Sequences

Reference Char Models

- https://papers.nips.cc/paper/5782-character-level-convolutional-networks-for-text-classification.pdf

- https://arxiv.org/abs/1606.01781

Recap

- LSTM
- Handwriting example
- MNIST Example

An Alternate View

- Same MNIST Example

Sequence Of Pixels

- 784 Pixels per image
- Sequence of shape (784, 1)

Moral Of The Story

- Do what you want

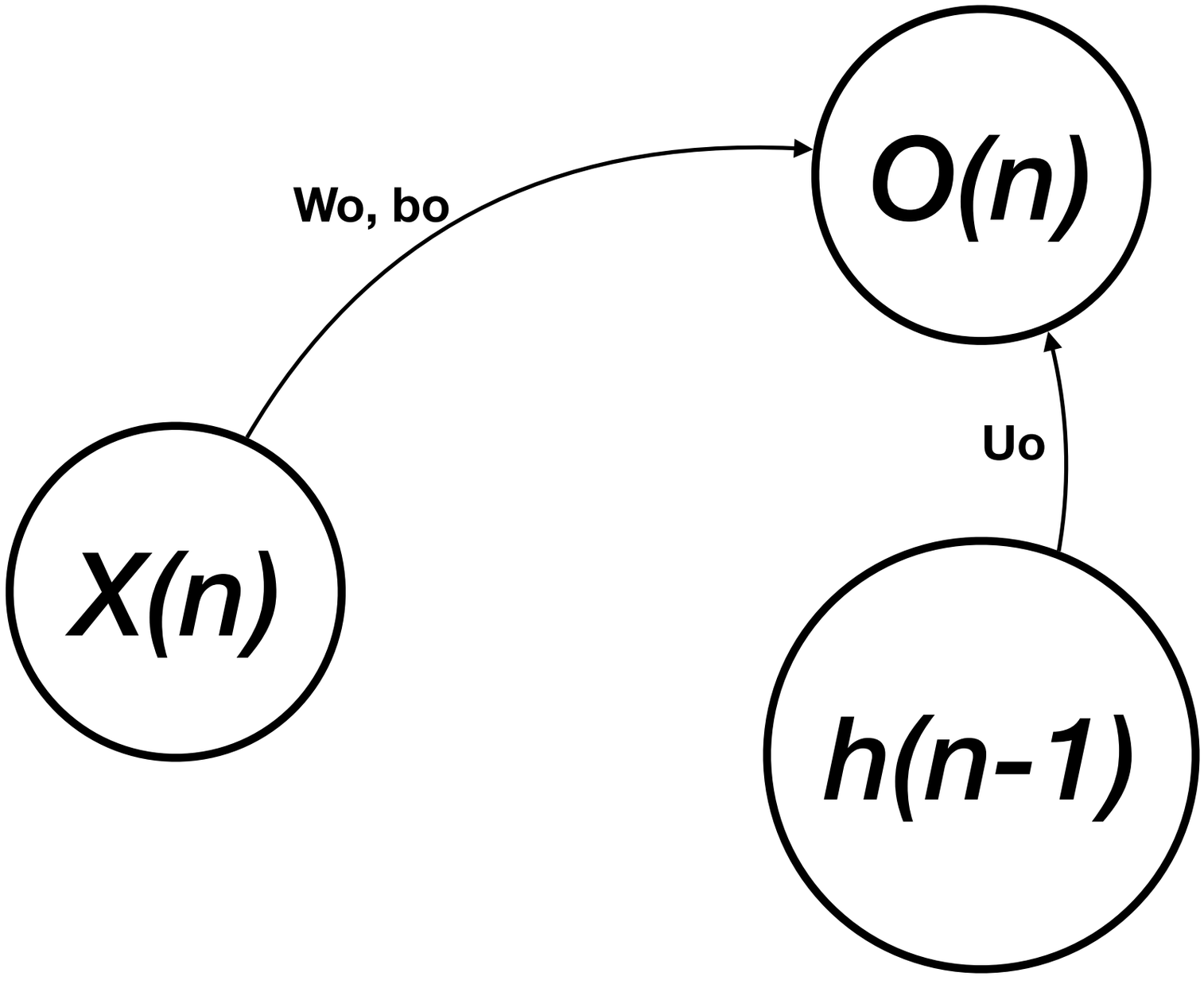
LSTM Training

- Backprop

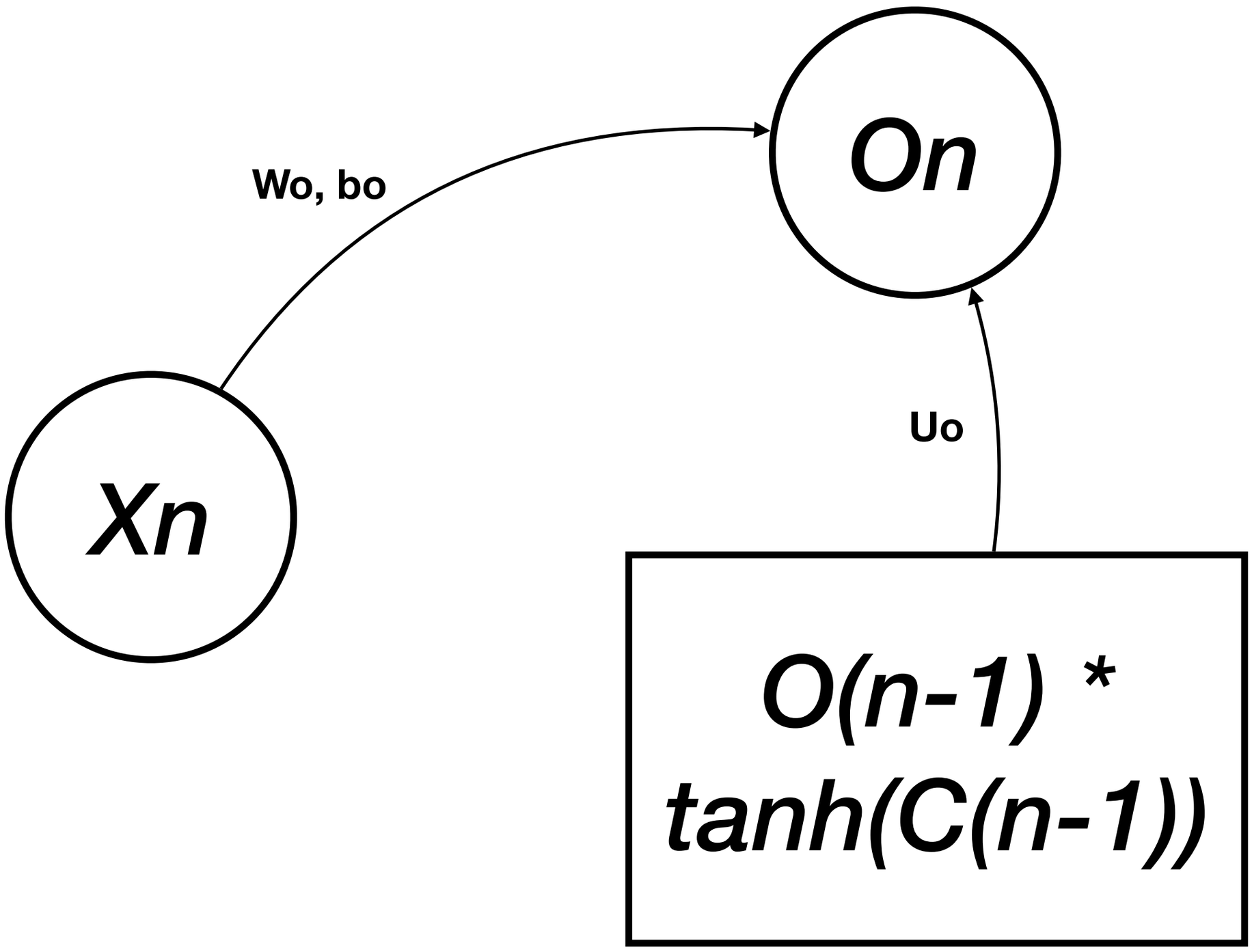
How?

