Deep Learning with Keras and TensorFlow

Recurrent Neural Networks (RNN)

Learning Objectives

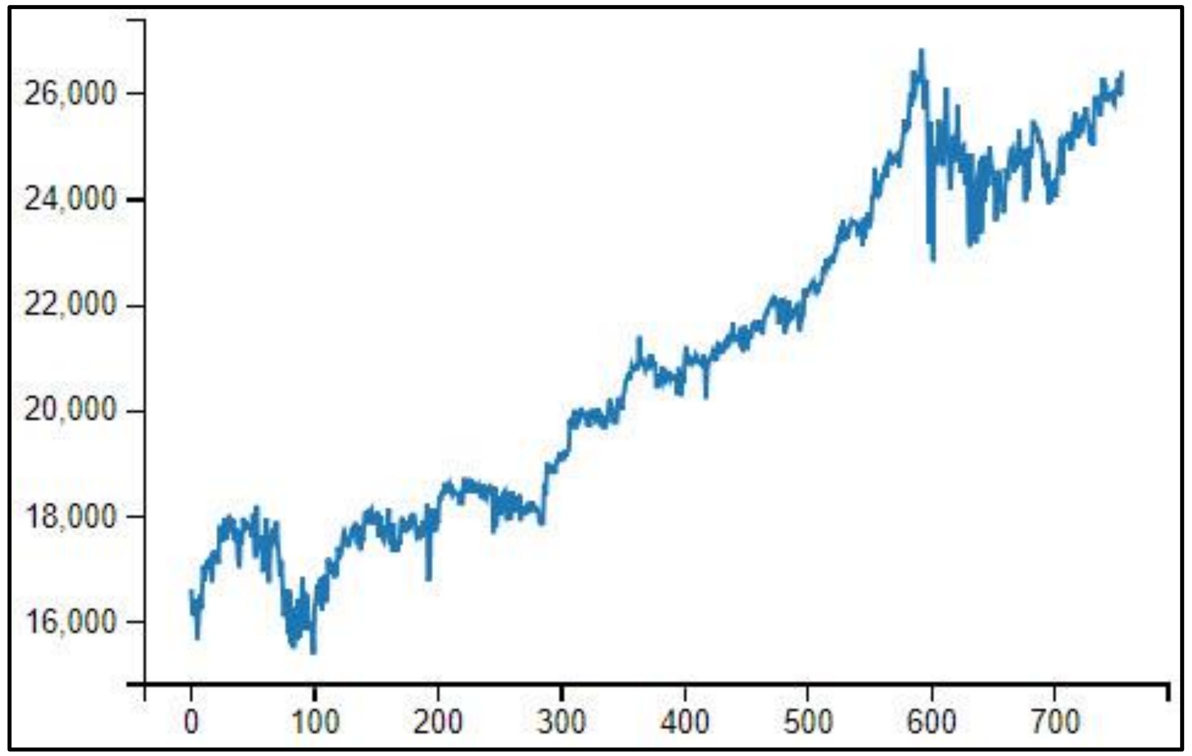
By the end of this lesson, you will be able to:

- Implement RNNs for sequential data
- Use LSTMs for memory operations within RNNs
- Perform gated operations in LSTMs using GRUs
- Improve the performance of LSTMs using the Attention mechanism

Sequence Data

What Is Sequential Data?

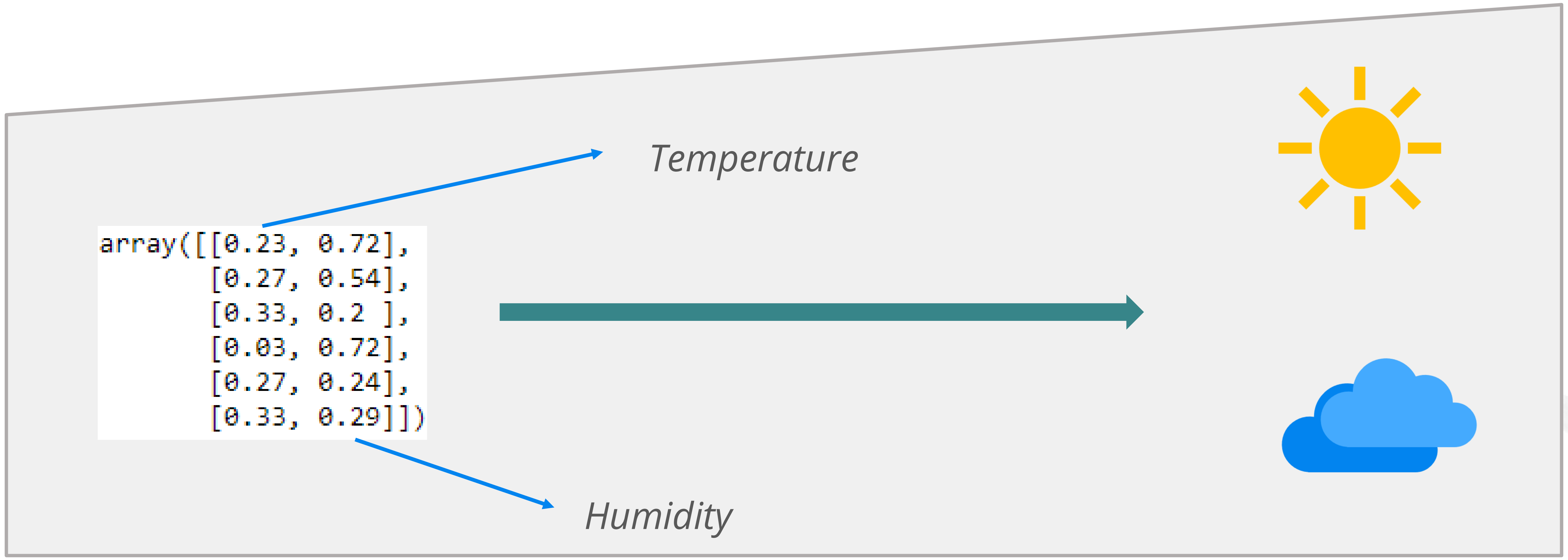
The dataset is said to be sequential when the data points are dependent on other data points within a dataset.



Example: Time Series Data

Sequential Data: Problems

Consider you have a sequential data that contains temperature and humidity values for everyday.



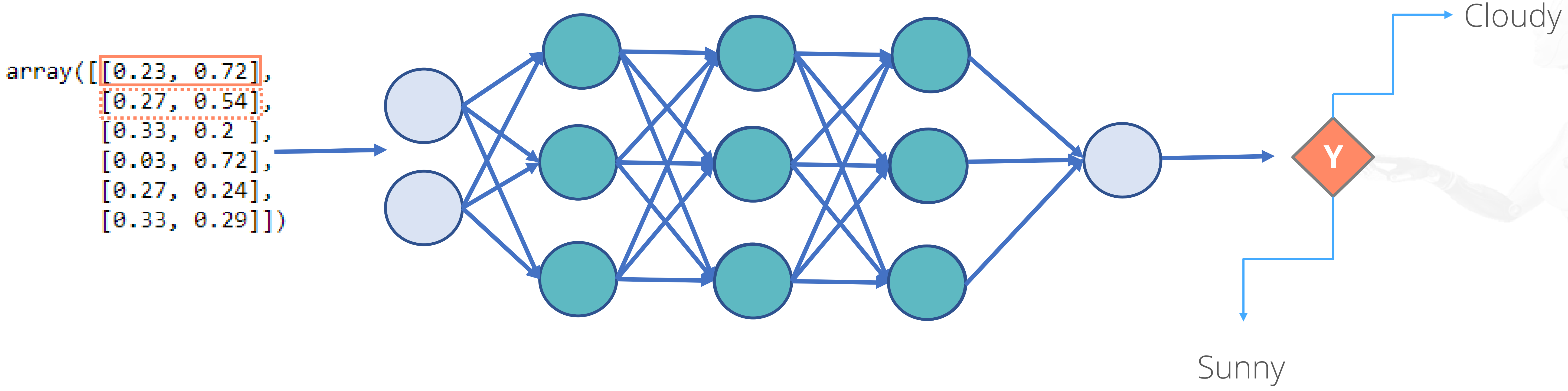
```
array([[0.23, 0.72],
       [0.27, 0.54],
       [0.33, 0.2 ],
       [0.03, 0.72],
       [0.27, 0.24],
       [0.33, 0.29]])
```

Goal: To build a neural network that imports the temperature and humidity values of a given day and predicts if the weather for that day is sunny or rainy.

Sequential Data: Problems

The data then flows to the hidden layers, where the weights and biases are applied.

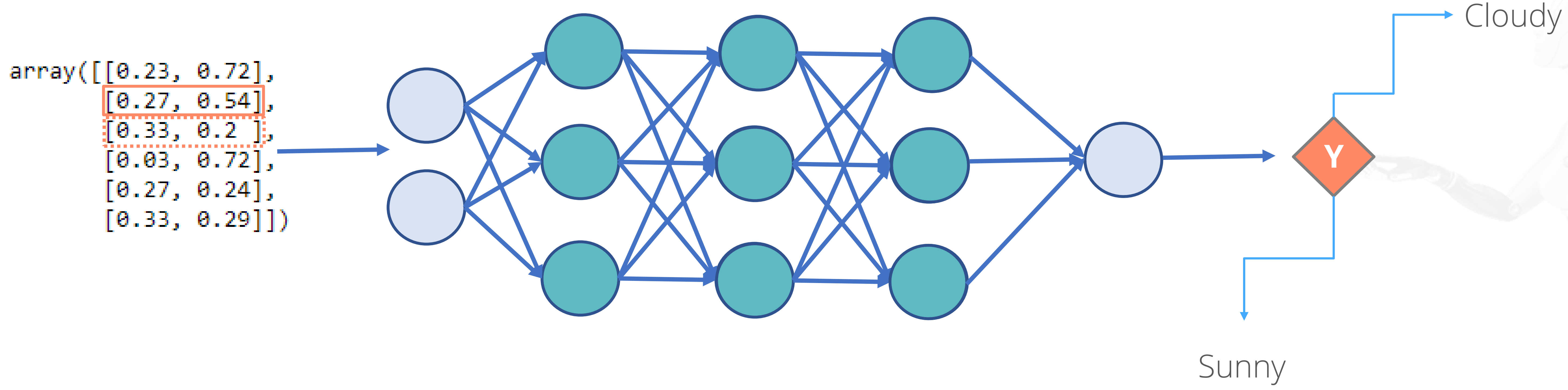
Weights ([0.23 + 0.72]) + Bias



```
array([[0.23, 0.72],
       [0.27, 0.54],
       [0.33, 0.2 ],
       [0.03, 0.72],
       [0.27, 0.24],
       [0.33, 0.29]])
```

Sequential Data: Problems

A traditional neural network assumes that the data is non-sequential and each data point is independent of the others.

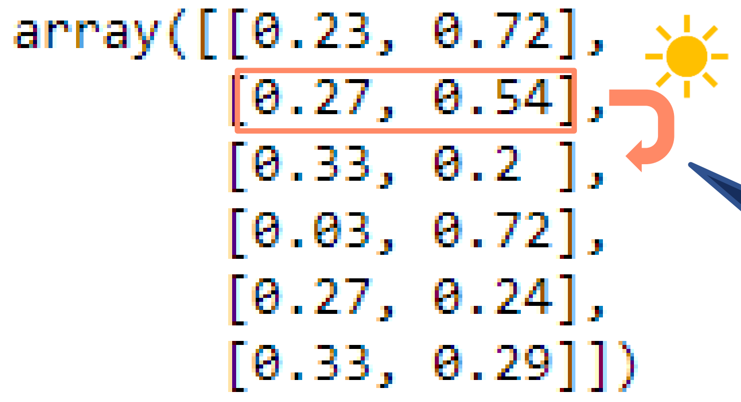


```
array([[0.23, 0.72],
       [0.27, 0.54],
       [0.33, 0.2 ],
       [0.03, 0.72],
       [0.27, 0.24],
       [0.33, 0.29]])
```

Note: The network does not remember what it gives as an output. It just accepts the next data point.

Sequential Data: Problems

In the weather data, there is a strong correlation between the weather from one day and the weather in subsequent days. The former has influence over the latter.

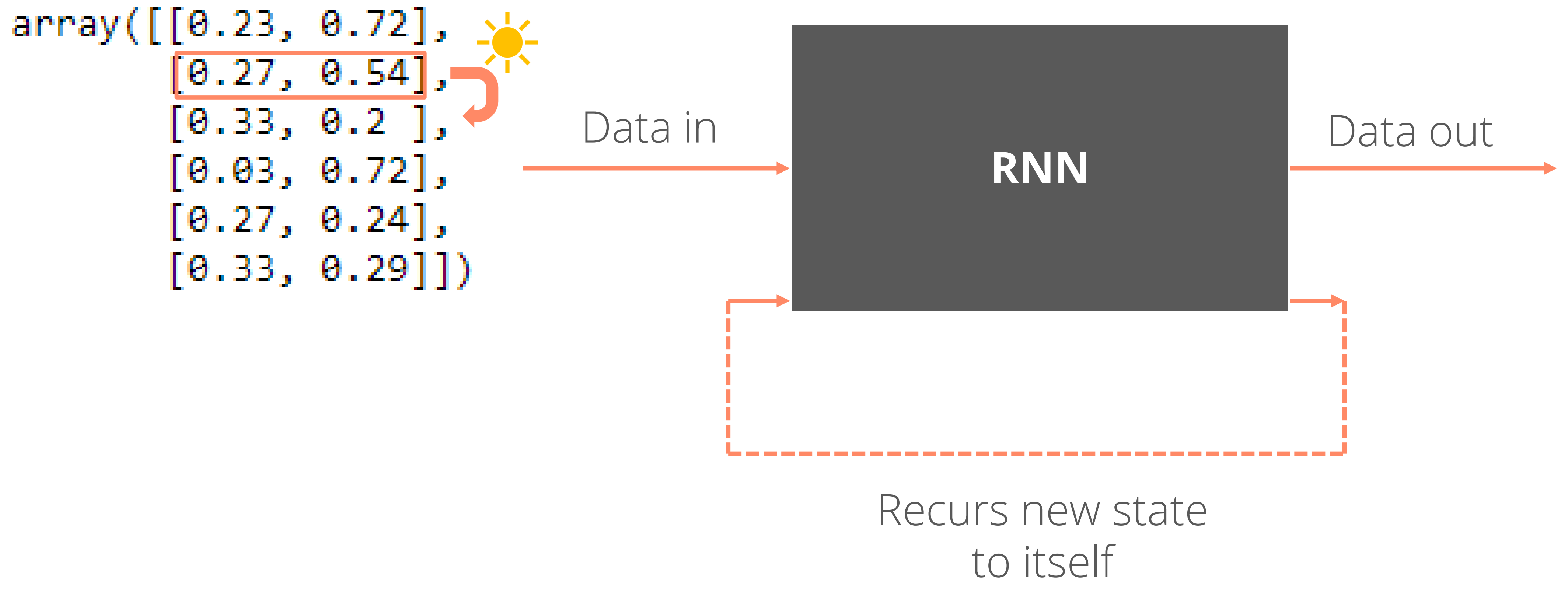


```
array([[0.23, 0.72],
       [0.27, 0.54],
       [0.33, 0.2 ],
       [0.03, 0.72],
       [0.27, 0.24],
       [0.33, 0.29]])
```

If it was sunny on one day in the middle of summer, it's easy to to presume that it'll also be sunny on the following day.

Sequential Data: Solution

An RNN has a mechanism that can handle a sequential dataset.

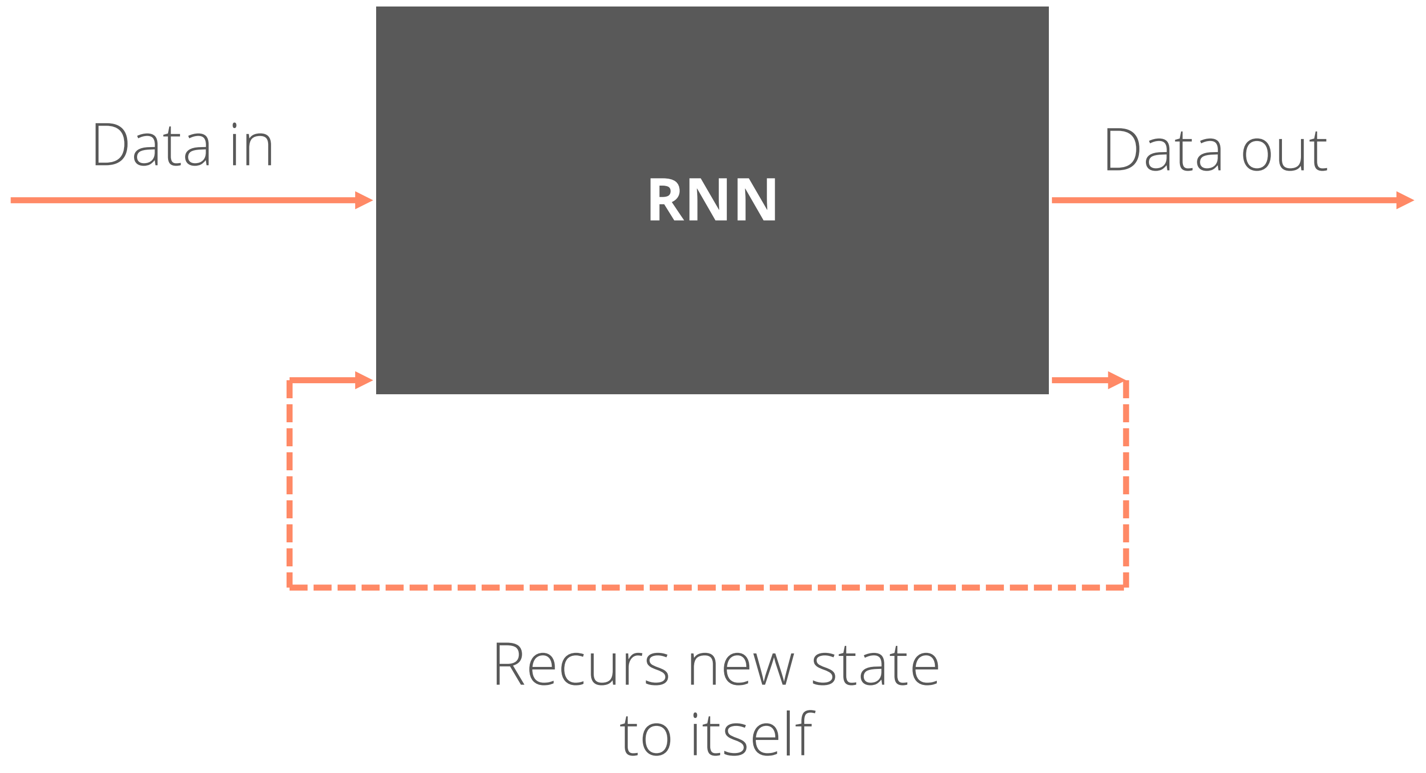


```
array([[0.23, 0.72],
       [0.27, 0.54],
       [0.33, 0.2 ],
       [0.03, 0.72],
       [0.27, 0.24],
       [0.33, 0.29]])
```

RNN Model

The RNN Model

The RNN remembers the analysis done upto a given point by maintaining a **state**.

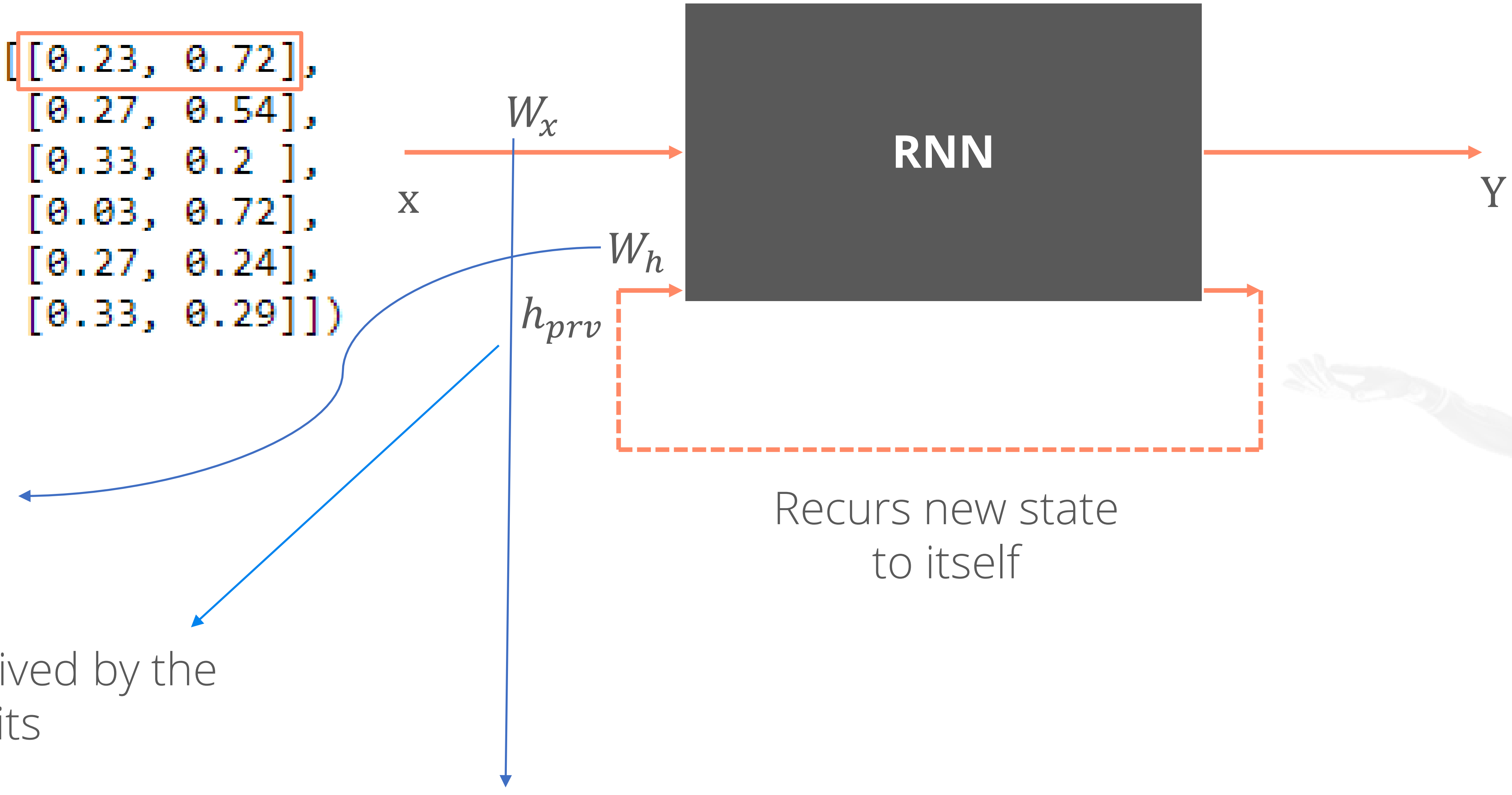


Note: You can think of the **state** as the **memory** of RNN which recurs into the net with each new input.

