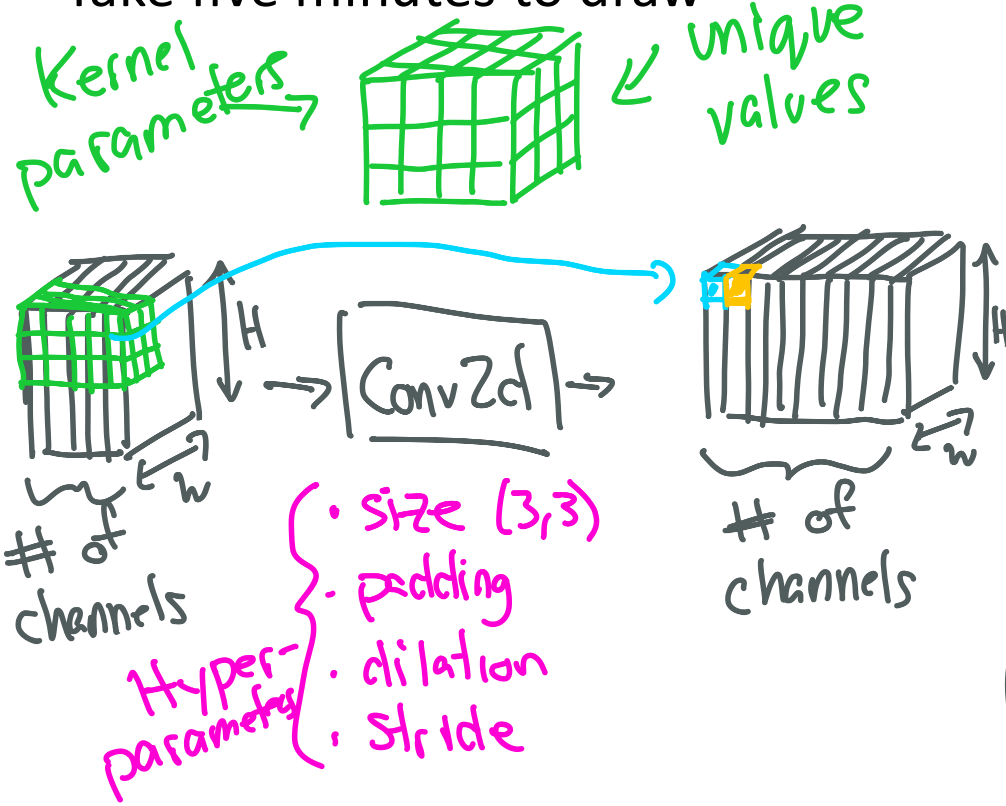
Recurrent Neural Networks

Outline

- Recap convolutional neural networks
- Compare conventional and recurrent neural networks (RNNs)
- Text translation example
- Backpropagation through time
- Code comparison
- Parts-of-speech example
- RNN paradigms
- LSTMs
- A mention of attention

