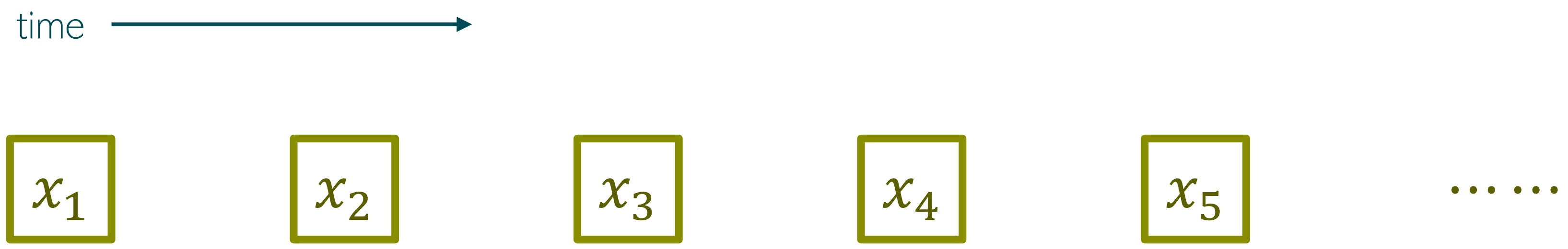
COMP0197: Applied Deep Learning

Yipeng Hu
yipeng.hu@ucl.ac.uk

Sequence Modelling

Independent and identically distributed random variables – the i.i.d. assumption



$$p(x_1, \cdots x_N) = p(x_1) \prod_{n=2}^{N} p(x_n | x_1, \cdots x_{n-1})$$

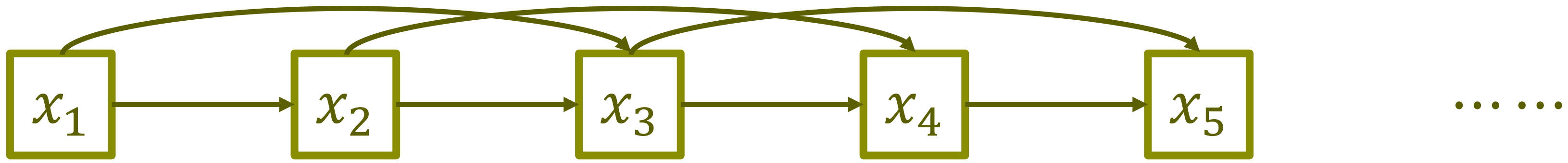
$$p(x_1, \cdots x_N) = \prod_{n=1}^{N} p(x_n)$$

The first-order Markov model

$$\boxed{x_1} \to \boxed{x_2} \to \boxed{x_3} \to \boxed{x_4} \to \boxed{x_5} \quad \cdots\cdots$$

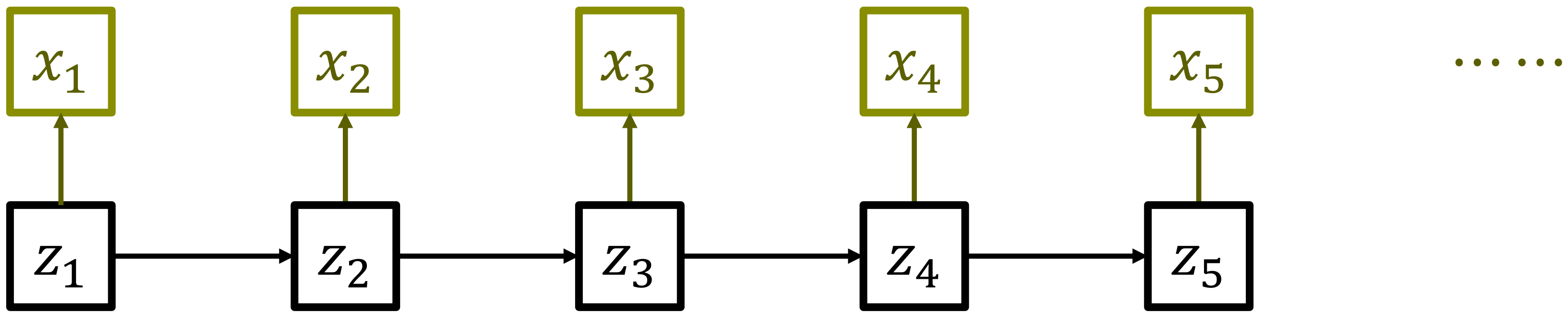
$$p(x_1, \cdots x_N) = p(x_1) \prod_{n=2}^{N} p(x_n | x_{n-1})$$

The second-order Markov model



$$p(x_1, \cdots x_N) = p(x_1)p(x_2|x_1)\prod_{n=3}^{N} p(x_n|x_{n-1}, x_{n-2})$$

The hidden Markov model



$$p(x_1, \cdots x_N, z_1, \cdots z_N) = p(z_1) \prod_{n=2}^{N} p(z_n | z_{n-1}) \prod_{n=1}^{N} p(x_n | z_n)$$

- Sequential data

  NLP, word embedding

- Recurrent neural networks

  Unfolding, output recurrence, input as context, bidirectional RNNs,

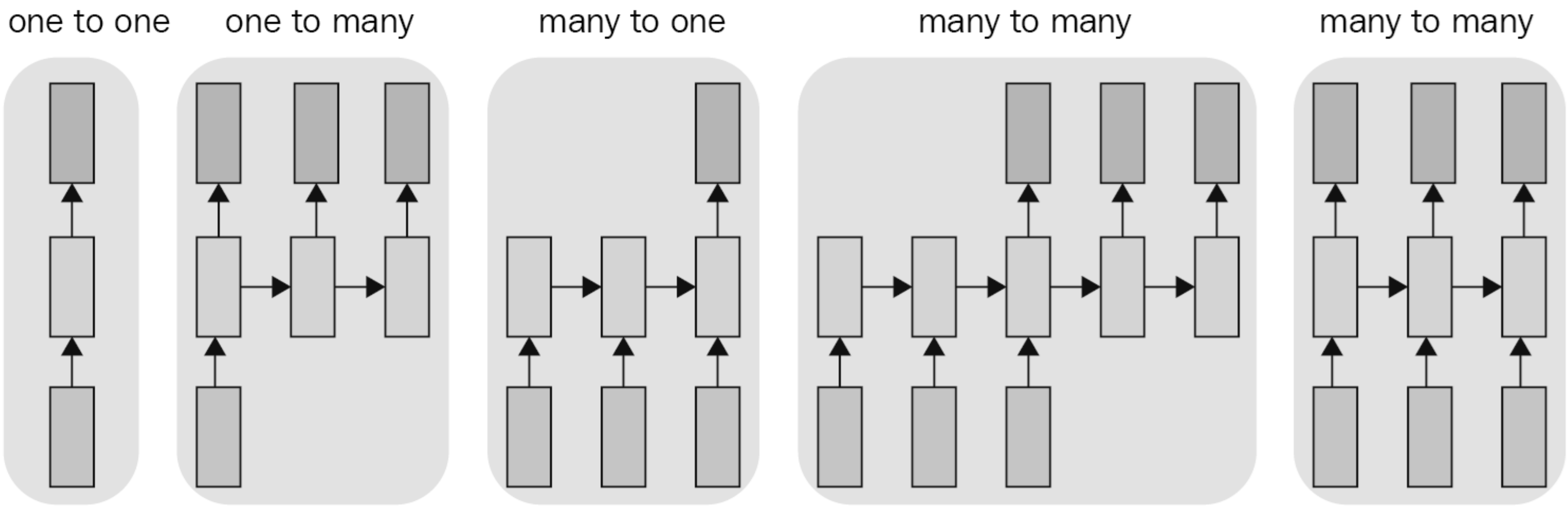
- Backpropagation through time
- Long short term memory
- Encoder-decoder
- Attention mechanism

Sequence Modelling | Sequential Data

Natural language processing

Data: audio, speech, text, word, video, automobile/robotic sensory...

Applications: speech recognition, text analysis, word completion, translation, video captioning...



Language models

Token-based language models

– Modelling sequences of tokens (phrases, n-grams, (sub-)words, characters)

1. Standardisation before tokenisation, e.g. lowercasing, punctuation stripping
2. Tokenisation (10k – 100k tokens), e.g.
    - Word-level tokenisation: straightforward vocabulary construction
    - Byte Pair Encoding (BPR), SentencePeice, WordPeice: adaptive sub-word tokenisation for desirable* vocabulary
        - Size of vocabulary vs. rare / out-of-distribution word
3. Transforming each (string) example into a (e.g. pad-to-max-tokens, trim) vector of token indices

```
Language Learning Models (LLMs) have revolutionized the field of natural
 language processing, enabling machines to understand and generate human-
like text. At the core of LLMs lies the concept of tokens, which serve as
 the fundamental building blocks for processing and representing text
 data. In this blog post, we'll demystify tokens in LLMs, unraveling
 their significance and exploring how they contribute to the power and
 flexibility of these remarkable models.
```

Language models

Word embeddings

- One-hot (independence, extremely sparse and inefficient)
- Indexing (meaningless ordinal)
- Embedding in dense representation
  - One-hot is a special case that encodes no similarity, i.e. independence
  - Learnable representation
  - Embedding space size and dimensionality, oft. 8-1024 trainable weights, (token length: BERT 512-768, GPT 1024-768)
  - Vocabulary vs. dictionary (e.g. key-value-query in transformer)?

One-hot encoding

	cat	mat	on	sat	the
the =>	0	0	0	0	1
cat =>	1	0	0	0	0
sat =>	0	0	0	1	0
...	...				

A 4-dimensional embedding

cat =>	1.2	-0.1	4.3	3.2
mat =>	0.4	2.5	-0.9	0.5
on =>	2.1	0.3	0.1	0.4
...	...			

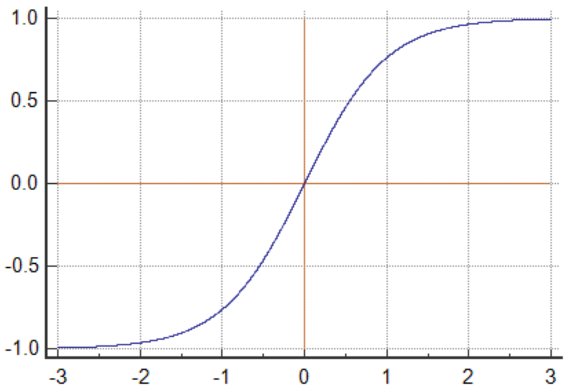
Sequence Modelling | Recurrent Neural Networks

