Language Modeling with Recurrent Neural Networks in TensorFlow

APPLYING BIDIRECTIONAL RECURRENT NEURAL NETWORKS TO WORD RECOGNITION

Janani Ravi
CO-FOUNDER, LOONYCORN
www.loonycorn.com

Overview

Recurrent Neural Networks (RNNs) work well with sequential data

Neurons in RNNs maintain internal state

This helps RNNs learn and remember the past

LSTM and GRU neurons are recurrent neurons that can remember even the distant past

Bidirectional RNNs are a type of RNN that make two passes over sequential input data

Bidirectional RNNs out-perform RNNs in word recognition and speech recognition

Prerequisites and Course Outline

Prerequisite Courses

Understanding the Foundations of TensorFlow

Sentiment Analysis with Recurrent Neural Networks in TensorFlow

Building Classification Models in TensorFlow

Software and Skills

Have TensorFlow installed

Know how to run TensorFlow code

Understand the basics of machine learning and neural networks

Understand Recurrent Neural Networks (RNNs)

Be comfortable with Python programming

Course Outline

Understand bidirectional RNNs

- Using them in Optical Character Recognition

Implement OCR using bidirectional RNNs

Understand character prediction using RNNs

- Additional state using multi-RNN cells, perplexity measure for language modeling

Implement character prediction and text generation using RNNs

Recurrent Neural Networks (RNNs)

```
y = f(x)
```

Machine Learning

Machine learning algorithms seek to “learn” the function f that links the features and the labels

$$y_t = f(x_t, y_{t-1})$$

Learning the Past

Relationships where past values of the effect variable drive current values are called auto-regressive

$$y_t = f(x_t, y_{t-1})$$

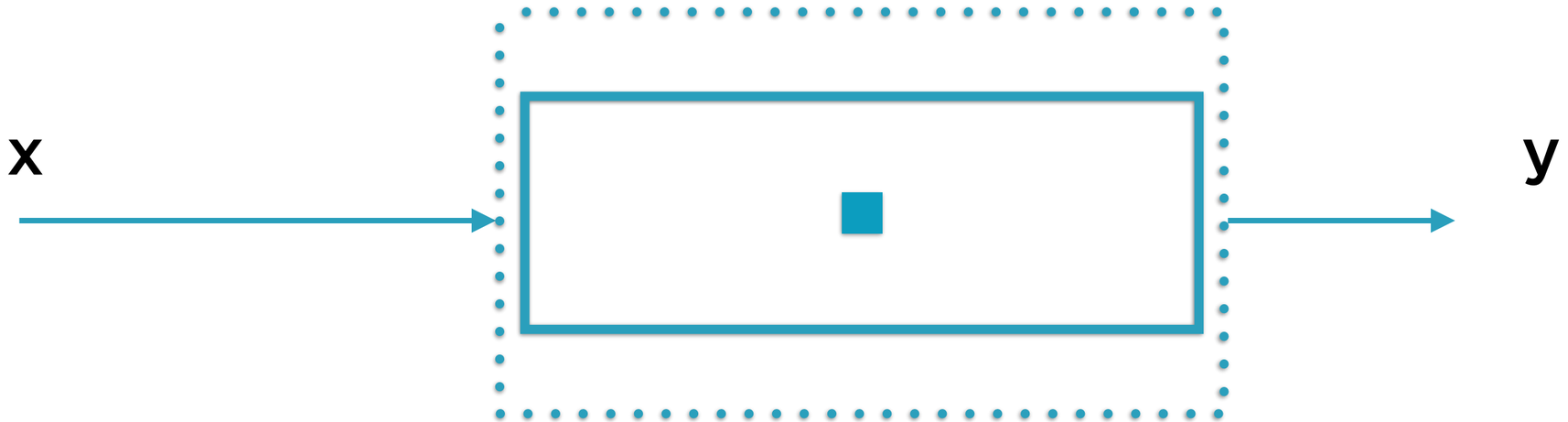
Learning the Past

Relationships where past values of the effect variable drive current values are called auto-regressive

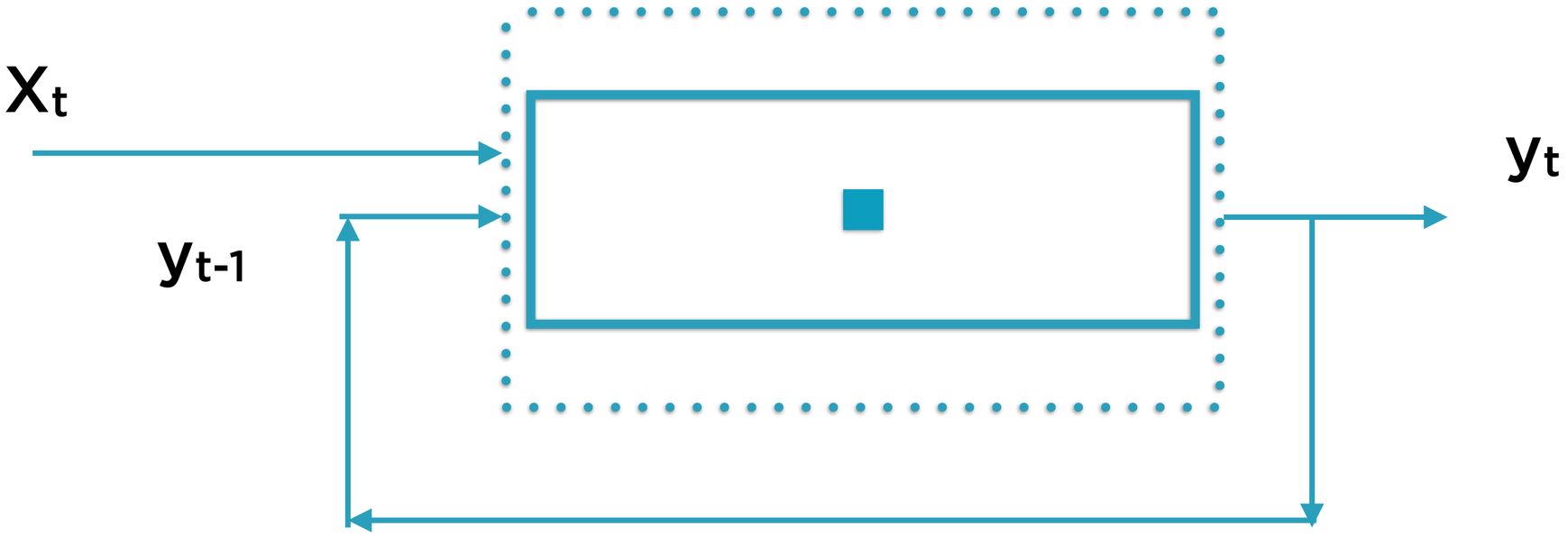
Feed-forward networks cannot learn from the **past**

Recurrent neural networks can

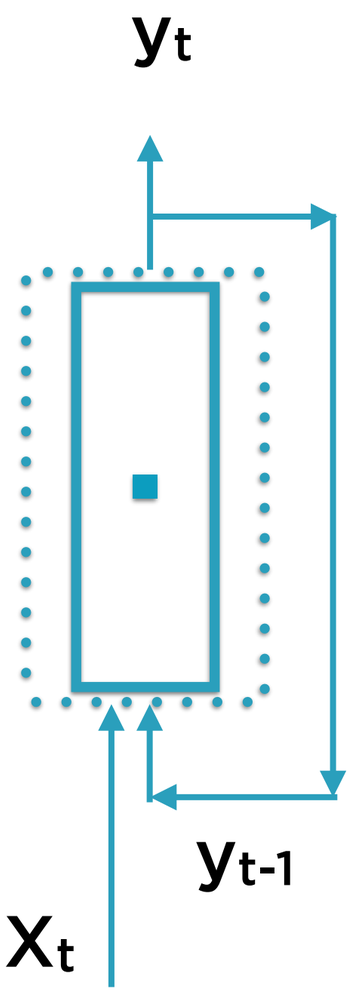
Simplest Feed-forward Neuron



Simplest Recurrent Neuron



Recurrent Neuron

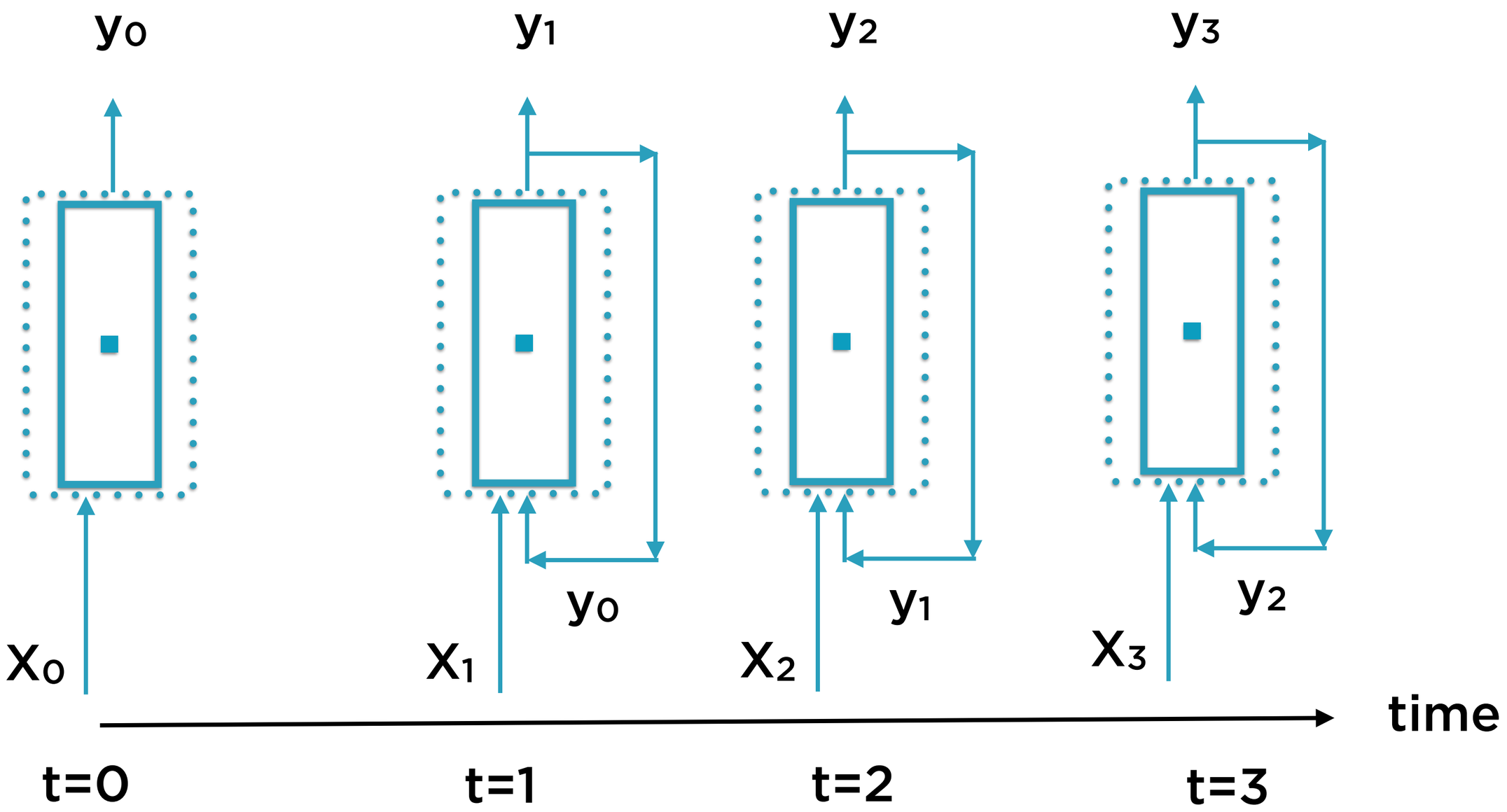


y_t = Output at time t

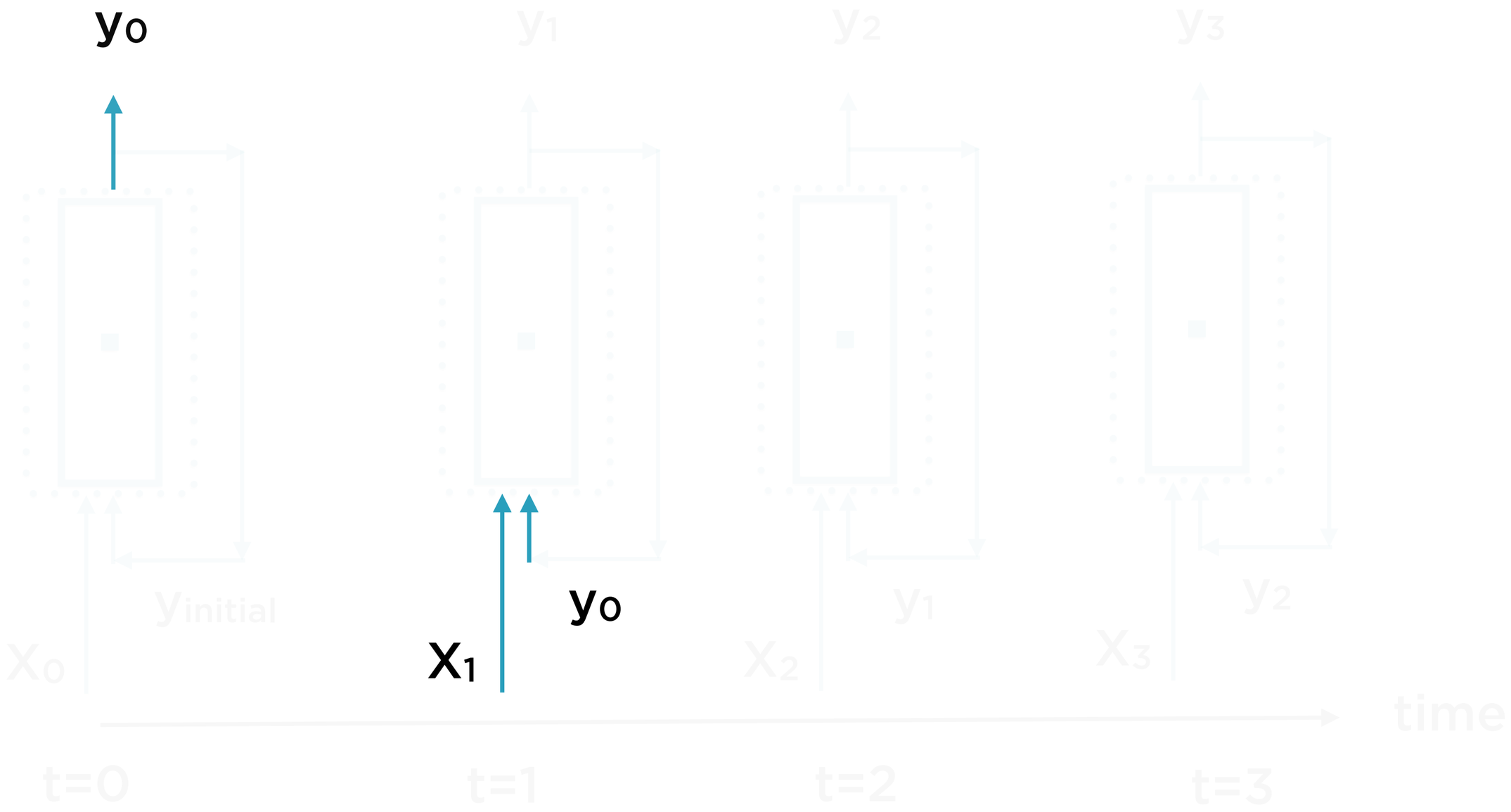
Depends upon

- y_{t-1} = Output at time t - 1
- x_t = New inputs available only at time t

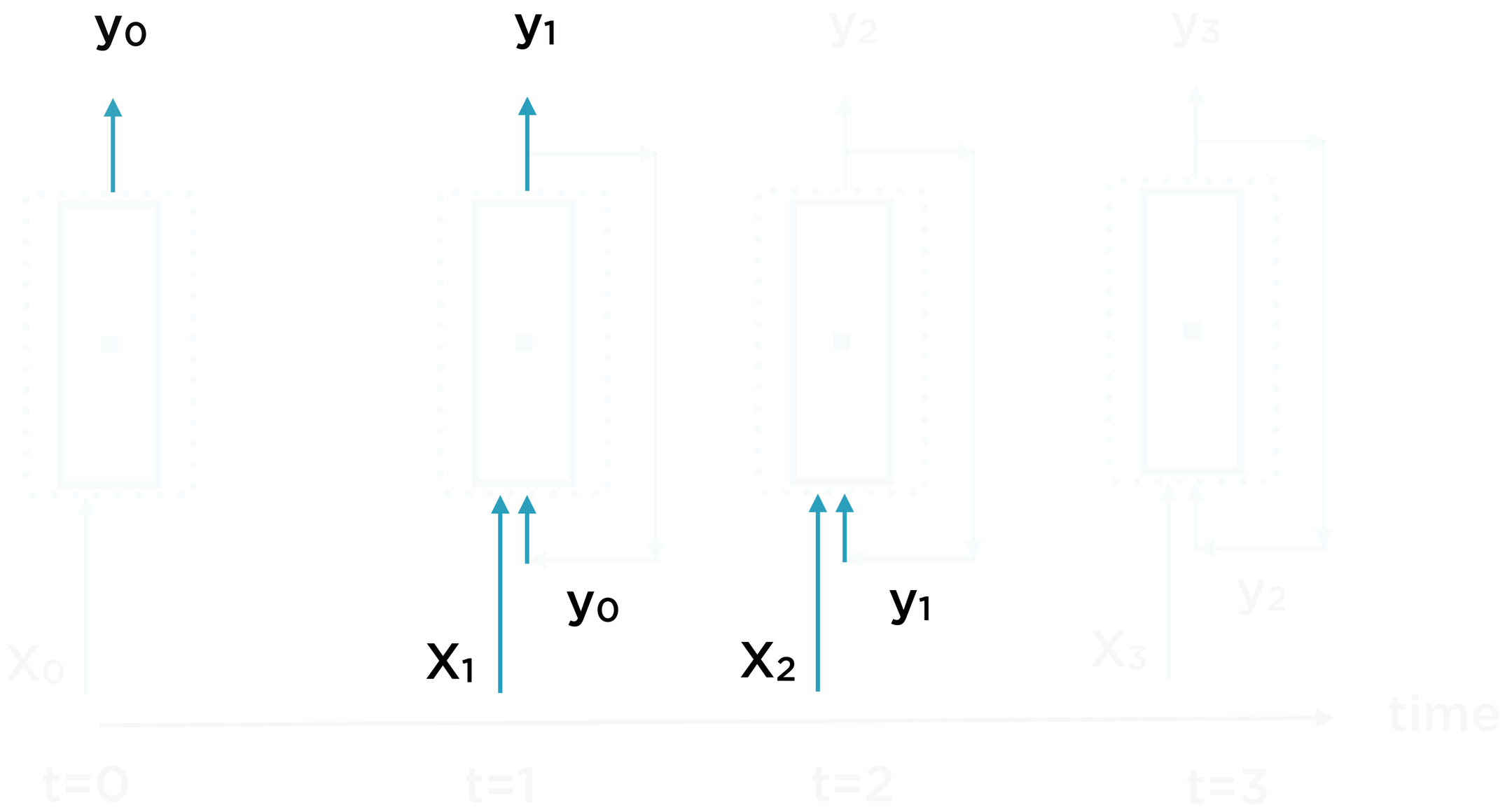
Unrolling Through Time



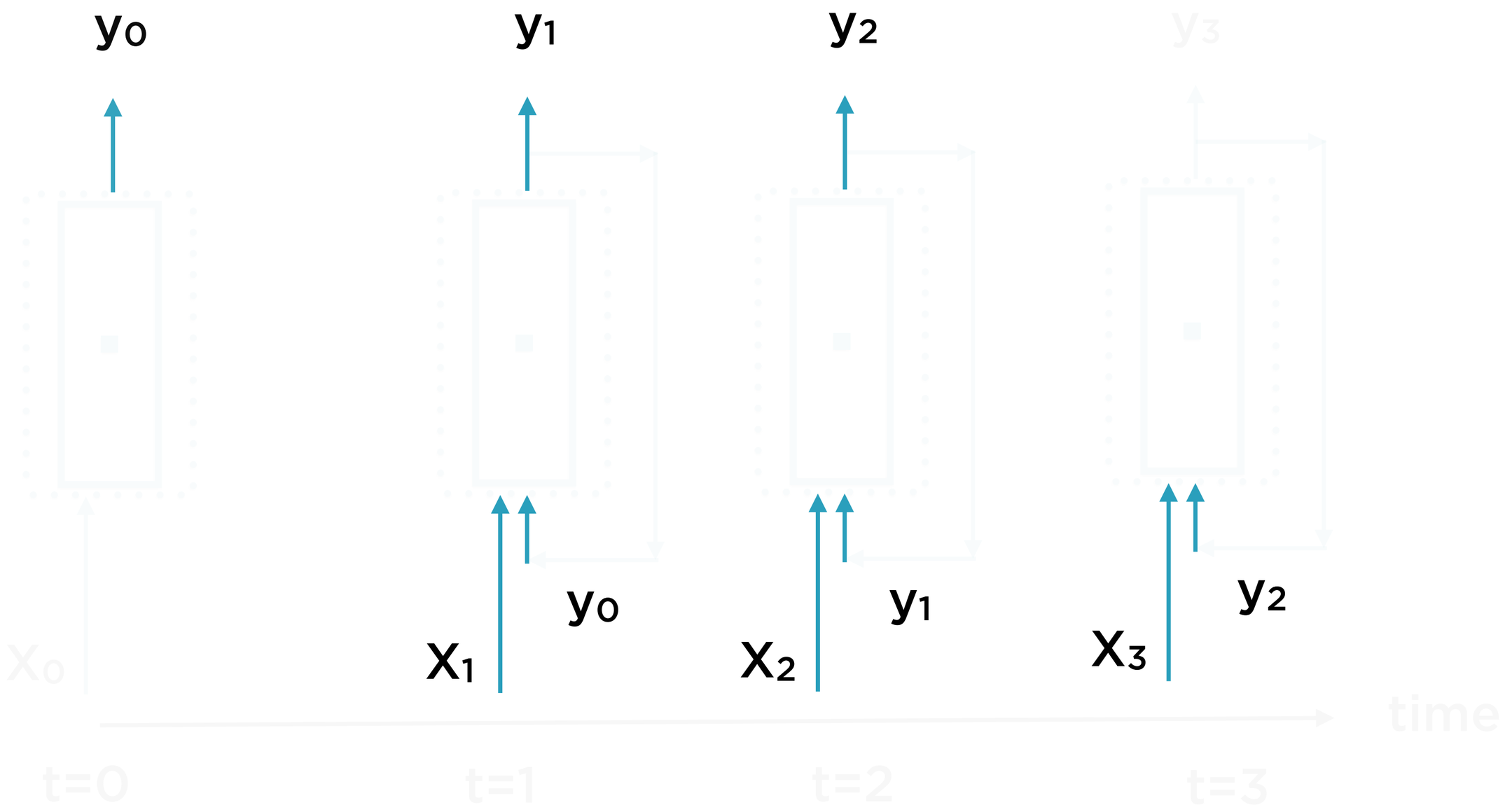
Unrolling Through Time



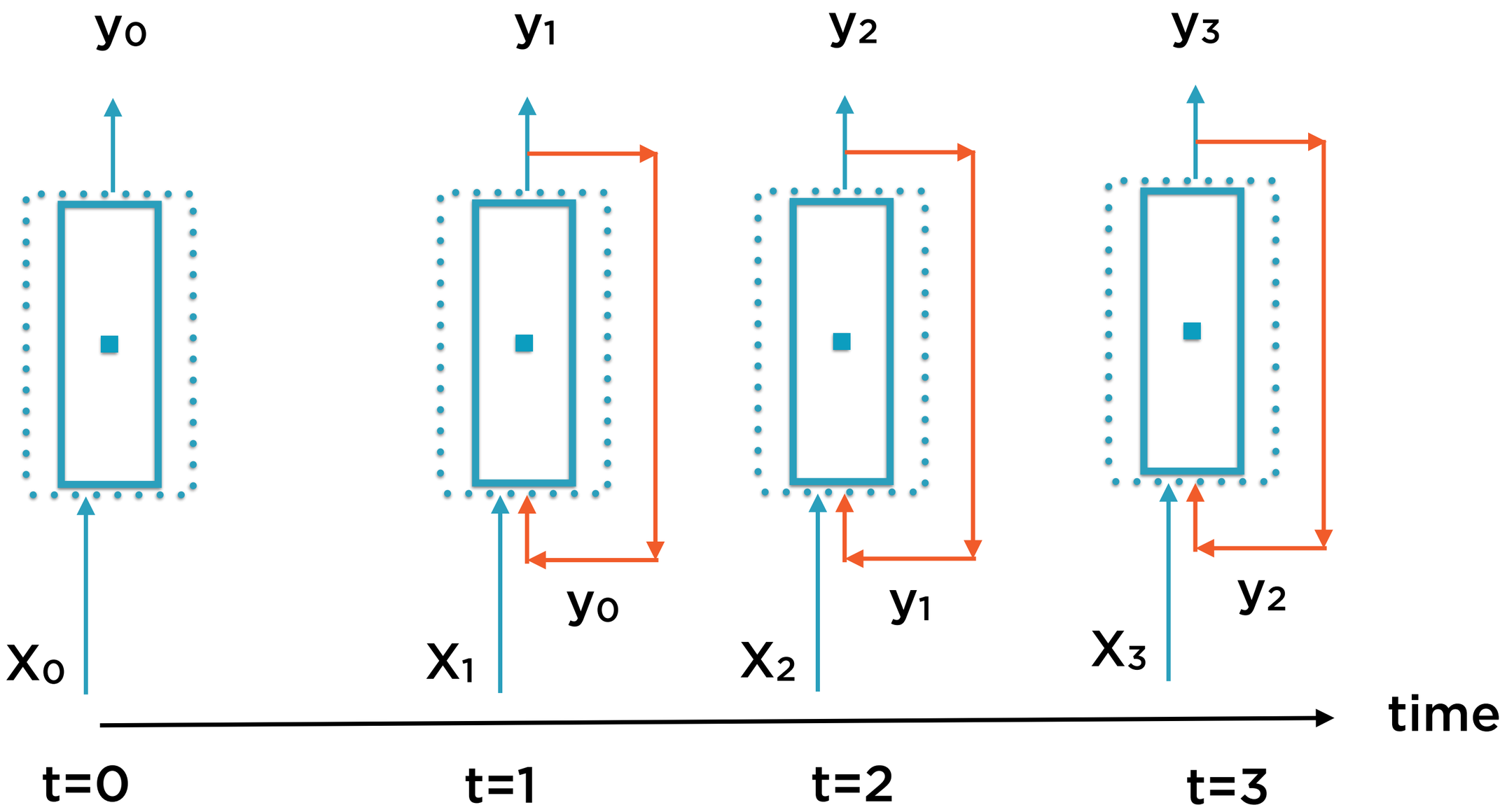
Unrolling Through Time



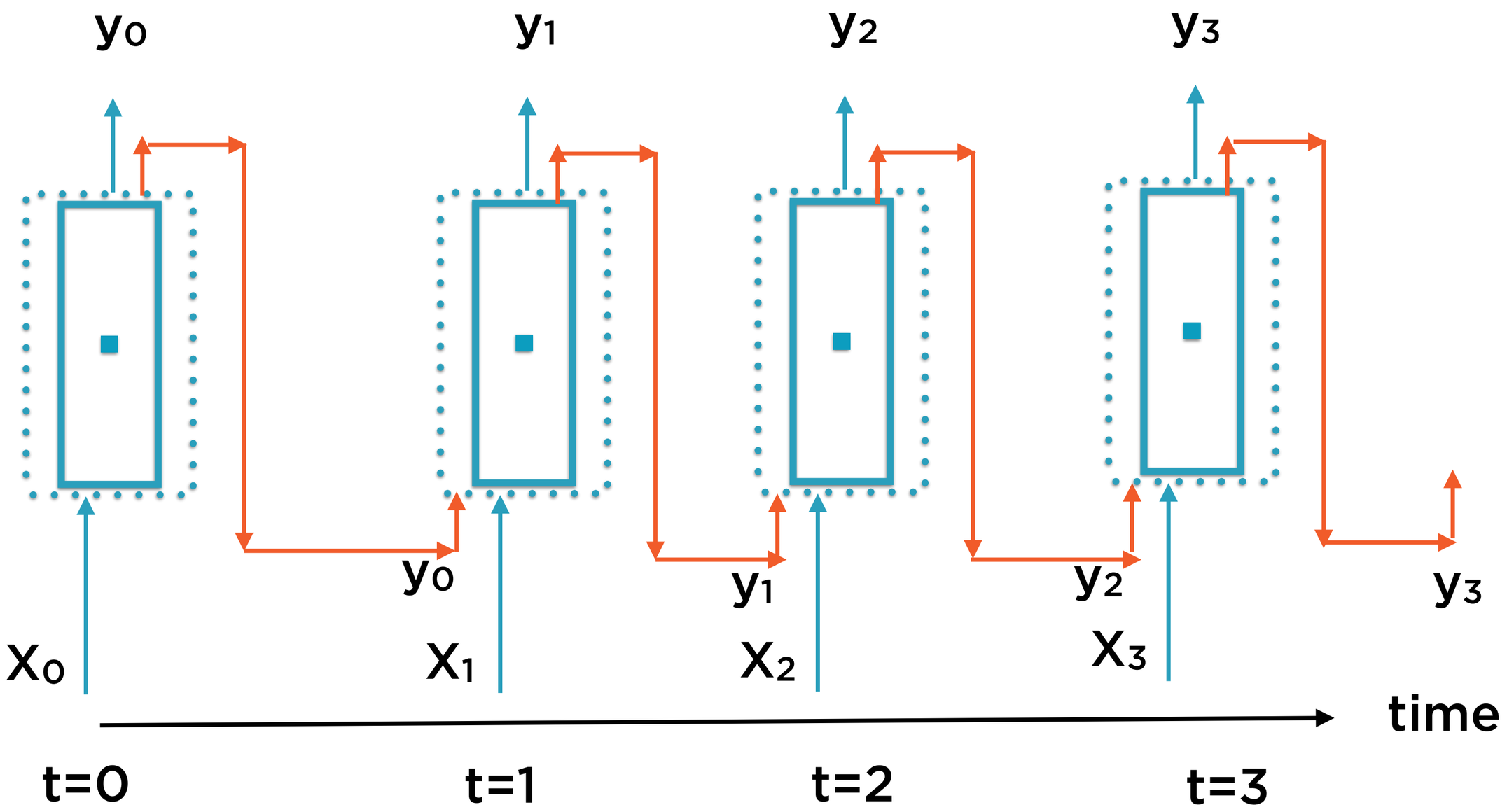
Unrolling Through Time



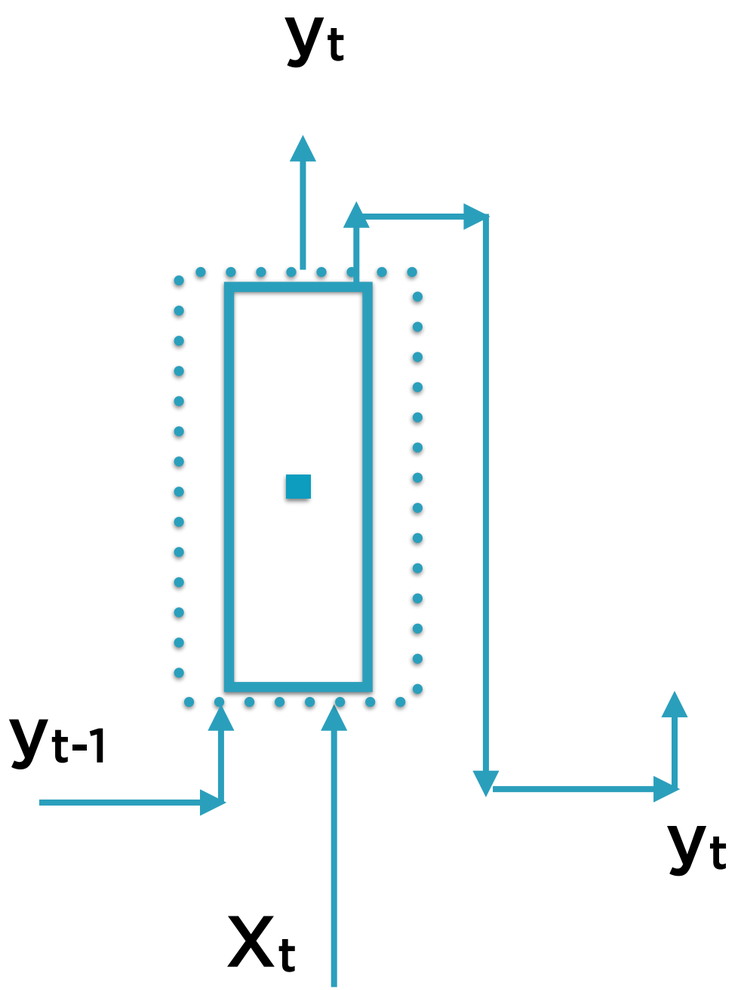
Unrolling Through Time



Unrolling Through Time



Recurrent Neuron



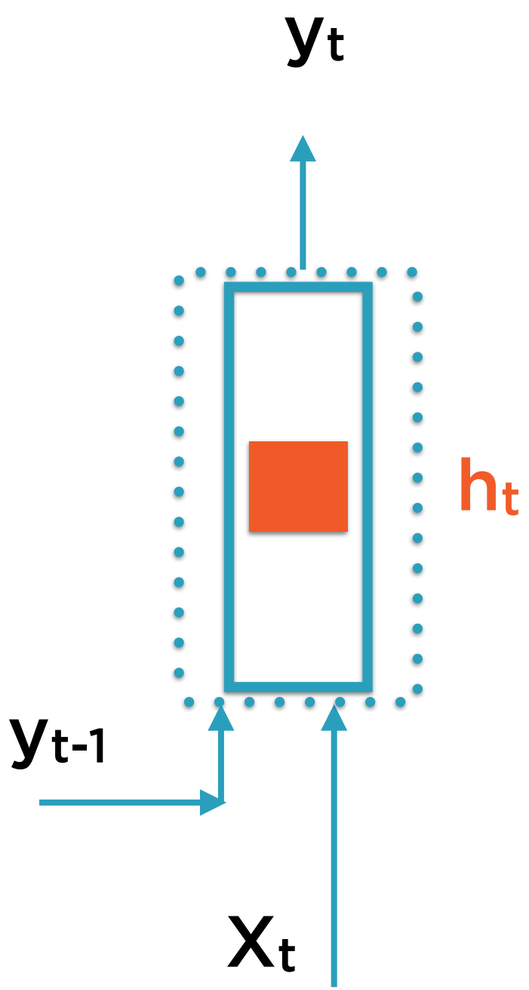
Regular neuron: input is feature vector, output is scalar

$Y = Wx + b$

Recurrent neuron: output is vector too

Input: $[X_0, X_1, \ldots X_t]$

Output: $[Y_0, Y_1, \ldots Y_t]$



Memory and State

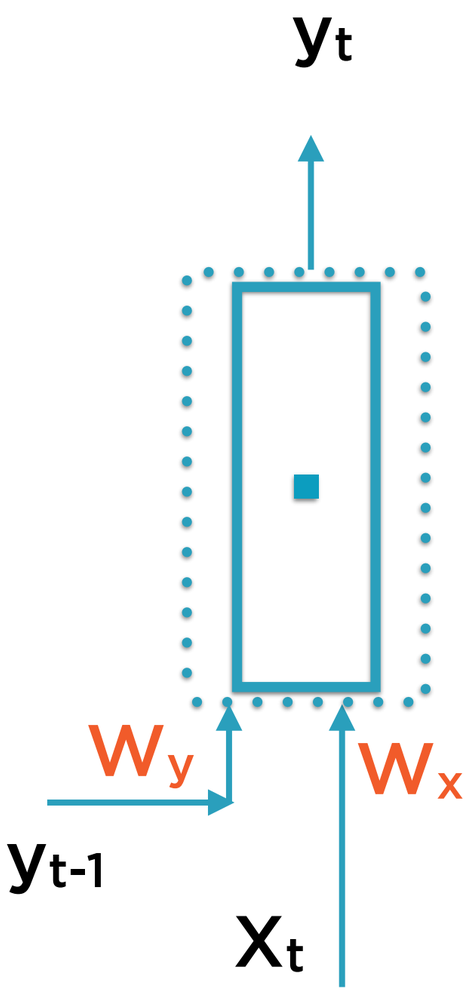
Recurrent neurons remember the past

They possess ‘memory’

