Outline

- RNNs and vanishing/exploding gradients
- Solutions

RNNs: Advantages

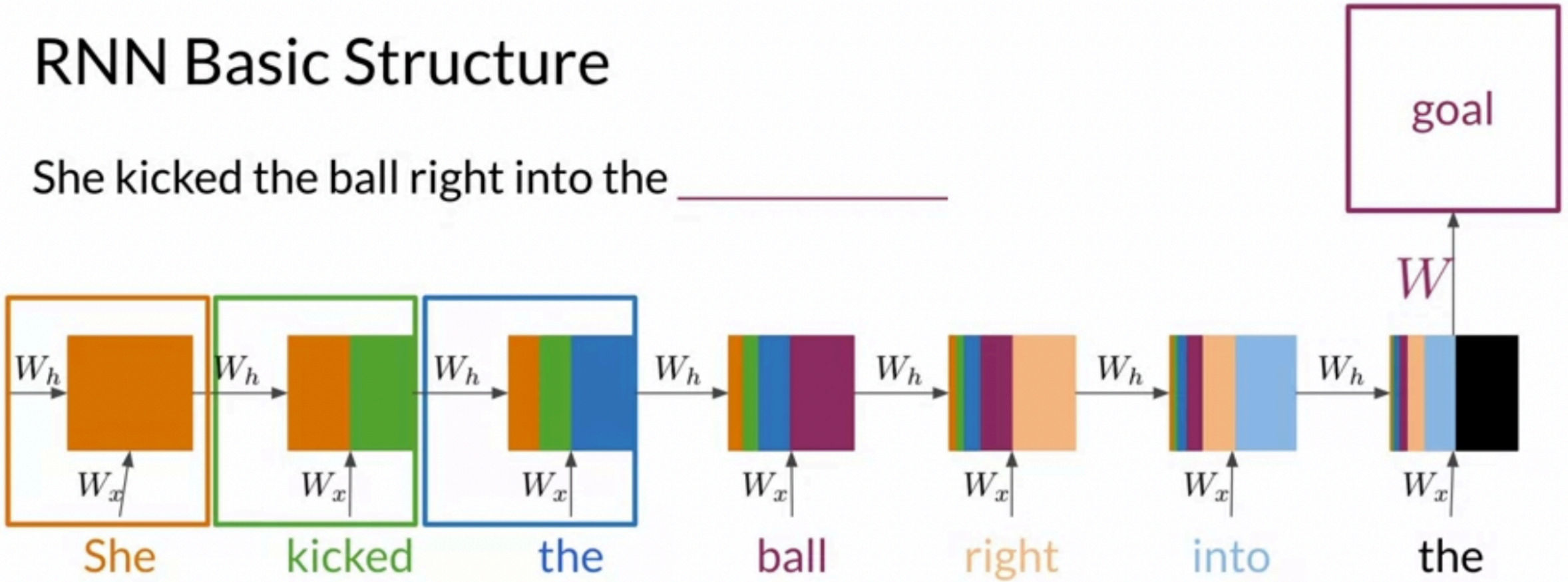
- Captures dependencies within a short range
- Takes up less RAM than other n-gram models

