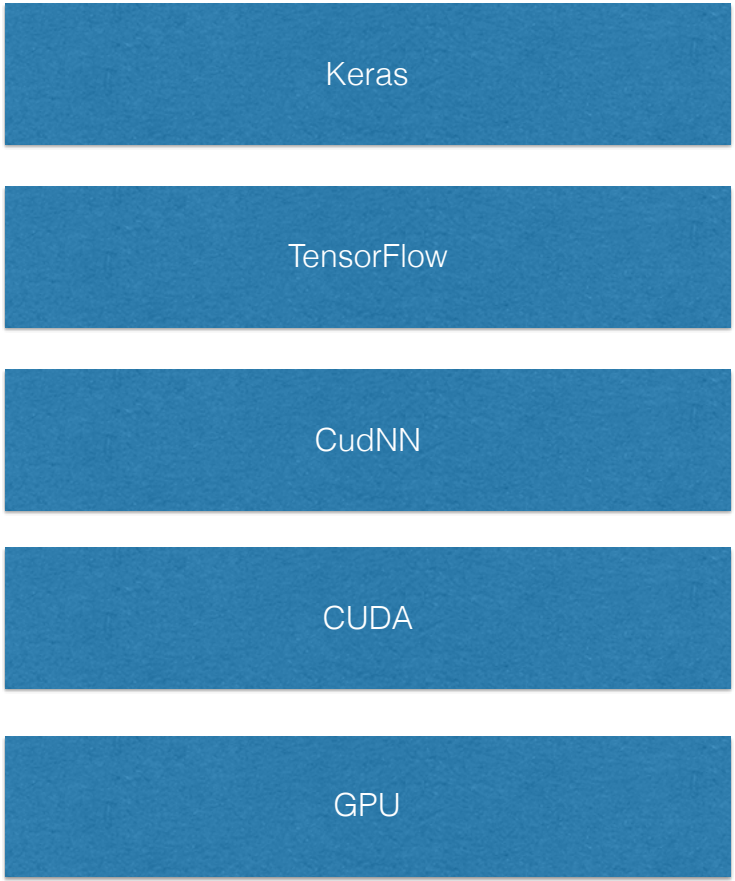
# Slides, Support, Code and Setup instructions: bit.ly/hub-setup

# Introductions

# Keras/LSTM Text Classification Agenda

| | | |
|---|---|---|
| 9:00 - 9:45 | Keras/CNN Review | examples/keras-sign |
| 9:45 - 10:30 | Simple RNNs and Time Series | examples/lstm/time-series |
| 10:30 - 11:00 | Break | |
| 10:30 - 11:00 | LSTMs, GRUs applied to Text Generation | examples/lstm/text-gen |
| 11:00 - 12:30 | Text Classification with Word Embeddings, Hybrid CNN/LSTMs | examples/lstm/imdb-classifier |

Code: [github.com/lukas/ml-class](github.com/lukas/ml-class)

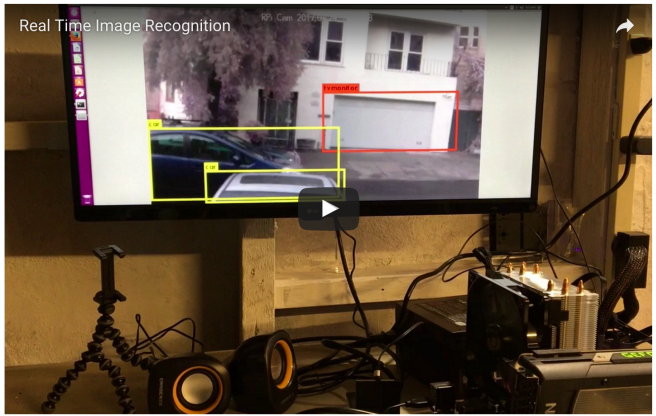

Keras
TensorFlow
CudNN
CUDA
GPU

# Build your own box

## Build a super fast deep learning machine for under $1,000

The adventures in deep learning and cheap hardware continue!

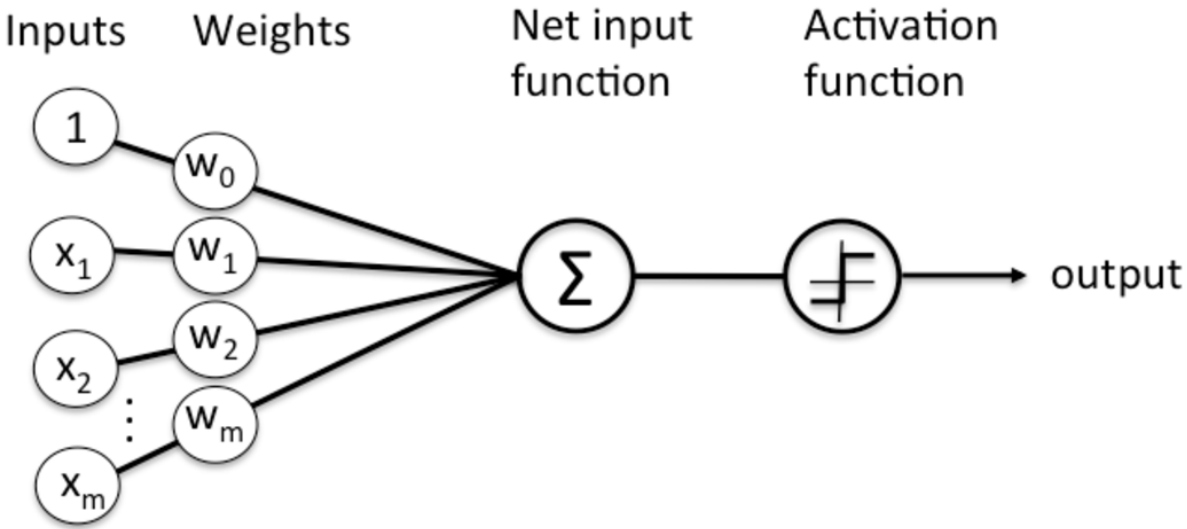
By Lukas Biewald. February 1, 2017




https://www.oreilly.com/learning/build-a-super-fast-deep-learning-machine-for-under-1000

# Keras Review (ml-class/examples/keras-sign)

# Perceptron




**Schematic of Rosenblatt's perceptron.**

perceptron-single.py

| | | | |
|---|---|---|---|
| 0 | 91 | 95 | 0 |
| 0 | 0 | 99 | 0 |
| 0 | 0 | 35 | 0 |
| 0 | 0 | 94 | 0 |

Input (28x28)

Flatten (1x784)

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 91 | 95 | 0 | 0 | 0 | 99 | 0 | 0 | 0 | 35 | 0 | 0 | 0 | 94 | 0 |

…

Weight1 = -0.01

Weight16 = 0.02

Σ 0.3

Dense (1x1)

1.0

Target

perceptron-linear.py

| | | | |
|---|---|---|---|
| 0 | 91 | 95 | 0 |
| 0 | 0 | 99 | 0 |
| 0 | 0 | 35 | 0 |
| 0 | 0 | 94 | 0 |

Input (28x28)

Flatten (1x784)

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 91 | 95 | 0 | 0 | 0 | 99 | 0 | 0 | 0 | 35 | 0 | 0 | 0 | 94 | 0 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0.01 | 0.85 | 0.00 | 0.1 | 0.01 | 0.02 | 0.00 | 0.01 | 0.02 | 0.01 |

Dense (1x10)

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

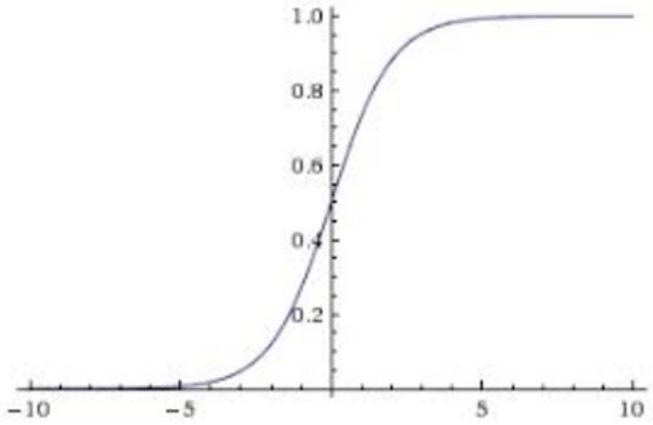
Target

# One hot encoding

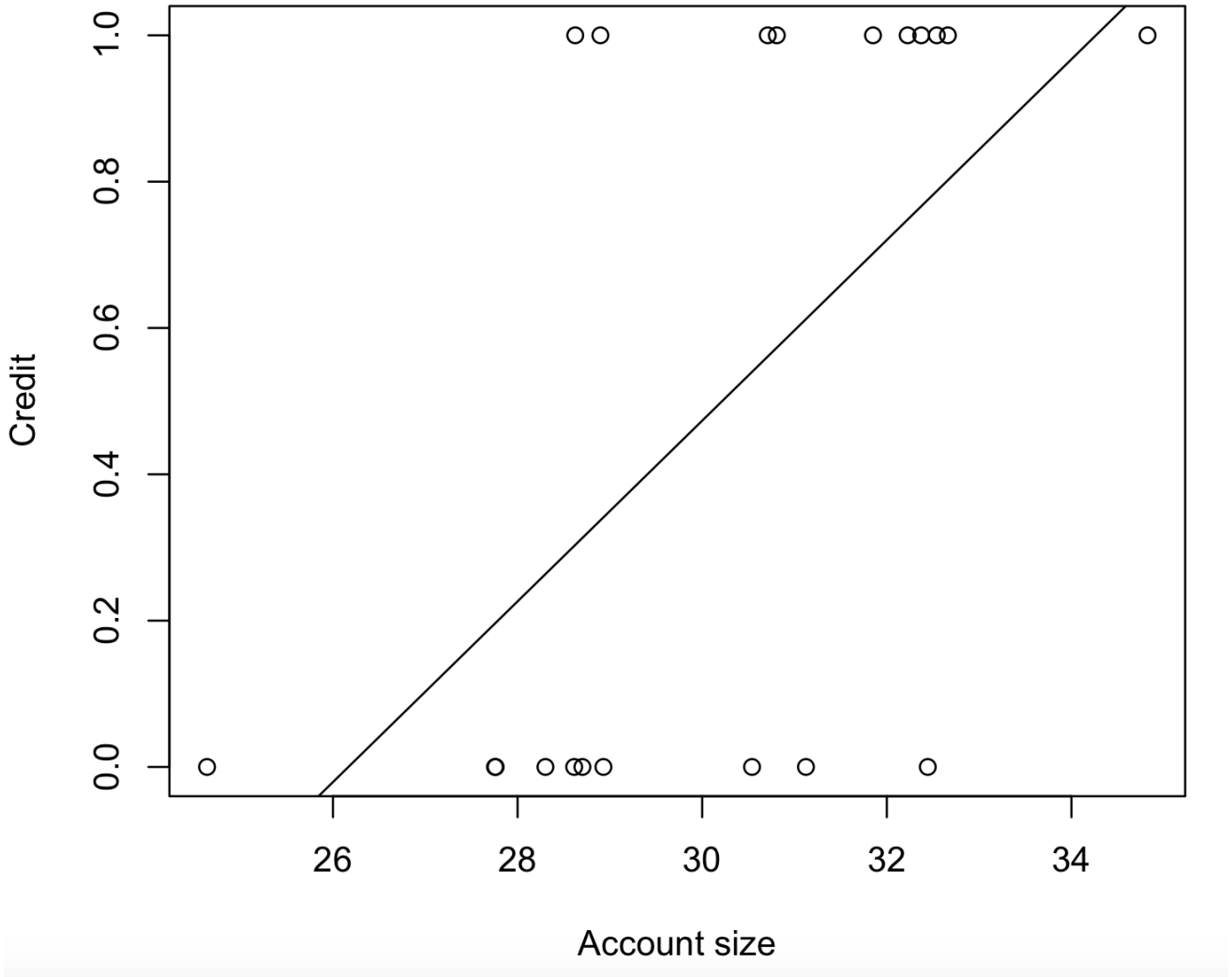
| Label |
|---|
| 0 |
| 4 |
| 4 |
| 3 |
| 0 |
| 9 |

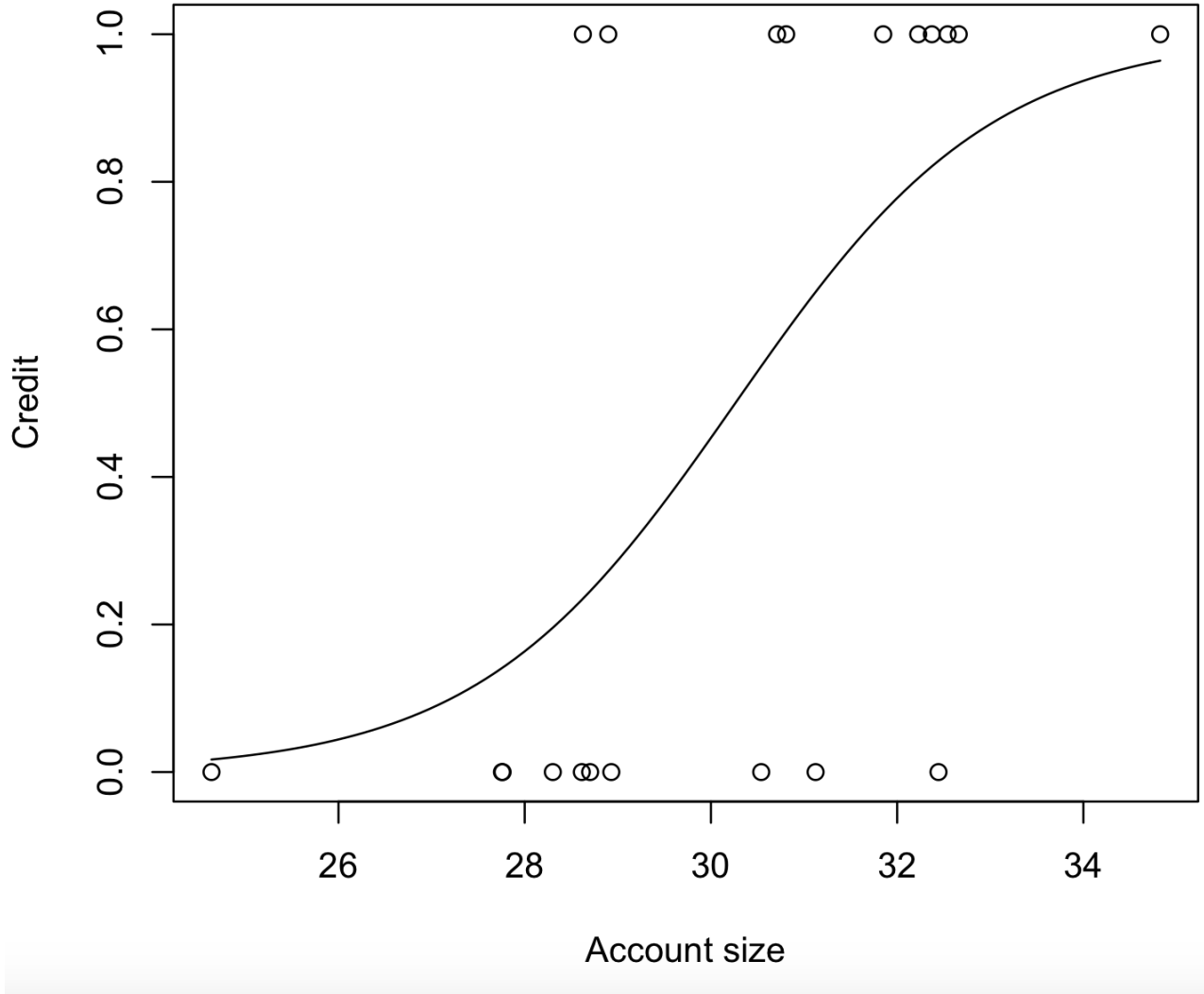
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

# Activation Functions: Sigmoid

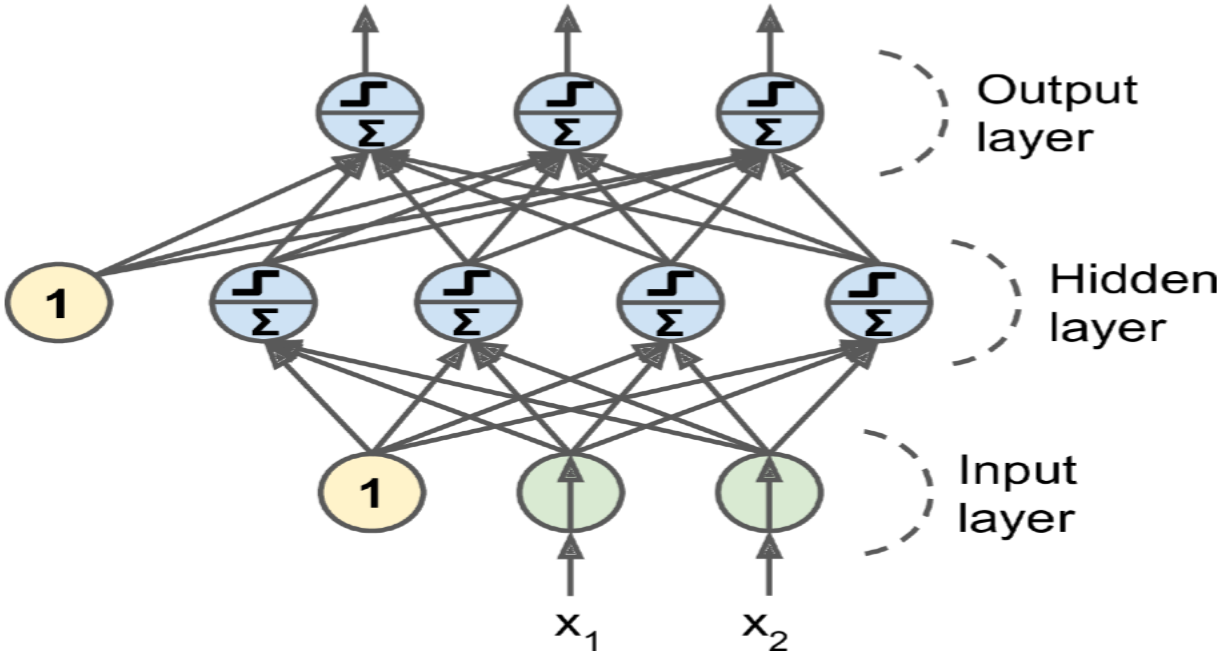



(special case of softmax and logistic)

