Implementing Character Recognition Using Bidirectional RNNs

Janani Ravi
CO-FOUNDER, LOONYCORN

www.loonycorn.com

Overview

Implement a RNN of GRU cells for OCR using a modified version of the MIT OCR dataset

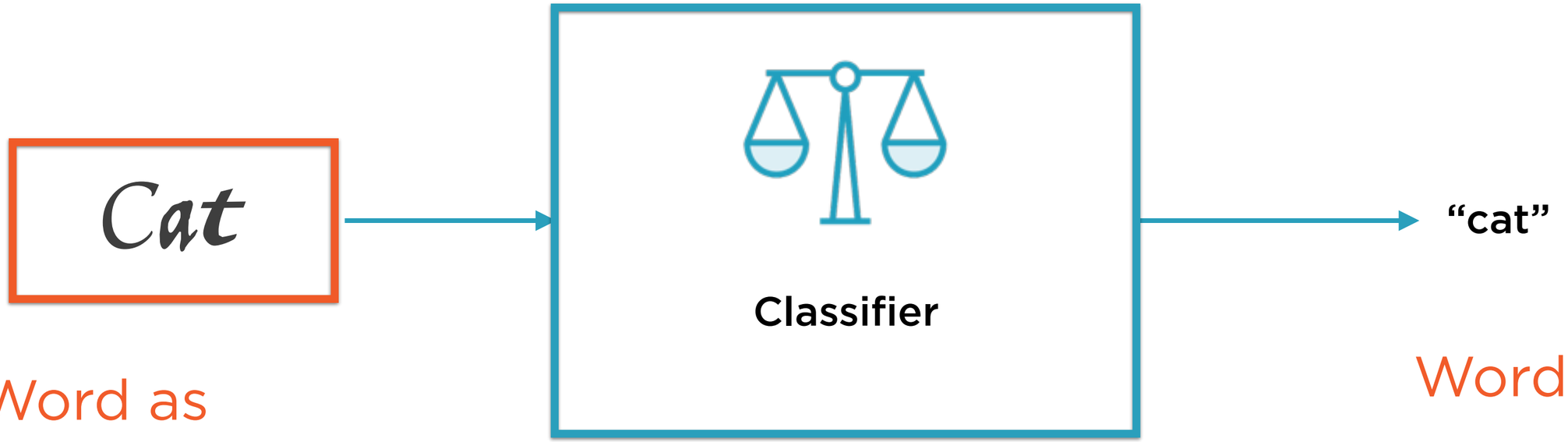
Use a conventional RNN for character recognition and calculate accuracy

Manually build a bidirectional RNN

Use the TensorFlow library to build a bidirectional RNN

OCR as a Sequence Labeling Problem

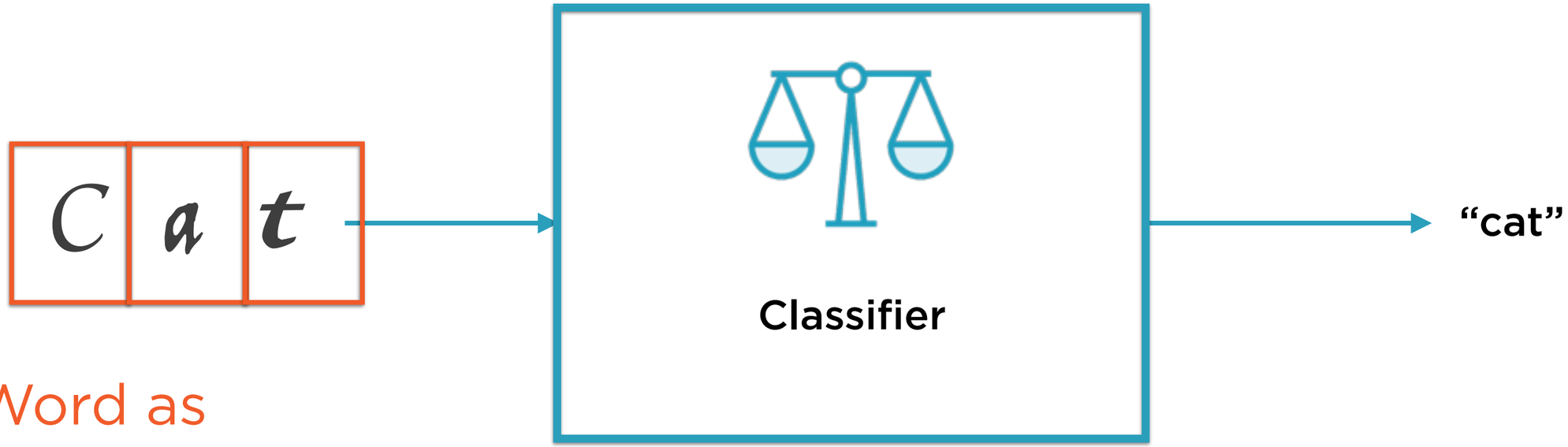
OCR Word Recognition



Word as sequence of character images

Word

OCR Word Recognition



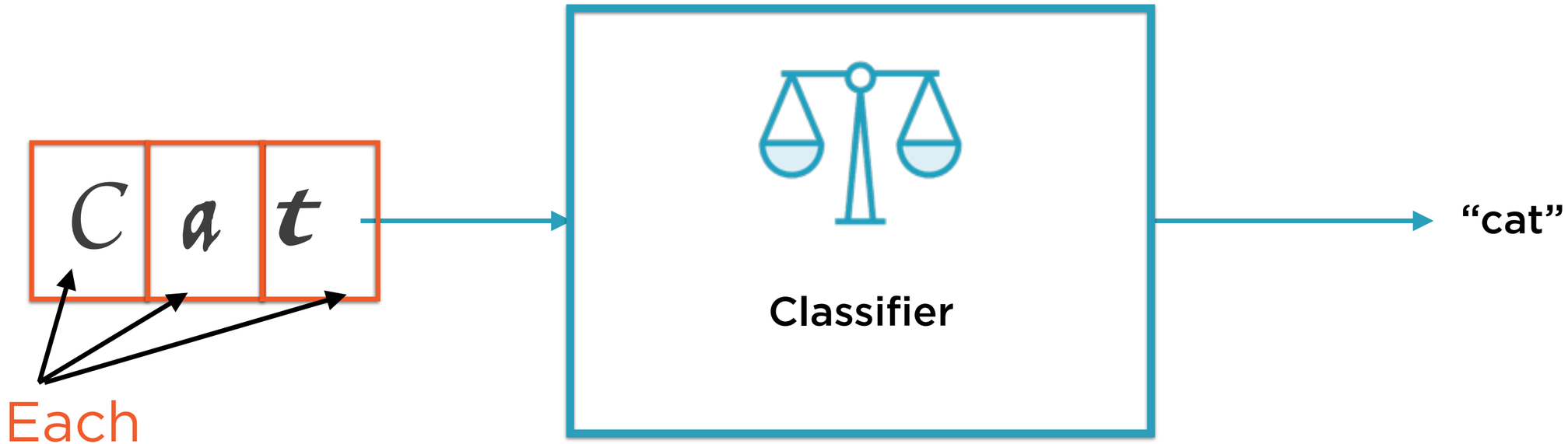
Word as sequence of character images

Demo

Perform optical character recognition using an RNN on a modified version of the MIT OCR dataset

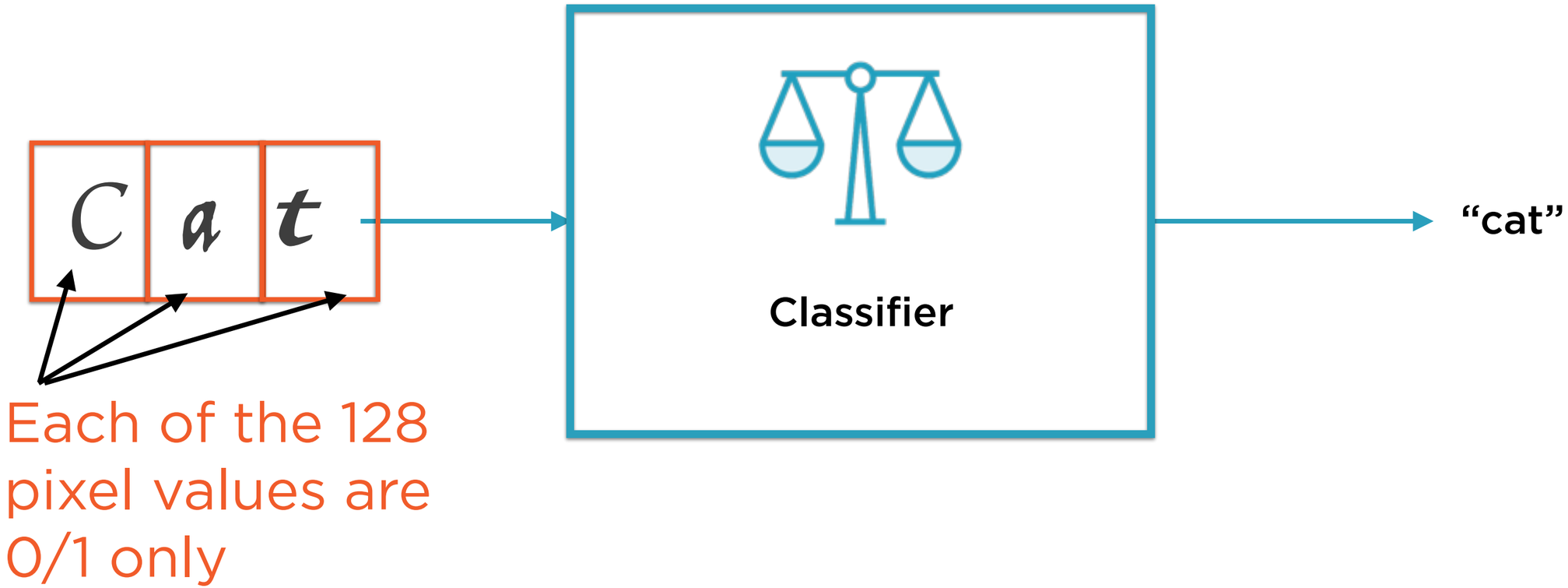
- Identify character images in the context of the word in which they occur

OCR Word Recognition



Each character image 16x8 px

OCR Word Recognition



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							

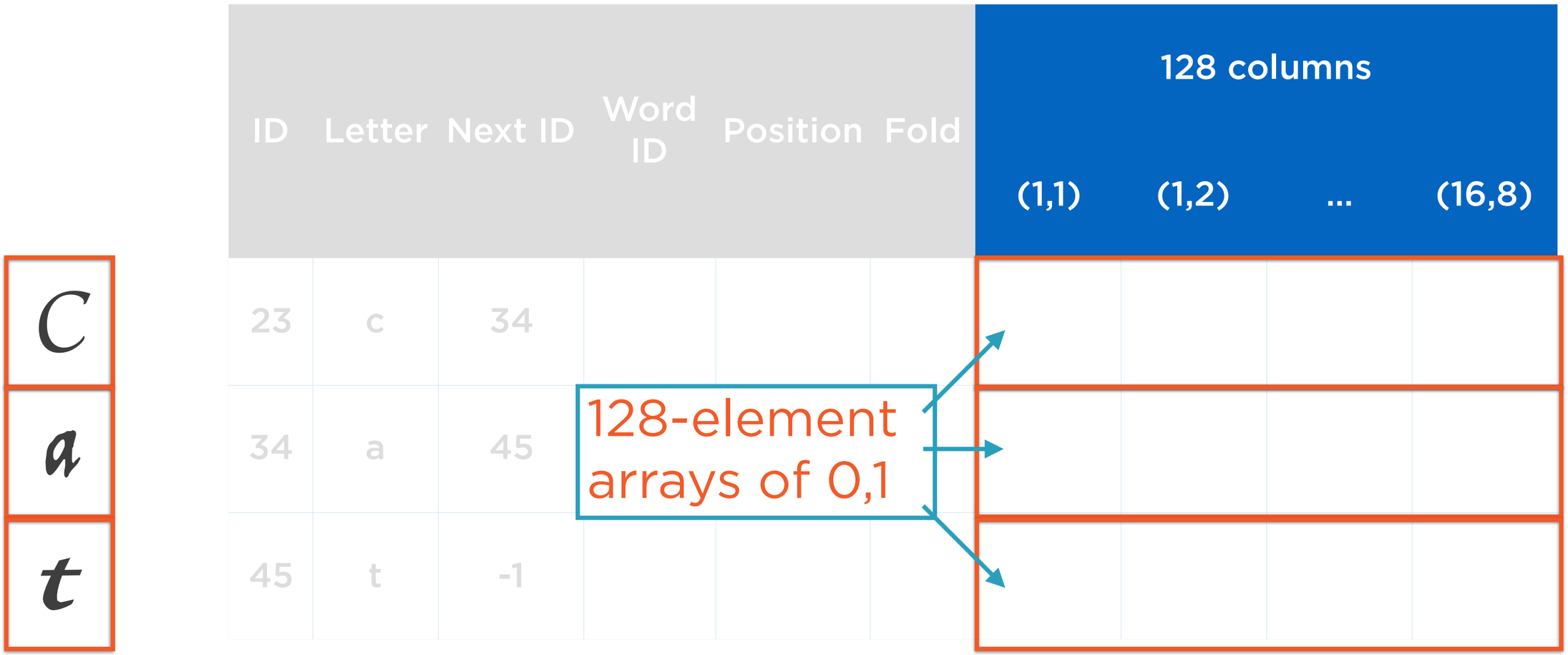
0-9 for cross-validation

MIT OCR File Format

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							

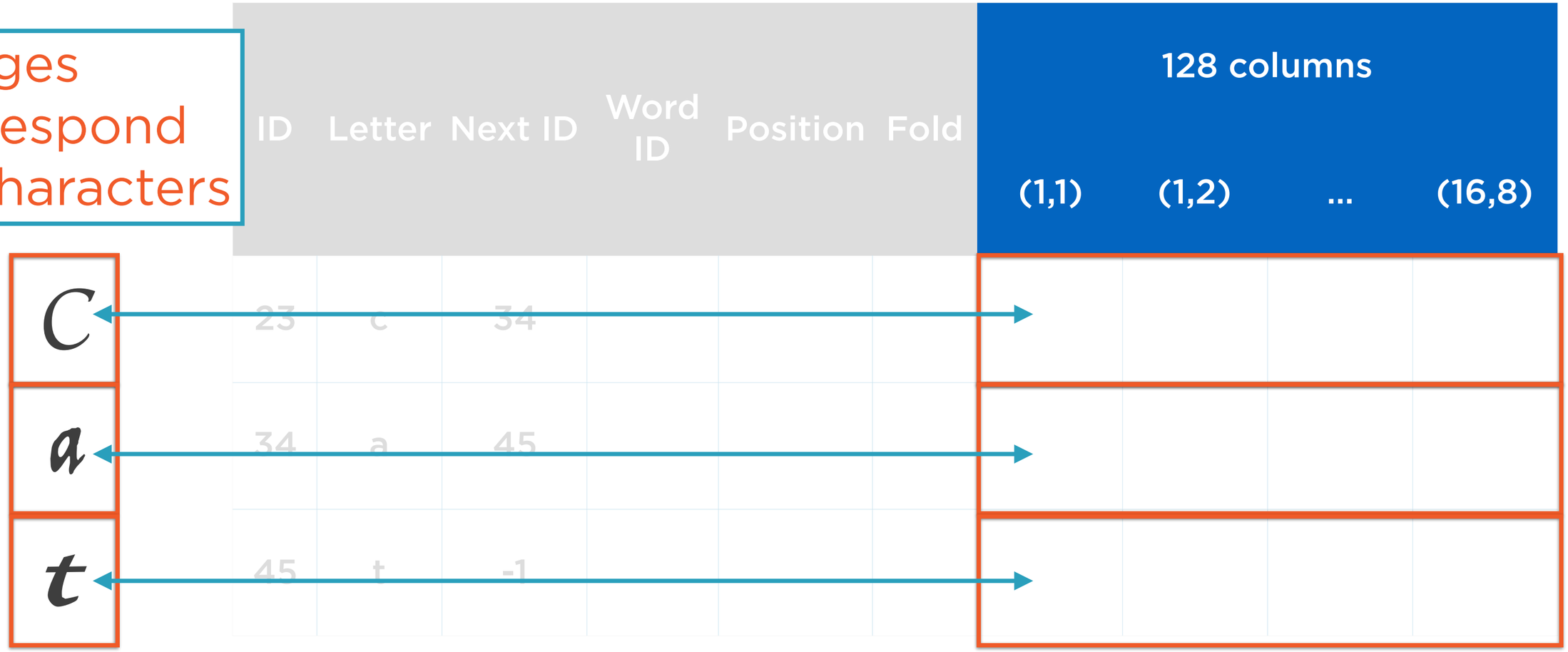
Each array is 1 image of 128 pixels

MIT OCR File Format



MIT OCR File Format

Images correspond to characters



Features and Labels

Labels

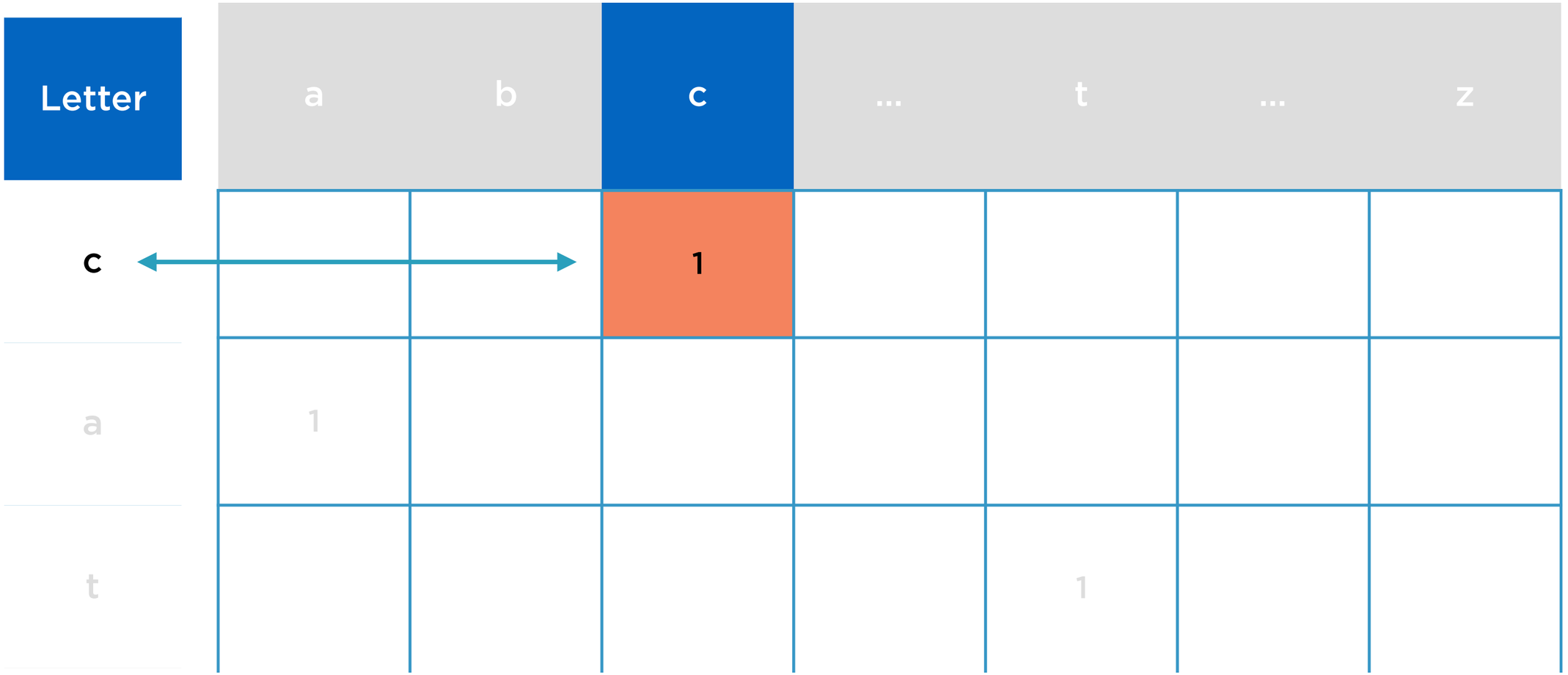
ID	Letter	Next ID	Word ID	Position	Fold	128 columns (1,1)	(1,2)	...	(16,8)
23	c	34							
34	a	45							
45	t	-1							

One-hot Encoded Labels

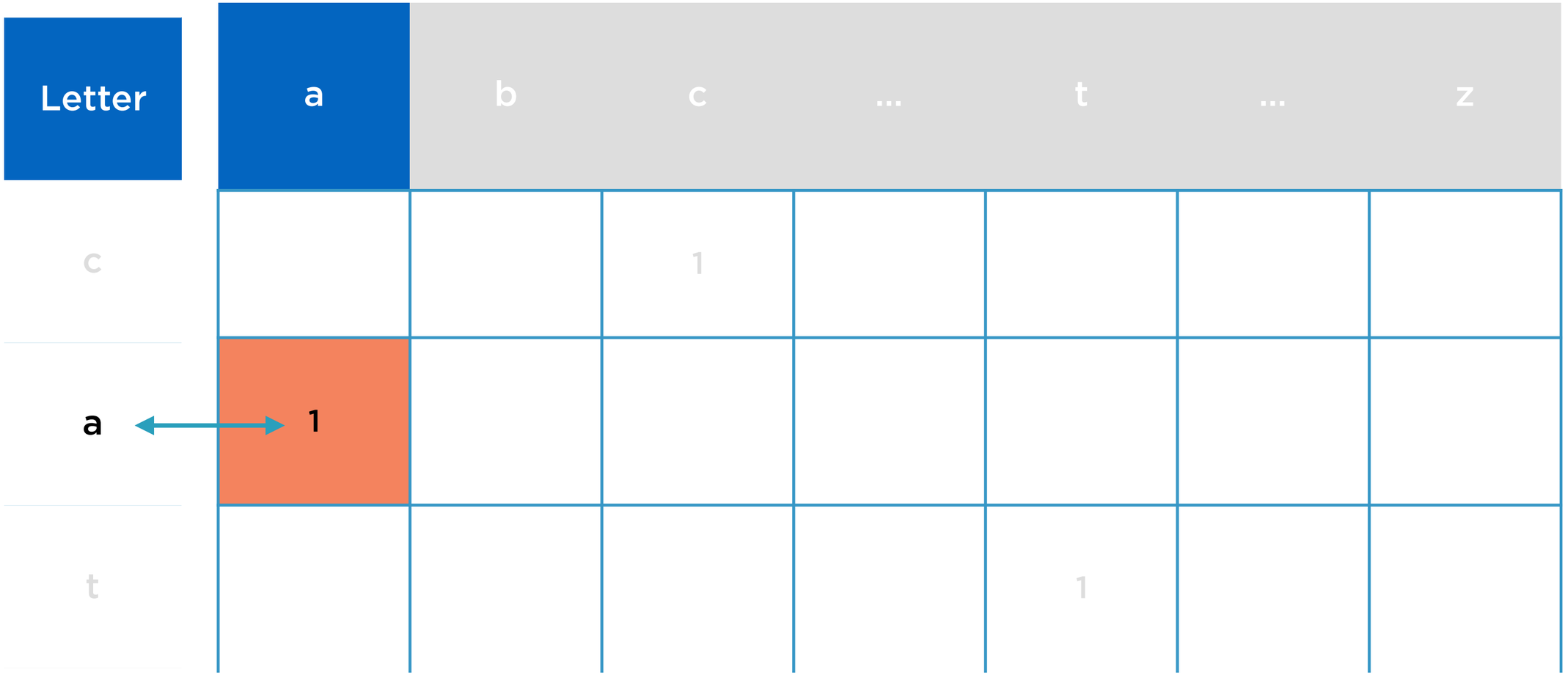
Letter	a	b	c	...	t	...	z
c			1				
a	1						
t							