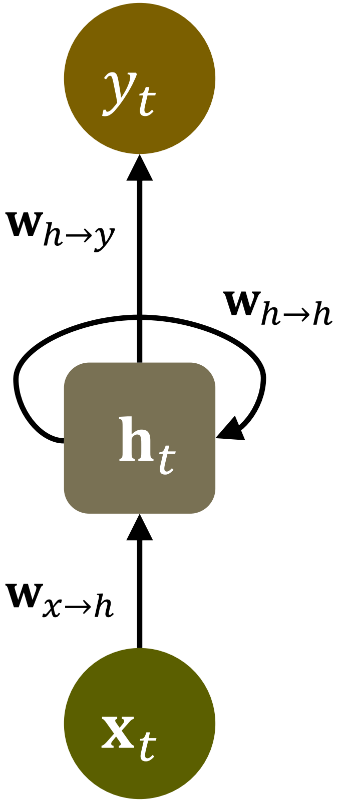
A basic example

- Input vector - $\mathbf{x}_t$

- Hidden layer - $\mathbf{h}_t = f(\mathbf{x}_t, \mathbf{h}_{t-1}, \mathbf{w}_{x\to h}, \mathbf{w}_{h\to h}) = \tau\left(\mathbf{w}_{x\to h}^{\mathrm{T}}\mathbf{x}_t + \mathbf{w}_{h\to h}^{\mathrm{T}}\mathbf{h}_{t-1}\right)$

where $\tau(z) = \tanh z = \frac{e^z - e^{-z}}{e^z + e^{-z}}$

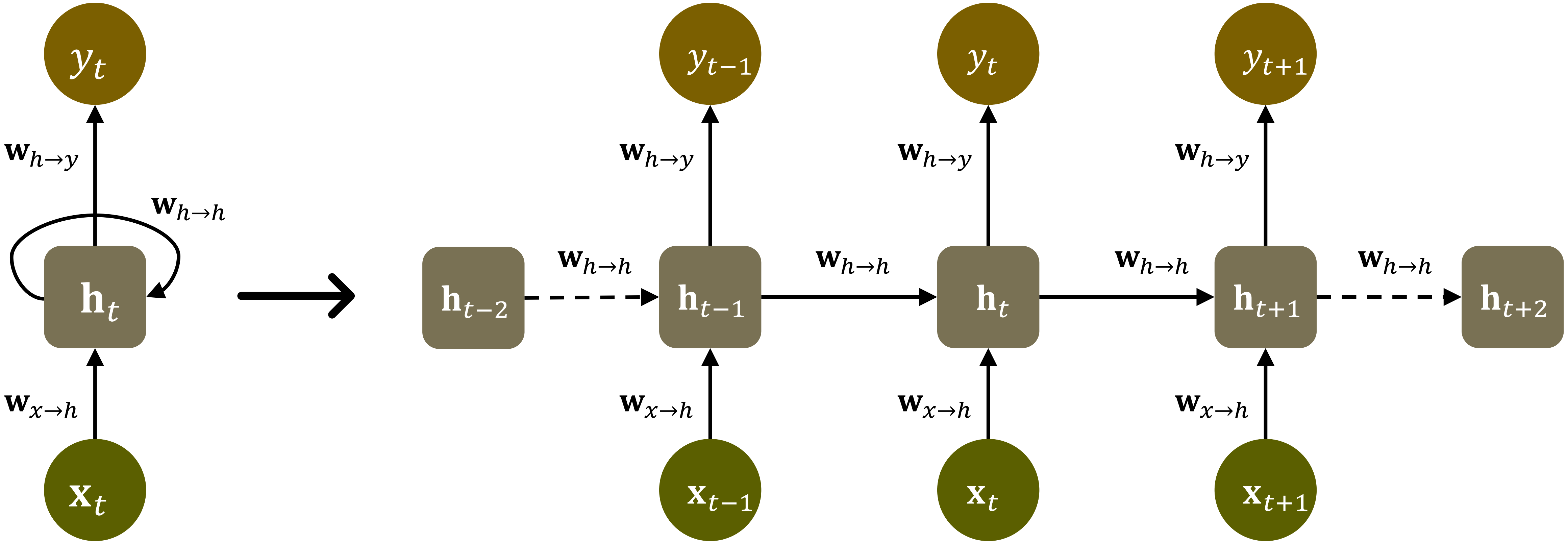




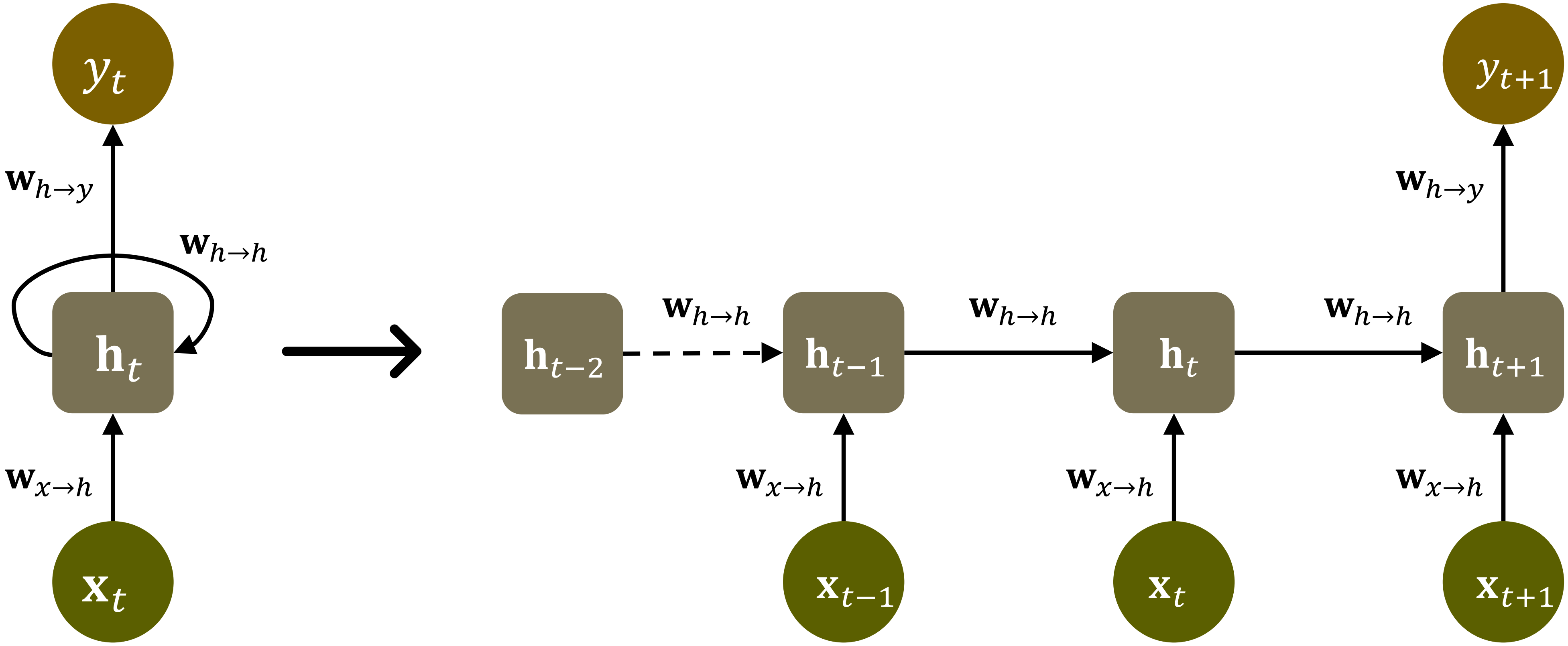
- Output - $y_t = g\left(\mathbf{w}_{h\to y}^{\mathrm{T}}\mathbf{h}_t\right)$

Choice of $g(\cdot)$?

Unfolding / unrolling

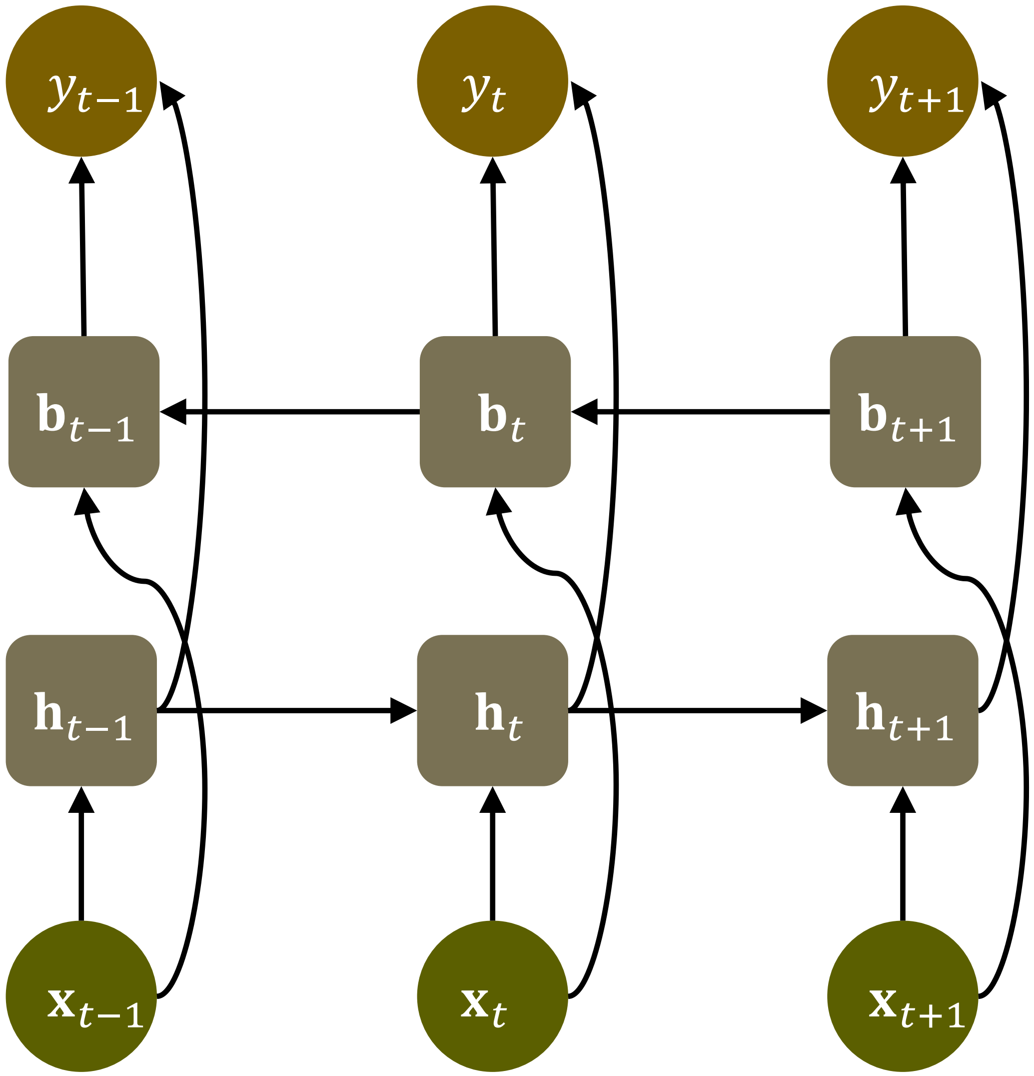


Single output



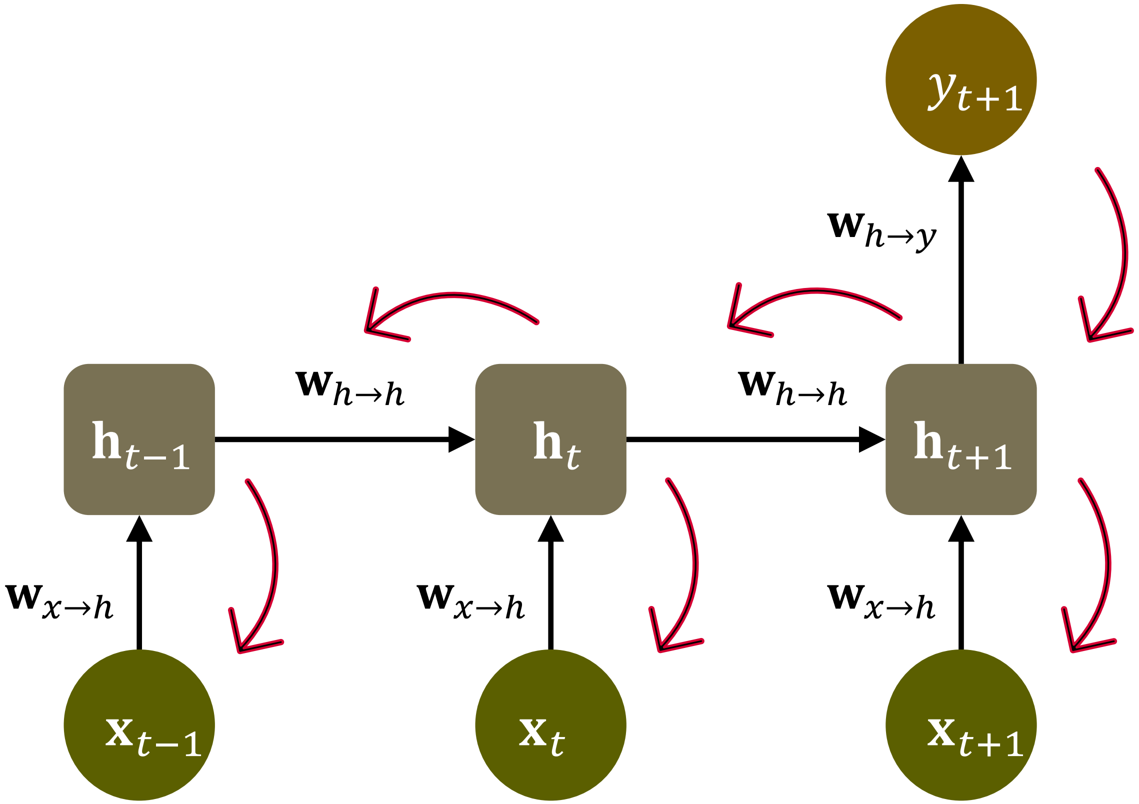
Bidirectional RNNs

- "Future-dependent" application*
- Performance gain?

