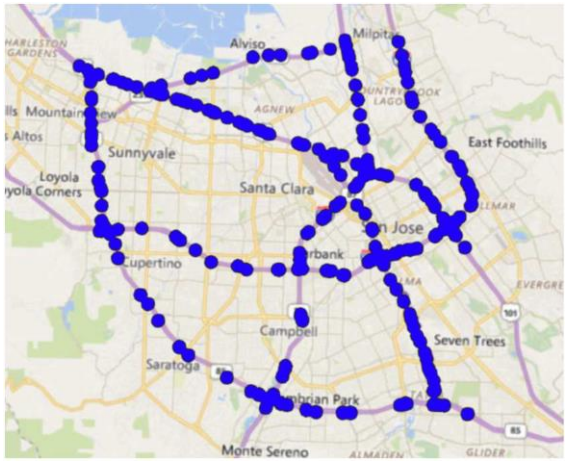
Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ingeniería de Transporte y Logística

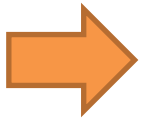
Sistemas Urbanos Inteligentes

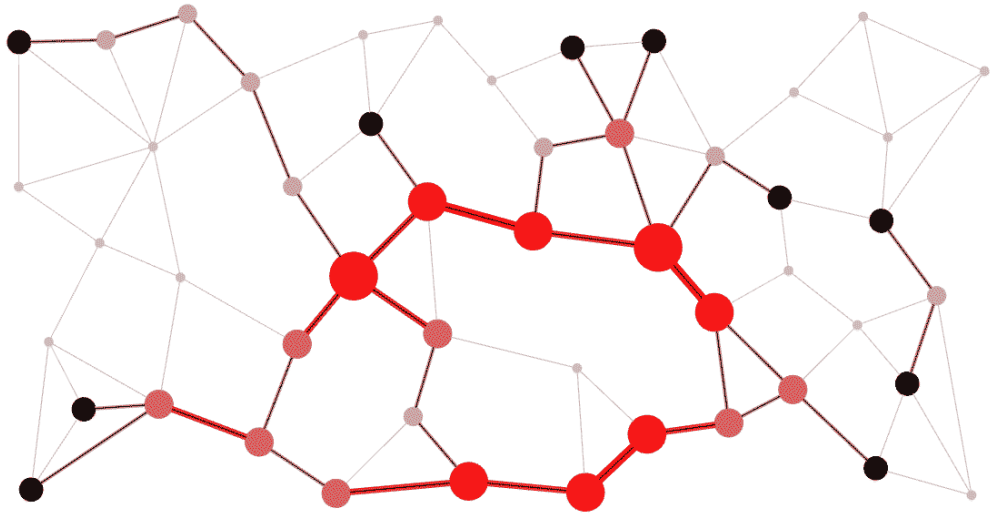
Redes Neuronales Recurrentes (RNN)

Hans Löbel
Dpto. Ingeniería de Transporte y Logística
Dpto. Ciencia de la Computación

Pensemos en un problema de predicción de velocidad en una red

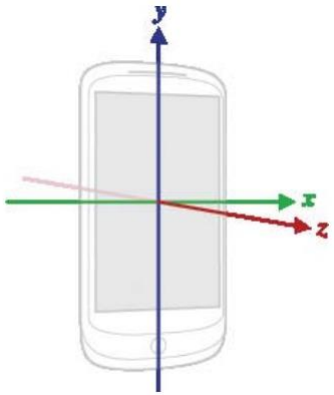


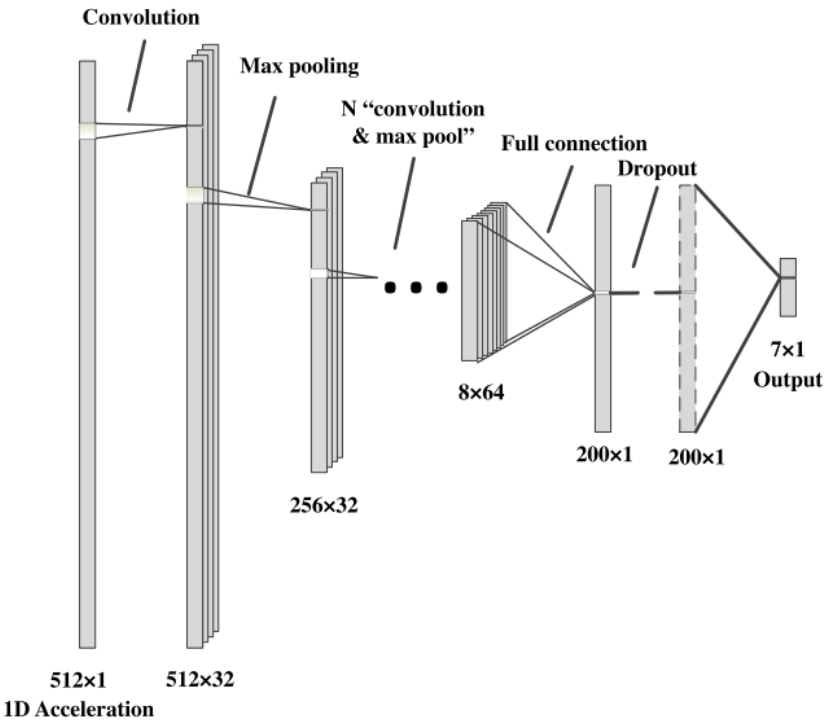




¿Qué limitación tiene este enfoque?

Pasemos ahora a un problema de estimación de modo

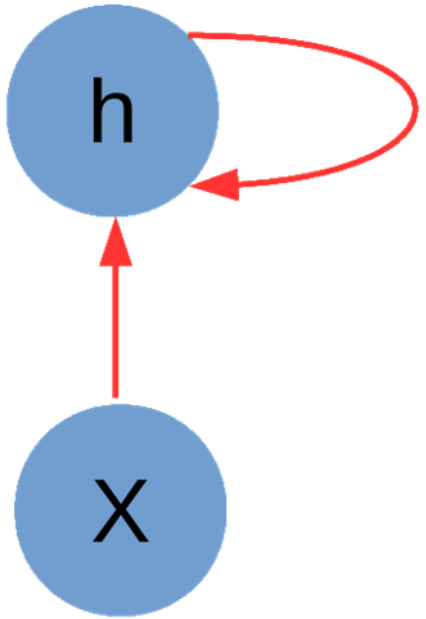




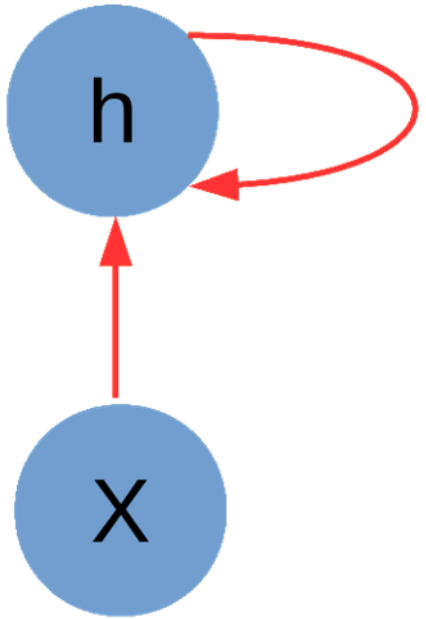
¿Qué limitación tiene este enfoque?

Afortunadamente, existen redes llamadas Redes Neuronales Recurrentes (RNN), que están hechas para procesar secuencias

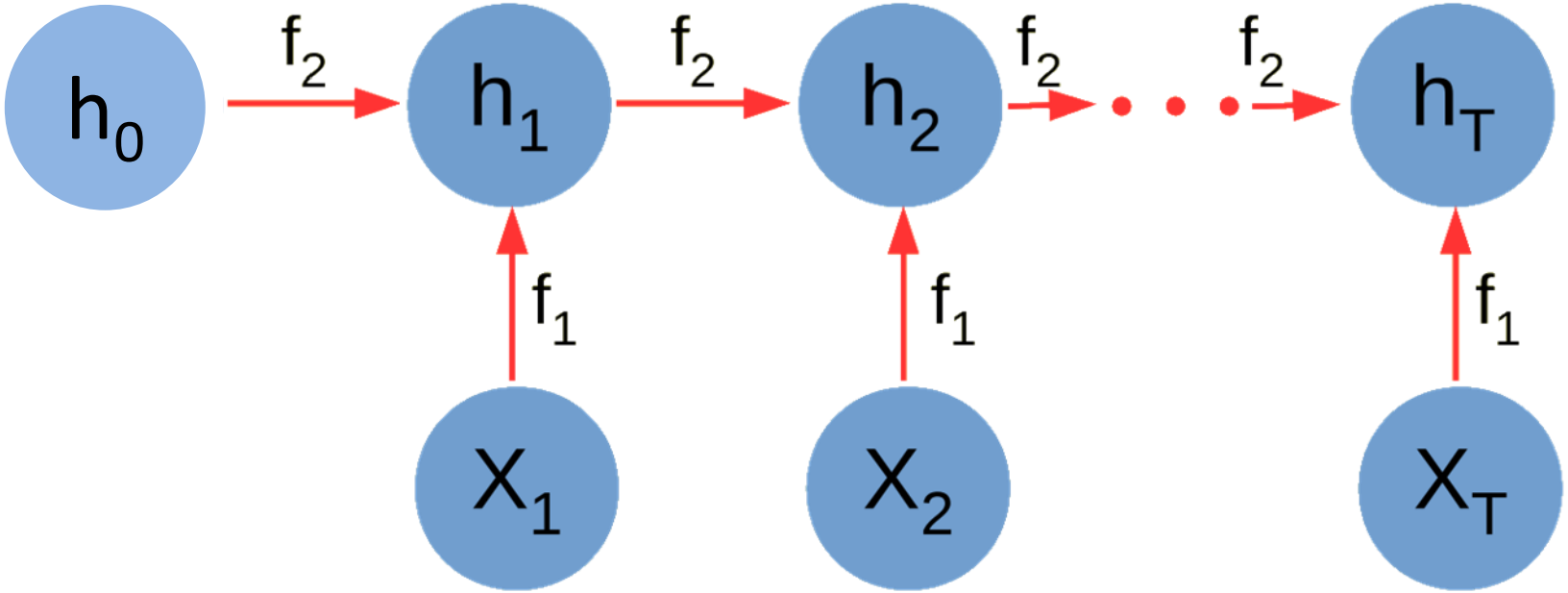
- Propuestas en la década de los 80.
- Trabajan de manera secuencial, procesando uno a uno los elementos de la secuencia/serie.
- A diferencia de métodos tradicionales y de otros tipos de redes, las RNN mantienen un estado, es decir, “tienen memoria”.
- Cómo se maneja esta memoria es lo que caracteriza a los distintos tipos de redes recurrentes existentes (RNN, LSTM, GRU).



Expandamos este diagrama, para ver explícitamente el funcionamiento de una RNN en el tiempo

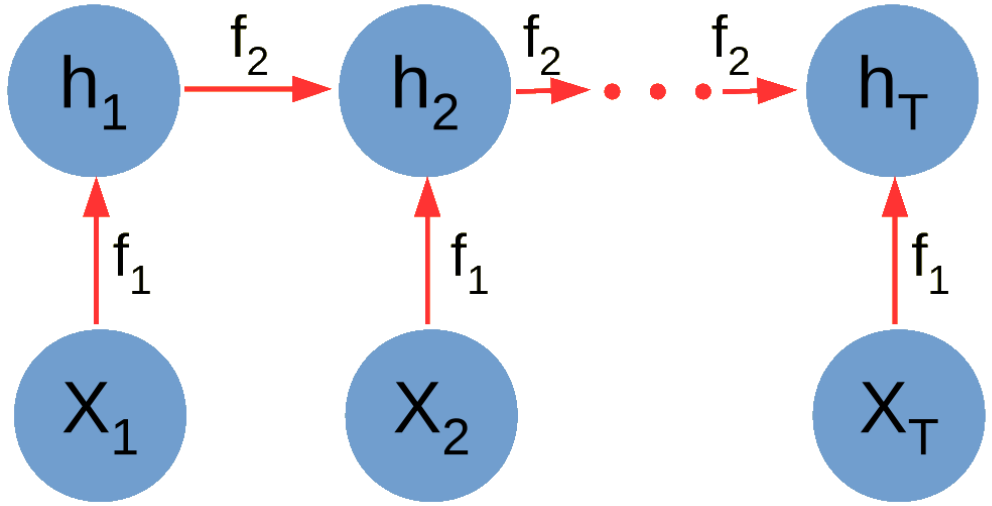


Expandamos este diagrama, para ver explícitamente el funcionamiento de una RNN en el tiempo



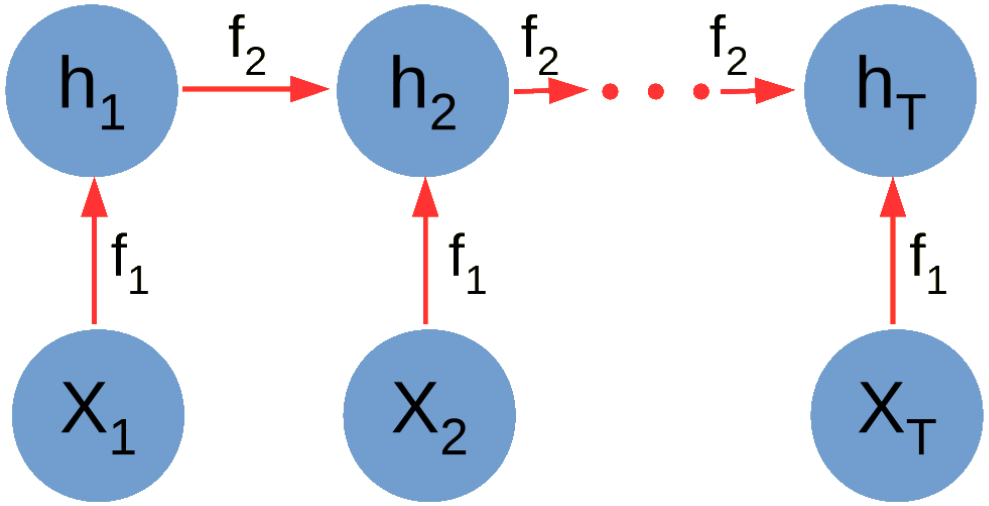
$$h_t = \sigma(f_1(x_t), f_2(h_{t-1}))$$

Formalicemos un poco esto



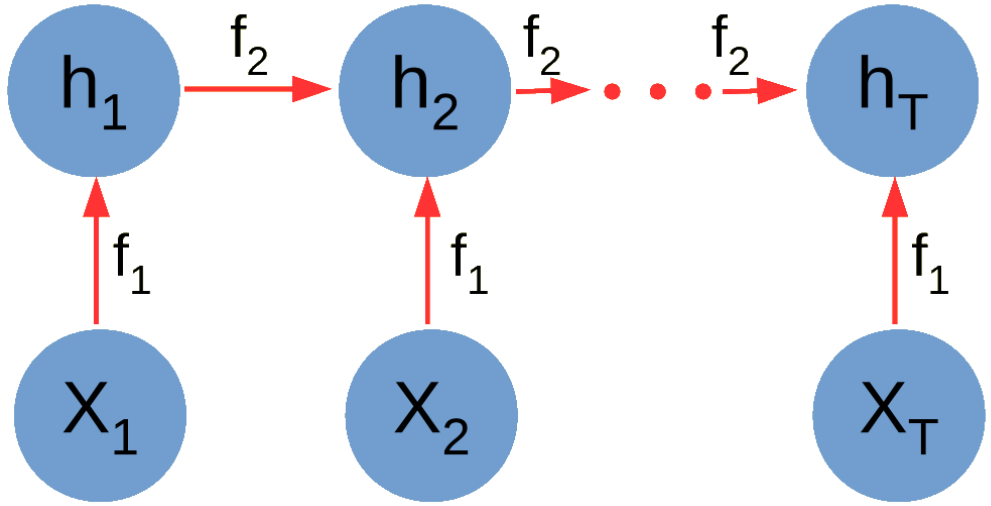
- x_t: vector de entrada que codifica la secuencia en el paso t
- h_t: vector de estado oculto (latente/interno) en el paso t, que codifica la historia de la entrada hasta ese momento
- f_1 y f_2: funciones paramétricas entrenables (aquí está la clave)
- t: paso de la secuencia, usualmente tiene significado temporal, pero puede representar cualquier relación de orden (espacio, ranking, etc).

