Applying RNNs to Character Prediction for Text Generation

Janani Ravi

CO-FOUNDER, LOONYCORN

www.loonycorn.com

Overview

Language modeling is an important area of ML research

Text prediction is one of several classic language modeling problems

Multi-RNNs are a specific type of RNN that work well in language modeling

A key technique is smartly re-initialising state of the RNN during prediction

An evaluation metric called perplexity is used to assess predictive performance

Language Modeling

Two Familiar Problems

Word Embeddings

Express a word in terms of context
in numeric form

Sentiment Analysis

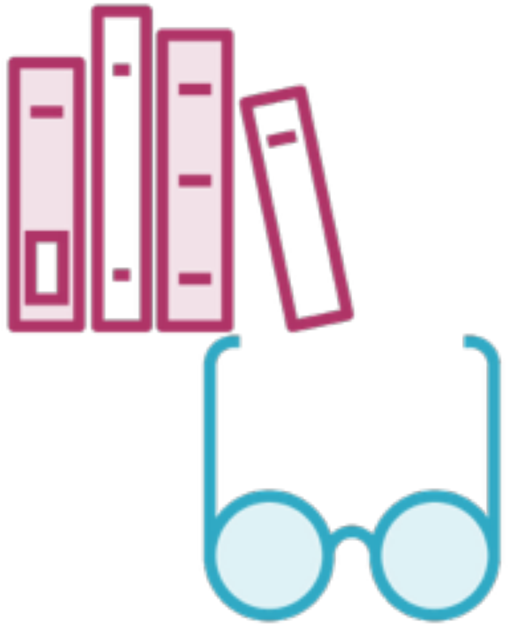
Classify a set of words

Types of Machine Learning Problems

Classification

Regression

Clustering



Rule-extraction

Types of Machine Learning Problems

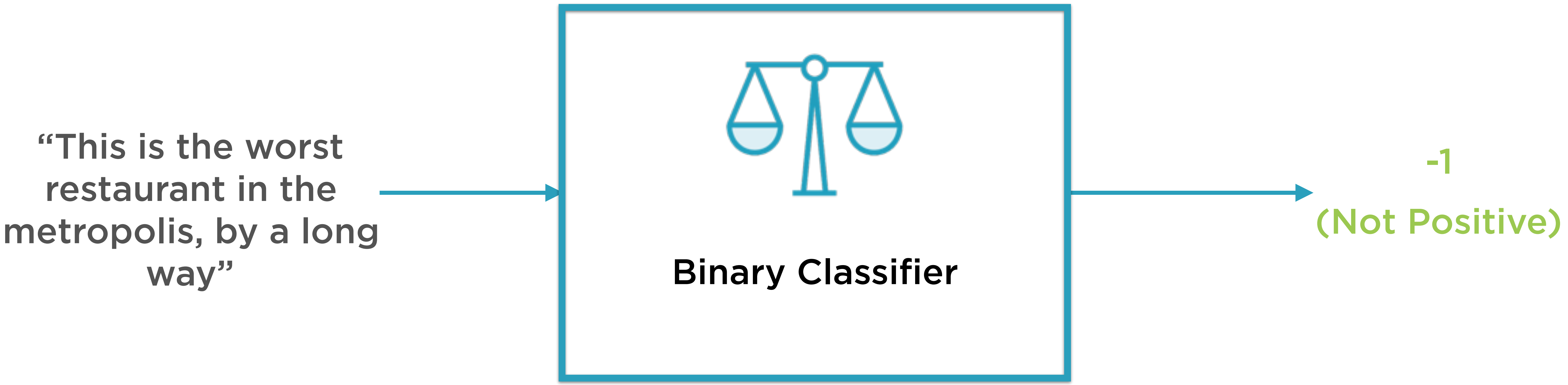
Classification

Regression

Clustering

Rule-extraction

Sentiment Analysis as Binary Classification



Two Familiar Problems

Word Embeddings

Express a word in terms of context in numeric form

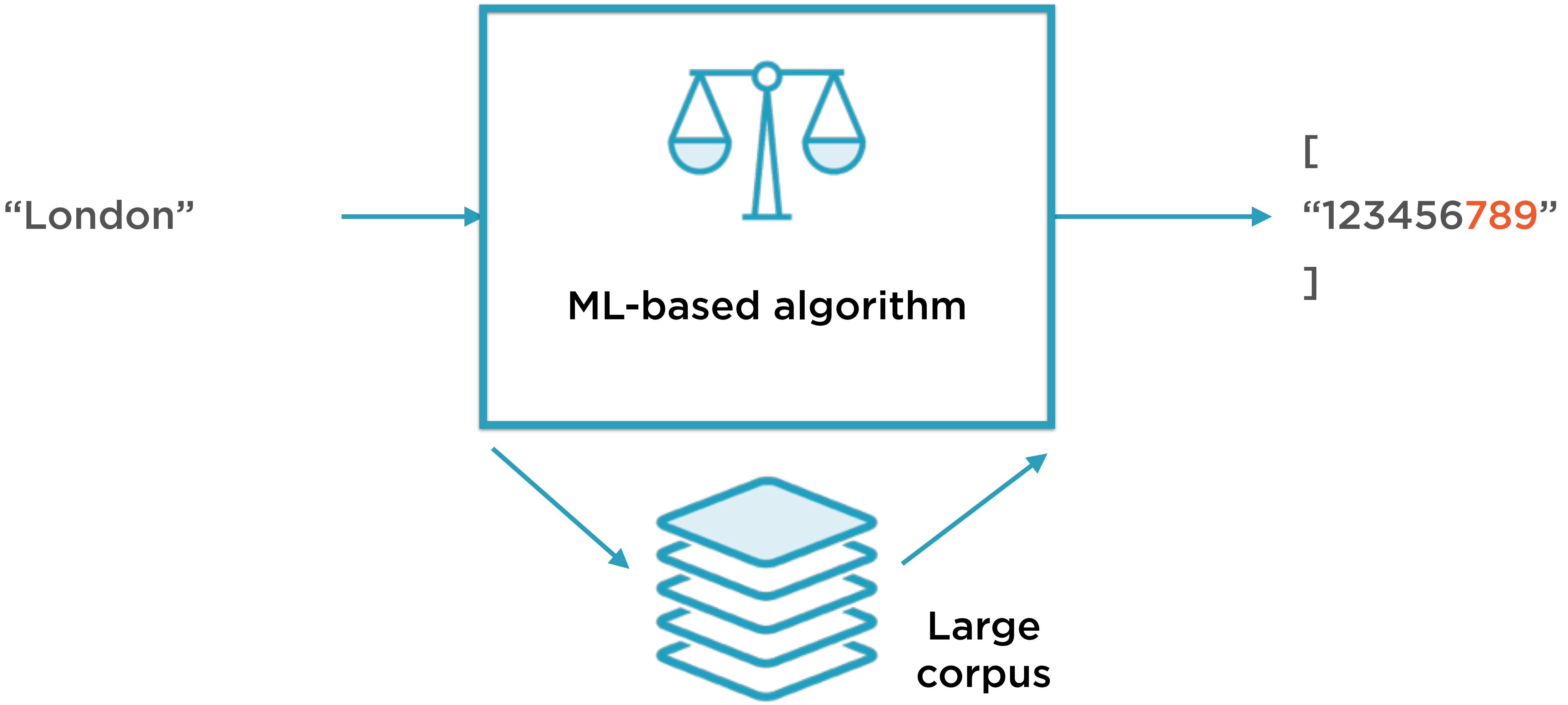
Sentiment Analysis

Classify a set of words

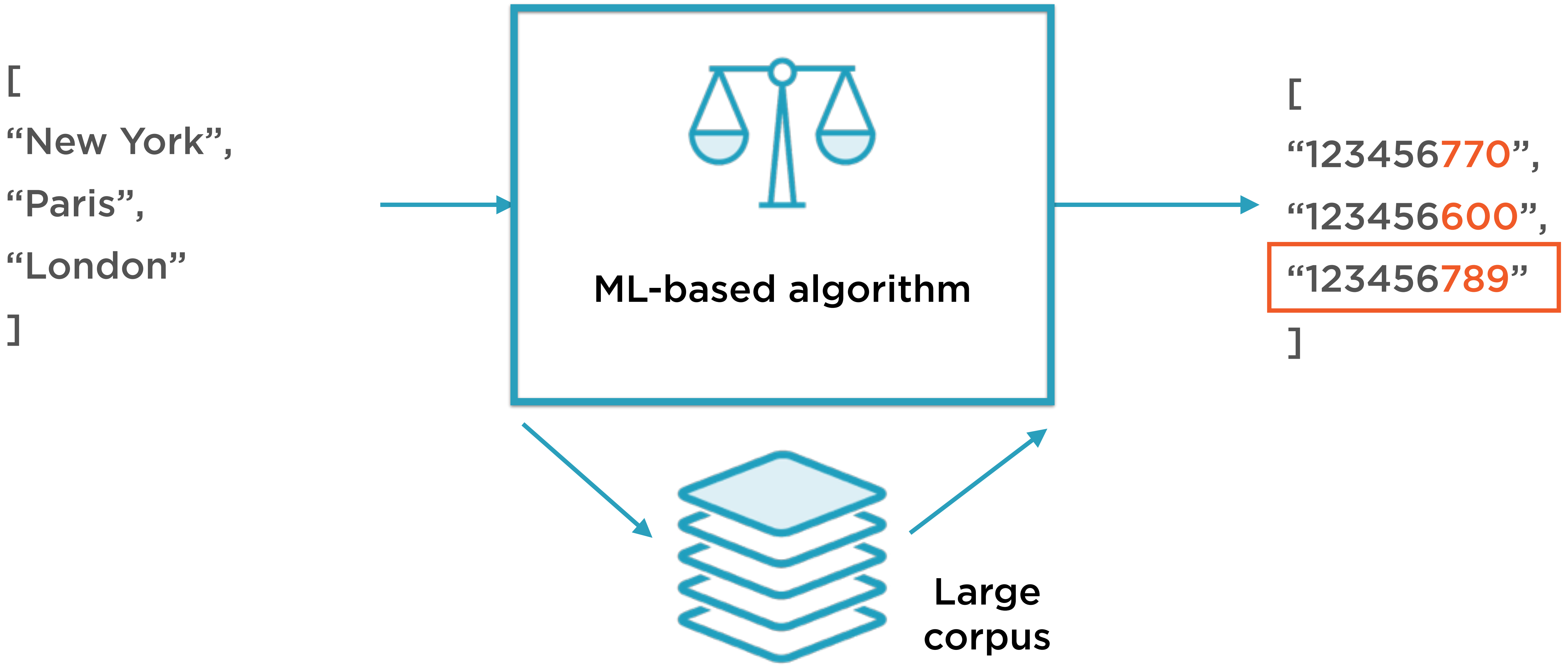
Given words from its context,
predict the word

Given a word, **predict the words in its context**

Prediction-based Word Embeddings

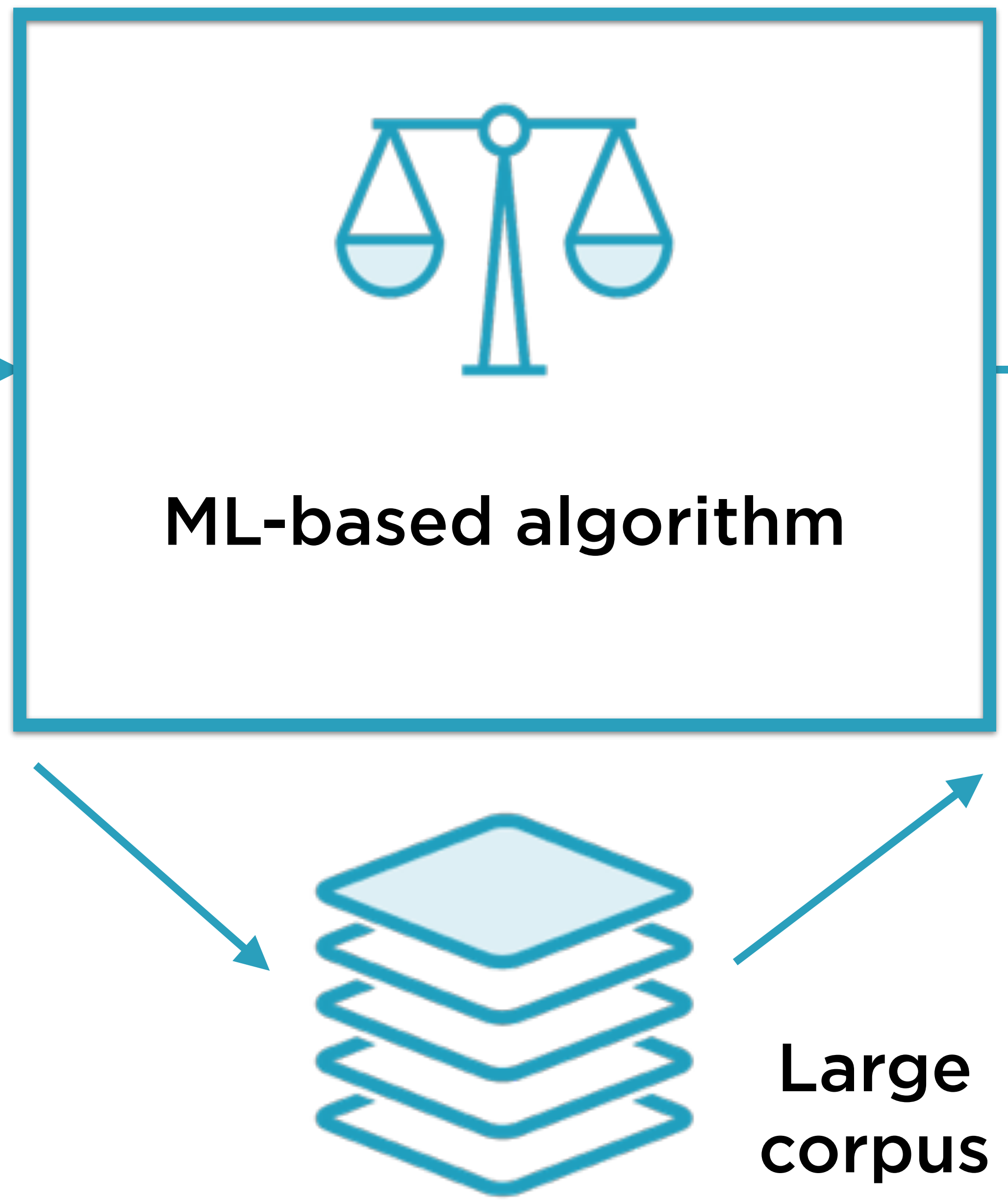


Prediction-based Word Embeddings



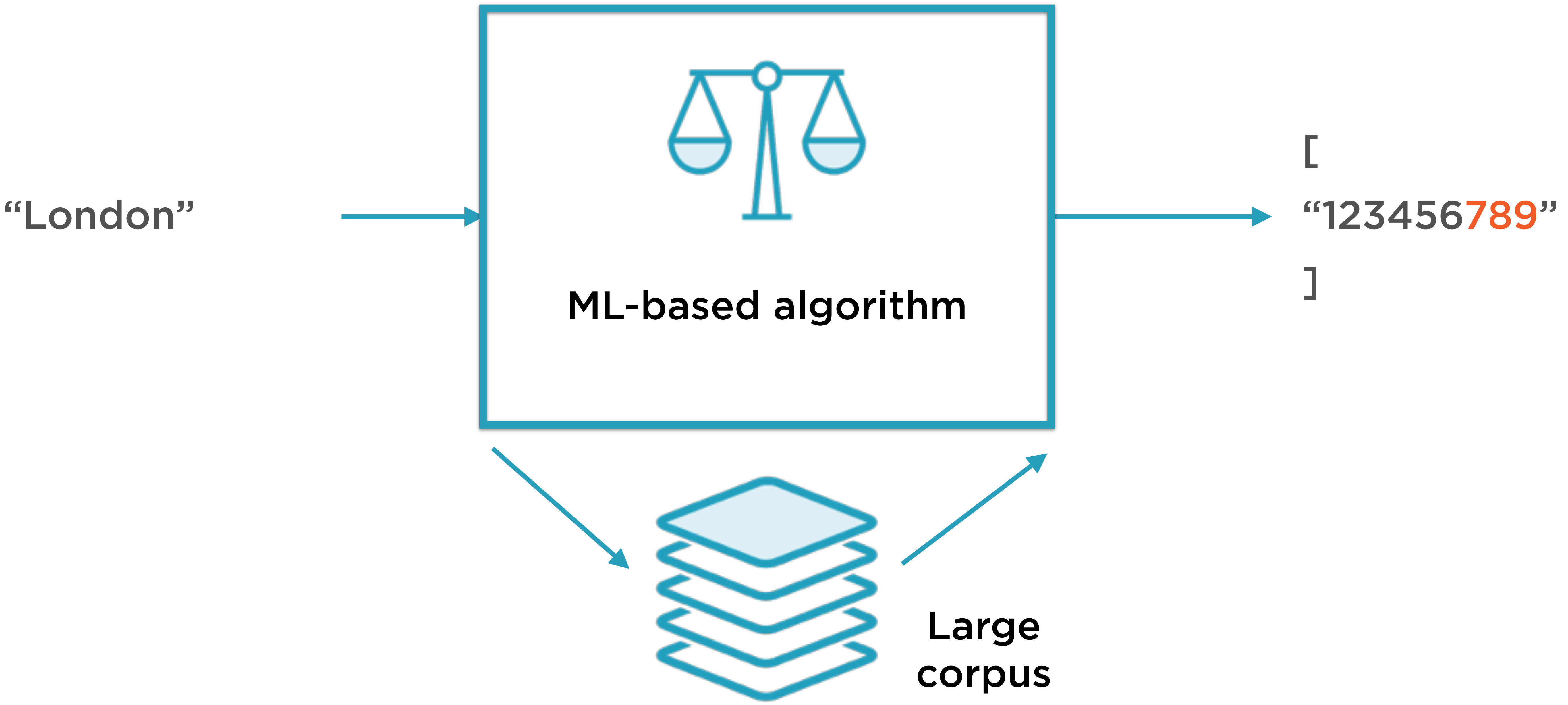
Prediction-based Word Embeddings

[
"New York",
"Paris",
"London"
]

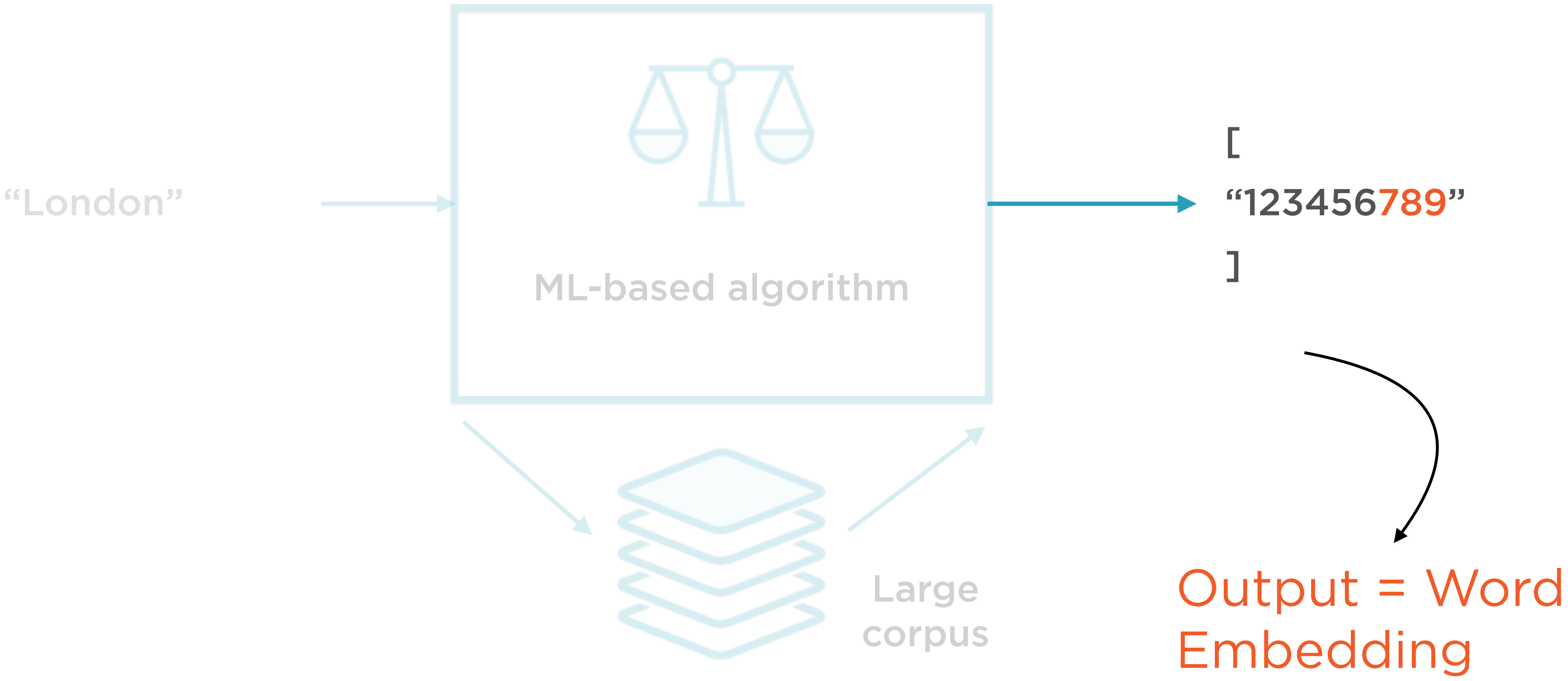


[
"123456770",
"123456600",
"123456789"
]

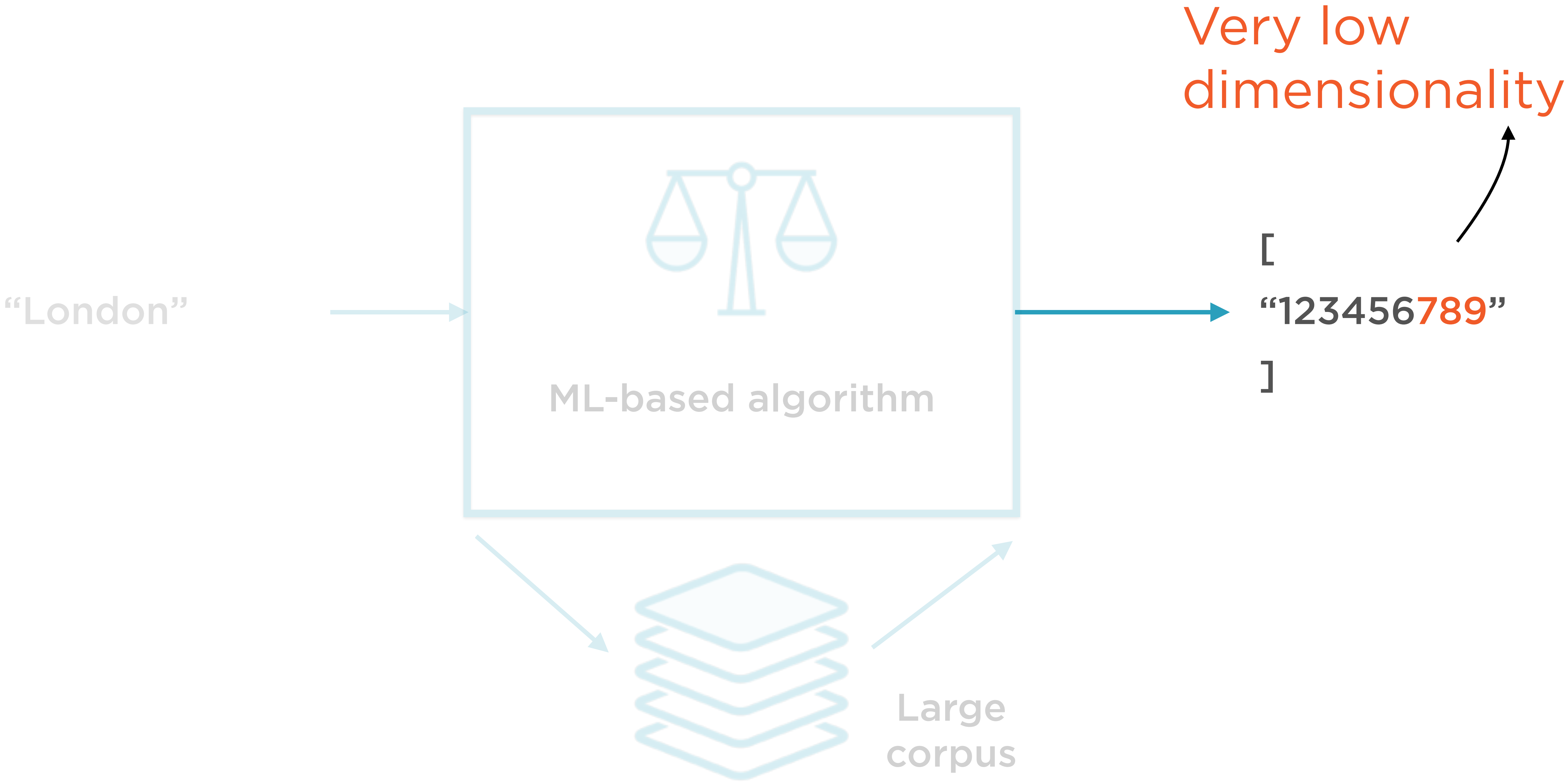
Prediction-based Word Embeddings



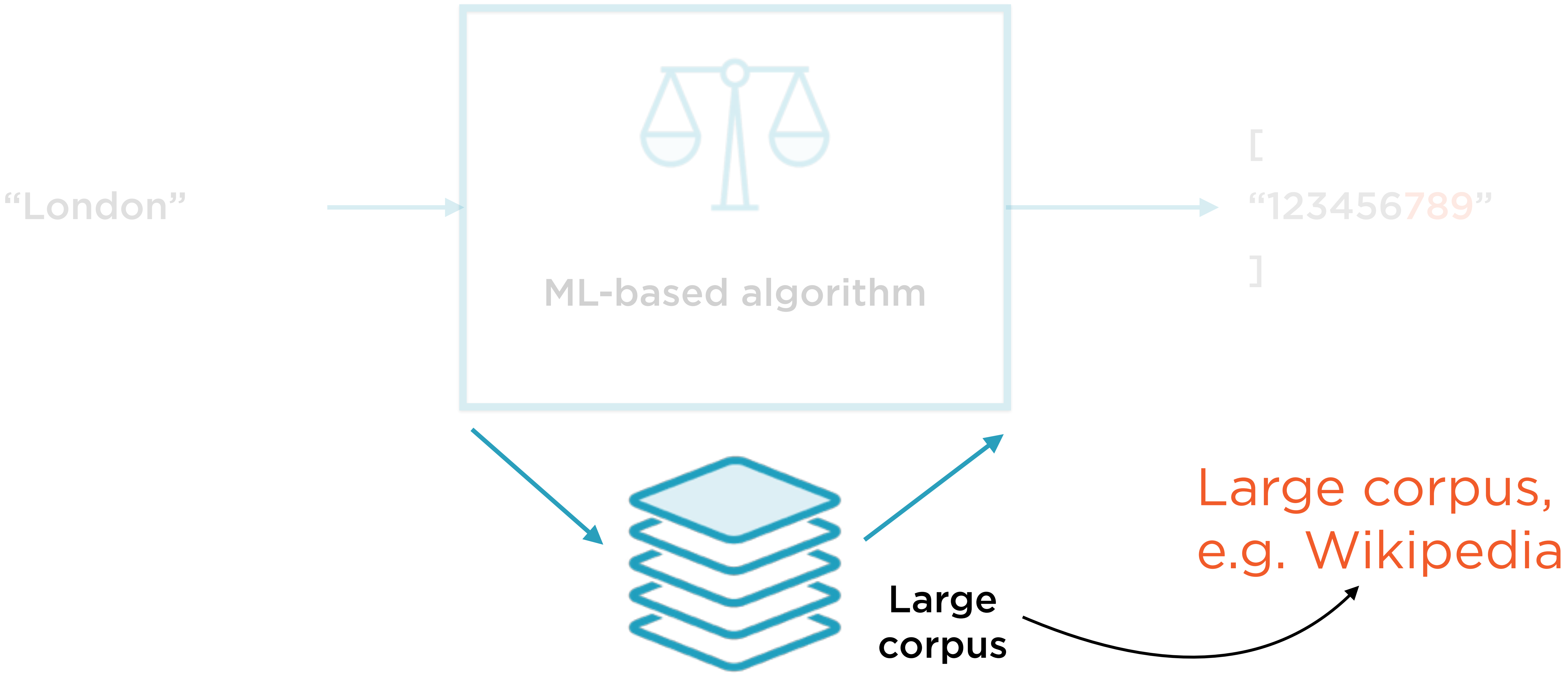
Prediction-based Word Embeddings



Prediction-based Word Embeddings

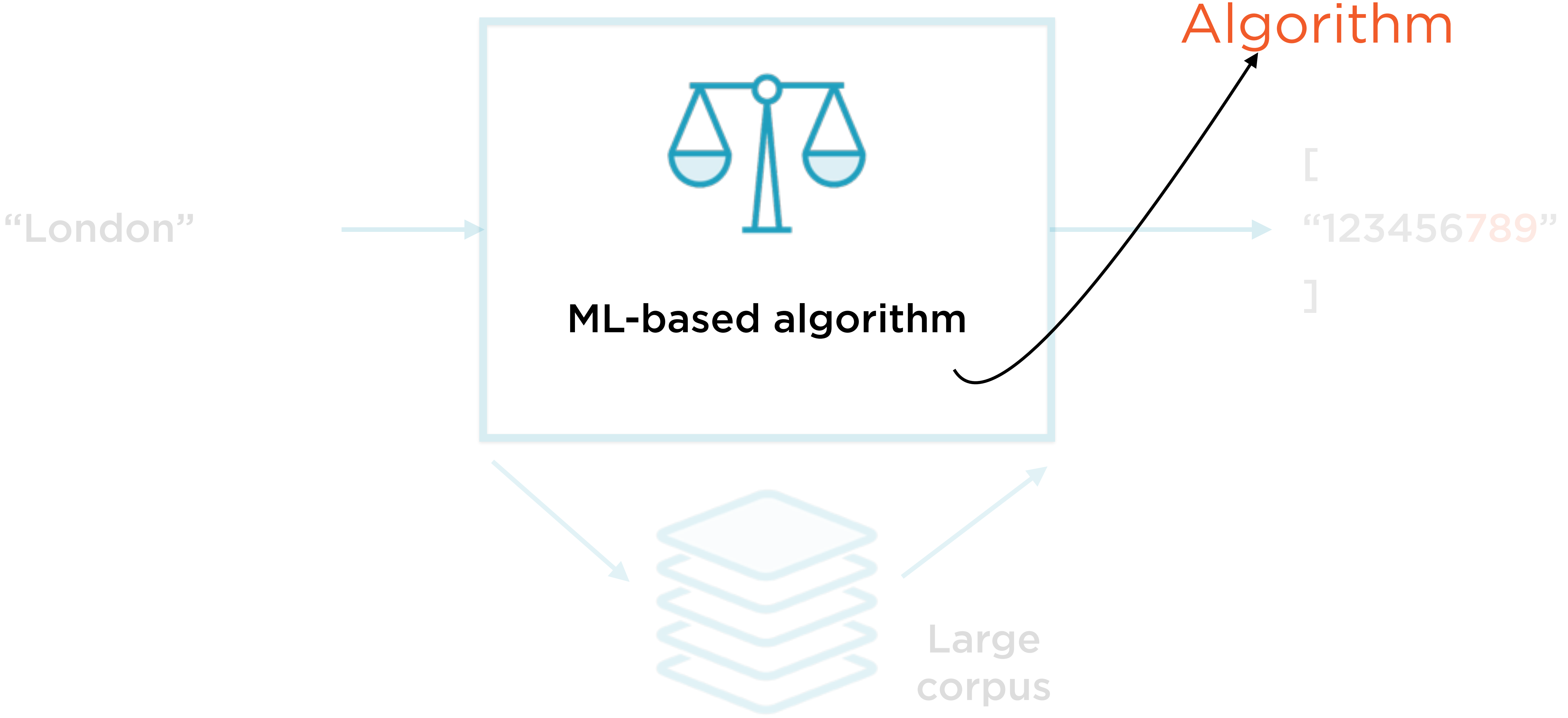


Prediction-based Word Embeddings



Prediction-based Word Embeddings

Unsupervised Learning Algorithm



Magic

Word embeddings capture meaning

“Queen” ~ “King” + “Woman” - “Man”

“Paris” ~ “France” + “London” - “England”

