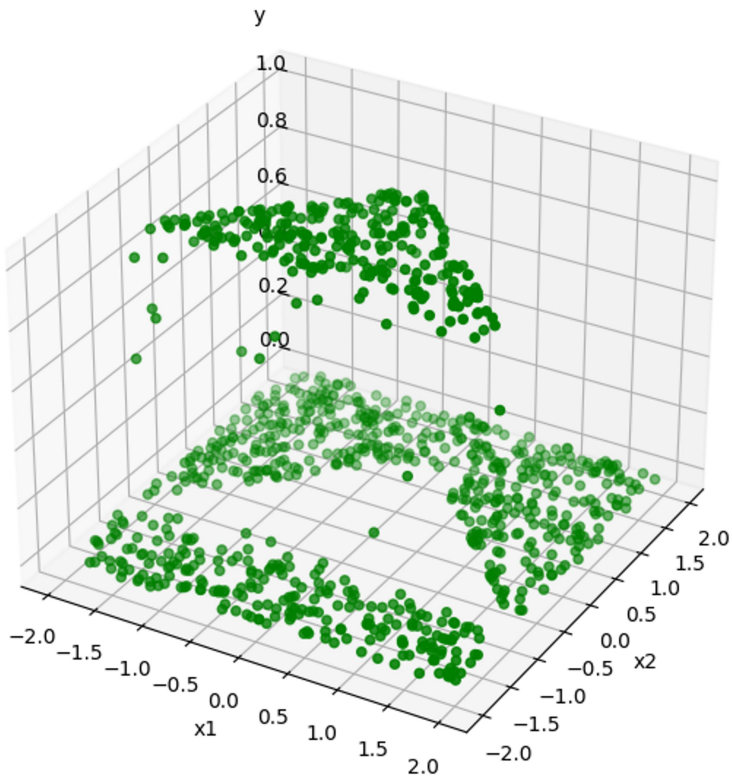



Note the faster convergence and the crisper output. The convergence is noisier due to the large step sizes, but it's controlled thanks to the adaptation of $\eta$ and the averaging inside the mini-batch.

(set batch_size=20 to see it get smoother but a bit slower.)

# L25

ECE/CS 559 Lecture 25 11/26/2024

Last time: RNNs

① **Language modeling**

Language as a sequence of symbols (tokens) $x_1, x_2, \dots, x_t, \dots$

Objective: Maximize $P(x_1, x_2, \dots, x_t, \dots)$ of data

However, just like images, this is a high-dimensional space. Any specific sequence has exponentially small probability!

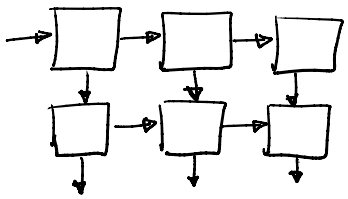
Instead: Write $P(x_1, x_2, \dots, x_t, \dots)$ as $P(x_1) \cdot P(x_2|x_1) \cdots P(x_t|x_{t-1}, \dots) \cdots$

Maximize $\sum_t \log P(x_{t+1} | x_t, x_{t-1}, \dots)$ ← this probability is not as small!

- RNNs can be used to represent these prediction probabilities. How? Often, we let $y_t$ (the output) be logits. We go from logits to probabilities via a softmax:

$$P(x_{t+1} = i \mid x_t, x_{t-1}, \dots) = \frac{\exp(y_t[i])}{\sum_{i'} \exp(y_t[i'])}$$

- Issue with RNN: they're best with multiple layers



← This layer has to finish before we can do backpropagation on this layer

Not Parallelizable

② **(Self) Attention**

- Instead of recurrence, why not directly assess what past information is relevant? (like in databases)
  - Query: What the current symbol represents
  - Key: The relevance of other sequence symbols to a query.
  - Value: What each sequence query contributes

weights $R^{k \times d}$ embedding $\in \mathbb{R}^d$ attention weights

$$Q_{t,h} = W_h^Q x_t$$
$$K_{t',h} = W_h^K x_{t'}$$ ← other

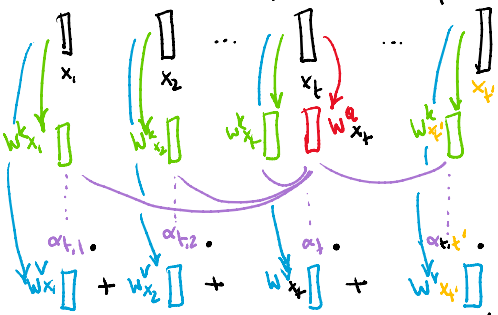
$$\alpha_{t,t',h} = \text{softmax}\left(\frac{1}{\sqrt{k}} Q_{t,h}^T K_{t',h}\right)$$

scaled, inner product

$$V_{t',h} = W_h^V x_{t'}$$

(ignore all h at first)