La configuración típica de una RNN es con funciones lineales y sigmoides



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

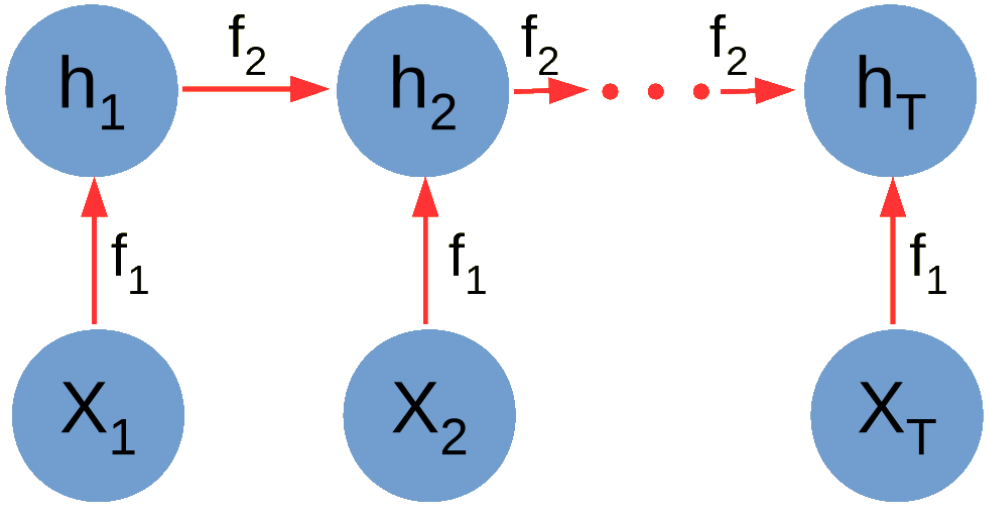
La configuración típica de una RNN es con funciones lineales y sigmoides



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

- Al ver esto en detalle, podemos inferir que se genera un grafo de cómputo profundo, sin un incremento en la cantidad de parámetros (¿por qué?).
- Esto implica que al utilizar una capa recurrente, podemos capturar dependencias de longitud arbitraria en los datos, controlando efectivamente la capacidad del modelo (no necesitamos filtros más grandes).

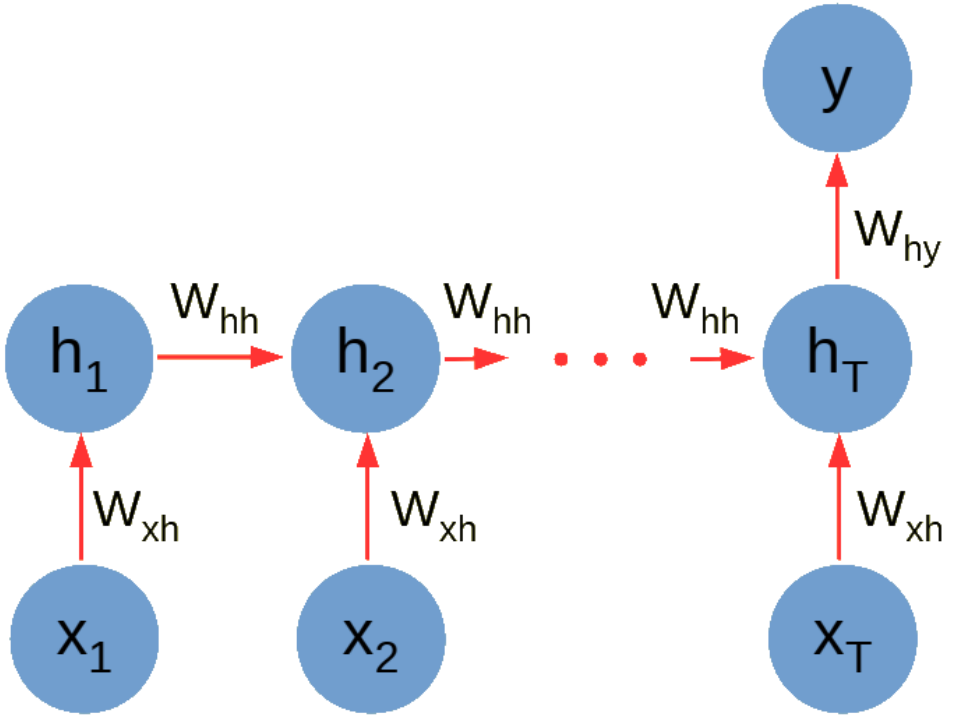
Las redes recurrentes como elementos de memoria



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

- Las RNN aprenden a usar el estado oculto h_t como una *feature* que representa las entradas hasta el paso t, es decir, captura solo la información relevante para la tarea.
- Esto permite capturar la estructura secuencial/temporal de la entrada (análogamente a como las convoluciones capturan la estructura espacial en una CNN).

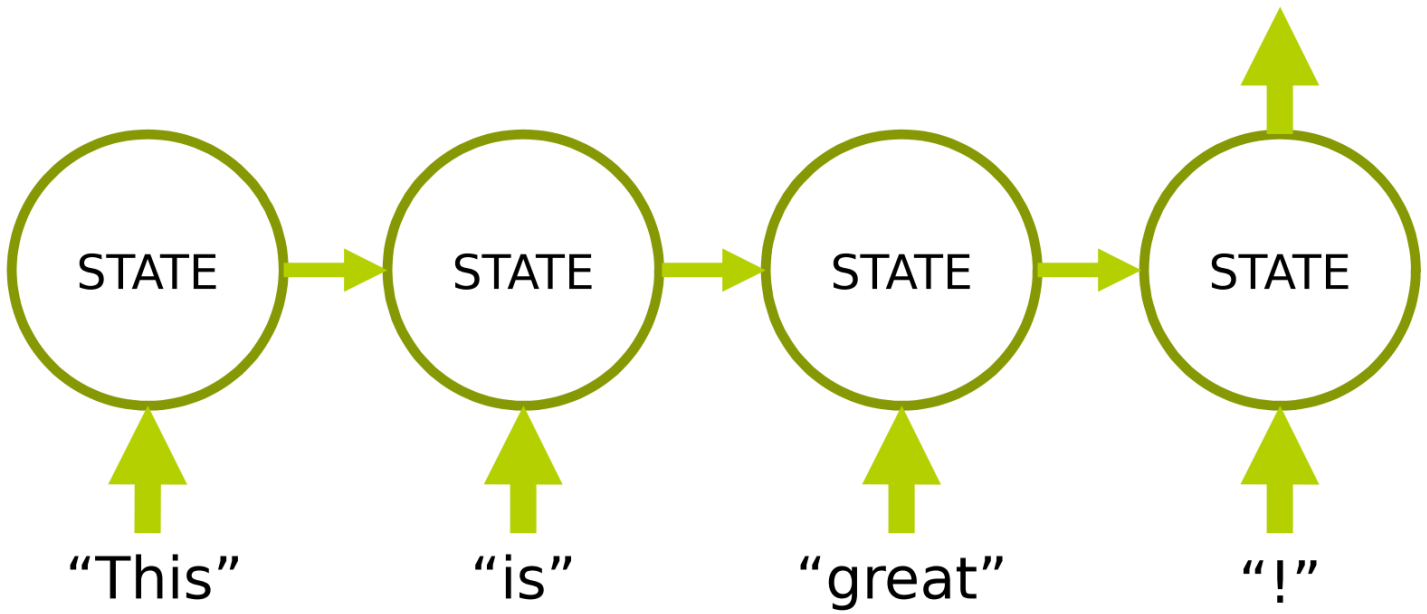
Aún nos falta generar la salida que nos permita utilizar supervisión para el aprendizaje



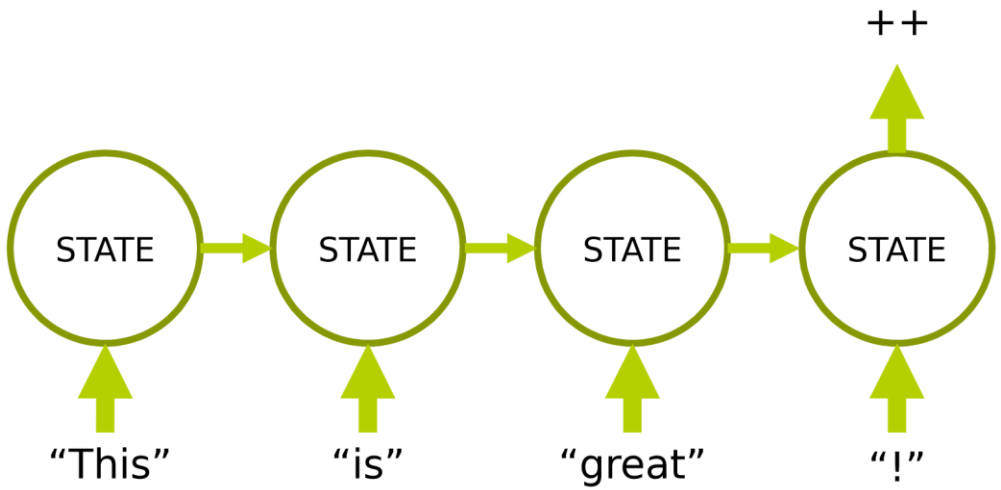
$$h_t = \sigma(W_{hh}\, h_{t-1} + W_{xh}\, x_t)$$

$$y = \sigma(W_{hy}\, h_T)$$

Veamos un caso aplicado: predicción de sentimiento

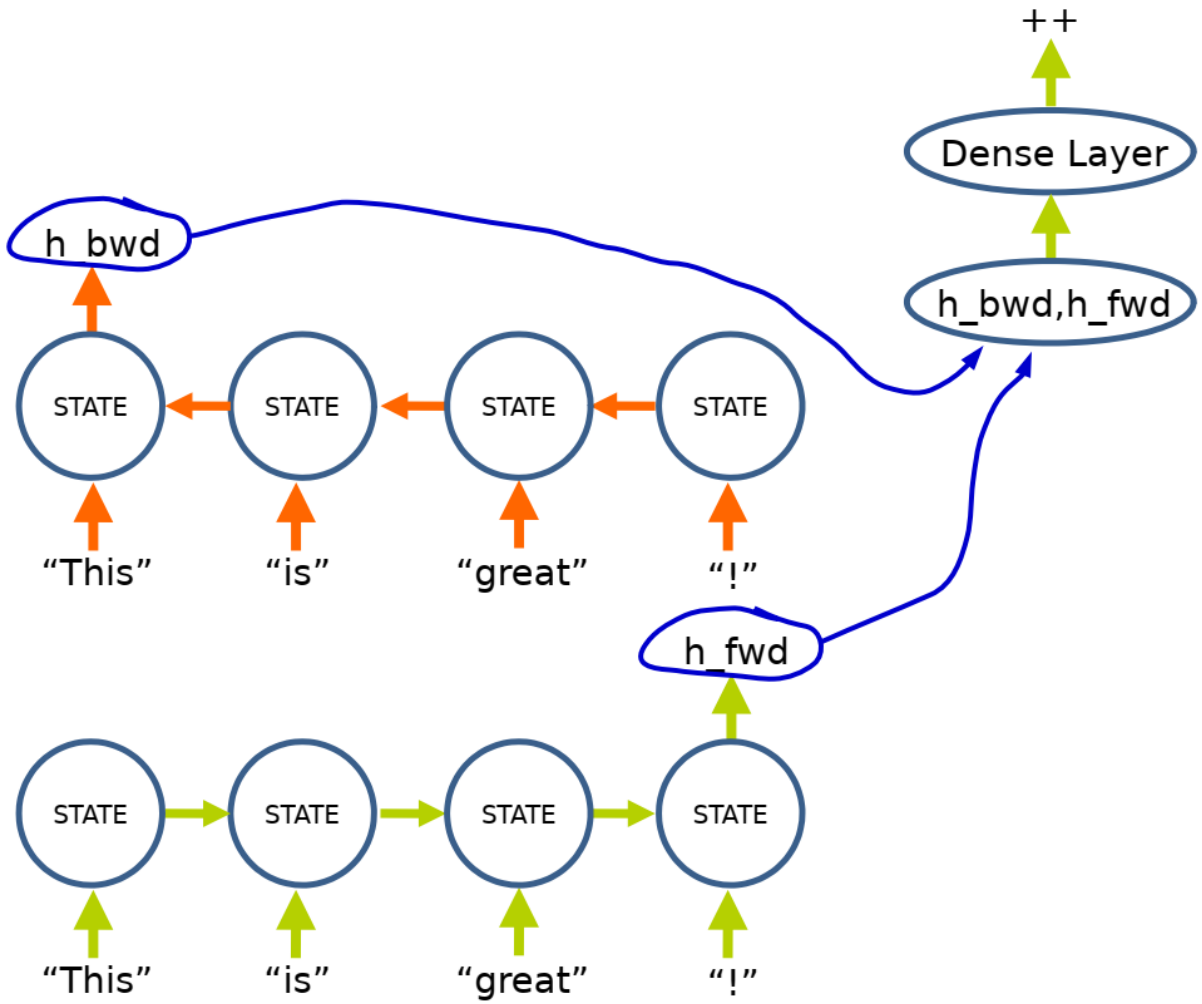


Veamos un caso aplicado: predicción de sentimiento

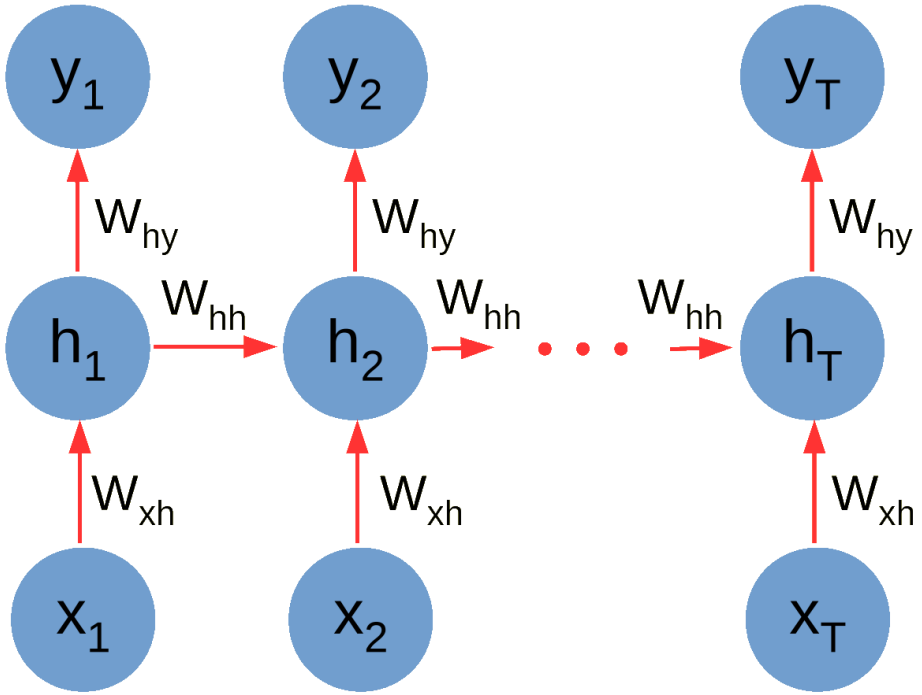


- En cada paso, el estado oculto modela la historia/características de la secuencia hasta el momento.
- En este problema particular, podemos también capturar la información que viene a continuación, mediante una RNN bidireccional.

