Implementing RNNs for Character Prediction Used to Generate Text

Janani Ravi

CO-FOUNDER, LOONYCORN

www.loonycorn.com

Overview

Implement an RNN which trains on abstracts of technical papers

Use a multi-RNN cell to store additional state

Generate text one character at a time

Re-initialize state of the RNN during prediction to improve output

Use perplexity as an evaluation metric

Demo

Train an RNN on technical papers from https://arxiv.org/

- Use multi-RNN cells to store additional 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

Slide the window

50

The quick brown fox jumps over the lazy dog

Rinse-and-repeat

Sliding Window in Training

Slide the window

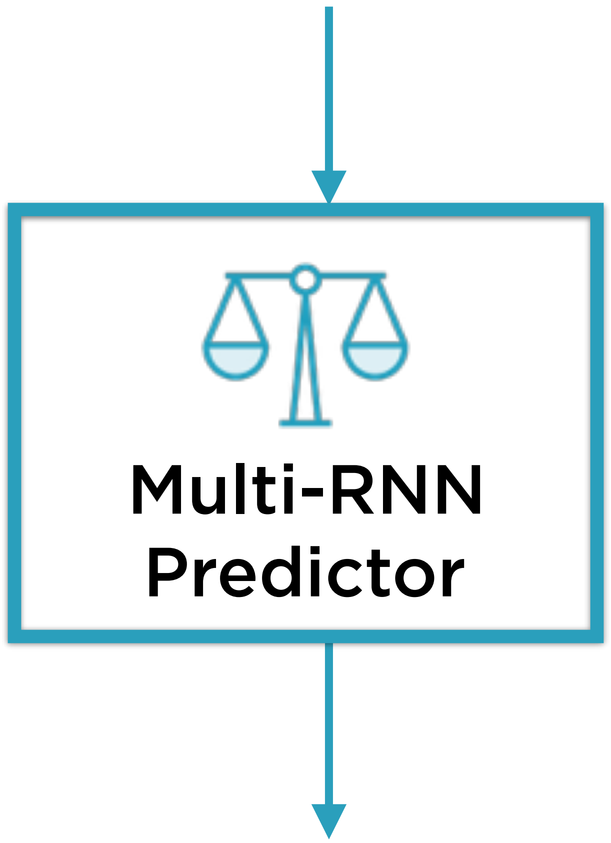
50

The quick brown fox jumps over the lazy dog

Rinse-and-repeat

Training Phase

The quick brown fox jumps



he quick brown fox jumps o

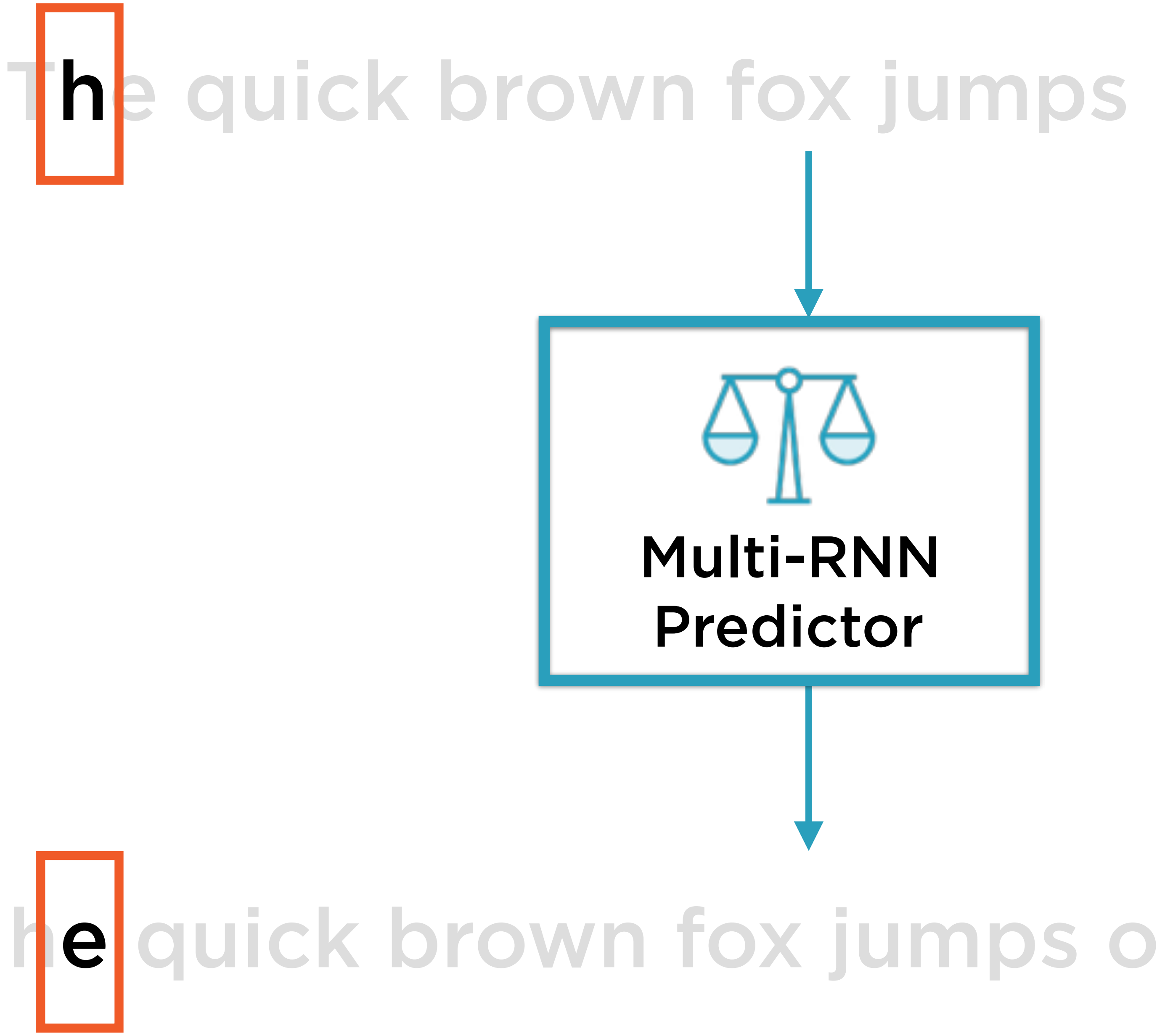
Training Phase

The quick brown fox jumps

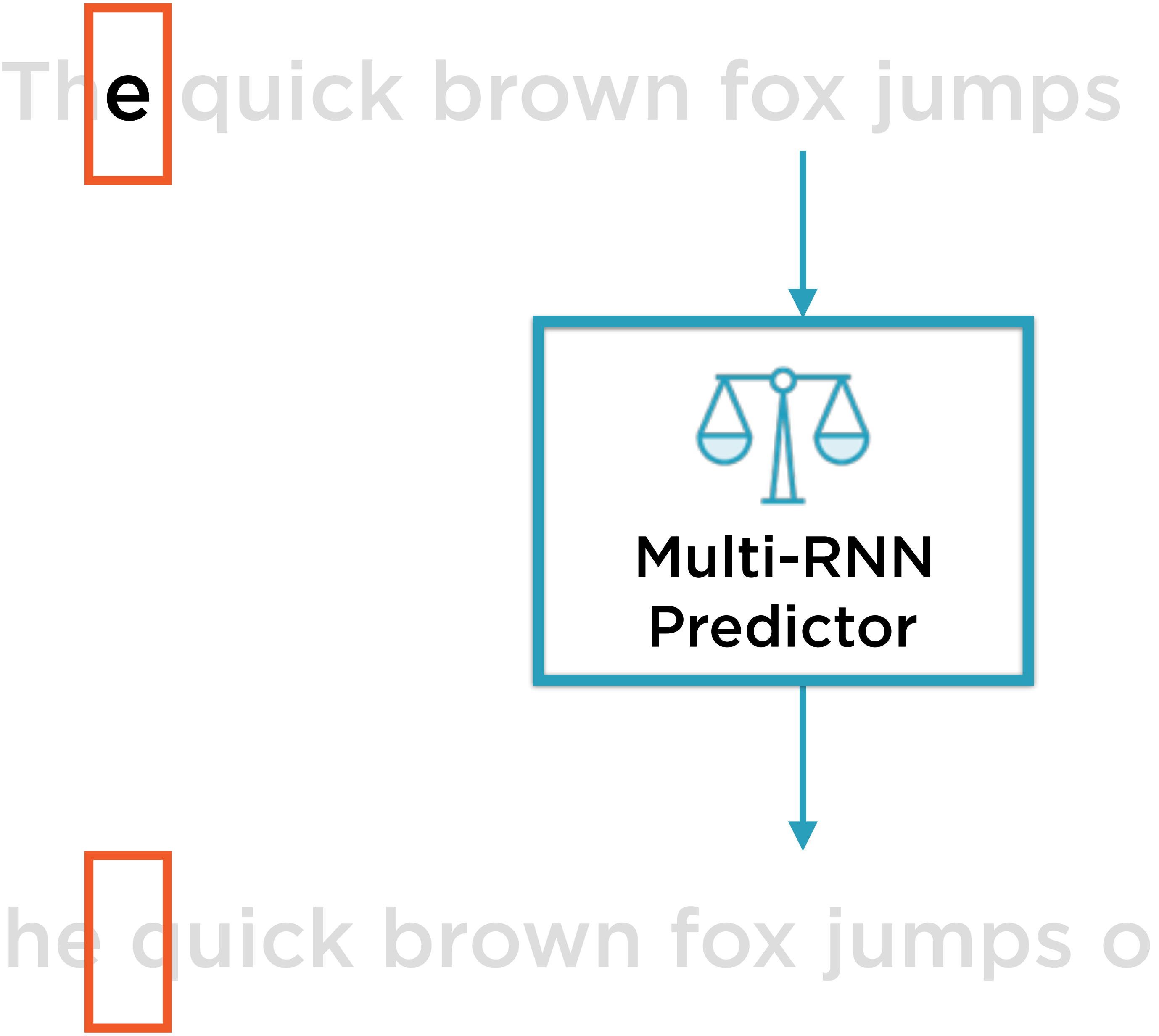
Multi-RNN
Predictor

he quick brown fox jumps o

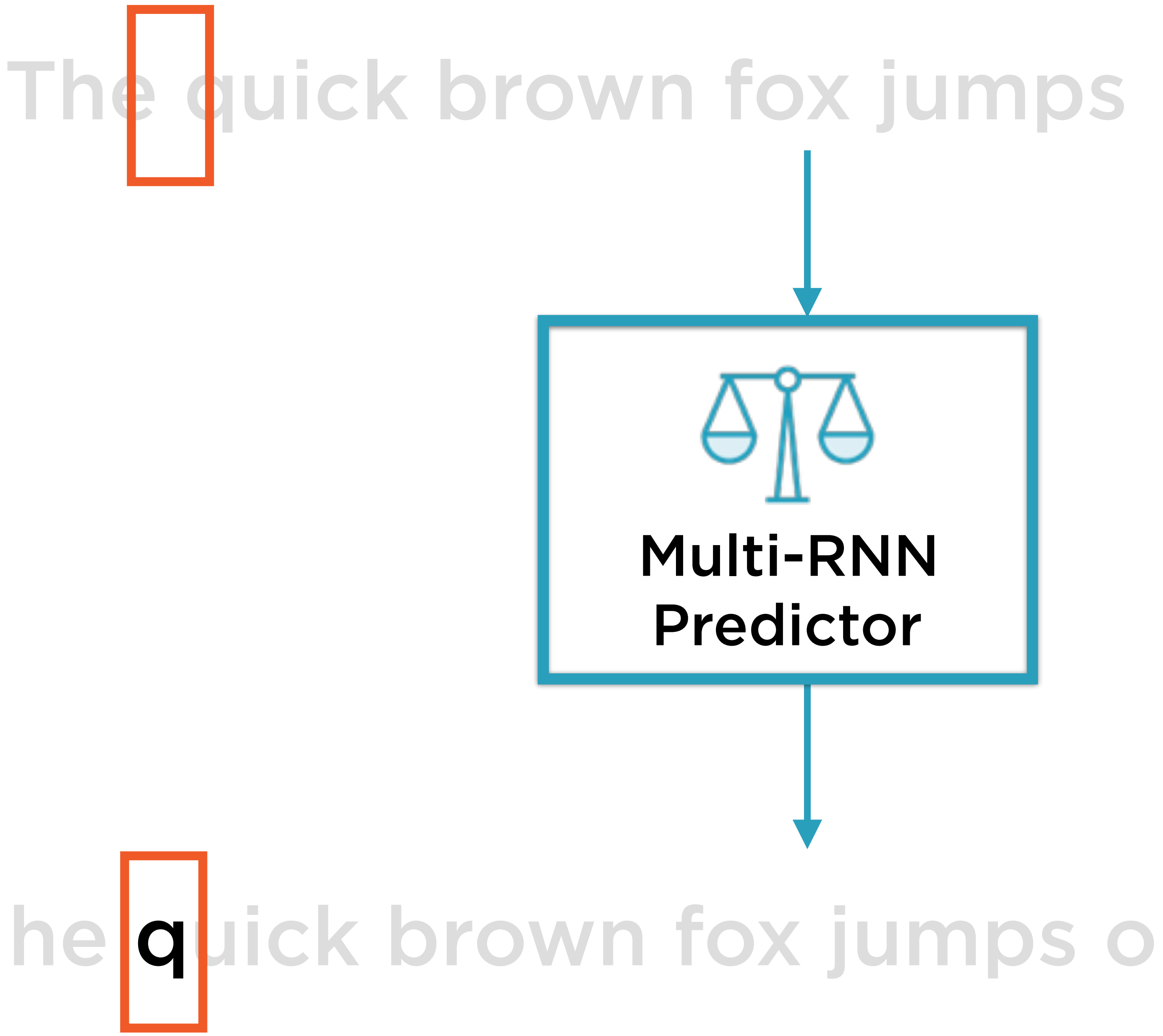
Training Phase



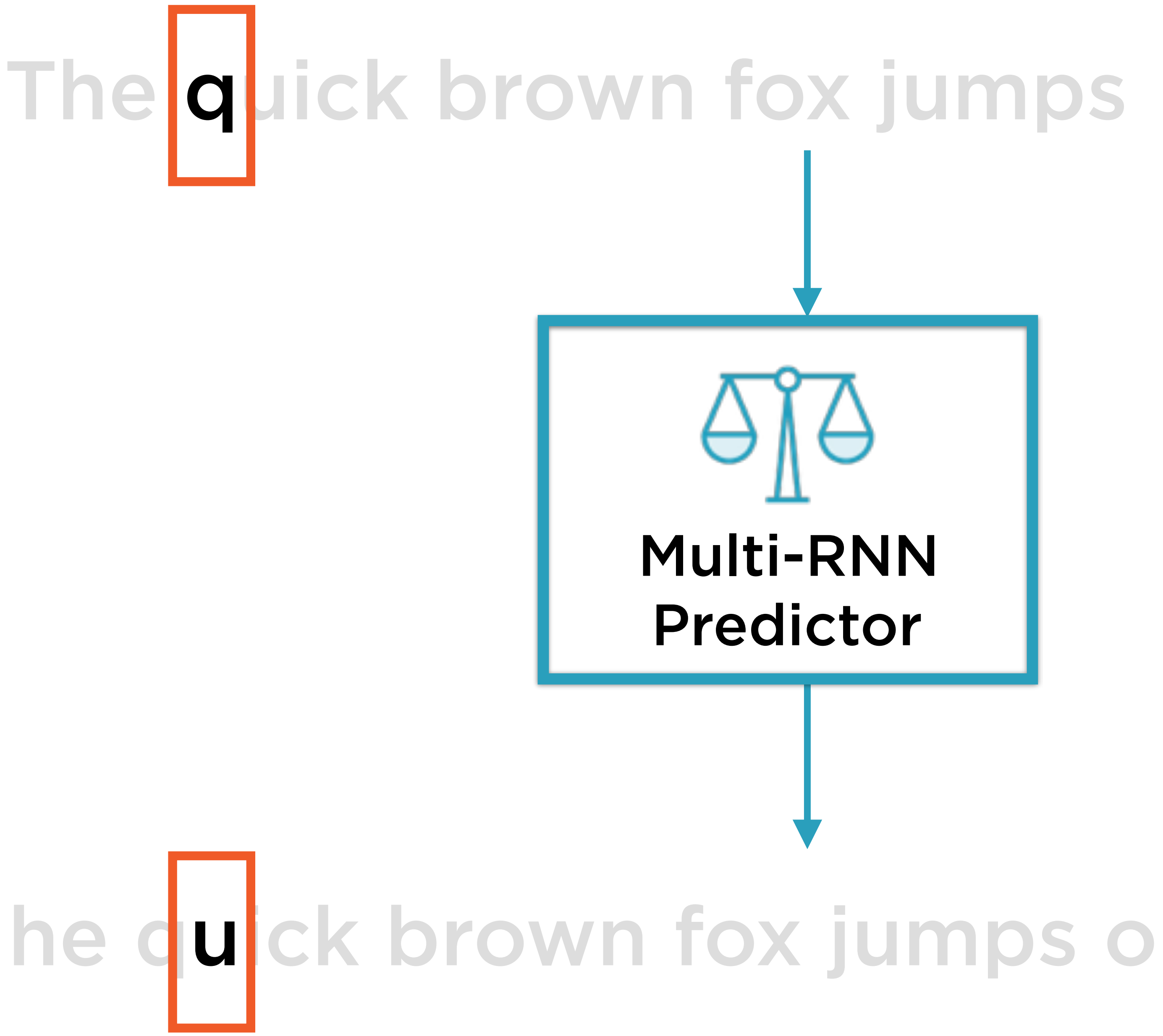
Training Phase



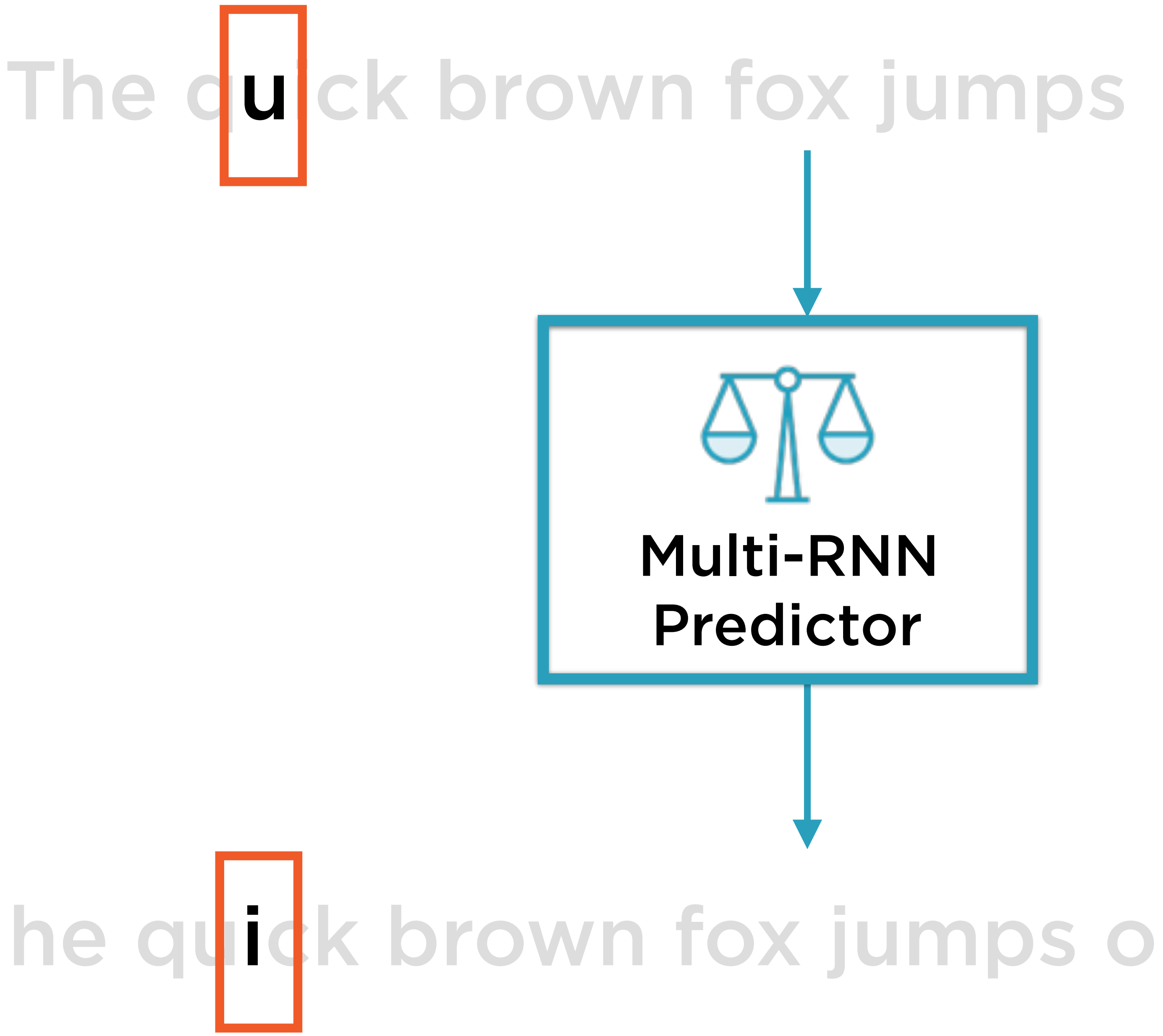
Training Phase



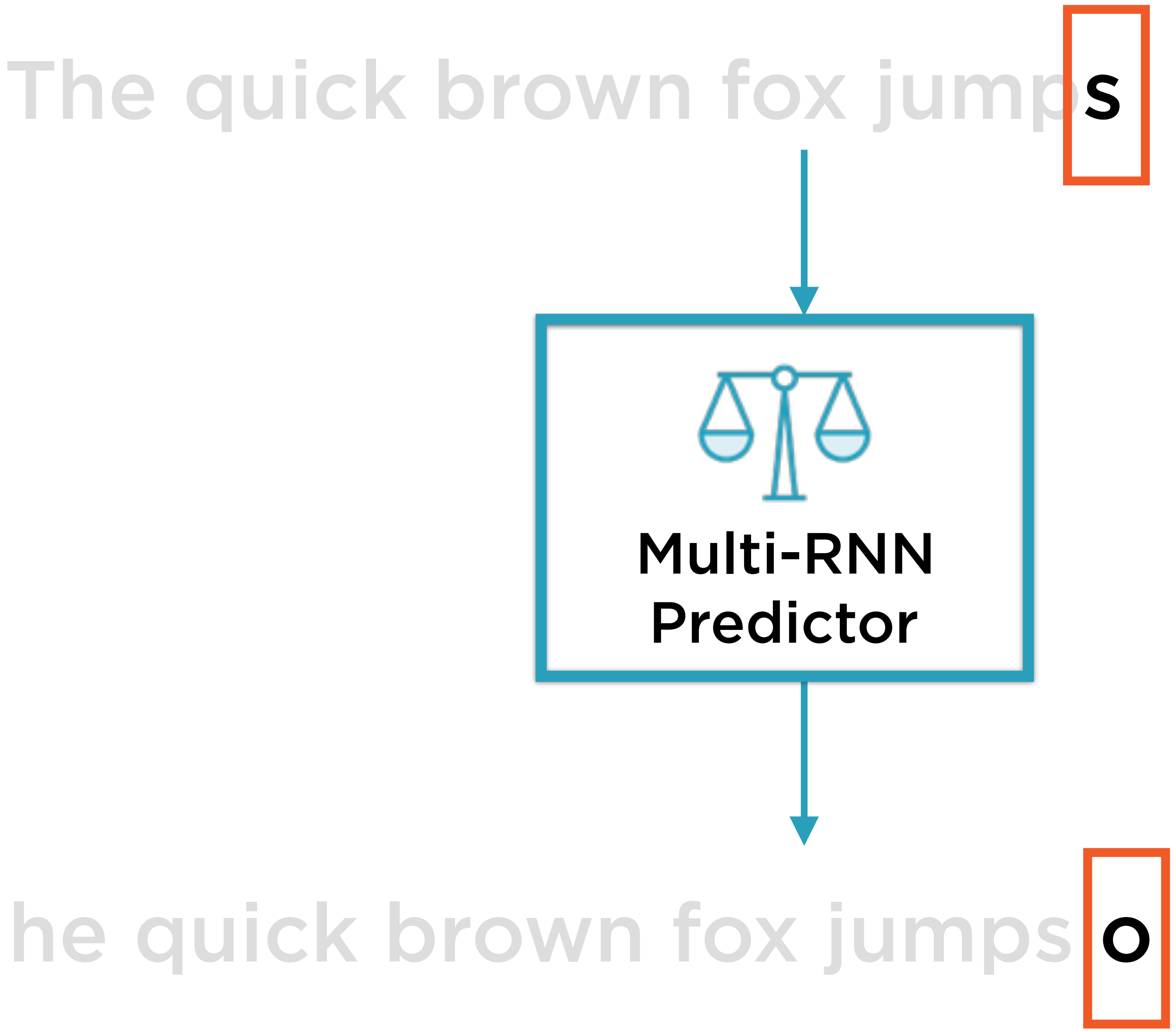
Training Phase

