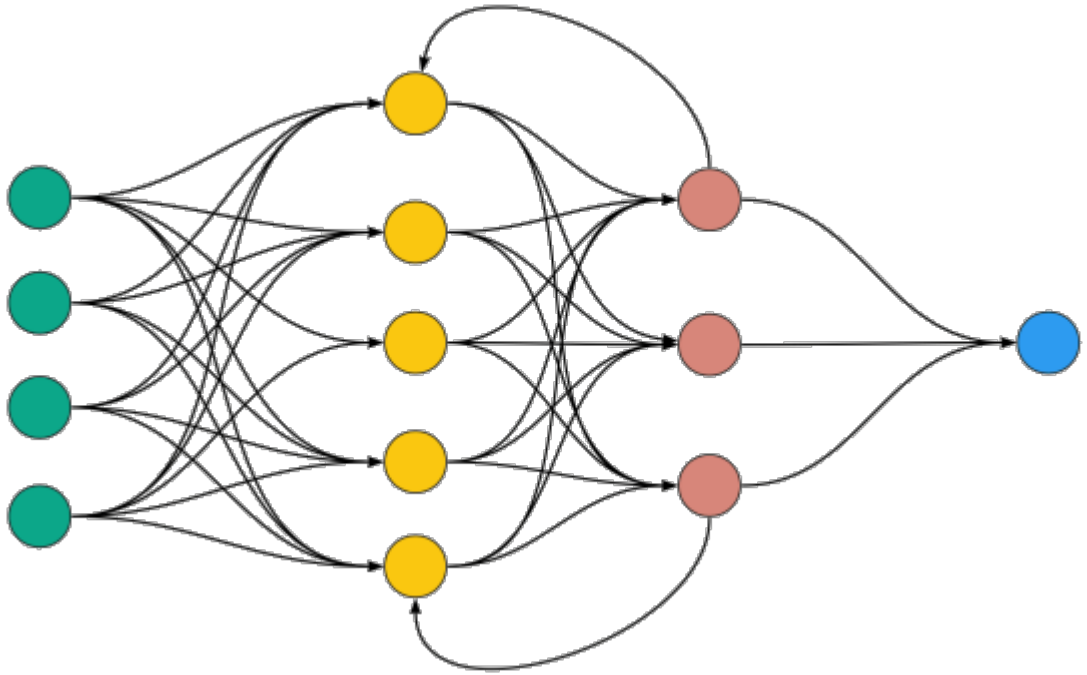


Redes Neuronales Recurrentes RNN y Long Short Term Memory LSTM

PhD(e). Jonnatan Arias Garcia – jonnatan.arias@utp.edu.co – jariasg@uniquindio.edu.co

PhD. David Cardenas peña - dcardenasp@utp.edu.co

PhD. Hernán Felipe Garcia - hernanf.garcia@udea.edu.co

Contenido

- **RNN**
- **Sequence Learning**
- **¿Como funciona una RNN?**
- **Tipos de RNN**
- **RNN Vs. CNN**
- **Variantes RNN**
  1. **Vanilla RNN**
  2. **LSTM**
  3. **GRU**
- **Limitaciones**
- **Aplicaciones**

RNN

Modelo de aprendizaje profundo entrenado para procesar y convertir una entrada de datos secuencial en una salida de datos secuencial específica. (Sequence learning)

Los datos secuenciales: como palabras, oraciones o datos de serie temporal, en los que los componentes secuenciales se interaccionan en función de reglas semánticas y sintácticas complejas.

Una RNN es un sistema que imita la forma en que los humanos realizan conversiones de datos secuenciales.

Las RNN están siendo reemplazadas en gran medida por la inteligencia artificial (IA) basada en transformadores y modelos de lenguaje de gran tamaño (LLM), que son mucho más eficientes en el procesamiento secuencial de datos.

Sequence Learning

El aprendizaje secuencial es el estudio de algoritmos de machine learning diseñados para datos secuenciales.

- Suele aplicarse en sistemas dinámicos discretos basados en delay steps.

- También como un modelo de lenguaje siendo el mas utilizando para etiquetado secuencial y traducciones

¿Cómo funciona una RNN? (memoria)

- Pasan datos secuenciales a las capas ocultas paso a paso. Sin embargo, también tienen un flujo de trabajo *recurrente* o en bucle: la capa oculta puede recordar y utilizar las entradas anteriores para realizar predicciones futuras en un componente de memoria a corto plazo. Utiliza la entrada actual y la memoria almacenada para predecir la siguiente secuencia.

Por ejemplo, consideremos la secuencia: *La manzana es roja.* Quiere que la RNN prediga *roja* cuando reciba la secuencia de entrada que es *la manzana es*. Cuando la capa oculta procesa la palabra *manzana*, guarda una copia en su memoria. Luego, cuando ve la palabra *es*, recuerda *manzana* de su memoria y entiende la secuencia completa: *la manzana es* para el contexto.

A continuación, puede predecir *roja* para mejorar la precisión. Esto hace que las RNN sean útiles en el reconocimiento de voz, la traducción automática y otras tareas de modelado de idiomas.

¿Cómo funciona una RNN? (BPTT)

Utilizan retropropagación en el tiempo (BPTT) para calcular el error del modelos y ajustar los pesos.

La BPTT revierte la salida al paso de tiempo anterior y recalcula la tasa de error.

De esta manera, puede identificar qué estado oculto de la secuencia está causando un error significativo y reajustar el peso para reducir el margen de error.

Tipos de RNN

one to one

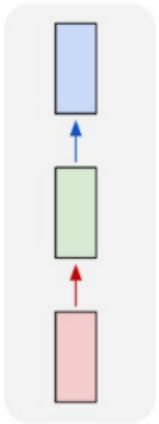


Image Classification
Vanilla RNN

one to many

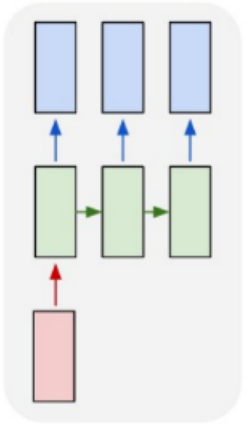
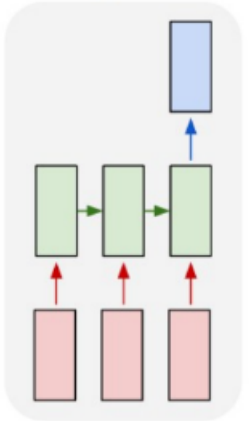


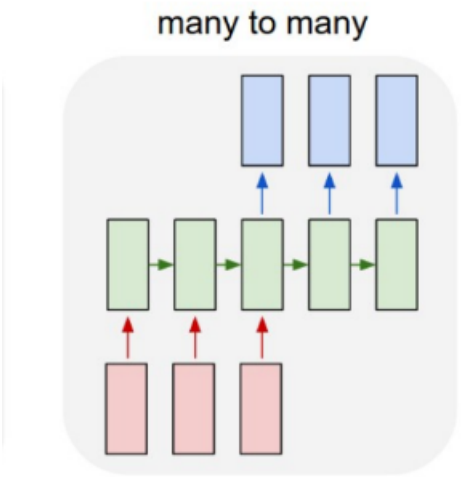
Image Captioning

many to one

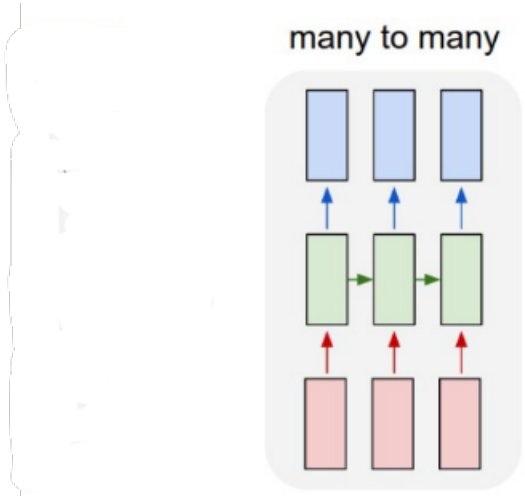


Sentiment Classification

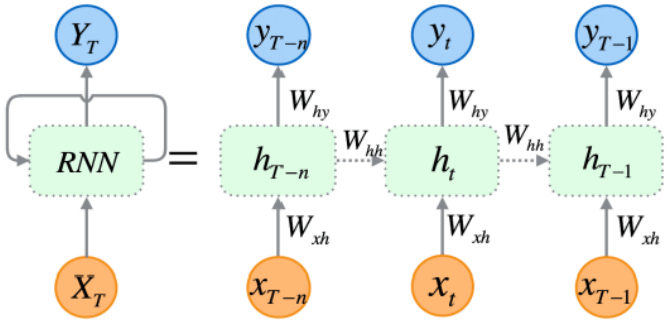
Tipos de RNN



Machine Translation



Video Labeling (frame)