26 elements

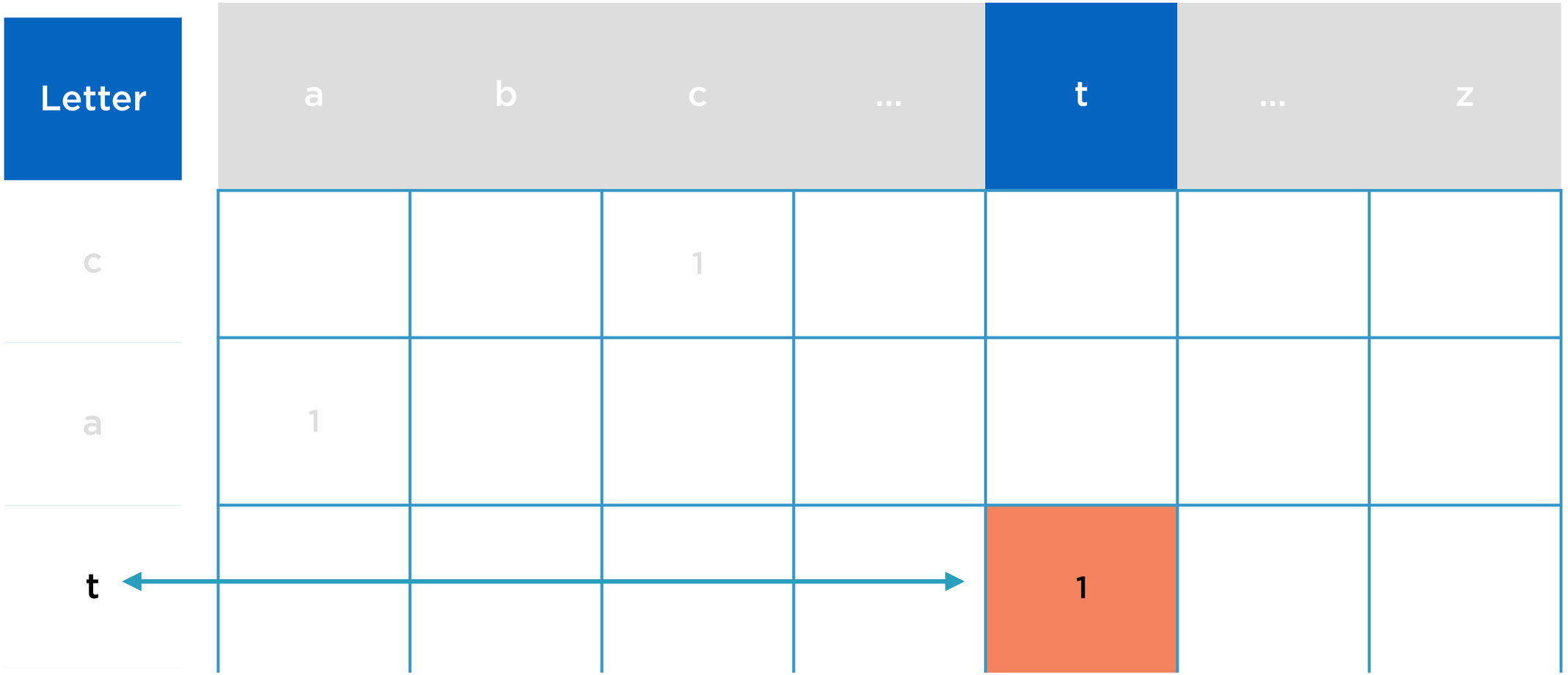
One-hot Encoded Labels



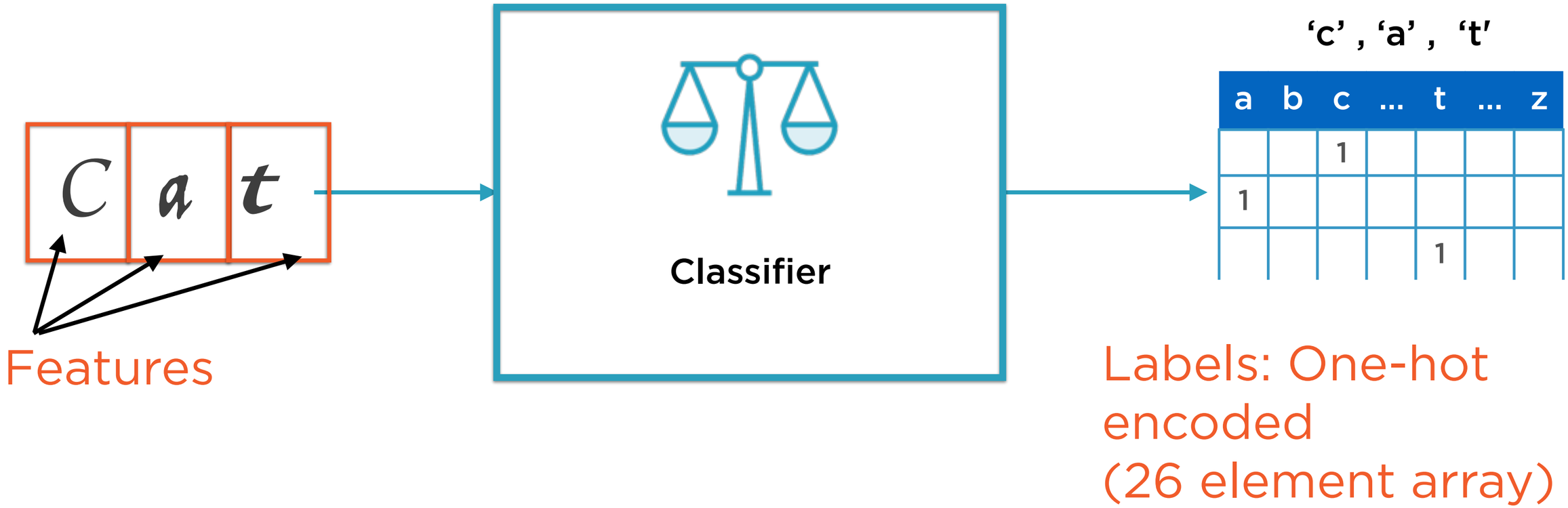
One-hot Encoded Labels



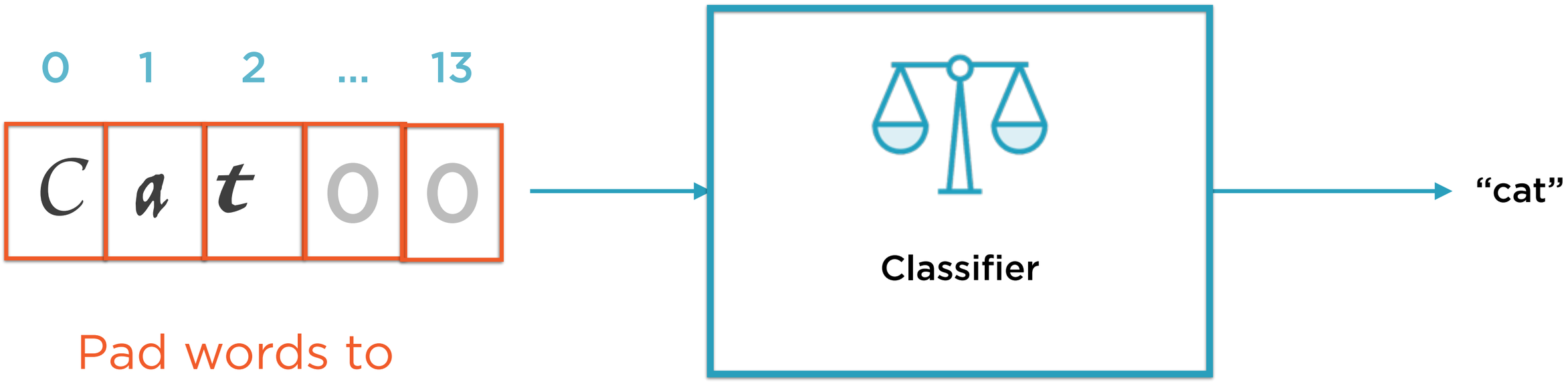
One-hot Encoded Labels



Features and Labels

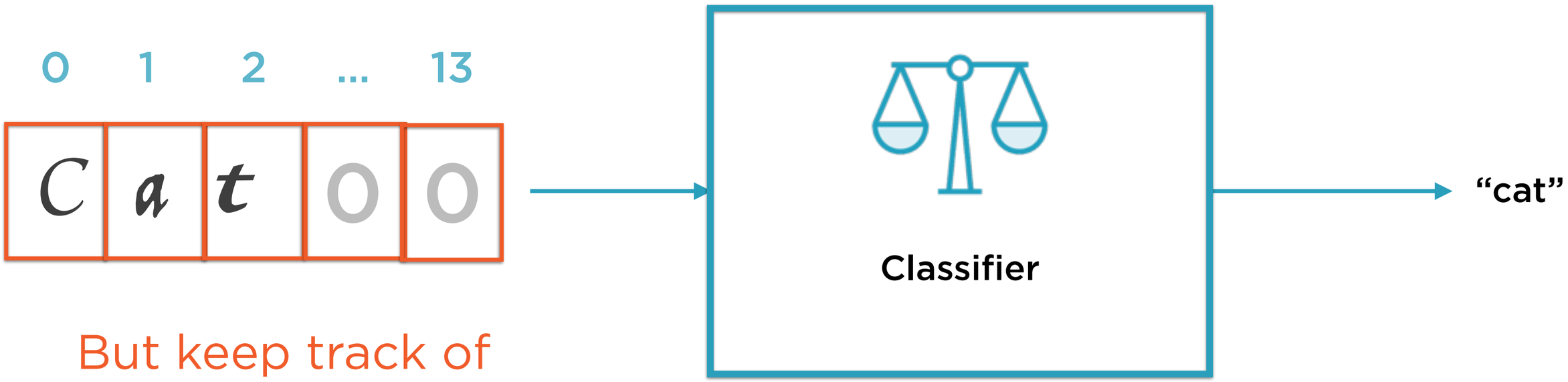


OCR Word Recognition



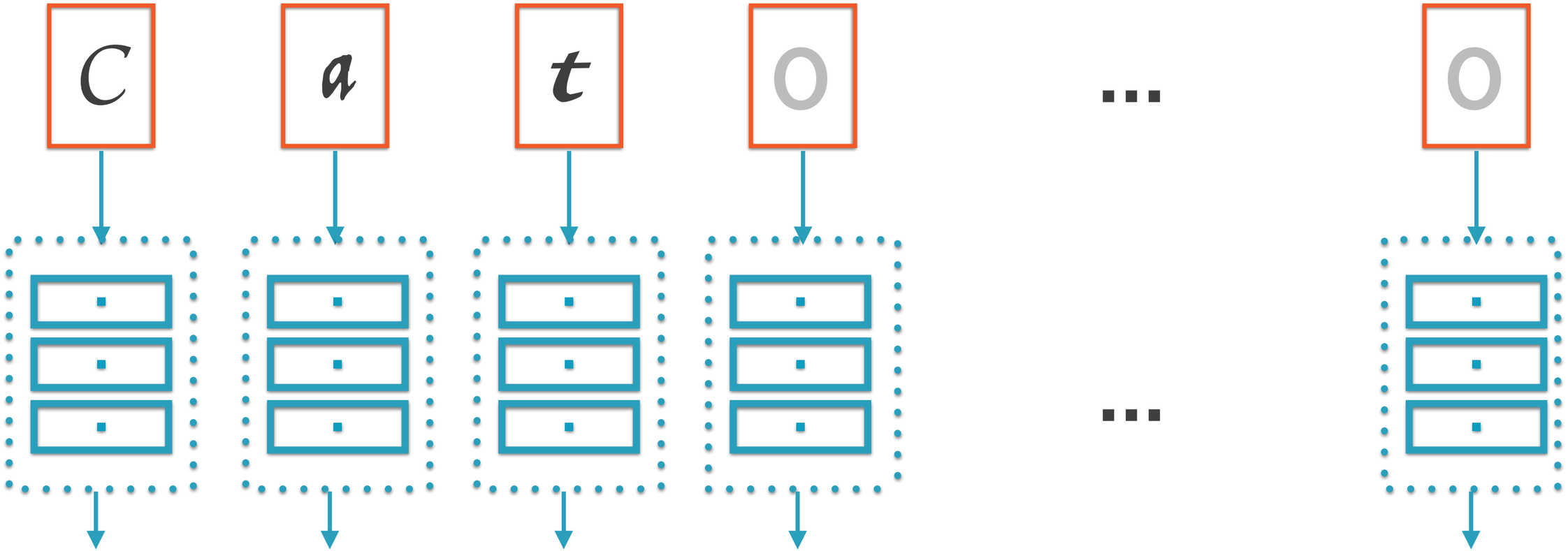
Pad words to all be of 14 characters

OCR Word Recognition

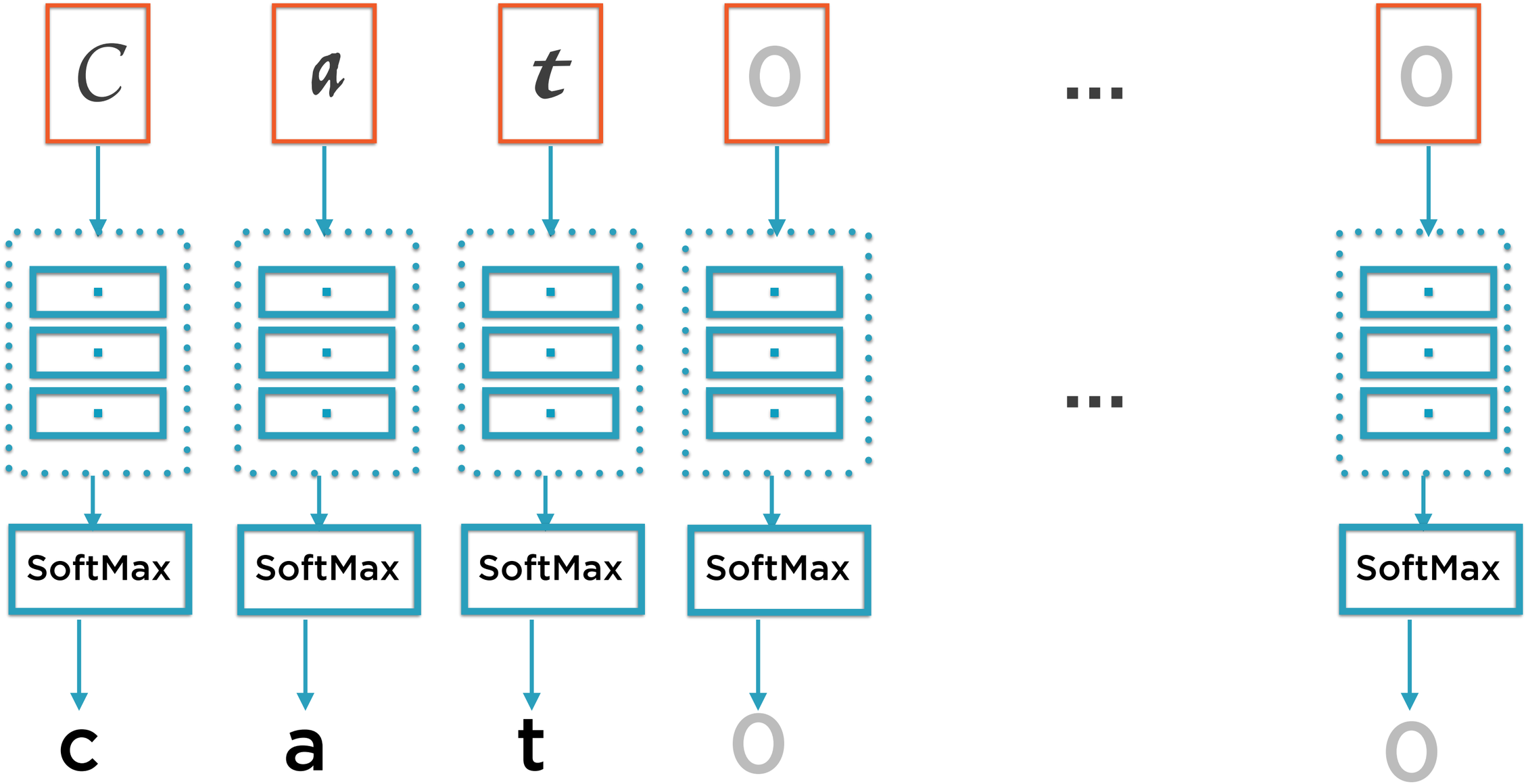


But keep track of sequence lengths of each word

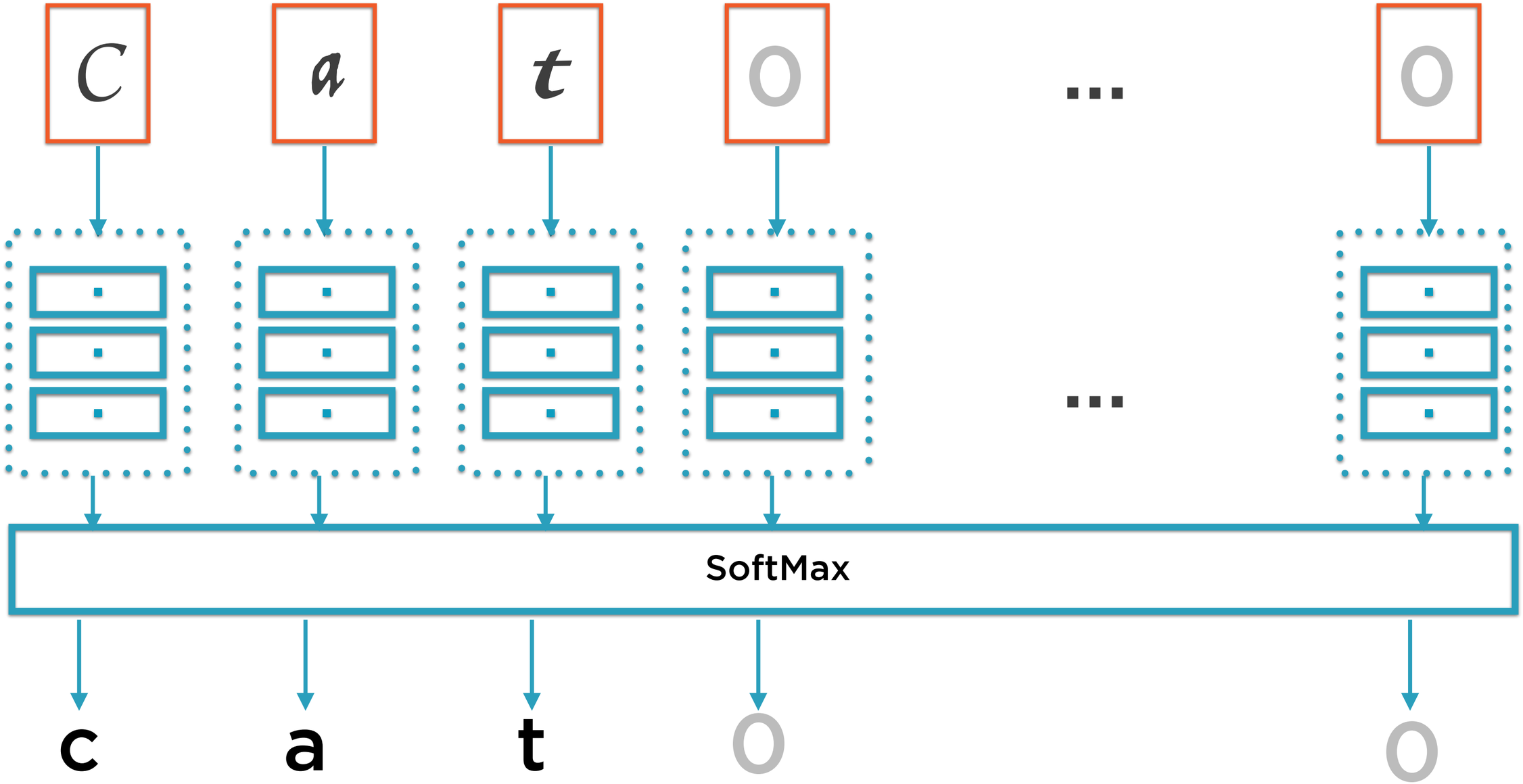
RNN Architecture



Softmax for Prediction