RNNs: Disadvantages

- Struggles with longer sequences
- Prone to vanishing or exploding gradients

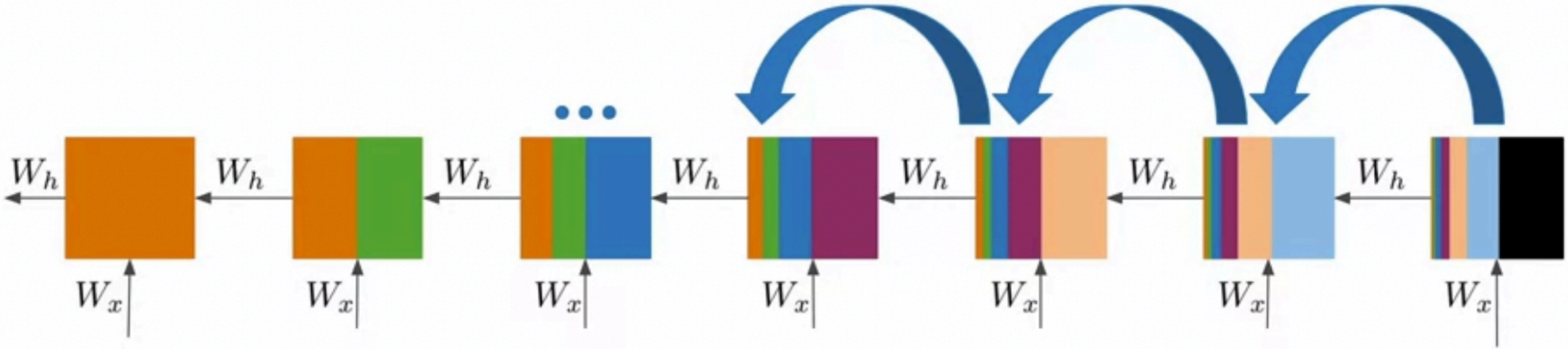
RNN Basic Structure

She kicked the ball right into the _____



Learnable parameters

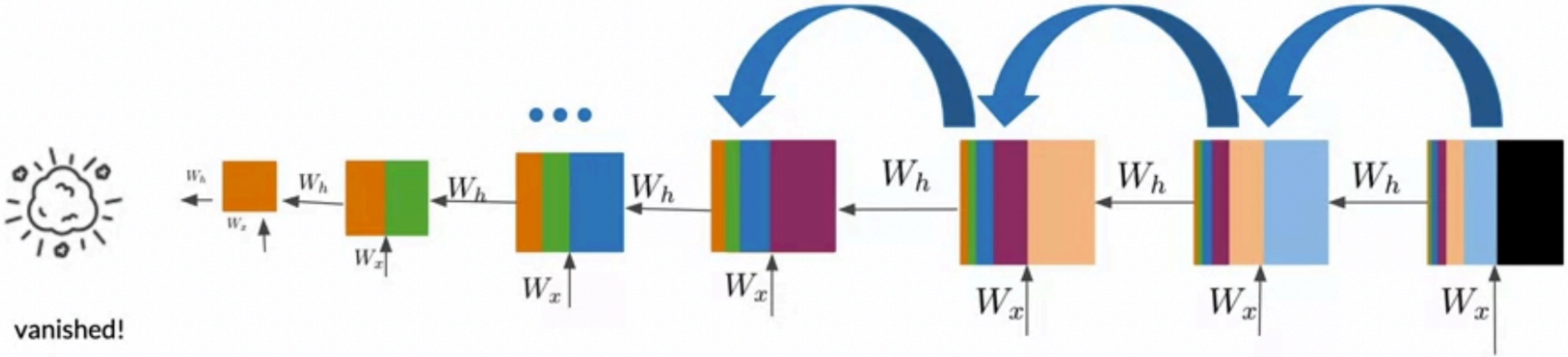
Backpropagation through time



between 0 and 1

between -1 and 1

The vanishing gradient problem



Solving for vanishing or exploding gradients

- Identity RNN with ReLU activation

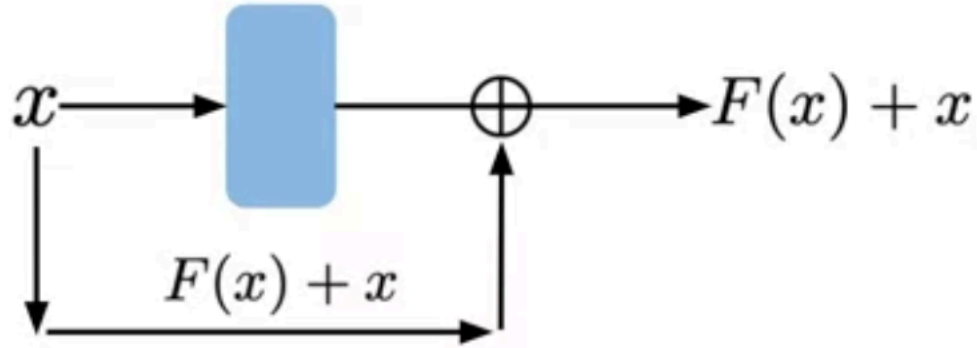
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

-1 ⟶ 0

- Gradient clipping

32 ⟶ 25

- Skip connections



Outline

- Meet the Long short-term memory unit!
- LSTM architecture
- Applications

LSTMs: a memorable solution

- Learns when to remember and when to forget
- Basic anatomy:
  - A cell state
  - A hidden state with three gates
  - Loops back again at the end of each time step
- Gates allow gradients to flow unchanged

