Rodolfo Ferro
ferro@cimat.mx
https://rodolfoferro.xyz

Escuela Nacional de Estudios Superiores
Unidad León

Diplomado en Ciencia de Datos
Escuela Nacional de Estudios Superiores, Unidad León

Agosto, 2024

Aprendizaje profundo

Módulo 5

Tutor del módulo

Aprendizaje profundo

Rodolfo Ferro (ferro@cimat.mx)

- Sr. Machine Learning | Computer Vision Engineer @ AIAEC
- Facilitador DS/AI @ DECI ENES León (UNAM) & EdgeHub School of Innovation.
- **Formación:** Esp. en Métodos Estadísticos (c/enf. en DS) @ CIMAT, BMath (UG), CSysEng (UVEG)
- **Experiencia:** Sr. SWE @ Bisconic (US), ML Engineer @ Vindoo.ai (España), Sherpa Digital de IA @ Microsoft México, AI Research Assistant @ CIMAT & AI Research Intern @ Harvard.

ferro@cimat.mx

Unidad León Escuela Nacional de Estudios Superiores

Tabla de contenidos

ferro@cimat.mx

Section 1: **Intro al aprendizaje profundo**

Motivación

Intro al aprendizaje profundo

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

ferro@cimat.mx

Motivación

Intro al aprendizaje profundo

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

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

ferro@cimat.mx

Motivación

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;
}
```

ferro@cimat.mx

Motivación

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;
}
```

```
// ????
```

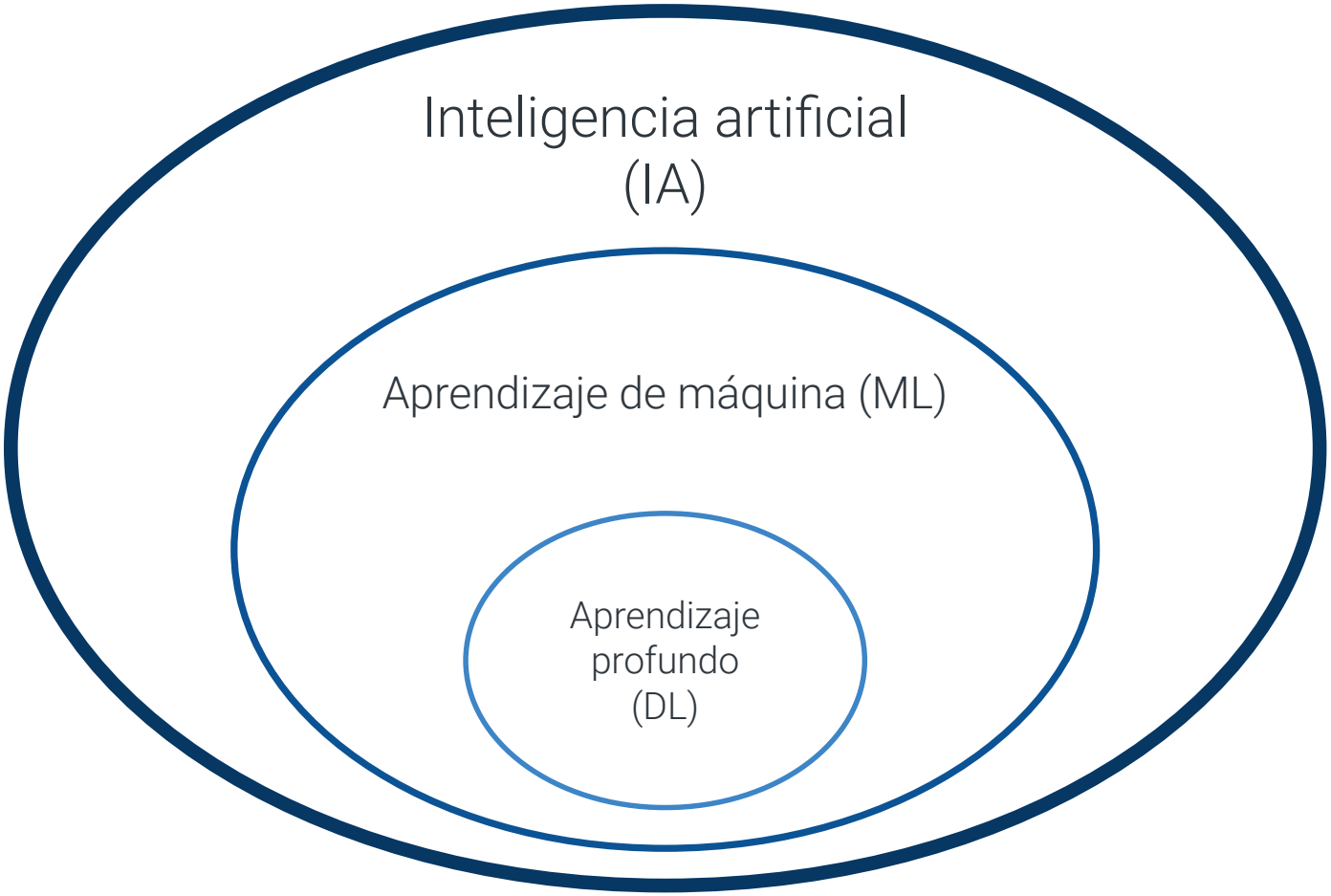
ferro@cimat.mx

¿Qué es el *Deep Learning*?

> *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)

ferro@cimat.mx

¿Qué es el *Deep Learning*?



ferro@cimat.mx

¿Qué es el *Deep Learning*?

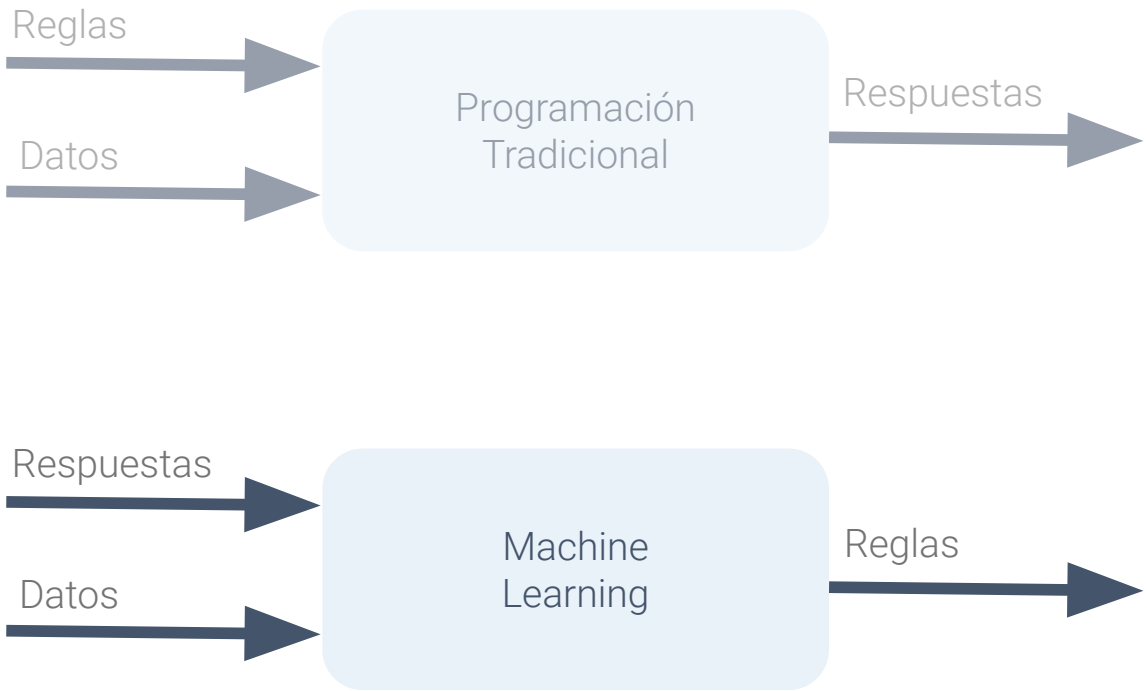
- **Inteligencia artificial:** Cualquier técnica que permita a las computadoras emular o imitar el comportamiento humano.
- **Aprendizaje de máquina:** Capacidad de aprender sin ser programado explícitamente, enfoque en los algoritmos y la matemtica.
- **Aprendizaje profundo:** Extrae patrones de datos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

ferro@cimat.mx

¿Por qué el *Deep Learning*?

ferro@cimat.mx

¿Por qué el *Deep Learning*?



ferro@cimat.mx

¿Por qué el *Deep Learning*?

- La ingeniería de características requiere mucho tiempo, es suceptible a errores y no es escalable consistentemente con datos complejos. Mejor busquemos aprender las características subyacentes directamente de los datos.
- Las redes neuronales artificales 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.

ferro@cimat.mx

Contexto histórico

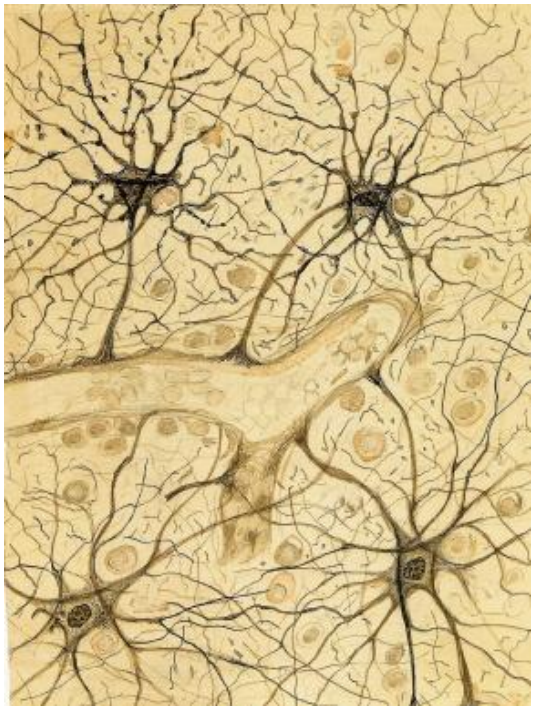
Intro al aprendizaje profundo

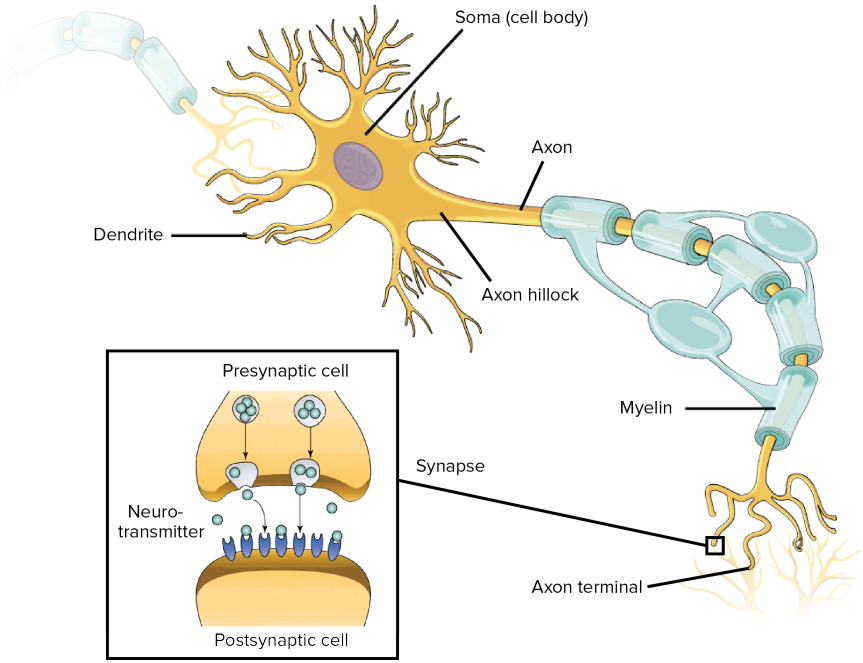
Figure: Santiago Ramón y Cajal

ferro@cimat.mx

Contexto histórico

Intro al aprendizaje profundo





ferro@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

TLU

Intro al aprendizaje profundo

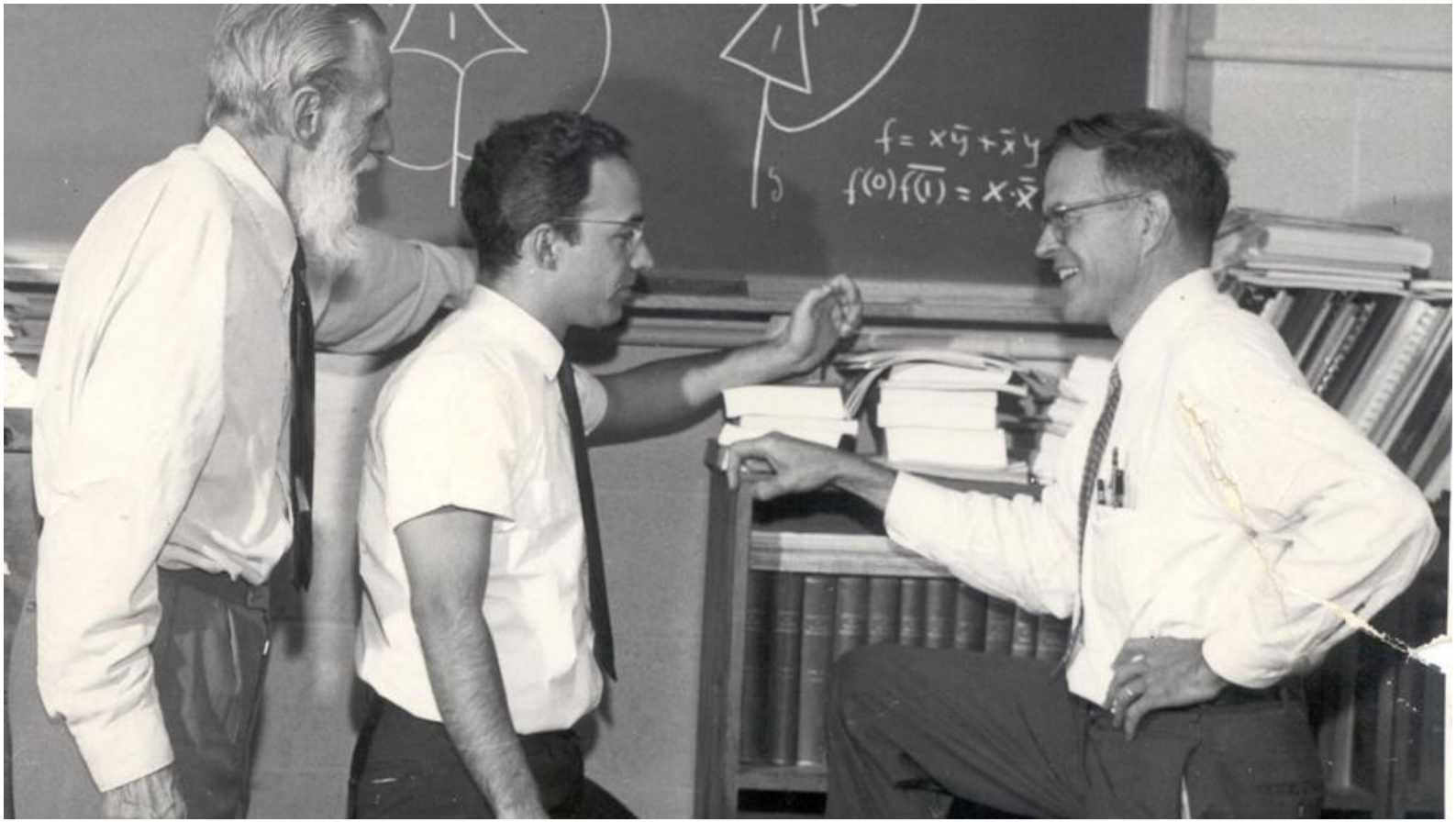
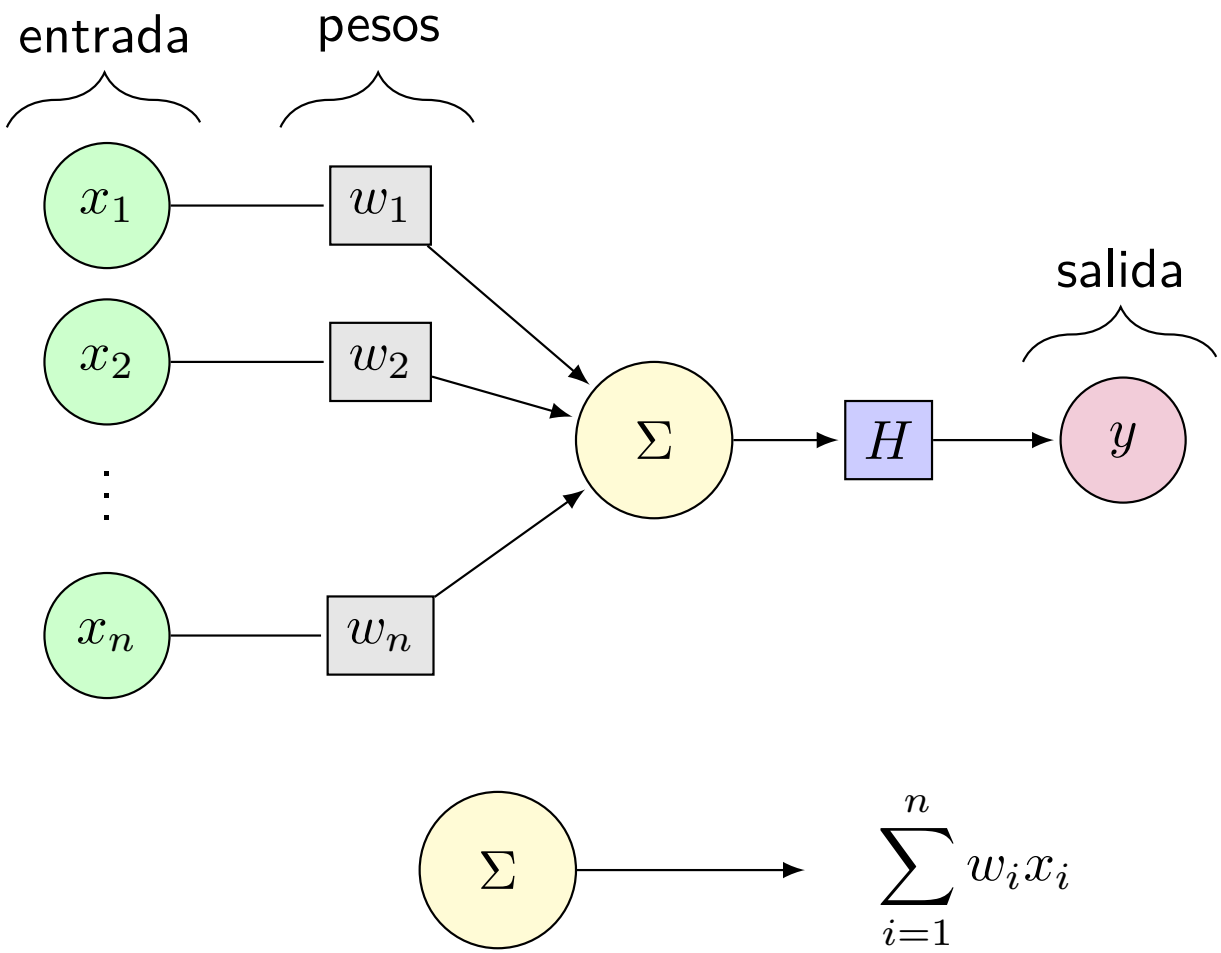


Figure: Warren McCulloch & Walter Pitts

ferro@cimat.mx

TLU



ferro@cimat.mx

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

ferro@cimat.mx

Bias y función de activación

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

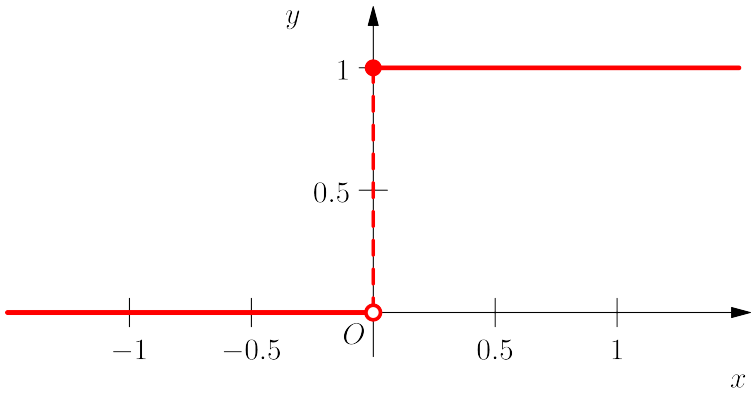
$$f(\mathbf{x}) = \begin{cases} 0 & \text{si } \sum_i w_i x_i + b < 0 \\ 1 & \text{si } \sum_i w_i x_i + b > 0 \end{cases}$$

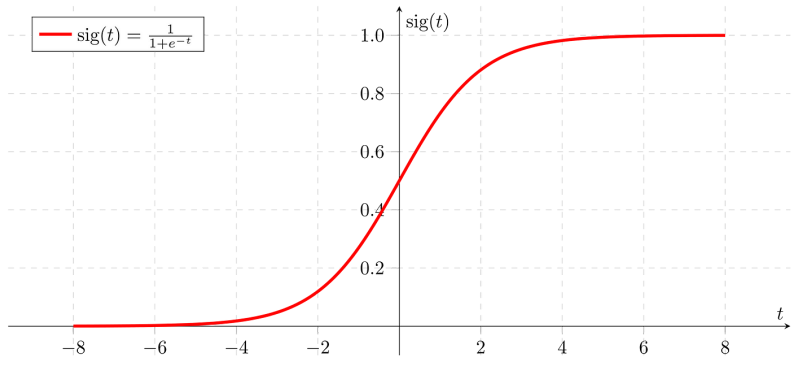
donde $\mathbf{x} = (x_1, x_2, ..., x_n)$ y $i \in \{1, 2, ..., n\}$.
A esto que escribimos como b, también se le conoce como **bias**, y una interpretación es describir *qué tan susceptible es la neurona a* ***dispararse*** (como se exploró en el ejemplo práctico de la identificación de la actividad de Julieta).

ferro@cimat.mx

Bias y función de activación

Intro al aprendizaje profundo





ferro@cimat.mx

El Perceptrón

Intro al aprendizaje profundo

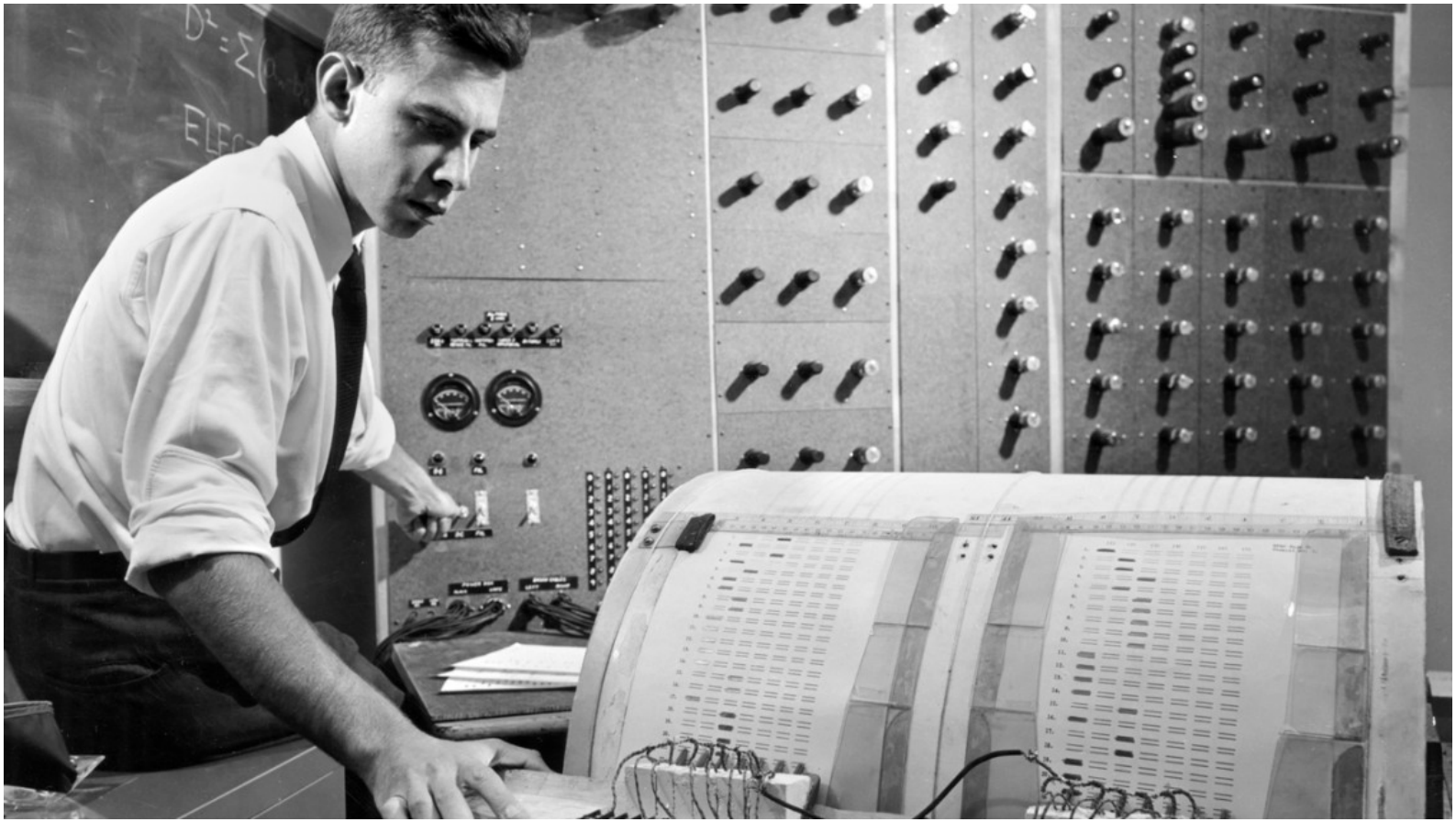
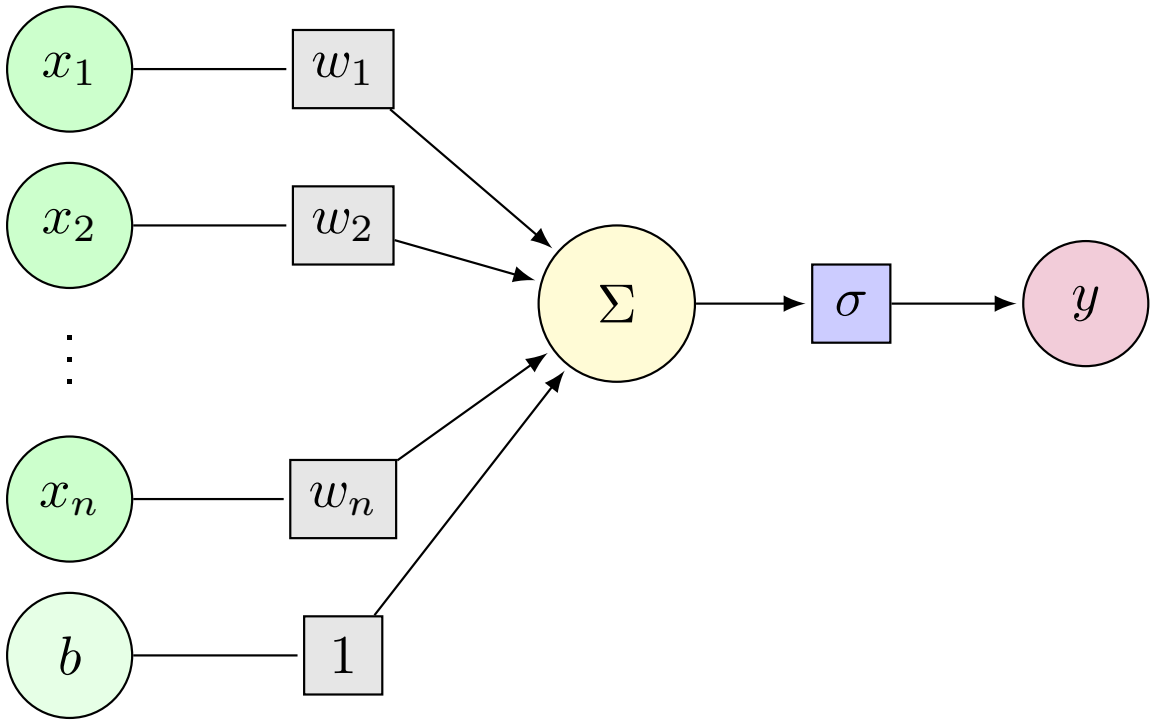


