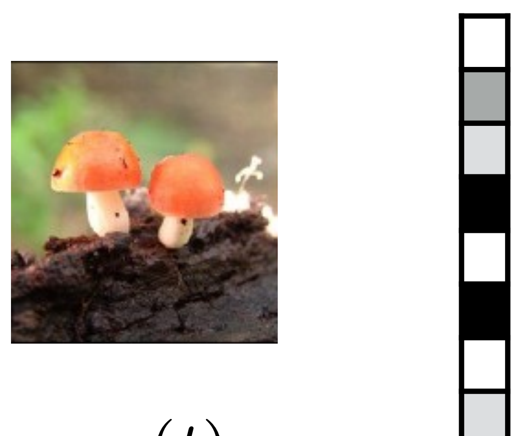
6.036 Introduction to Machine Learning

Recurrent Neural Networks

NN recap (what you should know)

- Feed-forward (layered) neural networks
  - units, weights, activation functions
  - layer-wise forward propagation
  - back-propagation of error
  - stochastic gradient descent
- Convolutional neural networks (CNNs)
  - filters, feature maps, pooling



mushroom

$x^{(t)}$ $y^{(t)}$

Multi-way classification

- (explained on the board; see notes)

Temporal/sequence problems

- How to cast as a supervised learning problem?

Temporal/sequence problems

- How to cast as a supervised learning problem?

- Historical data can be broken down into feature vectors and target values (sliding window)

$$\underbrace{\begin{bmatrix} 0.82 \\ 0.80 \\ 0.73 \\ 0.72 \end{bmatrix}}_{x^{(t)}} \qquad \underbrace{0.89}_{y^{(t)}}$$

Temporal/sequence problems

- Language modeling: what comes next?

This course has been a tremendous …

Temporal/sequence problems

- Language modeling: what comes next?

This course has been a tremendous ...

tremendous $\begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix}$

?

a $\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$

$x^{(t)}$ $y^{(t)}$

Temporal/sequence problems

- Language modeling: what comes next?

This course has been a | tremendous …

a $\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$

tremendous

been $\begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}$

$x^{(t)}$ $y^{(t)}$

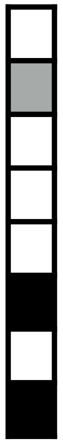
What are we missing?

- Sequence prediction problems can be recast in a form amenable to feed-forward neural networks
- But we have to engineer how “history” is mapped to a vector (representation). This vector is then fed into, e.g., a neural network
  - how many steps back should we look at?
  - how to retain important items mentioned far back?
- Instead, we would like to learn how to encode the “history” into a vector

Learning to encode/decode

- Language modeling

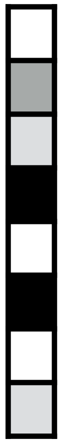
This course has been a



success (?)

- Sentiment classification

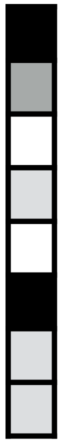
I have seen better lectures



-1

- Machine translation

I have seen better lectures



Olen nähnyt parempia luentoja

encoding **decoding**

Key concepts

- **Encoding**
  - e.g., mapping a sequence to a vector
- **Decoding**
  - e.g., mapping a vector to, e.g., a sequence

Encoding everything

words $\begin{bmatrix} .1 \\ .3 \\ .4 \end{bmatrix} \begin{bmatrix} .7 \\ .1 \\ .0 \end{bmatrix} \begin{bmatrix} .2 \\ .8 \\ .3 \end{bmatrix}$ ...

“Efforts and courage are not enough without purpose and direction” — JFK

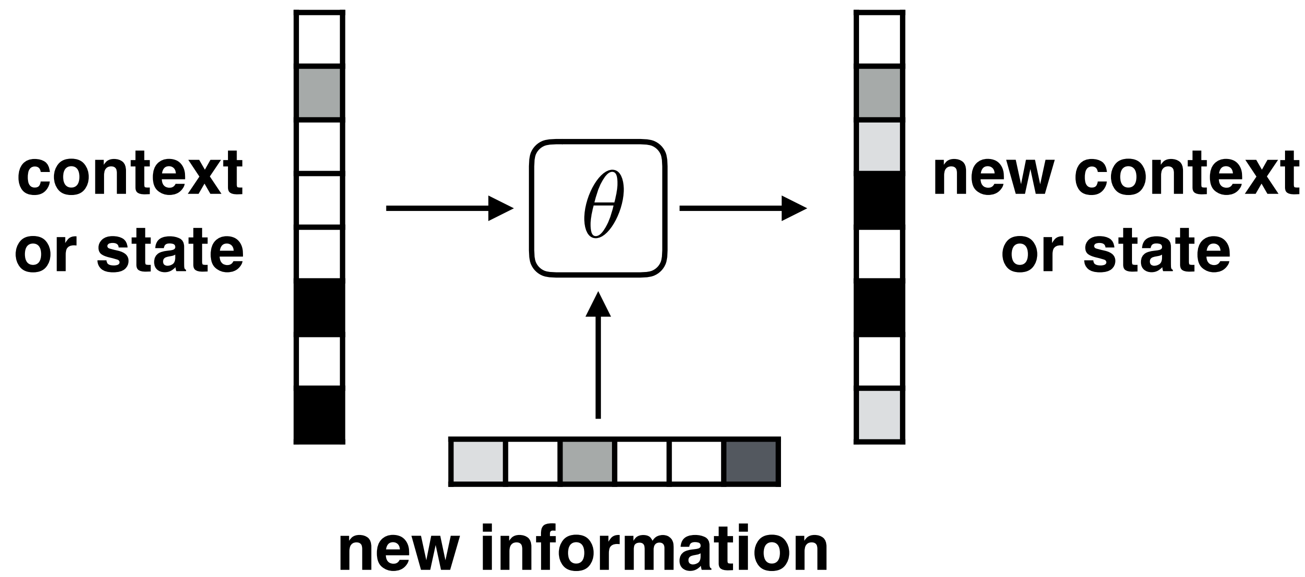
sentences $\begin{bmatrix} .2 \\ .3 \\ .6 \end{bmatrix}$

images $\begin{bmatrix} .3 \\ .3 \\ .5 \end{bmatrix}$

events $\begin{bmatrix} .2 \\ .4 \\ .6 \end{bmatrix}$

Example: encoding sentences

- Easy to introduce adjustable "lego pieces" and optimize them for end-to-end performance



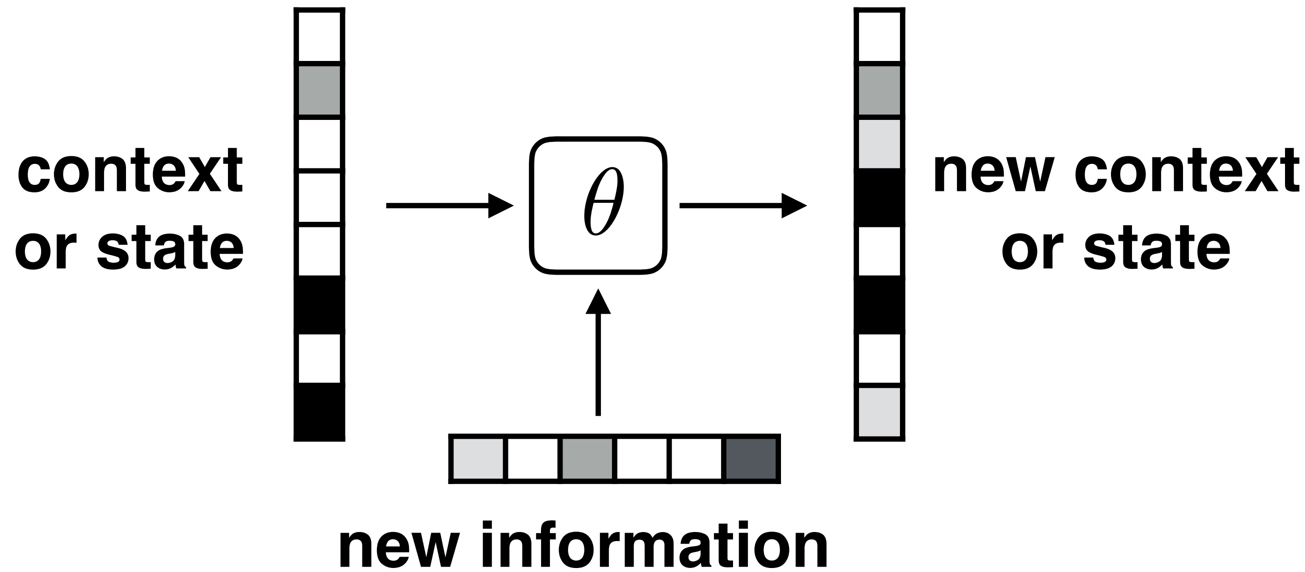
RNN

<null>

Efforts and courage are not …

Example: encoding sentences

- Easy to introduce adjustable "lego pieces" and optimize them for end-to-end performance



