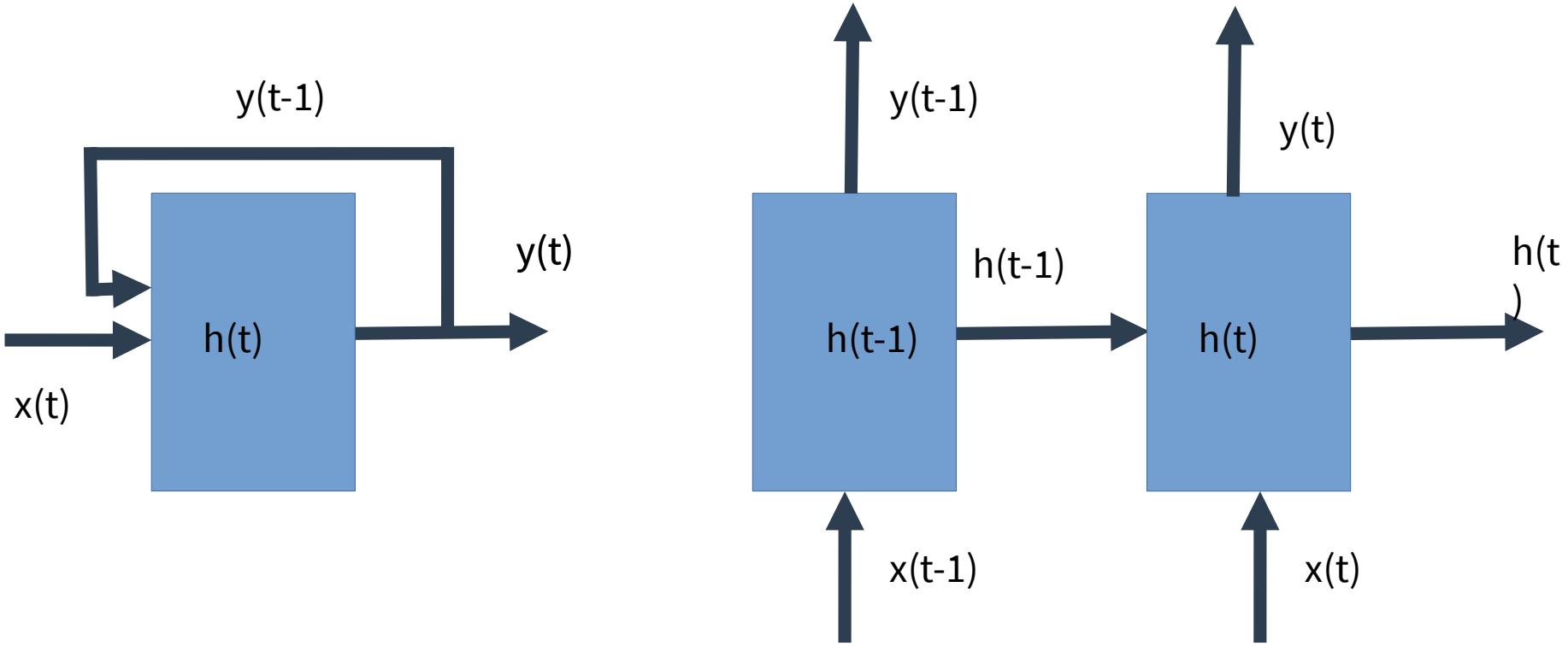
Рекуррентные архитектуры глубоких сетей

2020

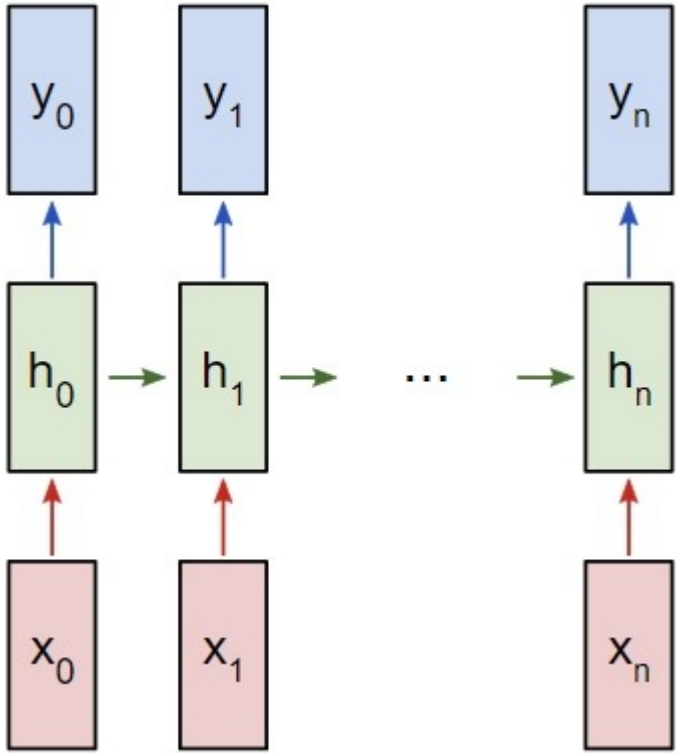
Рекурентные сети RNN



https://python-scripts.com/recurrent-neural-network

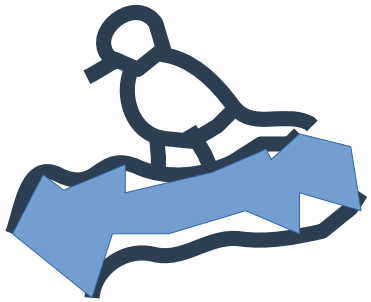
https://habr.com/ru/company/wunderfund/blog/331310/

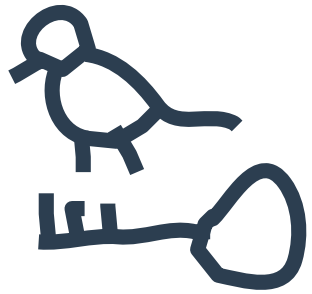
Вычисление RNN



Зачем?

