LSTM and Named Entity Recognition

FPT Education
FPT UNIVERSITY

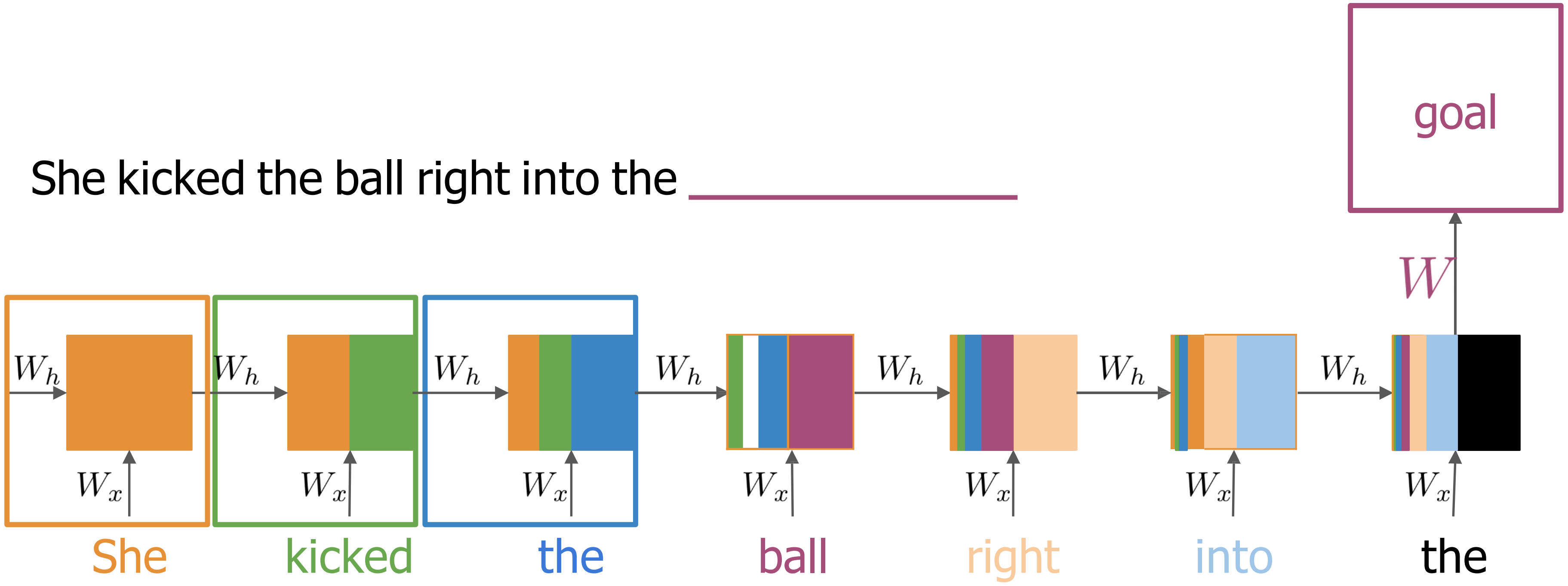
- RNNs and Vanishing Gradients
- LSTMs
- Named Entity Recognition

RNNs and Vanishing Gradients

- RNNs
  - Backprop through time
  - RNNs and vanishing/exploding gradients
  - Solutions
- Advantages
  - Captures dependencies within a short range
  - Takes up less RAM than other n-gram models
- Disadvantages
  - Struggles to capture long term dependencies
  - Prone to vanishing or exploding gradients