Figure: Frank Rosenblatt

ferro@cimat.mx

El Perceptrón



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

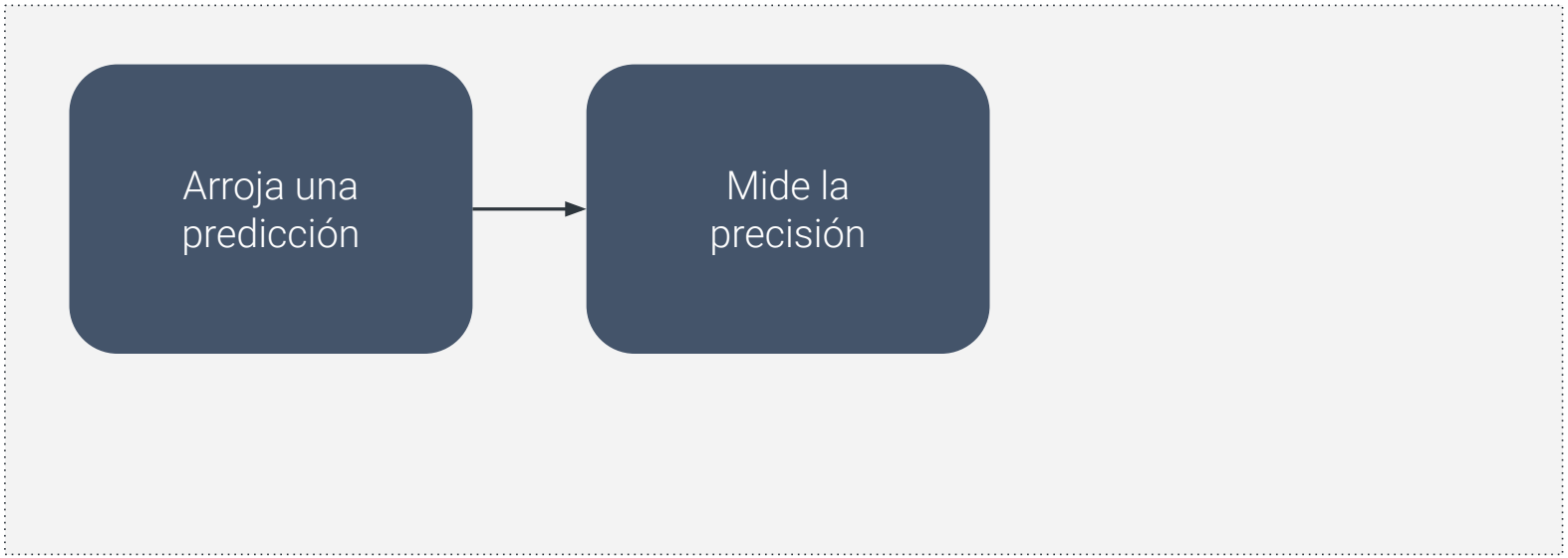
ferro@cimat.mx

Idea intuitiva de entrenamiento

Arroja una predicción

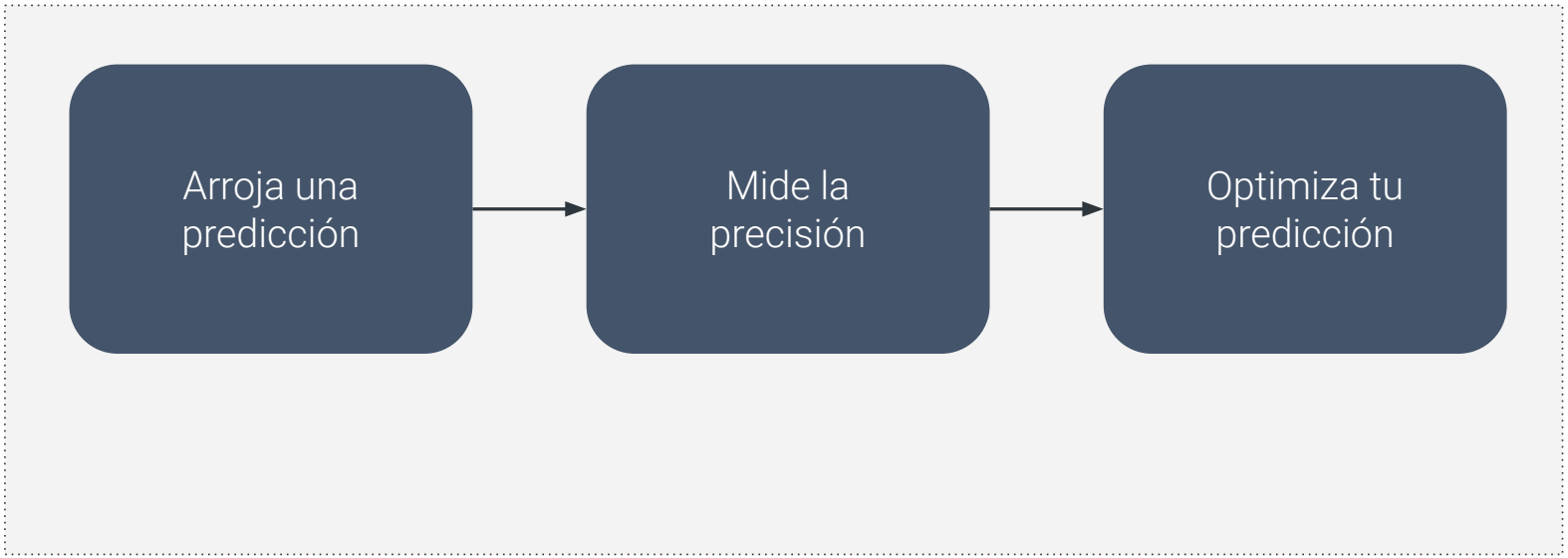
ferro@cimat.mx

Idea intuitiva de entrenamiento



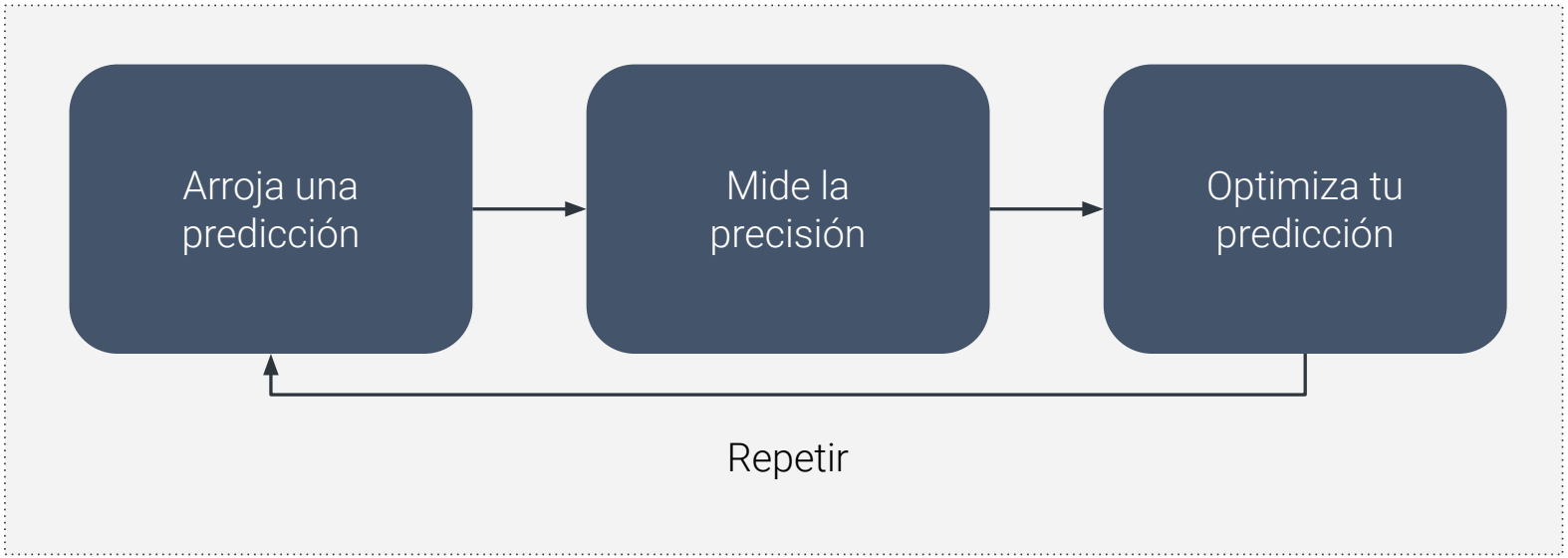
ferro@cimat.mx

Idea intuitiva de entrenamiento



ferro@cimat.mx

Idea intuitiva de entrenamiento



ferro@cimat.mx

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

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

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

ferro@cimat.mx

Medición del error

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

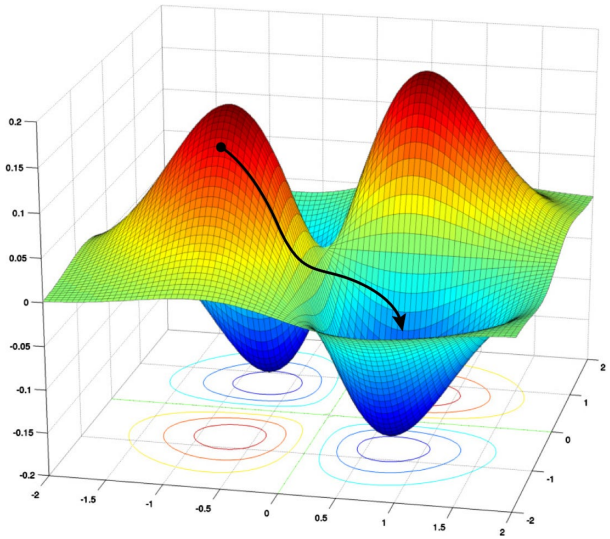
$$X = \begin{bmatrix} 0.5 \\ 0.3 \\ 0.7 \end{bmatrix}$$

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

ferro@cimat.mx

Optimización del error

- **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



ferro@cimat.mx

Observaciones

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?**

ferro@cimat.mx

Ejercicio: Manzanas vs. Naranjas

ferro@cimat.mx

Lecturas recomendadas

- 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

ferro@cimat.mx

Producto matricial

Recordemos cómo opera el producto matricial:

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

ferro@cimat.mx

Producto matricial

Recordemos cómo opera el producto matricial:

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

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

ferro@cimat.mx

Producto matricial

Recordemos cómo opera el producto matricial:

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

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

ferro@cimat.mx

Producto matricial

Recordemos cómo opera el producto matricial:

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

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

ferro@cimat.mx

Producto matricial

Recordemos cómo opera el producto matricial:

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

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

ferro@cimat.mx

Producto matricial

Recordemos cómo opera el producto matricial:

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

$$\dots$$

ferro@cimat.mx

Producto matricial

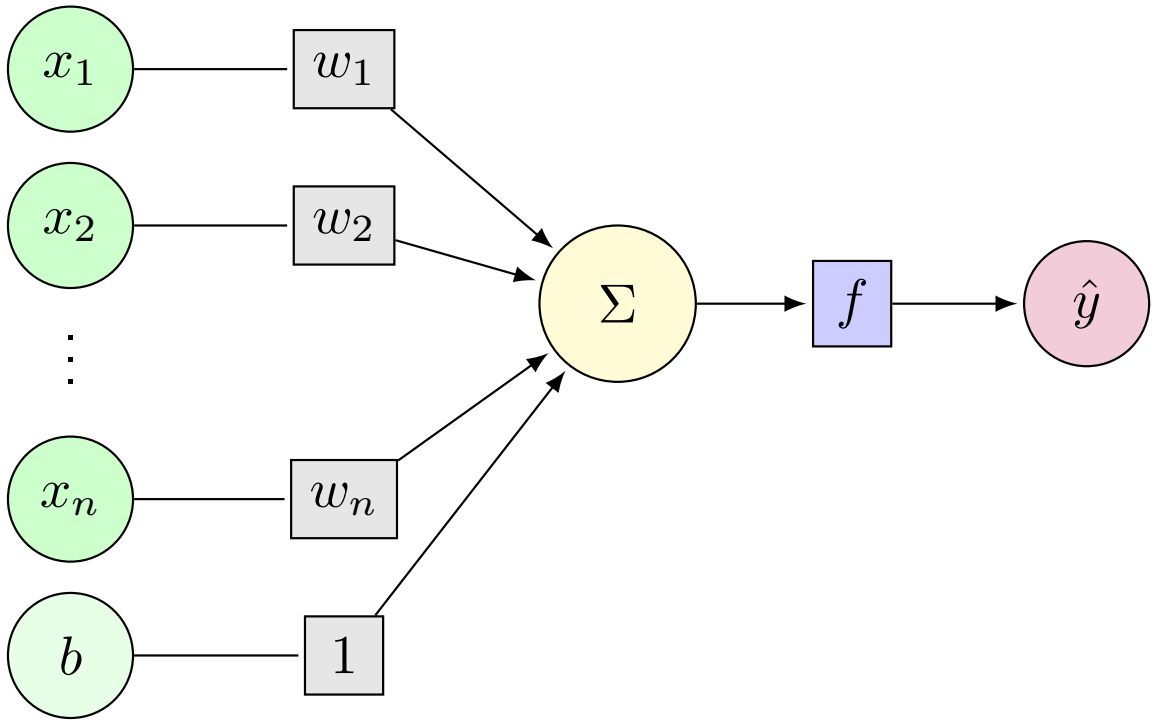
Recordemos cómo opera el producto matricial:

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

ferro@cimat.mx

El Perceptrón

Intro al aprendizaje profundo

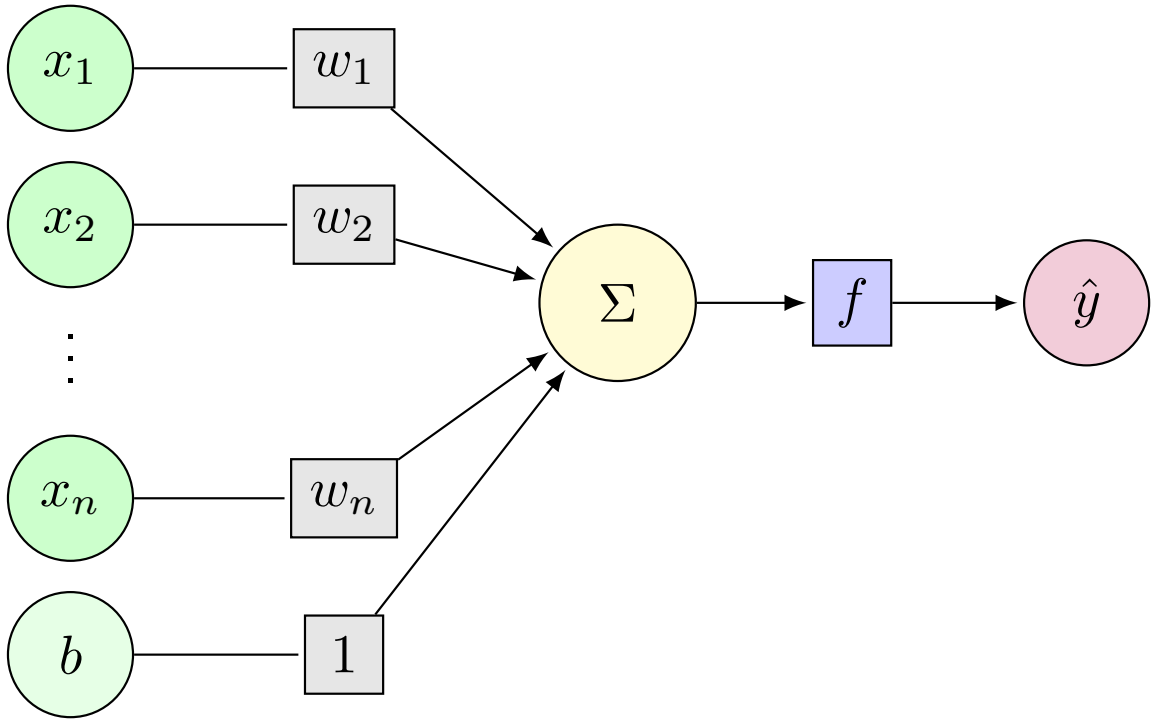


$$\hat{y} = f\left(\sum_i^n w_i x_i + b\right)$$

ferro@cimat.mx

El Perceptrón

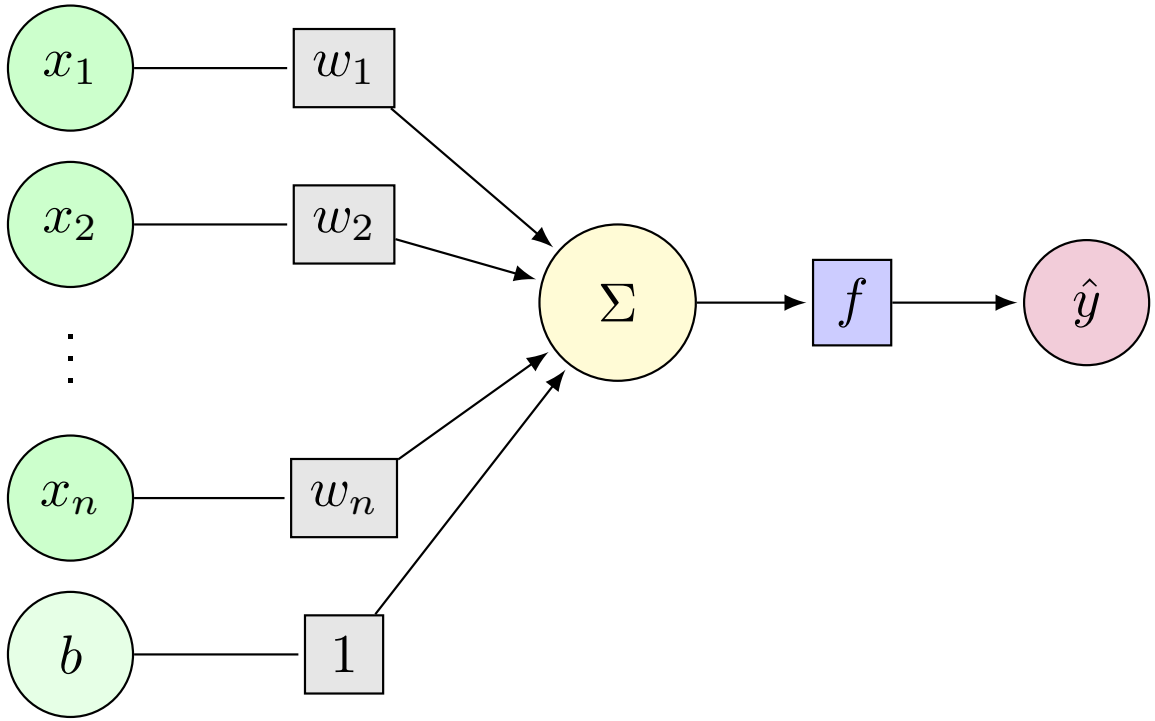
Intro al aprendizaje profundo



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

ferro@cimat.mx

El Perceptrón

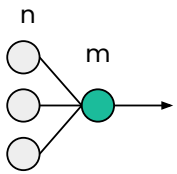


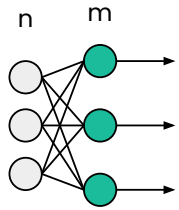
$$\mathbf{w}^{\mathbf{T}}\mathbf{x} = [w_1 w_2 \cdots w_n] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

ferro@cimat.mx

Producto matricial

Intro al aprendizaje profundo





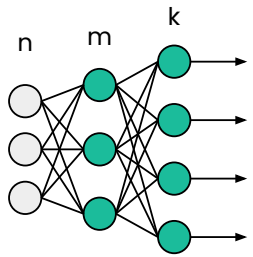
$$\mathbf{W}\cdot\mathbf{X}+\mathbf{b} = [w_1 \;\; w_2 \;\; \cdots \;\; w_n]\cdot\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + b \longrightarrow \mathbf{W}_{Layer}\cdot\mathbf{X}+\mathbf{b} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\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

$$f(\cdot)$$

ferro@cimat.mx

Producto matricial

Intro al aprendizaje profundo



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

$f(\cdot)$

ferro@cimat.mx

Composición de funciones

Intro al aprendizaje profundo

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

$$\downarrow$$

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

$$\downarrow$$

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

ferro@cimat.mx

Composición de funciones

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

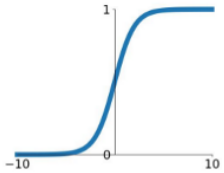
$$F'(x) = f'(g(x)) \cdot g'(x)$$

ferro@cimat.mx

Funciones de activación

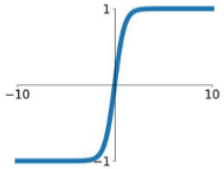
Sigmoid

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



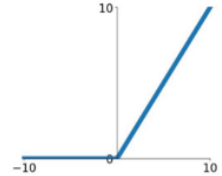
tanh

$\tanh(x)$



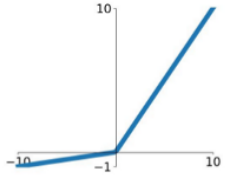
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

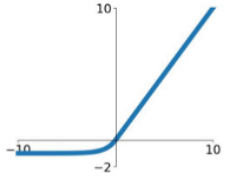


Maxout

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

ELU

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

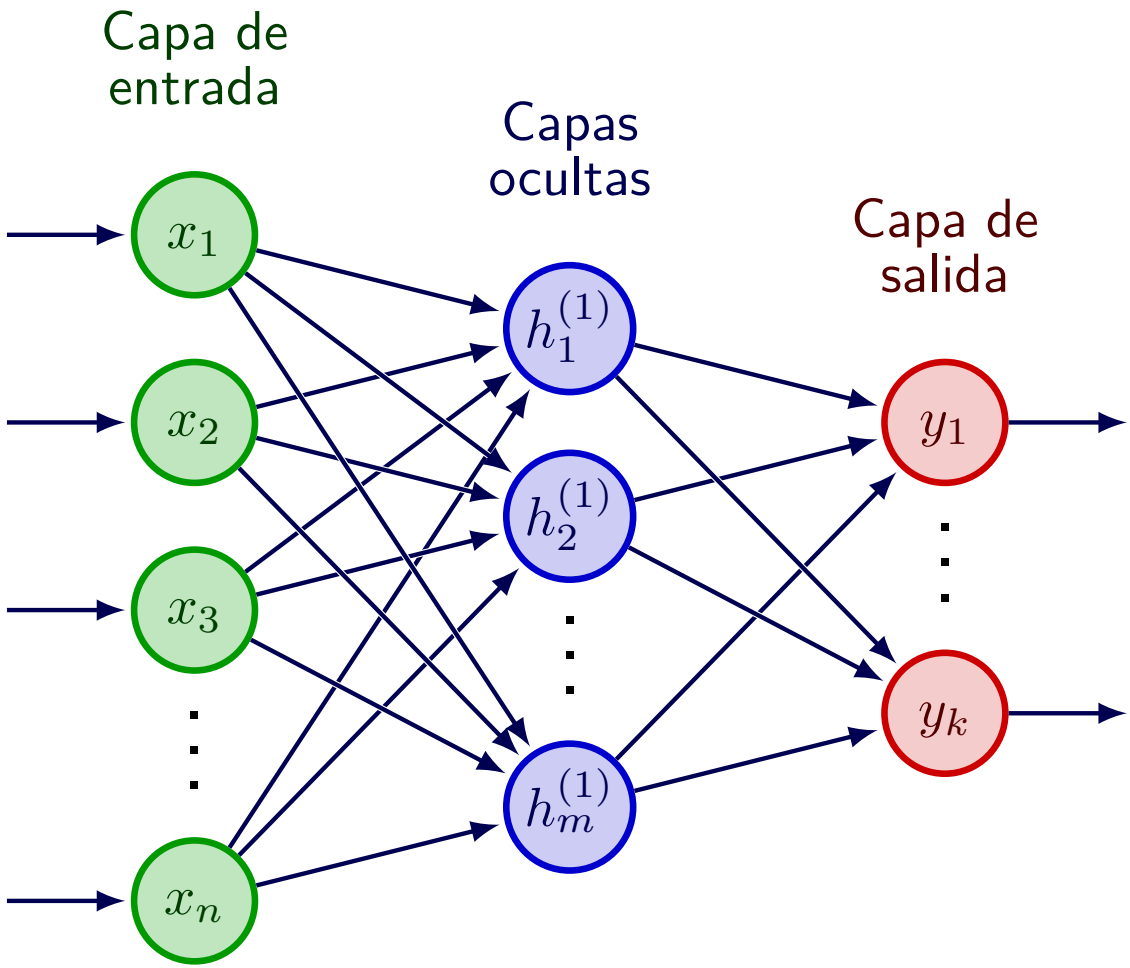


$$f(x) = \sigma(x) = \frac{1}{1+e^{-x}} \Rightarrow f'(x) = f(x)(1-f(x))$$

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

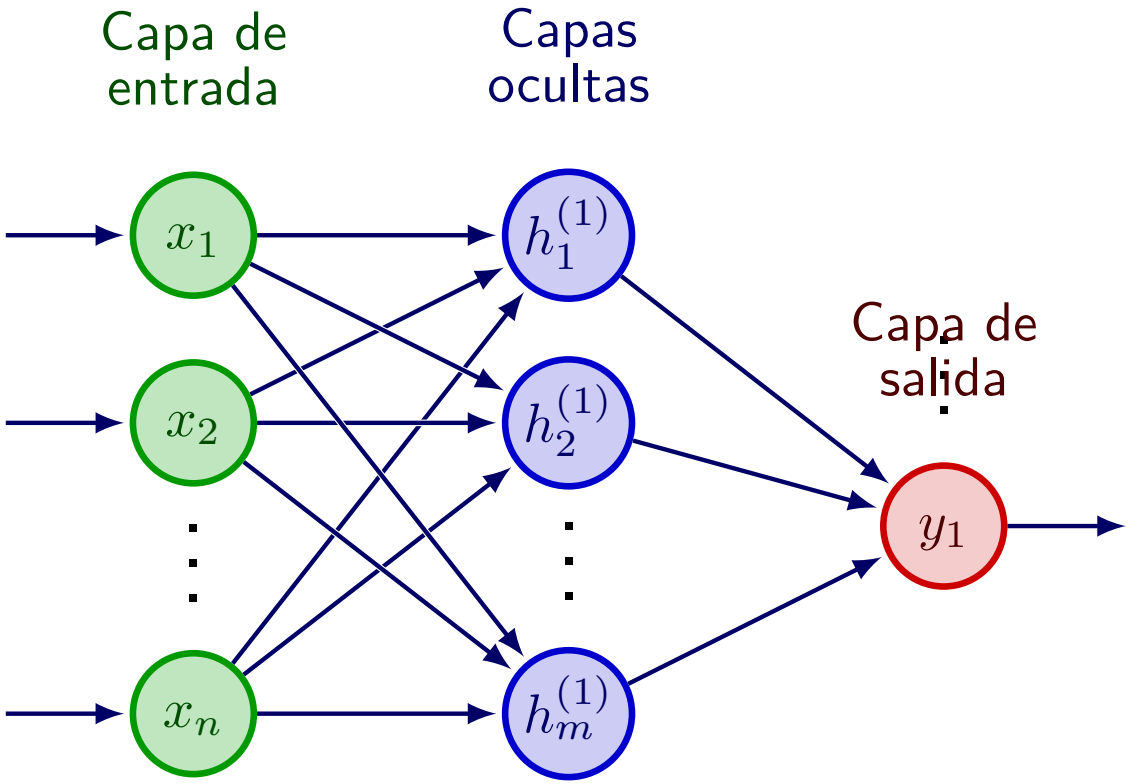
ferro@cimat.mx

El perceptrón multicapa



ferro@cimat.mx

El perceptrón multicapa



Para $\mathbf{x}^{(i)} = (x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)})$, $y^{(i)}$, tendríamos que la salida es

$$\hat{y}_1^{(i)} = f(x^{(i)}).$$

ferro@cimat.mx

Cuantificación del error

- 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.