RNN

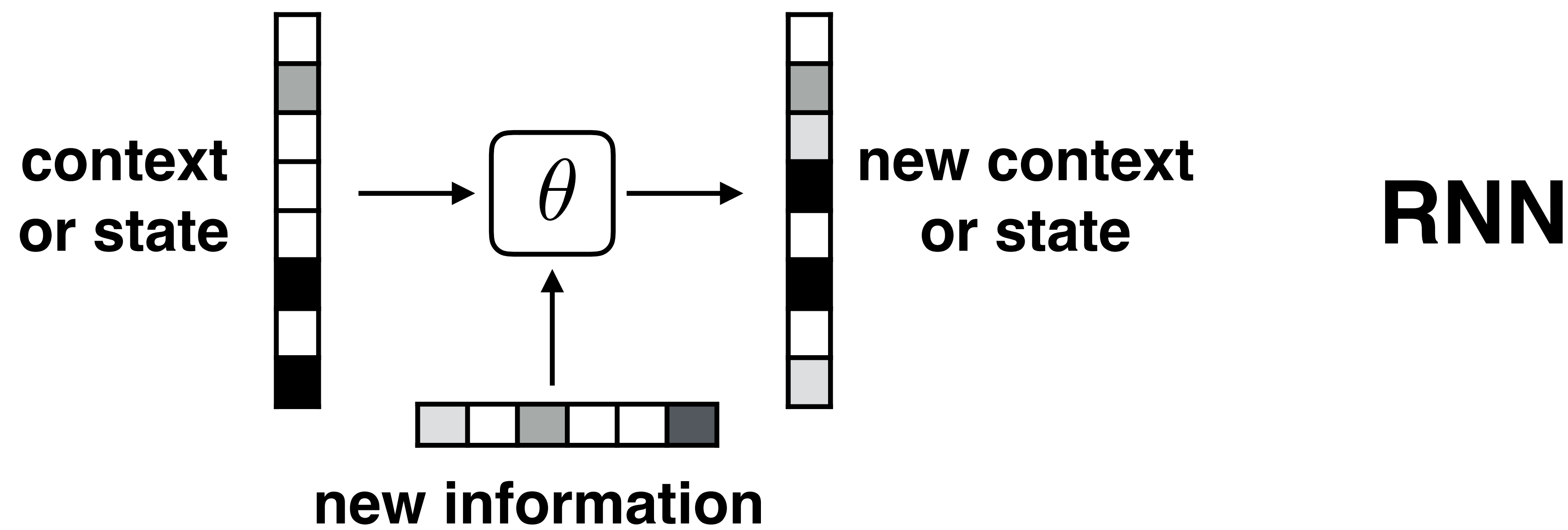
$$s_t = \tanh(W^{s,s}s_{t-1} + W^{s,x}x_t)$$

<null>

Efforts and courage are not …

Example: encoding sentences

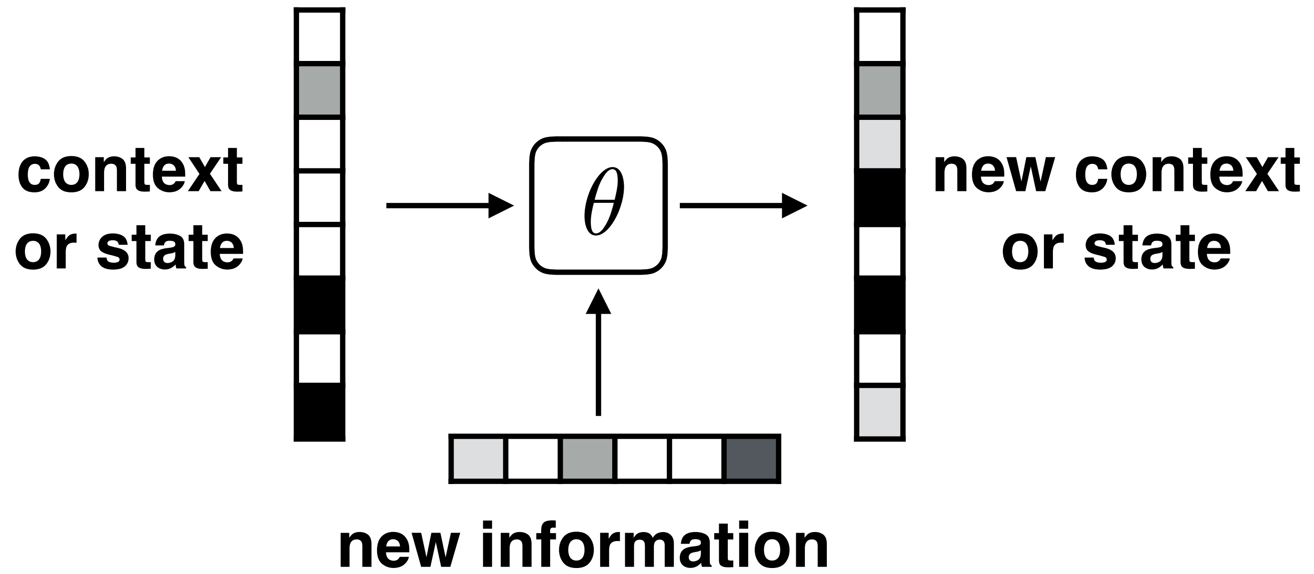
- Easy to introduce adjustable "lego pieces" and optimize them for end-to-end performance



$$s_t = \tanh(W^{s,s} s_{t-1} + W^{s,x} x_t)$$

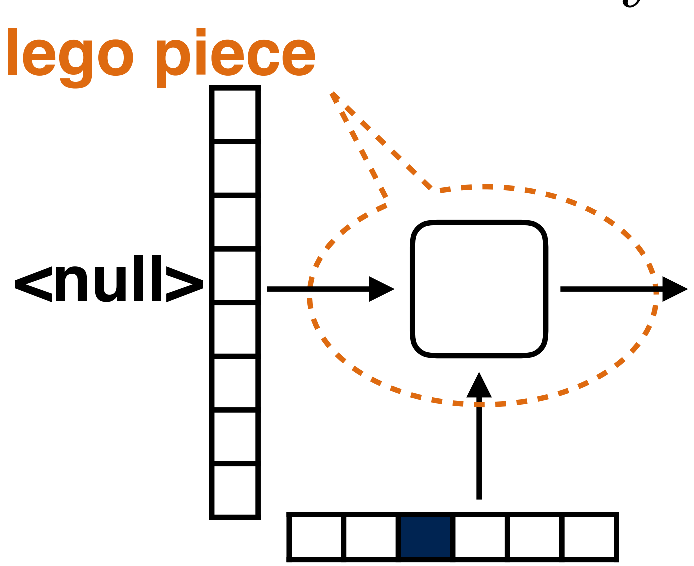
Example: encoding sentences

- Easy to introduce adjustable “lego pieces” and optimize them for end-to-end performance



RNN

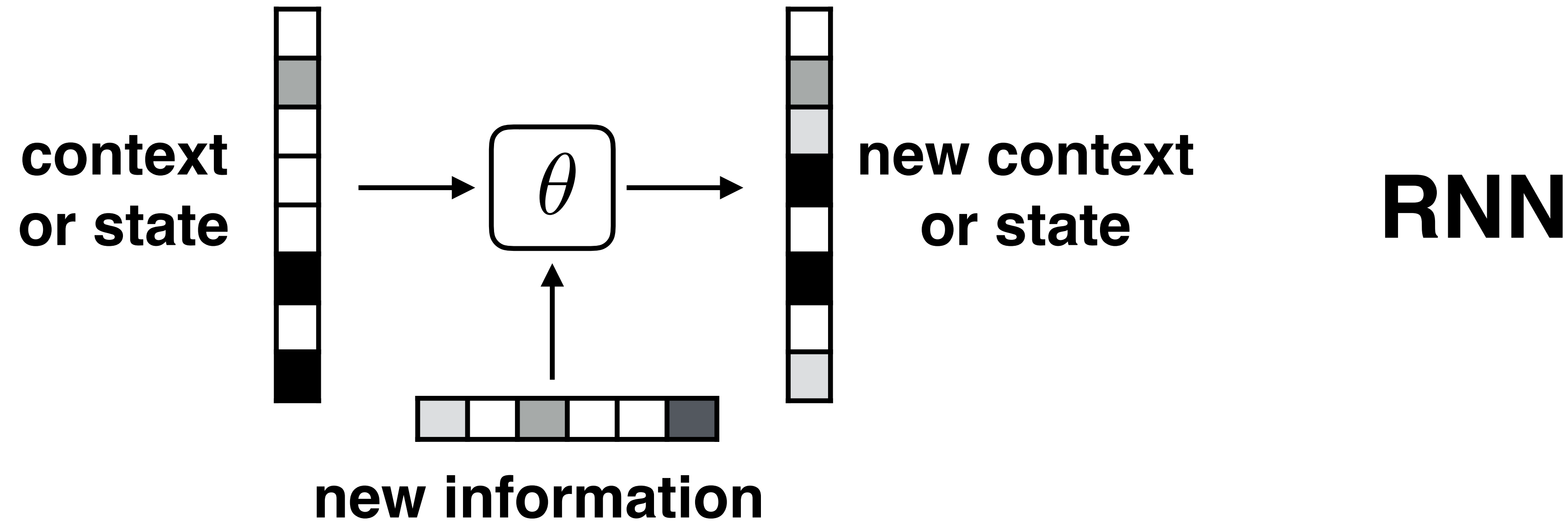
$$s_t = \tanh(W^{s,s} s_{t-1} + W^{s,x} x_t)$$



