



Semillero de Investigación “Hands - on” Computer Vision



SESIÓN 3: DEEP LEARNING

Contenidos

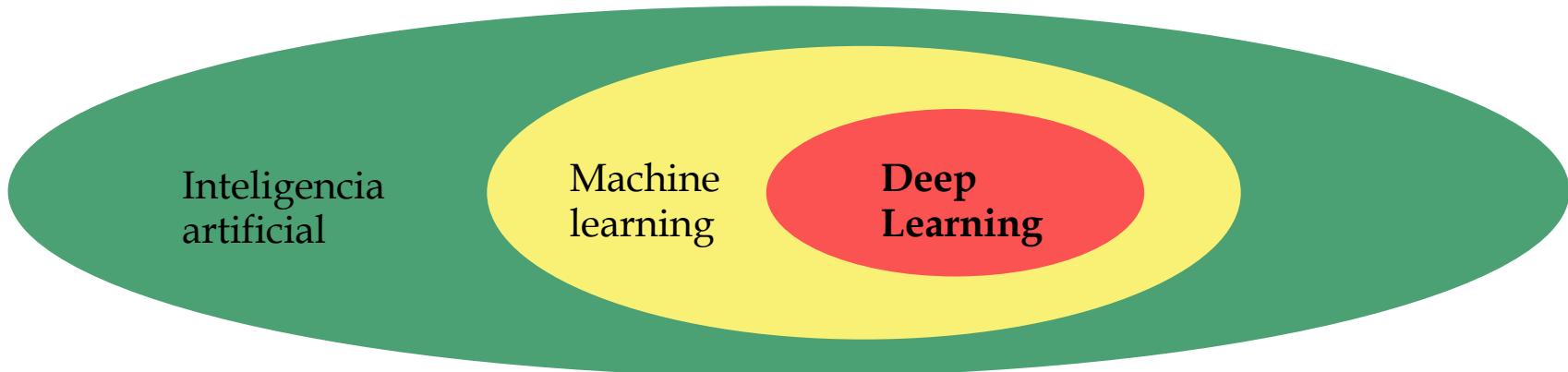
1. ¿Que es el Deep Learning?
2. Aplicaciones
3. Fundamentos
4. No-linealidad
5. Losses
6. Backprograpation, SGD
7. Interpretación de W
8. Redes neuronales convolucionales
9. Deep Learning: HOY

1. ¿Que es el Deep Learning?

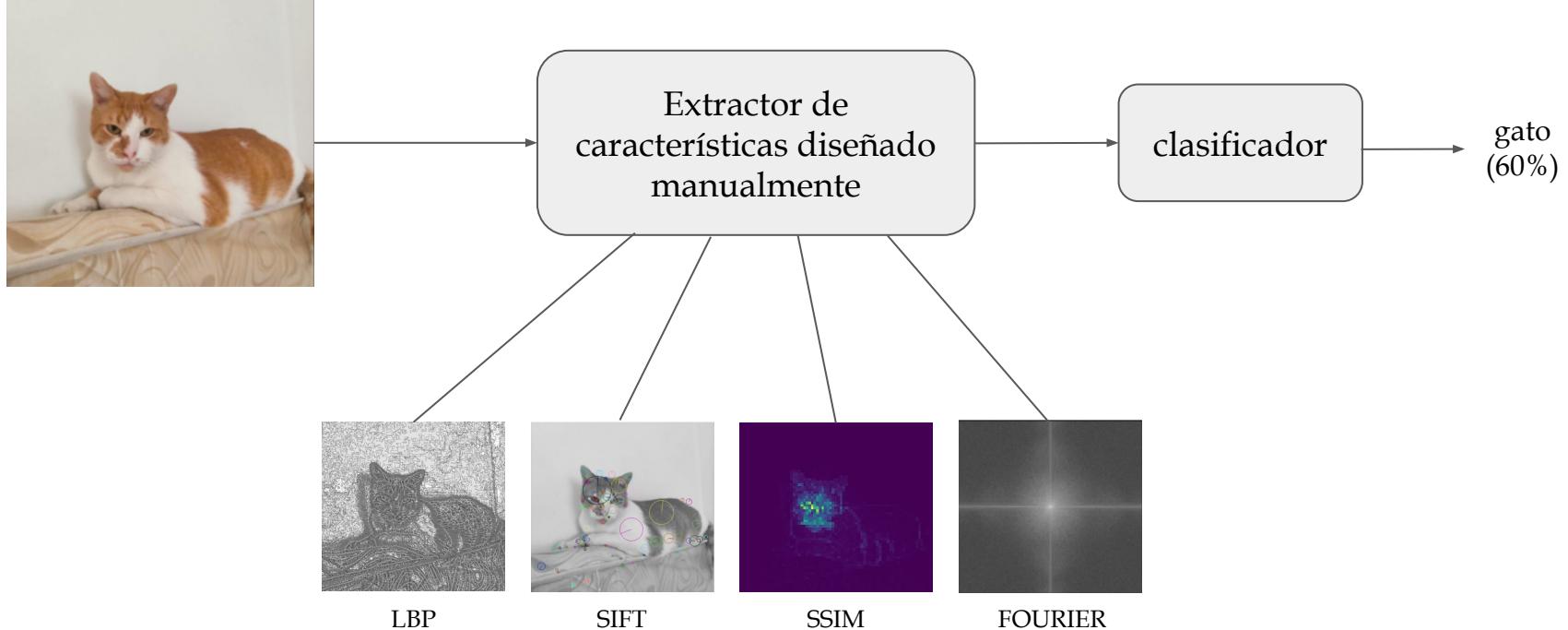
Deep Learning

“El aprendizaje profundo permite a los **modelos computacionales**, que están compuestos por **múltiples capas de procesamiento**, aprender **representaciones de datos** con múltiples **niveles de abstracción**.”

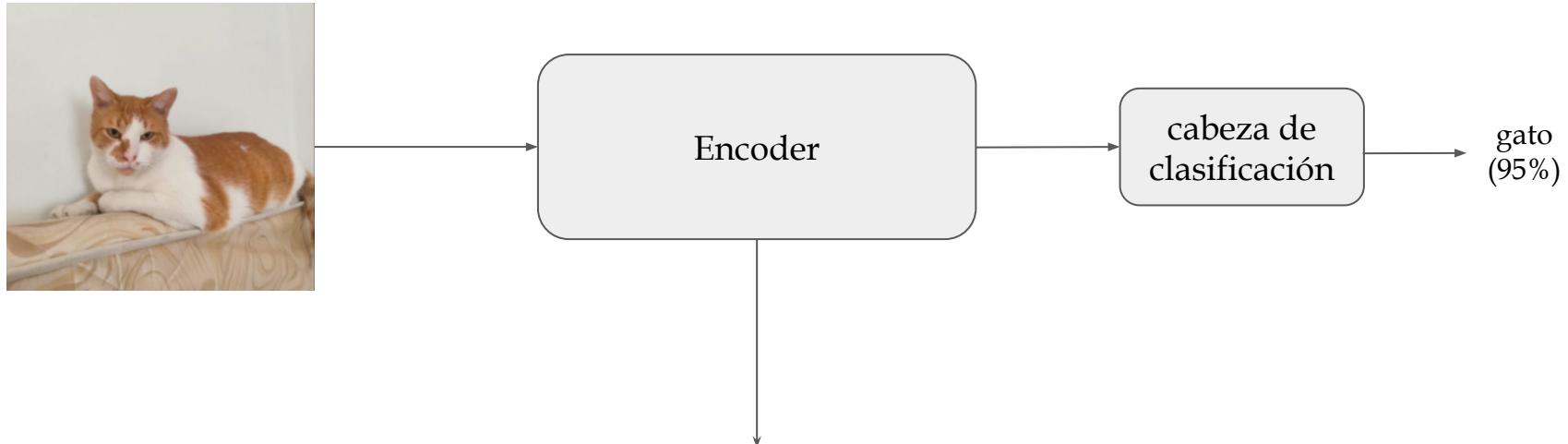
Deep Learning by Y. LeCun et al. Nature 2015



Antes del Deep Learning



Con Deep Learning



extrae representaciones subyacentes a partir de los datos

Deep Learning go brrr



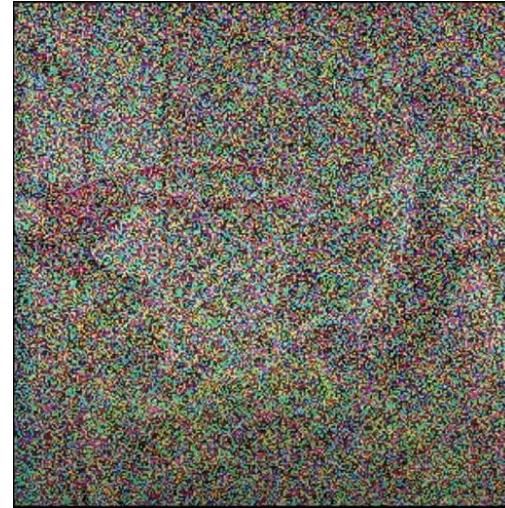
"a modern house with windows"



donde queda esto?

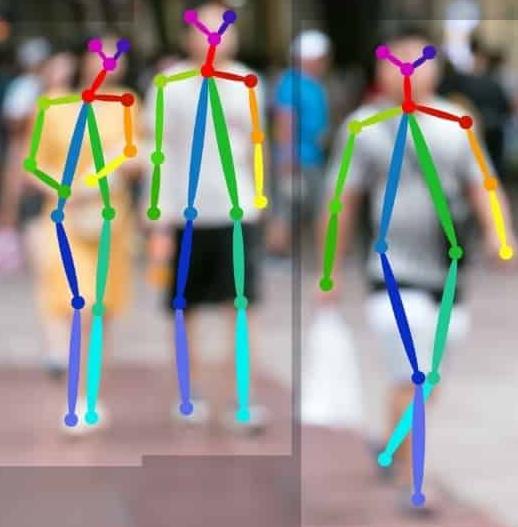
ChatGPT

La imagen muestra un edificio grande con muros de ladrillo y un diseño arquitectónico moderno. El edificio tiene un letrero que dice "Luis A. Calvo", pero no puedo proporcionar una ubicación exacta basada solo en la imagen. Sin embargo, este nombre podría referirse a una sala de conciertos o un espacio cultural, que podría ser parte de una institución educativa o un centro cultural en un país de habla hispana. Si necesitas información más específica, podrías proporcionar un contexto adicional o detalles sobre el edificio.



