



# Redes Neurais Recorrentes (RNN)

*Erick nicolas*

*Gabriel porto*

*ELC1014-INTELIGÊNCIA ARTIFICIAL*

*"O passado alimenta o presente para que possamos  
antecipar o futuro."*

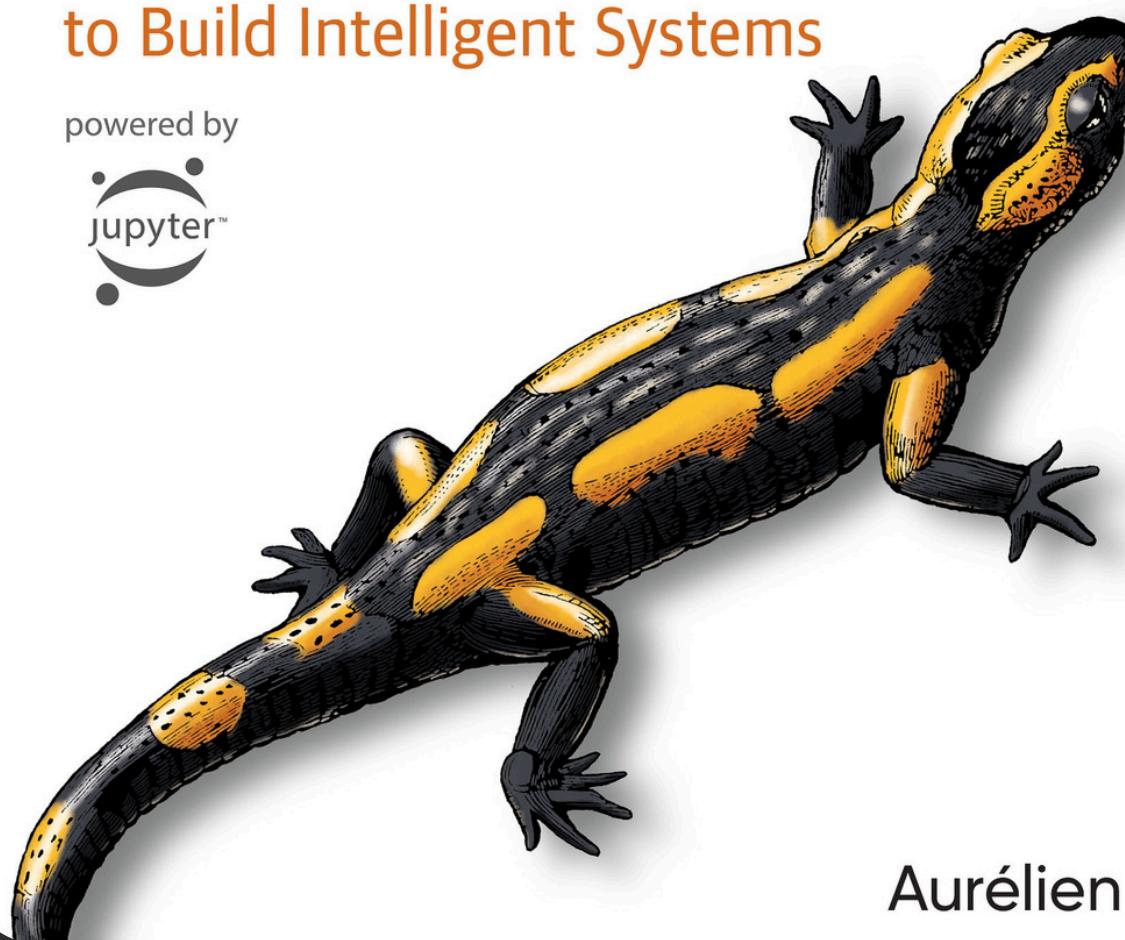
O'REILLY®

Third  
Edition

# Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow

Concepts, Tools, and Techniques  
to Build Intelligent Systems

powered by



Aurélien Géron

# bibliografia

*Cap 15- Sequências de processamento usando  
RNNs e CNNs*

*"This book is a great introduction to the theory  
and practice of solving problems with neural  
networks; I recommend it to anyone  
interested in learning about practical ML."*

Pete Warden  
Mobile Lead for TensorFlow

# topicos

*O que são redes neurais recorrentes*

*Para que servem*

*Neurônios*

*Camadas Recorrentes*

*Células de memória*

*Sequências de entrada e saída*

*Treinamento de RNNs*

*Previsão usando uma RNN*

*Manipulando sequências longas*

*Combatendo o problema dos gradientes instáveis*

*Lidando com o problema da memória de curto prazo*

# O que são redes neurais recorrentes

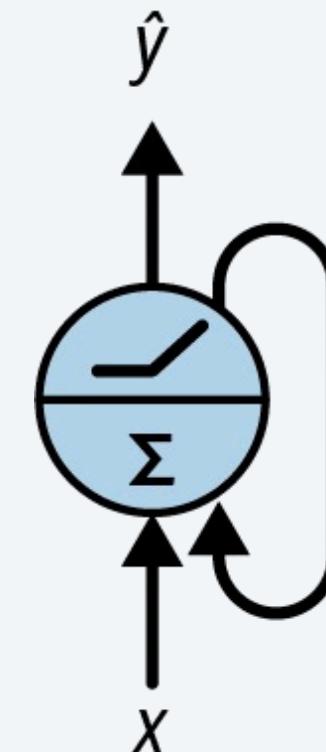
RNN é uma rede com loop interno que permite o processamento sequencial. Diferente das redes feedforward, possuem conexões que apontam para trás.