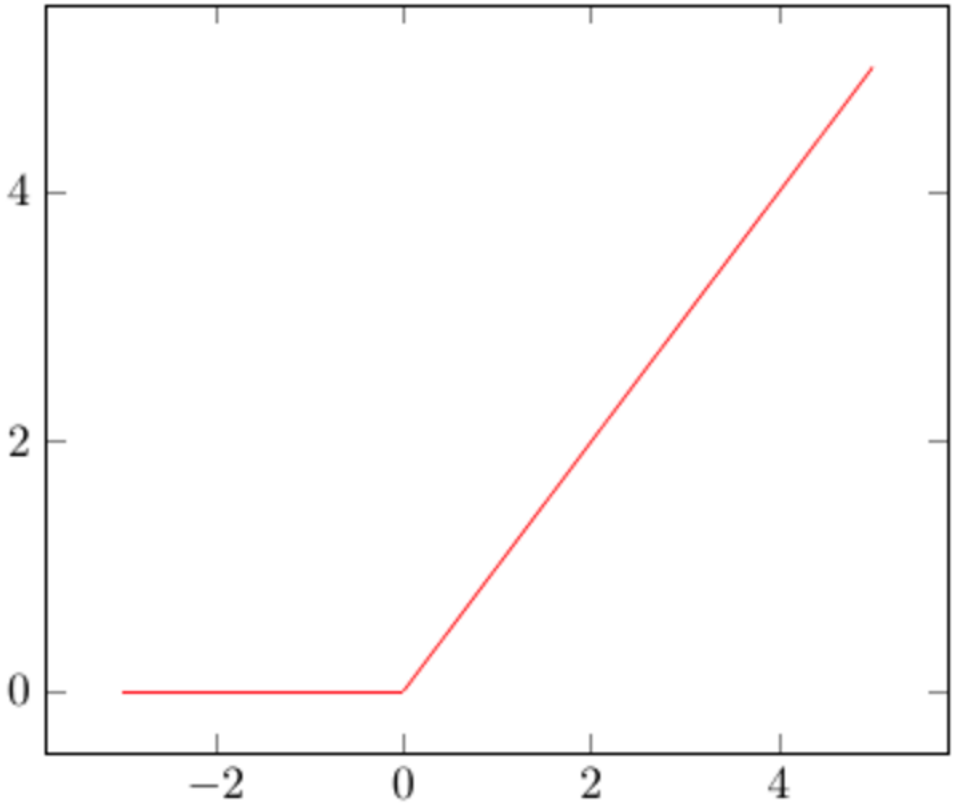

Credit
0.0
0.2
0.4
0.6
0.8
1.0
26
28
30
32
34
Account size


Credit
0.0
0.2
0.4
0.6
0.8
1.0
26
28
30
32
34
Account size

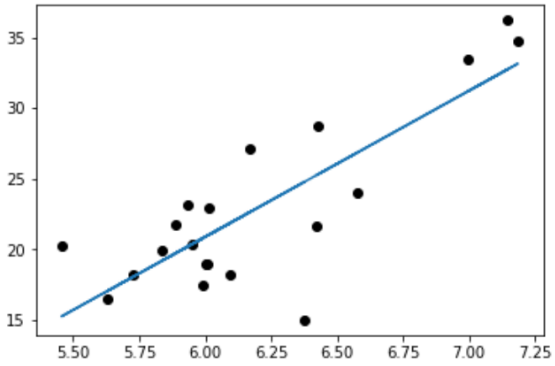
# Two Layers of Perceptrons

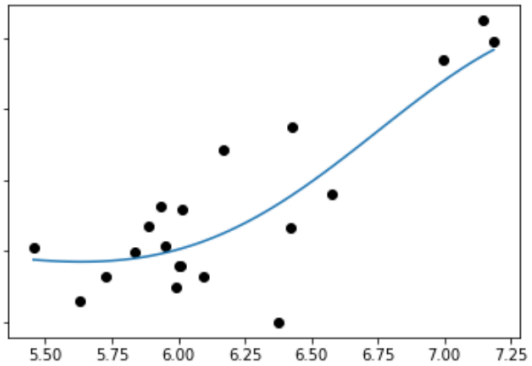


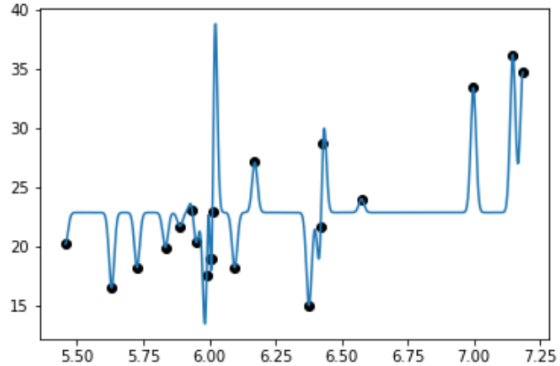
# MLP Activation Function: ReLU



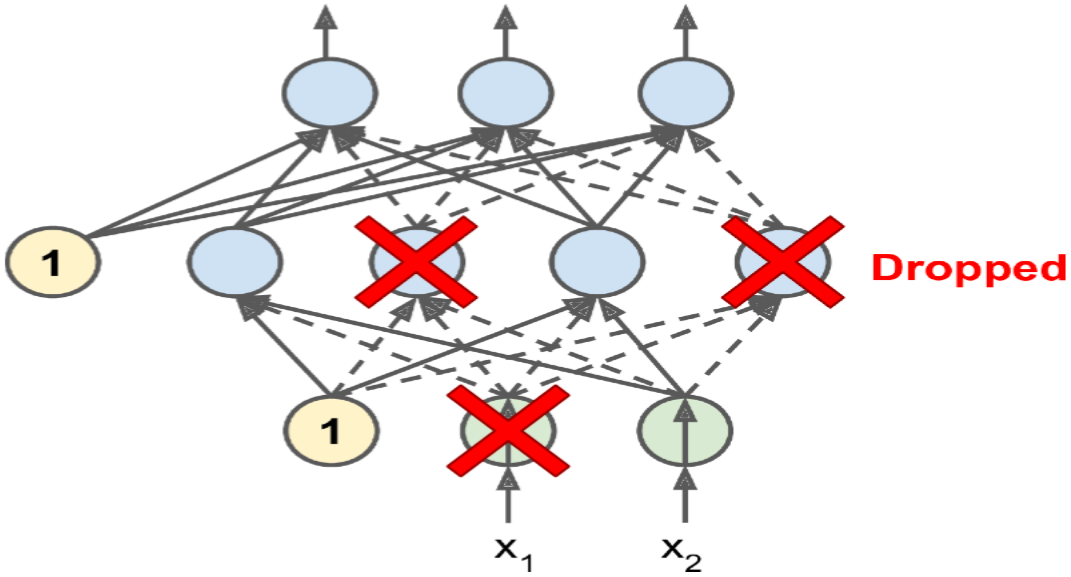
# Overfitting

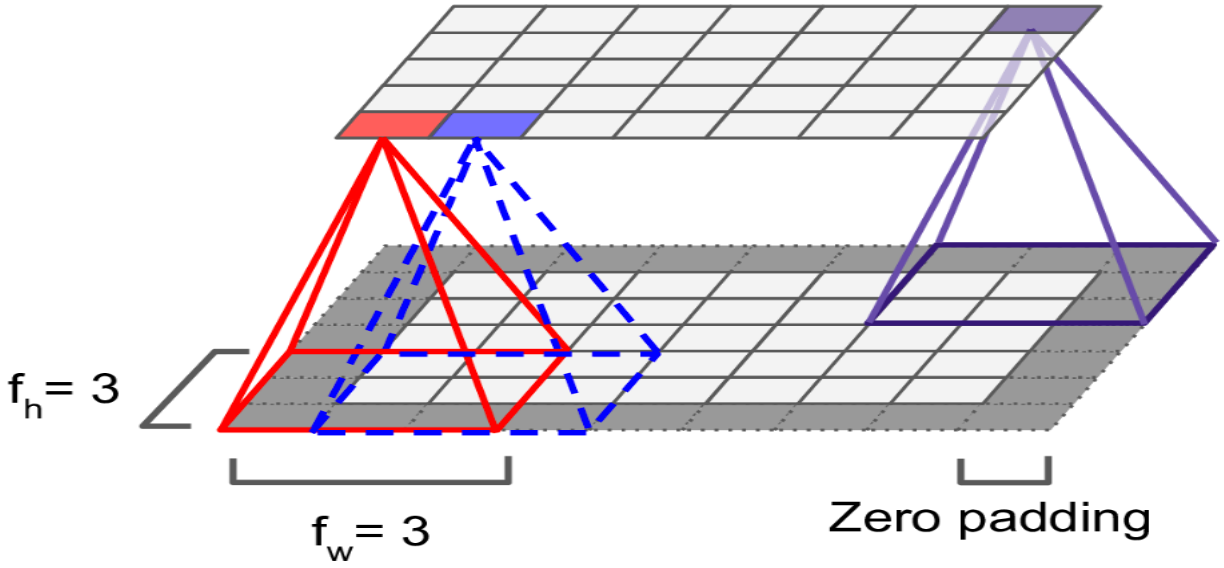


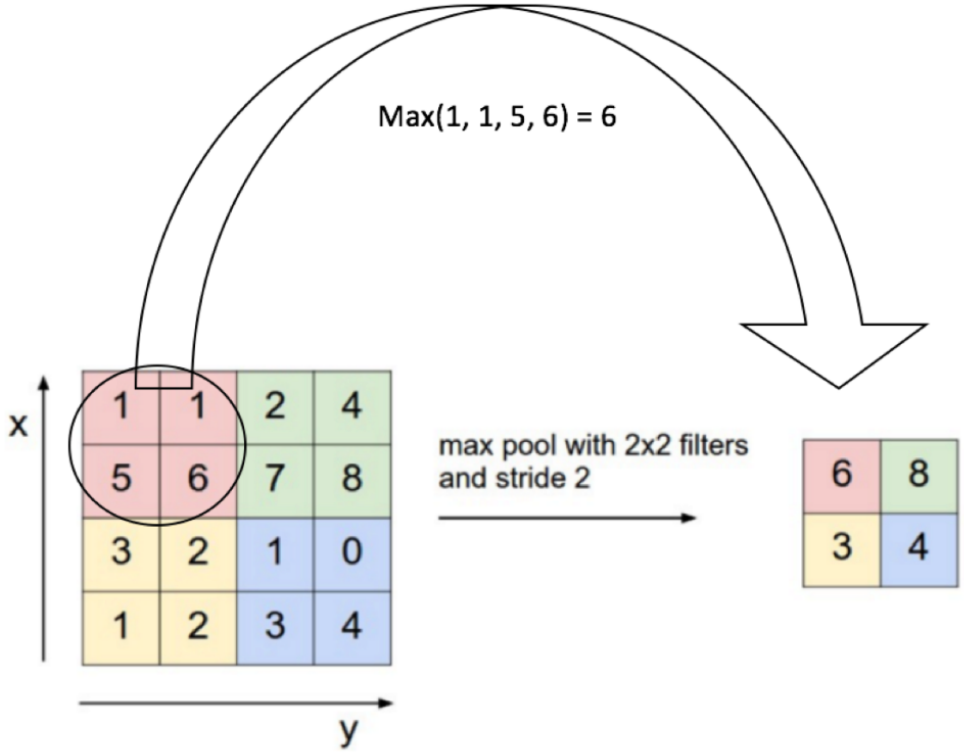



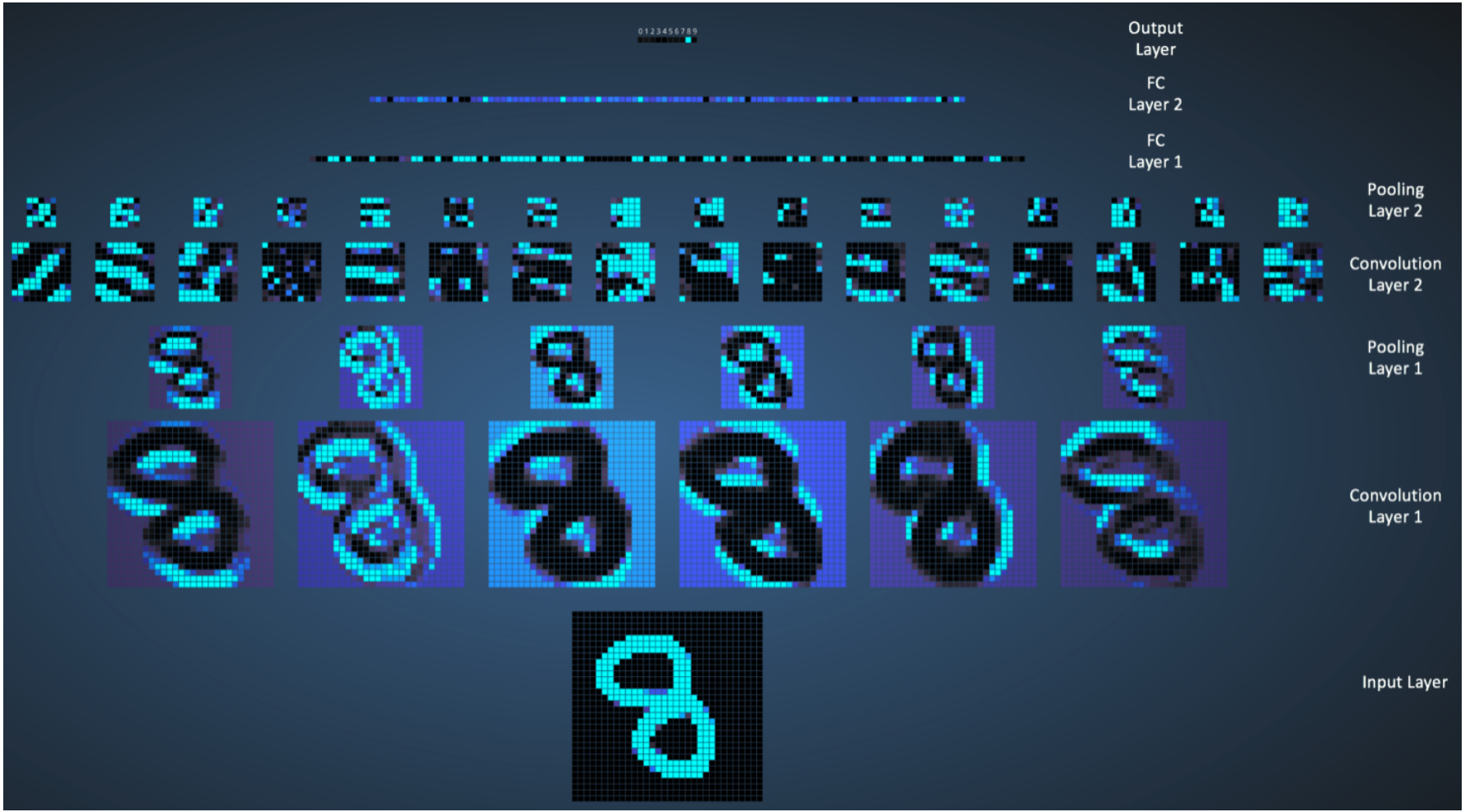
# Dropout




$f_h = 3$
$f_w = 3$
Zero padding

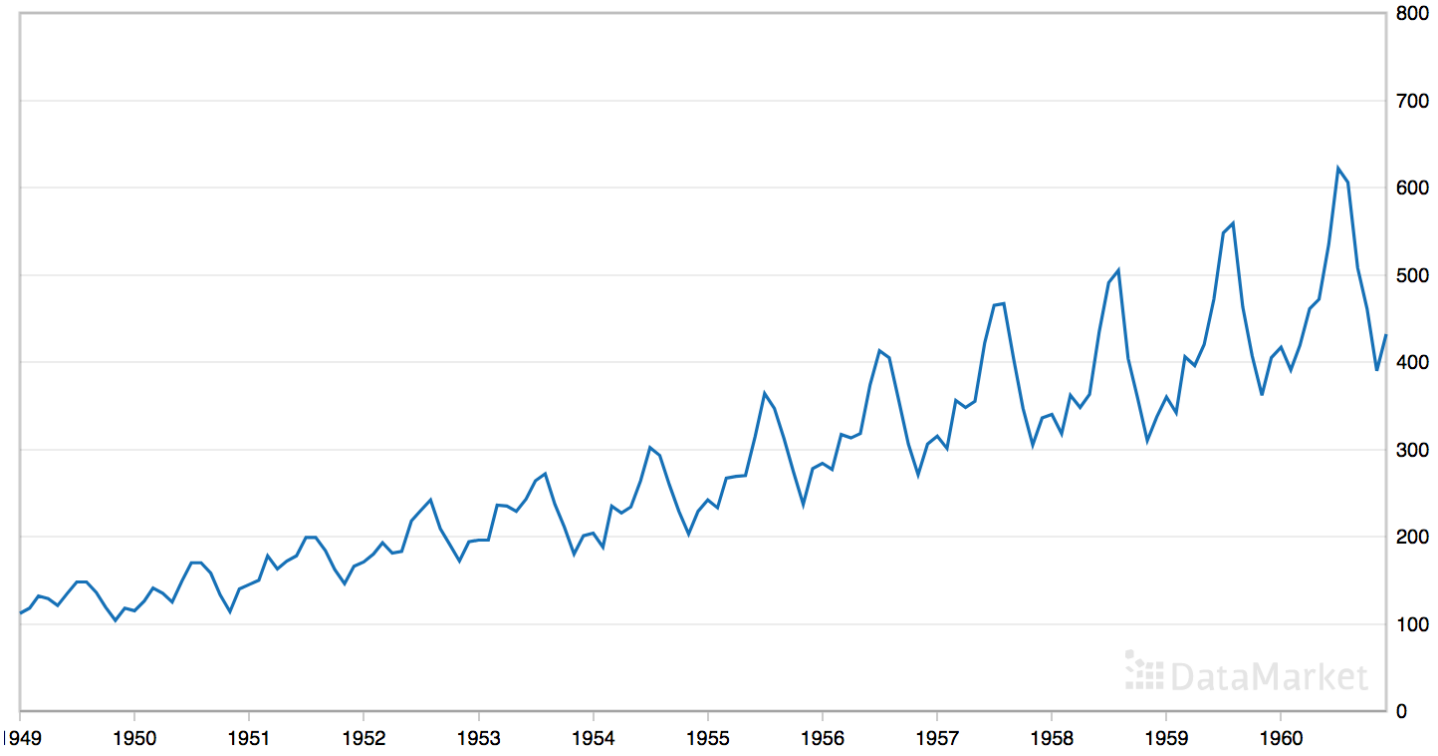
# Pooling


Max(1, 1, 5, 6) = 6
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

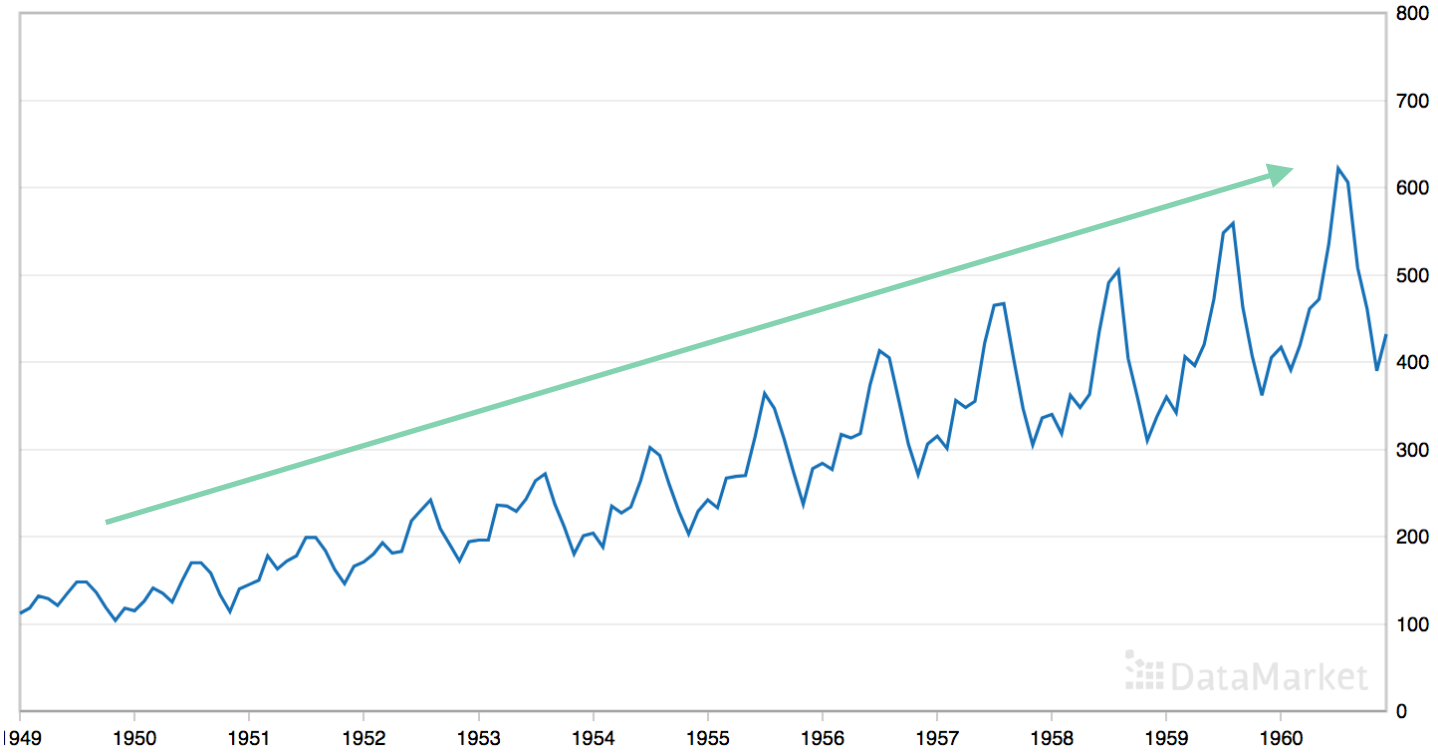

0123456789
Output Layer
FC Layer 2
FC Layer 1
Pooling Layer 2
Convolution Layer 2
Pooling Layer 1
Convolution Layer 1
Input Layer

# Time Series and RNNs (ml-class/examples/lstm/time-series)

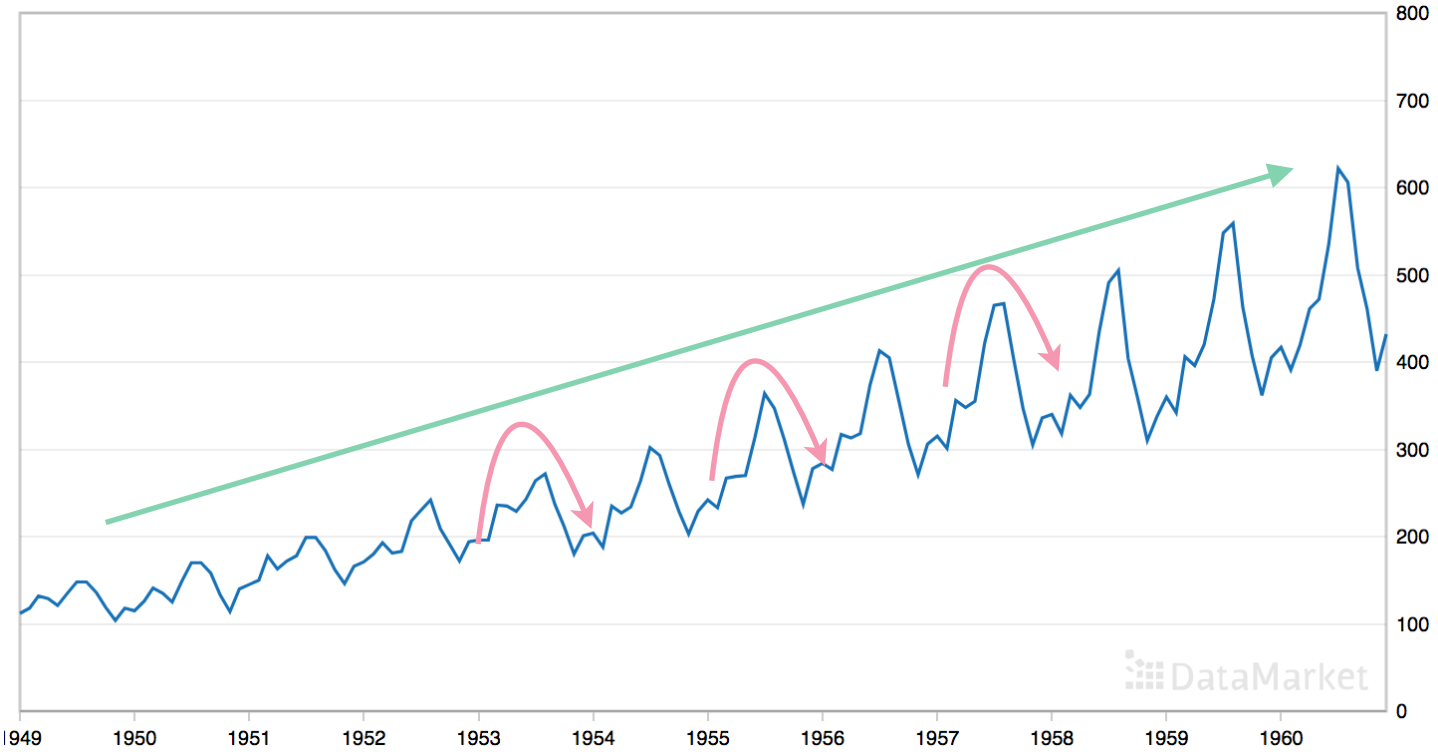
# Timeseries (Airline Sales)



# Timeseries (Airline Sales)



# Timeseries (Airline Sales)



# Framing Timeseries as an ML problem

| Time | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Airline Sales | 1 | 3 | 4 | 7 | 11 | 18 | 29 | 31 | 42 | 55 | 62 | 74 | 78 |

# Framing Timeseries as an ML problem

| Time | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Airline Sales | 1 | 3 | 4 | 7 | 11 | 18 | 29 | 31 | 42 | 55 | 62 | 74 | 78 |

Train

Label

| X1 | X2 | X3-9 | X10 | Label |
|---|---|---|---|---|
| 1 | 3 | … | 55 | 62 |

# Framing Timeseries as an ML problem

| Time | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Airline Sales | 1 | 3 | 4 | 7 | 11 | 18 | 29 | 31 | 42 | 55 | 62 | 74 | 78 |

Train

Label

| X1 | X2 | X3-9 | X10 | Label |
|---|---|---|---|---|
| 1 | 3 | … | 55 | 62 |
| 3 | 4 | … | 62 | 74 |

# Framing Timeseries as an ML problem

| Time | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Airline Sales | 1 | 3 | 4 | 7 | 11 | 18 | 29 | 31 | 42 | 55 | 62 | 74 | 78 |

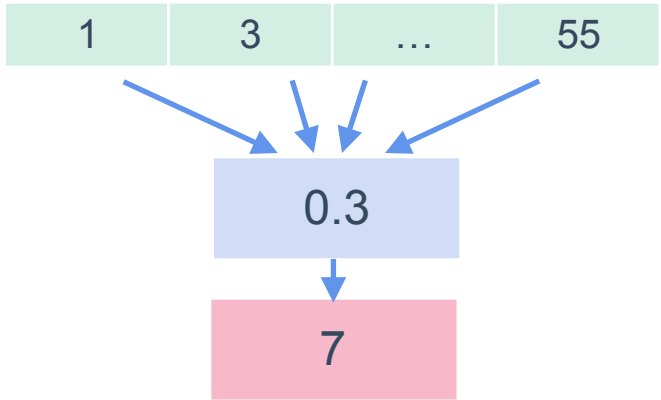
Train

Label

| X1 | X2 | X3-9 | X10 | Label |
|---|---|---|---|---|
| 1 | 3 | … | 55 | 62 |
| 3 | 4 | … | 62 | 74 |
| 4 | 7 | … | 74 | 78 |

# Perceptron Classifier on Timeseries

| X1 | X2 | X3-9 | X10 | Label |
|---|---|---|---|---|
| 1 | 3 | … | 55 | 62 |
| 3 | 4 | … | 62 | 74 |
| 4 | 7 | … | 74 | 78 |

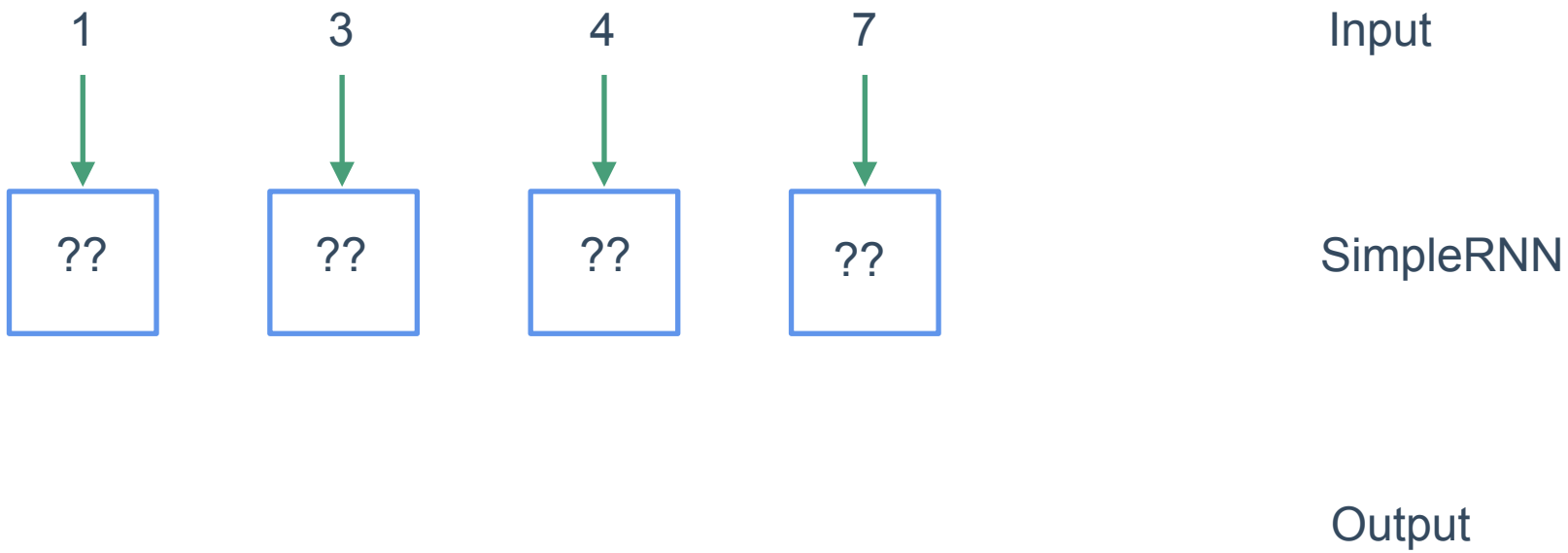



Input

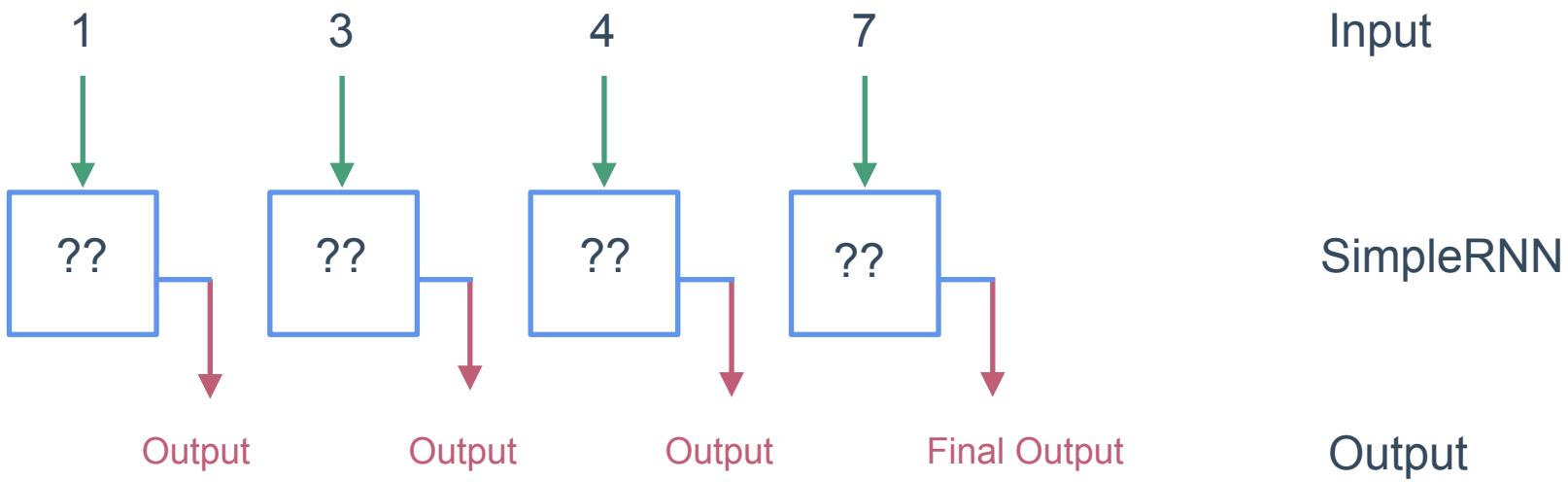
Dense (1x1)

Target

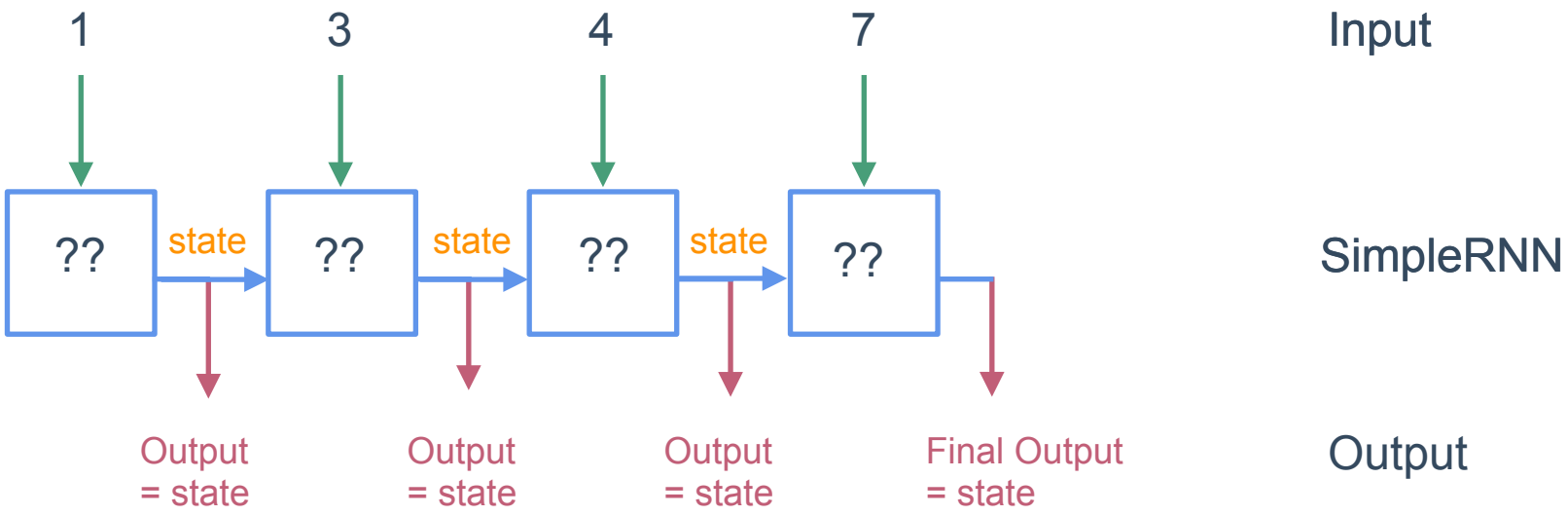
# Recurrent Neural Network (RNN)