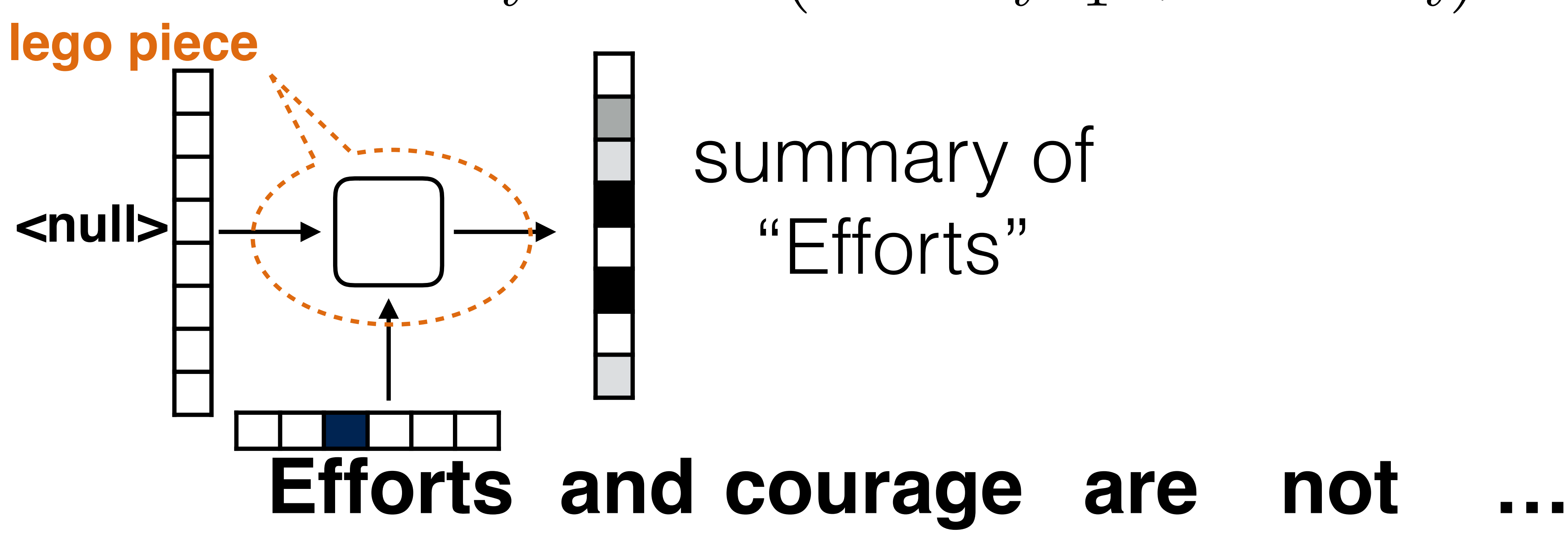
Efforts and courage are not ...

Example: encoding sentences

- Easy to introduce adjustable “lego pieces” and optimize them for end-to-end performance

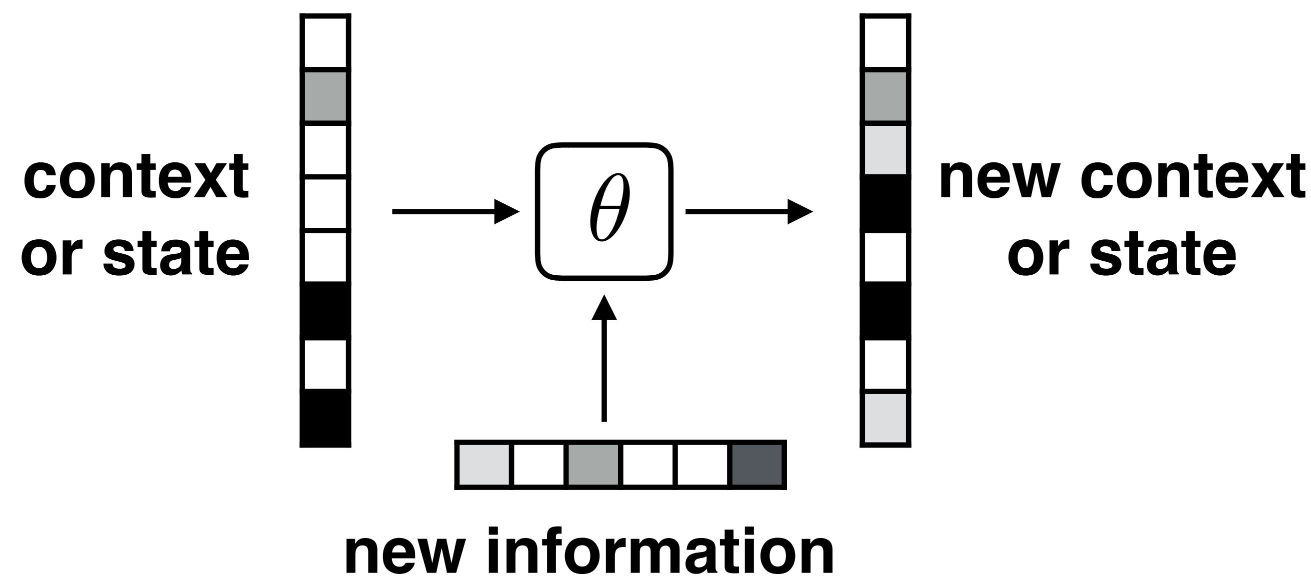


$$s_t = \tanh(W^{s,s} s_{t-1} + W^{s,x} x_t)$$



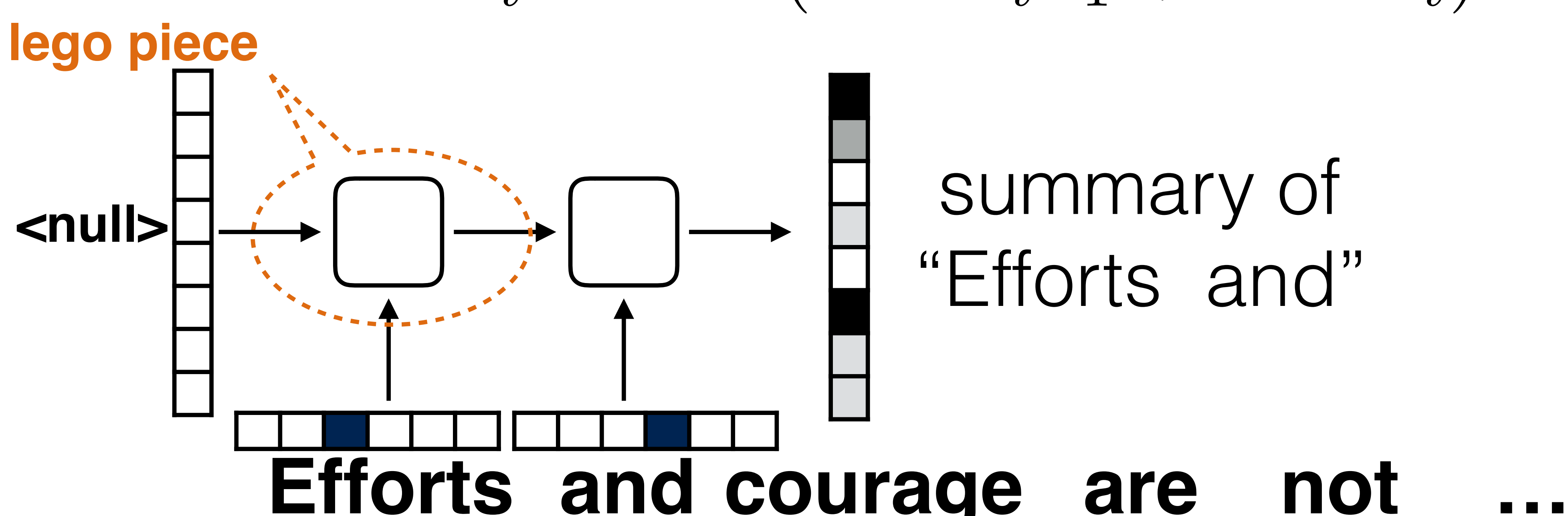
Example: encoding sentences

- Easy to introduce adjustable “lego pieces” and optimize them for end-to-end performance