Possue Memória Temporal, Ideal para análise de séries temporais.

- Capacidade de "lembrar" informações anteriores da sequência.
- Ideal para tarefas onde a ordem dos dados importa.

Aplicações Típicas

- Análise de Séries Temporais
- Temperatura horária em uma cidade
- Trajetórias de carros próximos



# Para que servem

## **Previsão de Séries Temporais**

Antecipar valores futuros com base em dados históricos (ex: preços de ações, clima, demanda de energia).

- Reconhecimento de Fala e Comandos de Voz

Permite a compreensão e interpretação de áudio por máquinas (ex: assistentes virtuais como Alexa e Siri).

- Tradução Automática entre Idiomas

Converte textos de uma língua para outra com contexto e fluidez (ex: português → inglês).

- Geração de Texto

Criação de frases e parágrafos coerentes a partir de entradas curtas (ex: geração automática de legendas, respostas em chatbots).

- Análise de Sentimentos

Reconhece letras escritas à mão analisando o traçado contínuo (usado em aplicativos de anotações ou tablets).

- Predição de Comportamento do Usuário

Usado em recomendações (ex: o que você vai assistir ou comprar em seguida), com base em interações passadas.

**Ou seja, RNNs são ideais para resolver qualquer problema que envolva a memória de sequências de dados ao longo do tempo.**

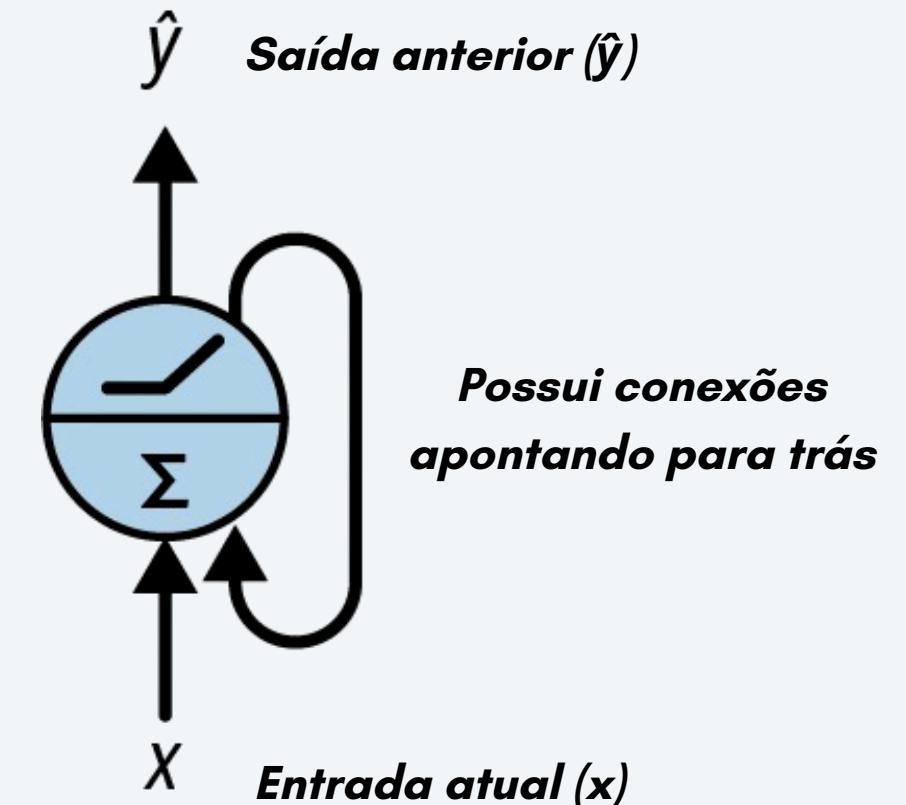
# Neurônios

As ativações não fluem mais apenas em uma direção, como nas redes feedforward

**Cada neurônio possui dois conjuntos de pesos:**

$W_x$ : pesos aplicados à entrada atual ( $x$ )

$W\hat{y}$ : pesos aplicados à saída anterior ( $\hat{y}$ )



