

Introducción al Procesamiento de Lenguaje Natural (NLP)

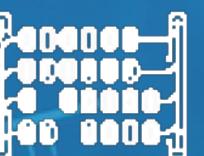
Clase IV: Sistemas de Reconocimiento
Automático de Voz (ASR) IV. RNN

Alexander Caicedo



Universidad del
Rosario

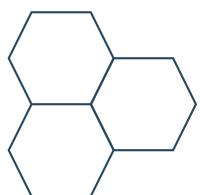
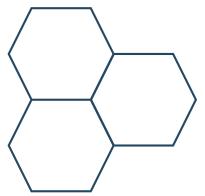
Escuela de Ingeniería,
Ciencia y Tecnología



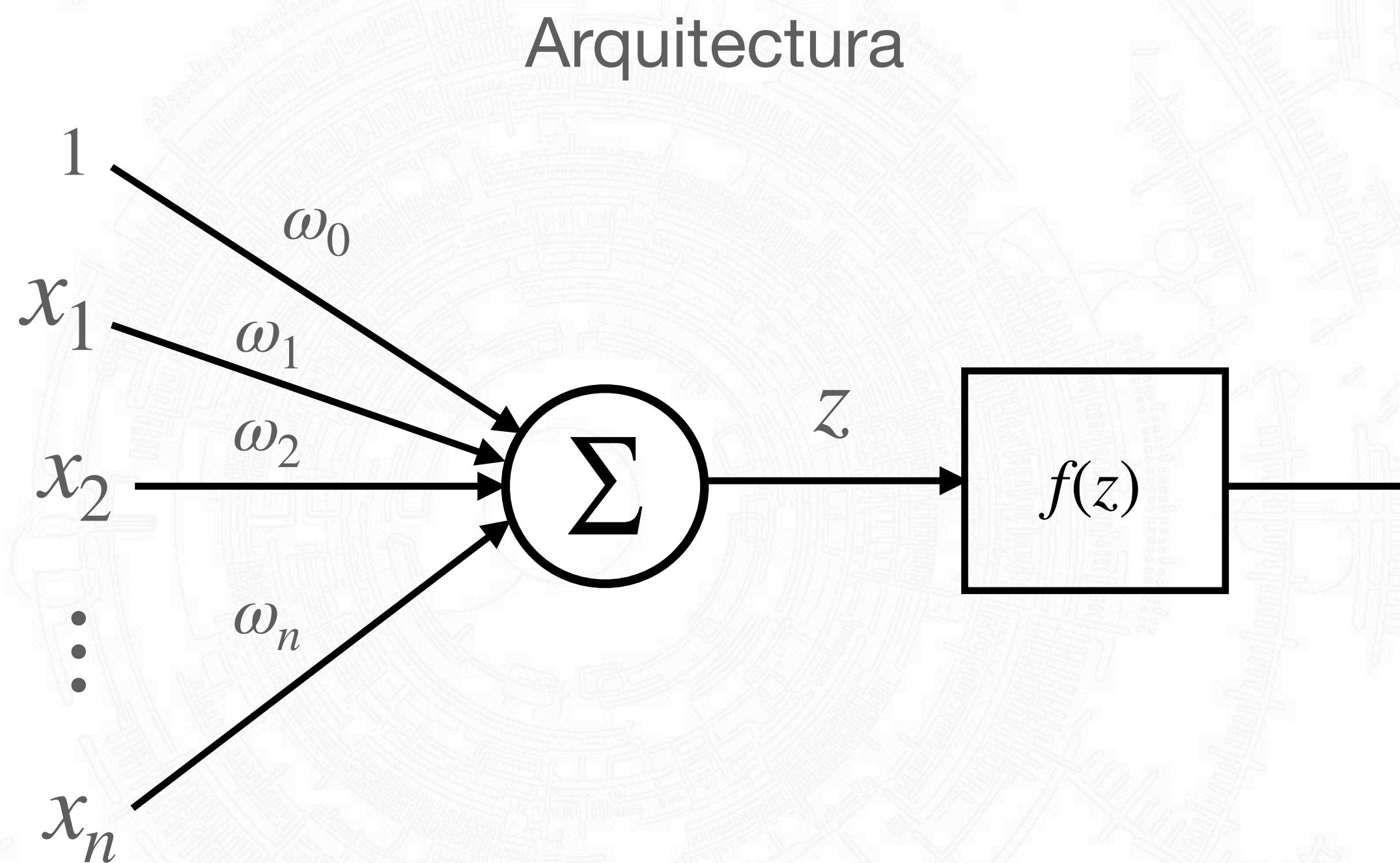
MACC
Matemáticas Aplicadas y
Ciencias de la Computación

Contenido

Neural Networks Arquitectura. Recurrent Neural Networks



Perceptron



$$y = f(\mathbf{x}^T \boldsymbol{\omega})$$

Formulación

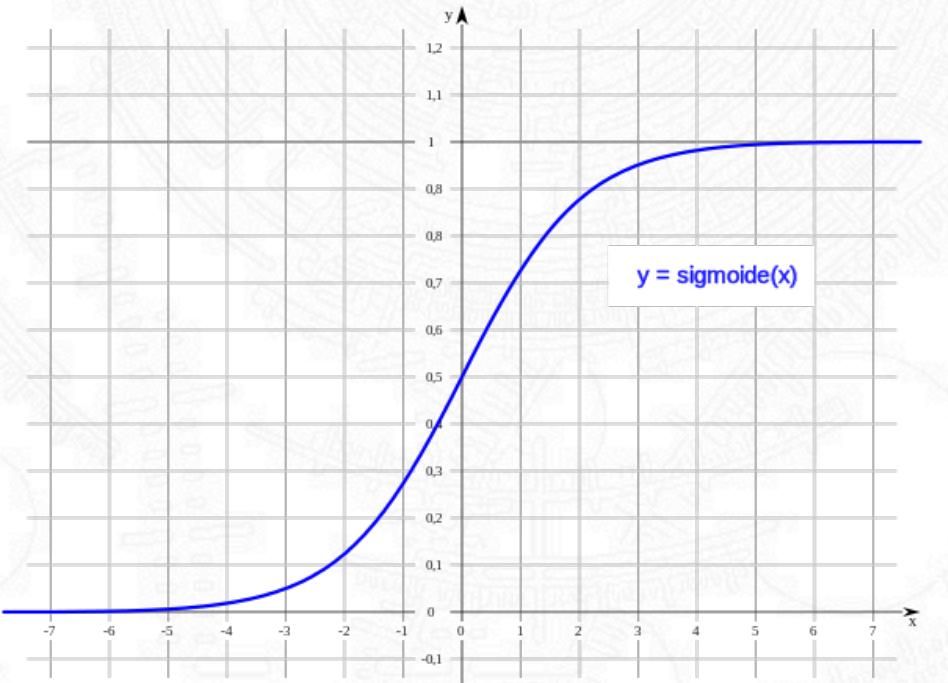
$$y = f\left(\omega_0 + \sum_{i=1}^n \omega_i x_i\right)$$
$$\mathbf{x} = [1 \quad x_1 \quad \dots \quad x_n]^T$$
$$\boldsymbol{\omega} = [\omega_0 \quad \omega_1 \quad \dots \quad \omega_n]^T$$