ferro@cimat.mx

Cuantificación del error

- 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).

ferro@cimat.mx

Algunos ejemplos

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

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

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

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

ferro@cimat.mx

Optimización del error

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

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

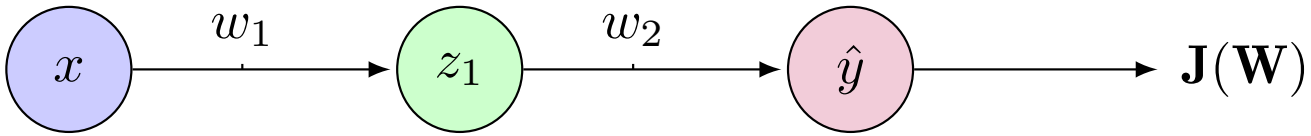
$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}} \mathbf{J}(\mathbf{W})$$

ferro@cimat.mx

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*

ferro@cimat.mx

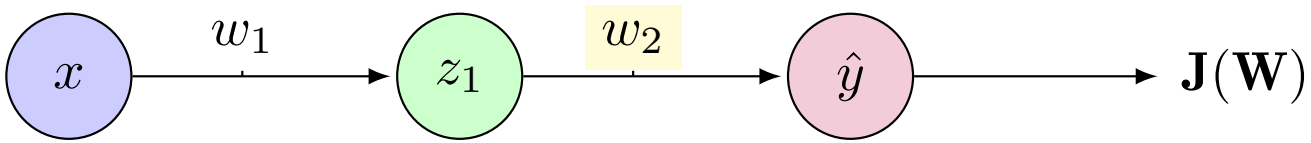


¿Cómo se calculan los gradientes?
Con la regla de la cadena.

ferro@cimat.mx

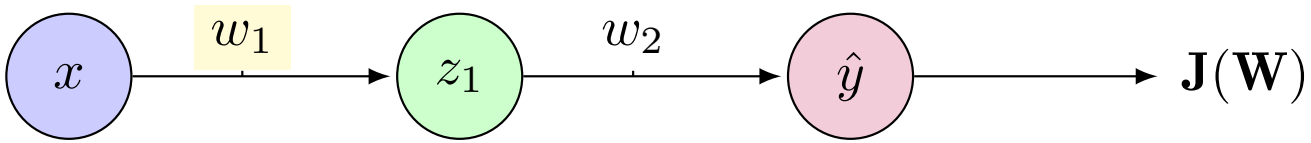
Retropropagación

Intro al aprendizaje profundo



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

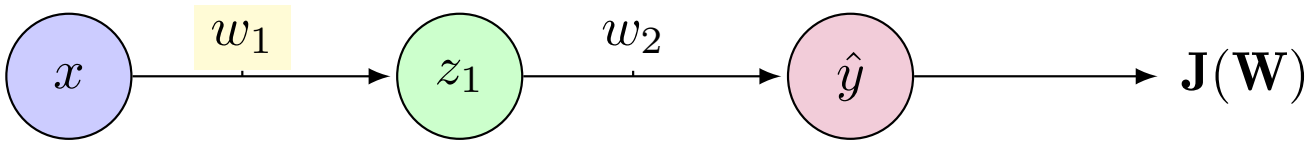
ferro@cimat.mx



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

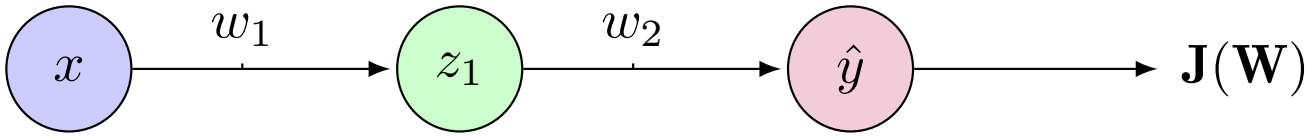
ferro@cimat.mx

Retropropagación



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

ferro@cimat.mx

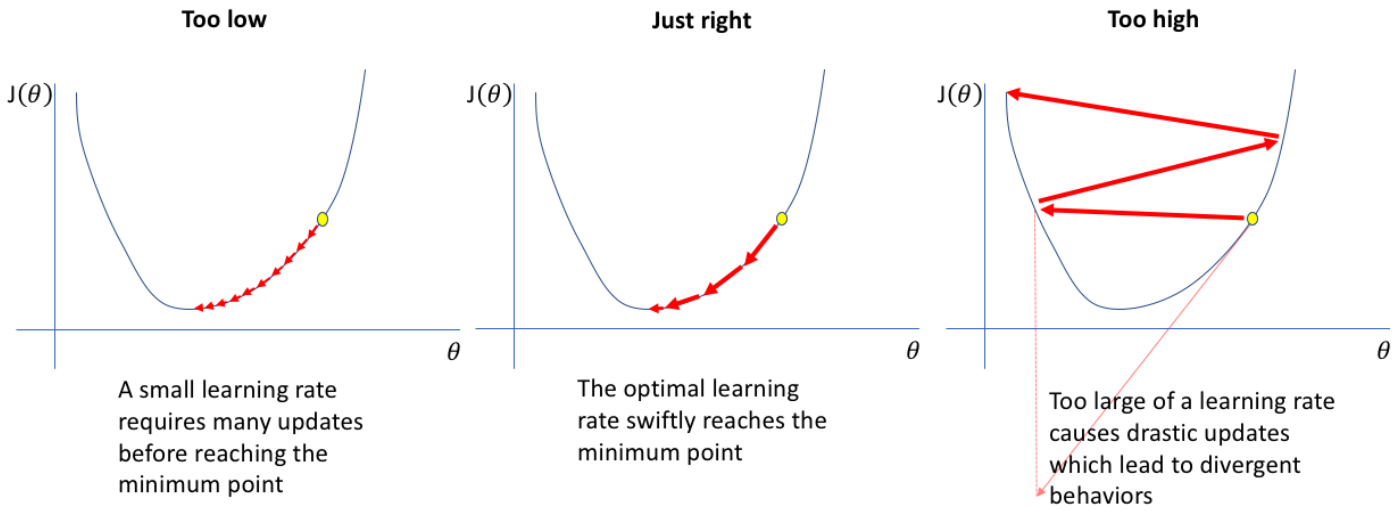


¿Cómo se calculan los gradientes?
Con la regla de la cadena.
Esto se repite para **cada peso** en la red neuronal, usando los gradientes de las capas posteriores.

ferro@cimat.mx

La actualización de pesos está dada por:

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



ferro@cimat.mx

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).

ferro@cimat.mx

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.

Descenso de gradiente 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.

ferro@cimat.mx

Observaciones

- 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.

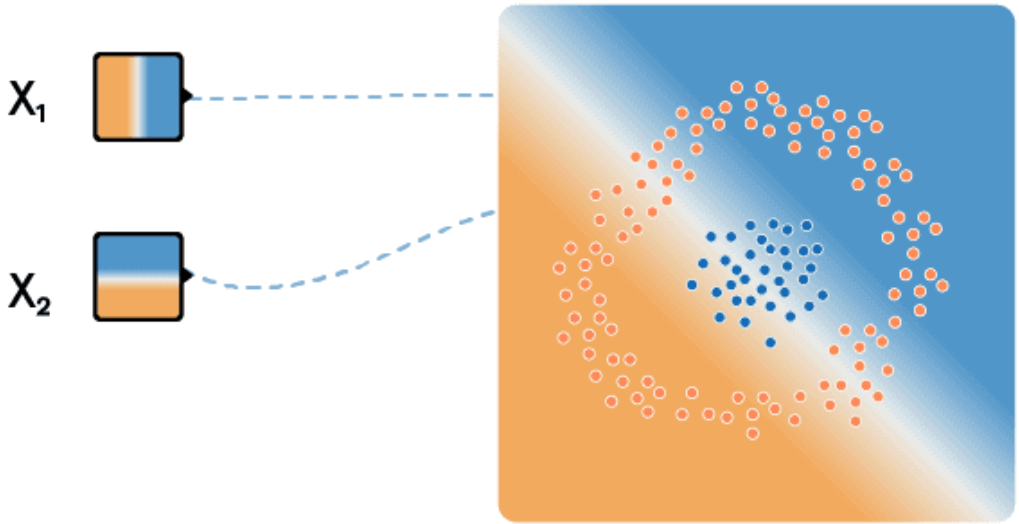
ferro@cimat.mx

TensorFlow

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

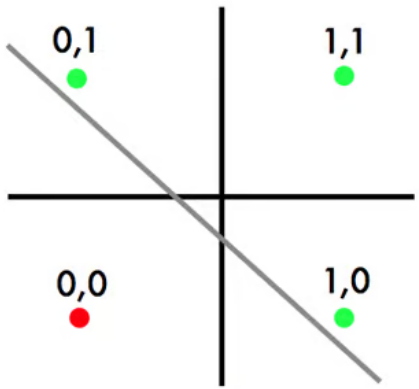
ferro@cimat.mx

Ejercicio: Exploración del TensorFlow Playground

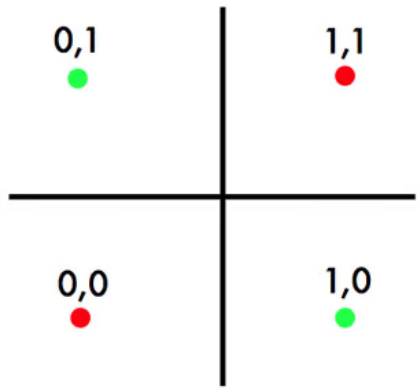


ferro@cimat.mx

Ejercicio: Problema de separabilidad lineal



OR



XOR

ferro@cimat.mx

Ejercicio: Exploración con TensorFlow

TensorFlow

ferro@cimat.mx

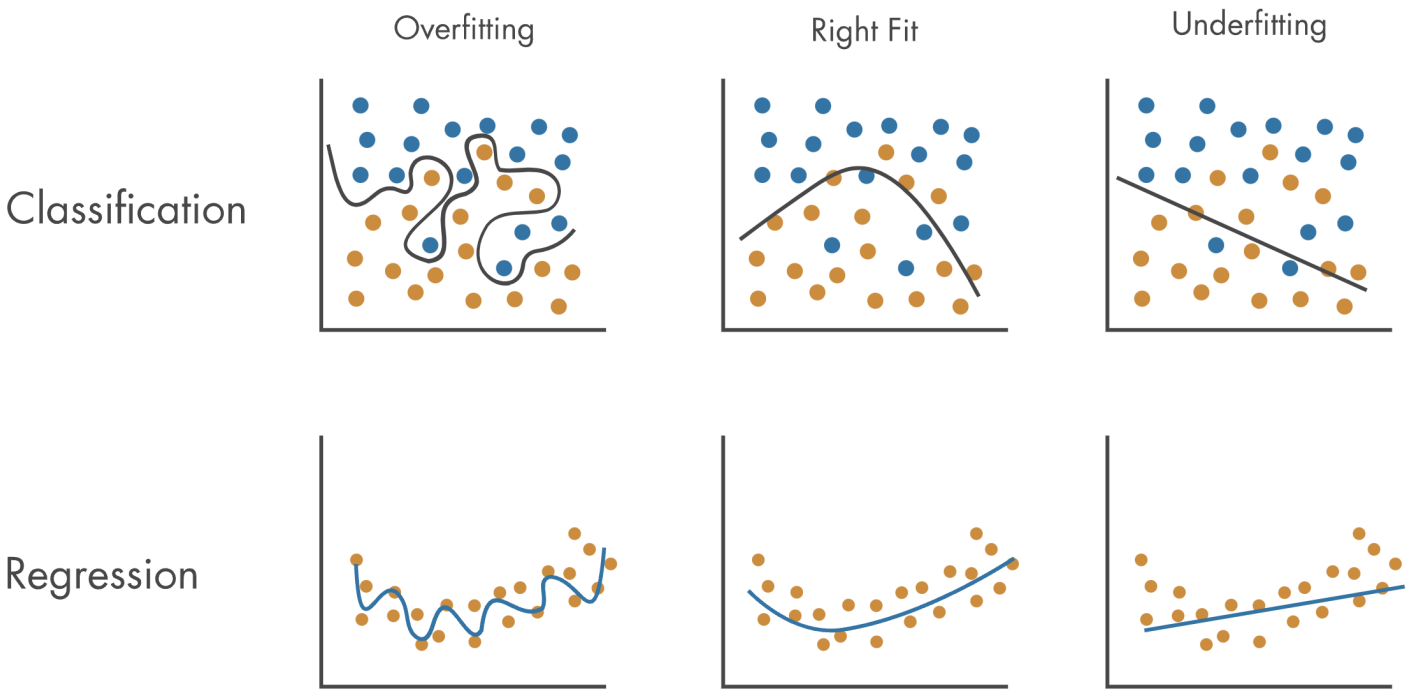
Lecturas recomendadas

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

ferro@cimat.mx

El problema del overfitting

Intro al aprendizaje profundo



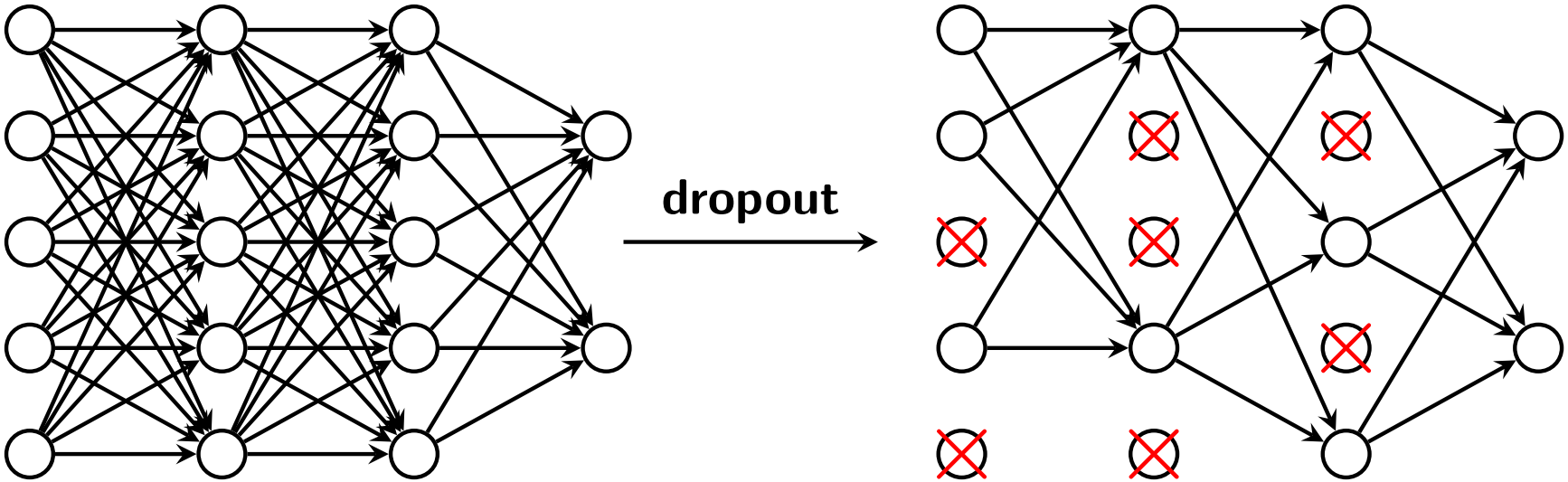
ferro@cimat.mx

Regularización

- 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*

ferro@cimat.mx

Dropout



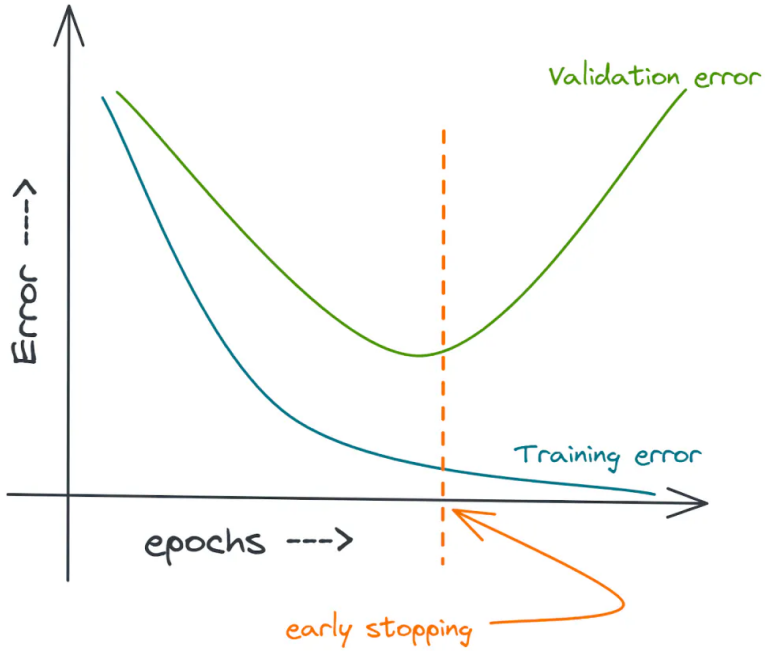
ferro@cimat.mx

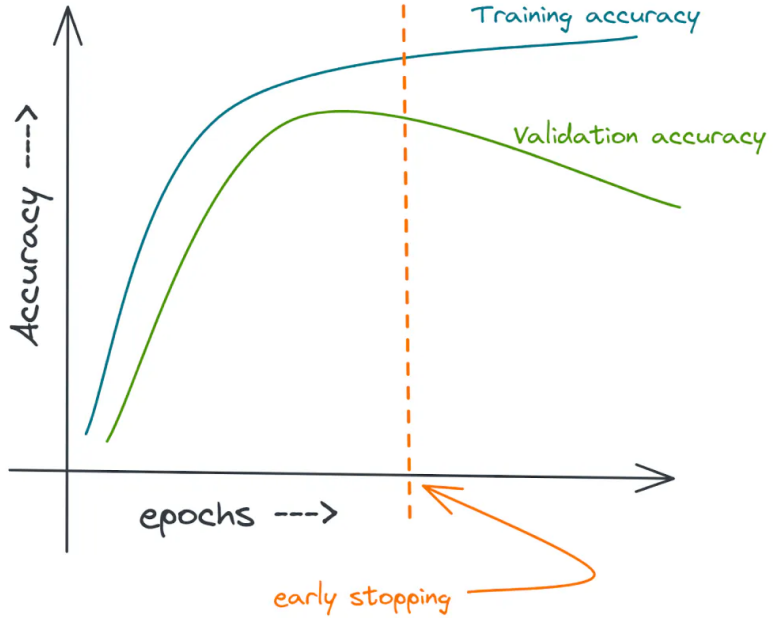
- Durante el entrenamiento, establecemos aleatoriamente algunas activaciones en 0
  - Típicamente hacemos "drop" del 50% de activaciones en una capa.
  - Esto forza a la red a no depender de ningún nodo/neurona.
- Podemos realizar el dropout en TensorFlow utilizando la capa `tf.keras.layers.Dropout(0.5)`,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