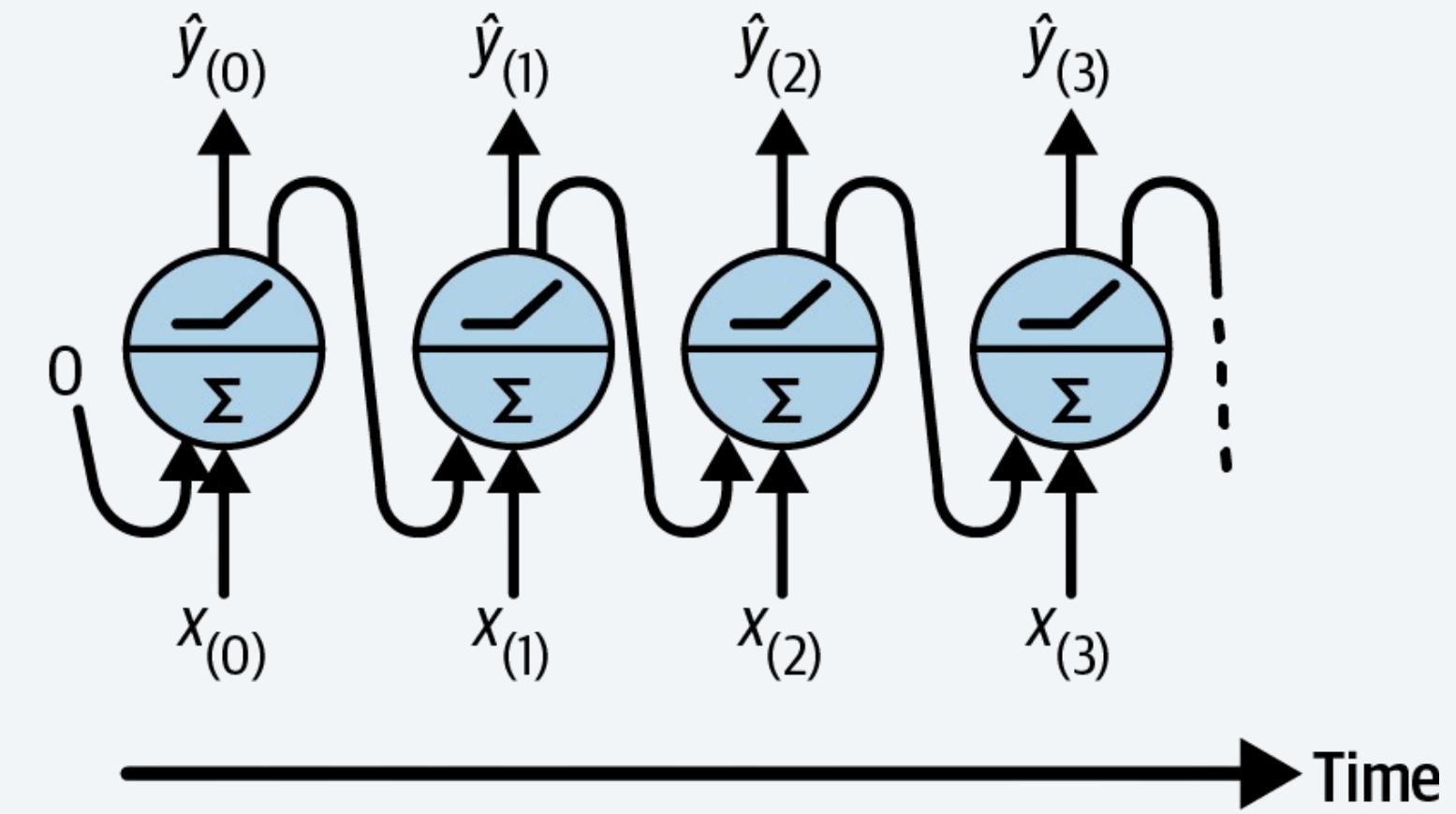
*Equation 15-1. Output of a recurrent layer for a single instance*

$$\hat{y}_{(t)} = \varphi(W_x^T x_{(t)} + W_{\hat{y}}^T \hat{y}_{(t-1)} + b)$$

Entradas e Saídas

Entrada ( $x$ ): vetor que pode ter qualquer tamanho, representando uma sequência (ex: temperatura ao longo do dia).

Saída ( $\hat{y}$ ): também é uma sequência de vetores, processada etapa por etapa.



# Camadas Recorrentes

Cada neurônio recorrente tem dois conjuntos de pesos:  
um para as entradas  $x(t)$  e o outro para as saídas do passo de tempo anterior,  $\hat{y}(t-1)$ .  
podemos colocar todos os vetores de peso em duas matrizes de peso:  $W_x$  e  $W_{\hat{y}}$ .

