



Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

Generalización

Rodrigo Toro Icarte (rntoro@uc.cl)

MCD - Aprendizaje profundo

31 de Octubre, 2023

Un poco sobre mí...

Cosas que ya saben de mí:

Un poco sobre mí...

Cosas que ya saben de mí:

- Me llamo Rodrigo.
- Trabajo en la Pontificia Universidad Católica de Chile.

Un poco sobre mí...

Cosas que ya saben de mí:

- Me llamo Rodrigo.
- Trabajo en la Pontificia Universidad Católica de Chile.

Cosas que no saben de mí:

- Hice mi doctorado en la Universidad de Toronto.
- Me especializo en aprendizaje reforzado.

Policy gradient methods¹

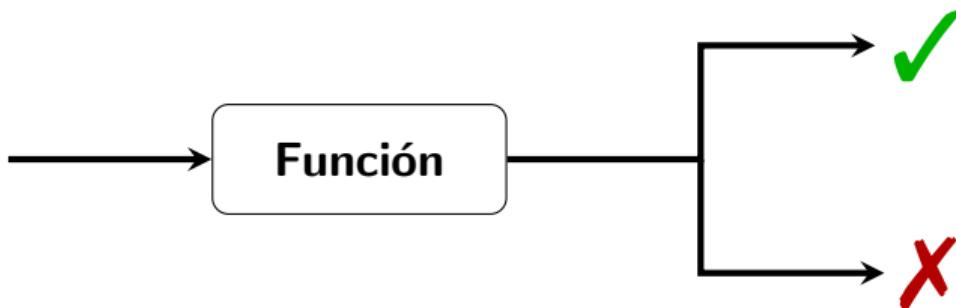
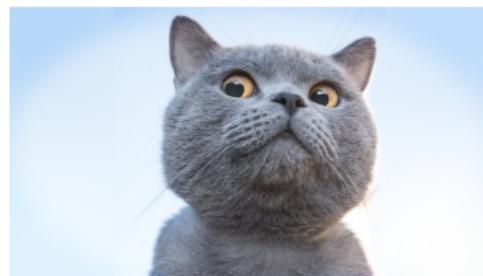
¹Heess et al. "Emergence of Locomotion Behaviours in Rich Environments." ArXiv (2017).

Learning Agile Soccer Skills²

²Tuomas Haarnoja et al. "Learning Agile Soccer Skills for a Bipedal Robot with Deep Reinforcement Learning." ArXiv (2023).

Recordatorio

Deep learning



Deep learning

Input	Output
	✓
	✓
	✓

Deep learning

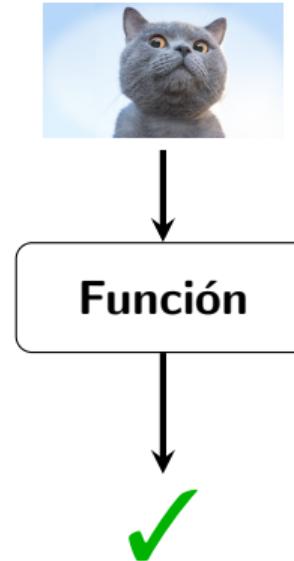
Input	Output
	✓
	✓
	✓
	✗
	✗
...	...

Deep learning

Input	Output	Función
	✓	
	✓	
	✓	
	✗	
	✗	
...	...	

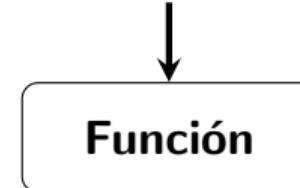
Deep learning

Input	Output
	✓
	✓
	✓
	✗
	✗
...	...



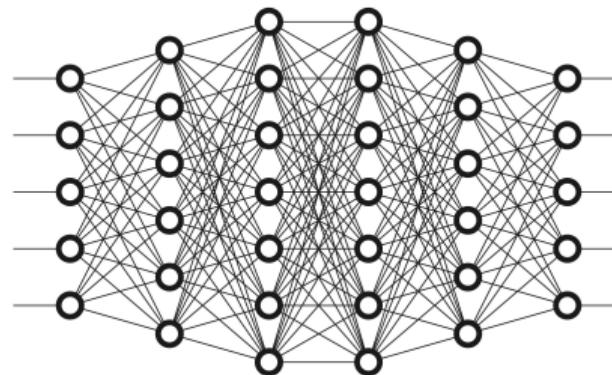
Deep learning

Input	Output
	✓
	✓
	✓
	✗
	✗
...	...



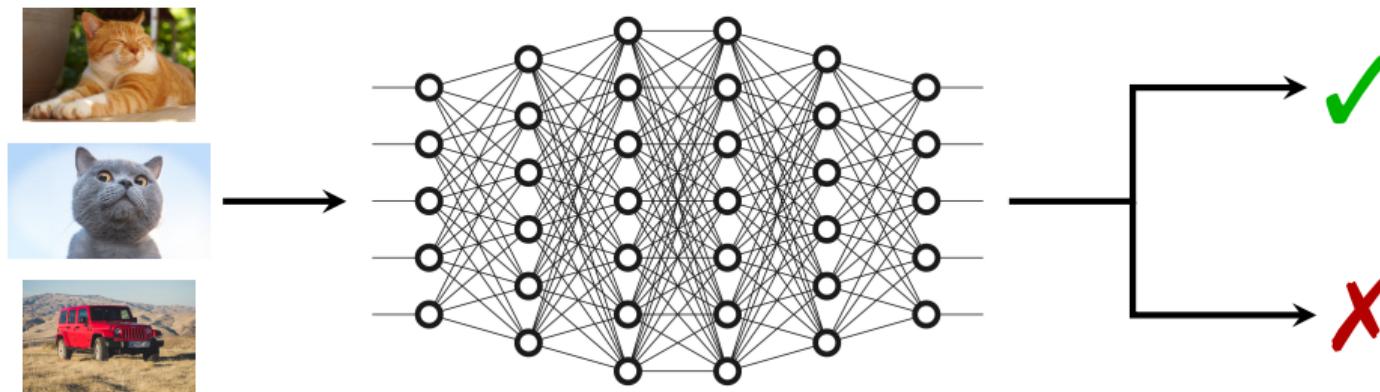
✗

Esta función es una red neuronal, cuya salida depende del valor de sus pesos.



Deep learning

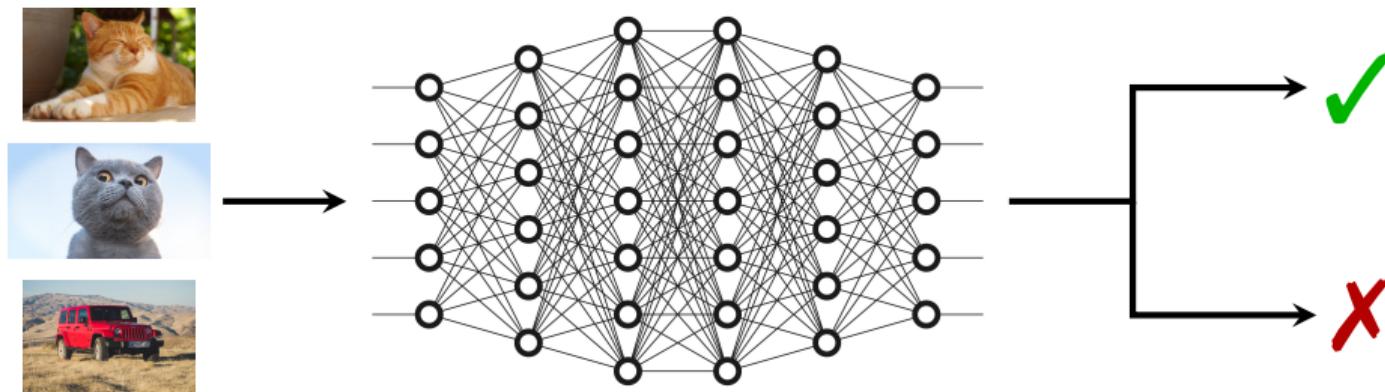
Esta función es una red neuronal, cuya salida depende del valor de sus pesos.



Entrenar la red significa encontrar pesos que generan la salida correcta.

Deep learning

Esta función es una red neuronal, cuya salida depende del valor de sus pesos.

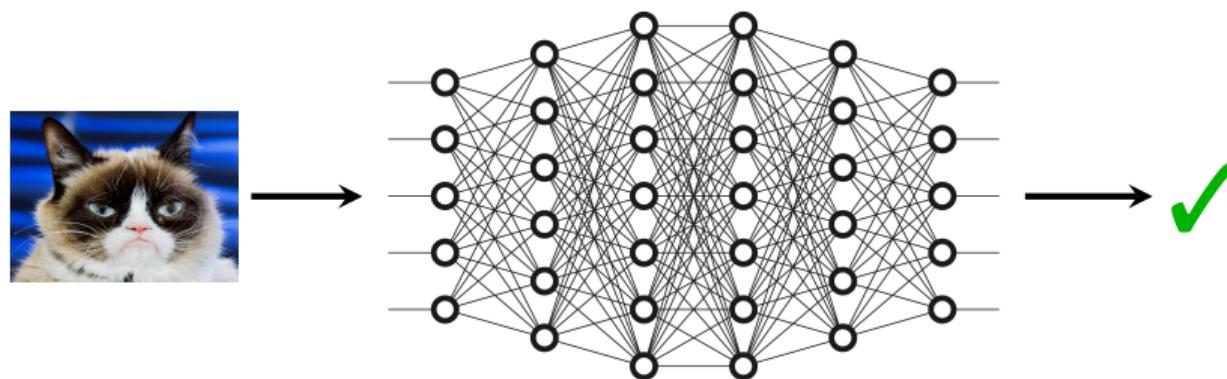


Entrenar la red significa encontrar pesos que generan la salida correcta.

La gracia de deep learning es que generaliza bien.

Deep learning

Esta función es una red neuronal, cuya salida depende del valor de sus pesos.



Entrenar la red significa encontrar pesos que generan la salida correcta.

La gracia de deep learning es que generaliza bien.

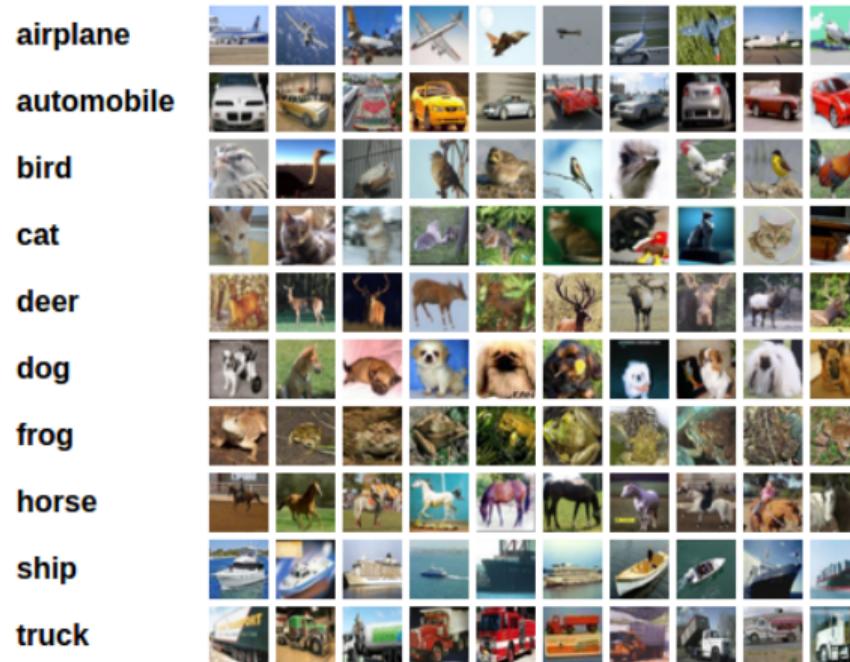
En la clase de hoy

En la clase de hoy:

- Cómo entrenar modelos con muchos datos.
- Generalización.

Entrenando con muchos datos

Ejemplo: CIFAR10 dataset



Objetivo

Entrenar una red que nos diga si una imagen contiene un avión, auto, pájaro, gato, ciervo, perro, rana, caballo, barco o camión.

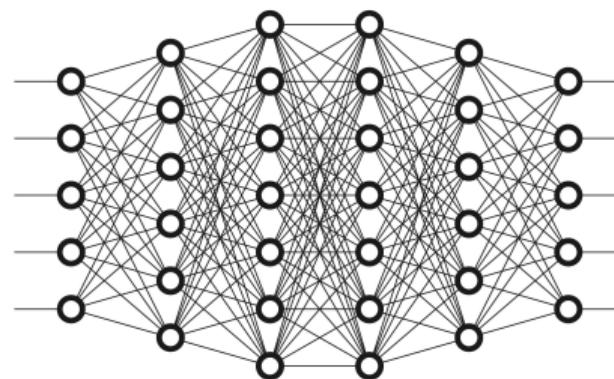
Objetivo

Entrenar una red que nos diga si una imagen contiene un avión, auto, pájaro, gato, ciervo, perro, rana, caballo, barco o camión.

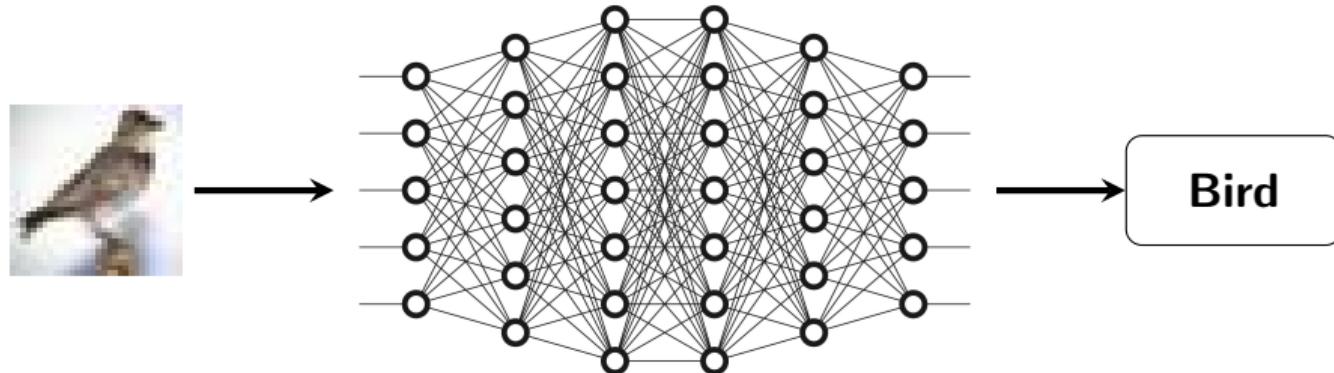
CIFAR-10:

- Input: Imágenes de 32×32 pixeles RGB.
- Output: Diez clases posibles.
- Contiene 60000 ejemplos en total.
- Hay 6000 ejemplos por clase.

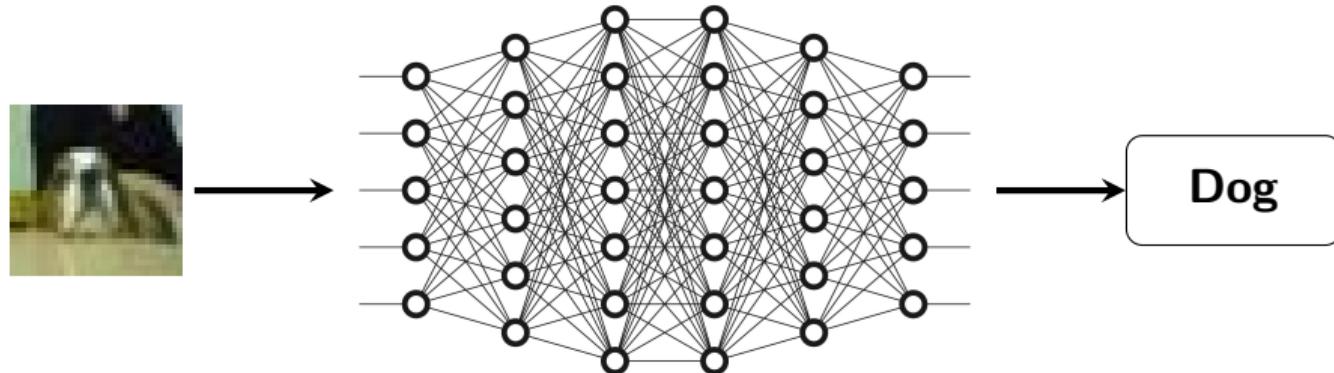
Deep learning



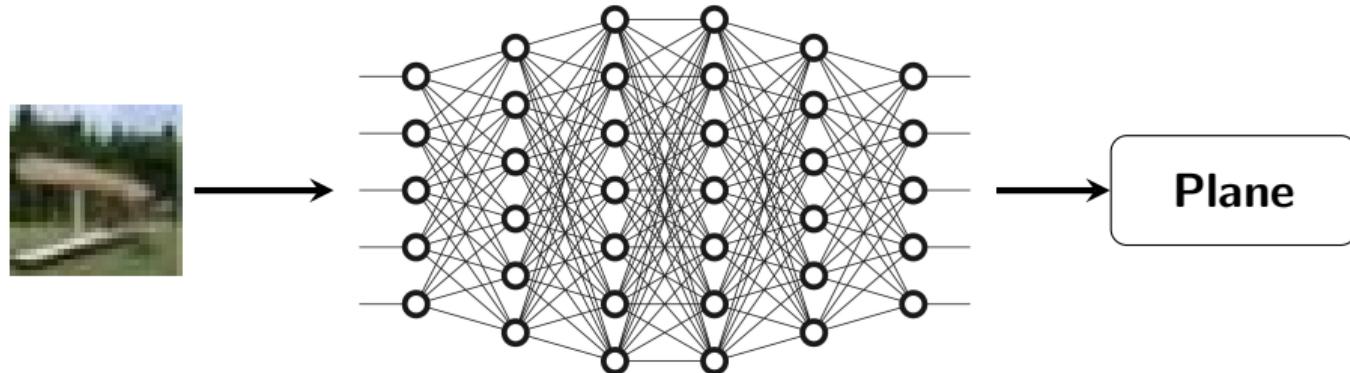
Deep learning



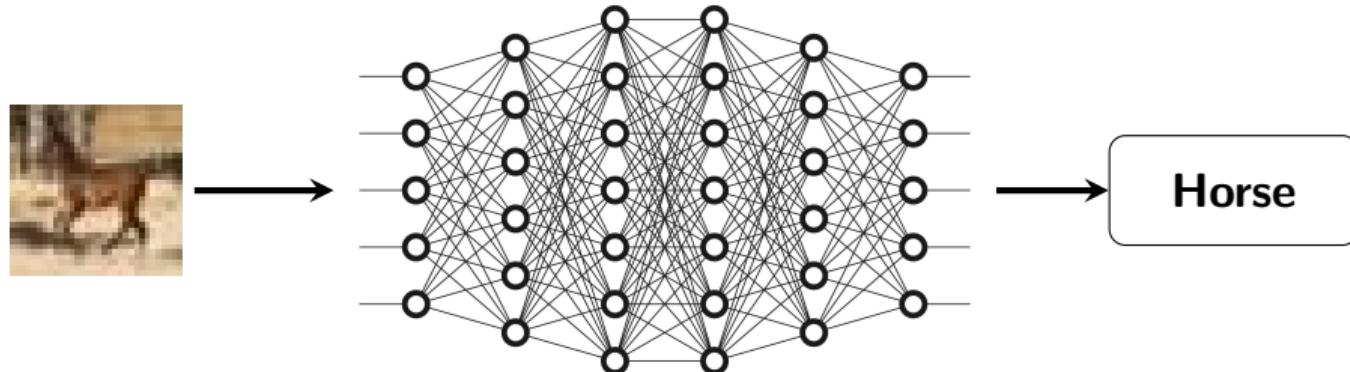
Deep learning



Deep learning

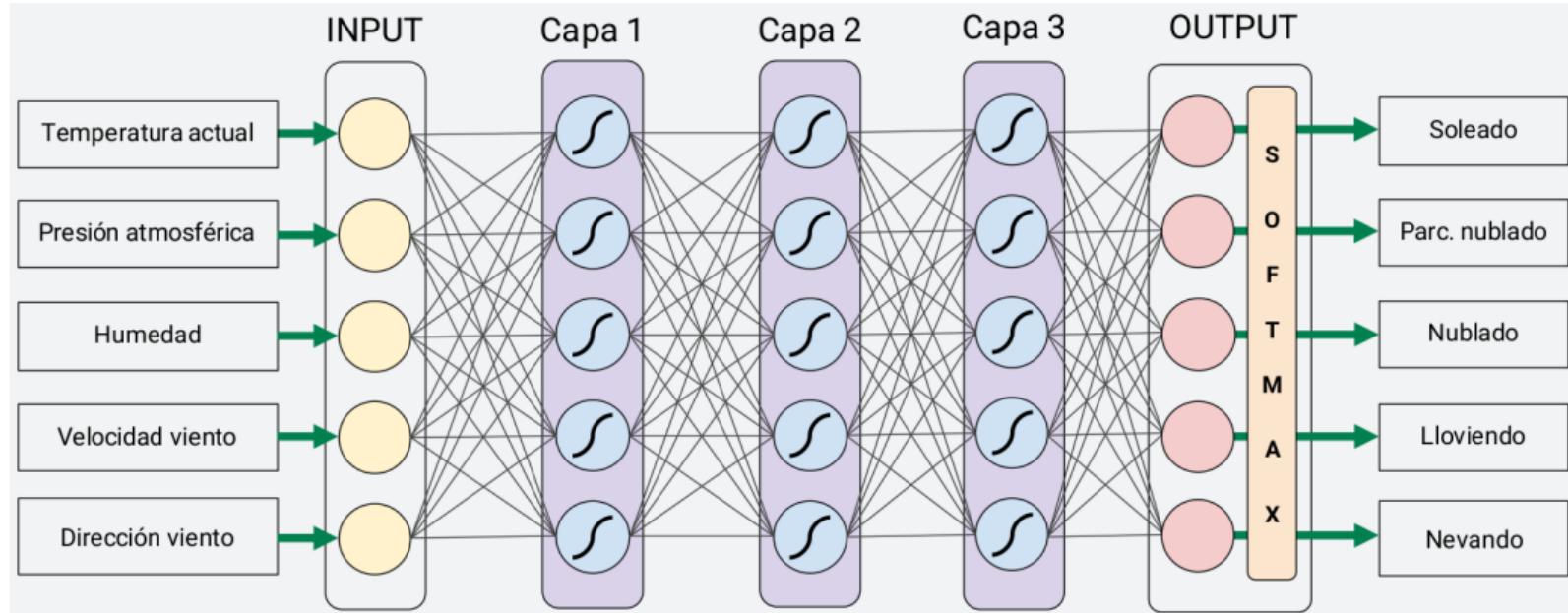


Deep learning



CIFAR-10

Como se trata de un problema de **clasificación** usaremos una red parecida a esta:



... que entrenamos usando **cross-entropy**.

¿Cómo codificamos el input de la red?

¿Cómo codificamos el input de la red?



¿Cómo codificamos el input de la red?



Input: Un vector de $32 \times 32 \times 3$ números entre 0 y 255.

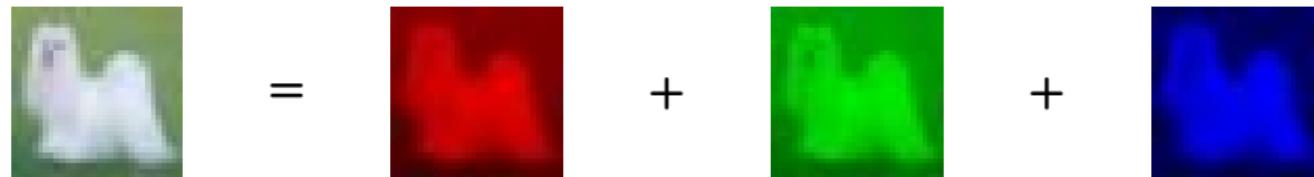
¿Cómo codificamos el input de la red?



Input: Un vector de $32 \times 32 \times 3$ números entre 0 y 255.



¿Cómo codificamos el input de la red?



Input: Un vector de $32 \times 32 \times 3$ números entre 0 y 255.



¿Cómo codificamos el input de la red?



Input: Un vector de $32 \times 32 \times 3$ números entre 0 y 255.



¿Cómo codificamos el output de la red?

¿Cómo codificamos el input de la red?



Input: Un vector de $32 \times 32 \times 3$ números entre 0 y 255.



¿Cómo codificamos el output de la red?

Tenemos 10 neuronas de salida, una por cada clase posible, con activación softmax.

Arquitectura red:

- Input: $32 \times 32 \times 3$.
- Capa fully-connected de 120 neuronas relu.
- Capa fully-connected de 84 neuronas relu.
- Capa fully-connected de 10 neuronas softmax.

Arquitectura red:

- Input: $32 \times 32 \times 3$.
- Capa fully-connected de 120 neuronas relu.
- Capa fully-connected de 84 neuronas relu.
- Capa fully-connected de 10 neuronas softmax.

Función de pérdida: Cross-entropy.

