

# Inteligencia Artificial Aplicada para la Economía

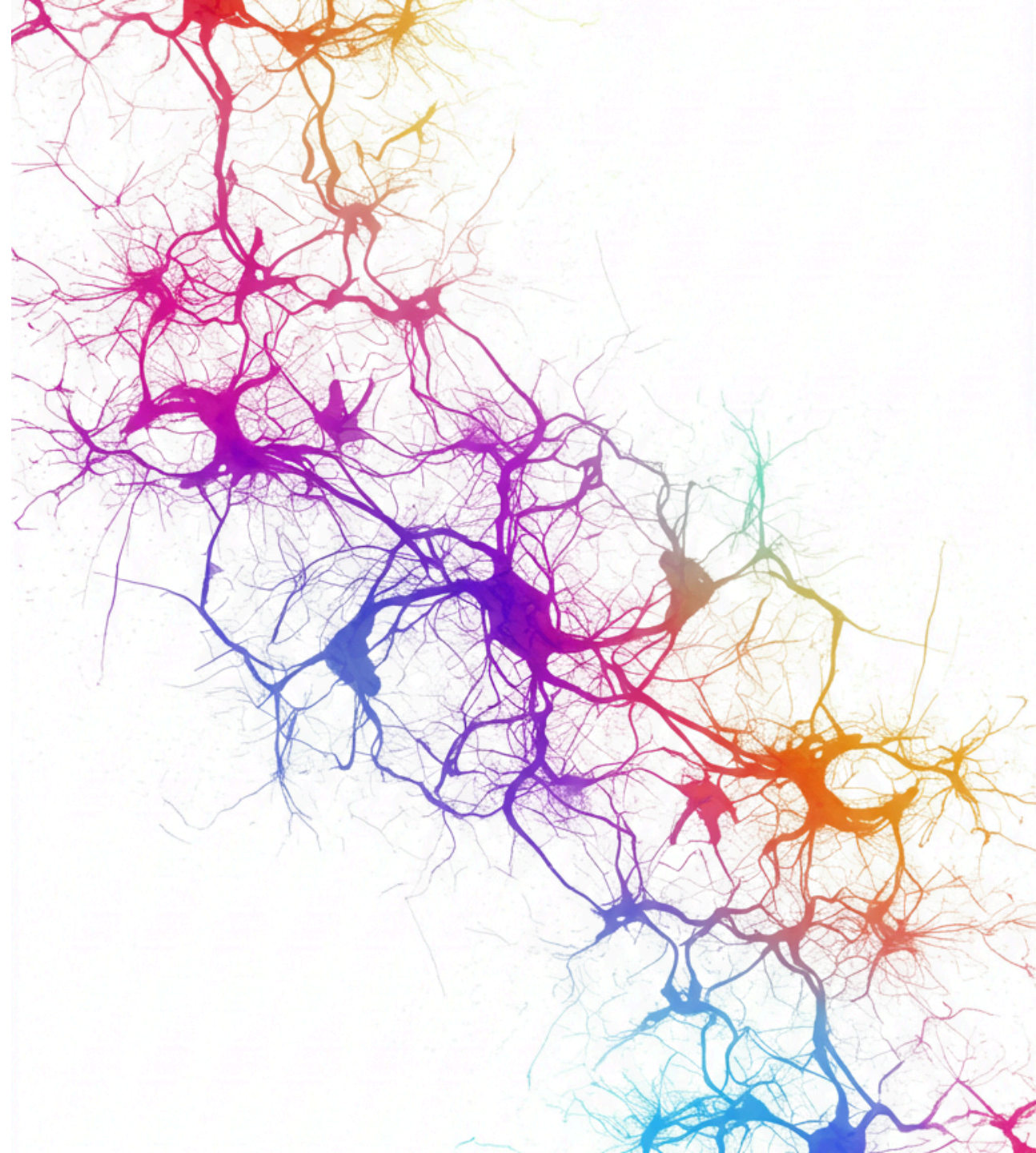
## Profesores

### Profesor Magistral

Camilo Vega Barbosa

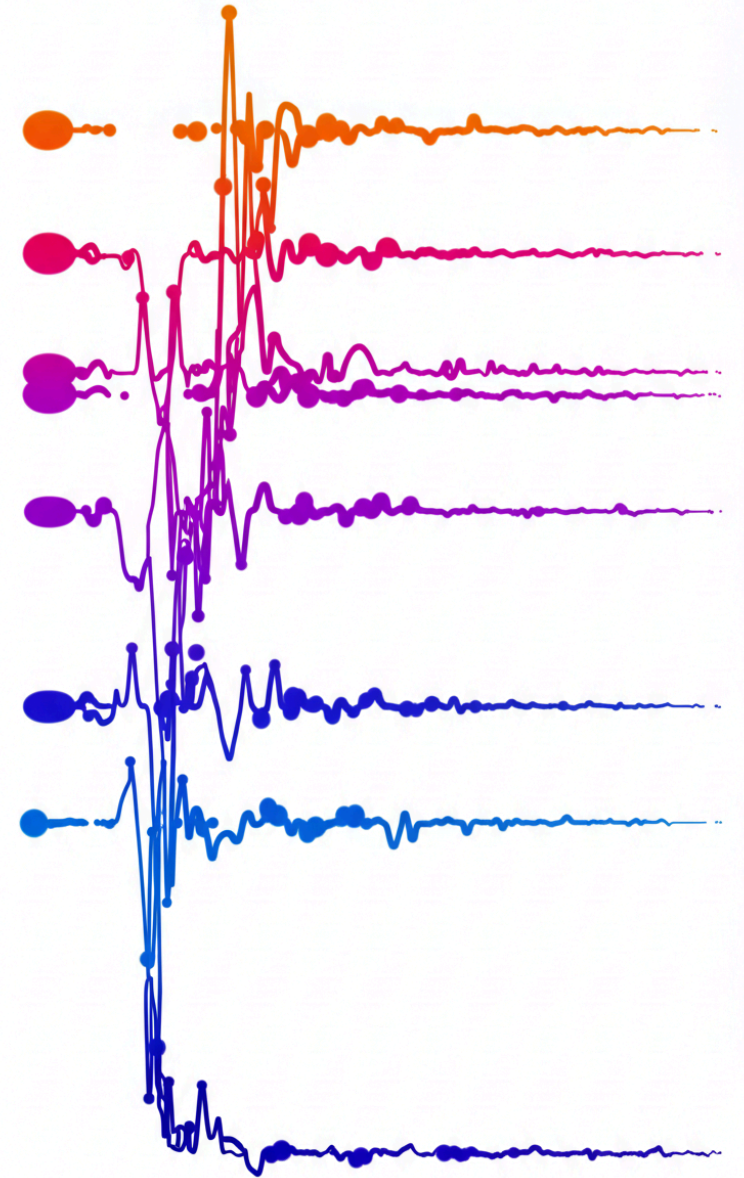
### Asistente de Docencia

Sergio Julian Zona Moreno



# Redes Neuronales Recurrentes (RNN)

## Modelado de Series de Tiempo



# Introducción a las Series Temporales

Las series temporales son secuencias de datos ordenados en tiempo que presentan desafíos únicos para las redes neuronales tradicionales.

## Características de las Series Temporales

- **Dependencia temporal:** El orden de los datos importa
- **Patrones estacionales:** Ciclos que se repiten periódicamente
- **Tendencias:** Dirección general a largo plazo

## ¿Por qué necesitamos modelos especiales?

Las redes neuronales tradicionales (feedforward) no pueden capturar estas dependencias temporales.



## Limitaciones de las Redes Neuronales Tradicionales

Las redes neuronales feedforward (MLP) tienen limitaciones significativas cuando trabajan con datos secuenciales:

- No mantienen estado interno o "memoria"
- Asumen independencia entre observaciones
- Procesan entradas de longitud fija
- No capturan dependencias temporales

Para analizar series temporales, necesitamos redes que puedan "recordar" información previa y procesar secuencias de longitud variable.