RNN Basic Structure

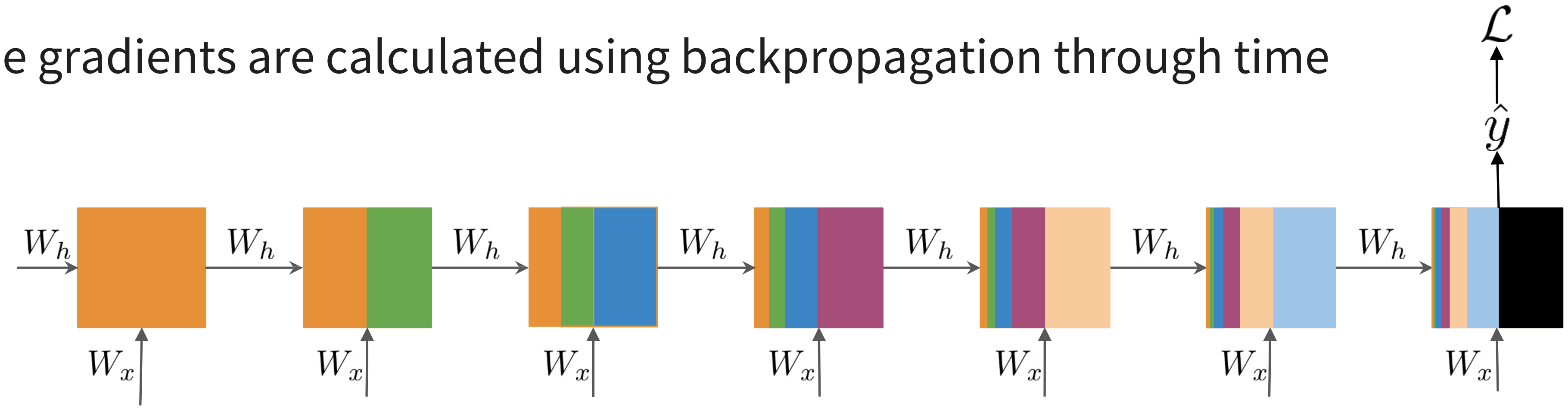
She kicked the ball right into the ___________

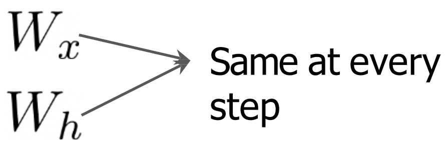


Learnable parameters

Backpropagation through time

- The gradients are calculated using backpropagation through time





$$\frac{\partial L}{\partial W_h} \propto \sum_{1 \leq k \leq t} \left(\prod_{t \geq i > k} \frac{\partial h_i}{\partial h_{i-1}} \right) \frac{\partial h_k}{\partial W_h}$$

Gradient is proportional to a sum of partial derivative products

Backpropagation through time

FPT Education
FPT UNIVERSITY

$$\frac{\partial L}{\partial W_h} \propto \sum_{1 \leq k \leq t} \boxed{\left(\boxed{\prod_{t \geq i > k} \frac{\partial h_i}{\partial h_{i-1}}} \right) \frac{\partial h_k}{\partial W_h}}$$

Contribution of hidden state ***k***

Length of the product proportional to how far ***k*** is from ***t***

$$\boxed{\frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial h_{t-2}} \frac{\partial h_{t-2}}{\partial h_{t-3}} \frac{\partial h_{t-3}}{\partial h_{t-4}} \frac{\partial h_{t-4}}{\partial h_{t-5}} \frac{\partial h_{t-5}}{\partial h_{t-6}} \frac{\partial h_{t-6}}{\partial h_{t-7}} \frac{\partial h_{t-7}}{\partial h_{t-8}} \frac{\partial h_{t-8}}{\partial h_{t-9}} \frac{\partial h_{t-9}}{\partial h_{t-10}} \frac{\partial h_{t-10}}{\partial W_h}}$$

Contribution of hidden state ***t-10***

- The contribution to the gradient of a hidden state that is 10 steps away from step t

Backpropagation through time

$$\frac{\partial L}{\partial W_h} \propto \sum_{1 \le k \le t} \boxed{\left(\boxed{\prod_{t \ge i > k} \frac{\partial h_i}{\partial h_{i-1}}} \right) \frac{\partial h_k}{\partial W_h}}$$

