E9: 309 Advanced Deep Learning
19-10-2020

Instructor: Sriram Ganapathy
sriramg@iisc.ac.in

Teaching Assistant : Akshara Soman, Prachi Singh, Jaswanth Reddy
aksharas@iisc.ac.in, prachis@iisc.ac.in, jaswanthk@iisc.ac.in

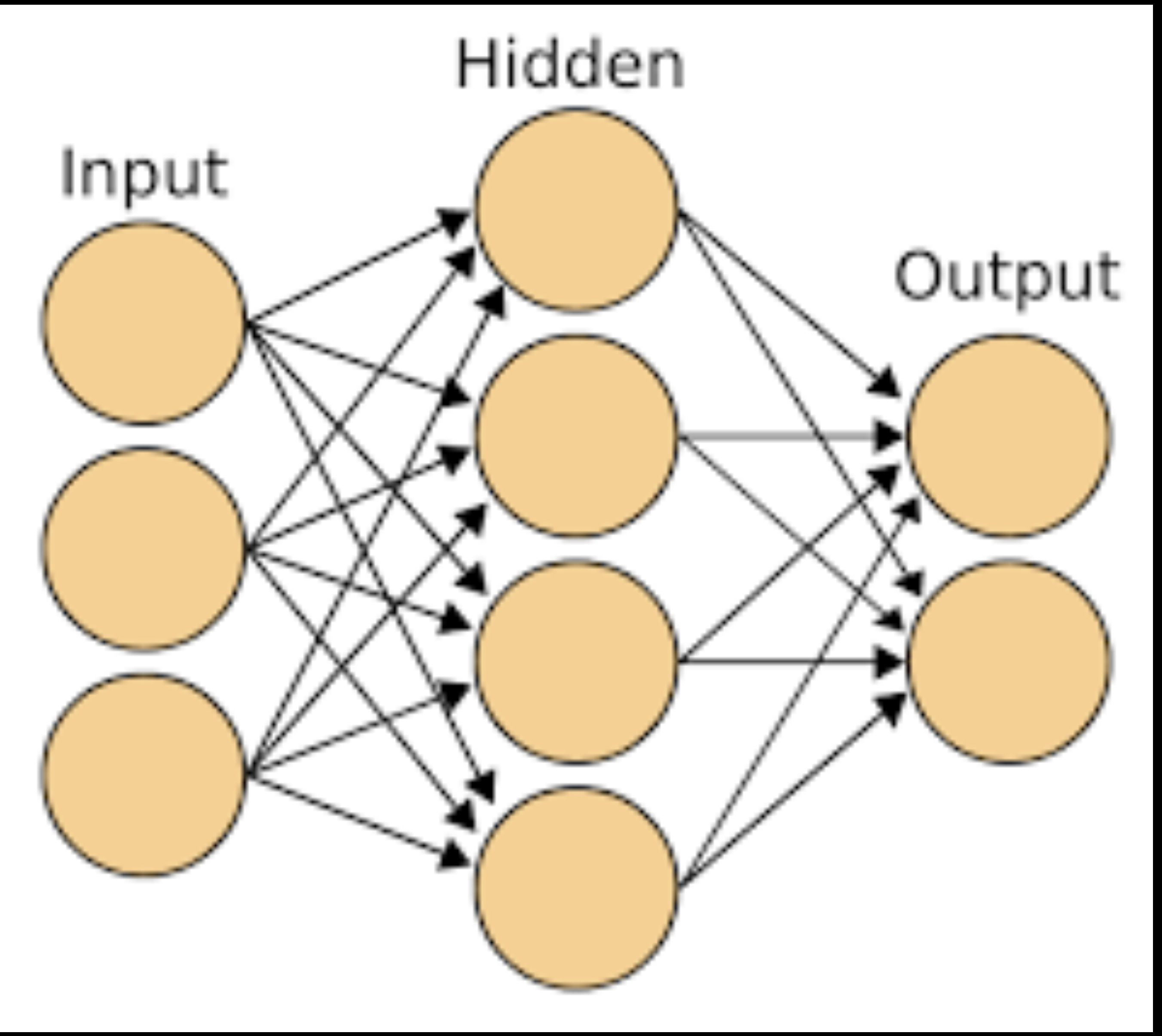
http://leap.ee.iisc.ac.in/sriram/teaching/ADL2020/

Housekeeping

- Filling the google form in the webpage
  - Contents will be made available to the folks in the creditors mailing lists.
    - Announcements regarding evaluations and projects will be shared only with creditors as well as video links.
      - Teams channel interaction and TA session for creditors only.
- Online registration portal from academics.iisc.ac.in
    - Your research/faculty advisor may need to approve also before the deadline (Oct. 20th?)

