Applied Artificial Intelligence

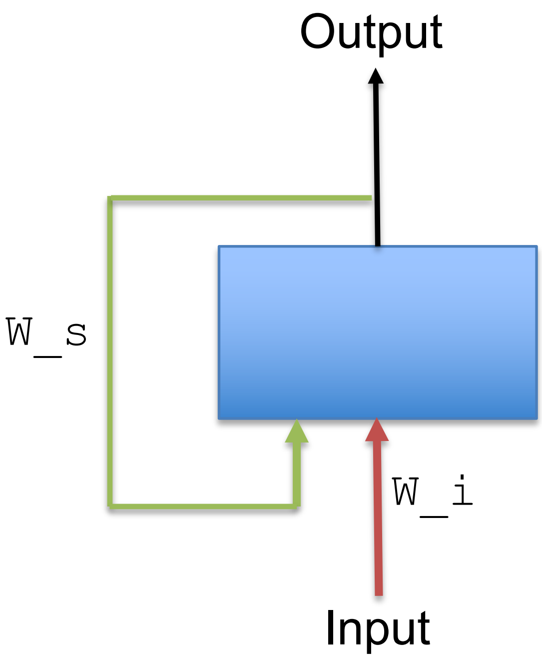
Session 23: RNN, LSTM, GRU, and Generative Models

Fall 2018
NC State University
Lecturer: Dr. Behnam Kia
Course Website: https://appliedai.wordpress.ncsu.edu/

Nov 15, 2018

RNN

```
State_t=0
For input_t in input_sequence:
    Output_t=activation(dot(W_i,input_t)+dot(W_s, state_t)+b)
    State_t=output_t
```