The stored state could be more complex than simply y_{t-1}

The internal state is represented by h_t

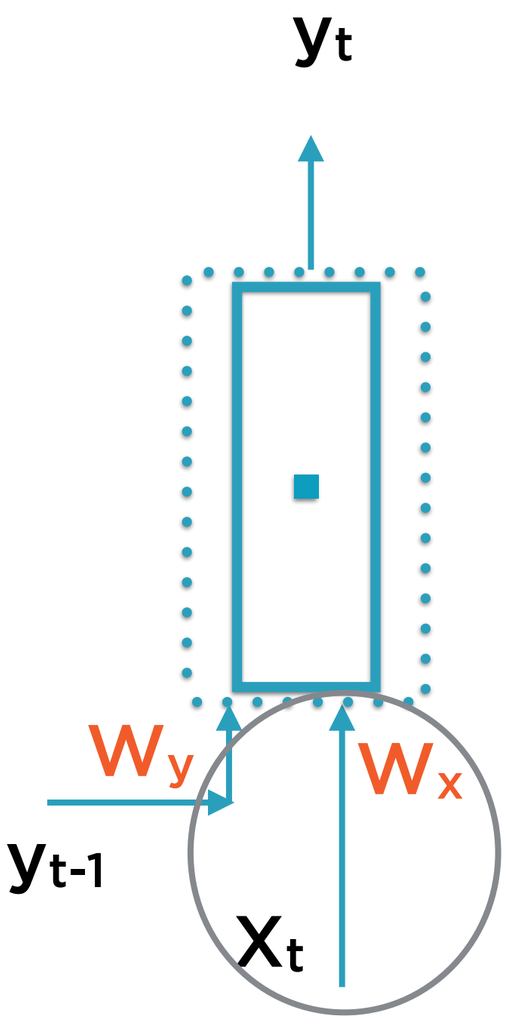
Recurrent Neuron



Now, each neuron has two weight vectors

W_x, W_y

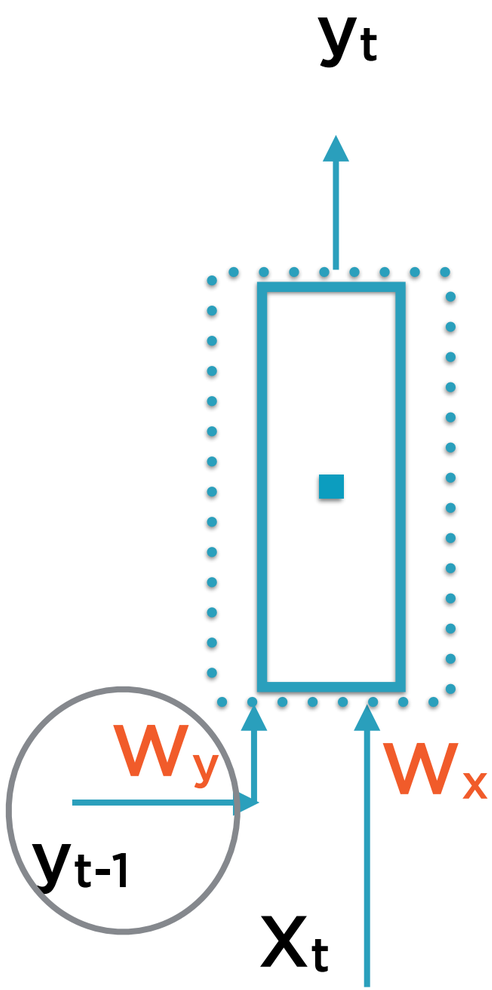
Recurrent Neuron



Now, each neuron has two weight vectors

W_x, W_y

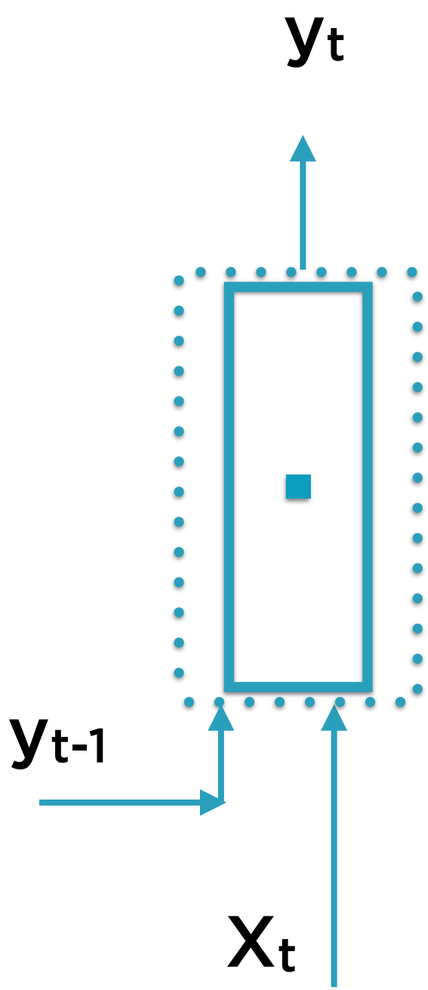
Recurrent Neuron



Now, each neuron has two weight vectors

W_x, W_y

Recurrent Neuron



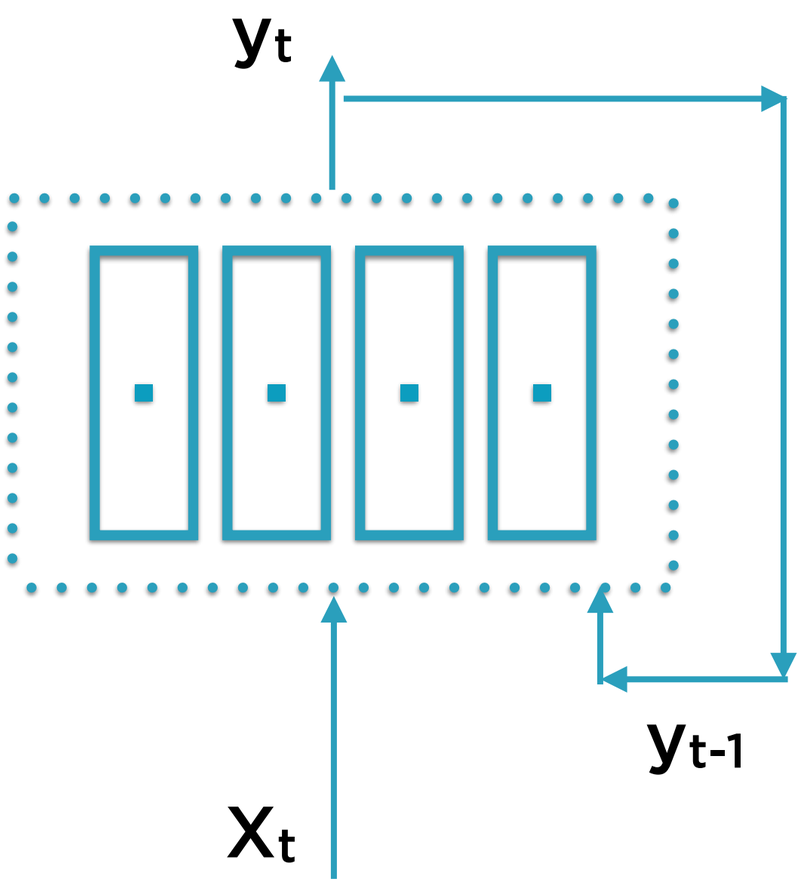
Output of neuron as a whole is given as

$$y_t = \Phi(X_t W_x + y_{t-1} W_y + b)$$

(Φ is the activation function)

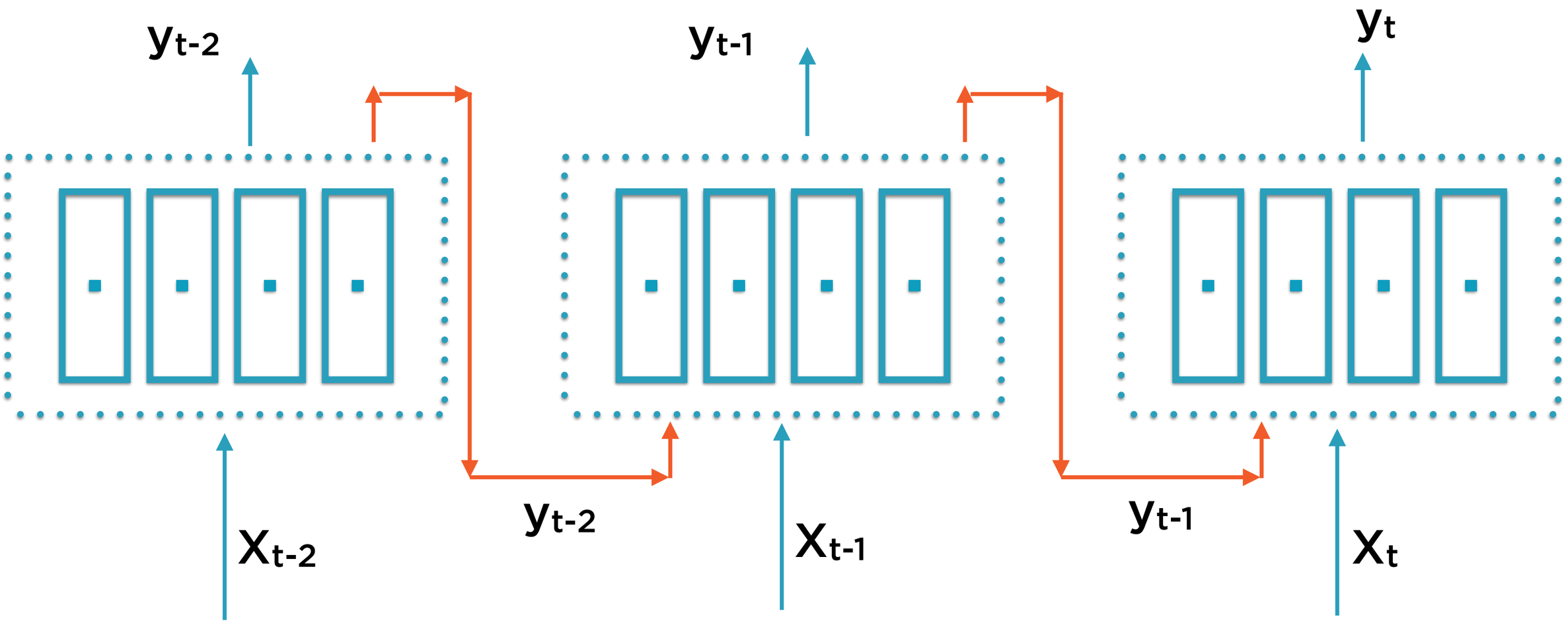
Training a Recurrent Neural Network

Layer of Recurrent Neurons



A layer of neurons forms an RNN cell - basic cell, LSTM cell, GRU cell (more on these later)

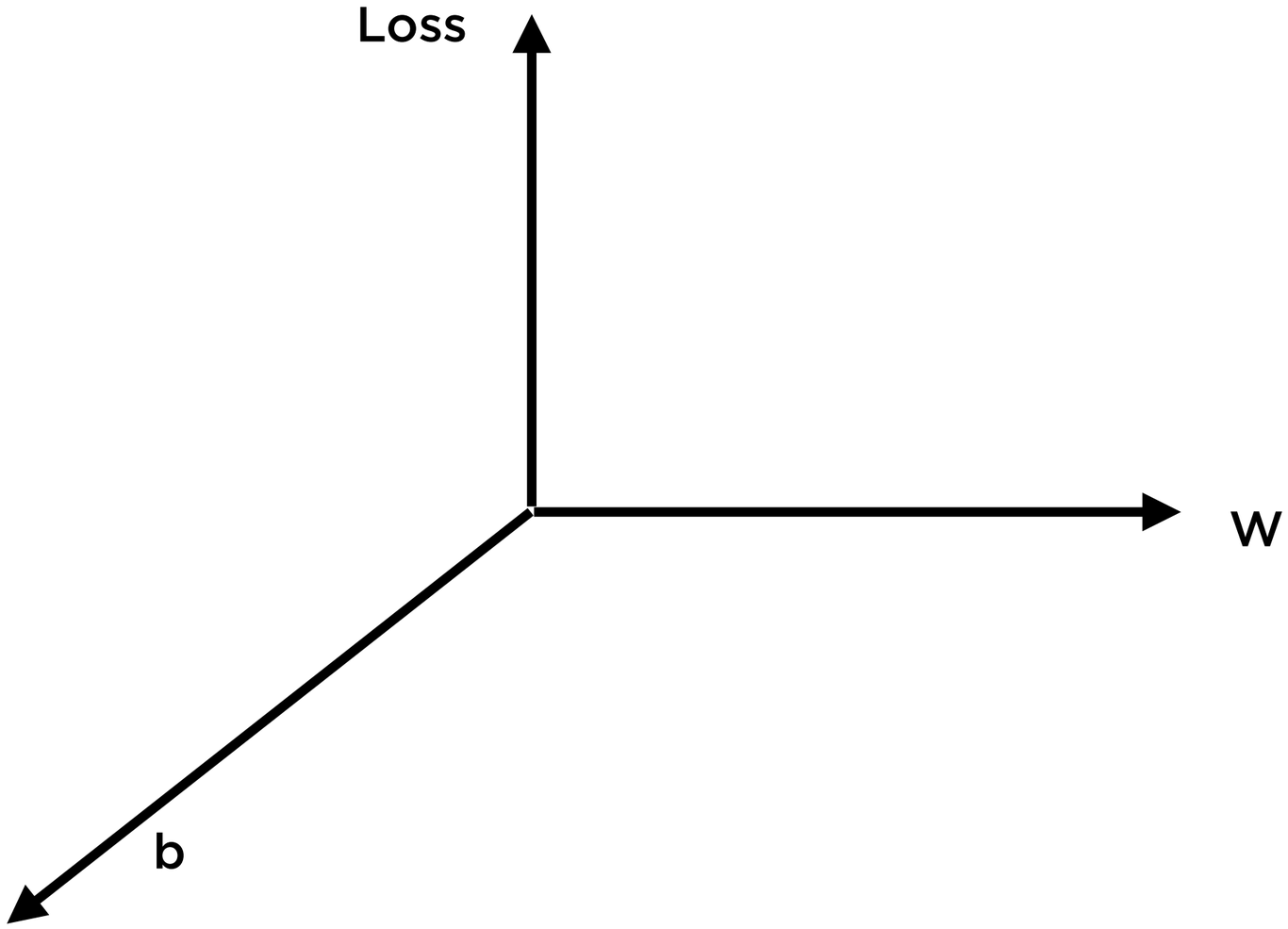
Layer of Recurrent Neurons



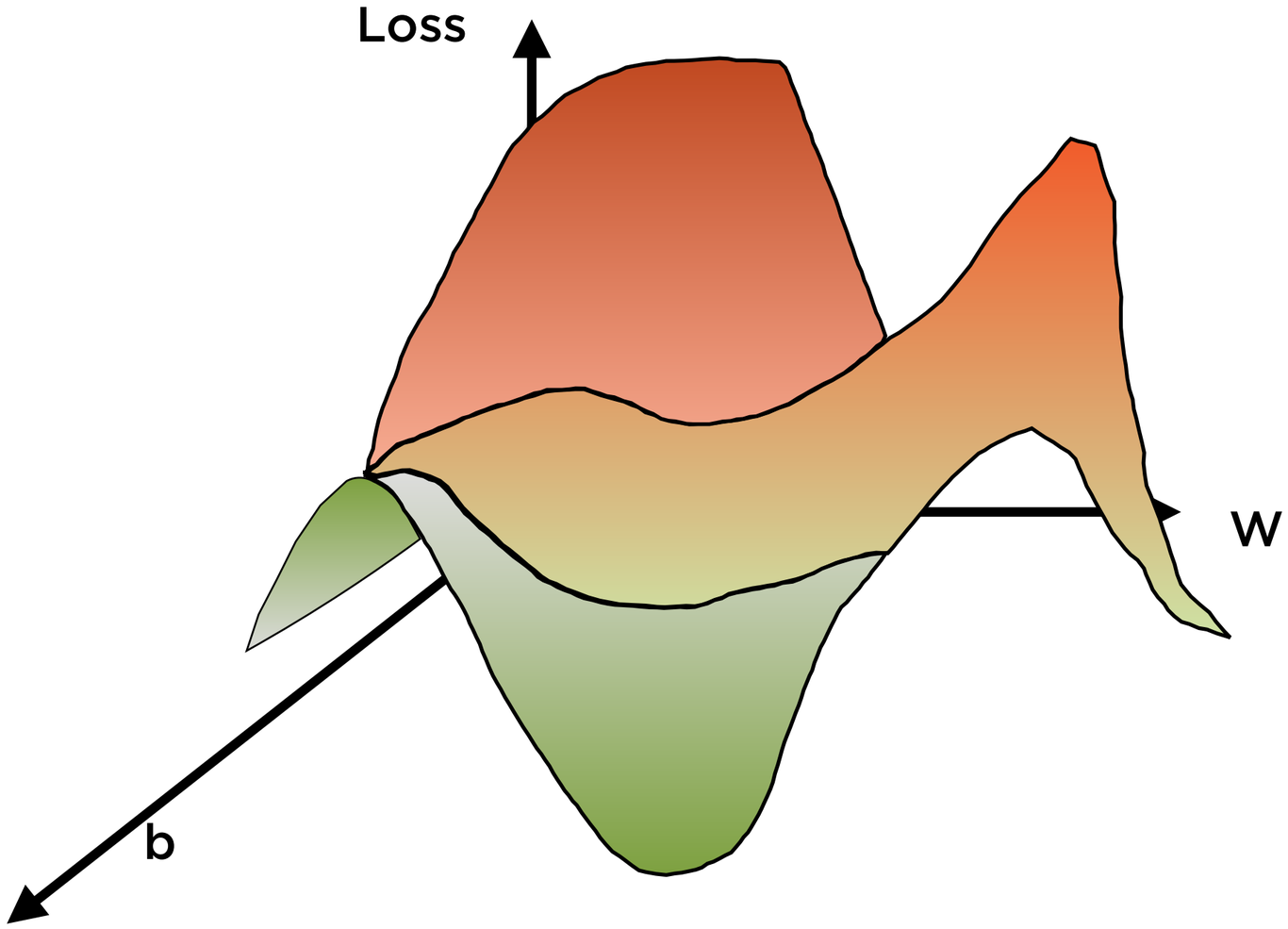
The cells unrolled through time form the layers of the neural network

The actual training of a neural network happens via Gradient Descent Optimization

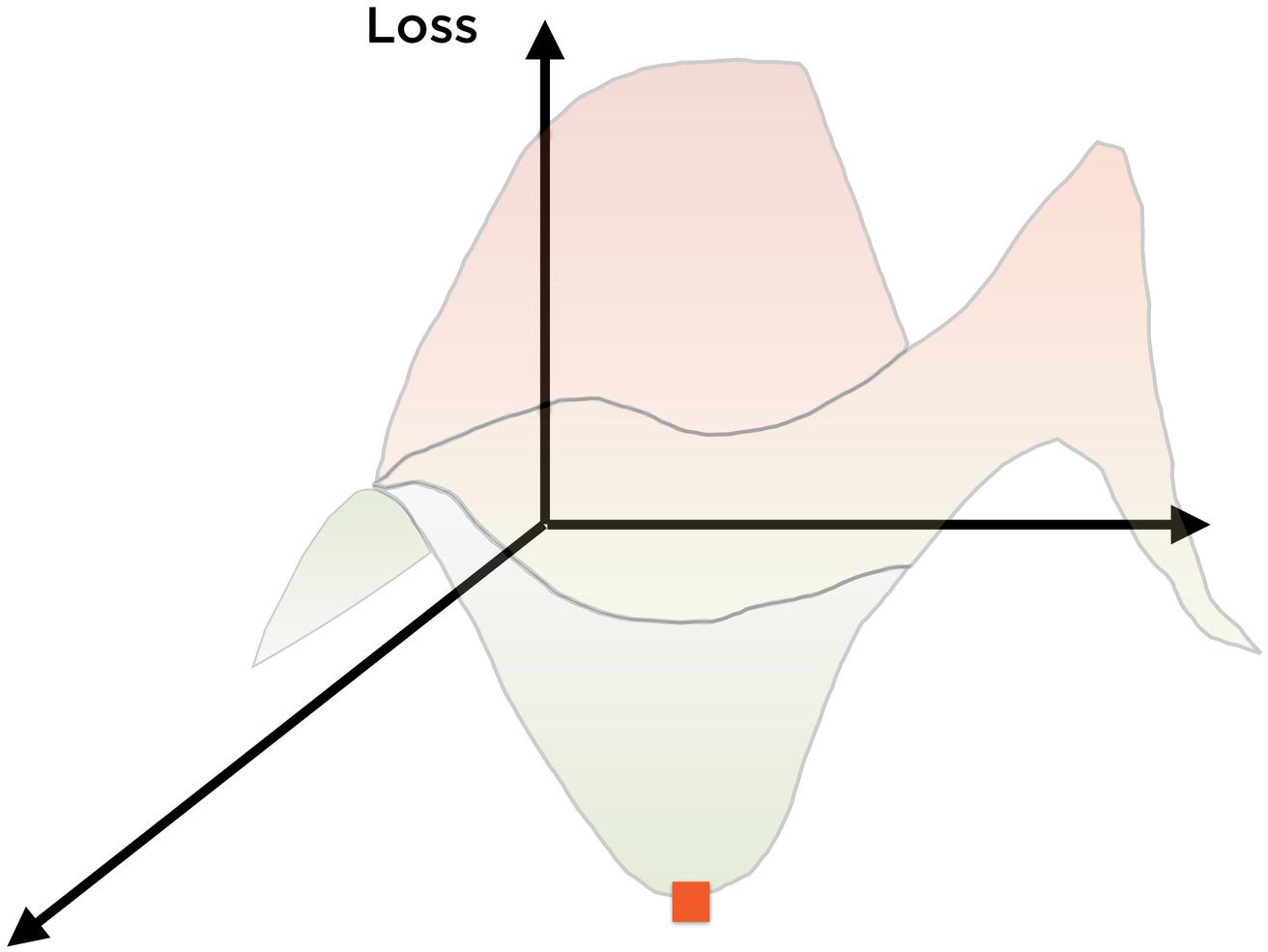
Minimizing Loss



Minimizing Loss

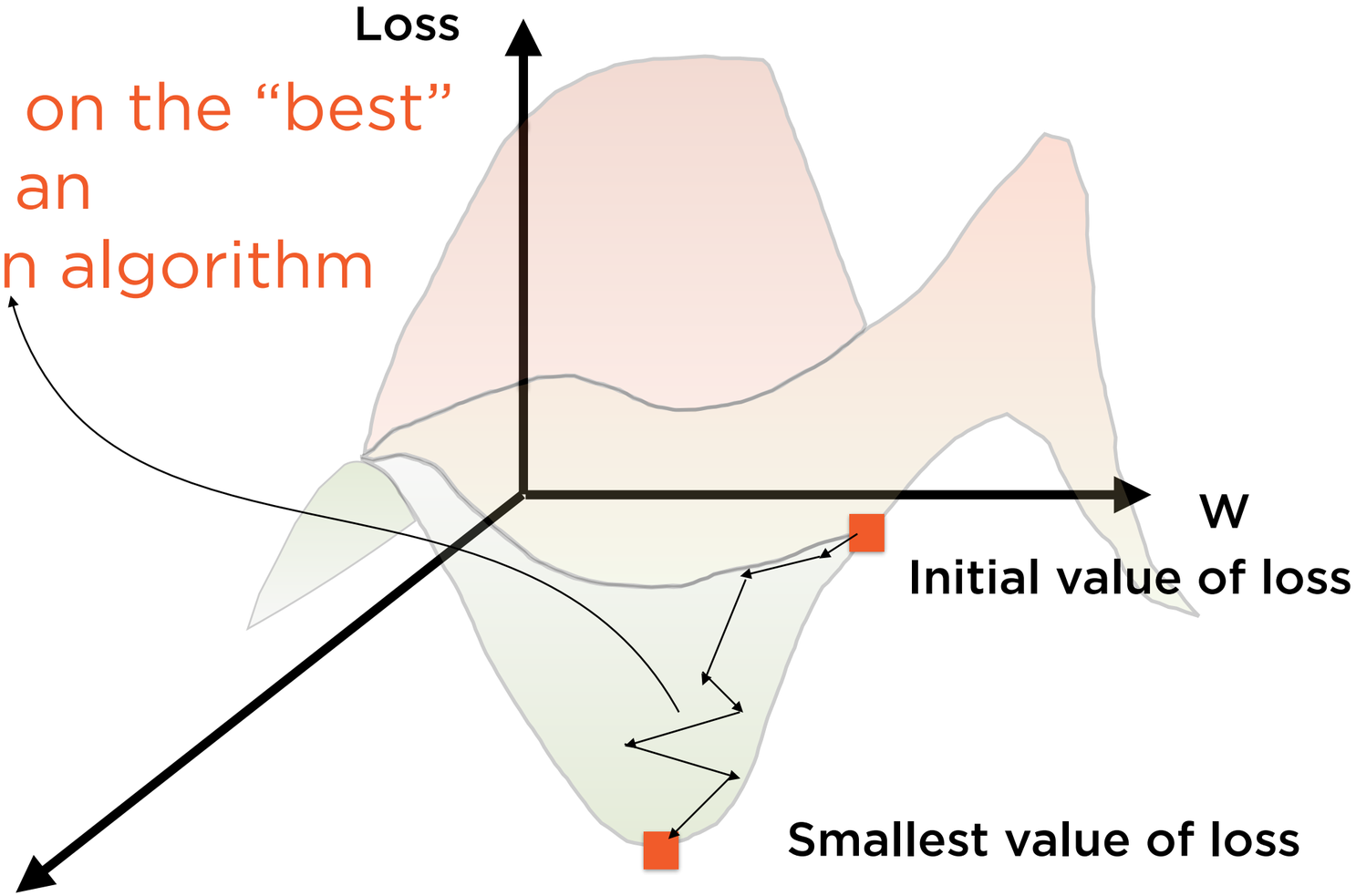


Minimizing Loss

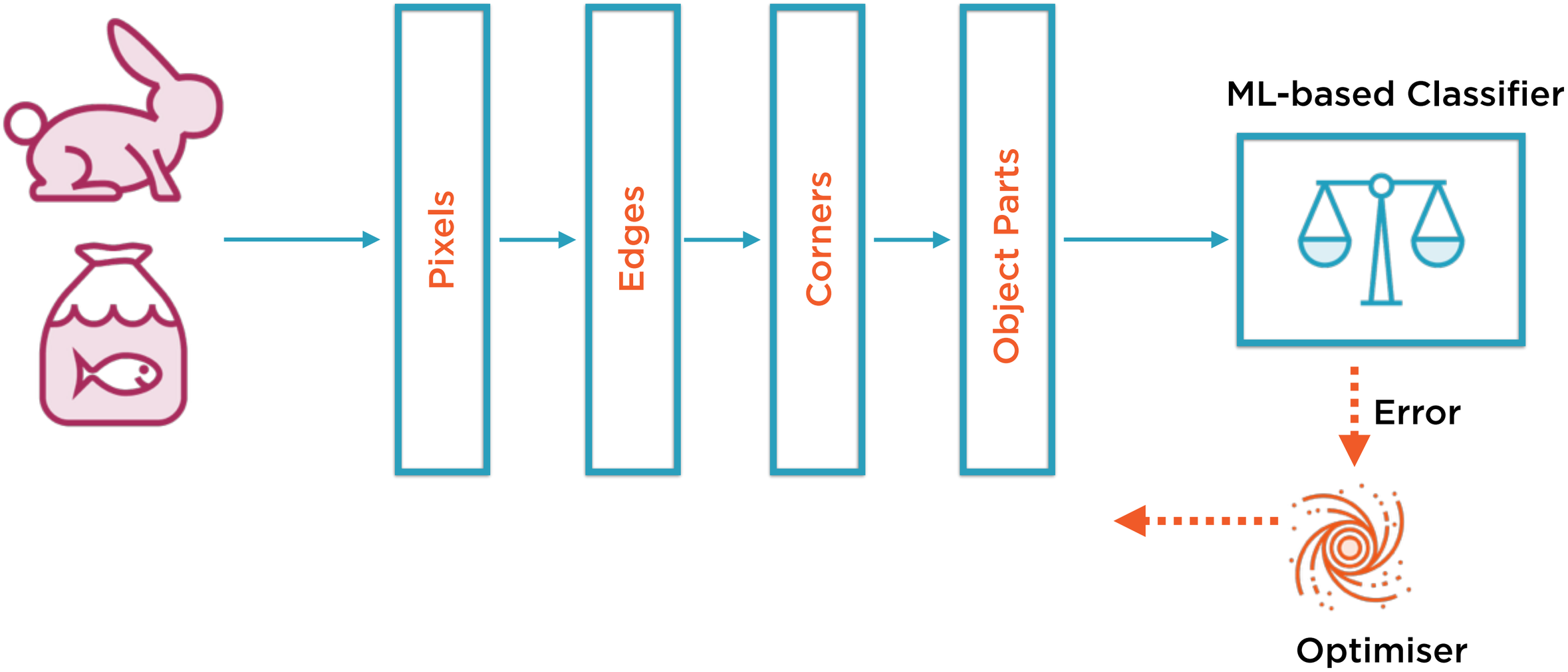


Gradient Descent

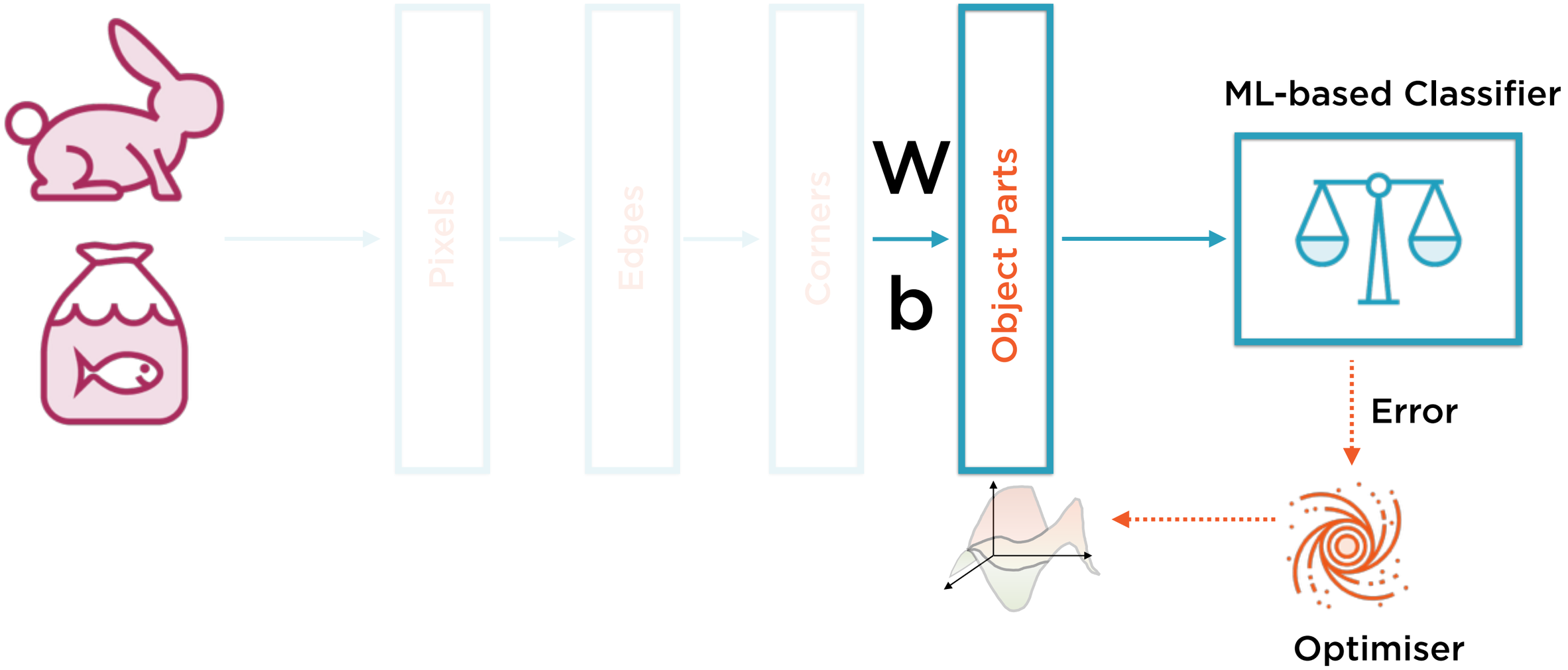
Converging on the “best” value using an optimization algorithm