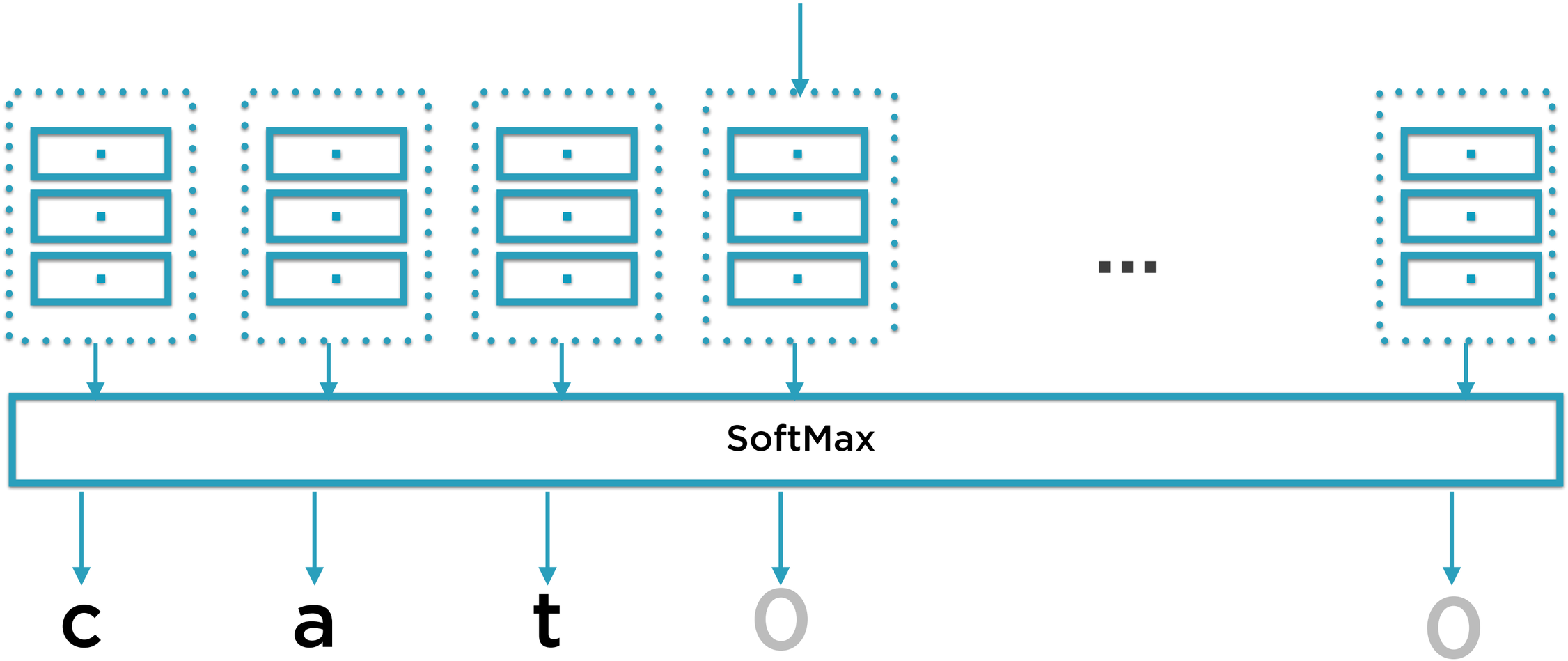


Shared Softmax

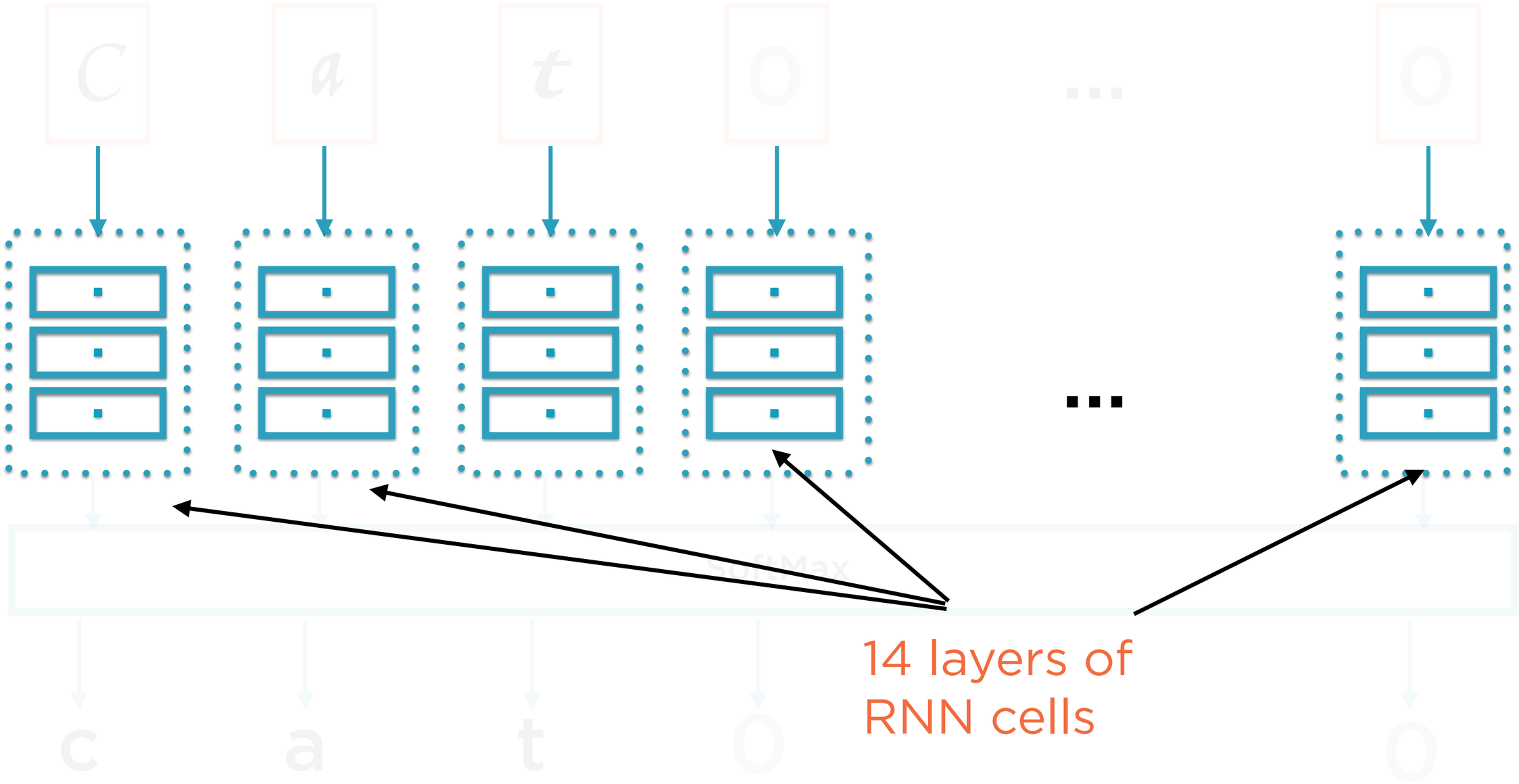


Input Tensor for Training

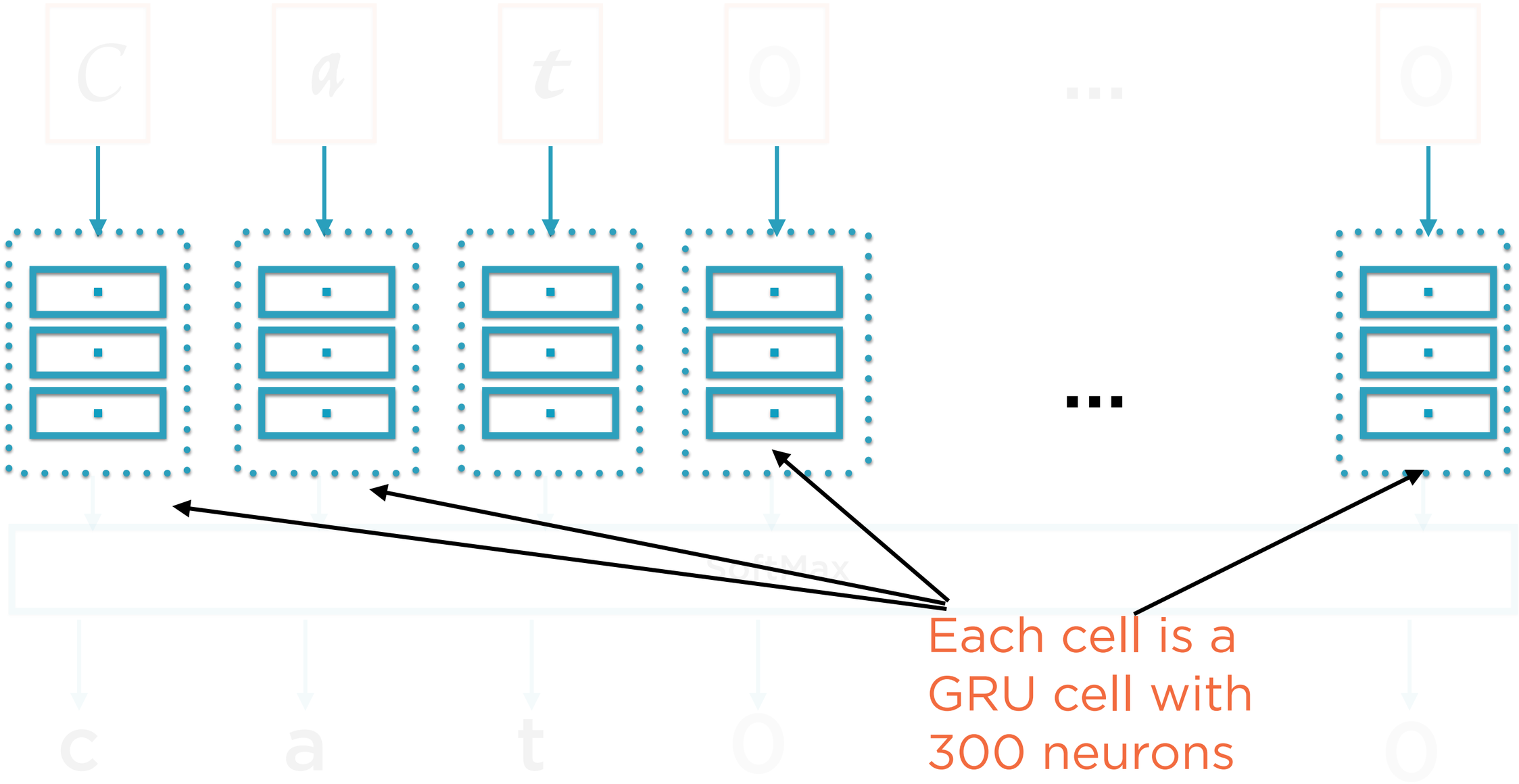
[batch_size, 14, 128]



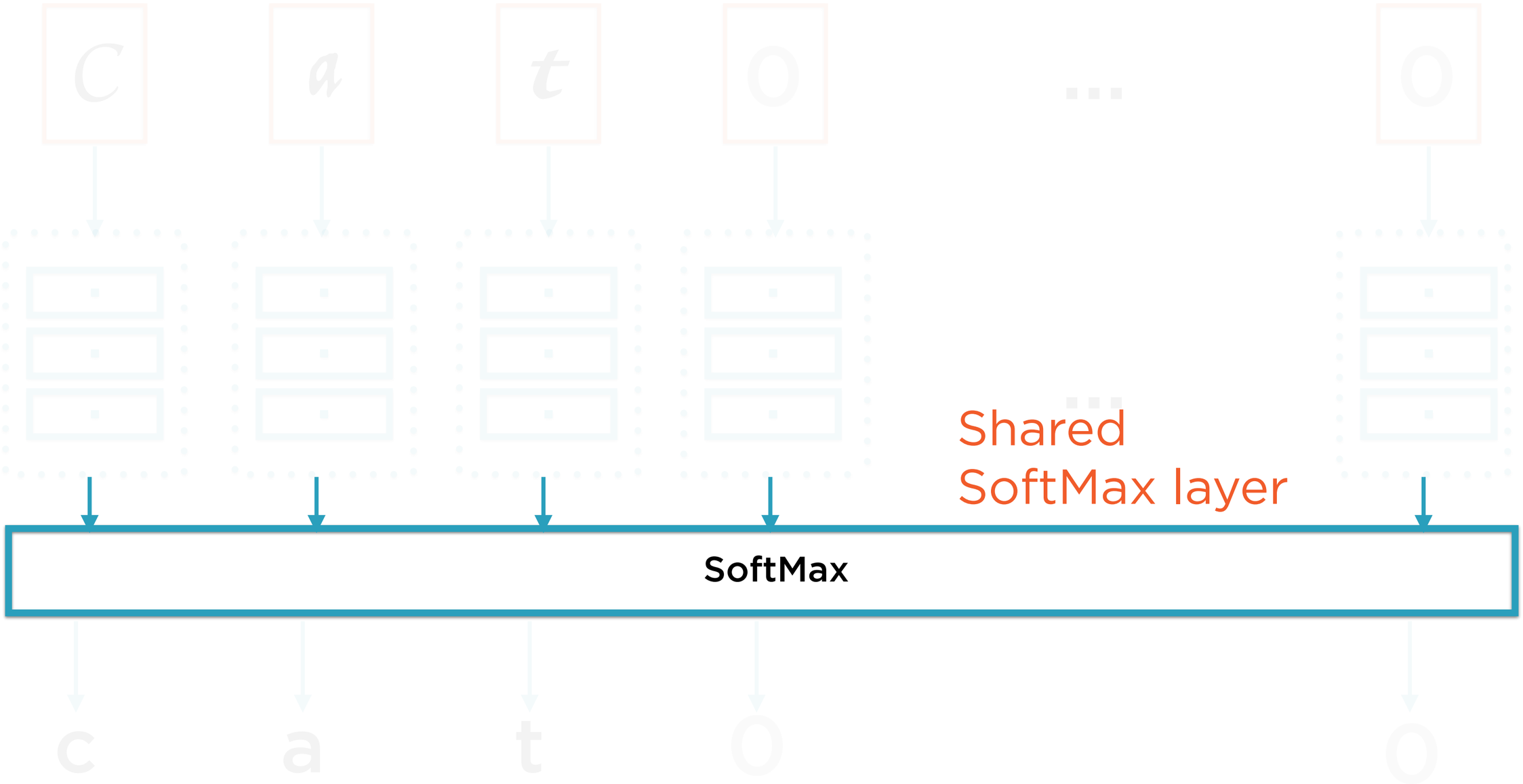
RNN Architecture



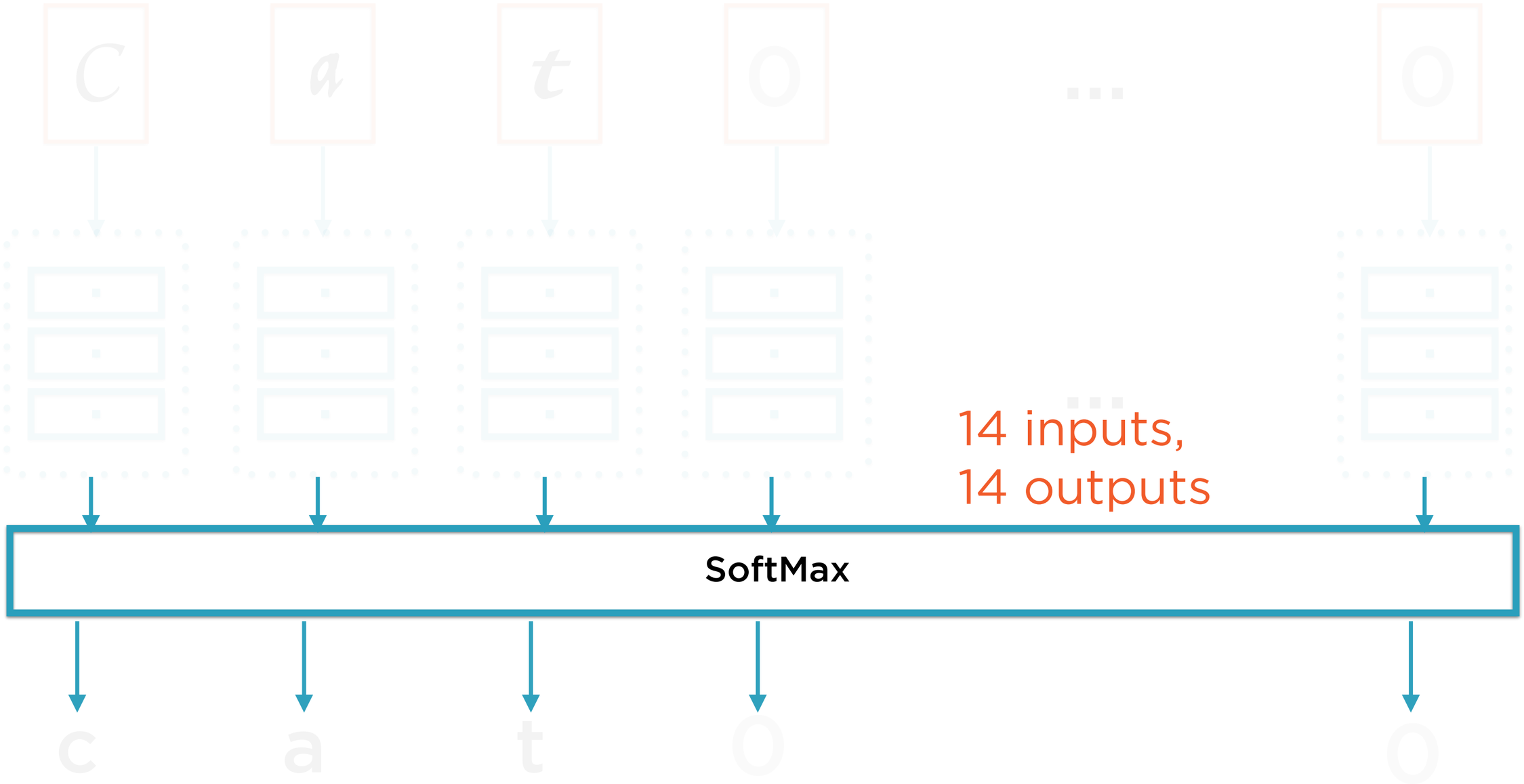
RNN Architecture



RNN Architecture

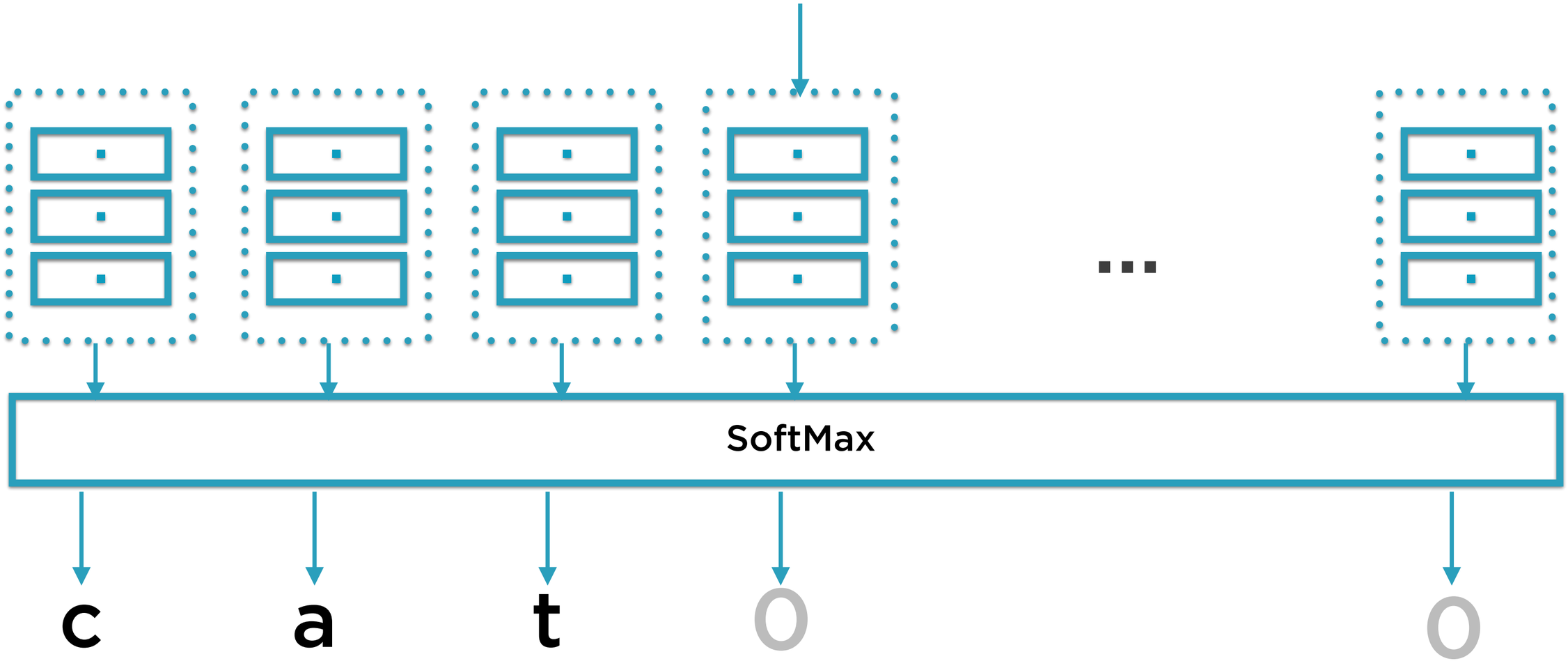


RNN Architecture



Input Tensor for Training

[batch_size, 14, 128]



Input Tensor for Training

[batch_size, 14, 128]

[[[ ], [ ], [ ] …]]

[ ], [ ], [ ] …]

…

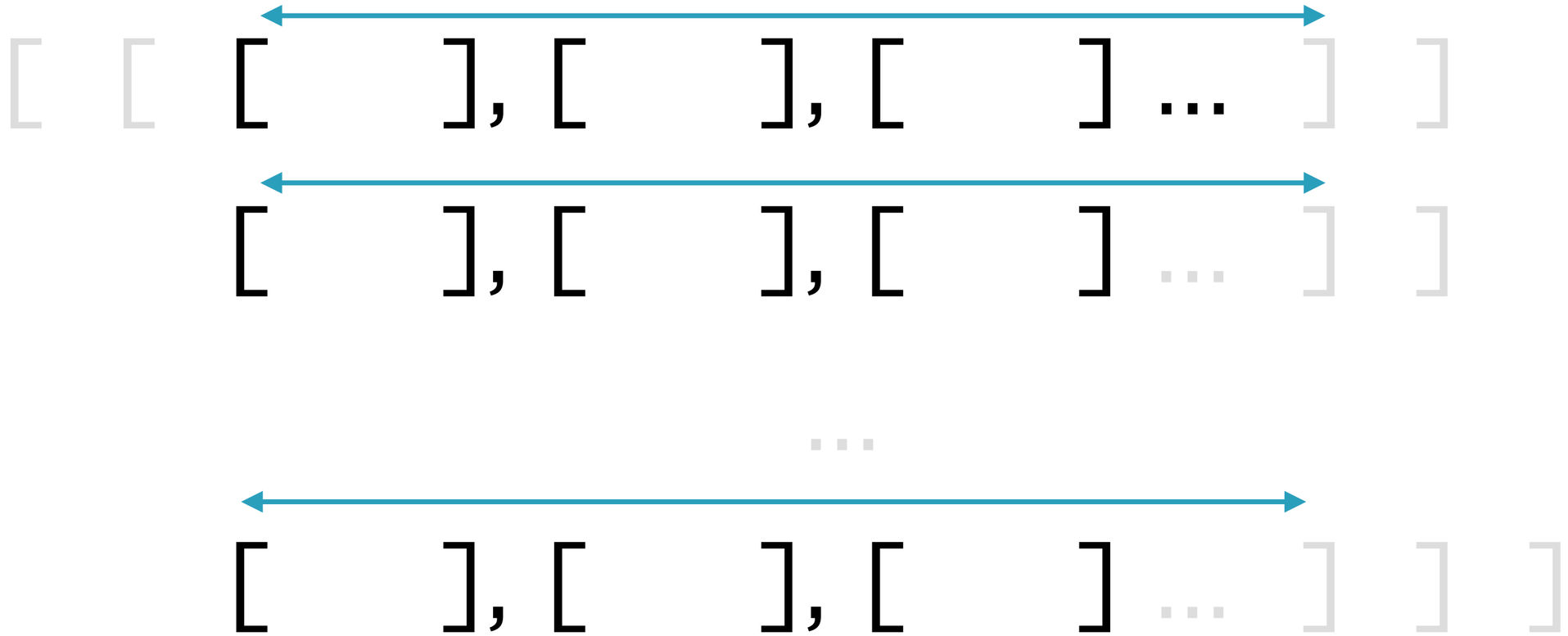
[ ], [ ], [ ] …]]]

