



Rodolfo Ferro

ferro@cimat.mx

<https://rodolfoferro.xyz>

*Diplomado en Ciencia de Datos*

*Escuela Nacional de Estudios Superiores, Unidad León*

Unidad León  
Escuela  
Nacional de  
Estudios  
Superiores

Agosto, 2024

# Aprendizaje profundo

*Módulo 5*

### Rodolfo Ferro (ferro@cimat.mx)

- › Sr. SWE (Data Engineer) @ Bionic México
- › Miembro del Consejo Consultivo para el Desarrollo Económico, Creatividad e Innovación de León
- › **Formación:** BMath, CSysEng, StatMethodsSpc (*ongoing*)
- › **Experiencia:** ML Engineer @ Vindoo.ai (España), Sherpa Digital en IA @ Microsoft México, AI Research Assistant @ CIMAT & AI Research Intern @ Harvard.



ferro@cimat.mx

# Tabla de contenidos

- 1 Intro al aprendizaje profundo
- 2 Visión computacional profunda

- 3 Modelado profundo de secuencias
- 4 Modelado generativo profundo
- 5 Panorama actual y futuro

## 1 Intro al aprendizaje profundo

- Motivación
- Introducción
- Contexto histórico
- Perceptrón
- Perceptrón multicapa
- Aprendizaje
- Regularización

## 2 Visión computacional profunda

- Introducción a imágenes
- Espacios de color
- Convoluciones & Pooling
- Redes neuronales convolucionales
- Clasificadores de imágenes (LeNet, VGG16, etc.)
- Autoencoders
- Trabajos relacionados y avances recientes

## 3 Modelado profundo de secuencias

## 4 Modelado generativo profundo

## 5 Panorama actual y futuro



## Intro al aprendizaje profundo



```
if(speed<4){  
    status=WALKING;  
}
```

## Intro al aprendizaje profundo



```
if(speed<4){  
    status=WALKING;  
}
```



```
if(speed<4){  
    status=WALKING;  
} else {  
    status=RUNNING;  
}
```

## Intro al aprendizaje profundo



```
if(speed<4){  
    status=WALKING;  
}
```



```
if(speed<4){  
    status=WALKING;  
} else {  
    status=RUNNING;  
}
```



```
if(speed<4){  
    status=WALKING;  
} else if(speed<12){  
    status=RUNNING;  
} else {  
    status=BIKING;  
}
```

## Intro al aprendizaje profundo



```
if(speed<4){  
    status=WALKING;  
}
```



```
if(speed<4){  
    status=WALKING;  
} else {  
    status=RUNNING;  
}
```



```
if(speed<4){  
    status=WALKING;  
} else if(speed<12){  
    status=RUNNING;  
} else {  
    status=BIKING;  
}
```

// ????



# ¿Qué es el *Deep Learning*?

Intro al aprendizaje profundo

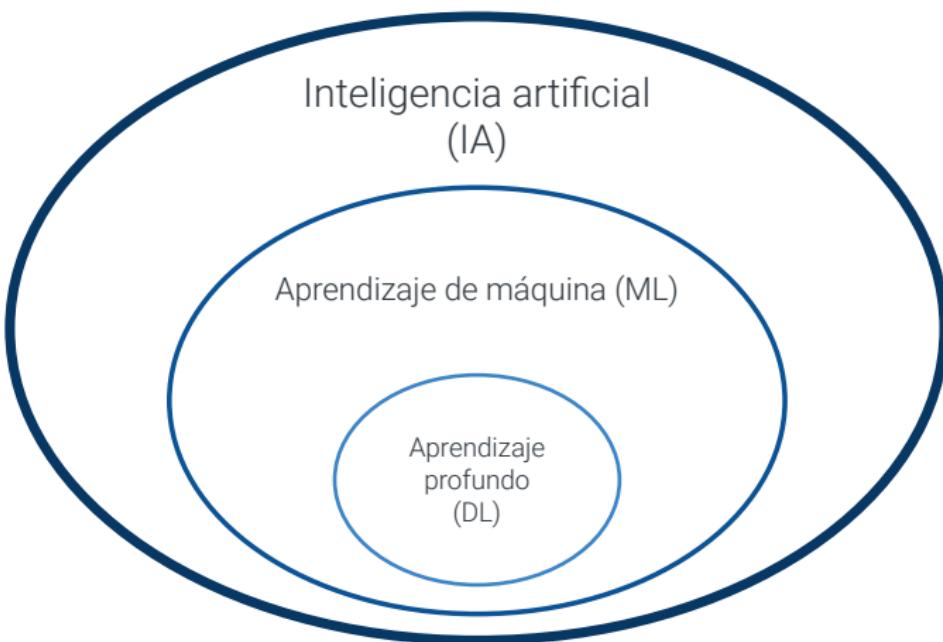
*El aprendizaje profundo (Deep Learning) comprende algoritmos de Machine Learning que (particularmente) utilizan múltiples capas apiladas de unidades de procesamiento para aprender representaciones en un alto nivel sobre datos no estructurados.*

David Foster (Generative Deep Learning)



# ¿Qué es el *Deep Learning*?

## Intro al aprendizaje profundo



ferro@cimat.mx

# ¿Qué es el *Deep Learning*?

Intro al aprendizaje profundo

- › **Inteligencia artificial:** Cualquier técnica que permita a las computadoras emular o imitar el comportamiento humano.
- › **Aprendizaje de máquina:** Capacidad de aprender sin ser programado explícitamente, enfoque en los algoritmos y la matemática.
- › **Aprendizaje profundo:** Extrae patrones de datos utilizando redes neuronales.



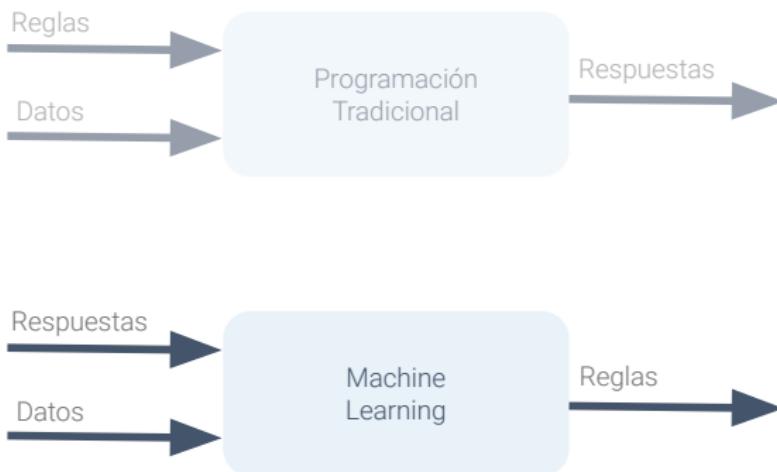
# ¿Por qué el *Deep Learning*?

## Intro al aprendizaje profundo



# ¿Por qué el *Deep Learning*?

## Intro al aprendizaje profundo



# ¿Por qué el *Deep Learning*?

Intro al aprendizaje profundo

- La ingeniería de características requiere mucho tiempo, es susceptible a errores y no es escalable consistentemente con datos complejos. Mejor busquemos aprender las características subyacentes directamente de los datos.
- Las redes neuronales artificiales existen desde hace décadas, pero su predominio actual reside principalmente en los siguientes tres aspectos:
  - 1 Hardware (GPUs, etc. + Paralelización)
  - 2 Software (Frameworks para trabajar con NNs)
  - 3 Grandes cantidades de datos
- El aprendizaje profundo está revolucionando muchos campos.

## Intro al aprendizaje profundo



ferrero@cimat.mx

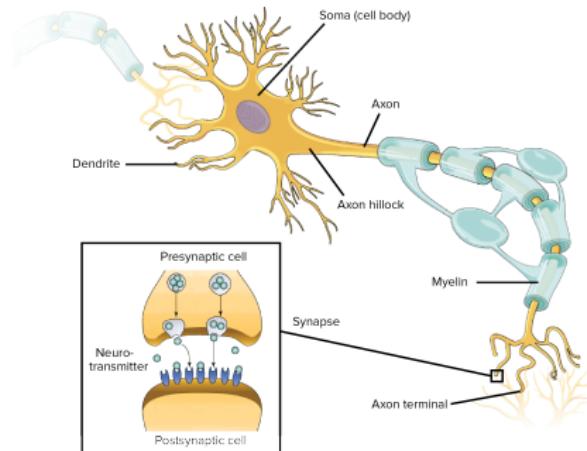
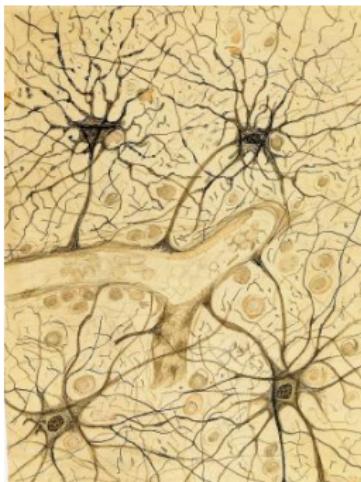
Figure: Santiago Ramón y Cajal



Escuela  
Nacional de  
Estudios  
Superiores

# Contexto histórico

## Intro al aprendizaje profundo



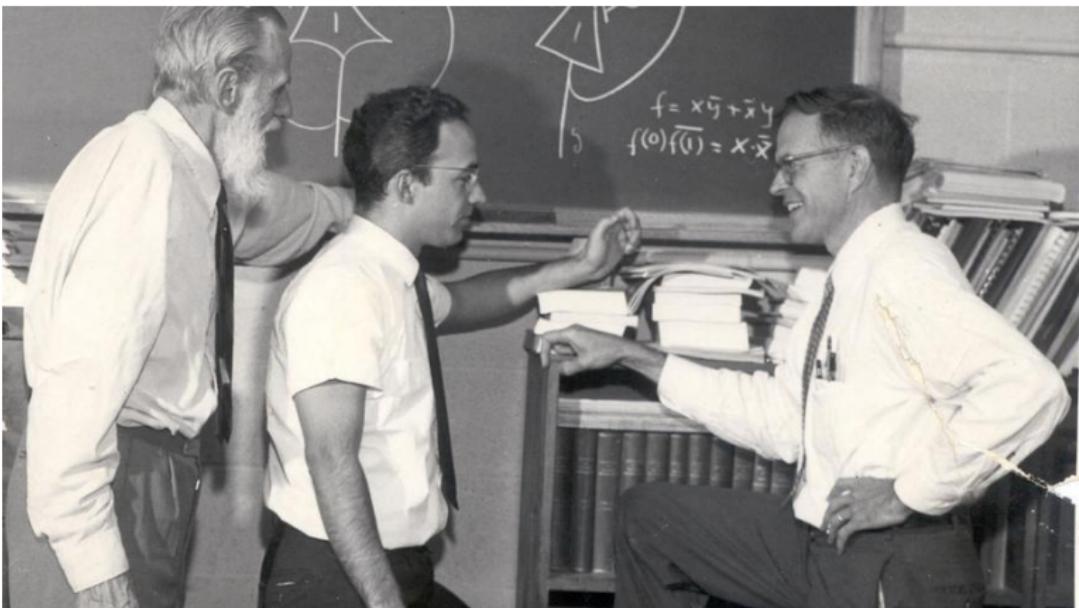
ferrero.cimat.mx

"Santiago Ramón y Cajal Drawings." Janelia Research Campus. Accessed July 5, 2024. <https://www.janelia.org>

"Overview of Neuron Structure and Function (Article)." Khan Academy. Accessed July 5, 2024. <https://www.khanacademy.org>



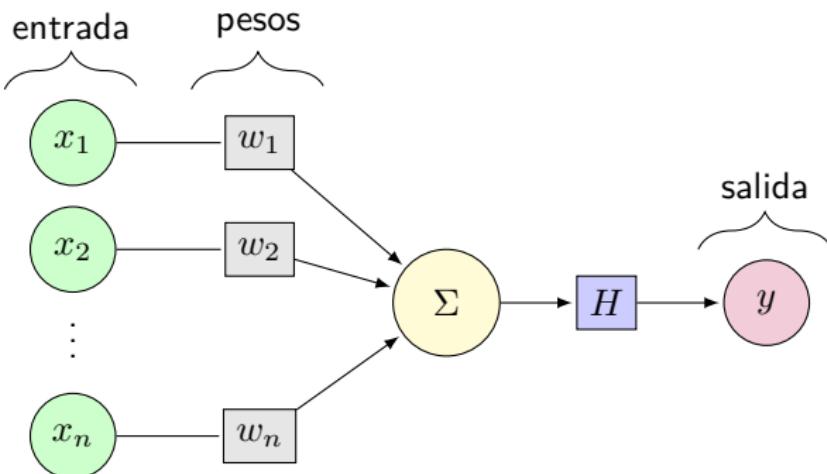
Escuela  
Nacional de  
Estudios  
Superiores



ferro@cimat.mx

Figure: Warren McCulloch &amp; Walter Pitts

Escuela  
Nacional de  
Estudios  
Superiores



$$\Sigma \rightarrow \sum_{i=1}^n w_i x_i$$



# Bias y función de activación

Intro al aprendizaje profundo

La operación matemática que realiza la neurona para la decisión de umbralización se puede escribir como:

$$f(\mathbf{x}) = \begin{cases} 0 & \text{si } \sum_i w_i x_i < \text{umbral o threshold} \\ 1 & \text{si } \sum_i w_i x_i \geq \text{umbral o threshold} \end{cases}$$

donde  $i \in \{1, 2, \dots, n\}$ , y así,  $\mathbf{x} = (x_1, x_2, \dots, x_n)$ .

