Principles of Deep Learning II

Itthi Chatnuntawech
Nanoinformatics and Artificial Intelligence (NAI)

November 21, 2022

Outline

- Recurrent Neural Network (RNN)
- Sequence-to-Sequence Model
  - Attention Mechanism
- Self-Attention and Transformer
- Graph Neural Network

Recurrent Neural Network (RNN)

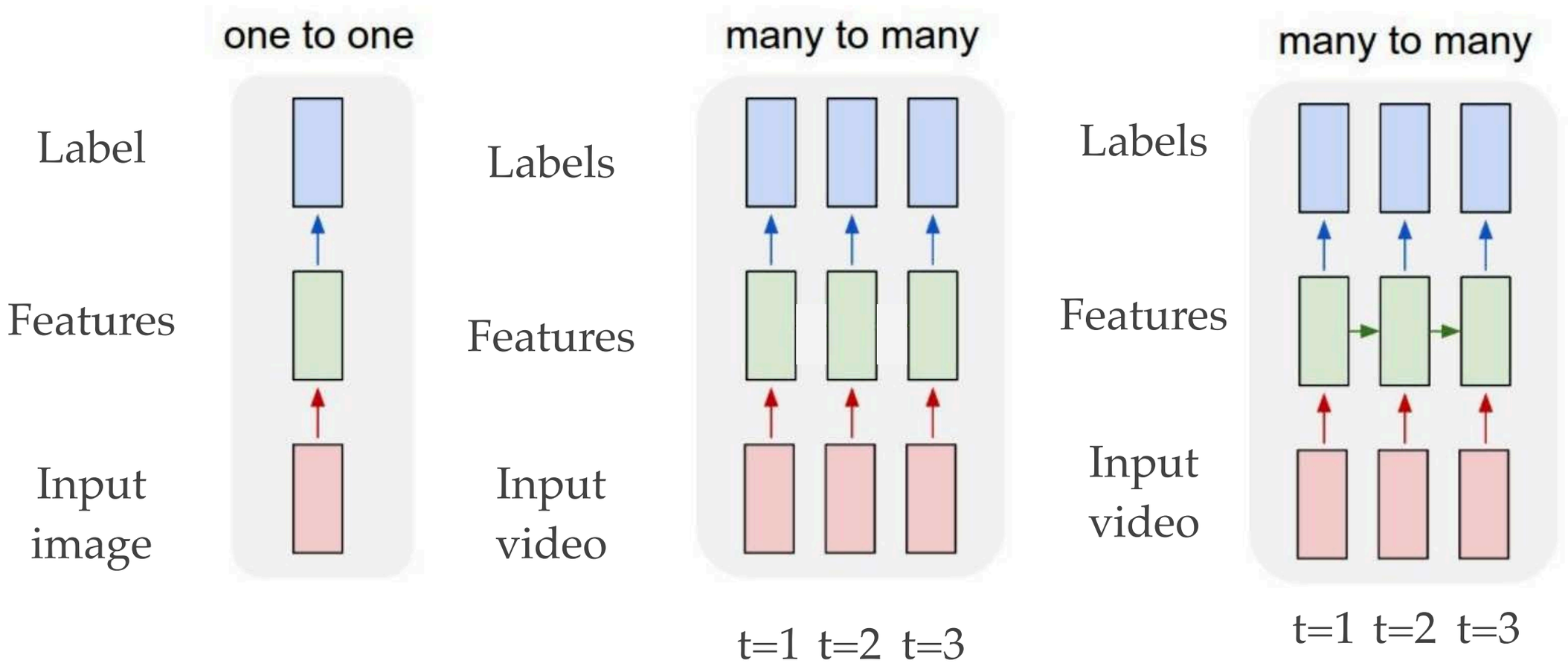


Image classification

Frame-by-frame video classification

Video classification

Recurrent Neural Network (RNN)

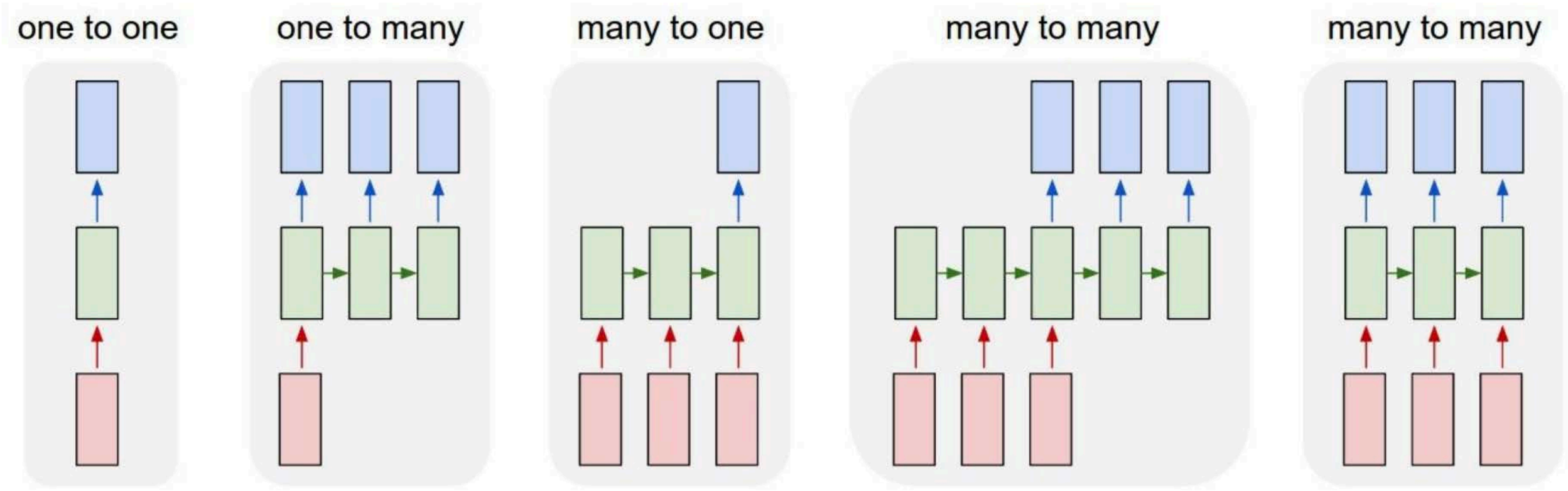


Image classification

Image captioning

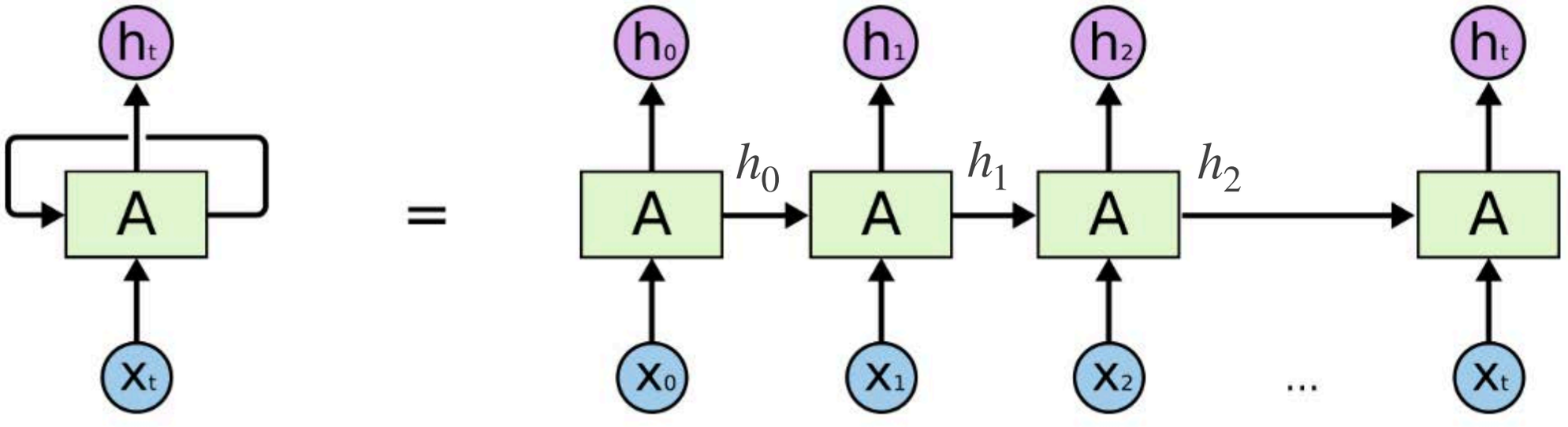
Video classification

Machine translation

Frame-by-frame video classification

From here on out, the goal is to understand the big picture through the pictorial explanation. Do not worry too much about the equations.

Recurrent Neural Network (RNN)



$$h_t = A(h_{t-1}, x_t)$$

New state — Old state — Current input

Summary of $x_0, x_1, \ldots, x_t$

Same weights over time

$$h_t = tanh(W_h h_{t-1} + W_x x_t)$$

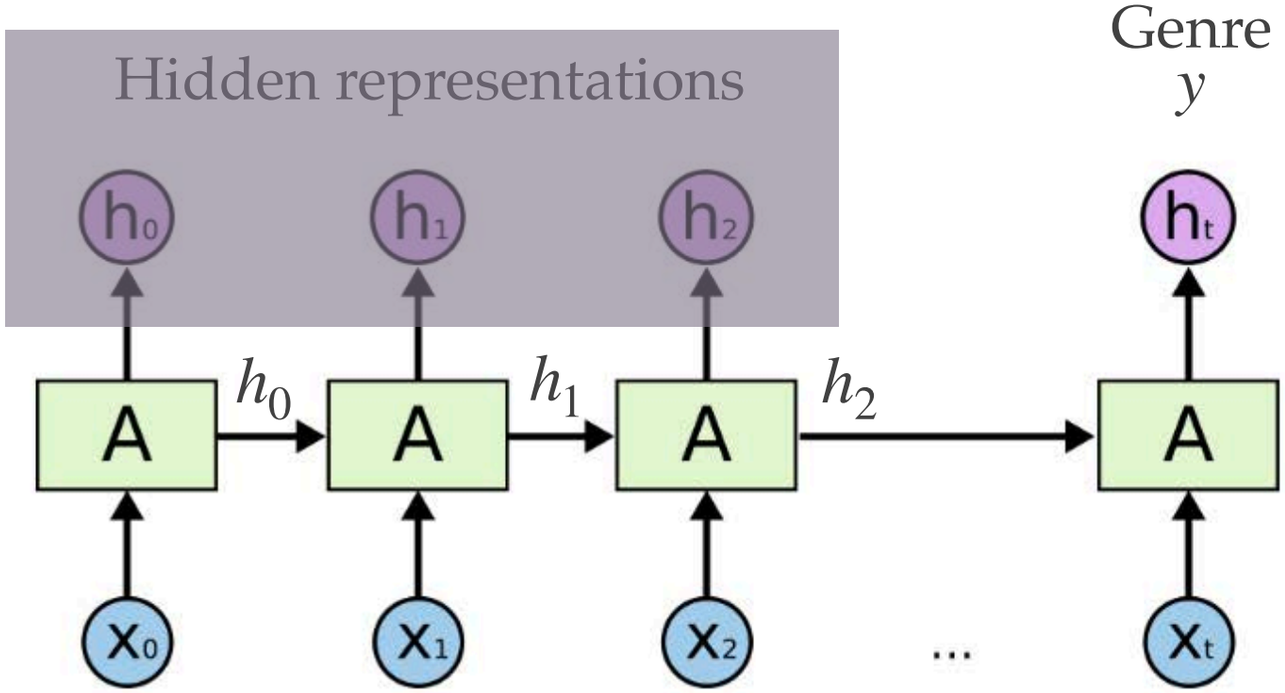
$t = 0 \quad h_0 = tanh(W_x x_0)$

$t = 1 \quad h_1 = tanh(W_h h_0 + W_x x_1)$

$t = 2 \quad h_2 = tanh(W_h h_1 + W_x x_2)$

Recurrent Neural Network (RNN)

Task: Video classification (e.g., genre prediction)