Dramatic dimensionality reduction

Word Embeddings as Unsupervised ML

Learnt using ML, often neural networks

Unsupervised deep learning

Pre-processing step before classification

Embeddings are a way to encode words capturing the **context** around them

Word2Vec

Most popular word embedding model

Mikolov (Google), 2013

Use simple NN (not deep) to learn embeddings

GloVe

Global Vectors for Word Representation

Jeffrey Pennington, Richard Socher, Christopher D. Manning, (Stanford) 2014

Uses word-word co-occurrence matrix, nearest-neighbors for word relationships

Two Familiar Problems

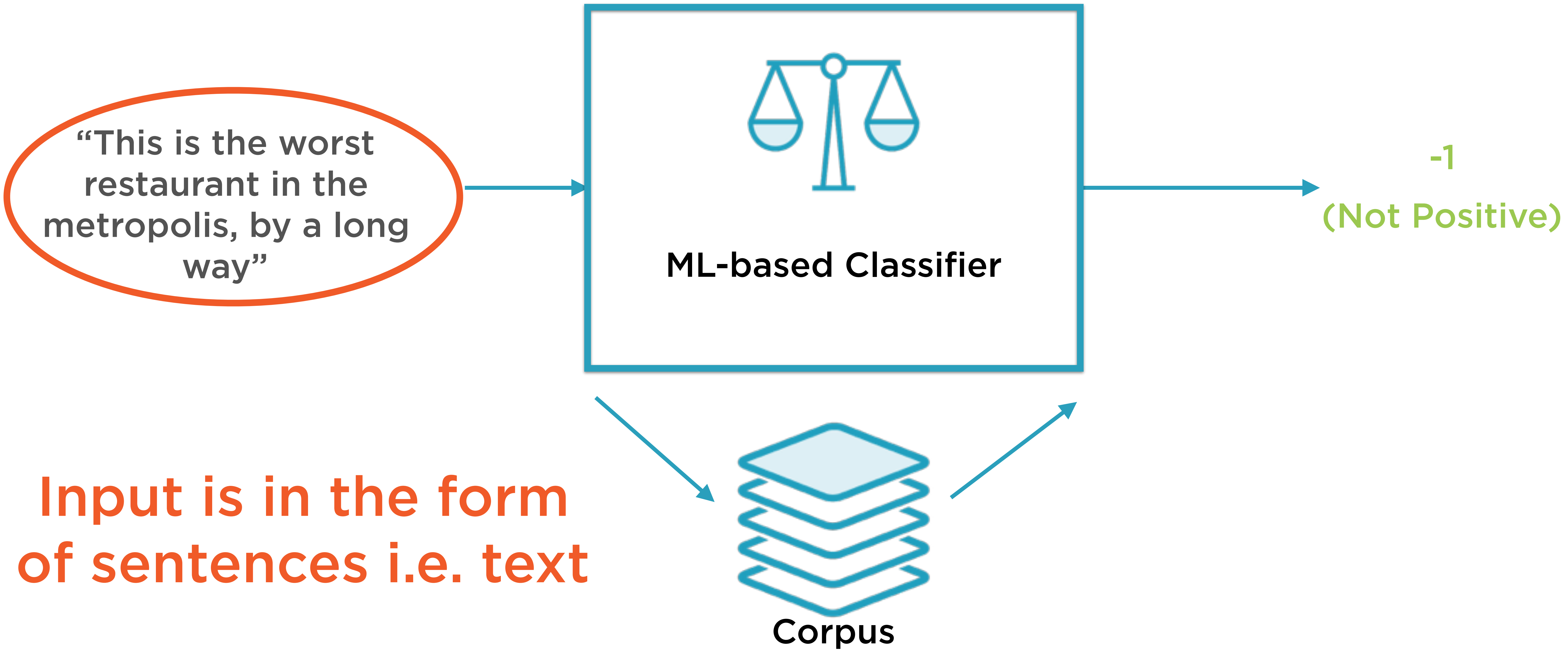
Word Embeddings

Express a word in terms of context in numeric form

Sentiment Analysis

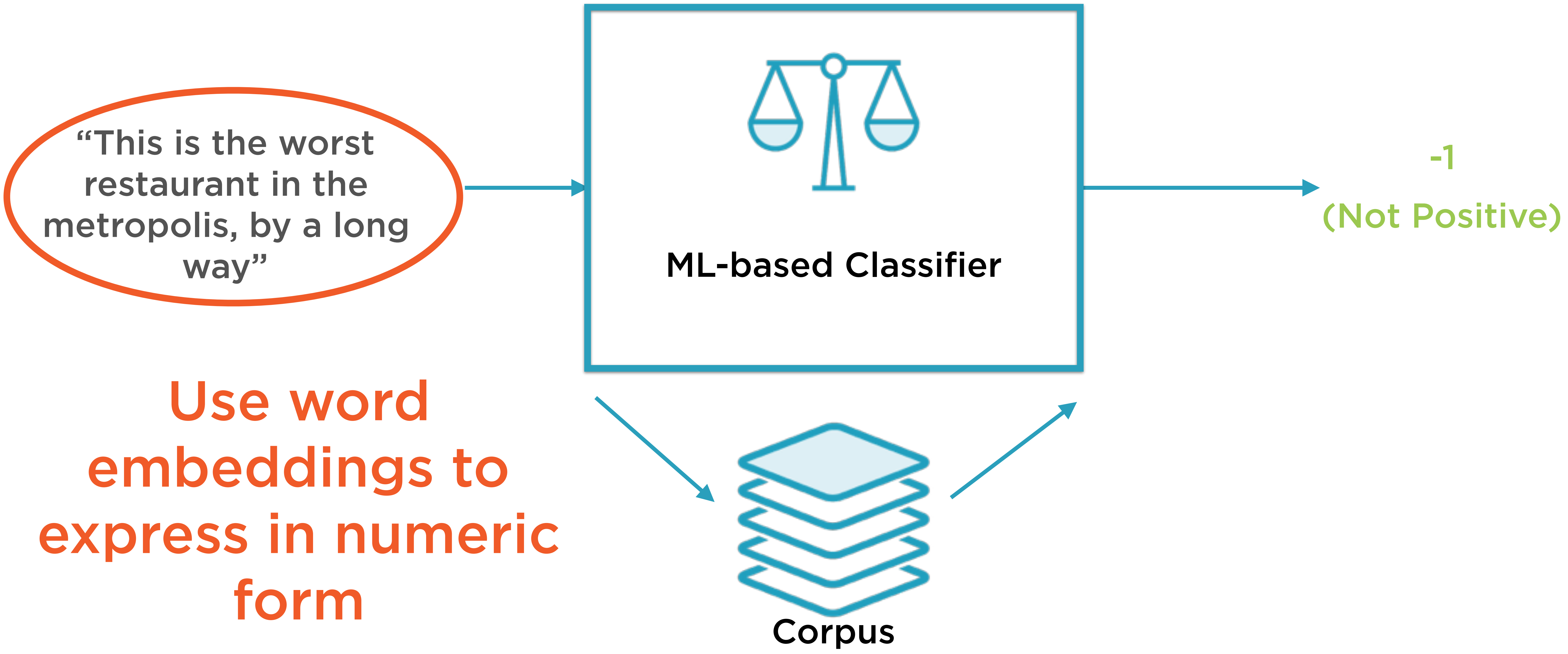
Classify a set of words

Sentiment Analysis Using Neural Networks



Input is in the form of sentences i.e. text

Sentiment Analysis Using Neural Networks



Neural networks are widely used in language modeling

SemEval-2017

Each year SIGLEX publishes tasks as open challenges

- Semantic comparison
- Sentiment, humor and truth
- Parsing semantic structures

Semantic comparison

Semantic Textual Similarity

Given two sentences, return 0-5 score

- 0: Sentences meanings are unrelated
- 5: Sentences have the same meaning

Sub-tasks for different language pairs

- Cross-lingual: Arabic-English, Spanish-English
- Mono-lingual: English-English, Spanish-Spanish

Semantic comparison

Community Question Answering

Input:

- Question
- Large number of user-submitted answers

Output:

- Ranking of user relevance

Additional sub-tasks: Reduce forum clutter

- Question similarity
- Relevance classification

SemEval-2017

Each year SIGLEX publishes tasks as open challenges

- Semantic comparison
- Sentiment, humor and truth
- Parsing semantic structures

Sentiment, humor and truth

Sentiment Analysis

Twitter sentiment analysis

Fine-grained analysis on financial microblogs

Sentiment, humor and truth

Detecting Humor

Given hashtag, find funniest tweet

Humor more subjective than sentiment

Binary classification approaches - simplistic for humor

Inside jokes/references: how to incorporate external knowledge?

Sentiment, humor and truth

Puns

Word sense disambiguation (WSD)

Homophonic (perfect) puns

“I used to be a banker but I lost interest”

Heterophonic (imperfect) puns

“With fronds like these, who needs anemones?”

SemEval-2017

Each year SIGLEX publishes tasks as open challenges

- Semantic comparison
- Sentiment, humor and truth
- Parsing semantic structures

Parsing semantic structures

Extracting Keyphrases

Given academic publication, extract key phrases and relationships

Identifying relationships is a classic language modeling task

Cause-effect Identification: Given a sentence, tag cause and effect

Parsing semantic structures

Types of Relationships

Cause-effect

"The *cancer* was caused by *radiation*"

Instrument-agency

"The *catcher* used a *mitt*"

Product-producer

"The *craftsman* built fine *watches*"

Content-container

"Old *wine* in new *bottles*"

Entity-origin

"*Friends* from *faraway places*"

Types of Cells Used in RNNs

Types of Neurons

Simple Neuron

Affine transformation, activation function

Recurrent Neuron

Feed output back as another input

LSTM Cell

Maintain complex additional state for long-memory

GRU Cell

Similar results as LSTM, but simpler internals

Multi-RNN Cell

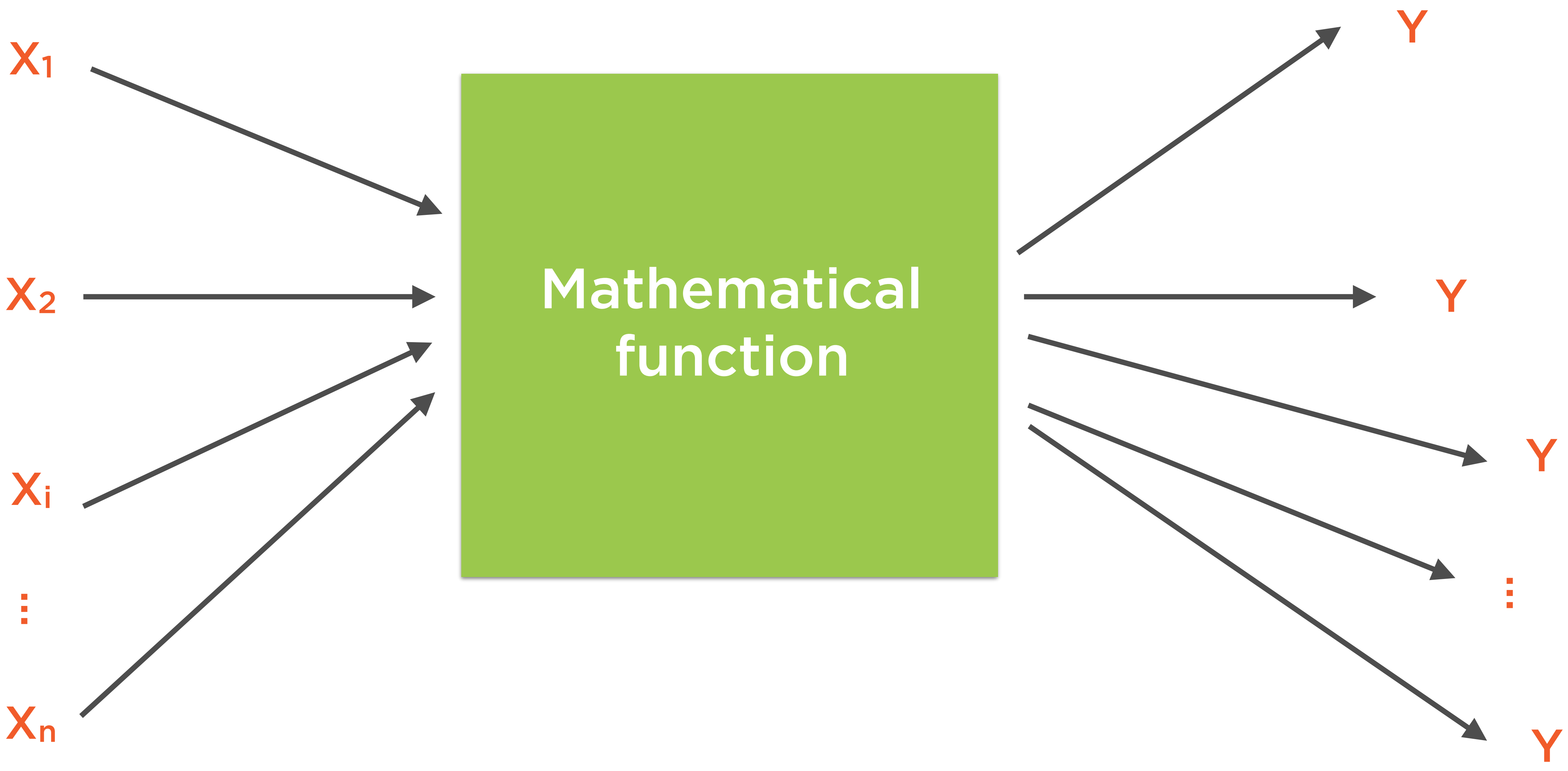
Wrap multiple GRU cells into single multi-layer cell

Types of Neurons

Simple Neuron

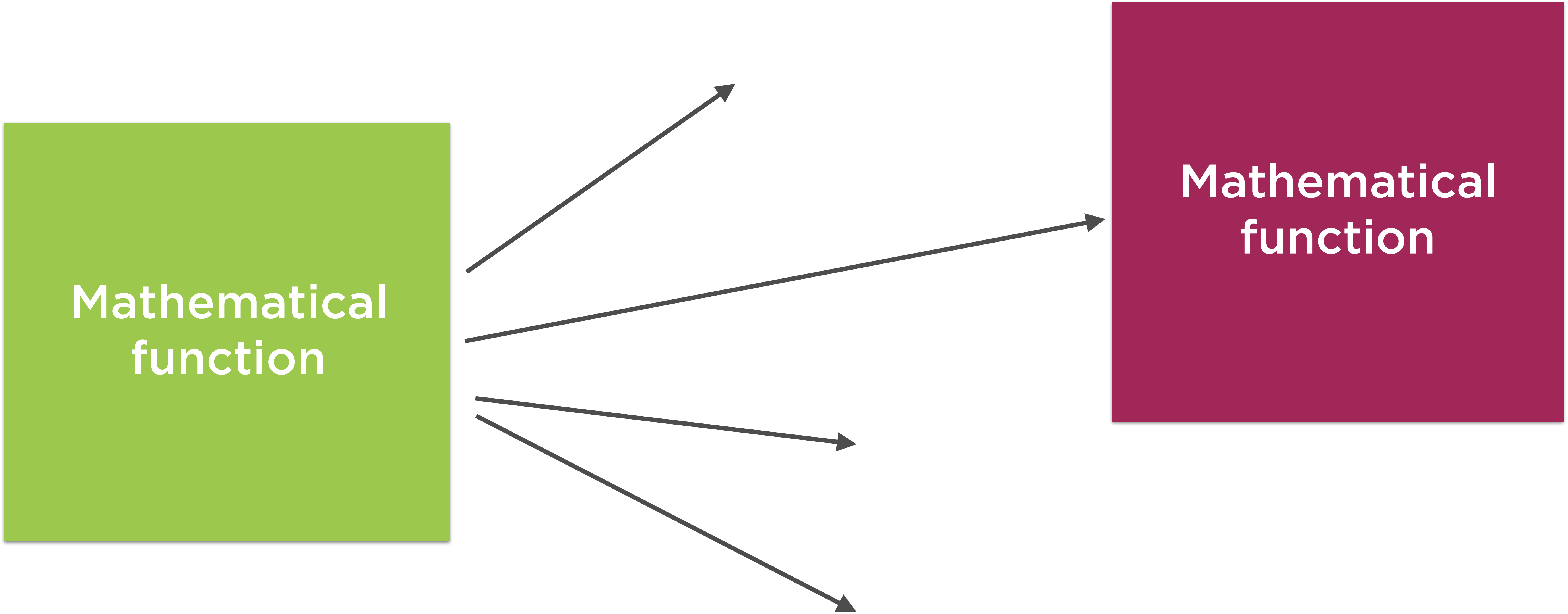
Affine transformation, activation function

Operation of a Single Neuron



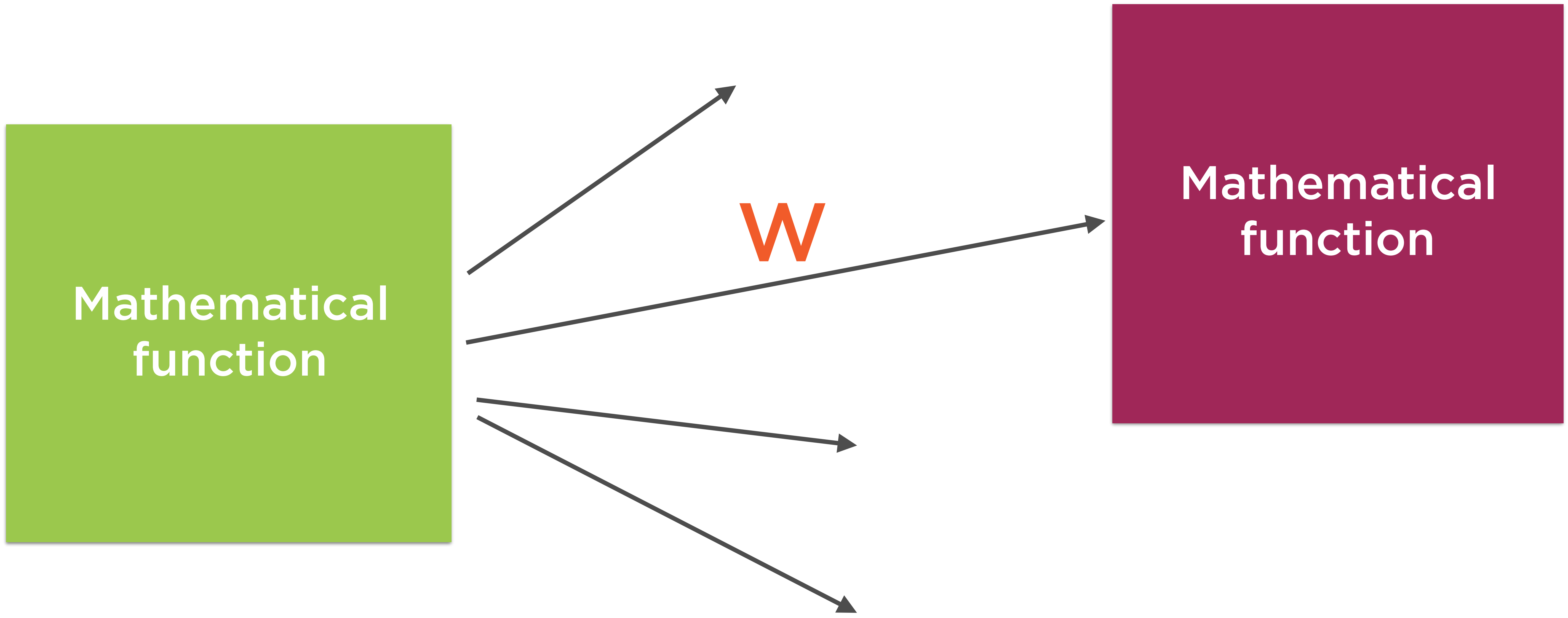
For an active neuron a change in inputs should trigger a corresponding change in the outputs

Operation of a Single Neuron



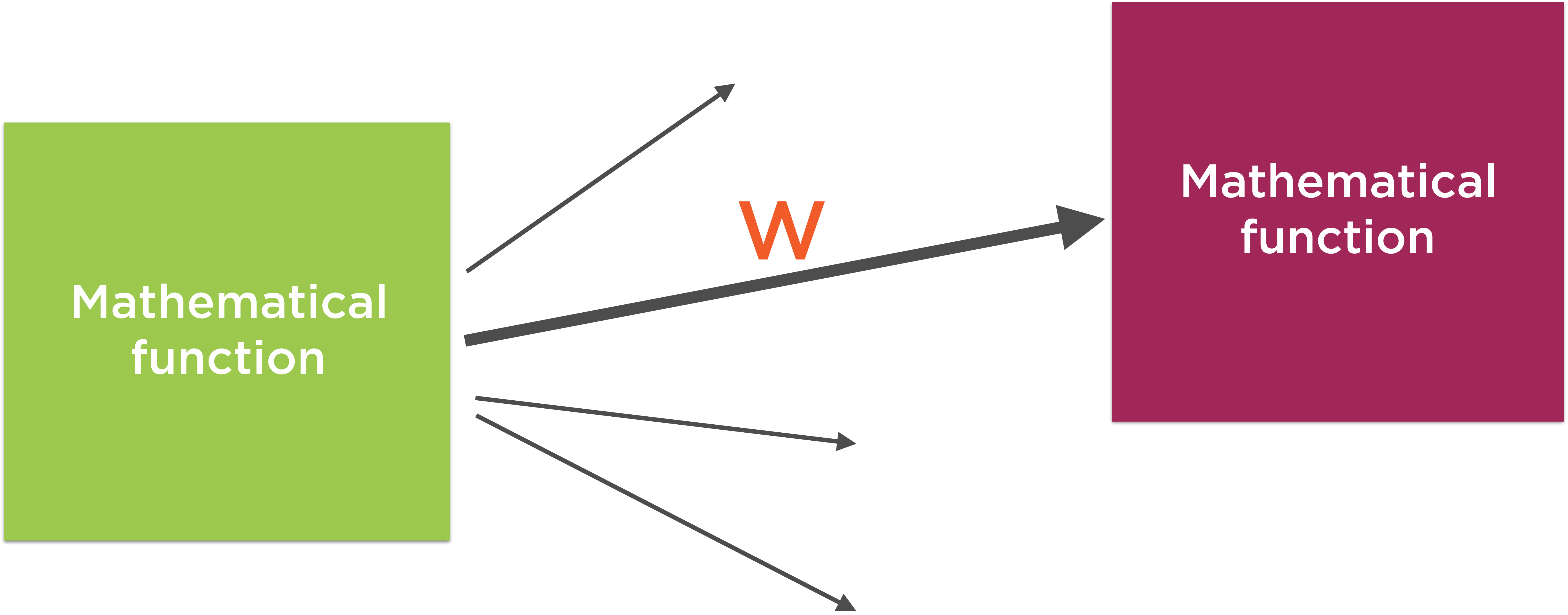
The outputs of neurons feed into the neurons from the next layer

Operation of a Single Neuron



Each connection is associated with a weight

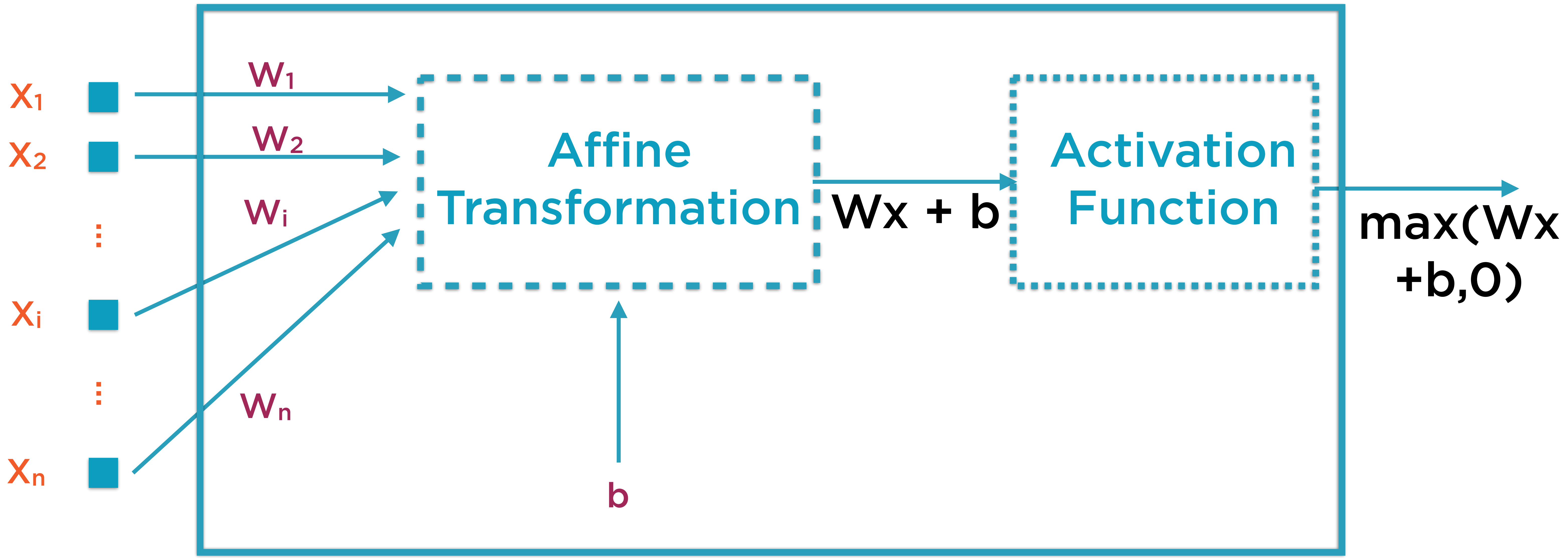
Operation of a Single Neuron



If the second neuron is sensitive to the output of the first neuron, the connection between them gets stronger

W increases

Neuron as a Learning Unit



The combination of the affine transformation and the activation function allows the neuron to learn any arbitrary relationship