Input Tensor for Training

[batch_size, 14, 128]

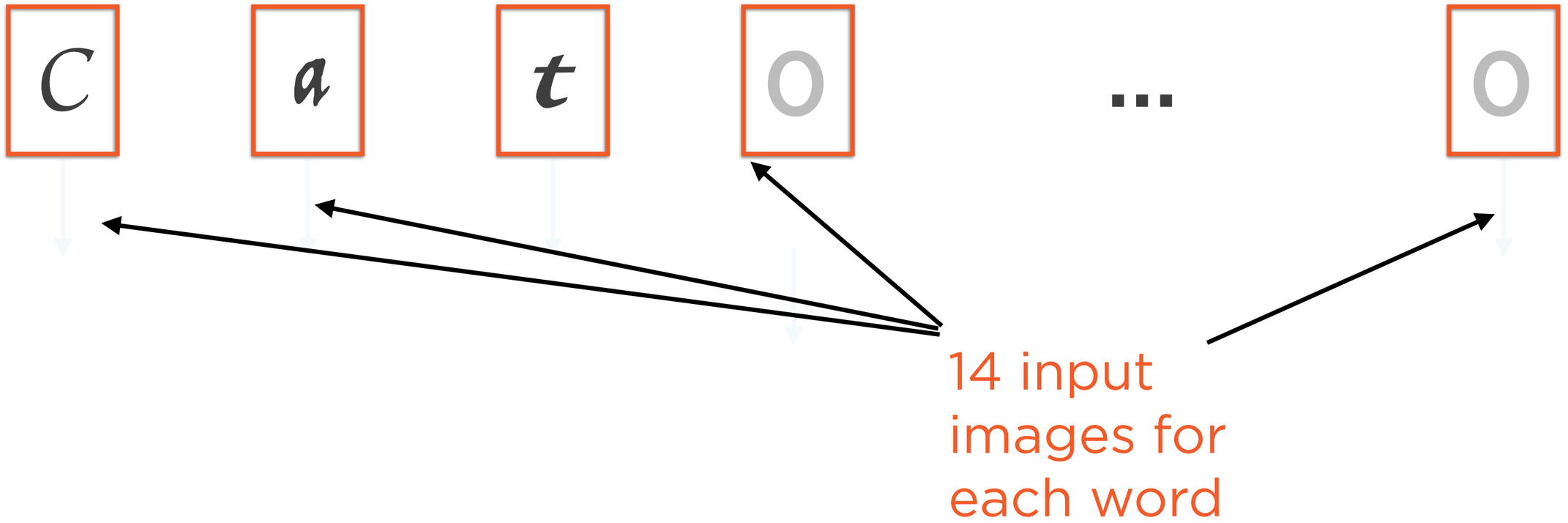
Input Tensor for Training

[batch_size, 14, 128]



Input Tensor for Training

[batch_size, 14, 128]



Input Tensor for Training

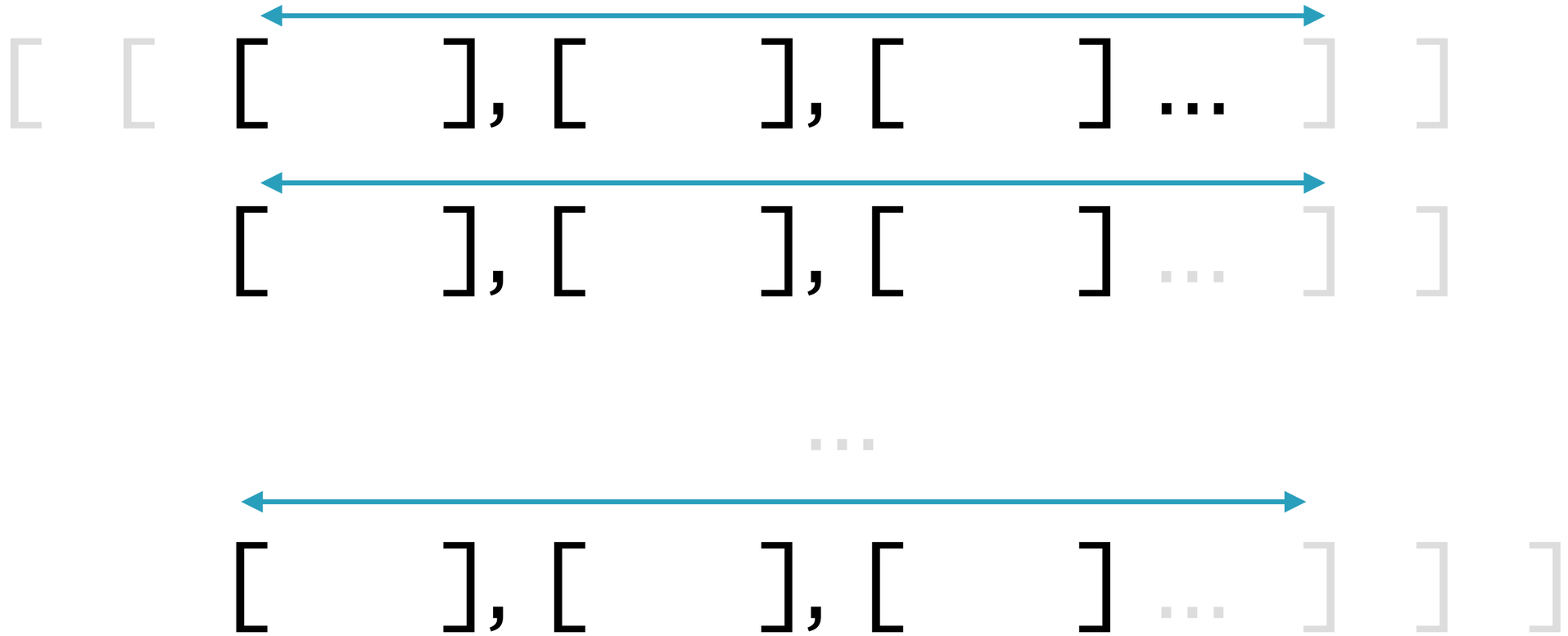
[batch_size, 14, 128]

3 actual characters

11 characters of padding

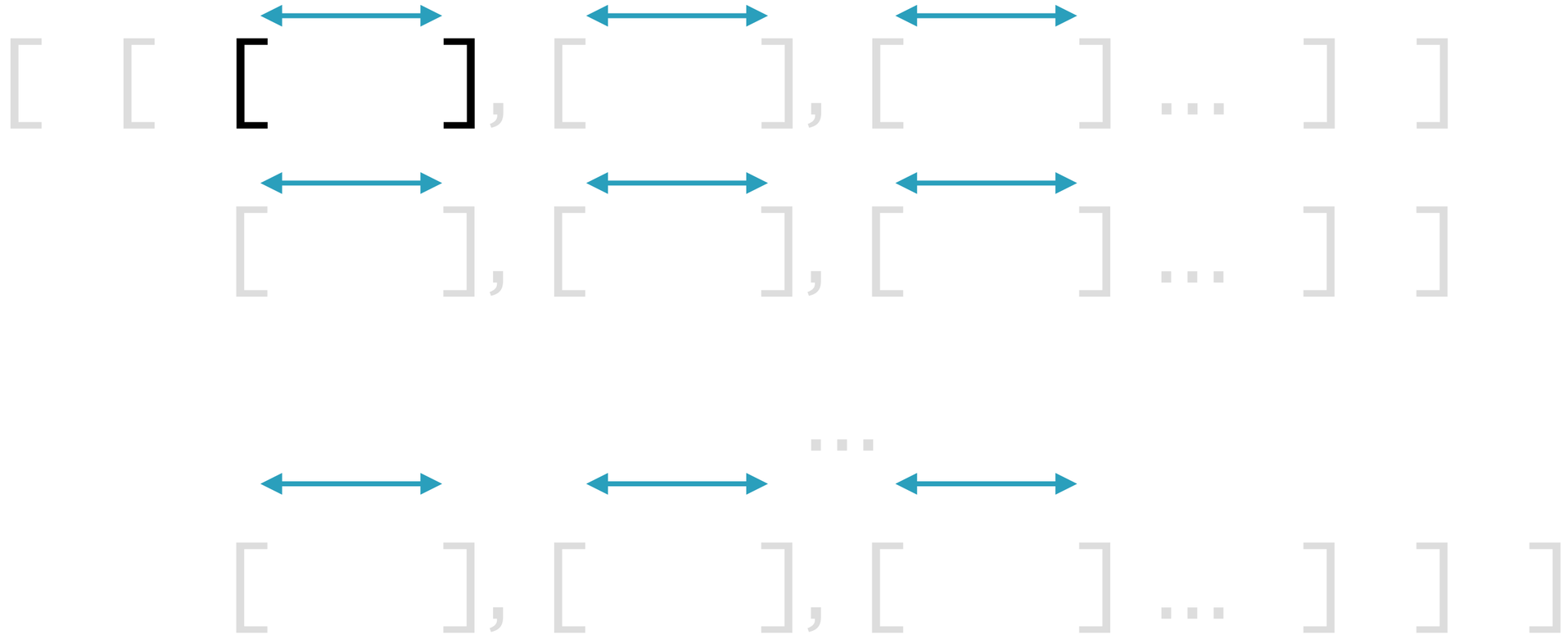
Input Tensor for Training

[batch_size, 14, 128]

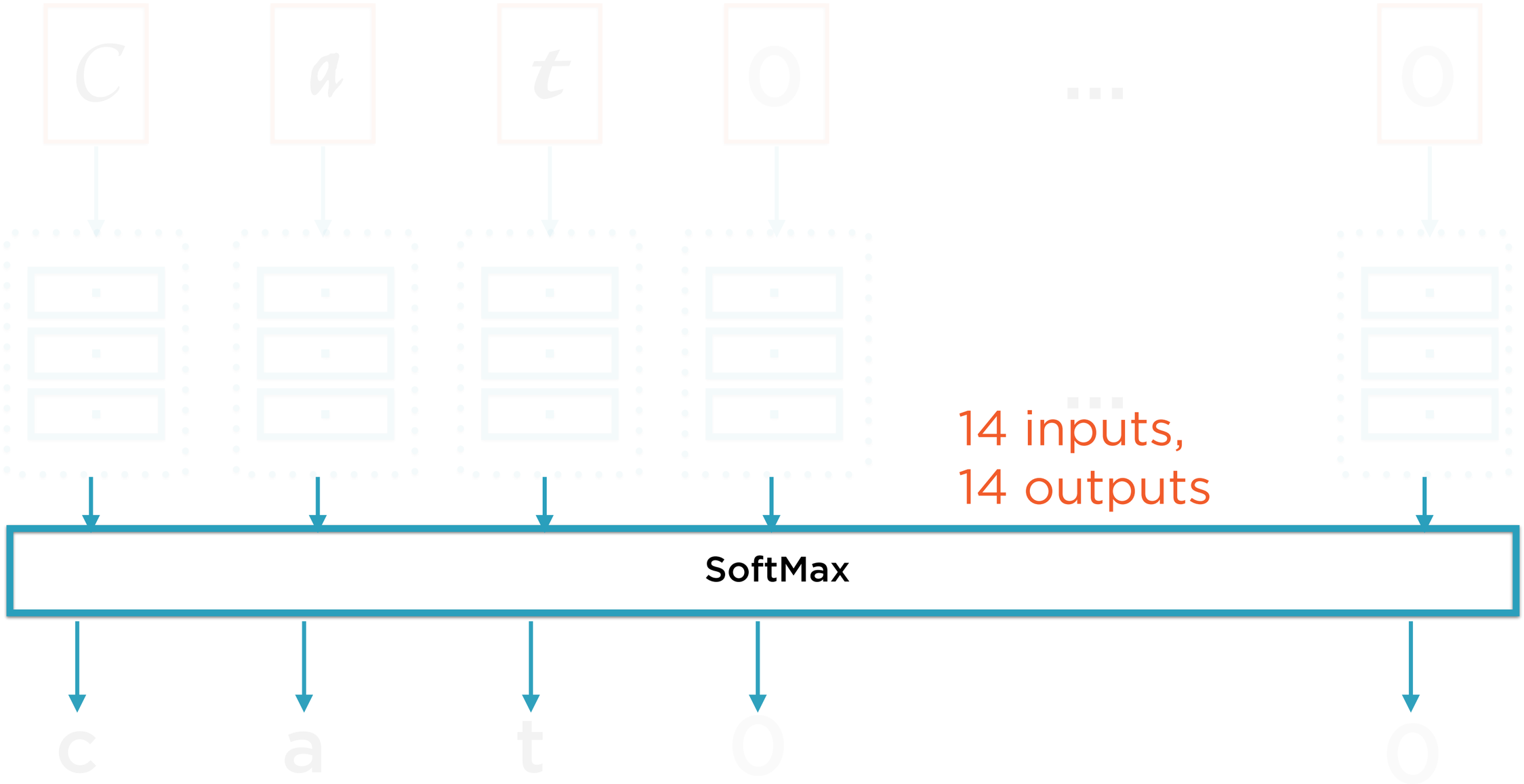


Input Tensor for Training

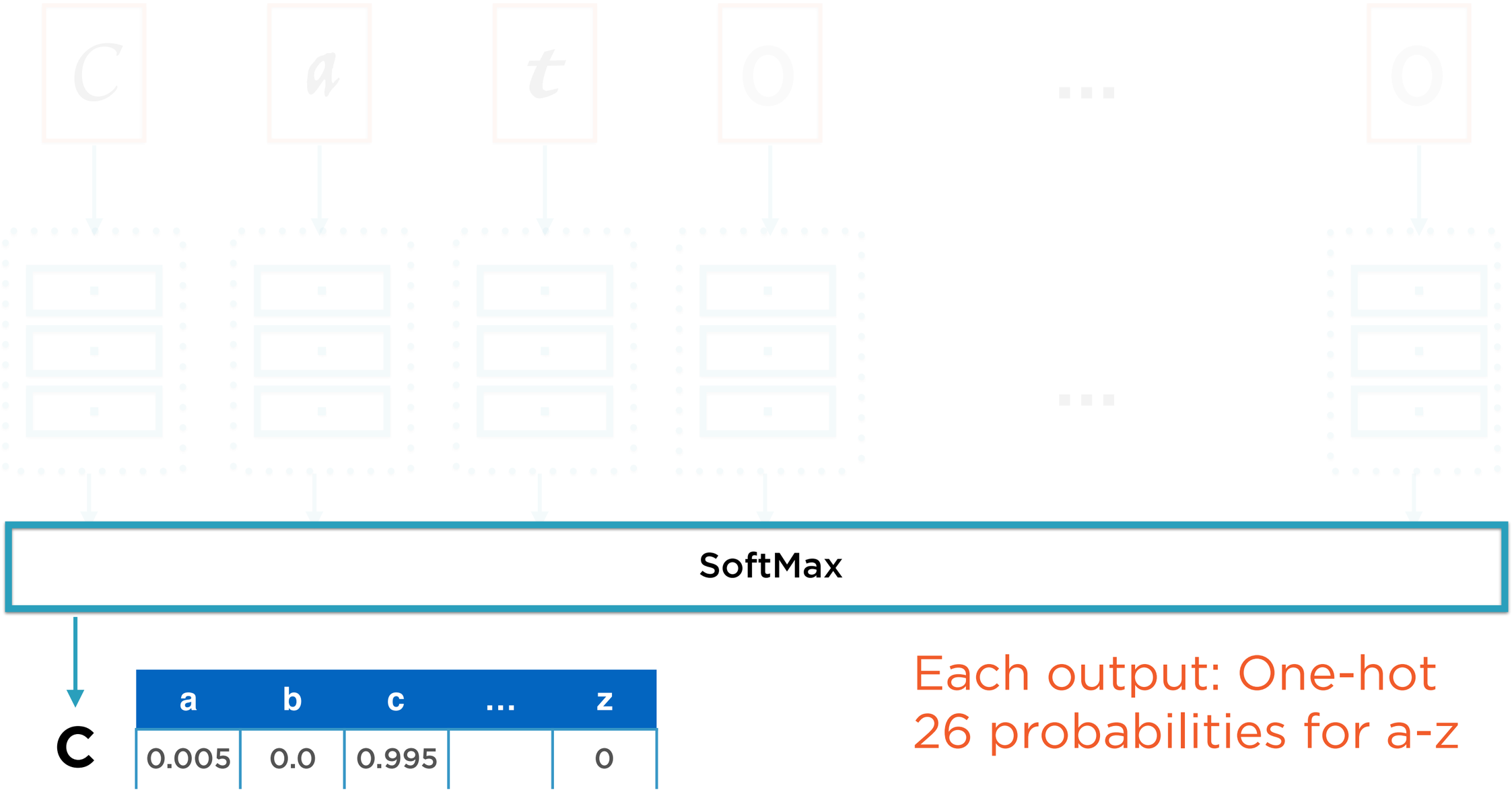
[batch_size, 14, 128]



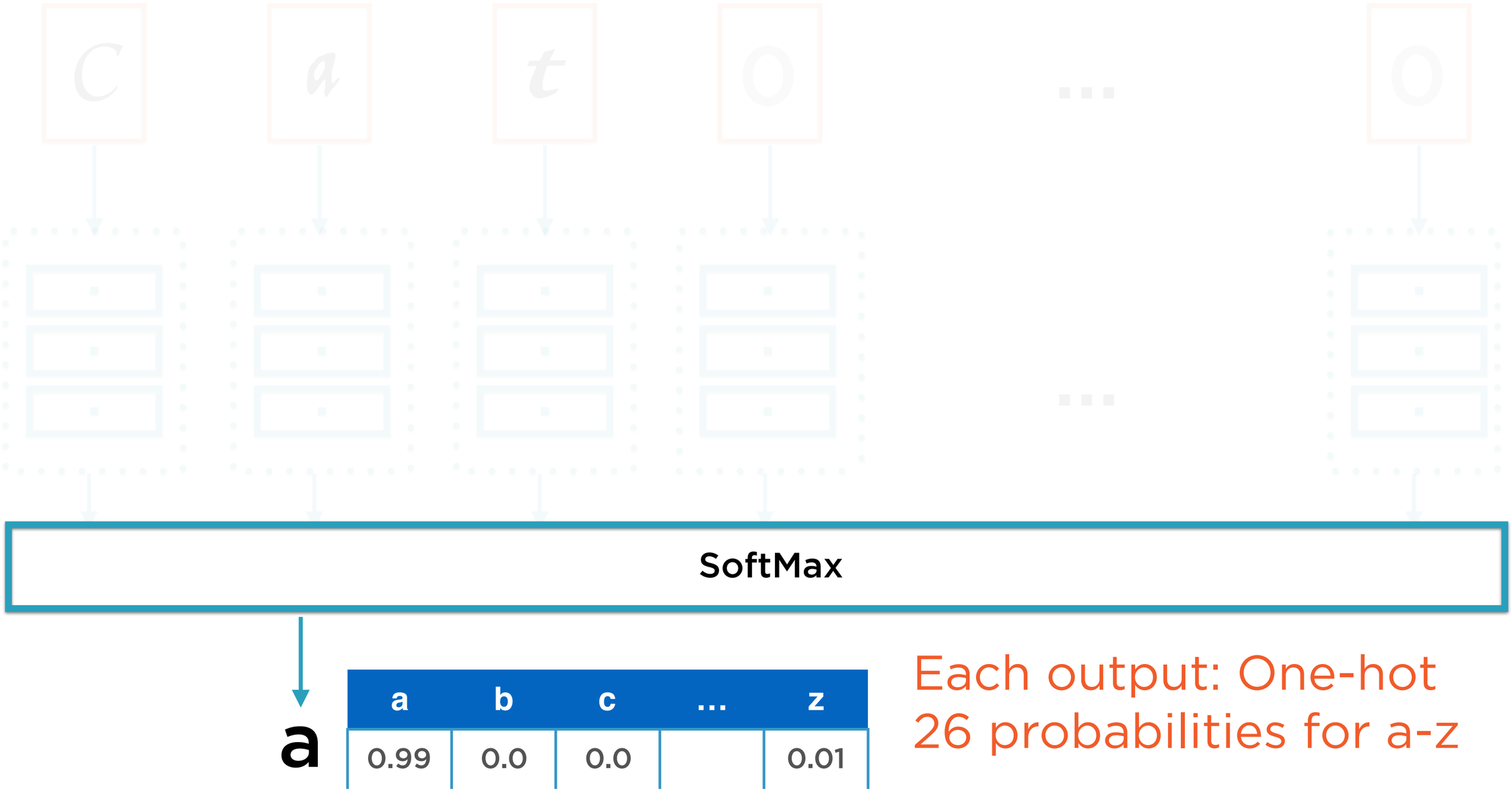
RNN Architecture



RNN Architecture

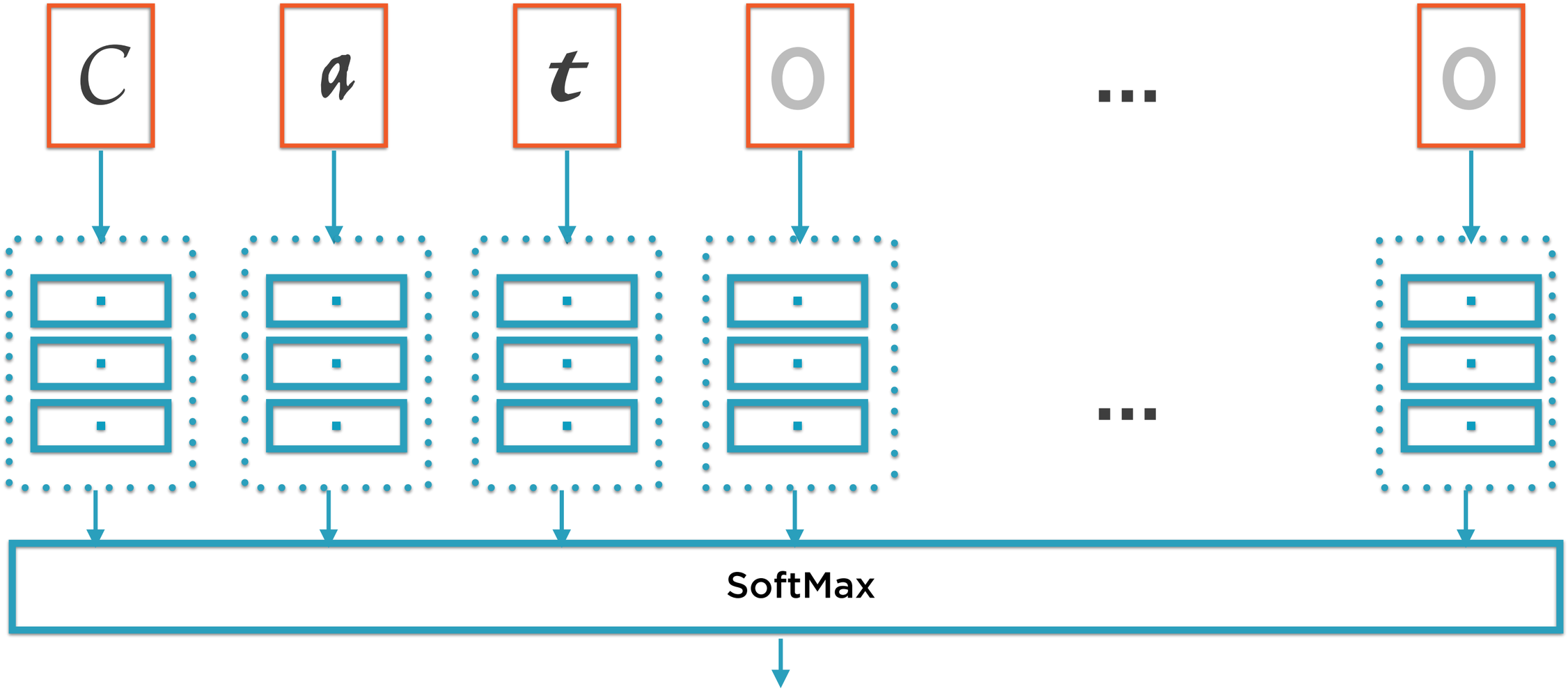


RNN Architecture



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

Output Tensor for Predicted Labels



[batch_size, 14, 26]

