Master in Computer Vision Barcelona

Module 6
Lecture: Recurrent Neural Networks

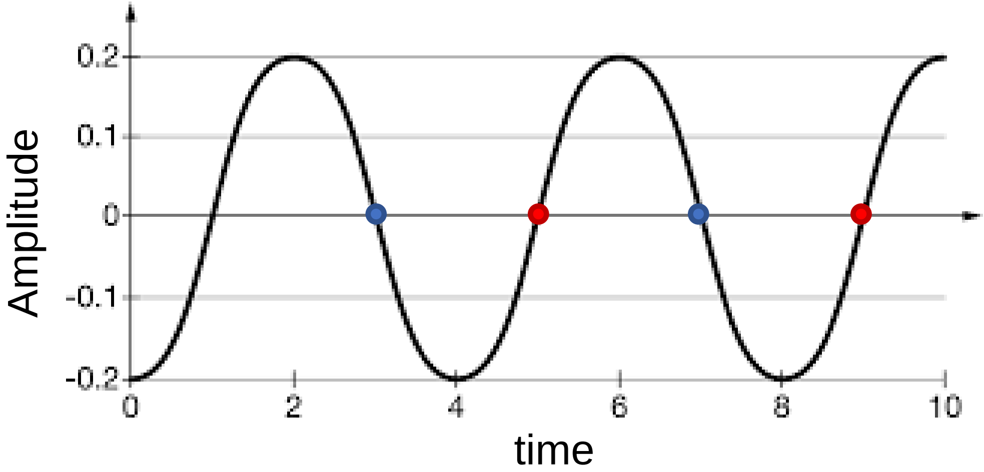
Federico Sukno

Sequences and Context

- RNNs are specialized networks designed to handle sequential data
- Sequences involve context
  - For example
    - Tell the 5th digit of your phone number
    - Sing your favorite song beginning at third sentence
    - Recall 10th character of the alphabet
- Two important aspects when dealing with sequences:
  - Memory of the past (history)
  - System's behavior depends on that history

Sequences and Context

- Two important aspects when dealing with sequences:
  - Memory of the past (history)
  - System's behavior depends on that history
- Consider the following examples
  - I would like to paint the walls in white color.
  - Ladies and gentlemen, let us welcome our next speaker: Mr. White.



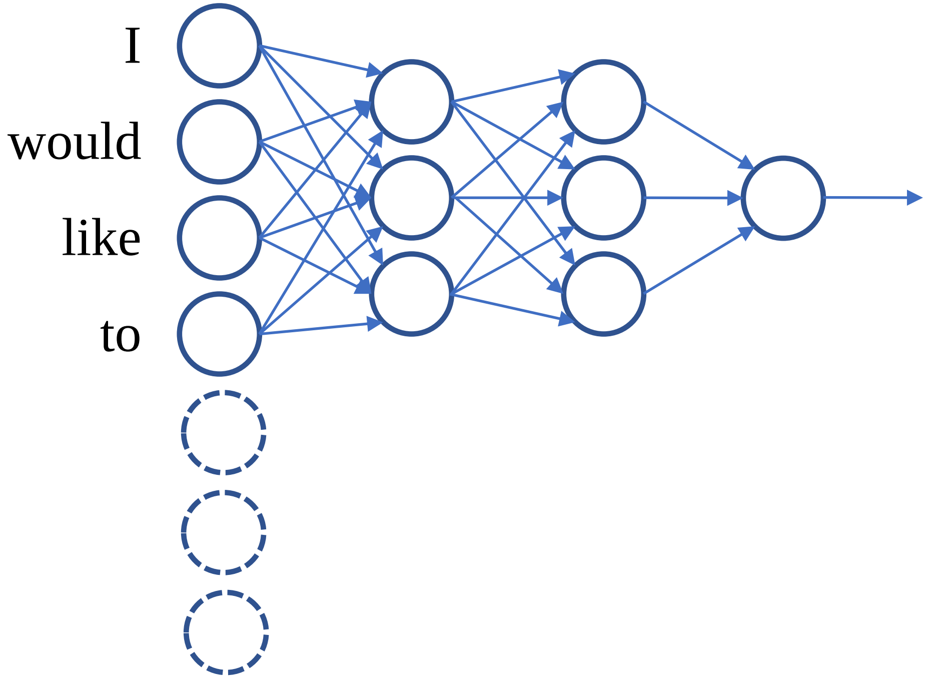
Recurrent Neural Networks

- Are specialized networks designed to handle sequential data
  - The output and the state of the network at time **t** can depend on both:
    - The input at time **t**
    - The “history” up to time **t** - 1
- Applications include
  - Any kind of audio / video processing and analysis
  - Text analysis, classification, and synthesis
  - Machine translation
  - DNA analysis

Do we really need RNNs ?

- How about using fully-connected layers?
  - Not computationally efficient
  - No parameter sharing

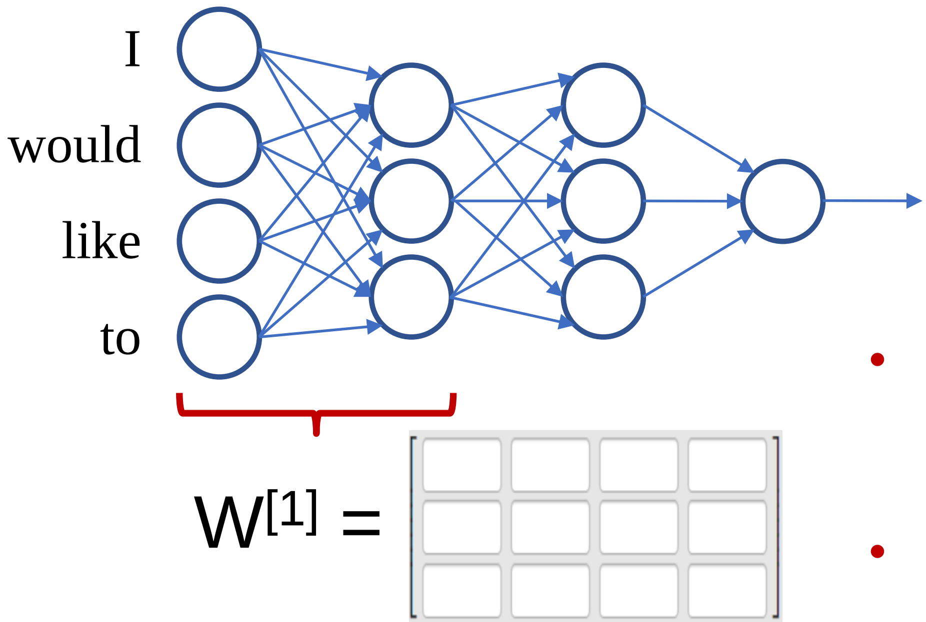
"I would like to paint the walls in white color."



Do we really need RNNs ?

- How about using fully-connected layers?
  - Not computationally efficient
  - No parameter sharing

"I would like to."

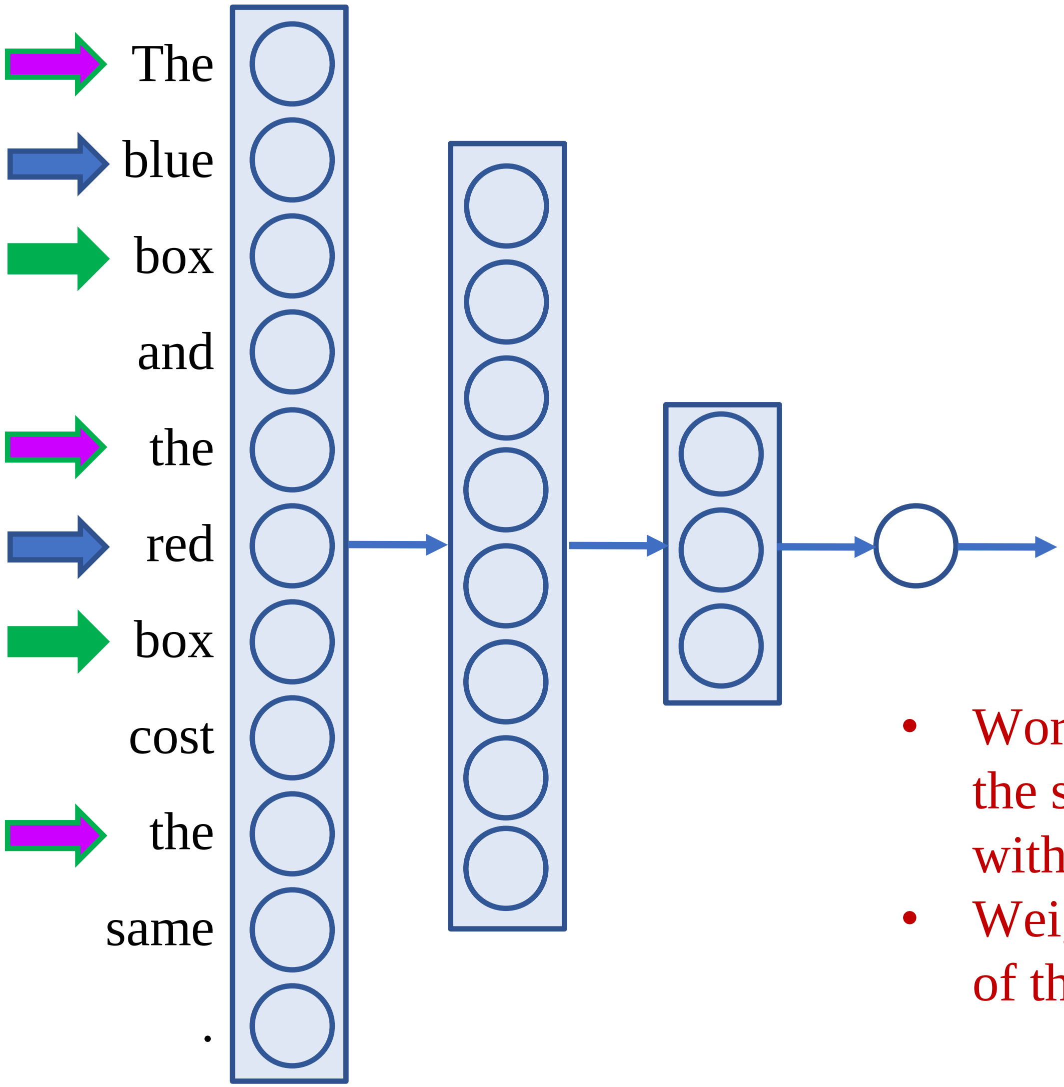


- It turns our that the input size would be:

  (#words) × (embedding-size)
- We shall consider the maximum sentence length

Do we really need RNNs ?

- No parameter sharing



- Words with a similar function in the sentence go through the NN with different weights
- Weights depend on the position of the word in the sentence.

RNNs: main idea

The

blue

box

and

the

red

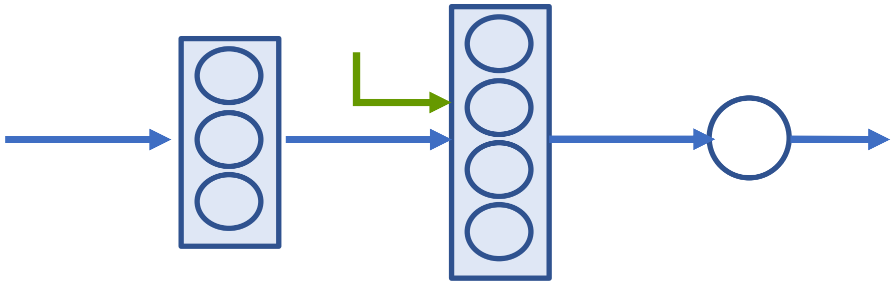
box

cost

the

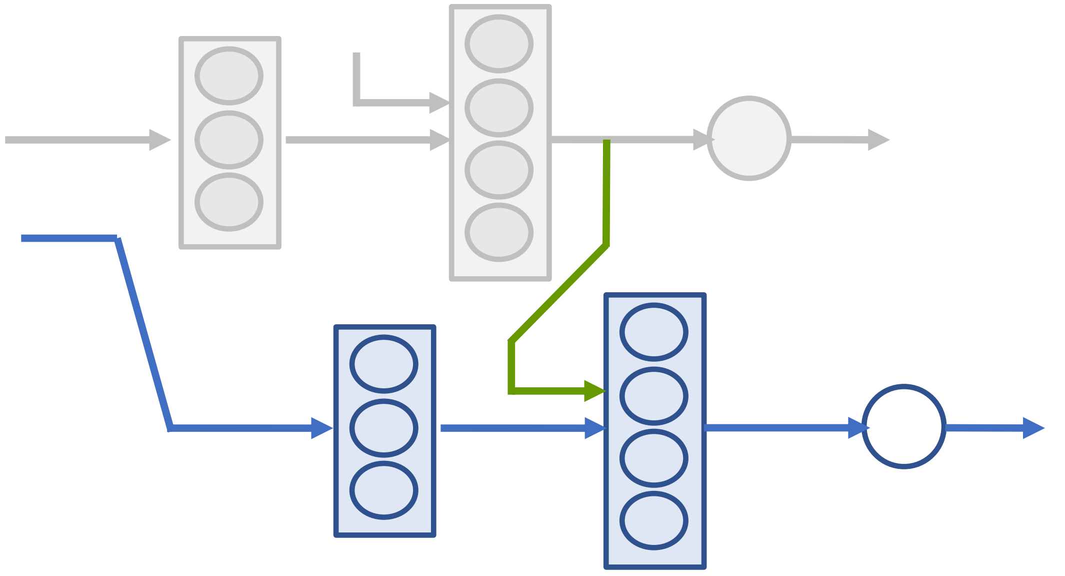
same

.



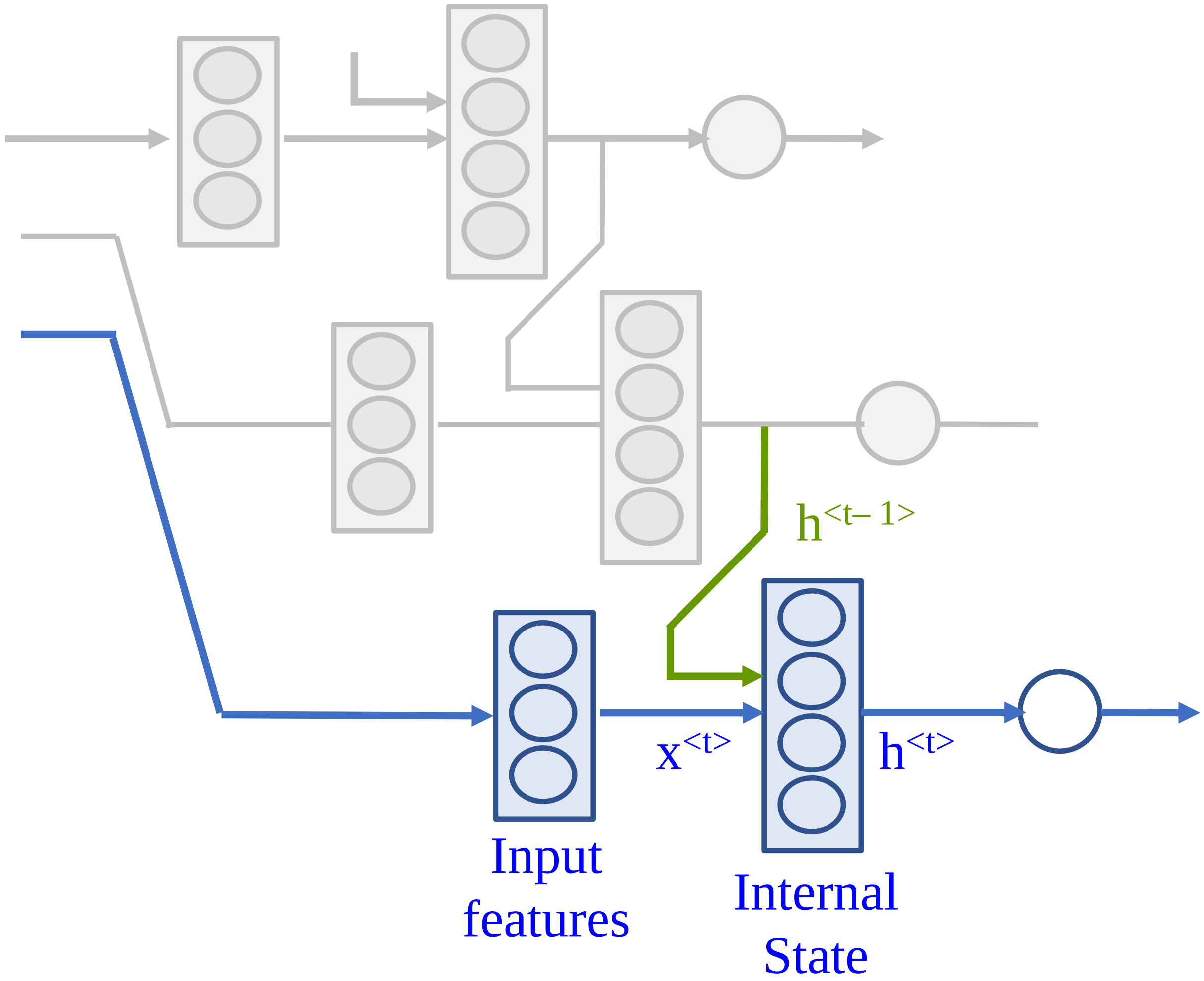
RNNs: main idea

The
blue
box
and
the
red
box
cost
the
same
.



RNNs: main idea

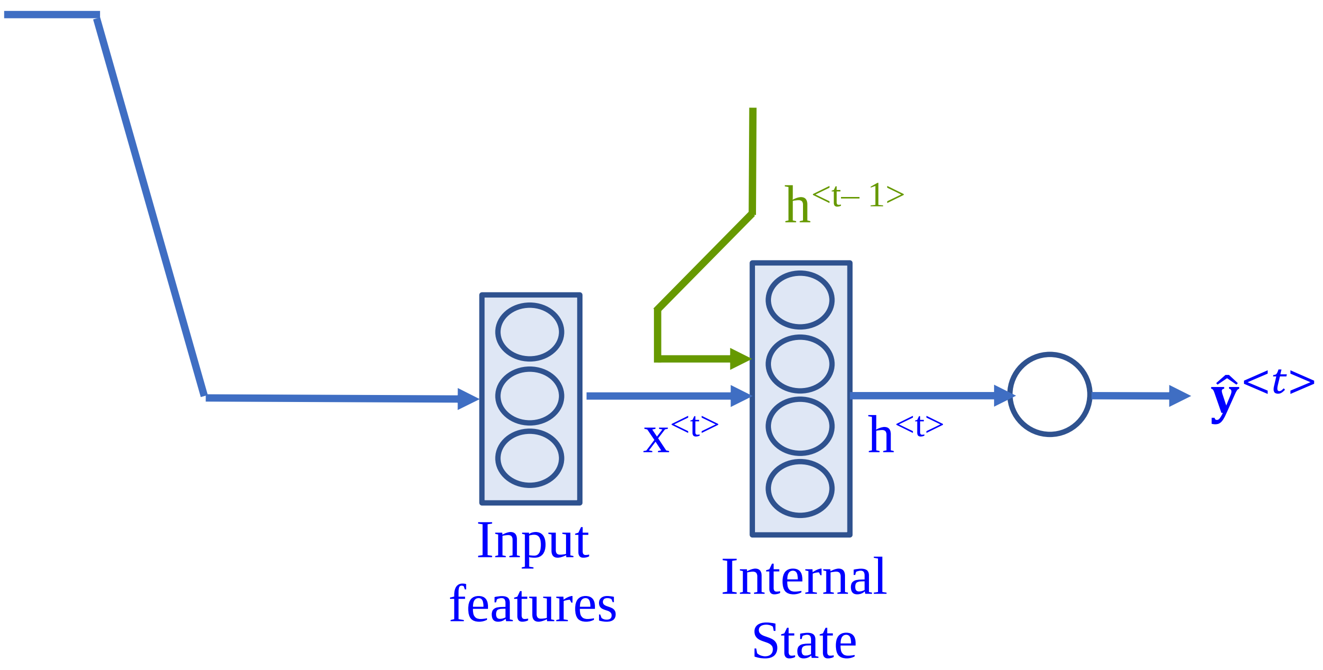
The
blue
box
and
the
red
box
cost
the
same
.



RNNs: main equations

$$\mathbf{h}^{<t>} = g_h\left(\mathbf{U}\mathbf{x}^{<t>} + \boldsymbol{W}\mathbf{h}^{<t-1>} + \mathbf{b}_h\right)$$

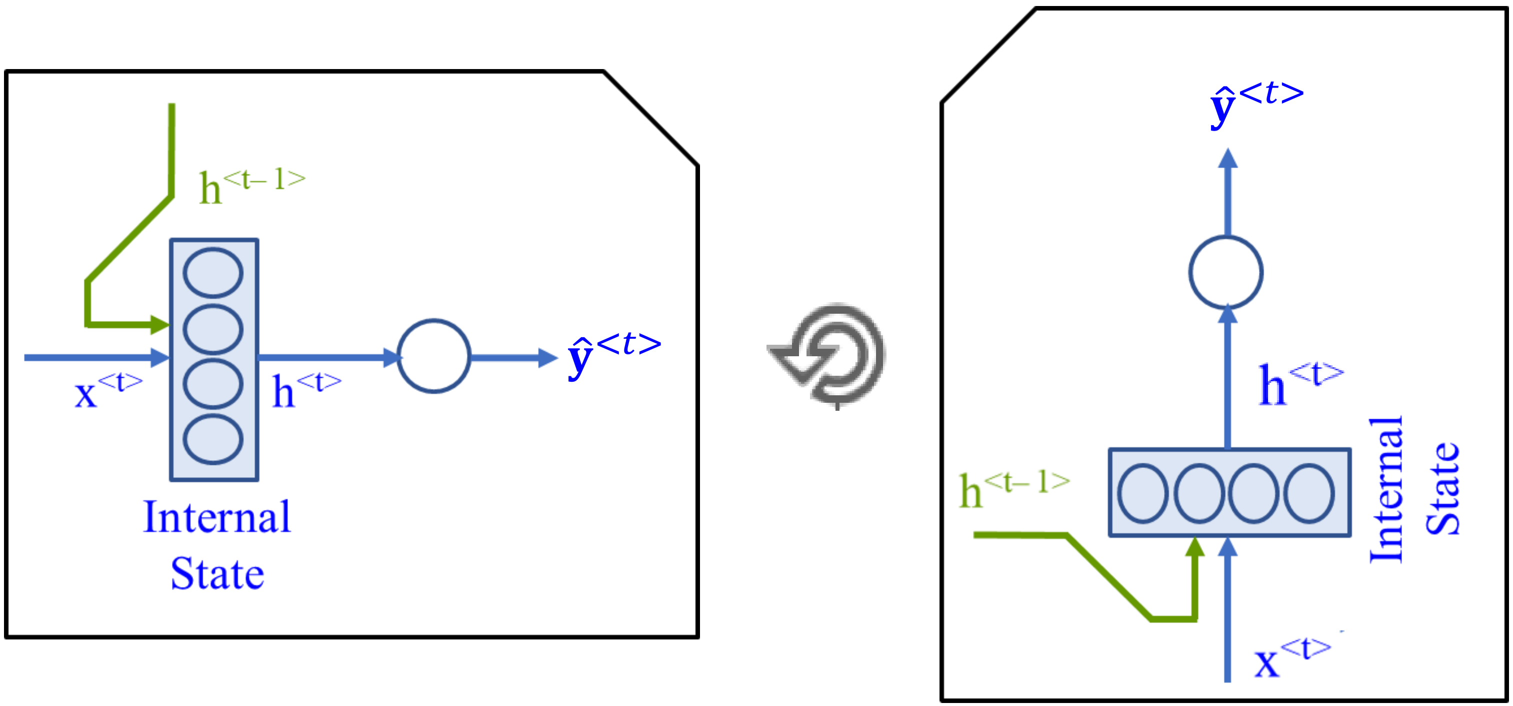
$$\hat{\mathbf{y}}^{<t>} = g_y\left(\boldsymbol{V}\mathbf{h}^{<t>} + \mathbf{b}_y\right)$$



RNN Diagrams

$$\mathbf{h}^{<t>} = g_h\left(\mathbf{U}\mathbf{x}^{<t>} + \boldsymbol{W}\mathbf{h}^{<t-1>} + \mathbf{b}_h\right)$$

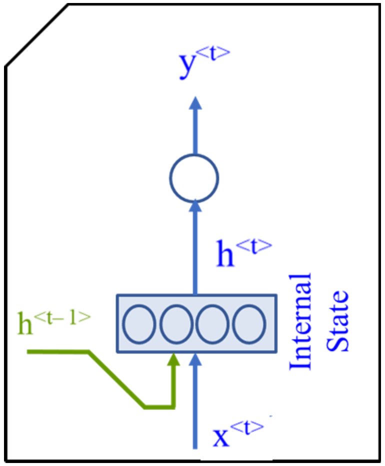
$$\hat{\mathbf{y}}^{<t>} = g_y\left(\boldsymbol{V}\mathbf{h}^{<t>} + \mathbf{b}_y\right)$$

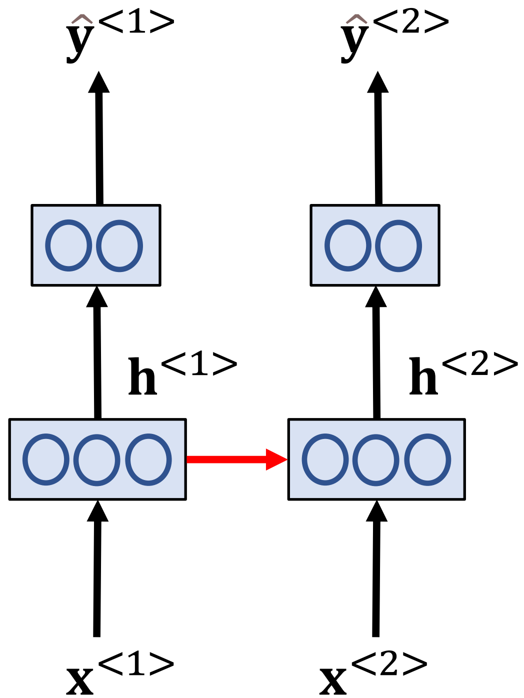


RNN Diagrams

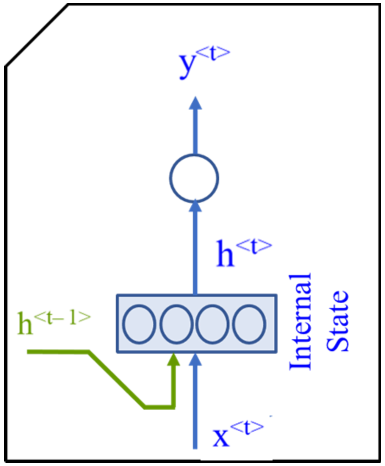
$$\mathbf{h}^{<t>} = g_h\left(\mathbf{U}\mathbf{x}^{<t>} + \boldsymbol{W}\mathbf{h}^{<t-1>} + \mathbf{b}_h\right)$$

$$\hat{\mathbf{y}}^{<t>} = g_y\left(\boldsymbol{V}\mathbf{h}^{<t>} + \mathbf{b}_y\right)$$



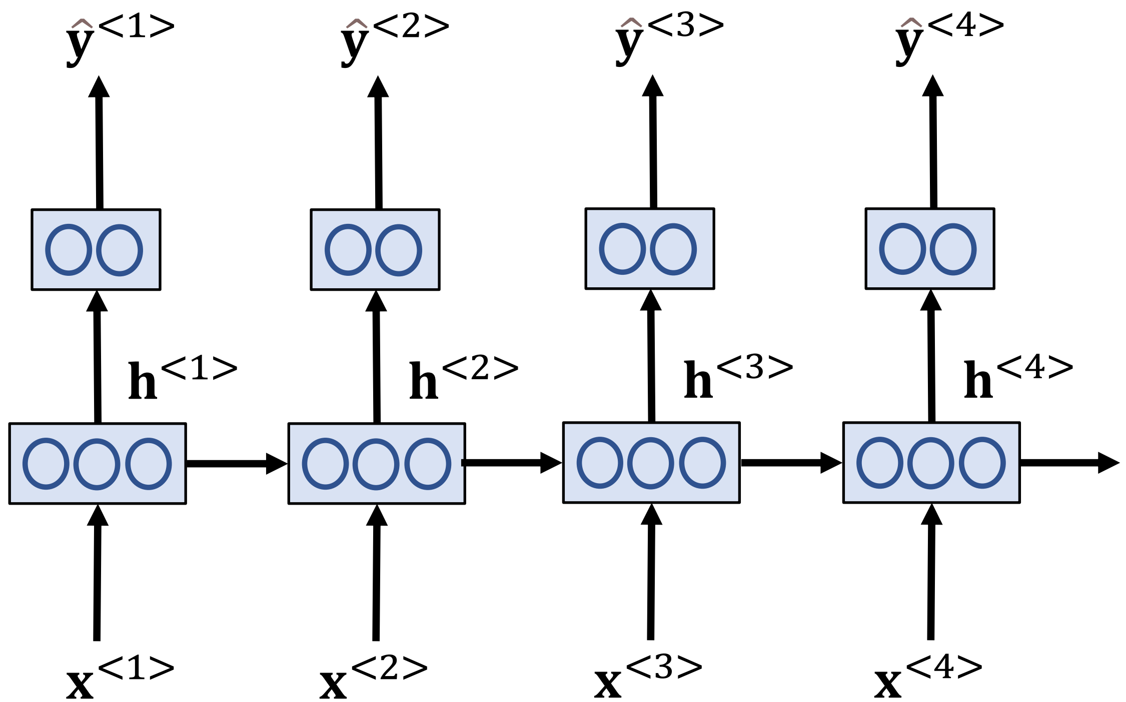


RNN Diagrams

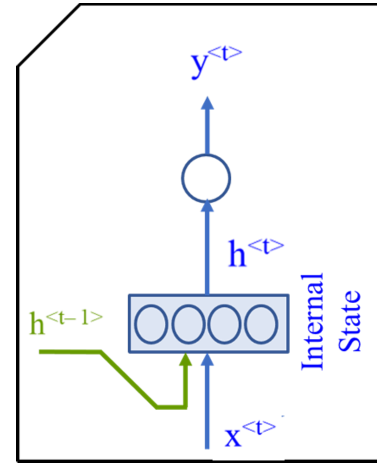


$$\mathbf{h}^{<t>} = g_h\left(\mathbf{U}\mathbf{x}^{<t>} + \boldsymbol{W}\mathbf{h}^{<t-1>} + \mathbf{b}_h\right)$$

$$\hat{\mathbf{y}}^{<t>} = g_y\left(\boldsymbol{V}\mathbf{h}^{<t>} + \mathbf{b}_y\right)$$