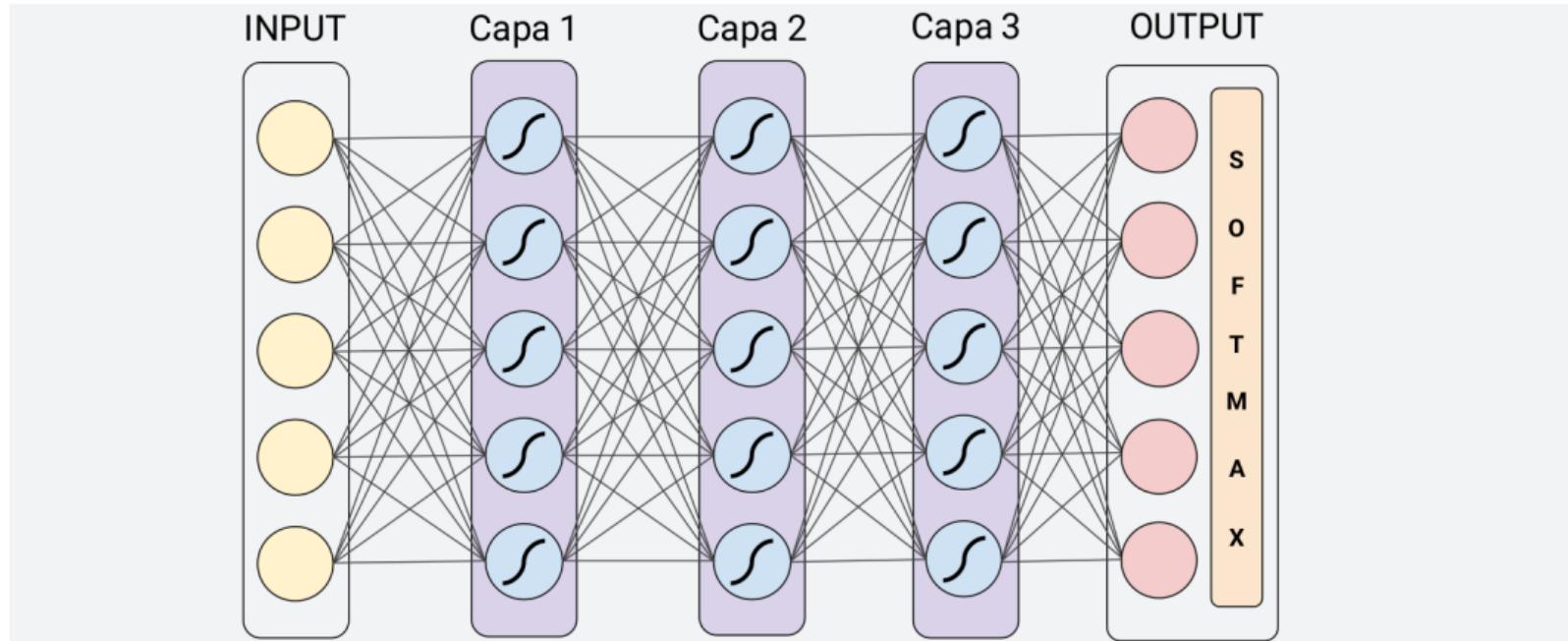
Arquitectura red:

- Input: $32 \times 32 \times 3$.
- Capa fully-connected de 120 neuronas relu.
- Capa fully-connected de 84 neuronas relu.
- Capa fully-connected de 10 neuronas softmax.

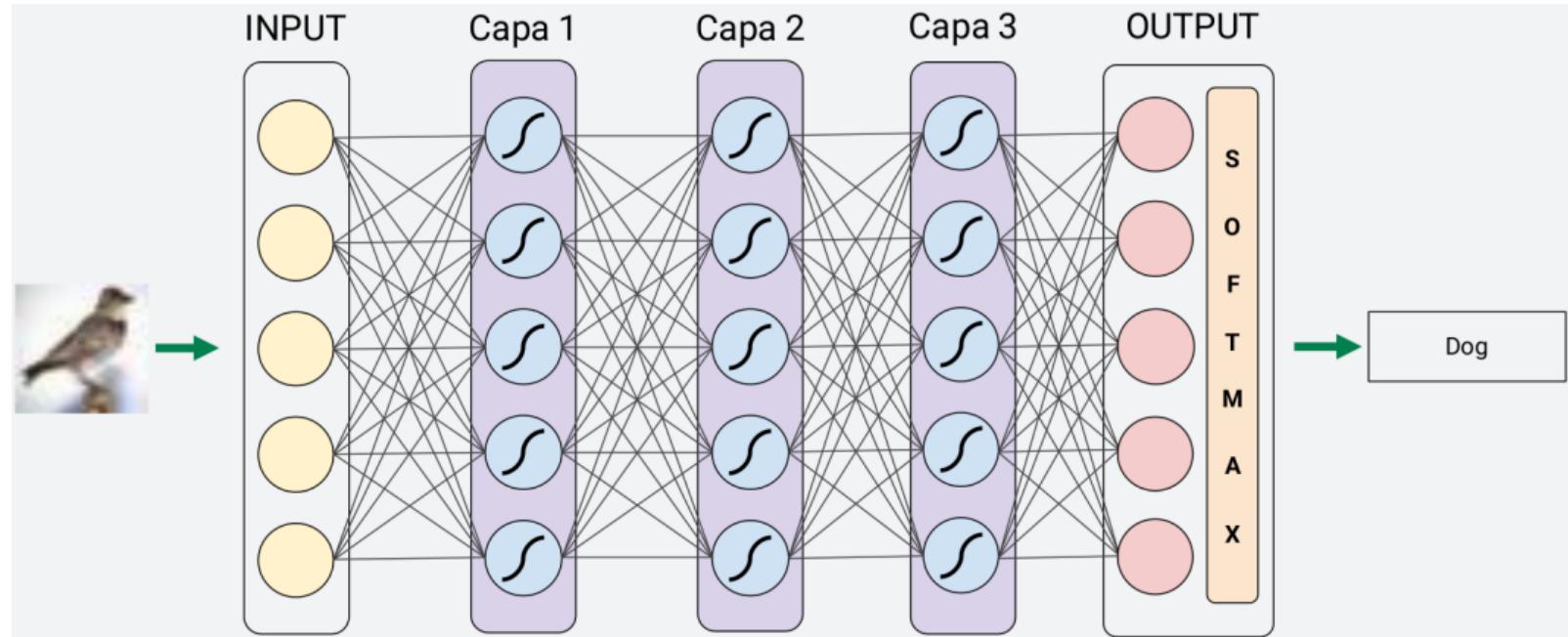
Función de pérdida: Cross-entropy.

... y entrenamos esta red usando descenso de gradiente.

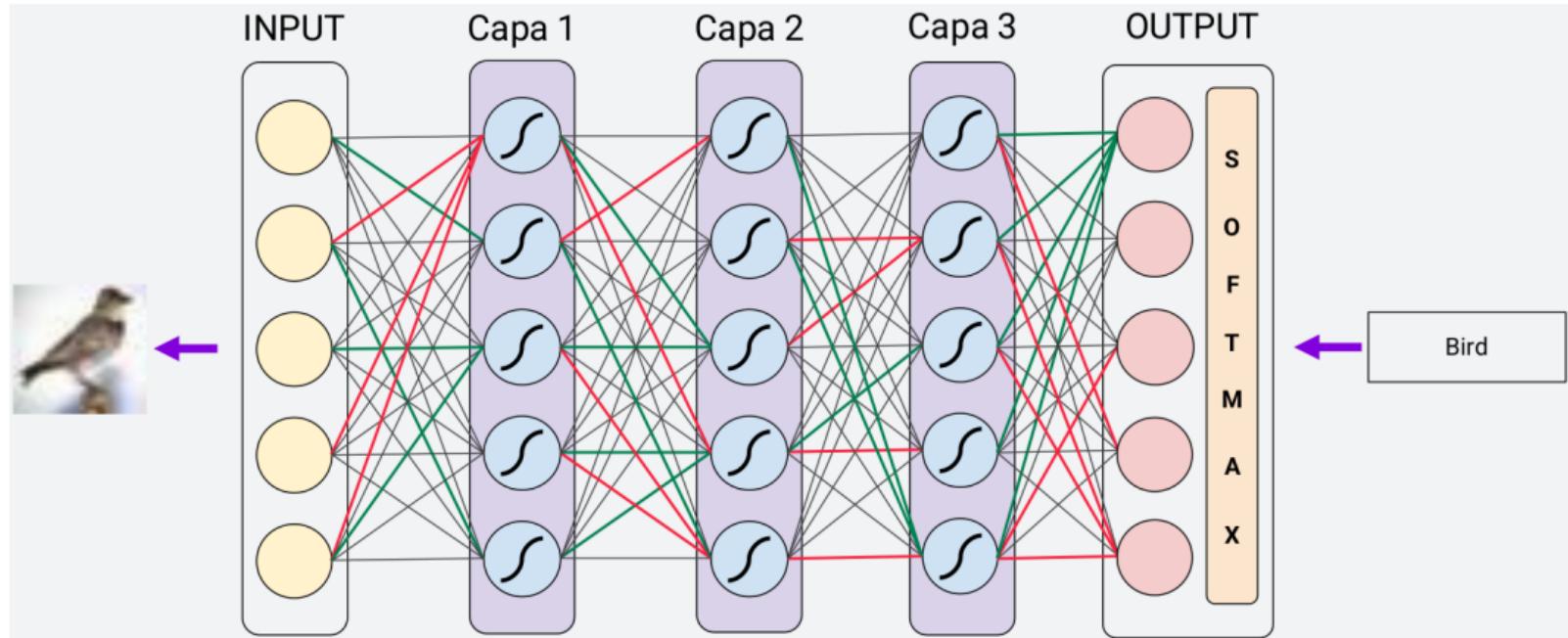
CIFAR-10



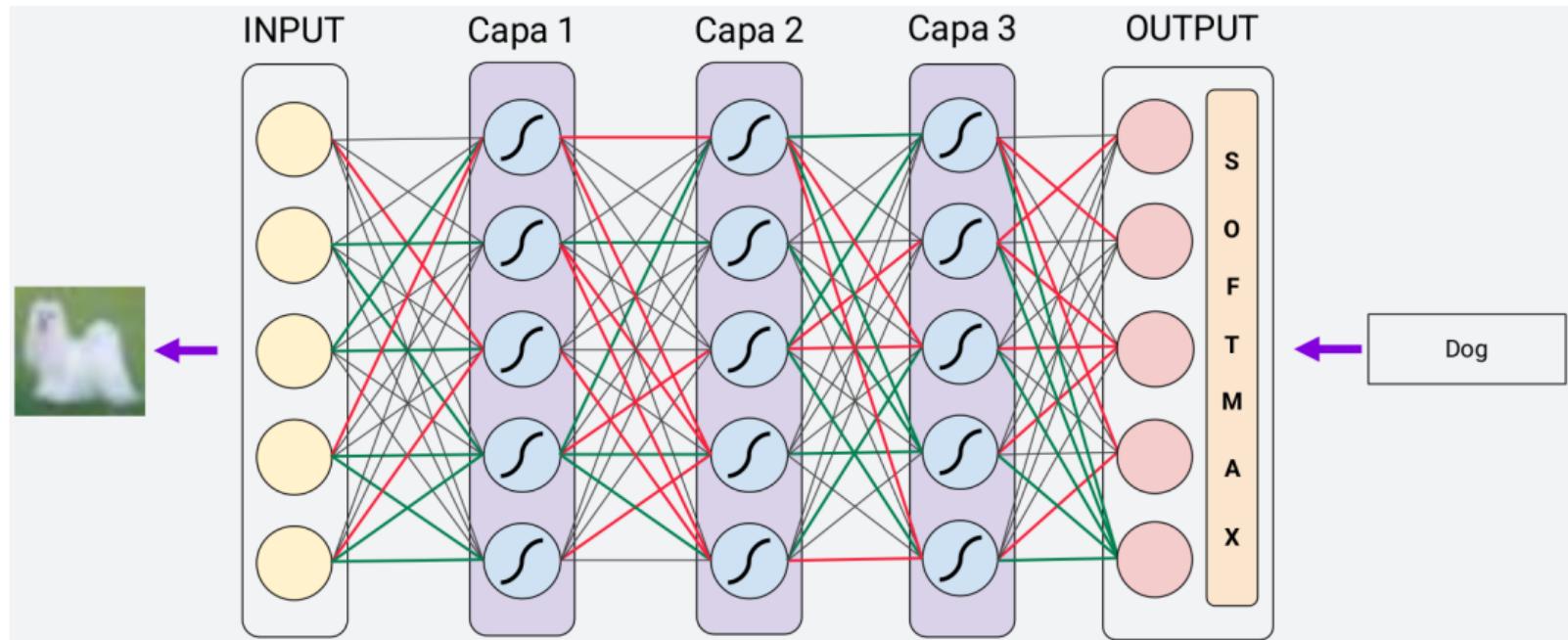
CIFAR-10



CIFAR-10



CIFAR-10



Queremos encontrar pesos θ para la red que minimizan la función de pérdida:

$$\arg \min_{\theta} J(\theta) = \arg \min_{\theta} - \sum_{(x,y) \in \mathcal{T}} \mathbf{e}_y^\top \cdot \log(f_\theta(x))$$

Queremos encontrar pesos θ para la red que minimicen la función de pérdida:

$$\arg \min_{\theta} J(\theta) = \arg \min_{\theta} - \sum_{(x,y) \in \mathcal{T}} \mathbf{e}_y^\top \cdot \log(f_\theta(x))$$

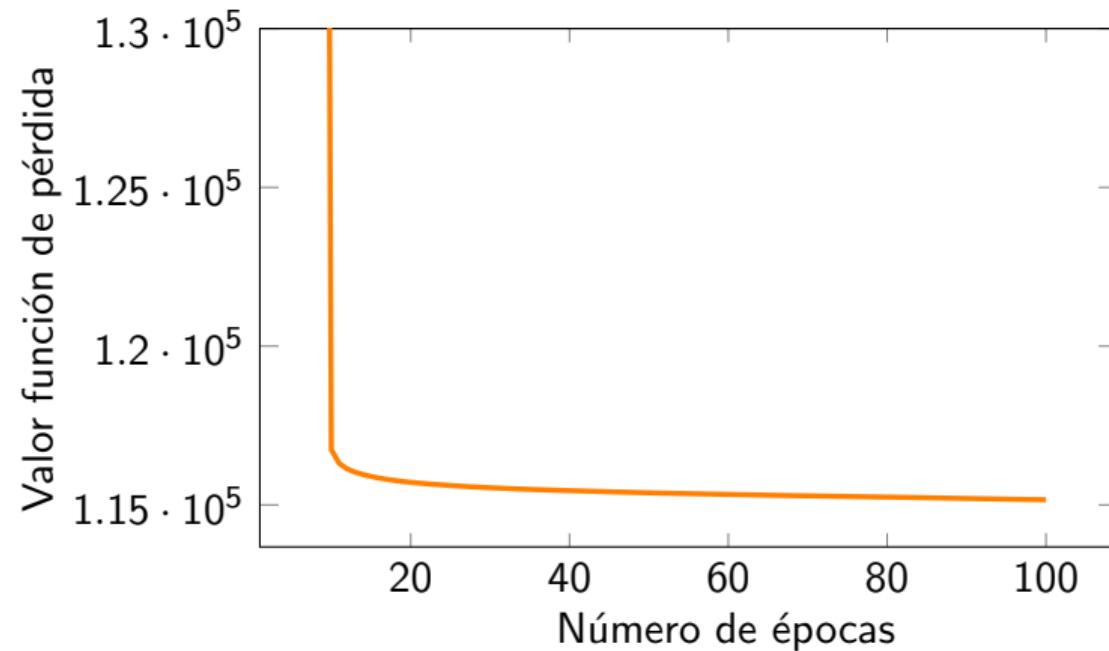
Descenso de gradiente (GD):

- Comenzamos desde pesos cualquiera θ_0 .
- Repetir por $t = 1 \dots T$ épocas:
 - Calculamos el gradiente respecto a **todos** los ejemplos de entrenamiento:

$$\nabla_{\theta} J(\theta_t) = \sum_{(x,y) \in \mathcal{T}} \nabla_{\theta} J(\theta_t; x, y)$$

- Actualizamos los pesos en contra del gradiente: $\theta_{t+1} \leftarrow \theta_t - \alpha \nabla_{\theta} J(\theta_t)$.

Usamos descenso de gradiente en CIFAR-10 y este fue el resultado:



Queremos encontrar pesos θ para la red que minimicen la función de pérdida:

$$\arg \min_{\theta} J(\theta) = \arg \min_{\theta} - \sum_{(x,y) \in \mathcal{T}} \mathbf{e}_y^\top \cdot \log(f_\theta(x))$$

Descenso de gradiente (GD):

- Comenzamos desde pesos cualquiera θ_0 .
- Repetir por $t = 1 \dots T$ épocas:
 - Calculamos el gradiente respecto a **todos** los ejemplos de entrenamiento:

$$\nabla_{\theta} J(\theta_t) = \sum_{(x,y) \in \mathcal{T}} J(\theta_t; x, y)$$

- Actualizamos los pesos en contra del gradiente: $\theta_{t+1} \leftarrow \theta_t - \alpha \nabla_{\theta} J(\theta_t)$.

Queremos encontrar pesos θ para la red que minimicen la función de pérdida:

$$\arg \min_{\theta} J(\theta) = \arg \min_{\theta} - \sum_{(x,y) \in \mathcal{T}} \mathbf{e}_y^\top \cdot \log(f_\theta(x))$$

Descenso de gradiente (GD):

- Comenzamos desde pesos cualquiera θ_0 .
- Repetir por $t = 1 \dots T$ épocas:
 - Calculamos el gradiente respecto a **todos** los ejemplos de entrenamiento:

$$\nabla_{\theta} J(\theta_t) = \sum_{(x,y) \in \mathcal{T}} J(\theta_t; x, y)$$

- Actualizamos los pesos en contra del gradiente: $\theta_{t+1} \leftarrow \theta_t - \alpha \nabla_{\theta} J(\theta_t)$.

Descenso de gradiente estocástico:

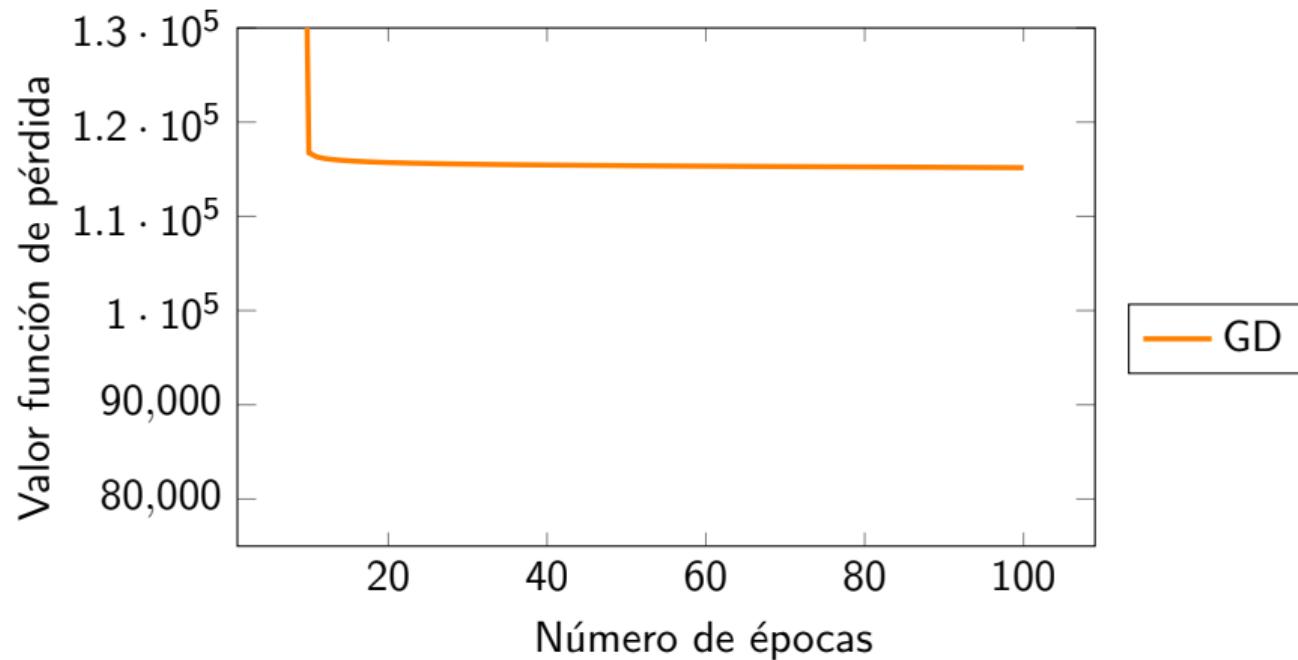
- Comenzamos desde pesos cualquiera θ_0 .
- Repetir por $t = 1 \dots T$ épocas:
 - Dividir el set de entrenamiento en N batches de manera aleatoria.
 - Repetir por $B_n \in \{B_1 \dots B_N\}$:
 - Calculamos el gradiente respecto al batch B_n :

$$\nabla_{\theta} J(\theta_t; B_n) = \sum_{(x,y) \in B_n} J(\theta_t; x, y)$$

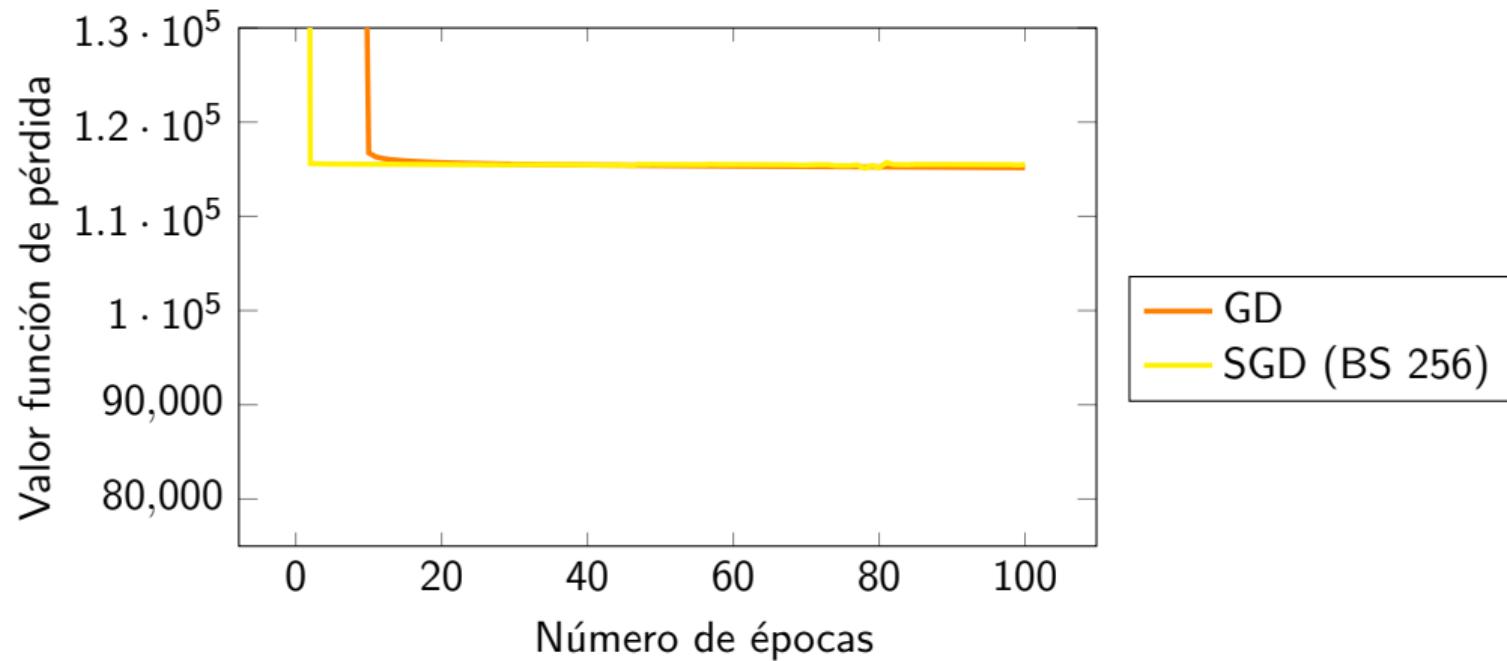
- Actualizamos los pesos en contra del gradiente:

$$\theta_{t+1} \leftarrow \theta_t - \alpha \nabla_{\theta} J(\theta_t; B_n)$$

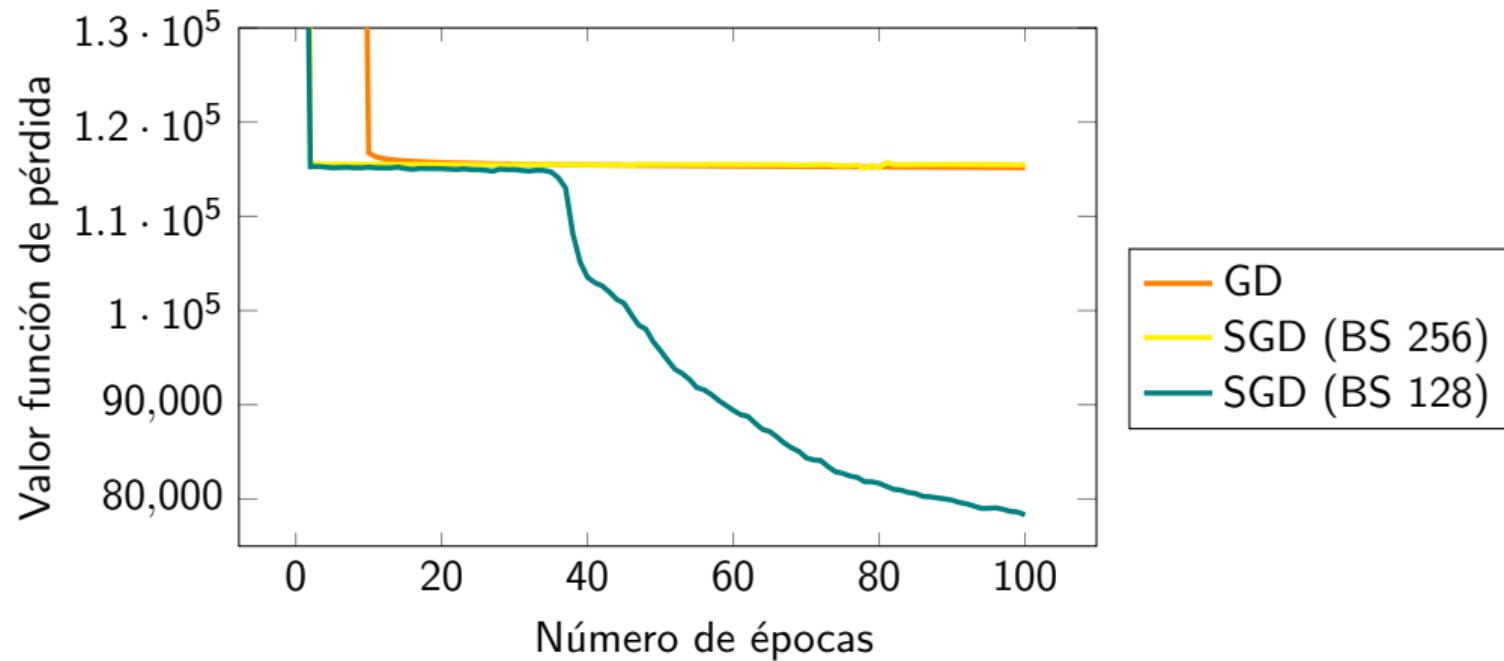
Repetimos el experimento usando distintos valores para el *batch size*.