Deep Learning go brrr

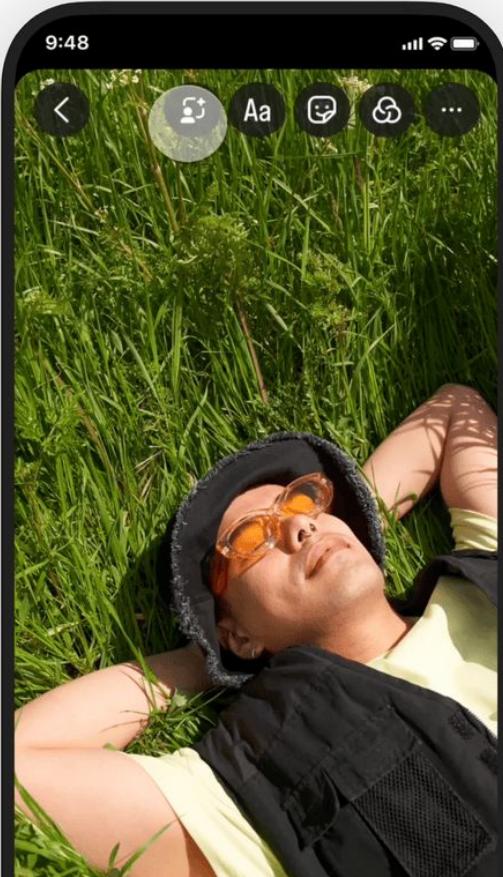




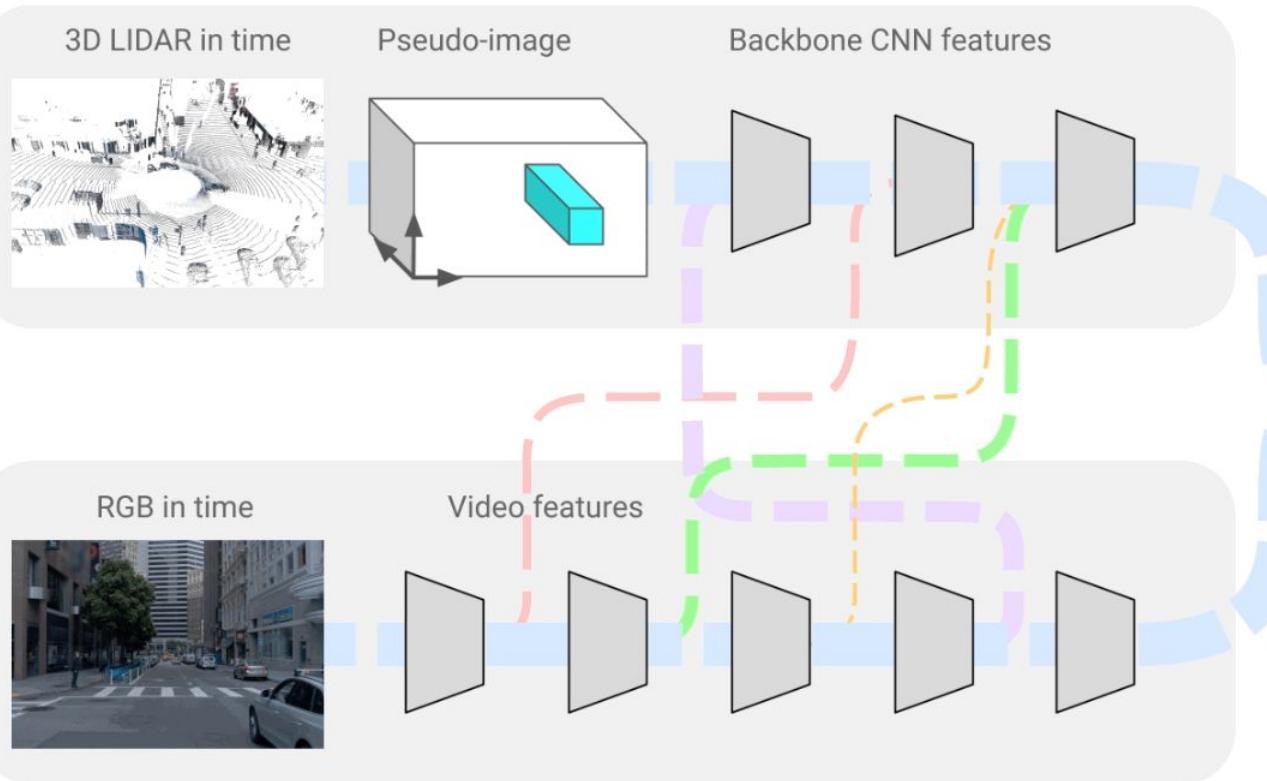
Deep Learning go brrr



Deep Learning go brrr



Deep Learning go brrr



3D boxes



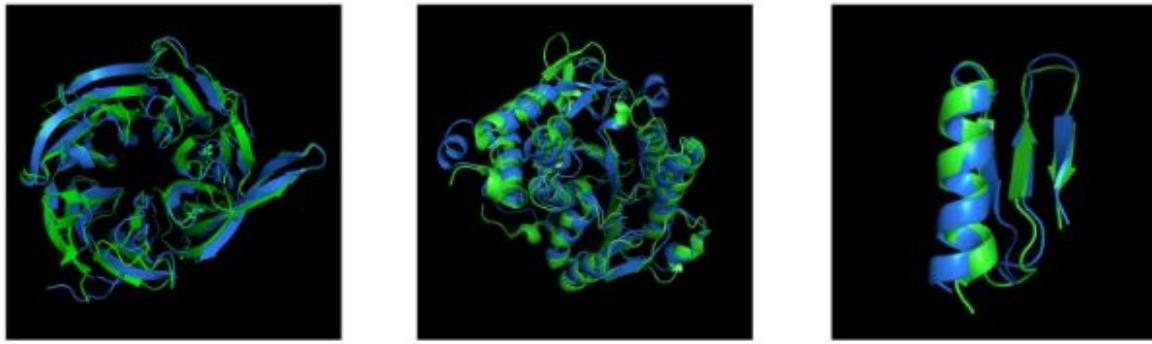


Frame: 1

ATP TENNIS TV

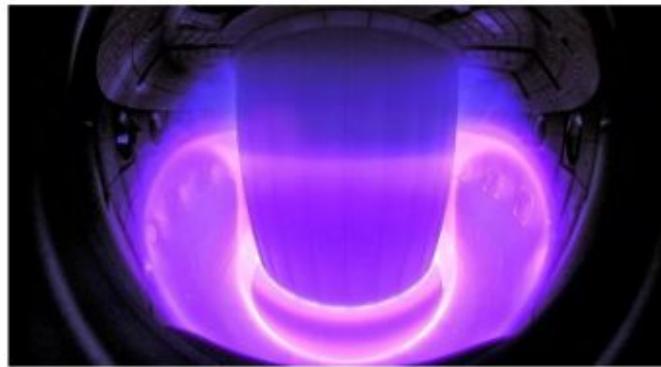


Protein folding prediction



(Jumper et al., 2021)

Plasma confinement



(Degrave et al., 2022)

segmentación en la web!

<https://semillerocv.github.io/models/segment.html>



Embedding extraido!

Reiniciar imagen

Limpiar puntos

Recortar mascara

Clic izquierdo = puntos positivos, clic derecho = puntos negativos.

2. ¿Cómo funciona?

El perceptrón

Imagen



tensor de $32 \times 32 \times 3$
(3072 números)

$$\rightarrow f(\mathbf{x}, \mathbf{W})$$



W

parámetros de
pesos

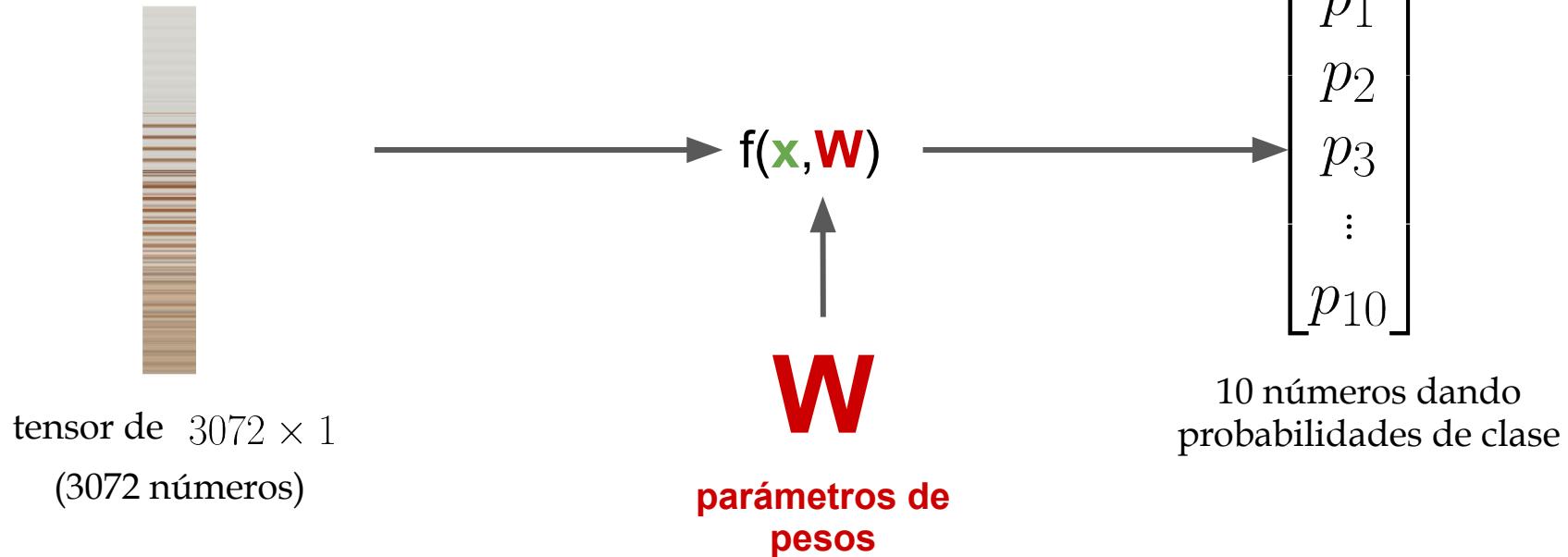
$$\begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ \vdots \\ p_{10} \end{bmatrix}$$

10 números dando
probabilidades de clase

El perceptrón

$$f(\mathbf{x}, \mathbf{W}) = \mathbf{W}^T \mathbf{x}$$

Imagen Vectorizada



El perceptrón

Imagen Vectorizada

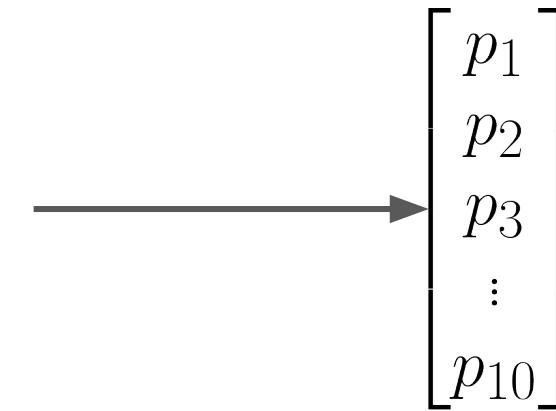


tensor de 3072×1
(3072 números)

$$f(\mathbf{x}, \mathbf{W}) = \mathbf{W}^T \mathbf{x}$$

1×10 3072×10

W
parámetros de
pesos



10 números dando
probabilidades de clase

El perceptrón

Imagen Vectorizada

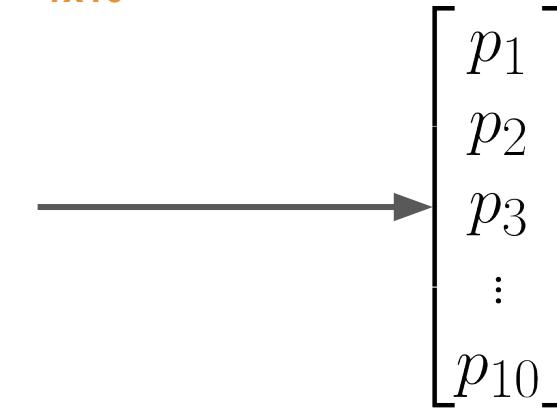


tensor de 3072×1
(3072 números)

$$\hat{y} = f(\mathbf{x}, \mathbf{W}) = \mathbf{W}^T \mathbf{x} + b$$

1×10 3072×10 1×10

W
parámetros de pesos



10 números dando
probabilidades de clase

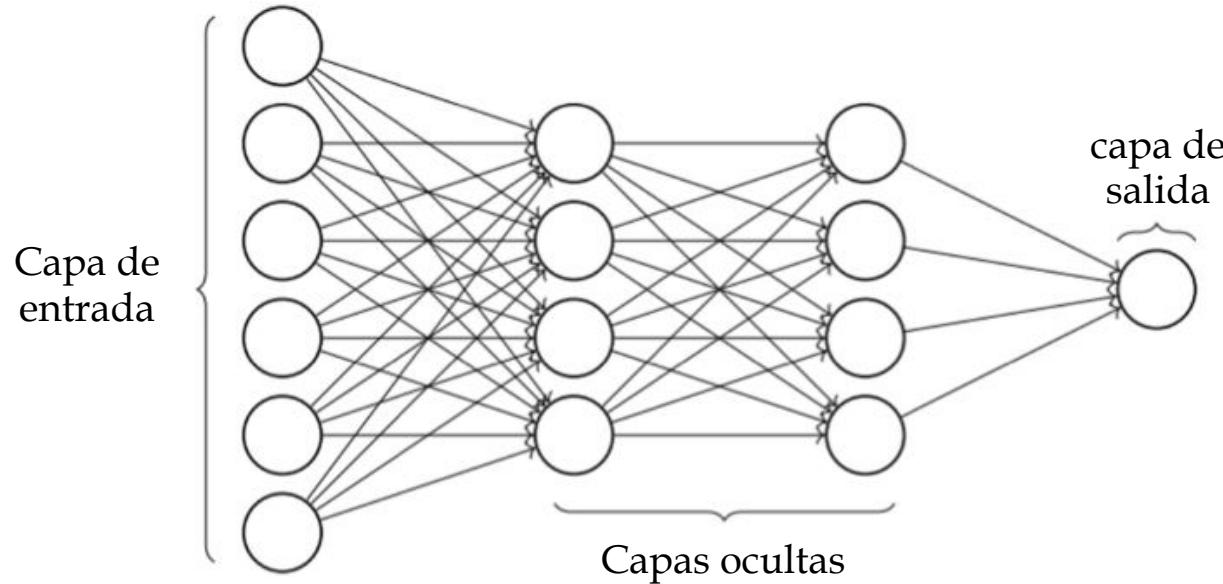
Funciones de activación

se pueden expresar como aplicar una función a la salida:

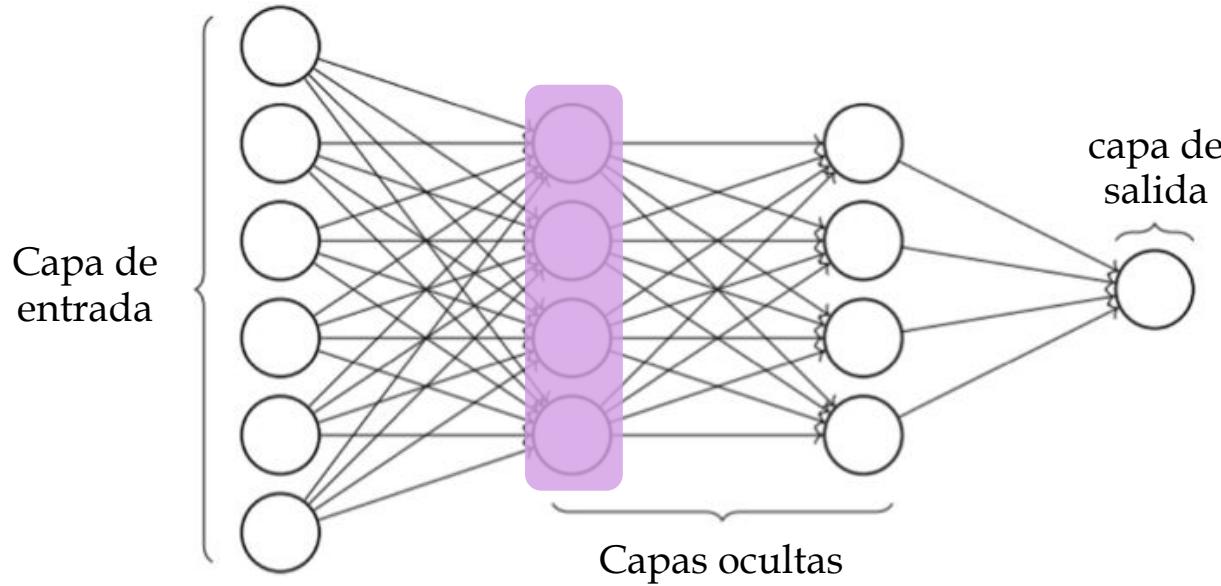
$$\hat{y} = f(\mathbf{x}, \mathbf{W}) = g(\mathbf{W}^T \mathbf{x} + b)$$

cualquier función puede ser aplicada, pero se espera que se aplique una **función no-lineal**