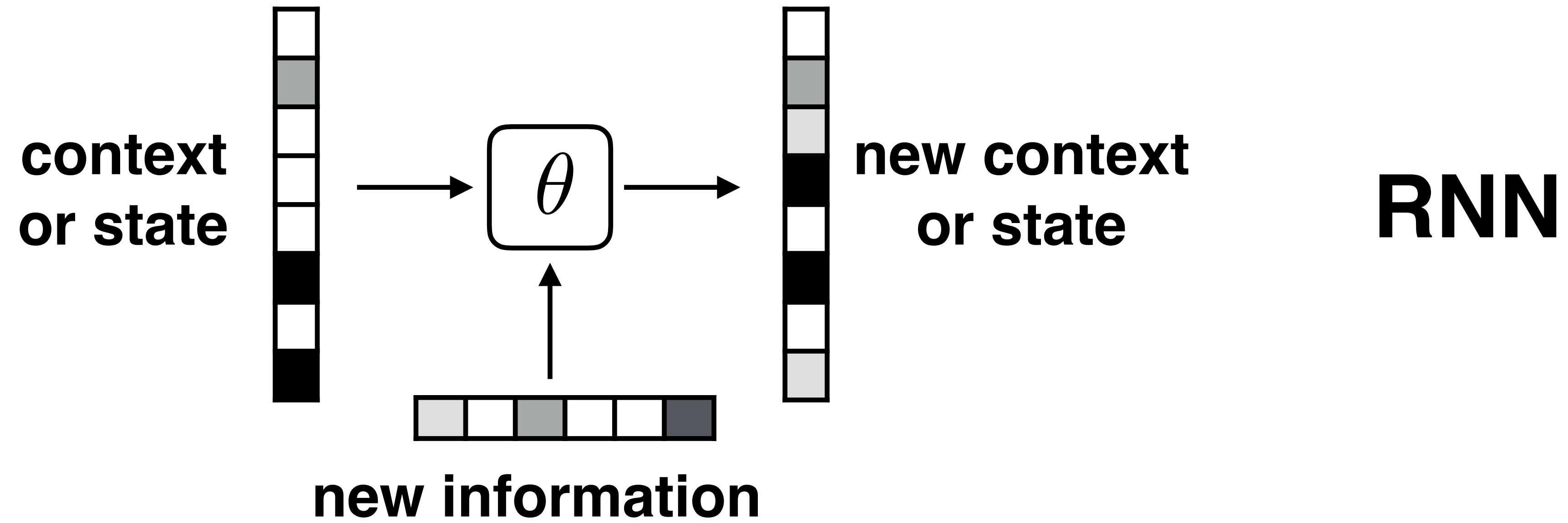
RNN

$$s_t = \tanh(W^{s,s} s_{t-1} + W^{s,x} x_t)$$

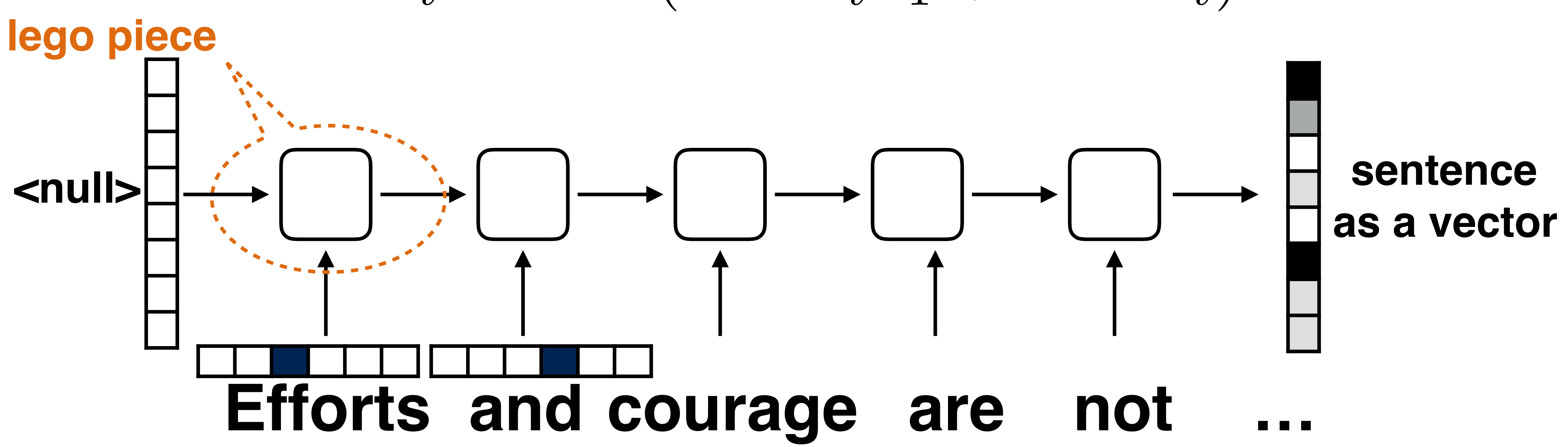


Example: encoding sentences

- Easy to introduce adjustable “lego pieces” and optimize them for end-to-end performance

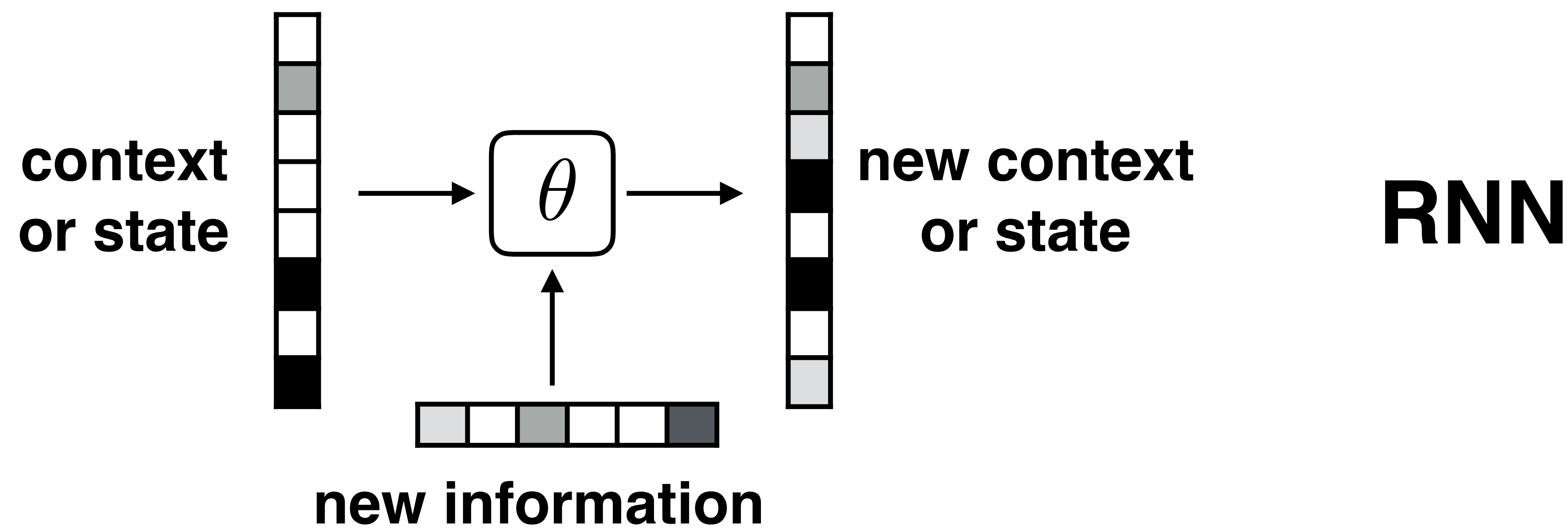


$$s_t = \tanh(W^{s,s} s_{t-1} + W^{s,x} x_t)$$

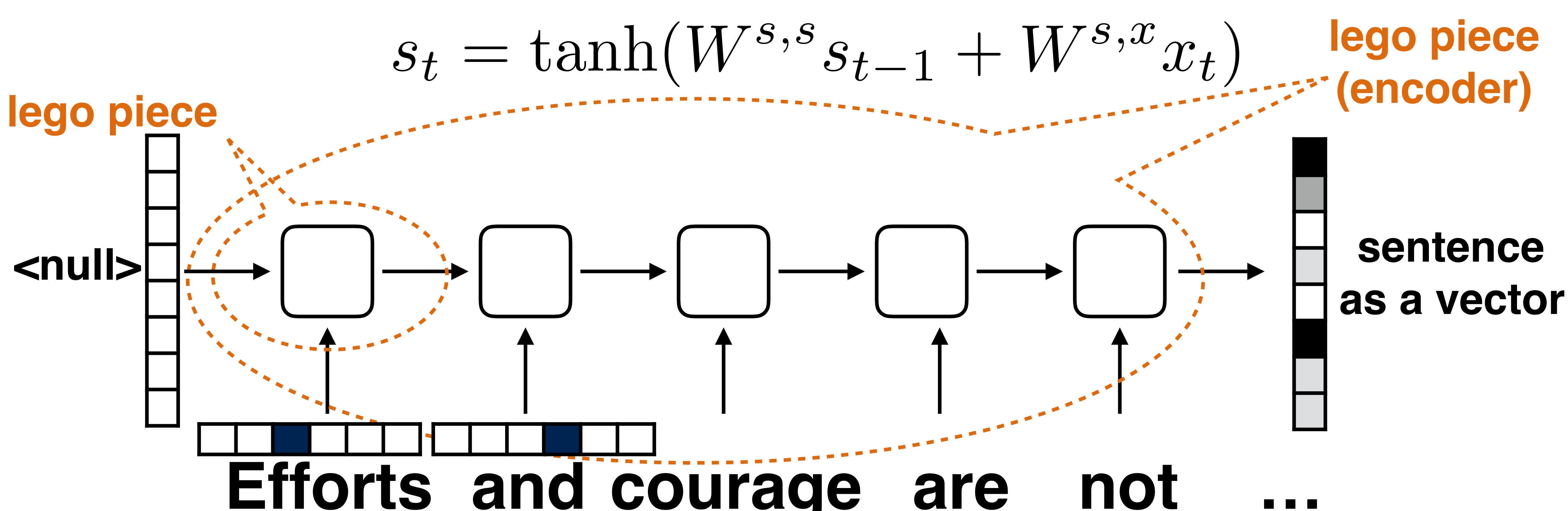


Example: encoding sentences

- Easy to introduce adjustable “lego pieces” and optimize them for end-to-end performance