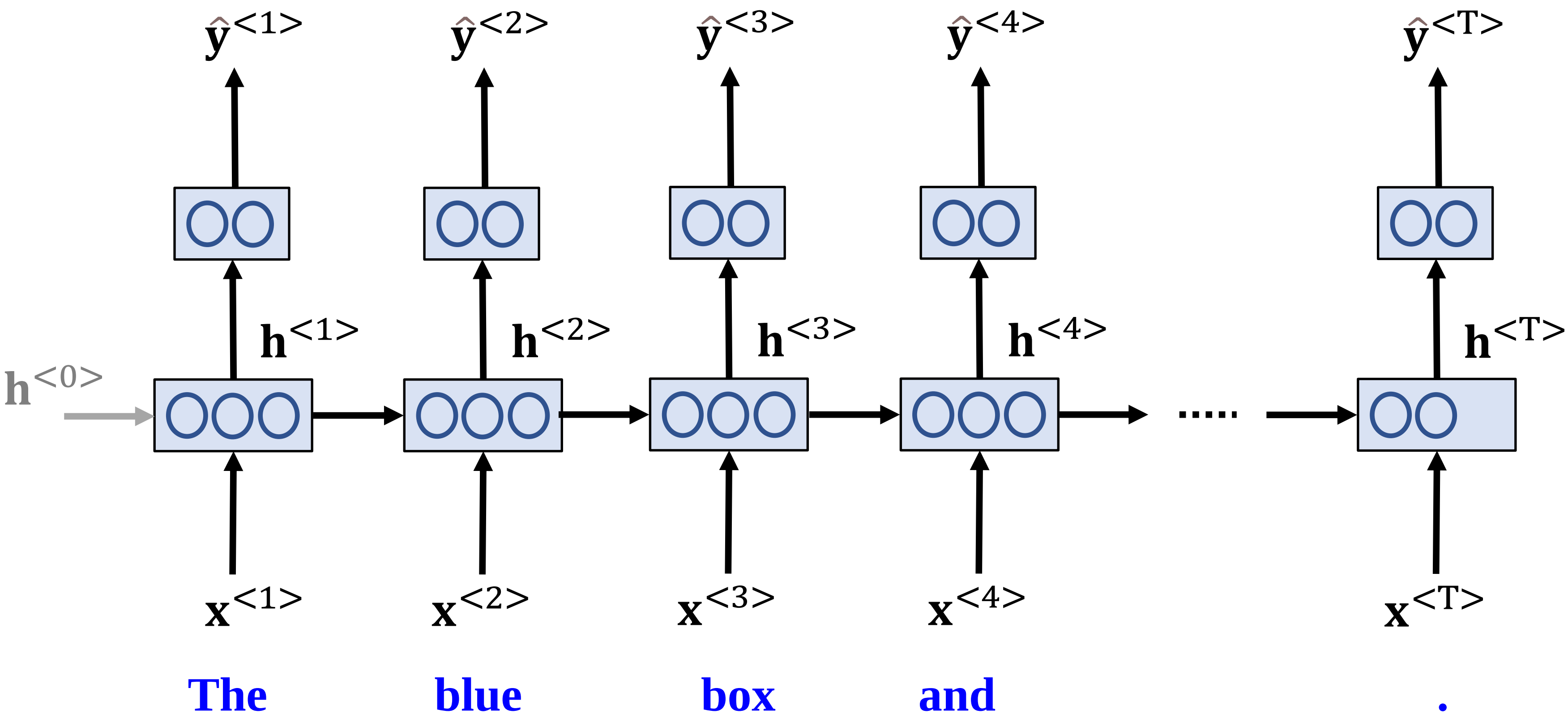


RNN Diagrams



$$\mathbf{h}^{<t>} = g_h\left(\mathbf{U}\mathbf{x}^{<t>} + \boldsymbol{W}\mathbf{h}^{<t-1>} + \mathbf{b}_h\right)$$

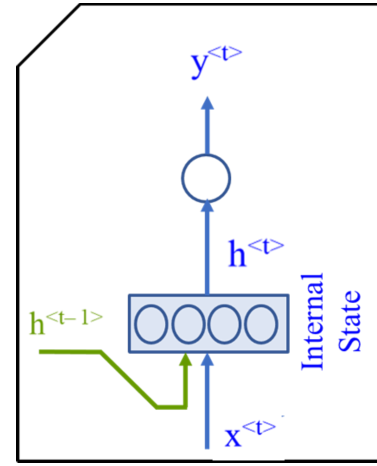
$$\hat{\mathbf{y}}^{<t>} = g_y\left(\boldsymbol{V}\mathbf{h}^{<t>} + \mathbf{b}_y\right)$$

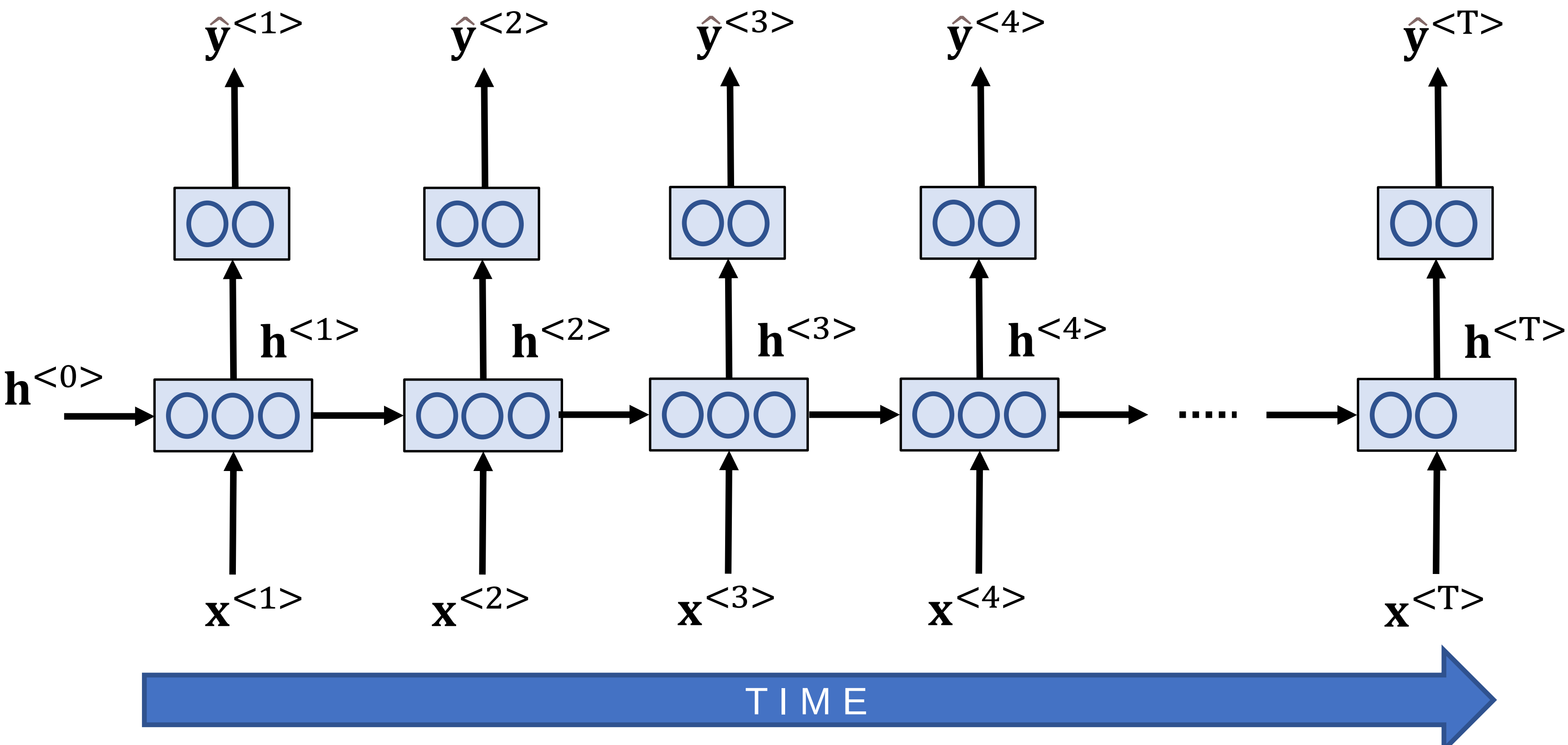


RNN Diagrams: Unfolded

$$\mathbf{h}^{<t>} = g_h\left(\mathbf{U}\mathbf{x}^{<t>} + \boldsymbol{W}\mathbf{h}^{<t-1>} + \mathbf{b}_h\right)$$

$$\hat{\mathbf{y}}^{<t>} = g_y\left(\boldsymbol{V}\mathbf{h}^{<t>} + \mathbf{b}_y\right)$$

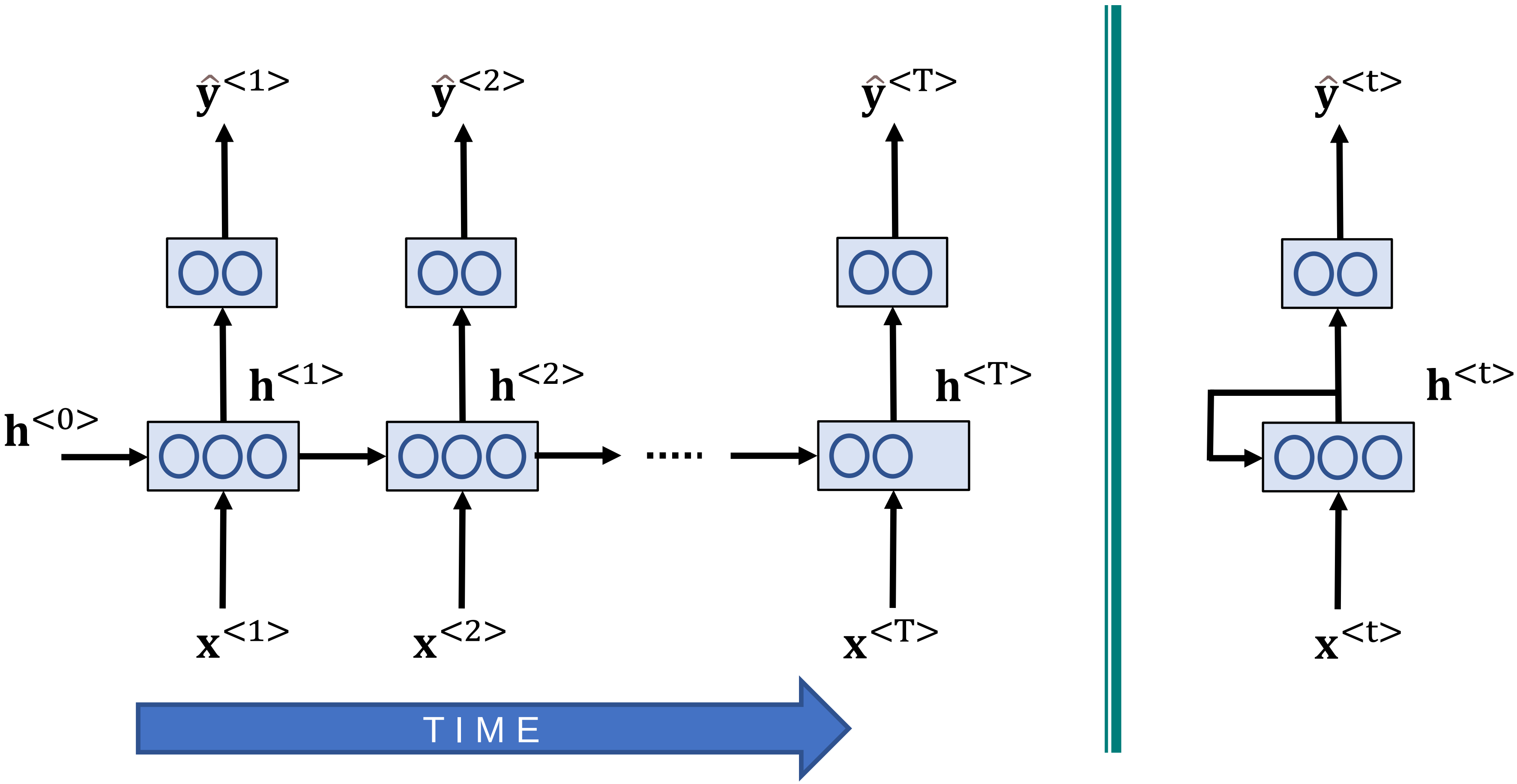




RNN Diagrams: Unfolded vs Compact

$$\mathbf{h}^{<t>} = g_h\left(\mathbf{U}\mathbf{x}^{<t>} + W\mathbf{h}^{<t-1>} + \mathbf{b}_h\right)$$

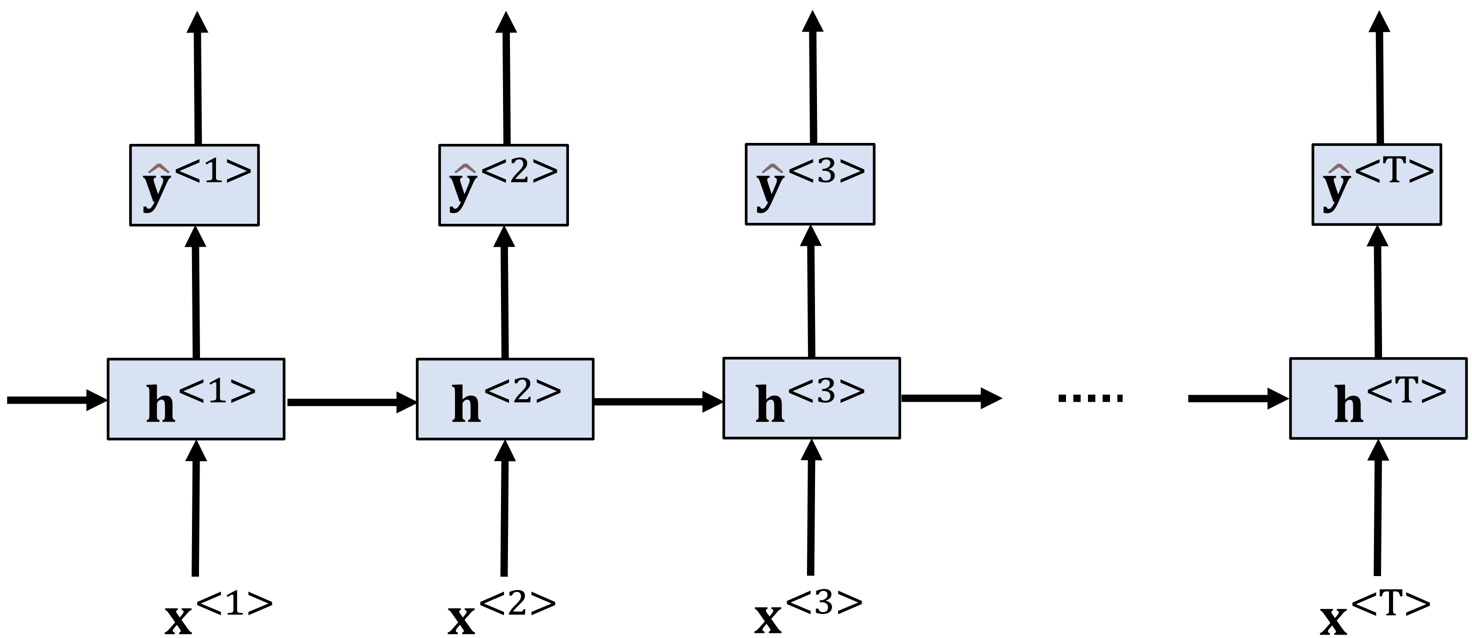
$$\hat{\mathbf{y}}^{<t>} = g_y\left(V\mathbf{h}^{<t>} + \mathbf{b}_y\right)$$



RNN Diagrams: Computational Graph

$$\mathbf{h}^{<t>} = g_h\left(\mathbf{U}\mathbf{x}^{<t>} + \boldsymbol{W}\mathbf{h}^{<t-1>} + \mathbf{b}_h\right)$$

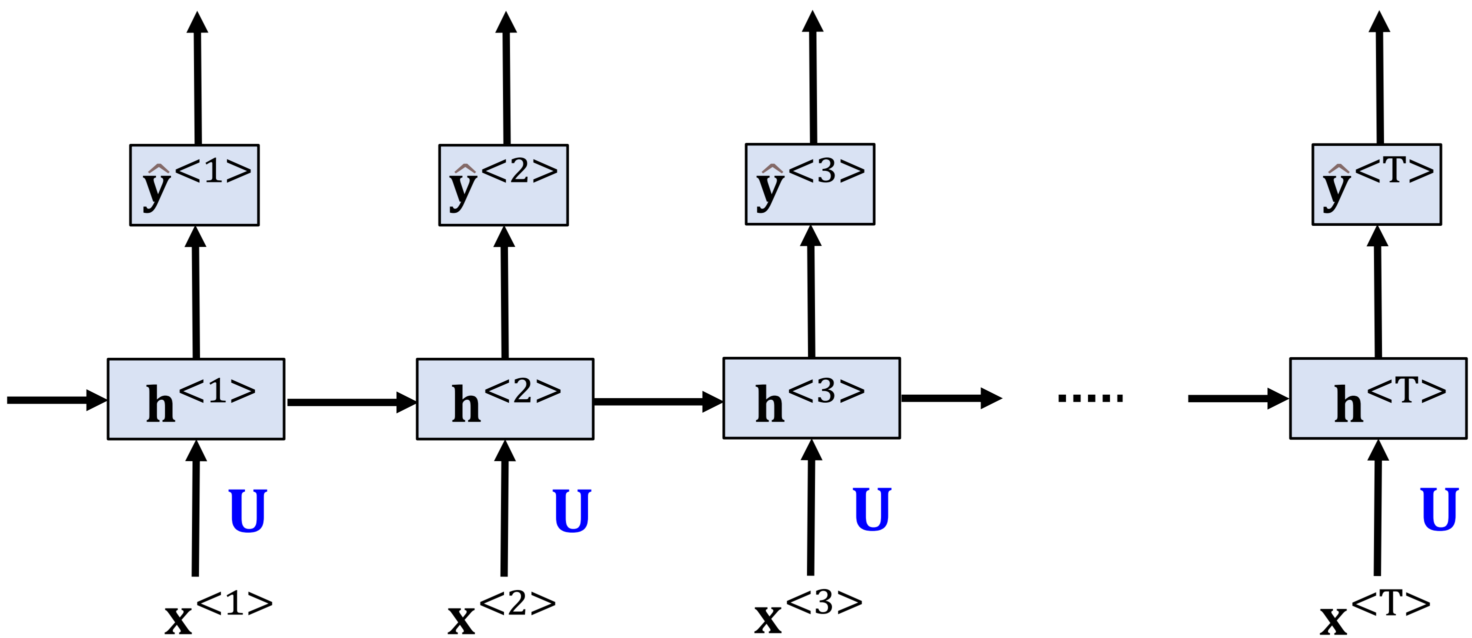
$$\hat{\mathbf{y}}^{<t>} = g_y\left(\boldsymbol{V}\mathbf{h}^{<t>} + \mathbf{b}_y\right)$$



RNN Diagrams: Computational Graph

$$\mathbf{h}^{<t>} = g_h\left(\mathbf{U}\mathbf{x}^{<t>} + \boldsymbol{W}\mathbf{h}^{<t-1>} + \mathbf{b}_h\right)$$

$$\hat{\mathbf{y}}^{<t>} = g_y\left(\boldsymbol{V}\mathbf{h}^{<t>} + \mathbf{b}_y\right)$$

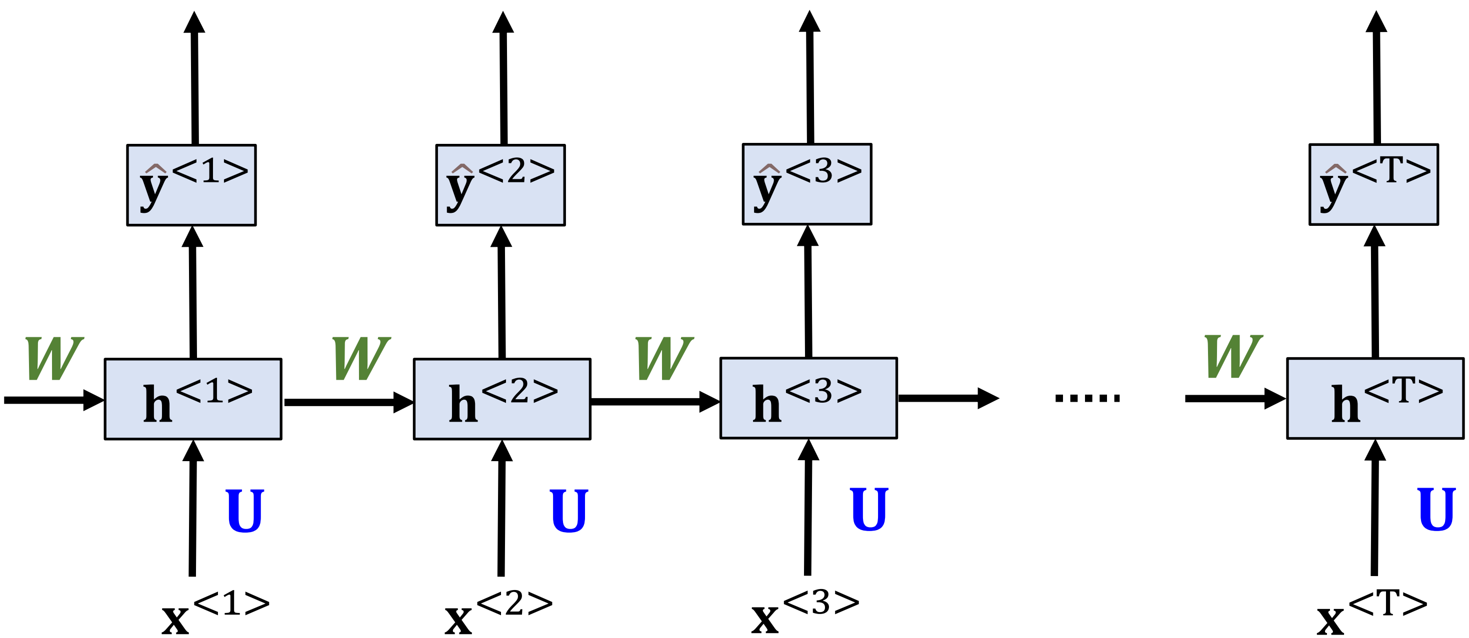


Parameter Sharing!

RNN Diagrams: Computational Graph

$$\mathbf{h}^{<t>} = g_h\left(\mathbf{U}\mathbf{x}^{<t>} + W\mathbf{h}^{<t-1>} + \mathbf{b}_h\right)$$

$$\hat{\mathbf{y}}^{<t>} = g_y\left(V\mathbf{h}^{<t>} + \mathbf{b}_y\right)$$

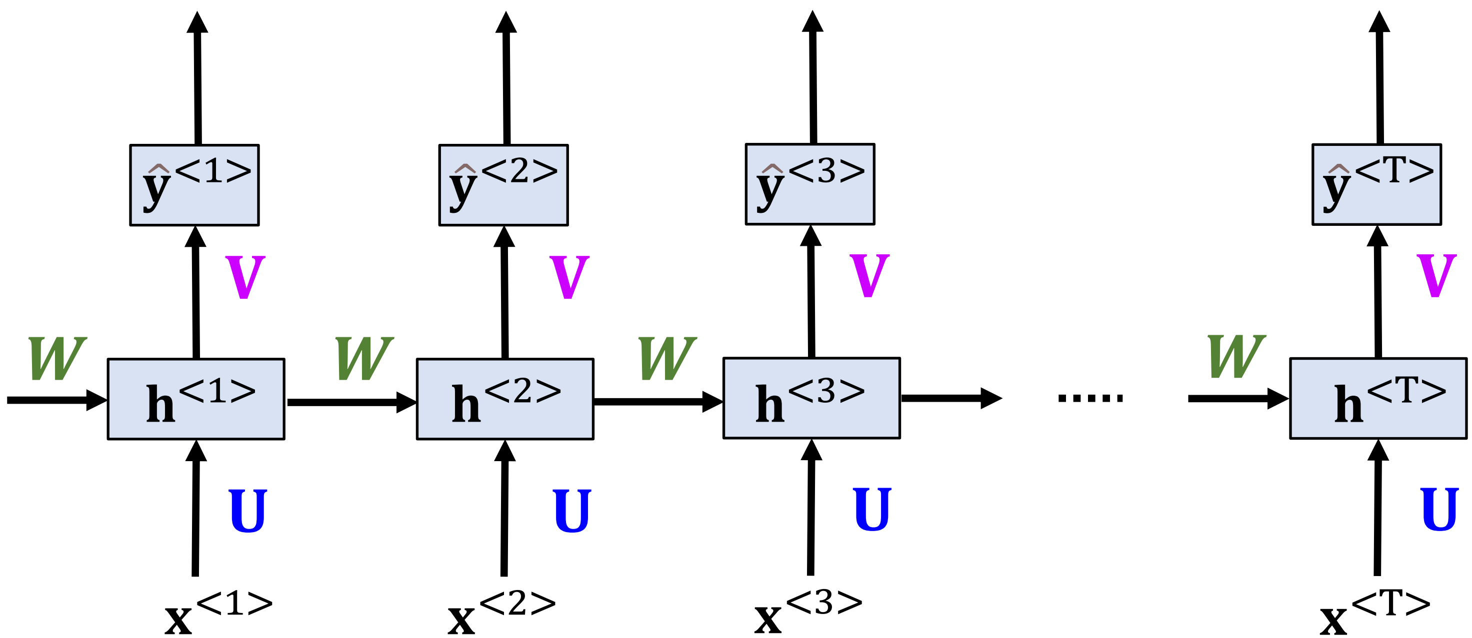


Parameter Sharing!

RNN Diagrams: Computational Graph

$$\mathbf{h}^{<t>} = g_h(\mathbf{U}\mathbf{x}^{<t>} + W\mathbf{h}^{<t-1>} + \mathbf{b}_h)$$

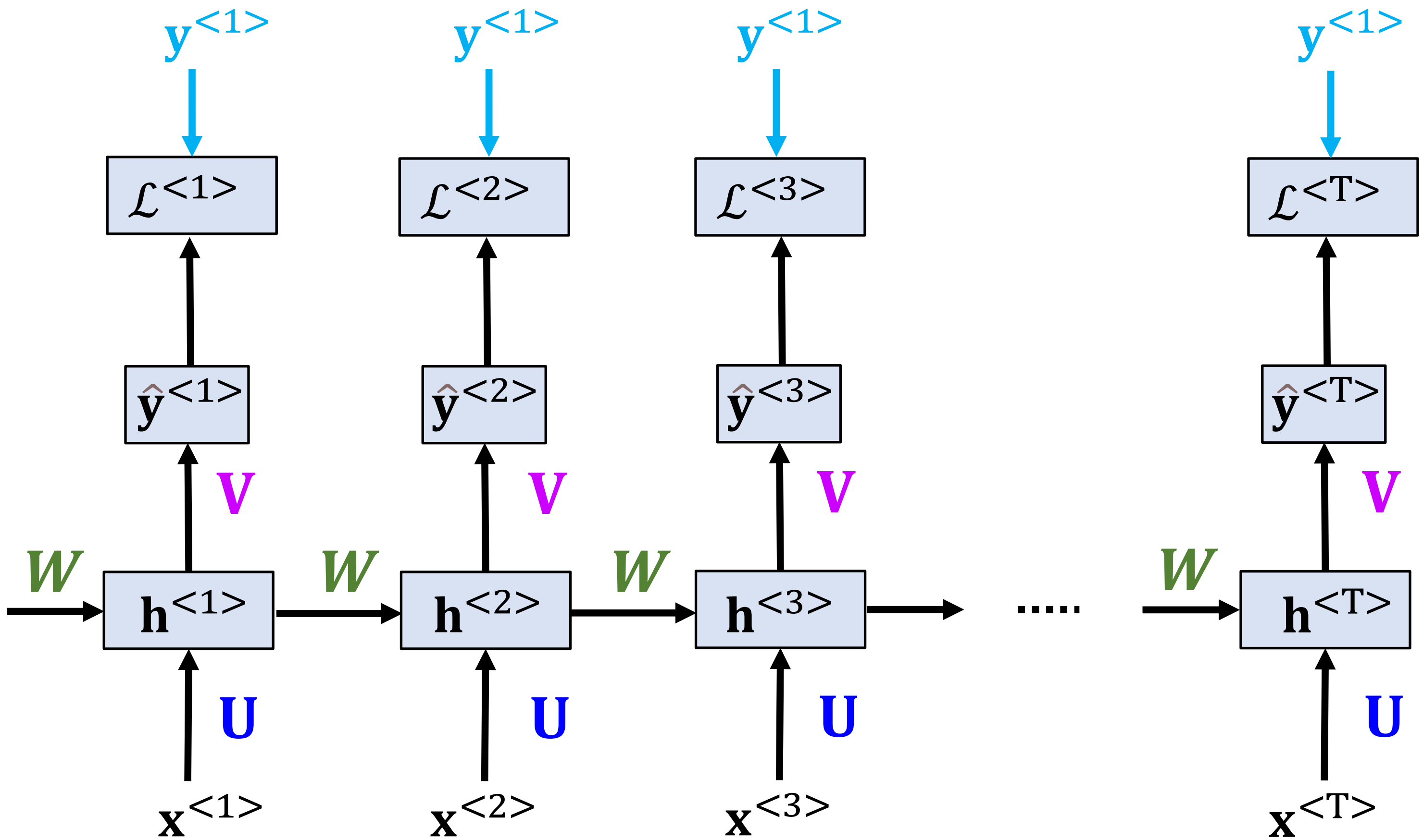
$$\hat{\mathbf{y}}^{<t>} = g_y(V\mathbf{h}^{<t>} + \mathbf{b}_y)$$



Parameter Sharing!