# Bias y función de activación

Intro al aprendizaje profundo

De lo anterior, podemos despejar el umbral y escribirlo como  $b$ , obteniendo:

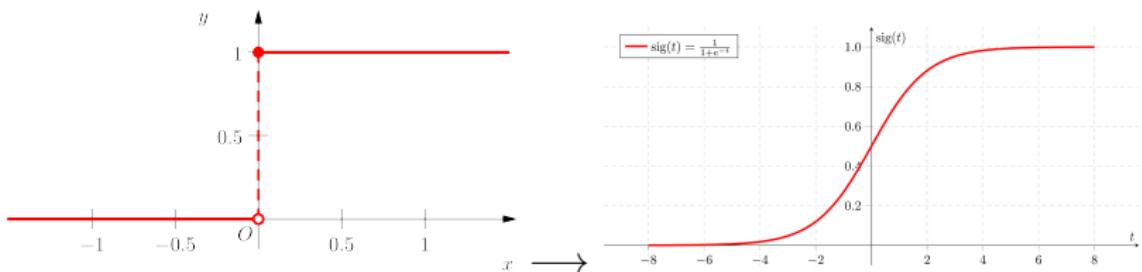
$$f(\mathbf{x}) = \begin{cases} 0 & \text{si } \sum_i w_i x_i + b < 0 \\ 1 & \text{si } \sum_i w_i x_i + b > 0 \end{cases}$$

donde  $\mathbf{x} = (x_1, x_2, \dots, x_n)$  y  $i \in \{1, 2, \dots, n\}$ .

A esto que escribimos como  $b$ , también se le conoce como **bias**, y una interpretación es describir *qué tan susceptible es la neurona a dispararse* (como se exploró en el ejemplo práctico de la identificación de la actividad de Julieta).

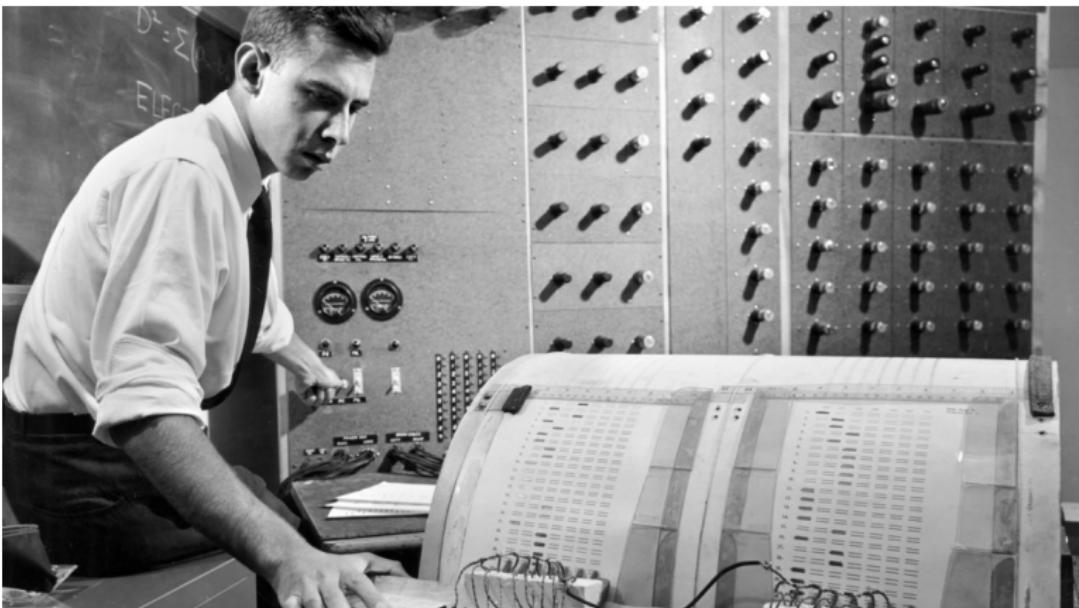
# Bias y función de activación

## Intro al aprendizaje profundo



# El Perceptrón

## Intro al aprendizaje profundo

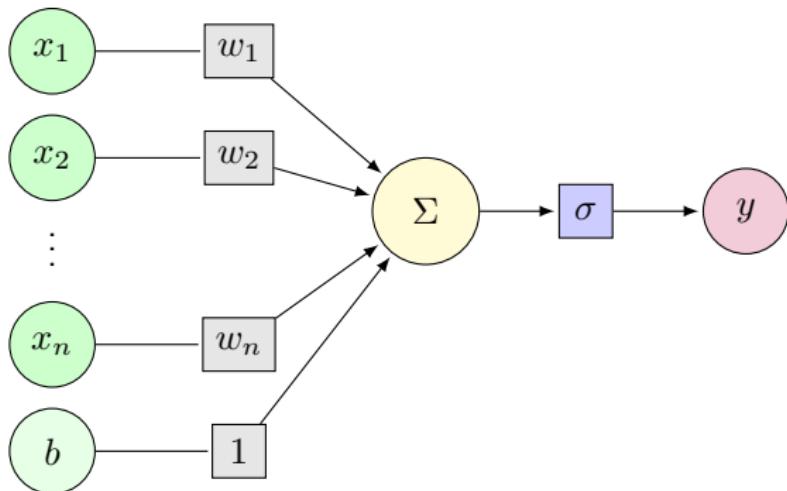


ferro@cmat.mx

Figure: Frank Rosenblatt



Escuela  
Nacional de  
Estudios  
Superiores  
Unidad León



¡Y se agrega un algoritmo formal de entrenamiento!  
(*Backpropagation*)

# Idea intuitiva de entrenamiento

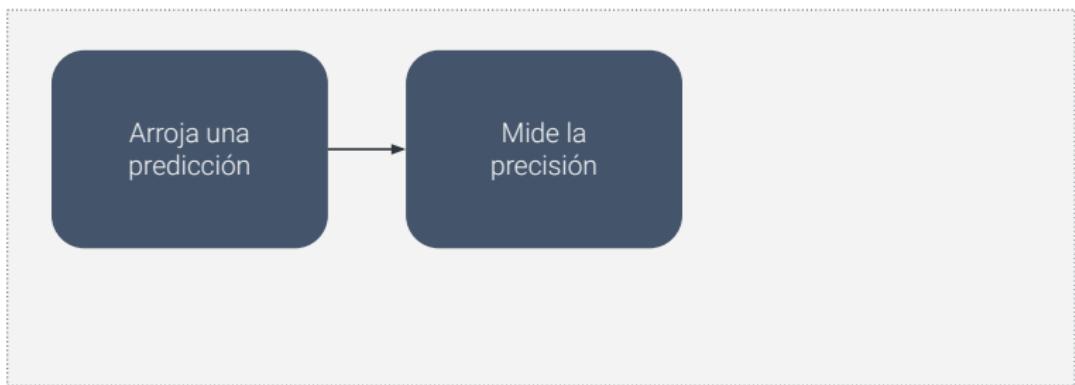
Intro al aprendizaje profundo

Arroja una  
predicción

ferro@cmat.mx

# Idea intuitiva de entrenamiento

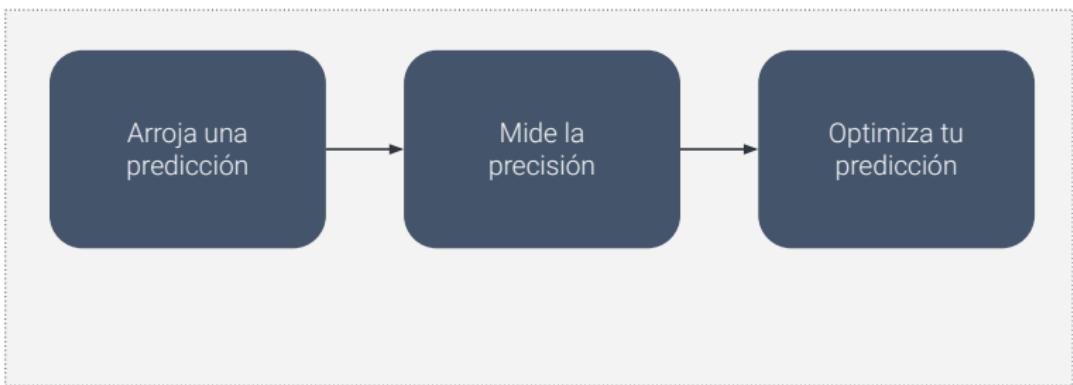
Intro al aprendizaje profundo



ferro@cimat.mx

# Idea intuitiva de entrenamiento

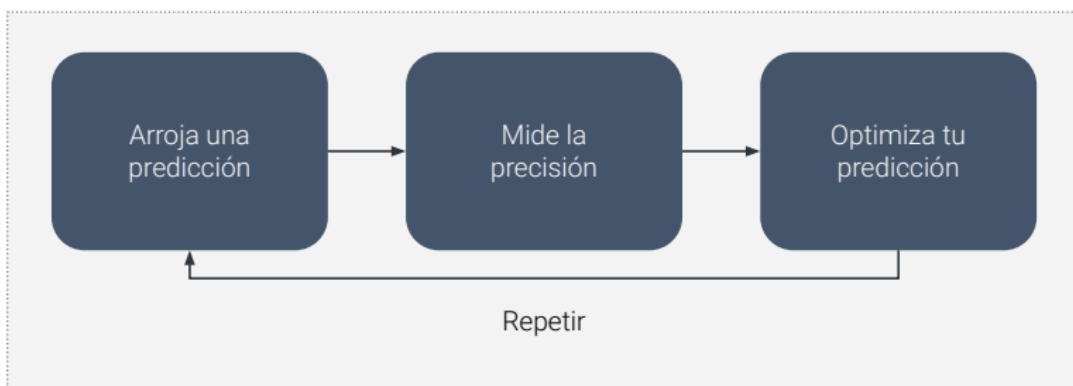
## Intro al aprendizaje profundo



ferro@cimat.mx

# Idea intuitiva de entrenamiento

## Intro al aprendizaje profundo



Dado el vector  $X$ , **¿qué vector  $(A, B, C)$  se le parece más?**

$$X = \begin{bmatrix} 0.5 \\ 0.3 \\ 0.7 \end{bmatrix}$$

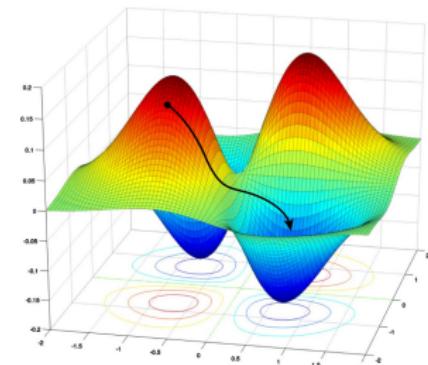
$$A = \begin{bmatrix} 0.3 \\ 0.3 \\ 0.3 \end{bmatrix}, B = \begin{bmatrix} 0.6 \\ 0.2 \\ 0.6 \end{bmatrix}, C = \begin{bmatrix} -0.5 \\ -0.3 \\ -0.7 \end{bmatrix}$$

Dado el vector  $X$ , **¿qué vector  $(A, B, C)$  se le parece más?**

$$X = \begin{bmatrix} 0.5 \\ 0.3 \\ 0.7 \end{bmatrix}$$

$$A = \begin{bmatrix} 0.3 \\ 0.3 \\ 0.3 \end{bmatrix}, B = \begin{bmatrix} 0.6 \\ 0.2 \\ 0.6 \end{bmatrix}, C = \begin{bmatrix} -0.5 \\ -0.3 \\ -0.7 \end{bmatrix}$$

- › **Error:** Es una función.
- › **Optimizar:** Maximizar o minimizar.
- › **Gradiente:** Derivada de una función vectorial, proporciona información sobre máximos o mínimos.
- › **Descenso de gradiente:** Algoritmo para, iterativamente, buscar optimizar una función.
- › **Limitantes:**
  - » Max's/min's locales.
  - » Tamaño de salto en gradiente



Hasta este punto, debemos notar que hay algunas observaciones importantes:

- › **TLUs:**
  - » No existe un algoritmo de aprendizaje formal → Búsqueda de pesos.
  - » Se limita a propagación hacia adelante (*forward pass/forward propagation*)
- › **Perceptrón:** Puede utilizar retropropagación, introducido en 1958.
- › **Retropropagación:** Algoritmo para realizar ajustes en los valores de los pesos.
- › **Limitantes:** Separabilidad lineal.
- › **¿Alguna otra observación?**



## Ejercicio: Manzanas vs. Naranjas



ferro@cimat.mx

- › Breve historia sobre el perceptrón
- › Post sobre el perceptrón de Rosenblatt
- › Post sobre la función de activación
- › Selección de threshold para clasificadores binarios
- › Post sobre redes neuronales por IBM

Recordemos cómo opera el producto matricial:

$$\begin{bmatrix} 2 & 5 & 2 \\ 1 & 0 & -2 \\ 3 & 1 & 1 \end{bmatrix} \begin{bmatrix} -2 & 1 & 0 \\ -2 & 2 & 1 \\ 0 & 0 & 3 \end{bmatrix} = \begin{bmatrix} \quad & \quad & \quad \\ \quad & \quad & \quad \\ \quad & \quad & \quad \end{bmatrix}$$

Recordemos cómo opera el producto matricial:

$$\begin{bmatrix} 2 & 5 & 2 \\ 1 & 0 & -2 \\ 3 & 1 & 1 \end{bmatrix} \begin{bmatrix} -2 & 1 & 0 \\ -2 & 2 & 1 \\ 0 & 0 & 3 \end{bmatrix} = \begin{bmatrix} 14 & & \\ & & \\ & & \end{bmatrix}$$

$$(2 \cdot -2) + (5 \cdot -2) + (2 \cdot 0) = -14$$

Recordemos cómo opera el producto matricial:

$$\begin{bmatrix} 2 & 5 & 2 \\ 1 & 0 & -2 \\ 3 & 1 & 1 \end{bmatrix} \begin{bmatrix} -2 & 1 & 0 \\ -2 & 2 & 1 \\ 0 & 0 & 3 \end{bmatrix} = \begin{bmatrix} 14 & 12 & \end{bmatrix}$$

$$(2 \cdot 1) + (5 \cdot 2) + (2 \cdot 0) = 12$$

Recordemos cómo opera el producto matricial:

$$\begin{bmatrix} 2 & 5 & 2 \\ 1 & 0 & -2 \\ 3 & 1 & 1 \end{bmatrix} \begin{bmatrix} -2 & 1 & 0 \\ -2 & 2 & 1 \\ 0 & 0 & 3 \end{bmatrix} = \begin{bmatrix} 14 & 12 & 11 \end{bmatrix}$$

$$(2 \cdot 0) + (5 \cdot 1) + (2 \cdot 3) = 11$$



Recordemos cómo opera el producto matricial:

$$\begin{bmatrix} 2 & 5 & 2 \\ 1 & 0 & -2 \\ 3 & 1 & 1 \end{bmatrix} \begin{bmatrix} -2 & 1 & 0 \\ -2 & 2 & 1 \\ 0 & 0 & 3 \end{bmatrix} = \begin{bmatrix} 14 & 12 & 11 \\ -2 & & \end{bmatrix}$$

$$(1 \cdot -2) + (0 \cdot -2) + (-2 \cdot 0) = -2$$

Recordemos cómo opera el producto matricial:

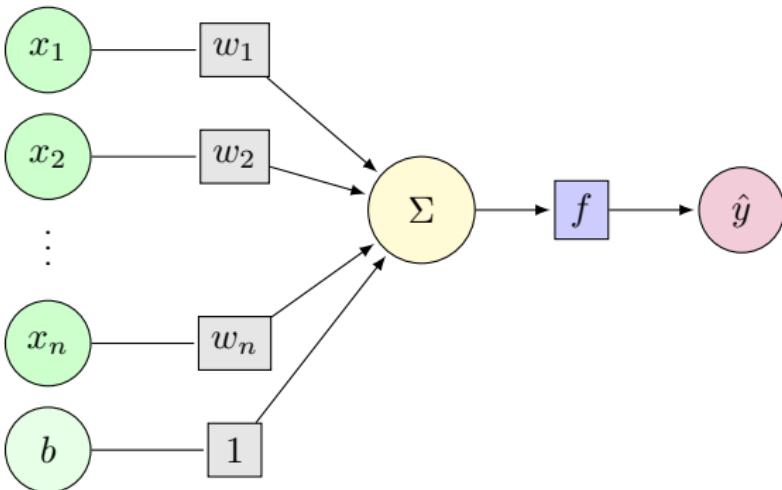
$$\begin{bmatrix} 2 & 5 & 2 \\ 1 & 0 & -2 \\ 3 & 1 & 1 \end{bmatrix} \begin{bmatrix} -2 & 1 & 0 \\ -2 & 2 & 1 \\ 0 & 0 & 3 \end{bmatrix} = \begin{bmatrix} 14 & 12 & 11 \\ -2 & & \\ & & \end{bmatrix}$$

...