Птица пьет из ключа

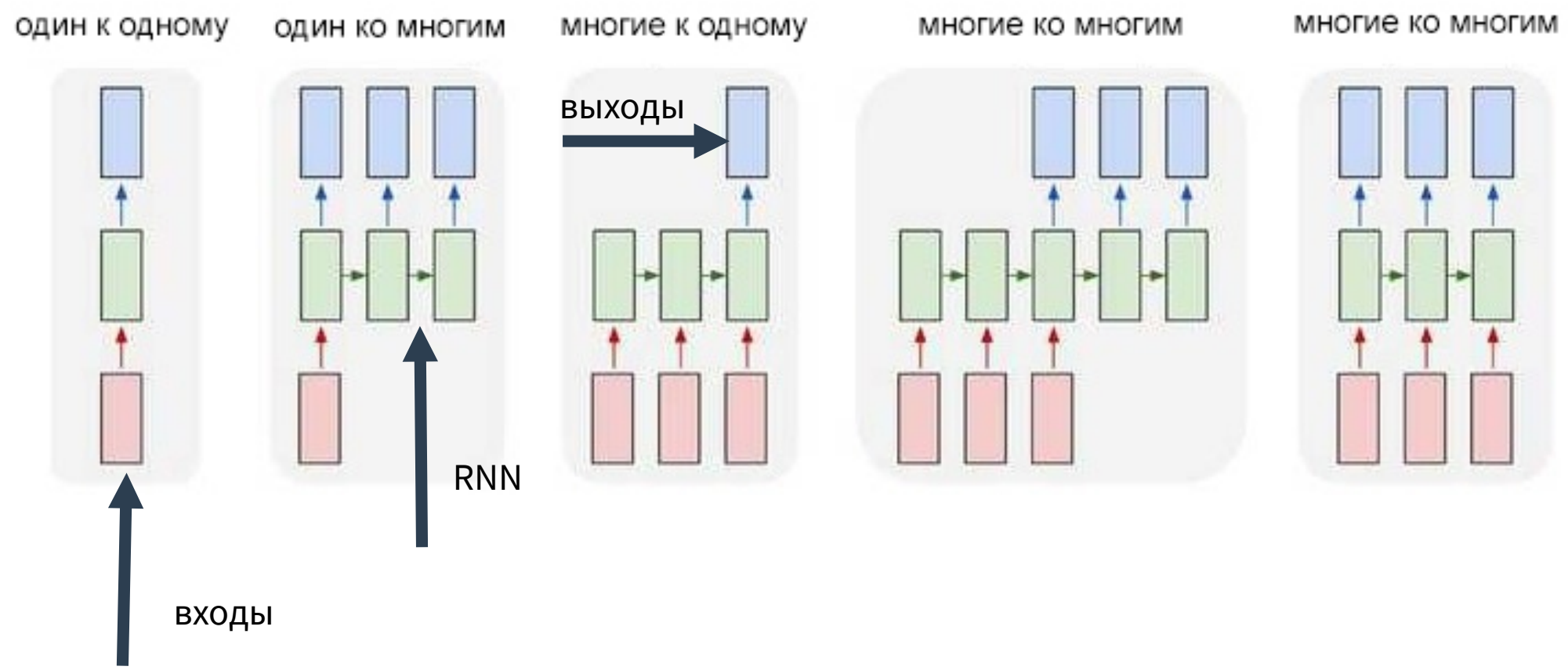




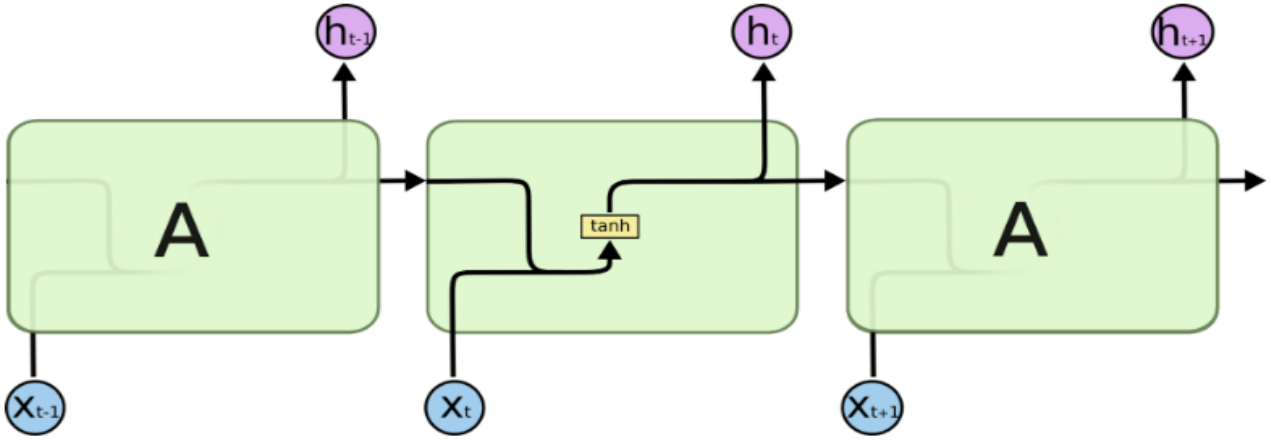
$$h_t = \tanh(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$$