El perceptrón multicapa

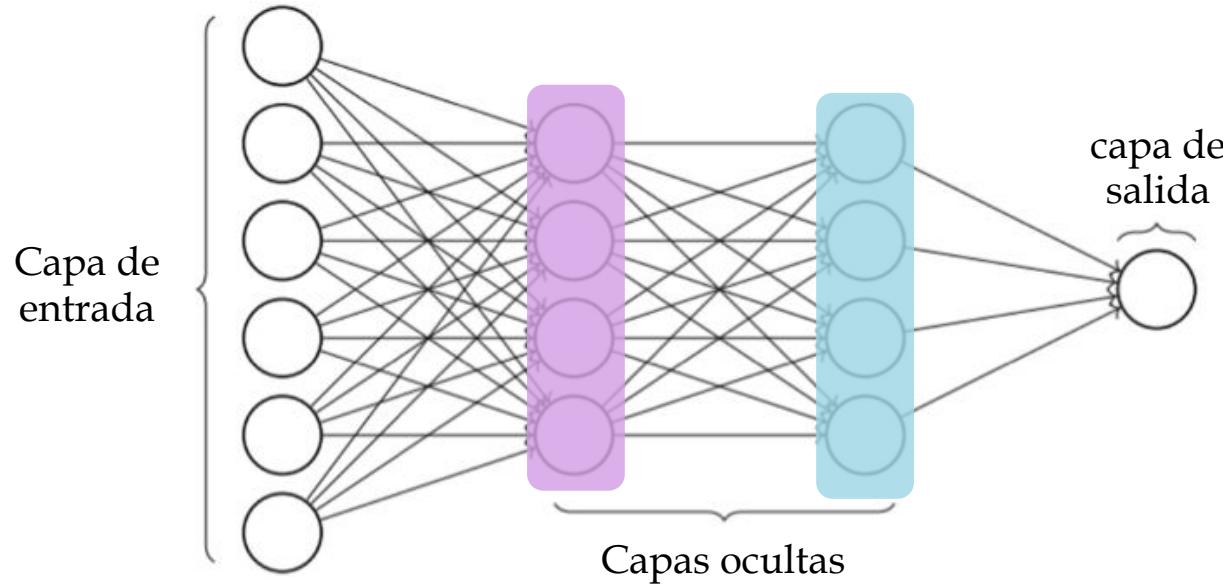


El perceptrón multicapa



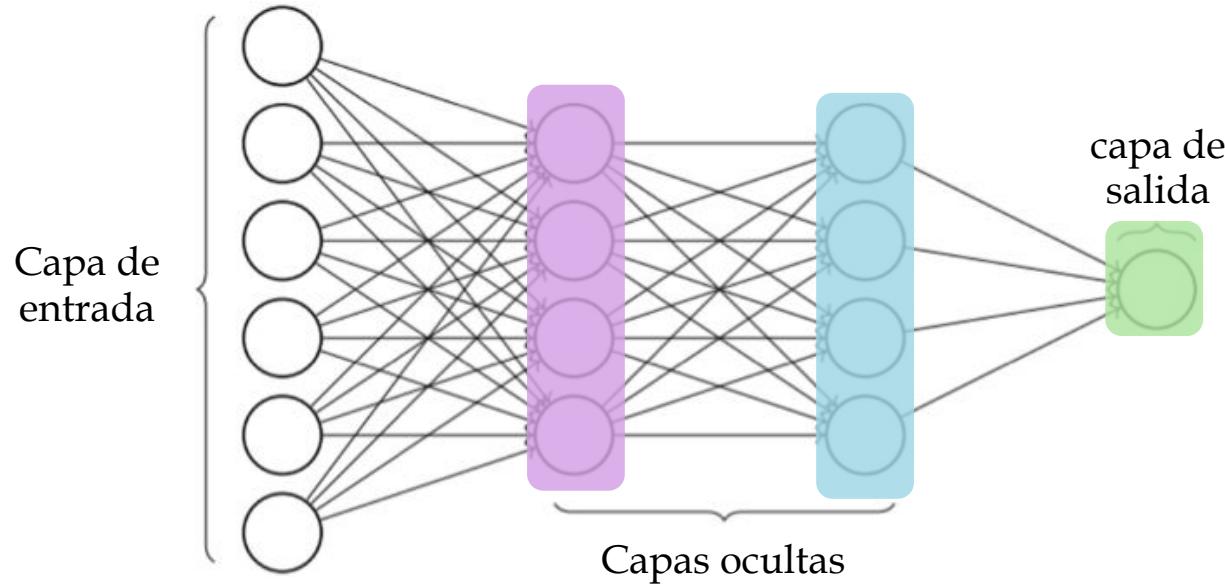
$$\hat{y} = g(W_3^\top \cdot g(W_2^\top \cdot g(W_1^\top \cdot X + b_1) + b_2) + b_3)$$

El perceptrón multicapa



$$\hat{y} = g(W_3^\top \cdot g(W_2^\top \cdot g(W_1^\top \cdot X + b_1) + b_2) + b_3)$$

El perceptrón multicapa



$$\hat{y} = g(W_3^\top \cdot g(W_2^\top \cdot g(W_1^\top \cdot X + b_1) + b_2) + b_3)$$

La necesidad de la no-linealidad

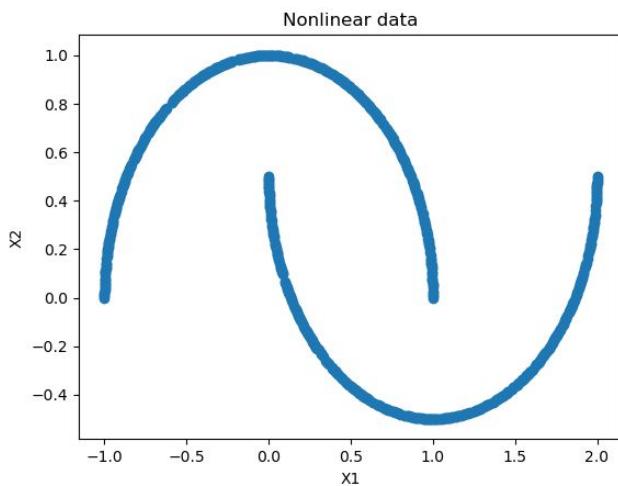
La no linealidad permite modelar la compleja estructura de los datos, capturando patrones que van más allá de las simples relaciones lineales.

$$\hat{y} = g(W_3^T \cdot g(W_2^T \cdot g(W_1^T \cdot X + b_1) + b_2) + b_3$$

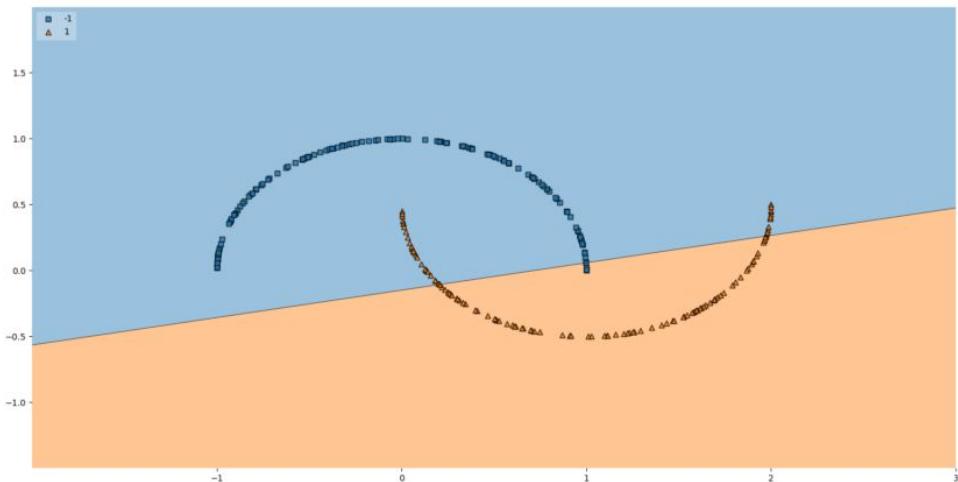
si $\mathbf{g}()$ es una función lineal es decir $\mathbf{g(x)=x}$

$$\begin{aligned}\hat{y} &= W_3^T \cdot (W_2^T \cdot (W_1^T \cdot X + b_1) + b_2) + b_3 \\&= W_3^T \cdot (W_2^T \cdot W_1^T \cdot X + W_2^T \cdot b_1 + b_2) + b_3 \\&= (W_3^T \cdot W_2^T \cdot W_1^T) \cdot X + (W_3^T \cdot W_2^T \cdot b_1 + W_3^T \cdot b_2 + b_3) \\&= W' \cdot X + b'\end{aligned}$$

La necesidad de la no-linealidad

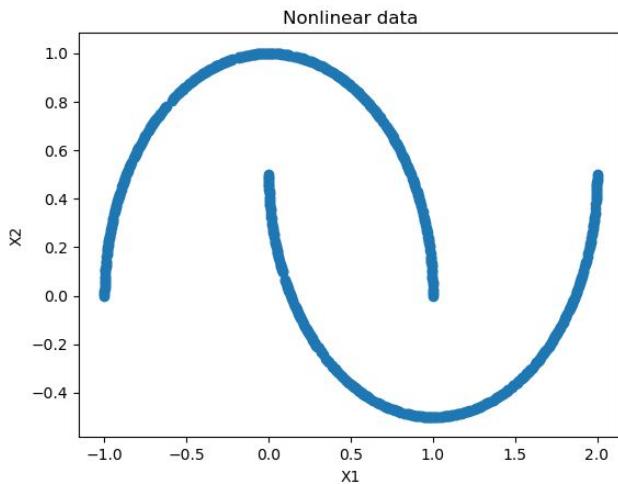


casi todos los problemas del mundo real son no-lineales

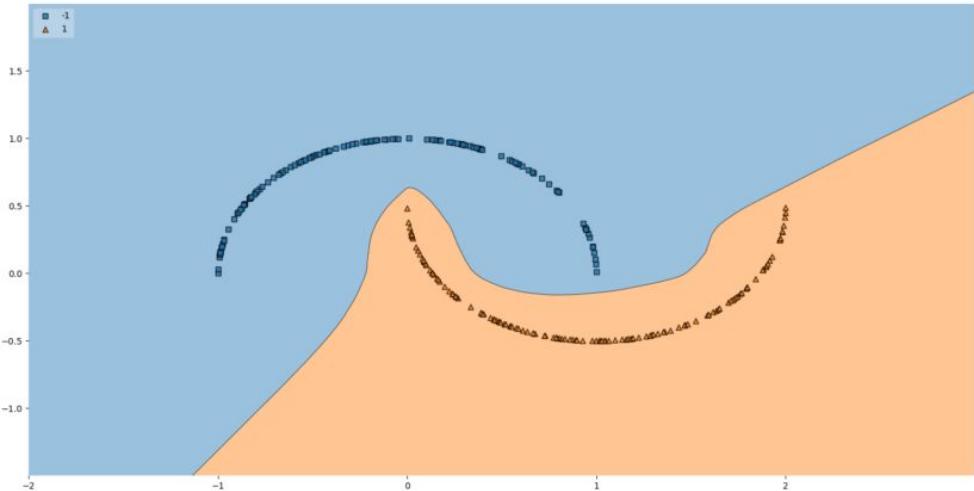


si usamos una función de activación lineal no podemos resolver el problema

La necesidad de la no-linealidad

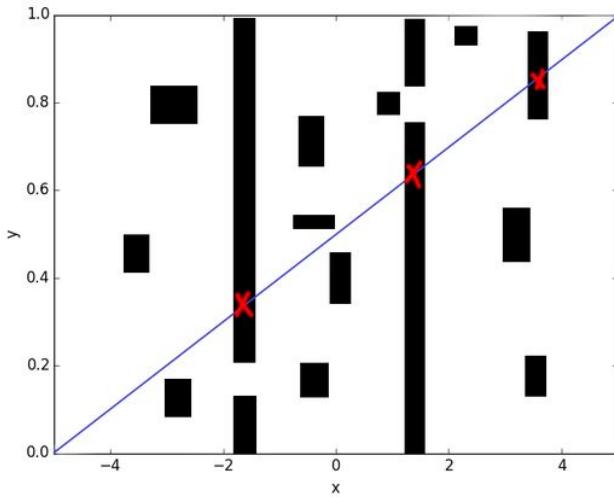


casi todos los problemas del mundo real son no-lineales

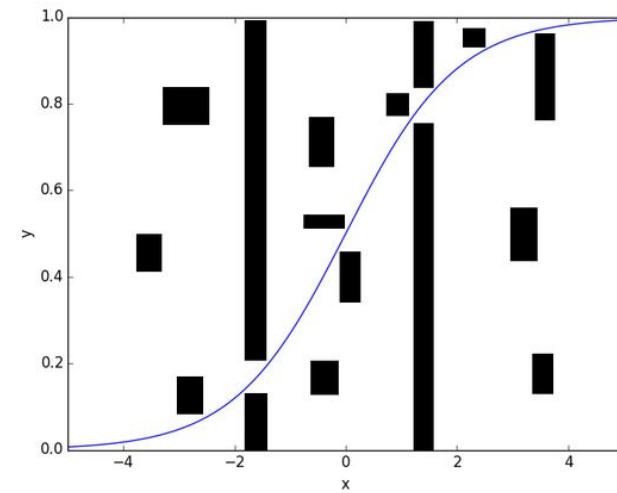


pero si introducimos una sigmoide, es bastante fácil de resolver

La necesidad de la no-linealidad

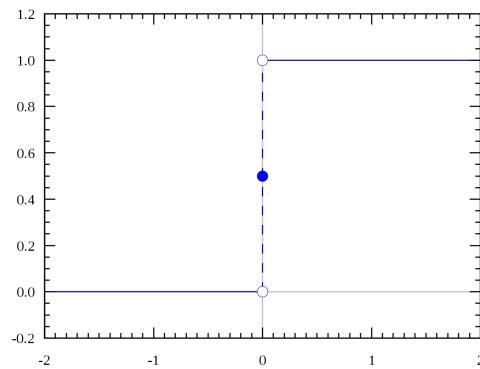


Sin una función de activación, tu red actuará más como un tubo de acero: fijo e inflexible.