Perceptron

Función de Activación

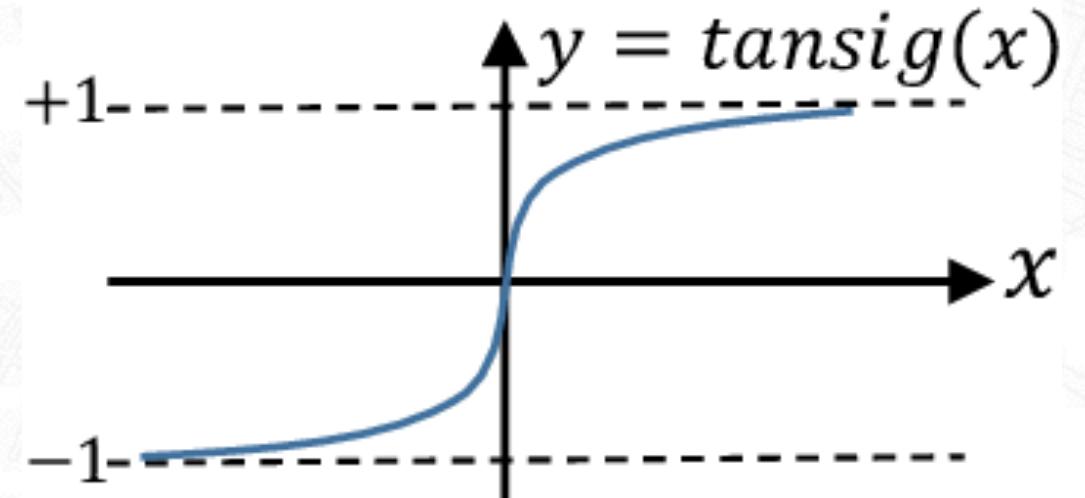
Logística

$$f(z) = \frac{1}{1 + e^{-z}}$$



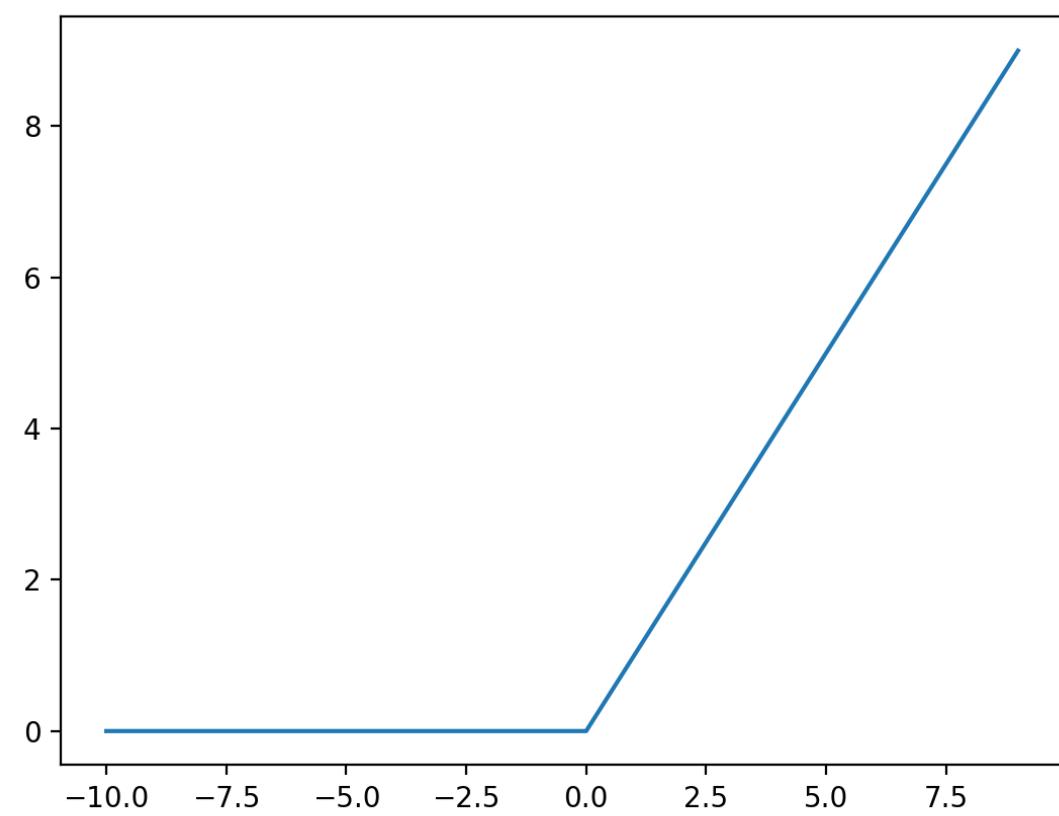
Tangente Hiperbolica

$$f(z) = \frac{2}{1 + e^{-2z}} - 1$$

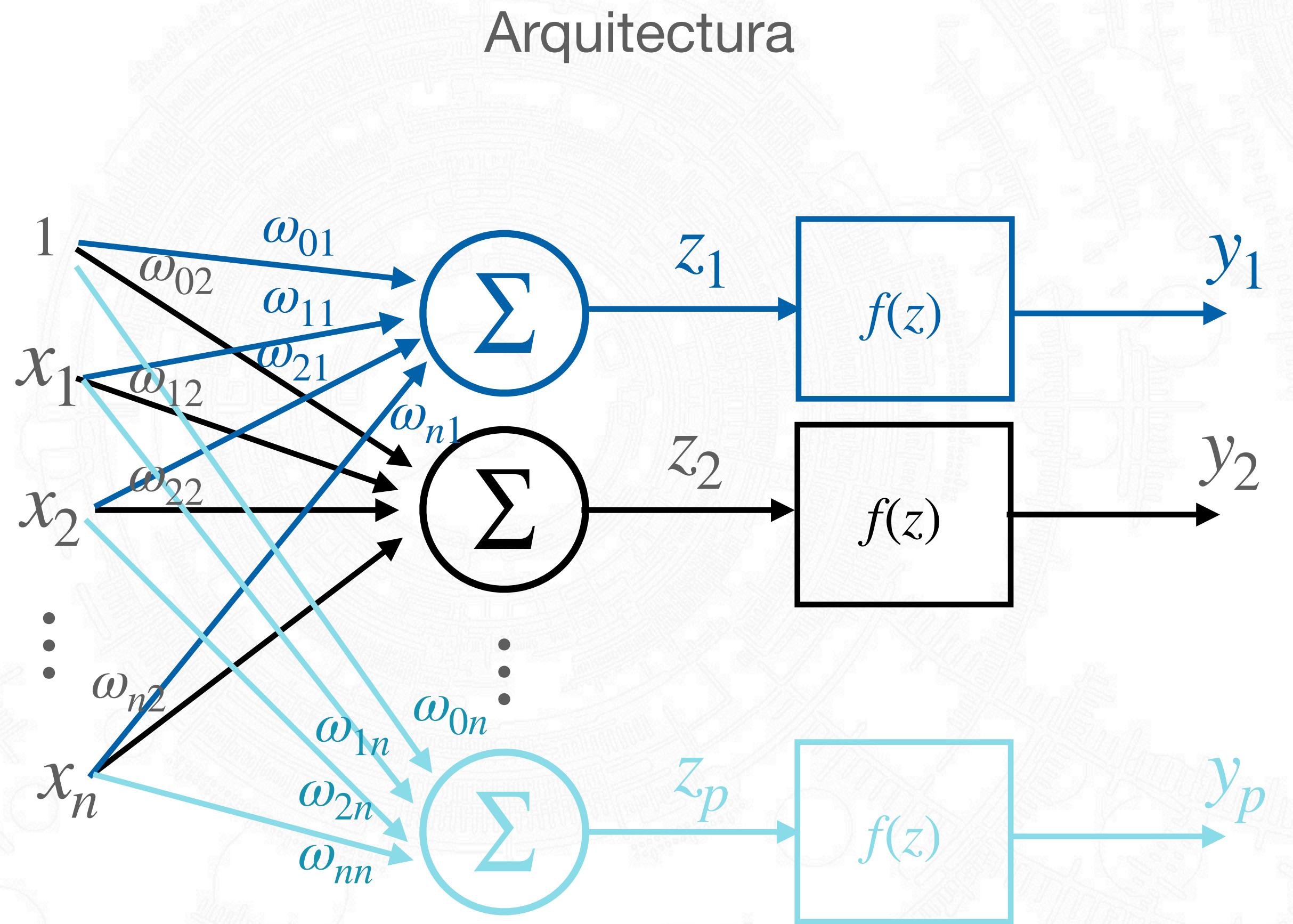


ReLU
Rectified Linear Unit

$$f(z) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$



Multiple salida



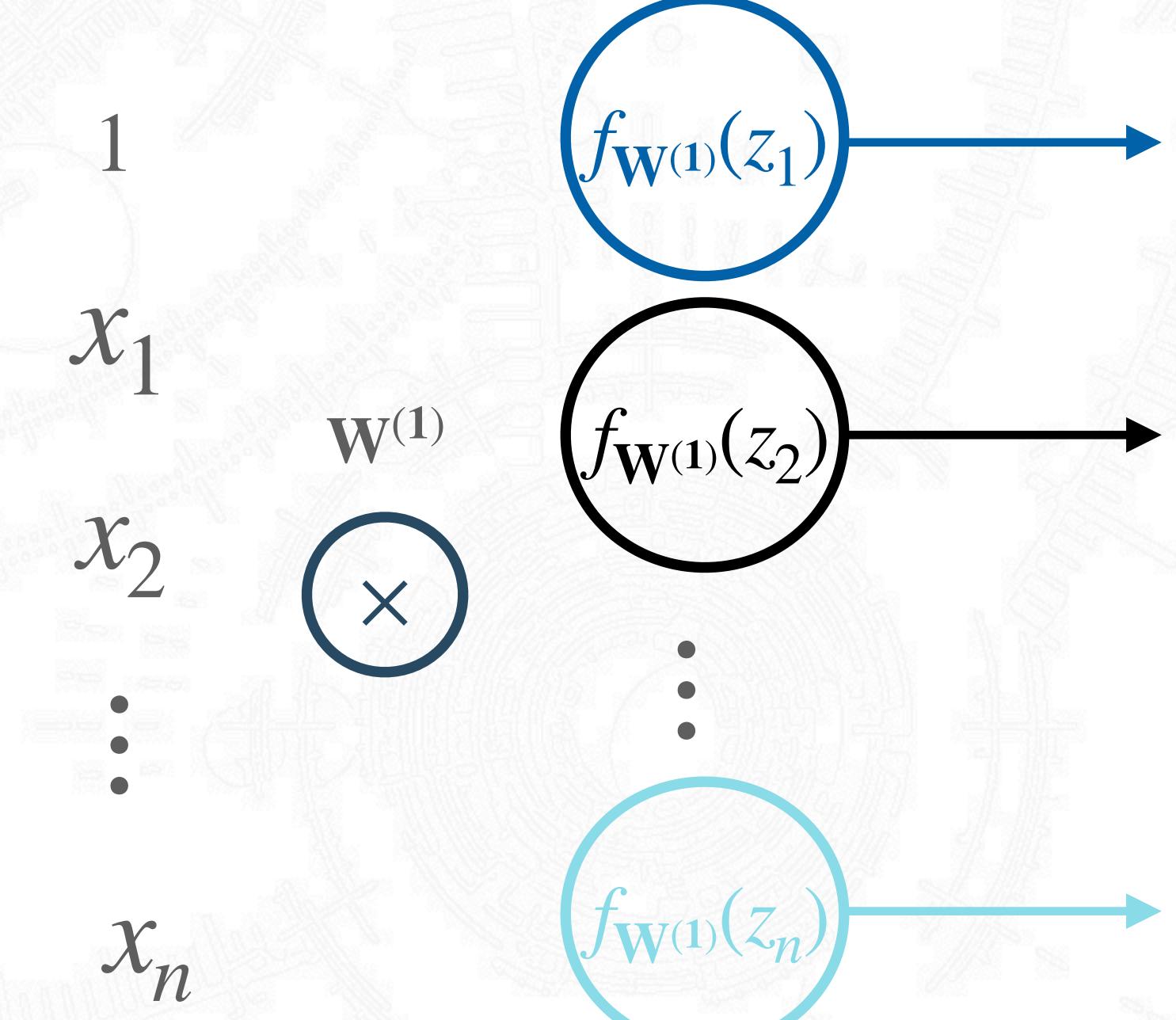