- Then, to generate the new state variable, we combine values based on attention $S_t = W^O \underset{h}{\text{concat}} \sum_{t'} \alpha_{t,t',h} V_{t',h}$
- We could also have multiple heads, copies w/ different weights




(issue: $T^2$ complexity)

Notes:
- Each head is given by the triple $(W^Q, W^K, W^V)$
- The weights don't change by position.
- Position info can be placed inside $x_t$ (positional embedding)
- Usually dimensions are maintained: #heads × d = k, $W^O$ square.

③ **Transformers**

- Attention only tells us where the relevant information is
- We still need to act on it. We do it using:

(a) Add and normalize (residual connection)

$$S' = \text{LayerNorm}(S + X)$$

$$\mu_t = \frac{1}{T}\sum_t S_{t,i} + X_{t,i}$$
$$\sigma_t^2 = \frac{1}{T}\sum_t (S_{t,i} + X_{t,i} - \mu_t)^2$$
$$S'_{t,i} = \gamma \frac{S_{t,i} + X_{t,i} - \mu_t}{\sqrt{\sigma_t^2 + \varepsilon}} + \beta$$

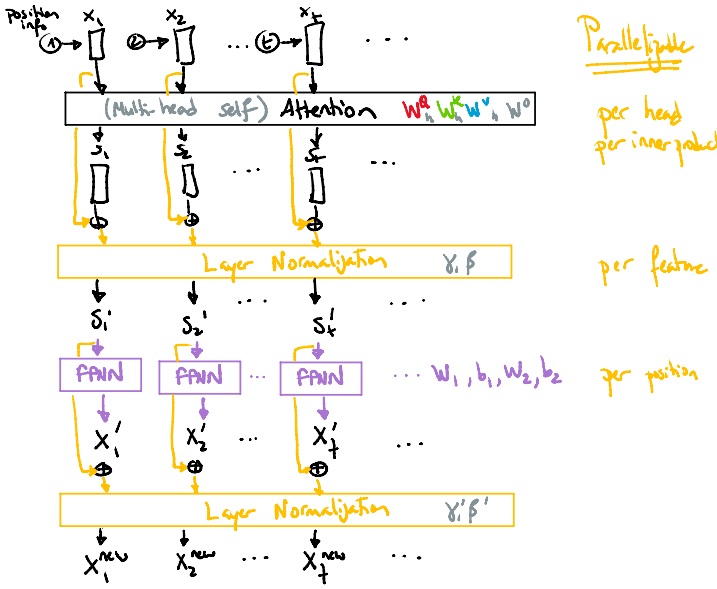
learnable, learnable

(b) 2-Layer Feedforward Neural Network

$$X'_t = W^2 \text{Relu}(W^1 S'_t + b^1) + b^2$$

(c) Add and normalize (again)