Output Tensor for Predicted Labels

[batch_size, 14, 26]

[[[ ], [ ], [ ] …]]

[ ], [ ], [ ] …]]

…

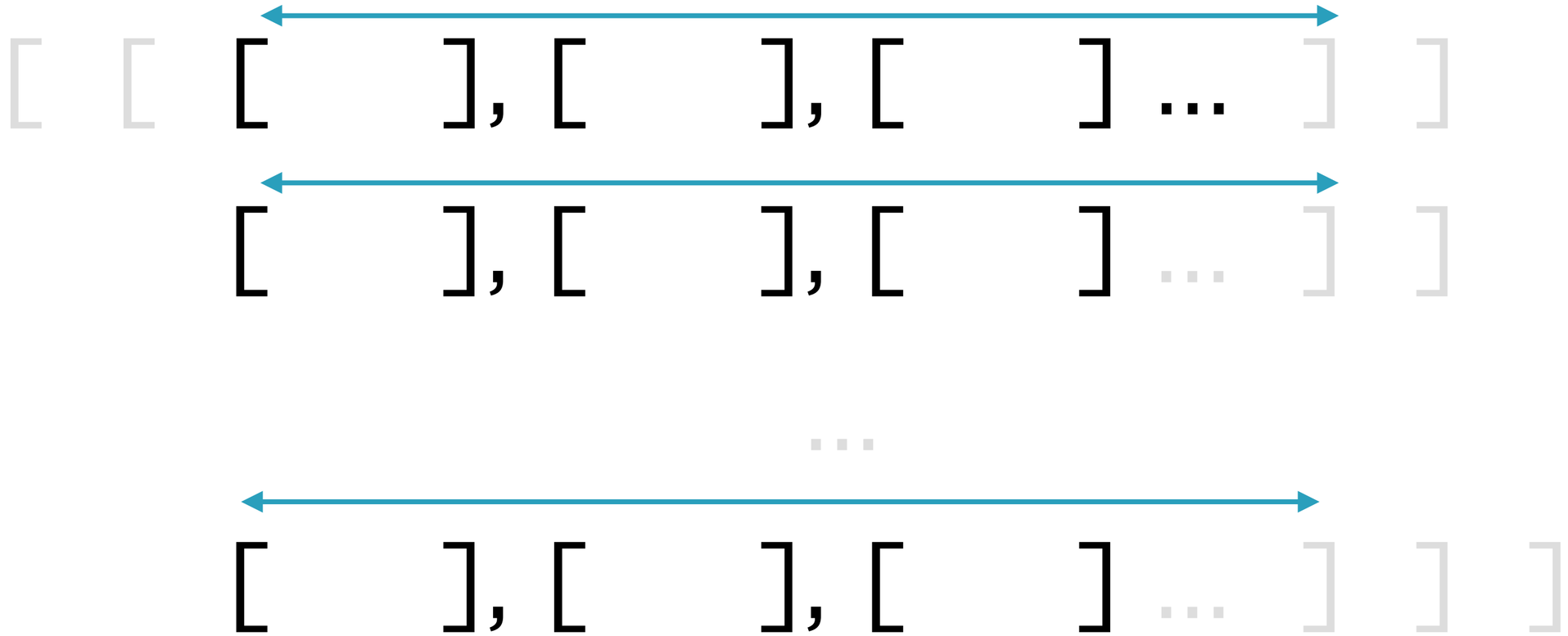
[ ], [ ], [ ] …]]]

Output Tensor for Predicted Labels

[batch_size, 14, 26]

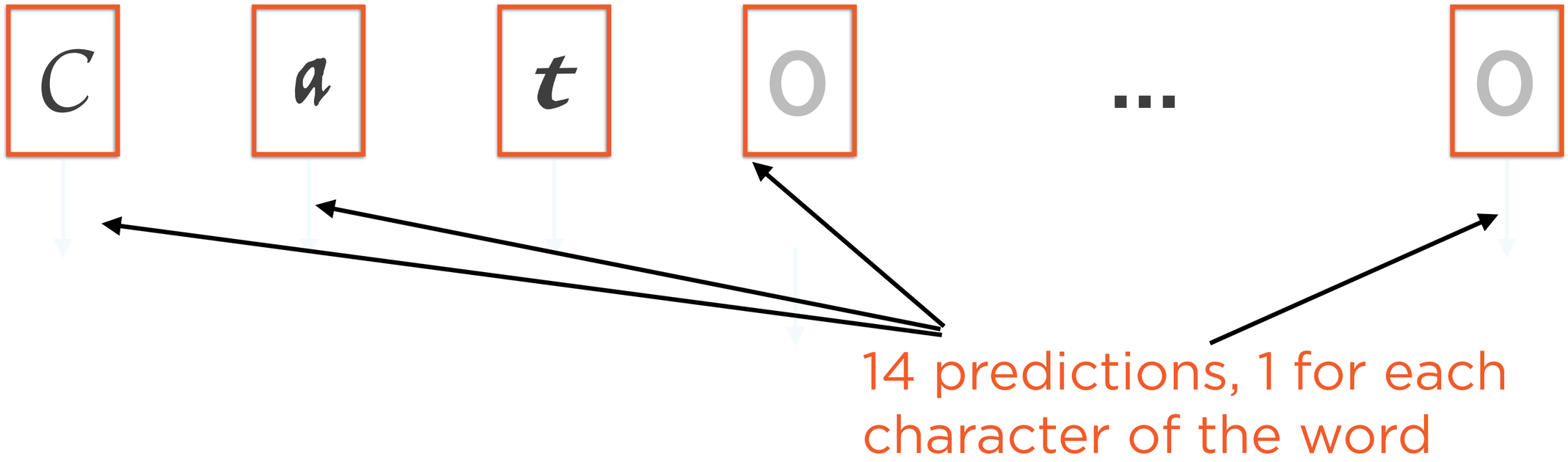
Output Tensor for Predicted Labels

[batch_size, 14, 26]



Output Tensor for Predicted Labels

[batch_size, 14, 26]



Output Tensor for Predicted Labels

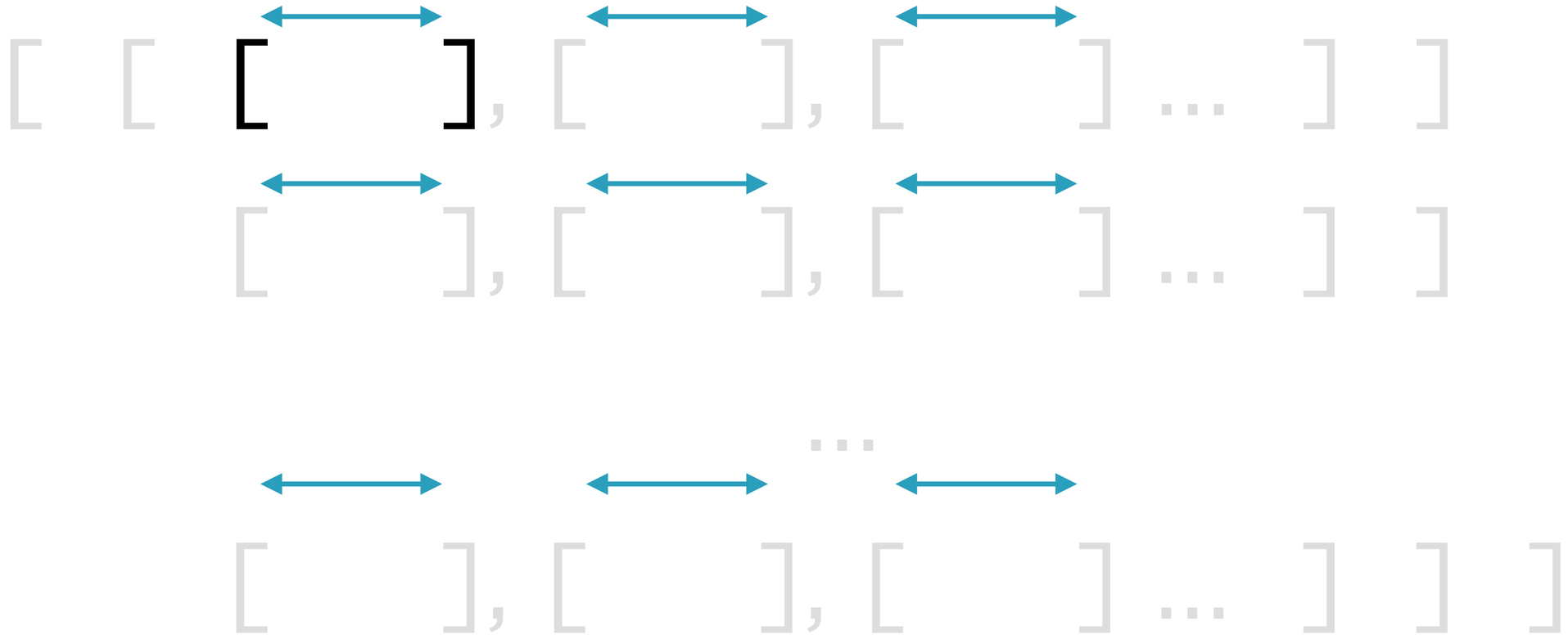
[batch_size, 14, 26]

3 predictions that we care about

11 predictions that we can ignore

Output Tensor for Predicted Labels

[batch_size, 14, 26]

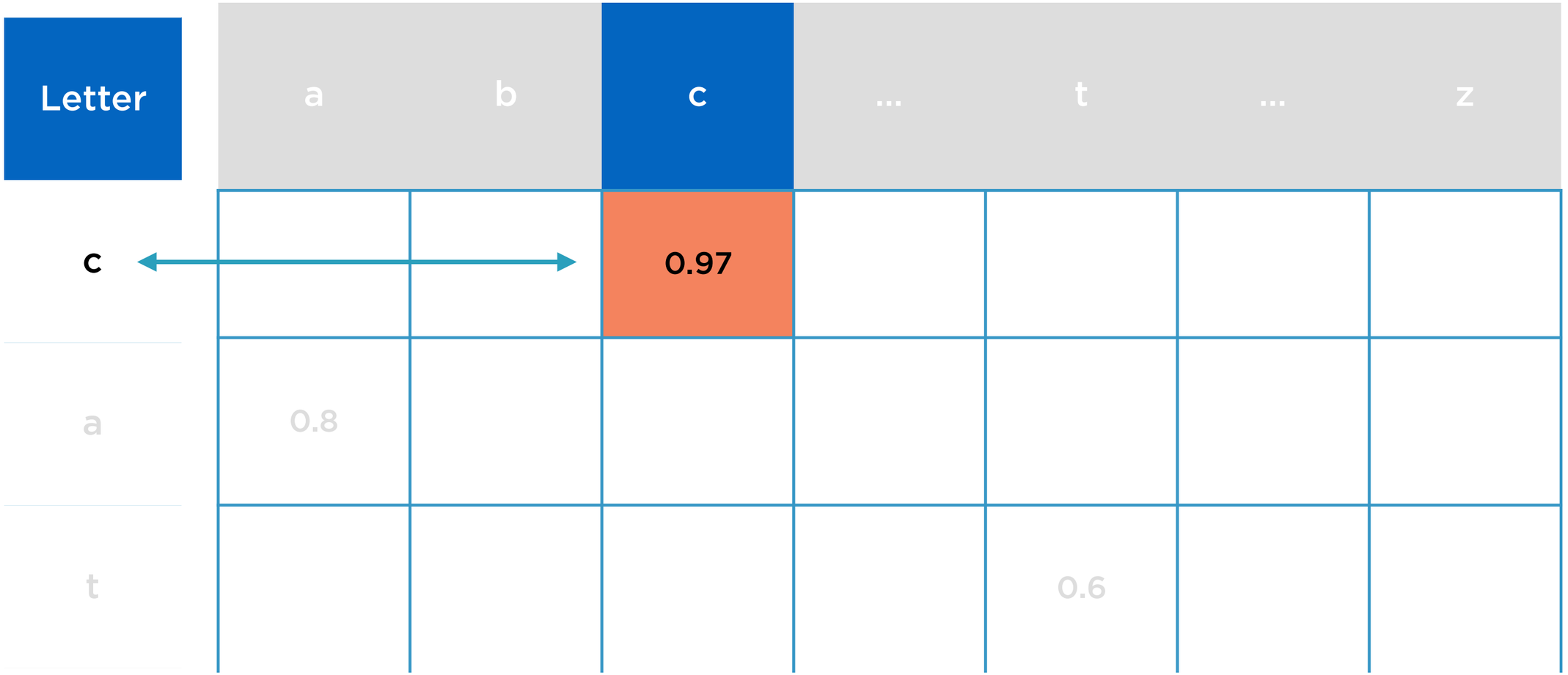


One-hot Encoded Predictions

Letter	a	b	c	...	t	...	z
c			0.97				
a	0.8						
t					0.6		

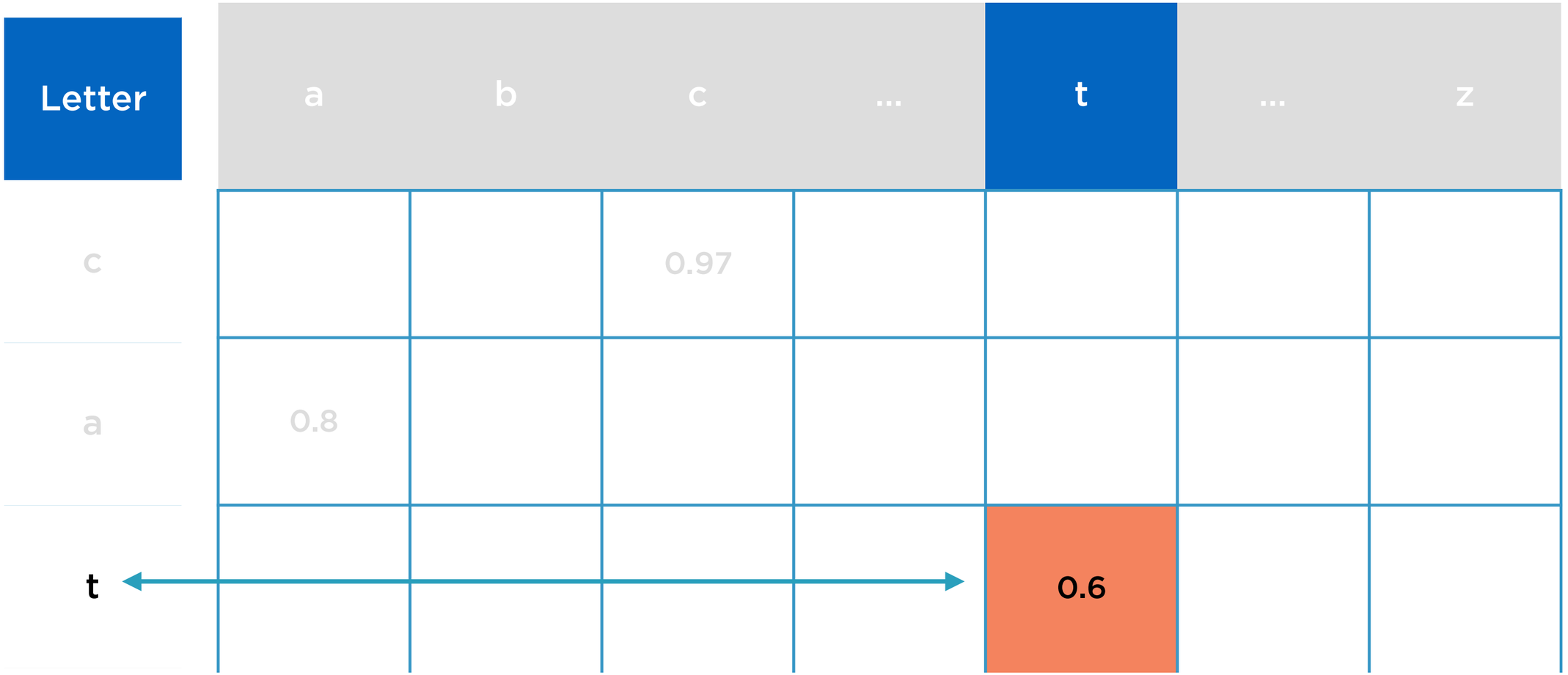
26 elements

One-hot Encoded Predictions

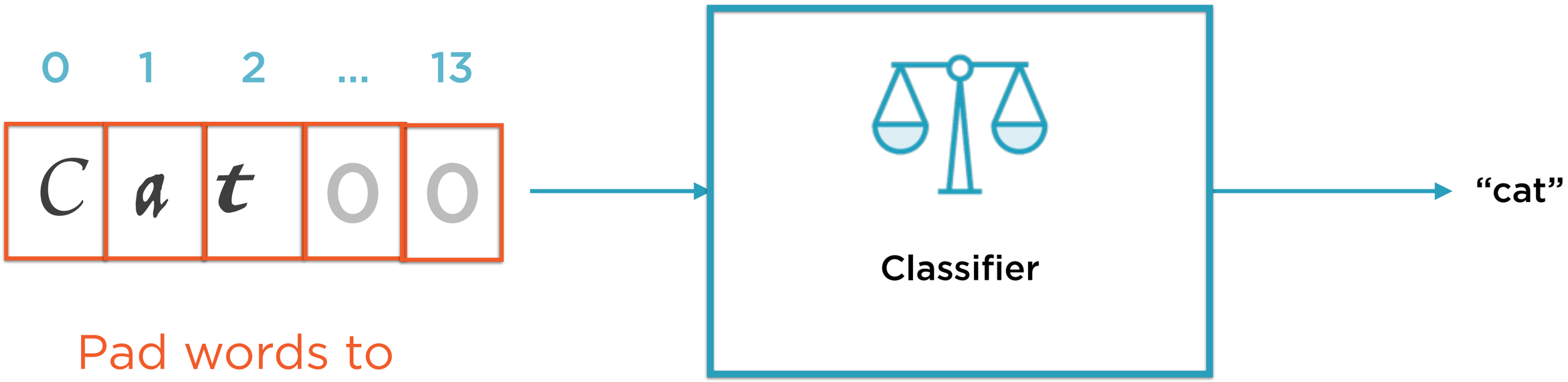


One-hot Encoded Predictions

One-hot Encoded Predictions

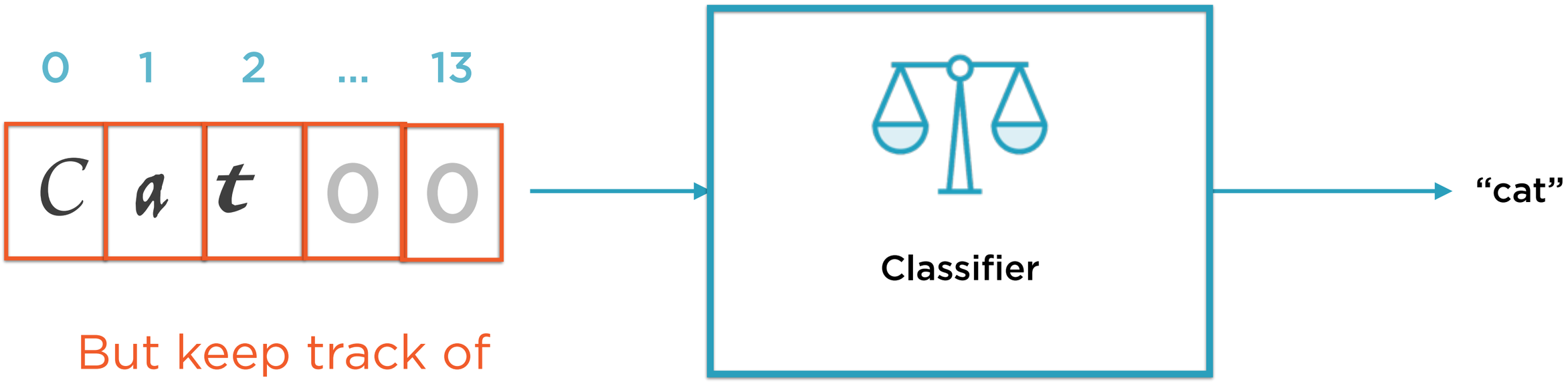


OCR Word Recognition



Pad words to all be of 14 characters

OCR Word Recognition



But keep track of sequence lengths of each word

Input Tensor for Training

[batch_size, 14, 128]

[[[ ], [ ], [ ] …]]