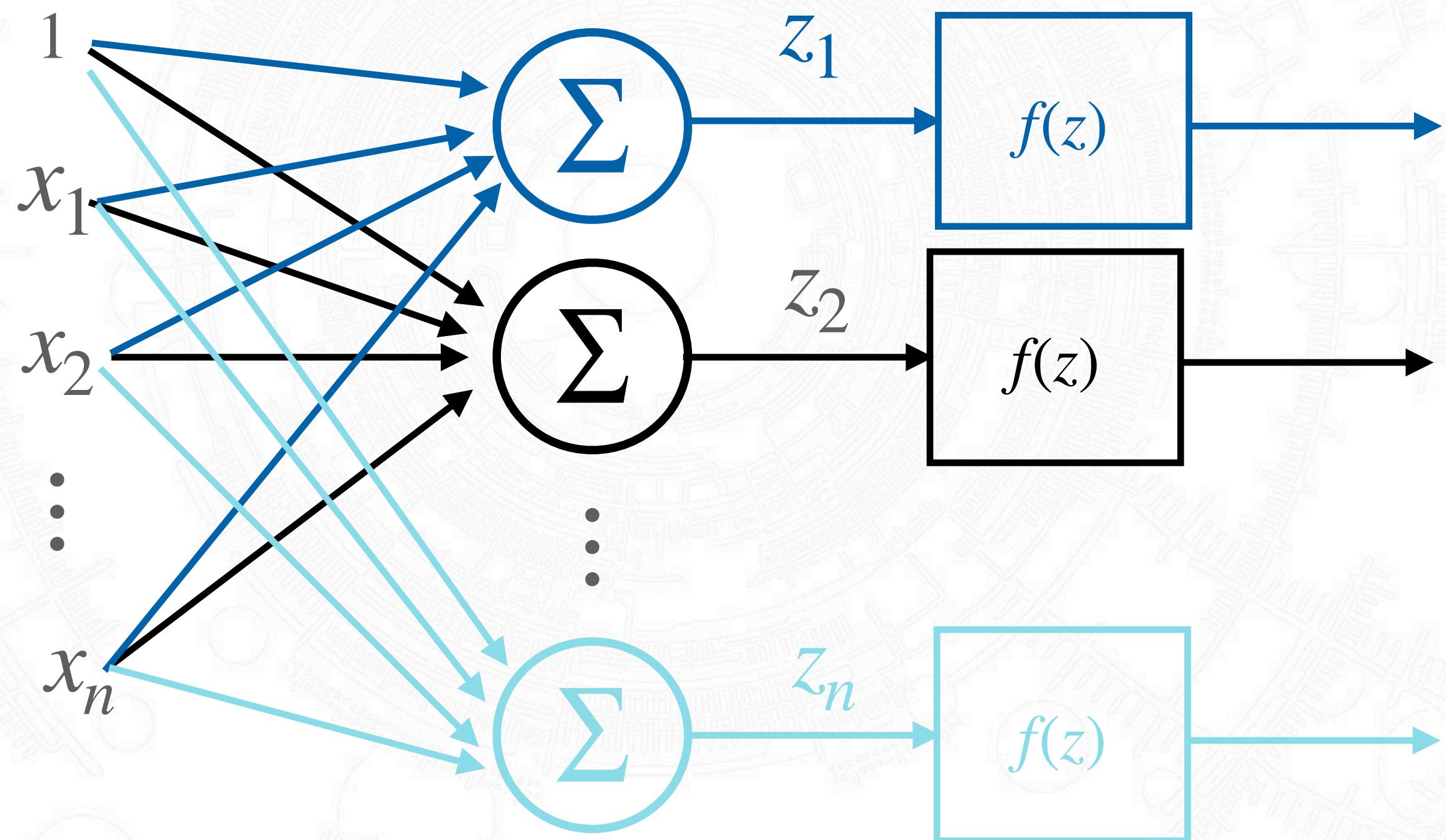
$$\mathbf{x} = [1 \quad x_1 \quad \dots \quad x_n]^T$$
$$\mathbf{W}_{n_1 \times n_2} = \begin{bmatrix} \omega_{01} & \omega_{11} & \dots & \omega_{n_2 1} \\ \omega_{02} & \omega_{12} & \dots & \omega_{n_2 2} \\ \vdots & \vdots & \ddots & \vdots \\ \omega_{0n_1} & \omega_{1n} & \dots & \omega_{n_1 n_2} \end{bmatrix}$$

n_1 : Número de datos de entrada.
 n_2 : Número de datos de Salida.