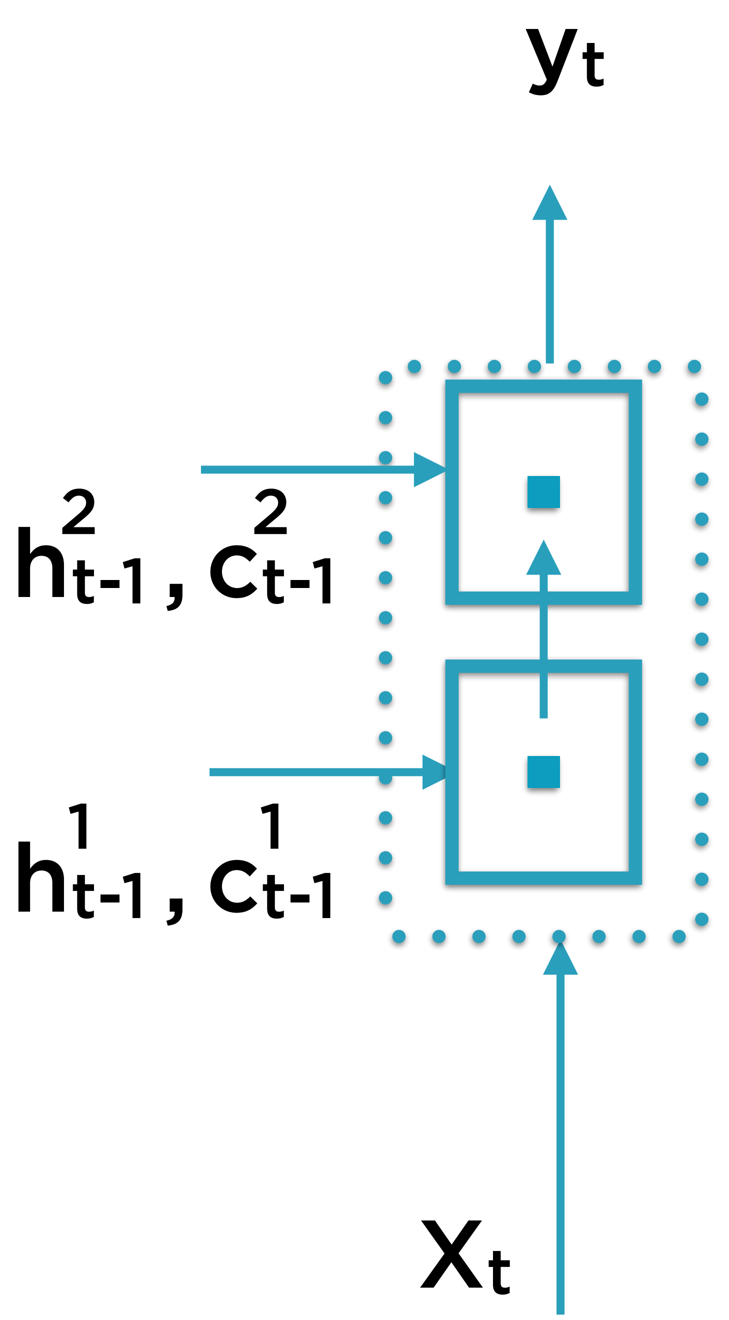


Training Phase



Training Phase



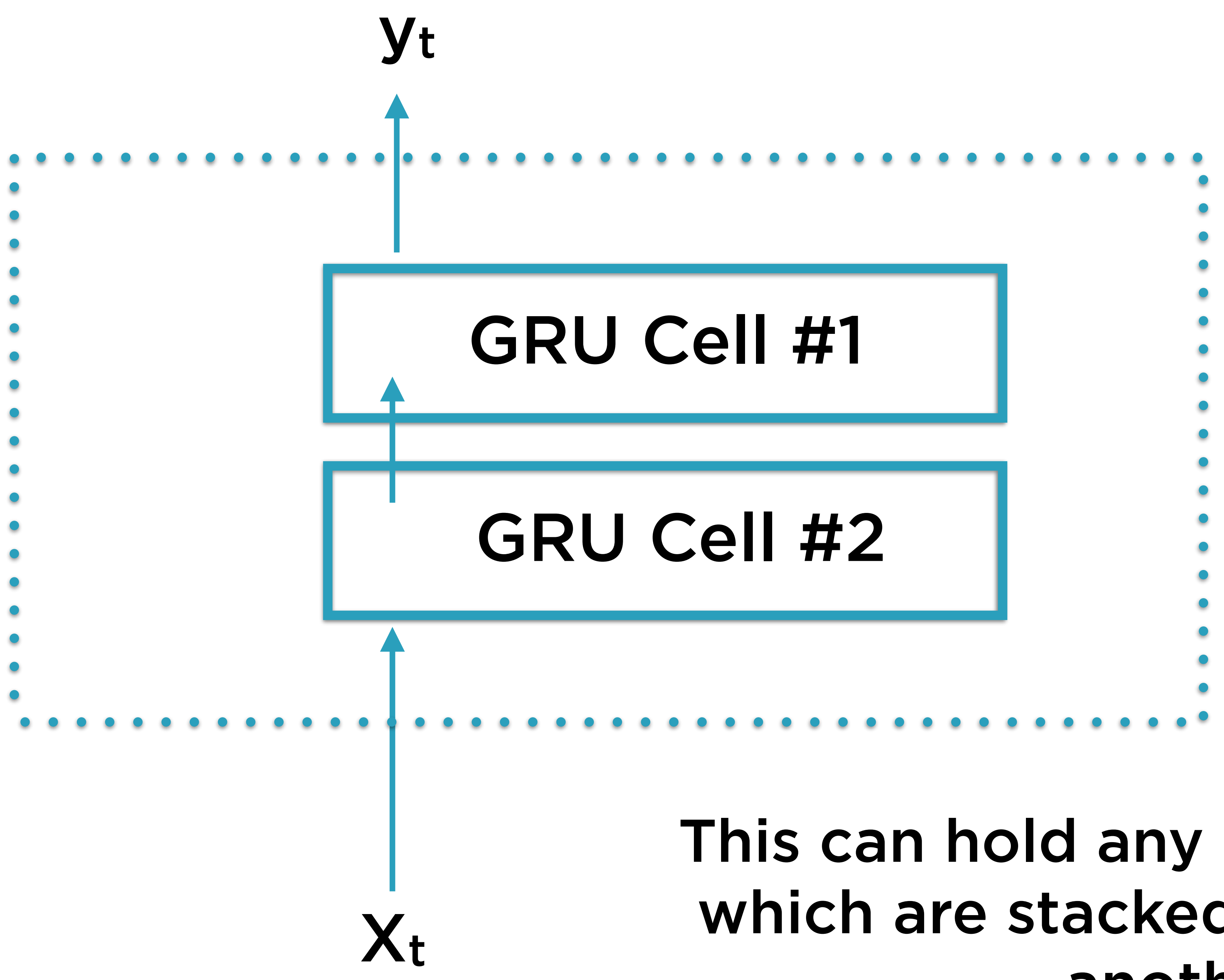


Multi-RNN Cell

Stack multiple RNN cells into “combined” RNN cell

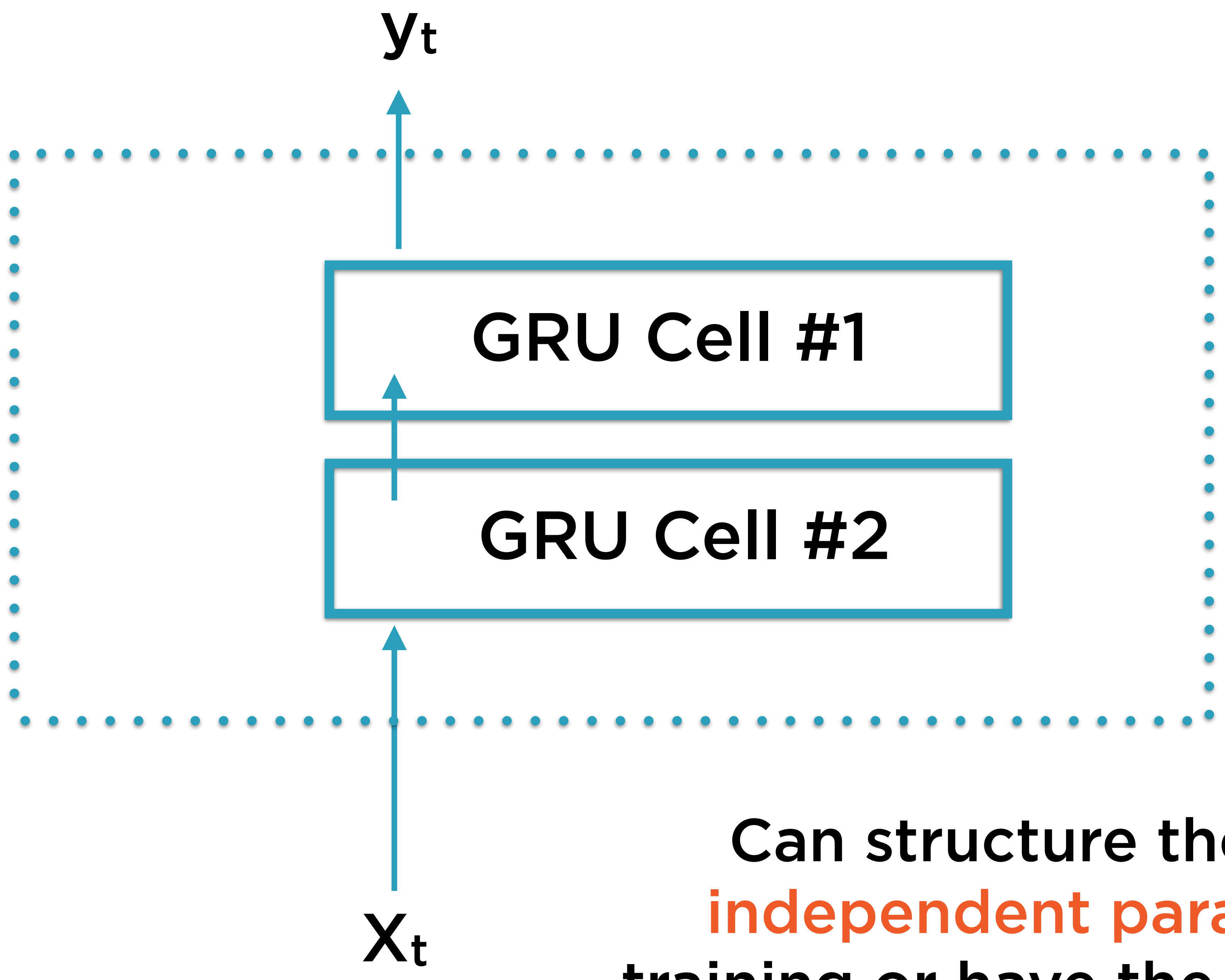
In our example, use 2 GRU cells inside each multi-RNN cell

Multi-RNN Cell



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

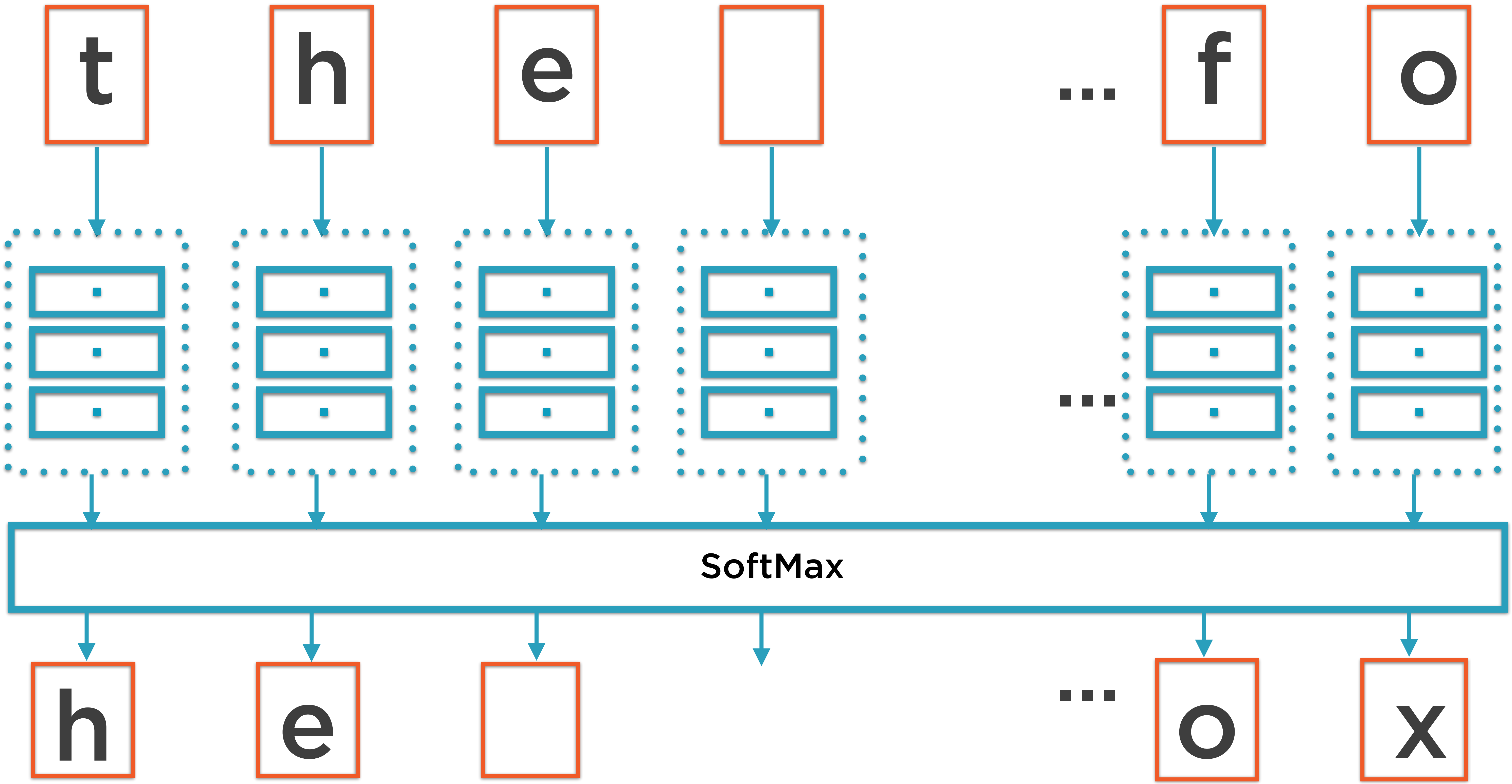
Multi-RNN Cell



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

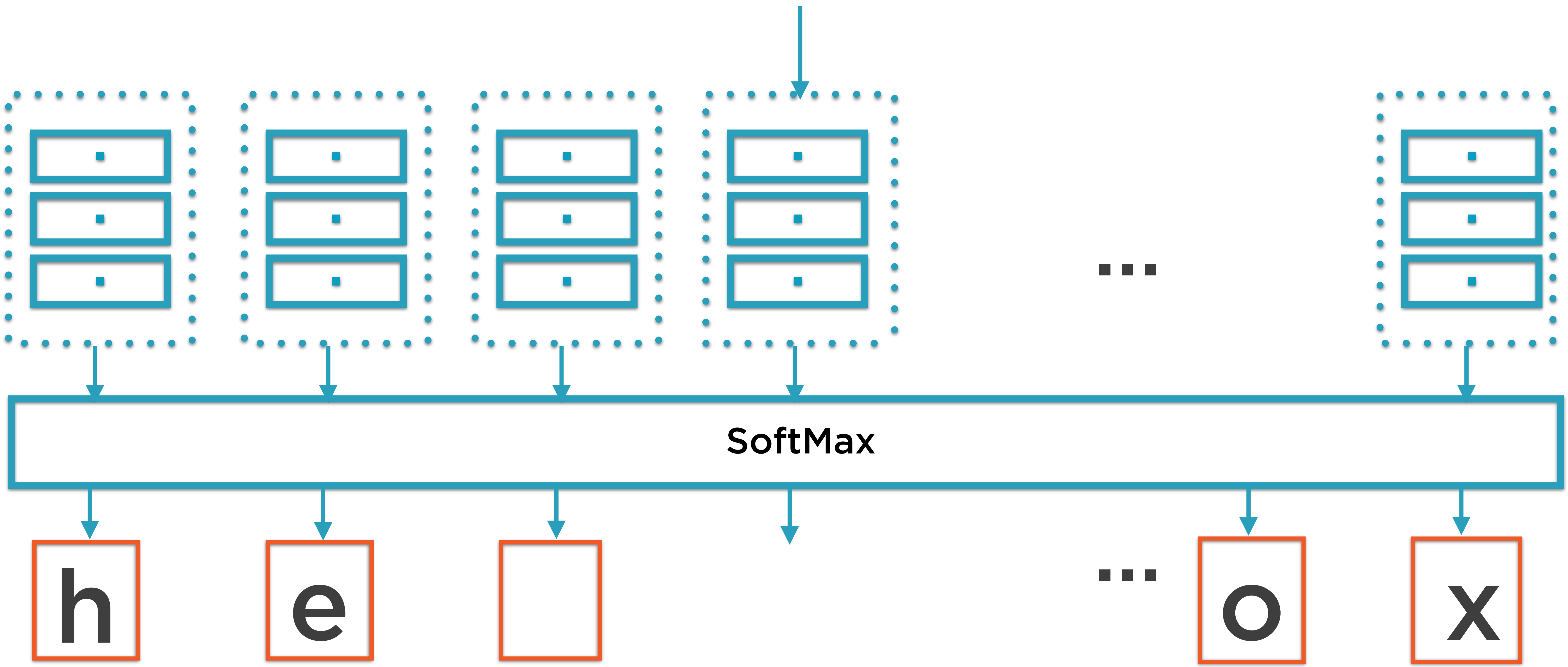
Multi-RNN cells allow you to wrap multiple cells allowing them to **look and behave like a single cell**