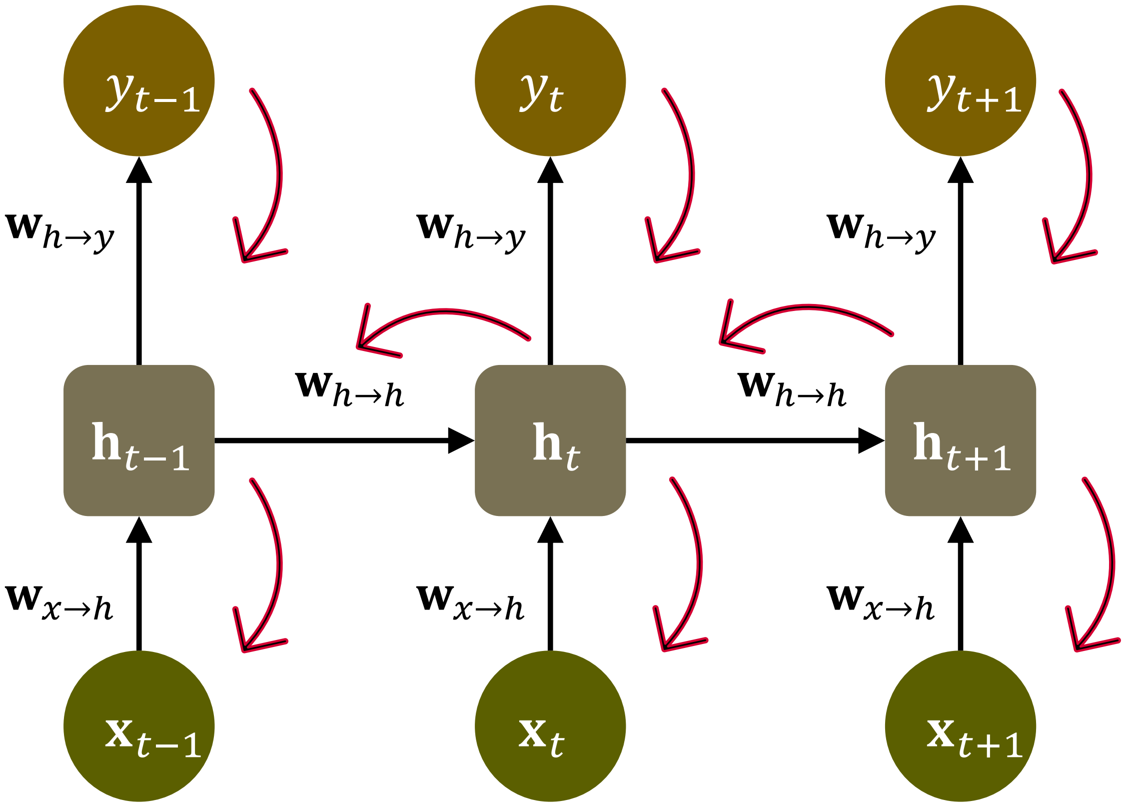


Sequence Modelling | Backpropagation Through Time

Single output

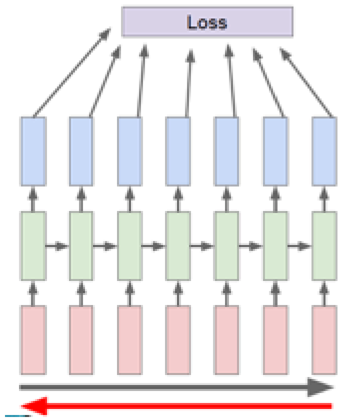


Multiple (variable-length) output



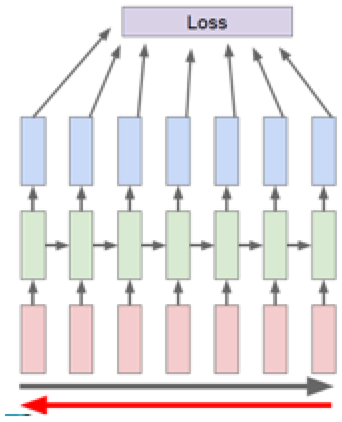
BPTT algorithm

1. Read a sequence of input and output pairs
2. “Unroll” the network
3. Forward evaluation
4. Backward gradient estimation
5. “Roll up” the network
6. Update weights using the accumulated gradients
7. Repeat

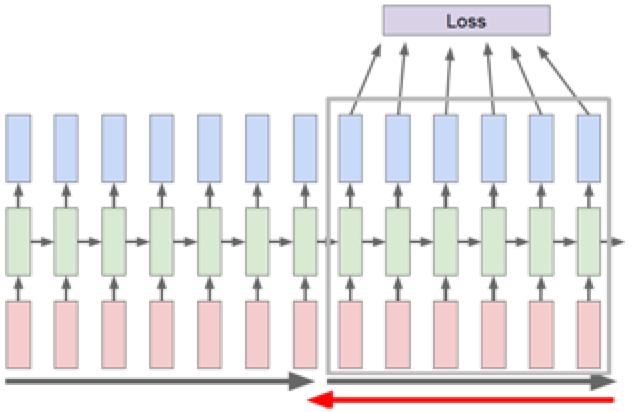


Examples of truncated BPTT algorithm

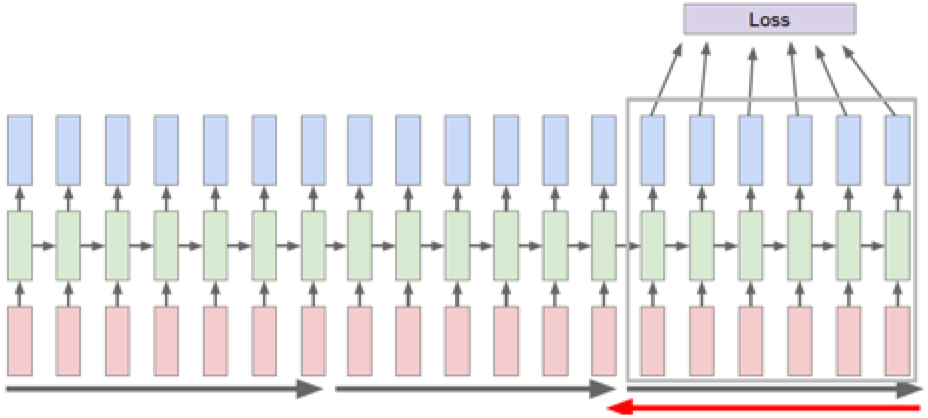
1. TBPTT(n,n): Updates are performed at the end of the sequence across all timesteps in the sequence (e.g. BPTT).
2. TBPTT(1,n): timesteps are processed one at a time followed by an update that covers all timesteps seen so far (e.g. classical TBPTT by Williams and Peng).
3. TBPTT(k1,1): The network likely does not have enough temporal context to learn, relying heavily on internal state and inputs.
4. TBPTT(k1,k2), where k1<k2<n: Multiple updates are performed per sequence which can accelerate training.
5. TBPTT(k1,k2), where k1=k2: A common configuration where a fixed number of timesteps are used for both forward and backward-pass timesteps (e.g. 10s to 100s).



(a)



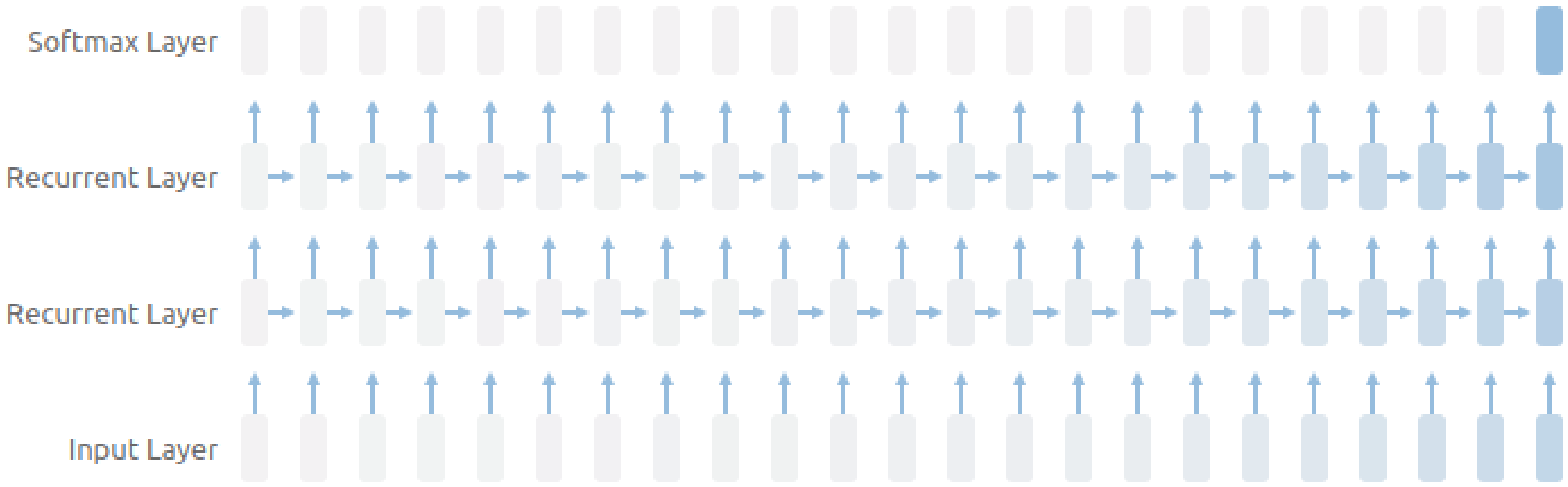
(b)



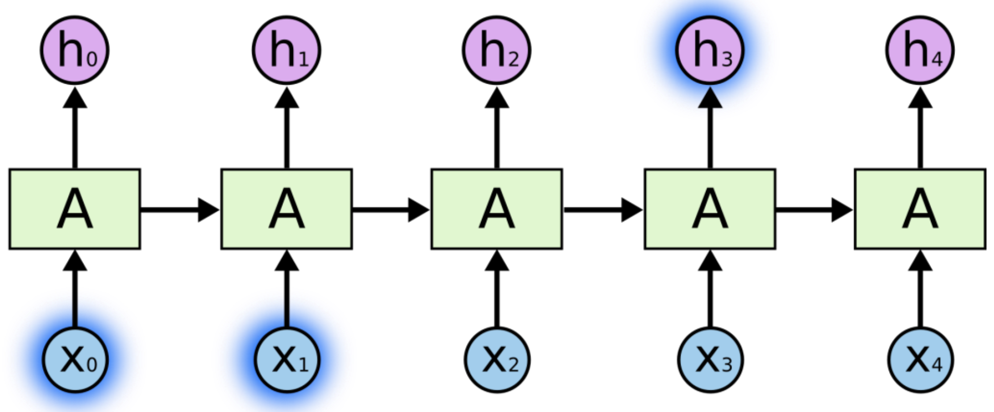
(c)