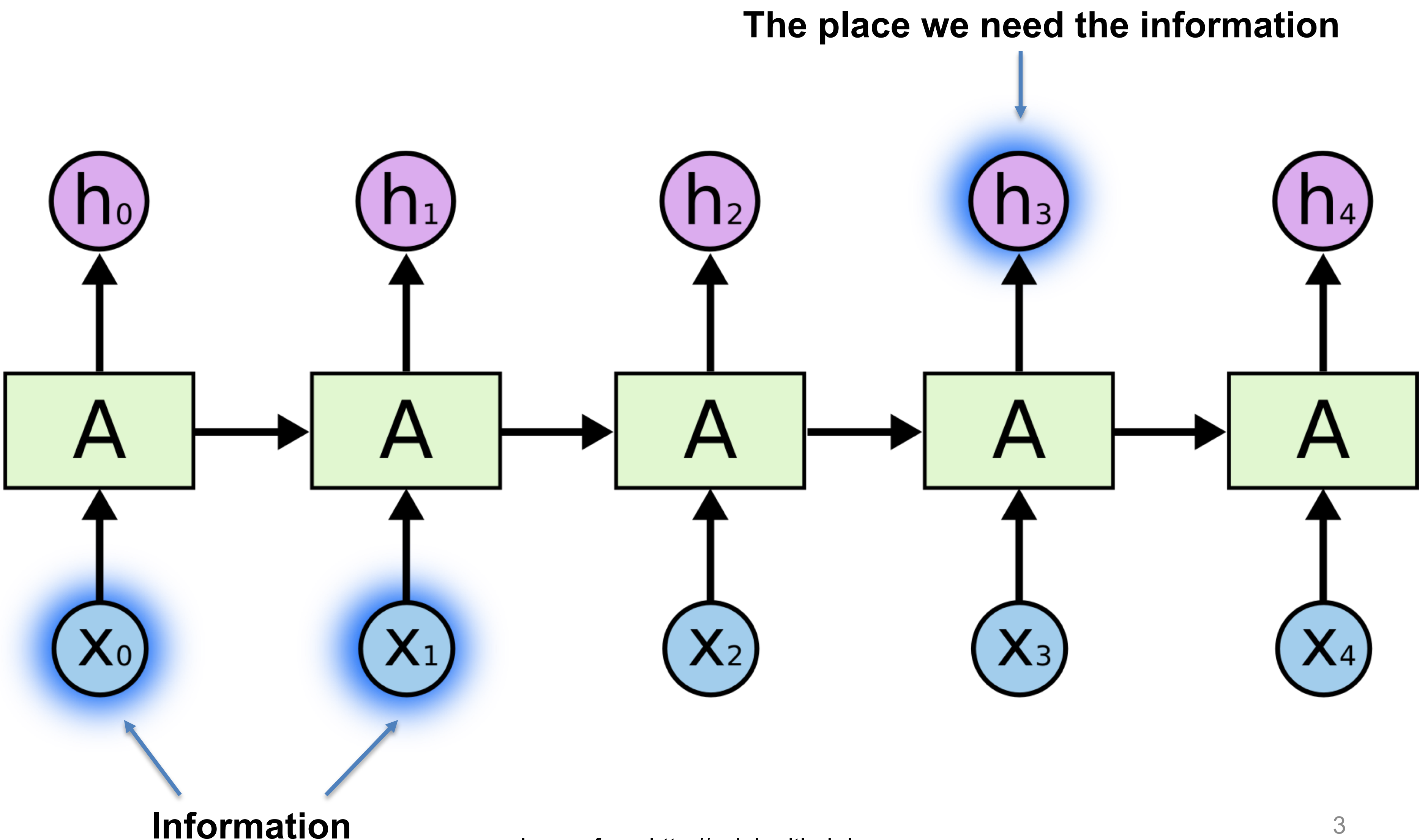


Image from http://colah.github.io

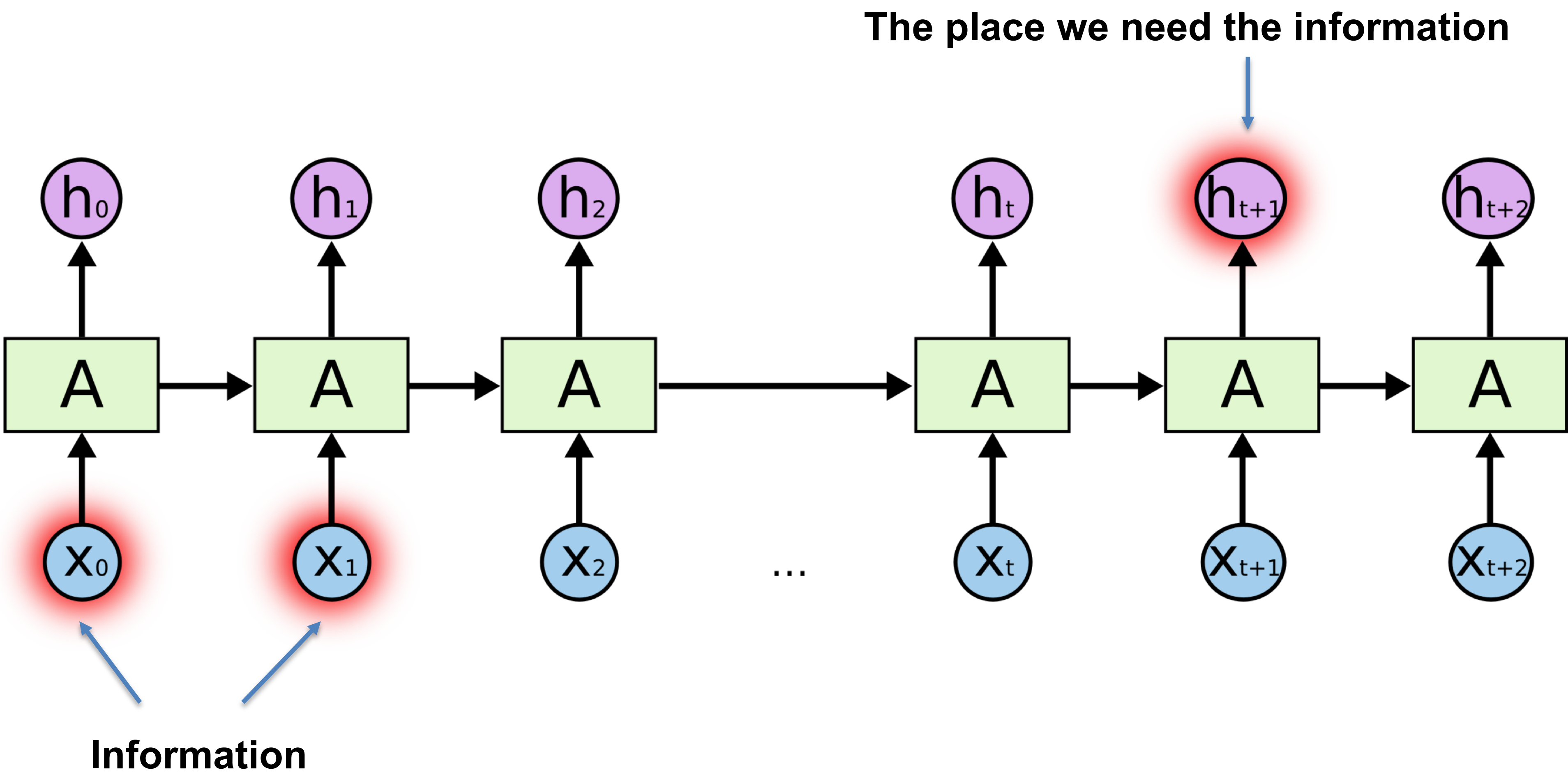


Image from http://colah.github.io

IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 5, NO. 2, MARCH 1994 157

Learning Long-Term Dependencies with Gradient Descent is Difficult

Yoshua Bengio, Patrice Simard, and Paolo Frasconi, *Student Member, IEEE*

***Abstract*— Recurrent neural networks can be used to map input sequences to output sequences, such as for recognition, production or prediction problems. However, practical difficulties have been reported in training recurrent neural networks to perform tasks in which the temporal contingencies present in the input/output sequences span long intervals. We show why gradient based learning algorithms face an increasingly difficult problem as the duration of the dependencies to be captured increases. These results expose a trade-off between efficient learning by gradient descent and latching on information for long periods. Based on an understanding of this problem, alternatives to standard gradient descent are considered.**

a fully connected recurrent network) but are local in time; i.e., they can be applied in an on-line fashion, producing a partial gradient after each time step. Another algorithm was proposed [10], [18] for training constrained recurrent networks in which dynamic neurons—with a single feedback to themselves—have only incoming connections from the input layer. It is local in time like the forward propagation algorithms and it requires computation only proportional to the number of weights, like the back-propagation through time algorithm. Unfortunately, the networks it can deal with have limited storage capabilities for dealing with general sequences

LONG SHORT-TERM MEMORY

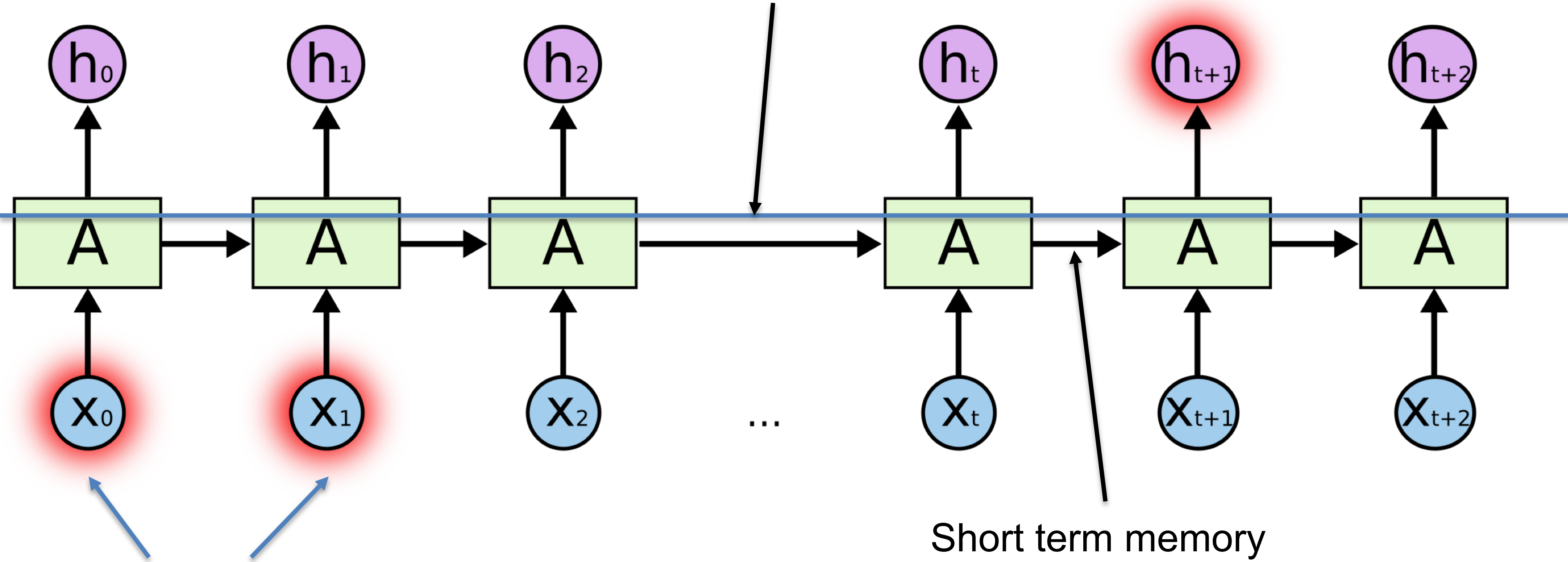
Neural Computation 9(8):1735–1780, 1997

Sepp Hochreiter
Fakultät für Informatik
Technische Universität München
80290 München, Germany
hochreit@informatik.tu-muenchen.de
http://www7.informatik.tu-muenchen.de/~hochreit

