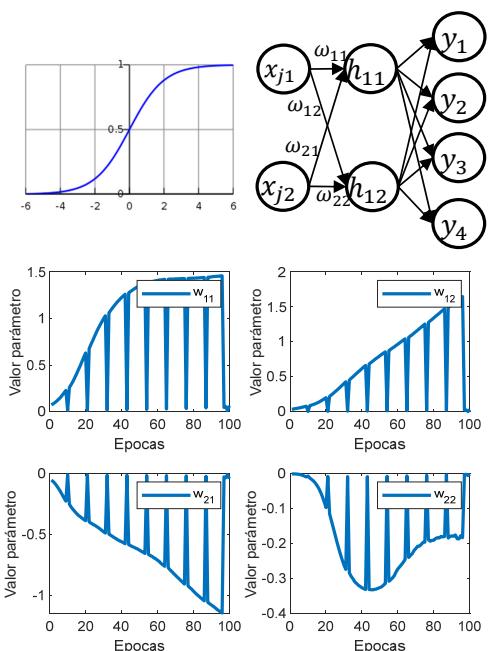
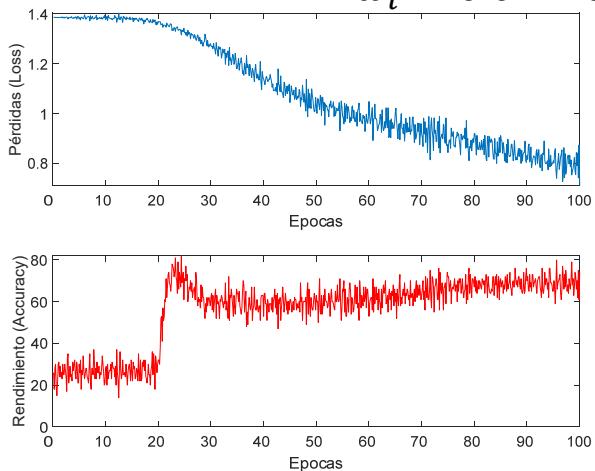
Rendimiento/accuracy (entrenamiento/test): 0.61/0.56

INICIALIZACION PARAMETROS

➤ Ejemplo para activación sigmoide

- Inicialización parámetros con

$$\omega_i = 0.01 \cdot \text{rand}(0,1)$$



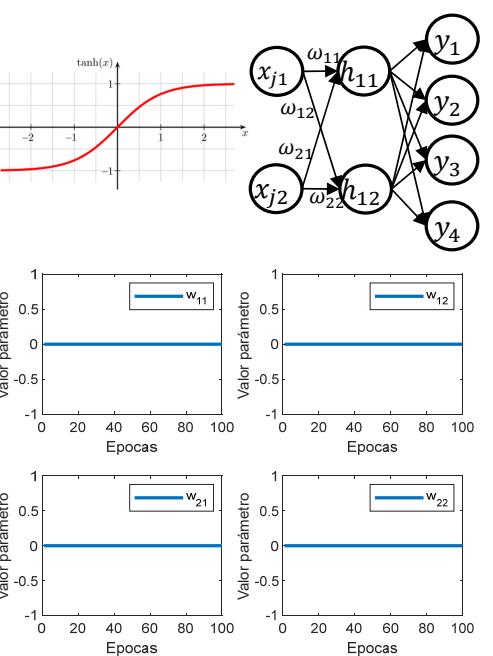
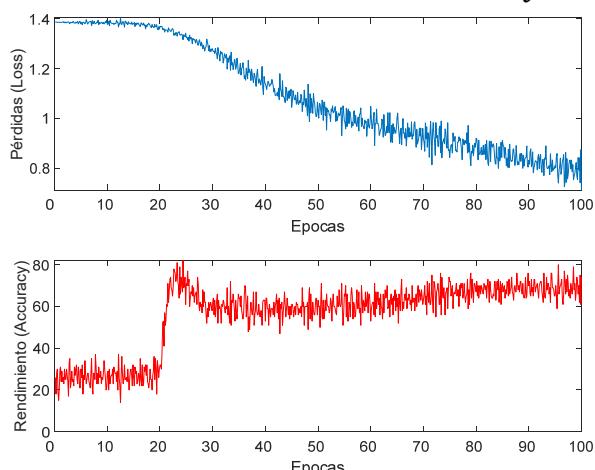
Rendimiento/accuracy (entrenamiento/test): 0.74/0.61

INICIALIZACION PARAMETROS

➤ Ejemplo para activación tanh

- Inicialización parámetros con

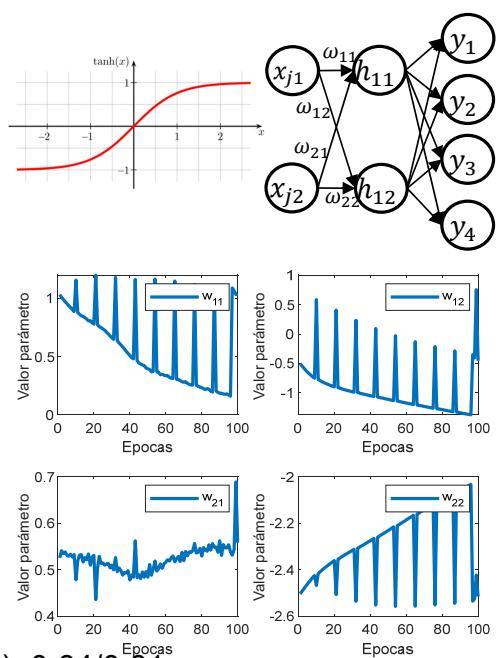
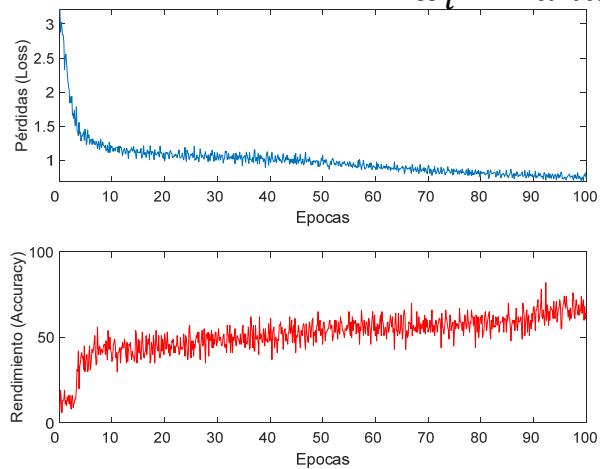
$$\omega_i = 0$$



Rendimiento/accuracy (entrenamiento/test): 0.28/0.19

INICIALIZACION PARAMETROS

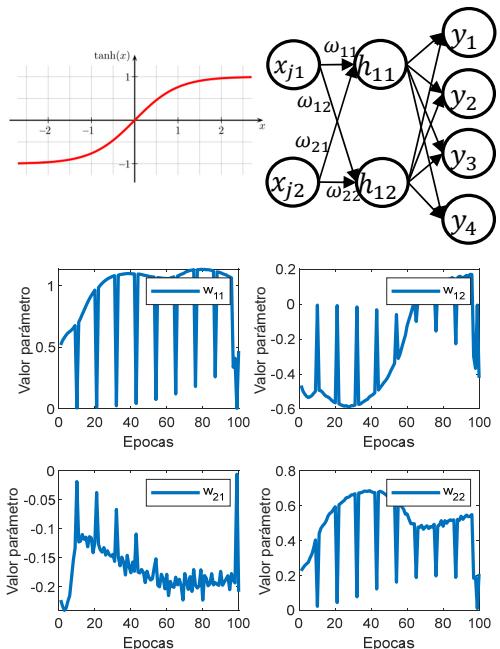
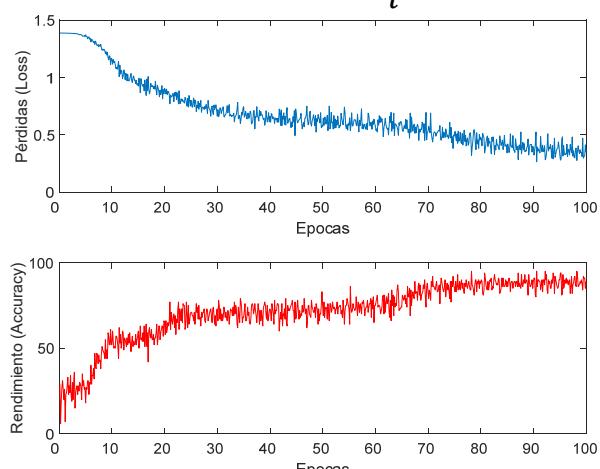
- Ejemplo para activación \tanh
- Inicialización parámetros con $\omega_i = \text{rand}(0,1)$