$$X^{new} = \text{LayerNorm}(S' + X')$$




Parallelizable

per head per inner product

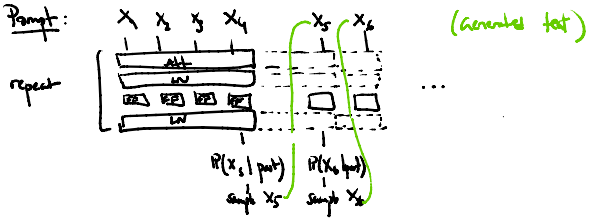
per feature

per position

## (4) Using a transformer:

### (a) Decoder-only architecture (ChatGPT)

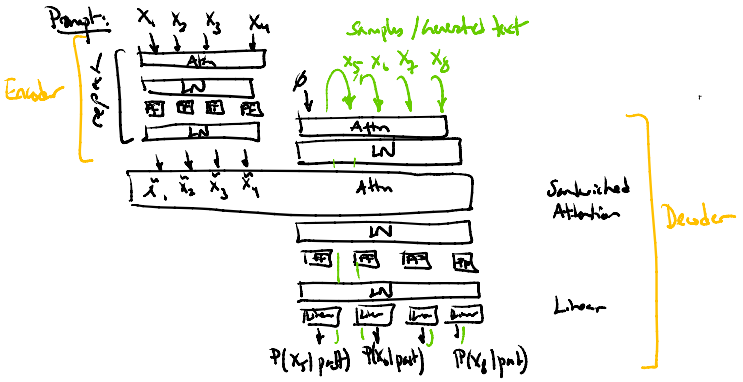
- After the prompt, we past outputs as inputs.
- Attention restricted to past outputs (since we generate forward)




- Variations: Beam Search of most likely generation:
  - Maintain k sequences, sample m (expand to km), prune back to top k

### (b) Encoder-Decoder Architectures

- The outputs of an encoder transformer become the inputs of a decoder transformer, usually in sandwiched layer:



# L24

# Recurrent Neural Networks (1)

Erdem Koyuncu

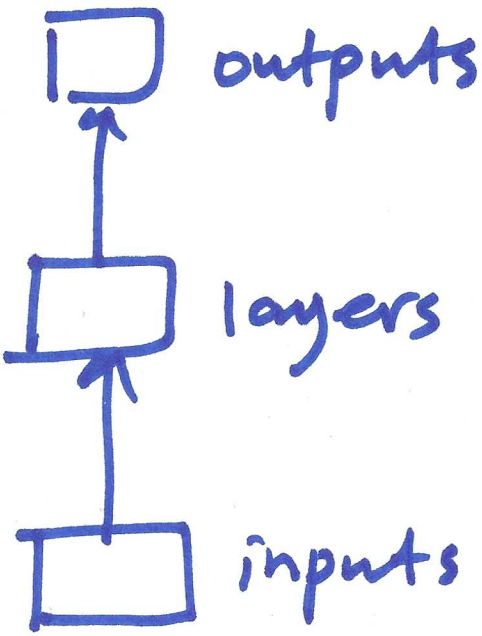
RNNs are used to process sequences (a sequence of images, words, letters, etc) in a way that takes into account the dependencies between the individual elements (or, the memory) of the sequence.

In regular (feedforward) neural networks, we provide an input to the network and observe the output after possibly many hidden layers. There is no feedback of the outputs back to the input.