Jürgen Schmidhuber
IDSIA
Corso Elvezia 36
6900 Lugano, Switzerland
juergen@idsia.ch
http://www.idsia.ch/~juergen

The place we need the information

Long term memory

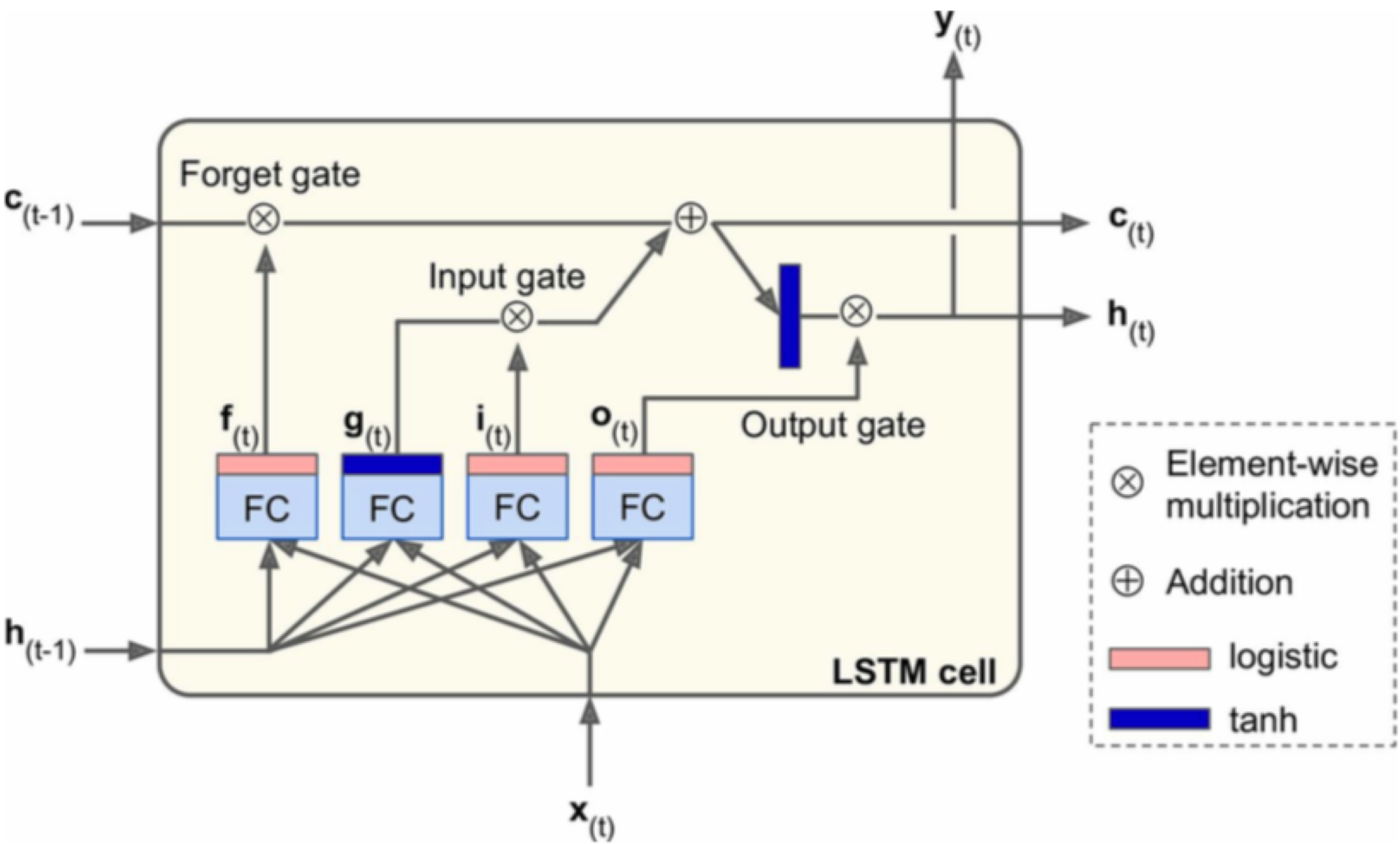


Short term memory

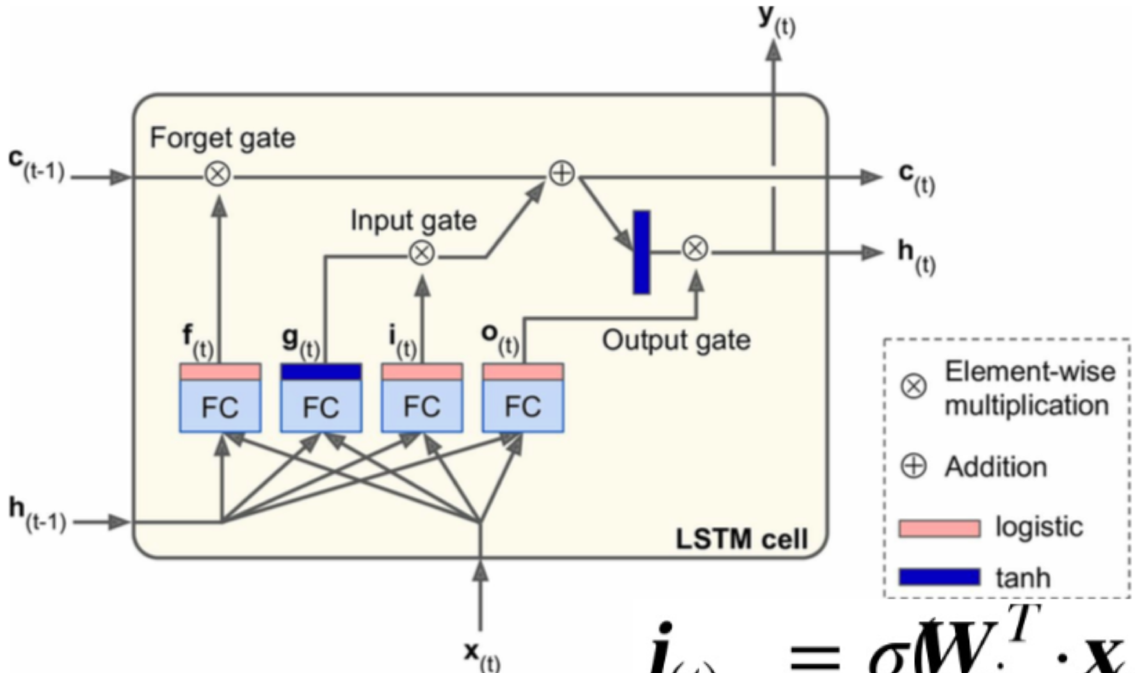
Information

Image from http://colah.github.io

LSTM



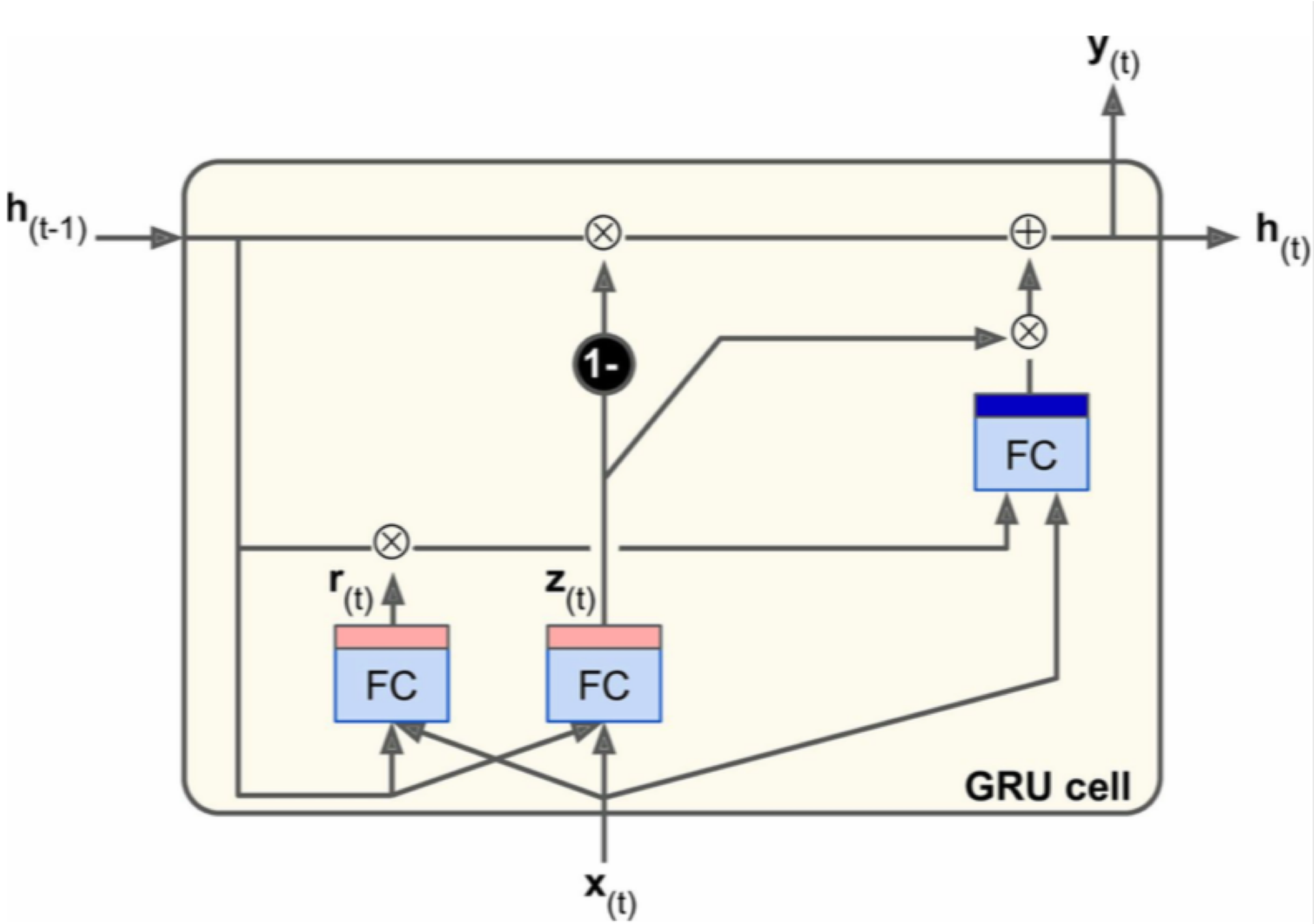
Picture from Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems



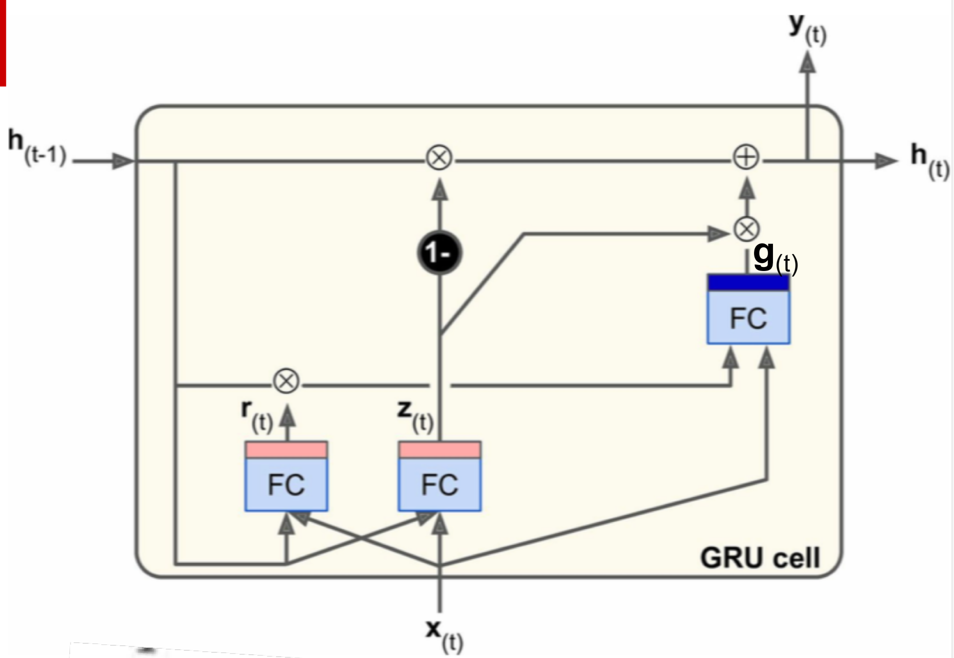
$$
\begin{aligned}
\boldsymbol{i}_{(t)} &= \sigma(\boldsymbol{W}_{xi}^{T} \cdot \boldsymbol{x}_{(t)} + \boldsymbol{W}_{hi}^{T} \cdot \boldsymbol{h}_{(t-1)} + \boldsymbol{b}_{i}) \\