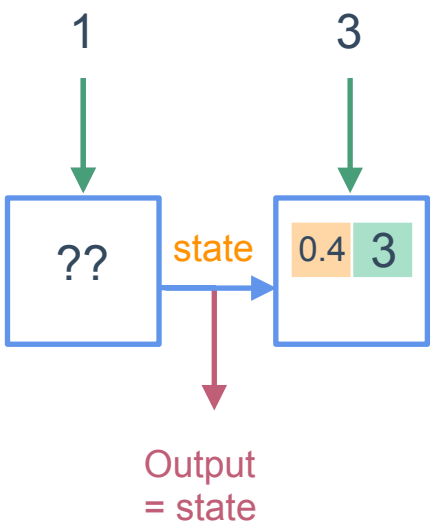
# Recurrent Neural Network (RNN)



# Recurrent Neural Network (RNN)



# RNN




Input

SimpleRNN

Output

# Inside the RNN

| State | Input |
| --- | --- |
| 0.4 | 3 |

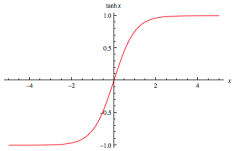
Input

| | |
| --- | --- |
| 0.2 | 0.4 |

Learned Weights

0.2*0.4 + 0.4*3 = 1.28

Weighted Sum

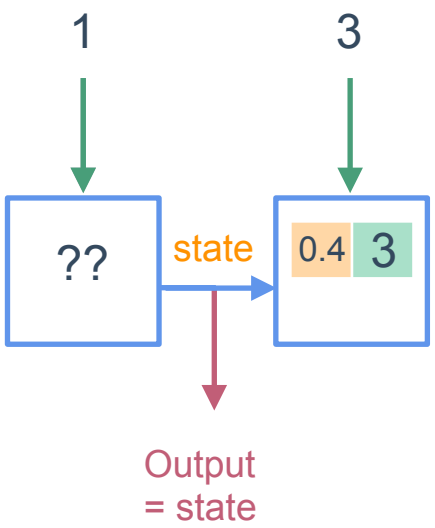


Activation Function

**0.9**

Output

# RNN

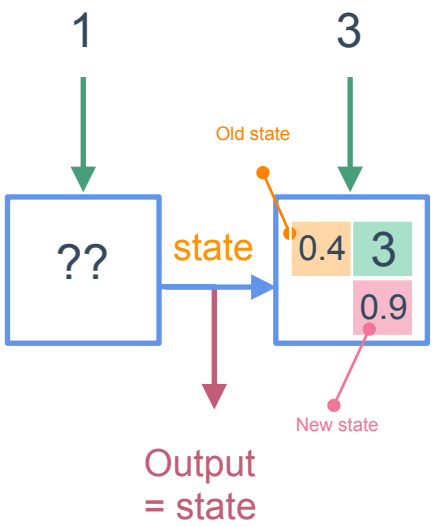



Input

SimpleRNN

Output

# Recurrent Neural Network (RNN)

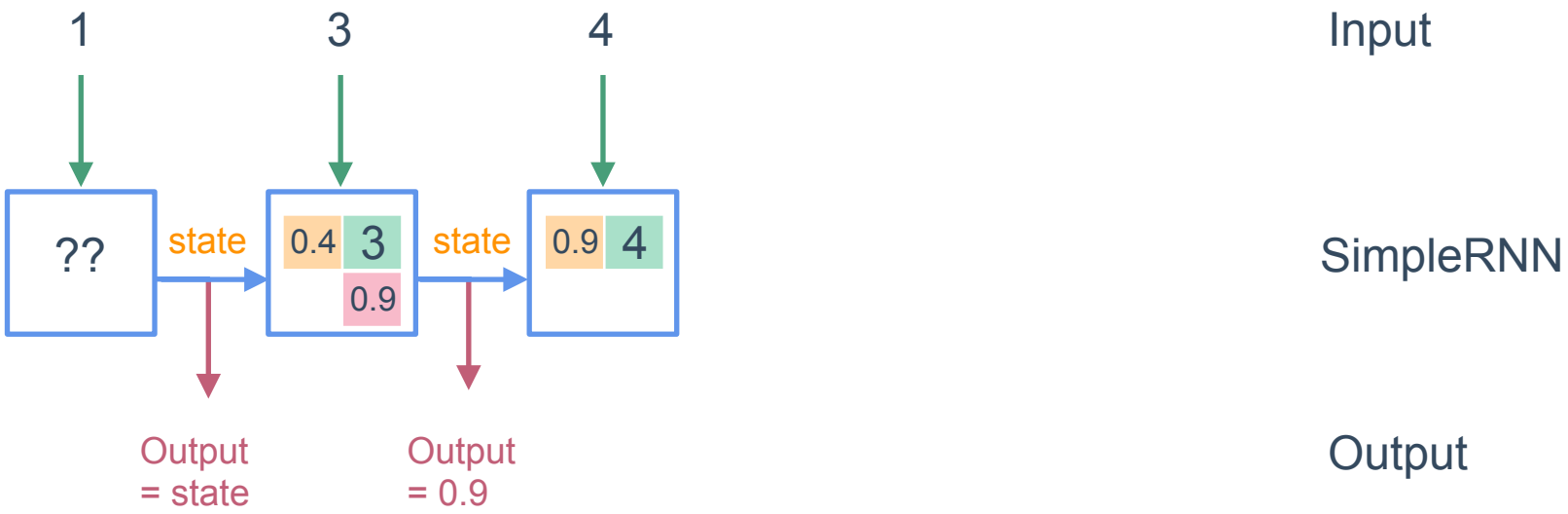



Input

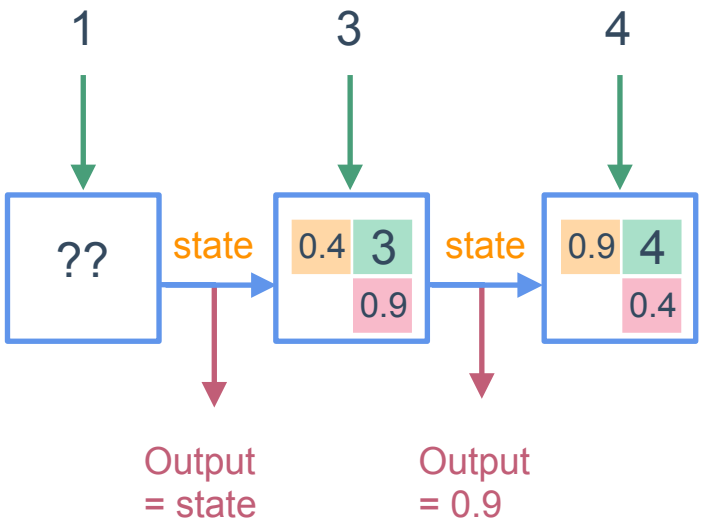
SimpleRNN

Output

# Recurrent Neural Network (RNN)



# Recurrent Neural Network (RNN)

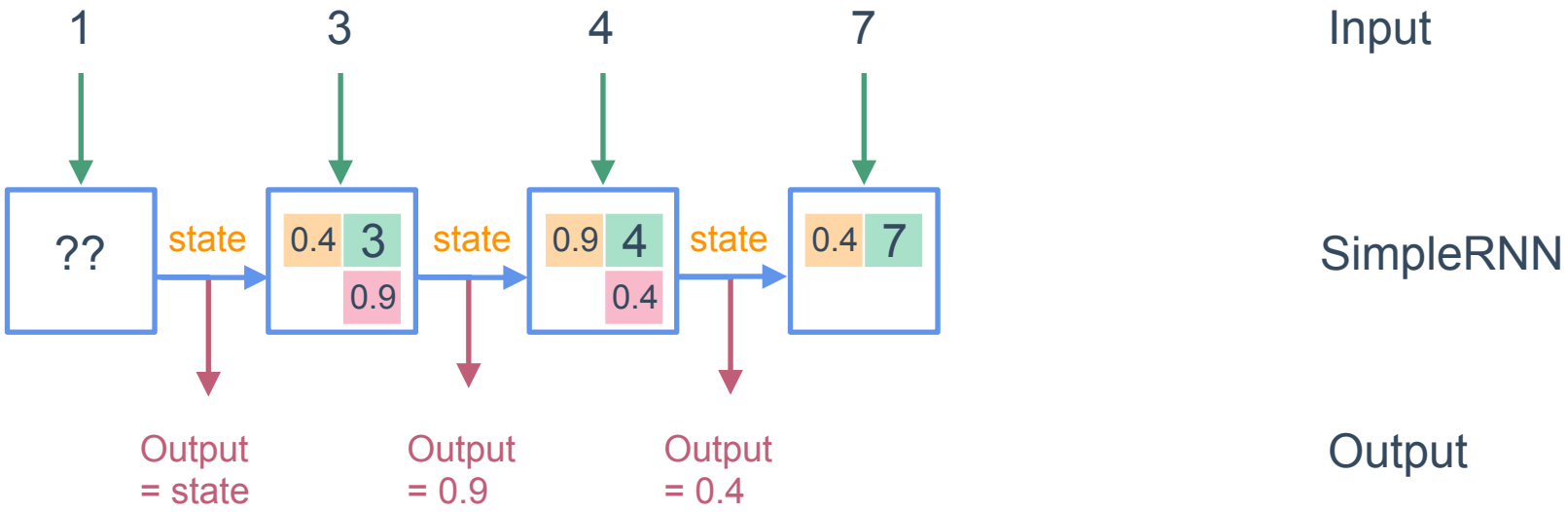



Input

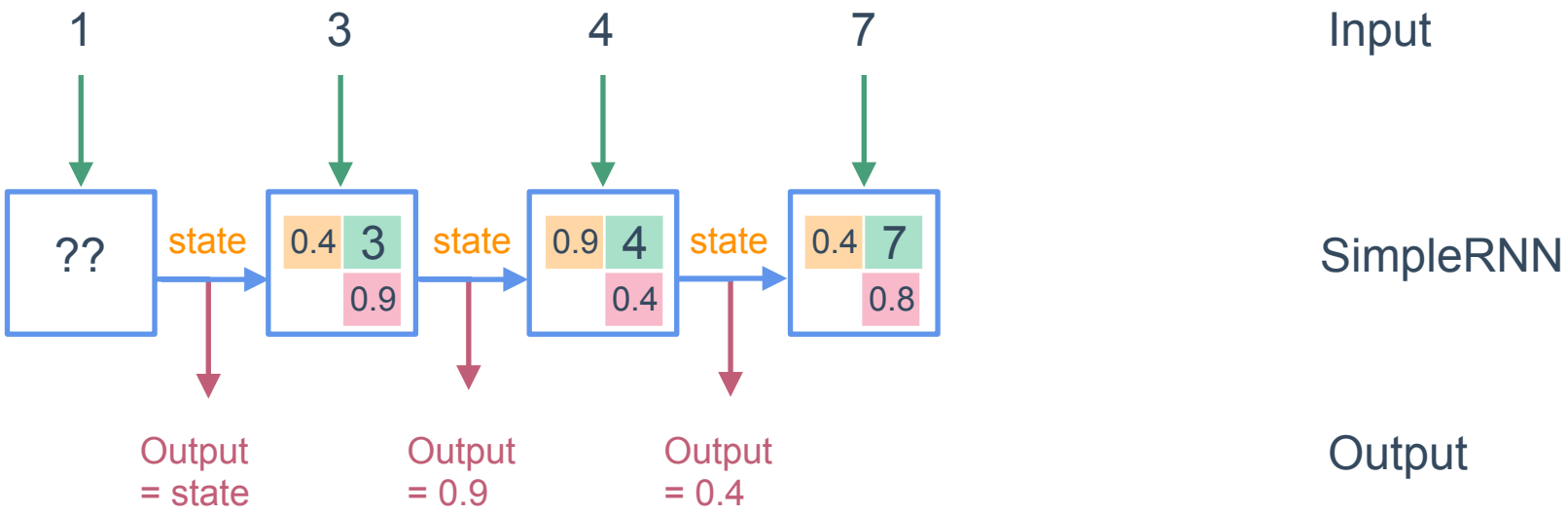
SimpleRNN

Output

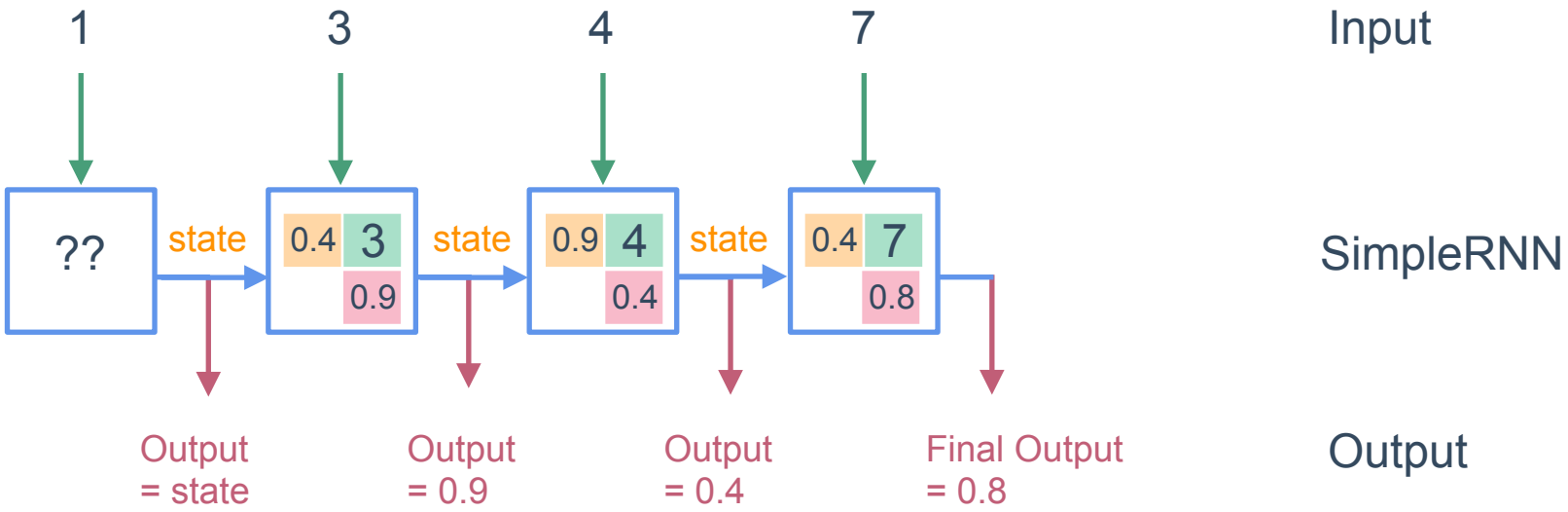
# Recurrent Neural Network (RNN)



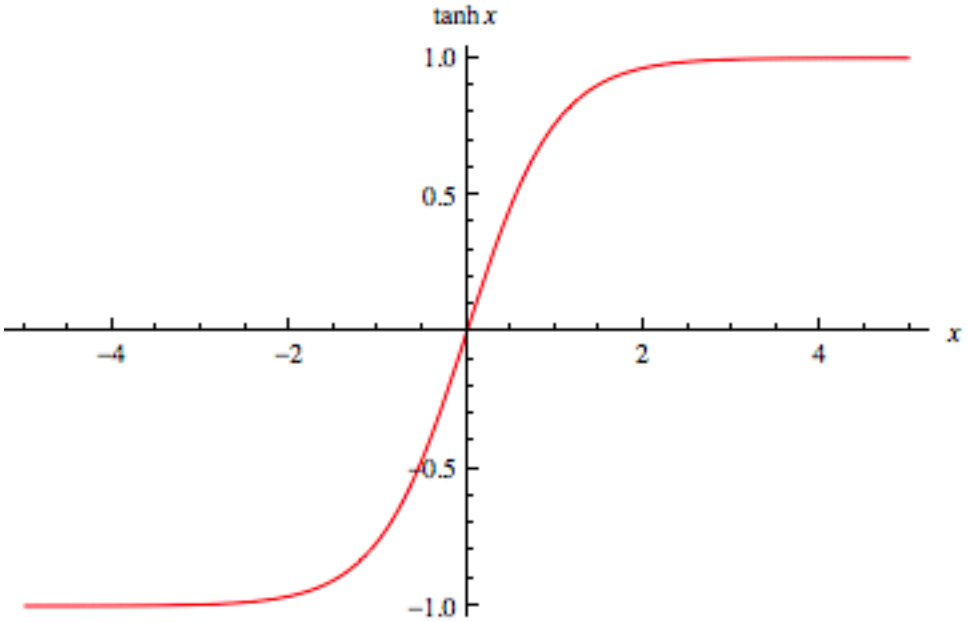
# Recurrent Neural Network (RNN)



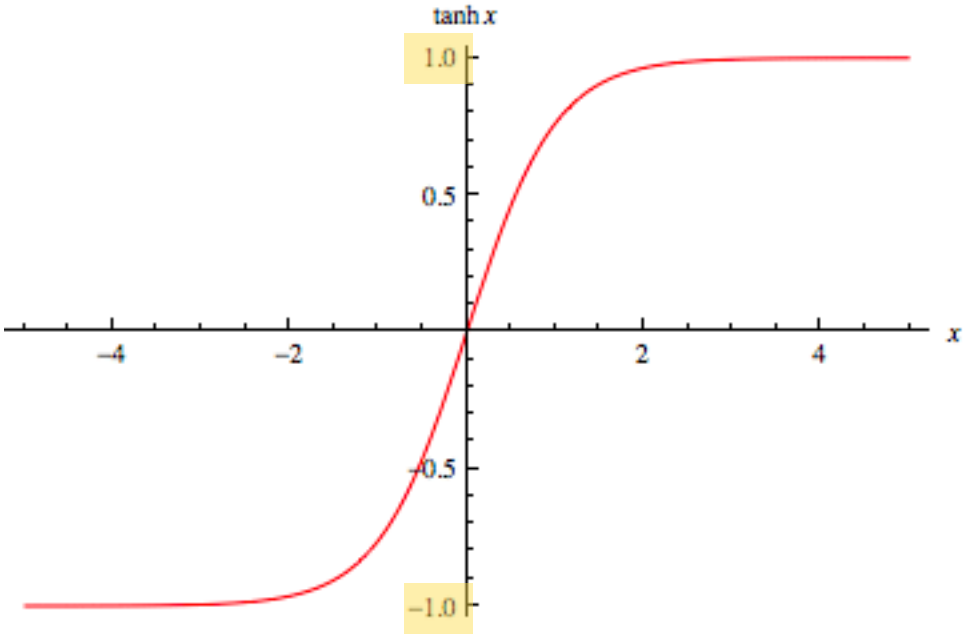
# Recurrent Neural Network (RNN)



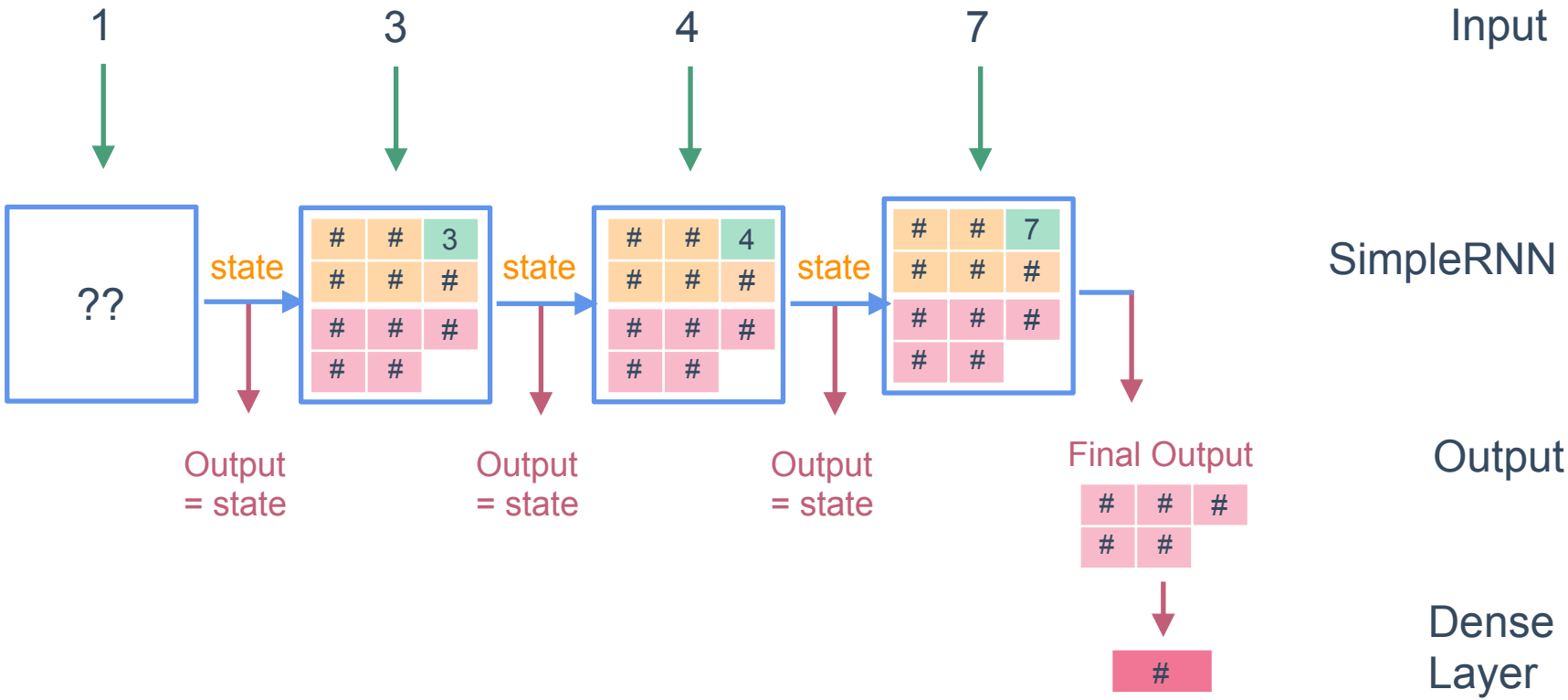
# Hyperbolic Tangent Activation Function (tanh)



# Hyperbolic Tangent Activation Function (tanh)



# Multidimensional State RNN



# LSTMs, GRUs & Text Generation
# ml-class/examples/lstm/text-gen

# Machine learning generated pranks

Put a pair of pants and shoes into your ice dispenser

Put marbles in the refrigerator

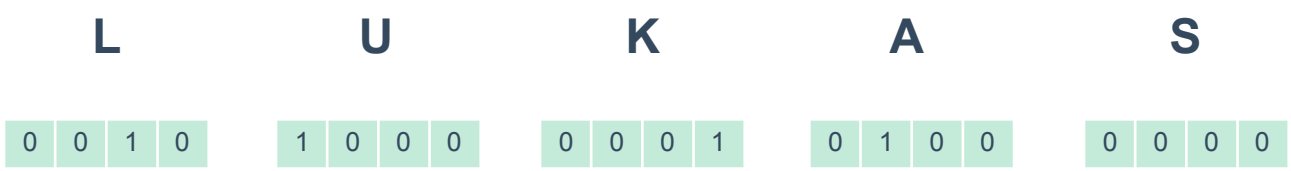
A meat and mash potato sundae makes for quite the hand soap dispenser

(Generated with an LSTM at aiweirdness.com)

# One-hot encoding

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| L → | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| U → | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| K → | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| A → | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| S → | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# Text as time series (character-level)




Input

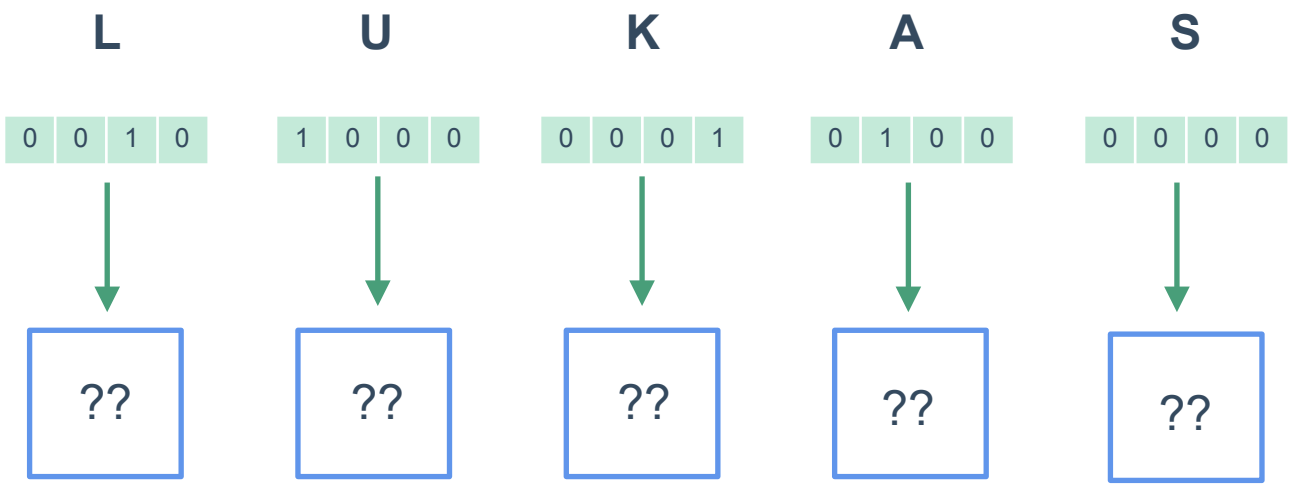
One-hot vector

SimpleRNN

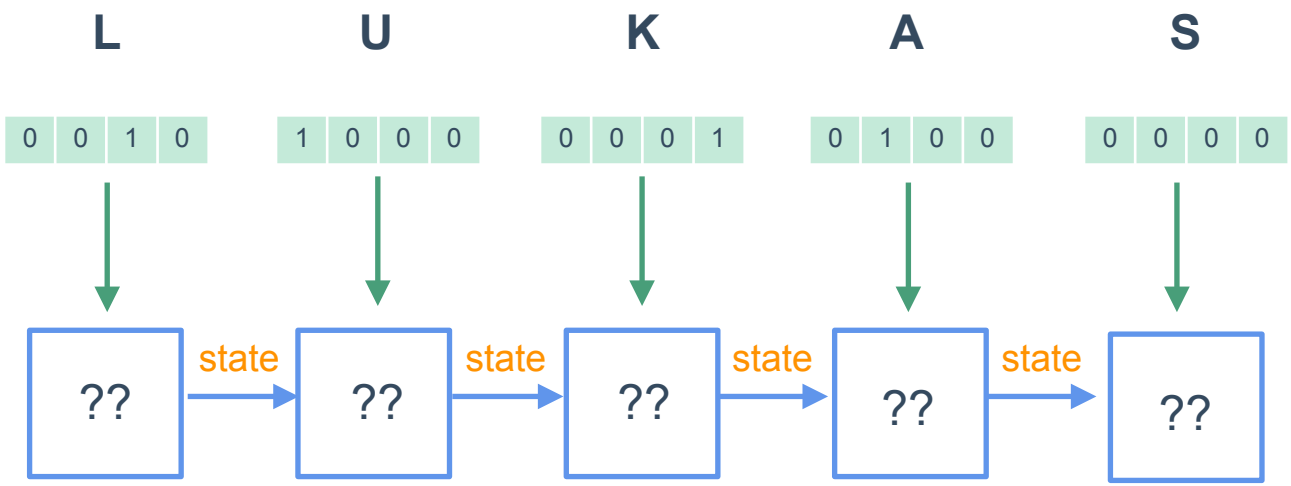
Encoded Output

Output next letter

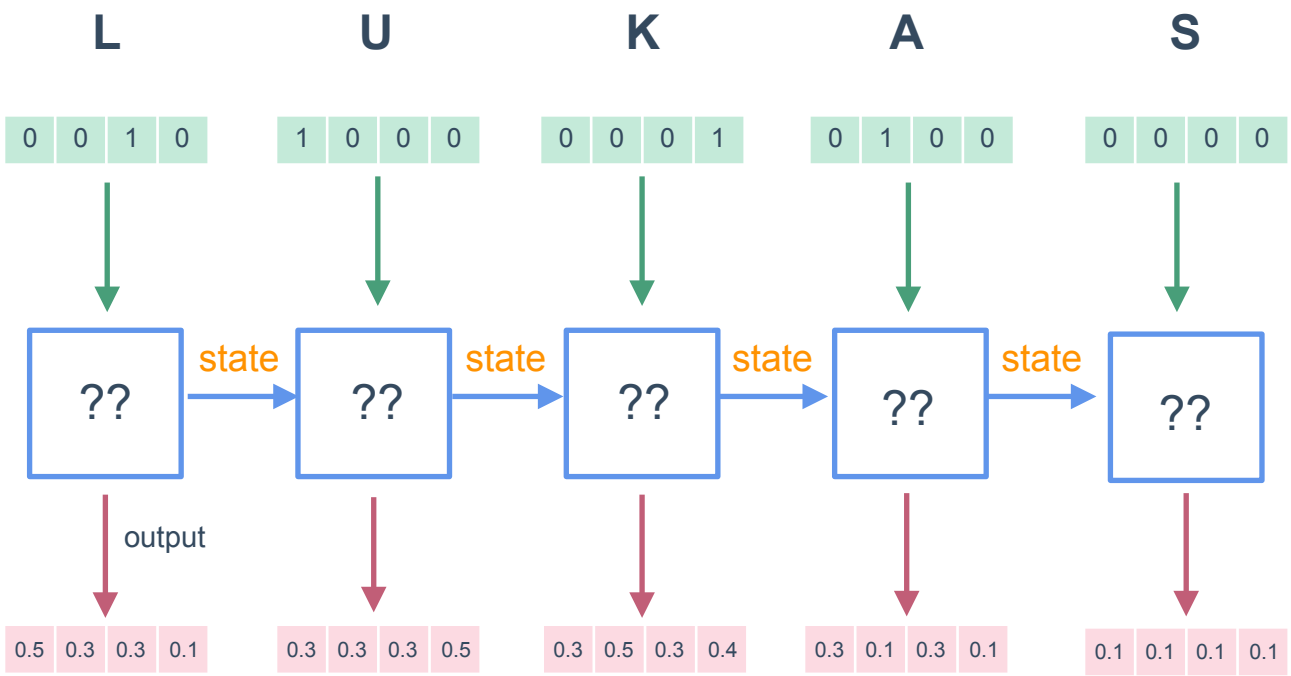
# Text as time series (character-level)