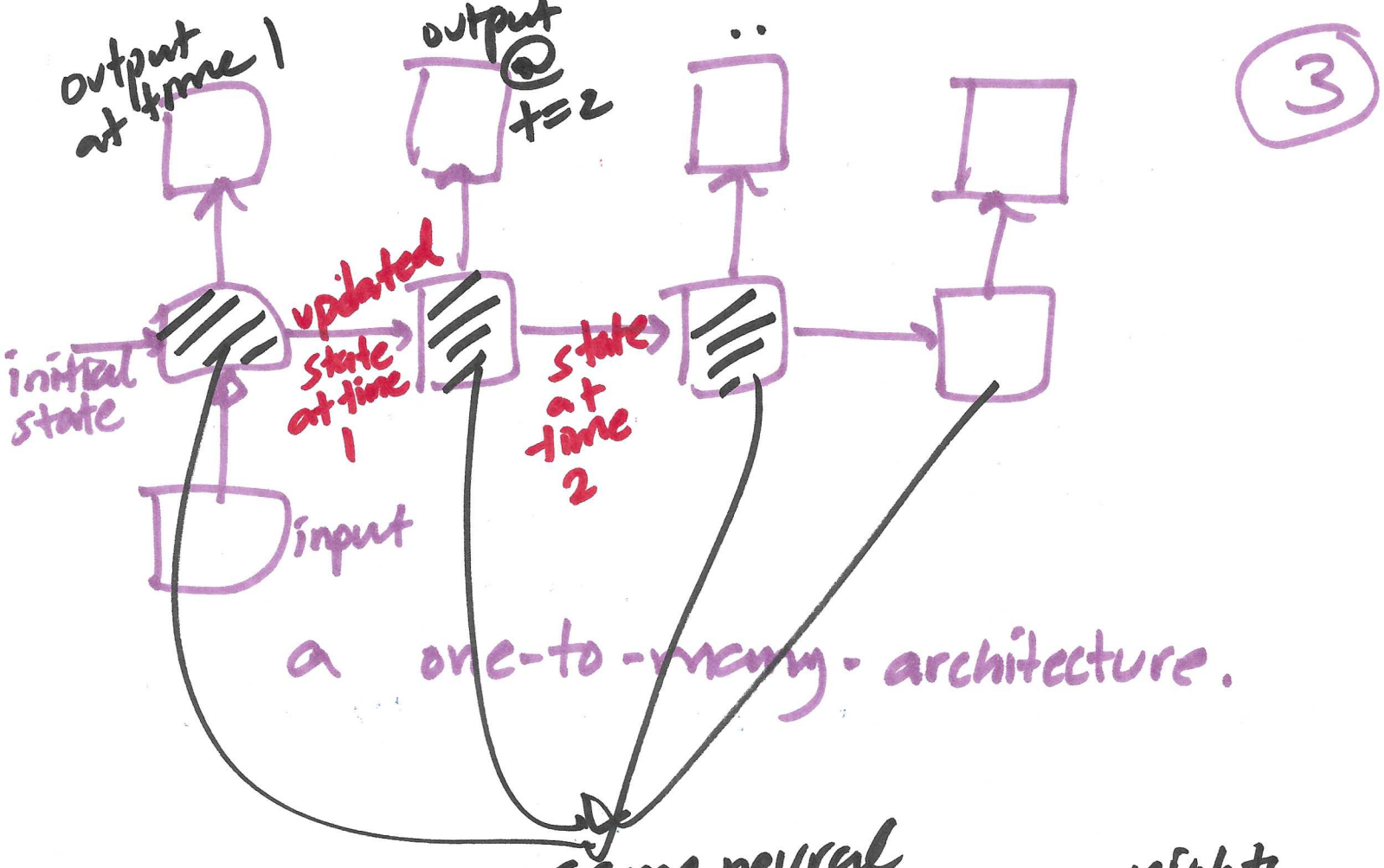
In RNNs, in general, we feed the input sequence one element at a time. Each input updates an internal state of the neural network,

which also serves as the RNN's memory. The output at any given time depends on the input at that time as well as the memory. One can imagine that the network feeds the internal states back to the network. (2)

Ordinary (feedforward) NN:




a one-to-one architecture.




3

Ex: image description/captioning

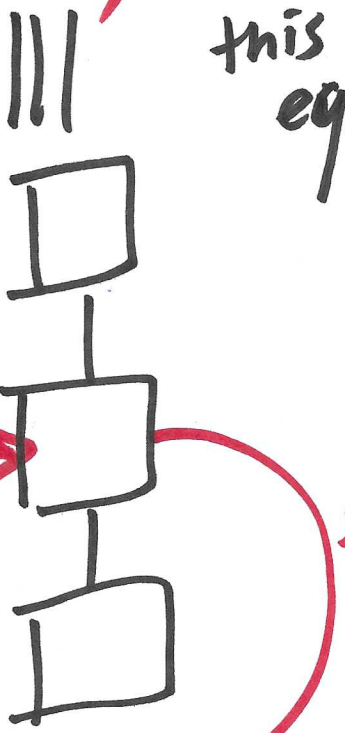
Input can be an image:

Output can be a sequence of words describing the image

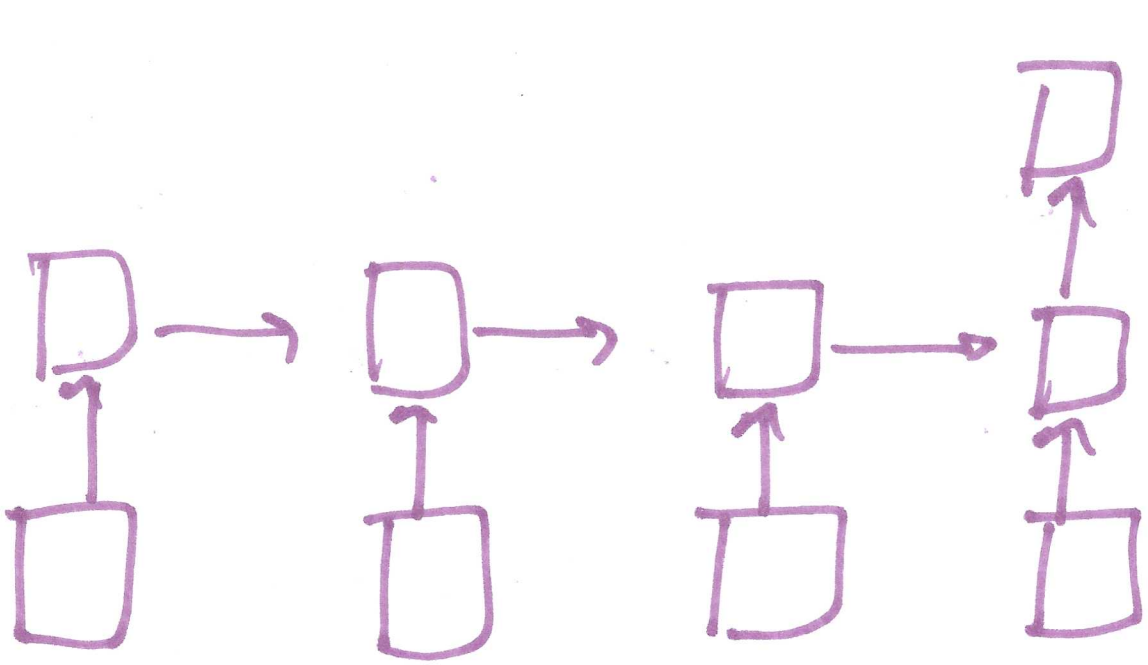
"person holding an apple"

outputs at different times will be different words.

this is equivalent to




unrolled 3 times.

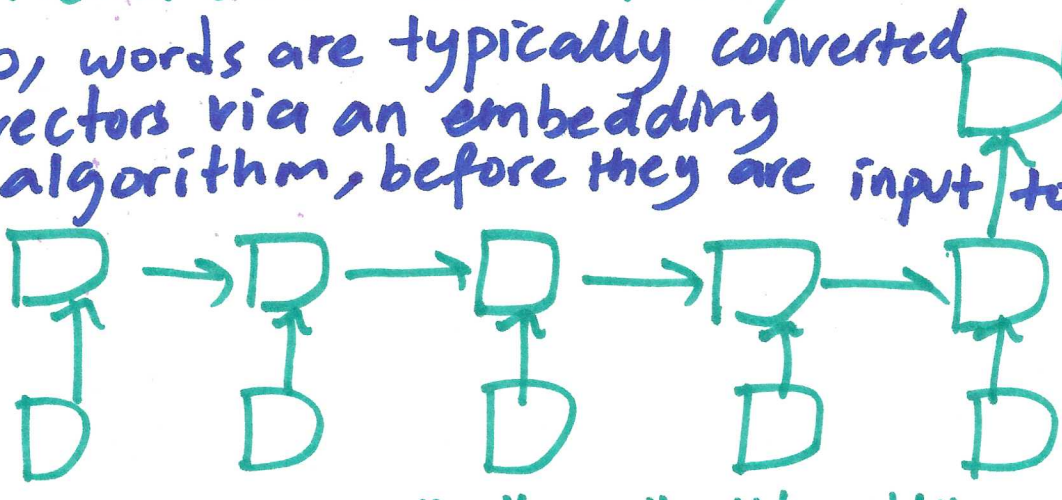


many-to-one architecture.

Ex: sentiment analysis
Input is a sequence of words
Output is whether that describes a good or bad sentiment (or any other class that we are interested in)

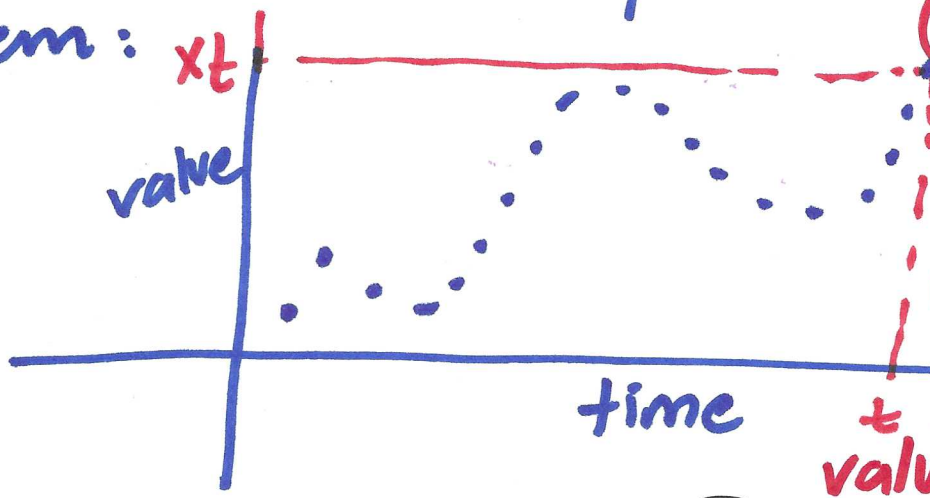
In most of these problems we will need an "END of sequence" word EOS.