\boldsymbol{f}_{(t)} &= \sigma(\boldsymbol{W}_{xf}^{T} \cdot \boldsymbol{x}_{(t)} + \boldsymbol{W}_{hf}^{T} \cdot \boldsymbol{h}_{(t-1)} + \boldsymbol{b}_{f}) \\
\boldsymbol{o}_{(t)} &= \sigma(\boldsymbol{W}_{xo}^{T} \cdot \boldsymbol{x}_{(t)} + \boldsymbol{W}_{ho}^{T} \cdot \boldsymbol{h}_{(t-1)} + \boldsymbol{b}_{o}) \\
\boldsymbol{g}_{(t)} &= \tanh(\boldsymbol{W}_{xg}^{T} \cdot \boldsymbol{x}_{(t)} + \boldsymbol{W}_{hg}^{T} \cdot \boldsymbol{h}_{(t-1)} + \boldsymbol{b}_{g}) \\
\boldsymbol{c}_{(t)} &= \boldsymbol{f}_{(t)} \otimes \boldsymbol{c}_{(t-1)} + \boldsymbol{i}_{(t)} \otimes \boldsymbol{g}_{(t)} \\
\boldsymbol{y}_{(t)} &= \boldsymbol{h}_{(t)} = \boldsymbol{o}_{(t)} \otimes \tanh(\boldsymbol{c}_{(t)})
\end{aligned}
$$