RNN: Working

The first data point flows into the network as input data, denoted as x.

```
array([[0.23, 0.72],
       [0.27, 0.54],
       [0.33, 0.2 ],
       [0.03, 0.72],
       [0.27, 0.24],
       [0.33, 0.29]])
```



Weights are multiplied by the previously hidden state

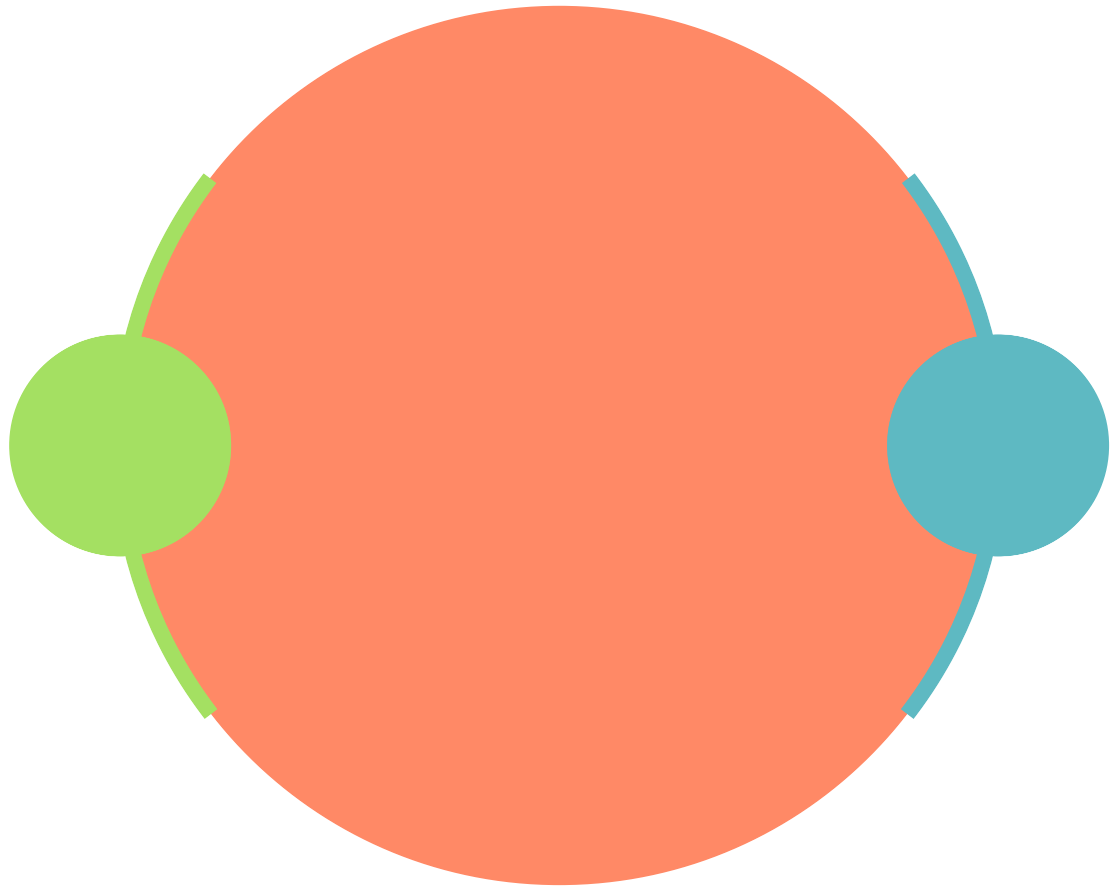
Previous state received by the hidden units

Recurs new state to itself

Weight matrix between input and hidden units

RNN: Working

Two values are calculated in the hidden layer as shown below:

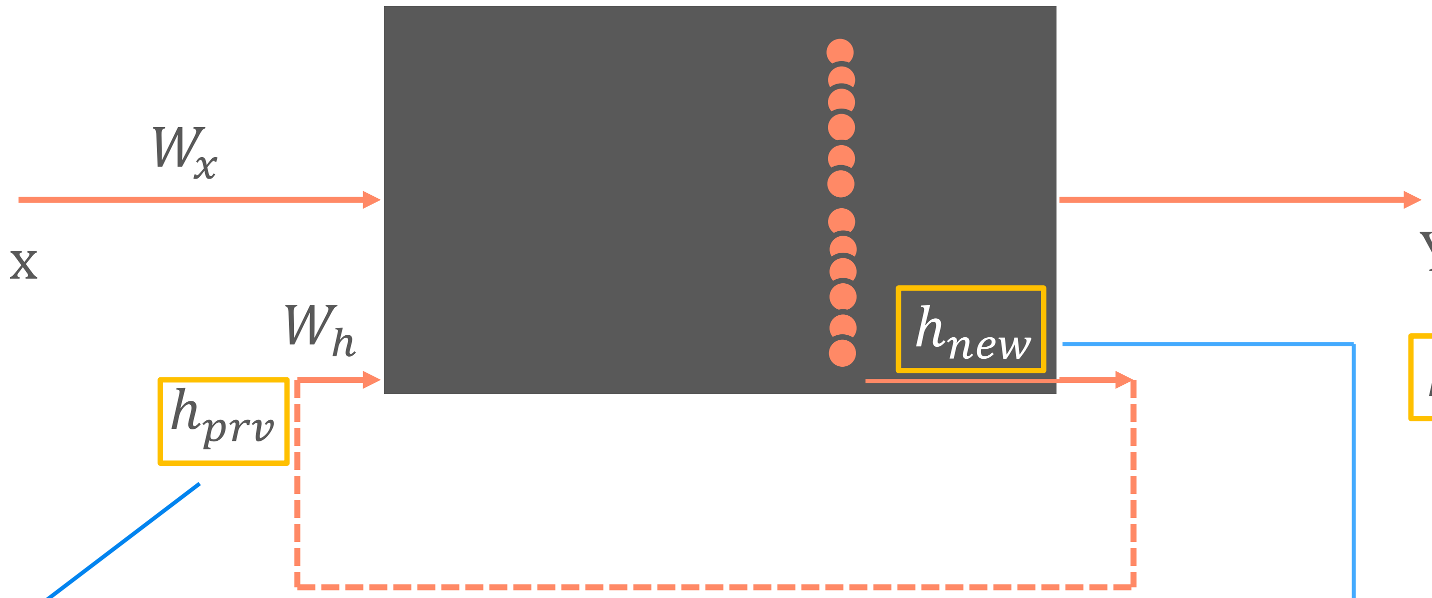


The new or updated state, denoted as h_new, is used for the next data point

The output of the network is denoted as y

RNN: Working

```
array([[0.23, 0.72],
       [0.27, 0.54],
       [0.33, 0.2 ],
       [0.03, 0.72],
       [0.27, 0.24],
       [0.33, 0.29]])
```



$$h_{new} = \tanh(W_h \cdot h_{prv} + W_x \cdot X)$$

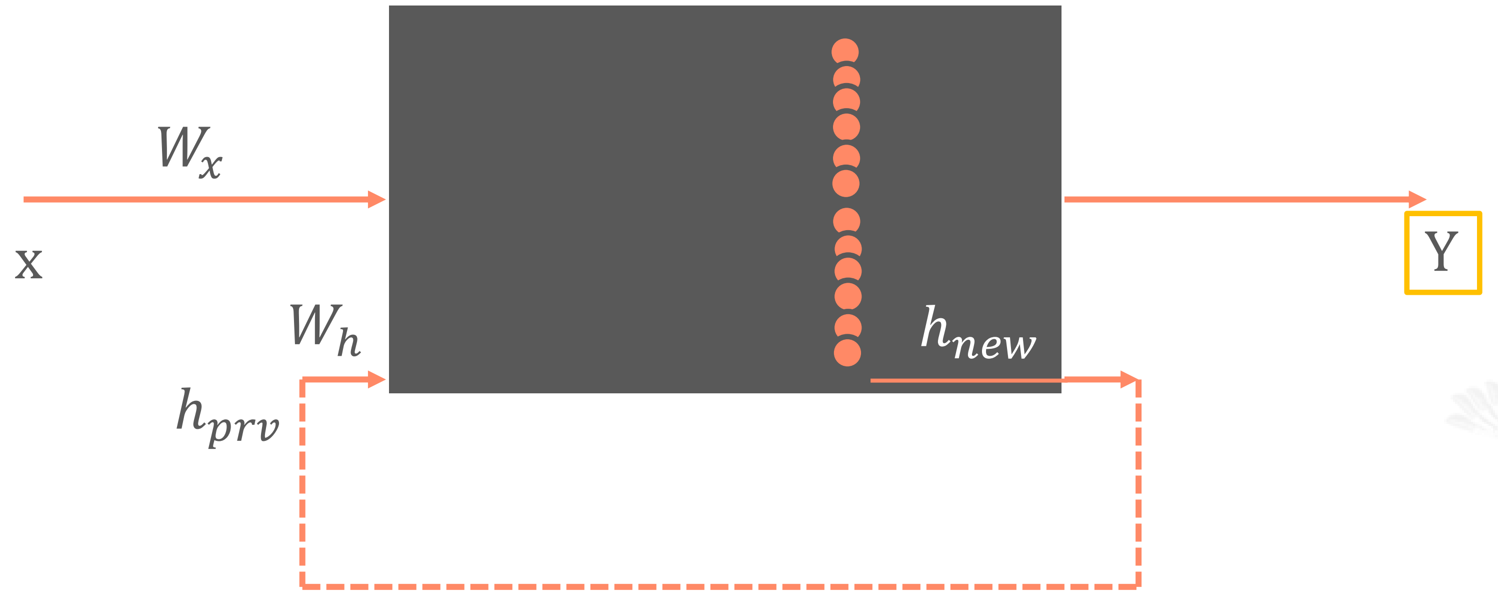
Previous state received by the hidden units

The new state is a function of the previous state and the input data

RNN: Working

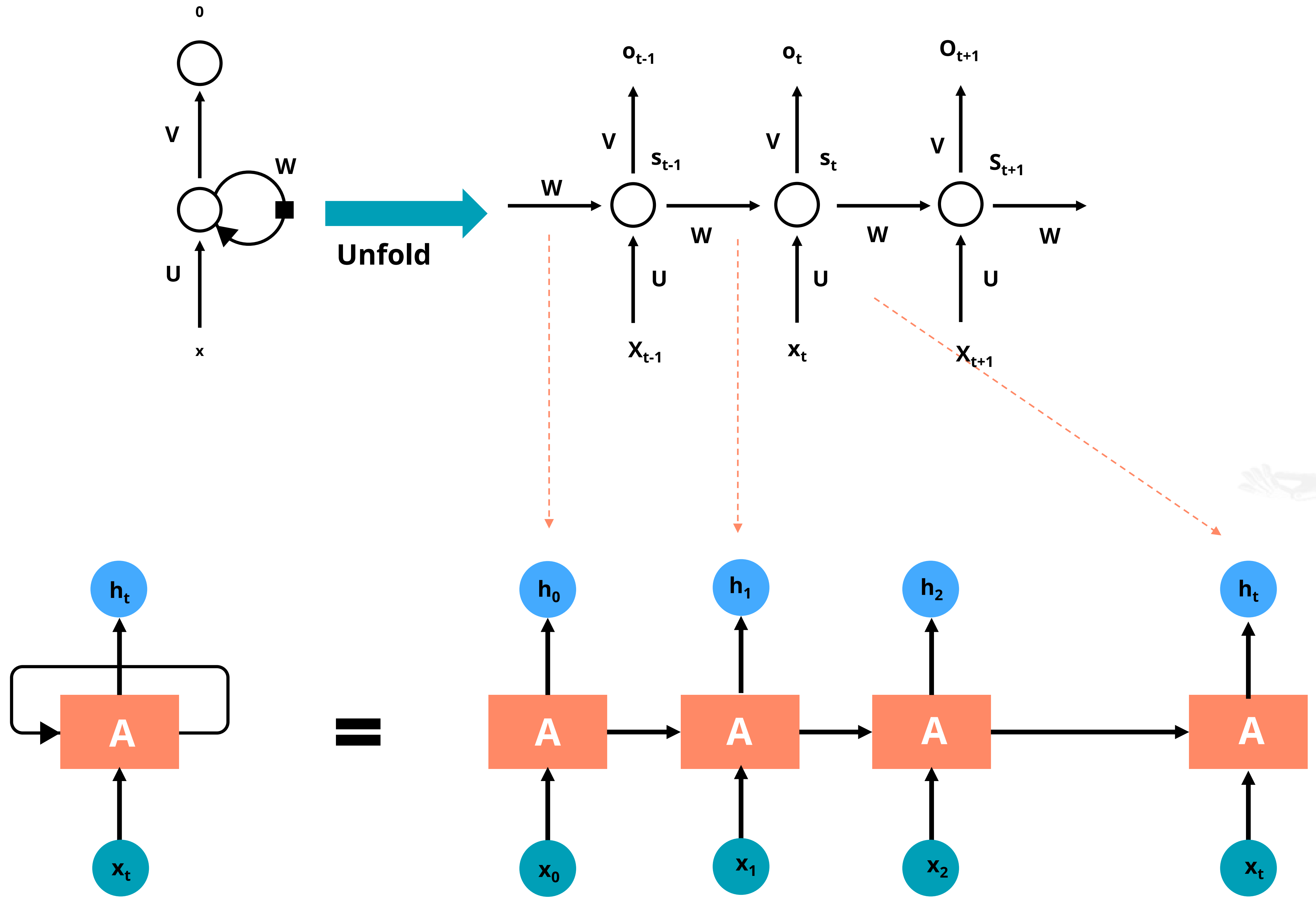
The output of the hidden unit is simply calculated by multiplication of the new hidden state and the output weight matrix.

```
array([[0.23, 0.72],
       [0.27, 0.54],
       [0.33, 0.2 ],
       [0.03, 0.72],
       [0.27, 0.24],
       [0.33, 0.29]])
```



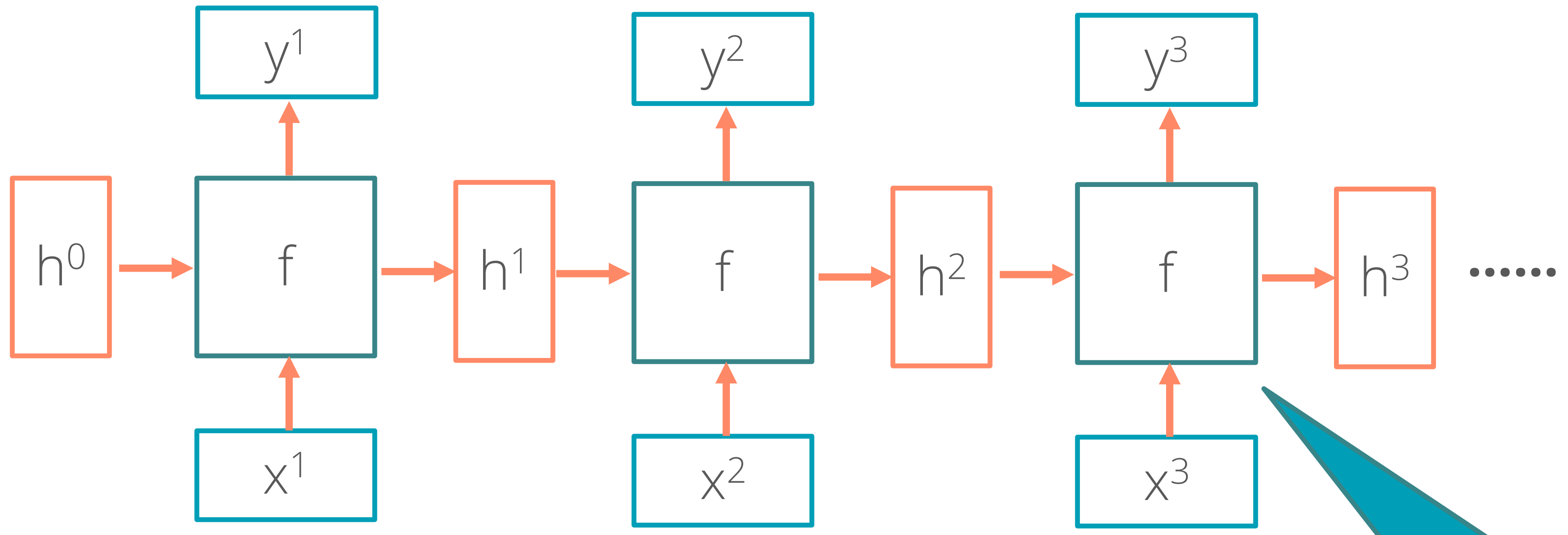
After processing the first data point, a new context is generated that represents the most recent point. Then, this context is fed back into the net with the next data point and we repeat these steps until all the data is processed.

A Typical RNN



Reduces Complexity

Given function **f: h', y = f (h,x): h** and **h'** are vectors with the same dimension.


$$\begin{tikzcd} & y^1 & & y^2 & & y^3 & & \\ h^0 \arrow[r] & f \arrow[u] \arrow[r] & h^1 \arrow[r] & f \arrow[u] \arrow[r] & h^2 \arrow[r] & f \arrow[u] \arrow[r] & h^3 & \cdots\cdots \\ & x^1 \arrow[u] & & x^2 \arrow[u] & & x^3 \arrow[u] & & \end{tikzcd}$$

We only need one function f, irrespective of the input and output sequences.

Applications of RNN

Speech Recognition

The goal is to consume a sequence of data and then produce another sequence.

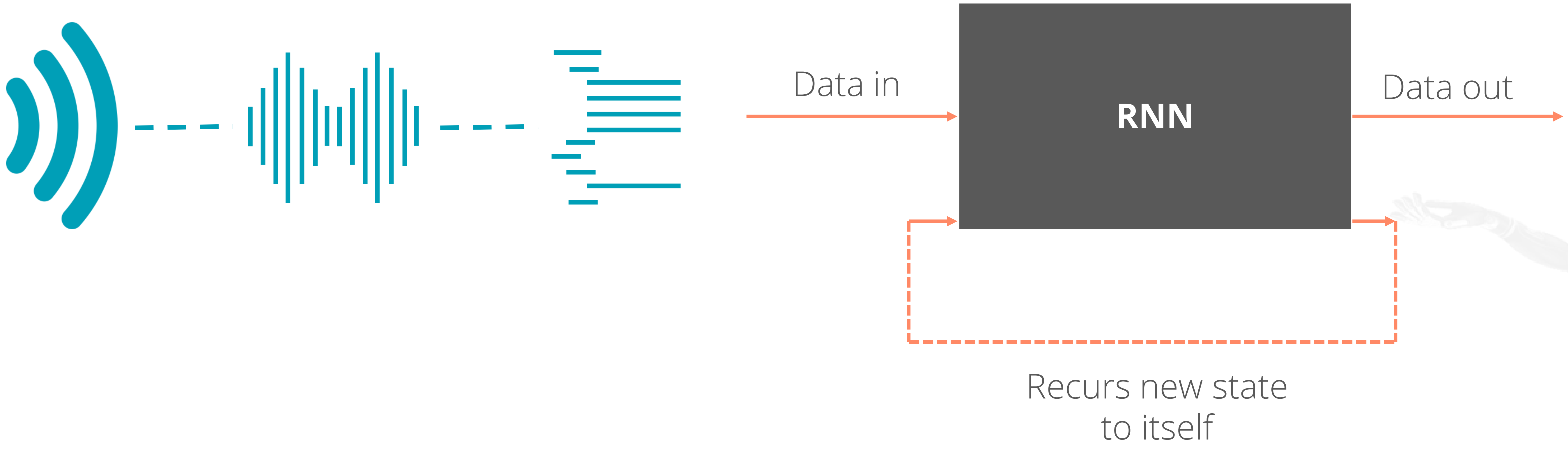
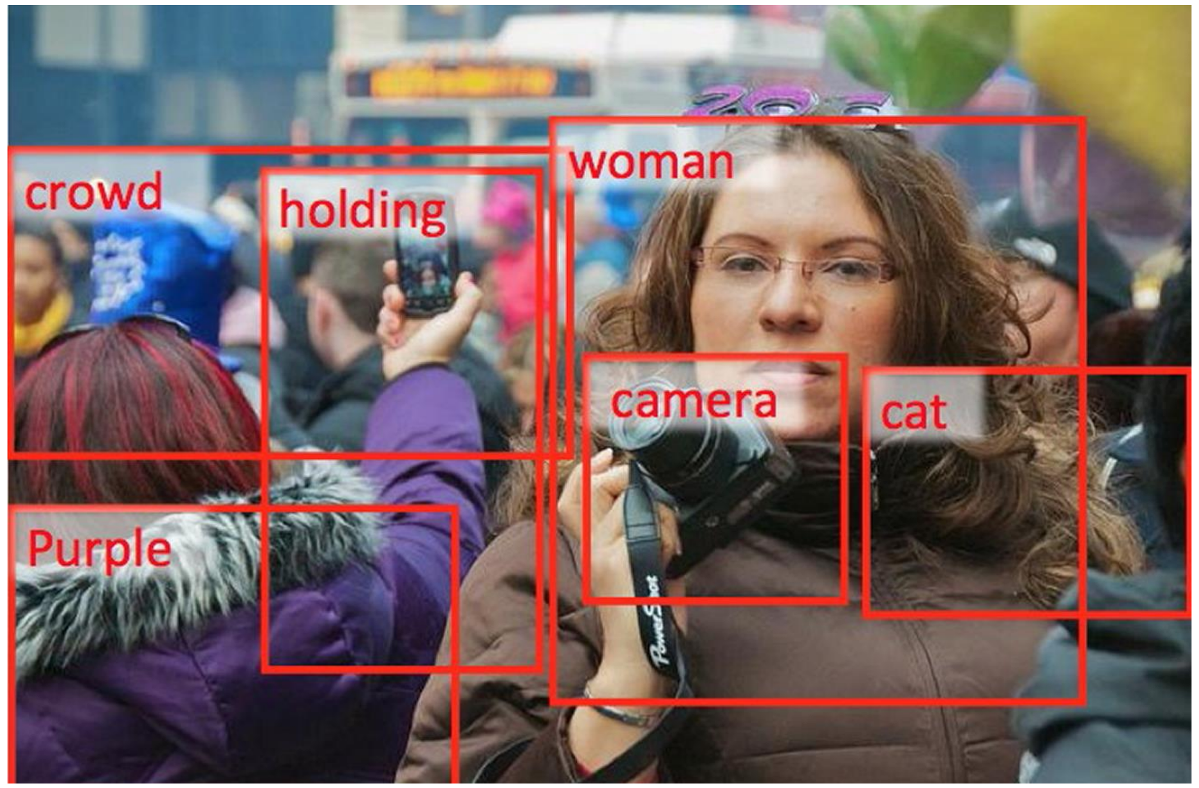
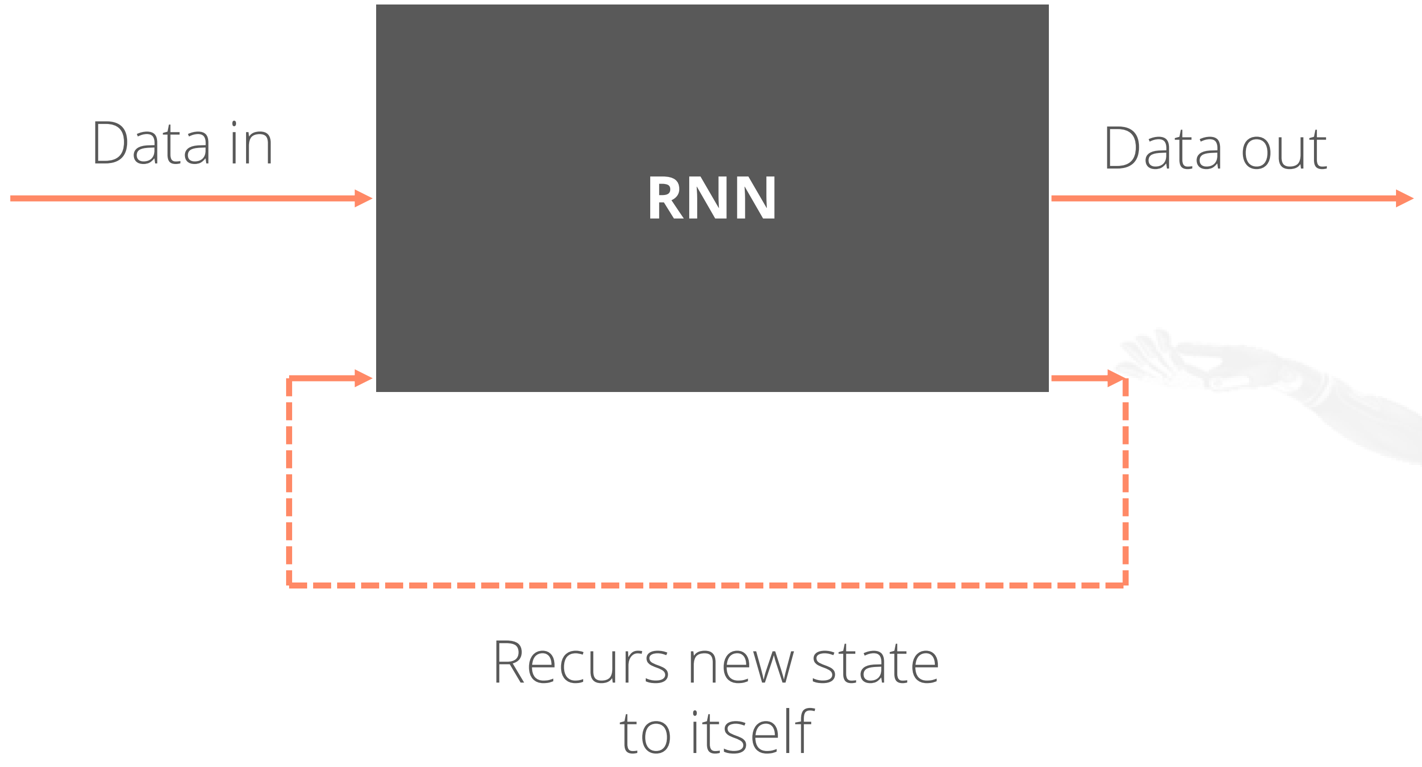


Image Captioning

You can create a model that's capable of understanding the elements in an image.

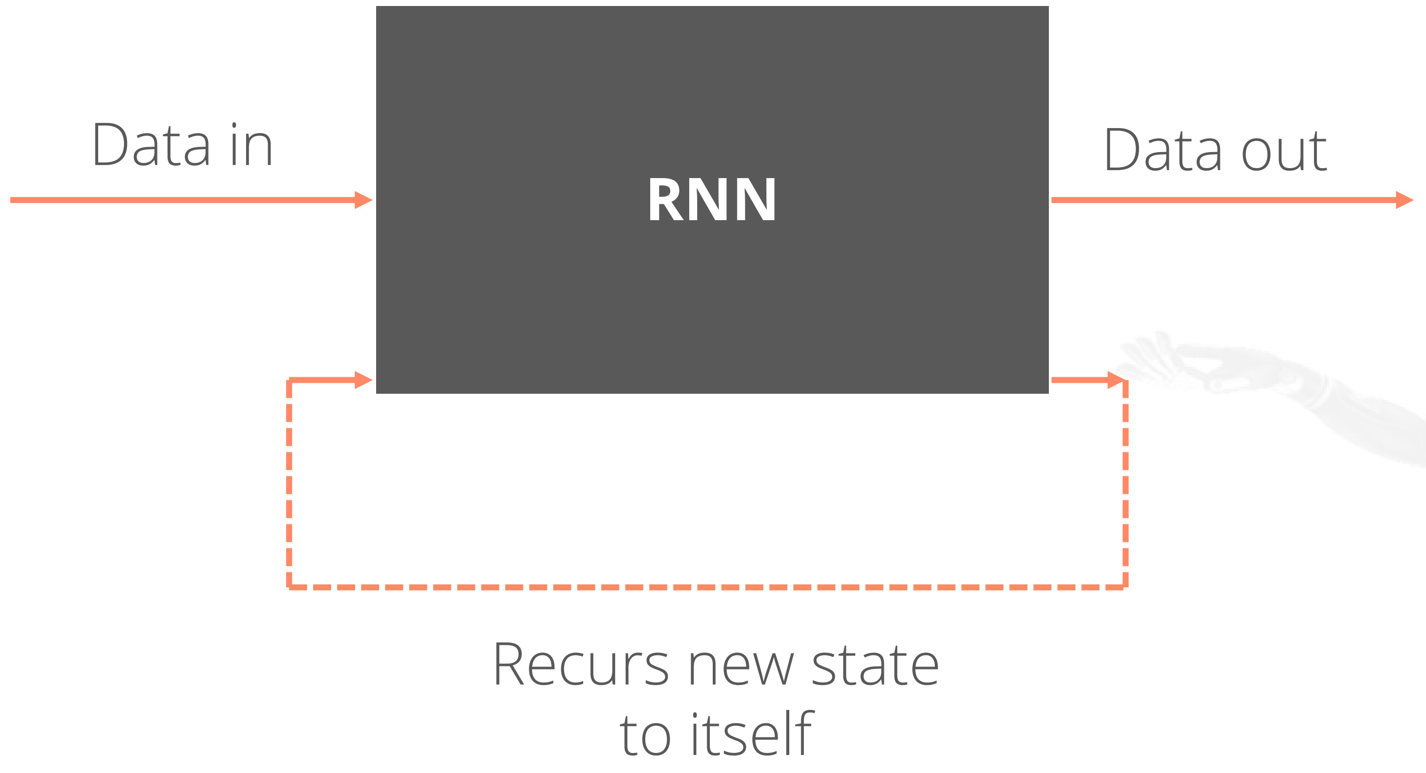




Note: There is just one input (the image) and the output is a sequence of words. Therefore, it is also known as **one-to-many**.

Sentiment Analysis

RNNs can be used for sentiment analysis, where it focuses only on the final output and not on the sentiment behind each word.