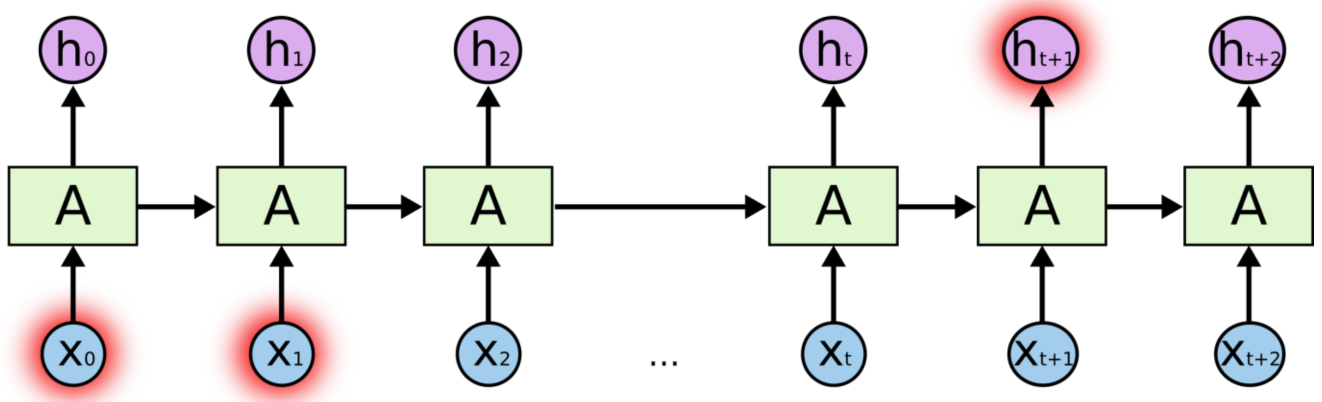
Sequence Modelling | Long Short Term Memory

Vanishing gradient in RNNs



Long-term dependencies

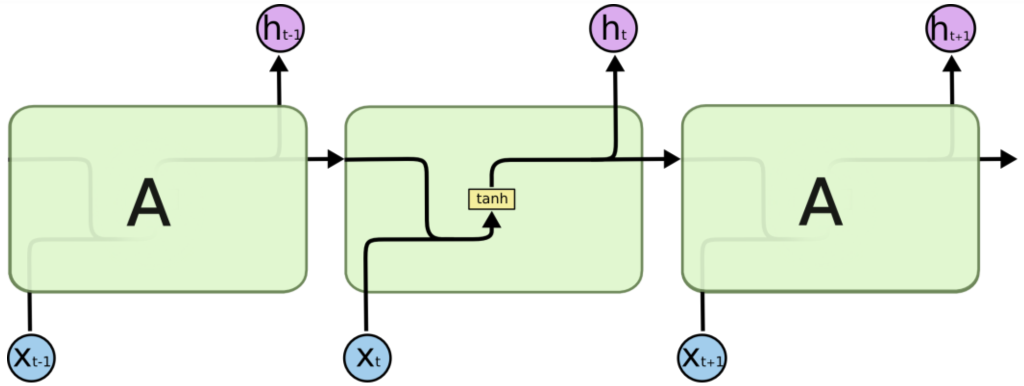


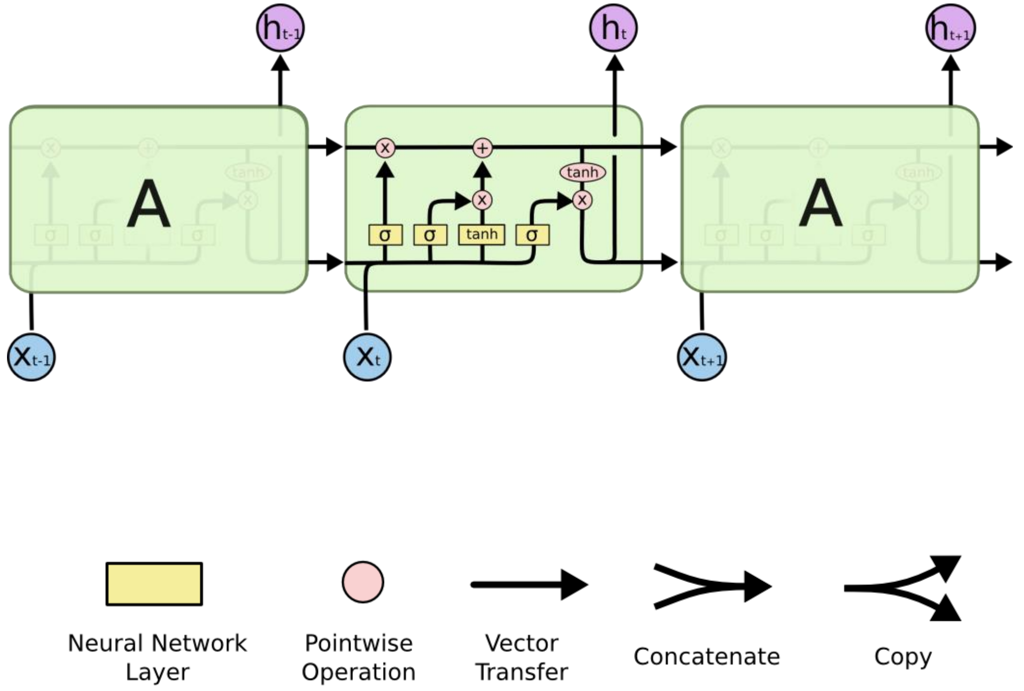


https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM networks

- Forget gate
- Input gate
- State update
- Output gate

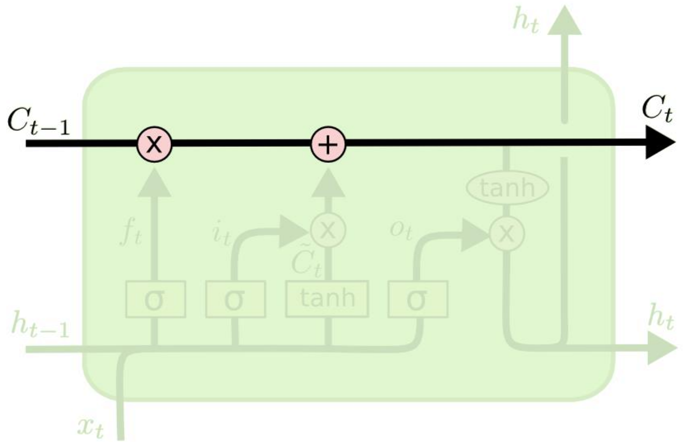




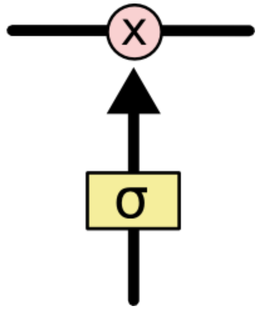
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