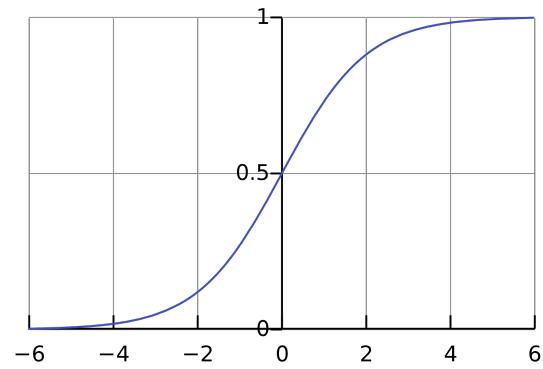


La función de activación otorga flexibilidad a la red, permitiéndole adaptarse a diferentes problemas

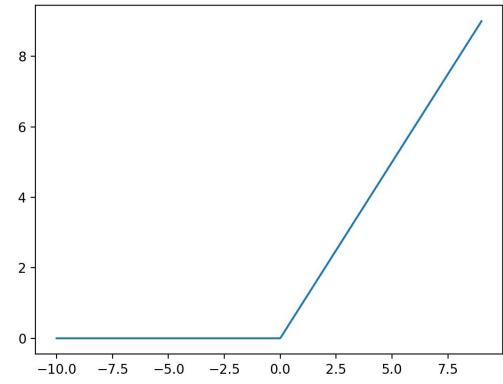
Funciones de activación comunes



$$g(x) = \begin{cases} 0 & \text{if } x > 0 \\ 1 & \text{if } x \leq 0 \end{cases}$$



$$g(x) = \frac{1}{1 + e^{-x}}$$

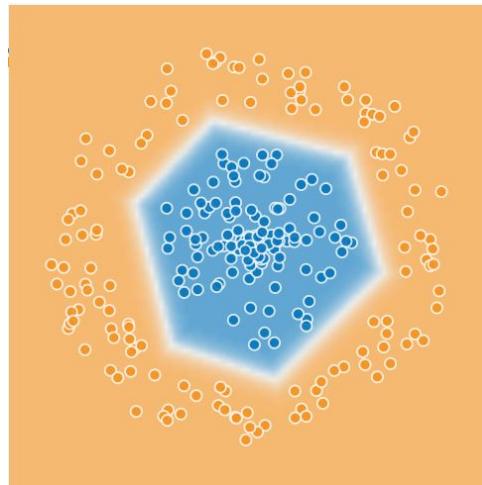


$$g(x) = \max(0, x)$$

¿Cómo escoger la función de activación?

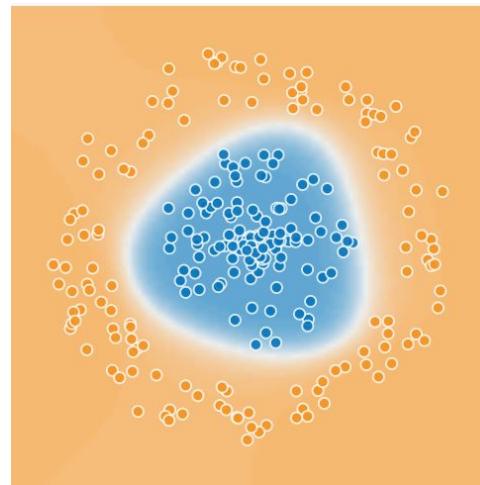
la función de activación define el comportamiento de las características aprendidas por la red

modelo entrenado con RELU



resulta en una
composición de rectas

modelo entrenado con SIGMOID



resulta en una
composición de curvas

demo tensorflow

<https://playground.tensorflow.org/>

Epoch 000,090 Learning rate 0.03 Activation Tanh Regularization None Regularization rate 0 Problem type Classification

DATA Which dataset do you want to use?

FEATURES Which properties do you want to feed in?

4 HIDDEN LAYERS

4 neurons 2 neurons 2 neurons 2 neurons

X₁ X₂ X₁₂ X₂₂ X_{1X2}

The outputs are mixed with varying weights, shown by the thickness of the lines.

This is the output from one neuron. Hover to see it larger.

REGENERATE sin(X₁)

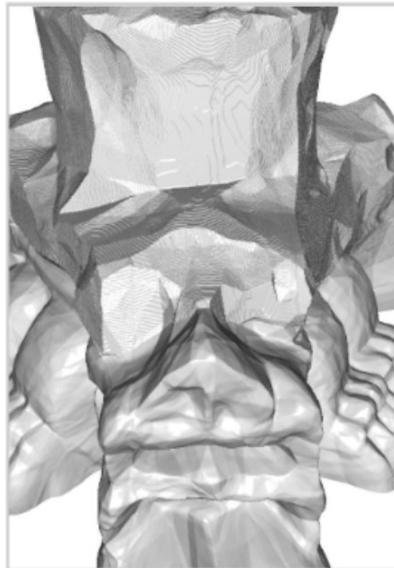
Test loss 0.479
Training loss 0.474

sin(X₁)

¿Cómo escoger la función de activación?

la función de activación define el comportamiento de las características aprendidas por la red

ReLU (baseline)

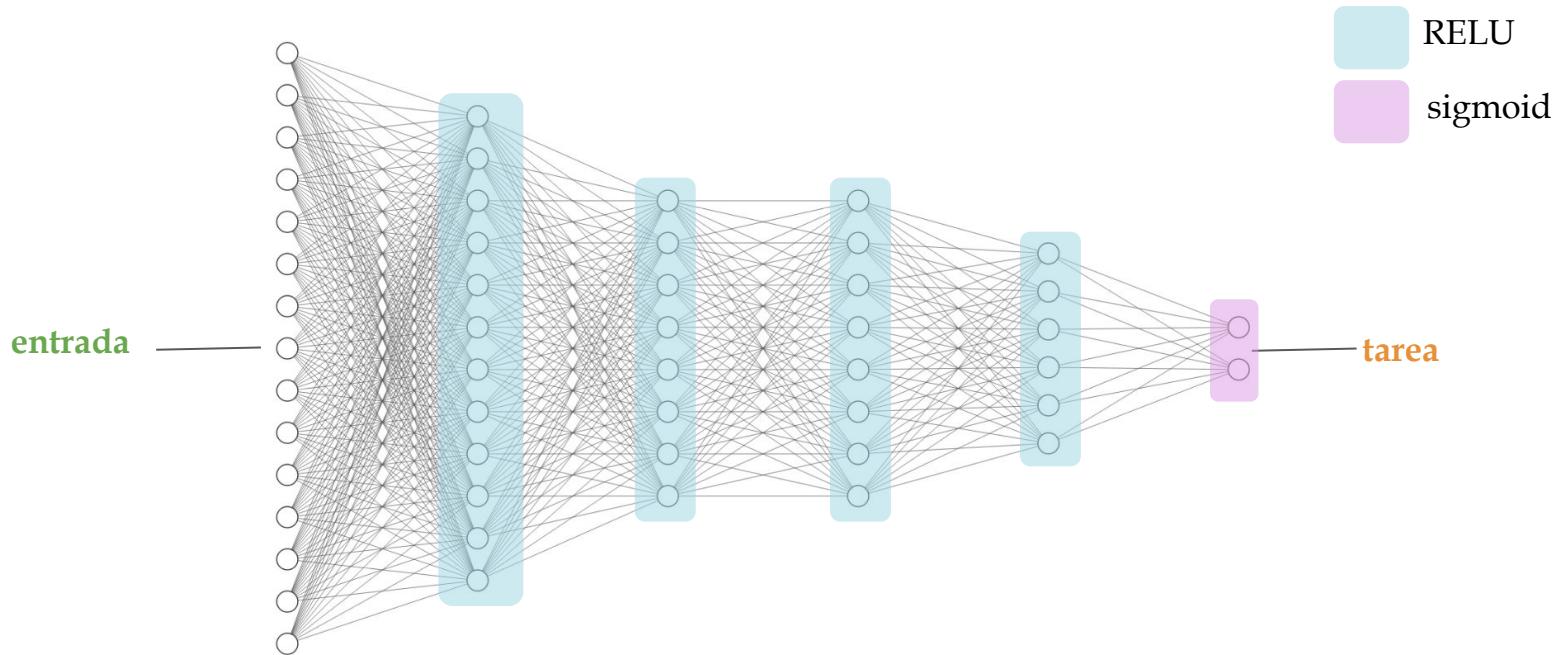


SIREN (ours)



¿Cómo escoger la función de activación?

habitualmente en la capas ocultas se usa la RELU y en la capa de salida, en la que se define la tarea se usa función de activación específica



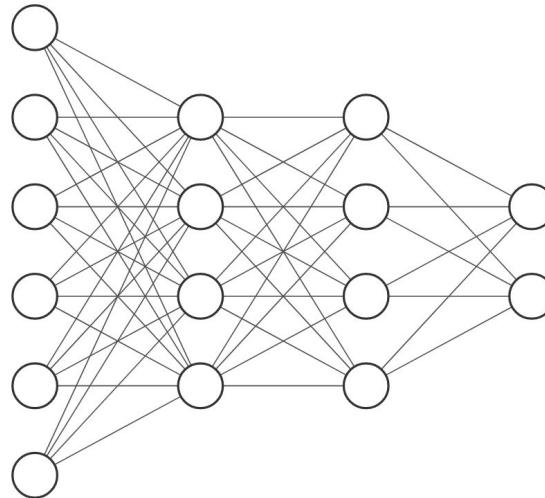
Función de costo

para escoger los pesos W debemos medir el error de nuestra red

Imagen Vectorizada



tensor de 3072×1
(3072 números)



perro 0.6
gato 0.4



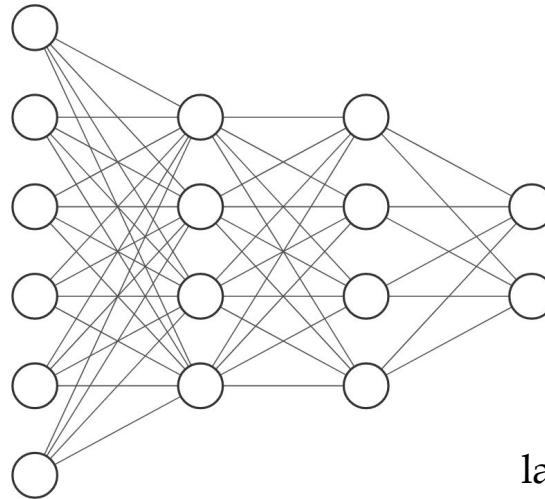
Función de costo

para escoger los pesos W debemos medir el error de nuestra red

Imagen Vectorizada



tensor de 3072×1
(3072 números)



perro 0.6
gato 0.4



la salida de la red debe ser:

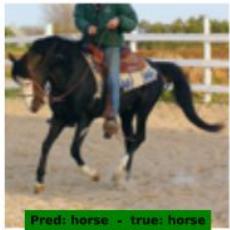
perro 0
gato 1



Función de costo

para escoger los pesos W debemos medir el error de nuestra red

$$\mathcal{L} = \frac{1}{N} \sum_i L_i(f(x_i, W), y_i)$$

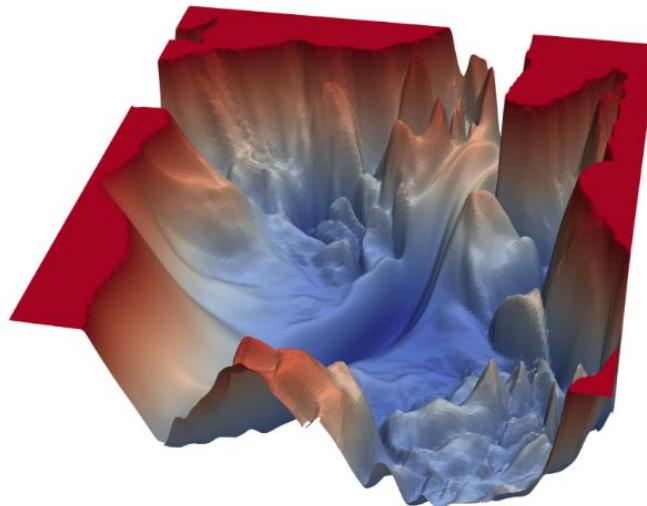


Función de costo

para escoger los pesos W debemos medir el error de nuestra red

$$\mathcal{L} = \frac{1}{N} \sum_i L_i(f(x_i, W), y_i)$$

la función de costo debe ser derivable



función de costo de una VGG16

Función de costo



Función de costo comunes

Binary Cross Entropy

Mean Squared Error

$$-\frac{1}{N} \sum_{i=1}^N y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)$$

Is used in binary **classification**

$$\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

Is used in **regression**

Backpropagation

Queremos encontrar los pesos de la red que logren el mejor rendimiento y la menor pérdida.