Contribution of hidden state ***k***

Length of the product proportional to how far ***k*** is from ***t***

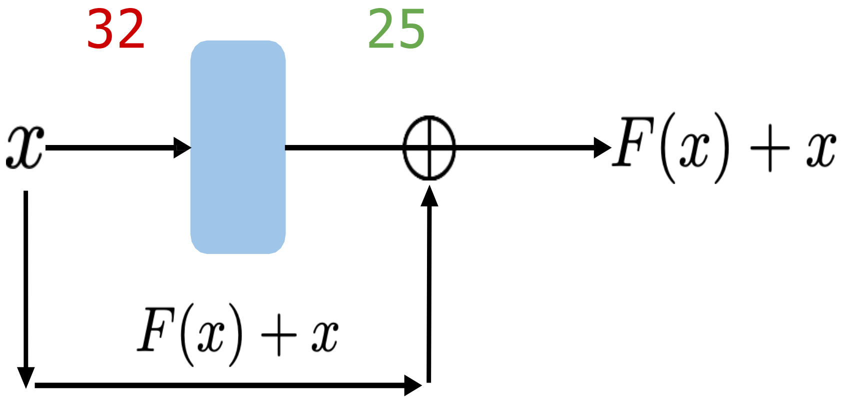
Partial derivatives <1	Contribution goes to 0	Vanishing Gradient
Partial derivatives >1	Contribution goes to infinity	**Exploding** Gradient

Solving for vanishing or exploding gradients

- Identity RNN with ReLU activation
- Gradient clipping
- Skip connections

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

-1



LSTMs

- Outline
  - Meet the Long short-term memory unit
  - LSTM architecture
  - Applications
- Memorable solution
  - Learns when to remember and when to forget
  - Basic anatomy:
    - A cell state
    - A hidden state
    - Multiple gates
  - Gates allow gradients to avoid vanishing and exploding