$t = 0$ $h_0 = A(x_0)$

$t = 1$ $h_1 = A(h_0, x_1)$

$t = 2$ $h_2 = A(h_1, x_2)$

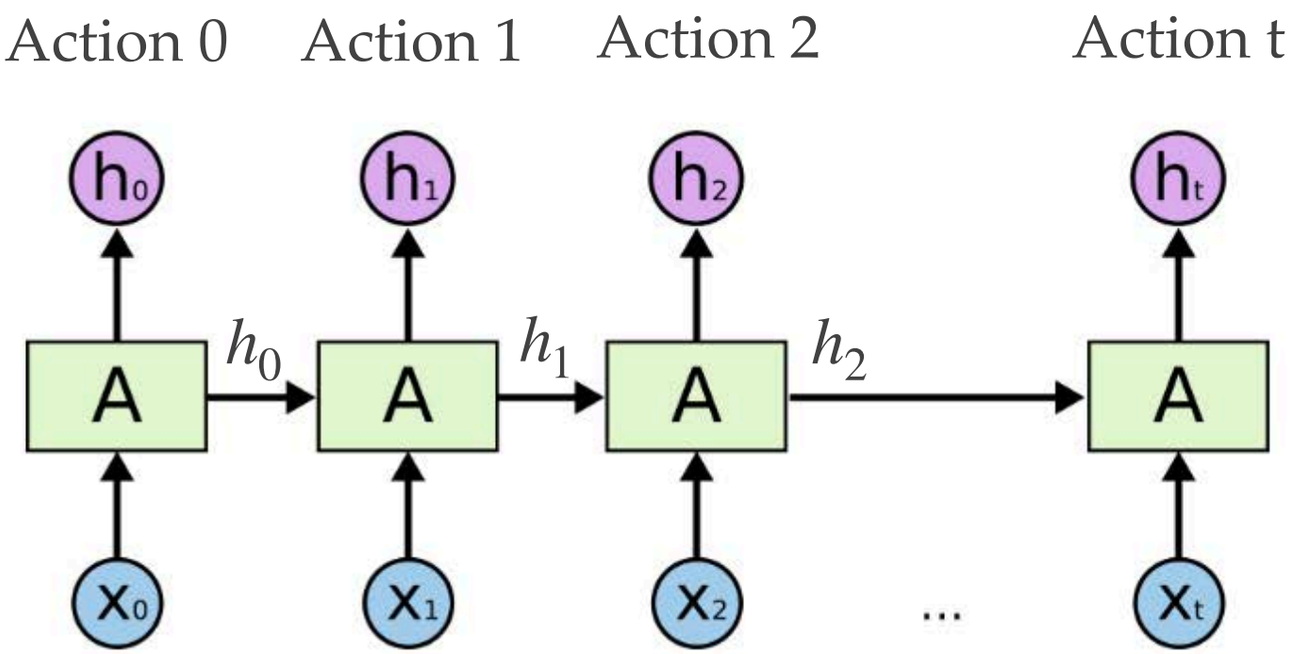
For example,

$y = B(h_t)$

Understanding LSTM Networks by Christopher Colah

Recurrent Neural Network (RNN)

Task: Frame-by-frame video classification (e.g., action prediction)



frame 0 frame 1 frame 2 Last frame

$t = 0$ $h_0 = A(x_0)$ $y_0 = B(h_0)$

$t = 1$ $h_1 = A(h_0, x_1)$ $y_1 = B(h_1)$

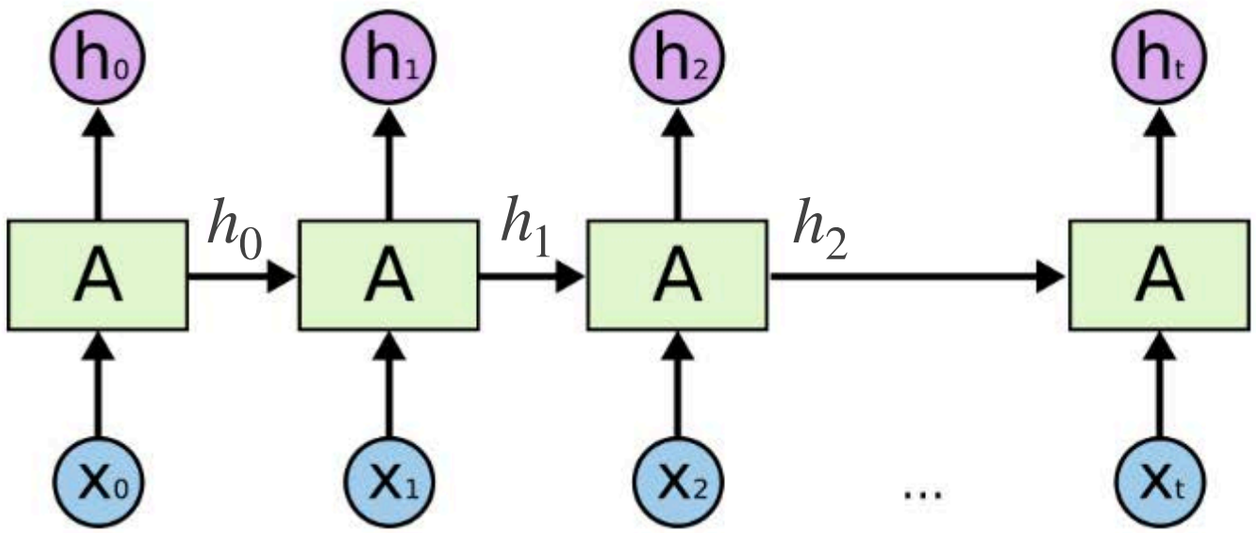
$t = 2$ $h_2 = A(h_1, x_2)$ $\vdots$

$y_t = B(h_t)$

Understanding LSTM Networks by Christopher Colah

Recurrent Neural Network (RNN)

Task: Image captioning (i.e., generate a sentence describing an input image)

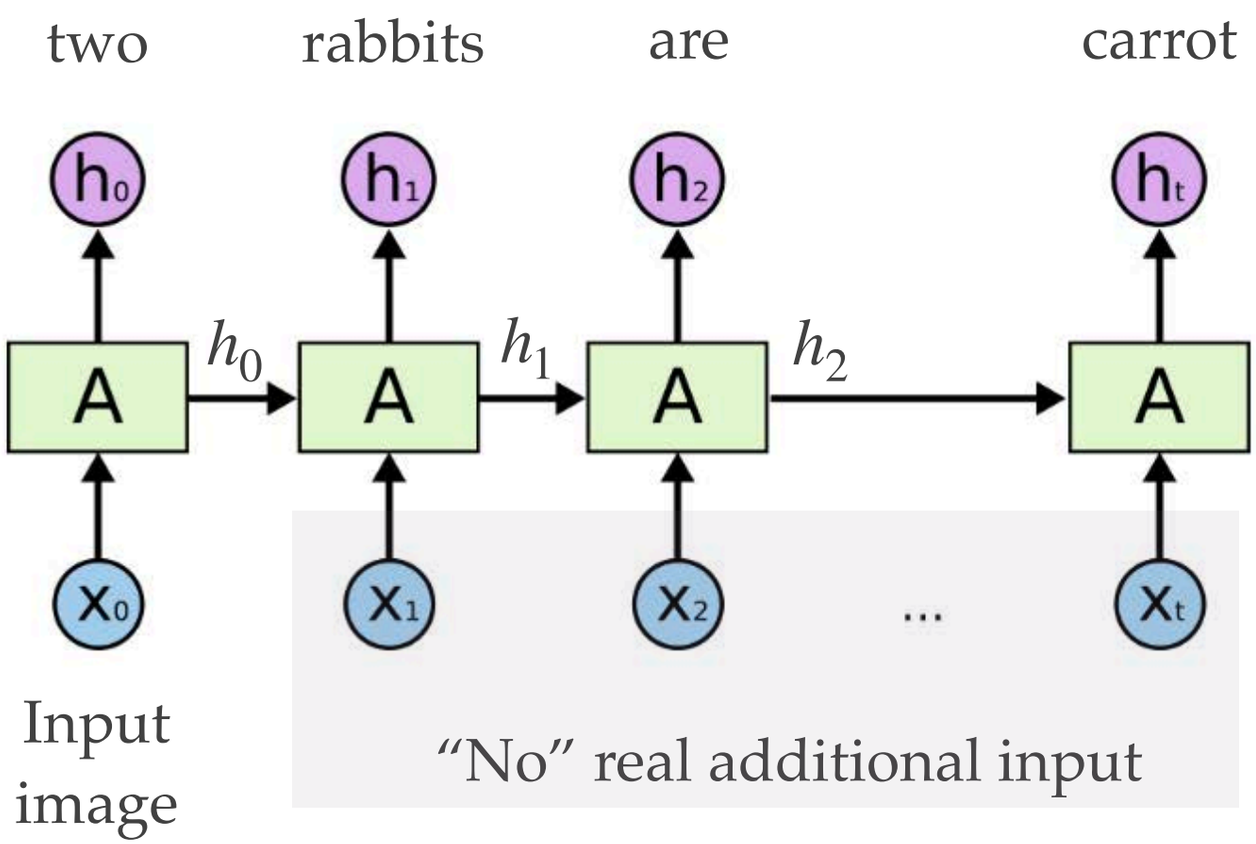


Two rabbits are eating a carrot

Understanding LSTM Networks by Christopher Colah

Recurrent Neural Network (RNN)

Task: Image captioning (i.e., generate a sentence describing an input image)

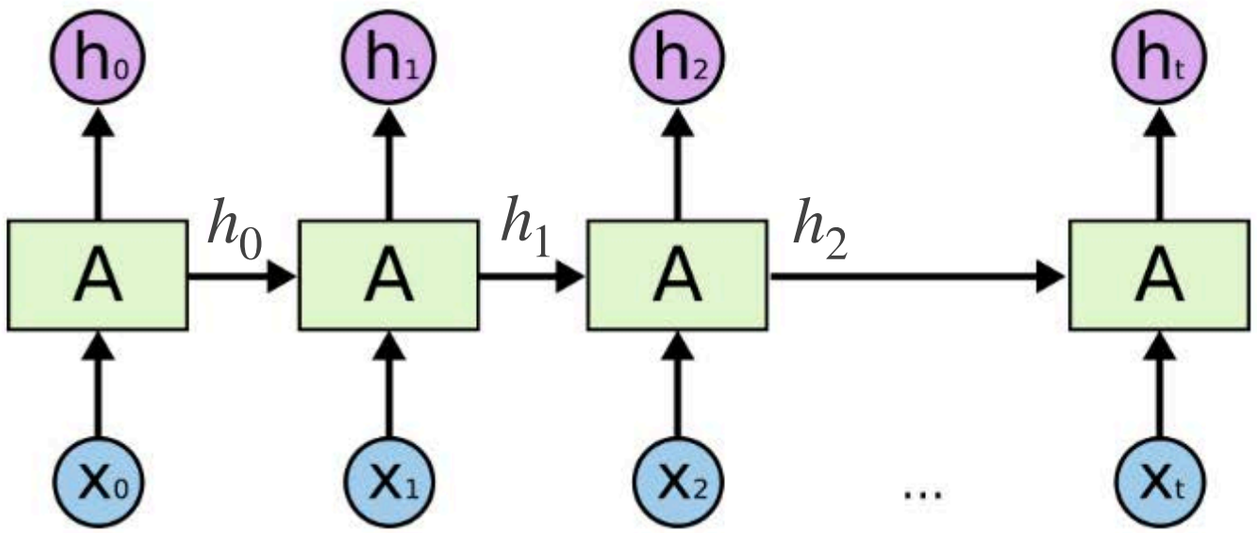


Two rabbits are eating a carrot

Understanding LSTM Networks by Christopher Colah

Recurrent Neural Network (RNN)

Task: Machine translation (e.g., translate language A to language B)

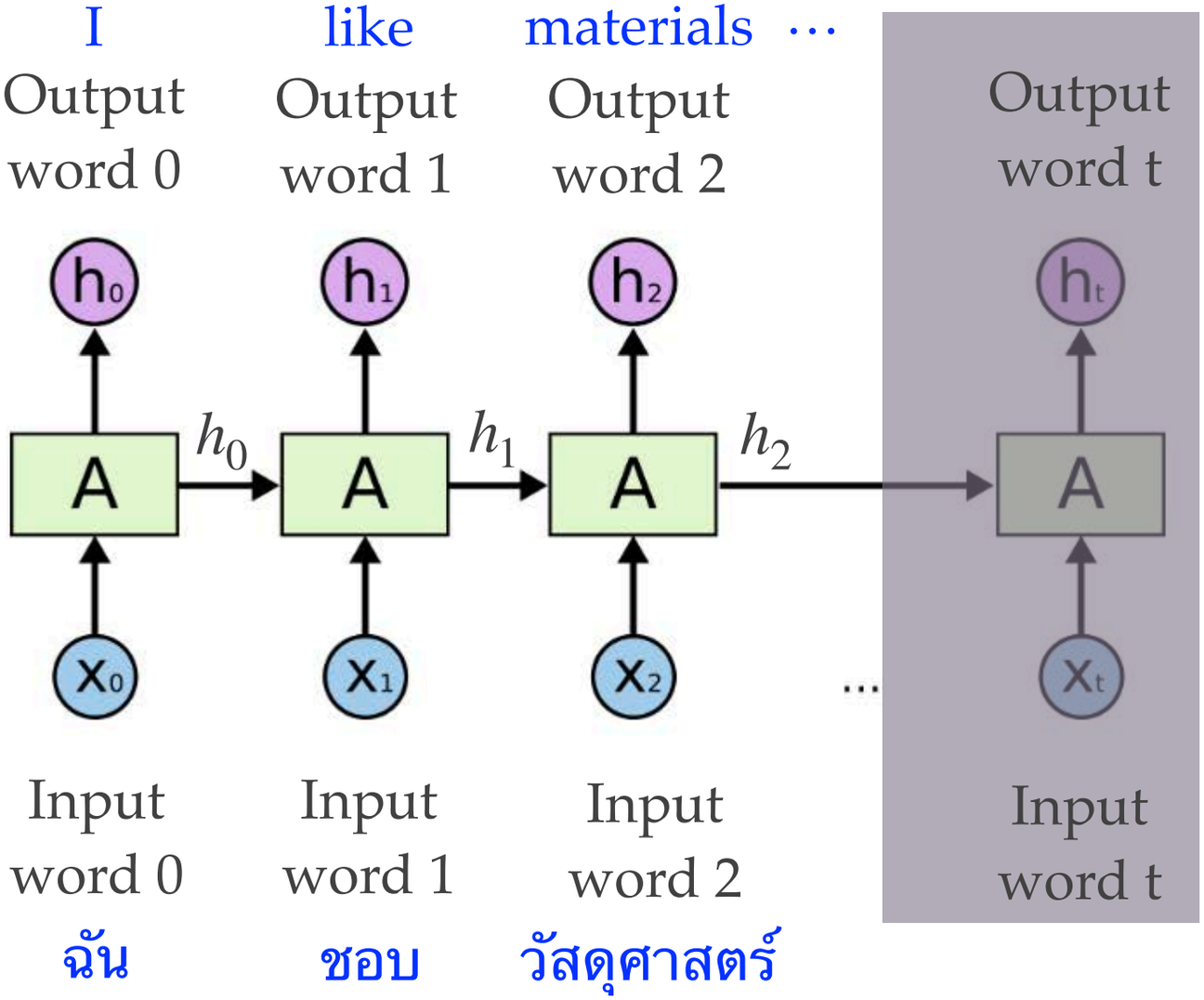


ฉันชอบวัสดุศาสตร์ → I like materials science

Understanding LSTM Networks by Christopher Colah

Recurrent Neural Network (RNN)