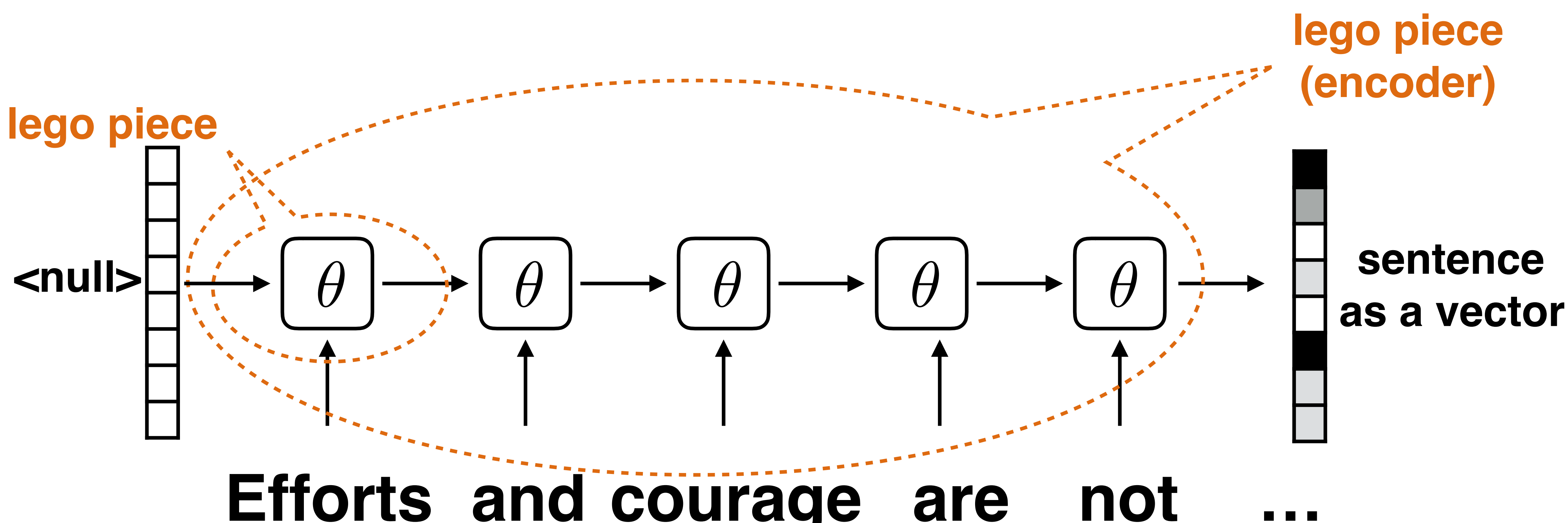


$$s_t = \tanh(W^{s,s}s_{t-1} + W^{s,x}x_t)$$



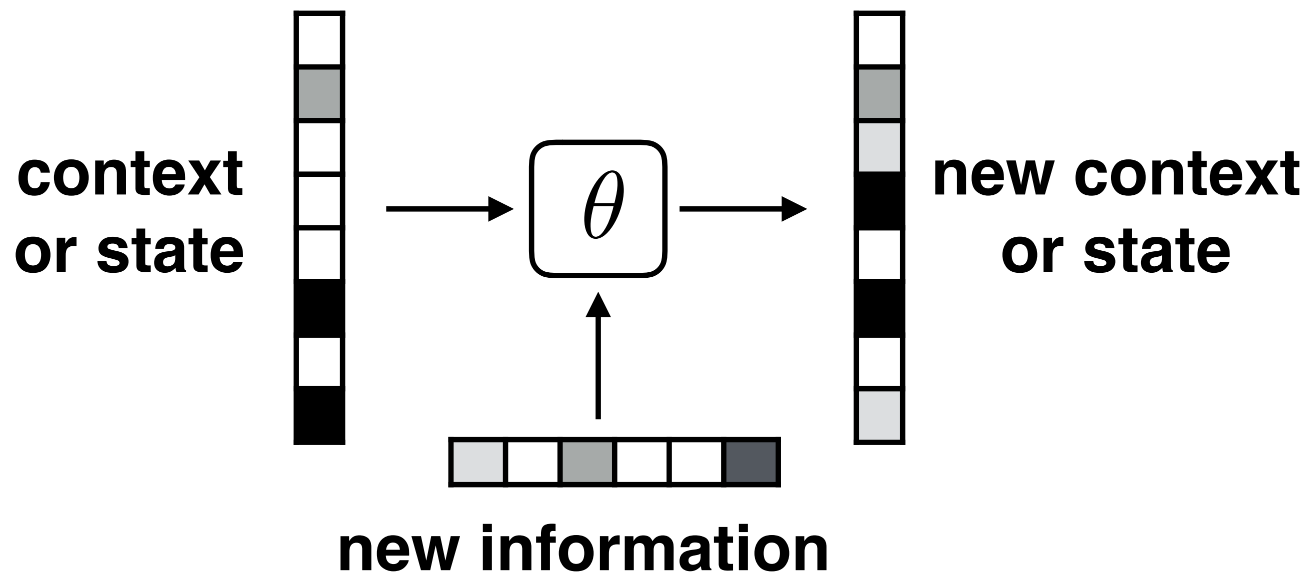
Example: encoding sentences

- There are three differences between the encoder (unfolded RNN) and a standard feed-forward architecture
  - input is received at each layer (per word), not just at the beginning as in a typical feed-forward network
  - the number of layers varies, and depends on the length of the sentence
  - parameters of each layer (representing an application of an RNN) are shared (same RNN at each step)



What's in the box?

- We can make the RNN more sophisticated…

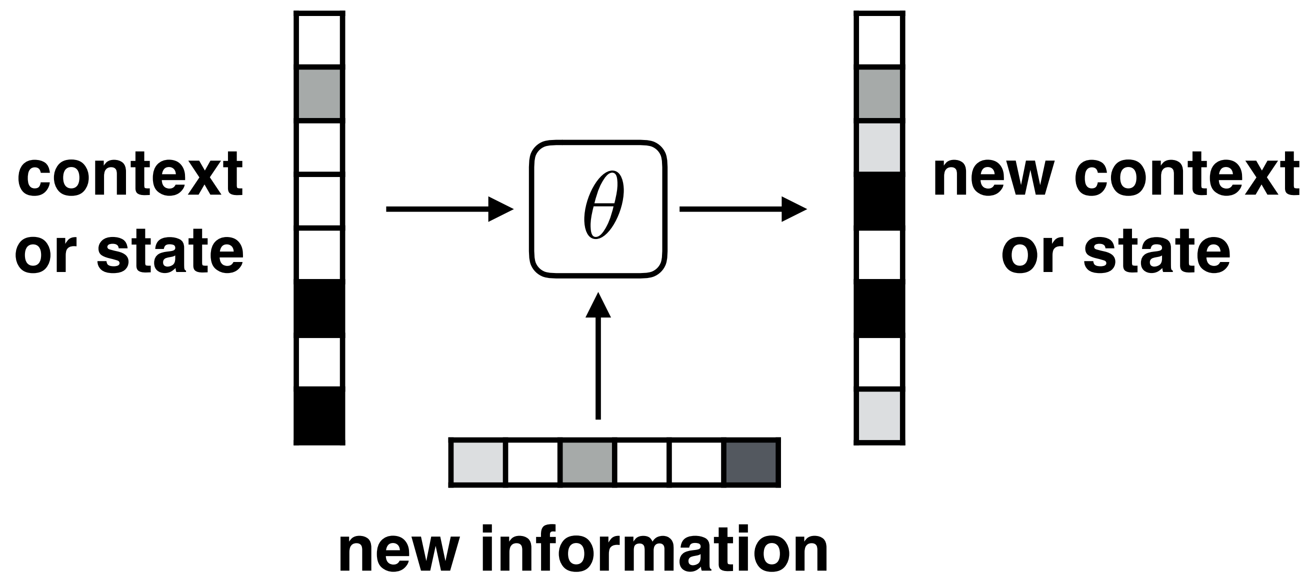


basic RNN

$$s_t = \tanh(W^{s,s} s_{t-1} + W^{s,x} x_t)$$

What's in the box?

- We can make the RNN more sophisticated…



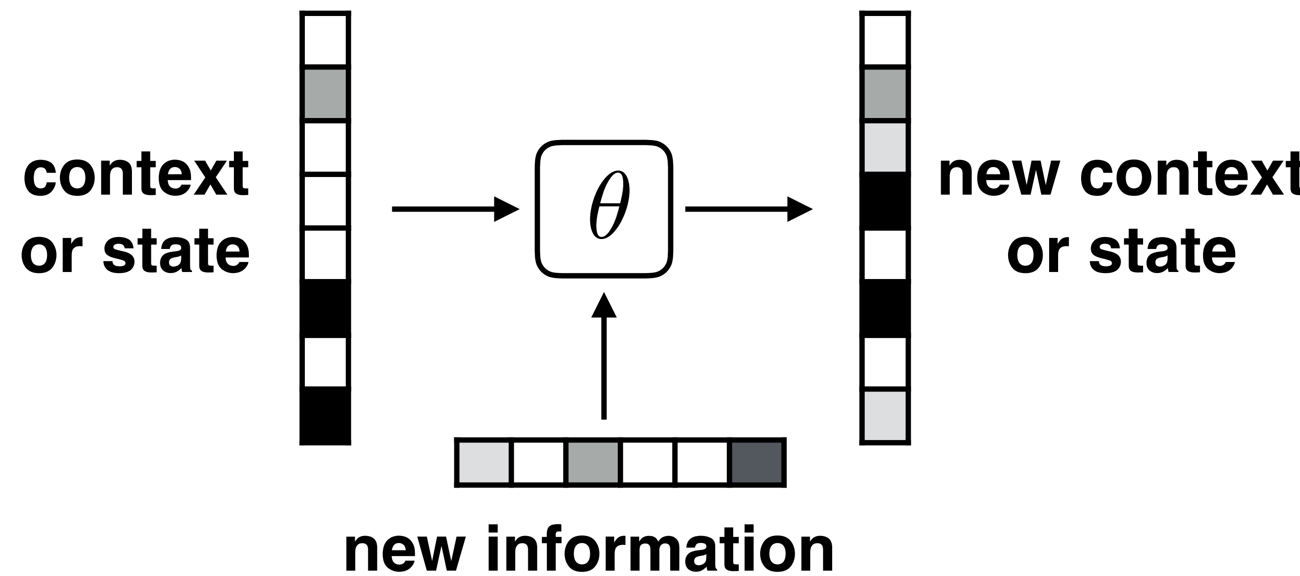
simple gated RNN

$$g_t = \text{sigmoid}(W^{g,s} s_{t-1} + W^{g,x} x_t)$$

$$s_t = (1 - g_t) \odot s_{t-1} + g_t \odot \tanh(W^{s,s} s_{t-1} + W^{s,x} x_t)$$

What's in the box?