Picture from Hands-On Machine Learning w

Gated Recurrent Unit (GRU)



Picture from Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems



nt Unit (GRU)

$$\mathbf{z}_{(t)} = \sigma\left(\mathbf{W}_{xz}{}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hz}{}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_z\right)$$

$$\mathbf{r}_{(t)} = \sigma\left(\mathbf{W}_{xr}{}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hr}{}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_r\right)$$

$$\mathbf{g}_{(t)} = \tanh\left(\mathbf{W}_{xg}{}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hg}{}^{T} \cdot \left(\mathbf{r}_{(t)} \otimes \mathbf{h}_{(t-1)}\right) + \mathbf{b}_g\right)$$

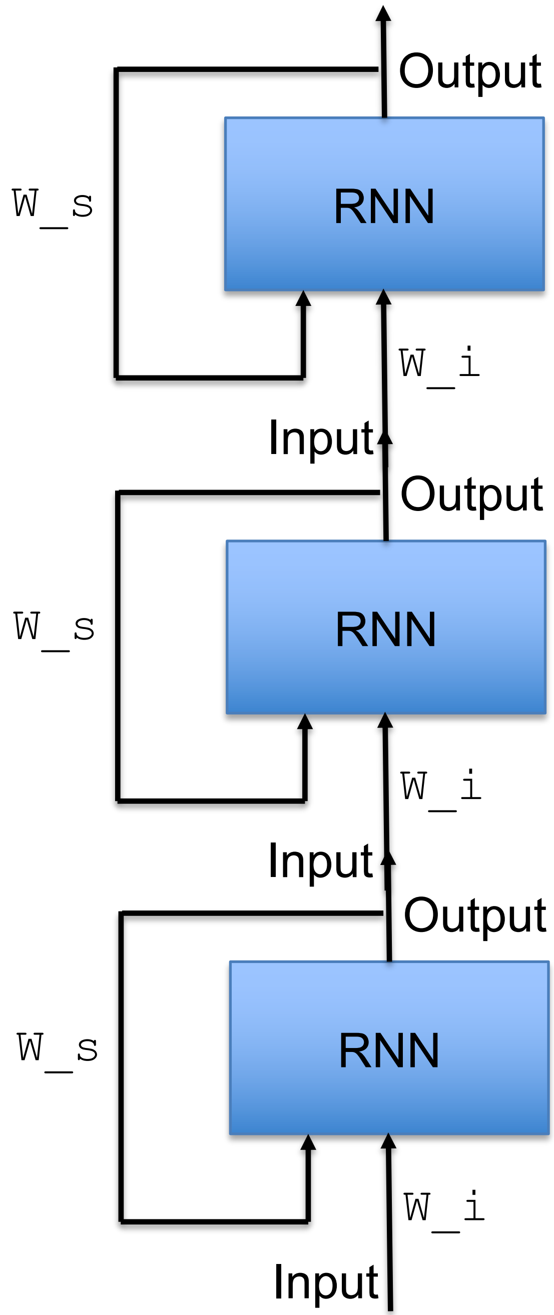
$$\mathbf{h}_{(t)} = \mathbf{z}_{(t)} \otimes \mathbf{h}_{(t-1)} + \left(1 - \mathbf{z}_{(t)}\right) \otimes \mathbf{g}_{(t)}$$

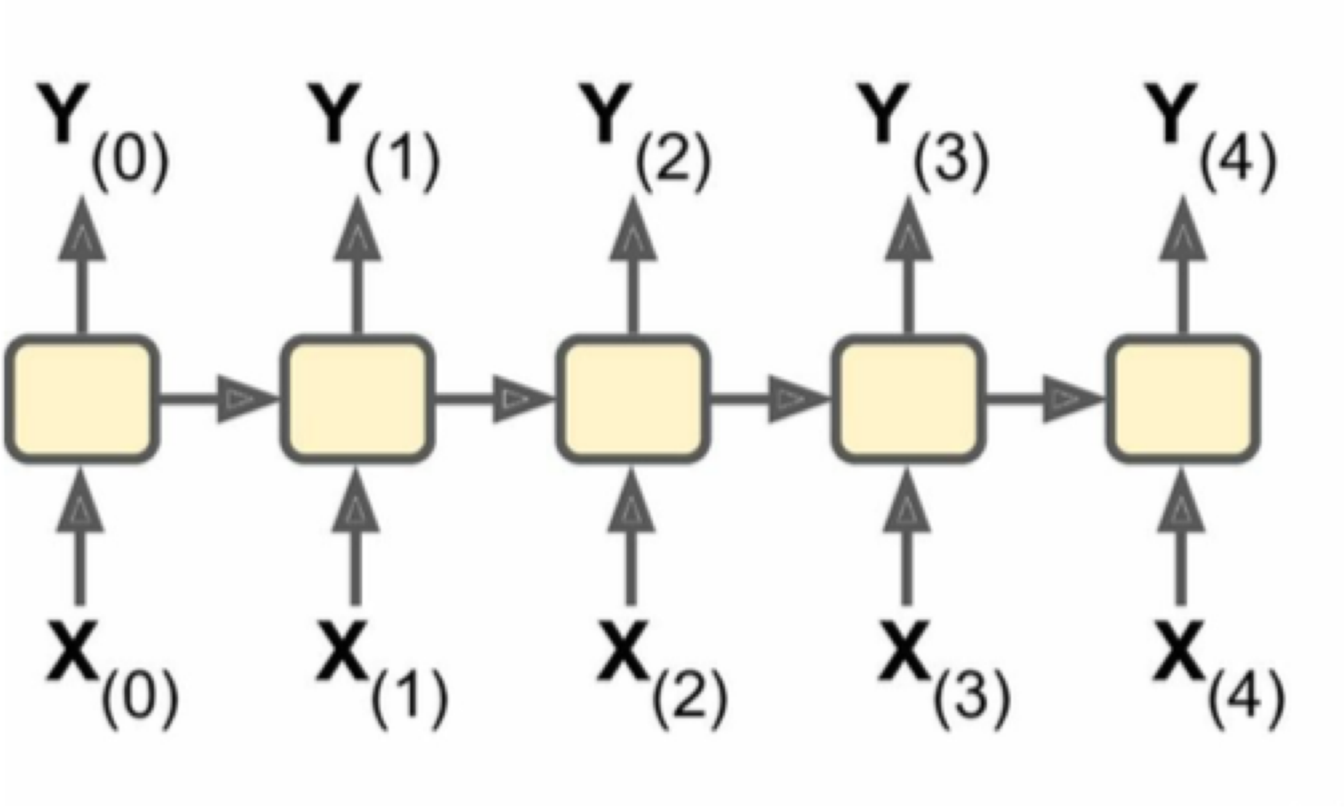
Picture from Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems

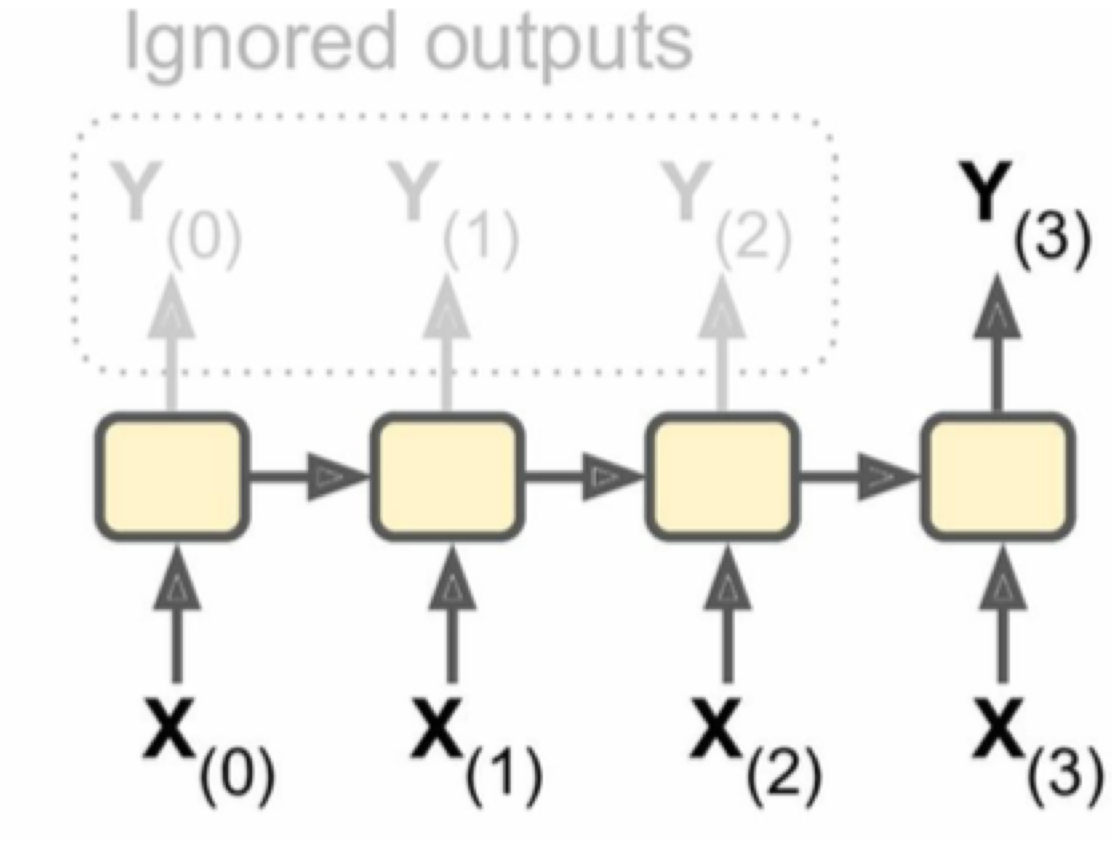
A different perspective to designing models:

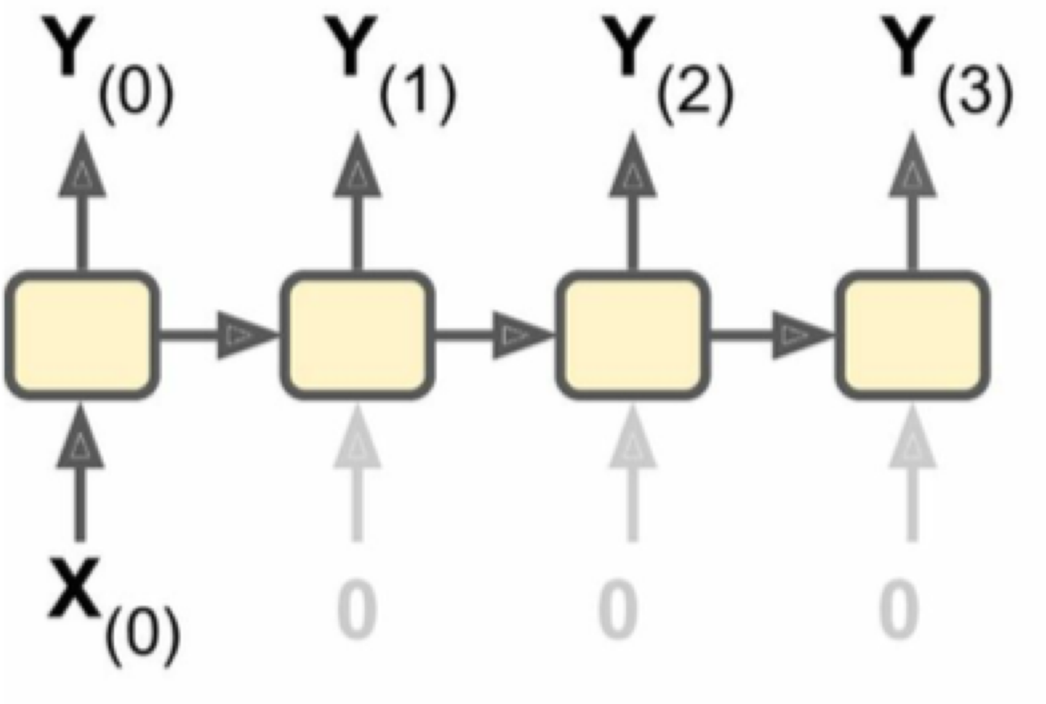
To a researcher, it seems that the choice of such constraints—the question of how to implement RNN cells—is better left to optimization algorithms (like genetic algorithms or reinforcement learning processes) than to human engineers. And in the future, that's how we'll build networks. In summary: you don't need to understand anything about the specific architecture of an LSTM cell; as a human, it shouldn't be your job to understand it. Just keep in mind what the LSTM cell is meant to do: allow past information to be reinjected at a later time, thus fighting the vanishing-gradient problem.

