

Recurrent Neural Network

Agenda

Review

Sequential data

Text and embedding

Recurrent neural network (RNN)

Applications

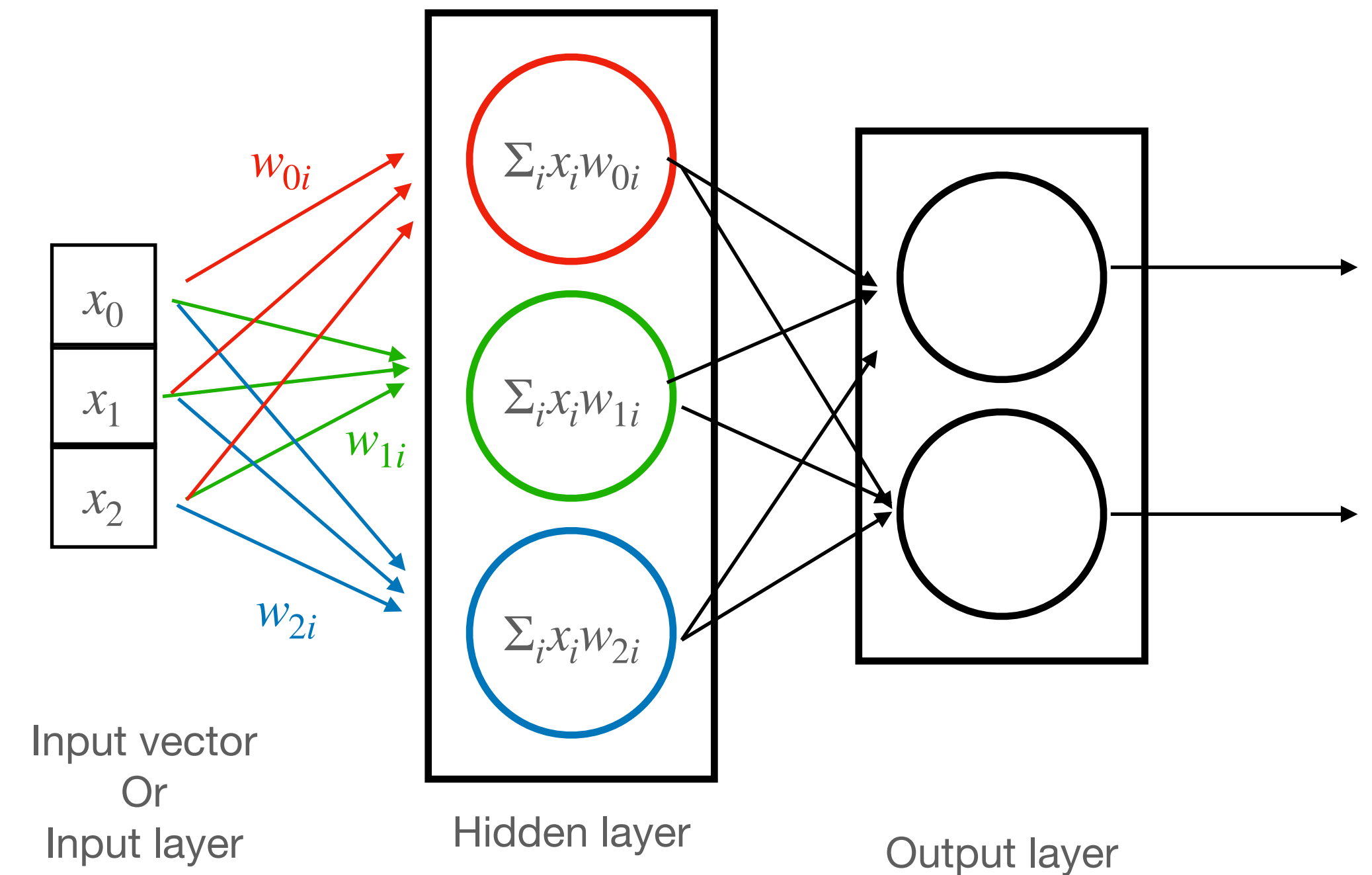
Fancy RNN

Summary

Review

Neural network

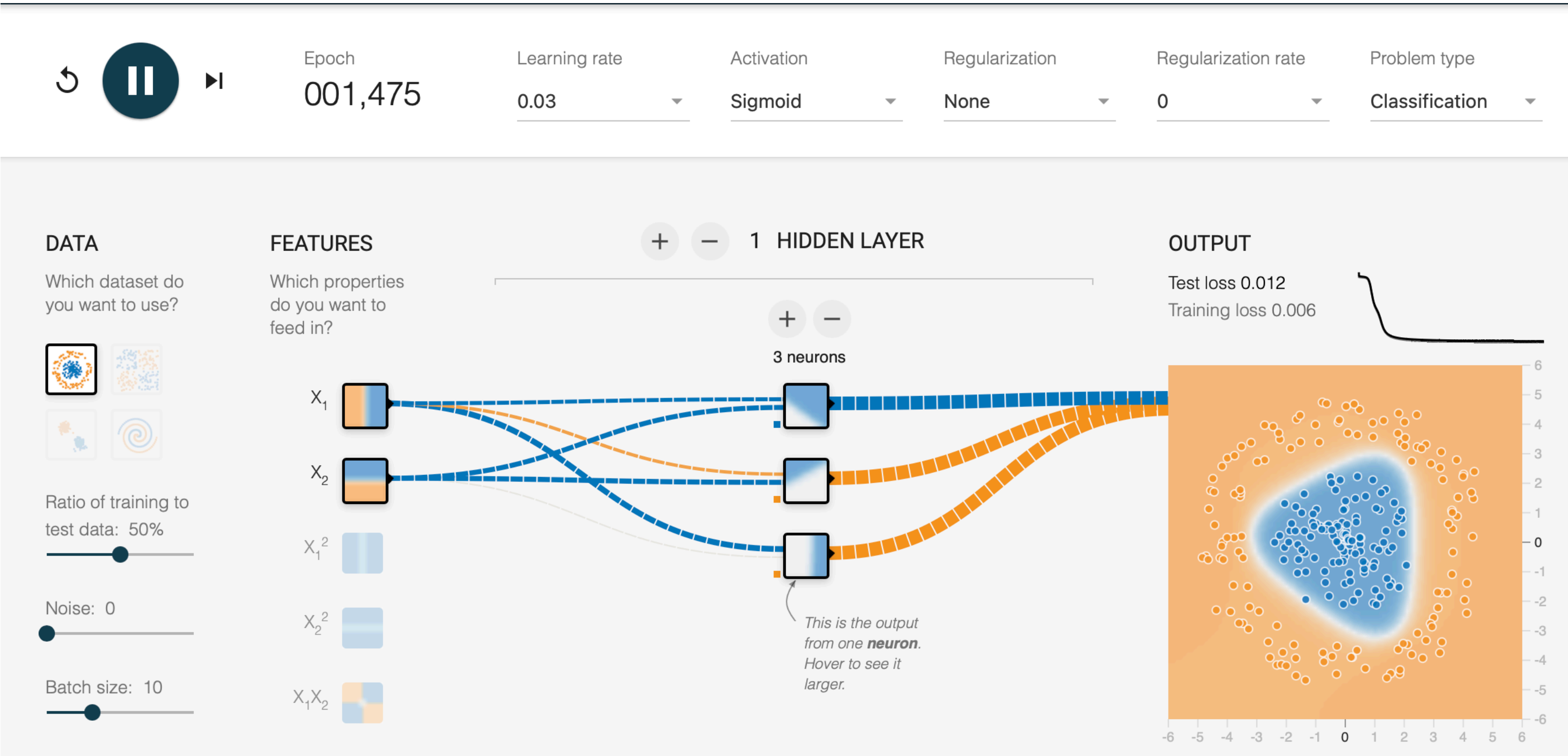