# Redes Neuronales Recurrentes (RNN)

Las RNN son redes diseñadas específicamente para procesar datos secuenciales como series temporales.

## Idea Clave

A diferencia de las redes tradicionales, las RNN tienen "memoria" a través de conexiones recurrentes que permiten que la información persista.

## Ventajas

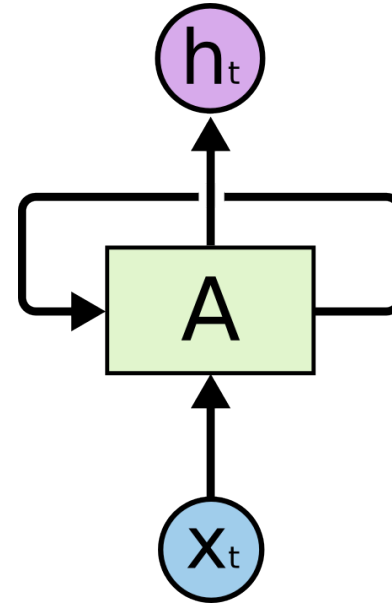
- Pueden procesar secuencias de longitud variable
- Mantienen contexto a lo largo del tiempo
- Comparten parámetros en cada paso temporal



# Arquitectura RNN: Componentes

## Elementos Principales

- **Entrada ( $x_{(t)}$ ):** Vector de datos en tiempo  $t$ 
  - *Ejemplo: [precio, volumen, volatilidad]*
- **Estado oculto ( $h_{(t)}$ ):** La "memoria" de la red
  - *Almacena información contextual*
- **Salida ( $y_{(t)}$ ):** Predicción en tiempo  $t$ 
  - *Ejemplo: precio futuro, probabilidad*



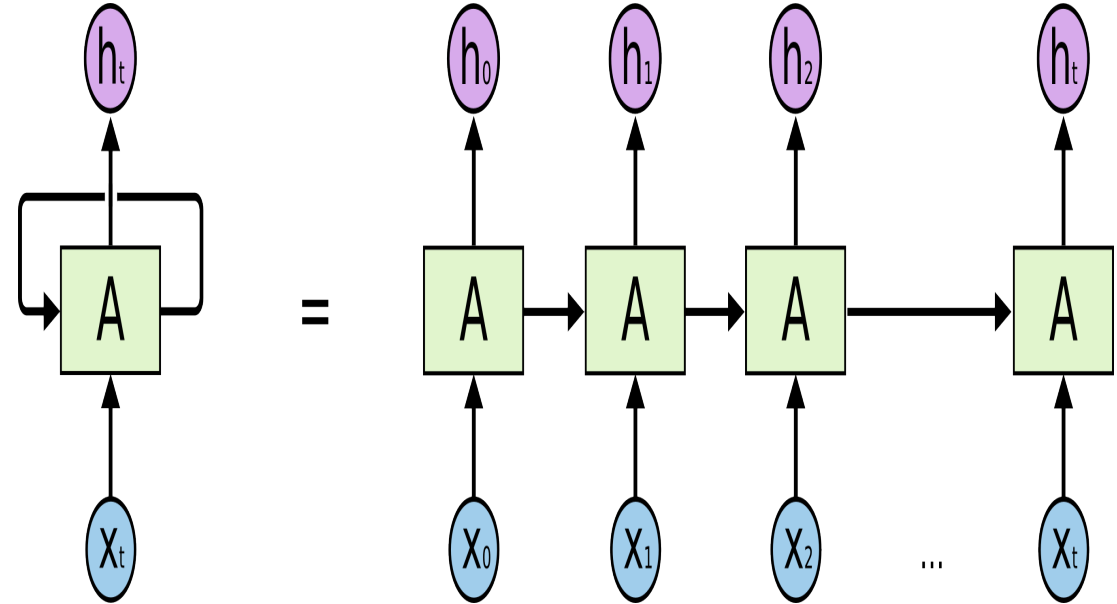
Tomado de: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

## Funcionamiento de la RNN

### Proceso Secuencial

En cada paso temporal, la RNN:

1. **Recibe** una nueva entrada  $x_{(t)}$
2. **Combina** esta entrada con el estado anterior  $h_{(t-1)}$
3. **Actualiza** el estado oculto  $h_{(t)}$
4. **Genera** una predicción  $y_{(t)}$



Tomado de: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

La misma función se aplica en cada paso, permitiendo procesar secuencias de cualquier longitud

# Matemática de las RNN: Simple pero Potente

## Ecuaciones Fundamentales

Estado oculto:

$$h_t = \tanh(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$$

Salida:

$$y_t = W_{hy}h_t + b_y$$

Donde:

- $W_{xh}$ : Pesos entrada-oculta
- $W_{hh}$ : Pesos recurrentes
- $W_{hy}$ : Pesos oculta-salida
- $b_h, b_y$ : Vectores de sesgo



## Matemática de las RNN: Simple pero Potente

### Ecuaciones Fundamentales

Estado oculto:

$$h_t = \tanh(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$$

Salida:

$$y_t = W_{hy}h_t + b_y$$

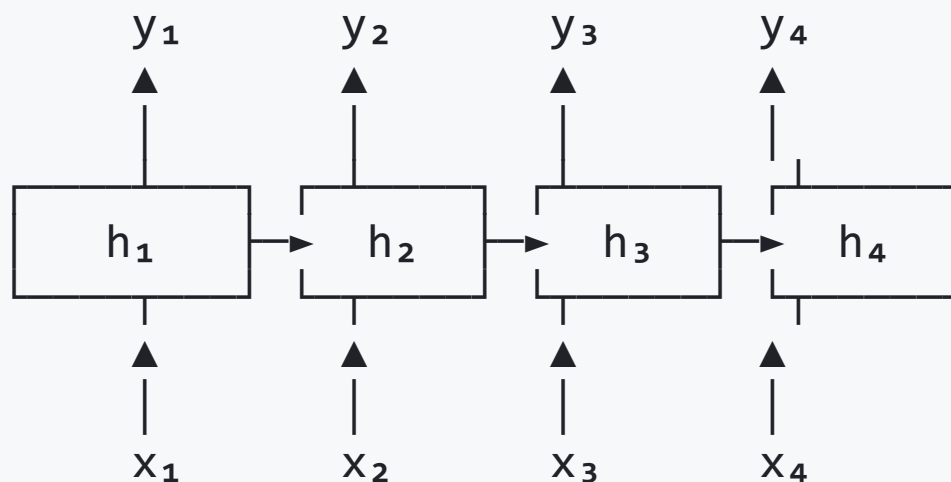
 **Dato importante:**

Los mismos parámetros se reutilizan en cada paso temporal.



## Visualizando el Despliegue en el Tiempo

Una RNN puede visualizarse "desplegada" a través del tiempo:



### Observaciones Clave

- La misma función se aplica en cada paso temporal
- Los mismos parámetros se comparten en toda la secuencia
- El estado  $h_{(t)}$  depende de  $h_{(t-1)}$  y  $x_{(t)}$



# Preparación de Datos para Series Temporales



## División Train-Test

- **Objetivo:** Evaluar la capacidad predictiva del modelo
- **Método estándar:** División cronológica (no aleatoria)
  - **80%** datos para entrenamiento
  - **20%** datos para prueba

*División cronológica de datos en series temporales*

```
# Ejemplo de código
n = len(datos)
train_size = int(n * 0.8)

train_data = datos[:train_size]
test_data = datos[train_size:]
```

A diferencia de otros problemas de ML, en series temporales la división debe respetar el orden cronológico.

