Department of Computer Science

UCL

COMP0197: Applied Deep Learning

Yipeng Hu
yipeng.hu@ucl.ac.uk

Sequence Modelling

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

time $\longrightarrow$

x_1 x_2 x_3 x_4 x_5 ……

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

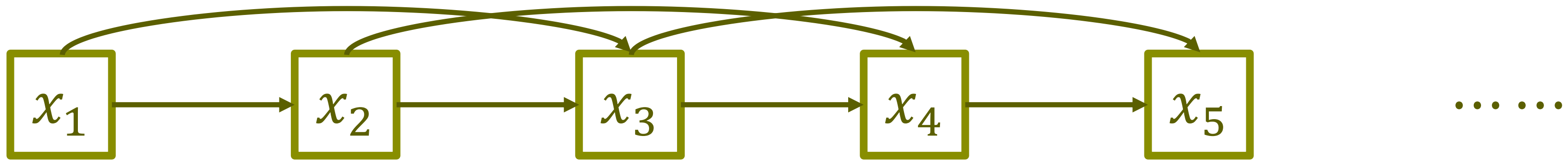
$$p(x_1, \cdots x_N) = \prod_{n=1}^{N} p(x_n)$$

The first-order Markov model

$$x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_4 \rightarrow x_5 \qquad \cdots\cdots$$

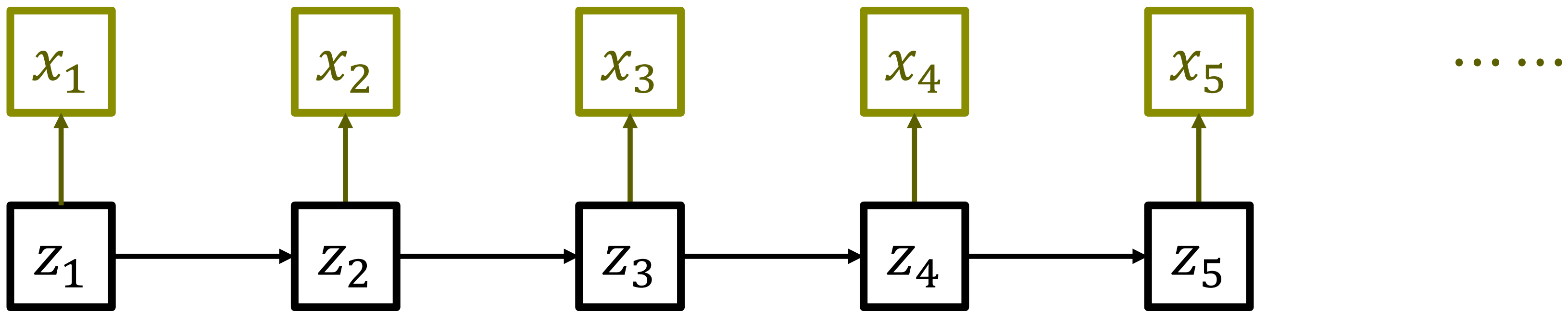
$$p(x_1, \cdots x_N) = p(x_1) \prod_{n=2}^{N} p(x_n | x_{n-1})$$

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,

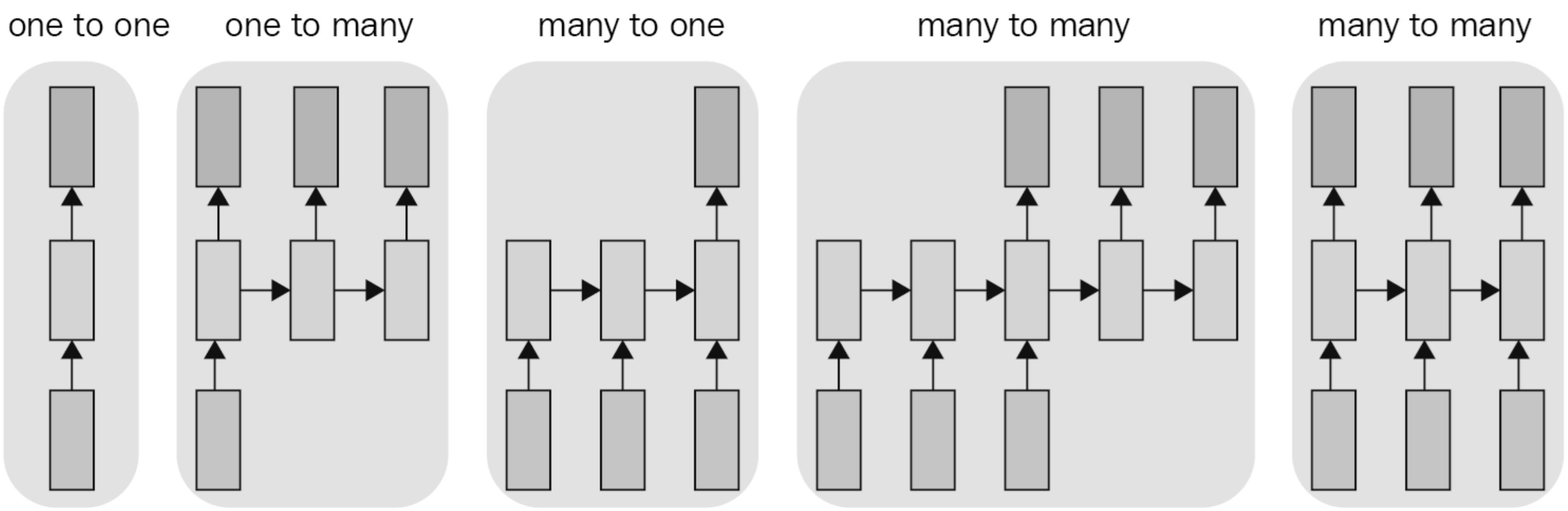
- Backpropagation through time
- Long short term memory
- Encoder-decoder
- Attention mechanism

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

Word-based language models

- Modelling sequences of tokens (phrases, words, characters)

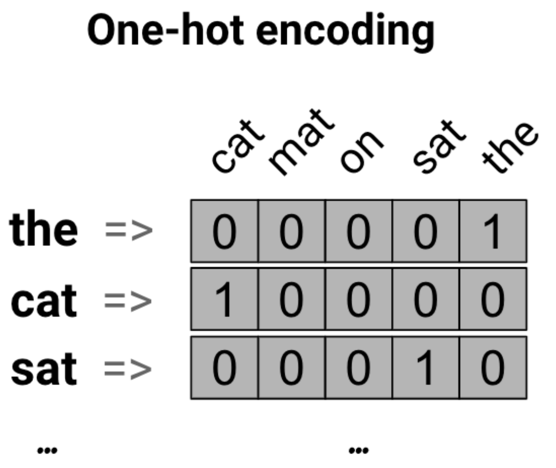
1. Standardisation before tokenisation, e.g. lowercasing, punctuation stripping
2. Tokenisation
   - Splitting into substring, oft. Words
   - *Combining consecutive substrings, oft. n-grams
   - Indexing tokens, e.g. integers with a vocabulary
   - *Clustering similar words, i.e. class-based language models
3. Transforming each (string) example into a (oft. pad-to-max-tokens) vector of token indices

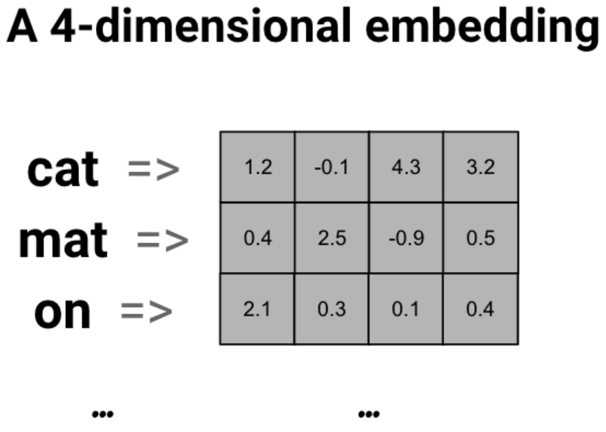
- Predefined or data-defined (adapted) vocabulary
- Integer or dense representation*
- Why words not letters?

Natural 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
  - Vocabulary size and dimensionality, oft. 8-1024 trainable weights





Word embeddings

N-Grams

- A bigram: “introduction to”
- A trigram: “introduction to deep”
- A 4-gram: “introduction to deep learning”

– The chain rule of probability

$$p(w_1, w_2, \cdots) = p(w_1, \cdots, w_i) \prod_i p(w_i | w_{i-1}, w_{i-2}, \cdots w_{i-n+1})$$

– Conditional probabilities, given previous “context” words

– Maximise the log-probability – encoding most frequent word co-occurrence

– Combining N-Gram, e.g. ensemble / additional input

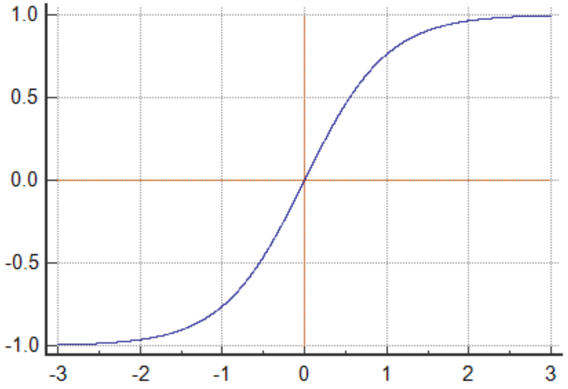
Word representation is itself a field of study, crucial in real-world applications.

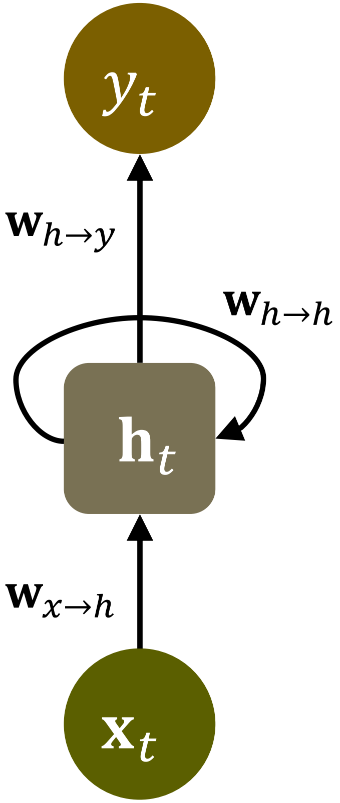
Sequence Modelling | Recurrent Neural Networks