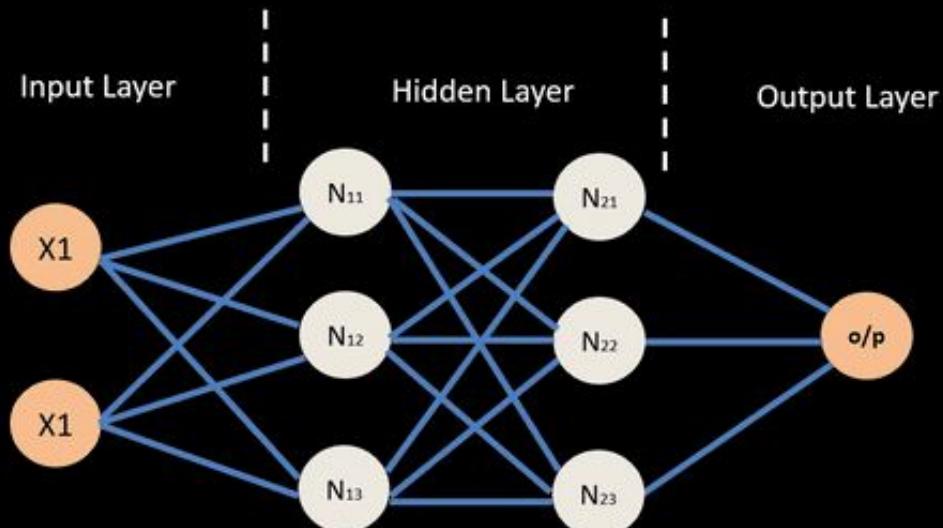
$$W^* = \underset{W}{\operatorname{argmin}} L(W)$$



optimizamos W para encontrar los pesos óptimos

Backpropagation

Neural Network – Backpropagation



Optimizadores: SGD

Queremos encontrar los pesos de la red que logren el mejor rendimiento y la menor pérdida.

$$W^* = \underset{W}{\operatorname{argmin}} L(W)$$



optimizamos W para encontrar los pesos óptimos

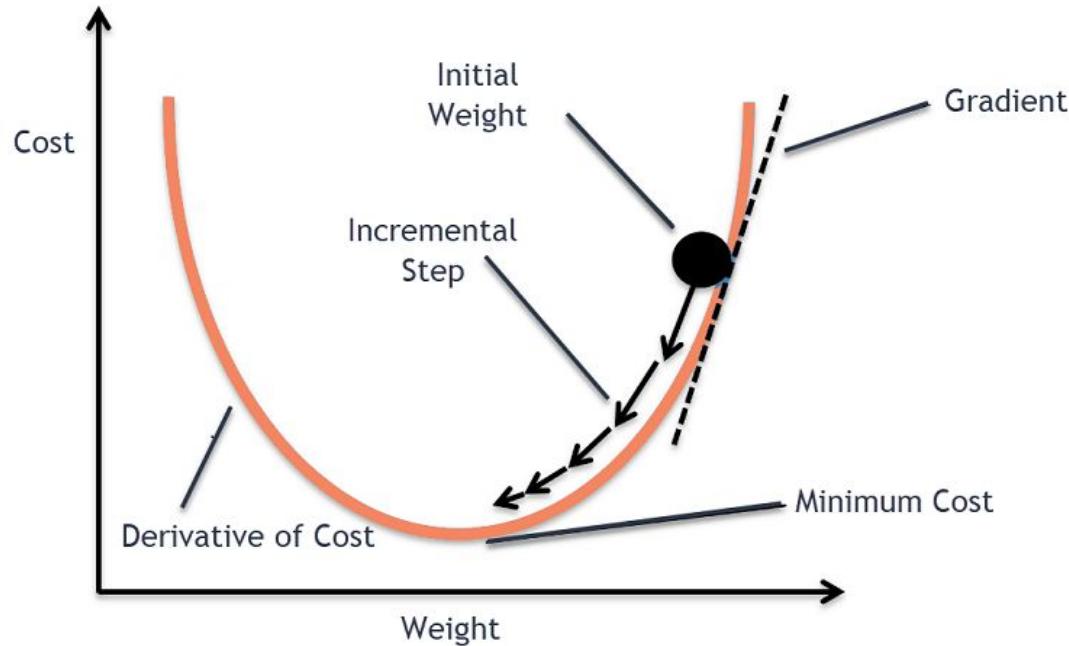
Optimizadores: SGD

se empieza con pesos aleatorios W

en cada iteración calculamos la derivada

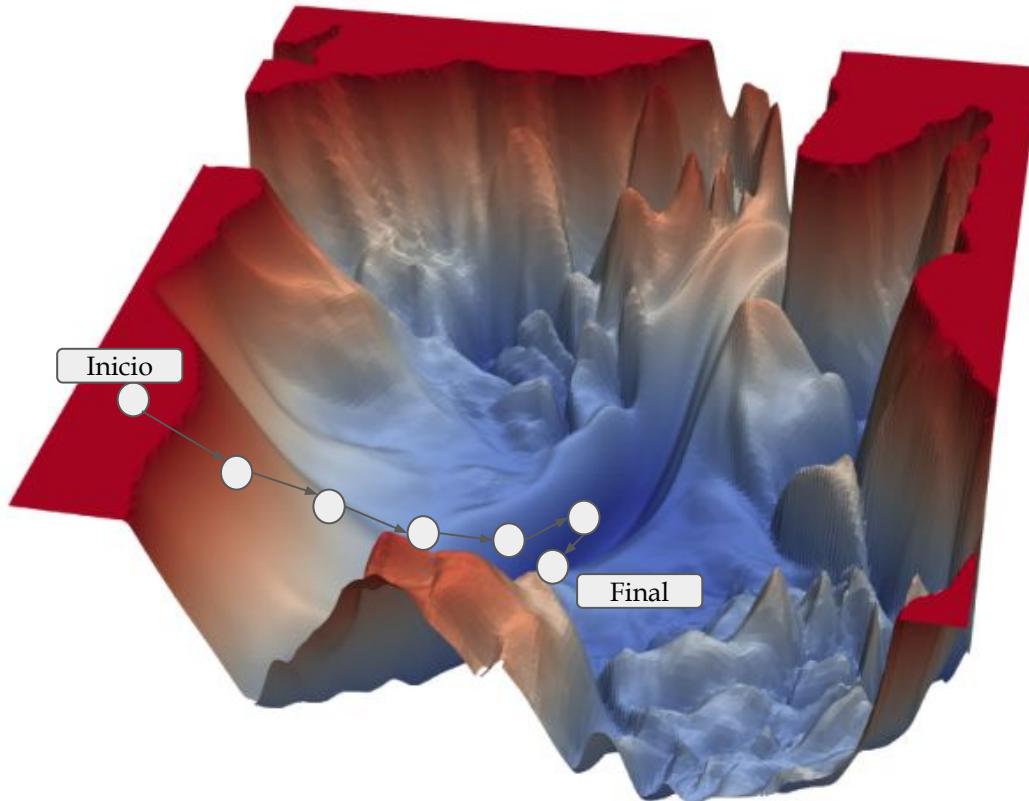
$$\nabla_W = \frac{\partial \mathcal{L}(W)}{\partial W}$$

este es el
gradiente!



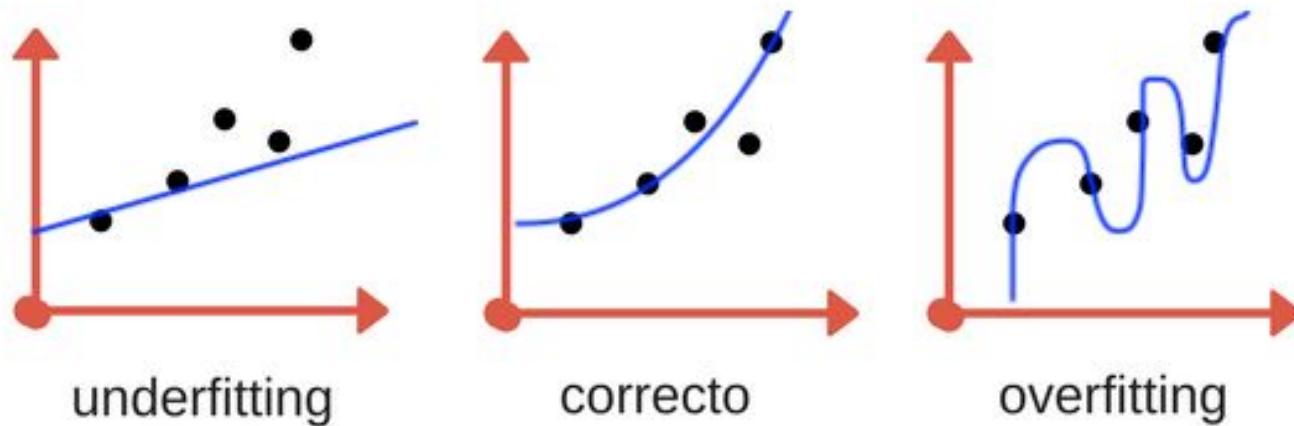
el gradiente se calcula con la regla de la cadena

Optimizadores: SGD



función de costo de una VGG16

overfitting, underfitting

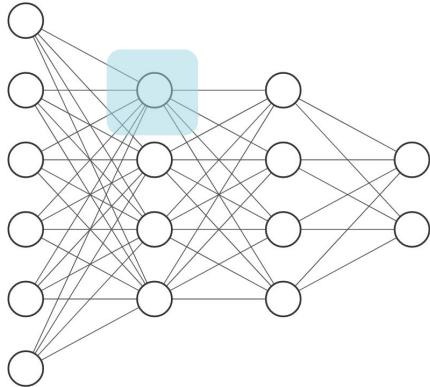


una interpretación de W^*

W tiene codificada información subyacente que es extraída a partir de los datos



Unsupervised sentiment neuron



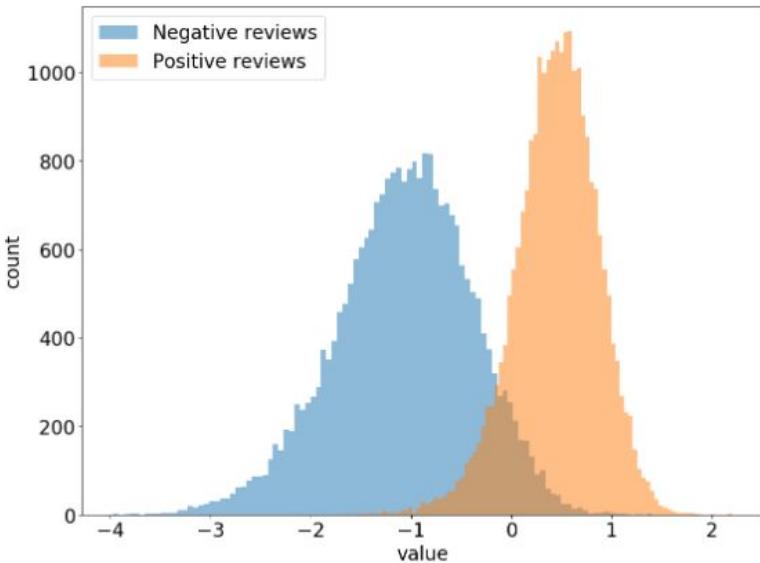
relative contributions of features on various datasets, we discovered a single unit within the mLSTM that directly corresponds to sentiment. In Figure 3 we show the his-



una interpretación de W*

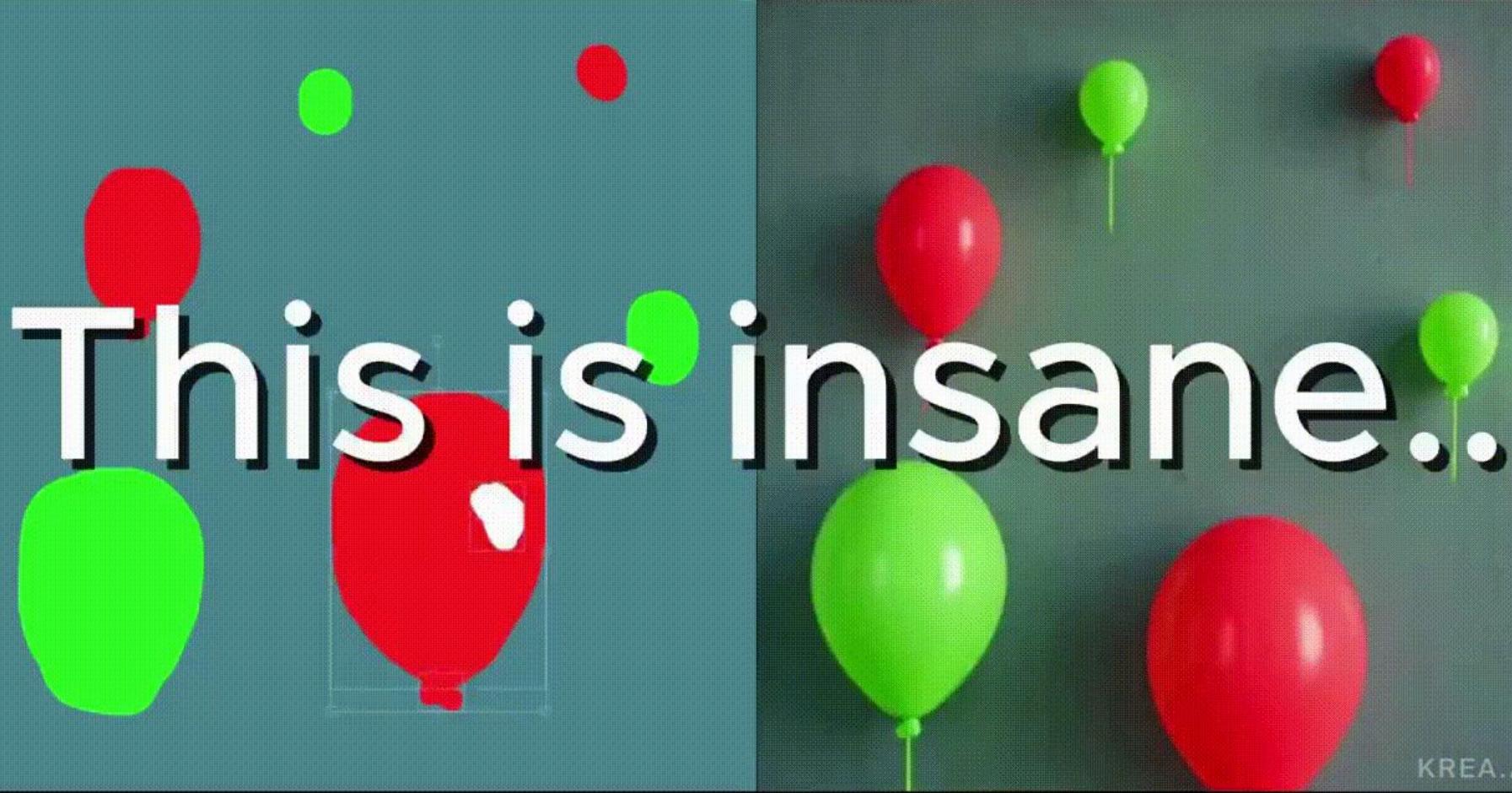
W tiene codificada información subyacente que es extraída a partir de los datos