Text Prediction: RNN Architecture

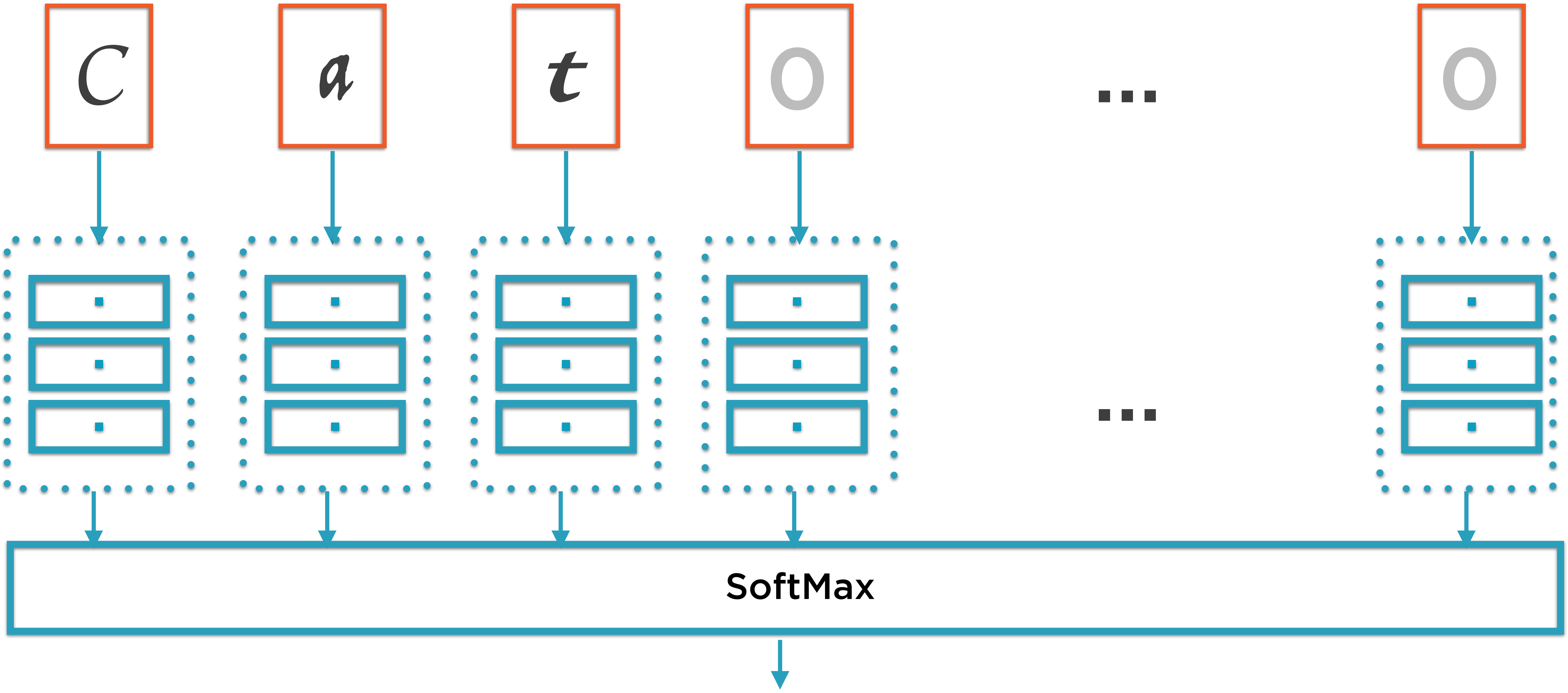


Text Prediction: Input Tensor for Training

[batch_size, 49, 83]

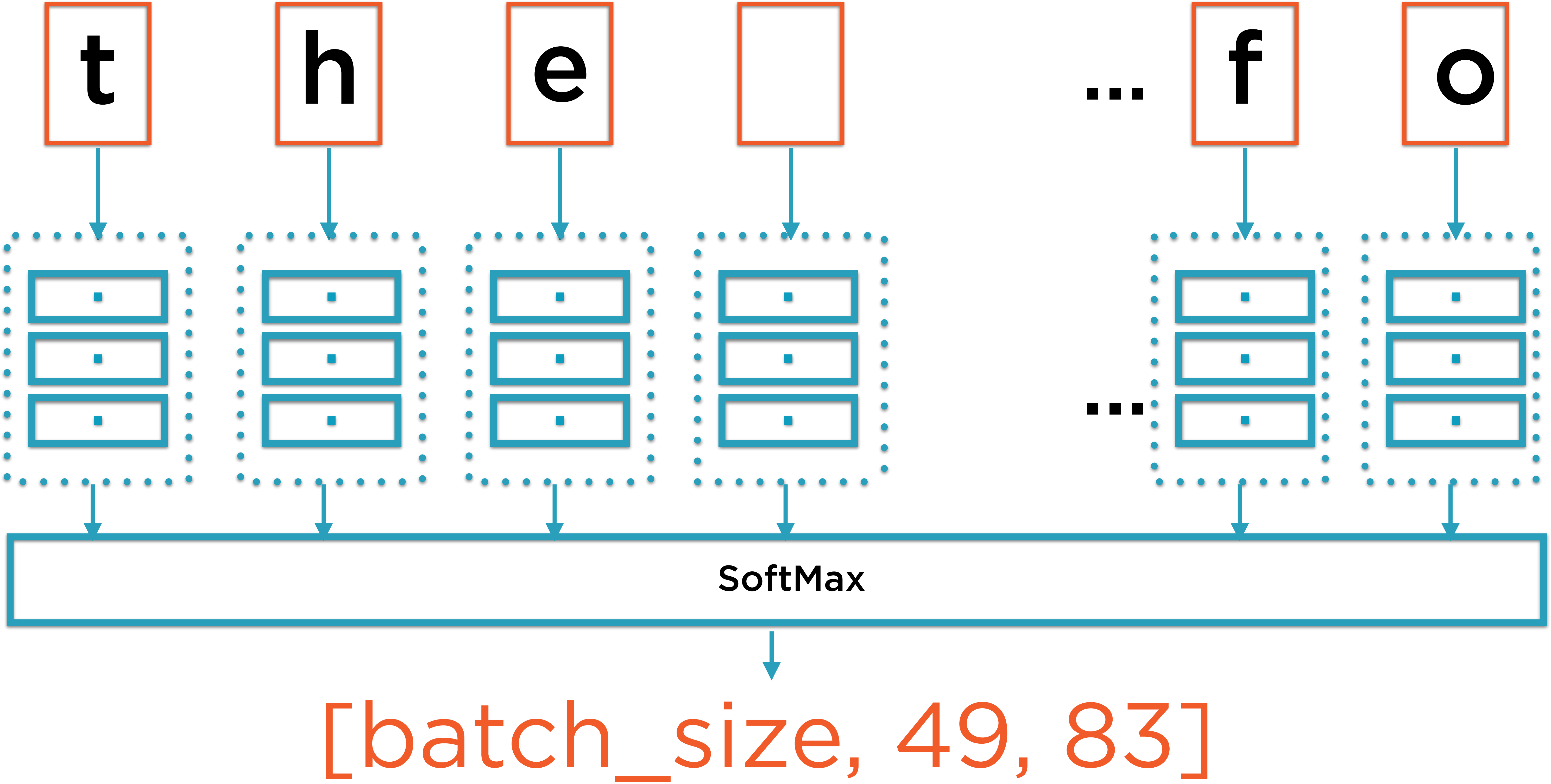


OCR: Output Tensor for Predicted Values

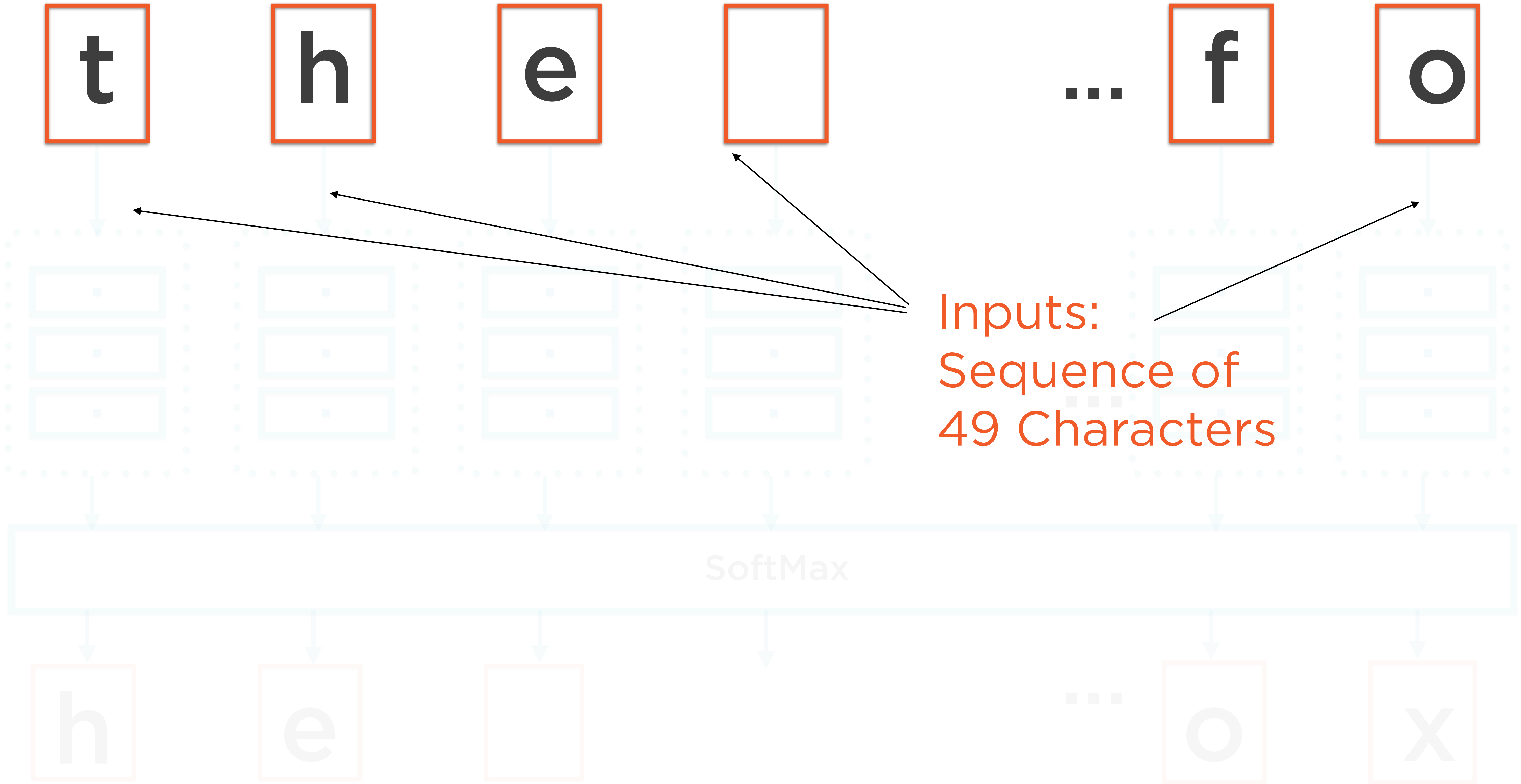


[batch_size, 14, 26]

