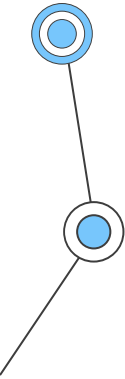
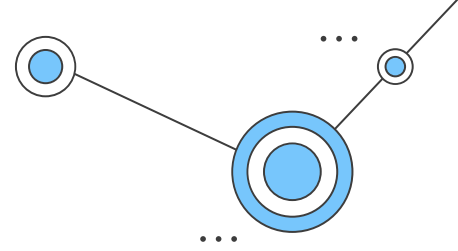


# Fundamentals of Recurrent Neural Networks (RNNs) and Long-Short Term Memory (LSTMs)



Doutoranda: Débora Lima

# Apresentação

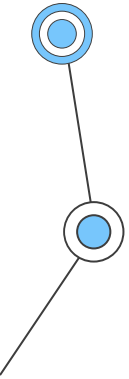


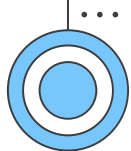
- **Formação**

- Técnica em Informática pelo IFRN - Campus Mossoró
- Bacharel em Biotecnologia pela UFRSA
- Mestre em Bioinformática
- Atualmente Doutoranda em Bioinformática

- **Linhas de Pesquisa**

- Projeto de Desenvolvimento do observatório de Dados de Câncer da Liga mossoroense de estudos e combate ao câncer
- Pesquisa com redes neurais para análise de dados clínicos de câncer de pulmão
- Pesquisa usando Deep Learning e Random Forest para análise de dados clínicos e moleculares de câncer de pulmão



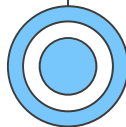


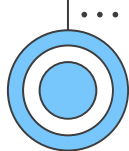
# Tipos de redes e seus dados

- MLP
  - Dados Tabulares
  - Cada observação tem sua própria linha e cada atributo tem sua própria coluna
- CNN
  - Dados mais complexos como dados de imagem
  - Dificilmente se enquadram em um conjunto de dados tabular típico
  - No entanto, os dados ainda tem comprimento fixo.

O que fazer quando nos deparamos com dados sequenciais?

...



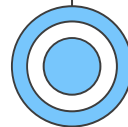


# RNNs e dados sequenciais

- Muitas tarefas de aprendizagem exigem lidar com dados sequenciais.
- Dados sequenciais estão em toda parte
  - Vídeo ( Sequência de imagens)
  - Audio ( Sequência de ondas sonoras)
  - Texto ( Sequência de caracteres/palavras)
  - Preço de ações
  - Sequências de DNA
  - Condições climáticas
- Diversos tipos de problemas
  - Classificação Binária
  - Classificação de Sentimentos
  - Legendagem de imagens ...
  - Tradução

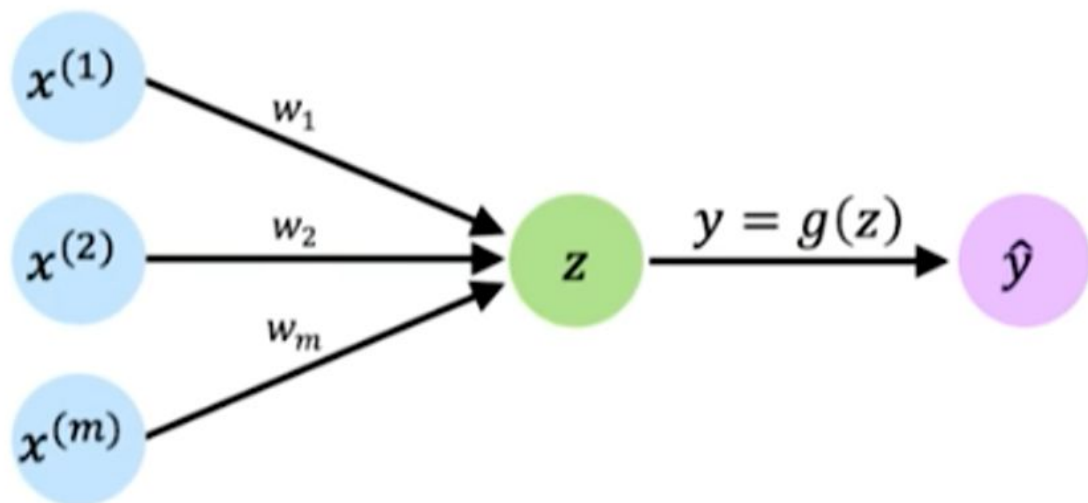


...

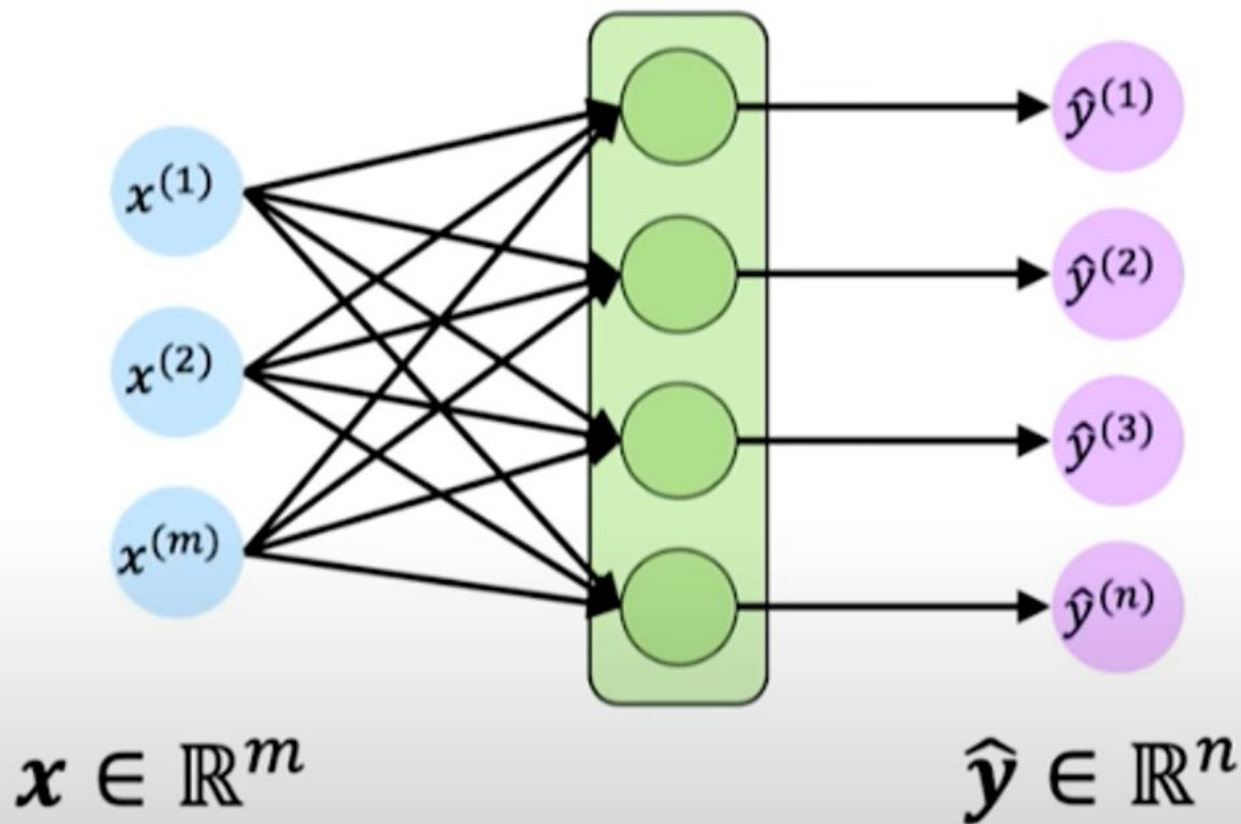


...