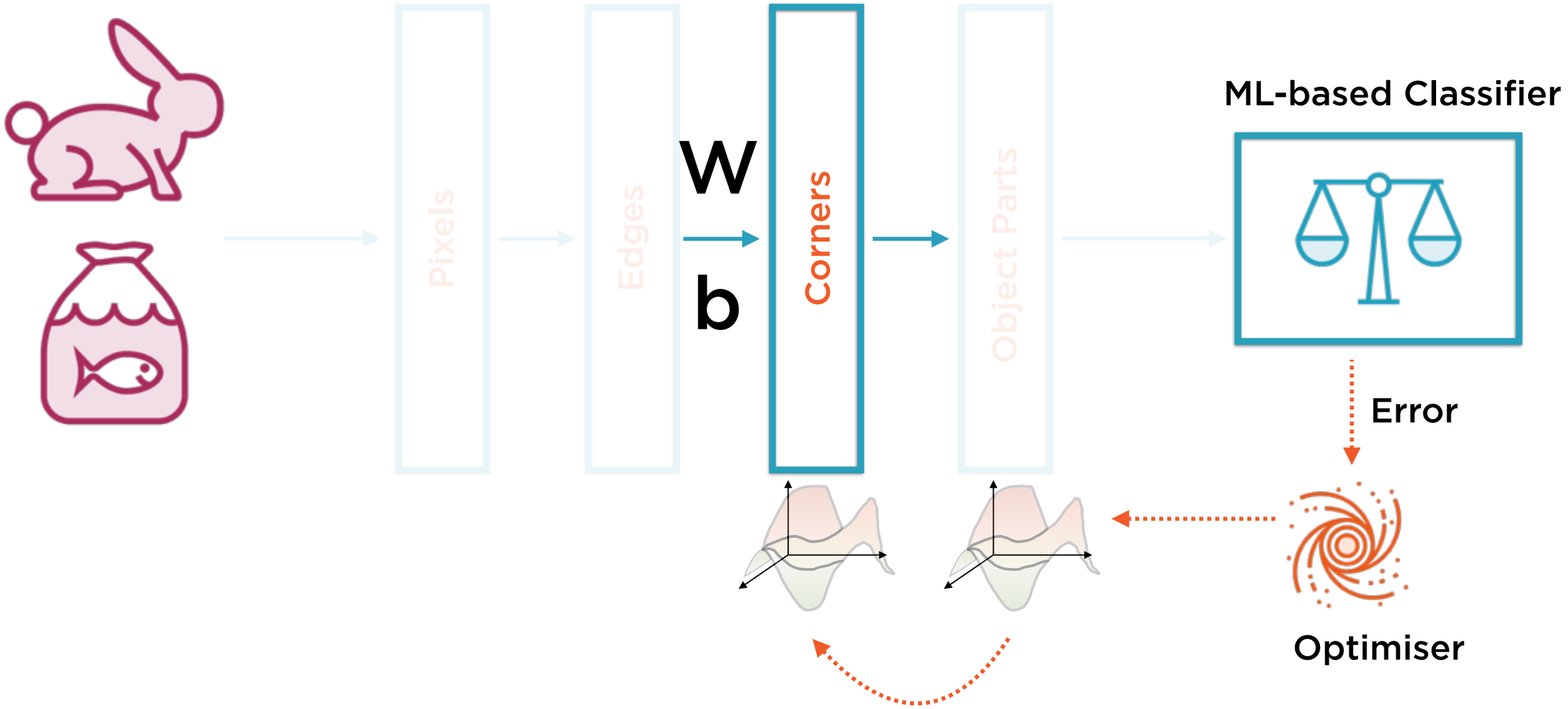
Back Propagation Through Time



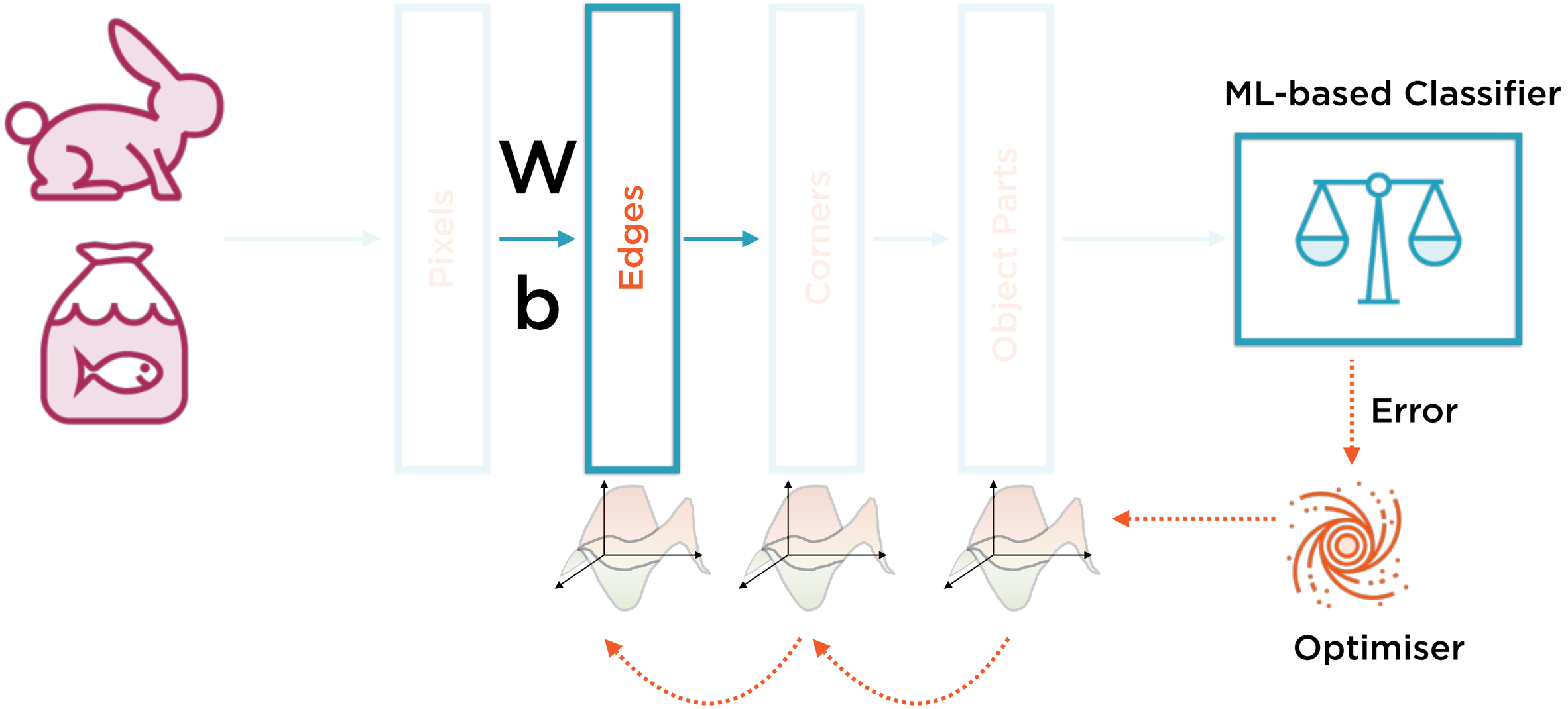
Back Propagation Through Time



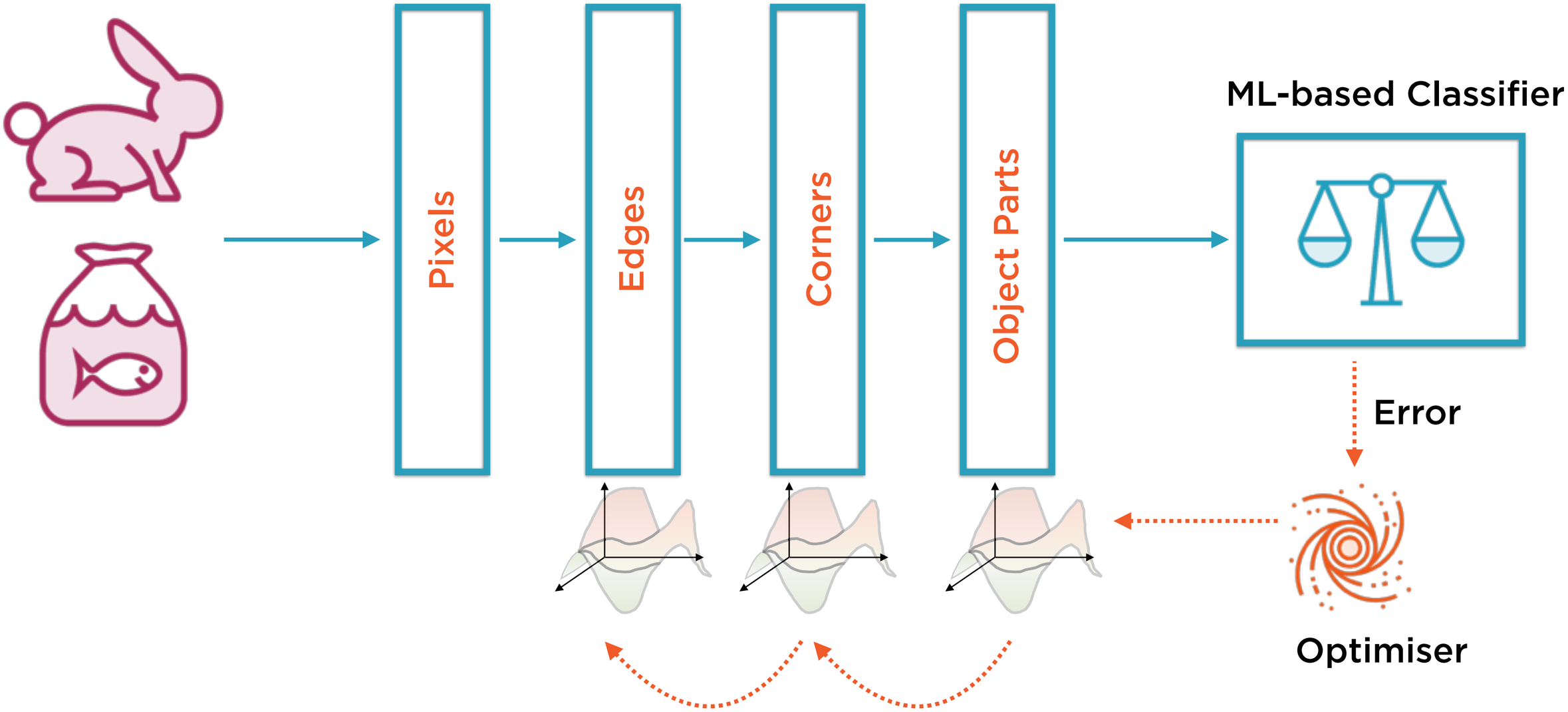
Back Propagation Through Time



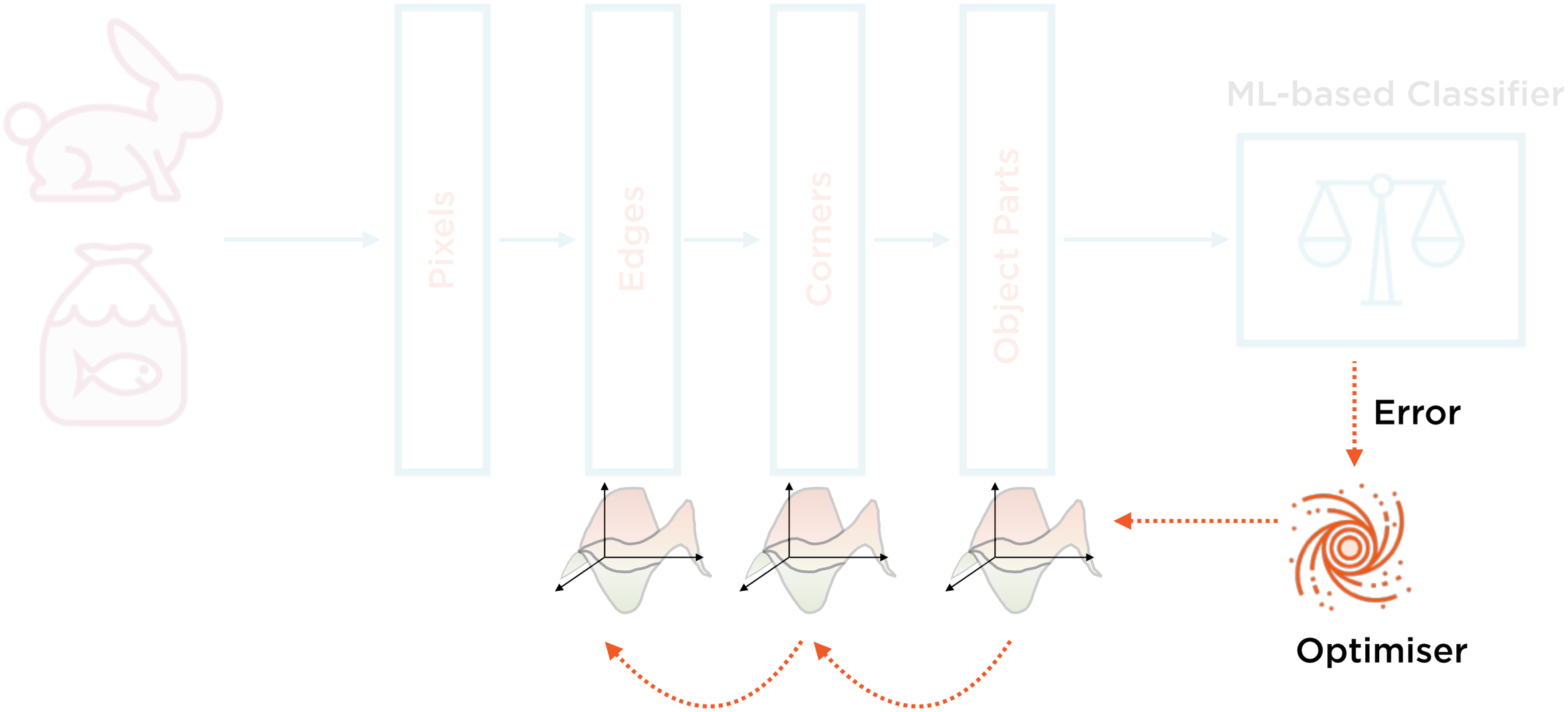
Back Propagation Through Time



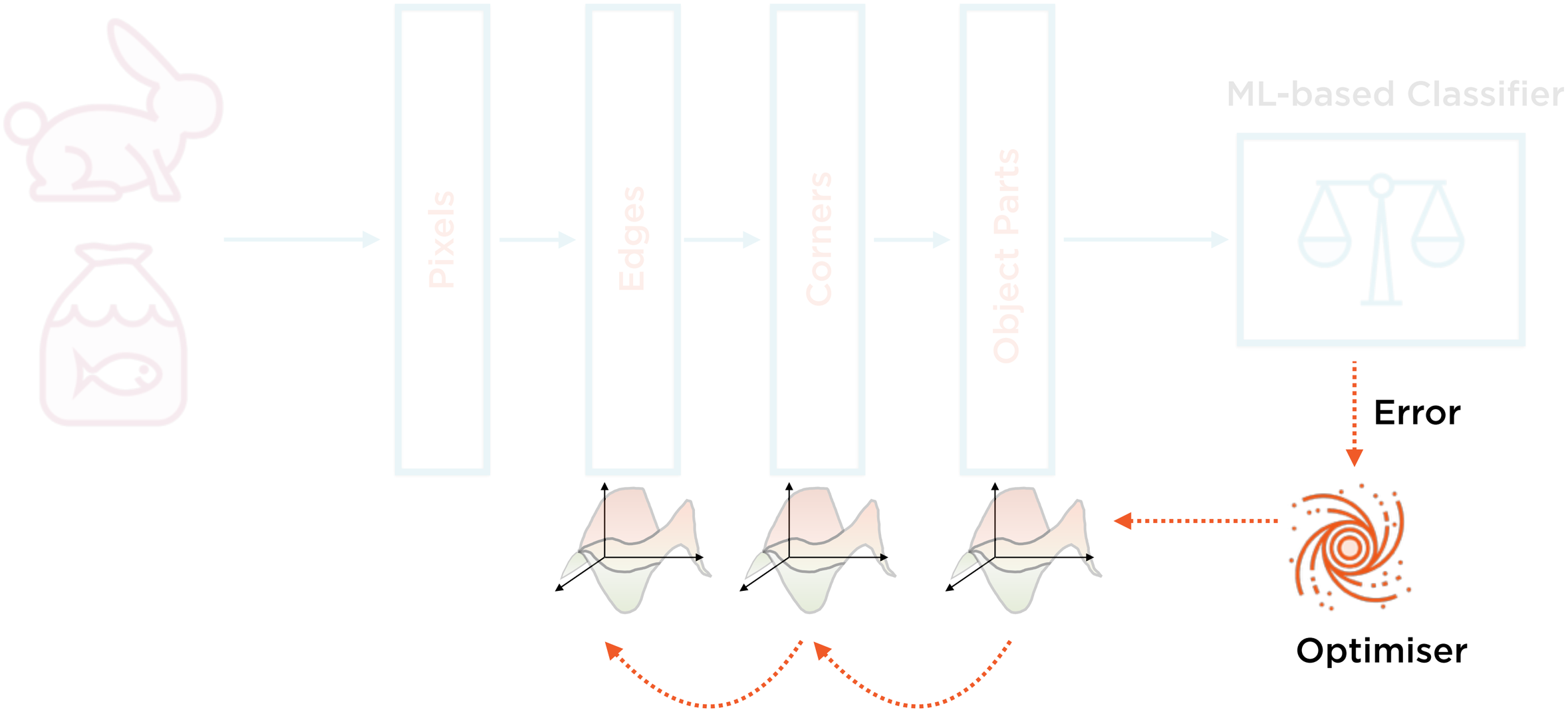
Back Propagation Through Time



Back Propagation Through Time



Back Propagation Through Time



Back propagation allows the weights and biases of the neurons to converge to their final values

$$y_t = f(x_t, y_{t-1}, y_{t-2} \ldots, y_{t-1000})$$

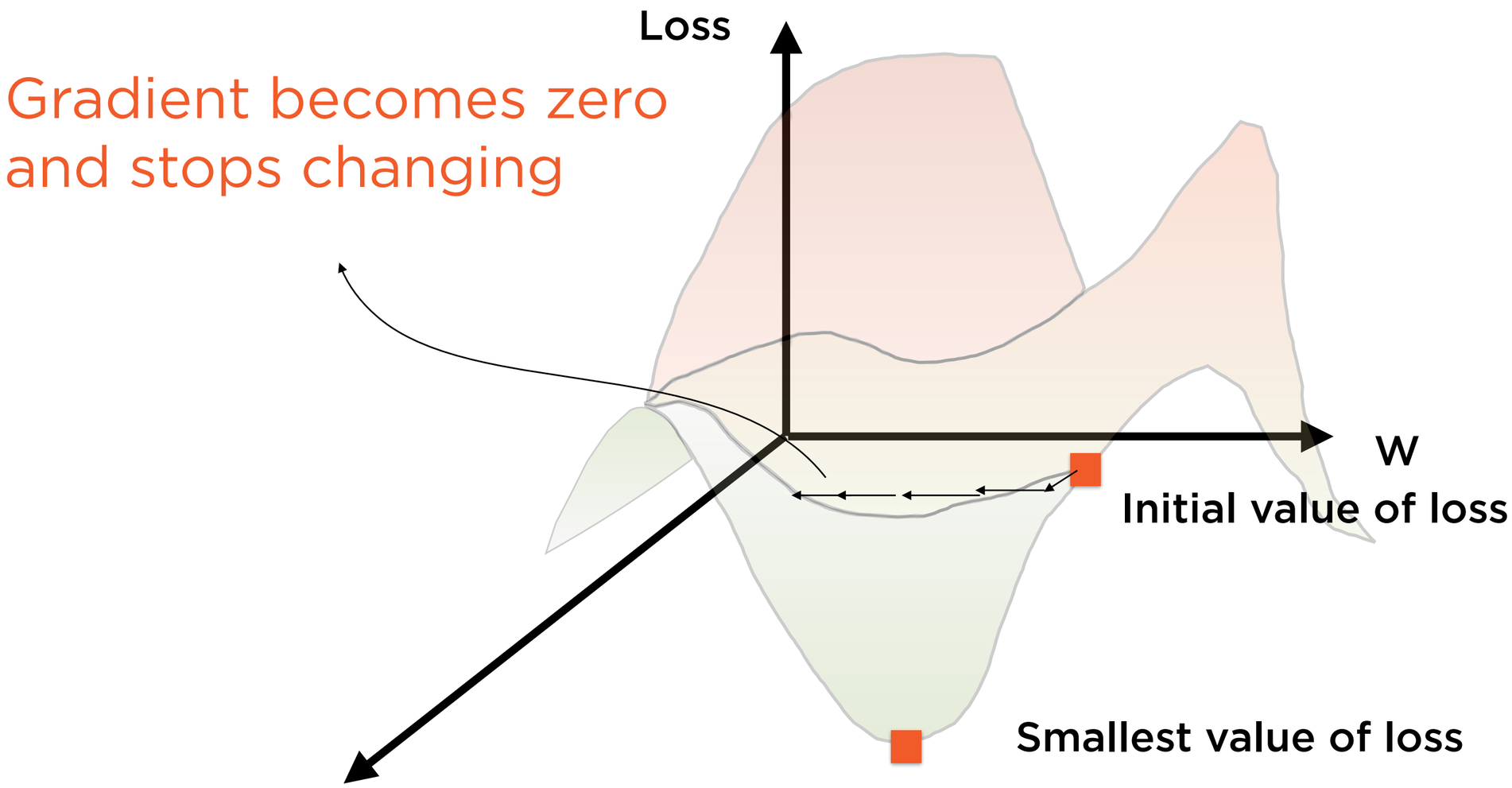
Learning the Distant Past

The unrolled RNN will be very, very deep - many layers to train, the gradient has to be propagated a long way

Recurrent neural networks may be unrolled **very far back in time**

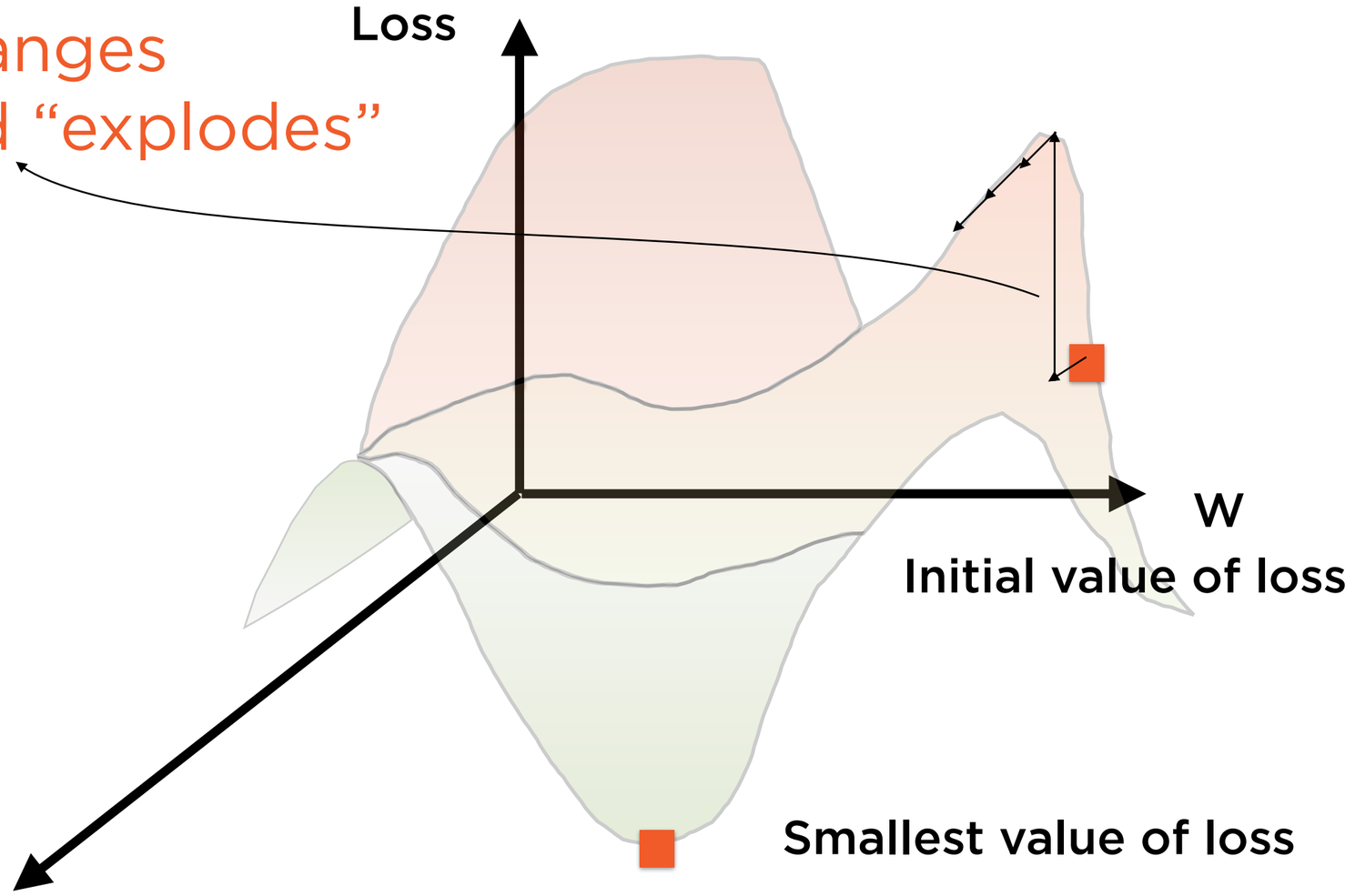
They're prone to the **vanishing** and **exploding** gradients issue

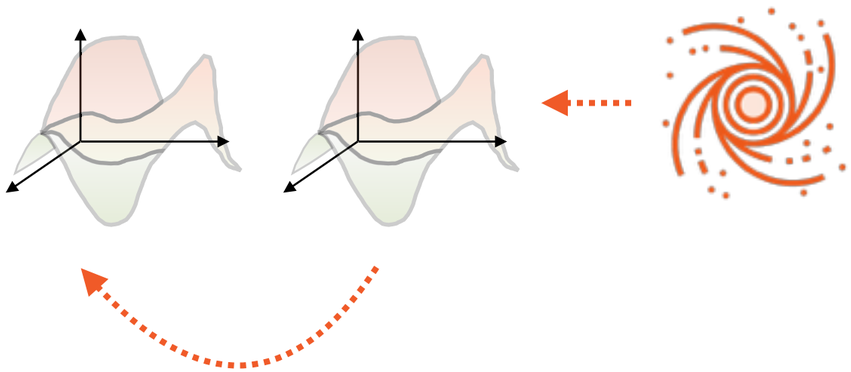
Vanishing Gradient Problem



Exploding Gradient Problem

Gradient changes abruptly and "explodes"





Vanishing and Exploding Gradients

Back propagation fails if:

- **gradients are vanishing**
- **gradients are exploding**

Coping with Vanishing/Exploding Gradients

Proper initialisation

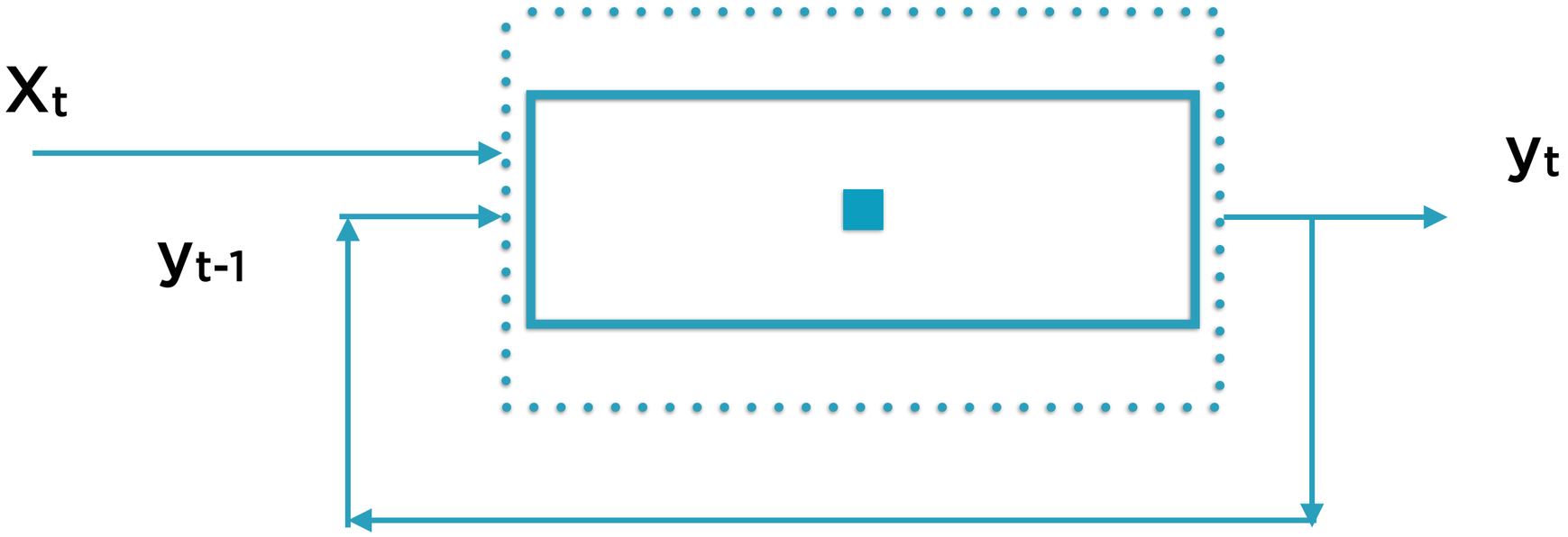
Non-saturating activation function

Batch normalisation

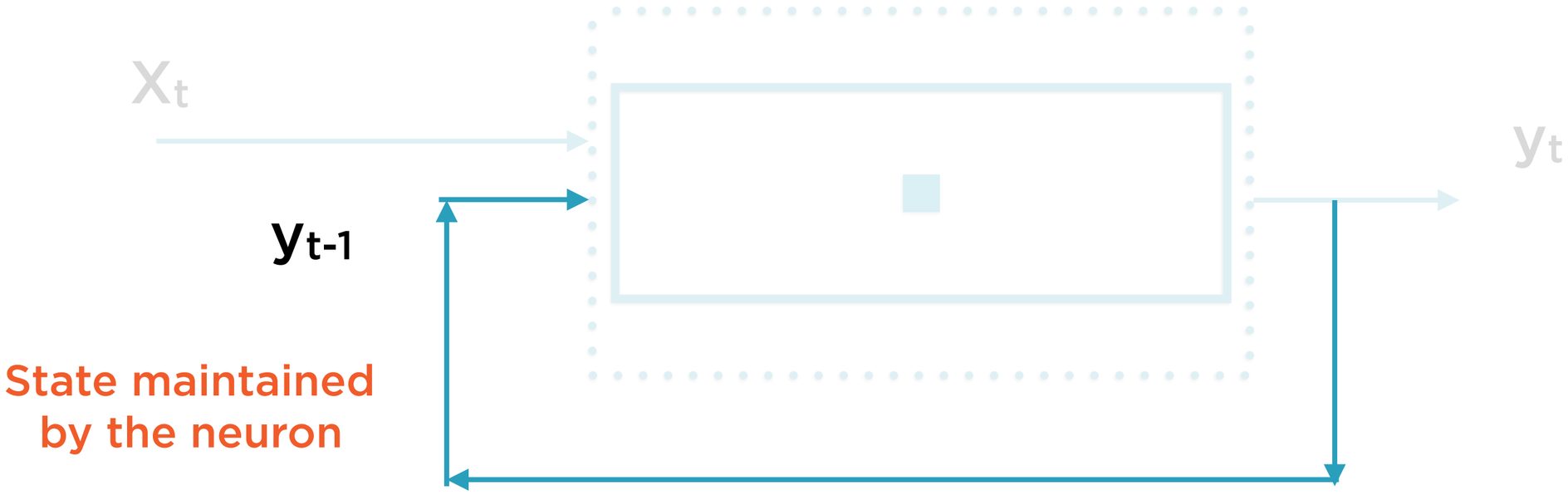
Gradient clipping

Long Memory RNN Cells

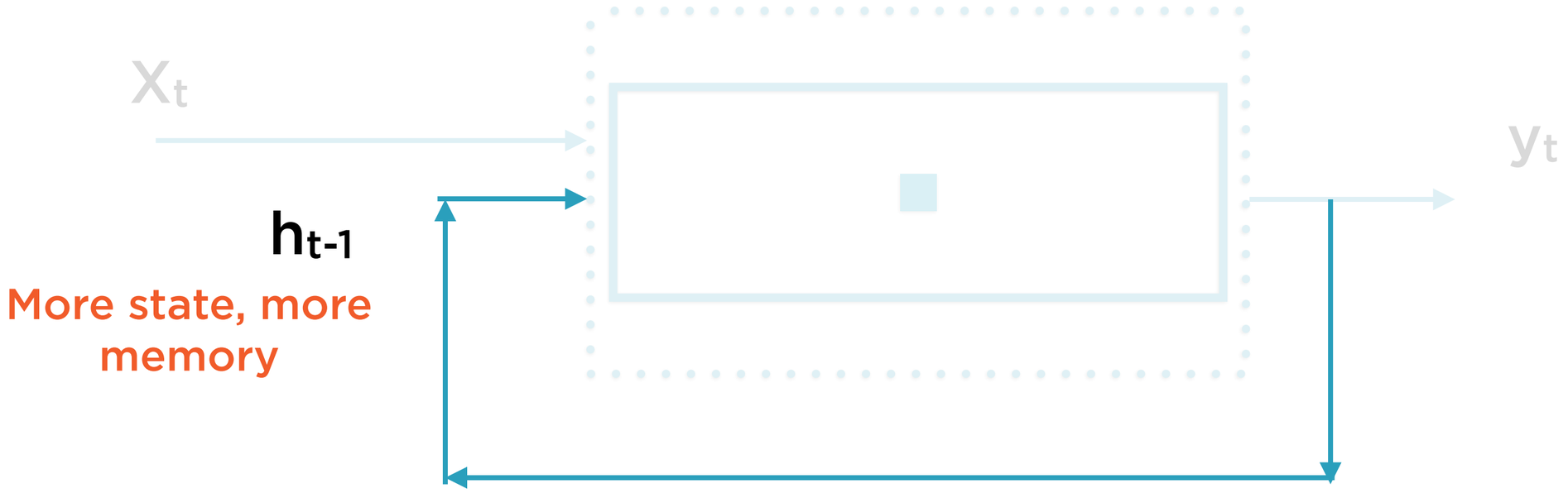
Simplest Recurrent Neuron

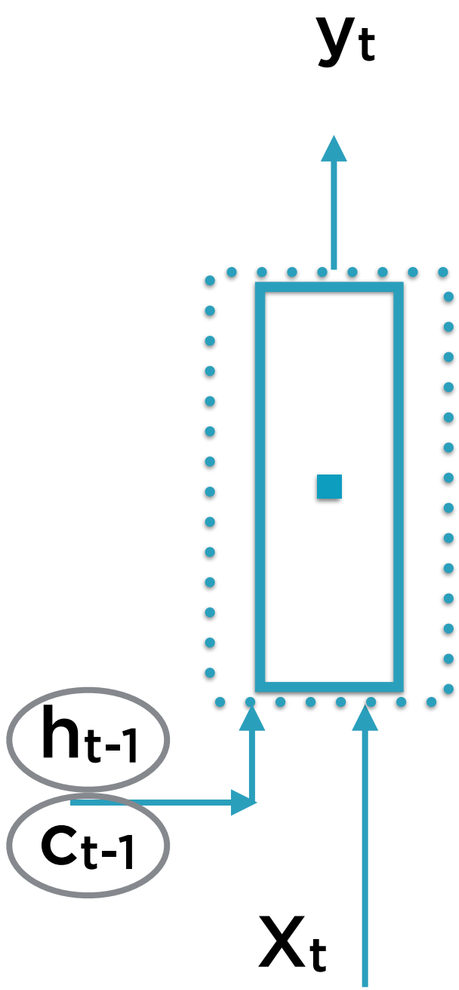


Simplest Recurrent Neuron



Long Memory Recurrent Neuron





Long Memory RNNs

Increase the amount of state in neuron

Effect is to increase memory of neuron

Could explicitly add:

- long-term state (c)
- short-term state (h)

Long Memory RNNs

Advantages in Training

Faster training, nicer gradients

Advantages in Prediction

No need to truncate BPTT

Long-term Dependencies in Text

The sky is cloudy, it looks like_____

The gap between the relevant information needed to predict the next word is small

Long-term Dependencies in Text

I've lived in France a long time. I first went there as a tourist, then applied for a job and I've been here ever since. I now speak fluent _____

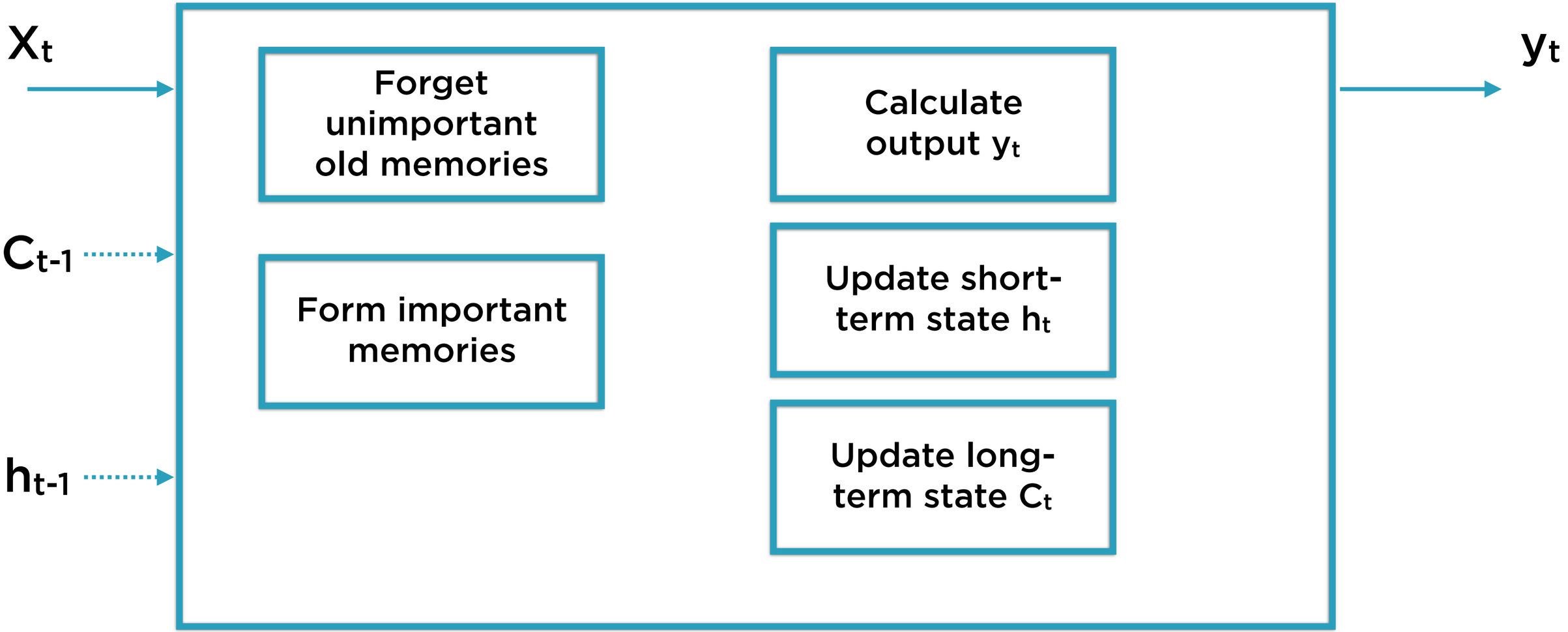
The context for which language to predict here is much farther back in the sentence

Long-term Dependencies in Text

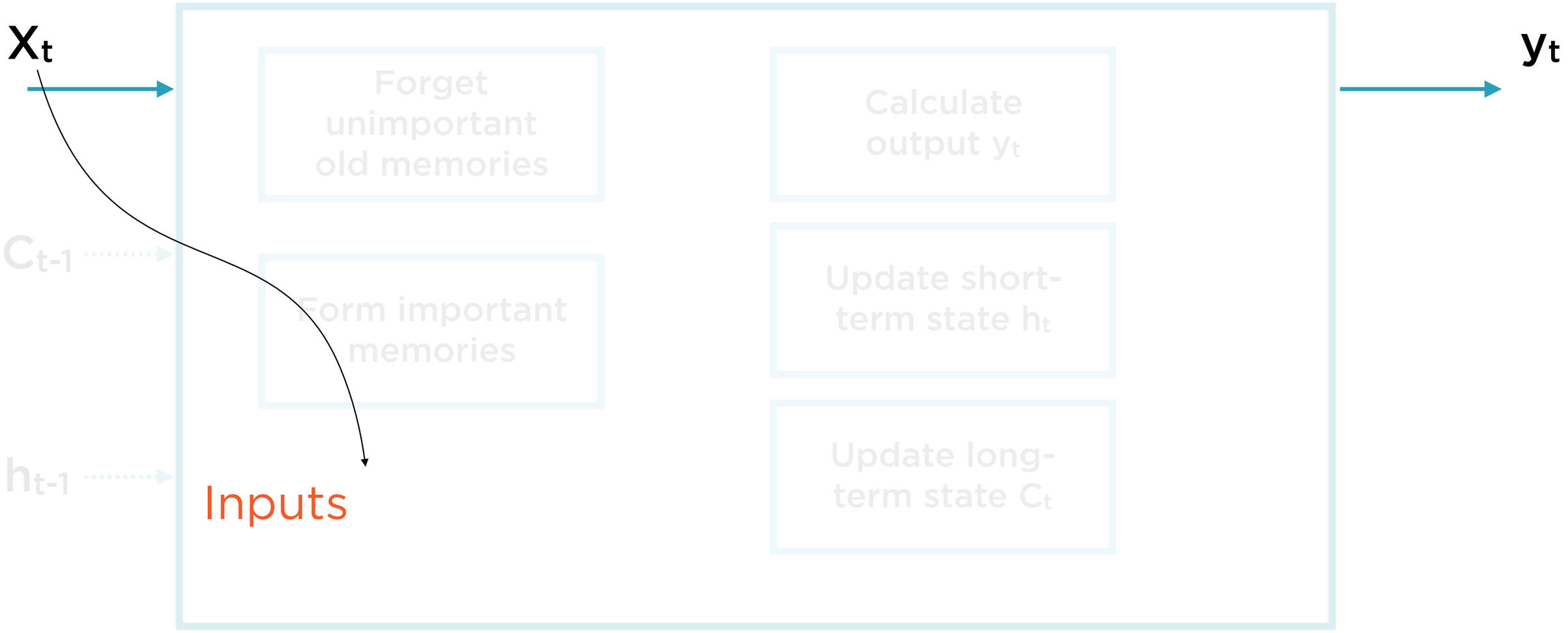
I've lived in France a long time. I first went there as a tourist, then applied for a job and I've been here ever since. I now speak fluent _____

As this gap grows, RNNs become unable to learn to connect the information

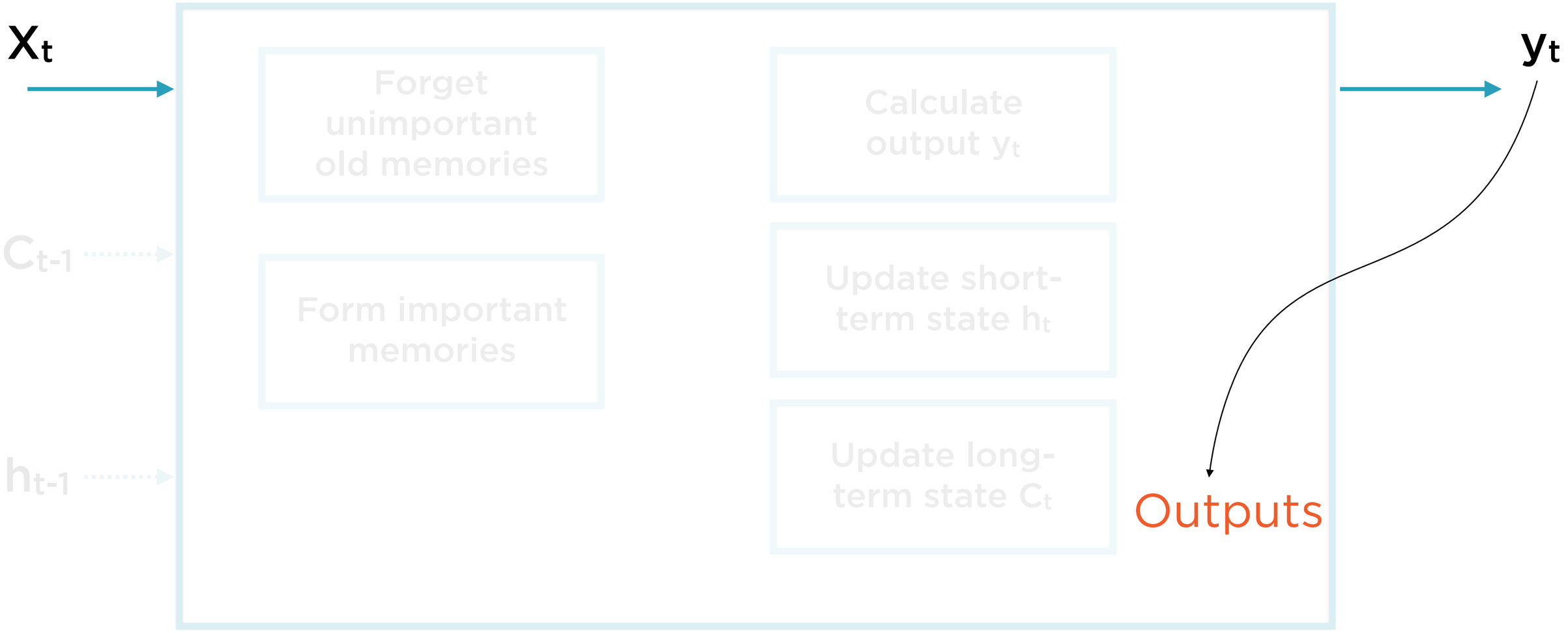
Long/Short-Term Memory Cell (LSTM)



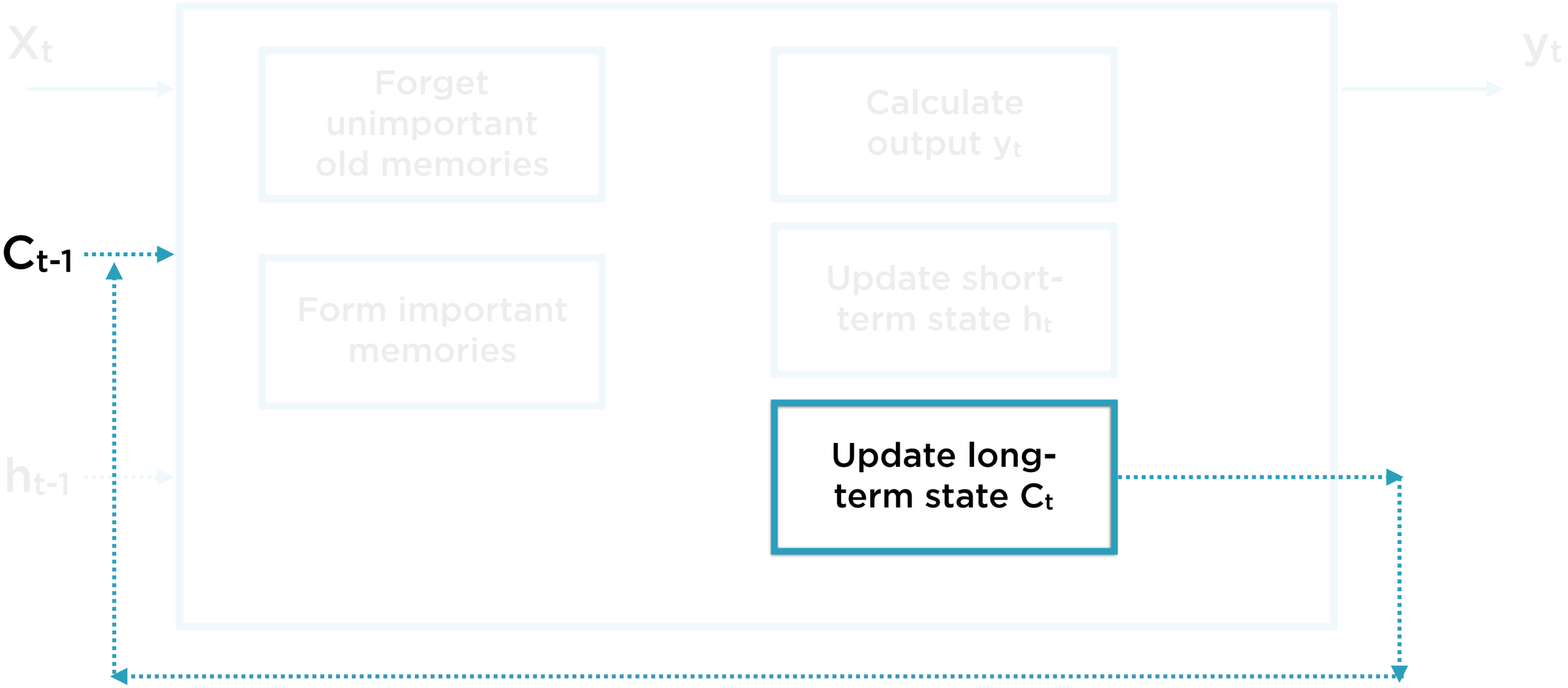
Long/Short-Term Memory Cell (LSTM)