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

Compute graph at nth step



At (n)th step

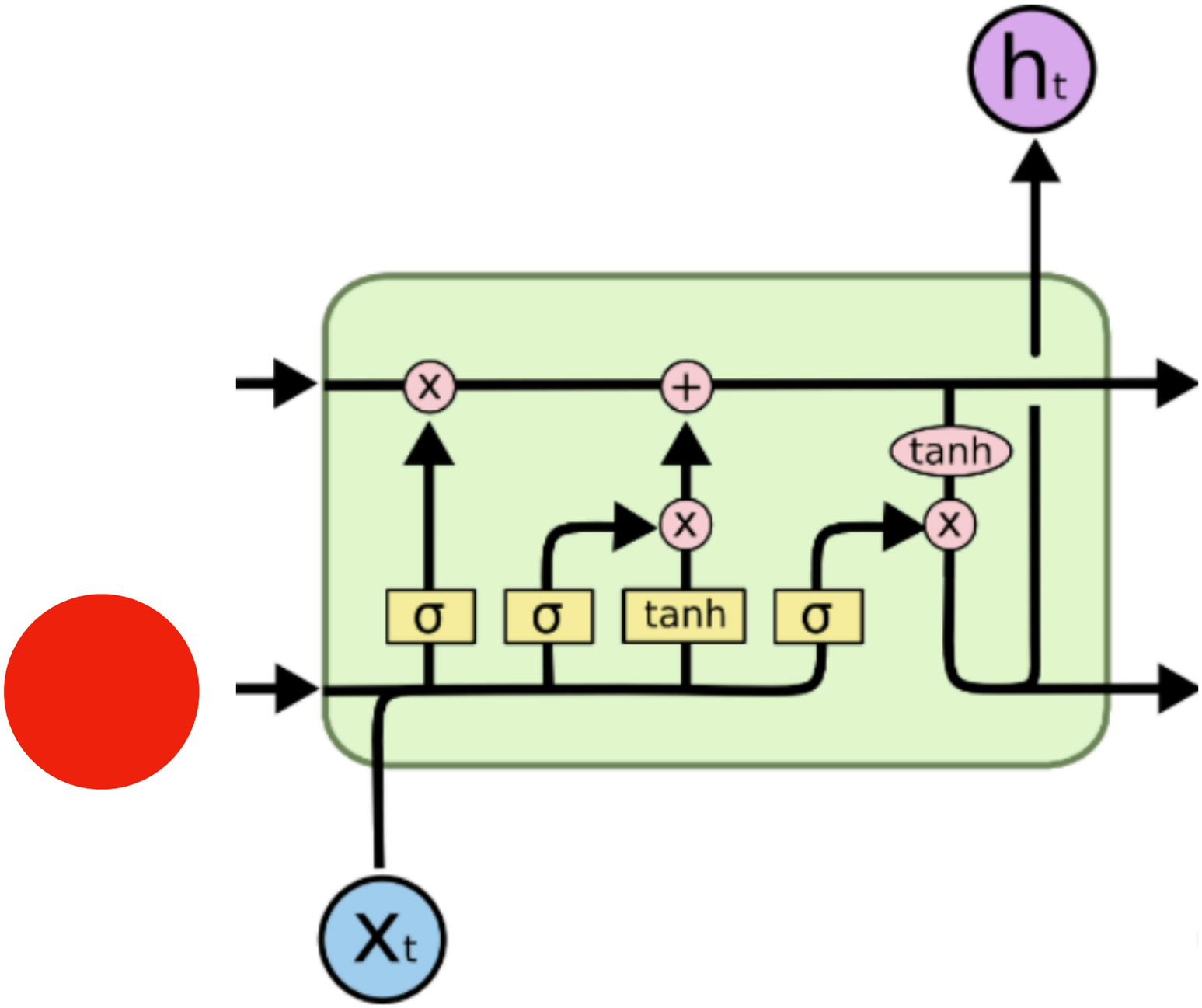


Details

- Keras example

- LSTMs actually accept a starting state

Diagram



Typically

- This value is a zero vector

- But not always - we will soon see when this is the case

Also

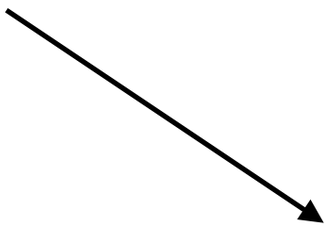
```
keras.layers.LSTM(
    . . .,
    return_sequences=False,
    return_state=False,
    . . .
)
```

Consider

- Sequential input
- And sequential output

Machine Translation

I speak French



Je parle en Français

Text To Speech

I have a dream

Speech To Text

I have a dream

Etcetera

Ideas?

Here’s One

- 2 LSTMs
- Encoder + Decoder

Input

- Sequence
- $< x^0, x^1, \ldots, x^t, \ldots, x^n >$

Output

- Sequence
- $< y^0, y^1, \ldots, y^t, \ldots, y^m >$

Encoder

- Work on the input sequence
- Construct a representation (i.e. encode)

Decoder

- Start with encoder’s encoded representation
- Generate sequence with it

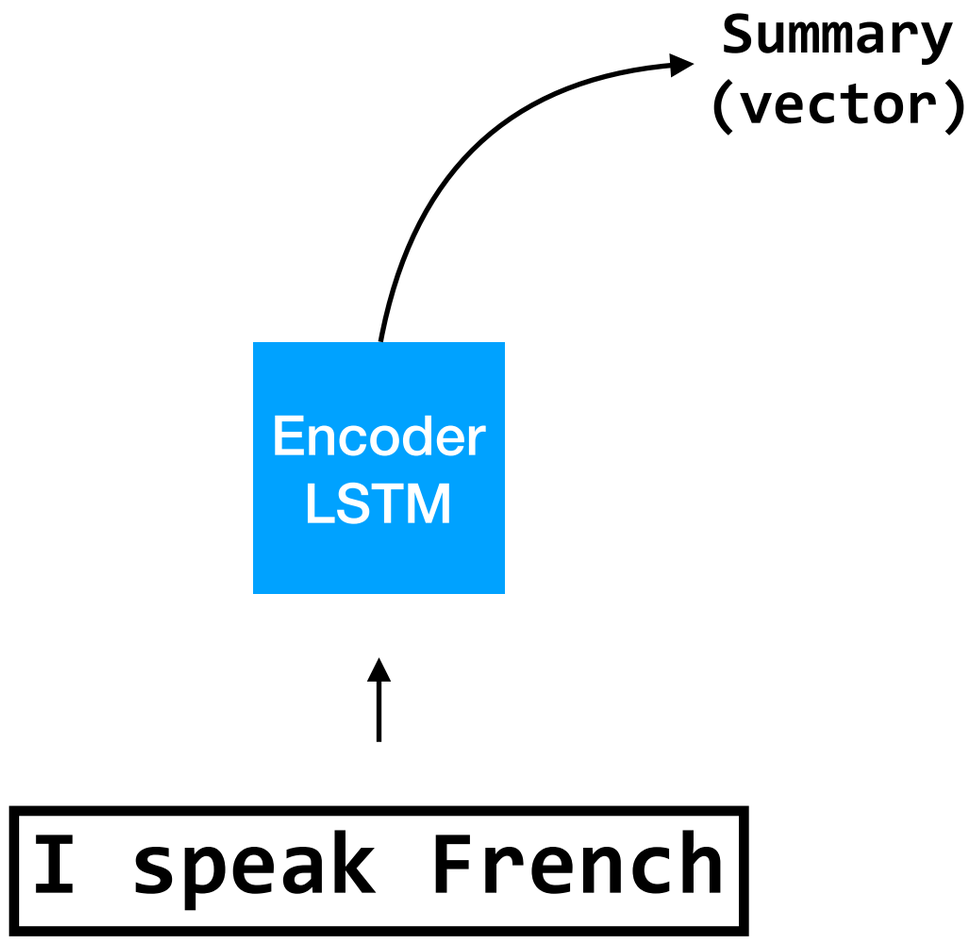
How?