- We can make the RNN more sophisticated…



LSTM
(one of many)

$$f_t = \text{sigmoid}(W^{f,h} h_{t-1} + W^{f,x} x_t)$$ forget gate

$$i_t = \text{sigmoid}(W^{i,h} h_{t-1} + W^{i,x} x_t)$$ input gate

$$o_t = \text{sigmoid}(W^{o,h} h_{t-1} + W^{o,x} x_t)$$ output gate

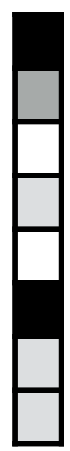
$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W^{c,h} h_{t-1} + W^{c,x} x_t)$$ memory cell

$$h_t = o_t \odot \tanh(c_t)$$ visible state

Decoding (into a sentence)

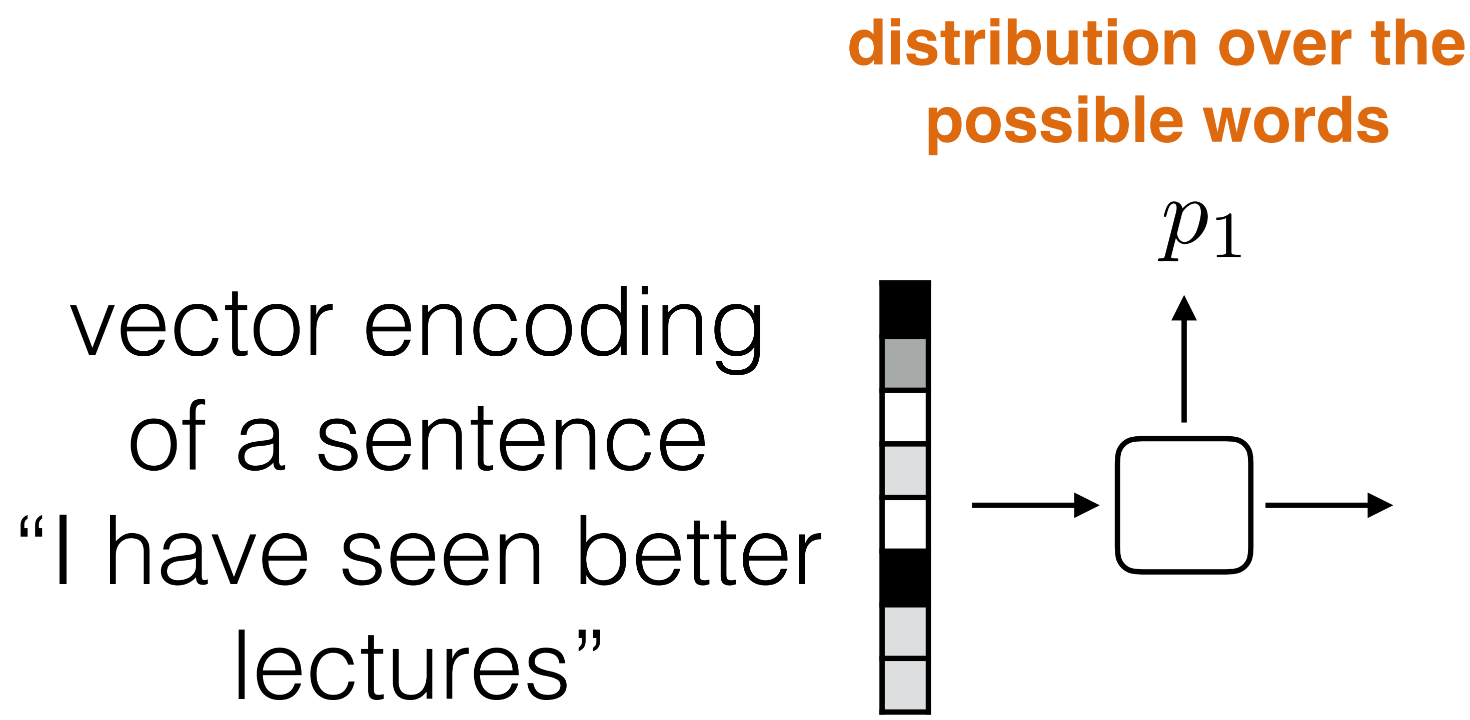
- Our RNN now needs to also produce an output (e.g., a word) as well as update its state

vector encoding
of a sentence
“I have seen better
lectures”



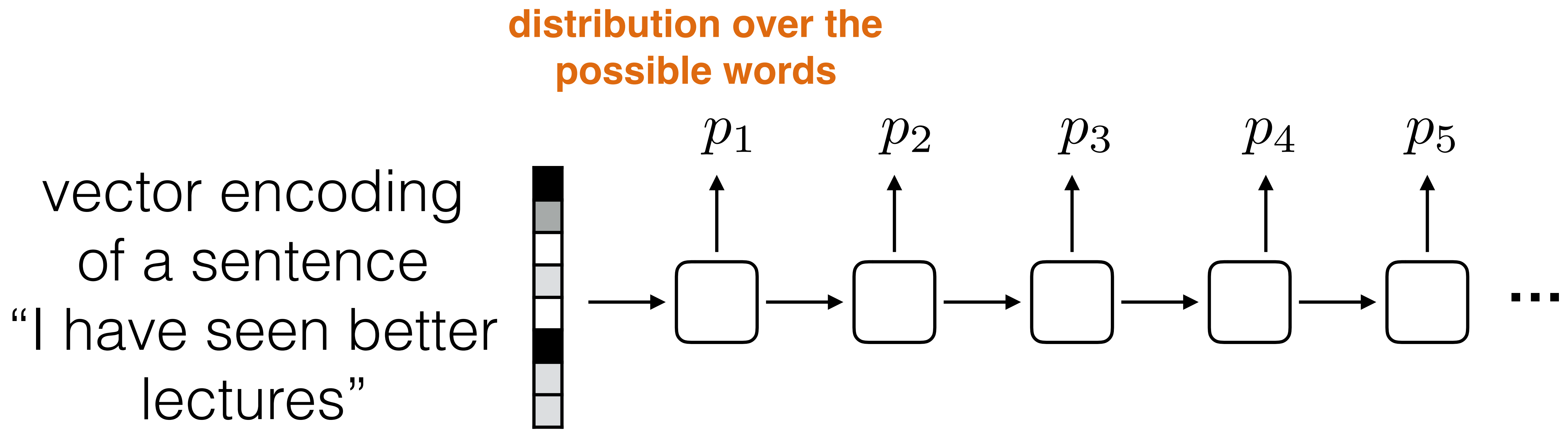
Decoding (into a sentence)

- Our RNN now needs to also produce an output (e.g., a word) as well as update its state



Decoding (into a sentence)

- Our RNN now needs to also produce an output (e.g., a word) as well as update its state

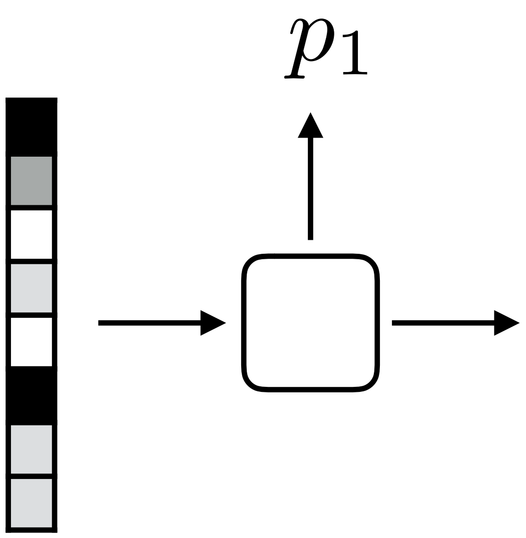


Decoding (into a sentence)

- Our RNN now needs to also produce an output (e.g., a word) as well as update its state