ferro@cimat.mx

Early stopping

Intro al aprendizaje profundo





ferro@cimat.mx

Early stopping

- 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])
```

ferro@cimat.mx

Lecturas recomendadas

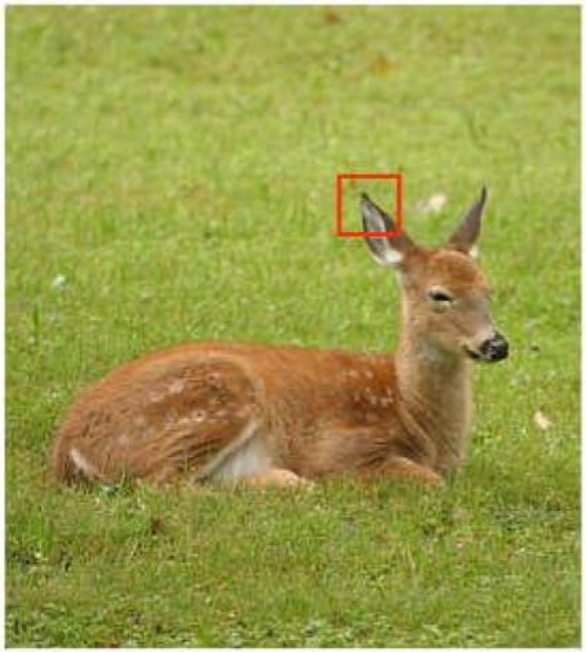
- Artículo "Dropout: A Simple Way to Prevent Neural Networks from Overfitting"
- Regularización en Redes Neuronales
- Vanishing and Exploding Gradients in Neural Network Models: Debugging, Monitoring, and Fixing

ferro@cimat.mx

Section 2: **Visión computacional profunda**

¿Qué es una imagen?

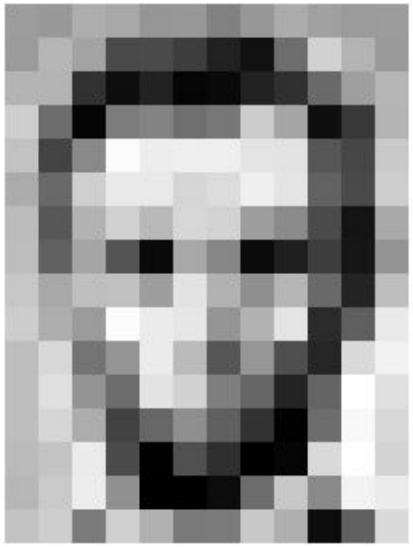
Visión computacional profunda



ferro@cimat.mx

¿Qué es una imagen?

Visión computacional profunda



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	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	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	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	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	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	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	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

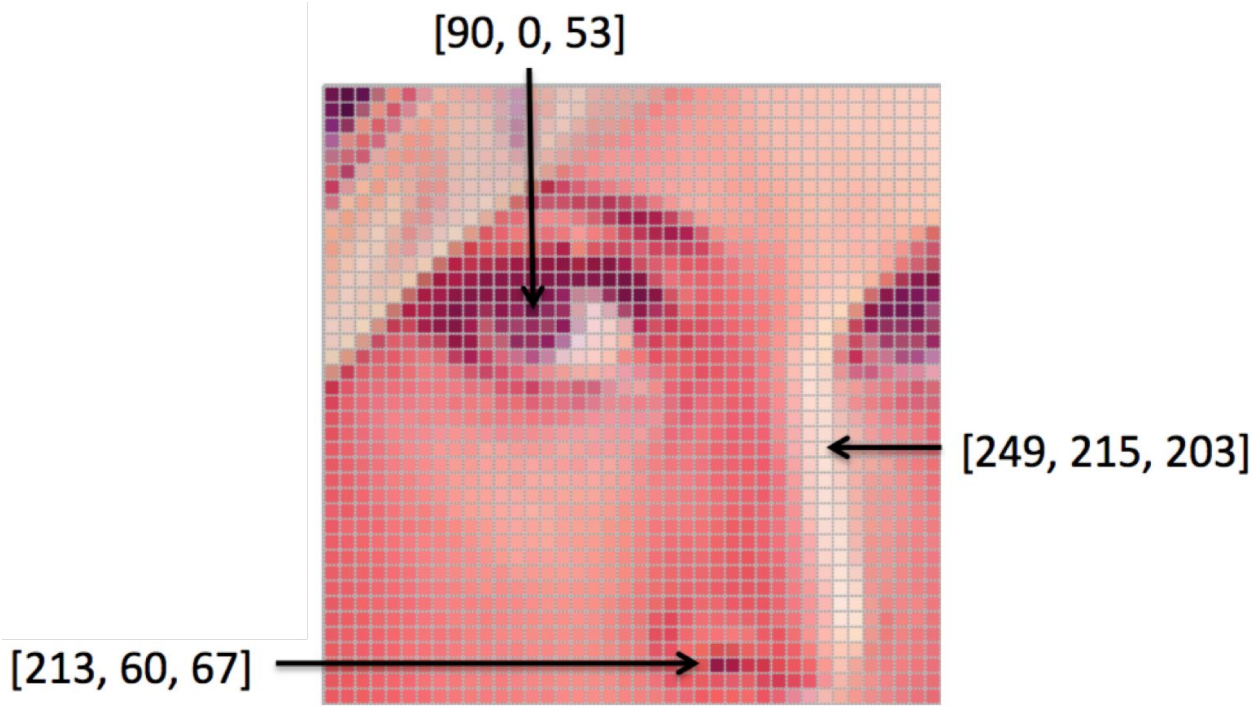
ferro@cimat.mx

Unidad León Escuela Nacional de Estudios Superiores

"Tutorial 1: Image Filtering." AI Stanford. https://ai.stanford.edu/~syyeung/cvweb/tutorial1.html

¿Qué es una imagen?

Visión computacional profunda



ferro@cimat.mx

"Tutorial 1: Image Filtering." AI Stanford. https://ai.stanford.edu/~syyeung/cvweb/tutorial1.html

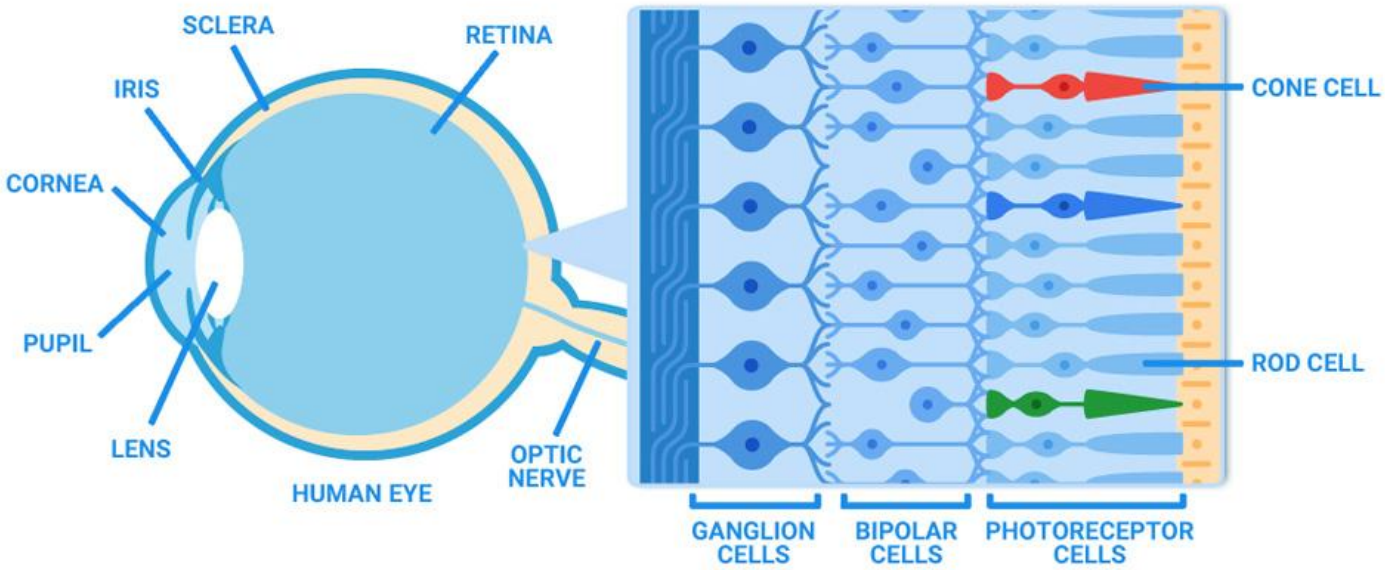
¿Qué es una imagen?

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

ferro@cimat.mx

La biología humana

Visión computacional profunda

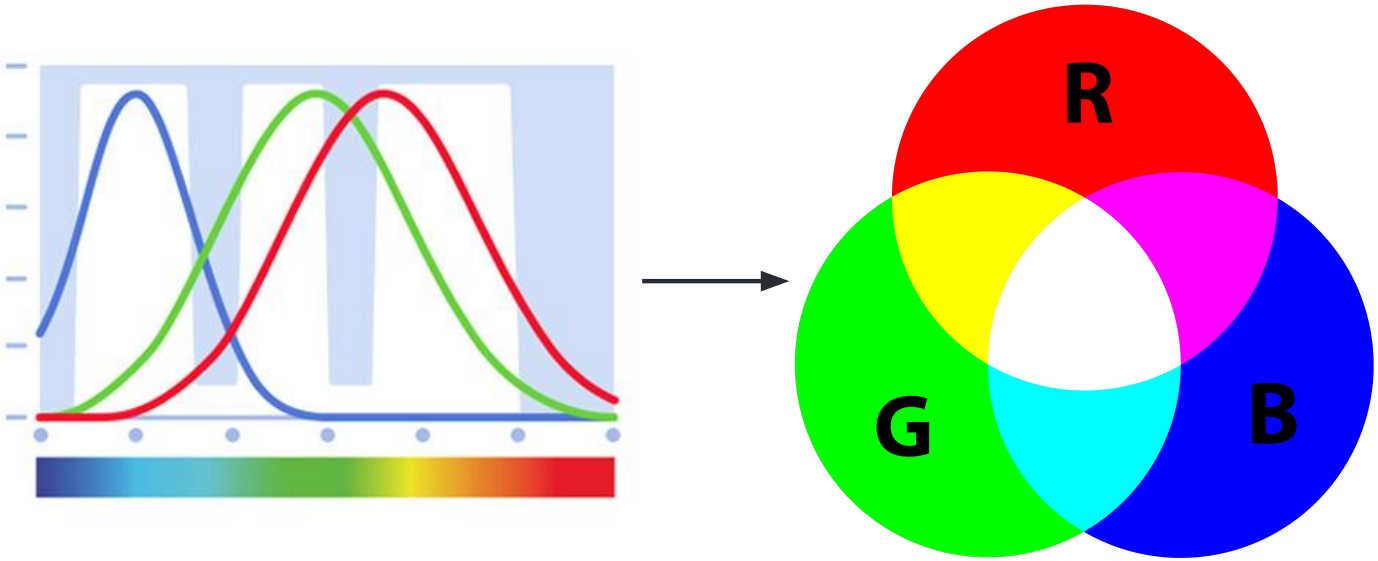


ferro@cimat.mx

"How EnChroma Color Blind Glasses Work." EnChroma. http://enchroma.com/technology/

La biología humana

Visión computacional profunda

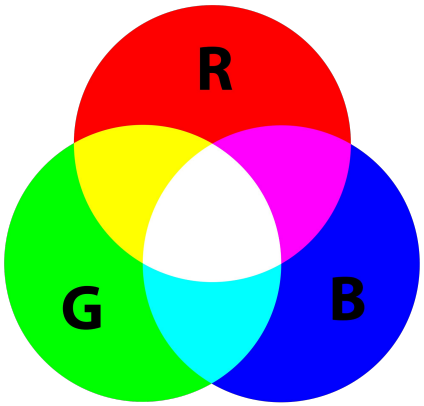


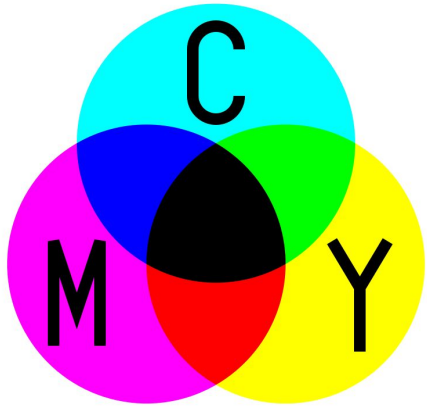
ferro@cimat.mx

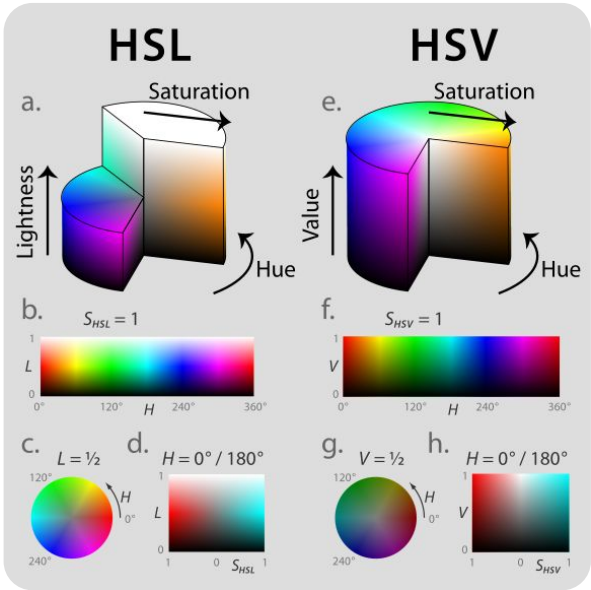
"How EnChroma Color Blind Glasses Work." EnChroma. http://enchroma.com/technology/

Espacios de color

Visión computacional profunda

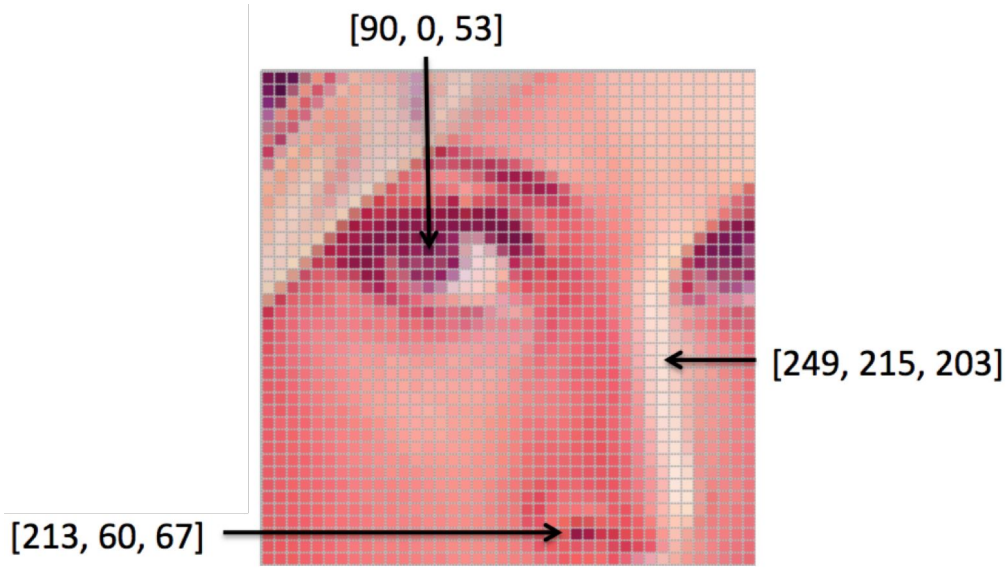






ferro@cimat.mx

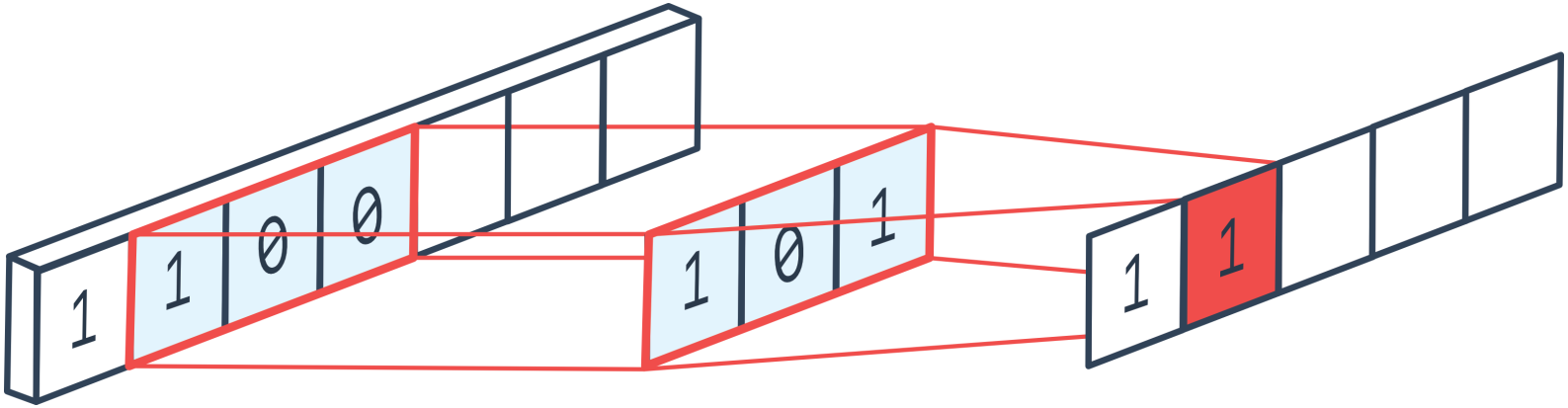
Ejercicio: Introducción a imágenes



ferro@cimat.mx

¿Qué es una convolución?

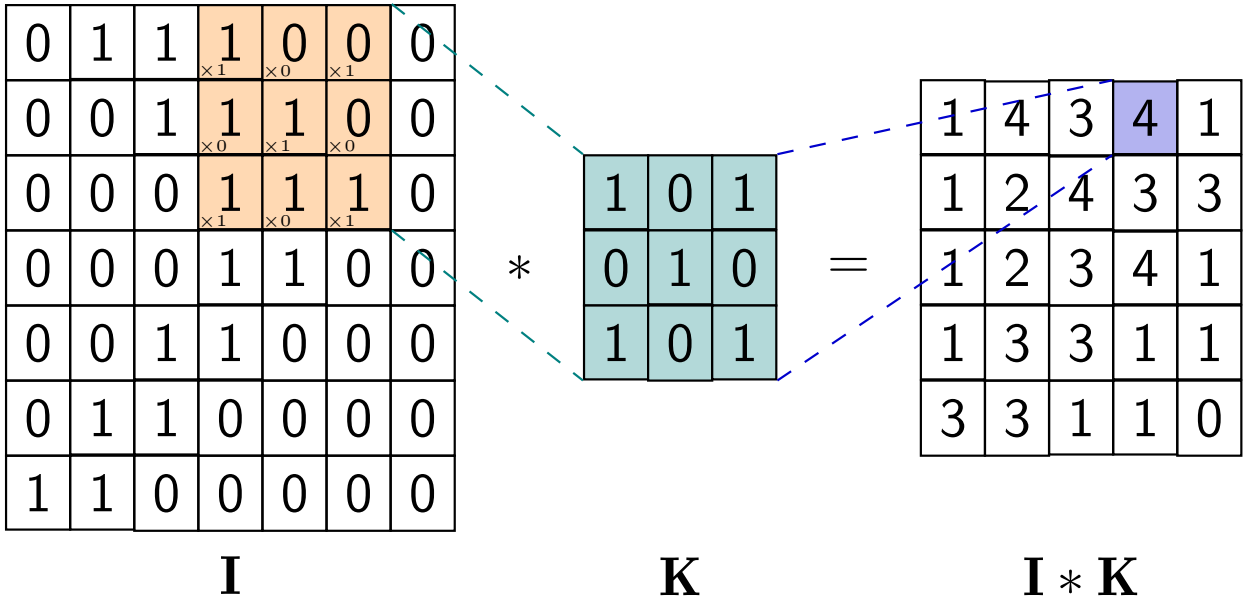
Visión computacional profunda



ferro@cimat.mx

¿Qué es una convolución?

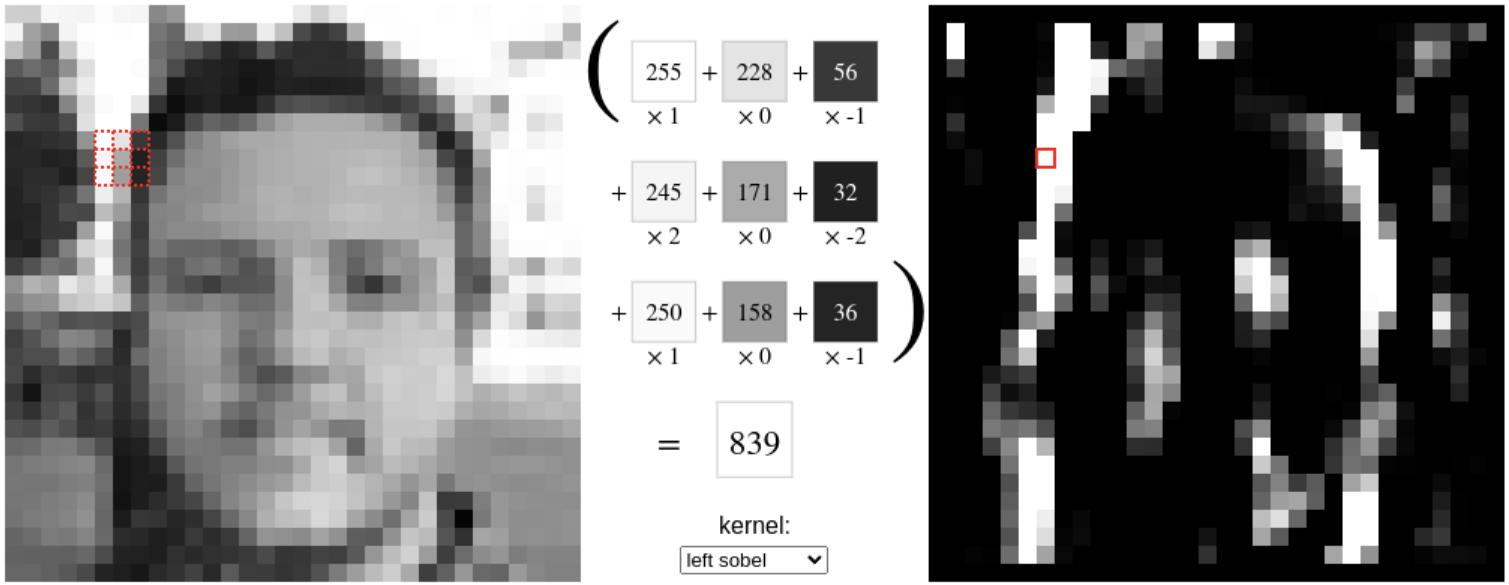
Visión computacional profunda



ferro@cimat.mx

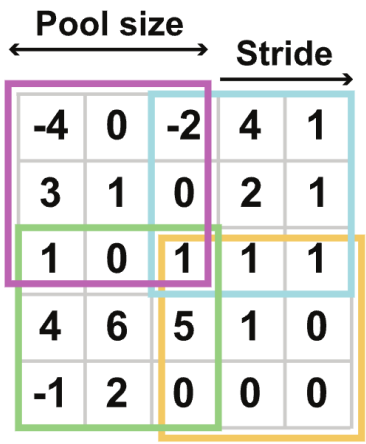
¿Qué es una convolución?

Visión computacional profunda



ferro@cimat.mx

Pooling



Features

Max Pooling

3	4
6	5

Min Pooling

-4	-2
-1	0

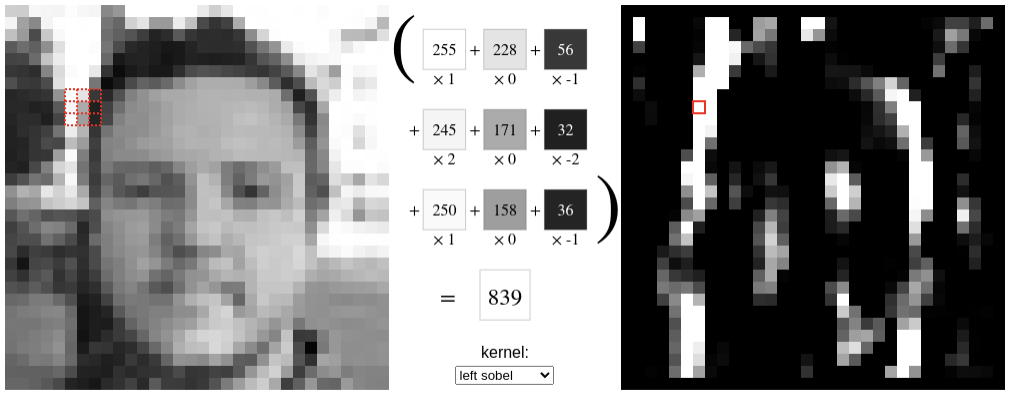
Average Pooling

0	1
2	1

Output

ferro@cimat.mx

Ejercicio: Convoluciones & Pooling



ferro@cimat.mx

Redes neuronales convolucionales

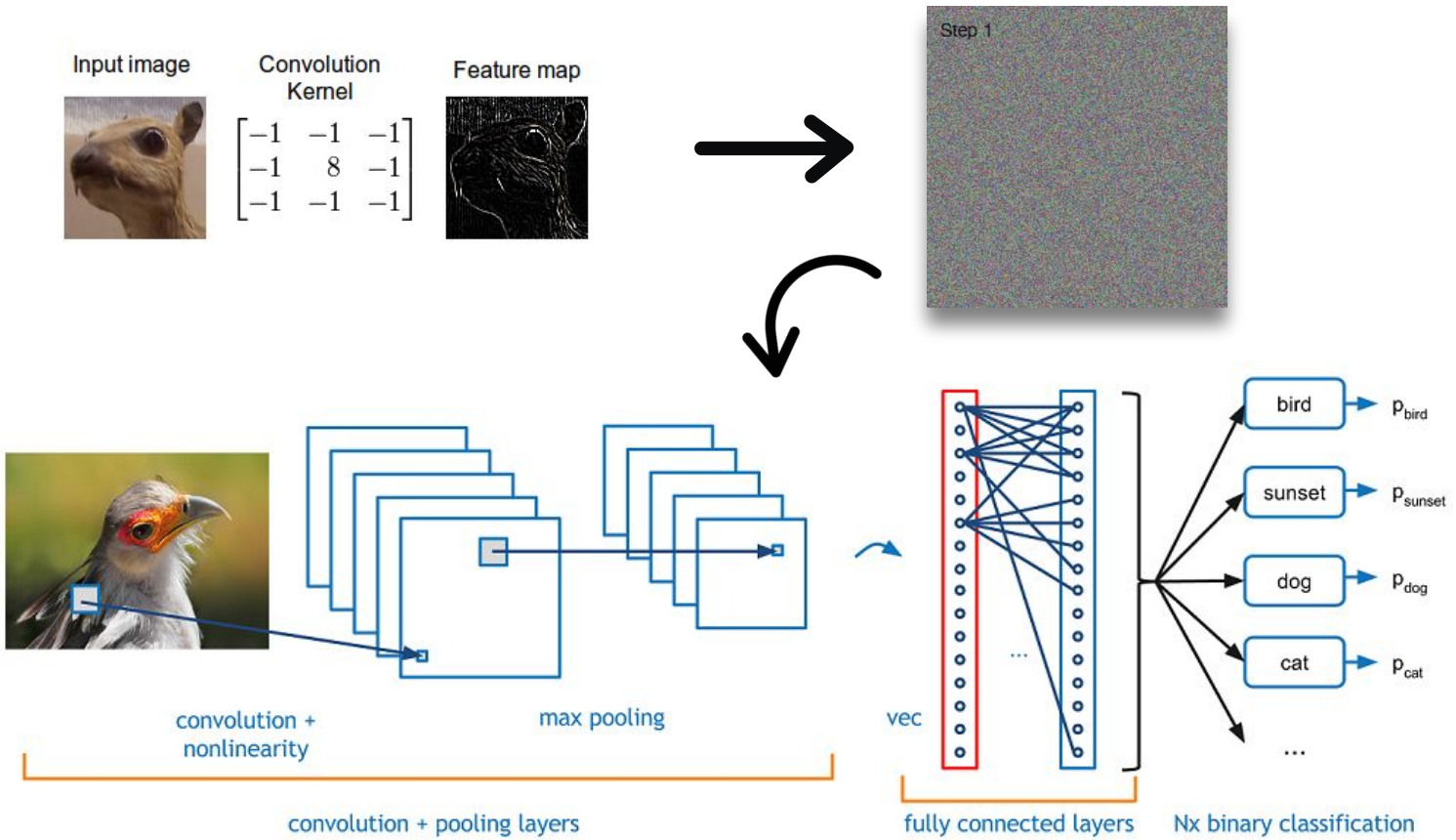
Visión computacional profunda

Figure: Yann LeCun

ferro@cimat.mx

Redes neuronales convolucionales

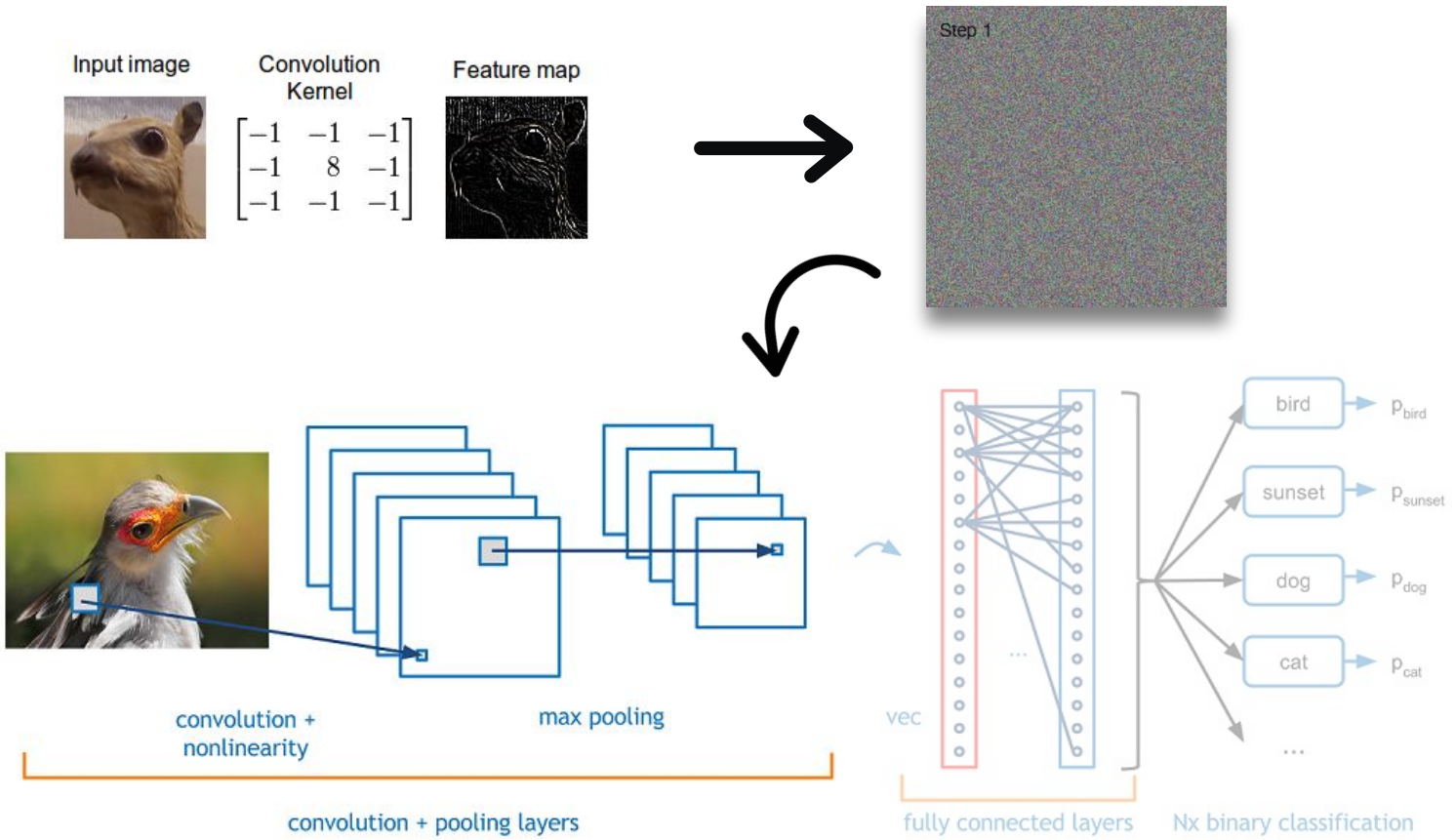
Visión computacional profunda



ferro@cimat.mx

Redes neuronales convolucionales

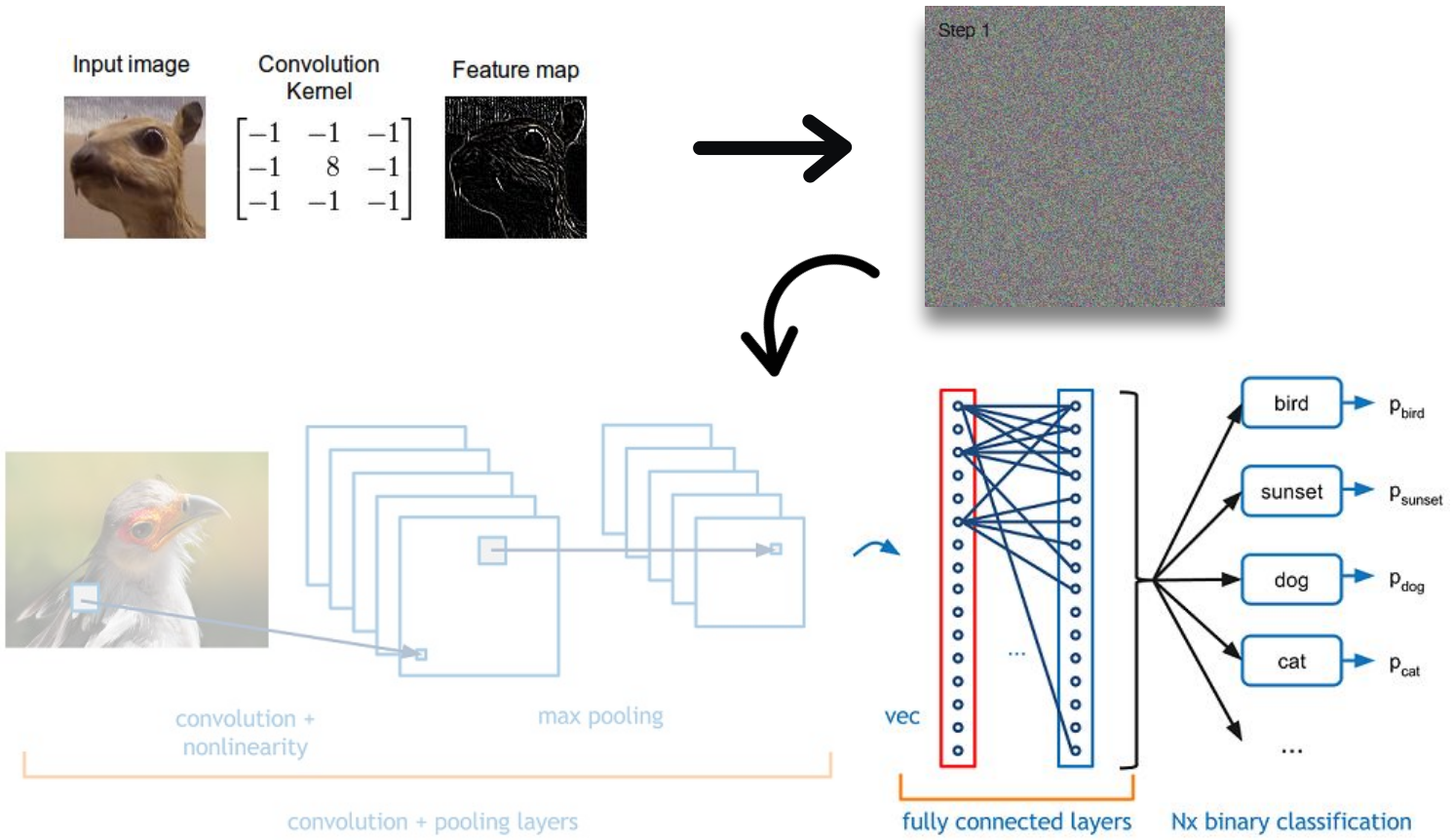
Visión computacional profunda



ferro@cimat.mx

Redes neuronales convolucionales

Visión computacional profunda



ferro@cimat.mx

LeNet-5

Visión computacional profunda

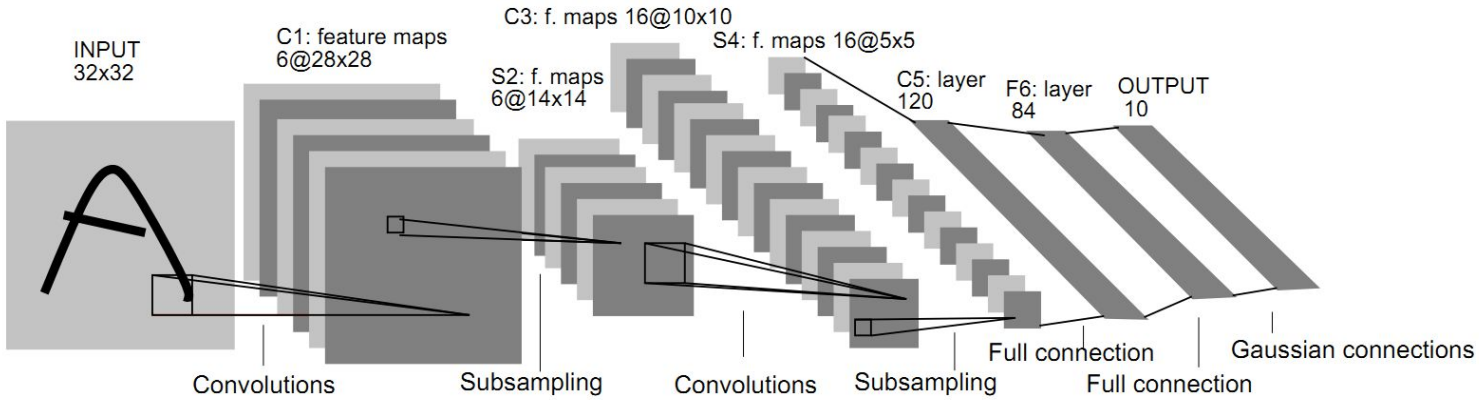


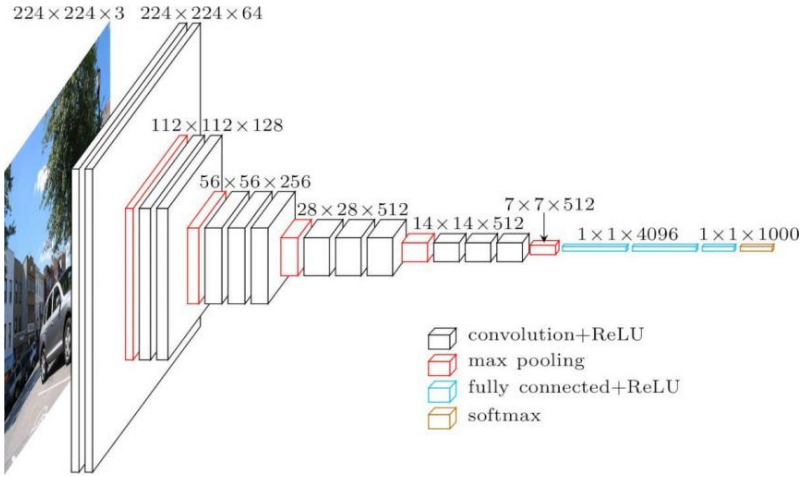
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

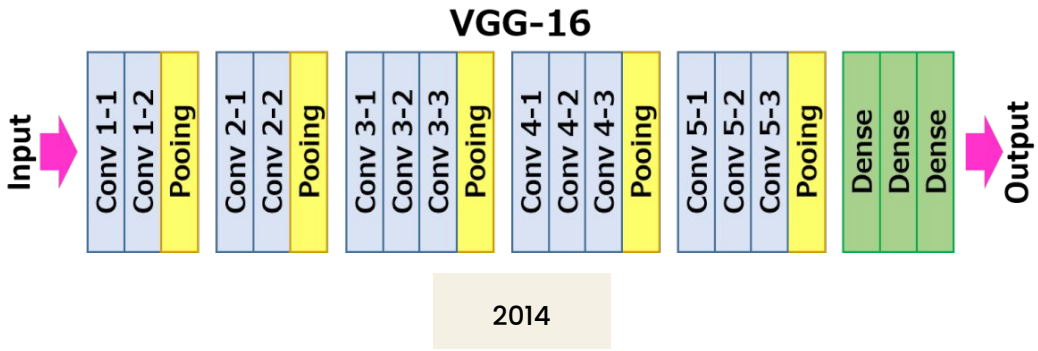
1998

ferro@cimat.mx

VGG16

Visión computacional profunda

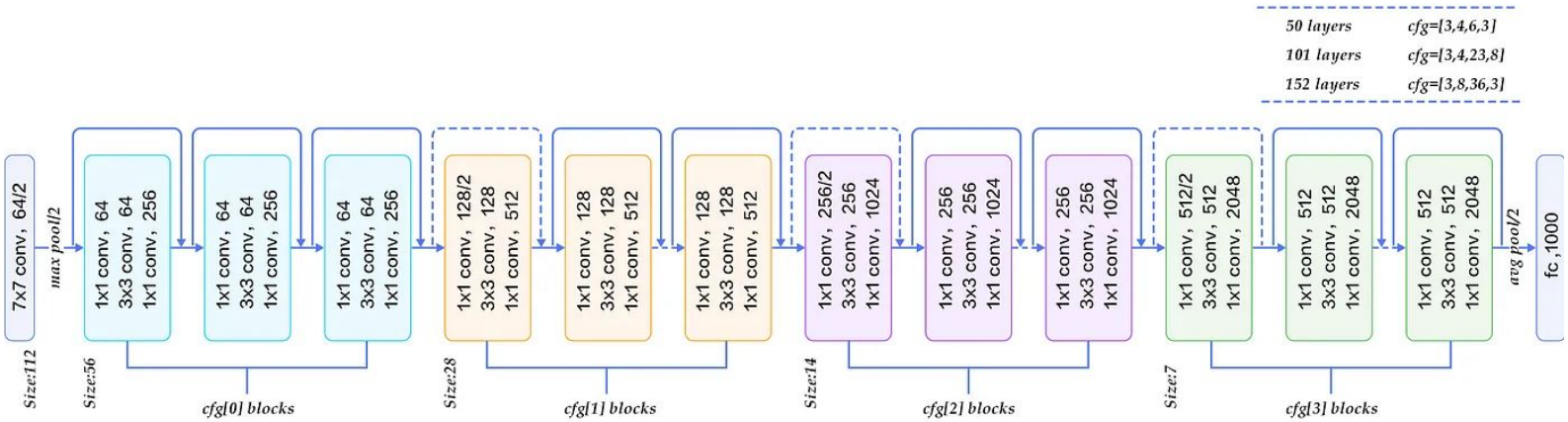




ferro@cimat.mx

Resnet50

Visión computacional profunda

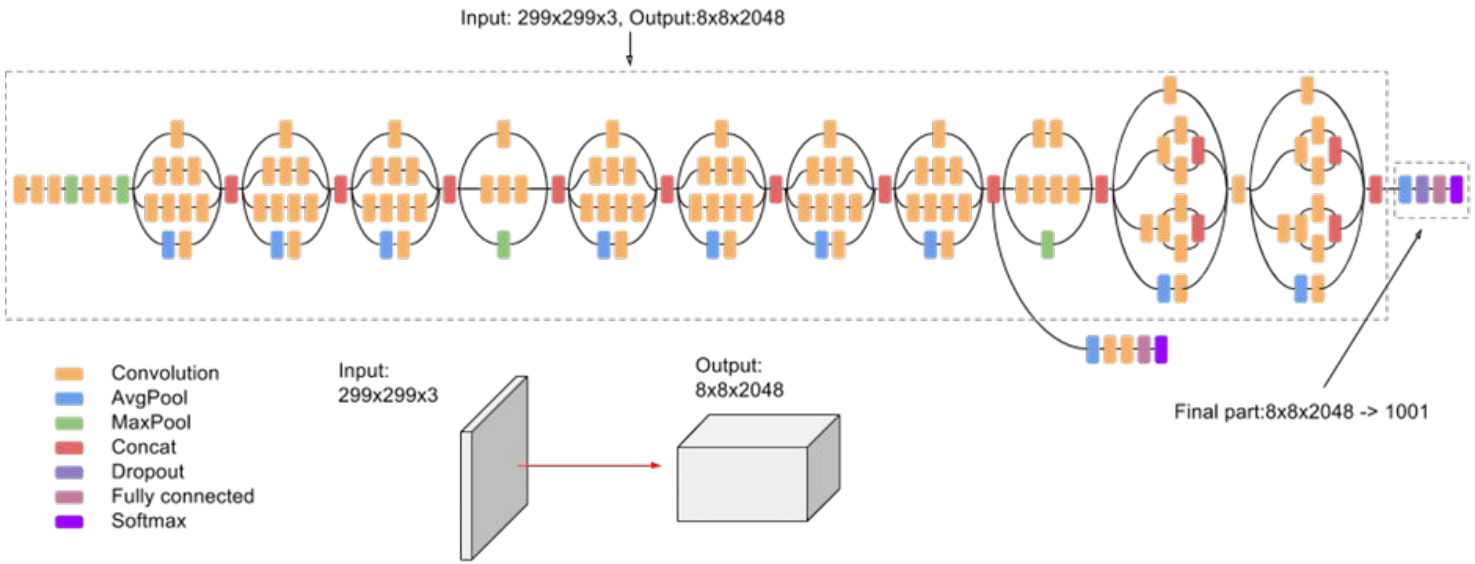


2015

ferro@cimat.mx

GoogLeNet

Visión computacional profunda



2015

ferro@cimat.mx

Ejercicio: Redes neuronales convolucionales

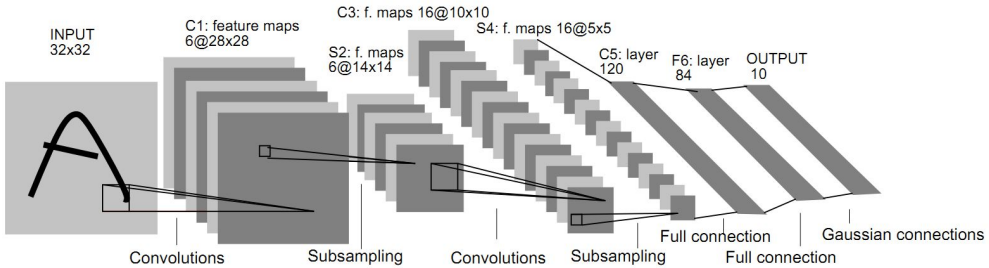


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

1998

ferro@cimat.mx

Lecturas recomendadas

- Tutorial 1: Image Filtering
- Image Kernels Explained Visually by Victor Powell
- Parameterized Pooling Layers by Hao Hao Tan
- TensorFlow Tutorials: Vision

ferro@cimat.mx

Reconstrucción de imágenes

- 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.

ferro@cimat.mx

Reconstrucción de imágenes

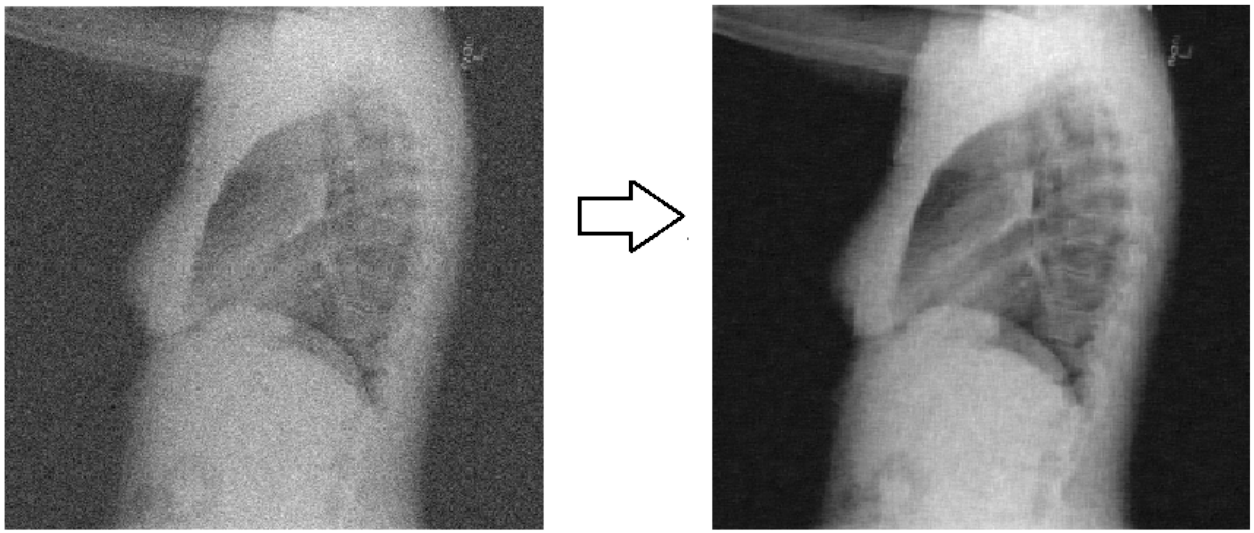
1. Restauración de la calidad visual.
2. Preservación de la información relevante.
3. Mejora de aplicaciones y análisis (eliminación de ruido).
4. Preservación de características y texturas.
5. Calidad y experiencia visual.

ferro@cimat.mx

Aplicaciones

Visión computacional profunda

Indiana University X-Ray Dataset



Noisy

Unsupervised Cleaning

ferro@cimat.mx

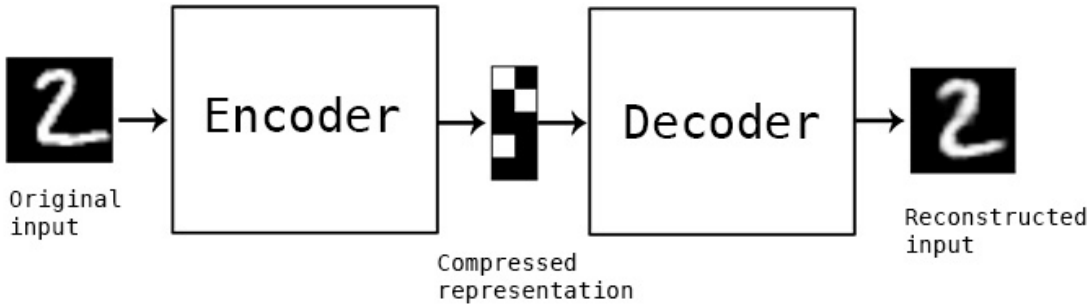
"Autoencoder For Denoising Images." Michel Kana.

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

Autoencoders

Visión computacional profunda

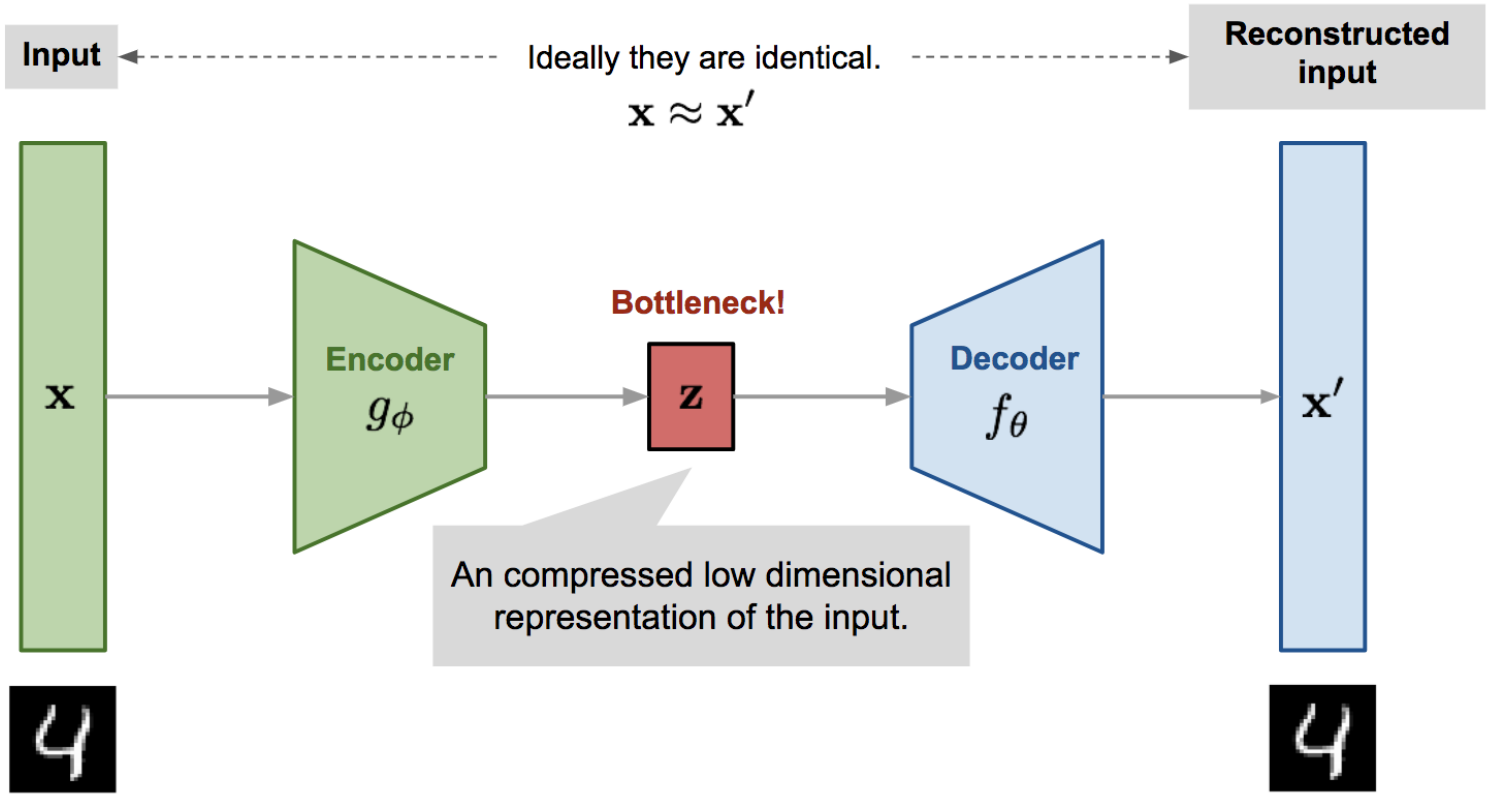
ANNs utilizadas para aprender representaciones eficientes de los datos de entrada sin necesidad de etiquetas o supervisión externa.



ferro@cimat.mx

Estructura de un autoencoder

Visión computacional profunda



ferro@cimat.mx

"From Autoencoder to Beta-VAE." Lilian Weng. https://lilianweng.github.io/posts/2018-08-12-vae/

Funcionamiento de autoencoders

- ***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.

ferro@cimat.mx

Funcionamiento de autoencoders

- ***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.

ferro@cimat.mx

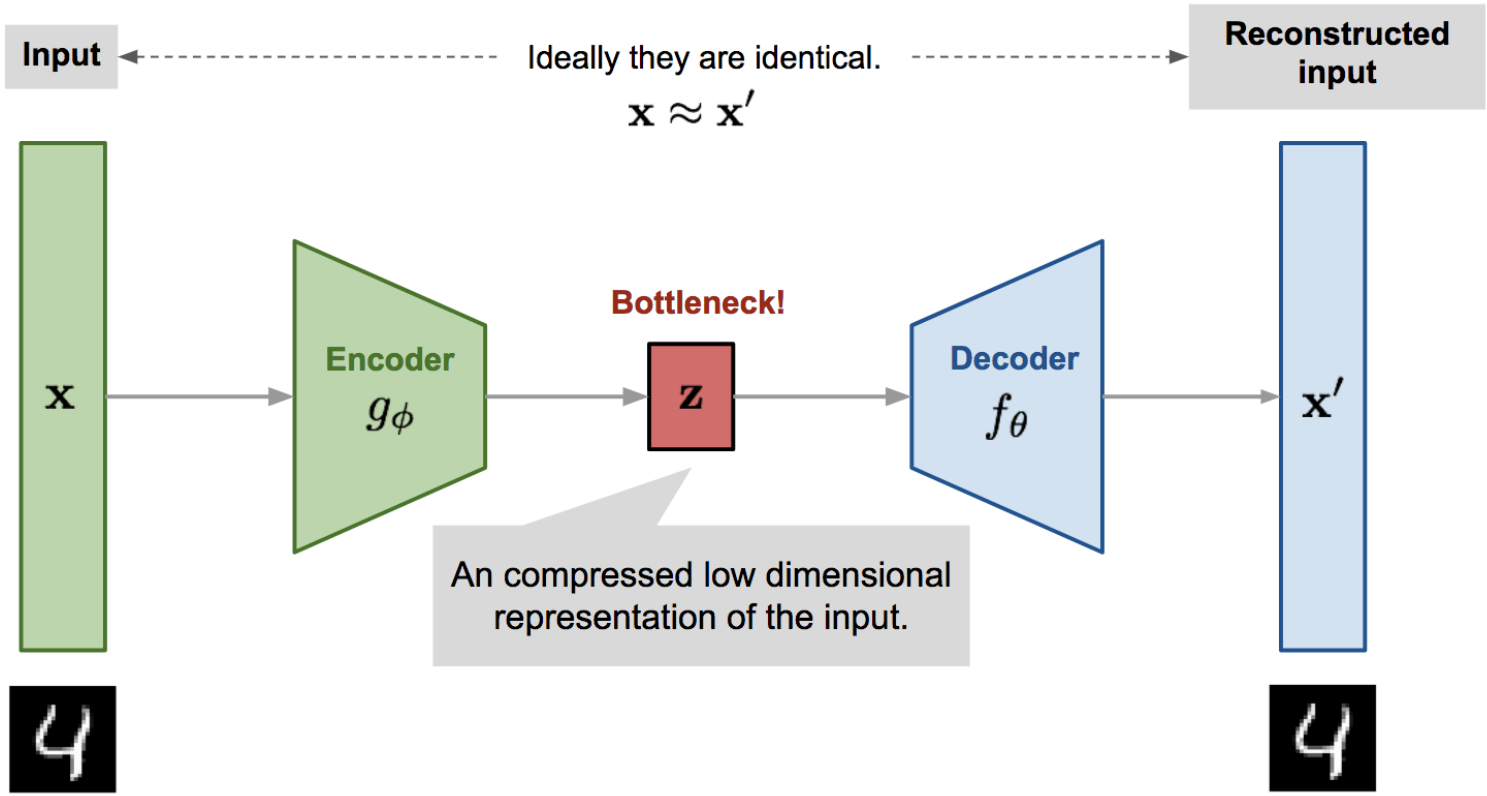
Funcionamiento de autoencoders

- 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.