Veamos un caso aplicado: predicción de sentimiento con una RNN bidireccional

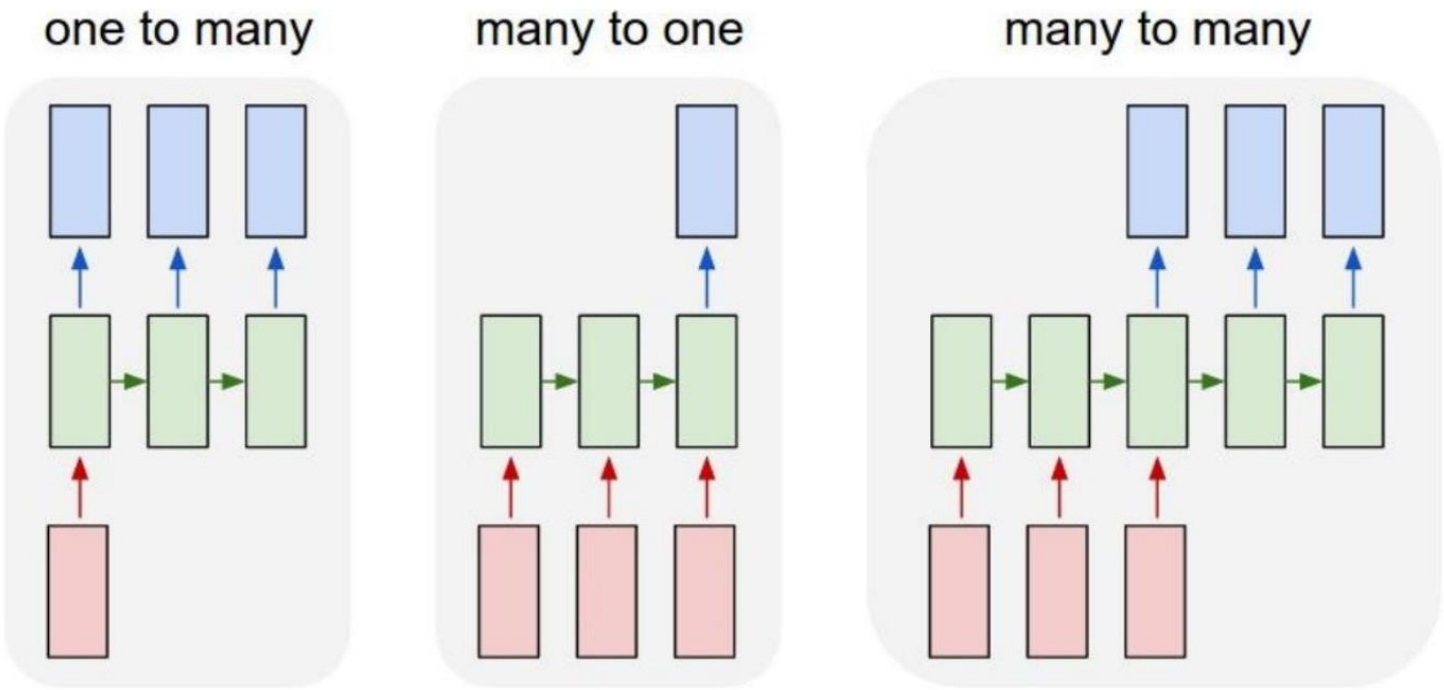


La salida de una RNN puede adecuarse a la tarea de turno

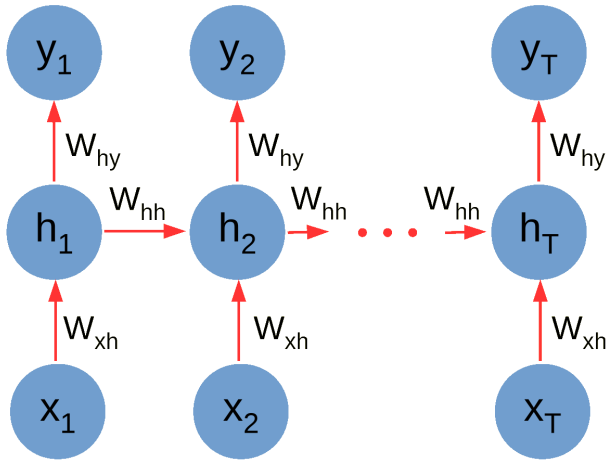


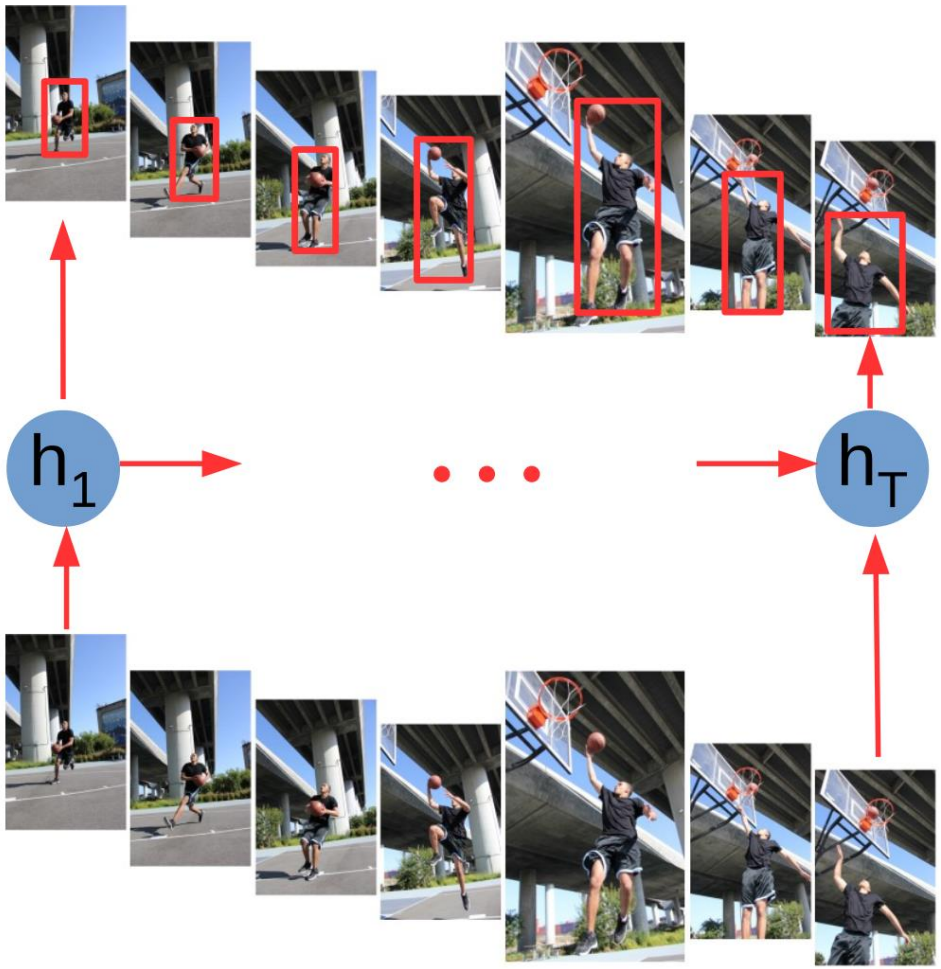
En esta configuración, la RNN genera una salida por cada valor de la secuencia de entrada.

La salida de una RNN puede adecuarse a la tarea de turno



Veamos más ejemplos: seguimiento en videos

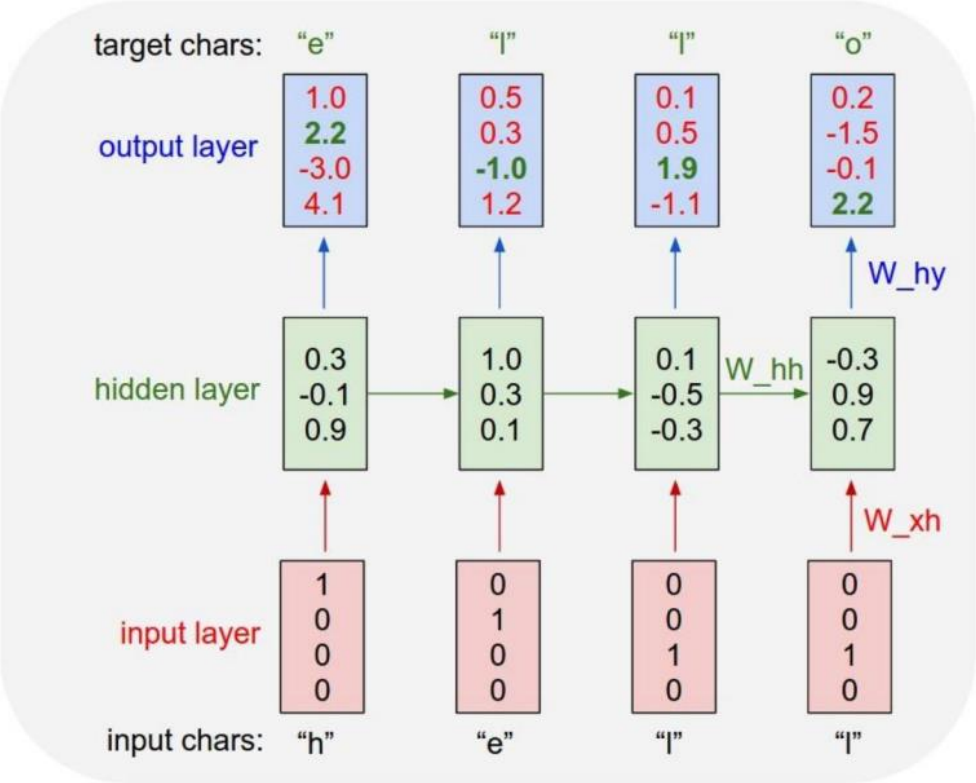




Veamos más ejemplos: generación de texto a nivel de caracteres

Vocabulary:
[h,e,l,o]

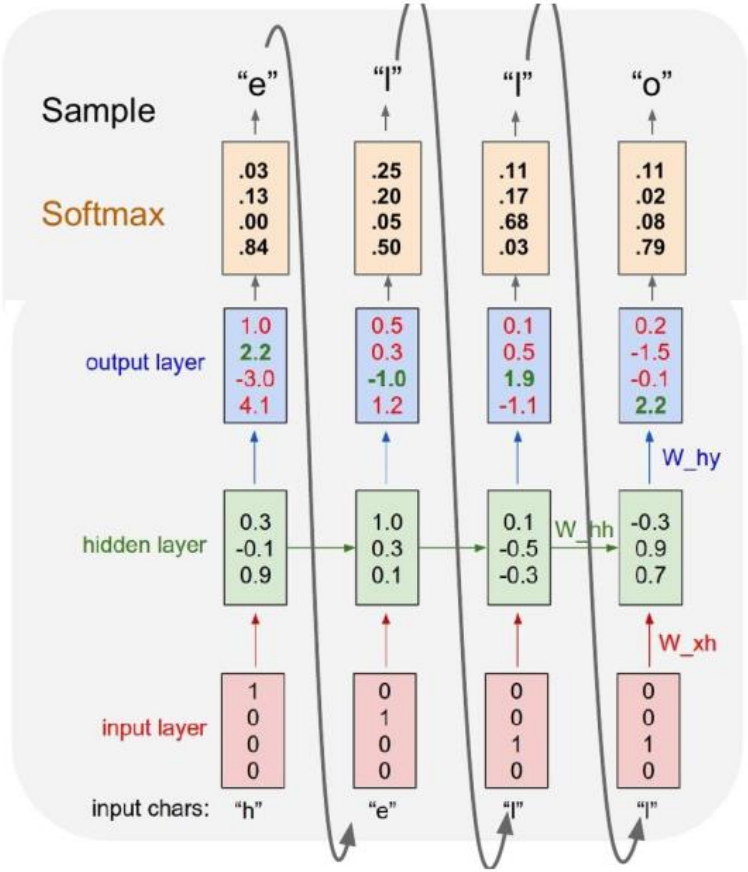
Example training sequence:
“hello”



Veamos más ejemplos: generación de texto a nivel de caracteres

Vocabulary:
[h,e,l,o]

At test-time sample characters one at a time, feed back to model



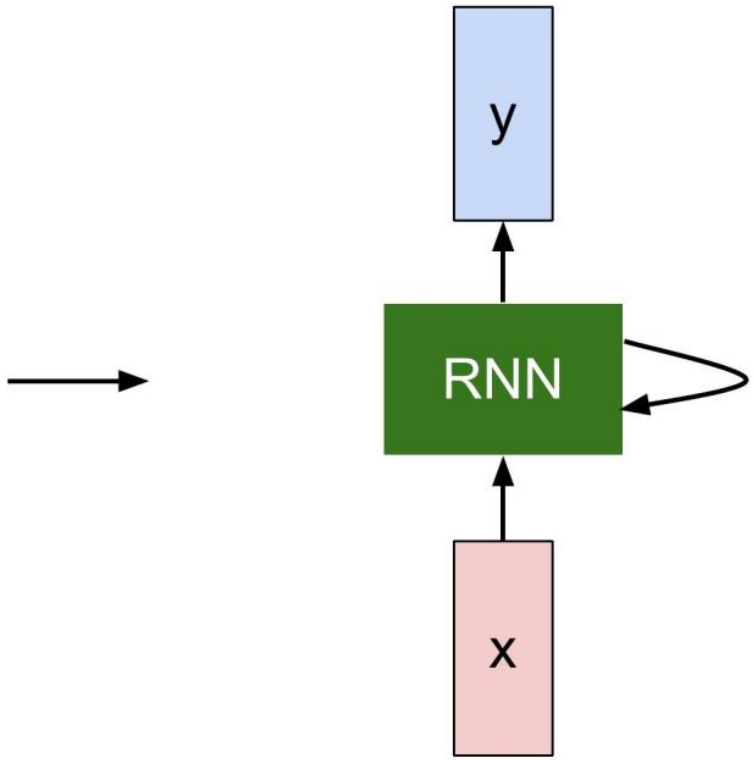
Veamos más ejemplos: generación de texto a nivel de caracteres

THE SONNETS

by William Shakespeare

From fairest creatures we desire increase,
That thereby beauty's rose might never die,
But as the riper should by time decease,
His tender heir might bear his memory:
But thou, contracted to thine own bright eyes,
Feed'st thy light's flame with self-substantial fuel,
Making a famine where abundance lies,
Thyself thy foe, to thy sweet self too cruel:
Thou that art now the world's fresh ornament,
And only herald to the gaudy spring,
Within thine own bud buriest thy content,
And tender churl mak'st waste in niggarding:
    Pity the world, or else this glutton be,
    To eat the world's due, by the grave and thee.