Types of Neurons

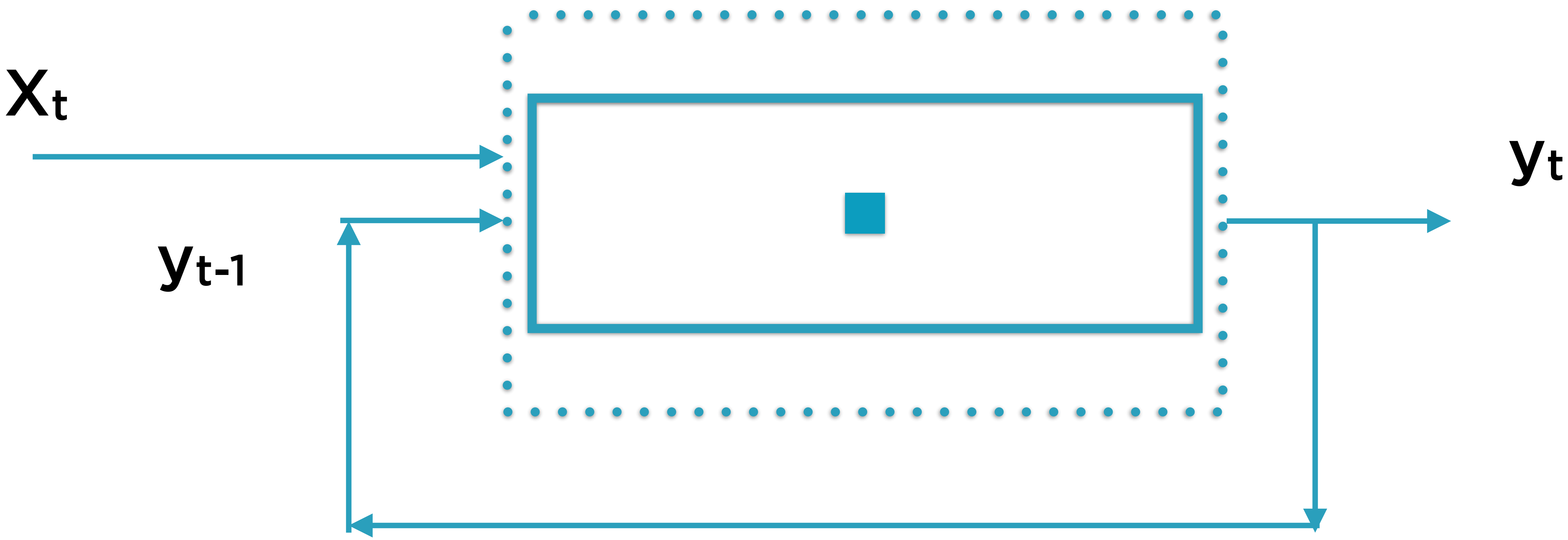
Simple Neuron

Affine transformation, activation function

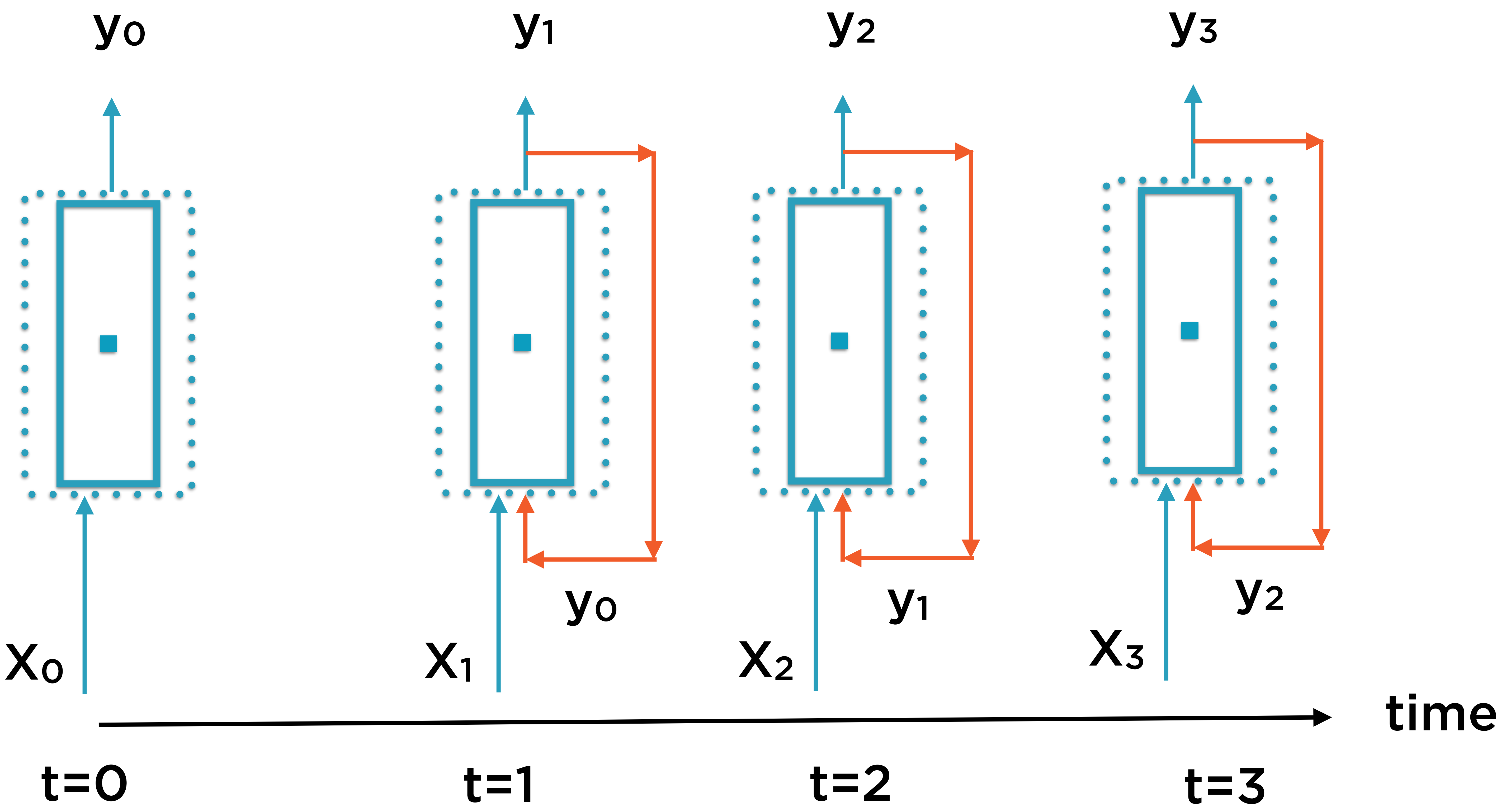
Recurrent Neuron

Feed output back as another input

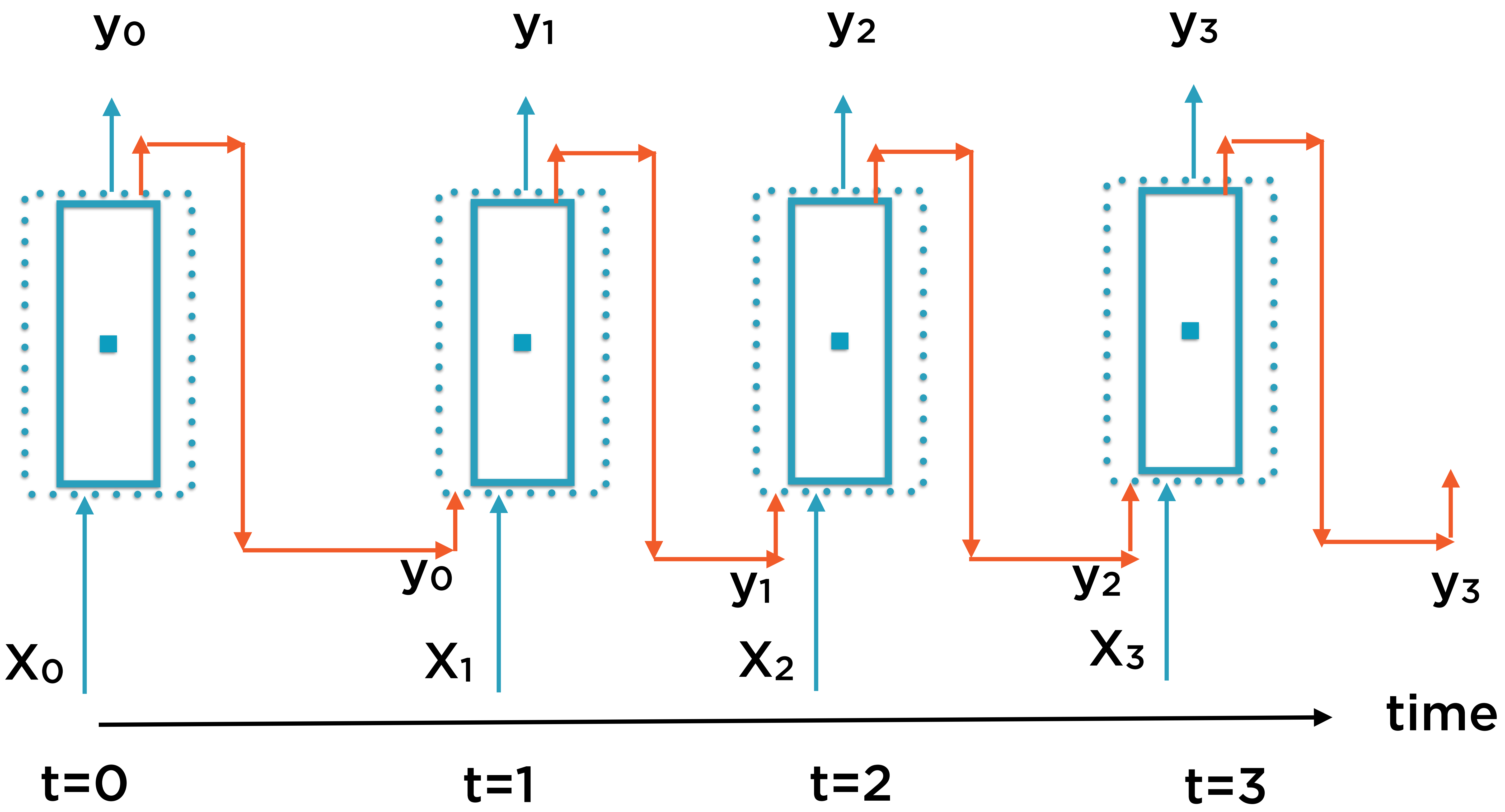
Simplest Recurrent Neuron



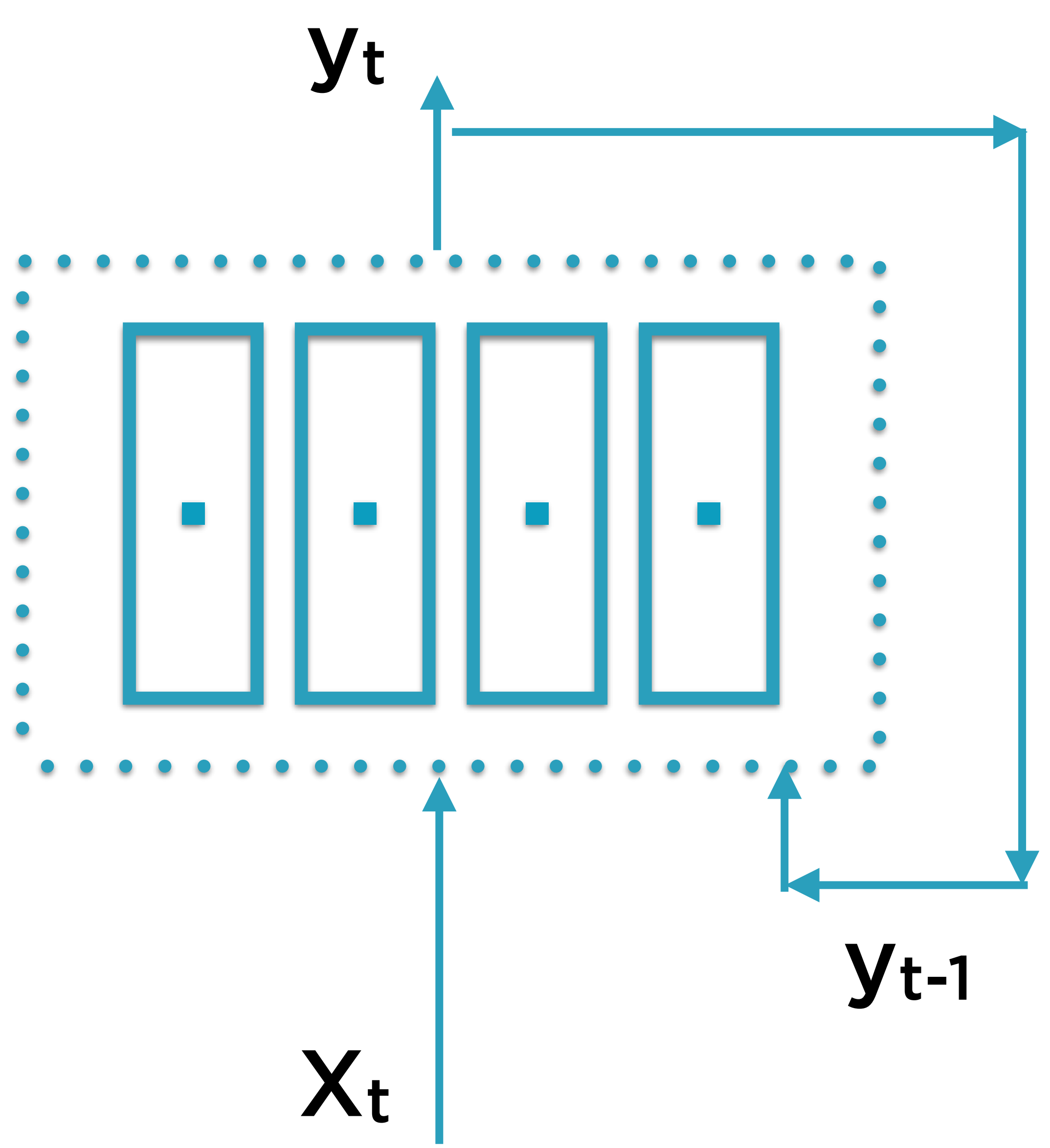
Unrolling Through Time



Unrolling Through Time

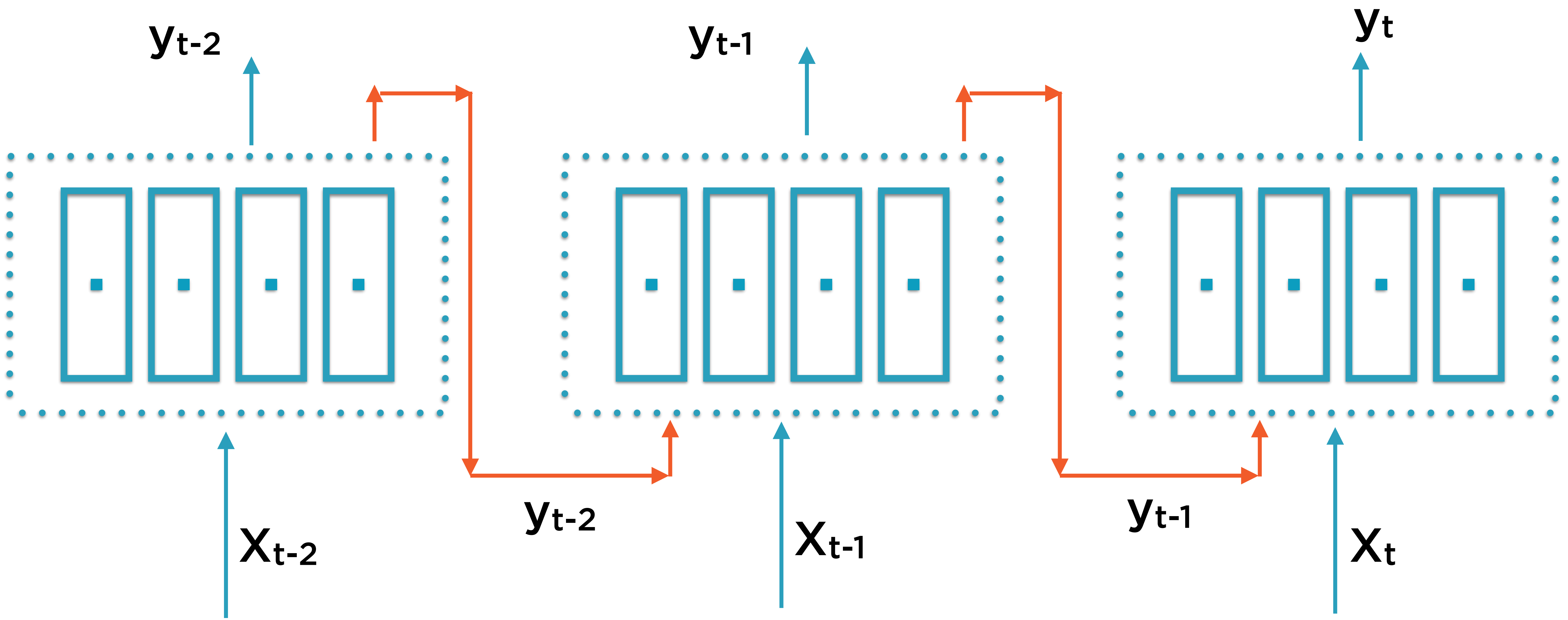


Layer of Recurrent Neurons



A layer of neurons forms an RNN cell

Layer of Recurrent Neurons



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

Types of Neurons

Simple Neuron

Affine transformation, activation function

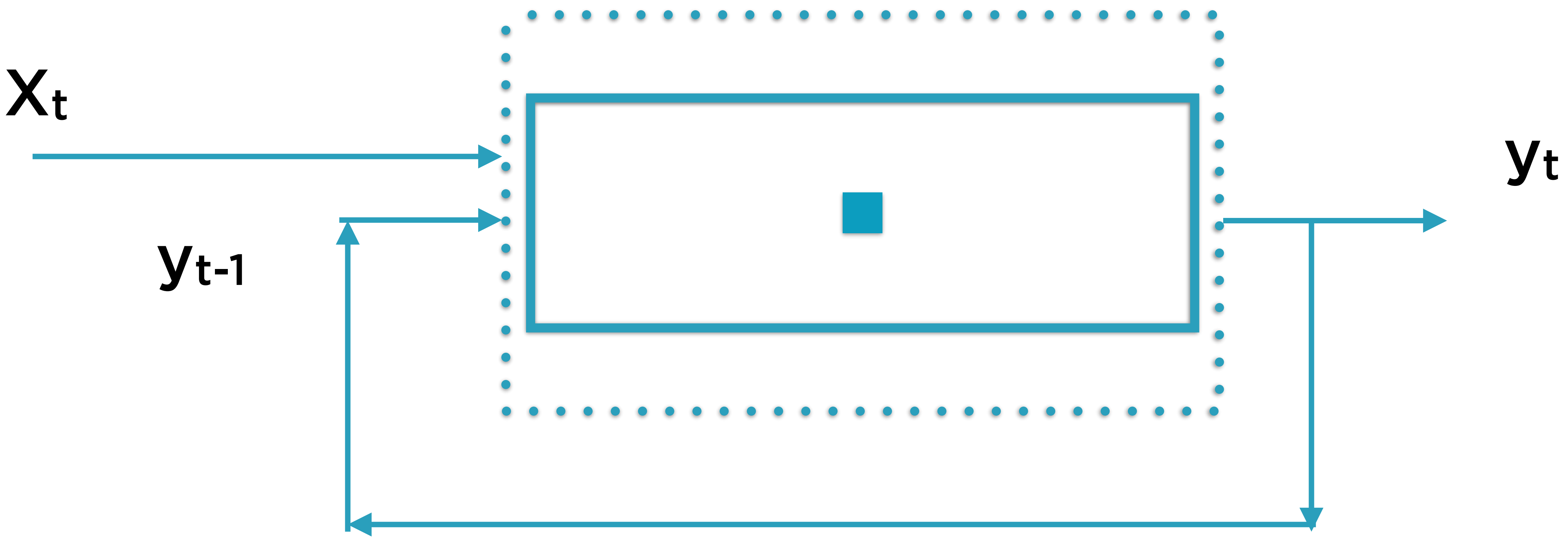
Recurrent Neuron

Feed output back as another input

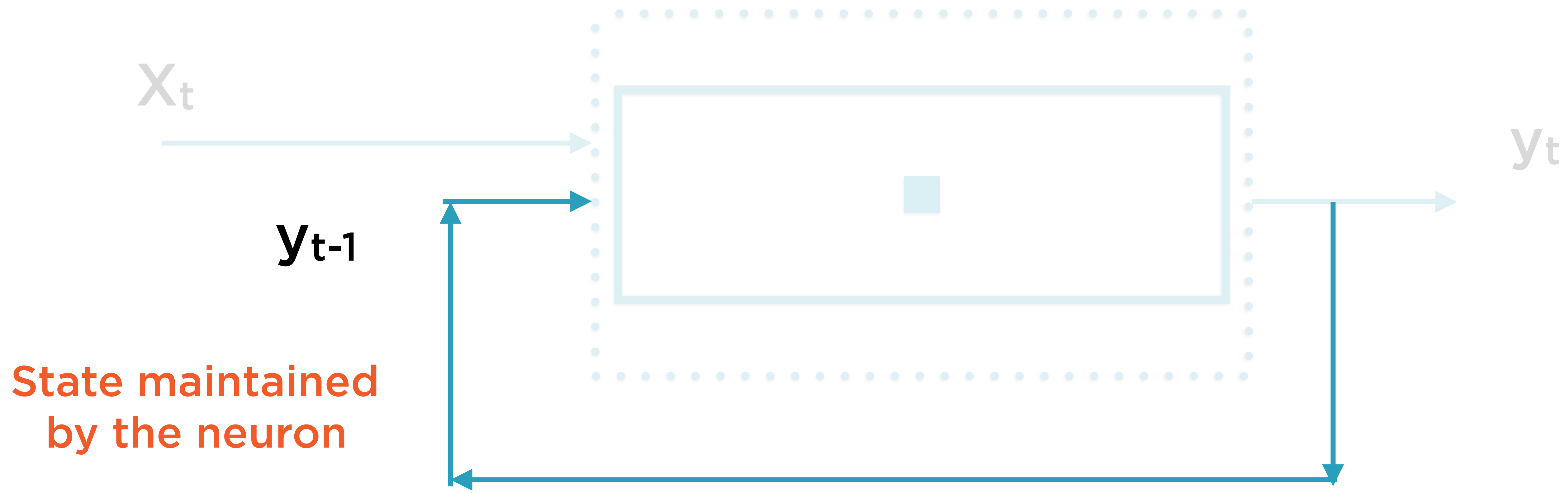
LSTM Cell

Maintain complex additional state for long-memory

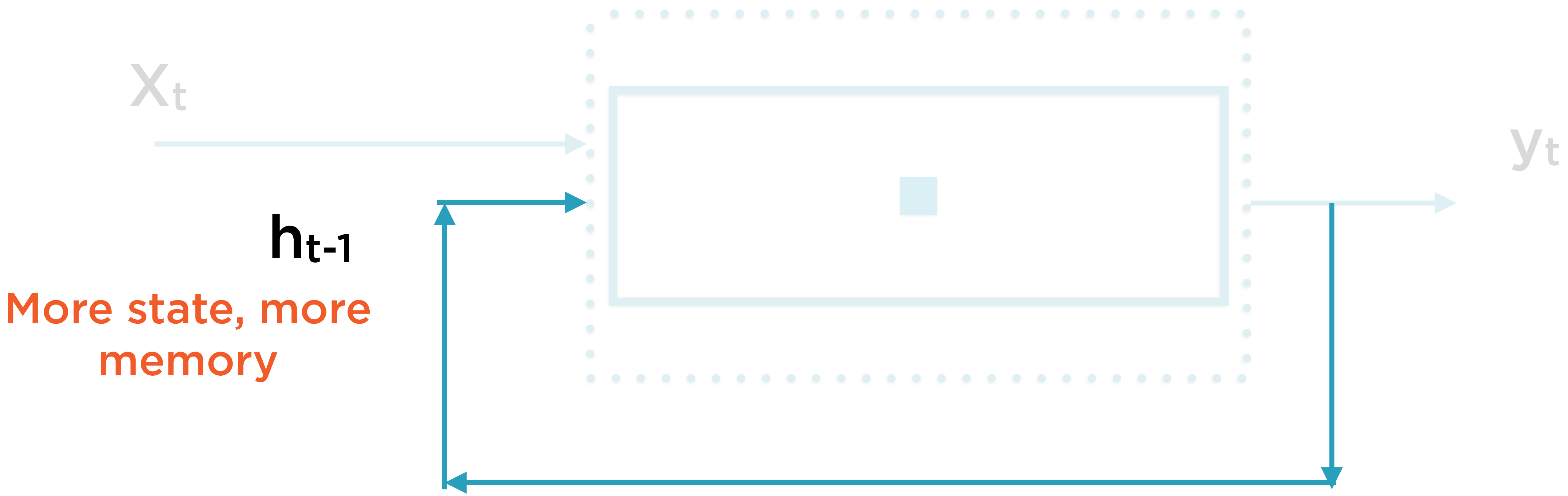
Simplest Recurrent Neuron

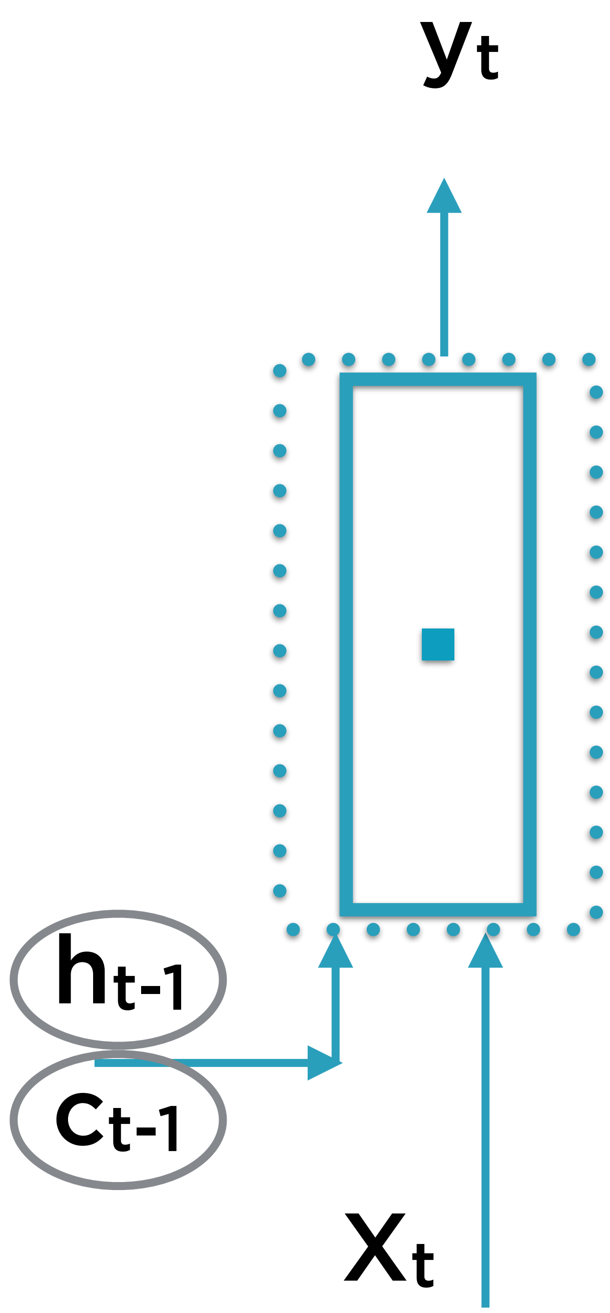


Simplest Recurrent Neuron



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 neurons have several advantages over basic RNNs

Long Memory RNNs

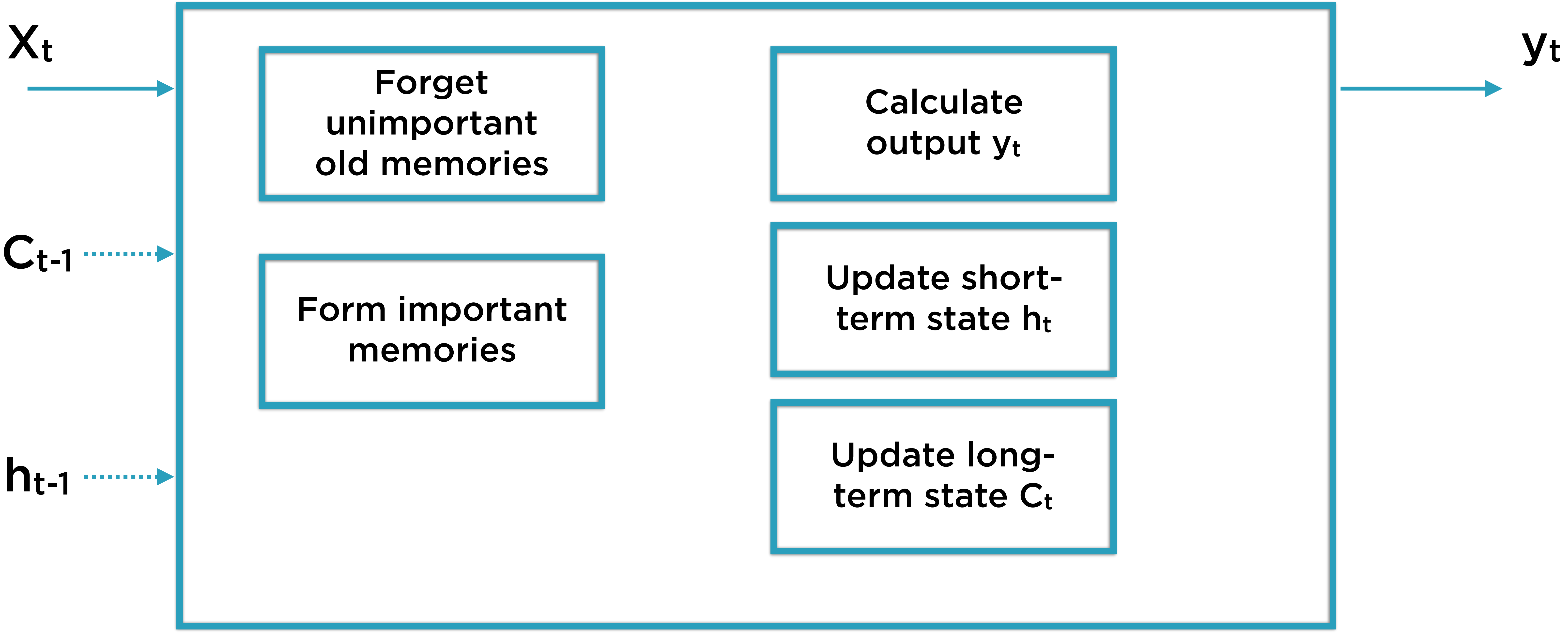
Advantages in Training

Faster training, nicer gradients

Advantages in Prediction

No need to truncate BPTT

Long/Short-Term Memory Cell (LSTM)



Types of Neurons

Simple Neuron

Affine transformation, activation function

Recurrent Neuron

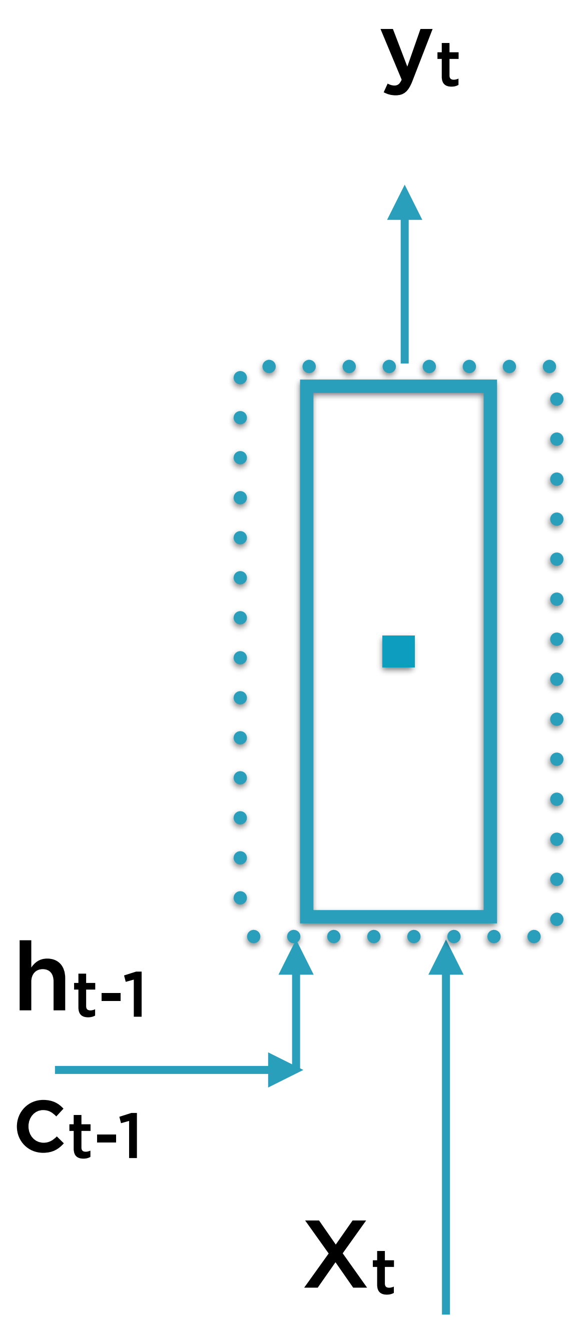
Feed output back as another input

LSTM Cell

Maintain complex additional state for long-memory

GRU Cell

Similar results as LSTM, but simpler internals



GRU

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

Multi-RNN Cell

Types of Neurons

Simple Neuron

Affine transformation, activation function

Recurrent Neuron

Feed output back as another input

LSTM Cell

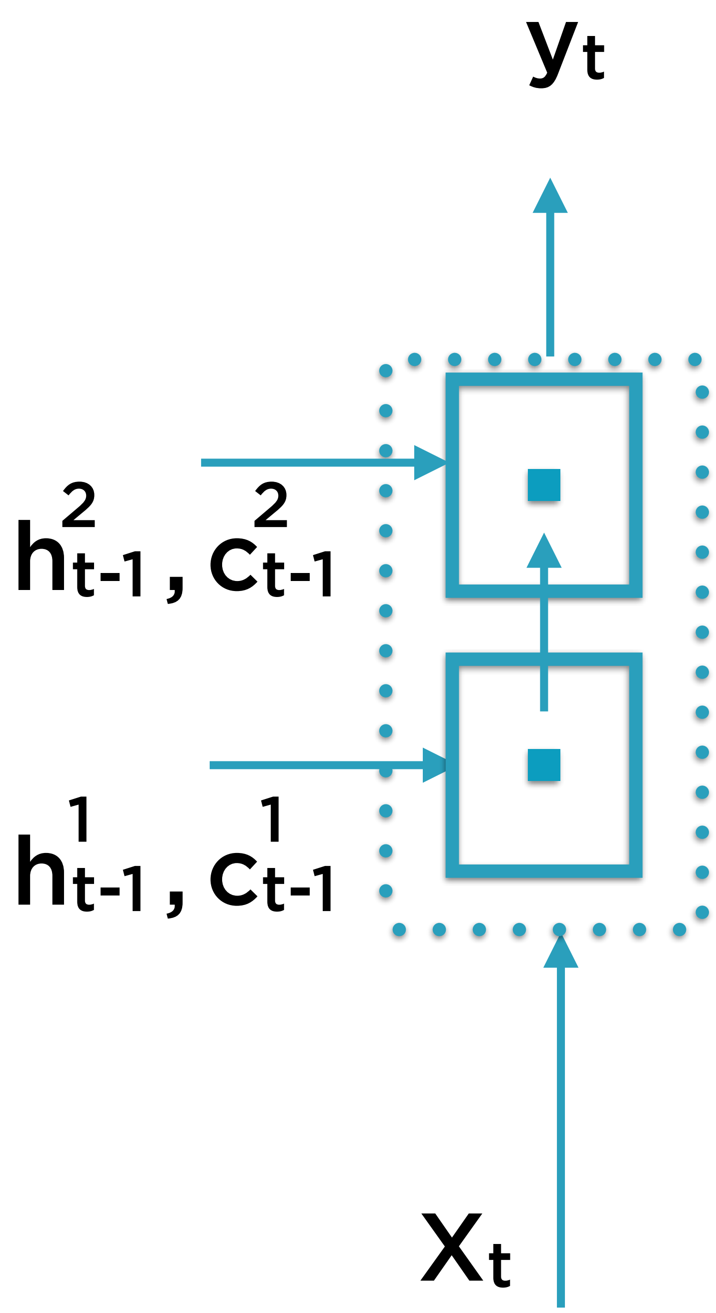
Maintain complex additional state for long-memory

GRU Cell

Similar results as LSTM, but simpler internals

Multi-RNN Cell

Wrap multiple GRU cells into single 2-layer cell

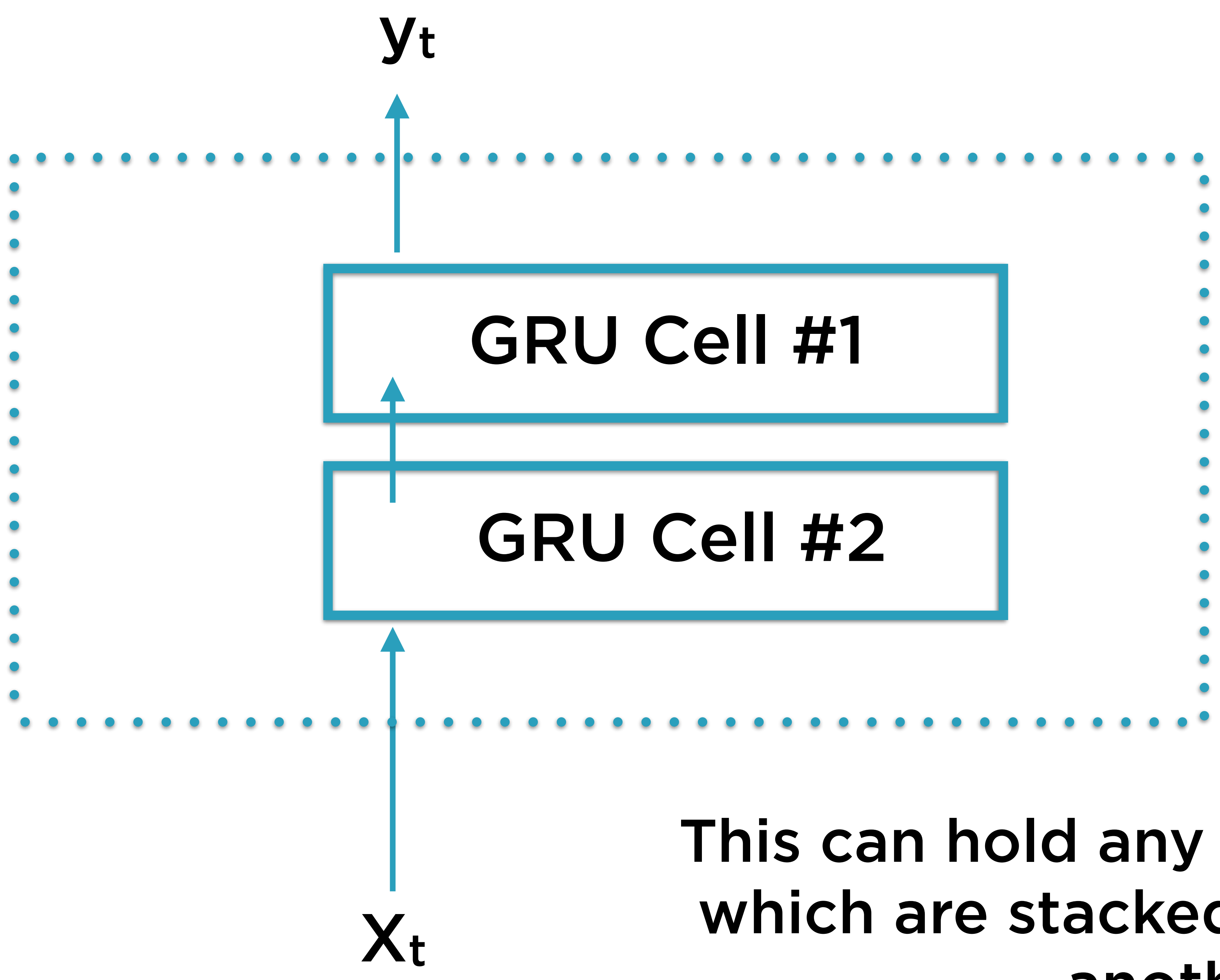


Multi-RNN Cell

Stack multiple RNN cells into “combined” RNN cell

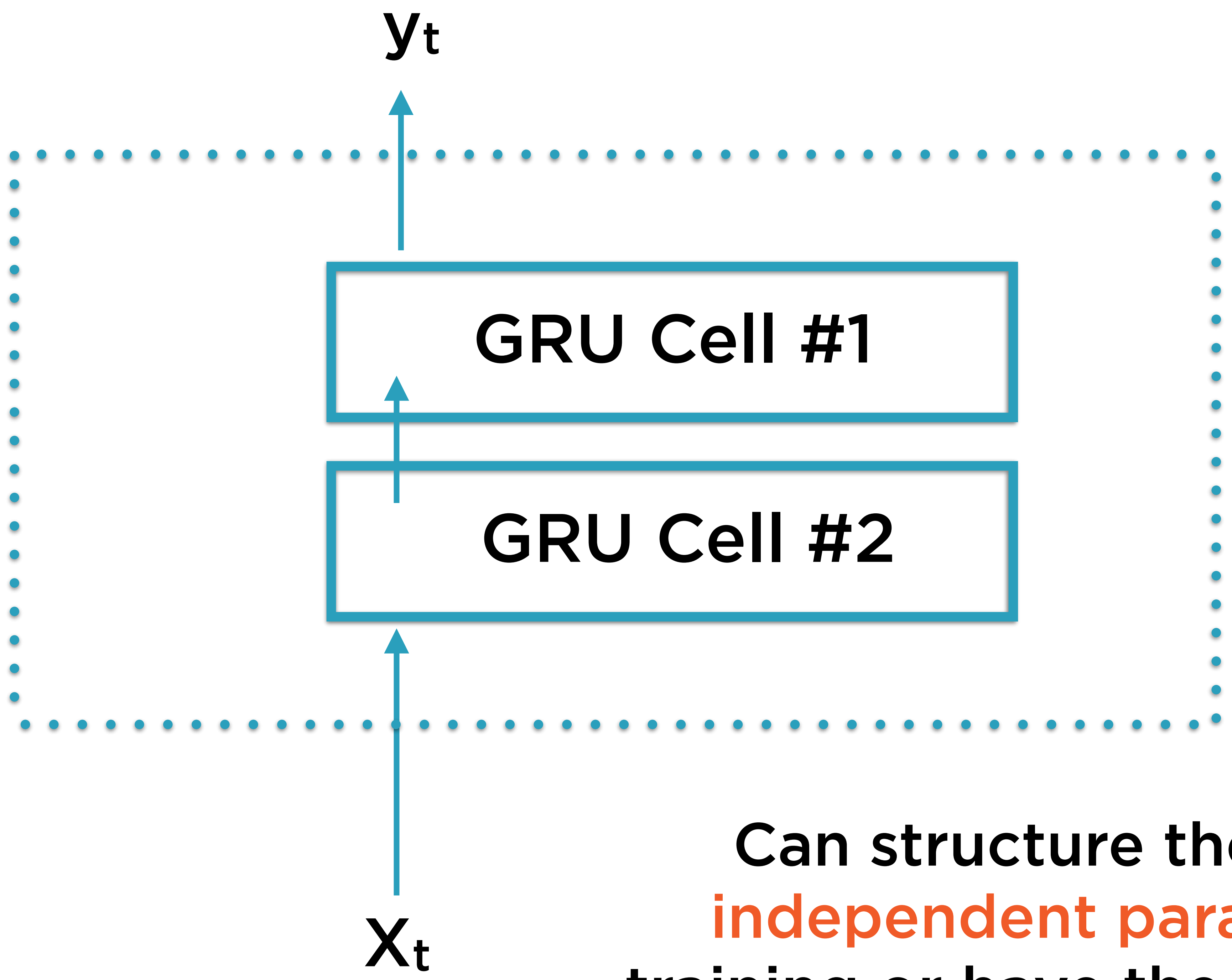
In our example, use 2 GRU cells inside each multi-RNN cell

Multi-RNN Cell



This can hold any number of cells which are stacked one on top of another

Multi-RNN Cell



Can structure the cells to have independent parameters during training or have the same parameters

Multi-RNN cells allow you to wrap multiple cells allowing them to **look and behave like a single cell**

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

Learning the (Recent) Past

Unrolling the RNN through time helps learn the past

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

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

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

Learning the Distant Past

Using LSTM and GRU cells helps maintain memory of the distant past with internal state rather than large number of layers