LSTMs: Based on previous understanding

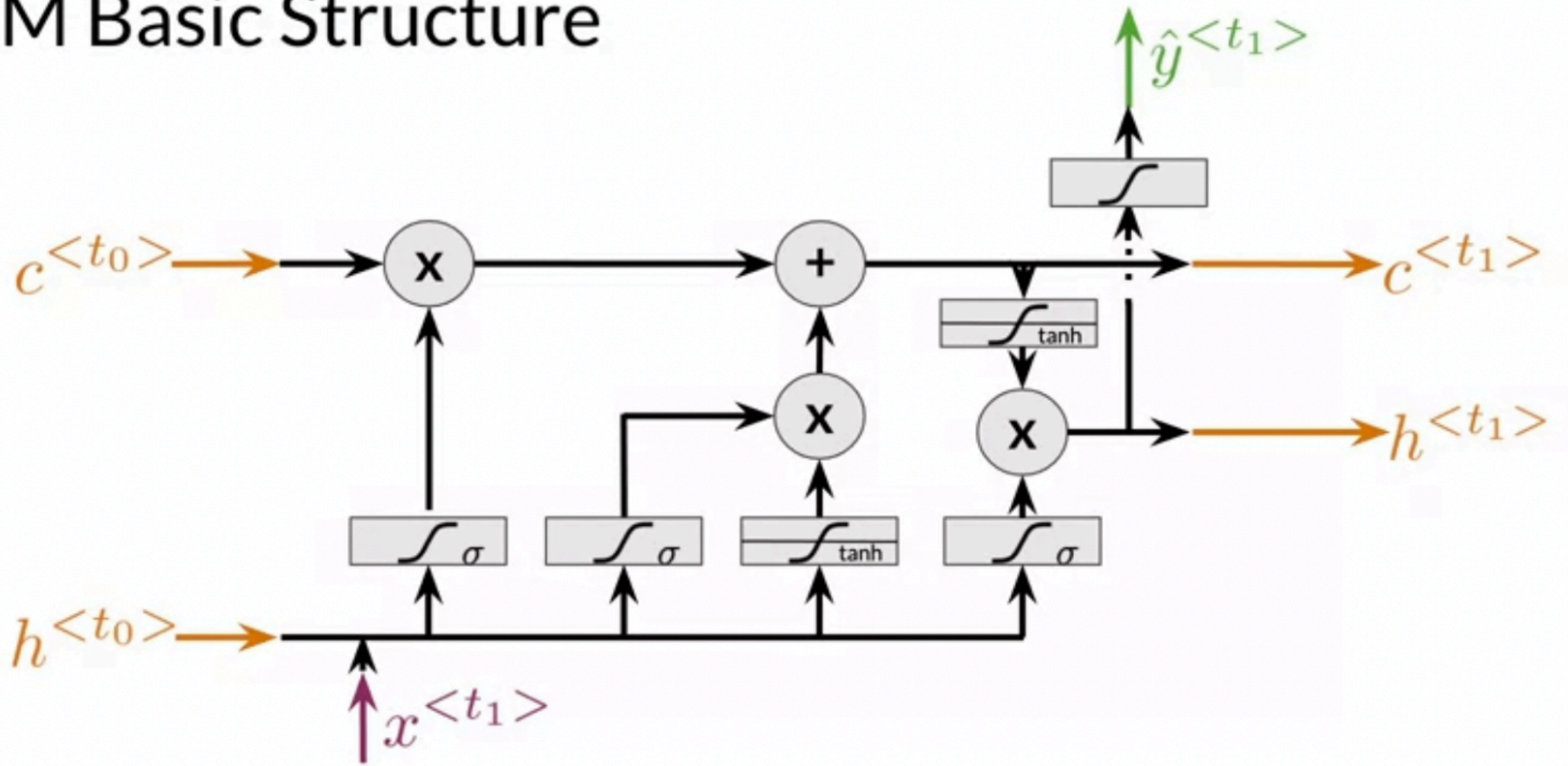
Cell state = before conversation

Forget gate = beginning of conversation

Input gate = thinking of a response
Output gate = responding

Updated cell state = after conversation