RNN & CNN

CNN son buenas extrayendo información espacial

Aplicable a sparse data

Mas poderosa que RNN

Requiere costo computacional finito y estandarizado con la entrada

Trabaja en capaz y tiempo

Las RNN capturan dependencias en datos secuenciales

Aplicable a data secuencias

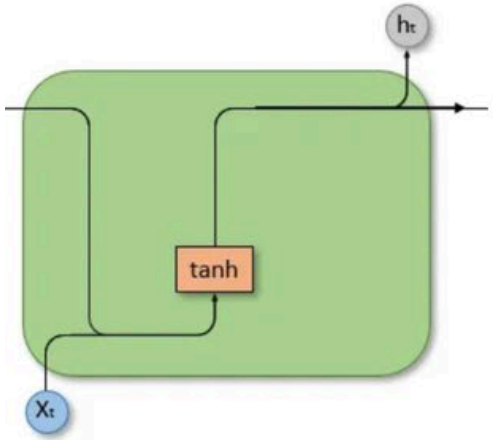
Menos capacidad pero mas enfocada

Permite entradas de diferentes tamaños

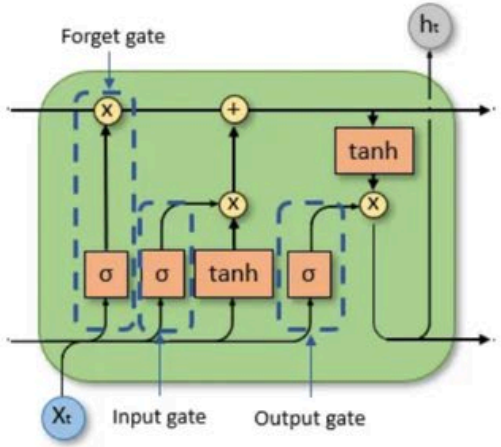
Trabaja en bucles

Variantes

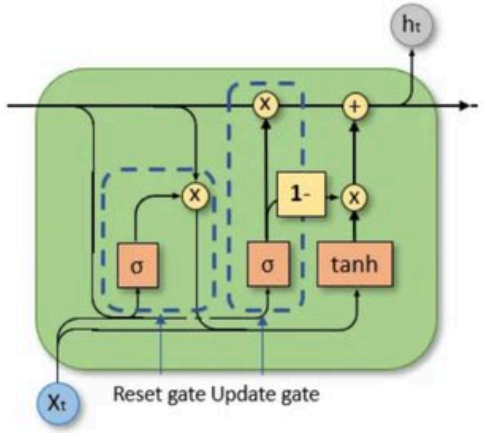
RNN



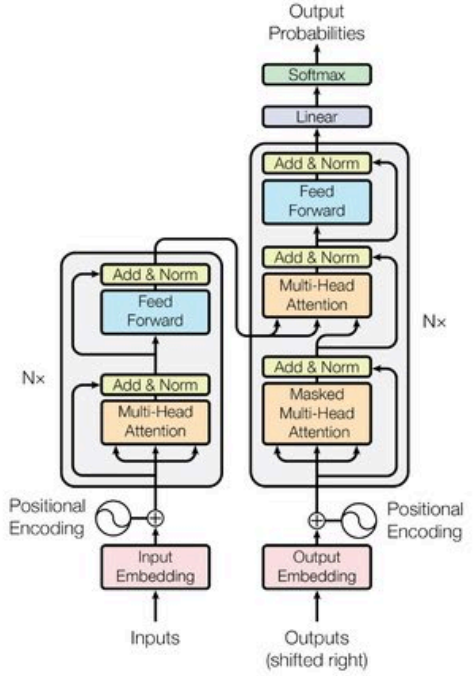
LSTM



GRU



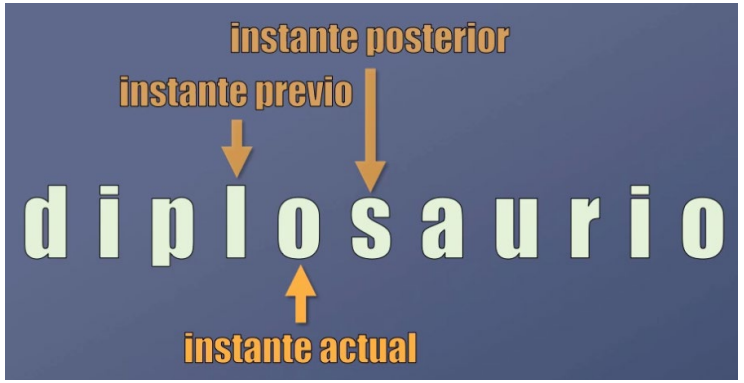
Transformers



RNN

Supongamos que entrenamos una red con nombres de dinosaurios, el modelo aprenderá a generar nombres carácter por carácter.

Sabemos que nuestra red se centra en analizar el comportamiento previo y posteriores de tiempo.



Asi: por ejemplo en la palabra “diplosaurio” para generar la primer “o”, la red debería tener en cuenta la “l” anterior.

RNN

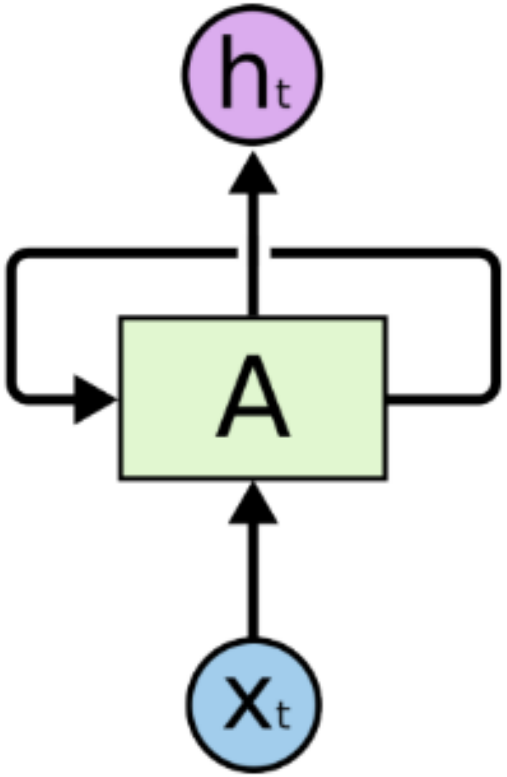
Las redes recurrentes son buenas en esto, pues tiene memoria de instantes previos lo que las hace ideales para analizar secuencias.

Estructura:

1. Los instantes de tiempo van ligados al instante de aparición, por ejemplo en “diplosaurio” tendríamos 11 instantes de tiempo

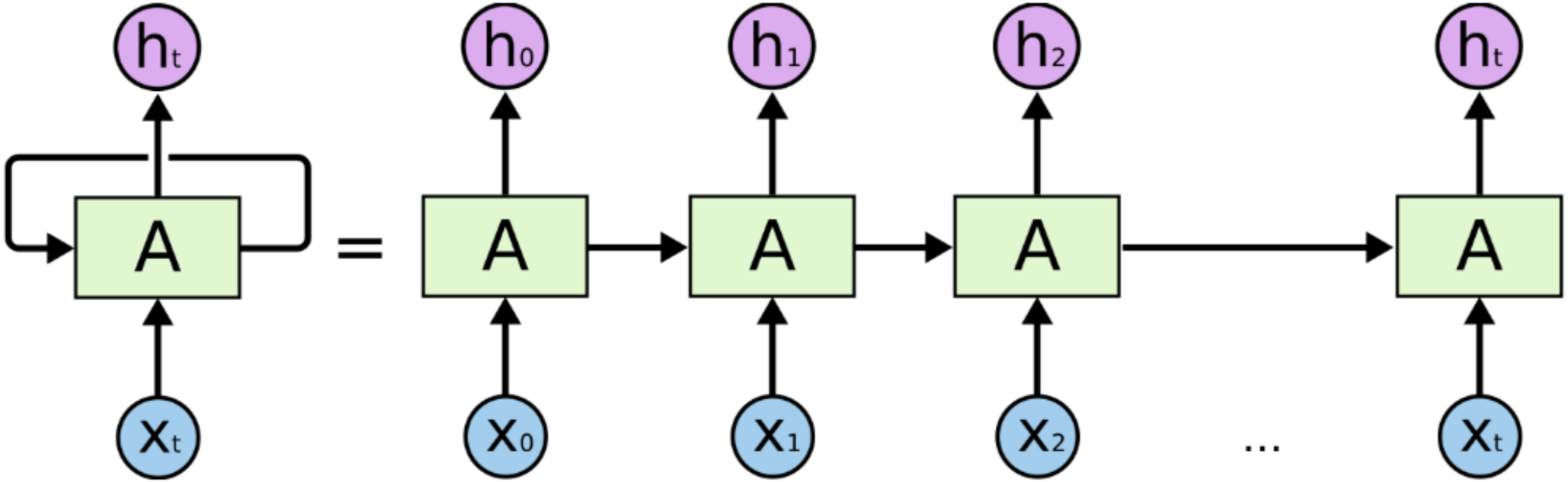
RNN

$$h_t \in \mathbb{R}^N, x_t \in \mathbb{R}^M \rightarrow h_t = f_W(h_{t-1}, x_t), y_t = g_{W_o}(h_t)$$