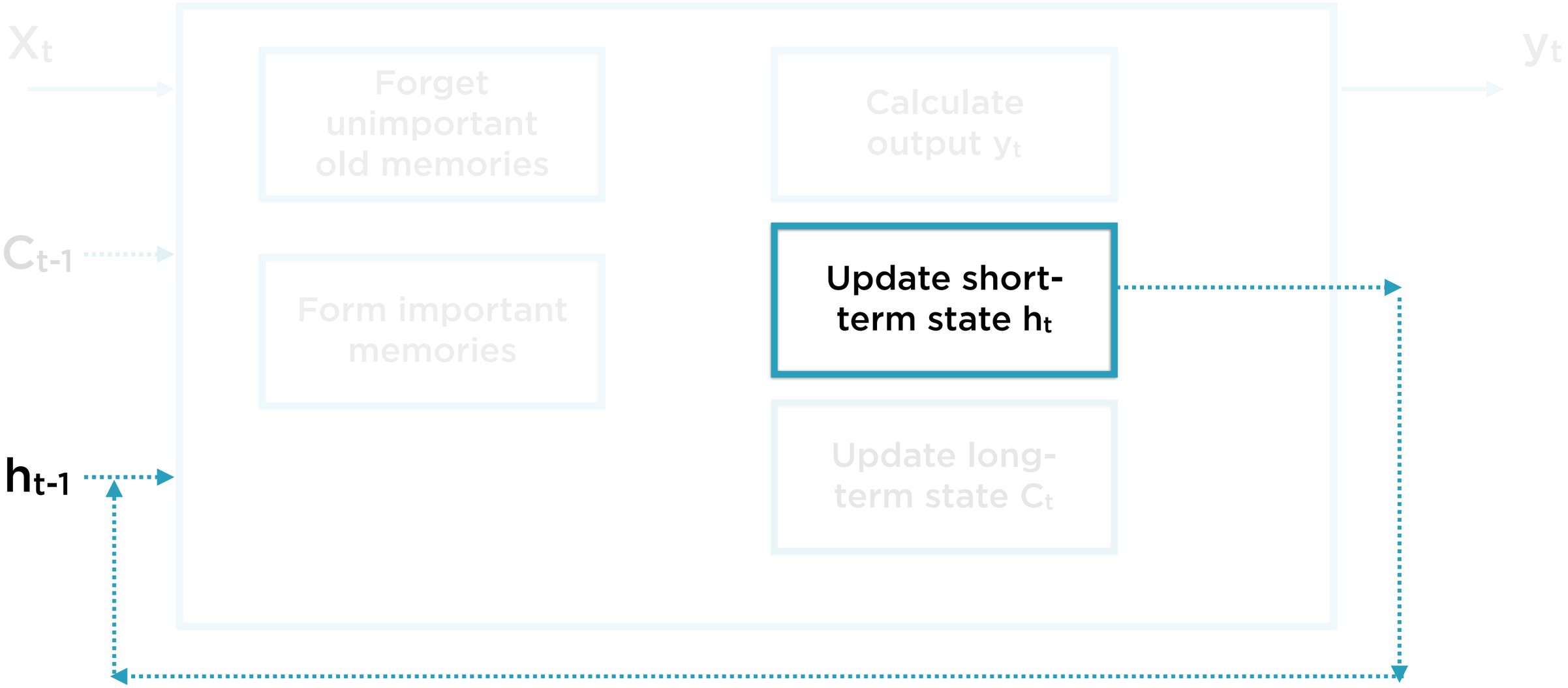
Long/Short-Term Memory Cell (LSTM)



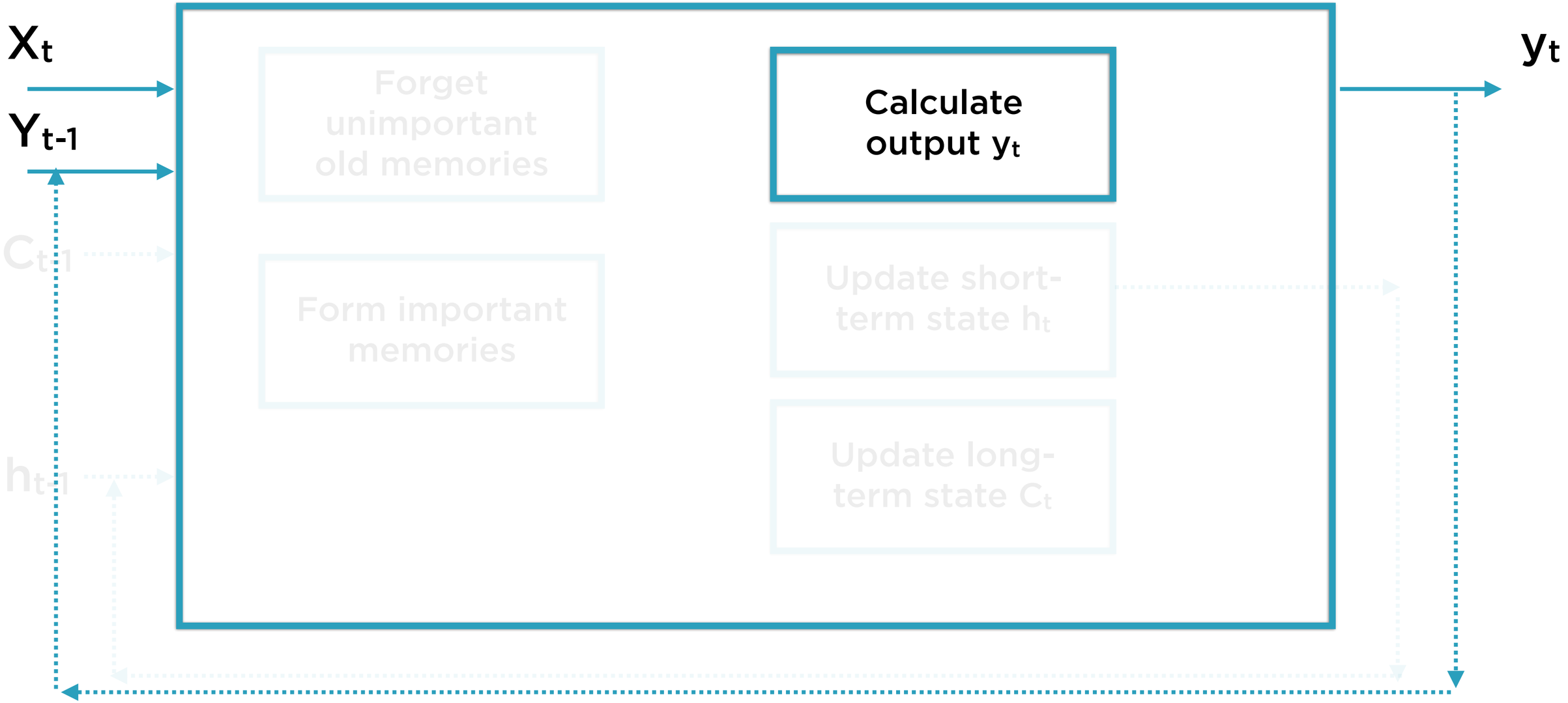
Long/Short-Term Memory Cell (LSTM)



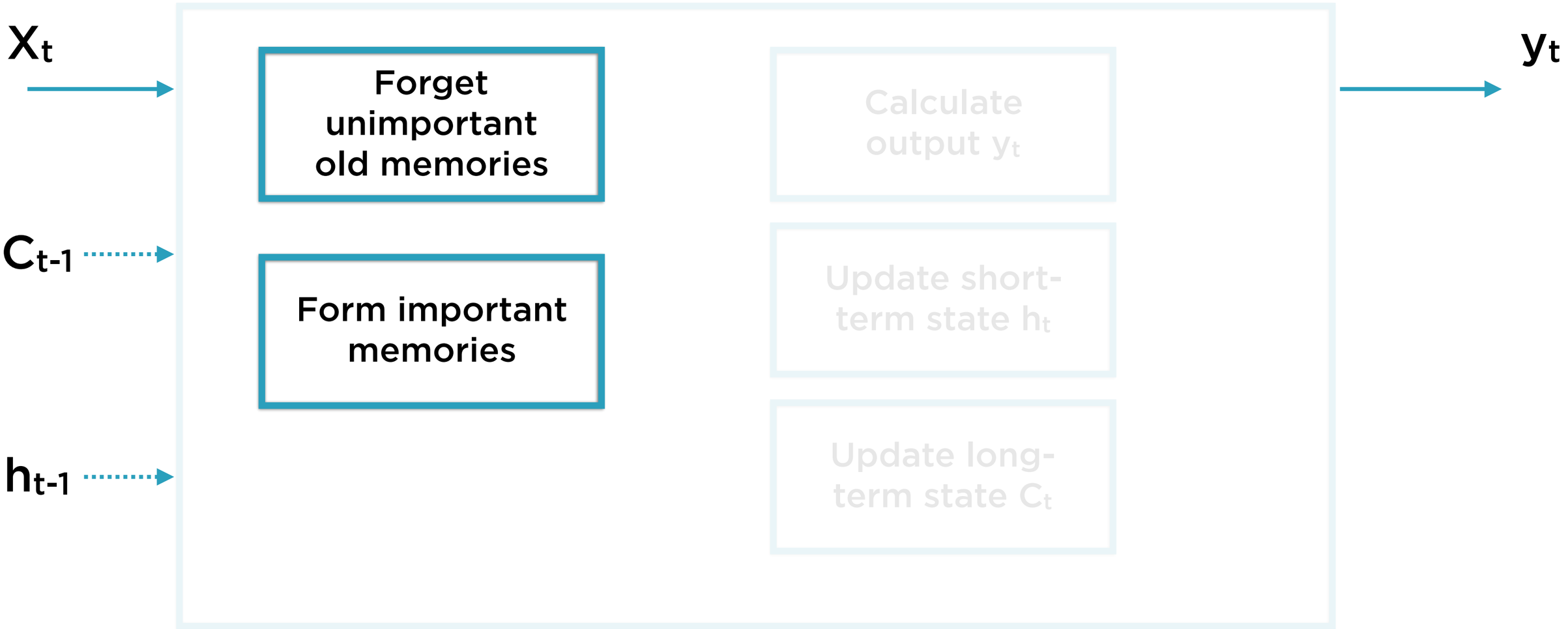
Long/Short-Term Memory Cell (LSTM)

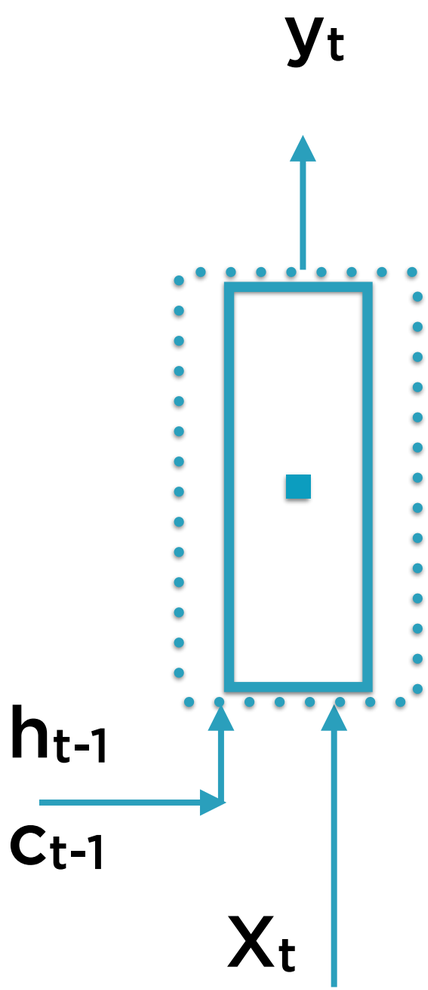


Basic RNN Cell



Long/Short-Term Memory Cell (LSTM)





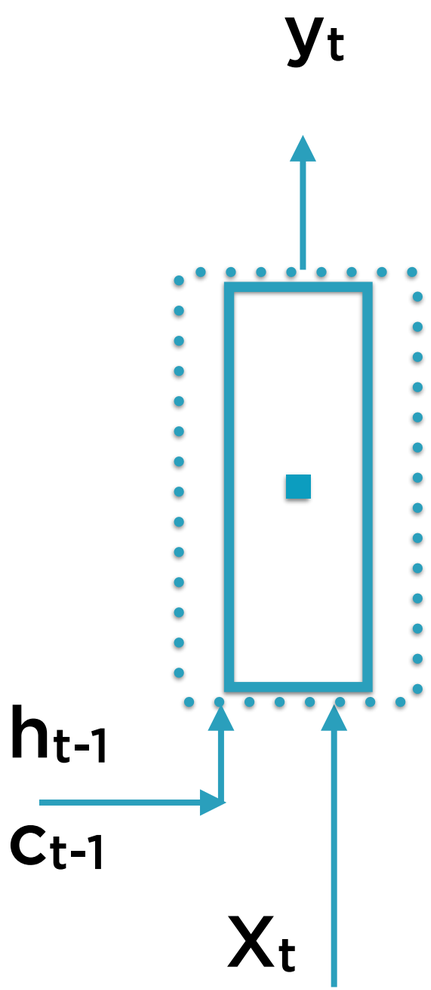
LSTM Cells

Functionally like basic RNN cell

Performance far better

Amazing success at long-term patterns

Long text sequences, time series...



Variants

Peephole connections: LSTM cells that store state for more than 1 period

Gated Recurrent Unit (GRU): Simplified LSTM with better performance

- Only 1 state vector
- Fewer internal gates and NNs

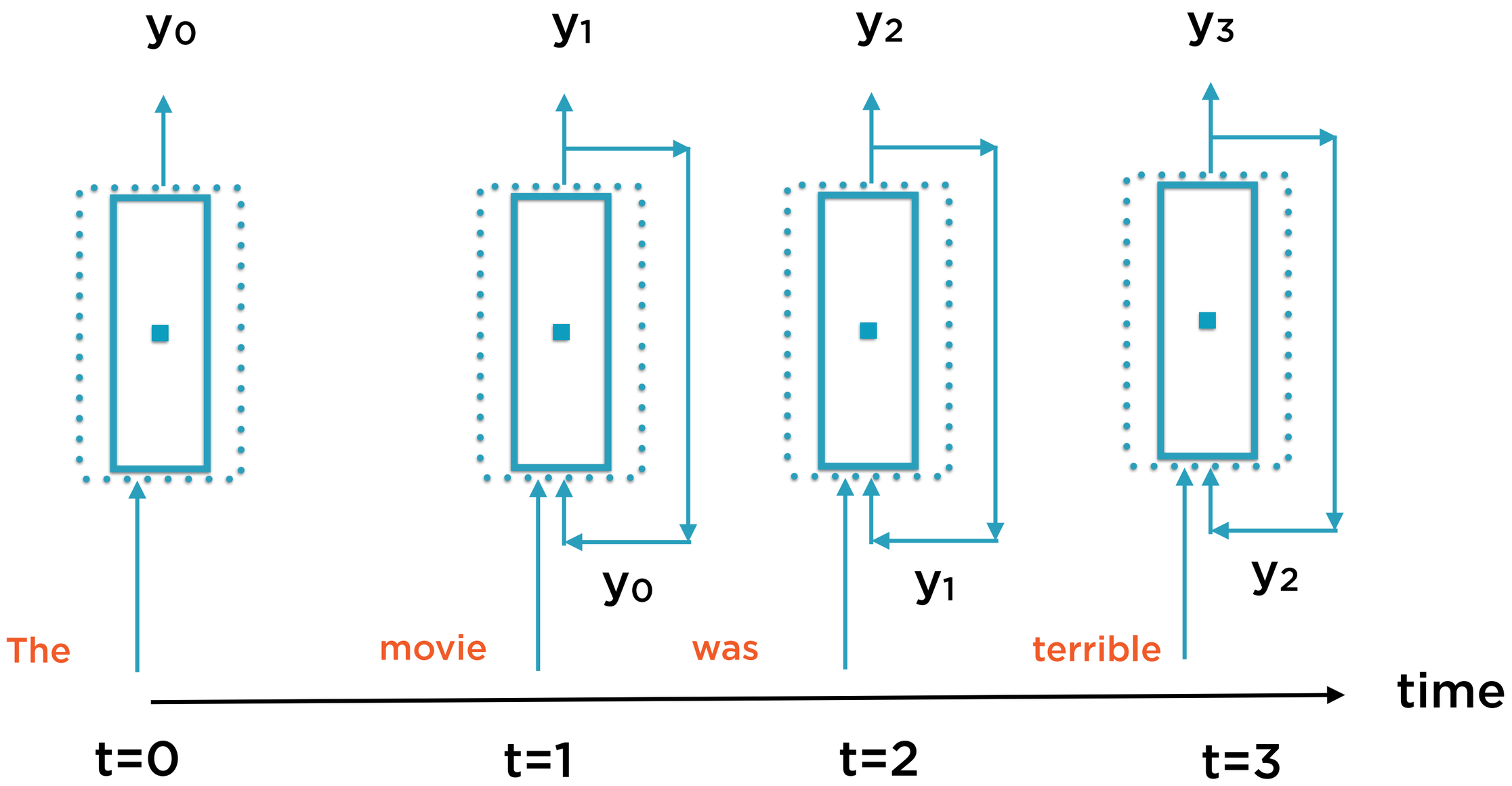
Bidirectional RNNs

```
d = “The movie was terrible. Hated it.”
```

Document as Word Sequence

Model a document as an ordered sequence of words

Conventional RNN



Conventional RNN

The sky is cloudy, it looks like_____

The blank slot will be filled in using preceding information only

Bidirectional RNN

The sky is _____, it looks like rain

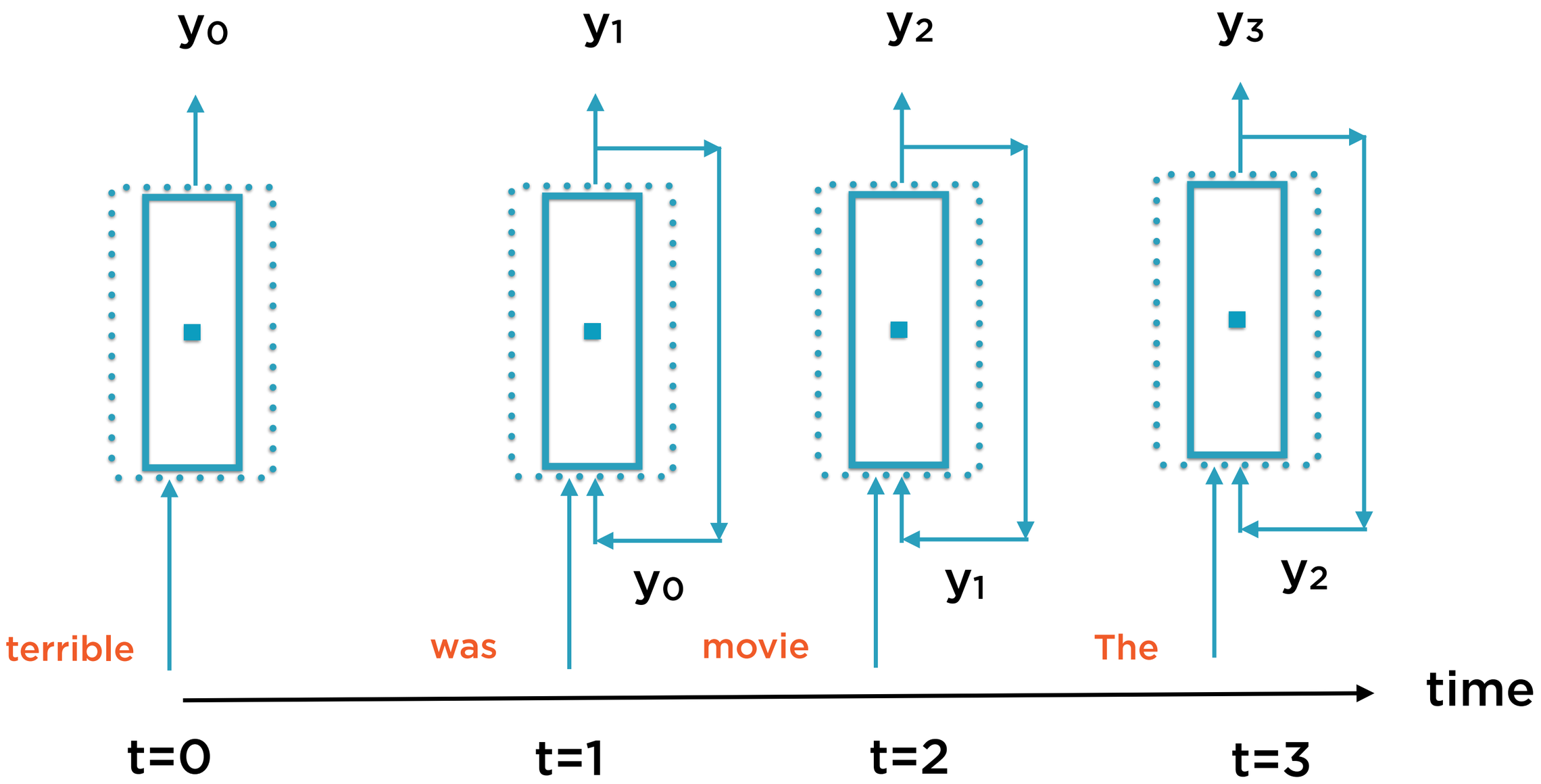
Bidirectional RNNs will also use following information

```
d = ".it Hated .terrible was movie The"
```

Reverse the Document Word Sequence

Bidirectional RNNs run words both forward and backward

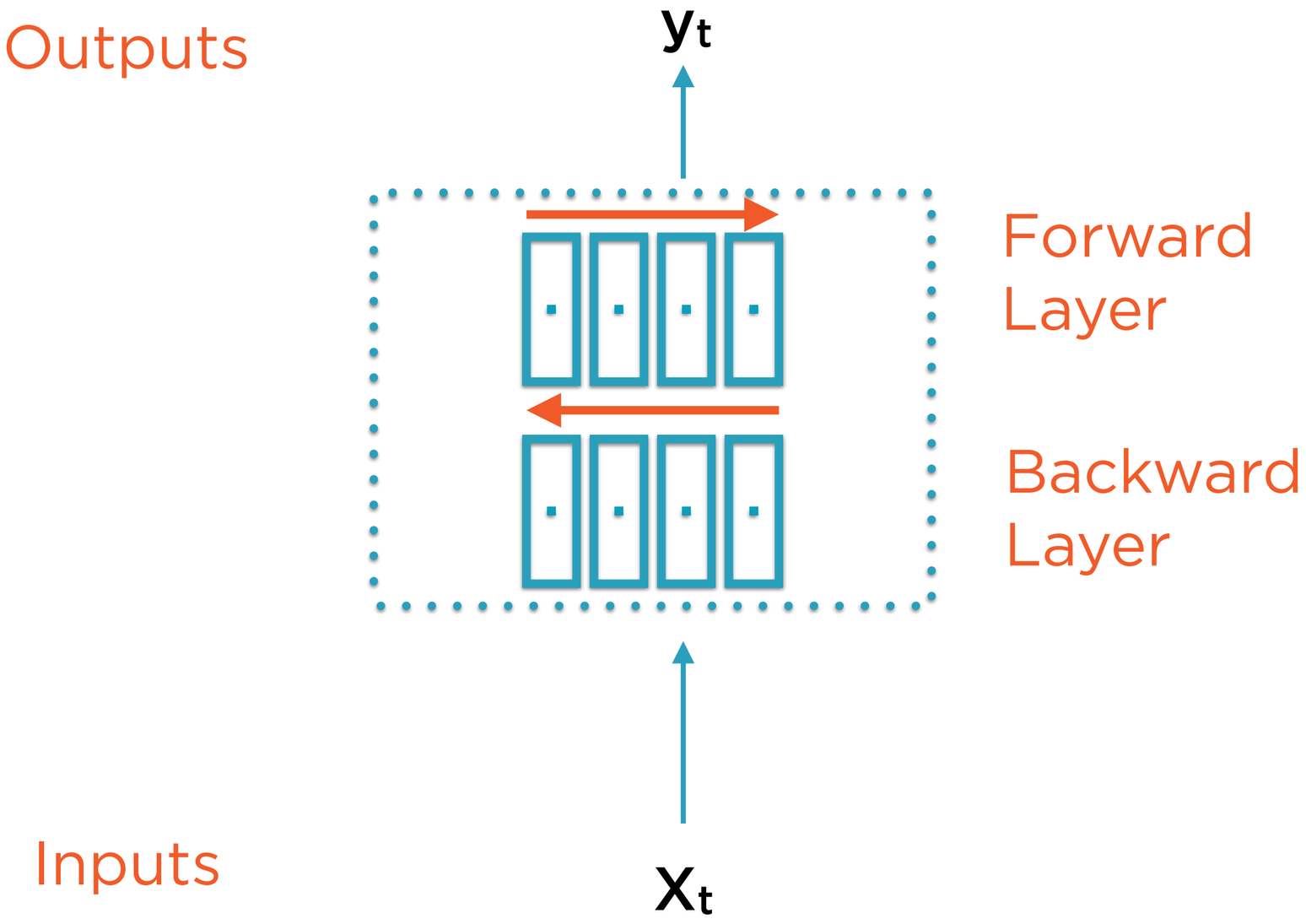
Conventional RNN

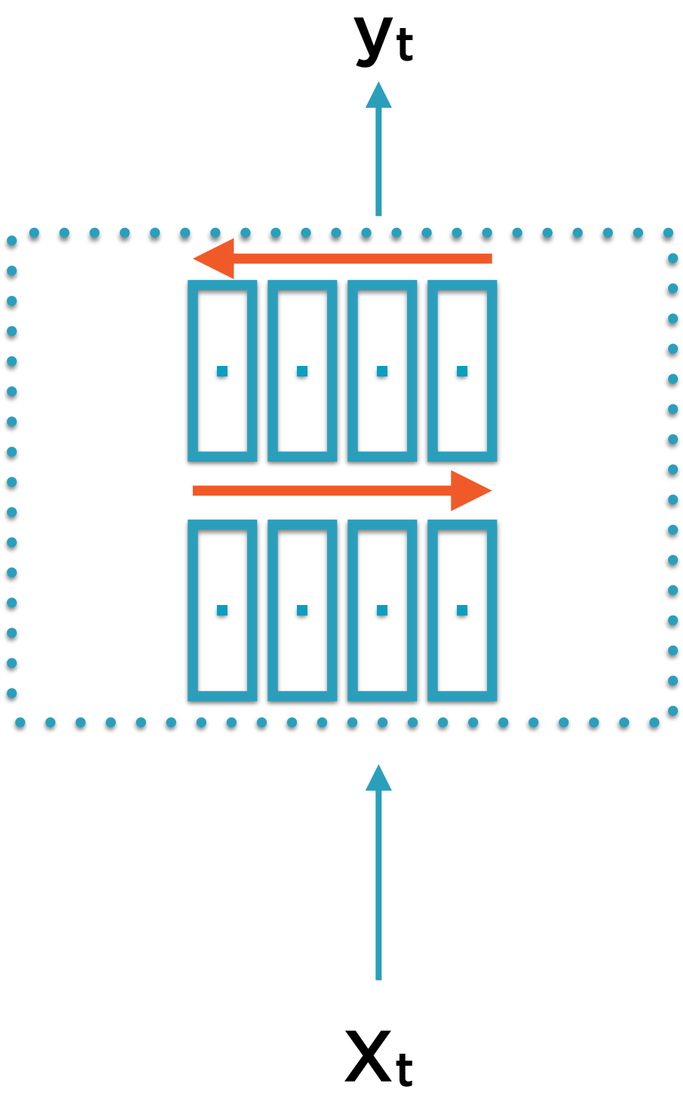


Is the use of a Bidirectional RNN **cheating**?

For streaming applications - yes; for batch applications - **no!**

Bidirectional RNNs





Bidirectional RNNs

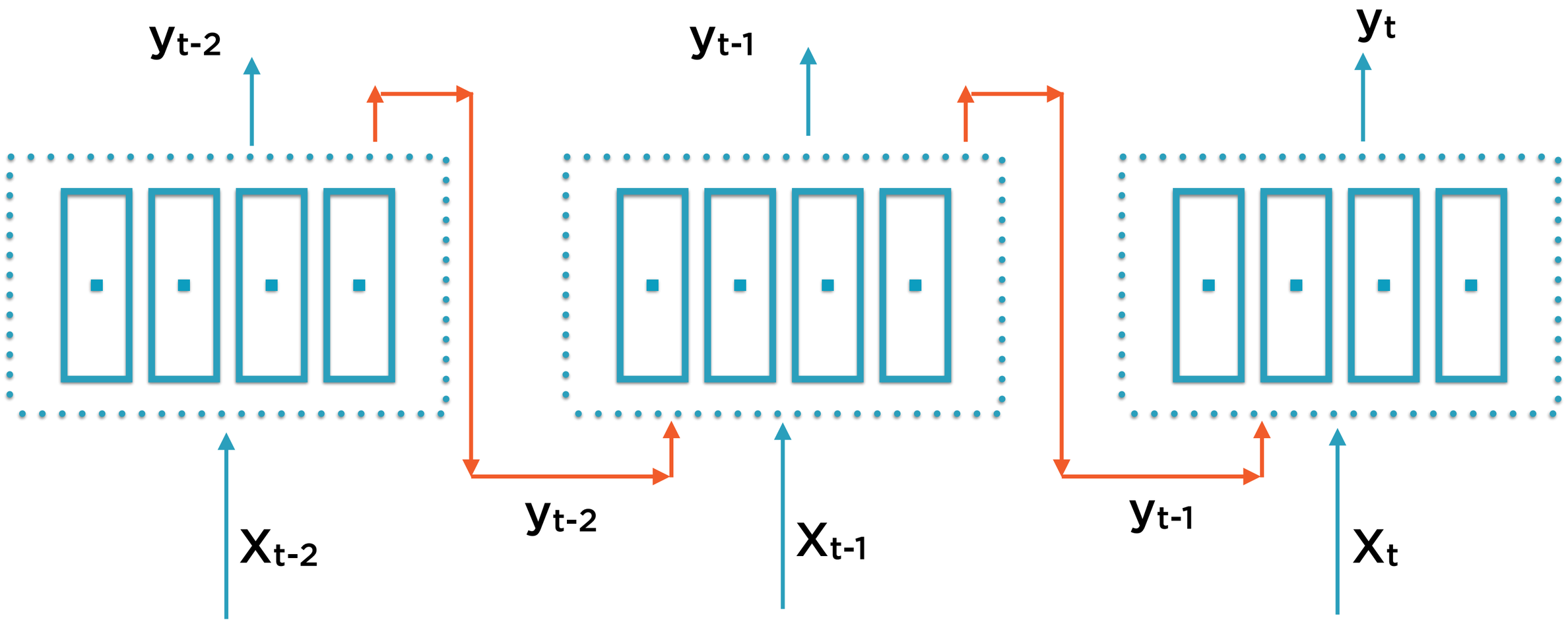
Several applications are inherently batch-oriented

Speech recognition: Utterance = batch of words spoken together

Text auto-complete, auto-correct: Fragment = batch of characters input so far

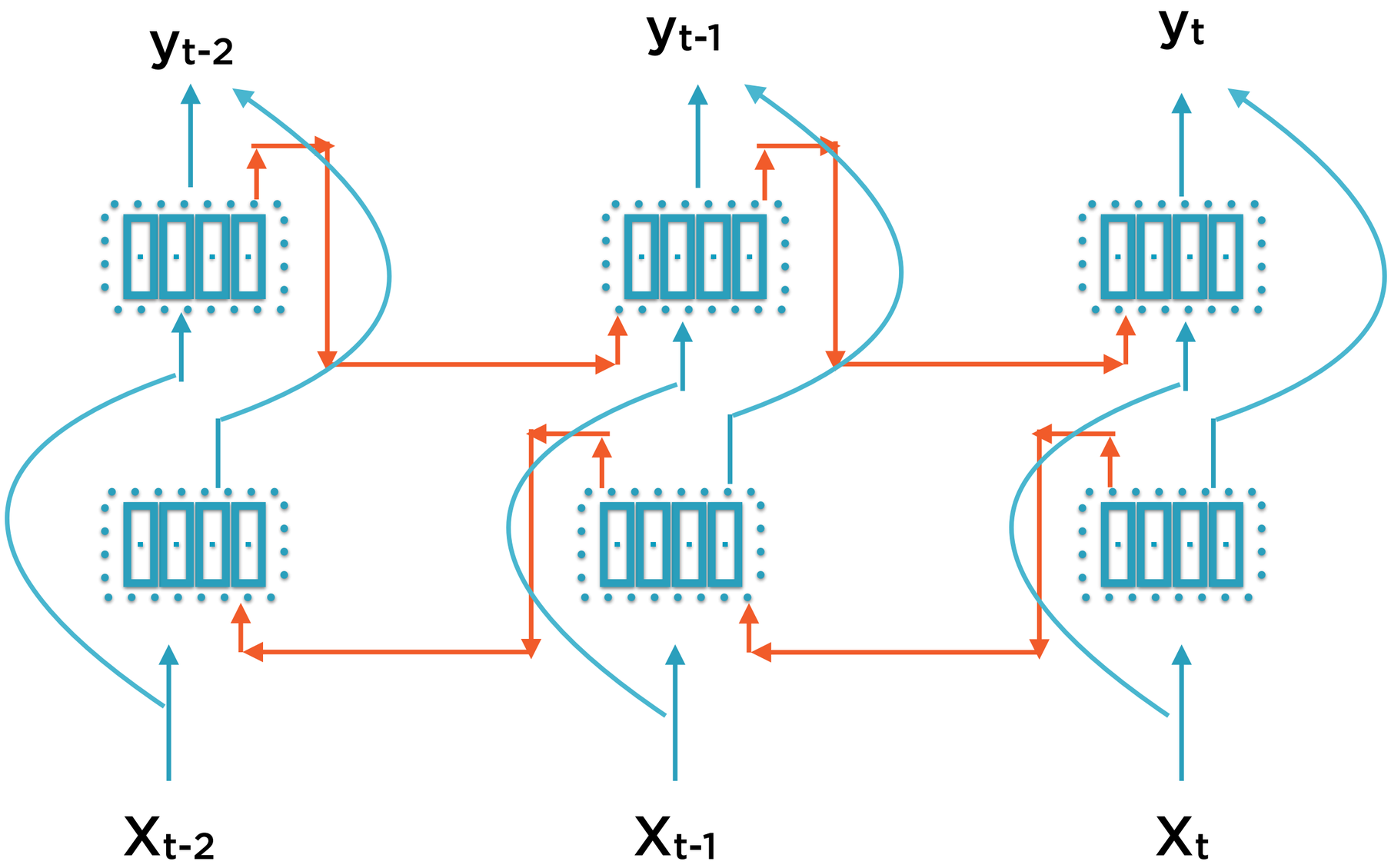
Language Modeling: e.g. Cause-effect relationships