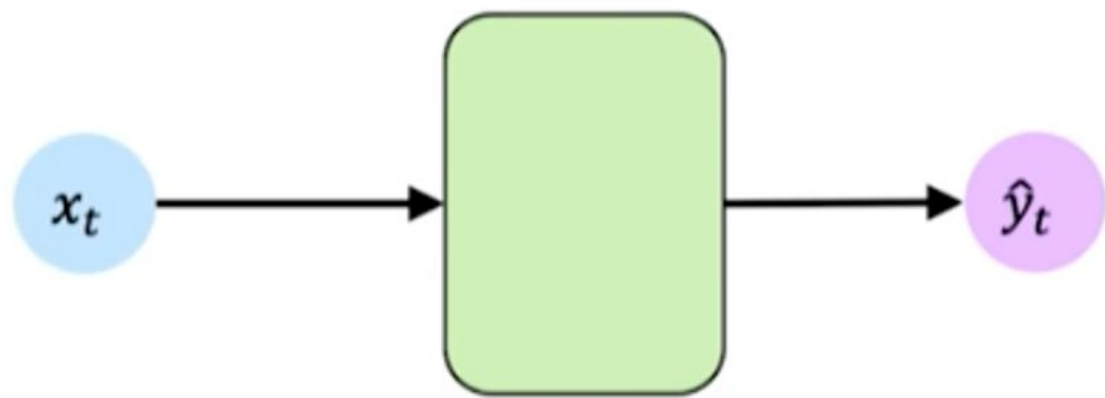
# The Perceptron Revisited



# Feed-Forward Networks Revisited



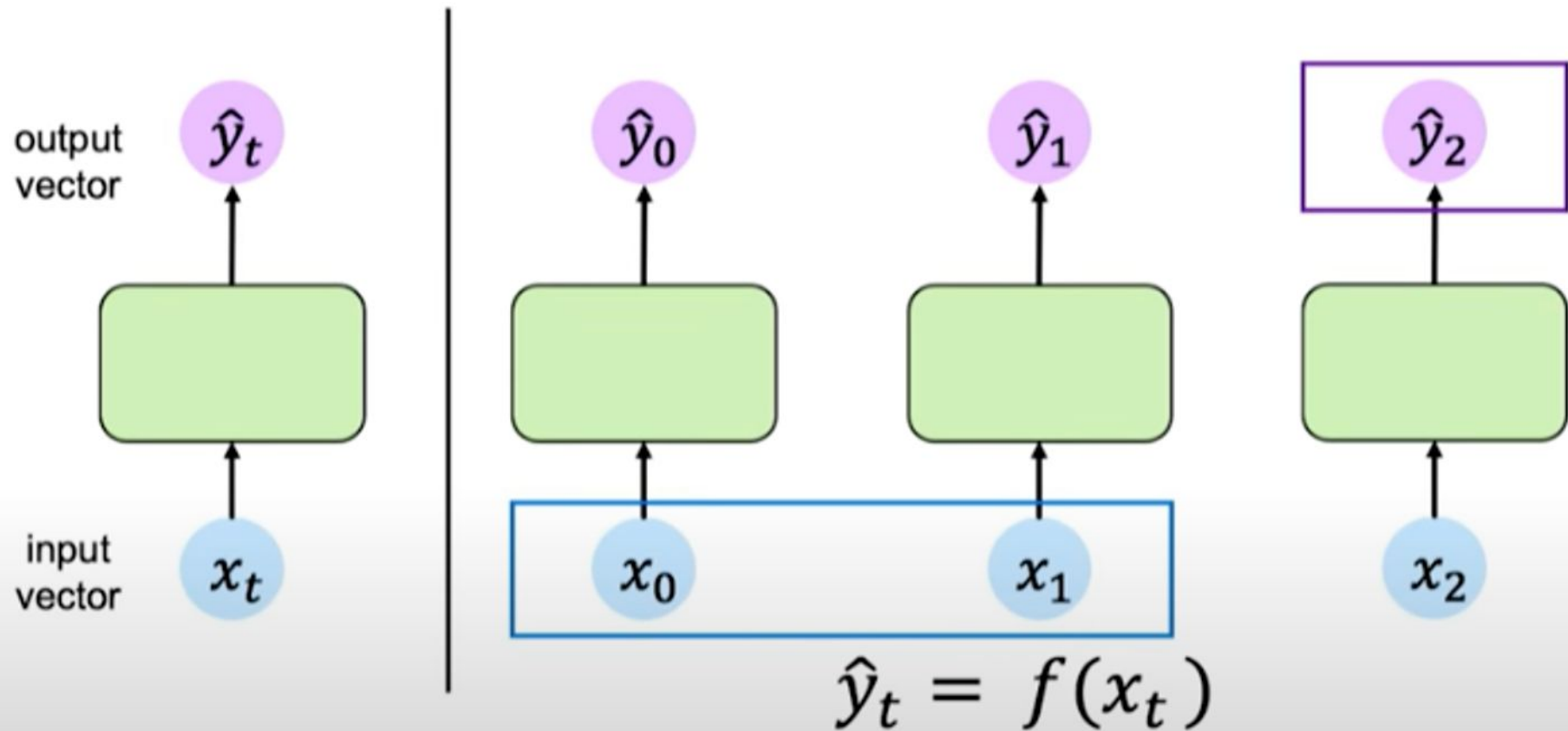
# Feed-Forward Networks Revisited



$$\mathbf{x}_t \in \mathbb{R}^m$$

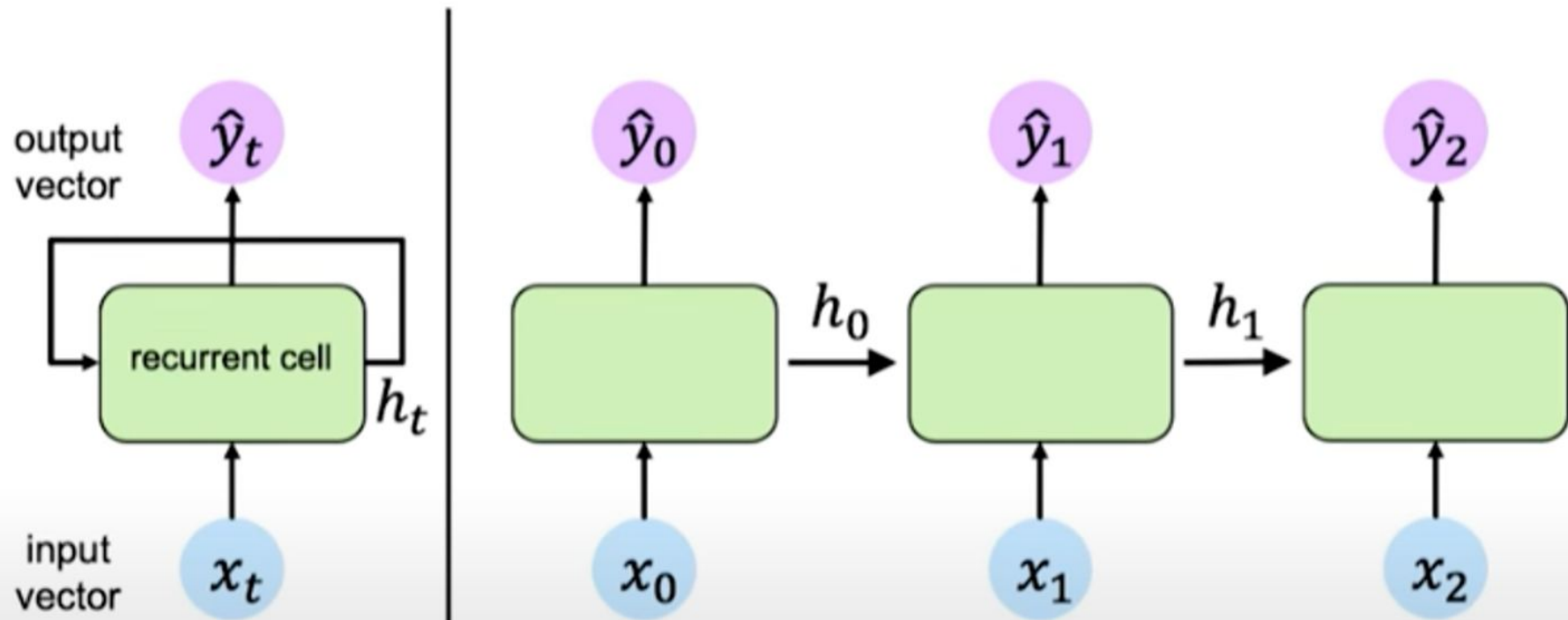
$$\hat{\mathbf{y}}_t \in \mathbb{R}^n$$

# Handling Individual Time Steps



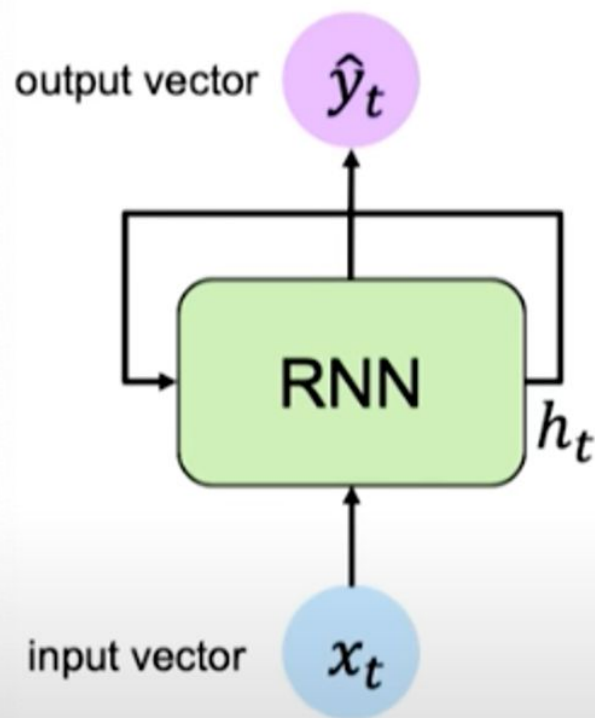


# Neurons with Recurrence



$$\hat{y}_t = f(\underbrace{x_t}_{\text{input}}, \underbrace{h_{t-1}}_{\text{past memory}})$$

# Recurrent Neural Networks (RNNs)



Apply a **recurrence relation** at every time step to process a sequence:

$$\boxed{h_t} = \boxed{f_W}(\boxed{x_t}, \boxed{h_{t-1}})$$

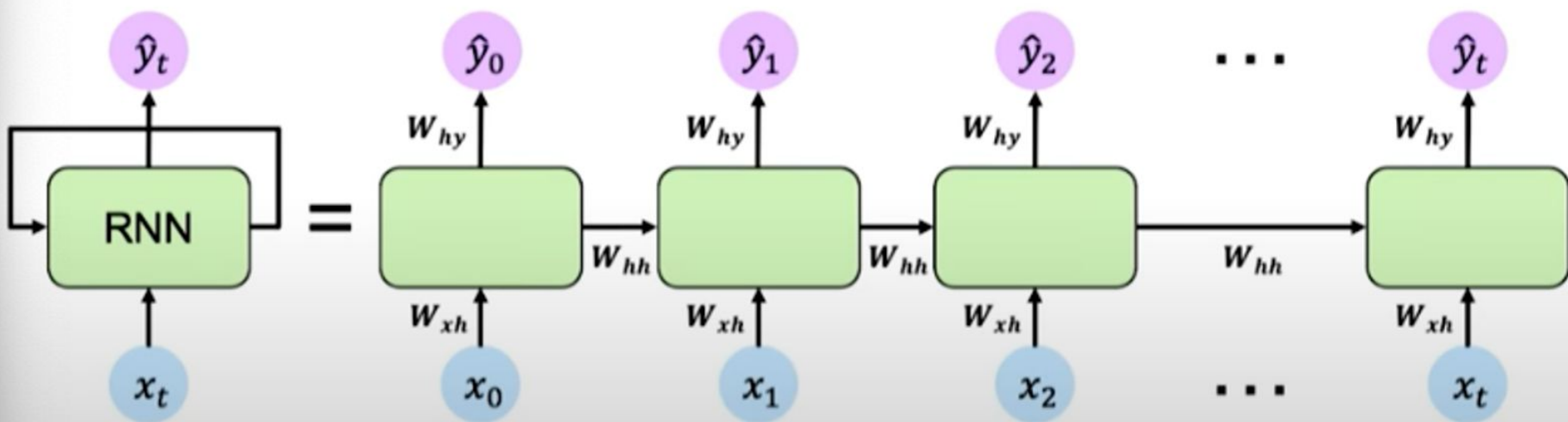
cell state      function with weights  $W$       input      old state

