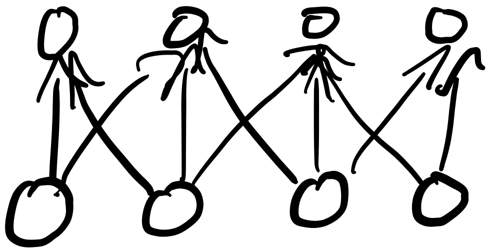
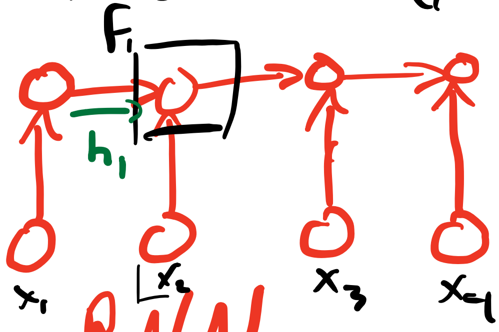
Recurrent Neural Networks

Recurrent Neural Network (RNN)

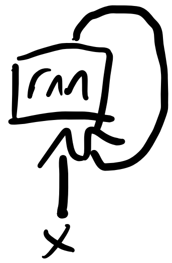


CNN



RNN

$$h_2 = f(x_2, h_1)$$

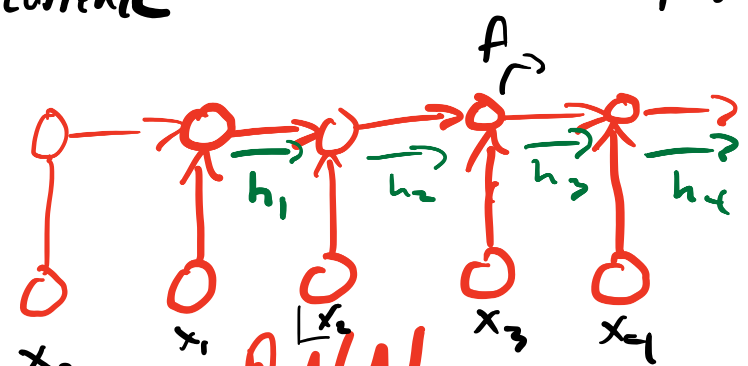


In general

$$h_t = f(x_t, h_{t-1}) = \sigma(x_t^T W^T + h_{t-1} U + b)$$

recurrence

2 parameters



RNN

$$h_4 = f(x_4, f(x_3, f(x_2, f(x_1, \ldots))))$$

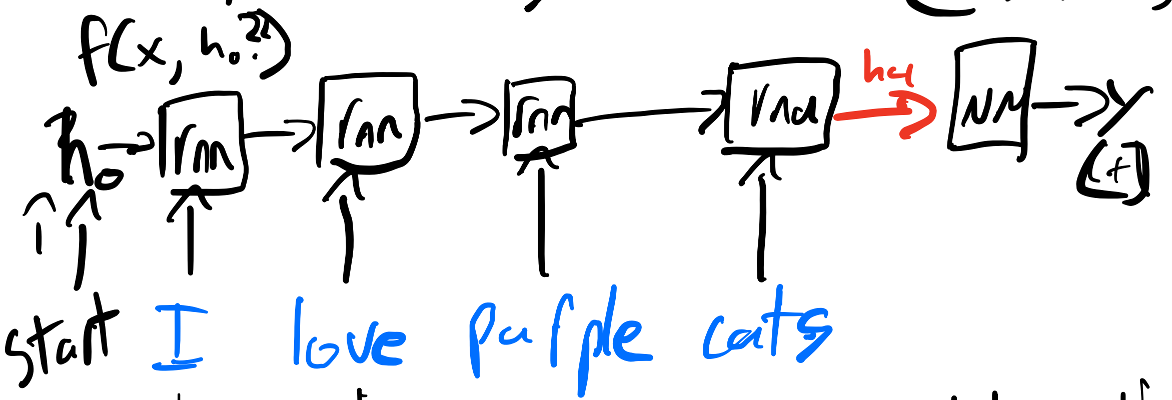
depend all inputs time ≤ 4

Many to one

Input: Variable length sequence

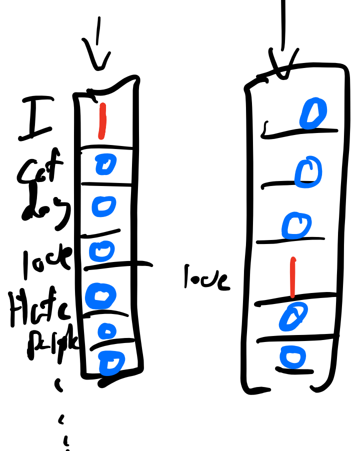
ex. I love purple cats

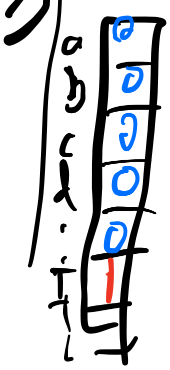
Output: single class $\begin{cases} (+) & \text{positive} \\ (-) & \text{negative} \end{cases}$



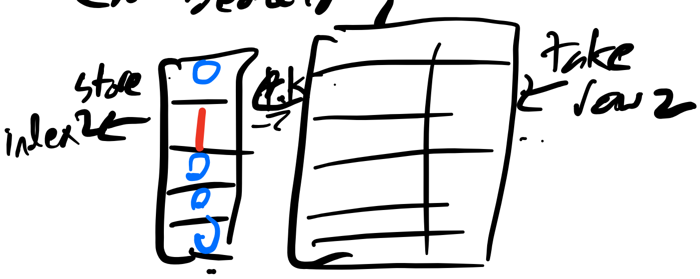
one hot encoding (words)

one-hot encoding (letters)