Rendimiento/accuracy (entrenamiento/test): 0.64/0.61

INICIALIZACION PARAMETROS

- Ejemplo para activación \tanh
- Inicialización parámetros con $\omega_i = 0.01 \cdot \text{rand}(0,1)$



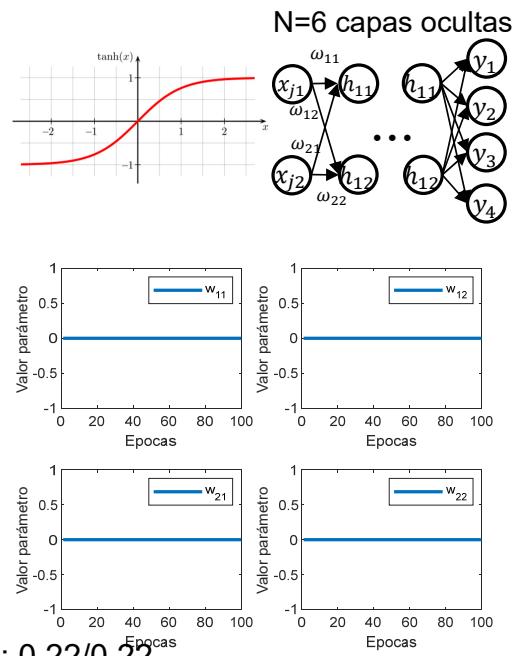
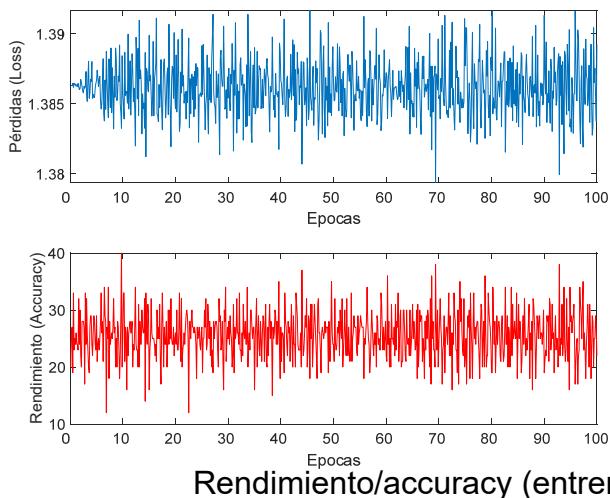
Rendimiento/accuracy (entrenamiento/test): 0.89/0.89

INICIALIZACION PARAMETROS

➤ Ejemplo para activación \tanh

- Inicialización parámetros con

$$\omega_i = 0.01 \cdot \text{rand}(0,1)$$



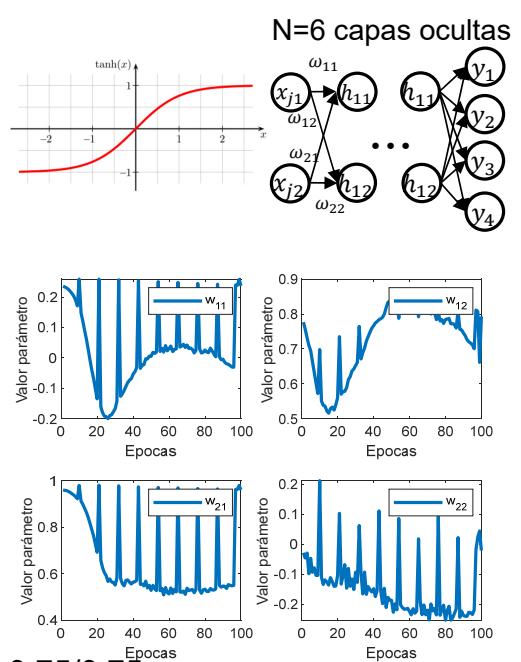
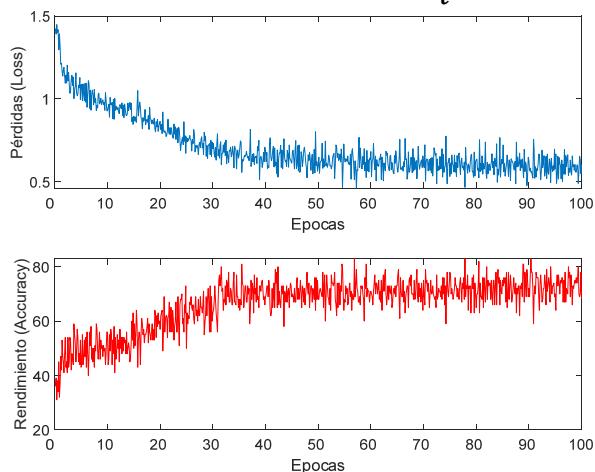
Rendimiento/accuracy (entrenamiento/test): 0.22/0.22

INICIALIZACION PARAMETROS

➤ Ejemplo para activación \tanh

- Inicialización parámetros con

$$\omega_i = \text{rand}(0,1)$$



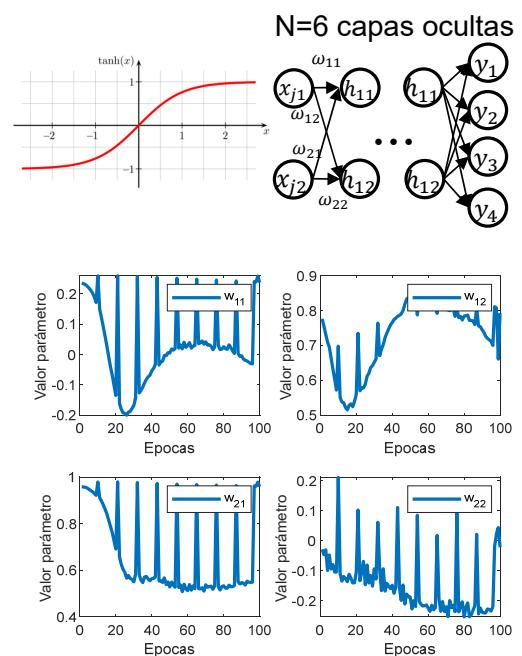
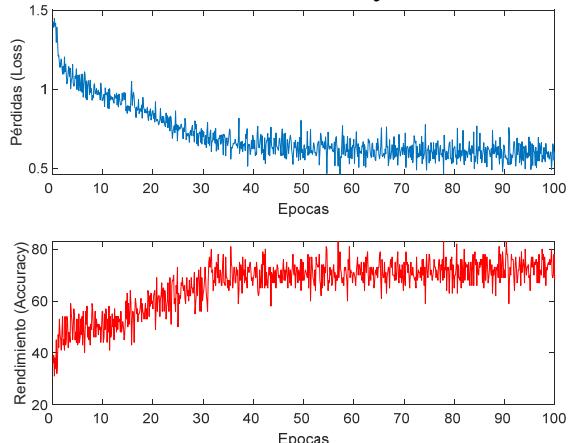
Rendimiento/accuracy (entrenamiento/test): 0.75/0.75

INICIALIZACION PARAMETROS

➤ Ejemplo para activación *tanh*

➤ Inicialización parámetros con

Xavier initialization
[Gorot, AISTATS10] $\omega_i = \text{rand}(0,1) \cdot \frac{1}{\sqrt{n}}$

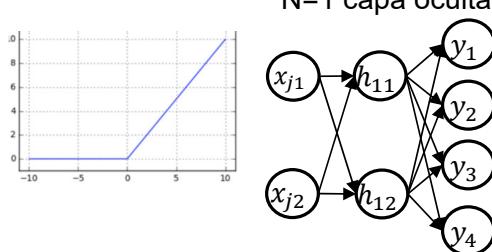
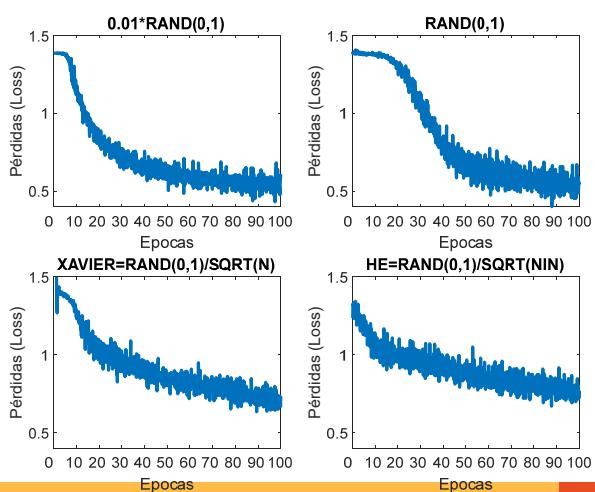