Recap: Convolutional Neural Networks

- Take five minutes to draw



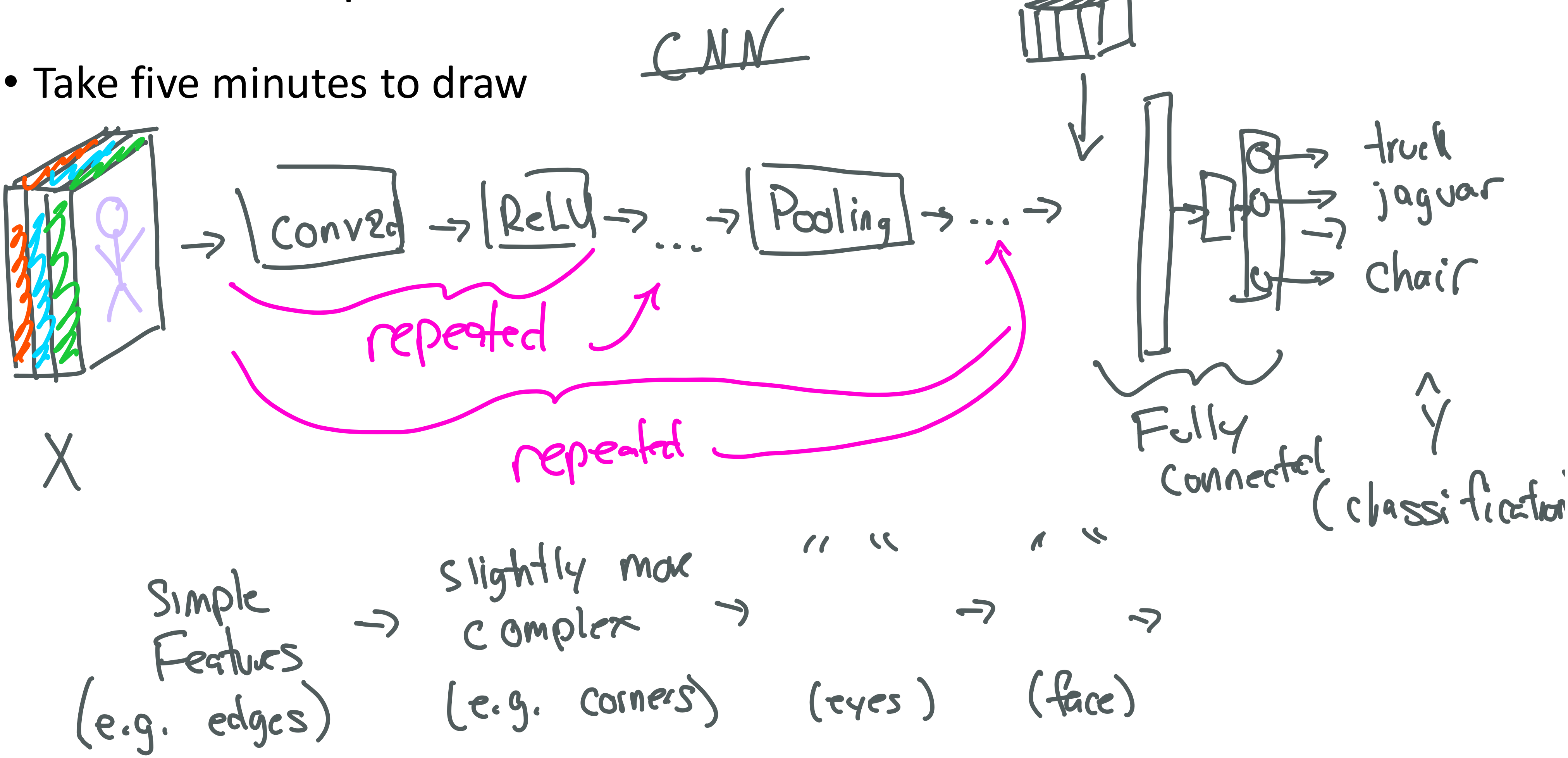
Zoomed-In

for each kernel → K
for w in width
for h in height
dot (X, K)

GPU collapses this to constant time

Recap: Convolutional Neural Networks

- Take five minutes to draw



Recap: Convolutional Neural Networks

- Take five minutes to draw

Reminder { Text in image generation

X = Blood Pressure

Sleep Stages

Image Dataset

- We need an image dataset for next week's lecture on inference/deploying.
- I want you all to take photos of whatever we're classifying.

- From previous semesters
  - Frank or Frary
  - Pine or Palm
  - Spoon or Fork
  - Cup or Bowl
  - Salt or Sugar

Slack

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.
- Output: J'adore les chats violets. Les chats sont soignés.

X

1 I
2 Love
3 Purple
4 Cats

4

5 Cats
6 are
7 neat

3

→ FF (FC/DN) →

Y

Je 1
adore 2

Word by word :(
Sentence by sentence :)
Bag of words :(

Recurrent Neural Networks

Operate over sequences (data with temporal dependencies).

Fully Connected NN

$$\hat{y} = f(x, \theta)$$

$$x \to \boxed{FCNN} \to \hat{y}$$

$$\underbrace{\text{I}}_{x_1} \quad \underbrace{\text{love}}_{x_2} \quad \underbrace{\text{cats}}_{x_3}$$

Recurrent NN

$$\hat{y} = f(x, \theta, h)$$

$$\begin{tikzcd} x \arrow[r] & \boxed{RNN} \arrow[r] \arrow[loop below, "h"] & \hat{y} \end{tikzcd}$$

$$\underbrace{\text{I}}_{x_1} \quad \underbrace{\text{love}}_{x_2} \quad \underbrace{\text{cats}}_{x_3}$$

Linear vs (Elman) Recurrent Neurons

```
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)
```

