MACHINE LEARNING FOR SIGNAL PROCESSING

12-3-2025

Sriram Ganapathy
LEAP lab, Electrical Engineering, Indian Institute of Science,
sriramg@iisc.ac.in

Viveka Salinamakki, Varada R.
LEAP lab, Electrical Engineering, Indian Institute of Science

http://leap.ee.iisc.ac.in/sriram/teaching/MLSP25/

STORY SO FAR

EM algorithm

Classification Problem

Decision Theory

Generative Modeling

Gaussian Modeling

Gaussian Mixture Modeling

Function Modeling

Linear Models for Regression and Classification

Kernel Machines & Max-margin classifiers

Support Vector Machines

Data Representations PCA, LDA

Discriminative Modeling

Gradient Descent

Neural Networks

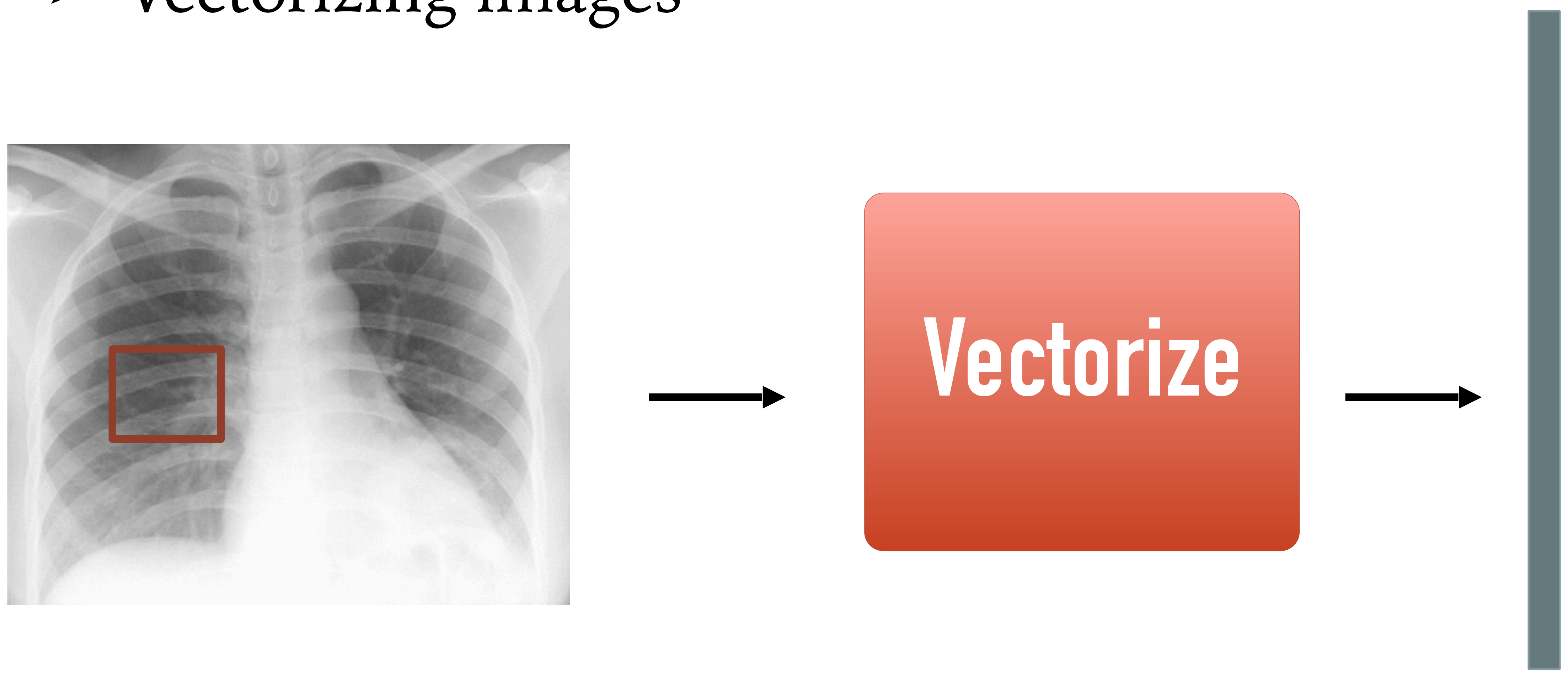
Learning Rules & Normalization

Deep neural architectures

NEURAL NETWORK ARCHITECTURES

WHAT MAKES DNN SUBOPTIMAL FOR IMAGES

- Vectorizing images



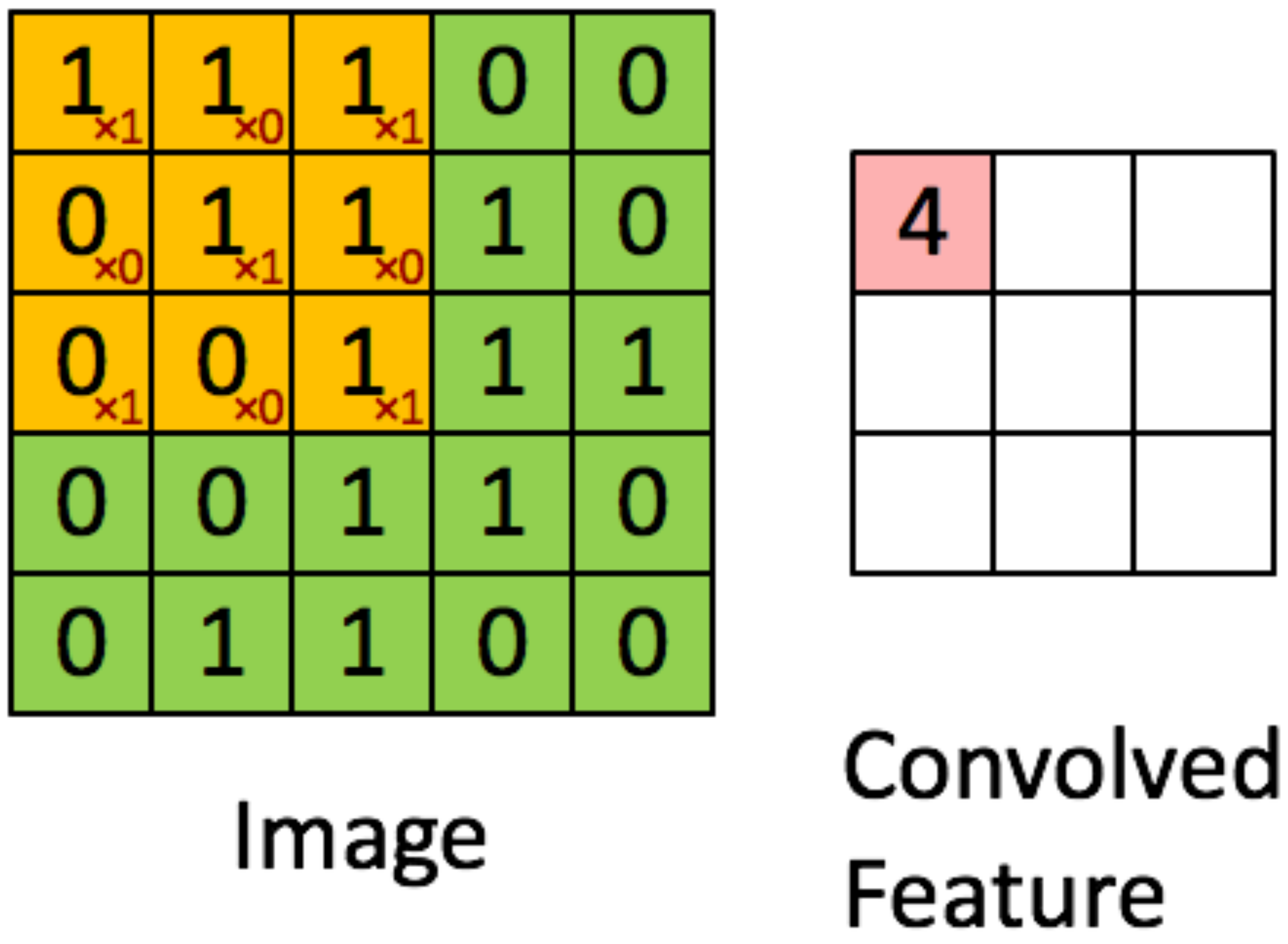
$$\mathbf{x} \in \mathcal{R}^D$$

- Ignores the local correlations in the pixels
  - geometric structure is not exploited in images

CONVOLUTIONAL NEURAL NETWORKS

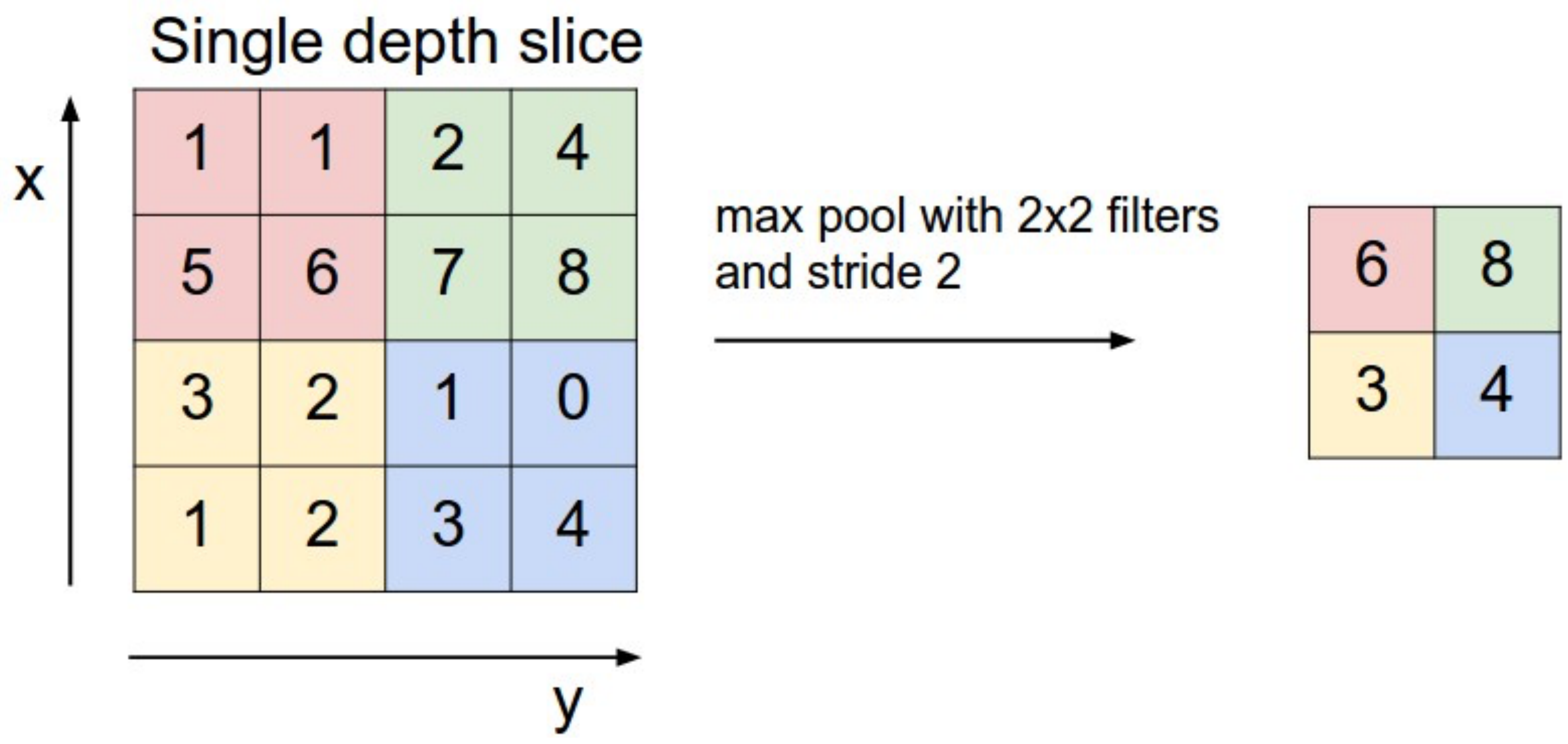
- 2-D convolution

$$A^1(i,j) = \sum_{m=0}^{M} \sum_{n=0}^{N} W^1(m,n) X(i+m, j+n)$$

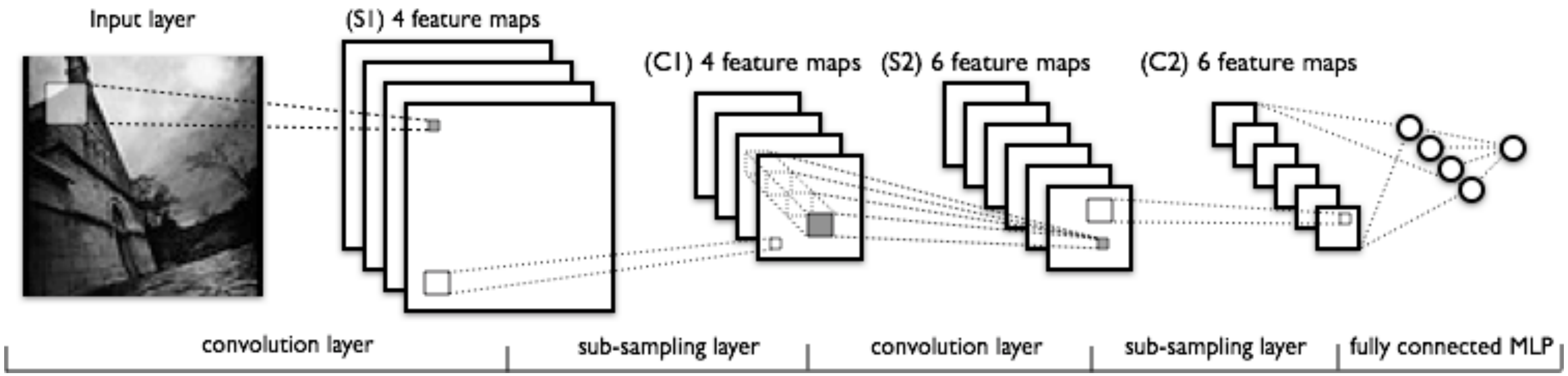


CONVOLUTIONAL NEURAL NETWORKS

- Reduce the size of images after convolution using pooling
  - Keep local maximum



CONVOLUTIONAL NEURAL NETWORKS

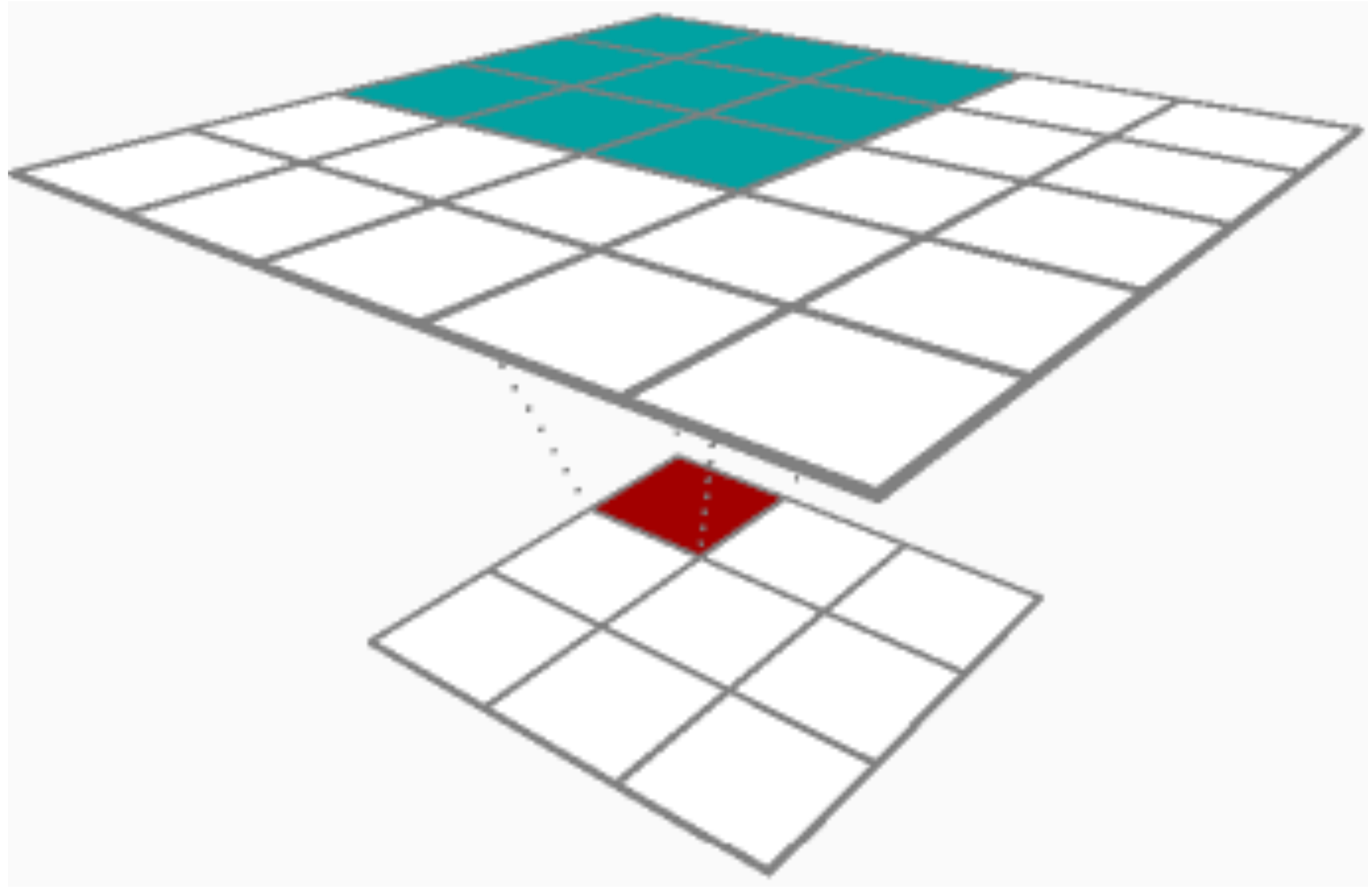


- Multiple levels of filtering and subsampling operations.
- Feature maps are generated at every layer.