Task: Machine translation (e.g., translate language A to language B)

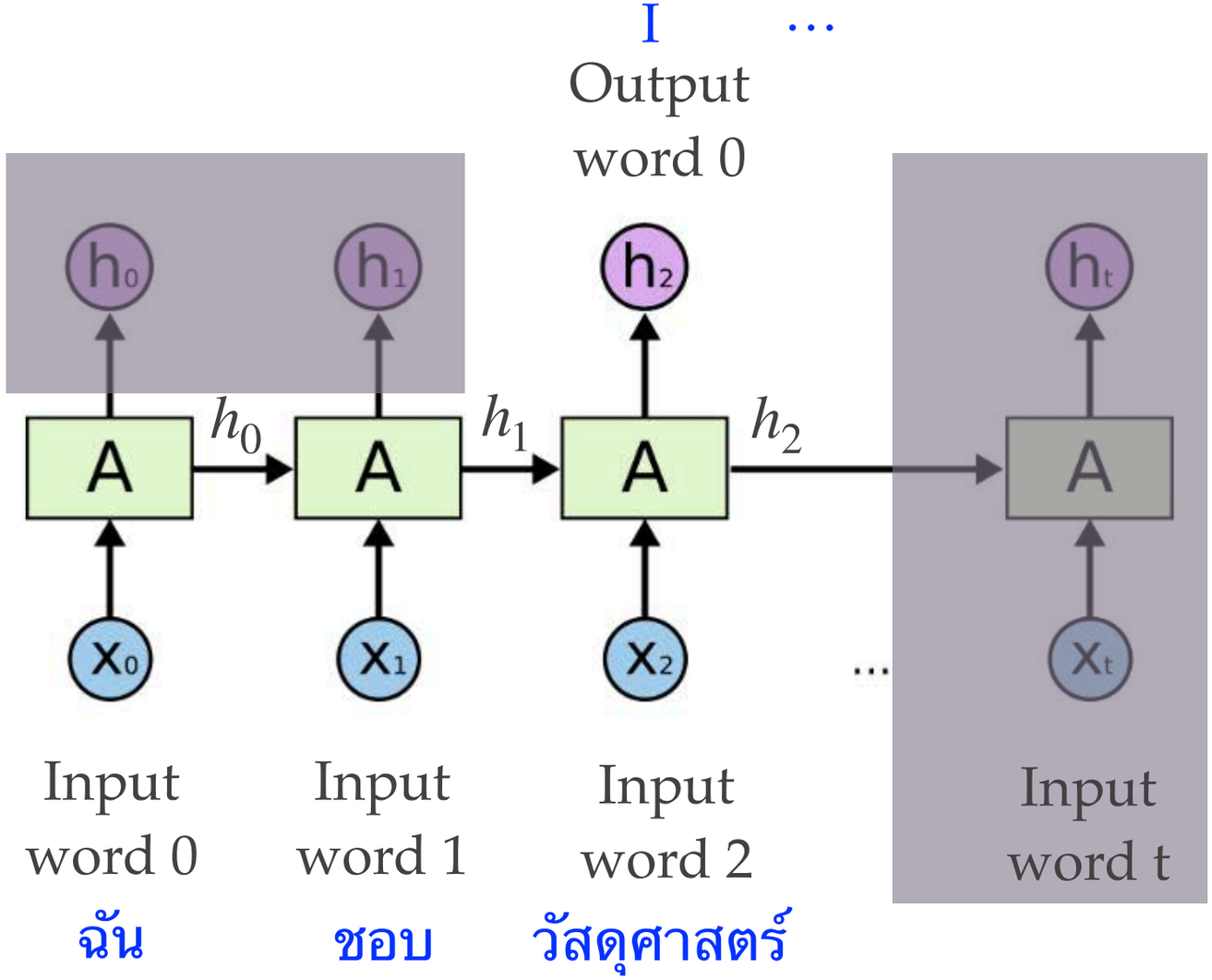


What is a potential problem with this model?

Understanding LSTM Networks by Christopher Colah

Recurrent Neural Network (RNN)

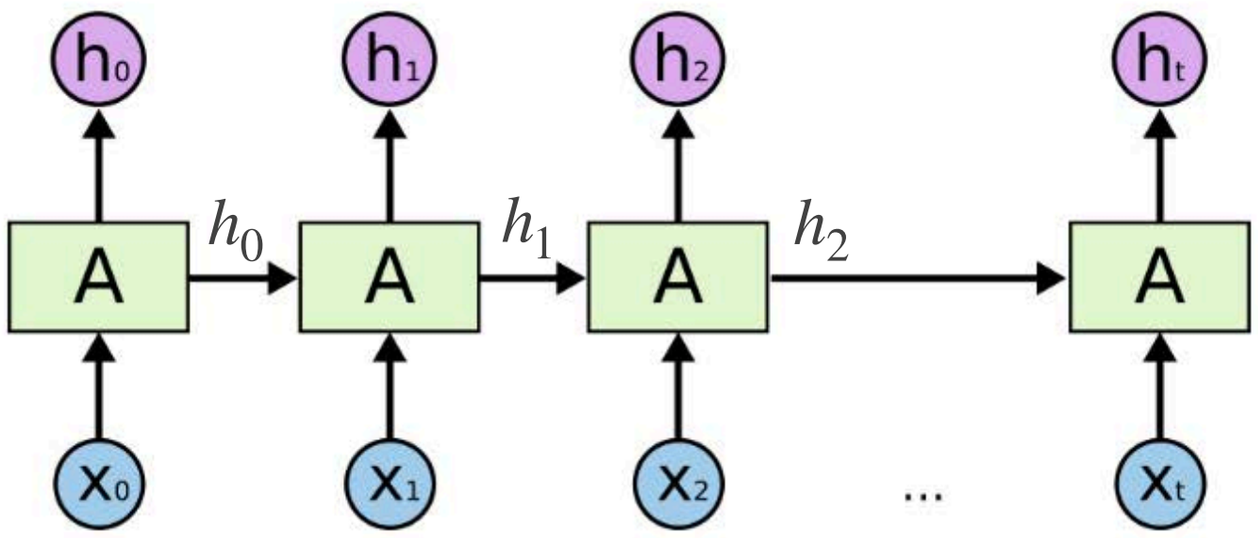
Task: Machine translation (e.g., translate language A to language B)



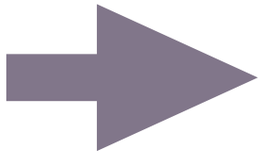
Understanding LSTM Networks by Christopher Colah

Recurrent Neural Network (RNN)

Task: Character-level language model sampling



h



(h)ello class. Today we are going to talk about recurrent neural networks.

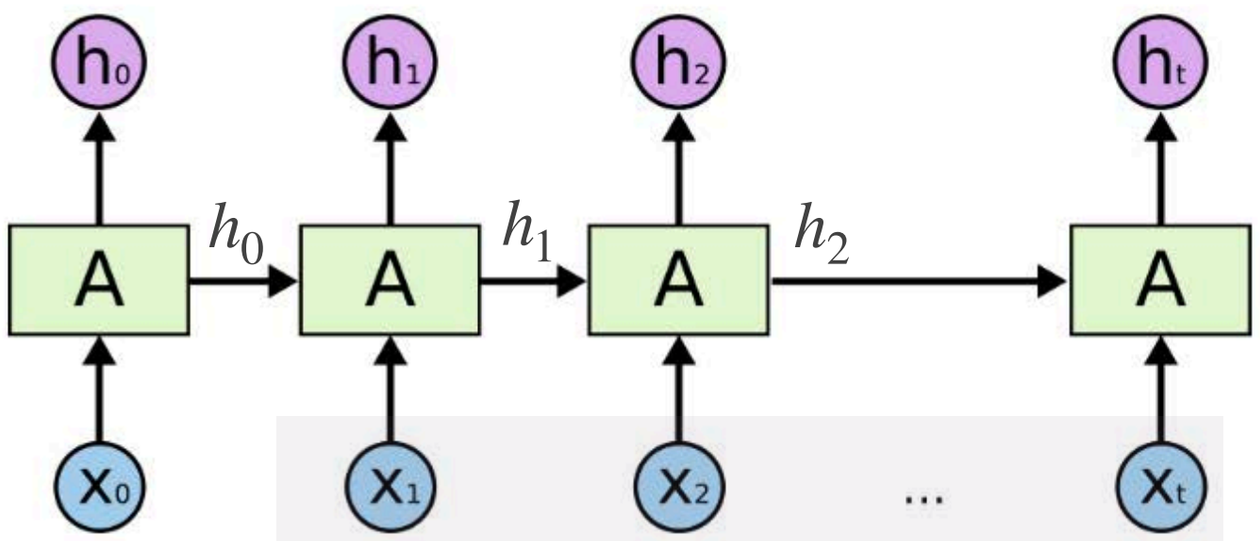
Understanding LSTM Networks by Christopher Colah

Recurrent Neural Network (RNN)

Task: Character-level language model sampling

e l l

Output character 0 | Output character 1 | Output character 2 | Output character t



Input character

h

"No" real additional input

Understanding LSTM Networks by Christopher Colah

keywords: greedy decoding, exhaustive search decoding, beam search decoding

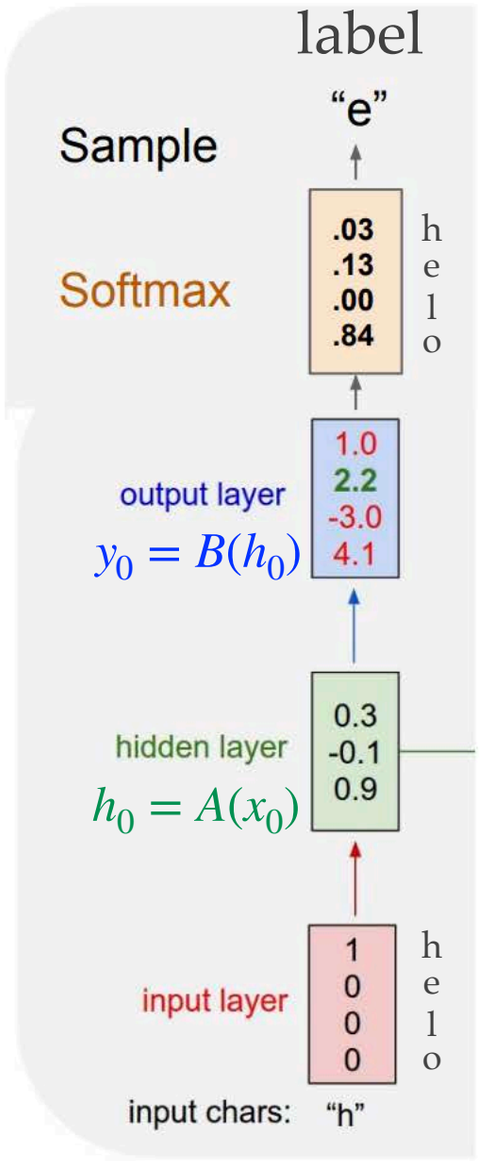
Recurrent Neural Network (RNN)

Task: Character-level language model sampling

Example: Character-level Language Model Sampling

Vocabulary:
[h,e,l,o]