Recap of previous class

Some notations

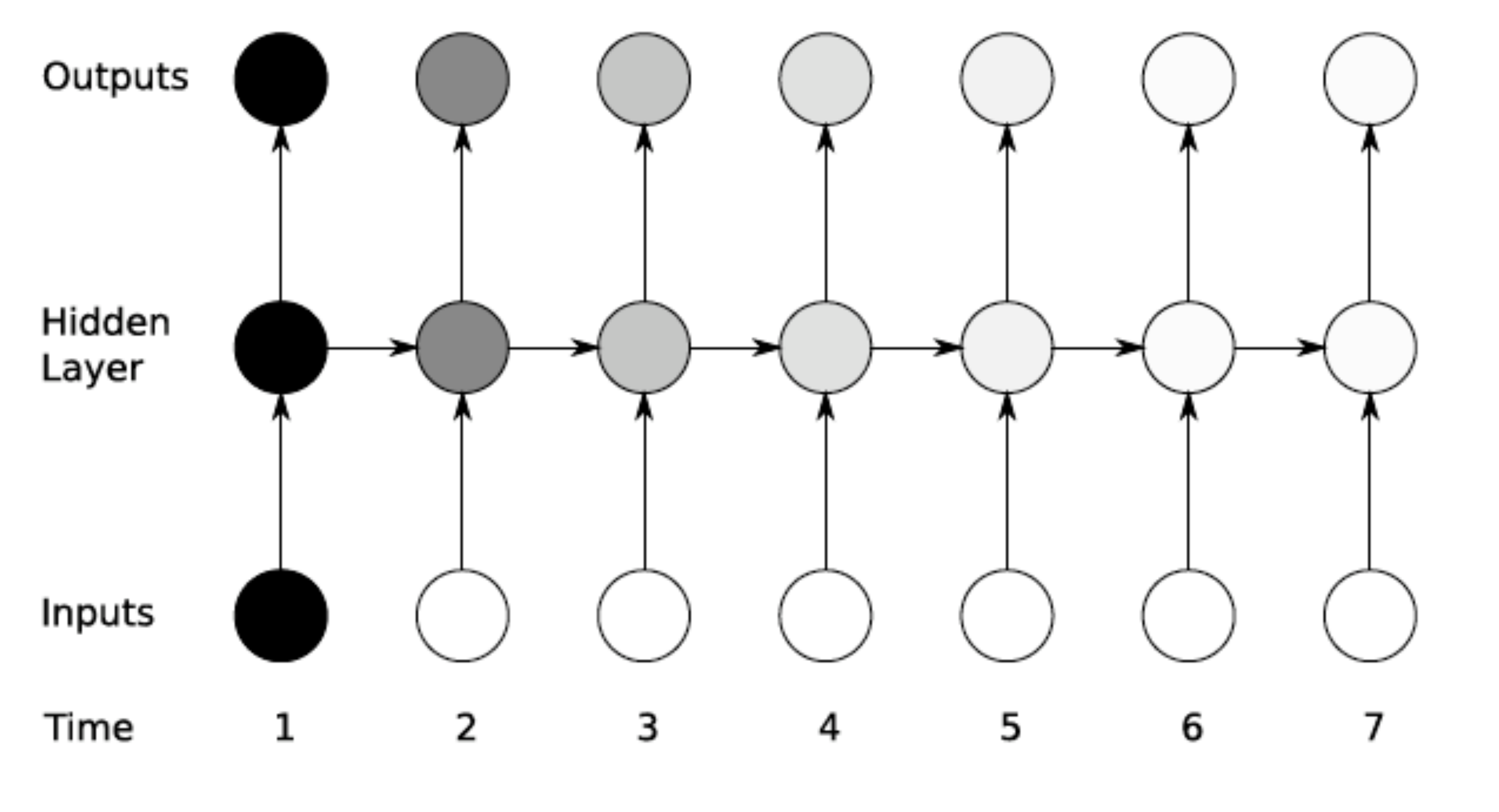


* $\boldsymbol{x} \in \mathcal{R}^D$ - input data.
* $\boldsymbol{y} \in \mathcal{R}^C$ - neural network targets.
* $\hat{\boldsymbol{y}} \in \mathcal{B}^C$ - model outputs.
* $\boldsymbol{e}, \boldsymbol{h} \in \mathcal{R}^d$ - hidden model representations or embeddings.
* $\boldsymbol{\Theta}$ - collection of learnable parameters in the model.
* $E(\boldsymbol{y}, \hat{\boldsymbol{y}})$ - error function used in the model training.

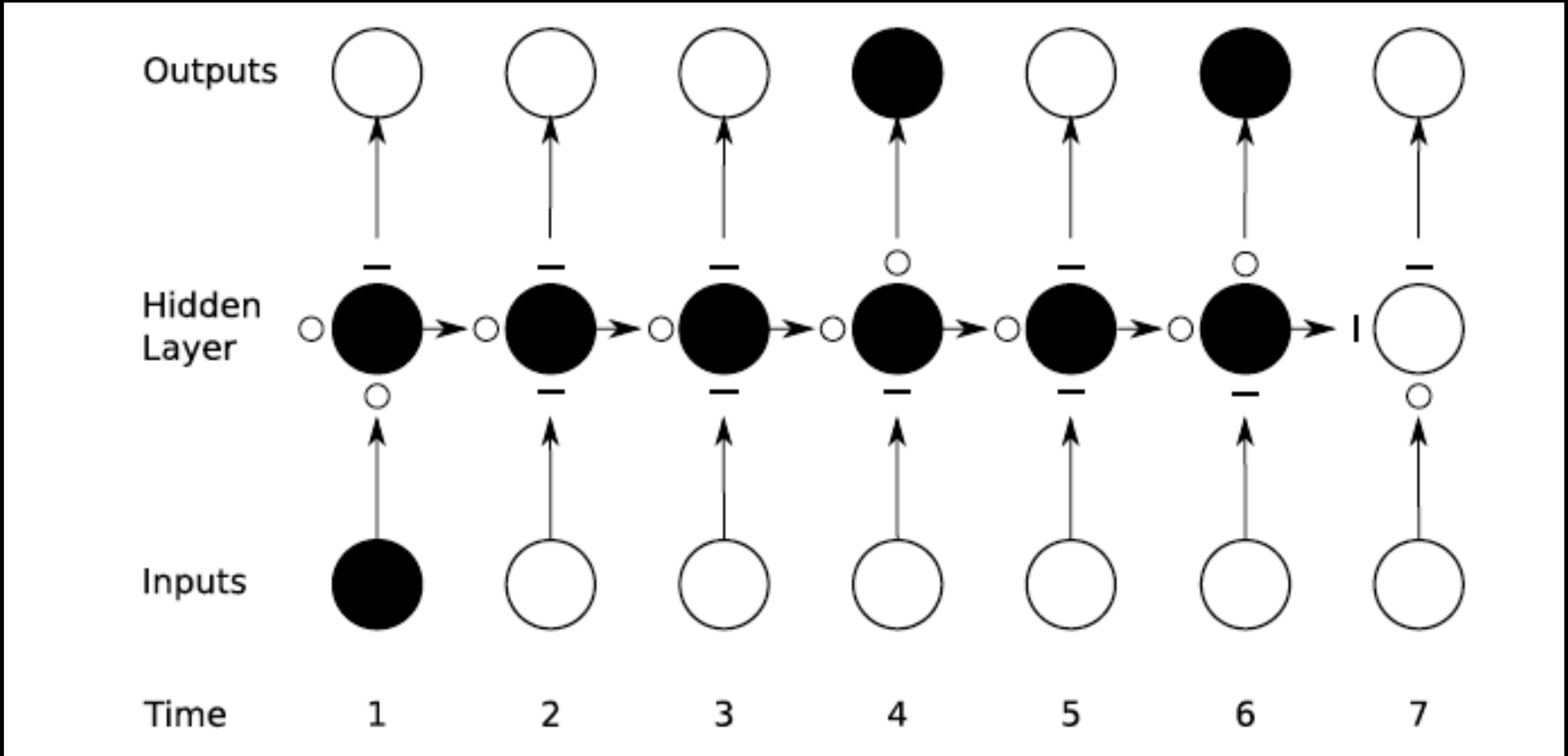
Some notations

- $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_N\}$ - labeled training data
- $q = \{1 \ldots Q\}$ - iteration index.
- $t = \{1 \ldots T\}$ - discrete time index.
- $l = \{1 \ldots L\}$ - layer index
- η - learning rate (hyper-parameter)
- N_b - mini-batch size and B is the number of mini-batches.