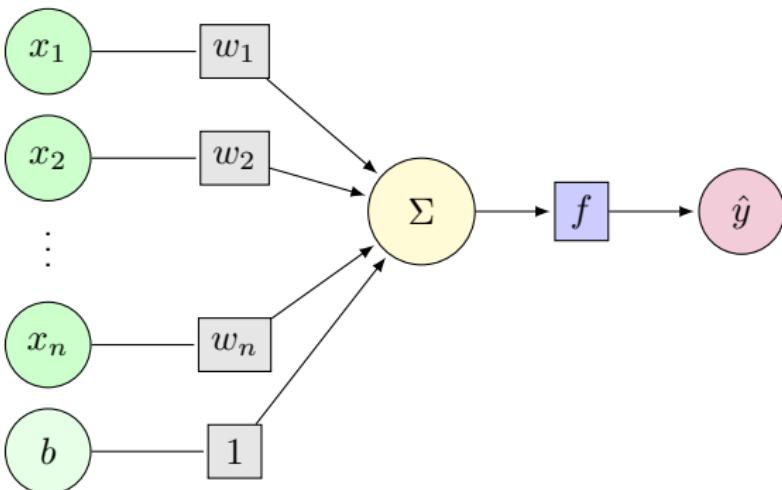


Recordemos cómo opera el producto matricial:

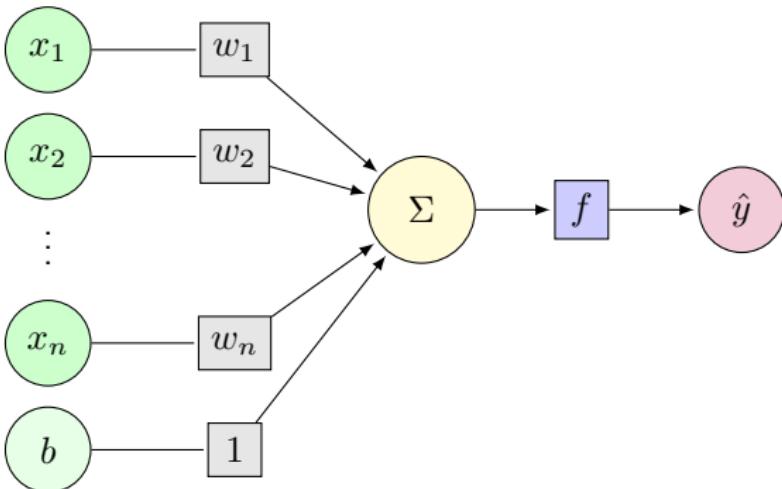
$$\begin{bmatrix} 2 & 5 & 2 \\ 1 & 0 & -2 \\ 3 & 1 & 1 \end{bmatrix} \begin{bmatrix} -2 & 1 & 0 \\ -2 & 2 & 1 \\ 0 & 0 & 3 \end{bmatrix} = \begin{bmatrix} 14 & 12 & 11 \\ -2 & 1 & -6 \\ -8 & 5 & 4 \end{bmatrix}$$



$$\hat{y} = f \left( \sum_i^n w_i x_i + b \right)$$



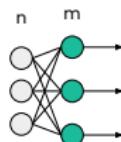
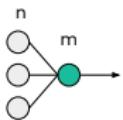
$$\sum_i^n w_i x_i = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$



$$\mathbf{w}^T \mathbf{x} = [w_1 w_2 \cdots w_n] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

# Producto matricial

## Intro al aprendizaje profundo

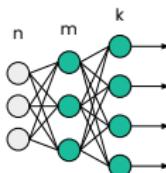


$$\mathbf{W} \cdot \mathbf{X} + \mathbf{b} = [w_1 \ w_2 \ \dots \ w_n] \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + b \longrightarrow \mathbf{W}_{Layer} \cdot \mathbf{X} + \mathbf{b} = \begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,n} \\ w_{2,1} & w_{2,2} & \dots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \dots & w_{m,n} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

$f(\cdot)$

# Producto matricial

## Intro al aprendizaje profundo



$$\mathbf{W}_{Layer_k} \cdot (\mathbf{W}_{Layer_n} \cdot \mathbf{X} + \mathbf{b}_m) + \mathbf{b}_k = \begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,m} \\ w_{2,1} & w_{2,2} & \dots & w_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,1} & w_{k,2} & \dots & w_{k,m} \end{bmatrix} \left( \begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,n} \\ w_{2,1} & w_{2,2} & \dots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \dots & w_{m,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \right) + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \end{bmatrix}$$

$f(\cdot)$

# Composición de funciones

## Intro al aprendizaje profundo

$$\mathbf{W}_{Layer_k} \cdot (\mathbf{W}_{Layer_m} \cdot \mathbf{X} + \mathbf{b}_m) + \mathbf{b}_k = \begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,m} \\ w_{2,1} & w_{2,2} & \dots & w_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,1} & w_{k,2} & \dots & w_{k,m} \end{bmatrix} \left( \begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,n} \\ w_{2,1} & w_{2,2} & \dots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \dots & w_{m,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \right) + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \end{bmatrix}$$

↓

$$f_k(\mathbf{W}_{Layer_k} \cdot f_m(\mathbf{W}_{Layer_m} \cdot \mathbf{X} + \mathbf{b}_m) + \mathbf{b}_k)$$

↓

$$f_n(\dots(f_2(\mathbf{W}_{Layer_k} \cdot f_1(\mathbf{W}_{Layer_1} \cdot \mathbf{X} + \mathbf{b}_1) + \mathbf{b}_2)) + \mathbf{b}_n)$$

# Composición de funciones

Intro al aprendizaje profundo

$$f_n(\dots(f_2(\mathbf{W}_{Layer_k} \cdot f_1(\mathbf{W}_{Layer_1} \cdot \mathbf{X} + \mathbf{b}_1) + \mathbf{b}_2)) + \mathbf{b}_n)$$



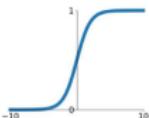
$$F'(x) = f'(g(x)) \cdot g'(x)$$

# Funciones de activación

Intro al aprendizaje profundo

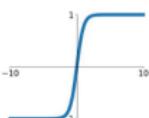
## Sigmoid

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



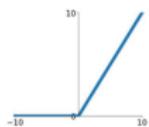
## tanh

$$\tanh(x)$$



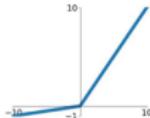
## ReLU

$$\max(0, x)$$



## Leaky ReLU

$$\max(0.1x, x)$$

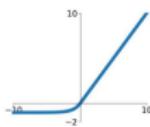


## Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

## ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



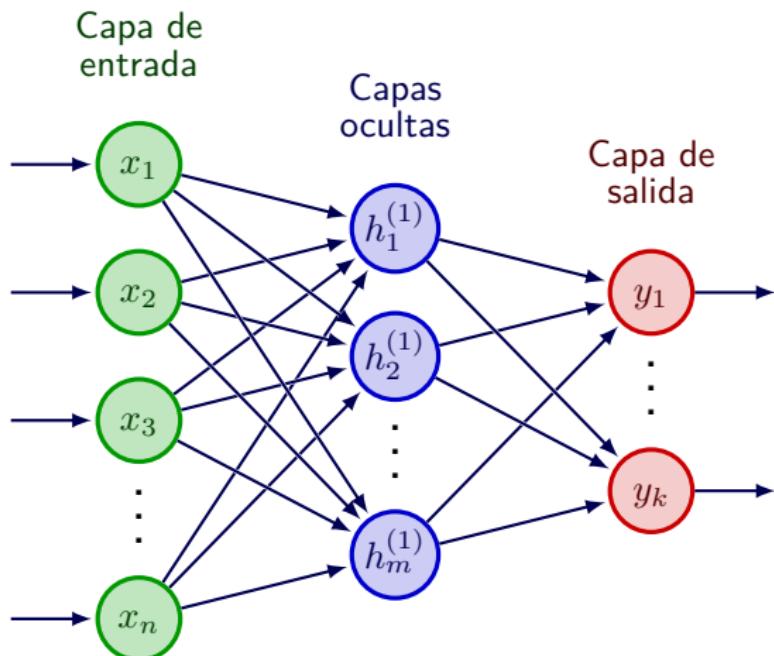
$$f(x) = \sigma(x) = \frac{1}{1+e^{-x}} \Rightarrow f'(x) = f(x)(1-f(x))$$

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \Rightarrow f'(x) = 1 - f(x)^2$$



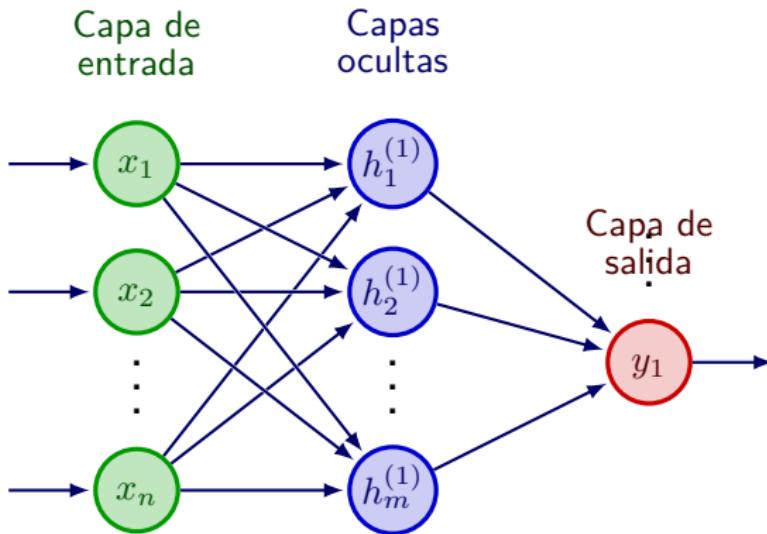
# El perceptrón multicapa

Intro al aprendizaje profundo



# El perceptrón multicapa

Intro al aprendizaje profundo



Para  $\mathbf{x}^{(i)} = (x_1^{(i)}, x_2^{(i)}, \dots, x_n^{(i)}), y^{(i)}$ , tendríamos que la salida es  
 $\hat{y}_1^{(i)} = f(x^{(i)})$ .

- › La **función de pérdida (loss)** de nuestra red neuronal *mide el costo* asociado a predicciones incorrectas.
- › Si observaciones (de entrada y salida)  $(x^{(i)}, y^{(i)})$  y consideramos a la salida como función de  $x^{(i)}$  y  $\mathbf{W}$ , entonces las salidas son  $\hat{y} = f(x^{(i)}; \mathbf{W})$  y la función de pérdida puede escribirse como:

$$\mathcal{L}(f(x^{(i)}; \mathbf{W}), y^{(i)})$$

Es decir, una función que mida la salida *real* con la *predicción*.  
Todo esto para una observación  $i$ .

- » Para todas las observaciones:

$$\mathbf{J}(\mathbf{W}) = \frac{1}{n} \sum_{i=1}^n \mathcal{L}(f(x^{(i)}; \mathbf{W}), y^{(i)})$$

A esta función se le conoce como función de costo o función objetivo (lo que queremos minimizar).