RNN Diagrams: Computational Graph

$$\mathcal{L}^{<t>} = \mathcal{L}(\hat{\mathbf{y}}^{<t>}, \boldsymbol{y}^{<t>})$$



RNN Loss

$$\mathcal{L}^{<t>} = \mathcal{L}(\hat{\mathbf{y}}^{<t>}, \boldsymbol{y}^{<t>})$$

- We can use any loss function (cross entropy, L2, etc(
- The losses at each time instant are combined to get a sequence loss:

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = \sum_{t=1}^{T_y} \mathcal{L}(\hat{\mathbf{y}}^{<t>} \boldsymbol{y}^{<t>})$$

Each training sequence has its own length

RNN Loss

$$\mathcal{L}^{<t>} = \mathcal{L}(\hat{\mathbf{y}}^{<t>}, \boldsymbol{y}^{<t>})$$

- We can use any loss function (cross entropy, L2, etc(
- The losses at each time instant are combined to get a sequence loss:

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = \sum_{t=1}^{T_y} \mathcal{L}(\hat{\mathbf{y}}^{<t>} \boldsymbol{y}^{<t>})$$

Each training sequence has its own length

  - Where T_y is the length of the sequence labels
  - Similarly, T_x is the length of the sequence inputs

$$\mathbf{x} = \mathbf{x}^{<1>}, \mathbf{x}^{<2>}, \mathbf{x}^{<3>}, \dots, \mathbf{x}^{<T_x>}$$

$$\hat{\mathbf{y}} = \hat{\mathbf{y}}^{<1>}, \hat{\mathbf{y}}^{<2>}, \hat{\mathbf{y}}^{<3>}, \dots, \hat{\mathbf{y}}^{<T_y>}$$

RNN Loss

$$\mathcal{L}^{<t>} = \mathcal{L}(\hat{\mathbf{y}}^{<t>}, \boldsymbol{y}^{<t>})$$

- We can use any loss function (cross entropy, L2, etc(
- The losses at each time instant are combined to get a sequence loss:

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = \sum_{t=1}^{T_y} \mathcal{L}(\hat{\mathbf{y}}^{<t>} \boldsymbol{y}^{<t>})$$

Each training sequence has its own length

  - Where T_y is the length of the sequence labels
  - Similarly, T_x is the length of the sequence inputs

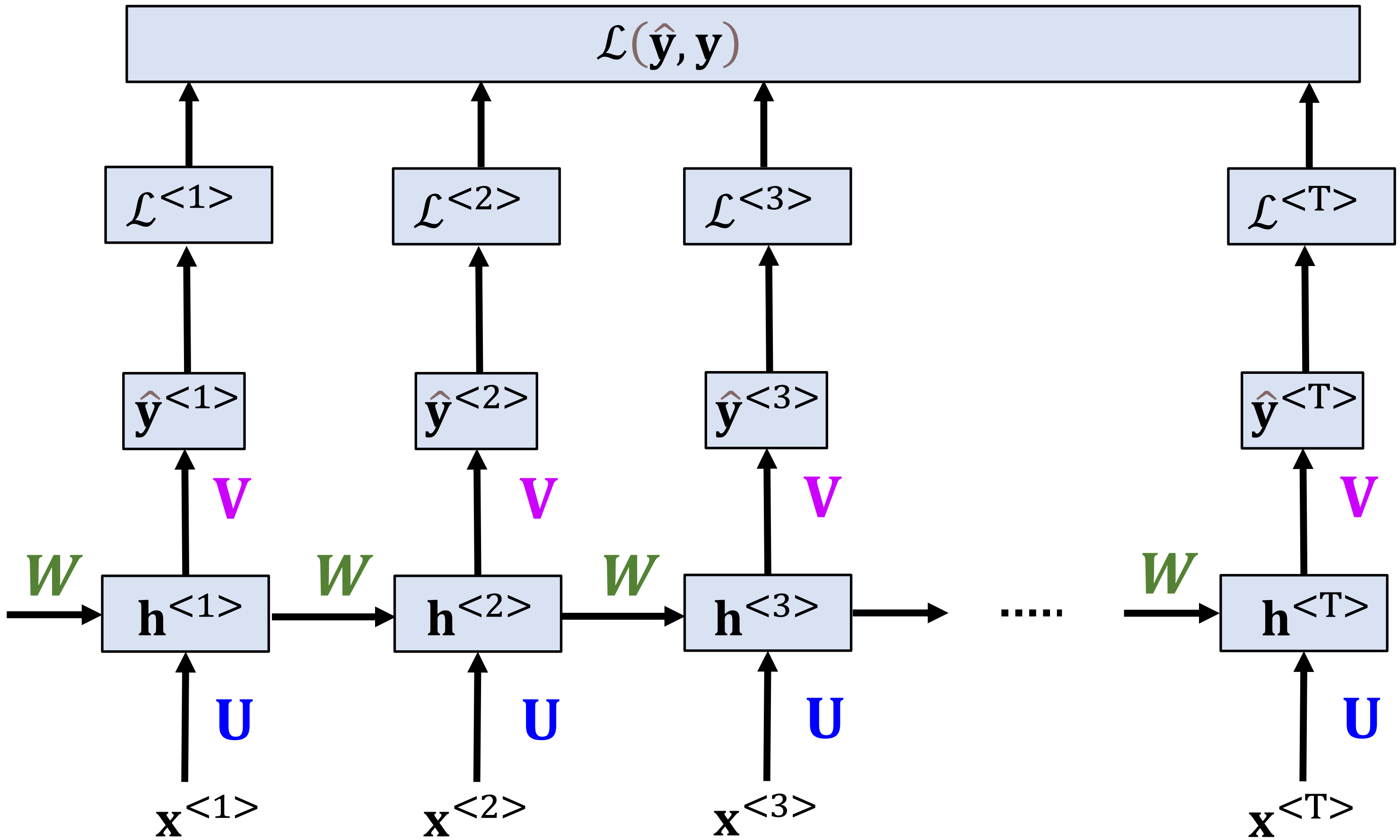
$$\mathbf{x} = \mathbf{x}^{<1>}, \mathbf{x}^{<2>}, \mathbf{x}^{<3>}, \ldots, \mathbf{x}^{<T_x>}$$

$$\hat{\mathbf{y}} = \hat{\mathbf{y}}^{<1>}, \hat{\mathbf{y}}^{<2>}, \hat{\mathbf{y}}^{<3>}, \ldots, \hat{\mathbf{y}}^{<T_y>}$$

  - We may have $T_x = T_y = T$ (as in the previous diagrams)
  - But RNNs can also handle $T_x \neq T_y$ as we shall see later

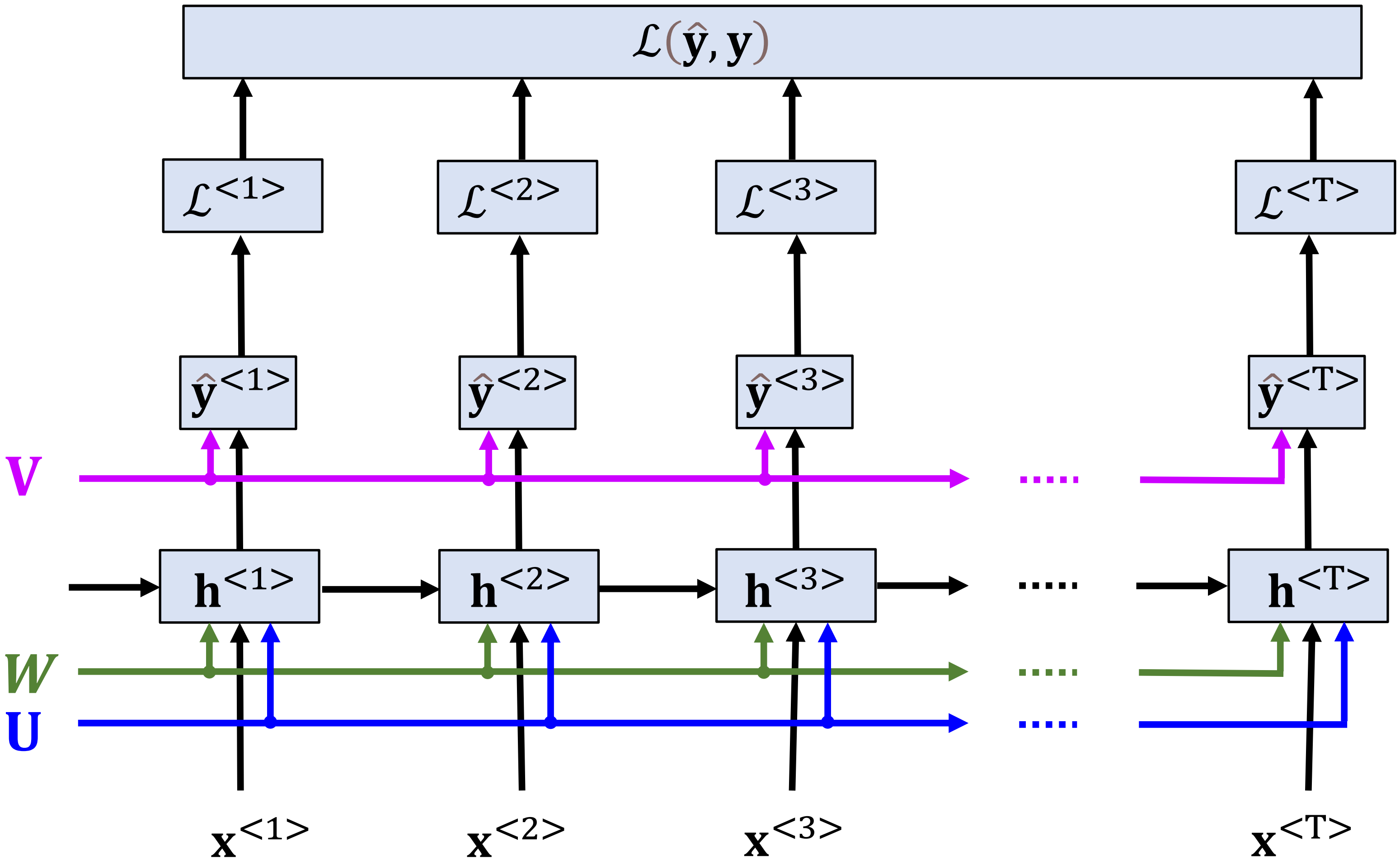
RNN Training: Backpropagation

- Because of parameter sharing, U, V and W matrices are unique



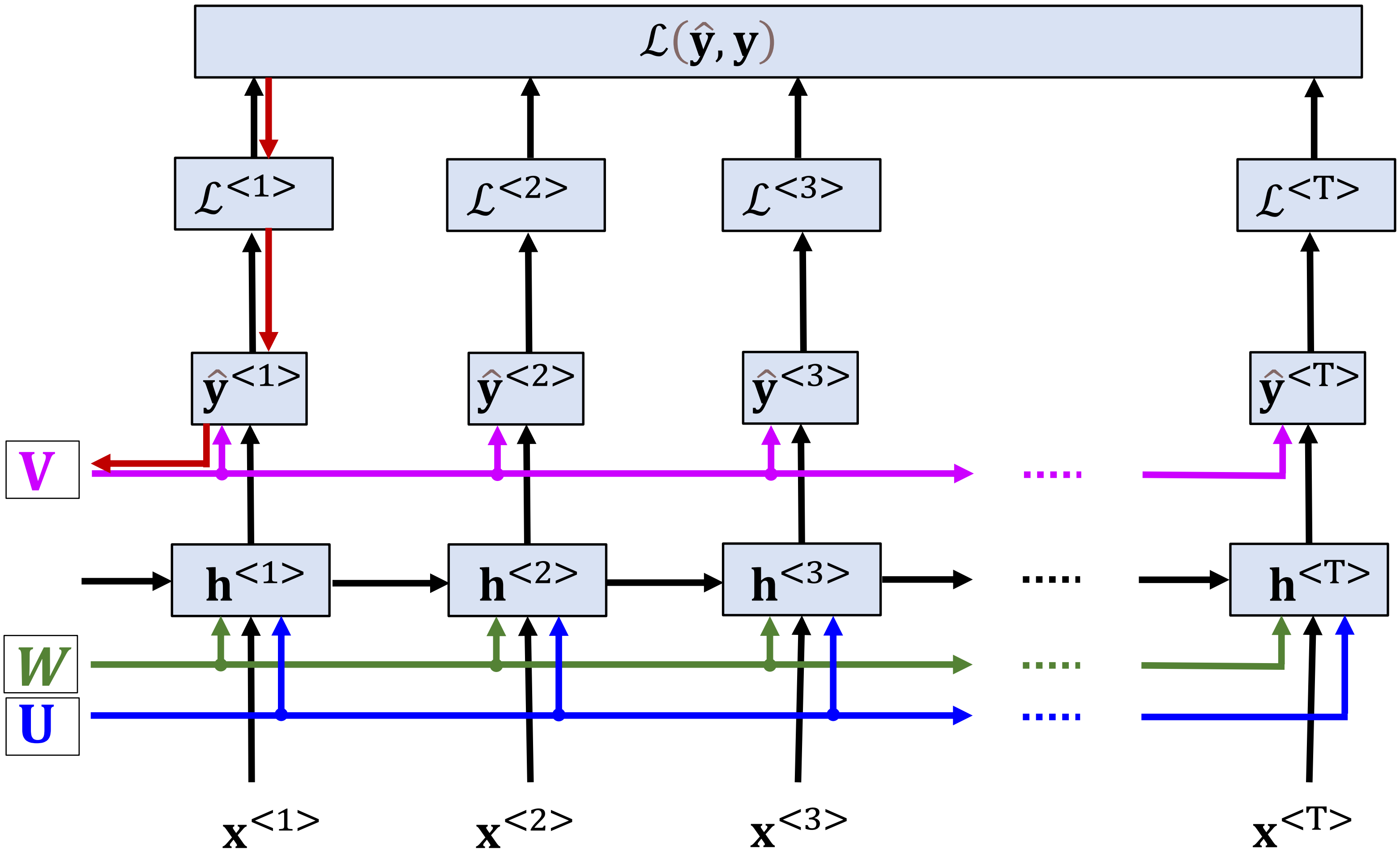
RNN Training: Backpropagation

- Because of parameter sharing, U, V and W matrices are unique



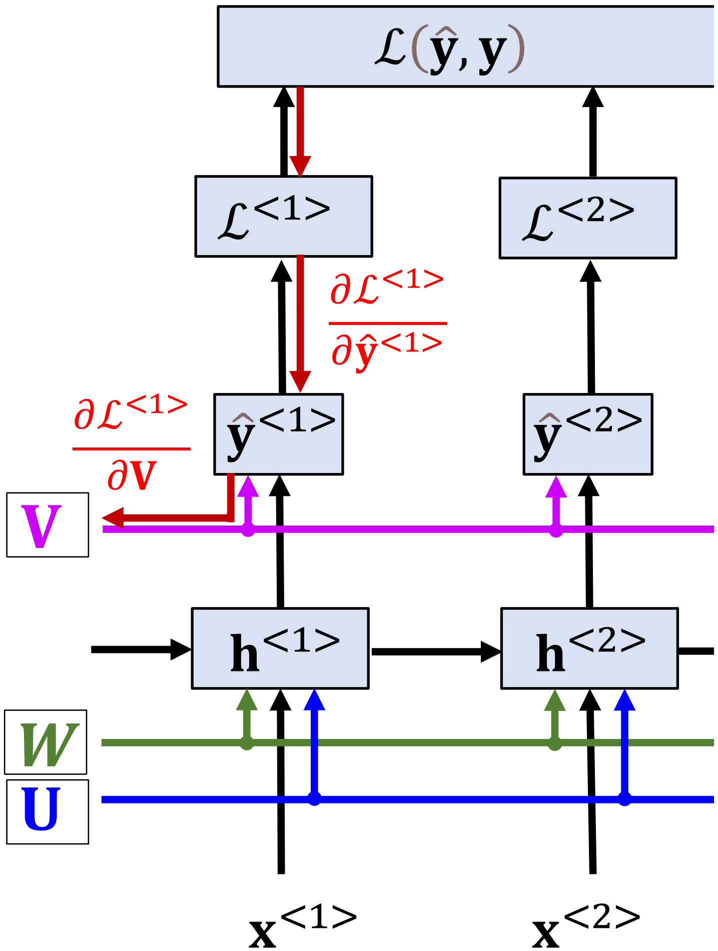
RNN Training: Backpropagation

- Let's start backpropagation for t = 1



RNN Training: Backpropagation

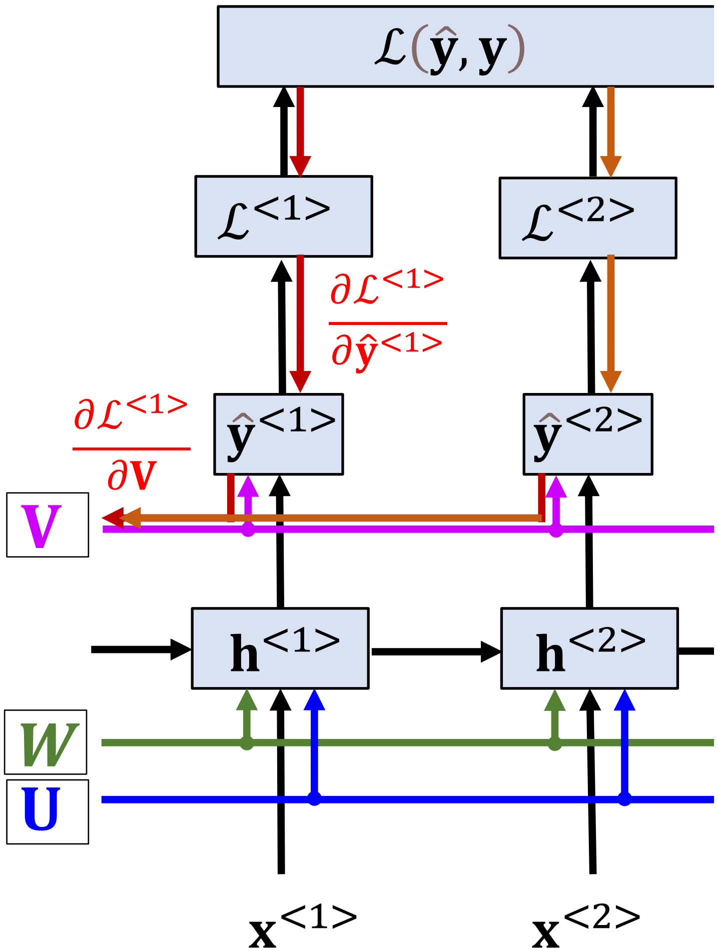
- Let's start backpropagation for t = 1



$$\frac{\partial \mathcal{L}^{<1>}}{\partial \mathbf{V}} = \frac{\partial \mathcal{L}^{<1>}}{\partial \hat{\mathbf{y}}^{<1>}} \frac{\partial \hat{\mathbf{y}}^{<1>}}{\partial \mathbf{V}}$$

RNN Training: Backpropagation

- Let's start backpropagation for t = 2

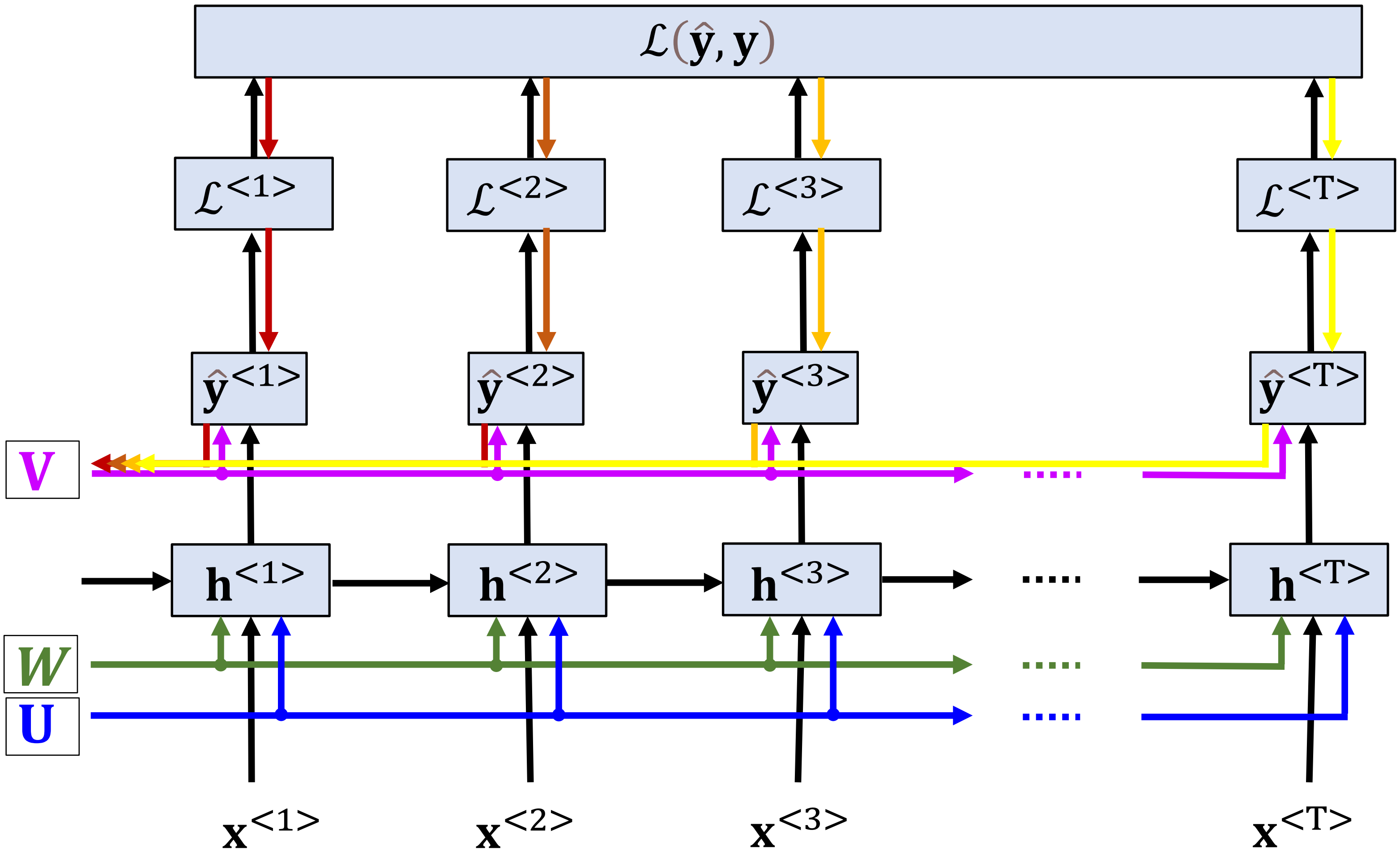


$$\frac{\partial \mathcal{L}^{<1>}}{\partial \mathbf{V}} = \frac{\partial \mathcal{L}^{<1>}}{\partial \hat{\mathbf{y}}^{<1>}} \frac{\partial \hat{\mathbf{y}}^{<1>}}{\partial \mathbf{V}}$$

$$\frac{\partial \mathcal{L}^{<2>}}{\partial \mathbf{V}} = \frac{\partial \mathcal{L}^{<2>}}{\partial \hat{\mathbf{y}}^{<2>}} \frac{\partial \hat{\mathbf{y}}^{<2>}}{\partial \mathbf{V}}$$

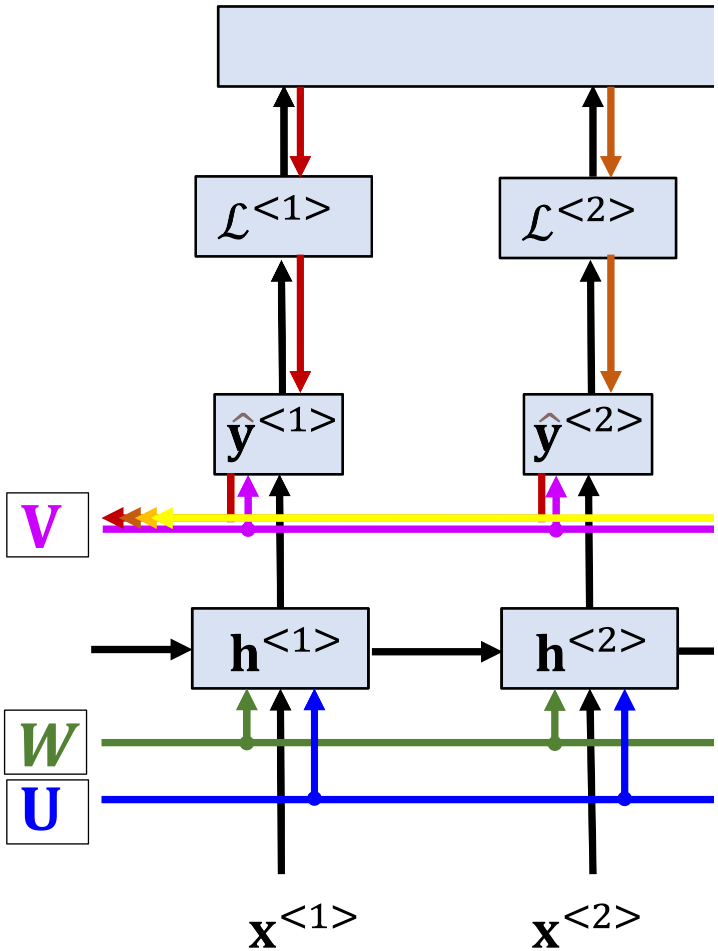
RNN Training: Backpropagation

- All time steps contribute to update **V**



RNN Training: Backpropagation

- All time steps contribute to update **V**



$$\frac{\partial \mathcal{L}^{<1>}}{\partial \mathbf{V}} = \frac{\partial \mathcal{L}^{<1>}}{\partial \hat{\mathbf{y}}^{<1>}} \frac{\partial \hat{\mathbf{y}}^{<1>}}{\partial \mathbf{V}}$$

$$\frac{\partial \mathcal{L}^{<2>}}{\partial \mathbf{V}} = \frac{\partial \mathcal{L}^{<2>}}{\partial \hat{\mathbf{y}}^{<2>}} \frac{\partial \hat{\mathbf{y}}^{<2>}}{\partial \mathbf{V}}$$

$$\frac{\partial \mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})}{\partial \mathbf{V}} = \sum_{t=1}^{\mathrm{T}} \frac{\partial \mathcal{L}^{<t>}}{\partial \hat{\mathbf{y}}^{<t>}} \frac{\partial \hat{\mathbf{y}}^{<t>}}{\partial \mathbf{V}}$$

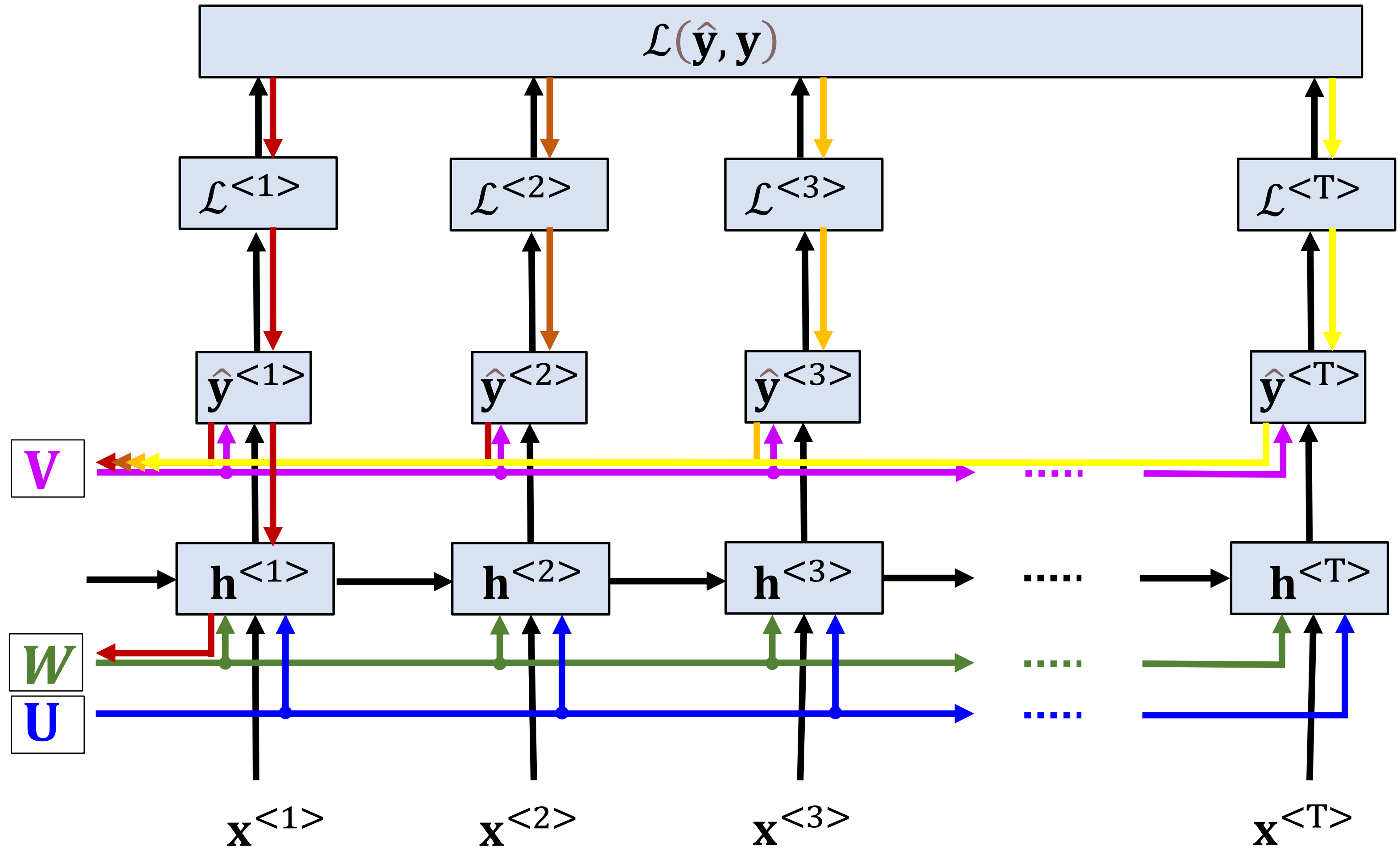
$$\hat{\mathbf{y}}^{<t>} = g_y\left(\boldsymbol{V}\mathbf{h}^{<t>} + \mathbf{b}_y\right)$$

$$\frac{\partial \hat{\mathbf{y}}^{<t>}}{\partial \mathbf{V}} = g_y'\left(\boldsymbol{V}\mathbf{h}^{<t>} + \mathbf{b}_y\right)\left(\mathbf{h}^{<t>}\right)^T$$