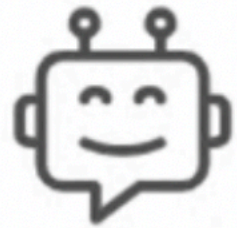
LSTM Basic Structure



Applications of LSTMs

Next-character prediction

Chatbots



Music composition

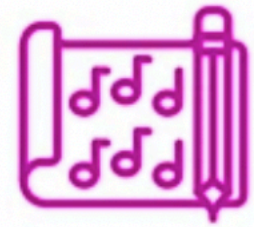


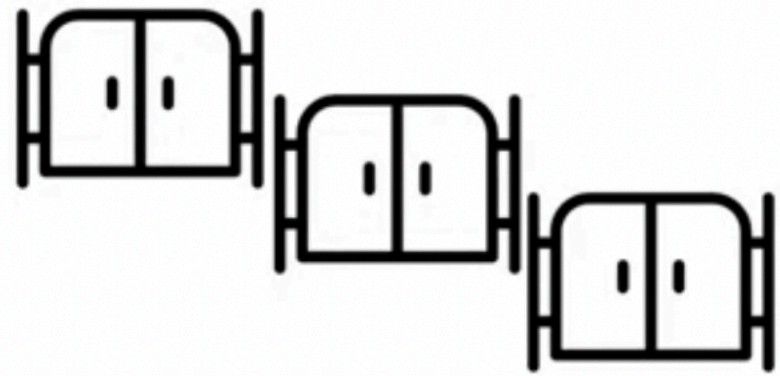
Image captioning