- Hit me with some ideas

Intuition

- Encoder = think of our MNIST example
- LSTM summarizes the pixel-row sequence
- Summary fed into MLP

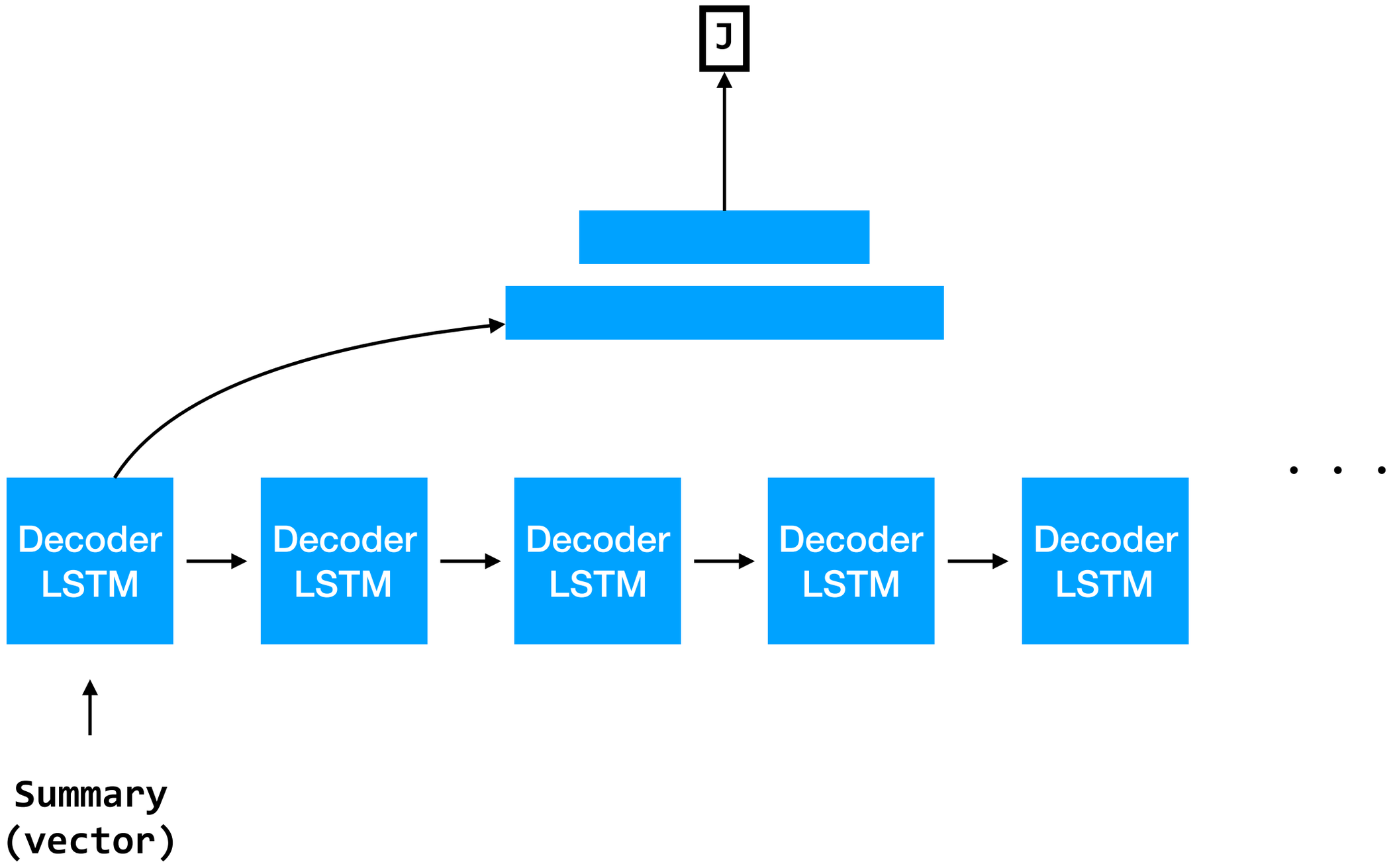
Encoder Diagram



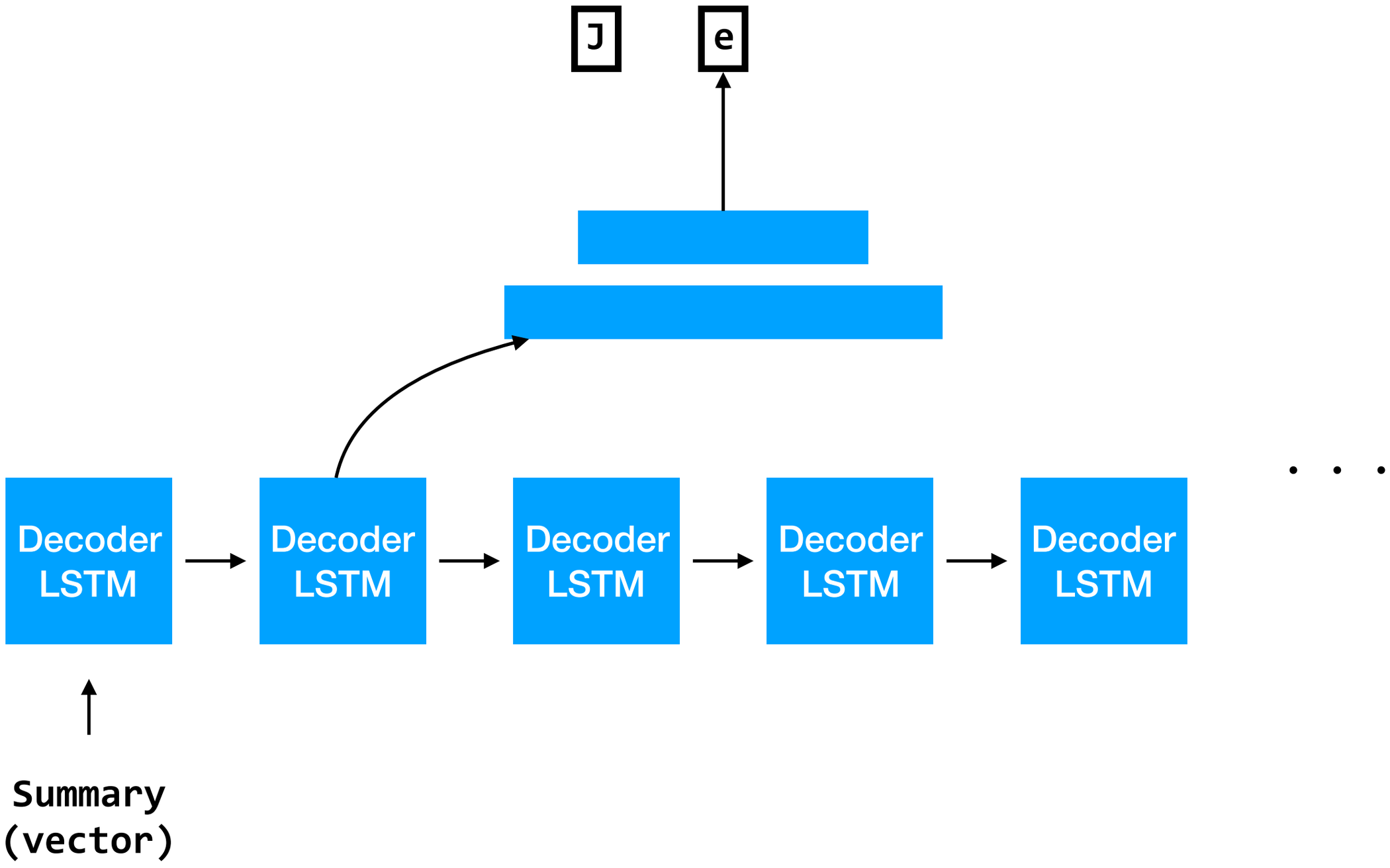
Decoder

- Summary -> Output Sequence