- › **Binary Cross Entropy Loss:** Se puede utilizar con modelos que devuelven como salida una probabilidad entre 0 y 1.

$$J(\mathbf{W}) = \frac{1}{n} \sum_{i=1}^n y^{(i)} \log(f(x^{(i)}; \mathbf{W})) + (1 - y^{(i)}) \log(1 - f(x^{(i)}; \mathbf{W}))$$

- › **Mean Squared Error (MSE) Loss:** Se puede utilizar con modelos de regresión que generan números reales continuos.

$$J(\mathbf{W}) = \frac{1}{n} \sum_{i=1}^n \left( y^{(i)} - f(x^{(i)}; \mathbf{W}) \right)^2$$

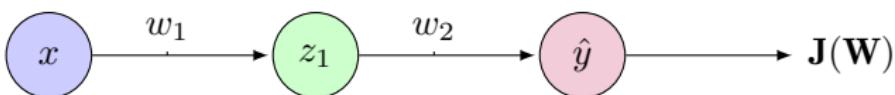
- › Queremos encontrar los pesos ideales de la red neuronal, los cuales minimizan  $\mathbf{J}(\mathbf{W})$ , es decir:

$$\mathbf{W}^* = \operatorname{argmin}_{\mathbf{W}} \frac{1}{n} \sum_{i=1}^n \mathcal{L}(f(x^{(i)}; \mathbf{W}), y^{(i)})$$

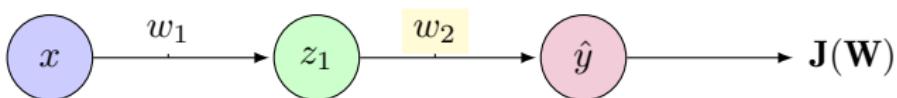
$$\mathbf{W}^* = \operatorname{argmin}_{\mathbf{W}} \mathbf{J}(\mathbf{W})$$

### Algoritmo: Descenso de gradiente

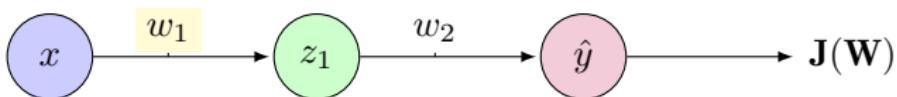
- 1 Inicializar los pesos aleatoriamente  $\sim \mathcal{N}(0, \sigma^2)$
- 2 Repetir hasta converger:
- 3 Calcular el gradiente  $\frac{\partial \mathbf{J}(\mathbf{W})}{\partial \mathbf{W}}$
- 4 Actualizar los pesos  $\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial \mathbf{J}(\mathbf{W})}{\partial \mathbf{W}}$
- 5 Devolver pesos *óptimos*



¿Cómo se calculan los gradientes?  
Con la regla de la cadena.



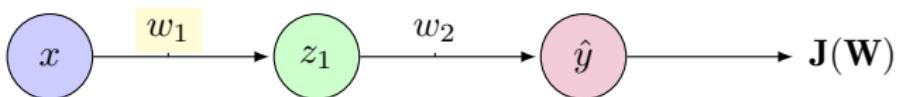
$$\frac{\partial \mathbf{J}(\mathbf{W})}{\partial w_2} = \frac{\partial \mathbf{J}(\mathbf{W})}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_2}$$



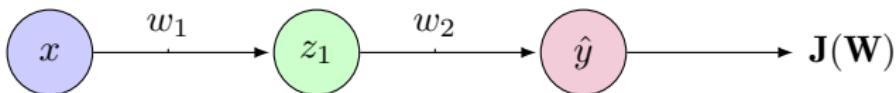
$$\frac{\partial \mathbf{J}(\mathbf{W})}{\partial w_1} = \frac{\partial \mathbf{J}(\mathbf{W})}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_1}$$

# Retropropagación

## Intro al aprendizaje profundo



$$\frac{\partial \mathbf{J}(\mathbf{W})}{\partial w_1} = \frac{\partial \mathbf{J}(\mathbf{W})}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_1}$$



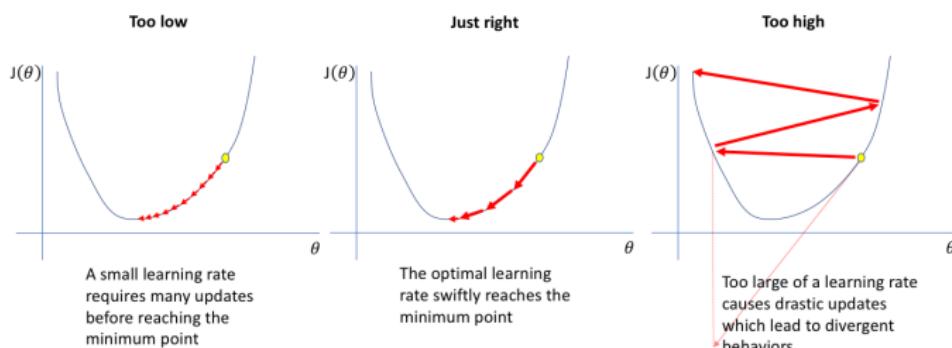
¿Cómo se calculan los gradientes?

Con la regla de la cadena.

Esto se repite para **cada peso** en la red neuronal, usando los gradientes de las capas posteriores.

La actualización de pesos está dada por:

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$$



## Algoritmo: Descenso de gradiente

- 1 Inicializar los pesos aleatoriamente  $\sim \mathcal{N}(0, \sigma^2)$
  - 2 Repetir hasta converger:
  - 3 Calcular el gradiente  $\frac{\partial \mathbf{J}(\mathbf{W})}{\partial \mathbf{W}}^*$
  - 4 Actualizar los pesos  $\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial \mathbf{J}(\mathbf{W})}{\partial \mathbf{W}}$
  - 5 Devolver pesos *óptimos*
- › \*Esto es muy pesado de calcular (computacionalmente).

# Descenso de gradiente estocástico

Intro al aprendizaje profundo

## Algoritmo: Descenso de gradiente estocástico

- 1 Inicializar los pesos aleatoriamente  $\sim \mathcal{N}(0, \sigma^2)$
  - 2 Repetir hasta converger:
  - 3 Seleccionar observación  $i$
  - 4 Calcular el gradiente 
$$\frac{\partial \mathbf{J}_i(\mathbf{W})}{\partial \mathbf{W}}^*$$
  - 5 Actualizar los pesos  $\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial \mathbf{J}(\mathbf{W})}{\partial \mathbf{W}}$
  - 6 Devolver pesos *óptimos*
- \*Esto es muy sencillo de calcular (computacionalmente), pero es estocástico.



## Algoritmo: Descenso de gradiente estocástico - *Mini batches*

1 Inicializar los pesos aleatoriamente  $\sim \mathcal{N}(0, \sigma^2)$

2 Repetir hasta converger:

3 Seleccionar un batch  $B$  de observaciones

4 Calcular el gradiente 
$$\frac{\partial \mathbf{J}(\mathbf{W})}{\partial \mathbf{W}} = \frac{1}{B} \sum_{k=1}^B \frac{\partial \mathbf{J}_k(\mathbf{W})}{\partial \mathbf{W}}^*$$

5 Actualizar los pesos  $\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial \mathbf{J}(\mathbf{W})}{\partial \mathbf{W}}$

6 Devolver pesos óptimos

› \*Esto es rápido de calcular (computacionalmente), y da una mejor estimación del gradiente.

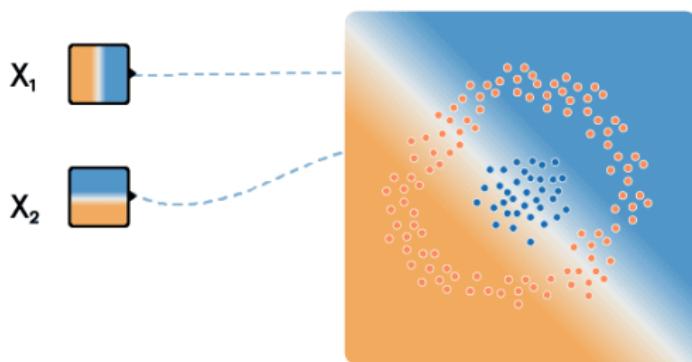
- › Los **frameworks** para aprendizaje profundo (como TensorFlow, PyTorch, etc.) ya hacen la diferenciación y optimización por nosotros, es decir, ya calculan el gradiente y actualizan los pesos.
- › Nosotros exploraremos el uso de **TensorFlow** a través de su API de alto nivel, **Keras**, para las redes neuronales que estaremos construyendo.
- › Comenzaremos retomando algunos de los problemas planteados en la sesión anterior.



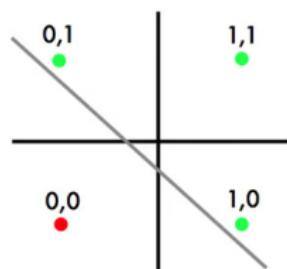
# TensorFlow

TensorFlow es un framework open-source para Machine Learning desarrollado por Google. Utilizado para construir y entrenar redes neuronales artificiales.

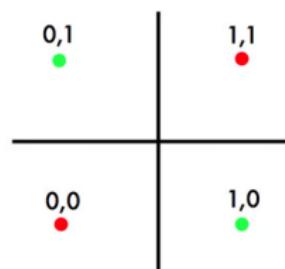
## Ejercicio: Exploración del TensorFlow Playground



## Ejercicio: Problema de separabilidad lineal



OR



XOR

## Ejercicio: Exploración con TensorFlow

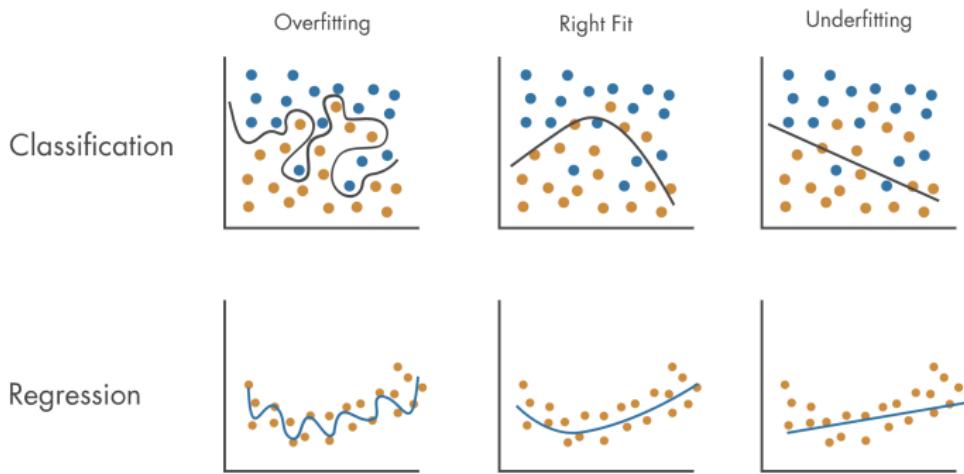


# TensorFlow

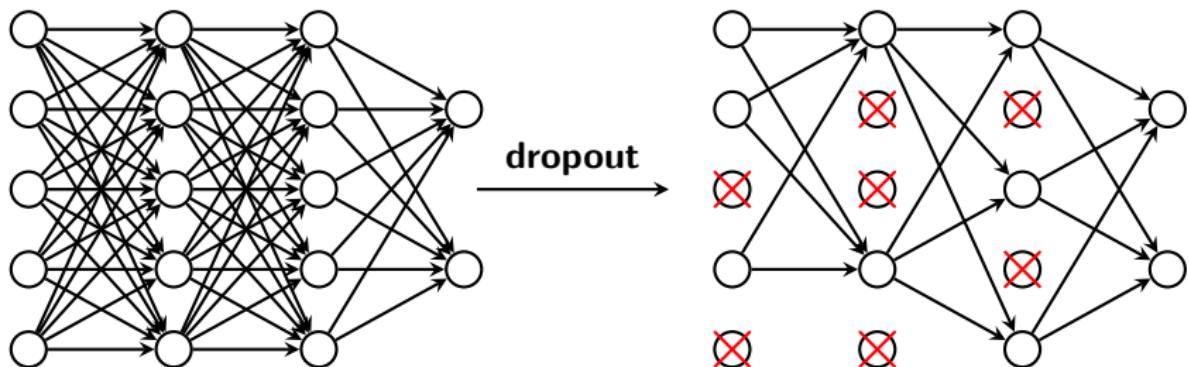
- › Setting the learning rate of your neural network
- › Retropropagación paso a paso
- › TensorFlow Tutorials
- › Libro Neural Networks and Deep Learning

# El problema del overfitting

## Intro al aprendizaje profundo



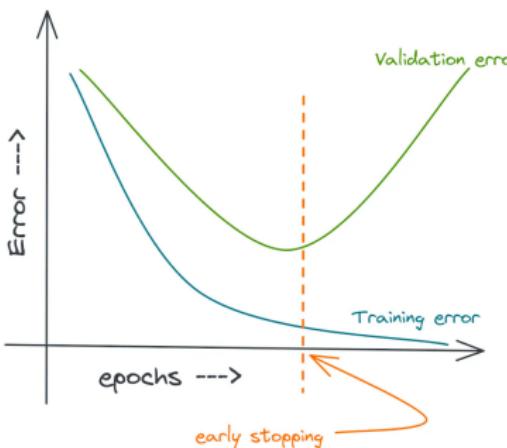
- › La **regularización** consiste en alguna técnica que sirve para evitar que un modelo se sobreajuste.
- › Es necesaria porque ayuda a mejorar la generalización de nuestro modelo con datos no vistos.
- › Los métodos de regularización que exploraremos serán:
  - » *Dropout*
  - » *Early stopping*



- › Durante el entrenamiento, establecemos aleatoriamente algunas activaciones en 0
  - » Típicamente hacemos "drop" del 50% de activaciones en una capa.
  - » Esto forza a la red a no depender de ningún nodo/neurona.
- › Podemos realizar el dropout en TensorFlow utilizando la capa `tf.keras.layers.Dropout(0.5)`, donde el 0.5 puede variar de acuerdo a lo especificado.