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

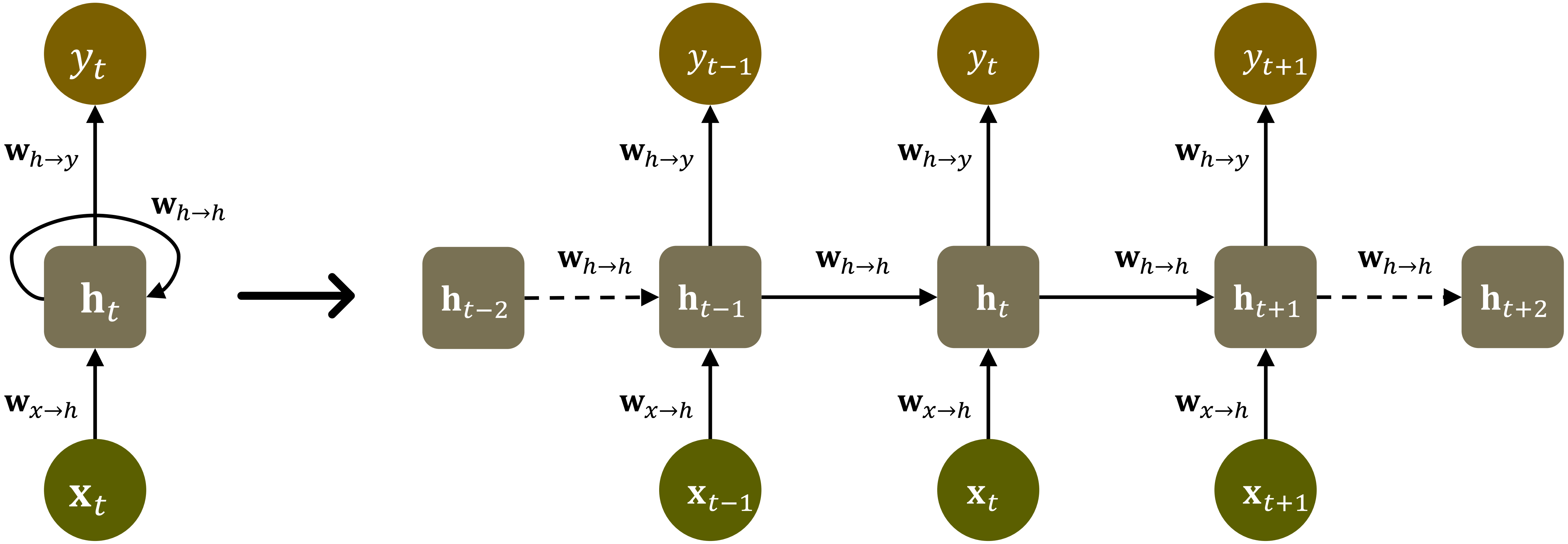




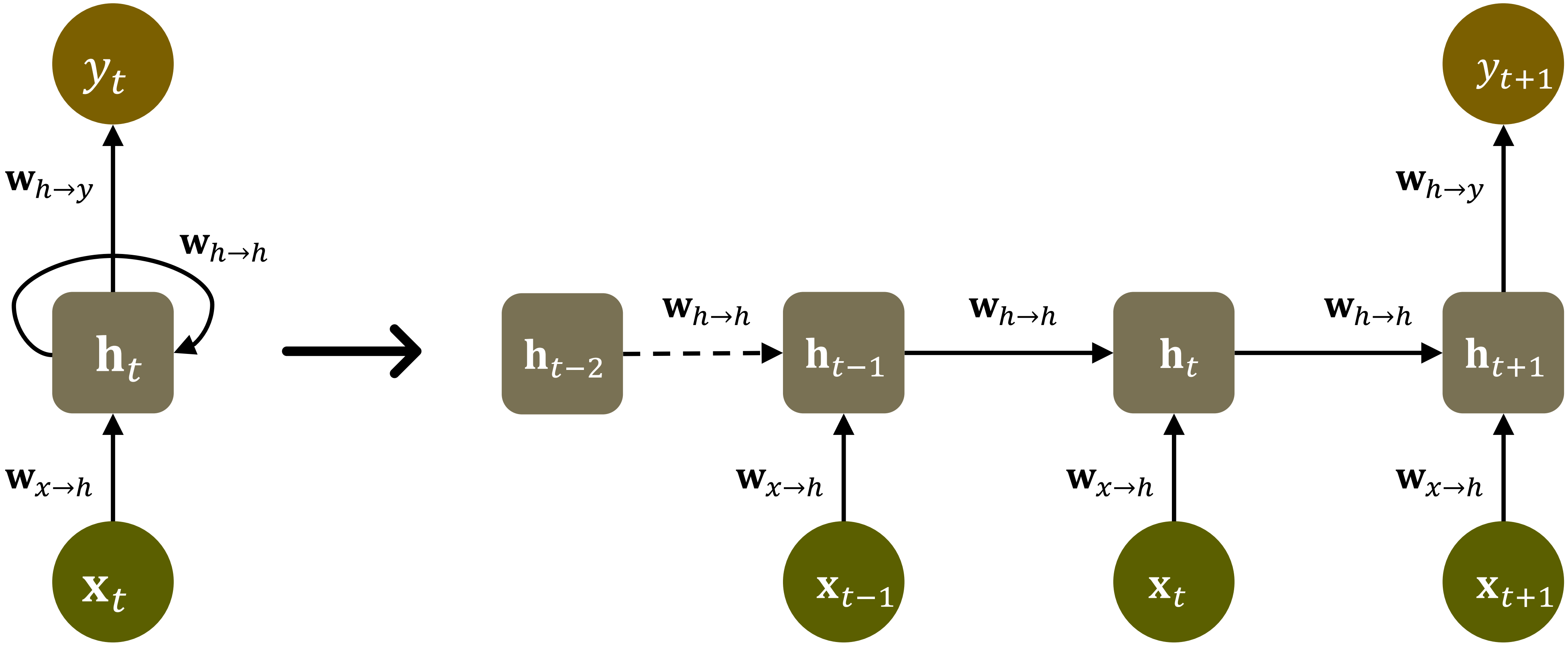
- Output - $y_t = g\left(\mathbf{w}_{h\to y}^{\mathrm{T}}\mathbf{h}_t\right)$

Choice of $g(\cdot)$?