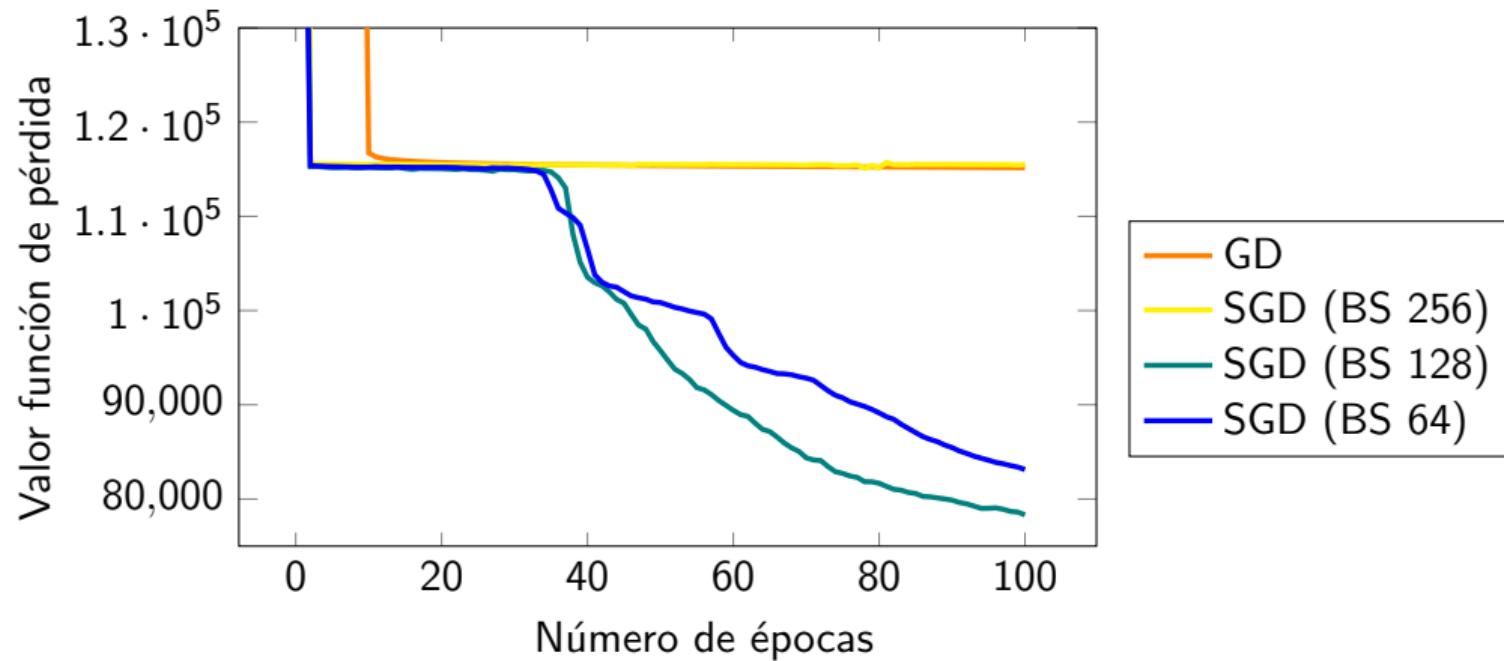
Repetimos el experimento usando distintos valores para el *batch size*.



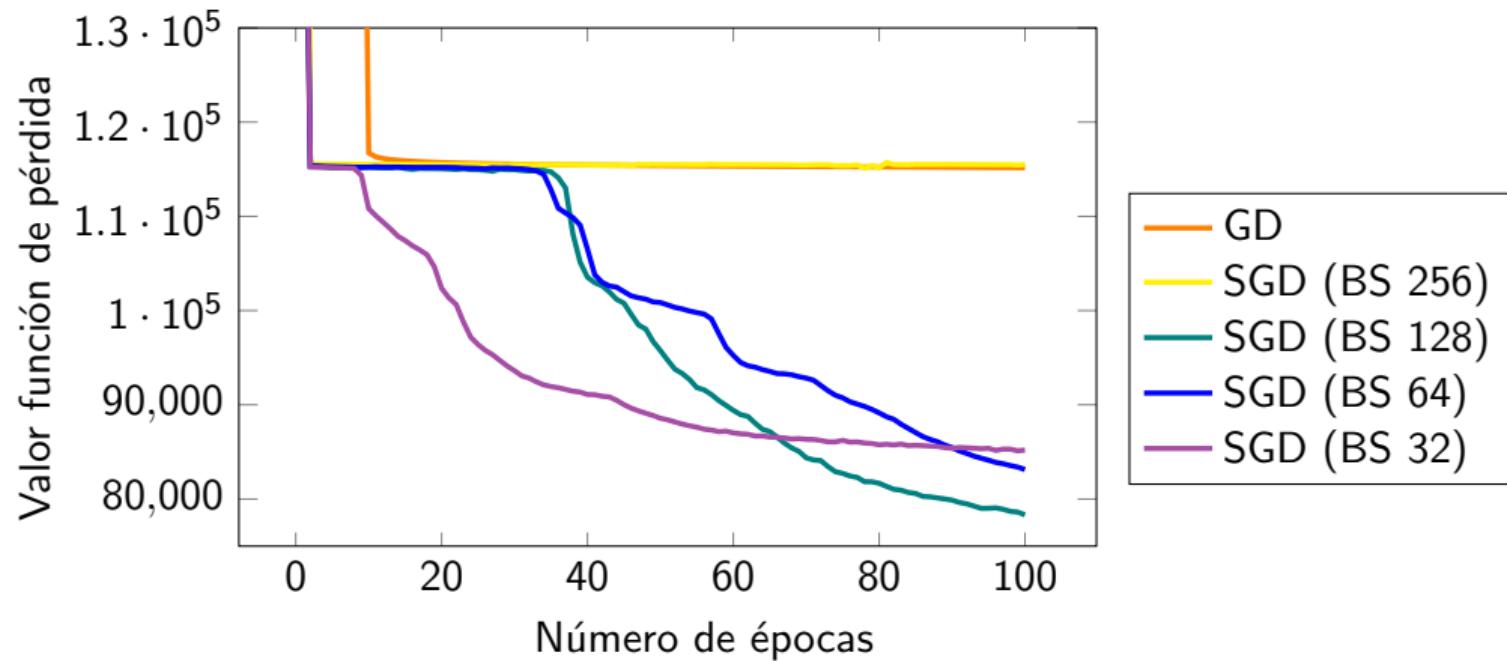
Repetimos el experimento usando distintos valores para el *batch size*.



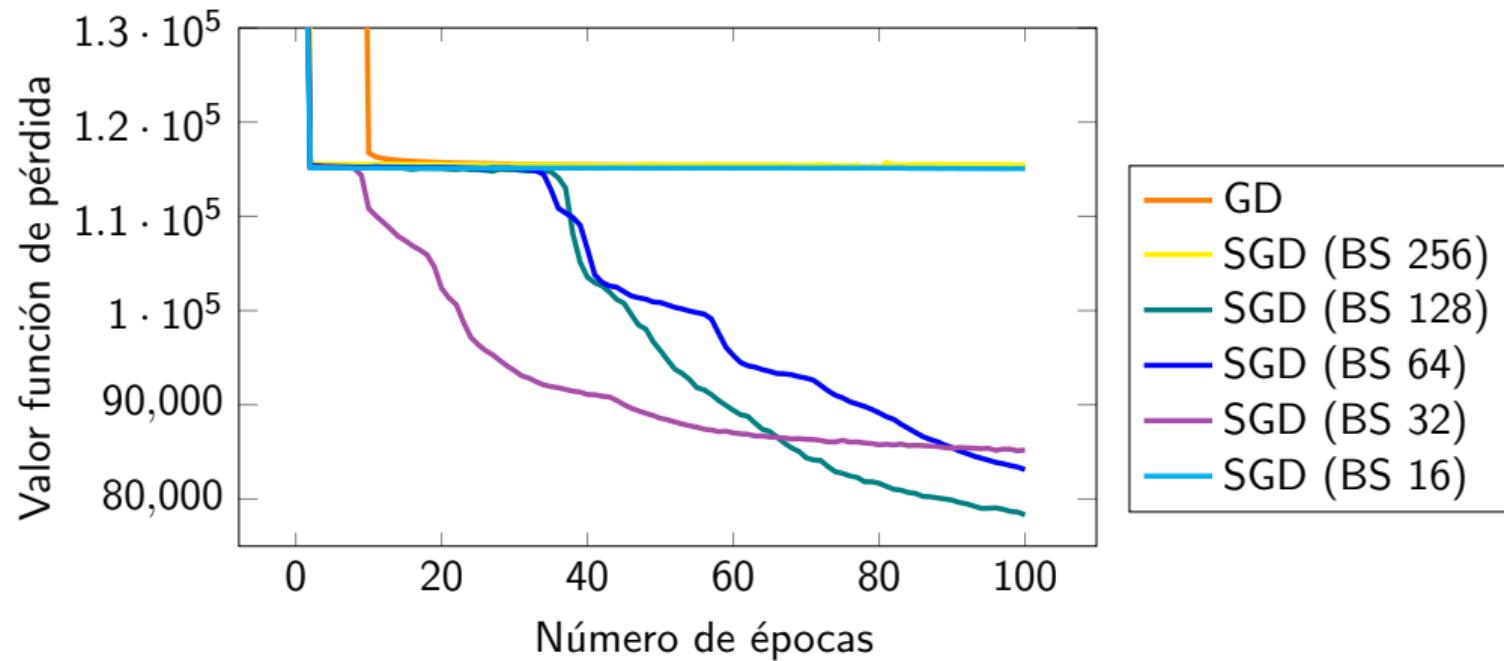
Repetimos el experimento usando distintos valores para el *batch size*.



Repetimos el experimento usando distintos valores para el *batch size*.



Repetimos el experimento usando distintos valores para el *batch size*.



Existe un trade-off:

- Mientras más chico el batch, más actualizaciones se hacen por episodio.
- Mientras más grande el batch, más parecido es $\nabla_{\theta}J(\theta_t; B_n)$ a $\nabla_{\theta}J(\theta_t)$.

Existe un trade-off:

- Mientras más chico el batch, más actualizaciones se hacen por episodio.
- Mientras más grande el batch, más parecido es $\nabla_{\theta}J(\theta_t; B_n)$ a $\nabla_{\theta}J(\theta_t)$.

En la práctica:

- Normalmente se usan batches de tamaños 32, 64 o 128.

Existe un trade-off:

- Mientras más chico el batch, más actualizaciones se hacen por episodio.
- Mientras más grande el batch, más parecido es $\nabla_{\theta}J(\theta_t; B_n)$ a $\nabla_{\theta}J(\theta_t)$.

En la práctica:

- Normalmente se usan batches de tamaños 32, 64 o 128.

¿Algo más que podamos hacer para mejorar el rendimiento de SGD?

Existe un trade-off:

- Mientras más chico el batch, más actualizaciones se hacen por episodio.
- Mientras más grande el batch, más parecido es $\nabla_{\theta}J(\theta_t; B_n)$ a $\nabla_{\theta}J(\theta_t)$.

En la práctica:

- Normalmente se usan batches de tamaños 32, 64 o 128.

¿Algo más que podamos hacer para mejorar el rendimiento de SGD?

Se han propuesto muchas mejoras a SGD.

Existe un trade-off:

- Mientras más chico el batch, más actualizaciones se hacen por episodio.
- Mientras más grande el batch, más parecido es $\nabla_{\theta}J(\theta_t; B_n)$ a $\nabla_{\theta}J(\theta_t)$.

En la práctica:

- Normalmente se usan batches de tamaños 32, 64 o 128.

¿Algo más que podamos hacer para mejorar el rendimiento de SGD?

Se han propuesto muchas mejoras a SGD. Por ejemplo, un problema que tiene SGD es que usa un learning rate fijo para todos los pesos. Esto es problemático dado que en algunos pesos los gradientes tienden a explotar o desbanecerse.

Existe un trade-off:

- Mientras más chico el batch, más actualizaciones se hacen por episodio.
- Mientras más grande el batch, más parecido es $\nabla_{\theta}J(\theta_t; B_n)$ a $\nabla_{\theta}J(\theta_t)$.

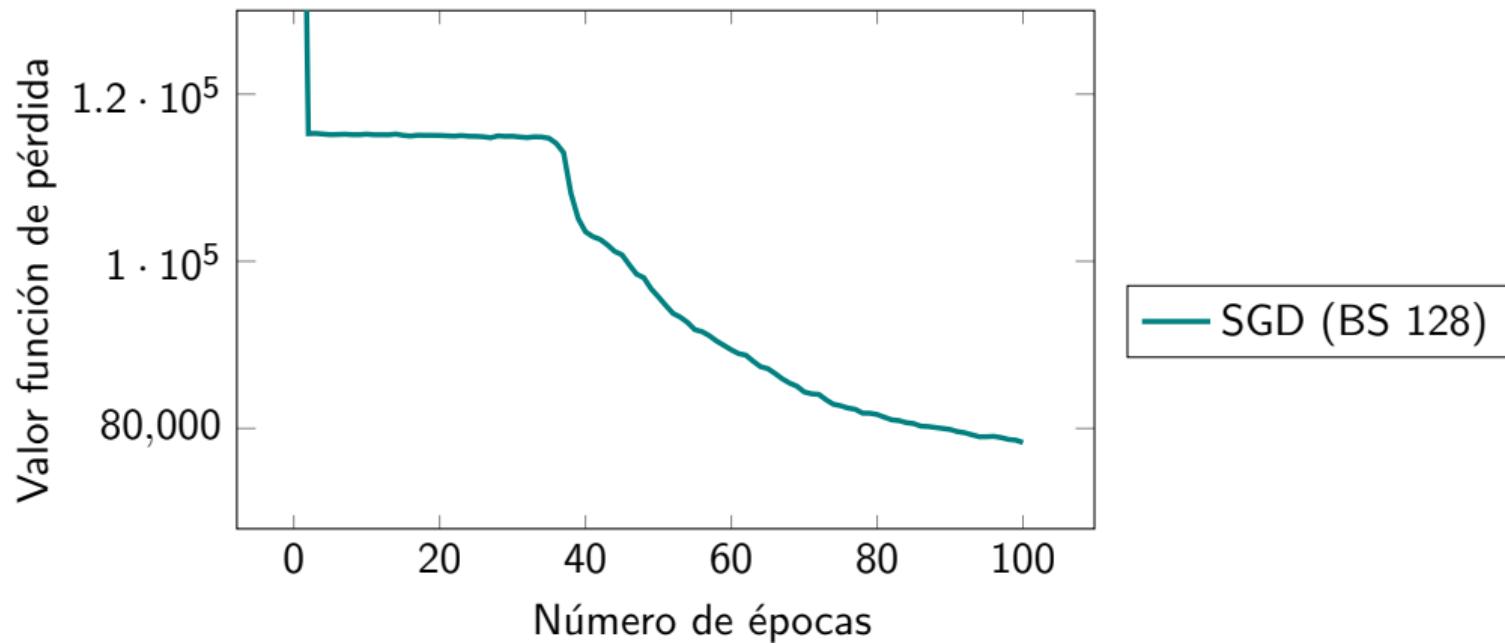
En la práctica:

- Normalmente se usan batches de tamaños 32, 64 o 128.

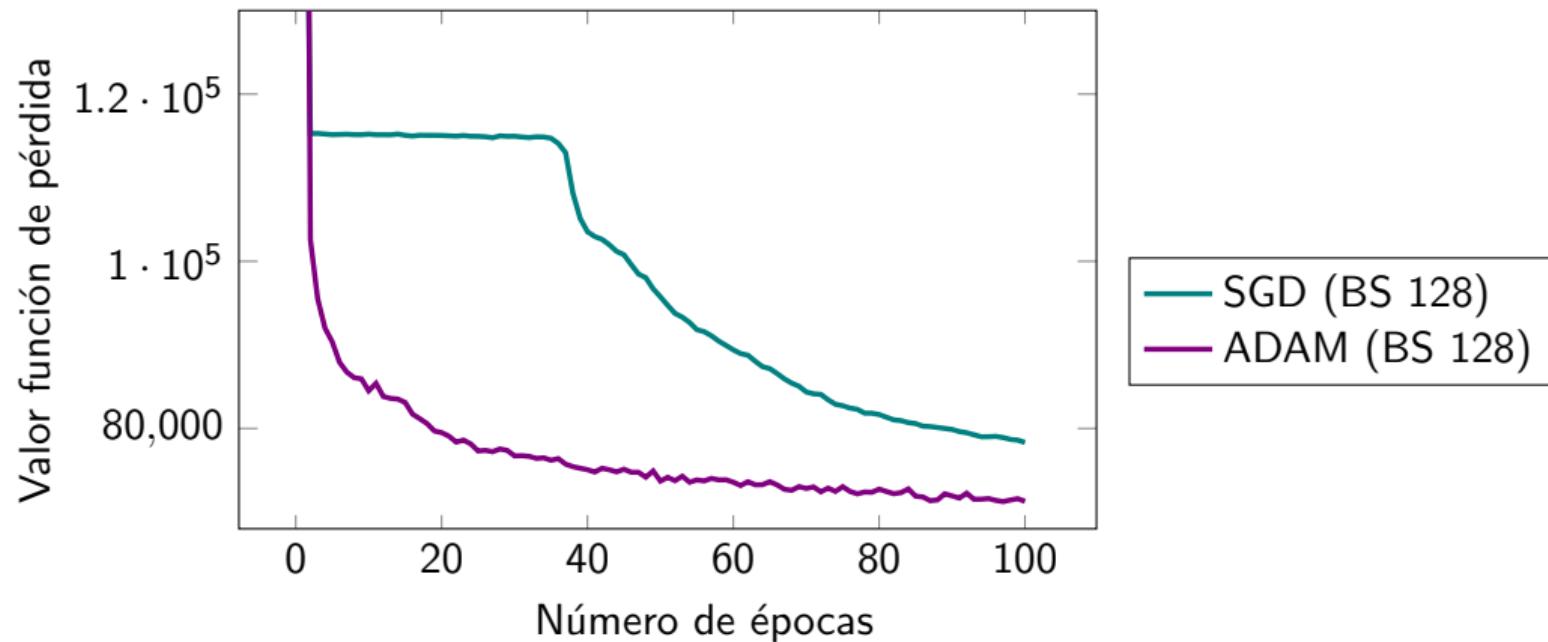
¿Algo más que podamos hacer para mejorar el rendimiento de SGD?

Se han propuesto muchas mejoras a SGD. Por ejemplo, un problema que tiene SGD es que usa un learning rate fijo para todos los pesos. Esto es problemático dado que en algunos pesos los gradientes tienden a explotar o desbanecerse. ADAM arregla esto ajustando los learning rates peso a peso de manera automática.

Repetimos el experimento usando ADAM en vez de SGD.

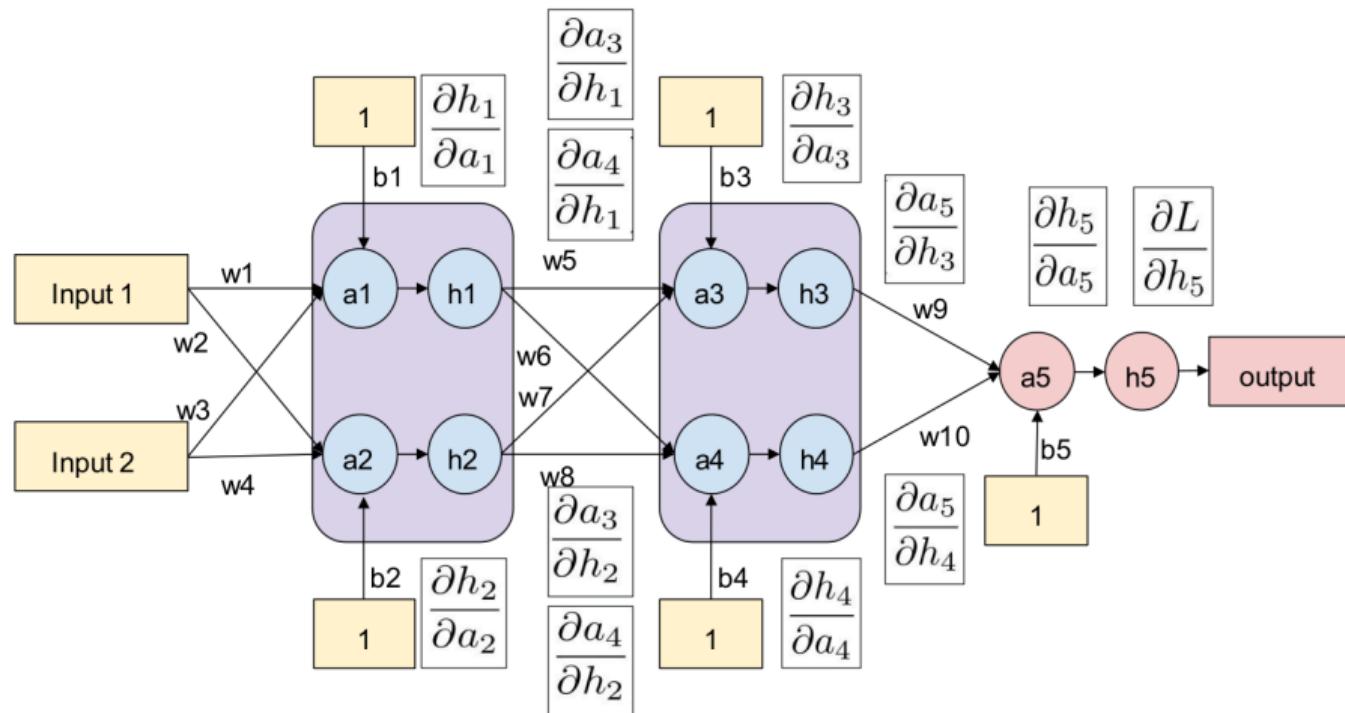


Repetimos el experimento usando ADAM en vez de SGD.



Existe una cosa más que podemos hacer para mejorar los resultados:

Existe una cosa más que podemos hacer para mejorar los resultados:



Existe una cosa más que podemos hacer para mejorar los resultados:

$$\nabla_{\theta} L(\theta) = \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial w_3} \\ \frac{\partial L}{\partial w_5} \\ \frac{\partial L}{\partial w_9} \end{bmatrix} = \begin{bmatrix} \frac{\partial L}{\partial h_5} \frac{\partial h_5}{\partial a_5} \left(\frac{\partial a_5}{\partial h_3} \frac{\partial h_3}{\partial a_3} \frac{\partial a_3}{\partial h_1} + \frac{\partial a_5}{\partial h_4} \frac{\partial h_4}{\partial a_4} \frac{\partial a_4}{\partial h_1} \right) \frac{\partial h_1}{\partial a_1} x_1 \\ \frac{\partial L}{\partial h_5} \frac{\partial h_5}{\partial a_5} \left(\frac{\partial a_5}{\partial h_3} \frac{\partial h_3}{\partial a_3} \frac{\partial a_3}{\partial h_1} + \frac{\partial a_5}{\partial h_4} \frac{\partial h_4}{\partial a_4} \frac{\partial a_4}{\partial h_1} \right) \frac{\partial h_1}{\partial a_1} x_2 \\ \frac{\partial L}{\partial h_5} \frac{\partial h_5}{\partial a_5} \frac{\partial a_5}{\partial h_3} \frac{\partial h_3}{\partial a_3} h_1 \\ \frac{\partial L}{\partial h_5} \frac{\partial h_5}{\partial a_5} h_3 \end{bmatrix}$$

Existe una cosa más que podemos hacer para mejorar los resultados:

$$\nabla_{\theta} L(\theta) = \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial w_3} \\ \frac{\partial L}{\partial w_5} \\ \frac{\partial L}{\partial w_9} \end{bmatrix} = \begin{bmatrix} \frac{\partial L}{\partial h_5} \frac{\partial h_5}{\partial a_5} \left(\frac{\partial a_5}{\partial h_3} \frac{\partial h_3}{\partial a_3} \frac{\partial a_3}{\partial h_1} + \frac{\partial a_5}{\partial h_4} \frac{\partial h_4}{\partial a_4} \frac{\partial a_4}{\partial h_1} \right) \frac{\partial h_1}{\partial a_1} \textcolor{red}{x_1} \\ \frac{\partial L}{\partial h_5} \frac{\partial h_5}{\partial a_5} \left(\frac{\partial a_5}{\partial h_3} \frac{\partial h_3}{\partial a_3} \frac{\partial a_3}{\partial h_1} + \frac{\partial a_5}{\partial h_4} \frac{\partial h_4}{\partial a_4} \frac{\partial a_4}{\partial h_1} \right) \frac{\partial h_1}{\partial a_1} \textcolor{red}{x_2} \\ \frac{\partial L}{\partial h_5} \frac{\partial h_5}{\partial a_5} \frac{\partial a_5}{\partial h_3} \frac{\partial h_3}{\partial a_3} \textcolor{red}{h_1} \\ \frac{\partial L}{\partial h_5} \frac{\partial h_5}{\partial a_5} \textcolor{red}{h_3} \end{bmatrix}$$

Los gradientes de cada peso están multiplicados por el input de ese peso.