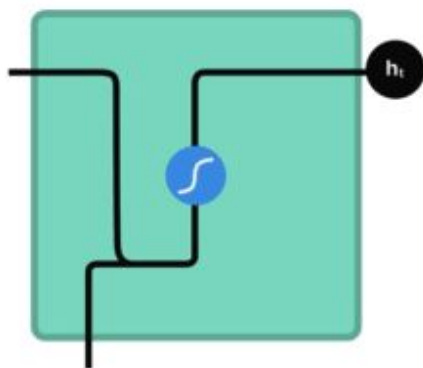
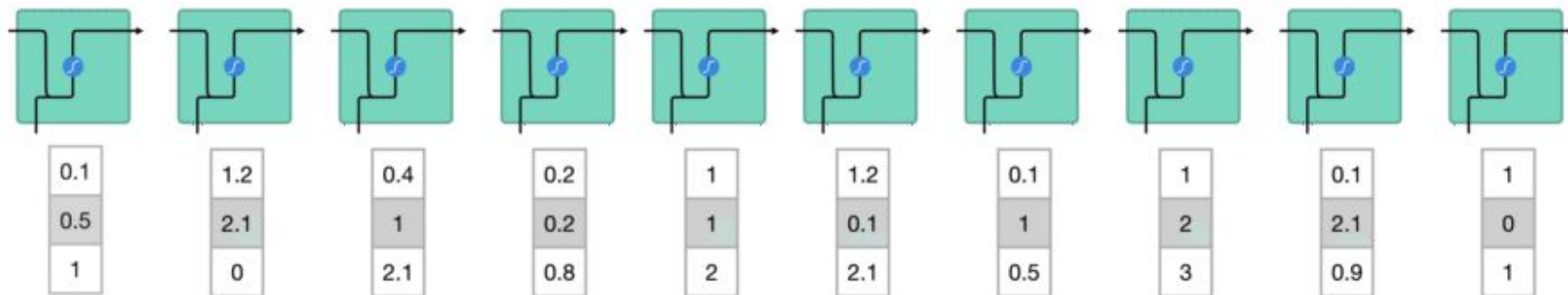
Note: the same function and set of parameters are used at every time step

RNNs have a **state**,  $h_t$ , that is updated **at each time step** as a sequence is processed

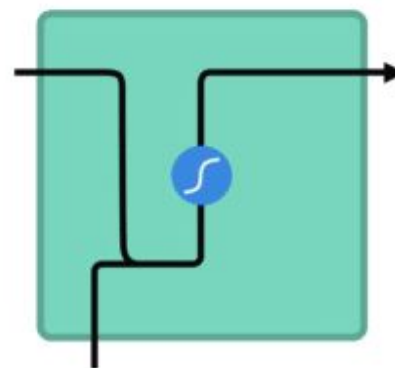
# RNNs: Computational Graph Across Time

Re-use the **same weight matrices** at every time step

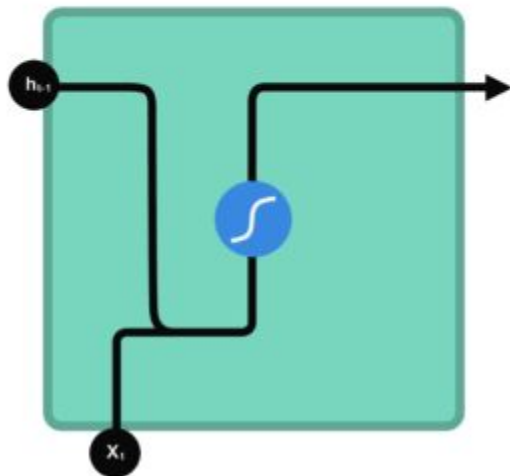




Tanh function



hidden state (memory)



Tanh function



new hidden state



previous hidden state

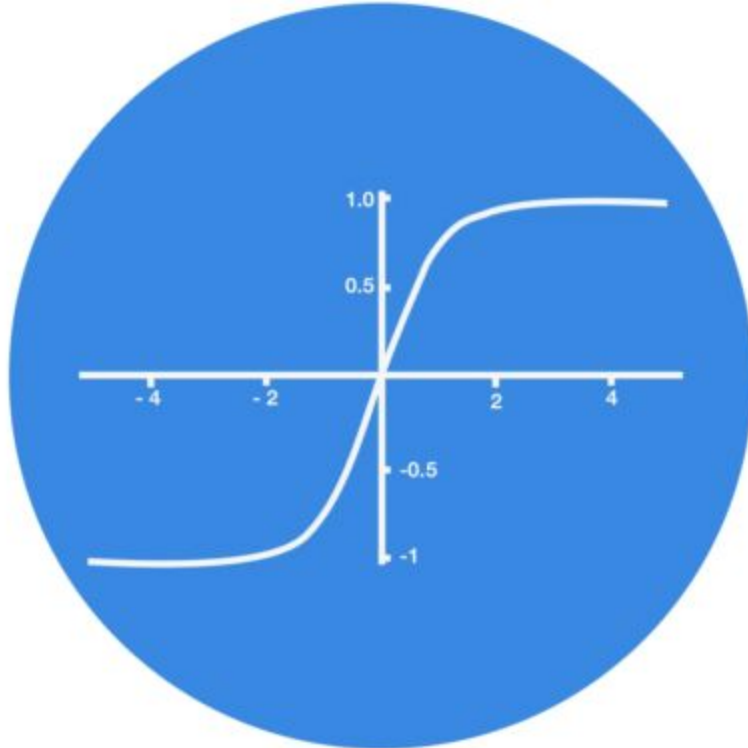


input

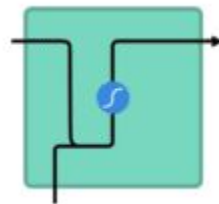
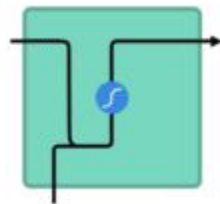
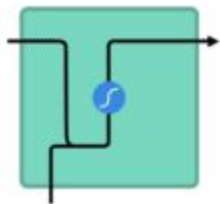
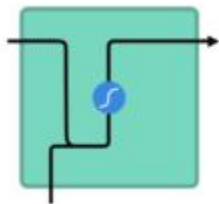


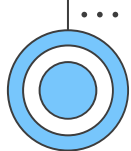
concatenation

5
0.1
-0.5



5
0.01
-0.5





# Modelos sequenciais

- Tipos de modelos sequenciais
  - One to One
  - Many to One
  - One to Many
  - Many to Many
- Critérios para modelos sequenciais
  - Lidar com sequências de comprimento variável
  - Rastrear dependências de longo prazo
  - Manter informações de ordem
  - Compartilhar parâmetros em toda a sequência

...



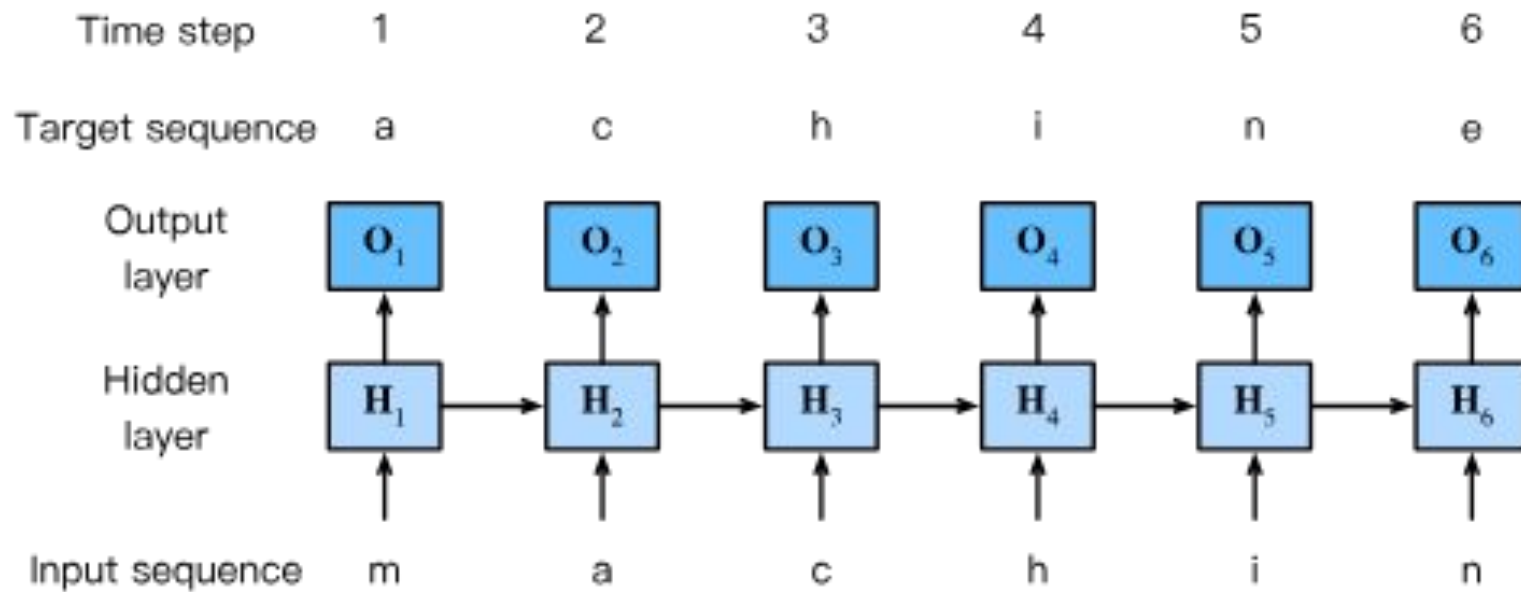
...