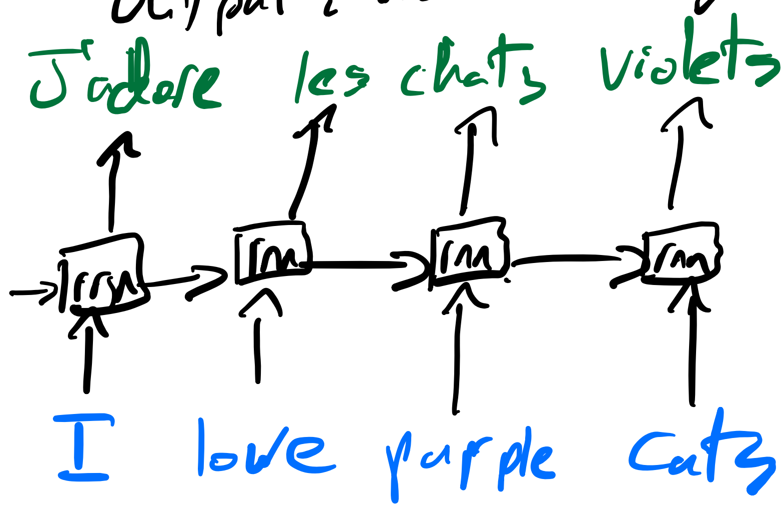
embedding



Many to Many

Input: variable length sequence

Output: Variable length sequence



: Translation x

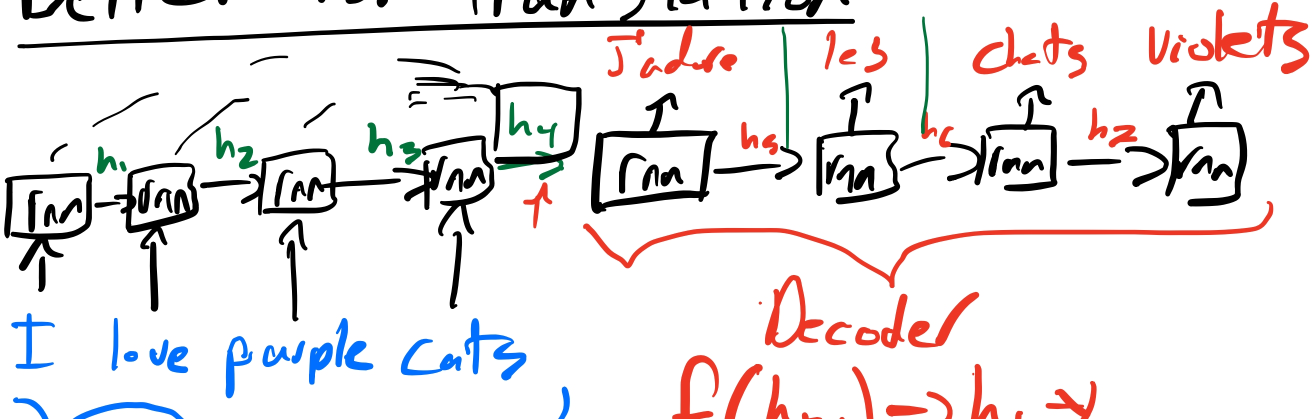
output may be in different or have different len.

Part of speech tagging

RNN w/ output

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