Consideremos los pesos w_1 y w_3 :

$$\nabla_{\theta} L(\theta) = \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial w_3} \end{bmatrix} = \begin{bmatrix} \frac{\partial L}{\partial h_5} \frac{\partial h_5}{\partial a_5} \left(\frac{\partial a_5}{\partial h_3} \frac{\partial h_3}{\partial a_3} \frac{\partial a_3}{\partial h_1} + \frac{\partial a_5}{\partial h_4} \frac{\partial h_4}{\partial a_4} \frac{\partial a_4}{\partial h_1} \right) \frac{\partial h_1}{\partial a_1} \color{red}{x_1} \\ \frac{\partial L}{\partial h_5} \frac{\partial h_5}{\partial a_5} \left(\frac{\partial a_5}{\partial h_3} \frac{\partial h_3}{\partial a_3} \frac{\partial a_3}{\partial h_1} + \frac{\partial a_5}{\partial h_4} \frac{\partial h_4}{\partial a_4} \frac{\partial a_4}{\partial h_1} \right) \frac{\partial h_1}{\partial a_1} \color{red}{x_2} \end{bmatrix}$$

Consideremos los pesos w_1 y w_3 :

$$\nabla_{\theta} L(\theta) = \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial w_3} \end{bmatrix} = \begin{bmatrix} \frac{\partial L}{\partial h_5} \frac{\partial h_5}{\partial a_5} \left(\frac{\partial a_5}{\partial h_3} \frac{\partial h_3}{\partial a_3} \frac{\partial a_3}{\partial h_1} + \frac{\partial a_5}{\partial h_4} \frac{\partial h_4}{\partial a_4} \frac{\partial a_4}{\partial h_1} \right) \frac{\partial h_1}{\partial a_1} x_1 \\ \frac{\partial L}{\partial h_5} \frac{\partial h_5}{\partial a_5} \left(\frac{\partial a_5}{\partial h_3} \frac{\partial h_3}{\partial a_3} \frac{\partial a_3}{\partial h_1} + \frac{\partial a_5}{\partial h_4} \frac{\partial h_4}{\partial a_4} \frac{\partial a_4}{\partial h_1} \right) \frac{\partial h_1}{\partial a_1} x_2 \end{bmatrix}$$

Si $x_1 \in [1000, 10000]$ mientras que $x_2 \in [0, 1]$, entonces los gradientes de w_1 serán hasta 10000 veces más grandes que los gradientes de w_3 .

Consideremos los pesos w_1 y w_3 :

$$\nabla_{\theta} L(\theta) = \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial w_3} \end{bmatrix} = \begin{bmatrix} \frac{\partial L}{\partial h_5} \frac{\partial h_5}{\partial a_5} \left(\frac{\partial a_5}{\partial h_3} \frac{\partial h_3}{\partial a_3} \frac{\partial a_3}{\partial h_1} + \frac{\partial a_5}{\partial h_4} \frac{\partial h_4}{\partial a_4} \frac{\partial a_4}{\partial h_1} \right) \frac{\partial h_1}{\partial a_1} x_1 \\ \frac{\partial L}{\partial h_5} \frac{\partial h_5}{\partial a_5} \left(\frac{\partial a_5}{\partial h_3} \frac{\partial h_3}{\partial a_3} \frac{\partial a_3}{\partial h_1} + \frac{\partial a_5}{\partial h_4} \frac{\partial h_4}{\partial a_4} \frac{\partial a_4}{\partial h_1} \right) \frac{\partial h_1}{\partial a_1} x_2 \end{bmatrix}$$

Si $x_1 \in [1000, 10000]$ mientras que $x_2 \in [0, 1]$, entonces los gradientes de w_1 serán hasta 10000 veces más grandes que los gradientes de w_3 .

Esto causa que para SGD, x_1 sea entre 1000 y 10000 veces más importante que x_2 , a pesar de que x_2 podría ser tan importante como x_1 para predecir la clase correcta.

Consideremos los pesos w_1 y w_3 :

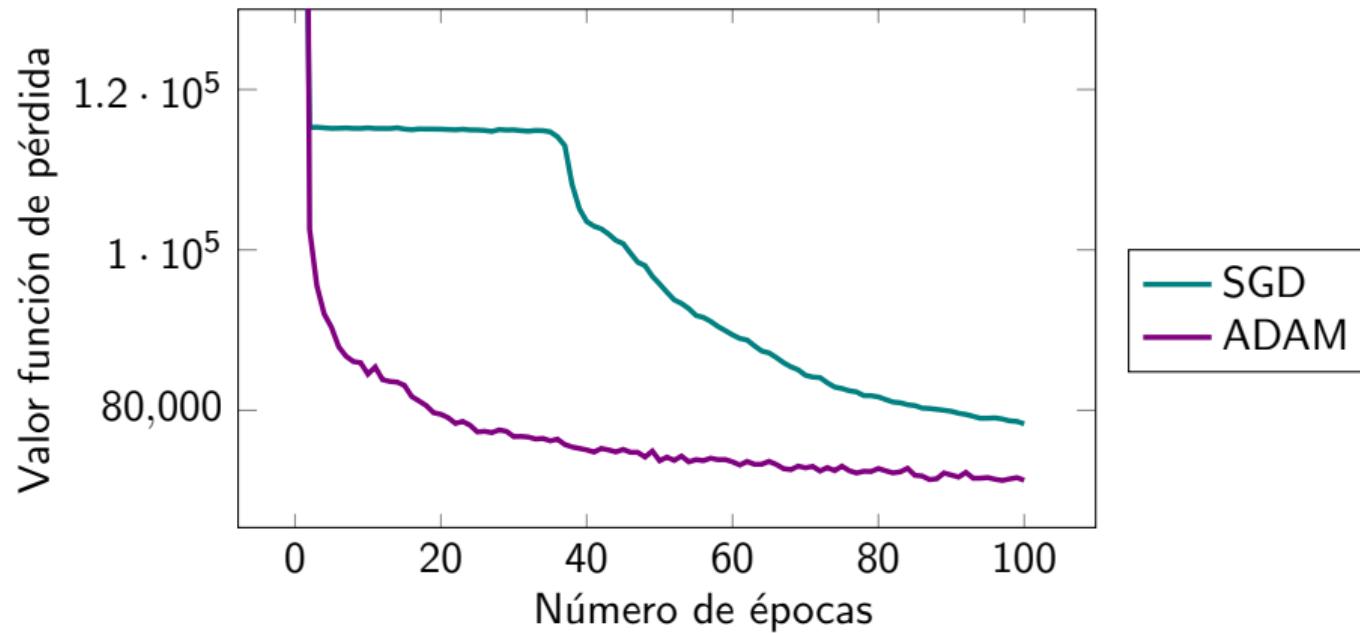
$$\nabla_{\theta} L(\theta) = \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial w_3} \end{bmatrix} = \begin{bmatrix} \frac{\partial L}{\partial h_5} \frac{\partial h_5}{\partial a_5} \left(\frac{\partial a_5}{\partial h_3} \frac{\partial h_3}{\partial a_3} \frac{\partial a_3}{\partial h_1} + \frac{\partial a_5}{\partial h_4} \frac{\partial h_4}{\partial a_4} \frac{\partial a_4}{\partial h_1} \right) \frac{\partial h_1}{\partial a_1} x_1 \\ \frac{\partial L}{\partial h_5} \frac{\partial h_5}{\partial a_5} \left(\frac{\partial a_5}{\partial h_3} \frac{\partial h_3}{\partial a_3} \frac{\partial a_3}{\partial h_1} + \frac{\partial a_5}{\partial h_4} \frac{\partial h_4}{\partial a_4} \frac{\partial a_4}{\partial h_1} \right) \frac{\partial h_1}{\partial a_1} x_2 \end{bmatrix}$$

Si $x_1 \in [1000, 10000]$ mientras que $x_2 \in [0, 1]$, entonces los gradientes de w_1 serán hasta 10000 veces más grandes que los gradientes de w_3 .

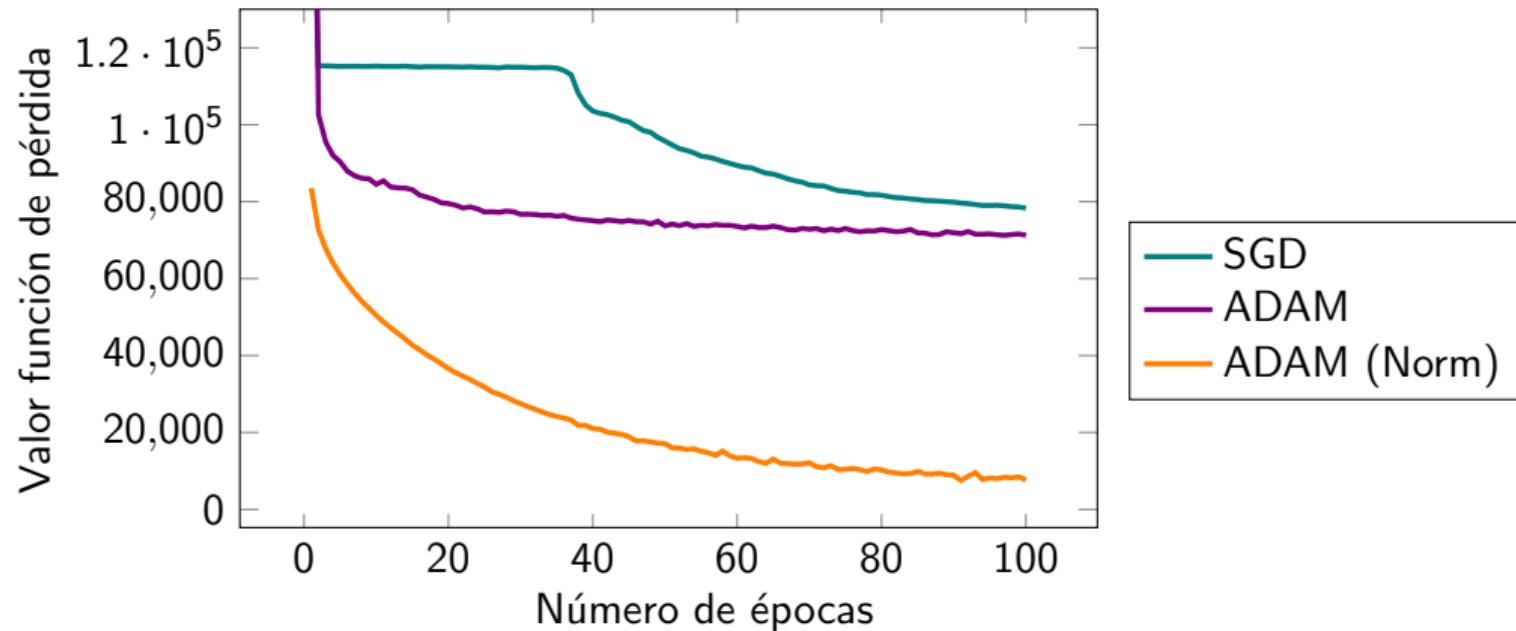
Esto causa que para SGD, x_1 sea entre 1000 y 10000 veces más importante que x_2 , a pesar de que x_2 podría ser tan importante como x_1 para predecir la clase correcta.

Solución: Normalizar los datos para que $x_i \in [-1, 1]$.

Repetimos el experimento normalizando los datos.



Repetimos el experimento normalizando los datos.



Finalmente, notar que aumentar el tamaño de la red debería permitirnos encontrar soluciones con menor pérdida.

Finalmente, notar que aumentar el tamaño de la red debería permitirnos encontrar soluciones con menor pérdida.

Modelo actual:

- Capa de 120 neuronas relu.
- Capa de 84 neuronas relu.
- Capa de 10 neuronas softmax.

Finalmente, notar que aumentar el tamaño de la red debería permitirnos encontrar soluciones con menor pérdida.

Modelo actual:

- Capa de 120 neuronas relu.
- Capa de 84 neuronas relu.
- Capa de 10 neuronas softmax.

Modelo más neuronas:

- Capa de 240 neuronas relu.
- Capa de 84 neuronas relu.
- Capa de 10 neuronas softmax.

Finalmente, notar que aumentar el tamaño de la red debería permitirnos encontrar soluciones con menor pérdida.

Modelo actual:

- Capa de 120 neuronas relu.
- Capa de 84 neuronas relu.
- Capa de 10 neuronas softmax.

Modelo más capas:

- Capa de 120 neuronas relu.
- Capa de 120 neuronas relu.
- Capa de 84 neuronas relu.
- Capa de 10 neuronas softmax.

Modelo más neuronas:

- Capa de 240 neuronas relu.
- Capa de 84 neuronas relu.
- Capa de 10 neuronas softmax.

Finalmente, notar que aumentar el tamaño de la red debería permitirnos encontrar soluciones con menor pérdida.

Modelo actual:

- Capa de 120 neuronas relu.
- Capa de 84 neuronas relu.
- Capa de 10 neuronas softmax.

Modelo más capas:

- Capa de 120 neuronas relu.
- Capa de 120 neuronas relu.
- Capa de 84 neuronas relu.
- Capa de 10 neuronas softmax.

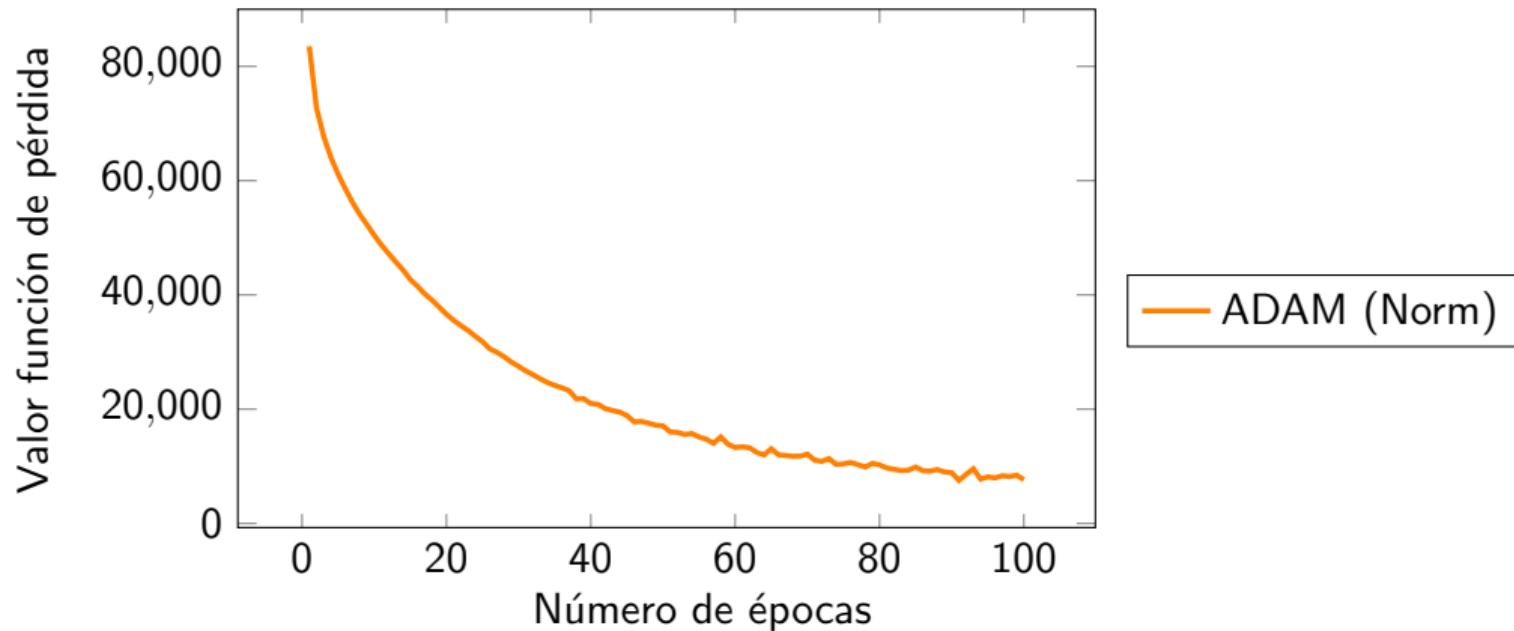
Modelo más neuronas:

- Capa de 240 neuronas relu.
- Capa de 84 neuronas relu.
- Capa de 10 neuronas softmax.

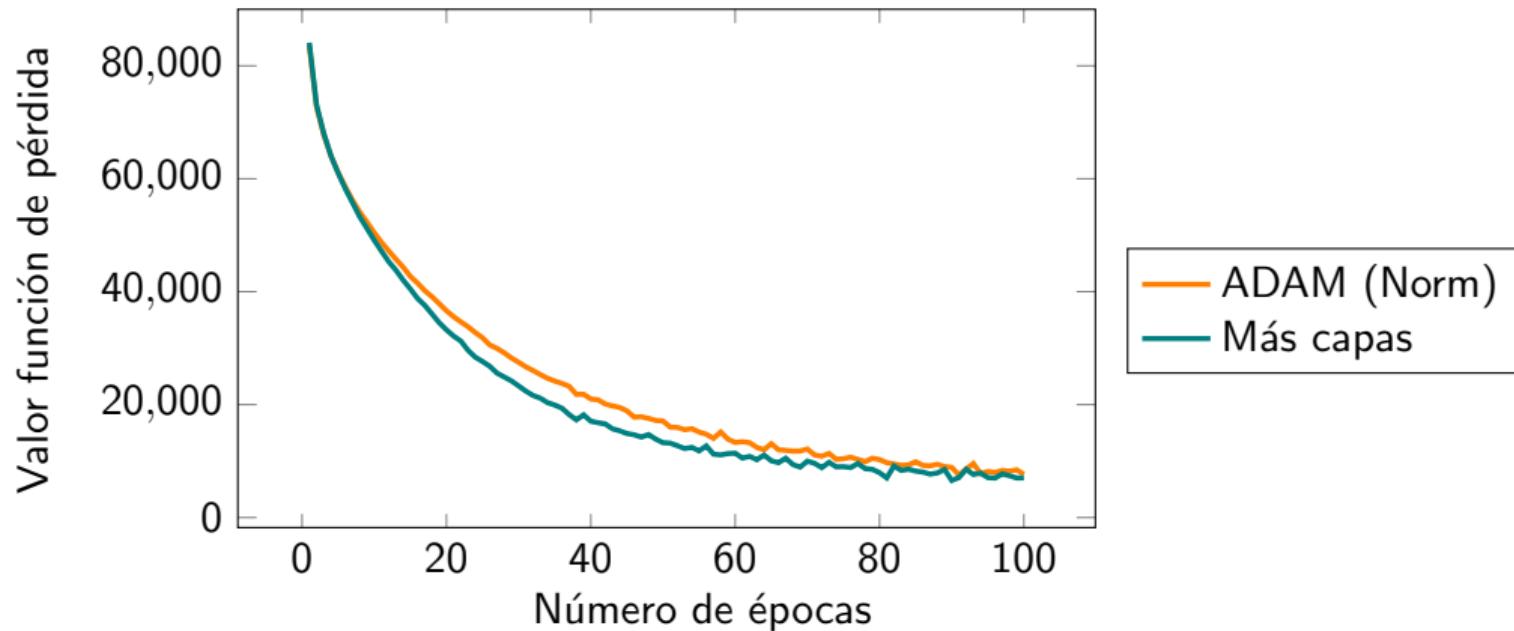
Modelo más capas y neuronas:

- Capa de 480 neuronas relu.
- Capa de 240 neuronas relu.
- Capa de 120 neuronas relu.
- Capa de 10 neuronas softmax.

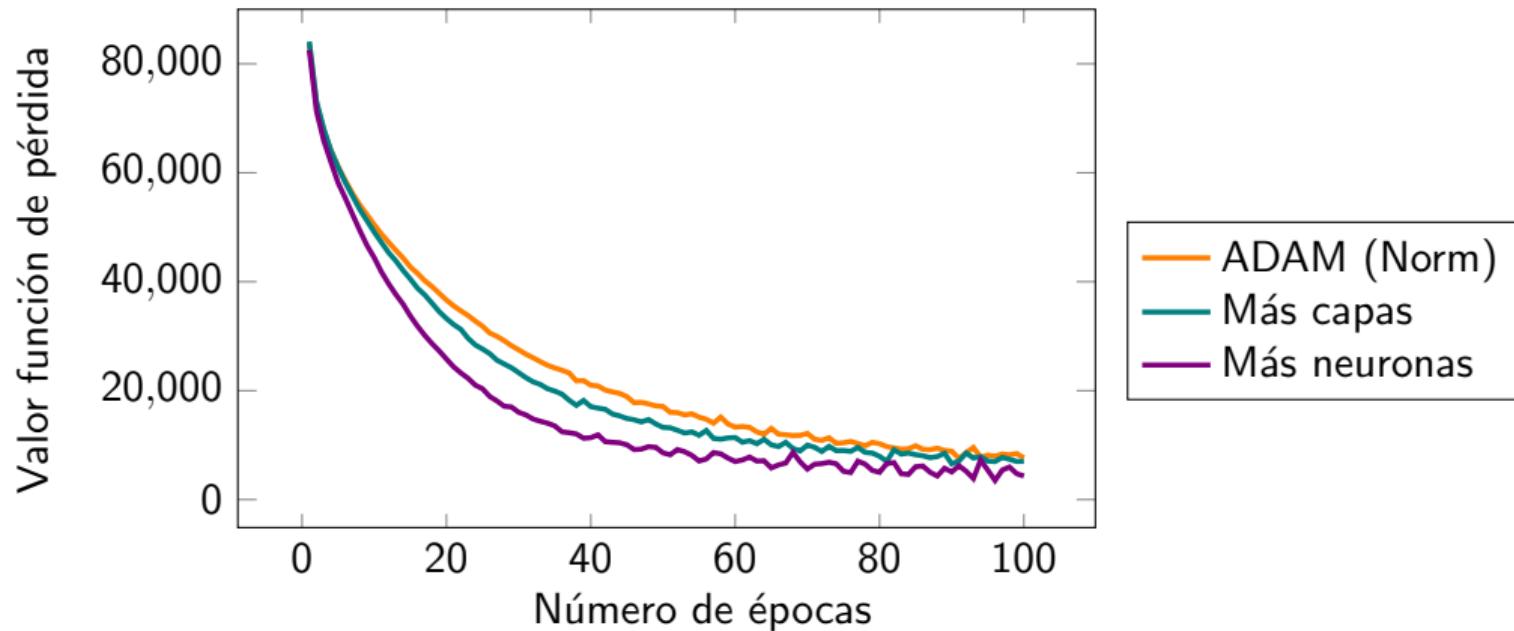
Repetimos el experimento normalizando los datos.



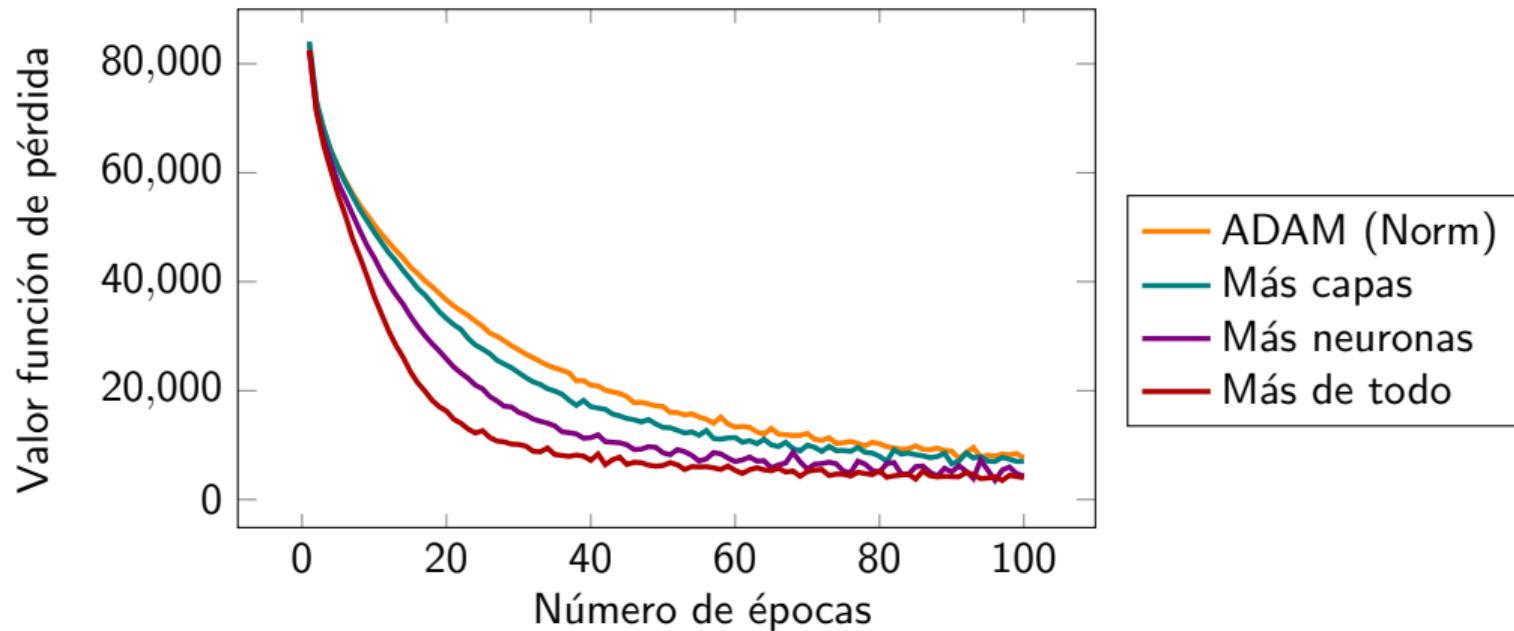
Repetimos el experimento normalizando los datos.