Speech recognition

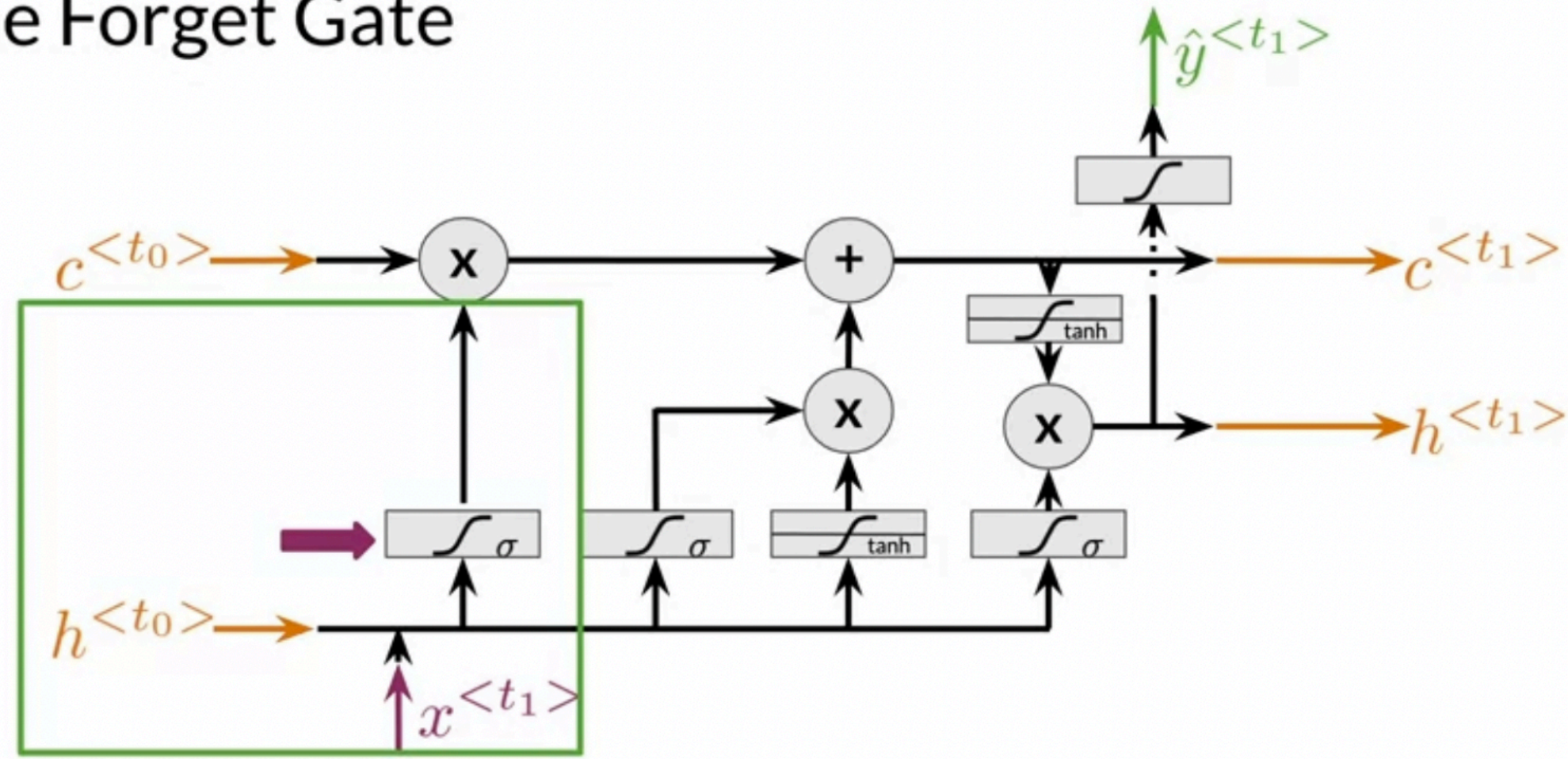
Summary

- LSTMs offer a solution to vanishing gradients

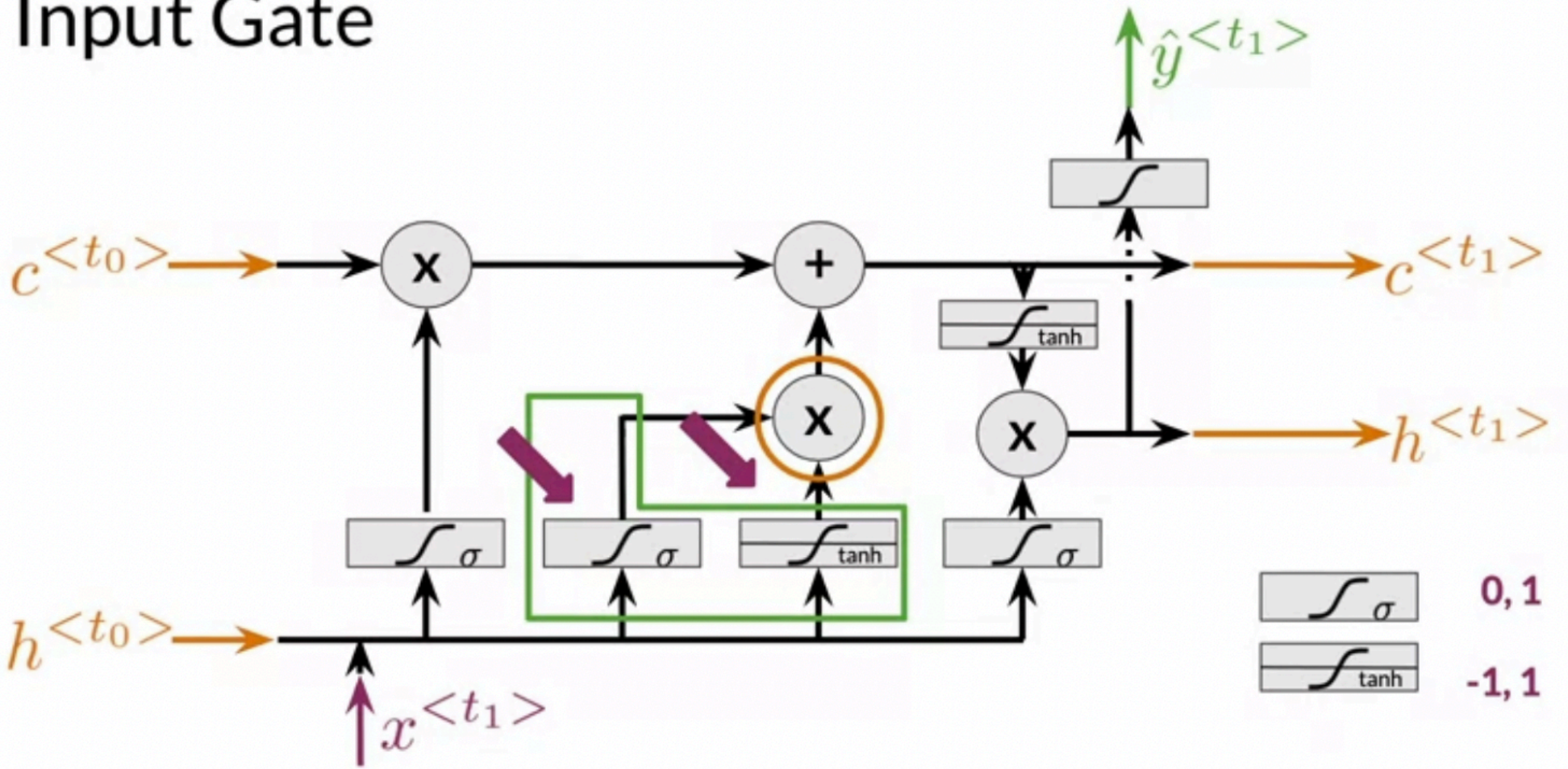
- Typical LSTMs have a cell and three gates:
  - Forget gate
  - Input gate
  - Output gate