A similar derivation applies to backprop for $\mathbf{b}_{\mathrm{y}}$

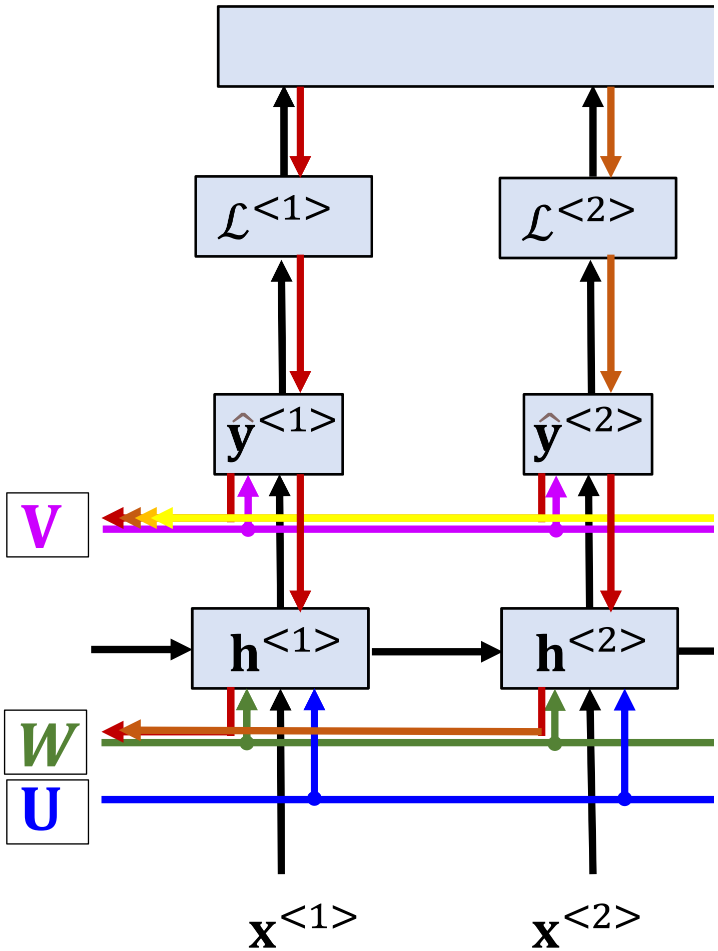
RNN Training: Backpropagation

- Let us continue to backprop for **W**



RNN Training: Backpropagation

- Let us continue to backprop for **W**



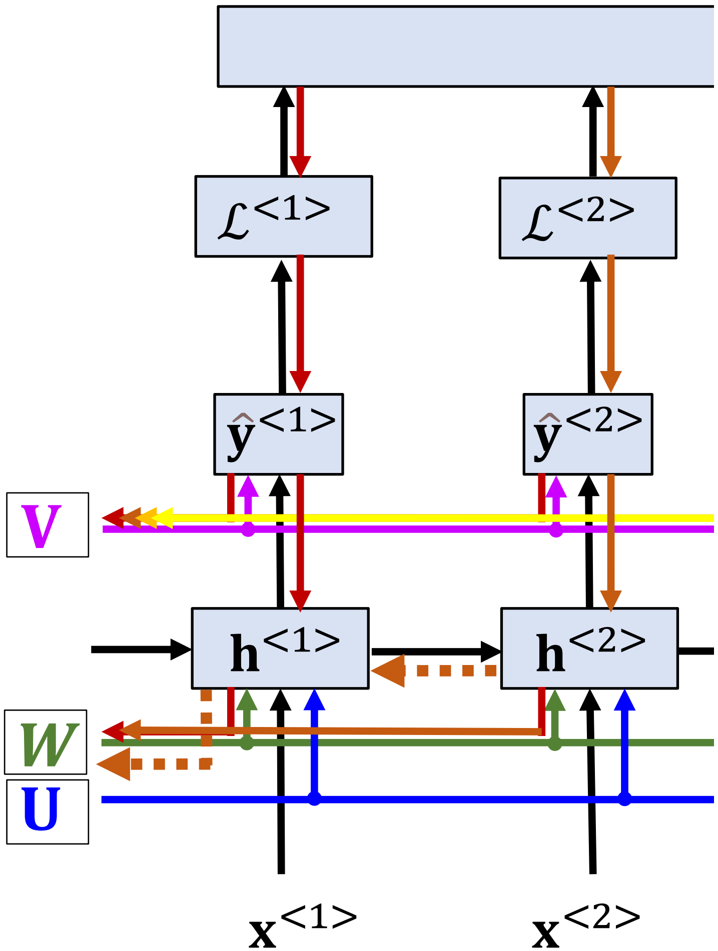
$$\frac{\partial \mathcal{L}^{<1>}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}^{<1>}}{\partial \hat{\mathbf{y}}^{<1>}} \frac{\partial \hat{\mathbf{y}}^{<1>}}{\partial \mathbf{h}^{<1>}} \frac{\partial \mathbf{h}^{<1>}}{\partial \mathbf{W}}$$

$$\frac{\partial \mathcal{L}^{<2>}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}^{<2>}}{\partial \hat{\mathbf{y}}^{<2>}} \frac{\partial \hat{\mathbf{y}}^{<2>}}{\partial \mathbf{h}^{<2>}} \frac{\partial \mathbf{h}^{<2>}}{\partial \mathbf{W}}$$

INCOMPLETE

RNN Training: Backpropagation

- Let us continue to backprop for **W**

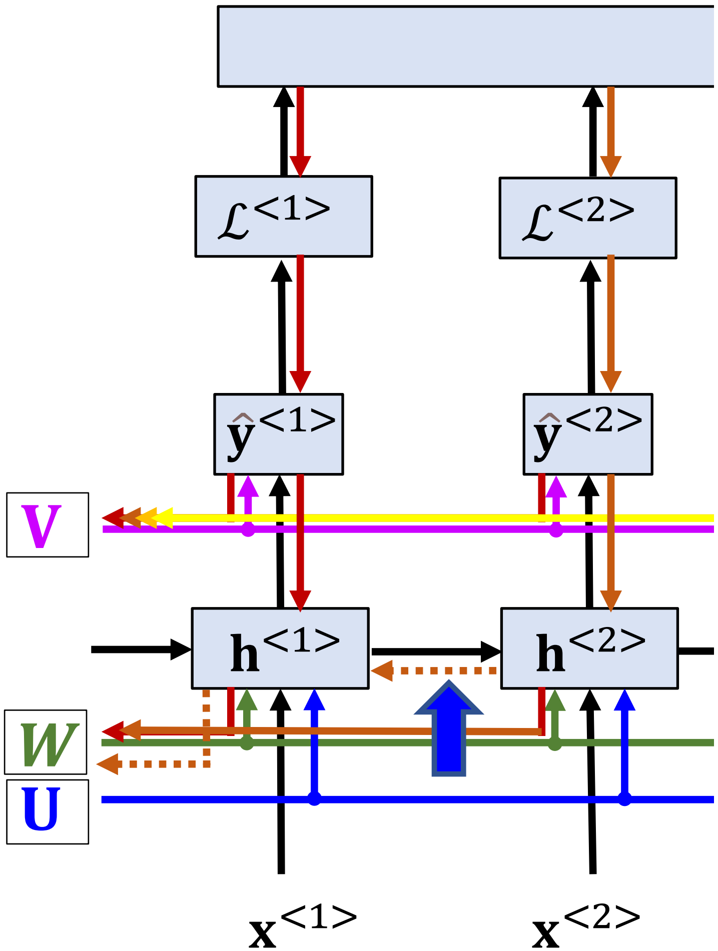


$$\frac{\partial \mathcal{L}^{<1>}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}^{<1>}}{\partial \hat{\mathbf{y}}^{<1>}} \frac{\partial \hat{\mathbf{y}}^{<1>}}{\partial \mathbf{h}^{<1>}} \frac{\partial \mathbf{h}^{<1>}}{\partial \mathbf{W}}$$

$$\frac{\partial \mathcal{L}^{<2>}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}^{<2>}}{\partial \hat{\mathbf{y}}^{<2>}} \frac{\partial \hat{\mathbf{y}}^{<2>}}{\partial \mathbf{h}^{<2>}} \frac{\partial \mathbf{h}^{<2>}}{\partial \mathbf{W}} + \\ + \frac{\partial \mathcal{L}^{<2>}}{\partial \hat{\mathbf{y}}^{<2>}} \frac{\partial \hat{\mathbf{y}}^{<2>}}{\partial \mathbf{h}^{<2>}} \frac{\partial \mathbf{h}^{<2>}}{\partial \mathbf{h}^{<1>}} \frac{\partial \mathbf{h}^{<1>}}{\partial \mathbf{W}}$$

Backpropagation Through Time (BTT)

- Let us continue to backprop for **W**

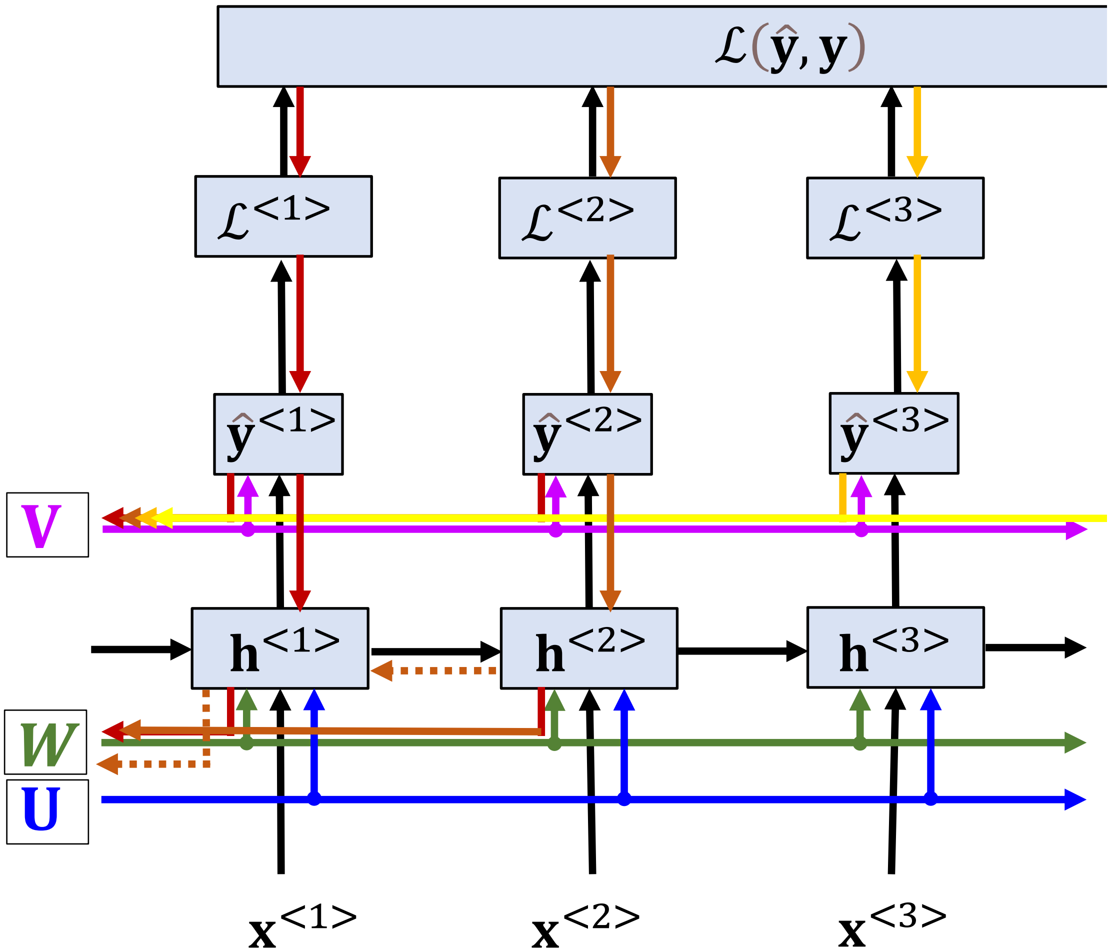


$$\frac{\partial \mathcal{L}^{<1>}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}^{<1>}}{\partial \hat{\mathbf{y}}^{<1>}} \frac{\partial \hat{\mathbf{y}}^{<1>}}{\partial \mathbf{h}^{<1>}} \frac{\partial \mathbf{h}^{<1>}}{\partial \mathbf{W}}$$

$$\frac{\partial \mathcal{L}^{<2>}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}^{<2>}}{\partial \hat{\mathbf{y}}^{<2>}} \frac{\partial \hat{\mathbf{y}}^{<2>}}{\partial \mathbf{h}^{<2>}} \frac{\partial \mathbf{h}^{<2>}}{\partial \mathbf{W}} + \\ + \frac{\partial \mathcal{L}^{<2>}}{\partial \hat{\mathbf{y}}^{<2>}} \frac{\partial \hat{\mathbf{y}}^{<2>}}{\partial \mathbf{h}^{<2>}} \boxed{\frac{\partial \mathbf{h}^{<2>}}{\partial \mathbf{h}^{<1>}}} \frac{\partial \mathbf{h}^{<1>}}{\partial \mathbf{W}}$$

Backpropagation Through Time (BTT)

- Let's look only at t=3

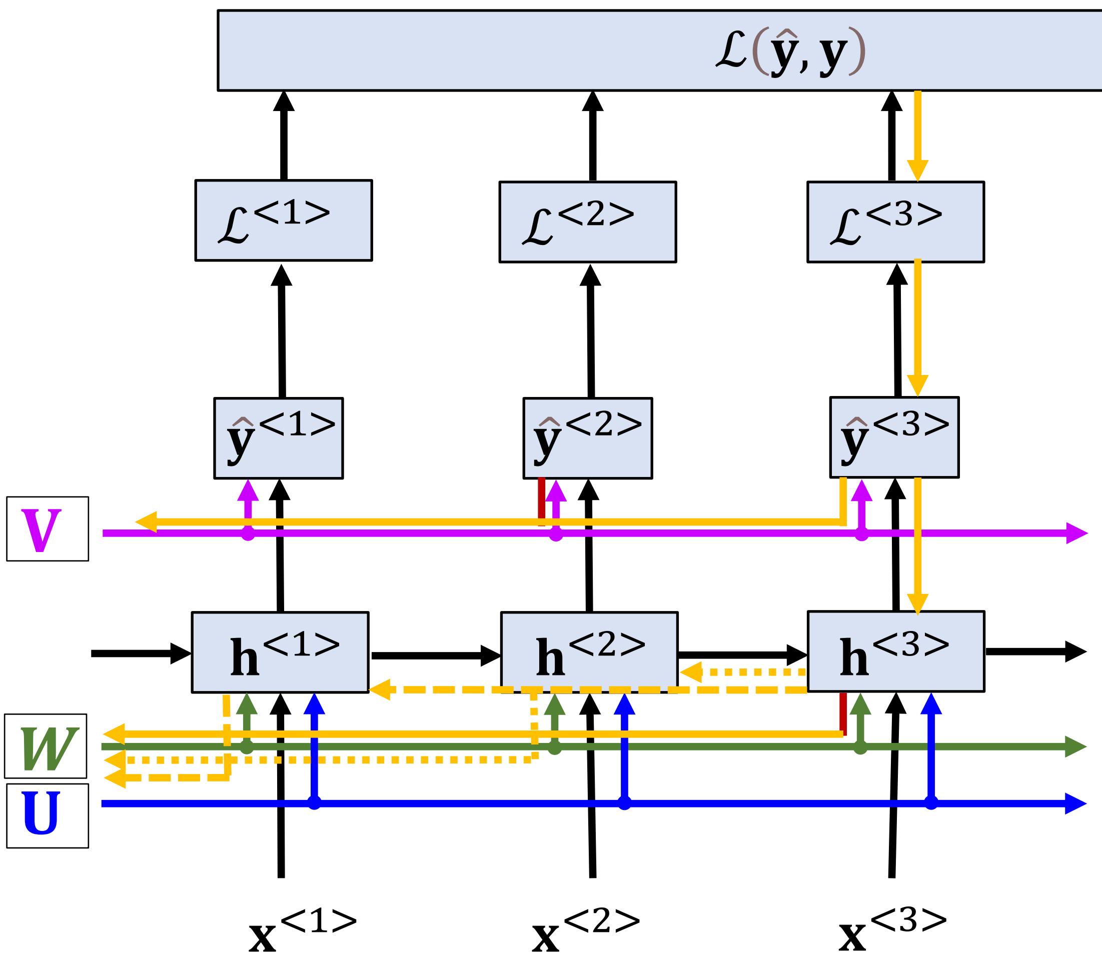


$$\frac{\partial \mathcal{L}^{<1>}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}^{<1>}}{\partial \hat{\mathbf{y}}^{<1>}} \frac{\partial \hat{\mathbf{y}}^{<1>}}{\partial \mathbf{h}^{<1>}} \frac{\partial \mathbf{h}^{<1>}}{\partial \mathbf{W}}$$

$$\frac{\partial \mathcal{L}^{<2>}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}^{<2>}}{\partial \hat{\mathbf{y}}^{<2>}} \frac{\partial \hat{\mathbf{y}}^{<2>}}{\partial \mathbf{h}^{<2>}} \frac{\partial \mathbf{h}^{<2>}}{\partial \mathbf{W}} + \\ + \frac{\partial \mathcal{L}^{<2>}}{\partial \hat{\mathbf{y}}^{<2>}} \frac{\partial \hat{\mathbf{y}}^{<2>}}{\partial \mathbf{h}^{<2>}} \frac{\partial \mathbf{h}^{<2>}}{\partial \mathbf{h}^{<1>}} \frac{\partial \mathbf{h}^{<1>}}{\partial \mathbf{W}}$$

Backpropagation Through Time (BTT)

- Let's look only at t=3



$$\frac{\partial \mathcal{L}^{<1>}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}^{<1>}}{\partial \hat{\mathbf{y}}^{<1>}} \frac{\partial \hat{\mathbf{y}}^{<1>}}{\partial \mathbf{h}^{<1>}} \frac{\partial \mathbf{h}^{<1>}}{\partial \mathbf{W}}$$

$$\frac{\partial \mathcal{L}^{<2>}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}^{<2>}}{\partial \hat{\mathbf{y}}^{<2>}} \frac{\partial \hat{\mathbf{y}}^{<2>}}{\partial \mathbf{h}^{<2>}} \frac{\partial \mathbf{h}^{<2>}}{\partial \mathbf{W}} + \\ + \frac{\partial \mathcal{L}^{<2>}}{\partial \hat{\mathbf{y}}^{<2>}} \frac{\partial \hat{\mathbf{y}}^{<2>}}{\partial \mathbf{h}^{<2>}} \frac{\partial \mathbf{h}^{<2>}}{\partial \mathbf{h}^{<1>}} \frac{\partial \mathbf{h}^{<1>}}{\partial \mathbf{W}}$$

$$\frac{\partial \mathcal{L}^{<3>}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}^{<3>}}{\partial \hat{\mathbf{y}}^{<3>}} \frac{\partial \hat{\mathbf{y}}^{<3>}}{\partial \mathbf{h}^{<2>}} \frac{\partial \mathbf{h}^{<2>}}{\partial \mathbf{W}} + \\ + \frac{\partial \mathcal{L}^{<3>}}{\partial \hat{\mathbf{y}}^{<3>}} \frac{\partial \hat{\mathbf{y}}^{<3>}}{\partial \mathbf{h}^{<3>}} \frac{\partial \mathbf{h}^{<3>}}{\partial \mathbf{h}^{<2>}} \frac{\partial \mathbf{h}^{<2>}}{\partial \mathbf{W}} + \\ + \frac{\partial \mathcal{L}^{<3>}}{\partial \hat{\mathbf{y}}^{<3>}} \frac{\partial \hat{\mathbf{y}}^{<3>}}{\partial \mathbf{h}^{<3>}} \frac{\partial \mathbf{h}^{<3>}}{\partial \mathbf{h}^{<2>}} \frac{\partial \mathbf{h}^{<2>}}{\partial \mathbf{h}^{<1>}} \frac{\partial \mathbf{h}^{<1>}}{\partial \mathbf{W}}$$

Total Derivatives

Consider the formula for the hidden state at a generic time instant “t”

$$\mathbf{h}^{<t>} = g_h(\mathbf{U}\mathbf{x}^{<t>} + \boldsymbol{W}\mathbf{h}^{<t-1>} + \mathbf{b}_h)$$

Total Derivatives

Consider the formula for the hidden state at a generic time instant "t"

$$\mathbf{h}^{<t>} = g_h(\mathbf{U}\mathbf{x}^{<t>} + \boldsymbol{W}\mathbf{h}^{<t-1>} + \mathbf{b}_h)$$

To differentiate with respect to **W** we need to realize that:

$$\mathbf{h}^{<t-1>} = g_h(\mathbf{U}\mathbf{x}^{<t-1>} + \boldsymbol{W}\mathbf{h}^{<t-2>} + \mathbf{b}_h)$$

$$\Rightarrow \quad \mathbf{h}^{<t>} = g_h(\mathbf{U}\mathbf{x}^{<t>} + \boldsymbol{W}\mathbf{h}^{<t-1>}(\mathbf{W}) + \mathbf{b}_h)$$

Total Derivatives

Consider the formula for the hidden state at a generic time instant "t"

$$\mathbf{h}^{<t>} = g_h(\mathbf{U}\mathbf{x}^{<t>} + \boldsymbol{W}\mathbf{h}^{<t-1>} + \mathbf{b}_h)$$

To differentiate with respect to **W** we need to realize that:

$$\mathbf{h}^{<t-1>} = g_h(\mathbf{U}\mathbf{x}^{<t-1>} + \boldsymbol{W}\mathbf{h}^{<t-2>} + \mathbf{b}_h)$$

$$\Rightarrow \quad \mathbf{h}^{<t>} = g_h(\mathbf{U}\mathbf{x}^{<t>} + \boldsymbol{W}\mathbf{h}^{<t-1>}(\mathbf{W}) + \mathbf{b}_h)$$

Then we need to use the total derivative:

$$\frac{d\mathbf{h}^{<t>}}{d\mathbf{W}} = \frac{\partial\mathbf{h}^{<t>}}{\partial\mathbf{W}} + \frac{\partial\mathbf{h}^{<t>}}{\partial\mathbf{h}^{<t-1>}}\frac{\partial\mathbf{h}^{<t-1>}}{\partial\mathbf{W}} + \cdots + \left(\prod_{i=2}^{t}\frac{\partial\mathbf{h}^{<i>}}{\partial\mathbf{h}^{<i-1>}}\right)\frac{\partial\mathbf{h}^{<1>}}{\partial\mathbf{W}}$$

Total Derivatives

Consider the formula for the hidden state at a generic time instant "t"

$$\mathbf{h}^{<t>} = g_h(\mathbf{Ux}^{<t>} + \boldsymbol{W}\mathbf{h}^{<t-1>} + \mathbf{b}_h)$$

To differentiate with respect to **W** we need to realize that:

$$\mathbf{h}^{<t-1>} = g_h(\mathbf{Ux}^{<t-1>} + \boldsymbol{W}\mathbf{h}^{<t-2>} + \mathbf{b}_h)$$

$$\Rightarrow \quad \mathbf{h}^{<t>} = g_h(\mathbf{Ux}^{<t>} + \boldsymbol{W}\mathbf{h}^{<t-1>}(\mathbf{W}) + \mathbf{b}_h)$$

Then we need to use the total derivative:

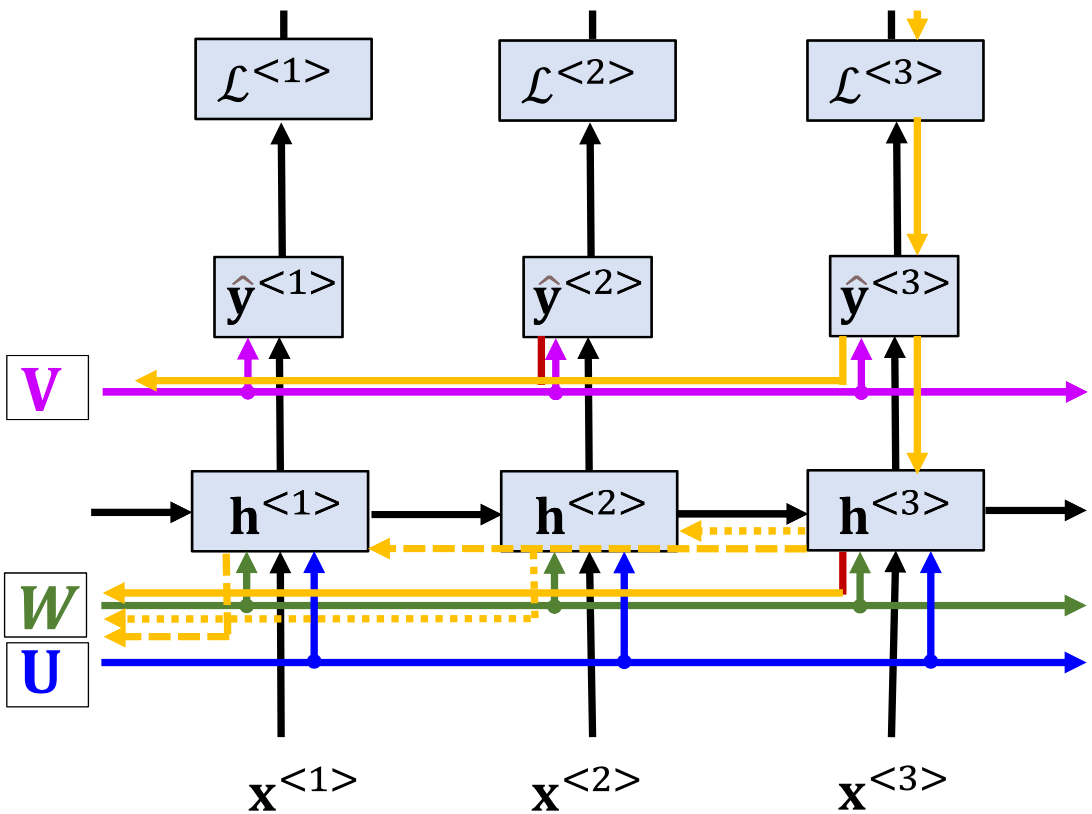
$$\frac{d\mathbf{h}^{<t>}}{d\mathbf{W}} = \frac{\partial\mathbf{h}^{<t>}}{\partial\mathbf{W}} + \frac{\partial\mathbf{h}^{<t>}}{\partial\mathbf{h}^{<t-1>}}\frac{\partial\mathbf{h}^{<t-1>}}{\partial\mathbf{W}} + \cdots + \left(\prod_{i=2}^{t}\frac{\partial\mathbf{h}^{<i>}}{\partial\mathbf{h}^{<i-1>}}\right)\frac{\partial\mathbf{h}^{<1>}}{\partial\mathbf{W}}$$

Therefore, the total contribution of the loss at time "t" to backprop for **W** is:

$$\frac{\partial\mathcal{L}^{<t>}}{\partial\mathbf{W}} = \frac{\partial\mathcal{L}^{<t>}}{\partial\hat{\mathbf{y}}^{<t>}}\frac{\partial\hat{\mathbf{y}}^{<t>}}{\partial\mathbf{h}^{<t>}}\sum_{k=1}^{t}\left(\prod_{i=k+1}^{t}\frac{\partial\mathbf{h}^{<i>}}{\partial\mathbf{h}^{<i-1>}}\right)\frac{\partial\mathbf{h}^{<k>}}{\partial\mathbf{W}}$$