Conventional RNNs

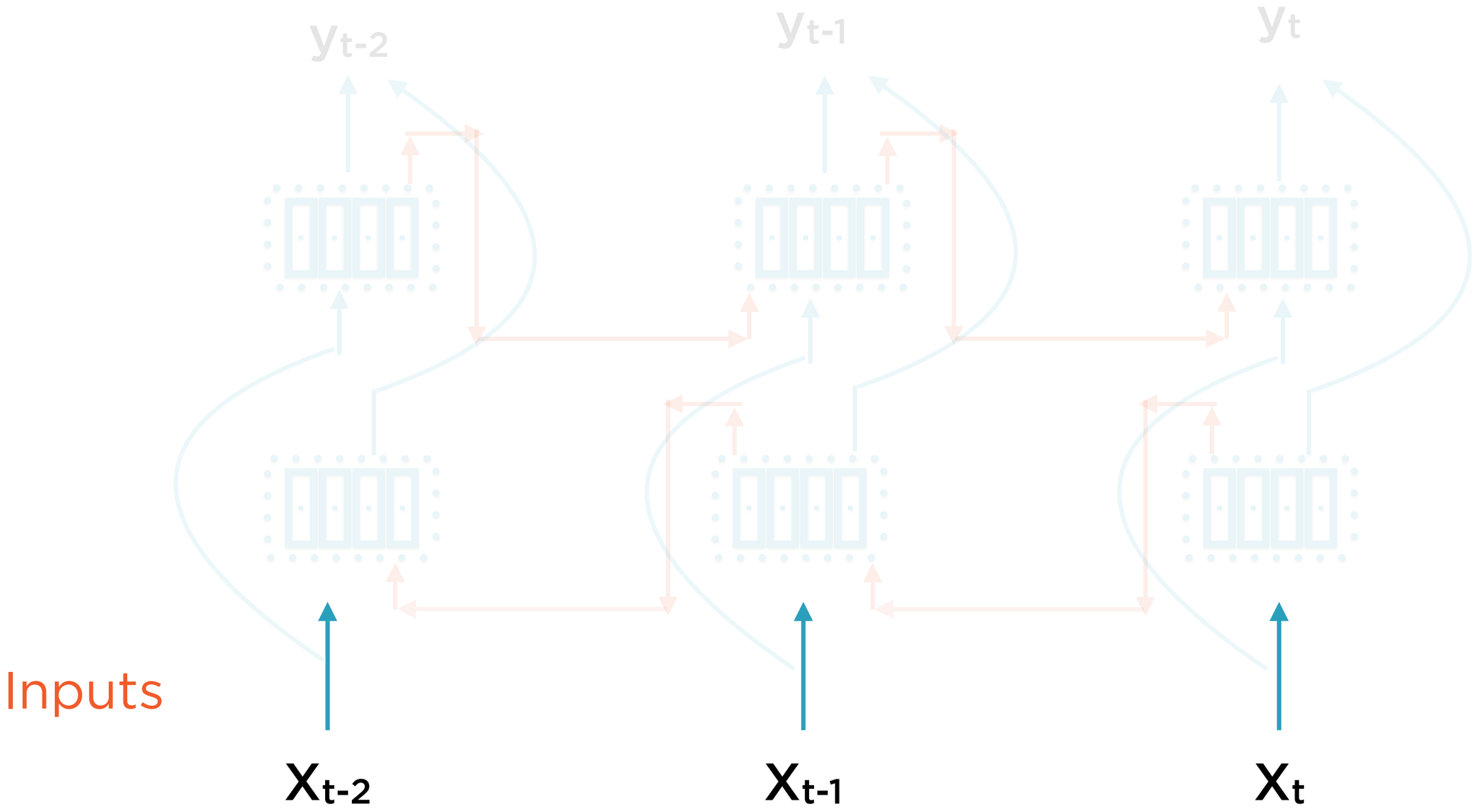


The cell unrolled through time form the layers of the neural network

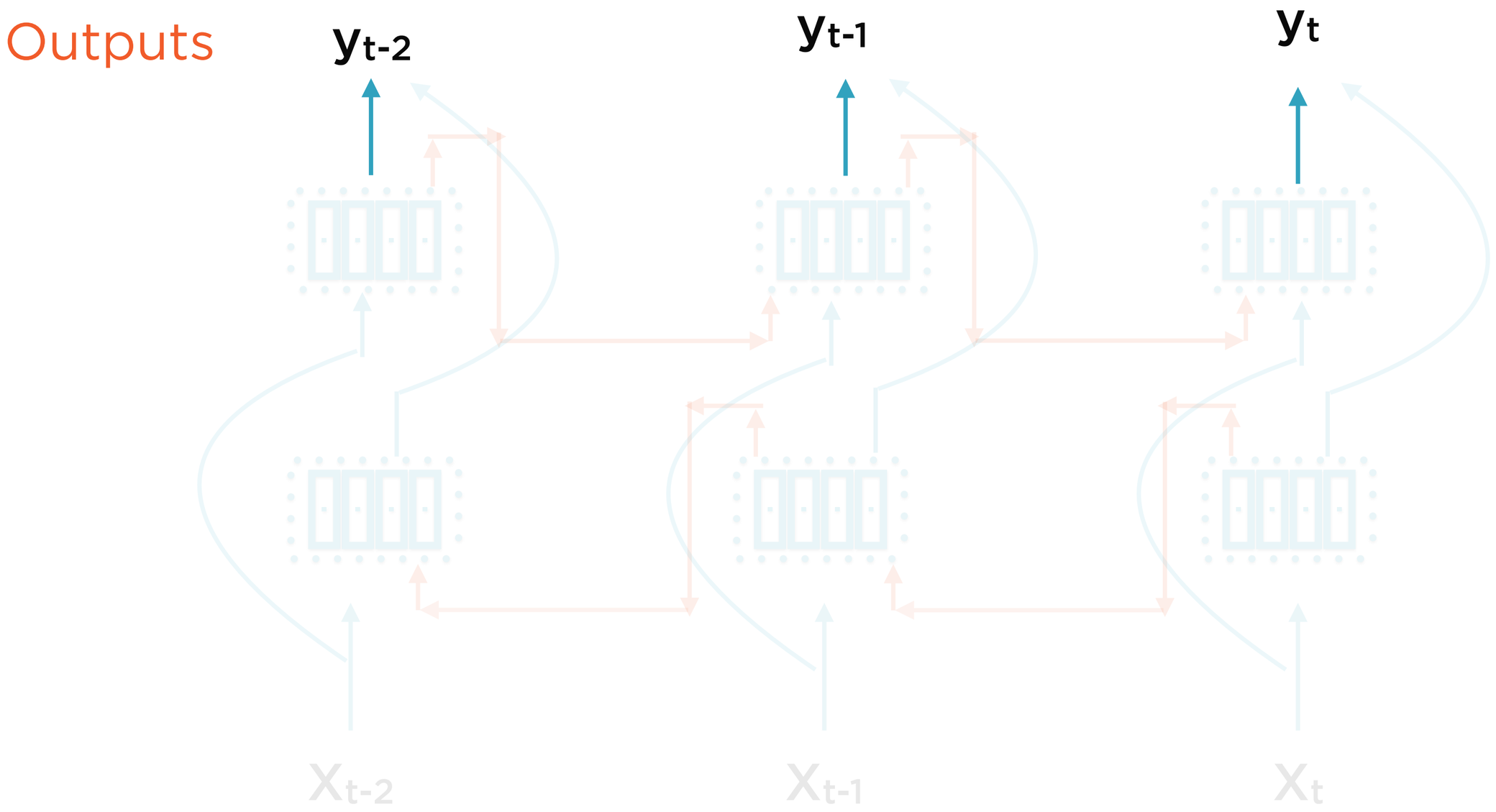
Bidirectional RNNs



Bidirectional RNNs

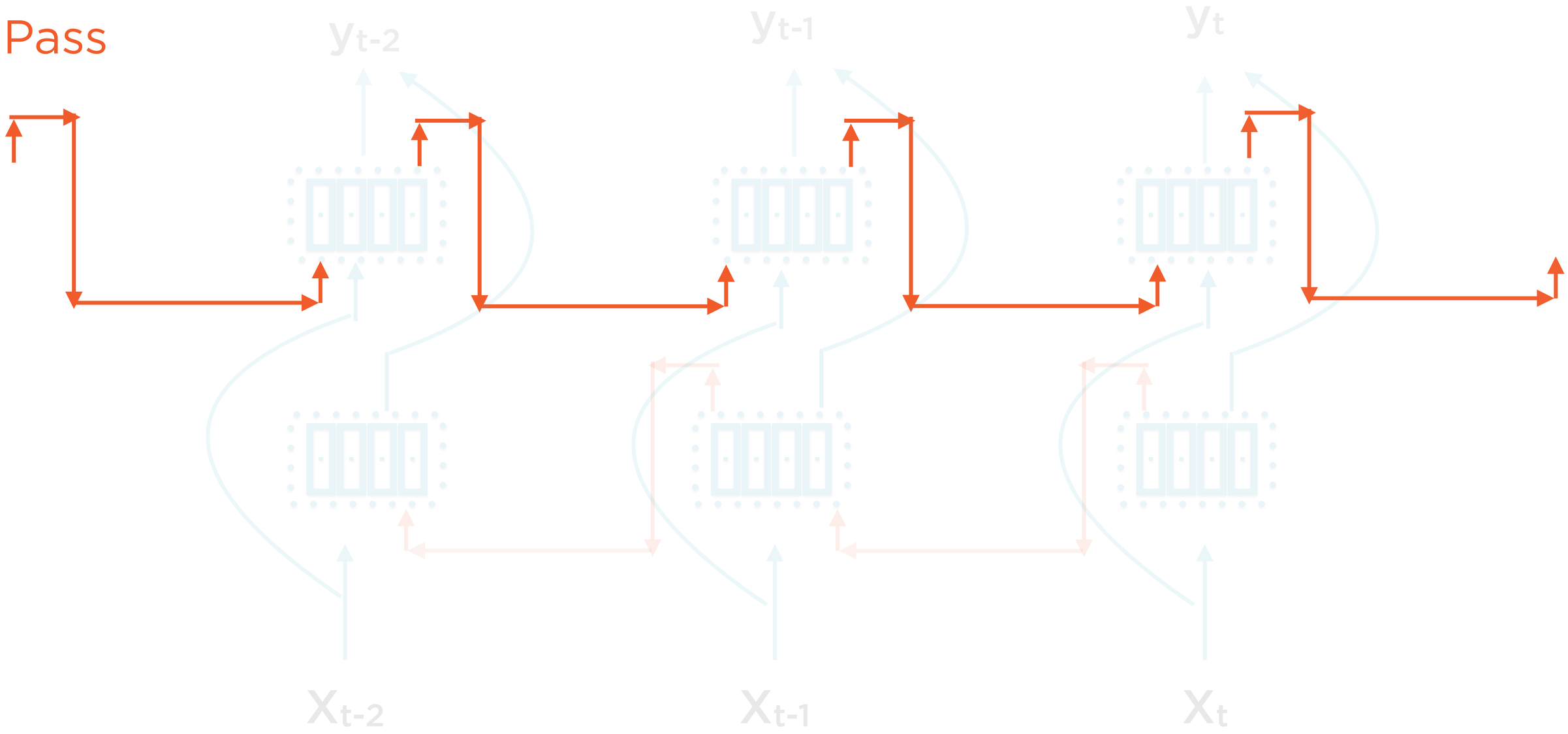


Bidirectional RNNs

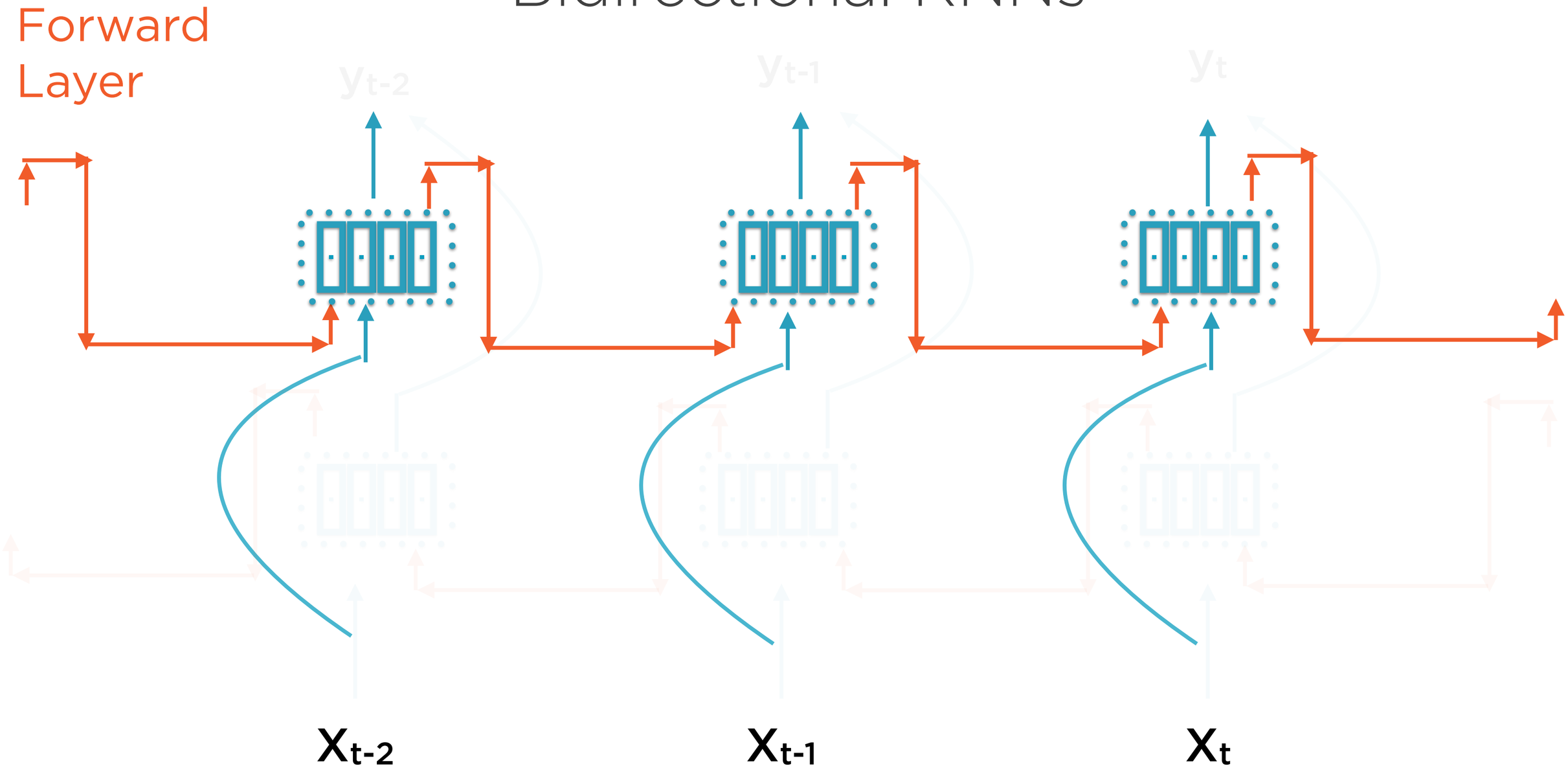


Bidirectional RNNs

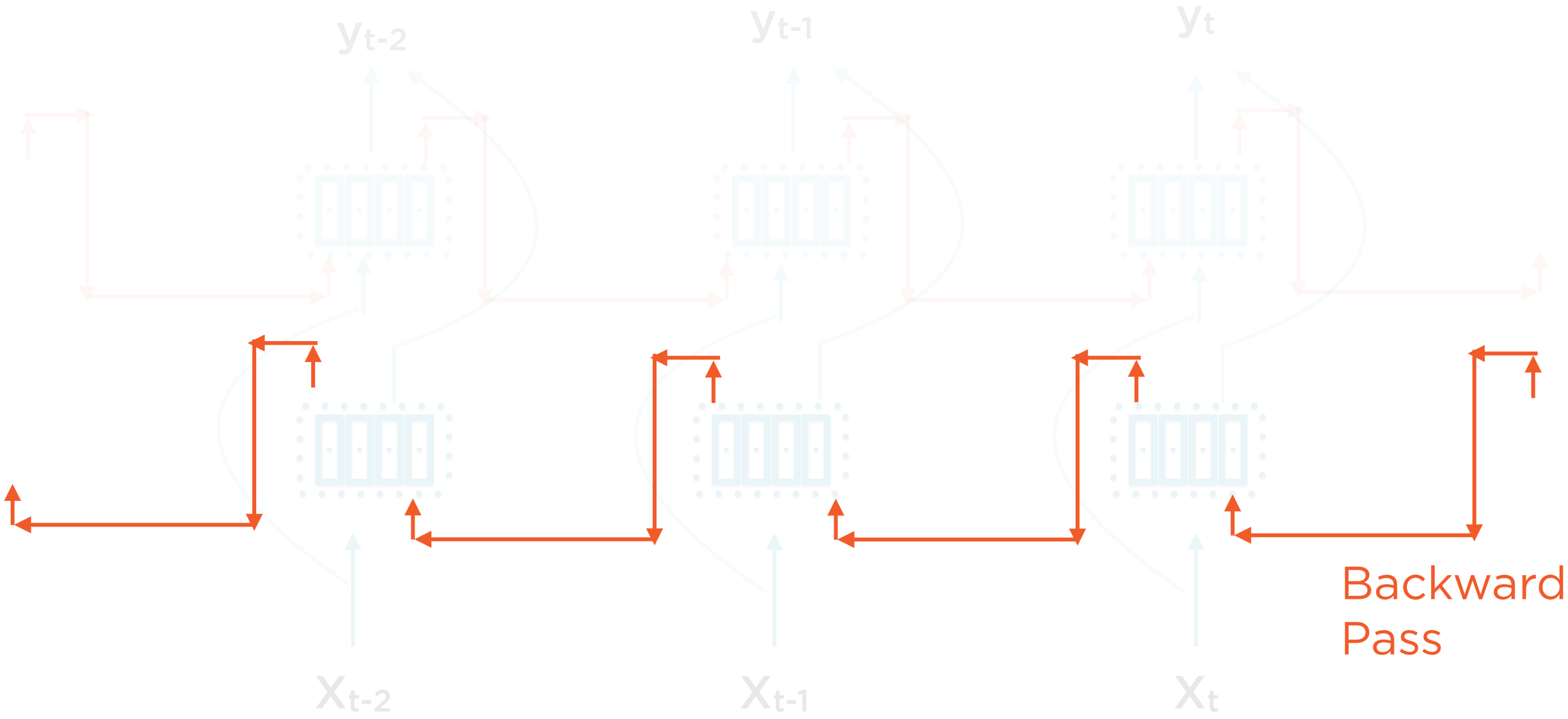
Forward Pass



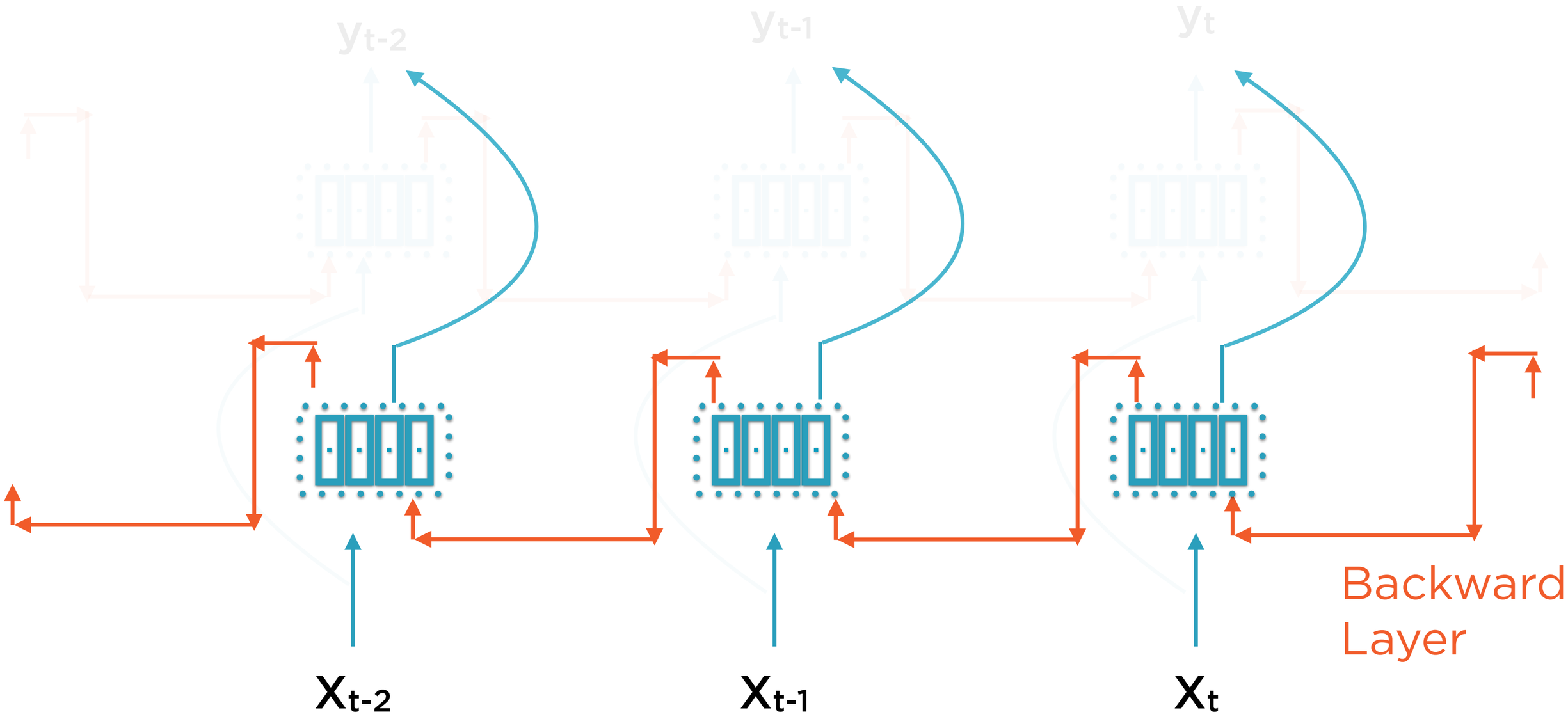
Bidirectional RNNs



Bidirectional RNNs

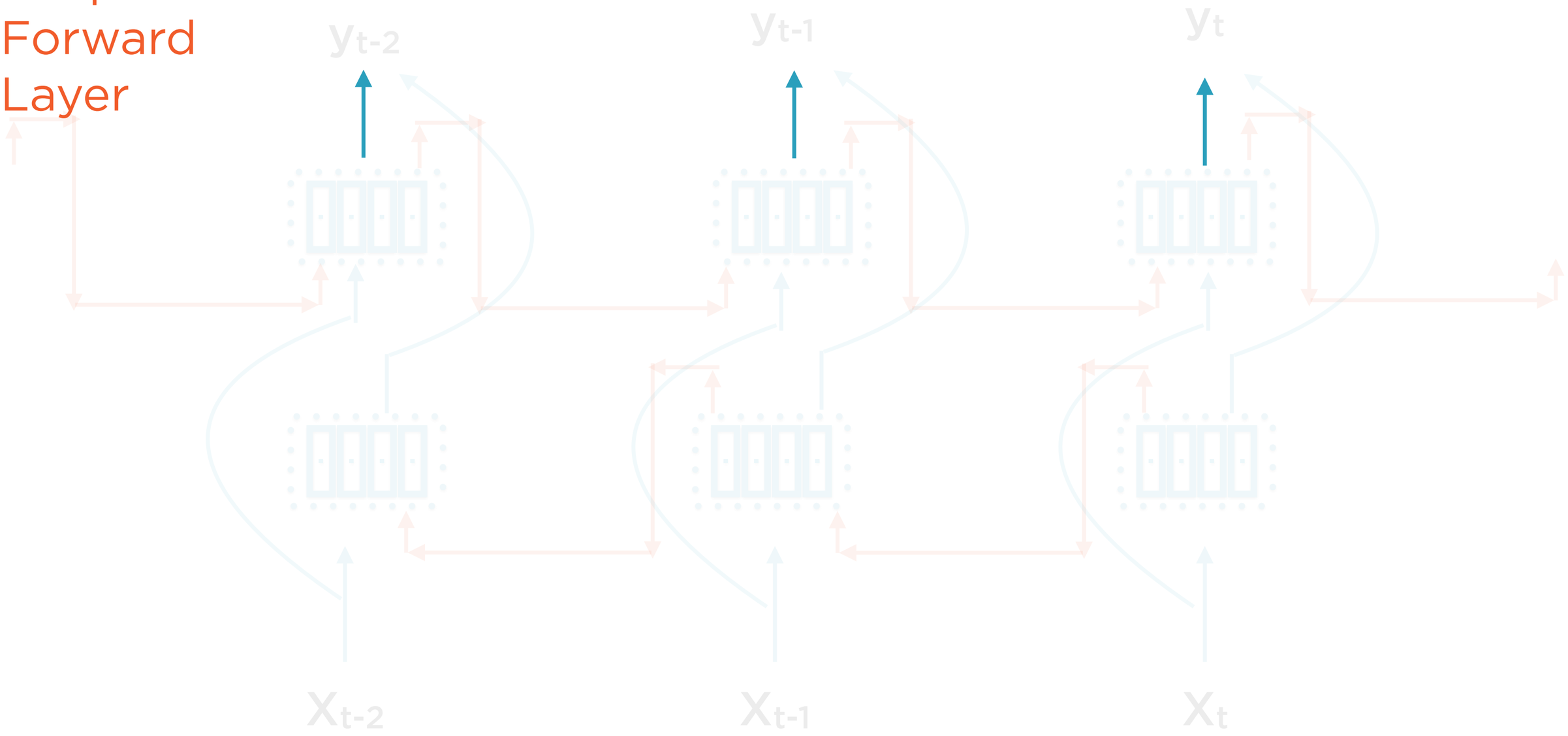


Bidirectional RNNs



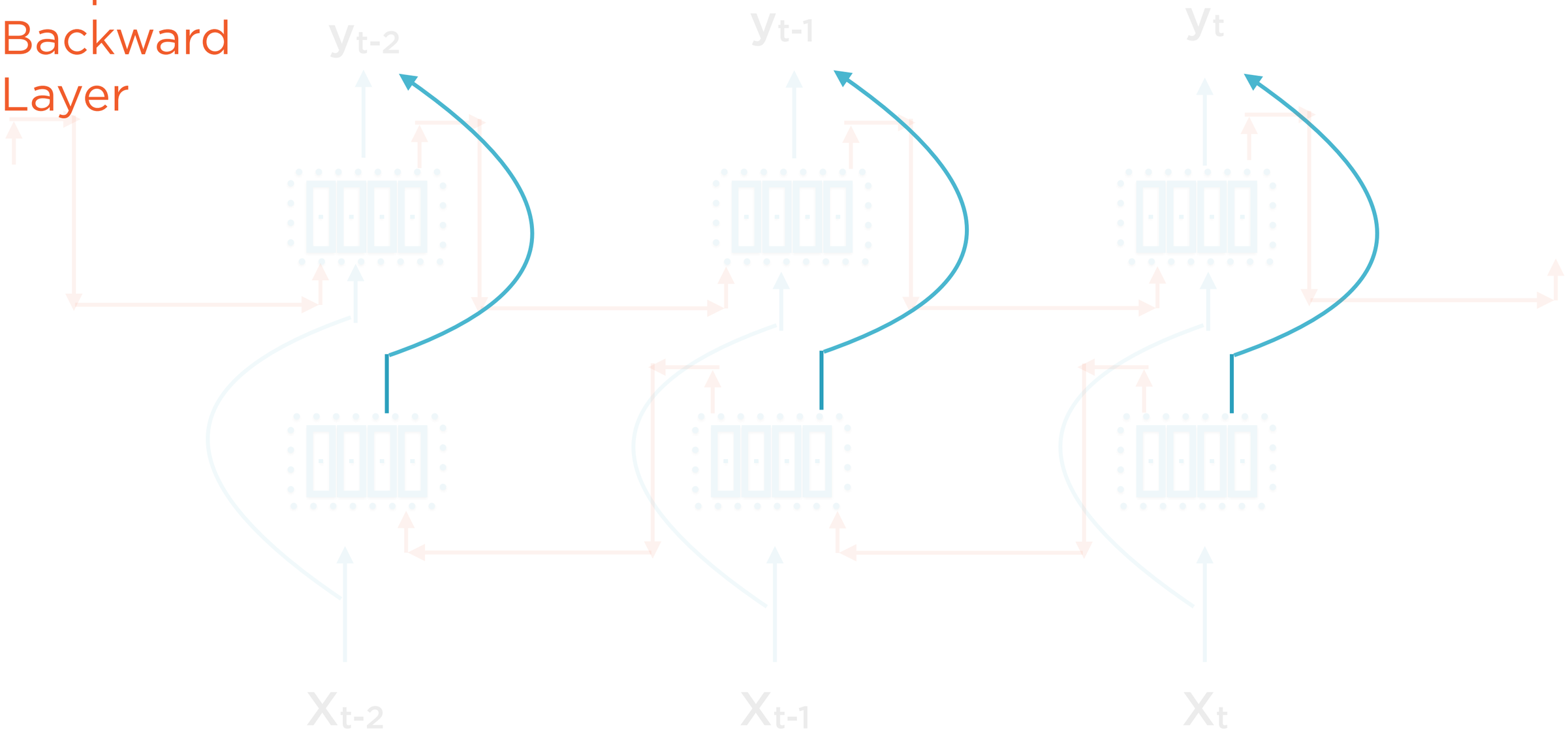
Bidirectional RNNs

Outputs of Forward Layer

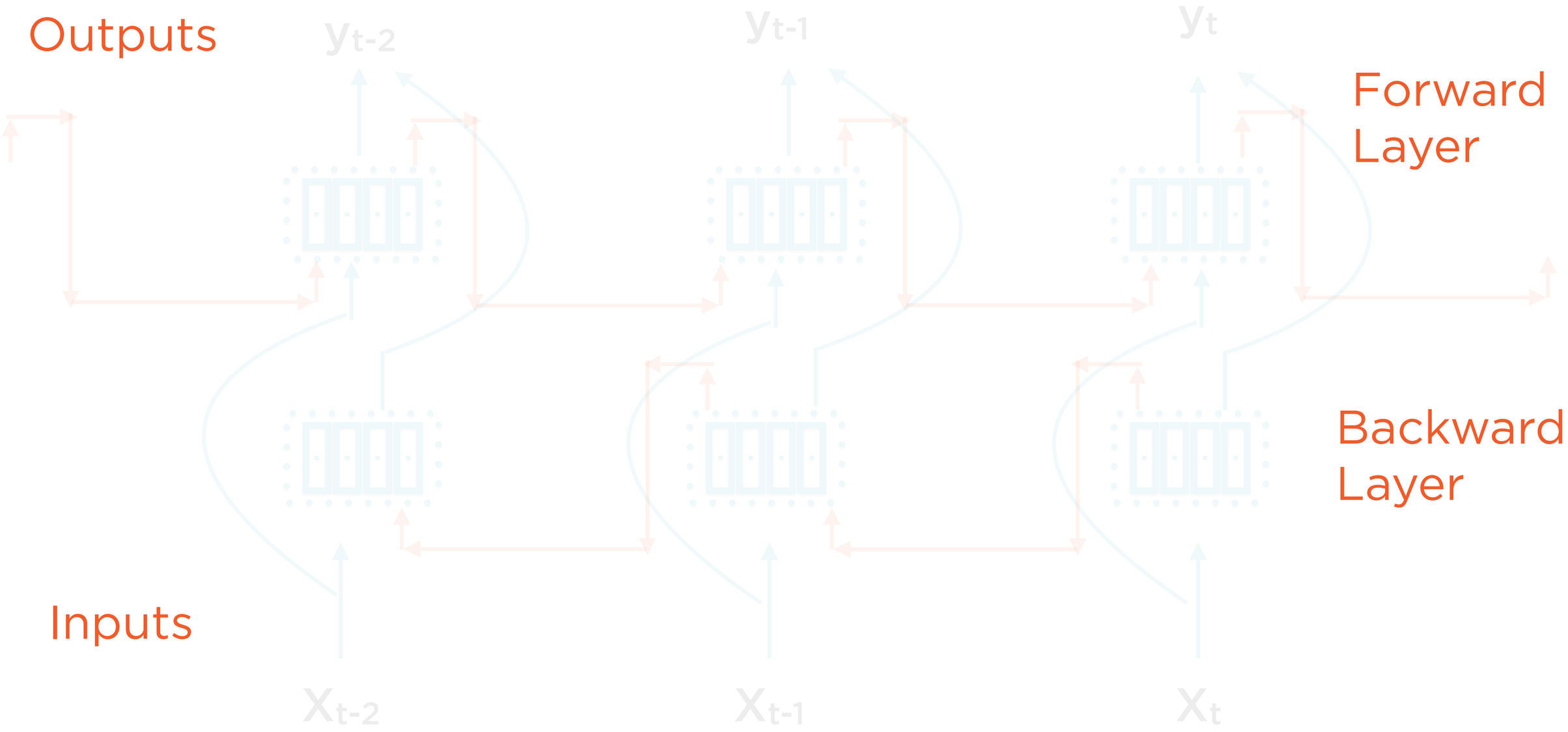


Bidirectional RNNs

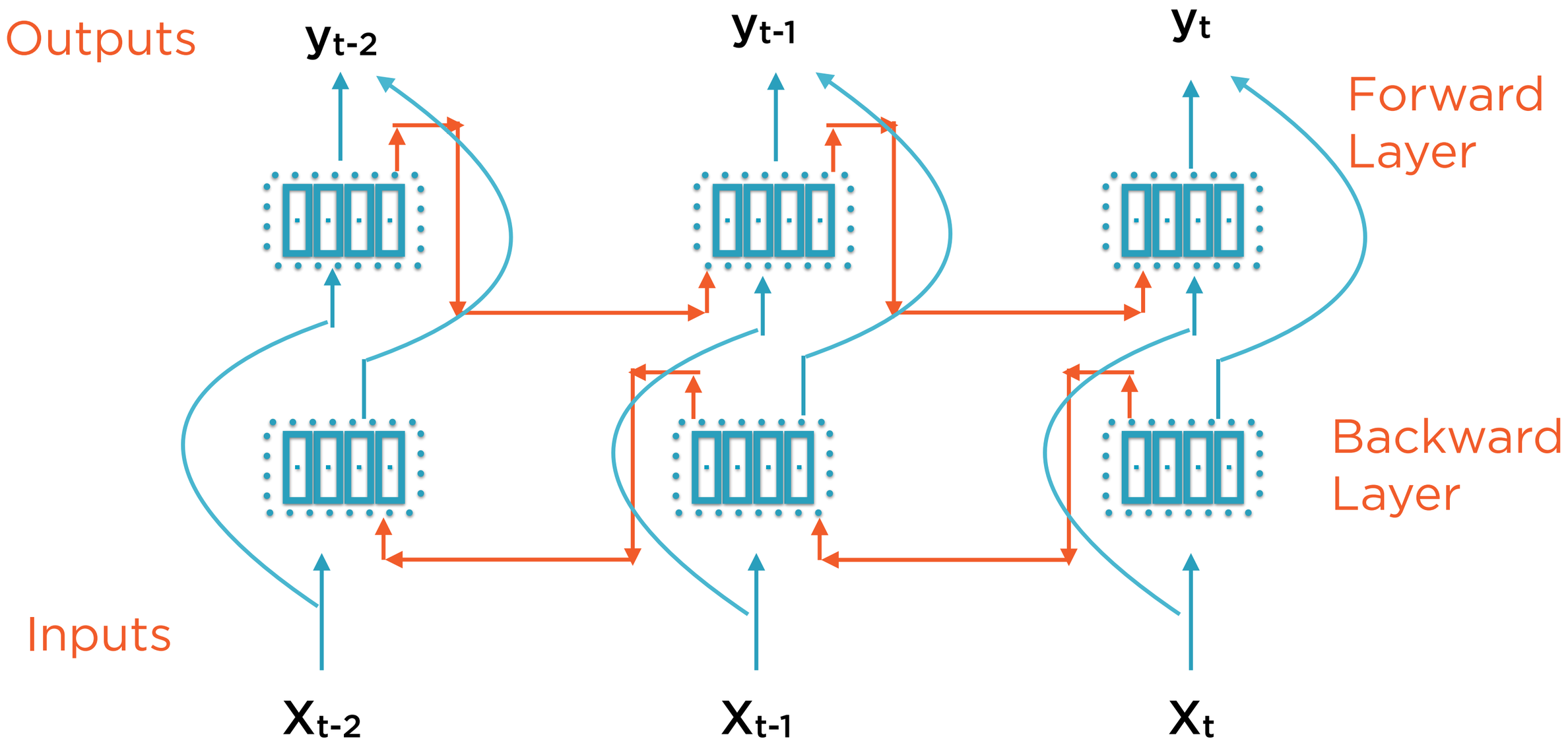
Outputs of Backward Layer



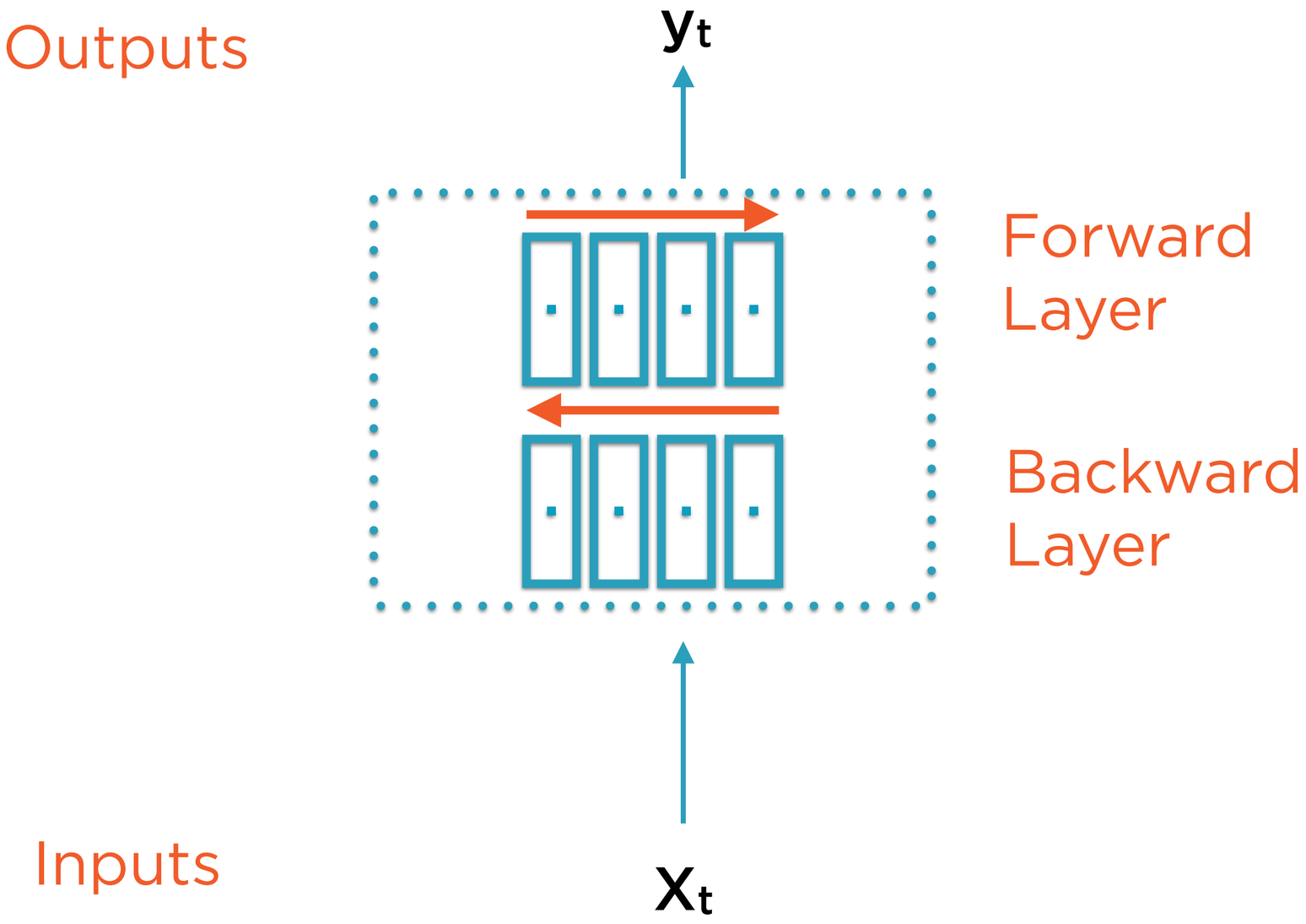
Bidirectional RNNs



Bidirectional RNNs



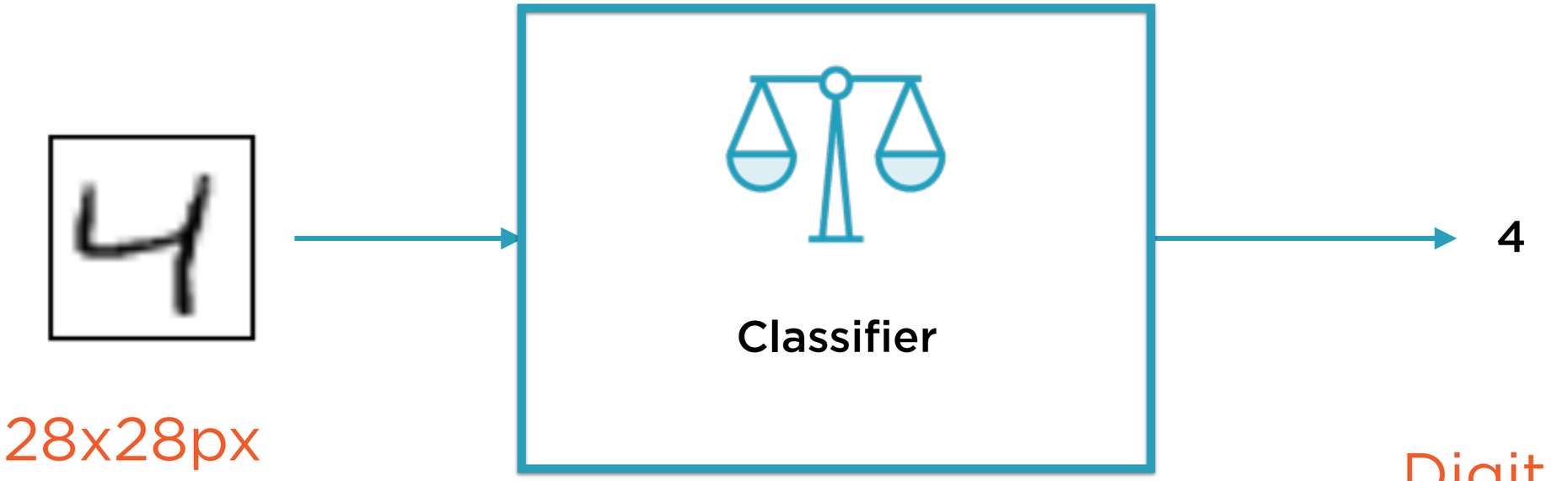
Bidirectional RNNs



Forward and backward outputs can be **combined** together in some manner

OCR as a Sequence Labeling Problem

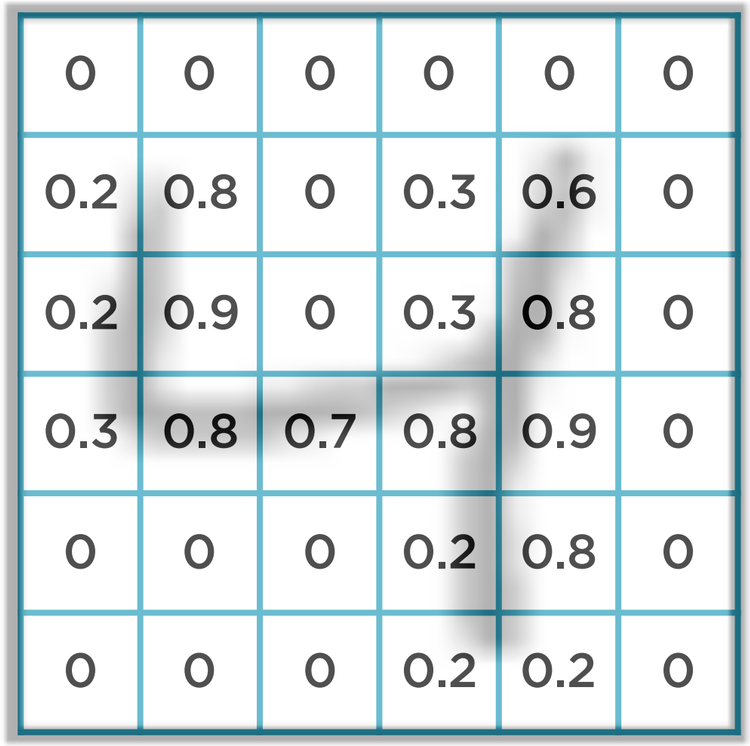
MNIST: OCR Digit Recognition



Representing Images as Matrices

28

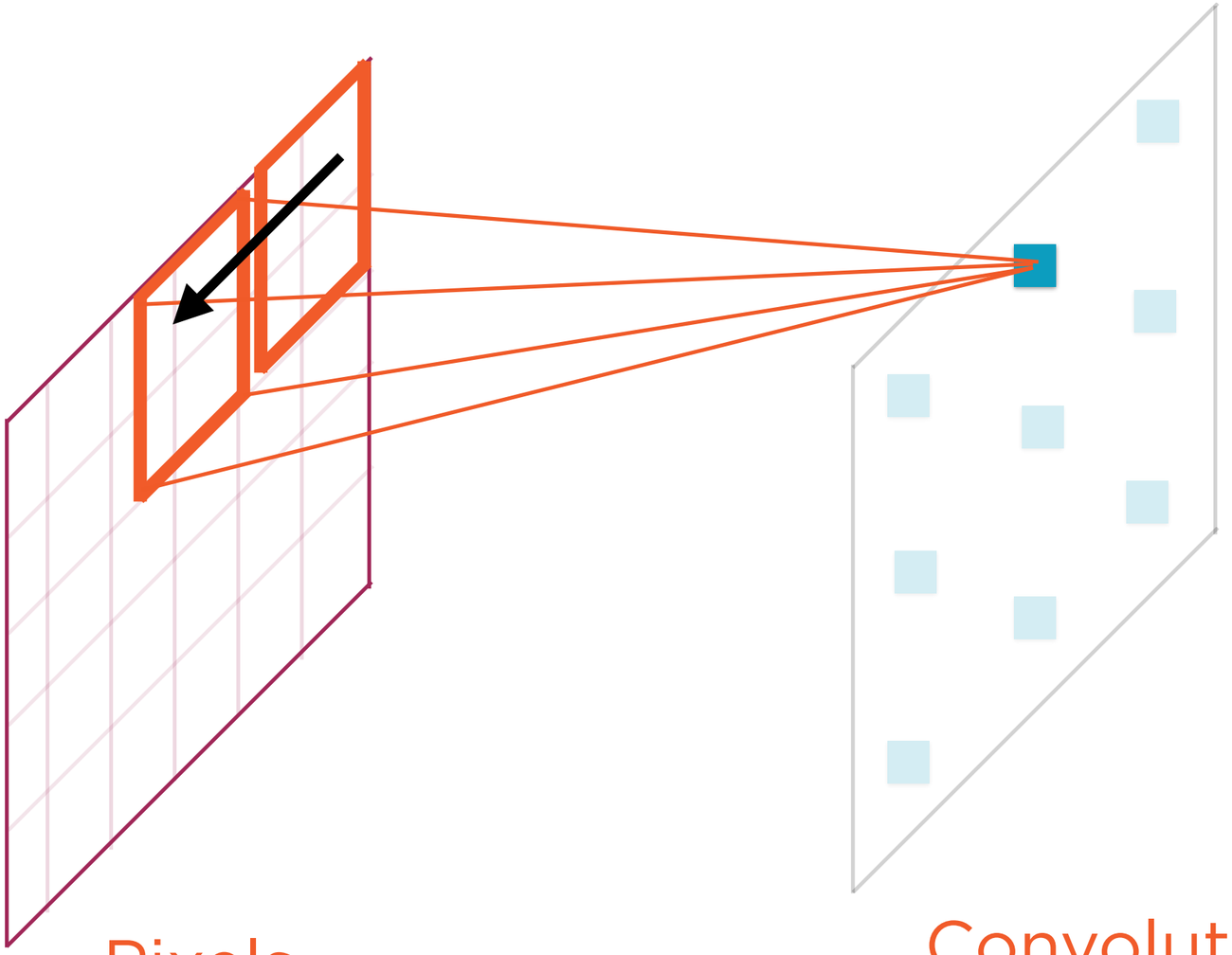
28



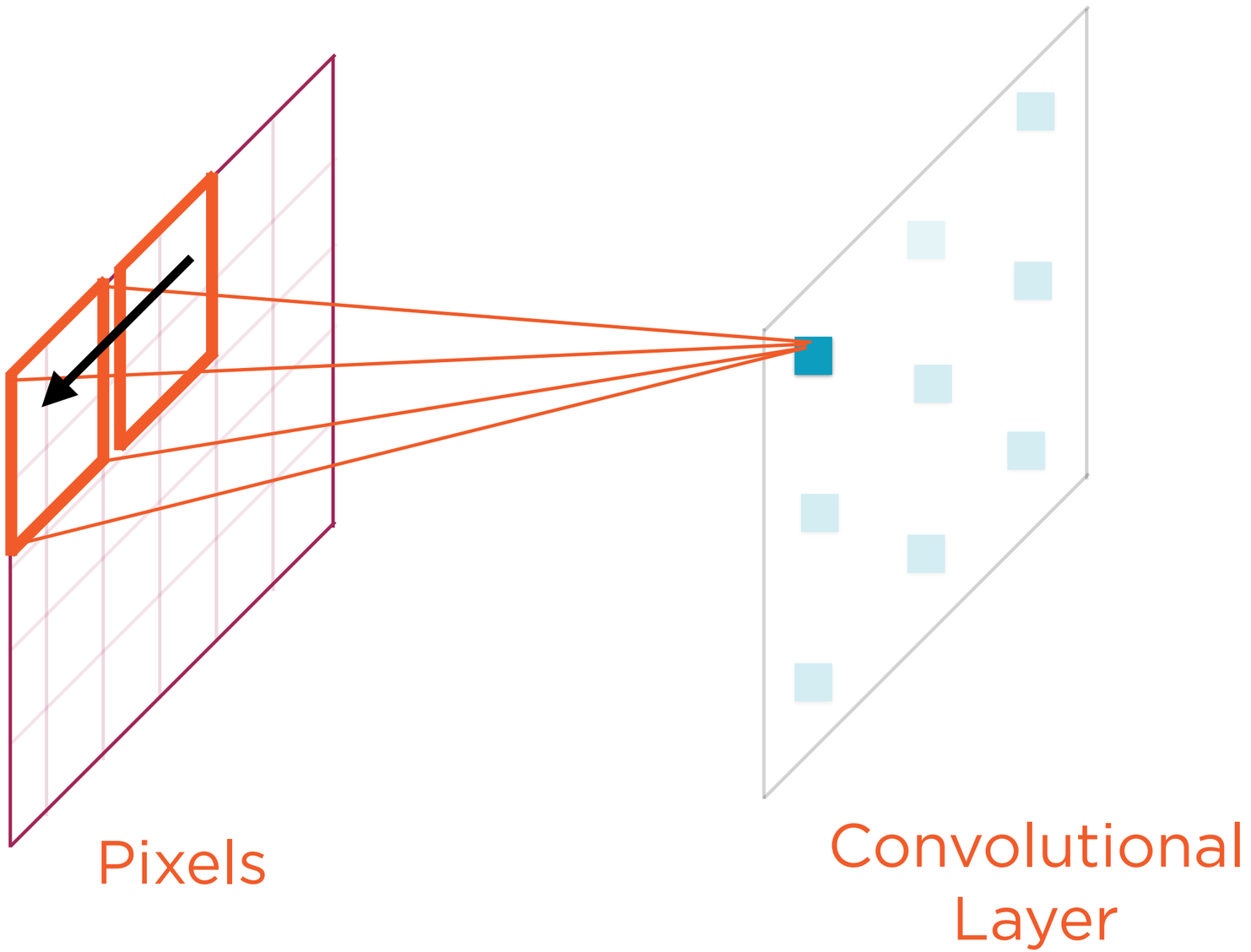
0	0	0	0	0	0
0.2	0.8	0	0.3	0.6	0
0.2	0.9	0	0.3	0.8	0
0.3	0.8	0.7	0.8	0.9	0
0	0	0	0.2	0.8	0
0	0	0	0.2	0.2	0

= 784 pixels

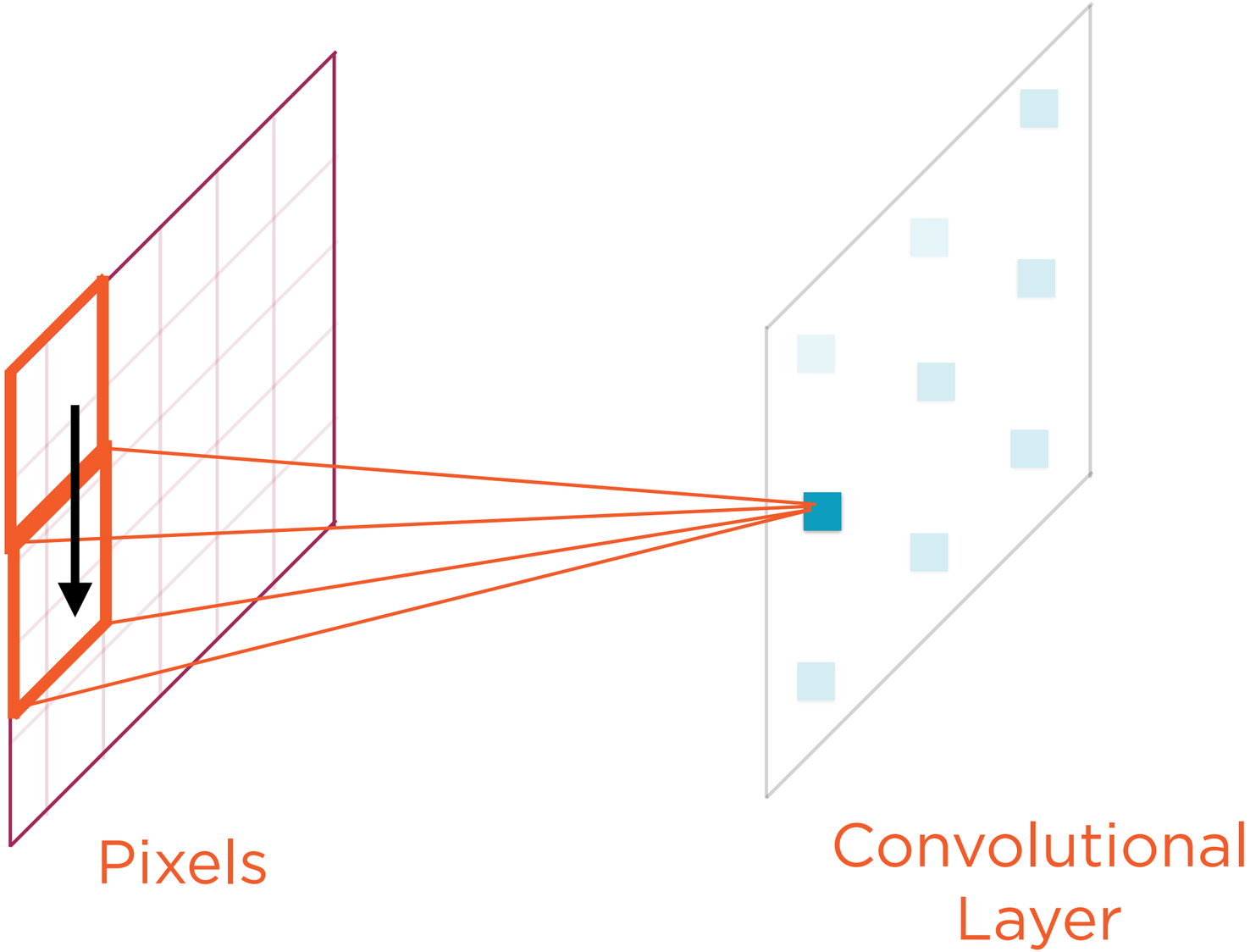
Convolution



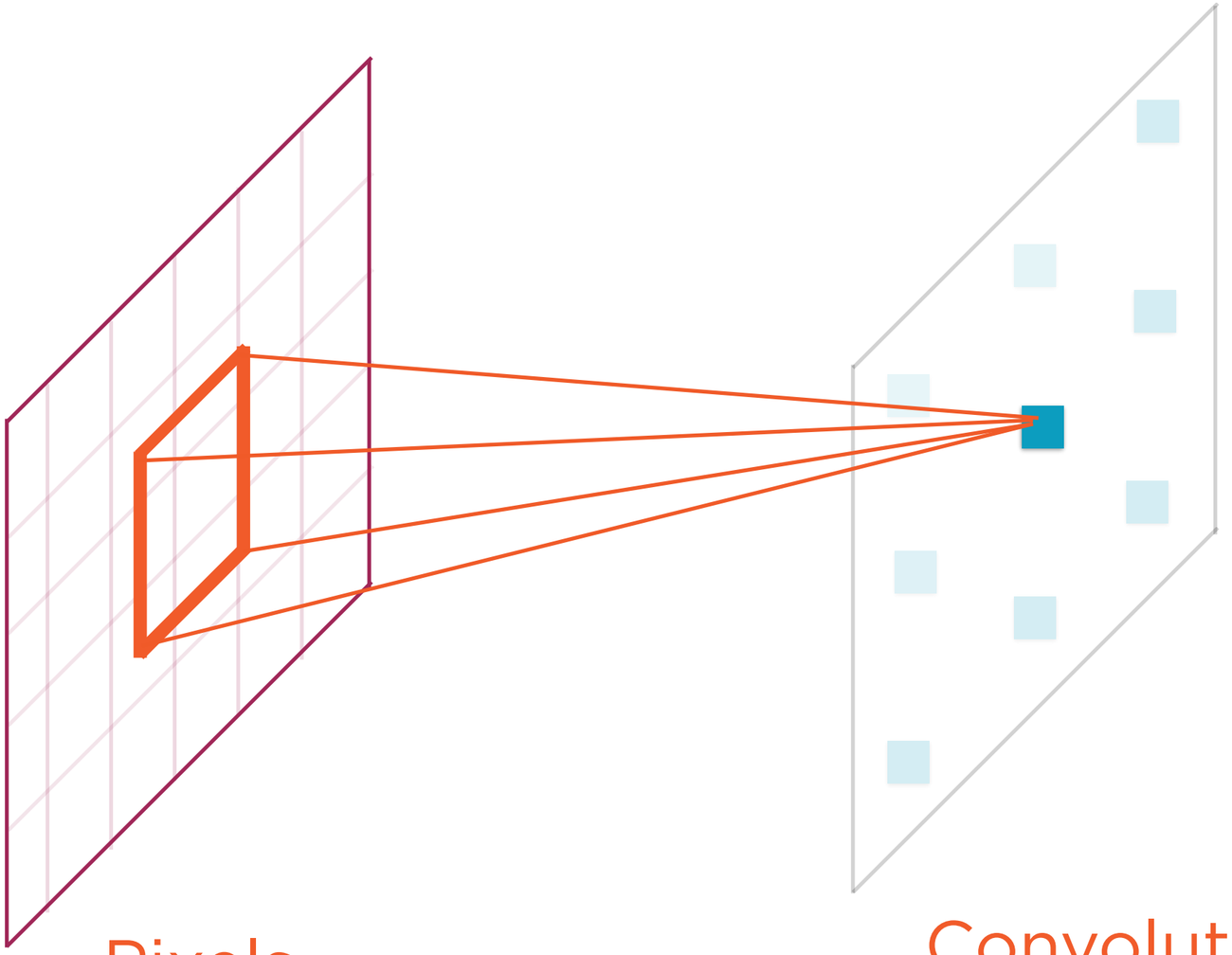
Convolution



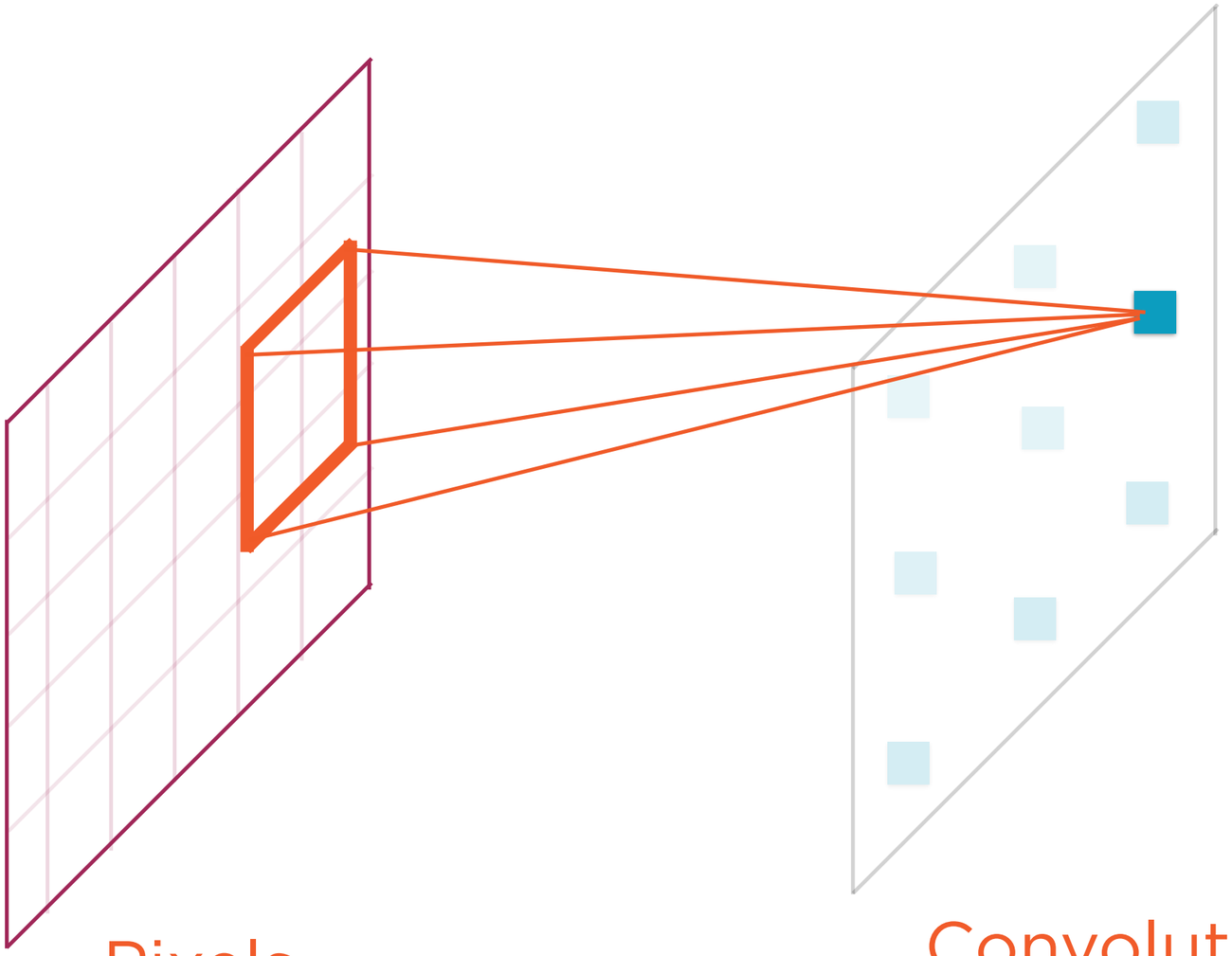
Convolution



Convolution

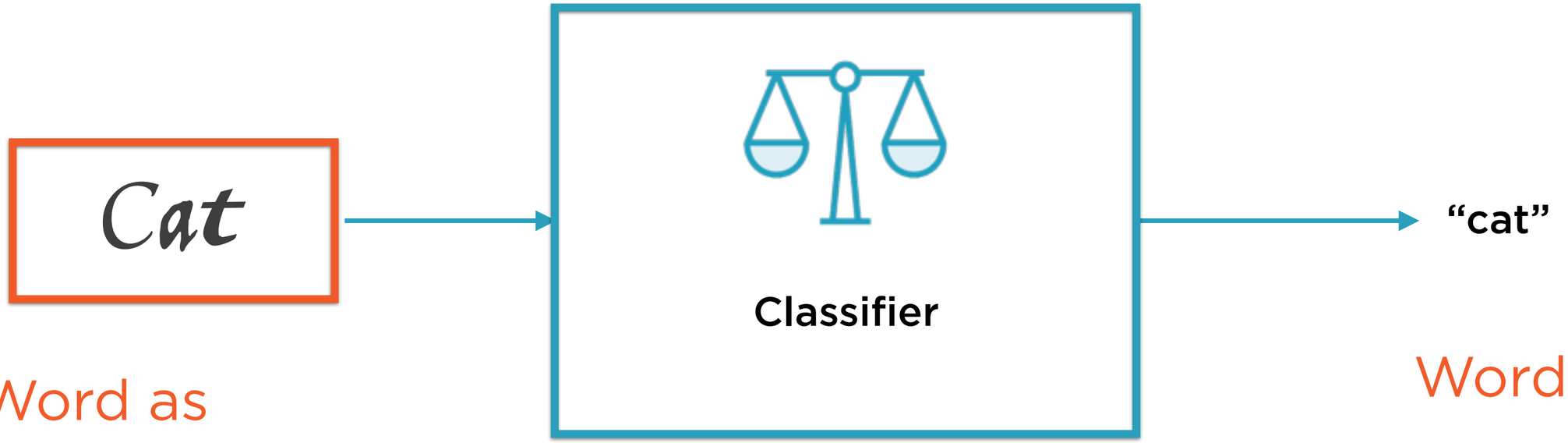


Convolution



CNNs, when used in character recognition, identify patterns within a **single** image

OCR Word Recognition



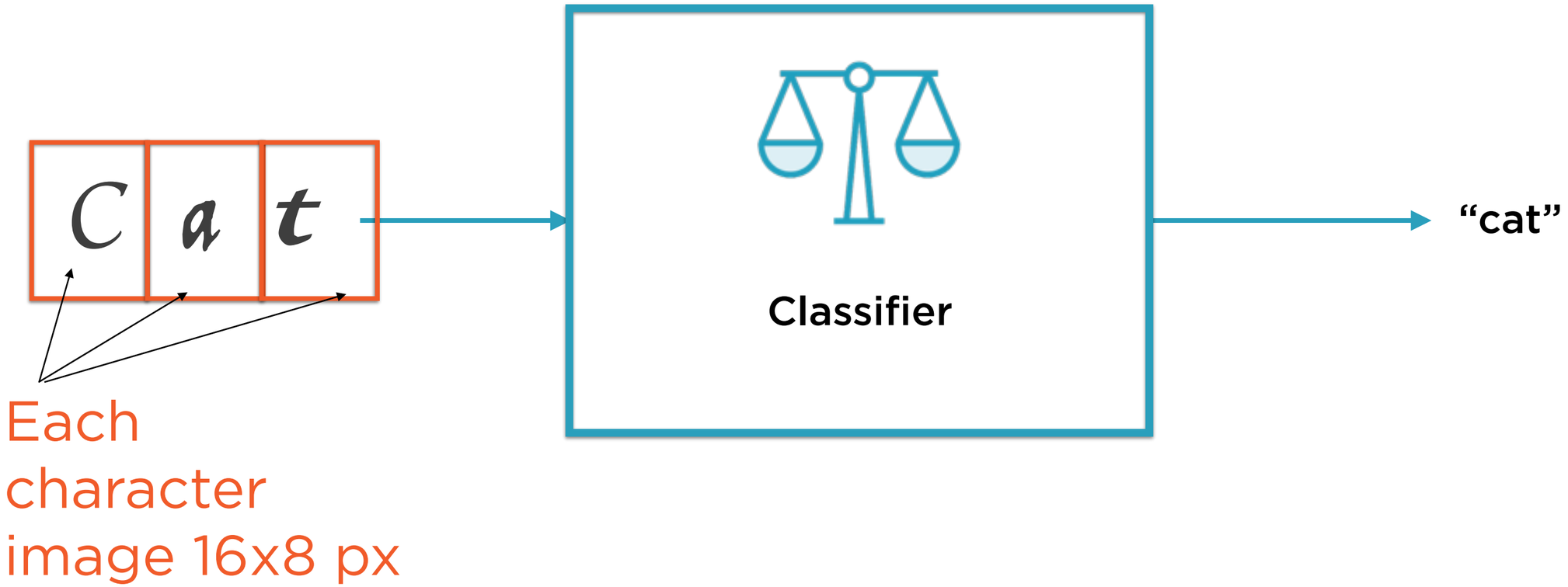
Word as sequence of character images

Word

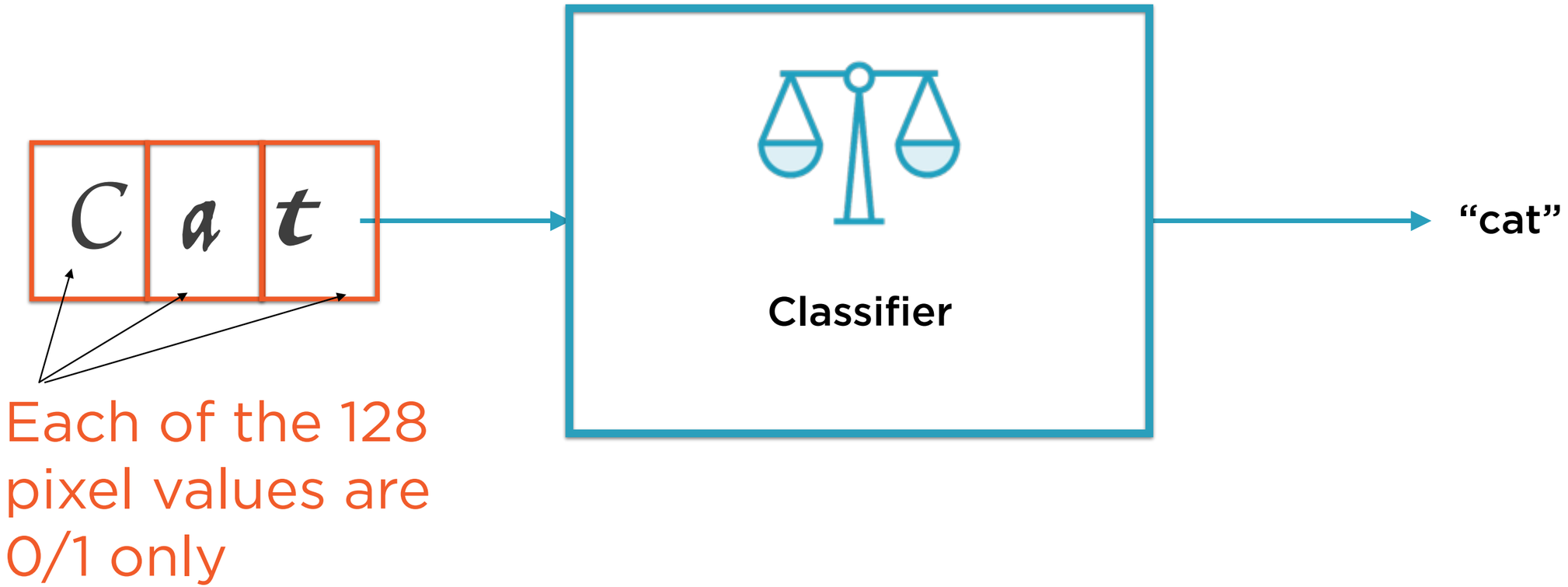
OCR Word Recognition

Word as sequence of character images

OCR Word Recognition



OCR Word Recognition



RNNs will also use the **context** of the character sequence to perform OCR

MIT OCR Dataset

OCR Word Recognition

Modified from the MIT OCR dataset

6800 words, up to 14 characters in length