Repetimos el experimento normalizando los datos.



Repetimos el experimento normalizando los datos.



Para entrenar redes con muchos datos:

Para entrenar redes con muchos datos:

- Hay que usar descenso de gradiente estocástico:
 - Dividir los datos de entrenamiento en batches.
 - Actualizar los pesos usando un batch a la vez.
 - Se suele obtener un buen rendimiento con batches de 32, 64 y 128 ejemplos.

Para entrenar redes con muchos datos:

- Hay que usar descenso de gradiente estocástico:
 - Dividir los datos de entrenamiento en batches.
 - Actualizar los pesos usando un batch a la vez.
 - Se suele obtener un buen rendimiento con batches de 32, 64 y 128 ejemplos.
- Es recomendable usar una versión mejorada de SGD:
 - SGD con momentum.
 - RMSProp.
 - ADAM.

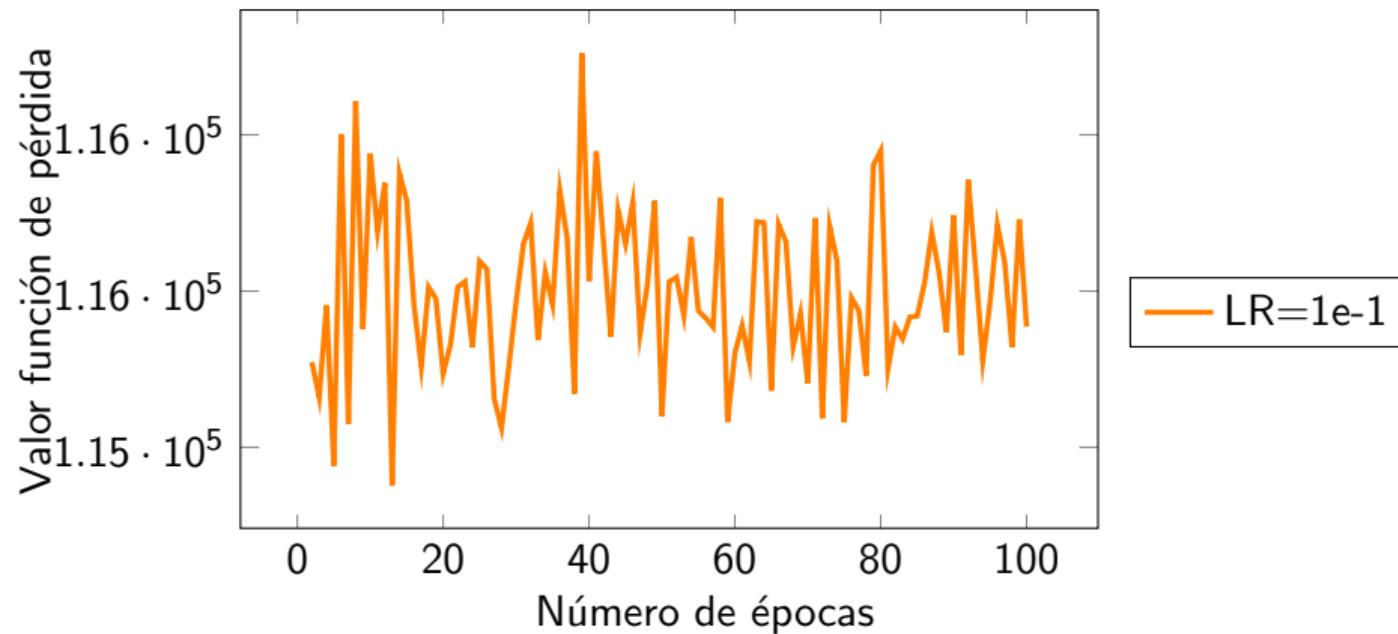
Para entrenar redes con muchos datos:

- Hay que usar descenso de gradiente estocástico:
 - Dividir los datos de entrenamiento en batches.
 - Actualizar los pesos usando un batch a la vez.
 - Se suele obtener un buen rendimiento con batches de 32, 64 y 128 ejemplos.
- Es recomendable usar una versión mejorada de SGD:
 - SGD con momentum.
 - RMSProp.
 - ADAM.
- Es **muy** importante que normalicemos los datos.

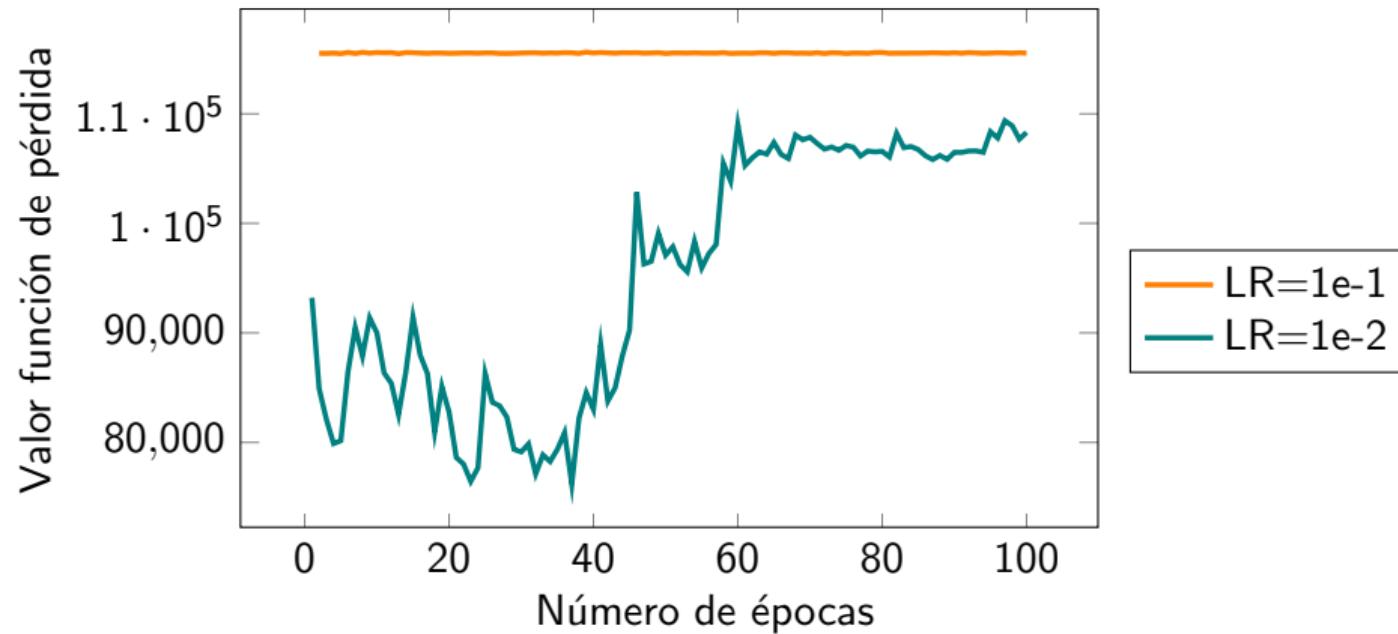
Para entrenar redes con muchos datos:

- Hay que usar descenso de gradiente estocástico:
 - Dividir los datos de entrenamiento en batches.
 - Actualizar los pesos usando un batch a la vez.
 - Se suele obtener un buen rendimiento con batches de 32, 64 y 128 ejemplos.
- Es recomendable usar una versión mejorada de SGD:
 - SGD con momentum.
 - RMSProp.
 - ADAM.
- Es **muy** importante que normalicemos los datos.
- Si la pérdida sigue siendo alta, intenta lo siguiente:
 - Agrandar la red.
 - Cambiar el learning rate.

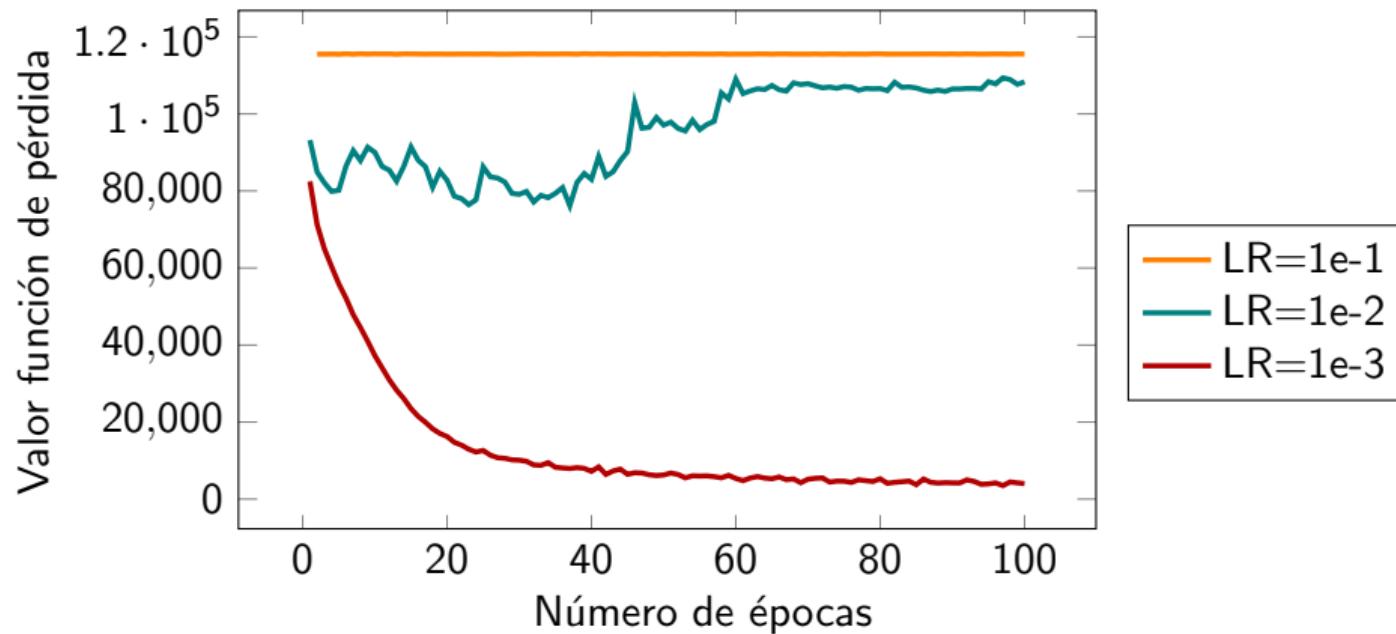
La importancia del learning rate.



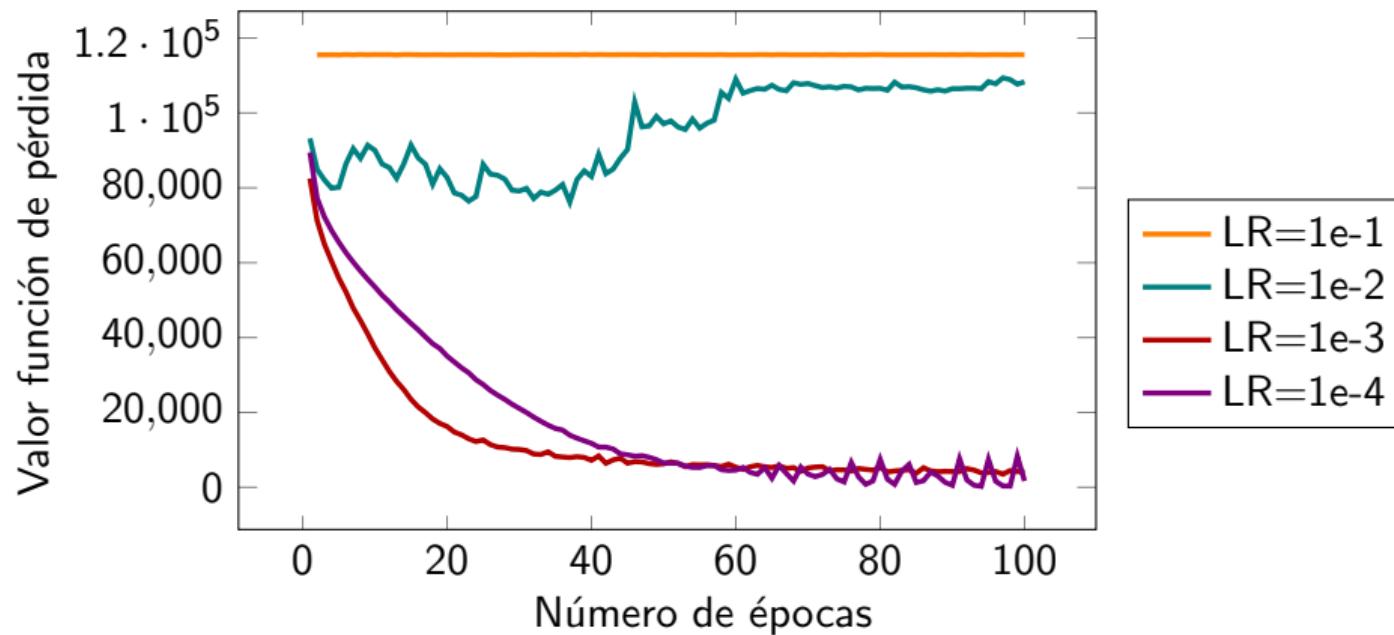
La importancia del learning rate.



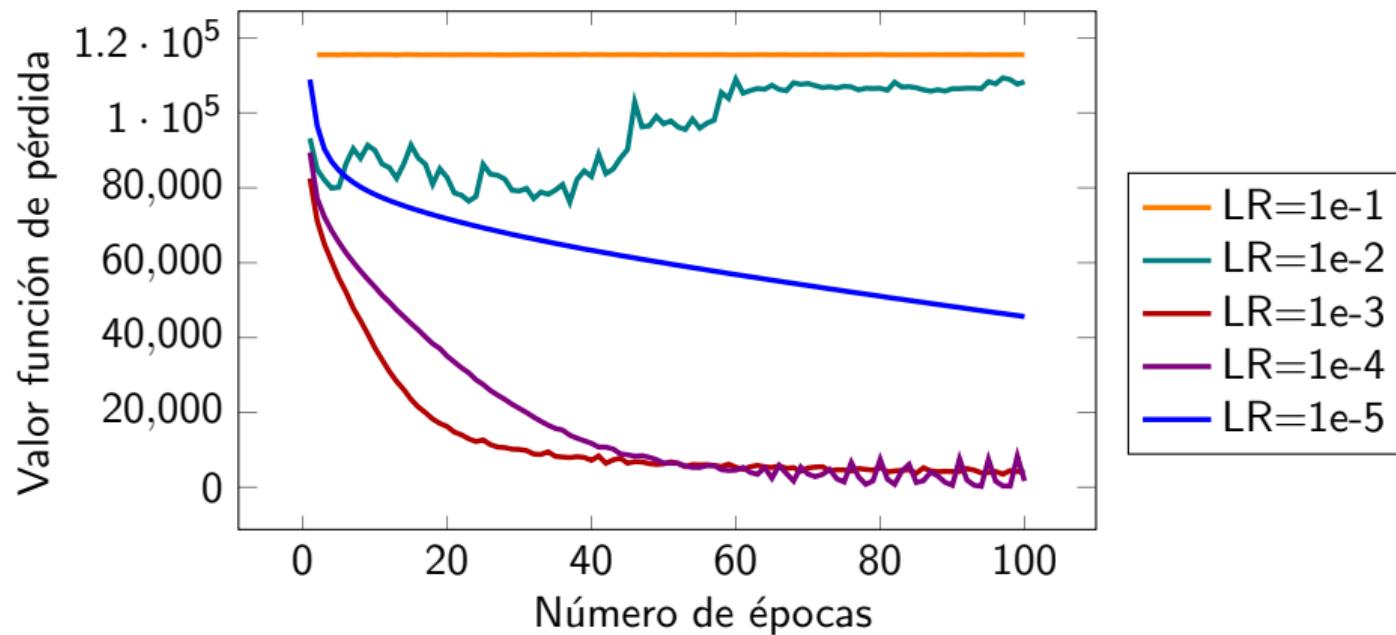
La importancia del learning rate.



La importancia del learning rate.



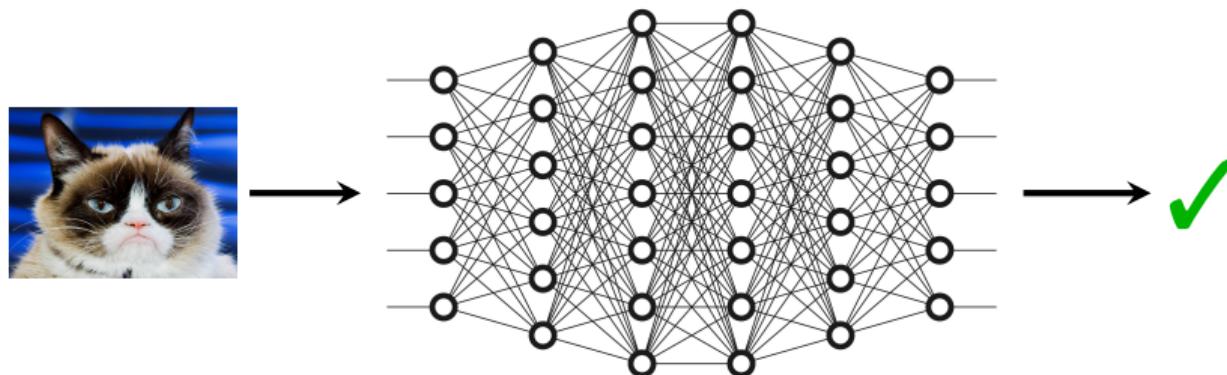
La importancia del learning rate.



Generalización

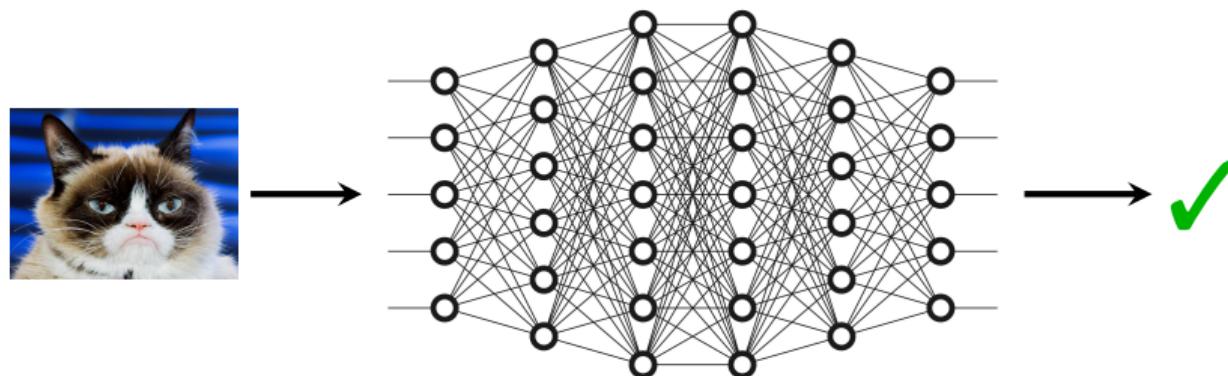
Generalización

La razón por la que usamos deep learning es porque generaliza muy bien.



Generalización

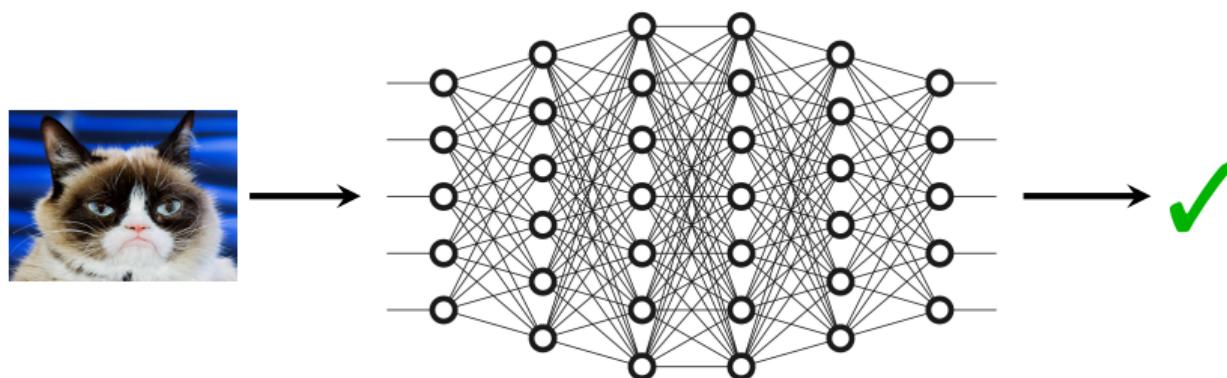
La razón por la que usamos deep learning es porque generaliza muy bien.



¿Pero qué significa esto?

Generalización

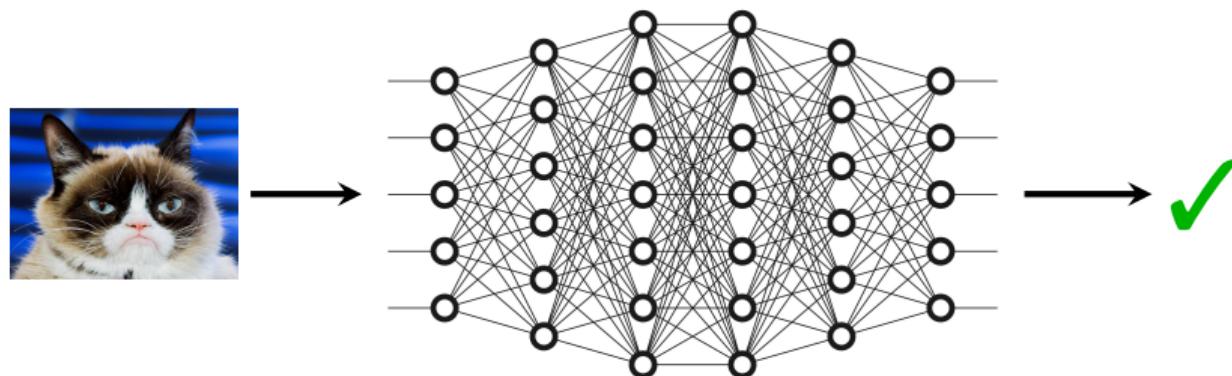
La razón por la que usamos deep learning es porque generaliza muy bien.



¿Pero qué significa esto? ¿Es deep learning todo poderoso y mágicamente funciona?

Generalización

La razón por la que usamos deep learning es porque generaliza muy bien.



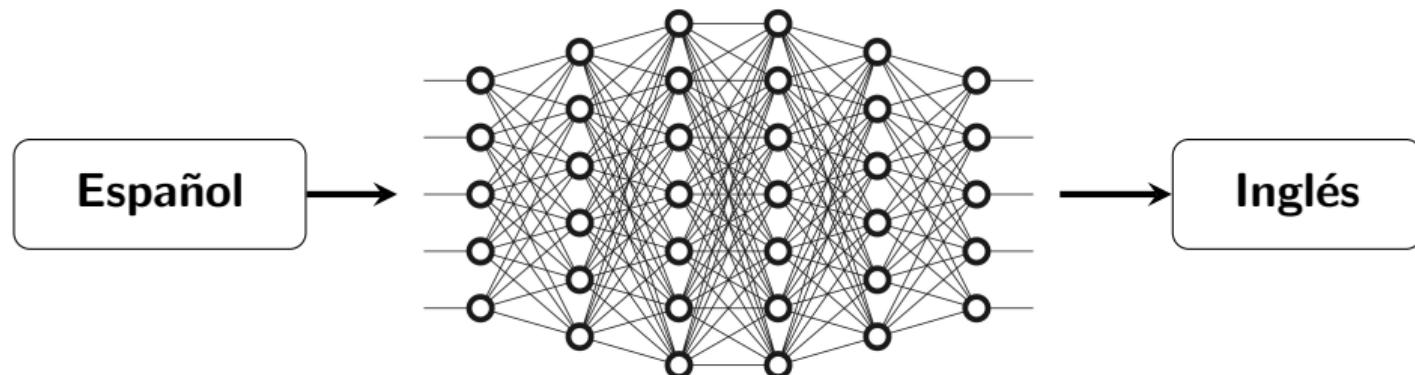
¿Pero qué significa esto? ¿Es deep learning todo poderoso y mágicamente funciona? ¿o existen limitaciones sobre lo que deep learning puede o no hacer?

Generalización

Primero que nada, deep learning **no** es magia.