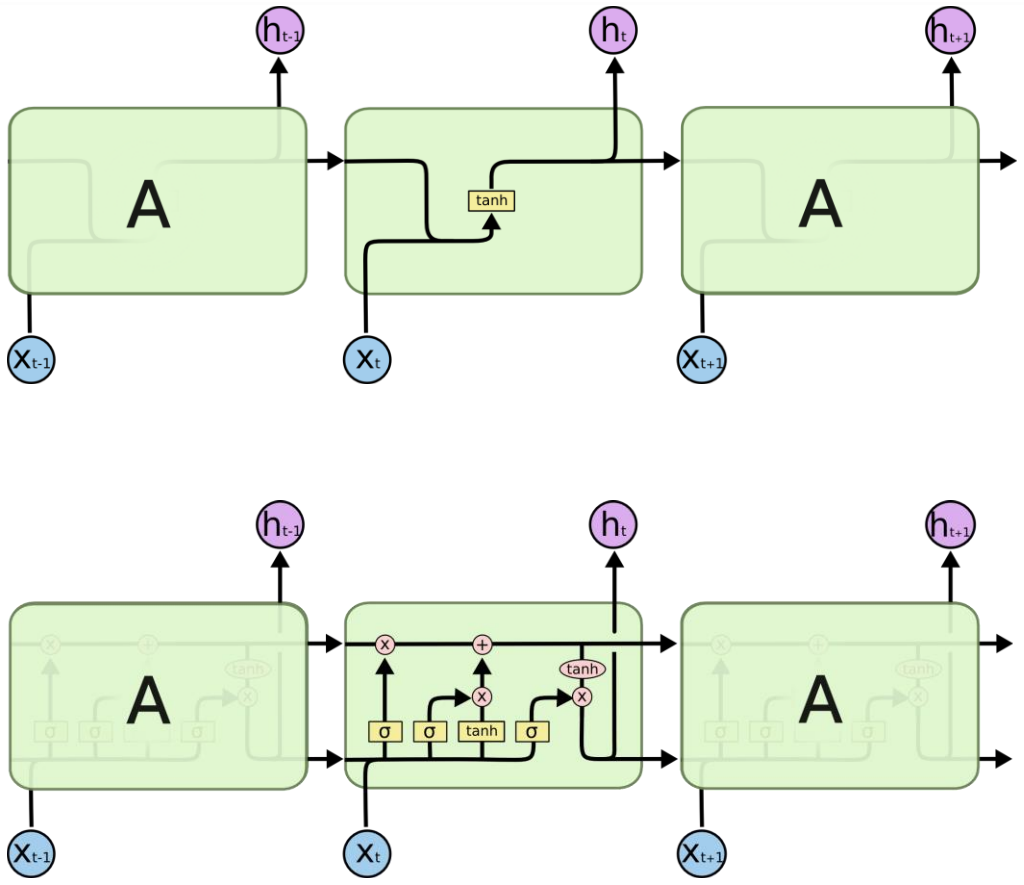
LSTM networks

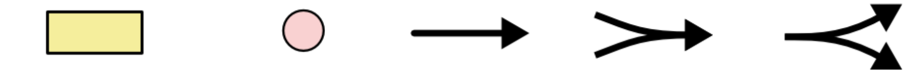
- Cell state



- Gates



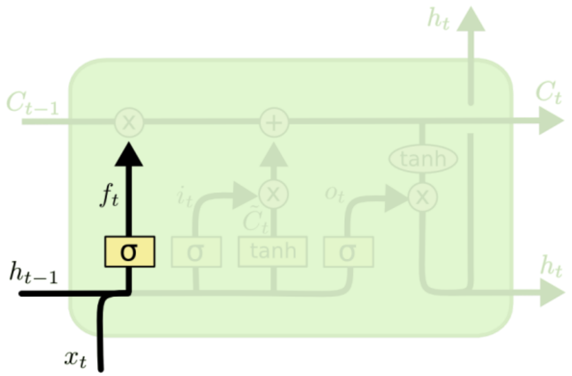




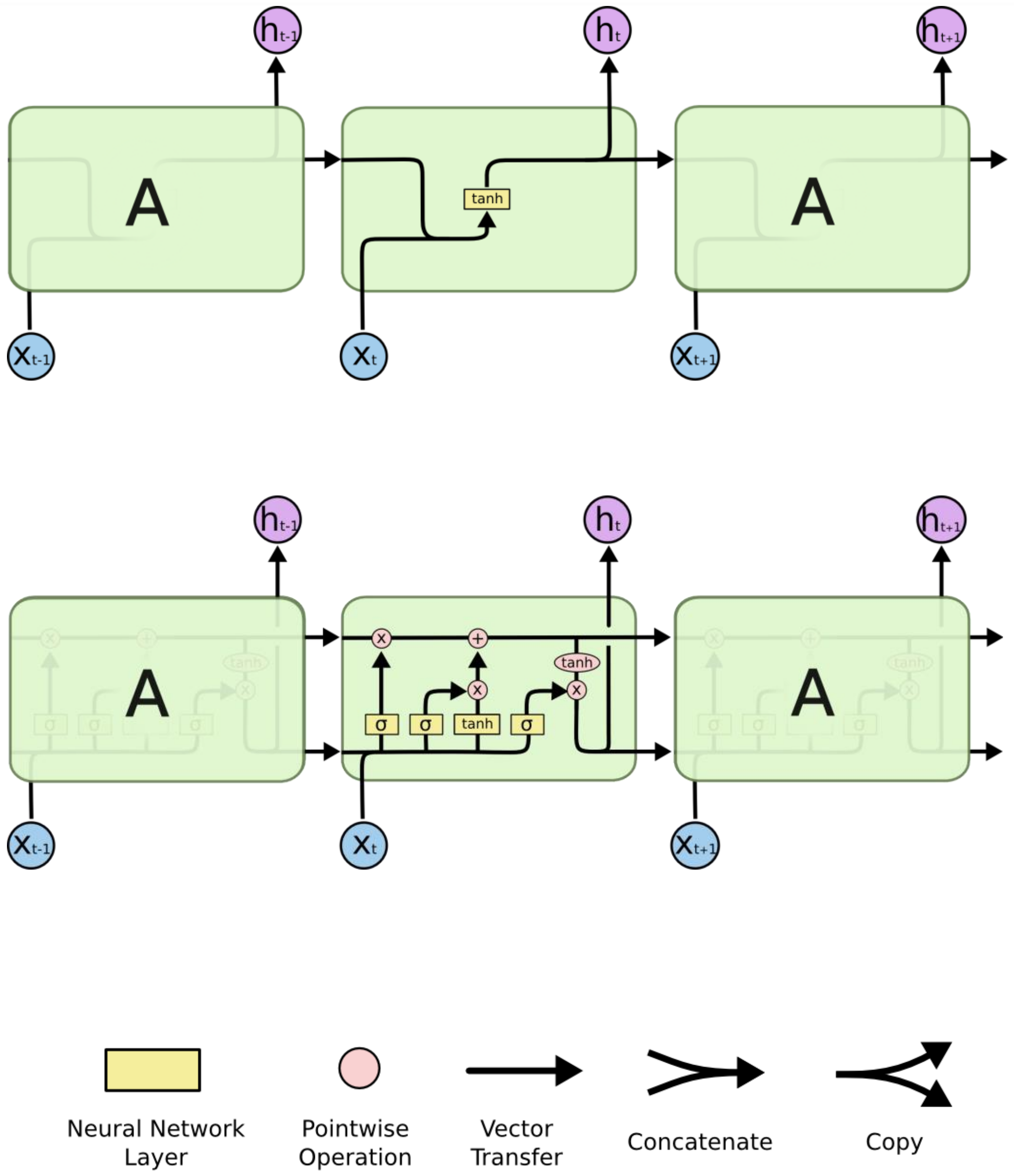
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM networks

– Forget gate



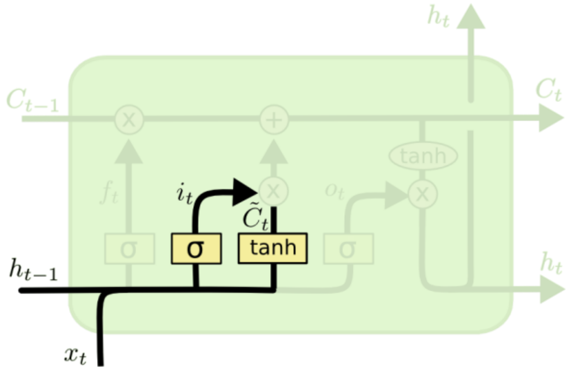
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \;+\; b_f\right)$$



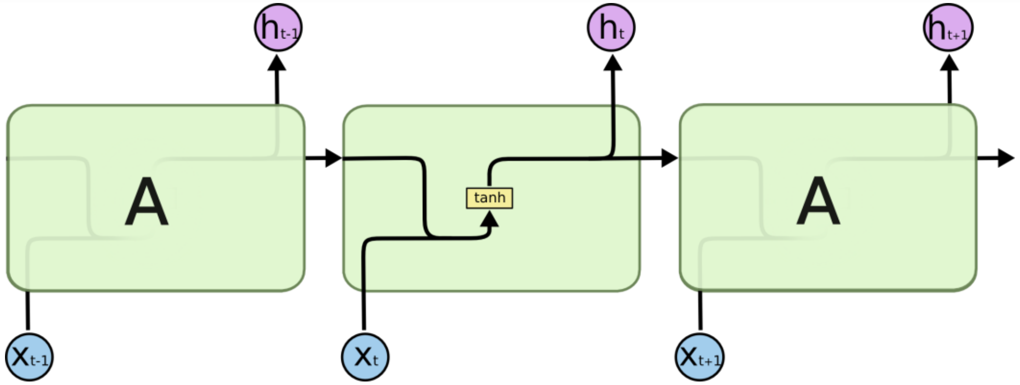
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

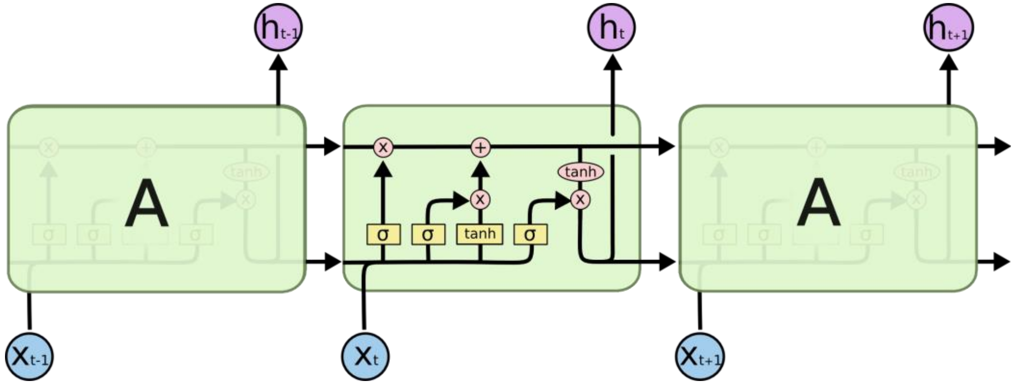
LSTM networks

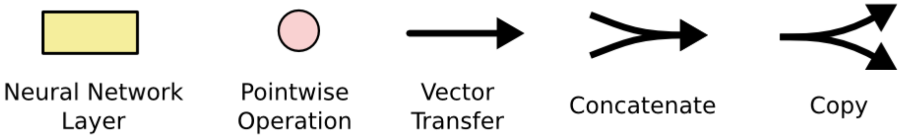
– Input gate



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$



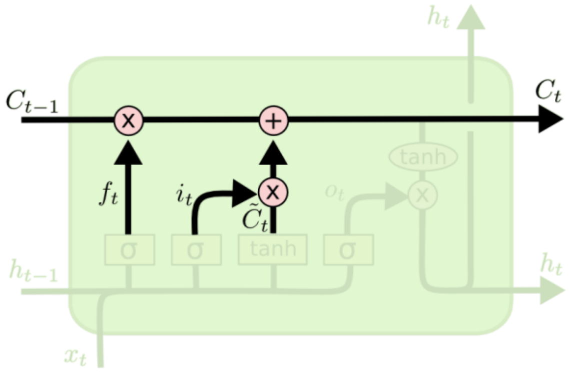




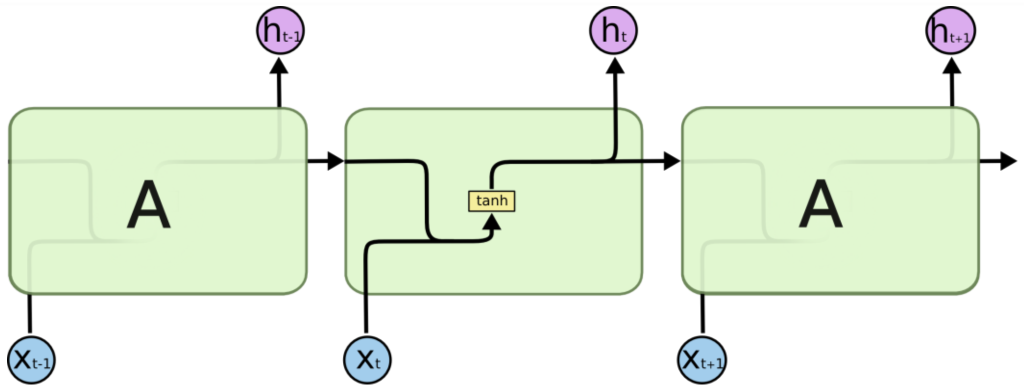
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