LSTMs: Based on previous understanding

Starting point with some irrelevant information

↓

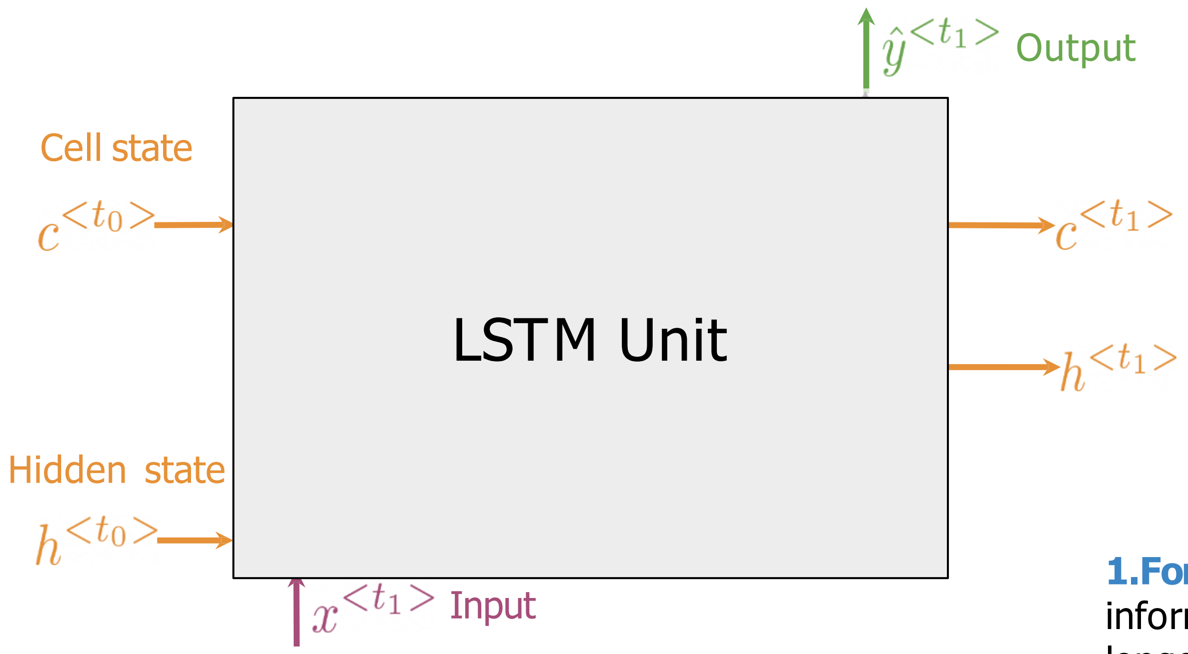
Cell and Hidden States

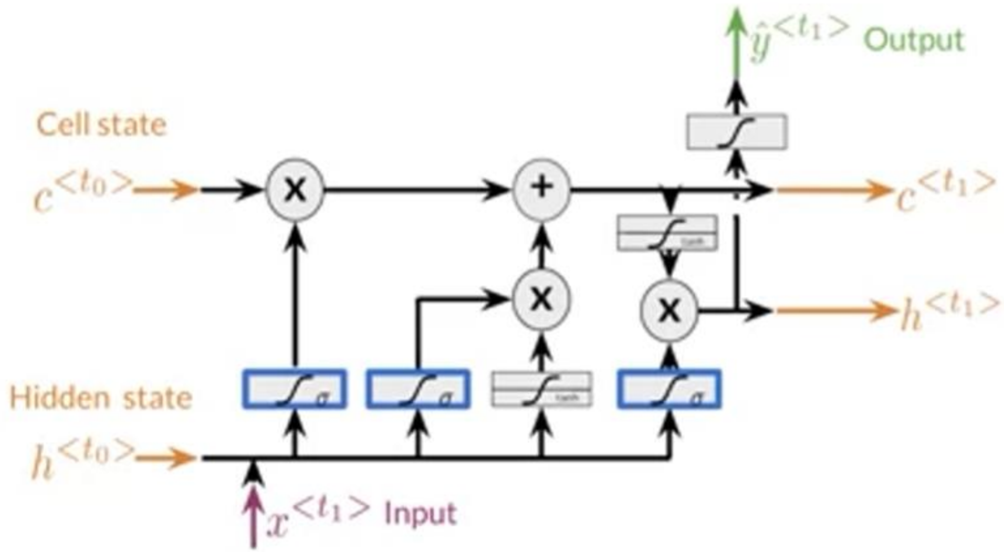
Discard anything irrelevant

Add important new information → Gates

Produce output

Gates in LSTM





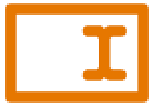
1.Forget Gate: information that is no longer important

2. Input Gate: information to be stored

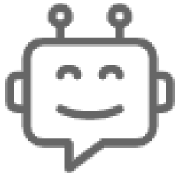
3.Output Gate: information to use at current step