Unfolding / unrolling

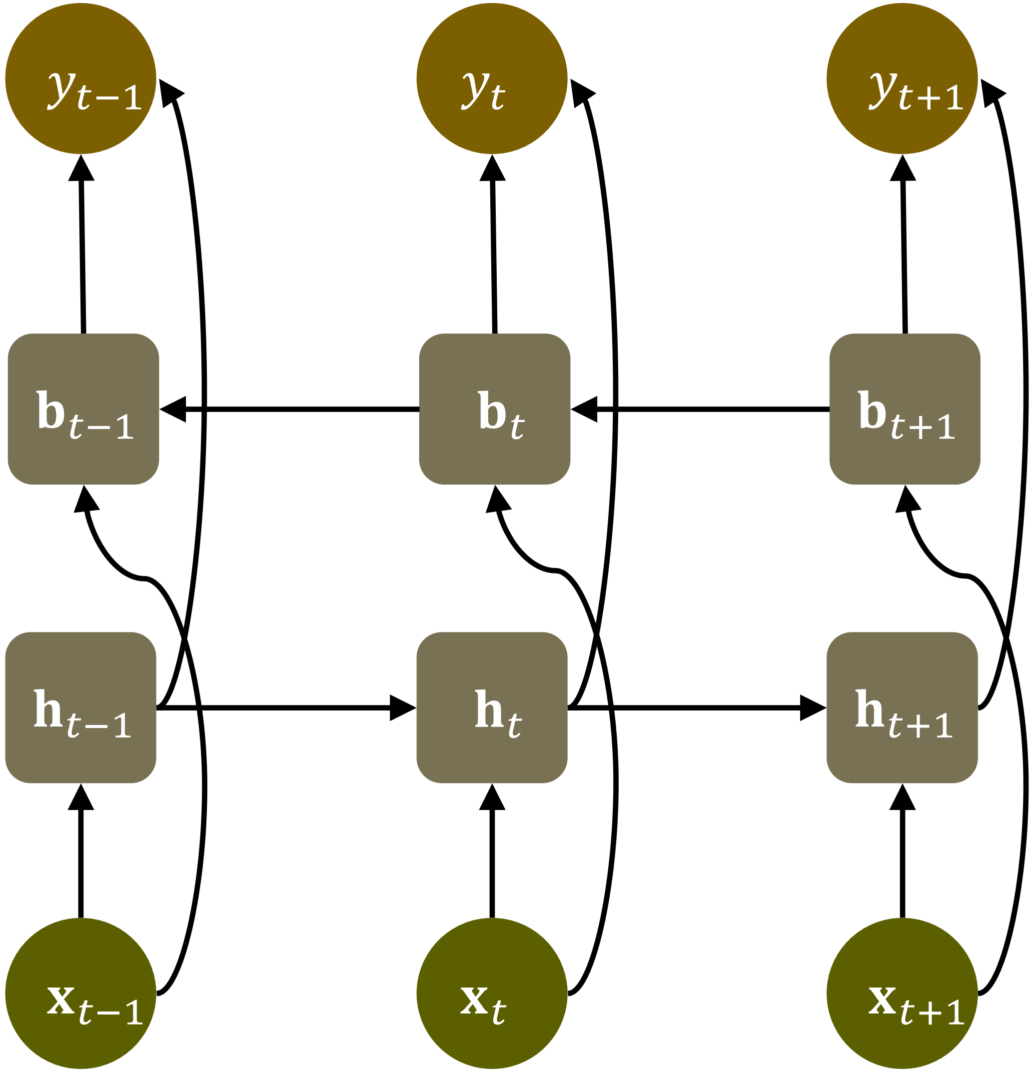


Single output



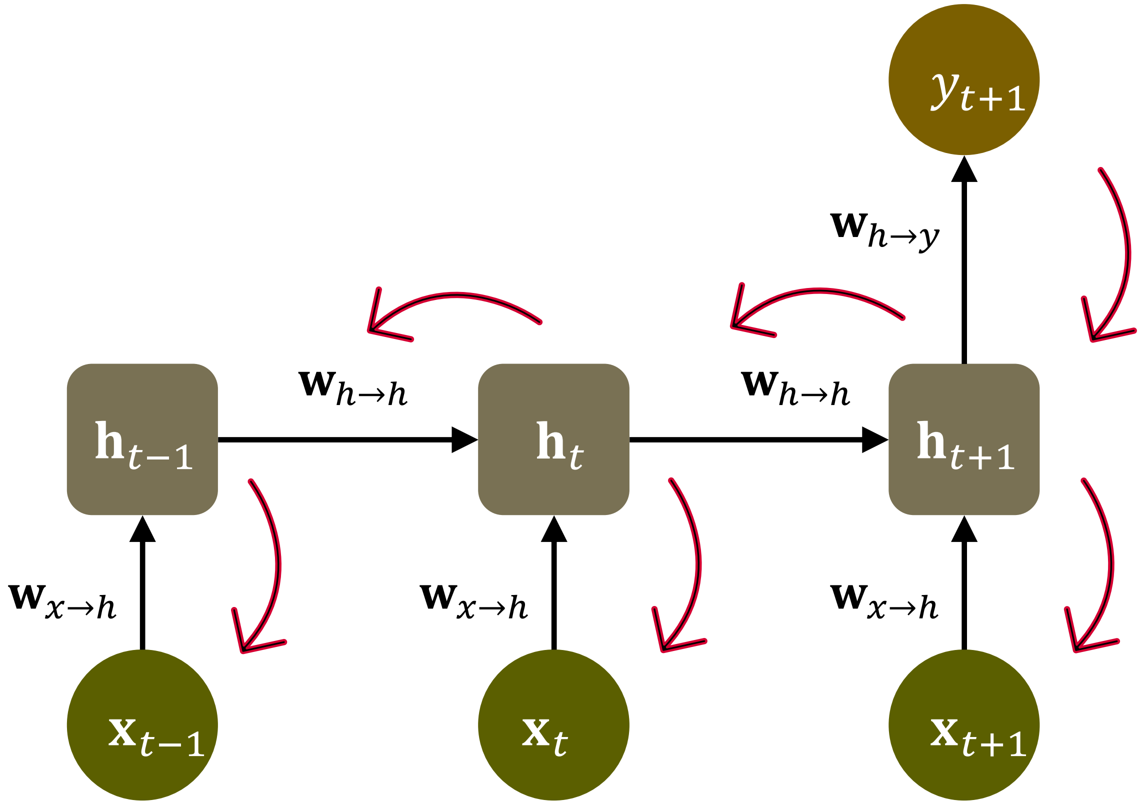
Bidirectional RNNs

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

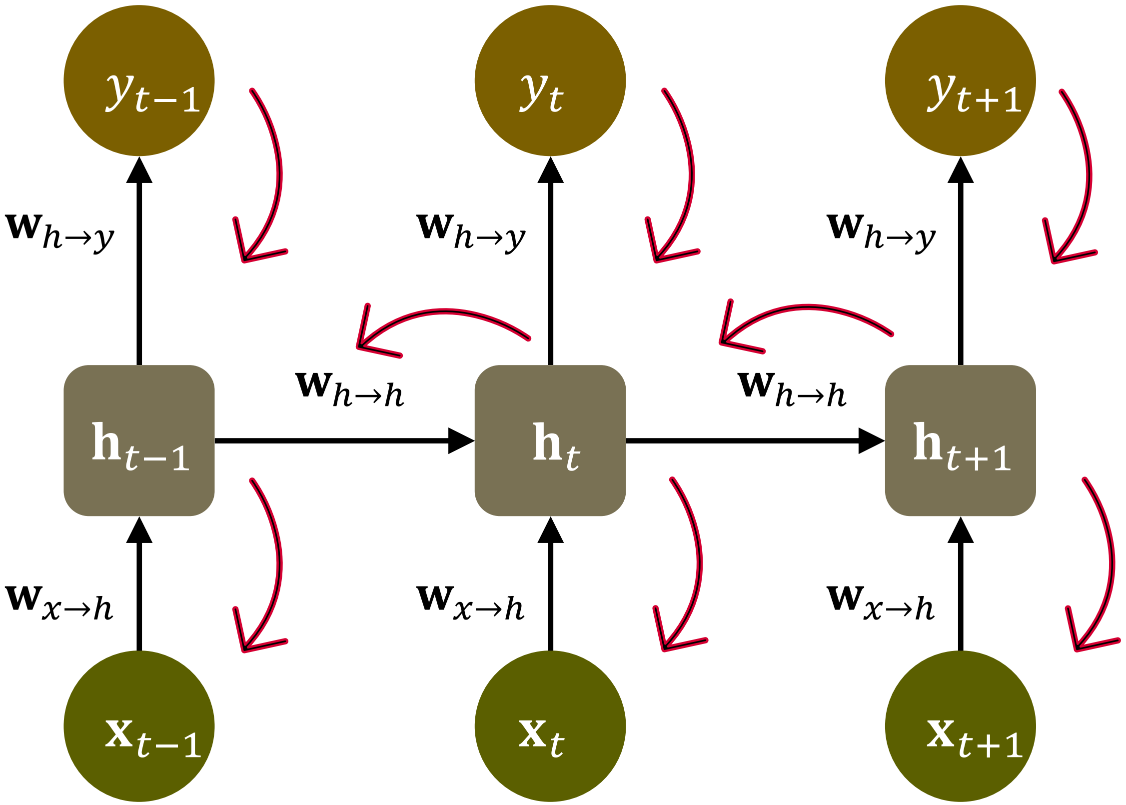


Sequence Modelling | Backpropagation Through Time

Single output

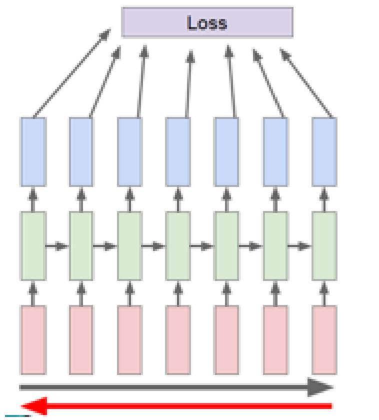


Variable-length output



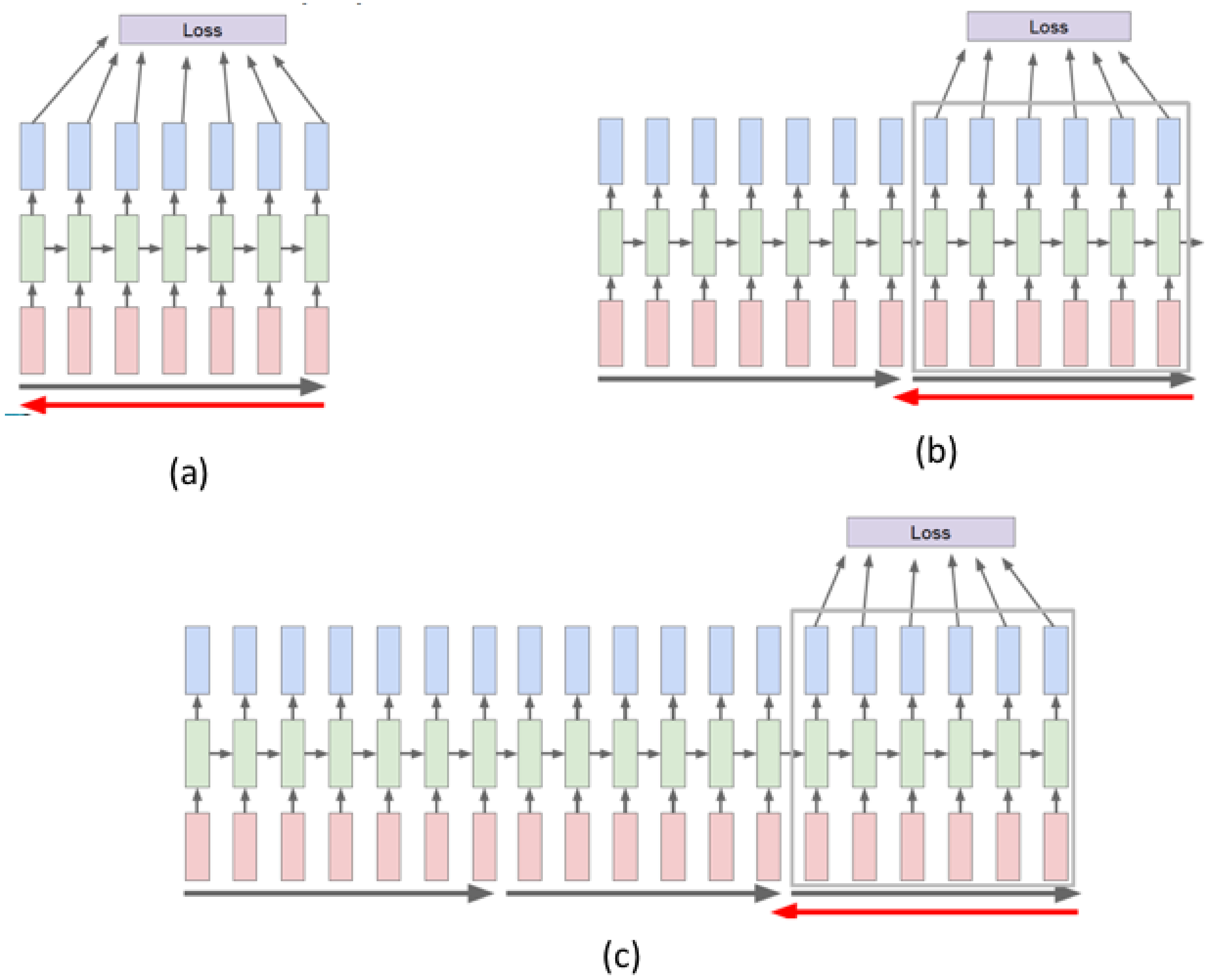
BPTT algorithm

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



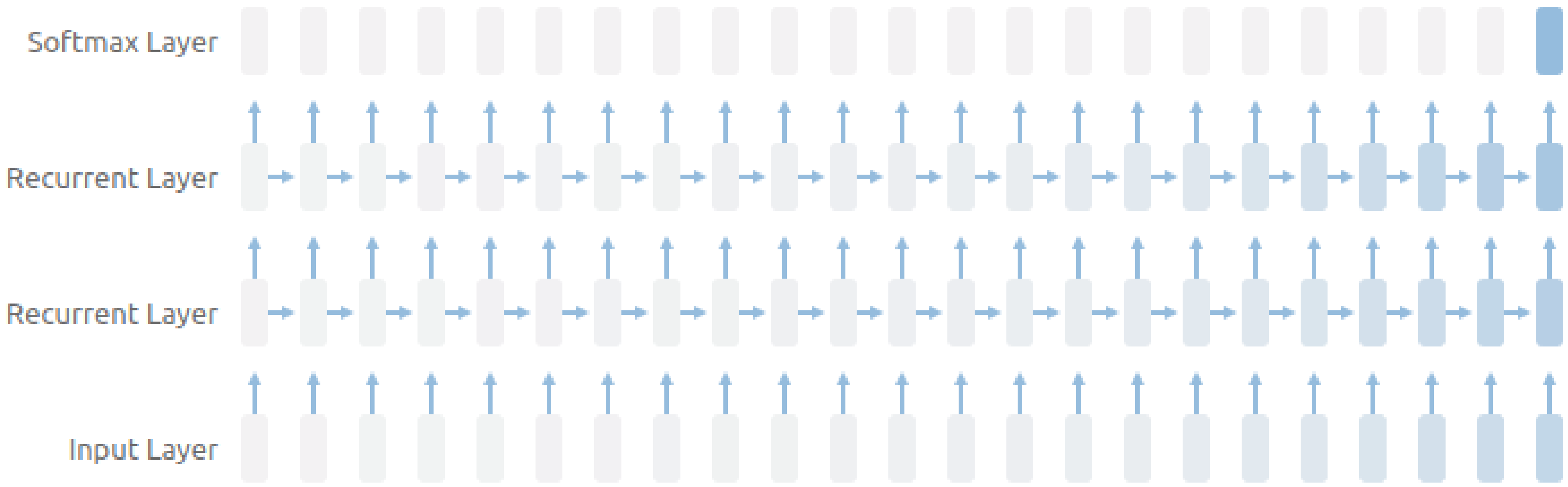
Examples of truncated BPTT algorithm

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).

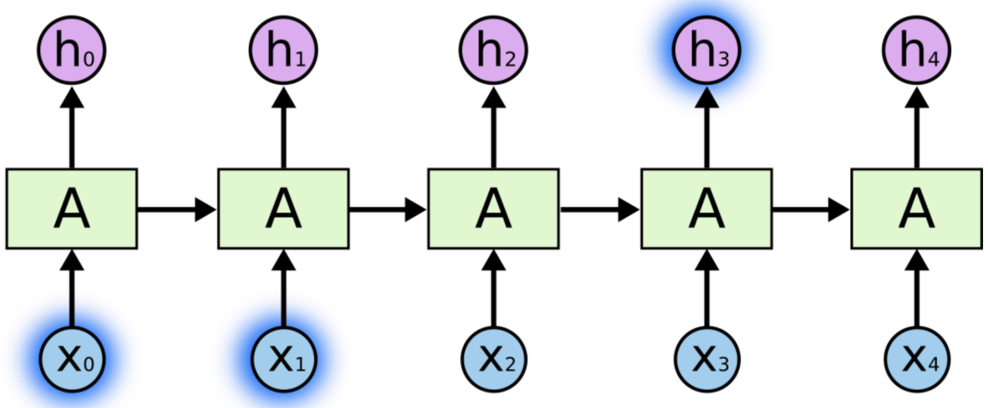


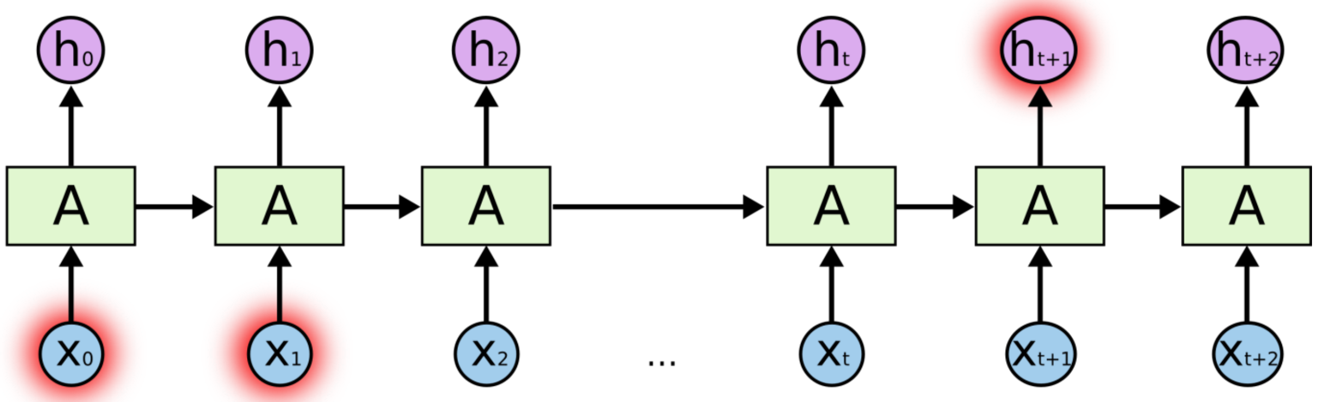
Sequence Modelling | Long Short Term Memory

Vanishing gradient in RNNs



Long-term dependencies



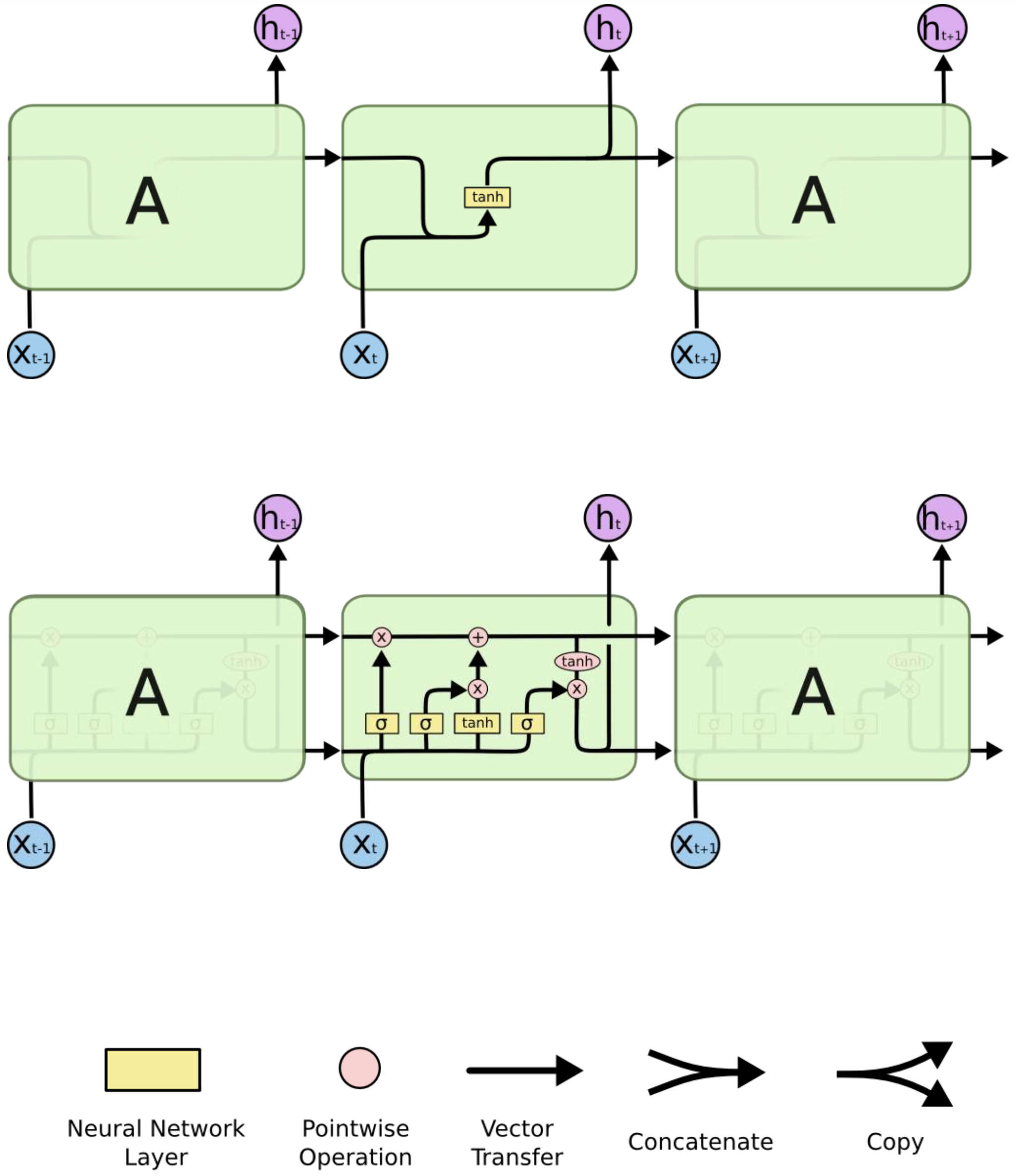


https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Sequence Modelling | Long Short Term Memory