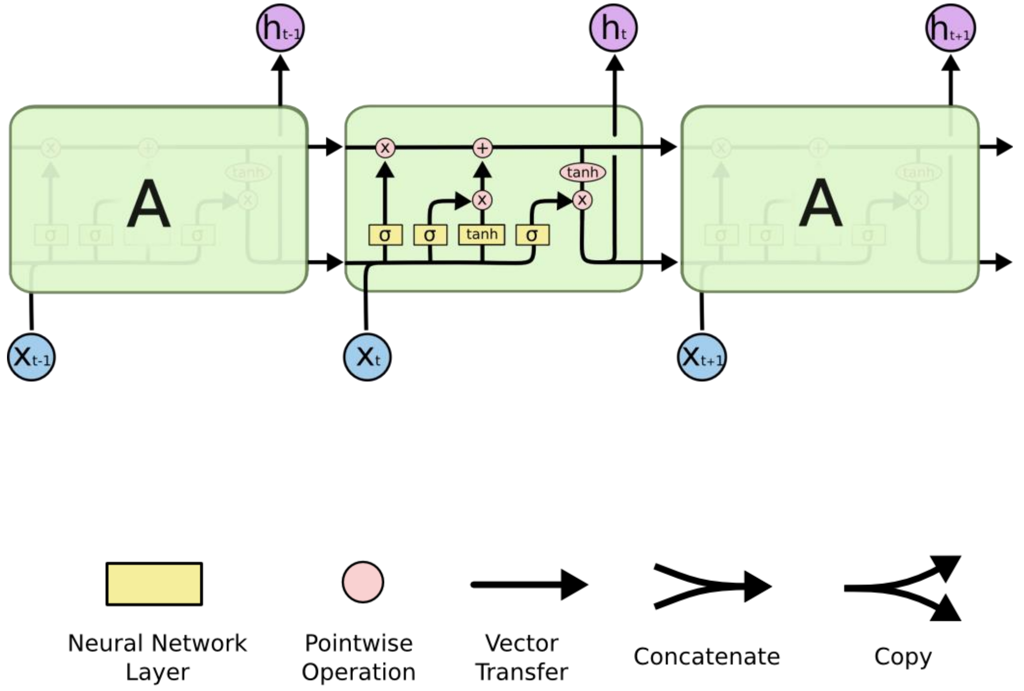
LSTM networks

– State update



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

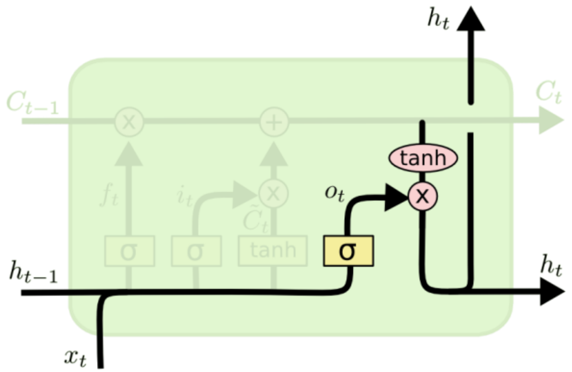




https://colah.github.io/posts/2015-08-Understanding-LSTMs/

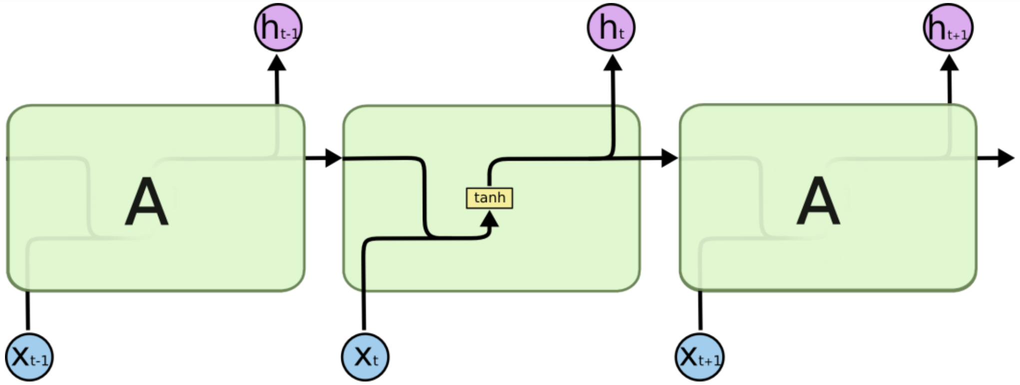
LSTM networks

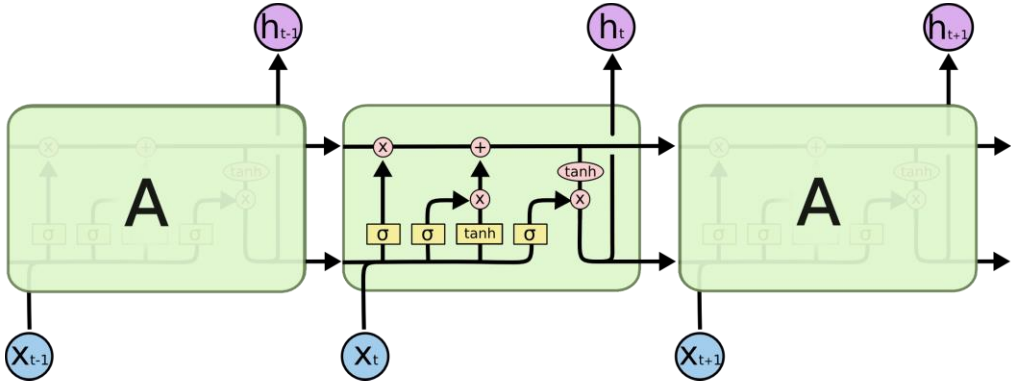
– Output gate

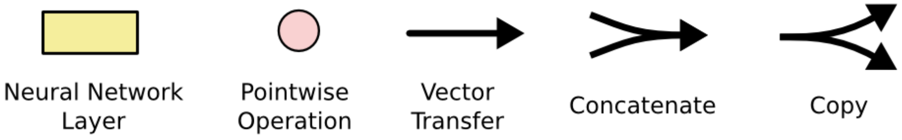


$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

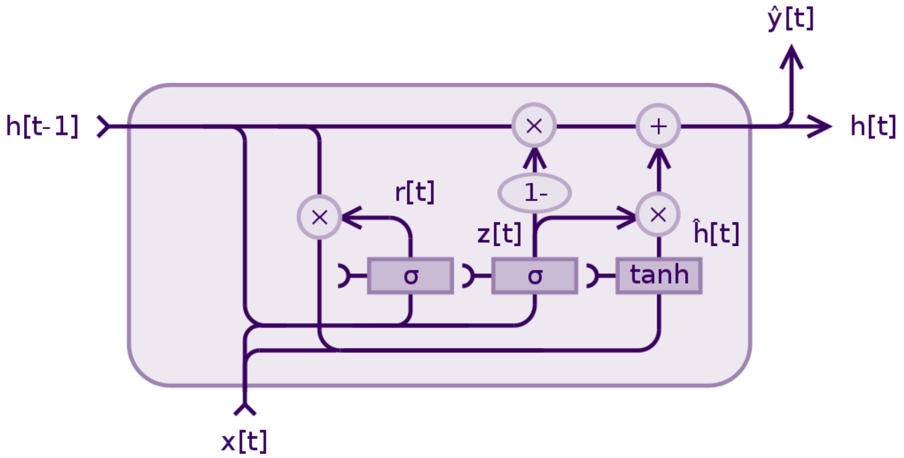
$$h_t = o_t * \tanh\left(C_t\right)$$

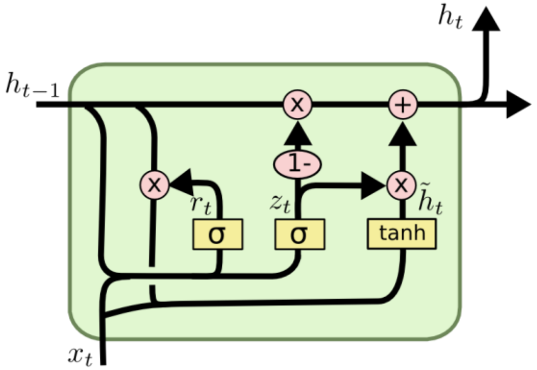






Gated recurrent units (GRUs) and LSTM variants



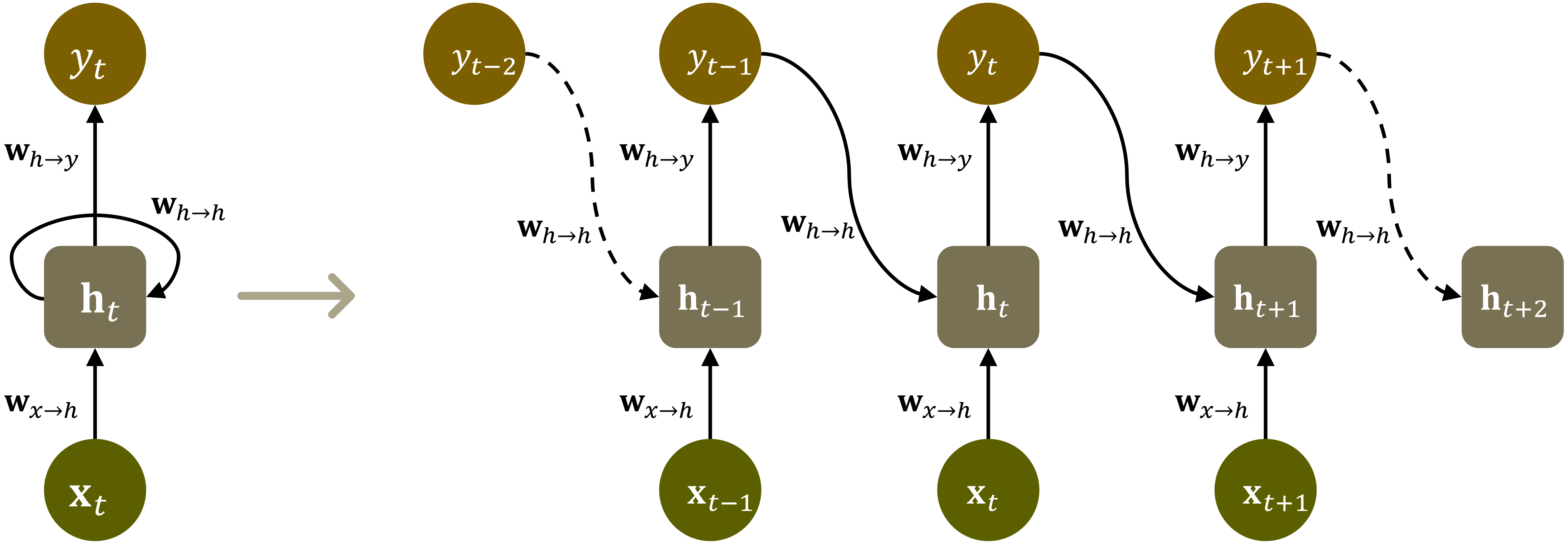


$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

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

Sequence Modelling | Encoder-Decoder

Output recurrence



- Restricted hidden representation – no direct link between hidden units, less capable but parallelisable*
- Teacher forcing – recurrent output

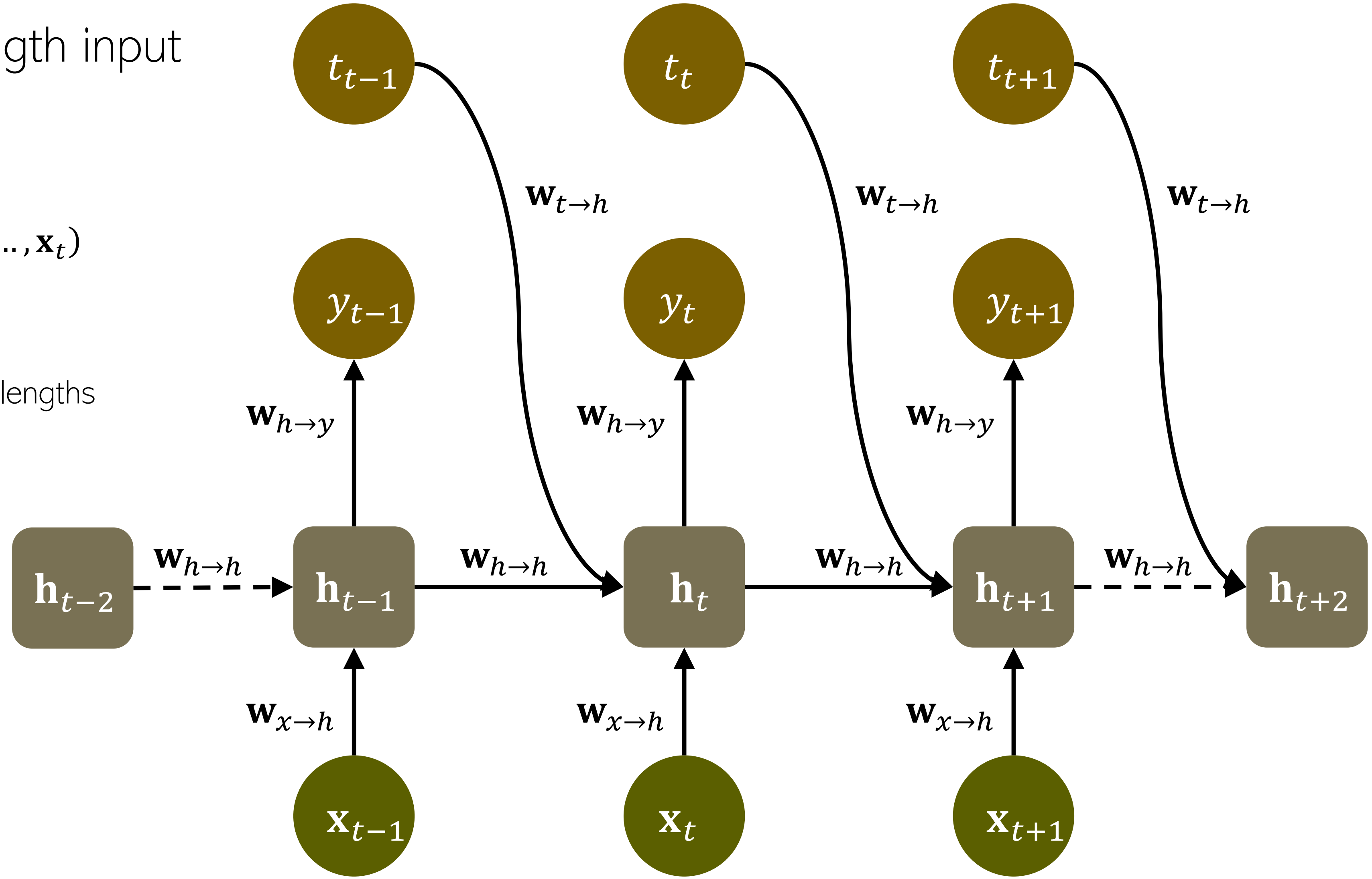
Input as context

Conditioned on variable-length input

– Conditional modelling of target $\boldsymbol{t}$

$$p(t_1, \dots, t_T | \mathbf{x}_1, \dots, \mathbf{x}_t) = \prod_t p(t_t \mid \mathbf{x}_1, \dots, \mathbf{x}_t)$$