Applications of LSTMs

Next-character prediction



Chatbots



Music composition

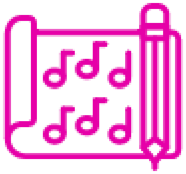
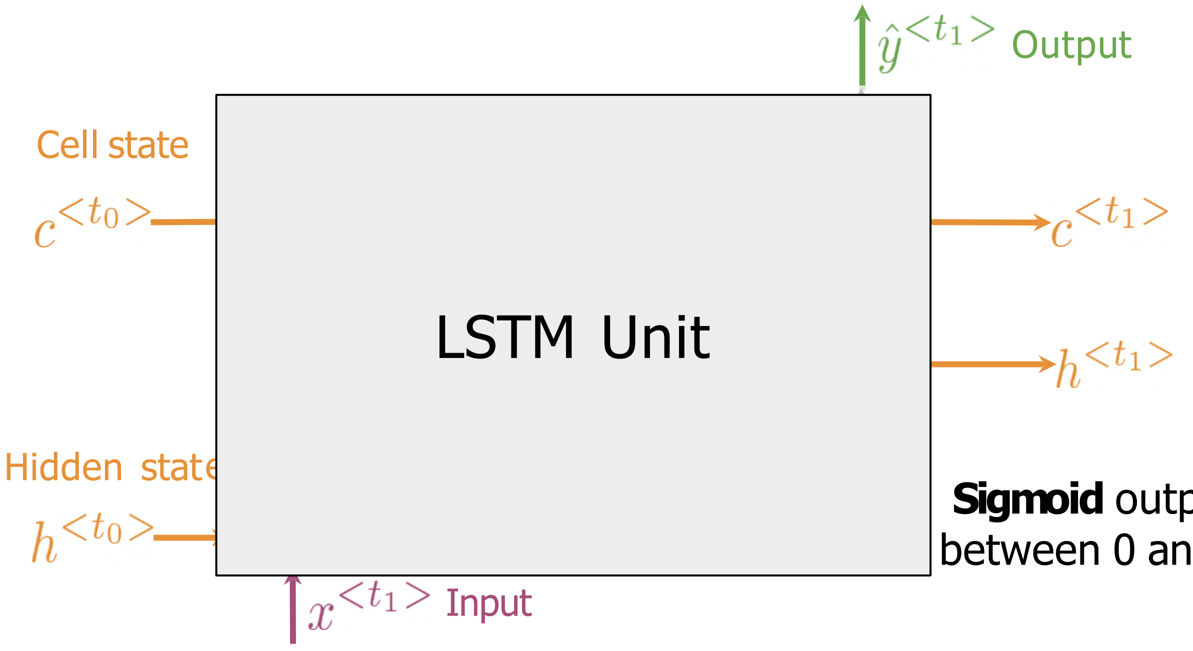