At test-time sample characters one at a time, feed back to model



One-hot encoding

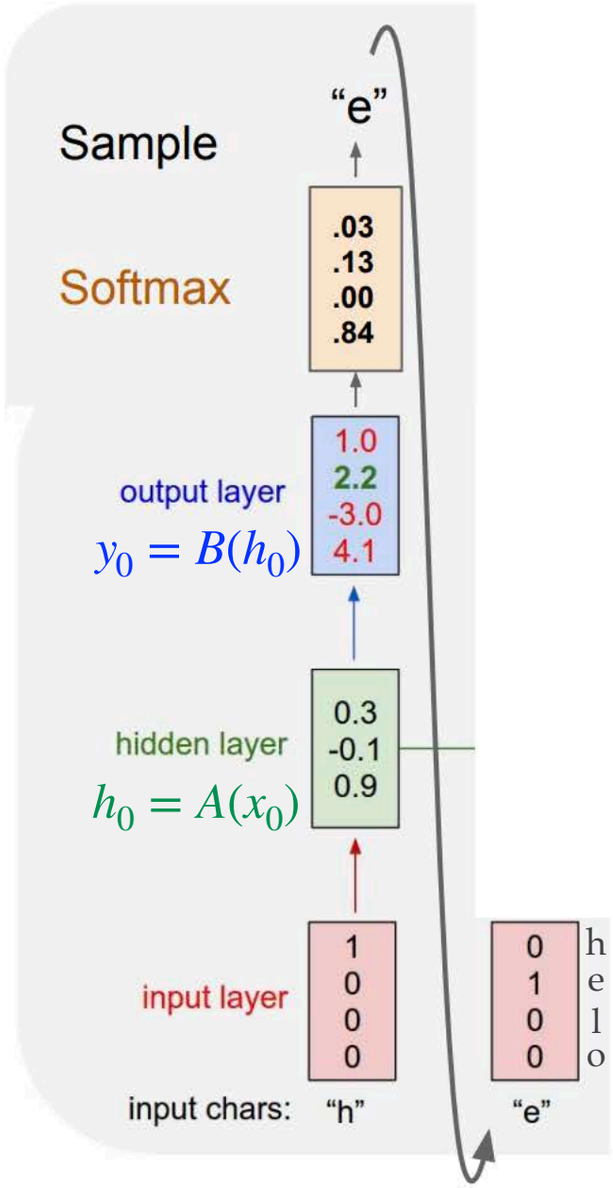
Recurrent Neural Network (RNN)

Task: Character-level language model sampling

Example: Character-level Language Model Sampling

Vocabulary:
[h,e,l,o]

At test-time sample characters one at a time, feed back to model



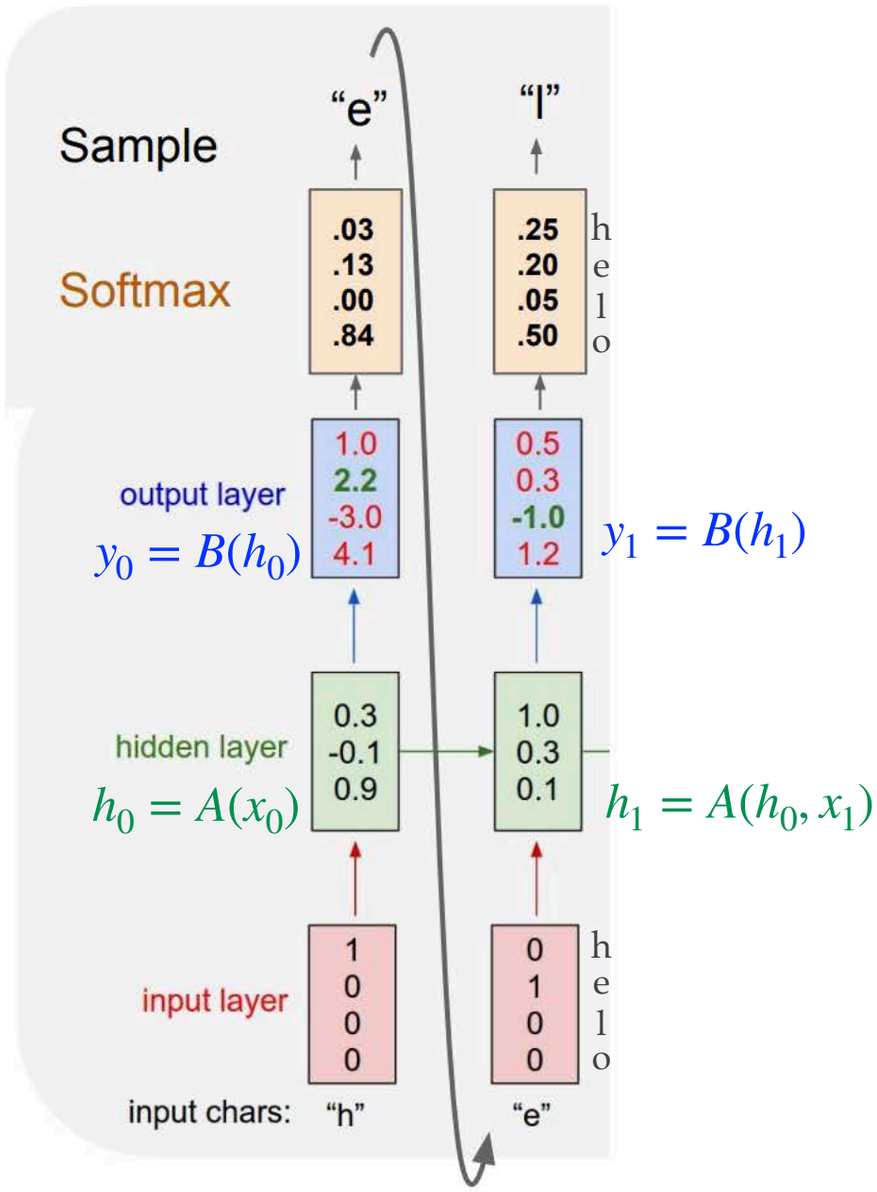
Recurrent Neural Network (RNN)

Task: Character-level language model sampling

Example: Character-level Language Model Sampling

Vocabulary:
[h,e,l,o]

At test-time sample characters one at a time, feed back to model



keywords: greedy decoding, exhaustive search decoding, beam search decoding

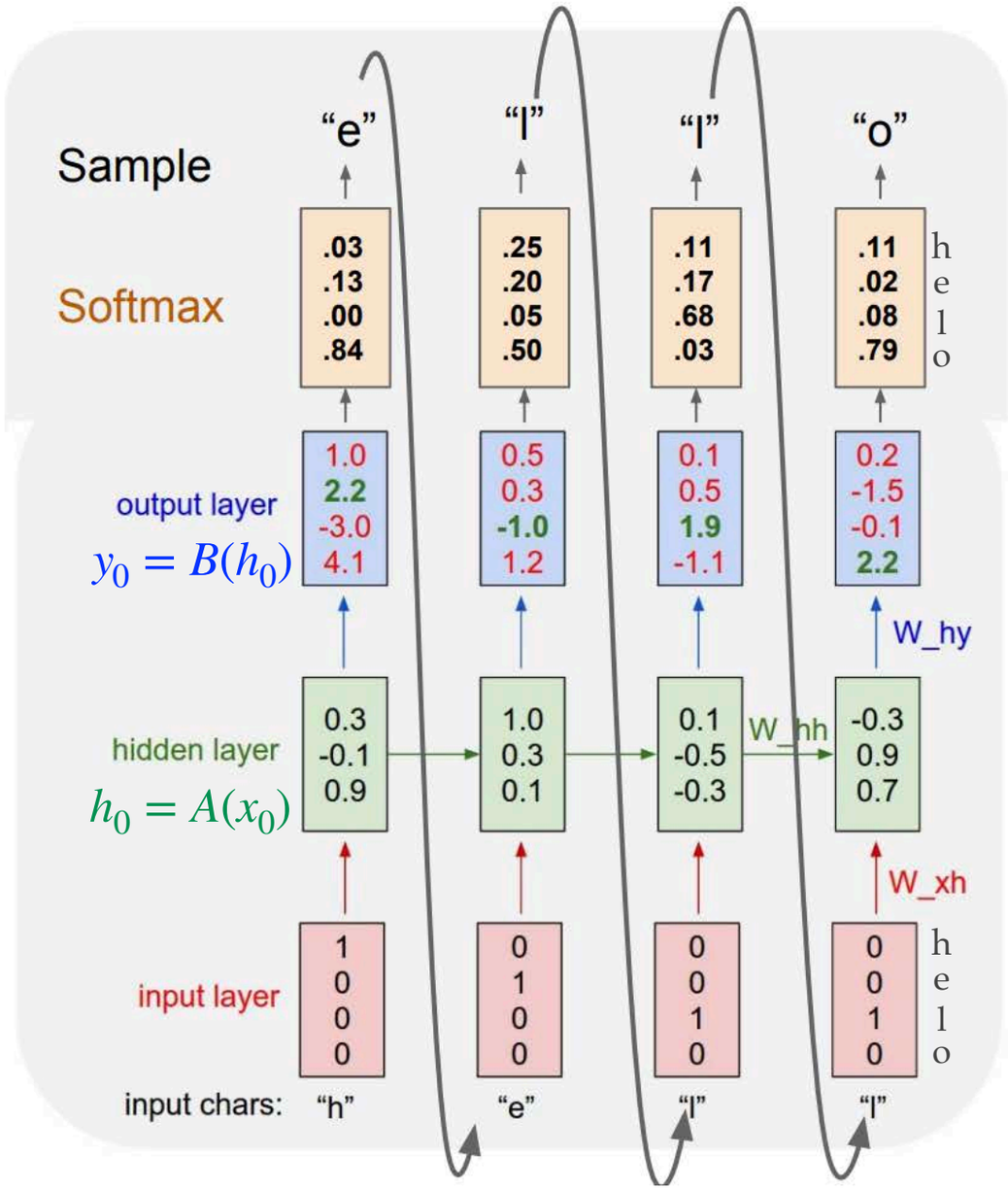
Recurrent Neural Network (RNN)

Task: Character-level language model sampling

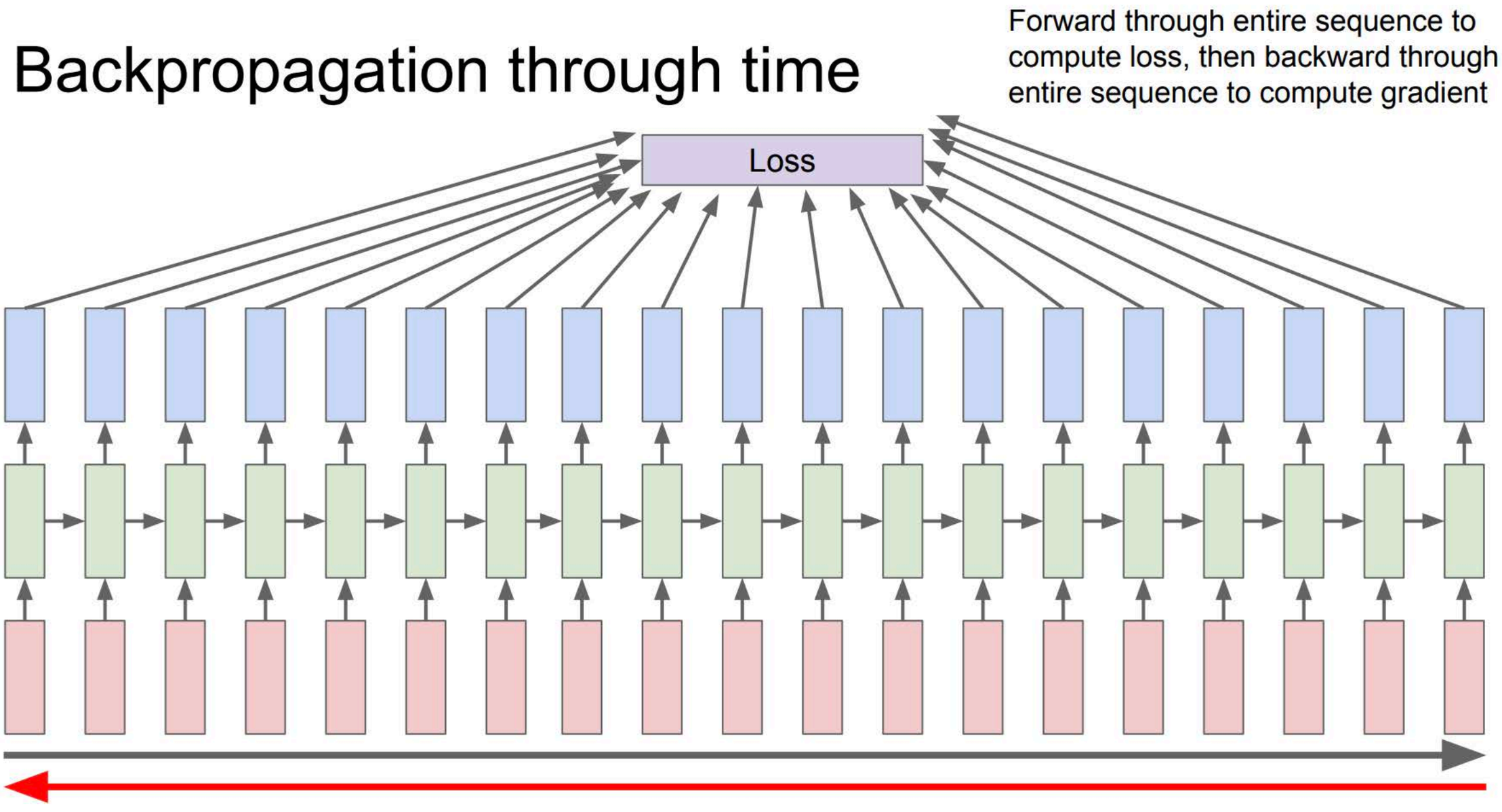
Example: Character-level Language Model Sampling