When forty winters shall besiege thy brow,
And dig deep trenches in thy beauty's field,
Thy youth's proud livery so gazed on now,
Will be a tatter'd weed of small worth held:
Then being asked, where all thy beauty lies,
Where all the treasure of thy lusty days;
To say, within thine own deep sunken eyes,
Were an all-eating shame, and thriftless praise.
How much more praise deserv'd thy beauty's use,
If thou couldst answer 'This fair child of mine
Shall sum my count, and make my old excuse,'
Proving his beauty by succession thine!
    This were to be new made when thou art old,
    And see thy blood warm when thou feel'st it cold.



Veamos más ejemplos: generación de texto a nivel de caracteres

```
tyntd-iafhatawiaoihrdemot  lytdws  e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e
plia tklrgd t o idoe ns,smtt   h ne etie h,hregtrs nigtike,aoaenns lng
```

train more

```
"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."
```

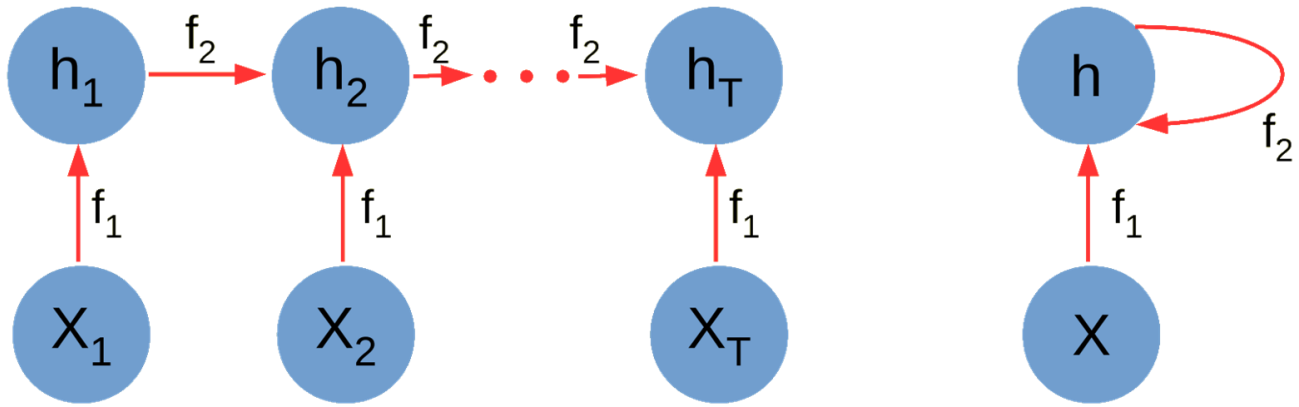
train more

```
Aftair fall unsuch that the hall for Prince Velzonski's that me of
her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort
how, and Gogition is so overelical and ofter.
```

train more

```
"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftened him.
Pierre aking his soul came to the packs and drove up his father-in-law women.
```

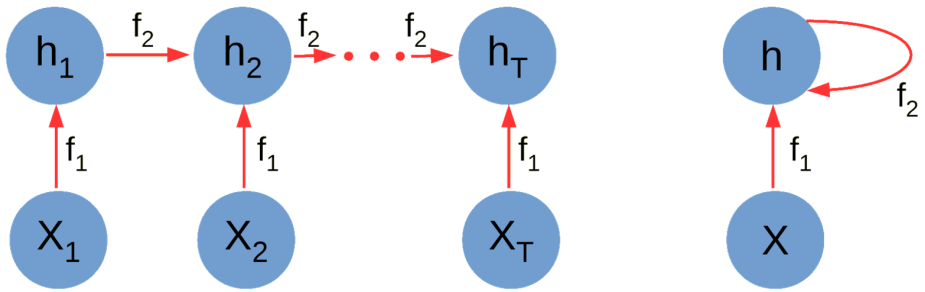
Si bien las RNN pueden tener diversas configuraciones, la clave es compartir pesos de manera profunda al incluir ciclos



$$h_t = f(h_{t-1}, x_t)$$

Los ciclos, o conexiones recurrentes, son la herramienta que provee cierto grado de memoria. La profundidad de esta recurrencia define cuán “grande” es la memoria.

Si bien las RNN pueden tener diversas configuraciones, la clave es compartir pesos de manera profunda al incluir ciclos

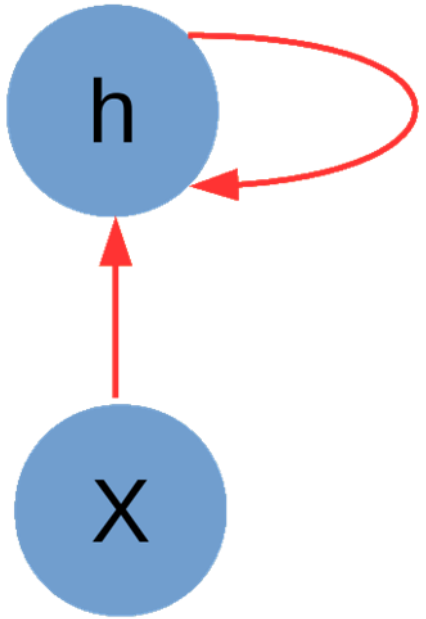


$$h_t = f(h_{t-1}, x_t)$$

- El compartir parámetros provee un mecanismo para modelar secuencias de largos arbitrarios.
- Al verlo en detalle, podemos notar que al desenrollar el procesamiento de una secuencia con una RNN, esta se transforma en algo parecido a un MLP donde todas las capas comparten los parámetros.
- A diferencia de la estructura fija para compartir parámetros de las CNN (filtro convolucional), las RNN permiten que este acto sea profundo (estado oculto actualizado de acuerdo al largo de la secuencia).

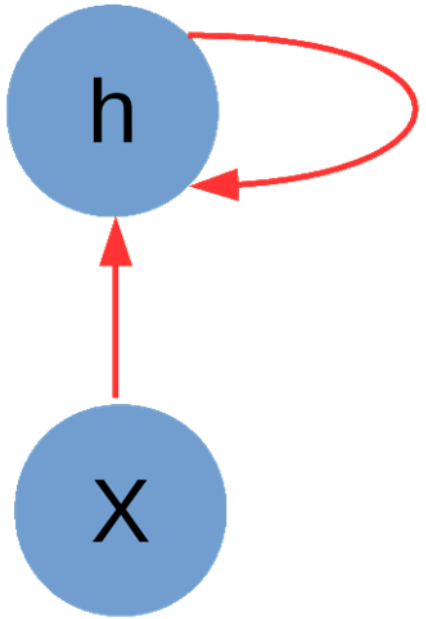
Volvamos a la idea con que partimos el capítulo, sobre usar redes neuronales para modelar secuencias

- CNN y MLP solo pueden ser aplicadas a problemas donde entrada y salida tienen un largo fijo.
- Para muchos dominios estos es una limitante fundamental, ya que la manera natural de representar sus datos es a través de elementos de largo variable y arbitrario
- El modo de compartir parámetros de una RNN permite transformar la información de una secuencia de tamaño arbitrario en un vector de tamaño fijo.
- Además, la salida puede generarse de manera secuencial, lo que permite salidas de largo arbitrario.

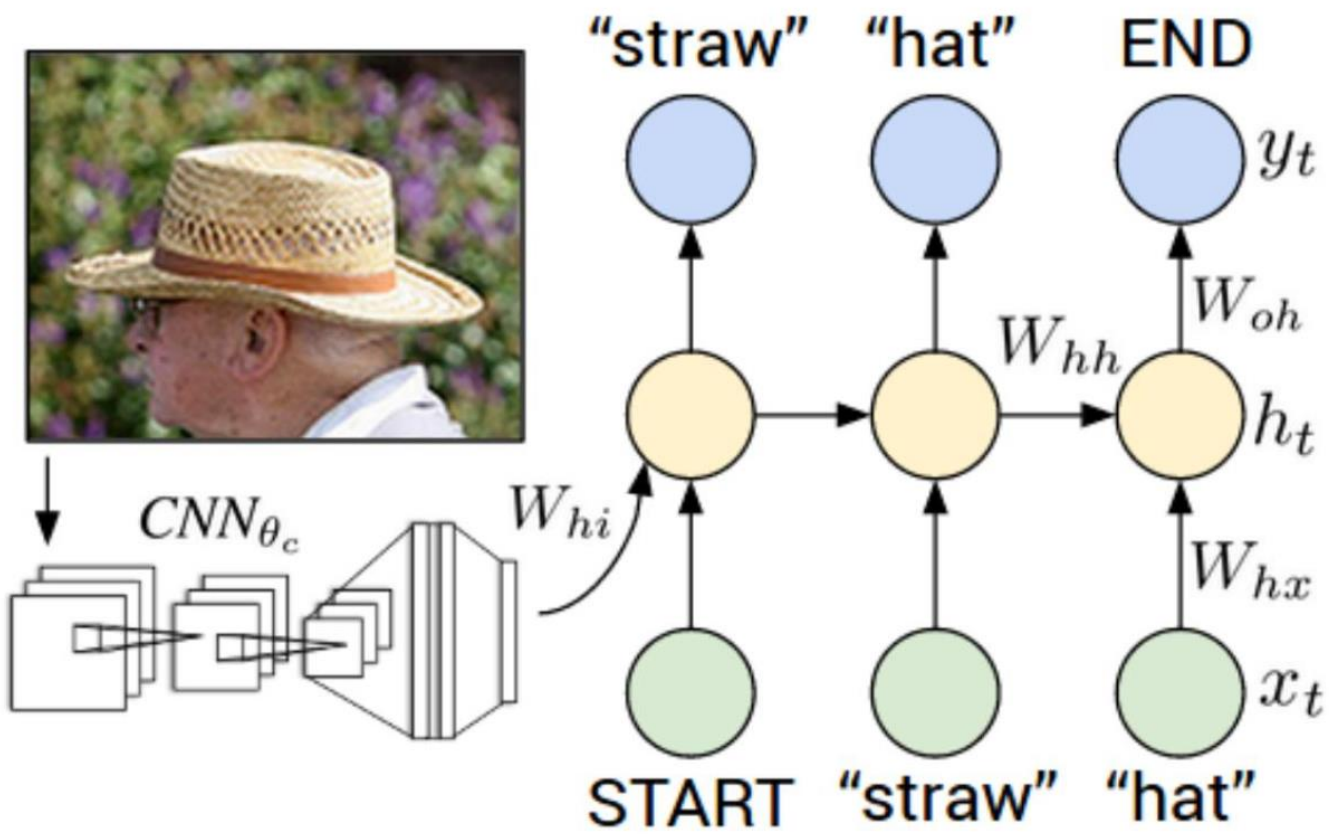


Volvamos a la idea con que partimos el capítulo, sobre usar redes neuronales para modelar secuencias

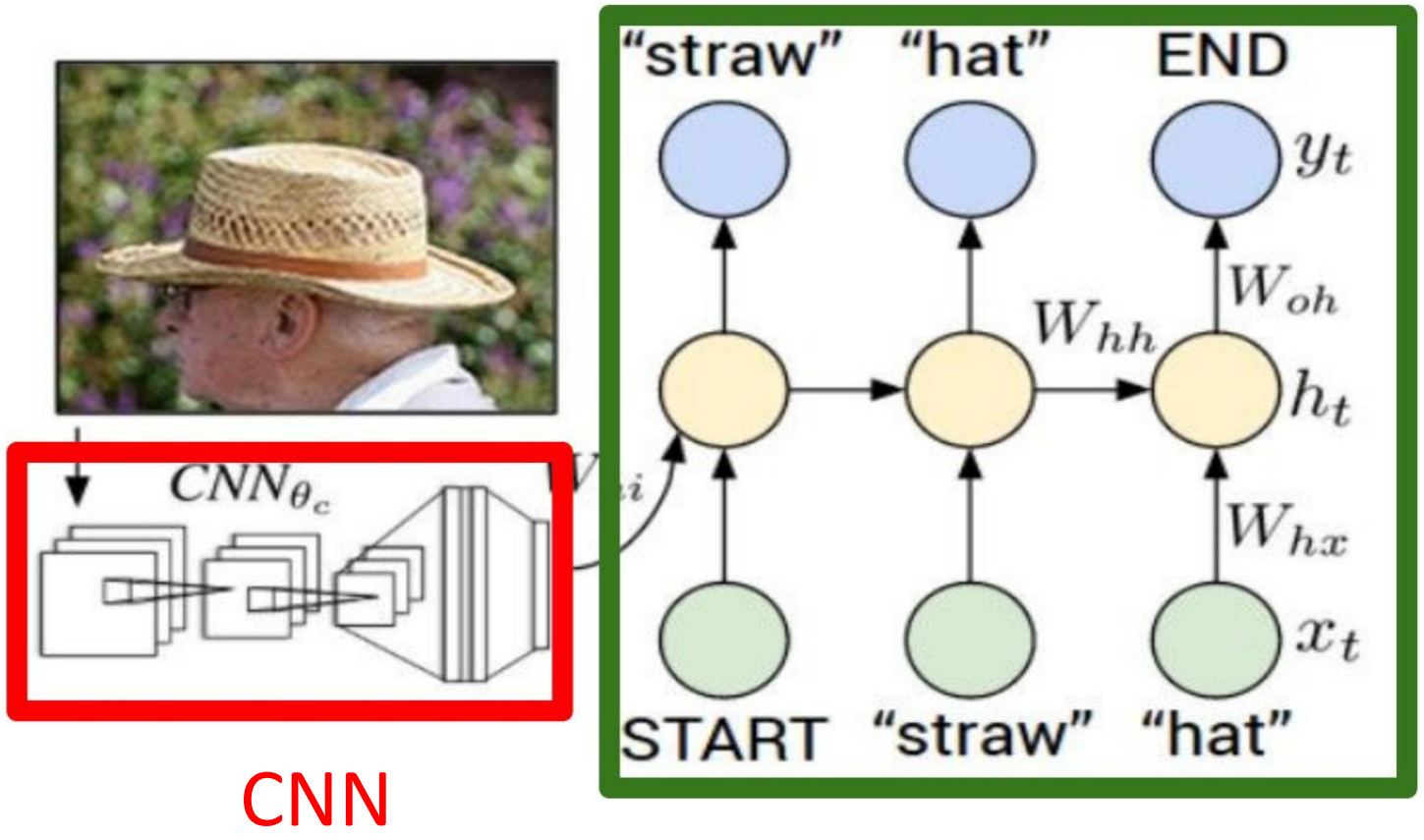
- Otra limitante fundamental de las CNN es que solo pueden modelar dependencias cortas en los datos.
- Si bien son jerárquicas y composicionales, la dependencias que pueden modelar las CNN están limitadas por el tamaño de los filtros y la cantidad de capas, que son fijos.
- La habilidad de distinguir y caracterizar el largo de las dependencias (contexto) es clave para construir sistemas inteligentes.
- (Más adelante vamos a ver que esto también es un problema para las RNN puras).