Key insight: Smart re-use of prior period state is key to prediction

$$y_t,\ Prev\_Internal\_State_t = f(x_t, y_{t-1}, Prev\_Internal\_State_{t-1})$$

Alternative Approach to the Distant Past

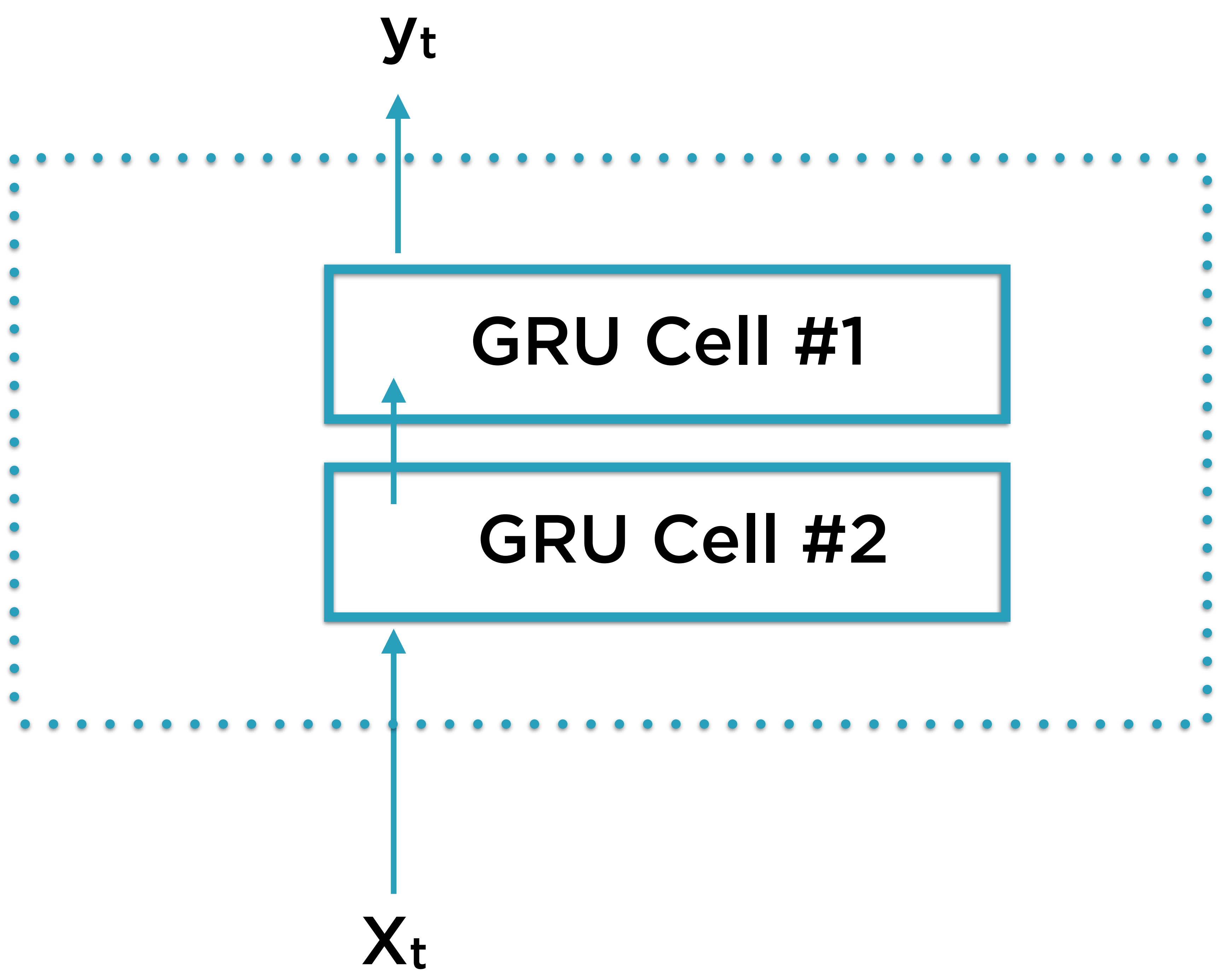
Re-using internal state in addition to using GRU gives great performance with less input

y_t, **$\text{Prev\_Internal\_State}_t$** $= f(x_t, y_{t-1},$ **$\text{Prev\_Internal\_State}_{t-1}$**$)$

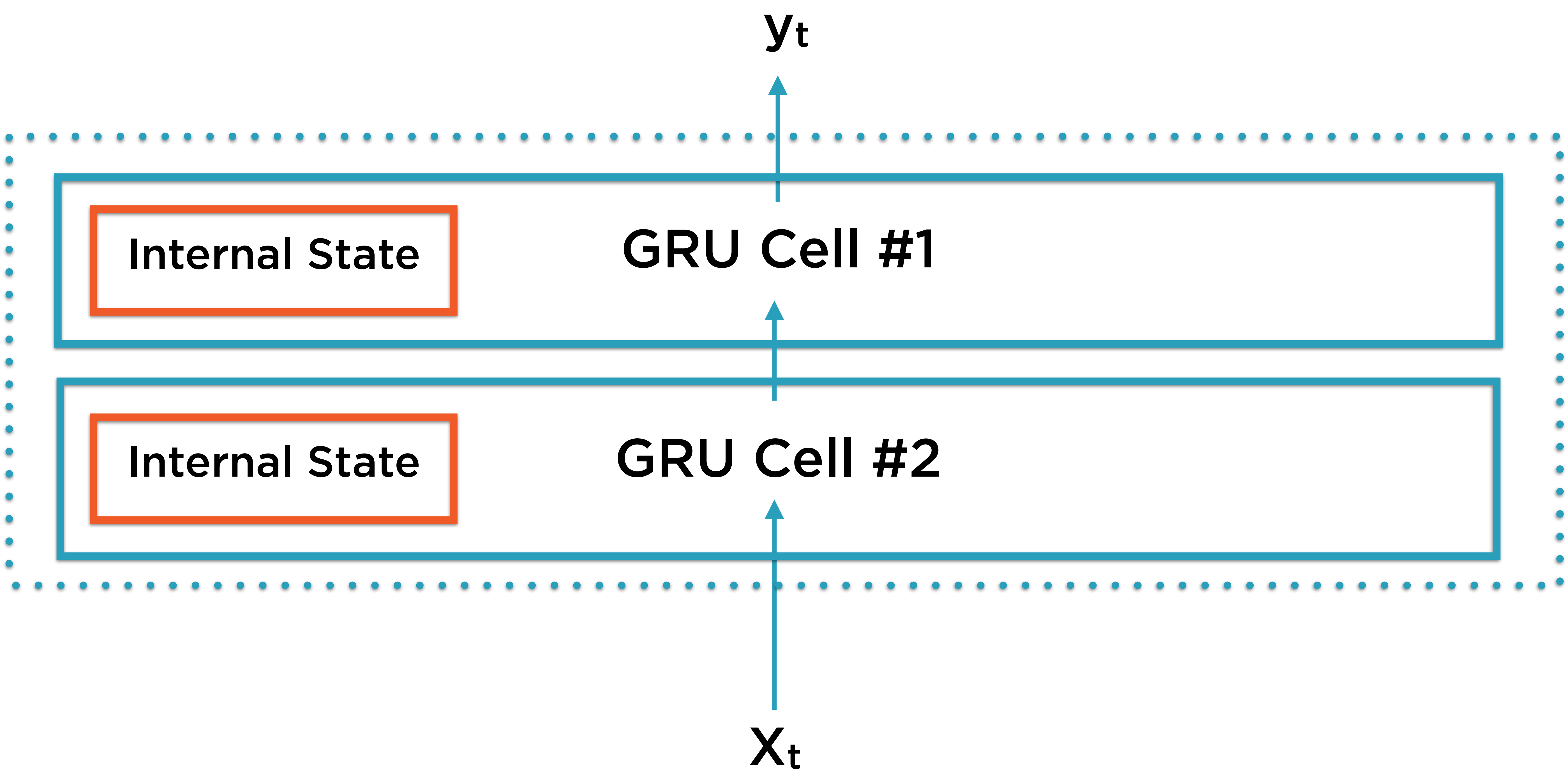
Feed the Previous State as Initial Values

Re-using internal state in addition to using GRU gives great performance with less input

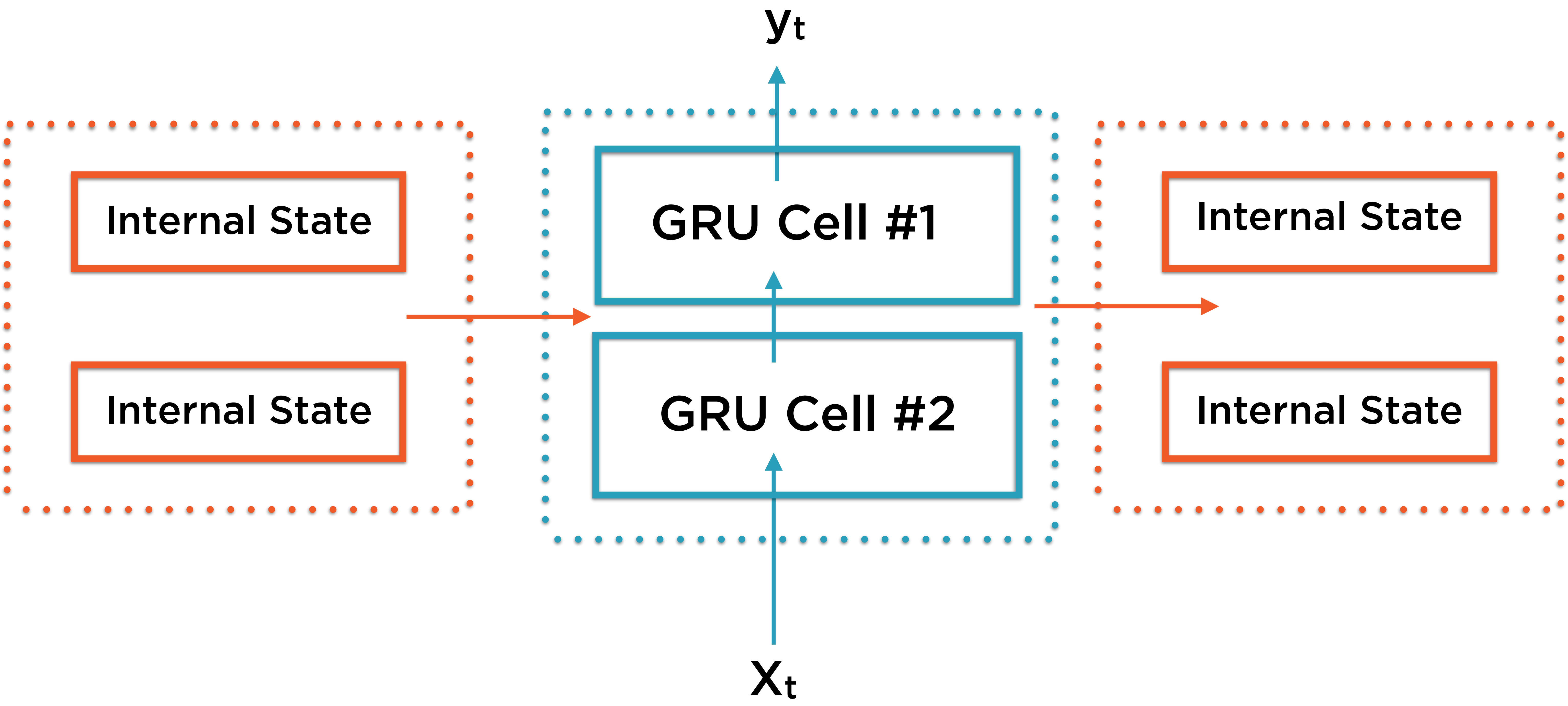
Re-using Internal State



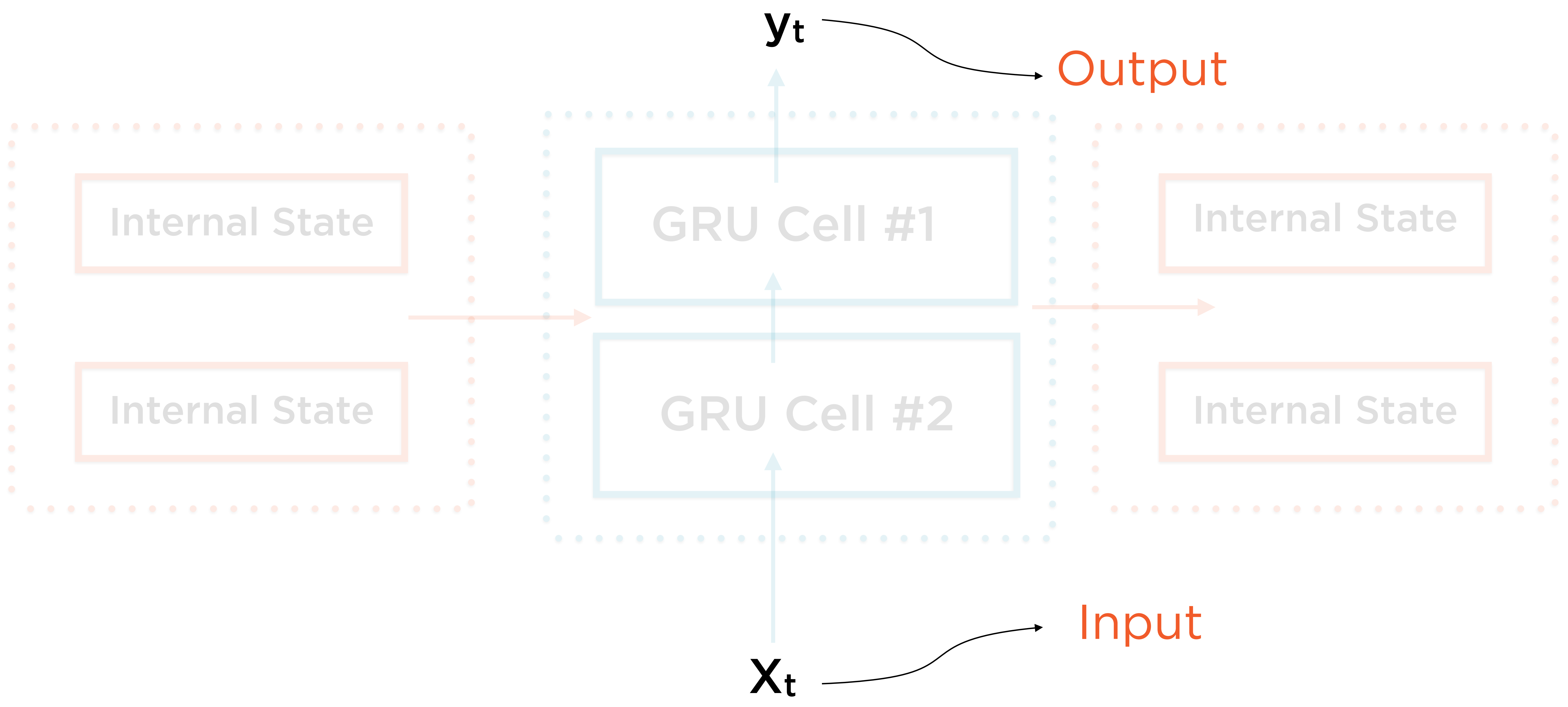
Re-using Internal State



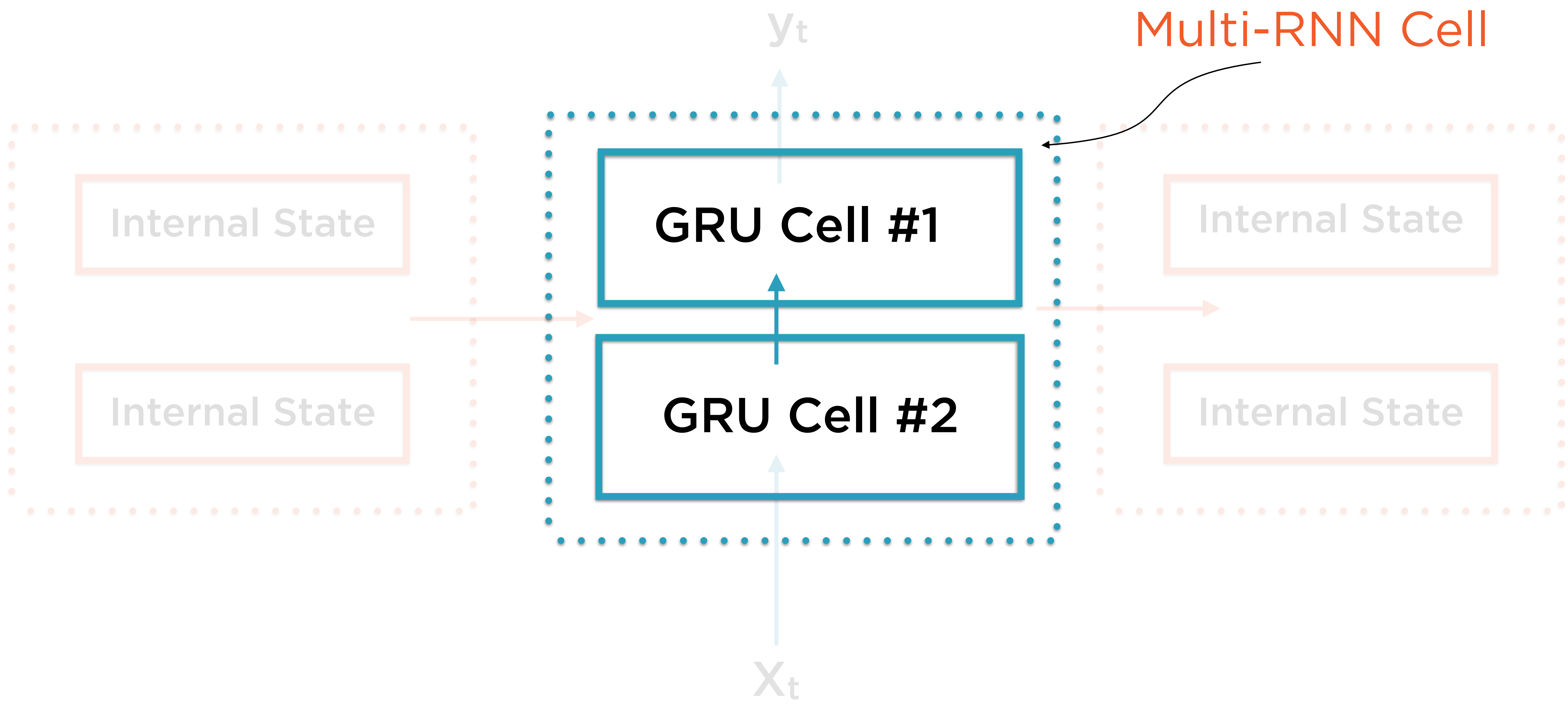
Re-using Internal State



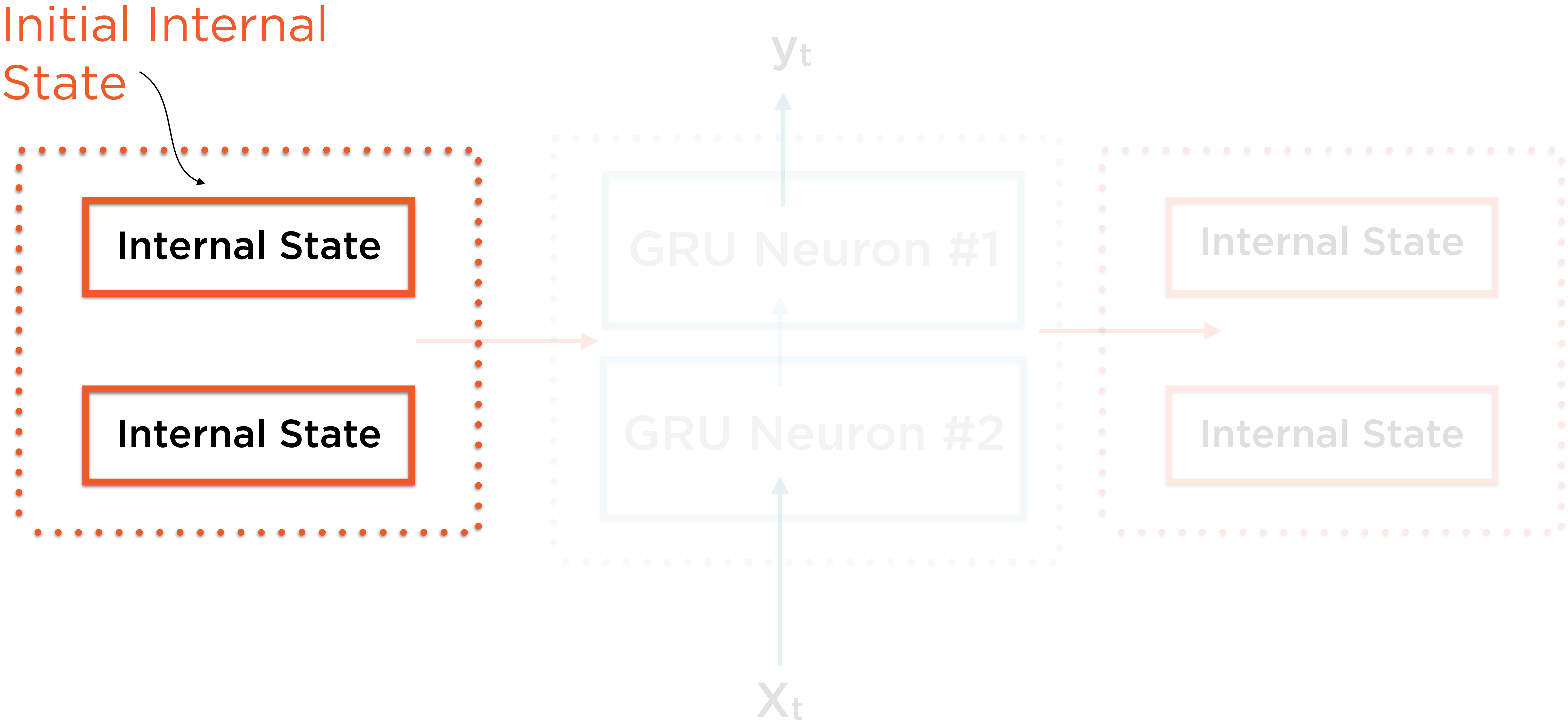
Re-using Internal State



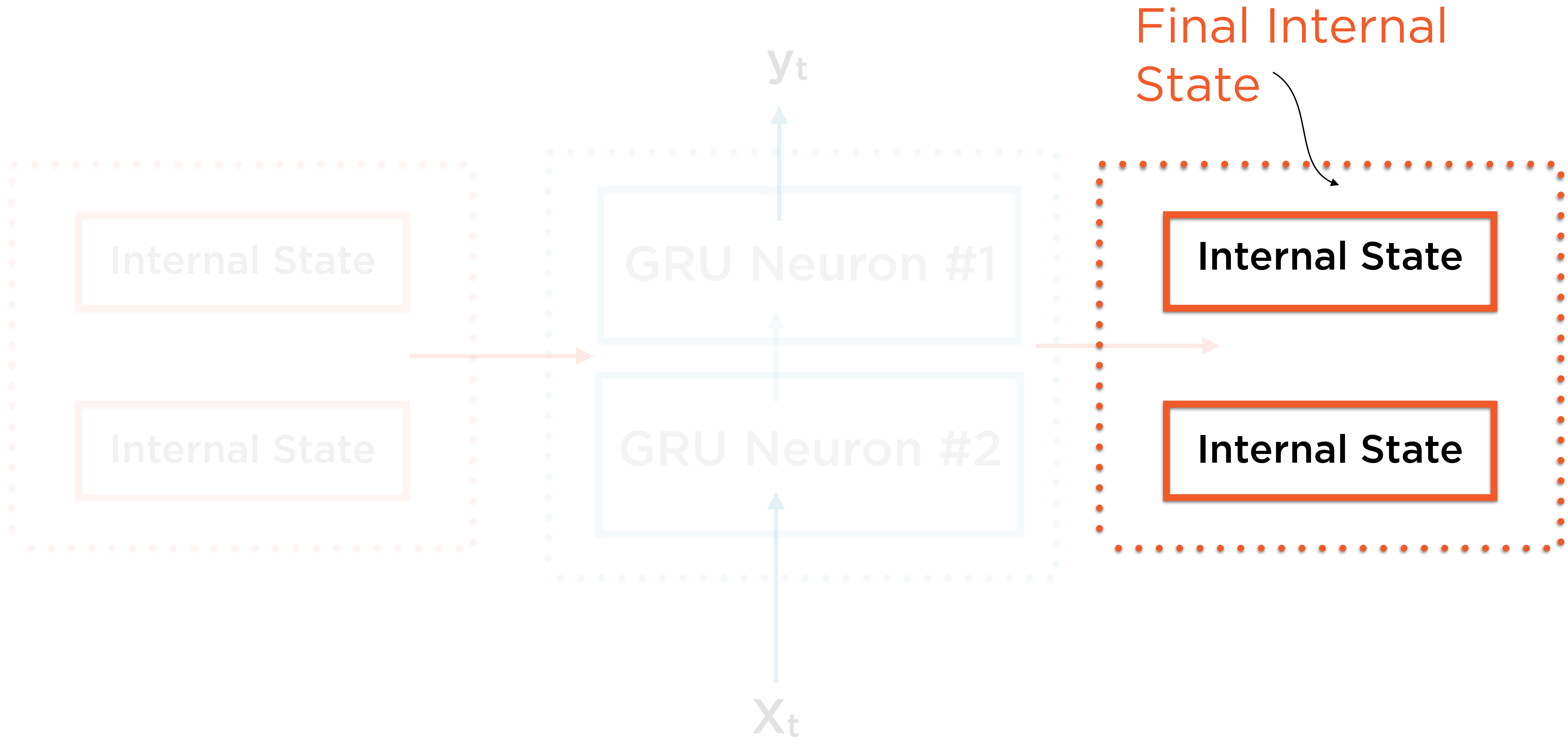
Re-using Internal State



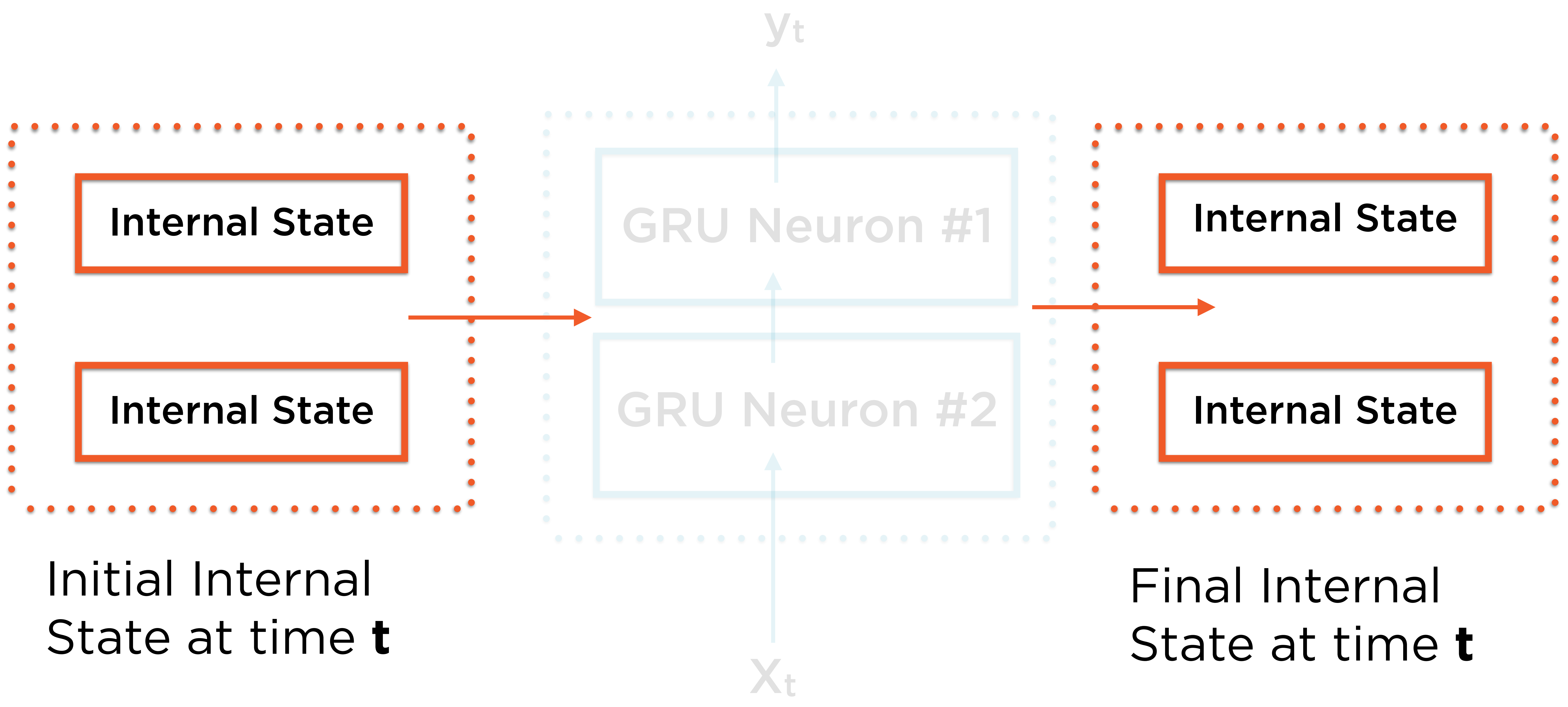
Re-using Internal State



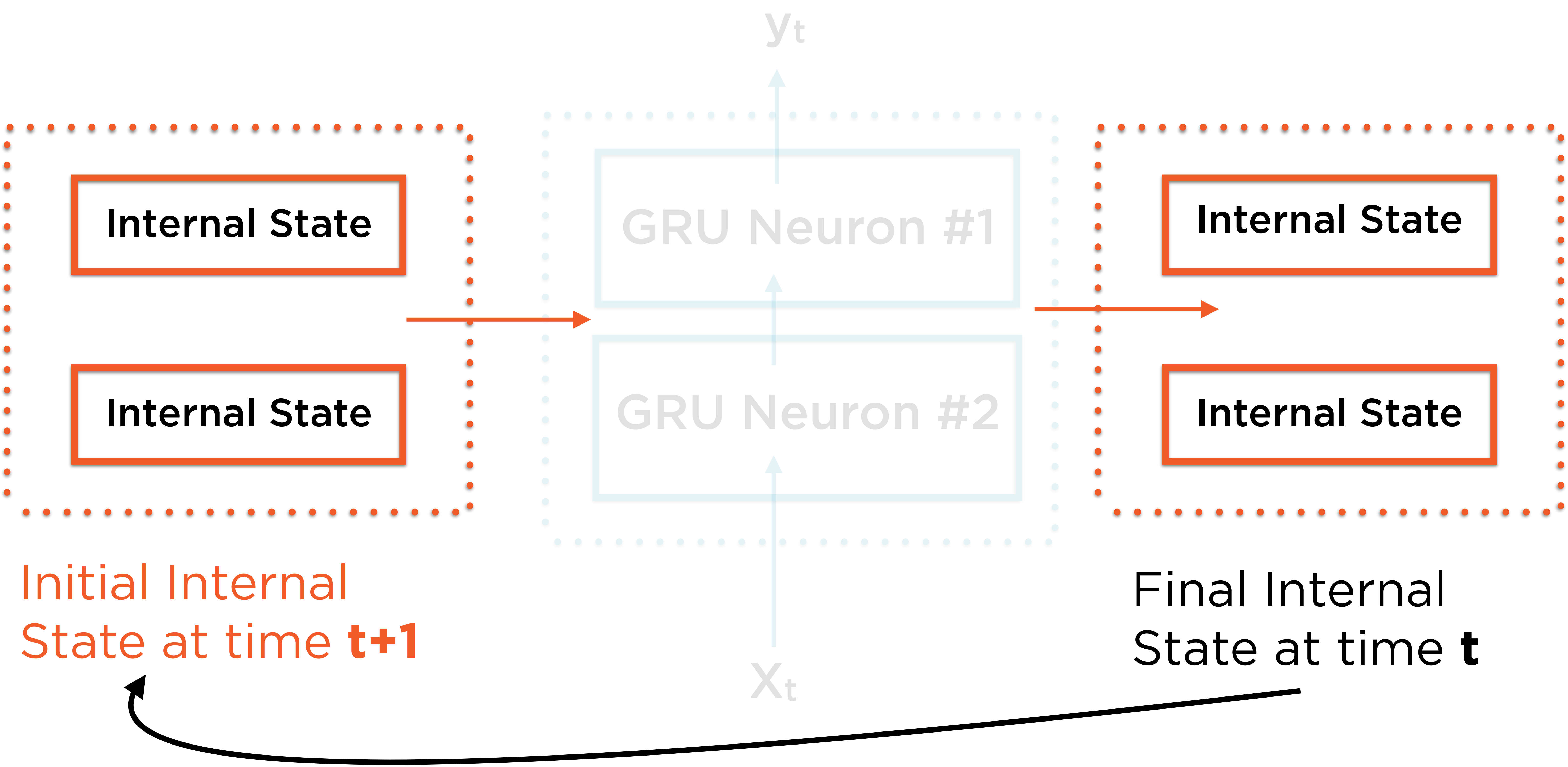
Re-using Internal State



Re-using Internal State



Re-using Internal State



Character Prediction to Generate Text

Problem statement: Given a sequence, predict what follows

Training and Prediction

Training

Train RNN with a huge dataset of character sequences

Prediction

Use trained model to generate text by feeding back internal state

Training and Prediction

Training

Train RNN with a huge dataset of character sequences

Prediction

Use trained model to generate text by feeding back internal state

Training Dataset of Technical Papers


```
<?xml version="1.0" encoding="UTF-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
  <link href="http://arxiv.org/api/query?search_query%3Dall%3Aneural%26id_list%3D%26start%3D0%26max_results%3D10" rel="self" type="application/atom+xml"/>
  <title type="html">ArXiv Query: search_query=all:neural&amp;id_list=&amp;start=0&amp;max_results=10</title>
  <id>http://arxiv.org/api/8JSYXq/Mw8td9LFNvtk3N0qiBr4</id>
  <updated>2018-01-08T00:00:00-05:00</updated>
  <opensearch:totalResults xmlns:opensearch="http://a9.com/-/spec/opensearch/1.1/">31759</opensearch:totalResults>
  <opensearch:startIndex xmlns:opensearch="http://a9.com/-/spec/opensearch/1.1/">0</opensearch:startIndex>
  <opensearch:itemsPerPage xmlns:opensearch="http://a9.com/-/spec/opensearch/1.1/">10</opensearch:itemsPerPage>
  <entry>
    <id>http://arxiv.org/abs/cs/0504056v1</id>
    <updated>2005-04-13T13:59:55Z</updated>
    <published>2005-04-13T13:59:55Z</published>
    <title>Self-Organizing Multilayered Neural Networks of Optimal Complexity</title>
    <summary>  The principles of self-organizing the neural networks of optimal complexity
is considered under the unrepresentative learning set. The method of
self-organizing the multi-layered neural networks is offered and used to train
the logical neural networks which were applied to the medical diagnostics.
</summary>
    <author>
      <name>V. Schetinin</name>
    </author>
    <link href="http://arxiv.org/abs/cs/0504056v1" rel="alternate" type="text/html"/>
    <link title="pdf" href="http://arxiv.org/pdf/cs/0504056v1" rel="related" type="application/pdf"/>
    <arxiv:primary_category xmlns:arxiv="http://arxiv.org/schemas/atom" term="cs.NE" scheme="http://arxiv.org/schemas/atom"/>
    <category term="cs.NE" scheme="http://arxiv.org/schemas/atom"/>
    <category term="cs.AI" scheme="http://arxiv.org/schemas/atom"/>
  </entry>
  <entry>
    <id>http://arxiv.org/abs/cs/0608073v1</id>
    <updated>2006-08-18T08:28:23Z</updated>
    <published>2006-08-18T08:28:23Z</published>
    <title>Parametrical Neural Networks and Some Other Similar Architectures</title>
    <summary>  A review of works on associative neural networks accomplished during last
four years in the Institute of Optical Neural Technologies RAS is given. The
presentation is based on description of parametrical neural networks (PNN). For
today PNN have record recognizing characteristics (storage capacity, noise
immunity and speed of operation). Presentation of basic ideas and principles is
accentuated.
```

```
  <opensearch:itemsPerPage xmlns:opensearch="http://a9.com/-/spec/opensearch/1.1/"
  <entry>
    <id>http://arxiv.org/abs/cs/0504056v1</id>
    <updated>2005-04-13T13:59:55Z</updated>
    <published>2005-04-13T13:59:55Z</published>
    <title>Self-Organizing Multilayered Neural Networks of Optimal Complexity</tit
    <summary>  The principles of self-organizing the neural networks of optimal co
is considered under the unrepresentative learning set. The method of
self-organizing the multi-layered neural networks is offered and used to train
the logical neural networks which were applied to the medical diagnostics.
</summary>
    <author>
      <name>V. Schetinin</name>
    </author>
    <link href="http://arxiv.org/abs/cs/0504056v1" rel="alternate" type="text/html
    <link title="pdf" href="http://arxiv.org/pdf/cs/0504056v1" rel="related" type=
    <arxiv:primary_category xmlns:arxiv="http://arxiv.org/schemas/atom" term="cs.N
    <category term="cs.NE" scheme="http://arxiv.org/schemas/atom"/>
    <category term="cs.AI" scheme="http://arxiv.org/schemas/atom"/>
  </entry>
  <entry>
    <id>http://arxiv.org/abs/cs/0608073v1</id>
    <updated>2006-08-18T08:28:23Z</updated>
    <published>2006-08-18T08:28:23Z</published>
```