Unsupervised sentiment neuron



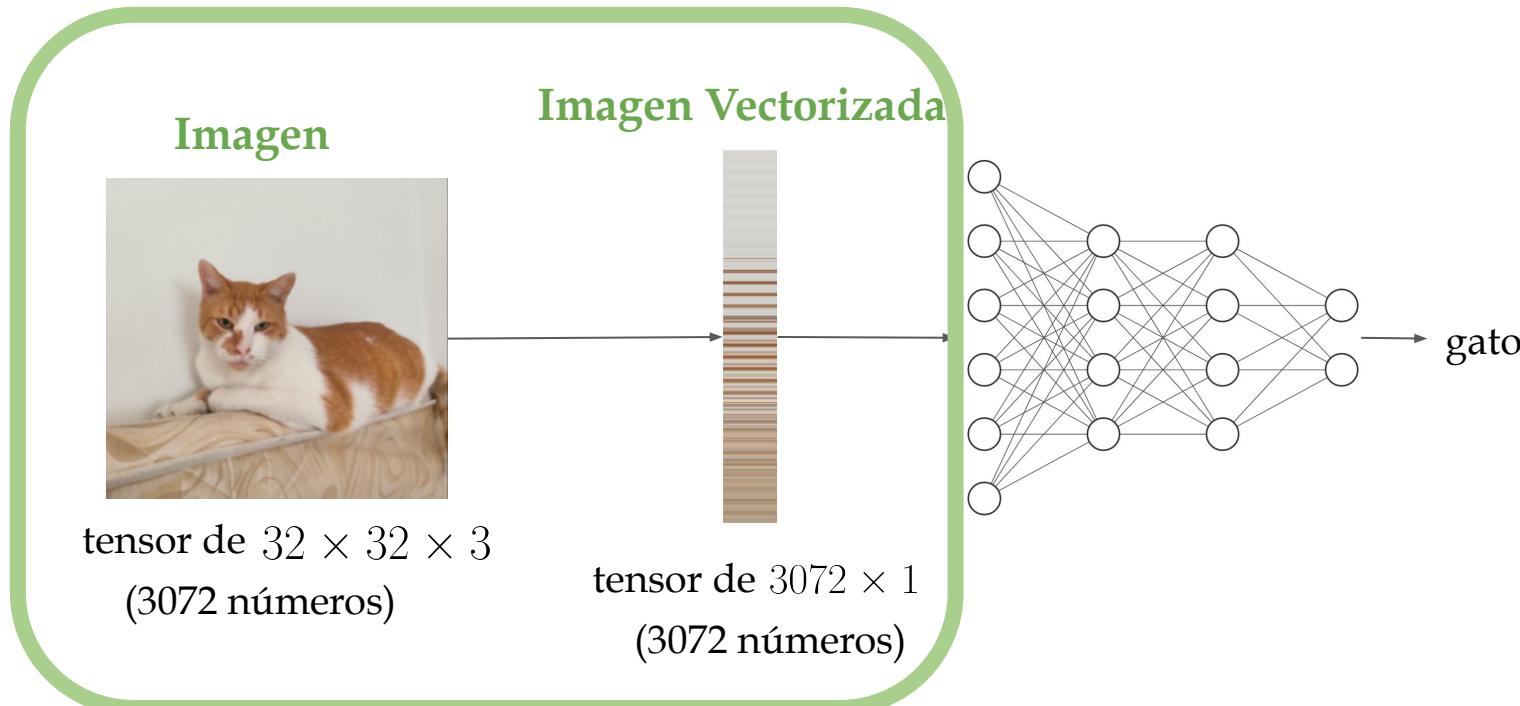
Judy Holliday struck gold in 1950 with George Cukor's film version of "Born Yesterday," and from that point forward, her career consisted of trying to find material good enough to allow her to strike gold again. It never happened. In "It Should Happen to You" (I can't think of a blander title, by the way), Holliday does yet one more variation on the dumb blonde who's maybe not so dumb after all, but everything about this movie feels warmed over and half hearted. Even Jack Lemmon, in what I believe was his first film role, can't muster up enough energy to enliven this recycled comedy. The audience knows how the movie will end virtually from the beginning, so mostly it just sits around waiting for the film to catch up. Maybe if you're enamored of Holliday you'll enjoy this; otherwise I wouldn't bother. Grade: C

una interpretación de W*



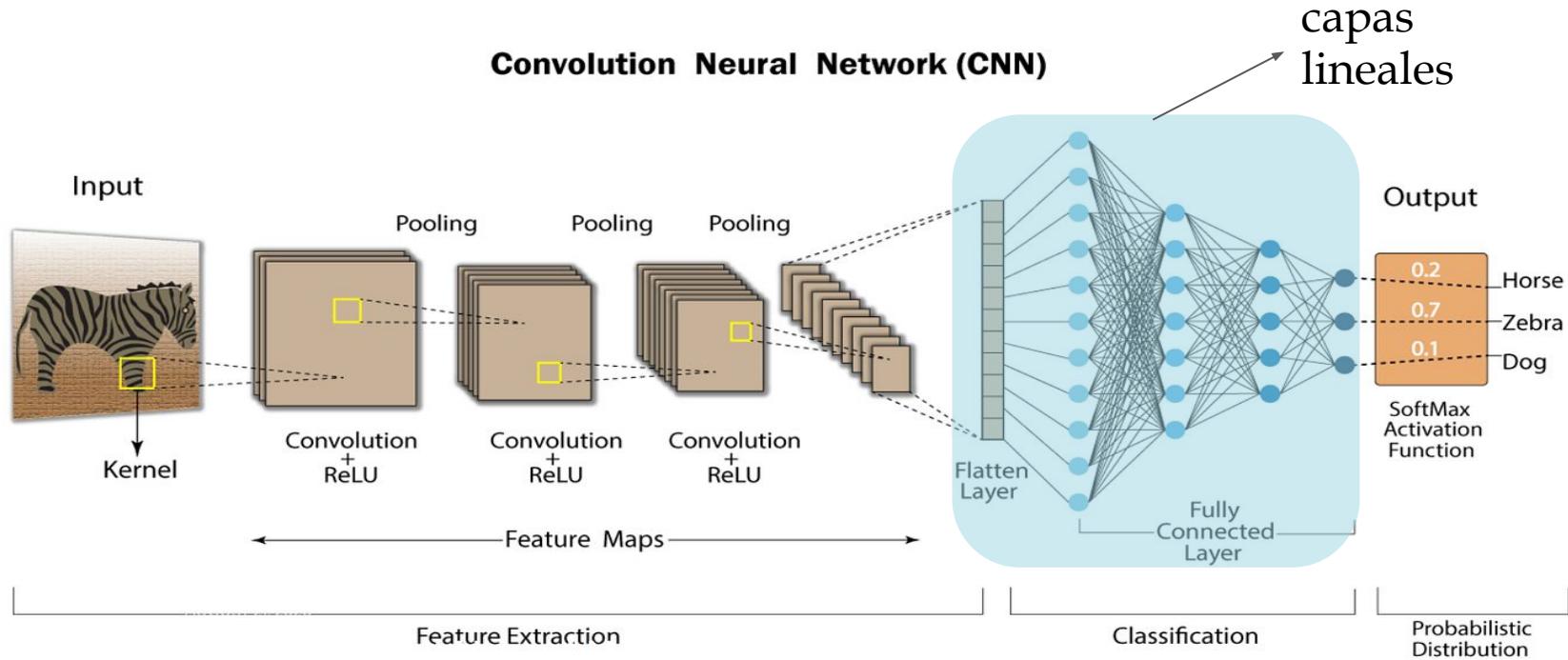
El problema de las capas lineales

el proceso de vectorización en imágenes destruye la información y relaciones espaciales



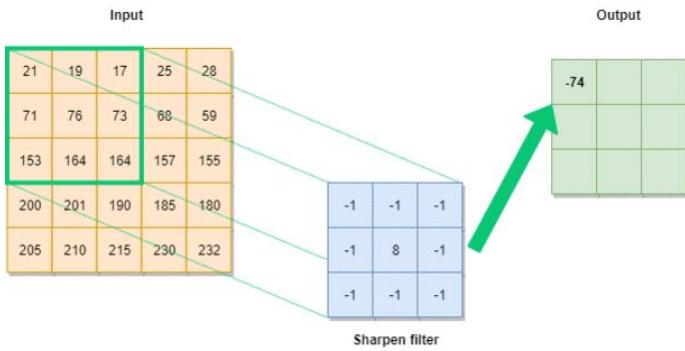
Redes Convolucionales

Las redes convolucionales solucionan este problema

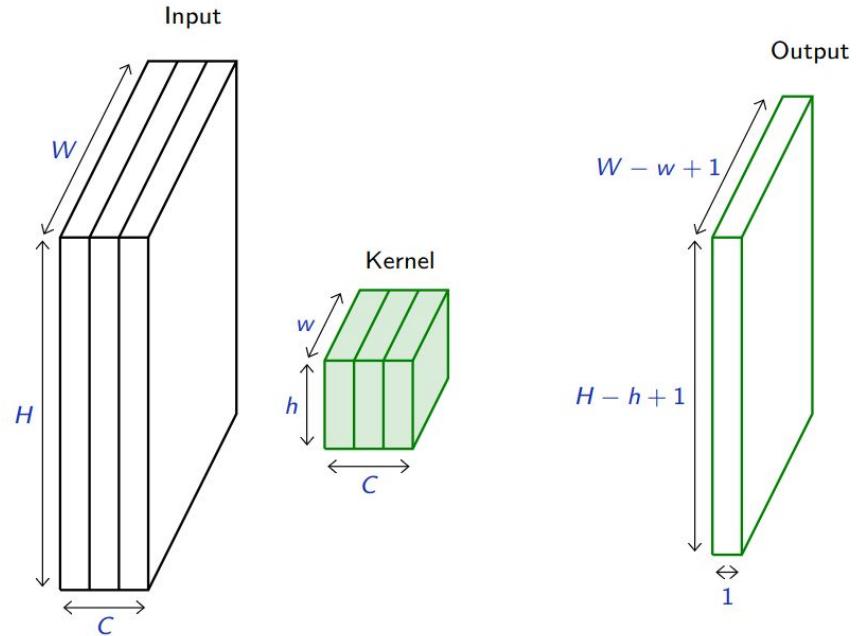


Redes Convolucionales

cada capa convolucional puede tener N cantidad de filtros diferentes



AIGeekProgrammer.com © 2019



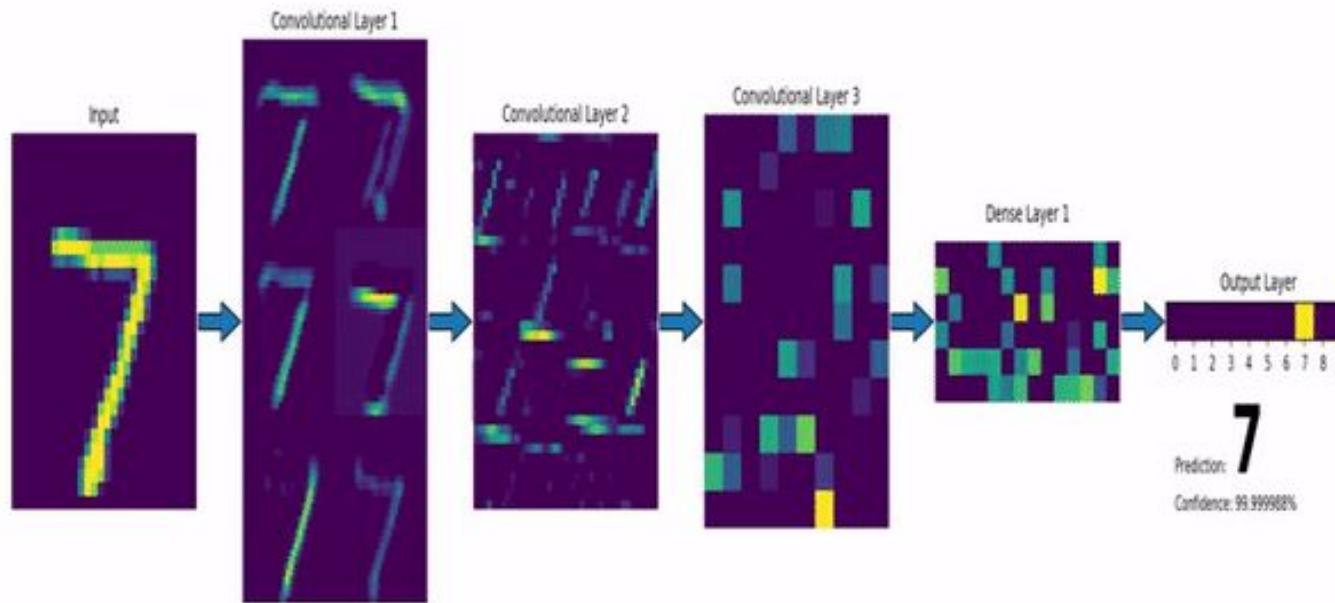
Redes Convolucionales



Input

Capas de filtros

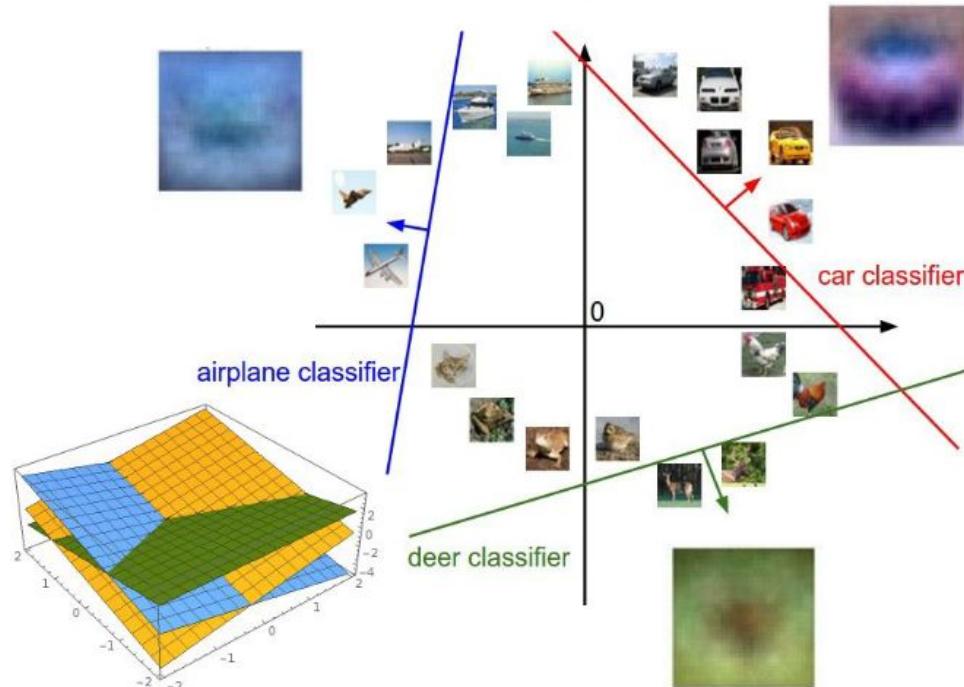
Cada capa se encarga de extraer características únicas de la imagen