Rendimiento/accuracy (entrenamiento/test): 0.93/0.89

INICIALIZACION PARAMETROS

➤ Ejemplo para activación *ReLU*

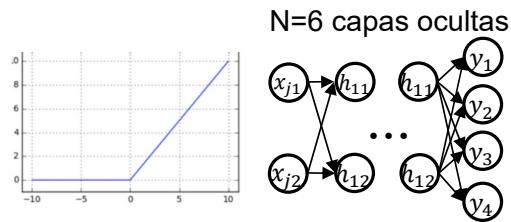
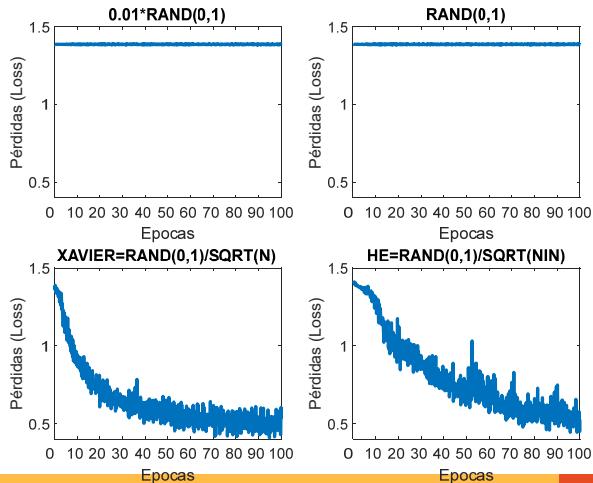
➤ Inicialización parámetros



Inicialización	Rendimiento (accuracy)	
	Train	Test
Ceros	0.26	0.21
0.01·rand(0,1)	0.66	0.65
rand(0,1)	0.80	0.78
Xavier	0.74	0.75
He	0.74	0.69

INICIALIZACION PARAMETROS

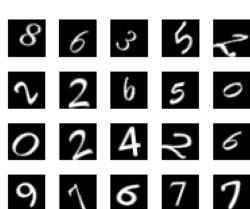
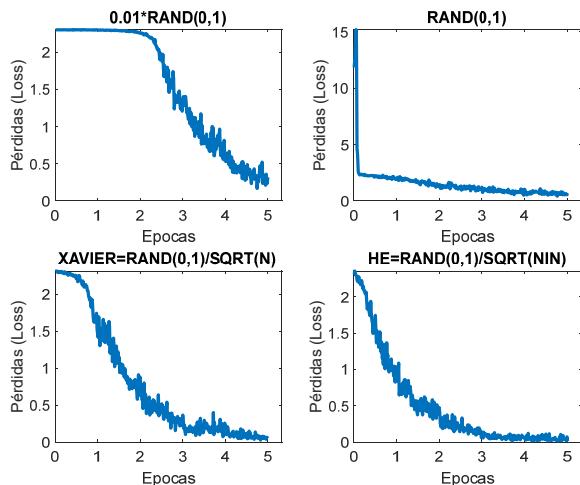
- Ejemplo para activación ReLU
 - Inicialización parámetros



Inicialización	Rendimiento (accuracy)	
	Train	Test
Ceros	0.25	0.23
0.01·rand(0,1)	0.26	0.23
rand(0,1)	0.26	0.23
Xavier	0.78	0.75
He	0.70	0.69

INICIALIZACION PARAMETROS

- Ejemplo para activación ReLU con CNN
 - Inicialización parámetros sólo en capas convolucionales



LeNet [LeCun, 1998]
 1x Conv2D 5x5 (6)
 1x maxPooling (/2)
 1x Conv2D 5x5 (16)
 1x maxPooling (/2)
 1x fullyConn (120)
 1x fullyConn (84)
 1x fullyConn (10)

Inicialización	Rendimiento (accuracy)	
	Train	Test
0.01·rand(0,1)	0.89	0.91
rand(0,1)	0.75	0.80
Xavier	0.99	0.97
He	0.99	0.99

- Se desea reconocer dígitos con imágenes similares a los ejemplos a su derecha
- Para ello se diseña una pequeña red CNN con la arquitectura mostrada a su derecha
- ¿Qué inicialización utilizaría para una activación no lineal tipo ReLU aplicada tras cada capa convolucionales?
 - a) 0
 - b) rand(0,1)
 - c) $rand(0,1) \cdot \frac{1}{\sqrt{n}}$

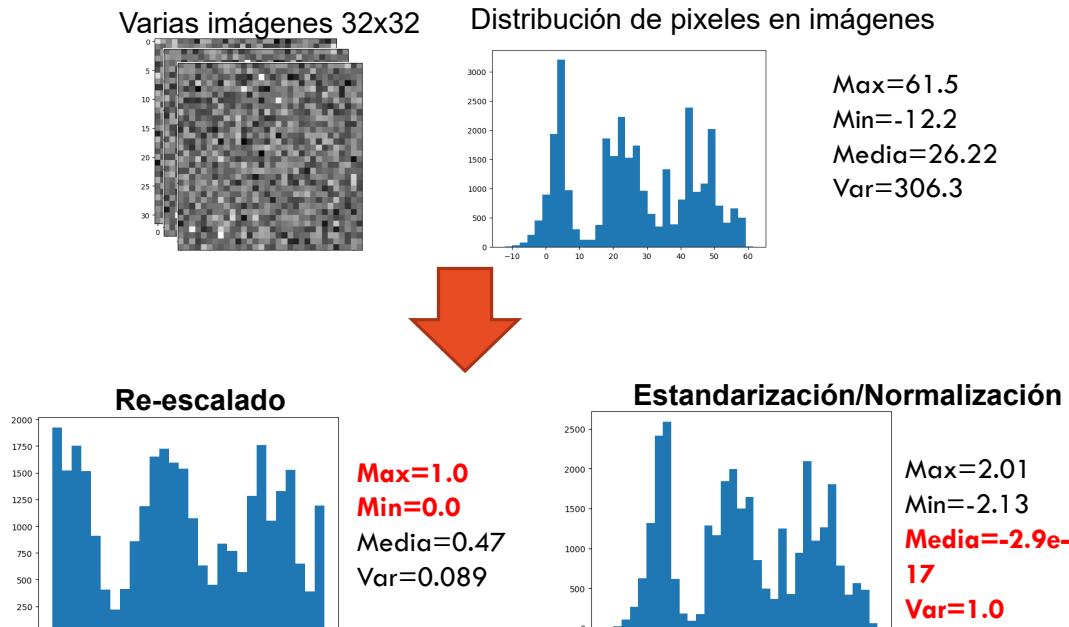
8	6	3	5	→
2	6	5	0	
0	2	4	2	6
9	1	6	7	7

1x Conv2D 3x3 (8)
 1x maxPooling (/2)
 1x Conv2D 3x3 (16)
 1x maxPooling (/2)
 1x Conv2D 3x3 (32)
 1x fullyConn(10)

INICIALIZACION PARAMETROS

- Recomendaciones
 - Capas convolucionales
 - Activación ReLU
 - Inicialización método He/Kaiming
 - Capas completamente conectadas (fullyConnected)
 - Activación ReLU/tanh
 - Inicialización método Xavier/Glorot

BATCH NORMALIZATION

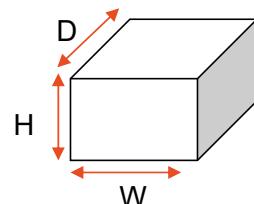


BATCH NORMALIZATION

- Volumen de datos $x: H \times W \times D$
 - Elementos del volumen $x_{j,i,d}$
 - Re-escalado y normalización se aplican a cada “canal” (3^a dimensión)
- Re-escalado: $\widetilde{x_{j,i,d}} = \frac{x_{j,i,d} - \min(x_{j,i,d})}{\max(x_{j,i,d}) - \min(x_{j,i,d})}, \forall j, i$
- Estandarización/Normalización:

$$\widetilde{x_{j,i,d}} = \frac{x_{j,i,d} - \mu_d}{\sqrt{\sigma_d^2 + \epsilon}}, \forall j, i \quad \sigma_d^2 = \frac{1}{H \times W} \sum_{j=1}^H \sum_{i=1}^W (x_{j,i,d} - \mu_d)^2$$