# Ventanas Deslizantes: El Corazón del Entrenamiento

## ¿Qué son las ventanas deslizantes?

Técnica para convertir series temporales en un formato adecuado para RNNs:

1. **Ventana de entrada:** n valores consecutivos
2. **Valor objetivo:** Siguiendo valor a predecir
3. **Deslizamiento:** Mover la ventana un paso a la vez

## Ejemplo con PIB Trimestral

Datos: [8.3, 8.6, 8.8, 9.1, 9.4, 9.8, 10.0, 10.3]

# Ventana tamaño 4

X1 = [8.3, 8.6, 8.8, 9.1] → Y1 = 9.4

X2 = [8.6, 8.8, 9.1, 9.4] → Y2 = 9.8

X3 = [8.8, 9.1, 9.4, 9.8] → Y3 = 10.0

X4 = [9.1, 9.4, 9.8, 10.0] → Y4 = 10.3

*Ilustración del proceso de ventana deslizante*

# Métricas de Evaluación para Series Temporales

## Métricas de Error Comunes

- MSE (Error Cuadrático Medio)

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

- RMSE (Raíz del Error Cuadrático Medio)

$$RMSE = \sqrt{MSE}$$

- MAE (Error Absoluto Medio)

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

Estas métricas se calculan comparando las predicciones del modelo con los valores reales en el conjunto de prueba.



## Proceso de Evaluación



### Consideraciones Importantes

- **Evaluación en datos no vistos:**  
El modelo entrenado con el 80% de los datos genera predicciones sobre el 20% restante
- **Conjunto de prueba como referencia:**  
Se comparan las predicciones con los valores reales del conjunto de prueba



### Estrategias de Predicción

- **Horizonte de predicción:**
  - **One-step:** Predecir solo el siguiente valor
  - **Multi-step:** Predecir varios valores futuros
- **Comparación con baseline:**
  - Método ingenuo (último valor)
  - Promedio móvil
  - ARIMA

# Proceso Completo de Modelado

## Flujo de Trabajo

### 1. Preparación:

- División train-test (80%-20%)
- Creación de ventanas deslizantes
- Normalización de datos

### 2. Entrenamiento:

- Alimentar ventanas al modelo
- Ajustar pesos para minimizar error
- Validación con datos no vistos

### 3. Evaluación:

- Predicción secuencial en test
- Cálculo de métricas (RMSE, MAE)
- Comparación con modelos baseline

### 4. Producción:

- Uso de toda la serie para entrenamiento final
- Predicciones sobre nuevos datos
- Monitoreo y reentrenamiento



# El Problema del Gradiente Desvaneciente

## ¿Qué ocurre?

Al entrenar RNNs con backpropagation, los gradientes:

- Se multiplican repetidamente por  $W_{hh}$
- Si los valores propios de  $W_{hh} < 1$ , el gradiente se desvanece
- Si los valores propios de  $W_{hh} > 1$ , el gradiente explota

## Consecuencias

- **Memoria a corto plazo:** La red olvida información pasada
- **Dificultad para capturar dependencias a largo plazo**
- **Entrenamiento inestable**

## El Problema del Gradiente Desvaneciente

### Implicaciones

- En secuencias largas, los gradientes tienden a desvanecerse, limitando la capacidad de aprendizaje.
- A medida que retropropagamos en el tiempo, los gradientes se vuelven tan pequeños que la red deja de aprender efectivamente.

## LSTM: La Solución al Gradiente Desvaneciente


Las **Long Short-Term Memory** (LSTM) son un tipo avanzado de RNN diseñadas para:


- Mantener información por largos períodos de tiempo
- Evitar el problema del gradiente desvaneciente
- Aprender qué información recordar y olvidar


### Componente clave

Las LSTM utilizan **compuertas** (gates) para controlar el flujo de información a través del tiempo.