$$\mathbf{y} = f(\mathbf{W}\mathbf{x})$$

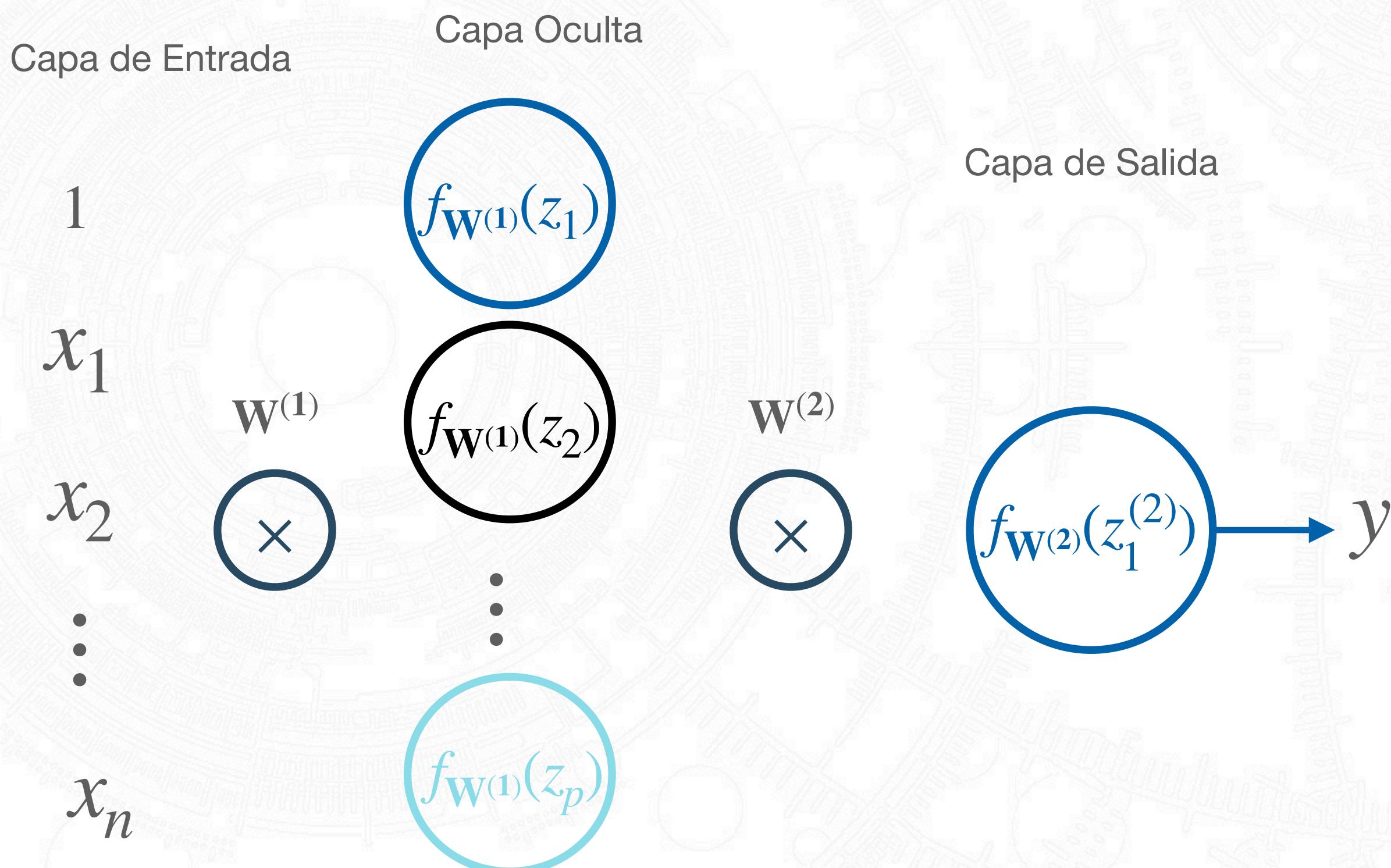
Red Neuronal FeedForward

Arquitectura



Red Neuronal FeedForward

Arquitectura

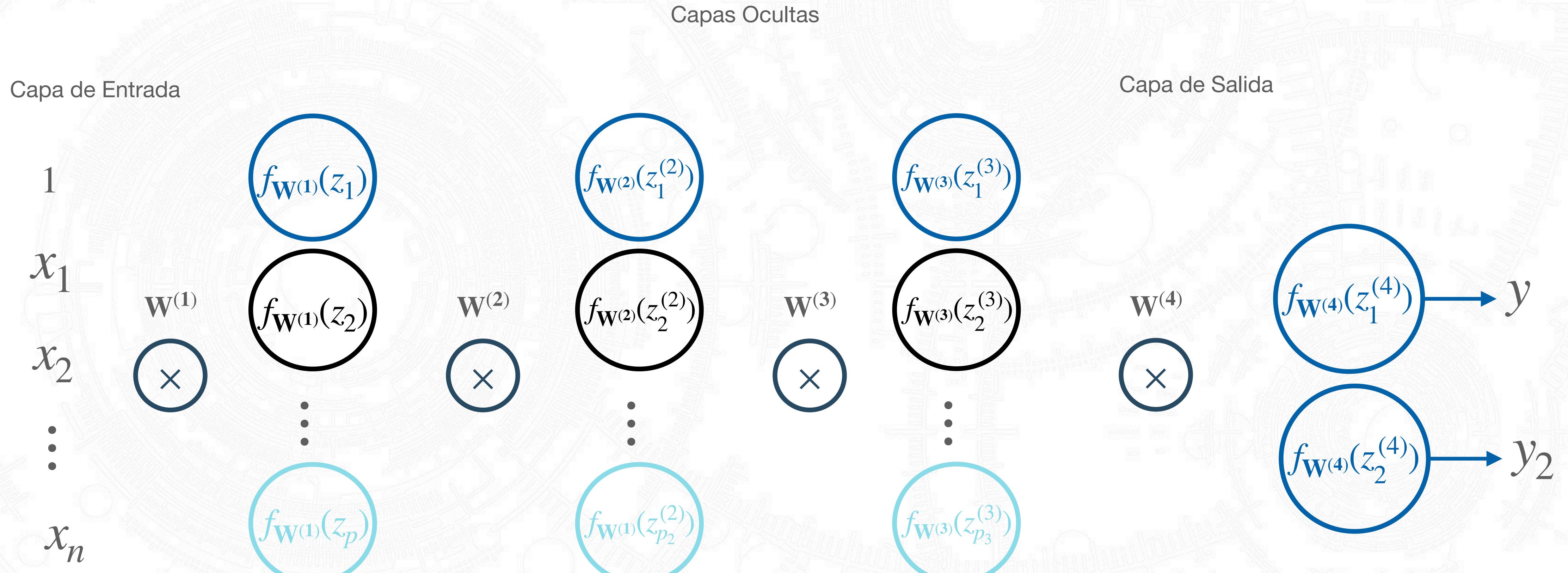


$$\mathbf{y} = f\left(\mathbf{W}^{(2)}f\left(\mathbf{W}^{(1)}\mathbf{x}\right)\right)$$

$$\mathbf{W}^{(1)} \in \mathbb{R}^{(n+1) \times p}$$

$$\mathbf{W}^{(2)} \in \mathbb{R}^{p \times 1}$$

Red Neuronal FeedForward



Defina la ecuación de la salida, y las dimensiones de cada una de las matrices.

Entrenamiento

¿Cómo entreno (aprende) el modelo?

Se utiliza el algoritmo de Backpropagation. El cual se basa en una combinación entre gradiente descendiente (como algoritmo de aprendizaje o actualización de pesos), y la regla de la cadena de la derivada (para encontrar los cambios de la función de costo en función de los parámetros)

¿Más sobre Back-propagation y Gradiente Descendiente? -> (Curso de Machine Learning)

Idea de Backpropagation

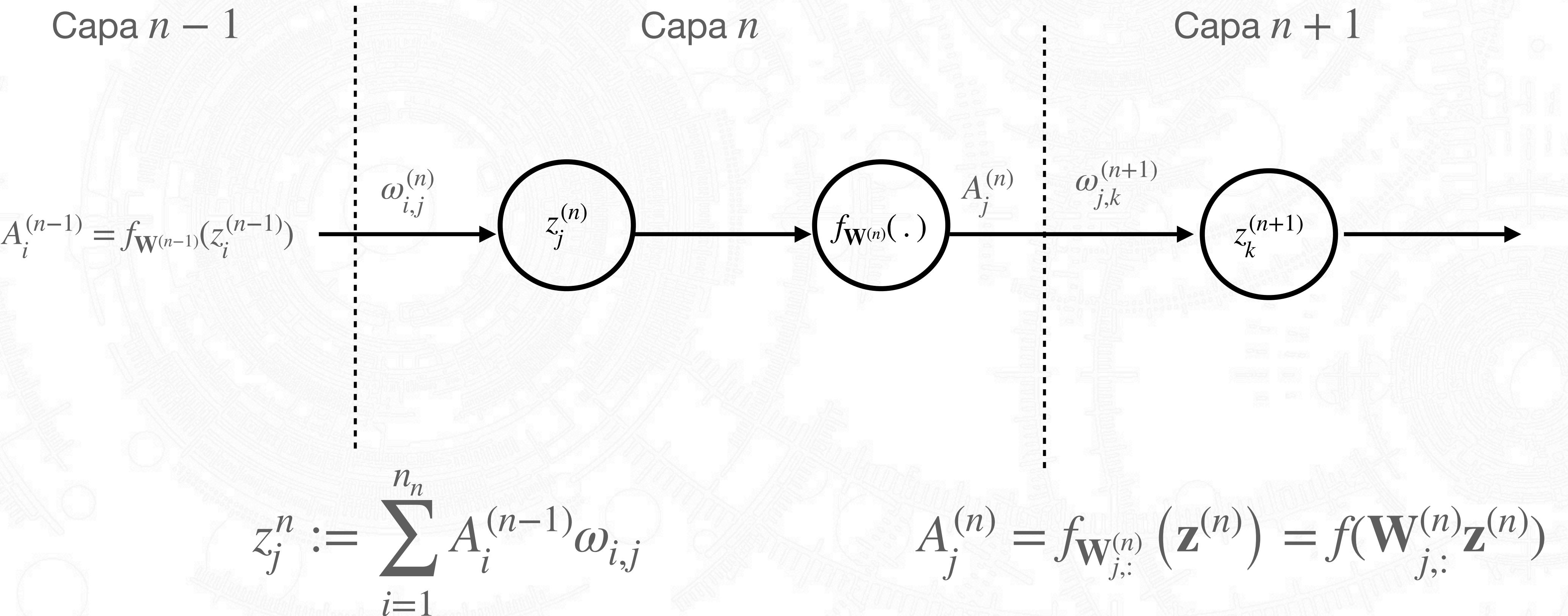
Gradiente Descendiente

$$L(W) = \frac{1}{m} \sum_{i=1}^m (y^{(i)} - \hat{y}^{(i)})^2$$

¿Dónde entra aquí la regla de la cadena?