Sliding Window in Training

Create window of 50 characters

The quick brown fox jumps over the lazy dog

Sliding Window in Training

Create window of 50 characters

The quick brown fox jumps o

Sliding Window in Training

This window is our sequence length

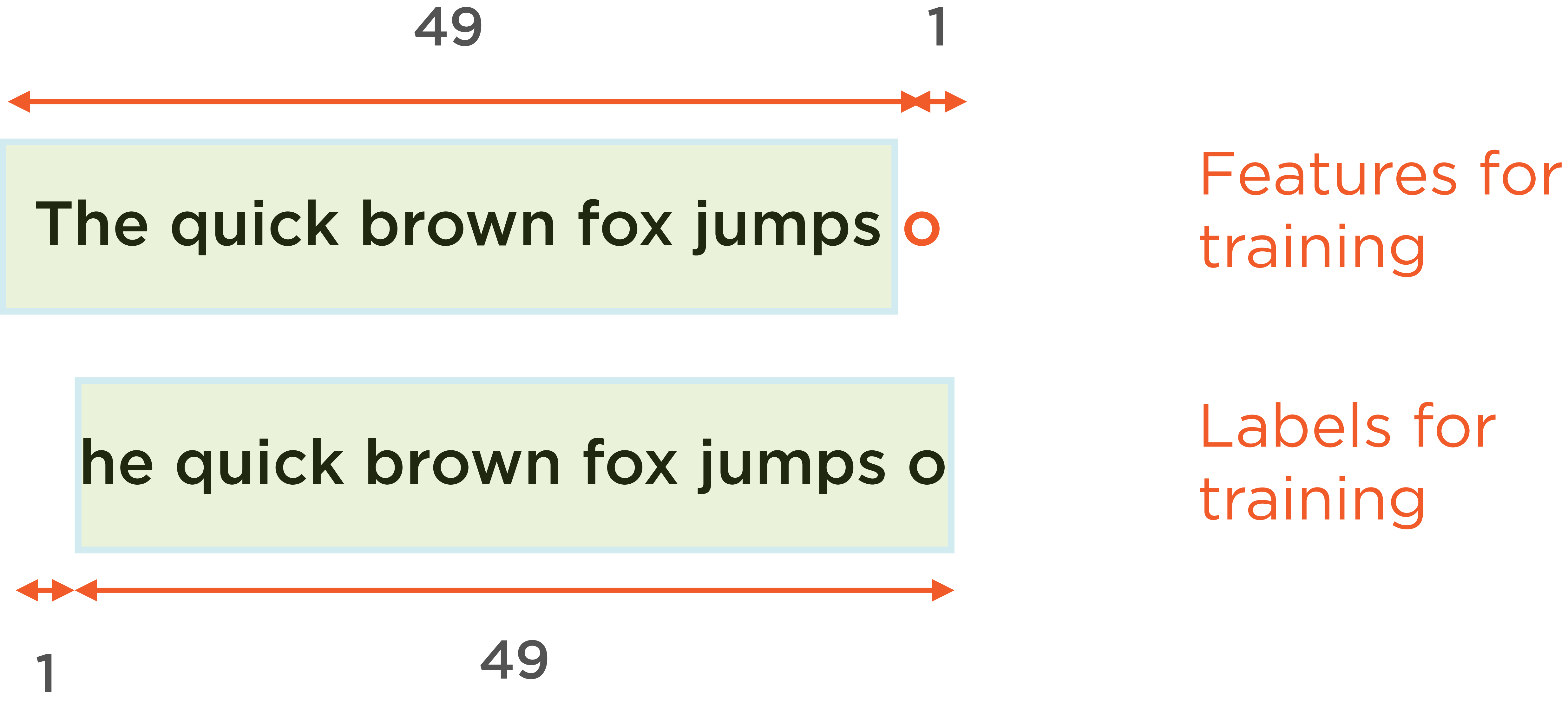
The quick brown fox jumps o

Sliding Window in Training

Use 0:48 as feature **1**

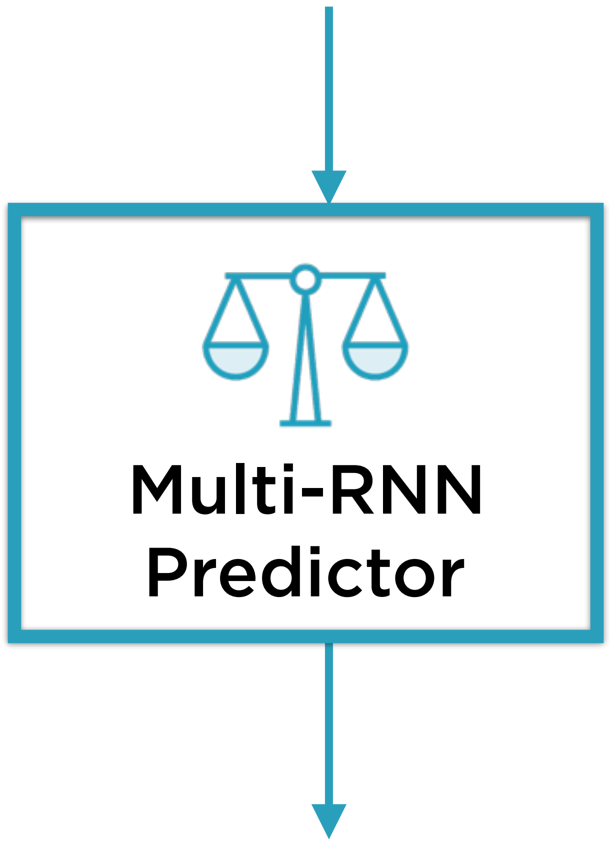
The quick brown fox jumps o

Sliding Window in Training



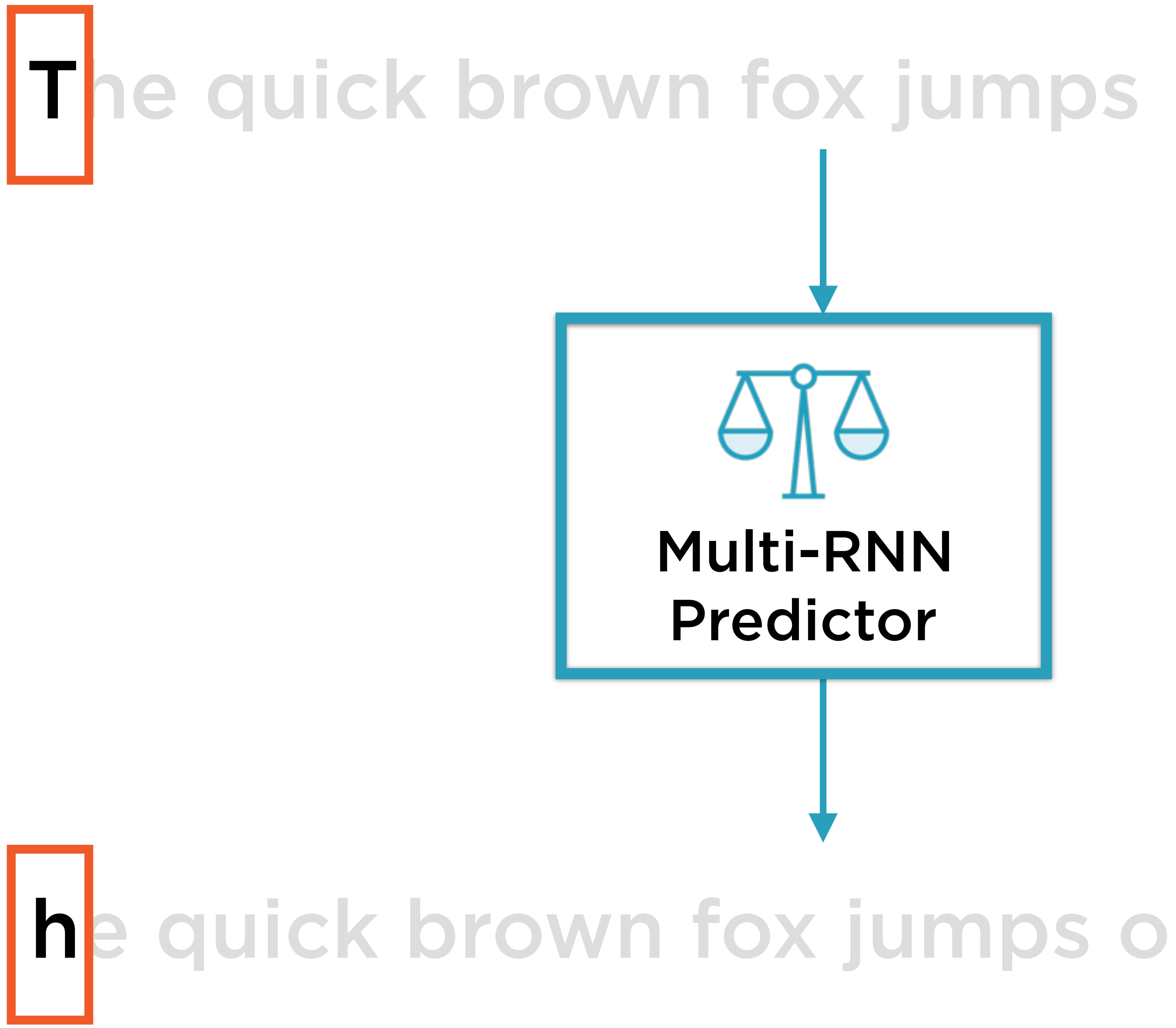
Training Phase

The quick brown fox jumps

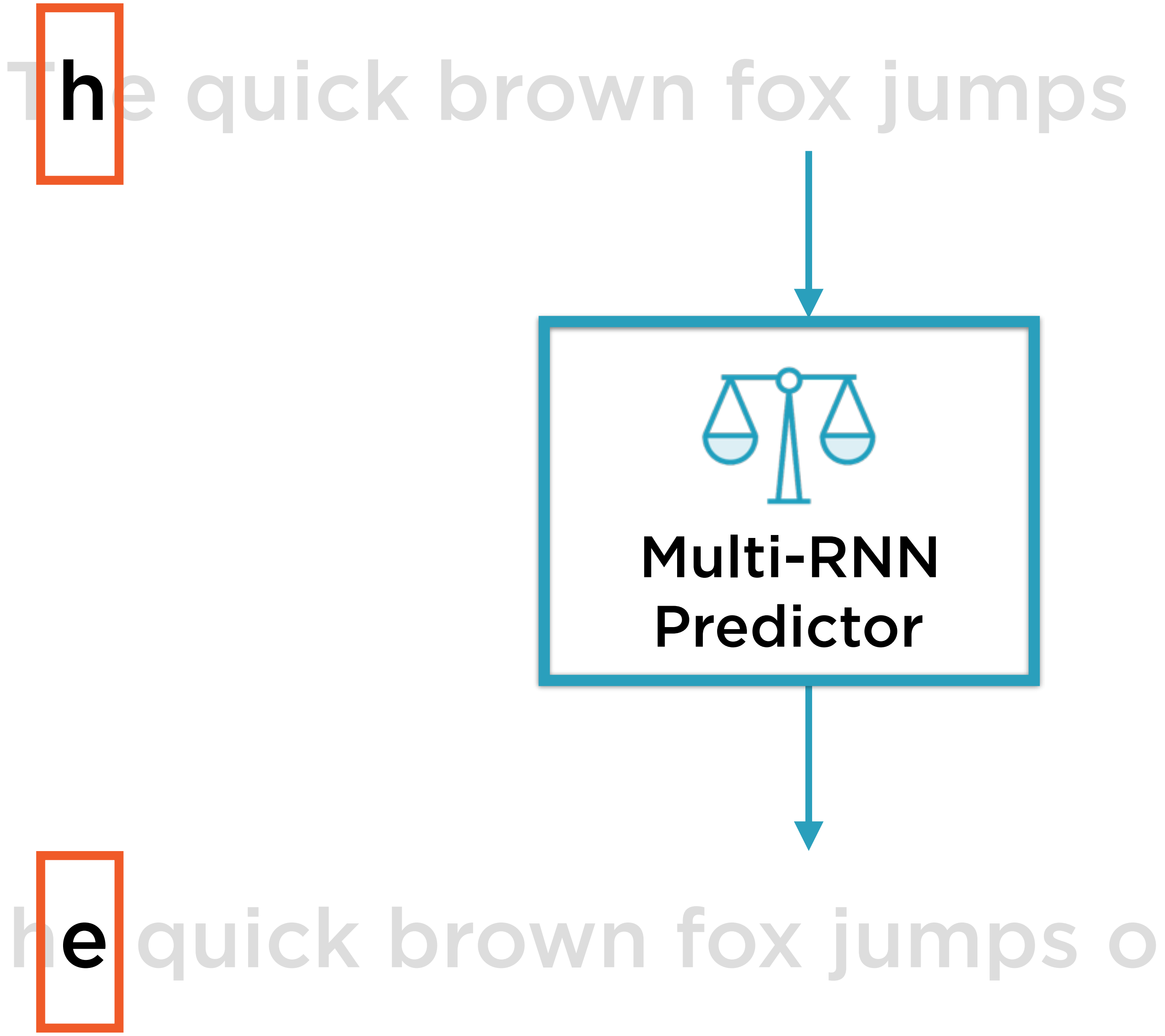


he quick brown fox jumps o

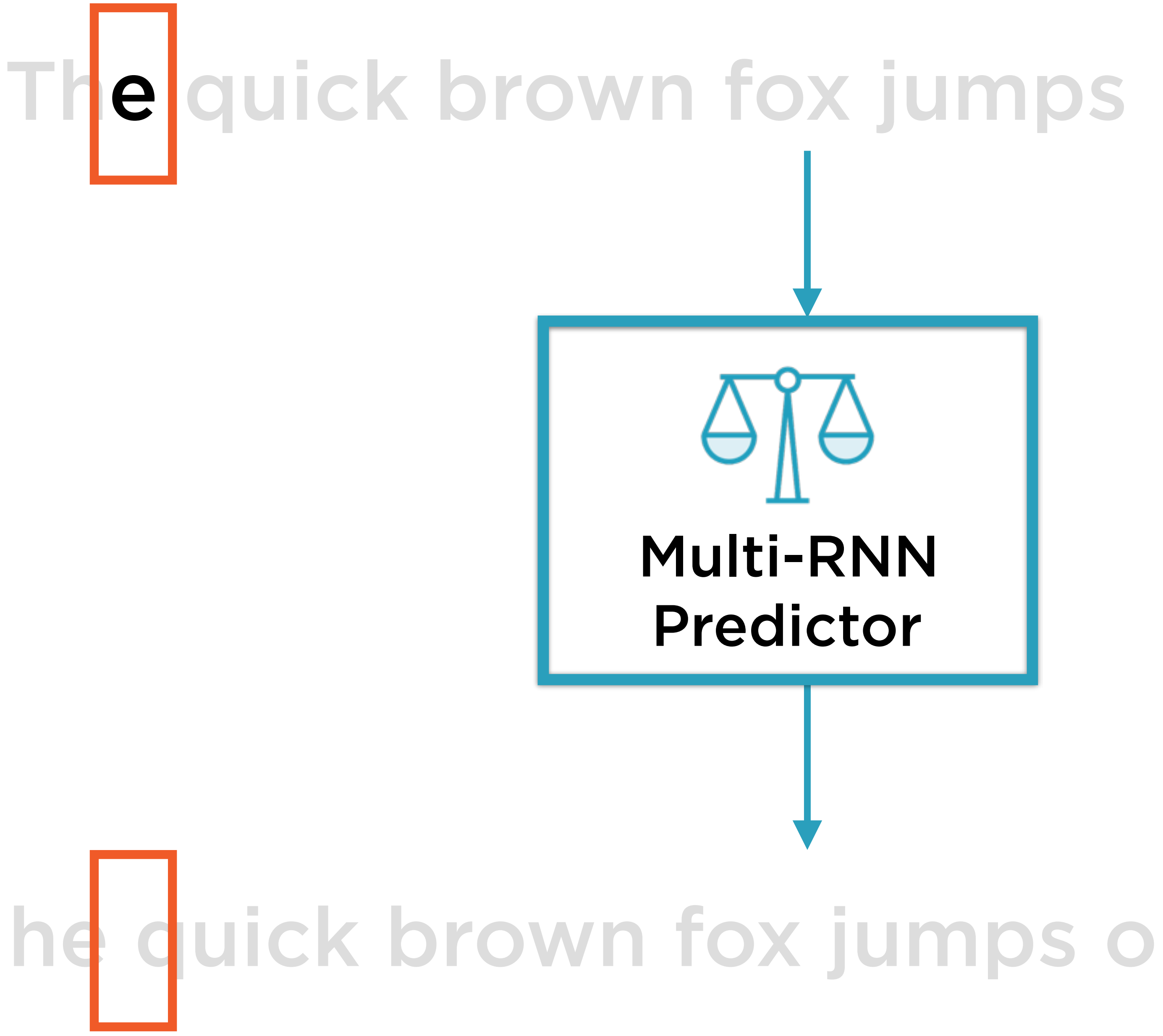
Training Phase



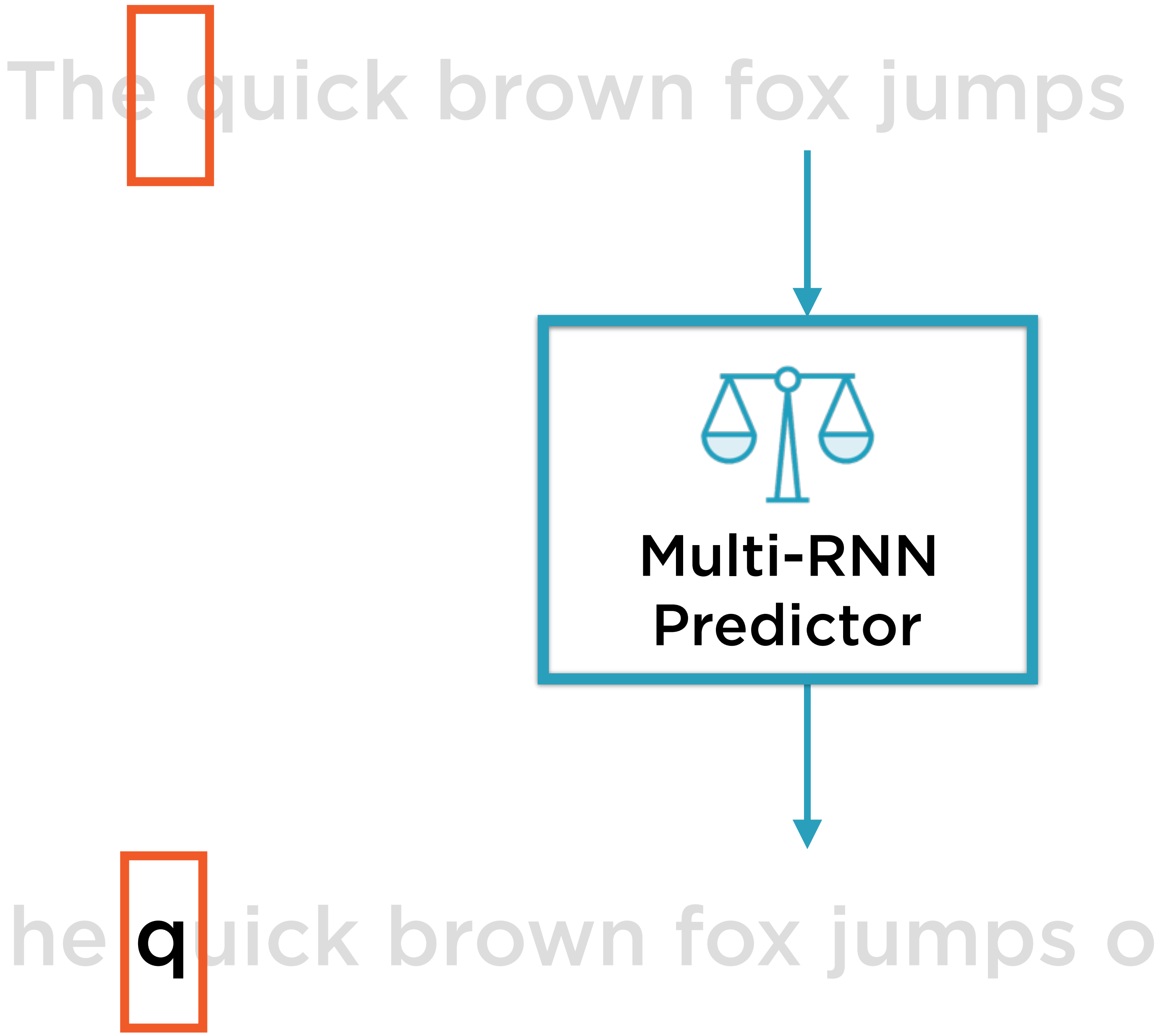
Training Phase



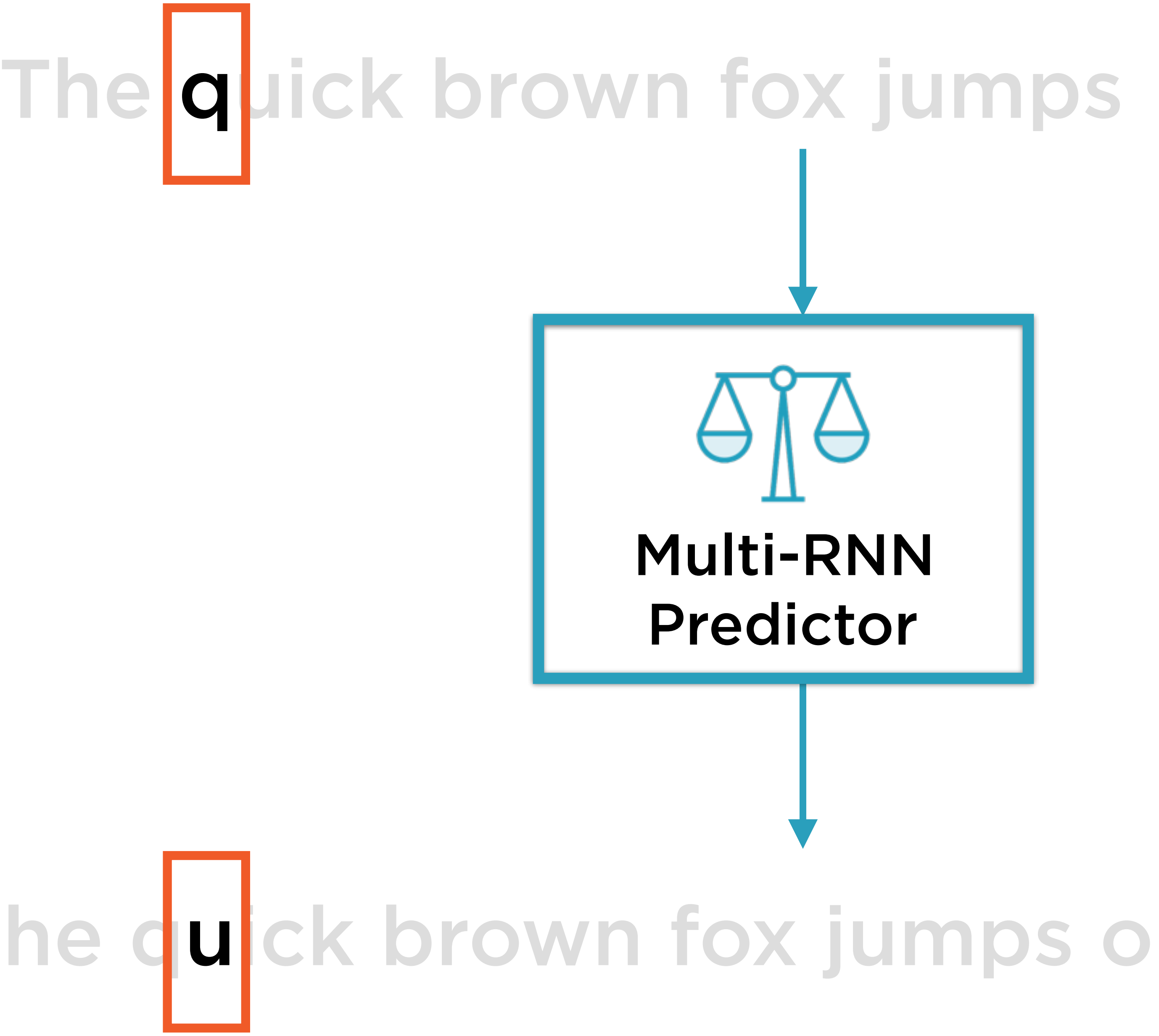
Training Phase



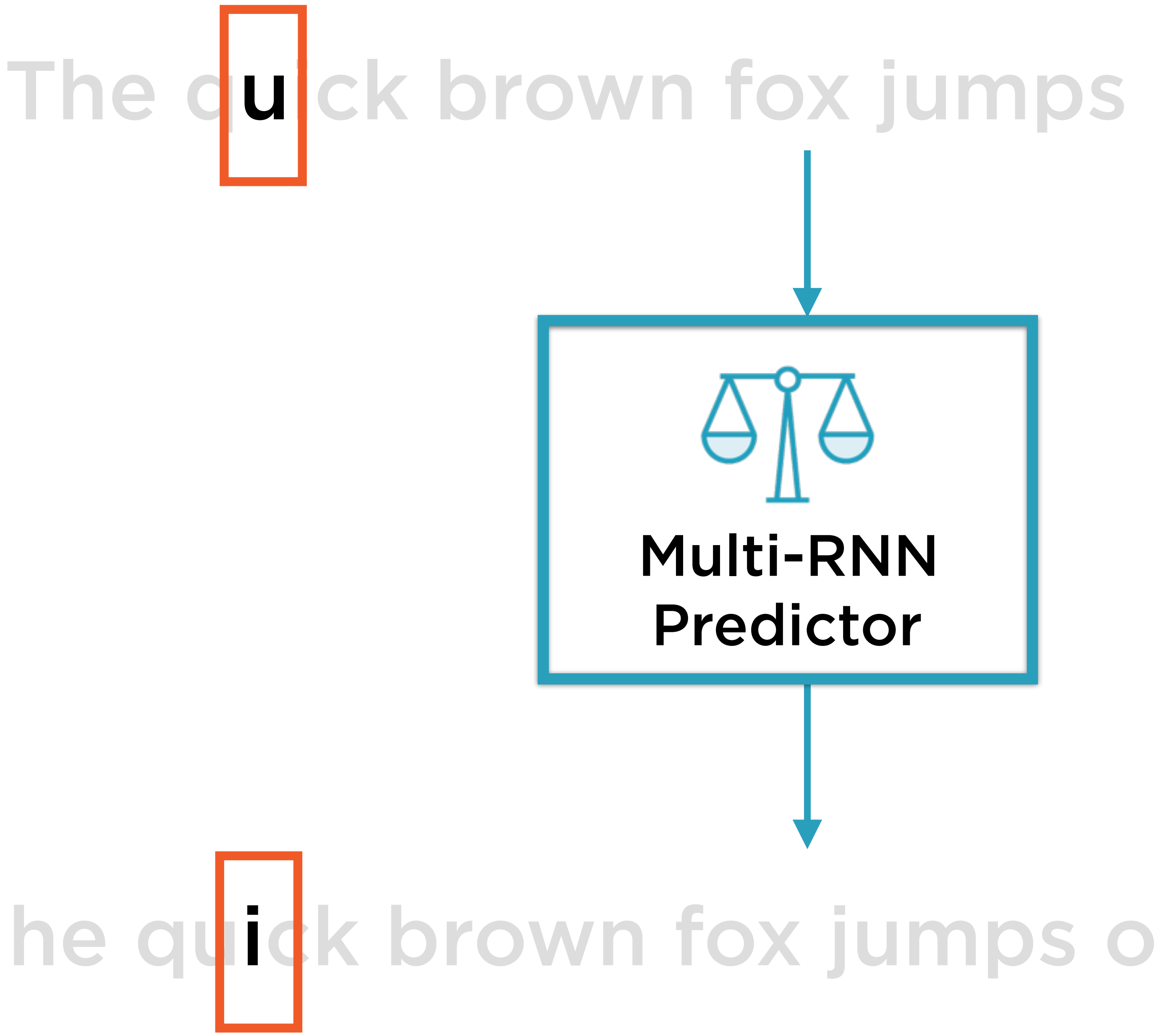
Training Phase



Training Phase

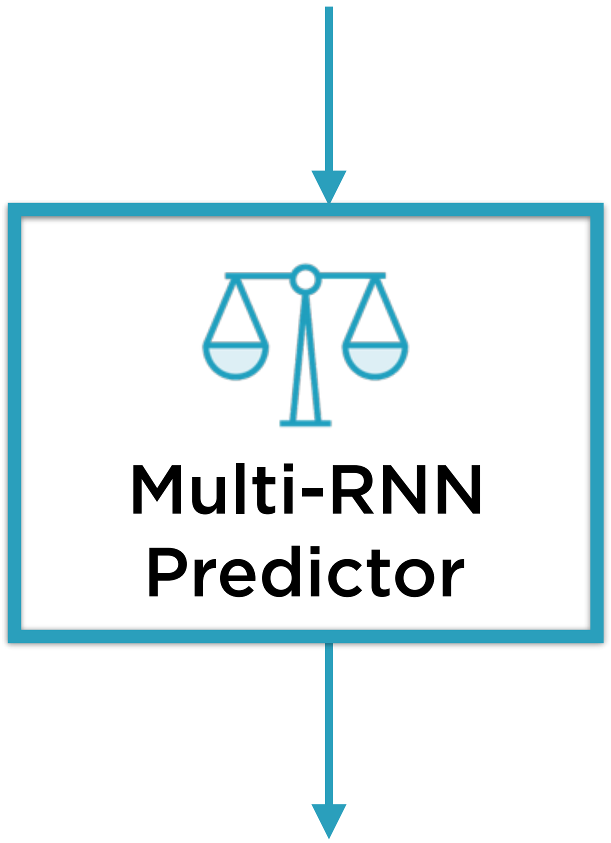


Training Phase



Training Phase

The quick brown fox jumps



he quick brown fox jumps o

Sliding Window in Training

Create window of 50 characters

The quick brown fox jumps over the lazy dog

Sliding Window in Training

Slide the window

50

The quick brown fox jumps over the lazy dog

Rinse-and-repeat

Sliding Window in Training

Slide the window

50

The quick brown fox jumps over the lazy dog

Rinse-and-repeat

Training and Prediction

Training

Train RNN with a huge dataset of character sequences

Prediction

Use trained model to generate text by feeding back internal state

Training and Prediction

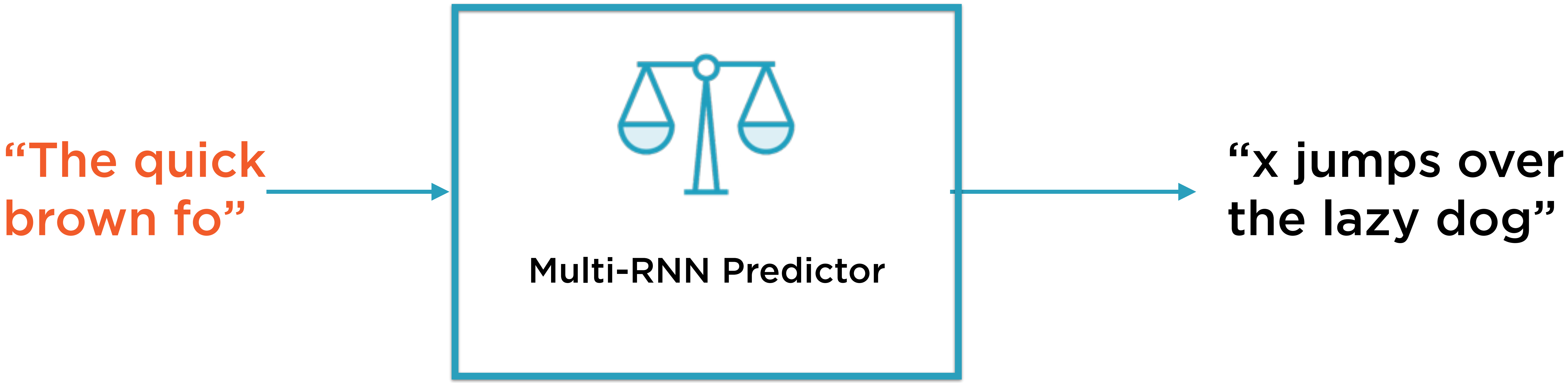
Training

Train RNN with a huge dataset of character sequences

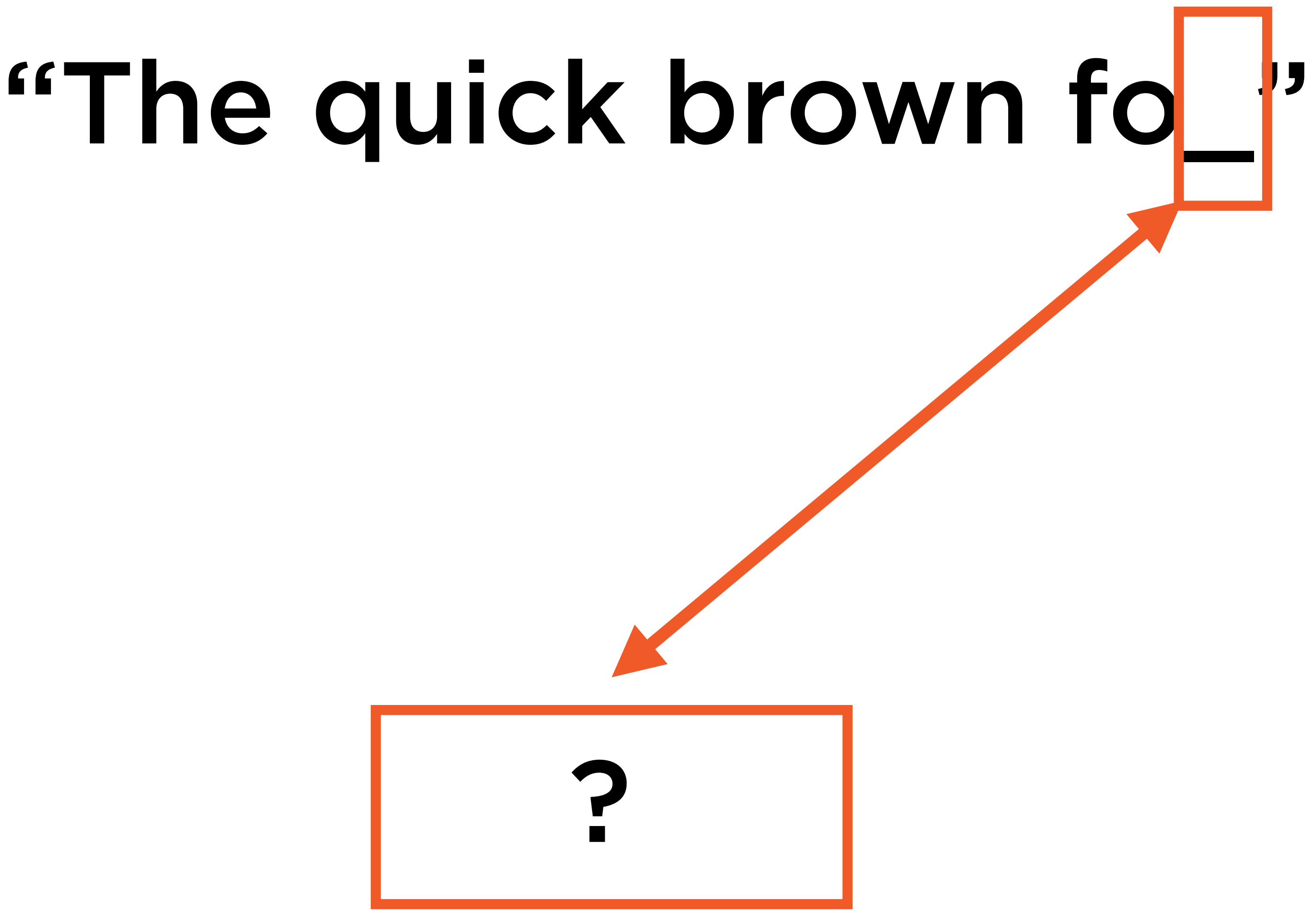
Prediction

Use trained model to generate text by feeding back internal state

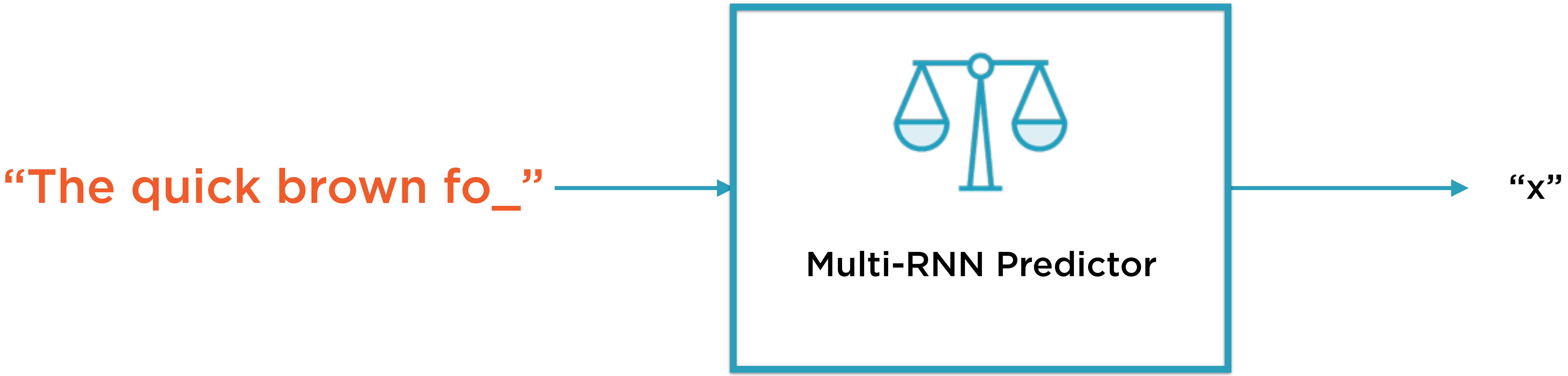