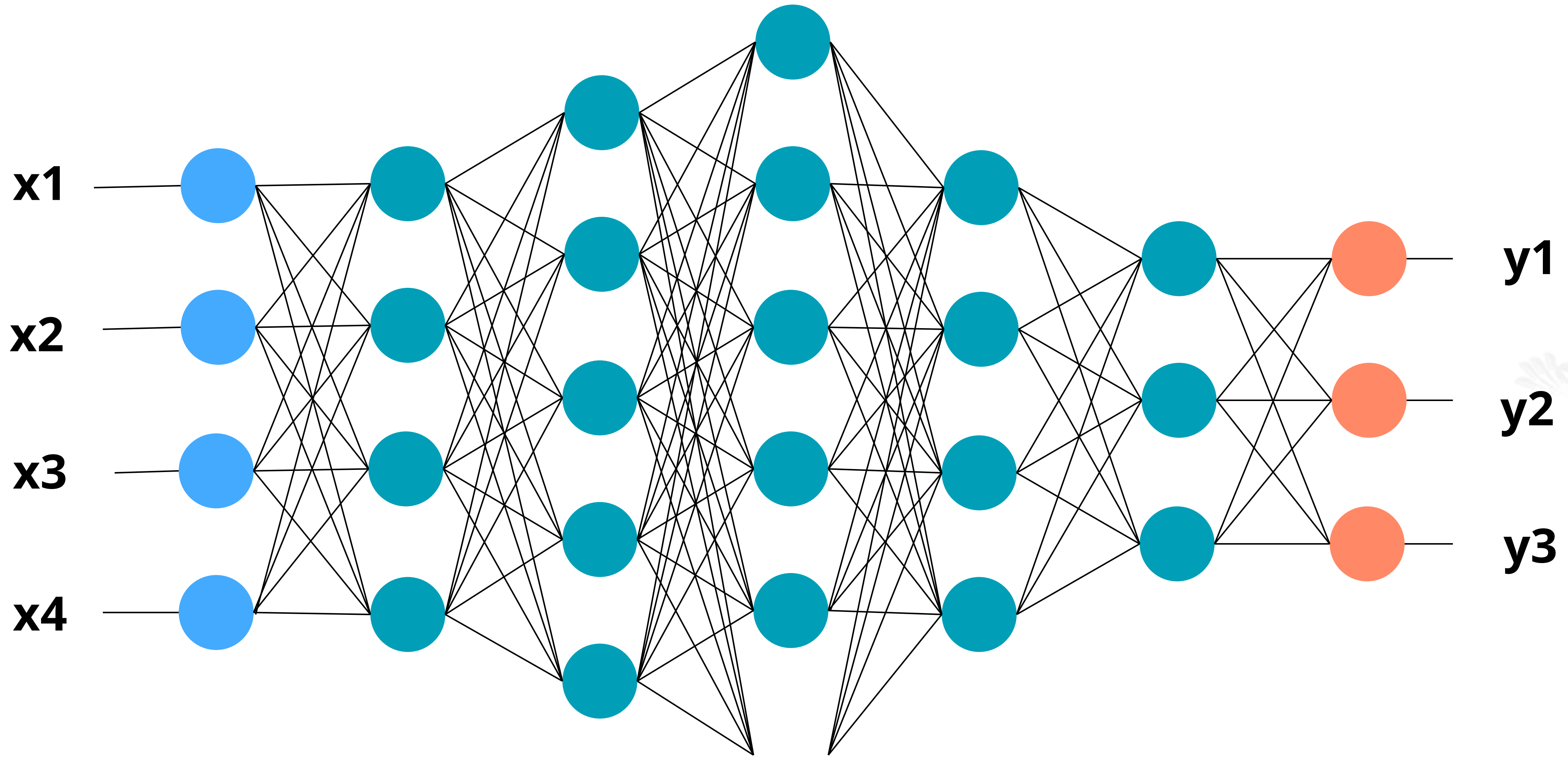


Note: The RNN here consumes a sequence of data and produces just one output. Therefore, it is also known as **many-to-one**.

Deep RNNs

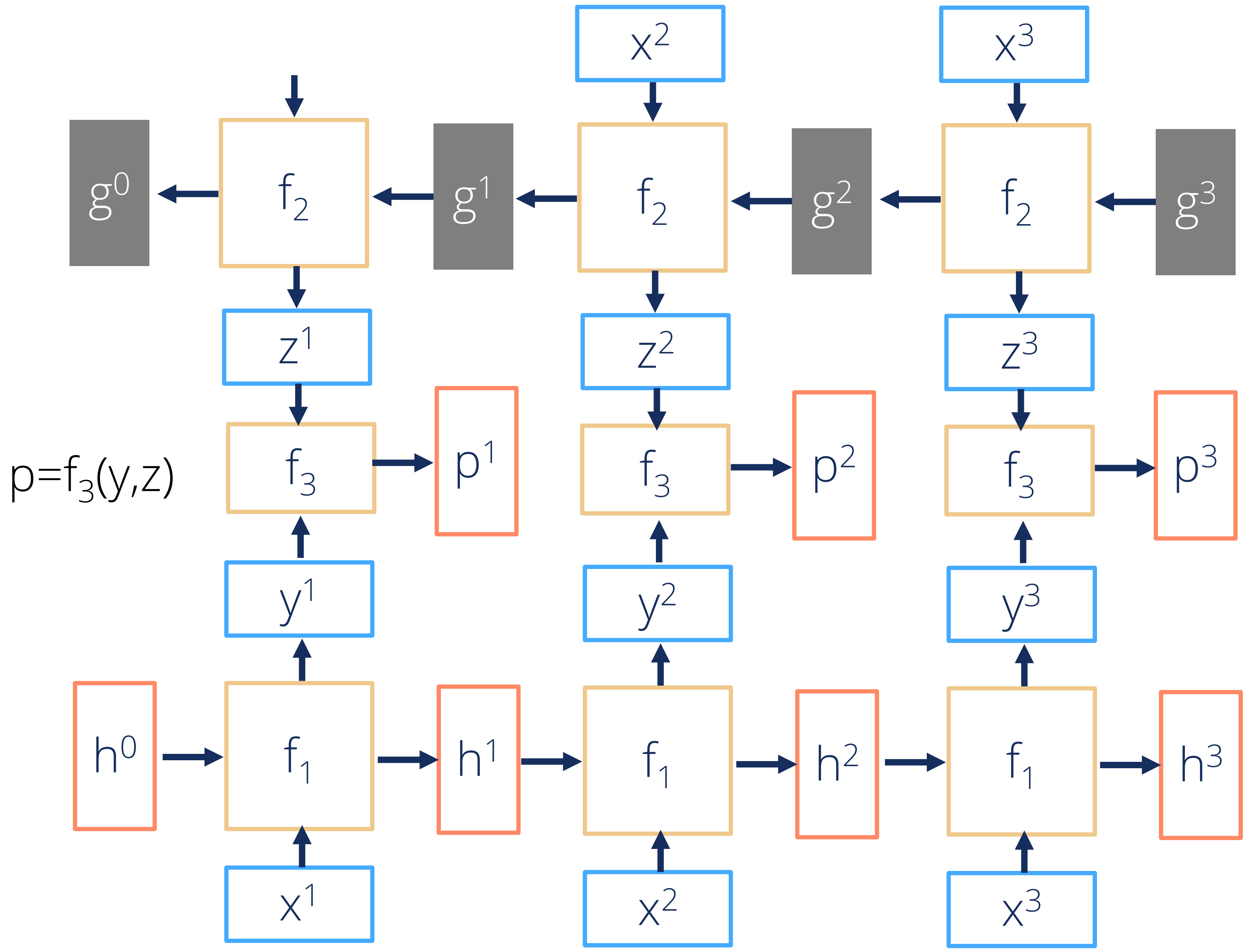
Problems with Smaller RNN Networks

If x_1 .... x_n is very large and continues to grow, the fully connected network will become too big.



Bidirectional RNNs

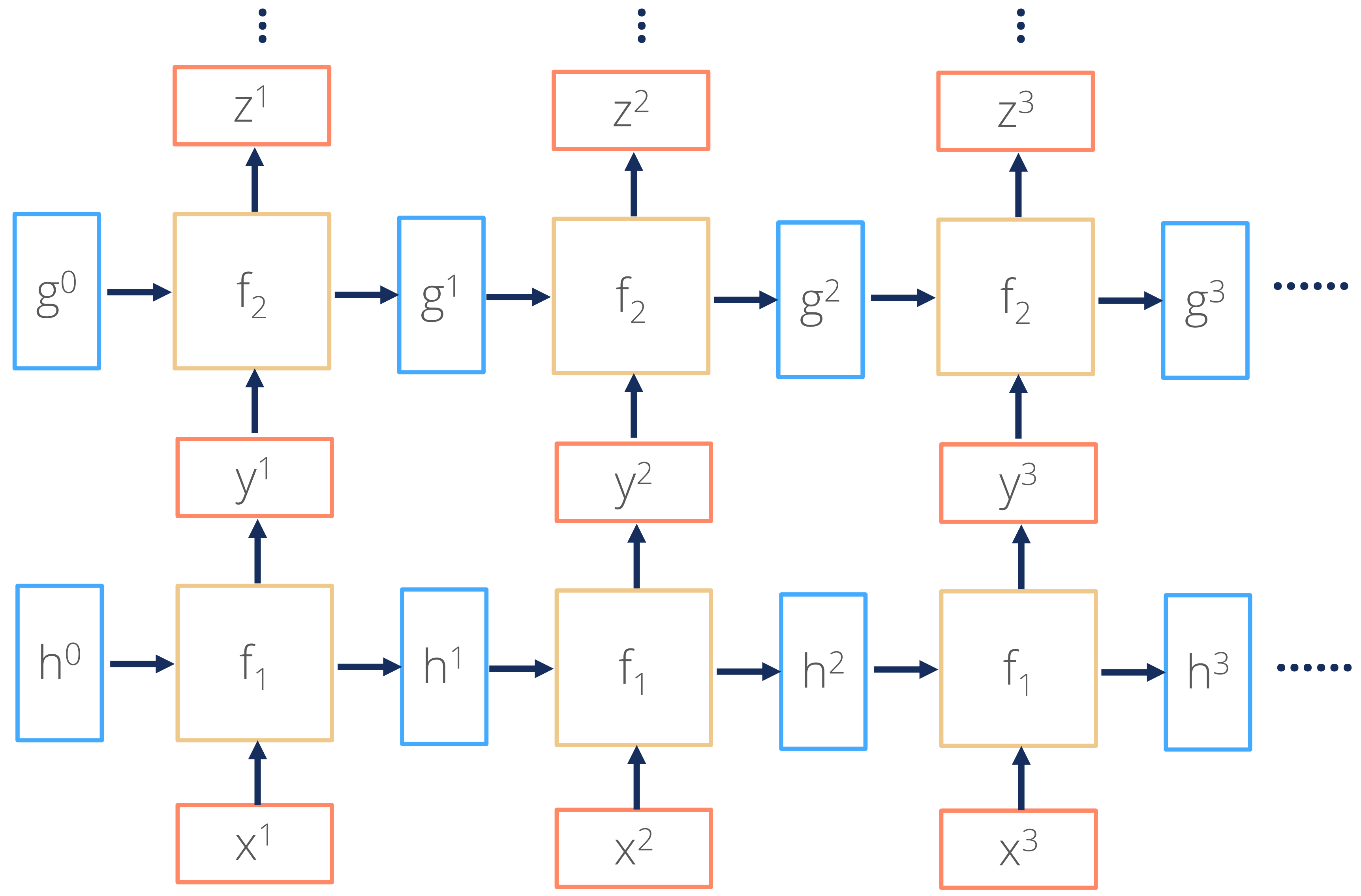
Bidirectional RNNs are constructed by putting two RNNs (f1 and f2) together. Mathematically, these are defined as **y,h=f1(x,h) and z,g = f2(g,x)**.



Deep RNNs

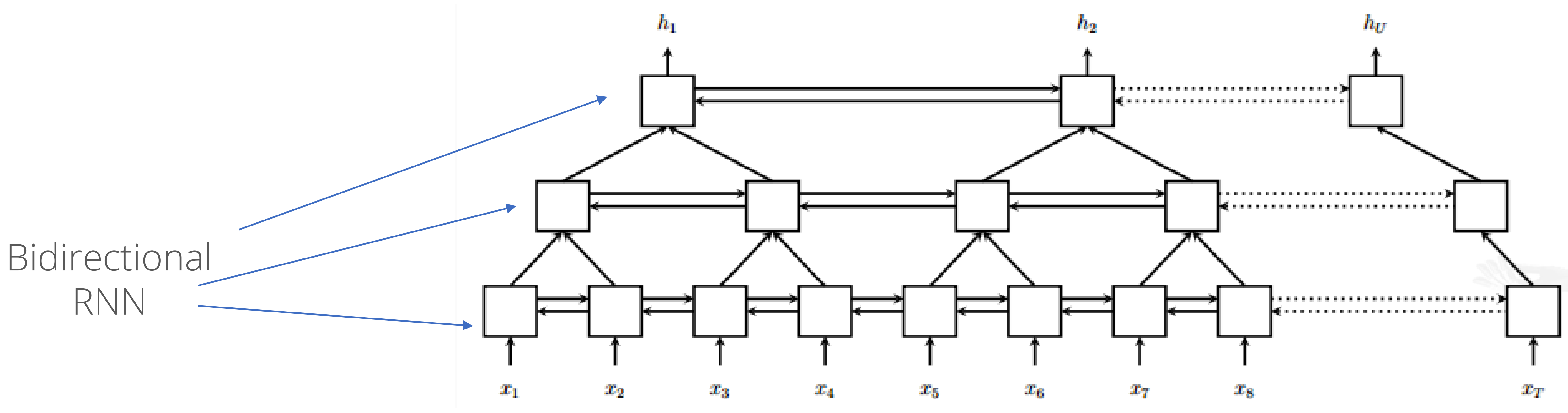
Deep RNNs are constructed by adding more layers to simple RNNs. Mathematically, it can be defined as

h',y = f1(h,x), g',z = f2(g,y)



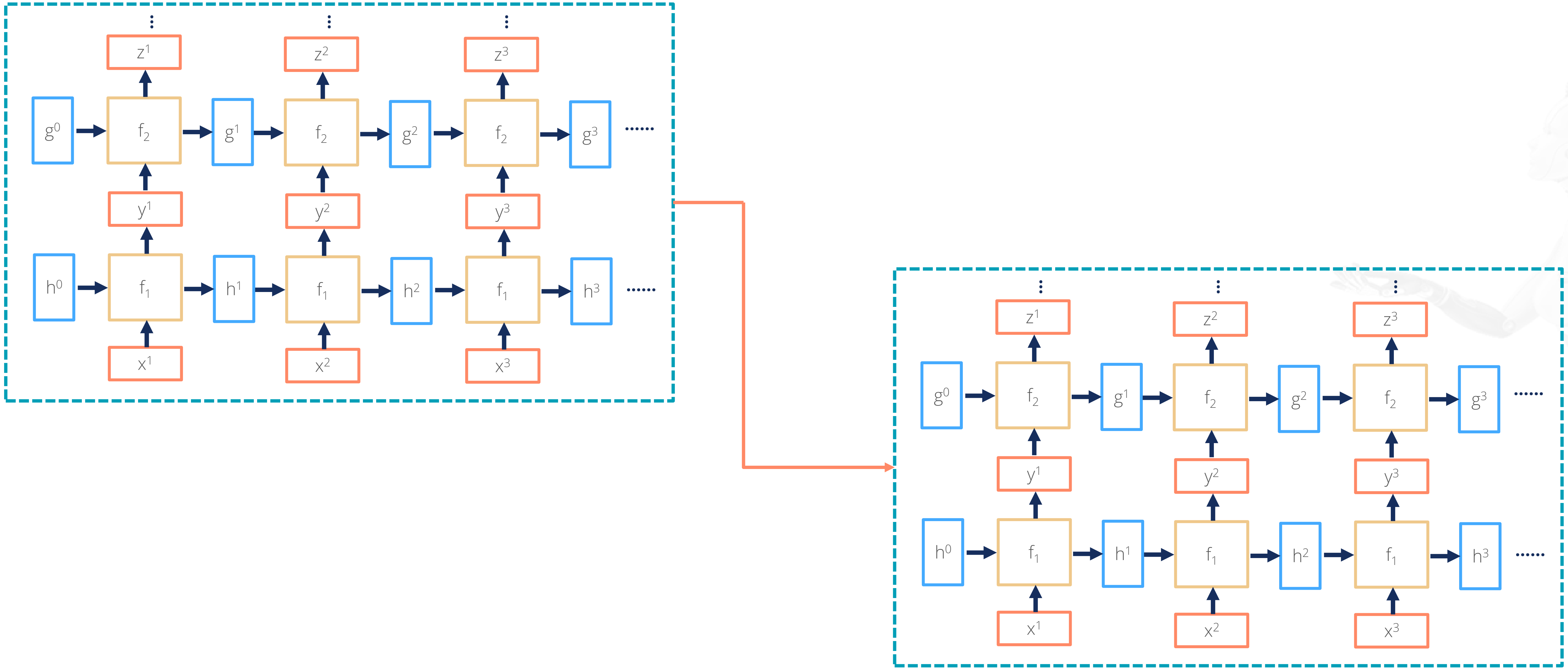
Pyramid RNNs

Pyramid RNNs speed up the training process by reducing the number of timesteps.



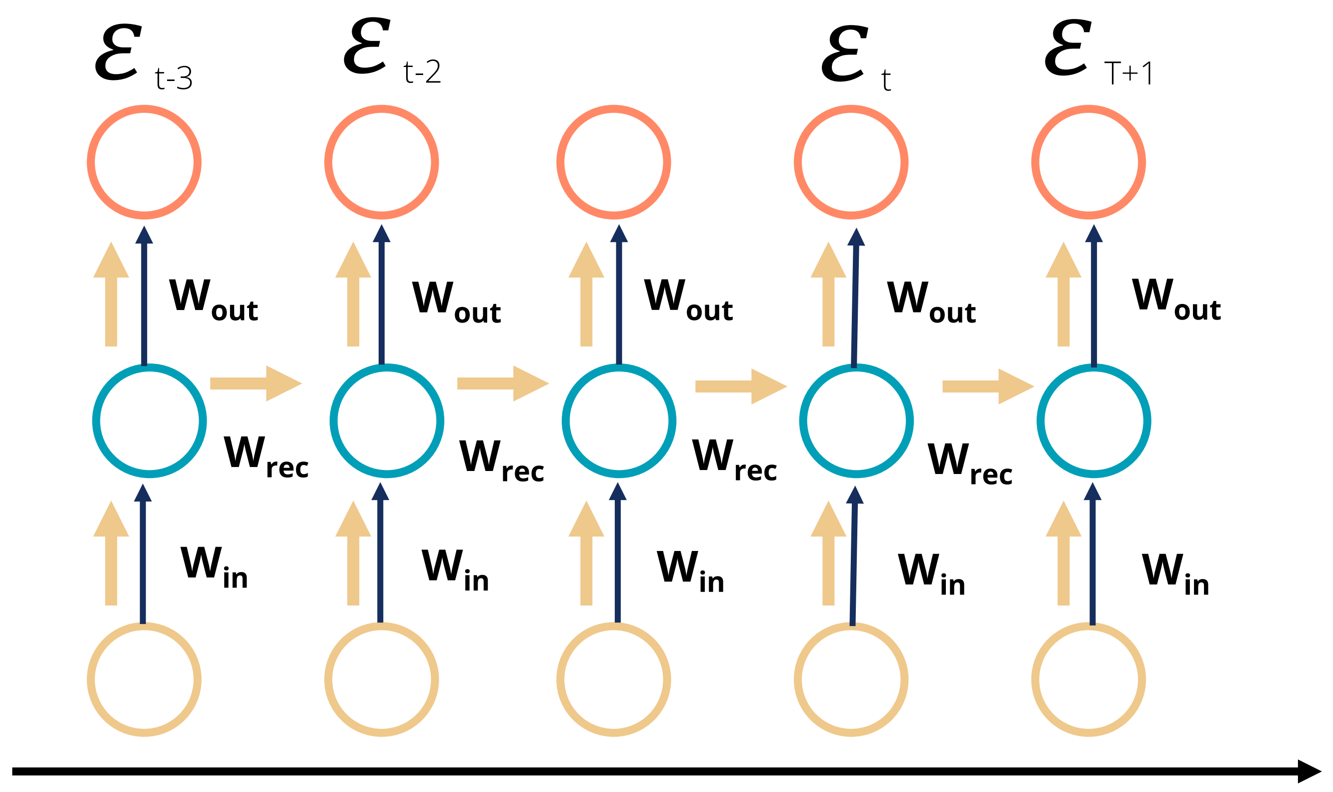
Problems with Deep RNNs

Deep RNNs are very hard to train and usually don't remember data beyond certain timesteps.



The Problem of Vanishing Gradient with RNNs

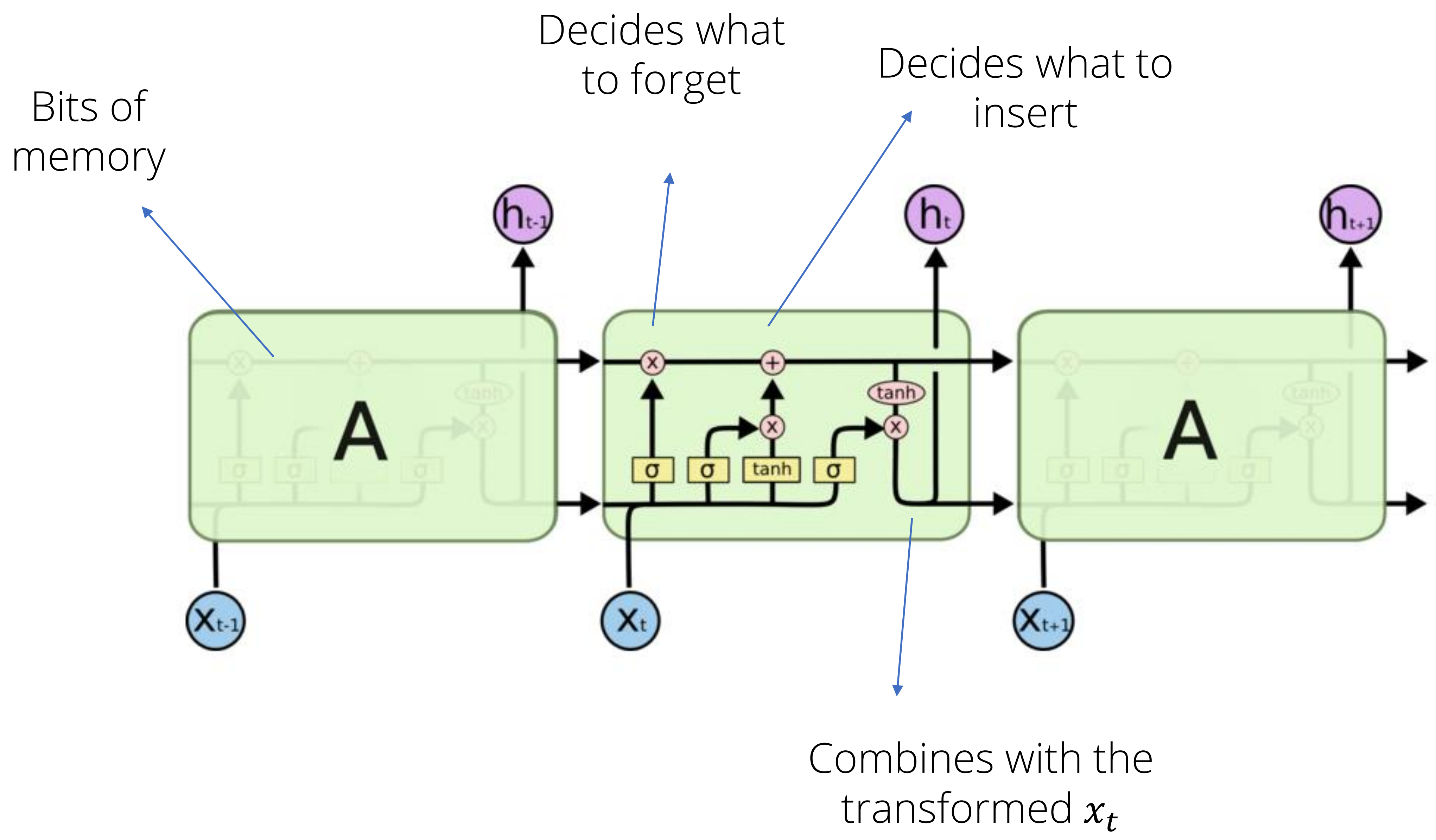
The problem arises while updating weights in RNNs. These weights connect the hidden layers to themselves in the unrolled temporal loop.



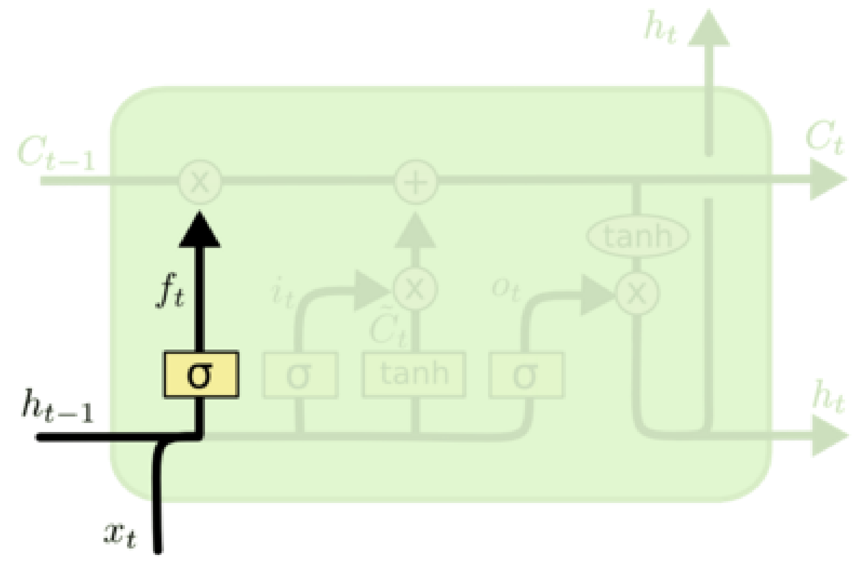
Note: When any figure is multiplied by a small number, its value decreases very quickly.