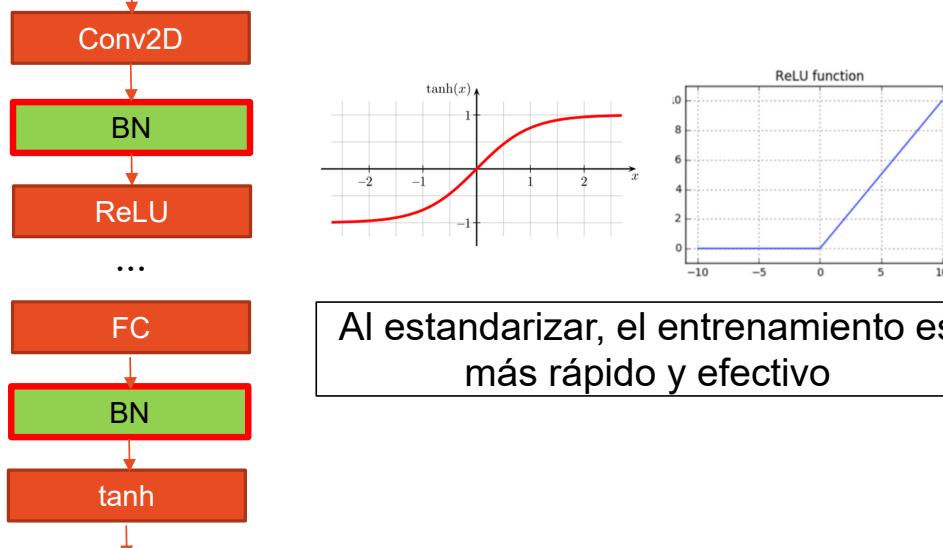
$$\widetilde{x_{j,i,d}} = \gamma_d x_{j,i,d} + \beta_d, \forall j, i$$



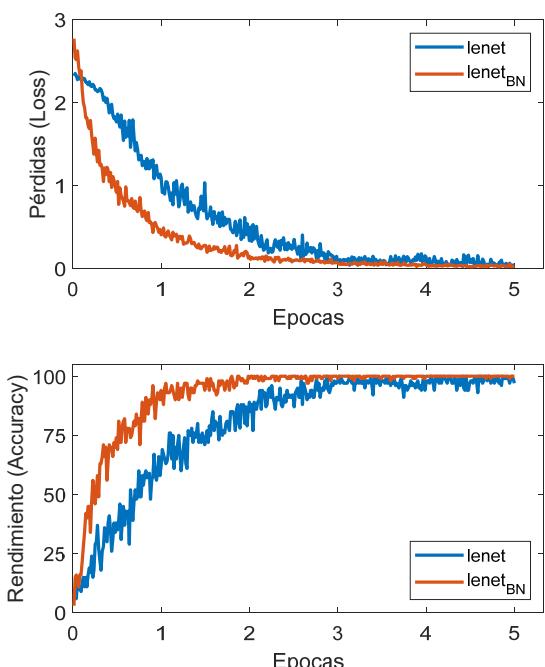
$$\mu_d = \frac{1}{H \times W} \sum_{j=1}^H \sum_{i=1}^W x_{j,i,d}$$

BATCH NORMALIZATION

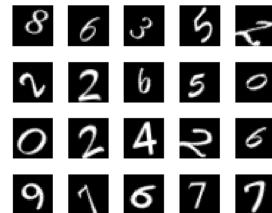
- Batch Normalization se aplica a los datos de entrada de la red y antes de las funciones de activación



BATCH NORMALIZATION



Accuracy Test: Lenet=0.98 Lenet_{BN}=0.99



LeNet [LeCun, 1998]

1x Conv2D 5x5 (6)
1x
BatchNorm+ReLU
1x maxPooling (/2)
1x Conv2D 5x5 (16)
1x
BatchNorm+ReLU
1x maxPooling (/2)
1x fullyConn (120)
1x
BatchNorm+ReLU
1x fullyConn (84)
1x
BatchNorm+ReLU
1x fullyConn (10)
1x softmax

LeNet_BN

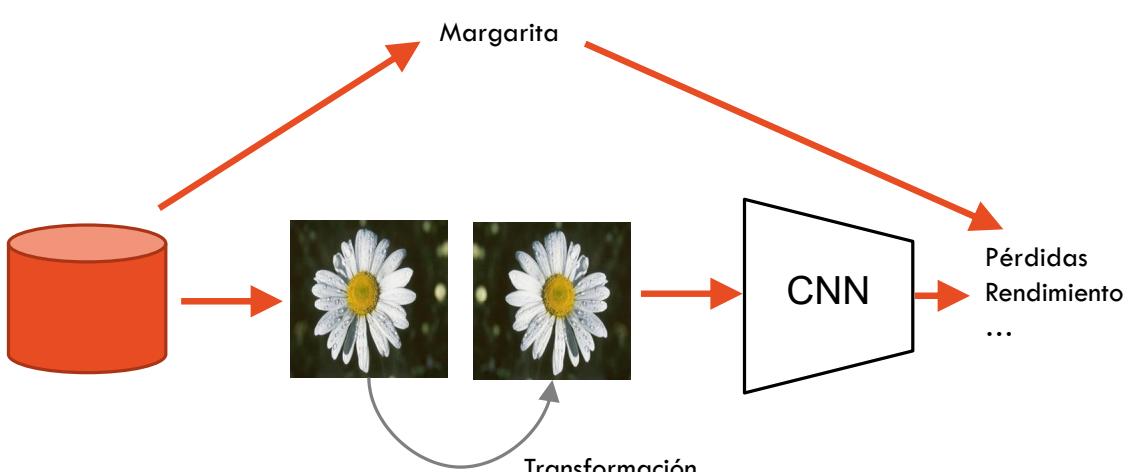
1x Conv2D 5x5 (6)
1x
BatchNorm+ReLU
1x maxPooling (/2)
1x Conv2D 5x5 (16)
1x
BatchNorm+ReLU
1x maxPooling (/2)
1x fullyConn (120)
1x
BatchNorm+ReLU
1x fullyConn (84)
1x
BatchNorm+ReLU
1x fullyConn (10)
1x softmax

- ¿Por qué es necesaria la etapa de normalización conocida como “batch normalization”?

- Seleccione la respuesta correcta
 - (a) Hace que la inicialización de los parámetros sea más rápida
 - (b) Normalización es otra forma de regularización... ayuda a reducir la variabilidad de los datos.
 - (c) Hace que la función de coste converja más rápidamente y el proceso de entrenamiento sea más rápido
 - (d) Facilita la visualización de los datos

DATA AUGMENTATION

- Transformación aleatoria de los datos utilizados durante el proceso de entrenamiento



DATA AUGMENTATION

➤ Transformaciones típicas:

Imágenes originales



Imágenes invertidas en ambas direcciones (*flip*)



Imágenes desplazadas en ambas direcciones (*jitter*)



Imágenes rotadas en ambas direcciones (*rotation*)



DATA AUGMENTATION

➤ Transformaciones típicas:

➤ *Cropping*

1. Seleccionar una región
2. Escalar la región seleccionada al tamaño necesario para la red



DATA AUGMENTATION

Dataset Flowers:

<http://bit.ly/37AIOKd>

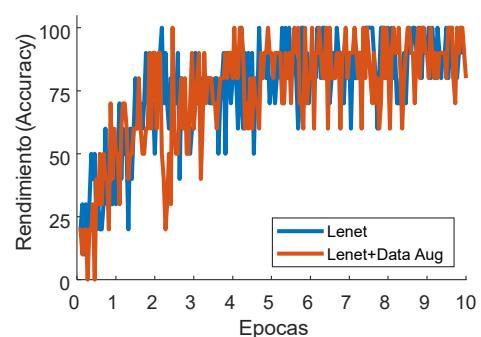
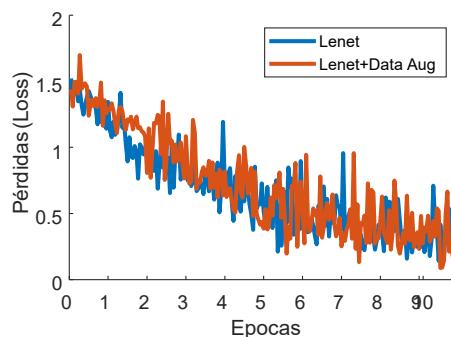
Subconjunto con 320 imágenes y 4 clases



Data Augmentation

- Rotación (-20,20)
- Jitter X (-5,5)
- Jitter Y (-5,5)