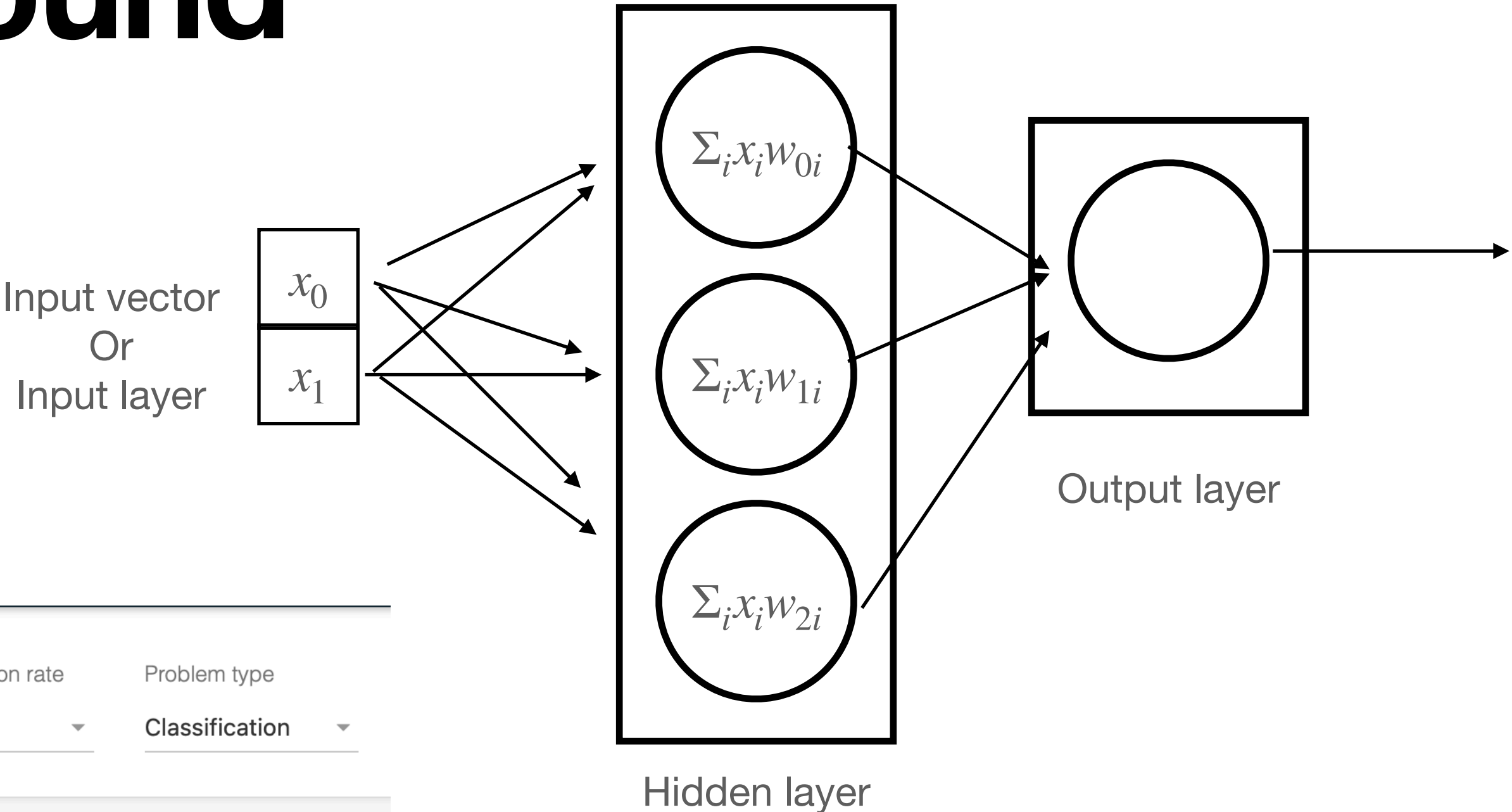
- **Layer**: multiple neurons
- **Multiple** layers: connecting many layers
- Neurons in each layers are connected
- Also called FC (fully connected) layers or MLP (multi-layer perceptron)