Also, words are typically converted to vectors via an embedding algorithm, before they are input to the RNN.

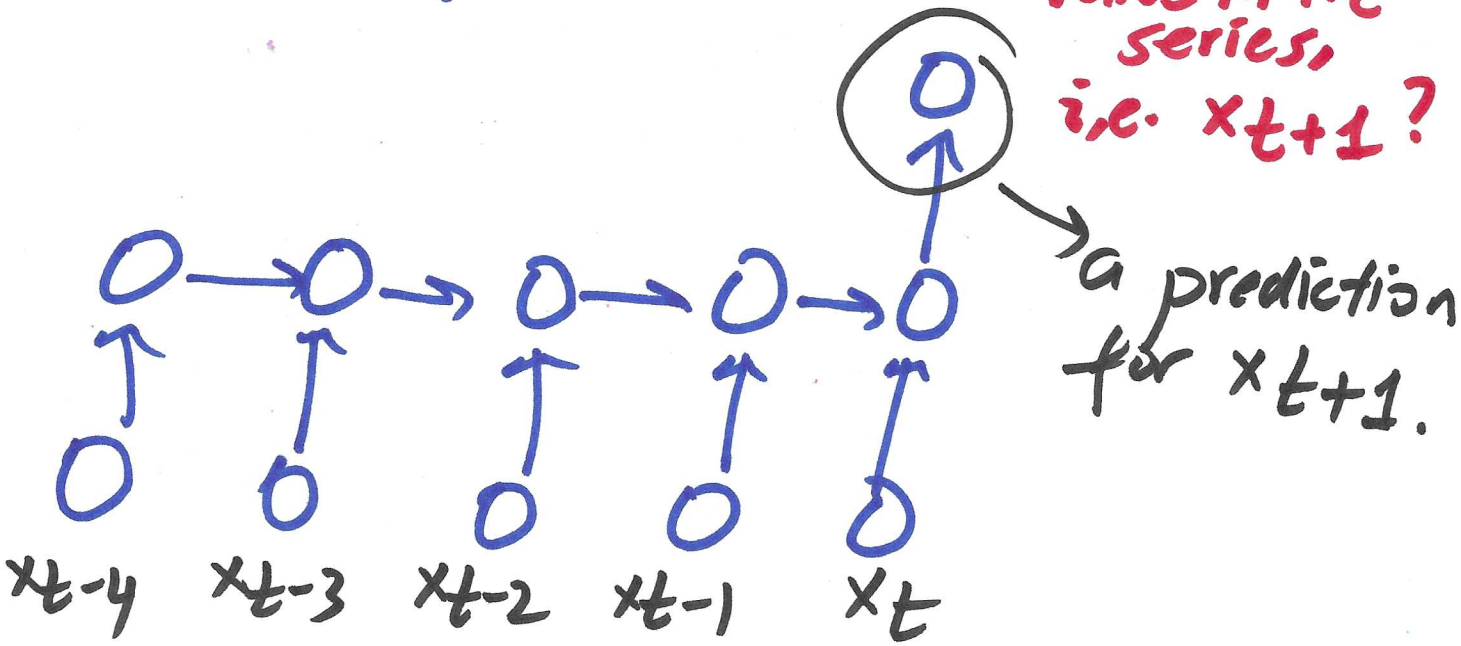



So, we are not inputting "words" of course but their embedded versions.

Another application of the many to one architecture could be a general time-series prediction problem:




what is the next value in the series, i.e. $x_{t+1}$?




a prediction for $x_{t+1}$.