RNN

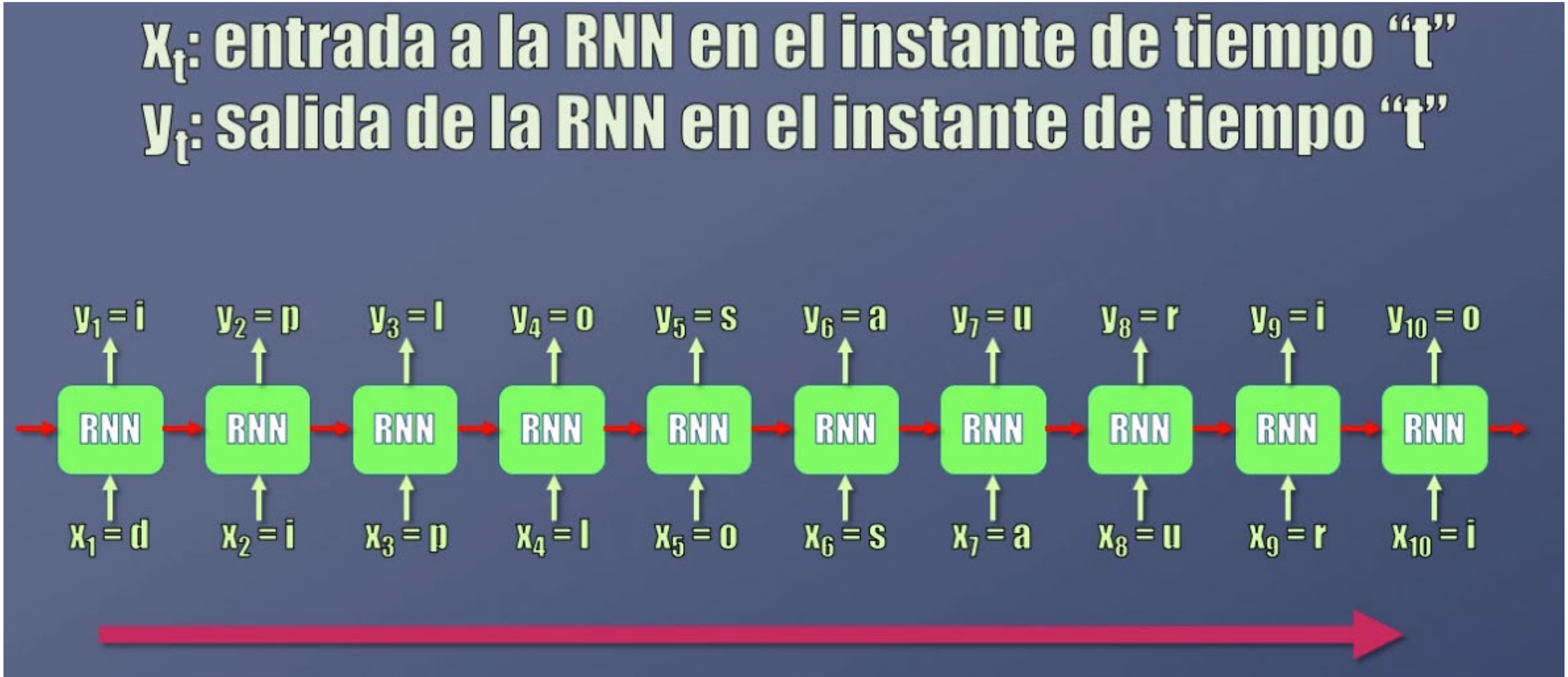
$$h_t \in \mathbb{R}^N, x_t \in \mathbb{R}^M \rightarrow h_t = f_W(h_{t-1}, x_t), y_t = g_{W_o}(h_t)$$



RNN

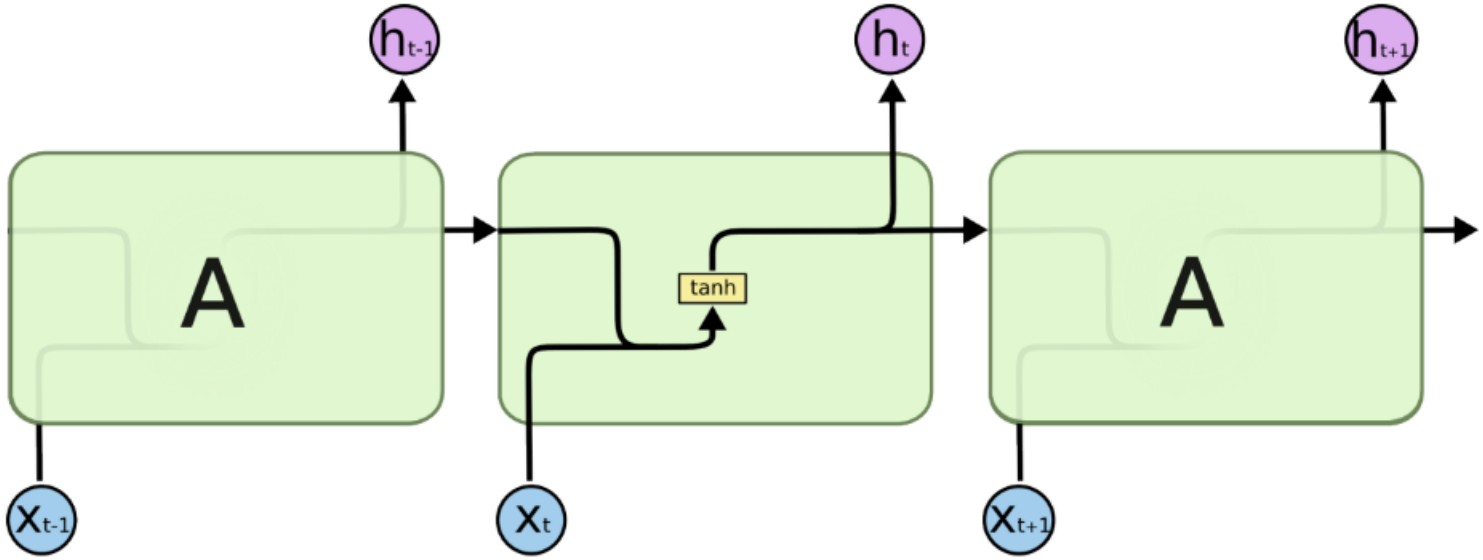
$$y_t = h_t$$

Nota: los bloques RNN es solo uno que se itera



Vanilla RNN

$$h_t \in \mathbb{R}^N, x_t \in \mathbb{R}^M \rightarrow h_t = \tanh(W_{hh}h_{t-1} + W_{hx}x_t), y_t = \mathbf{W}_o h_t$$



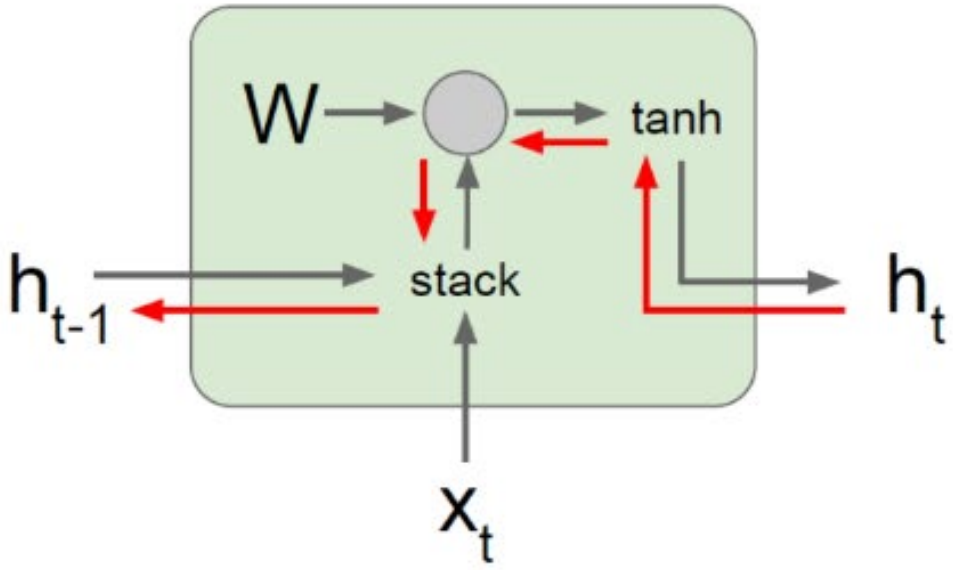
La red tiene dos entradas y dos salidas:

-Dato actual x_t -Activación anterior $h_{t-1}{}'$

-Predicción actual h_t -Activación actual $h_t{}'$ (hidden state)

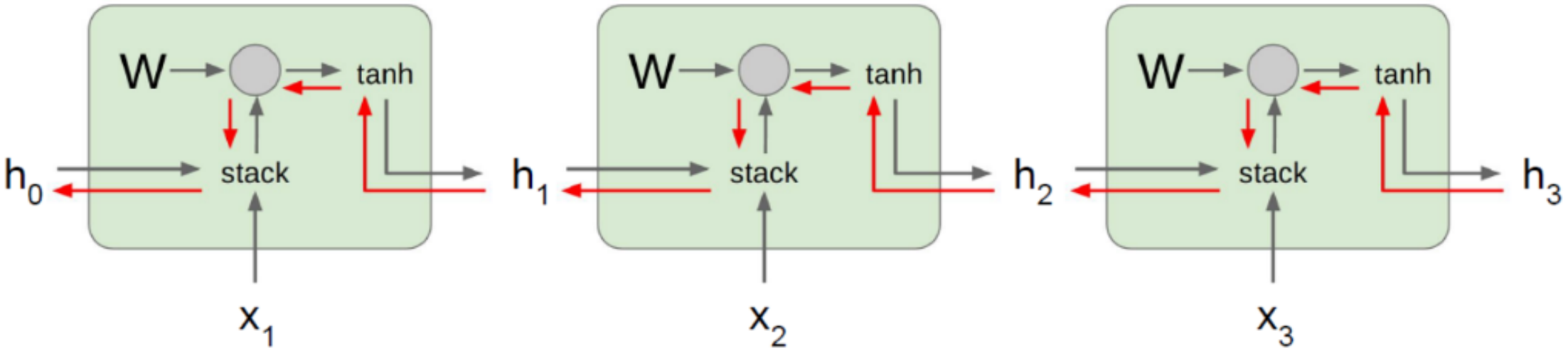
Las activaciones son las encargadas de ser “La memoria de la red”

Vanilla RNN



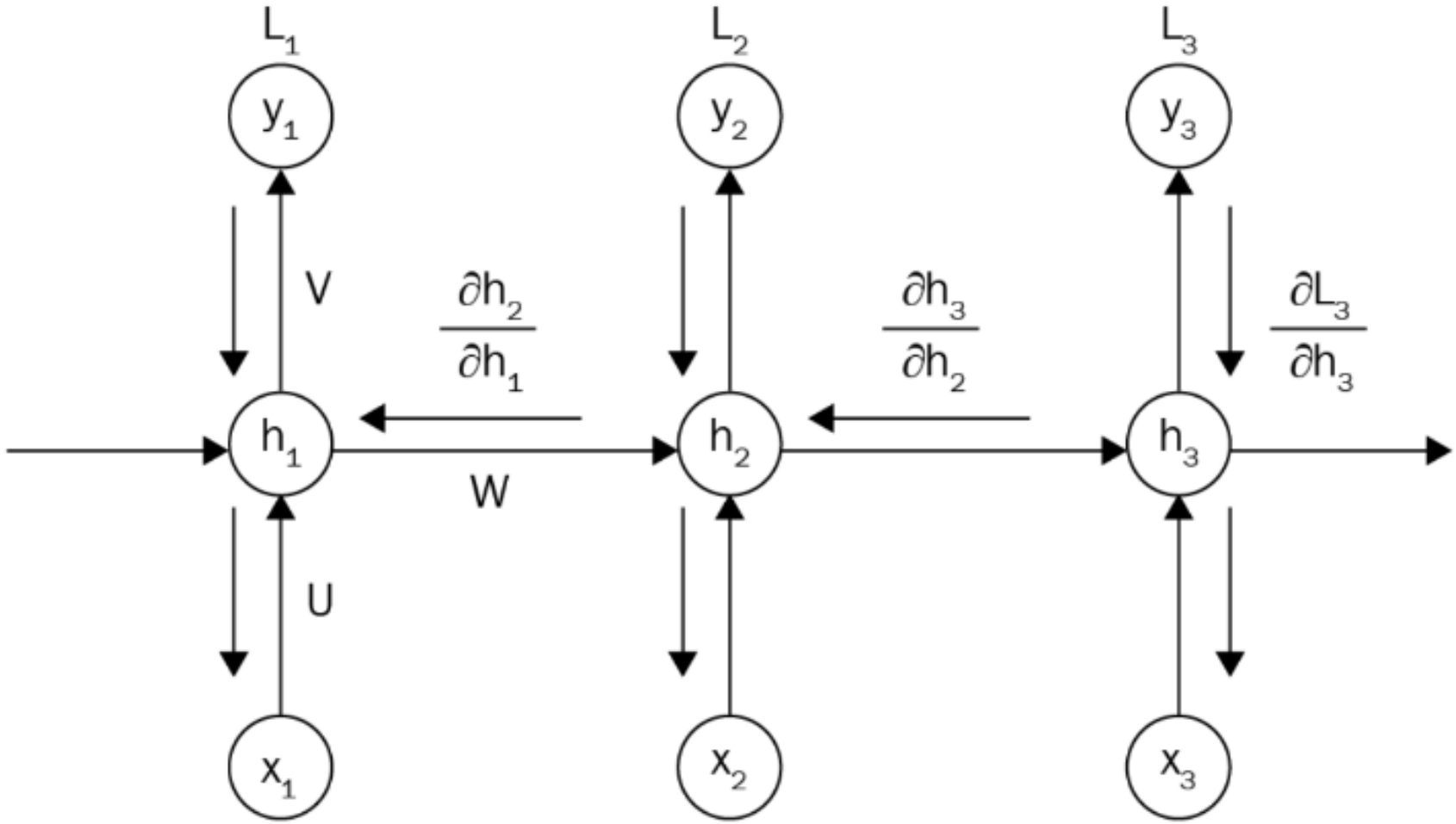
$$
\begin{aligned}
h_t &= \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \\
&= \tanh\left((W_{hh} \quad W_{hx})\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right) \\
&= \tanh\left(W\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)
\end{aligned}
$$

Vanilla RNN



$$h_1 = \tanh(W_{hh}h_0 + W_{hx}x_1)$$
$$h_2 = \tanh(W_{hh}(\tanh(W_{hh}h_0 + W_{hx}x_1) + W_{hx}x_2)$$
$$h_3 = \tanh(W_{hh}(\tanh(W_{hh}\tanh(W_{hh}h_0 + W_{hx}x_1) + W_{hx}x_2) + W_{hx}x_3)$$

Vanilla RNN: Gradientes



Vanilla RNN: Problemas con Gradientes

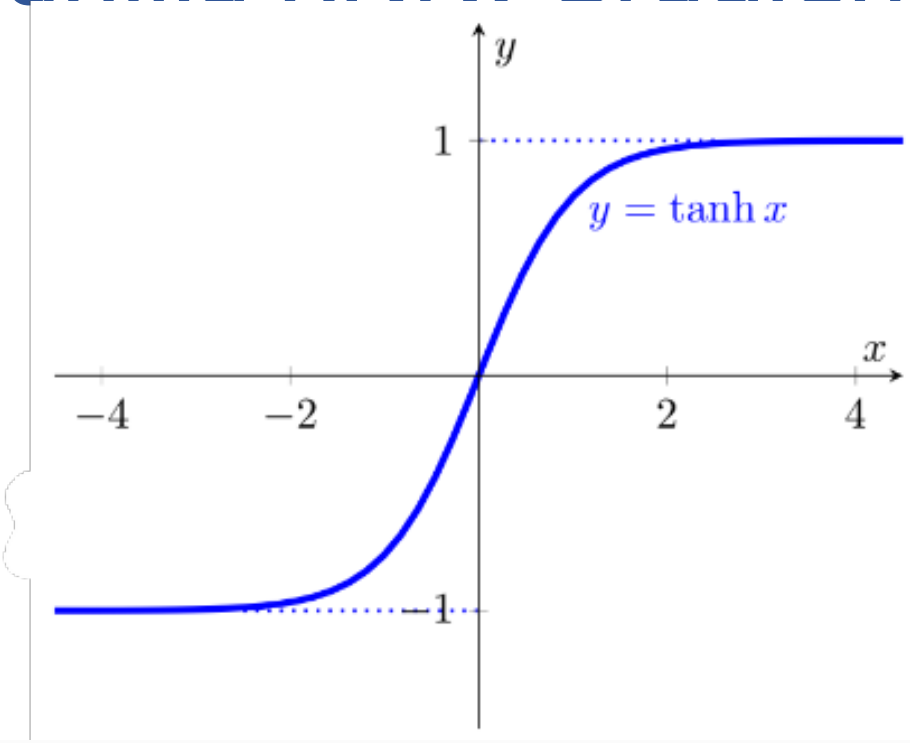
$$\frac{\partial E}{\partial \theta} = \sum_{1 \leq t \leq T} \frac{\partial E_t}{\partial \theta}$$