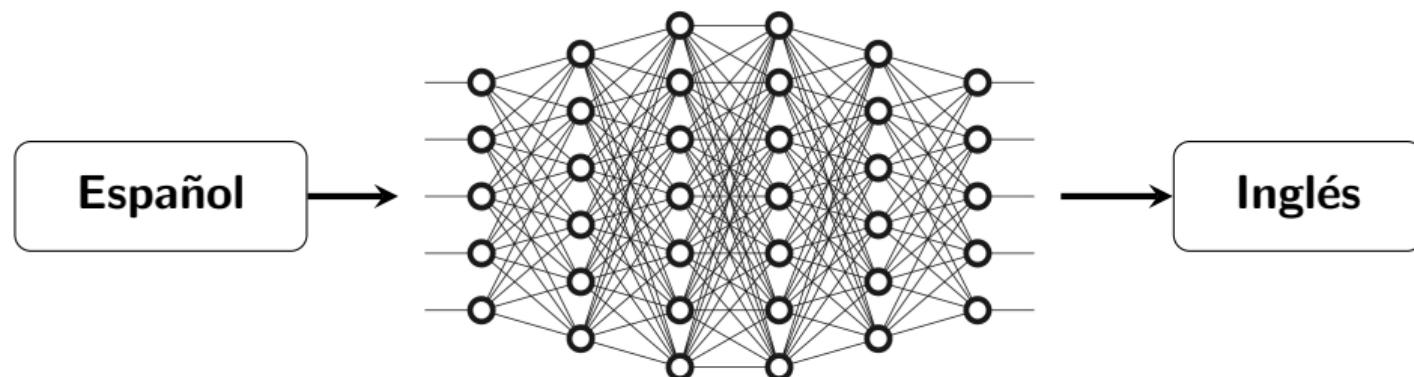
Generalización

Primero que nada, deep learning **no** es magia.



Generalización

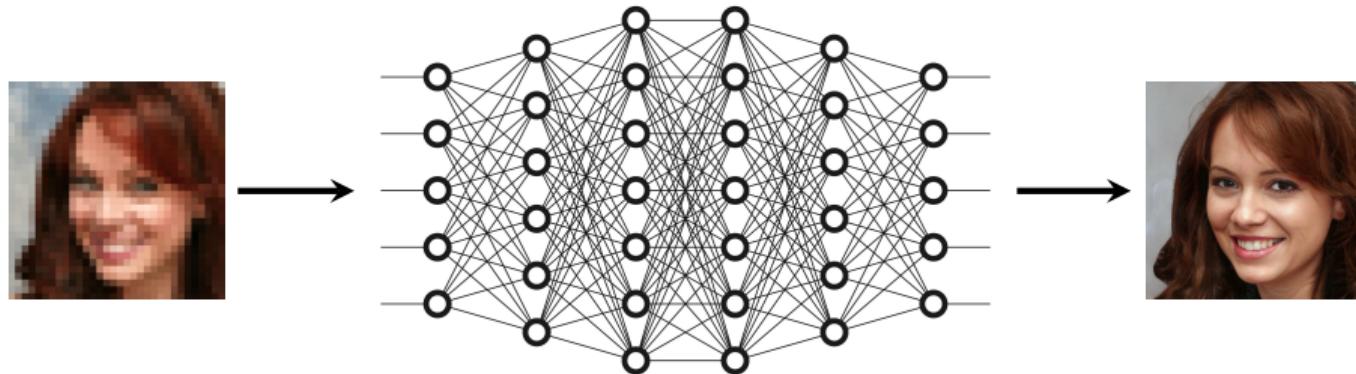
Primero que nada, deep learning **no** es magia.



Si le damos una oración en chino **no** va a funcionar.

Generalización

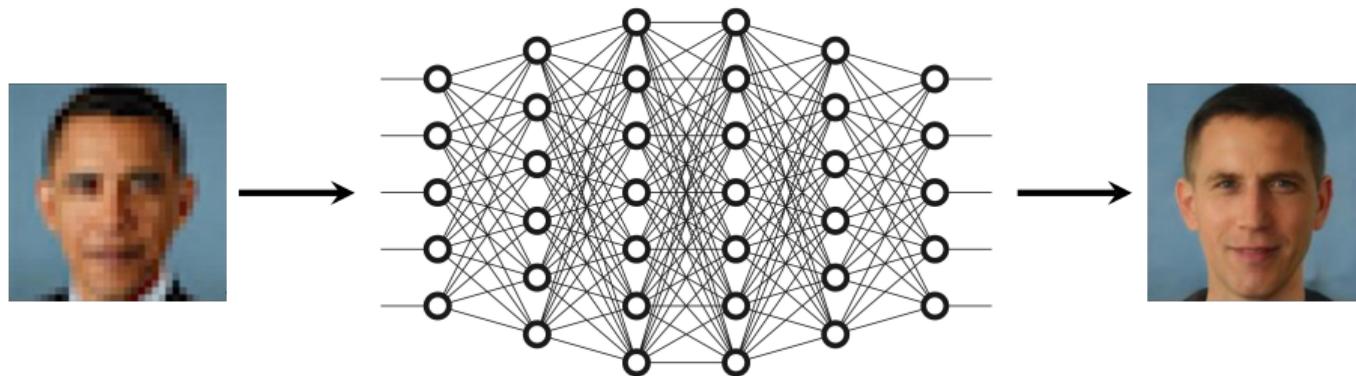
Segundo, deep learning muchas veces generaliza mal.³



³ "PULSE: Self-Supervised Photo Upsampling via Latent Space Exploration of Generative Models" by Menon et al. (CVPR20).

Generalización

Segundo, deep learning muchas veces generaliza mal.³



³ "PULSE: Self-Supervised Photo Upsampling via Latent Space Exploration of Generative Models" by Menon et al. (CVPR20).

Generalización

Segundo, deep learning muchas veces generaliza mal.

Generalización

¿Cómo podemos estudiar generalización en deep learning?

Generalización

¿Cómo podemos estudiar generalización en deep learning?

- Lo primero es ser capaz de **medir** generalización.

Generalización

¿Cómo podemos estudiar generalización en deep learning?

- Lo primero es ser capaz de **medir** generalización.

Para medir la generalización de nuestro modelo, lo ideal sería salir al mundo a testearlo en ejemplos nuevos... pero eso suele ser costoso.

Generalización

¿Cómo podemos estudiar generalización en deep learning?

- Lo primero es ser capaz de **medir** generalización.

Para medir la generalización de nuestro modelo, lo ideal sería salir al mundo a testearlo en ejemplos nuevos... pero eso suele ser costoso.

En la práctica, separamos nuestros datos en tres grupos:

- **Train:** Datos que usamos para entrenar el modelo.
- **Test:** Datos que usamos para testear la generalización del modelo. Es crítico que estos datos **no** se usen durante el entrenamiento.
- **Validación:** Datos que usamos para tunear algunos hiperparámetros.

Generalización

¿Cómo podemos estudiar generalización en deep learning?

- Lo primero es ser capaz de **medir** generalización.

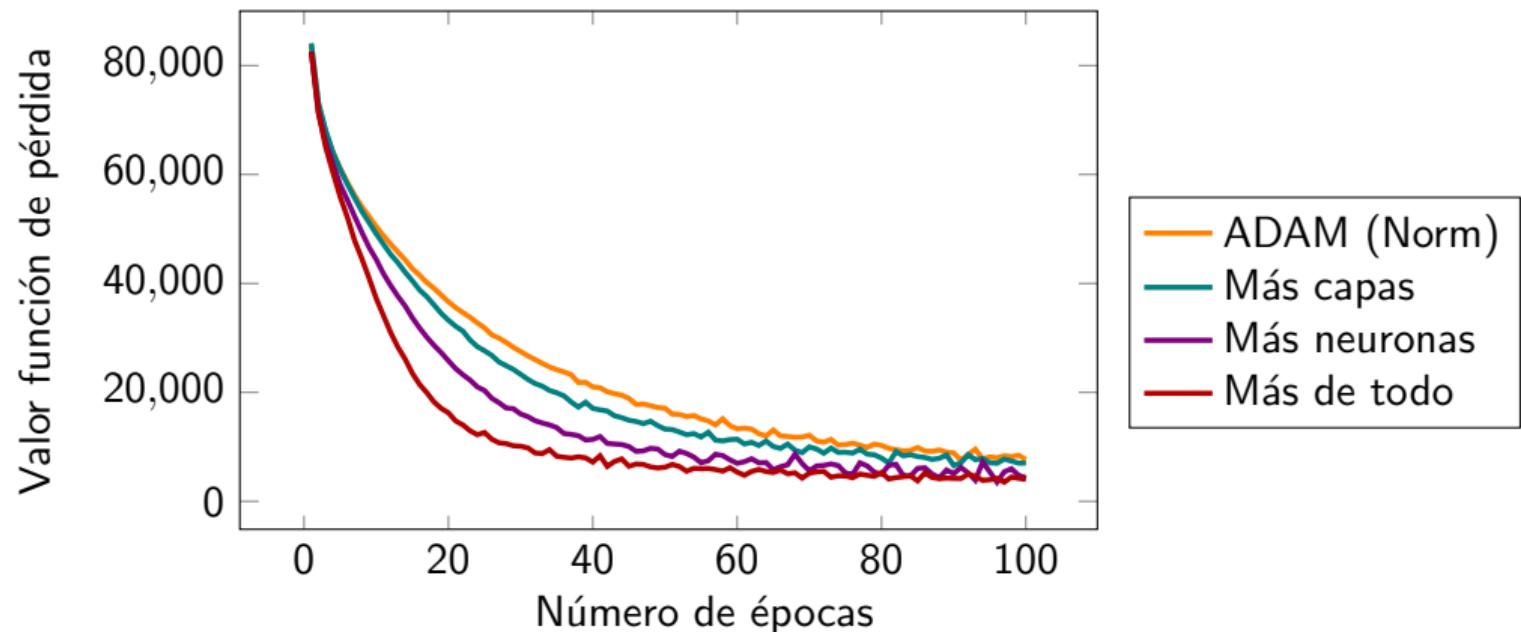
Para medir la generalización de nuestro modelo, lo ideal sería salir al mundo a testearlo en ejemplos nuevos... pero eso suele ser costoso.

En la práctica, separamos nuestros datos en tres grupos:

- **Train:** Datos que usamos para entrenar el modelo.
- **Test:** Datos que usamos para testear la generalización del modelo. Es crítico que estos datos **no** se usen durante el entrenamiento.
- **Validación:** Datos que usamos para tunear algunos hiperparámetros.

... A todo esto, qué tan bien generalizaba nuestra red entrenada en CIFAR-10?

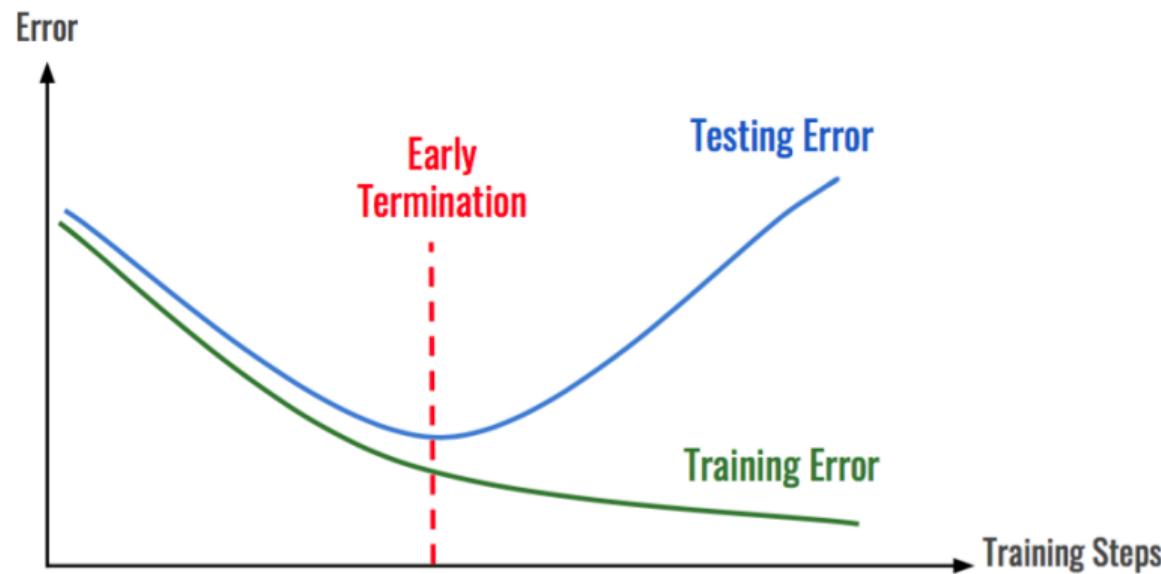
Generalización



Modelo más grande: 97% train pero 52% test :(

Generalización

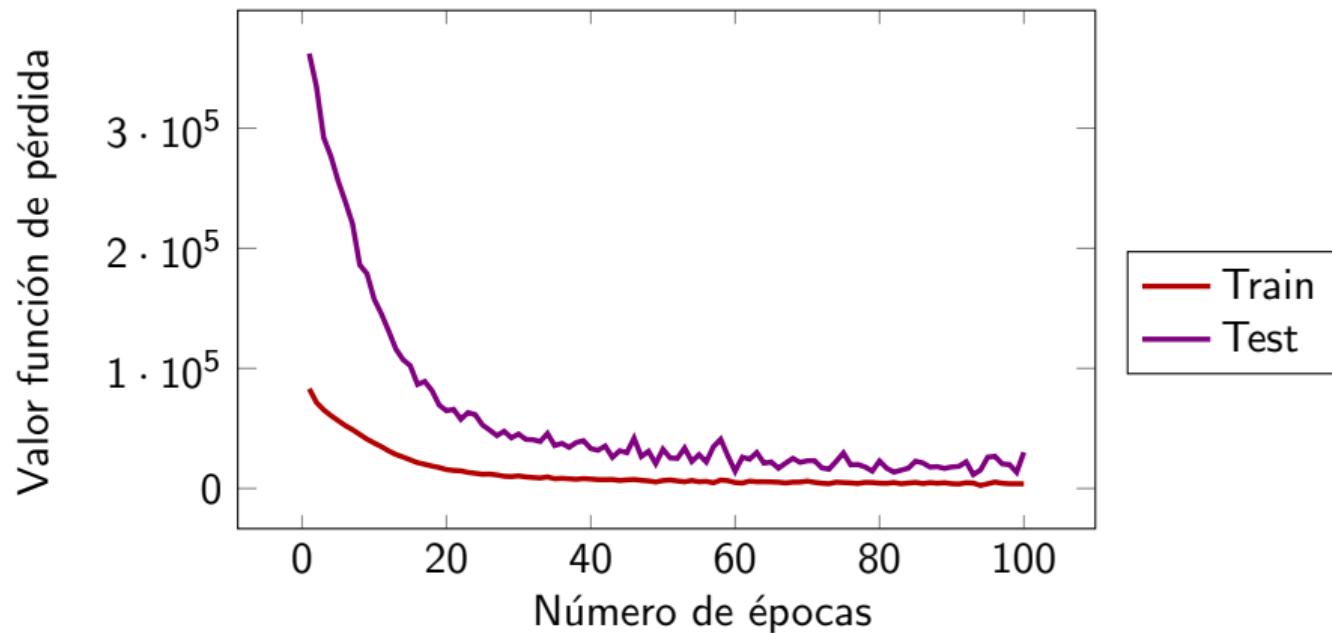
Les presento a overfitting 😱



Tener una baja pérdida en entrenamiento **no implica** generalizar bien.

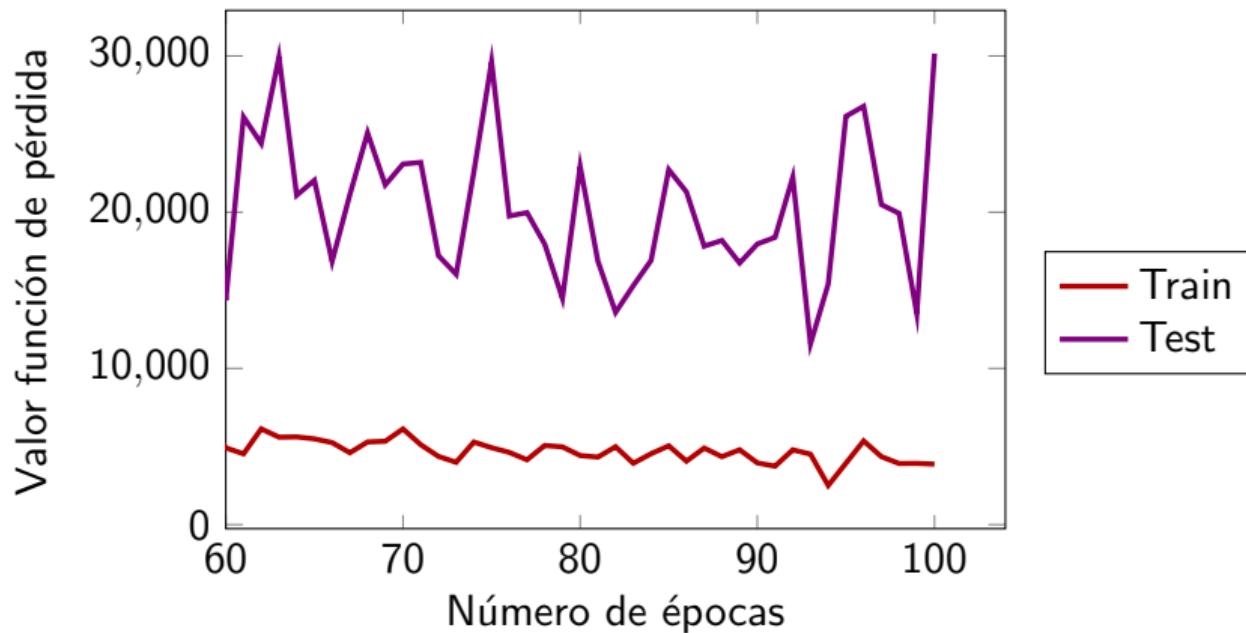
Generalización

En la práctica, el comportamiento es más errático que en el gráfico anterior.



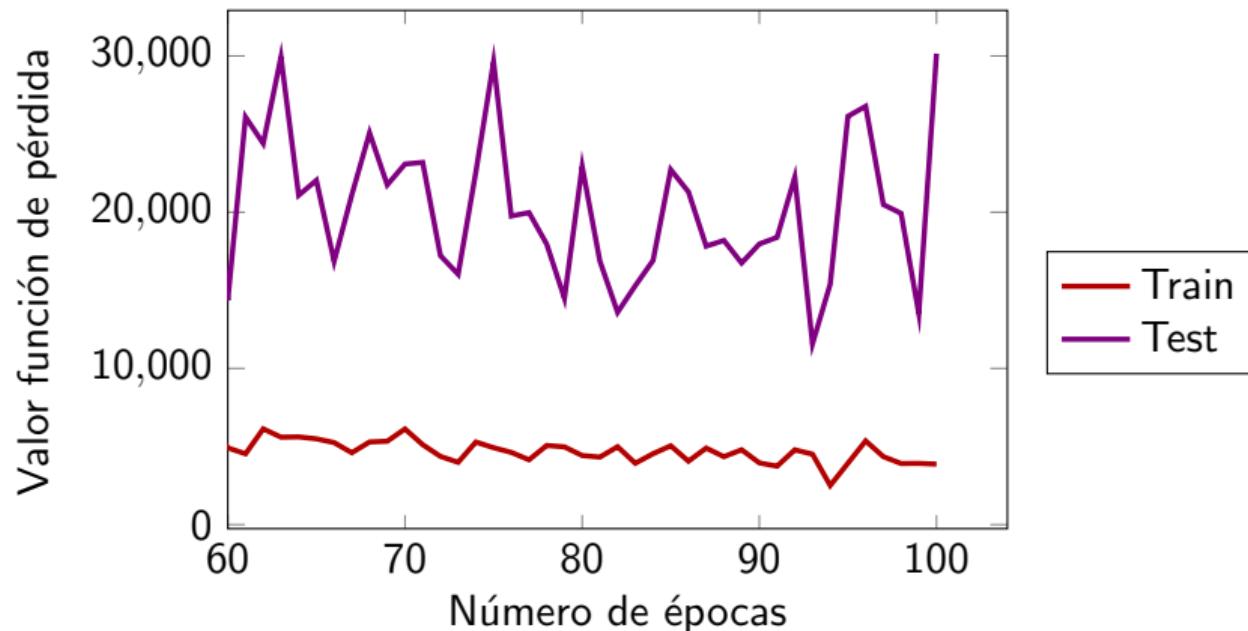
Generalización

En la práctica, el comportamiento es más errático que en el gráfico anterior.



Generalización

En la práctica, el comportamiento es más errático que en el gráfico anterior.



¿Cómo podemos lograr que nuestro modelo generalice mejor?

Generalización

Para estudiar generalización se hace el siguiente supuesto:

- Existe una distribución de probabilidad \mathcal{D} que genera los datos.
- Los datos de entrenamiento son muestreado de esta distribución $(x_i, y_i) \stackrel{\text{iid}}{\sim} \mathcal{D}$.
- La distribución \mathcal{D} no la conocemos (puede venir del mundo).
- Para que sea fáctible aprender algo, \mathcal{D} debe poseer cierta estructura.

Generalización

Para estudiar generalización se hace el siguiente supuesto:

- Existe una distribución de probabilidad \mathcal{D} que genera los datos.
- Los datos de entrenamiento son muestrado de esta distribución $(x_i, y_i) \stackrel{\text{iid}}{\sim} \mathcal{D}$.
- La distribución \mathcal{D} no la conocemos (puede venir del mundo).
- Para que sea fáctible aprender algo, \mathcal{D} debe poseer cierta estructura.

Existen dos formas de generalización:

- *In-distribution generalization:*
- *Out-of-distribution generalization:*

Generalización

Para estudiar generalización se hace el siguiente supuesto:

- Existe una distribución de probabilidad \mathcal{D} que genera los datos.
- Los datos de entrenamiento son muestreado de esta distribución $(x_i, y_i) \stackrel{\text{iid}}{\sim} \mathcal{D}$.
- La distribución \mathcal{D} no la conocemos (puede venir del mundo).
- Para que sea fáctible aprender algo, \mathcal{D} debe poseer cierta estructura.

Existen dos formas de generalización:

- *In-distribution generalization*: Mide que podamos predecir correctamente la salida de nuevas instancias muestreadas de $(x_t, y_t) \sim \mathcal{D}$.
- *Out-of-distribution generalization*:

Generalización

Para estudiar generalización se hace el siguiente supuesto:

- Existe una distribución de probabilidad \mathcal{D} que genera los datos.
- Los datos de entrenamiento son muestreado de esta distribución $(x_i, y_i) \stackrel{\text{iid}}{\sim} \mathcal{D}$.
- La distribución \mathcal{D} no la conocemos (puede venir del mundo).
- Para que sea fáctible aprender algo, \mathcal{D} debe poseer cierta estructura.

Existen dos formas de generalización:

- *In-distribution generalization*: Mide que podamos predecir correctamente la salida de nuevas instancias muestreadas de $(x_t, y_t) \sim \mathcal{D}$.
- *Out-of-distribution generalization*: Mide que podamos predecir correctamente la salida de instancias muestreadas de otra distribución \mathcal{D}' que no es igual a \mathcal{D} .

Generalización

Para estudiar generalización se hace el siguiente supuesto:

- Existe una distribución de probabilidad \mathcal{D} que genera los datos.
- Los datos de entrenamiento son muestreado de esta distribución $(x_i, y_i) \stackrel{\text{iid}}{\sim} \mathcal{D}$.
- La distribución \mathcal{D} no la conocemos (puede venir del mundo).
- Para que sea fáctible aprender algo, \mathcal{D} debe poseer cierta estructura.

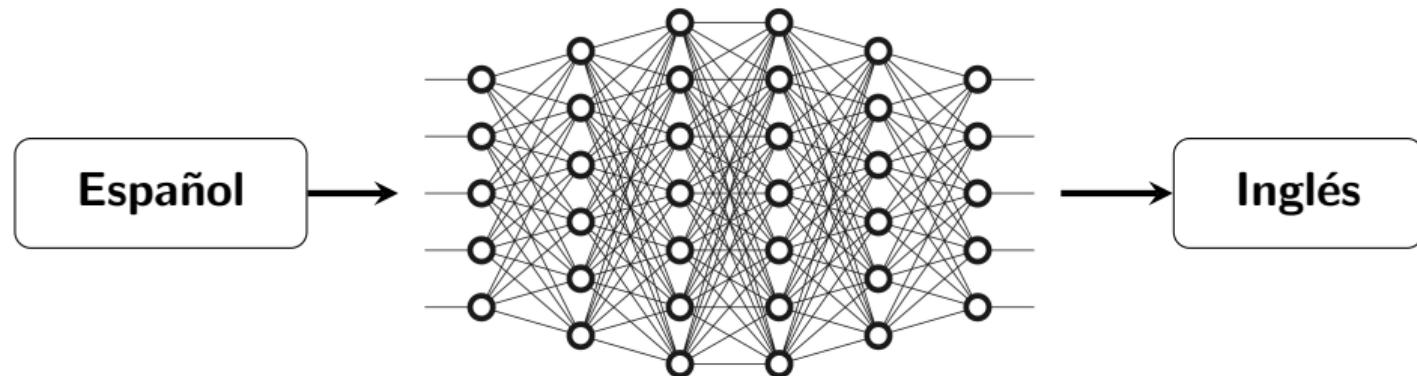
Existen dos formas de generalización:

- *In-distribution generalization*: Mide que podamos predecir correctamente la salida de nuevas instancias muestreadas de $(x_t, y_t) \sim \mathcal{D}$.
- *Out-of-distribution generalization*: Mide que podamos predecir correctamente la salida de instancias muestreadas de otra distribución \mathcal{D}' que no es igual a \mathcal{D} .

En general, deep learning solo logra generalizar *in distribution*.