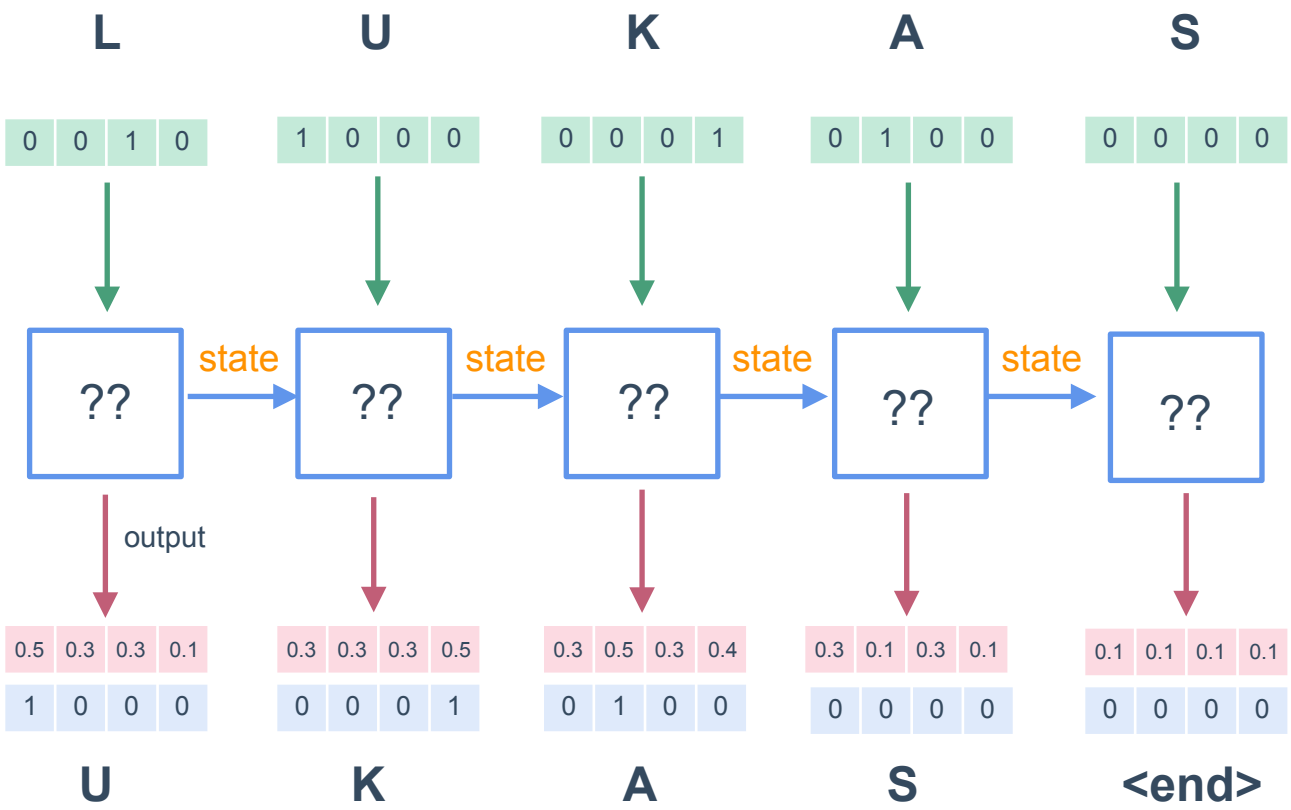
# Text as time series (character-level)



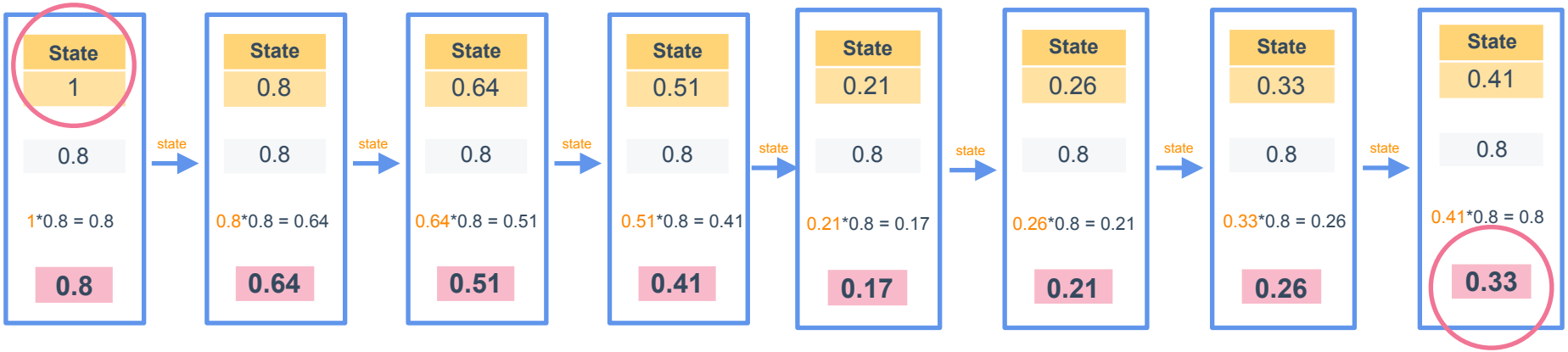
# Text as time series (character-level)



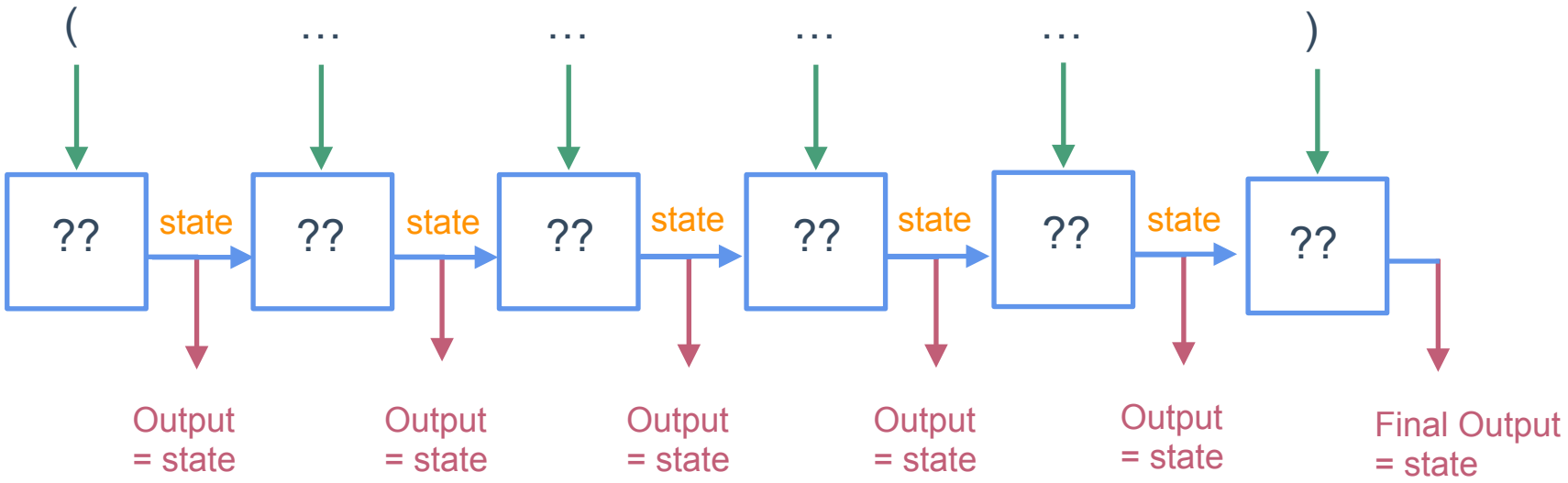
# Text as time series (character-level)



# The problem with simple RNNs



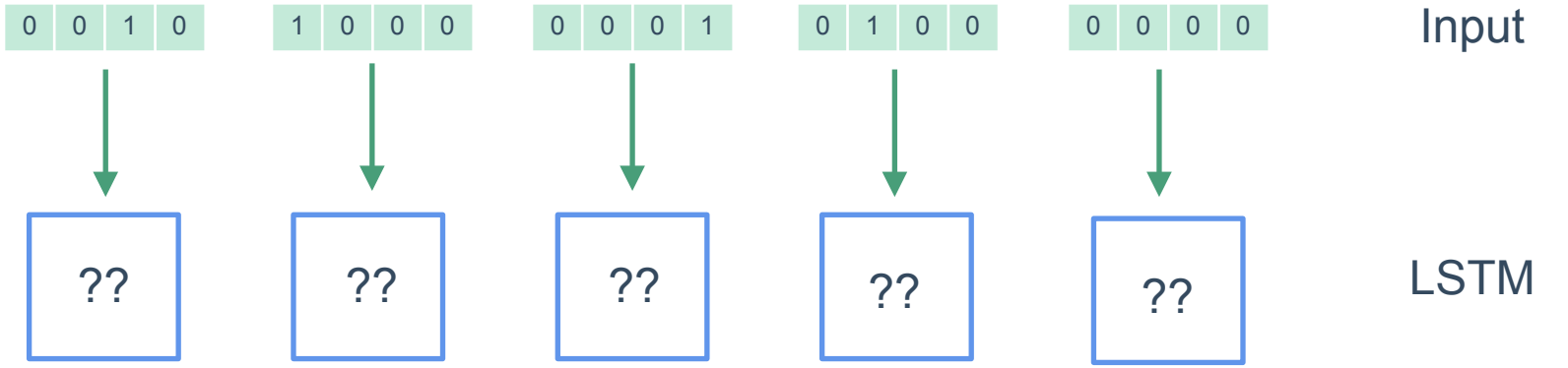
# The problem with simple RNNs



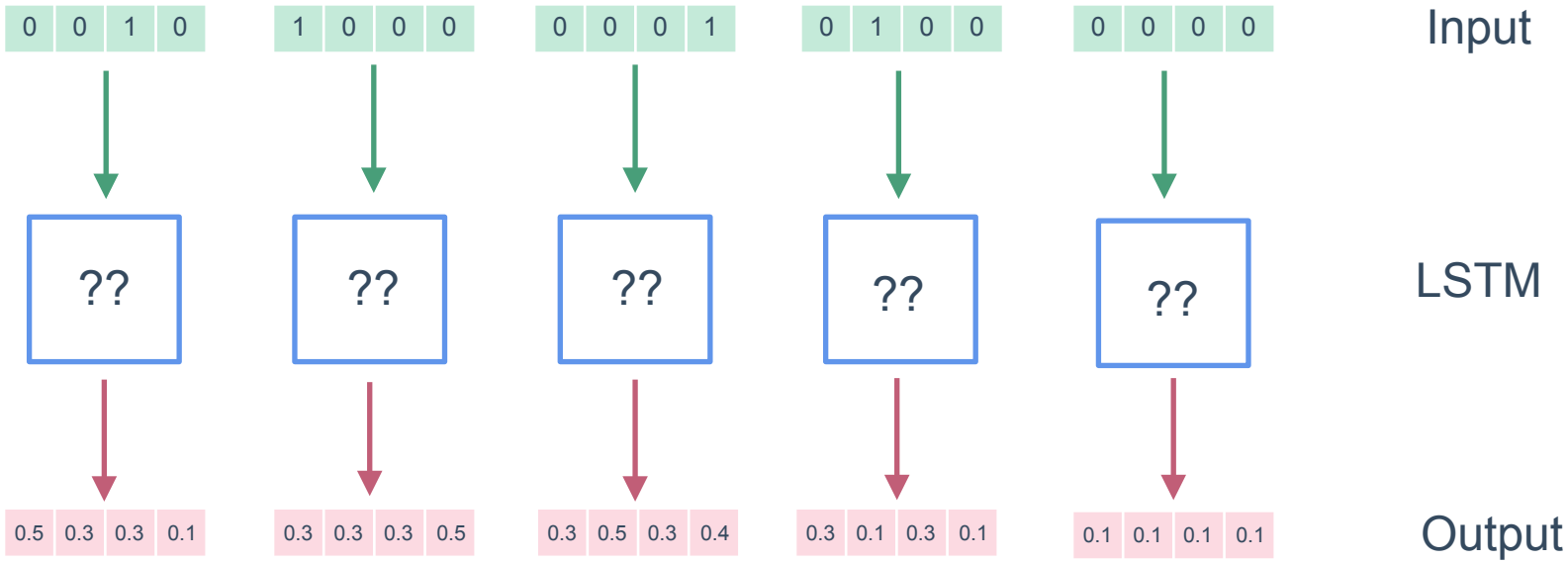
# Long Short Term Memory  (LSTM) 1997

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
$$

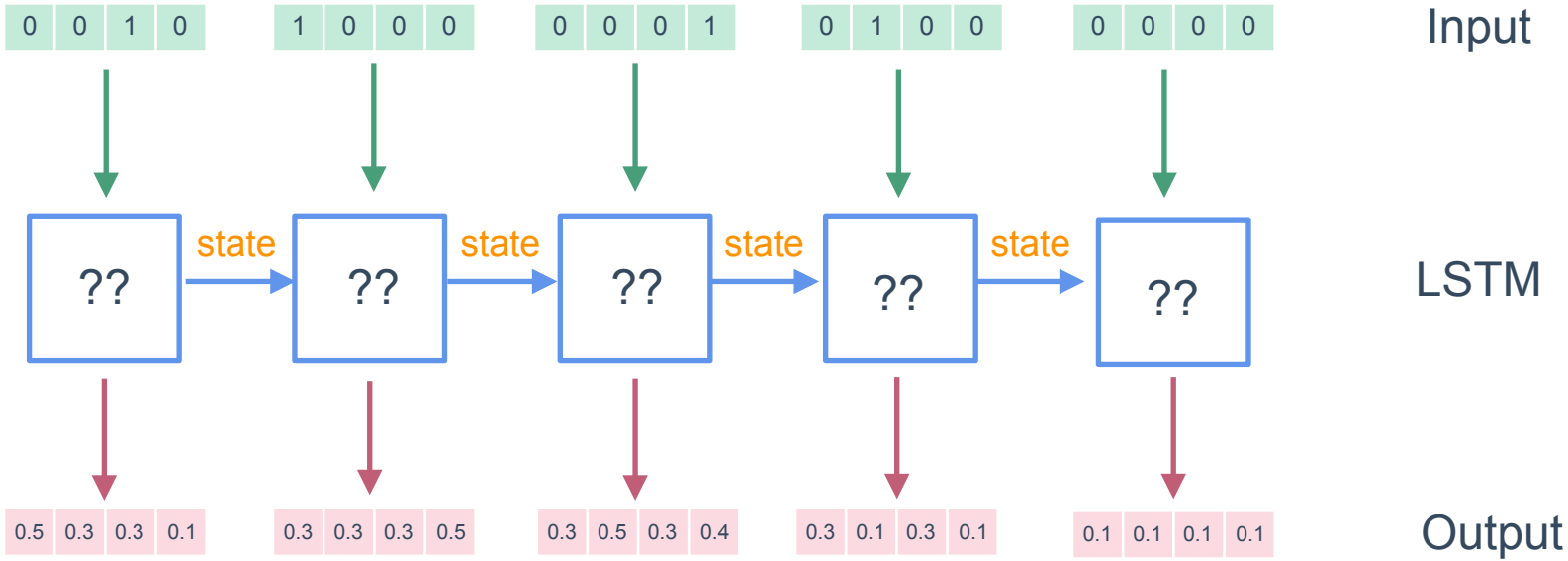
# LSTM



# LSTM


0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0
0 0 0 0
Input
??
??
??
??
??
LSTM
0.5 0.3 0.3 0.1
0.3 0.3 0.3 0.5
0.3 0.5 0.3 0.4
0.3 0.1 0.3 0.1
0.1 0.1 0.1 0.1
Output

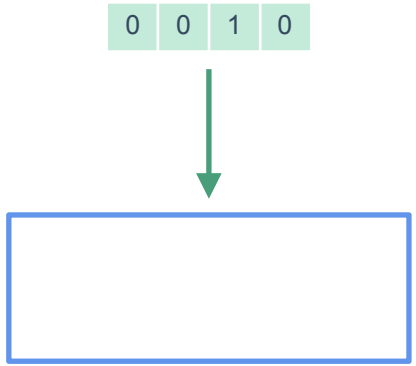
# LSTM



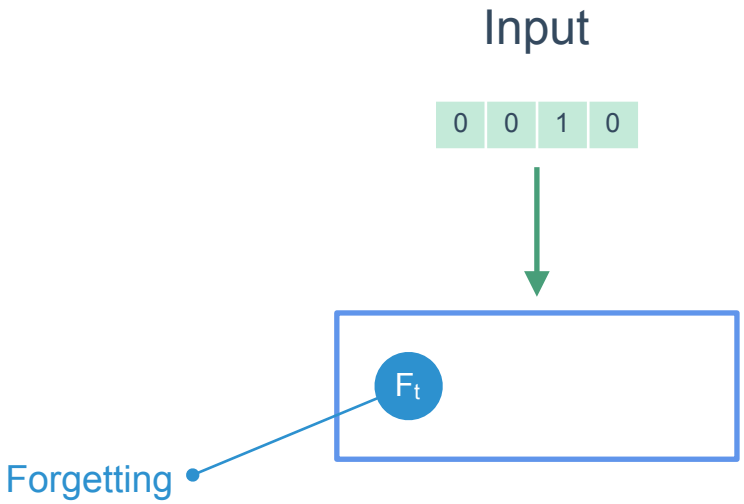

Output =/= state

# LSTM

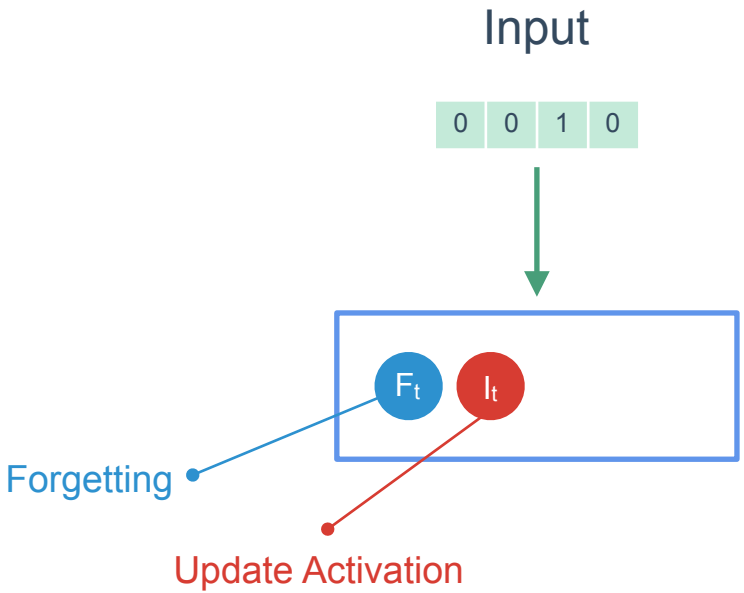
Input



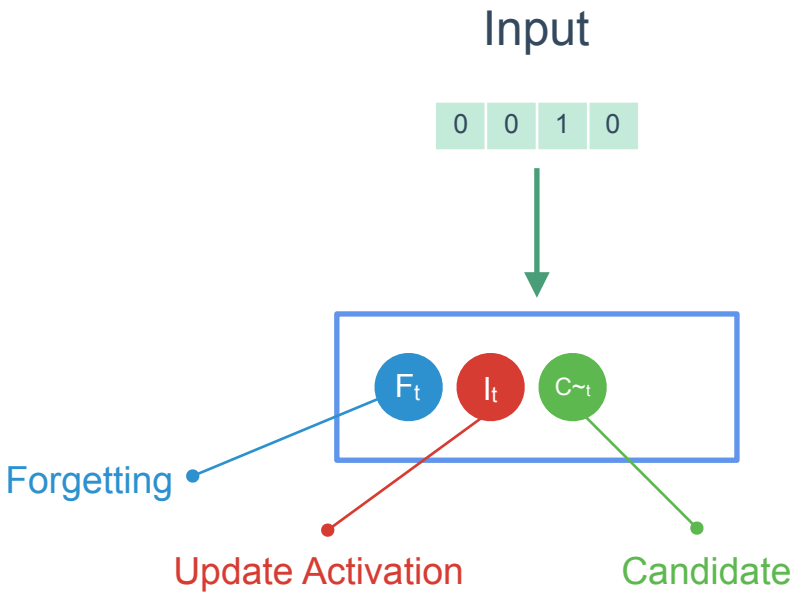
# LSTM



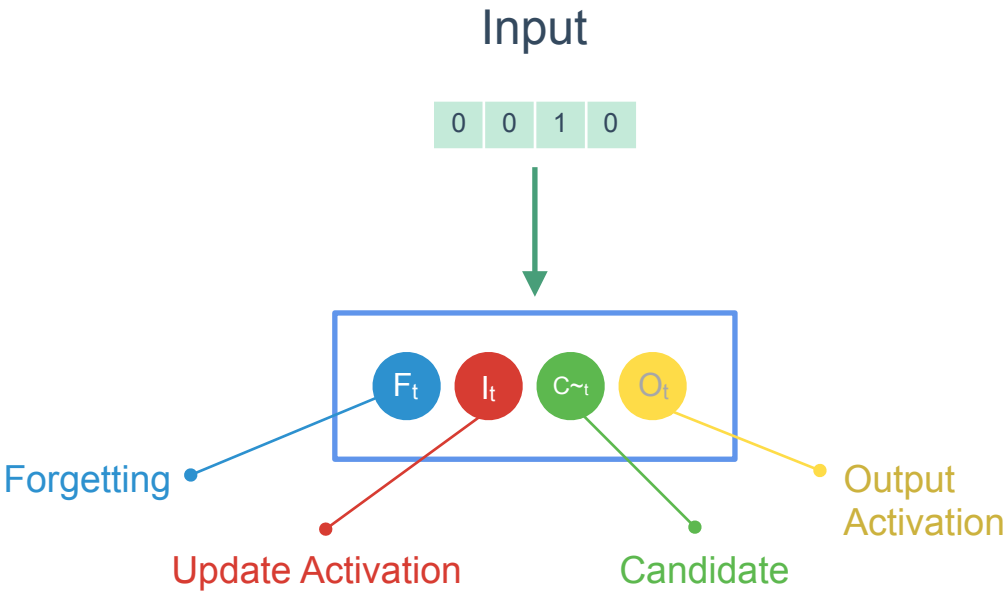
# LSTM



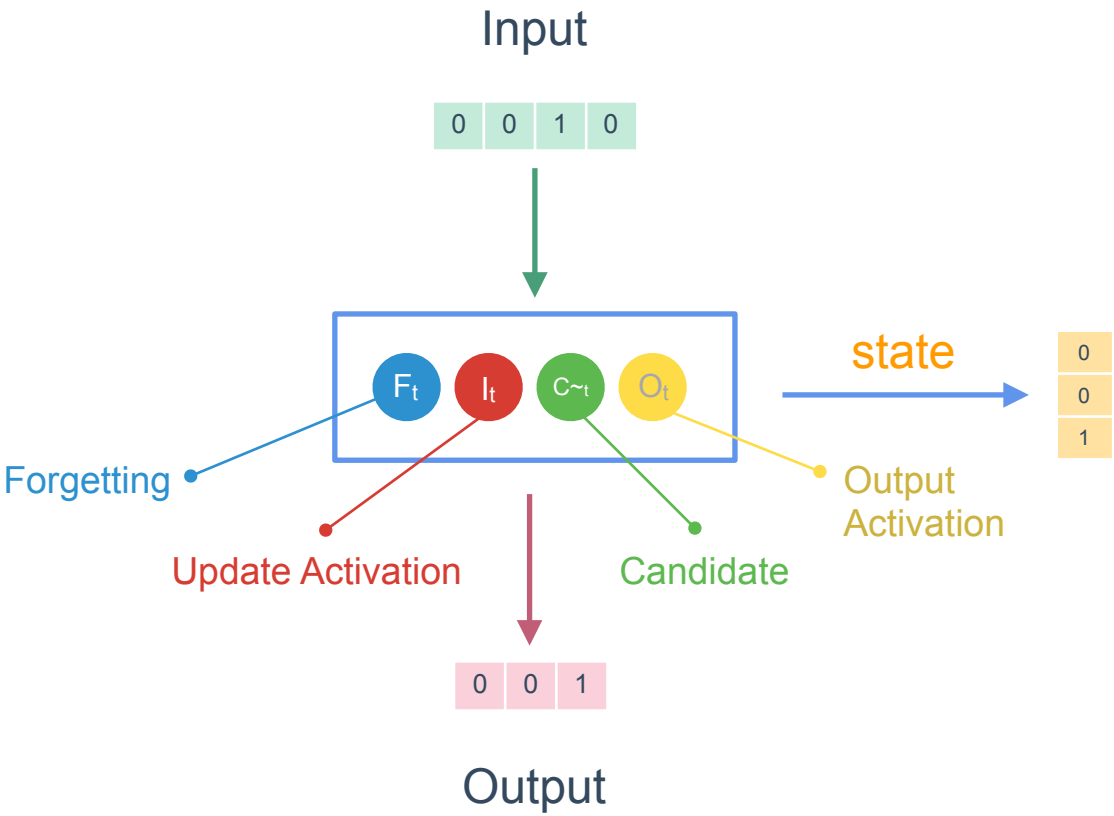
# LSTM


Input
0 0 1 0
$F_t$
$I_t$
$C\sim_t$
Forgetting
Update Activation
Candidate

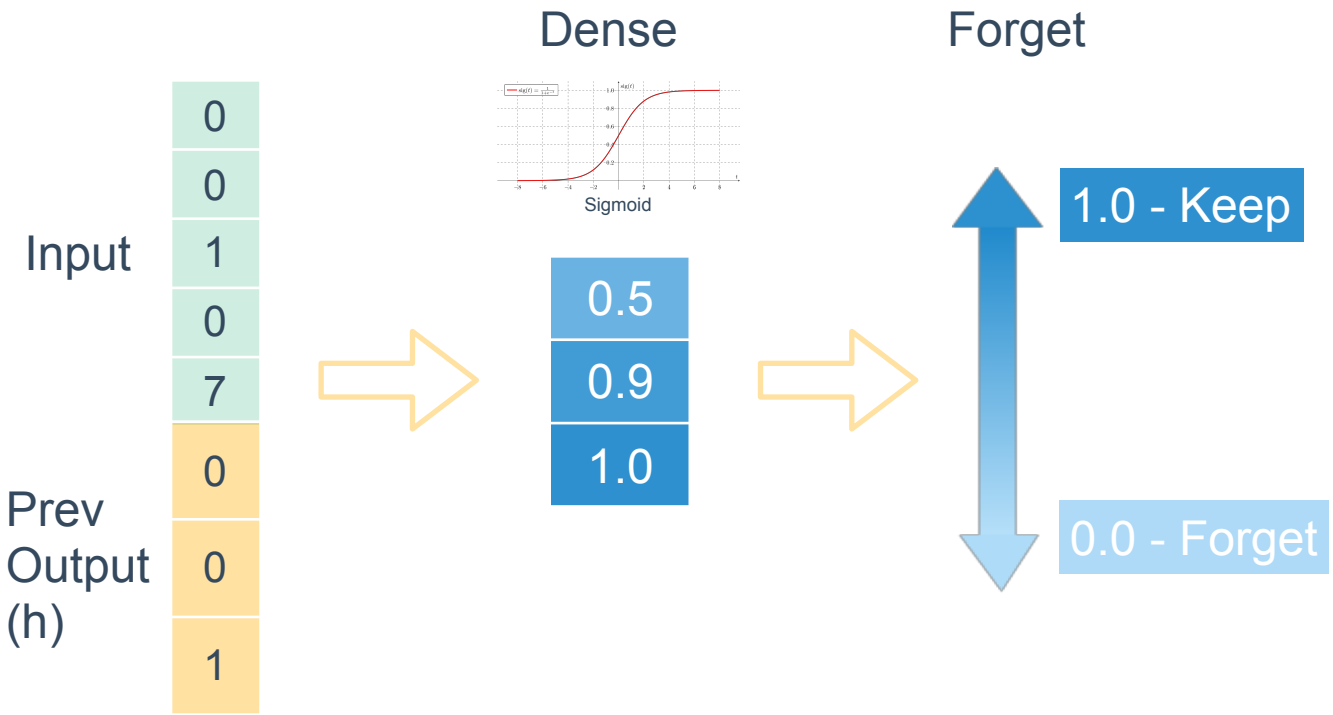
# LSTM


Input
0 0 1 0
$F_t$
$I_t$
$C\sim_t$
$O_t$
Forgetting
Update Activation
Candidate
Output Activation