– Variable but equal input and output lengths



Input as context

Conditioned on fixed-length input $\mathbf{x}$

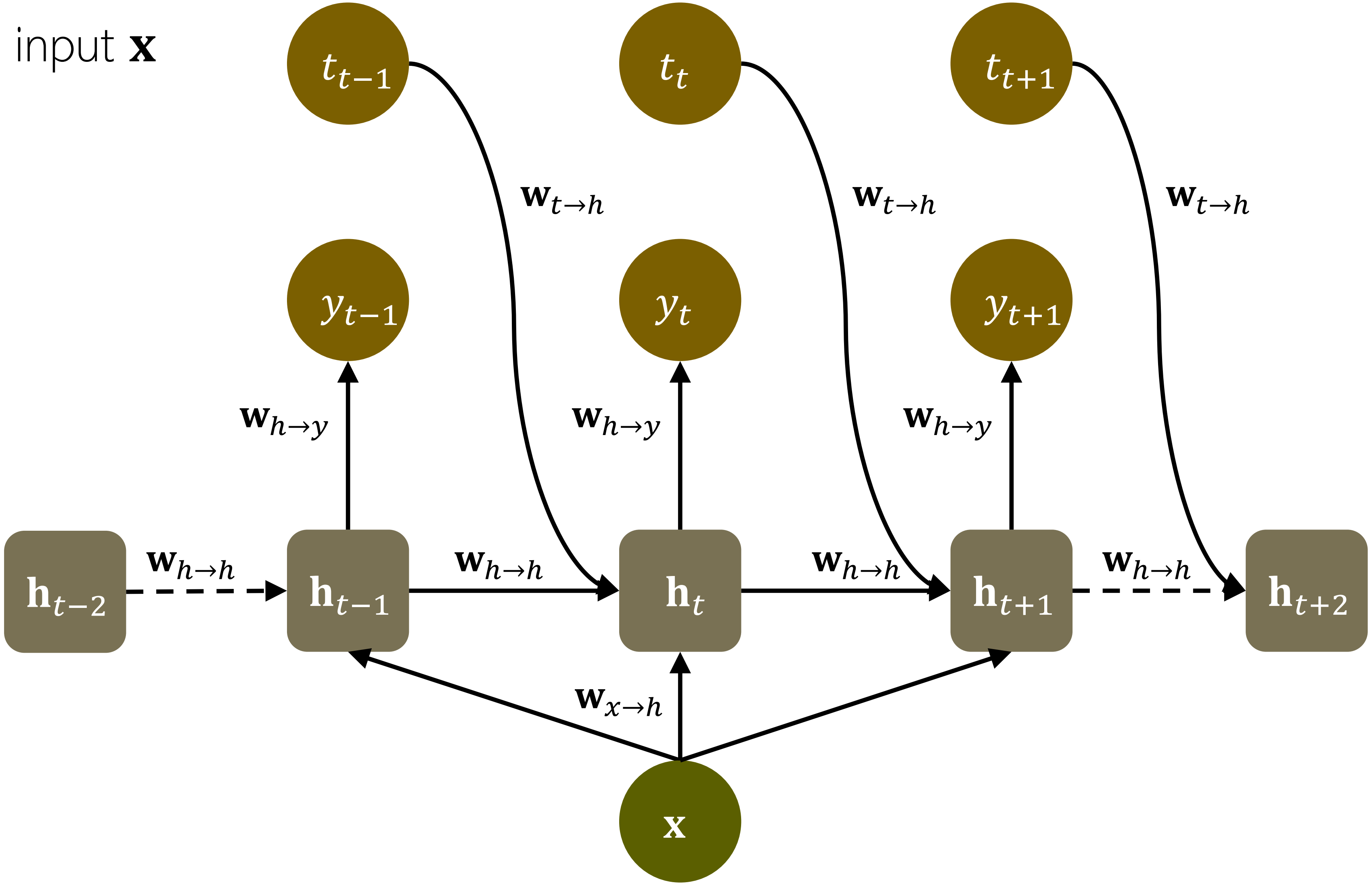
– Conditional modelling of target $\boldsymbol{t}$

$$p(t_1, \dots, t_T | \mathbf{x}) = \prod_t p(t_t \mid \mathbf{x})$$

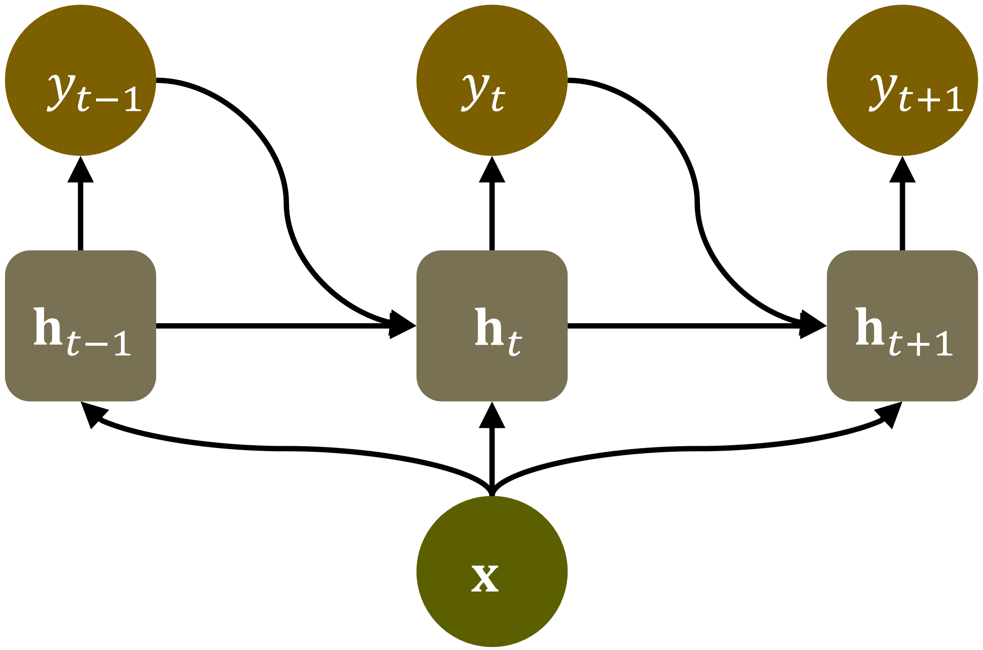
– Conditional independence given $\mathbf{x}$