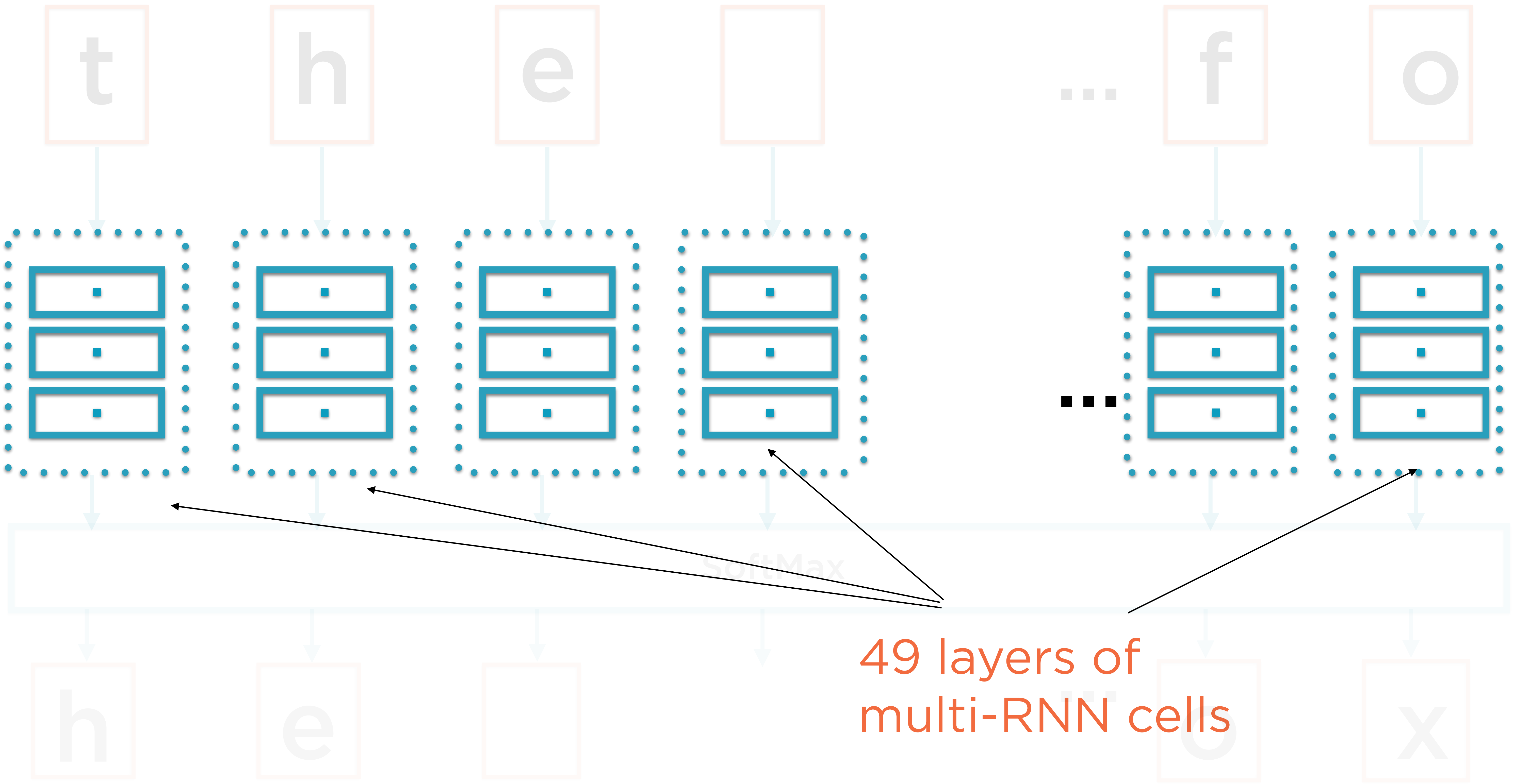
Text Prediction: RNN Architecture



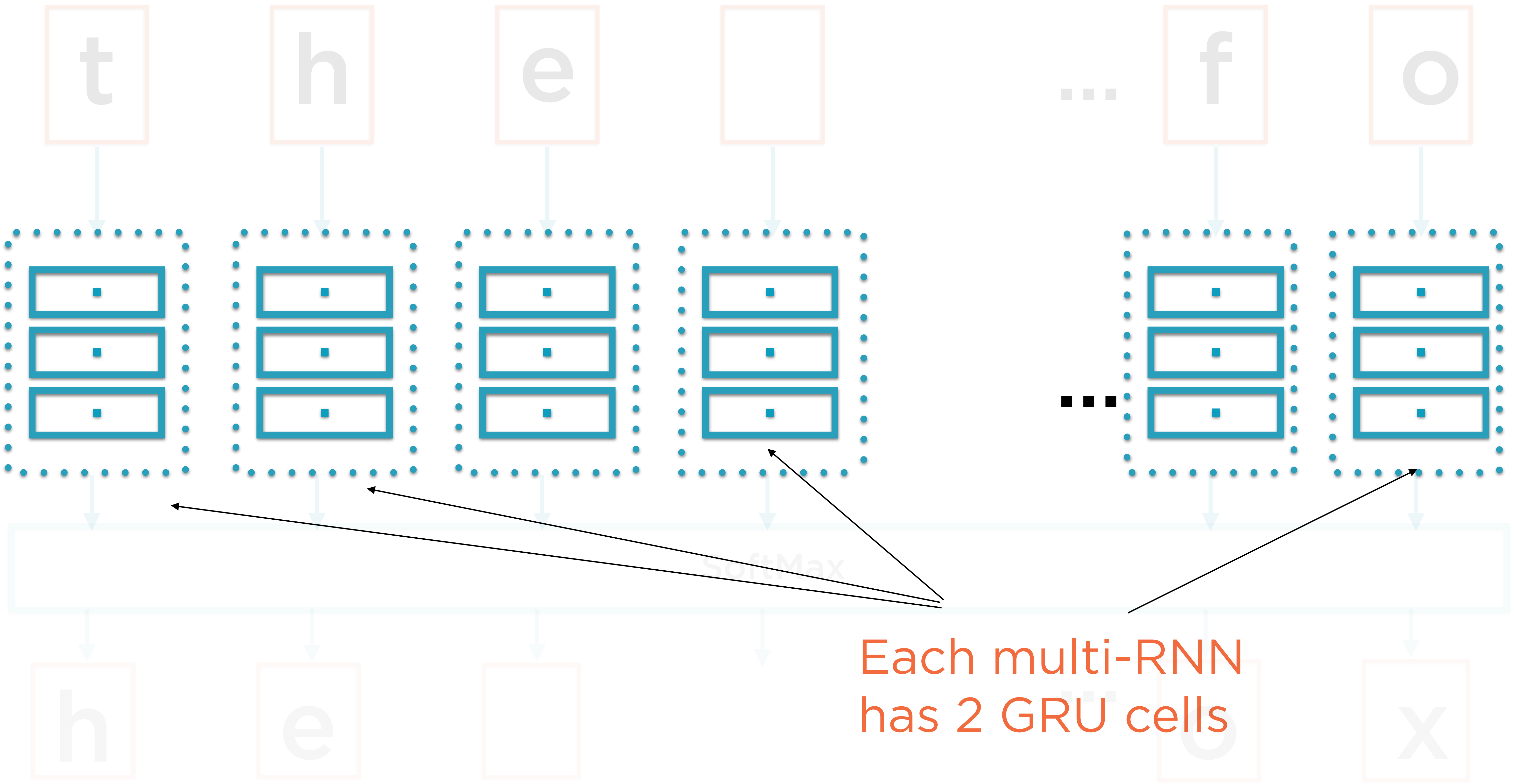
Text Prediction: RNN Architecture



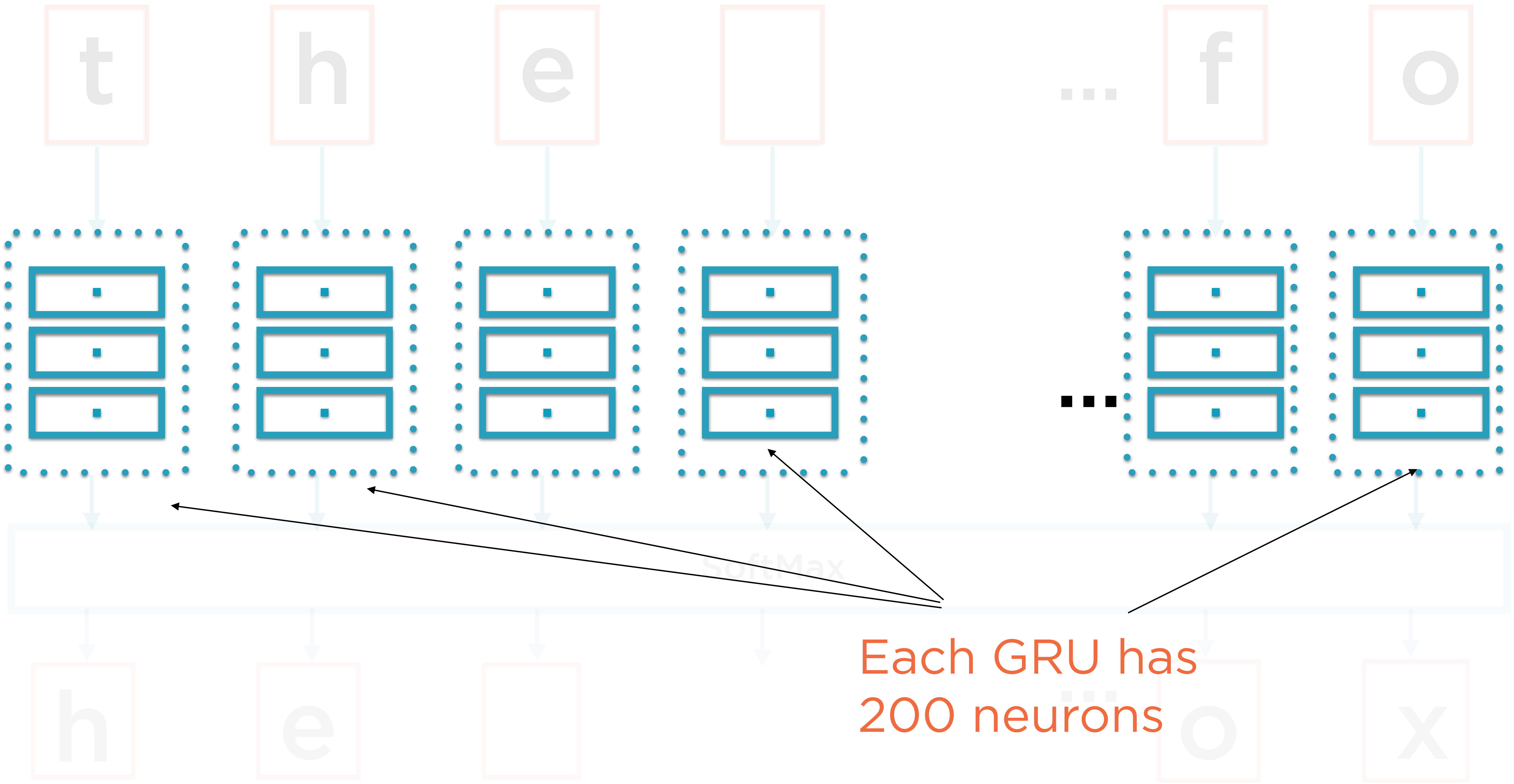
Text Prediction: RNN Architecture



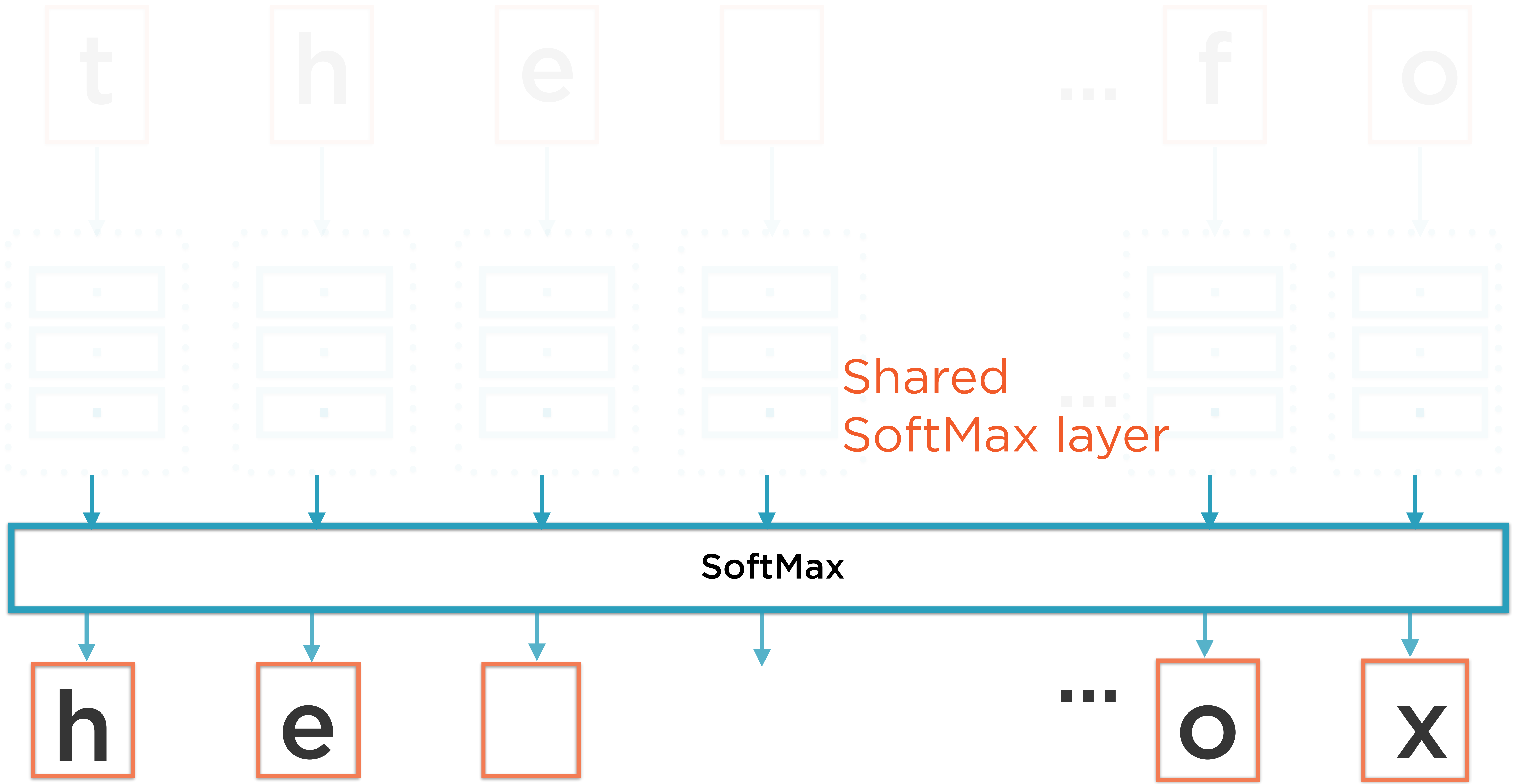
Text Prediction: RNN Architecture



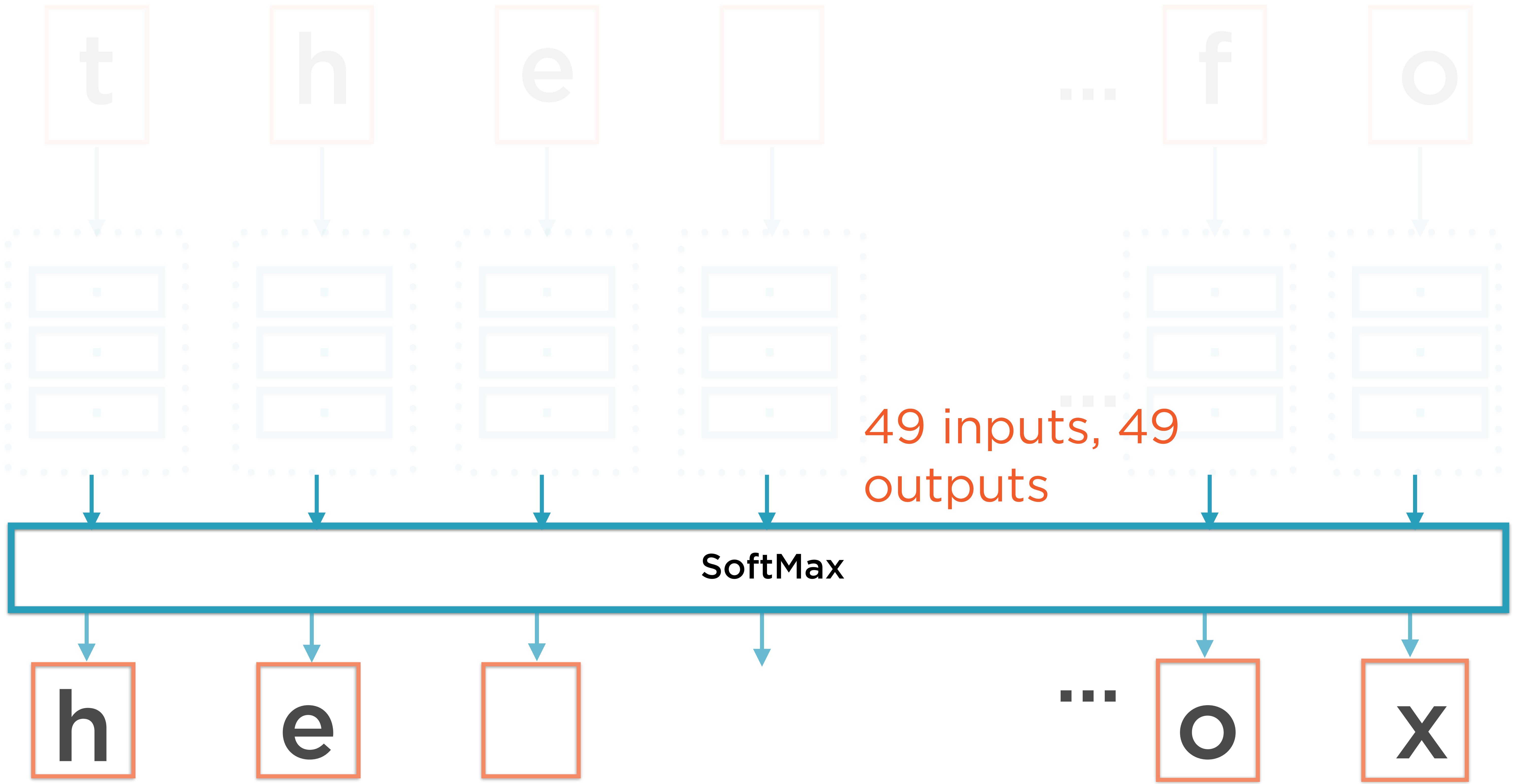
Text Prediction: RNN Architecture



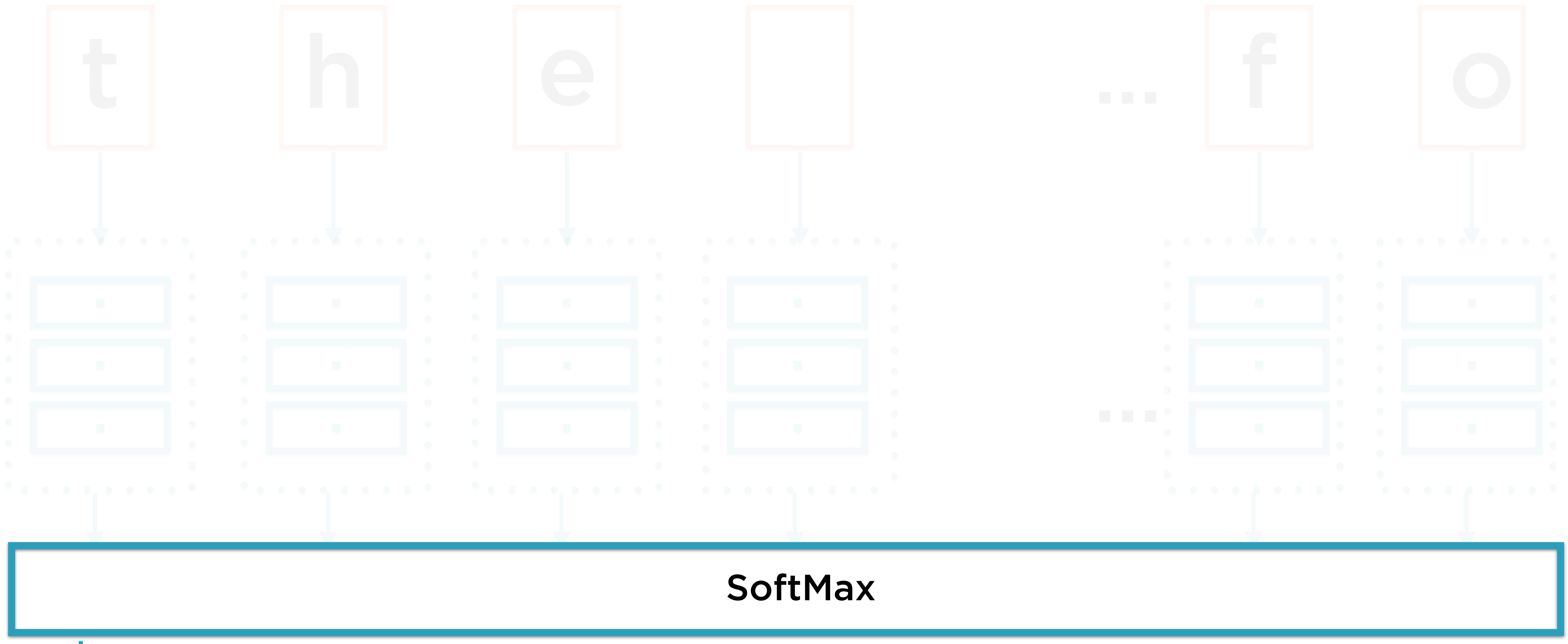
Text Prediction: RNN Architecture



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

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

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