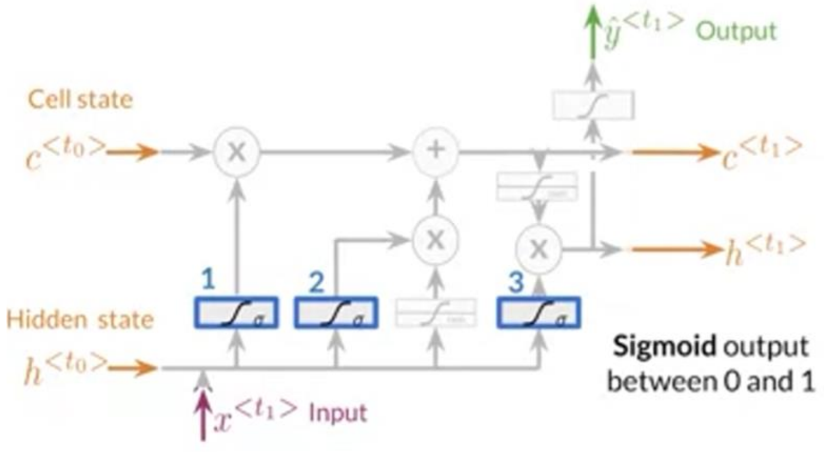
Image captioning

Speech recognition

LSTM Architecture

- Gates in LSTM



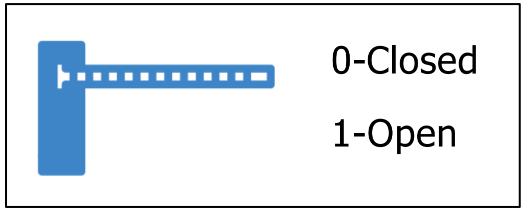


1. Forget Gate: information that is no longer important

2. Input Gate: information to be stored

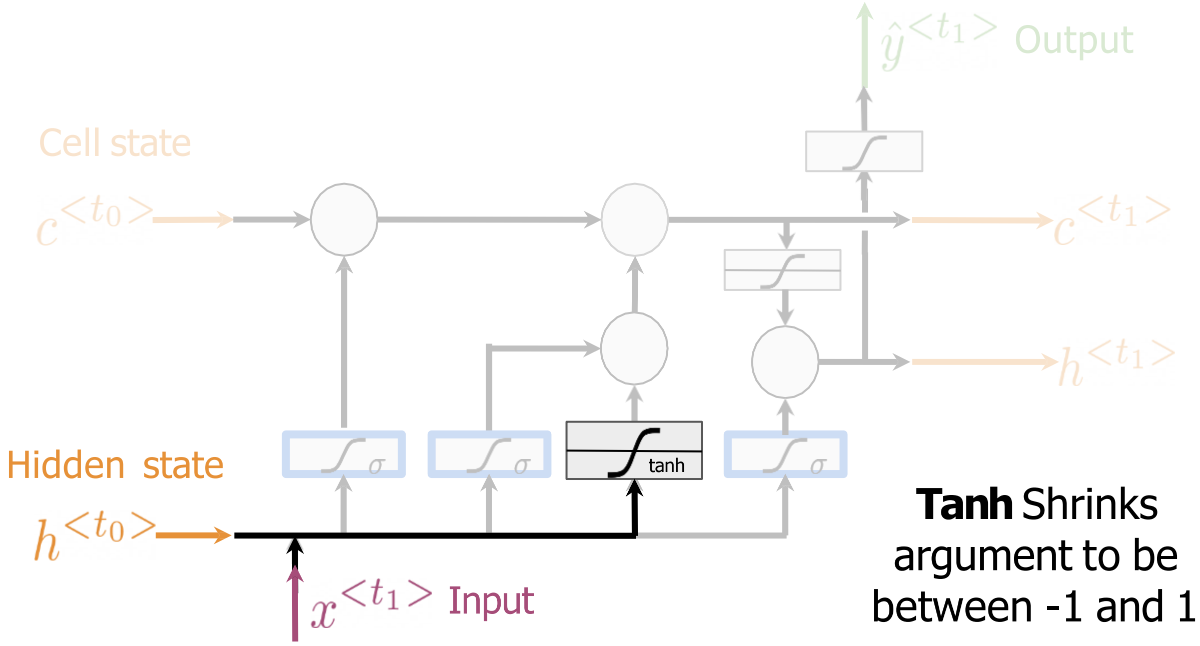
3. Output Gate: information to use at current step

Sigmoid output between 0 and 1



Candidate Cell State

- Transform the information from the previous hidden states and the current inputs
- A hyperbolic tangent activation function shrinks the information to be between -1 and 1