$$z_{FC} = X W^T + b$$

$$z_R = X W_x^T + b + h W_h^T$$

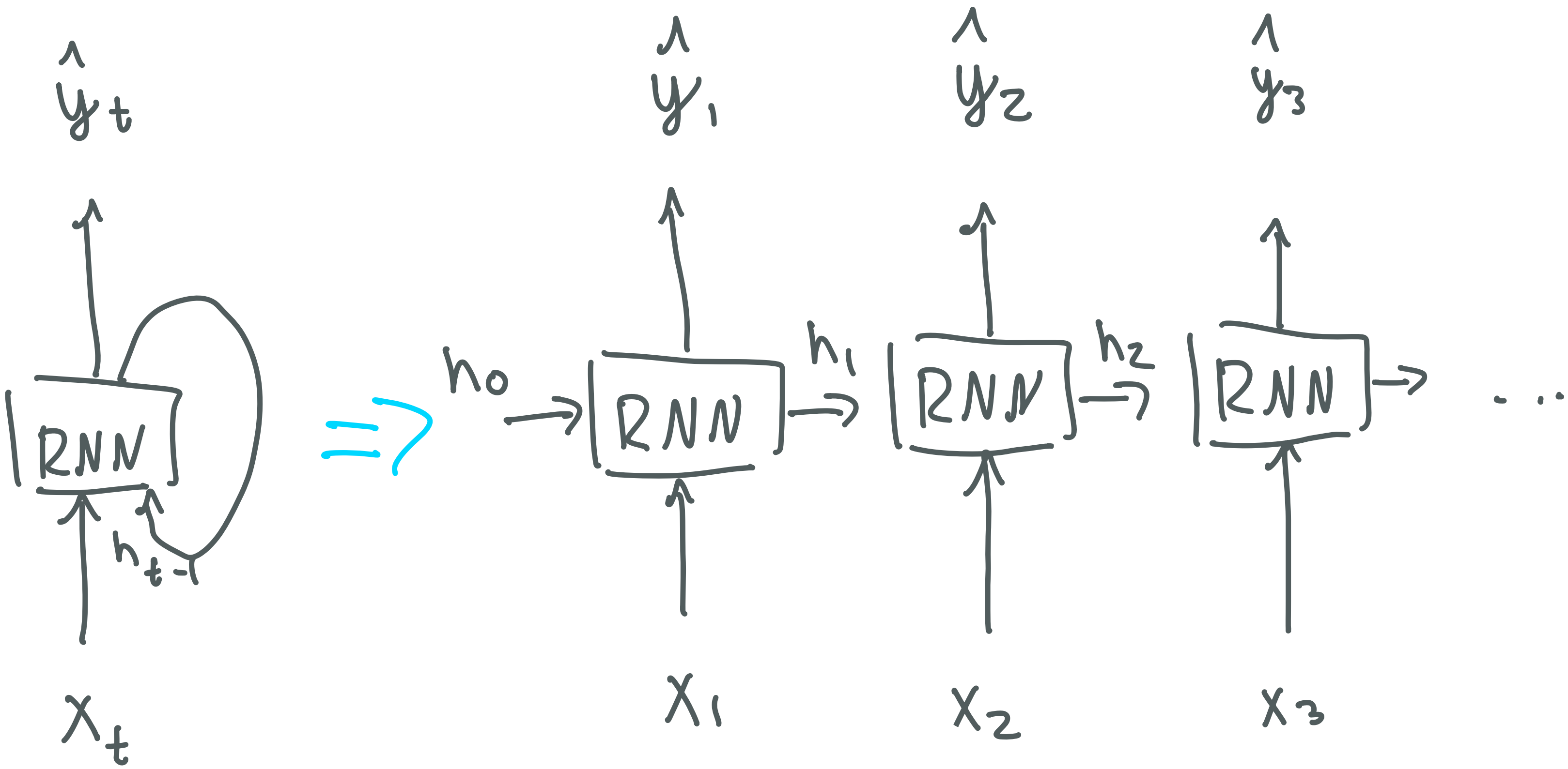
- Input shape:
- Output shape:

```
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:
- Output shape:

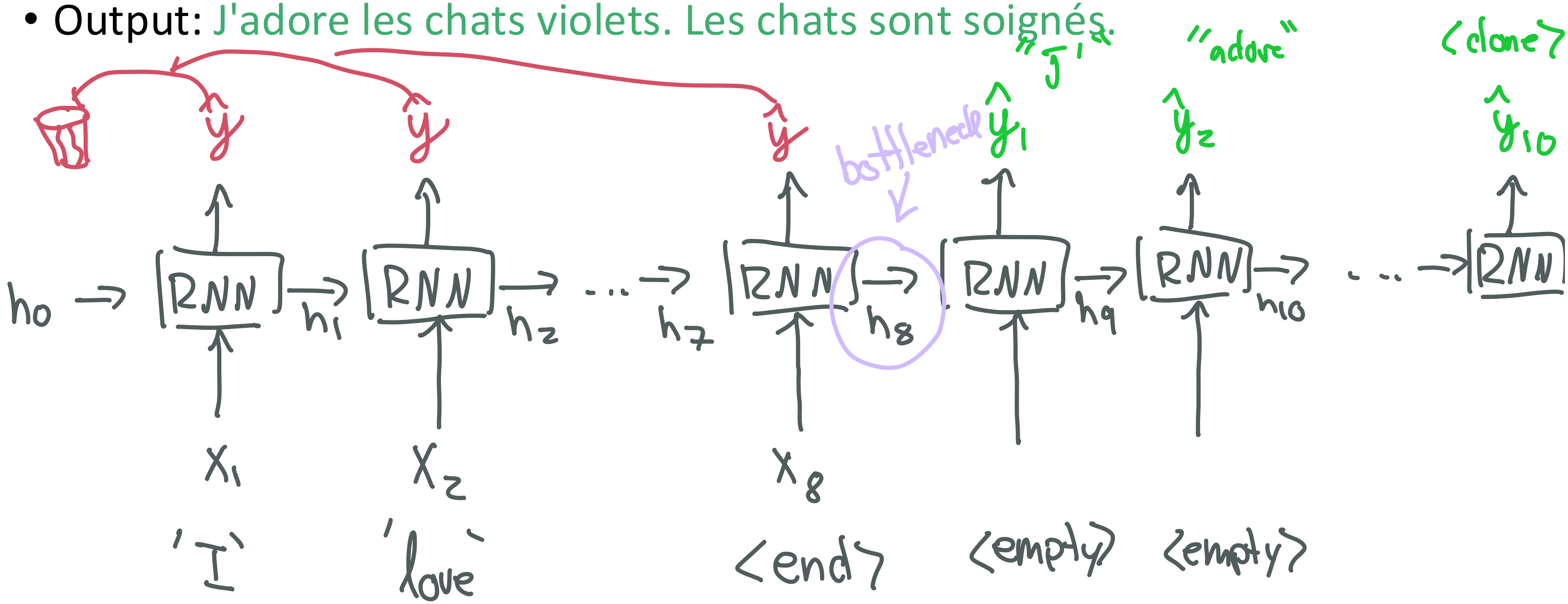
*Untested code.

Unrolled Visualization



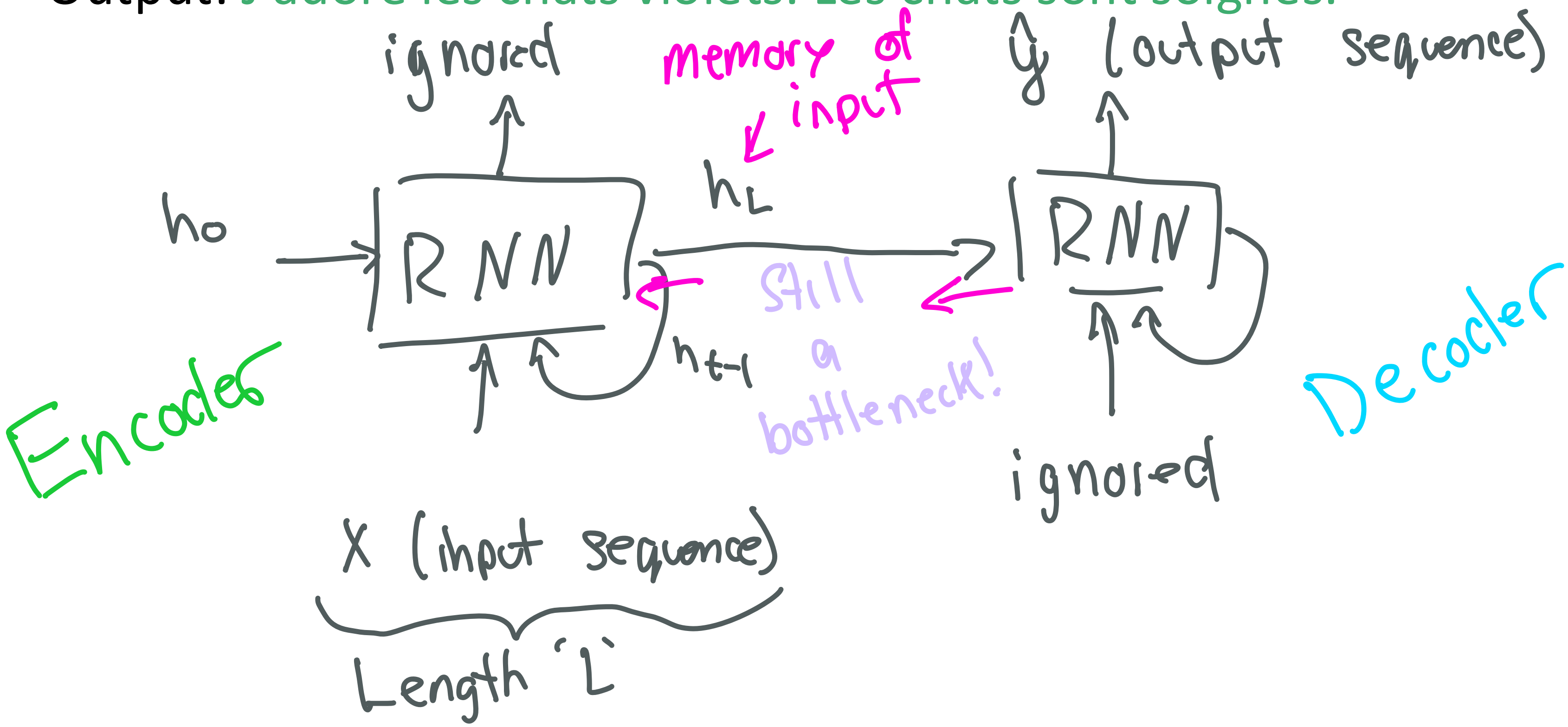