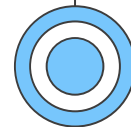
...



# Modelos de Linguagem



...





# A Sequence Modeling Problem: Predict the Next Word

"This morning I took my cat for a walk."

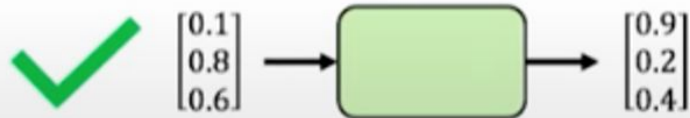
given these words

predict the  
next word

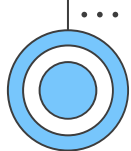
## Representing Language to a Neural Network



*Neural networks cannot interpret words*



*Neural networks require numerical inputs*



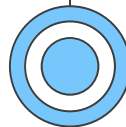
# Perplexidade

- RNNs são constantemente utilizadas para modelos de linguagem
- Perplexidade para medição da qualidade do modelo de linguagem

- "Está chovendo **lá fora**"
- "Está chovendo **bananeira**"
- "Está chovendo **piouw;kcj pwepoiut**"

$$\exp \left( -\frac{1}{n} \sum_{t=1}^n \log P(x_t | x_{t-1}, \dots, x_1) \right)$$

- Inverso da média geométrica do número de escolhas reais que temos ao decidir qual token escolher em seguida.
  - Melhor hipótese, o modelo sempre estima perfeitamente a probabilidade do token alvo como 1.
  - Pior hipótese, o modelo sempre prevê a probabilidade do token alvo como 0. Nessa situação, a perplexidade é infinita positiva.
  - Se o modelo prevê uma distribuição uniforme sobre todos os tokens disponíveis do vocabulário, a perplexidade é igual ao número de tokens únicos do vocabulário.



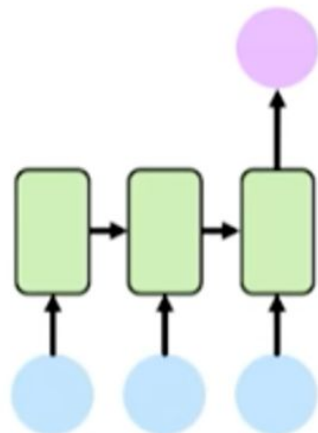
# Sequence Modeling Applications



One to One  
**Binary Classification**



"Will I pass this class?"  
Student → Pass?

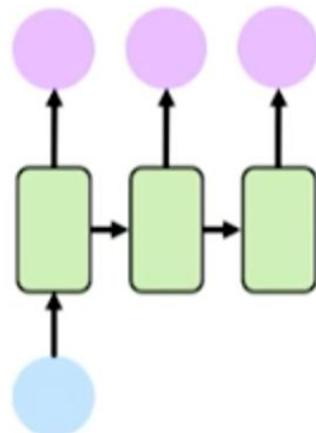


Many to One  
**Sentiment Classification**



The @MIT Introduction to #DeepLearning is definitely one of the best courses of its kind currently available online  
[introtodeeplearning.com](http://introtodeeplearning.com)

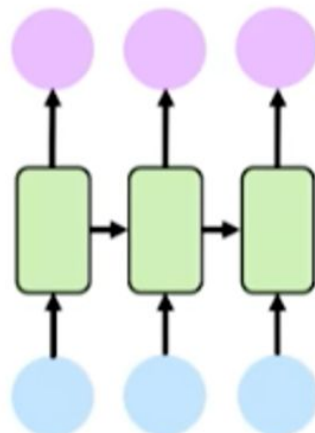
12:45 PM · 12 Feb 2018



One to Many  
**Image Captioning**



"A baseball player throws a ball."



Many to Many  
**Machine Translation**



# Aplicação

```
class RNN(d2l.Module): #@save
    """The RNN model implemented with high-level APIs."""
    def __init__(self, num_hiddens):
        super().__init__()
        self.save_hyperparameters()
        self.rnn = tf.keras.layers.SimpleRNN(
            num_hiddens, return_sequences=True, return_state=True,
            time_major=True)

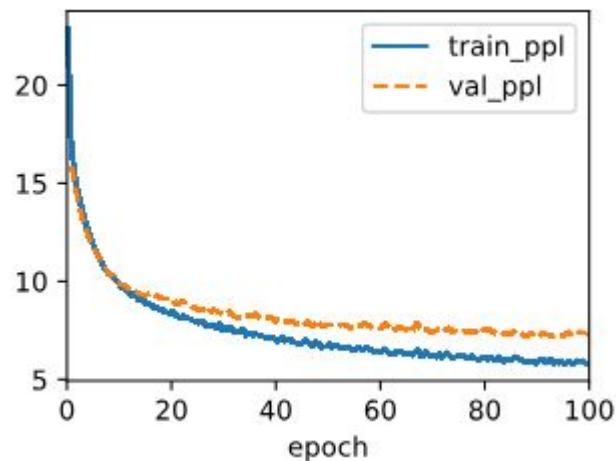
    def forward(self, inputs, H=None):
        outputs, H = self.rnn(inputs, H)
        return outputs, H

class RNNLM(d2l.RNNLMScratch): #@save
    """The RNN-based language model implemented with high-level APIs."""
    def init_params(self):
        self.linear = tf.keras.layers.Dense(self.vocab_size)

    def output_layer(self, hiddens):
        return tf.transpose(self.linear(hiddens), (1, 0, 2))
```

```
data = d2l.TimeMachine(batch_size=1024, num_steps=32)
rnn = RNN(num_hiddens=32)
model = RNNLM(rnn, vocab_size=len(data.vocab), lr=1)
model.predict('it has', 20, data.vocab)
```

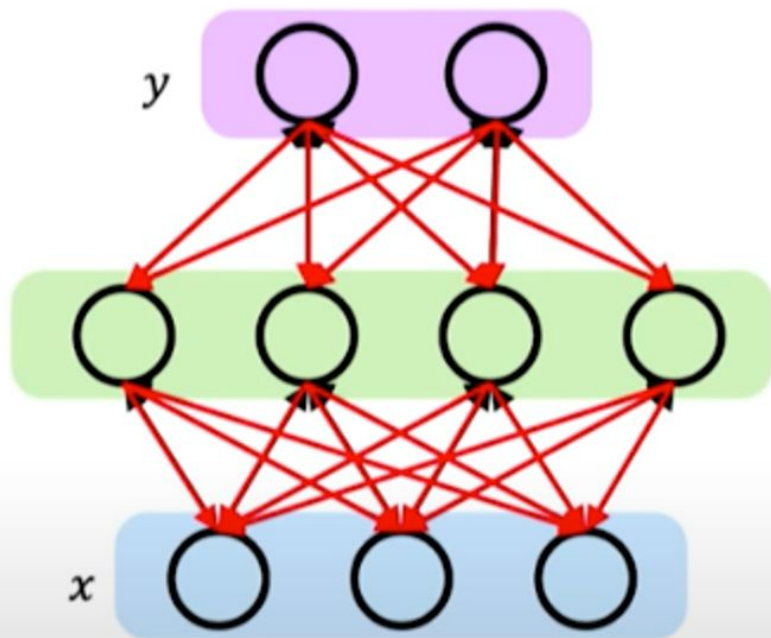
```
'it hasretsnrnrnxnrnrgczntgq'
```



```
model.predict('it has', 20, data.vocab)
```

```
'it has and the pas an and '
```