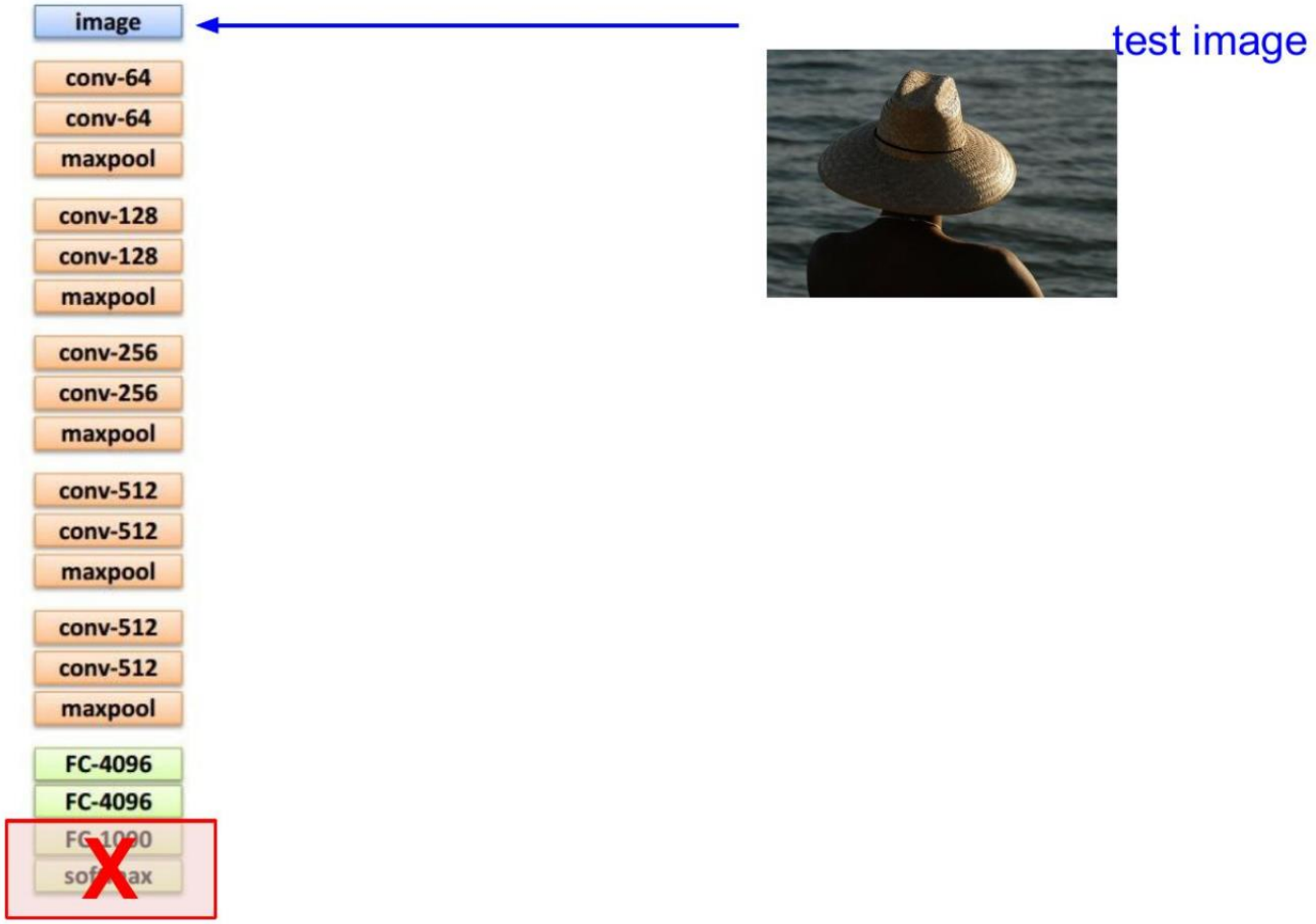
Podemos combinar CNN y RNN, aprovechando las fortalezas de cada una. Un ejemplo es *image captioning*



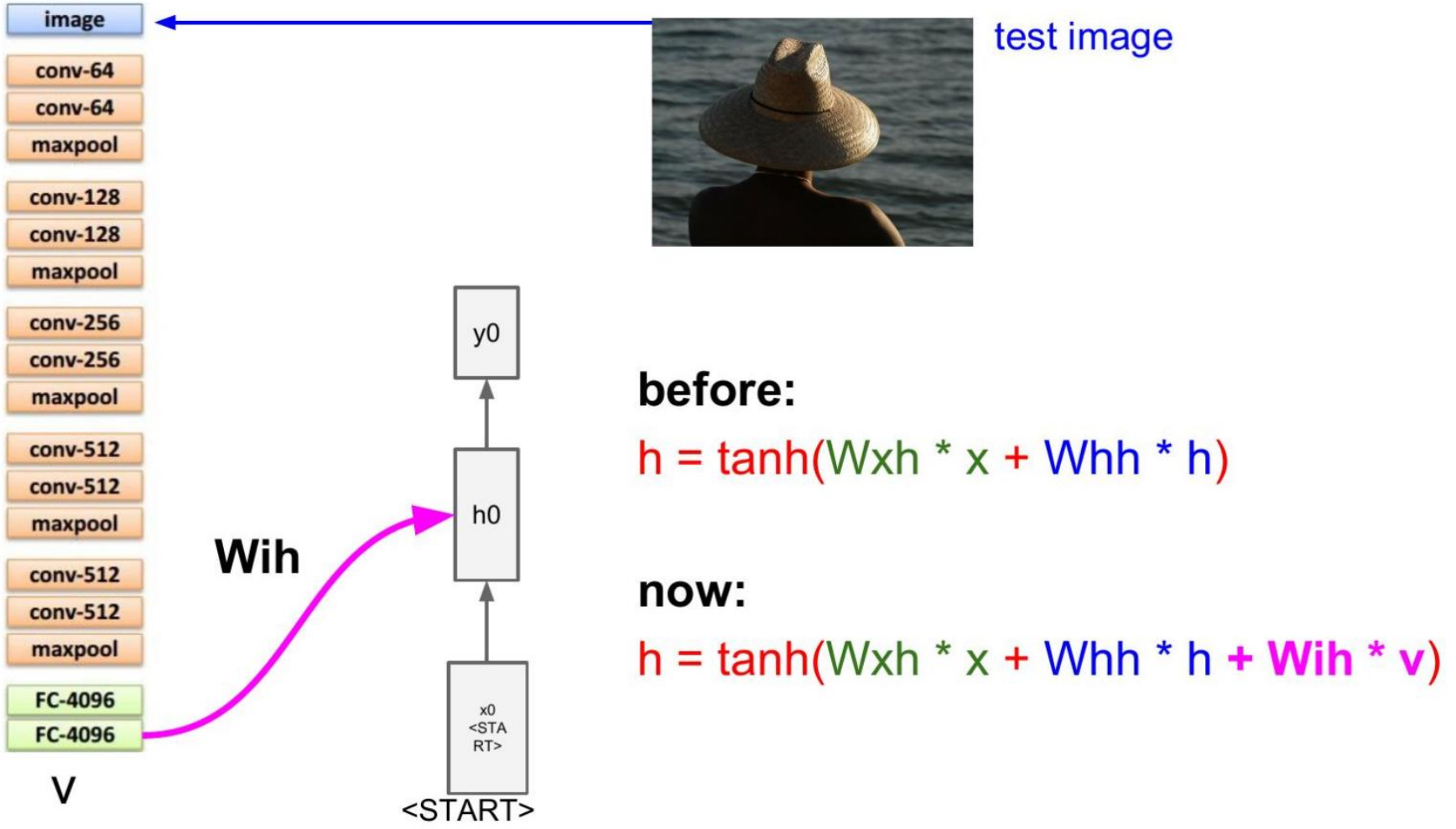
Podemos combinar CNN y RNN, aprovechando las fortalezas de cada una. Un ejemplo es *image captioning*



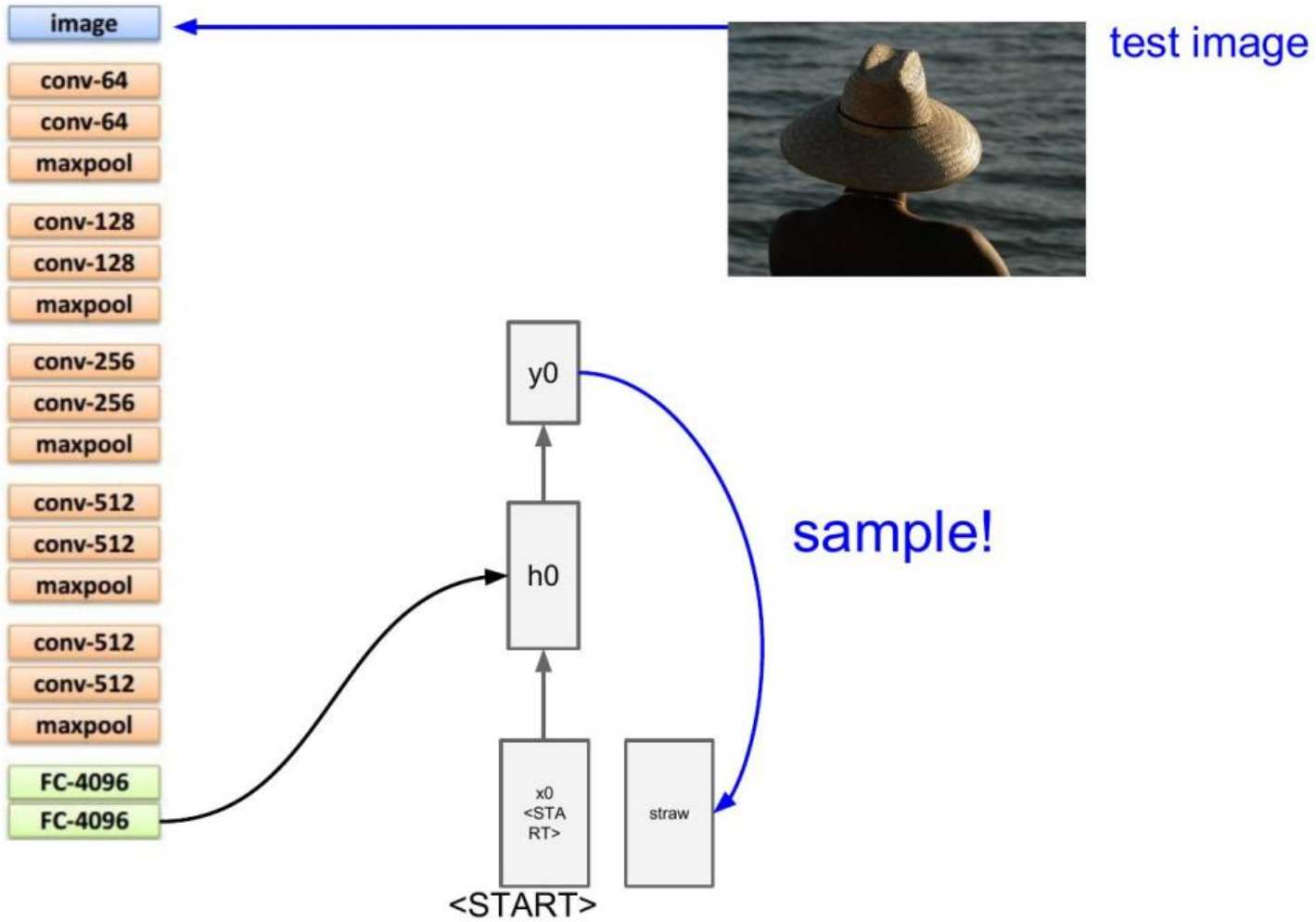
Podemos combinar CNN y RNN, aprovechando las fortalezas de cada una. Un ejemplo es *image captioning*



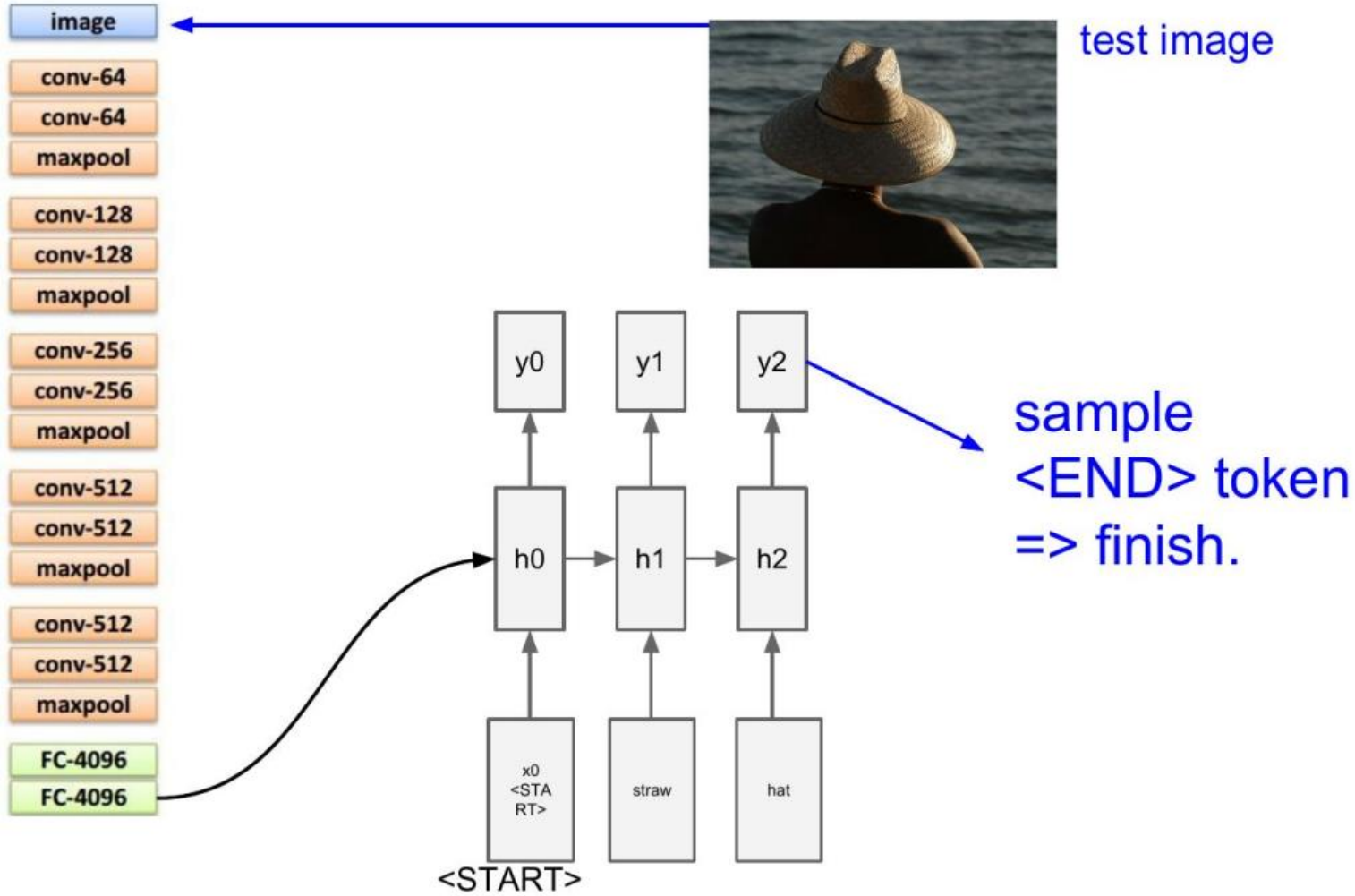
Podemos combinar CNN y RNN, aprovechando las fortalezas de cada una. Un ejemplo es *image captioning*