Long Short-Term Memory (LSTM)

LSTM Architecture



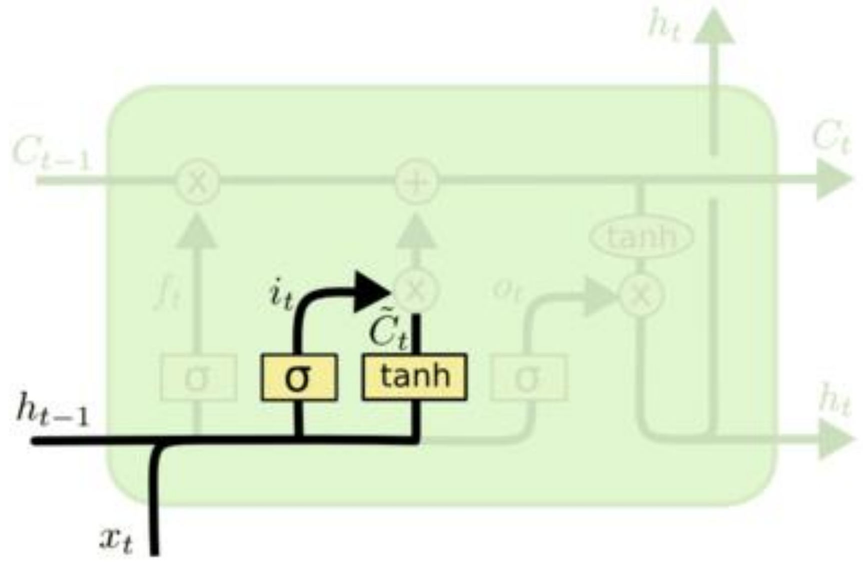
LSTM Architecture



Decides which part of memory to **forget**. The part to be forgotten is denoted with 0

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

LSTM Architecture



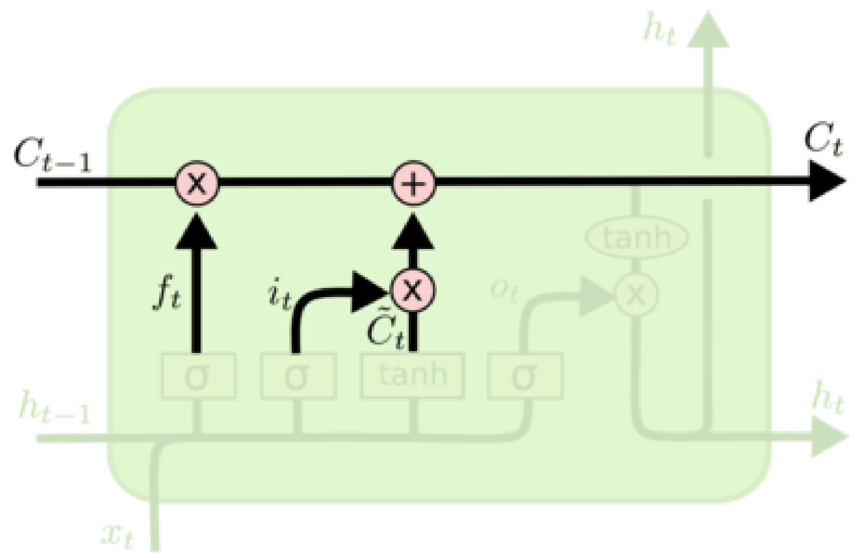
Decides what bits to insert in the next states

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Decides what content to store in the next states

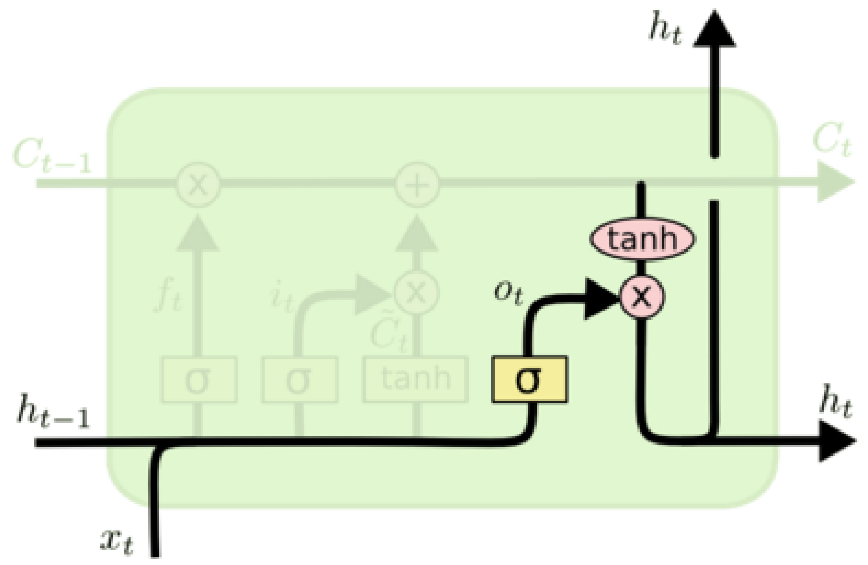
LSTM Architecture



Decides the content of the next memory cell, which is a mixture of the not forgotten part from previous cell and insertion

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

LSTM Architecture



Decides on what part of cell to output

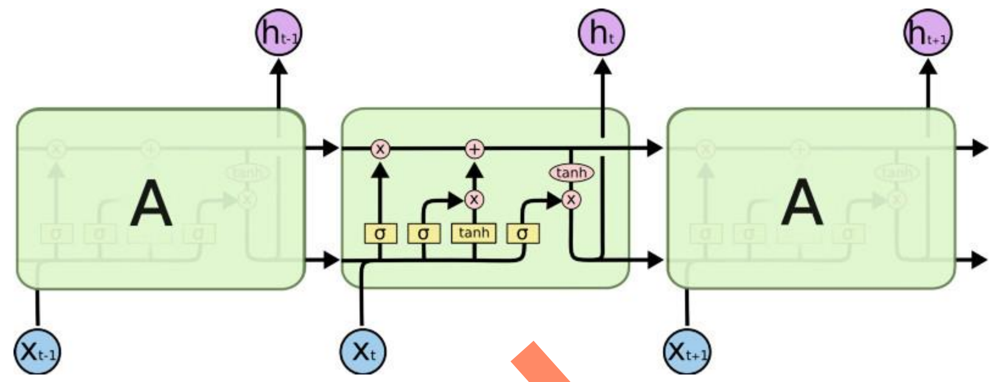
$$o_t = \sigma\left(W_o\,[h_{t-1}, x_t] + b_o\right)$$

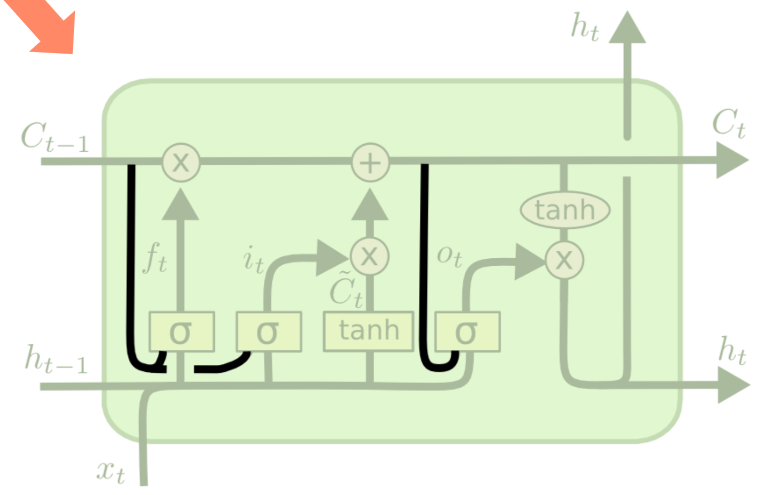
$$h_t = o_t * \tanh\left(C_t\right)$$

Maps bits within -1 and +1 range

A Peephole LSTM

A peephole LSTM allows **peeping** into the memory.

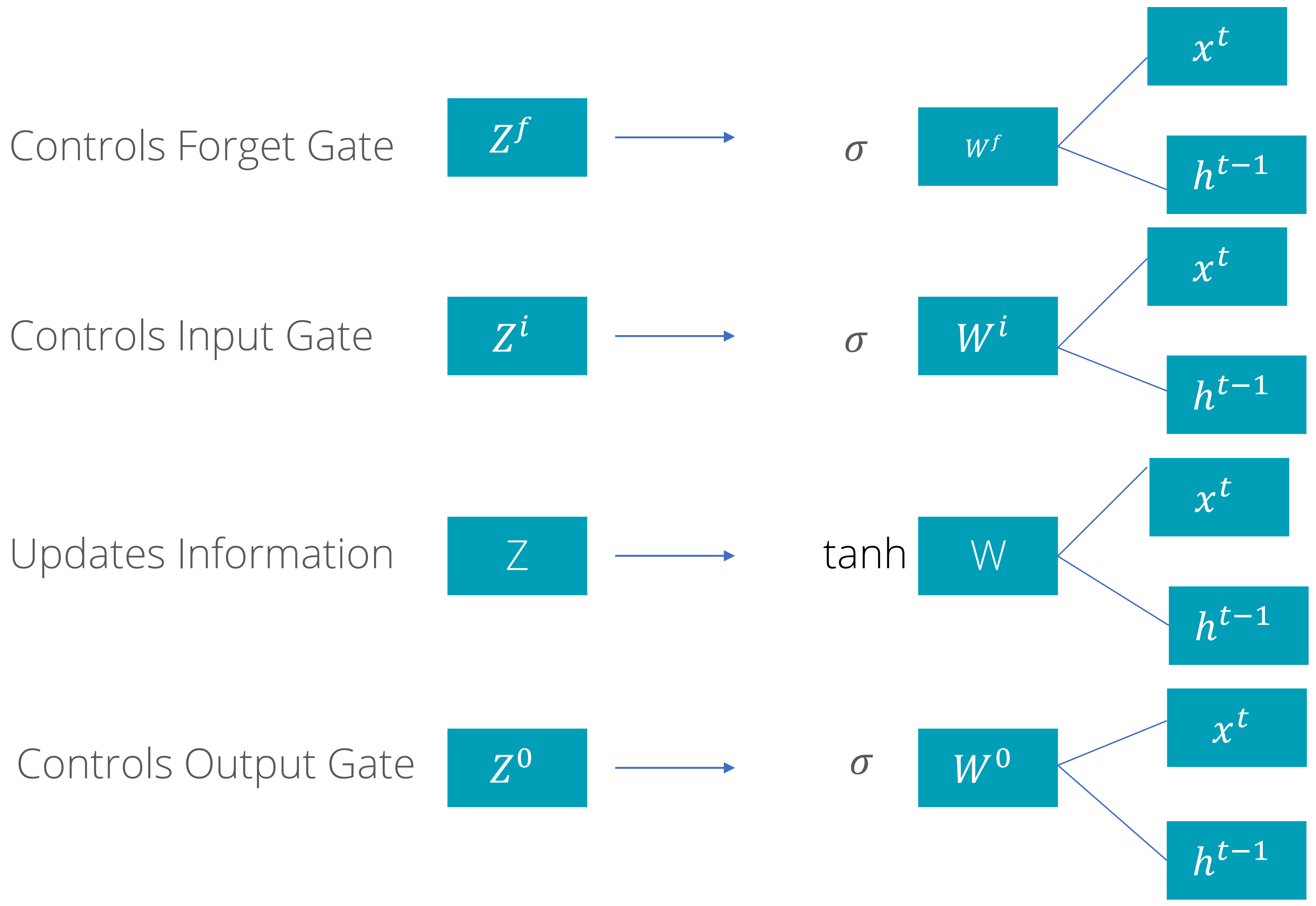




$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \ + \ b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \ + \ b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \ + \ b_o\right)$$

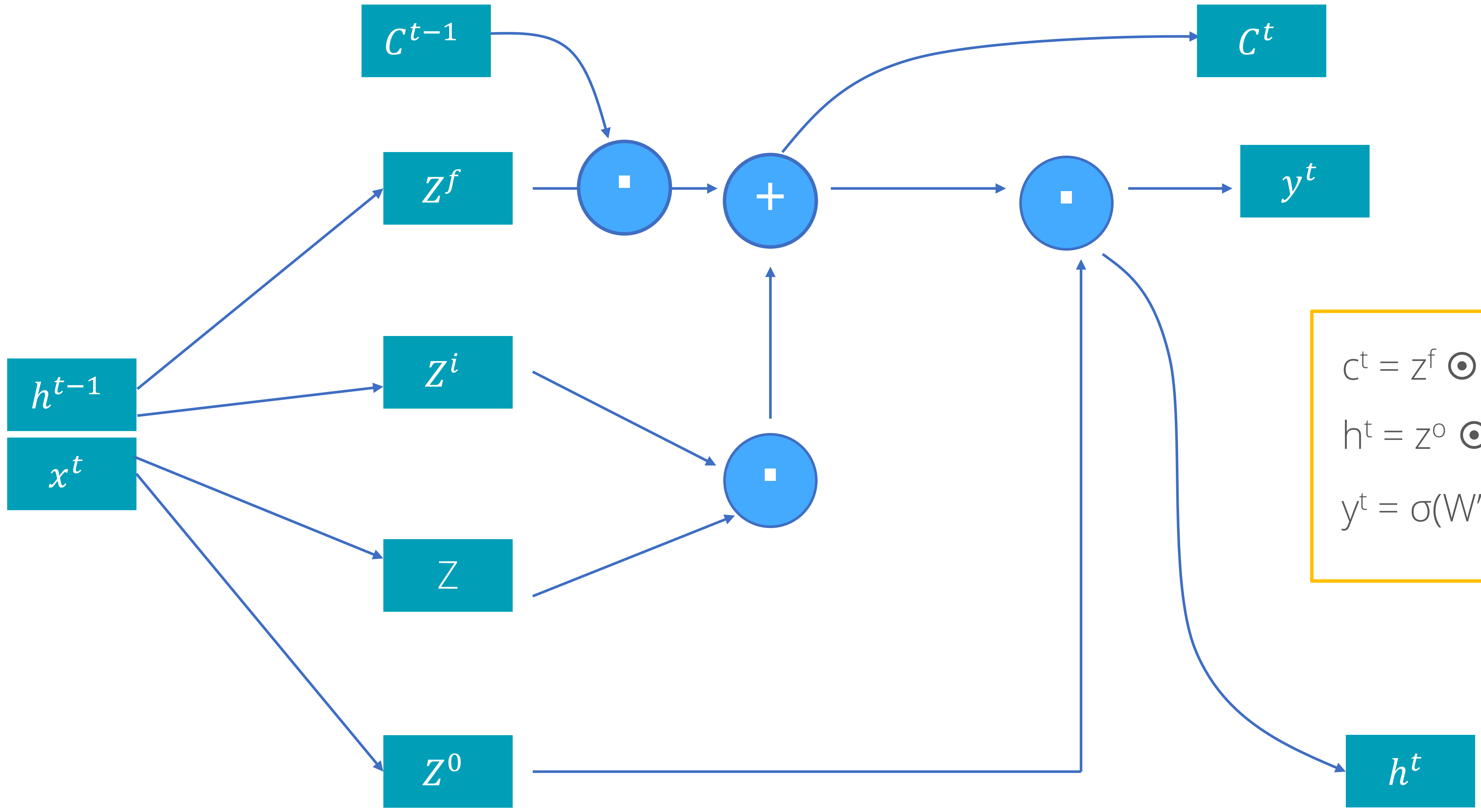
Information Flow in LSTM

Information Flow in LSTM



Note: Above four matrix computations are done concurrently.

Information Flow in LSTM



$c^t = z^f \odot c^{t-1} + z^i \odot z$

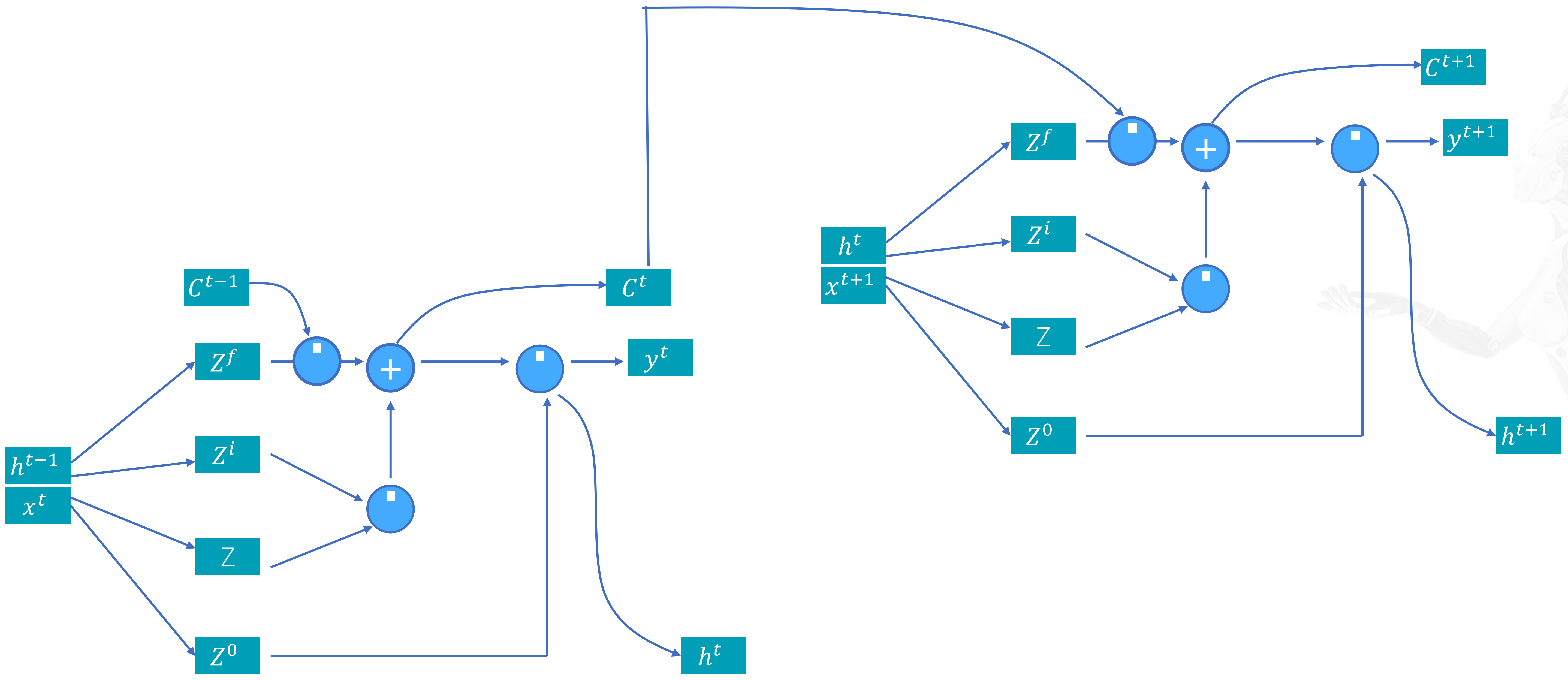
$h^t = z^o \odot \tanh(c^t)$

$y^t = \sigma(W' h^t)$

Note: $\odot$ signifies element-wise multiplication.

Information Flow in LSTM

The memory from one state is fed to another state along with the new input.



Stock Price Prediction Using LSTM

Problem Statement: Forecasting stock prices has been a difficult task for many of the researchers and analysts. There are a lot of complicated financial indicators, as a result of which the fluctuation of the stock market is highly volatile. The prediction of the market value is of great importance to help in maximizing the profit of stock option purchase while keeping the risk low.

Objective: Use LSTM approach to predict stock market indices on the dataset **prices.csv**.

Note: Prices dataset are fetched from Yahoo Finance, fundamentals are from Nasdaq Financials, extended by some fields from EDGAR SEC databases.

Access: Click on the Labs tab on the left side panel of the LMS. Copy or note the username and password that are generated. Click on the Launch Lab button. On the page that appears, enter the username and password in the respective fields, and click Login.

ASSISTED PRACTICE

Multiclass Classification Using LSTM

Problem Statement: You are given a news aggregator dataset which contains news headlines, URLs, and categories for 422.937 news stories collected by a web aggregator. These news articles have to be categorized into business, science and technology, entertainment, and health.

Objective: Perform multiclass classification using LSTM.

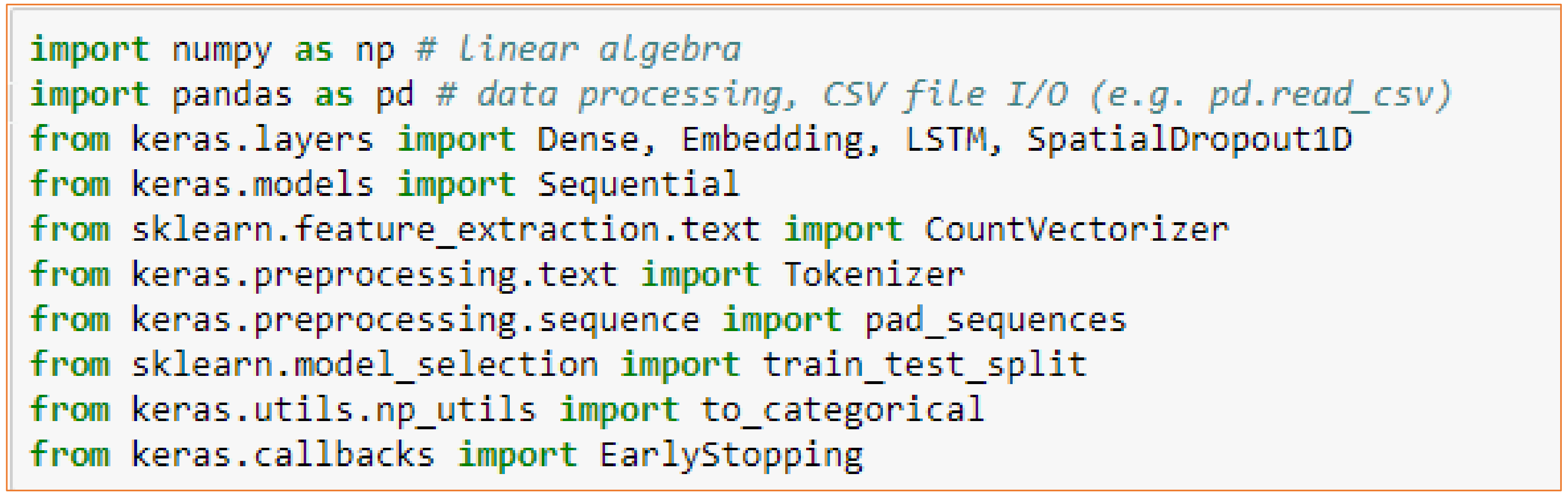
Note: Use **uci-news-aggregator.csv** for the above task.

Access: Click on the Labs tab on the left side panel of the LMS. Copy or note the username and password that are generated. Click on the Launch Lab button. On the page that appears, enter the username and password in the respective fields, and click Login.

UNASSISTED PRACTICE

Load Libraries

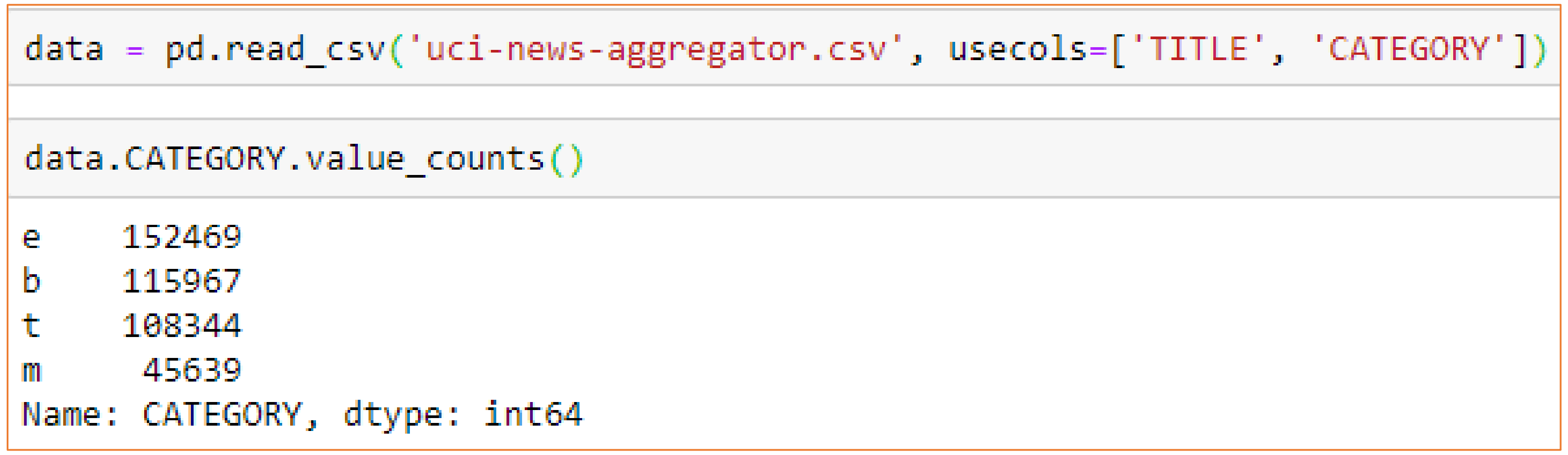
Import the necessary libraries.



```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D
from keras.models import Sequential
from sklearn.feature_extraction.text import CountVectorizer
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split
from keras.utils.np_utils import to_categorical
from keras.callbacks import EarlyStopping
```

Load Data

Load the .csv file and check the data count in each class.



```
data = pd.read_csv('uci-news-aggregator.csv', usecols=['TITLE', 'CATEGORY'])
```

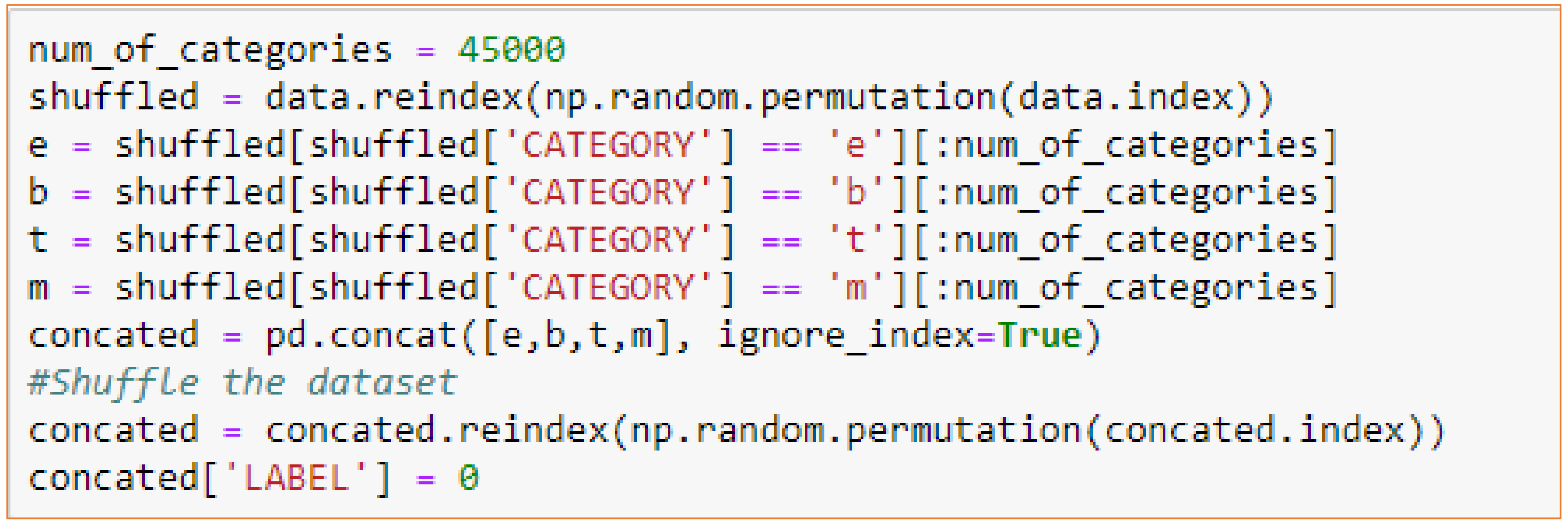
```
data.CATEGORY.value_counts()
```

```
e    152469
b    115967
t    108344
m     45639
Name: CATEGORY, dtype: int64
```

Note: m-class has way less data than the others, thus the classes are unbalanced.