ferro@cimat.mx

Estructura de un autoencoder

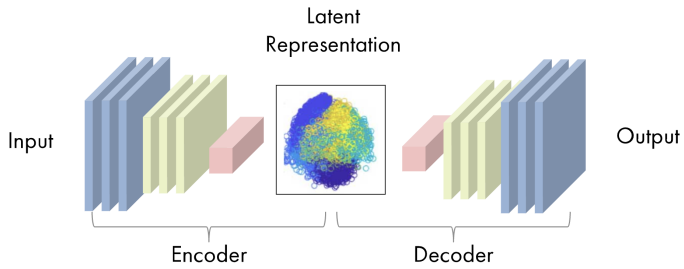
Visión computacional profunda



ferro@cimat.mx

"From Autoencoder to Beta-VAE." Lilian Weng. https://lilianweng.github.io/posts/2018-08-12-vae/

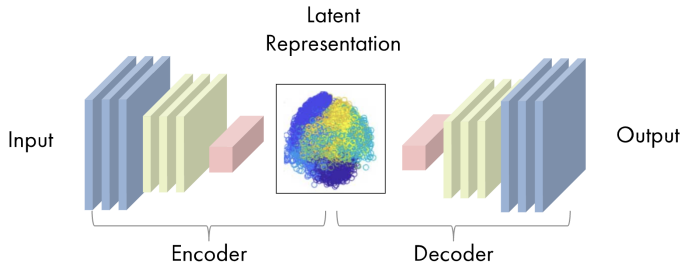
Aplicaciones de los autoencoders



- **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.

ferro@cimat.mx

Aplicaciones de los autoencoders



- **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.

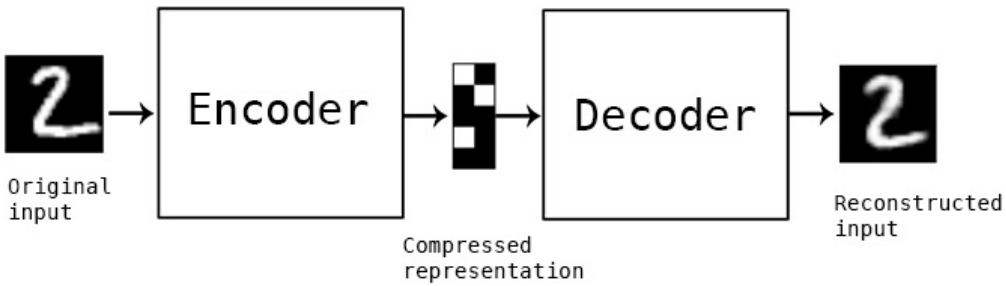
ferro@cimat.mx

Entrenamiento de autoencoders

- **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.

ferro@cimat.mx

Ejercicio: Autoencoder básico



ferro@cimat.mx

Limitantes

- **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.

ferro@cimat.mx

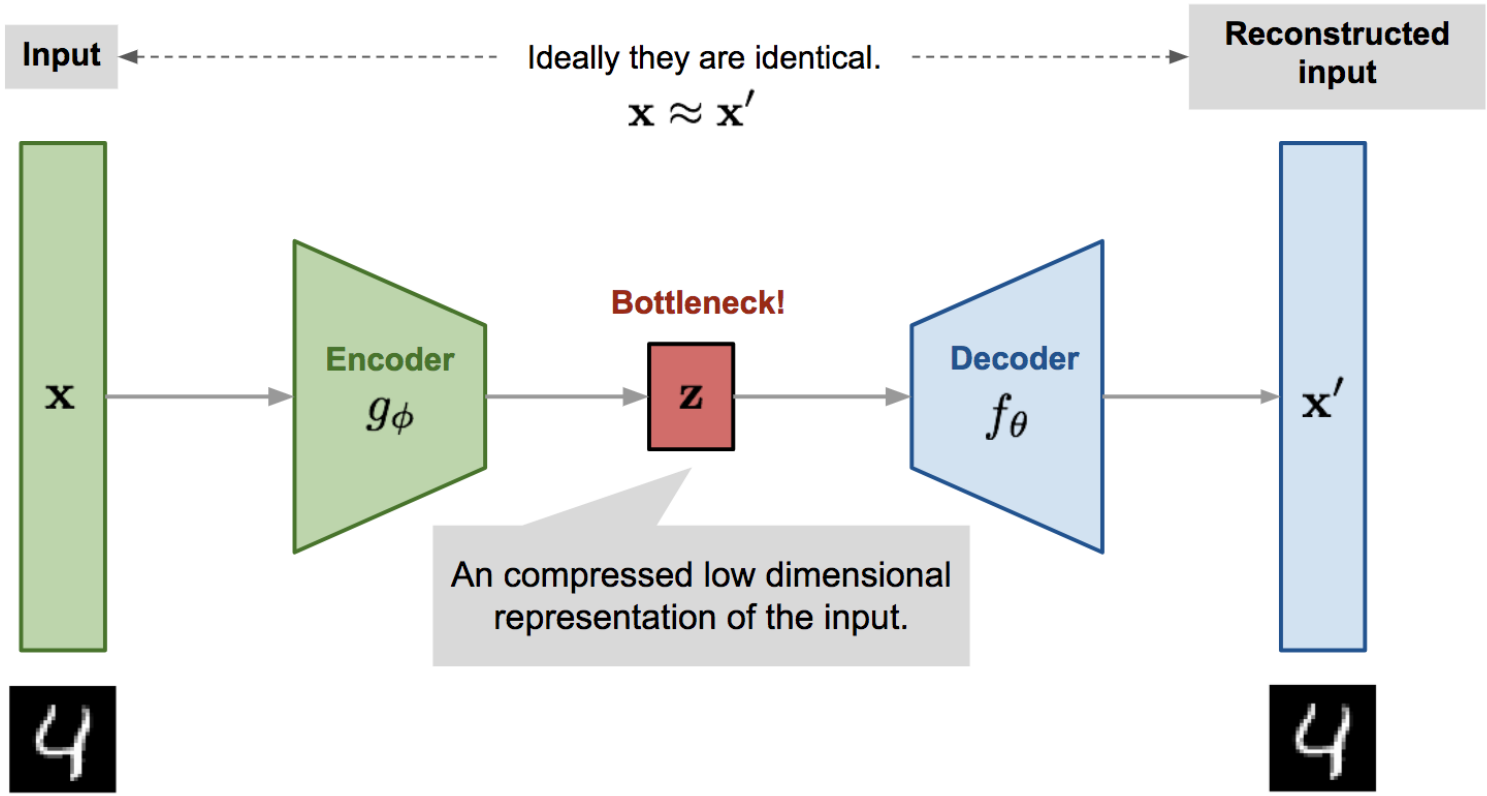
Limitantes

- **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

ferro@cimat.mx

Estructura de un autoencoder

Visión computacional profunda

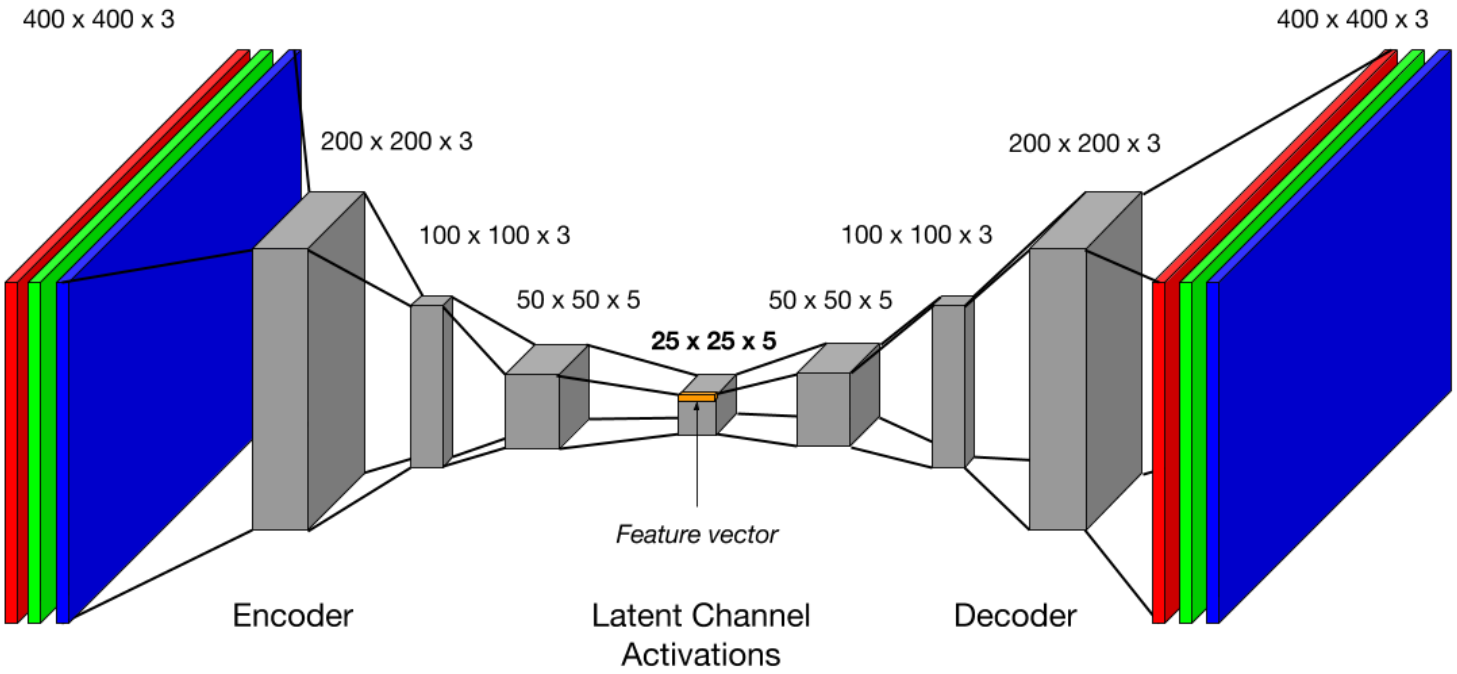


ferro@cimat.mx

"From Autoencoder to Beta-VAE." Lilian Weng. https://lilianweng.github.io/posts/2018-08-12-vae/

Estructura de un autoencoder

Visión computacional profunda

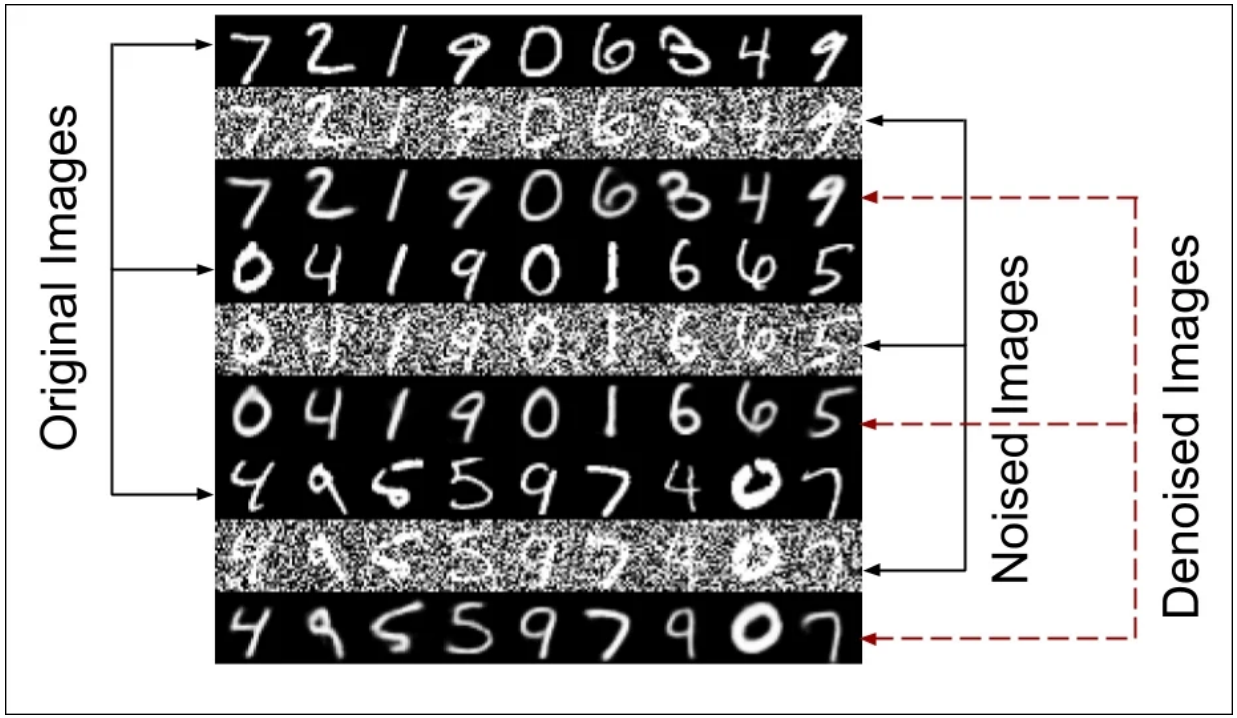


ferro@cimat.mx

"IROS 2017: Feature discovery and visualization of robot mission data using convolutional autoencoders and Bayesian nonparametric topic modeling." https://warp.whoi.edu/iros2017/

Denoising autoencoder (DAE/DCAE)

Visión computacional profunda

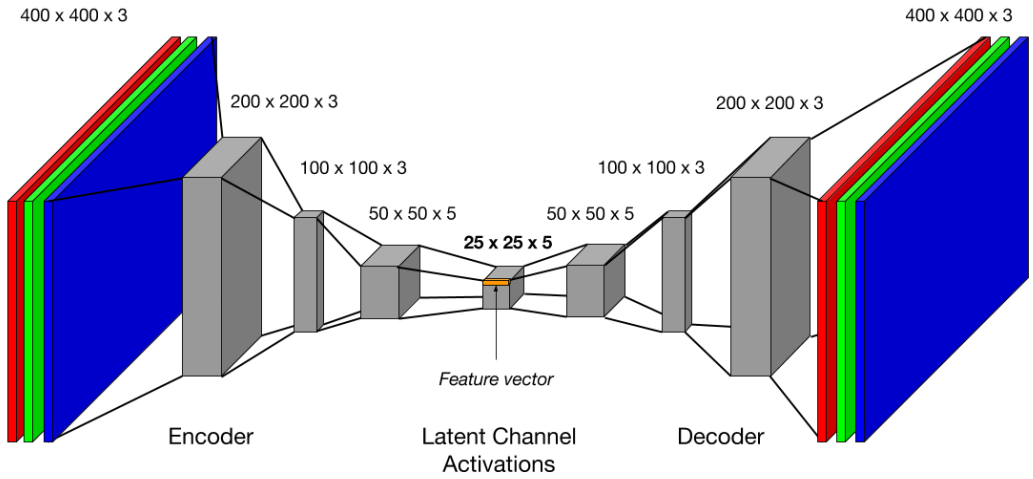


"Denoising autoencoder (DAE)."

https://subscription.packtpub.com/book/data/9781788629416/3/ch03lvl1sec19/denoising-autoencoder-dae

ferro@cimat.mx

Ejercicio: Autoencoder convolucional

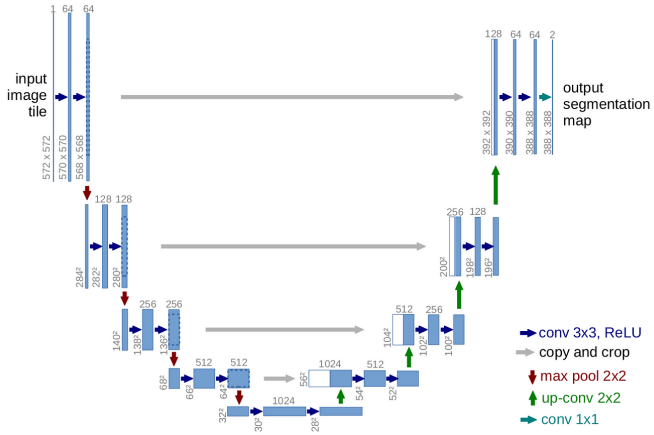


ferro@cimat.mx

Trabajos relacionados y avances recientes

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

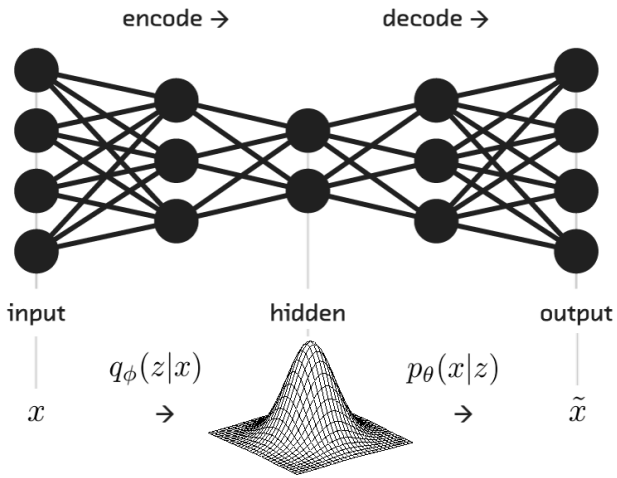
- **UNet:** Es ampliamente utilizada en el campo de la segmentación de imágenes, pero también se ha aplicado con éxito en tareas de denoising.



ferro@cimat.mx

Trabajos relacionados y avances recientes

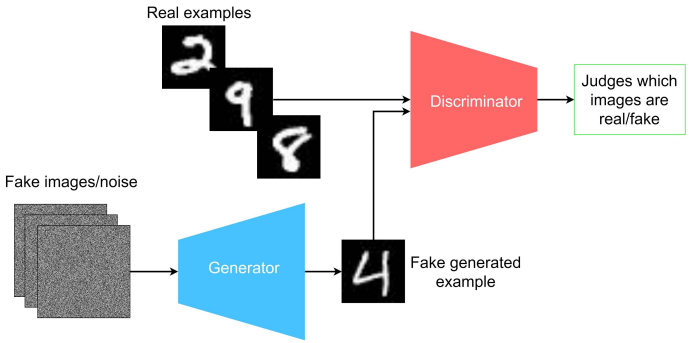
- **Variational Autoencoders (VAEs):** Los VAEs son una variante de los autoencoders que se utilizan para el aprendizaje de distribuciones latentes. Han demostrado ser efectivos en el denoising de imágenes al aprender representaciones latentes que siguen una distribución probabilística, lo que permite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



ferro@cimat.mx

Trabajos relacionados y avances recientes

- **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.



ferro@cimat.mx

Tareas en el campo de visión artificial

- **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 delinear 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.

ferro@cimat.mx

Tareas en el campo de visión artificial

- **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.

ferro@cimat.mx

Tareas en el campo de visión artificial

- **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.

ferro@cimat.mx

Lecturas recomendadas

- 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

ferro@cimat.mx

Segmentación de imágenes médicas

ferro@cimat.mx

Olaf

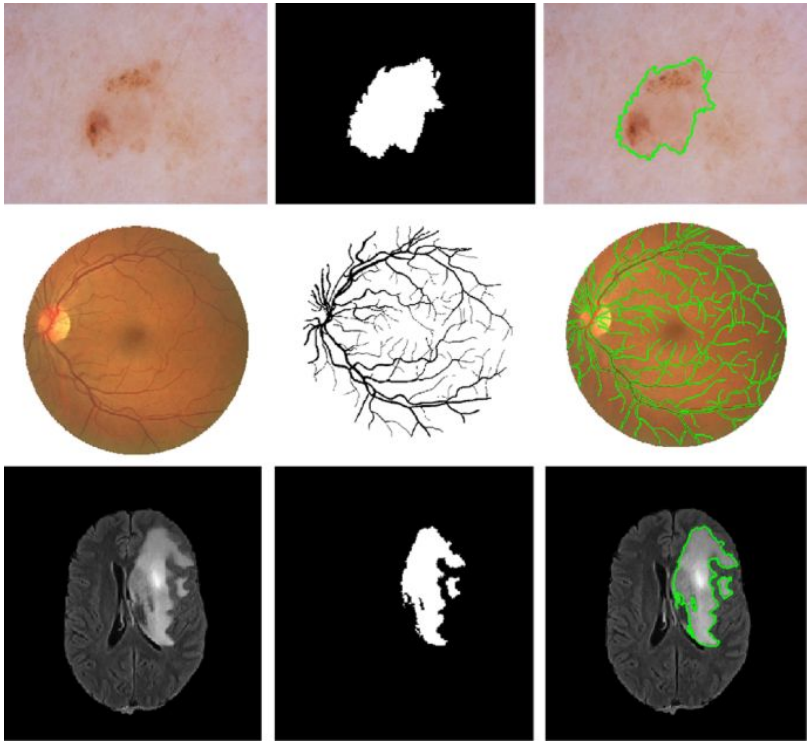
Ronneberger

et al.,

2015

ferro@cimat.mx

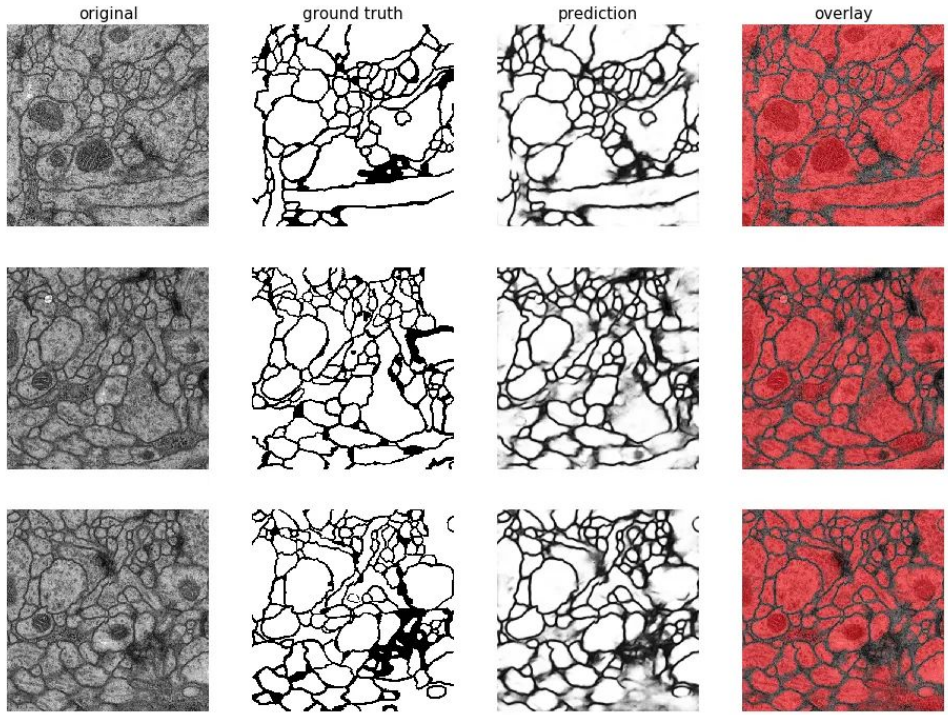
Segmentación de imágenes



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

ferro@cimat.mx

Segmentación con U-Net



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

ferro@cimat.mx

Segmentación con U-Net

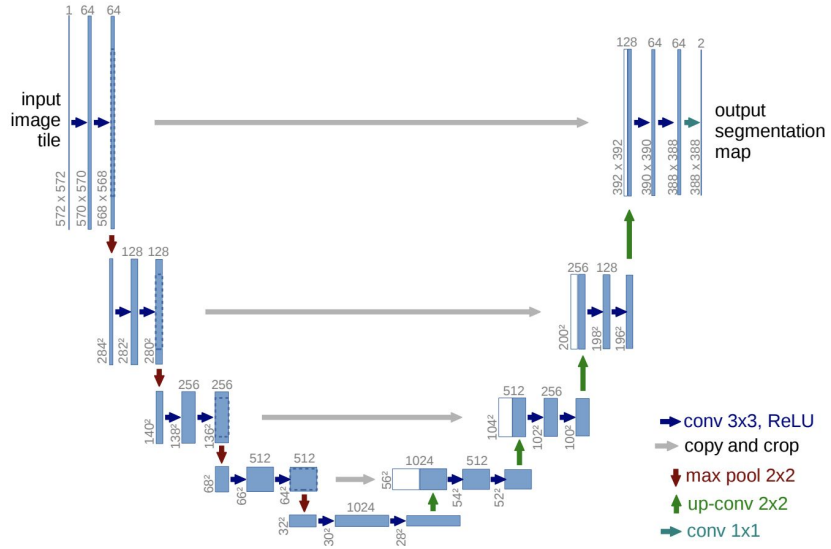
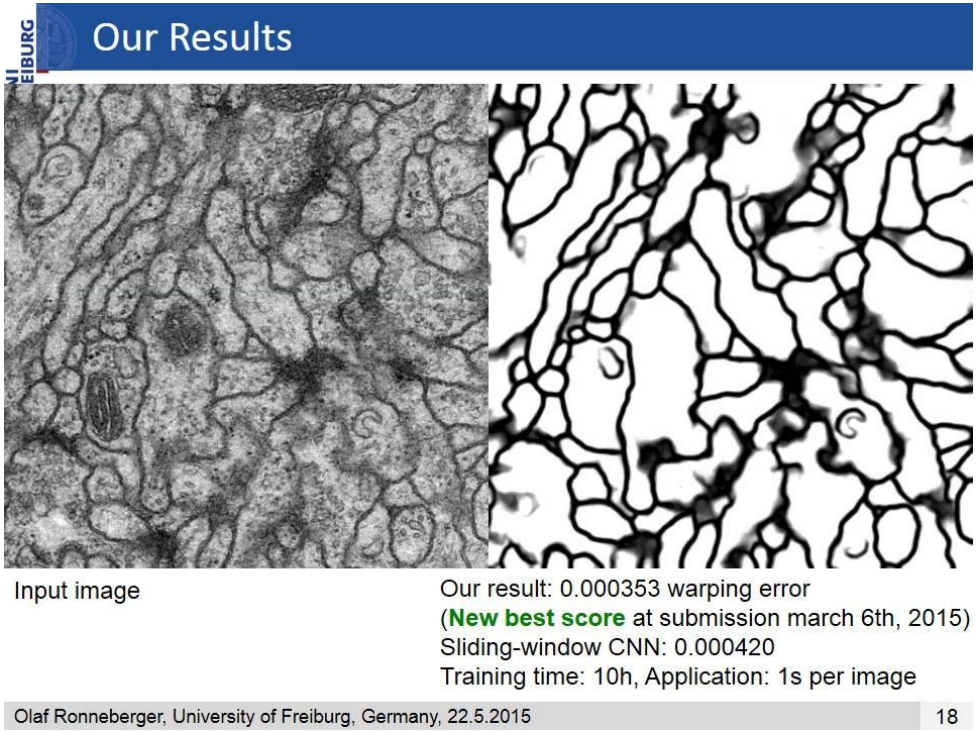


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows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

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

ferro@cimat.mx

Segmentación con U-Net



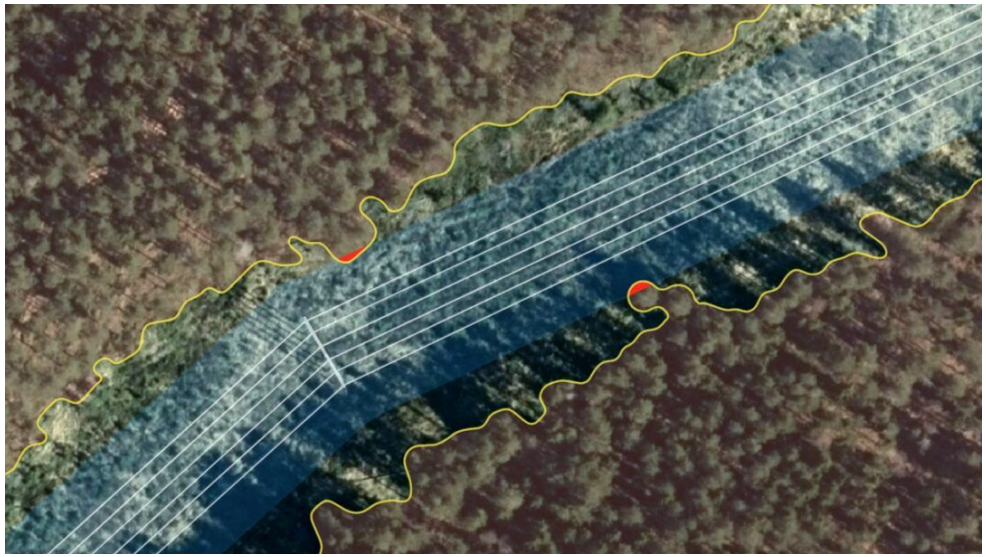
https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/

ferro@cimat.mx

Caso de estudio

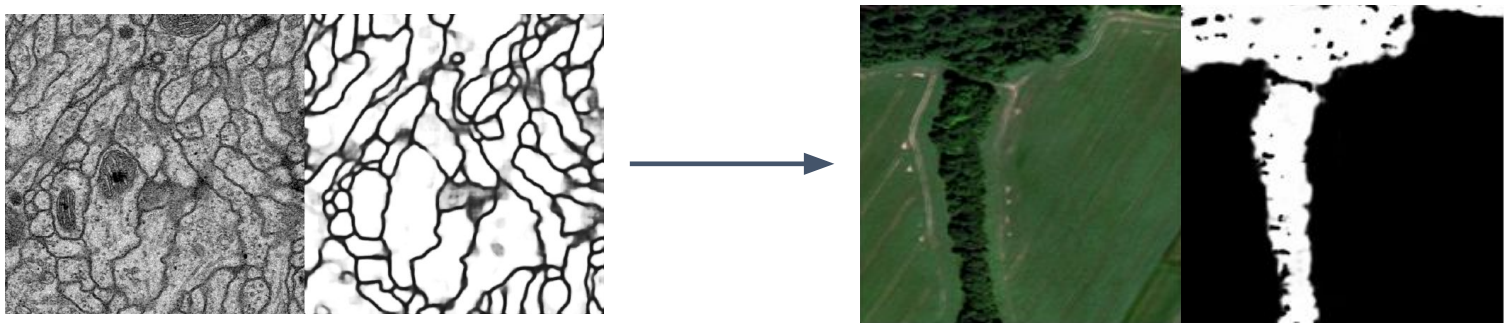
ferro@cimat.mx

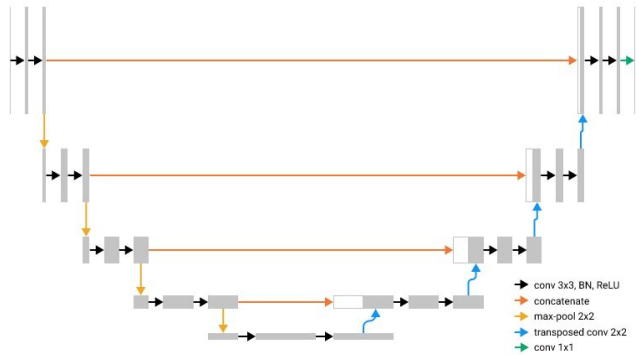
Prevención de incendios



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

ferro@cimat.mx





U-Net: Convolutional Networks for Biomedical Image Segmentation.

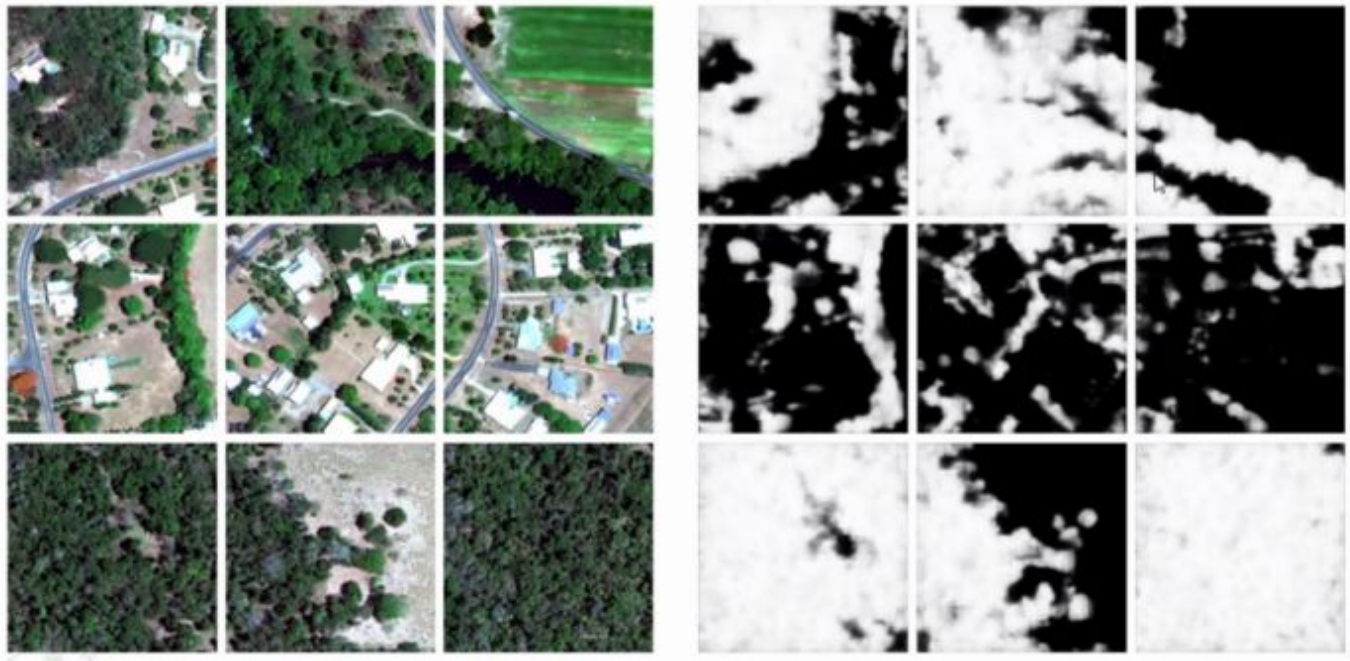
ferro@cimat.mx

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
- **¡Resolvimos el challenge!**

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

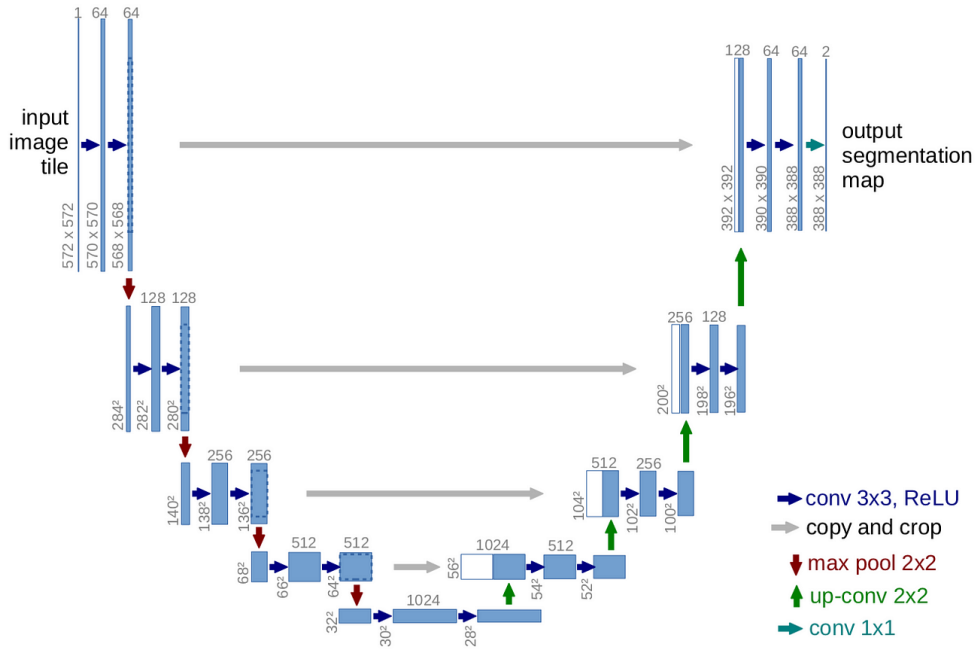
ferro@cimat.mx



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

ferro@cimat.mx

Ejercicio: Entrenamiento de U-Net



ferro@cimat.mx

Section 3: **Modelado profundo de secuencias**

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

ferro@cimat.mx

Section 4: **Modelado generativo profundo**

¿Qué son los Modelos Generativos?

- Un modelo generativo es un tipo de modelo que puede generar nuevos datos a partir de un conjunto de datos de entrenamiento.
- A diferencia de los modelos discriminativos, que se centran en predecir etiquetas a partir de características, los modelos generativos buscan aprender la distribución subyacente de los datos para poder generar nuevas muestras.
- ***En esencia...*** Los modelos generativos se enfocan en modelar cómo se distribuyen los datos de forma que puedan generar datos que "parecen" reales.

ferro@cimat.mx

¿Qué son los Modelos Generativos?

Modelado generativo profundo

ferro@cimat.mx

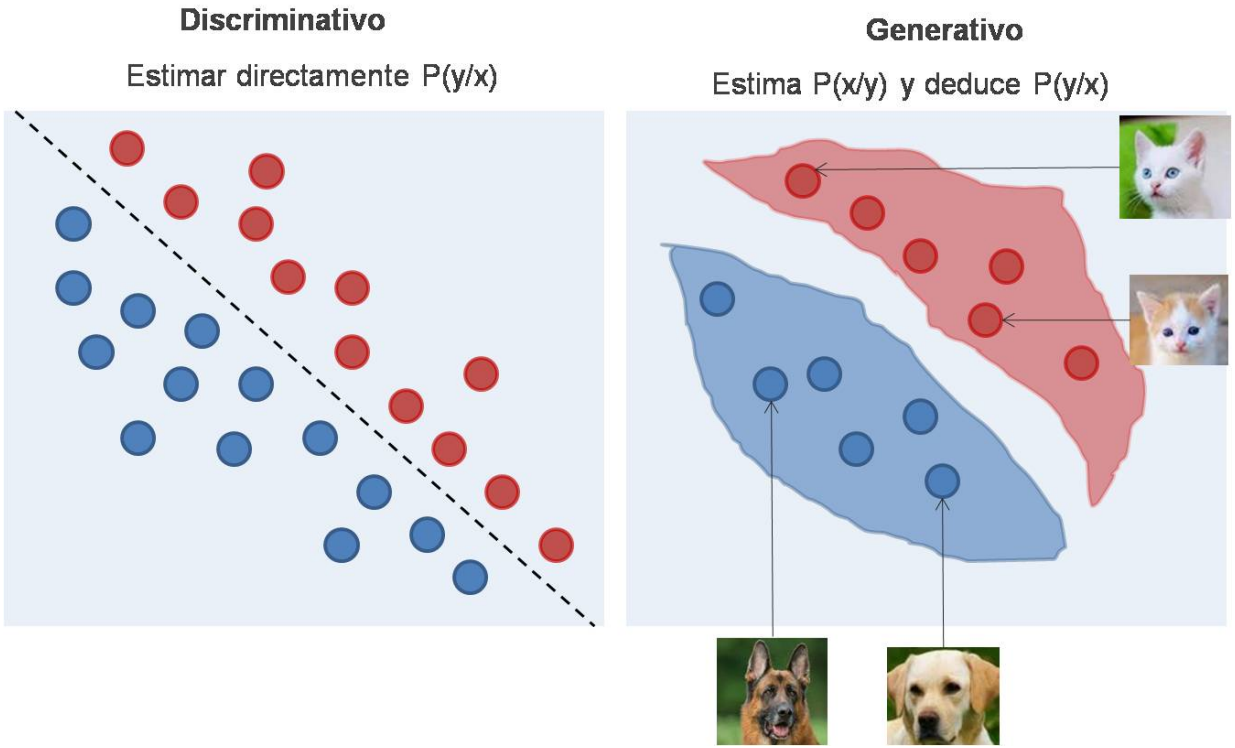
Modelos generativos vs. discriminativos

- **Modelos discriminativos:**
  - Se enfocan en la predicción de etiquetas.
  - Ejemplos: Regresión logística, SVM, redes neuronales feedforward.