Long-term dependency issues



Long short term memory (LSTM) idea



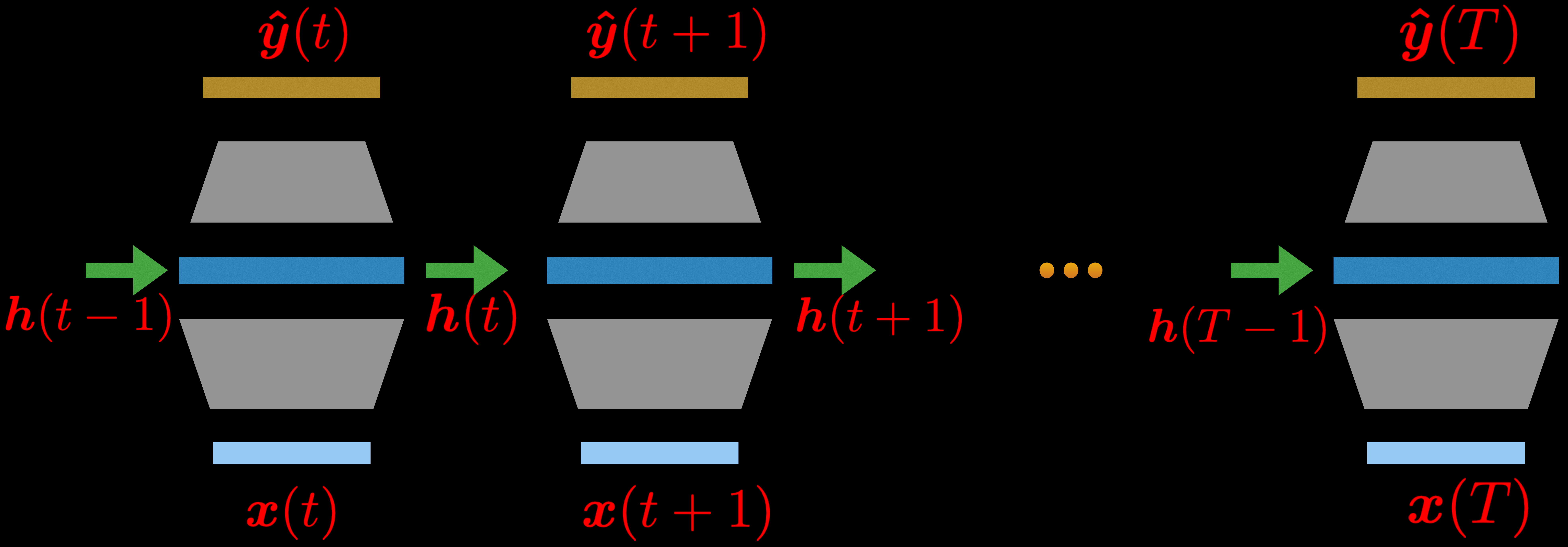
State of affairs

- Recurrent networks
  - First order recurrence
  - Back propagation in RNNs
- Issues with long-term dependency
  - Gates as neural networks
  - LSTM and GRUs

Reading Assignment - "A review of recurrent neural networks - LSTM cells and Network Architectures"
https://www.mitpressjournals.org/doi/pdfplus/10.1162/neco_a_01199