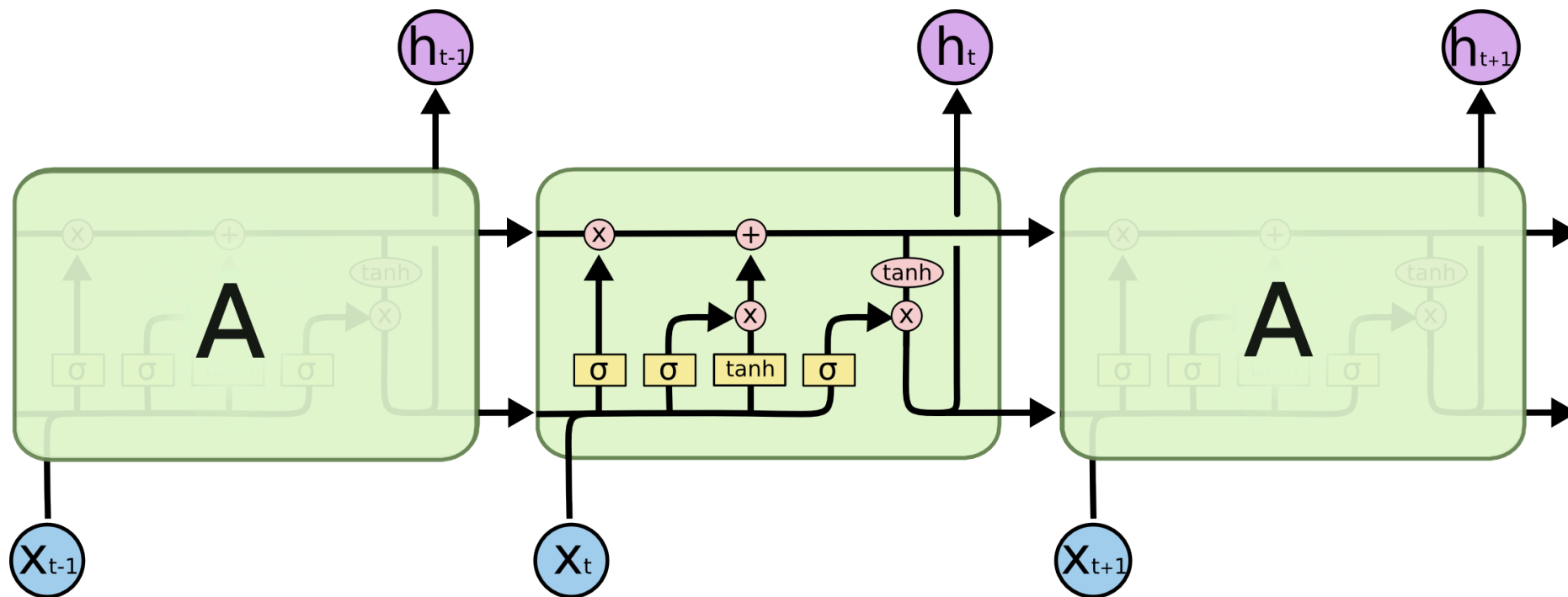
## ¿Qué son las compuertas?

 Las **compuertas** son **mecanismos matemáticos** que **controlan el flujo de información** a través de la red neuronal. Técnicamente, son **funciones diferenciables** que transforman los datos de entrada mediante **pesos entrenables** y una **activación sigmoide**, produciendo valores entre 0 y 1 que determinan **cuánta información puede pasar**.

 Estos componentes **se entrenan durante el proceso de backpropagation**, **NO** son **hiperparámetros** que se configuren manualmente.

 La red **aprende automáticamente a abrir o cerrar** estas "**válvulas inteligentes**" según el contexto, permitiendo que la LSTM **decida qué información recordar, actualizar u olvidar** en cada paso temporal.

## LSTM: Explicación Visual

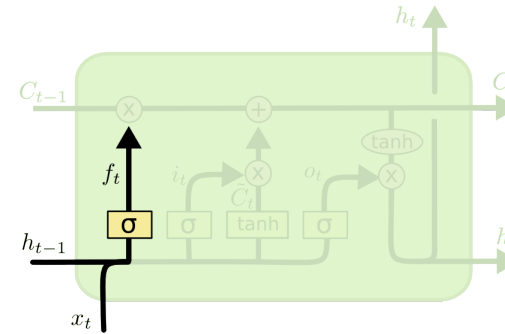


Tomado de: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

Arquitectura de una celda LSTM con sus compuertas de control

## 🔍 Los tipos de compuertas LSTM

1. **Compuerta de olvido ( $f_{(t)}$ ):**  
Decide qué información descartar
2. **Compuerta de entrada ( $i_{(t)}$ ):**  
Determina qué nueva información almacenar
3. **Compuerta de salida ( $o_{(t)}$ ):**  
Controla qué información se transmite al siguiente paso
4. **Celda de memoria ( $c_{(t)}$ ):**  
Almacena información a largo plazo