- **Modelos generativos:**
  - Modelan la distribución conjunta de los datos.
  - Ejemplos: Redes Bayesianas, GANs, VAEs.
- ***En esencia...*** Los modelos discriminativos responden a preguntas como "¿A qué clase pertenece esta muestra?", mientras que los generativos responden a "¿Cómo se ve una muestra típica de esta clase?".

ferro@cimat.mx

¿Qué son los Modelos Generativos?

Modelado generativo profundo



ferro@cimat.mx

Aplicaciones de modelos generativos

- **Imágenes:**
  - Generación de rostros humanos (e.g., StyleGAN).
  - Generación de arte abstracto.
- **Texto:**
  - Composición automática de párrafos o historias.
  - Traducción automática y generación de poesía.
- **Música:**
  - Composición de melodías.
  - Generación de pistas basadas en un estilo dado.
- **Videojuegos:**
  - Generación de mundos y personajes.
  - Creación de niveles procedurales.

ferro@cimat.mx

EJEMPLOS

Pregunta: ¿cómo están revolucionando estas industrias? ¿cuáles son las implicaciones éticas y creativas de estas aplicaciones?

ferro@cimat.mx

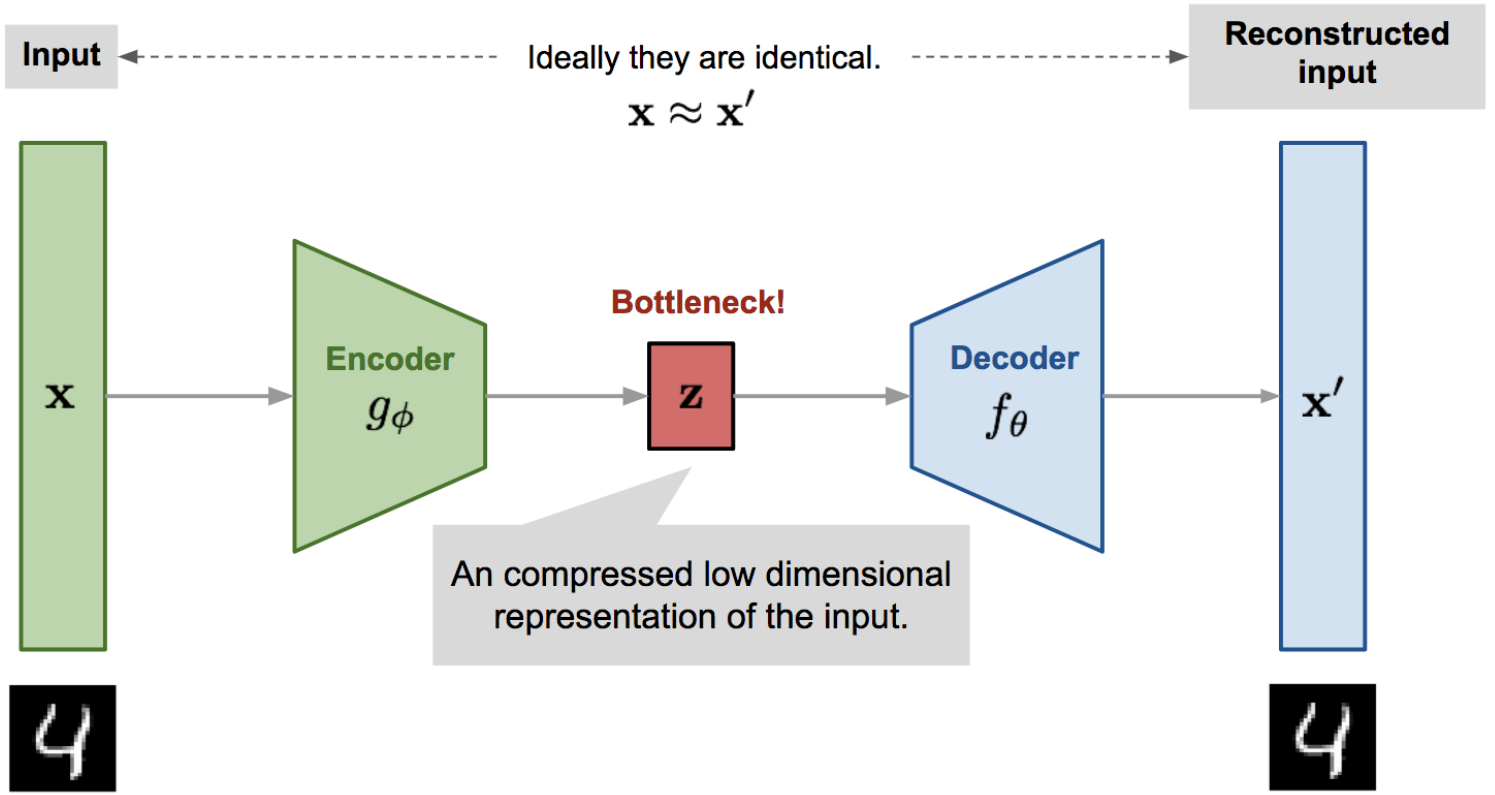
Autoencoders (AEs)

- Un autoencoder es una red neuronal diseñada para aprender una representación comprimida (codificación) de los datos de entrada.
- **Codificador (Encoder):** Reduce la dimensionalidad de los datos.
- **Decodificador (Decoder):** Reconstruye los datos originales a partir de la representación comprimida.
- ***En esencia...*** La idea principal detrás de los autoencoders es aprender a copiar la entrada a la salida, pero a través de una representación más pequeña (la representación latente), lo que obliga al modelo a aprender características esenciales de los datos.

ferro@cimat.mx

Estructura de un autoencoder

Modelado generativo profundo



ferro@cimat.mx

"From Autoencoder to Beta-VAE." Lilian Weng. https://lilianweng.github.io/posts/2018-08-12-vae/

Autoencoders

- **Reducción de dimensionalidad:** Los autoencoders pueden utilizarse para reducir la dimensionalidad de los datos, similar al Análisis de Componentes Principales (PCA).
- **Detección de anomalías:** Las reconstrucciones defectuosas pueden indicar datos anómalos o inusuales.
- **Generación de datos sintéticos:** Generar nuevas muestras modificando la representación latente.

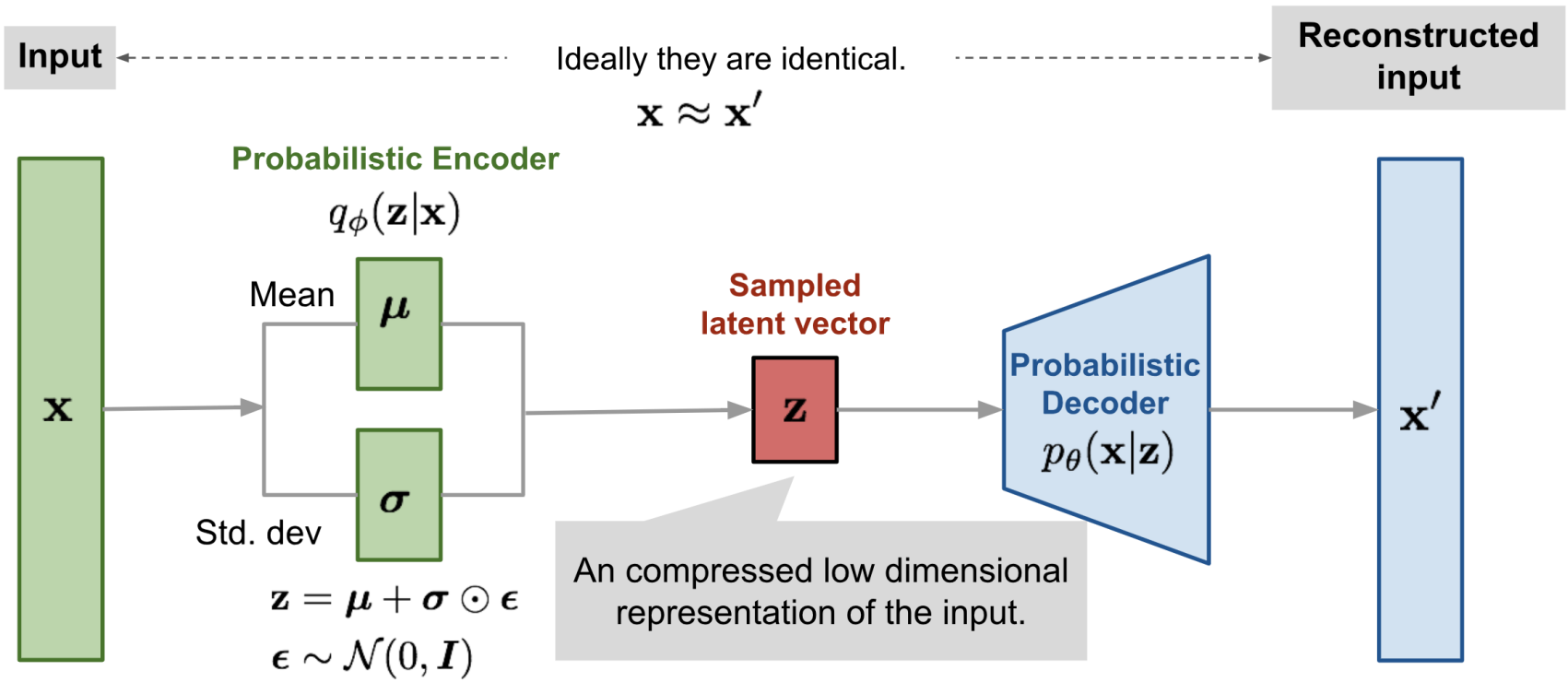
ferro@cimat.mx

Variational Autoencoders (VAEs)

- Un VAE es una versión probabilística de un autoencoder que genera nuevas muestras al aprender la distribución subyacente de los datos.
- Las principales diferencias con un *AE*:
  - **Codificación estocástica:** Los VAEs aprenden a mapear datos a una distribución (e.g., normal) en lugar de un punto fijo.
  - **Regularización:** Introduce una pérdida de regularización para garantizar que la distribución latente sea continua.
- ***En esencia...*** Los VAEs no sólo intentan reconstruir los datos, sino que también buscan aprender una estructura probabilística que permita generar nuevas muestras plausibles al azar.

ferro@cimat.mx

Estructura de un variational autoencoder



"From Autoencoder to Beta-VAE." Lilian Weng. https://lilianweng.github.io/posts/2018-08-12-vae/

Observaciones (VAEs)

- En lugar de aprender un vector de características latentes fijo para cada entrada, un VAE aprende los parámetros (media y varianza) de una distribución que representa posibles codificaciones latentes para esa entrada.
- A partir de estos parámetros, se toma una muestra para obtener la representación latente que se pasará al decodificador (decoder). Este proceso de muestreo introduce un componente estocástico (aleatorio), que permite que un mismo dato de entrada pueda ser codificado en múltiples puntos del espacio latente, lo que facilita la generación de nuevas muestras similares pero no idénticas.

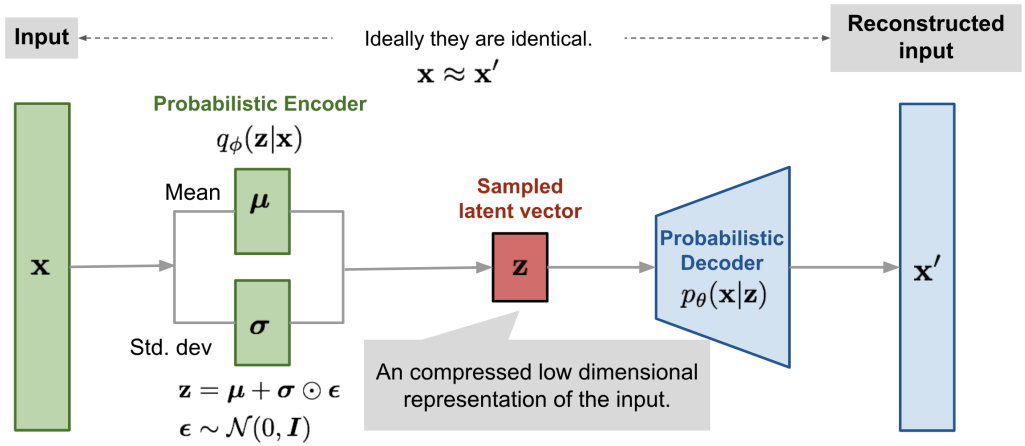
ferro@cimat.mx

Variational Autoencoders (VAEs)

- **Generación de nuevos datos:** Los VAEs pueden generar nuevas muestras variando las entradas en la distribución latente.
- **Interpolación y Manipulación de Latentes:** Permite crear transiciones suaves entre diferentes muestras (e.g., morfing de imágenes).
- ***En esencia...*** Los VAEs son potentes para generar datos nuevos y realistas, especialmente en aplicaciones como la generación de rostros o la creación de contenido sintético.

ferro@cimat.mx

Ejercicio: Tutoriales de TensorFlow



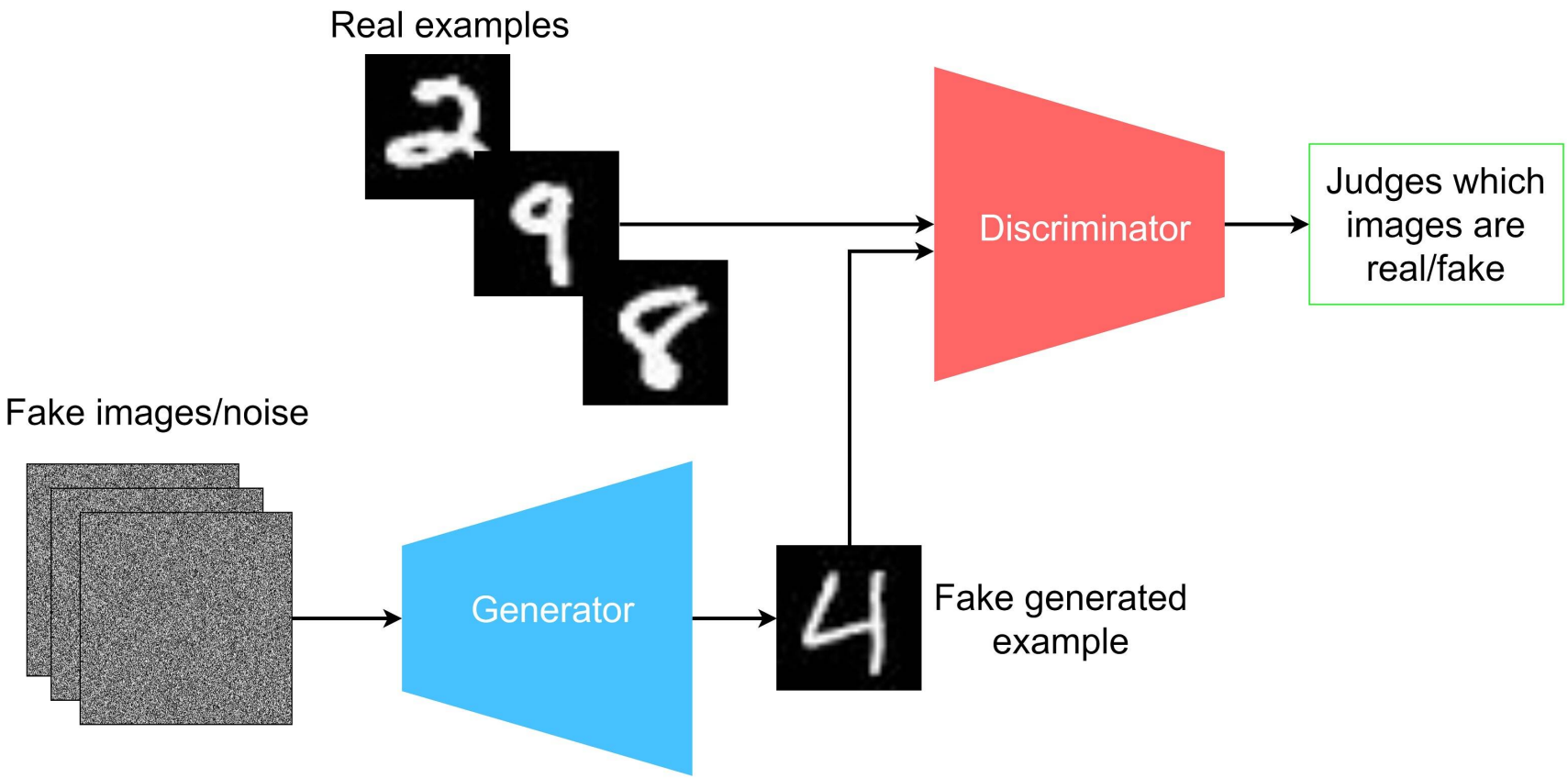
ferro@cimat.mx

Redes Generativas Antagónicas/Adversariales (GANs)

- Una GAN es una clase de redes neuronales en la que dos redes (un generador y un discriminador) compiten en un juego de suma cero.
- **Componentes principales:**
  - **Generador (G):** Crea datos falsos que intentan parecer reales.
  - **Discriminador (D):** Intenta distinguir entre datos reales y datos generados por el generador.
- ***En esencia...*** El generador y el discriminador se entrenan simultáneamente en un proceso adversarial. El generador intenta mejorar sus datos generados para engañar al discriminador, mientras que el discriminador intenta mejorar su capacidad para distinguir entre datos reales y generados.

ferro@cimat.mx

Estructura de una GAN



ferro@cimat.mx

Funcionamiento de GANs)

- **Entrenamiento:**
  - **Generación de datos:** El generador crea una muestra a partir de ruido.
  - **Evaluación del discriminador:** El discriminador evalúa si la muestra es real o falsa.
  - **Retroalimentación:** El generador y el discriminador actualizan sus pesos en función de la retroalimentación recibida.
- **Objetivo:**
  - **Generador:** Minimizar la probabilidad de que el discriminador identifique las muestras como falsas.