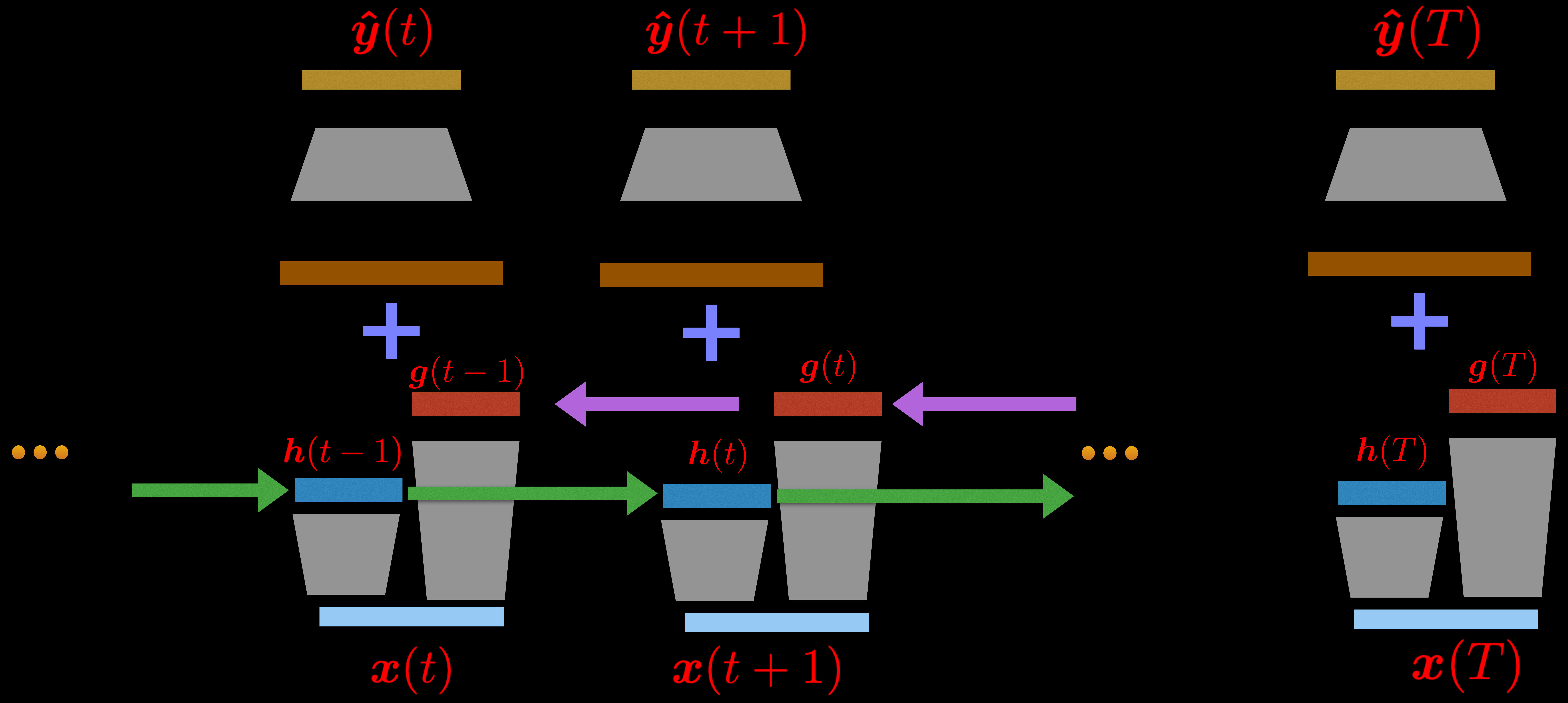
Other network architectures

* Multiple input multiple output



Bidirectional RNNs

* Multiple input multiple output

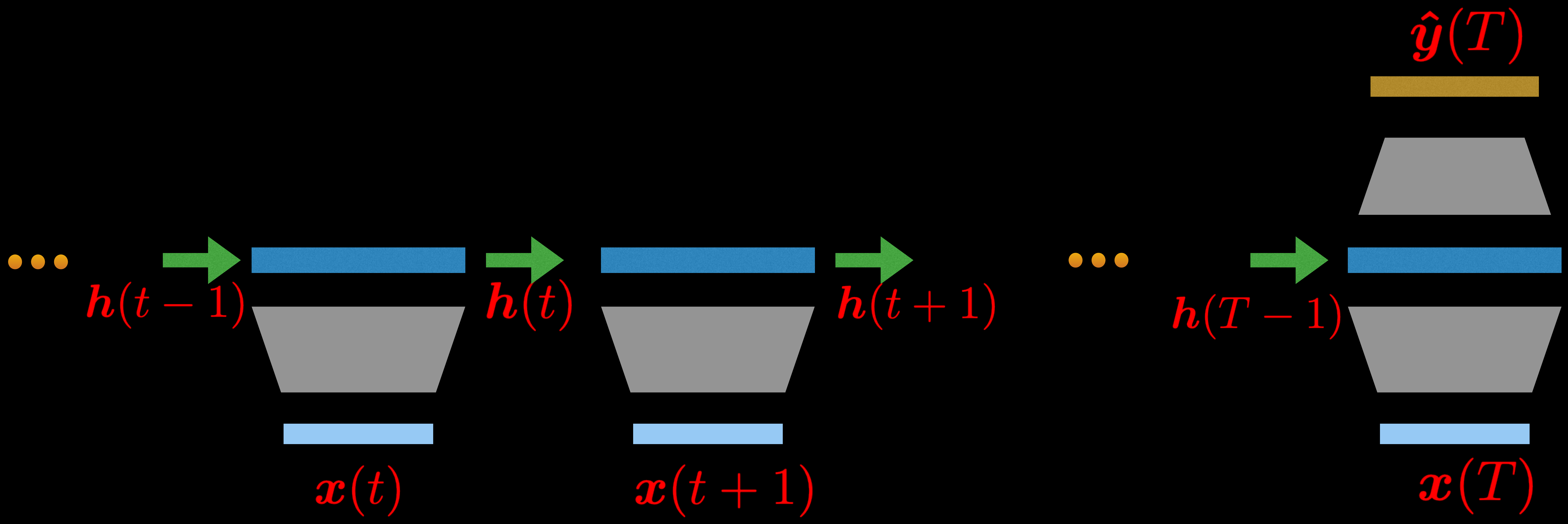


Bidirectional RNNs

- Forward and backward recurrence variables are summed.
- Can we implemented using LSTMS - called BLSTMs
- Backpropagation
  - Gradients for the forward recurrence will have a backward relationship
  - Gradients for the backward recurrence will have a forward relationship
- Commonly used in offline processing of sequence data
  - Mostly improves over the forward LSTM/RNN models.

Other network architectures

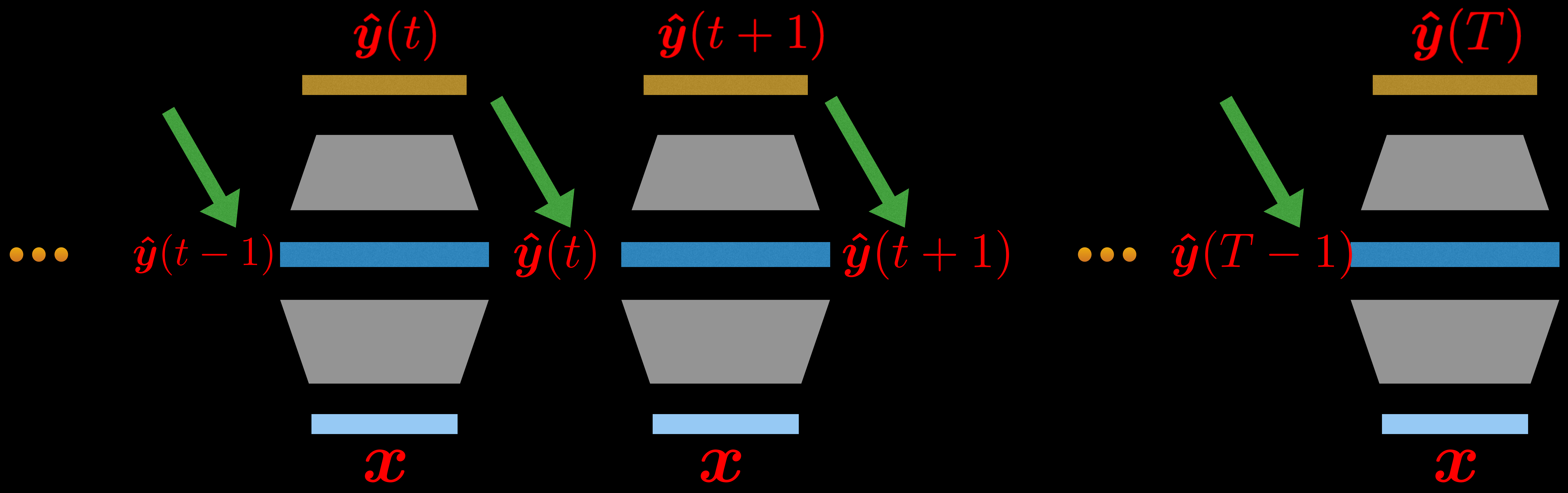
- Multiple input single output (seq2vec)