En las capas finales de una red convolucional, se resumen las características aprendidas para tomar decisiones de alto nivel

Capas de filtros

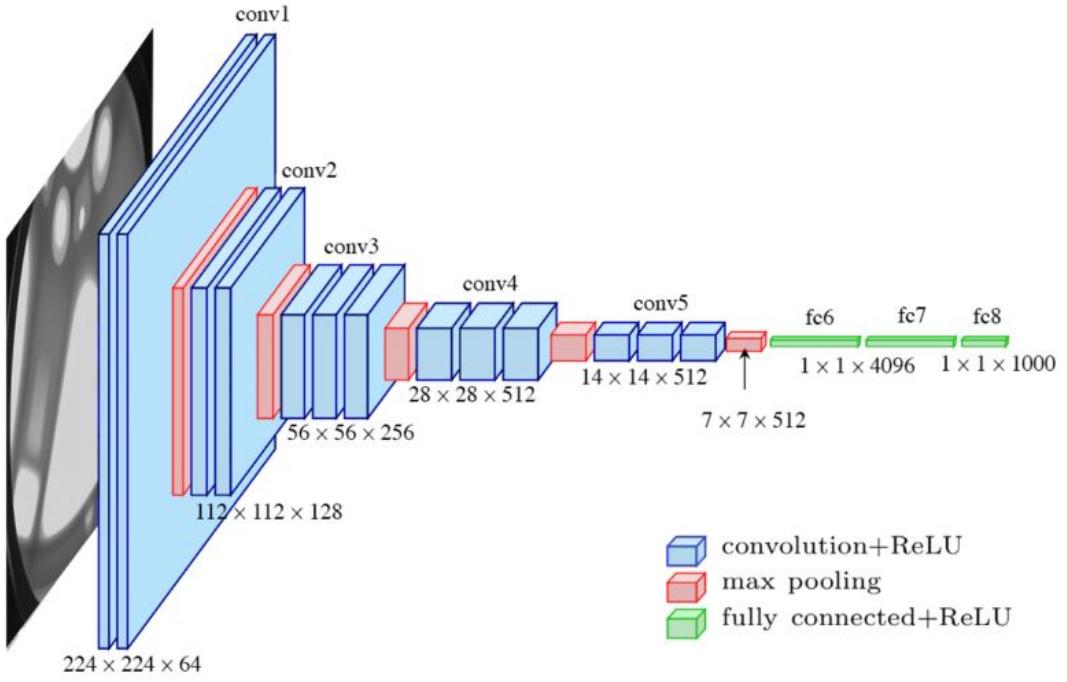
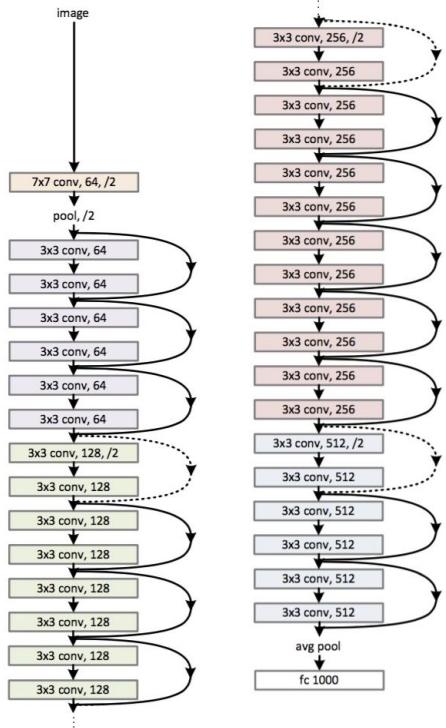
campo de decisión de una CNN



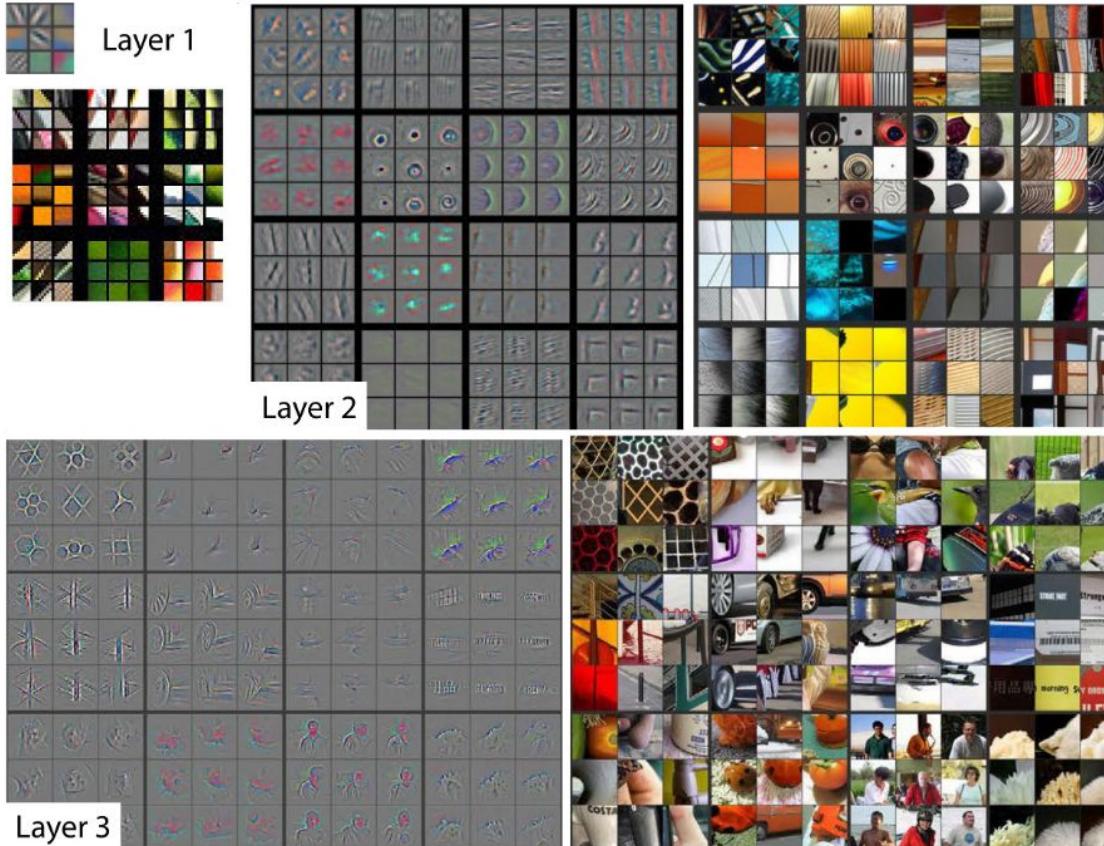
Deep Convolutional Networks

se han propuesto mucho tipos de arquitecturas CNN para cada tarea

34-layer residual

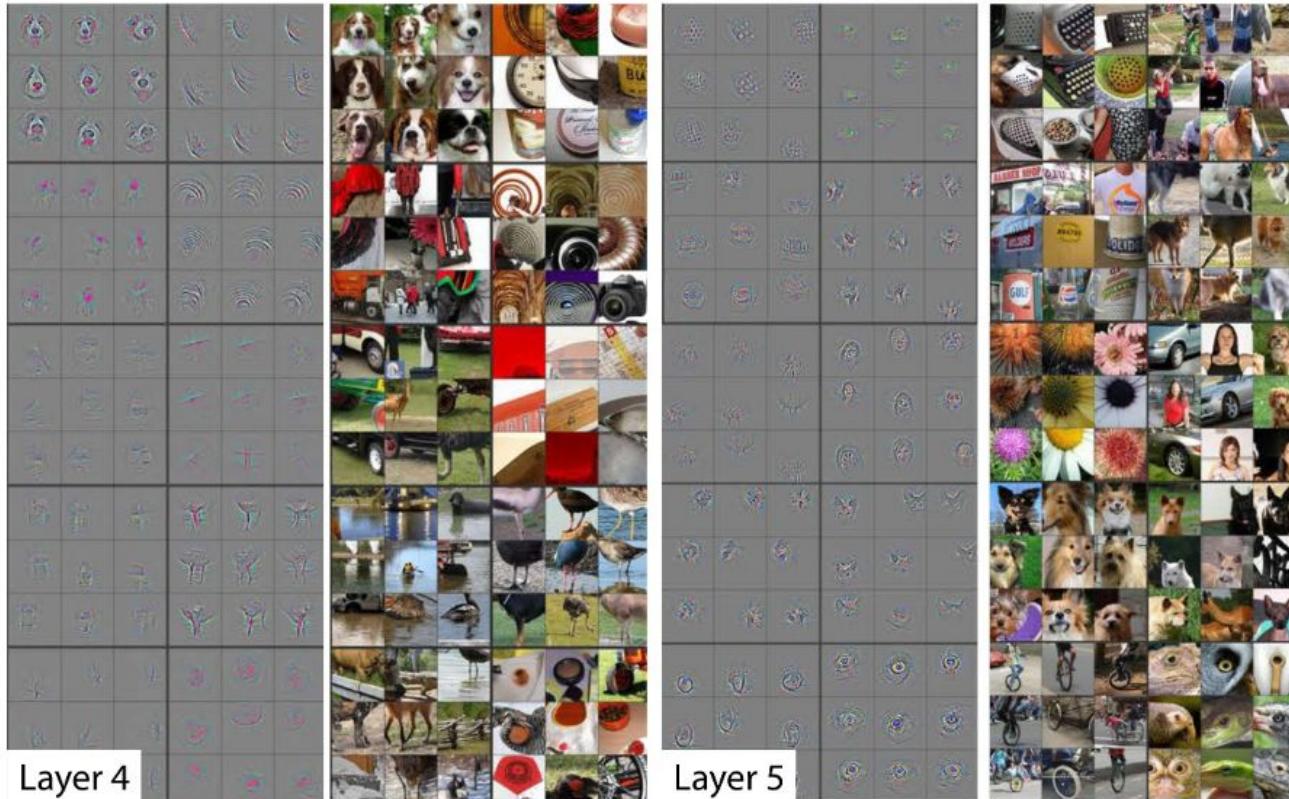


Deep Convolutional Networks



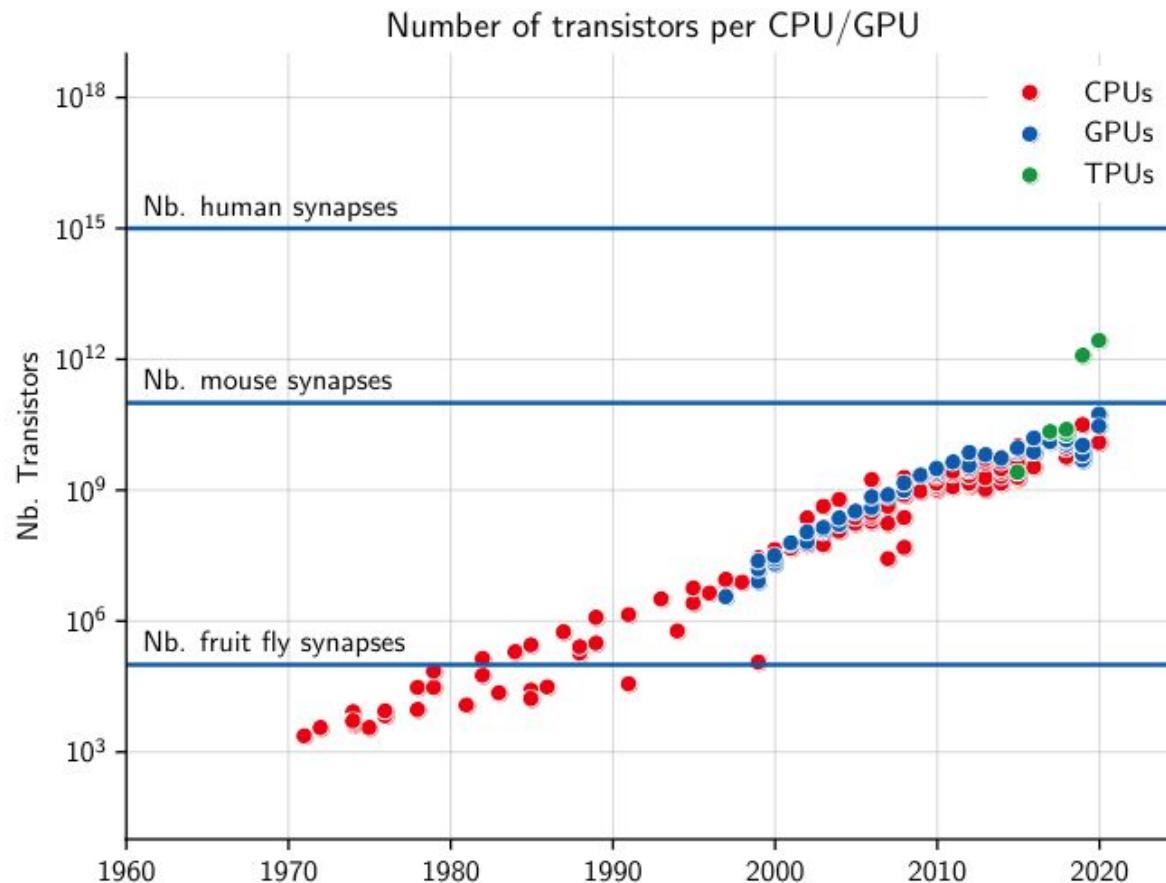
(Zeiler and Fergus, 2014)