  - **Discriminador:** Maximizar la precisión al clasificar muestras como reales o falsas.

ferro@cimat.mx

Variantes de GANs)

- **Deep Convolutional GANs (DCGANs):** Utilizan capas convolucionales en lugar de capas completamente conectadas para mejorar la calidad de las imágenes generadas. **Aplicaciones:** Generación de imágenes de alta resolución y texturas más realistas.
- **Conditional GANs (cGANs):** Incorporan información adicional (etiquetas) en el proceso de generación para controlar el tipo de datos generados. **Aplicaciones:** Generación de imágenes en categorías específicas, como rostros con expresiones particulares o escenas con características predefinidas.
- **CycleGAN:** Permite la conversión de imágenes entre dos dominios diferentes sin necesidad de pares de imágenes coincidentes, utilizando ciclos de consistencia para asegurar que la conversión sea realista en ambas direcciones. **Aplicaciones:** Transferencia de estilo entre fotos y pinturas, traducción de imágenes entre estaciones o entre distintos tipos de objetos.

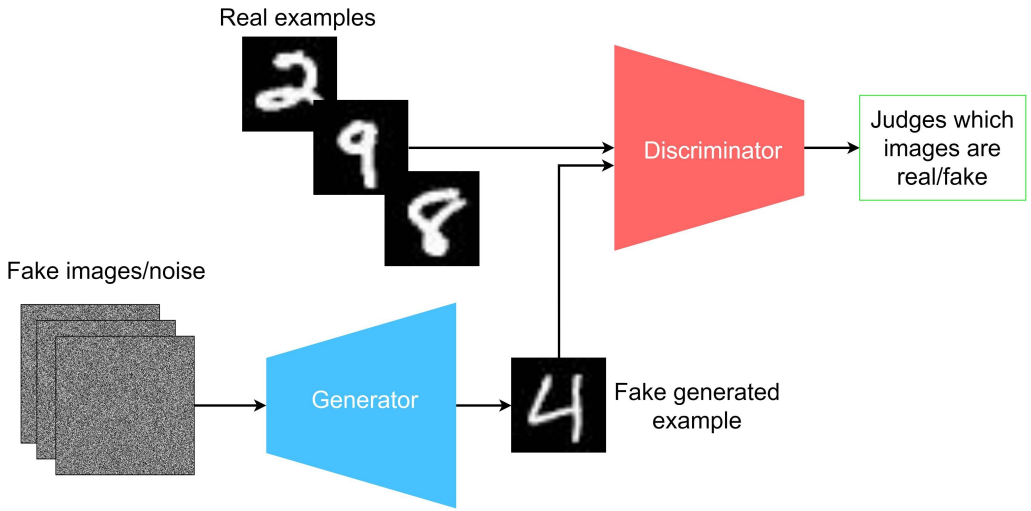
ferro@cimat.mx

Aplicaciones de las GANs)

- *Generación de imágenes realistas:* Creación de imágenes fotorrealistas a partir de ruido aleatorio.
- *Deepfakes:* Sustitución de rostros en videos con gran realismo.
- *Estilo de transferencia:* Aplicación de estilos artísticos a imágenes.
- *Aumento de datos:* Generación de datos sintéticos para mejorar los conjuntos de datos en problemas de aprendizaje automático.

ferro@cimat.mx

Ejercicio: Tutoriales de TensorFlow



ferro@cimat.mx

Lecturas recomendadas

- From Autoencoder to Beta-VAE
- Variational AutoEncoder
- Variational Autoencoder in TensorFlow
- What are Diffusion Models?
- Generative adversarial networks explained

ferro@cimat.mx

Section 5: **Panorama actual y futuro**

Evolución de la IA

- **1950s - 1970s: Nacimiento de la IA**
  - **1943: Primeras Ideas Inspiradas en el Cerebro.**
    Warren McCulloch y Walter Pitts proponen el primer modelo computacional de una red neuronal, sentando las bases teóricas para la IA inspirada en el funcionamiento del cerebro humano.
  - **1950: Alan Turing y el Test de Turing.**
    Alan Turing publica "Computing Machinery and Intelligence", donde introduce el concepto de "máquinas pensantes" y el Test de Turing para evaluar la inteligencia de las máquinas.
  - **1956: La Conferencia de Dartmouth.**
    Considerada el nacimiento oficial de la IA como campo académico, John McCarthy, Marvin Minsky, Nathaniel Rochester, y Claude Shannon organizan una conferencia donde acuñan el término "Inteligencia Artificial".
  - **1960s: Primeros Sistemas de IA.**
    Desarrollo de programas pioneros como ELIZA (un bot conversacional) y SHAKEY (el primer robot móvil con capacidad para razonar sobre sus acciones).

ferro@cimat.mx

Evolución de la IA

Panorama actual y futuro

- **1980s - 1990s: Primer Invierno de la IA y Resurgimiento**
  - **1970s - 1980s: Invierno de la IA.**
    Desilusión y retroceso: Los resultados limitados y el alto costo computacional llevan a una disminución en la financiación y el interés por la IA, un período conocido como el "Invierno de la IA".
  - **1980s: Resurgimiento con *Expert Systems*.**
    Sistemas expertos: Aumento del interés en sistemas basados en reglas que emulan la toma de decisiones humana en dominios específicos, como MYCIN en medicina.
  - **Finales de 1980s: Introducción del aprendizaje profundo**
    Backpropagation (Retropropagación): Se populariza como un método eficiente para entrenar redes neuronales multicapa, marcando un hito en la IA.

ferro@cimat.mx

Evolución de la IA

- **1990s - 2010s: Expansión y dominio del Deep Learning**
  - **1997: Deep Blue vence a Garry Kasparov.**
    Deep Blue de IBM derrota al campeón mundial de ajedrez Garry Kasparov, demostrando la capacidad de las máquinas para superar a humanos en tareas complejas.
  - **2000s: Auge de los datos y el poder computacional.**
    Big Data: La proliferación de datos digitales y la mejora en el poder computacional permiten entrenar modelos más grandes y complejos.
    Internet y Cloud Computing: La IA se beneficia de la creciente disponibilidad de datos en línea y el acceso a recursos computacionales en la nube.
  - **2012: AlexNet y la revolución del Deep Learning.**
    AlexNet gana el concurso ImageNet con una red neuronal convolucional profunda, marcando el inicio de la revolución del deep learning en visión por computadora.
  - **2014: Generative Adversarial Networks (GANs).**
    Ian Goodfellow introduce las GANs, que permiten generar datos sintéticos realistas, revolucionando áreas como la creación de imágenes.

ferro@cimat.mx

- **2010s - Presente: Transformers y la nueva era de la IA**
  - **2017: Introducción de Transformers.**
    Vaswani et al. presentan el Transformer en el artículo "Attention is All You Need", revolucionando el procesamiento de lenguaje natural y superando a los RNNs en muchas tareas.
  - **2018: BERT y el avance en NLP.**
    Big Data: BERT (Bidirectional Encoder Representations from Transformers) redefine el estado del arte en NLP con un enfoque bidireccional, mejorando la comprensión contextual.
  - **2020s: Modelos de Lenguaje de Gran Escala (LLMs).**
    GPT-3: Los modelos de lenguaje como GPT-3 de OpenAI muestran capacidades asombrosas en la generación de texto, abriendo nuevas posibilidades y desafíos éticos.
  - **IA Multimodal y Modelos de Difusión.**
    Modelos Multimodales: Integración de múltiples tipos de datos (texto, imagen, audio) en un solo modelo. *Modelos de Difusión:* Innovación en la generación de imágenes y otros datos complejos, ampliando el horizonte de la creatividad artificial.