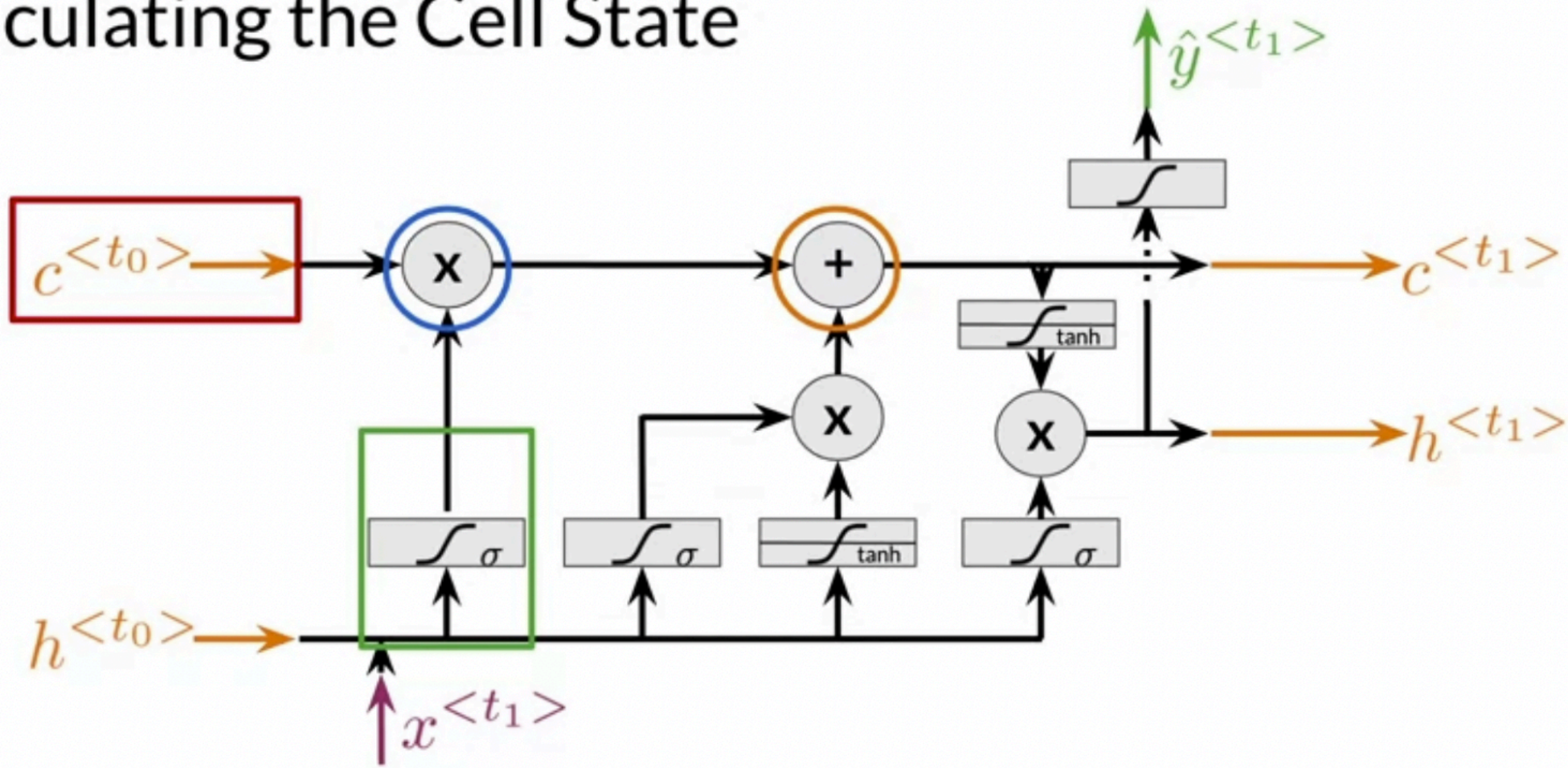
The Forget Gate



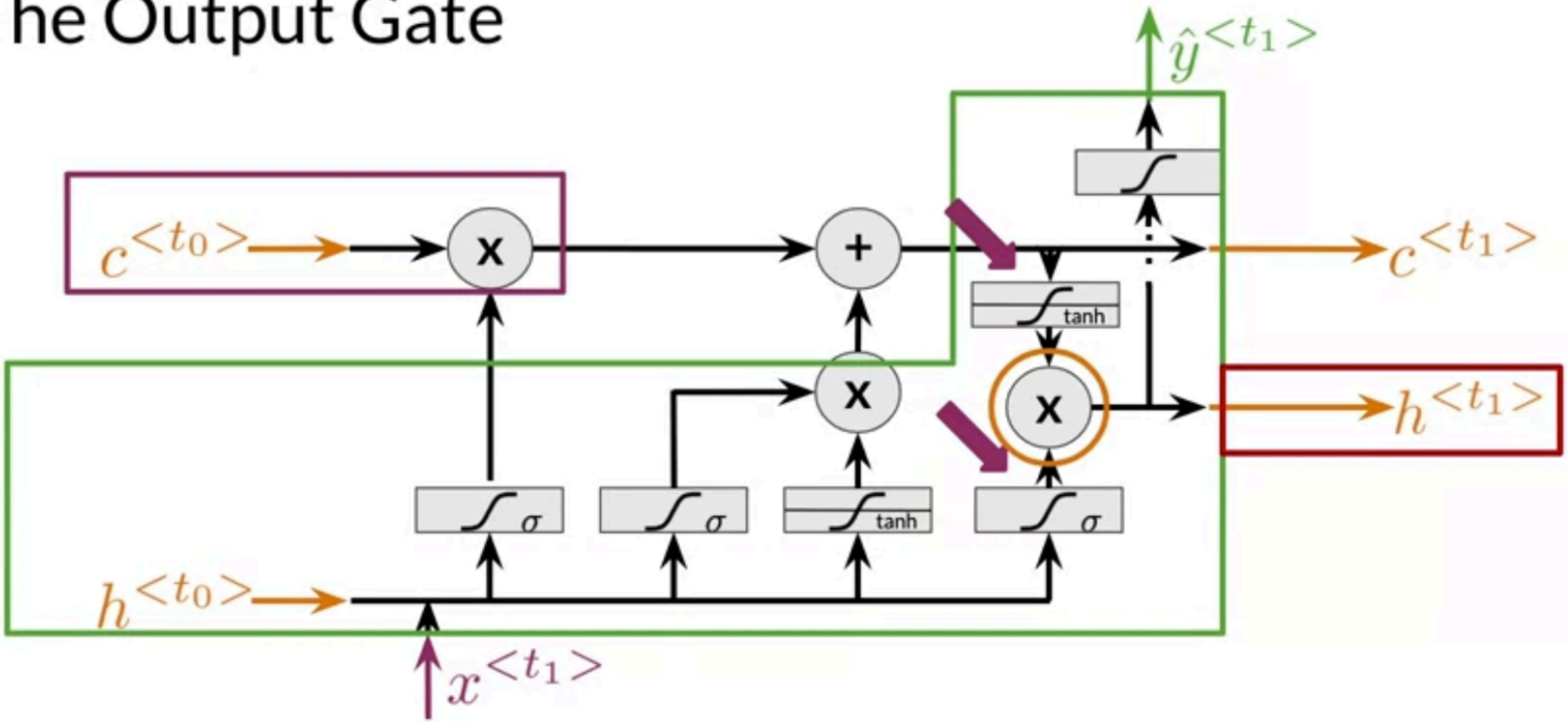
The Input Gate



Calculating the Cell State



The Output Gate



Summary

- LSTMs use a series of gates to decide which information to keep:
  - Forget gate decides what to keep
  - Input gate decides what to add
  - Output gate decides what the next hidden state will be

- One time step is completed after updating the states