$$\begin{bmatrix} w_{00}x_0 + w_{01}x_1 + w_{02}x_2 \\ w_{10}x_0 + w_{11}x_1 + w_{12}x_2 \\ w_{20}x_0 + w_{21}x_1 + w_{22}x_2 \end{bmatrix} = \begin{bmatrix} w_{00} & w_{01} & w_{02} \\ w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$$

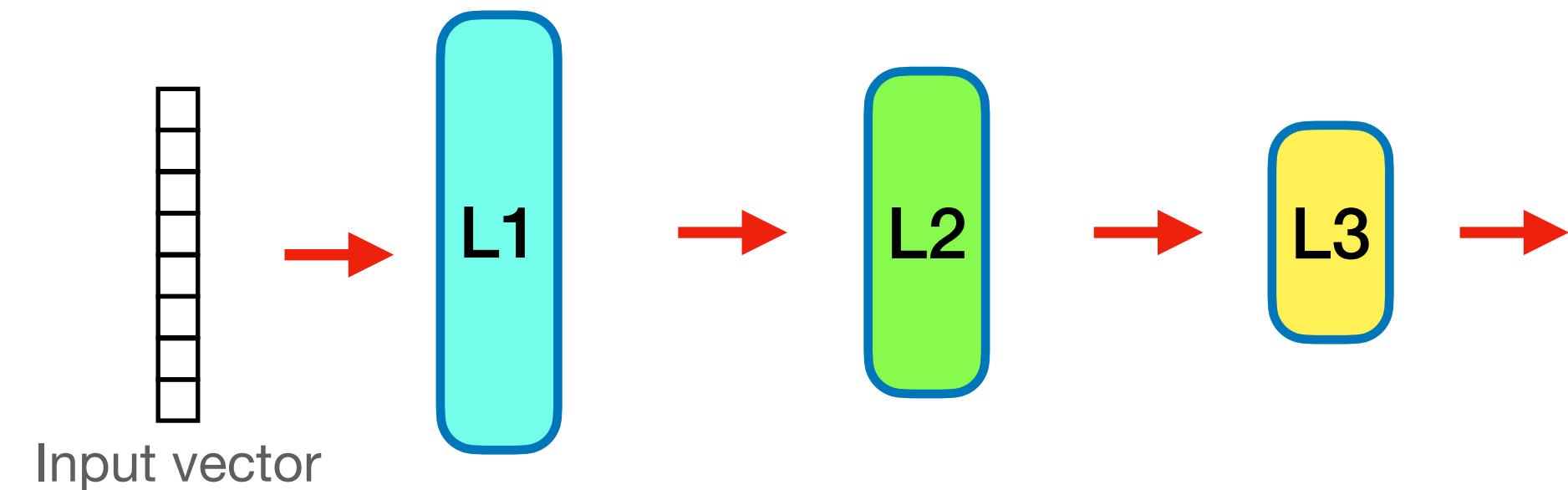
Neural network playground

- Great way to gain intuition
- <http://playground.tensorflow.org>



Simple neural network example

- Simple 3 layers neural network
- In **each layer**:
 - Input: vector
 - Output: vector
 - Learned parameters: matrix
 - Operations:
 - Multiply the input vector with weight matrix
 - Apply element-wise non-linear operations: ReLU *



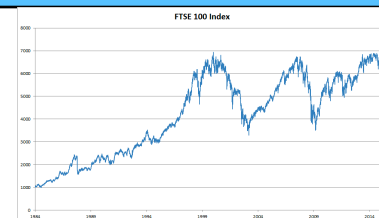




- Neural network training
 - Forward propagation
 - Backward propagation
 - Weight update