BACKPROPAGATION IN CNNS

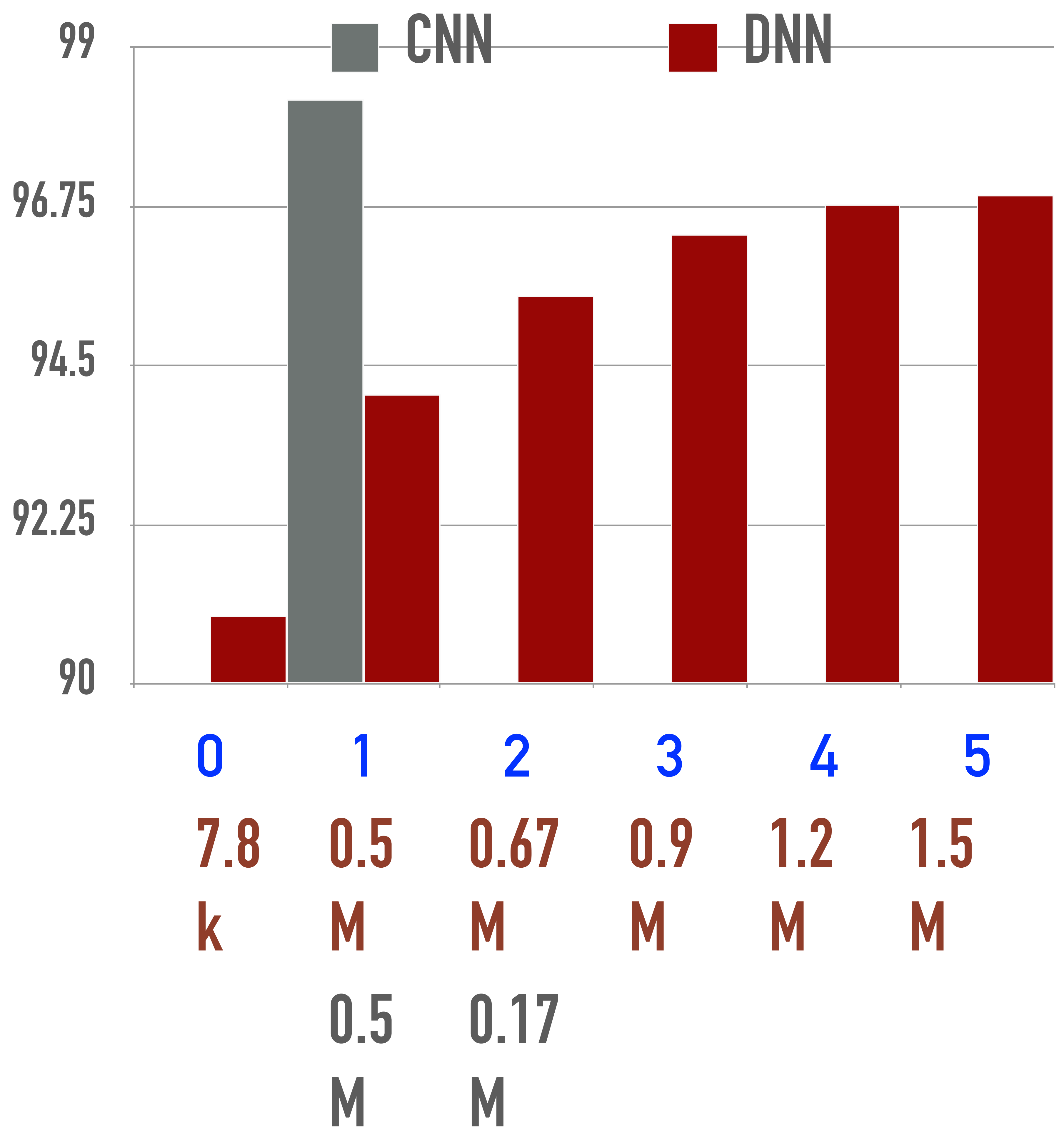
PROPERTIES OF CNN

- Reduce number of parameters
  - due to weight sharing.
  - Depth does not necessarily increase the parameter size.
- Preserving local structure
  - CNN filters operate on local weights
  - Deeper layers
    - capture wider input context.
- Training is more memory intensive
  - Accumulate gradients.



CNNS FOR MNIST

- Providing the right architecture
  - Improves the performance
  - also reduces the number of trainable parameters



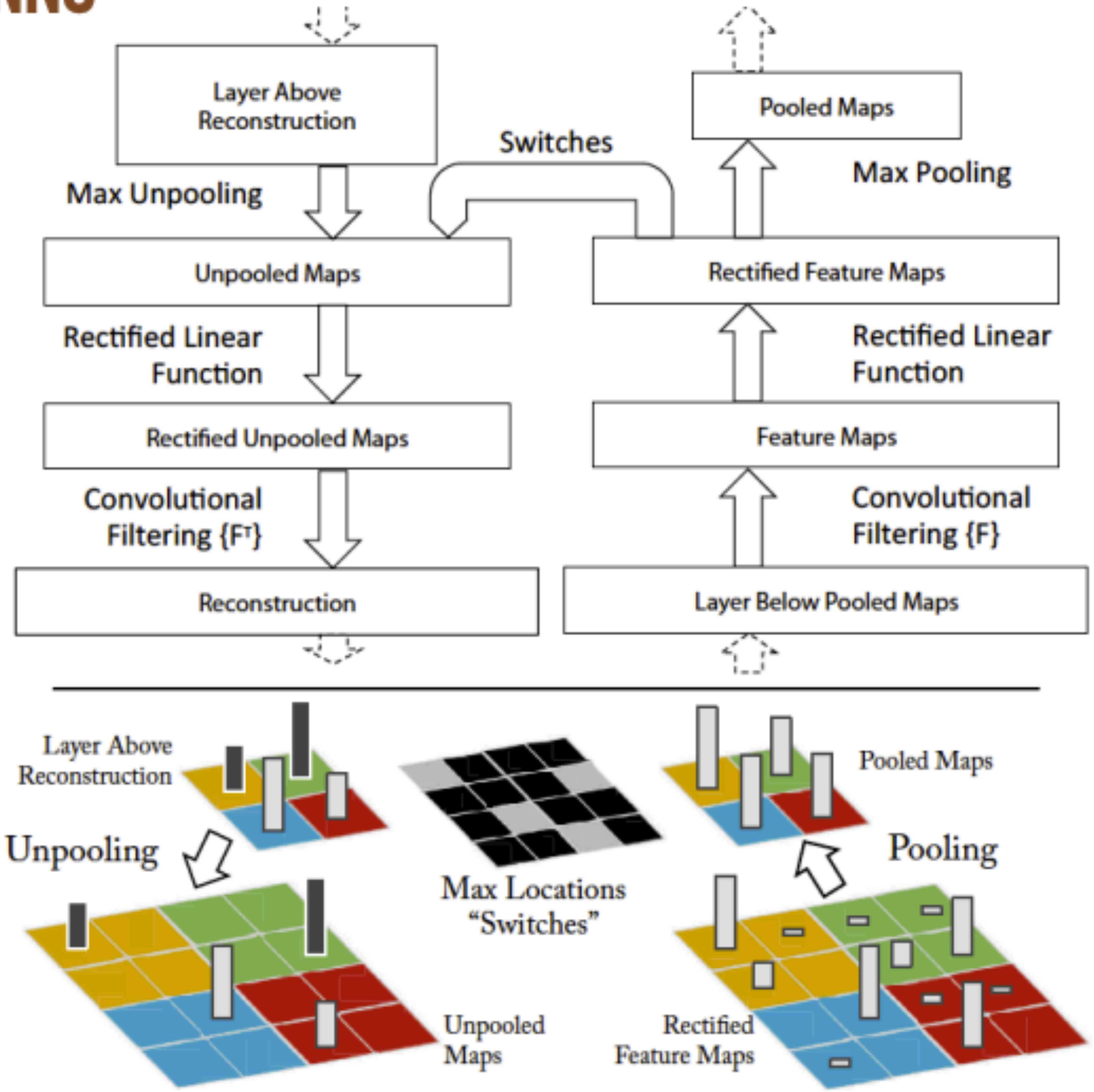
Visualizing and Understanding Convolutional Networks

Matthew D. Zeiler and Rob Fergus

Dept. of Computer Science,
New York University, USA
{zeiler,fergus}@cs.nyu.edu

UNDERSTANDING CNNS USING USING CNNS

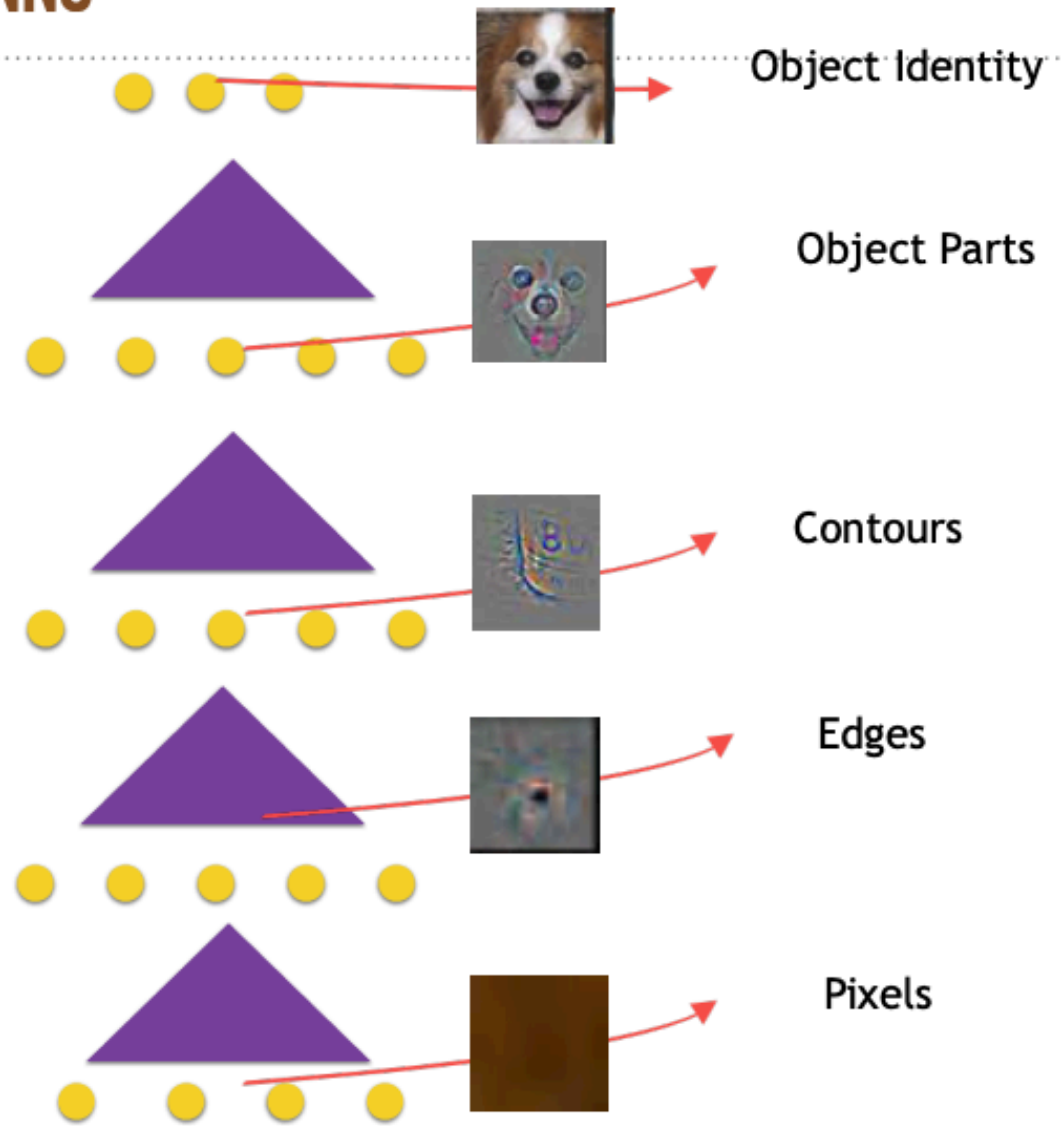
- Analyze a trained model
  - Take a model which is trained to perform object classification
  - Map the layer outputs back to the original space of images
  - Analyze the reconstruction outputs on a held out data



UNDERSTANDING CNNS USING USING CNNS

- Model learns a hierarchy
  - Earlier layers perform simpler tasks like edge detection.
  - Final layers perform complex tasks like object parts

- **Deep networks perform hierarchical data abstraction.**



Recurrent Neural Networks

INTRODUCTION

- The standard DNN/CNN paradigms
  - (x,y) - ordered pair of data vectors/images (x) and target (y)
- Moving to sequence data
  - (x(t),y(t)) where this could be sequence to sequence mapping task.
  - (x(t),y) where this could be a sequence to vector mapping task.
    - Input features / output targets are correlated in time.
    - Unlike standard models where each pair is independent.
    - Need to model dependencies in the sequence over time.

WHY DO NEED RECURRENT MODELS

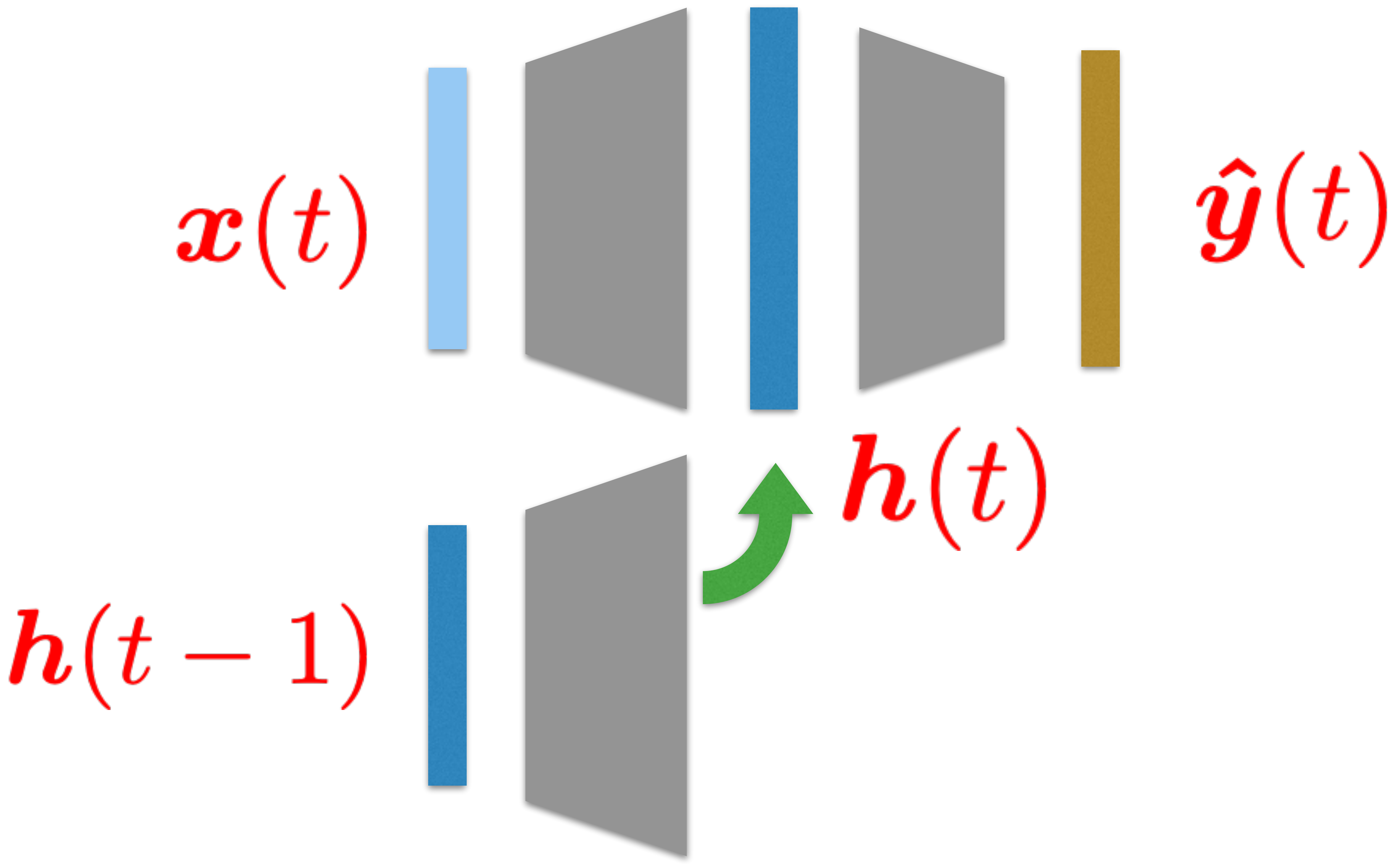
- An interesting subset of this problem is where the input alone is a time series or have different indices $\boldsymbol{x}(t), \boldsymbol{y}$

- Examples
  - Text sequences
  - Speech and audio
  - Video sequences
  - ECG/EEG data
  - Wearable sensor data

FIRST ORDER RECURRENCE – HIDDEN LAYER

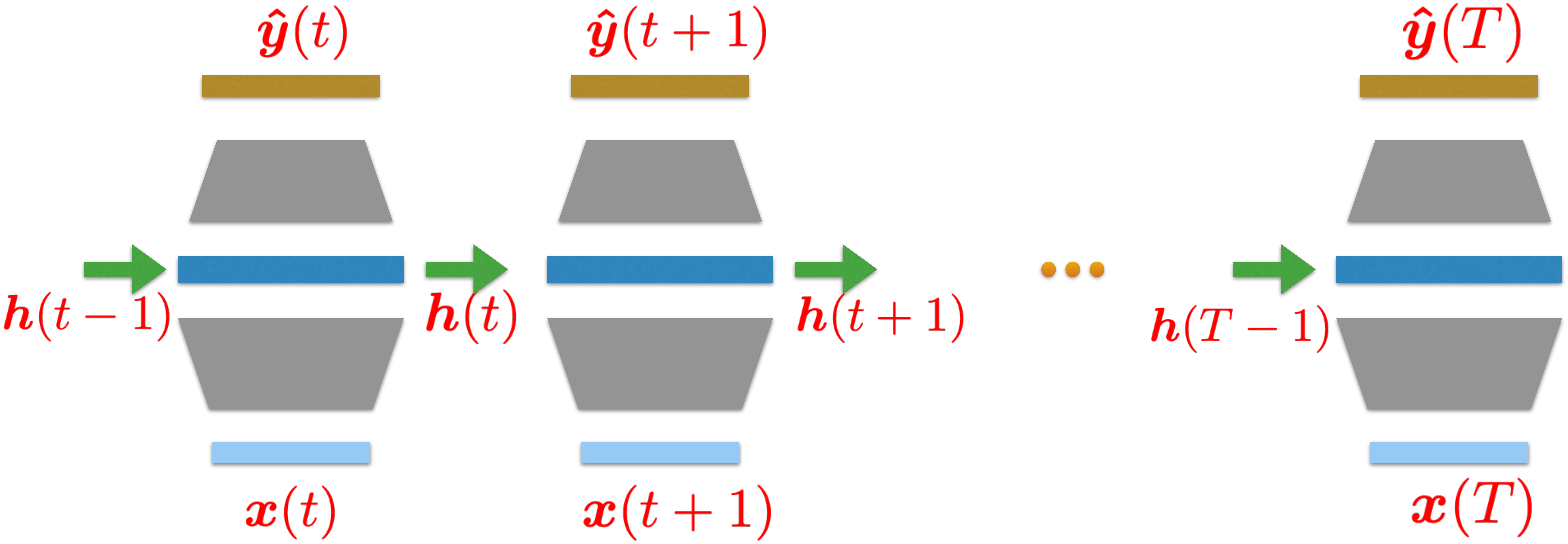
- Making the hidden layer a function of the previous outputs from the hidden layer along with the input

$$\boldsymbol{h}(t) = f\left(\boldsymbol{h}(t-1), \boldsymbol{x}(t)\right)$$



FIRST ORDER RECURRENCE – HIDDEN LAYER

- Making the hidden layer a function of the previous outputs from the hidden layer along with the input.