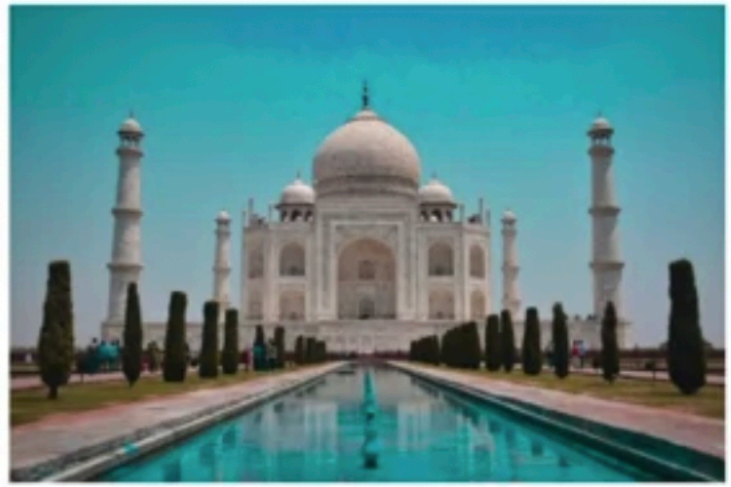
What is Named Entity Recognition?

- Locates and extracts predefined entities from text
- Places, organizations, names, time and dates

Types of Entities

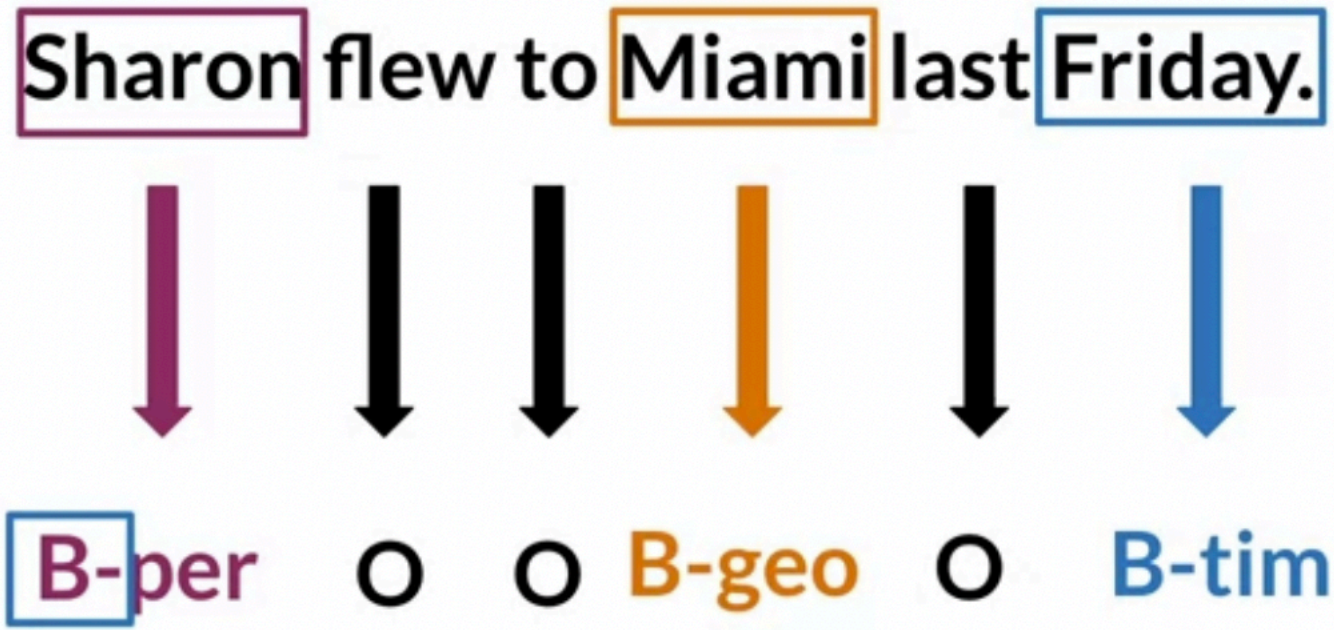
Thailand:
Geographical

Google:
Organization



Indian:
Geopolitical

Example of a labeled sentence



Applications of NER systems

- Search engine efficiency
- Recommendation engines
- Customer service
- Automatic trading