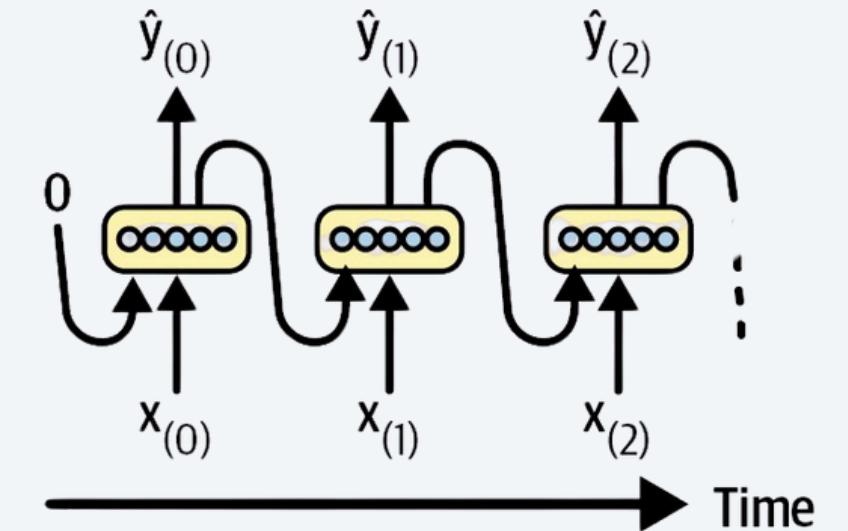
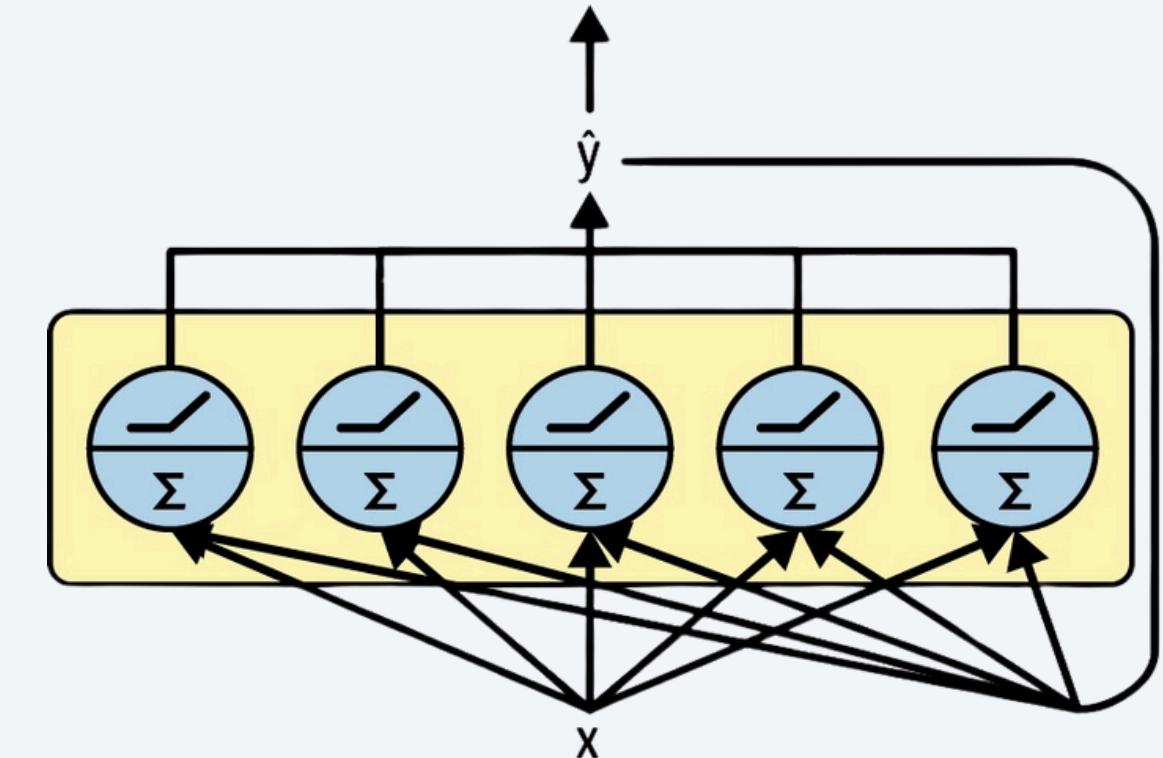
Onde  $b$  é o vetor de polarização e  $\phi(\cdot)$  é a função de ativação (por exemplo, ReLU1).

*Equation 15-2. Outputs of a layer of recurrent neurons for all instances in a pass:  
[mini-batch]*

$$\begin{aligned}\hat{Y}_{(t)} &= \varphi(\mathbf{X}_{(t)} \mathbf{W}_x + \hat{Y}_{(t-1)} \mathbf{W}_{\hat{y}} + \mathbf{b}) \\ &= \varphi([\mathbf{X}_{(t)} \quad \hat{Y}_{(t-1)}] \mathbf{W} + \mathbf{b}) \text{ with } \mathbf{W} = \begin{bmatrix} \mathbf{W}_x \\ \mathbf{W}_{\hat{y}} \end{bmatrix}\end{aligned}$$

Entradas e Saídas

- A entrada ( $x$ ) é um vetor ou sequência de vetores, representando dados temporais (ex.: temperatura ao longo do dia).
- A camada recorrente processa a sequência passo a passo, mantendo memória dos estados anteriores.
- A saída ( $\hat{y}$ ) é uma sequência de vetores, com um vetor gerado em cada etapa de tempo.