Thus, add connections $\mathbf{w}_{t \to h}$

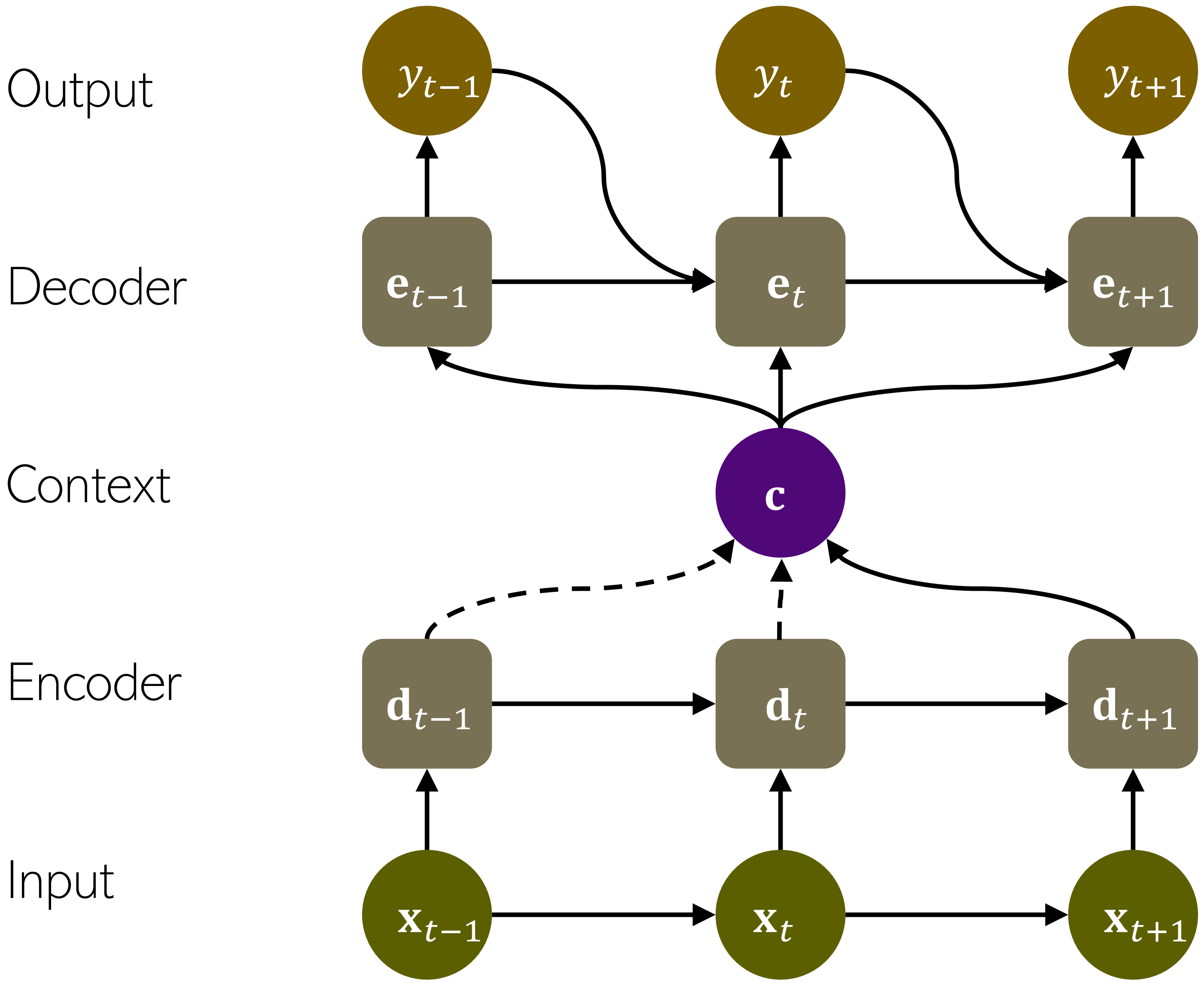
– e.g. seq2seq*, attention*



Sequence-to-sequence with variable lengths

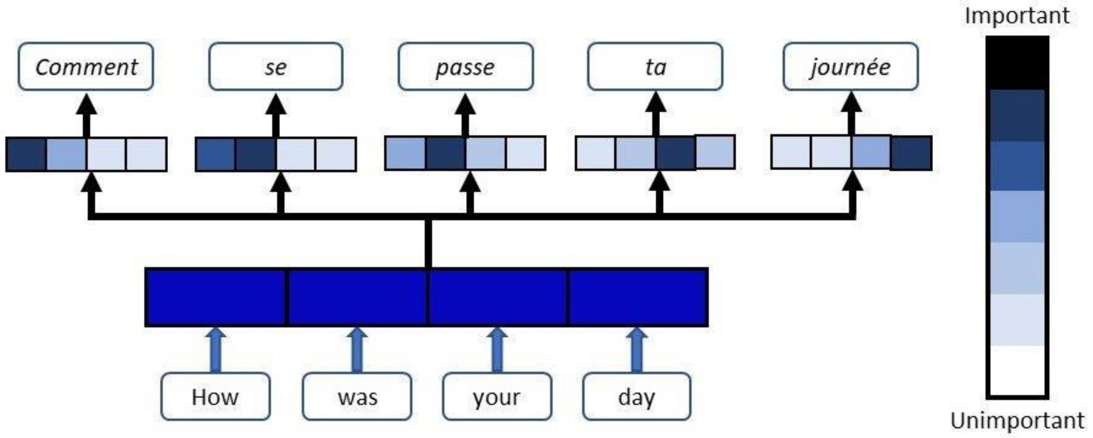


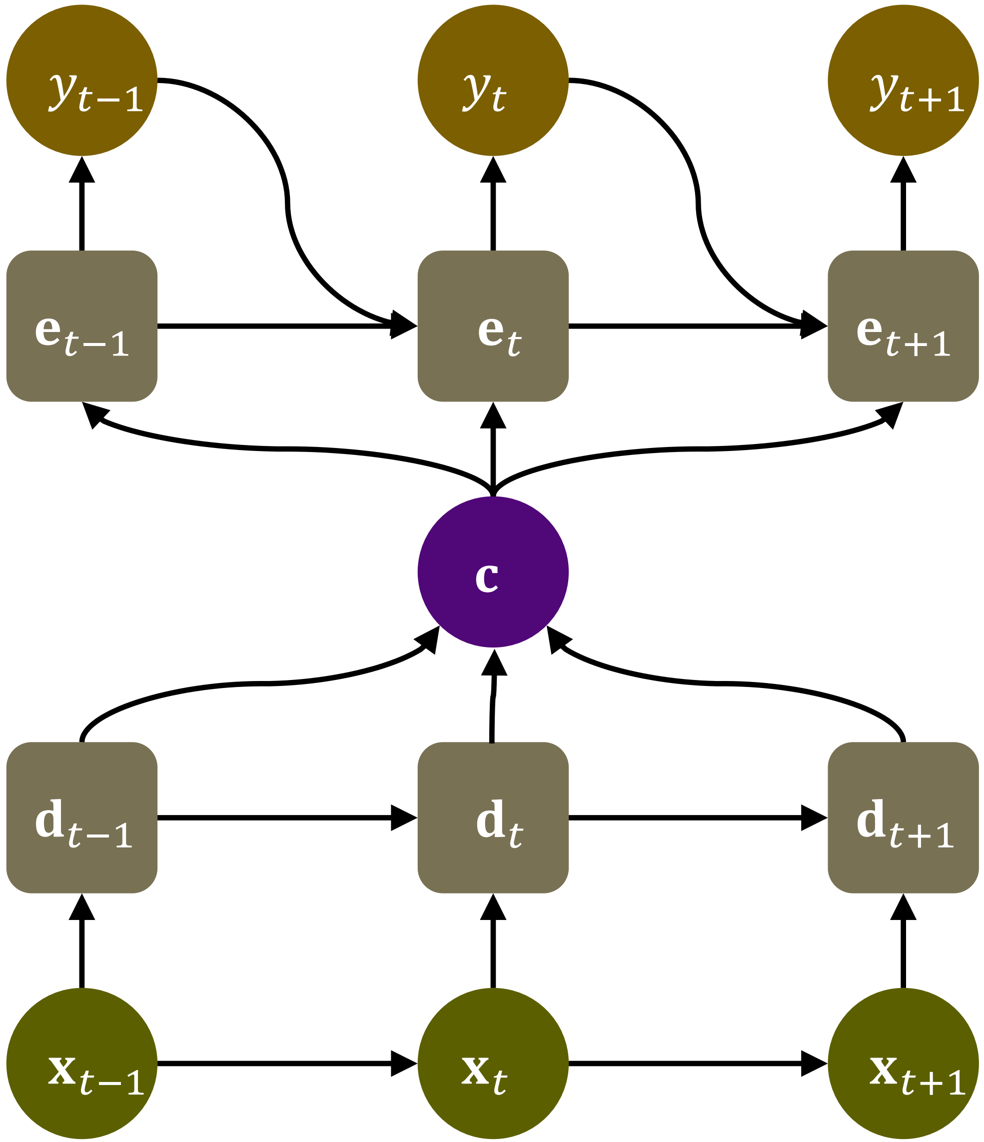
Sequence-to-sequence with variable lengths



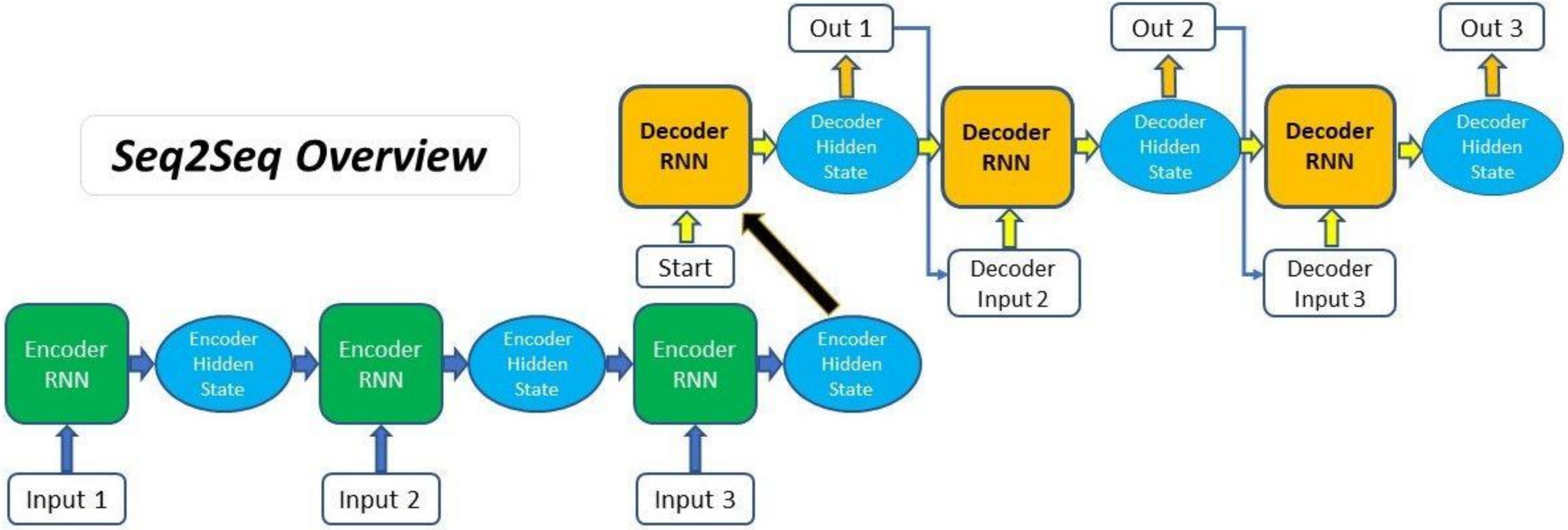
Learnable weights on context vector

(Self-)Attention mechanism





Seq2Seq architecture



Transformer

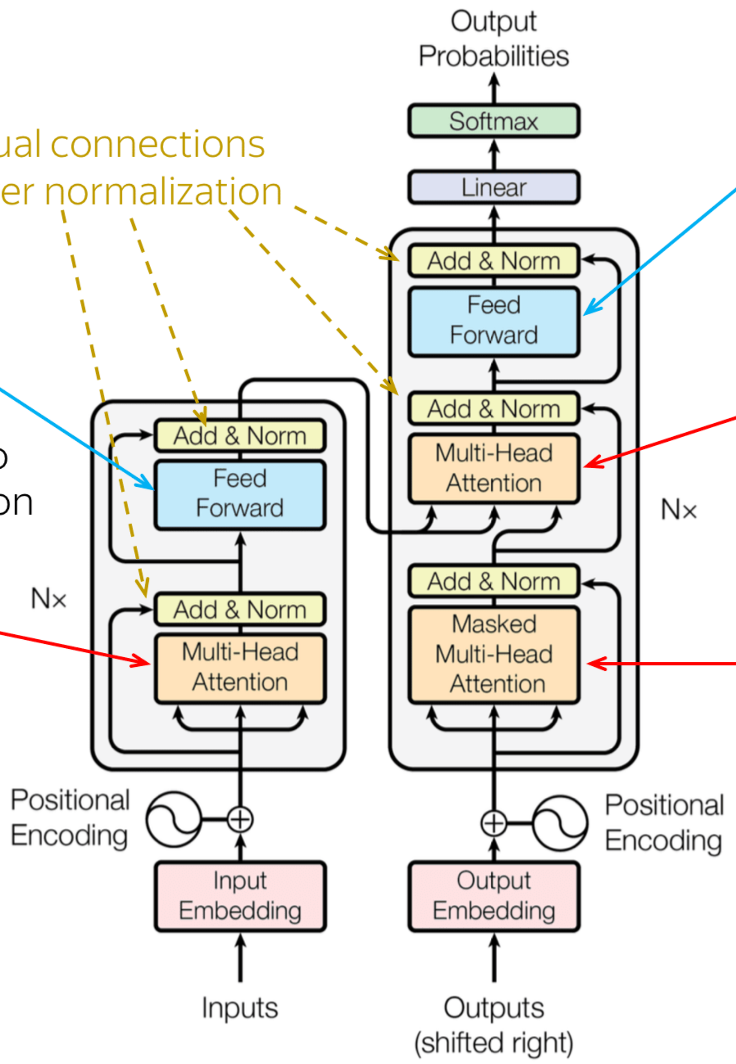
Without sequence-aligned RNNs

Residual connections and layer normalization

Feed-forward network: after taking information from other tokens, take a moment to think and process this information

Encoder self-attention: tokens look at each other

queries, keys, values are computed from encoder states



Feed-forward network: after taking information from other tokens, take a moment to think and process this information

Decoder-encoder attention: target token looks at the source

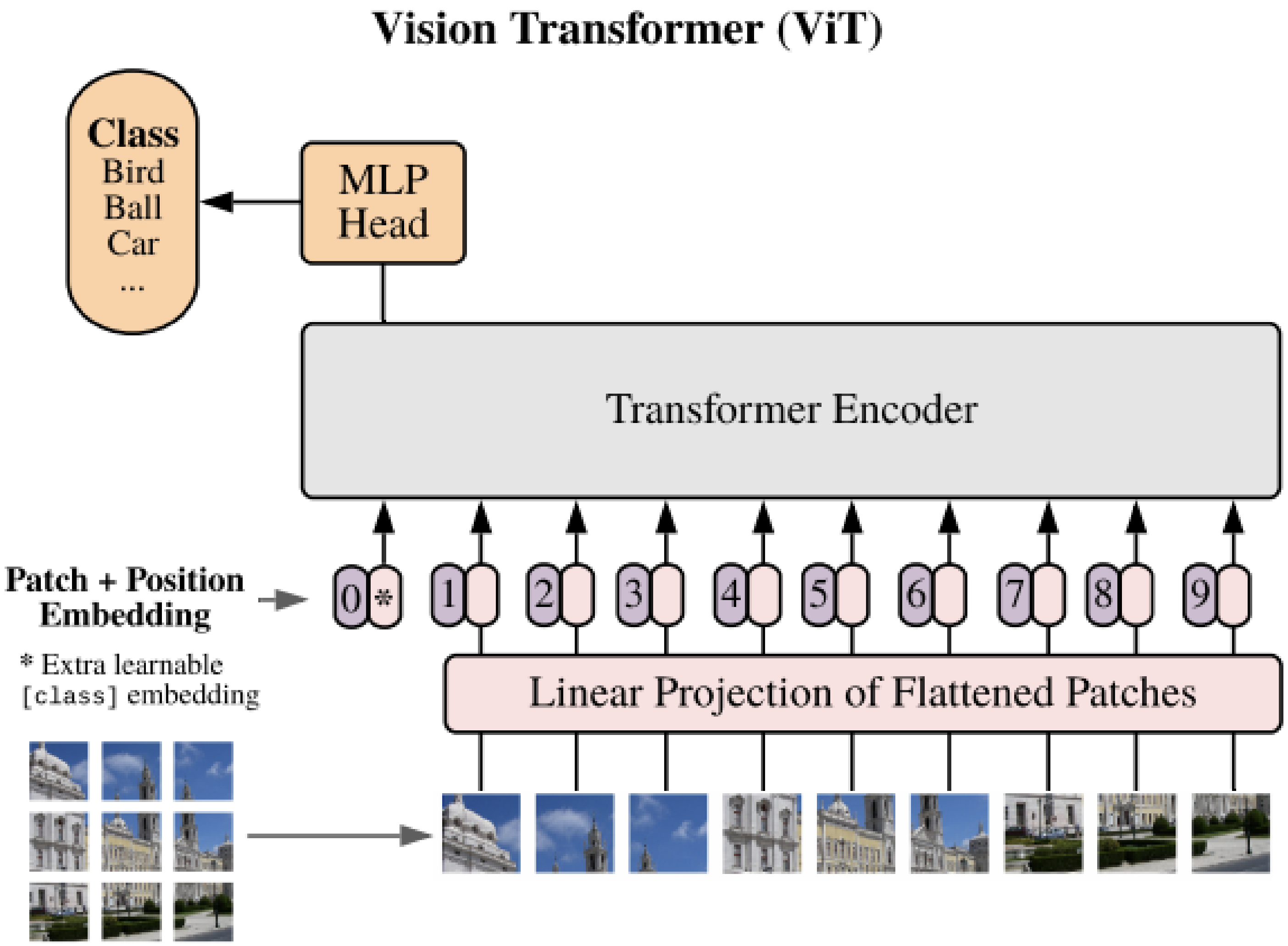
queries – from decoder states; keys and values from encoder states

Decoder self-attention (masked): tokens look at the previous tokens

queries, keys, values are computed from decoder states

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008).

Vision transformer



Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S. and Uszkoreit, J., 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Change the RNN in the "text classification" tutorial to a CNN
Compare the model performance