Motivating Example: Text Translation

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



Motivating Example: Text Translation

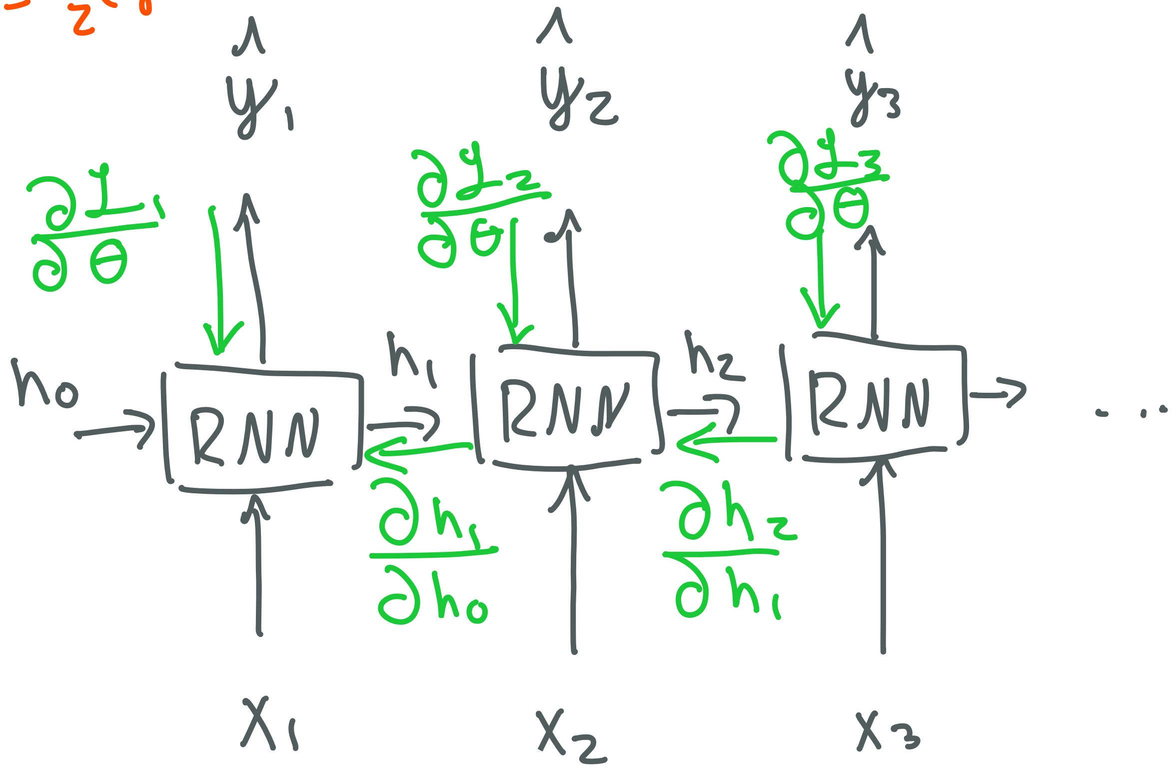
- Input: I love purple cats. Cats are neat.
- Output: J'adore les chats violets. Les chats sont soignés.



Unrolled

Backpropgation Through Time (BPTT)

$\mathcal{L} = \frac{1}{2}(\hat{y} - y)^2$