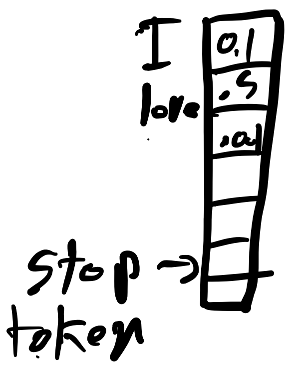
Better for translation



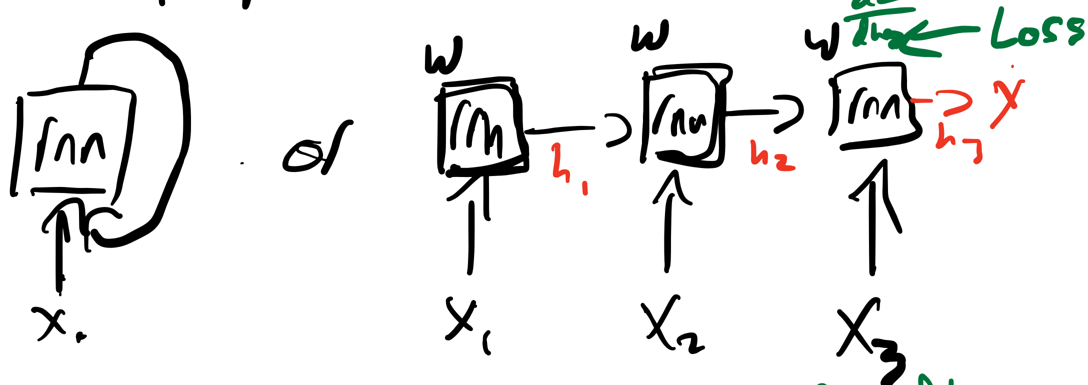
Decoder $f(h_{t-1}) \to h_t, y_t$

Encoder $\to f(x_t, h_{t-1}) \to h_t$

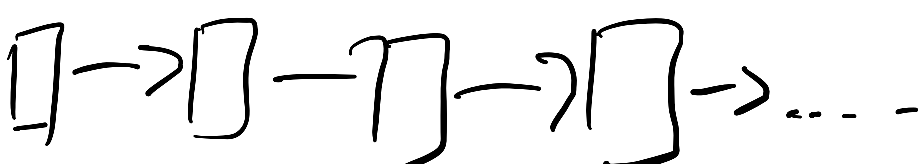
$$\underbrace{P(y_t \mid h_t) = \text{softmax}(W_{out} h_t)}_{\text{Multinomial logistic prediction}}$$



Backpropagation (reverse-mode AD)