Text Prediction



Predict One Character at a Time



Predict One Character at a Time



Predict One Character at a Time

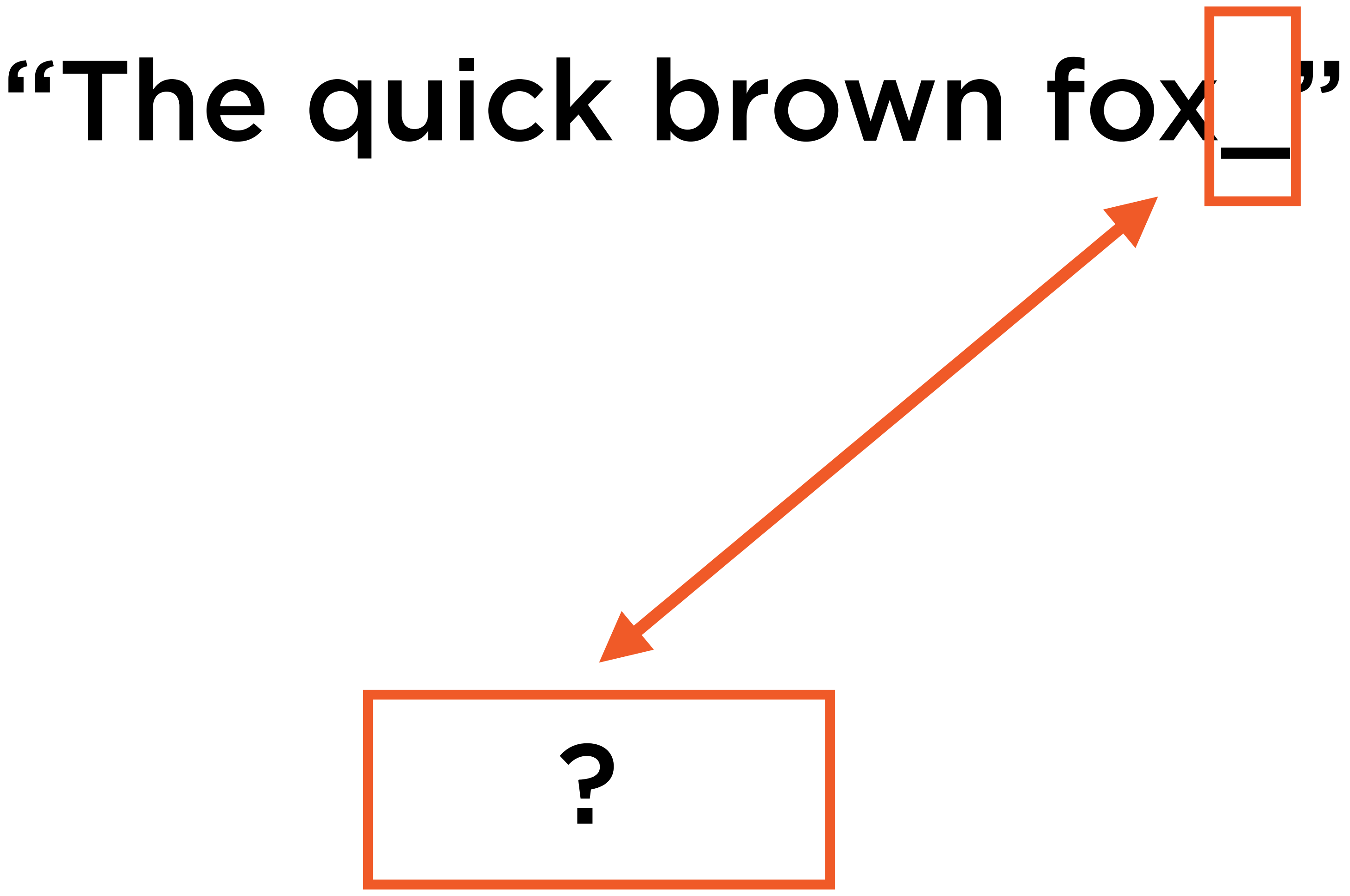
"The quick brown fo_"

x

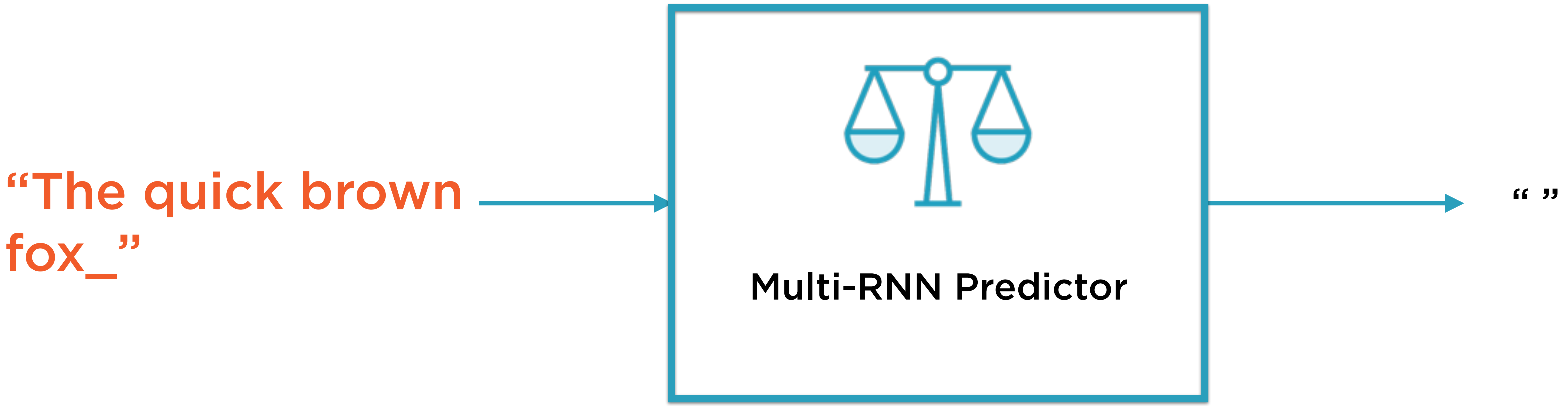
Predict One Character at a Time

"The quick brown fox"

Predict One Character at a Time



Predict One Character at a Time



Predict One Character at a Time

“The quick brown fox_”

“ “

Predict One Character at a Time

“The quick brown fox ”

Text Prediction

“The quick brown fox jumps over the lazy do_”

g

Text Prediction

“The quick brown fox jumps over the lazy dog”

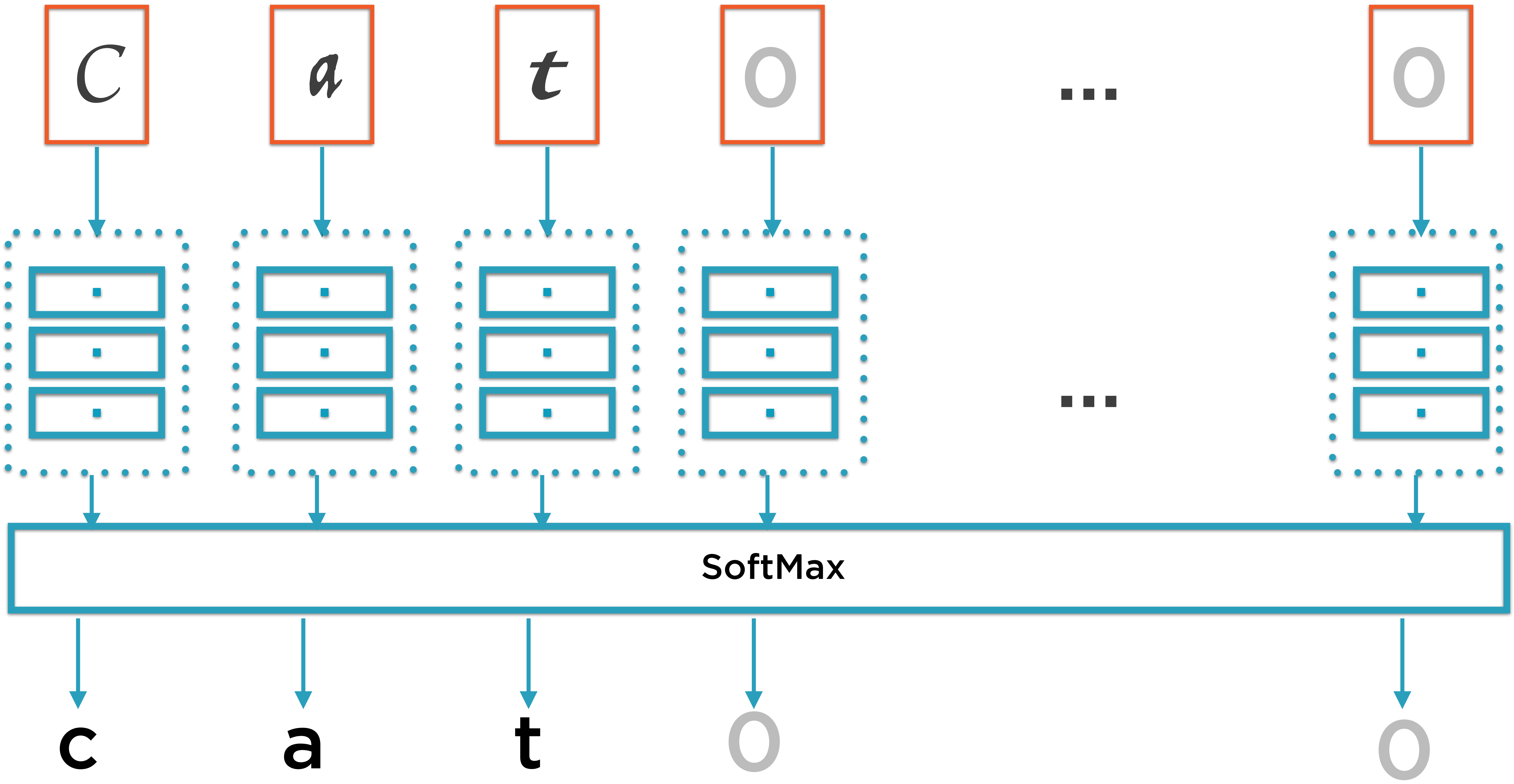
Problem statement: Given a sequence, predict what follows

Solution Outline: Use a RNN to predict words, character-by-character

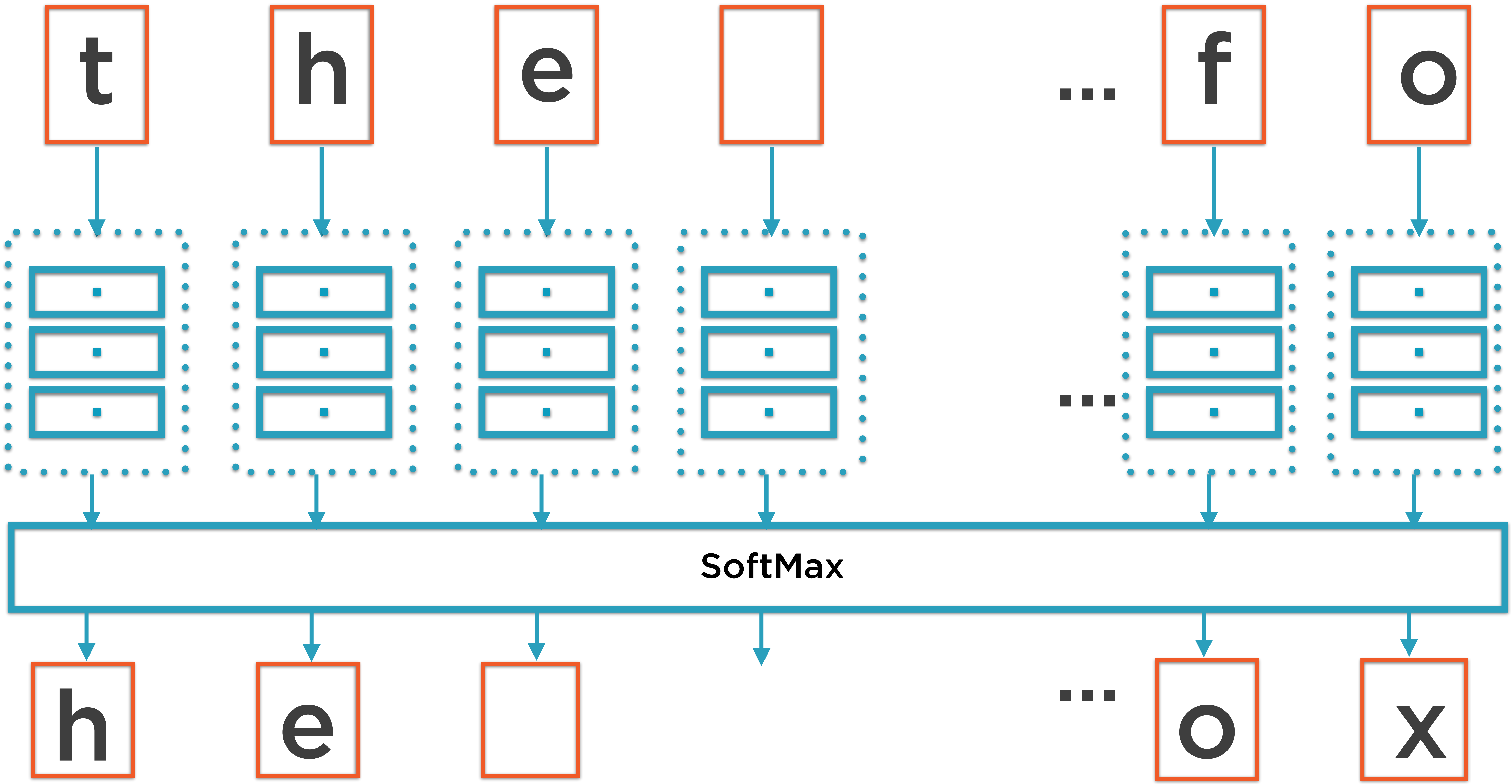
Contrasting Architectures

OCR Classification	Text Prediction
Classification	Prediction
Bi-directional RNN	Multi-RNN Cell
Input 128px image, output character	Input character, output next character

OCR: RNN Architecture



Text Prediction: RNN Architecture



Contrasting Architectures

OCR Classification

Input tensor

[batch_size, 14, 128]

Text Prediction

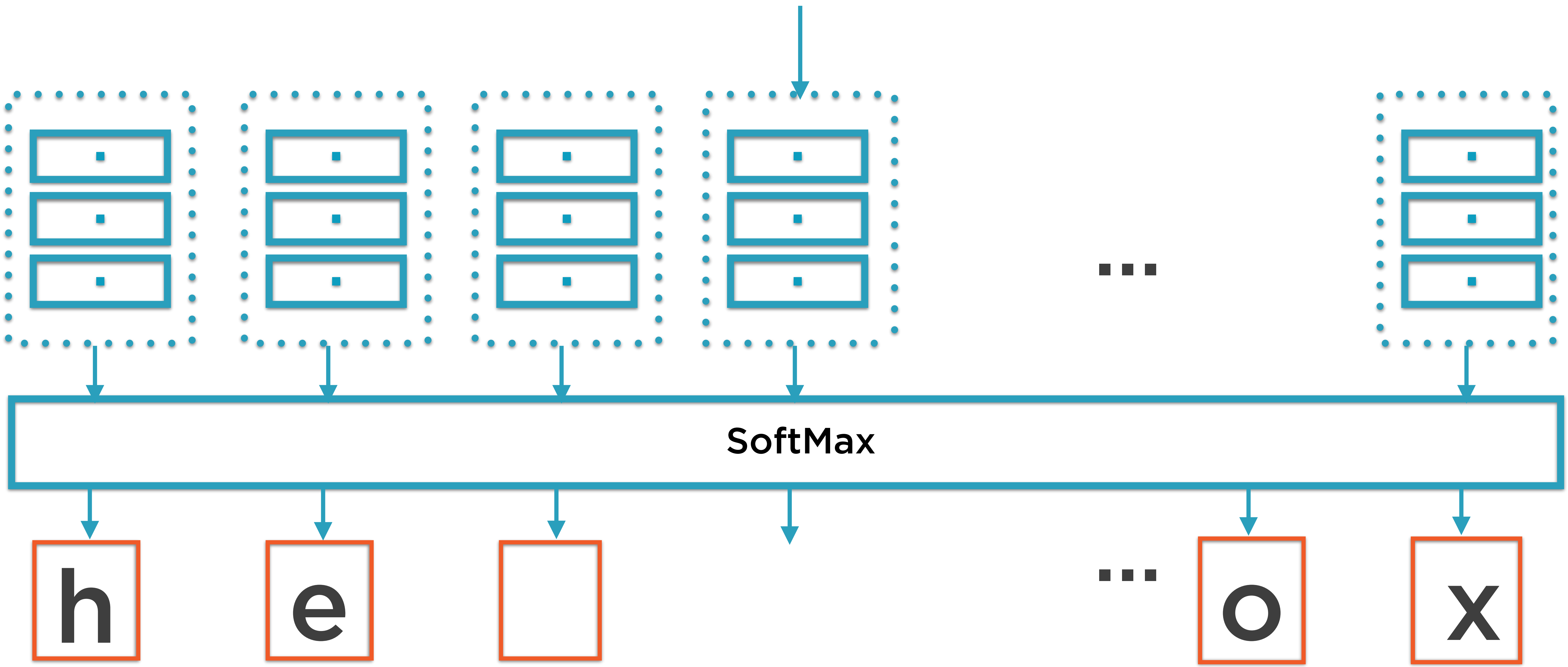
Input tensor

[batch_size, 49, 83]

Larger vocabulary because we include special characters

Text Prediction: Input Tensor for Training

[batch_size, 49, 83]



Contrasting Architectures

OCR Classification

Output tensor

[batch_size, 14, 26]

One-hot with 26 characters

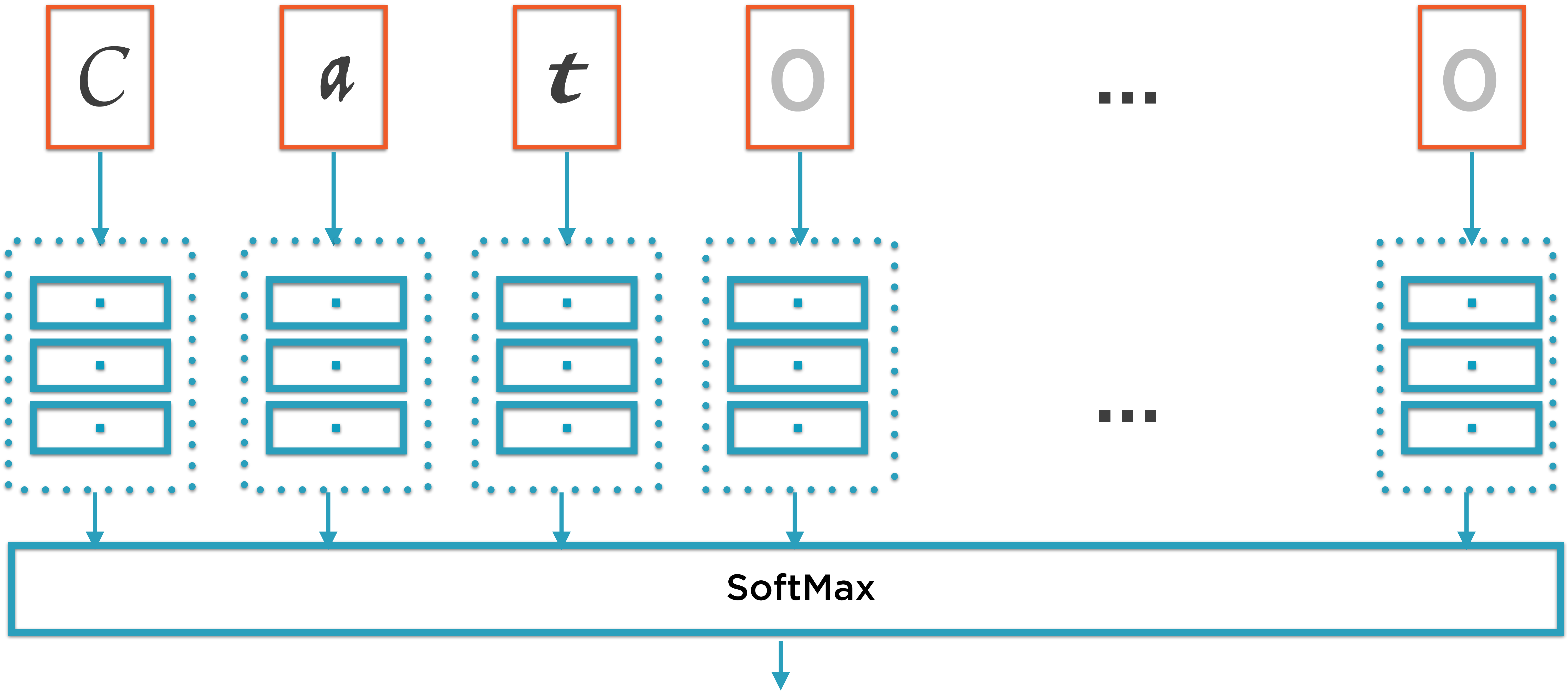
Text Prediction

Output tensor

[batch_size, 49, 83]

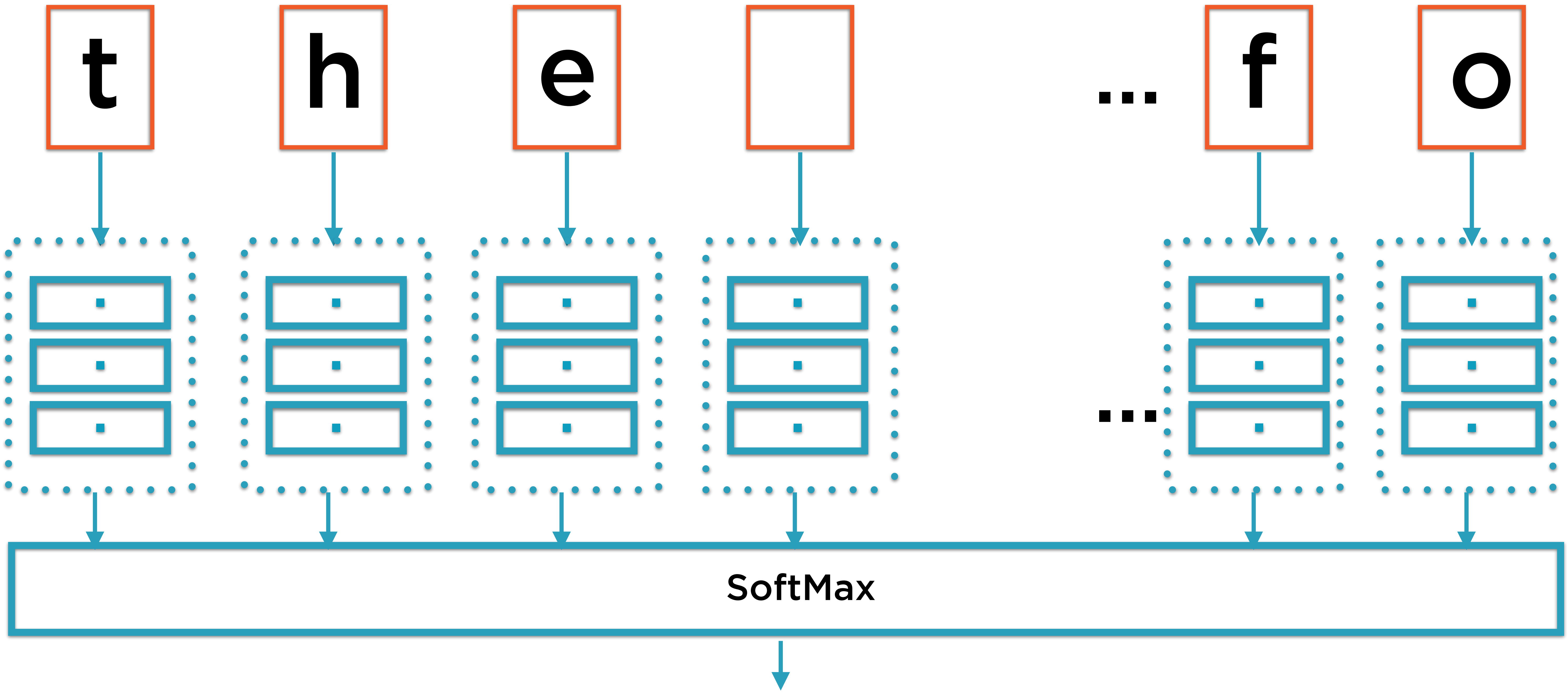
One-hot with 83 character

OCR: Output Tensor for Predicted Values



[batch_size, 14, 26]

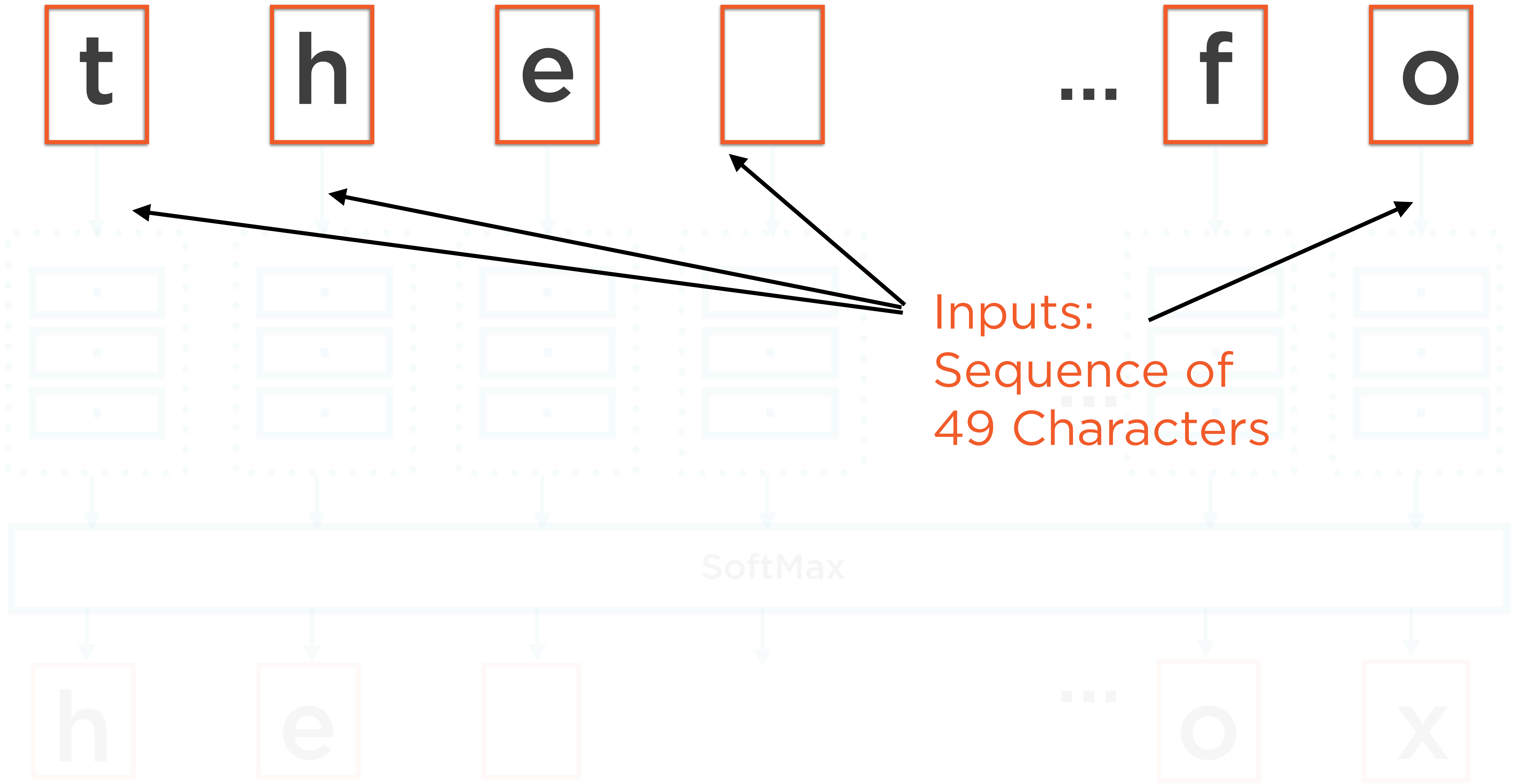
Text Prediction: RNN Architecture



[batch_size, 49, 83]