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

Tomado de: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

Las compuertas utilizan funciones sigmoides (valores entre 0 y 1) para controlar el flujo de información

## Matemática de las LSTM: Ecuaciones Fundamentales

1. Compuerta de olvido:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

2. Compuerta de entrada:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

3. Actualización de la celda de memoria:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

4. Compuerta de salida y estado oculto:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$



# Analogía: LSTM como Analista Financiero

## 1 Compuerta de Olvido

- Como descartar informes obsoletos
- *"El mercado ha cambiado, estos datos ya no son relevantes"*
- Ejemplo: Ignorar patrones pre-pandemia en análisis actuales

## 2 Compuerta de Entrada

- Como archivar información crucial
- *"Esta alza de tasas es importante, debo guardarla"*

## 3 Celda de Memoria

- Como una caja fuerte de información
- *"Aquí guardo las tendencias importantes a largo plazo"*
- Ejemplo: Mantener patrones de ciclos económicos

## 4 Compuerta de Salida

- Como preparar un informe actual
- *"Para la predicción de hoy, selecciono esta información"*

## GRU: La Alternativa Eficiente

Las **Gated Recurrent Unit** (GRU) son una variante más simple de las LSTM, introducidas por Cho et al. en 2014.

### Características Principales

- Fusionan las compuertas de olvido y entrada
- Eliminan la celda de memoria separada
- Menos parámetros que LSTM
- Rendimiento similar en muchas tareas
- Entrenamiento más rápido y eficiente

## Matemática de las GRU: Simplicidad Efectiva

Las GRU utilizan sólo dos compuertas para controlar el flujo de información:

1. Compuerta de actualización ( $z_{(t)}$ ):

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$$

2. Compuerta de reinicio ( $r_{(t)}$ ):

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$$

3. Candidato a nuevo estado ( $\hat{h}_{(t)}$ ):

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t] + b)$$

4. Estado oculto final:

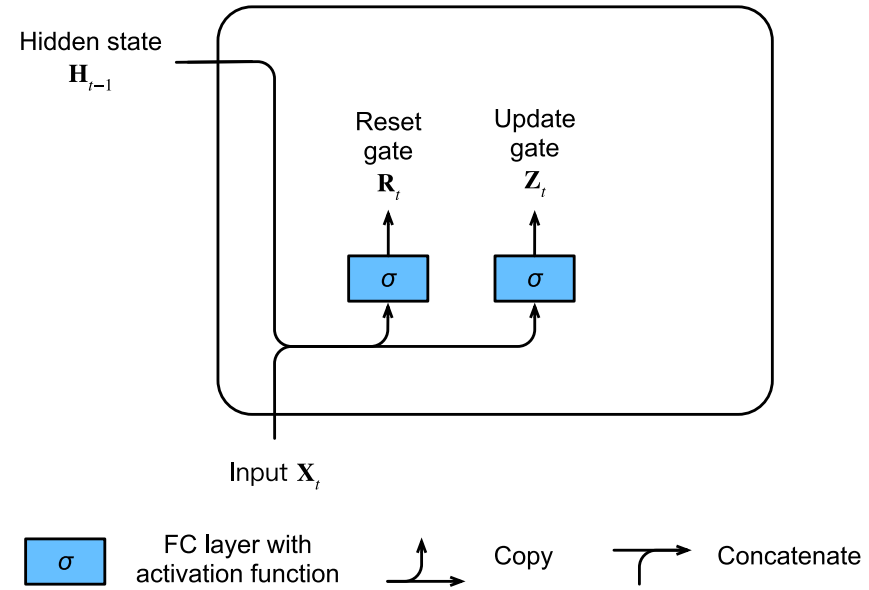
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

# 🔄 Analogía para entender las compuertas GRU

## 📖 Primera compuerta

### 1. Compuerta de Actualización ( $z_{(t)}$ )

- Decide cuánto confiar en información nueva vs. antigua
- *"¿Sigo mi estrategia anterior o doy peso al anuncio reciente?"*
- Si hay un anuncio importante → prioriza la nueva información
- Si es ruido de mercado → se aferra a su estrategia previa