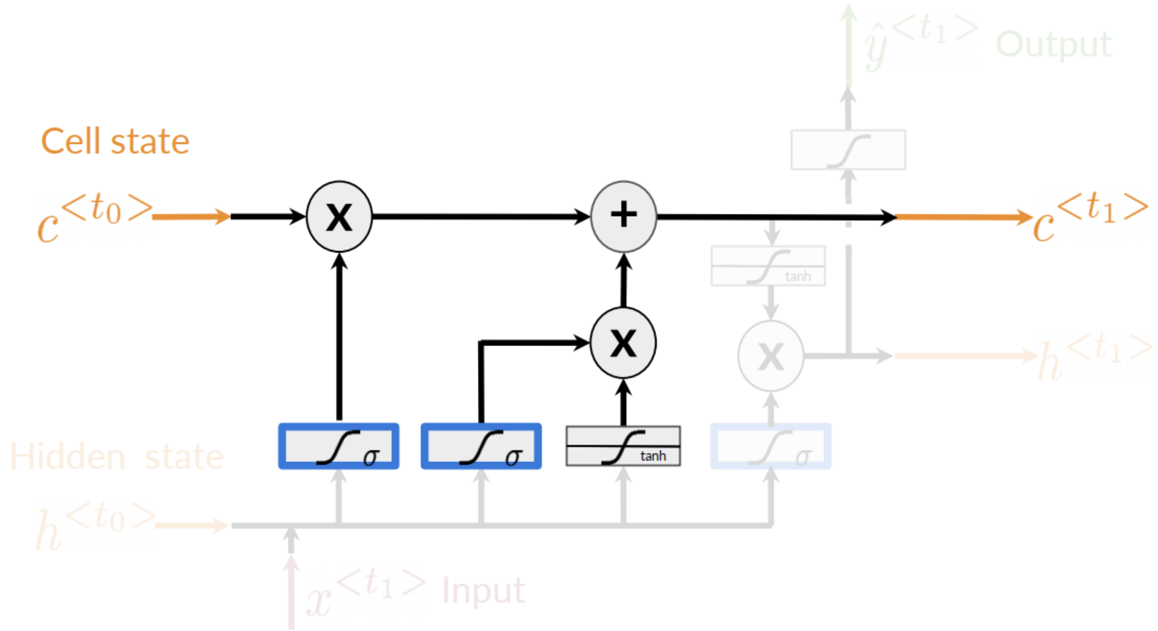


Candidate cell state

Information from the previous **hidden state** and current **input**

Tanh Shrinks argument to be between -1 and 1 → **Other activations could be used**

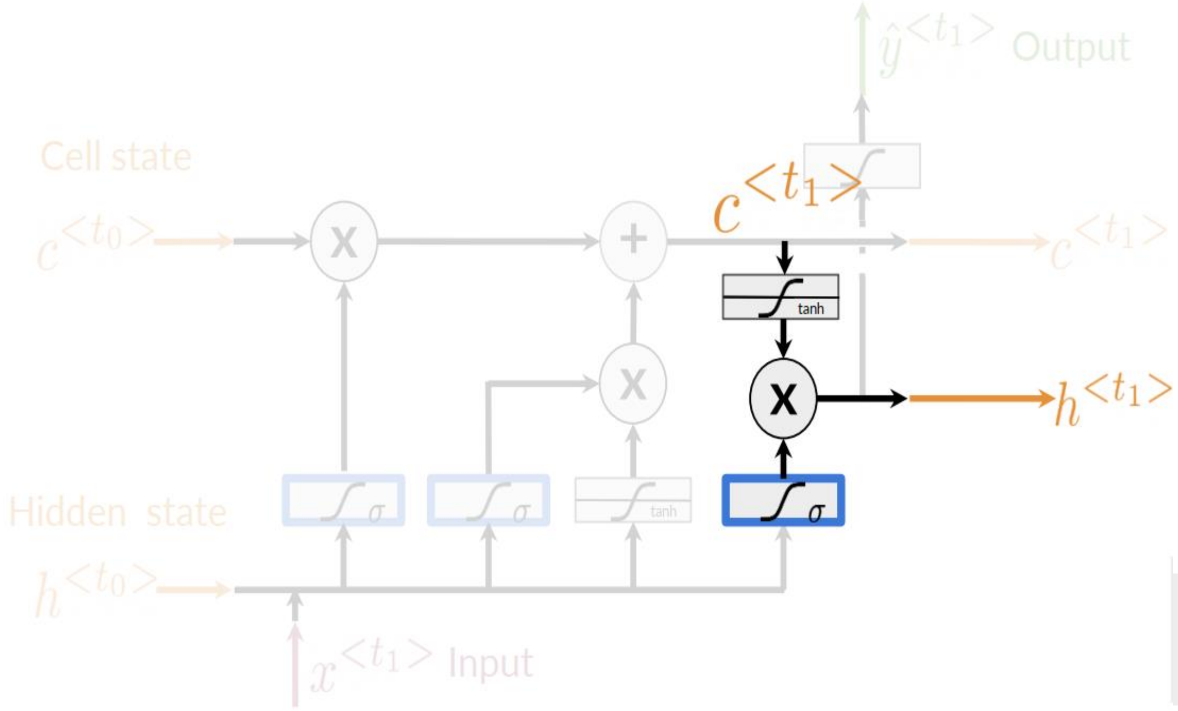
New Cell State



New Cell state

Add information from the **candidate cell state** using the **forget** and **input gates**