Deep Convolutional Networks



(Zeiler and Fergus, 2014)

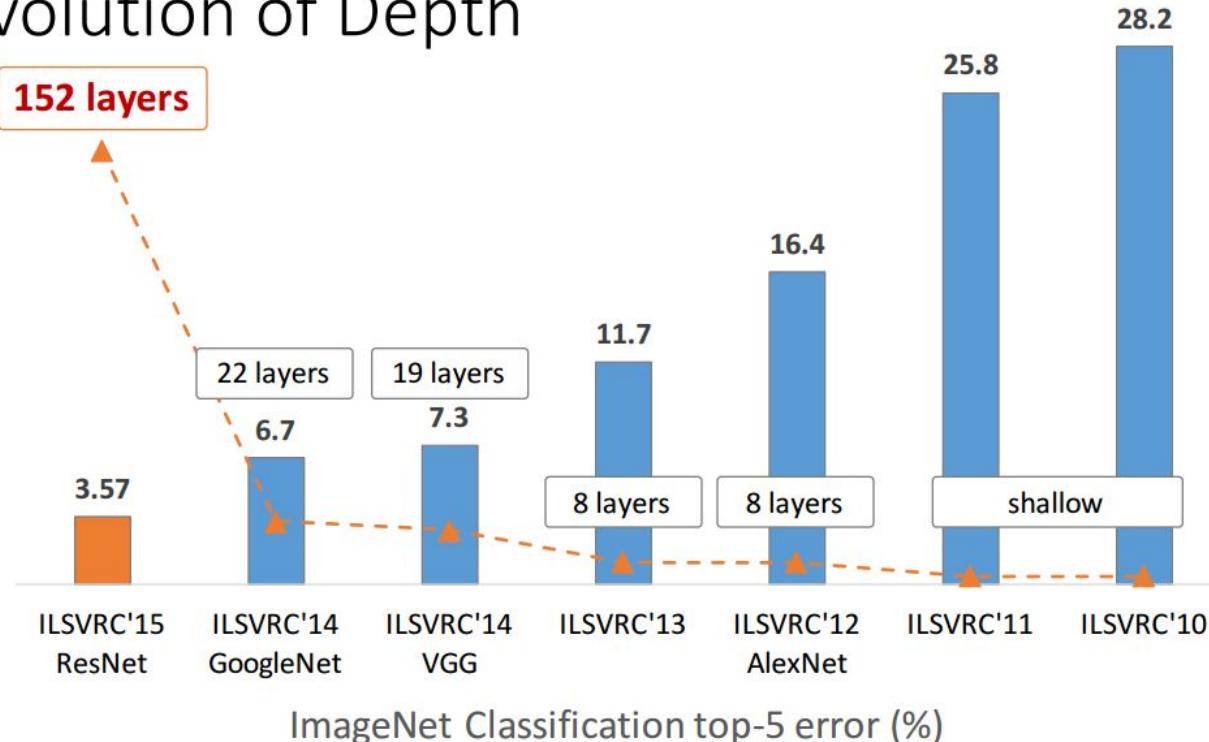
Escalabilidad



Escalabilidad

más capas lleva a un mejor rendimiento

Revolution of Depth



Deep Learning: Hoy



2014

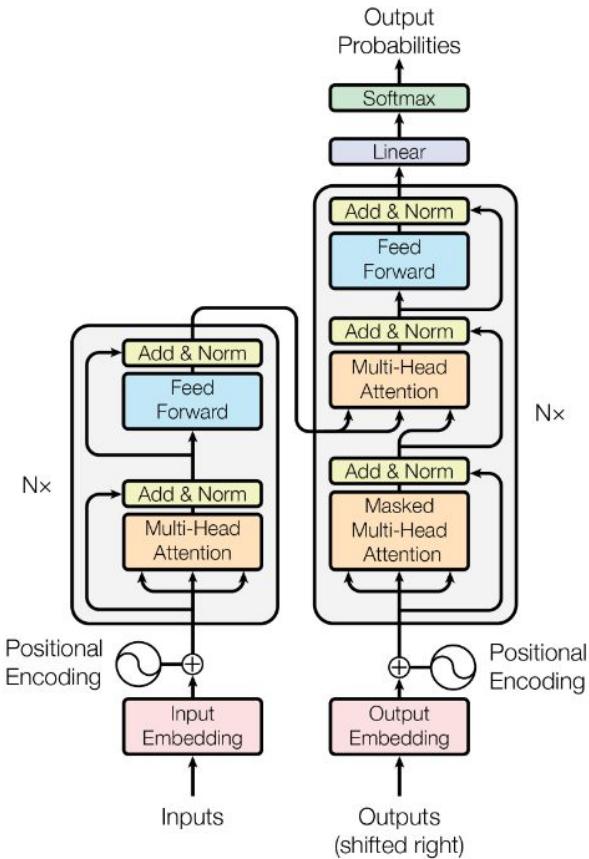


2019



2022

Deep Learning: Hoy



La receta

Hardware



CPUs/GPUs/almacenamiento
desarrollados

Datos



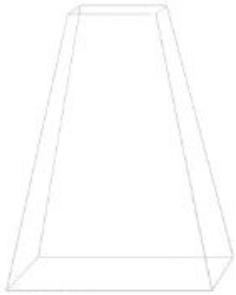
un montón de datos de
“internet”

Mejoras algorítmicas



un montón de datos de
“internet”

Multimodalidad



?

El deep learning está en todos lados



El deep learning es una herramienta general que se puede aplicar para cualquier tarea

NVIDIA GTC 2024

Workshops March 17-21 | AI Conference and Expo March 18-21 | Keynote March 18 | San Jose, CA and Virtual EN

Keynote Session Catalog Agenda Attend Sponsors & Exhibitors More Log In Join Now

NVIDIA GTC 2024 Keynote

Don't Miss This Transformative Moment in AI

Watch NVIDIA CEO Jensen Huang's GTC keynote to catch all the announcements on AI advances that are shaping our future.

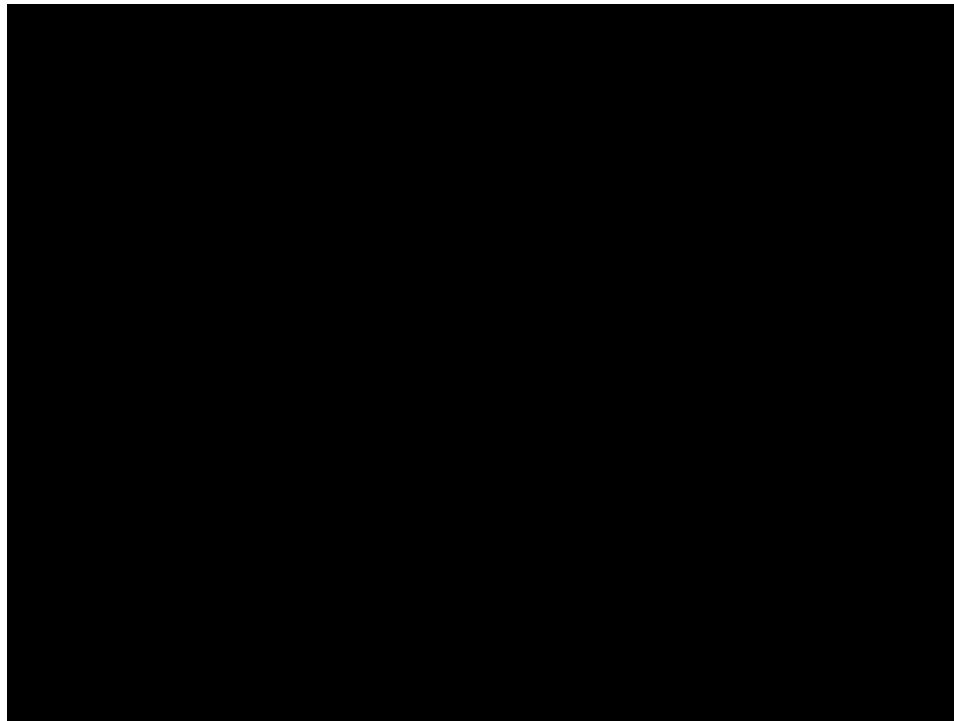
Monday, March 18
1-3 p.m. PDT

Hands-on

1. Preparación de datos
2. Definición arquitectura
3. Loss y Optimizador
4. Bucle entrenamiento
5. Evaluación

Hands-on

aproximemos la función $\sin(x)$



Hands-on

clasificacion de imagenes con una red neuronal convolucional

```
Epoch: 0: 100% [██████████] 782/782 [01:10<00:00, 11.02it/s]
Epoch [1/30], Average Loss:for epoch[1, 2.0429]
Epoch: 1: 100% [██████████] 782/782 [01:11<00:00, 10.94it/s]
Epoch [2/30], Average Loss:for epoch[2, 1.8800]
Epoch: 2: 100% [██████████] 782/782 [01:10<00:00, 11.15it/s]
Epoch [3/30], Average Loss:for epoch[3, 1.8318]
Epoch: 3: 100% [██████████] 782/782 [01:11<00:00, 10.98it/s]
Epoch [4/30], Average Loss:for epoch[4, 1.8073]
Epoch: 4: 100% [██████████] 782/782 [01:10<00:00, 11.09it/s]
Epoch [5/30], Average Loss:for epoch[5, 1.7886]
Epoch: 5: 100% [██████████] 782/782 [01:10<00:00, 11.08it/s]
Epoch [6/30], Average Loss:for epoch[6, 1.7773]
Epoch: 6: 100% [██████████] 782/782 [01:10<00:00, 11.11it/s]
Epoch [7/30], Average Loss:for epoch[7, 1.7676]
Epoch: 7: 100% [██████████] 782/782 [01:11<00:00, 11.00it/s]
Epoch [8/30], Average Loss:for epoch[8, 1.7589]
Epoch: 8: 100% [██████████] 782/782 [01:09<00:00, 11.23it/s]
Epoch [9/30], Average Loss:for epoch[9, 1.7521]
Epoch: 9: 100% [██████████] 782/782 [01:10<00:00, 11.03it/s]
Epoch [10/30], Average Loss:for epoch[10, 1.7481]
Epoch: 10: 100% [██████████] 782/782 [01:10<00:00, 11.15it/s]
Epoch [11/30], Average Loss:for epoch[11, 1.7404]
Epoch: 11: 100% [██████████] 782/782 [01:10<00:00, 11.17it/s]
Epoch [12/30], Average Loss:for epoch[12, 1.7354]
Epoch: 12: 100% [██████████] 782/782 [01:09<00:00, 11.18it/s]
Epoch [13/30], Average Loss:for epoch[13, 1.7316]
Epoch: 13: 100% [██████████] 782/782 [01:21<00:00, 9.54it/s]
Epoch [14/30], Average Loss:for epoch[14, 1.7271]
Epoch: 14: 100% [██████████] 782/782 [01:10<00:00, 11.09it/s]
Epoch [15/30], Average Loss:for epoch[15, 1.7235]
Epoch: 15: 100% [██████████] 782/782 [01:15<00:00, 10.41it/s]
```



ship



dog



plane



plane



frog



car



bird

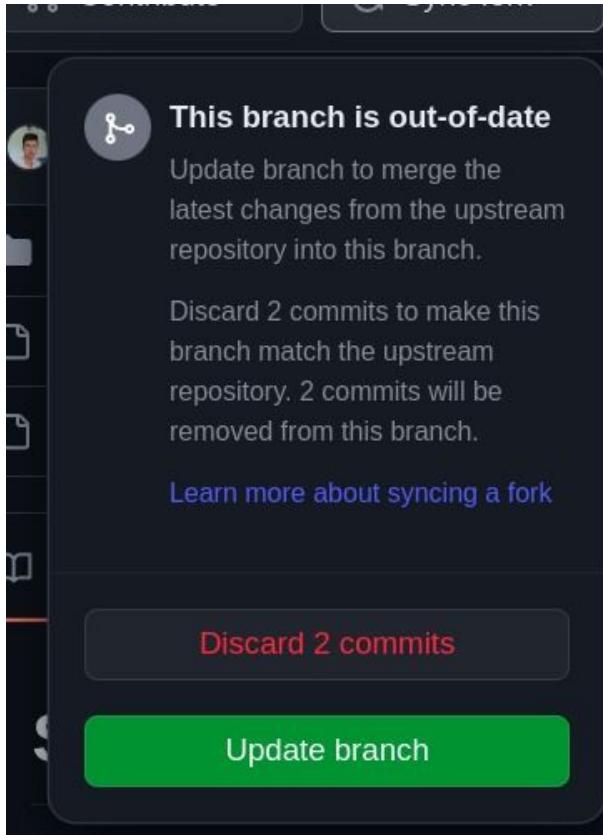


ship

Actualicemos el repositorio!

A screenshot of a GitHub repository page. At the top, there are navigation links: 'main' (with a dropdown arrow), '1 Branch' (with a dropdown arrow), '0 Tags' (with a dropdown arrow). To the right are search bar ('Go to file'), a user icon ('t'), 'Add file' (with a dropdown arrow), and a green 'Code' button. Below this, a message states: 'This branch is 2 commits ahead of, 3 commits behind semilleroCV/Hands-on-Computer-Vision:main.' At the bottom left, there are two buttons: 'Contribute' (with a dropdown arrow) and 'Sync fork' (with a dropdown arrow). The 'Sync fork' button is highlighted with a thick red border.

Actualicemos el repositorio!



Abrelo en colab usando “githubtocolab”

🔗 <https://github.com/semilleroCV/Hands-on-Computer-Vision/blob/main/Sesiones/Sesion1/Exercise1.ipynb>



Agrega la palabra “tocolab” y da ENTER

🔍 <https://githubtocolab.com/semilleroCV/Hands-on-Computer-Vision/blob/main/Sesiones/Sesion1/Exercise1.ipynb>