Podemos combinar CNN y RNN, aprovechando las fortalezas de cada una. Un ejemplo es *image captioning*



Podemos combinar CNN y RNN, aprovechando las fortalezas de cada una. Un ejemplo es *image captioning*



Podemos combinar CNN y RNN, aprovechando las fortalezas de cada una. Un ejemplo es *image captioning*

A cat sitting on a suitcase on the floor

A cat is sitting on a tree branch

A dog is running in the grass with a frisbee

Two people walking on the beach with surfboards

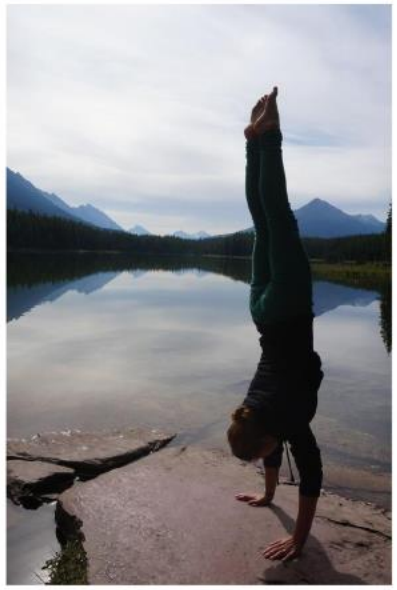
A tennis player in action on the court

Two giraffes standing in a grassy field

Podemos combinar CNN y RNN, aprovechando las fortalezas de cada una. Un ejemplo es *image captioning*

A woman is holding a cat in her hand

A person holding a computer mouse on a desk



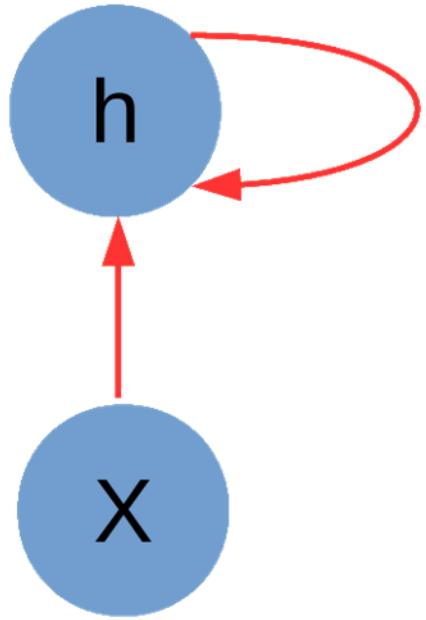
A woman standing on a beach holding a surfboard

A bird is perched on a tree branch

A man in a baseball uniform throwing a ball

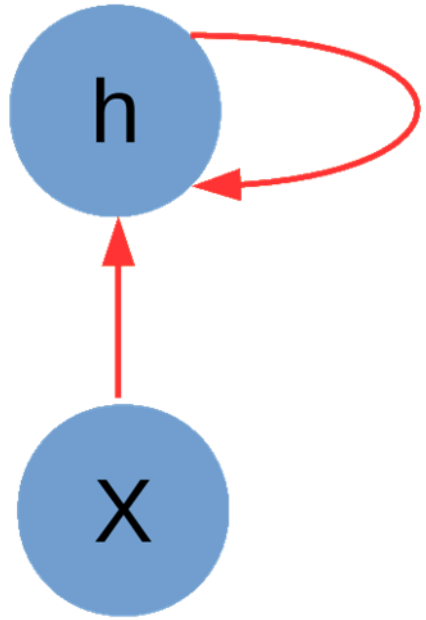
Vayamos ahora como entrenar una RNN

- Las operaciones en una RNN pueden descomponerse generalmente en tres bloques:
  1. Desde la entrada hasta el estado oculto W_{xh}
  2. Dese el estado oculto anterior al nuevo W_{hh}
  3. Desde el estado oculto a la salida W_{hy}
- El entrenamiento requiere lógicamente definir una función de pérdida adecuada a la tarea (cross-entropy, ranking, MSE, etc.).

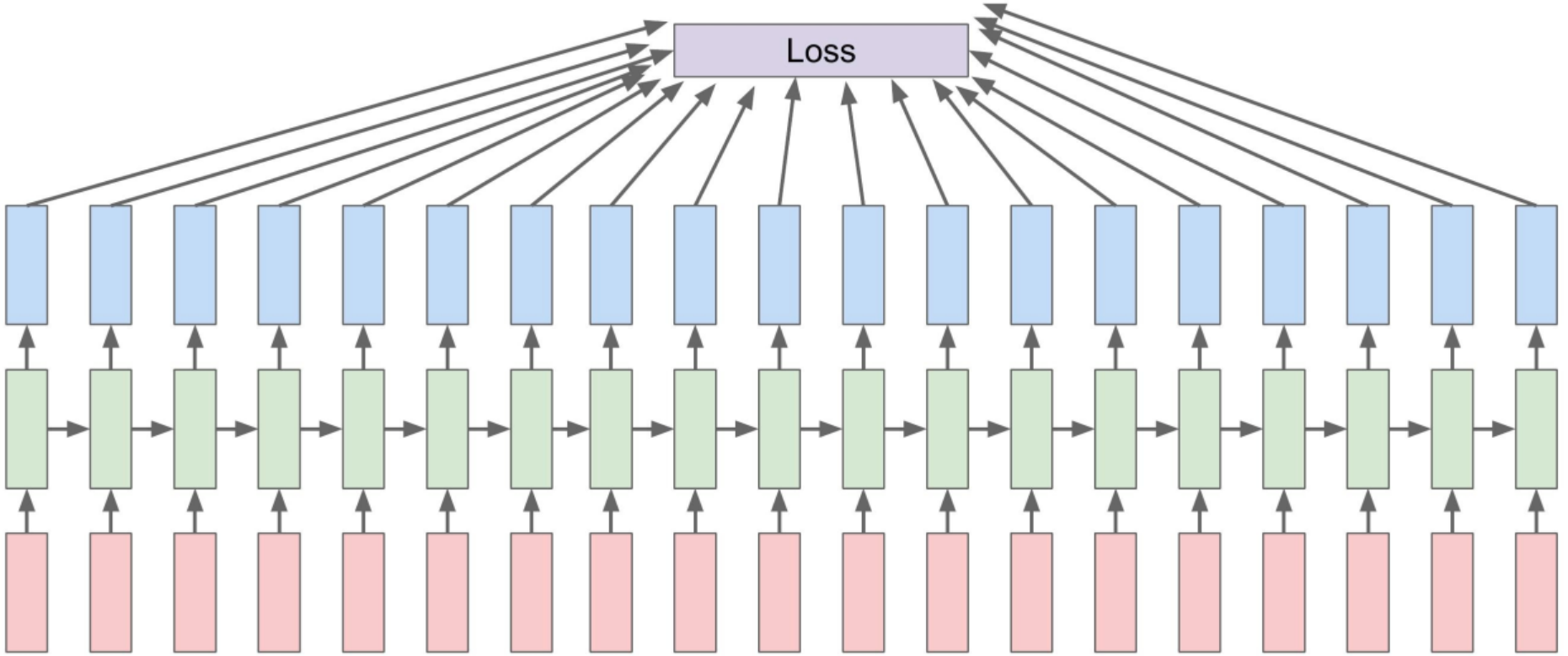


Vayamos ahora como entrenar una RNN

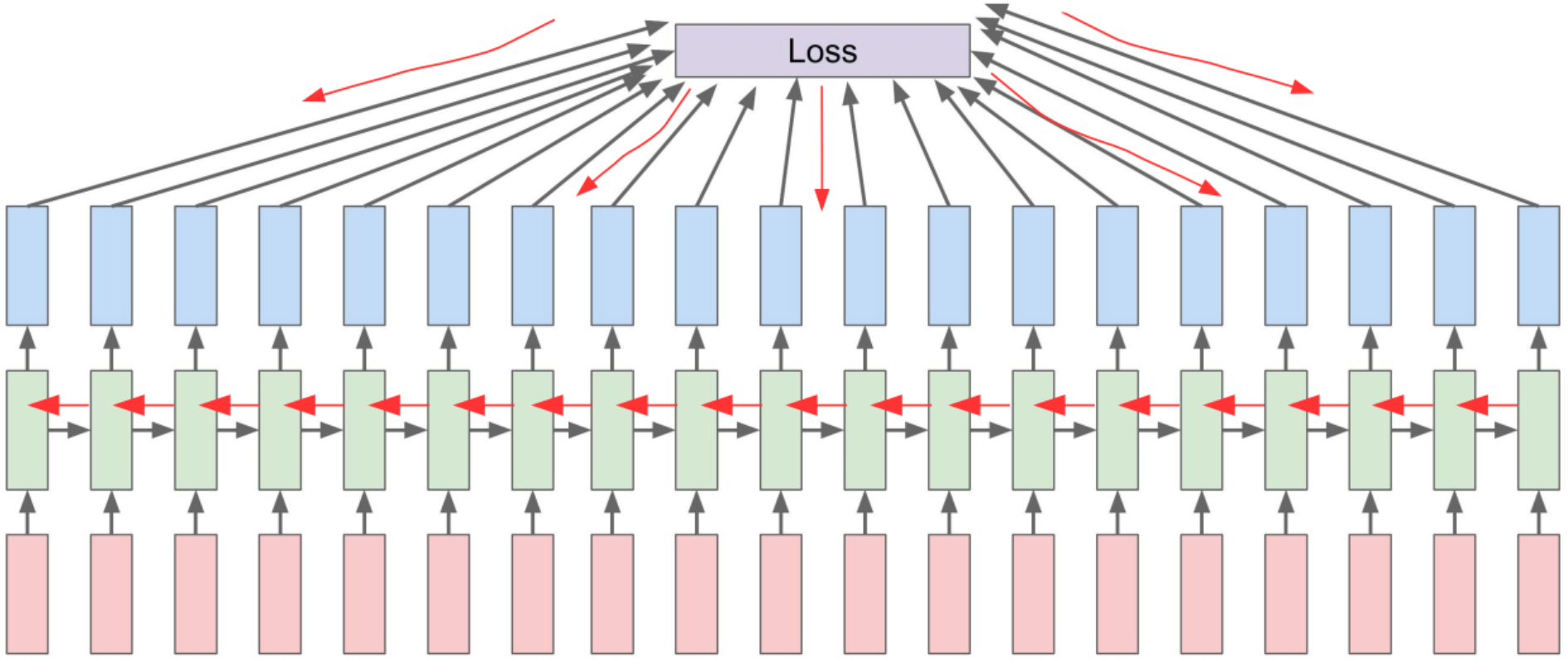
- Al igual que en las otras redes que hemos visto, la minimización de la función objetivo se hace con SGD.
- Dado que una RNN desenrollada es muy similar a un MLP, para calcular los gradientes basta aplicar *backpropagation* al grafo de cómputo (desenrollado).
- Esta técnica se conoce como *backpropagation trough time* (BPTT).



Realizamos los cálculos “hacia adelante” a través de la secuencia completa, para calcular la pérdida

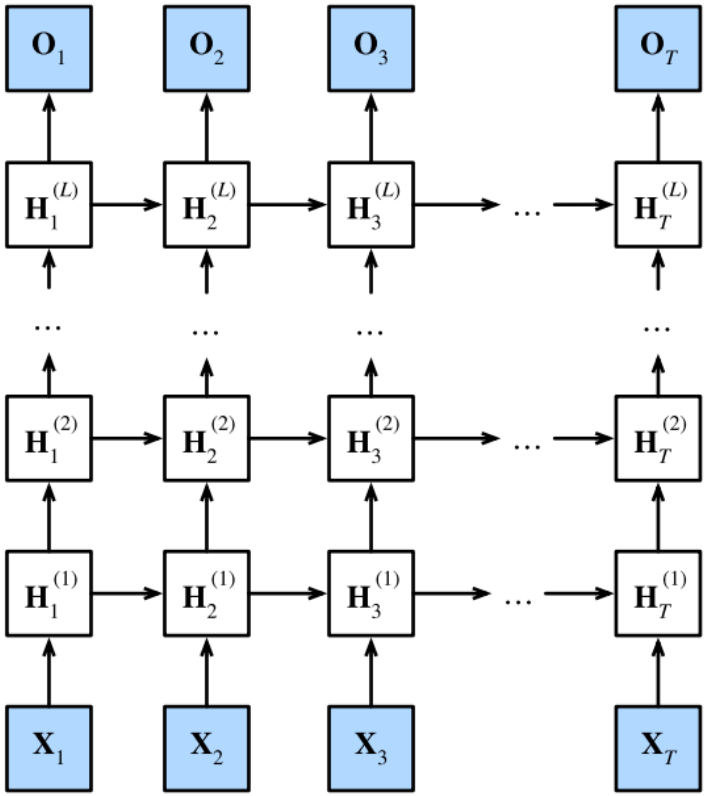


Luego propagamos la señal de error hacia atrás, a través de la secuencia, calculando y acumulando los gradientes (pesos son compartidos entre pasos *t*)



¿Qué problemas puede traer este esquema para calcular los gradientes?