LSTM networks

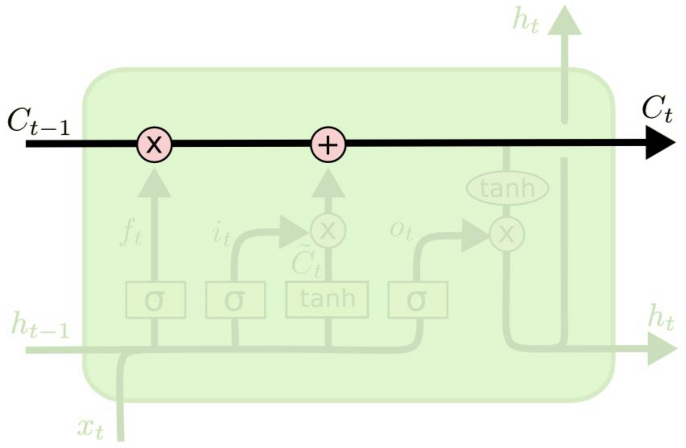
- Forget gate
- Input gate
- State update
- Output gate



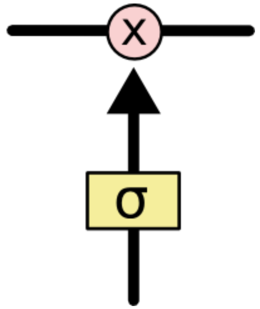
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

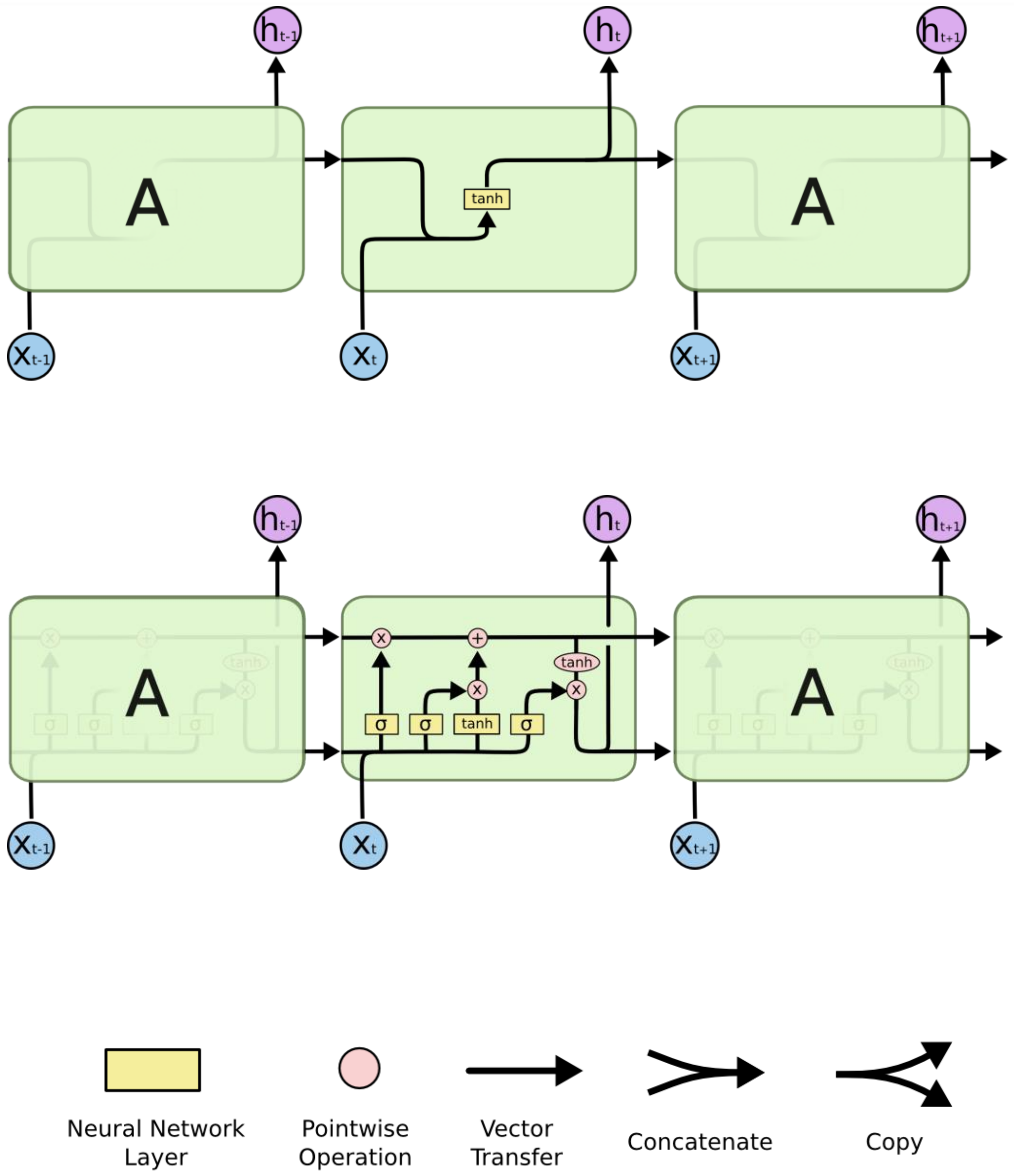
LSTM networks

– Cell state



– Gates

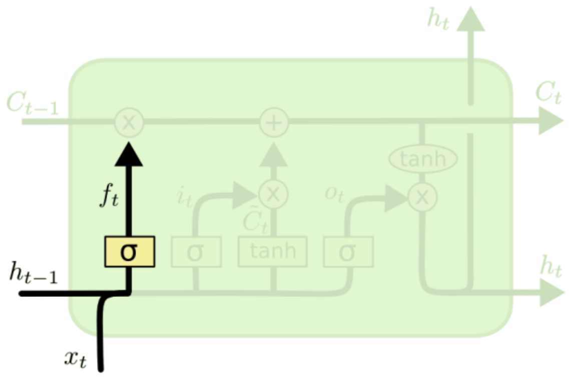




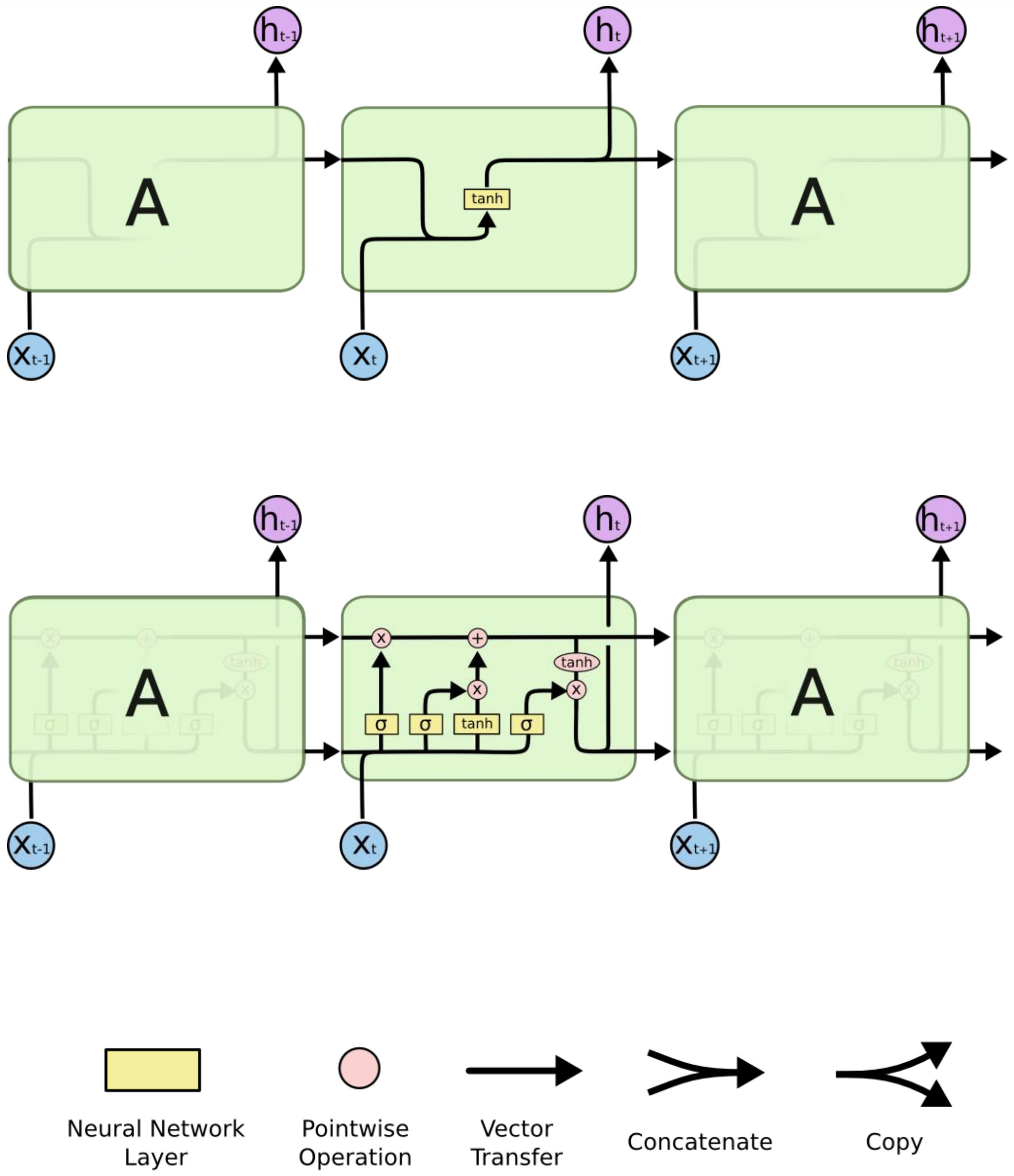
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM networks

– Forget gate



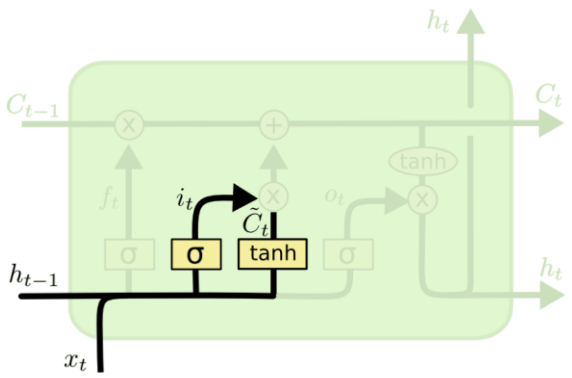
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$