Vocabulary:
[h,e,l,o]

At test-time sample characters one at a time, feed back to model

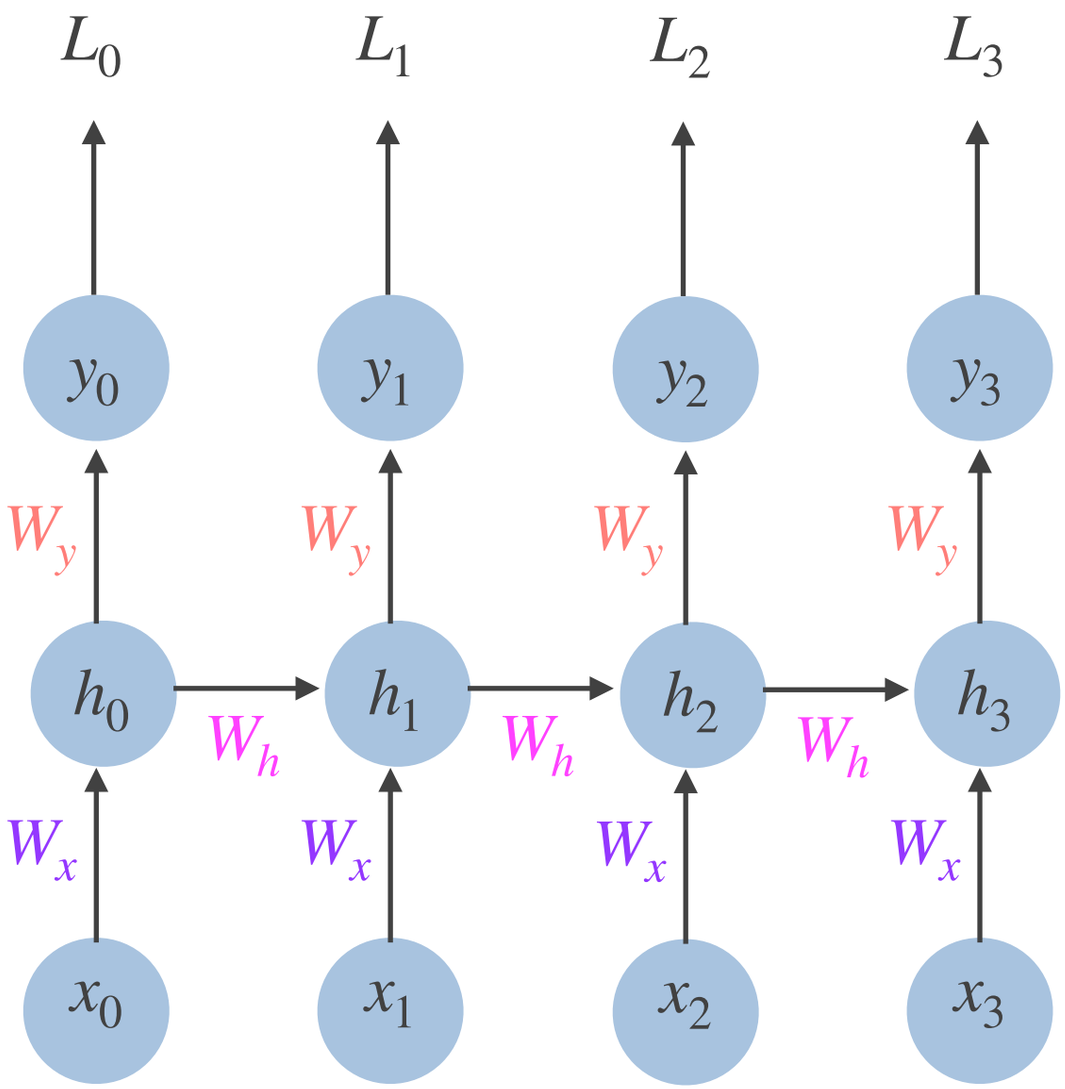


RNN - Backpropagation



Potentially encounter vanishing/exploding gradient problems!

RNN - Backpropagation



$$h_t = \tanh(W_x x_t + W_h h_{t-1})$$

$$y_t = f(W_y h_t)$$

Want to update W_x using stochastic gradient descent

Loss function $L = L_0 + L_1 + L_2 + L_3$

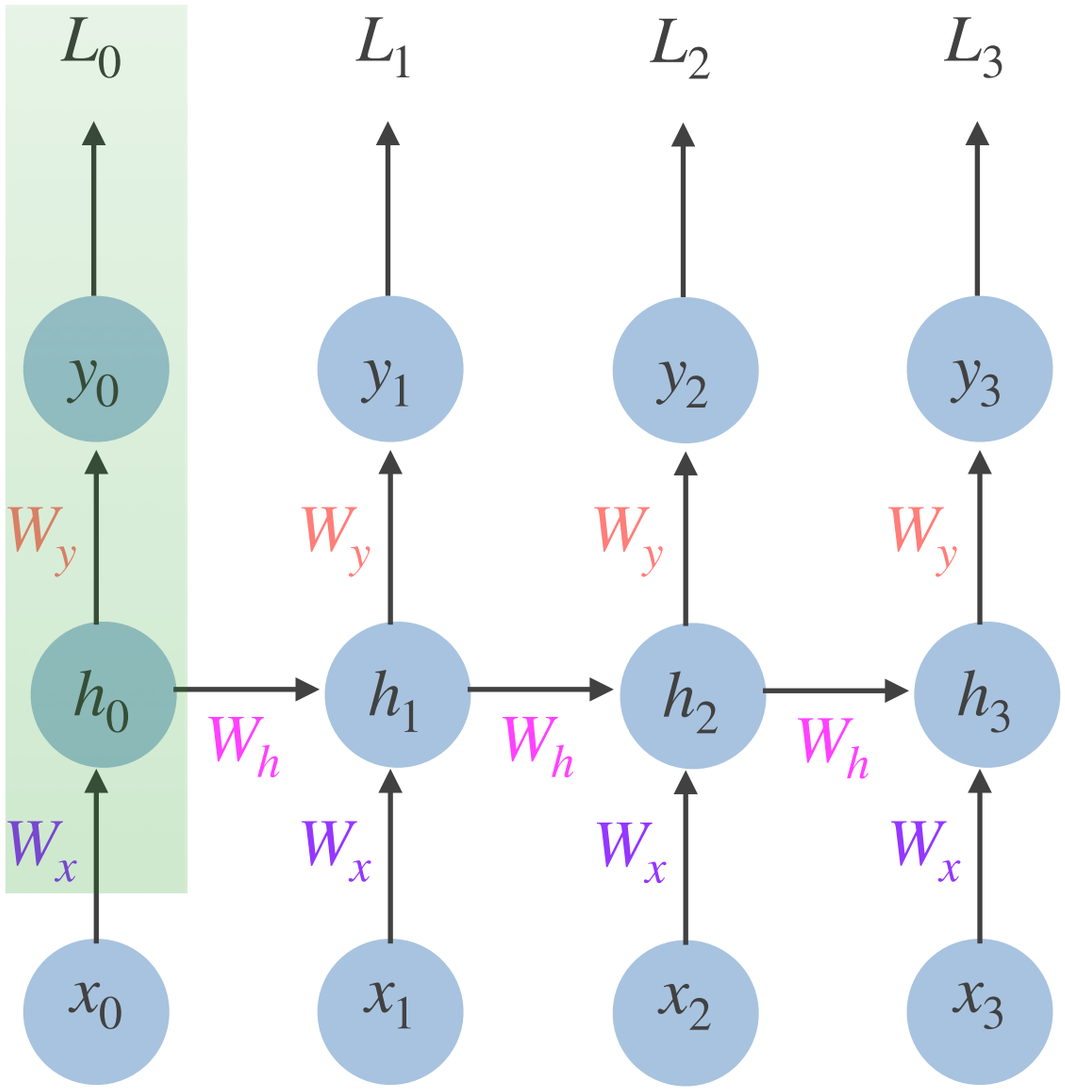
Update formula for iteration k

$$W_{x,(k)} = W_{x,(k-1)} - \alpha \nabla_{W_x} L(W_x)\,|_{W_x = W_{x,(k-1)}}$$

Decompose the gradient of L

$$\nabla_{W_x} L = \nabla_{W_x} L_0 + \nabla_{W_x} L_1 + \nabla_{W_x} L_2 + \nabla_{W_x} L_3$$

RNN - Backpropagation



$$\nabla_{W_x} L = \nabla_{W_x} L_0 + \nabla_{W_x} L_1 + \nabla_{W_x} L_2 + \nabla_{W_x} L_3$$

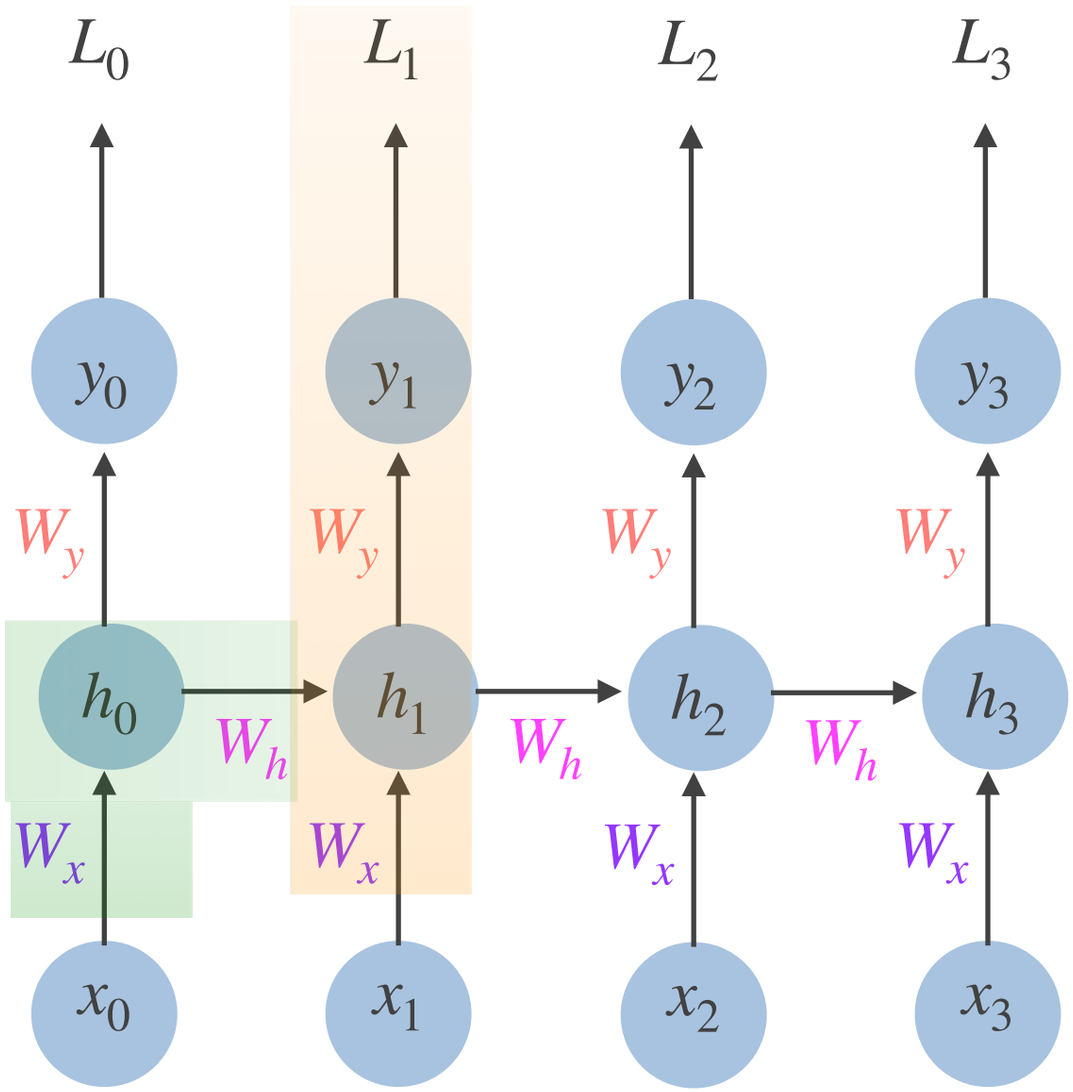
Using the chain rule

$$\frac{\partial L_0}{\partial W_x} = \frac{\partial L_0}{\partial y_0}\frac{\partial y_0}{\partial h_0}\frac{\partial h_0}{\partial W_x}$$

$$h_t = \tanh(W_x x_t + W_h h_{t-1})$$

$$y_t = f(W_y h_t)$$

RNN - Backpropagation



$$\nabla_{W_x} L = \nabla_{W_x} L_0 + \nabla_{W_x} L_1 + \nabla_{W_x} L_2 + \nabla_{W_x} L_3$$