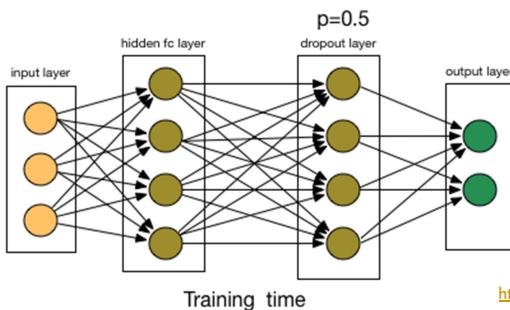
LeNet [LeCun, 1998]
1x Conv2D 5x5 (6)
1x ReLU
1x maxPooling (/2)
1x Conv2D 5x5 (16)
1x ReLU
1x maxPooling (/2)
1x fullyConn (120)
1x ReLU
1x fullyConn (84)
1x ReLU
1x fullyConn (10)
1x softmax



Accuracy Test: Lenet=0.79; Test LeNet+Data Augmentation=0.82

DROPOUT

- En cada *iteración* de entrenamiento, aleatoriamente se anula un sub-conjunto de parámetros en una capa
- El efecto es que las neuronas se vuelven menos dependientes de la salida de las neuronas que están conectadas a cada una
- Se aprenden características más robustas que sean útiles para más subconjuntos



<https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf>

DROPOUT

MNIST dataset 60K/ 10K train/test

0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7 7 7 7 7 7
8 8 8 8 8 8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9 9 9 9 9

Table 1. Classification error on MNIST

method	classification error (%)	
	average of 5 models	ensemble of 5 models
Baseline (without dropout)	0.604 ± 0.0829	0.57
Dropout ($p = 0.2$)	0.430 ± 0.0212	0.38
Spatial dropout ($p = 0.1$)	0.504 ± 0.0493	0.42
Feature-wise max-drop ($p = 0.2$)	0.488 ± 0.0657	0.42
Channel-wise max-drop ($p = 0.5$)	0.502 ± 0.0148	0.40
Stochastic dropout ($N(0.2, 0.05)$)	0.410 ± 0.0122	0.38
Stochastic dropout ($U(0.1, 0.3)$)	0.448 ± 0.0363	0.42

CIFAR-10 dataset 50K/10K train/test



Table 3. Classification error on CIFAR-10 dataset.

method	classification error (%)
Baseline	16.84%
Dropout ($p = 0.1$)	12.22%
Spatial Dropout ($p = 0.05$)	13.78%
Feature-wise max-drop ($p = 0.2$)	12.55%
Channel-wise max-drop ($p = 0.7$)	12.00%
Stochastic dropout ($N(0.0, 0.2)$)	11.79%
Stochastic dropout ($U(0.0, 0.4)$)	12.86%

[Analysis on the Dropout Effect in Convolutional Neural Networks, ACCV 2016](#)

BUSQUEDA HIPERPARAMETROS

➤ Hiper-parámetros

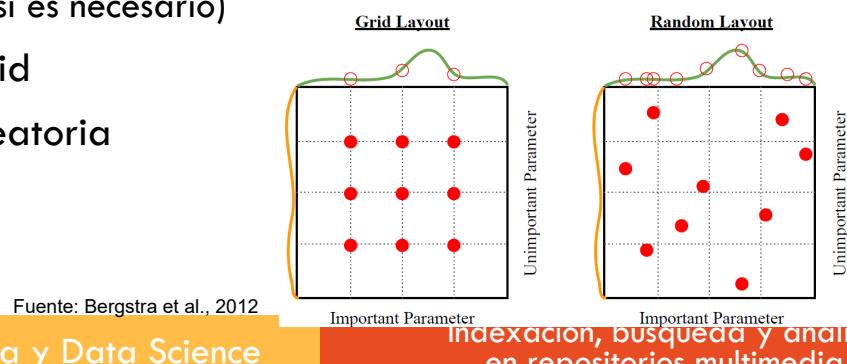
- Arquitectura de red (número de capas, número de filtros,...)
- Tasa de aprendizaje (*learning rate*)
- Factor de regularización (*decay rate*)
- Probabilidad de eliminación (*dropout*)
- ...



https://commons.wikimedia.org/wiki/File:Pioneer_DJ_equipment_-_scaled_left_-_ExpoMusic_2014.jpg

BUSQUEDA HIPERPARAMETROS

- Hiper-parámetros: búsqueda jerárquica de valores óptimos
 - Estrategia coarse-to-fine
 - 1^a etapa (coarse search): entrenamiento de pocas épocas para comprender que parámetros son relevantes
 - 2^a etapa (fine search): búsqueda exhaustiva con un mayor detalle de los parámetros seleccionados
 - ... (repetir si es necesario)
 - Búsqueda Grid
 - Búsqueda Aleatoria
 - ...

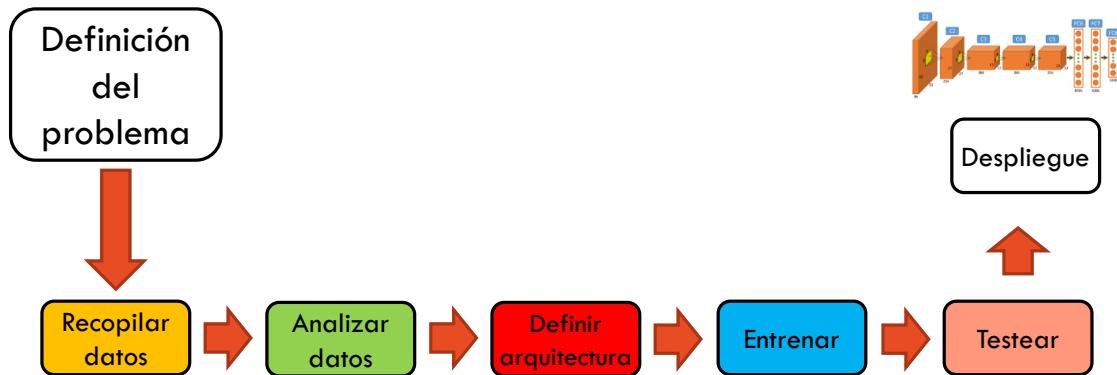


Agenda: Multimedia (imagen, video)

- Redes convolucionales: entrenamiento
 - Introducción
 - Conceptos básicos
 - Conceptos avanzados
 - Caso de estudio

CASO ESTUDIO: ESQUEMA

- Esquema general similar al método clásico de entrenamiento para aprendizaje automático

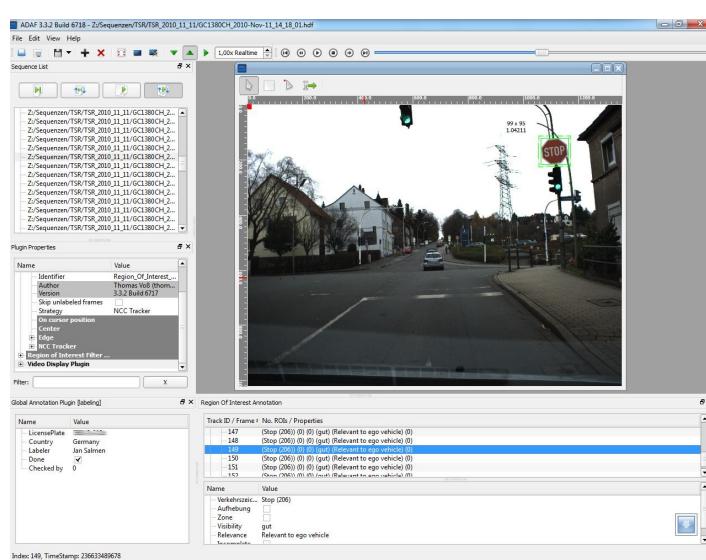


CASO ESTUDIO: ESQUEMA

- Ejemplo



Clasificar las imágenes del dataset
[German Traffic Sign image classification](#)
en las clases predefinidas



Fuente: <https://towardsdatascience.com/pipelines-mind-maps-and-convolutional-neural-networks-34bfc94db10c>

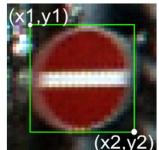
CASO ESTUDIO: RECOPIALAR DATOS

➤ Datos

- Emplear datasets estándar siempre que sea posible
- Recopilar datos de diferentes fuentes y condiciones
- Anotar datos mediante (varios) expertos

German Traffic Sign Recognition Benchmark

- Problema clasificación multi-clase
- Reconocimiento sobre imágenes
- +40 clases
- +50,000 imágenes en total
- Base de datos extensa y real