$$\omega_{i,j} := \omega_{i,j} - \eta \frac{\partial L(W)}{\partial \omega_{i,j}}$$

Idea de Backpropagation



Idea de Backpropagation

$$\frac{\partial L}{\partial \omega_{i,j}} = \frac{\partial L}{\partial z_j^n} \frac{\partial z_j^{(n)}}{\partial \omega_{i,j}}$$

$$\frac{\partial L}{\partial \omega_{i,j}} = \frac{\partial L}{\partial A_j^{(n)}} \frac{\partial A_j^{(n)}}{\partial z_j^{(n)}} A_i^{(n-1)}$$

$$\frac{\partial L}{\partial \omega_{i,j}} = \frac{\partial L}{\partial A_j^{(n)}} (A_j^{(n)}(1 - A_j^{(n)}) A_i^{(n-1)})$$

$$\frac{\partial z_j^{(n)}}{\partial \omega_{i,j}} = A_i^{(n-1)}$$

$$\frac{\partial A_j^{(n)}}{\partial z_j^{(n)}} = A_j^{(n)}(1 - A_j^{(n)})$$

Idea de Backpropagation

$$\frac{\partial L}{\partial A_j^{(n)}} = \sum_k \left[\frac{\partial L}{\partial z_k^{(n+1)}} \frac{\partial z_k^{(n+1)}}{\partial A_j^{(n)}} \right]$$

$$\frac{\partial z_k^{(n+1)}}{\partial A_j^{(n)}} = \omega_{j,k}^{(n+1)}$$

$$\frac{\partial L}{\partial \omega_{i,j}} = \sum_k \left[\frac{\partial L}{\partial z_k^{(n+1)}} \omega_{i,j}^{(n+1)} \right] (A_j^{(n)}(1 - A_j^{(n)})A_i^{(n-1)})$$

Para encontrar la derivada con respecto a un peso ubicado en la capa n , necesito la derivada del costo en función de la salida de la capa $n + 1$

Idea de Backpropagation

Consideramos que resolvemos los gradientes en propagación hacia atrás. Si estamos en la última capa, tendríamos:

$$\frac{\partial \mathbf{L}}{\partial \omega_{i,j}} = \frac{\partial \mathbf{L}}{\partial z_j^n} \frac{\partial z_j^{(n)}}{\partial \omega_{i,j}}$$

$$\frac{\partial z_j^{(n)}}{\partial \omega_{i,j}} = A_i^{(n-1)}$$

$$\mathbf{L}(\mathbf{W}) = \frac{1}{m} \sum_{i=1}^m (y^{(i)} - \hat{y}^{(i)})^2 := (y - z_j^{(n)})^2$$

$$\frac{\partial \mathbf{L}}{\partial z_j^n} = -2(y - z_j^{(n)})$$

$$\frac{\partial \mathbf{L}}{\partial \omega_{i,j}} = -2(y - z_j^{(n)}) A_i^{(n-1)}$$

Con estos valores empiezo a propagar hacia atrás el error, y de esa forma se actualizan los pesos.

Aplicaciones

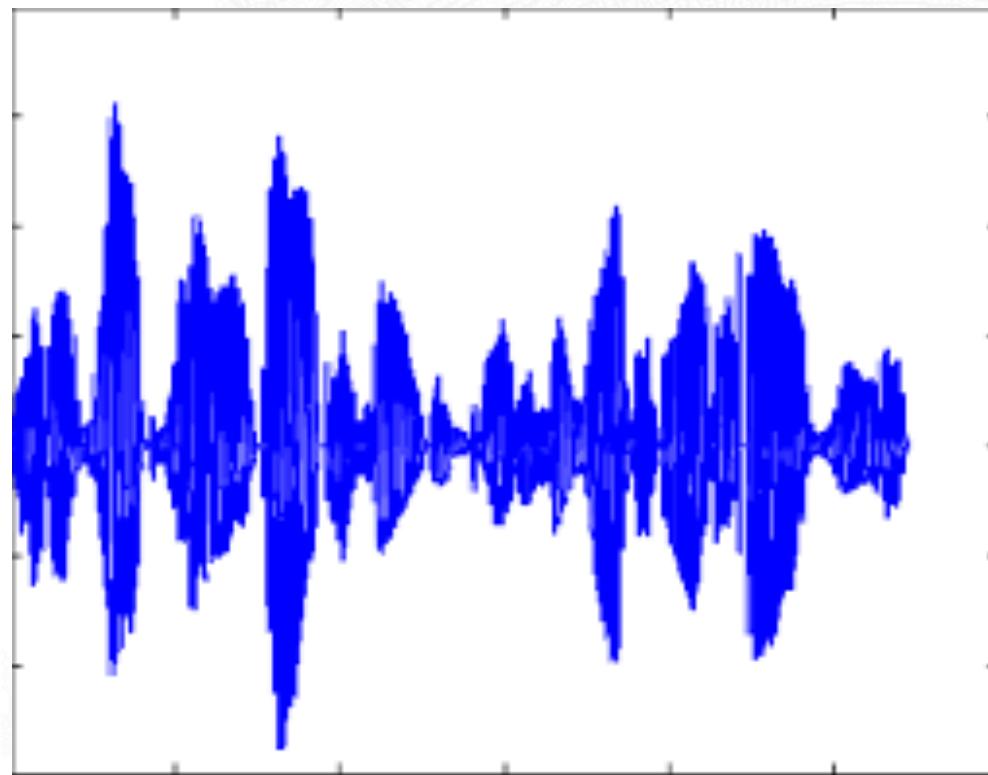
Manejan muy bien datos de entrada que no son secuenciales.
(Datos cuya salida actual no depende entradas o salidas anteriores.)

Sí tengo un sistema dinámico, para predecir su comportamiento requiero no solamente de información actual del sistema sino, también, información del pasado.

¿Cómo puedo obtener esa información?

¿Porqué datos secuenciales?

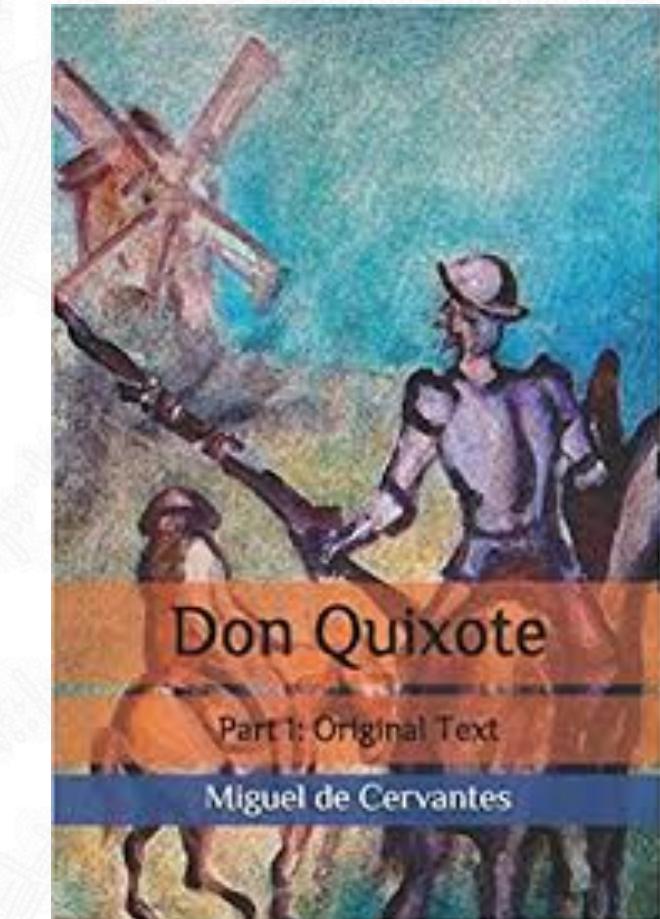
Voz (Habla)