Balance the Data

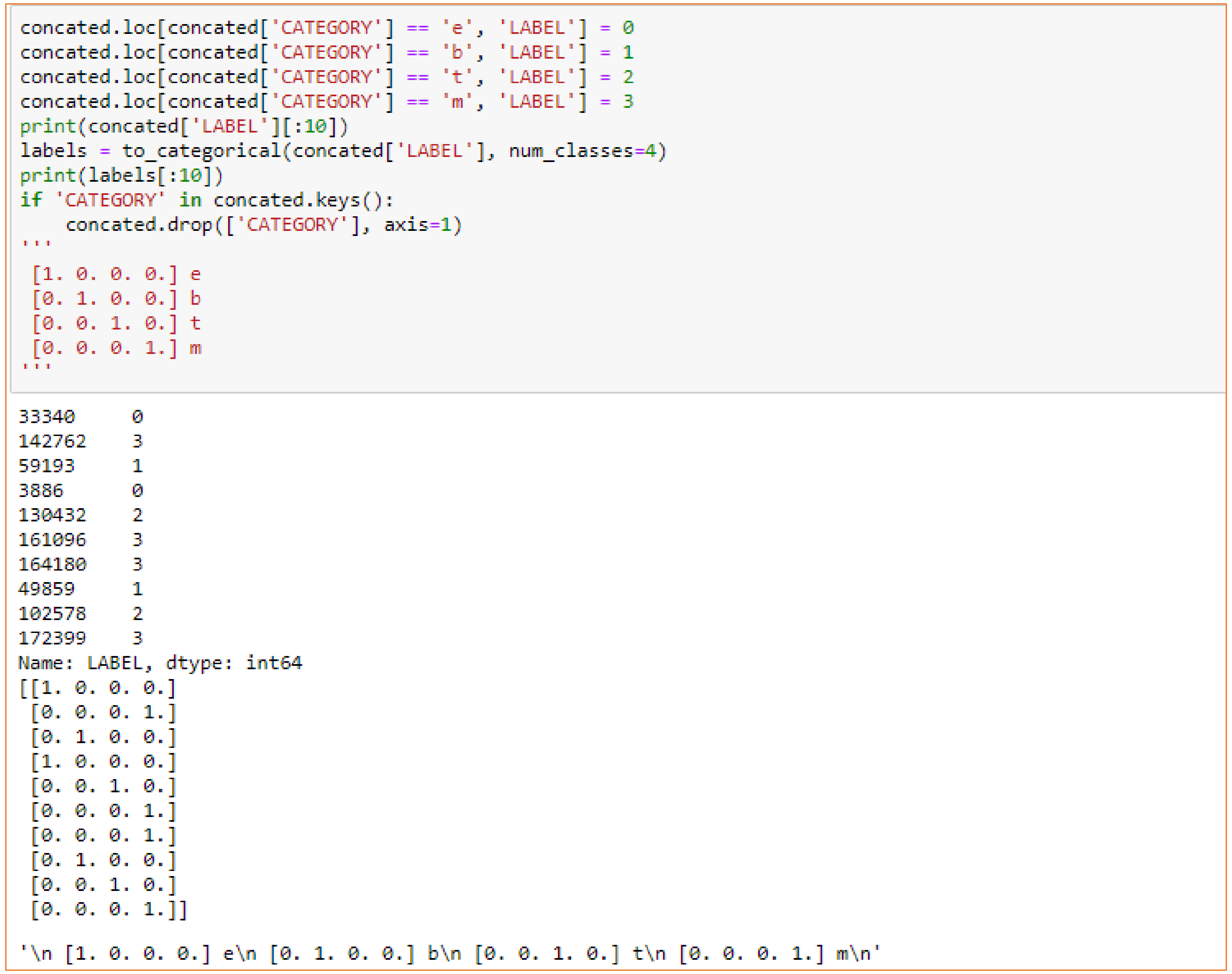
Perform shuffling to balance the classes.



```python
num_of_categories = 45000
shuffled = data.reindex(np.random.permutation(data.index))
e = shuffled[shuffled['CATEGORY'] == 'e'][:num_of_categories]
b = shuffled[shuffled['CATEGORY'] == 'b'][:num_of_categories]
t = shuffled[shuffled['CATEGORY'] == 't'][:num_of_categories]
m = shuffled[shuffled['CATEGORY'] == 'm'][:num_of_categories]
concated = pd.concat([e,b,t,m], ignore_index=True)
#Shuffle the dataset
concated = concated.reindex(np.random.permutation(concated.index))
concated['LABEL'] = 0
```

Encode the Data

Perform one-hot encoding on the labels data.

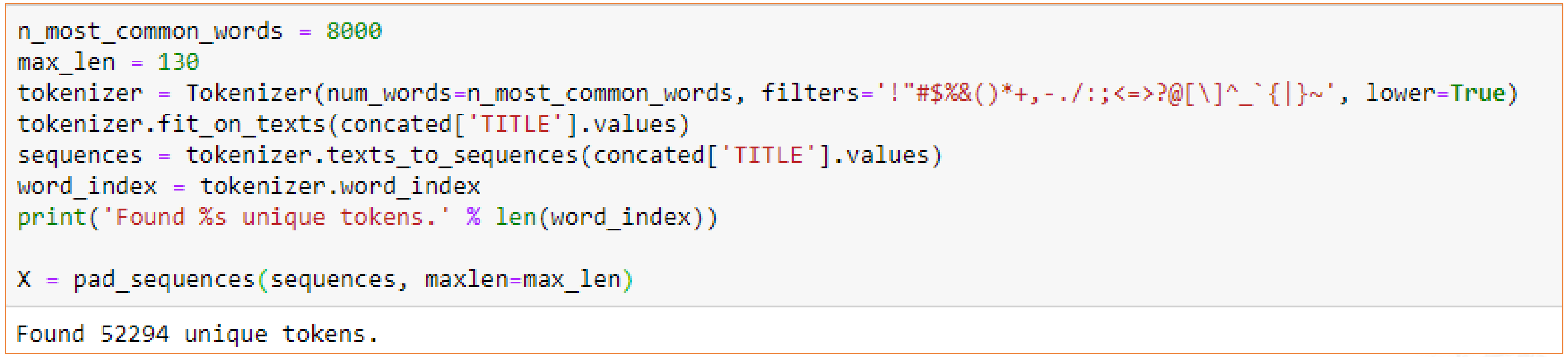


```
concated.loc[concated['CATEGORY'] == 'e', 'LABEL'] = 0
concated.loc[concated['CATEGORY'] == 'b', 'LABEL'] = 1
concated.loc[concated['CATEGORY'] == 't', 'LABEL'] = 2
concated.loc[concated['CATEGORY'] == 'm', 'LABEL'] = 3
print(concated['LABEL'][:10])
labels = to_categorical(concated['LABEL'], num_classes=4)
print(labels[:10])
if 'CATEGORY' in concated.keys():
    concated.drop(['CATEGORY'], axis=1)
'''
 [1. 0. 0. 0.] e
 [0. 1. 0. 0.] b
 [0. 0. 1. 0.] t
 [0. 0. 0. 1.] m
'''
```

```
33340     0
142762    3
59193     1
3886      0
130432    2
161096    3
164180    3
49859     1
102578    2
172399    3
Name: LABEL, dtype: int64
[[1. 0. 0. 0.]
 [0. 0. 0. 1.]
 [0. 1. 0. 0.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]
 [0. 0. 0. 1.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]

'\n [1. 0. 0. 0.] e\n [0. 1. 0. 0.] b\n [0. 0. 1. 0.] t\n [0. 0. 0. 1.] m\n'
```

Tokenization

Perform tokenization and identify the number of unique tokens.

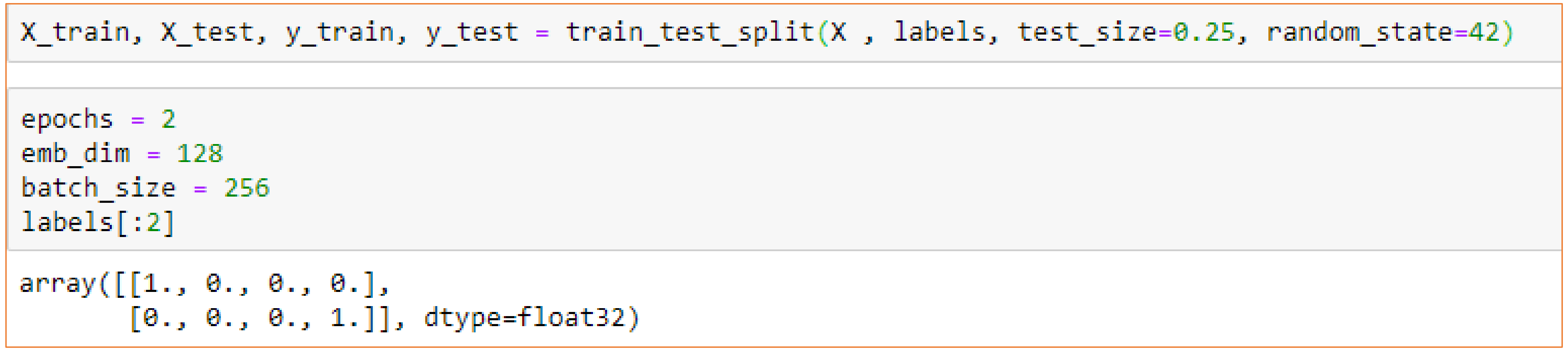


```
n_most_common_words = 8000
max_len = 130
tokenizer = Tokenizer(num_words=n_most_common_words, filters='!"#$%&()*+,-./:;<=>?@[\]^_`{|}~', lower=True)
tokenizer.fit_on_texts(concated['TITLE'].values)
sequences = tokenizer.texts_to_sequences(concated['TITLE'].values)
word_index = tokenizer.word_index
print('Found %s unique tokens.' % len(word_index))

X = pad_sequences(sequences, maxlen=max_len)
```

```
Found 52294 unique tokens.
```

Create Train and Test Sets

Split the dataset into training and testing sets. Also, define epochs, batch size, and labels for the same.



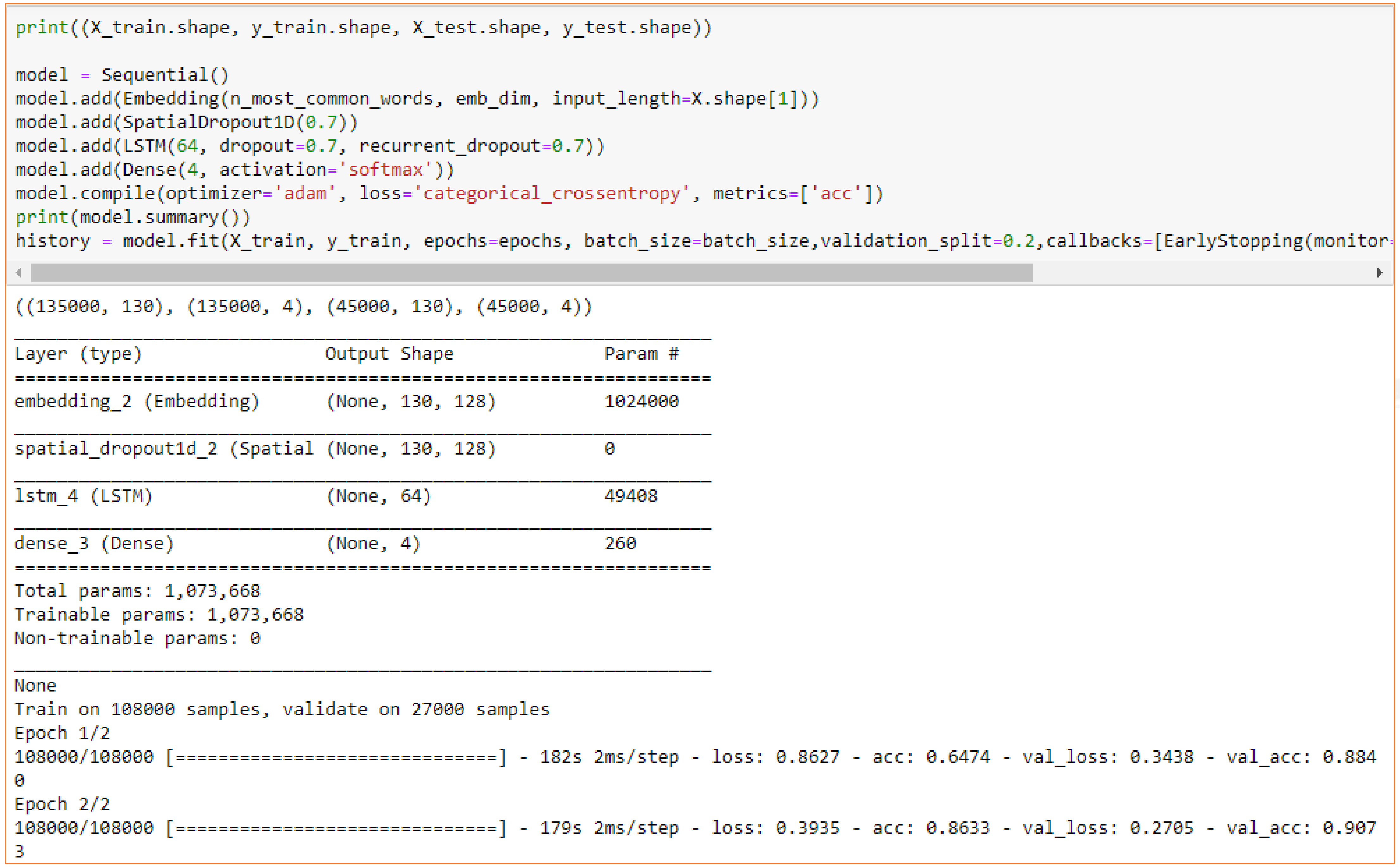
```
X_train, X_test, y_train, y_test = train_test_split(X , labels, test_size=0.25, random_state=42)
```

```
epochs = 2
emb_dim = 128
batch_size = 256
labels[:2]
```

```
array([[1., 0., 0., 0.],
       [0., 0., 0., 1.]], dtype=float32)
```

Define the LSTM Model

Code the LSTM model and fit the same into the processed data.

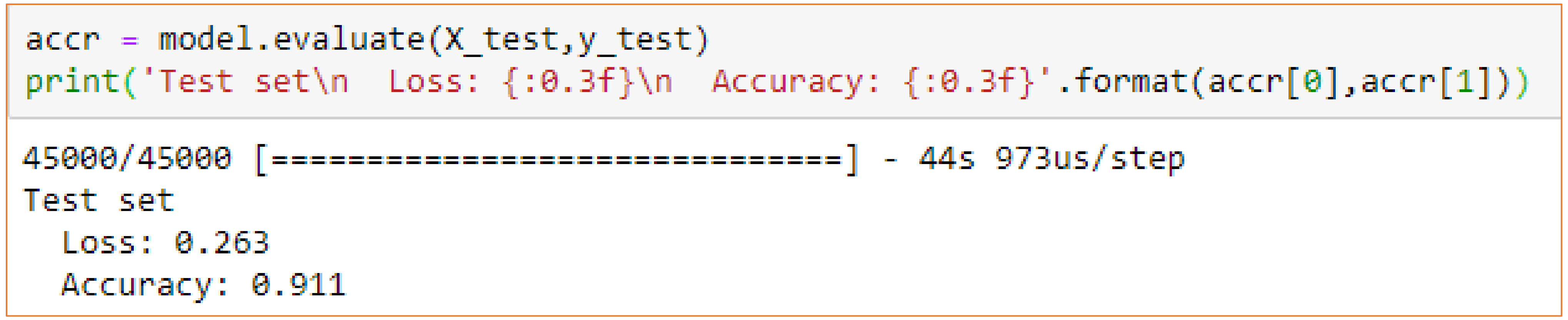


```
print((X_train.shape, y_train.shape, X_test.shape, y_test.shape))

model = Sequential()
model.add(Embedding(n_most_common_words, emb_dim, input_length=X.shape[1]))
model.add(SpatialDropout1D(0.7))
model.add(LSTM(64, dropout=0.7, recurrent_dropout=0.7))
model.add(Dense(4, activation='softmax'))
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['acc'])
print(model.summary())
history = model.fit(X_train, y_train, epochs=epochs, batch_size=batch_size,validation_split=0.2,callbacks=[EarlyStopping(monitor=
```

```
((135000, 130), (135000, 4), (45000, 130), (45000, 4))
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding_2 (Embedding)      (None, 130, 128)          1024000   
_________________________________________________________________
spatial_dropout1d_2 (Spatial (None, 130, 128)          0         
_________________________________________________________________
lstm_4 (LSTM)                (None, 64)                49408     
_________________________________________________________________
dense_3 (Dense)              (None, 4)                 260       
=================================================================
Total params: 1,073,668
Trainable params: 1,073,668
Non-trainable params: 0
_________________________________________________________________
None
Train on 108000 samples, validate on 27000 samples
Epoch 1/2
108000/108000 [==============================] - 182s 2ms/step - loss: 0.8627 - acc: 0.6474 - val_loss: 0.3438 - val_acc: 0.884
0
Epoch 2/2
108000/108000 [==============================] - 179s 2ms/step - loss: 0.3935 - acc: 0.8633 - val_loss: 0.2705 - val_acc: 0.907
3
```

Check Performance

Evaluate the results on training and testing sets and obtain accuracy of the model.

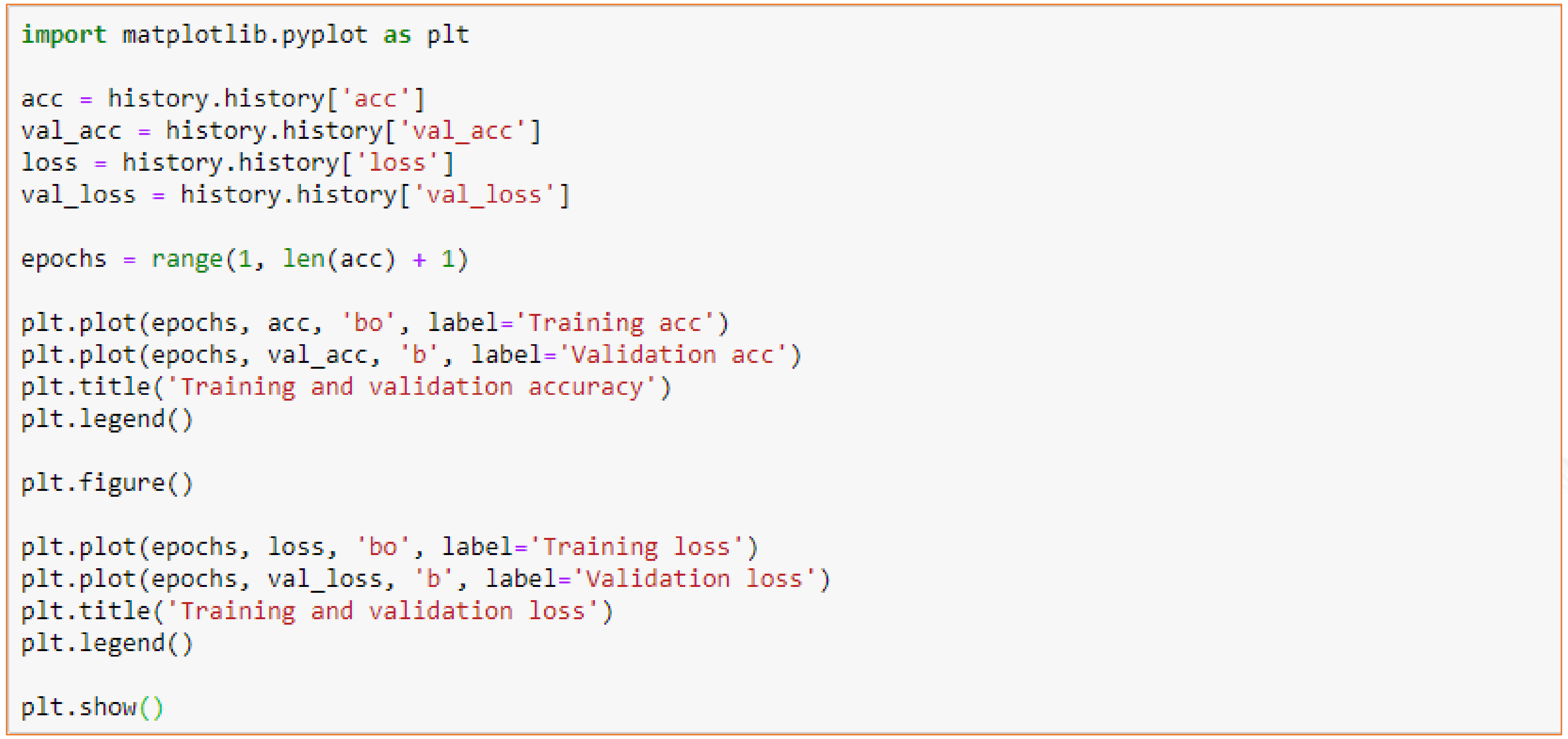


```
accr = model.evaluate(X_test,y_test)
print('Test set\n  Loss: {:0.3f}\n  Accuracy: {:0.3f}'.format(accr[0],accr[1]))
```

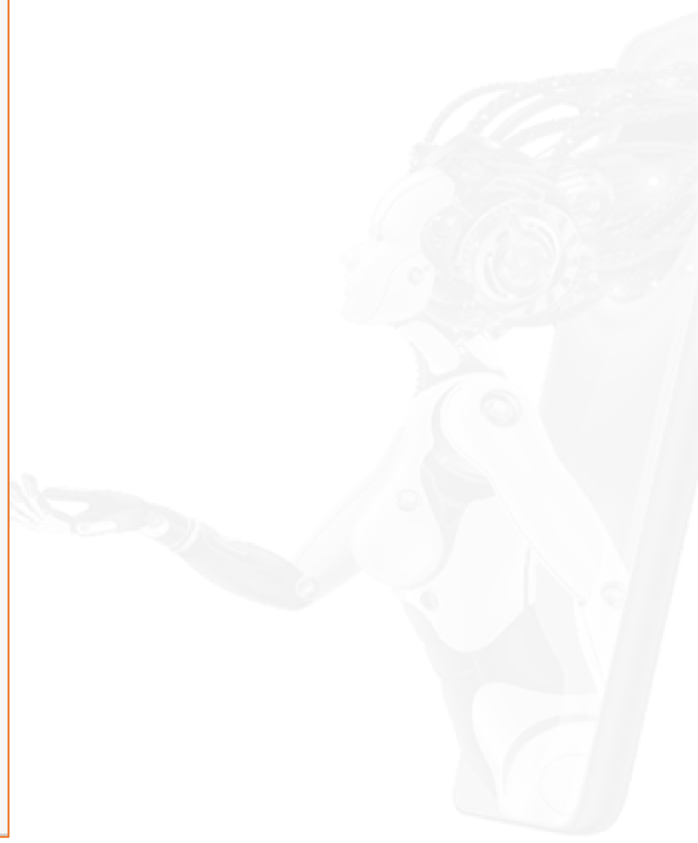
```
45000/45000 [==============================] - 44s 973us/step
Test set
  Loss: 0.263
  Accuracy: 0.911
```

Plot Metrics

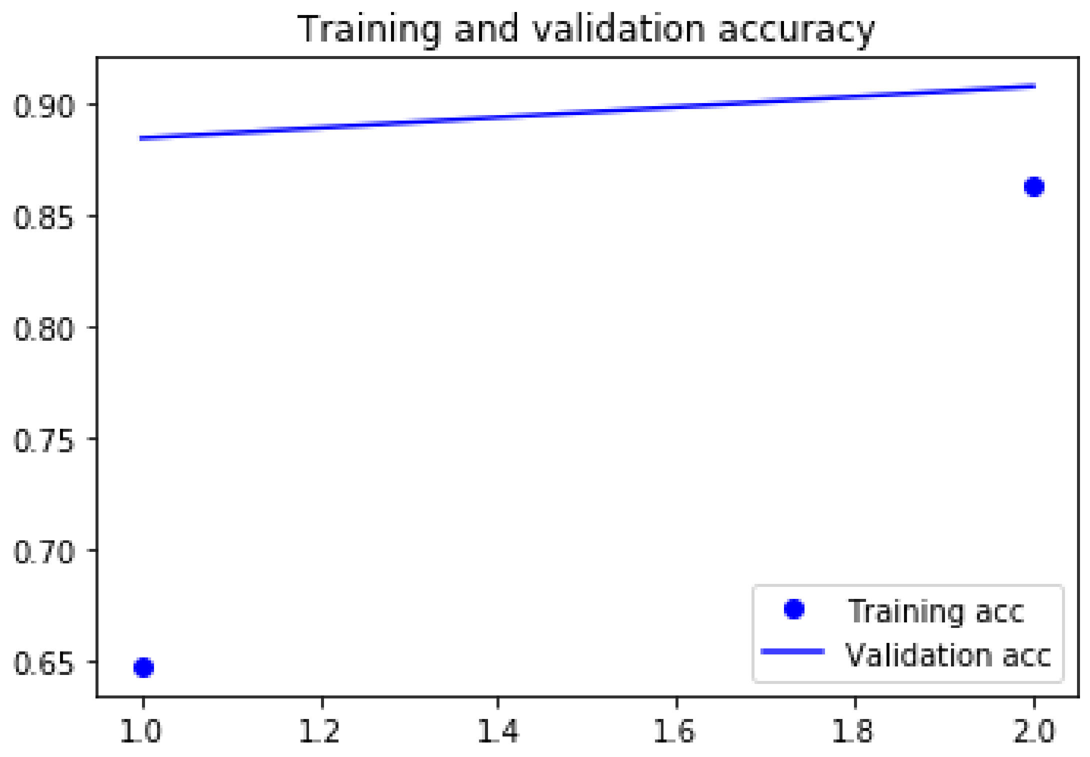
Plot the model’s training accuracy versus validation accuracy and training loss versus validation loss.