# Early stopping

## Intro al aprendizaje profundo



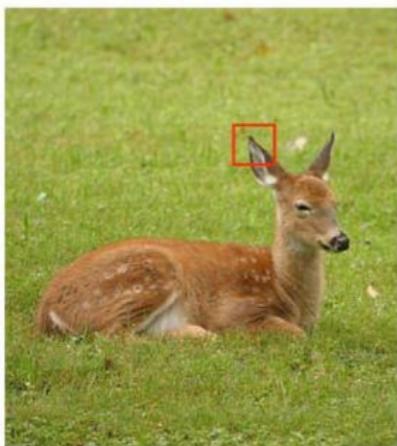
- El *Early Stopping* puede ser realizado en TensorFlow de manera sencilla creando un callback (función que se llama en cada iteración durante el entrenamiento de la red neuronal):

```
model = tf.keras.models.Sequential(...)  
callback = tf.keras.callbacks.EarlyStopping(monitor='loss',  
patience=3)  
history = model.fit(..., callbacks=[callback])
```

- › Artículo "Dropout: A Simple Way to Prevent Neural Networks from Overfitting"
- › Regularización en Redes Neuronales
- › Vanishing and Exploding Gradients in Neural Network Models:  
Debugging, Monitoring, and Fixing

# ¿Qué es una imagen?

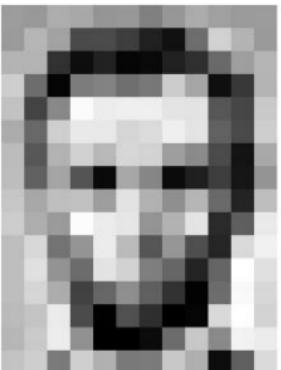
Visión computacional profunda



ferro@icmat.mx

# ¿Qué es una imagen?

Visión computacional profunda

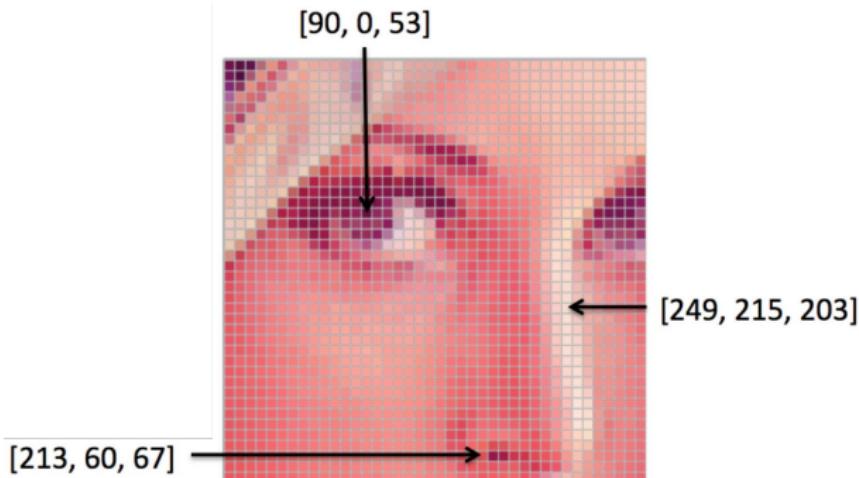


157	153	174	168	150	152	129	161	172	151	155	156
155	182	163	74	75	62	39	17	110	230	180	154
180	180	60	34	54	6	10	39	48	106	169	191
206	159	5	124	131	111	120	204	165	15	56	180
194	68	137	251	237	239	239	226	227	87	71	201
172	106	207	233	233	214	220	239	228	38	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	64	16	168	134	11	31	52	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	85	150	79	38	218	241
190	224	147	108	227	210	177	102	36	101	265	224
190	214	172	66	103	143	95	50	2	109	249	213
187	196	239	75	1	81	47	0	6	217	255	211
183	202	237	148	6	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	178	13	96	218

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	60	34	54	6	10	39	48	106	169	181
206	159	5	124	131	111	120	204	165	15	56	180
194	68	137	251	237	239	239	226	227	87	71	201
172	106	207	233	233	214	220	239	228	38	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	64	16	168	134	11	31	52	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	85	150	79	38	218	241
190	224	147	108	227	210	177	102	36	101	265	224
190	214	172	66	103	143	95	50	2	109	249	213
187	196	239	75	1	81	47	0	6	217	255	211
183	202	237	148	6	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	178	13	96	218

# ¿Qué es una imagen?

Visión computacional profunda



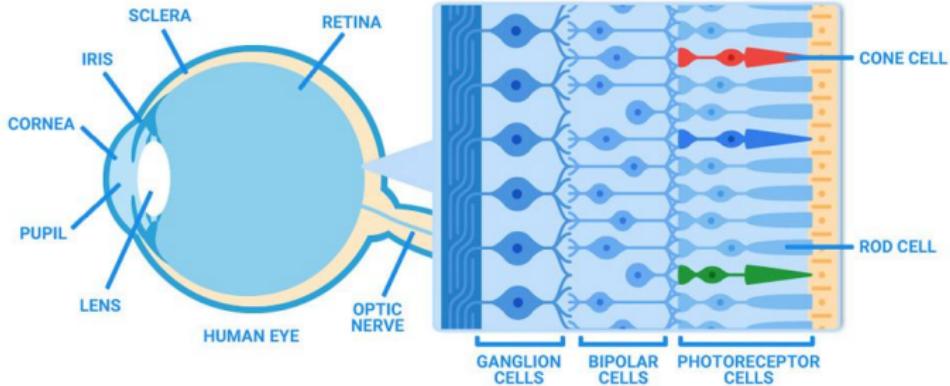
# ¿Qué es una imagen?

Visión computacional profunda

- › Una imagen es un arreglo de pixeles, la cual puede tener 1 o más canales de color. Usualmente:
  - » 1 canal de color → Escala de grises
  - » 3 canales de color → Escala RGB
  - » 4 canales de color → Escala RGBA
- › Un pixel puede ser visto como un objeto 5-dimensional  $(x, y, r, g, b)$ .

# La biología humana

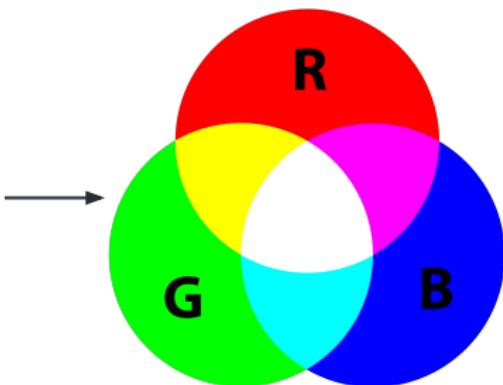
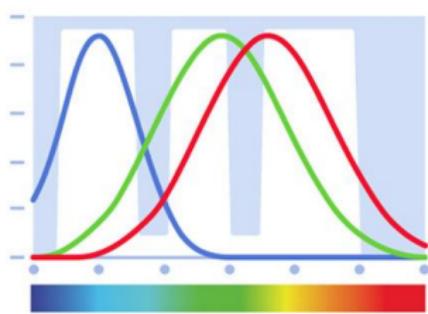
## Visión computacional profunda



ferro@cimat.mx

# La biología humana

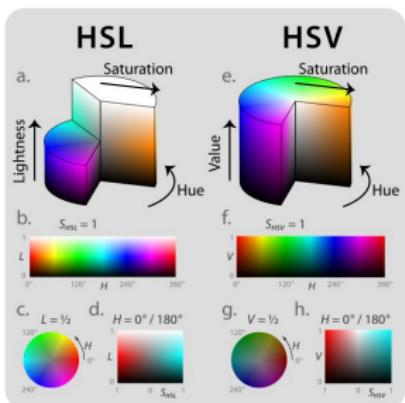
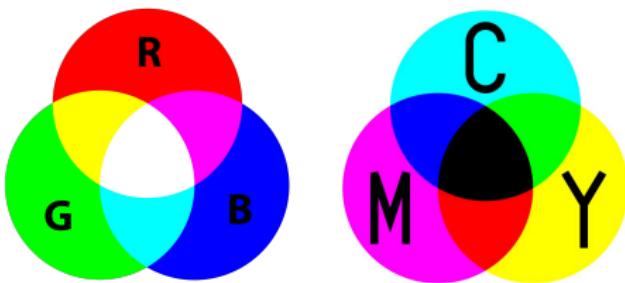
## Visión computacional profunda



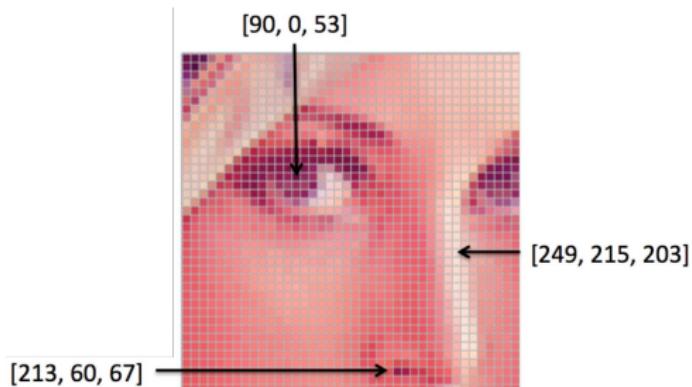
ferro@cimat.mx

# Espacios de color

Visión computacional profunda

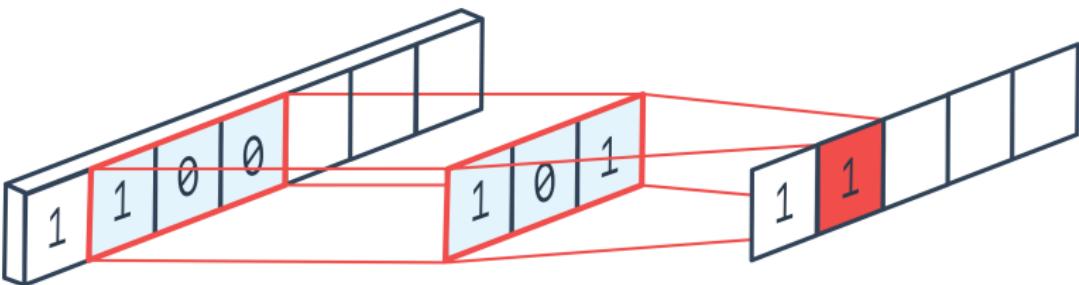


## Ejercicio: Introducción a imágenes



# ¿Qué es una convolución?

Visión computacional profunda



# ¿Qué es una convolución?

Visión computacional profunda

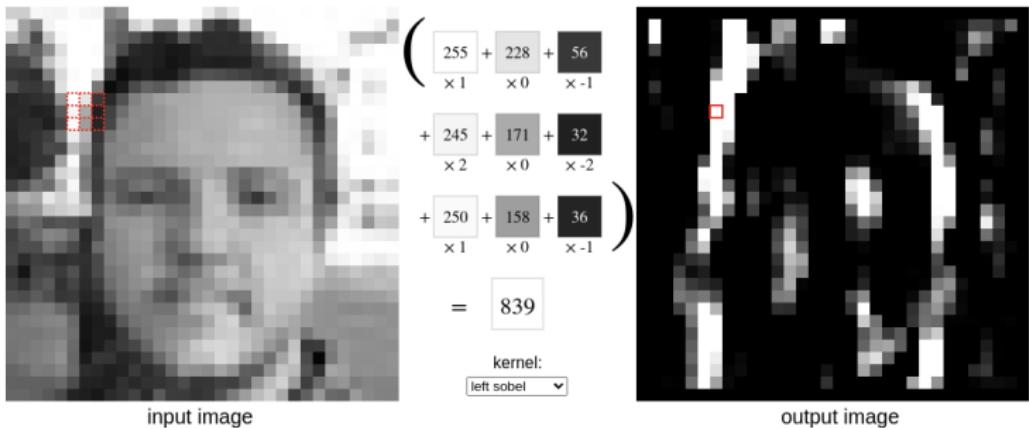
$$\begin{matrix} \begin{matrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{matrix} & * & \begin{matrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{matrix} & = & \begin{matrix} 1 & 4 & 3 & 4 & 1 \\ 1 & 2 & 4 & 3 & 3 \\ 1 & 2 & 3 & 4 & 1 \\ 1 & 3 & 3 & 1 & 1 \\ 3 & 3 & 1 & 1 & 0 \end{matrix} \end{matrix}$$

**I**                   **K**                    **$I * K$**

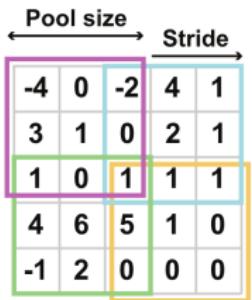
The diagram illustrates a convolution operation. Matrix I (Input) is a 7x7 grid with values 0, 1, and 2. Matrix K (Kernel) is a 3x3 grid with values 1, 0, 1. The result of the convolution is matrix  $I * K$ , which is a 5x5 grid. Dashed arrows indicate the receptive field of each output unit in  $I * K$ . The calculation shows how the kernel K slides across the input I to produce the output  $I * K$ .