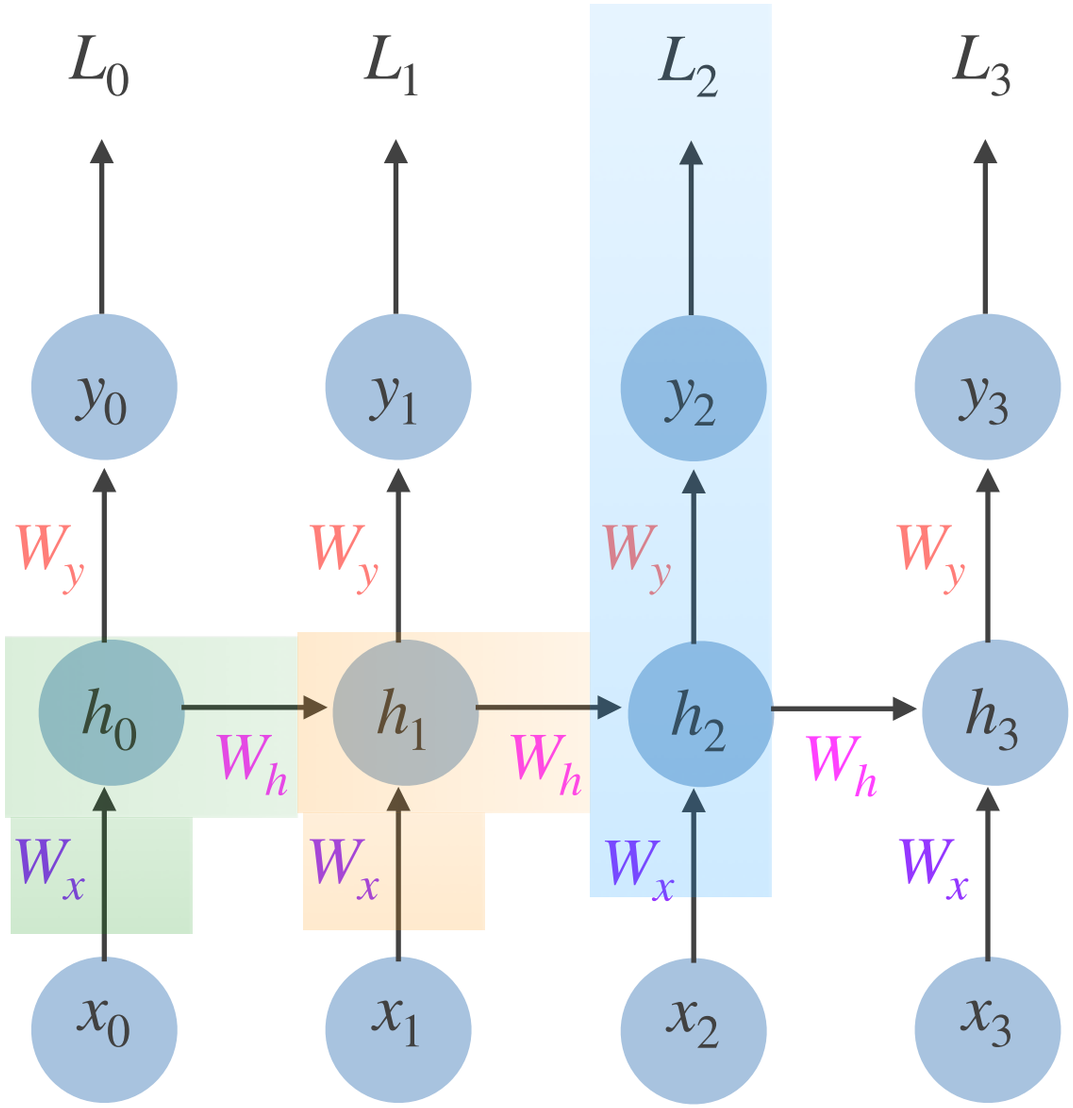
Using the chain rule

$$\frac{\partial L_1}{\partial W_x} = \frac{\partial L_1}{\partial y_1}\frac{\partial y_1}{\partial h_1}\left[\frac{\partial h_1}{\partial W_x} + \frac{\partial h_1}{\partial h_0}\frac{\partial h_0}{\partial W_x}\right]$$

$$h_t = \tanh(W_x x_t + W_h h_{t-1})$$

$$y_t = f(W_y h_t)$$

RNN - Backpropagation



$$h_t = \tanh(W_x x_t + W_h h_{t-1})$$

$$y_t = f(W_y h_t)$$

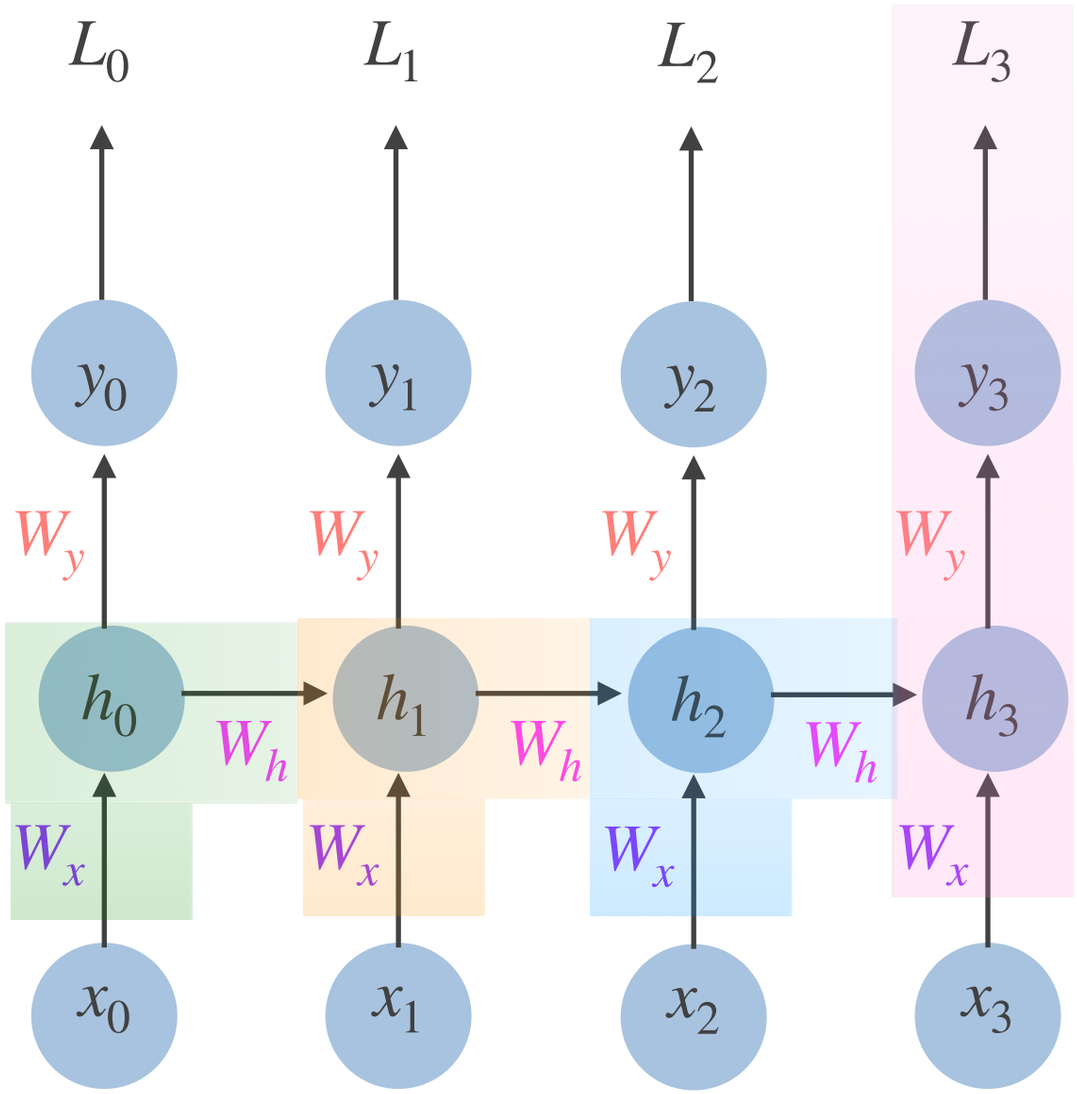
$$\nabla_{W_x} L = \nabla_{W_x} L_0 + \nabla_{W_x} L_1 + \nabla_{W_x} L_2 + \nabla_{W_x} L_3$$

Using the chain rule

$$\frac{\partial L_2}{\partial W_x} = \frac{\partial L_2}{\partial y_2}\frac{\partial y_2}{\partial h_2}\cdots$$

$$\left[\frac{\partial h_2}{\partial W_x} + \frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial W_x} + \frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial h_0}\frac{\partial h_0}{\partial W_x}\right]$$

RNN - Backpropagation



$$h_t = \tanh(W_x x_t + W_h h_{t-1})$$

$$y_t = f(W_y h_t)$$

$$\nabla_{W_x} L = \nabla_{W_x} L_0 + \nabla_{W_x} L_1 + \nabla_{W_x} L_2 + \nabla_{W_x} L_3$$

Using the chain rule

$$\frac{\partial L_3}{\partial W_x} = \frac{\partial L_3}{\partial y_3}\frac{\partial y_3}{\partial h_3}\cdots$$

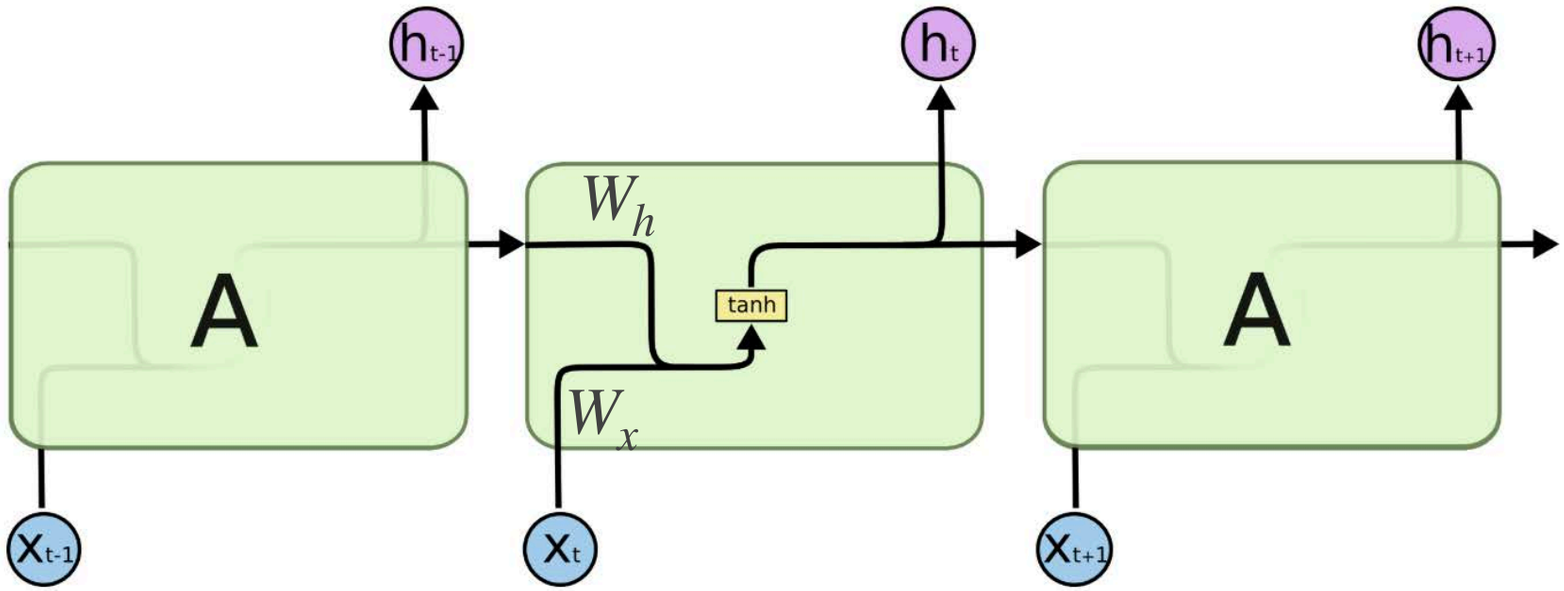
$$\left[\frac{\partial h_3}{\partial W_x} + \frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial W_x} + \frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial W_x} + \frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial h_0}\frac{\partial h_0}{\partial W_x}\right]$$

1 term 2 terms 3 terms 4 terms

More terms…potential vanishing/ exploding gradient problems

While the exploding gradient problem can be controlled with gradient clipping, the vanishing gradient problem is more difficult to solve.