Outline

- Convert words and entity classes into arrays
- Token padding
- Create a data generator

Processing data for NERs

- Assign each class a number

1	2	3
B-per	B-geo	B-tim

Processing data for NERs

- Assign each class a number
- Assign each word a number

Sharon flew to Miami last Friday.

[4282, 853, 187, 5388, 2894, 7]

B-per O O B-geo O B-tim

Token padding

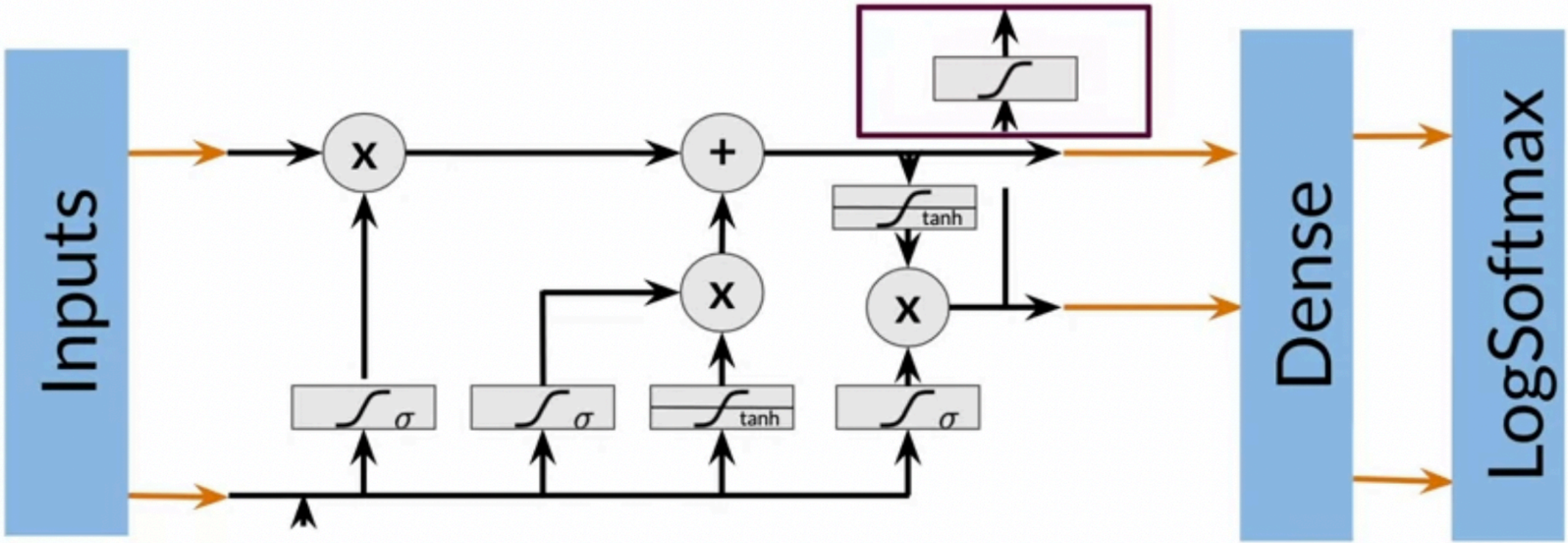
For LSTMs, all sequences need to be the same size.

- Set sequence length to a certain number
- Use the <PAD> token to fill empty spaces

Training the NER

1. Create a tensor for each input and its corresponding number
2. Put them in a batch ⟶ 64, 128, 256, 512 ...
3. Feed it into an LSTM unit
4. Run the output through a dense layer
5. Predict using a log softmax over K classes

Training the NER



Layers in Trax

```
model = tl.Serial(
    tl.Embedding(),
    tl.LSTM(),
    tl.Dense()
    tl.LogSoftmax()
  )
```

Summary

- Convert words and entities into same-length numerical arrays
- Train in batches for faster processing
- Run the output through a final layer and activation

Evaluating the model

1. Pass test set through the model
2. Get arg max across the prediction array
3. Mask padded tokens
4. Compare outputs against test labels

Evaluating the model in Python

```
def evaluate_model(test_sentences, test_labels, model):
    pred = model(test_sentences)
    outputs = np.argmax(pred, axis=2)
    mask = ...
    accuracy = np.sum(outputs==test_labels)/float(np.sum(mask))

    return accuracy
```

Summary

- If padding tokens, remember to mask them when computing accuracy
- Coding assignment!