Backpropagation Through Time (BTT)

$$\frac{\partial \mathcal{L}^{<t>}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}^{<t>}}{\partial \hat{\mathbf{y}}^{<t>}} \frac{\partial \hat{\mathbf{y}}^{<t>}}{\partial \mathbf{h}^{<t>}} \sum_{k=1}^{t} \boxed{\left(\prod_{i=k+1}^{t} \frac{\partial \mathbf{h}^{<i>}}{\partial \mathbf{h}^{<i-1>}} \right)} \frac{\partial \mathbf{h}^{<k>}}{\partial \mathbf{W}}$$



$$\frac{\partial \mathcal{L}^{<1>}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}^{<1>}}{\partial \hat{\mathbf{y}}^{<1>}} \frac{\partial \hat{\mathbf{y}}^{<1>}}{\partial \mathbf{h}^{<1>}} \frac{\partial \mathbf{h}^{<1>}}{\partial \mathbf{W}}$$

$$\frac{\partial \mathcal{L}^{<2>}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}^{<2>}}{\partial \hat{\mathbf{y}}^{<2>}} \frac{\partial \hat{\mathbf{y}}^{<2>}}{\partial \mathbf{h}^{<2>}} \frac{\partial \mathbf{h}^{<2>}}{\partial \mathbf{W}} + \frac{\partial \mathcal{L}^{<2>}}{\partial \hat{\mathbf{y}}^{<2>}} \frac{\partial \hat{\mathbf{y}}^{<2>}}{\partial \mathbf{h}^{<2>}} \boxed{\frac{\partial \mathbf{h}^{<2>}}{\partial \mathbf{h}^{<1>}}} \frac{\partial \mathbf{h}^{<1>}}{\partial \mathbf{W}}$$

$$\frac{\partial \mathcal{L}^{<3>}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}^{<3>}}{\partial \hat{\mathbf{y}}^{<3>}} \frac{\partial \hat{\mathbf{y}}^{<3>}}{\partial \mathbf{h}^{<2>}} \frac{\partial \mathbf{h}^{<2>}}{\partial \mathbf{W}} + \frac{\partial \mathcal{L}^{<3>}}{\partial \hat{\mathbf{y}}^{<3>}} \frac{\partial \hat{\mathbf{y}}^{<3>}}{\partial \mathbf{h}^{<3>}} \boxed{\frac{\partial \mathbf{h}^{<3>}}{\partial \mathbf{h}^{<2>}}} \frac{\partial \mathbf{h}^{<2>}}{\partial \mathbf{W}} + \frac{\partial \mathcal{L}^{<3>}}{\partial \hat{\mathbf{y}}^{<3>}} \frac{\partial \hat{\mathbf{y}}^{<3>}}{\partial \mathbf{h}^{<3>}} \boxed{\frac{\partial \mathbf{h}^{<3>}}{\partial \mathbf{h}^{<2>}} \frac{\partial \mathbf{h}^{<2>}}{\partial \mathbf{h}^{<1>}}} \frac{\partial \mathbf{h}^{<1>}}{\partial \mathbf{W}}$$

Backpropagation Through Time (BTT)

$$\frac{\partial \mathcal{L}^{<t>}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}^{<t>}}{\partial \hat{\mathbf{y}}^{<t>}} \frac{\partial \hat{\mathbf{y}}^{<t>}}{\partial \mathbf{h}^{<t>}} \sum_{k=1}^{t} \boxed{\left(\prod_{i=k+1}^{t} \frac{\partial \mathbf{h}^{<i>}}{\partial \mathbf{h}^{<i-1>}} \right)} \frac{\partial \mathbf{h}^{<k>}}{\partial \mathbf{W}}$$

- The product of Jacobians makes it possible BTT
  - It helps the RNN capture the temporal dependencies of the sequences
  - Further temporal dependencies implies products with more terms
    - This makes RNN training more challenging
    - Two main issues
      - Vanishing gradients
      - Exploding gradients

Exploding Gradient

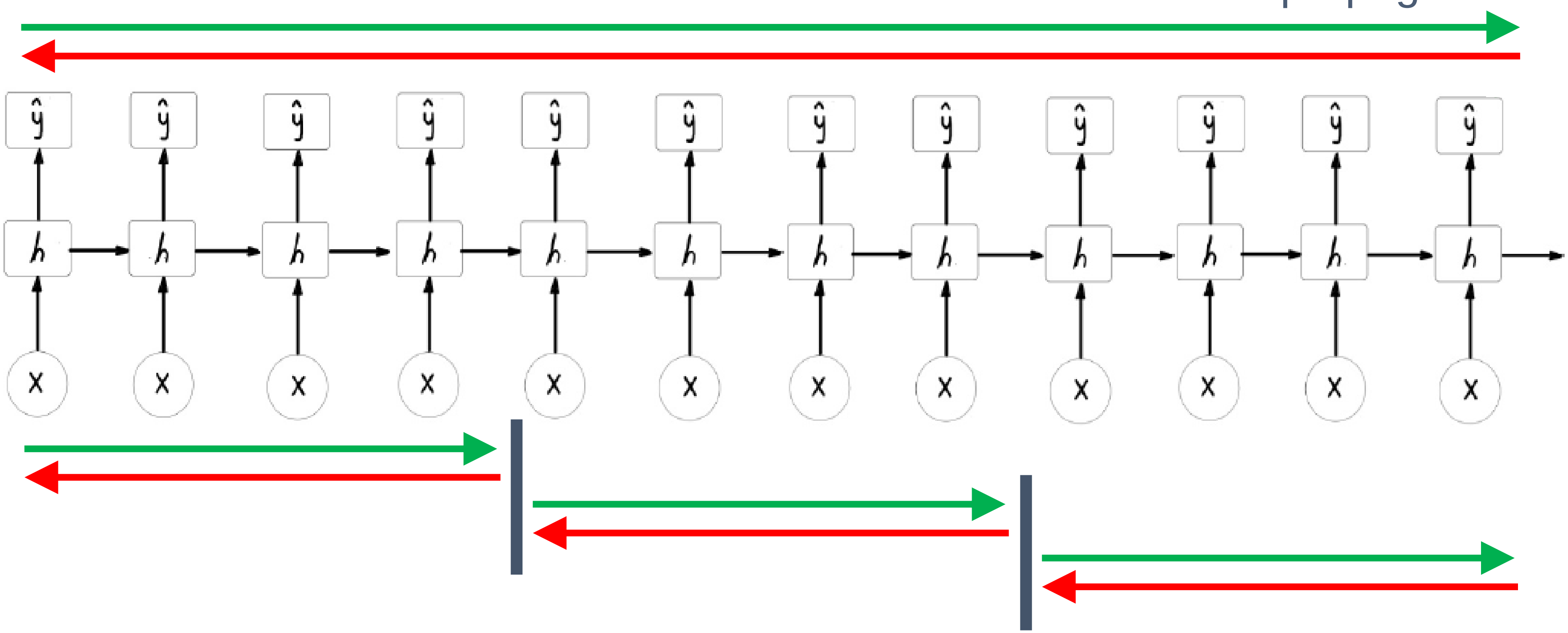
$$\frac{\partial \mathcal{L}^{<t>}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}^{<t>}}{\partial \hat{\mathbf{y}}^{<t>}} \frac{\partial \hat{\mathbf{y}}^{<t>}}{\partial \mathbf{h}^{<t>}} \sum_{k=1}^{t} \boxed{\left(\prod_{i=k+1}^{t} \frac{\partial \mathbf{h}^{<i>}}{\partial \mathbf{h}^{<i-1>}} \right)} \frac{\partial \mathbf{h}^{<k>}}{\partial \mathbf{W}}$$

- In case the gradients have norms consistently above 1
  - The product of inner-state Jacobians grows exponentially
  - Can lead to excessively large gradients
    - Relatively easy to detect
      - Loss curves
    - Two simple solutions to deal with this
      - Gradient Clipping
      - Truncated Backpropagation

Truncated Backpropagation

$$\frac{\partial \mathcal{L}^{<t>}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}^{<t>}}{\partial \hat{\mathbf{y}}^{<t>}} \frac{\partial \hat{\mathbf{y}}^{<t>}}{\partial \mathbf{h}^{<t>}} \sum_{k=1}^{t} \boxed{\left(\prod_{i=k+1}^{t} \frac{\partial \mathbf{h}^{<i>}}{\partial \mathbf{h}^{<i-1>}} \right)} \frac{\partial \mathbf{h}^{<k>}}{\partial \mathbf{W}}$$

Forward & Backward propagation



Truncated Backward propagation

Vanishing Gradient

$$\frac{\partial \mathcal{L}^{<t>}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}^{<t>}}{\partial \hat{\mathbf{y}}^{<t>}} \frac{\partial \hat{\mathbf{y}}^{<t>}}{\partial \mathbf{h}^{<t>}} \sum_{k=1}^{t} \boxed{\left(\prod_{i=k+1}^{t} \frac{\partial \mathbf{h}^{<i>}}{\partial \mathbf{h}^{<i-1>}} \right)} \frac{\partial \mathbf{h}^{<k>}}{\partial \mathbf{W}}$$

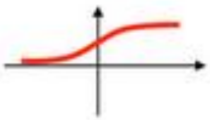
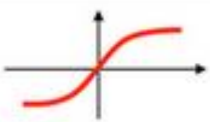
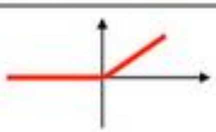
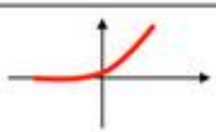
- In case the gradients have norms consistently below 1
  - The product of inner-state Jacobians decreases exponentially
  - Can lead to excessively small gradients
    - Failure to capture long-range dependencies
    - Difficult to assess
      - Low gradients can occur naturally due to absence of actual long-range dependencies
    - Several solutions have been proposed

Ways to deal with vanishing gradients

1. Use ReLU activation functions
   - We can understand the rational from the partial derivatives:

$$\mathbf{h}^{<t>} = g_h(\mathbf{U}\mathbf{x}^{<t>} + \boldsymbol{W}\mathbf{h}^{<t-1>} + \mathbf{b}_h)$$

$$\frac{\partial \mathbf{h}^{<t>}}{\partial \mathbf{h}^{<t-1>}} = diag(g_h'(\mathbf{U}\mathbf{x}^{<t>} + \mathbf{W}\mathbf{h}^{<t-1>} + \mathbf{b}_h))\mathbf{W}$$

Logistic (sigmoid)	$\phi(z) = \frac{1}{1+e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^z)$	Multi-layer Neural Networks	

Ways to deal with vanishing gradients

2. Use orthogonal initialization / parameterization
   - We can understand the rational from the partial derivatives:

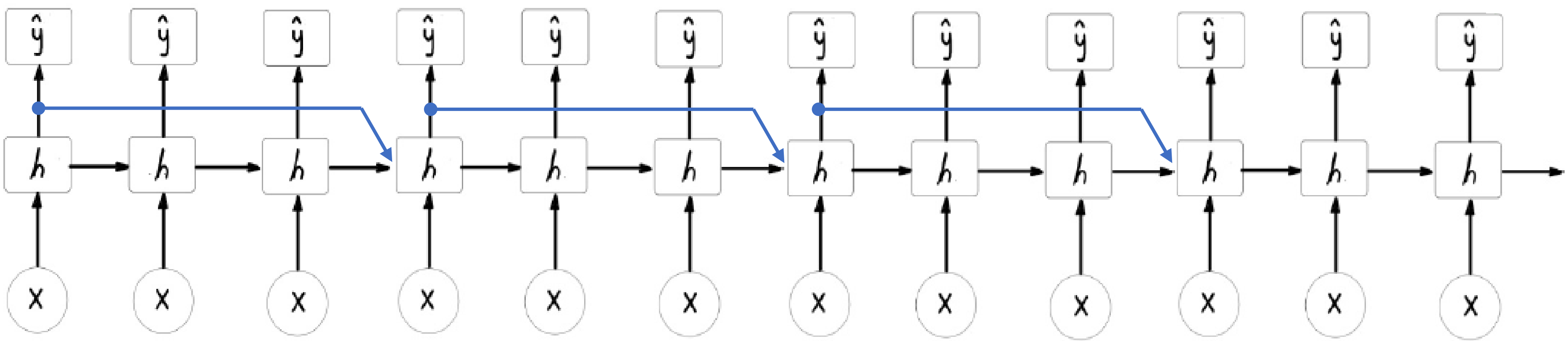
$$\mathbf{h}^{<t>} = g_h(\mathbf{U}\mathbf{x}^{<t>} + \boldsymbol{W}\mathbf{h}^{<t-1>} + \mathbf{b}_h)$$

$$\frac{\partial \mathbf{h}^{<t>}}{\partial \mathbf{h}^{<t-1>}} = diag\left(g_h'(\mathbf{U}\mathbf{x}^{<t>} + \mathbf{W}\mathbf{h}^{<t-1>} + \mathbf{b}_h)\right)\mathbf{W}$$

   - If we set **W** = **Q**, where **Q** is an orthogonal matrix
     - The spectral norm will be 1
     - The norm of multiple matrix products will also be 1
       - Easy to initialize
       - Need for reparameterization to maintain **W** orthogonal

Ways to deal with vanishing gradients

3. Skip connections
   - Also known as short-cuts or residual connections



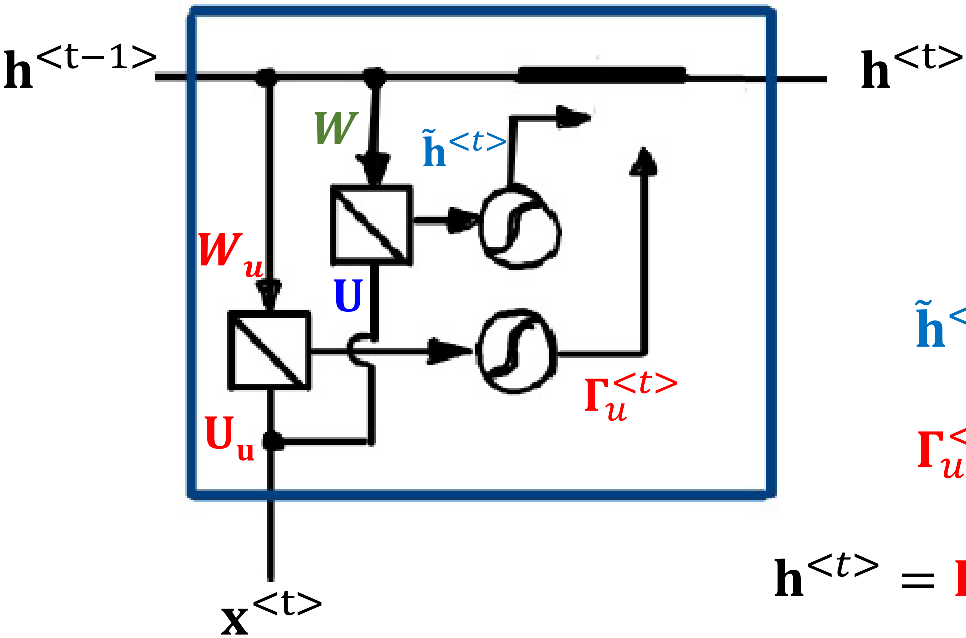
- In this way, the gradient can backpropagate much further
- The number of states to skip is a hyperparameter

Ways to deal with vanishing gradients

4. Gated Units
   - The vanishing gradient problem has motivated important advances in RNNs
     - Gated Recurrent Unit (GRU)
     - Long Short Term Memory (LSTM)
   - The key idea is the use of gates
     - The gate can be fully opened of fully closed
     - Therefore
       - It allows to block propagation
       - It also allows lossless propagation
     - The operation of the gate is determined automatically by the RNN

A basic Gated Unit

- We start with a "simplified" version of GRU



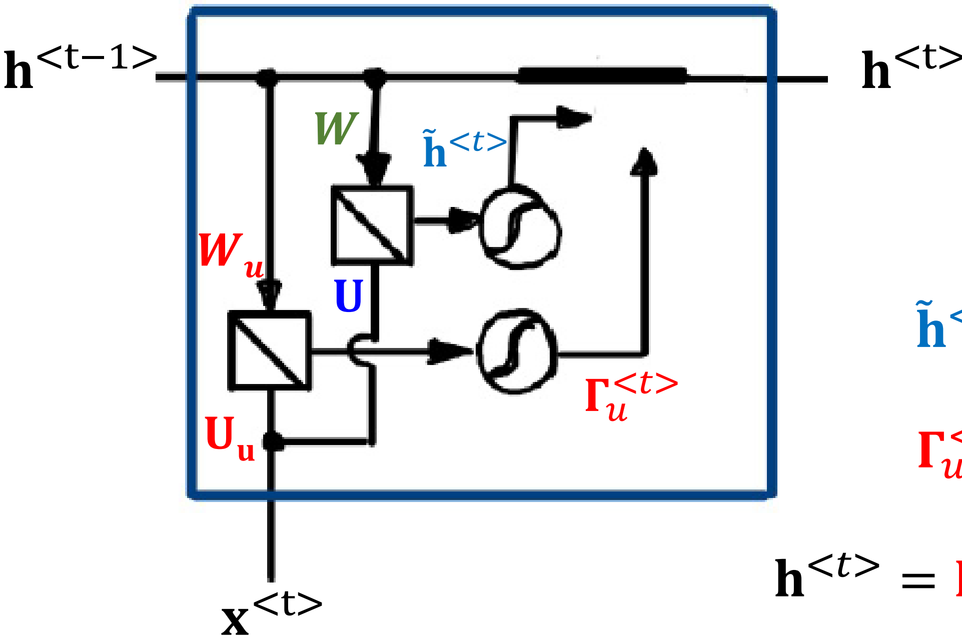
$$\tilde{\mathbf{h}}^{<t>} = g_h(\mathbf{U}_{\boldsymbol{h}}\mathbf{x}^{<t>} + \boldsymbol{W}_{\boldsymbol{h}}\mathbf{h}^{<t-1>} + \mathbf{b}_h)$$

$$\boldsymbol{\Gamma}_u^{<t>} = g_u(\mathbf{U}_{\boldsymbol{u}}\mathbf{x}^{<t>} + \boldsymbol{W}_{\boldsymbol{u}}\mathbf{h}^{<t-1>} + \mathbf{b}_u)$$

$$\mathbf{h}^{<t>} = \boldsymbol{\Gamma}_u^{<t>} \odot \tilde{\mathbf{h}}^{<t>} + (1 - \boldsymbol{\Gamma}_u^{<t>}) \odot \mathbf{h}^{<t-1>}$$

A basic Gated Unit

- We start with a "simplified" version of GRU



$$\tilde{\mathbf{h}}^{<t>} = g_h(\mathbf{U}_h\mathbf{x}^{<t>} + \boldsymbol{W}_h\mathbf{h}^{<t-1>} + \mathbf{b}_h)$$

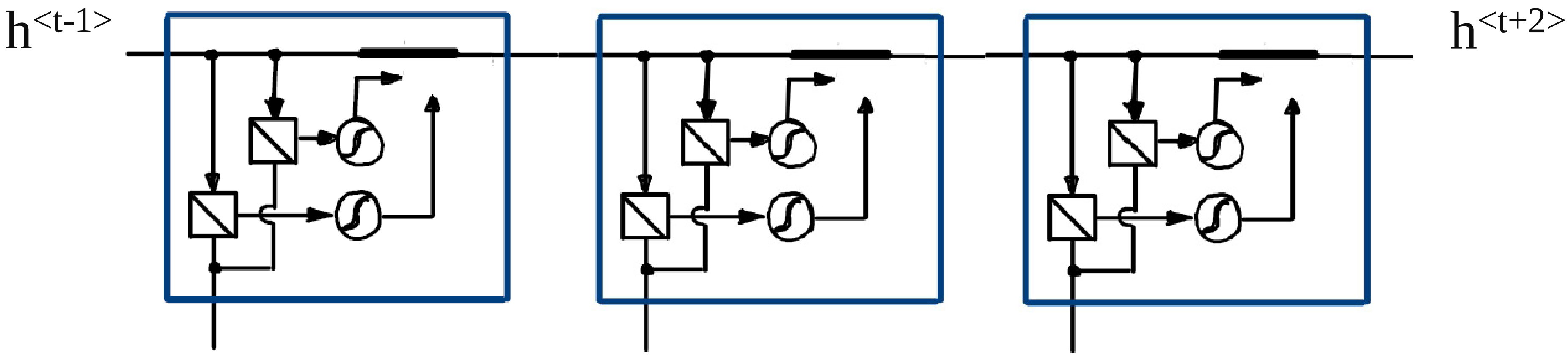
$$\boldsymbol{\Gamma}_u^{<t>} = g_u(\mathbf{U}_u\mathbf{x}^{<t>} + \boldsymbol{W}_u\mathbf{h}^{<t-1>} + \mathbf{b}_u)$$

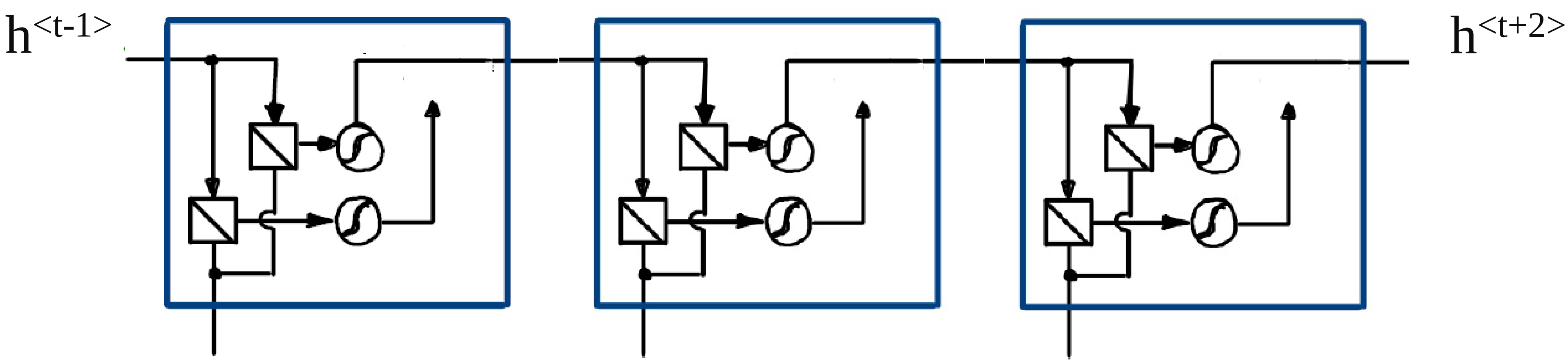
$$\mathbf{h}^{<t>} = \boldsymbol{\Gamma}_u^{<t>} \odot \tilde{\mathbf{h}}^{<t>} + (1 - \boldsymbol{\Gamma}_u^{<t>}) \odot \mathbf{h}^{<t-1>}$$

- If the gate is closed (0), the unit can keep its state unchanged
- If the gate is opened (1), the unit can update its state

Chains of Gated Units

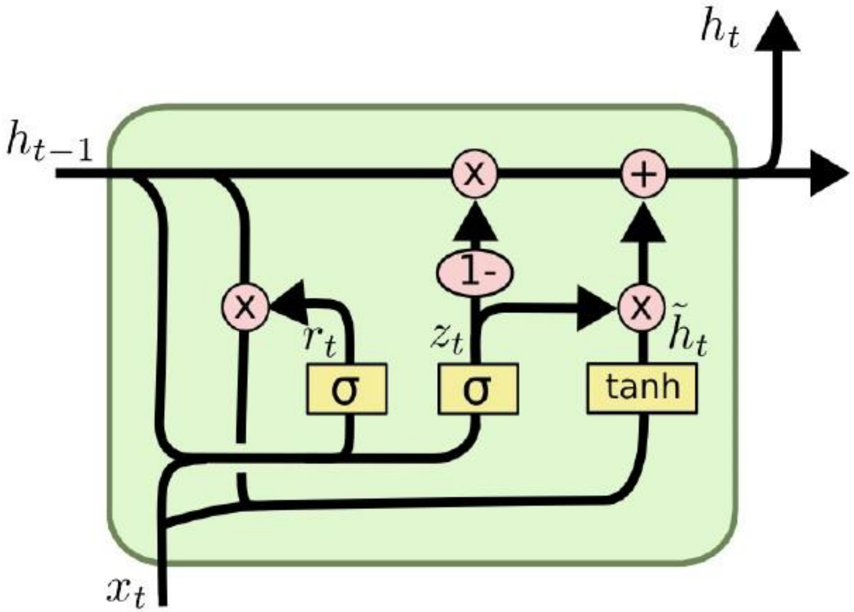
- If the gate is closed (0), the unit can keep its state unchanged
- f the gate is opened (1), the unit can update its state





Gated Recurrent Unit

- Full equations

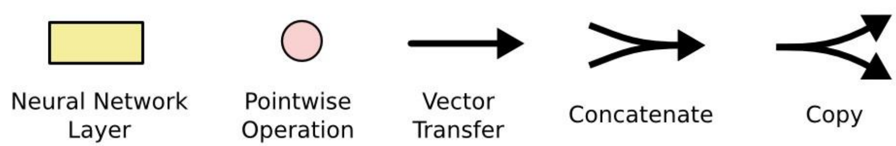


$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

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

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

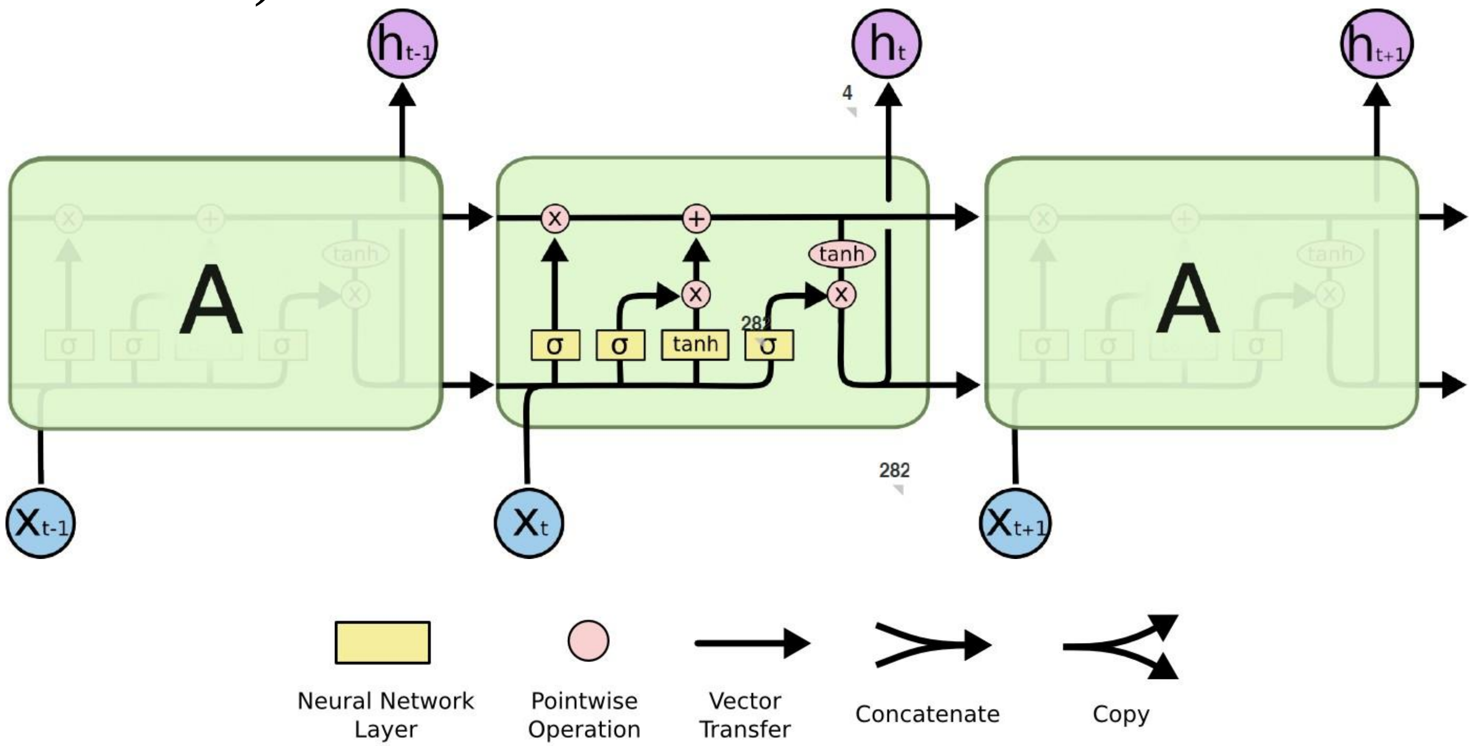
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$



Cho, Kyunghyun, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning phrase representations using RNN encoder-decoder for

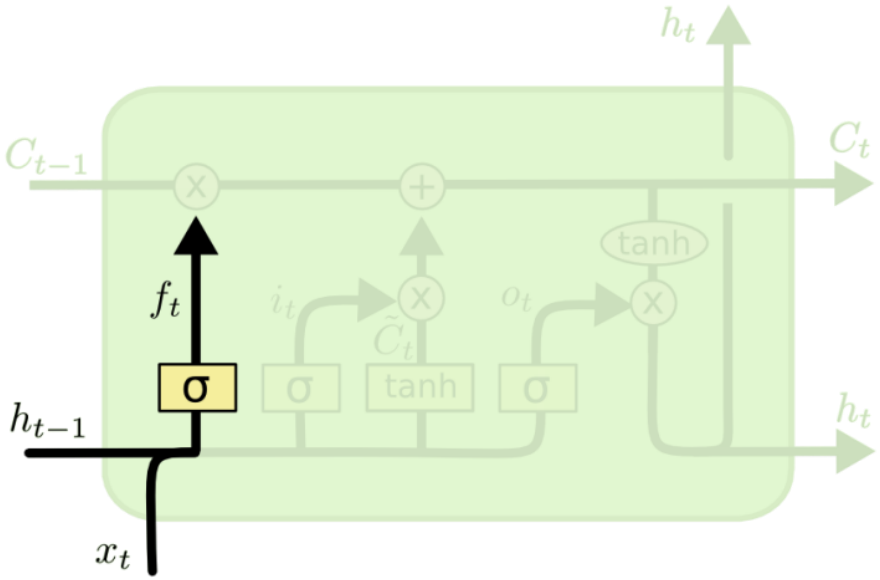
Long-Short Term Memory (LSTM)

- Memory cell separated from the inner state.
- This was actually the first gated unit (it was presented well before GRU)



Long-Short Term Memory (LSTM)

- Gates controlled by sigmoid activations
  - Input, Output, Forget and Update gates



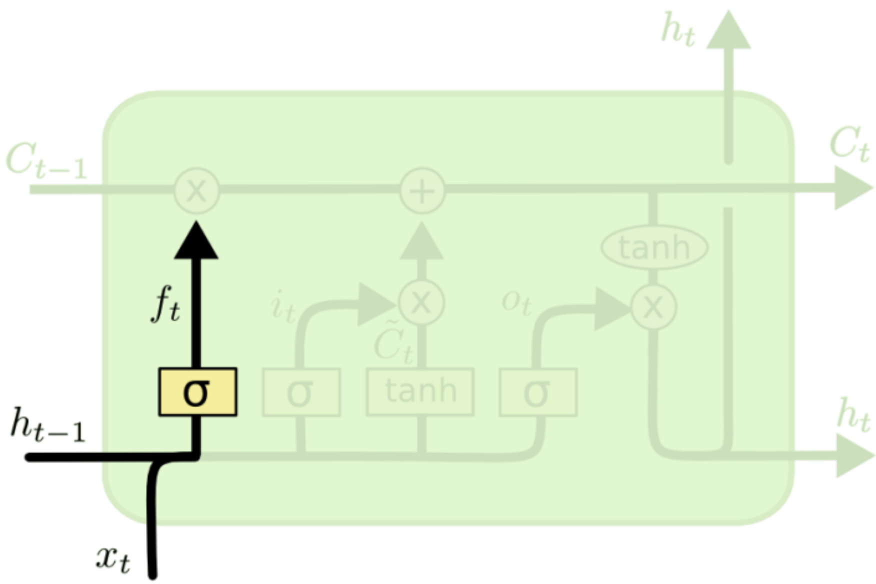
Forget Gate:

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

Figure: Cristopher Olah, "Understanding LSTM Networks" (2015)

Long-Short Term Memory (LSTM)

- Gates controlled by sigmoid activations
  - Input, Output, Forget and Update gates



Forget Gate:

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

LANGUAGE MODELING

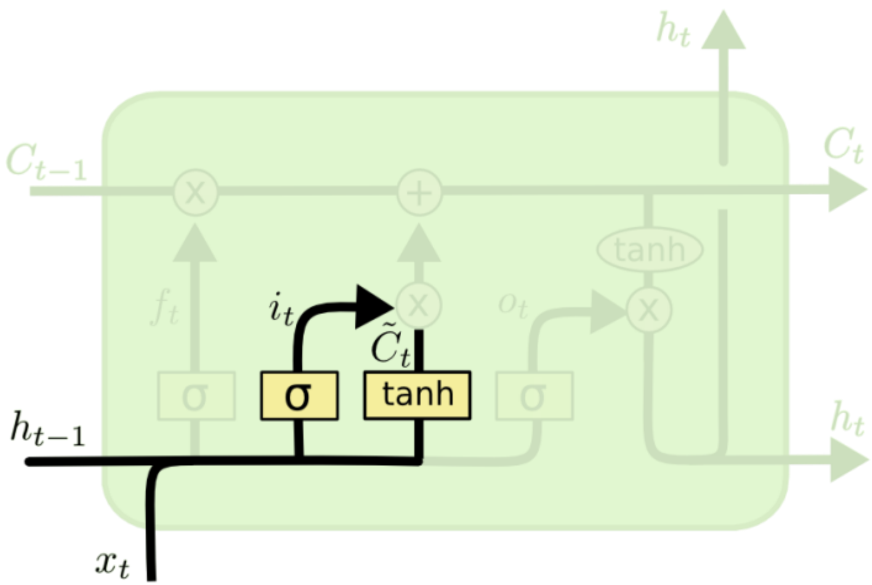
John is a very active boy and Anna is a very quiet girl

Forget about ”male” gender

Figure: Cristopher Olah, “Understanding LSTM Networks” (2015)

Long-Short Term Memory (LSTM)

- Gates controlled by sigmoid activations
  - Input, Output, Forget and Update gates



Input Gate Layer

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

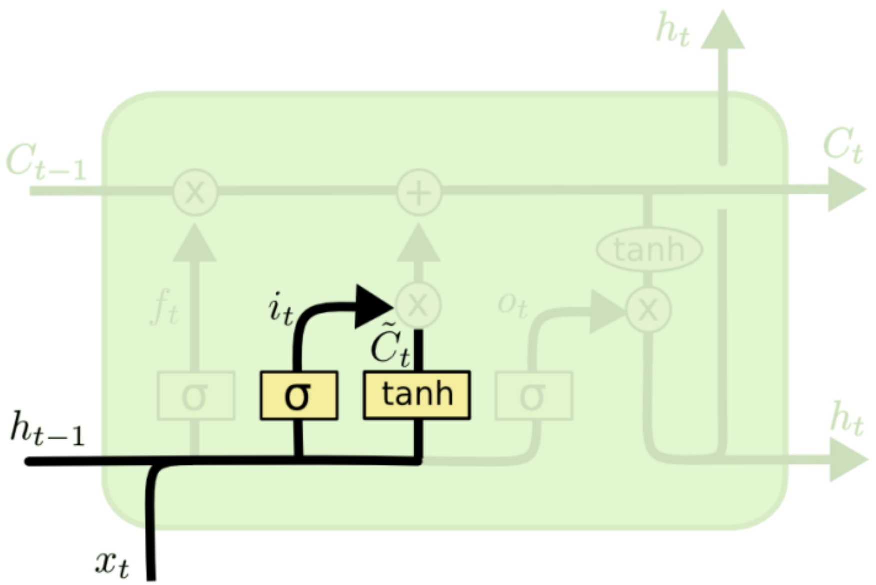
New contribution to cell state

$$\tilde{C}_t = \underbrace{\tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)}_{\text{Classic neuron}}$$

Figure: Cristopher Olah, "Understanding LSTM Networks" (2015)

Long-Short Term Memory (LSTM)

- Gates controlled by sigmoid activations
  - Input, Output, Forget and Update gates



Input Gate Layer

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

LANGUAGE MODELING

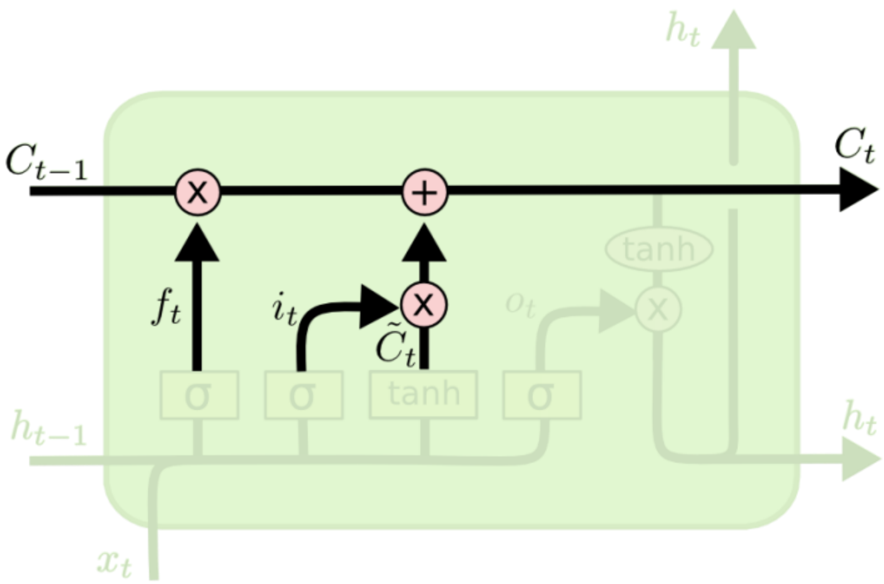
John is a very active boy and Anna is a very quiet girl

Input about ”female” gender

Figure: Cristopher Olah, “Understanding LSTM Networks” (2015)

Long-Short Term Memory (LSTM)

- Gates controlled by sigmoid activations
  - Input, Output, Forget and Update gates



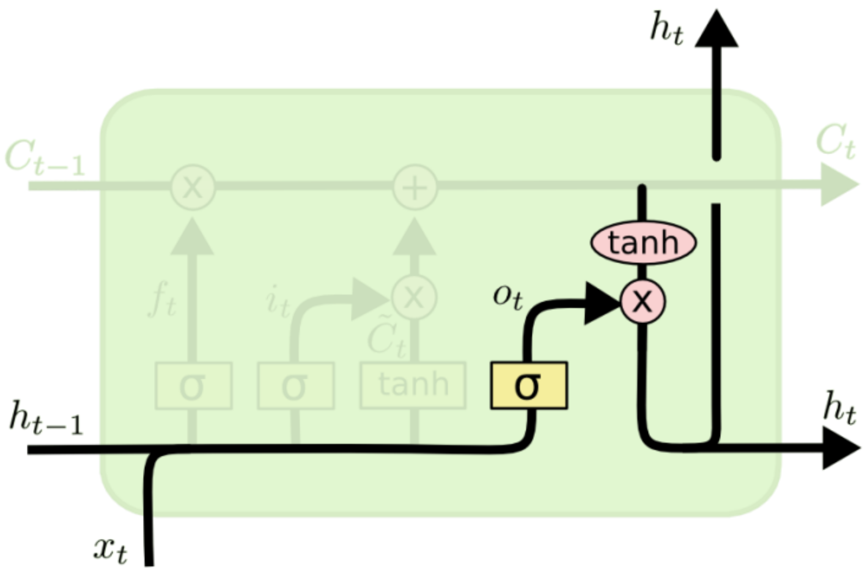
Update Cell State (memory):

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

Figure: Cristopher Olah, "Understanding LSTM Networks" (2015)

Long-Short Term Memory (LSTM)

- Gates controlled by sigmoid activations
  - Input, Output, Forget and Update gates



Output Gate Layer

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

Output to next layer

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

Figure: Cristopher Olah, "Understanding LSTM Networks" (2015)

RNN Applications

RNNs are suitable for any application in which we can exploit sequential dependencies

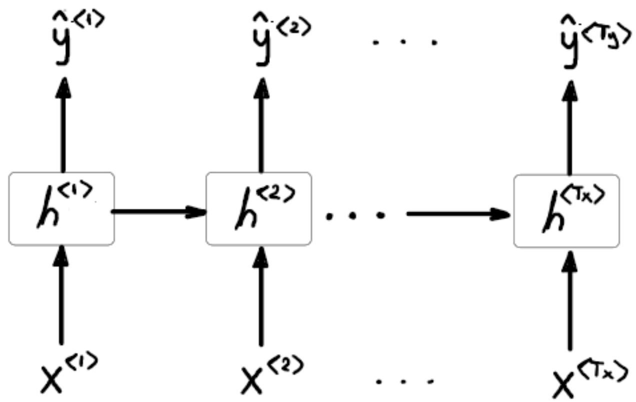
	x - input	y - output
Speech recognition		Will we ever forget it
Music generation	∅	
Sentiment classification	"Decent effort. The plot could have been better."	
DNA analysis	ACTGTACCCATGTGACTGCCC	ACTGTACCCATGTGACTGCCC
Automatic translation	Vés a pastar fang	Get lost.
Video activity recognition		Running
Name entity recognition	John White traveled to New Jersey last week	John White traveled to New Jersey last week

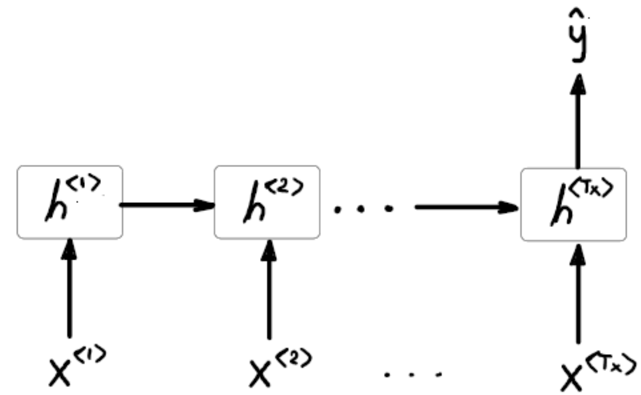
https://datahacker.rs/003-rnn-architectural-types-of-different-recurrent-neural-networks/

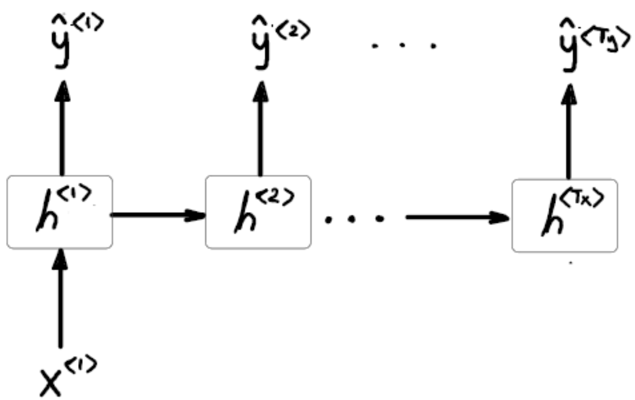
RNN Input/Output Architectures

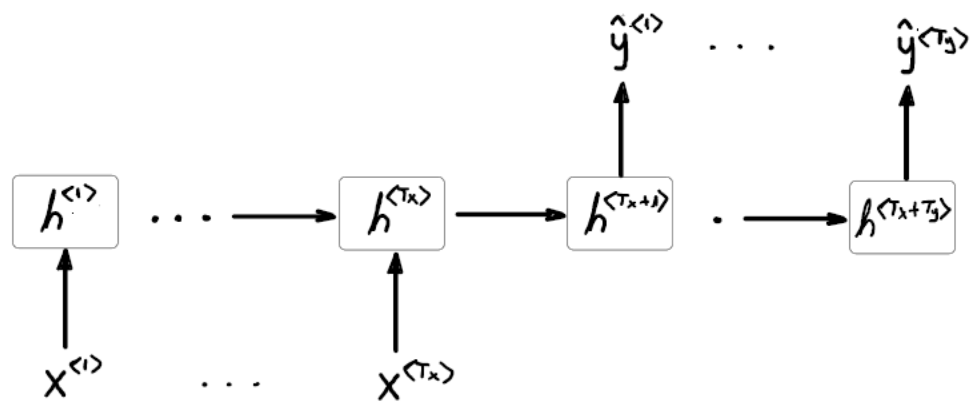
- RNNs can handle different input and output lengths

$$\mathbf{x} = \mathbf{x}^{<1>}, \mathbf{x}^{<2>}, \mathbf{x}^{<3>}, \dots, \mathbf{x}^{<T_x>} \qquad \hat{\mathbf{y}} = \hat{\mathbf{y}}^{<1>}, \hat{\mathbf{y}}^{<2>}, \hat{\mathbf{y}}^{<3>}, \dots, \hat{\mathbf{y}}^{<T_y>}$$





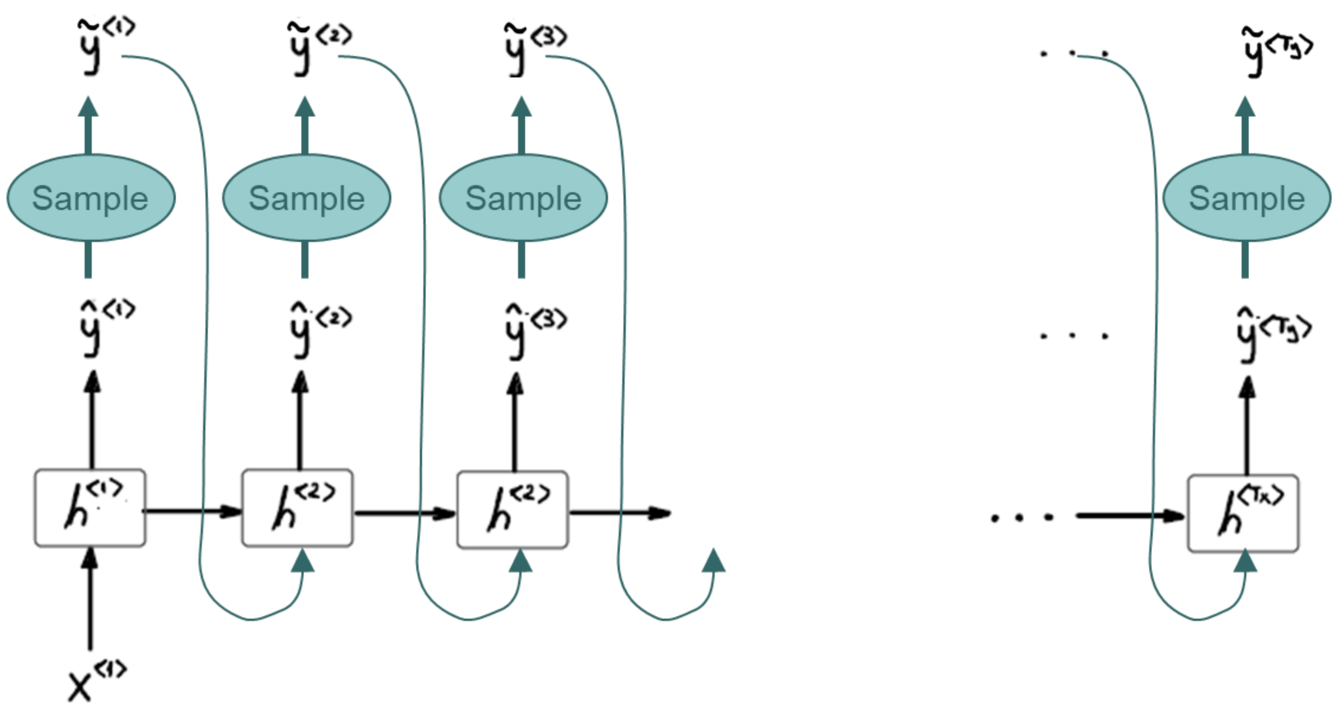




One-to-Many Architecture

- Useful for Sequence Generation

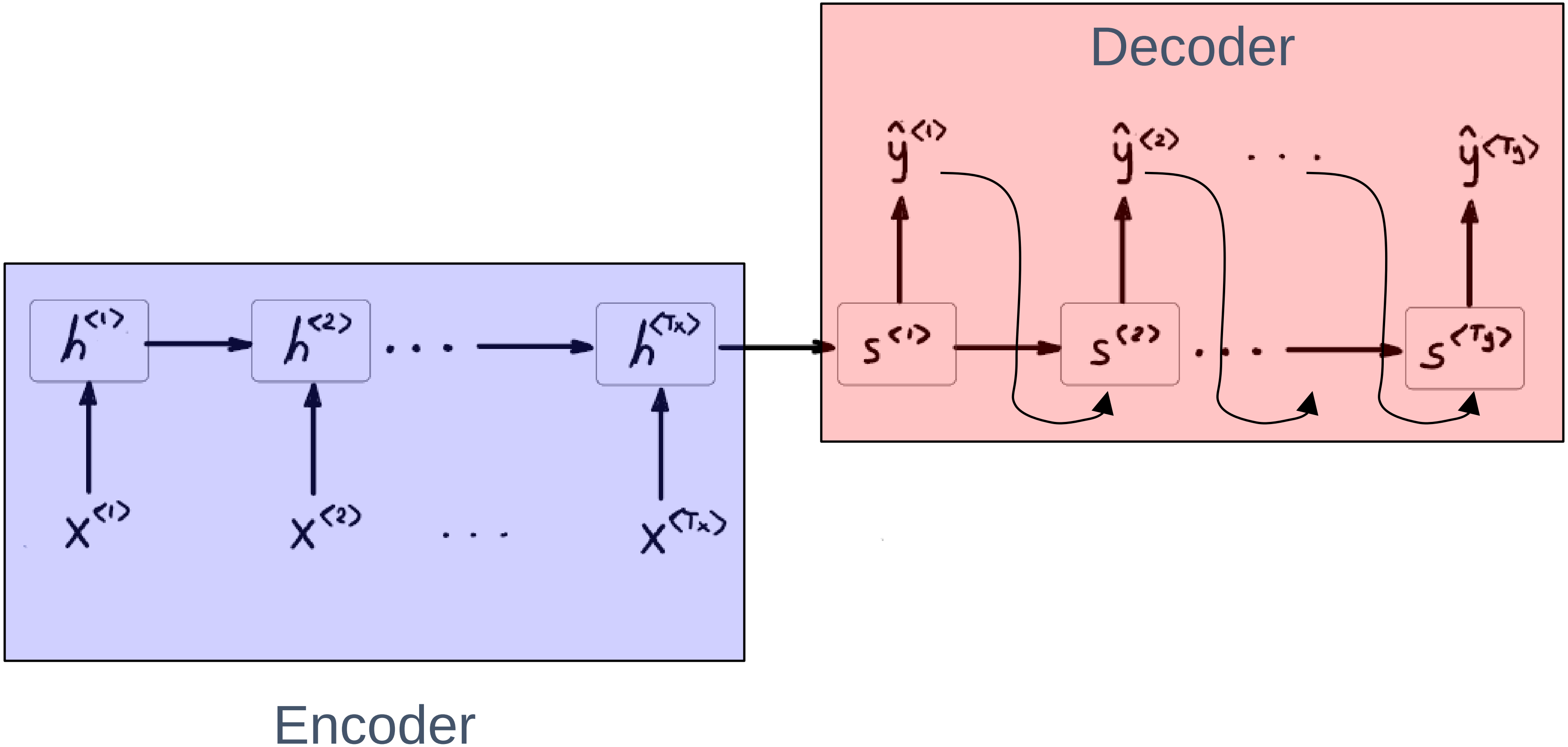
$$\mathbf{x} = \mathbf{x}^{<1>}, \mathbf{x}^{<2>}, \mathbf{x}^{<3>}, \dots, \mathbf{x}^{<T_x>} \qquad \hat{\mathbf{y}} = \hat{\mathbf{y}}^{<1>}, \hat{\mathbf{y}}^{<2>}, \hat{\mathbf{y}}^{<3>}, \dots, \hat{\mathbf{y}}^{<T_y>}$$



Many-to-Many Architecture

- Sequence to Sequence models

$$\mathbf{x} = \mathbf{x}^{<1>}, \mathbf{x}^{<2>}, \mathbf{x}^{<3>}, \dots, \mathbf{x}^{<T_x>} \qquad \hat{\mathbf{y}} = \hat{\mathbf{y}}^{<1>}, \hat{\mathbf{y}}^{<2>}, \hat{\mathbf{y}}^{<3>}, \dots, \hat{\mathbf{y}}^{<T_y>}$$

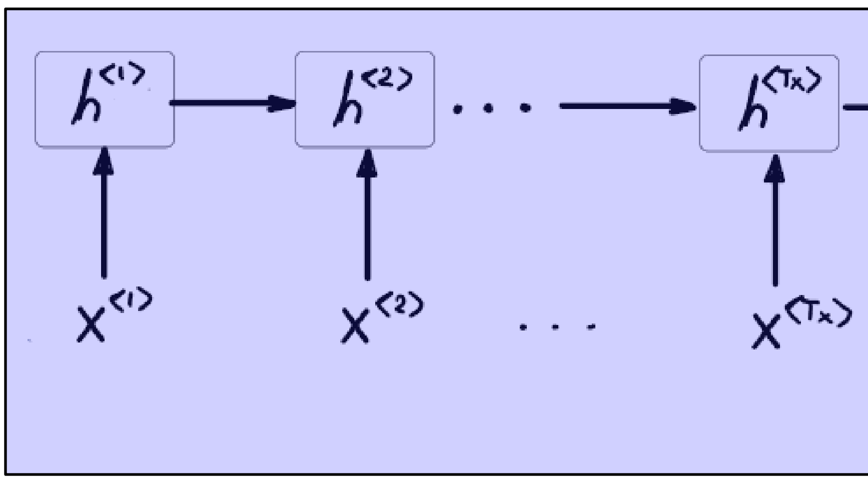


Many-to-Many Architecture

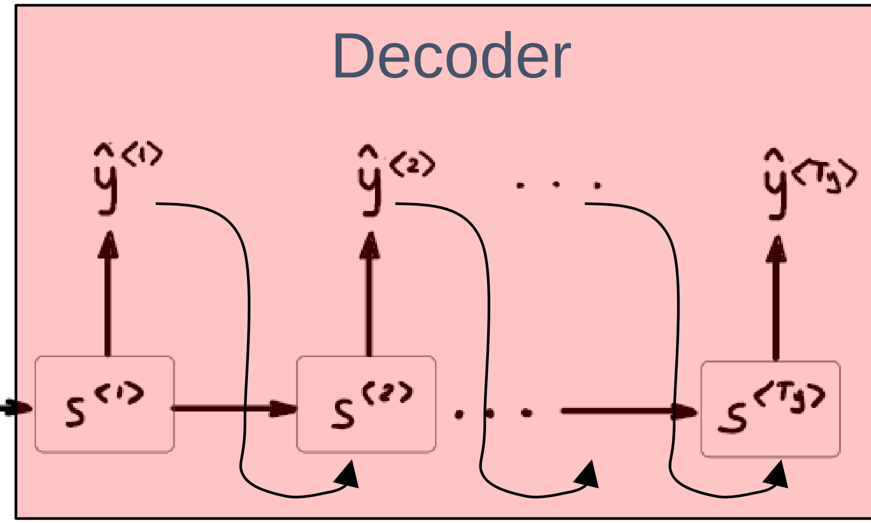
- Sequence to Sequence models

$$\mathbf{x} = \mathbf{x}^{<1>}, \mathbf{x}^{<2>}, \mathbf{x}^{<3>}, \dots, \mathbf{x}^{<T_x>} \qquad \hat{\mathbf{y}} = \hat{\mathbf{y}}^{<1>}, \hat{\mathbf{y}}^{<2>}, \hat{\mathbf{y}}^{<3>}, \dots, \hat{\mathbf{y}}^{<T_y>}$$

All the information about the input sequence is encoded into the internal state of the RNN at time T_x

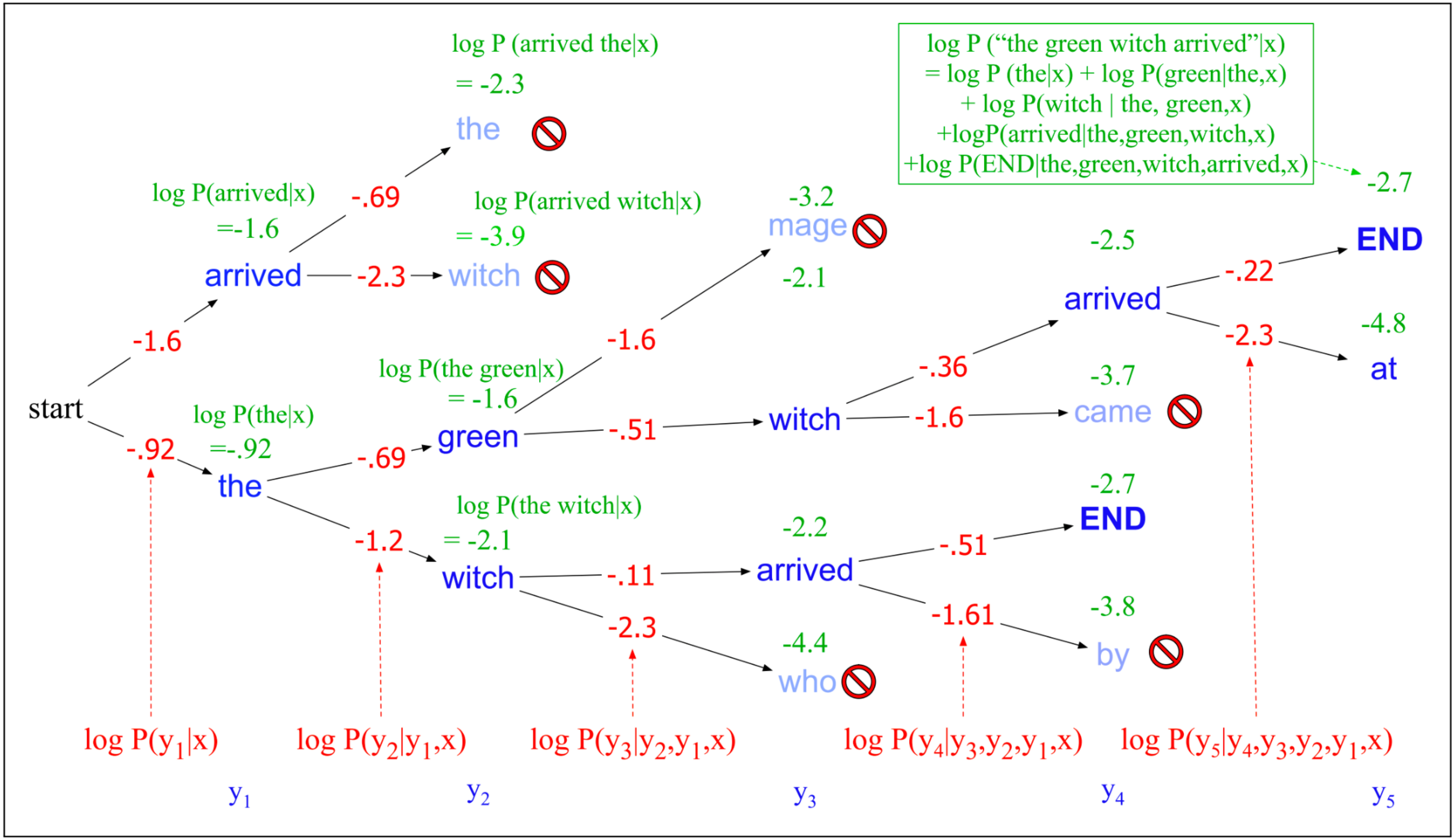


Encoder



- The encoded information is decoded into a different sequence
- The prediction is fed as input for the next time step.
- Various decoding strategies

Beam Search

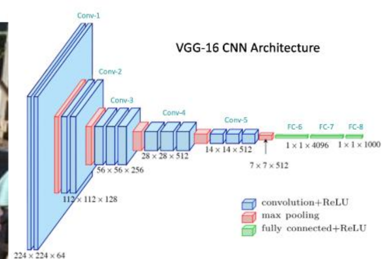


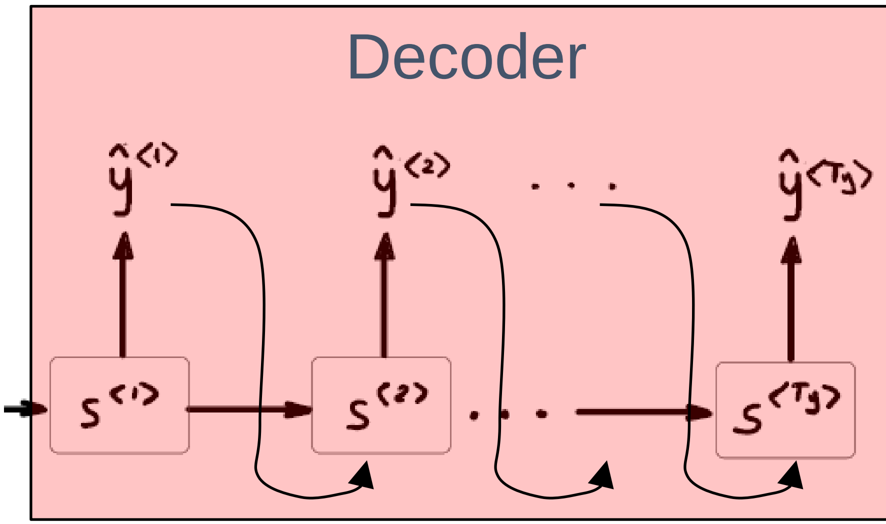
https://localcoder.org/what-does-the-beam-size-represent-in-the-beam-search-algorithm

Image Captioning

- The notion of encoder / decoder can be extended to other domains

We can use a CNN to encode the information contained in an image, and an RNN to decode it (e.g. to text)

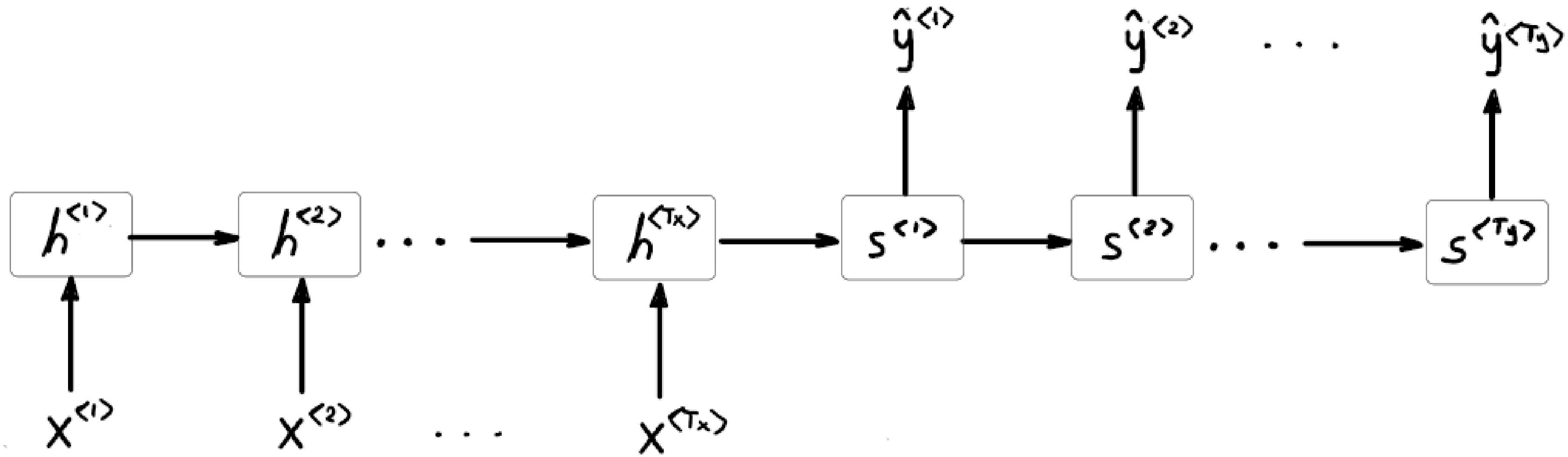




A group of people shopping at an outdoor market.