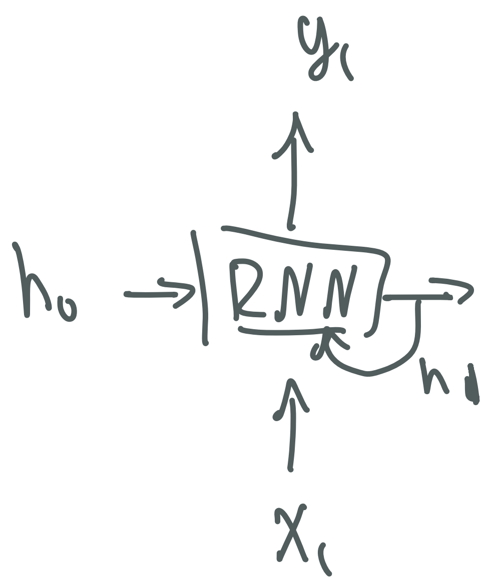
$h_t = x W_x^T + h_{t-1} W_h^T + b$

Backpropgation Through Time (BPTT)

Some BPTT Math

RNN Paradigm: One to One (RNN unneeded)

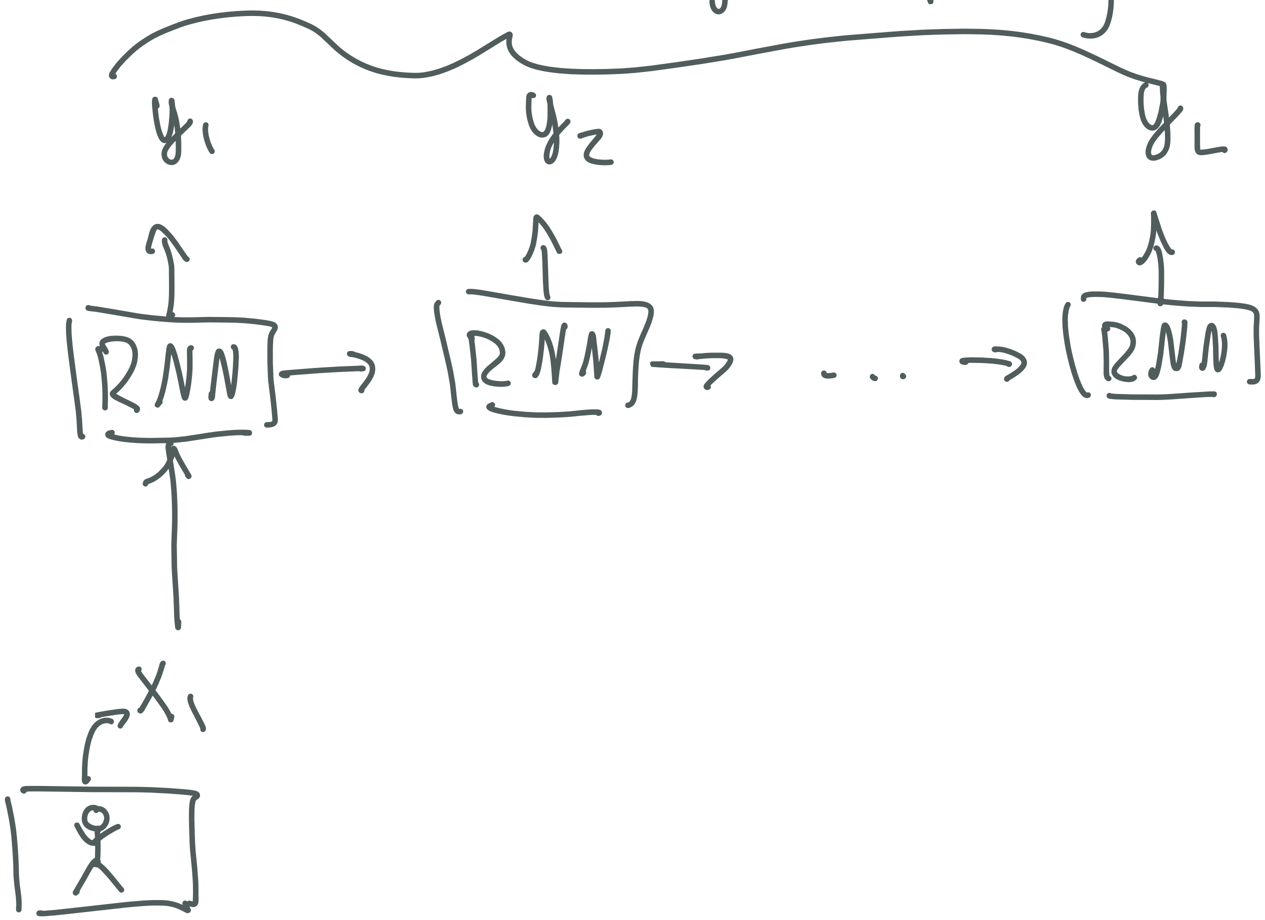
No need for recurrent connections.



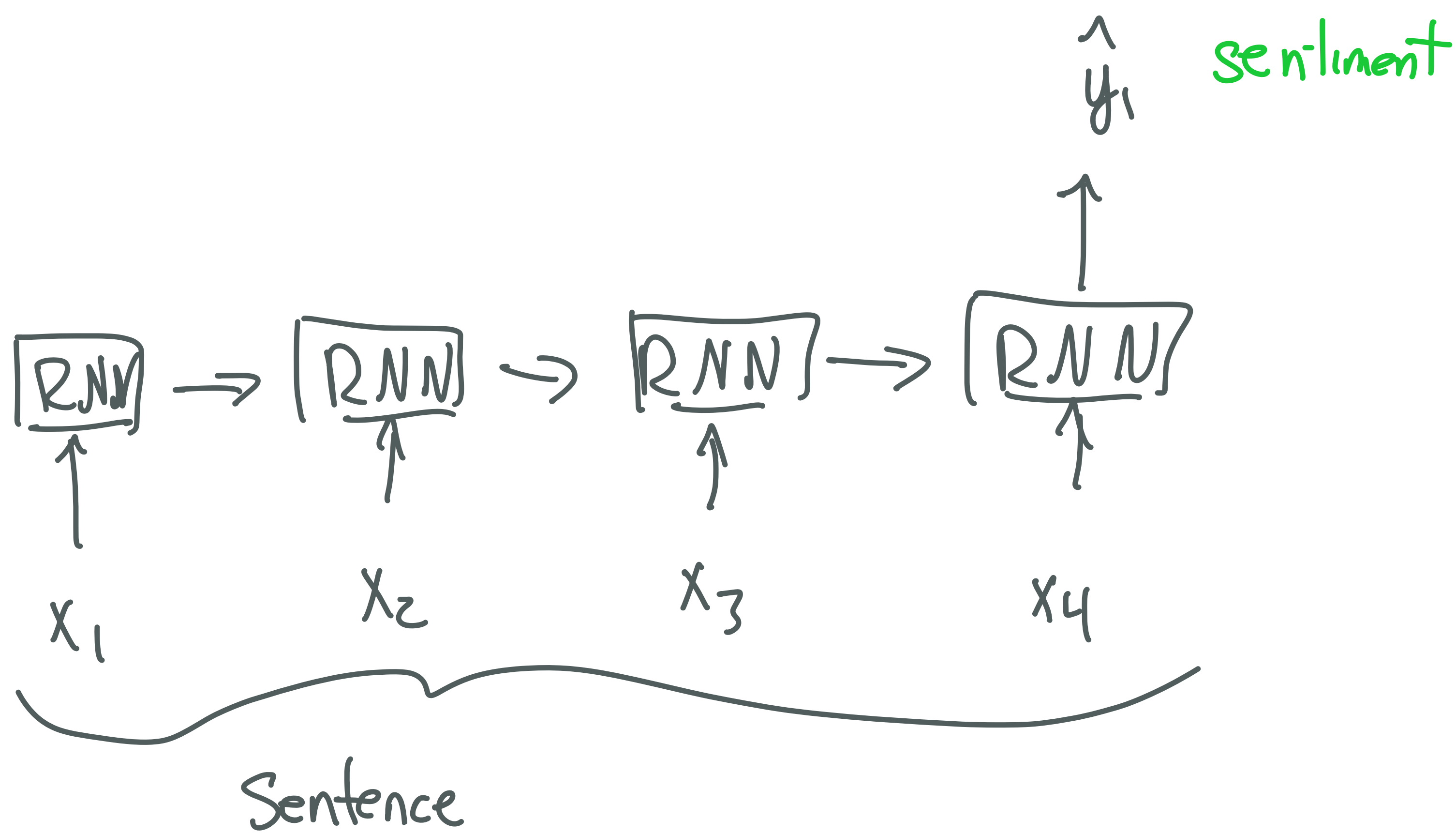
Bag of words

RNN Paradigm: One to Many