New Hidden State



New Hidden State

Select information from the **new cell state** using the **output gate**

The **Tanh** activation could be omitted

- 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

Introduction to Named Entity Recognition

- 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

More Types of Entities

December:
Time Indicator

Egyptian statue:
Artifact

Barack Obama:
Person

Example of a labeled sentence

- Applications of NER systems
  - Search engine efficiency
  - Recommendation engines
  - Customer service
  - Automatic trading

Training NERs: Data Processing

FPT Education
FPT UNIVERSITY

  - Convert words and entity classes into arrays
  - Token padding
  - Create a data generator
- Processing data for NERs
  - Assign each class a number
  - Assign each word a number

Sharon flew to Miami last Friday.

[42821, 853, 187, 53882, 2894, 73]

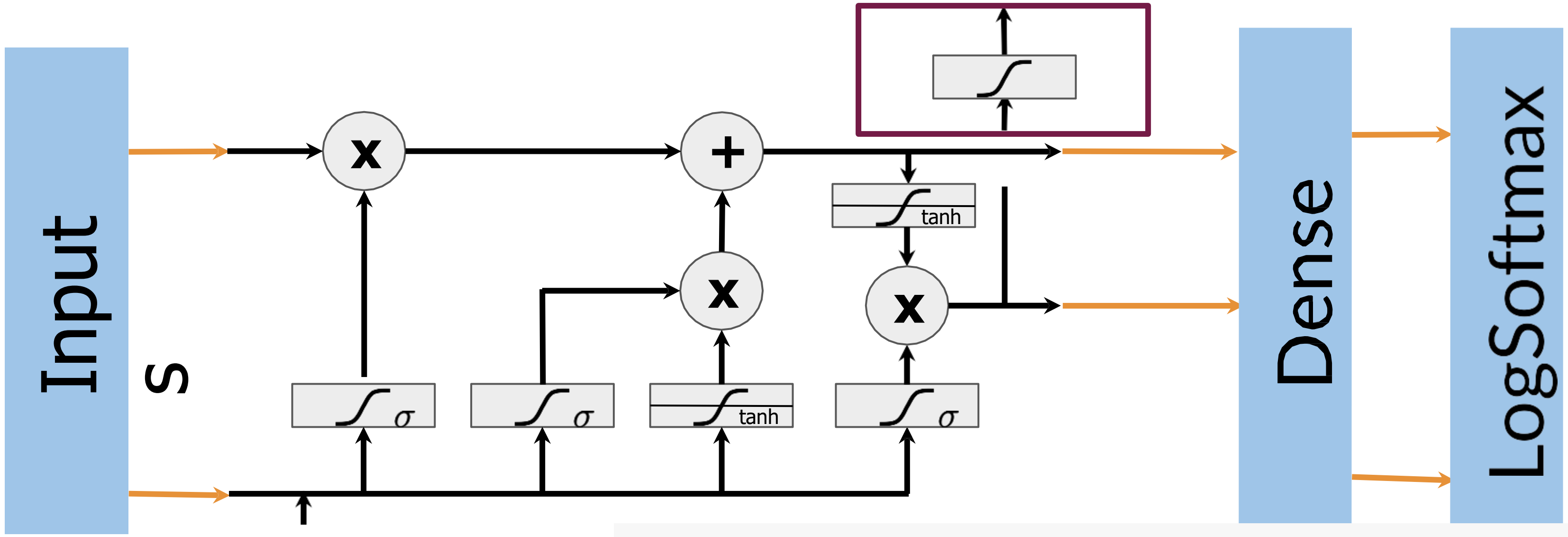
B-per O O B-geo O B-tim

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.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

Computing Accuracy

FPT Education
FPT UNIVERSITY

- 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
```

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