Video



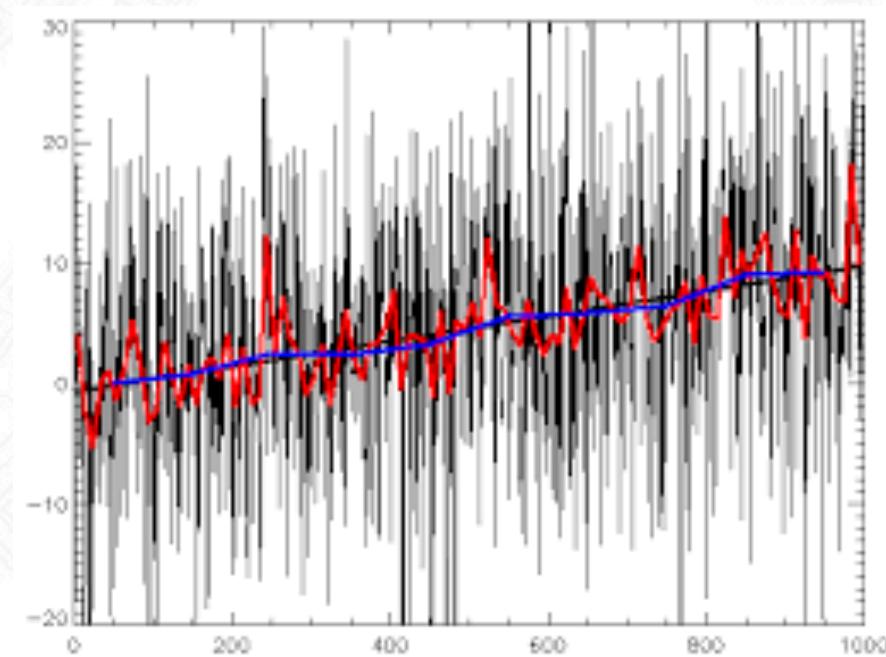
Texto



Música

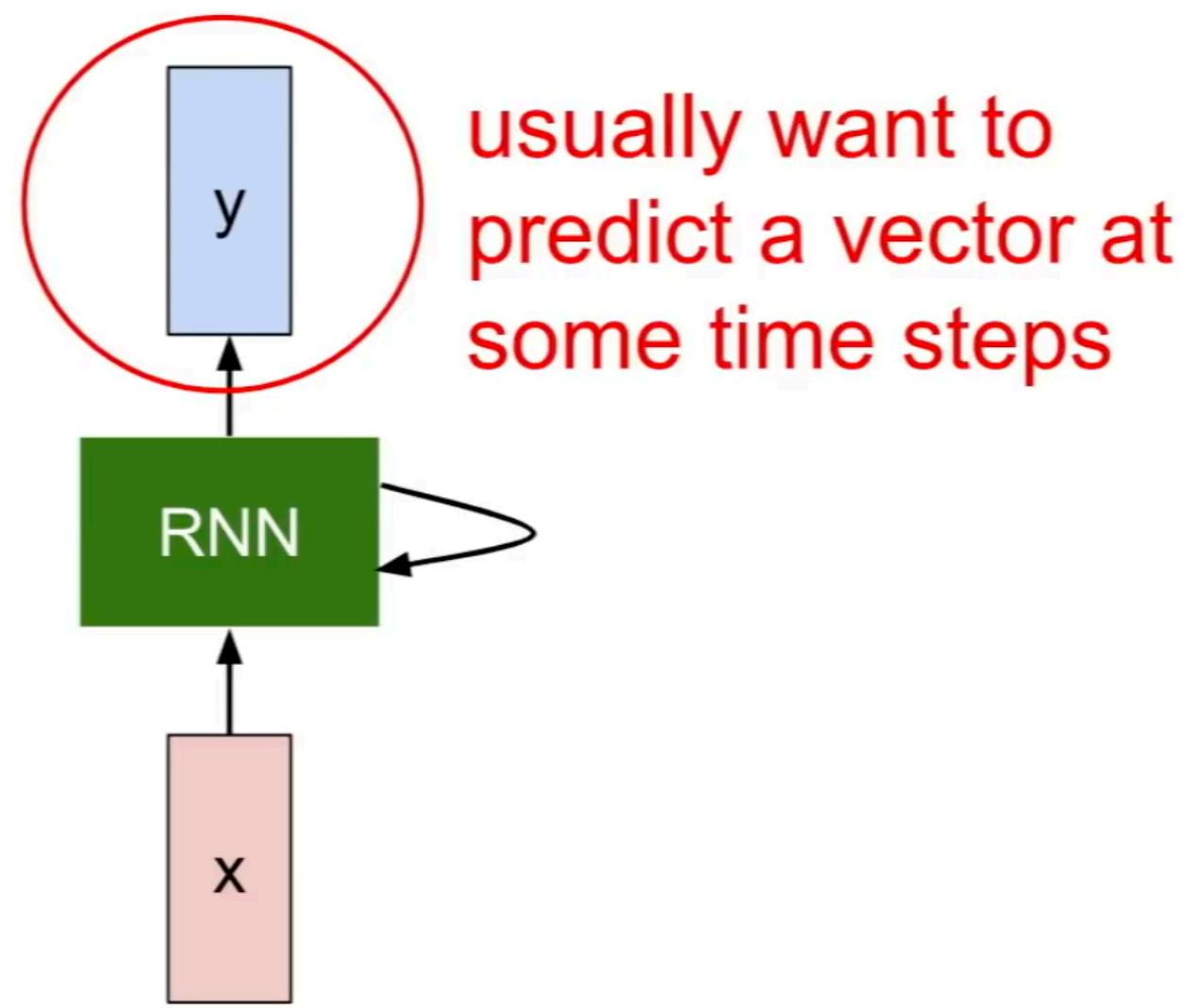


Series de Tiempo



Entre otros ...

¿Porqué datos secuenciales?



$$\mathbf{h}_t = f_{\mathbf{W}}(\mathbf{h}_{t-1}, \mathbf{x}_t)$$

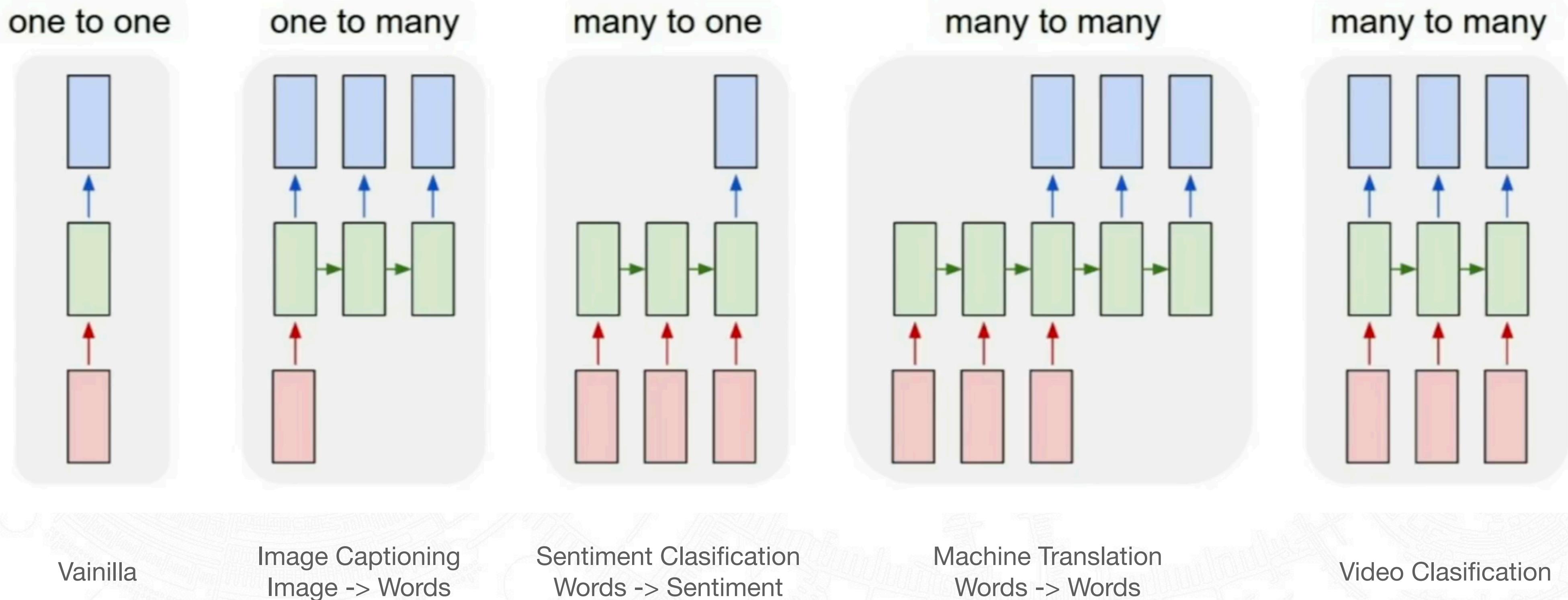
$$\mathbf{h}_t = \tanh(\mathbf{W}_{\mathbf{hh}}\mathbf{h}_{t-1} + \mathbf{W}_{\mathbf{xh}}\mathbf{x}_t)$$