There are many vegetables at the fruit stand.

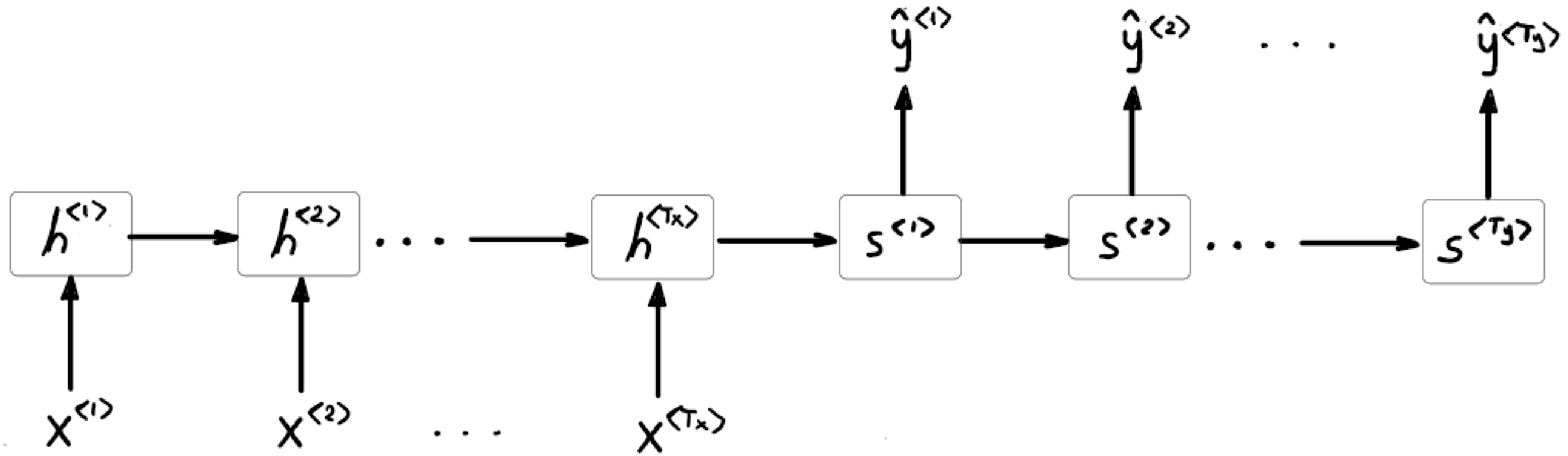
Application Example



Power resides where men believe it resides. It's a trick, a shadow on the wall. And, a very small man can cast a very large shadow.

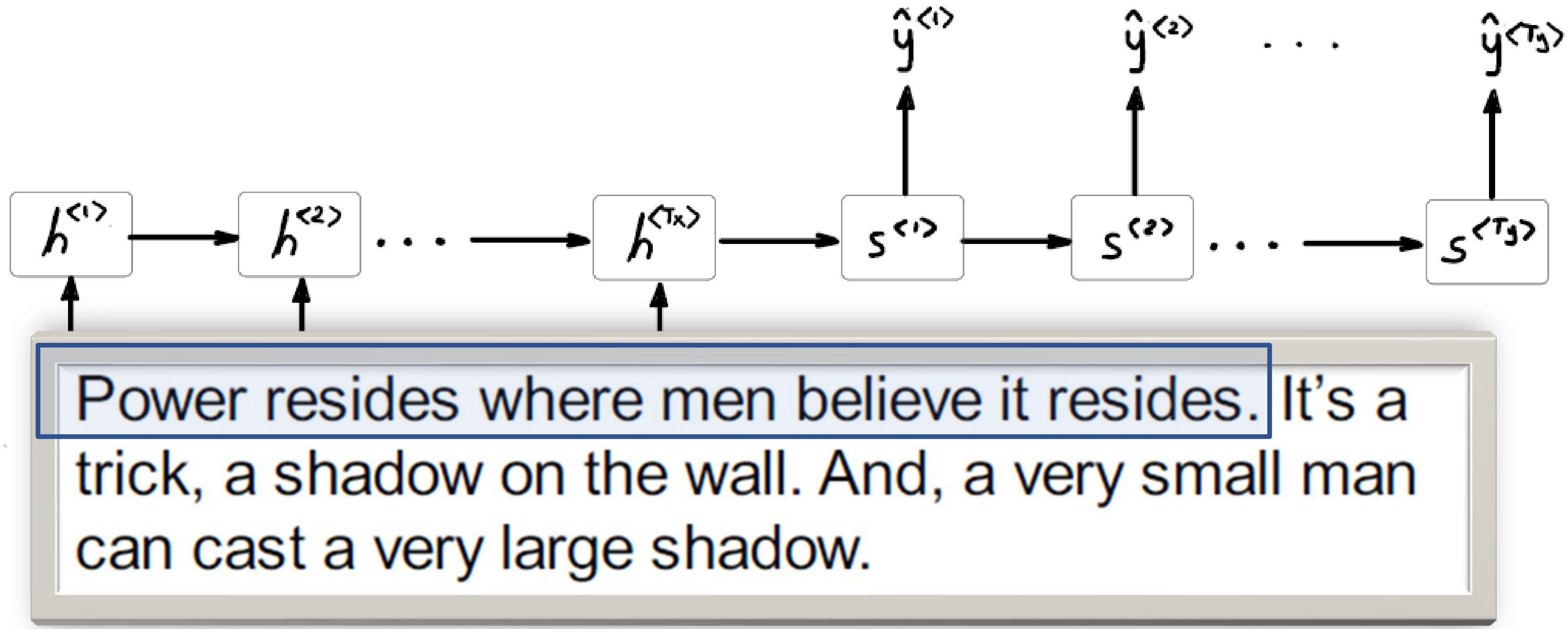
Application Example

Catalan translation?



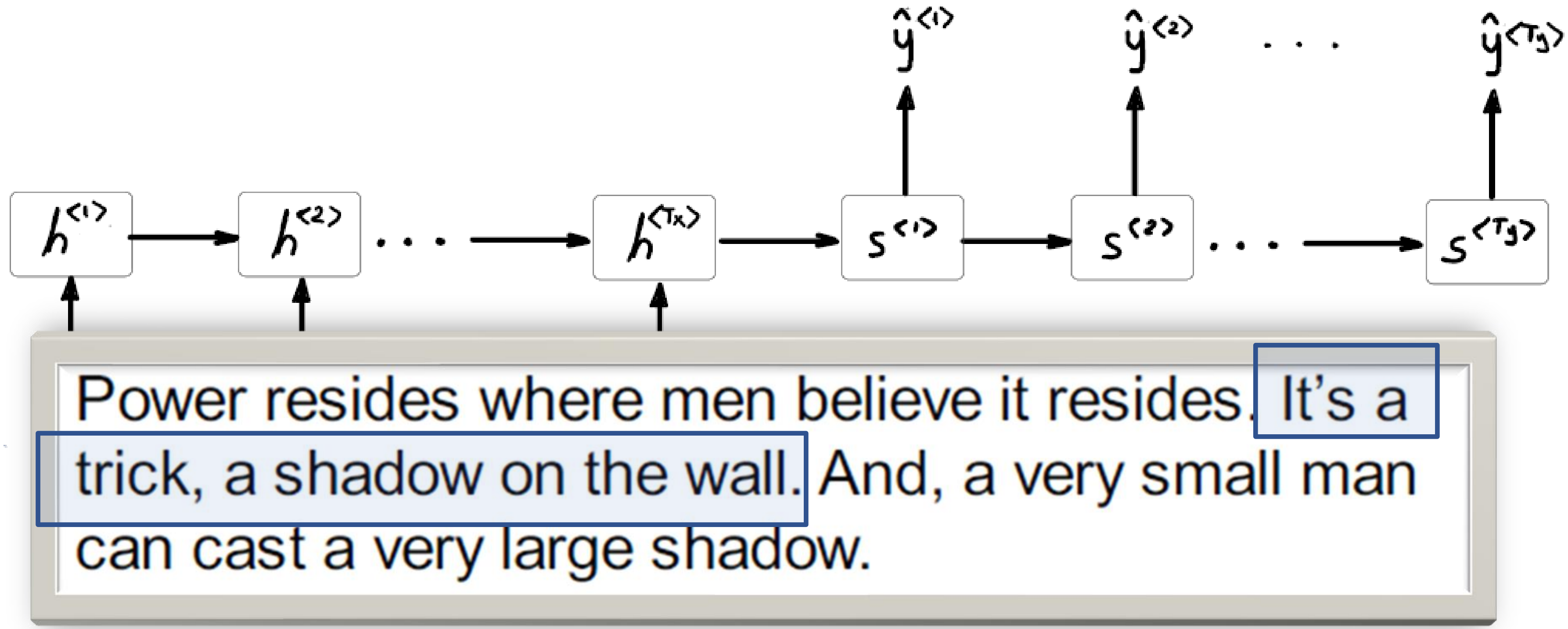
Motivating Attention

- To translate a text, we would usually focus on different parts sequentially



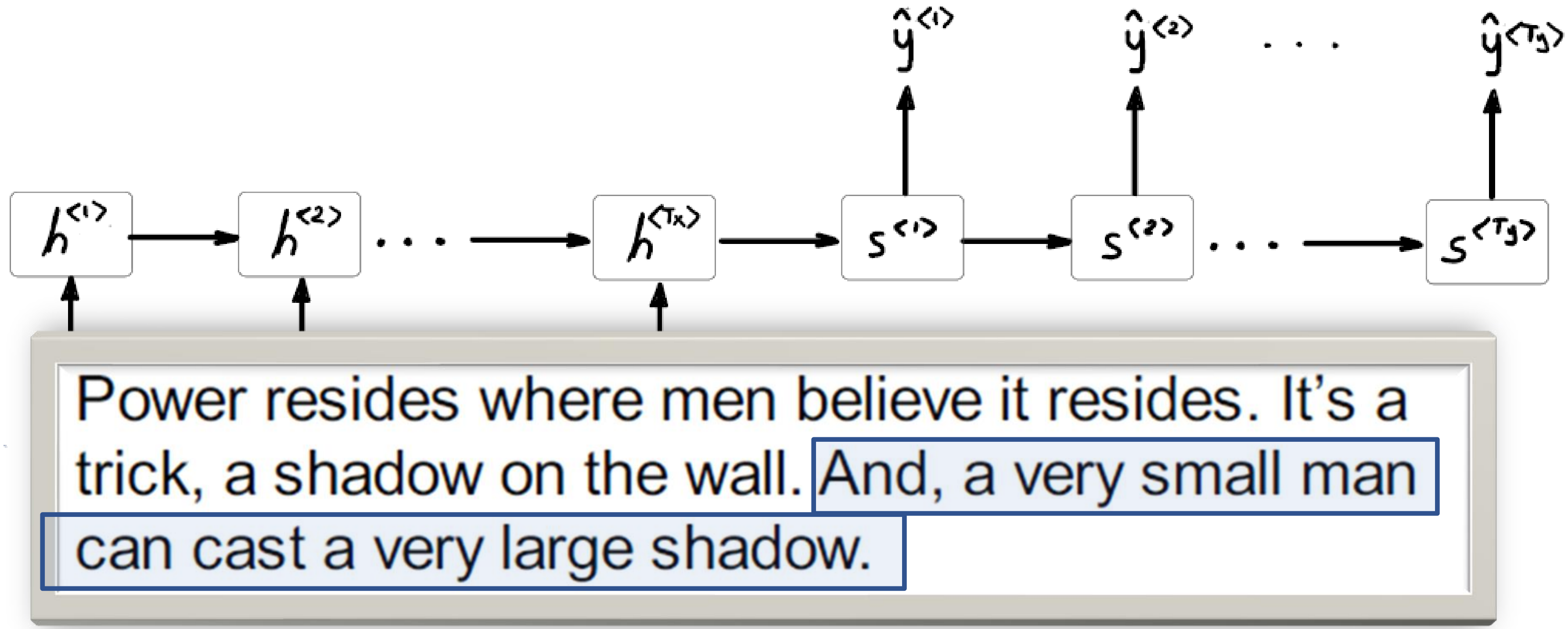
Motivating Attention

- To translate a text, we would usually focus on different parts sequentially



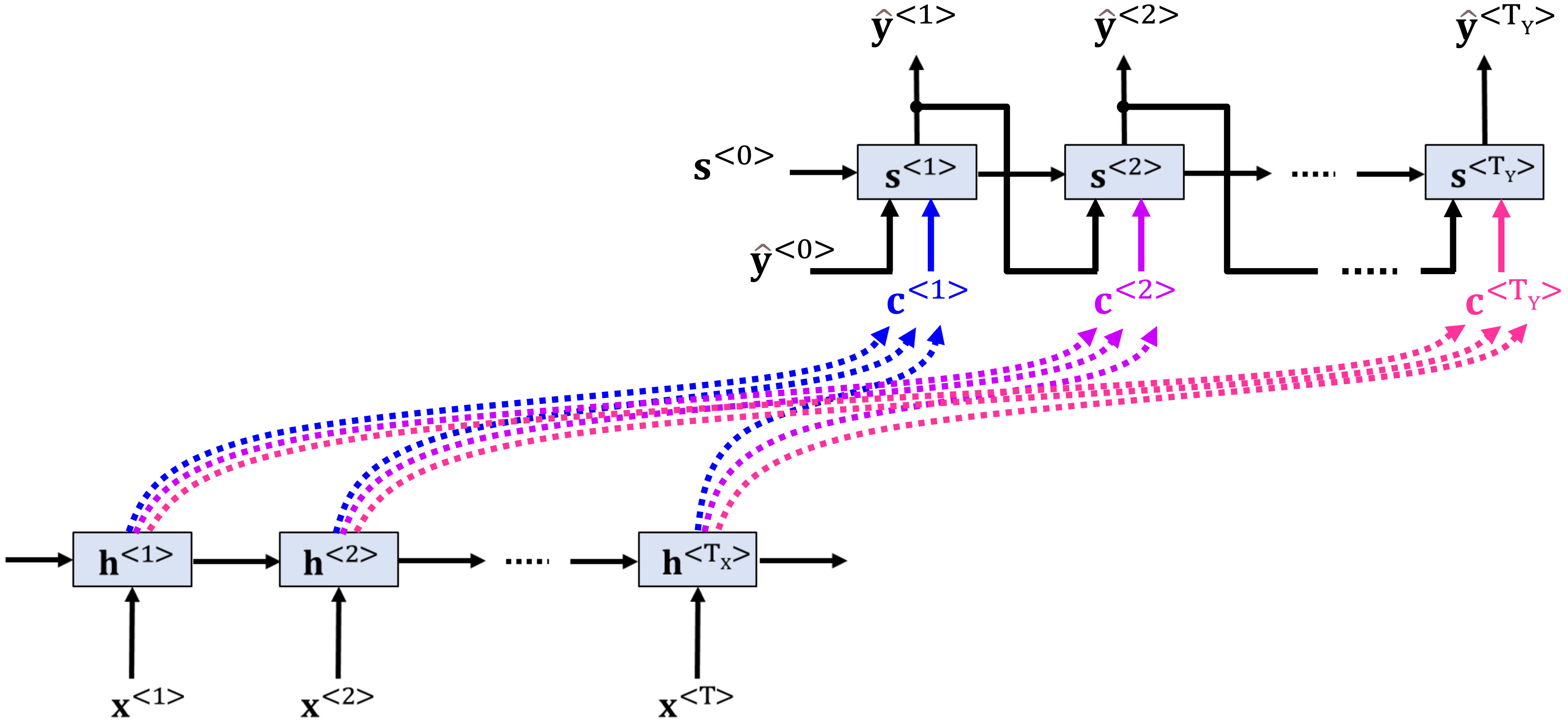
Motivating Attention

- To translate a text, we would usually focus on different parts sequentially



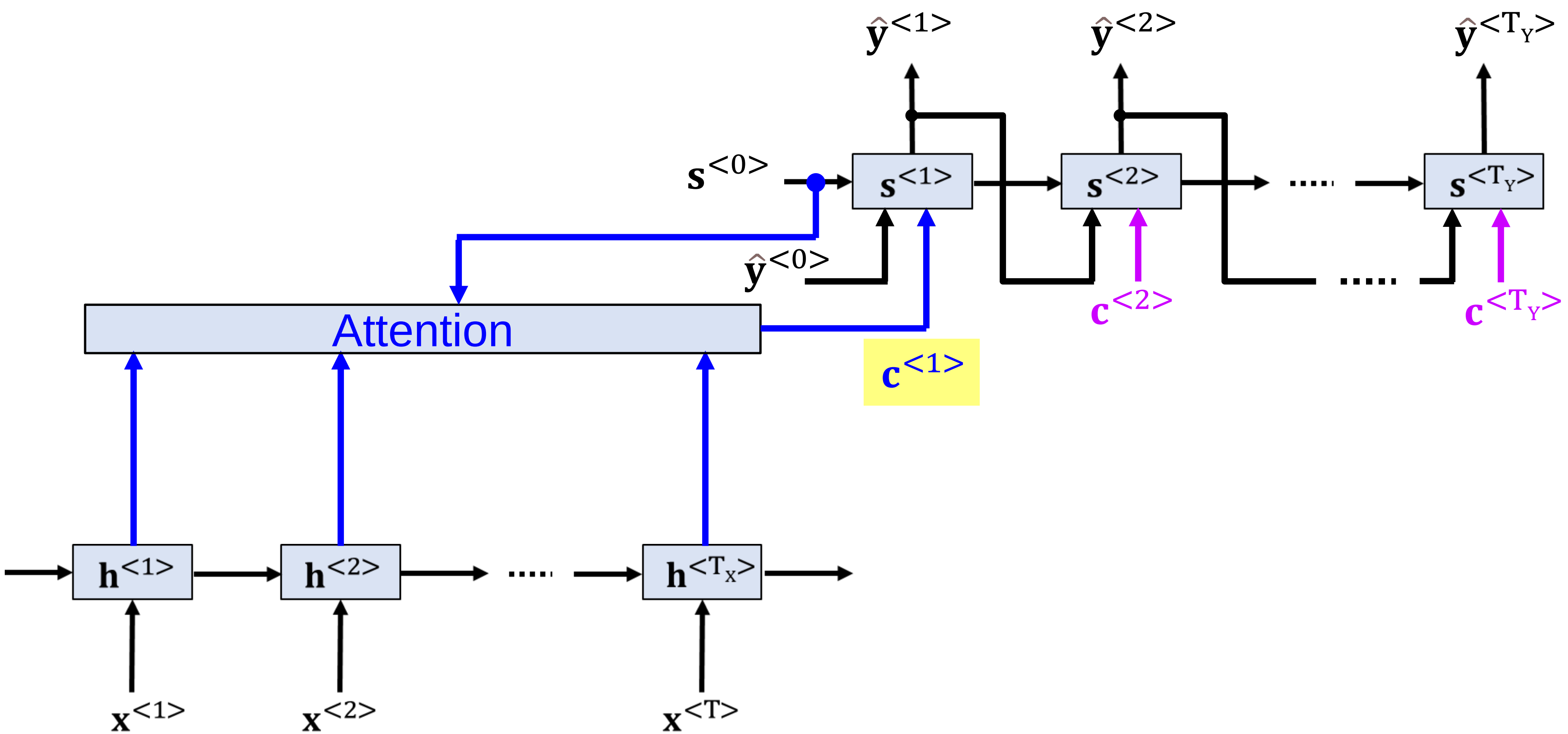
Attention Mechanism

- To translate a text, we would usually focus on different parts sequentially



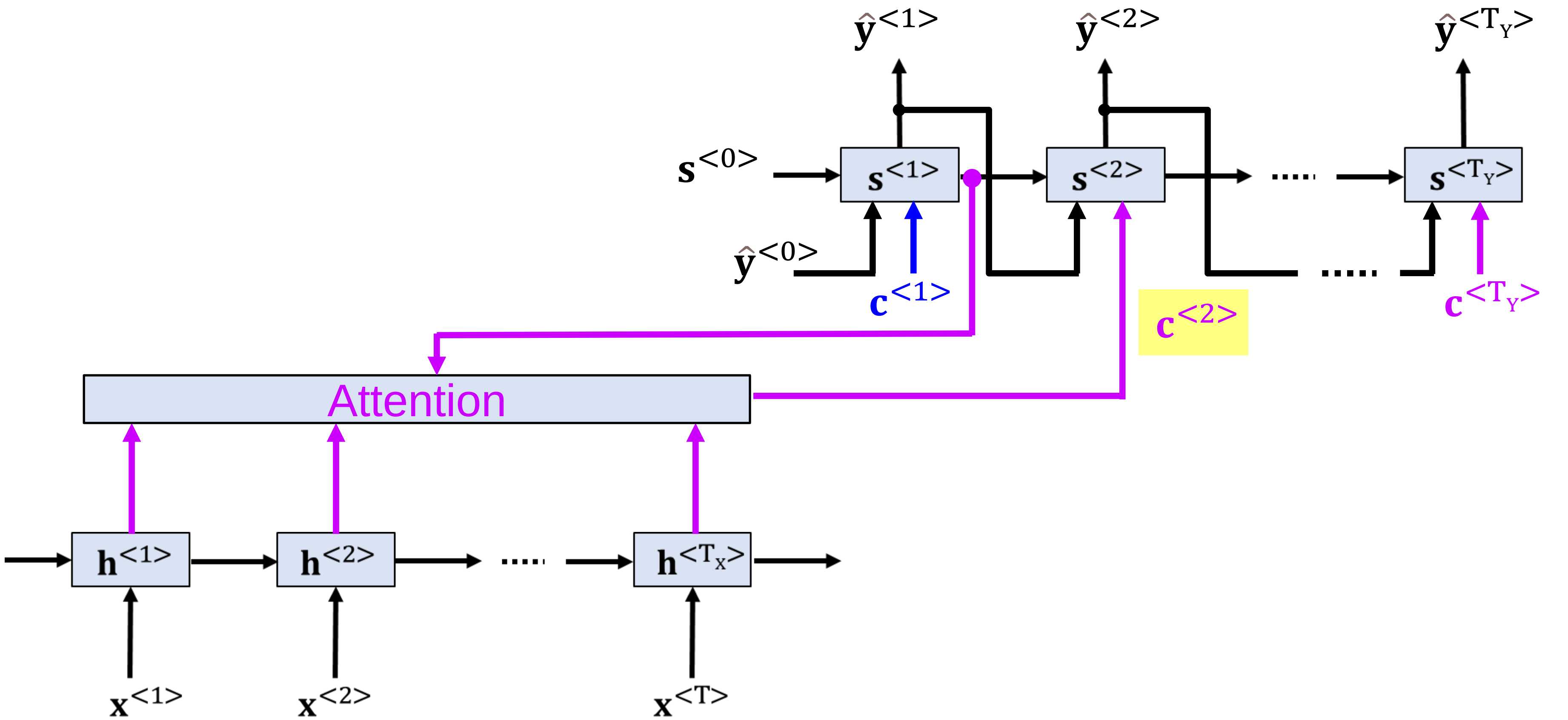
Attention Mechanism

- To translate a text, we would usually focus on different parts sequentially



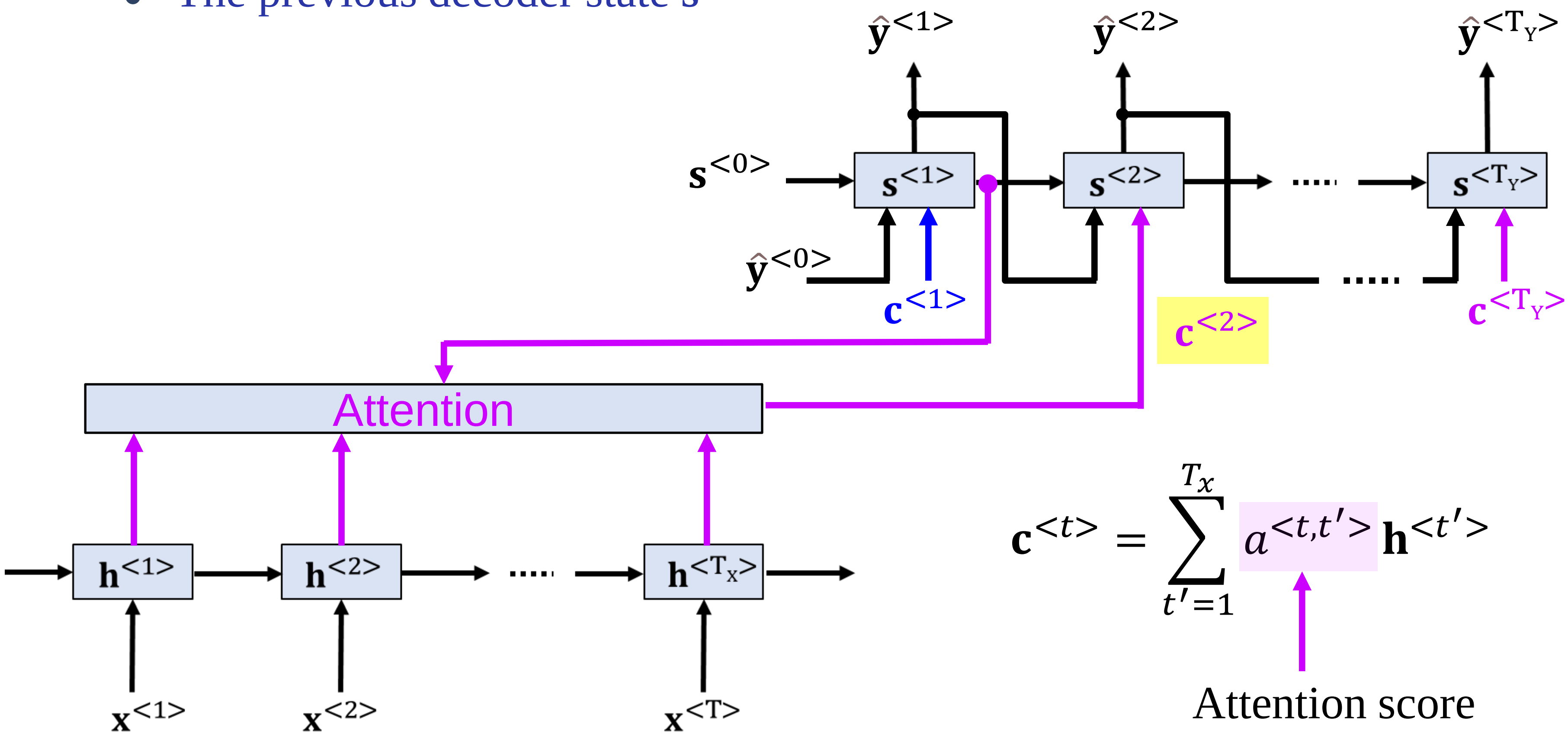
Attention Mechanism

- To translate a text, we would usually focus on different parts sequentially



Attention Mechanism

- For a generic output time “t” we consider
  - All encoder states $\mathbf{h}^{<t'>}$
  - The previous decoder state $\mathbf{s}^{<t-1>}$