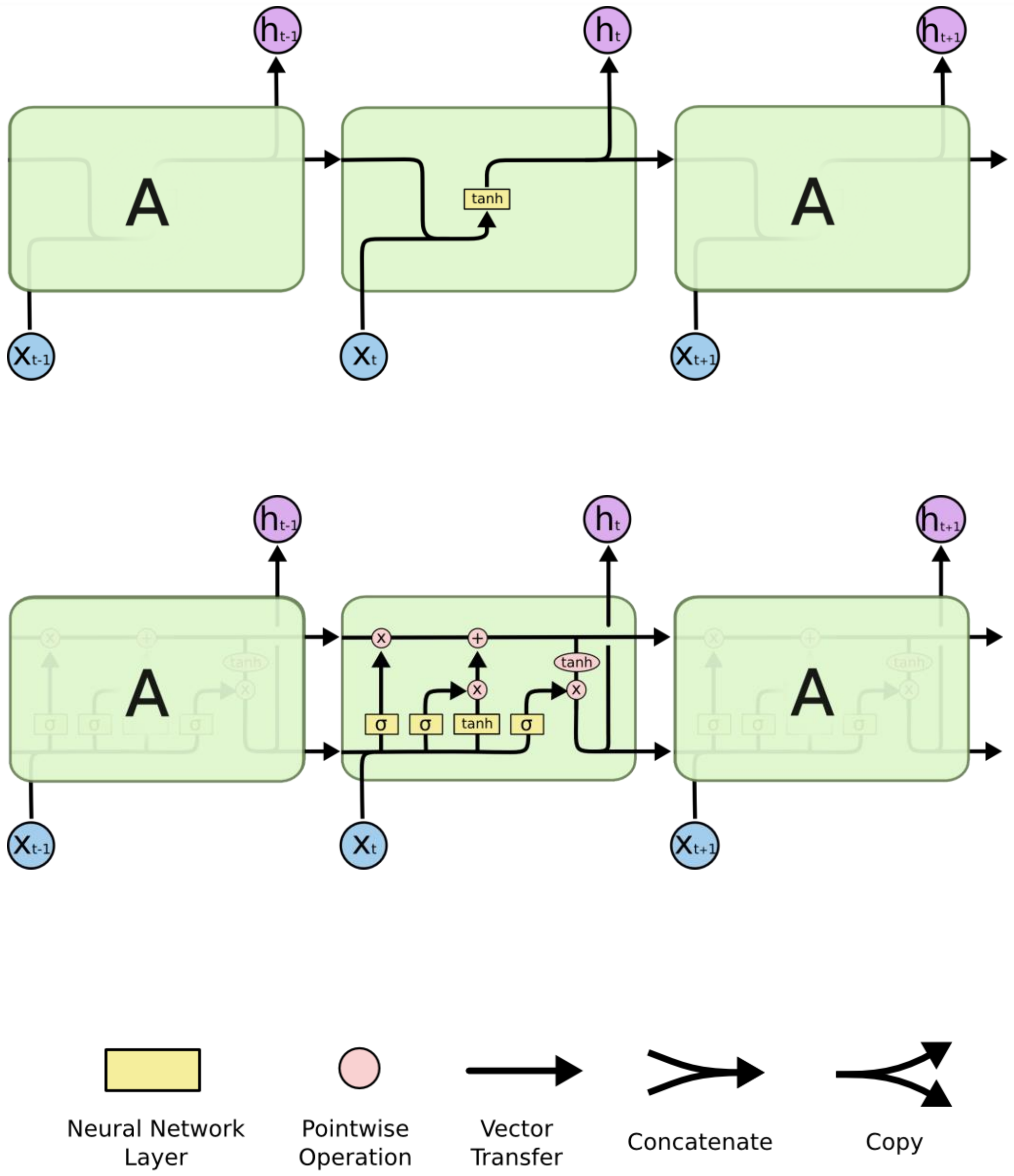
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM networks

– Input gate



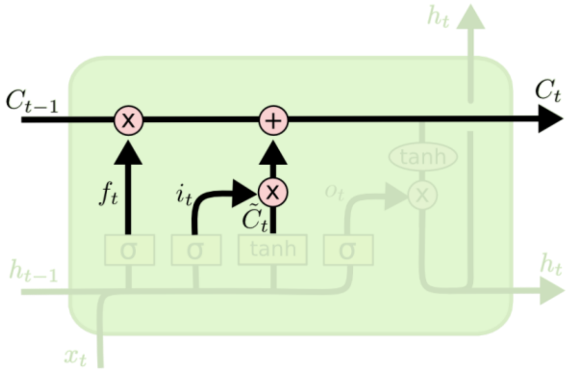
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$



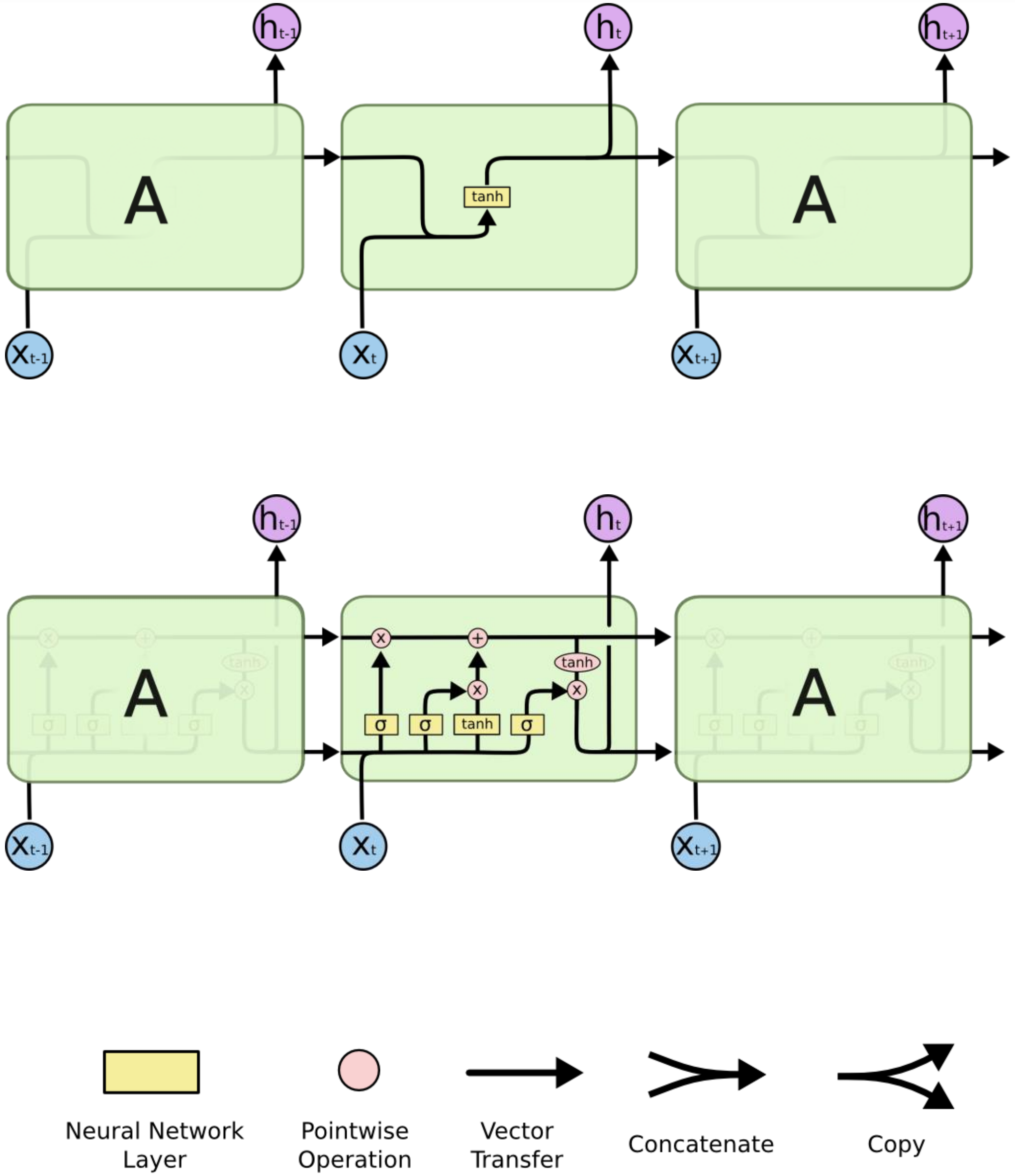
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM networks

– State update



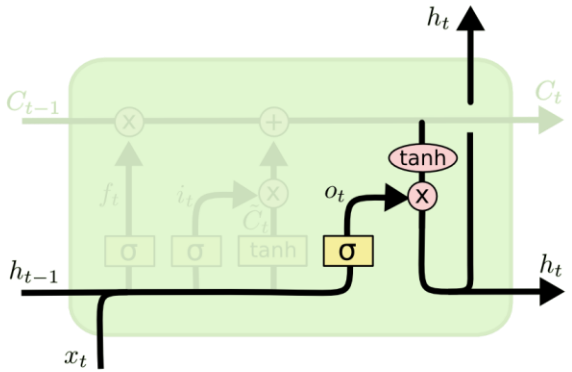
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$



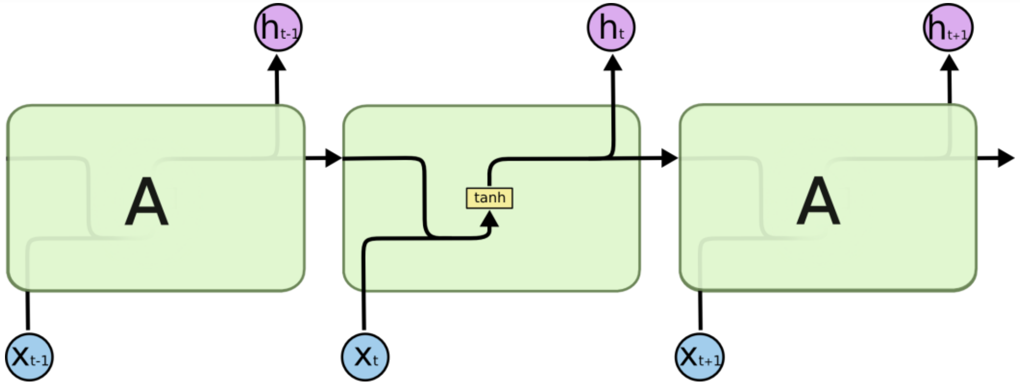
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

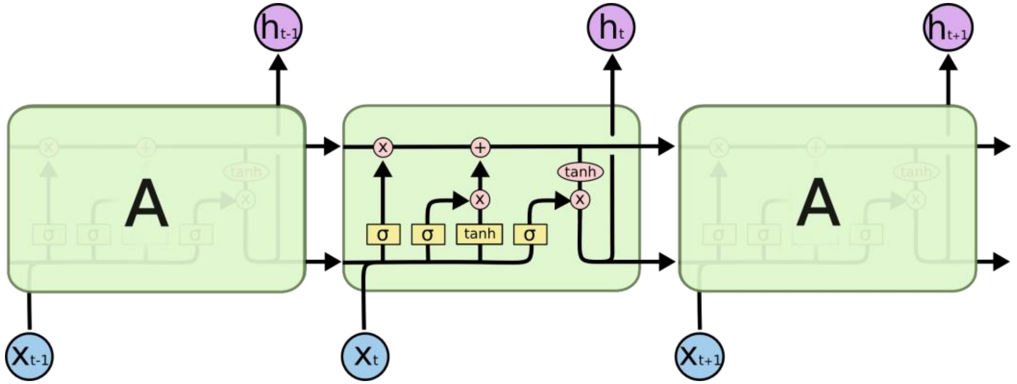
LSTM networks

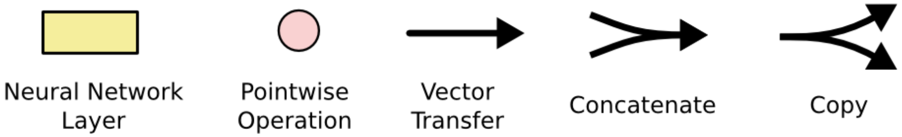
– Output gate



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

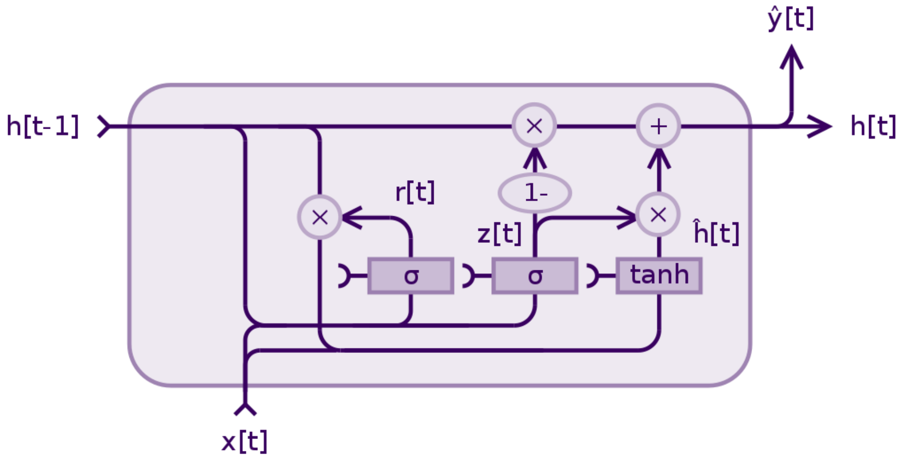


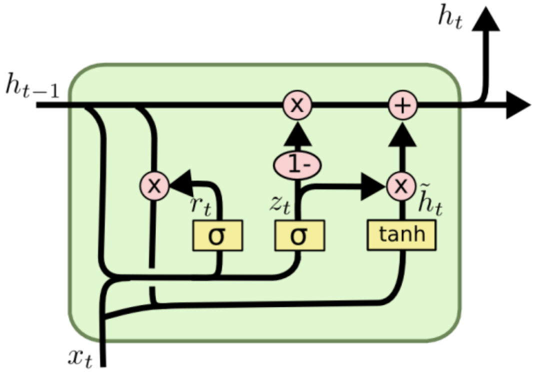




https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Gated recurrent units (GRUs) and LSTM variants