- Applications like topic summarization of text, speaker identification of speech.

Other network architectures

✳ Single input multiple output



✳ Image captioning.

Encoder-decoder models

- Multiple input multiple output (with different label index) - Seq2seq

$$x(1)...x(T)$$

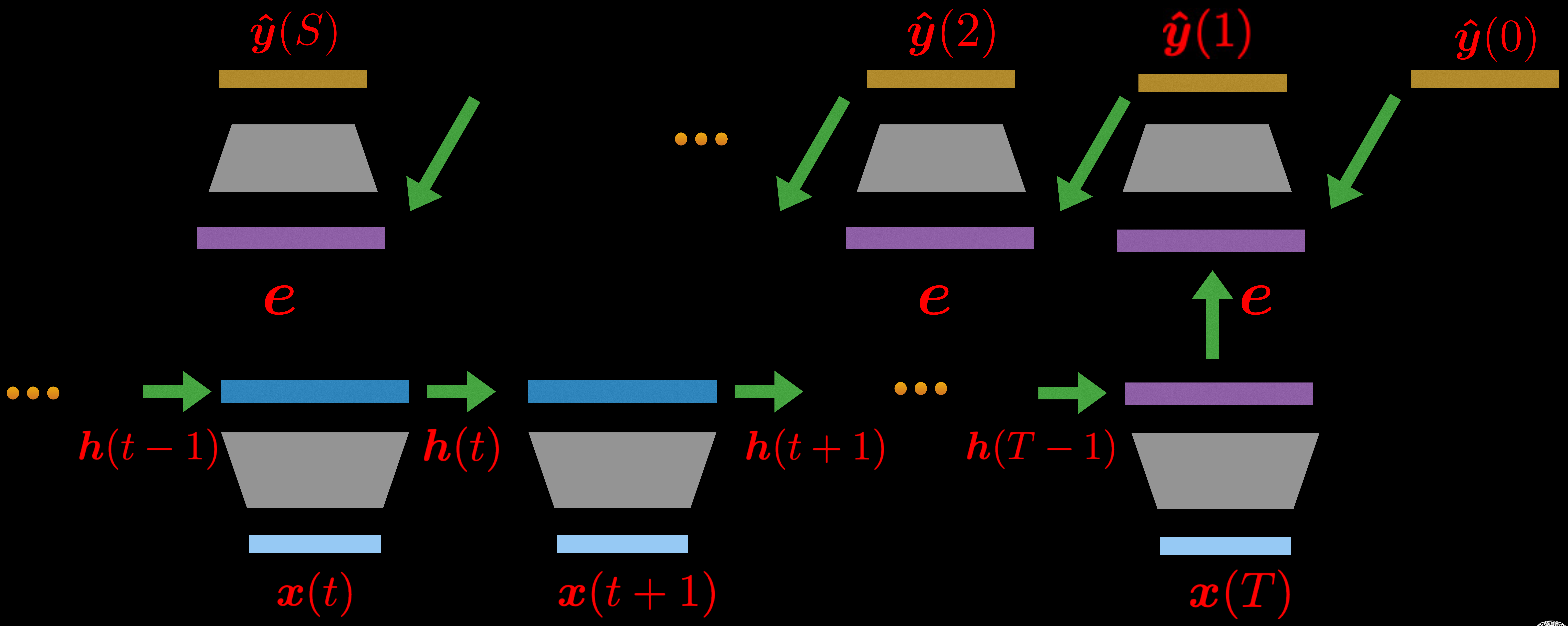
$$\Downarrow$$

$$y(1)...y(S)$$

- Applications
  - ✓ Machine translation.
  - ✓ Speech recognition.
  - ✓ Video captioning.