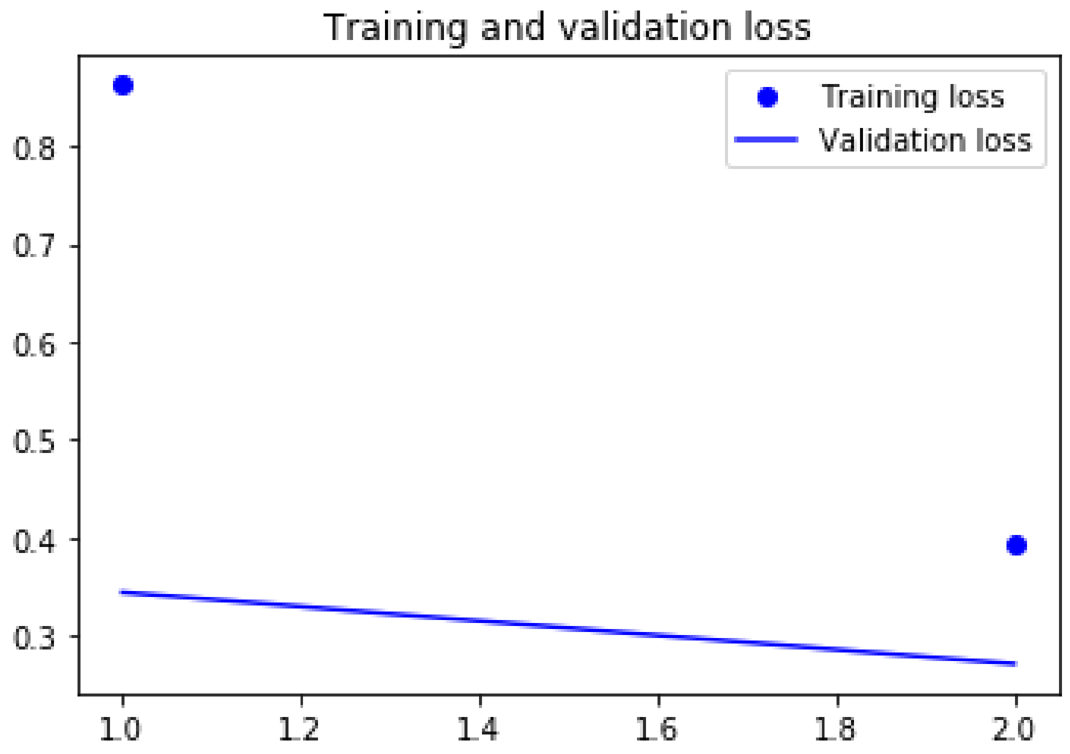


```
import matplotlib.pyplot as plt

acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()

plt.figure()

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```



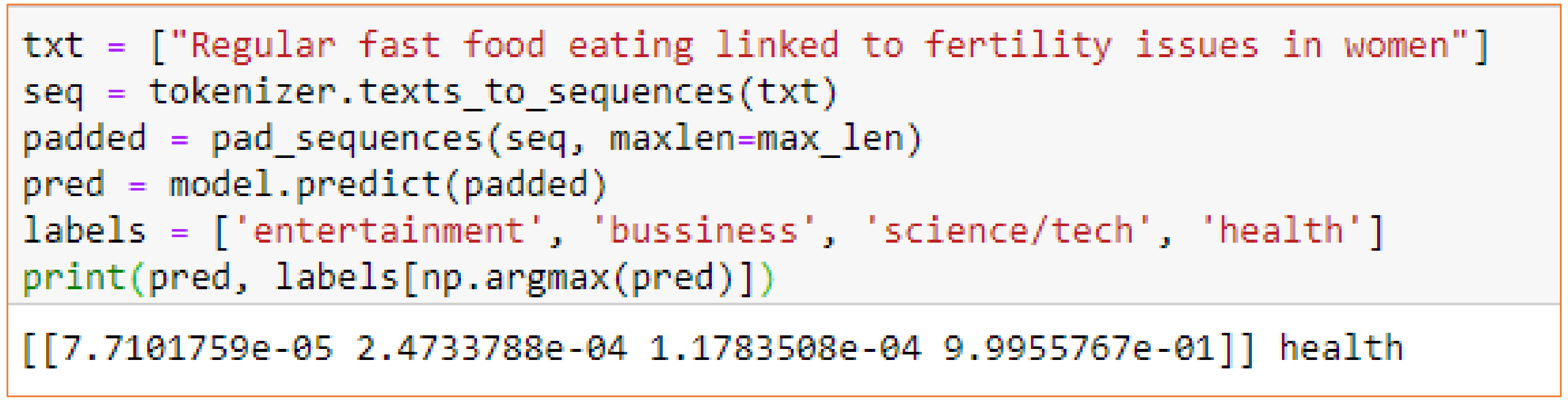
Plot Metrics





Perform Predictions

Perform label predictions against random data.



```
txt = ["Regular fast food eating linked to fertility issues in women"]
seq = tokenizer.texts_to_sequences(txt)
padded = pad_sequences(seq, maxlen=max_len)
pred = model.predict(padded)
labels = ['entertainment', 'bussiness', 'science/tech', 'health']
print(pred, labels[np.argmax(pred)])
```