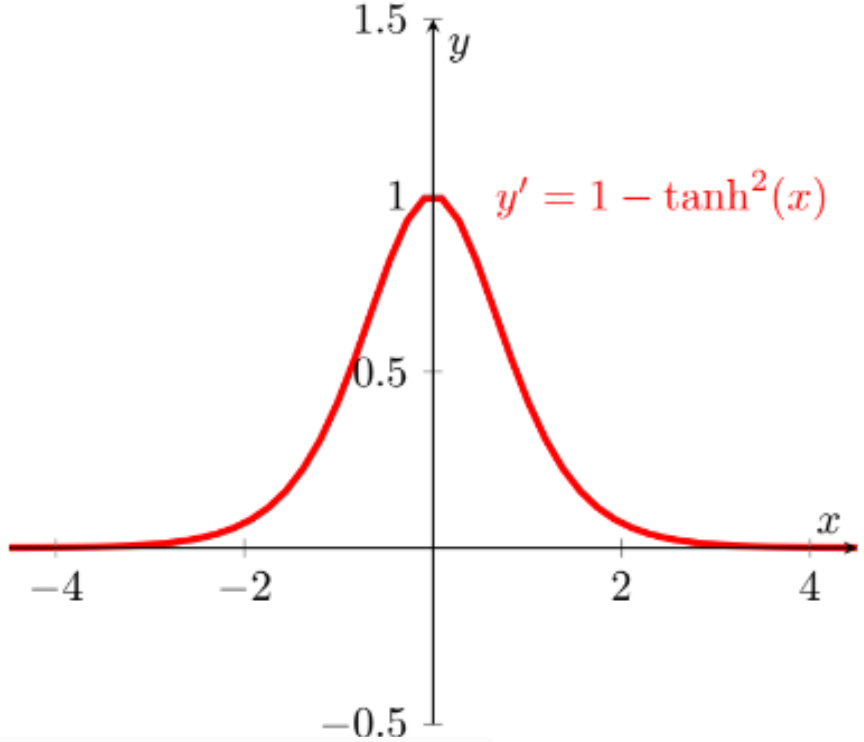
$$\frac{\partial E_t}{\partial \theta} = \sum_{1 \leq k \leq t} \left(\frac{\partial E_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial^+ h_k}{\partial \theta} \right)$$

$$\frac{\partial h_t}{\partial h_k} = \prod_{t \geq i > k} \frac{\partial h_i}{\partial h_{i-1}} = \prod_{t \geq i > k} W_{hh}^\top \operatorname{diag}(\sigma'(h_{i-1}))$$

R. Pascanu - On the difficulty of training recurrent neural networks, 2013.

Vanilla RNN: Gradientes desvanecido



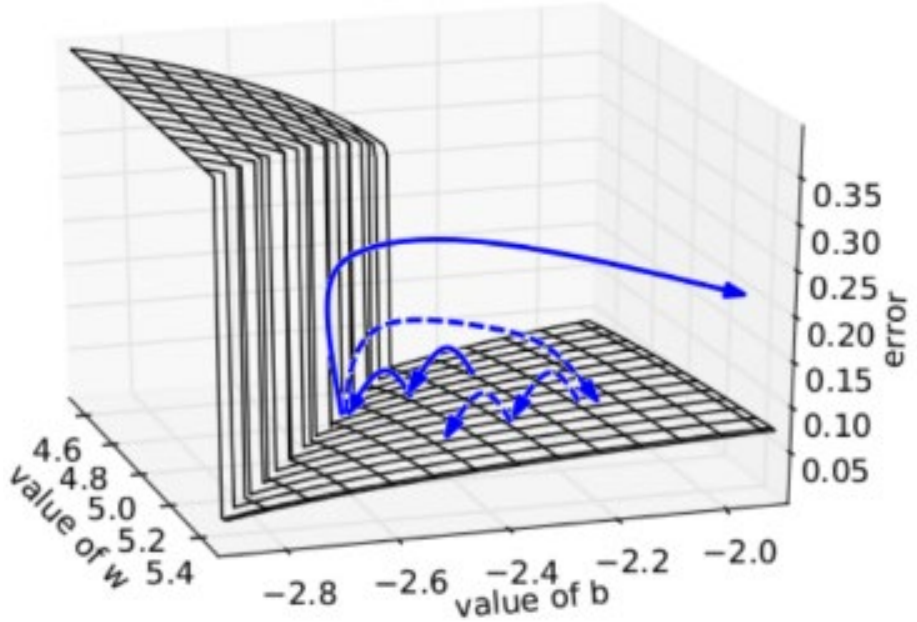


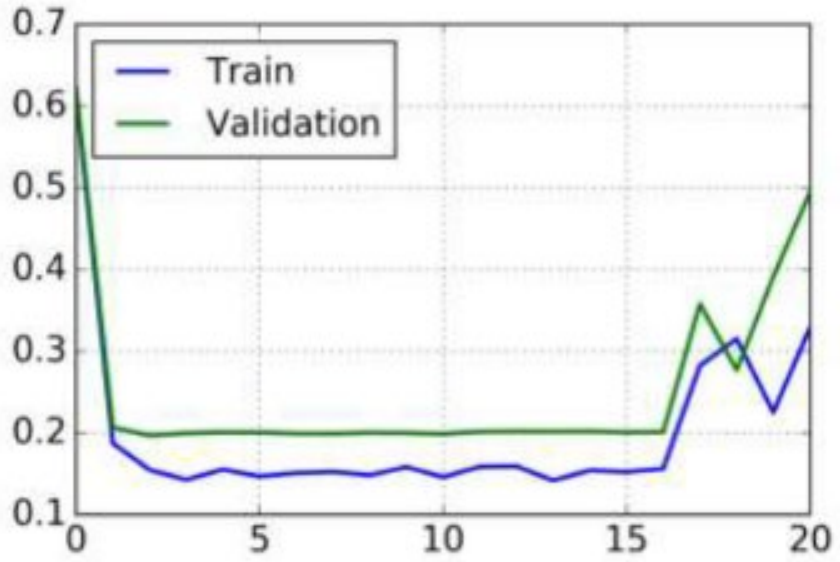
Ocurre cuando los gradientes se hacen nulos al propagarse hacia atrás.

Esto se puede deber a:

- Secuencias de entrada muy largas (disminución exponencial del gradiente)
- Activaciones saturadas: derivada se acerca a cero
- Inicializaciones de pesos inapropiadas: si son muy pequeños puede disminuir rápidamente

Vanilla RNN: Explosión del Gradiente





$$\frac{\partial h_t}{\partial h_k} = \prod_{t>i>k} \frac{\partial h_i}{\partial h_{i-1}} = \prod_{t>i>k} W_{hh}^{\top} \operatorname{diag}(\sigma'(h_{i-1}))$$

Ocurre cuando los gradientes se hacen extremadamente grandes al propagarse hacia atrás.

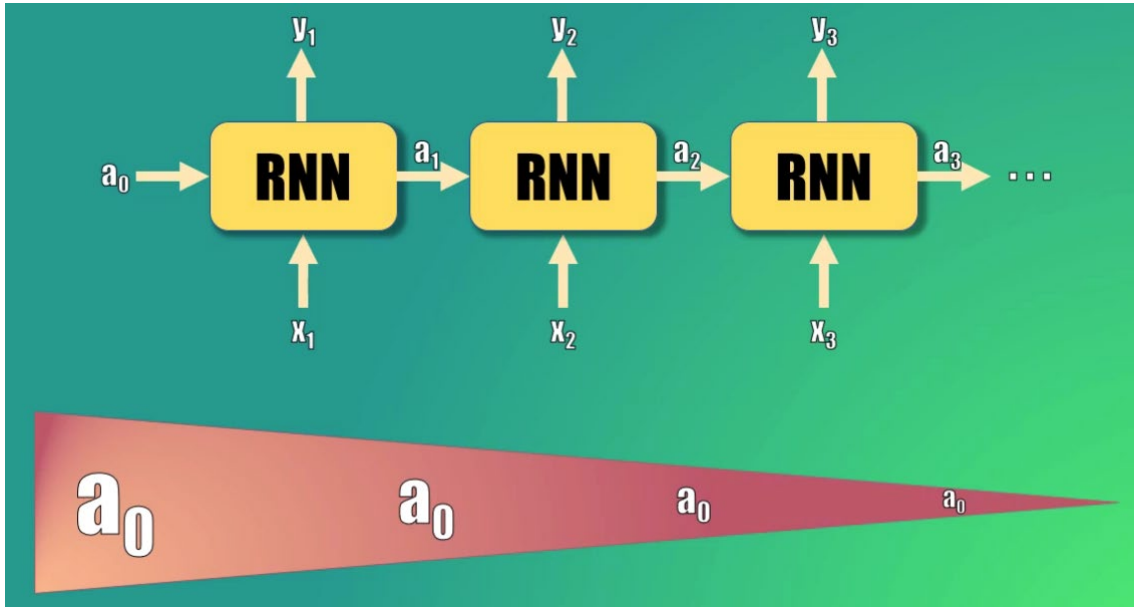
Esto se puede deber a:

- Secuencias de entrada muy largas (aumento exponencial del gradiente)
- Activaciones no lineales: sigmoide o tanh puede hacer que los gradientes se amplifiquen.
- Inicializaciones de pesos inapropiadas: si son muy grandes pueden aumentar rápidamente.