$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

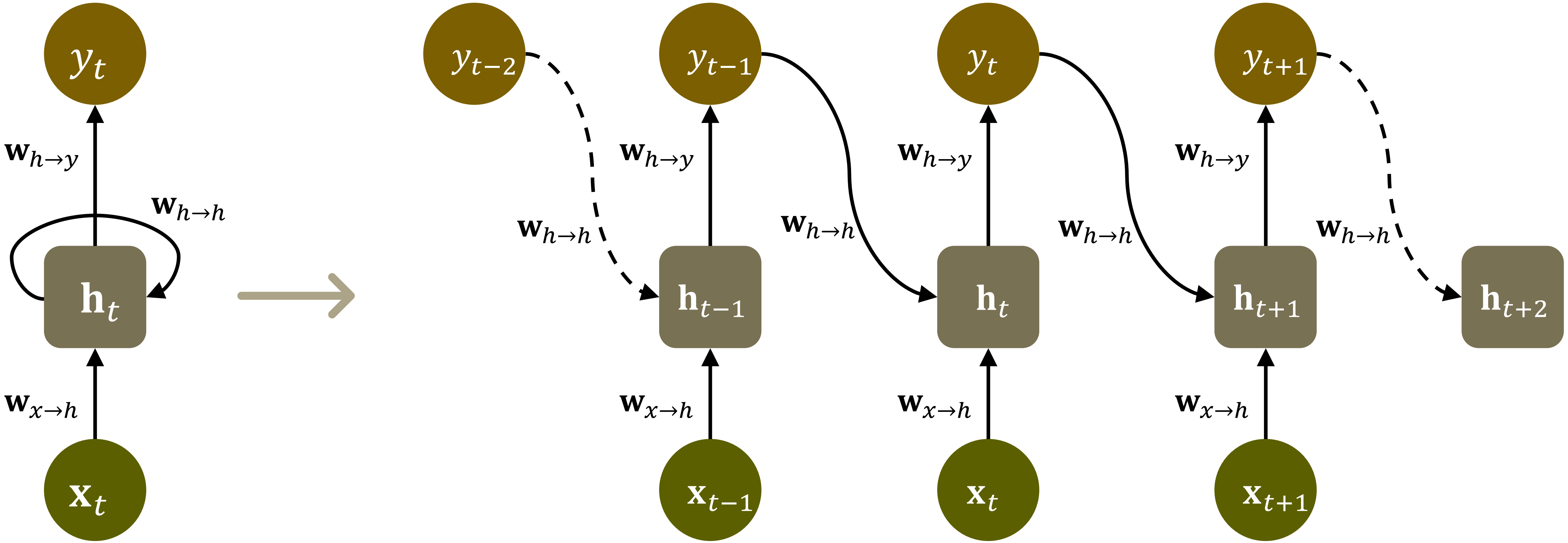
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

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 fixed-length input $\mathbf{x}$

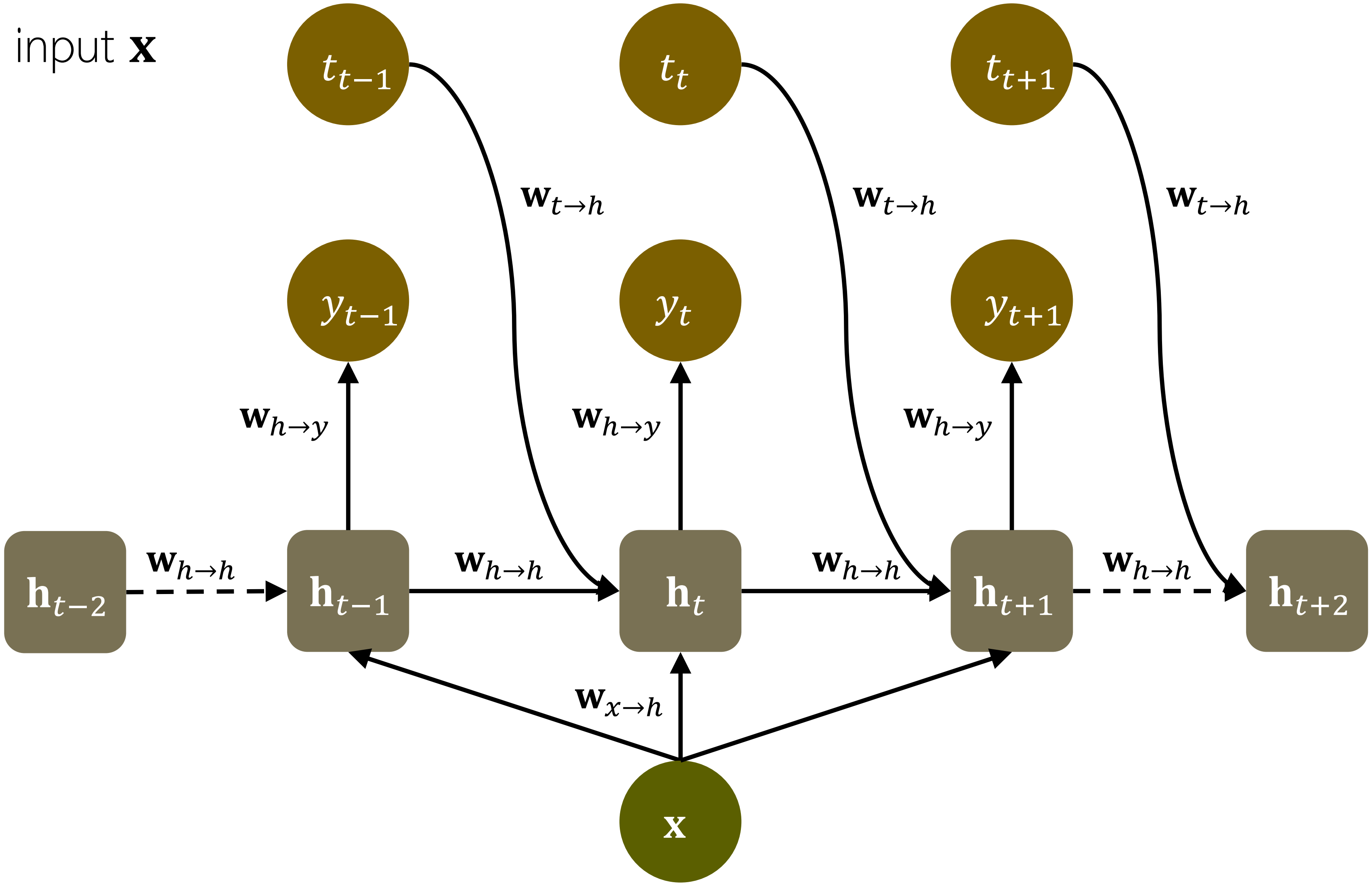
- Conditional modelling of target $\boldsymbol{t}$