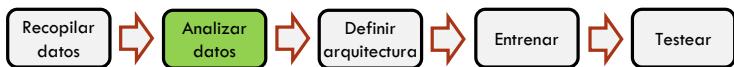
CASO ESTUDIO: ANALIZAR DATOS



- Comprender los objetivos del problema
- Visualizar datos con etiquetas correspondientes



CASO ESTUDIO: ANALIZAR DATOS

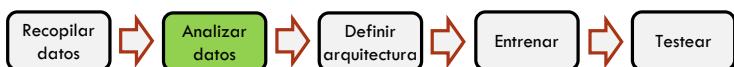


➤ Analizar datos para obtener ideas del diseño de la red



- Las imágenes tienen diferentes tamaños
- Las señales en imágenes tienen diferentes formas
- El brillo de las imágenes es aleatorio pero el color no lo es tanto
- Las imágenes pueden no ser frontales
- Los objetos no siempre se localizan en el centro de la imagen
- Las imágenes pueden estar ligeramente rotadas

CASO ESTUDIO: ANALIZAR DATOS



➤ Analizar datos para obtener ideas del diseño de la red



La distribución de clases no es homogénea

CASO ESTUDIO: ANALIZAR DATOS



➤ Conclusiones

- Redimensionar todas las imágenes al mismo tamaño
- Obtener nuevos datos para compensar clases minoritarias
- Normalización de imágenes para homogeneizar
- Experimentar con distintos espacios de color
- ...

CASO ESTUDIO: ANALIZAR DATOS

➤ Train/val/test



- División de los datos para entrenamiento y test
 - Entrenamiento (*train*)
 - Desarrollo y validación (*dev*) - opcional
 - Testeo (*test*)
- La proporción train/test depende del tamaño del dataset
 - Si el dataset es grande (~millones) → dev/test pequeños (~1%)
 - Si el dataset es pequeño (~millones) → dev/test moderados (~10-30%)
 - En datasets grandes ineficaz usar K particiones (*k-fold cross validation*)

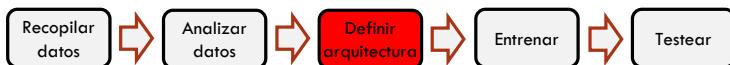


Train: 39209 imágenes (80%)

Dev: 0 imágenes

Test: 8000 imágenes (20%)

CASO ESTUDIO: DEFINIR ARQUITECTURA



➤ Opciones para definir la arquitectura

1. Definición completa del modelo (preciso pero costoso)

Crear una nueva red desde cero y aprender todos los parámetros. Requiere una cantidad considerable de datos de entrenamiento

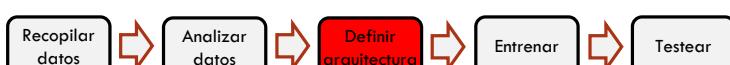
2. Utilizar un modelo pre-entrenado (ahorra tiempo)

Utilizar una arquitectura pre-entrenada que funciona para un problema similar. Aplicar *transfer Learning* para re-entrenar parcialmente un número reducido de parámetros.

3. Adaptar modelos de otras tareas (también ahorra tiempo)

Buscar modelos exitosos para otras tareas y adaptarlo a nuestro problema. Buenas arquitecturas pueden reutilizarse para varios problemas. Requiere modificaciones en la arquitectura y posiblemente el aprendizaje de un número reducido de parámetros

CASO ESTUDIO: DEFINIR ARQUITECTURA



➤ Bloques típicamente utilizados



CASO ESTUDIO: DEFINIR ARQUITECTURA



- Lectura de imágenes y redimensionado
- Conversión de espacio de color
- Normalización

Imagen original
(cualquier tamaño)



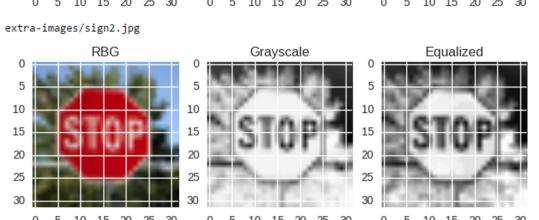
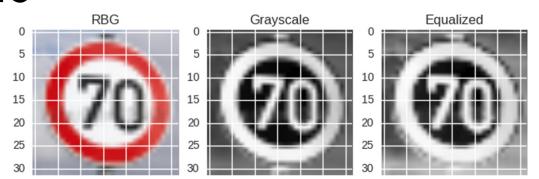
32x32 imagen
(el tamaño
depende de la
arquitectura de red
diseñada)

Indexación, búsqueda y análisis
en repositorios multimedia

CASO ESTUDIO: DEFINIR ARQUITECTURA



- Lectura de imágenes y redimensionado
- Conversión de espacio de color
- Normalización



CASO ESTUDIO: DEFINIR ARQUITECTURA



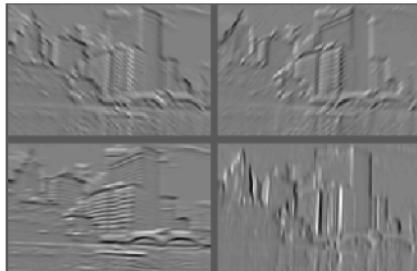
- Lectura de imágenes y redimensionado
- Conversión de espacio de color
- Normalización
 - Todas las imágenes en el dataset o cada batch de N imágenes
Input imágenes: $x_{i,j} \in N \times D$ Parámetros: $\gamma, \beta \in D$

$$\tilde{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}} \quad \begin{matrix} \text{media} \\ \text{varianza} \end{matrix} \qquad y_{i,j} = \gamma_j \tilde{x}_{i,j} + \beta_j$$

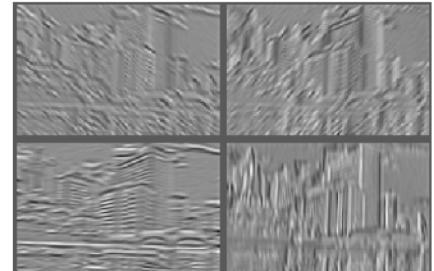
CASO ESTUDIO: DEFINIR ARQUITECTURA



- Lectura de imágenes y redimensionado
- Conversión de espacio de color
- Normalización



Feature Maps



Feature Maps
After Contrast Normalization

CASO ESTUDIO: DEFINIR ARQUITECTURA



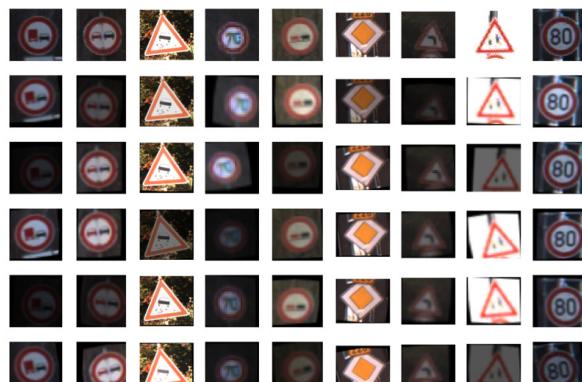
- Lectura de imágenes y redimensionado
- Conversión de espacio de color
- Normalización

Cualquier operación de normalización realizada debe ser calculada solamente con datos de entrenamiento pero aplicada siempre en los conjuntos train/dev/test

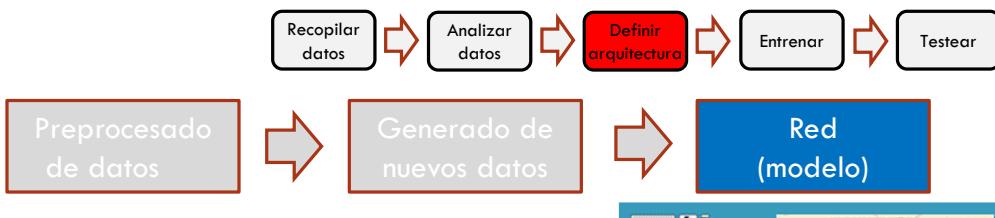
CASO ESTUDIO: DEFINIR ARQUITECTURA



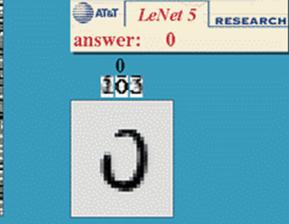
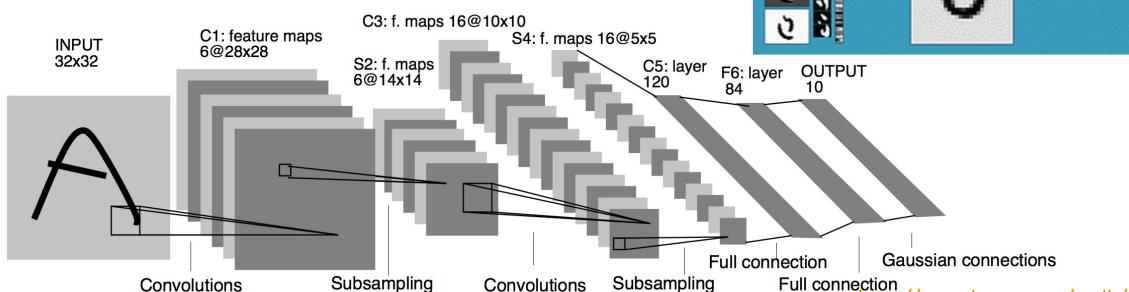
1. Introducir cambios aleatorios cada época de entrenamiento
 - Brillo
 - Pequeñas rotaciones
 - Pequeñas traslaciones
 - Pequeñas deformaciones
2. Introducir nuevos datos modificando imágenes (e.g. cambios anteriores)