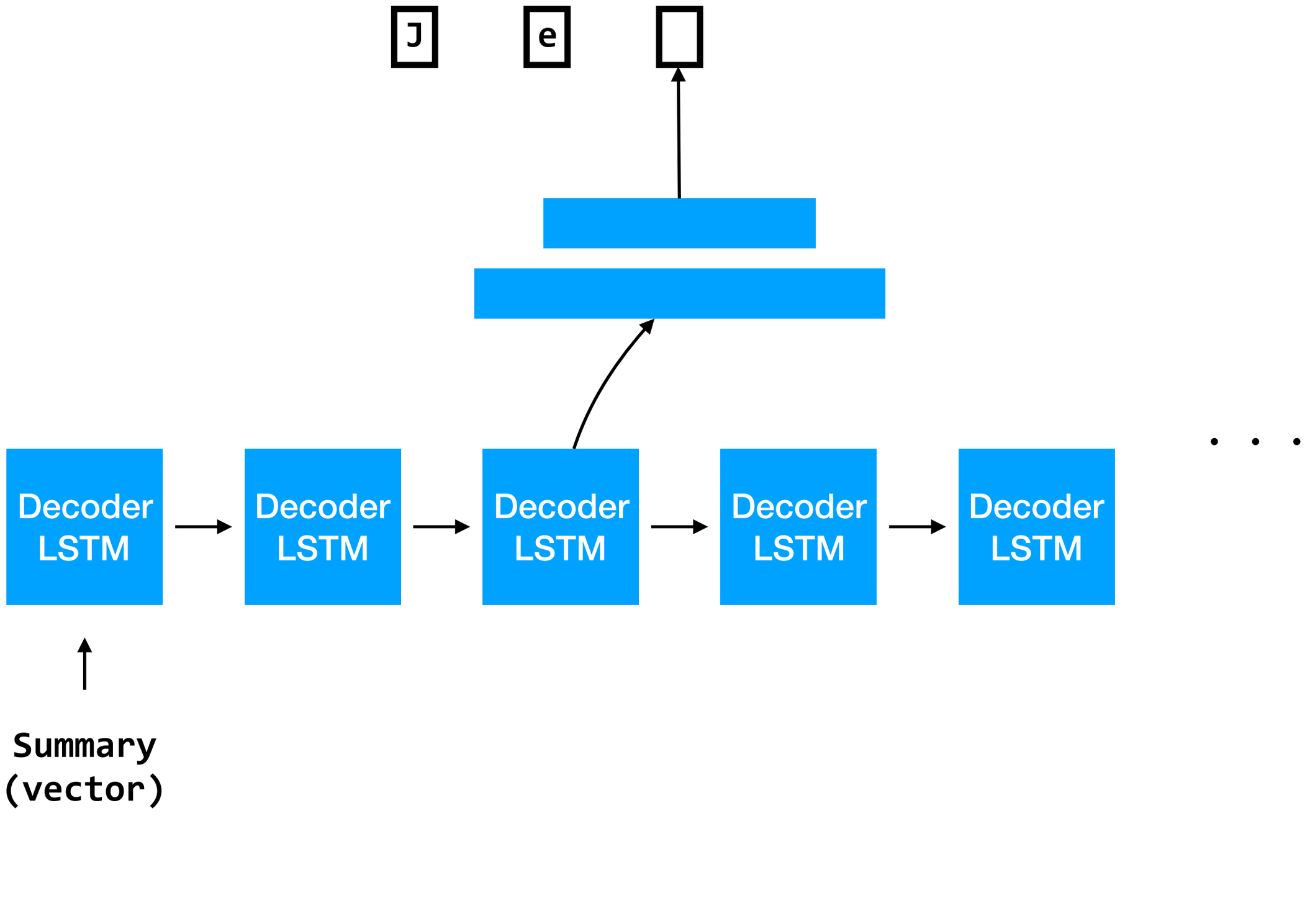
Intuition


J
Decoder LSTM
Decoder LSTM
Decoder LSTM
Decoder LSTM
Decoder LSTM
Summary (vector)

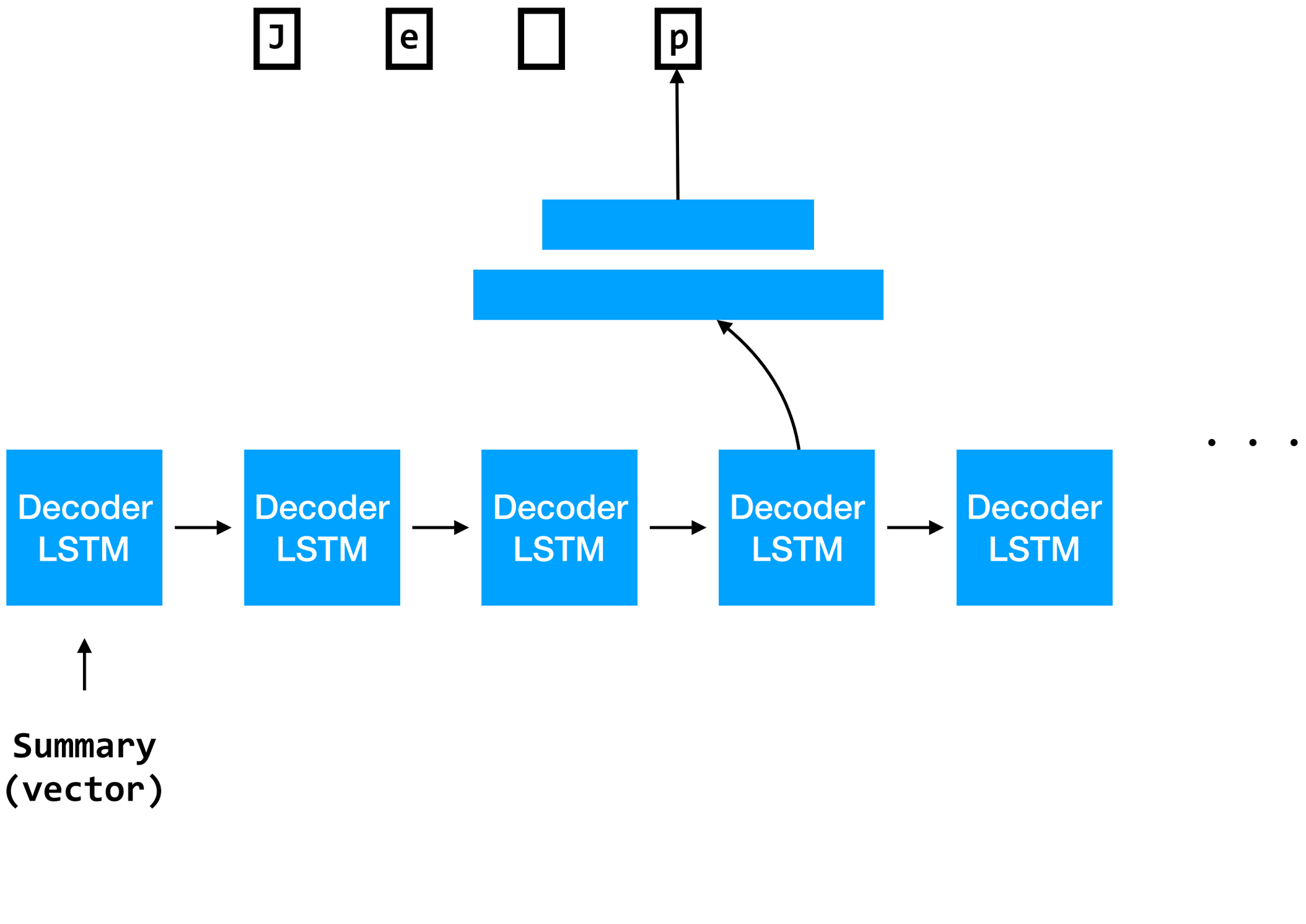
Intuition


J
e
Decoder LSTM
Decoder LSTM
Decoder LSTM
Decoder LSTM
Decoder LSTM
Summary (vector)