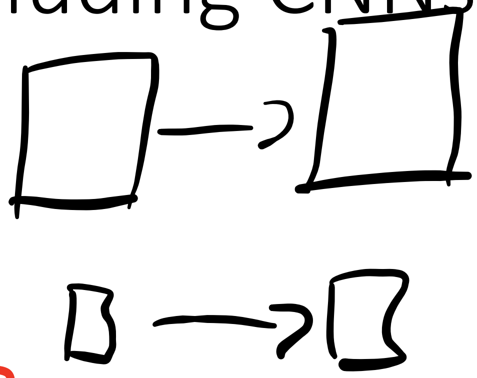


$$\frac{dL}{dh_2} = \frac{dL}{dh_3} \cdot \frac{dh_3}{dh_2}$$



Many Layers → Exploding gradients

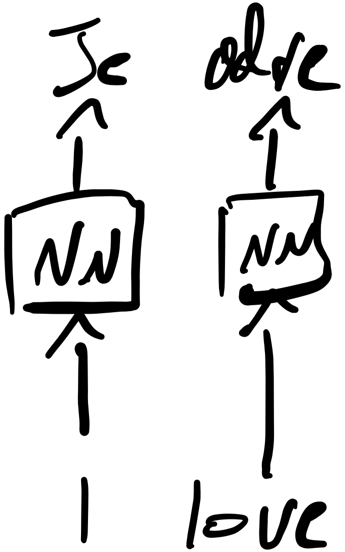
Conventional Neural Networks (including CNNs)



- Input: fixed sized tensor
  - Though the batch size can be any value due to broadcasting
- Output: fixed sized tensor
  - Though the batch size can be any value due to broadcasting
- Functionally deterministic (always produce the same output for a given input)
  - When might you want different outputs on the same input?

Motivating Example: Text Translation

- Input: I love purple cats. Cats are neat. ⇐ Any sized sequence
- Output: J'adore les chats violets. Les chats sont soignés.



Standard

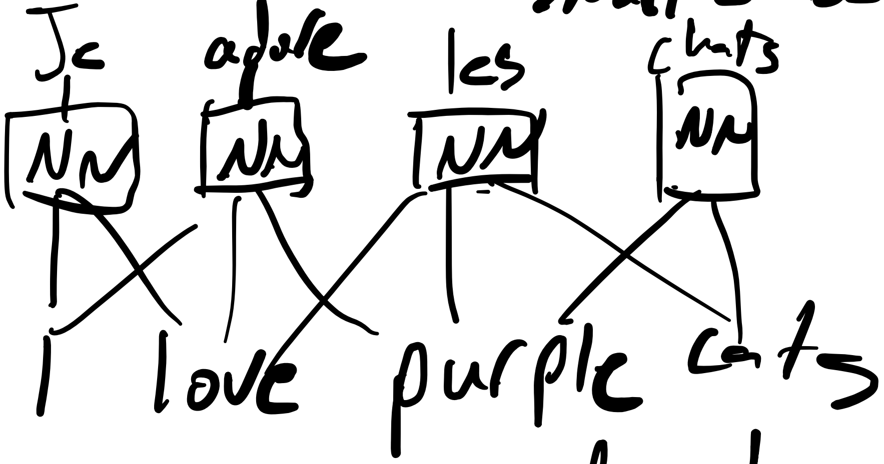
$f(x) = \sigma(x^T W^T + b)$

size?

purple cats

convolutional

small window



output size depends on input

Motivating Example: Text Translation

- Input: I love purple cats. Cats are neat.
- Output: J'adore les chats violets. Les chats sont soignés.

Parts-Of-Speech Example

You all must submit sentences for the dataset.

```
"I is a teeth"
```

How do we pass this into a neural network?

Parts-Of-Speech Example

You all must submit sentences for the dataset.

`"I is a teeth"`

How do we pass this into a neural network?

Recurrent Neural Networks

Operate over sequences (data with temporal dependencies).