# ¿Qué es una convolución?

Visión computacional profunda



ferro@cimat.mx



Features

Max Pooling



Min Pooling

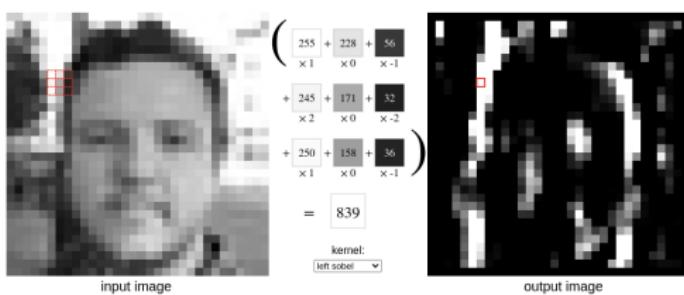


Average  
Pooling



Output

### Ejercicio: Convoluciones & Pooling



# Redes neuronales convolucionales

Visión computacional profunda



ferro@cimat.mx

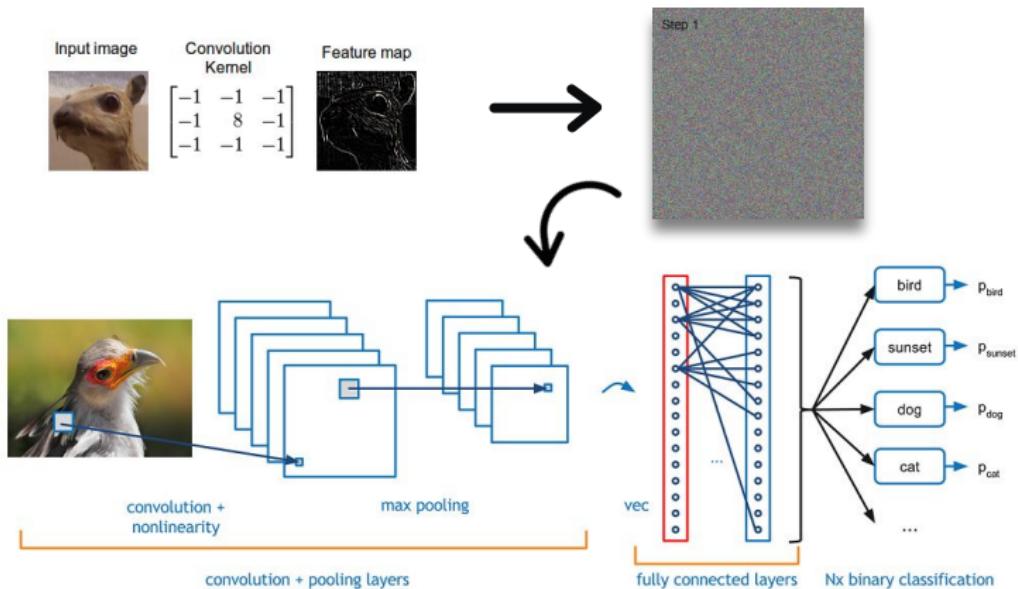
Figure: Yann LeCun



Escuela  
Nacional de  
Estudios  
Superiores

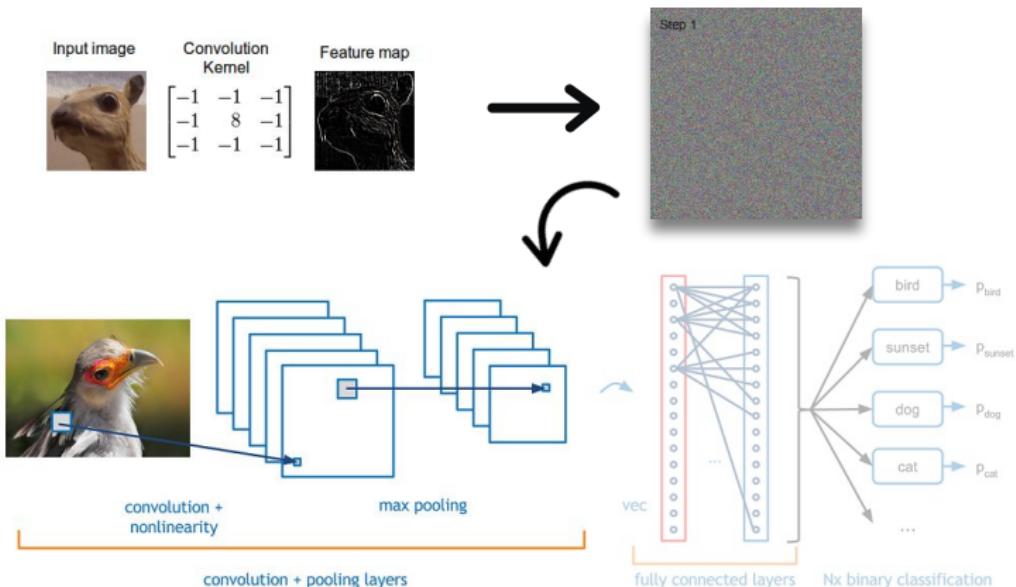
# Redes neuronales convolucionales

Visión computacional profunda



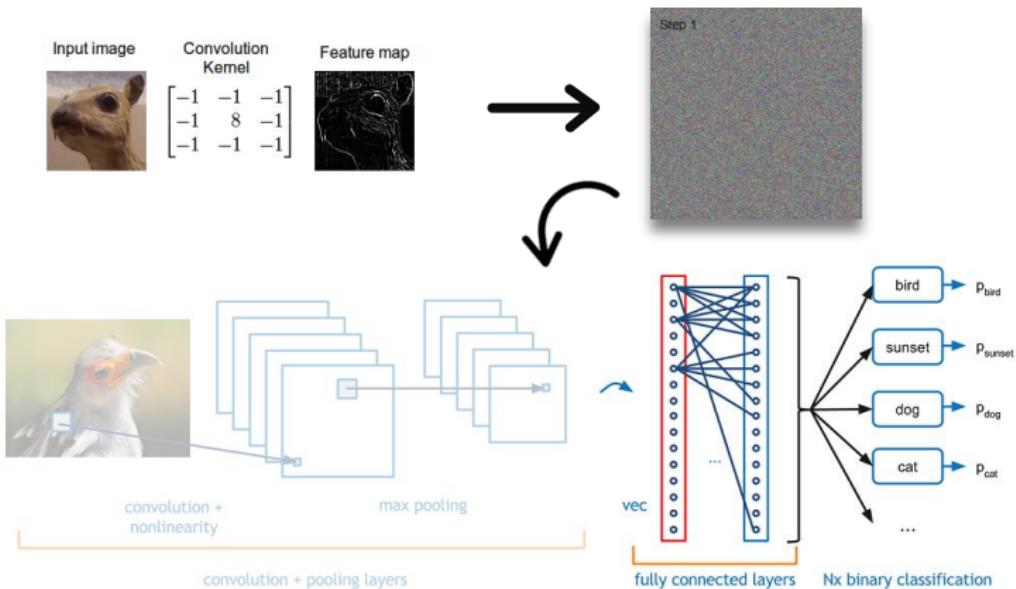
# Redes neuronales convolucionales

Visión computacional profunda



# Redes neuronales convolucionales

Visión computacional profunda



## Visión computacional profunda

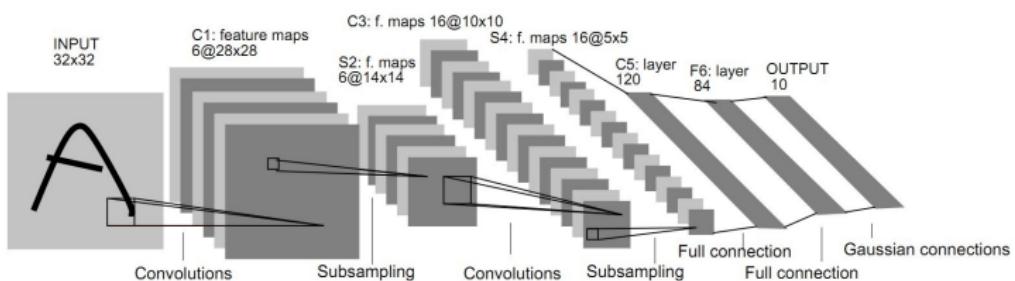
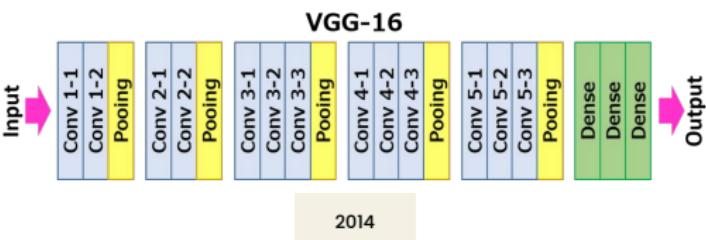
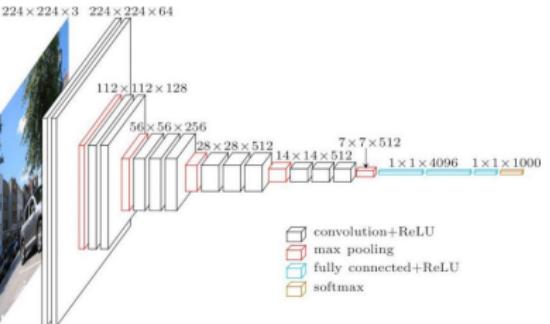


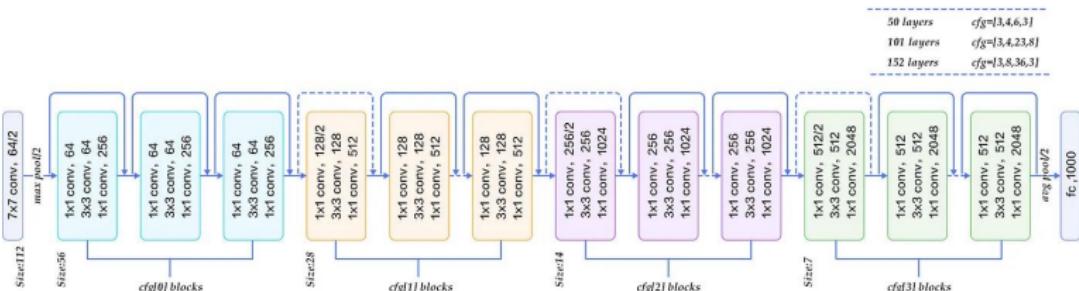
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

1998



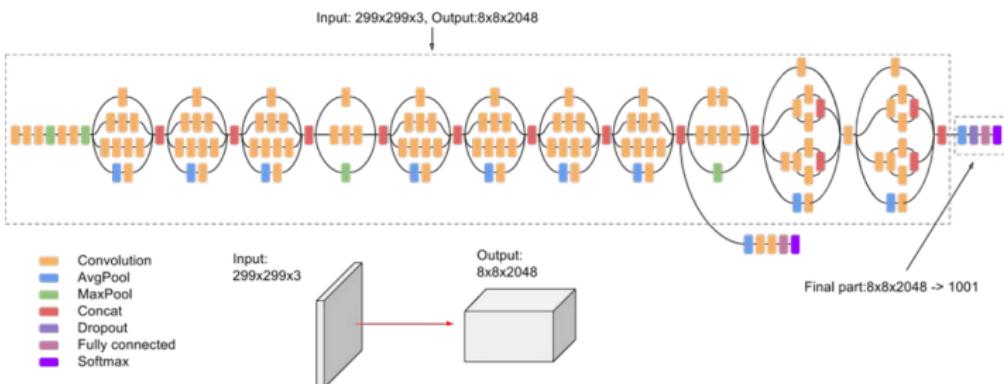


## Visión computacional profunda



2015

## **Visión computacional profunda**



2015

# Ejercicio: Redes neuronales convolucionales

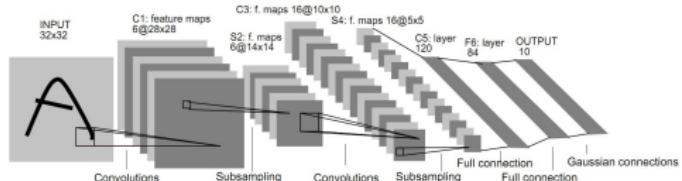


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

1998

- › Tutorial 1: [Image Filtering](#)
- › [Image Kernels Explained Visually](#) by Victor Powell
- › [Parameterized Pooling Layers](#) by Hao Hao Tan
- › [TensorFlow Tutorials: Vision](#)

# Reconstrucción de imágenes

## Visión computacional profunda

- Proceso de generar una imagen de salida a partir de una de entrada, generalmente con el objetivo de restaurar o mejorar la calidad de la imagen original.
- En el contexto de los autoencoders y la tarea de denoising, la reconstrucción implica generar una versión limpia y libre de ruido de una imagen ruidosa o de baja calidad.

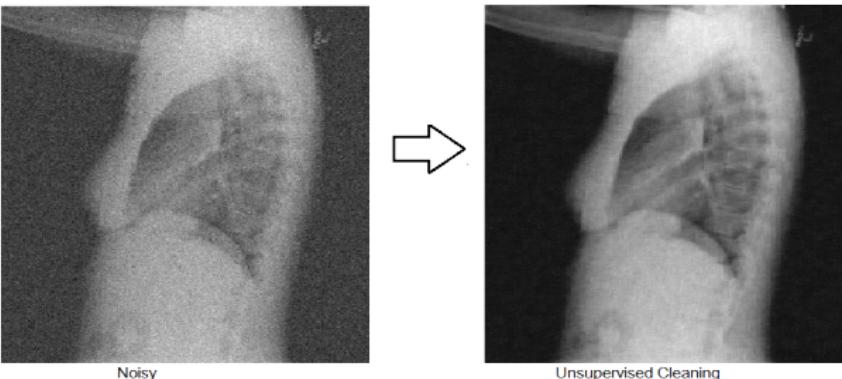
# Reconstrucción de imágenes

Visión computacional profunda

- 1 Restauración de la calidad visual.
- 2 Preservación de la información relevante.
- 3 Mejora de aplicaciones y análisis (eliminación de ruido).
- 4 Preservación de características y texturas.
- 5 Calidad y experiencia visual.

## Visión computacional profunda

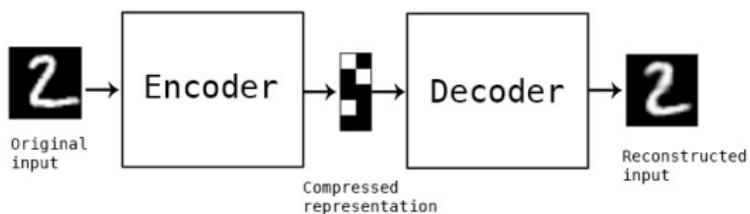
Indiana University X-Ray Dataset



"Autoencoder For Denoising Images." Michel Kana.

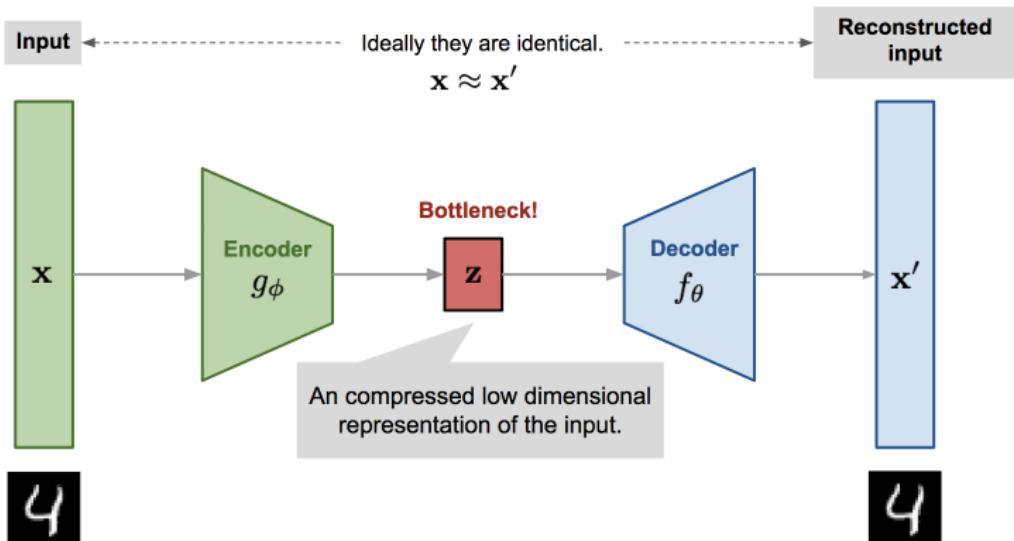
<https://towardsdatascience.com/autoencoder-for-denoising-images-7d63a0831bfd>

ANNs utilizadas para aprender representaciones eficientes de los datos de entrada sin necesidad de etiquetas o supervisión externa.