# Recall: Backpropagation in Feed Forward Models

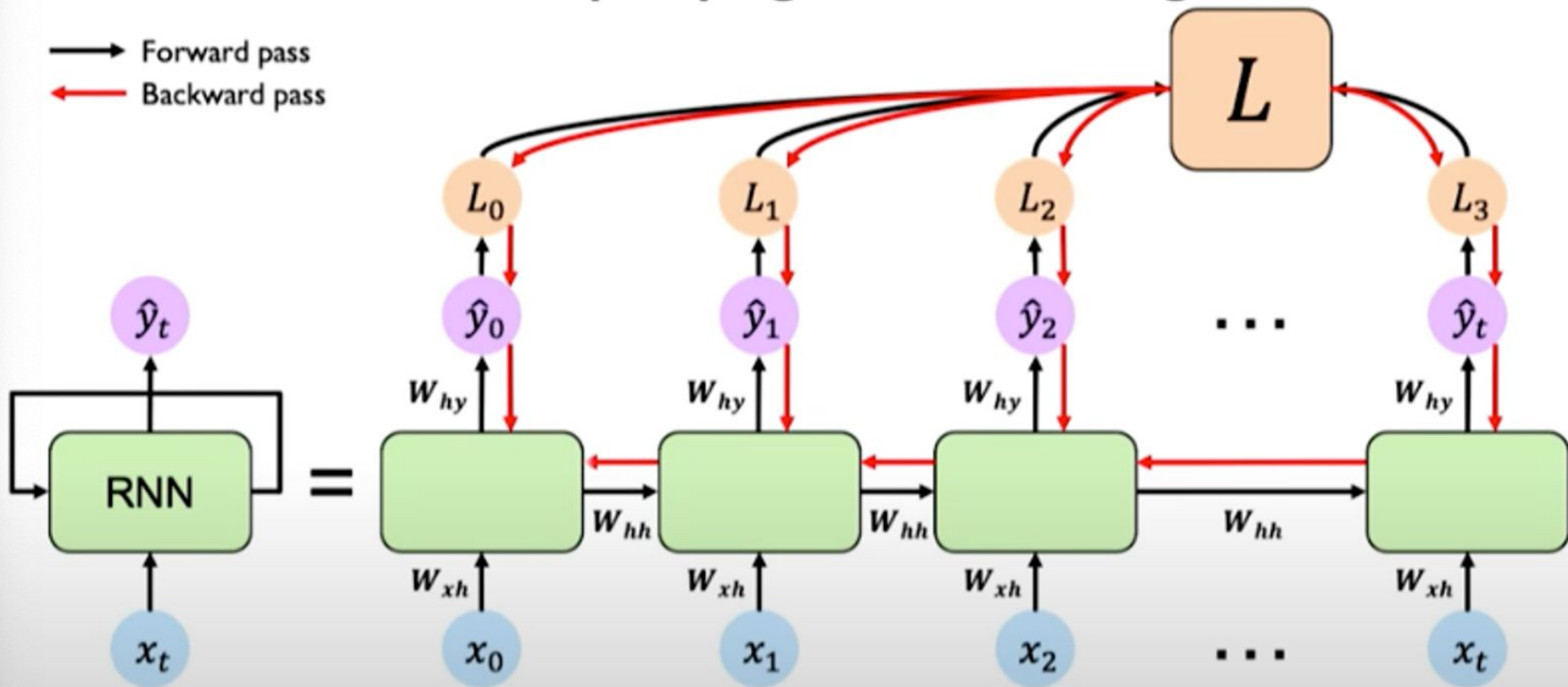


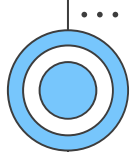
## Backpropagation algorithm:

1. Take the derivative (gradient) of the loss with respect to each parameter
2. Shift parameters in order to minimize loss

# RNNs: Backpropagation Through Time

→ Forward pass  
← Backward pass





# Backpropagation em modelos sequenciais

- Em RNNs, o avanço envolve avançar no tempo.
  - Os erros retrocedem no tempo até o início da sequência.
- Para calcular o gradiente, são necessárias muitas multiplicações de matrizes envolvendo a matriz de pesos, bem como cálculos repetidos de gradientes.
- Esta operação de multiplicação repetida é problemática pois pode levar à Explosão ou esvanecimento do gradiente.

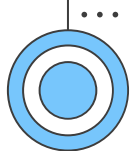
...



...



...



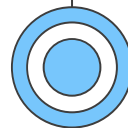
# Gradiente em modelos sequenciais

- Explosão do gradiente durante o treinamento na situação em que muitos valores são muito maiores que 1.
  - Pode ser usado gradient clipping
- Esvanecimento do gradiente onde os valores dos pesos são muito pequenos.
  - Multiplicar muitos números menores pode enviesar o modelo levando o foco nas dependências de curto prazo e ignorando as dependências de longo prazo.

...

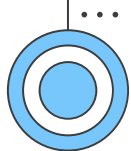


...



...

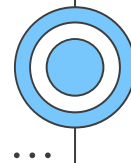


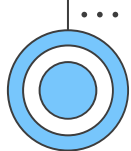


# Métodos para evitar o esvanecimento do gradiente

- Funções de Ativação
  - Evitar que o gradiente diminua drasticamente.
  - ReLU é uma boa escolha porque o valor da função de ativação aumenta para 1 quando  $x > 0$ .
- Inicialização de parâmetros
  - Podemos inicializar os pesos na matriz identidade para evitar que os pesos diminuam para zero.

...



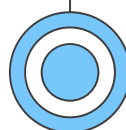


# Métodos para evitar o esvanecimento do gradiente

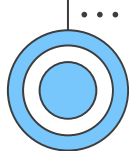
- Gated Cells
  - Uma solução mais robusta
  - Unidade recorrente mais complexa para rastrear de forma mais eficaz as dependências de longo prazo.
  - Gates ajudam a adicionar ou remover seletivamente informações dentro de cada unidade recorrente.
  - Um exemplo é as LSTM.
  - Nos LSTMs, existem muitas operações introduzidas para ajudar a controlar o fluxo de informações.



...

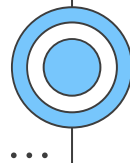
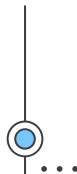


...



# Long-Short Term Memory

- Primeiros RNNs (Elman, 1990 )
- Problemas de aprendizagem (Bengio et al. , 1994 , Hochreiter et al. , 2001 )
- LSTM (Hochreiter e Schmidhuber ( 1997 )
- LSTMs se assemelham a redes neurais recorrentes padrão, mas:
  - Cada nó recorrente comum é substituído por uma célula de memória.
  - Caracterizada pela cell state a e seus vários gates.
- “Memória de longo e curto prazo”
  - Intuição: Redes neurais recorrentes simples possuem memória de longo prazo na forma de pesos.
  - O modelo LSTM introduz um tipo intermediário de armazenamento através da célula de memória.



## Customers Review 2,491

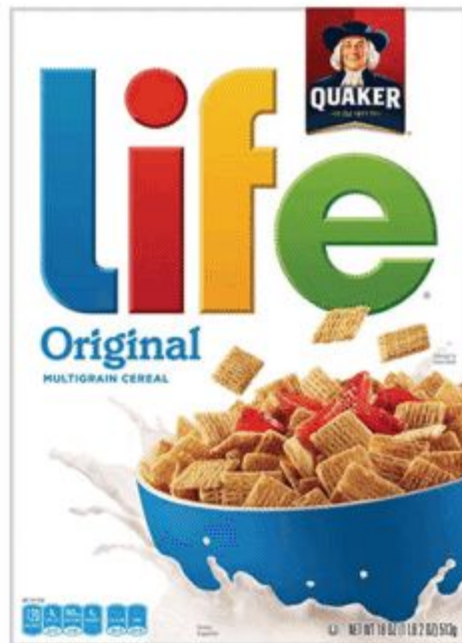


Thanos

September 2018

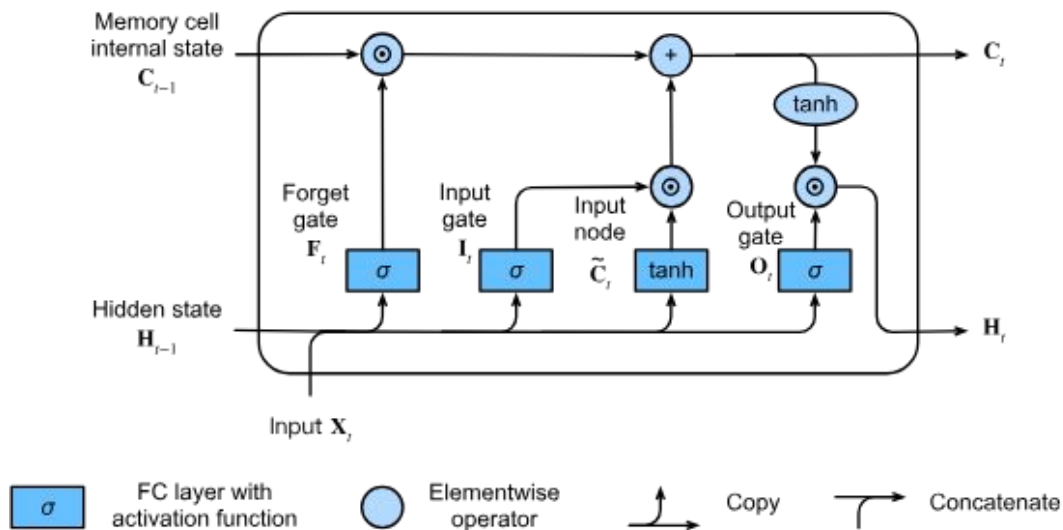
Verified Purchase

**Amazing! This box of cereal gave me a perfectly balanced breakfast, as all things should be. I only ate half of it but will definitely be buying again!**

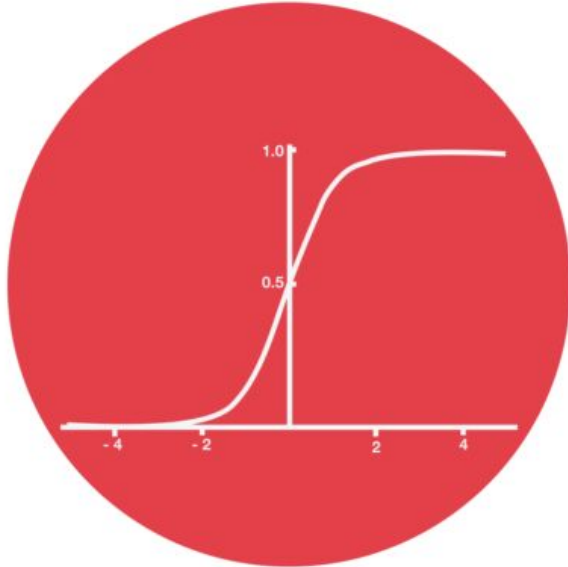


**A Box of Cereal**  
**\$3.99**

- Cell State
  - Transfere informações relativas por toda a cadeia de sequência.
  - Pode transportar informações relevantes ao longo do processamento da sequência.
  - Leva informações dos intervalos de tempo anteriores aos intervalos de tempo posteriores
  - Informações são adicionadas ou removidas do cell state por meio de gates.
- Gates são diferentes redes neurais que decidem quais informações são permitidas no cell state.