ferro@cimat.mx

Factores clave en la evolución de la IA

- **Big Data:**
  - La abundancia de datos ha permitido entrenar modelos más complejos y precisos.
- **Potencia computacional:**
  - **GPUs y TPUs:** Facilitación del entrenamiento de redes profundas.
  - **Cloud computing:** Acceso a recursos computacionales a gran escala.
- **Algoritmos y optimización:**
  - **Retropropagación y optimizadores:** Mejoras en la eficiencia y precisión del entrenamiento de redes neuronales.
- **Democratización de la IA:**
  - **Contenido técnico:** El contenido técnico especializado es más accesible.
  - **Plataformas, repositorios y ecosistemas:** Creación de plataformas y ambientes completos que permiten hacer uso y re-entrenamiento de modelos robustos de forma sencilla, así como acceso a grandes conjuntos de datos para entrenamiento.

ferro@cimat.mx

Transformers y su revolución

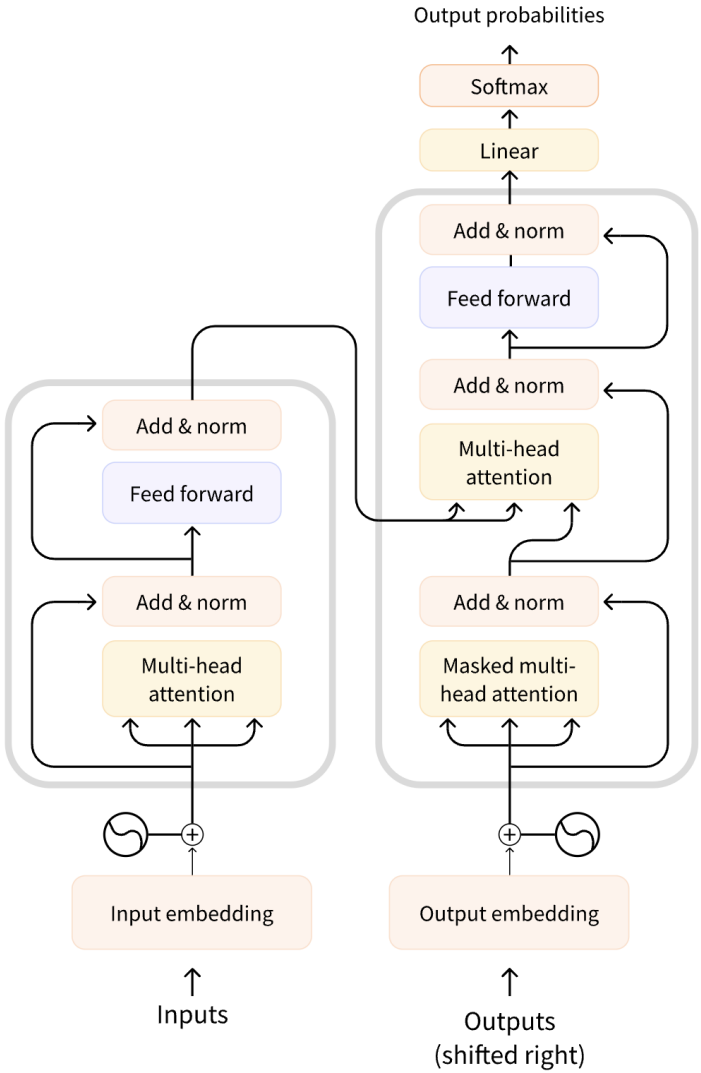
Panorama actual y futuro

- Es un modelo de deep learning que procesa secuencias de datos en paralelo utilizando mecanismos de autoatención.
- **Autoatención:** Permite a los modelos enfocarse en diferentes partes de una secuencia simultáneamente, mejorando la eficiencia y precisión.
- **Arquitectura:**
  - **Encoder-decoder:** Transformadores clásicos para traducción automática.
  - **Self-Attention (Autoatención):** Clave para manejar dependencias a largo plazo.

ferro@cimat.mx

Transformers y su revolución

Panorama actual y futuro



ferro@cimat.mx

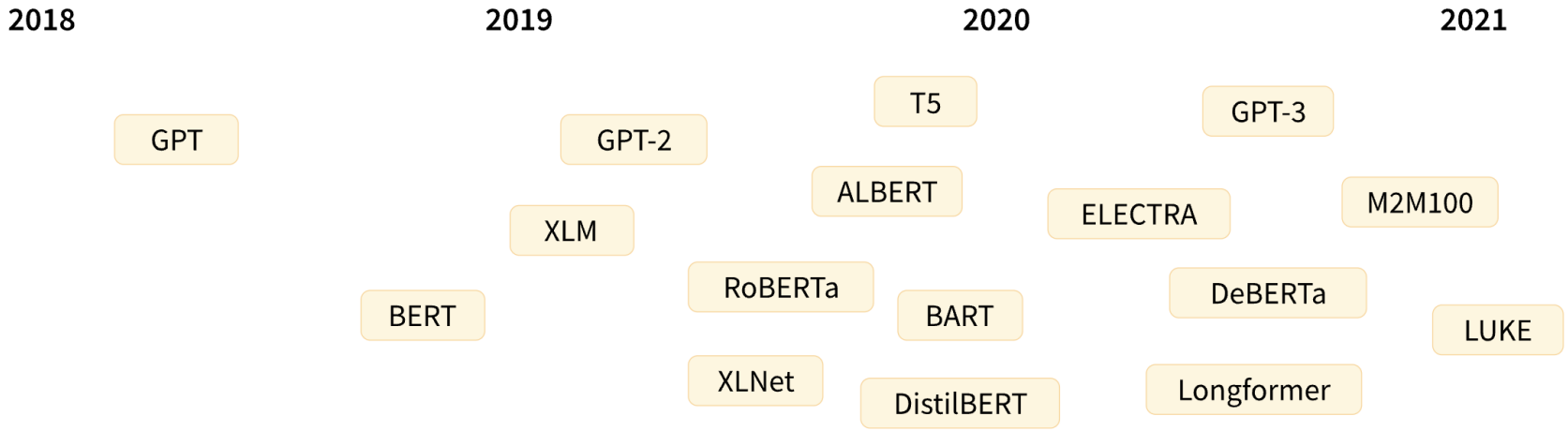
Impacto y aplicaciones de los Transformers

- **Procesamiento de Lenguaje Natural (NLP):**
  - **BERT (2018):** Representación bidireccional del lenguaje que mejoró la comprensión contextual.
  - **GPT (Generative Pretrained Transformer):** Generación de texto coherente y fluido.
- **Visión por computadora:**
  - **Vision Transformers (ViT):** Aplicación de transformers para tareas de clasificación de imágenes.
- **Variantes avanzadas:**
  - **GPT-3 (2020):** Uno de los modelos de lenguaje más grandes, con capacidad para generar texto coherente y realizar tareas complejas.
  - **Transformers Multimodales:** Integración de texto, imagen, y otras modalidades en un solo modelo.

ferro@cimat.mx

Impacto y aplicaciones de los Transformers

Panorama actual y futuro



ferro@cimat.mx

Modelos de lenguaje de gran escala (LLMs)

- Son modelos entrenados en grandes volúmenes de texto para generar, comprender, y manipular lenguaje natural.
- **Características:**
  - **Pre-entrenamiento en datasets masivos:** Aprende patrones generales del lenguaje antes de ser ajustado para tareas específicas.
  - **Capacidades avanzadas:** Desde completar texto hasta realizar traducciones y análisis sentimentales.

ferro@cimat.mx

Aplicaciones clave de LLMs

- **Generación de texto:** Creación de contenido automatizado, desde artículos hasta código.
- **Chatbots y asistentes virtuales:** Asistentes como ChatGPT, que pueden sostener conversaciones coherentes.
- **Traducción automática:** Mejora en la precisión y fluidez en traducciones entre diferentes idiomas.
- **Asistencia en programación:** Herramientas que ayudan a los desarrolladores a escribir y depurar código.

ferro@cimat.mx

Desafíos y consideraciones éticas

Panorama actual y futuro

- **Sesgos en modelos:** Los LLMs pueden reflejar y amplificar sesgos presentes en los datos de entrenamiento.
- **Uso ético y responsable:** Implicaciones de confiar en modelos que pueden generar información incorrecta o sesgada.
- **Tendencias futuras:**
  - **Modelos multimodales:** Expansión hacia modelos que pueden manejar múltiples tipos de datos simultáneamente.
  - **Optimización y sostenibilidad:** Reducción del costo computacional y mejora en la eficiencia.

ferro@cimat.mx

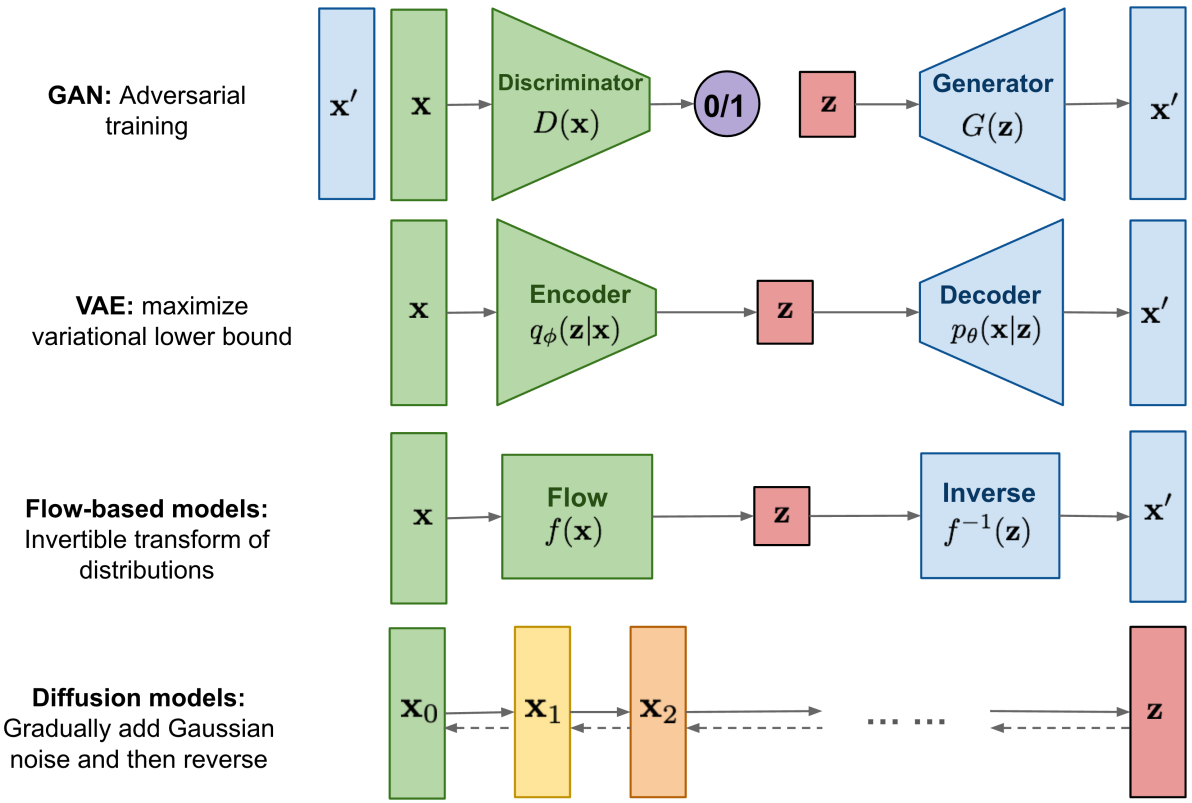
Ejercicio: Exploración de HuggingFace

Hugging Face

ferro@cimat.mx

Estructura de modelos generativos

Panorama actual y futuro



ferro@cimat.mx

Modelos de difusión

Panorama actual y futuro

- Son modelos generativos que aprenden a generar datos simulando el proceso inverso de difundir ruido a través de datos.
- **Funcionamiento:**
  - **Proceso de difusión:** Agregación gradual de ruido a los datos de entrenamiento.
  - **Proceso inverso:** El modelo aprende a revertir este ruido, generando datos realistas.

ferro@cimat.mx

Ejemplos y aplicaciones de modelos de difusión

- **Generación de Imágenes:**
  - **DALL-E 2:** Generación de imágenes a partir de descripciones textuales.
  - **Stable Diffusion:** Generación de imágenes de alta calidad con control sobre el proceso creativo.
- **Simulaciones realistas:** Creación de mundos virtuales y simulaciones precisas para aplicaciones en videojuegos y simulación científica.

ferro@cimat.mx

Desafíos y consideraciones éticas

- **Controlabilidad:** Desafíos en el control preciso de la salida generada por los modelos.
- **Complejidad computacional:** La generación puede ser costosa en términos de tiempo y recursos.
- **Potencial futuro:**
  - **Ampliación a otras modalidades:** Aplicación en generación de audio, video, y otros tipos de datos.
  - **Innovaciones en creatividad artificial:** Implicaciones para el arte, el diseño, y la innovación tecnológica.

ferro@cimat.mx

Ejercicio: Exploración de Dream Studio

stability.ai

ferro@cimat.mx

Lecturas recomendadas

- How Transformers Work: A Detailed Exploration of Transformer Architecture
- NLP Course by HuggingFace
- Transformers, explained: Understand the model behind GPT, BERT, and T5
- What are Diffusion Models?
- Dream Studio by stability.ai

ferro@cimat.mx

Section 6: **Empaquetado de modelos**