$$\mathbf{y}_t = \mathbf{W}_{\mathbf{hy}}\mathbf{h}_t$$

Grafique la arquitectura

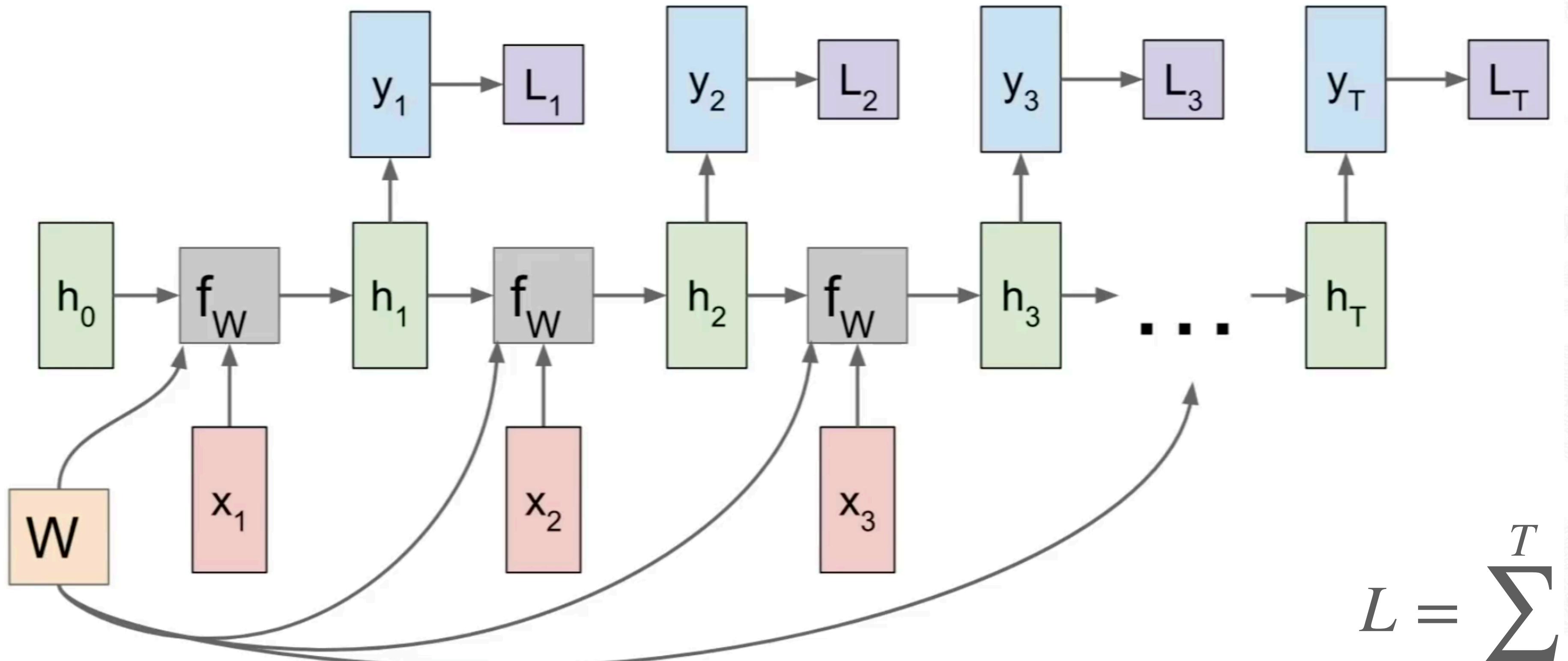
Source: <https://www.youtube.com/watch?v=6niqTuYFZLQ>

¿Porqué datos secuenciales?



Source: <https://www.youtube.com/watch?v=6niqTuYFZLQ>

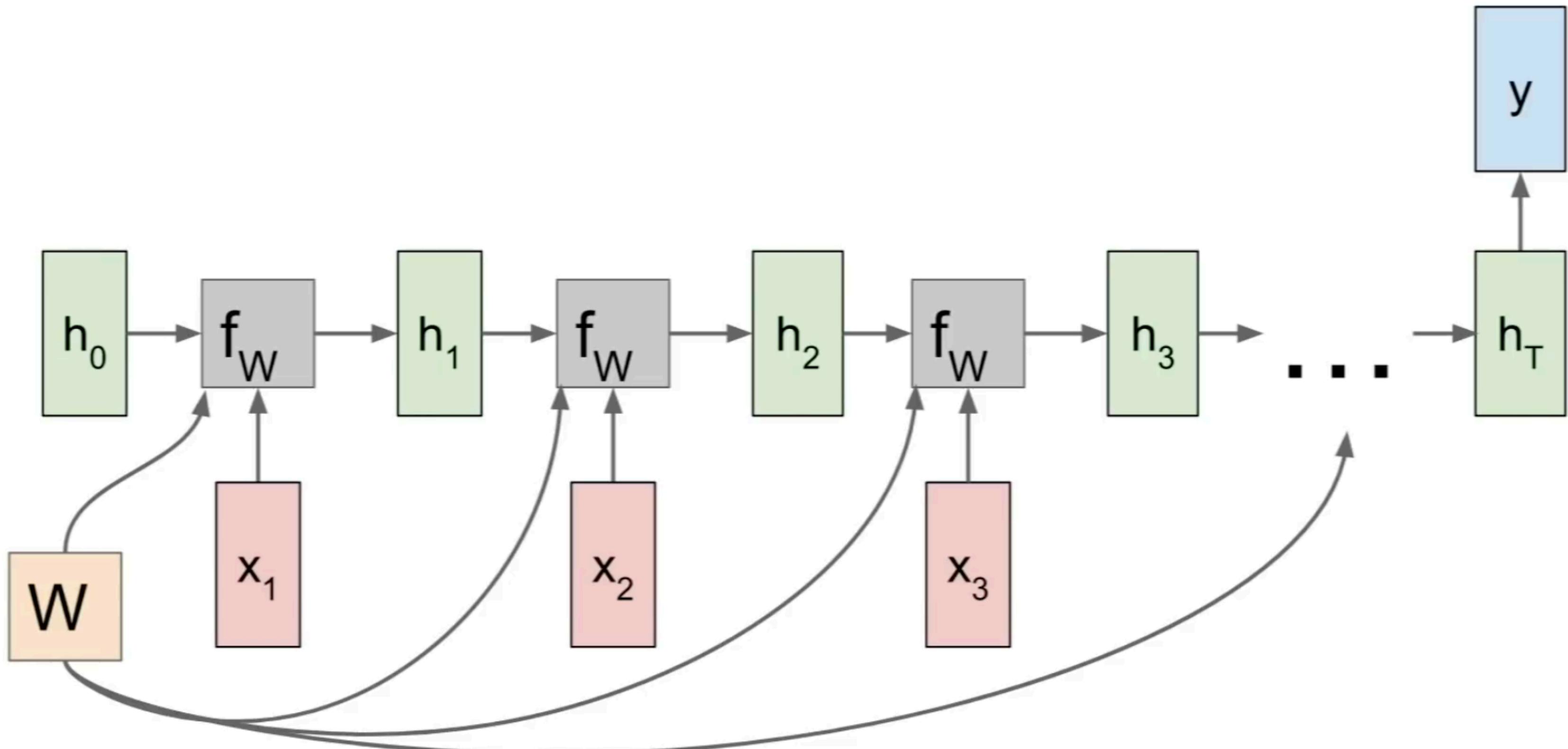
¿Porqué datos secuenciales?