*Other types of non-linear ops: sigmoid, tanh

Sequential data

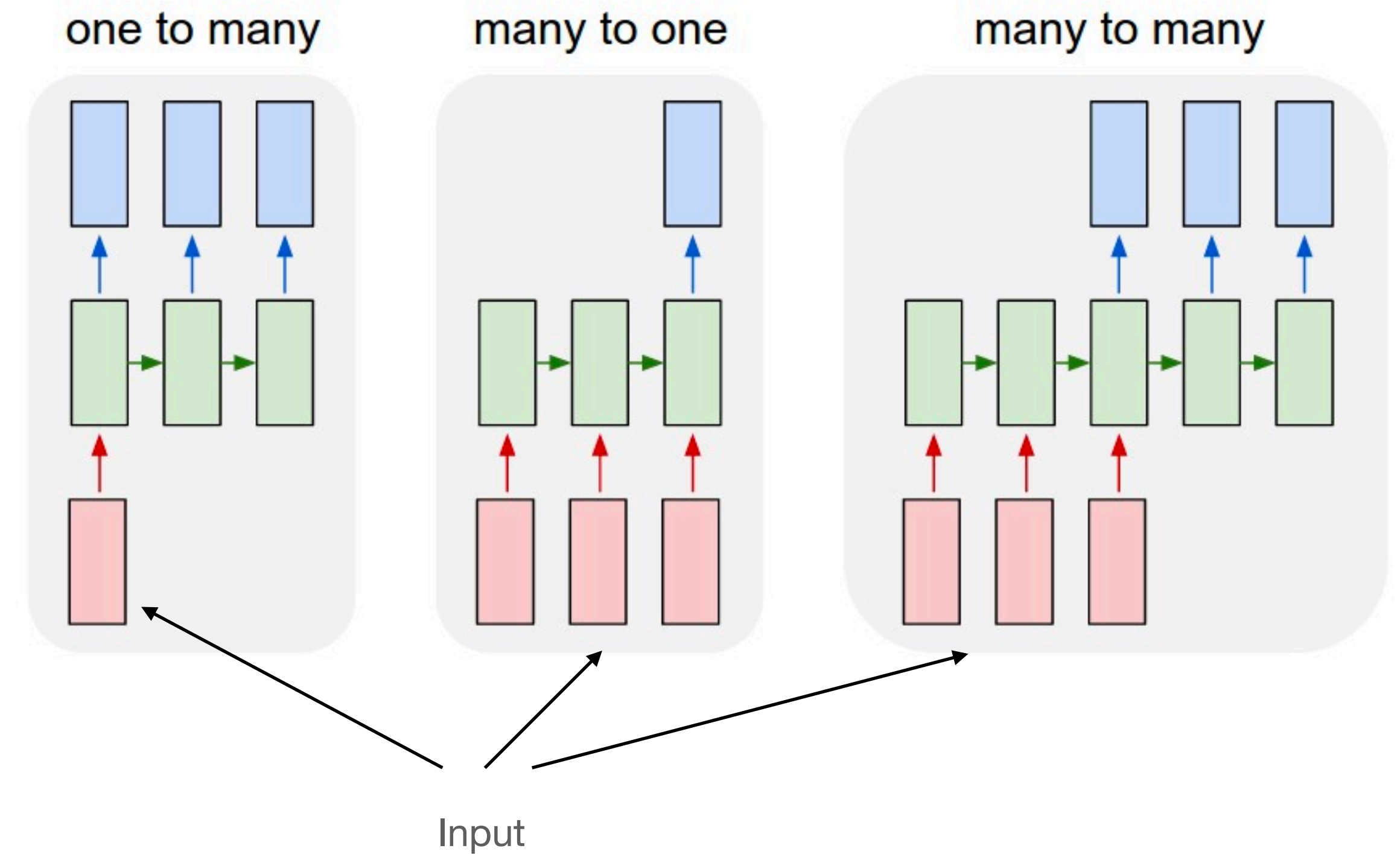
Sequential problems

- There are many applications requires sequential data processing
- Sequential data: data at time t has relationship with previous time steps
- Various types of input and outputs used in sequential applications
- A regular neural network might not use the sequential nature of the input data
- A neural network might not generate sequential outputs

Types	Input	Output
Time series forecasting	Time series 	Next time step prediction
Sentiment analysis	movie review 	Predicted sentiment 
Language translation	English text	日本語 普通话 한국어
Speech recognition		Transcribed text
Document summarization	Document text	Summarized text
Image captioning		Describing scene, planes flying
Question answering	what is the tallest building in the world?	Shanghai Tower

Sequential problems

- Problem setup of sequential problems
- One to many:
 - Image captioning
- Many to one:
 - Document summarization, time series forecast
- Many to many:
 - Machine translation



Text and embedding

Text and one-hot encoding

- Text
- Tokenization:
- Vocabulary: fixed sized dictionary
- Token (word): can be converted to number using vocabulary
- One hot encoding, all 0, except one 1

the time machine by h g wells

['the', 'time', 'machine', 'by', 'h', 'g', 'wells']

[1, 19, 50, 40, 2183, 2184, 400]

Tokens example with vocab size 10 : [1, 5, 9, 3]

One-hot encoding of [1, 5, 9, 3]

[0,	1,	0,	0,	0,	0,	0,	0,	0,	0]
[0,	0,	0,	0,	0,	1,	0,	0,	0,	0]
[0,	0,	0,	0,	0,	0,	0,	0,	0,	1]
[0,	0,	0,	1,	0,	0,	0,	0,	0,	0]