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

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

Demo

Use the trained RNN to predict the next character and generate text

- Re-initialise the state to get better predictions

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

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

Alternative Approach to the Distant Past

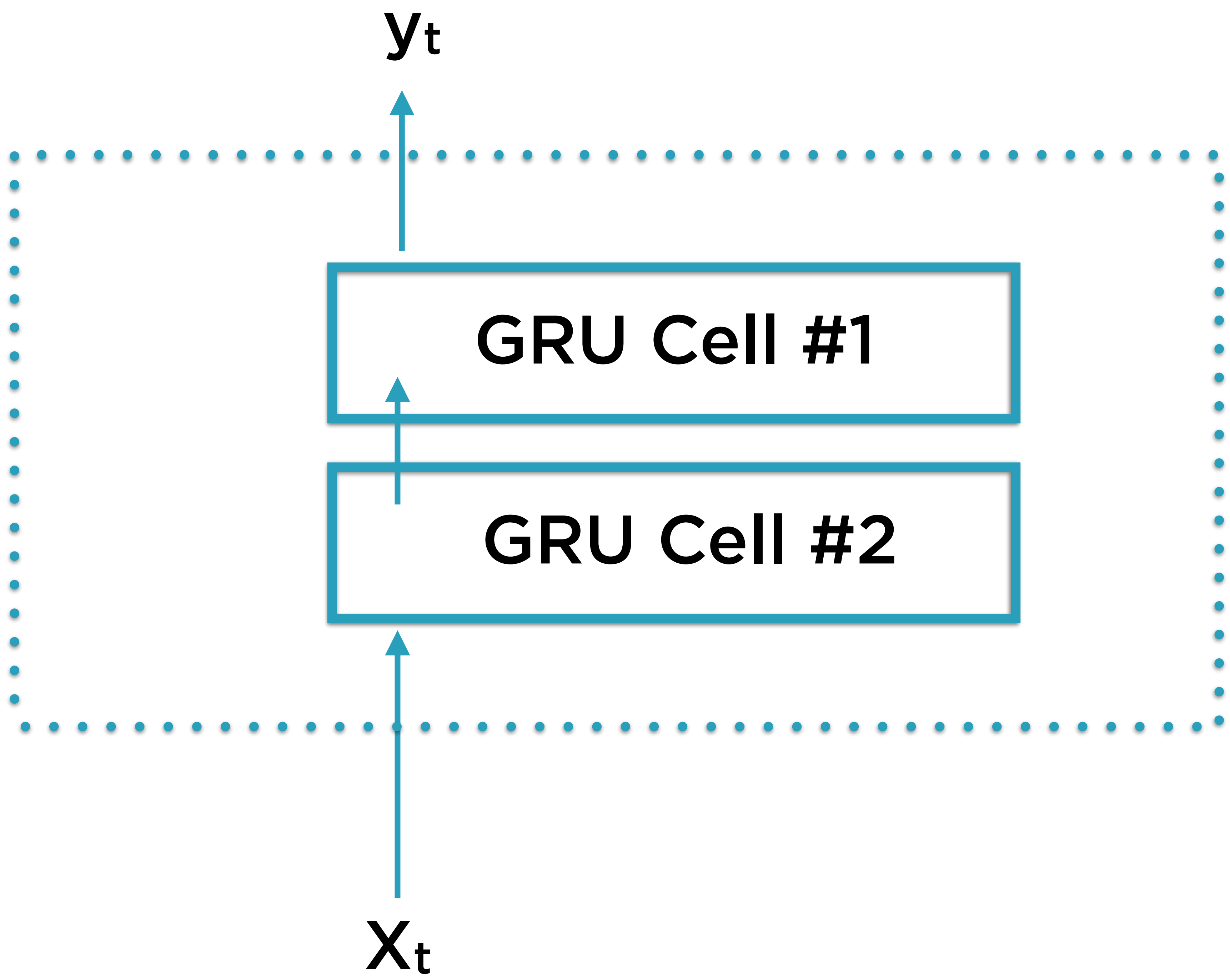
Re-using internal state in addition to using GRU gives great performance with less input

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

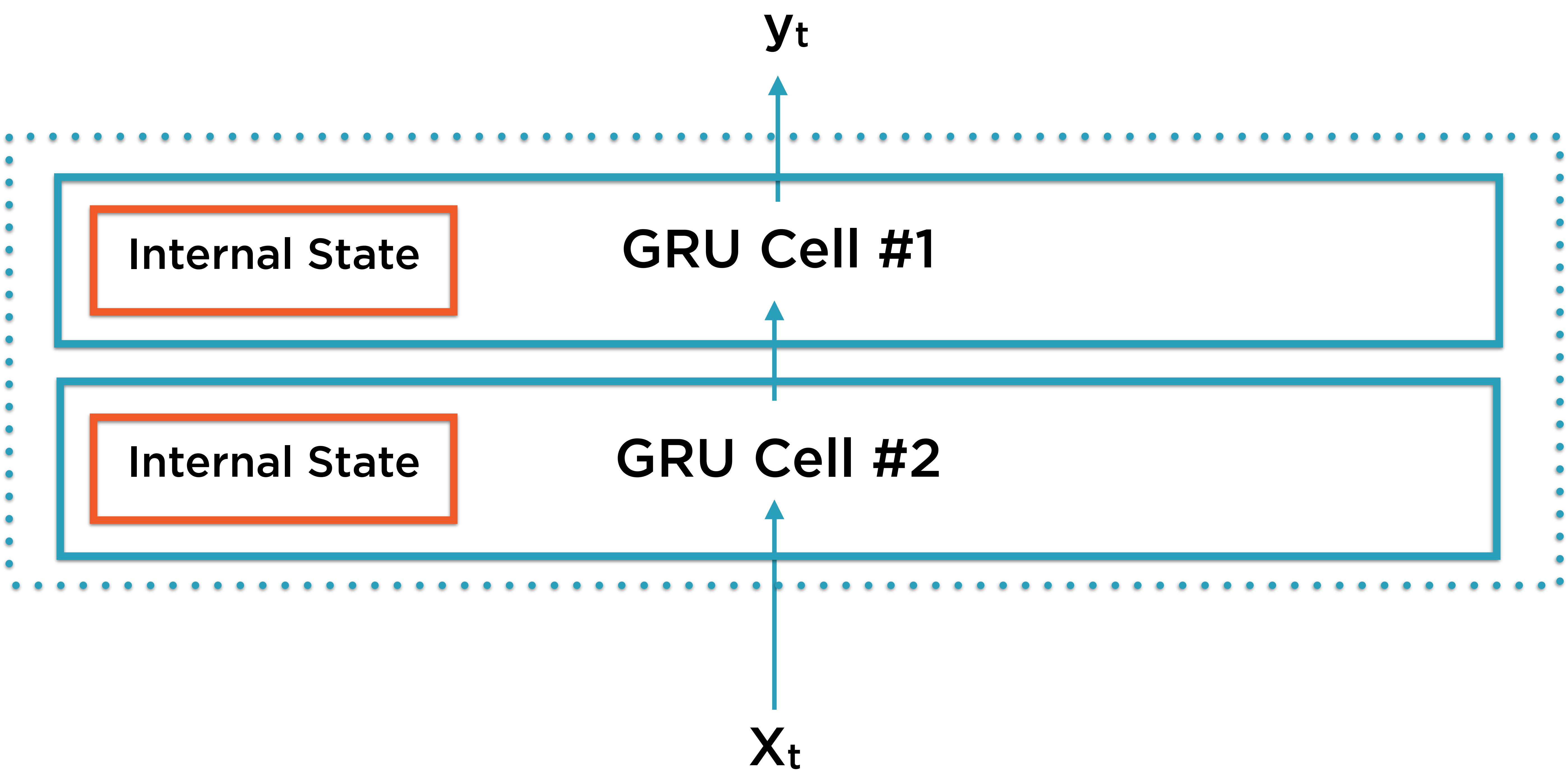
Feed the Previous State as Initial Values