# Células de memória

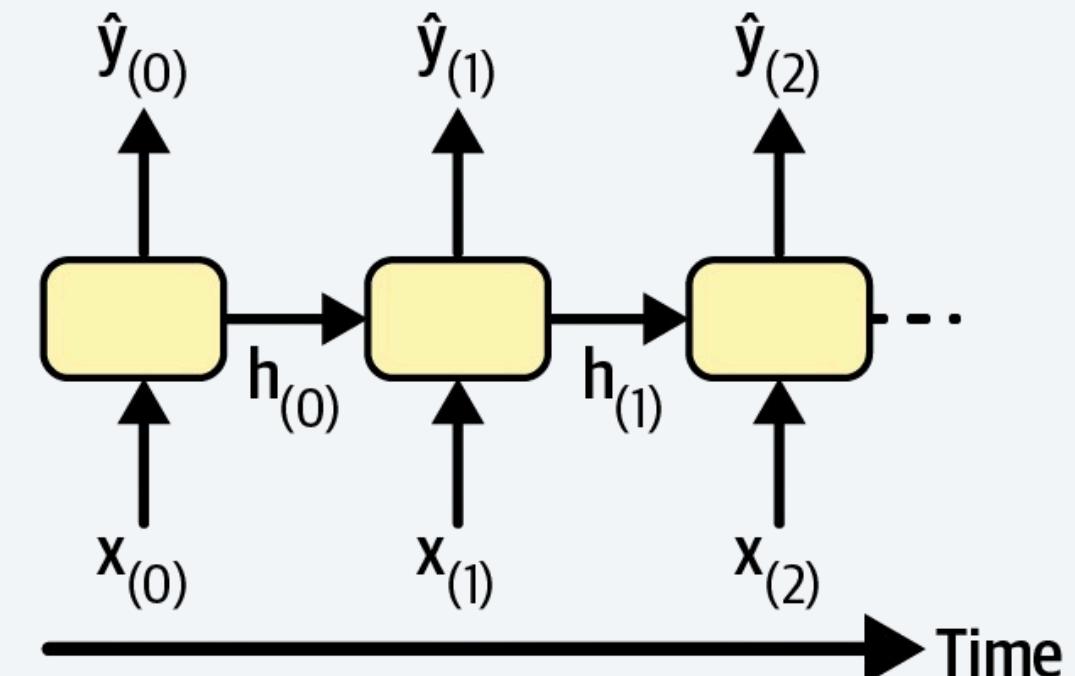
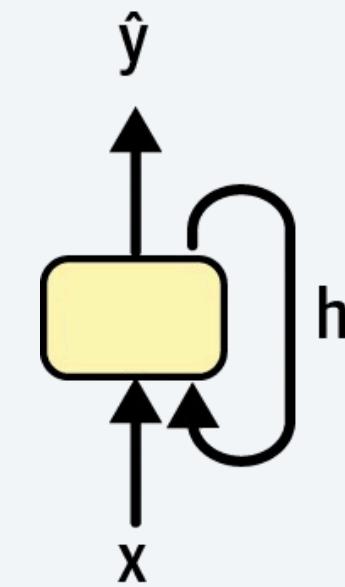
**A parte de uma rede neural que preserva algum estado ao longo de passos de tempo é chamada de célula de memória**

## Estado de uma Célula na RNN

- No tempo  $t$ , o estado oculto é denotado por  $h(t)$ .
- $h(t)$  depende de:
  - Entrada atual:  $x(t)$ .
  - Estado anterior:  $h(t-1)$ .
- Expressão matemática:
  - $h(t) = f(x(t), h(t-1))$
- Permite à rede acumular e processar informações ao longo do tempo.

## Saída na RNN

- Saída no tempo  $t$ :  $\hat{y}(t)$ .
- Depende do estado atual:  $h(t)$ .
- Em células simples:  $\hat{y}(t) = h(t)$ .
- Células mais complexas podem ter saídas distintas.



# Sequências de entrada e saída

## Sequência para Sequência

Modelo que aprende uma sequência de entrada e gera uma sequência de saída.

Exemplo: prever o consumo diário de energia com base nos últimos  $N$  dias, estimando o valor do próximo dia.

## Sequência para Vetor

Recebe uma sequência (ex: texto) e gera uma única saída.

Exemplo: analisar uma crítica de filme e retornar uma pontuação de sentimento entre 0 (ódio) e 1 (amor).