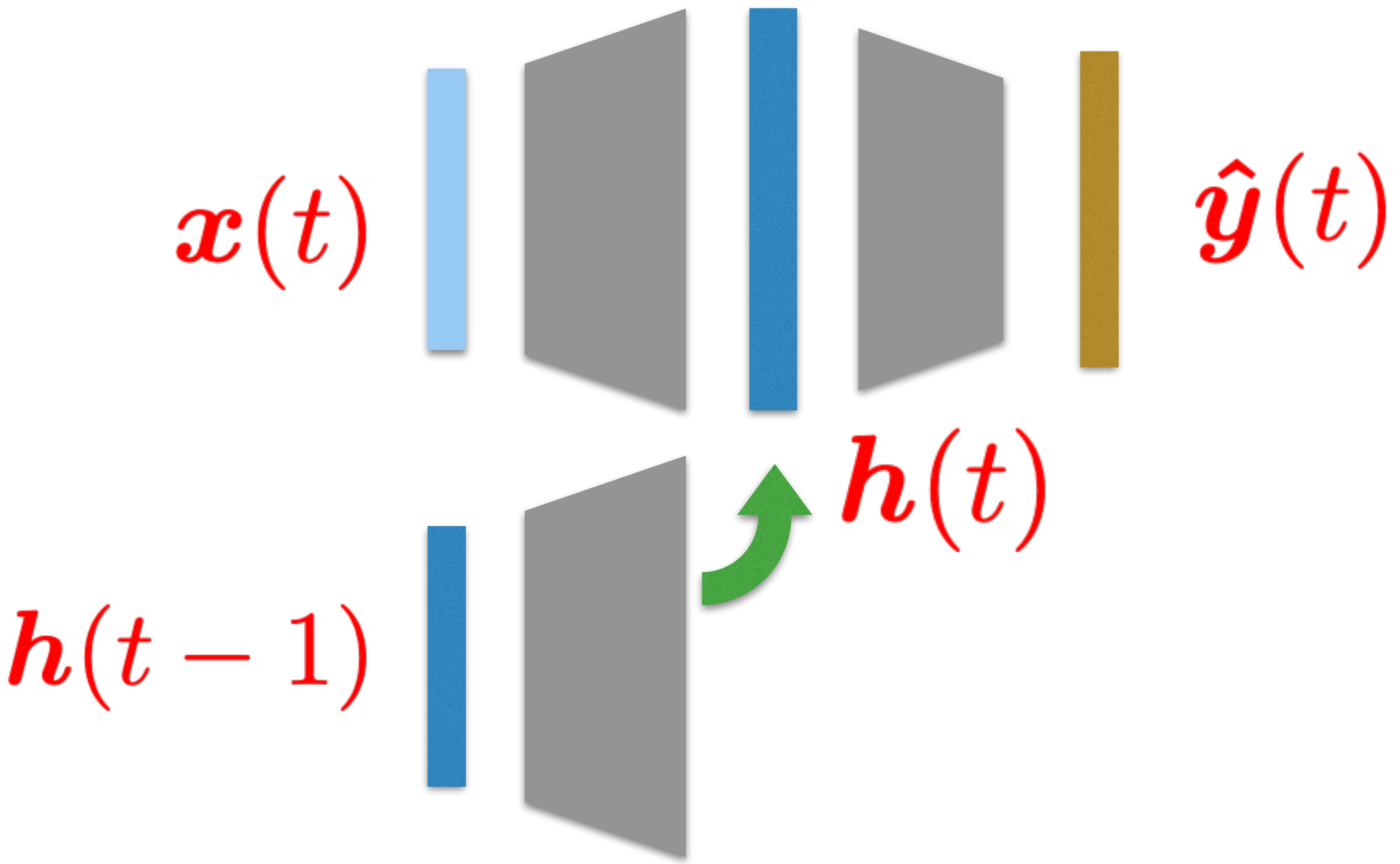


- Makes the hidden layer dependent of previous layer outputs in a recurring fashion.

FIRST ORDER RECURRENCE - HIDDEN LAYER

- Making the hidden layer a function of the previous outputs from the hidden layer along with the input.

$$\boldsymbol{h}(t) = f\left(\boldsymbol{h}(t-1), \boldsymbol{x}(t)\right)$$



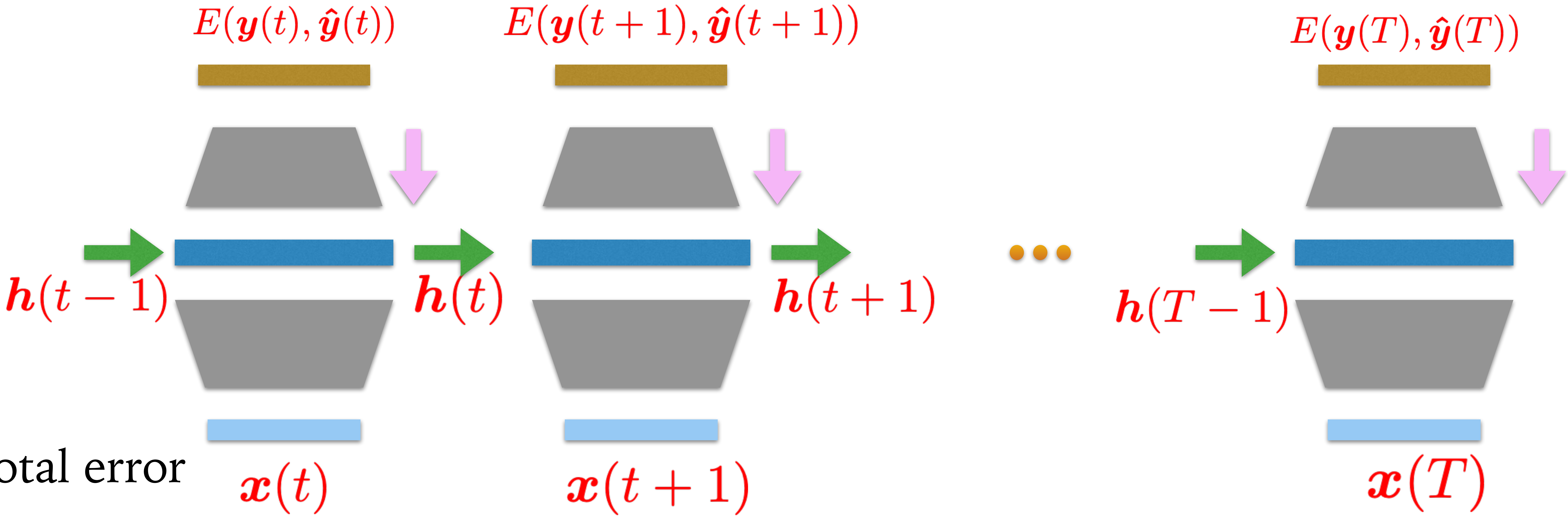
Model Forward Pass - 1 hidden layer

$$\boldsymbol{a}^1(t) = \boldsymbol{W}^1\boldsymbol{x}(t) + \boldsymbol{U}^1\boldsymbol{h}^1(t-1)$$
$$\boldsymbol{h}^1(t) = tanh(\boldsymbol{a}^1(t))$$
$$\boldsymbol{a}^2(t) = \boldsymbol{W}^2\boldsymbol{h}^1(t)$$
$$\hat{\boldsymbol{y}}(t) = S(\boldsymbol{a}^2(t))$$

- Makes the hidden layer dependent of previous layer outputs in a recurring fashion.

ERROR BACKPROPAGATION

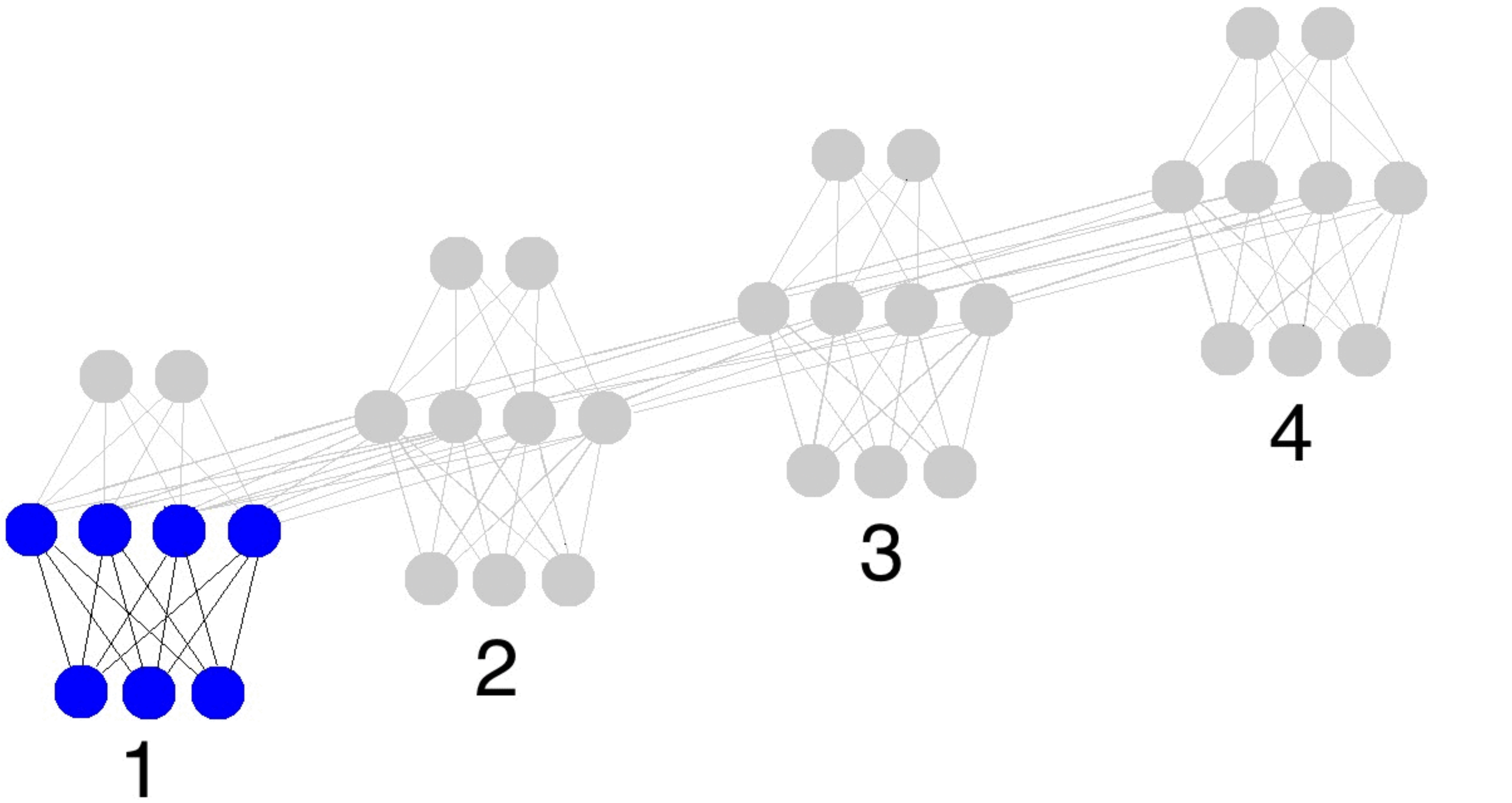
- Error functions are computed at every time-instant