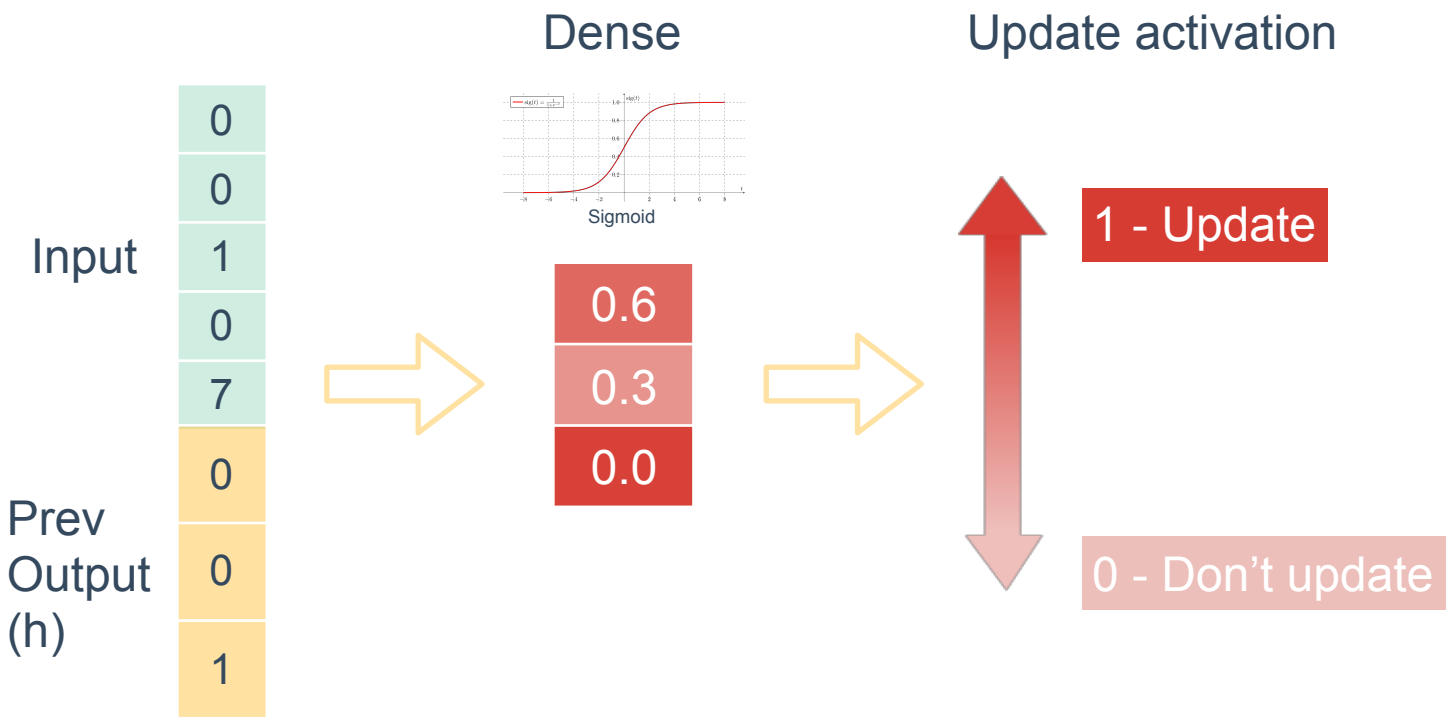
# LSTM


Input
0 0 1 0
$F_t$
$I_t$
$C\sim_t$
$O_t$
state
0
0
1
Forgetting
Update Activation
Candidate
Output Activation
0 0 1
Output

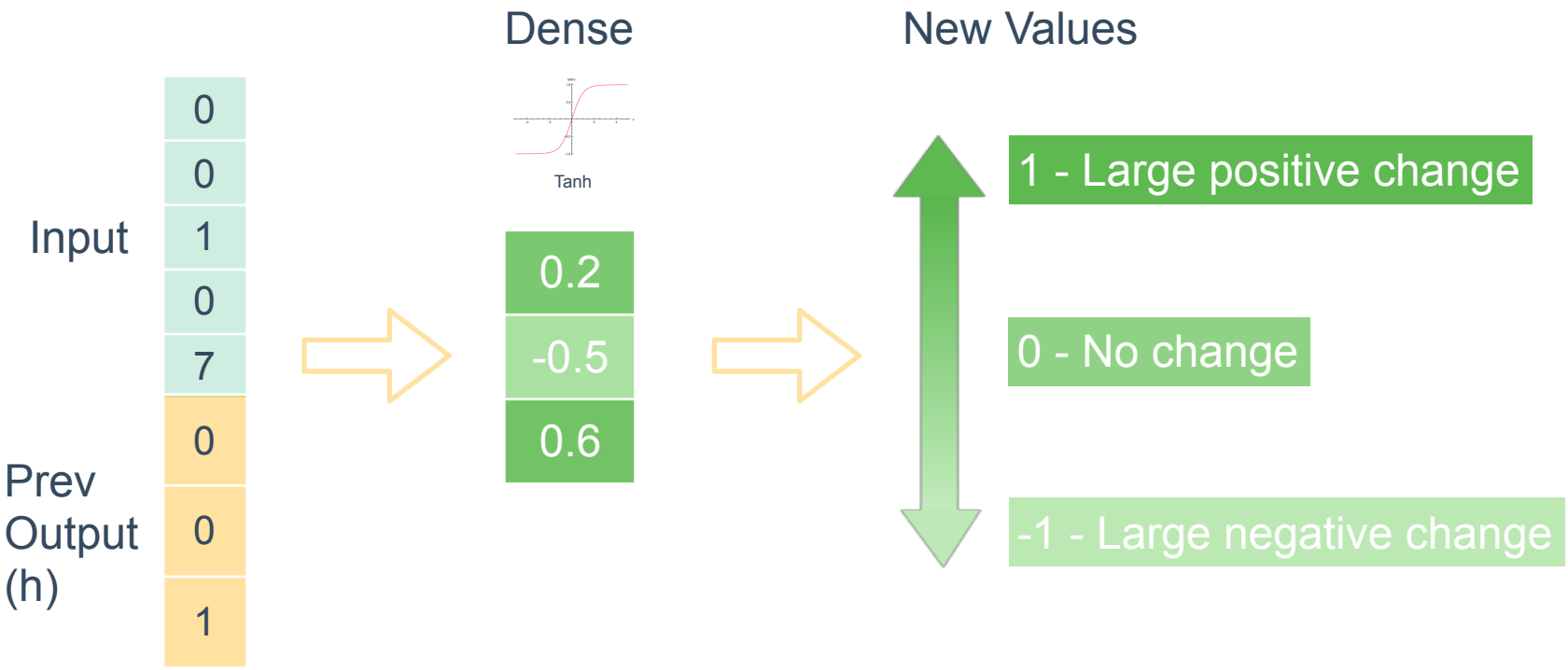
# LSTM Forgetting ($F_t$)



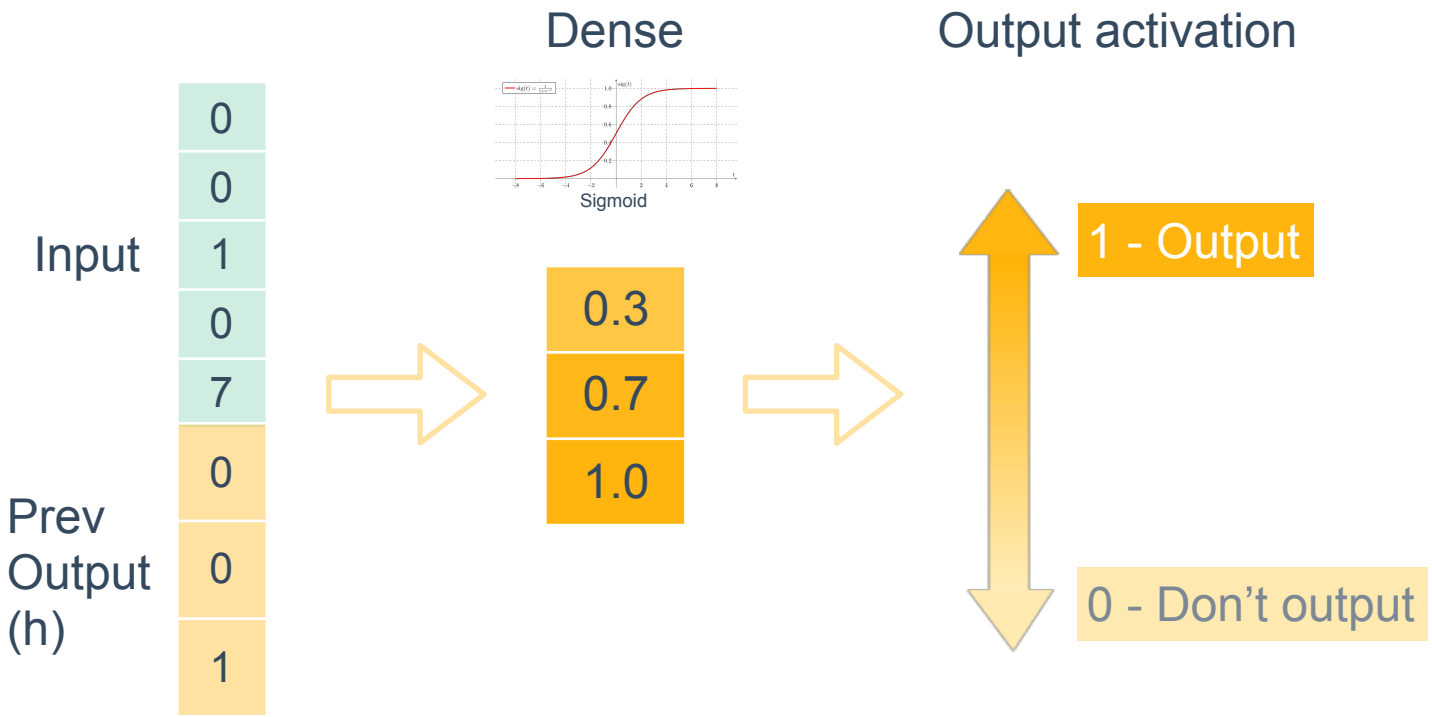
# LSTM Update Activation ($I_t$)



# LSTM Candidate ( $\tilde{C}_t$ )



# LSTM Output Activation ($O_t$)



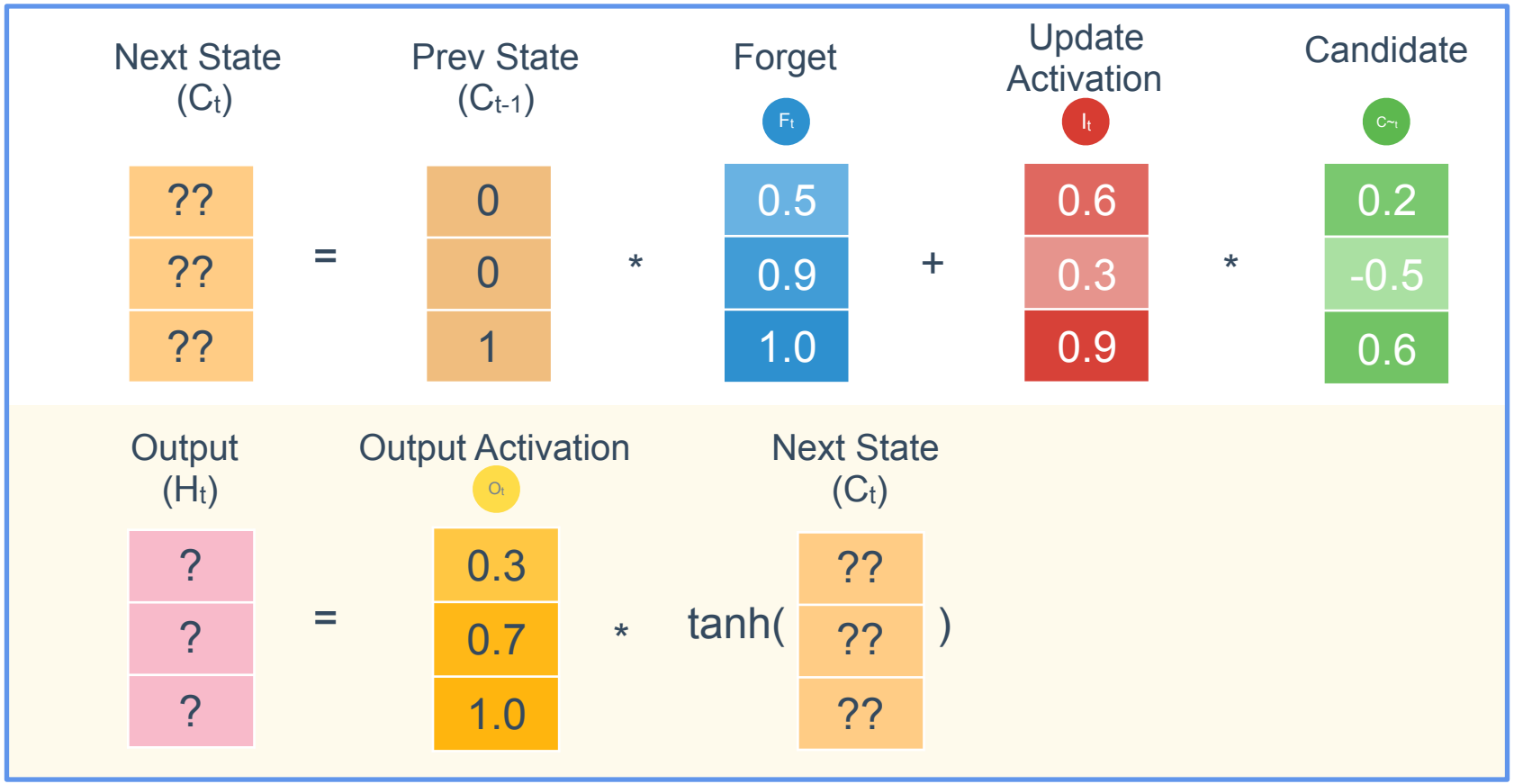

# Putting it all together

| Next State ($C_t$) | | Prev State ($C_{t-1}$) | | Forget $F_t$ |
|---|---|---|---|---|
| ?? | = | 0 | * | 0.5 |
| ?? | | 0 | | 0.9 |
| ?? | | 1 | | 1.0 |

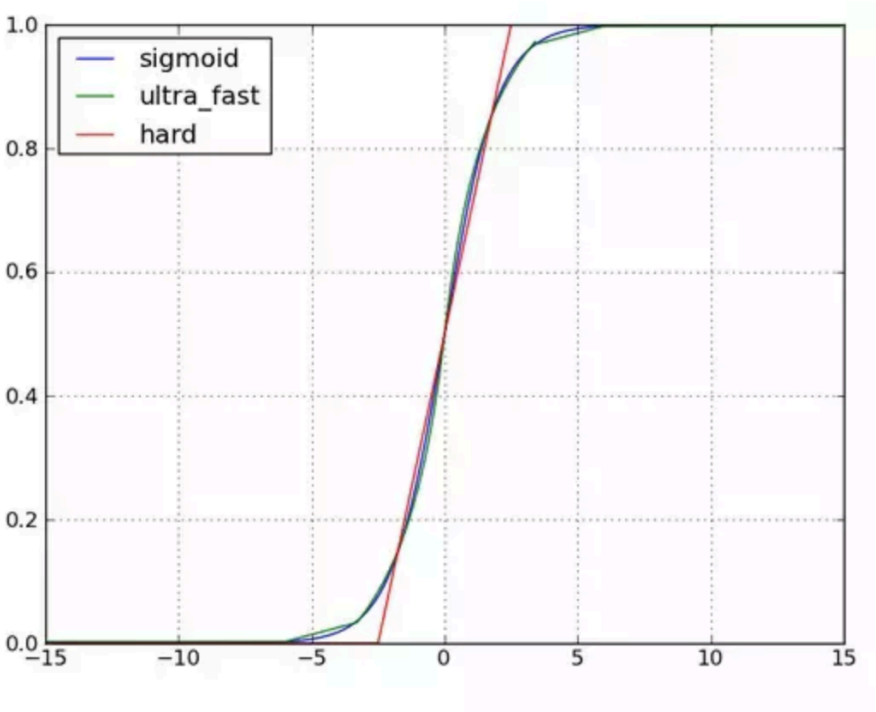

# Putting it all together

| Next State ($C_t$) | | Prev State ($C_{t-1}$) | | Forget $F_t$ | | Update Activation $I_t$ | | Candidate $C\sim_t$ |
|---|---|---|---|---|---|---|---|---|
| ?? | = | 0 | * | 0.5 | + | 0.6 | * | 0.2 |
| ?? | | 0 | | 0.9 | | 0.3 | | -0.5 |
| ?? | | 1 | | 1.0 | | 0.0 | | 0.6 |

# Putting it all together



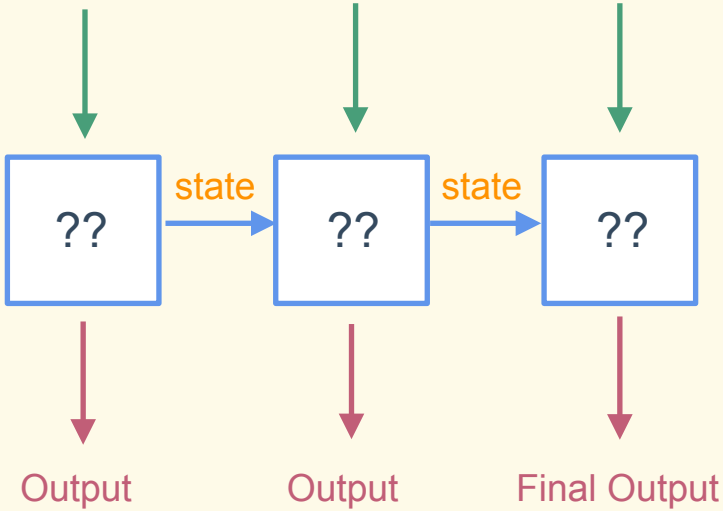
# Hard Sigmoid



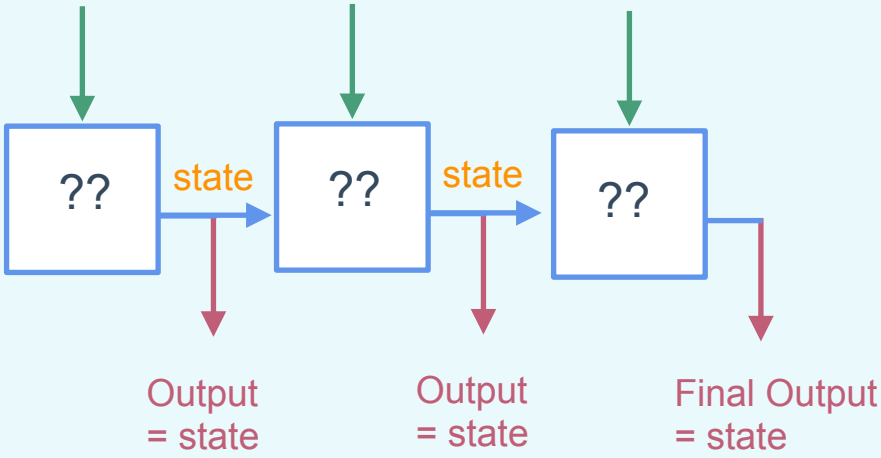
# Gated Recurrent Unit (GRU) 2014

$$
\begin{aligned}
z_t &= \sigma\left(x_t U^z + h_{t-1} W^z\right) \\
r_t &= \sigma\left(x_t U^r + h_{t-1} W^r\right) \\
\tilde{h}_t &= \tanh\left(x_t U^h + (r_t * h_{t-1}) W^h\right) \\
h_t &= (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t
\end{aligned}
$$

# LSTMs vs GRUs (1)

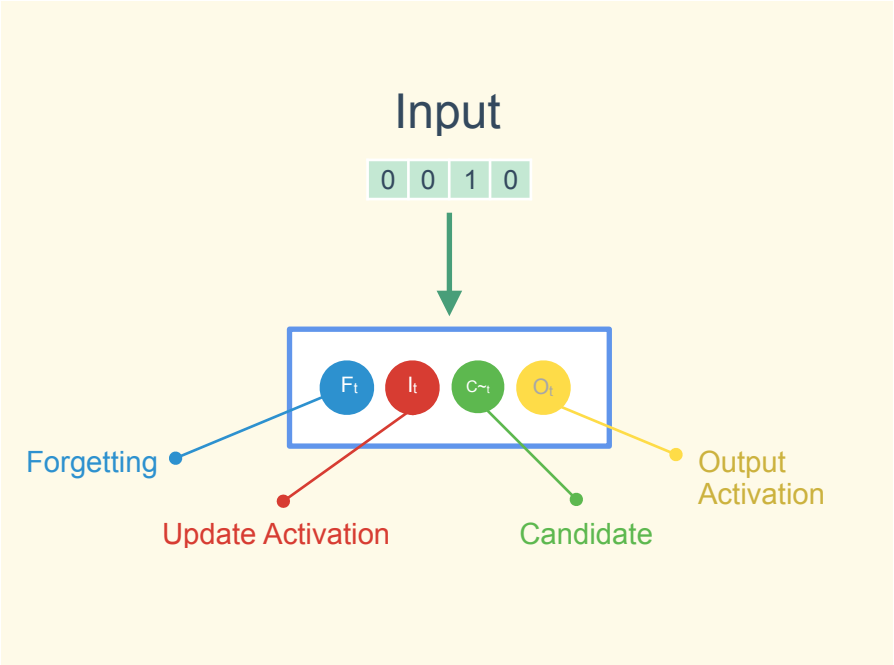



**LSTM**: State =/= Output

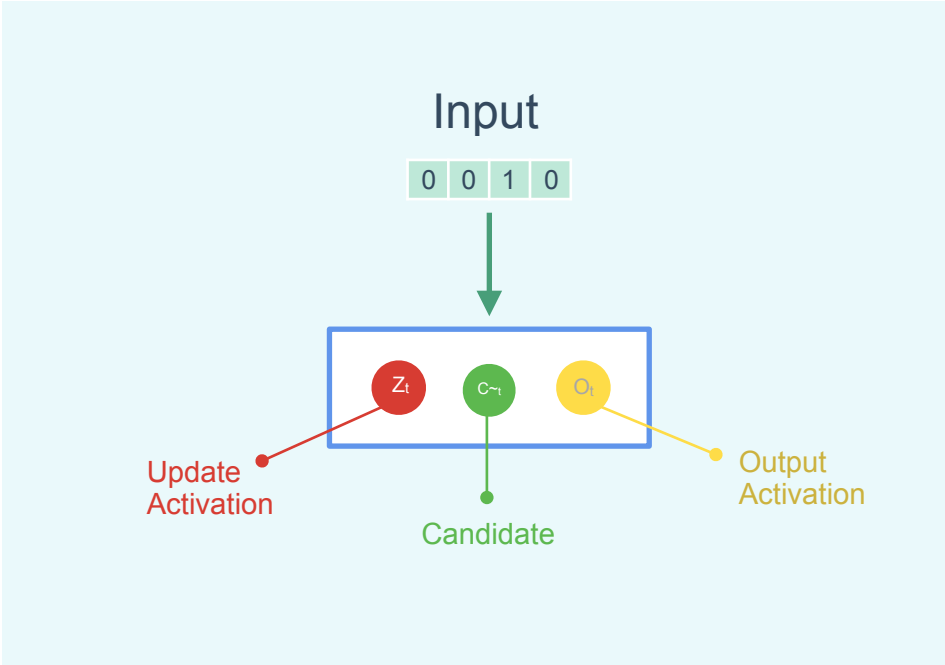



**GRU**: State == Output

# LSTMs vs GRUs (2)




**LSTM**: 4 vectors




**GRU**: 3 vectors

# Text Classification, Embeddings and 1D Convolutions

ml-class/examples/lstm/imdb-classifier

# Character encoding

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| L → | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| U → | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| K → | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| A → | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| S → | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