LSTM – Long-Short Term Memory

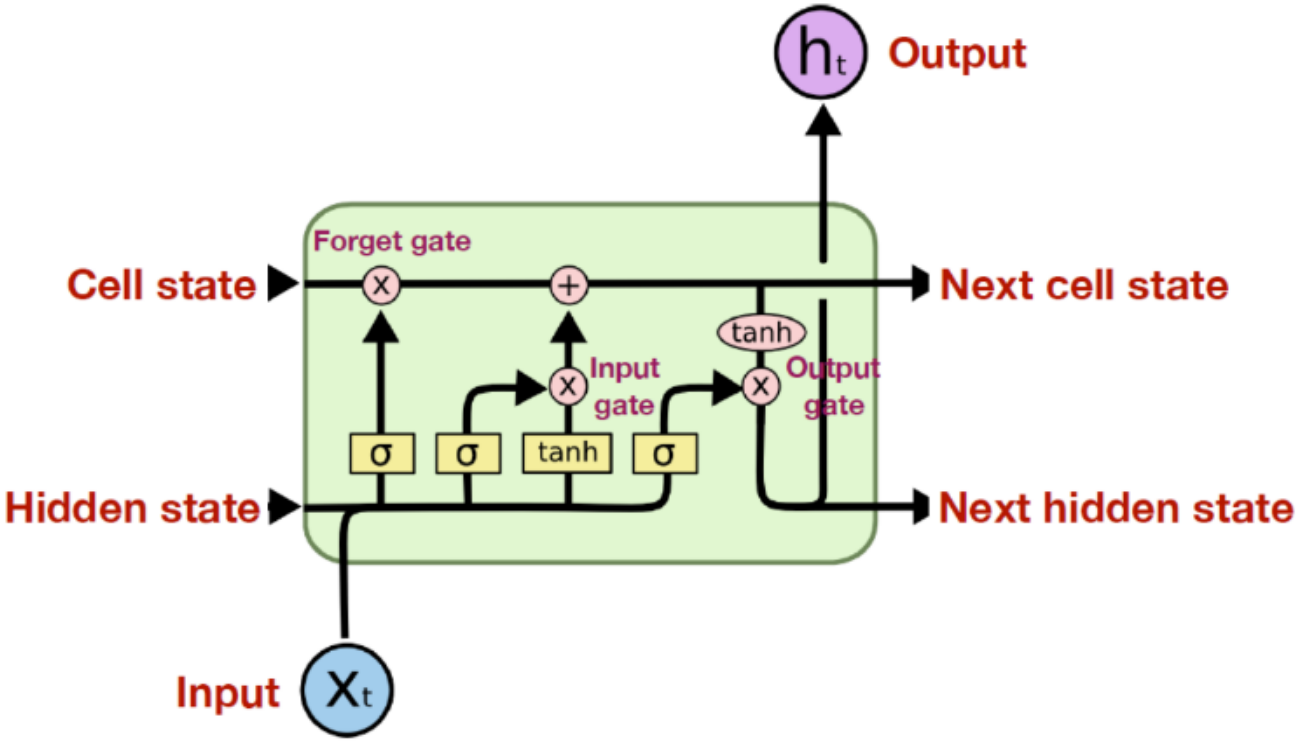
RNN funcionaba bien para datos cortos (memoria corta)

El efecto de cambios anteriores a la salida (a o h) debido a la activación hace que a la memoria a lo mucho sea 1 así que a largo plazo será cero.



LSTM mejora esto, siendo capaz de recordar también a largo plazo e igual con capacidad de añadir o eliminar información relevante

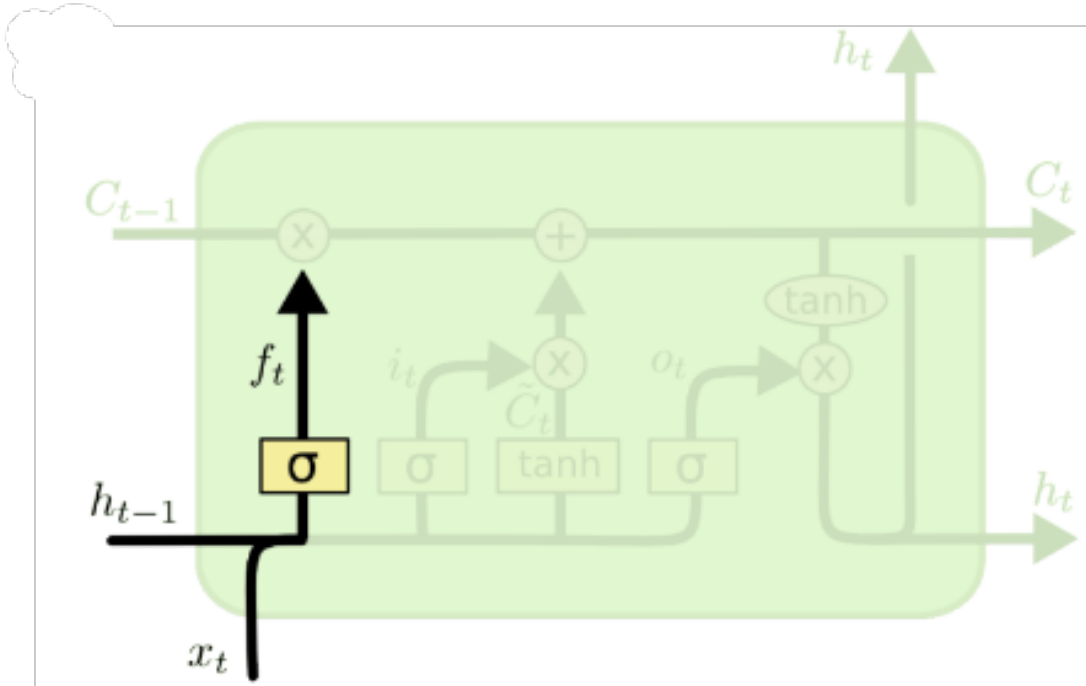
LSTM – Long-Short Term Memory



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Tiene adicional una Celda de estado: Es la clave de LSTM, como una banda transportadora donde se añaden o remueven datos de la memoria.

LSTM – Forget and Keep gate

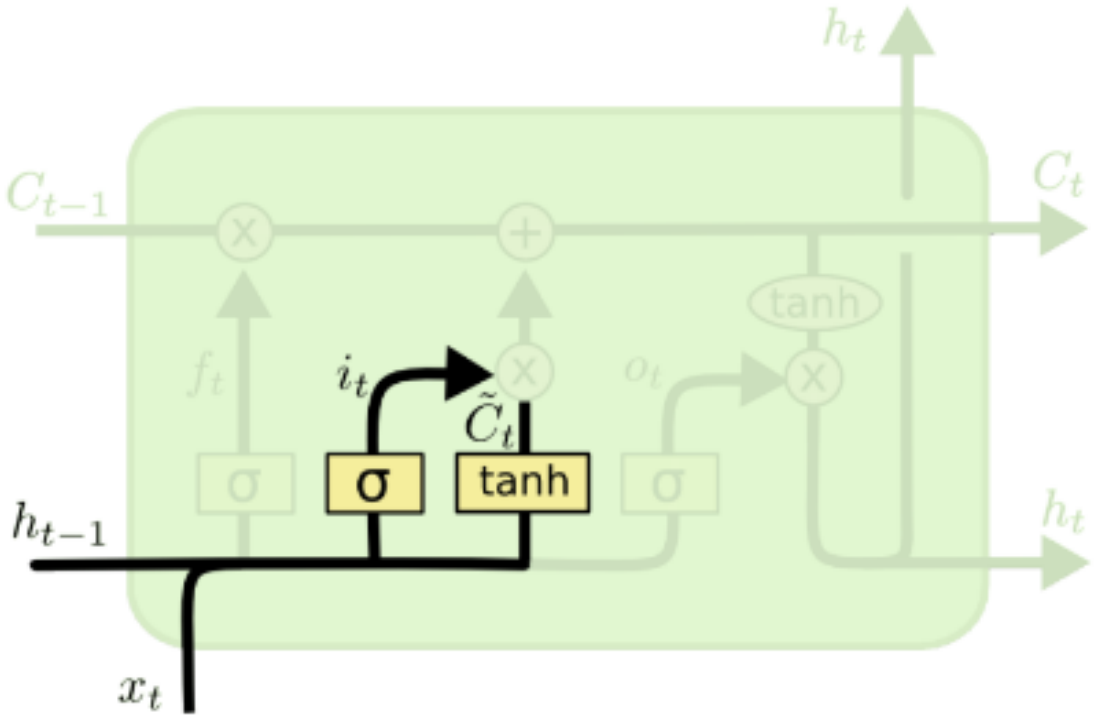


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Forget Gate
Decide qué información descartar de la celda de estado