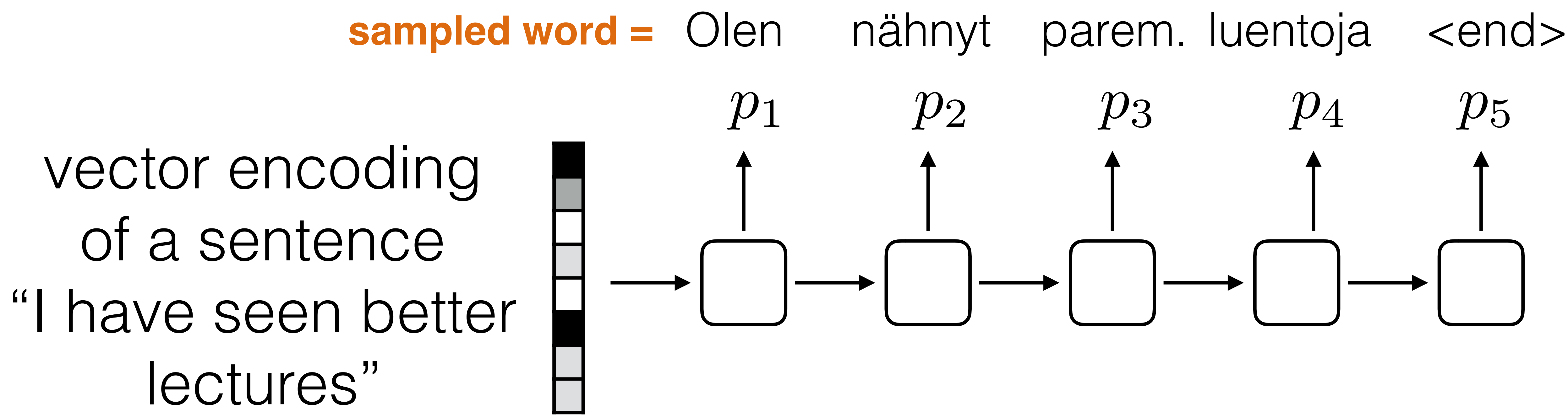
sampled word = Olen

vector encoding of a sentence "I have seen better lectures"



Decoding (into a sentence)

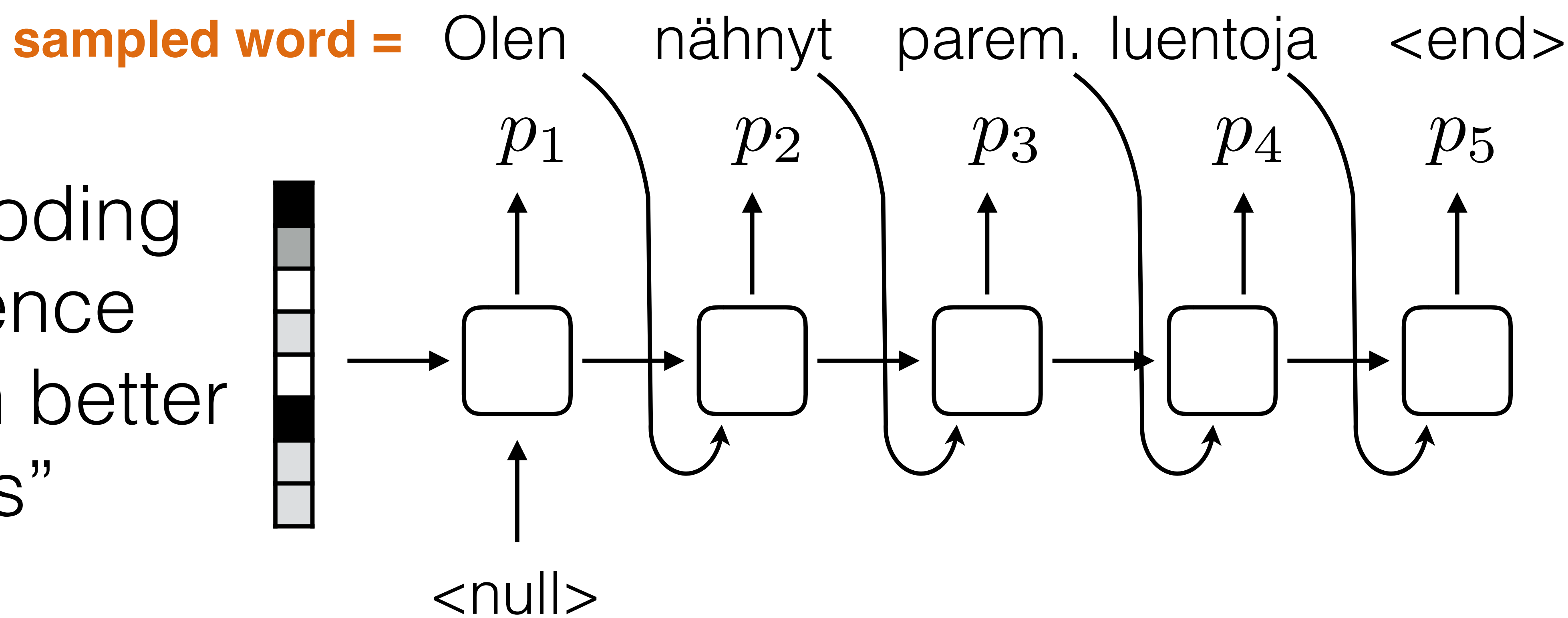
- Our RNN now needs to also produce an output (e.g., a word) as well as update its state



Decoding (into a sentence)

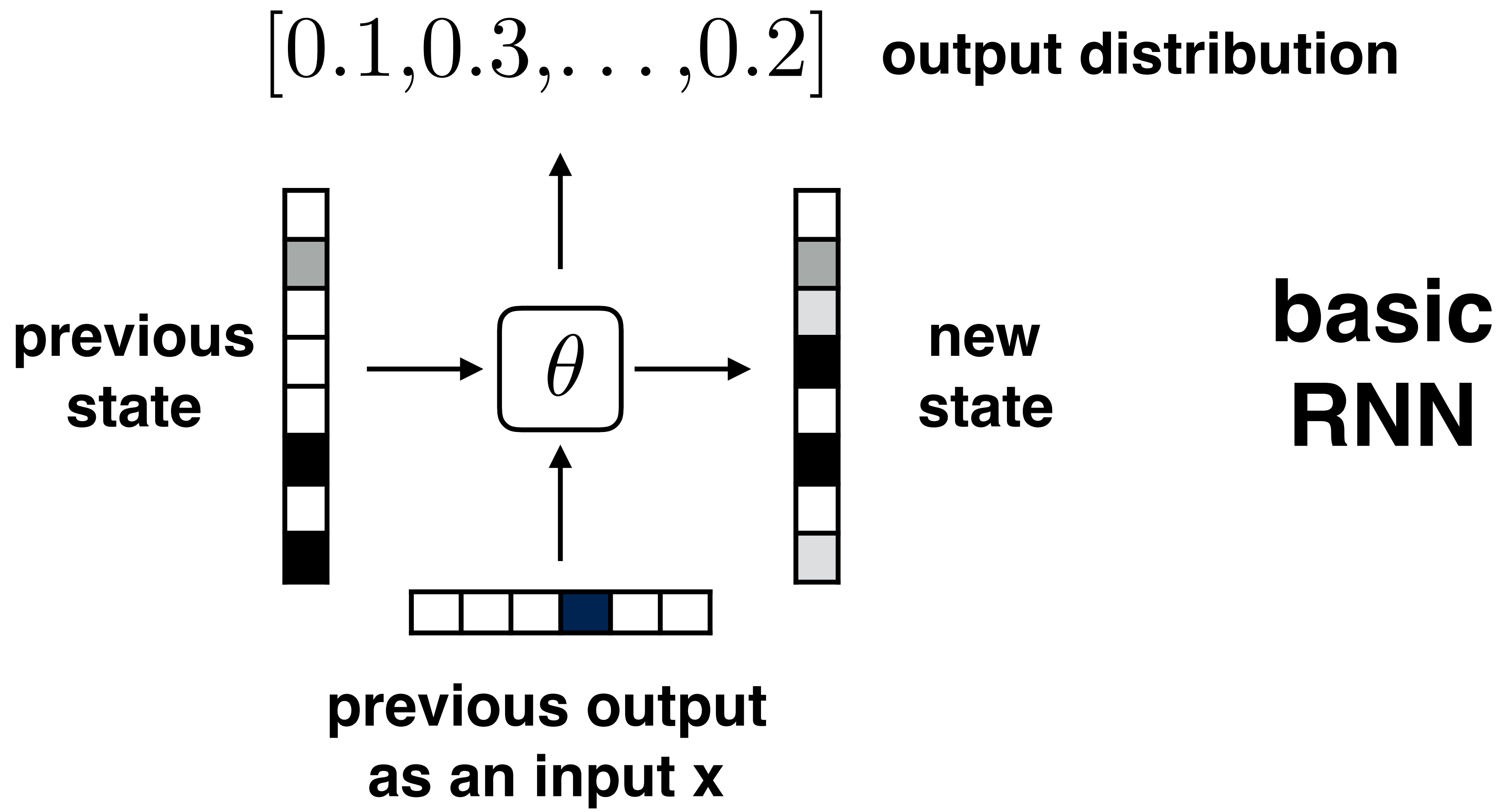
- Our RNN now needs to also produce an output (e.g., a word) as well as update its state
- The output is fed in as an input (to gauge what's left)

vector encoding of a sentence "I have seen better lectures"