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

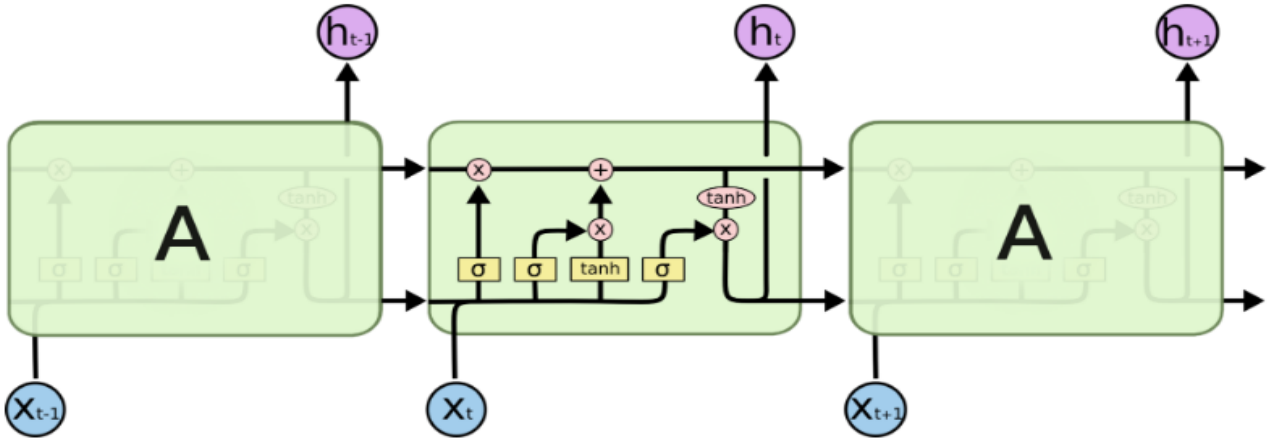
Варианты рекурентных сетей



RNN vs LSTM

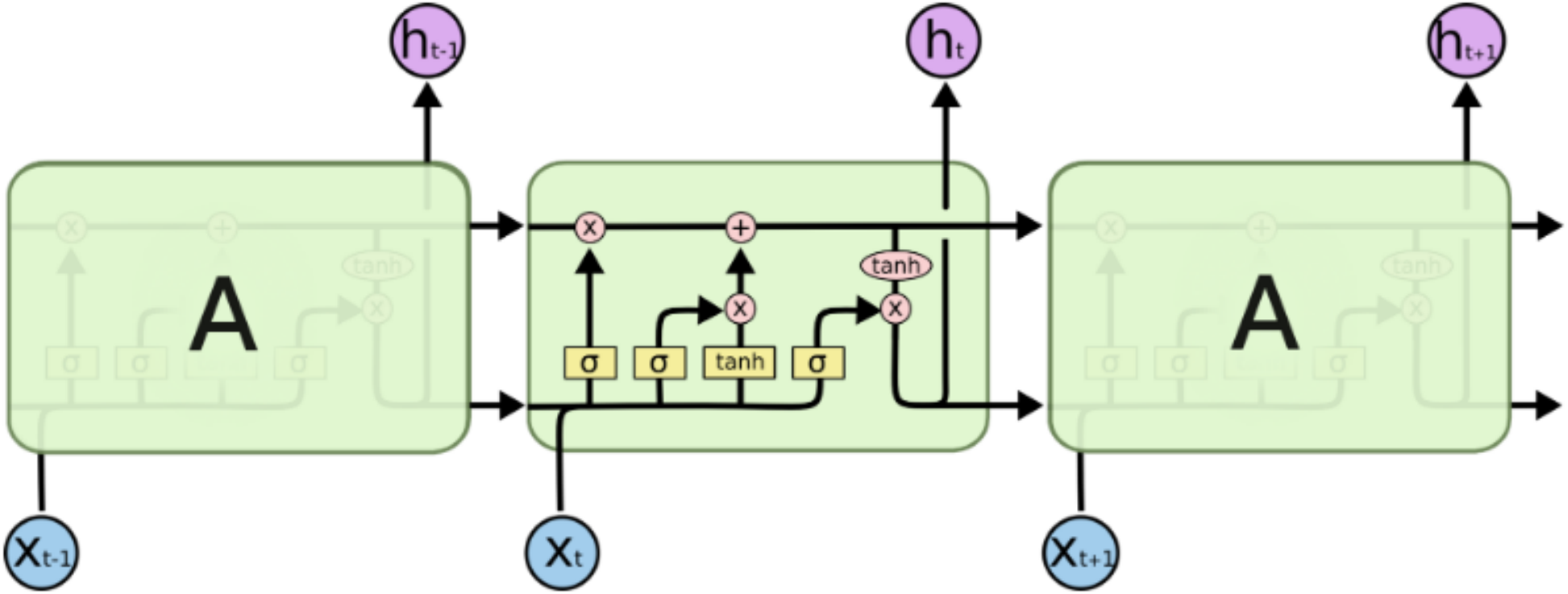


Повторяющийся модуль в стандартной RNN состоит из одного слоя.