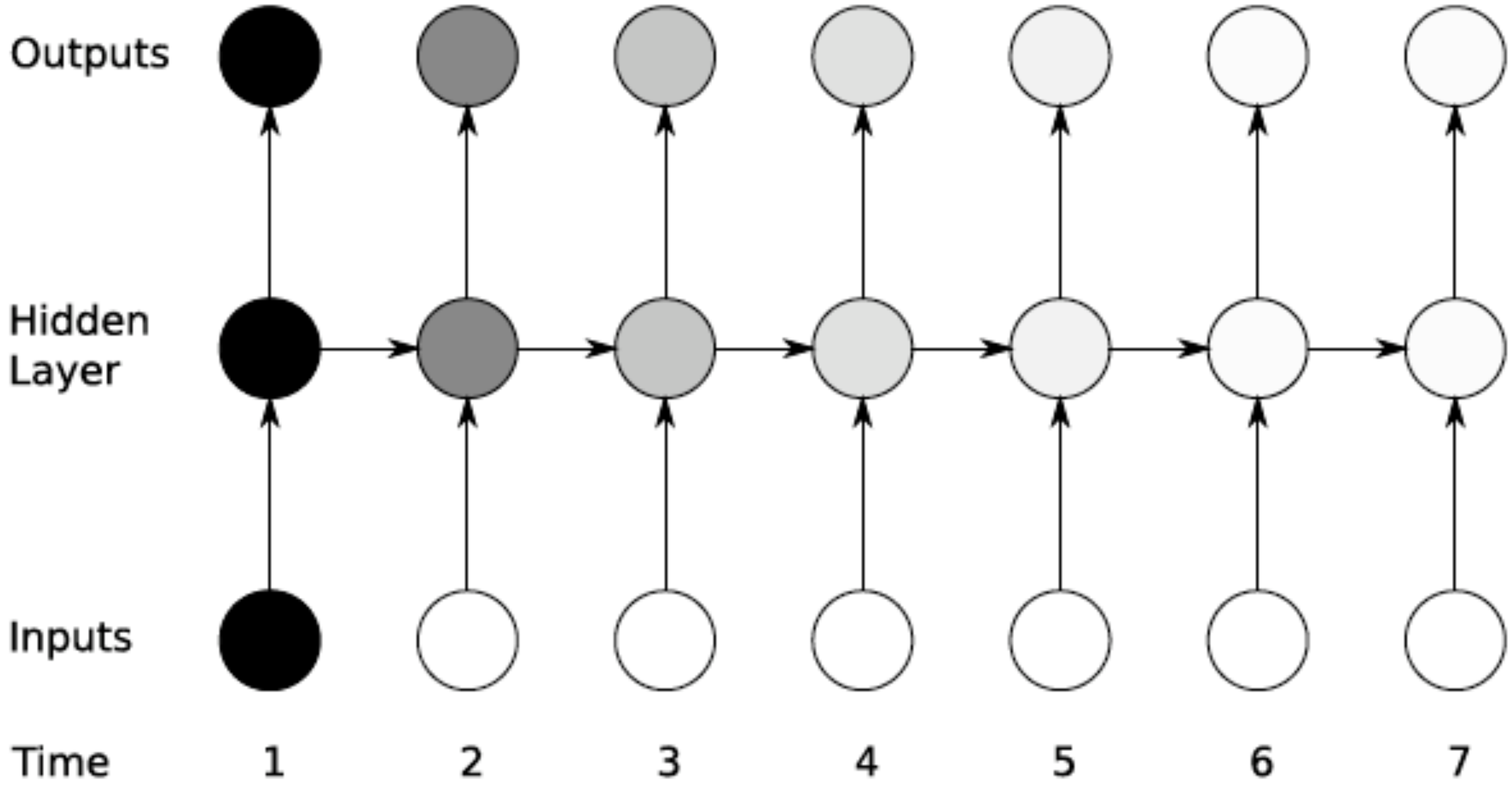
- Total error

$$E = \sum_t E(\boldsymbol{y}(t), \hat{\boldsymbol{y}}(t))$$

BACK PROPAGATION THROUGH TIME

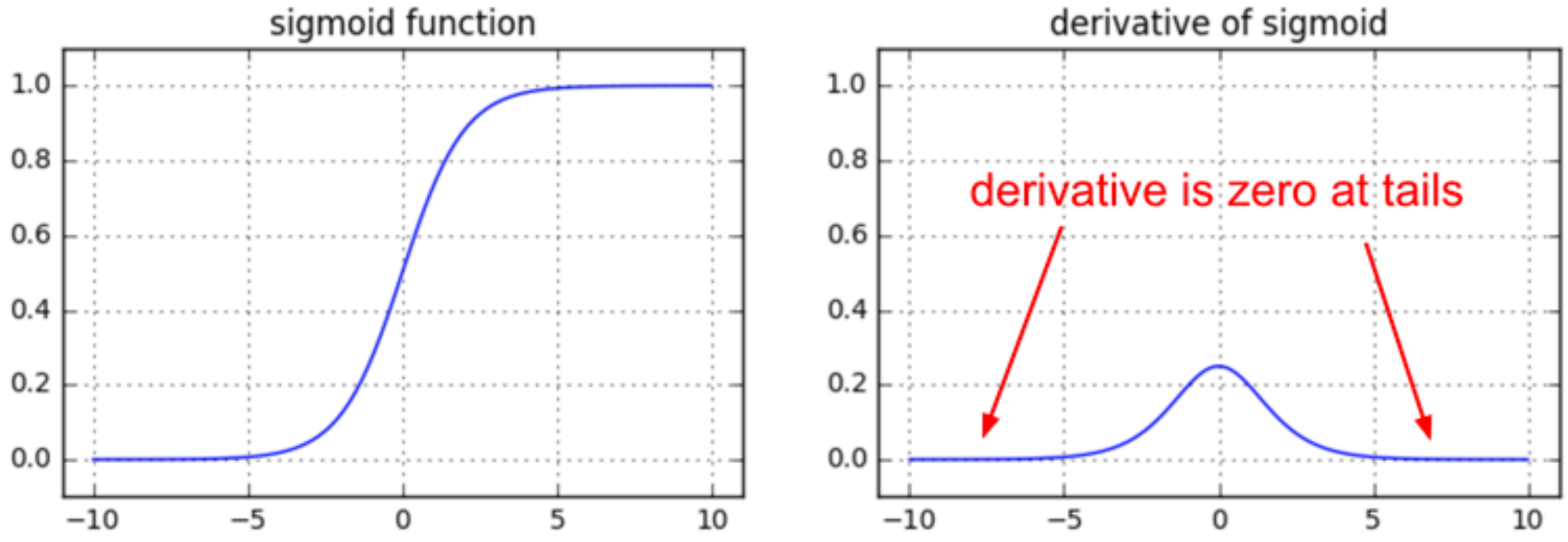


LONG-TERM DEPENDENCY ISSUES

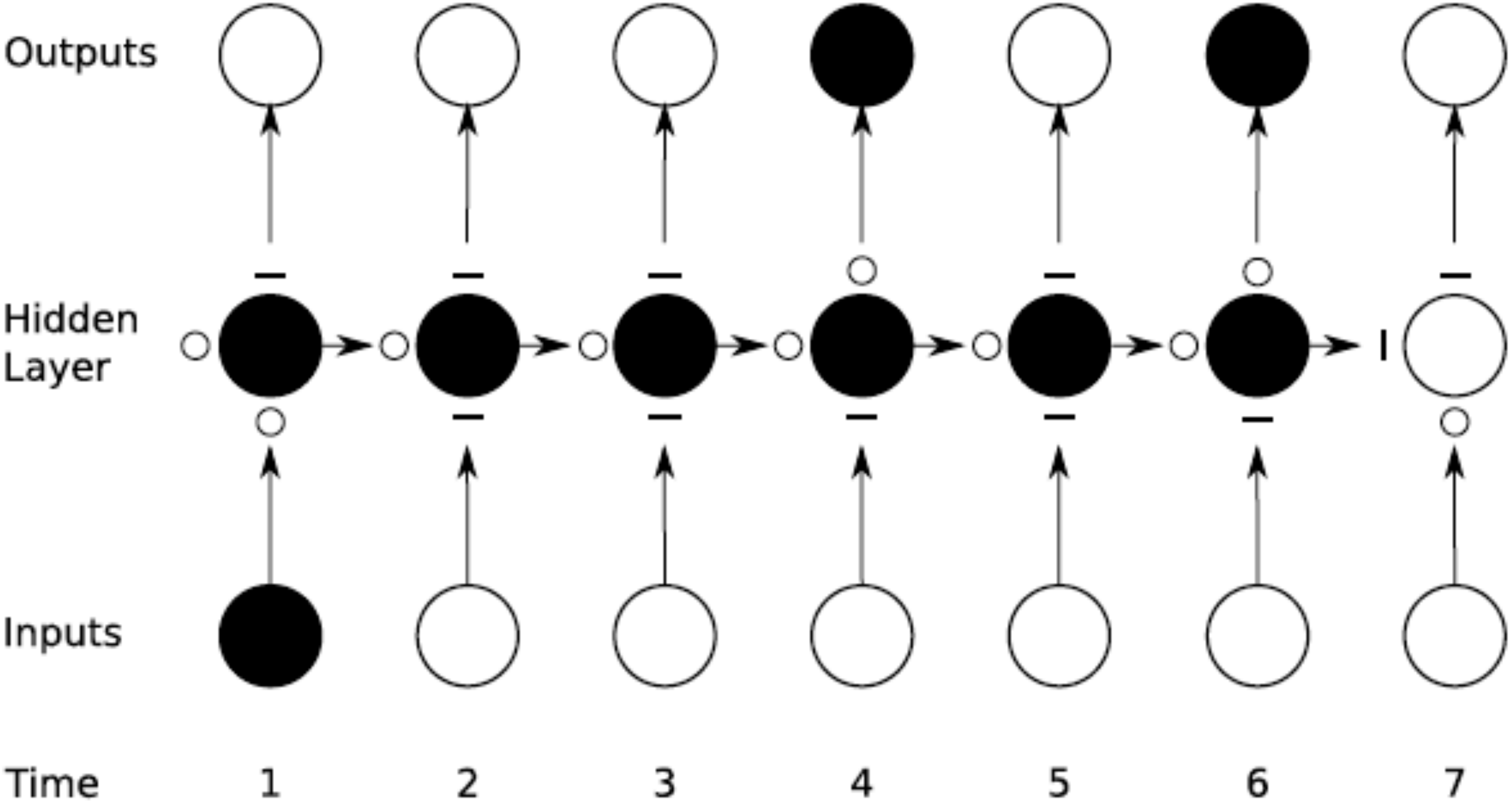


LONG-TERM DEPENDENCY ISSUES

- Gradients tend to vanish or explode



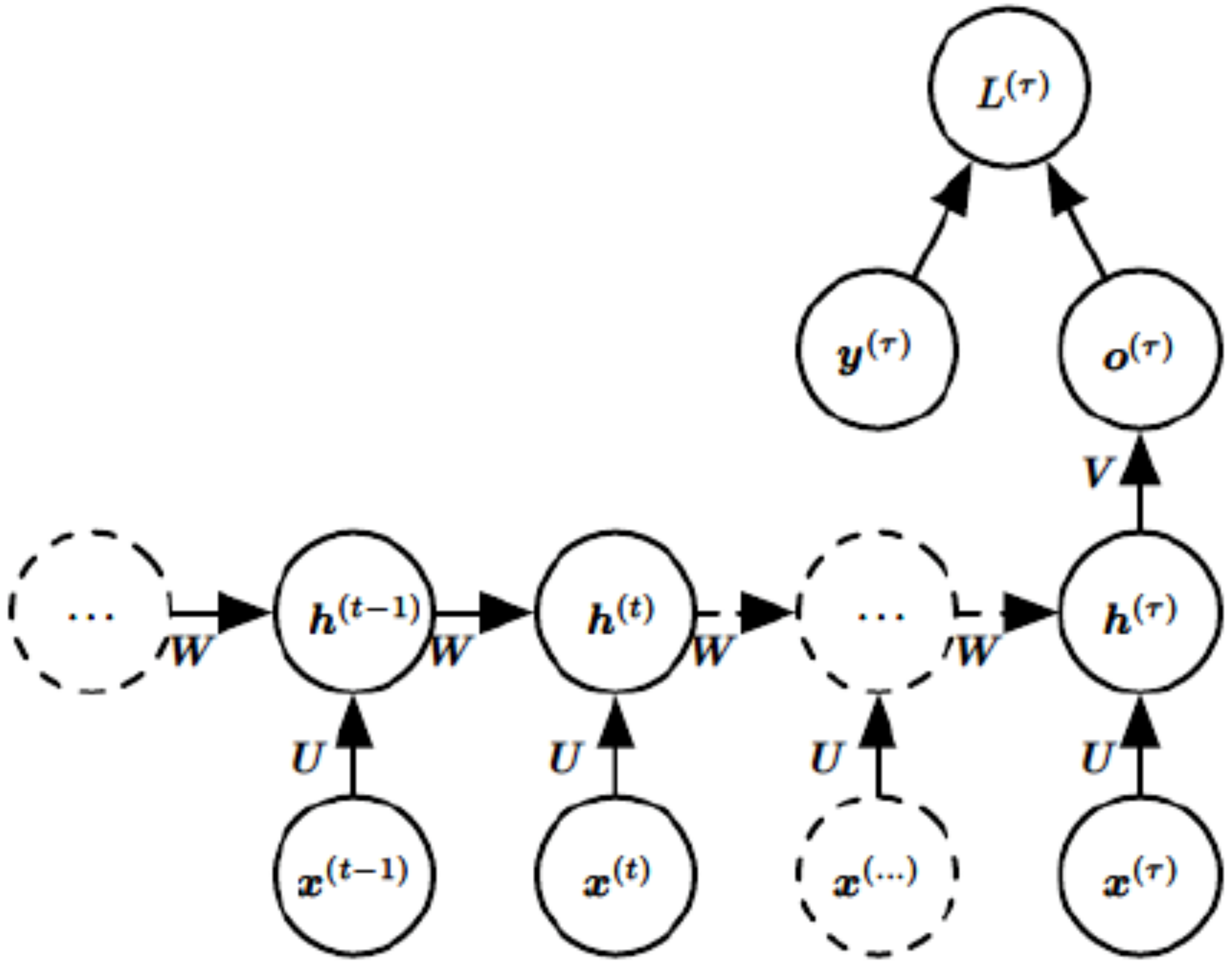
LONG SHORT TERM MEMORY (LSTM) IDEA



MODELING QUESTIONS

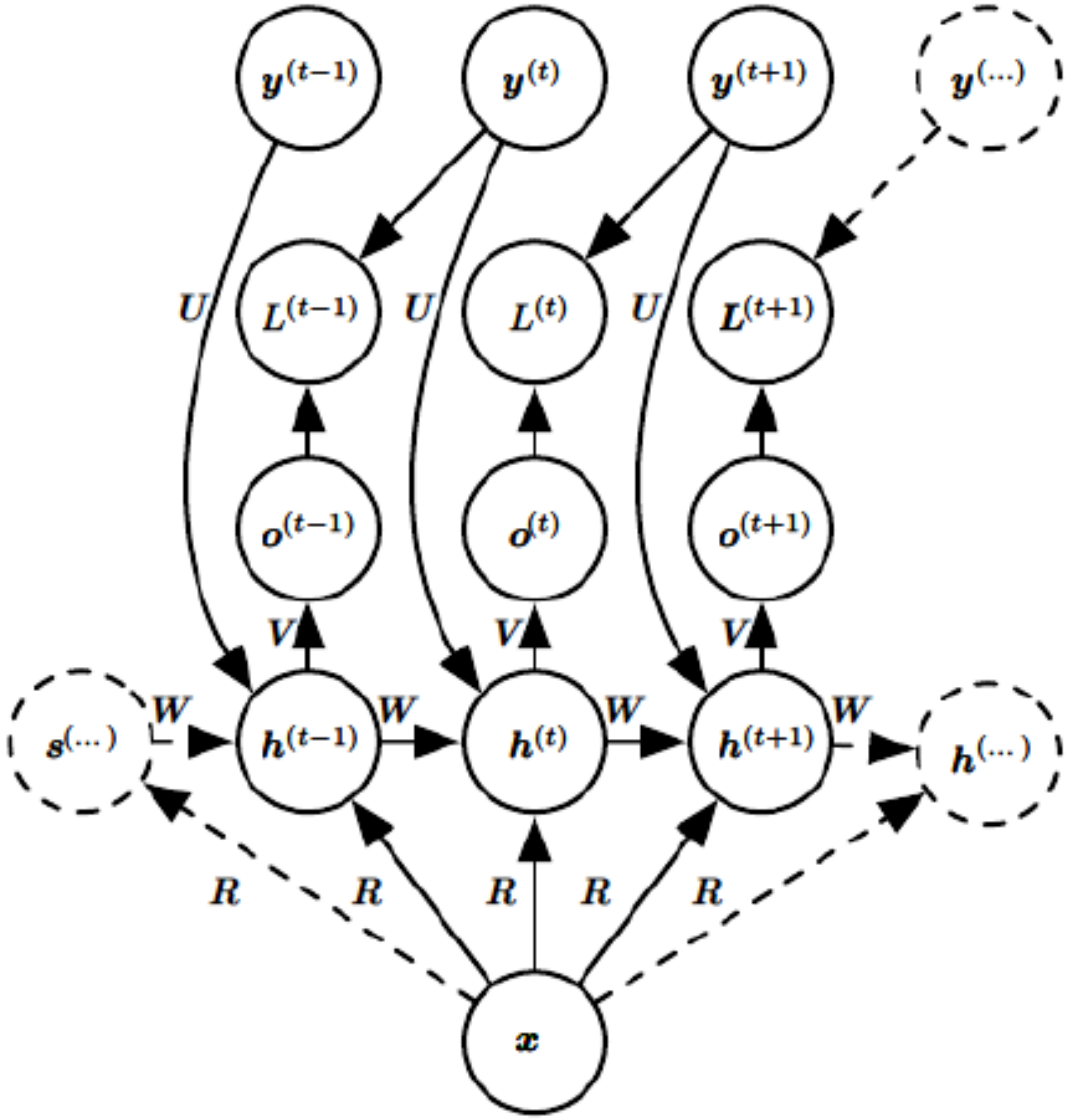
- How can we make adaptable gates with neural networks
  - How can we make gates dependent on the data itself.
    - Gates can be implemented as neural layers with sigmoidal outputs ?
      - Sigmoids can approximate 0-1 functions
    - Modulate the gate output with inputs, hidden layer outputs or outputs