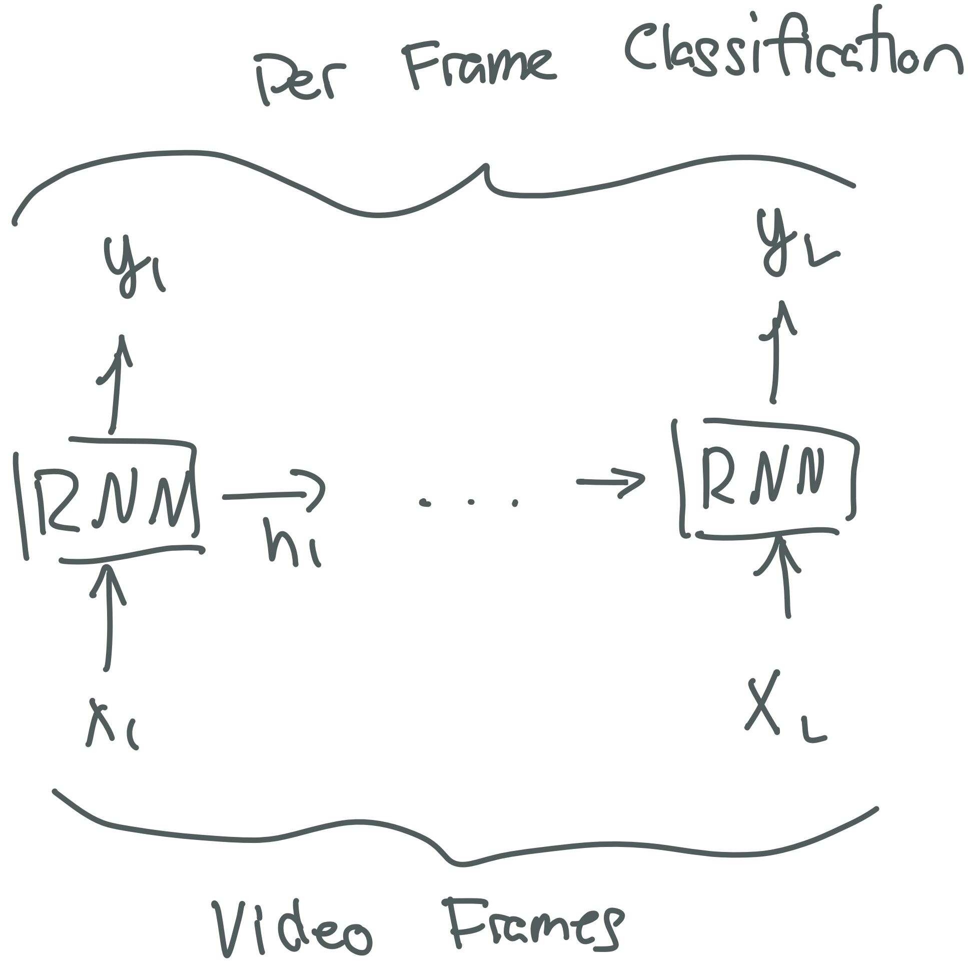
Auto- Image Captioning



RNN Paradigm: Many to One



RNN Paradigm: Many to Many (Synced)



RNN Paradigm: Many to Many (Encoder/Decoder)

Text translation
from previous
slides

Parts-Of-Speech Example

You all must submit sentences for the dataset.