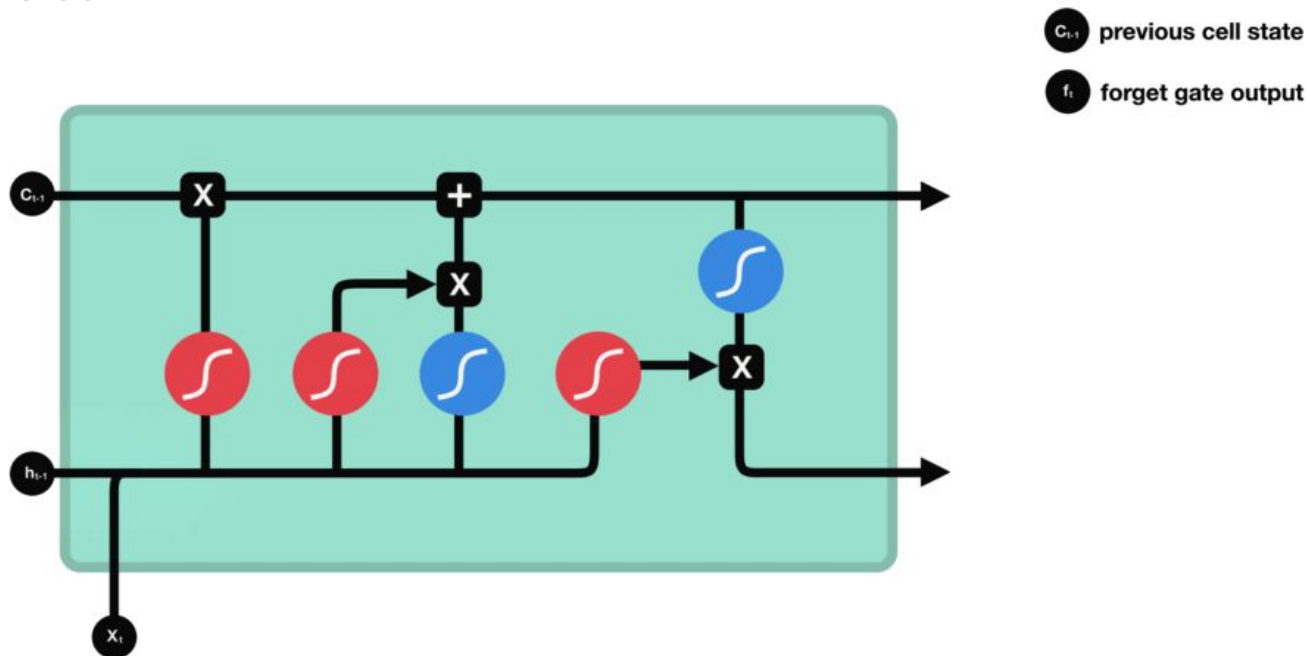


5
0.1
-0.5



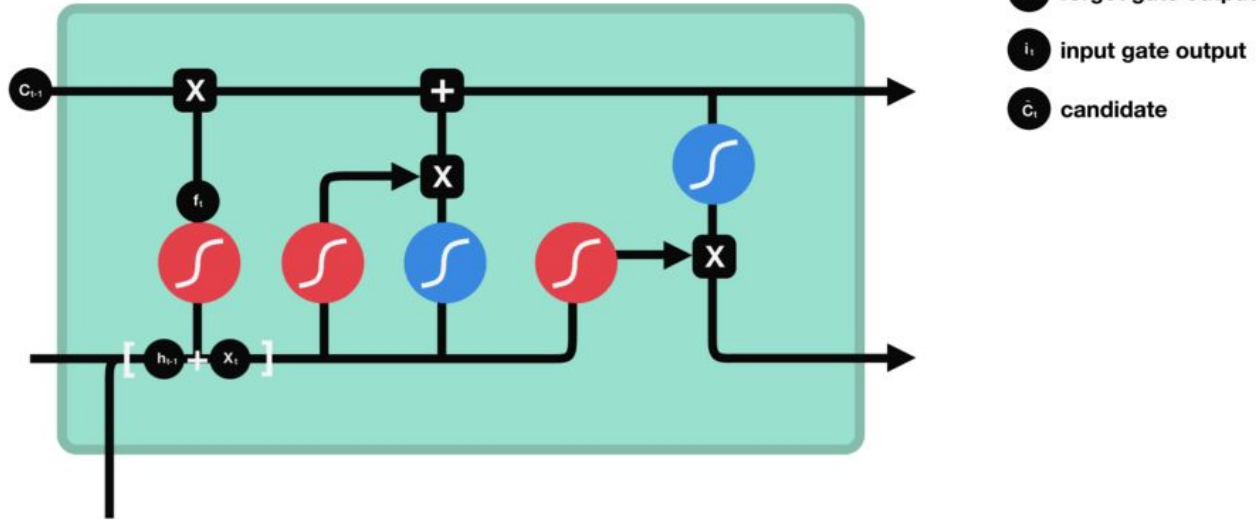
- Gates contém ativações sigmóides.
- Uma ativação sigmóide é semelhante à ativação tanh.
  - Tanh comprime valores entre -1 e 1
  - Sigmoid comprime valores entre 0 e 1
  - Útil para atualizar ou esquecer dados
    - Número multiplicado por 0 é 0, fazendo com que os valores desapareçam ou sejam “esquecidos”.
    - Número multiplicado por 1 tem o mesmo valor, portanto esse valor permanece o mesmo ou é “mantido”.

# Forget Gate



- Este gate decide quais informações devem ser descartadas ou guardadas.
- As informações do estado oculto anterior e as informações da entrada atual são passadas pela função sigmóide.
- Os valores ficam entre 0 e 1. Próximo a 0 esquece e próximo a 1 mantém

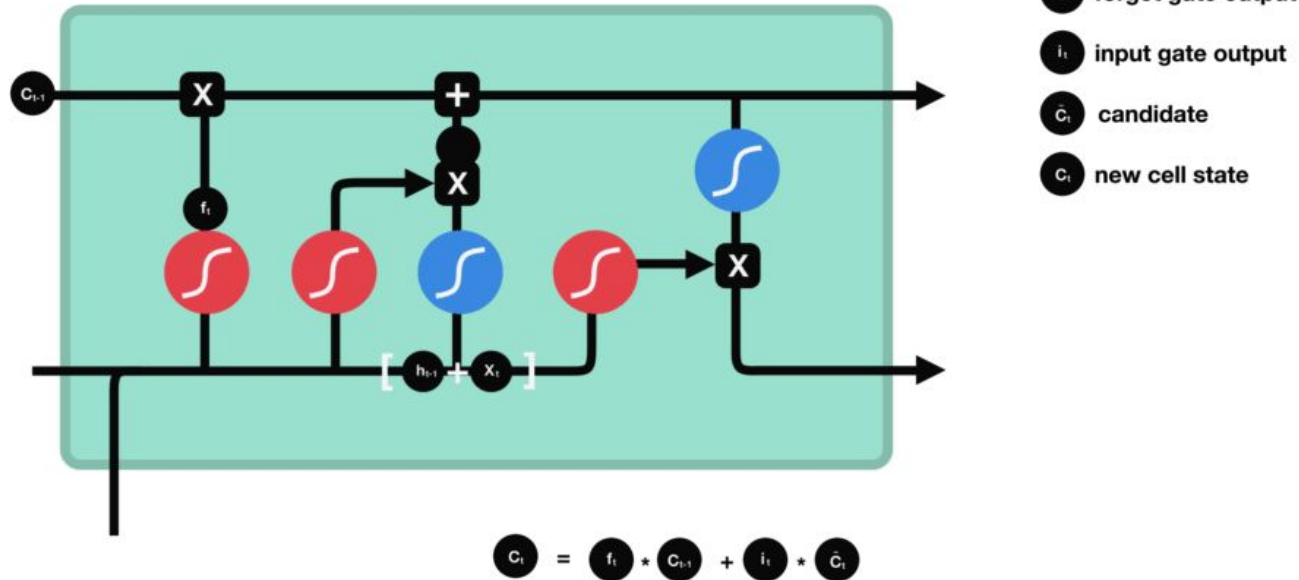
# Input Gate



- Para atualizar o cell state, temos o input gate
- Estado oculto anterior e a entrada atual para uma função sigmóide.
- Valores serão atualizados transformando os valores entre 0 e 1. 0 significa não importante e 1 significa importante.
- Estado oculto e a entrada atual para a função tanh para comprimir valores entre -1 e 1 para ajudar a regular a rede. Então você multiplica a saída tanh pela saída sigmóide.
- A saída sigmóide decidirá quais informações são importantes para manter na saída tanh.