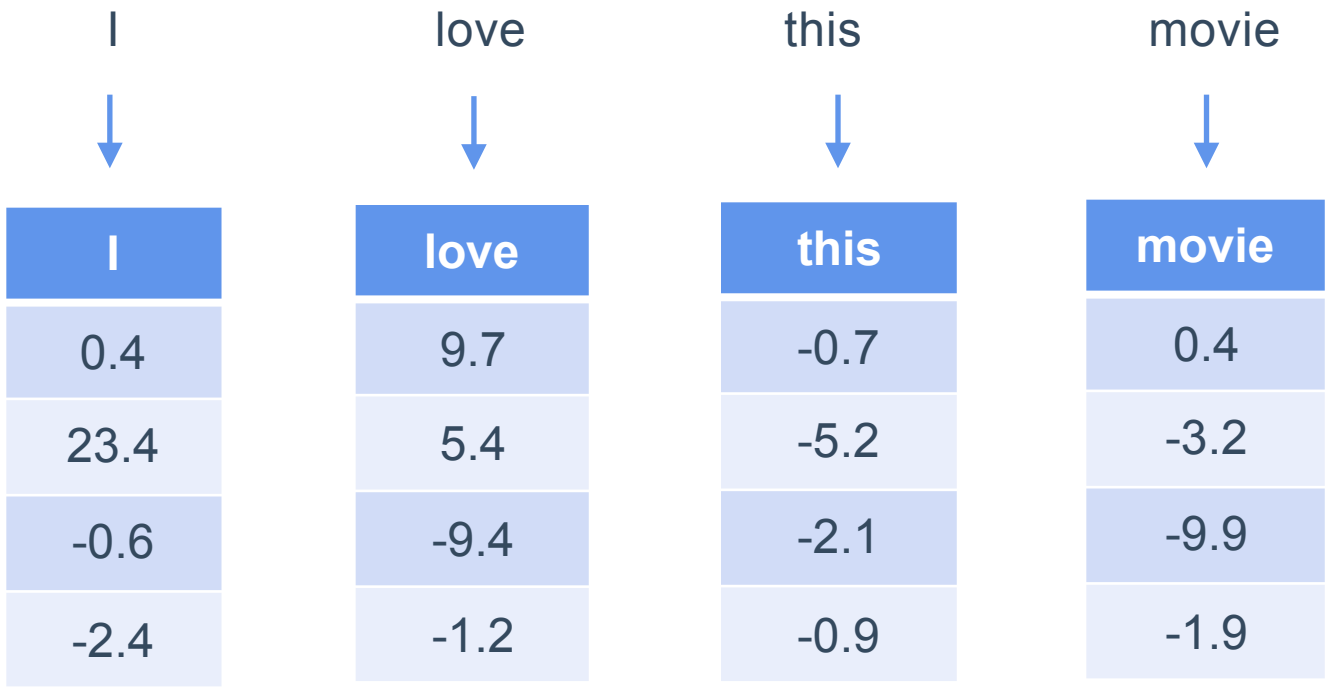
# Bag of Words

Input Text

"Bag of Words"

| | a | … | hate | I | iPhone | love | my | … | zoo |
|---|---|---|---|---|---|---|---|---|---|
| I love my iPhone → | 0 | … | 0 | 1 | 1 | 1 | 1 | … | 0 |
| I hate my iPhone → | 0 | … | 1 | 1 | 1 | 0 | 1 | … | 0 |

# Word Embedding



# Embeddings

| | Val 1 | Val 2 | Val 3 | Val 4 |
|---|---|---|---|---|
| **a** | 0.1 | -0.3 | 1.7 | 2.4 |
| **aardvark** | -2.3 | 4.1 | -5.2 | 3.1 |
| **…** | … | … | … | … |
| **<unknown>** | 0.3 | 0.9 | 0.8 | 0.2 |

# Pre-computing encoding

## GloVe: Global Vectors for Word Representation

Jeffrey Pennington, Richard Socher, Christopher D. Manning

### Introduction

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

### Getting started (Code download)

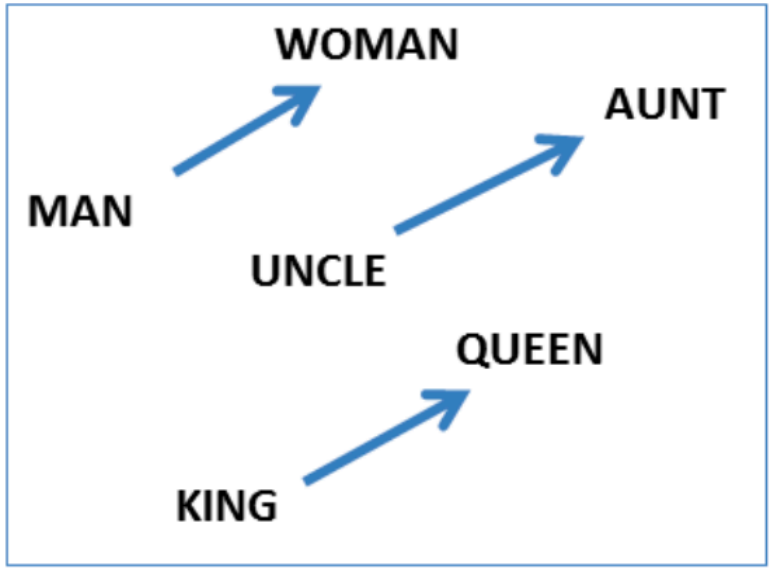
- Download the code (licensed under the Apache License, Version 2.0)
- Unpack the files: unzip GloVe-1.2.zip
- Compile the source: cd GloVe-1.2 && make
- Run the demo script: ./demo.sh
- Consult the included README for further usage details, or ask a question
- The code is also available on GitHub

### Download pre-trained word vectors

- Pre-trained word vectors. This data is made available under the Public Domain Dedication and License v1.0 whose full text can be found at: http://www.opendatacommons.org/licenses/pddl/1.0/.
  - Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocab, uncased, 50d, 100d, 200d, & 300d vectors, 822 MB download): glove.6B.zip
  - Common Crawl (42B tokens, 1.9M vocab, uncased, 300d vectors, 1.75 GB download): glove.42B.300d.zip
  - Common Crawl (840B tokens, 2.2M vocab, cased, 300d vectors, 2.03 GB download): glove.840B.300d.zip
  - Twitter (2B tweets, 27B tokens, 1.2M vocab, uncased, 25d, 50d, 100d, & 200d vectors, 1.42 GB download): glove.twitter.27B.zip
- Ruby script for preprocessing Twitter data

keras-imdb/imdb-embedding.py

# GloVe + word2vec




## Word Analogy Task

*man* is to *woman* as *king* is to ___ ?

*good* is to *best* as *smart* is to ___ ?

*china* is to *beijing* as *russia* is to ___ ?

Turns out the word-context based vector model we just learnt is good for such analogy tasks,

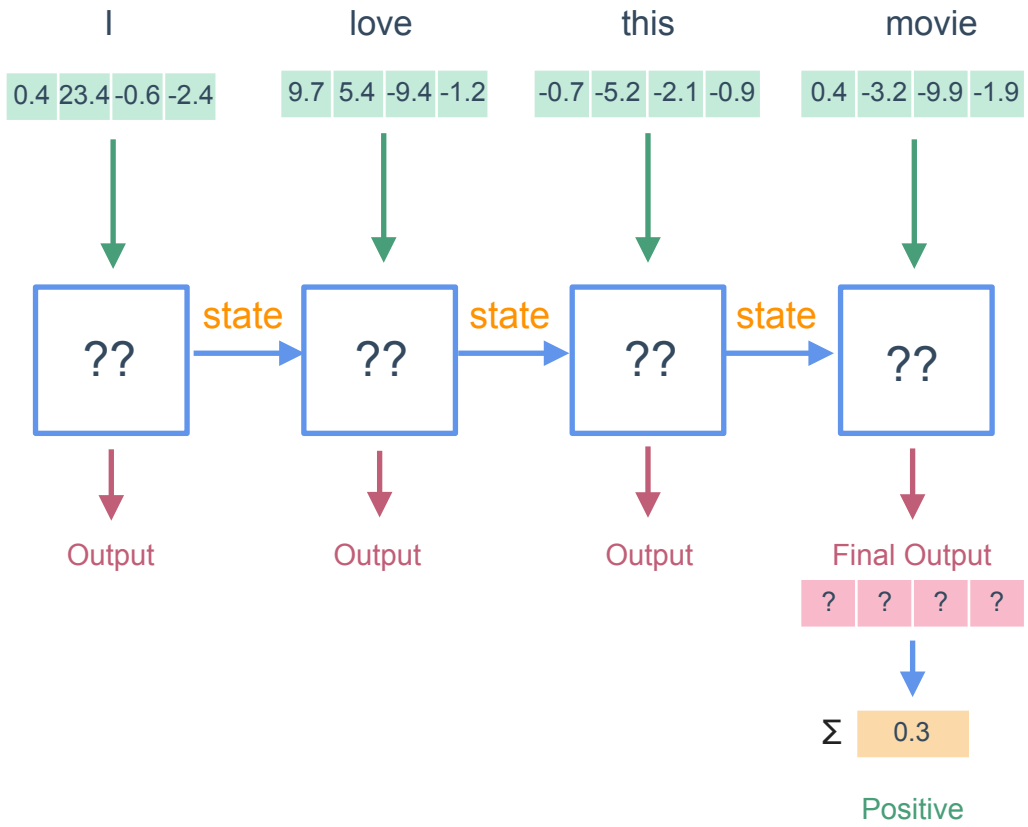
[king] – [man] + [woman] ≈ [queen]

Microsoft Levy, Goldberg, and Israel, *Linguistic Regularities in Sparse and Explicit Word Representations*, CoNLL 2014.

keras-imdb/imdb-embedding.py

# Classification with LSTMs

# Classification LSTM




Input

Word Embeddings

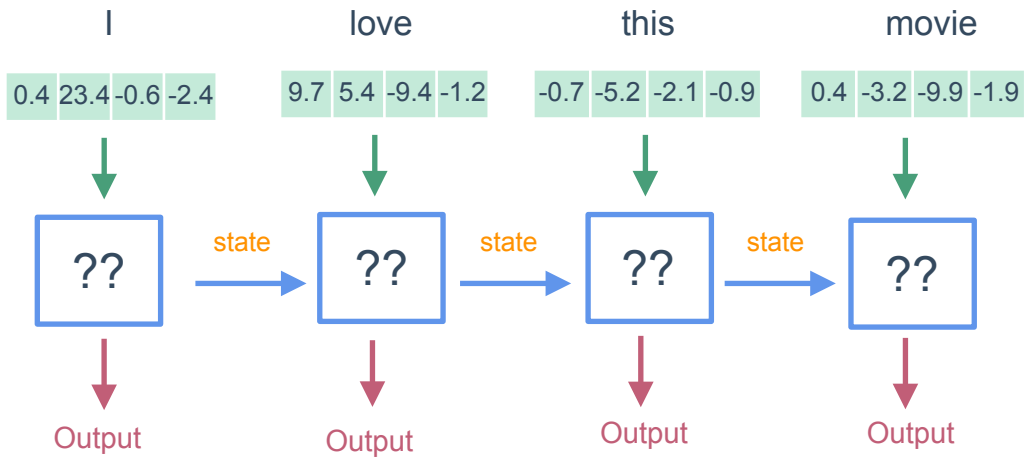
LSTM

Encoded Output

Dense layer

Classification

# Deep LSTM

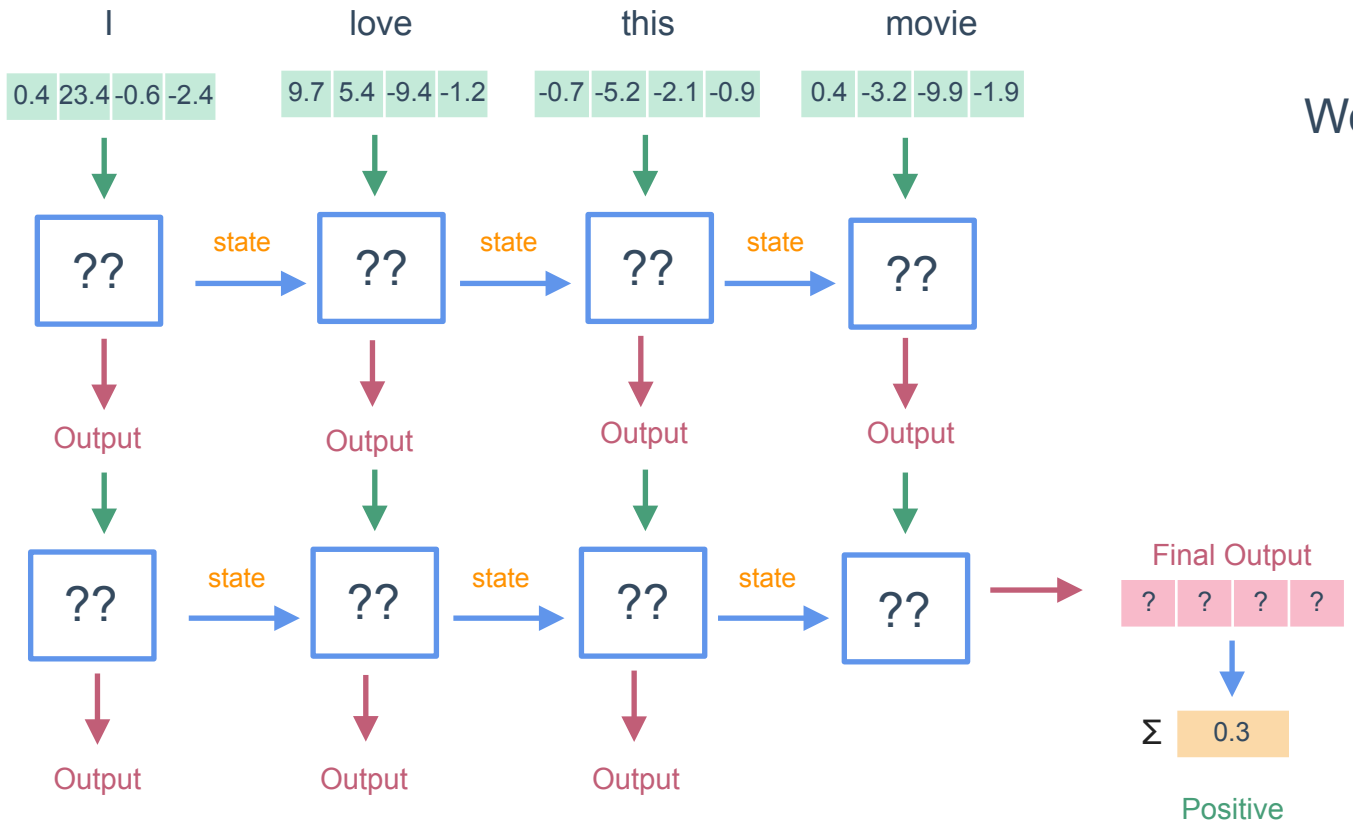



Input

Word Embeddings

LSTM

# Deep LSTM




Input

Word Embeddings

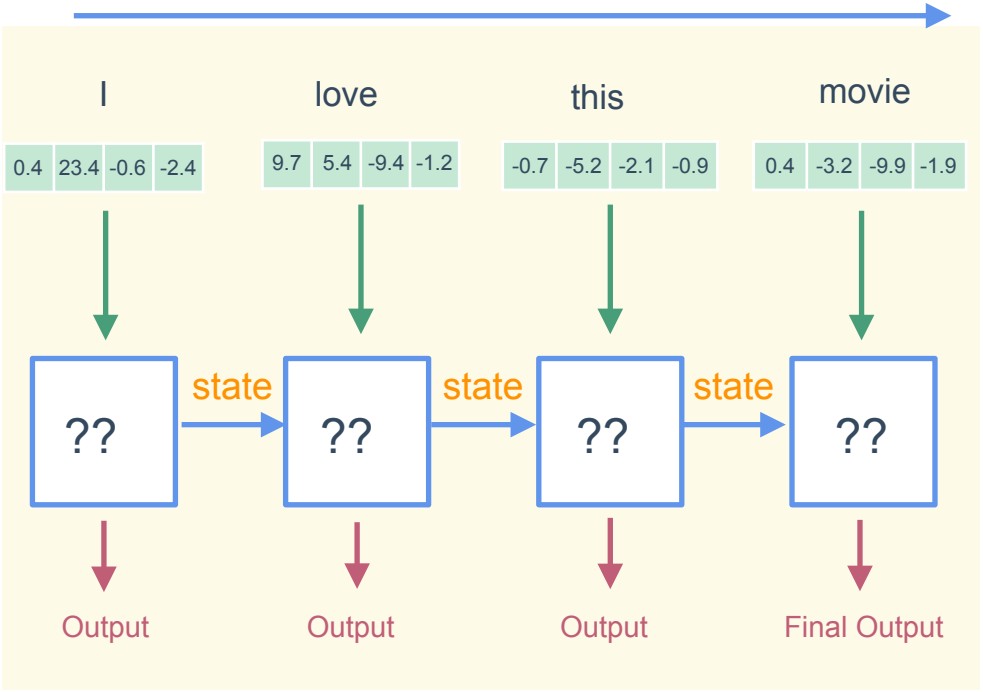
LSTM

LSTM

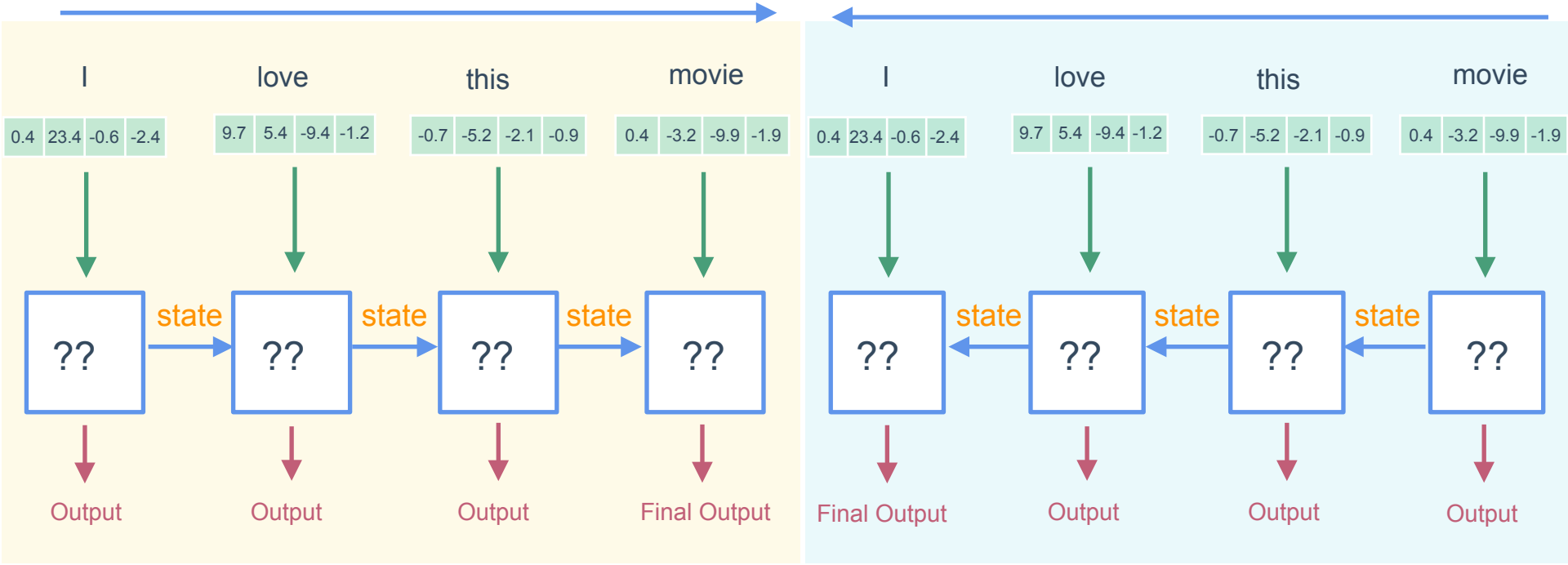
Dense layer

Classification

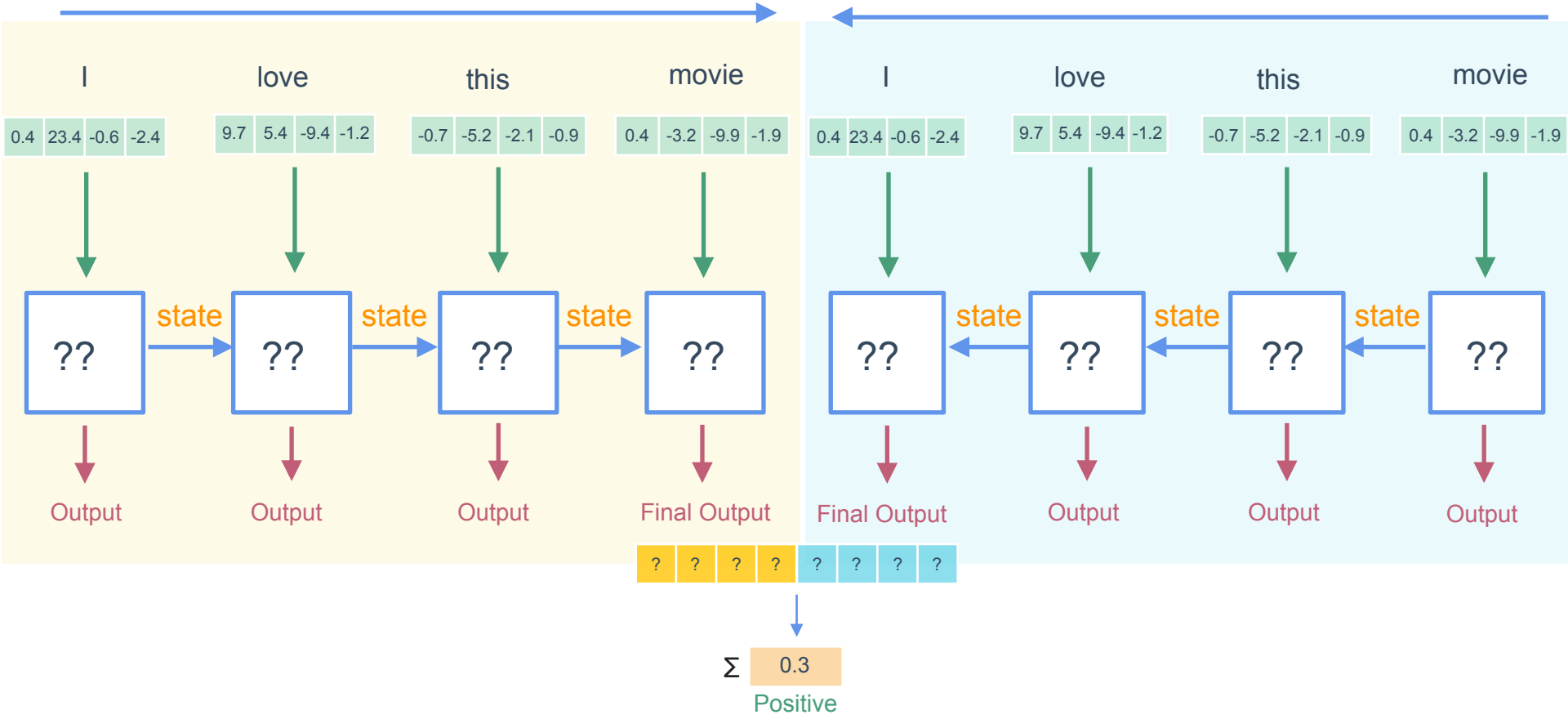
# Bidirectional LSTM



# Bidirectional LSTM

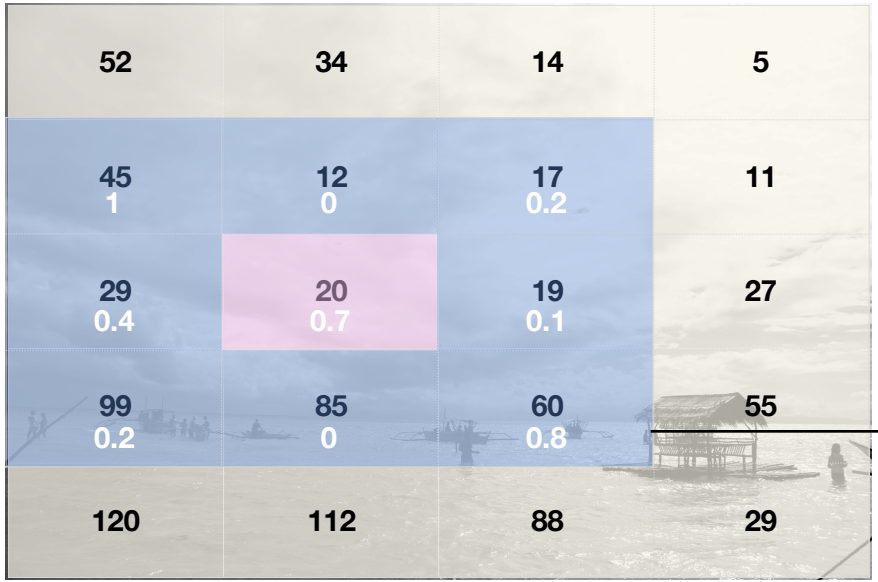


# Bidirectional LSTM



# Text Classification with CNNs

# 2D Convolution Review




Input * Kernel

(45 x 1)
(12 x 0)
(17 x 0.2)
(29 x 0.4)
(20 x 0.7)
(19 x 0.1)
(99 x 0.2)
(85 x 0)
(60 x 0.8)