## Vetor para Sequência

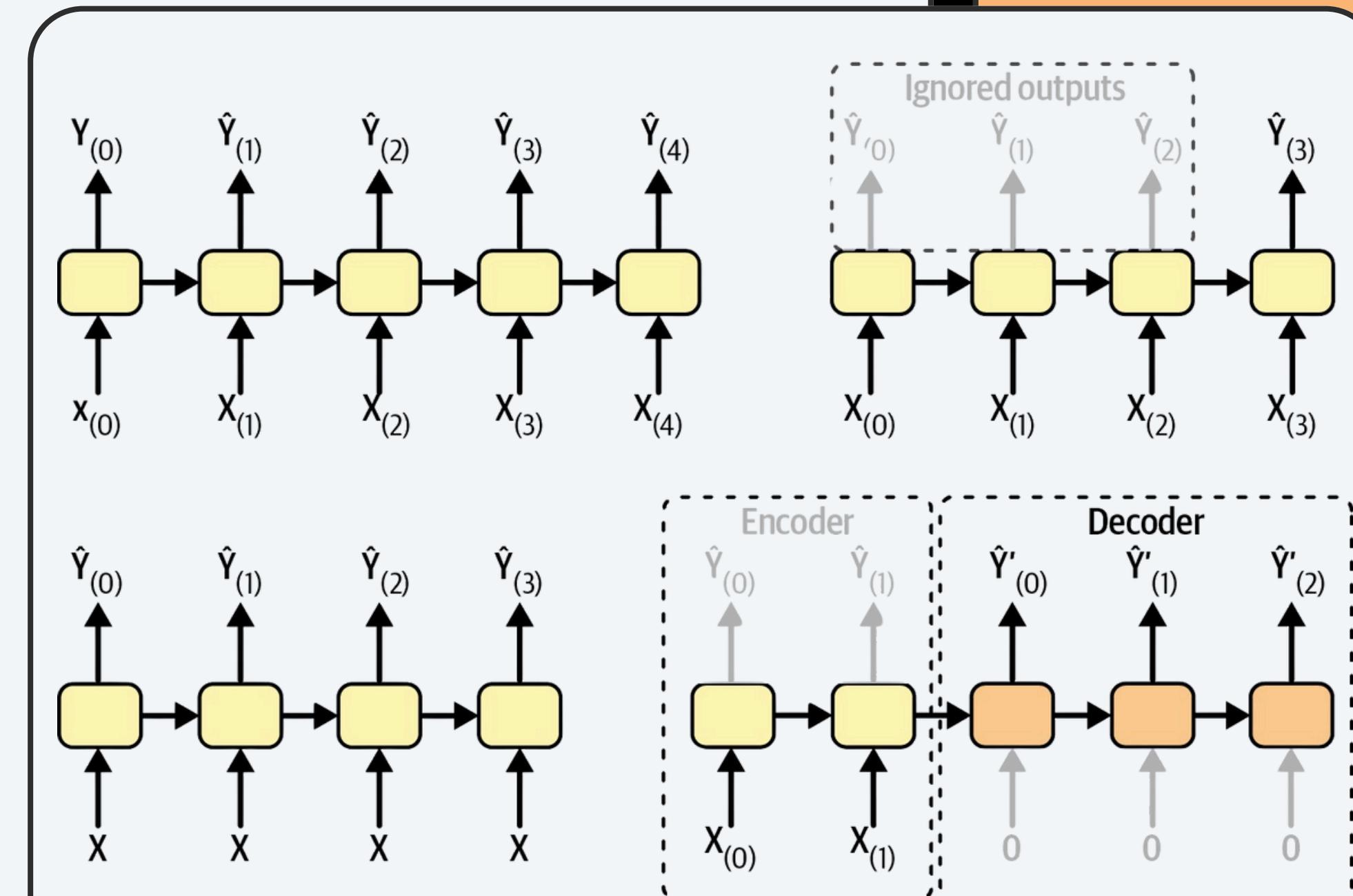
Recebe uma entrada fixa e gera uma sequência como saída.

Exemplo: gerar uma legenda descritiva para uma imagem analisada

## Codificador-Decodificador

Converte uma sequência de entrada em uma representação vetorial e depois gera uma nova sequência a partir dela.

Exemplo: traduzir uma frase de um idioma para outro.



# Treinamento de RNNs

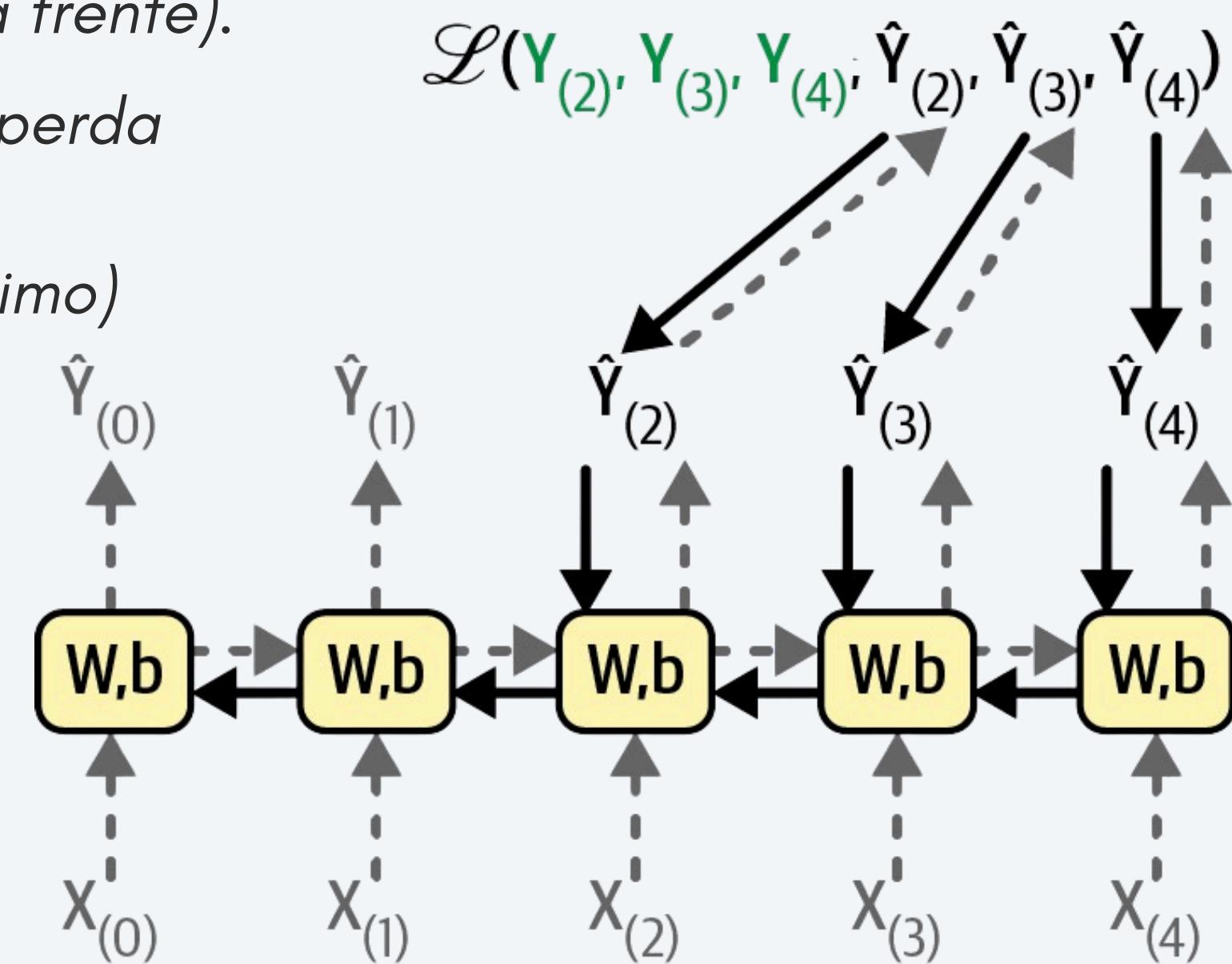
A RNN é "desenrolada no tempo" e treinada com retropropagação comum aplicada em cada etapa. Essa técnica é chamada de retropropagação ao longo do tempo (BPTT).