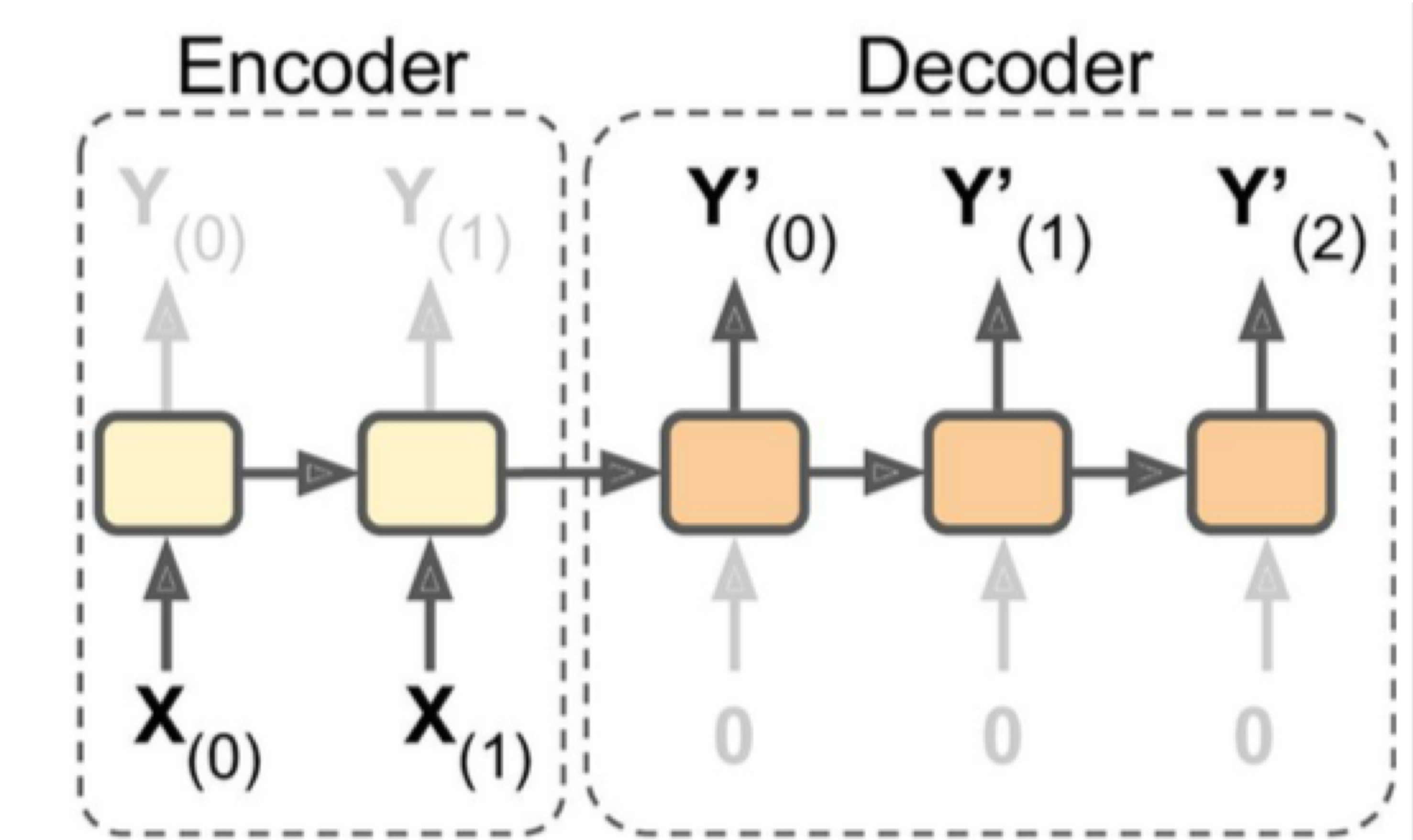
- Francois Chollet (Deep Learning with Python)

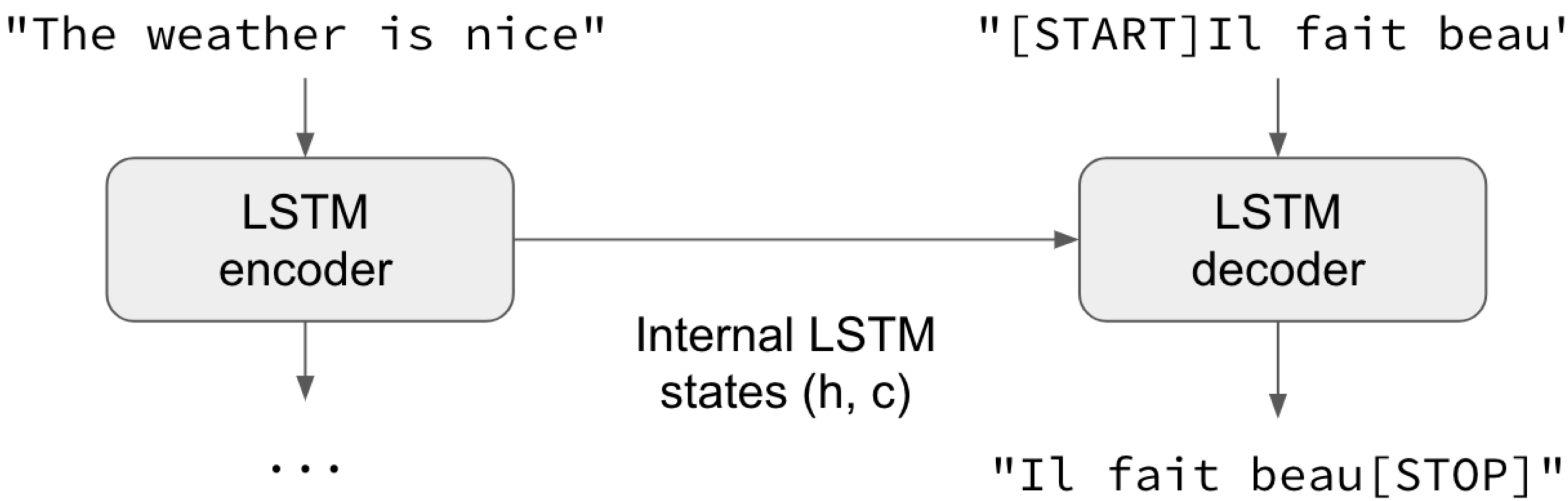

Output
W_s
RNN
W_i
Input
Output
W_s
RNN
W_i
Input
Output
W_s
RNN
W_i
Input


$Y_{(0)}$
$Y_{(1)}$
$Y_{(2)}$
$Y_{(3)}$
$Y_{(4)}$
$X_{(0)}$
$X_{(1)}$
$X_{(2)}$
$X_{(3)}$
$X_{(4)}$


Ignored outputs
$Y_{(0)}$
$Y_{(1)}$
$Y_{(2)}$
$Y_{(3)}$
$X_{(0)}$
$X_{(1)}$
$X_{(2)}$
$X_{(3)}$