# Estructura de un autoencoder

Visión computacional profunda



# Funcionamiento de autoencoders

Visión computacional profunda

- › **Encoder:** Toma los datos de entrada y los transforma en una representación de menor dimensionalidad, también conocida como representación latente.
- › **El objetivo del codificador es...**  
Comprimir la información esencial de los datos en esta representación latente.
- › Usualmente, el codificador está compuesto por capas de neuronas que reducen gradualmente la dimensionalidad de los datos a medida que se propagan a través de la red. Este proceso de reducción dimensional es crucial para extraer las características más importantes de los datos y deshacerse de la información redundante o ruidosa.

# Funcionamiento de autoencoders

Visión computacional profunda

- **Decoder:** Toma la representación latente generada y la reconstruye en una salida que se asemeja lo más posible a la entrada original.
- **El objetivo del decodificador es...**  
Descomprimir la representación latente y generar una reconstrucción fiel de los datos de entrada.
- Al igual que el encoder, el decodificador está compuesto por capas de neuronas, pero en este caso, las capas aumentan gradualmente la dimensionalidad de la representación latente hasta que se obtiene una salida de la misma dimensión que los datos de entrada originales.

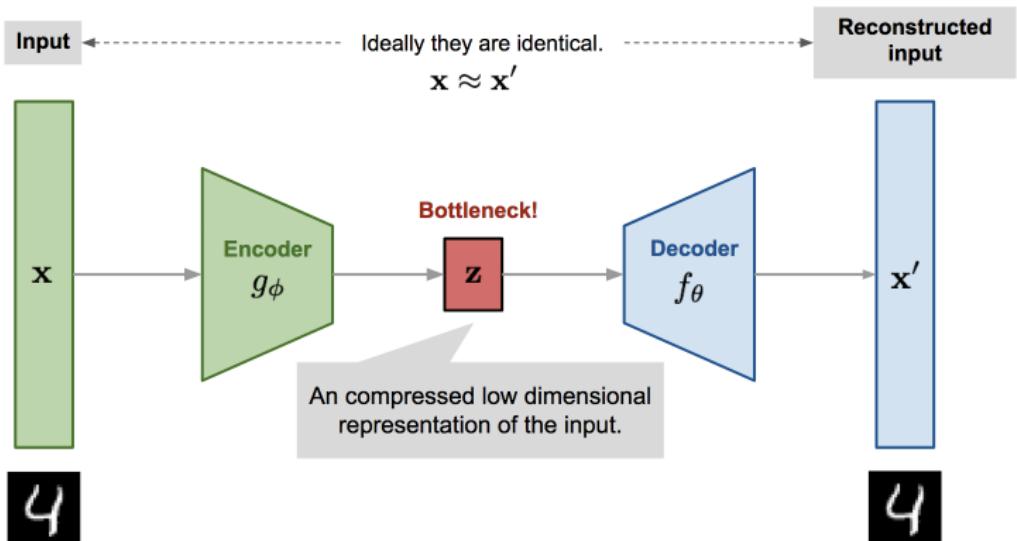
# Funcionamiento de autoencoders

Visión computacional profunda

- › La idea central es que, al restringir la capacidad de reconstrucción de la red, se obliga al codificador a aprender representaciones más compactas y significativas de los datos.
- › En otras palabras, se busca que el codificador capture las características más importantes y relevantes de los datos en la representación latente, mientras que el decodificador se encarga de reconstruir los datos de entrada a partir de esa representación latente.

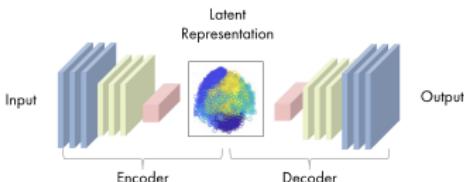
# Estructura de un autoencoder

Visión computacional profunda



# Aplicaciones de los autoencoders

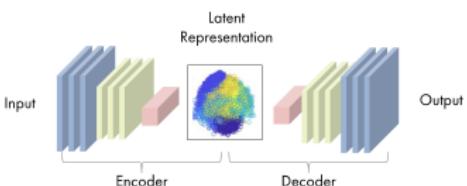
Visión computacional profunda



- › **Denoising de imágenes:** Se utilizan para eliminar el ruido de las imágenes y reconstruir versiones más limpias.
- › **Reducción de dimensionalidad:** Los autoencoders se utilizan para reducir la dimensionalidad de los datos, lo que facilita la visualización y comprensión de datos complejos. Esto es útil en análisis exploratorio de datos, visualización de datos de gran dimensión y clustering.

# Aplicaciones de los autoencoders

Visión computacional profunda



- **Generación de contenido:** Pueden generar contenido nuevo y original aprendiendo las características latentes de un conjunto de datos. Esto se utiliza en aplicaciones como la generación de imágenes sintéticas, la creación de música o la generación de texto.
- **Detección de anomalías:** Se utilizan para detectar patrones anormales o inusuales en los datos. Esto se aplica en áreas como la detección de fraudes en transacciones financieras, la detección de intrusiones en sistemas de seguridad o la detección de anomalías en imágenes médicas.

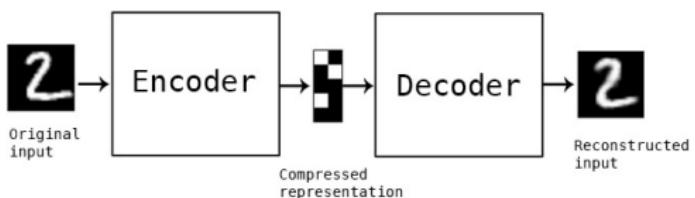
# Entrenamiento de autoencoders

Visión computacional profunda

- › **Aprendizaje:** Durante el entrenamiento de los autoencoders, se utilizan pares de datos de entrada y salida correspondientes.
- › La red se entrena para **minimizar** la diferencia entre la entrada original y la salida reconstruida.



### Ejercicio: Autoencoder básico

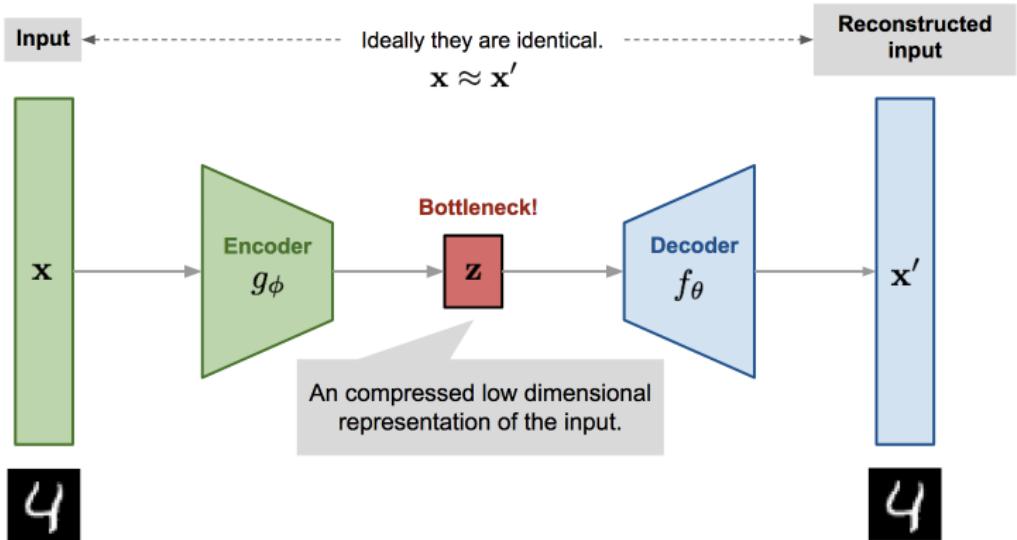


- › **Incapacidad para capturar información espacial:** Los autoencoders básicos pueden tener dificultades para capturar la estructura espacial de las imágenes, ya que no tienen en cuenta la información de vecindad de los píxeles.
- › **Sensibilidad a las transformaciones:** Pueden ser sensibles a las transformaciones geométricas, como la rotación o el desplazamiento de la imagen, lo que puede afectar su capacidad de reconstrucción.

- › **Autoencoders convolucionales como solución:** Son una variante de los autoencoders que incorporan capas convolucionales para abordar las limitaciones mencionadas.
- › **Ventajas de los autoencoders convolucionales:** Mejoran con la capacidad para capturar algunas características espaciales, preservar la estructura y ser más robustos frente a transformaciones geométricas

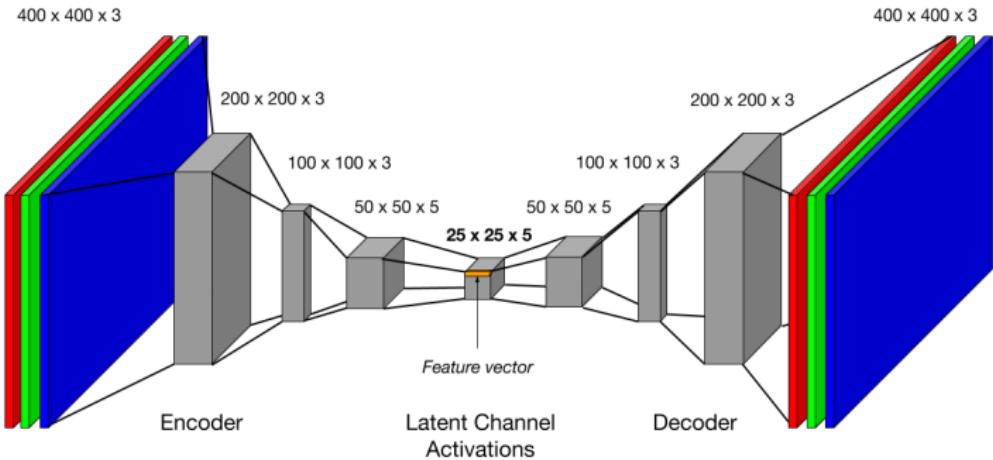
# Estructura de un autoencoder

Visión computacional profunda



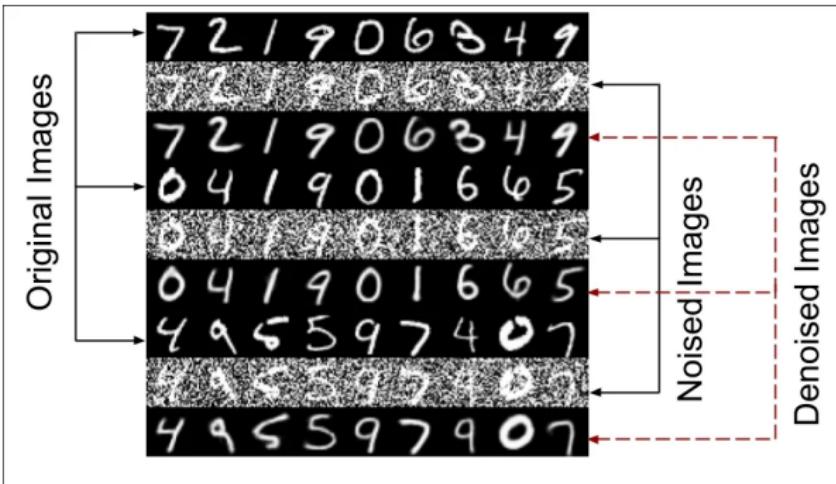
# Estructura de un autoencoder

## Visión computacional profunda



# Denoising autoencoder (DAE/DCAE)

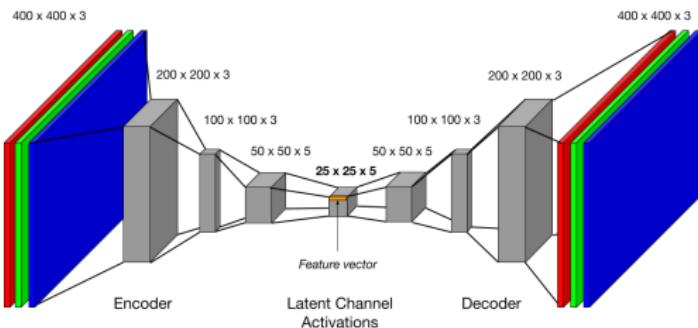
Visión computacional profunda



"Denoising autoencoder (DAE)."

<https://subscription.packtpub.com/book/data/9781788629416/3/ch03lvlsec19/denoising-autoencoder-dae>

### Ejercicio: Autoencoder convolucional

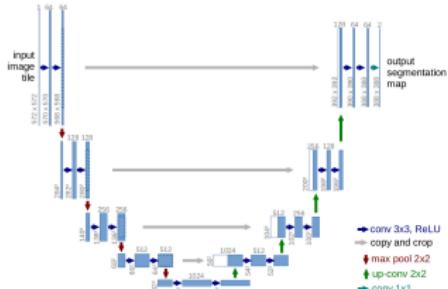


# Trabajos relacionados y avances recientes

## Visión computacional profunda

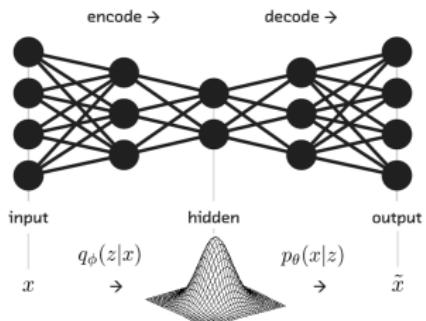
Han habido varios trabajos de investigación y avances recientes que han contribuido al desarrollo de nuevas arquitecturas, técnicas de entrenamiento mejoradas y aplicaciones emergentes.

- **UNet:** Es ampliamente utilizada en el campo de la segmentación de imágenes, pero también se ha aplicado con éxito en tareas de denoising.



### ➤ **Variational Autoencoders (VAEs):**

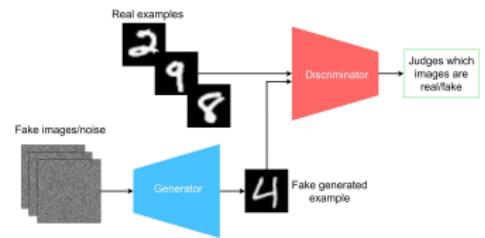
Los VAEs son una variante de los autoencoders que se utilizan para el aprendizaje de distribuciones latentes. Han demostrado ser efectivos en el denoising de imágenes al aprender representaciones latentes que siguen una distribución probabilística, lo que permite una generación más controlada y realista de imágenes limpias.