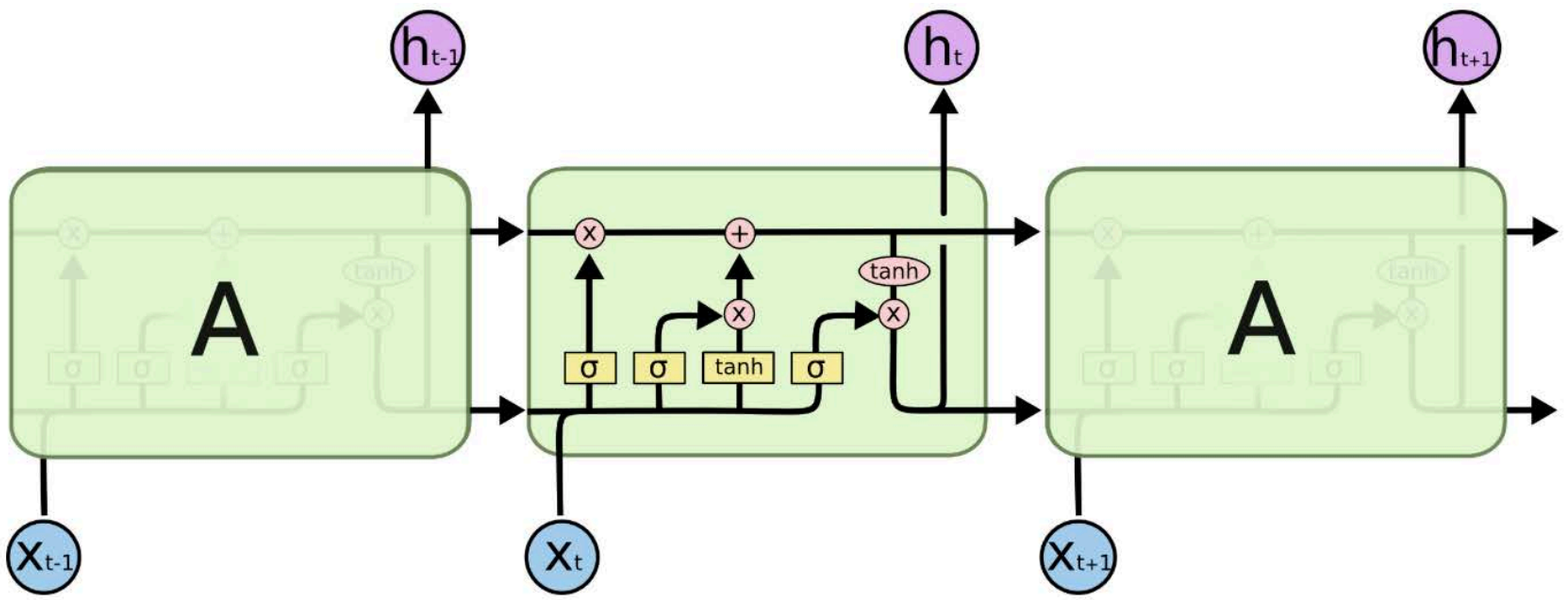
Long Short-Term Memory (LSTM)



The repeating module in a standard RNN contains a single layer.

$$h_t = \tanh(W_x x_t + W_h h_{t-1} + b)$$

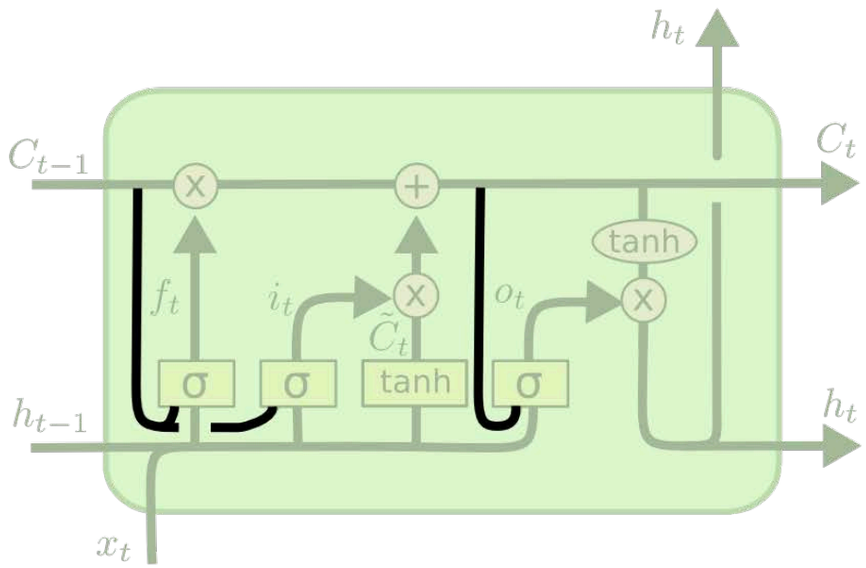
h_t could be the hidden state/output



The repeating module in an LSTM contains four interacting layers.

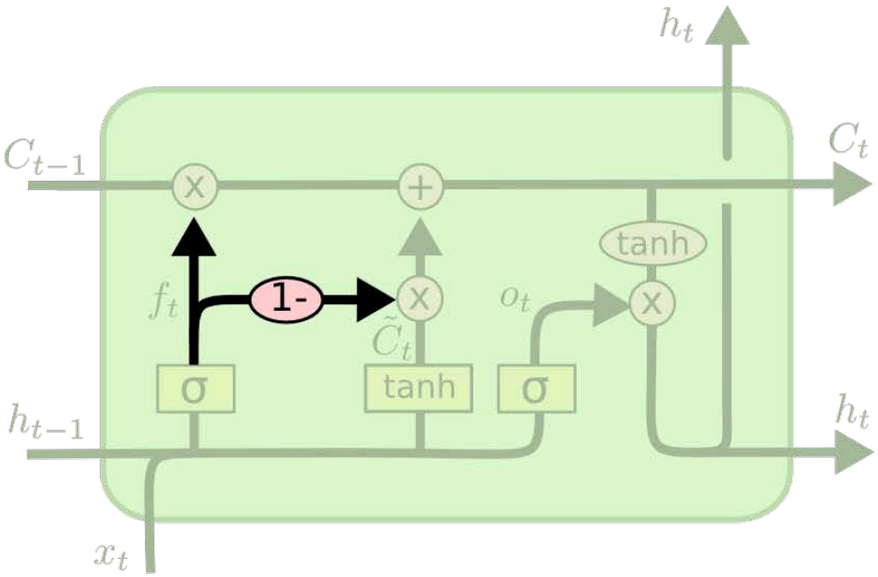
$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$
$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * \tanh(C_t)$$

Understanding LSTM Networks by Christopher Colah



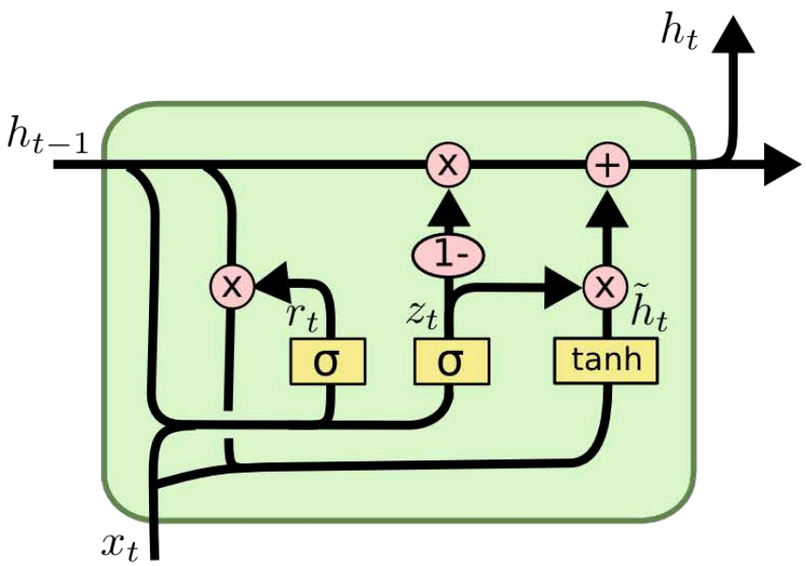
LSTM with peephole connections

$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$



LSTM with coupled input and forget gates

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



Gated Recurrent Units (GRU)

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

Examples

at first:

```
tyntd-iafhatawiaoihrdemot  lytdws  e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e
plia tklrgd t o idoe ns,smtt   h ne etie h,hregtrs nigtike,aoaenns lng
```

train more

```
"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."
```

train more

```
Aftair fall unsuch that the hall for Prince Velzonski's that me of
her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort
how, and Gogition is so overelical and ofter.
```

train more

```
"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftened him.
Pierre aking his soul came to the packs and drove up his father-in-law women.
```

Examples

Proof. Omitted. □

Lemma 0.1. *Let $\mathcal{C}$ be a set of the construction.*

Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

.

Proof. This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{étale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

Lemma 0.2. *This is an integer $\mathcal{Z}$ is injective.*

Proof. See Spaces, Lemma ??. □

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

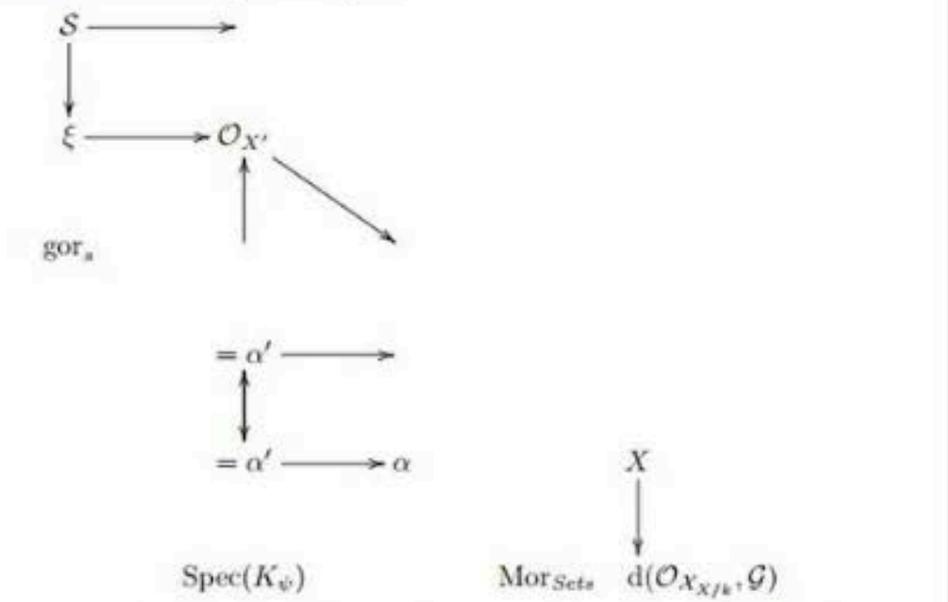
be a morphism of algebraic spaces over S and Y.

Proof. Let X be a nonzero scheme of X. Let X be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

(1) $\mathcal{F}$ is an algebraic space over S.
(2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram



is a limit. Then $\mathcal{G}$ is a finite type and assume S is a flat and $\mathcal{F}$ and $\mathcal{G}$ is a finite type f_*. This is of finite type diagrams, and

- the composition of $\mathcal{G}$ is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

□

Proof. We have see that $X = \mathrm{Spec}(R)$ and $\mathcal{F}$ is a finite type representable by algebraic space. The property $\mathcal{F}$ is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U. □

Proof. This is clear that $\mathcal{G}$ is a finite presentation, see Lemmas ??.
A *reduced above* we conclude that U is an open covering of $\mathcal{C}$. The functor $\mathcal{F}$ is a "field

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\overline{x}} \quad \text{-}1(\mathcal{O}_{X_{\acute{e}tale}}) \longrightarrow \mathcal{O}_{X_i}^{-1}\mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^{\overline{w}})$$

is an isomorphism of covering of $\mathcal{O}_{X_i}$. If $\mathcal{F}$ is the unique element of $\mathcal{F}$ such that X is an isomorphism.
The property $\mathcal{F}$ is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme $\mathcal{O}_X$-algebra with $\mathcal{F}$ are opens of finite type over S.
If $\mathcal{F}$ is a scheme theoretic image points. □

If $\mathcal{F}$ is a finite direct sum $\mathcal{O}_{X_\lambda}$ is a closed immersion, see Lemma ??. This is a sequence of $\mathcal{F}$ is a similar morphism.