$$L = \sum_{i=1}^T L_i$$

Source: <https://www.youtube.com/watch?v=6niqTuYFZLQ>

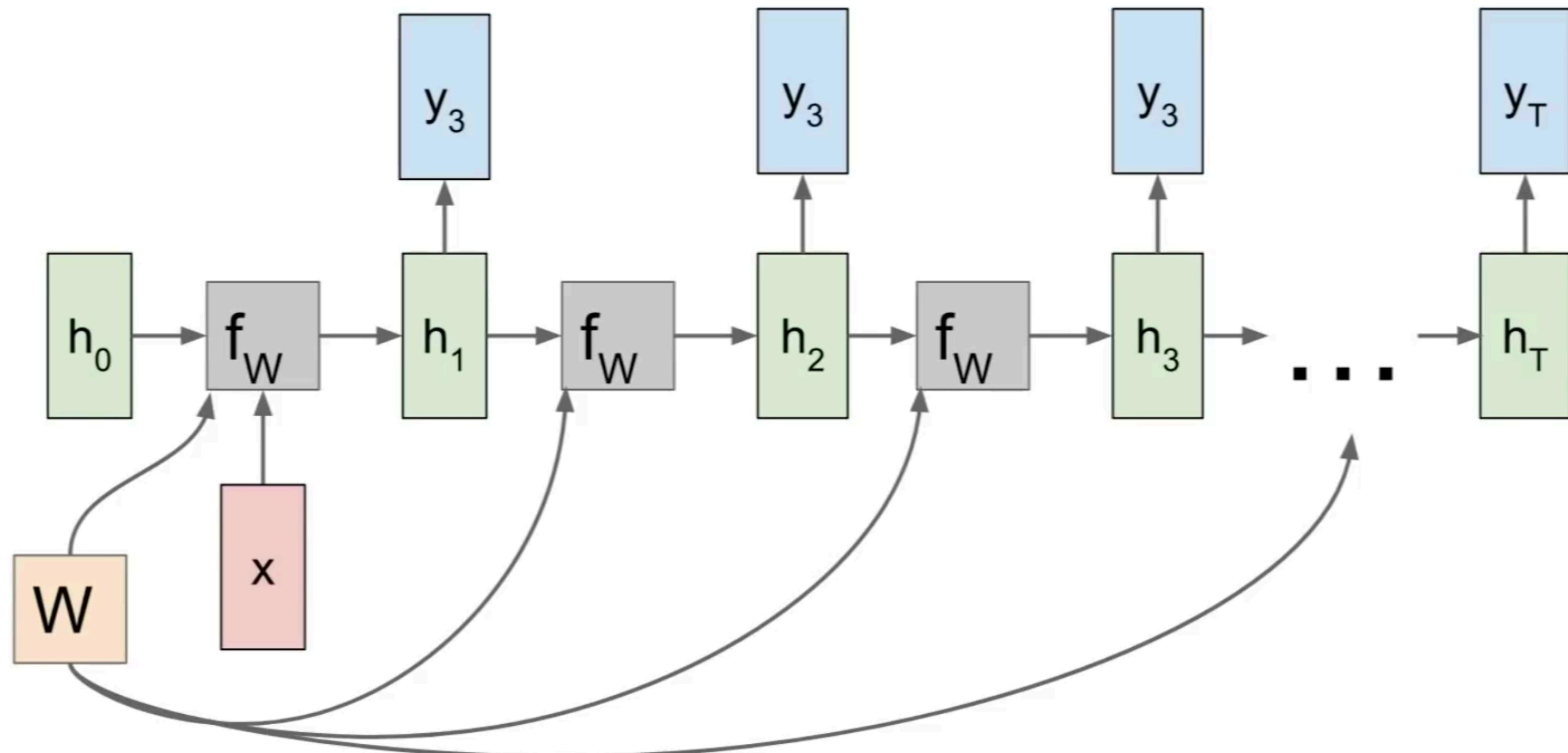
¿Porqué datos secuenciales?



Many to one

Source: <https://www.youtube.com/watch?v=6niqTuYFZLQ>

¿Porqué datos secuenciales?

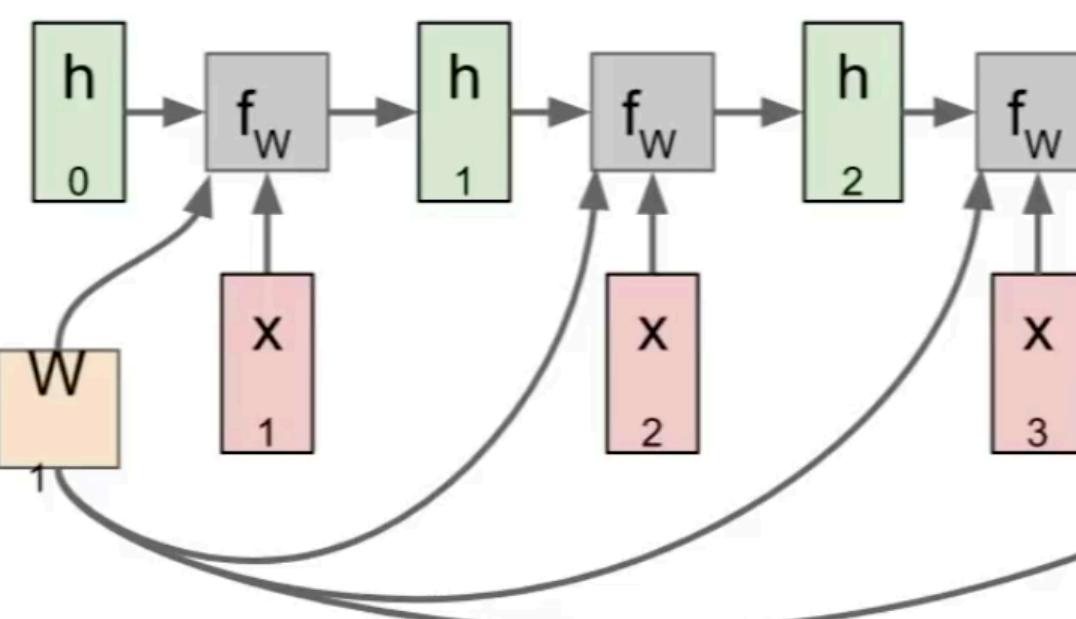


One to Many
Speech Synthesizer

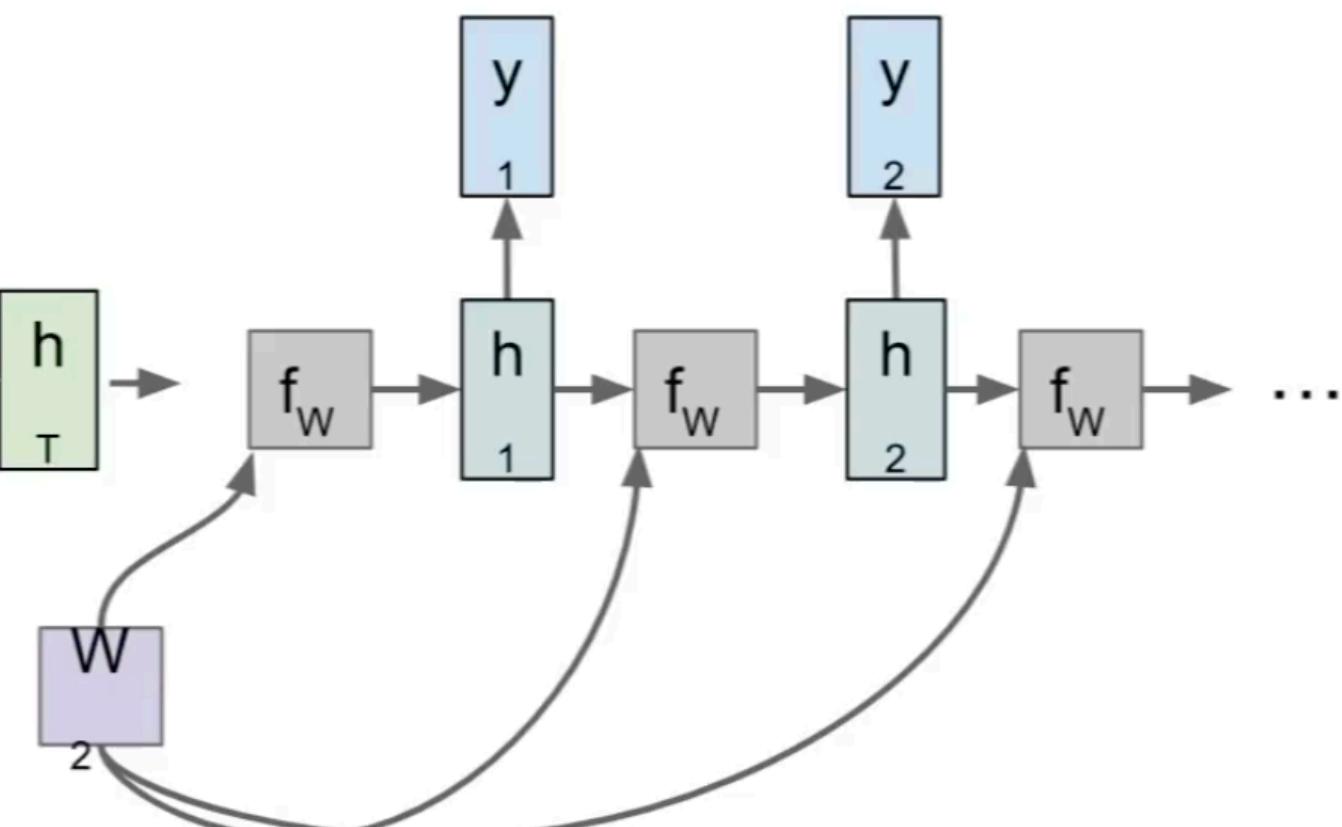
Source: <https://www.youtube.com/watch?v=6niqTuYFZLQ>

¿Porqué datos secuenciales?

Many to one: Encode input sequence in a single vector



One to many: Produce output sequence from single input vector



Machine Translation

Source: <https://www.youtube.com/watch?v=6niqTuYFZLQ>



Gracias
Preguntas?