Decoding: what's in the box

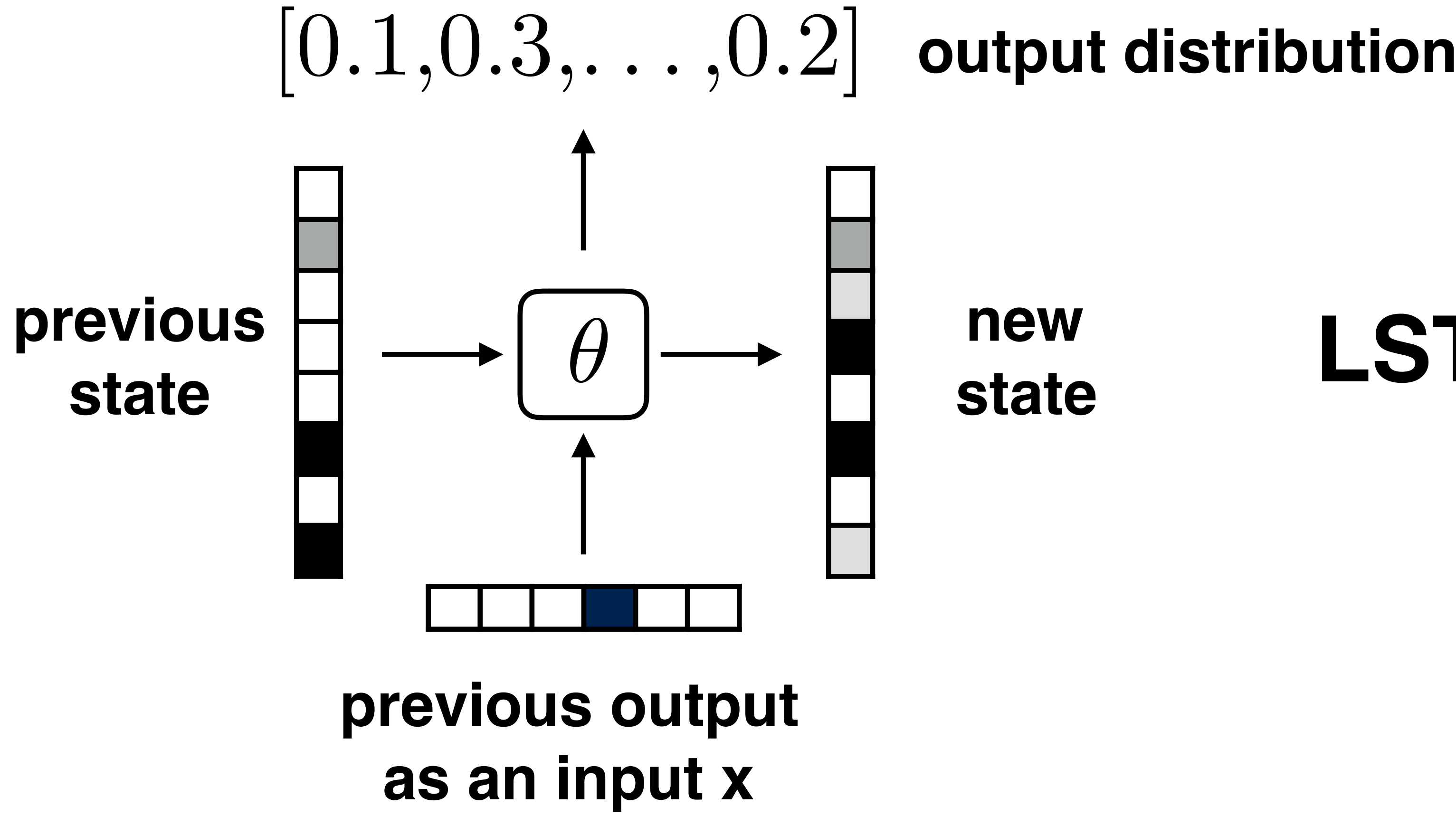
- Our RNN now needs to also produce an output (e.g., a word) as well as update its state



$$s_t = \tanh(W^{s,s} s_{t-1} + W^{s,x} x_t)$$ state

$$p_t = \text{softmax}(W^o s_t)$$ output distribution

Decoding: what's in the box



LSTM

$$f_t = \mathrm{sigmoid}(W^{f,h} h_{t-1} + W^{f,x} x_t)$$ forget gate

$$i_t = \mathrm{sigmoid}(W^{i,h} h_{t-1} + W^{i,x} x_t)$$ input gate

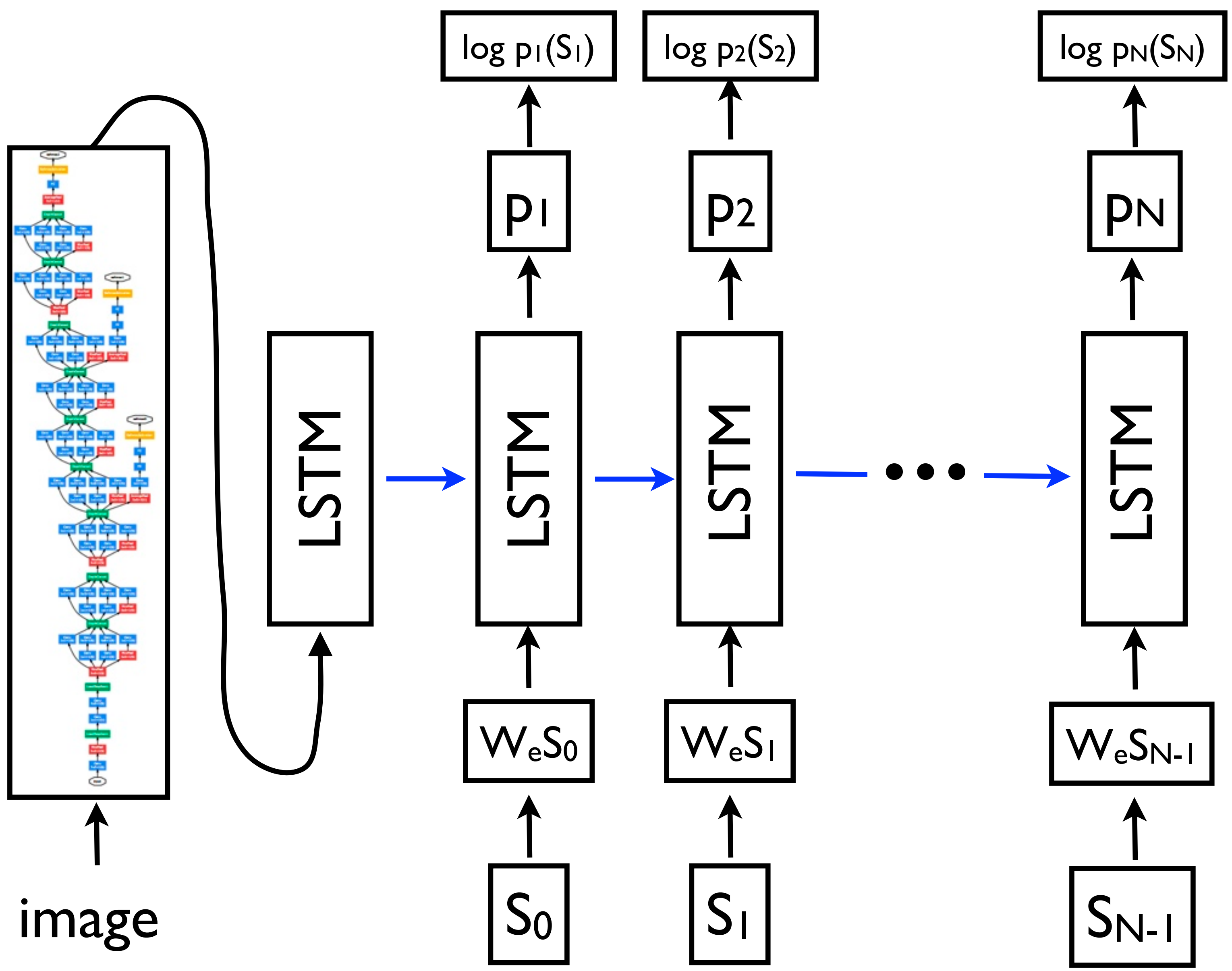
$$o_t = \mathrm{sigmoid}(W^{o,h} h_{t-1} + W^{o,x} x_t)$$ output gate

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W^{c,h} h_{t-1} + W^{c,x} x_t)$$ memory cell

$$h_t = o_t \odot \tanh(c_t)$$ visible state

$$p_t = \mathrm{softmax}(W^o h_t)$$ output distribution

Mapping images to text



Examples

A person riding a motorcycle on a dirt road.

Two dogs play in the grass.

A skateboarder does a trick on a ramp.

A dog is jumping to catch a frisbee.

A group of young people playing a game of frisbee.

Two hockey players are fighting over the puck.

A little girl in a pink hat is blowing bubbles.

A refrigerator filled with lots of food and drinks.

A herd of elephants walking across a dry grass field.

A close up of a cat laying on a couch.

A red motorcycle parked on the side of the road.

A yellow school bus parked in a parking lot.

Describes without errors | Describes with minor errors | Somewhat related to the image | Unrelated to the image

Key things

- Multi-way classification (explained on the board)
  - softmax, loss
- Neural networks for sequences
  - encoding/decoding
- RNNs, unfolded
  - state evolution, gates
  - relation to feed-forward neural networks
  - back-propagation (conceptually)
- Issues: vanishing/exploding gradient
- LSTM (operationally)