Luego propagamos la señal de error hacia atrás, a través de la secuencia, calculando y acumulando los gradientes (pesos son compartidos entre pasos *t*)



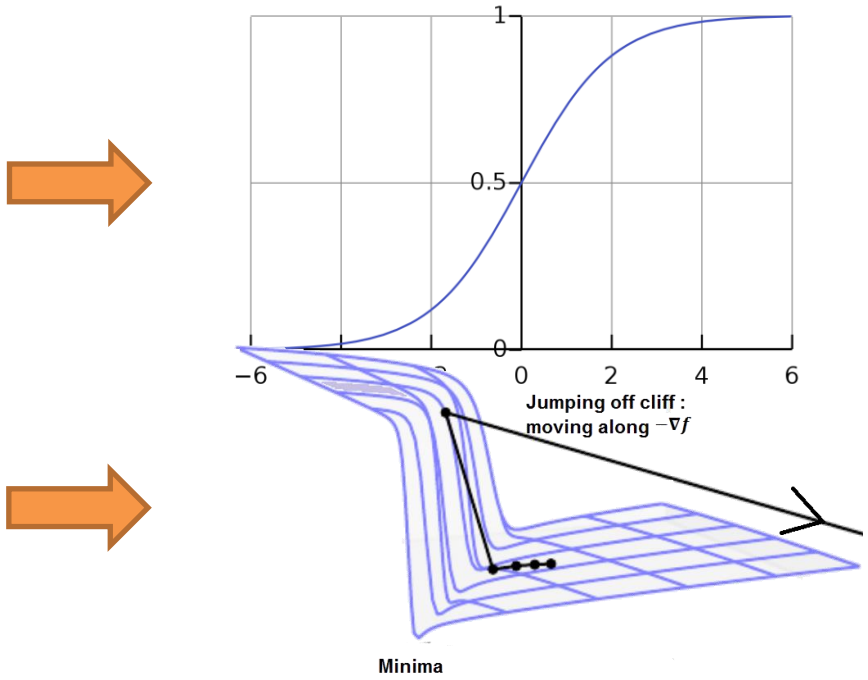
Esto se complica aún más si tenemos una RNN con múltiples capas

Problema central son las secuencias largas y su tiempo de cómputo

- Dada la estructura recursiva, el cálculo del gradiente es excesivamente demandante en recursos, pero…
- …una solución burdamente simple es truncarlo en cierto t < T (*truncated backprop*).
- De esta forma, el modelo se centrará más en la influencia de las dependencia cercanas que las lejanas.
- En la práctica, eso funciona muy bien y además evita el sobreentrenamiento (alta varianza por efecto mariposa).

Un segundo problema con las secuencias largas son los *exploding* & *vanishing gradients*

- Regla de la cadena implica el producto repetido de sigmoides, lo que implica una reducción progresiva de la norma del gradiente (*vanishing gradient*)
- Podríamos entonces cambiar la función de activación por una ReLU, pero esto lleva un aumento progresivo de la norma del gradiente (exploding gradient).
- Es posible evitar esto si se trunca el gradiente, pero el rendimiento se ve afectado.
- Numéricamente, estos dos problemas son caracterizados por el valor singular (SVD) más grande de las matrices W (< 1 gradiente se desvanece, > 1 gradiente explota), lo que no deja mucho espacio de maniobra para esta arquitectura.



Conceptualmente, el problema subyacente es la pérdida de memoria por parte de la RNN

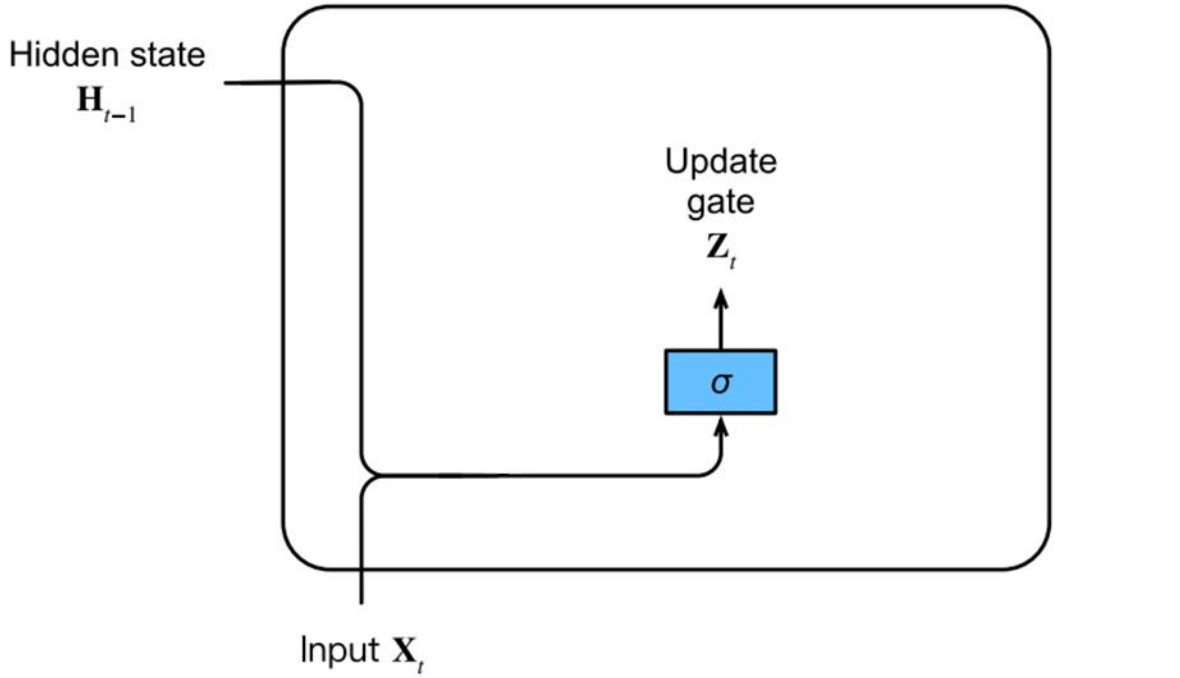
Revisitemos la regla de actualización para una RNN

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

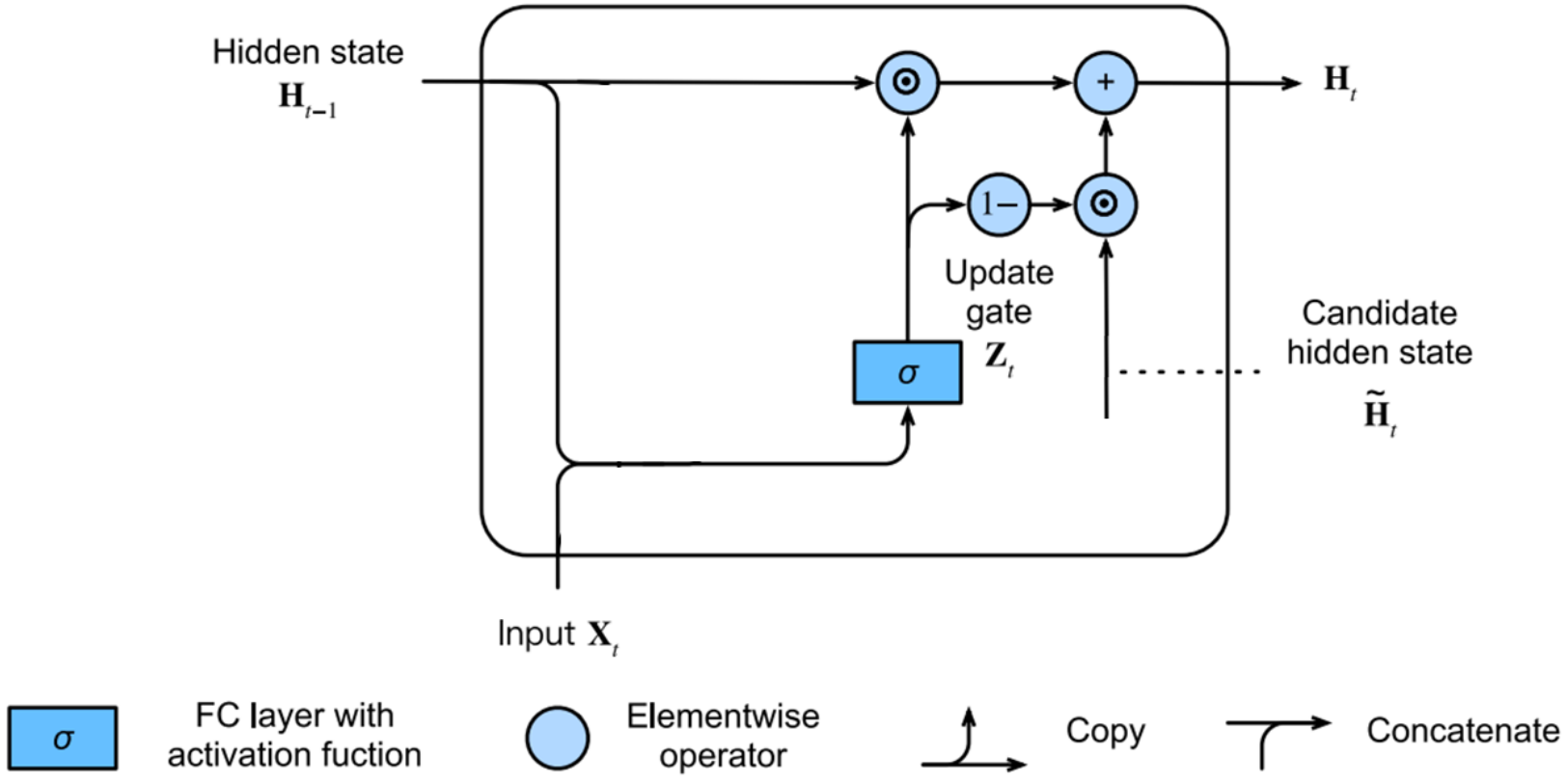
Podemos notar que para cada instante t, la memoria h_t es completamente recalculada, estando (casi) completamente a merced del efecto de x_t

Para evitar esto, es necesario utilizar una arquitectura distinta. Veremos dos ejemplos: Gated Recurrent Unit (GRU) y *Long short-term memory network* (LSTM).

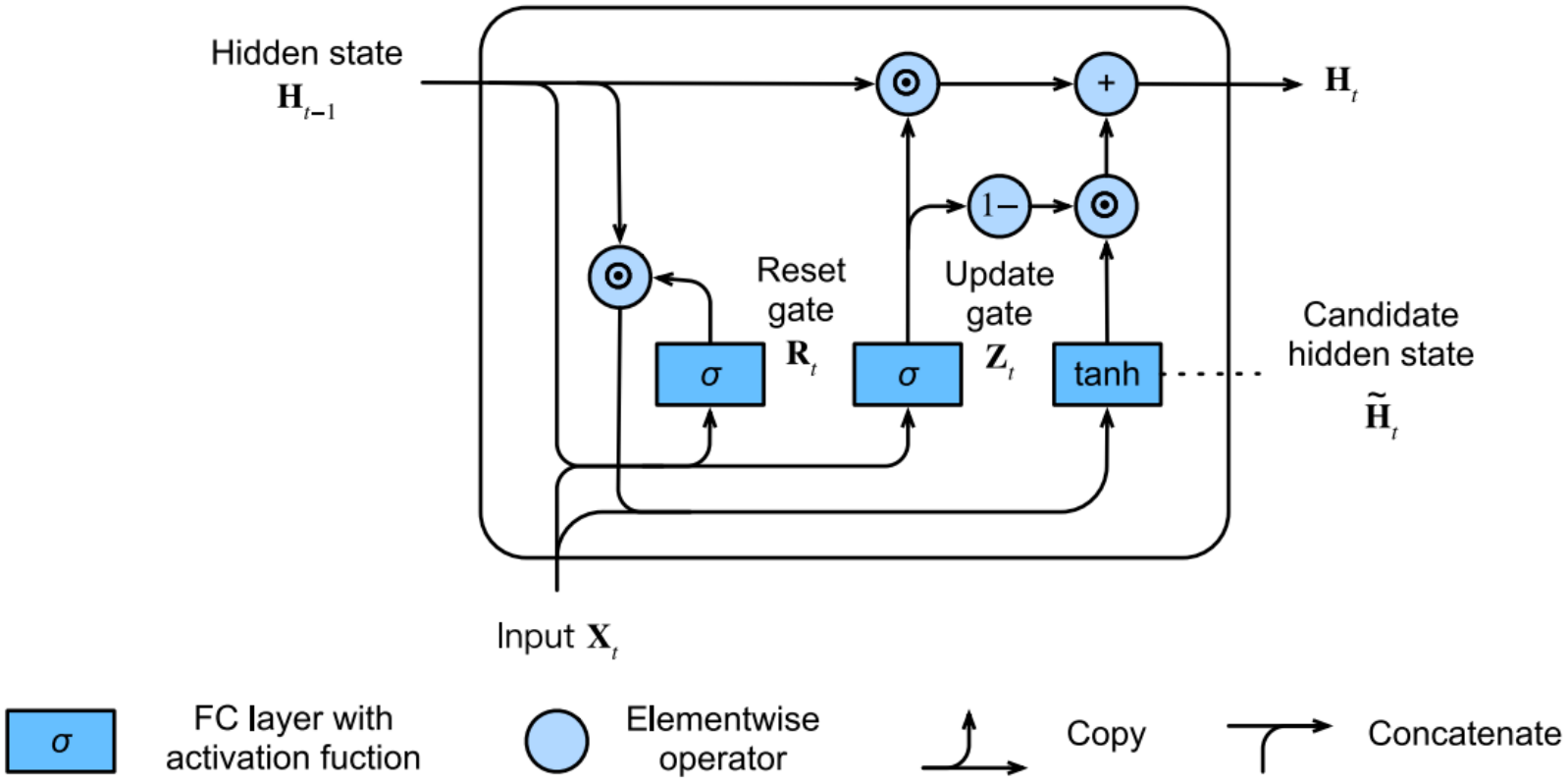
Como elemento principal, una GRU considera una compuerta de actualización de la memoria