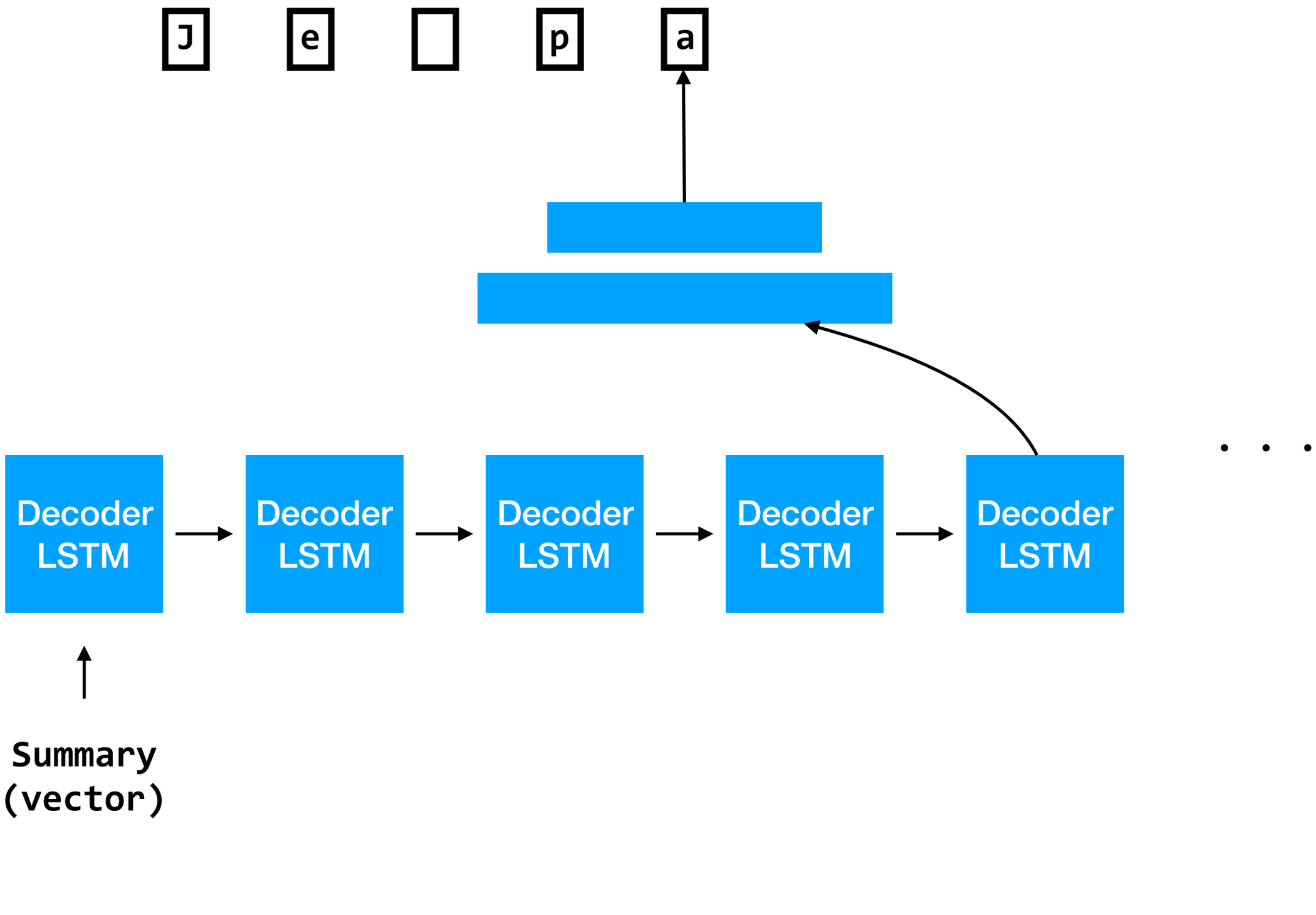
Intuition


J
e
Decoder LSTM
Decoder LSTM
Decoder LSTM
Decoder LSTM
Decoder LSTM
Summary (vector)

Intuition


J
e
p
Decoder LSTM
Decoder LSTM
Decoder LSTM
Decoder LSTM
Decoder LSTM
Summary (vector)

Intuition


J
e
p
a
Decoder LSTM
Decoder LSTM
Decoder LSTM
Decoder LSTM
Decoder LSTM
Summary (vector)