CASO ESTUDIO: DEFINIR ARQUITECTURA



➤ Arquitectura basada en LENET

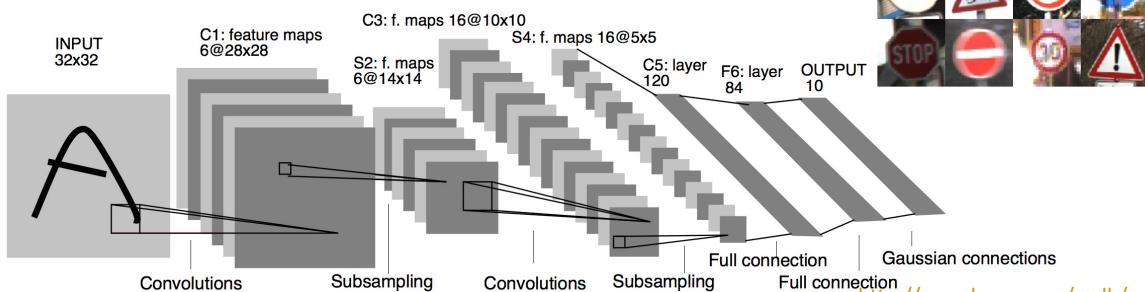


<http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf>

CASO ESTUDIO: DEFINIR ARQUITECTURA



➤ Arquitectura basada en LENET

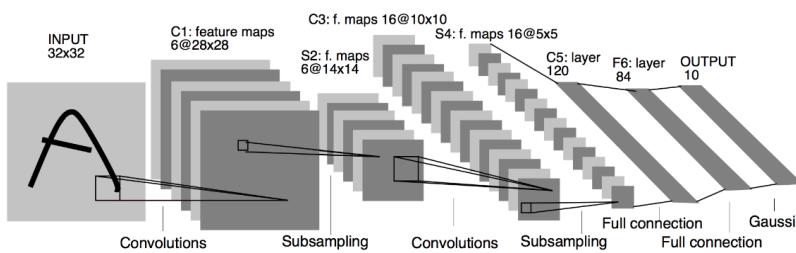


<http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf>

CASO ESTUDIO: DEFINIR ARQUITECTURA



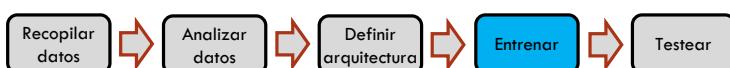
➤ Arquitectura basada en LENET



Layer	Shape
Input	32x32x3
Convolution (valid, 5x5x6)	28x28x6
Max Pooling (valid, 2x2)	14x14x6
Activation (ReLU)	14x14x6
Convolution (valid, 5x5x16)	10x10x16
Max Pooling (valid, 2x2)	5x5x16
Activation (ReLU)	5x5x16
Flatten	400
Dense	120
Activation (ReLU)	120
Dense	43
Activation (Softmax)	43

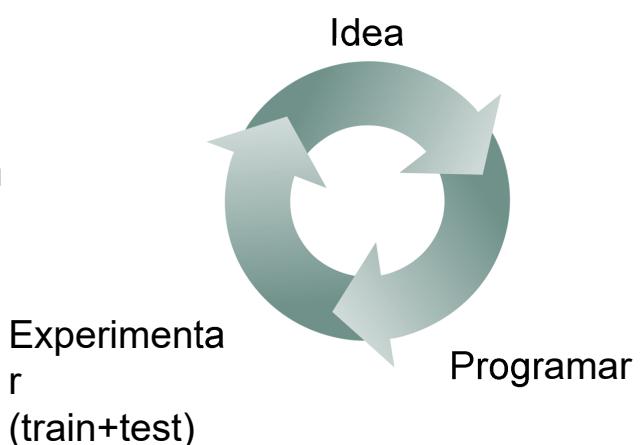
[ublis/pdf/lecun-98.pdf](http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf)

CASO ESTUDIO: ENTRENAMIENTO



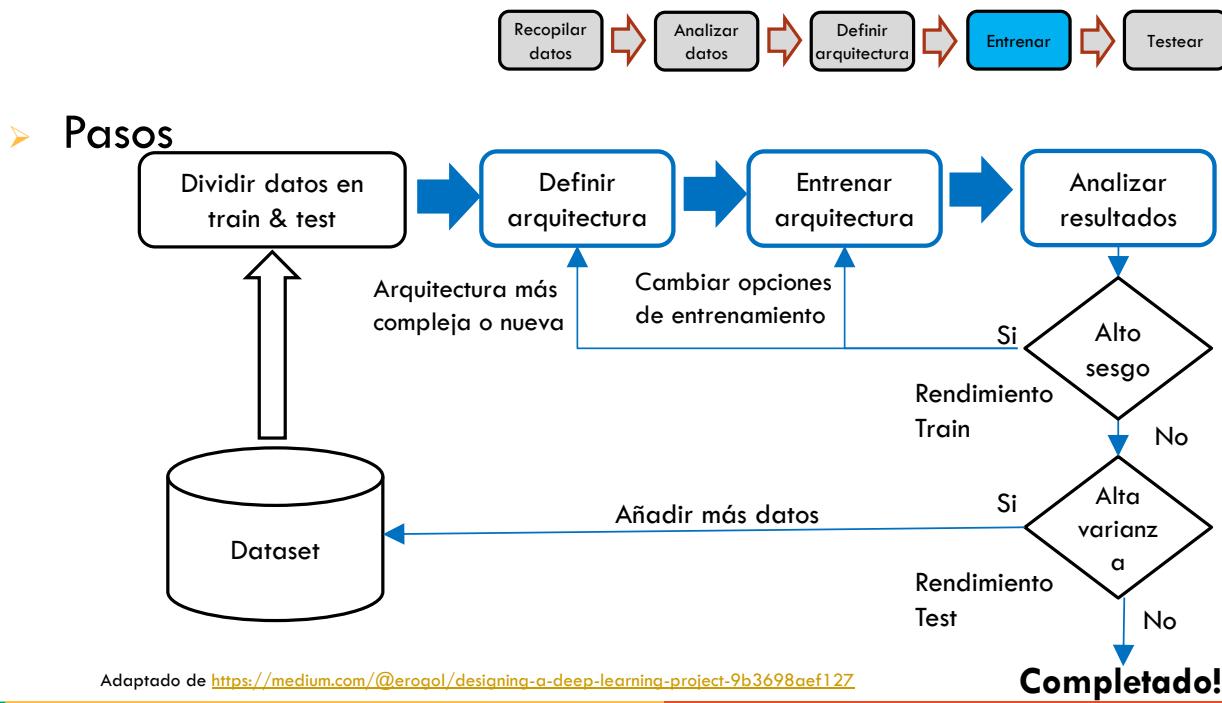
➤ Proceso iterativo

- # capas
- # unidades en cada capa
- Tasas de aprendizaje
- Funciones de activación
- Optimización
- Regularización
- ...