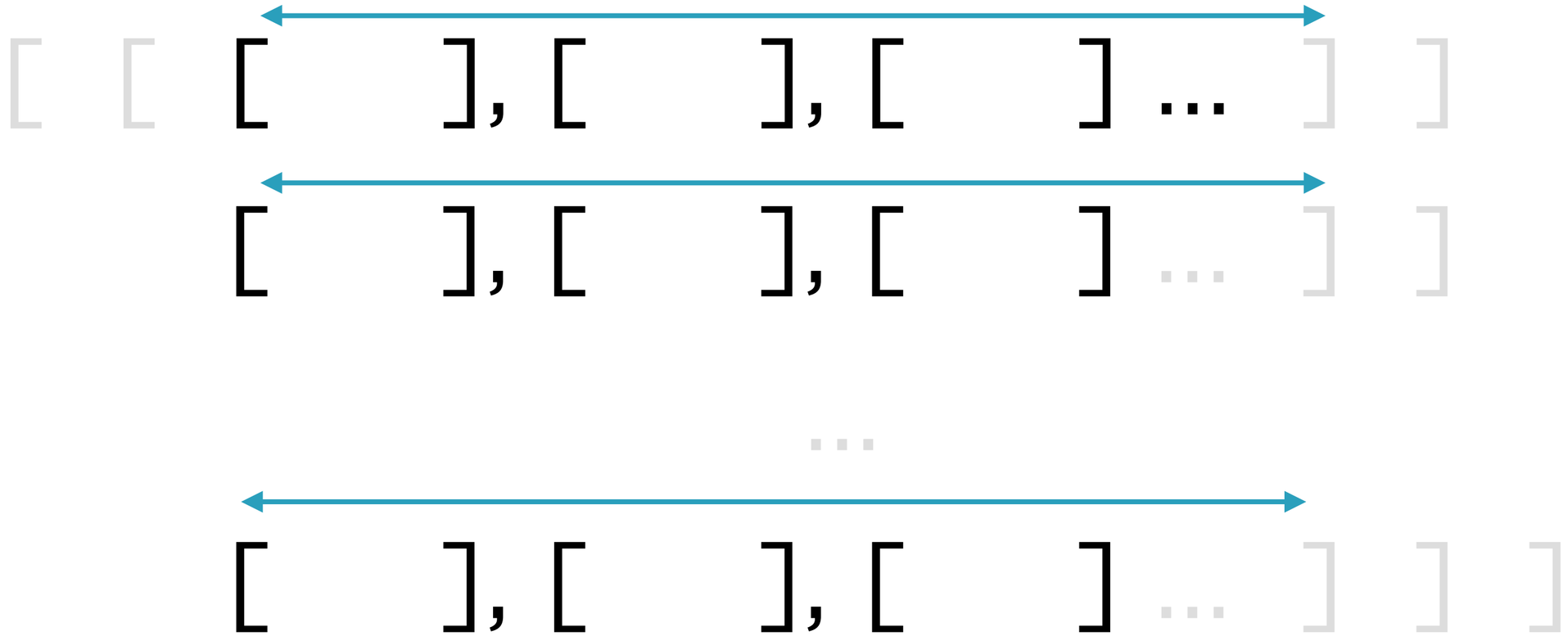
[ ], [ ], [ ] …]]

…

[ ], [ ], [ ] …]]]

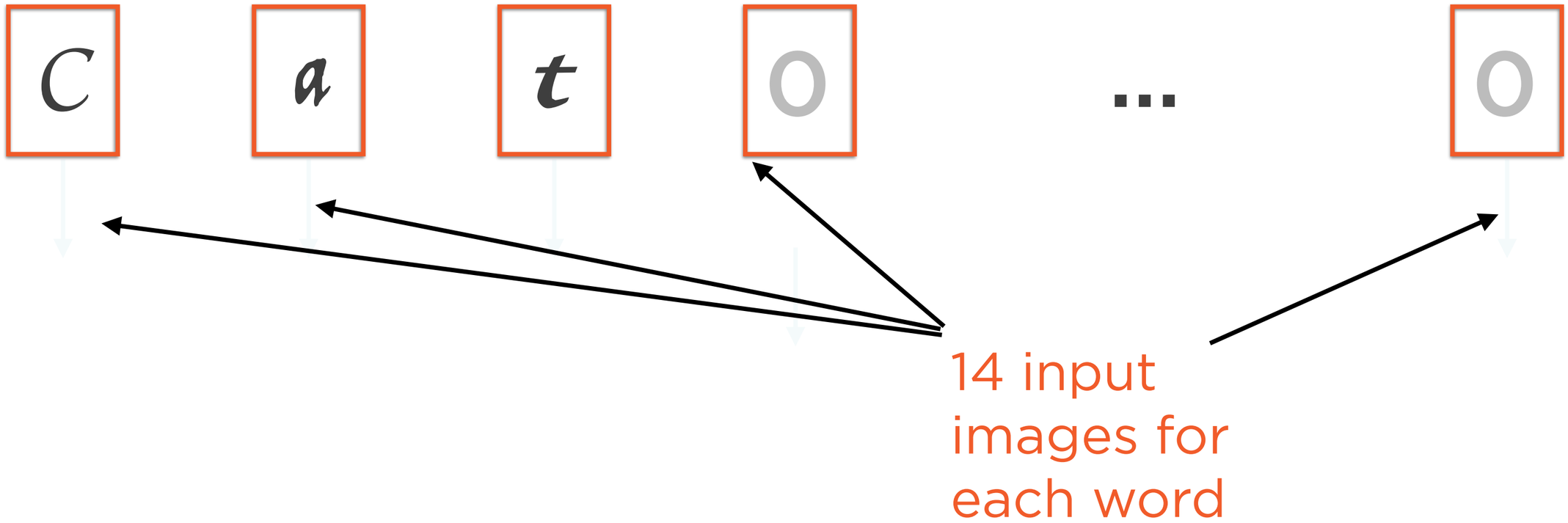
Input Tensor for Training

[batch_size, 14, 128]



Input Tensor for Training

[batch_size, 14, 128]



Input Tensor for Training

[batch_size, 14, 128]

3 actual characters

11 characters of padding

```
used = tf.sign(tf.reduce_max(tf.abs(X),
reduction_indices=2))
```

```
length = tf.reduce_sum(used,
reduction_indices=1)
```

```
sequence_length = tf.cast(length, tf.int32)
```

- Does the image contain any non-zero element?

```
used = tf.sign(tf.reduce_max(tf.abs(X),
reduction_indices=2))
```

```
length = tf.reduce_sum(used,
reduction_indices=1)
```

```
sequence_length = tf.cast(length, tf.int32)
```

- Used contains a list of 14 elements for each word

  e.g. ‘cat’ has 3 non zero elements

  [1 , 1 , 1 , 0 … 0]

```
used = tf.sign(tf.reduce_max(tf.abs(X),
reduction_indices=2))
```

```
length = tf.reduce_sum(used,
reduction_indices=1)
```

```
sequence_length = tf.cast(length, tf.int32)
```

- Sum them

  e.g. ‘cat’ length 3

```
used = tf.sign(tf.reduce_max(tf.abs(X),
reduction_indices=2))
```

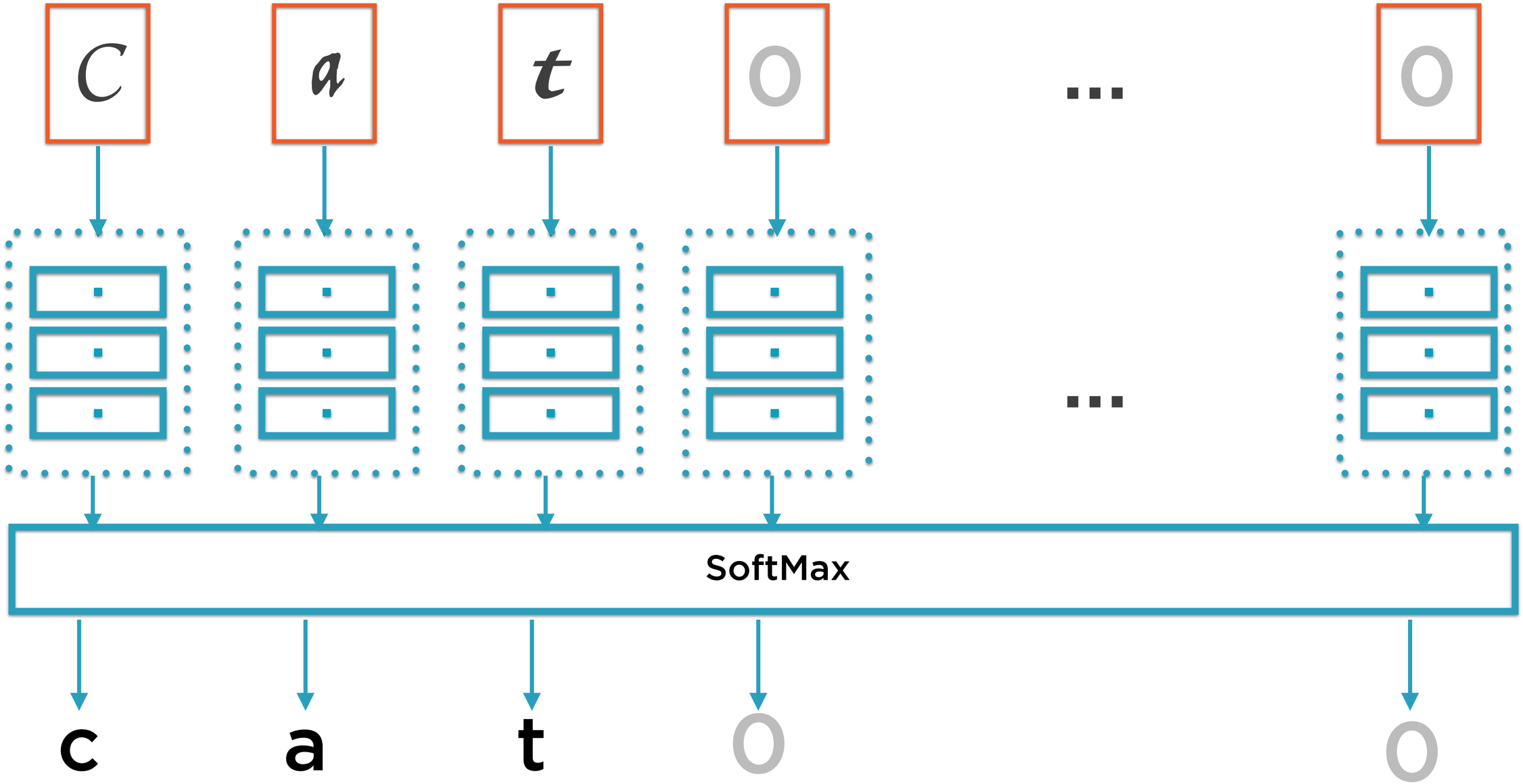
```
length = tf.reduce_sum(used,
reduction_indices=1)
```

```
sequence_length = tf.cast(length, tf.int32)
```

- That’s the sequence length

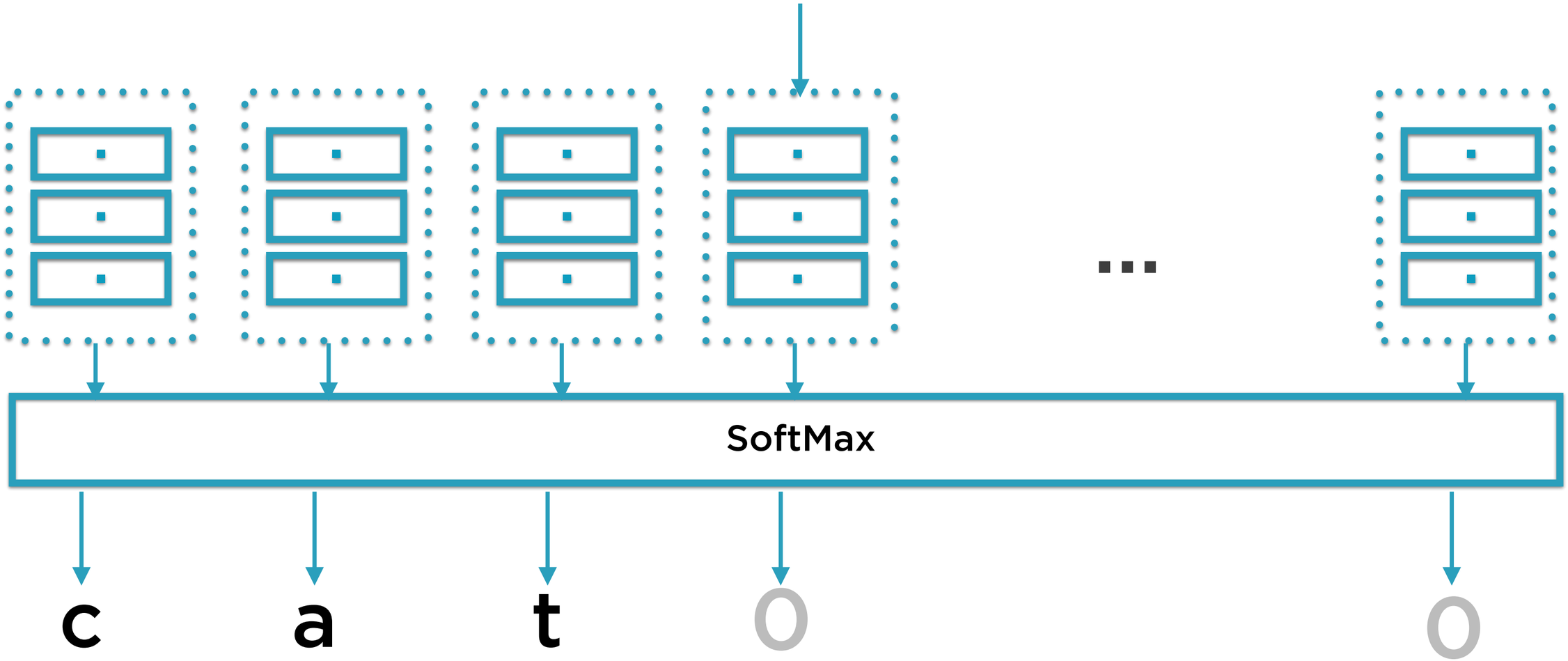
Accuracy and Error Calculations

RNN Architecture

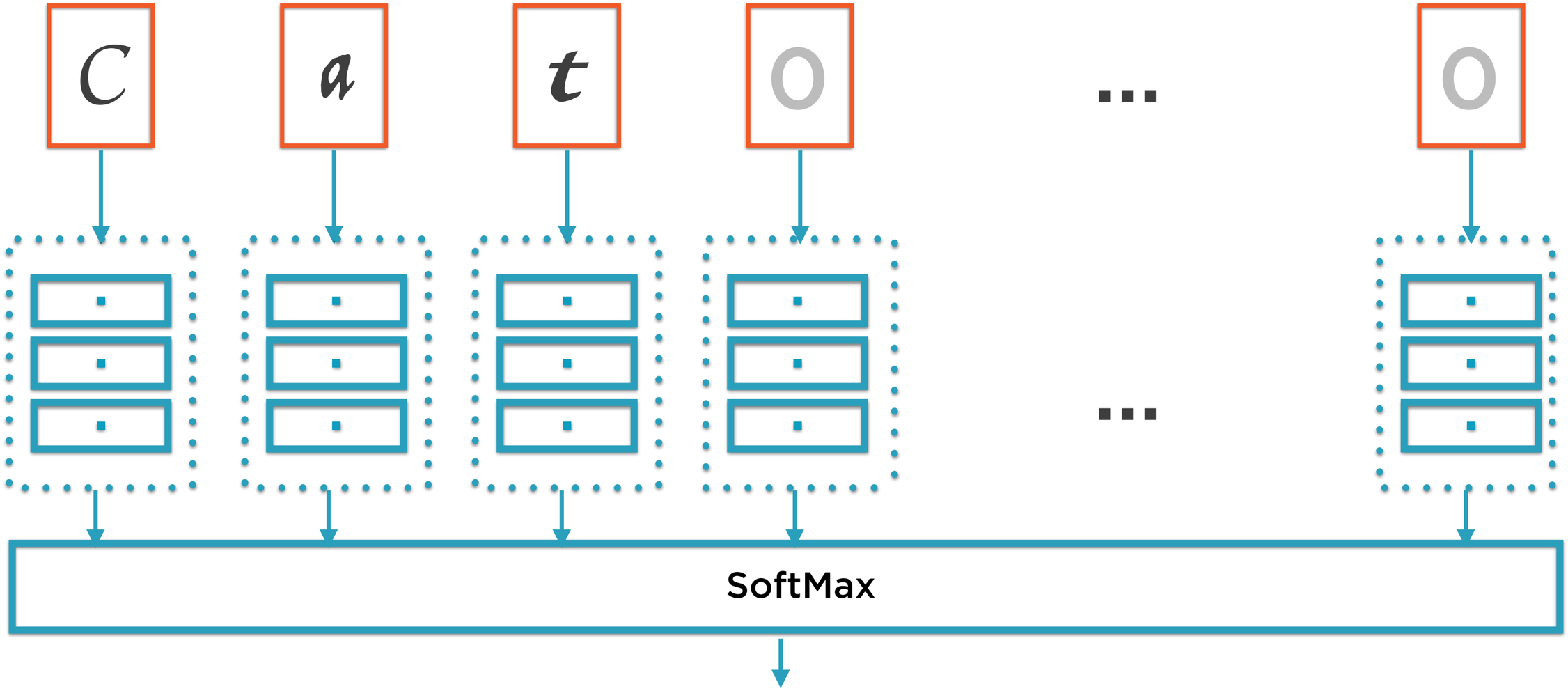


Input Tensor for Training

[batch_size, 14, 128]



Output Tensor for Predicted Labels



[batch_size, 14, 26]

Output Tensor for Predicted Labels

[batch_size, 14, 26]

[[[ ], [ ], [ ] …]]

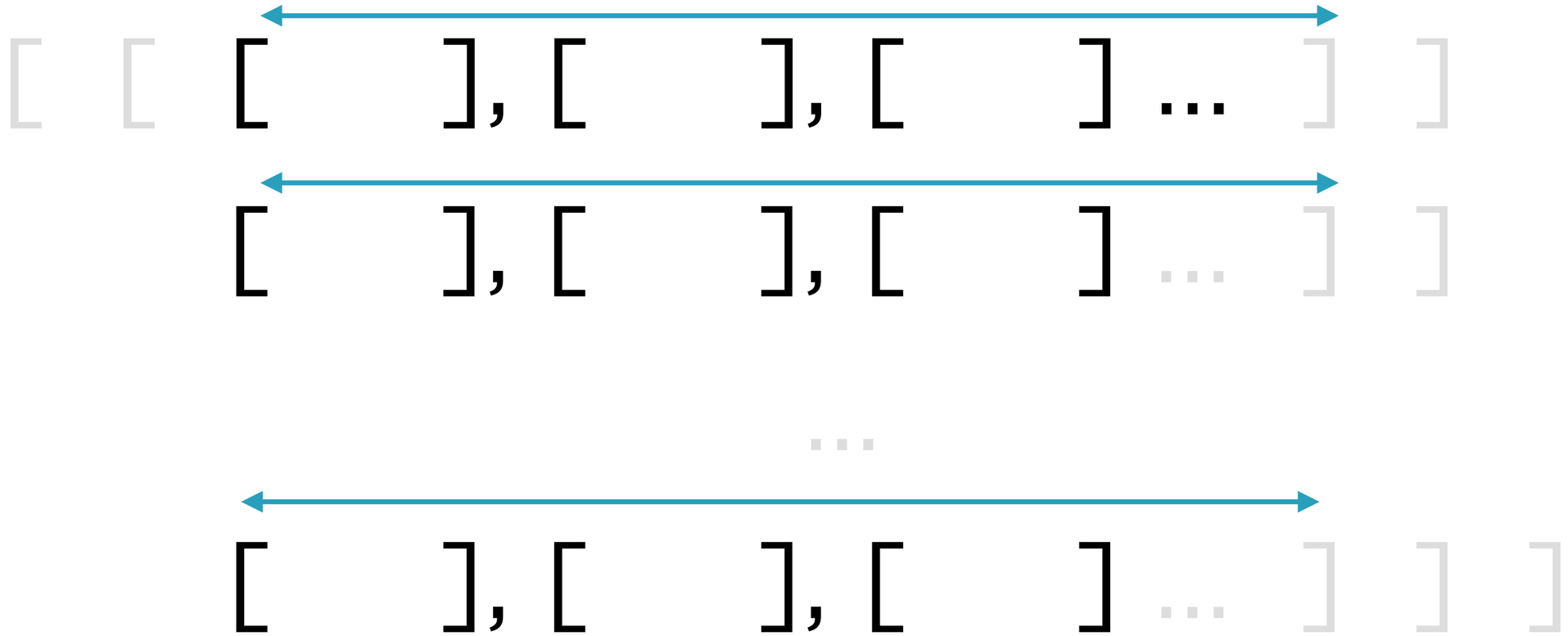
[ ], [ ], [ ] …]]

…

[ ], [ ], [ ] …]]]

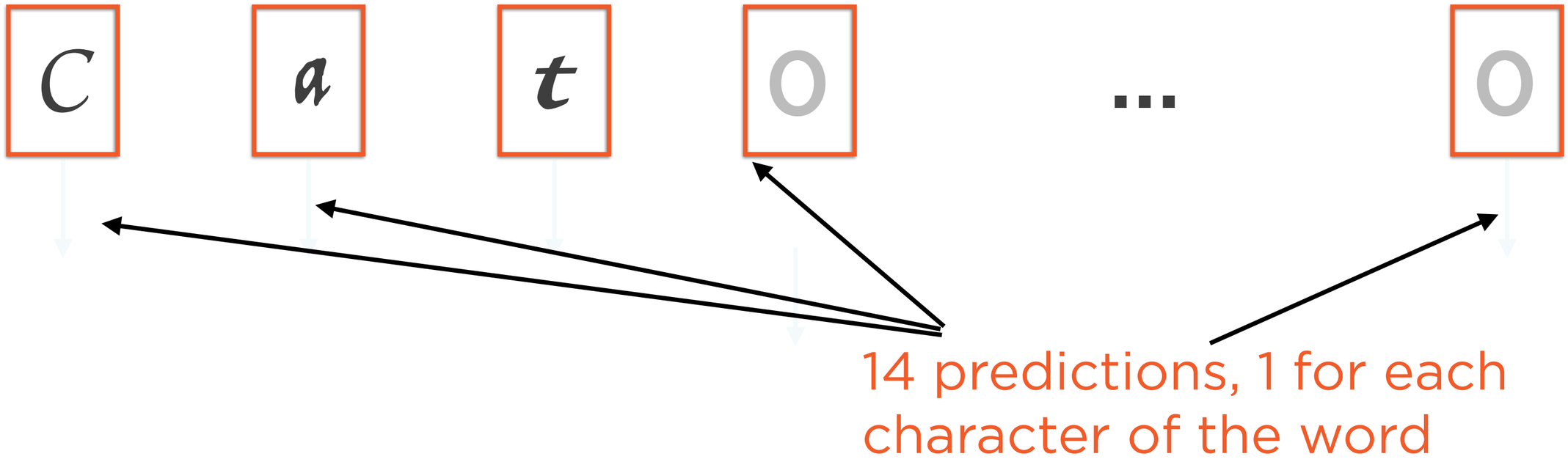
Output Tensor for Predicted Labels

[batch_size, 14, 26]



Output Tensor for Predicted Labels

[batch_size, 14, 26]



Output Tensor for Predicted Labels

[batch_size, 14, 26]