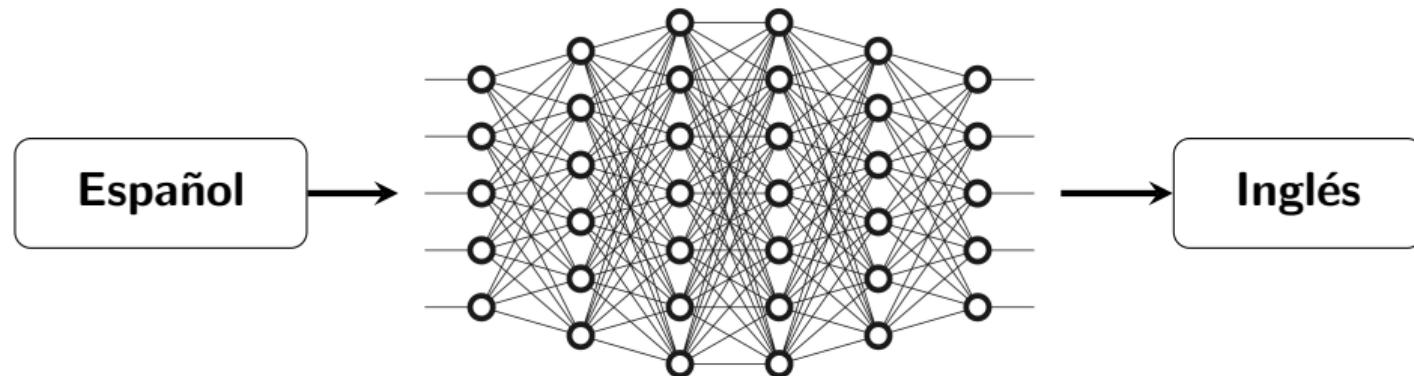
Generalización

Ejemplo: Si entrenaste el modelo usando oraciones en español.



Generalización

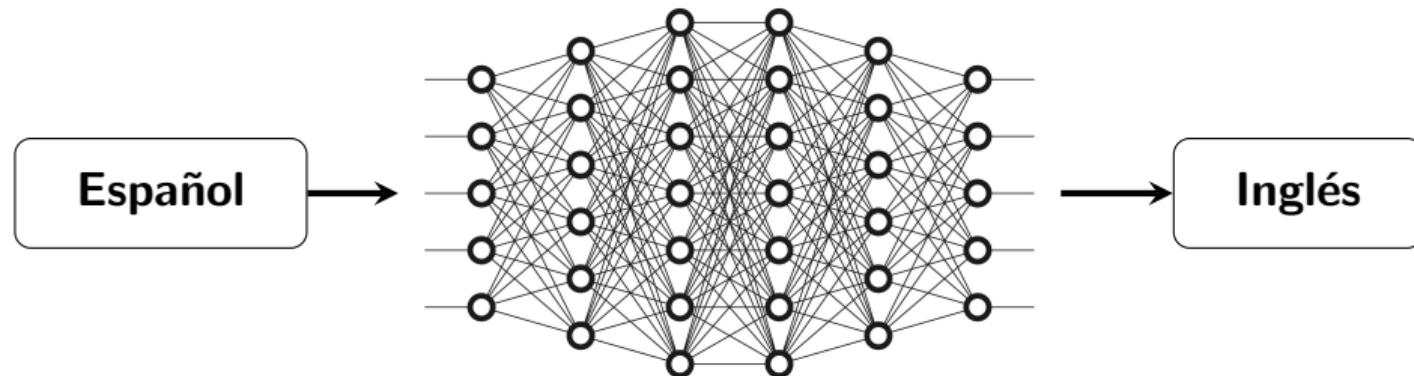
Ejemplo: Si entrenaste el modelo usando oraciones en español.



Testear con una nueva oración en español: *in distribution* ✓

Generalización

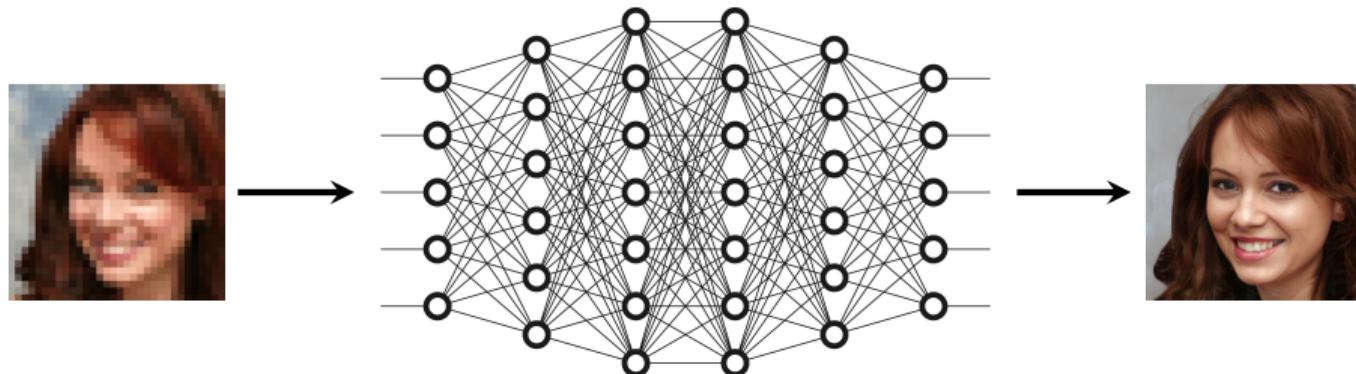
Ejemplo: Si entrenaste el modelo usando oraciones en español.



Testear con una oración en chino: *out of distribution* **X**

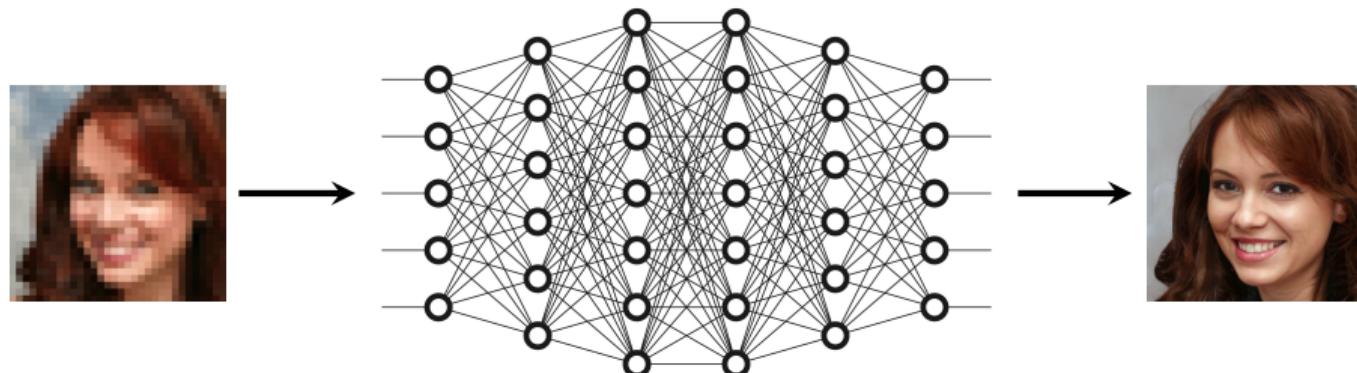
Generalización

Ejemplo: Si entrenaste el modelo usando fotos de personas blancas.



Generalización

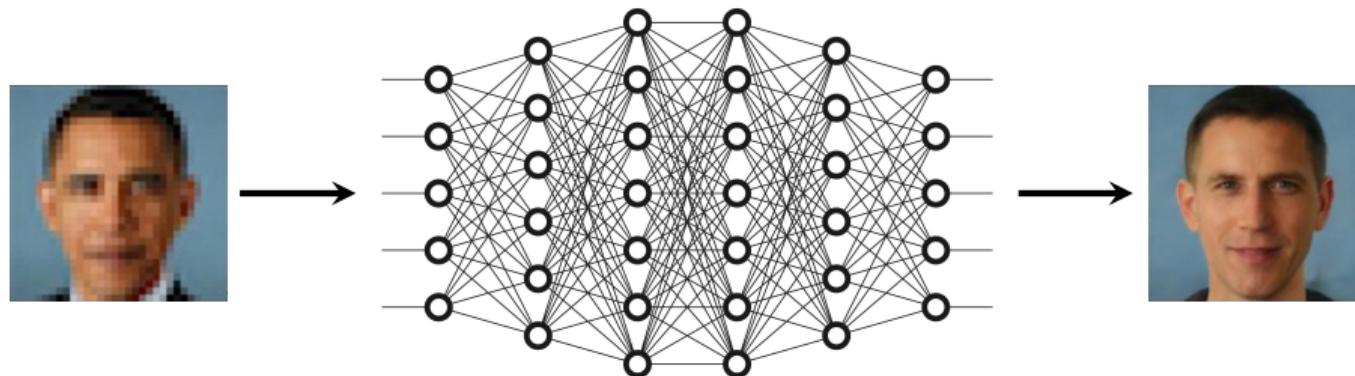
Ejemplo: Si entrenaste el modelo usando fotos de personas blancas.



Testear con una foto de una mujer blanca: *in distribution* ✓

Generalización

Ejemplo: Si entrenaste el modelo usando fotos de personas blancas.



Testear con una foto de un hombre afroamericano: *out of distribution* **X**

Generalización

Ejemplo: Los datasets de reconocimiento facial solían ser personas de frente.



Generalización

Ejemplo: Los datasets de reconocimiento facial solían ser personas de frente.



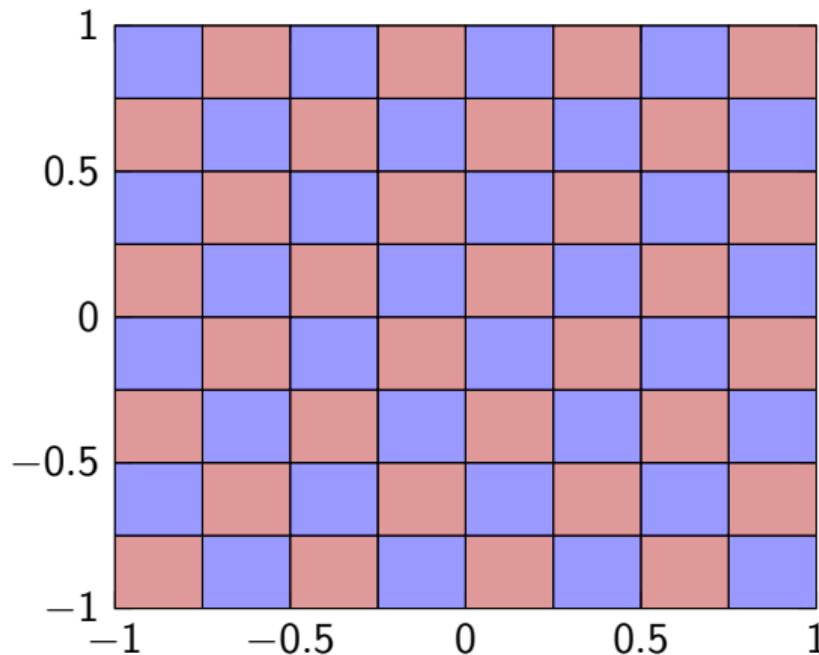
Consecuencia: Los reconocedores **no** podían reconocer personas de perfil.

Generalización

Les propongo estudiar generalización usando un ejemplo super simple.

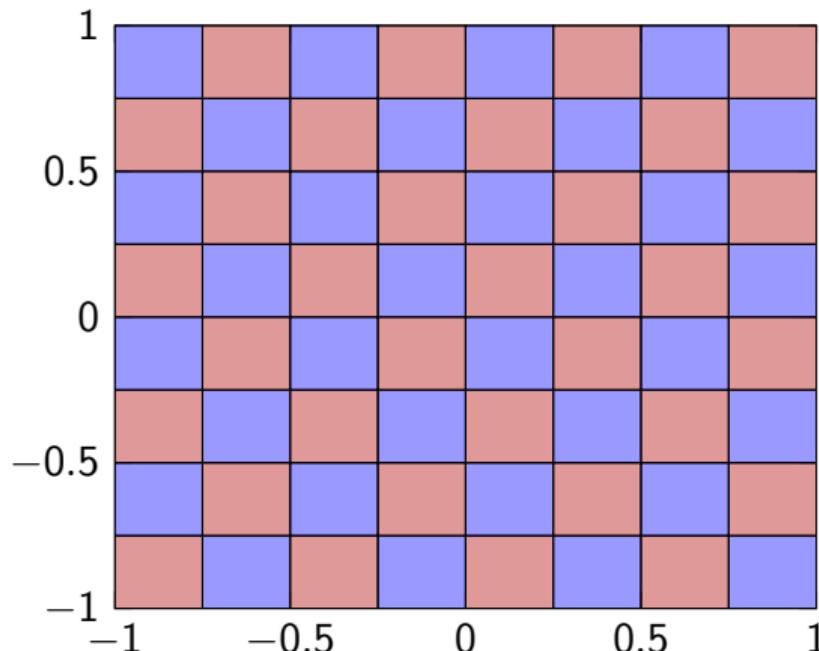
Generalización

Les propongo estudiar generalización usando un ejemplo super simple.



Generalización

Les propongo estudiar generalización usando un ejemplo super simple.

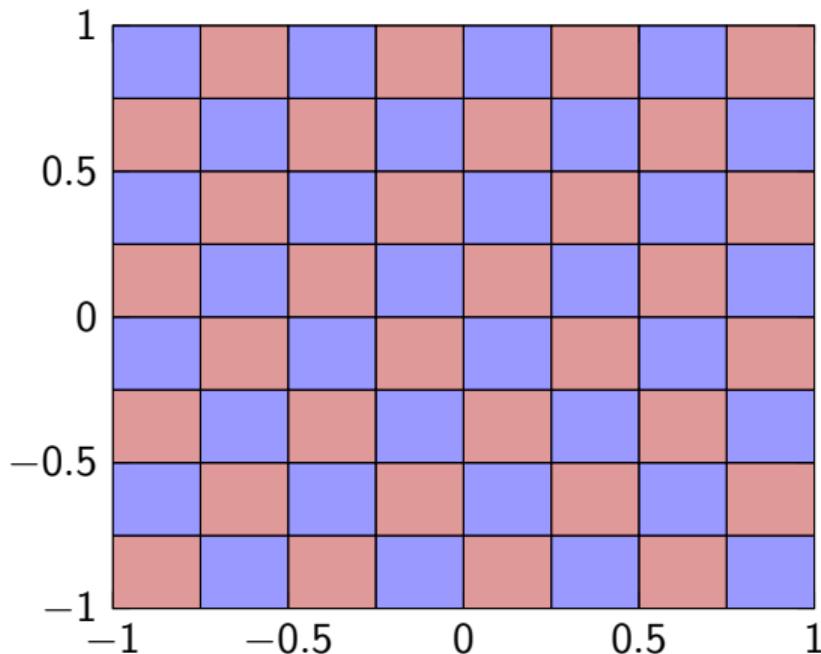


\mathcal{D} muestrea puntos aleatorios entre -1 y 1.

- Si cae en un cuadro rojo, su clase es 0.
- Si cae en un cuadro azul, su clase es 1.

Generalización

Les propongo estudiar generalización usando un ejemplo super simple.



\mathcal{D} muestrea puntos aleatorios entre -1 y 1.

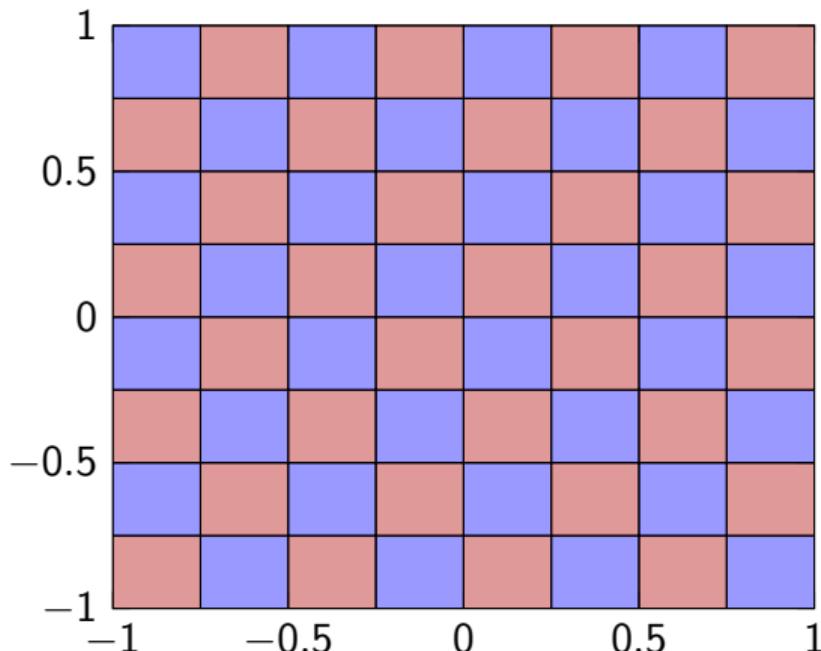
- Si cae en un cuadro rojo, su clase es 0.
- Si cae en un cuadro azul, su clase es 1.

Muestramos:

- 500 ejemplos de entrenamiento.

Generalización

Les propongo estudiar generalización usando un ejemplo super simple.



\mathcal{D} muestrea puntos aleatorios entre -1 y 1.

- Si cae en un cuadro rojo, su clase es 0.
- Si cae en un cuadro azul, su clase es 1.

Muestramos:

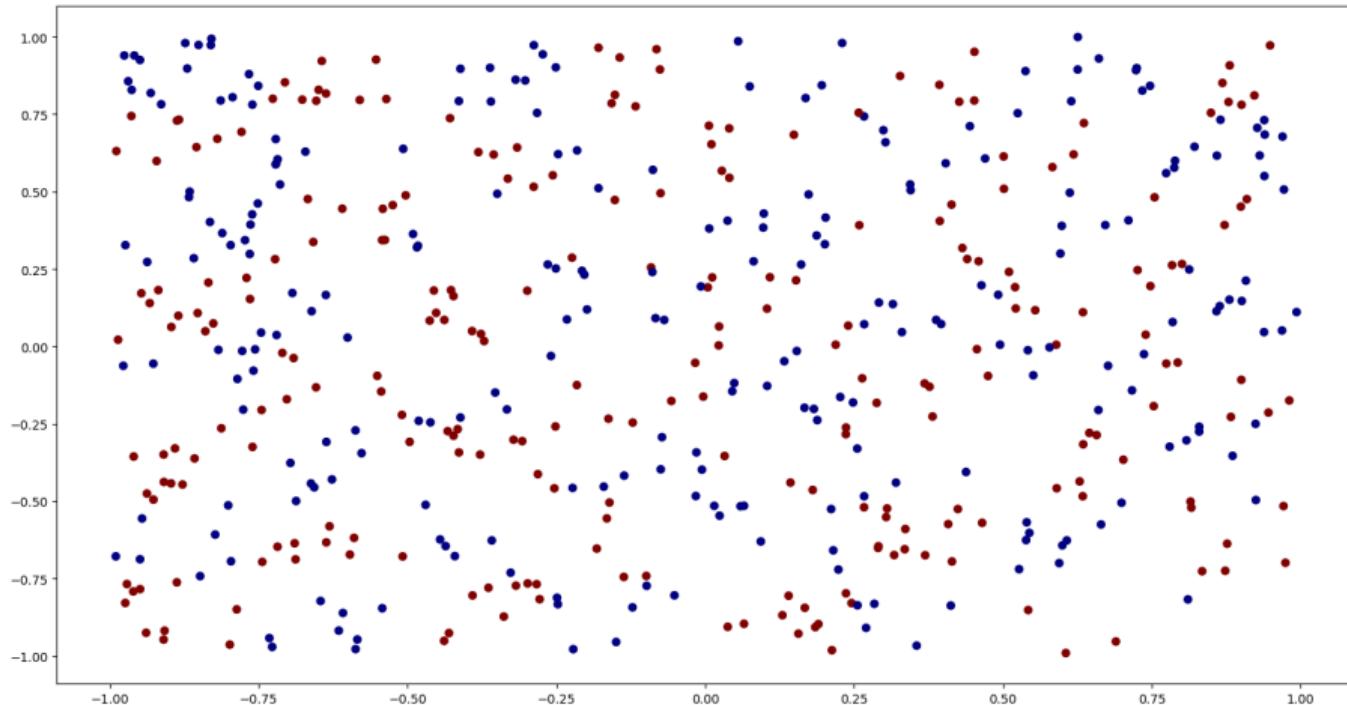
- 500 ejemplos de entrenamiento.

Testeamos en:

- 1000 ejemplos *in distribution*.
- 1000 ejemplos *out of distribution*.

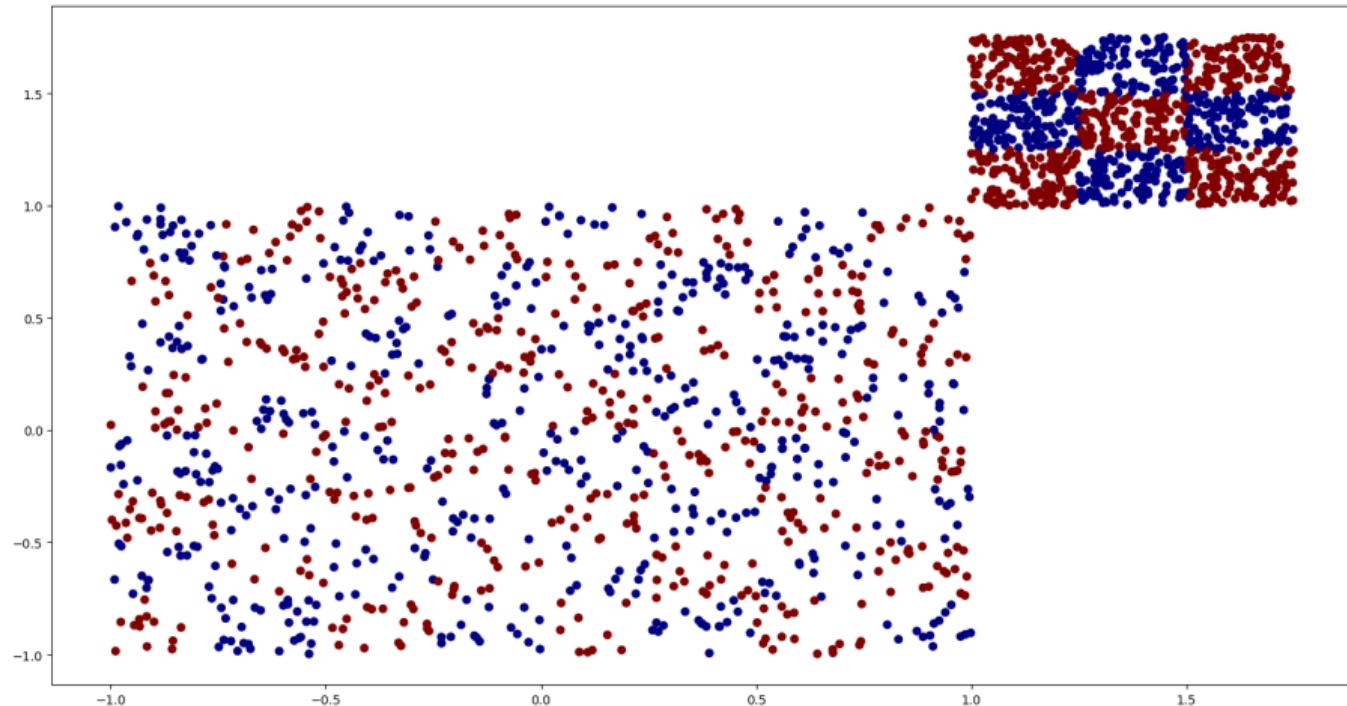
Generalización

Ejemplos de entrenamiento:



Generalización

Ejemplos de testeo:



Generalización

Este es un problema de clasificación binaria con dos inputs y un output.

- Capa de input de tamaño 2.
- Luego 4 capas ocultas de 256 neuronas Relu.
- Capa de output 1 neurona Sigmoid.

Generalización

Este es un problema de clasificación binaria con dos inputs y un output.

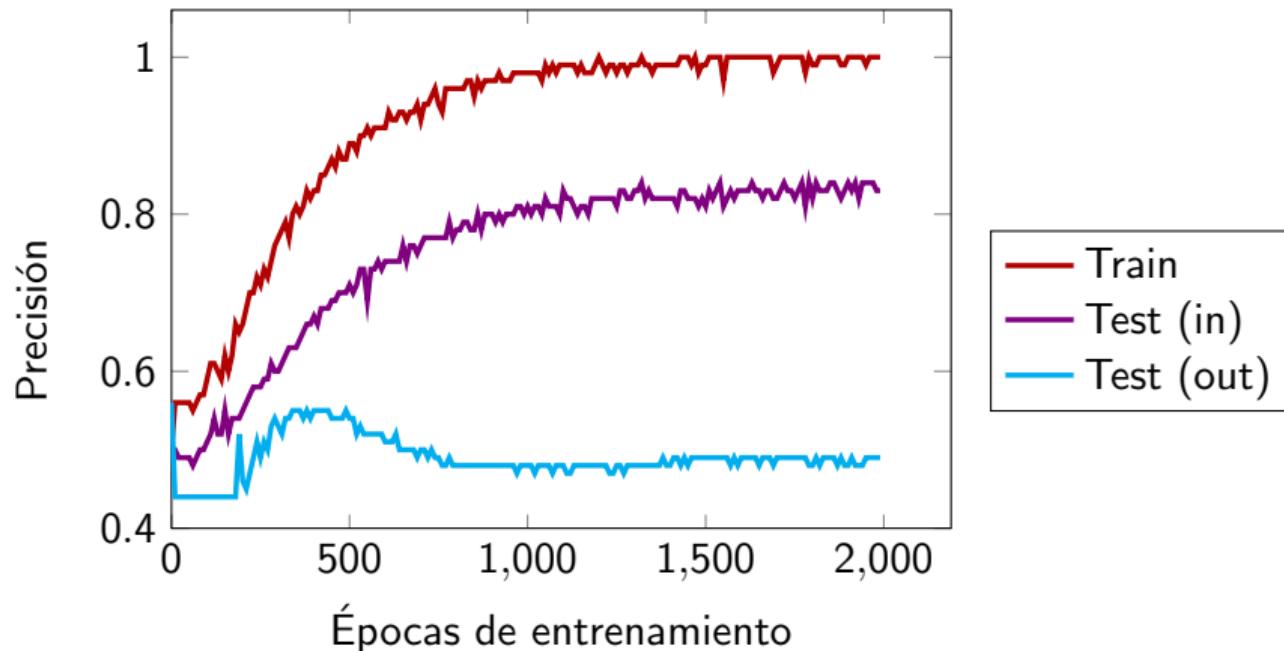
- Capa de input de tamaño 2.
- Luego 4 capas ocultas de 256 neuronas Relu.
- Capa de output 1 neurona Sigmoid.