Los sigmoid son los que actúan como válvulas y varían entre 0 y 1

LSTM – Input Gate

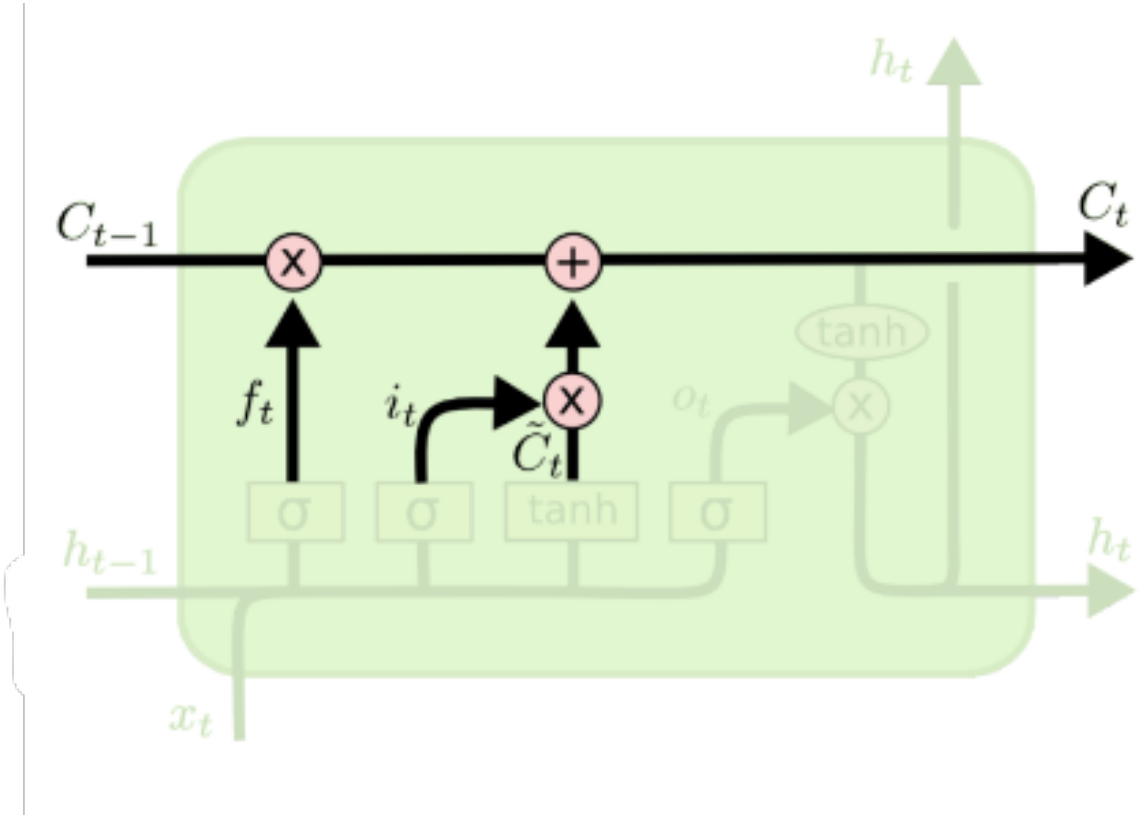


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

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

Update Gate:
Decide qué información se almacena en la celda de estado

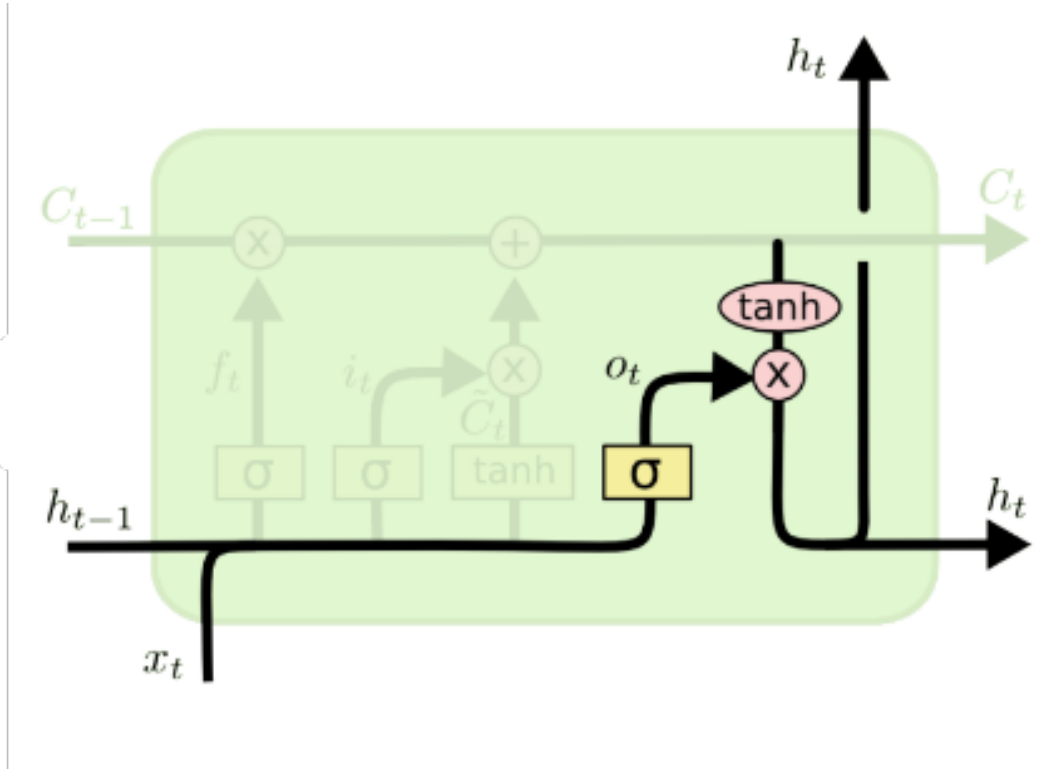
LSTM – Actualización de estado de celda



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

Cell state
Actualiza la celda de estado escalada por la entrada y olvida las gates