Alternate Method

- Remember handwriting

Handwriting

- LSTM consumes image columns
- Emits symbol

Handwriting

Handwriting

i

↑

LSTM

↑

And so on…

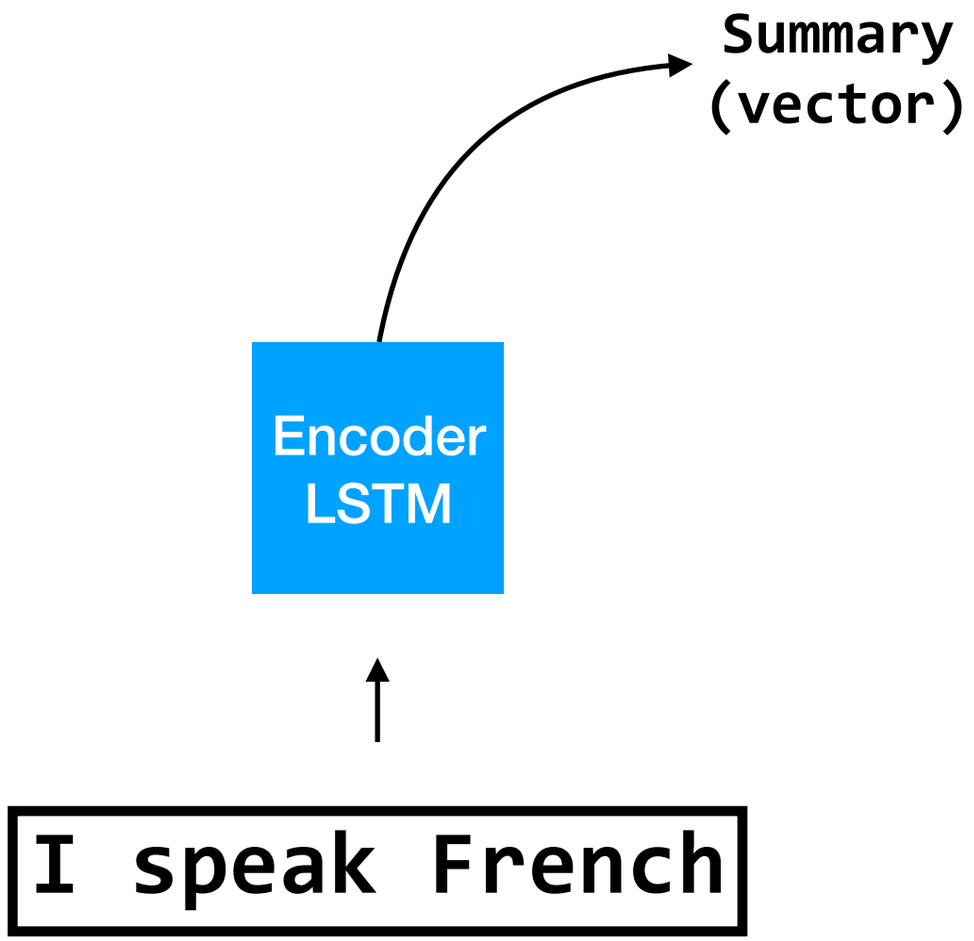
Decoder LSTM

- Can look like this

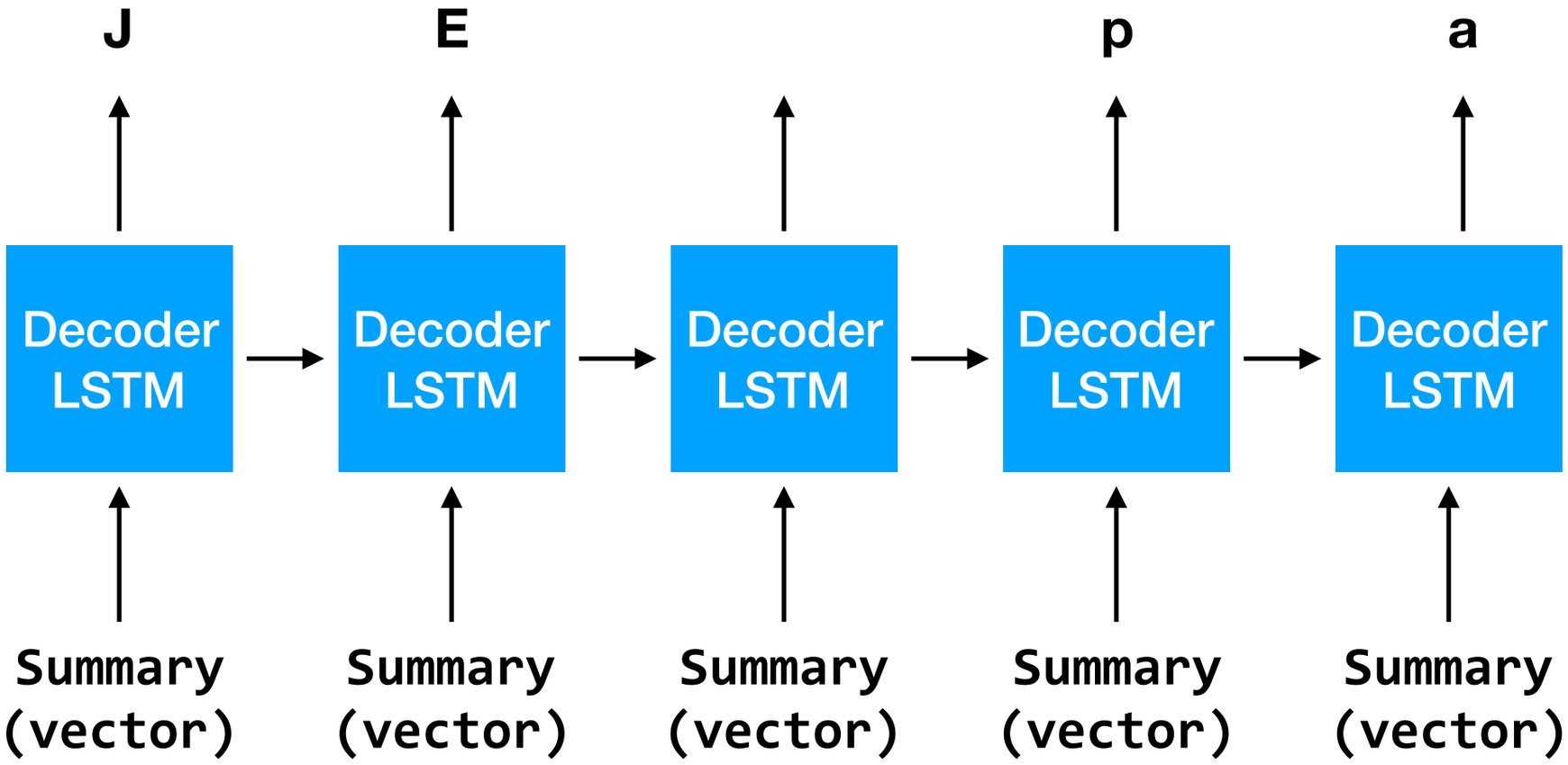
How

- Encoder gives you state
- Repeat this state m times (m = decoder lstm length)
- Map this m length sequence to your output

Intuition



And


J
E
p
a
Decoder LSTM
Decoder LSTM
Decoder LSTM
Decoder LSTM
Decoder LSTM
Summary (vector)
Summary (vector)
Summary (vector)
Summary (vector)
Summary (vector)

This Variant

- Is **EASIER** to implement in Keras (for now)
- So I will use it (suggest you too for next assignment)

Simple Example

- Addition

Specifically

- Add 2 numbers

Return

- A number

Input

?

Output

?

Our model

- Input
  - Binary strings
- Output
  - Binary string

Example

- 000000, 000001
  - = 000001
- 000001, 000001
  - = 000010

Model

?

Code

Extending This

- Input as strings of type ‘x+y’
- Output is a string of characters

Ideas?

One Idea

- Alphabet for seq2seq = digits 0 - 9 and the ‘+’ symbol

- Alphabet for output = digits 0 - 9

Training

- Regular seq2seq setup

Depth

Deep LSTM?

Ideas?

Simple

- LSTM emits one output vector per sequence position
- These form a sequence
- Input to another LSTM

Deep LSTM

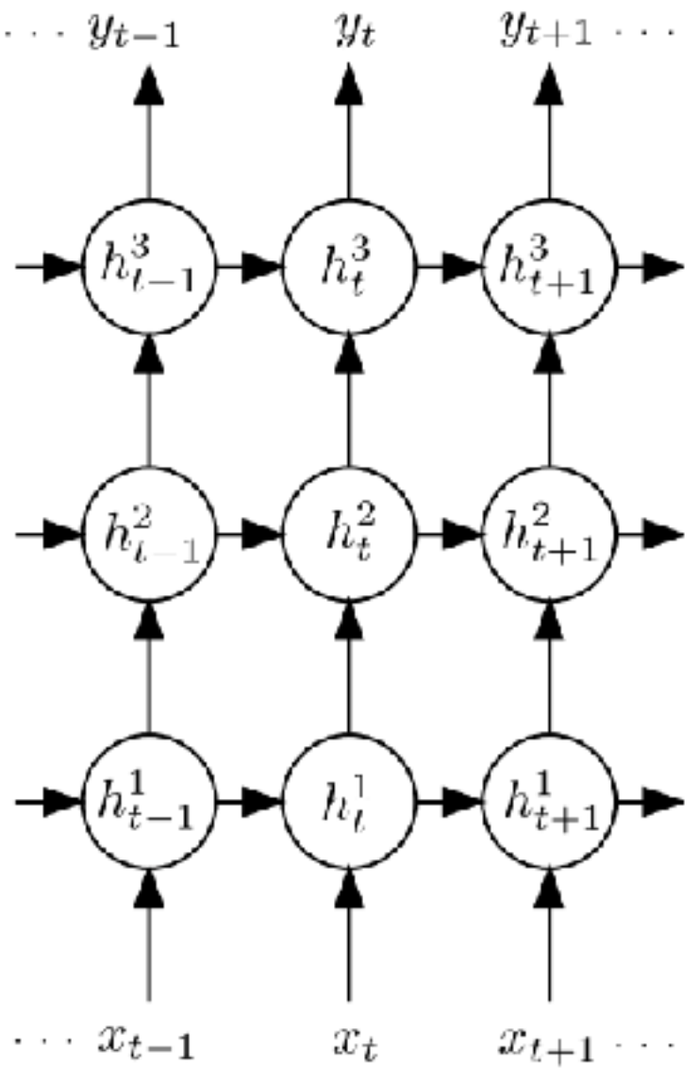


Fig. 3. Deep Recurrent Neural Network

Keras Code

- Very trivial

```
model.add(LSTM(hidden_size, return_sequences=True))
model.add(LSTM(hidden_size, return_sequences=True))
model.add(LSTM(hidden_size))
```

Extending This Idea

- Machine Translation

- How?

Empirically

- Limitations with plain seq2seq
- Brittle on longer inputs

MT Observations

- Reversing the sequence works well (!?!)
- Duplicating input seq works well (!?!)