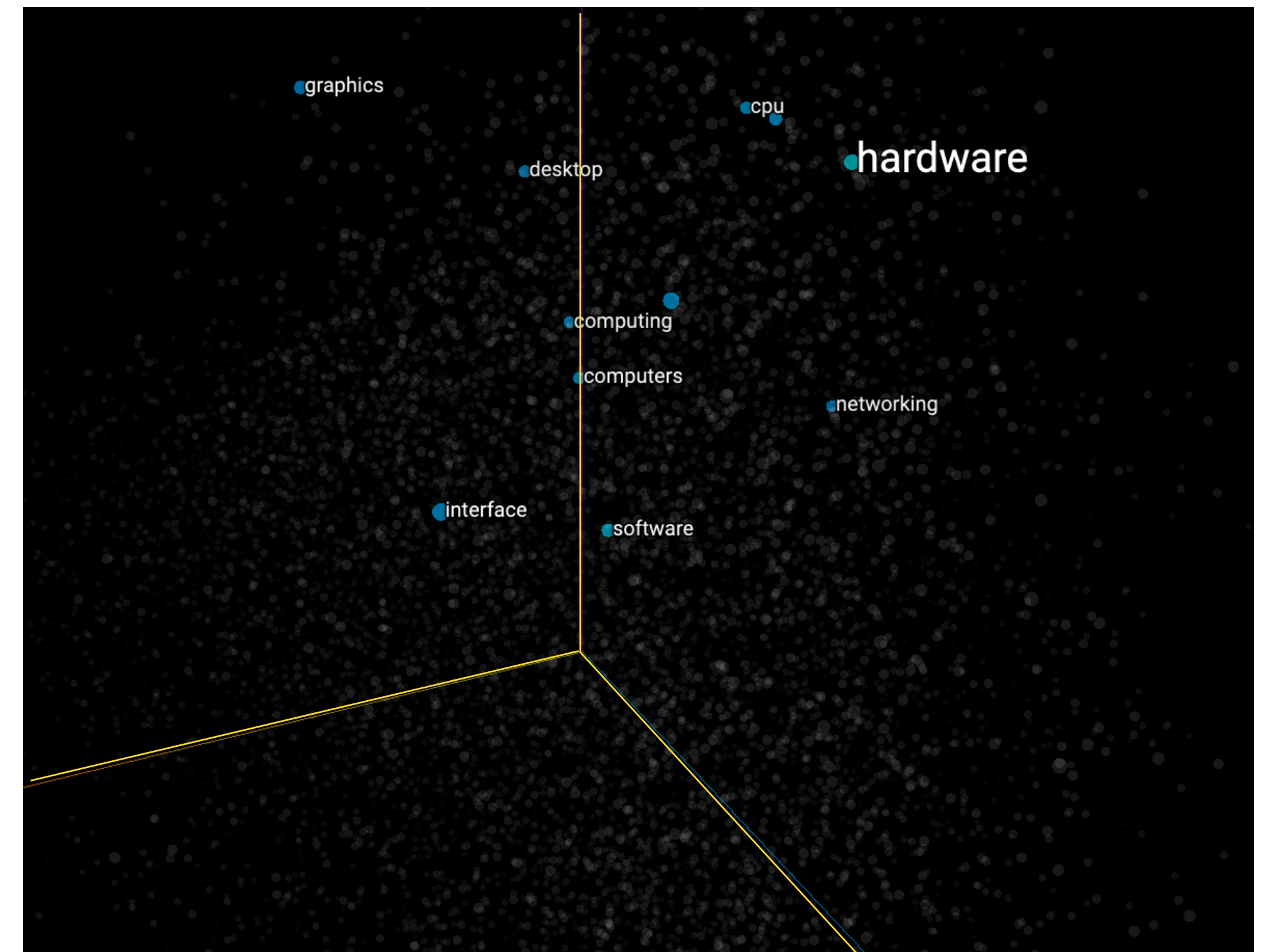
Text and embedding

- **Issue** of one-hot encoding
 - Using this discrete encoding, hard to find the relationship (similar or dissimilar) between tokens
- **Embeddings:**
 - Project one-hot encodings to high dimensional vector space so that tokens with similar meaning to be close to each other
 - Visualization: <https://projector.tensorflow.org>

One hot encoding:

```
Computers : [0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
Hardware  : [0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
```

Embedding example (word2vec)