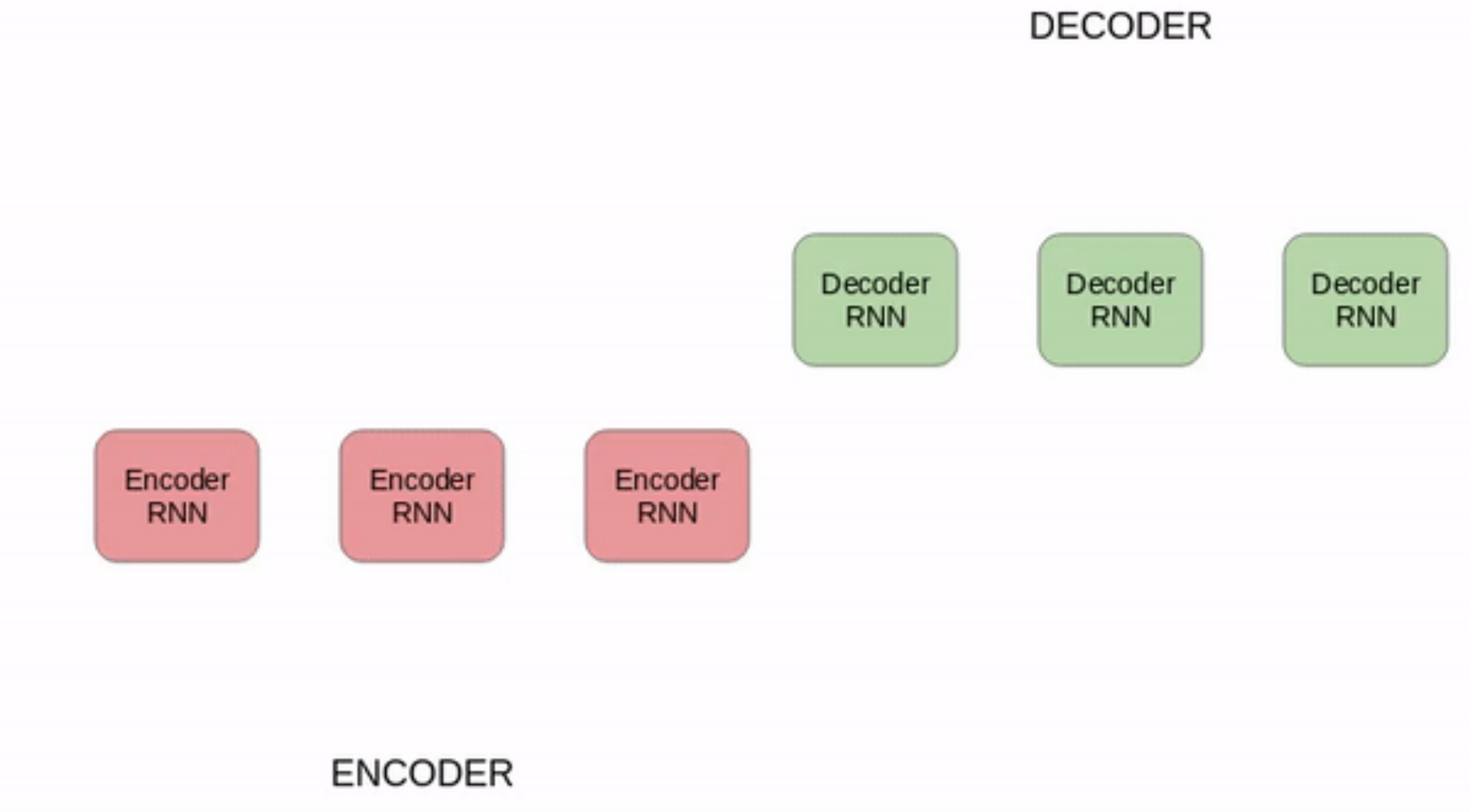
Encoder-decoder models

* Multiple input multiple output (with different label index) - Seq2seq



Encoder-decoder models

- Multiple input multiple output (with different label index) - Seq2seq



source: https://www.analyticsvidhya.com/blog/2019/06/understanding-transformers-nlp-state-of-the-art-models/

Encoder-decoder models

* Encoder — convert sequence $\mathbf{x} = \{\mathbf{x}(1), .., \mathbf{x}(T)\}$ to vector

$$\mathbf{h}(t) = f(\mathbf{h}(t-1), \mathbf{x}(t))$$
$$\mathbf{e} = f'(\mathbf{h}_1, ..., \mathbf{h}_T)$$

* The encoder can have multiple deep RNN layers.

* For simplicity

$$\mathbf{e} = \mathbf{h}_T$$

Encoder-decoder models

- Encoder — convert sequence $\mathbf{x} = \{\mathbf{x}(1), .., \mathbf{x}(T)\}$ to vector $\mathbf{e}$

- Decoder — converts the vector embedding from the encoder to the output sequence $\hat{\mathbf{y}} = \{\hat{\mathbf{y}}(1), ..., \hat{\mathbf{y}}(S)\}$ with different label index.

$$p(\hat{\mathbf{y}}) = \prod_{s=1}^{S} p(\hat{\mathbf{y}}(s)|\hat{\mathbf{y}}(1), ..., \hat{\mathbf{y}}(s-1))$$

- RNN decoder assumption

$$p(\hat{\mathbf{y}}(s)|\hat{\mathbf{y}}(1), ..., \hat{\mathbf{y}}(s-1)) = p(\hat{\mathbf{y}}(s)|\hat{\mathbf{y}}(s-1), \mathbf{e}) = softmax(\mathbf{V}\hat{\mathbf{y}}(s-1) + \mathbf{R}\mathbf{c}(s-1) + \mathbf{T}\mathbf{e} + \mathbf{d})$$

$$\mathbf{c}(s) = f(\mathbf{c}(s-1), \mathbf{e})$$

- The decoder can also have multiple layers of deep RNNs before softmax.

Encoder-decoder models

- Encoder — convert sequences to vectors
- Decoder — converts the vector embedding from the encoder to the output sequence with different label index.
  - ✓ Start and end label are also encoded as output vector indices.
    - ⋆ Enable the starting and ending of the output sequence.
- Assumption
  - ✓ The entire input sequence can be represented as a single vector $\mathbf{e}$
    - ⋆ May not be able to perform this efficiently for long sequences.

Encoder-decoder models

* Modification of encoder-decoder model

$$p(\hat{\mathbf{y}}(s)|\hat{\mathbf{y}}(1), ..., \hat{\mathbf{y}}(s-1)) = p(\hat{\mathbf{y}}(s)|\hat{\mathbf{y}}(s-1), \mathbf{e}) = softmax(\mathbf{V}\hat{\mathbf{y}}(s-1) + \mathbf{R}\mathbf{c}(s-1) + \mathbf{T}\mathbf{e} + \mathbf{d})$$

$$p(\hat{\mathbf{y}}(s)|\hat{\mathbf{y}}(1), ..., \hat{\mathbf{y}}(s-1)) = p(\hat{\mathbf{y}}(s)|\hat{\mathbf{y}}(s-1), \mathbf{e}(s)) = softmax(\mathbf{V}\hat{\mathbf{y}}(s-1) + \mathbf{R}\mathbf{c}(s-1) + \mathbf{T}\mathbf{e}(s) + \mathbf{d})$$