Recurrent Neural Networks

Operate over sequences (data with temporal dependencies).

Recurrent Neural Networks

Operate over sequences (data with temporal dependencies).

Motivating Example: Text Translation

- Input: I love purple cats. Cats are neat.
- Output: J'adore les chats violets. Les chats sont soignés.

Backpropgation Through Time (BPTT)

Linear vs (Elman) Recurrent Neurons

```python
class Neuron(torch.Module):
    def __init__(self, input_size, output_size):
        self.W = torch.randn(output_size, input_size) * 0.01
        self.b = torch.randn(output_size, 1) * 0.01


    def forward(self, X):
        linear = X @ self.W.T + self.b.T
        return F.sigmoid(linear)
```

- Input shape: (N, input_size)
- Output shape: (N, output_size)

```python
class RecurrentNeuron():
    def __init__(self, input_size, output_size):
        self.Wx = torch.randn(output_size, input_size) * 0.01
        self.Wh = torch.randn(output_size, output_size) * 0.01
        self.bh = torch.zeros(output_size, 1)
        self.output_size = output_size


    def forward(self, X, state=None):
        L, N, input_size = X.shape
        if not state:
            state = torch.zeros(N, self.output_size)


        output_sequence = []
        for x_t in X:
            state = F.tanh(x_t @ self.Wx + state @ self.Wh + self.bh)
            output_sequence.append(state)


        return torch.tensor(output_sequence), state
```

- Input shape: (L, N, input_size)
- Output shape: (L, N, output_size)

*Untested code.

Processing Natural Language with an NN

Here's one way to convert text into numbers

1. Assign every word a unique number (e.g., 1 .. vocab_size)
2. Assign every part-of-speech a unique number (e.g., 1 .. num_classes)
3. Convert sentences into index tensors (using mapping from step 1)
4. Pass index tensors into an [embedding]() layer (i.e., a simple lookup table)
5. Pass embedding outputs into the recurrent neural network (RNN)
6. Pass the RNN output into a fully-connected (FC) classification network
7. Convert the FC output into a part-of-speech (one-hot)

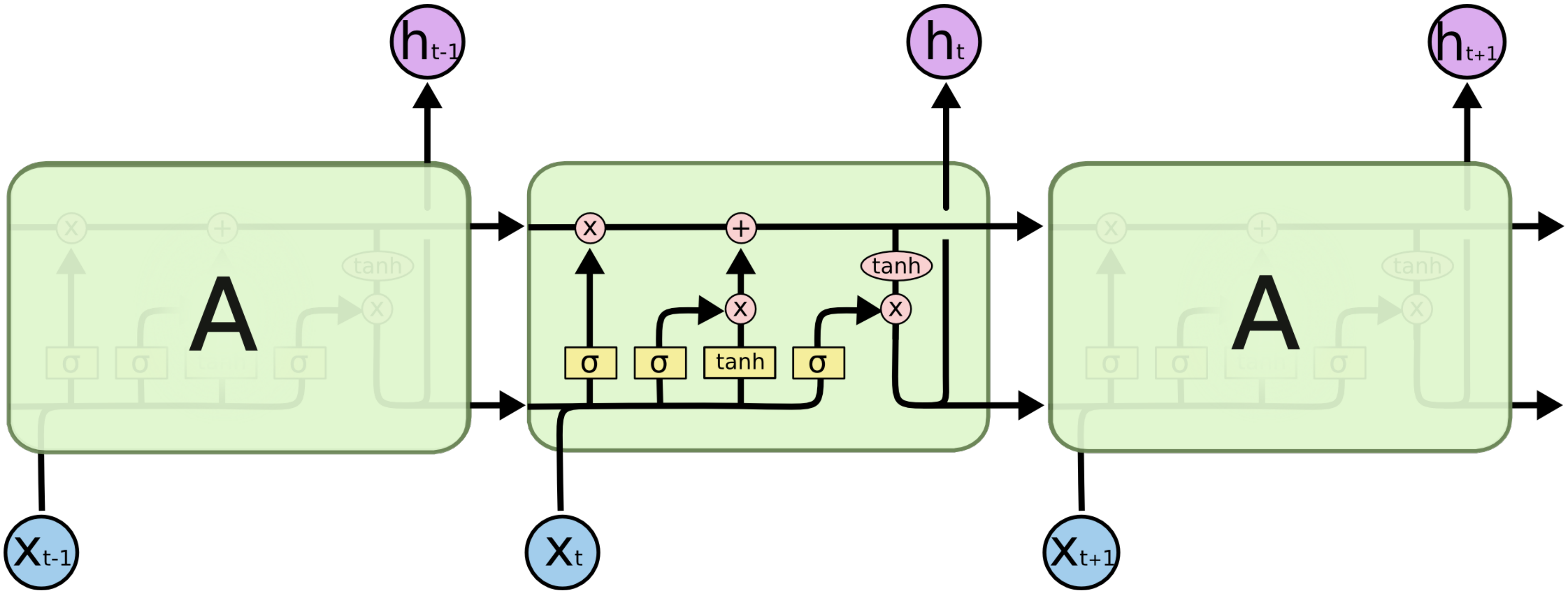
```python
class POS_LSTM(torch.nn.Module):
    """Parts-of-speech LSTM model."""

    def __init__(self, vocab_size, embed_dim, hidden_dim, num_layers, parts_size):
        super().__init__()
        self.embed = torch.nn.Embedding(vocab_size, embed_dim)
        self.lstm = torch.nn.LSTM(embed_dim, hidden_dim, num_layers=num_layers)
        self.linear = torch.nn.Linear(hidden_dim, parts_size)

    def forward(self, X):
        X = self.embed(X)
        X, _ = self.lstm(X.unsqueeze(1))
        return self.linear(X)
```