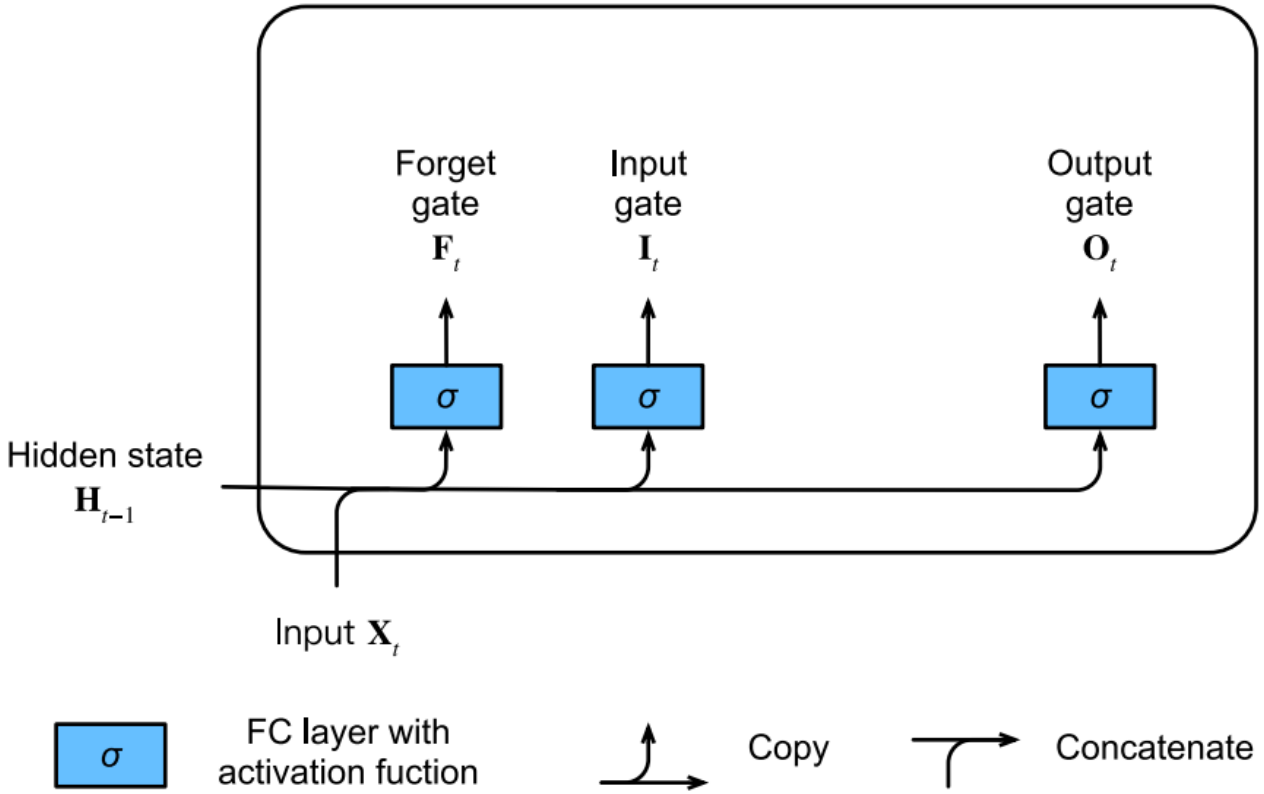
Esta compuerta controla cuanto se “olvida”
y cuanto se agrega a la memoria



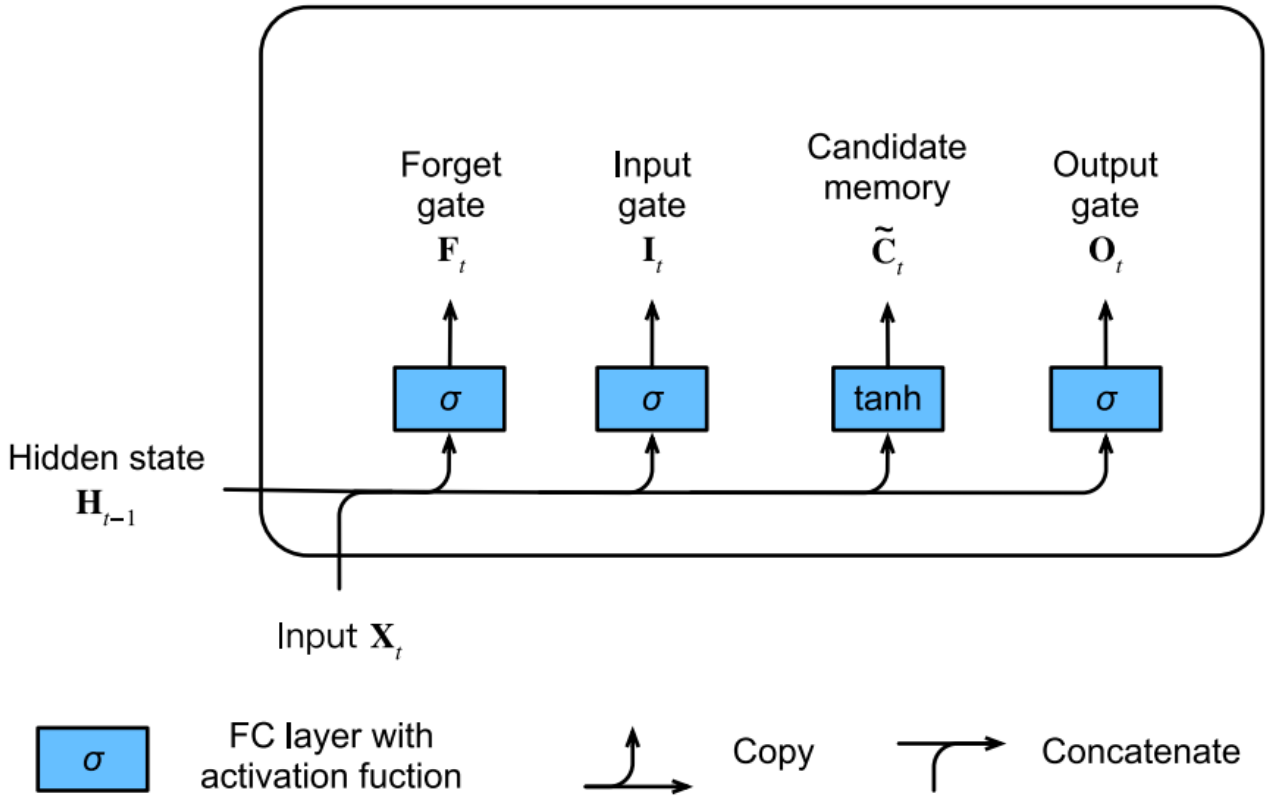
Lo que se “agrega” es a su vez modulado por una compuerta de reset (que cosas de la memoria son necesarias para actualizar el estado)



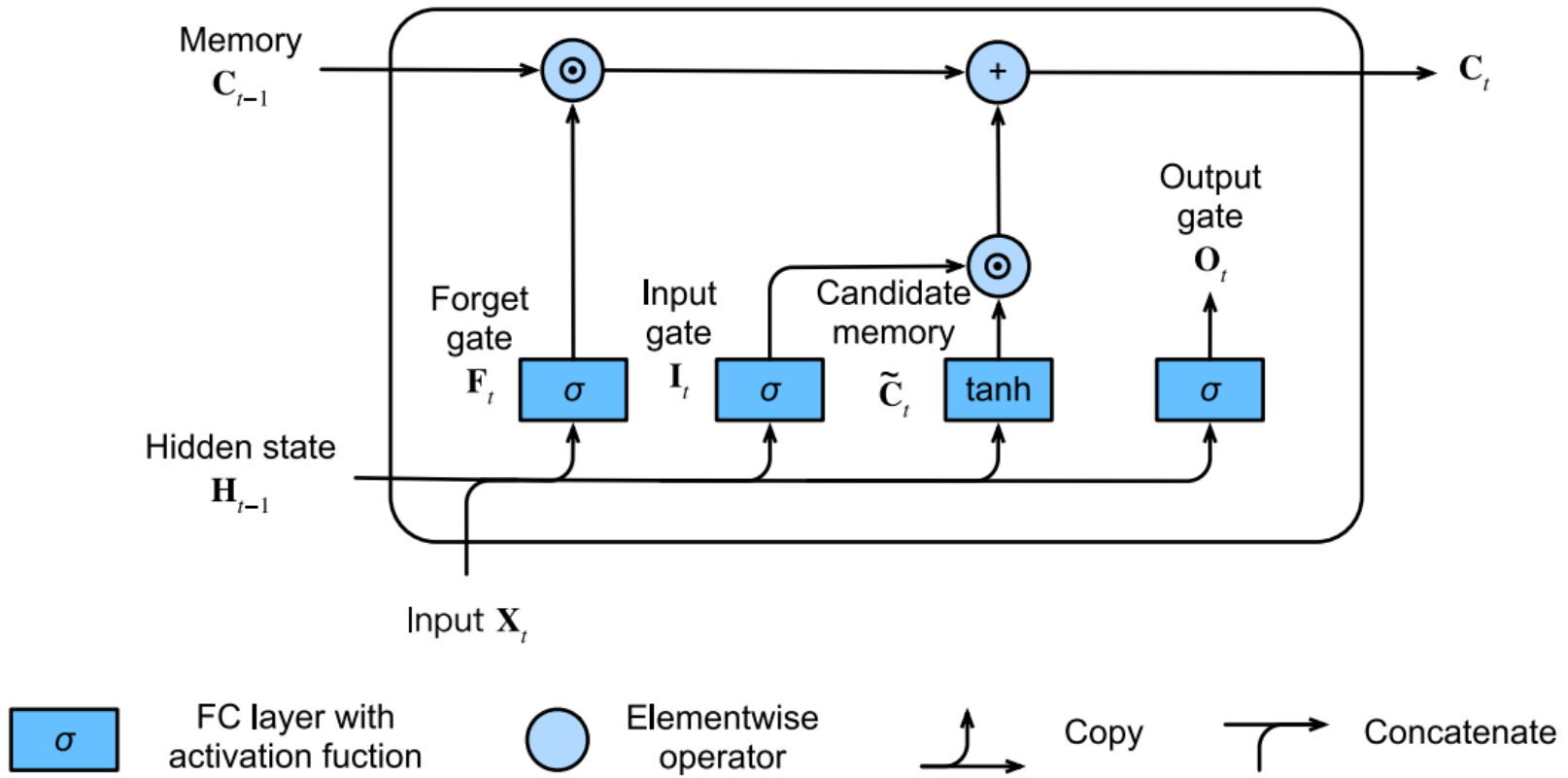
LSTMs extienden la idea de compuertas (pero son previas a GRU)



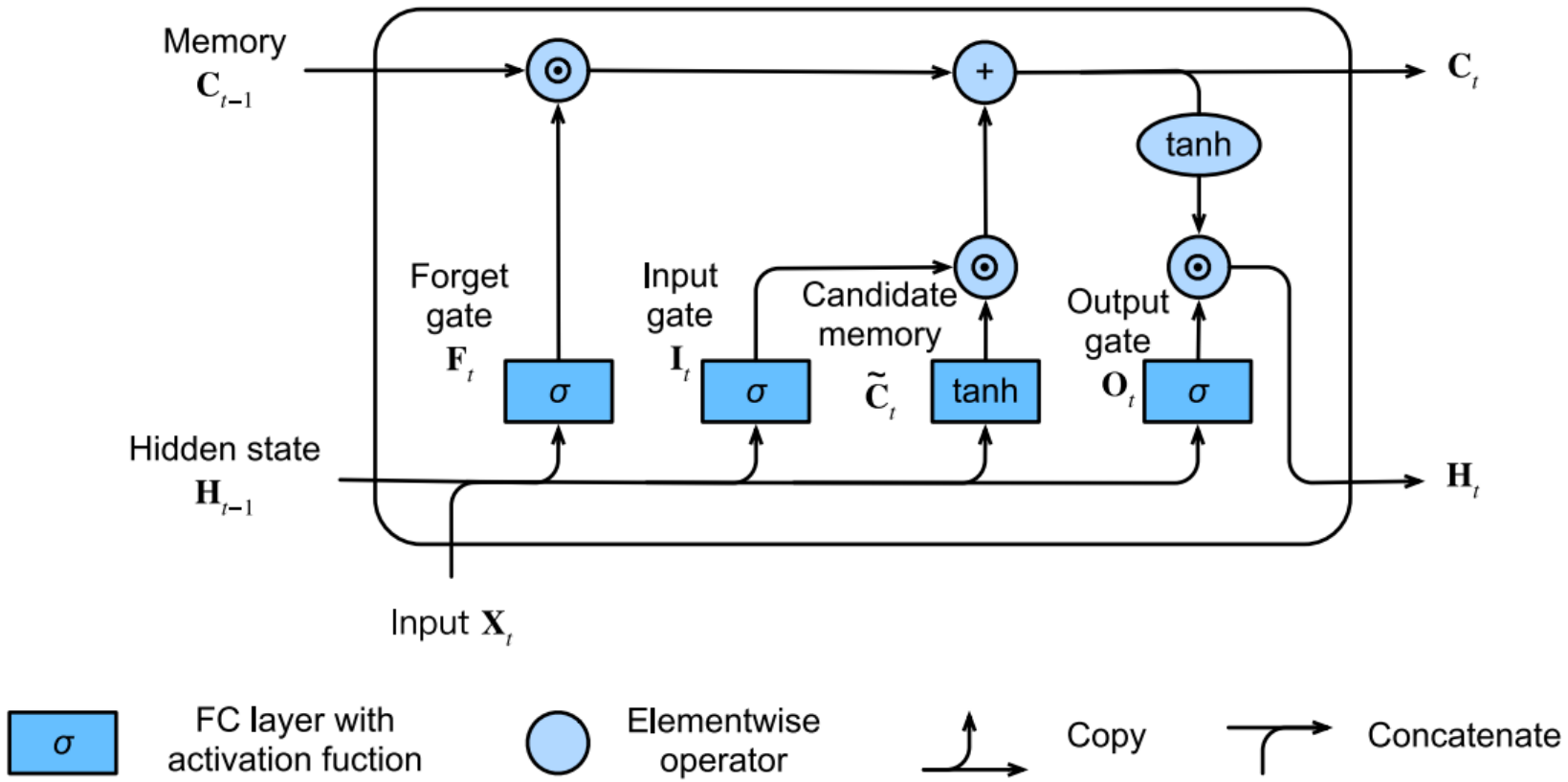
LSTMs separan explícitamente memoria de estado



LSTMs separan explícitamente memoria de estado

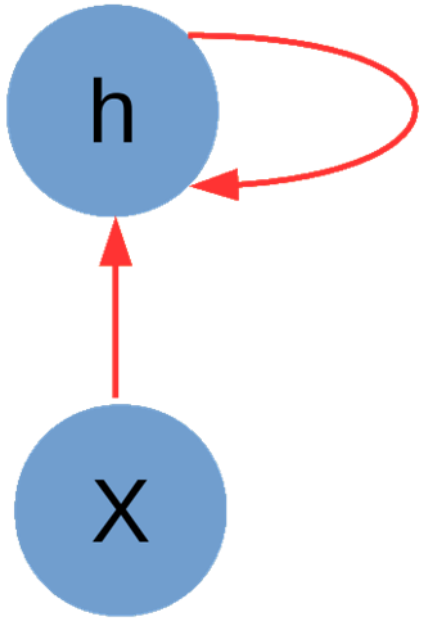


Esto les permite tener un flujo del gradiente menos accidentado



Cuáles son los conceptos centrales de esta primera parte del capítulo

- Redes recurrentes permiten modelar secuencias de largo arbitrario.
- El compartir parámetros de manera profunda a través de conexiones recurrentes les entrega gran poder.
- Alta flexibilidad y versatilidad les permite ser usadas en muchos dominios y acoplarse fácilmente con otras redes.
- Secuencias largas generan problemas en la memoria (flujo del gradiente). Arquitecturas GRU y LSTM permiten controlar esto.
- En la práctica LSTM y GRU son las arquitecturas más usadas.



Algunas lecturas de profundización

- Understanding LSTM Networks: http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- Exploring LSTMs: http://blog.echen.me/2017/05/30/exploring-lstms/
- Visualizing memorization in RNNs: https://distill.pub/2019/memorization-in-rnns/
- The Unreasonable Effectiveness of Recurrent Neural Networks: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ingeniería de Transporte y Logística

Sistemas Urbanos Inteligentes

Redes Neuronales Recurrentes (RNN)

Hans Löbel
Dpto. Ingeniería de Transporte y Logística
Dpto. Ciencia de la Computación