Recurrent Networks



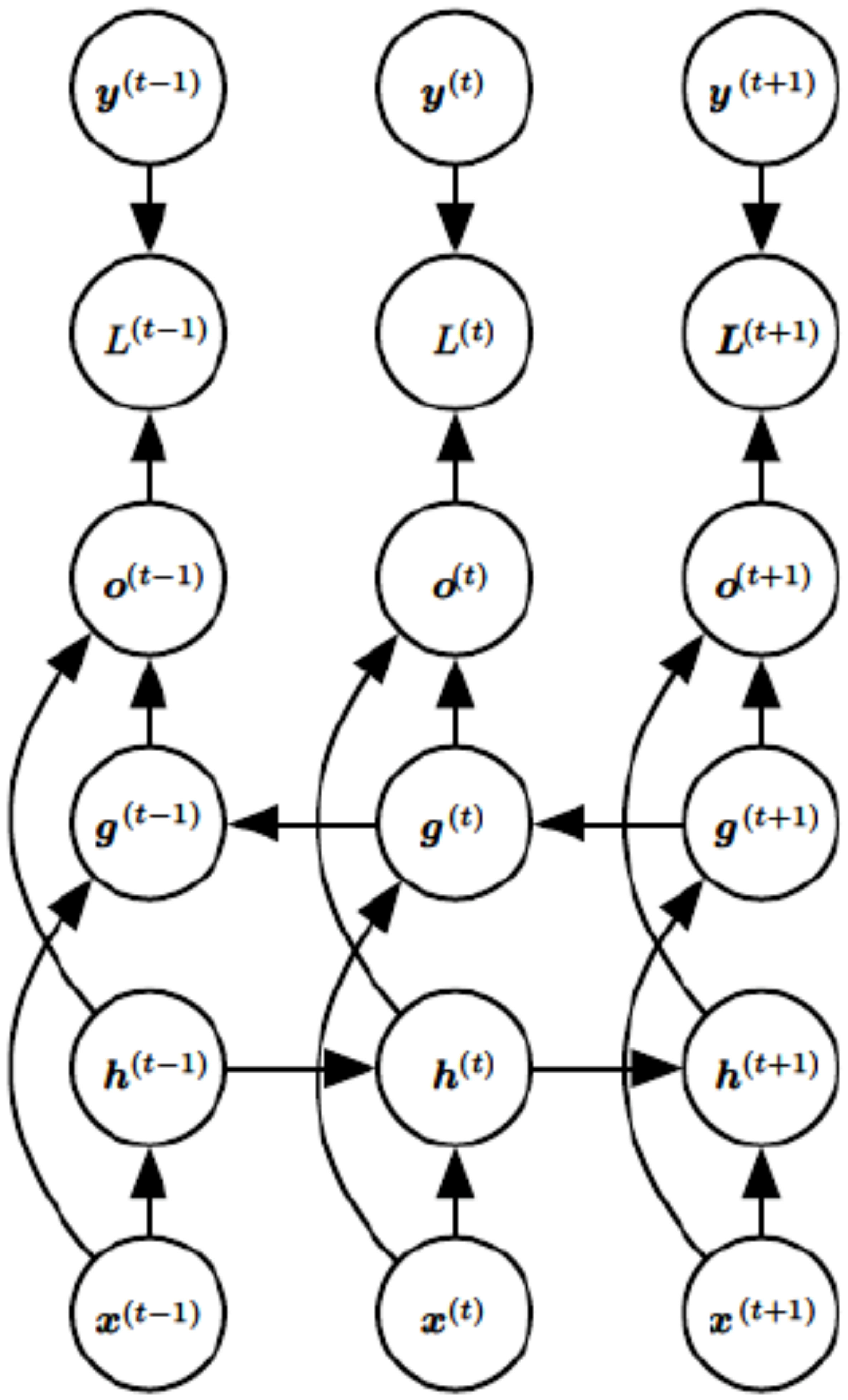
Multiple Input
Single Output

Recurrent Networks



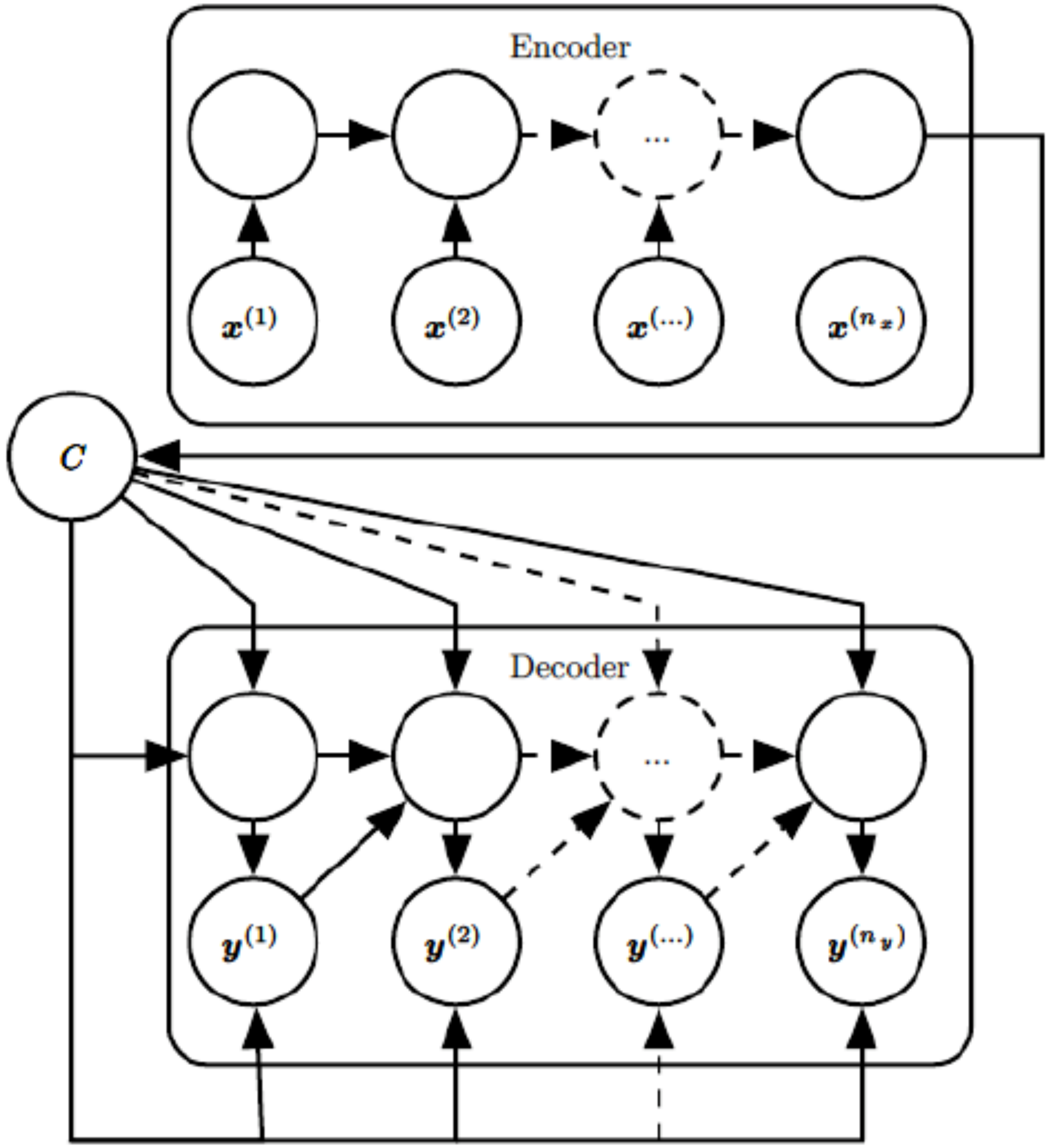
Single Input
Multiple Output

Recurrent Networks



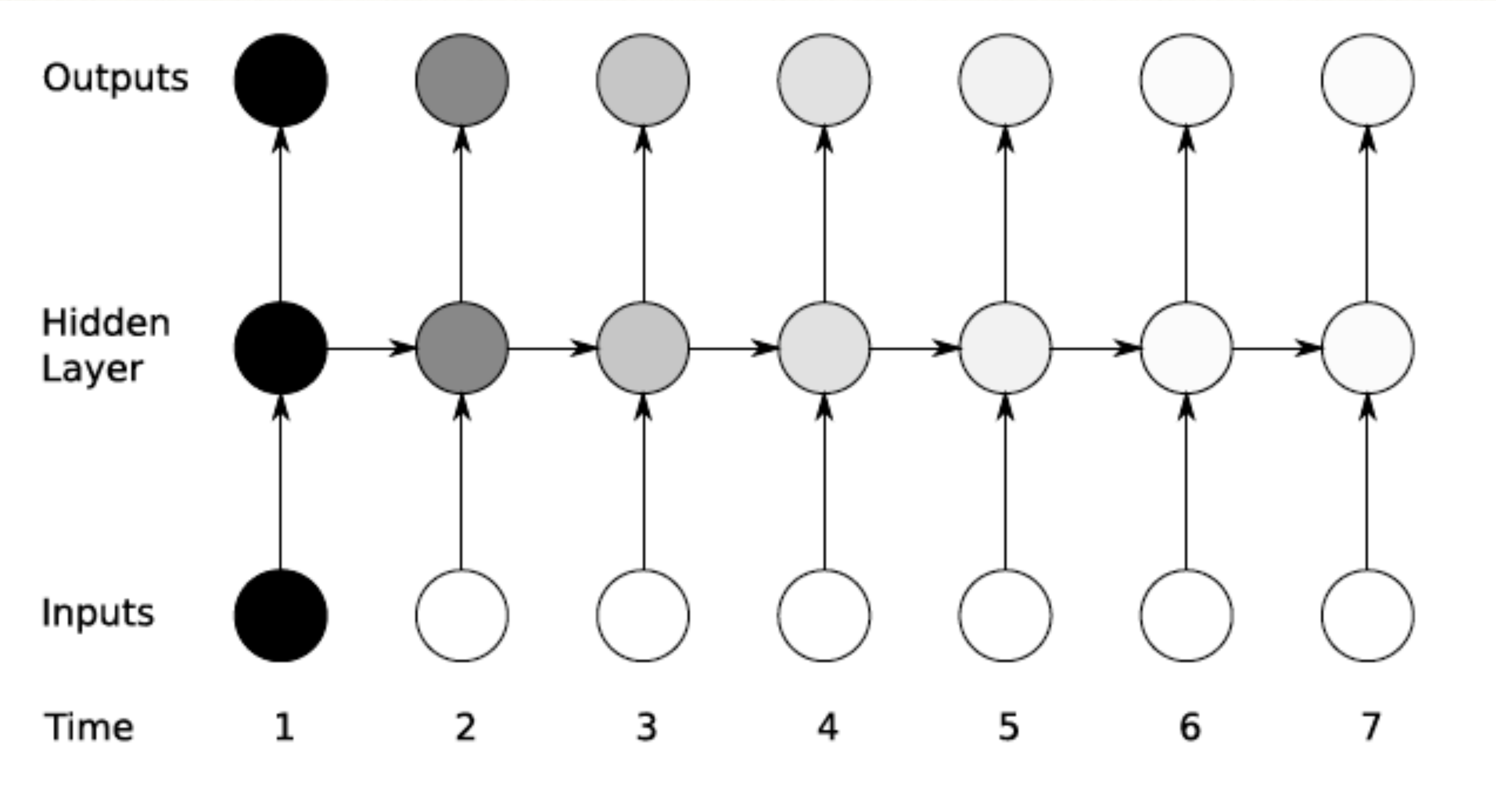
Bi-directional Networks

Recurrent Networks

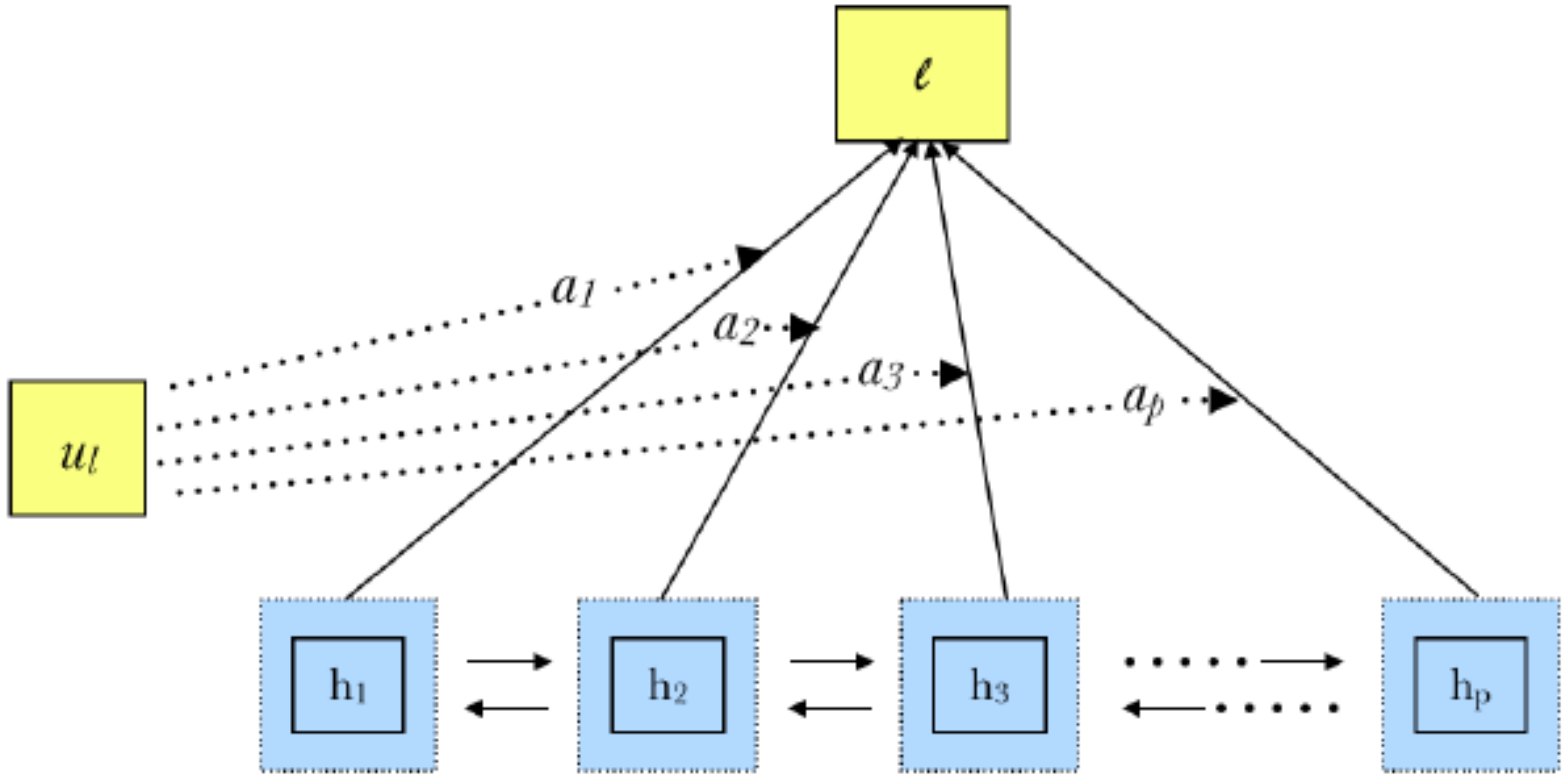


Sequence to Sequence Mapping Networks

Long-term Dependency Issues



Attention in LSTM Networks



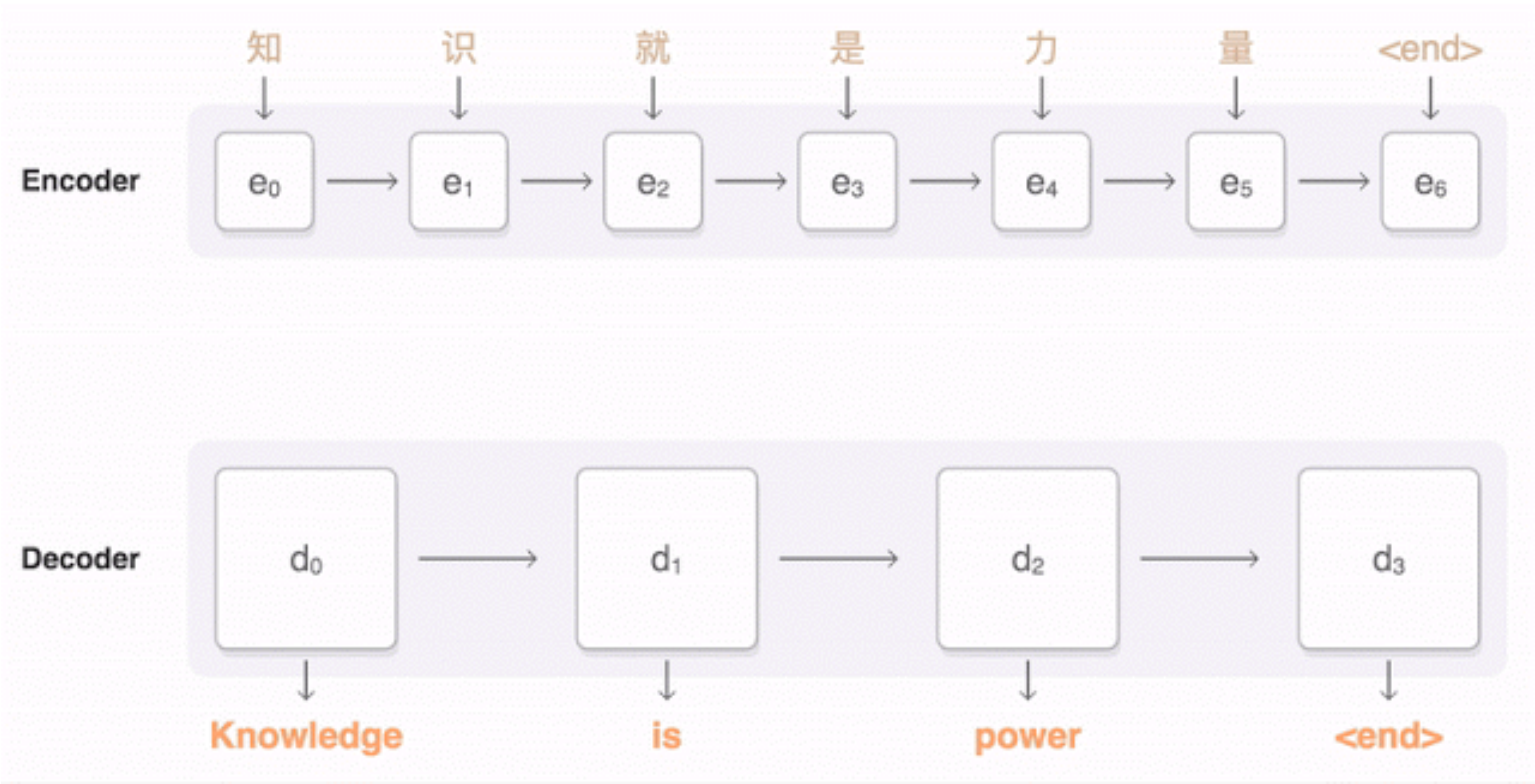
$$\mathbf{u}_t = tanh(\mathbf{W}_l \mathbf{h}_t + \mathbf{b}_l)$$

$$a_t = \frac{exp(\mathbf{u}_t^T \mathbf{u}_l)}{\sum_t exp(\mathbf{u}_t^T \mathbf{u}_l)}$$

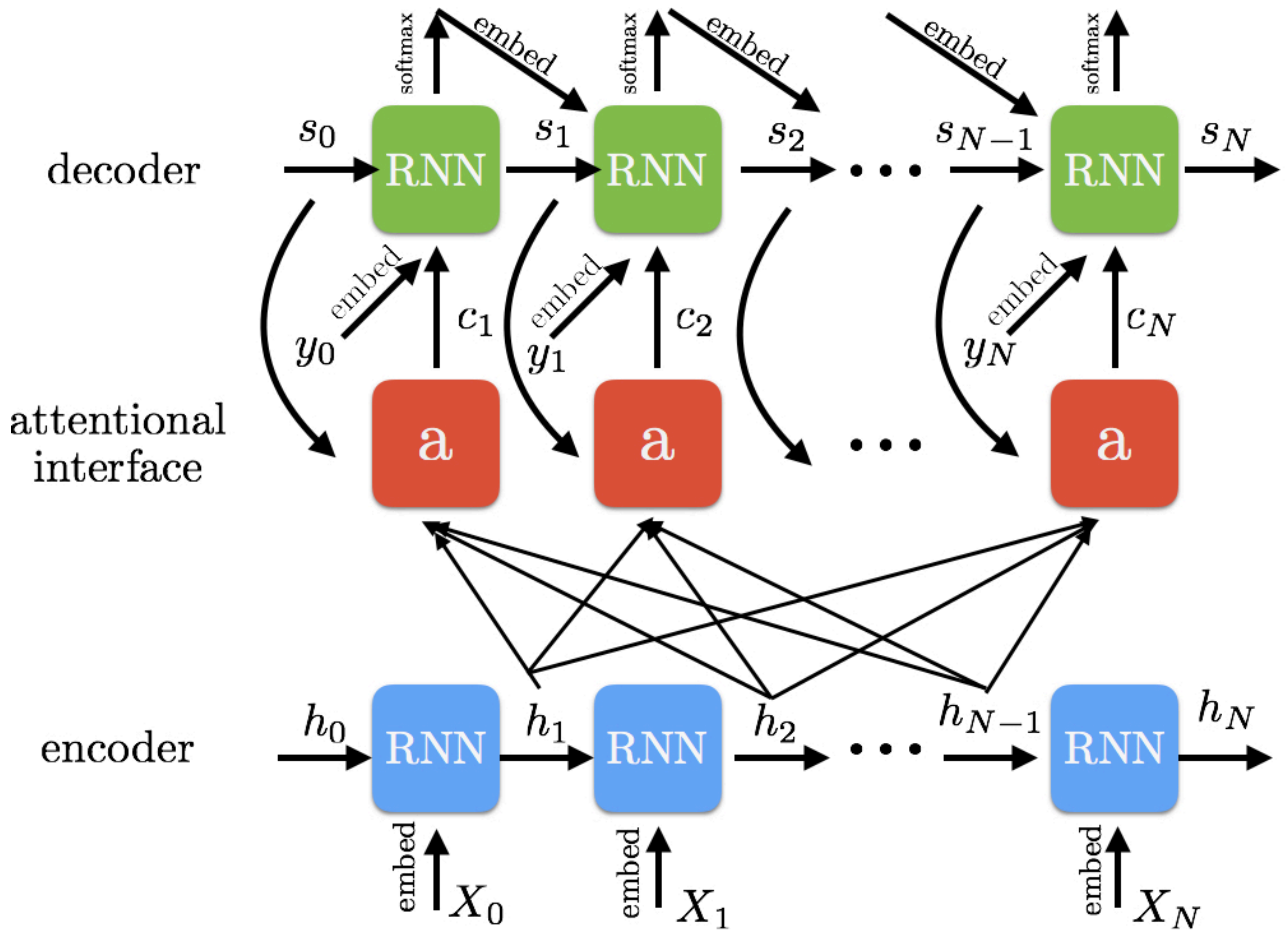
$$\boldsymbol{l} = \sum_t a_t \mathbf{h}_t$$