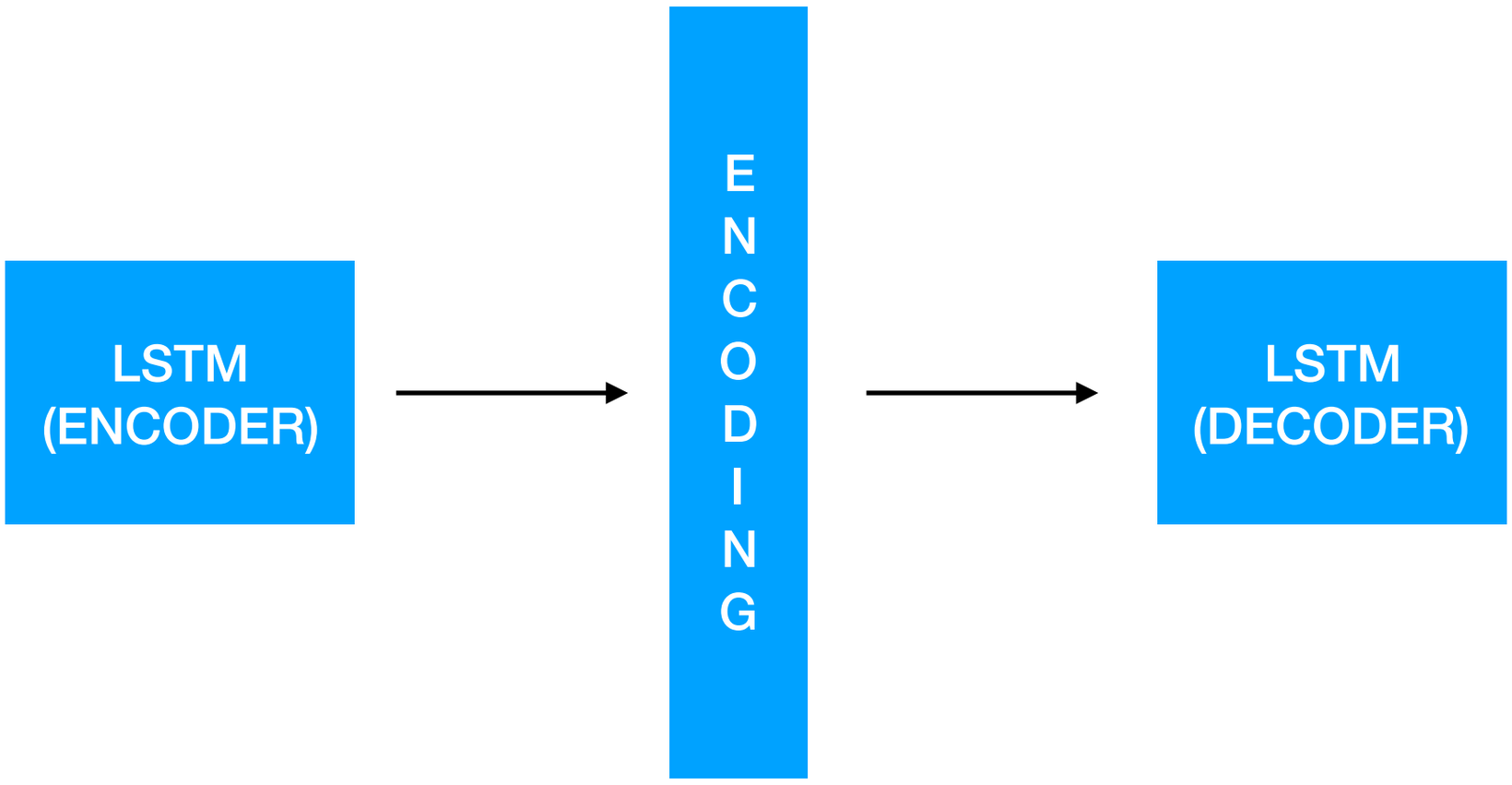
Conclusions

- 1 representation for whole sequence
- Representation must hold everything about sequence
- Assumption might not hold for longer sequences

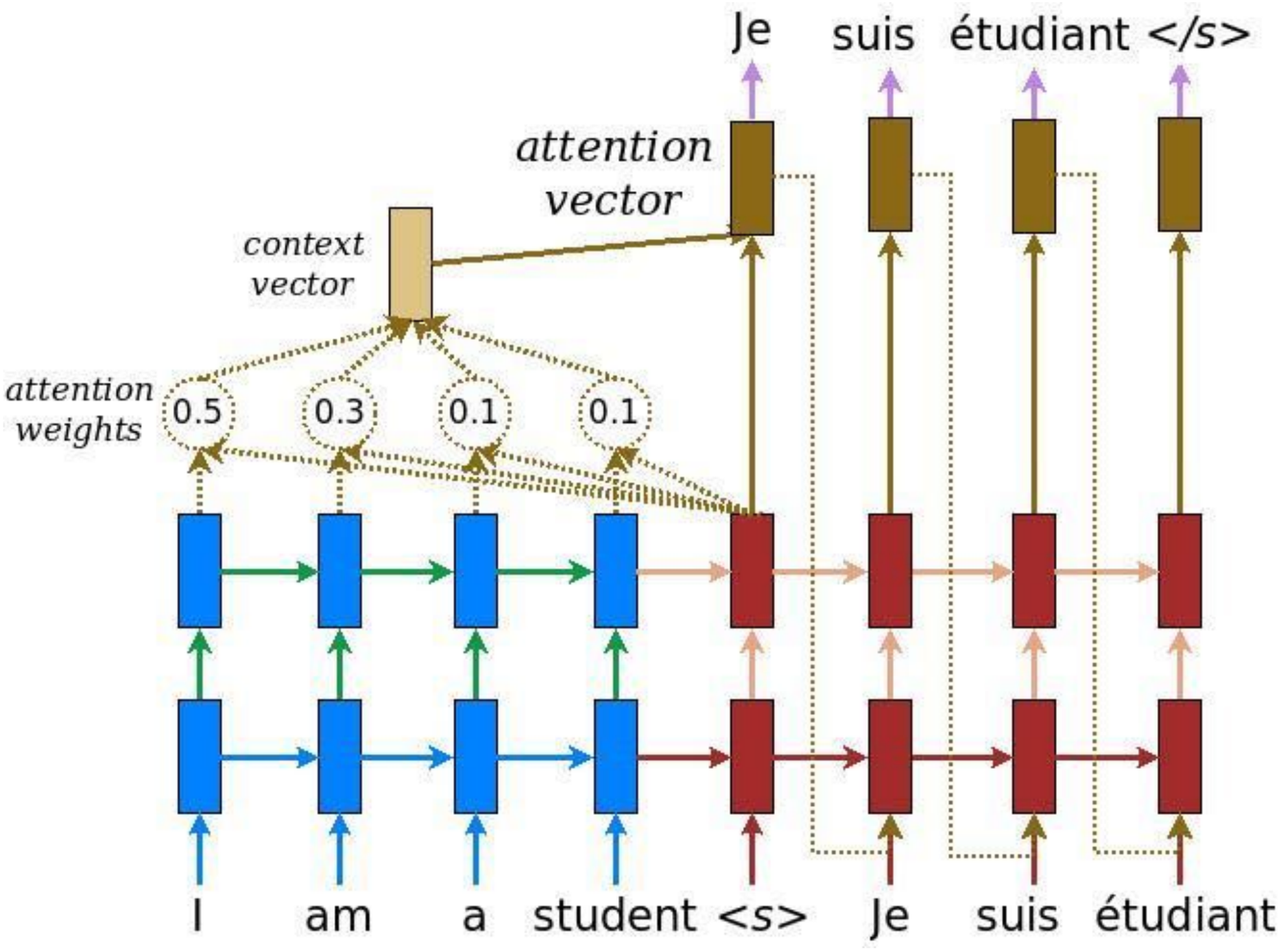
Innovation

- Don’t use just the final hidden state of encoder LSTM

seq2seq



Innovation


Je
suis
étudiant </s>
attention vector
context vector
attention weights
0.5
0.3
0.1
0.1
I
am
a
student
<s>
Je
suis
étudiant

$$\alpha_{ts} = \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'=1}^{S} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)} \qquad \text{[Attention weights]} \qquad (1)$$

$$\boldsymbol{c}_t = \sum_s \alpha_{ts} \bar{\boldsymbol{h}}_s \qquad \text{[Context vector]} \qquad (2)$$

$$\boldsymbol{a}_t = f(\boldsymbol{c}_t, \boldsymbol{h}_t) = \tanh(\boldsymbol{W}_{\boldsymbol{c}}[\boldsymbol{c}_t; \boldsymbol{h}_t]) \qquad \text{[Attention vector]} \qquad (3)$$