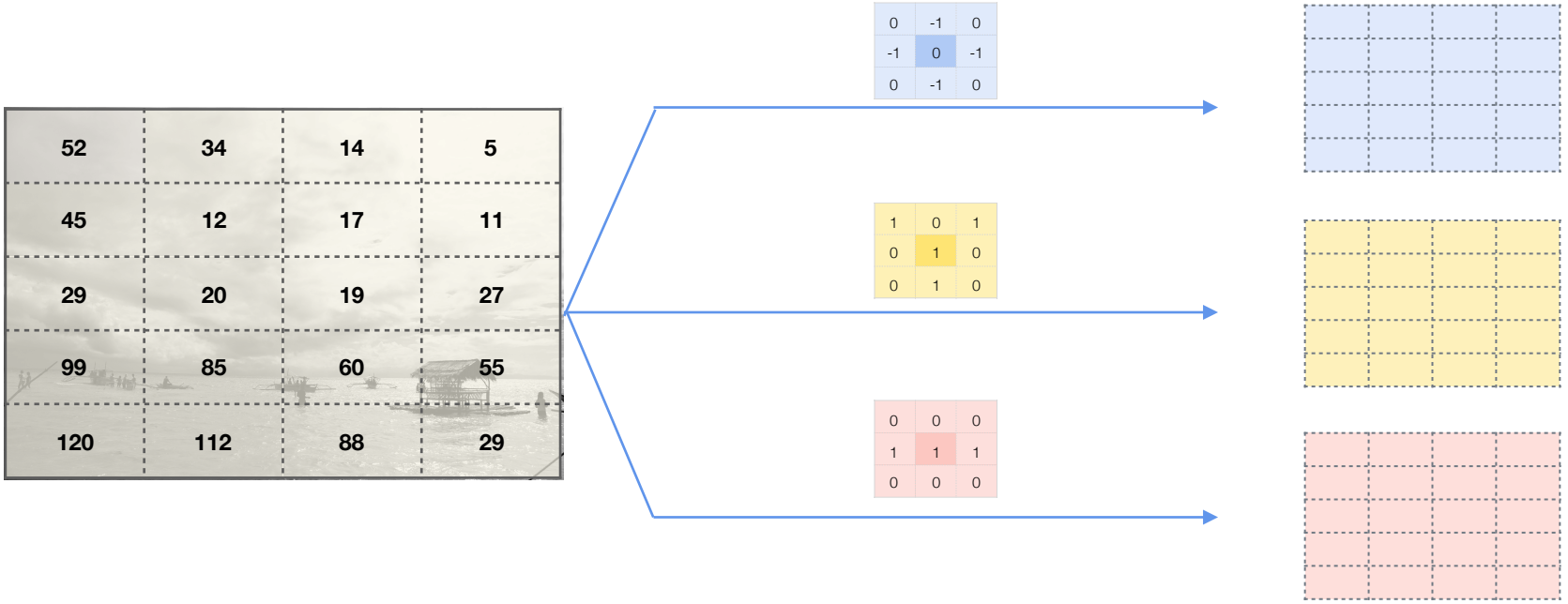
\+

143.7

| | | | |
|---|---|---|---|
| 131 | 122 | 97 | 17 |
| 114 | 104 | 78 | 88 |
| | 144 | | |
| | | | |
| | | | |

Output

# 2D Convolution Review Multiple Outputs

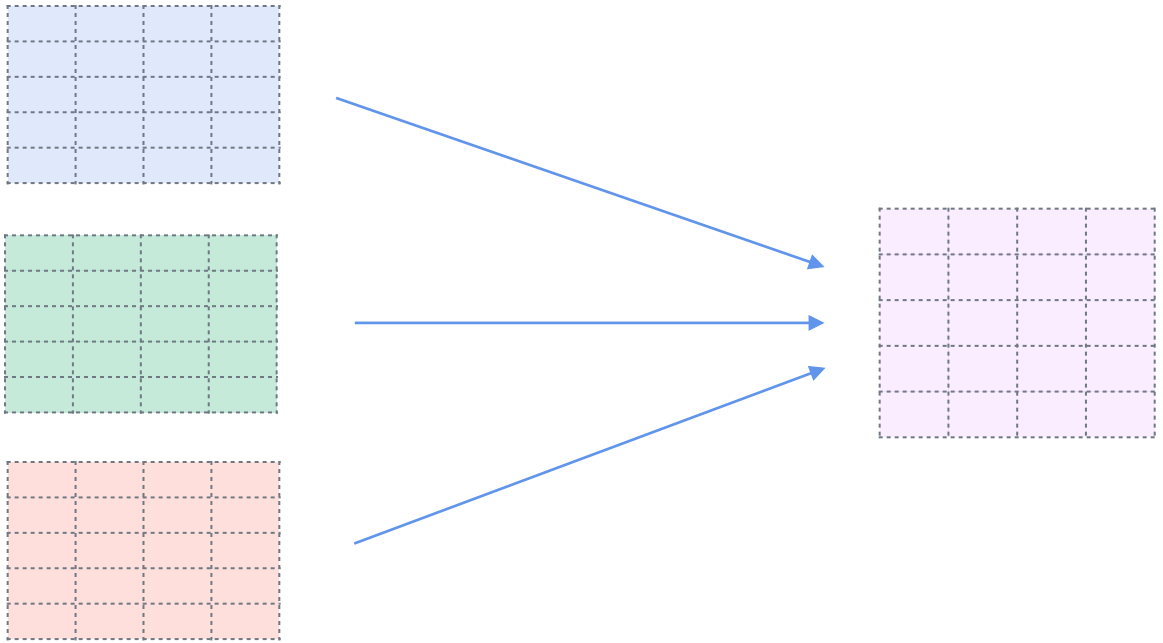



Original Image

3 Convolutions with different kernels

Multiple Outputs

# 2D Convolution Review Multiple Inputs



Multiple Convolutions

Sum all convolutions

# 1D Convolution

| | I | love | this | movie |
|---|---|---|---|---|
| Channel 1 | 0.4 (0.2) | 9.7 (0.3) | -0.7 (0.2) | 0.4 |
| Channel 2 | 23.4 | 5.4 | -5.2 | -3.2 |
| Channel 3 | -0.6 | -9.4 | -2.1 | -9.9 |
| Channel 4 | -2.4 | -1.2 | -0.9 | -1.9 |

0.4*0.2 + 9.7*0.3 + (-0.7)*0.2
= 2.85

| Conv 1 |
|---|
| 2.85 |
| |
| |
| |

# 1D Convolution

| | I | love | this | movie |
|---|---|---|---|---|
| Channel 1 | 0.4 | 9.7 | -0.7 | 0.4 |
| Channel 2 | 23.4 (0.2) | 5.4 (0.3) | -5.2 (0.2) | -3.2 |
| Channel 3 | -0.6 | -9.4 | -2.1 | -9.9 |
| Channel 4 | -2.4 | -1.2 | -0.9 | -1.9 |

23.4*0.2 + 5.4*0.3 + (-5.2)*0.2
= 5.26

| Conv 1 |
|---|
| 2.85 |
| 5.26 |
| |
| |

# 1D Convolution

| | I | love | this | movie |
|---|---|---|---|---|
| Channel 1 | 0.4 | 9.7 | -0.7 | 0.4 |
| Channel 2 | 23.4 | 5.4 | -5.2 | -3.2 |
| Channel 3 | -0.6 (0.2) | -9.4 (0.3) | -2.1 (0.2) | -9.9 |
| Channel 4 | -2.4 | -1.2 | -0.9 | -1.9 |

(-0.6)*0.2 + (-9.4)*0.3 + (-2.1)*0.2
= -3.36

| Conv 1 |
|---|
| 2.85 |
| 5.26 |
| -3.36 |
| |

# 1D Convolution

| | I | love | this | movie |
|---|---|---|---|---|
| Channel 1 | 0.4 | 9.7 | -0.7 | 0.4 |
| Channel 2 | 23.4 | 5.4 | -5.2 | -3.2 |
| Channel 3 | -0.6 | -9.4 | -2.1 | -9.9 |
| Channel 4 | -2.4 | -1.2 | -0.9 | -1.9 |
| | 0.2 | 0.3 | 0.2 | |

(-2.4)*0.2 + (-1.2)*0.3 + (-0.9)*0.2
= -1.02

| Conv 1 |
|---|
| 2.85 |
| 5.26 |
| -3.36 |
| -1.02 |

# 1D Convolution

| | I | love | this | movie |
|---|---|---|---|---|
| Channel 1 | 0.4 | 9.7 (0.3) | -0.7 (0.3) | 0.4 (0.1) |
| Channel 2 | 23.4 | 5.4 | -5.2 | -3.2 |
| Channel 3 | -0.6 | -9.4 | -2.1 | -9.9 |
| Channel 4 | -2.4 | -1.2 | -0.9 | -1.9 |

| Conv 1 | Conv 2 |
|---|---|
| 2.85 | 2.74 |
| 5.26 | |
| -3.36 | |
| -1.02 | |

9.7*0.3 + (-0.7)*0.3 + 0.4*0.1
= 2.74

# 1D Convolution

| | I | love | this | movie |
|---|---|---|---|---|
| Channel 1 | 0.4 | 9.7 | -0.7 | 0.4 |
| Channel 2 | 23.4 | 5.4 (0.3) | -5.2 (0.3) | -3.2 (0.1) |
| Channel 3 | -0.6 | -9.4 | -2.1 | -9.9 |
| Channel 4 | -2.4 | -1.2 | -0.9 | -1.9 |

| Conv 1 | Conv 2 |
|---|---|
| 2.85 | 2.74 |
| 5.26 | -0.26 |
| -3.36 | |
| -1.02 | |

5.4*0.3 + (-5.2)*0.3 + (-3.2)*0.1
= -0.26

# 1D Convolution

| | I | love | this | movie |
|---|---|---|---|---|
| Channel 1 | 0.4 | 9.7 | -0.7 | 0.4 |
| Channel 2 | 23.4 | 5.4 | -5.2 | -3.2 |
| Channel 3 | -0.6 | -9.4 (0.3) | -2.1 (0.3) | -9.9 (0.1) |
| Channel 4 | -2.4 | -1.2 | -0.9 | -1.9 |

| Conv 1 | Conv 2 |
|---|---|
| 2.85 | 2.74 |
| 5.26 | -0.26 |
| -3.36 | -4.44 |
| -1.02 | |

(-9.4)*0.3 + (-2.1)*0.3 + (-9.9)*0.1
= -4.44

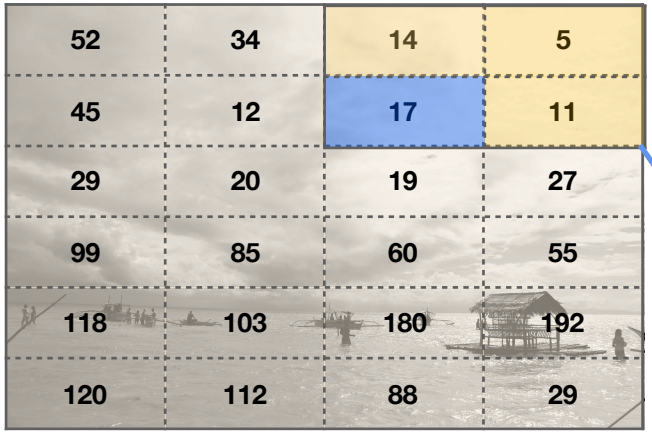
# 1D Convolution

| | I | love | this | movie |
|---|---|---|---|---|
| Channel 1 | 0.4 | 9.7 | -0.7 | 0.4 |
| Channel 2 | 23.4 | 5.4 | -5.2 | -3.2 |
| Channel 3 | -0.6 | -9.4 | -2.1 | -9.9 |
| Channel 4 | -2.4 | -1.2 (0.3) | -0.9 (0.3) | -1.9 (0.1) |

| Conv 1 | Conv 2 |
|---|---|
| 2.85 | 2.74 |
| 5.26 | -0.26 |
| -3.36 | -4.44 |
| -1.02 | -0.82 |

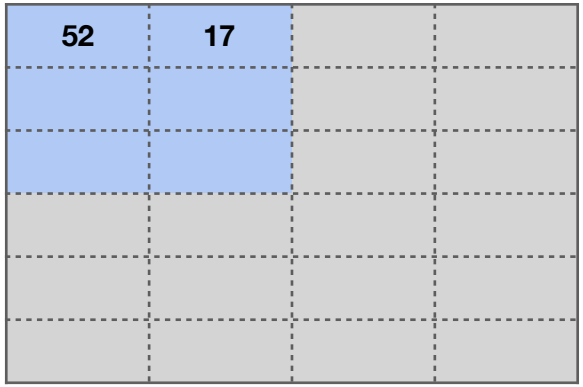
(-1.2)*0.3 + (-0.9)*0.3 + (-1.9)*0.1
= -0.82

# 2D Max Pooling Review

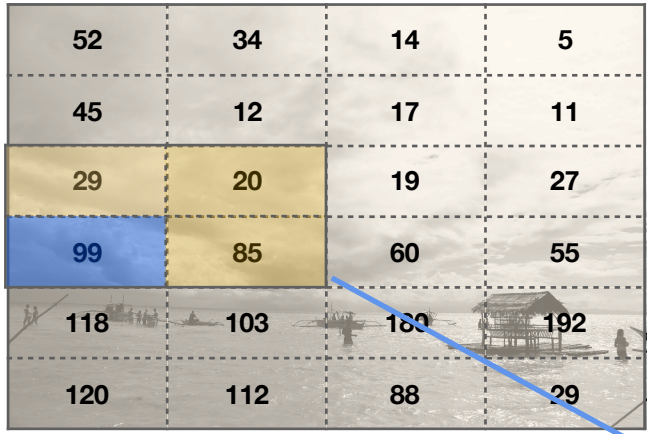



Input 6x4

Output 3x2



# 2D Max Pooling Review

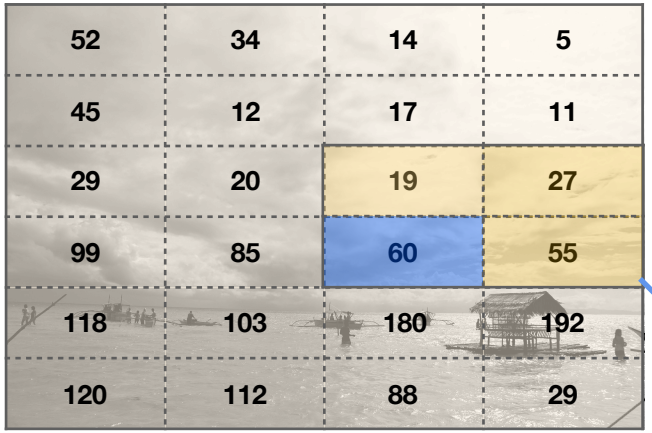



Input 6x4

Output 3x2

| | | | |
|---|---|---|---|
| 52 | 17 | | |
| 99 | | | |
| | | | |
| | | | |
| | | | |
| | | | |

# 2D Max Pooling Review




Input 6x4

Output 3x2

| | | | |
|---|---|---|---|
| 52 | 17 | | |
| 99 | 65 | | |
| | | | |
| | | | |
| | | | |
| | | | |

# 1D Max Pooling

| | I | love | this | movie |
|---|---|---|---|---|
| Channel 1 | 0.4 | 9.7 | -0.7 | 0.4 |
| Channel 2 | 23.4 | 5.4 | -5.2 | -3.2 |
| Channel 3 | -0.6 | -9.4 | -2.1 | -9.9 |
| Channel 4 | -2.4 | -1.2 | -0.9 | -1.9 |

| Max Pooling 1 | Max Pooling 2 |
|---|---|
| 9.7 | |
| | |
| | |
| | |

# 1D Max Pooling

| | I | love | this | movie |
|---|---|---|---|---|
| Channel 1 | 0.4 | 9.7 | -0.7 | 0.4 |
| Channel 2 | 23.4 | 5.4 | -5.2 | -3.2 |
| Channel 3 | -0.6 | -9.4 | -2.1 | -9.9 |
| Channel 4 | -2.4 | -1.2 | -0.9 | -1.9 |

| Max Pooling 1 | Max Pooling 2 |
|---|---|
| 9.7 | |
| 23.4 | |
| | |
| | |

# 1D Max Pooling

| | I | love | this | movie |
|---|---|---|---|---|
| Channel 1 | 0.4 | 9.7 | -0.7 | 0.4 |
| Channel 2 | 23.4 | 5.4 | -5.2 | -3.2 |
| Channel 3 | -0.6 | -9.4 | -2.1 | -9.9 |
| Channel 4 | -2.4 | -1.2 | -0.9 | -1.9 |

| Max Pooling 1 | Max Pooling 2 |
|---|---|
| 9.7 | |
| 23.4 | |
| -0.6 | |
| | |

# 1D Max Pooling

| | I | love | this | movie |
|---|---|---|---|---|
| Channel 1 | 0.4 | 9.7 | -0.7 | 0.4 |
| Channel 2 | 23.4 | 5.4 | -5.2 | -3.2 |
| Channel 3 | -0.6 | -9.4 | -2.1 | -9.9 |
| Channel 4 | -2.4 | -1.2 | -0.9 | -1.9 |

| Max Pooling 1 | Max Pooling 2 |
|---|---|
| 9.7 | |
| 23.4 | |
| -0.6 | |
| -1.2 | |

# 1D Max Pooling

| | I | love | this | movie |
|---|---|---|---|---|
| Channel 1 | 0.4 | 9.7 | -0.7 | 0.4 |
| Channel 2 | 23.4 | 5.4 | -5.2 | -3.2 |
| Channel 3 | -0.6 | -9.4 | -2.1 | -9.9 |
| Channel 4 | -2.4 | -1.2 | -0.9 | -1.9 |

| Max Pooling 1 | Max Pooling 2 |
|---|---|
| 9.7 | 0.4 |
| 23.4 | |
| -0.6 | |
| -1.2 | |

# 1D Max Pooling

| | I | love | this | movie |
|---|---|---|---|---|
| Channel 1 | 0.4 | 9.7 | -0.7 | 0.4 |
| Channel 2 | 23.4 | 5.4 | -5.2 | -3.2 |
| Channel 3 | -0.6 | -9.4 | -2.1 | -9.9 |
| Channel 4 | -2.4 | -1.2 | -0.9 | -1.9 |

| Max Pooling 1 | Max Pooling 2 |
|---|---|
| 9.7 | 0.4 |
| 23.4 | -3.2 |
| -0.6 | |
| -1.2 | |

# 1D Max Pooling

| | I | love | this | movie |
|---|---|---|---|---|
| Channel 1 | 0.4 | 9.7 | -0.7 | 0.4 |
| Channel 2 | 23.4 | 5.4 | -5.2 | -3.2 |
| Channel 3 | -0.6 | -9.4 | -2.1 | -9.9 |
| Channel 4 | -2.4 | -1.2 | -0.9 | -1.9 |

| Max Pooling 1 |
|---|
| 9.7 |
| 23.4 |
| -0.6 |
| -1.2 |

| Max Pooling 2 |
|---|
| 0.4 |
| -3.2 |
| -2.1 |
| |

# 1D Max Pooling

| | I | love | this | movie |
|---|---|---|---|---|
| Channel 1 | 0.4 | 9.7 | -0.7 | 0.4 |
| Channel 2 | 23.4 | 5.4 | -5.2 | -3.2 |
| Channel 3 | -0.6 | -9.4 | -2.1 | -9.9 |
| Channel 4 | -2.4 | -1.2 | -0.9 | -1.9 |

| Max Pooling 1 | Max Pooling 2 |
|---|---|
| 9.7 | 0.4 |
| 23.4 | -3.2 |
| -0.6 | -2.1 |
| -1.2 | -0.9 |

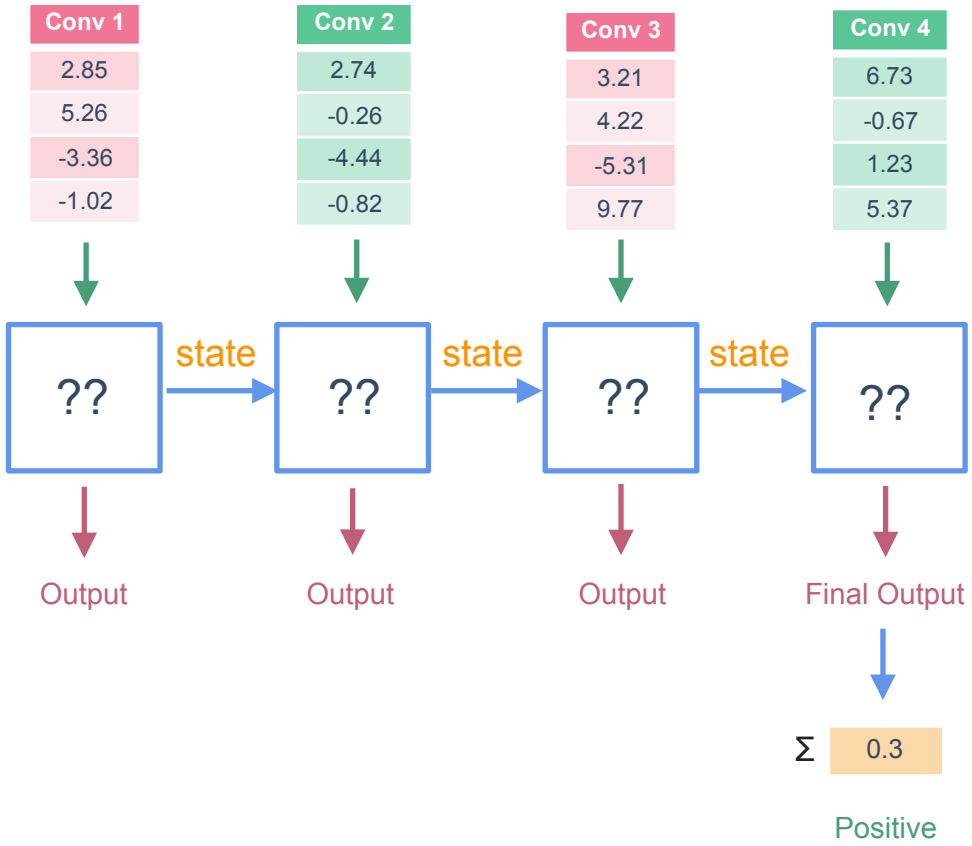
# 1D Convolution

| | I | love | this | movie |
|---|---|---|---|---|
| Channel 1 | 0.4 | 9.7 | -0.7 | 0.4 |
| Channel 2 | 23.4 | 5.4 | -5.2 | -3.2 |
| Channel 3 | -0.6 | -9.4 | -2.1 | -9.9 |
| Channel 4 | -2.4 | -1.2 (0.3) | -0.9 (0.3) | -1.9 (0.1) |

| Conv 1 | Conv 2 |
|---|---|
| 2.85 | 2.74 |
| 5.26 | -0.26 |
| -3.36 | -4.44 |
| -1.02 | -0.82 |

(-1.2)*0.3 + (-0.9)*0.3 + (-1.9)*0.1
= -0.82

# CNN/LSTM Hybrid




Input

Word embeddings

Convolutions

LSTM

Encoded Output

Dense layer

Classification

# Seq2Seq
# ml-class/lstm/seq2seq

# Seq2Seq in translation

Input

My name is Lukas

??

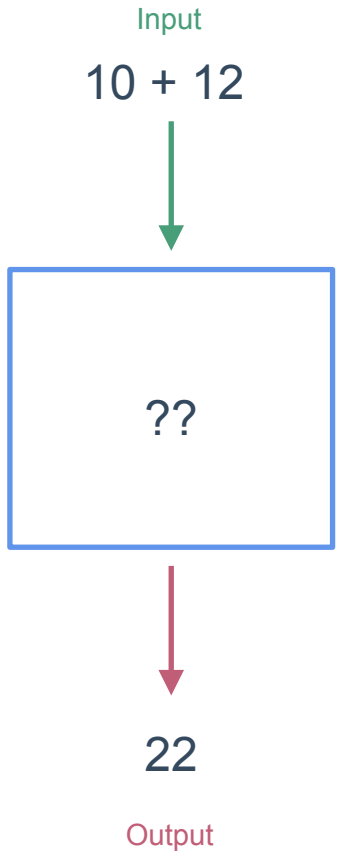
私の名前はルーカスです

Output

# seq2seq: the clown car of deep learning

tl; dr: Translating arbitrary-length sequences back and forth is easier than you think