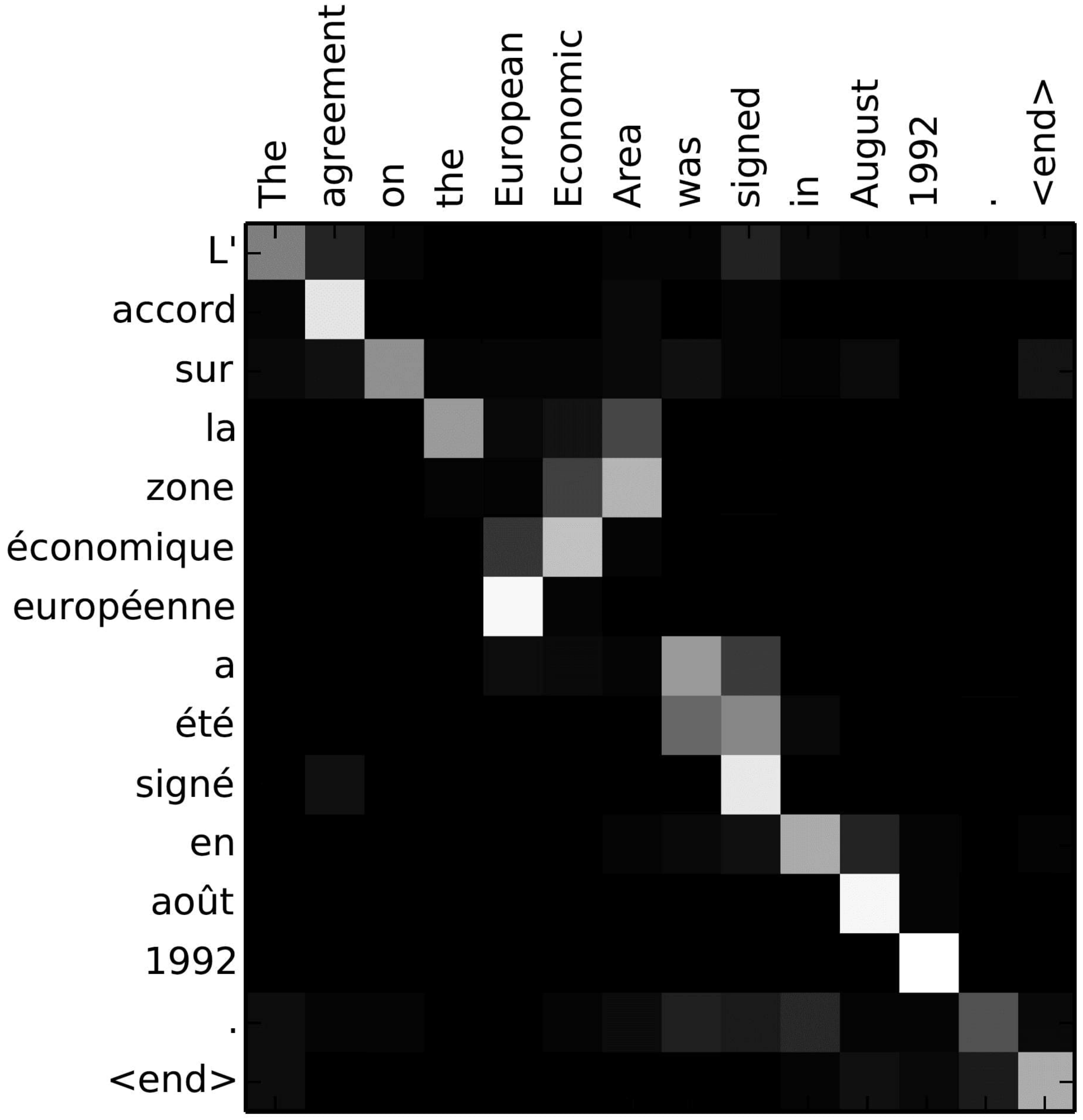
- Attentions allows a mechanism to add relevance
  - Certain regions of the audio have more importance than the rest for the task at hand.

Encoder - Decoder Networks with Attention



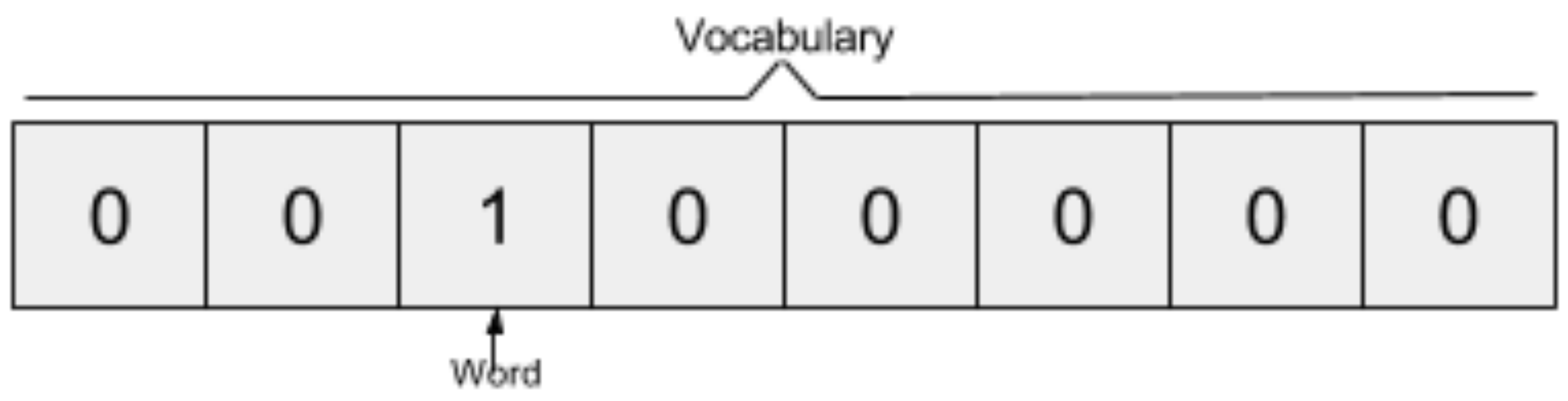
Self-Attention Models





Natural language processing - representations of text

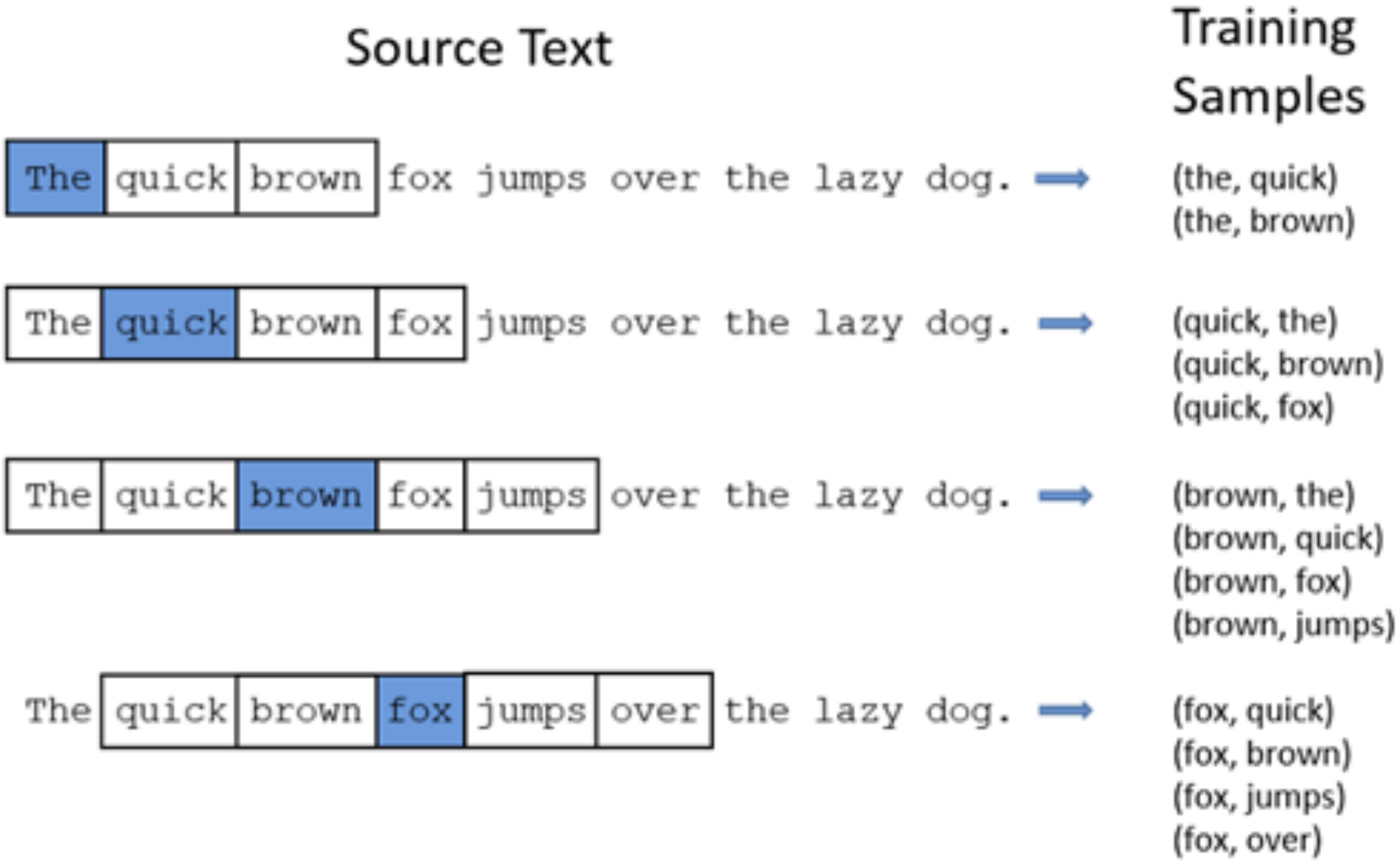
- Converting text into fixed representations in a vector space.



- Using one-hot vectors is really high dimensional.

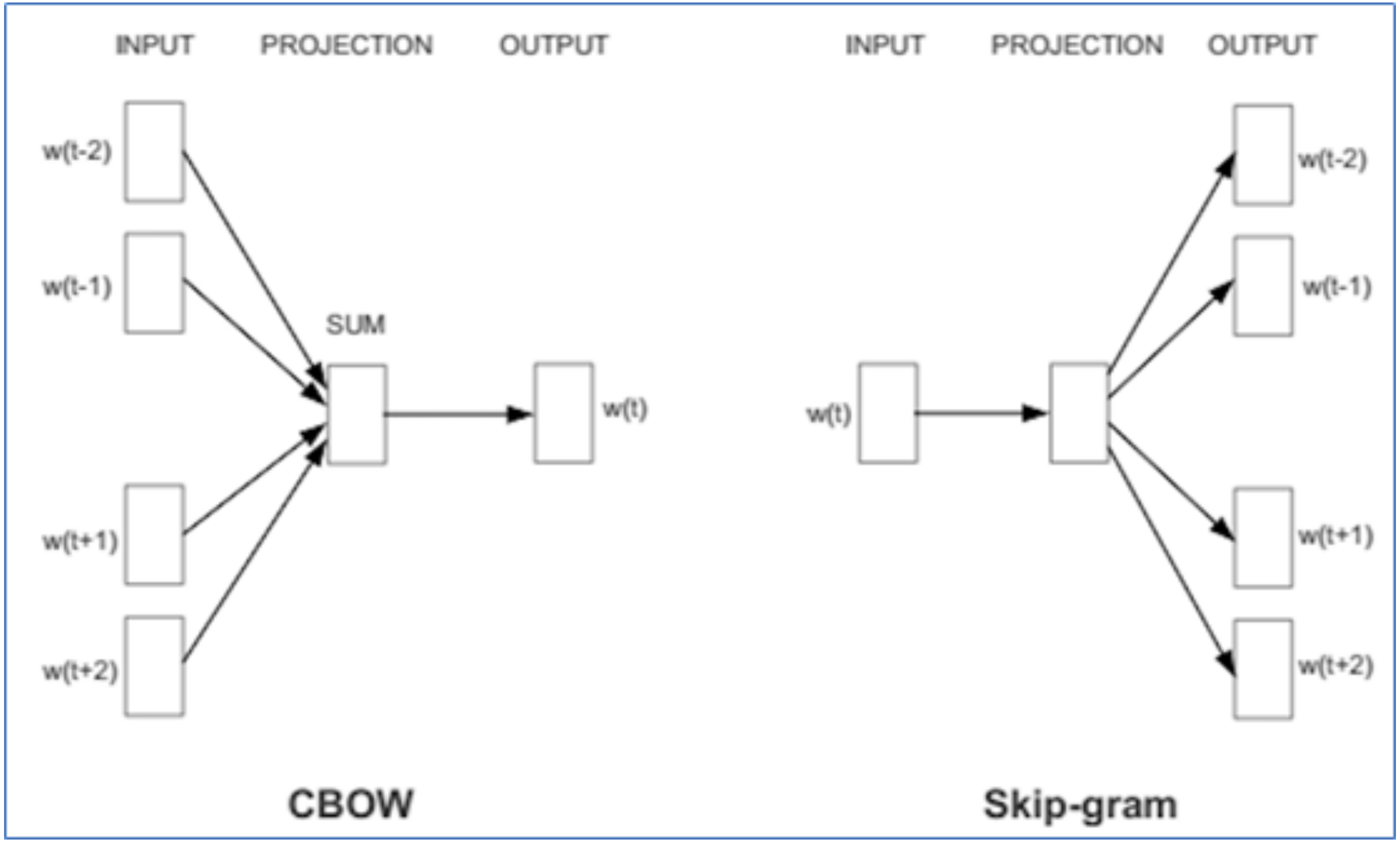
- Need a concise word representations that embeds semantics.