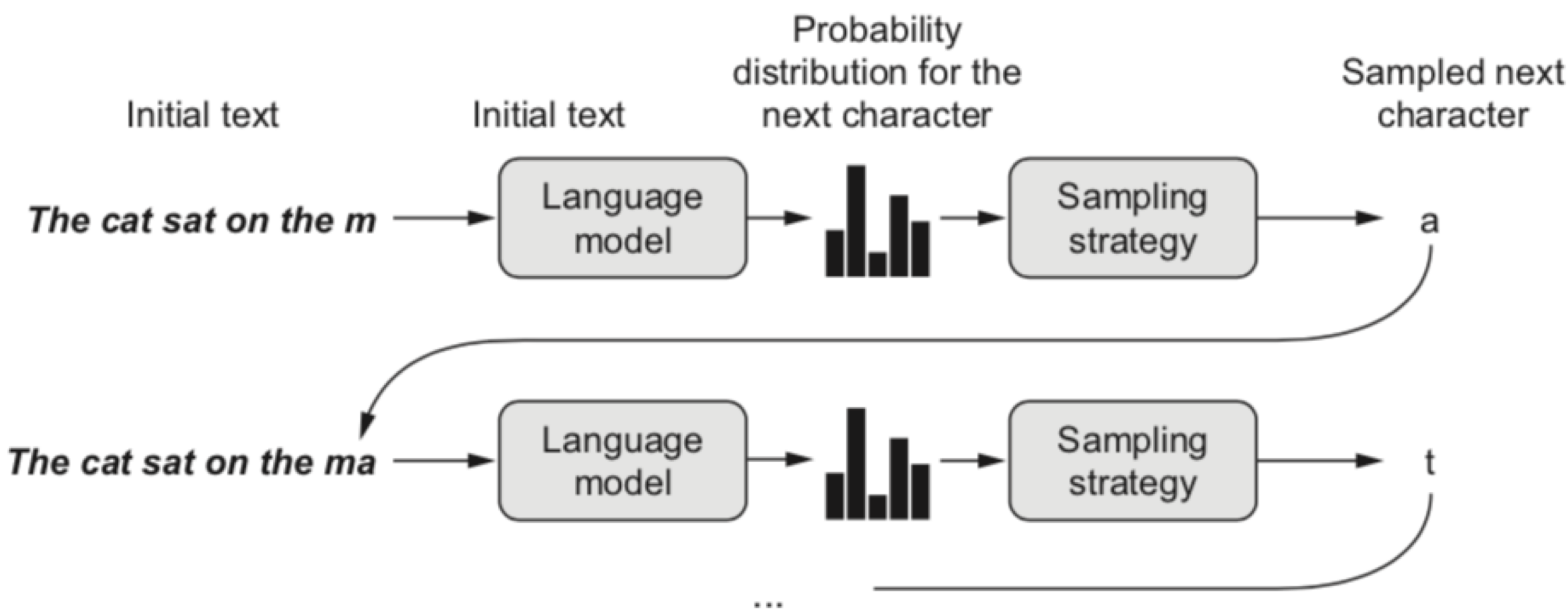

$Y_{(0)}$
$Y_{(1)}$
$Y_{(2)}$
$Y_{(3)}$
$X_{(0)}$
0
0
0


Encoder
Decoder
Y(0)
Y(1)
Y'(0)
Y'(1)
Y'(2)
X(0)
X(1)
0
0
0



- Please see “A ten-minute introduction to sequence-to-sequence learning in Keras”

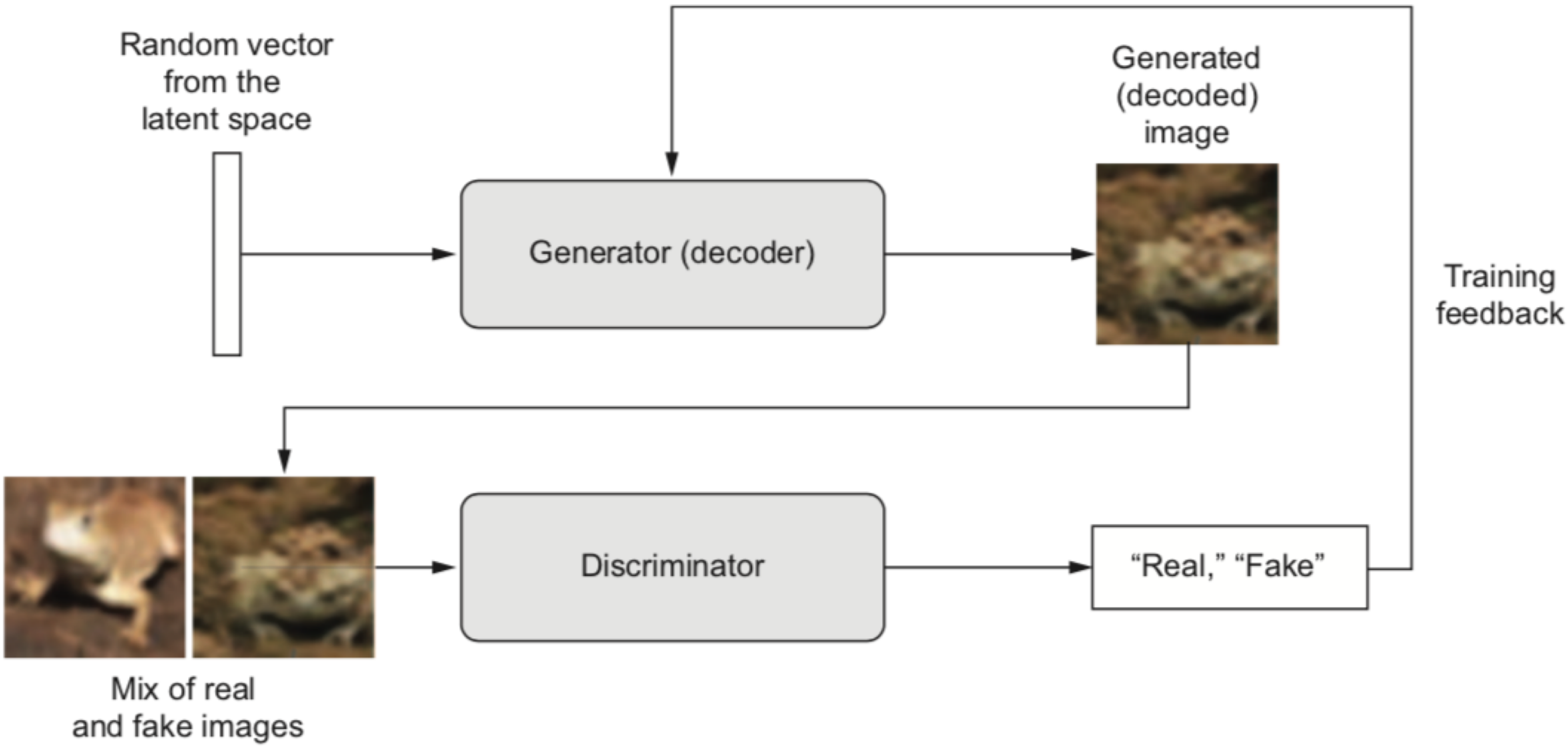
Generative Recurrent Network



- Please see Deep Learning with Python book.

Picture from Brock, Andrew, Jeff Donahue, and Karen Simonyan. "Large scale gan training for high fidelity natural image synthesis." *arXiv preprint arXiv:1809.11096* (2018).

Generative Adversarial Network (GAN)



Picture from Deep Learning with Python, Francois Chollet