# Trabajos relacionados y avances recientes

## Visión computacional profunda

- **Generative Adversarial Networks (GANs):** Estos modelos aprovechan la capacidad de los GANs para generar imágenes realistas y para aprender representaciones latentes eficientes. Los GANs han demostrado ser efectivos en el denoising y la generación de imágenes de alta calidad, entre otros.



- › **Clasificación de imágenes:** La tarea de clasificación de imágenes implica asignar una etiqueta o categoría a una imagen de entrada. Esto implica entrenar un modelo para reconocer y distinguir diferentes objetos, personas o escenas en una imagen.
- › **Detección de objetos:** La detección de objetos implica localizar y clasificar múltiples objetos en una imagen. El objetivo es detectar la presencia y la ubicación de objetos específicos en una escena, a menudo utilizando cuadros delimitadores para delinejar las regiones donde se encuentran los objetos.
- › **Denoising o reconstrucción de imágenes:** Consiste en eliminar o reducir el ruido presente en una imagen, obteniendo una versión más limpia y clara. Esta tarea es relevante en áreas como la fotografía, la medicina y la seguridad.

# Tareas en el campo de visión artificial

Visión computacional profunda

- **Segmentación semántica:** La segmentación semántica implica asignar una etiqueta a cada píxel de una imagen para identificar y delimitar las diferentes regiones o objetos presentes. El objetivo es comprender la estructura y el contenido de una imagen a nivel de píxel.
- **Detección de rostros:** La detección de rostros es una tarea específica de la visión artificial que implica detectar y localizar los rostros en una imagen. Es ampliamente utilizado en aplicaciones de reconocimiento facial, análisis de emociones y sistemas de seguridad.
- **Reconocimiento y verificación facial:** El reconocimiento facial se refiere a la tarea de identificar y reconocer a una persona específica a partir de una imagen o secuencia de imágenes. La verificación facial se enfoca en verificar si una imagen de rostro coincide con una identidad específica.



Escuela  
Nacional de  
Estudios  
Superiores

# Tareas en el campo de visión artificial

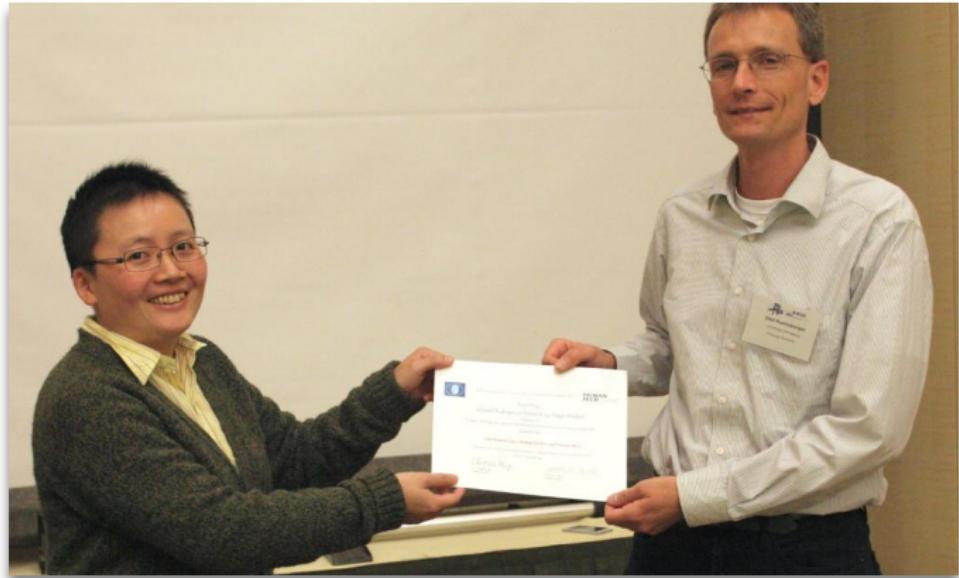
## Visión computacional profunda

- **Estimación de pose:** La estimación de pose se refiere a la tarea de determinar la posición y orientación de un objeto o persona en una imagen. Esto implica detectar y rastrear las articulaciones o puntos clave en una imagen para comprender la postura y el movimiento.
- **Estimación de profundidad:** La estimación de profundidad implica inferir la información de la distancia o la profundidad de los objetos en una imagen. Es útil en aplicaciones de realidad virtual, conducción autónoma y sistemas de navegación.
- **Super-resolución:** La super-resolución se refiere a aumentar la resolución o la calidad de una imagen de baja resolución. El objetivo es generar una versión de alta resolución que capture más detalles y claridad.

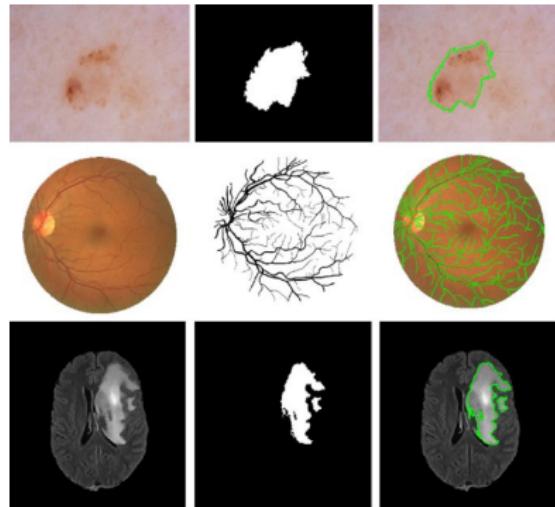
- › From Autoencoder to Beta-VAE
- › Building Autoencoders in Keras
- › Autoencoders: explicación y tutorial en Python
- › Autoencoder For Denoising Images
- › Medical image denoising using convolutional denoising autoencoders

# Segmentación de imágenes médicas

Olaf  
**Ronneberger**  
et al.,  
**2015**

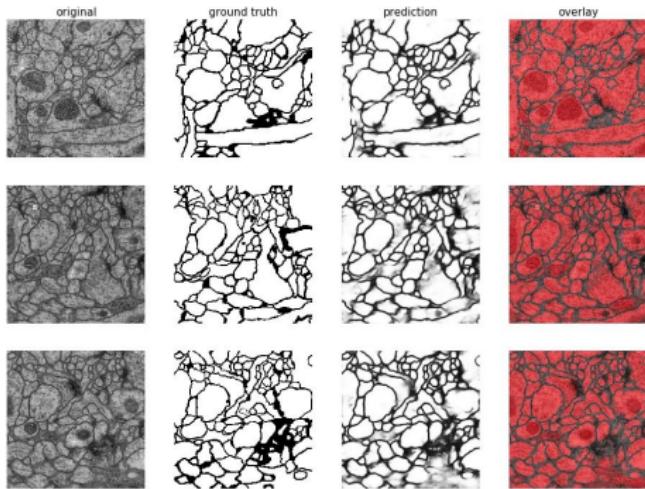


# Segmentación de imágenes



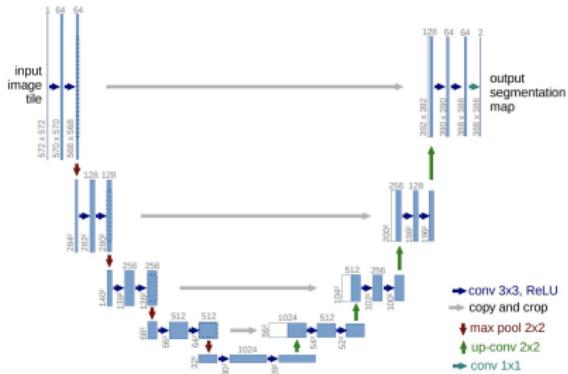
<https://www.nature.com/articles/s41598-021-89686-3>

# Segmentación con U-Net



<https://medium.com/@venkateshtata9/semantic-segmentation-on-medical-images-3ba8264cda5e>

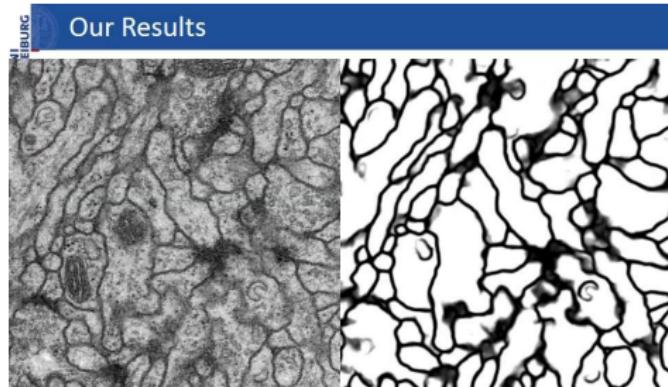
# Segmentación con U-Net



**Fig. 1.** U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

<https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/>

# Segmentación con U-Net



Input image

Our result: 0.000353 warping error

(**New best score** at submission march 6th, 2015)

Sliding-window CNN: 0.000420

Training time: 10h, Application: 1s per image

Olaf Ronneberger, University of Freiburg, Germany, 22.5.2015

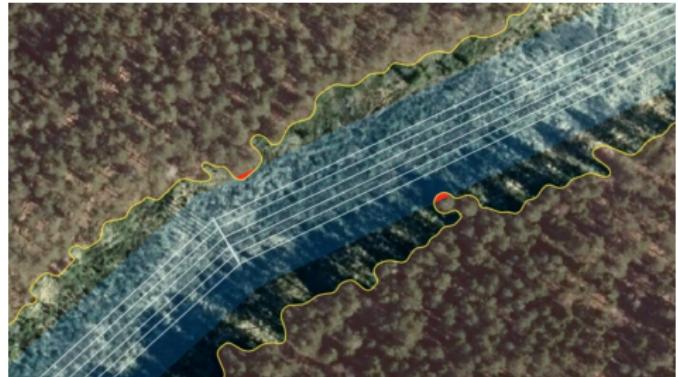
18

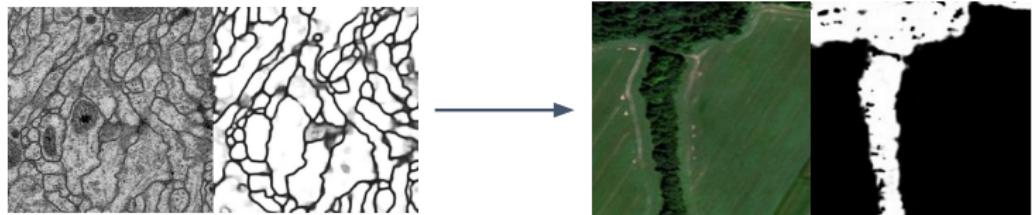
<https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/>



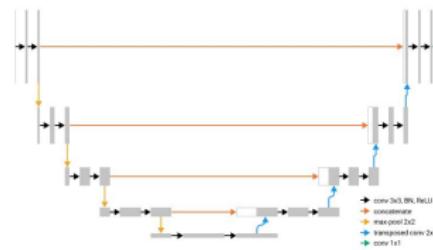
# Caso de estudio

# Prevención de incendios





mdena

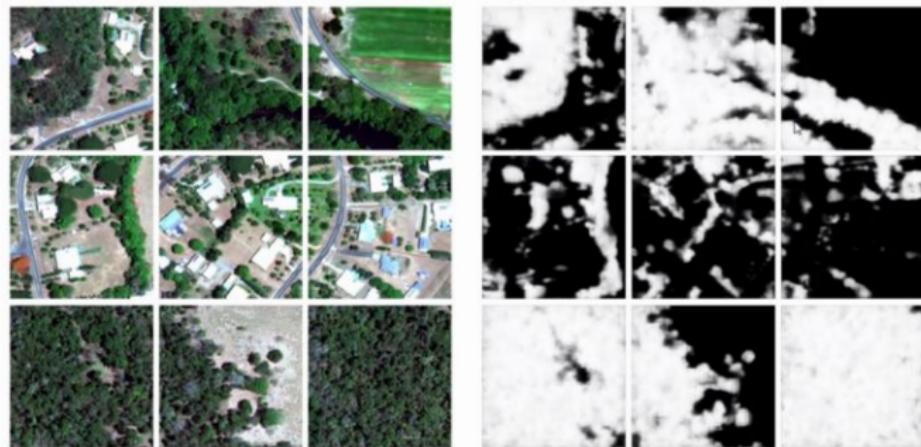


**U-Net:** Convolutional Networks for Biomedical Image Segmentation.

# Prevención de incendios

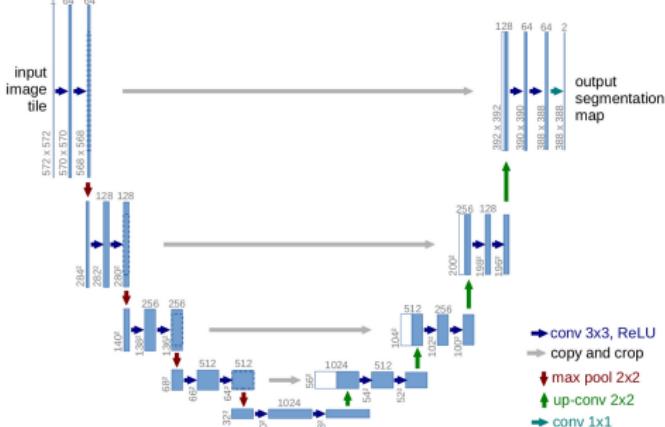
- **Omdena + Spacept**, una startup sueca
- 36 colaborador@s a nivel global (al menos 3 mexicanos)
- Dataset de 200 imágenes satelitales (Australia)
- Desarrollamos un modelo con 95% de precisión
- Desarrollamos un módulo que preprocesa imágenes
- **iResolvimos el challenge!**

<https://omdena.com/projects/ai-prevent-forest-fires/>



<https://omdena.com/projects/ai-prevent-forest-fires/>

### Ejercicio: Entrenamiento de U-Net



## Próxima sesión: Modelado profundo de secuencias (con Mtro. Paco)