OCR Input Tensors

Text Prediction: RNN Architecture



Contrasting Architectures

OCR Classification

One RNN layer per input image

14 RNN layers

Every bidirectional cell had 300 neurons

Text Prediction

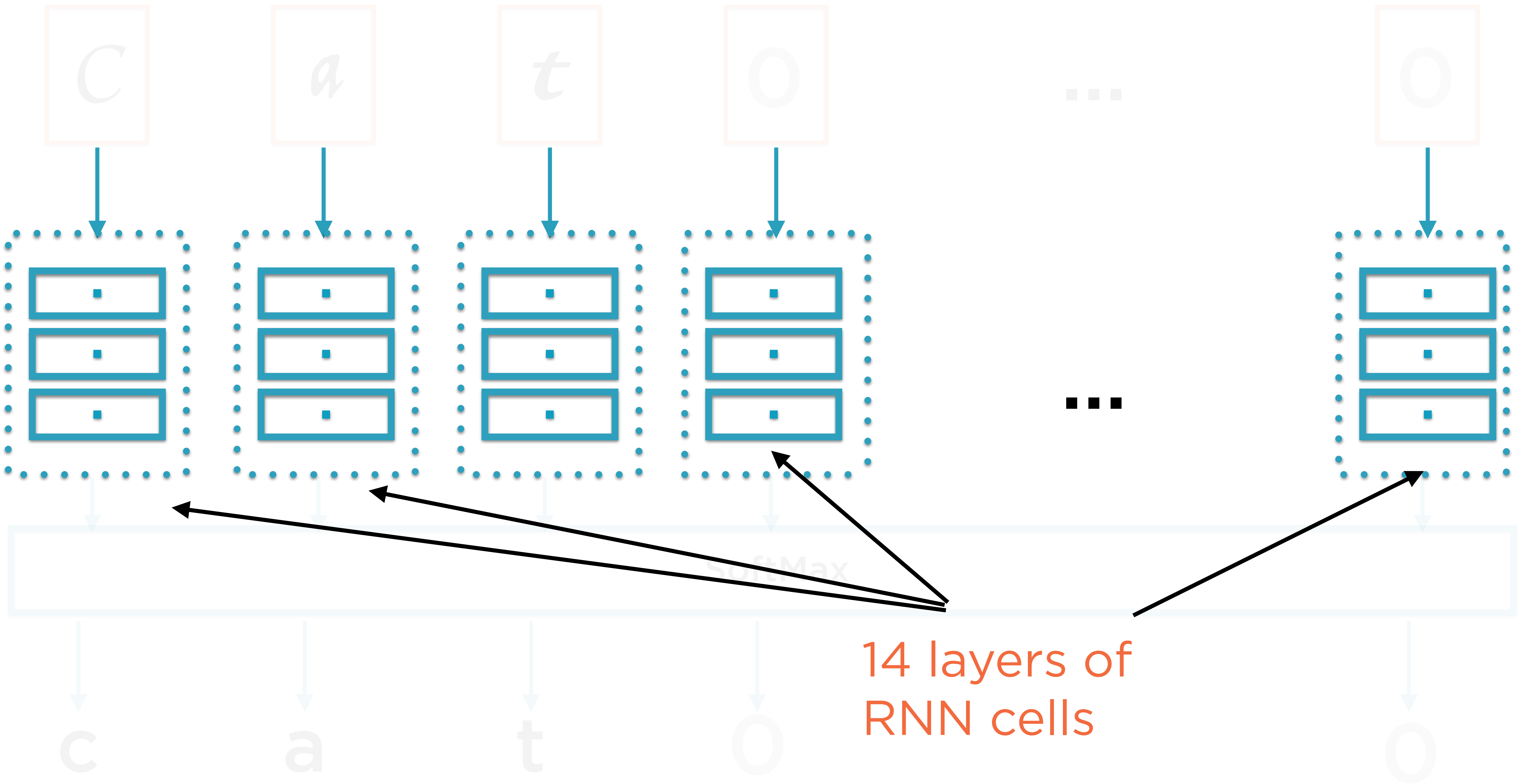
One RNN layer per input character

49 RNN layers

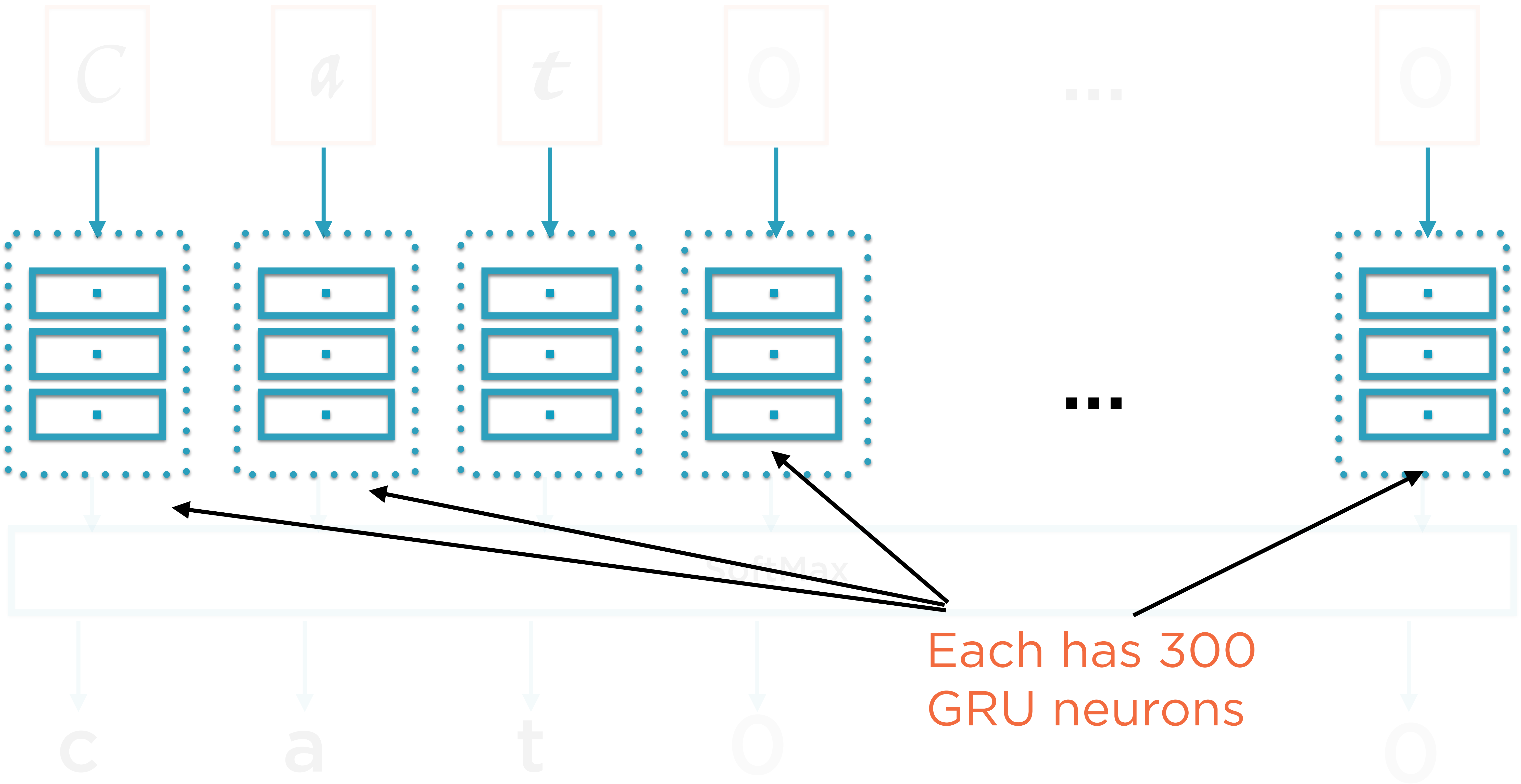
Each multi-RNN cell has 2 GRU cells

Each GRU cell has 200 neurons

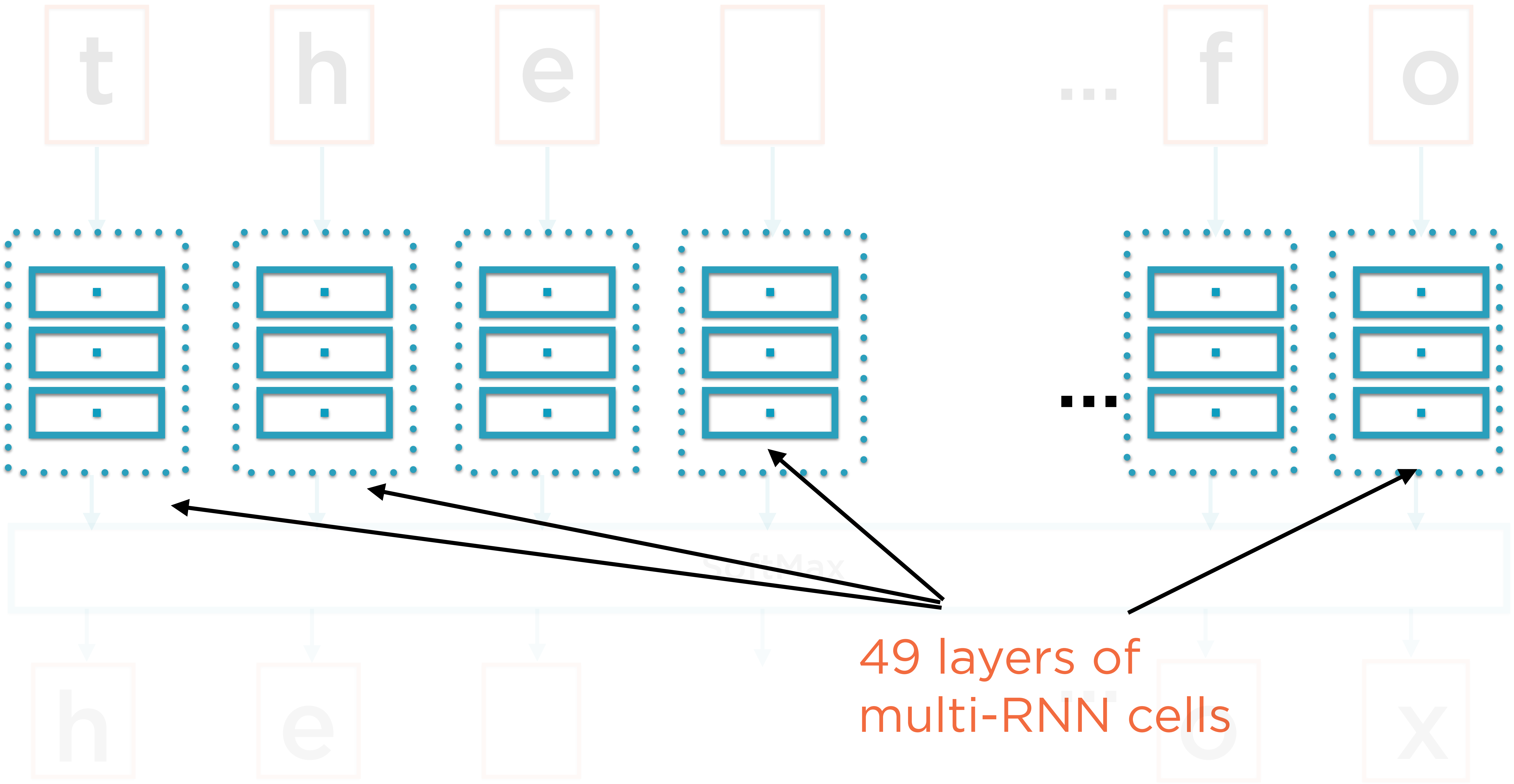
OCR: RNN Architecture



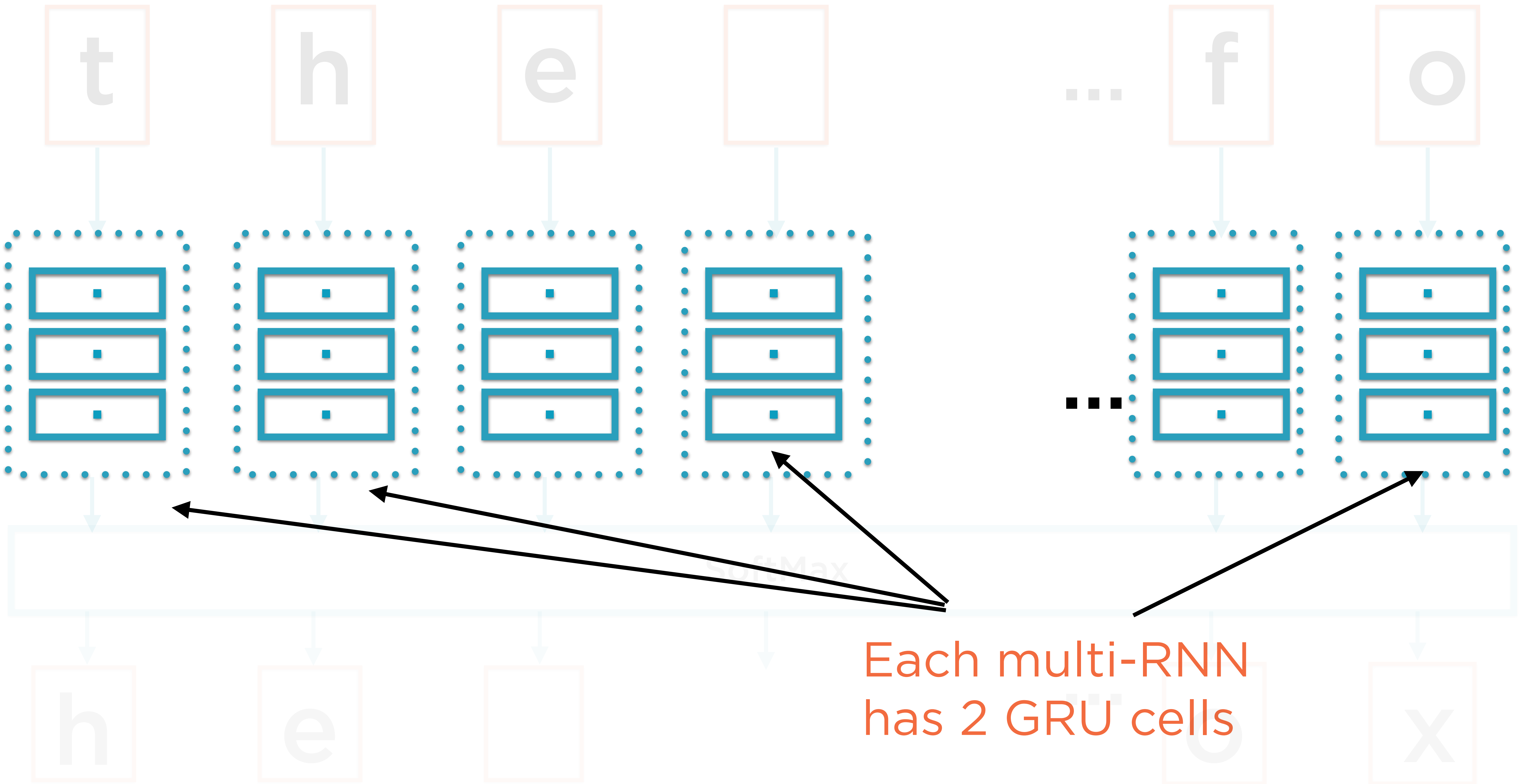
OCR: RNN Architecture



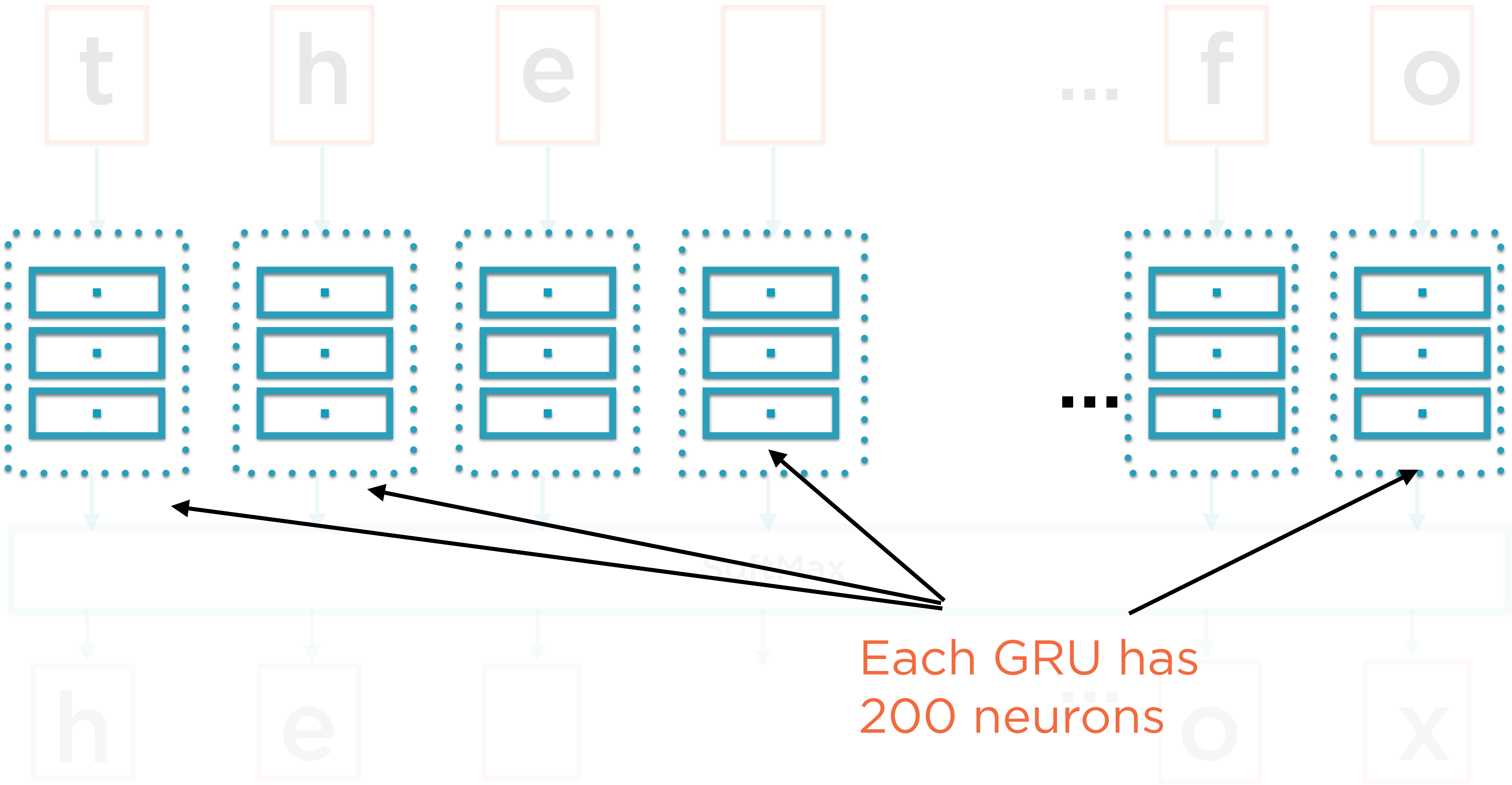
Text Prediction: RNN Architecture



Text Prediction: RNN Architecture



Text Prediction: RNN Architecture



Contrasting Architectures

OCR Classification

Shared Softmax layer

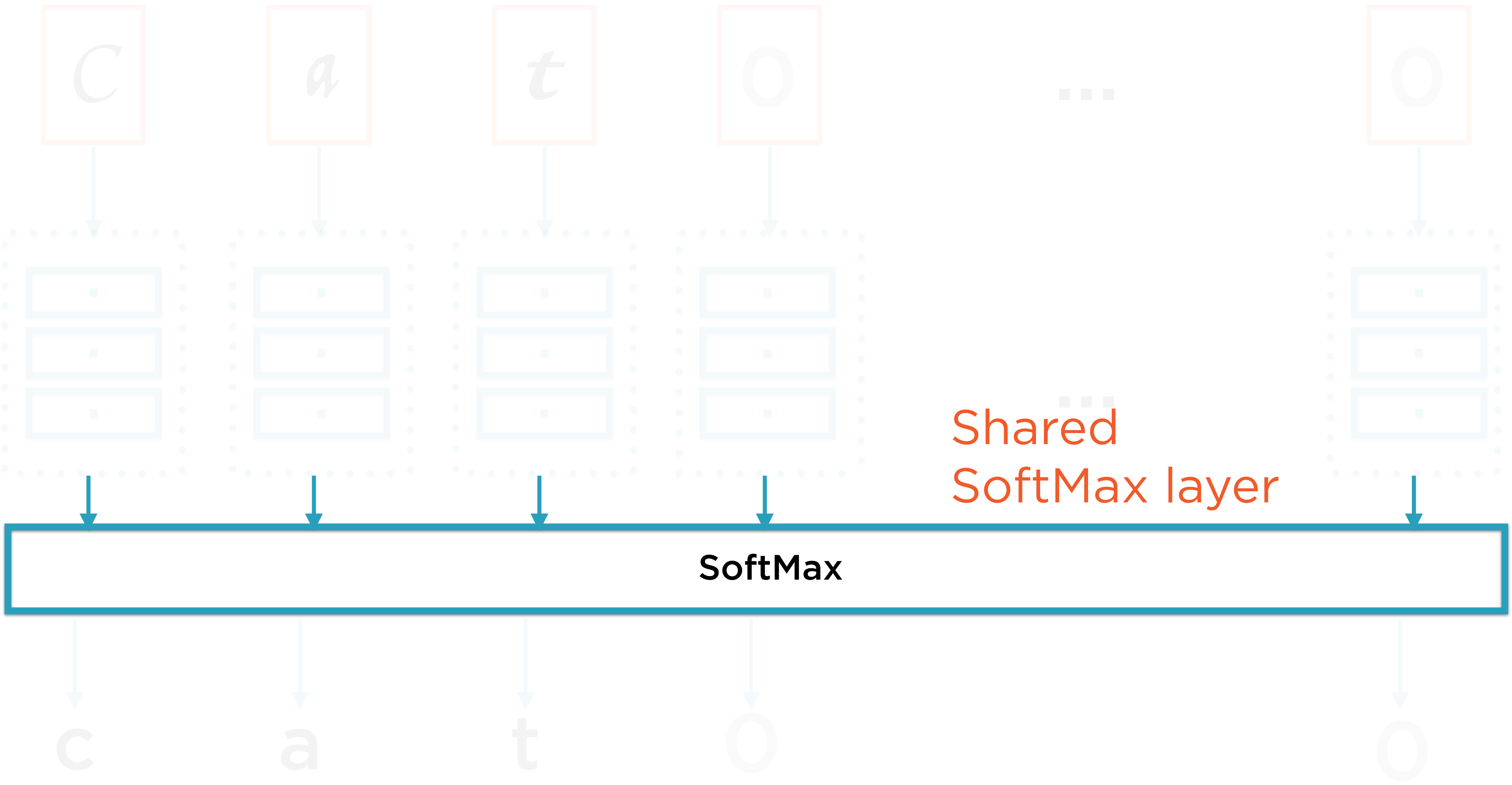
Output probabilities of 26 elements

Text Prediction

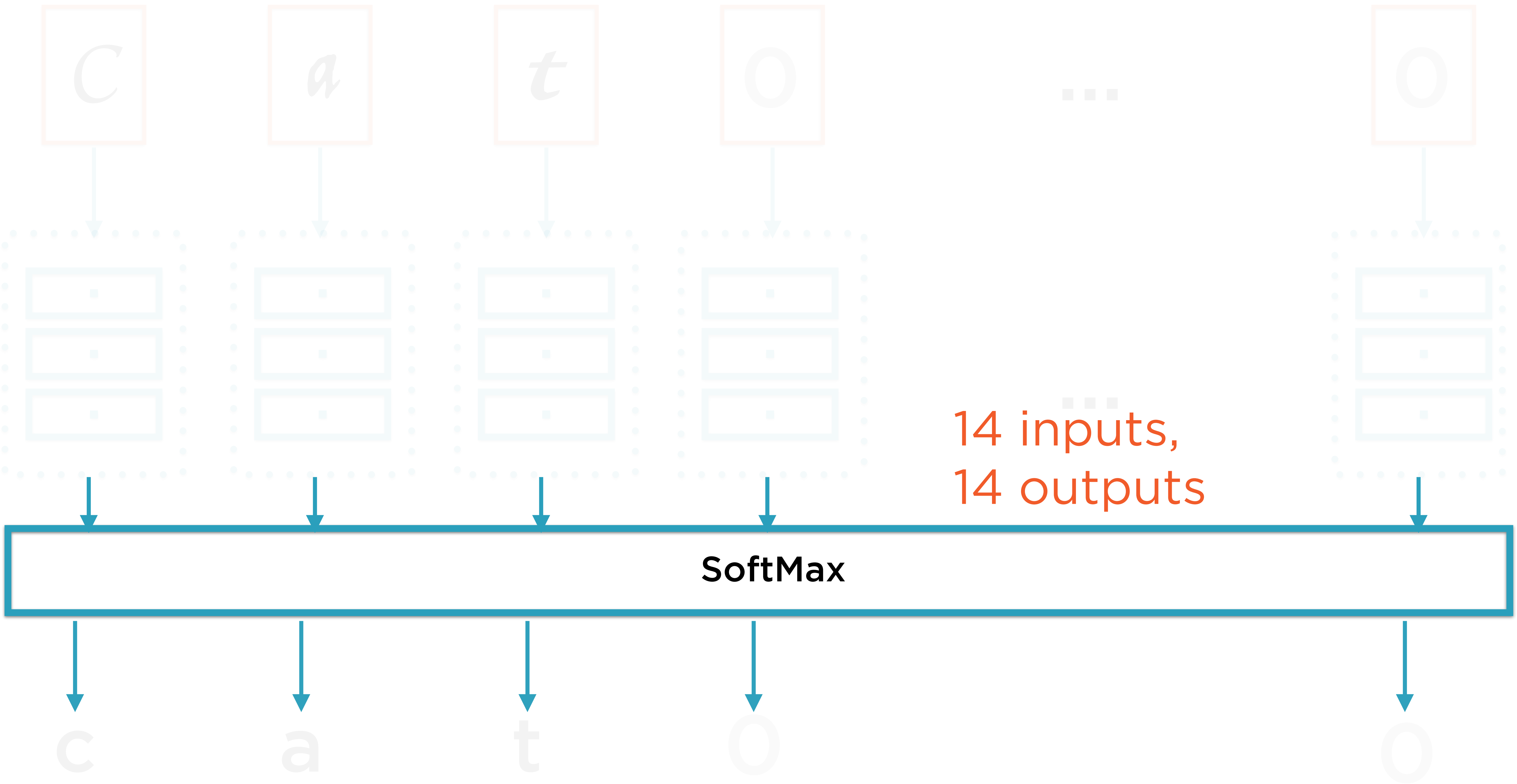
Shared Softmax layer

Output probabilities of 83 elements

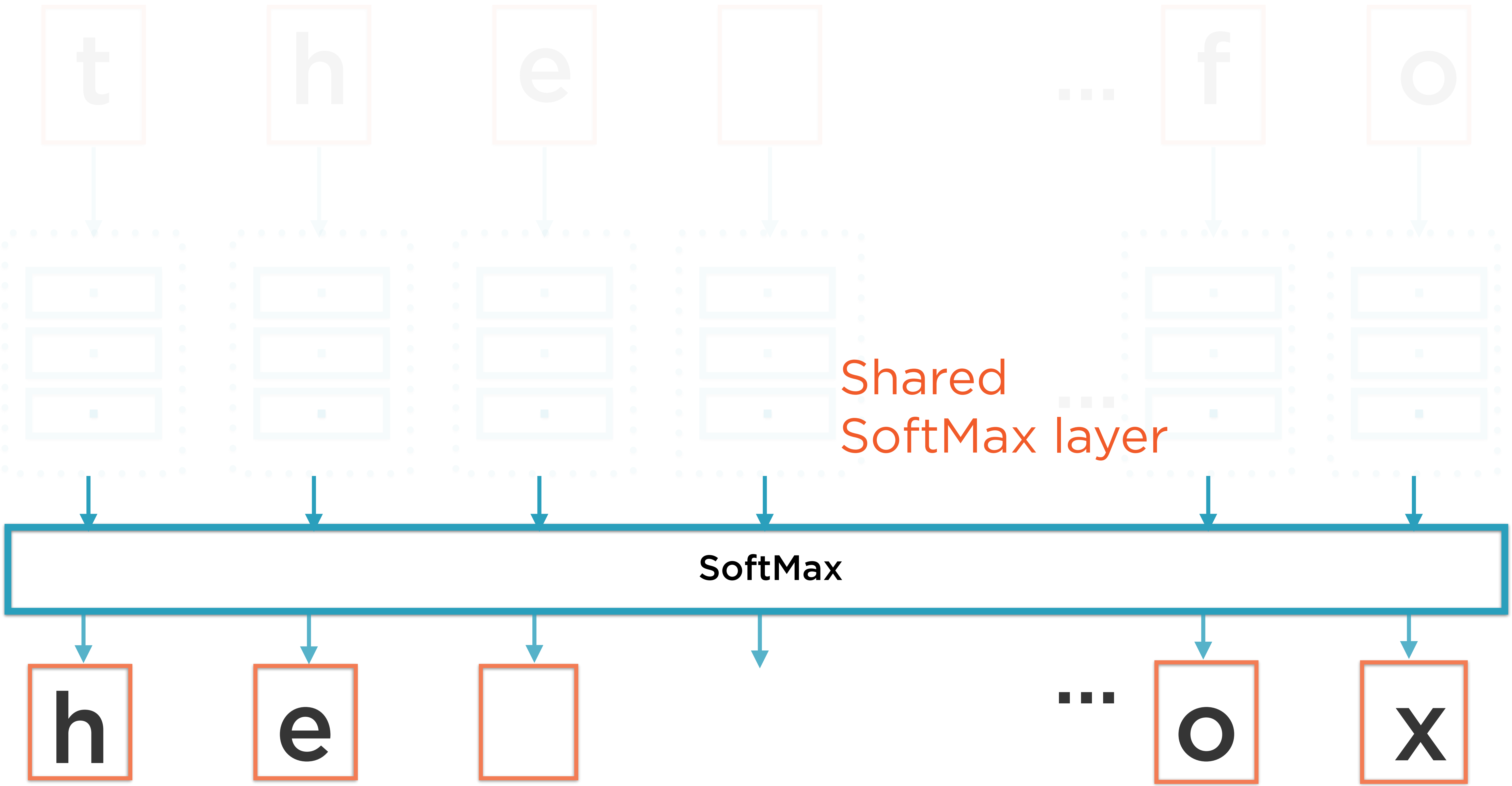
OCR: RNN Architecture



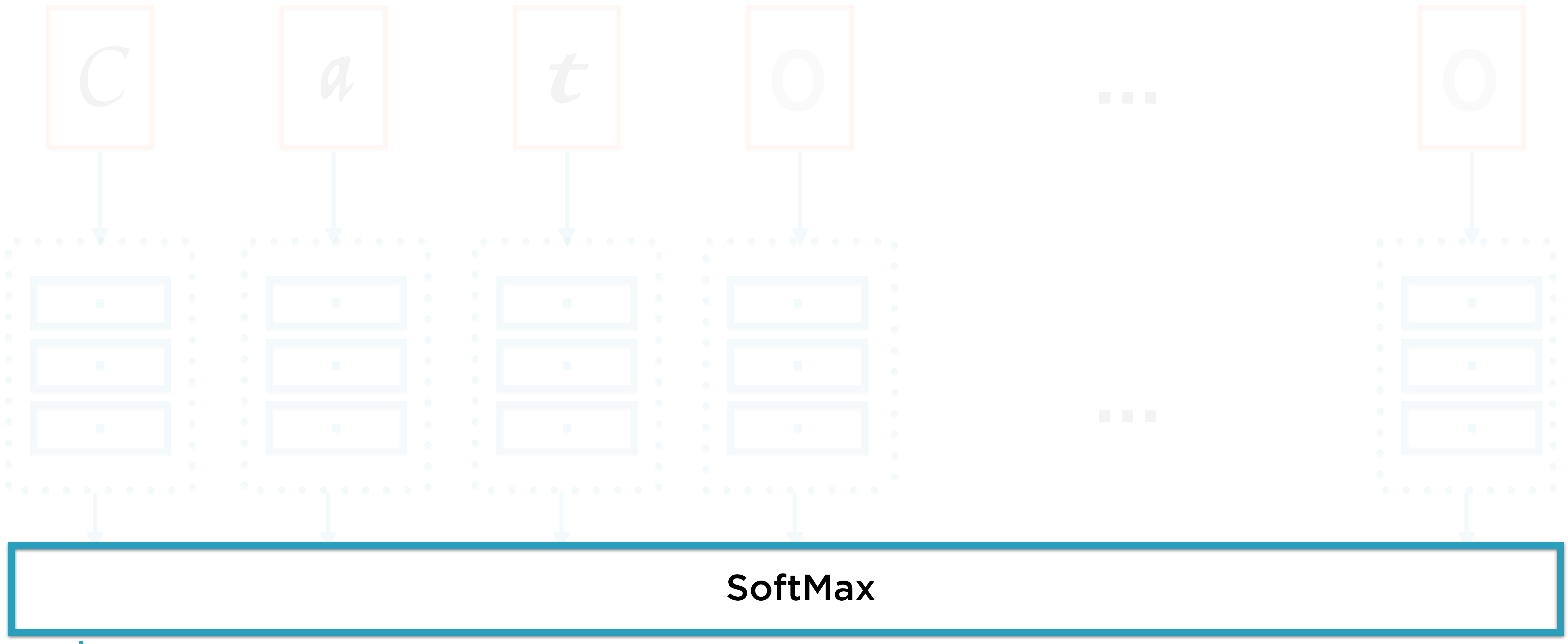
OCR: RNN Architecture



Text Prediction: RNN Architecture



OCR: RNN Architecture

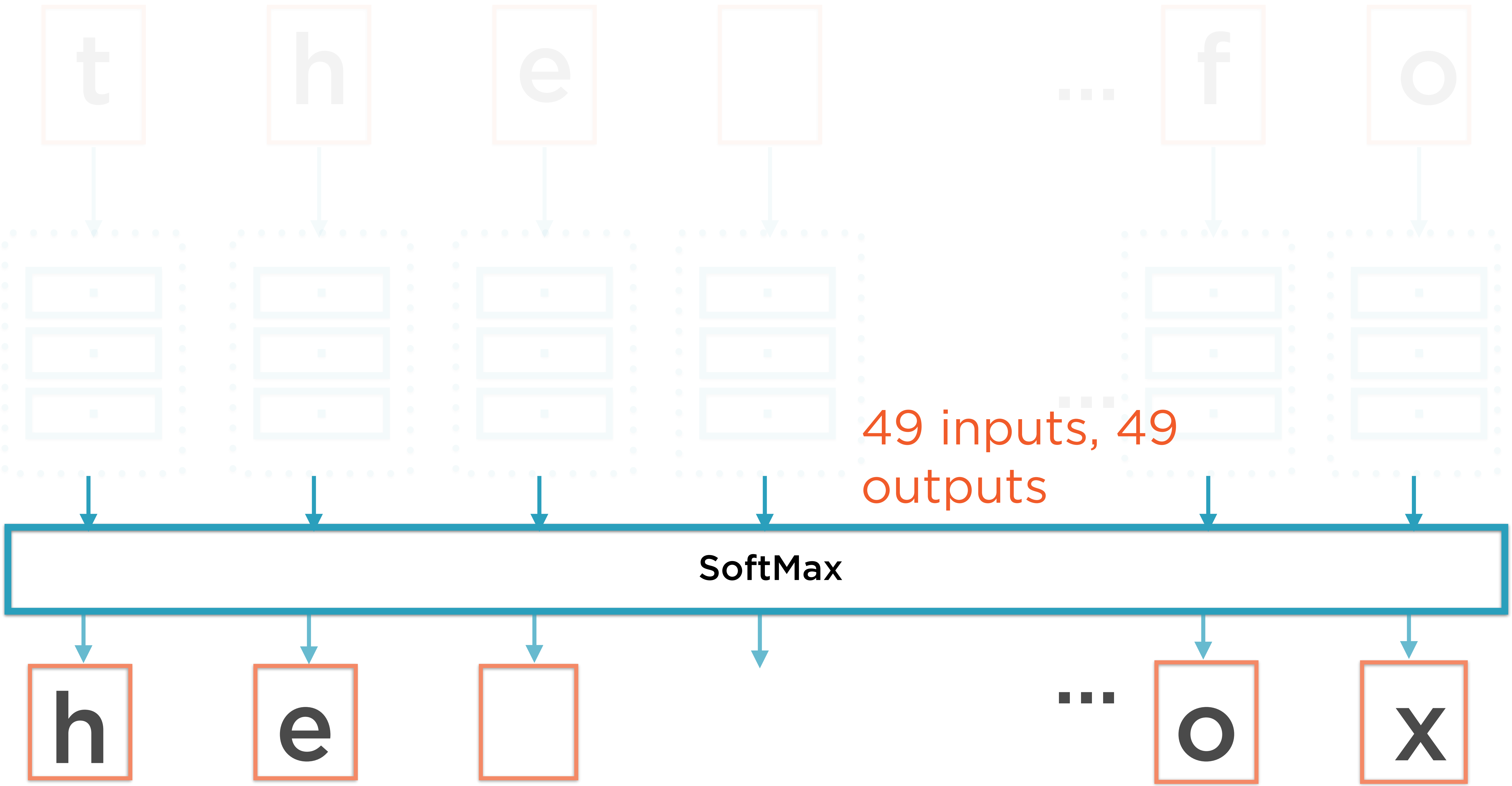


C

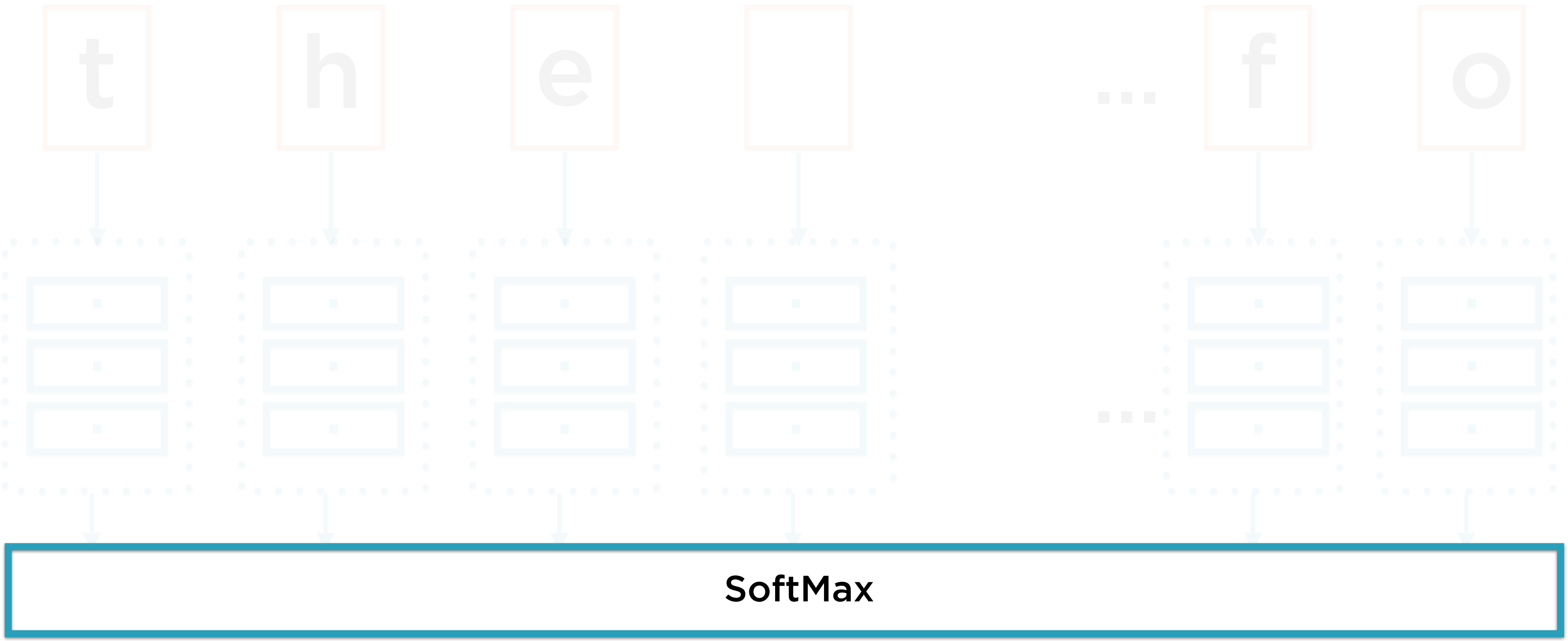
a	b	c	...	z
0.005	0.0	0.995		0

Each output: One-hot
26 probabilities for a-z

Text Prediction: RNN Architecture



Text Prediction: RNN Architecture



h

a	b	c		h	...
0.0	0.0	0.0		0.99	0

Each output: One-hot 83 probabilities for alphanumeric characters

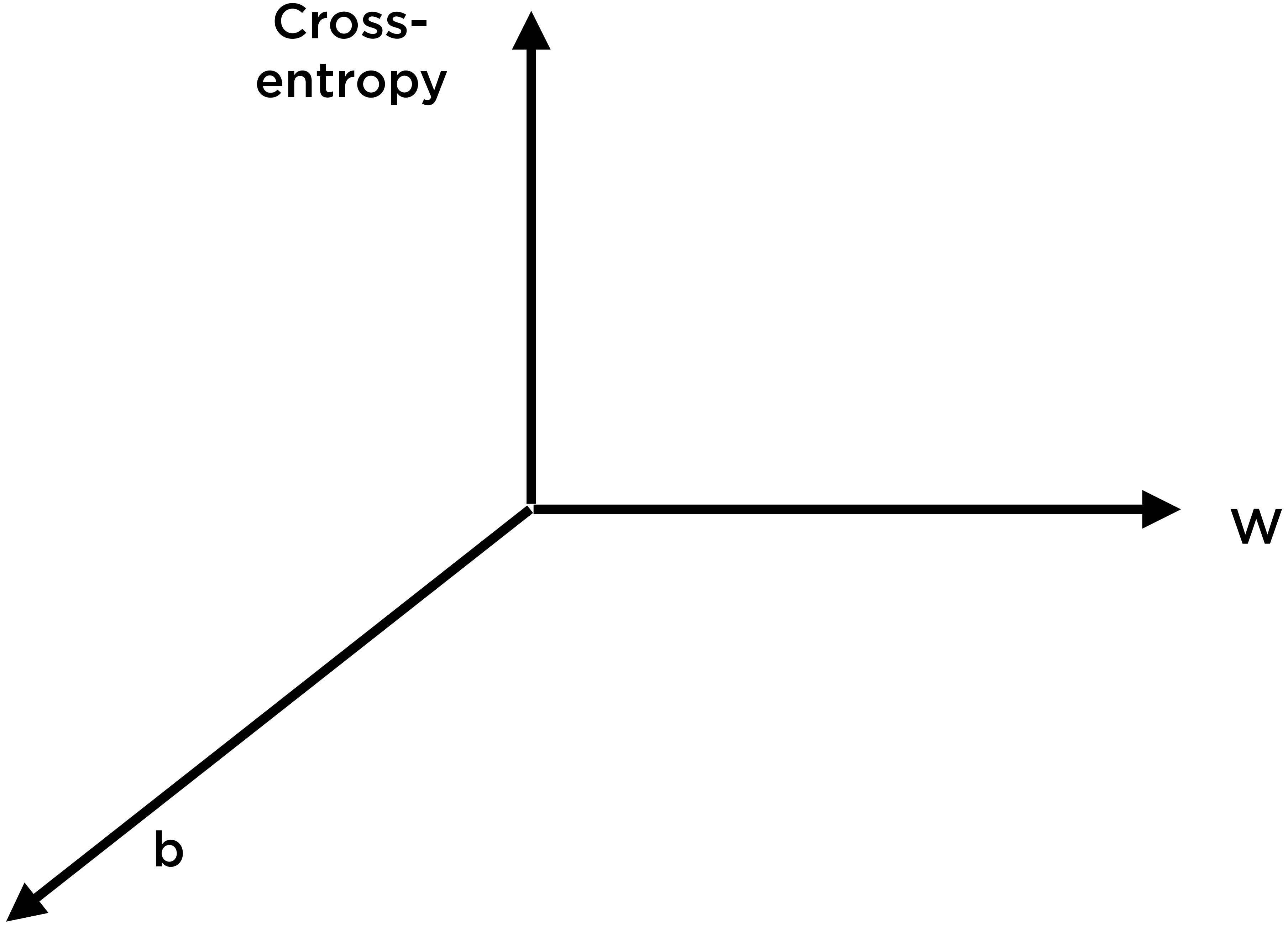
Perplexity

Contrasting Architectures

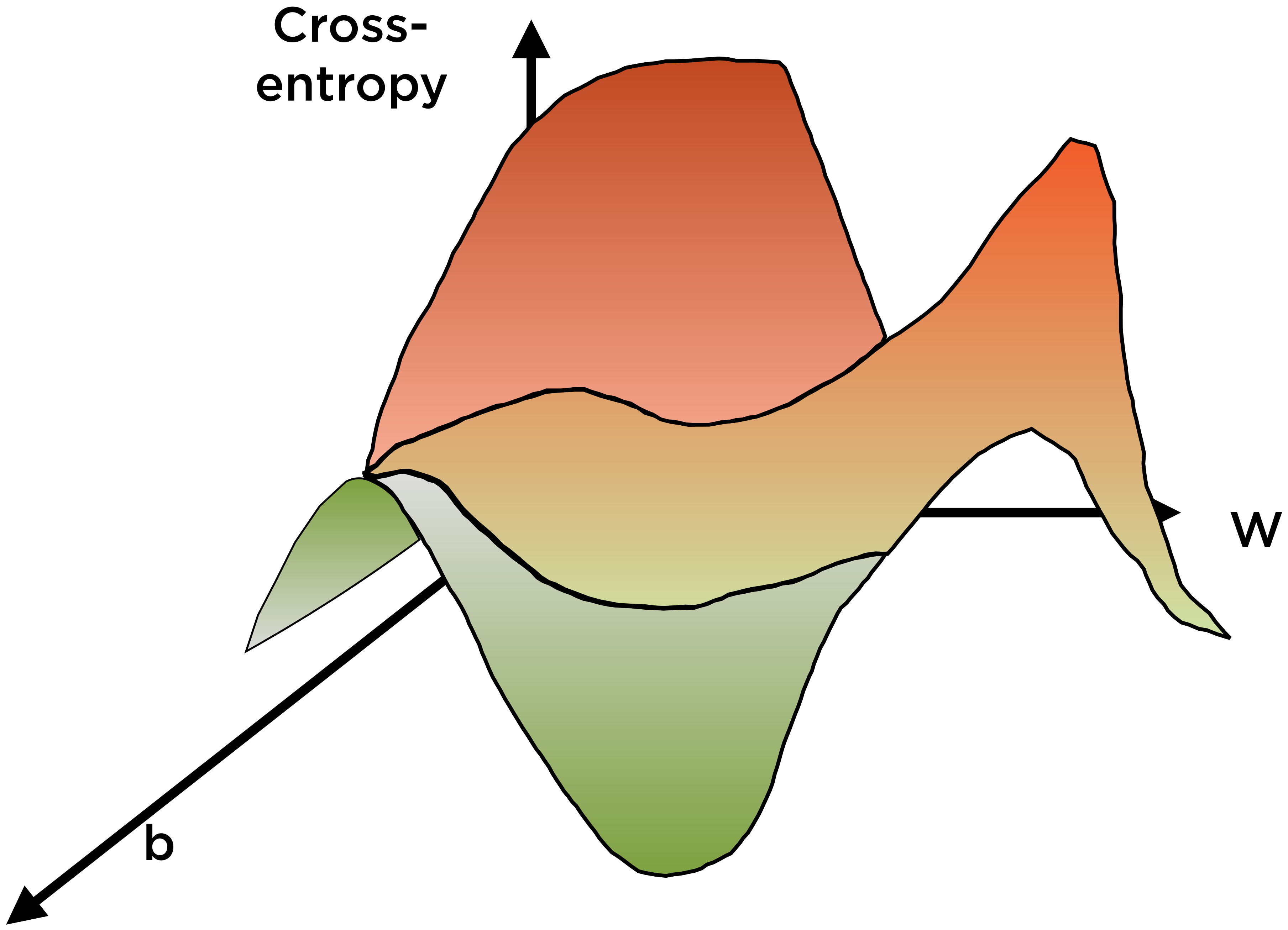
OCR Classification	Text Prediction
Cross-entropy as cost function	Cross-entropy as cost function
Accuracy as evaluation metric	Perplexity as evaluation metric

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

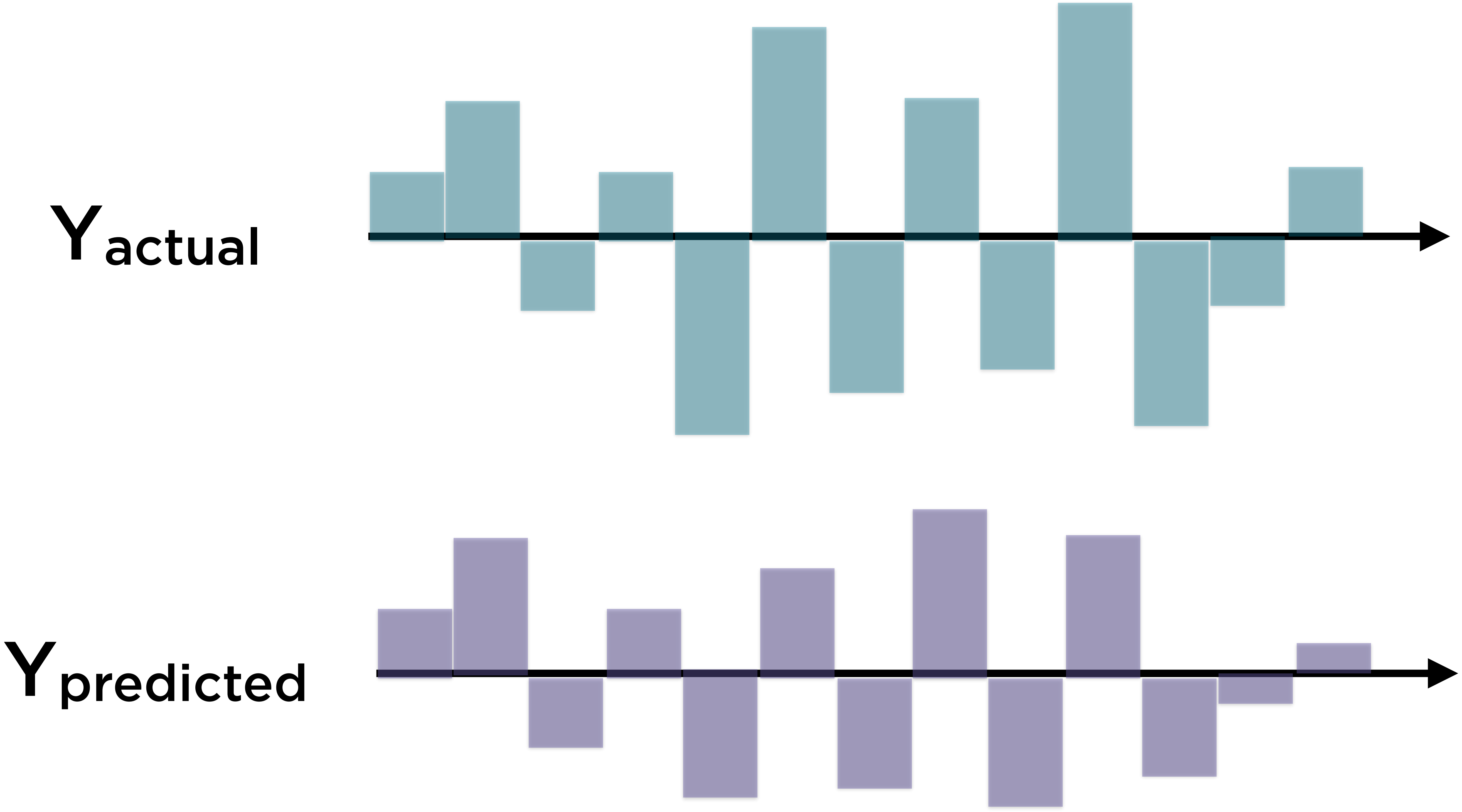
Minimizing Cross-entropy



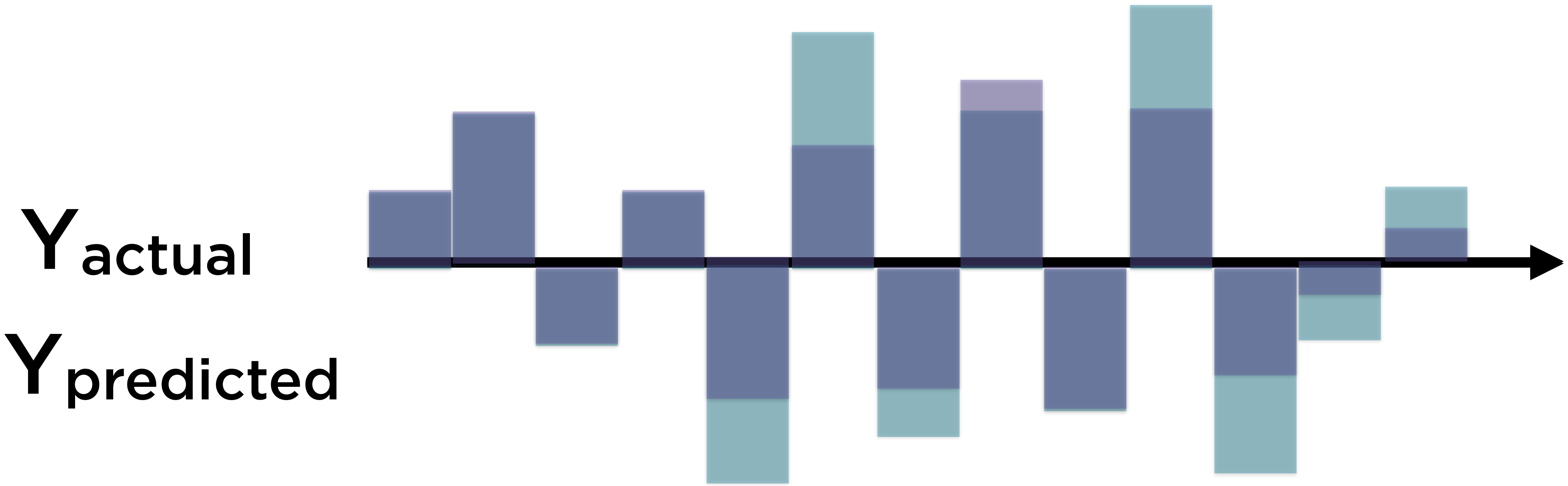
Minimizing Cross-entropy



Intuition: Low Cross-entropy

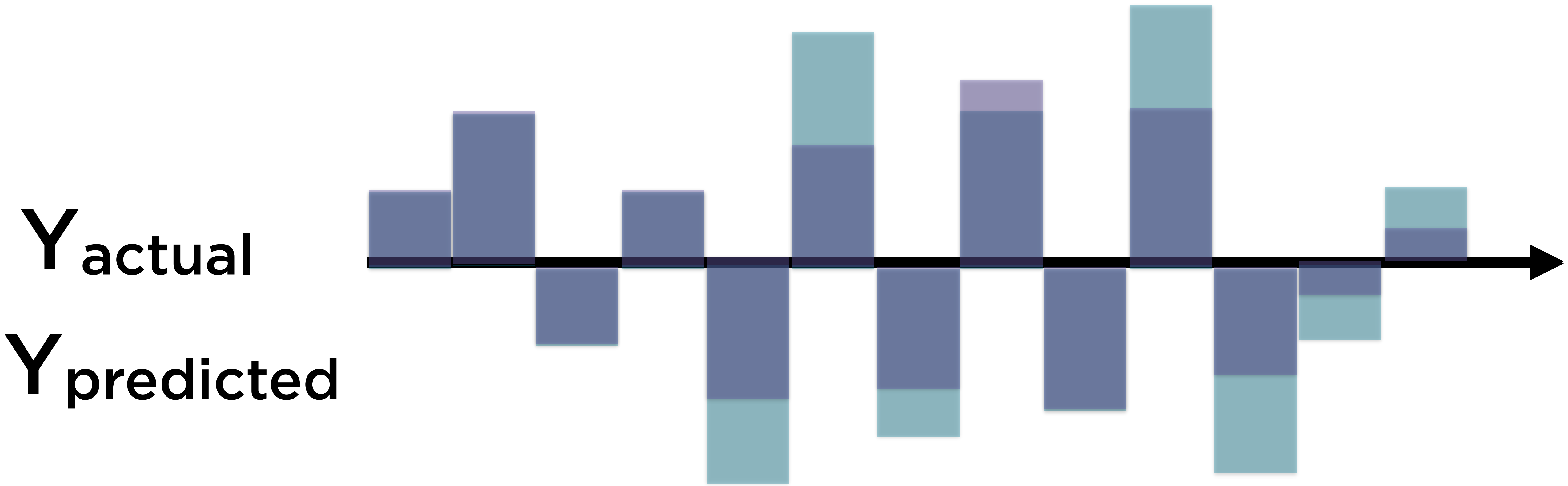


Intuition: Low Cross-entropy



The labels of the two series are in-synch

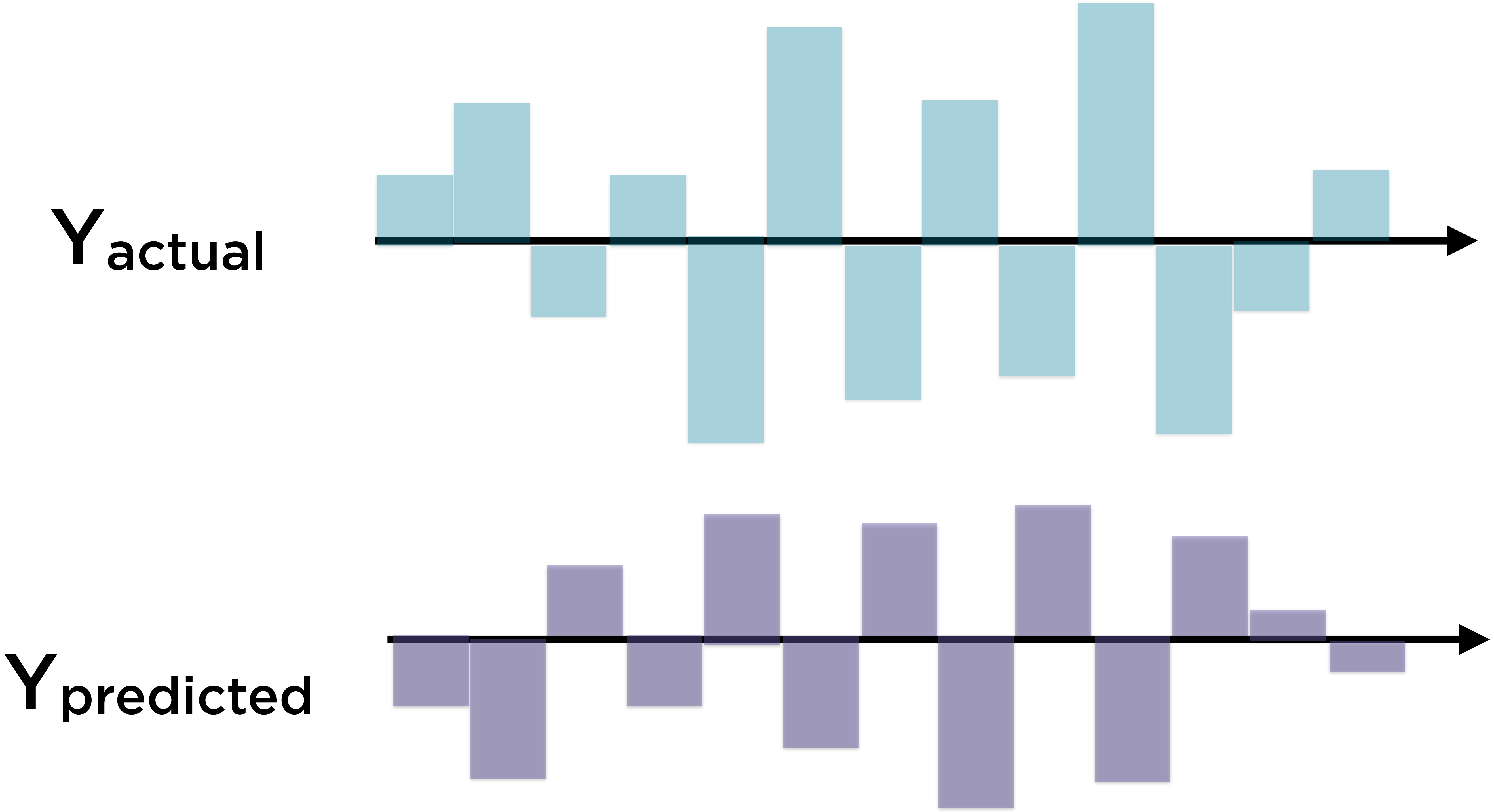
Intuition: Low Cross-entropy



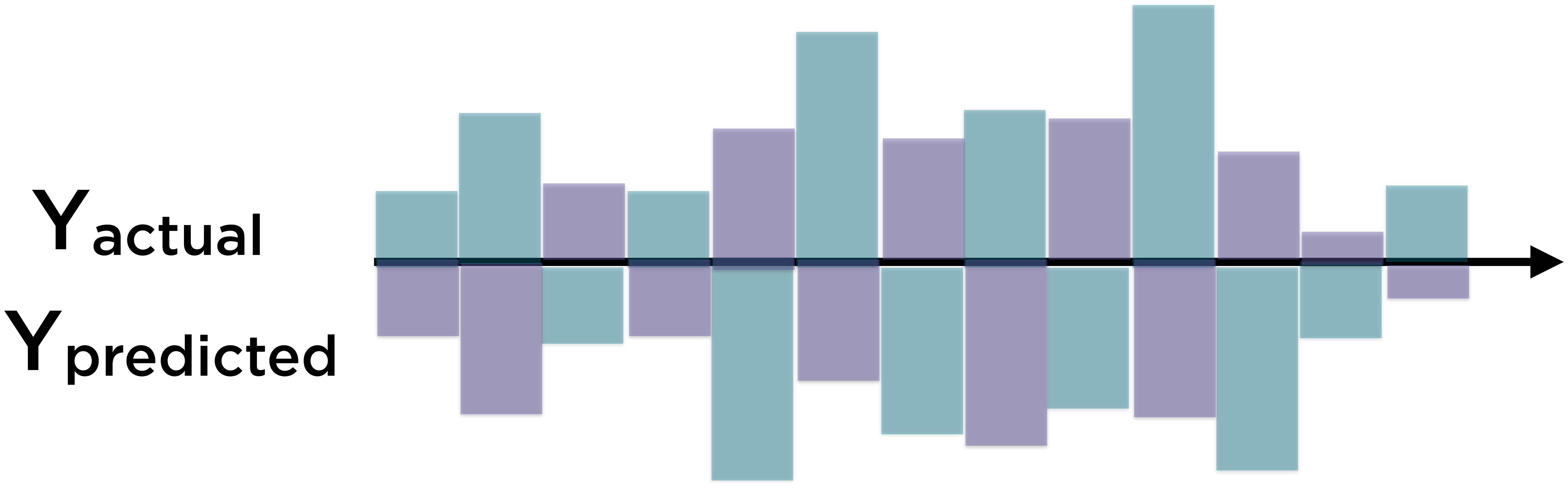
$\boxed{-Sum(\ P(Y_{actual}) * \log [\ P(Y_{predicted})]\ )}$ will be small

Cross-entropy

Intuition: High Cross-entropy

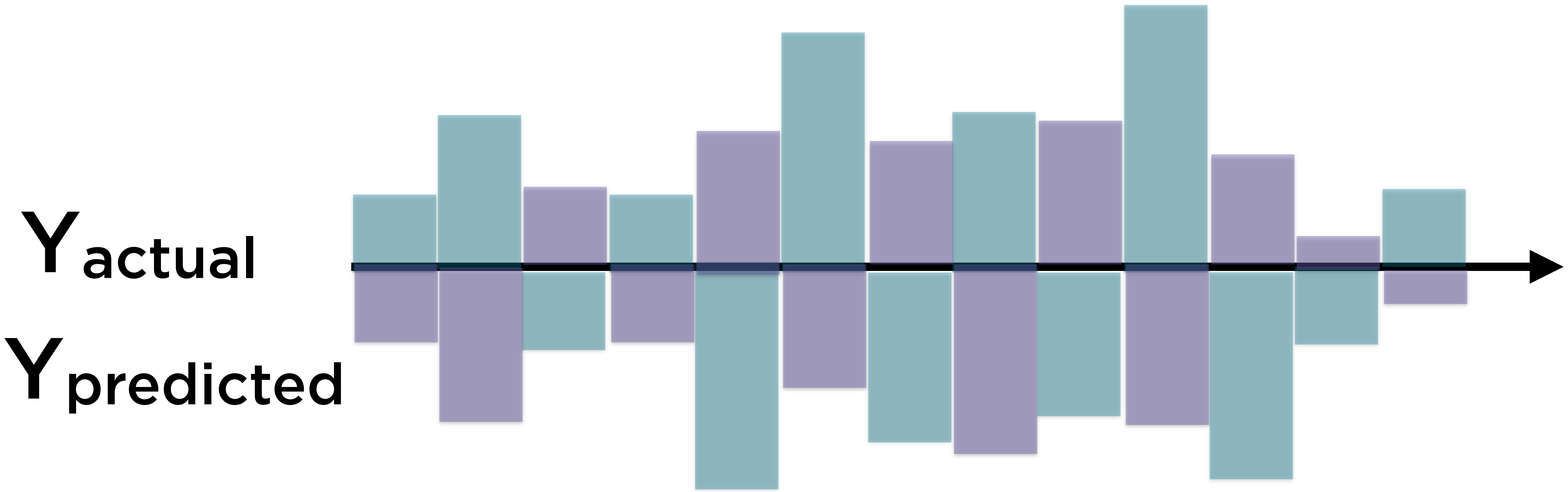