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

- Conditional independence given $\mathbf{x}$

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

- e.g. seq2seq*, attention*

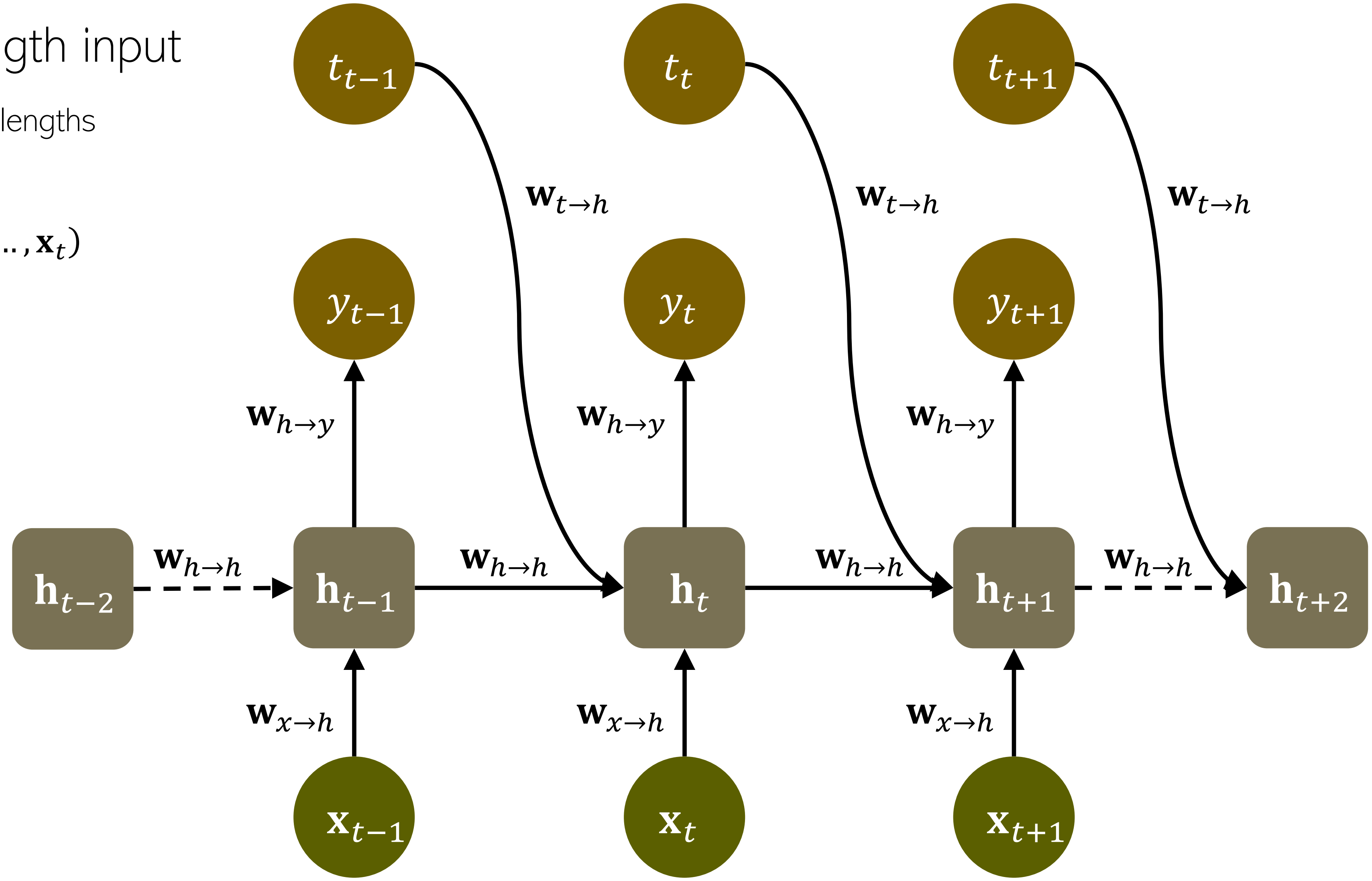


Input as context

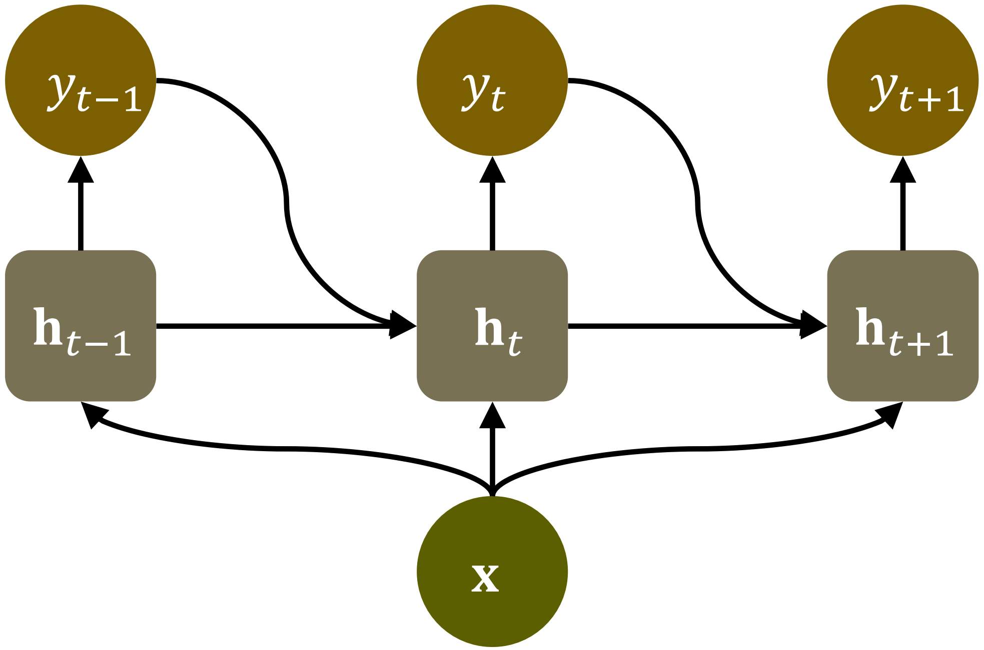
Conditioned on variable-length input

- Variable but equal input and output lengths

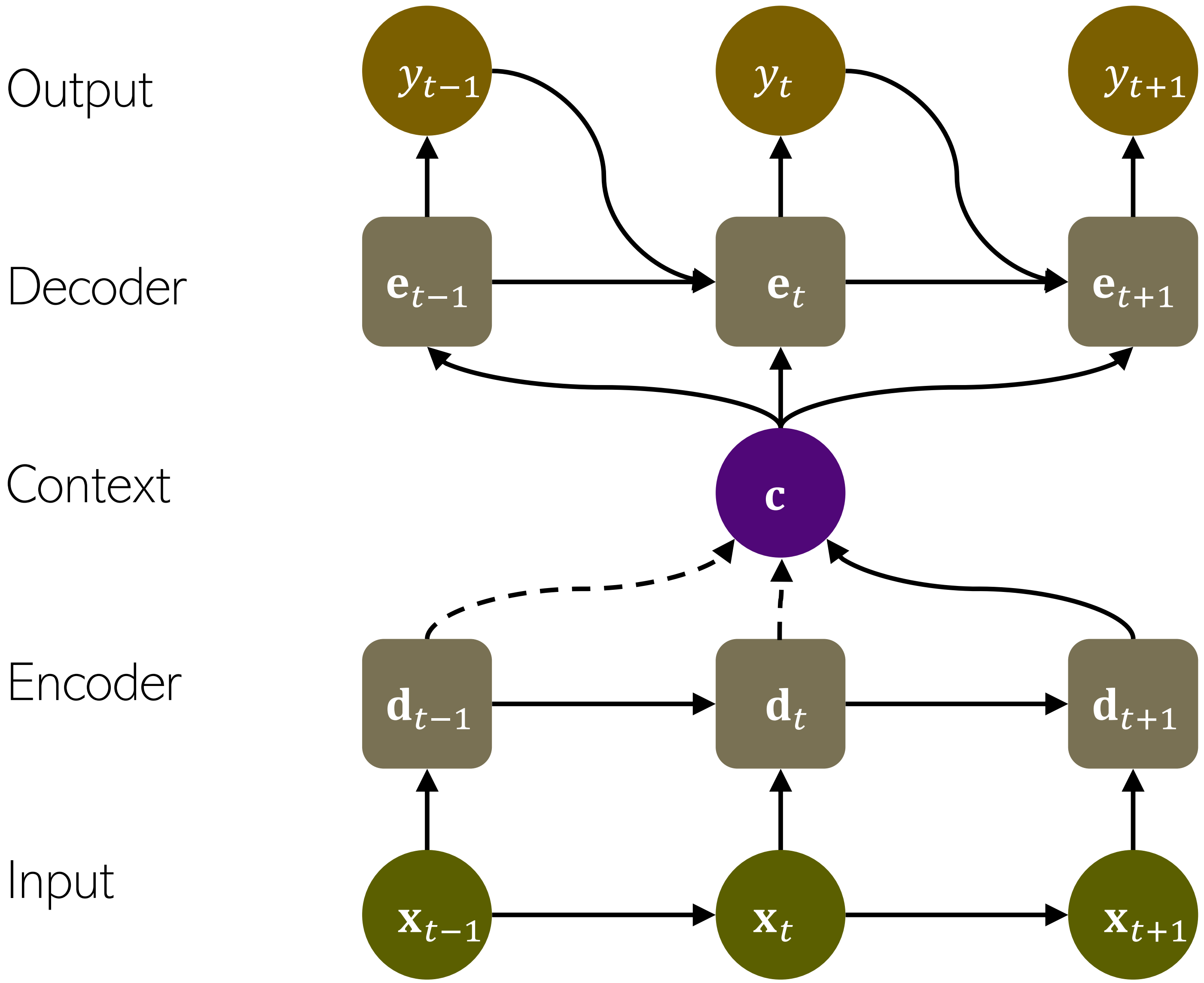
$$p(t_1, \dots, t_T | \mathbf{x}_1, \dots, \mathbf{x}_t) = \prod_t p(t_t \mid \mathbf{x}_1, \dots, \mathbf{x}_t)$$



Sequence-to-sequence with variable lengths

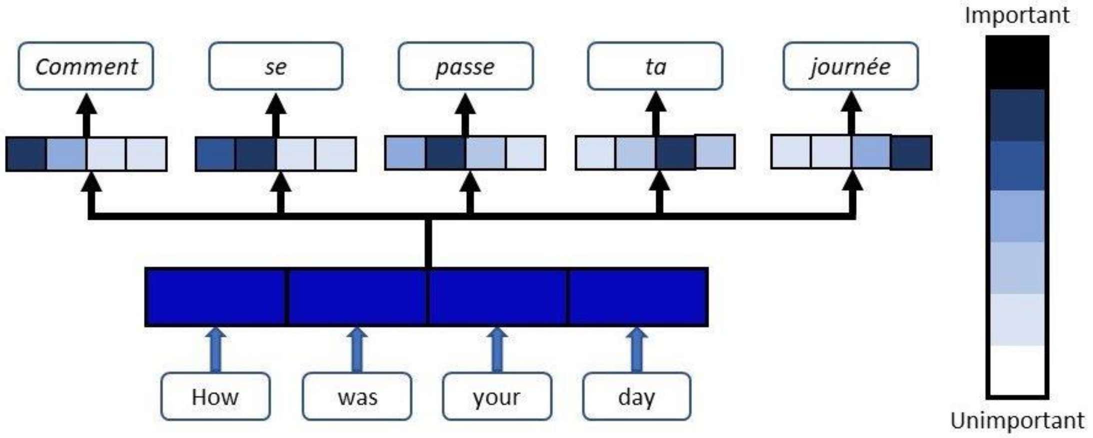


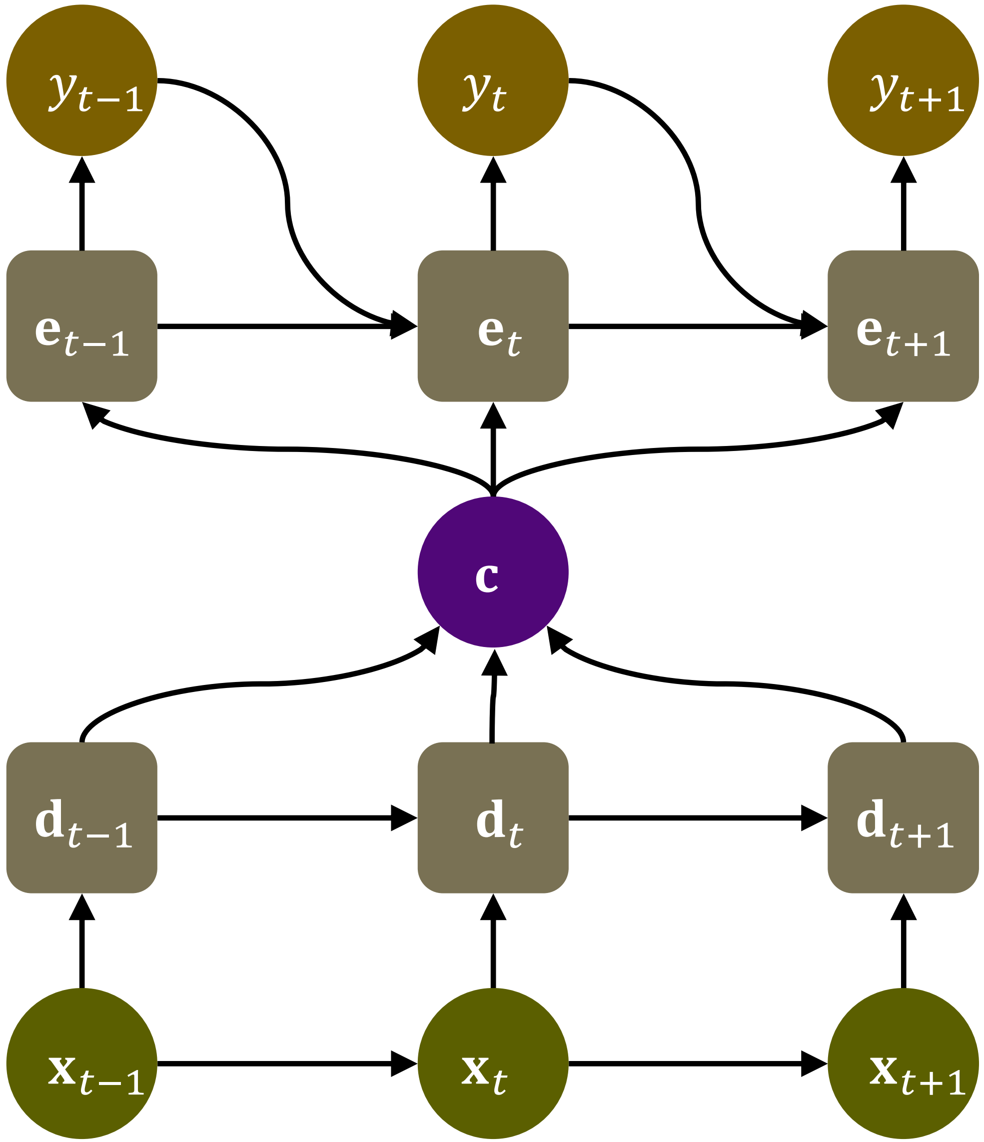
Sequence-to-sequence with variable lengths