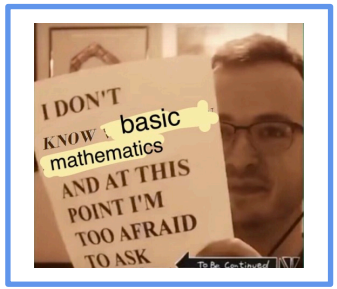
# Seq2Seq in this tutorial



# Seq2Seq in this tutorial

Input

10 + 12

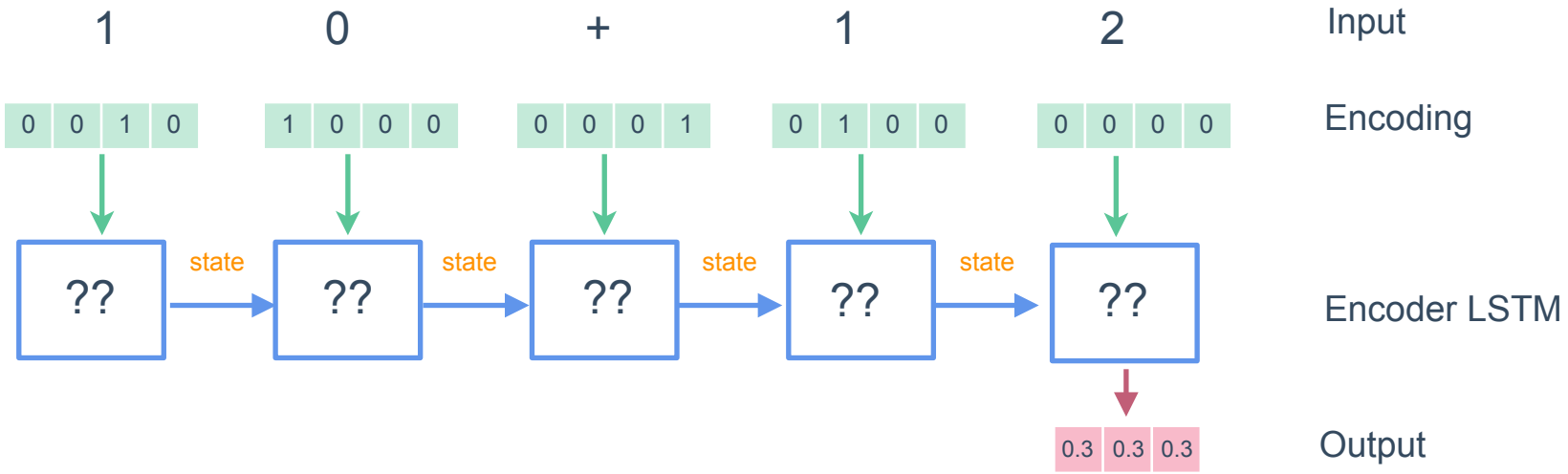


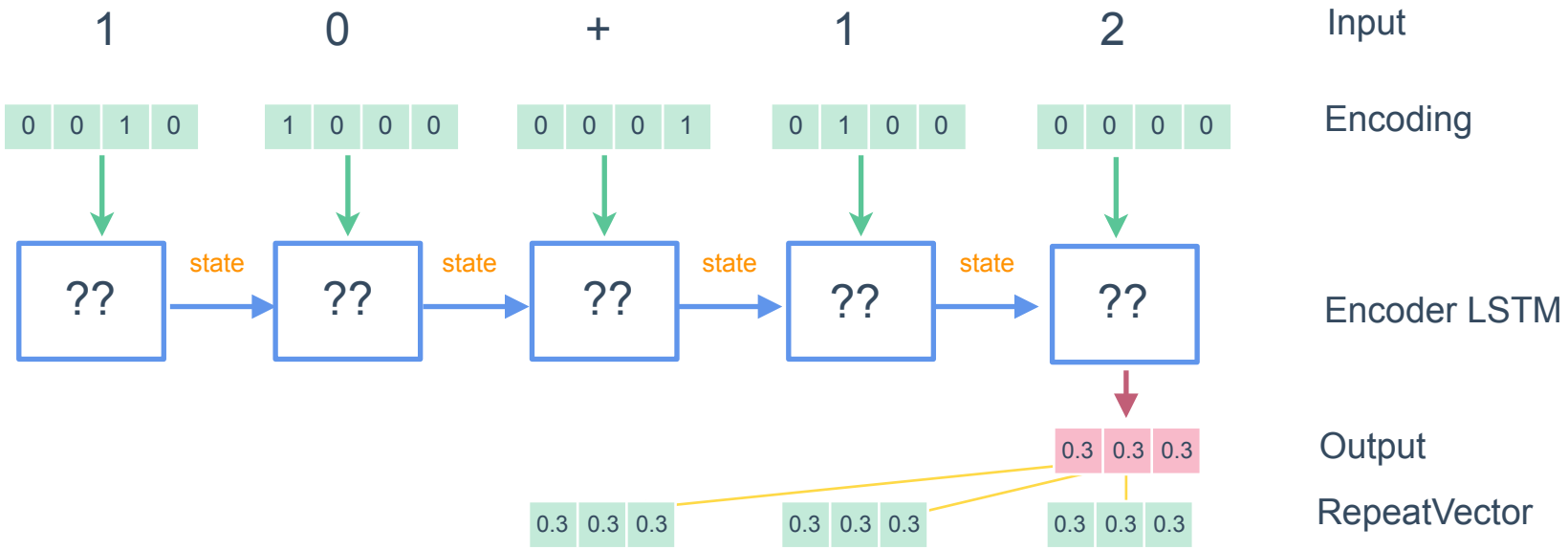

22

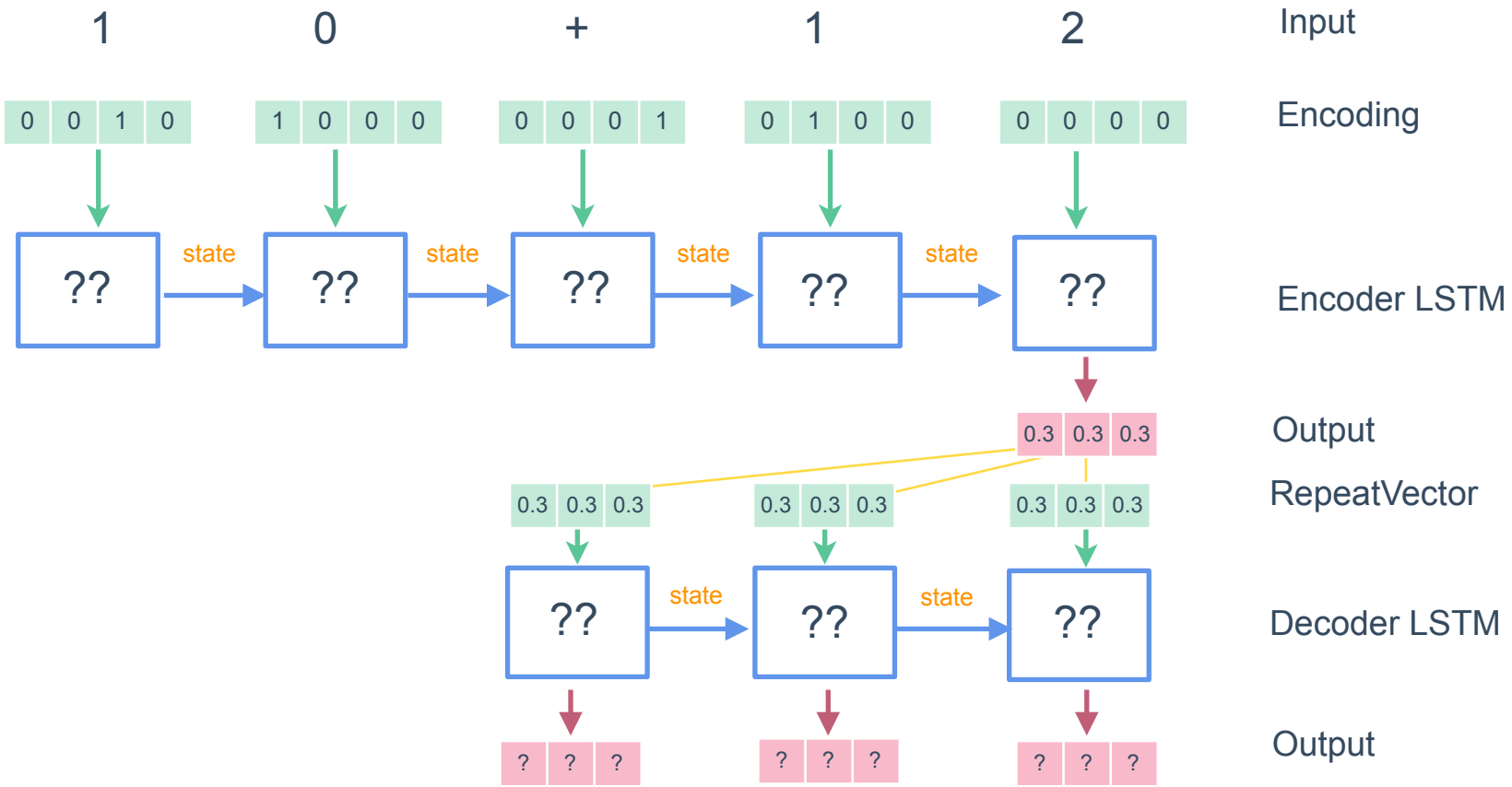
Output

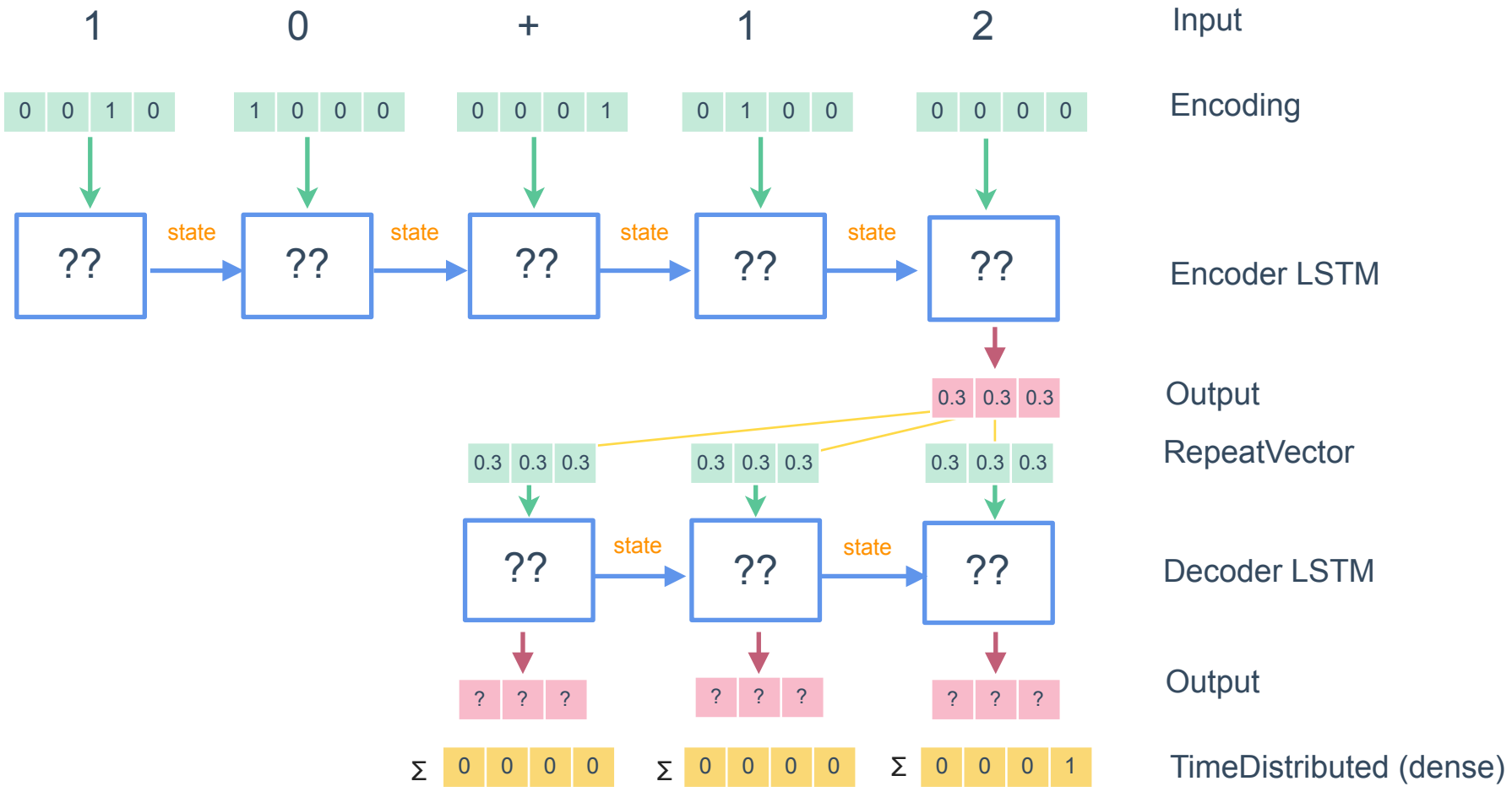
| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | + | 1 | 2 | Input |

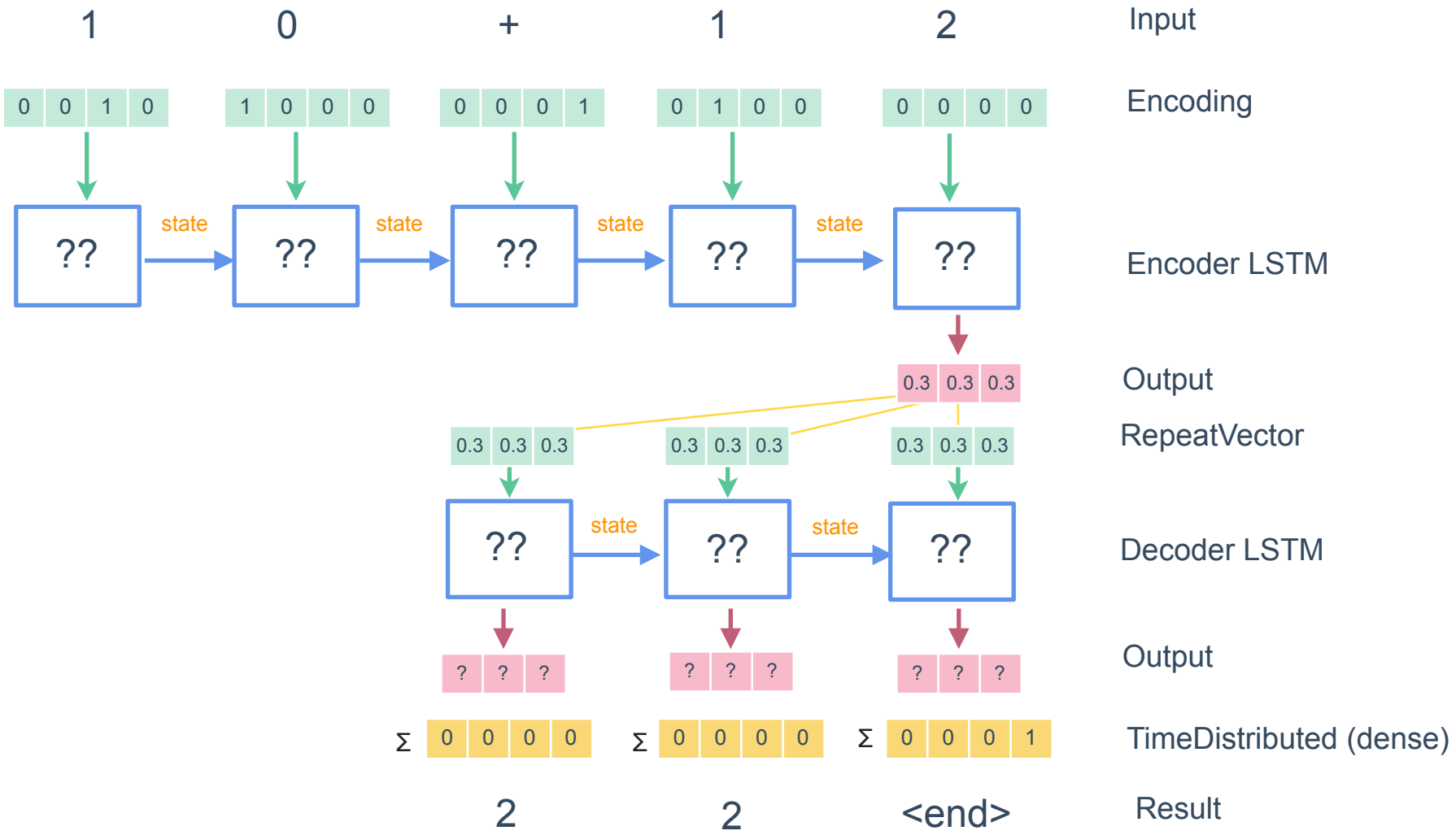
| 1 | 0 | + | 1 | 2 | Input |
|---|---|---|---|---|---|
| 0 0 1 0 | 1 0 0 0 | 0 0 0 1 | 0 1 0 0 | 0 0 0 0 | Encoding |


1
0
+
1
2
Input
0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0
0 0 0 0
Encoding
??
state
??
state
??
state
??
state
??
Encoder LSTM
0.3 0.3 0.3
Output

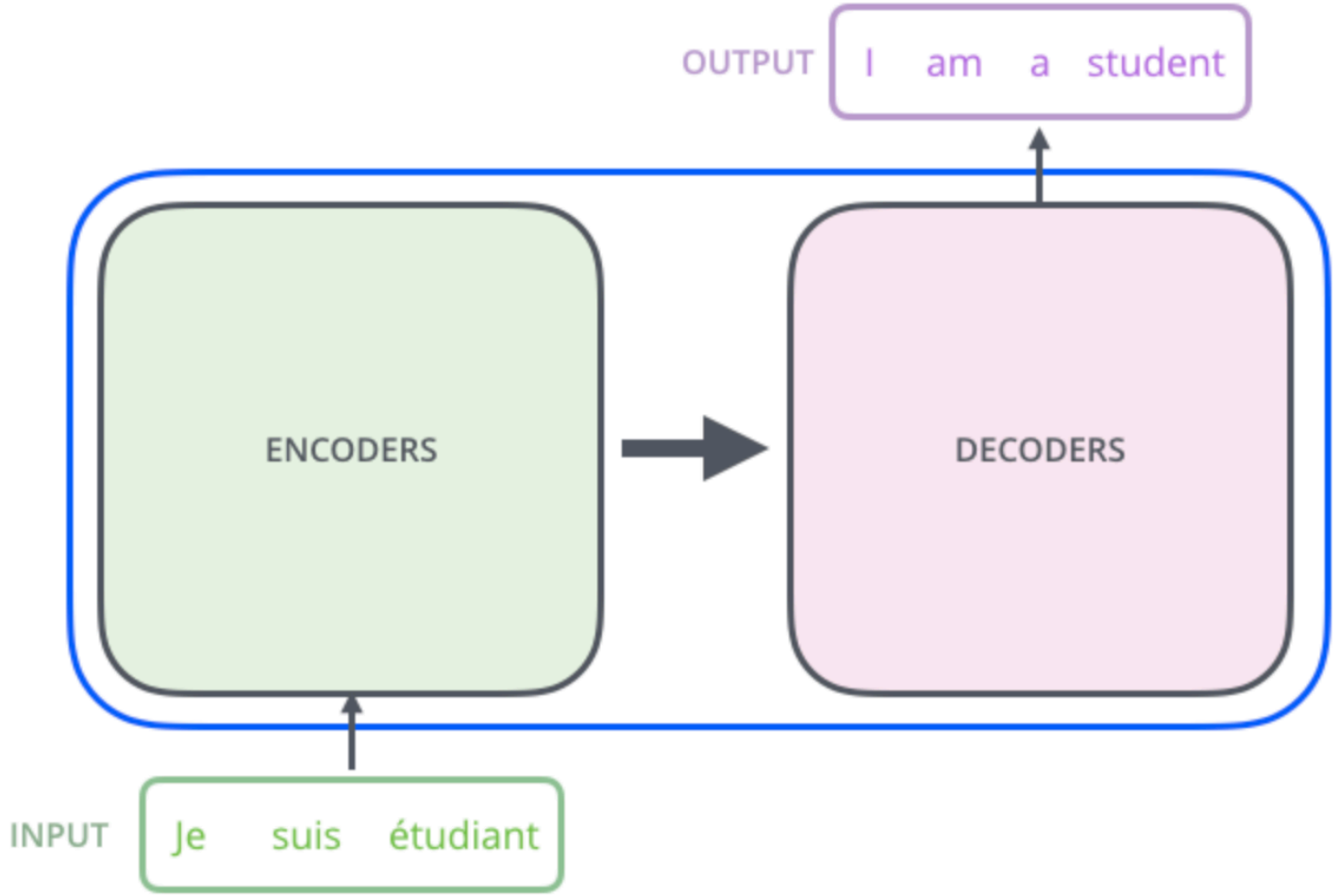

1
0
+
1
2
Input
0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0
0 0 0 0
Encoding
??
??
??
??
??
state
state
state
state
Encoder LSTM
0.3 0.3 0.3
Output
0.3 0.3 0.3
0.3 0.3 0.3
0.3 0.3 0.3
RepeatVector


1
0
+
1
2
Input
0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0
0 0 0 0
Encoding
??
state
??
state
??
state
??
state
??
Encoder LSTM
0.3 0.3 0.3
Output
RepeatVector
0.3 0.3 0.3
0.3 0.3 0.3
0.3 0.3 0.3
??
state
??
state
??
Decoder LSTM
? ? ?
? ? ?
? ? ?
Output

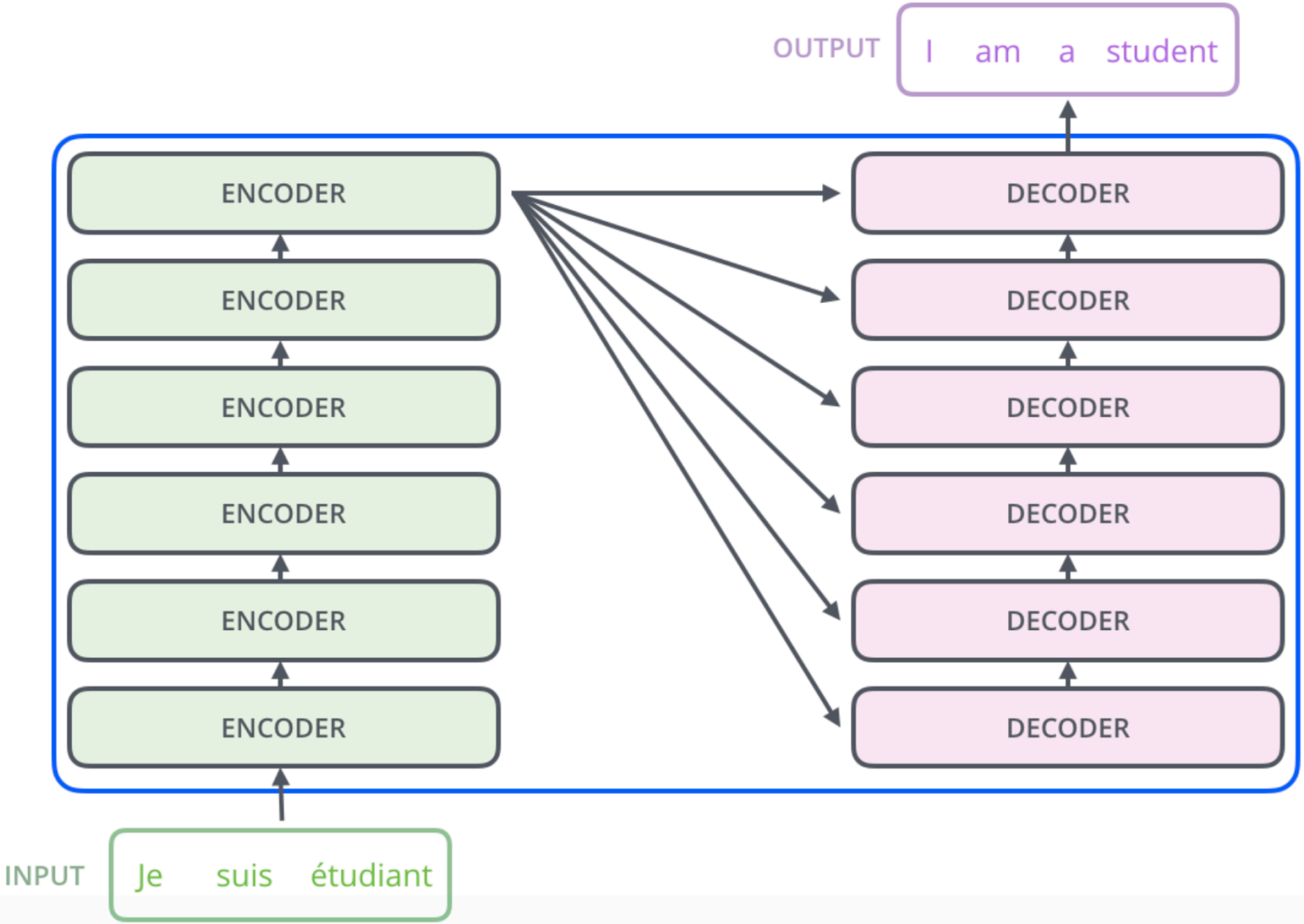

1
0
+
1
2
Input
0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0
0 0 0 0
Encoding
??
state
??
state
??
state
??
state
??
Encoder LSTM
0.3 0.3 0.3
Output
0.3 0.3 0.3
0.3 0.3 0.3
0.3 0.3 0.3
RepeatVector
??
state
??
state
??
Decoder LSTM
? ? ?
? ? ?
? ? ?
Output
Σ 0 0 0 0
Σ 0 0 0 0
Σ 0 0 0 1
TimeDistributed (dense)

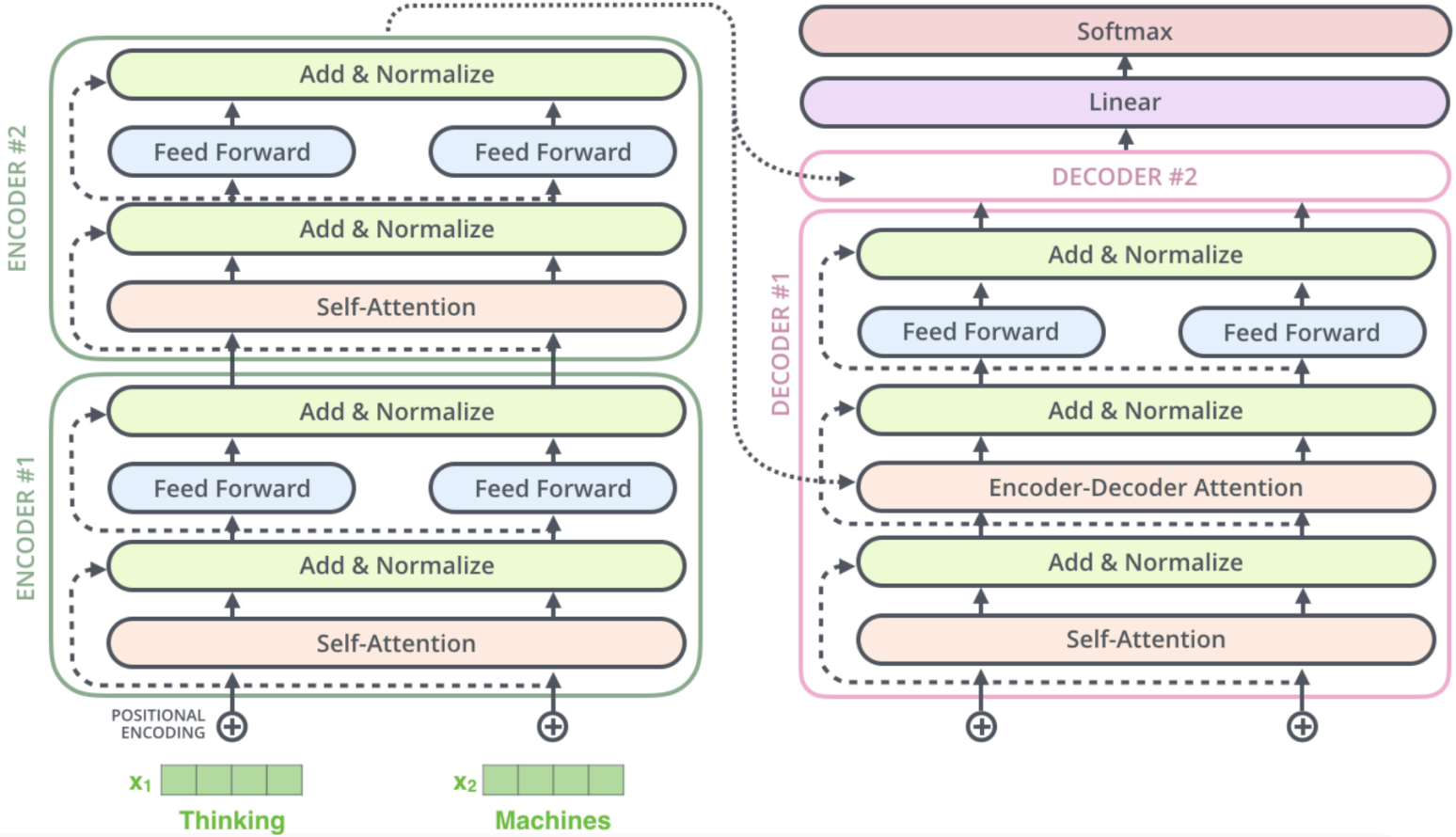

1
0
+
1
2
Input
0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0
0 0 0 0
Encoding
??
state
??
state
??
state
??
state
??
Encoder LSTM
0.3 0.3 0.3
Output
0.3 0.3 0.3
0.3 0.3 0.3
0.3 0.3 0.3
RepeatVector
??
state
??
state
??
Decoder LSTM
? ? ?
? ? ?
? ? ?
Output
Σ 0 0 0 0
Σ 0 0 0 0
Σ 0 0 0 1
TimeDistributed (dense)
2
2
<end>
Result

# Transformers




http://jalammar.github.io/illustrated-transformer/


OUTPUT
I am a student
ENCODER
ENCODER
ENCODER
ENCODER
ENCODER
ENCODER
DECODER
DECODER
DECODER
DECODER
DECODER
DECODER
INPUT
Je suis étudiant


Softmax
Linear
DECODER #2
DECODER #1
Add & Normalize
Feed Forward
Feed Forward
Add & Normalize
Encoder-Decoder Attention
Add & Normalize
Self-Attention
ENCODER #2
Add & Normalize
Feed Forward
Feed Forward
Add & Normalize
Self-Attention
ENCODER #1
Add & Normalize
Feed Forward
Feed Forward
Add & Normalize
Self-Attention
POSITIONAL ENCODING
x1
x2
Thinking
Machines

# Conclusion

# More about LSTMs

- More online tutorials https://www.wandb.com/classes
- Colah's blog http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- Andrej's blog http://karpathy.github.io/2015/05/21/rnn-effectiveness/
- Attention networks https://richliao.github.io/supervised/classification/2016/12/26/textclassifier-HATN/
- Stanford CS 224N

# More Resources for Deep Learning/ML

- Books
  - Deep Learning Book (http://www.deeplearningbook.org/)
  - Artificial Intelligence: A Modern Approach
  - Hands-On Machine Learning with Scikit-Learn and TensorFlow
  - Deep Learning with Python
- Video Classes
  - wandb.com/classes
- Hands-on
  - wandb.com/benchmarks