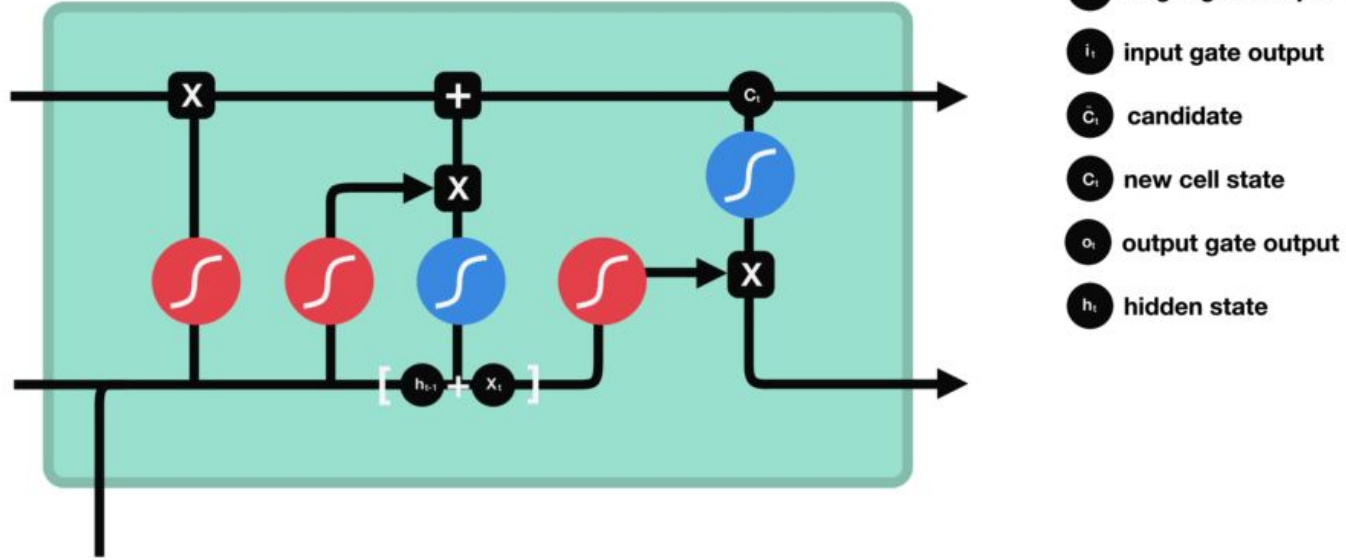


# Cell State



- Calcular o cell state.
- Cell state é multiplicado pontualmente pelo vetor de esquecimento.
  - Possibilidade de diminuir valores no estado da célula se for multiplicado por valores próximos a 0.
- Pegamos a saída da porta de entrada e fazemos uma adição pontual que atualiza o estado da célula para novos valores que a rede neural considera relevantes. Isso nos dá nosso novo cell state

# Output Gate



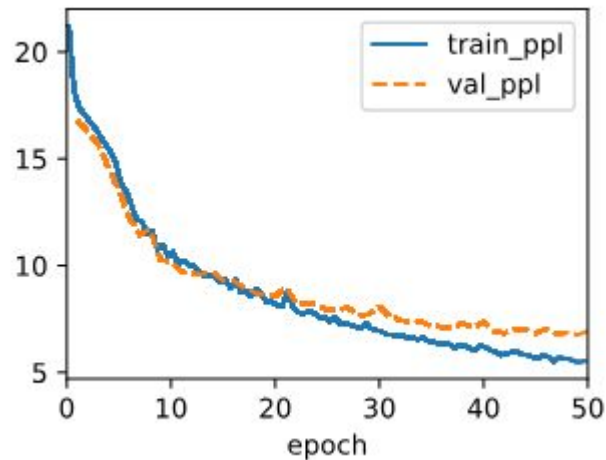
- O output gate decide qual deve ser o próximo estado oculto.
- O estado oculto contém informações sobre entradas anteriores e também é usado para previsões.
- O estado oculto anterior e a entrada atual é passado para uma função sigmóide.
- O estado da célula recém-modificado passa para a função tanh.
- Multiplicamos a saída tanh pela saída sigmóide para decidir quais informações o estado oculto deve transportar.
- A saída é o estado oculto.
- O novo estado da célula e o novo oculto são então transferidos para a próxima etapa de tempo.

# Aplicações

```
class LSTM(d2l.RNN):
    def __init__(self, num_hiddens):
        d2l.Module.__init__(self)
        self.save_hyperparameters()
        self.rnn = tf.keras.layers.LSTM(
            num_hiddens, return_sequences=True,
            return_state=True, time_major=True)

    def forward(self, inputs, H_C=None):
        outputs, *H_C = self.rnn(inputs, H_C)
        return outputs, H_C

lstm = LSTM(num_hiddens=32)
with d2l.try_gpu():
    model = d2l.RNNLM(lstm, vocab_size=len(data.vocab), lr=4)
trainer.fit(model, data)
```

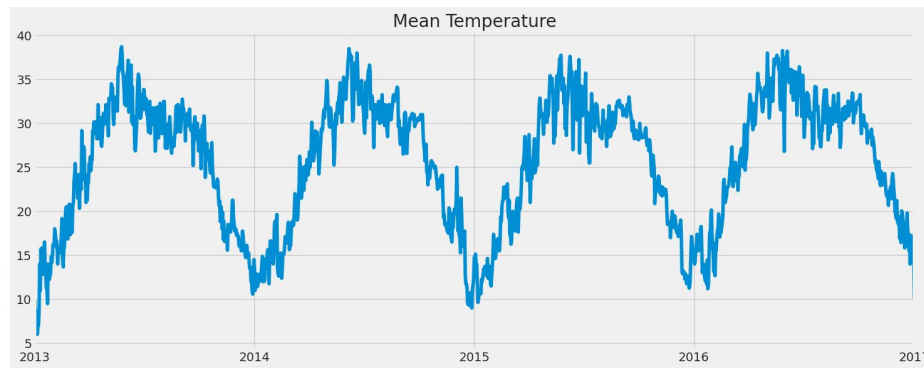


```
model.predict('it has', 20, data.vocab)
```

```
'it has a dimension a dimen'
```

# Aplicações – Previsão de temperatura média

	date	meantemp	humidity	wind_speed	meanpressure
0	2013-01-01	10.000000	84.500000	0.000000	1015.666667
1	2013-01-02	7.400000	92.000000	2.980000	1017.800000
2	2013-01-03	7.166667	87.000000	4.633333	1018.666667
3	2013-01-04	8.666667	71.333333	1.233333	1017.166667
4	2013-01-05	6.000000	86.833333	3.700000	1016.500000



```
# Creating a Training set with 60 time-steps
```

```
x_train = []
```

```
y_train = []
```

```
time_steps = 60
```

```
n_cols = 1
```

```
for i in range(time_steps, len(scaled_data)):
    x_train.append(scaled_data[i-time_steps:i, :n_cols])
    y_train.append(scaled_data[i, :n_cols])
    if i<=time_steps:
        print('X_train: ', x_train)
        print('y_train: ', y_train)
```

```
# Reshaping the input to (n_samples, time_steps, n_feature)
```

```
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], n_cols))
```

```
x_train.shape , y_train.shape
```

```
((1402, 60, 1), (1402, 1))
```

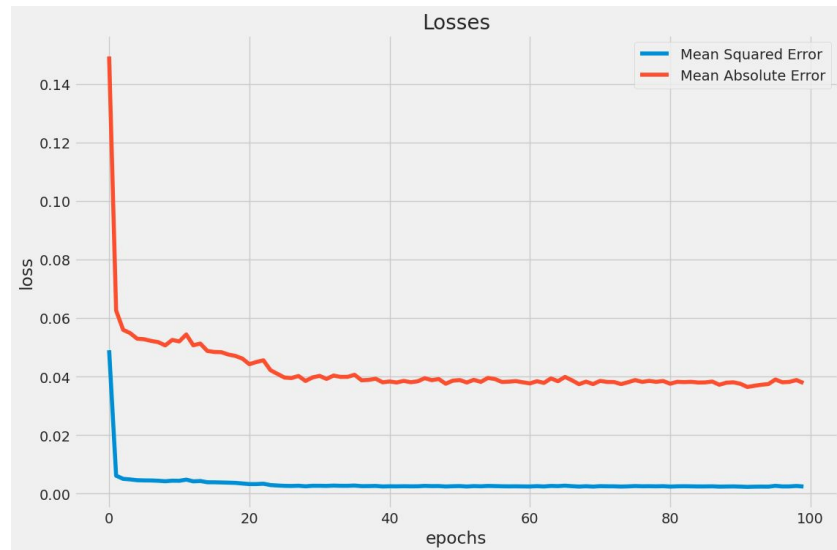
# Aplicações – Previsão de temperatura média

```
model = Sequential([
    LSTM(50, return_sequences= True, input_shape= (x_train.shape[1], n_cols)),
    LSTM(64, return_sequences= False),
    Dense(32),
    Dense(16),
    Dense(n_cols)
])

model.compile(optimizer= 'adam', loss= 'mse' , metrics= "mean_absolute_error")
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
=====		
lstm (LSTM)	(None, 60, 50)	10400
lstm_1 (LSTM)	(None, 64)	29440
dense (Dense)	(None, 32)	2080
dense_1 (Dense)	(None, 16)	528
dense_2 (Dense)	(None, 1)	17



# Aplicações – Previsão de temperatura média

```
# Creating a testing set with 60 time-steps and 1 output
time_steps = 60
test_data = scaled_data[train_size - time_steps:, :]

x_test = []
y_test = []
n_cols = 1

for i in range(time_steps, len(test_data)):
    x_test.append(test_data[i-time_steps:i, 0:n_cols])
    y_test.append(test_data[i, 0:n_cols])
x_test, y_test = np.array(x_test), np.array(y_test)
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], n_cols))
```