Each word is a sequence of images

Each image represents one character

Each character image is 16x8 pixels

Lowercase only (uppercase just filtered out)

MIT OCR File Format

One line for each letter of each word

Each letter (line) has an id

Each line has link to the next letter's id

The next id is -1 for the start of a new word

Line includes 128 fields for pixels

Fields are tab delimited

MIT OCR File Format

ID	Letter	Next ID	Word ID	Position	Fold	128 columns			
						(1,1)	(1,2)	...	(16,8)
23	c	34							
34	a	45							
45	t	-1							

MIT OCR File Format

ID	Letter	Next ID	Word ID	Position	Fold	128 columns			
						(1,1)	(1,2)	...	(16,8)
23	c	34							
34	a	45							
45	t	-1							

MIT OCR File Format

ID	Letter	Next ID	Word ID	Position	Fold	128 columns			
						(1,1)	(1,2)	...	(16,8)
23	c	34							
34	a	45							
45	t	-1							

MIT OCR File Format

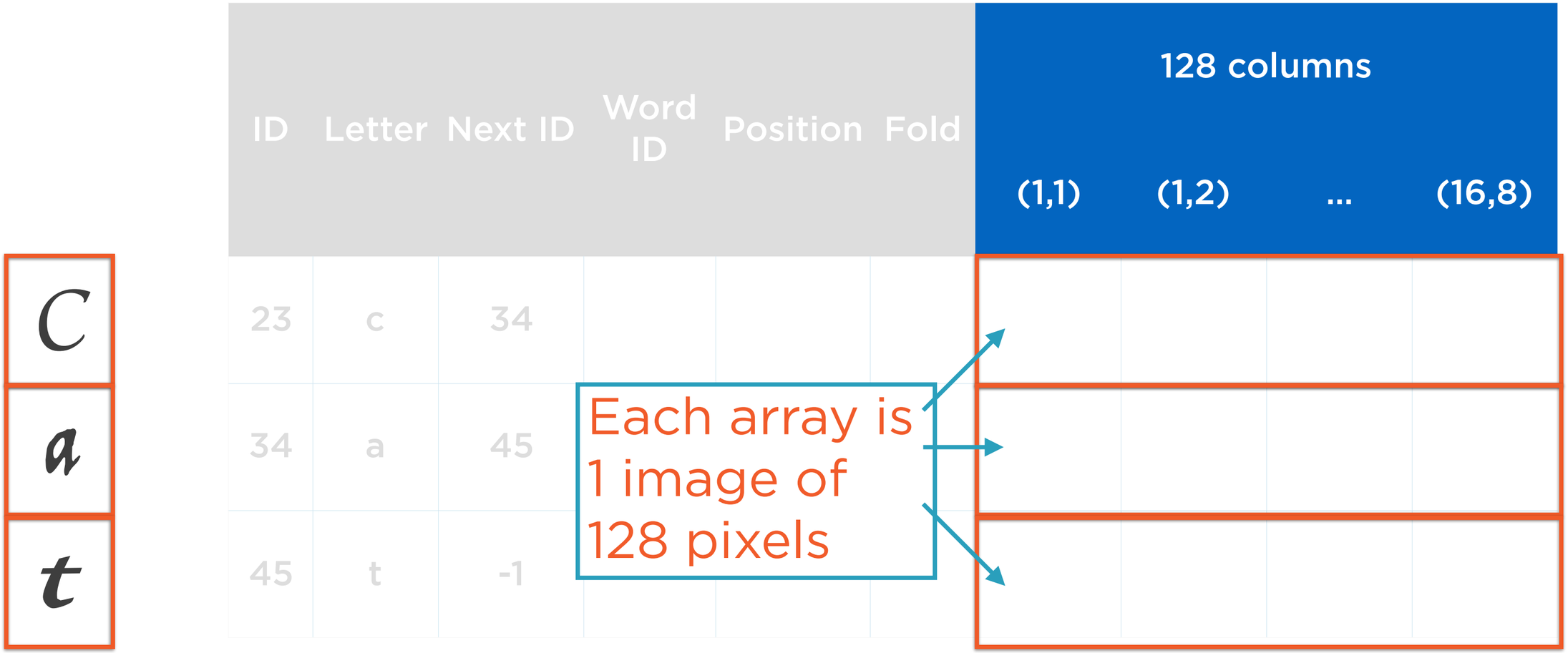
ID	Letter	Next ID	Word ID	Position	Fold	128 columns			
						(1,1)	(1,2)	...	(16,8)
23	c	34	Not used in dataset						
34	a	45							
45	t	-1							

MIT OCR File Format

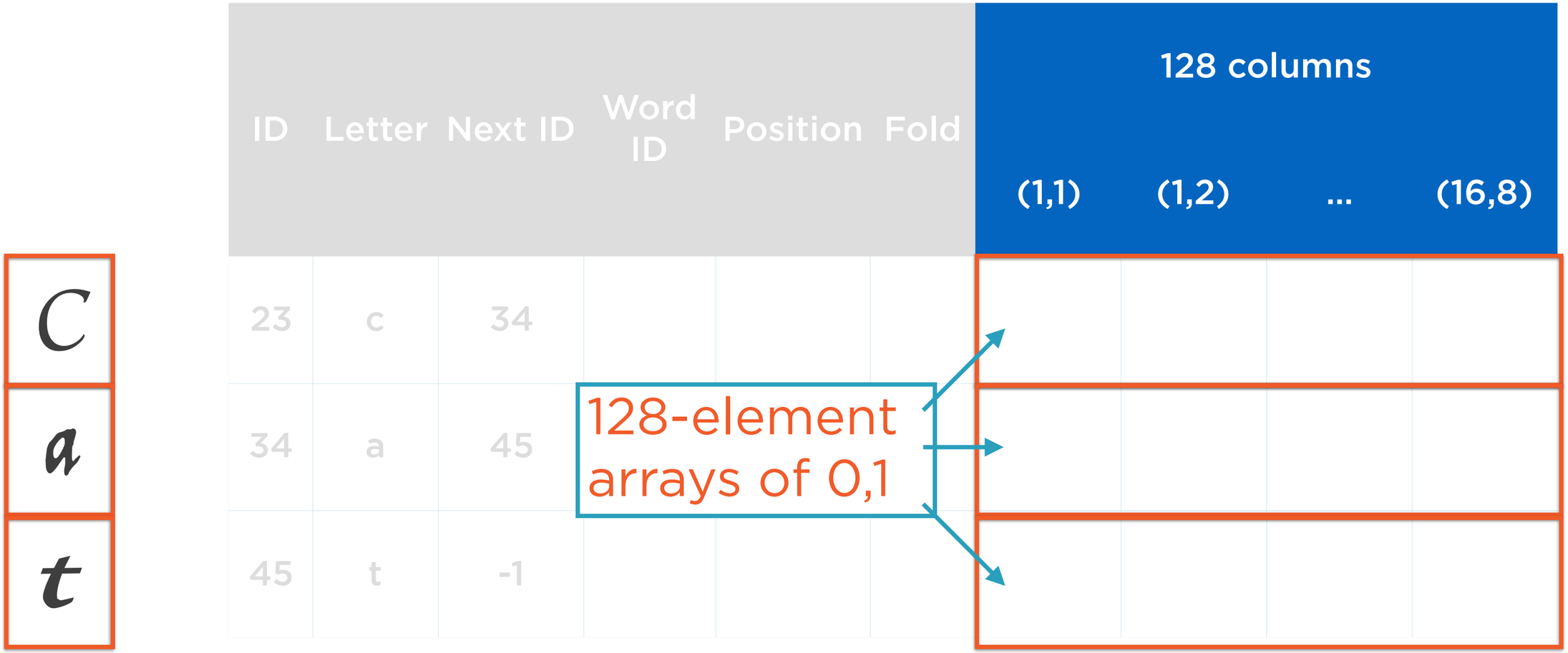
ID	Letter	Next ID	Word ID	Position	Fold	128 columns			
						(1,1)	(1,2)	...	(16,8)
23	c	34							
34	a	45							
45	t	-1							

0-9 for cross-validation

MIT OCR File Format



MIT OCR File Format



MIT OCR File Format

Images correspond to characters

	ID	Letter	Next ID	Word ID	Position	Fold	128 columns			
							(1,1)	(1,2)	...	(16,8)
C	23	c	34							
a	34	a	45							
t	45	t	-1							

Features and Labels for OCR

Features and Labels

Labels

ID	Letter	Next ID	Word ID	Position	Fold	128 columns			
						(1,1)	(1,2)	...	(16,8)
23	c	34							
34	a	45							
45	t	-1							

One-hot Encoded Labels

Letter	a	b	c	...	t	...	z
c			1				
a	1						
t					1		

26 elements

One-hot Encoded Labels

Letter	a	b	c	...	t	...	z
c			1				
a	1						
t					1		

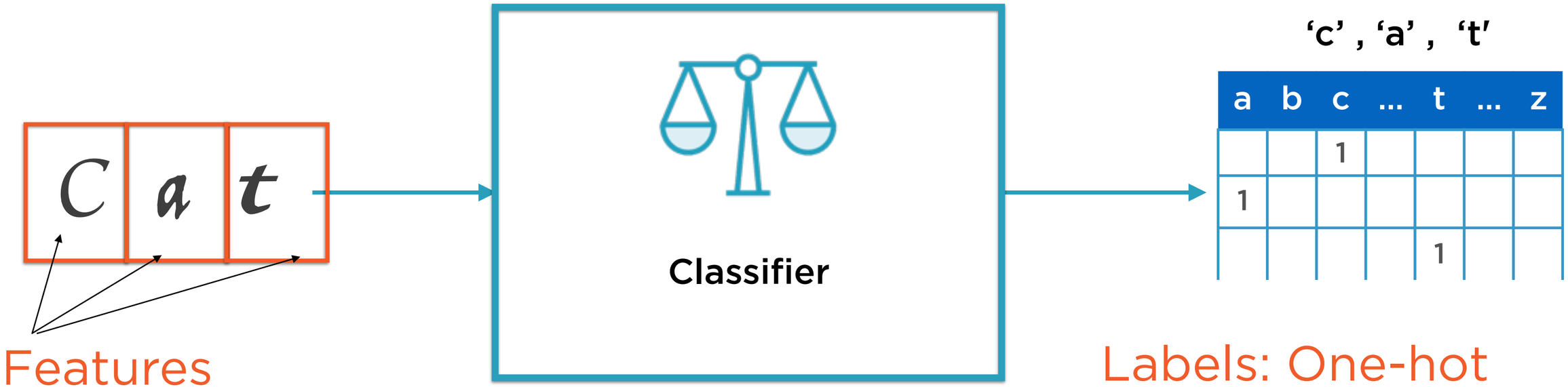
One-hot Encoded Labels

Letter	a	b	c	...	t	...	z
c			1				
a	1						
t					1		

One-hot Encoded Labels

Letter	a	b	c	...	t	...	z
c			1				
a	1						
t					1		

Features and Labels



Labels: One-hot encoded
(26 element array)