* where

$$\mathbf{e}(s) = \sum_{t=1}^{T} \alpha(s,t)\mathbf{h}(t)$$

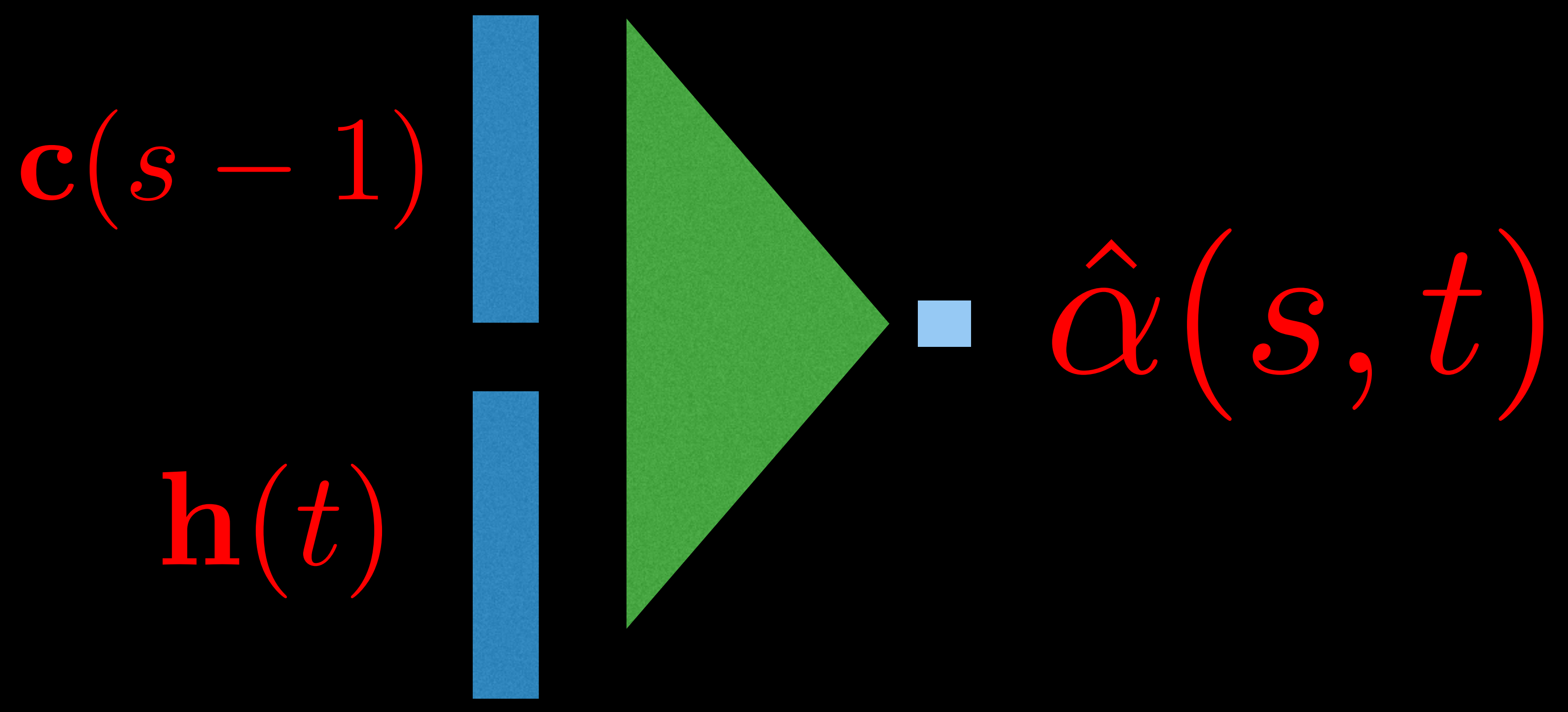
* Here $\alpha(s,t)$ captures the contribution of input at time **t** with output at time **s**

Encoder-decoder models

* Obtaining the relative contribution $\alpha(s,t)$

  ✓ Implementing this automatically using network-in-network

Attention network



$$\hat{\alpha}(s,t) = \mathbf{A}[\mathbf{c}(s-1);\mathbf{h}(t)]$$

$$\alpha(s,t) = S(\hat{\alpha}(s,t)) = \frac{exp(\hat{\alpha}(s,t))}{\sum_{t'} exp(\hat{\alpha}(s,t'))}$$

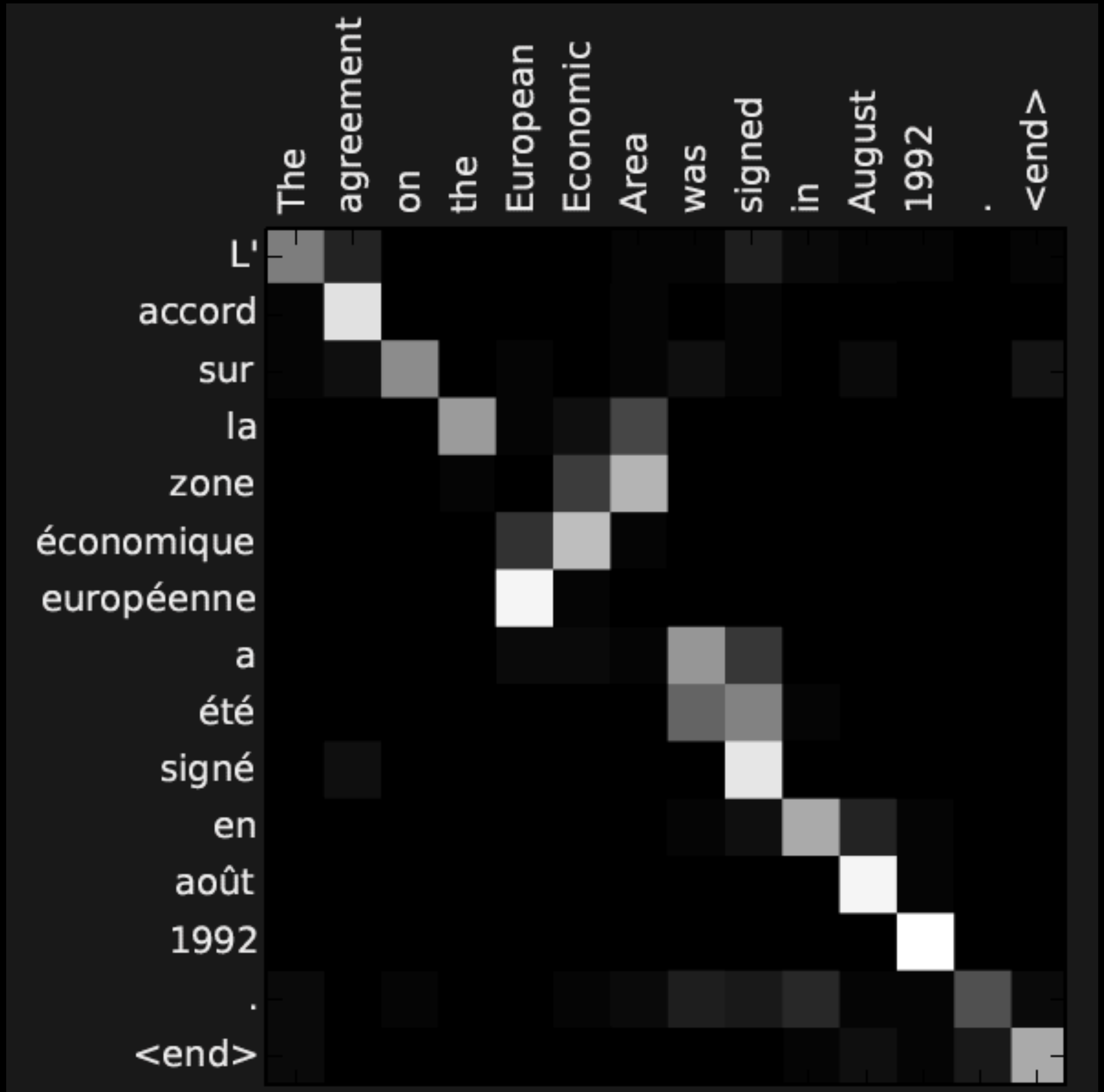
  ✓ The values $\alpha(s,t)$ are called attention weights.