https://docs.google.com/spreadsheets/d/1HJmlehaYhGWclDo1t0k6i1VHxN15zr8ZmJj7Rf_VEaI/edit#gid=1031300490

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

How do we pass this into a neural network?

Processing Natural Language with an NN

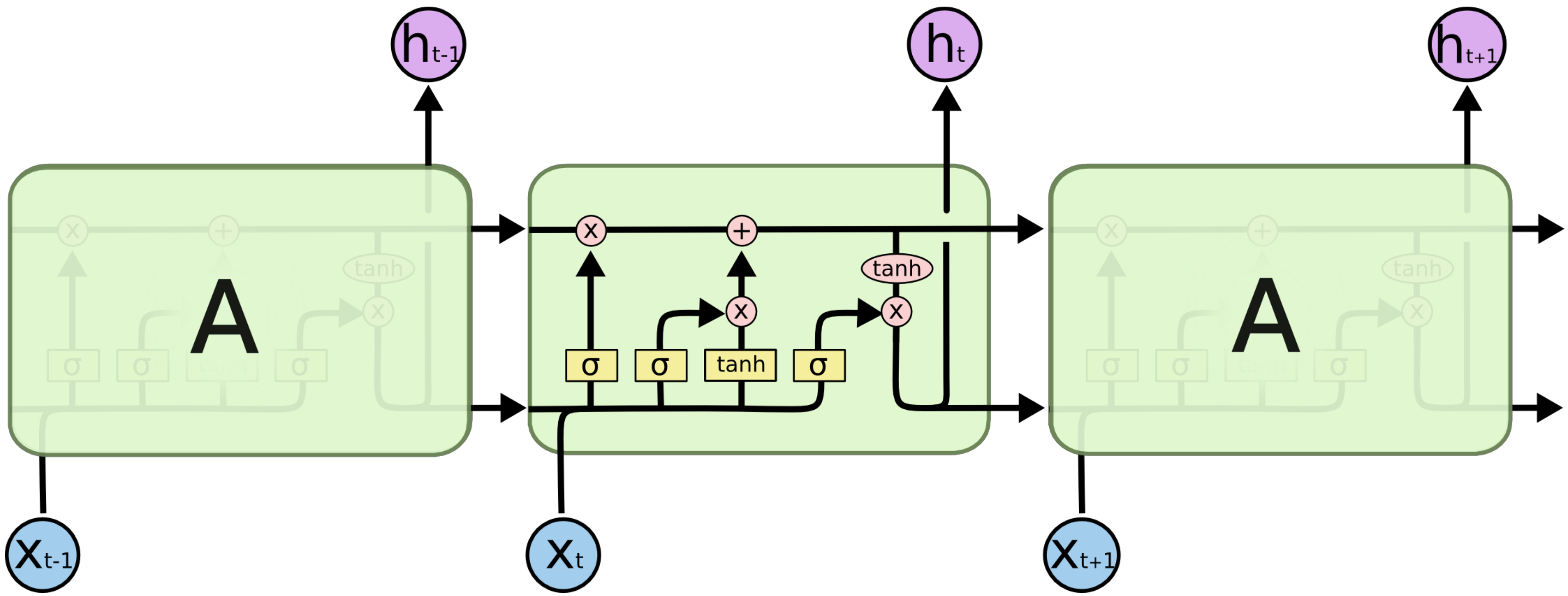
Here's one way to convert text into numbers

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)
```