```
[[7.7101759e-05 2.4733788e-04 1.1783508e-04 9.9955767e-01]] health
```

Sentiment Analysis Using LSTM

Problem Statement: Sentiment Analysis is one of the common problems that companies are working on. The most important application of sentiment analysis comes while working on natural language processing tasks. The motive of your company behind building a sentiment analyzer is to determine employee concerns and to develop programs to help improve the likelihood of employees remaining in their jobs.

Objective: Use LSTM to perform sentiment analysis in Keras.

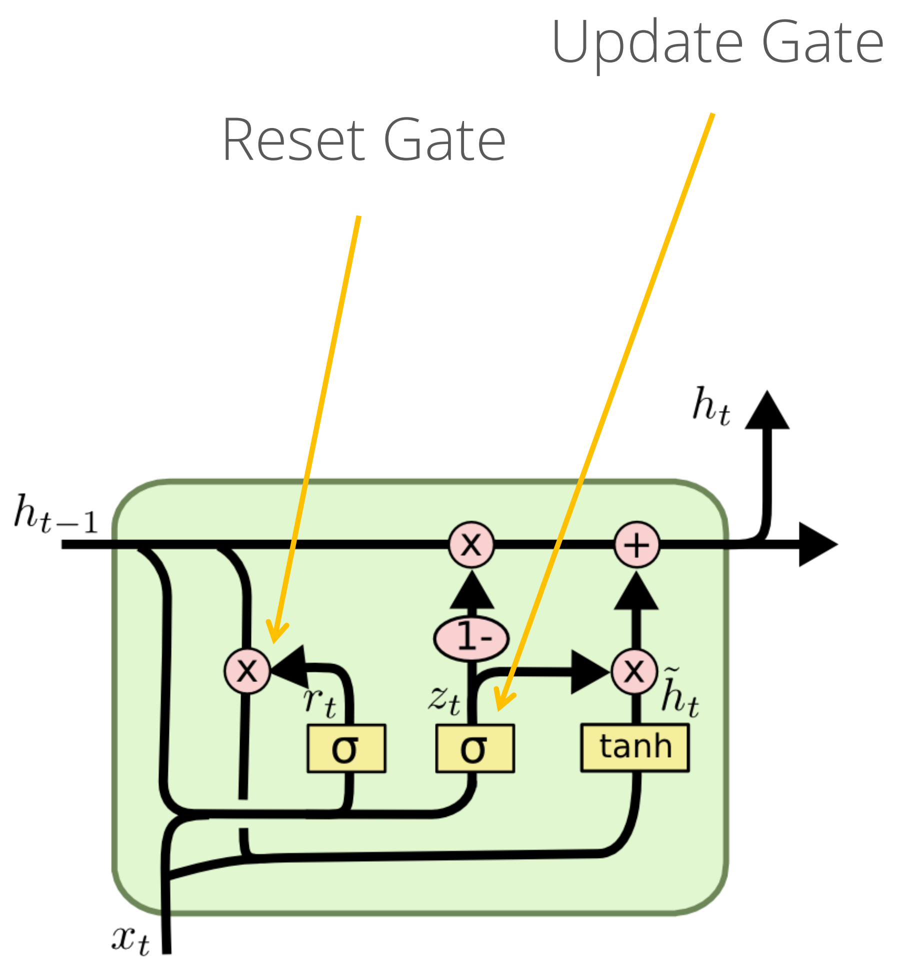
Note: Use the inbuilt dataset **imdb** from keras.datasets for this task.

Access: Click on the Labs tab on the left side panel of the LMS. Copy or note the username and password that are generated. Click on the Launch Lab button. On the page that appears, enter the username and password in the respective fields, and click Login.

Gated Recurrent Unit (GRU)

GRU Architecture

Performs label predictions against random data.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

GRU Architecture

Update Gate

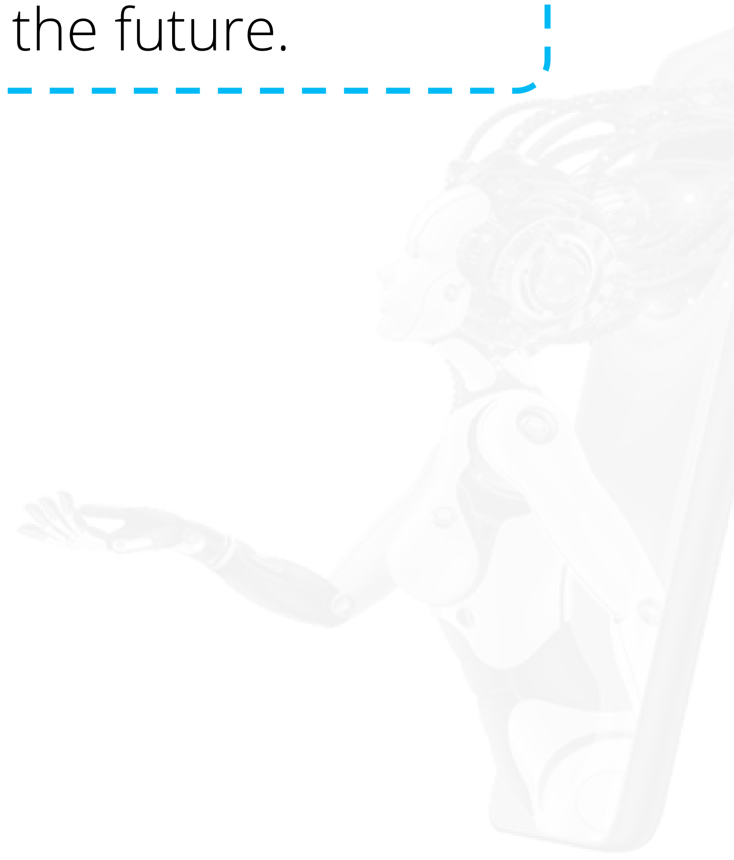
Reset Gate

Current Memory

Current State

Determines how much of the past information (from the previous time steps) needs to be passed along to the future.

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1})$$



GRU Architecture

Update Gate

Reset Gate

Current Memory

Current State

Determines how much of the past information needs to be forgotten.

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1})$$

simplilearn

GRU Architecture

Update Gate

Reset Gate

Current Memory

Current State

The current memory is computed using the reset gate to store relevant information from the past.

$$h'_t = \tanh(Wx_t + r_t \odot Uh_{t-1})$$

Note: Here tanh is the nonlinear activation function.

Update Gate

Reset Gate

Current Memory

Current State

GRU Architecture

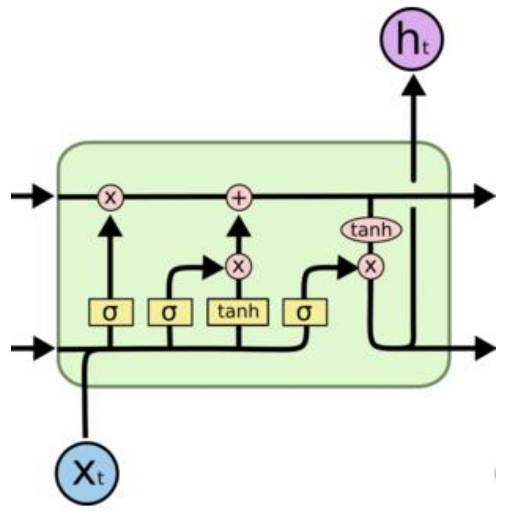
At the final stage, h_t vector is calculated such that it holds the information for the current unit and passes it down to the network.

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h_t'$$

simplilearn

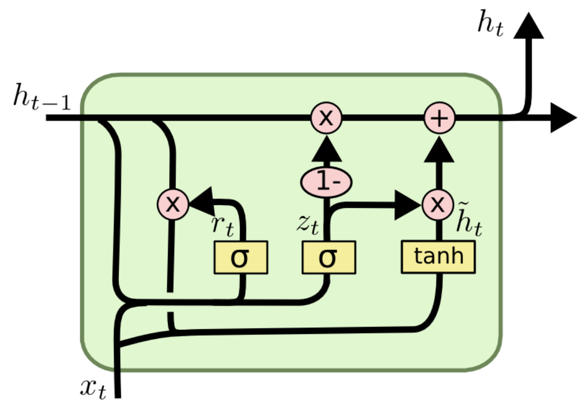
LSTM vs. GRU

LSTM



- Tracks long-term dependencies while mitigating the vanishing or exploding gradient problems. It does so via input, forget, and output gates.
- Controls the exposure of memory content

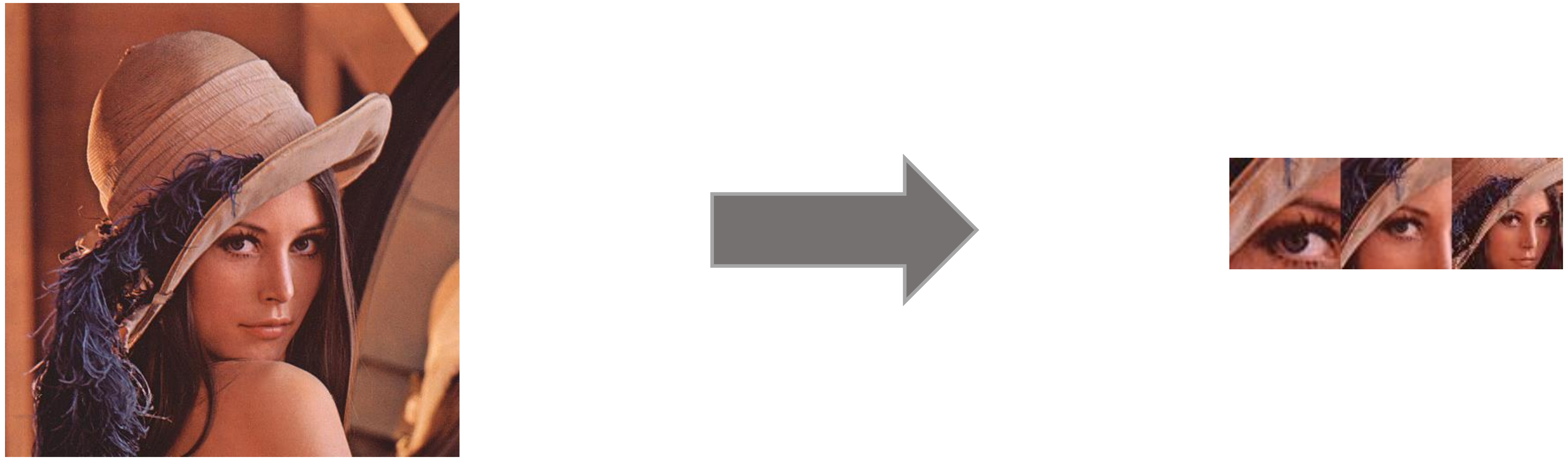
GRU

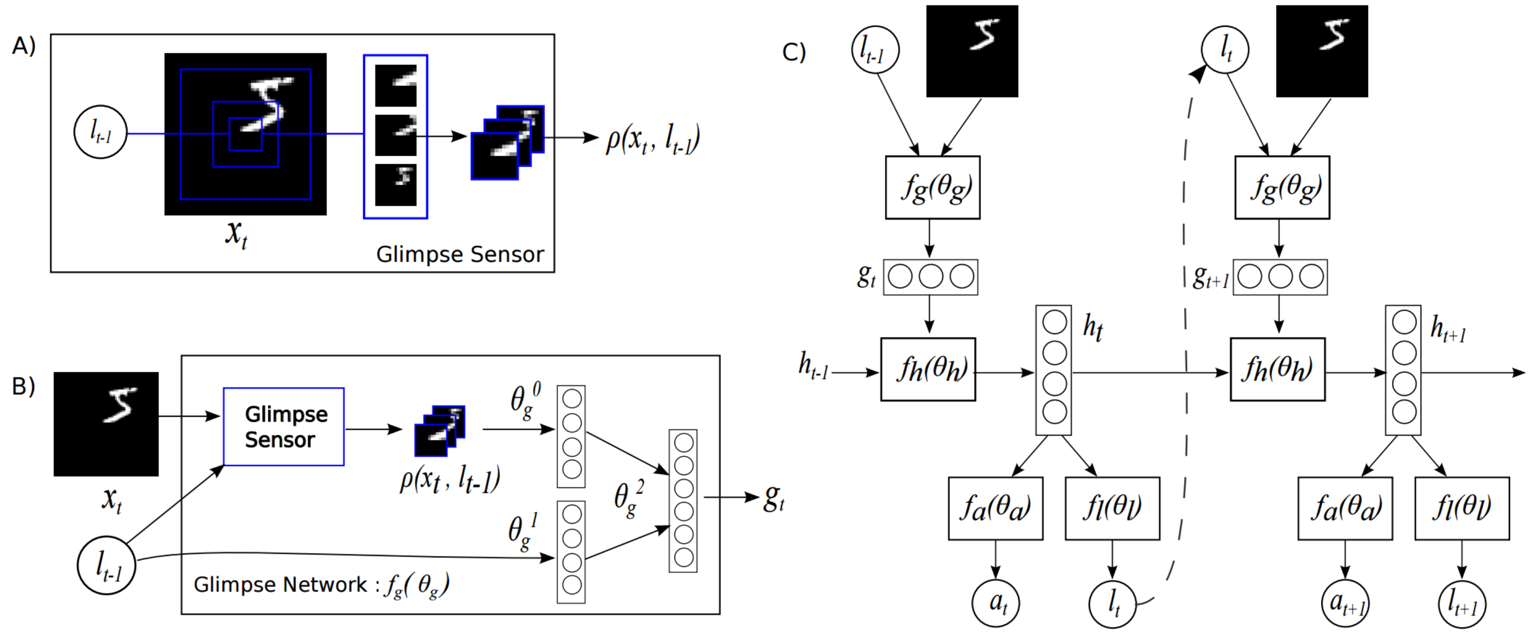


- Tracks long-term dependencies using a reset gate and an update gate
- Exposes the entire cell state to other units in the network

The Attention Model

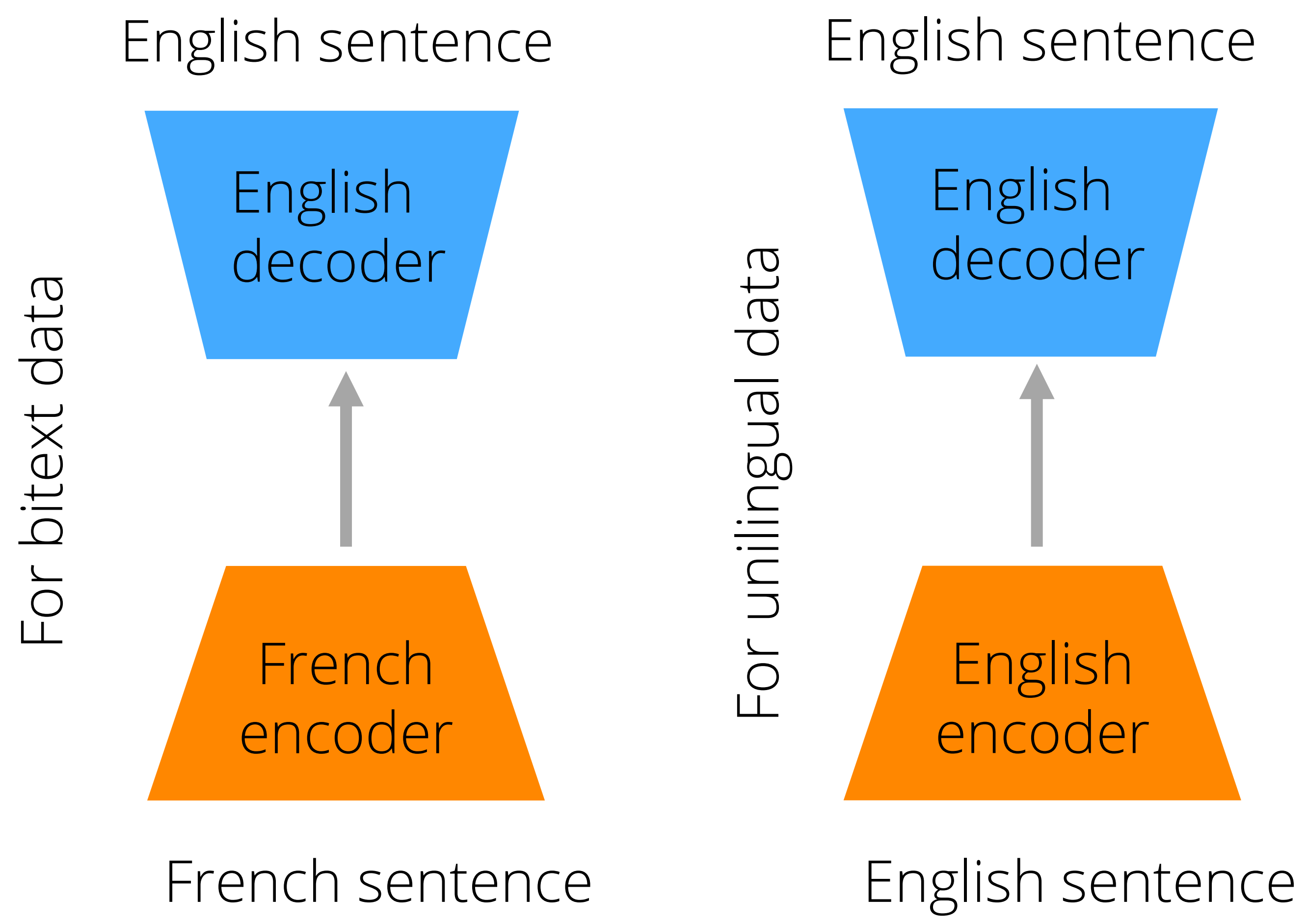
Attention Model





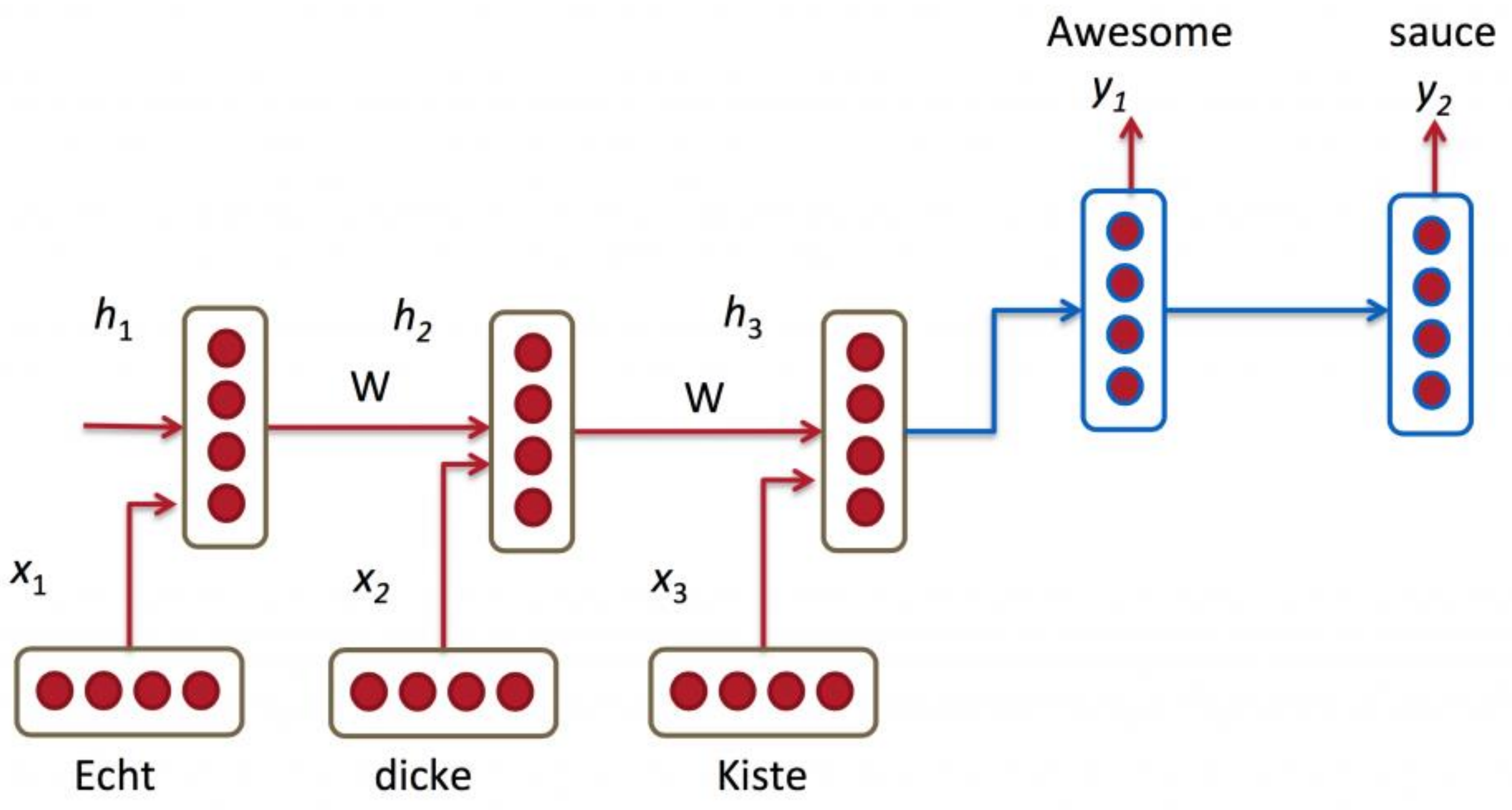
Encoder-Decoder Framework

- Encoder: From word sequence to sentence representation
- Decoder: From representation to word sequence distribution
- Universal Representation: Intermediate representation of meaning



Motivation

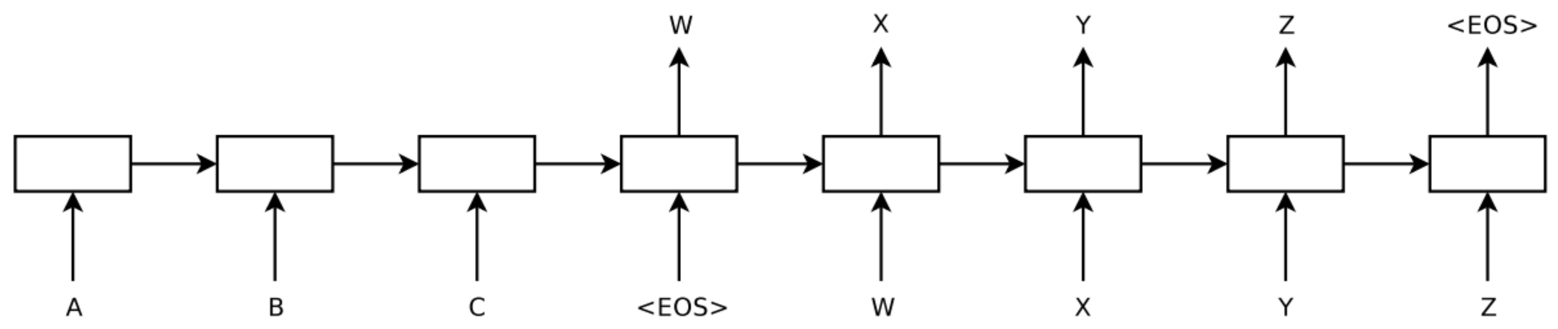
- Limited representation
- Constrained over longer distances

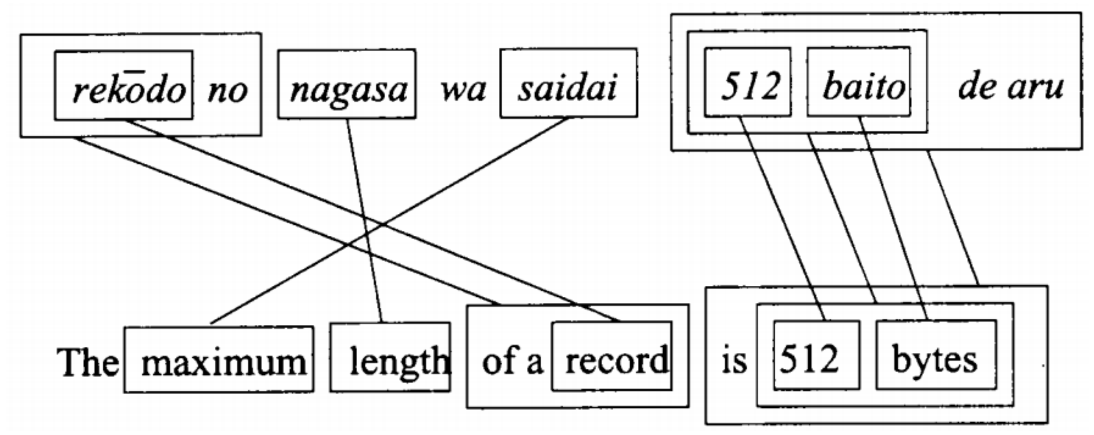


Improving Performance with LSTM

Reversing the order

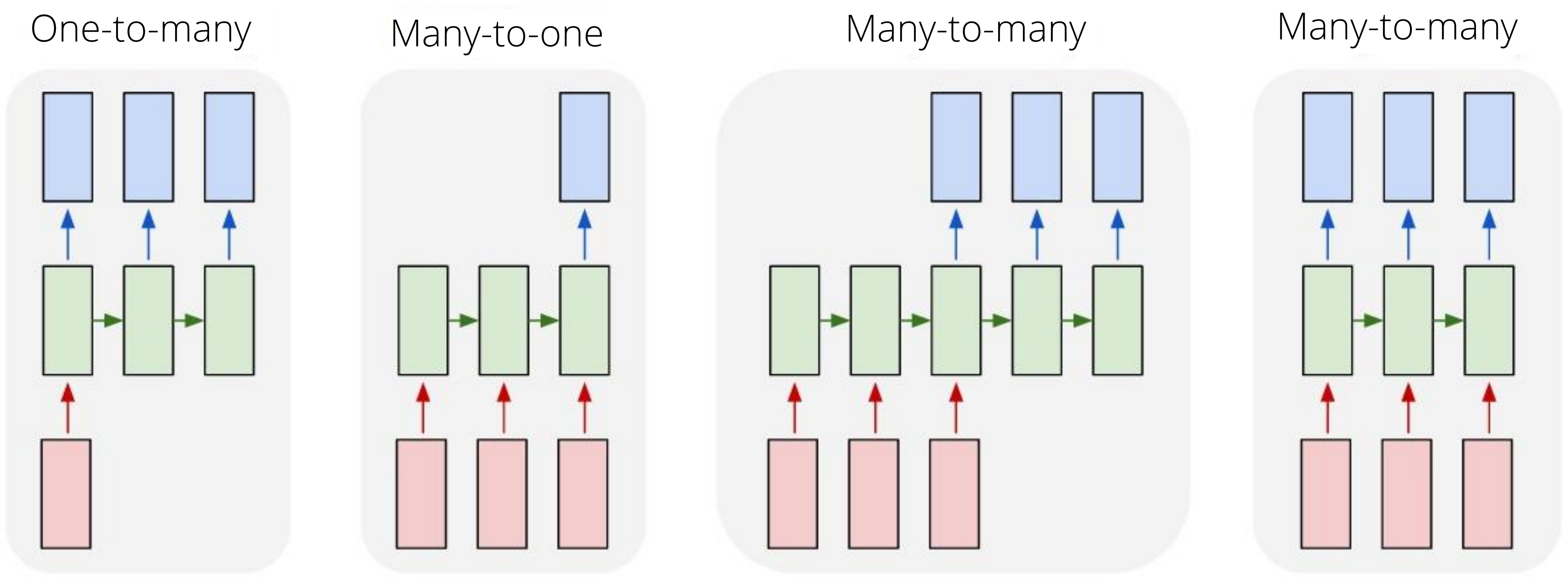
Instead of mapping the sentence **a, b, c** to the sentence **α, β, γ**, the LSTM is asked to map **c, b, a** to **α, β, γ**, where **α, β, γ** is the translation of **a, b, c**. This way, **a** is in close proximity to **α, b** is fairly close to **β**, and so on.





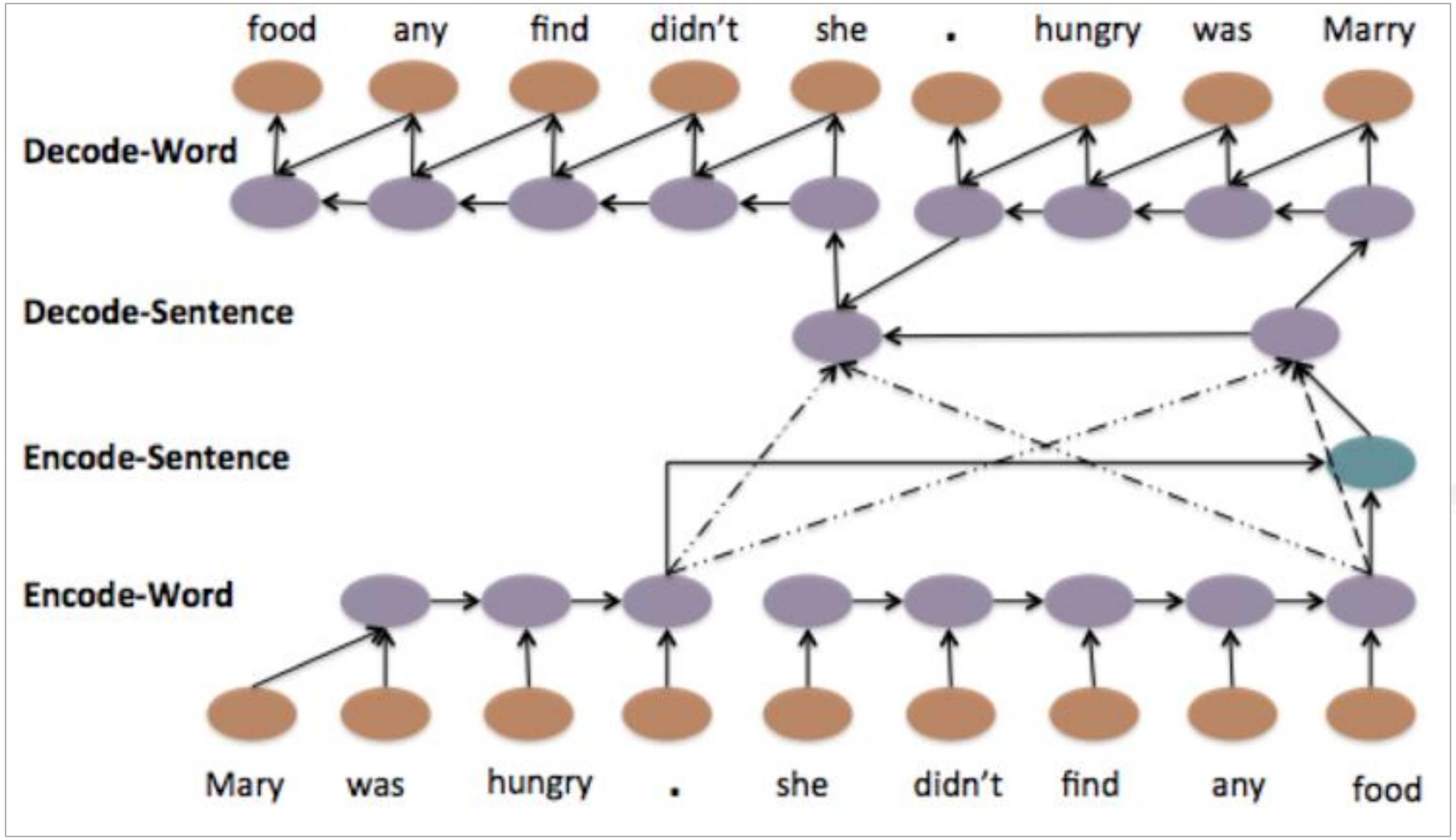
Examples of Attention

Example 1



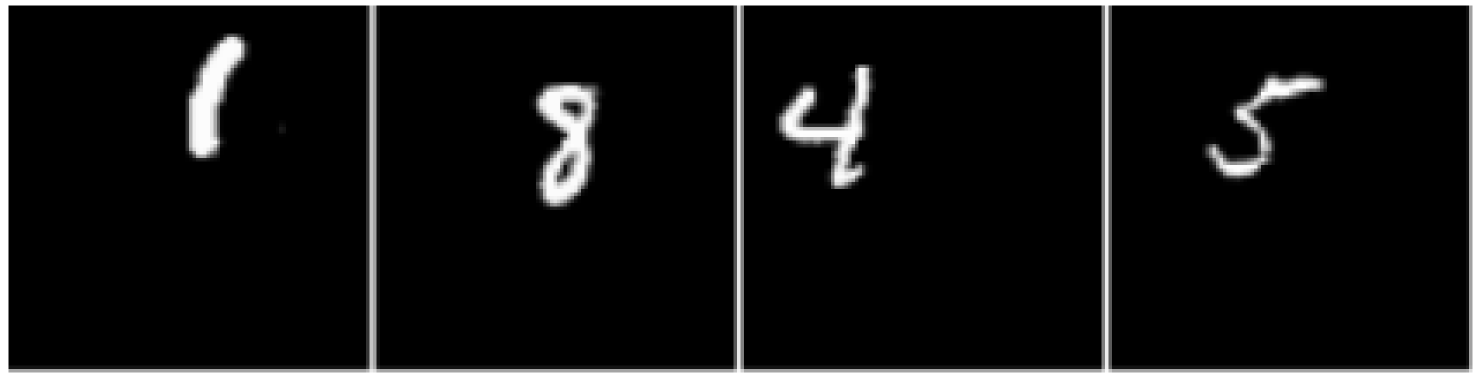
The way an LSTM chooses what to **forget** and what to **insert** into memory, determines what inputs the network will **attend to** in the generation phase

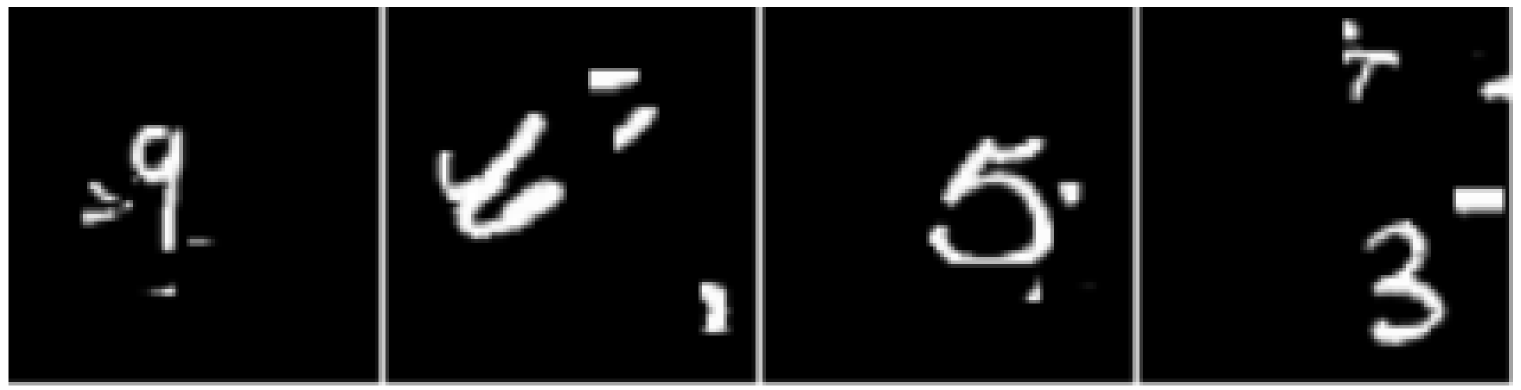
Example 2


food
any
find
didn't
she
.
hungry
was
Marry
Decode-Word
Decode-Sentence
Encode-Sentence
Encode-Word
Mary
was
hungry
.
she
didn't
find
any
food

Example 3

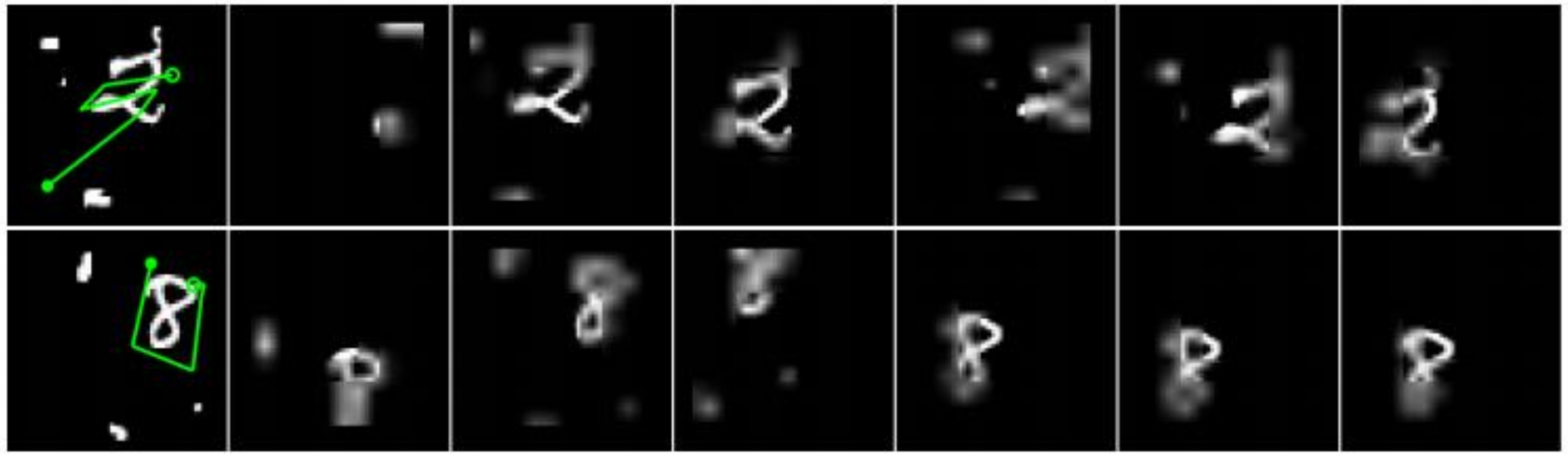
Translated MNIST inputs





Cluttered Translated MNIST inputs

Example 4

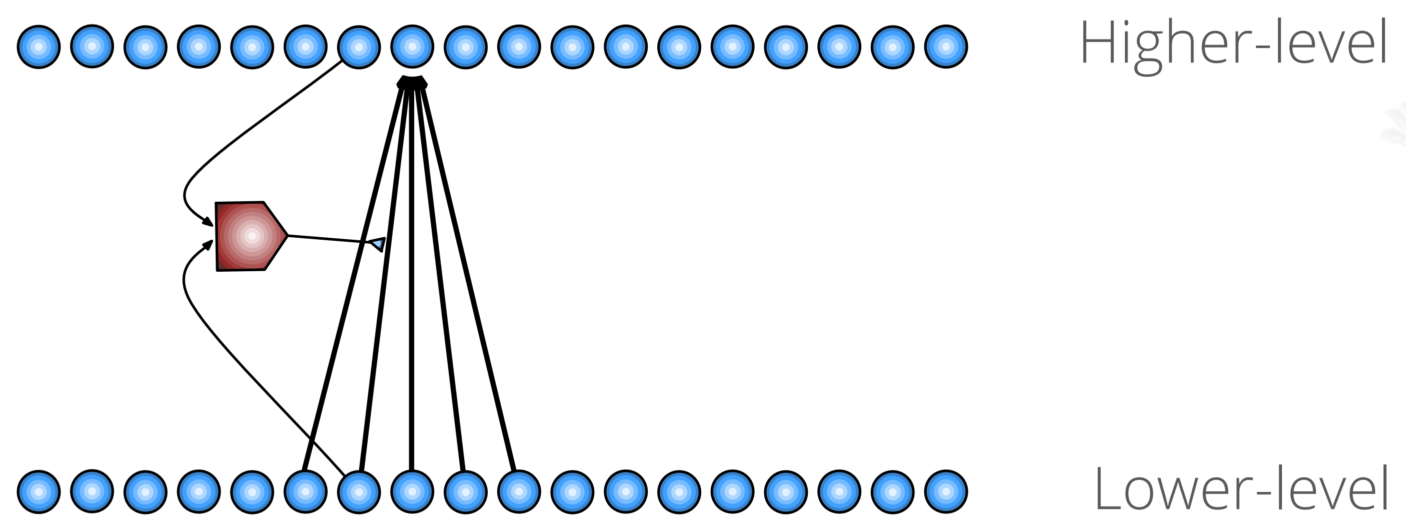


Similar to the way an LSTM chooses what to **forget** and **insert** into memory, allow a network to **choose a path to focus on** in the visual field

The Attention Mechanism

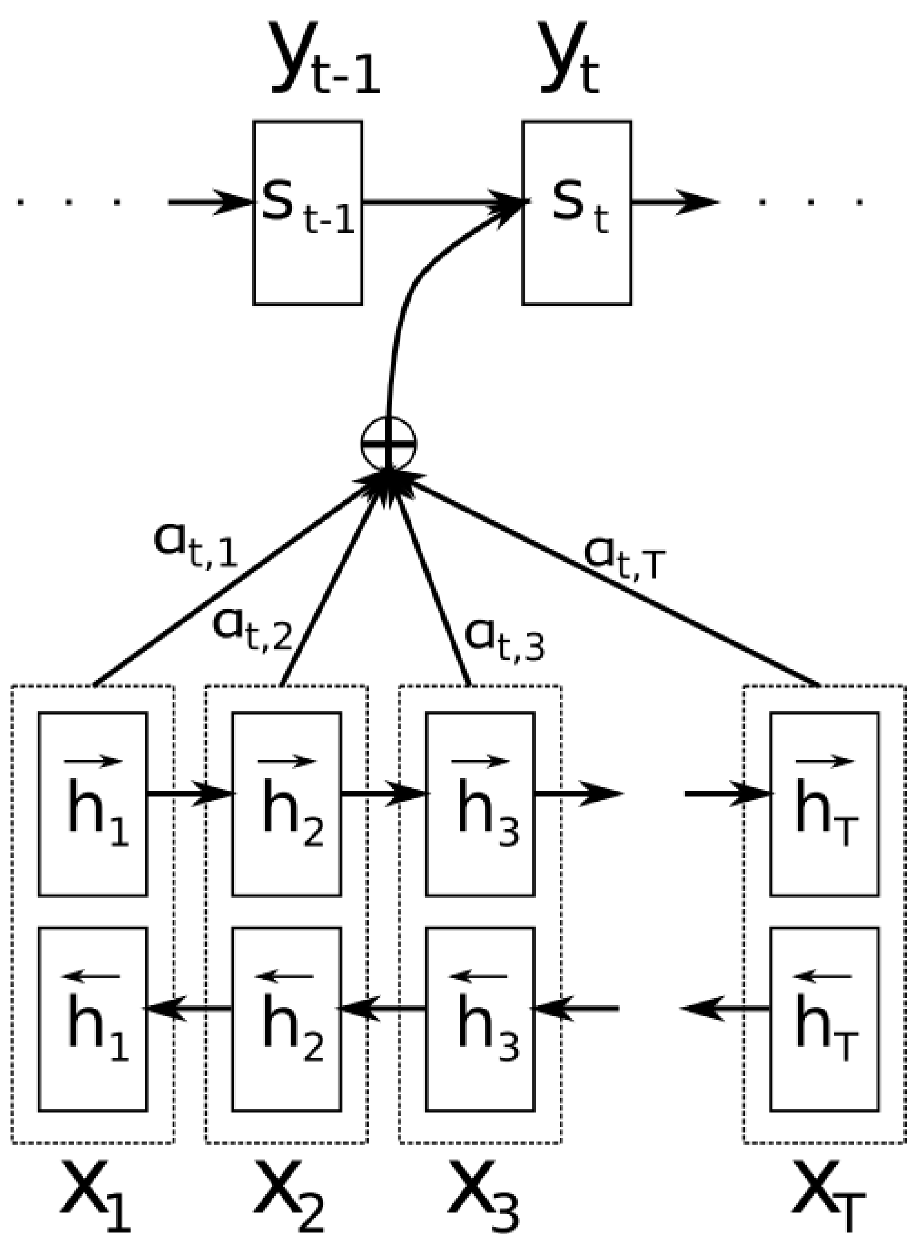
Improving Performance with LSTM

- Consider an input (or intermediate) sequence or image
- Consider an upper level representation, which can choose where to look, by assigning a weight or probability to each input position, applied at each position



Softmax over lower locations conditioned on context at lower and higher locations

NMT with Recurrent Nets and Attention Mechanism



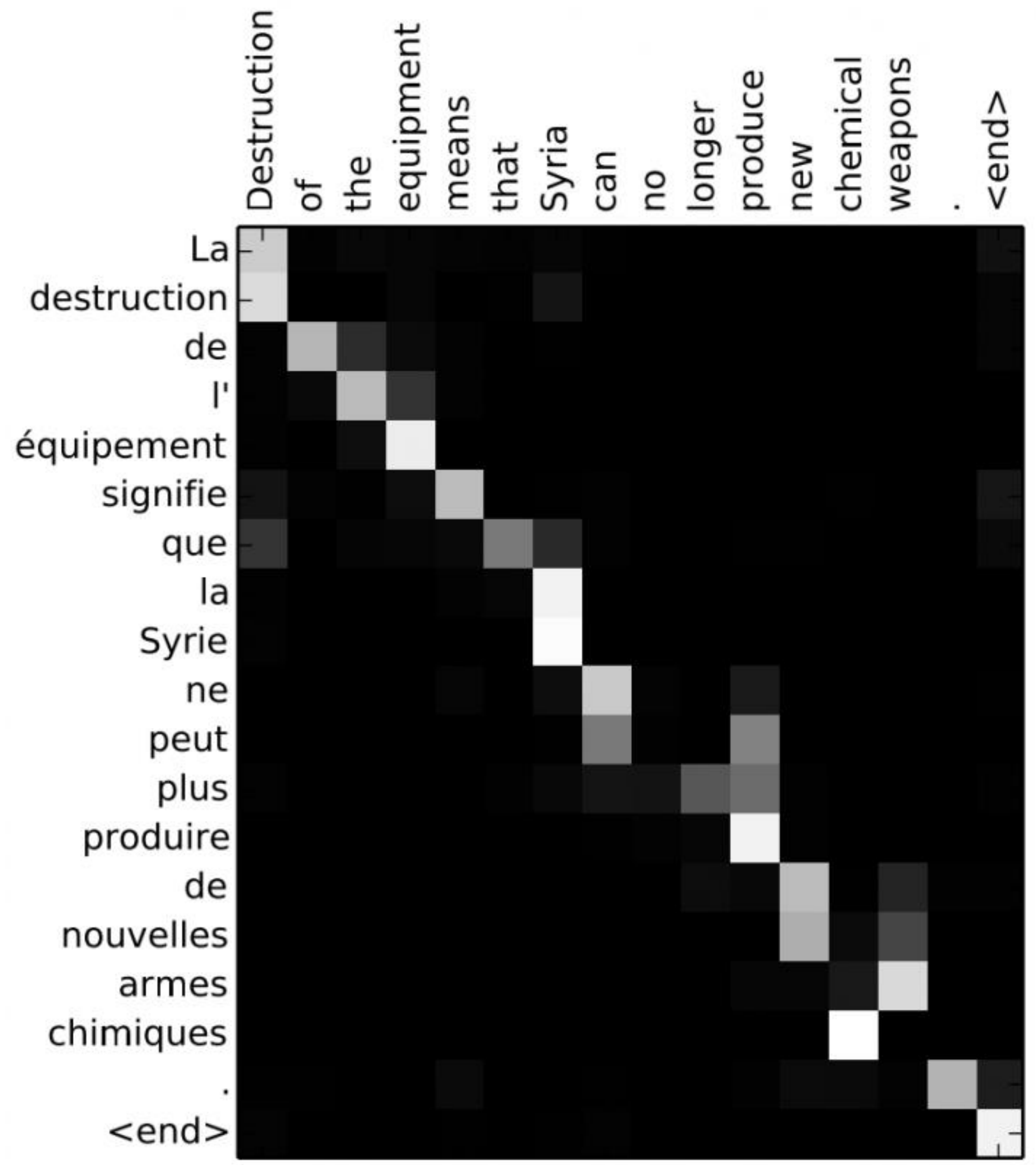
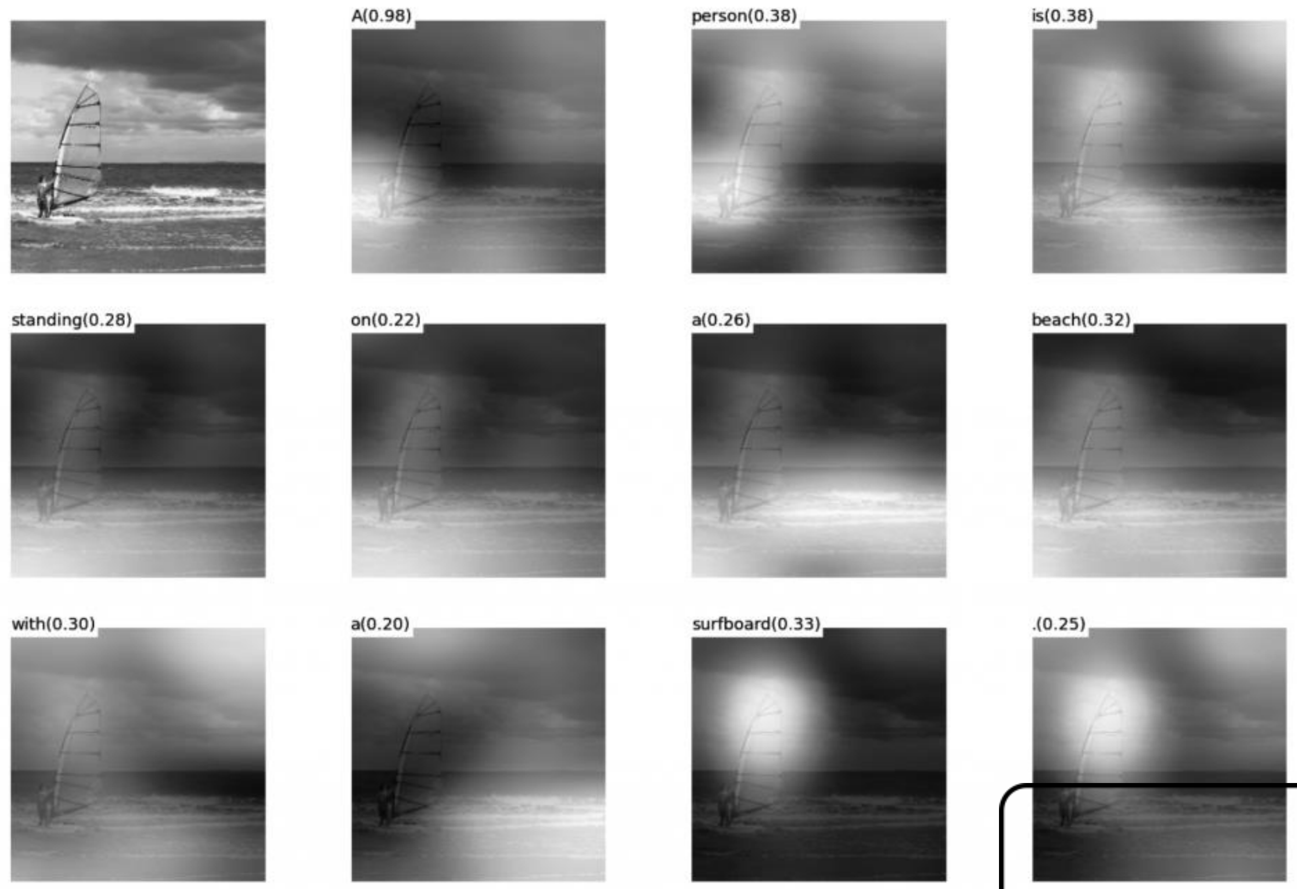
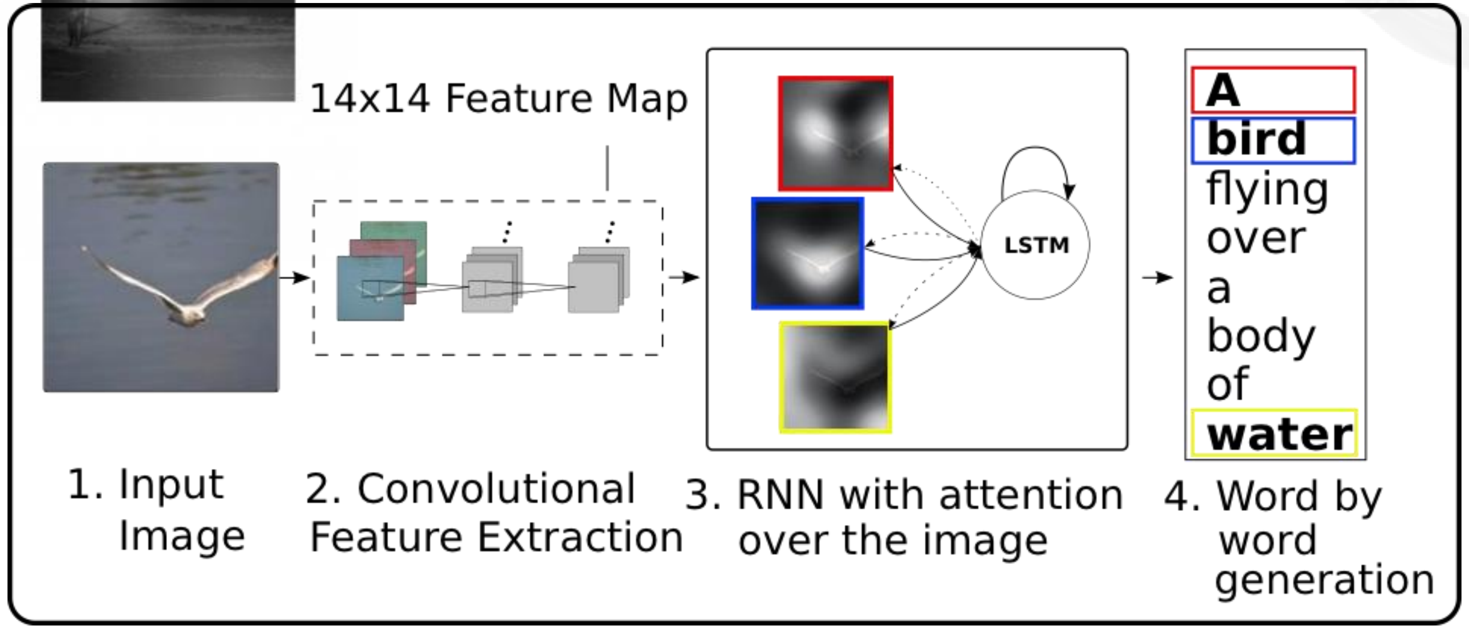


Image-to-Text: Caption Generation with Attention



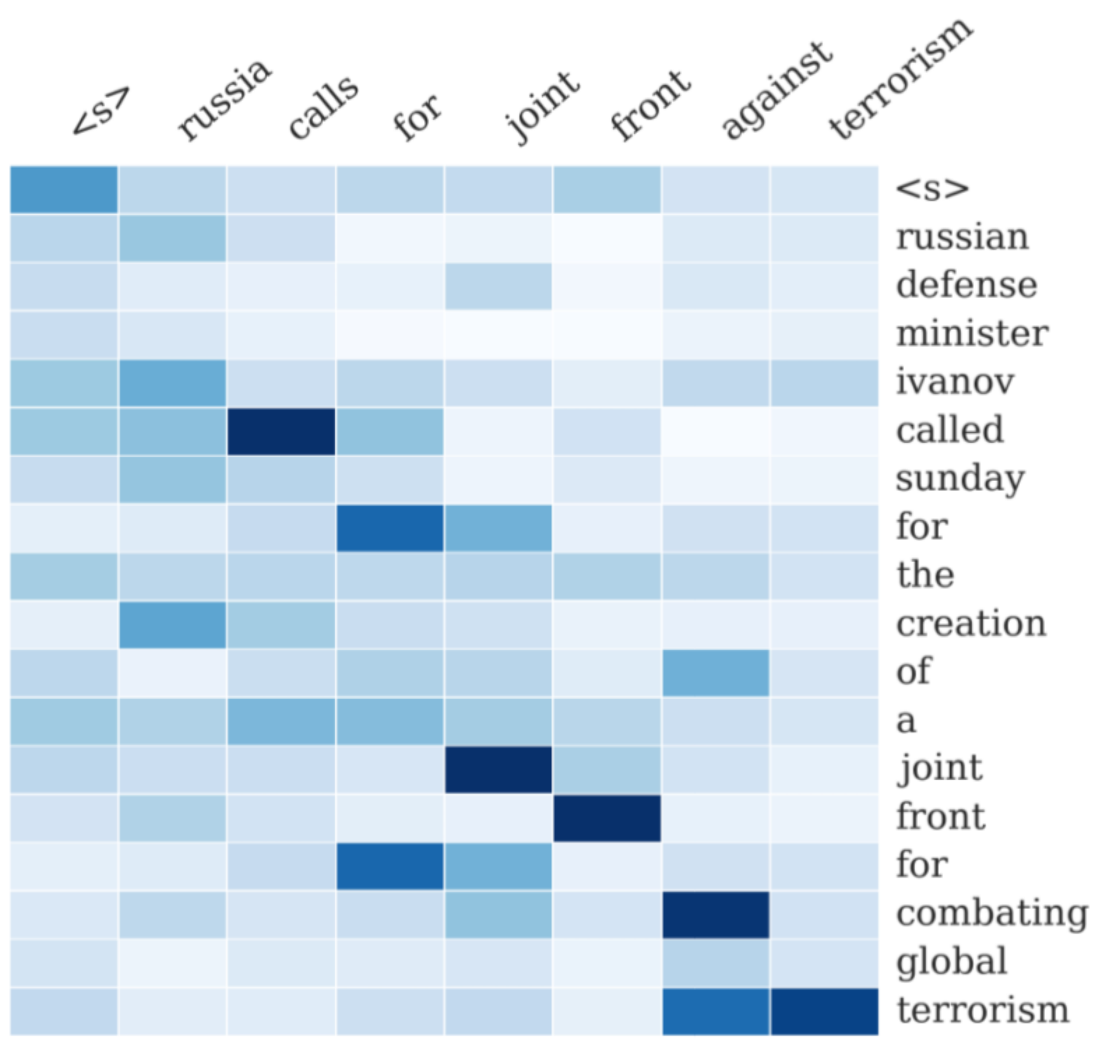
(b) A person is standing on a beach with a surfboard.



14x14 Feature Map
LSTM
A
bird
flying
over
a
body
of
water
1. Input Image
2. Convolutional Feature Extraction
3. RNN with attention over the image
4. Word by word generation

Neural Attention Models

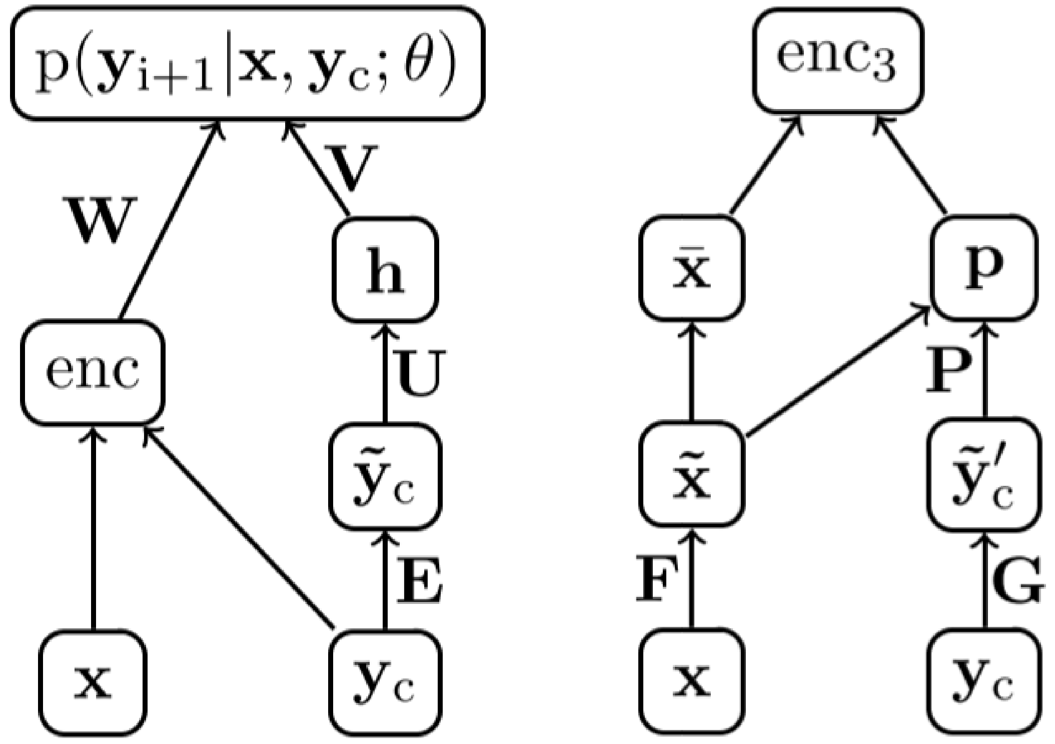
Neural Attention Model for Sentence Summarization