word2vec models as text representations

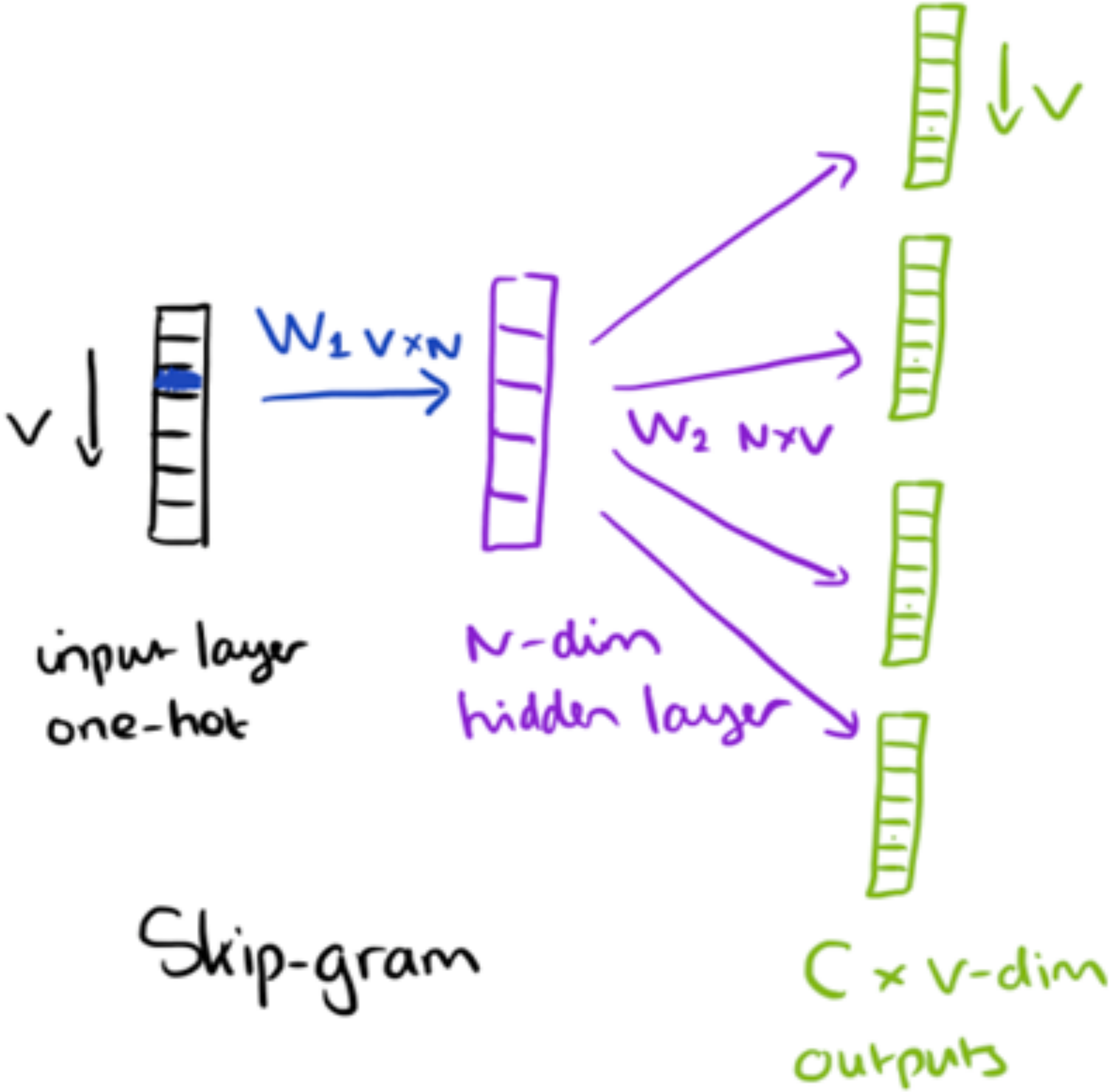


word2vec representations

word2vec models as text representations

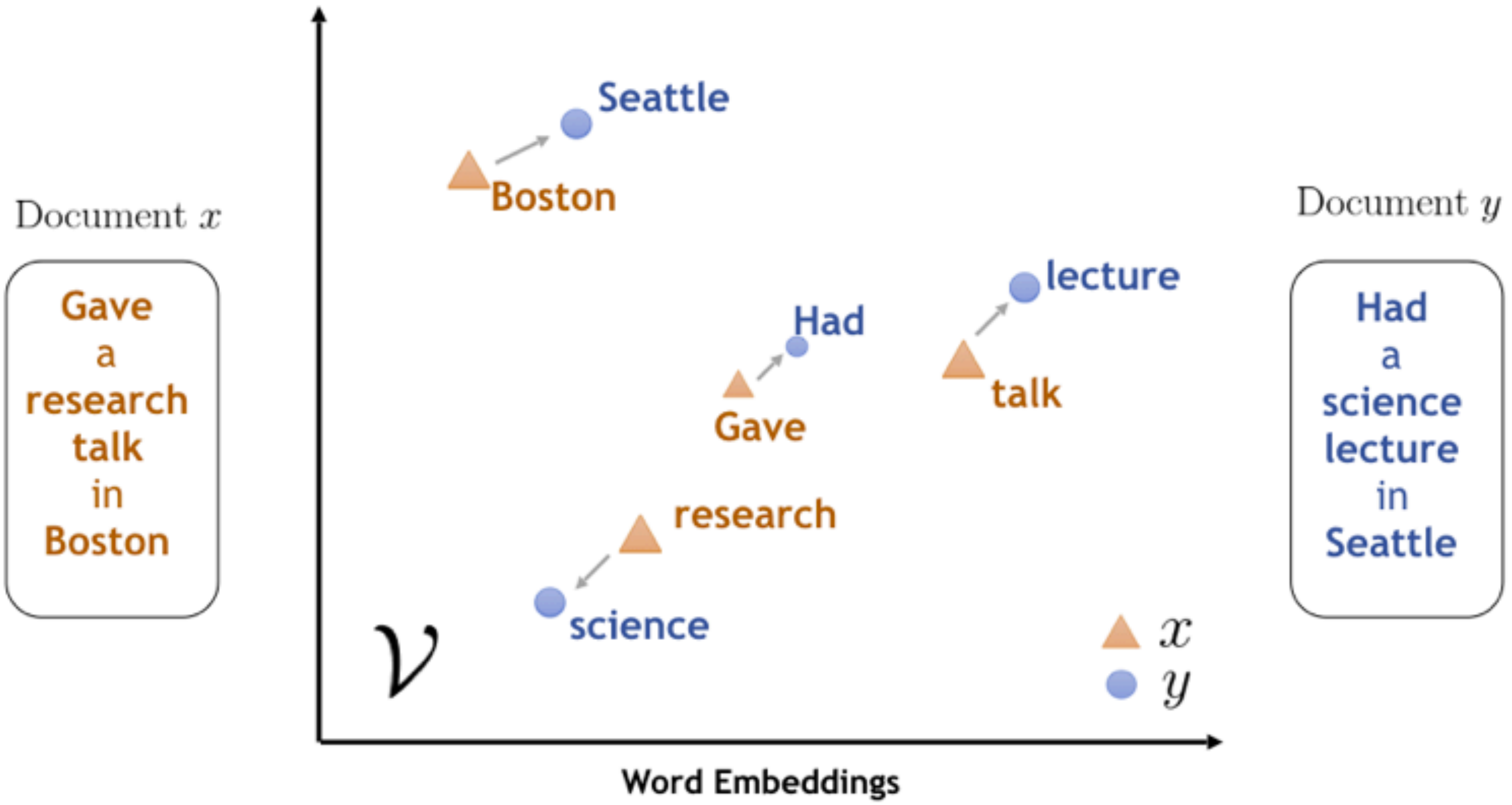


word2vec representations



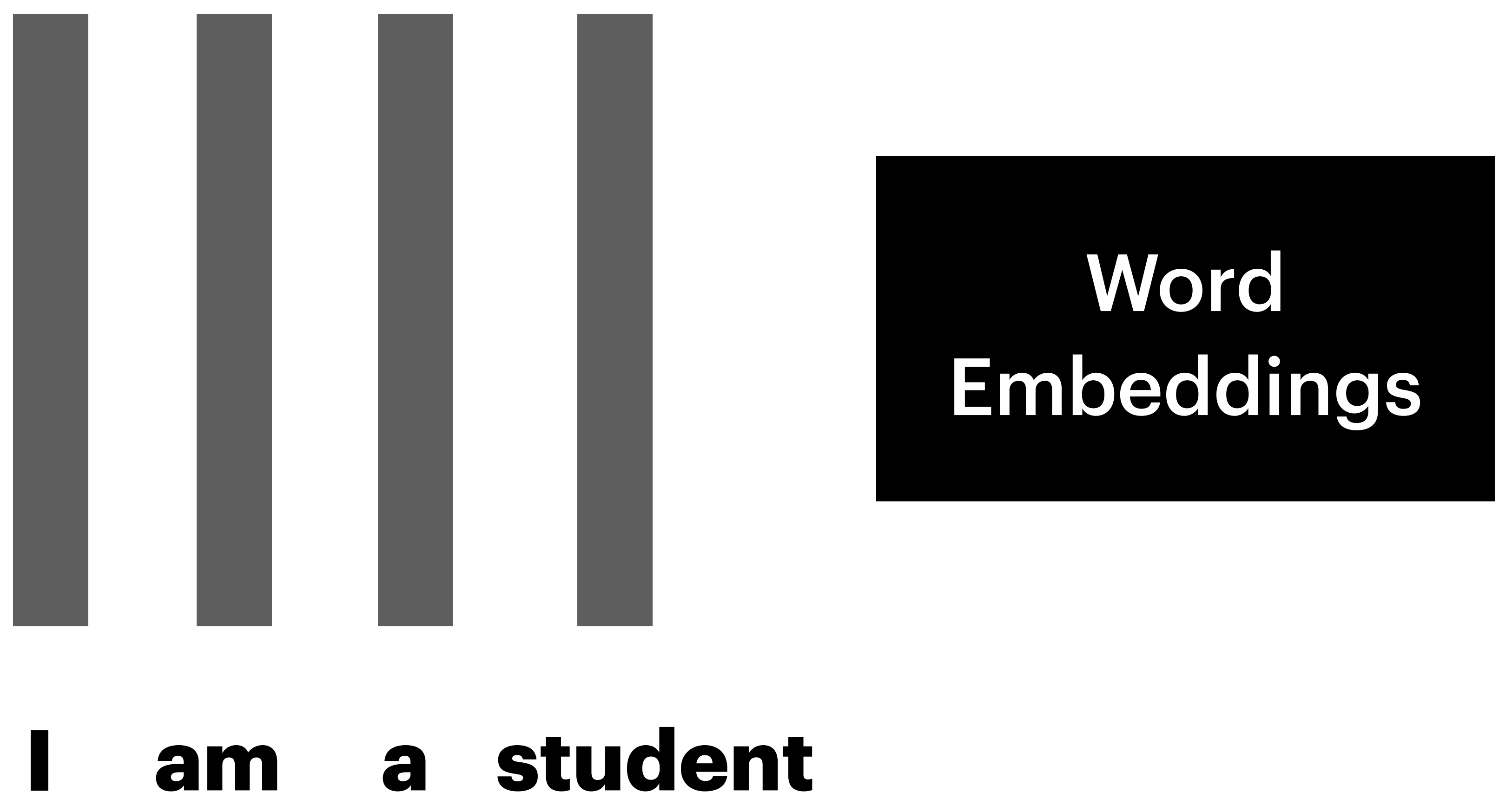
$$\underset{\substack{\text{input}\\ 1\times V}}{\begin{bmatrix} 0 & 1 & 0 \end{bmatrix}} \underset{W_1}{\overset{\substack{W_1\\ V\times N}}{\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{bmatrix}}} = \underset{\substack{\text{hidden}\\ 1\times N}}{\begin{bmatrix} e & f & g & h \end{bmatrix}}$$

word2vec visualization



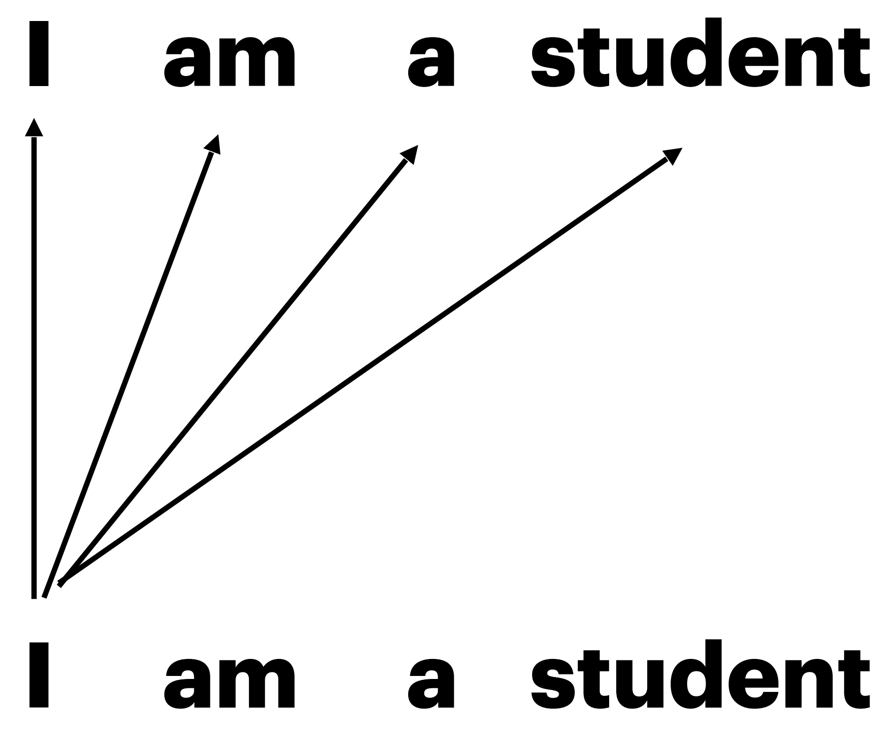
Transformers

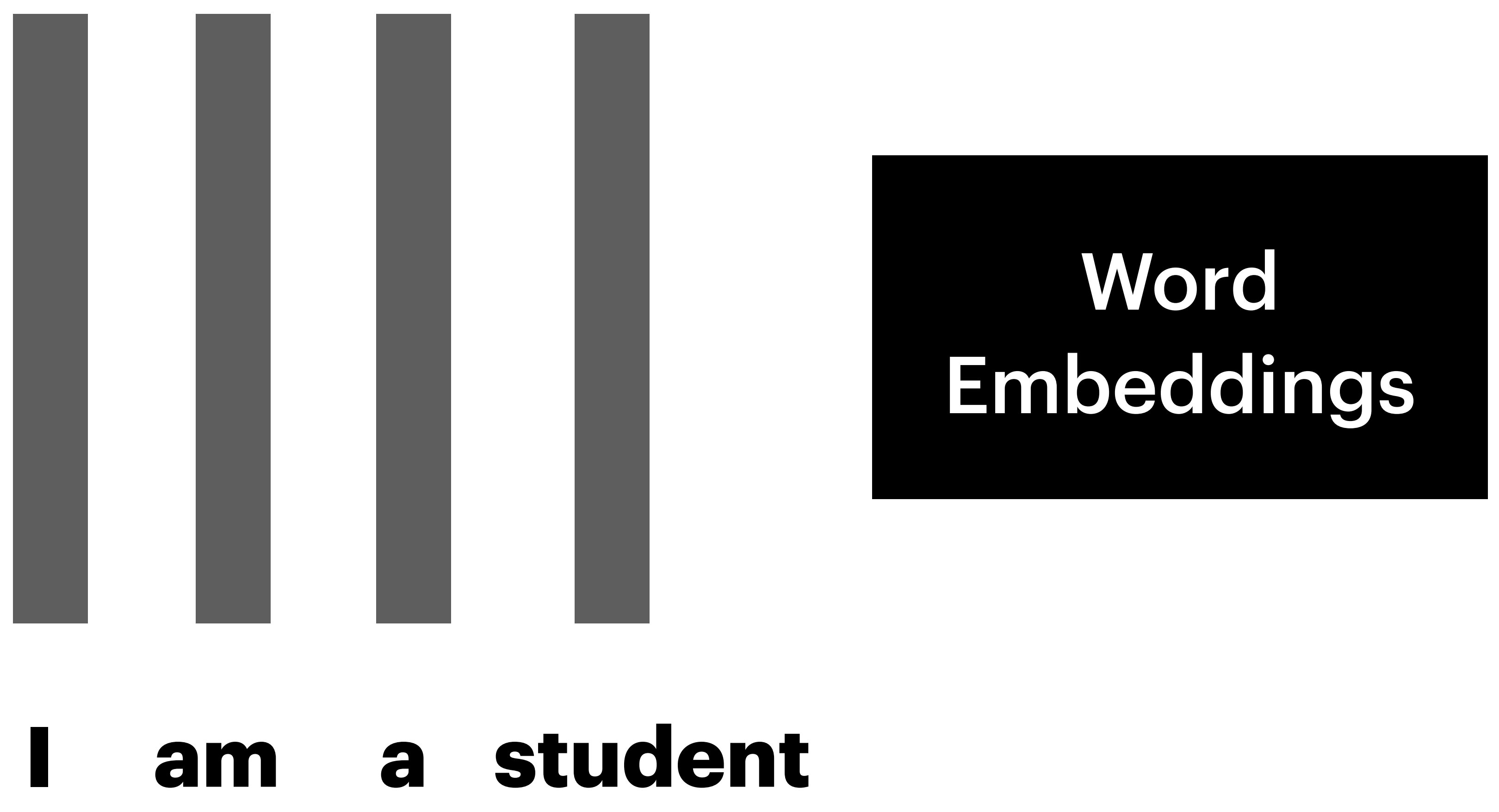
- Embedding context in sequence inputs



Transformers - self-attention

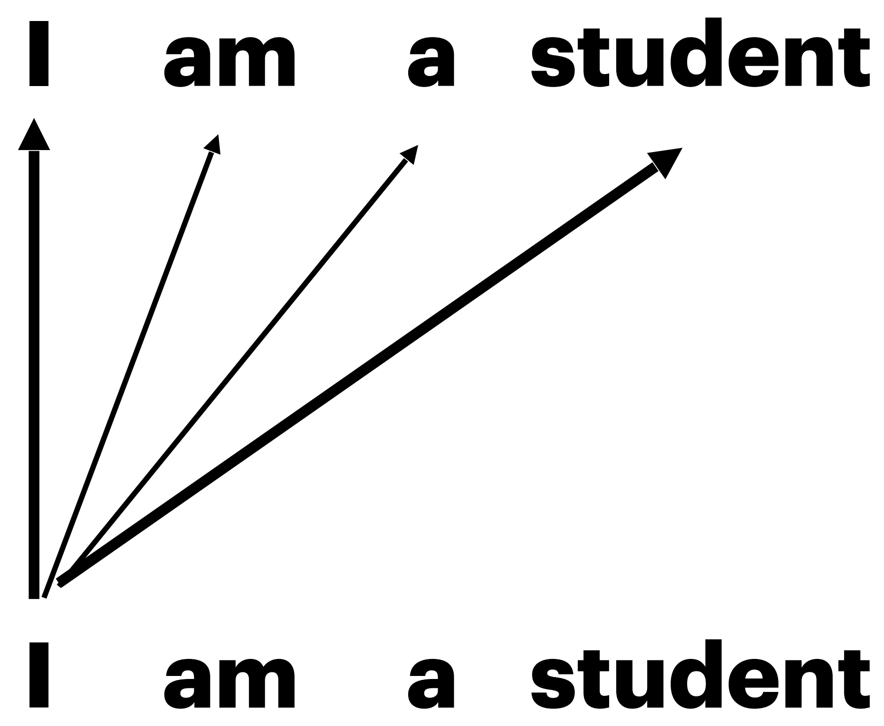
- Embedding context in sequence inputs
  - ✳ Let us take an example

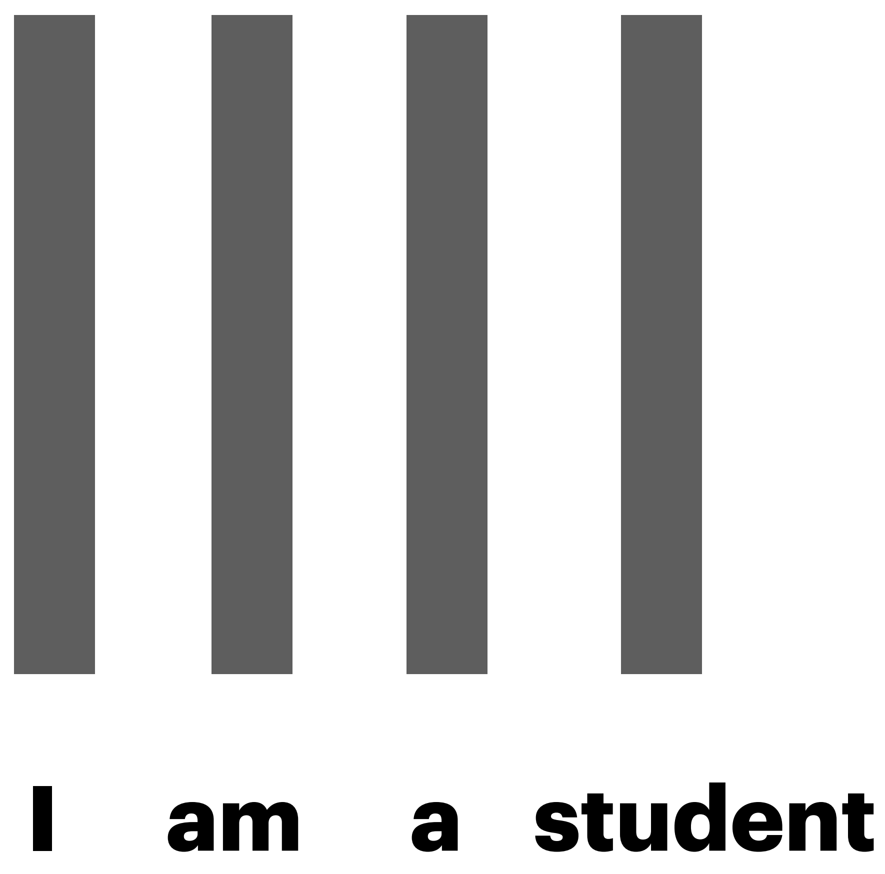




Transformers - self-attention

- Embedding context in sequence inputs
  - ✳ Let us take an example
  - ✳ Using word embeddings as the input representation





$$\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_T\}; \mathbf{x}_t \in \mathscr{R}^D$$