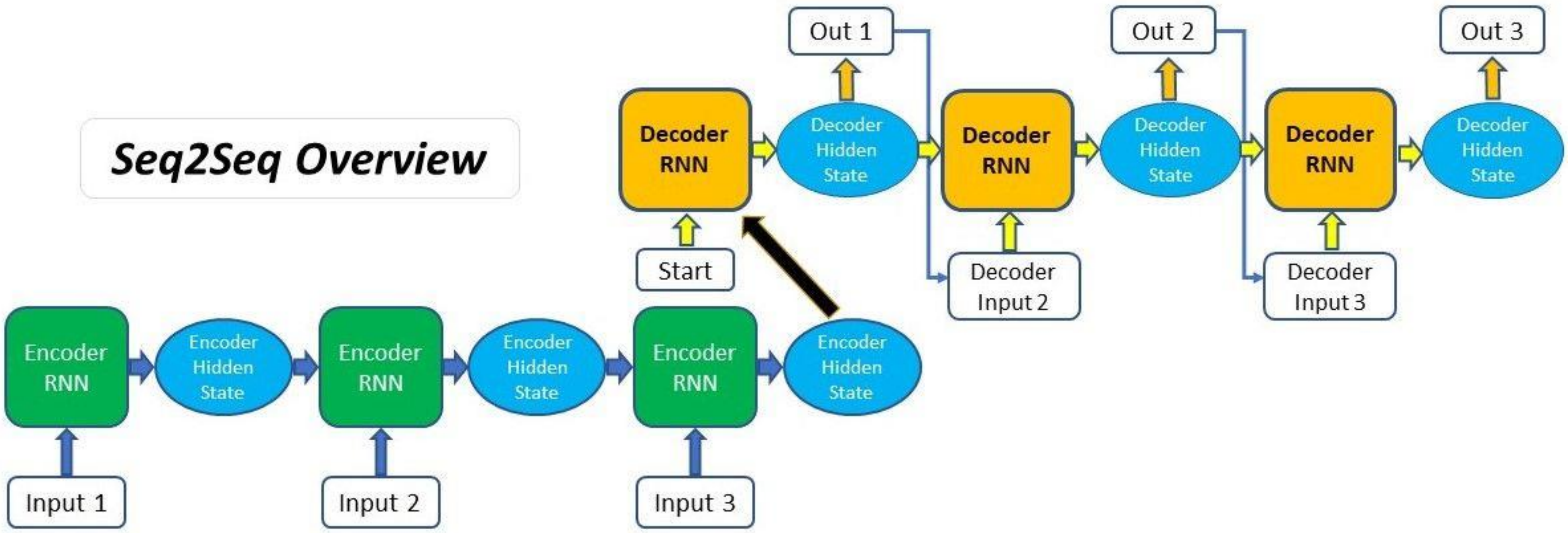
Learnable weights on context vector

(Self-)Attention mechanism

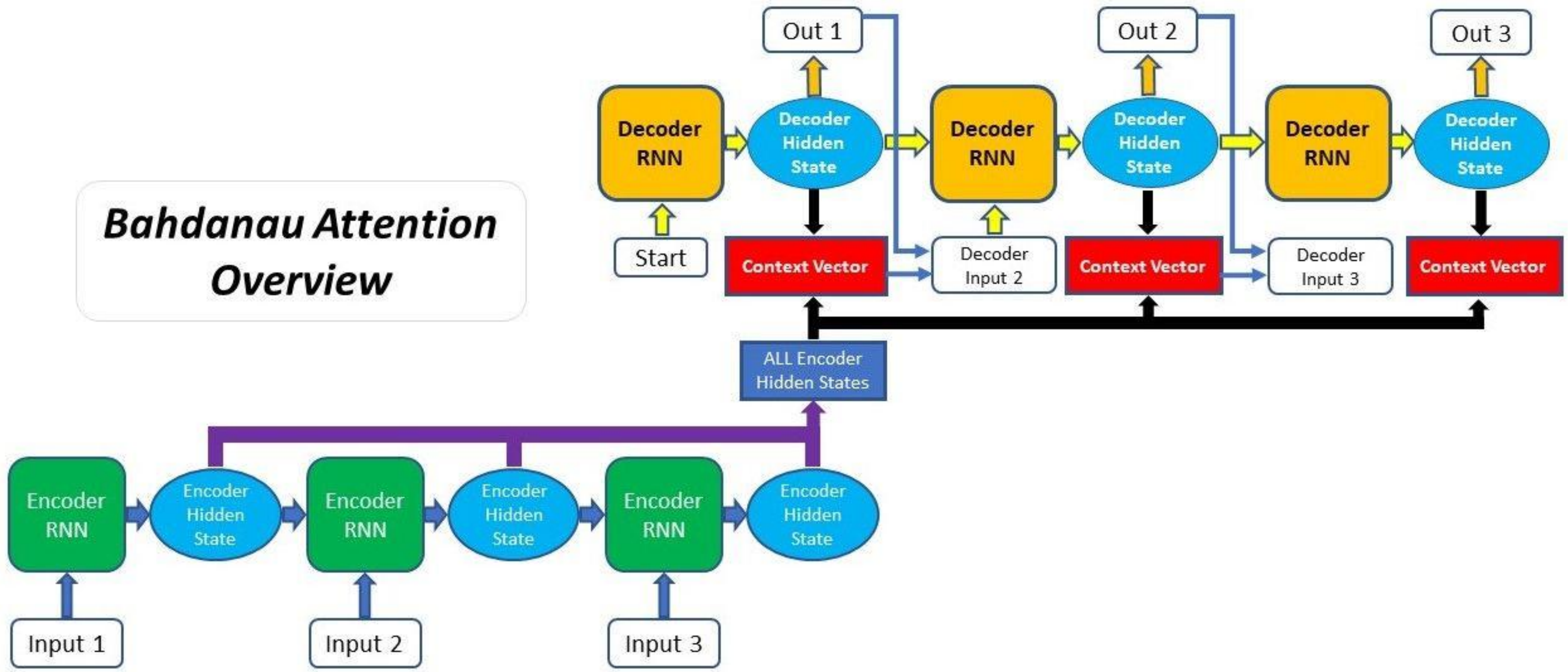




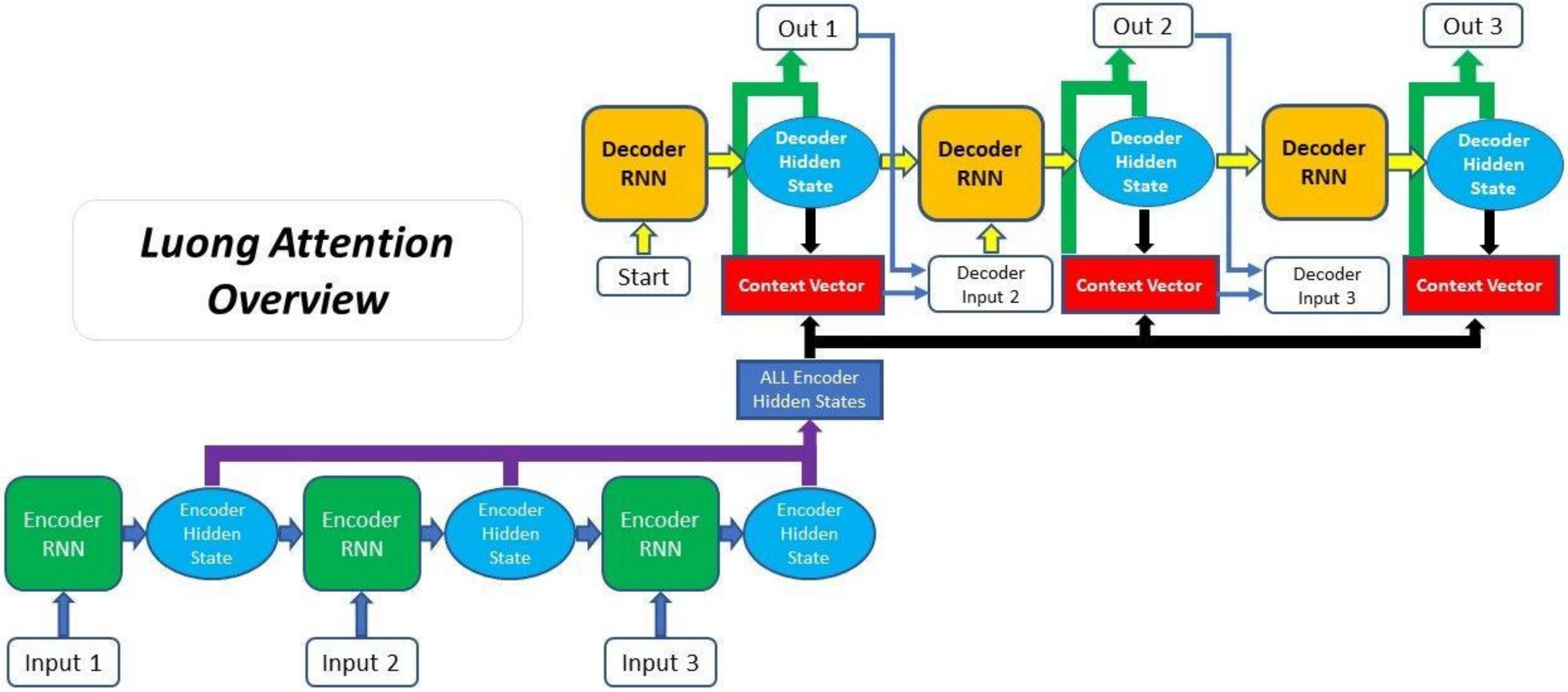
Seq2Seq architecture



Additive attention



Multiplicative attention



Transformer

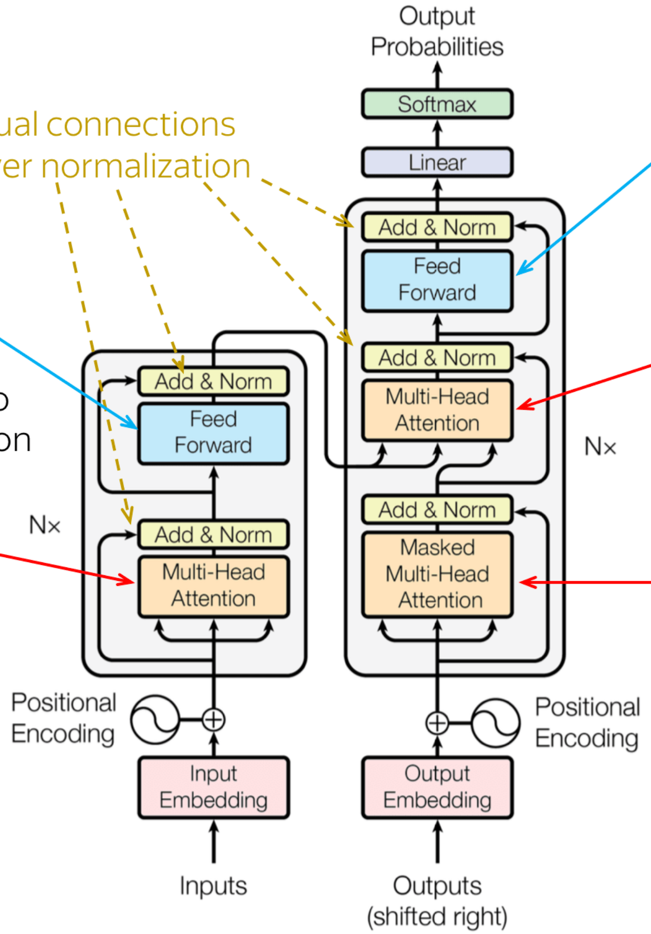
Without sequence-aligned RNNs

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

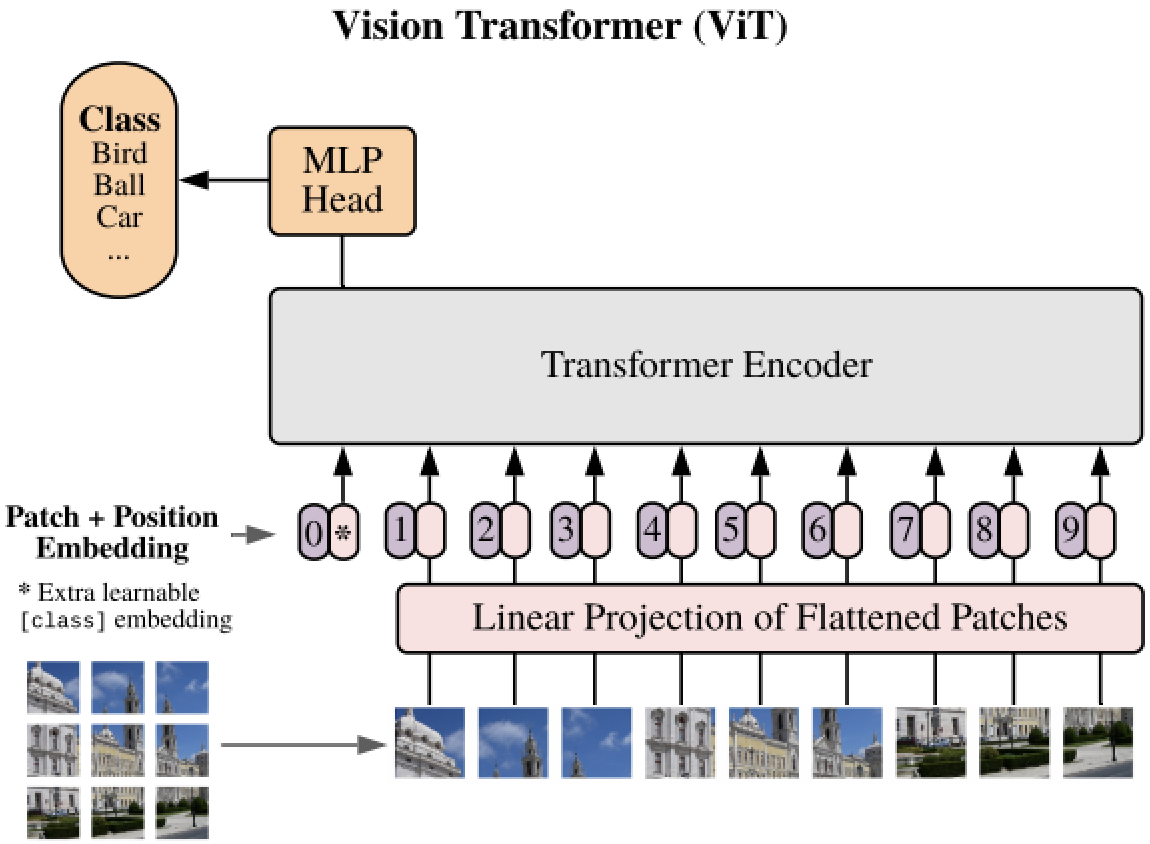
queries – from decoder states; keys
and values from encoder states

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