At Each Decoding Step

- Look at all encoder step hidden states

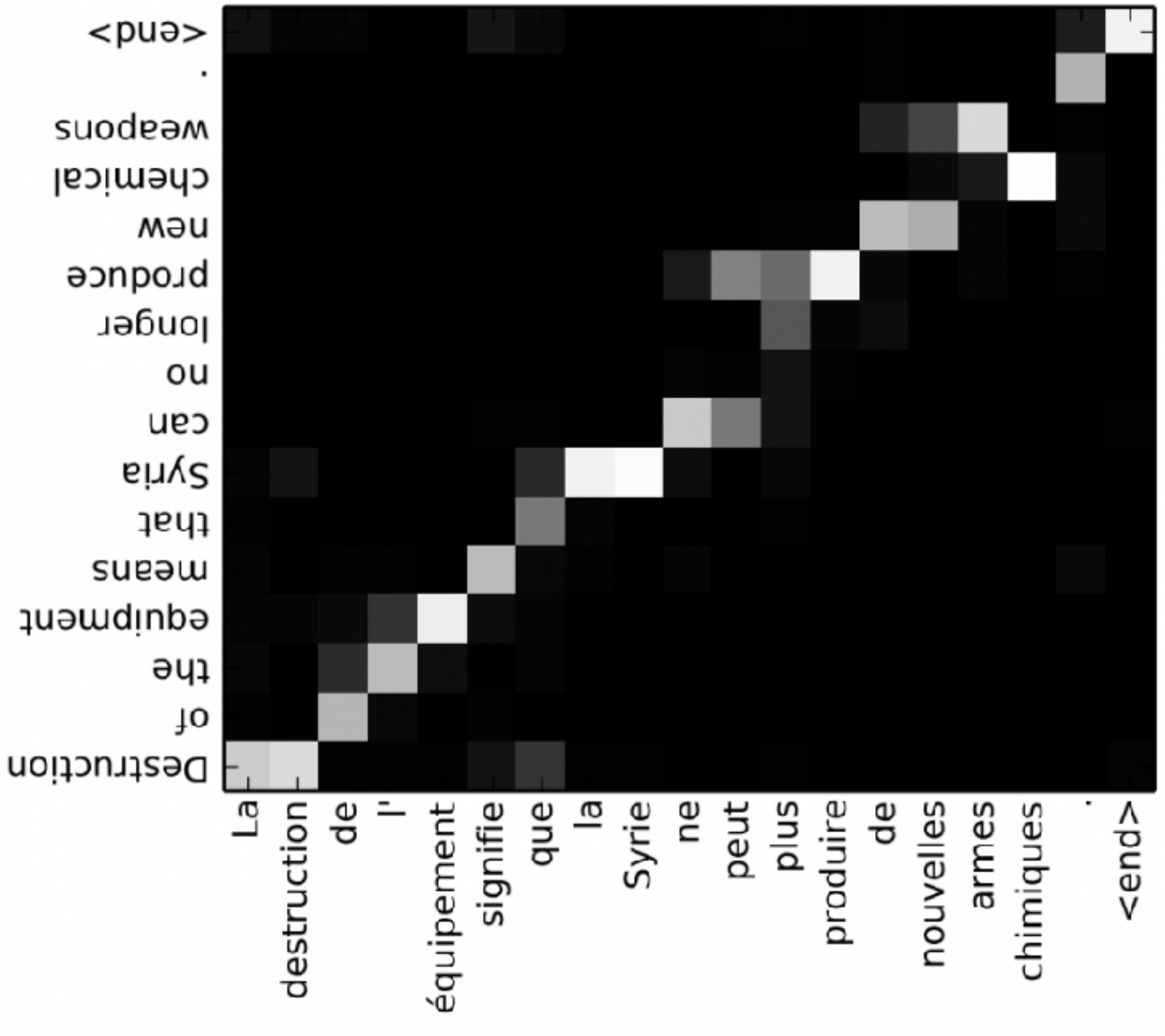
- Weigh (computed by a neural net) inputs

This Is Called

- Attention

Specifically

- The decoder weighs the input sequence (in addition to whatever it has generated so far)
- Decides what parts of the input sequence are “important”


Destruction
of
the
equipment
means
that
Syria
can
no
longer
produce
new
chemical
weapons
.
<end>
La
destruction
de
l'
équipement
signifie
que
la
Syrie
ne
peut
plus
produire
de
nouvelles
armes
chimiques
.
<end>

Some Creative Ideas

CRNN

- Convolutional backbone processes image
- Get a representation
- This representation is a sequence
- Input to LSTM model (or seq2seq if you want)

CRNN

- Handwriting

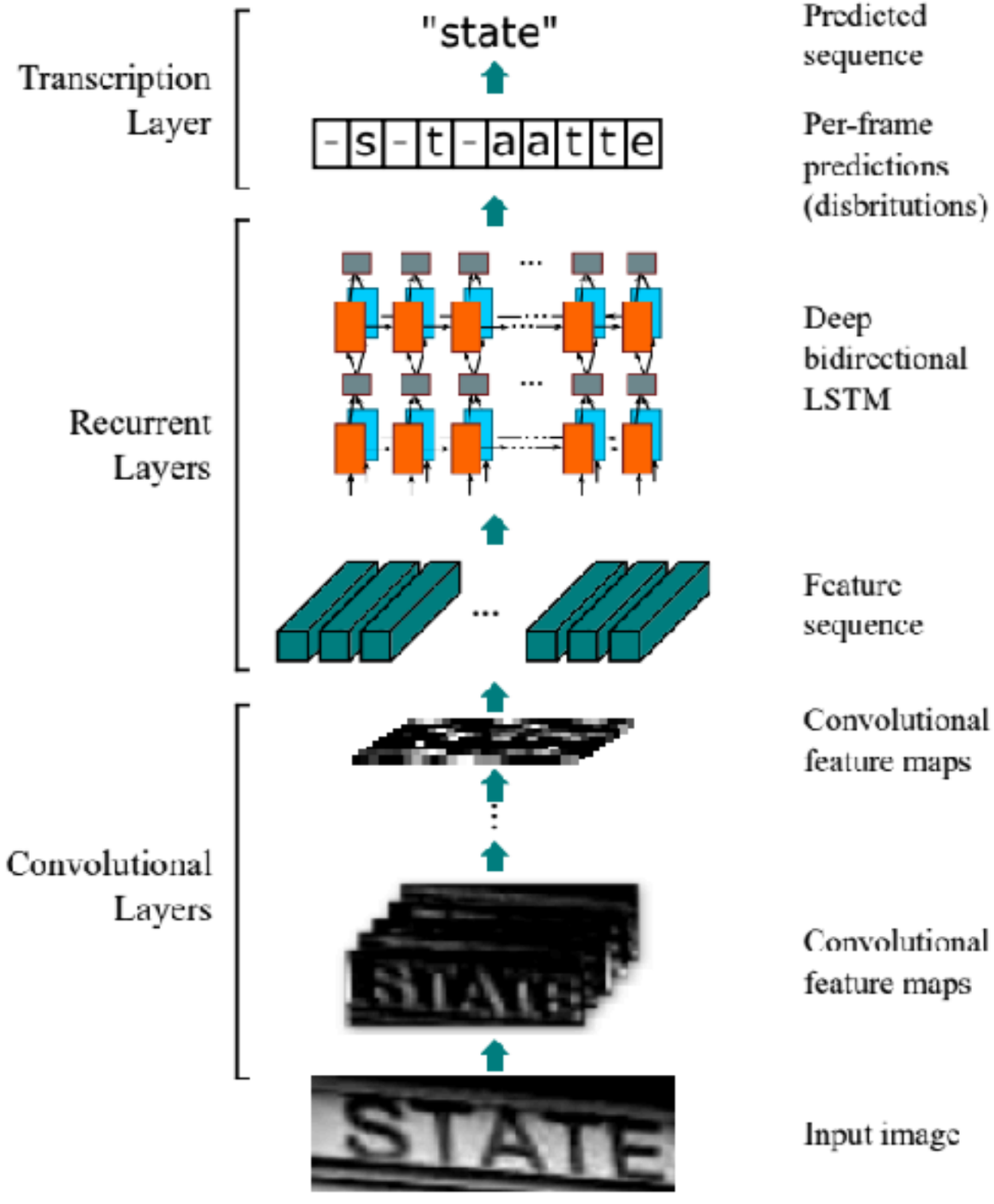

Transcription Layer
"state"
-s-t-aatte
Predicted sequence
Per-frame predictions (disbritutions)
Recurrent Layers
Deep bidirectional LSTM
Feature sequence
Convolutional Layers
Convolutional feature maps
Convolutional feature maps
STATE
Input image
STATE

Image Captioning

- CNN for image

- Input to seq2seq for “describing” the representation it produces

Tensorflow

https://www.tensorflow.org/tutorials/seq2seq