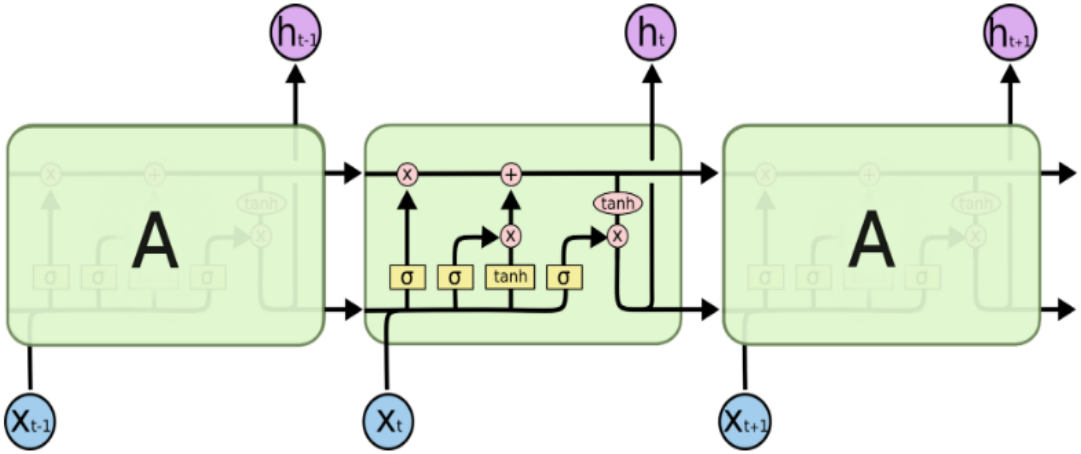
Повторяющийся модель в LSTM сети состоит из четырех взаимодействующих слоев.

RNN vs LSTM



https://nicodjimenez.github.io/2014/08/08/lstm.html

LSTM + BPTT



$$l(t) = f(h(t), y(t)) = \|h(t) - y(t)\|^2$$

$$L = \sum_{t=1}^{T} l(t)$$

$$L(t) = \begin{cases} l(t) + L(t+1) & \text{if } t < T \\ l(t) & \text{if } t = T \end{cases}$$

$$\frac{dL(t)}{dh(t)} = \frac{dl(t)}{dh(t)} + \frac{dL(t+1)}{dh(t)}$$

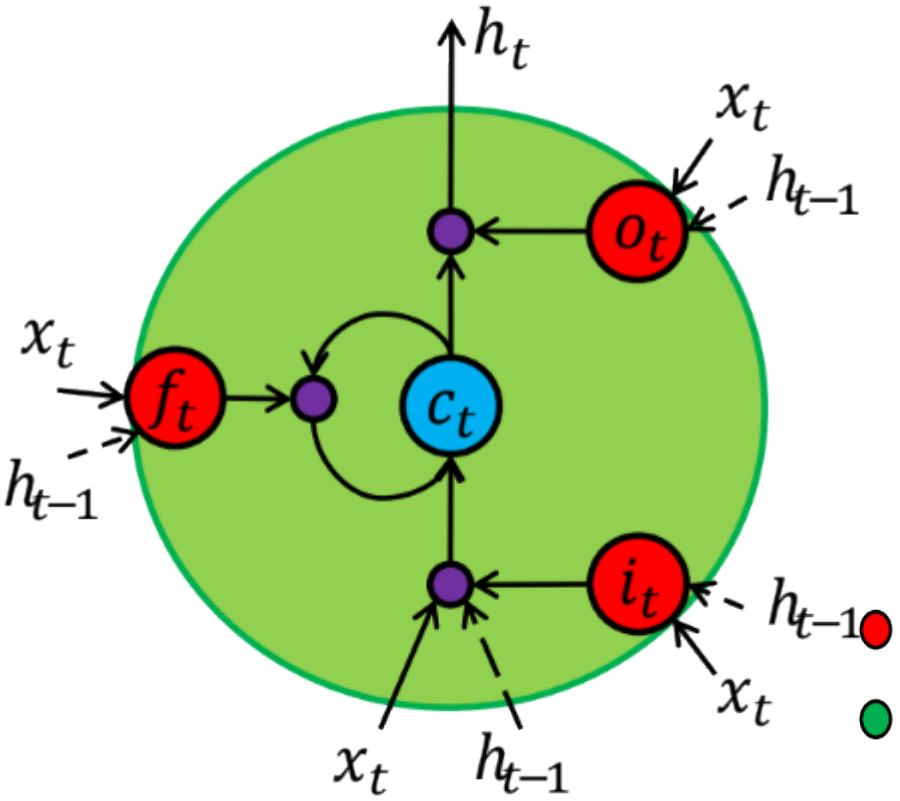
$$\frac{dL(T)}{dh(T)} = \frac{dl(T)}{dh(T)}$$