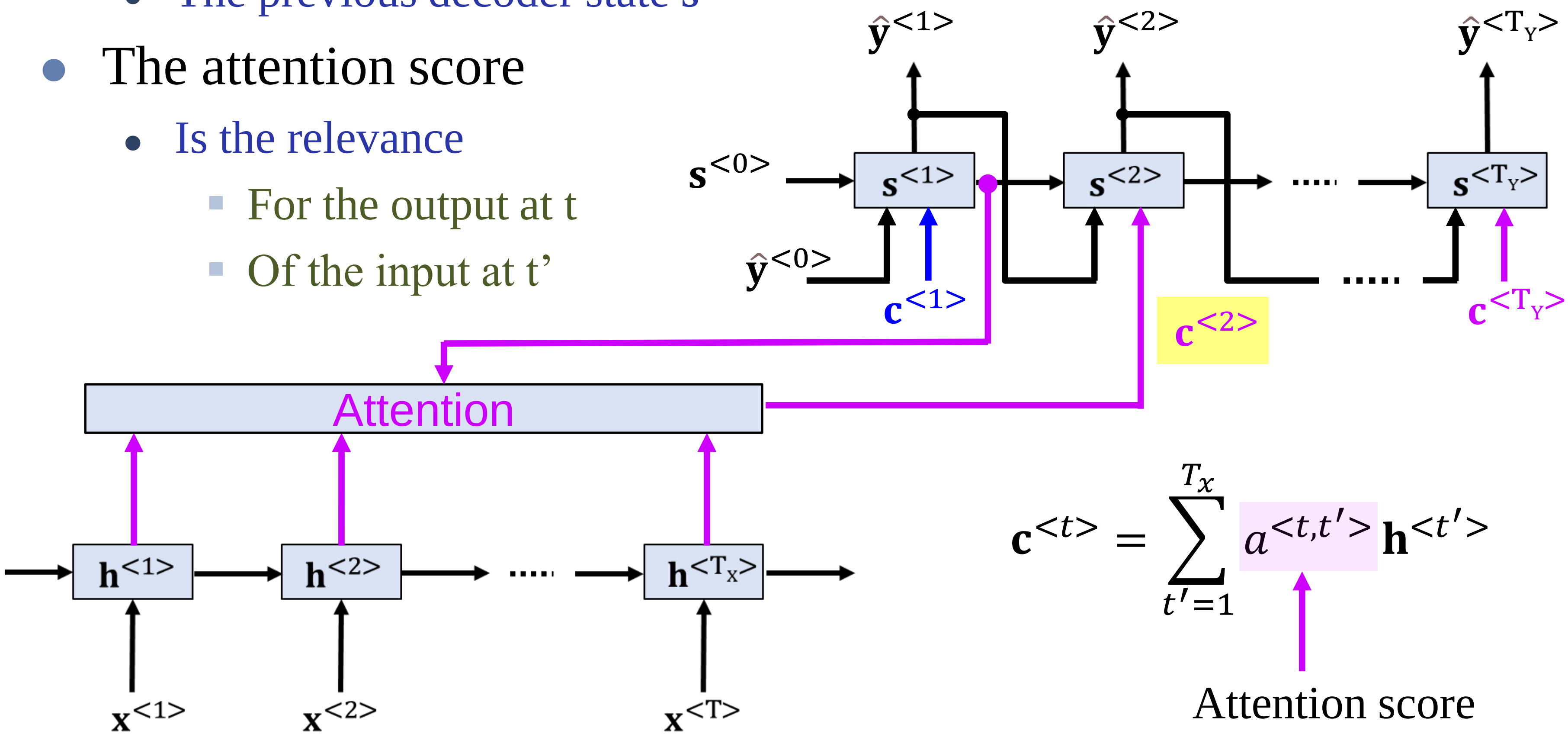


$$\mathbf{c}^{<t>} = \sum_{t'=1}^{T_x} a^{<t,t'>} \mathbf{h}^{<t'>}$$

Attention score

Attention Mechanism

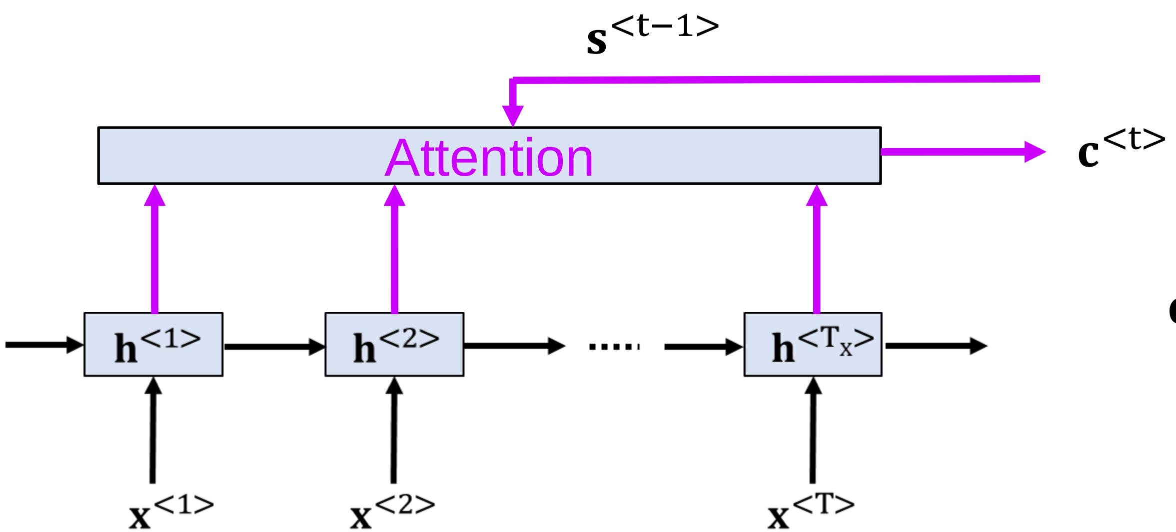
- For a generic output time "t" we consider
  - All encoder states $\mathbf{h}^{<t'>}$
  - The previous decoder state $\mathbf{s}^{<t-1>}$
- The attention score
  - Is the relevance
    - For the output at t
    - Of the input at t'



$$\mathbf{c}^{<t>} = \sum_{t'=1}^{T_x} a^{<t,t'>} \mathbf{h}^{<t'>}$$

Attention score

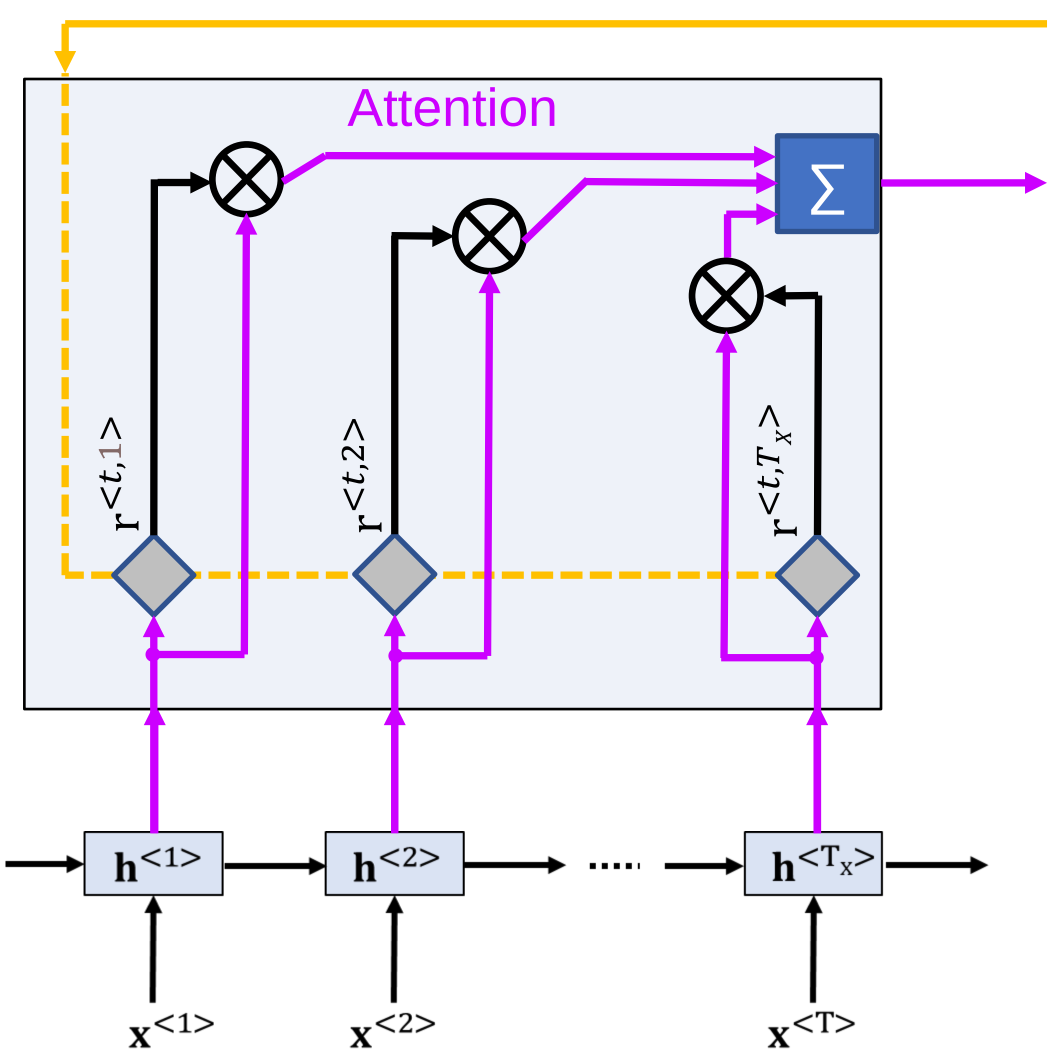
Attention Mechanism



$$\mathbf{c}^{<t>} = \sum_{t'=1}^{T_x} a^{<t,t'>} \mathbf{h}^{<t'>}$$

Attention score

Attention Mechanism

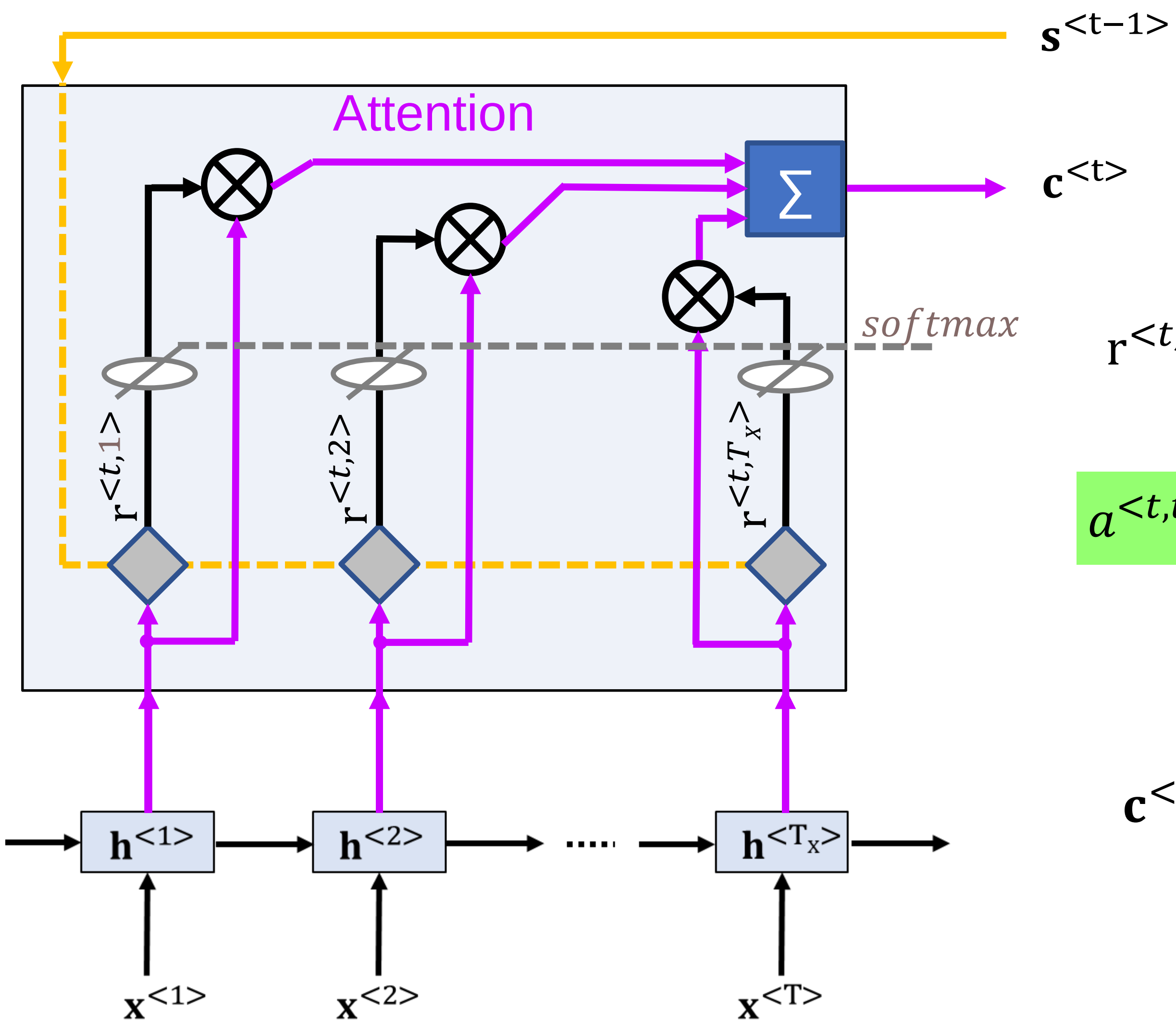


$$r^{<t,t'>} = f\left(\mathbf{s}^{<t-1>}, \mathbf{h}^{<t'>}\right)$$

$$\mathbf{c}^{<t>} = \sum_{t'=1}^{T_x} a^{<t,t'>} \mathbf{h}^{<t'>}$$

Attention score

Attention Mechanism



$$\mathrm{r}^{<t,t'>} = f\left(\mathbf{s}^{<t-1>}, \mathbf{h}^{<t'>}\right)$$

$$a^{<t,t'>} = \frac{exp\left(\mathrm{r}^{<t,t'>}\right)}{\sum_{t'=1}^{T_x} exp\left(r^{<t,t'>}\right)}$$

$$\mathbf{c}^{<t>} = \sum_{t'=1}^{T_x} a^{<t,t'>} \mathbf{h}^{<t'>}$$

Attention score

Key, Query, Value

- The attention score can be understood as some normalized similarity score
  - Similarity score is between and $\mathbf{s}^{<t-1>}$
    - And all input states
  - The resulting scores weight each $\mathbf{h}^{<t'>}$

$$\mathrm{r}^{<t,t'>} = f\left(\mathbf{s}^{<t-1>}, \mathbf{h}^{<t'>}\right)$$

$$a^{<t,t'>} = \frac{exp\left(\mathrm{r}^{<t,t'>}\right)}{\sum_{t'=1}^{T_x} exp\left(r^{<t,t'>}\right)}$$

Key, Query, Value

- The attention score can be understood as some normalized similarity score
  - Similarity score is between and $\mathbf{s}^{<t-1>}$
    - And all input states
  - The resulting scores weight each $\mathbf{h}^{<t'>}$
- Therefore, for a given output-time t
  - Prior state $\mathbf{s}^{<t-1>}$ acts as a **query** to search
  - We search over all input states $\mathbf{h}^{<t'>}$
    - Here, the states themselves are the **key**
  - The value retrieved by the search are, again, the states $\mathbf{h}^{<t'>}$

$$\mathrm{r}^{<t,t'>} = f\left(\mathbf{s}^{<t-1>}, \mathbf{h}^{<t'>}\right)$$

$$a^{<t,t'>} = \frac{exp\left(\mathrm{r}^{<t,t'>}\right)}{\sum_{t'=1}^{T_x} exp\left(r^{<t,t'>}\right)}$$

Key, Query, Value

- The attention score can be understood as some normalized similarity score
  - Similarity score is between and $\mathbf{s}^{<t-1>}$
    - And all input states
  - The resulting scores weight each $\mathbf{h}^{<t'>}$
- Therefore, for a given output-time t
  - Prior state $\mathbf{s}^{<t-1>}$ acts as a **query** to search
  - We search over all input states $\mathbf{h}^{<t'>}$
    - Here, the states themselves are the **key**
  - The value retrieved by the search are, again, the states $\mathbf{h}^{<t'>}$
- The (key, query, value) analogy
  - Borrows from database search
  - But we do not look fot a unique result
    - We want a similarity weight

$$\mathrm{r}^{<t,t'>} = f\left(\mathbf{s}^{<t-1>}, \mathbf{h}^{<t'>}\right)$$

$$a^{<t,t'>} = \frac{exp\left(\mathrm{r}^{<t,t'>}\right)}{\sum_{t'=1}^{T_x} exp\left(r^{<t,t'>}\right)}$$

Similarity Functions

- We still haven't specified

$$r^{<t,t'>} = f\left(\mathbf{s}^{<t-1>}, \mathbf{h}^{<t'>}\right)$$

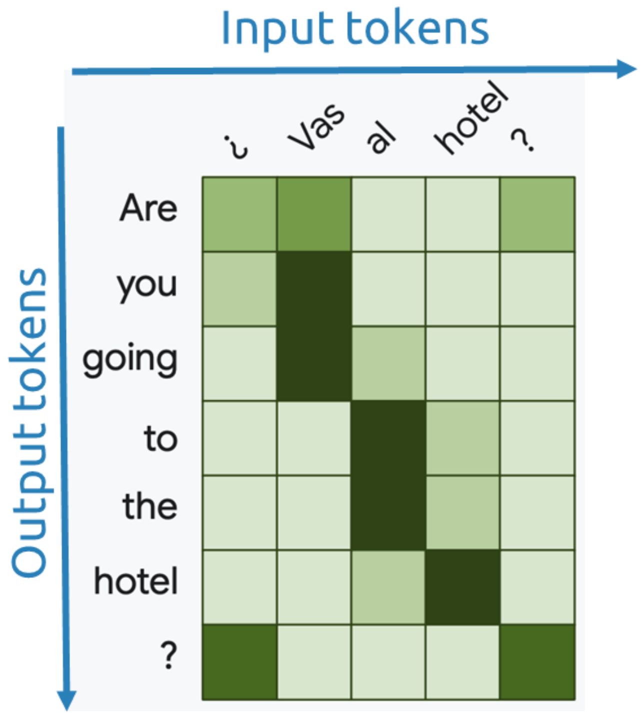
- Two main types of similarity functions
  - Additive

$$r^{<t,t'>} = \mathbf{W}_A \begin{bmatrix} \mathbf{s}^{<t-1>} \\ \mathbf{h}^{<t'>} \end{bmatrix}$$

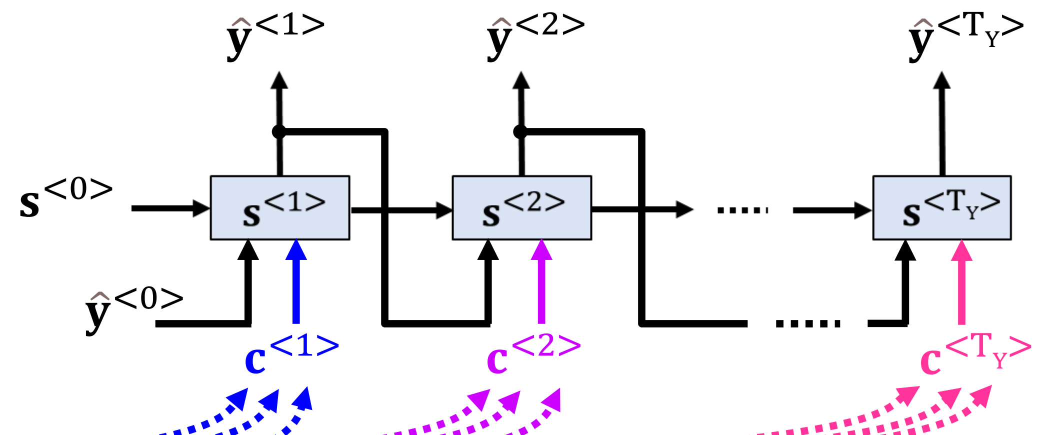
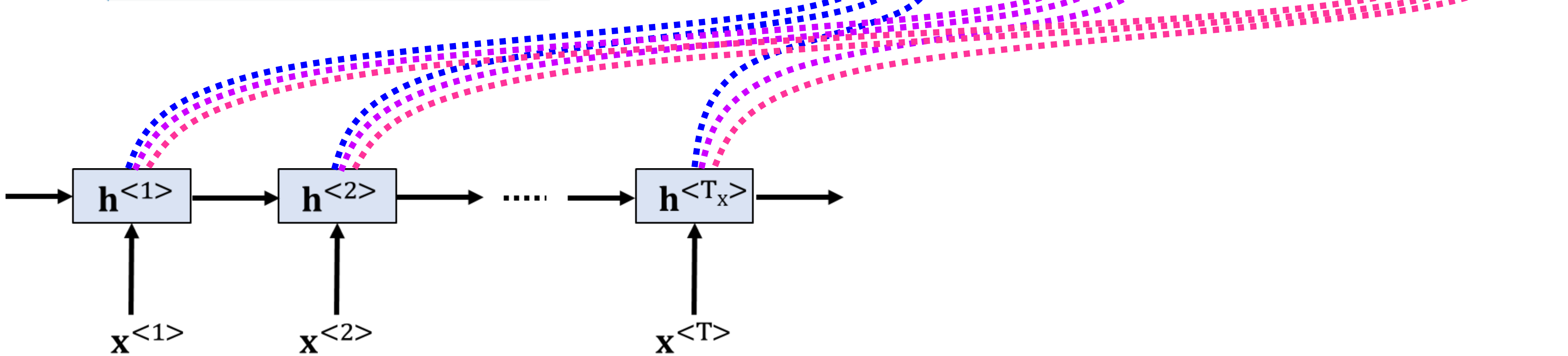
  - Dot-product

$$r^{<t,t'>} = \left(\mathbf{s}^{<t-1>}\right)^T \mathbf{W}_A \, \mathbf{h}^{<t'>}$$

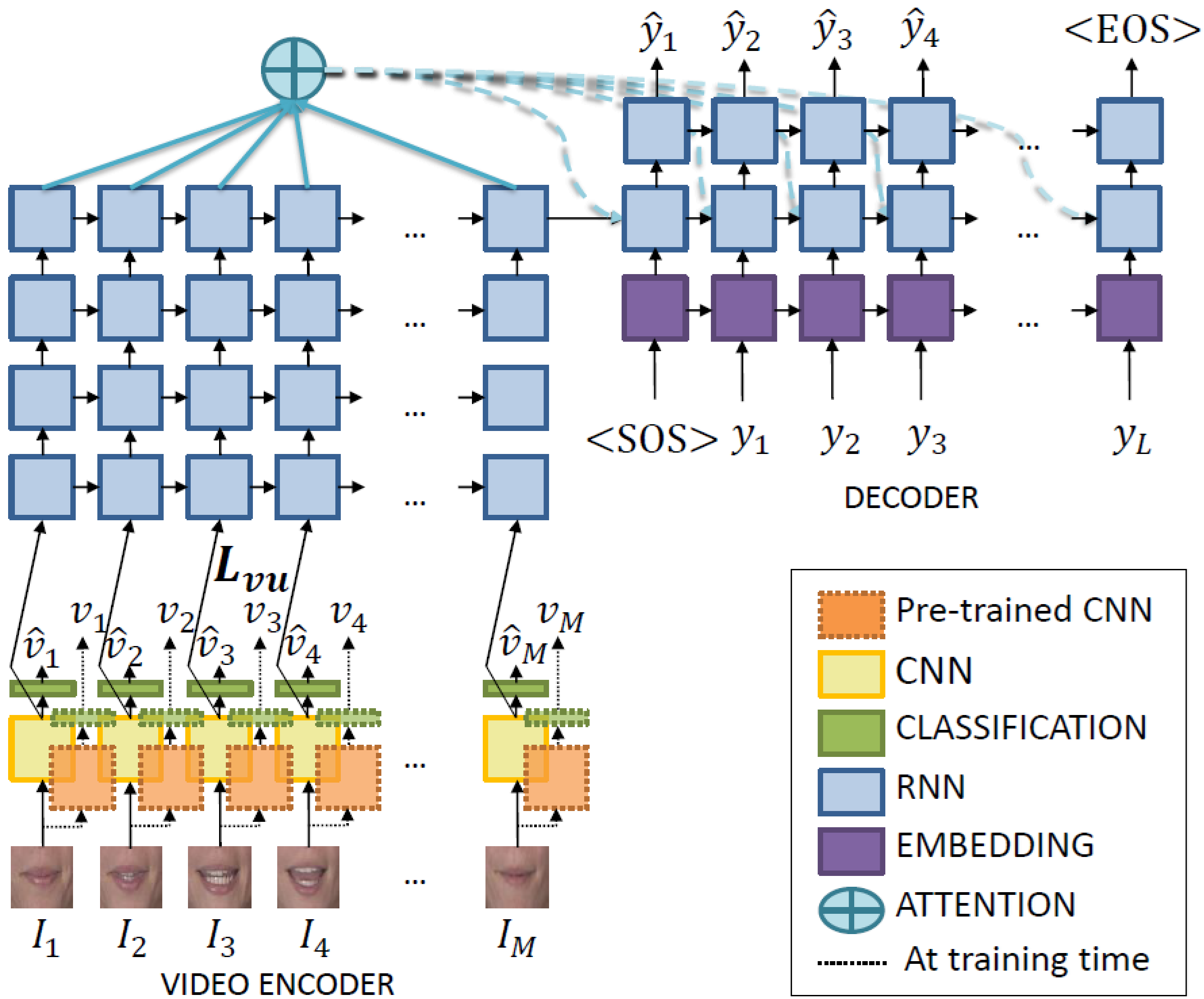
Attention Alignment



The attention matrix provides a visual representation of which input tokens are attended to produce each output token.

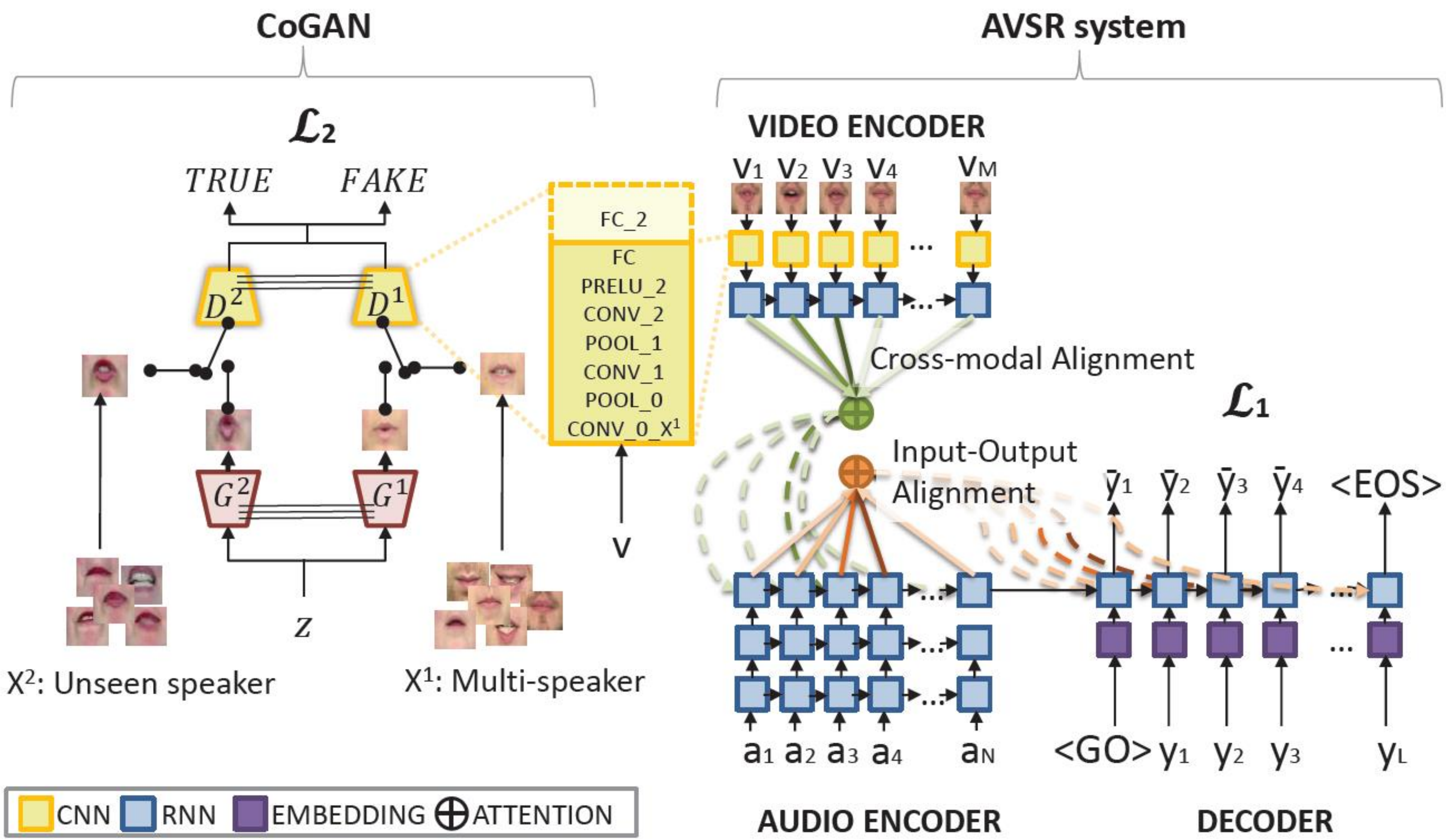



Example: A lip-reading system



Fernanderz-Lopez, et al. (2022) End-to-End Lip-Reading Without Large-Scale Data, IEEE/ACM T Audio, Speech, and Language Proc

Example II: AVSR System



Fernanderz-Lopez, et al. (2020) Cogans For Unsupervised Visual Speech Adaptation to New Speakers, ICASSP

Attention example in the visual domains

Attention for image captioning

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Xu, Kelvin, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention." ICML 2015