LSTM – Output Gate

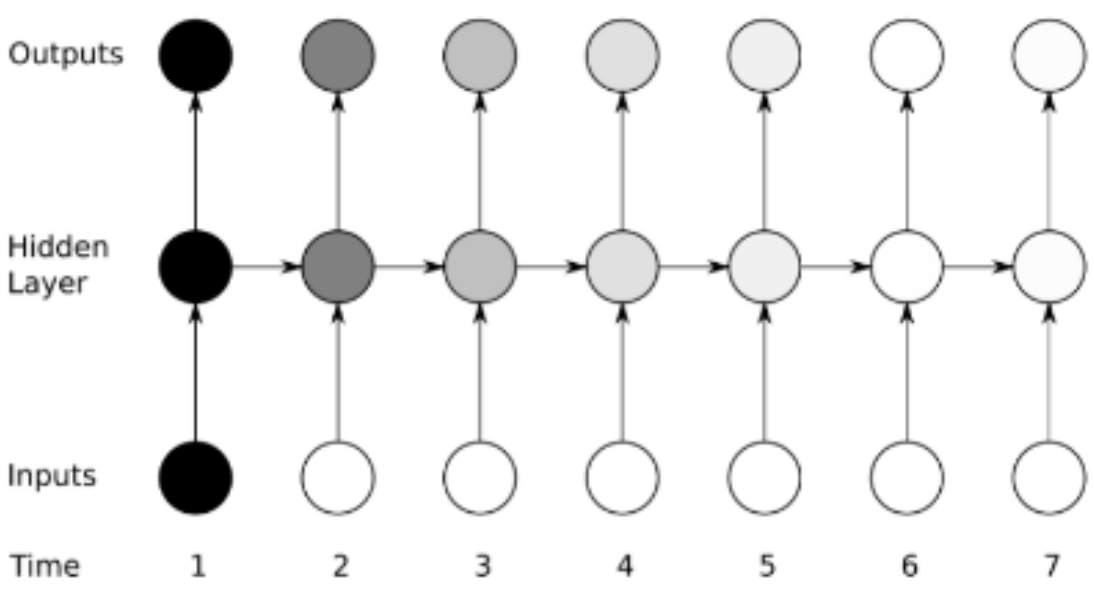


$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

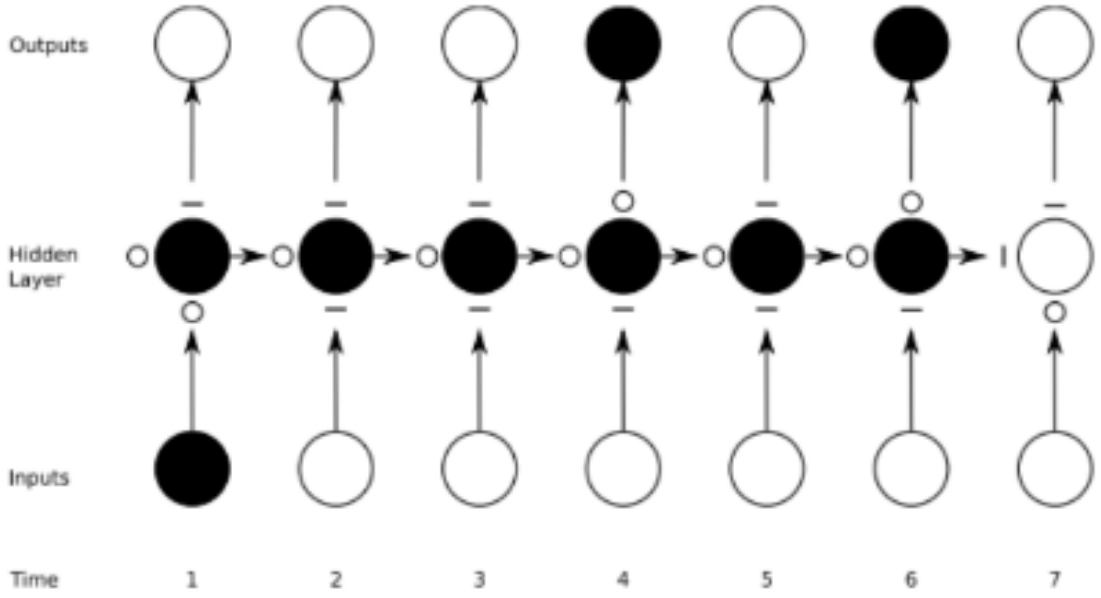
$$h_t = o_t * \tanh\left(C_t\right)$$

Output Gate:
Salida basada en la célula de estado actualizada

LSTM Vs. Vanilla RNN



Vanilla RNN



LSTM

Vanilla RNN va perdiendo información

LSTM mantiene información a no ser que se elimine total del cell state

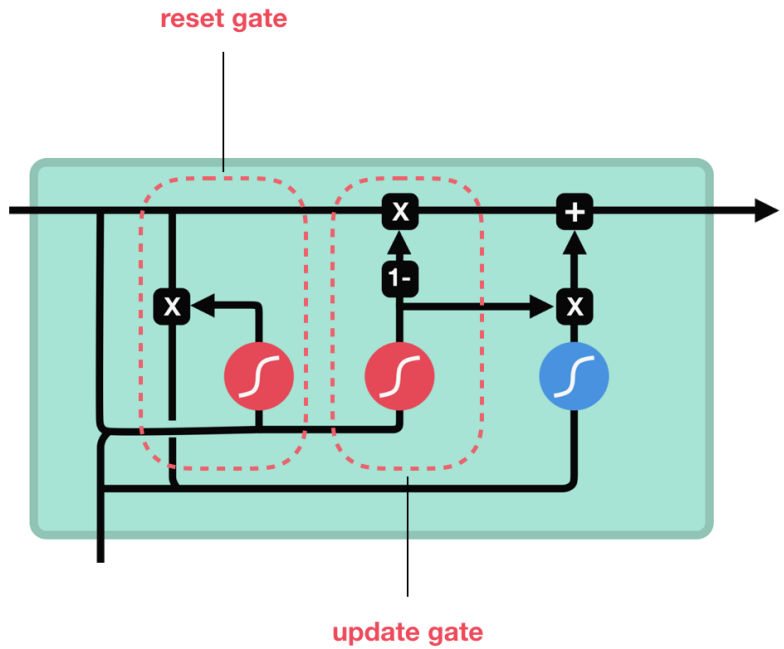
GRU – Gated Recurrent Unit

GRU es la mas nueva de las RNN y similar a LSTM, excepto que se deshicieron de la celda de estado y usaron el estado oculto para transferir información.

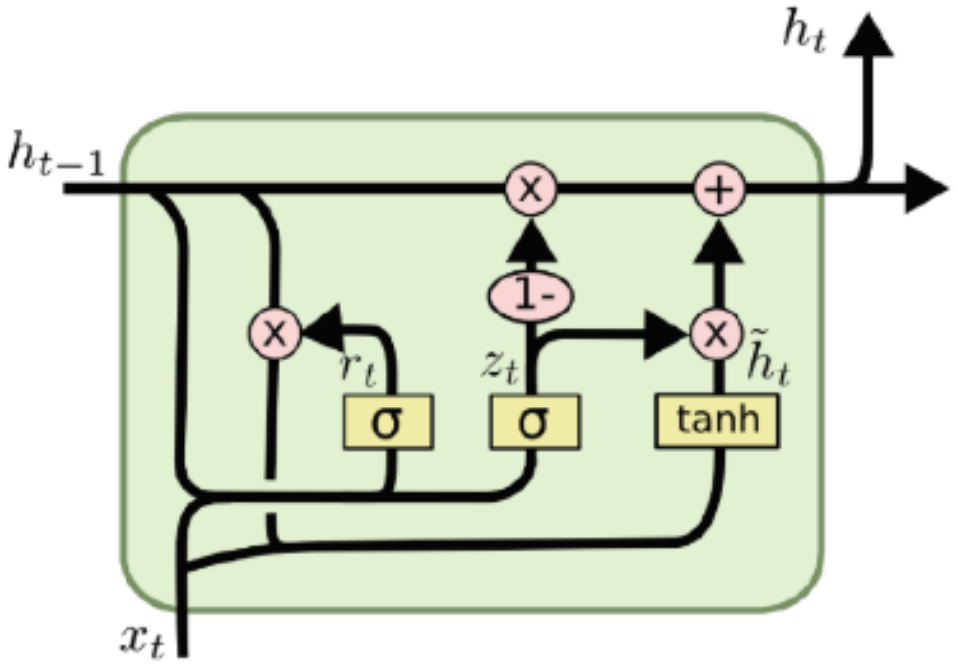
Dos puertas

1. Actualización: puerta de olvida, decide que mantener, desechar y agregar.
2. Reinicio: decide cuanta información del pasado olvidar

El estado oculto candidato se realiza igual a LSTM, la diferencia es que dentro del cálculo candidato, se restablecerá parte de la puerta anterior.



GRU – Gated Recurrent Unit



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

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

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

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

https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

Limitaciones RNN

1. **Desvanecimiento y explosión del gradiente**: Problemas con la propagación de gradientes a través de múltiples pasos de tiempo.(sobreajuste ó deje de aprender por gradientes 0)
2. **Memoria a corto plazo**: Dificultad para retener información relevante a largo plazo.
3. **Sensibilidad al orden de los datos**: Dependencia en el orden de los datos de entrada, lo que puede afectar su capacidad para modelar secuencias complejas.
4. **Problema de información olvidada**: La información relevante puede perderse a medida que la secuencia avanza.
5. **Limitaciones de paralelización**: Dificultad para paralelizar el entrenamiento e inferencia debido a su naturaleza secuencial.
6. **Incapacidad para manejar dependencias a larga distancia**: Dificultad para capturar relaciones complejas en datos de series temporales o texto a lo largo de intervalos largos.

Aplicaciones del RNN

1. **Procesamiento del Lenguaje Natural (NLP)**: Traducción automática, análisis de sentimientos, resumen de textos, reconocimiento de voz.
2. **Modelado de Series Temporales**: Pronóstico del tiempo, análisis financiero, predicción de demanda, monitorización de la salud.
3. **Reconocimiento de Voz**: Reconocimiento y transcripción de voz, control de dispositivos, transcripción de llamadas.
4. **Generación de Texto y Música**: Creación de historias, composición musical, generación de diálogos.
5. **Procesamiento de Imágenes y Video**: Etiquetado de imágenes, segmentación semántica, generación de descripciones, reconocimiento de actividad.
6. **Modelado de Secuencias Biológicas**: Predicción de estructuras de proteínas, análisis de secuencias de ADN y ARN.
7. **Robótica y Control**: Control de robots autónomos, predicción de trayectorias.
8. **Aplicaciones en Juegos**: Generación de niveles, aprendizaje de políticas para agentes.