3 predictions that we care about

11 predictions that we can ignore

Output Tensor for Predicted Labels

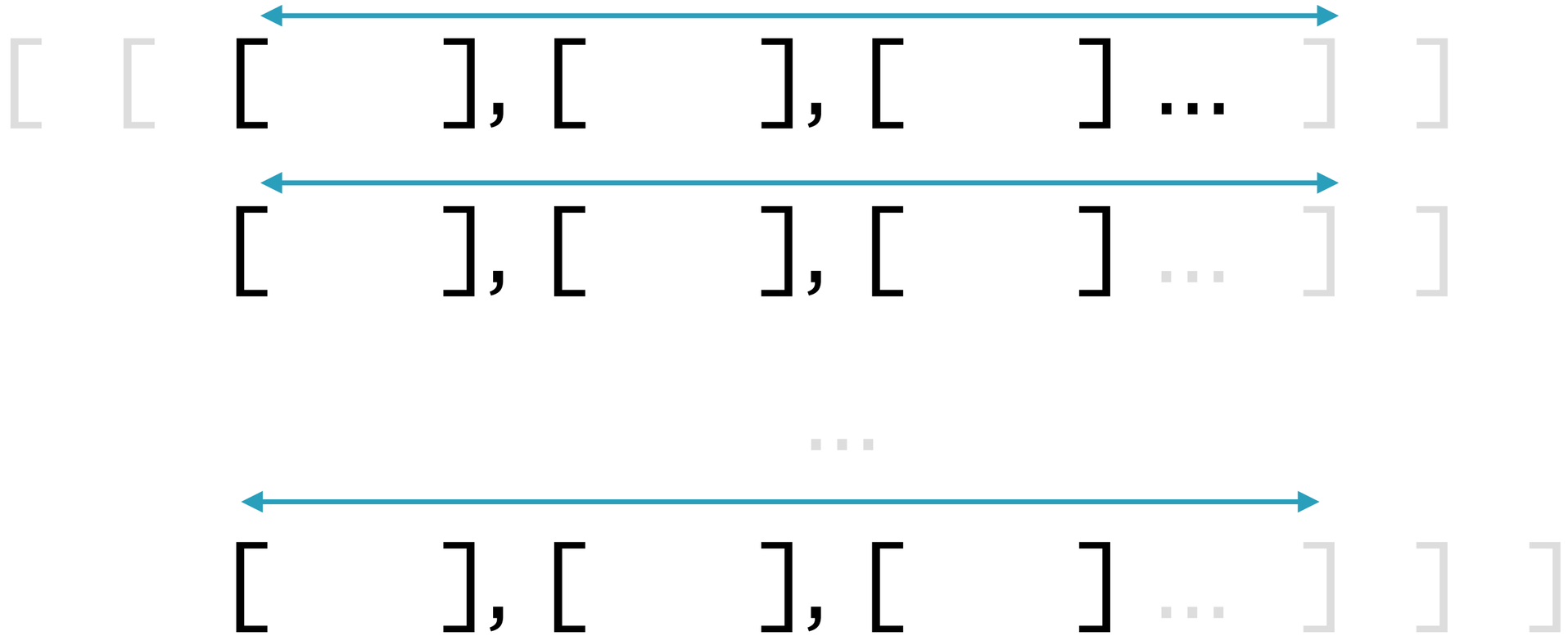
[batch_size, 14, 26]

Compare actual and predicted labels to calculate accuracy

Mask out while calculating accuracy

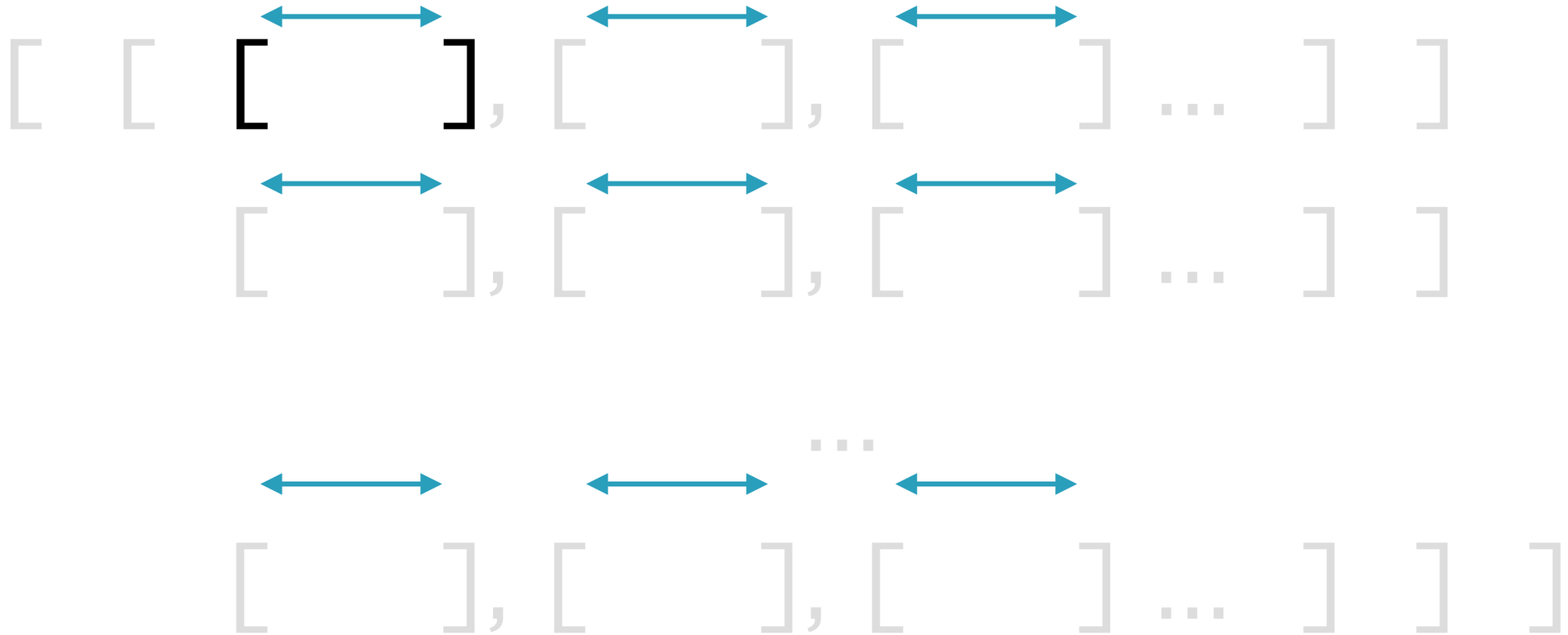
Output Tensor for Predicted Labels

[batch_size, 14, 26]



Output Tensor for Predicted Labels

[batch_size, 14, 26]



One-hot Encoded Predictions

Letter	a	b	c	...	t	...	z
c			0.97				
a	0.8						
t					0.6		

26 elements

One-hot Encoded Labels

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

26 elements

```
mistakes = tf.not_equal(
            tf.argmax(y_actual, 2), tf.argmax(y_predicted, 2))
```

Total Mistakes

Index of largest element in actual label should match index of largest element in predicted label - if not, mistake

```
mistakes = tf.not_equal(
            tf.argmax(y_actual, 2), tf.argmax(y_predicted, 2))
```

Total Mistakes

Mistakes ~ list of 14 elements, each either 1 (mistake) or 0

```
mistakes *= mask
```

Exclude Mistakes on Padding

Ignore any mistakes on padding characters

```
mistakes *= mask
```

Exclude Mistakes on Padding

Mistakes ~ list of 14 elements, each either 1 (real character) or 0 (padding)

```
mask = tf.sign(tf.reduce_max(tf.abs(y), reduction_indices=2))
```

Calculate Mask from Actual Labels

For each character of original word, compute max over all 128 pixels in image

```
mistakes /= tf.cast(sequence_length, tf.float64)
error = tf.reduce_mean(mistakes)
```

Calculate Error

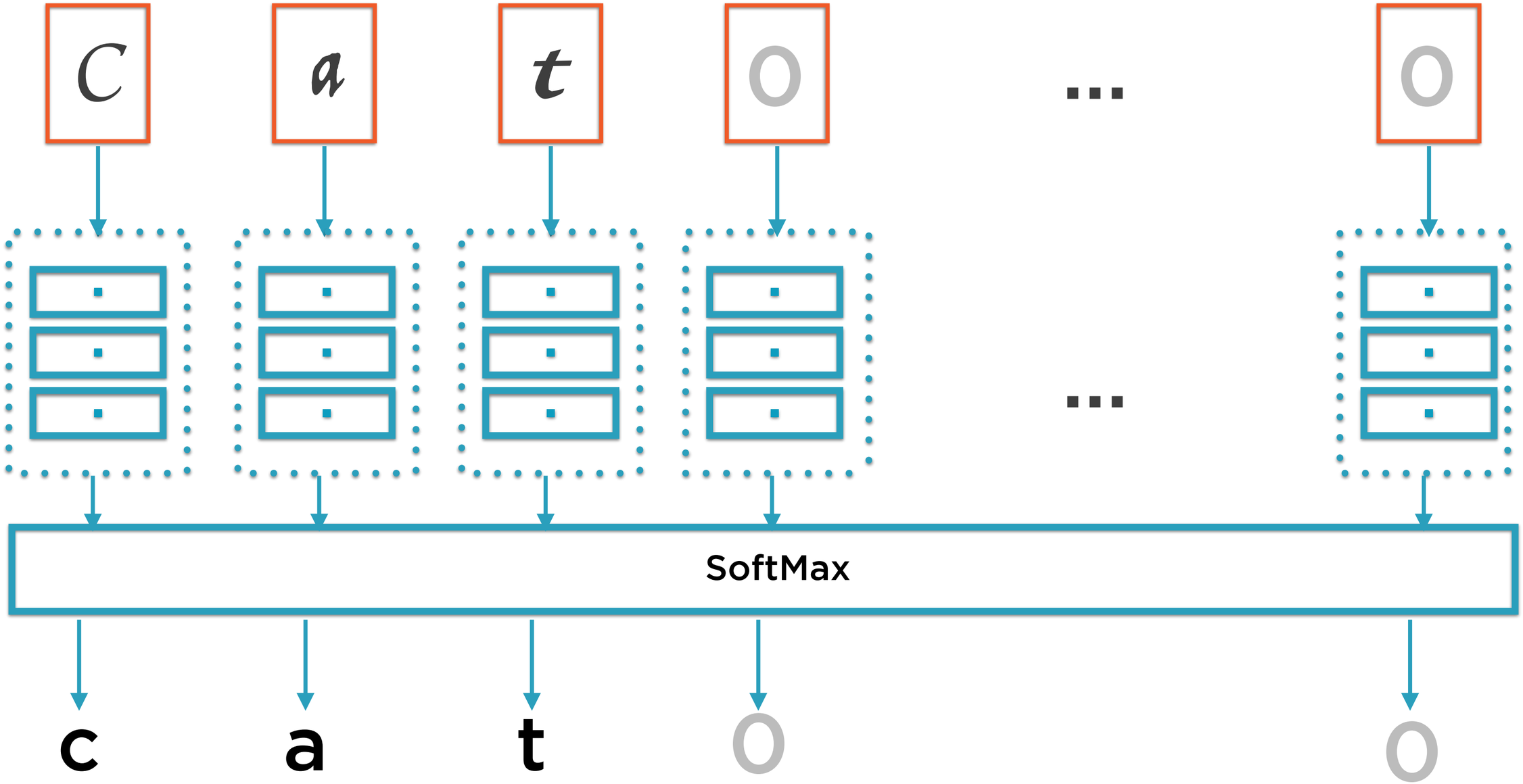
Find average number of mistakes per word

Demo

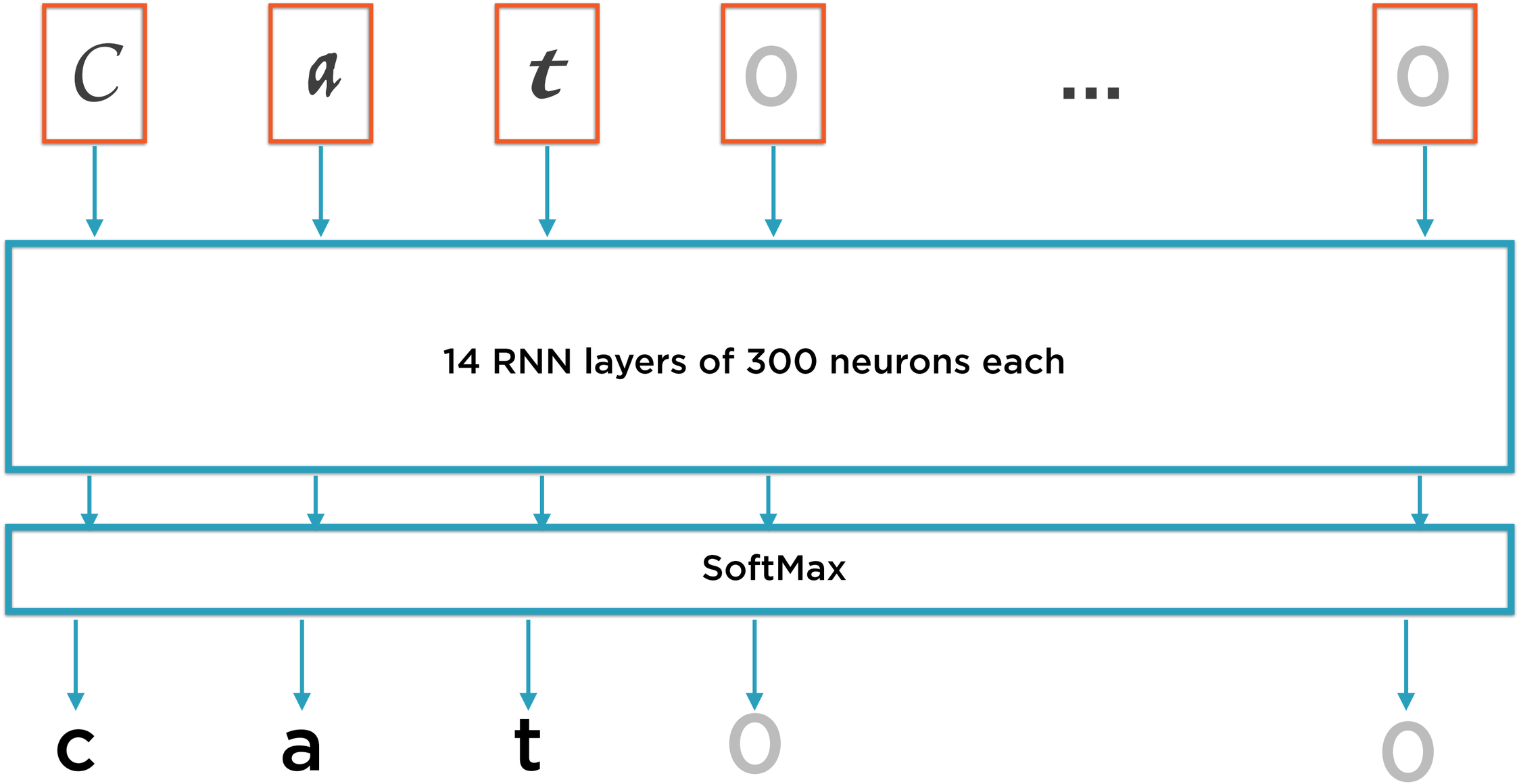
Perform optical character recognition using a bidirectional RNN on a modified version of the MIT OCR dataset

- Build the bidirectional RNN by hand using 2 forward RNNs
- The data to the backward RNN is fed in reverse