Analysis of attention networks

- Attention weights $\alpha(s, t)$
  - ✓ Probability of linking (attending) to input at t for generating output at s
  - ✓ Useful in analyzing the internal structure of the encoder-decoder model

Visualizing the attention weights

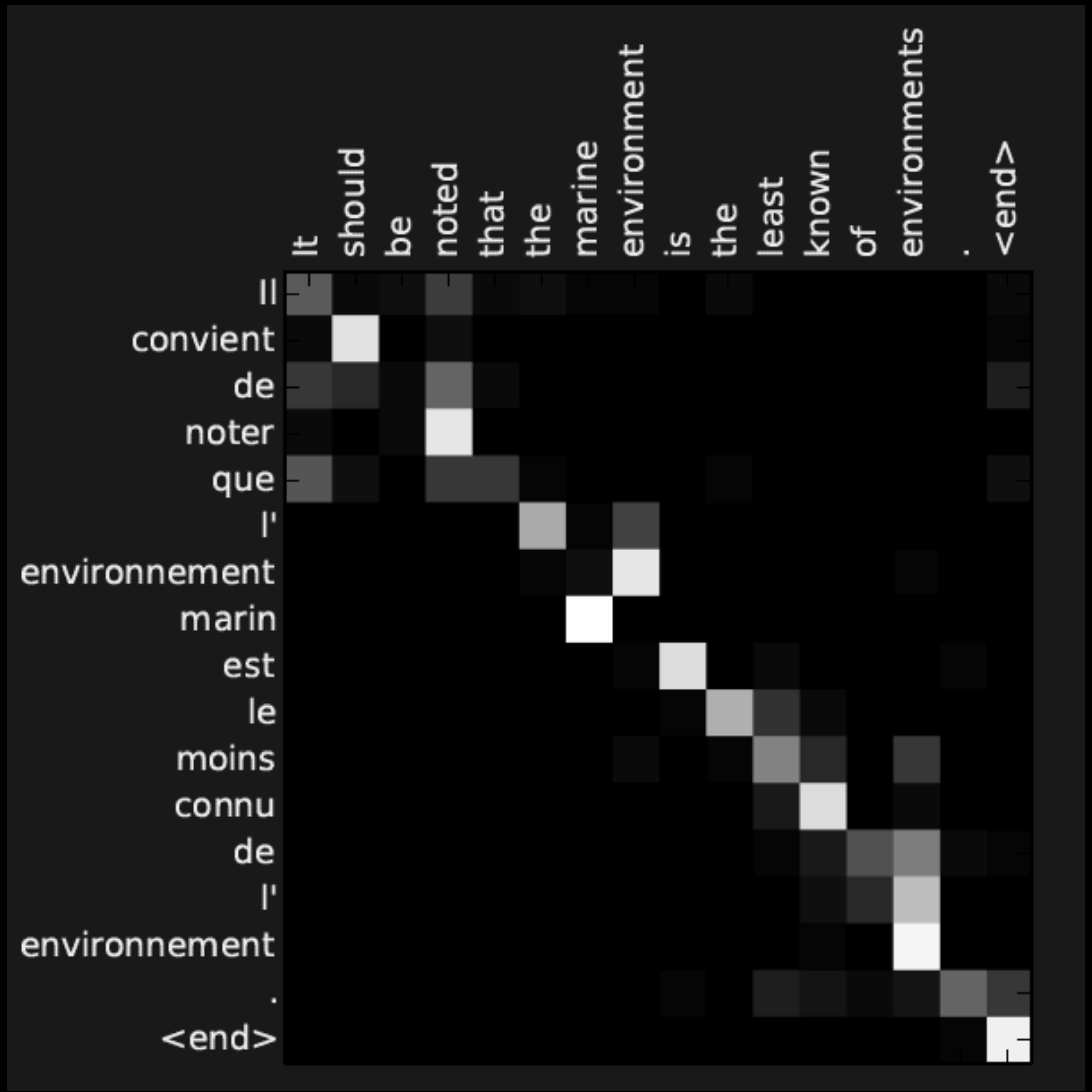
Reading Assignment - "Neural Machine Translation by Jointly Learning to Align and Translate"
https://arxiv.org/pdf/1409.0473.pdf



Analysis of attention networks

* Attention weights $\alpha(s, t)$
  - ✓ Probability of linking (attending) to input at t for generating output at s
  - ✓ Useful in analyzing the internal structure of the encoder-decoder model

Visualizing the attention weights



Reading Assignment - "Neural Machine Translation by Jointly Learning to Align and Translate"
https://arxiv.org/pdf/1409.0473.pdf

Visualizing attention