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

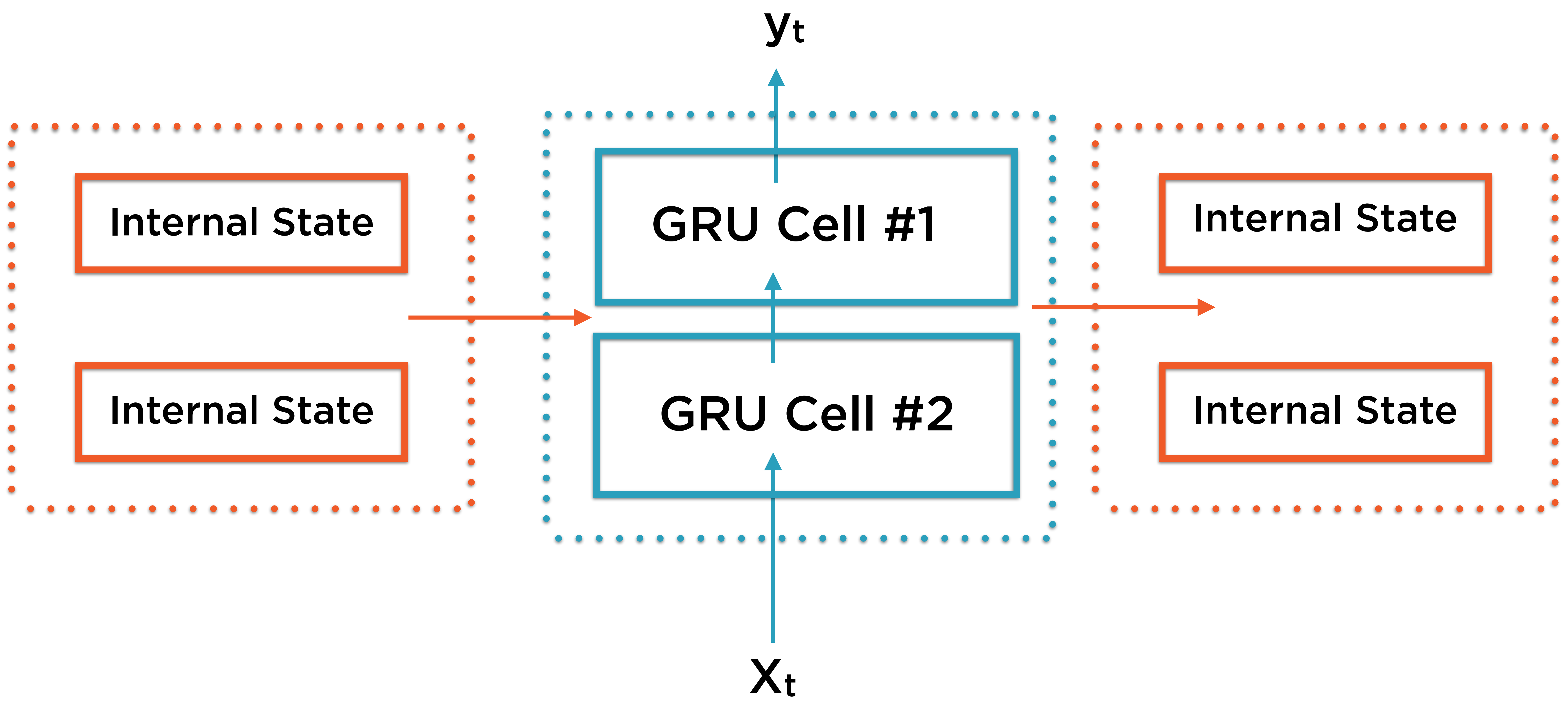
Re-using Internal State



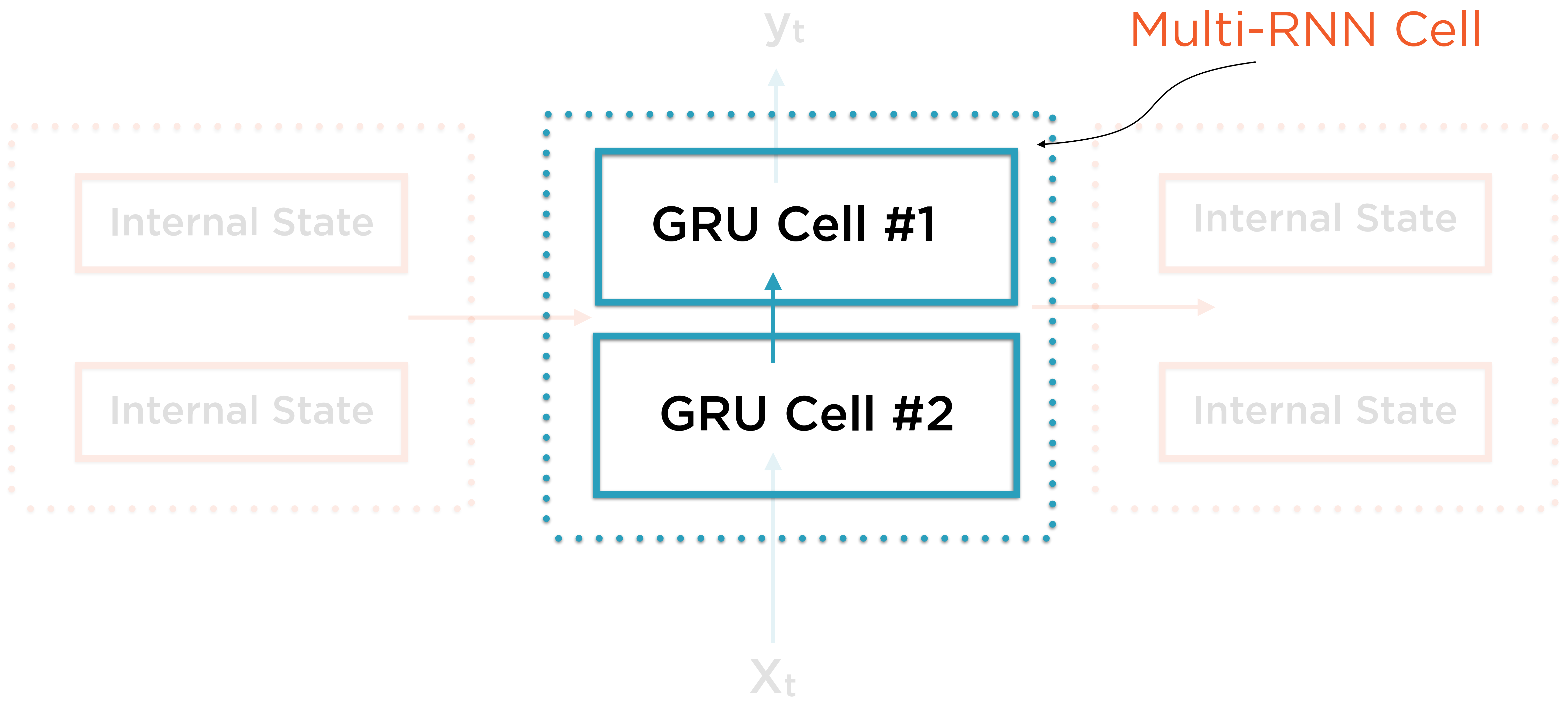
Re-using Internal State



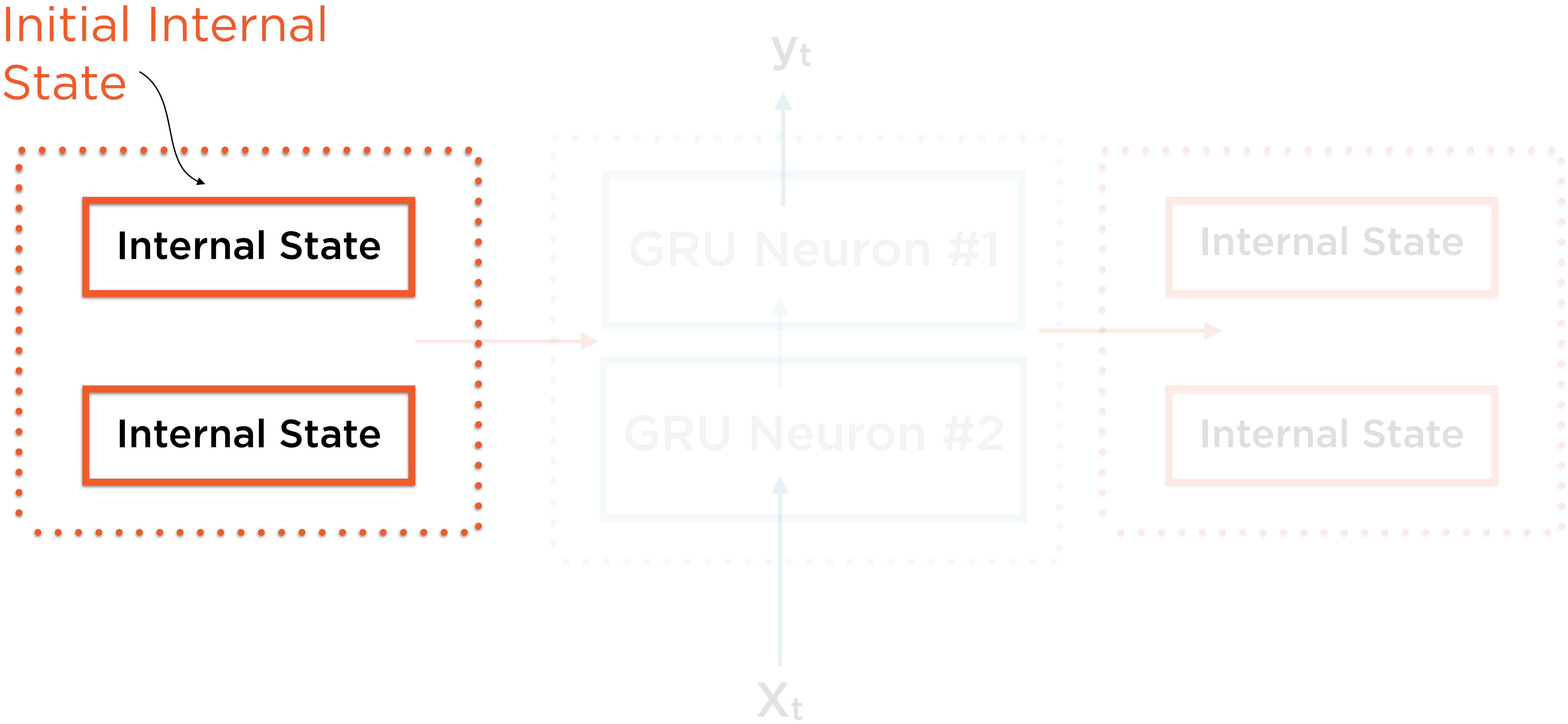
Re-using Internal State



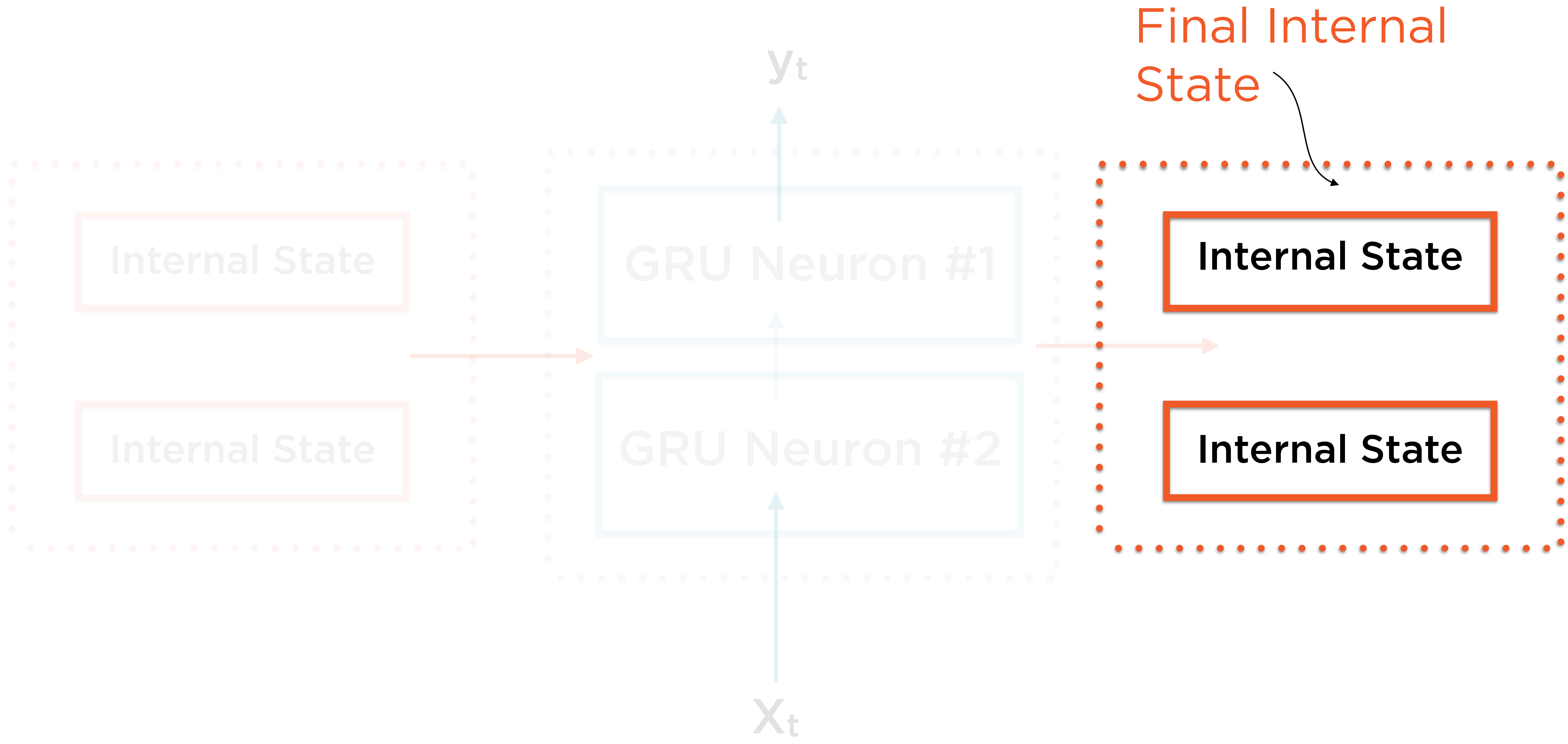
Re-using Internal State



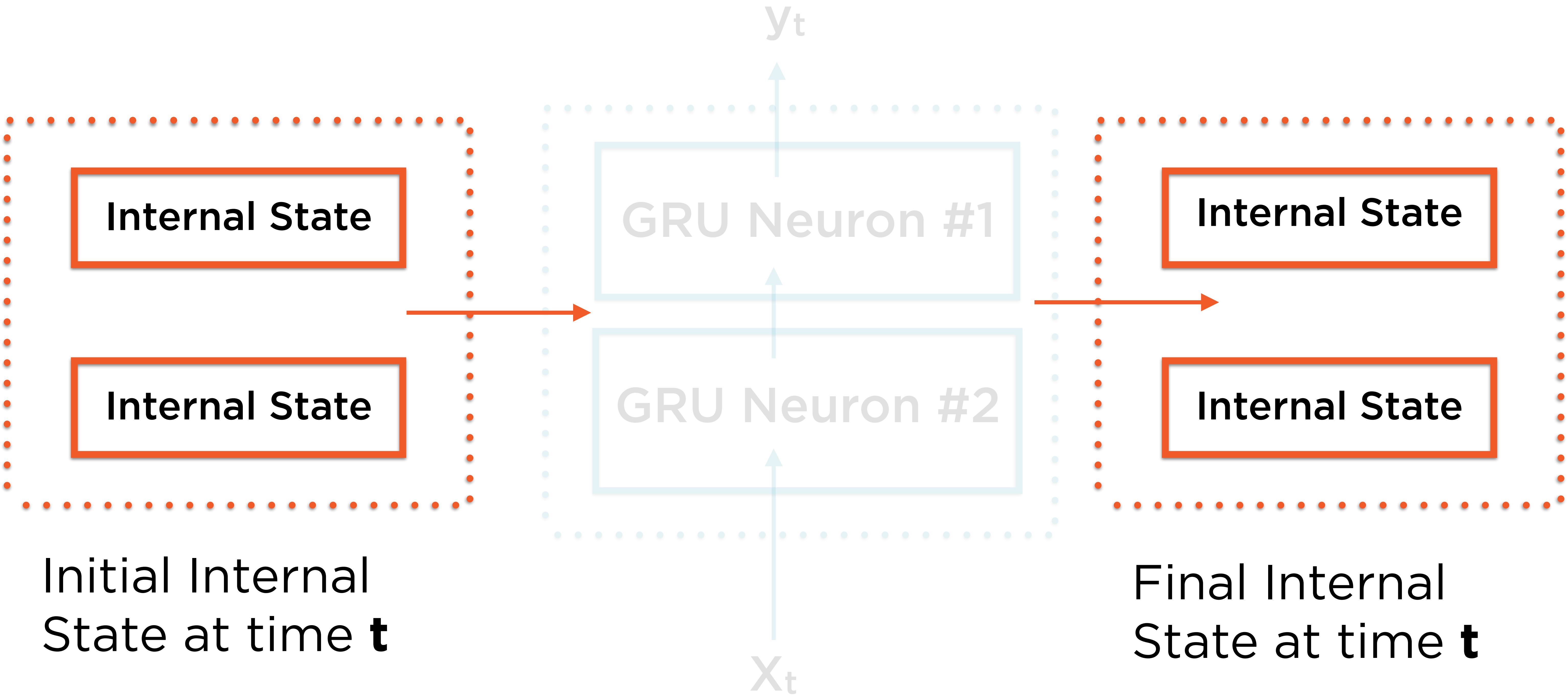
Re-using Internal State



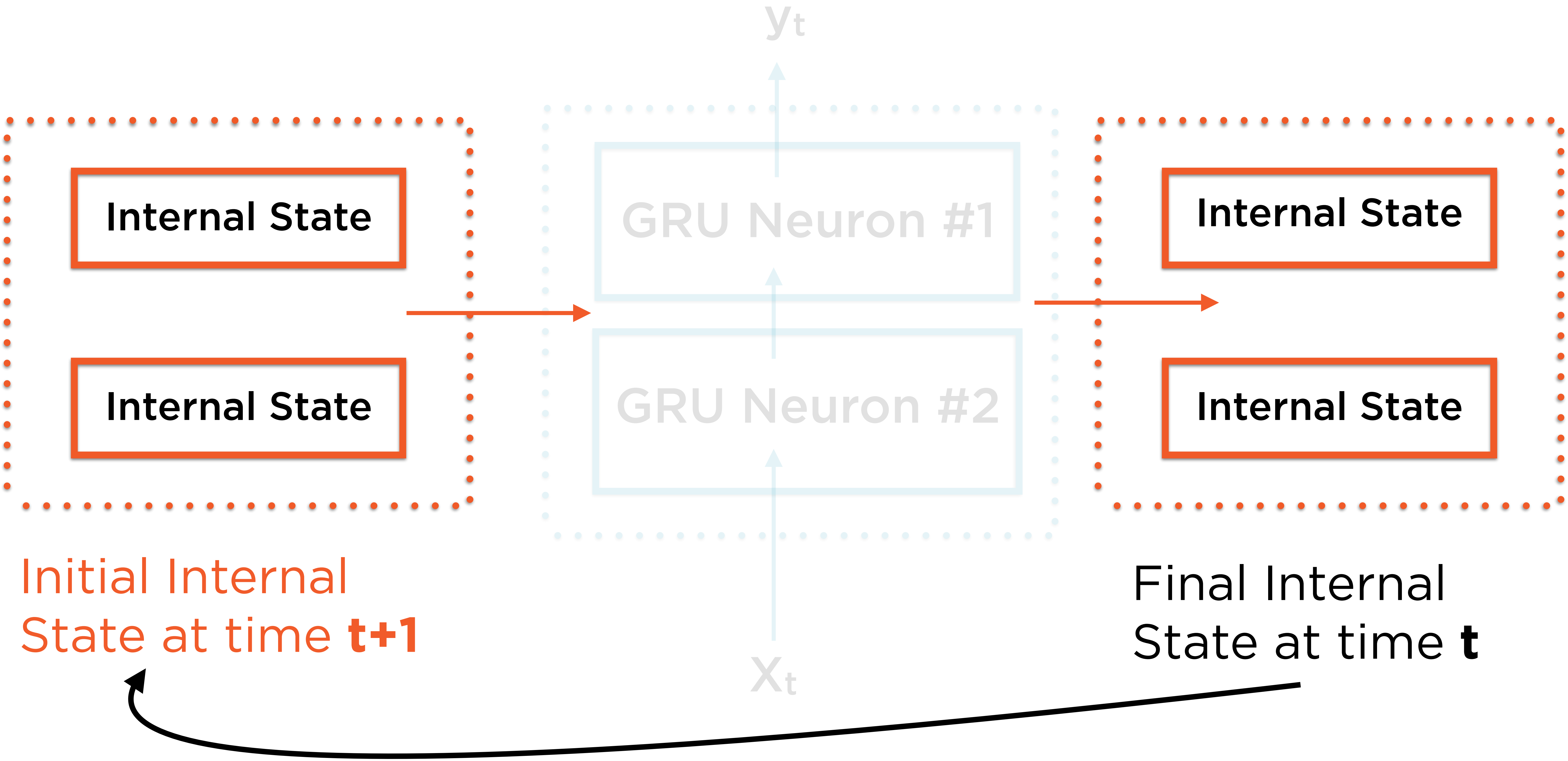
Re-using Internal State



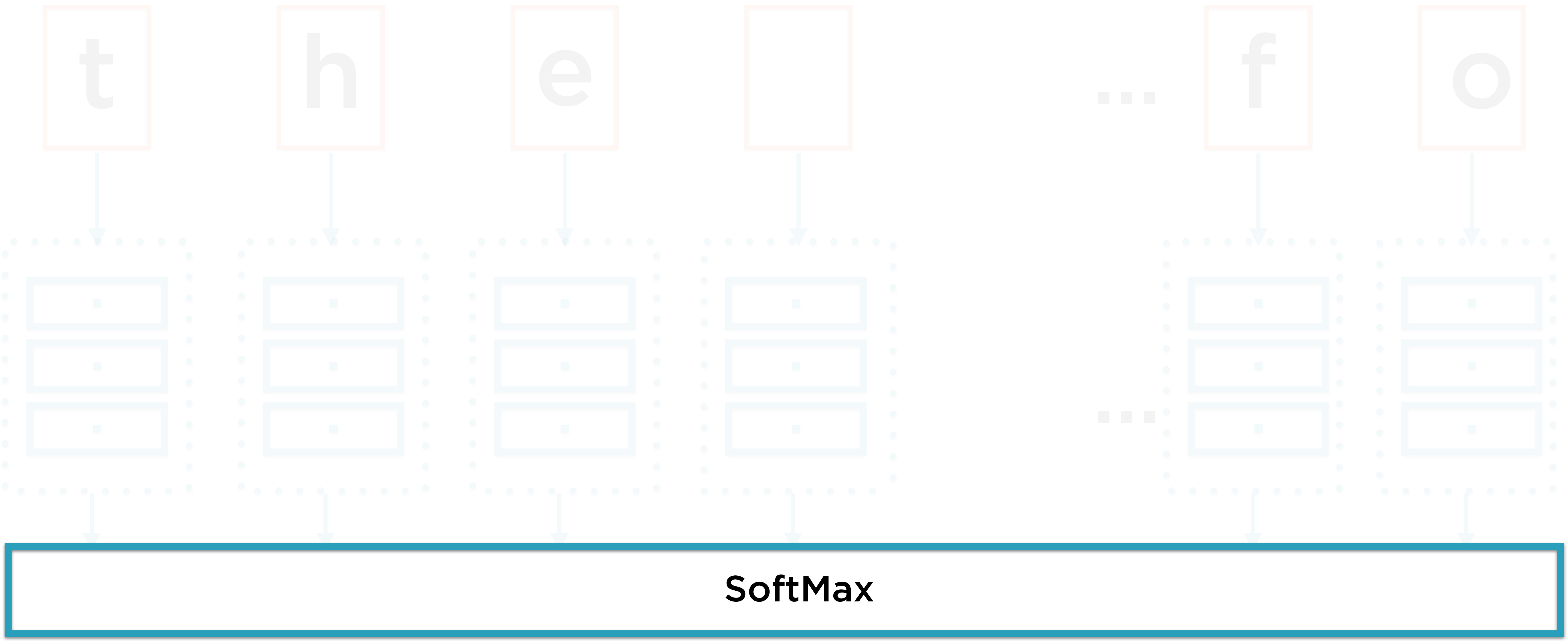
Re-using Internal State



Re-using Internal State



Text Prediction: RNN Architecture

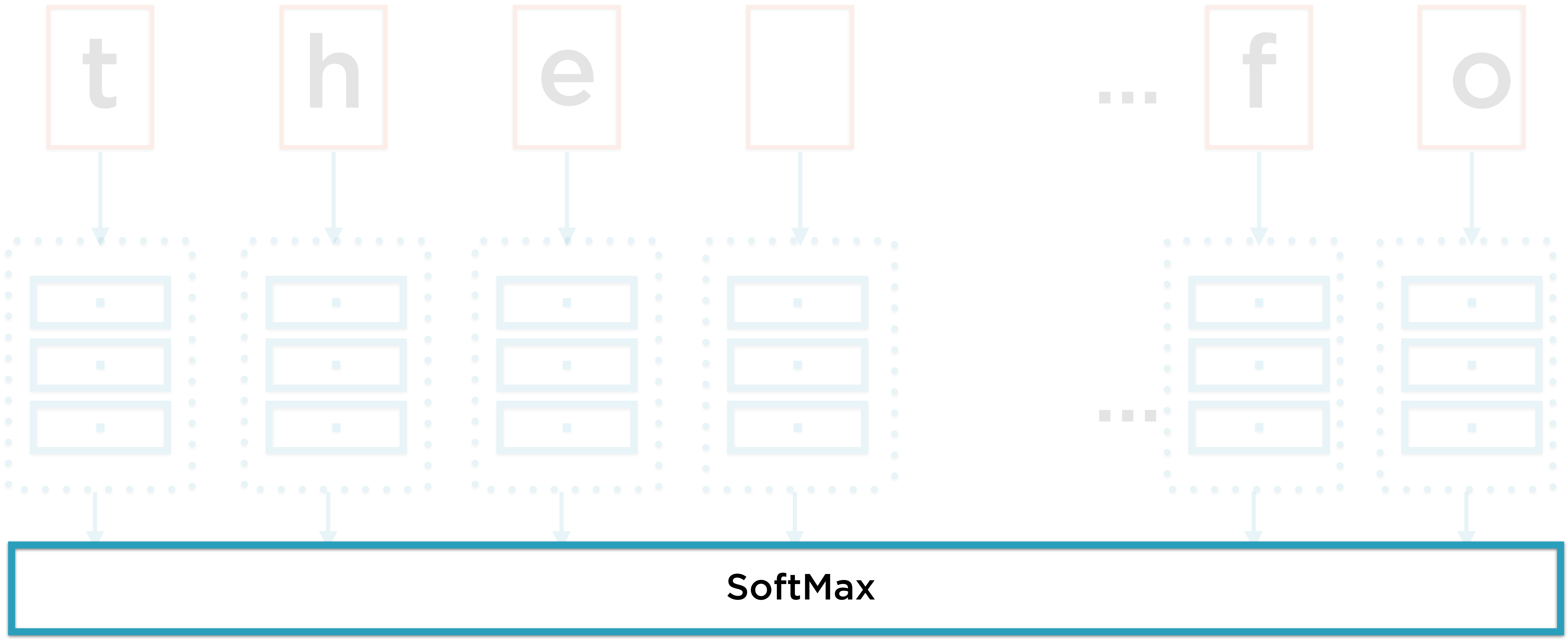


h

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

Each output: One-hot 83 probabilities for alphanumeric characters

Text Prediction: RNN Architecture



h

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

Tweak: Sampling temperature

Sampling Temperature

Introduce a temperature parameter T

Use this to rescale the probabilities output by SoftMax

h

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

character

83

Sampling Temperature

83

h

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

character

Probabilities output by SoftMax

Sampling Temperature

h

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

character

P(char = ‘a’)

Sampling Temperature

h

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

character

P(char = ‘b’)

Sampling Temperature

h

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

character

P(char = ‘h’)

P(char = ‘h’) > any other probability, so output prediction is ‘h’

Sampling Temperature

Probabilities output by SoftMax

h

character

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

a	b	c		h	...
0.01	0.0	0.01		0.98	0

Transformed probabilities

Sampling Temperature

$$P^{\text{'h'}}_{\text{Transformed}} = \frac{{P^{\text{'h'}}_{\text{Softmax}}}^{\frac{1}{T}}}{\sum_{i} {P^{i}_{\text{Softmax}}}^{\frac{1}{T}}}$$

T is a parameter called the sampling temperature

Sampling Temperature

$$P^{\text{'h'}}_{\text{Transformed}} = \frac{\left(P^{\text{'h'}}_{\text{Softmax}}\right)^{\frac{1}{T}}}{\sum_{i} \left(P^{i}_{\text{Softmax}}\right)^{\frac{1}{T}}}$$

New (transformed) probability that the character is ‘h’

Sampling Temperature