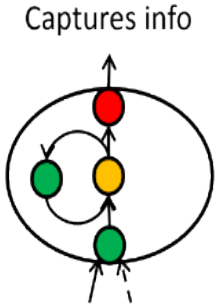
$$g(t) = \phi(W_{gx}x(t) + W_{gh}h(t-1) + b_g)$$
$$i(t) = \sigma(W_{ix}x(t) + W_{ih}h(t-1) + b_i)$$
$$f(t) = \sigma(W_{fx}x(t) + W_{fh}h(t-1) + b_f)$$
$$o(t) = \sigma(W_{ox}x(t) + W_{oh}h(t-1) + b_o)$$
$$s(t) = g(t) * i(t) + s(t-1) * f(t)$$
$$h(t) = s(t) * o(t)$$

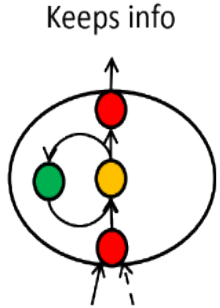
$$\frac{dL}{dw} = \sum_{t=1}^{T}\sum_{i=1}^{M} \frac{dL(t)}{dh_i(t)} \frac{dh_i(t)}{dw}$$

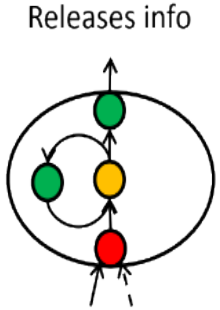
https://nicodjimenez.github.io/2014/08/08/lstm.html

RNN vs LSTM



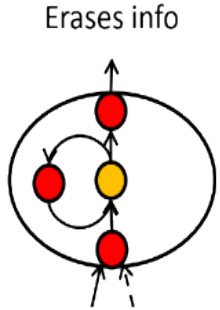


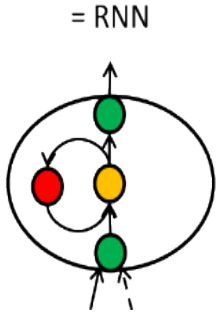




- gate is close

- gate is open





Keras

SimpleRNN

LSTM

GRU RNN Gated Recurrent Unit (GRU)

Keras

```
LSTM(units,  activation="tanh",
recurrent_activation="sigmoid",
   use_bias=True,   kernel_initializer="glorot_uniform",
   recurrent_initializer="orthogonal",
bias_initializer="zeros",
   unit_forget_bias=True,
   Dropout=0.0,   recurrent_dropout=0.0,
   return_sequences=False,   return_state=False)
```

https://keras.io/api/layers/recurrent_layers/lstm/

Keras

CallBacks

```
my_callbacks = [
    tf.keras.callbacks.EarlyStopping(patience=2),
    tf.keras.callbacks.ModelCheckpoint(filepath='model.
{epoch:02d}-{val_loss:.2f}.h5'),
    tf.keras.callbacks.TensorBoard(log_dir='./logs'),
]
model.fit(dataset, epochs=10, callbacks=my_callbacks)
```

https://keras.io/api/callbacks/

Keras

Embeding

```
tf.keras.layers.Embedding(
    input_dim,
    output_dim
)
```