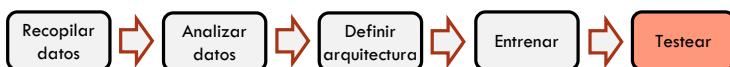


Objetivo: minimizar una función de coste $L(x_j, y_j; \theta)$
mediante modificaciones de los parámetros θ

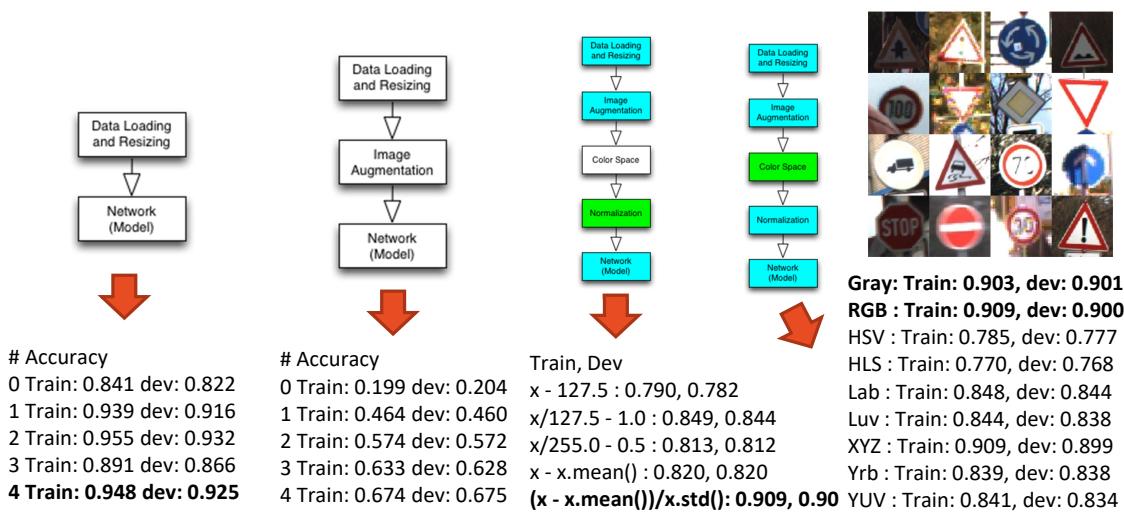
CASO ESTUDIO: ENTRENAMIENTO



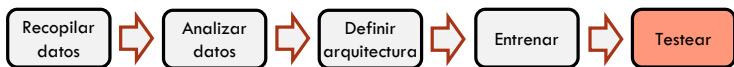
CASO ESTUDIO: TEST



Pruebas iniciales



CASO ESTUDIO: TEST



➤ Pruebas con distintas arquitecturas

```
network1 = (NeuralNetwork()
    .input([32, 32, 3])
    .conv([5, 5, 6])
    .max_pool()
    .relu()
    .conv([5, 5, 16])
    .max_pool()
    .relu()
    .flatten()
    .dense(120)
    .relu()
    .dense(43))
```

Best case → Train: 0.948 dev: 0.925

```
network2 = NeuralNetwork()
    .input([32, 32, 3])
    .conv([5, 5, 12]) #
doubled
    .max_pool()
    .relu()
    .conv([5, 5, 32]) #
doubled
    .max_pool()
    .relu()
    .flatten()
.dense(240) # doubled
    .relu()
    .dense(43)
```

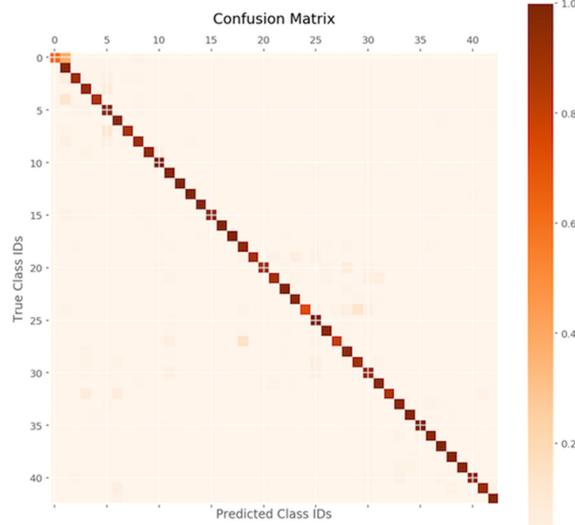
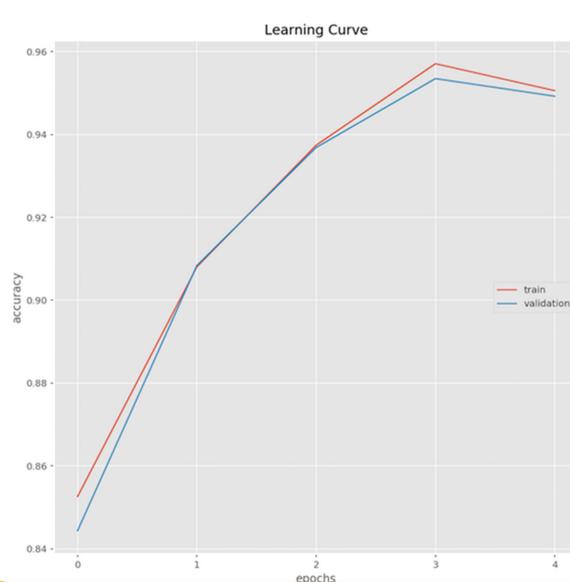
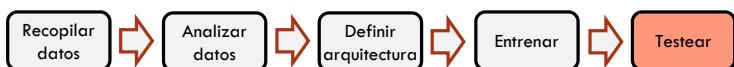
0 Train: 0.853 dev: 0.844
1 Train: 0.908 dev: 0.908
2 Train: 0.937 dev: 0.937
3 Train: 0.957 dev: 0.954
4 Train: 0.951 dev: 0.949



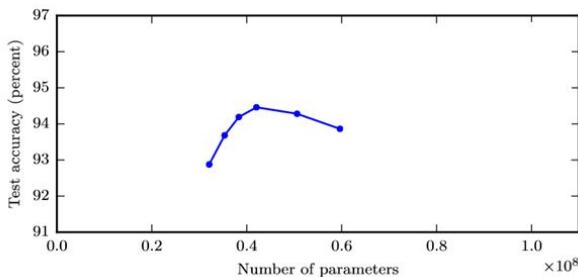
Mejora cambiando la complejidad de la red

CASO ESTUDIO: TEST

➤ Resultado final



- El siguiente gráfico muestra la precisión de una red neuronal convolucional de 3 capas entrenada frente al número de parámetros que se pueden definir para la arquitectura de 3 capas



- Indique las posibles razones de la disminución que observa
 - (a) Incluso si aumenta el número de parámetros, solo algunos de ellos se utilizan para la predicción
 - (b) A medida que aumenta el número de parámetros, el poder predictivo de la red neuronal disminuye
 - (c) A medida que aumenta el número de parámetros, éstos comienzan a correlacionarse entre sí, especializando la red que a su vez ayuda al sobreajuste

CASO ESTUDIO: TEST

➤ ¿Después?



- Datos: nuevas capturas para balancear distribución de clases en el dataset
- Complejidad red: más capas convoluciones y filtros, capas fully connected
- Cambiar función de activación no-lineal: leaky RELU,...
- Entrenamiento:
 - Más épocas (hasta 500)
 - Reducir tasa de aprendizaje (o variable)
 - Inicialización de parámetros (e.g. aleatoria)
 - Cambiar optimizador
- ...



RESUMEN

- Existen múltiples estrategias cuyo objetivo es evitar el sobreajuste (*overfitting*) de la arquitectura diseñada
 - Coeficiente de regularización, *batch normalization*, *data augmentation*, *dropout*,...
 - Aunque también existen otras formas de “regularizar”: añadir nuevos datos, Redimensionado espacial (*spatial pooling*),...
- El ajuste hiper-parámetros es una tarea con alto componente de investigación

REFERENCIAS

- Básico:
 - Ian Goodfellow, Yoshua Bengio, and Aaron Courville, "Deep Learning", MIT Press, 2016 **Secciones 6 , 7.1-7.5, 7.12** (libro disponible gratuitamente en <http://www.deeplearningbook.org/>)
- Adicional:
 - [Boyd & Vandenberghe 2004] Boyd, Stephen; Vandenberghe, Lieven (2004). "Unconstrained Minimization". Convex Optimization. New York: Cambridge University Press. pp. 457–520. ISBN 0-521-83378-7.
 - [Rumelhart et al, 1981] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1988). Learning representations by back-propagating errors. Cognitive modeling, 5(3), 1.
 - [LeCun, 1998] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proc. of the IEEE, 86(11), 2278-2324.
 - [Glorot, AISTATS10] Glorot, X., & Bengio, Y. (2010, March). Understanding the difficulty of training deep feedforward neural networks. Int. Conference on Artificial Intelligence and Statistics (pp. 249-256).
 - [He et al, CVPR15] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on Imagenet classification. In Proceedings of the IEEE Int conference on computer vision (pp. 1026-1034).