Intuition: High Cross-entropy



The labels of the two series are out-of-synch

Intuition: High Cross-entropy



-Sum(P(Y_{actual}) * log [P($Y_{predicted}$)]) will be large

Cross-entropy

Perplexity

In information theory, perplexity is a measurement of how well a probability model predicts a sample. A low perplexity indicates the model is good at predicting the sample

Perplexity

In information theory, perplexity is a measurement of how well a probability model predicts a sample. A low perplexity indicates the model is good at predicting the sample

Perplexity

In information theory, perplexity is a measurement of how well a probability model predicts a sample. A low perplexity indicates the model is good at predicting the sample

Training a Model

Fact	Explanation
Characters in our universe are drawn from distribution p	This distribution p is unknown
Training samples are drawn from probability distribution p	Use these to train a model
Model estimates some probability distribution q	q should be as close to p as possible
Training process minimises cross-entropy between p and q	Cross-entropy is a measure of distance between two distributions

Evaluating Prediction

Fact	Explanation
Draw test sample $x_1, x_2, x_3 \ldots x_N$ from corpus	The test sample follows distribution p
Use the model to predict what follows	Model will predict using distribution q
Calculate perplexity to evaluate how well the predictor did	Perplexity is just $2^{\text{cross-entropy}}$

Perplexity

$$\text{Cross-entropy} = -\sum \left(P(Y_{actual}) * \log \left[P(Y_{predicted}) \right] \right)$$

$$\text{Perplexity} = 2^{\text{cross-entropy}}$$

A perfect model has cross-entropy of 0, and perplexity of 1

Perplexity

$$\text{Cross-entropy} = -\sum \left(P(Y_{actual}) * \log\left[P(Y_{predicted}) \right] \right)$$

$$\text{Perplexity} = 2^{\text{cross-entropy}}$$

Perplexity captures how many different options the model has to choose between

The more choices - the more perplexed (confused) the model is

A perfect model has cross-entropy of 0, and perplexity of 1

Summary

Text prediction is one of several classic language modeling problems

Multi-RNNs are a specific type of RNN that work well in language modeling

A key technique is smartly re-initialising state of the RNN during prediction

An evaluation metric called perplexity is used to assess predictive performance