1-os dados passam pela RNN desenrolada (passagem para frente).

2-Depois, a saída gerada é avaliada com uma função de perda para medir o erro.

onde  $Y(i)$  é o ( $i$ ) é a previsão  $i$  e  $T$  é o passo de tempo máximo)

- A perda considera só as últimas saídas ( $\hat{Y}(2)$ ,  $\hat{Y}(3)$ ,  $\hat{Y}(4)$ ).
- Os gradientes fluem apenas dessas saídas e são acumulados nos pesos compartilhados ( $W$  e  $b$ ).
- Após o cálculo dos gradientes, o BPTT atualiza os parâmetros como no backprop tradicional.



\*A função pode ignorar algumas saídas. Por exemplo, em uma RNN de sequência para vetor, todas as saídas são ignoradas, exceto a última.

# Previsão usando uma RNN simples

A RNN mais básica, contendo uma única camada recorrente com apenas um neurônio recorrente

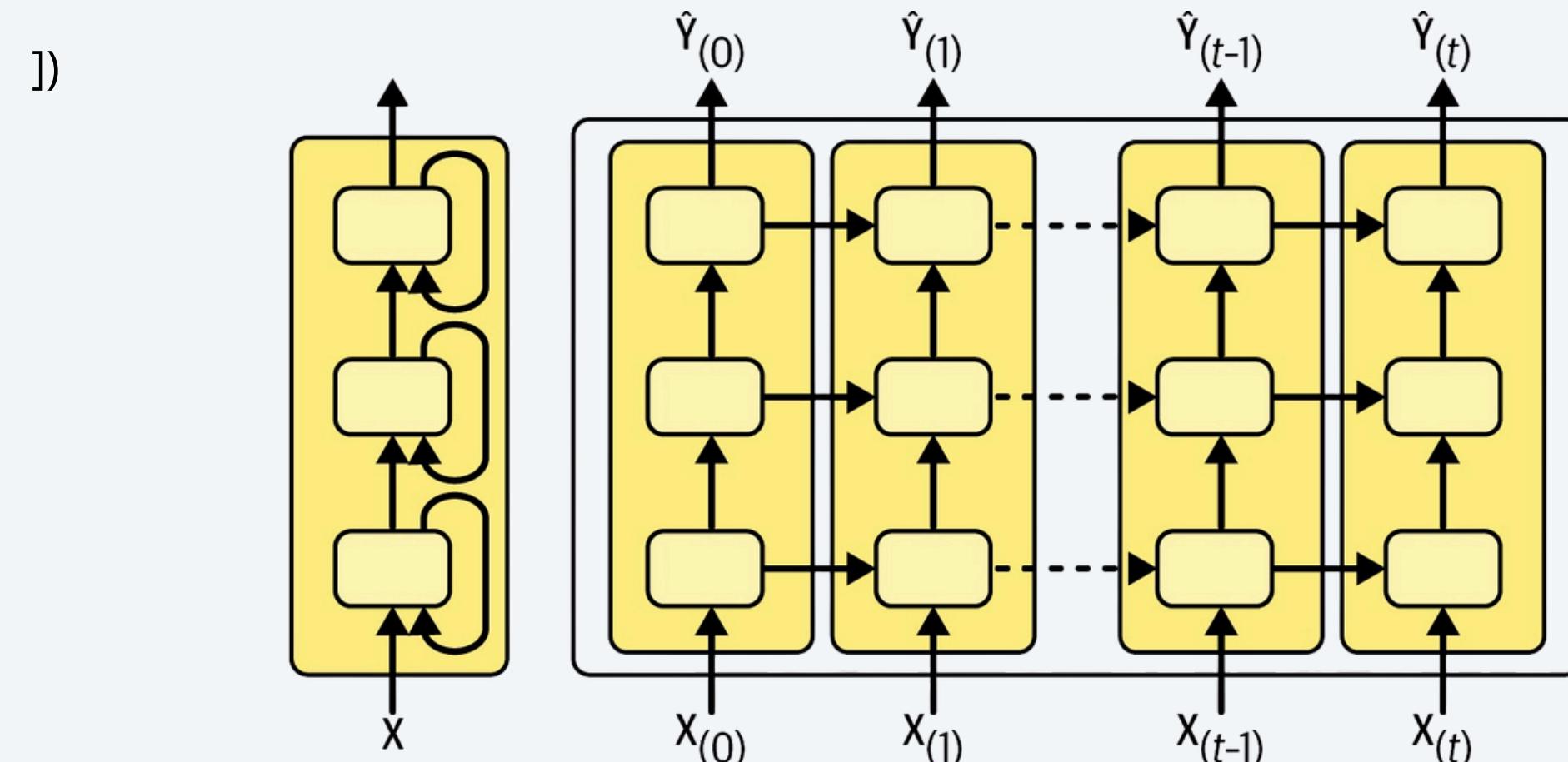
```
univar_model = tf.keras.Sequential([
    tf.keras.layers.SimpleRNN(32, input_shape=[None, 1]),
    tf.keras.layers.Dense(1) # no activation function by default
])
```