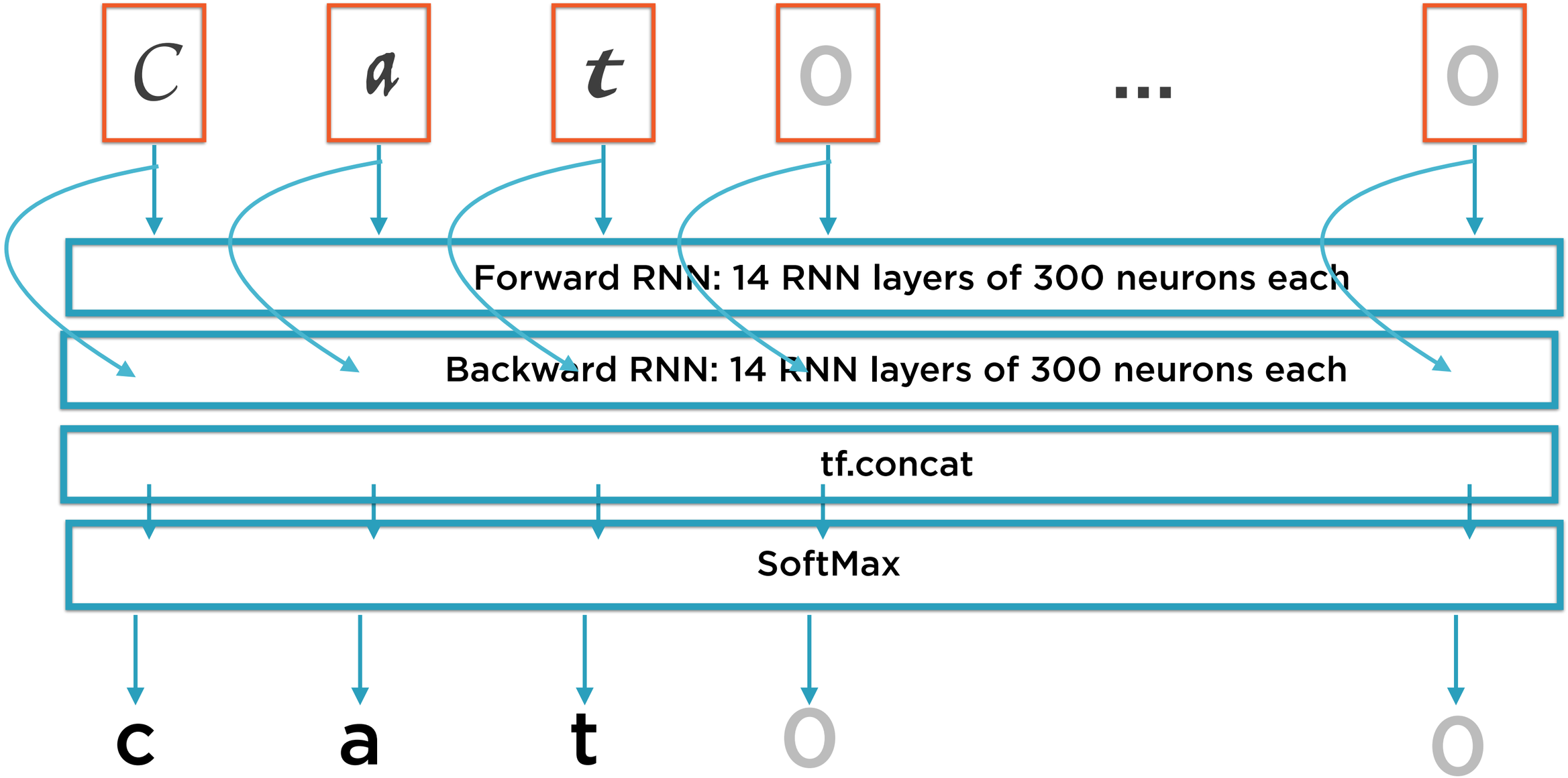
Conventional RNN Architecture



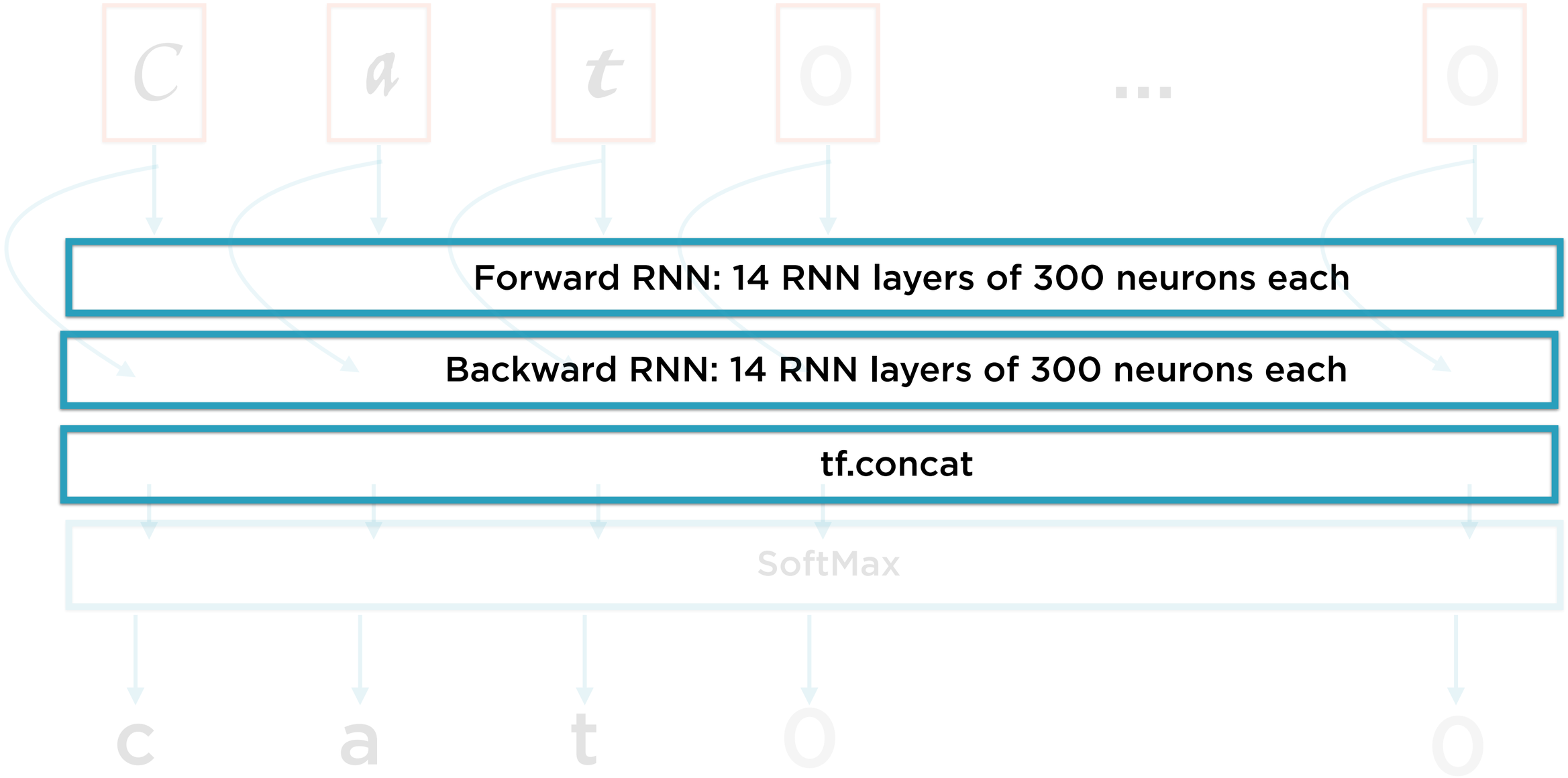
Conventional RNN Architecture



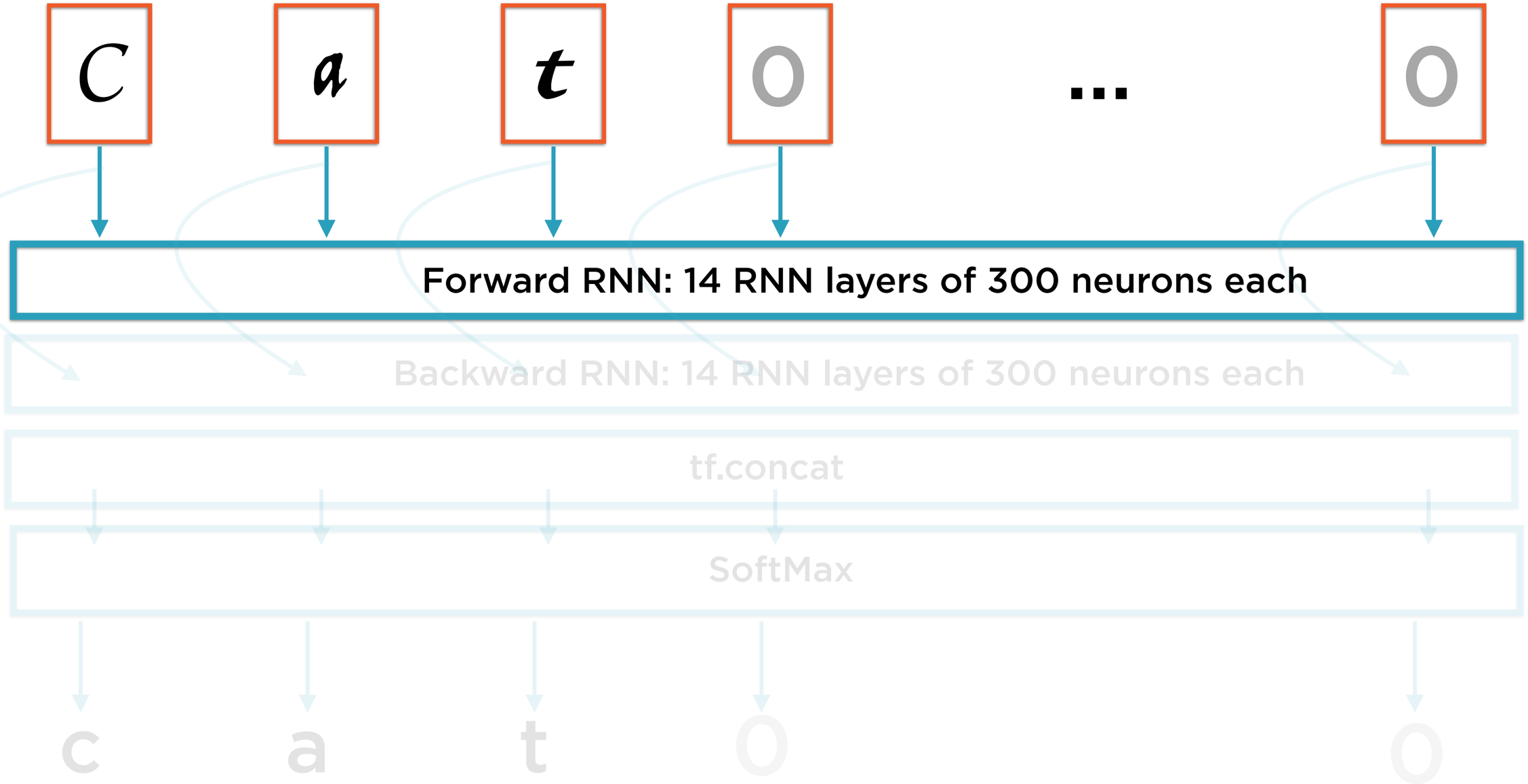
Bidirectional RNN Architecture



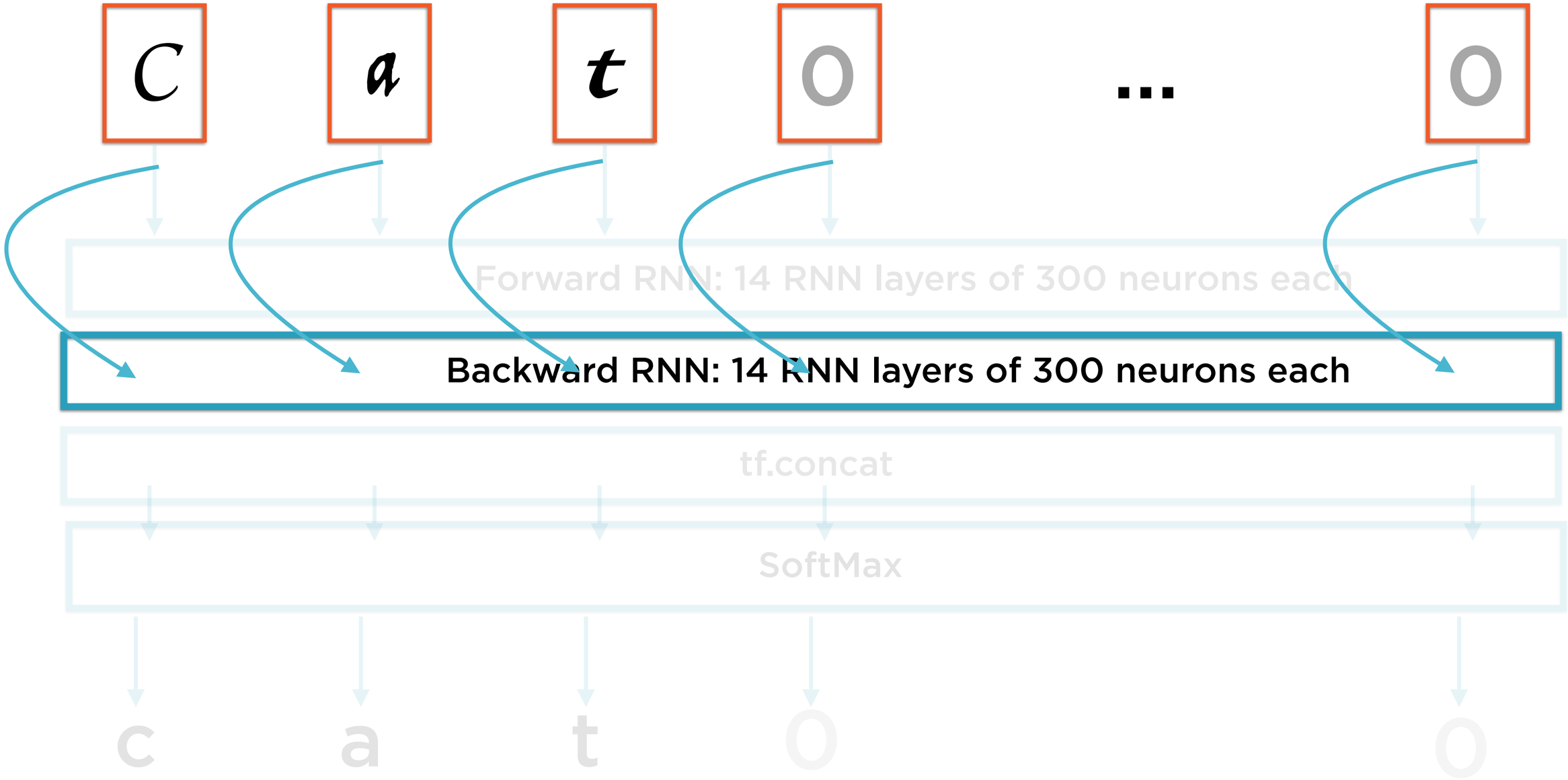
Bidirectional RNN Architecture



Bidirectional RNN Architecture

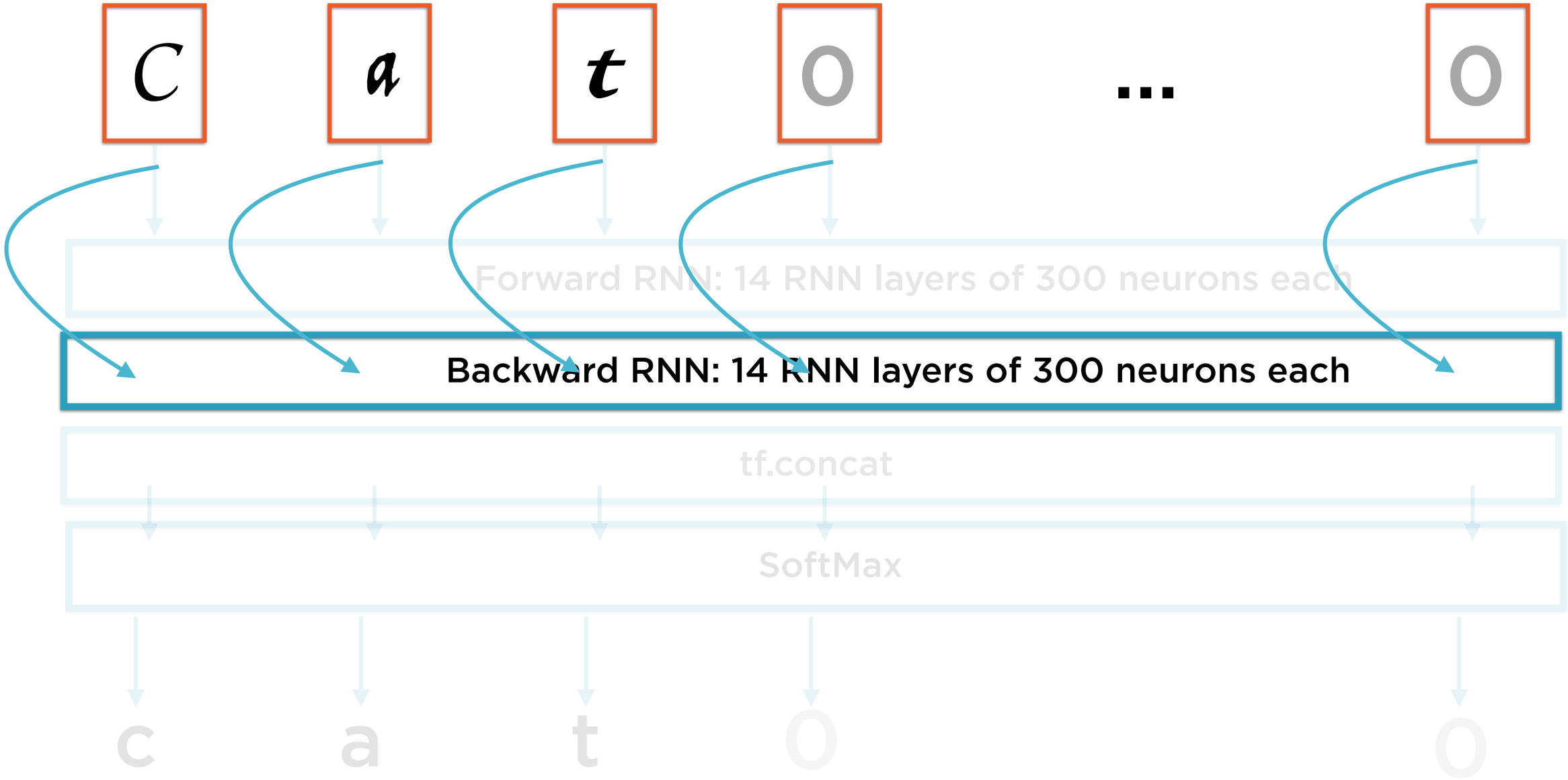


Bidirectional RNN Architecture

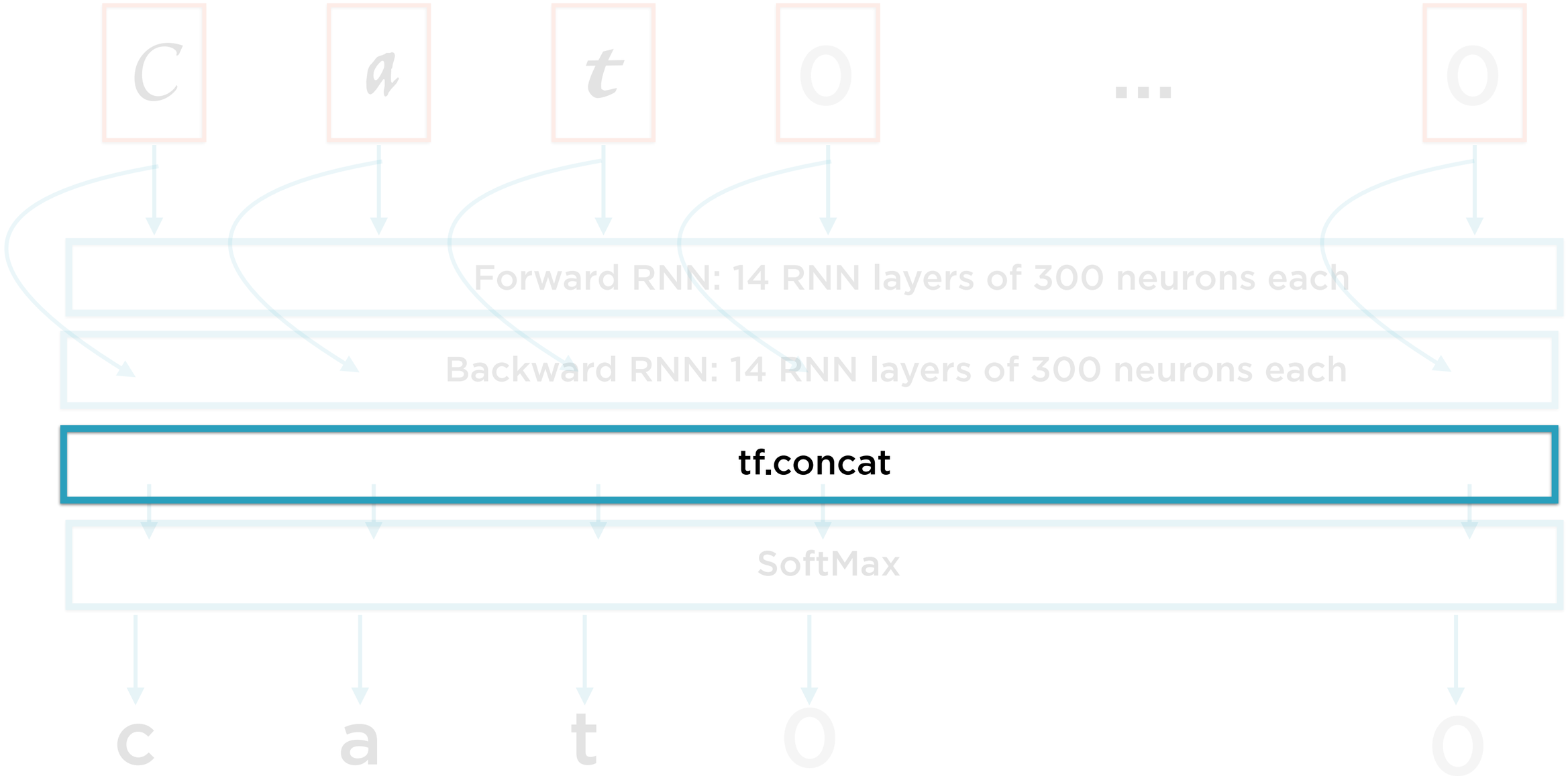


The input sequence is **reversed** and passed in through a forward RNN

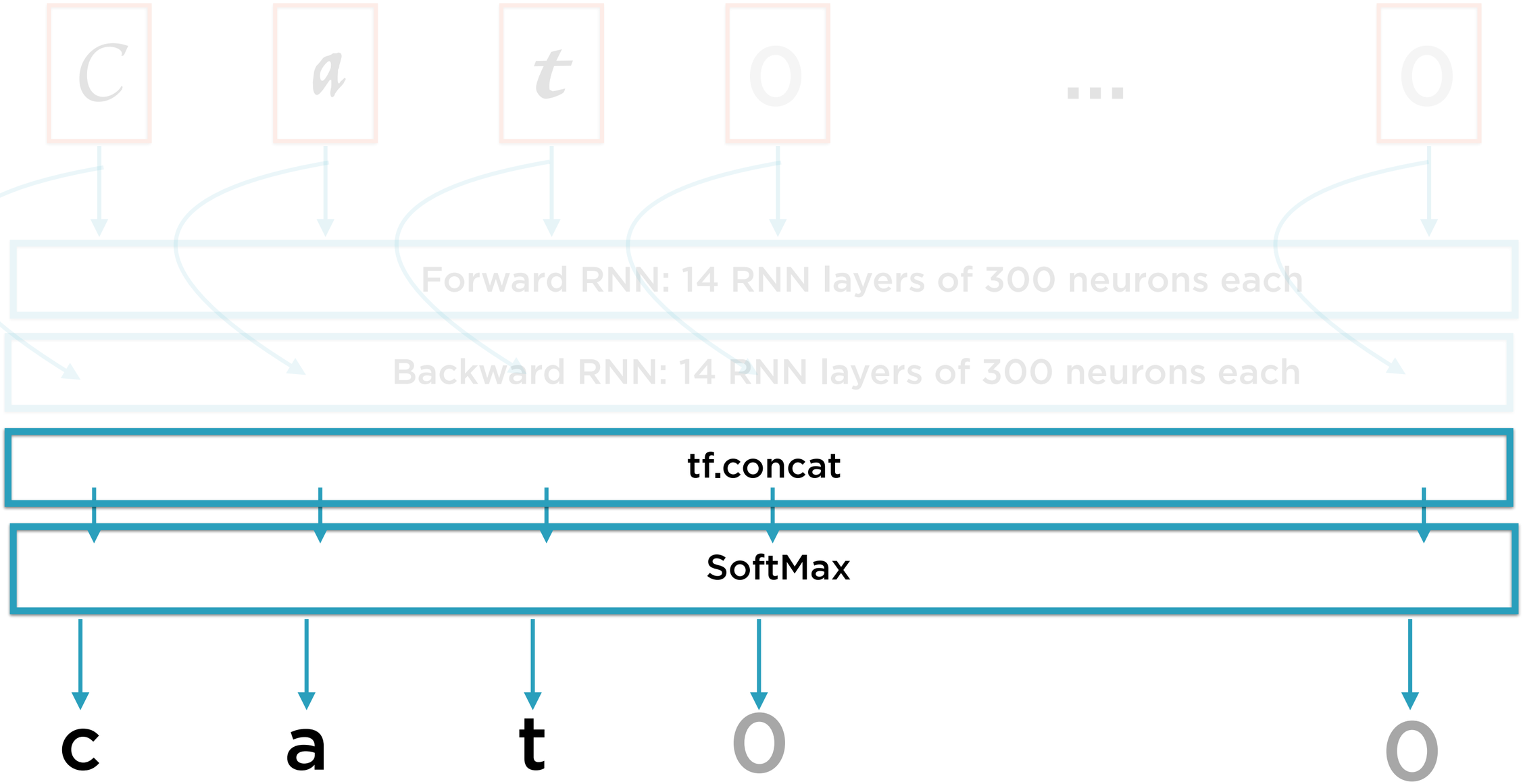
Bidirectional RNN Architecture



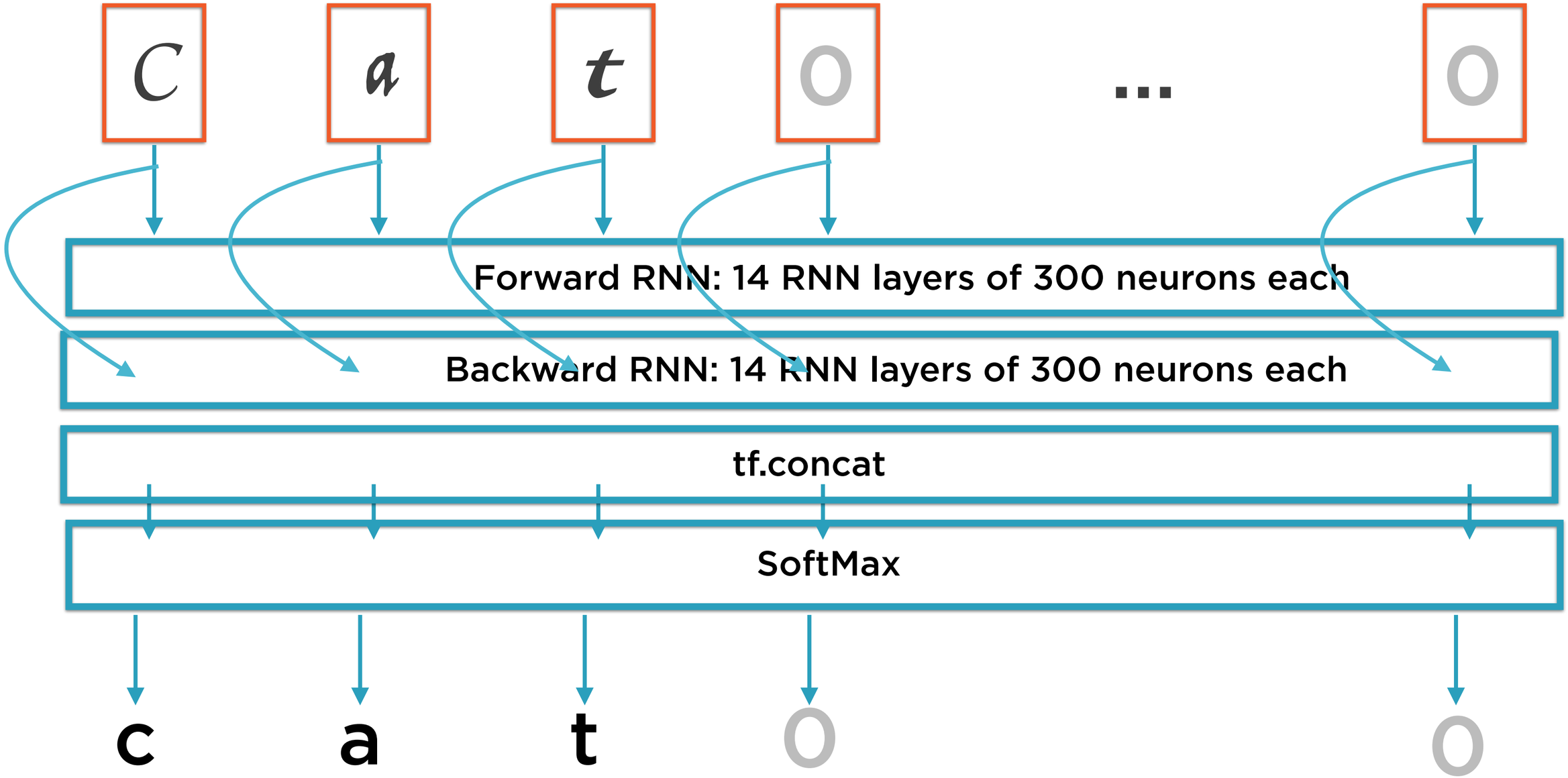
Bidirectional RNN Architecture



Bidirectional RNN Architecture



Bidirectional RNN Architecture



Demo

Perform optical character recognition using a bidirectional RNN on a modified version of the MIT OCR dataset

- Use the TensorFlow library to construct a bidirectional RNN

Summary

Performed optical character recognition using RNNs

Compared accuracy of a conventional RNN with a bidirectional RNNs