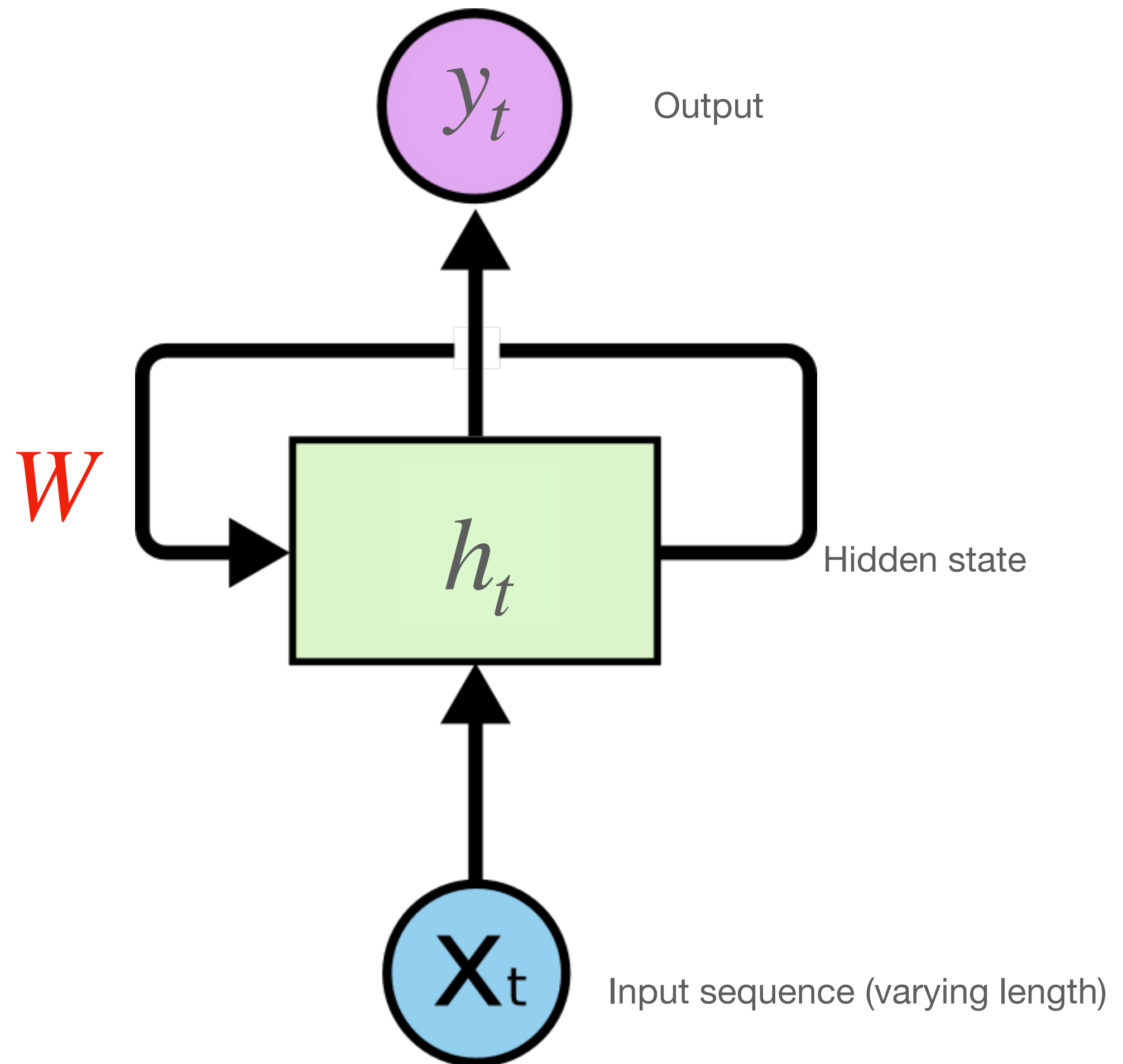


RNN

Recurrent neural network

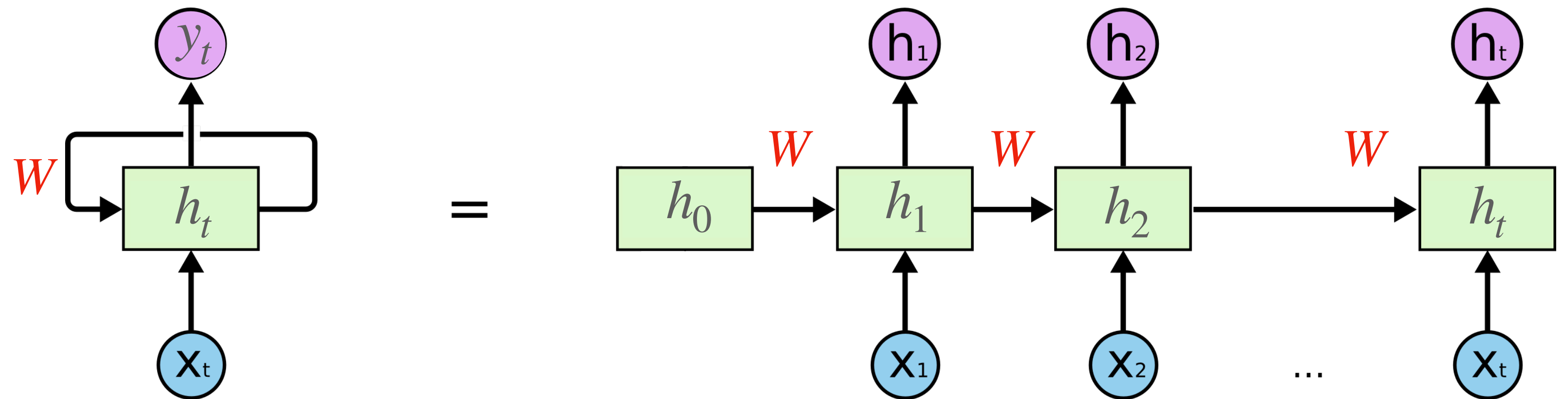
- Neural network has two inputs
 - Input from the current time t
 - Hidden state from time $(t - 1)$
- Allows **stateful** computation
- Any input sequence lengths can be processed
 - This cannot be done with NN
- Apply same W repeatedly

$$y_t, h_t = f(x_t, h_{t-1})$$



RNN

- The rollout representation
- Same model is used



$$y_t, h_t = f(x_t, h_{t-1})$$

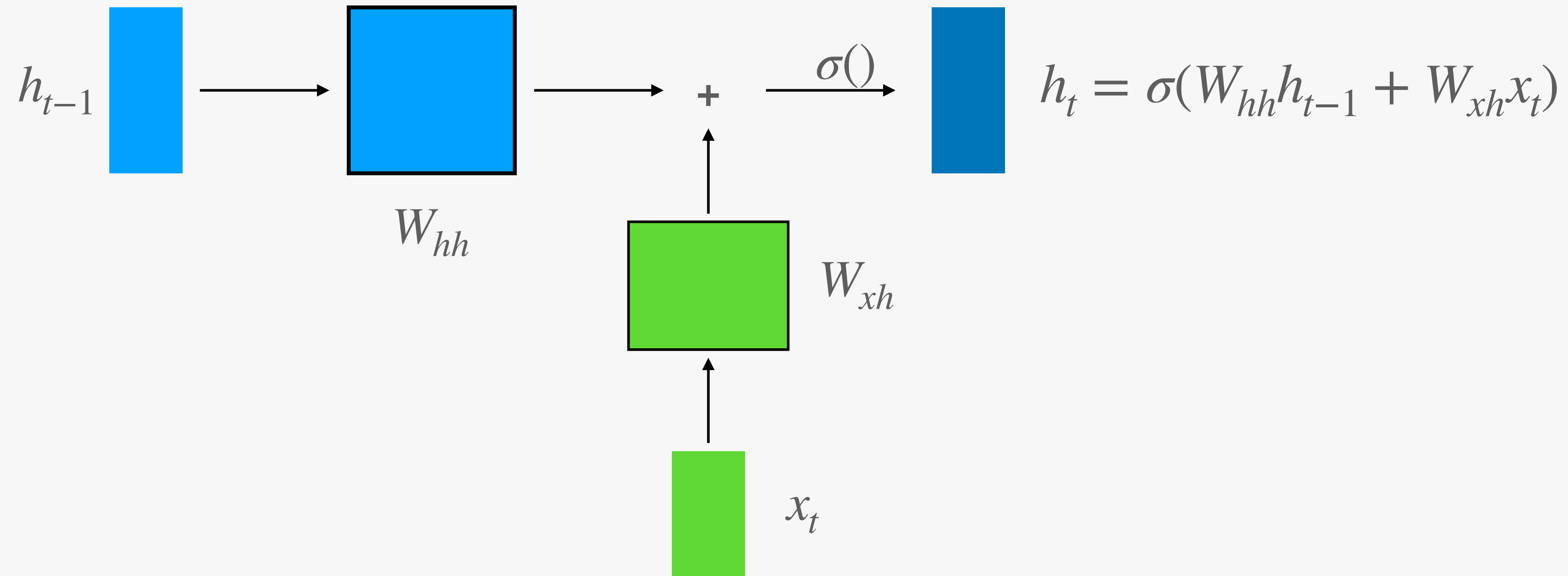
RNN in code

```
class RNN:
    # ...
    def compute_next_h(self, x):
        # Vanilar RNN hidden computation
        self.h = np.tanh(np.dot(self.W_hh, self.h) + np.dot(self.W_xh, x))
        return h
    # ...
    def step(self, x):
        # update the hidden state
        self.h = self.compute_next_h(x)
        # compute the output vector
        y = np.dot(self.W_hy, self.h)
        return y
```