Output activation function handled by torch.nn.CrossEntropyLoss

LSTMs



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

RNNs

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

RNNs

Advantages

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

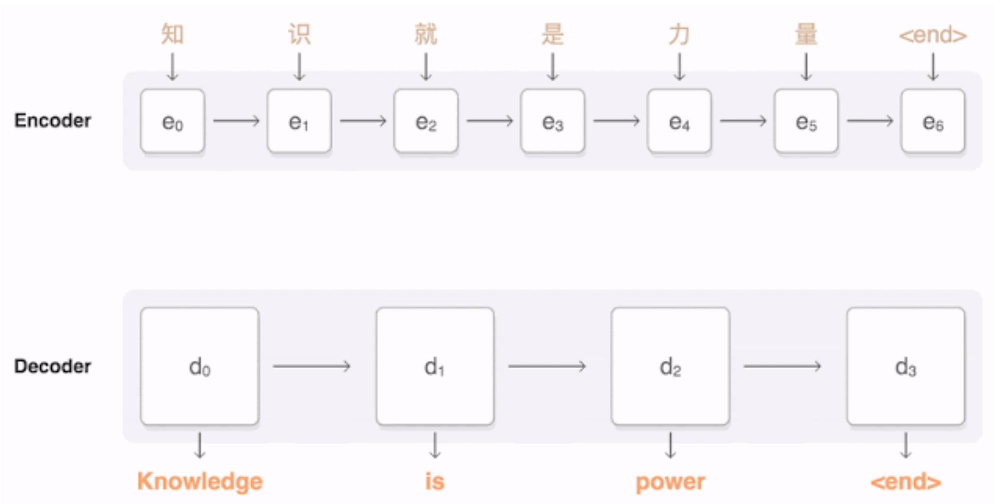
Input: I love purple cats. Cats are neat.
Output: J'adore les chats violets. Les chats sont soignés.

Disadvantages

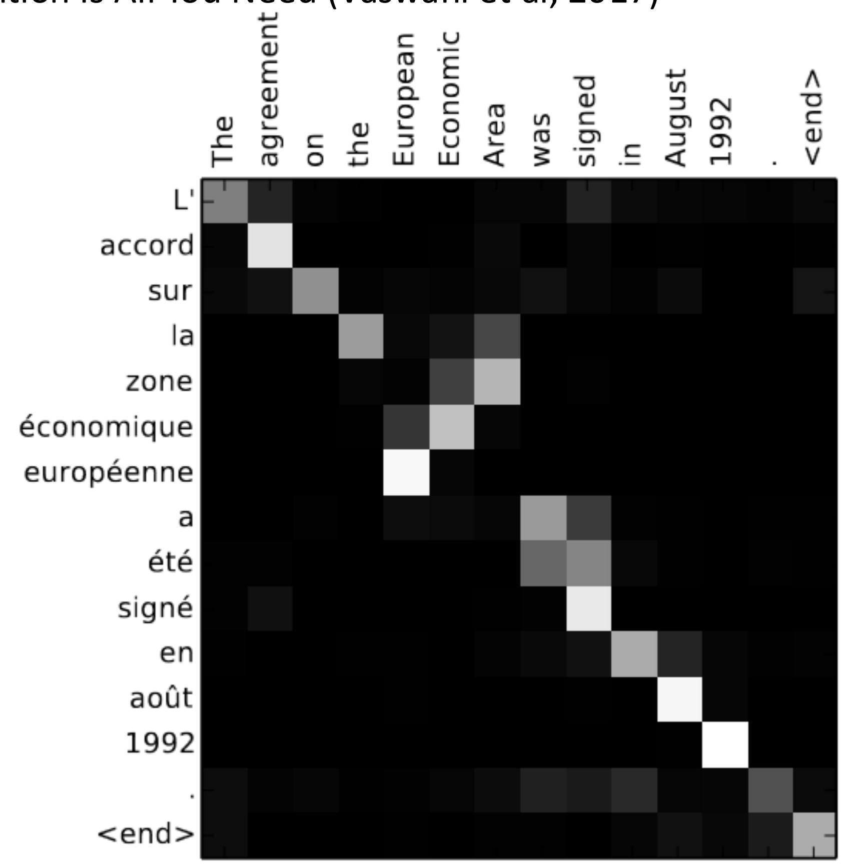
- Slower to computer
- Poor handling of long-term dependencies
- Does not consider future inputs to produce current state
- Largely replaced by transformers

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