Features

	ID	Letter	Next ID	Word ID	Position	Fold	128 columns			
							(1,1)	(1,2)	...	(16,8)
C	23	c	34							
a	34	a	45							
t	45	t	-1							

Each array is 1 image of 128 pixels

Features

	ID	Letter	Next ID	Word ID	Position	Fold	128 columns			
							(1,1)	(1,2)	...	(16,8)
C	23	c	34							
a	34	a	45							
t	45	t	-1							

128-element arrays of 0,1

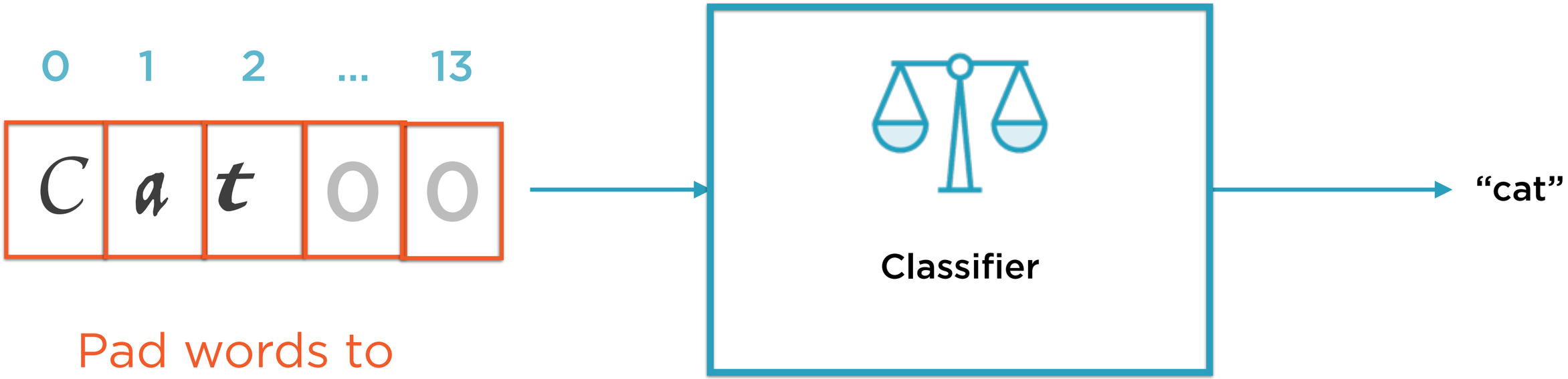
RNN Architecture

RNNs

Each word is a sequence of characters

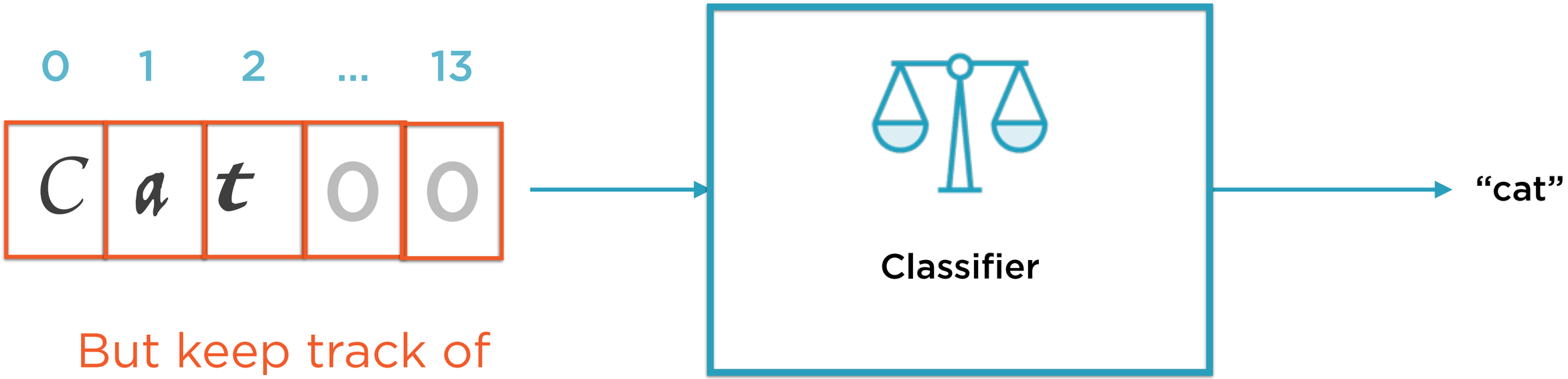
RNNs are a natural choice for sequential data

OCR Word Recognition



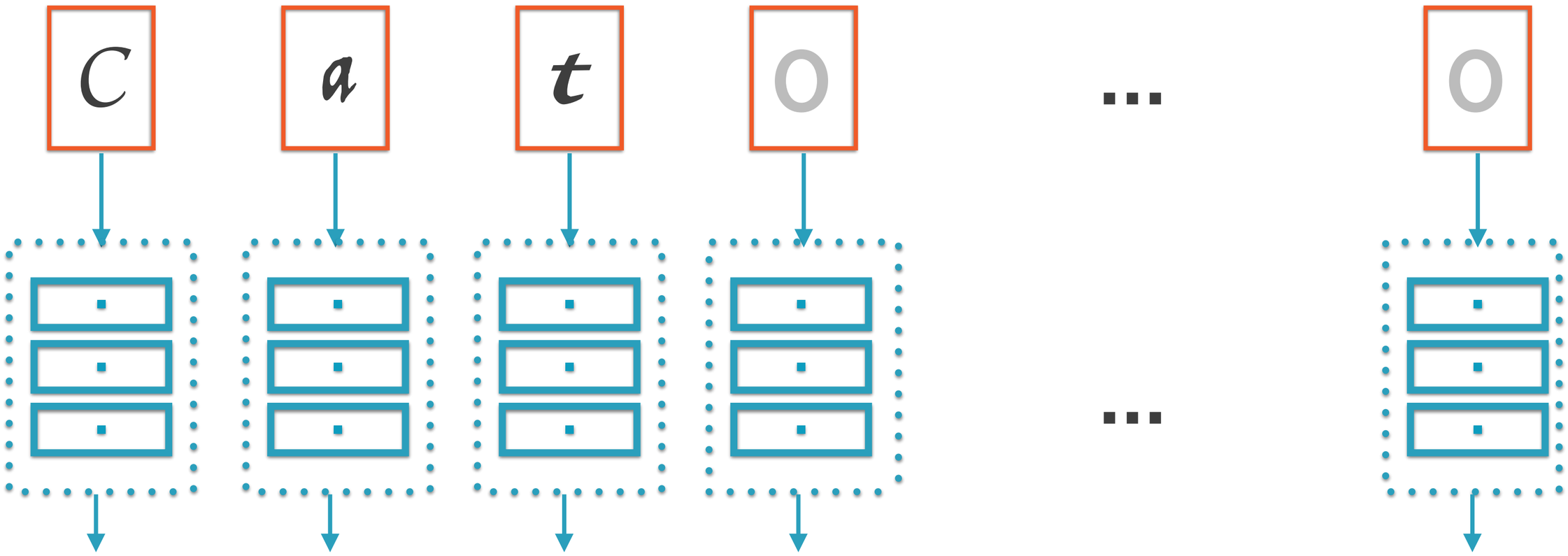
Pad words to all be of 14 characters

OCR Word Recognition

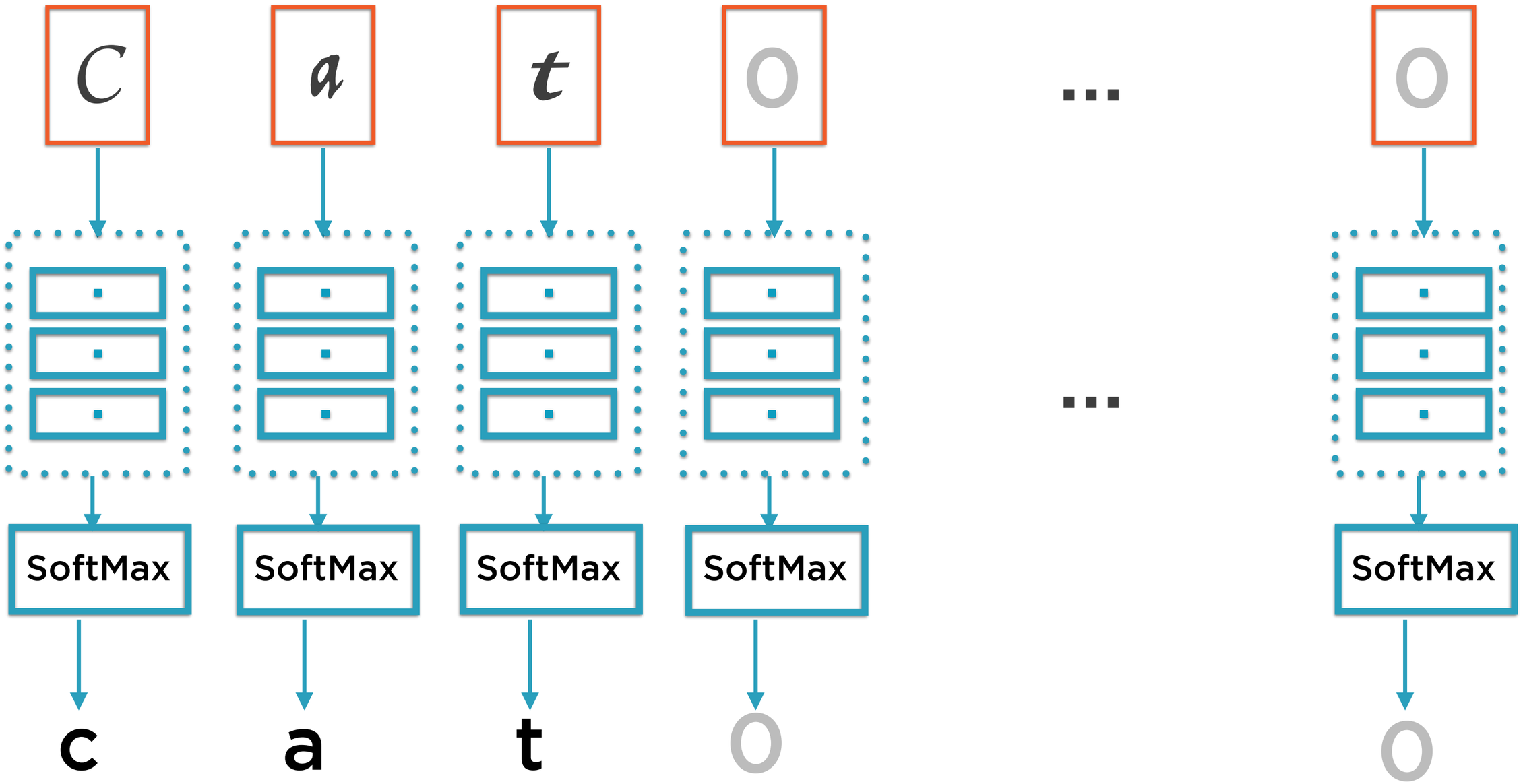


But keep track of sequence lengths of each word

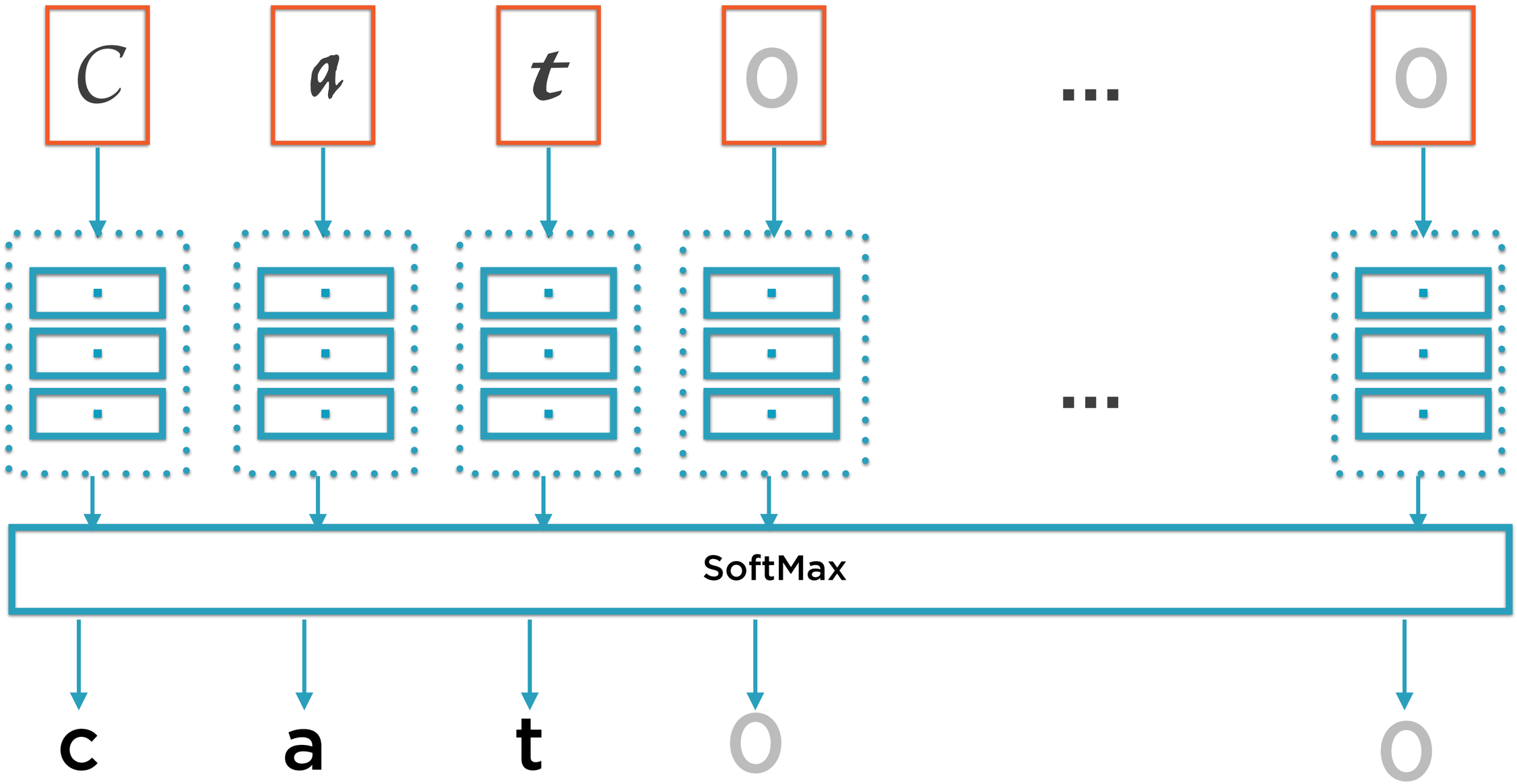
RNN Architecture



Softmax for Prediction

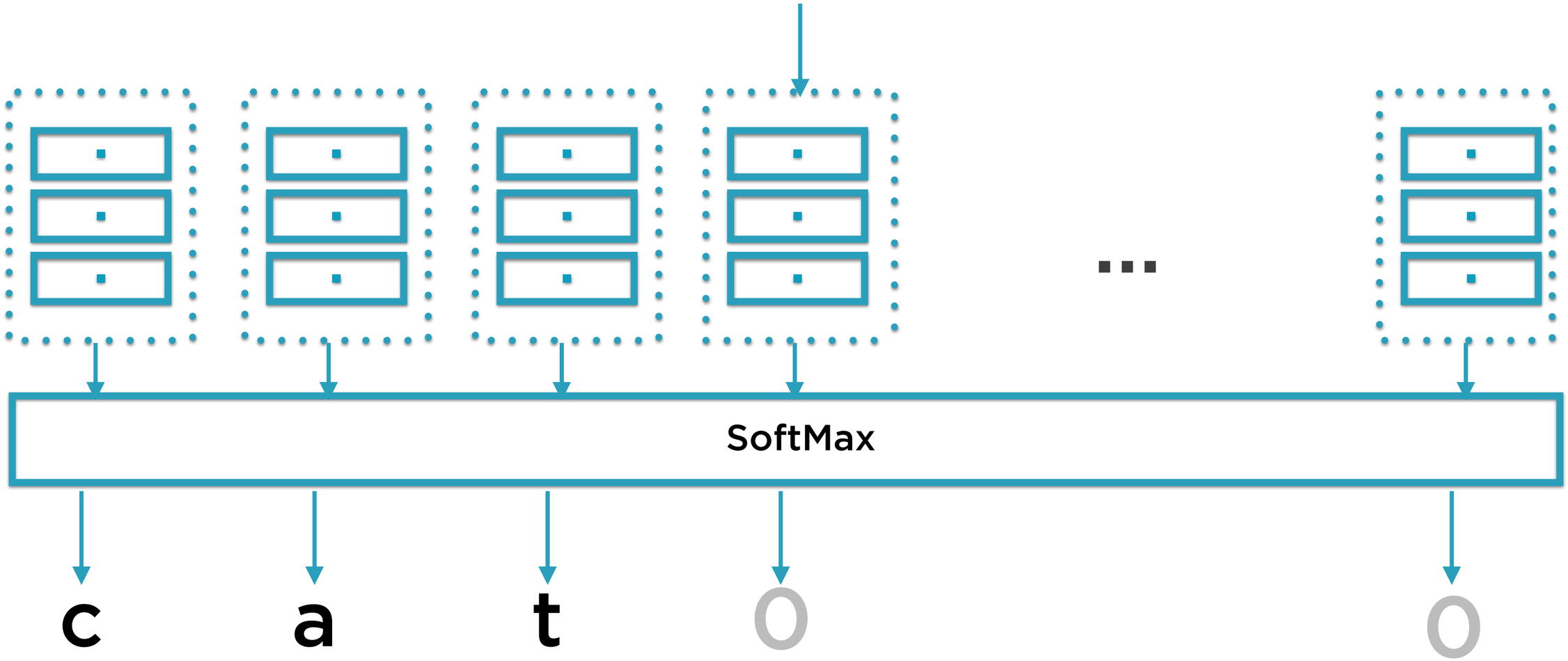


Shared Softmax



Input Tensor for Training

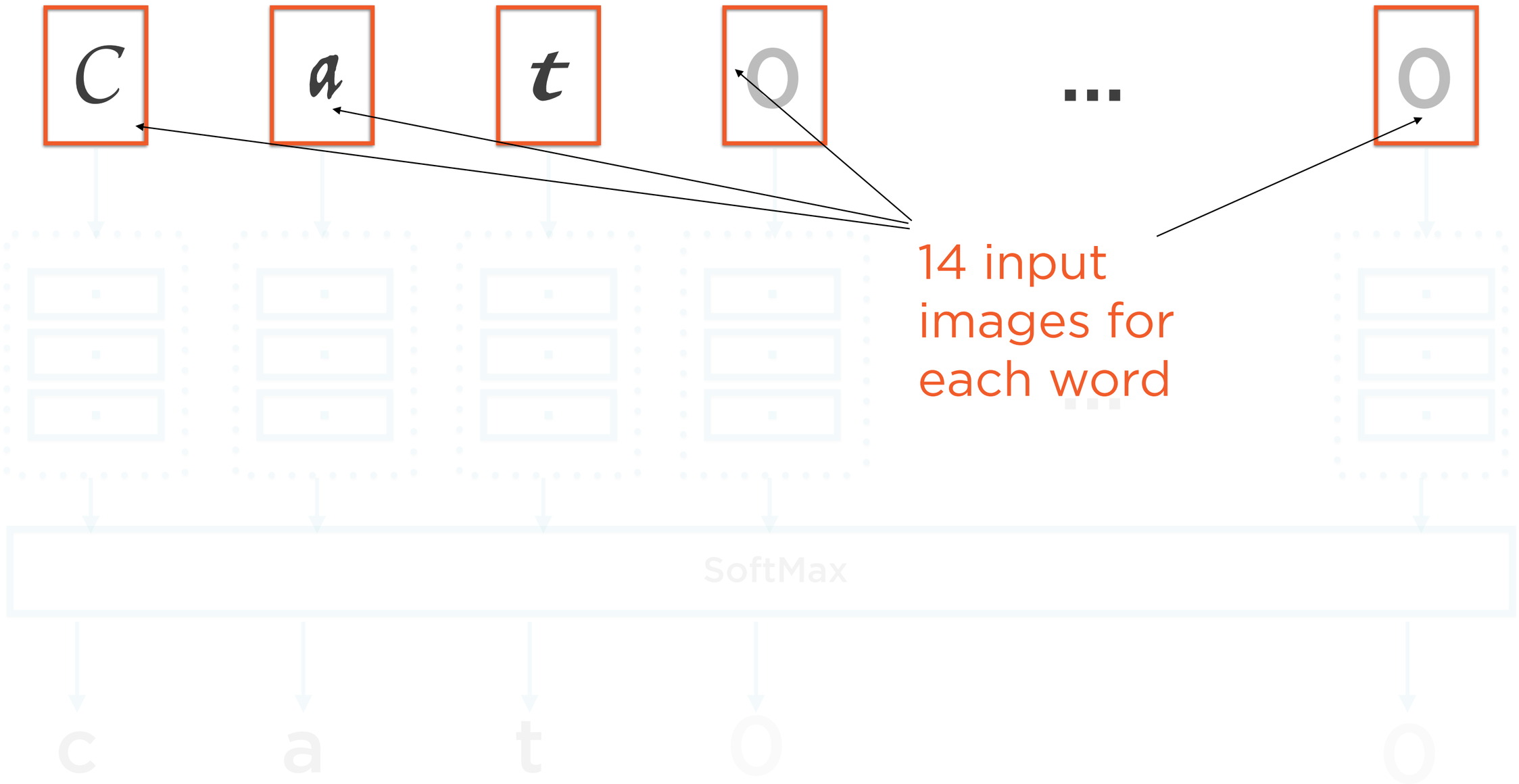
[batch_size, 14, 128]