Otros parámetros de configuración:

- Usamos negative log-likelihood como función de pérdida.
- Los datos ya están normalizados.
- Batches de tamaño 128.
- Learning rate de 1e-4.
- Optimizamos usando ADAM.

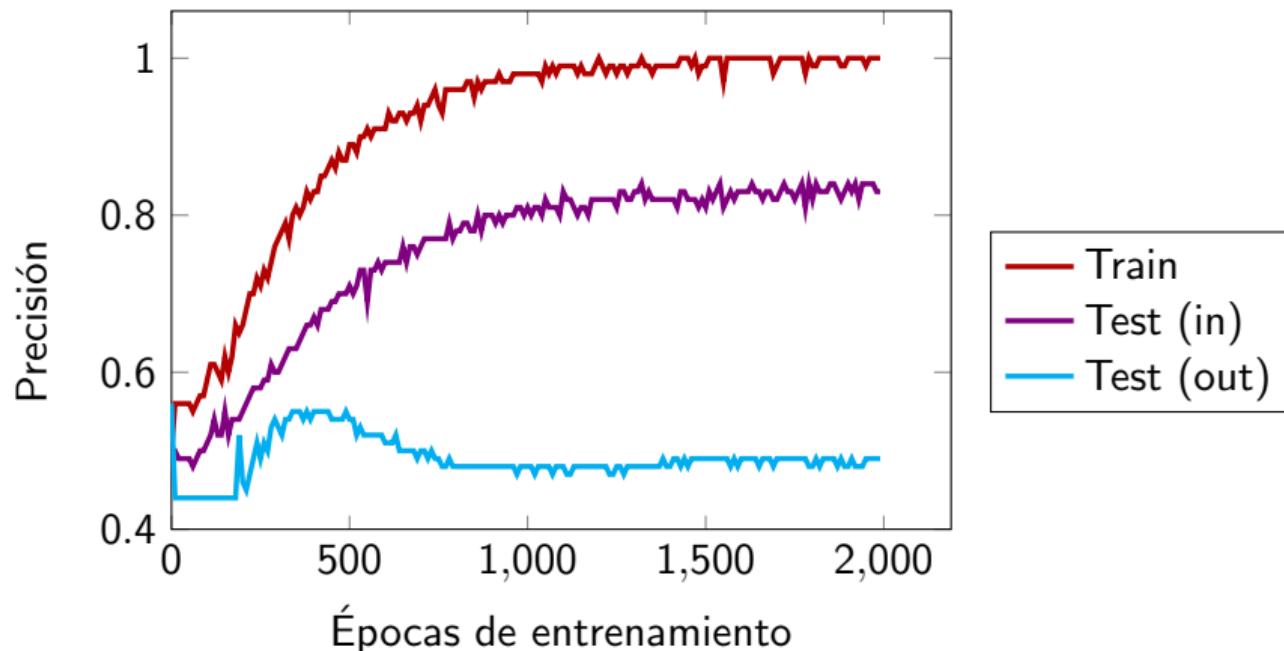
Generalización

Resultados usando 500 ejemplos de entrenamiento:



Generalización

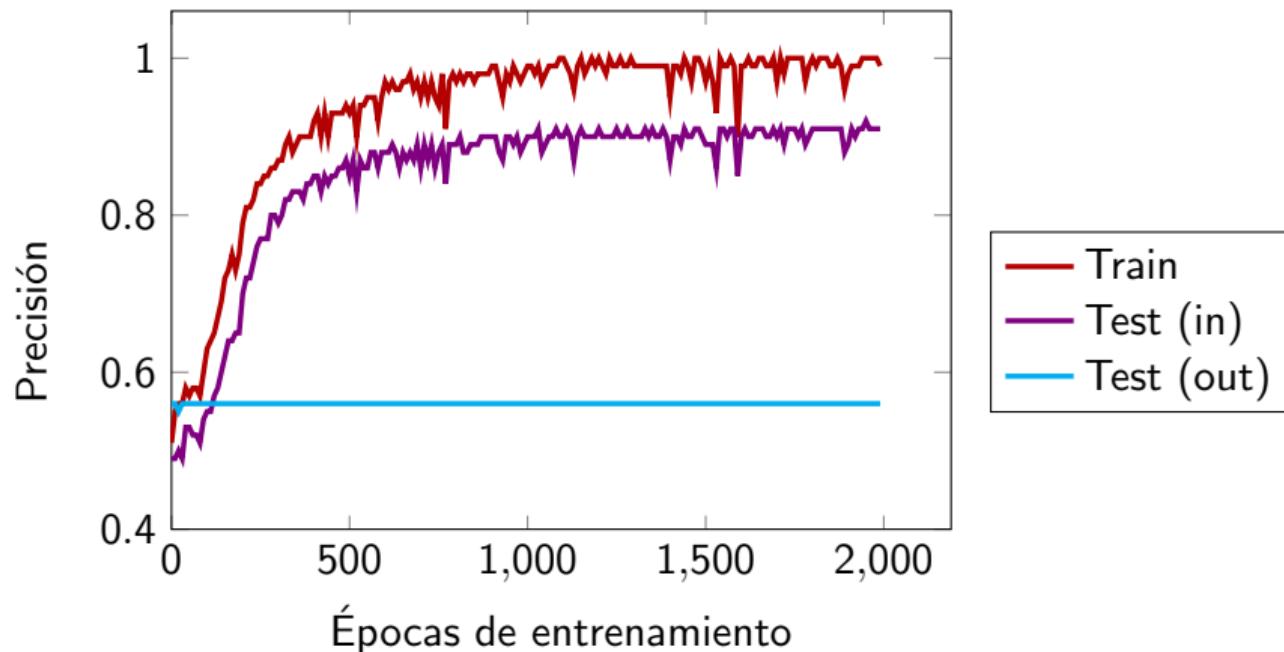
Resultados usando 500 ejemplos de entrenamiento:



¿Cuál es la forma más fácil de mejorar la generalización del modelo?

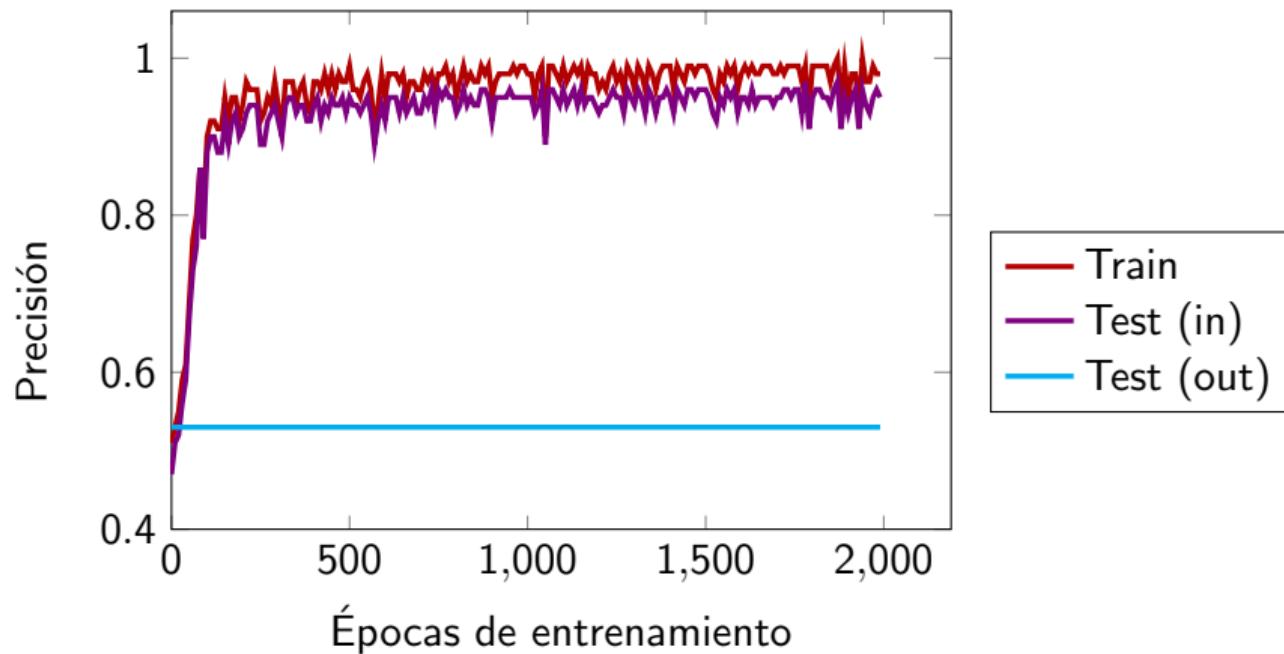
Generalización

Resultados usando 1000 ejemplos de entrenamiento:



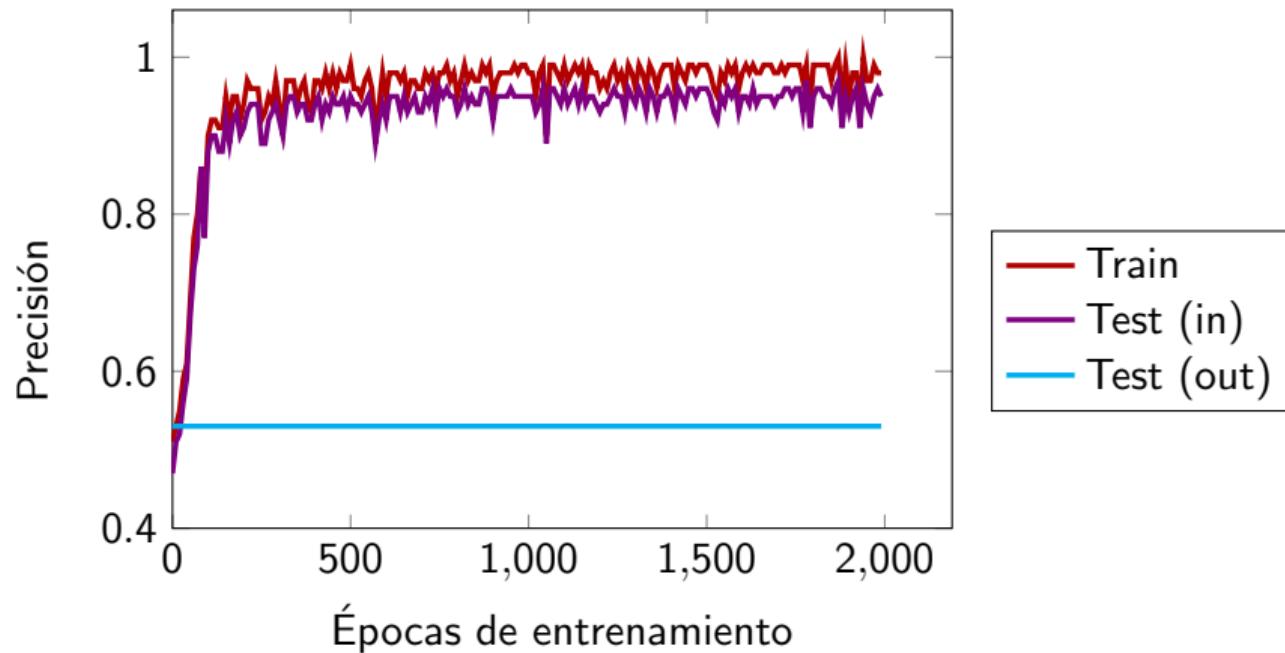
Generalización

Resultados usando 5000 ejemplos de entrenamiento:



Generalización

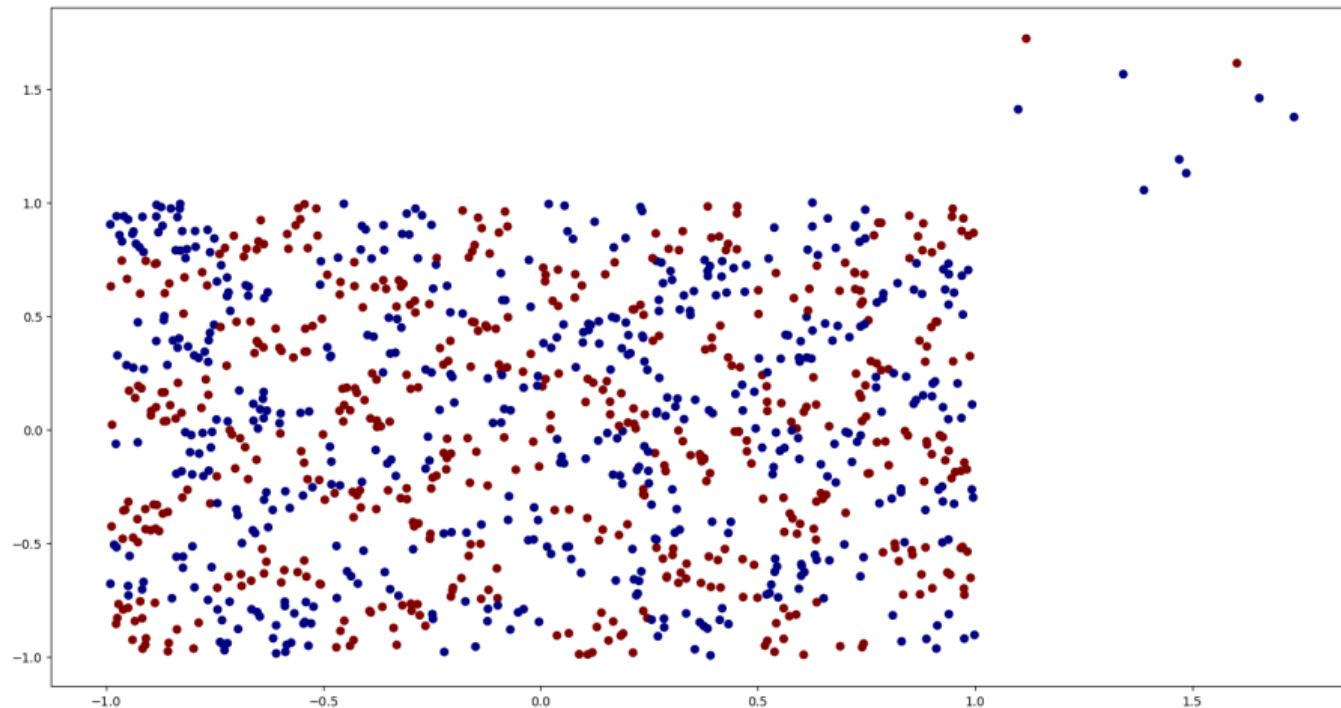
Resultados usando 5000 ejemplos de entrenamiento:



¿Existe algo que podamos hacer para mejorar el rendimiento *out of distribution*?

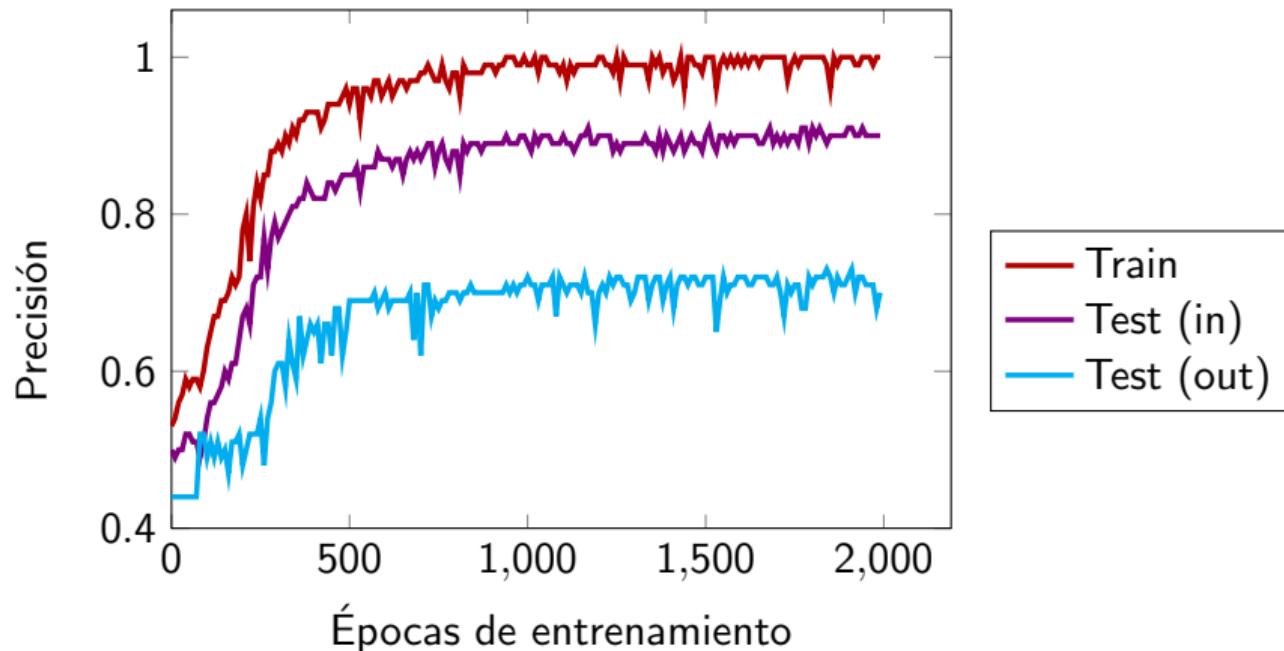
Generalización

Conseguir algunos ejemplos *out of distribution* y agregarlos a *train*.



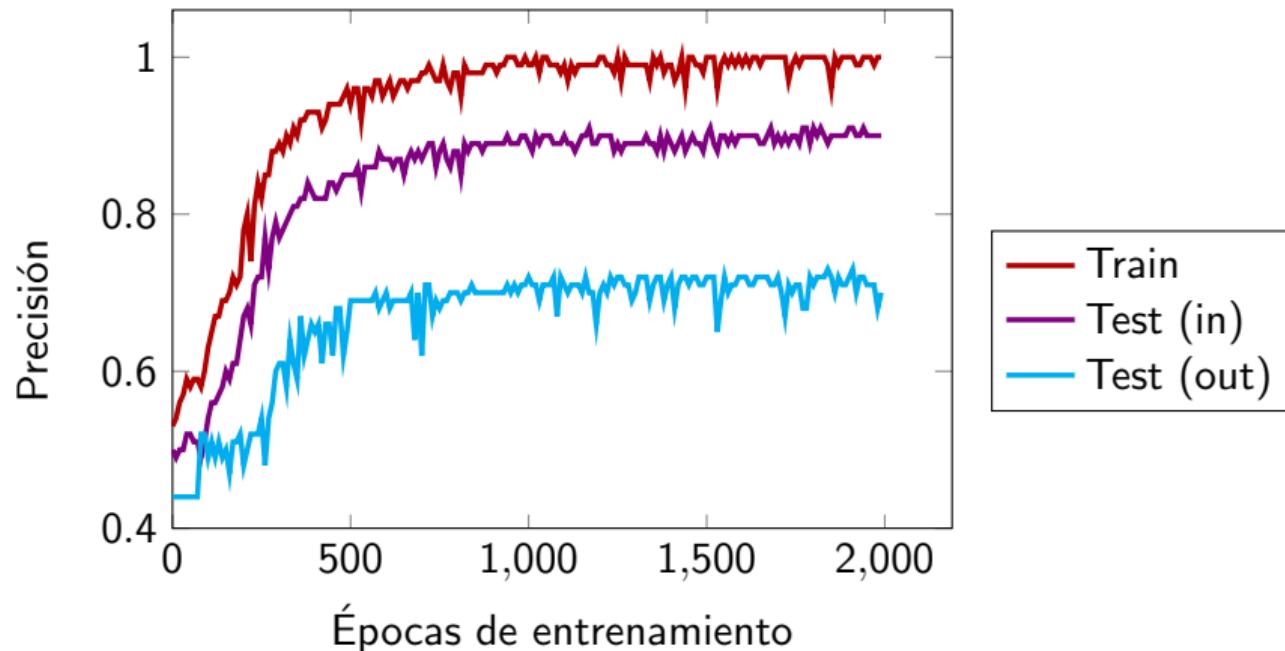
Generalización

Resultados usando 1000 ejemplos de entrenamiento + 9 ejemplos OOD.



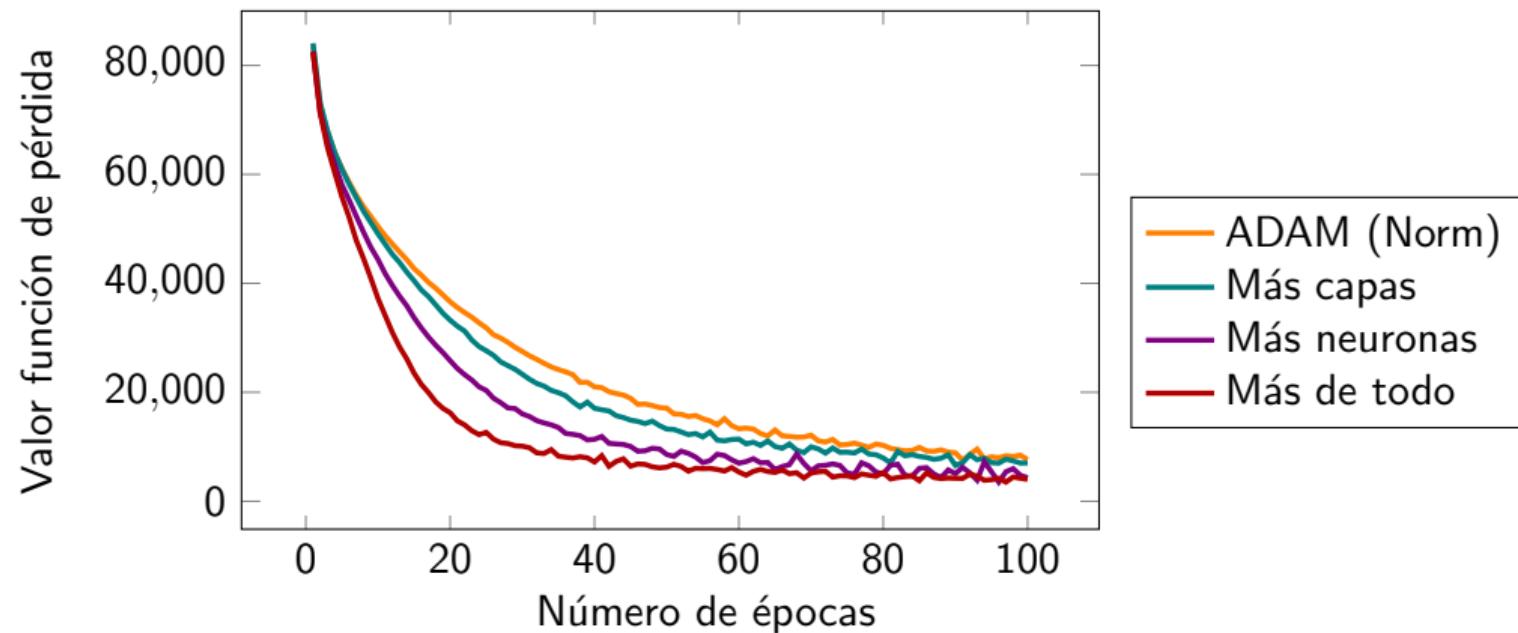
Generalización

Resultados usando 1000 ejemplos de entrenamiento + 9 ejemplos OOD.



¿... y si no tenemos más datos? ¿hay algo que podamos hacer?

Generalización



Redes actuales obtienen sobre 90.65% de test en CIFAR-10 (sin usar más datos⁴).

⁴ “Maxout Networks” by Goodfellow et al. (ICML13).

Ideas Finales

Para entrenar redes neuronales sobre grandes volúmenes de datos:

- Usamos alguna variante de descenso de gradiente estocástico.
- Hay que normalizar los datos.
- Hay que asegurarse de usar una red suficientemente grande.
- Hay que ajustar el learning rate.

Para entrenar redes neuronales sobre grandes volúmenes de datos:

- Usamos alguna variante de descenso de gradiente estocástico.
- Hay que normalizar los datos.
- Hay que asegurarse de usar una red suficientemente grande.
- Hay que ajustar el learning rate.

Generalización:

- Para evaluar generalización usamos un set de testeo.
- **Overfitting:** Buen rendimiento en entrenamiento, mal rendimiento en test.
- La clave para generalizar bien es tener más datos.
- Los datos de entrenamiento deben parecerse a los datos de test.

Preguntas