$$P_{Transformed}^{'h'} = \frac{\left(P_{Softmax}^{'h'}\right)^{\frac{1}{T}}}{\sum_i \left(P_{Softmax}^{i}\right)^{\frac{1}{T}}}$$

Softmax (non-transformed) probability that the character is ‘h’

Sampling Temperature

$$P_{Transformed}^{\text{'h'}} = \frac{\left(P_{Softmax}^{\text{'h'}}\right)^{\frac{1}{T}}}{\sum_{i} \left(P_{Softmax}^{i}\right)^{\frac{1}{T}}}$$

Softmax (non-transformed)
probabilities of all other characters i

Sampling Temperature

$$P^{\text{'h'}}_{Transformed} = \frac{\left(P^{\text{'h'}}_{Softmax}\right)^{\frac{1}{T}}}{\sum_i \left(P^{i}_{Softmax}\right)^{\frac{1}{T}}}$$

Sum over all such characters i

Sampling Temperature

$$P_{\text{Transformed}}^{\text{'h'}} = \frac{\left(P_{\text{Softmax}}^{\text{'h'}}\right)^{\frac{1}{T}}}{\sum_i \left(P_{\text{Softmax}}^{i}\right)^{\frac{1}{T}}}$$

Sampling temperature T

Sampling Temperature

$$P_{Transformed}^{'h'} = \frac{P_{Softmax}^{'h' \; \frac{1}{T}}}{\sum_i P_{Softmax}^{i \; \frac{1}{T}}}$$

When $T = 1$, $P_{Transformed} = P_{Softmax}$

Sampling Temperature

$$P_{Transformed}^{'h'} = \frac{P_{Softmax}^{'h' \, \frac{1}{T}}}{\sum_{i} P_{Softmax}^{i \, \frac{1}{T}}}$$

When T approaches 0, $P_{Transformed}$ approaches one-hot

Sampling Temperature

$$P_{Transformed}^{\text{'h'}} = \frac{\left(P_{Softmax}^{\text{'h'}}\right)^{\frac{1}{T}}}{\sum_i \left(P_{Softmax}^{i}\right)^{\frac{1}{T}}}$$

When T approaches Infinity,
$P_{Transformed}$ boosts small probabilities

Sampling Temperature

Value of T	Implication
T = 1	SoftMax probabilities used as-is
Large values of T	Up-weight small probabilities
Small values of T	Down-weight small probabilities
T = 0	Equivalent to one-hot - set largest softmax probability to 1, rest to 0

Increasing sampling temperature makes output more creative, but less correct

Summary

Implemented an RNN trained on abstracts of technical papers

A multi-RNN allows you to compose several cells as a single one, can store additional state

Re-initialize state of the RNN during prediction to improve output

Perplexity is a common evaluation metric for language modeling problems

Books

Hands-On Machine Learning with Scikit-Learn and TensorFlow

Aurélien Géron

Related Courses

Building Unsupervised Learning Models with TensorFlow

Building Classification Models with TensorFlow