$$h_t = \sigma(W_{hh}h_{t-1} + W_{xh}x_t)$$

RNN in code

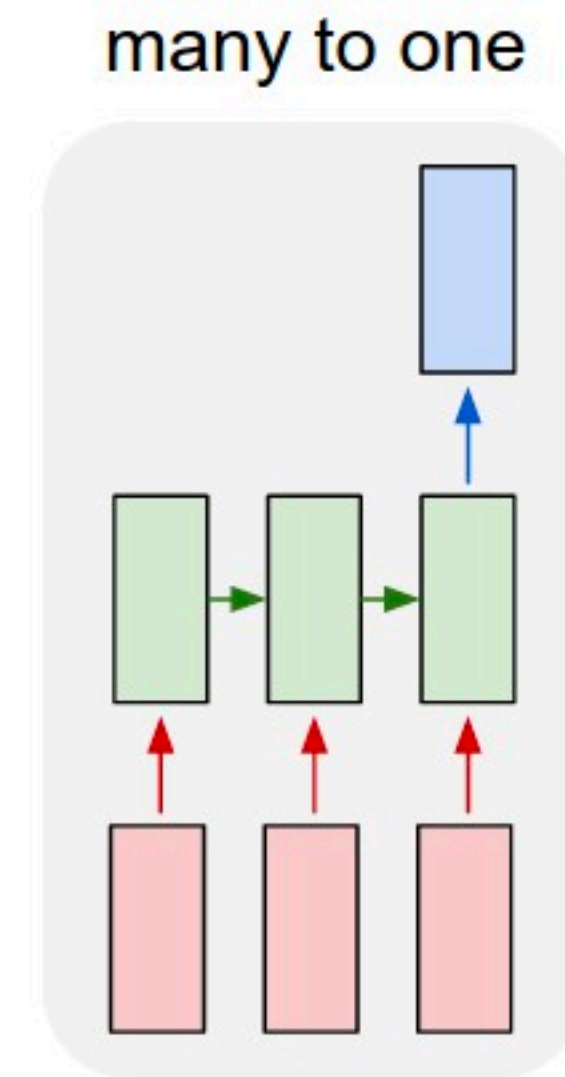
```
class RNN:
```



Applications

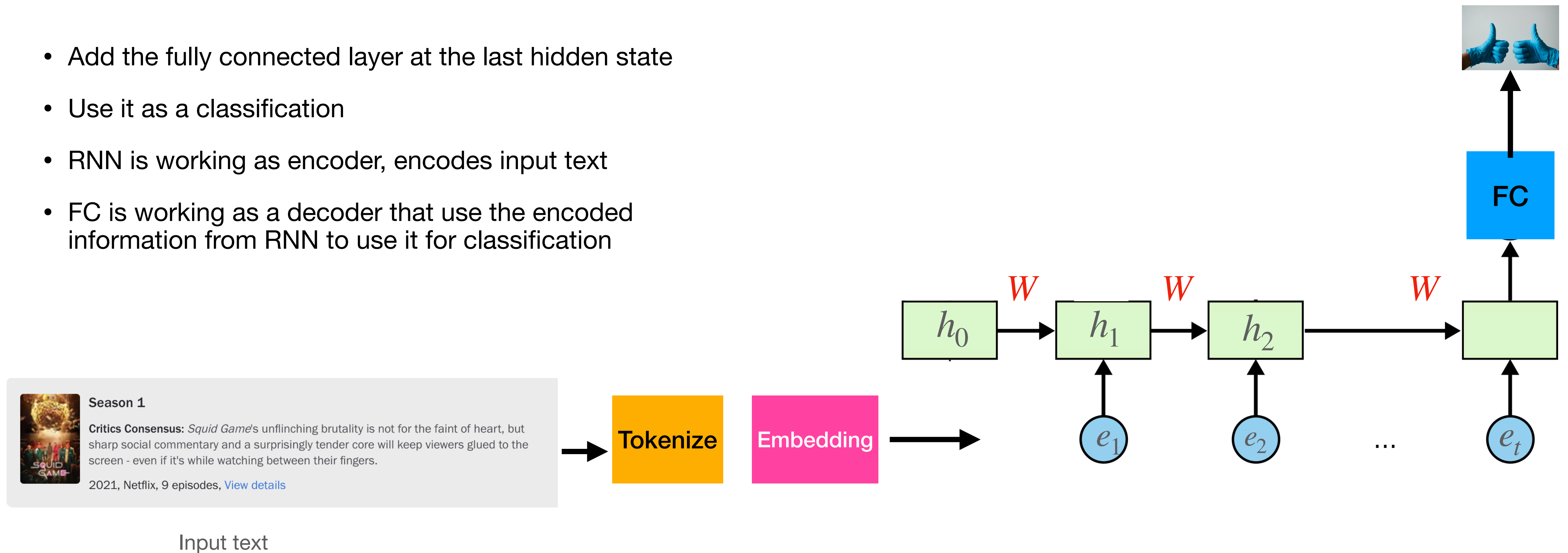
Many to one setup

- Give input sequence, compute a single output
- Example, sentiment analysis



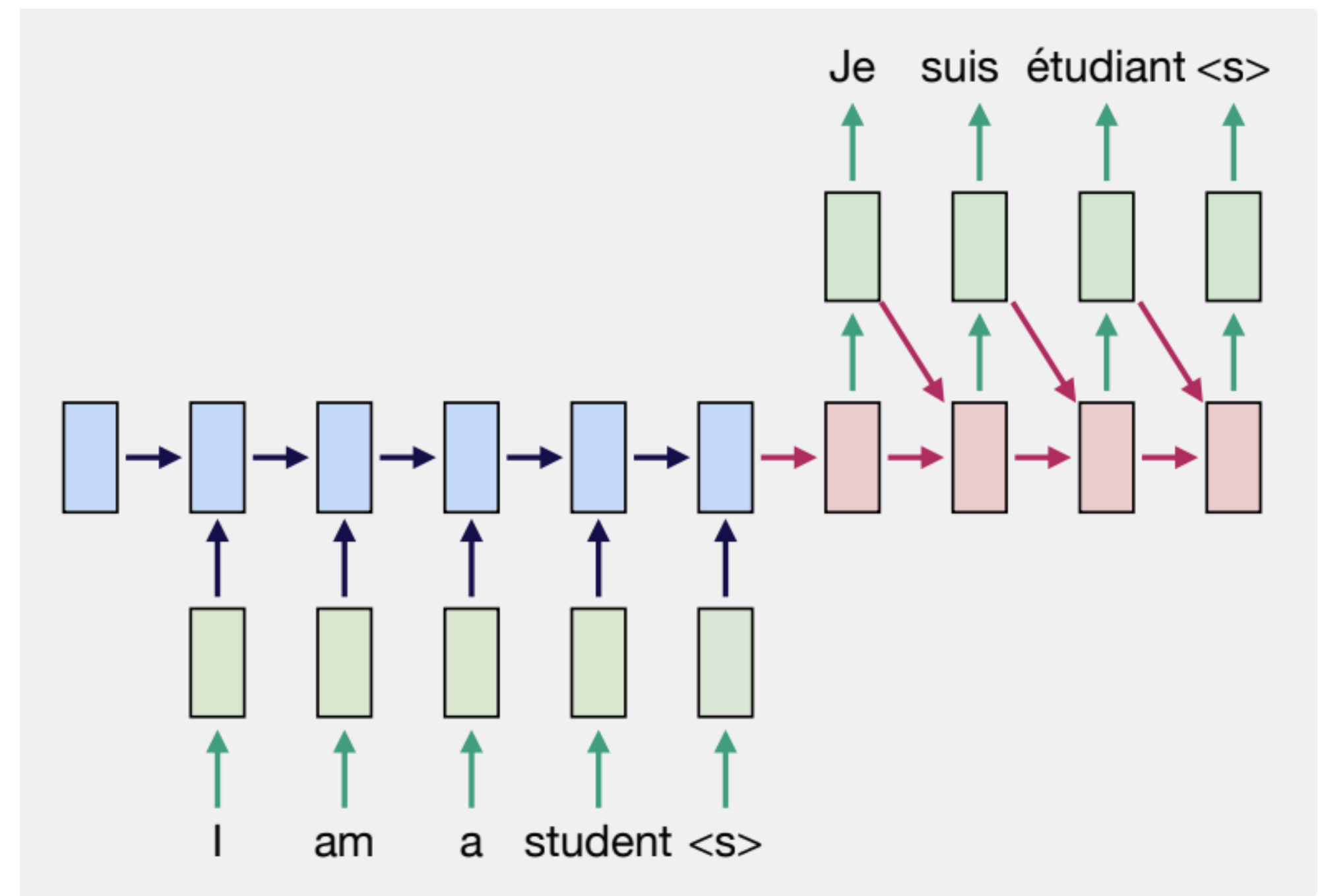
Many to one setup

- Add the fully connected layer at the last hidden state
- Use it as a classification
- RNN is working as encoder, encodes input text
- FC is working as a decoder that use the encoded information from RNN to use it for classification



Many to many setup

- Many to many setup consists of encoder RNN and decoder RNN
- Encoder decoder architecture
 - Also called sequence to sequence (seq2seq)
- Dramatically replace all statistical machine translation method to neural machine translation in 2014



Fancy RNN

Fancy RNNs

- Vanilla RNN has difficulty of capturing long term relations due to vanishing gradient
 - Fancy RNN that allows to capture past state better, LSTM (long short term memory), GRU (Gated recurrent unit)
- Stacked RNN: stacking RNNs to capture more complex information
- Bi-directional RNN: to capture in both forward and backward directional relations

Summary

Summary

- RNN can capture sequential relations from input data
- There are several configurations to use RNN, many-to-one, many-to-many, etc.
- Text dataset will go through tokenization, and then use embedding to represent high-dim information efficiently
- There are advanced RNN models, LSTM, GRU, and stacked RNN and bi-directional RNN

Credits

The following materials are used for this slide.

- “Recurrent Neural Networks” section of Dive into Deep Learning
- "RNNs" section of Full Stack Deep Learning
- Understanding LSTM Networks

References

- Neural network
- Language Models and Recurrent Neural Networks