# Previsão usando uma RNN profunda

Uma RNN profunda (esquerda) desenrolada ao longo do tempo (direita).

O exemplo utiliza três camadas SimpleRNN . As duas primeiras são sequência-a-sequência, a última é sequência-vetor. Depois, uma camada densa gera a previsão final. as saídas intermediárias são ignoradas e só a última saída alimenta a camada densa para produzir a previsão.

```
deep_model = tf.keras.Sequential([
    tf.keras.layers.SimpleRNN(32, return_sequences=True, input_shape=[None, 1]),
    tf.keras.layers.SimpleRNN(32, return_sequences=True),
    tf.keras.layers.SimpleRNN(32), tf.keras.layers.Dense(1)
])
```



# Manipulando sequências longas

***Para treinar uma RNN em sequências longas, precisamos executá-la em vários passos de tempo, tornando a RNN desenrolada uma rede muito profunda.***

## Problemas das RNNs

- *Gradientes instáveis:*
- *Com redes profundas, as RNNs sofrem com o problema dos gradientes que podem explodir ou desaparecer.*
  - *Gradientes desaparecendo: a rede deixa de aprender dependências de longo prazo.*
  - *Gradientes explodindo: os pesos crescem demais e o treinamento fica instável.*
- *Dificuldade com sequências longas:*
- *Quando o tamanho da sequência é grande demais, a RNN tende a esquecer as primeiras entradas, limitando a capacidade de capturar relações de longo prazo.*
- *Rede muito profunda:*
- *Para processar sequências longas, a RNN precisa ser desenrolada em muitos passos de tempo, funcionando como uma rede neural profunda.*

# Combatendo o problema dos gradientes instáveis

Existem técnicas que podem ser utilizadas para combater o problema dos gradientes instáveis, como:

- Inicialização e otimização:
  - Boa inicialização de parâmetros.
  - Otimizadores mais rápidos (ex.: Adam).
- Ajustar a taxa de aprendizado:
  - Utilizar taxas menores pode reduzir o risco de gradientes instáveis.
- Usar funções de ativação melhores:
  - Funções de saturação, como a tangente hiperbólica, ajudam a estabilizar o treinamento (por isso são padrão em RNNs).
- Monitoramento e controle dos gradientes:
  - Monitorar o tamanho dos gradientes (ex.: TensorBoard).
  - Aplicar gradient clipping (recorte de gradiente) quando necessário.
- Normalização:
  - Batch Normalization não é eficaz dentro das RNNs, apenas com leve benefício nas entradas (Laurent et al., 2015).
  - Layer Normalization funciona melhor em RNNs.

# Lidando com o problema da memória de curto prazo

*Devido às transformações pelas quais os dados passam ao percorrer uma RNN, algumas informações são perdidas a cada passo de tempo*

Para lidar com esse problema, vários tipos de células com memória de longo prazo foram introduzidos.

## **Células LSTM-foi proposta em 1997**

- A célula LSTM funciona como uma caixa-preta semelhante à RNN básica, mas com desempenho superior: treina mais rápido e captura melhor padrões de longo prazo.

## **Células GRU-foi proposta por Kyunghyun Cho et al. em um artigo de 2014**

- A célula GRU é uma versão simplificada da LSTM, com desempenho similar. Ela usa um único vetor de estado  $h(t)h(t)h(t)$  e uma única porta  $z(t)z(t)z(t)$ , que controla o esquecimento e a entrada. Antes de armazenar novas informações, o local é apagado. Tanto LSTM quanto GRU são responsáveis pelo sucesso das RNNs.