The heatmap represents a soft alignment between the input and the generated summary.

Output of the attention-based summarization system.

Neural Attention Model for Sentence Summarization

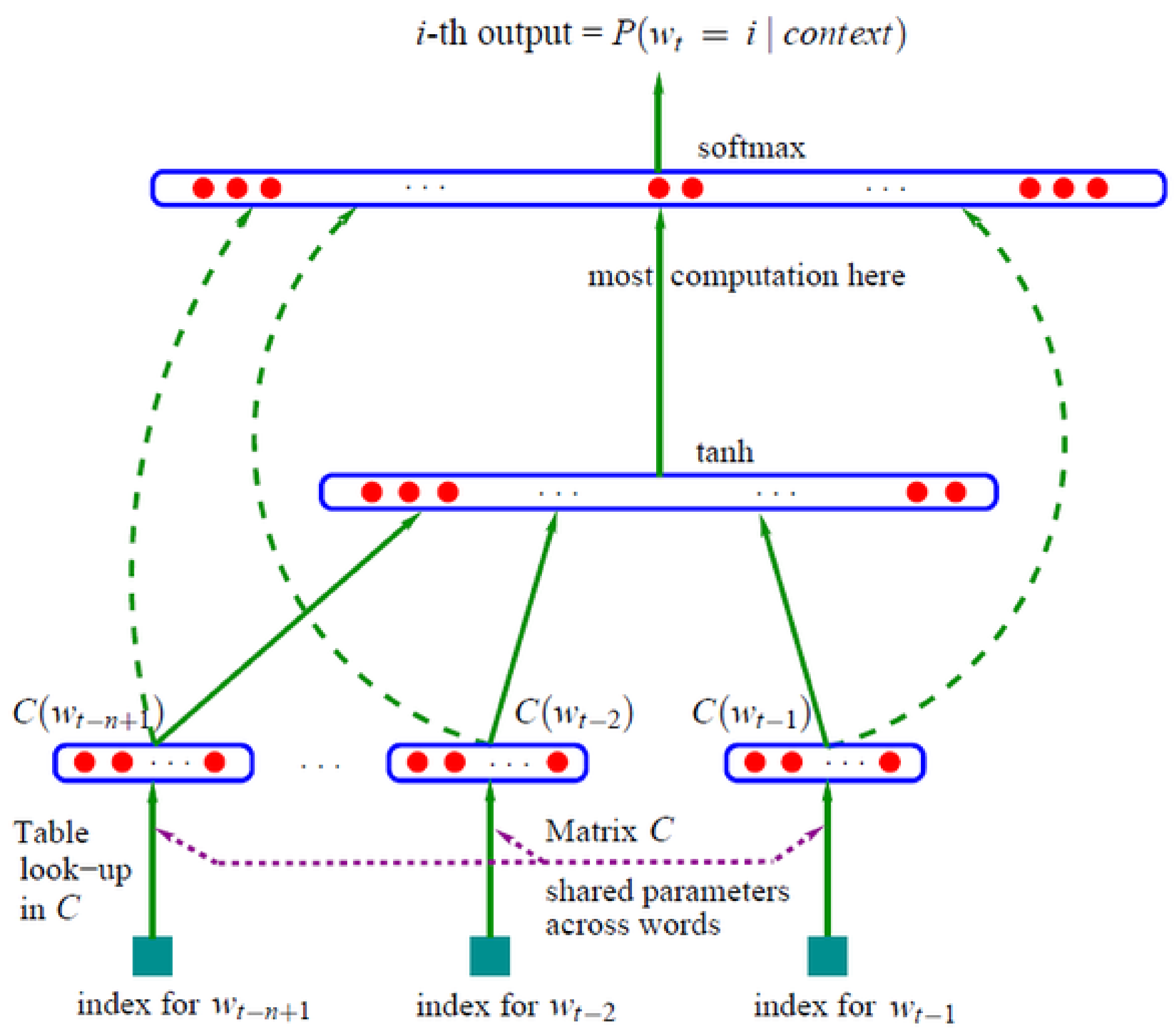


NNLM (Neural Net Language Models) decoder with additional encoder element

Attention-based encoder enc3

Neural Attention Model for Sentence Summarization

Decoder: NNLM



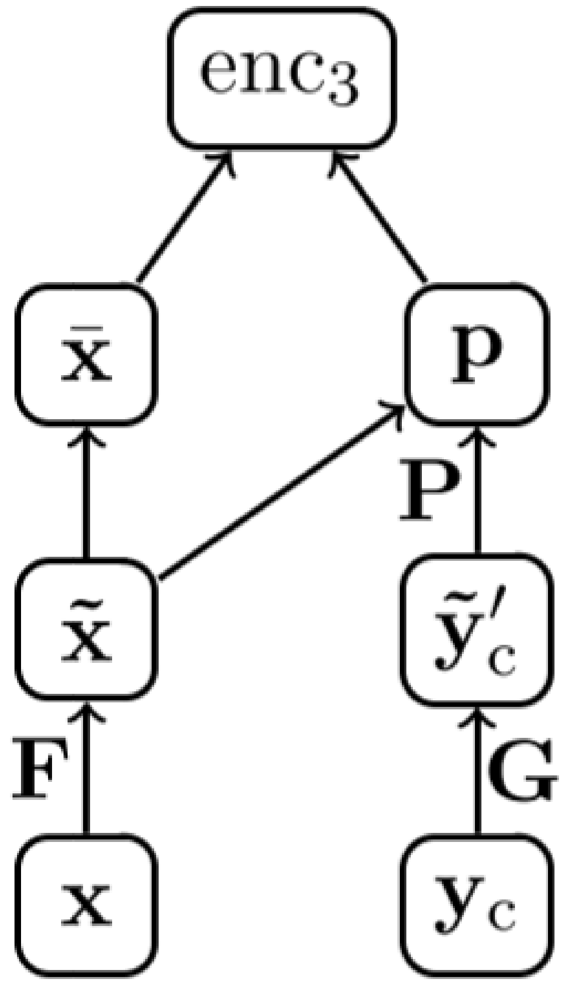
$$
\begin{aligned}
p(\mathbf{y}_{i+1}|\mathbf{y}_c,\mathbf{x};\theta) &\propto \exp(\mathbf{V}\mathbf{h} + \mathbf{W}\text{enc}(\mathbf{x},\mathbf{y}_c)),\\
\tilde{\mathbf{y}}_c &= [\mathbf{E}\mathbf{y}_{i-C+1},\ldots,\mathbf{E}\mathbf{y}_i],\\
\mathbf{h} &= \tanh(\mathbf{U}\tilde{\mathbf{y}}_c).\\
\theta &= (\mathbf{E},\mathbf{U},\mathbf{V},\mathbf{W})
\end{aligned}
$$

$$\begin{tikzcd}
 & \boxed{\mathrm{p}(\mathbf{y}_{i+1}|\mathbf{x},\mathbf{y}_c;\theta)} & \\
\boxed{\text{enc}} \arrow[ur, "\mathbf{W}"] & & \boxed{\mathbf{h}} \arrow[ul, "\mathbf{V}"'] \\
 & & \boxed{\tilde{\mathbf{y}}_c} \arrow[u, "\mathbf{U}"'] \\
\boxed{\mathbf{x}} \arrow[uu] & & \boxed{\mathbf{y}_c} \arrow[u, "\mathbf{E}"'] \arrow[uull]
\end{tikzcd}$$

Neural Attention Model for Sentence Summarization

Encoder: NNLM

$$
\begin{aligned}
\mathrm{enc}_3(\mathbf{x}, \mathbf{y}_c) &= \mathbf{p}^\top \bar{\mathbf{x}}, \\
\mathbf{p} &\propto \exp(\tilde{\mathbf{x}} \mathbf{P} \tilde{\mathbf{y}}'_c), \\
\tilde{\mathbf{x}} &= [\mathbf{F}\mathbf{x}_1, \ldots, \mathbf{F}\mathbf{x}_M], \\
\tilde{\mathbf{y}}'_c &= [\mathbf{G}\mathbf{y}_{i-C+1}, \ldots, \mathbf{G}\mathbf{y}_i], \\
\forall i \quad \bar{\mathbf{x}}_i &= \sum_{q=i-Q}^{i+Q} \tilde{\mathbf{x}}_i / Q.
\end{aligned}
$$



Quora Insincere Questions Classification

Problem Statement: An existential problem for any major website today is how to handle toxic and divisive content. Quora wants to tackle this problem head-on to keep their platform a place where users can feel safe sharing their knowledge with the world. As an approach to the solution, you must create models that identify and flag insincere questions (a question intended to make a statement rather than look for helpful answers.)

Objective: Predict whether a question asked on Quora is sincere or not.

Note: Use the word embeddings provided along with the datasets to accomplish your goal. Also, use tf1.14 for accessing tensorflow.contrib.

Access: Click on the Labs tab on the left side panel of the LMS. Copy or note the username and password that are generated. Click on the Launch Lab button. On the page that appears, enter the username and password in the respective fields, and click Login.

ASSISTED PRACTICE

Key Takeaways

- RNNs have a mechanism that can handle a sequential dataset
- The memory from one state is fed to another state along with the new input in LSTMs
- Attention models use encoder-decoder framework

Knowledge Check

Knowledge Check

1

Why is an RNN (Recurrent Neural Network) used for machine translation, say translating English to French?

a. It can be trained as an unsupervised learning problem

b. It is strictly more powerful than a Convolutional Neural Network (CNN)

c. It is applicable when the input/output is a sequence (e.g., a sequence of words)

d. RNNs represent the recurrent process of Idea->Code->Experiment->Idea->....

Knowledge Check

1

Why is an RNN (Recurrent Neural Network) used for machine translation, say translating English to French?

a. It can be trained as an unsupervised learning problem

b. It is strictly more powerful than a Convolutional Neural Network (CNN)

c. It is applicable when the input/output is a sequence (e.g., a sequence of words)

d. RNNs represent the recurrent process of Idea->Code->Experiment->Idea->....

The correct answer is **c**

RNNs are effective on sequential data.

Knowledge Check

2

What is the probable approach when dealing with “Vanishing Gradient” problem in RNNs?

a. Use modified architectures like LSTM and GRUs

b. Gradient Clipping

c. Dropout

d. All the above

Knowledge Check

2

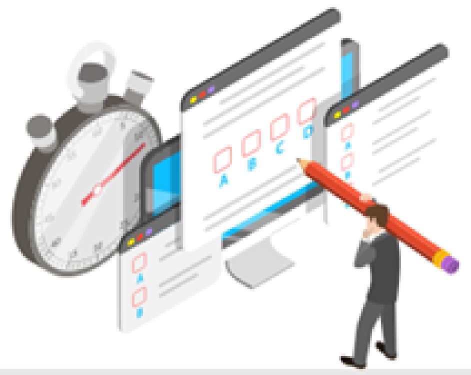
What is the probable approach when dealing with “Vanishing Gradient” problem in RNNs?

a. Use modified architectures like LSTM and GRUs

b. Gradient Clipping

c. Dropout

d. All the above



The correct answer is **a**

LSTMs and GRUs avoid vanishing gradient problem by incorporating gates within RNNs such that only relevant information is passed forward.

Stock Price Forecasting

Problem Statement: It's hard not to think of the stock market as a person. It has moods that can turn from irritable to euphoric. Stock price prediction is of great use for the investors. They constantly review past pricing history and use it to influence their future investment decisions. Considering LSTMs as very powerful networks in sequence prediction, build a deep learning model to predict the future behavior of stock prices.

Objective: Use LSTM for forecasting stock data.

Note: Use the **NSE-TATAGLOBAL.csv** to train your model and perform the testing on **tatatest.csv** file.

Access: Click on the Labs tab on the left side panel of the LMS. Copy or note the username and password that are generated. Click on the Launch Lab button. On the page that appears, enter the username and password in the respective fields, and click Login.

Thank You