RNN Paradigms

LSTMs



https://colah.github.io/posts/2015-08-Understanding-LSTMs/

RNNs

Eploding gradients

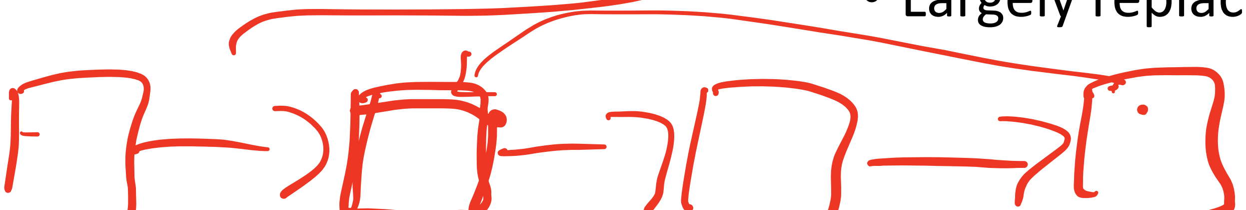
Advantages

- Process varying input length
- Model size remains constant
- Maintains historical information

Input: I love purple cats. Cats are neat.
Output: J'adore les chats violets. Les chats sont soignés.

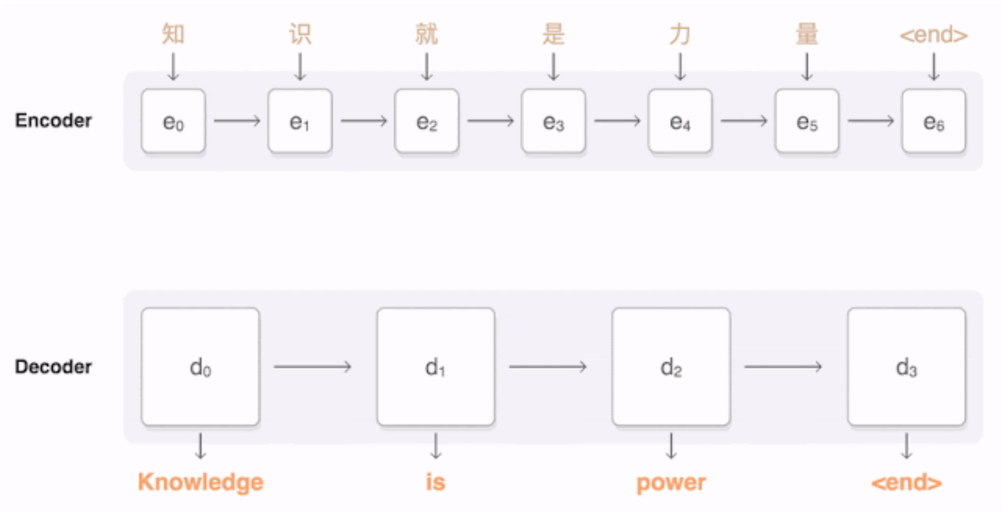
Disadvantages

- Slower to compute
- Poor handling of long-term dependencies
- Does not consider future inputs to produce current state
- Largely replaced by transformers

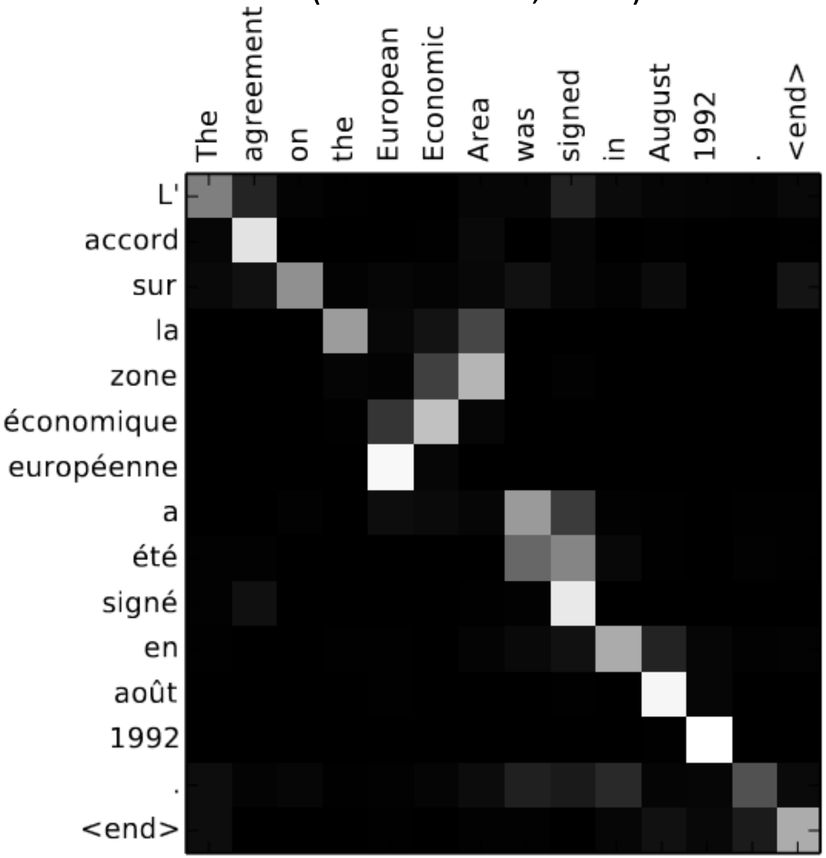


We propose a new simple network architecture, the Transformer,
based solely on attention mechanisms, dispensing with recurrence and convolutions entirely.

Attention Is All You Need (Vaswani et al, 2017)



https://ai.googleblog.com/2016/09/a-neural-network-for-machine.html



Bahdanau et al., ICLR 2015

Summary

- Recurrent neural networks maintain an internal state (memory)
- This internal state is useful when data has a temporal component
- They were frequently used in translation and audio processing
- We don’t see them as much over the last few years, but the concepts are still worthwhile to know