Tomado de: [https://d2l.ai/chapter\\_recurrent-neural-networks/gru.html](https://d2l.ai/chapter_recurrent-neural-networks/gru.html)

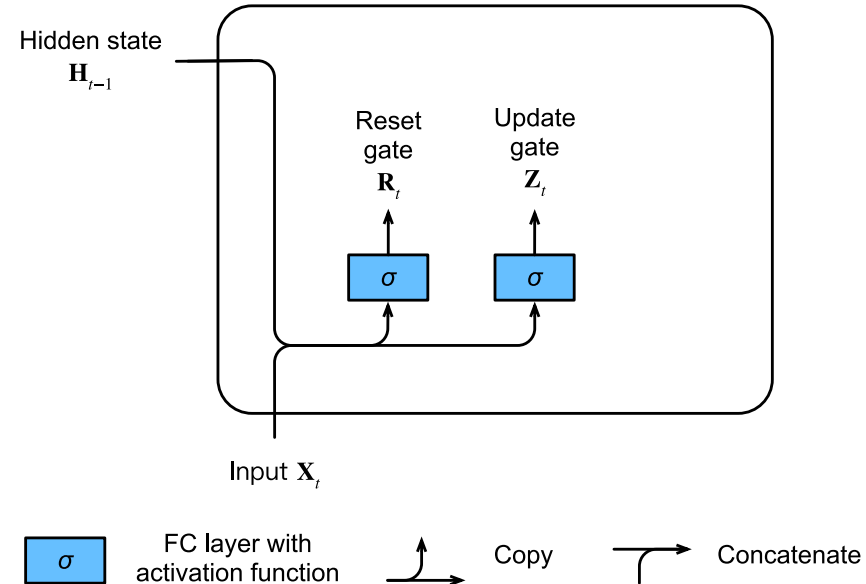
La GRU es como un operador de mercado ágil: sabe cuándo cambiar de estrategia y cuándo ignorar el ruido.

# 🔄 Analogía para entender las compuertas GRU

## 📖 Segunda compuerta

### 2. Compuerta de Reinicio ( $r_{(t)}$ )

- Decide cuándo "empezar de cero" tras un cambio de paradigma  
-- "*¿Este cambio en política monetaria invalida mis análisis previos?*"



Tomado de: [https://d2l.ai/chapter\\_recurrent-neural-networks/gru.html](https://d2l.ai/chapter_recurrent-neural-networks/gru.html)

La GRU es como un operador de mercado ágil: sabe cuándo cambiar de estrategia y cuándo ignorar el ruido.



## Comparativa de Modelos

Basado en experiencias de predicción de mercados financieros

Desempeño relativo:

Modelo	Precisión	Velocidad	Memoria	Recursos necesarios
RNN	★ ★	★ ★ ★ ★ ★	★ ★	★ ★
LSTM	★ ★ ★ ★ ★	★ ★ ★	★ ★ ★ ★ ★	★ ★ ★ ★ ★
GRU	★ ★ ★ ★	★ ★ ★ ★	★ ★ ★ ★	★ ★ ★



## Casos de uso ideales

### RNN Simple

- Mercados estables sin interrupciones
- Predicciones a muy corto plazo
- Sistemas con recursos limitados
- Procesamiento en tiempo real sencillo
- Alto volumen de datos simples

### LSTM/GRU

- Mercados volátiles o con interrupciones
- Predicciones a medio/largo plazo
- Patrones económicos complejos
- Detección de cambios de régimen
- Análisis de crisis financieras

**Ejemplo real:** Durante la crisis del 2008, los modelos LSTM predijeron correctamente patrones de recuperación basándose en crisis anteriores, mientras que las RNN simples fallaron consistentemente.



## Recursos del Curso

## Plataformas y Enlaces Principales

### GitHub del curso

 [github.com/CamiloVga/IA\\_Aplicada](https://github.com/CamiloVga/IA_Aplicada)

### Asistente IA para el curso

 [Google Notebook LLM](#)