```
# Get Prediction
predictions = model.predict(x_test)
```

```
12/12 [=====] - 1s 18ms/step
```

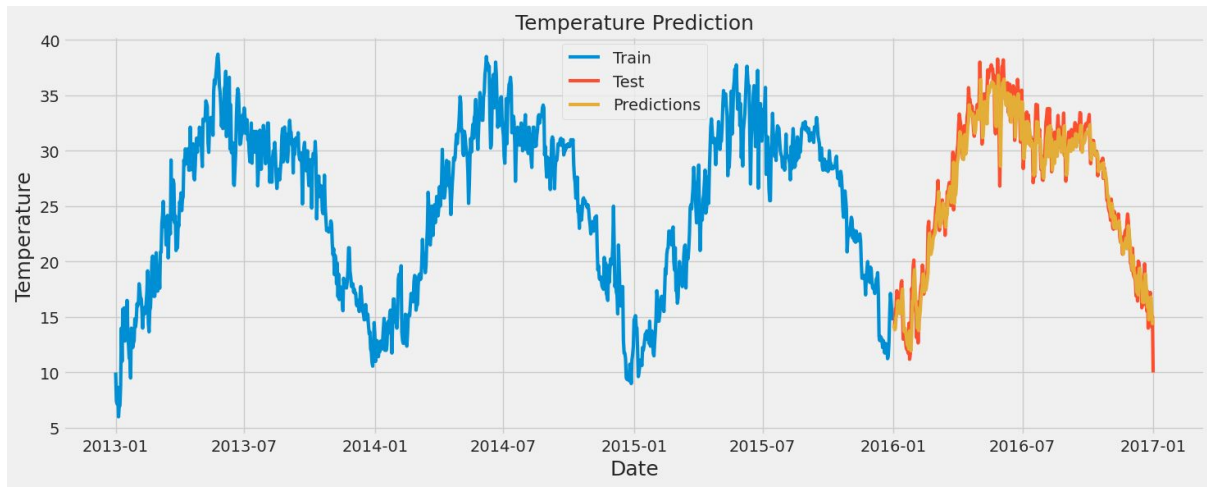
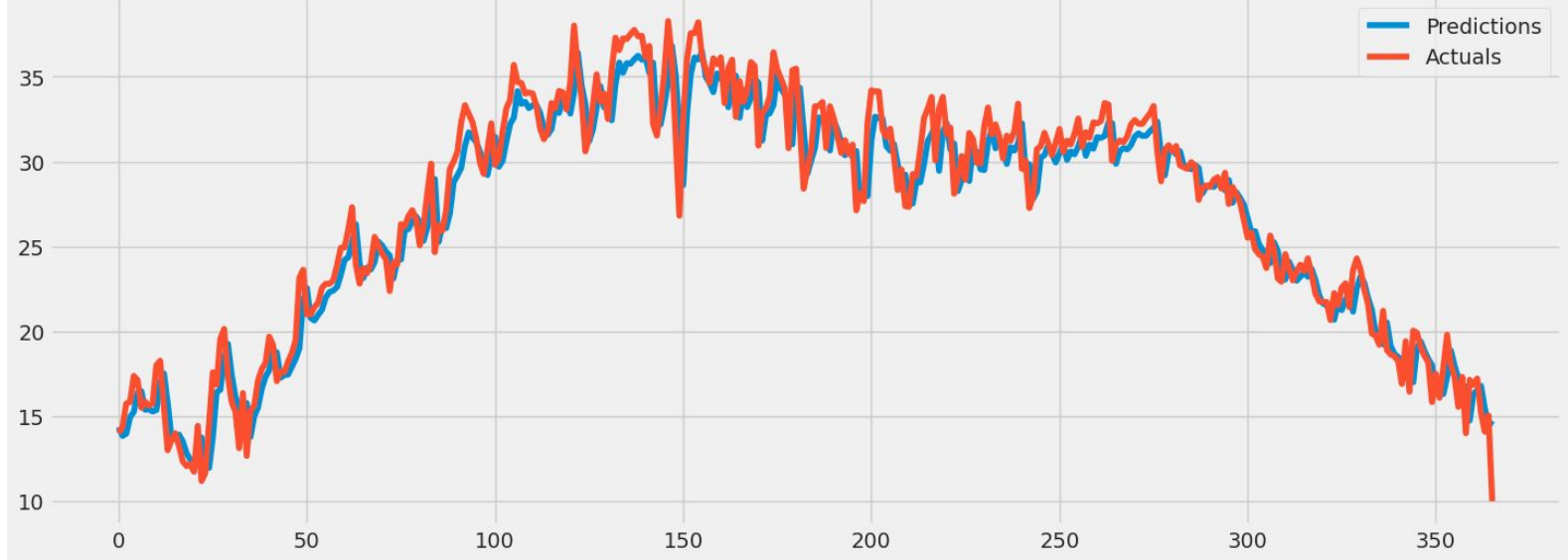
```
predictions.shape
```

```
(366, 1)
```

	Predictions	Actuals
<b>0</b>	14.367847	14.000000
<b>1</b>	13.843797	14.375000
<b>2</b>	13.974187	15.750000
<b>3</b>	14.946880	15.833333
<b>4</b>	15.261067	17.375000
...	...	...
<b>361</b>	16.526890	17.217391
<b>362</b>	16.794012	15.238095
<b>363</b>	15.496777	14.095238
<b>364</b>	14.405824	15.052632
<b>365</b>	14.742517	10.000000

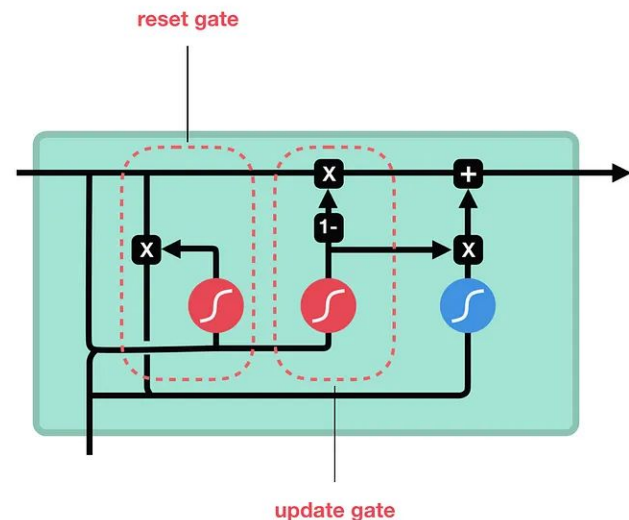
366 rows x 2 columns





# GRU – Gated Recurrent Units

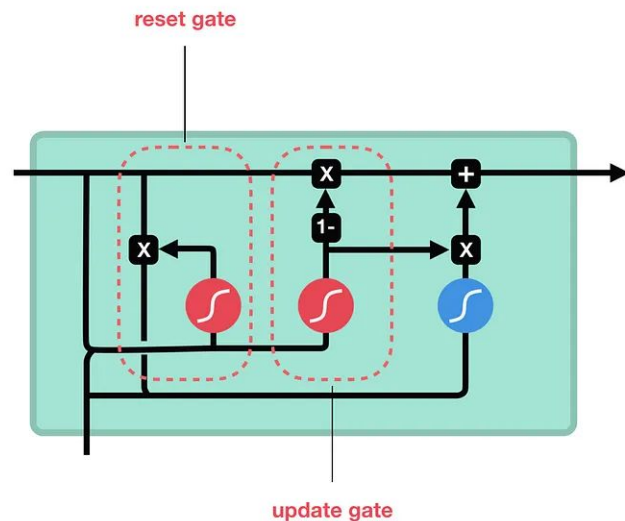
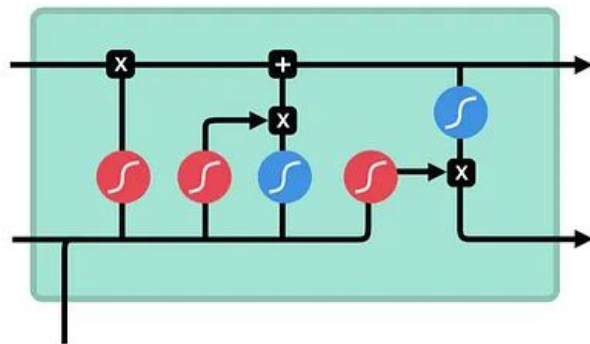
- Geração mais recente de redes neurais recorrentes e é semelhante a LSTM.
- Não usa cell state, usa o estado oculto para transferir informações. Possui apenas dois gates:
  - Reset Gate
    - Usada para decidir quanta informação passada deve ser esquecida.
  - Update Gate
    - Atua de forma semelhante à Forget e Input gate da LSTM.
    - Decide quais informações descartar e quais novas informações adicionar.
- GRU tem menos operações de tensor;
- Um pouco mais rápidos de treinar do que os LSTM.
- LSTM x GRU: tentar ambos para determinar qual funciona melhor para seu caso de uso.





# Resumo

- Redes Neurais Recorrentes são usadas para dados sequenciais
- Diversas aplicações
- RNNs modernas
  - Possuem mais seletividade de acordo com a relevância da informação
  - LSTM: Forget gate, Input gate, State cell, Output gate
  - GRU: Reset Gate, Update gate



# Referências

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

[https://d2l.ai/chapter\\_recurrent-neural-networks/language-model.html](https://d2l.ai/chapter_recurrent-neural-networks/language-model.html)

[https://d2l.ai/chapter\\_recurrent-modern/lstm.html](https://d2l.ai/chapter_recurrent-modern/lstm.html)

<https://dair-ai.notion.site/Lecture-2-RNNs-and-Transformers-71fb3ba2a24f4b6c8cc77281fc19cfab>

<https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>

<https://www.kaggle.com/code/hadeerismail/daily-climate-time-series-analysis-lstm/notebook>

<https://www.kaggle.com/datasets/sumanthvrao/daily-climate-time-series-data/data>