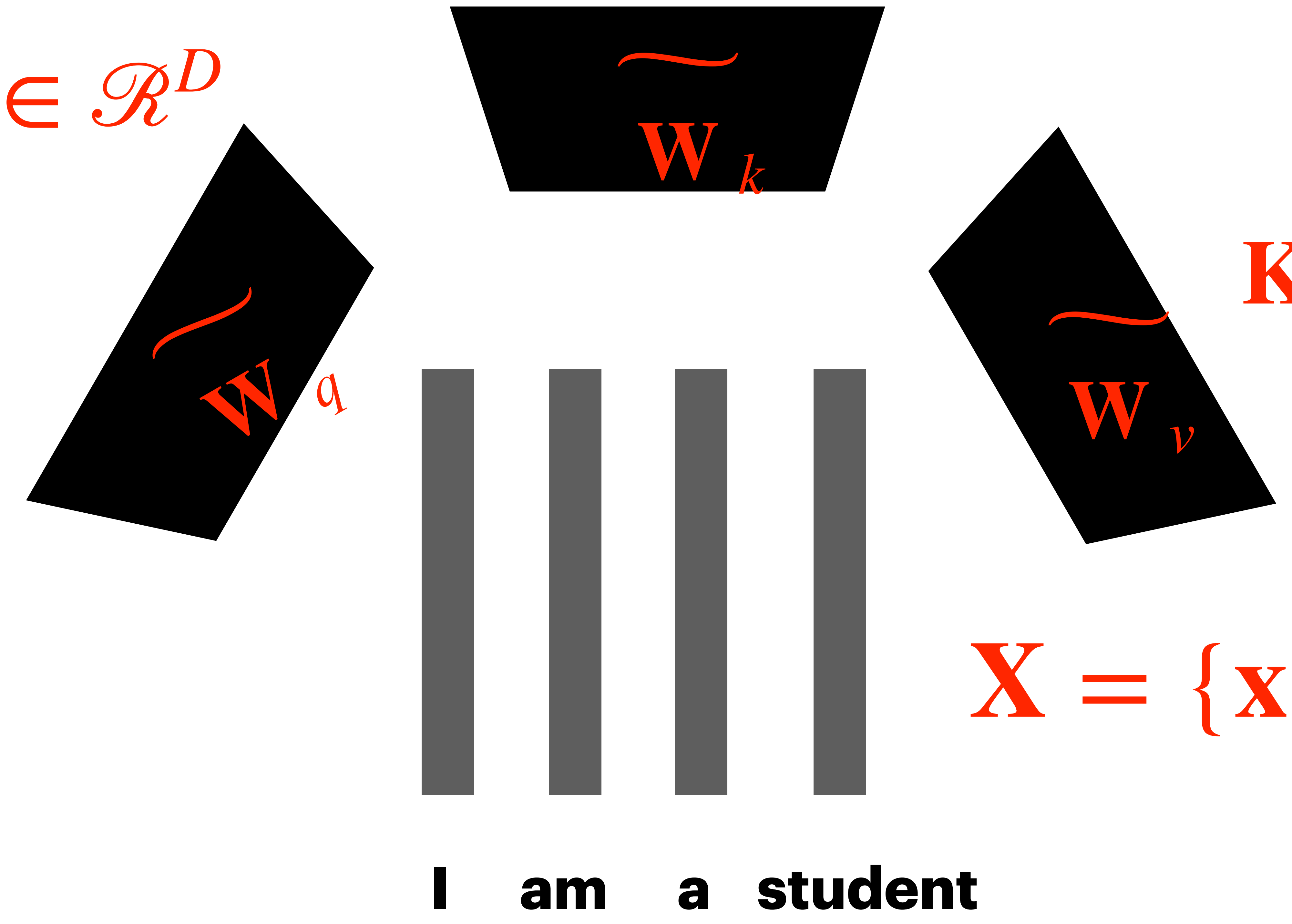
Word Embeddings

Transformers - self-attention

$$\mathbf{K} = \{\mathbf{k}_1, \mathbf{k}_2, \ldots \mathbf{k}_T\}; \mathbf{k}_t \in \mathscr{R}^D$$

$$\mathbf{Q} = \{\mathbf{q}_1, \mathbf{q}_2, \ldots \mathbf{x}_T\}; \mathbf{q}_t \in \mathscr{R}^D$$

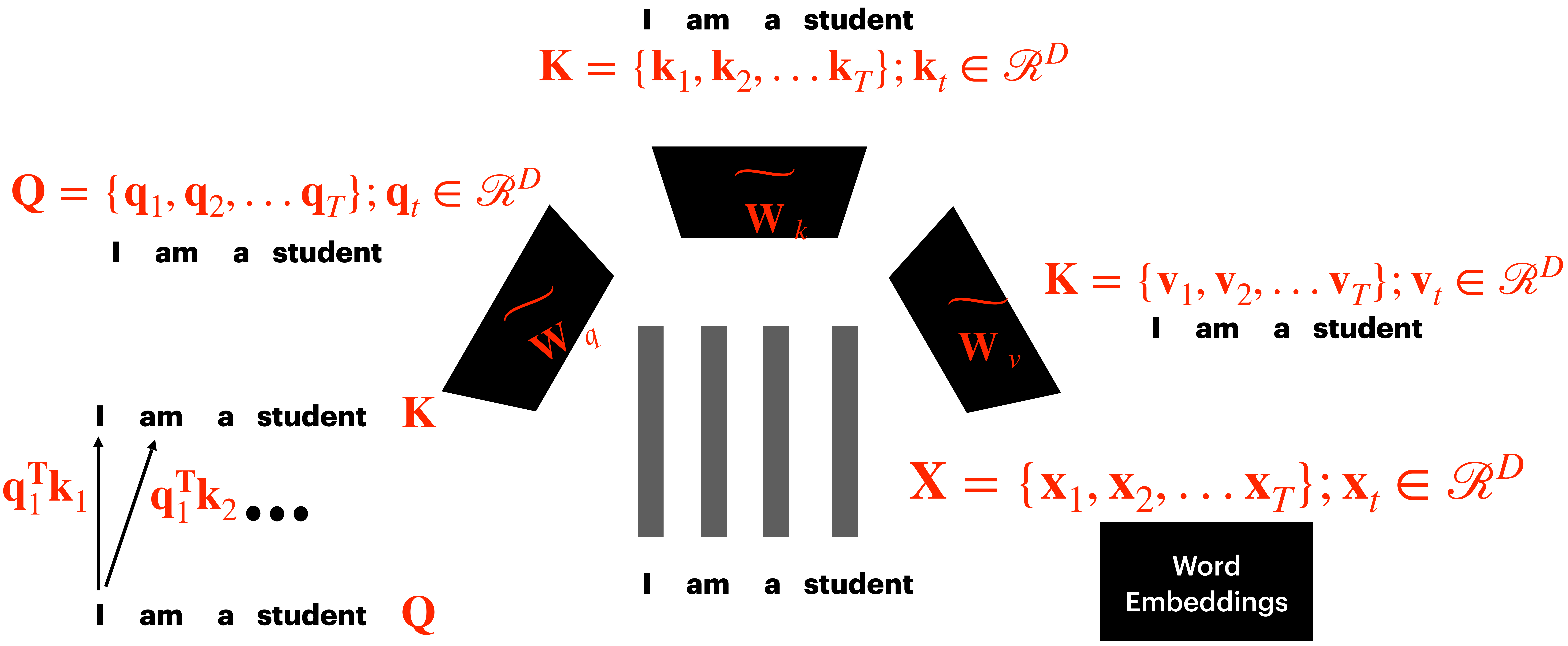
$$\mathbf{K} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots \mathbf{v}_T\}; \mathbf{v}_t \in \mathscr{R}^D$$



$$\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_T\}; \mathbf{x}_t \in \mathscr{R}^D$$

Word Embeddings

Transformers - self-attention



Transformers - self-attention

I am a student

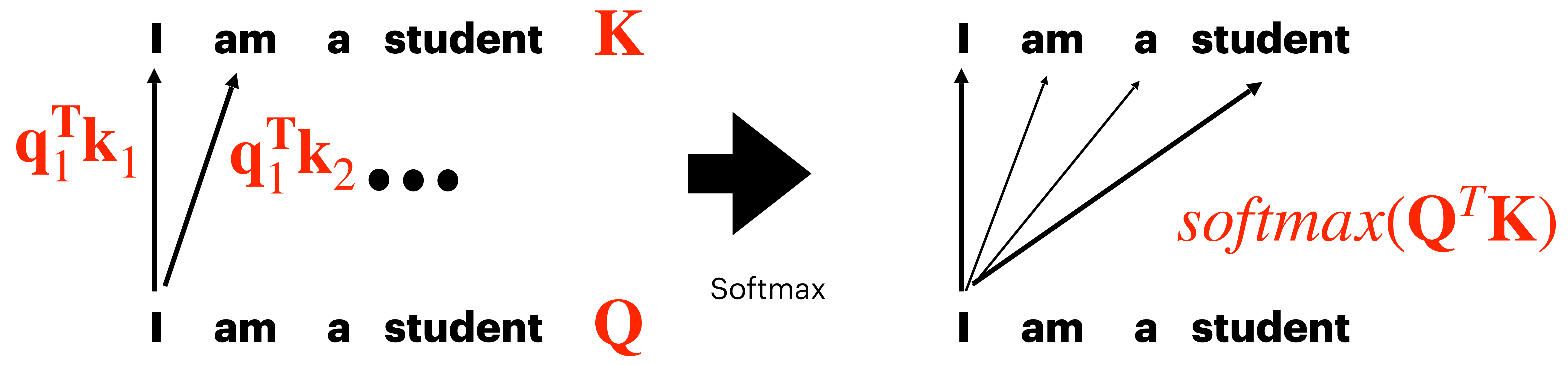
$$\mathbf{K} = \{\mathbf{k}_1, \mathbf{k}_2, \ldots \mathbf{k}_T\}; \mathbf{k}_t \in \mathscr{R}^D$$

I am a student

$$\mathbf{Q} = \{\mathbf{q}_1, \mathbf{q}_2, \ldots \mathbf{q}_T\}; \mathbf{q}_t \in \mathscr{R}^D$$

$$\mathbf{K} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots \mathbf{v}_T\}; \mathbf{v}_t \in \mathscr{R}^D$$

I am a student



Pics taken from : https://jalammar.github.io/illustrated-transformer/

Transformers - self-attention

I am a student

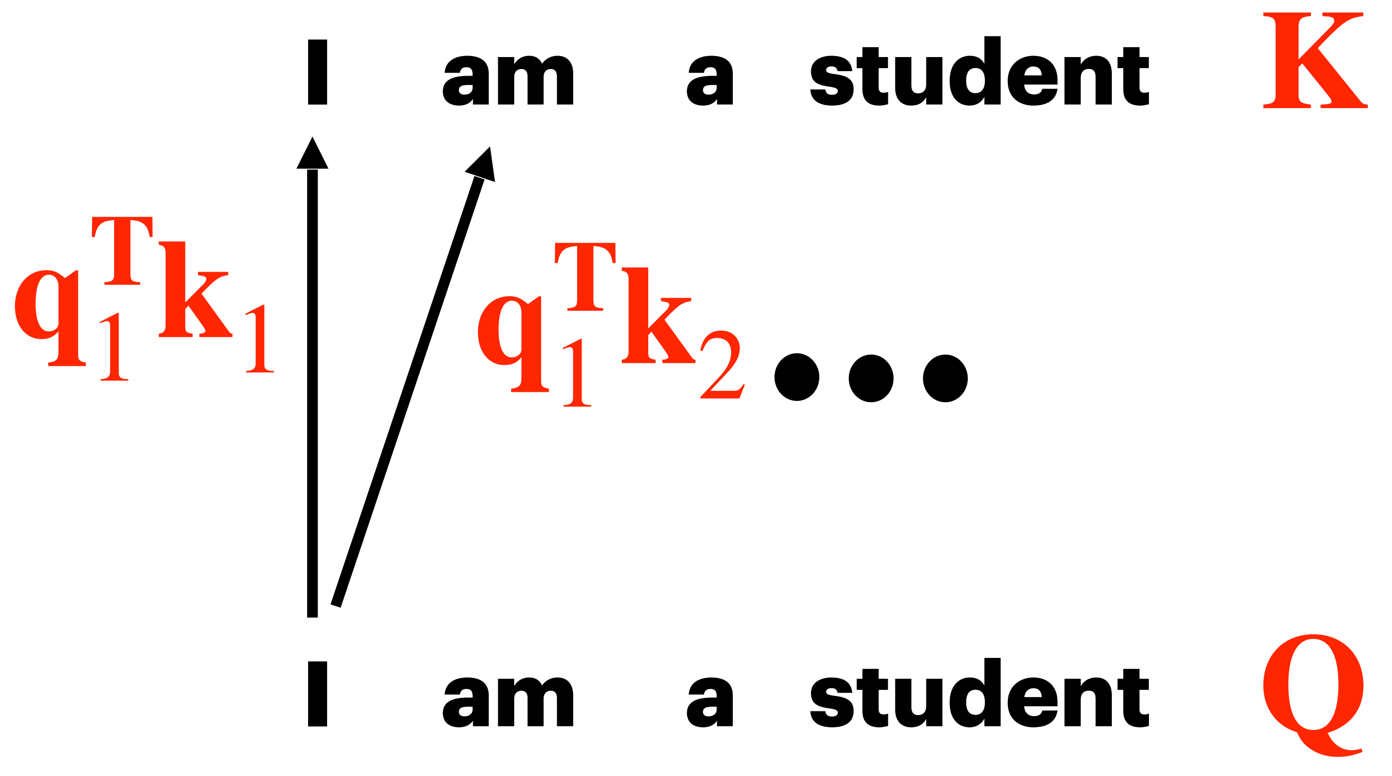
$$\mathbf{K} = \{\mathbf{k}_1, \mathbf{k}_2, \ldots \mathbf{k}_T\}; \mathbf{k}_t \in \mathscr{R}^D$$

I am a student

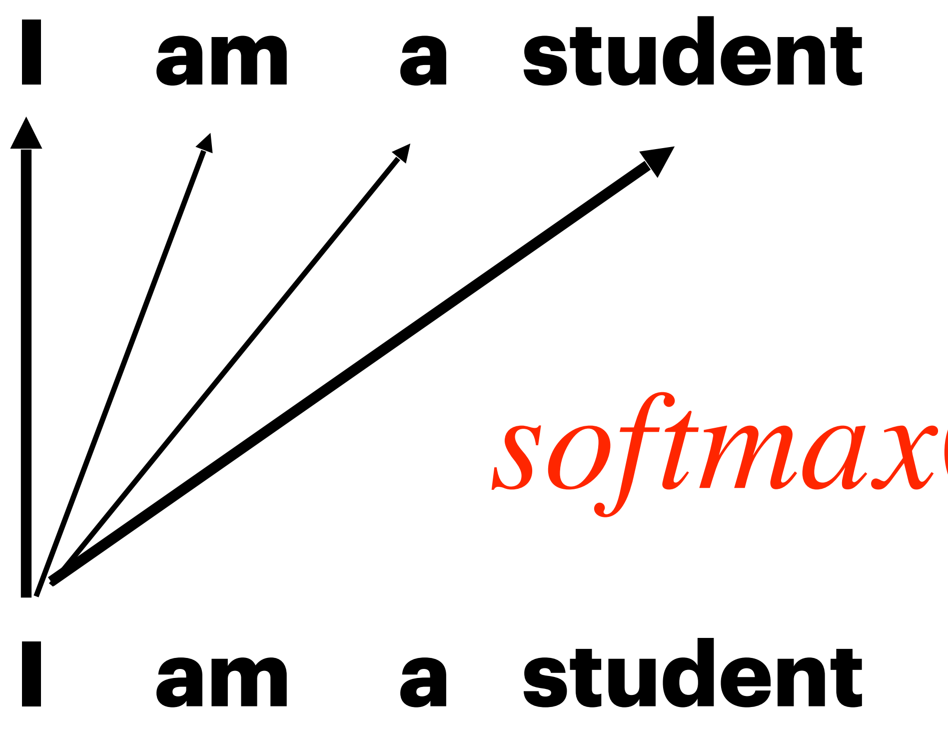
$$\mathbf{Q} = \{\mathbf{q}_1, \mathbf{q}_2, \ldots \mathbf{q}_T\}; \mathbf{q}_t \in \mathscr{R}^D$$

$$\mathbf{K} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots \mathbf{v}_T\}; \mathbf{v}_t \in \mathscr{R}^D$$

I am a student



Softmax



$softmax(\mathbf{Q}^T\mathbf{K})$

Weigh
Each word
With mask

$$\mathbf{V} \times softmax(\mathbf{Q}^T\mathbf{K})$$

Pics taken from : https://jalammar.github.io/illustrated-transformer/

THANK YOU

Sriram Ganapathy and TA team
LEAP lab, C328, EE, IISc
sriramg@iisc.ac.in