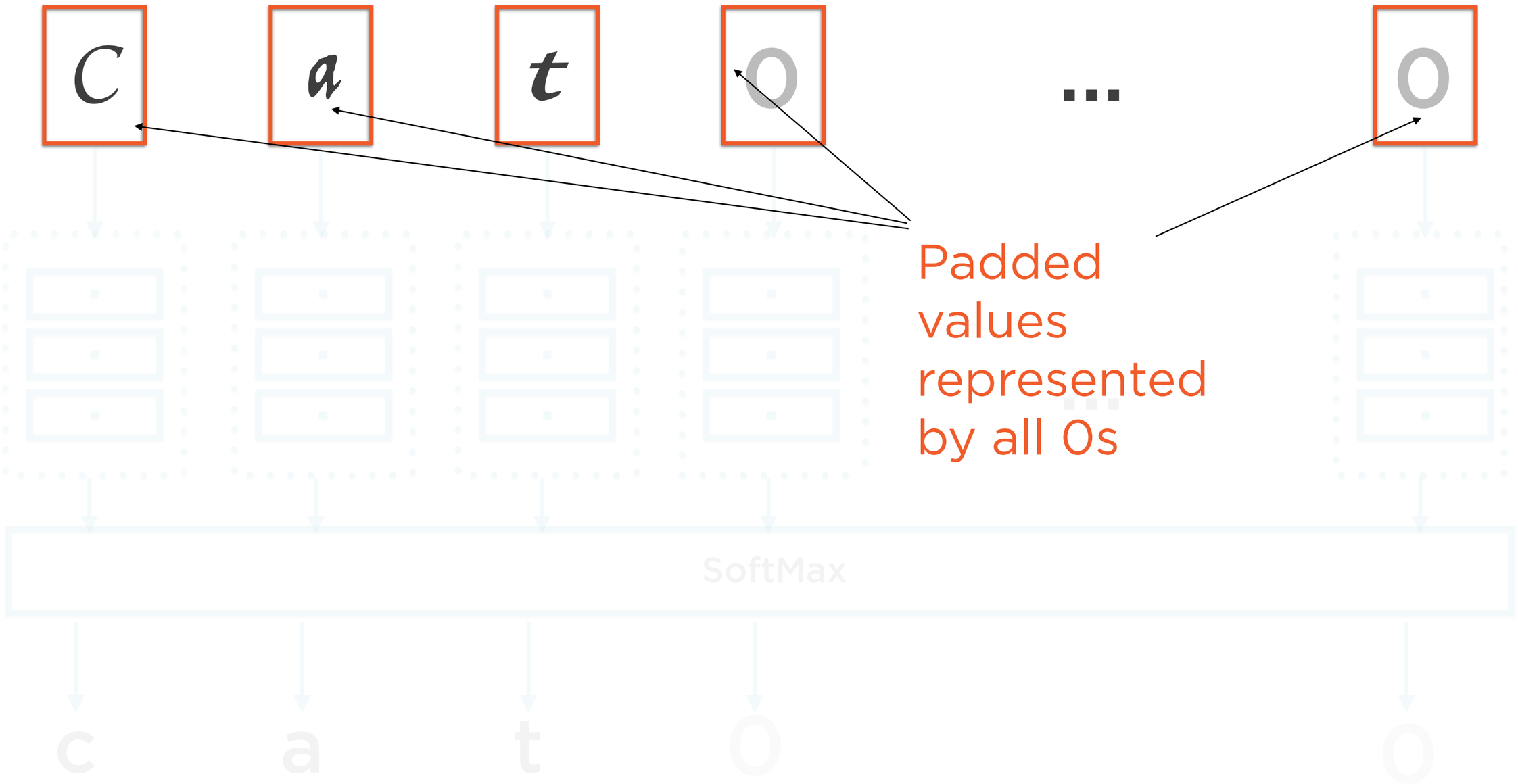
RNN Architecture

RNN Architecture

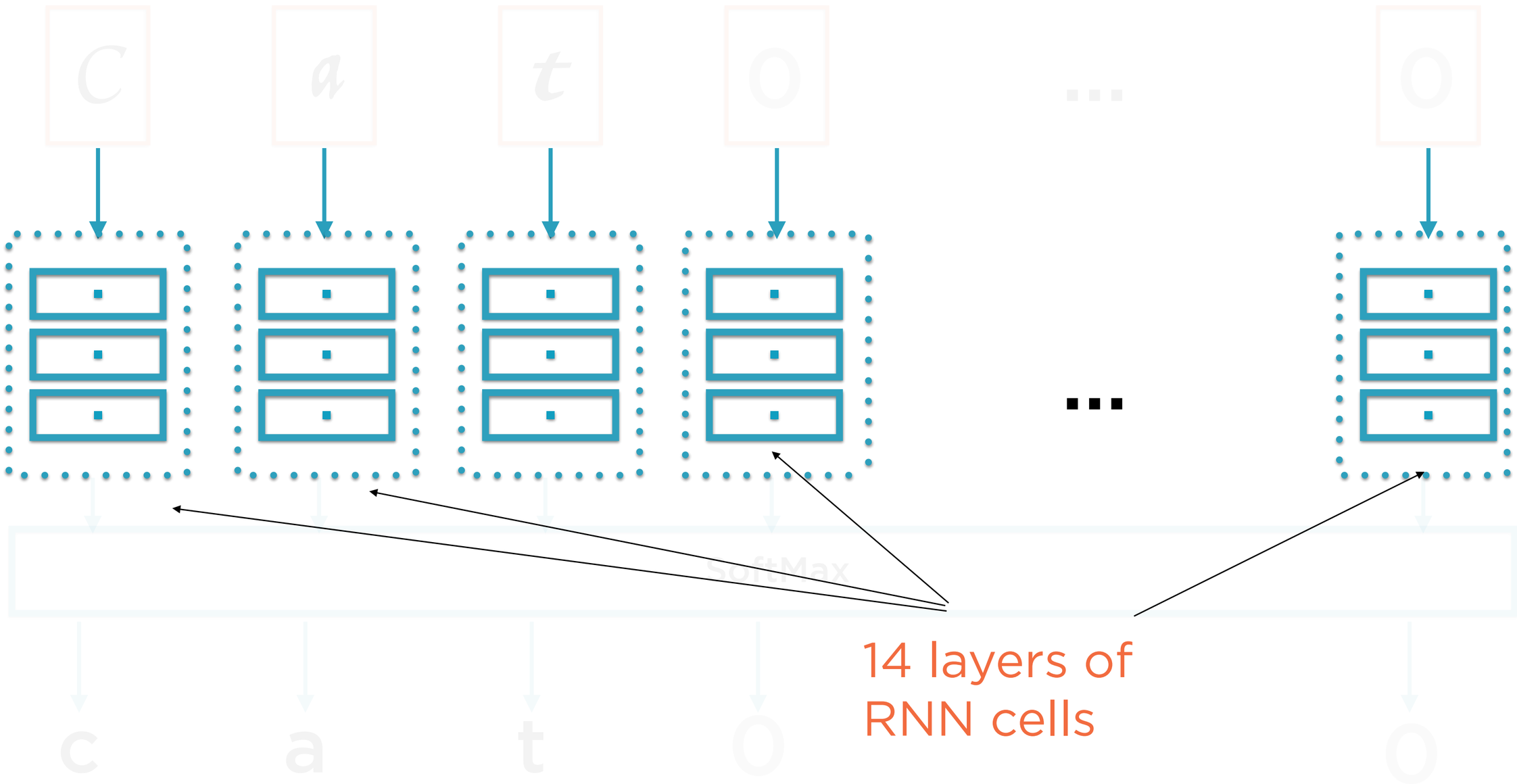
RNN Architecture



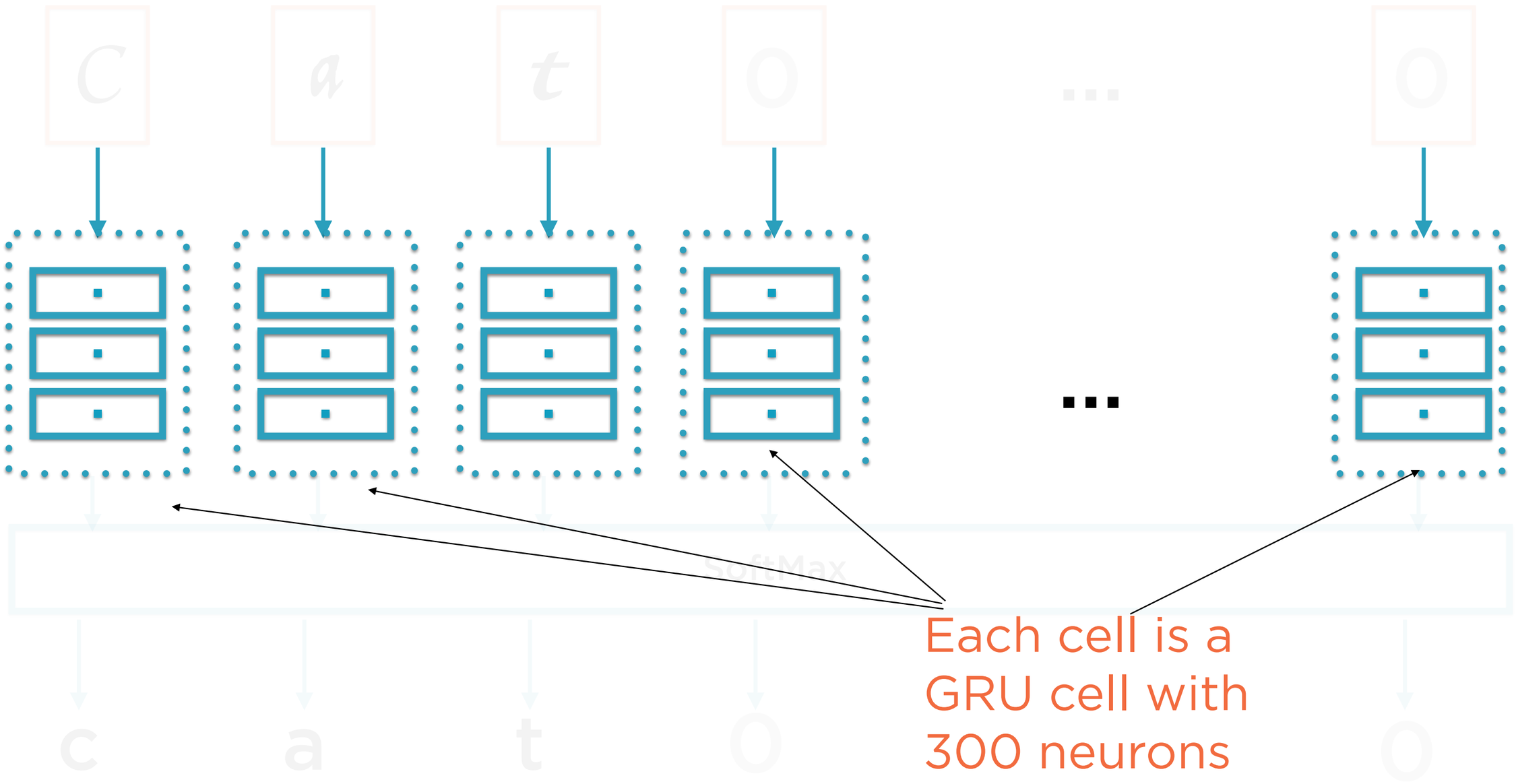
RNN Architecture



RNN Architecture

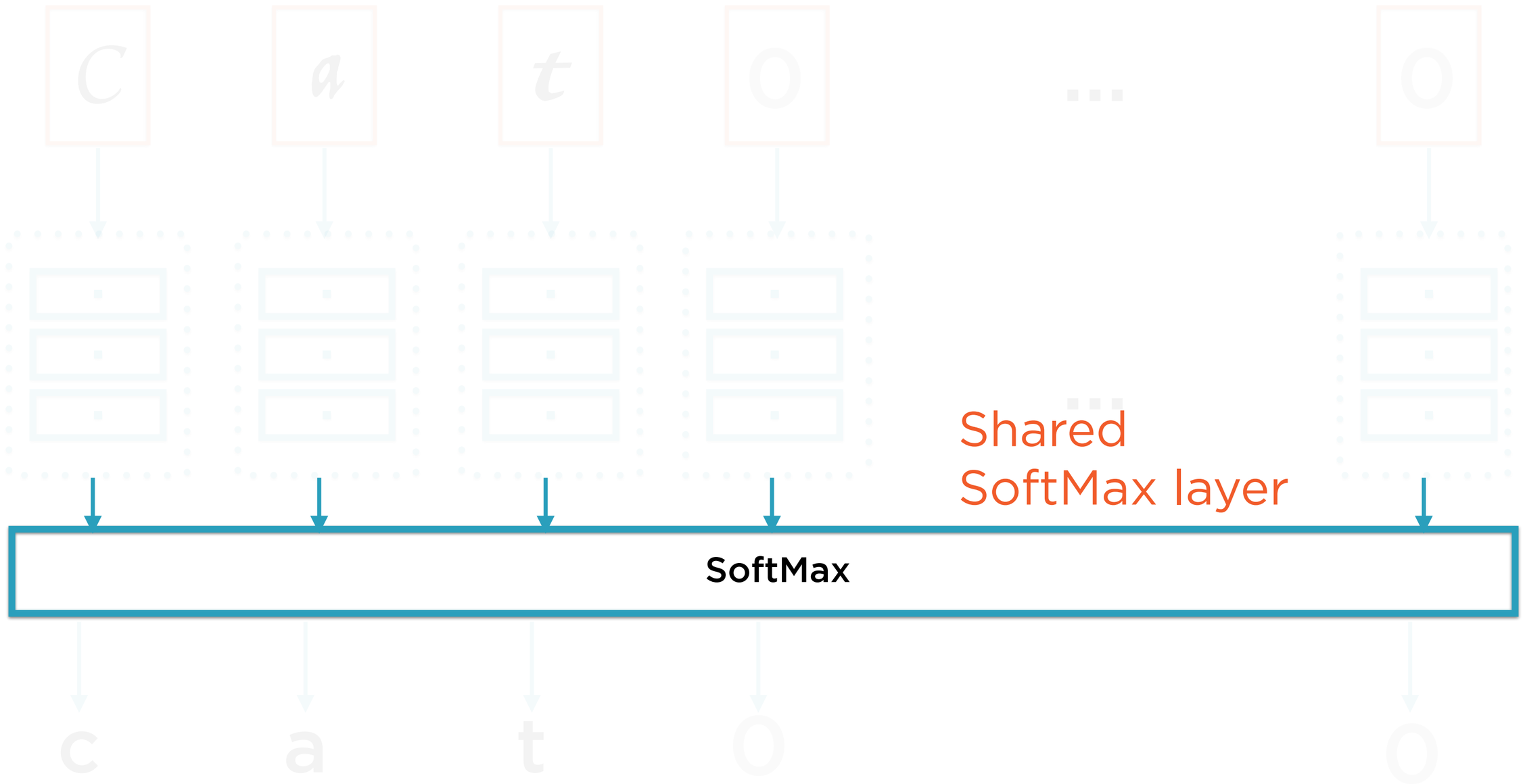


RNN Architecture

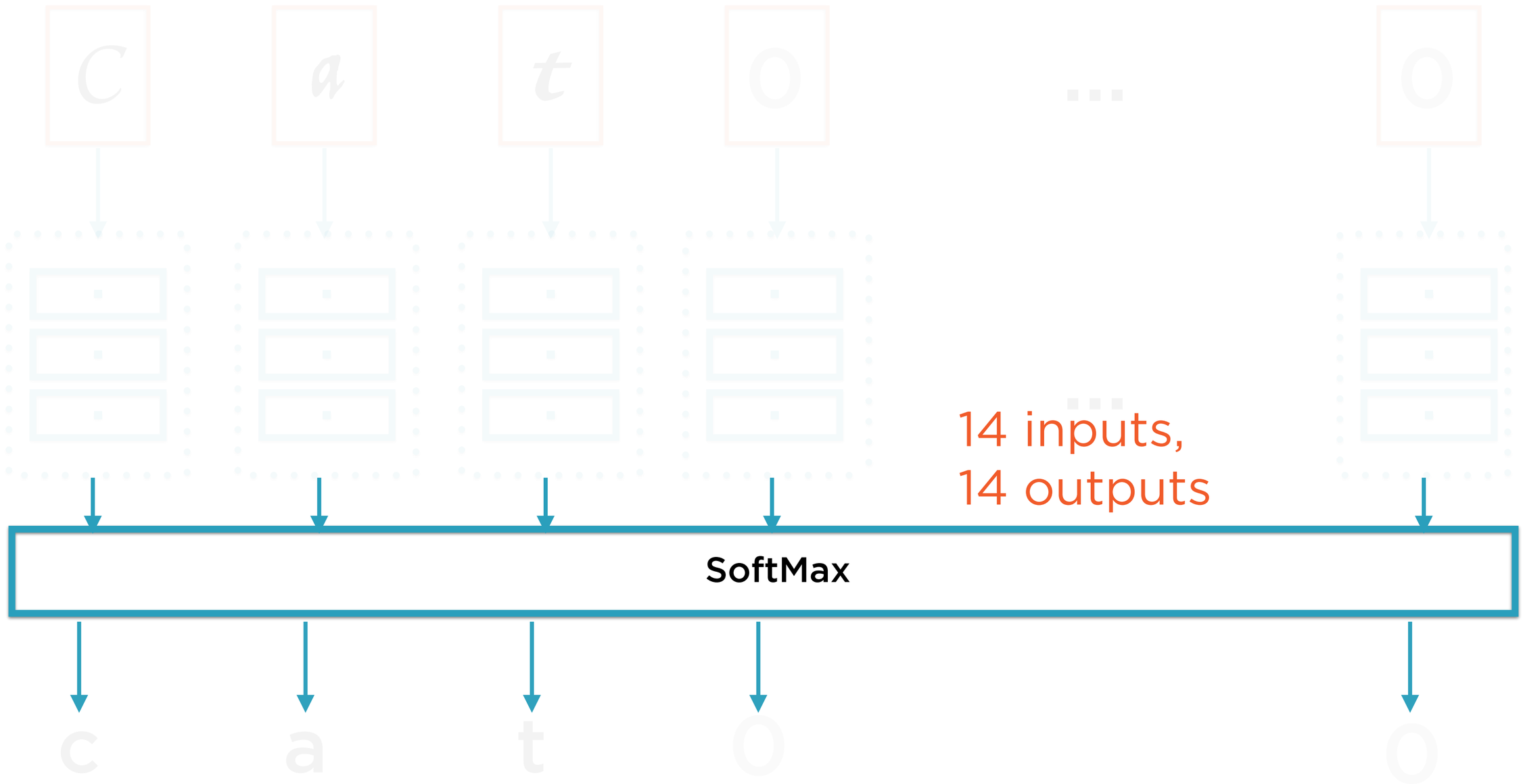


Each cell is a GRU cell with 300 neurons

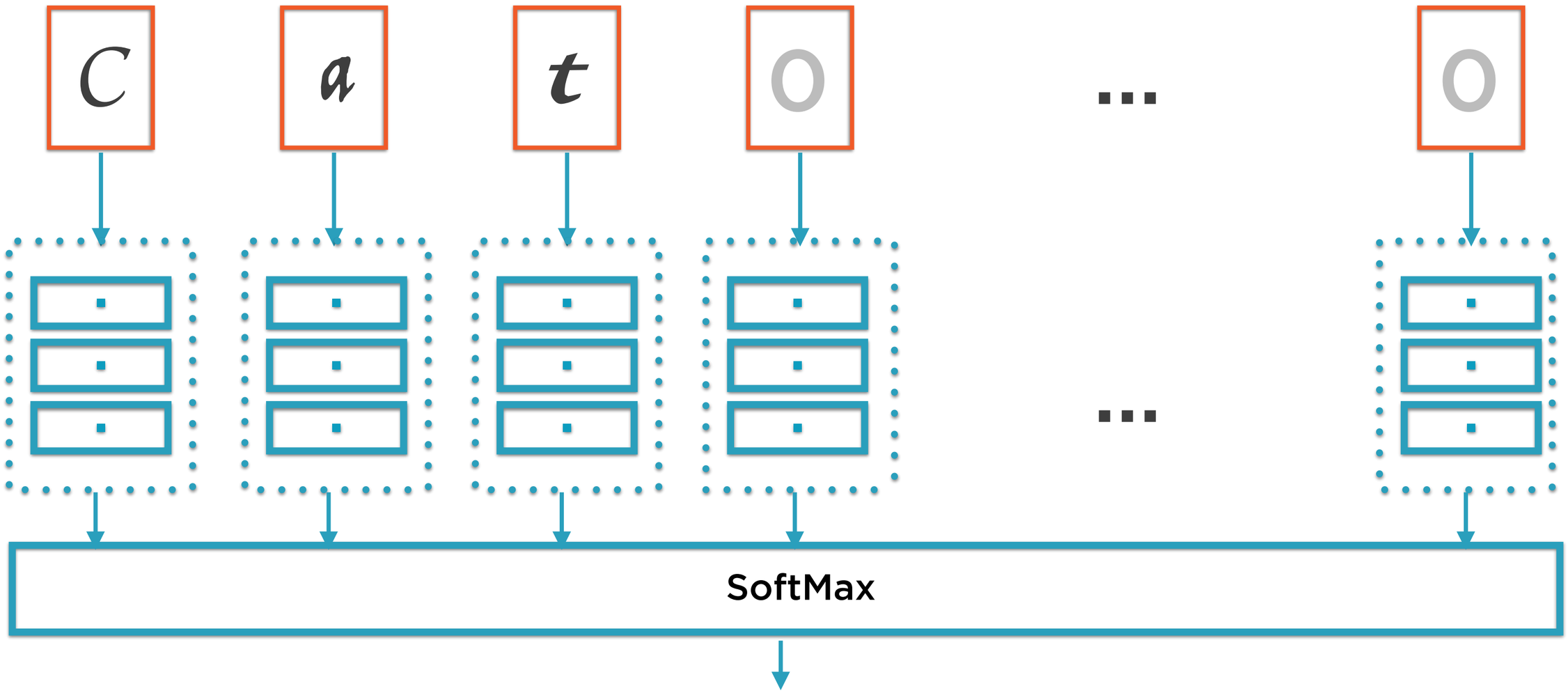
RNN Architecture



RNN Architecture

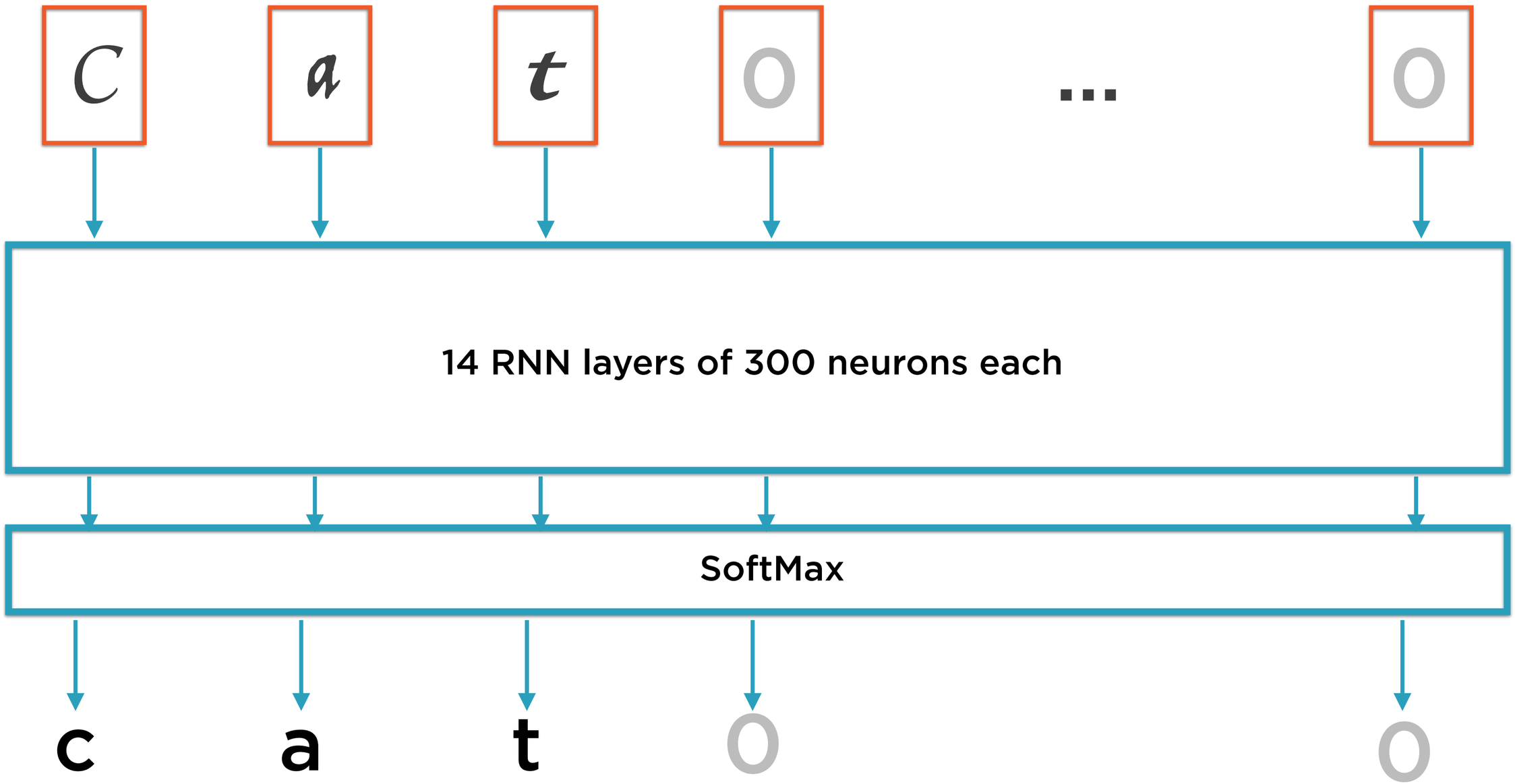


Output Tensor for Predicted Labels



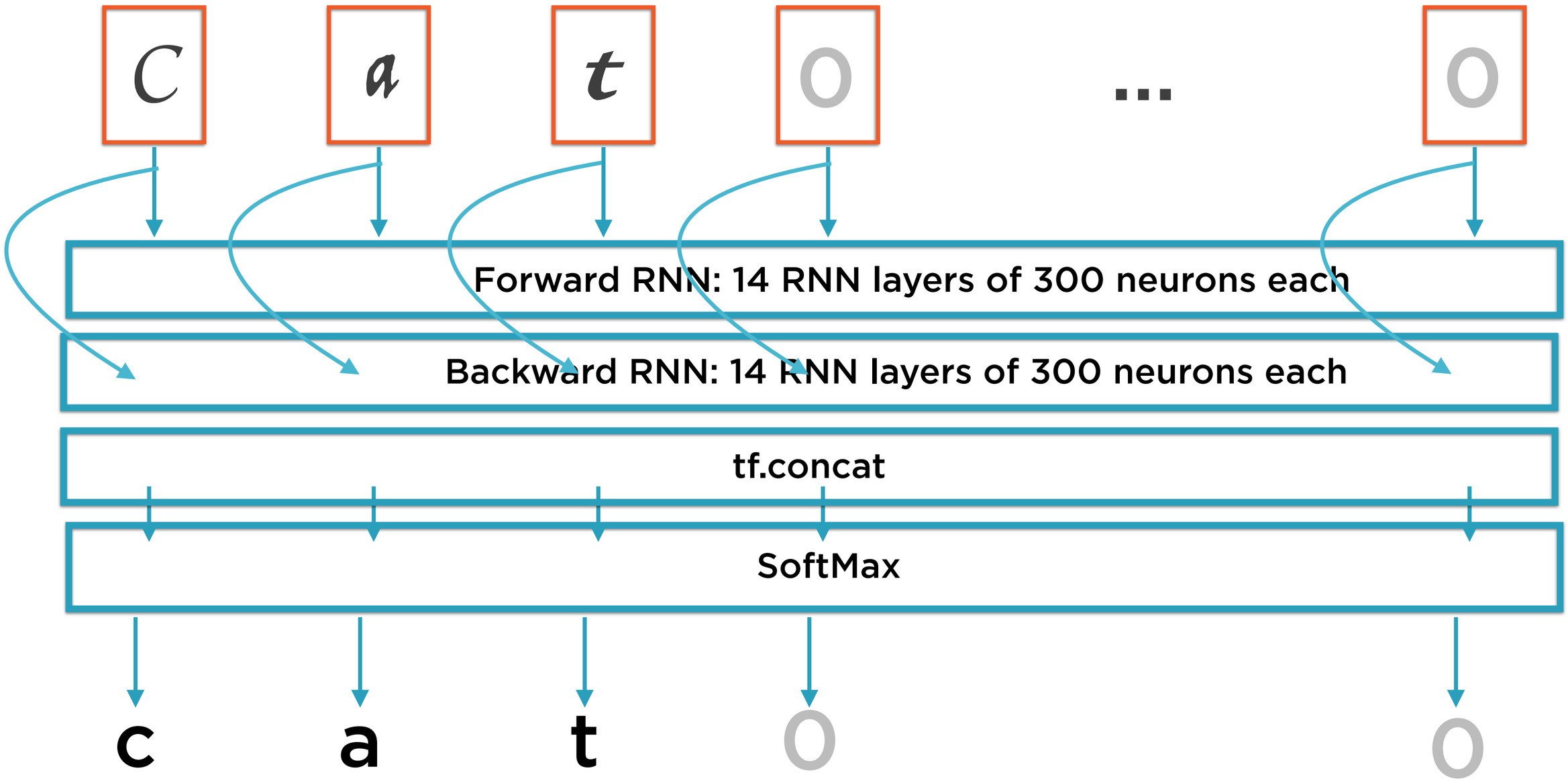
[batch_size, 14, 26]

Conventional RNN Architecture

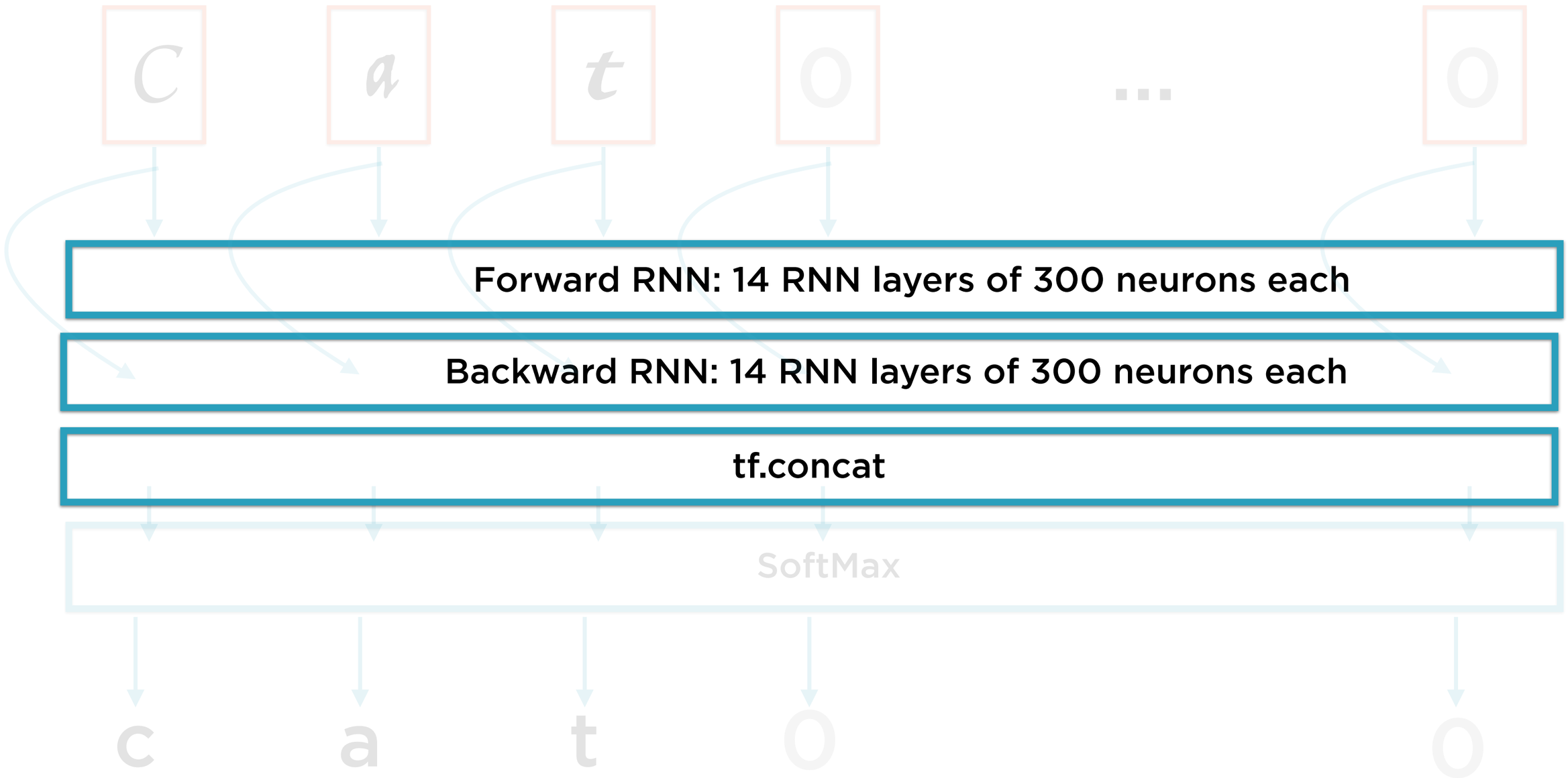


Bidirectional RNN Architecture

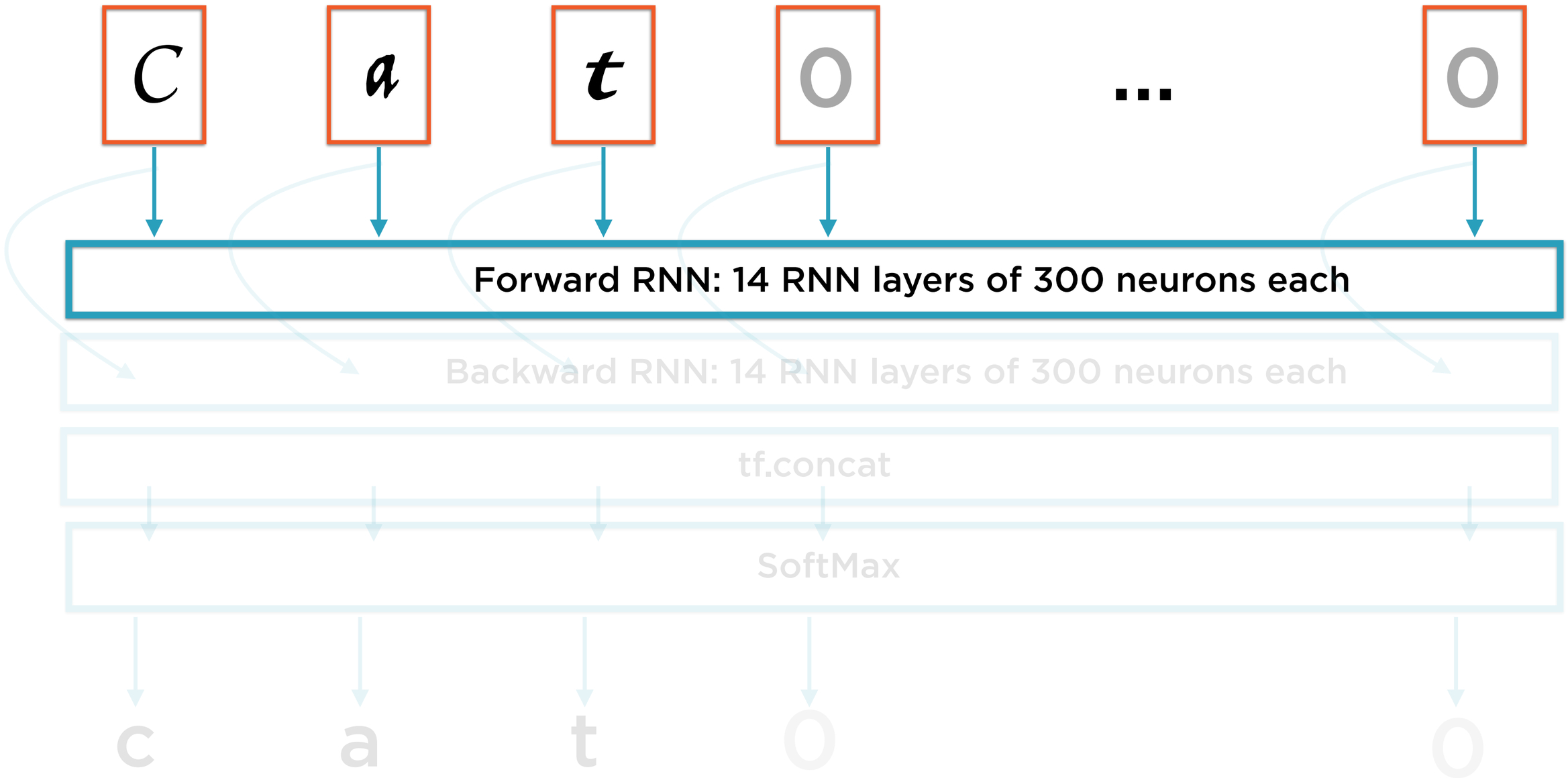
Bidirectional RNN Architecture



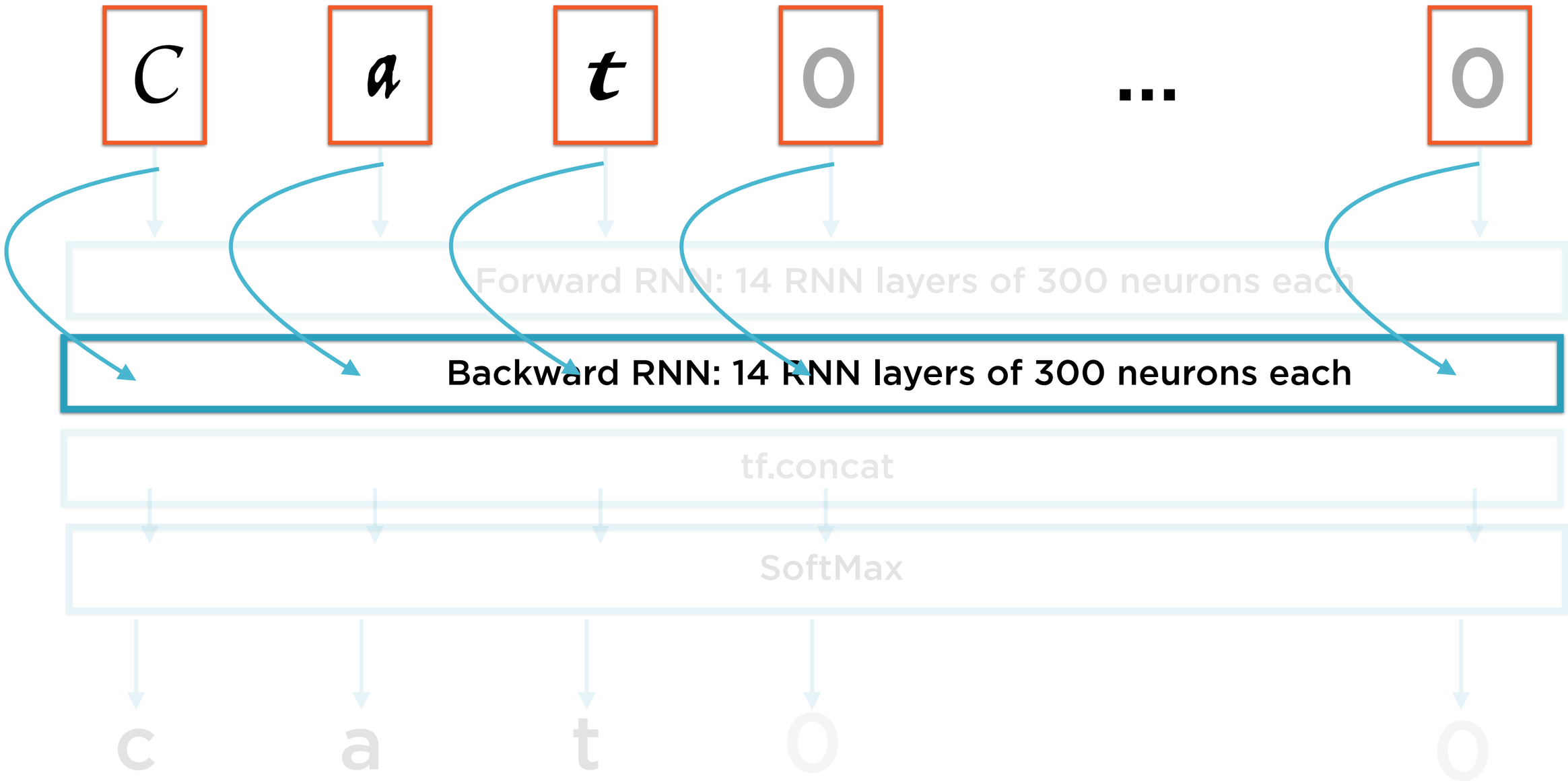
Bidirectional RNN Architecture



Bidirectional RNN Architecture

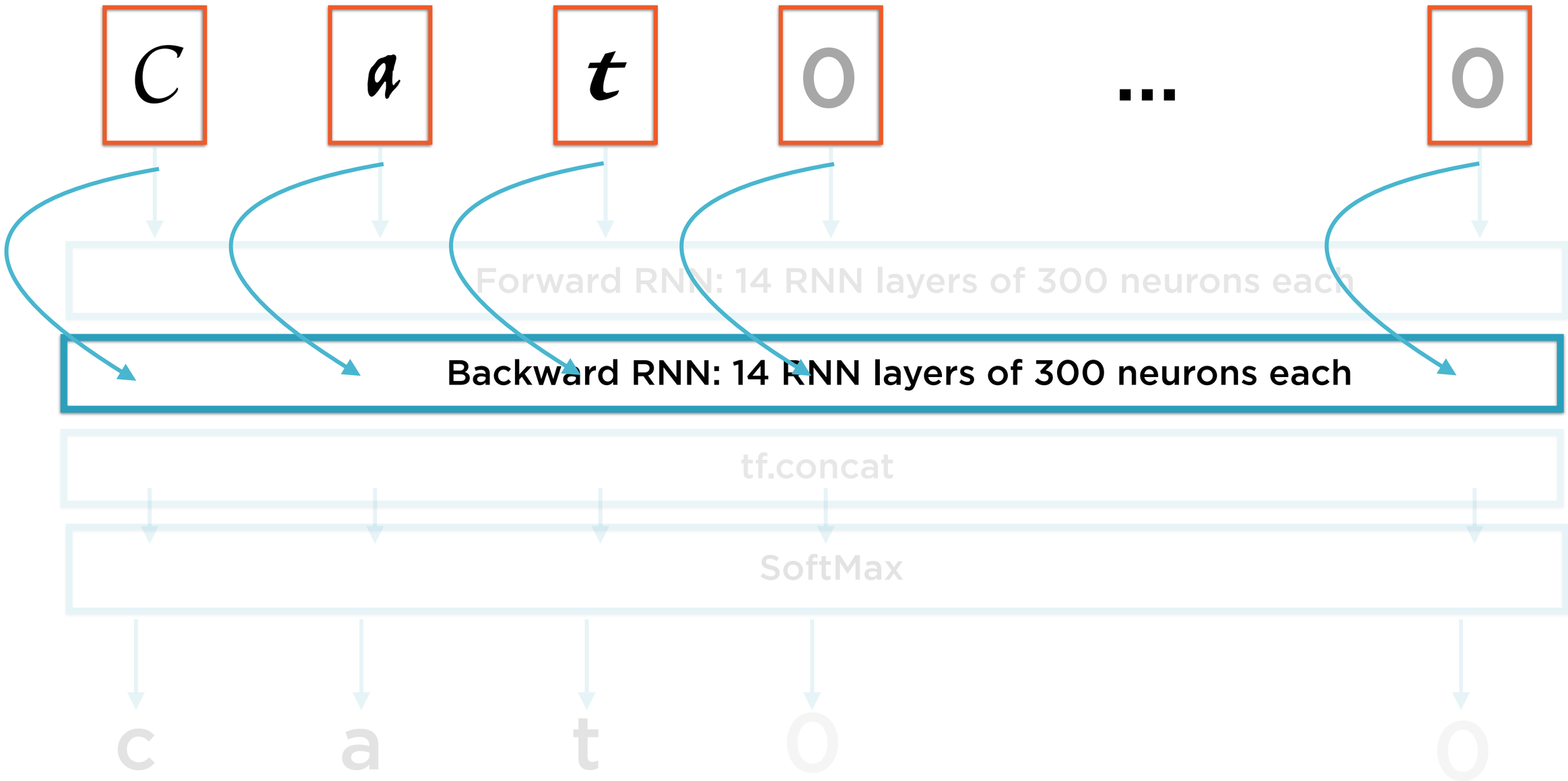


Bidirectional RNN Architecture

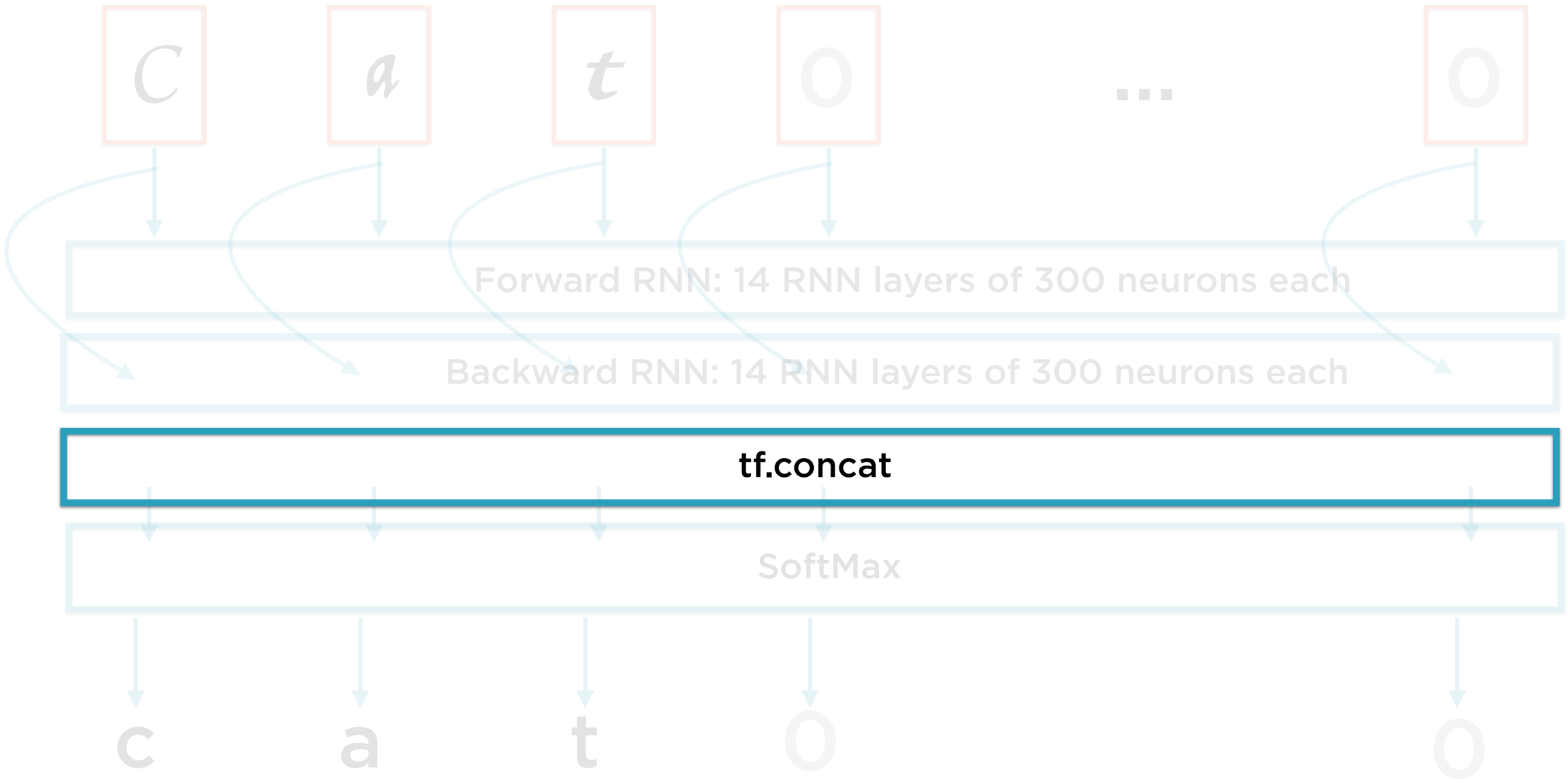


The input sequence is **reversed** and passed in through a conventional RNN

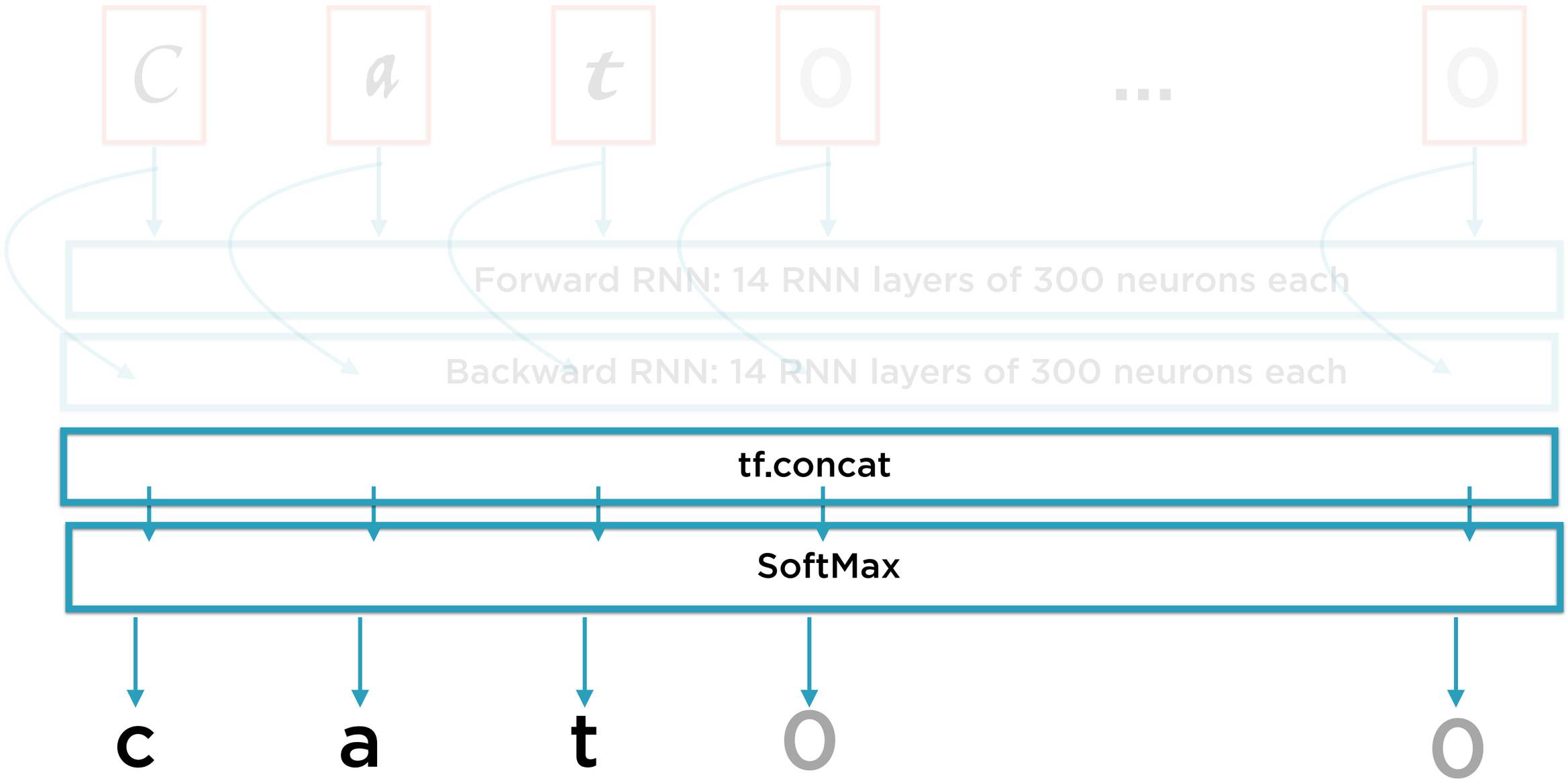
Bidirectional RNN Architecture



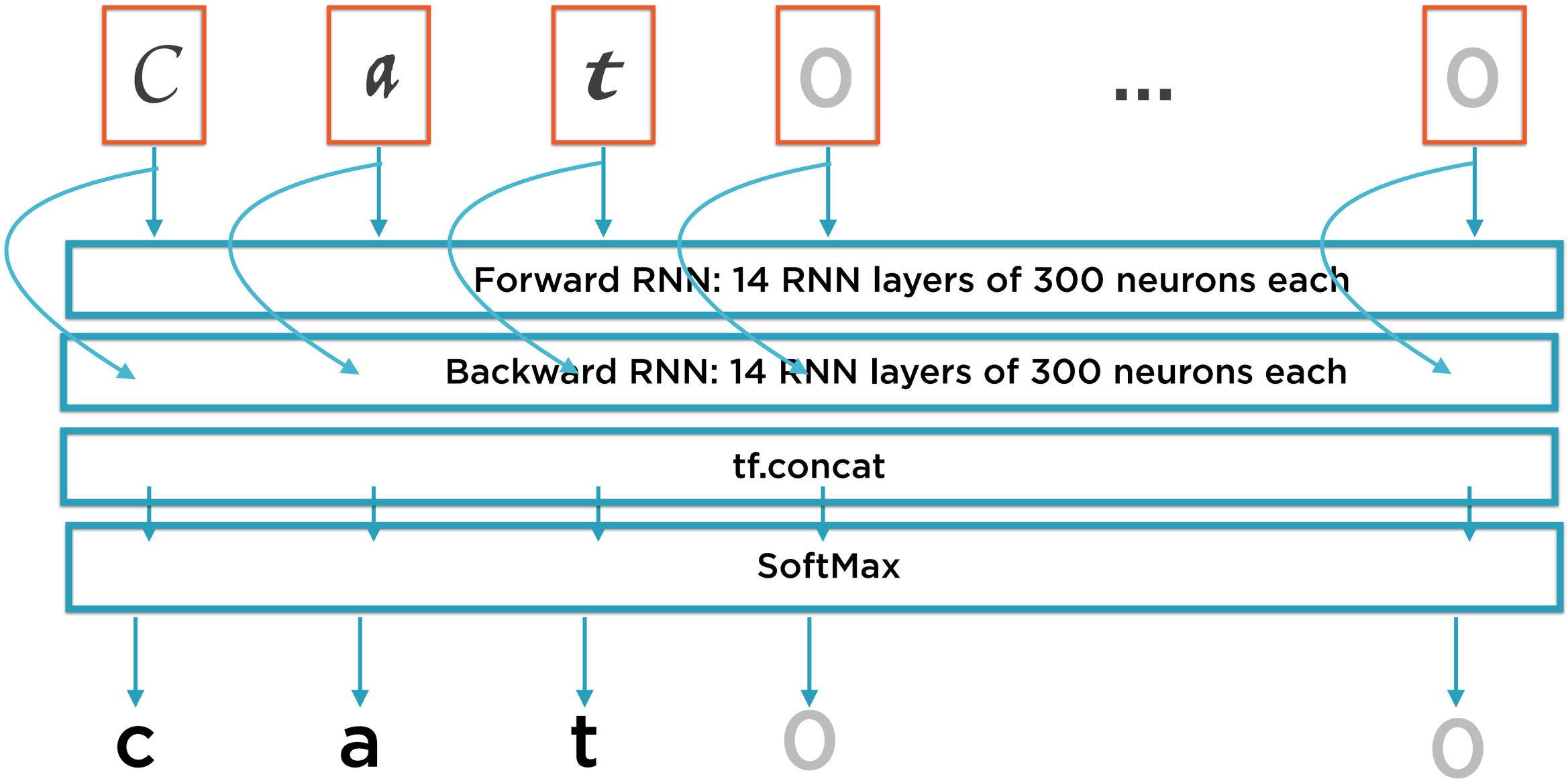
Bidirectional RNN Architecture



Bidirectional RNN Architecture



Bidirectional RNN Architecture



Summary

Recurrent Neural Networks (RNNs) work well with sequential data

Neurons in RNNs maintain internal state

This helps RNNs learn and remember the past

LSTM and GRU neurons are recurrent neurons that can remember even the distant past

Bidirectional RNNs are a type of RNN that make two passes over sequential input data

Bidirectional RNNs out-perform RNNs in word recognition and speech recognition