Examples

```
static void do_command(struct seq_file *m, void *v)
{
  int column = 32 << (cmd[2] & 0x80);
  if (state)
    cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
  else
    seq = 1;
  for (i = 0; i < 16; i++) {
    if (k & (1 << 1))
      pipe = (in_use & UMXTHREAD_UNCCA) +
        ((count & 0x00000000ffffffff8) & 0x000000f) << 8;
    if (count == 0)
      sub(pid, ppc_md.kexec_handle, 0x20000000);
    pipe_set_bytes(i, 0);
  }
  /* Free our user pages pointer to place camera if all dash */
  subsystem_info = &of_changes[PAGE_SIZE];
  rek_controls(offset, idx, &soffset);
  /* Now we want to deliberately put it to device */
  control_check_polarity(&context, val, 0);
  for (i = 0; i < COUNTER; i++)
    seq_puts(s, "policy ");
}
```

Generated C code

Outline

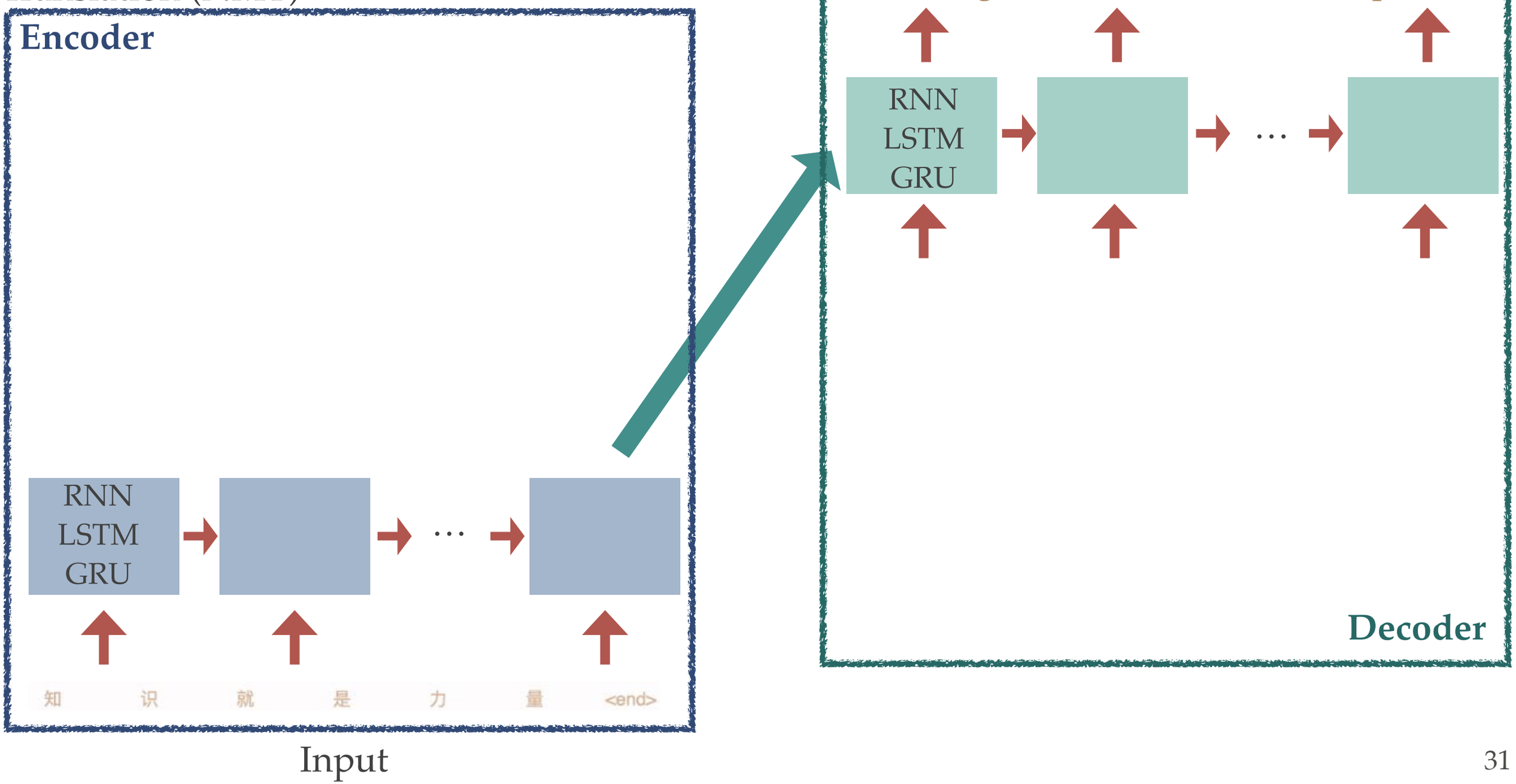
- Recurrent Neural Network (RNN)
- Sequence-to-Sequence Model
  - Attention Mechanism
- Self-Attention and Transformer
- Graph Neural Network

Sequence-to-Sequence Model

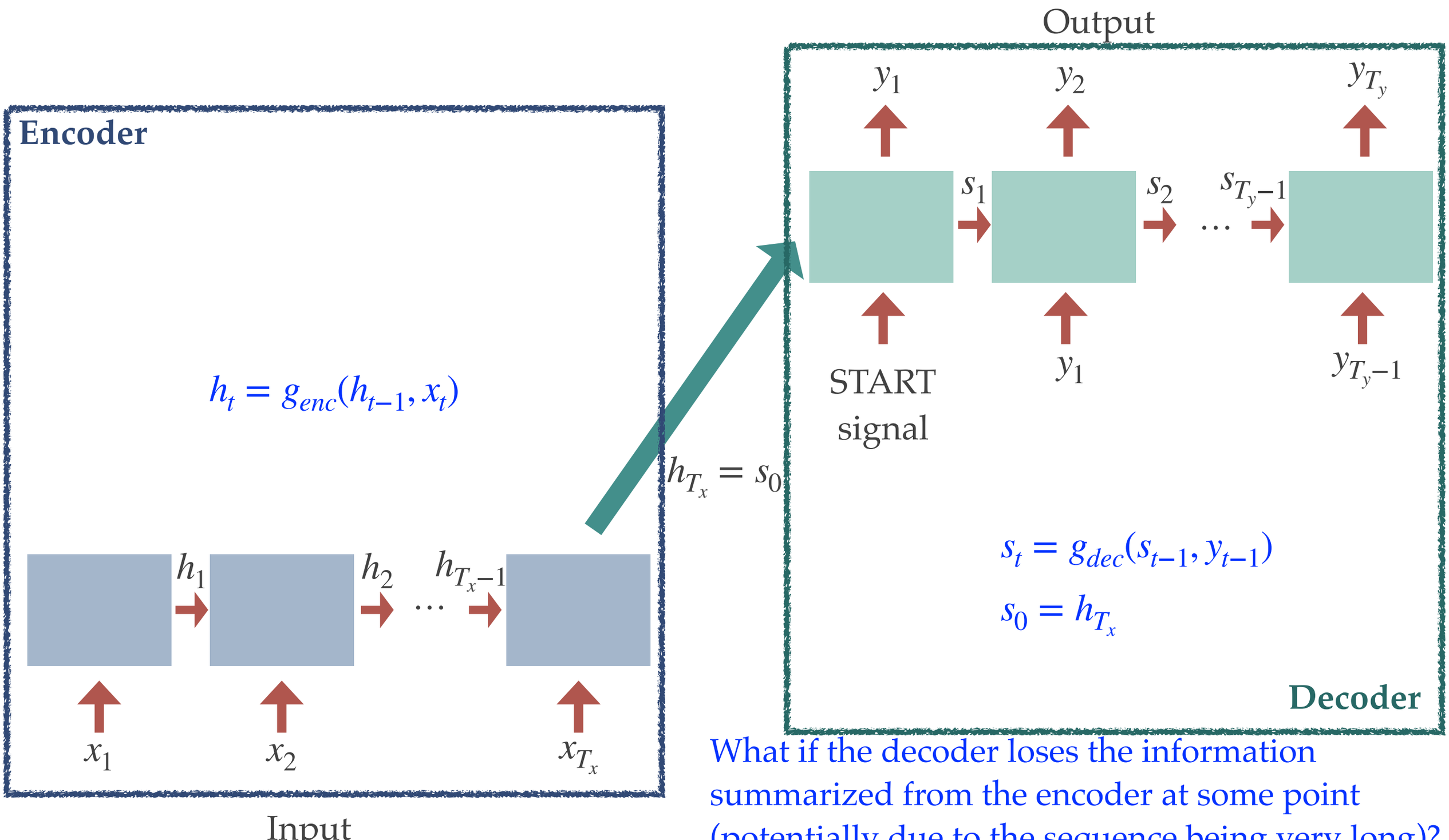
An RNN encoder-decoder model

The encoder summarizes the information from the input sequence and send it to the decoder for further processing

Task: Chinese-to-English Neural Machine Translation (NMT)

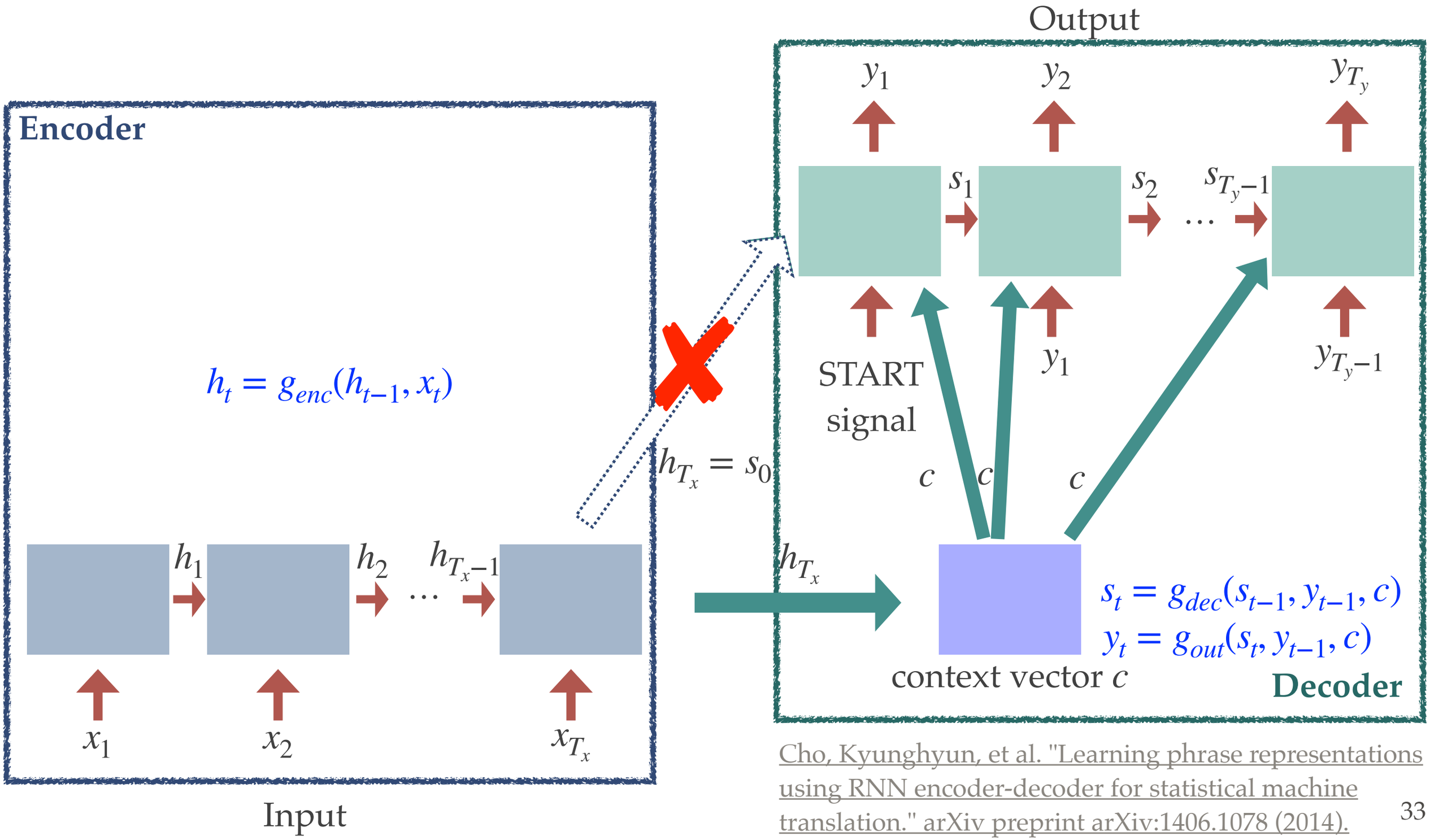


Sequence-to-Sequence Model



What if the decoder loses the information summarized from the encoder at some point (potentially due to the sequence being very long)?

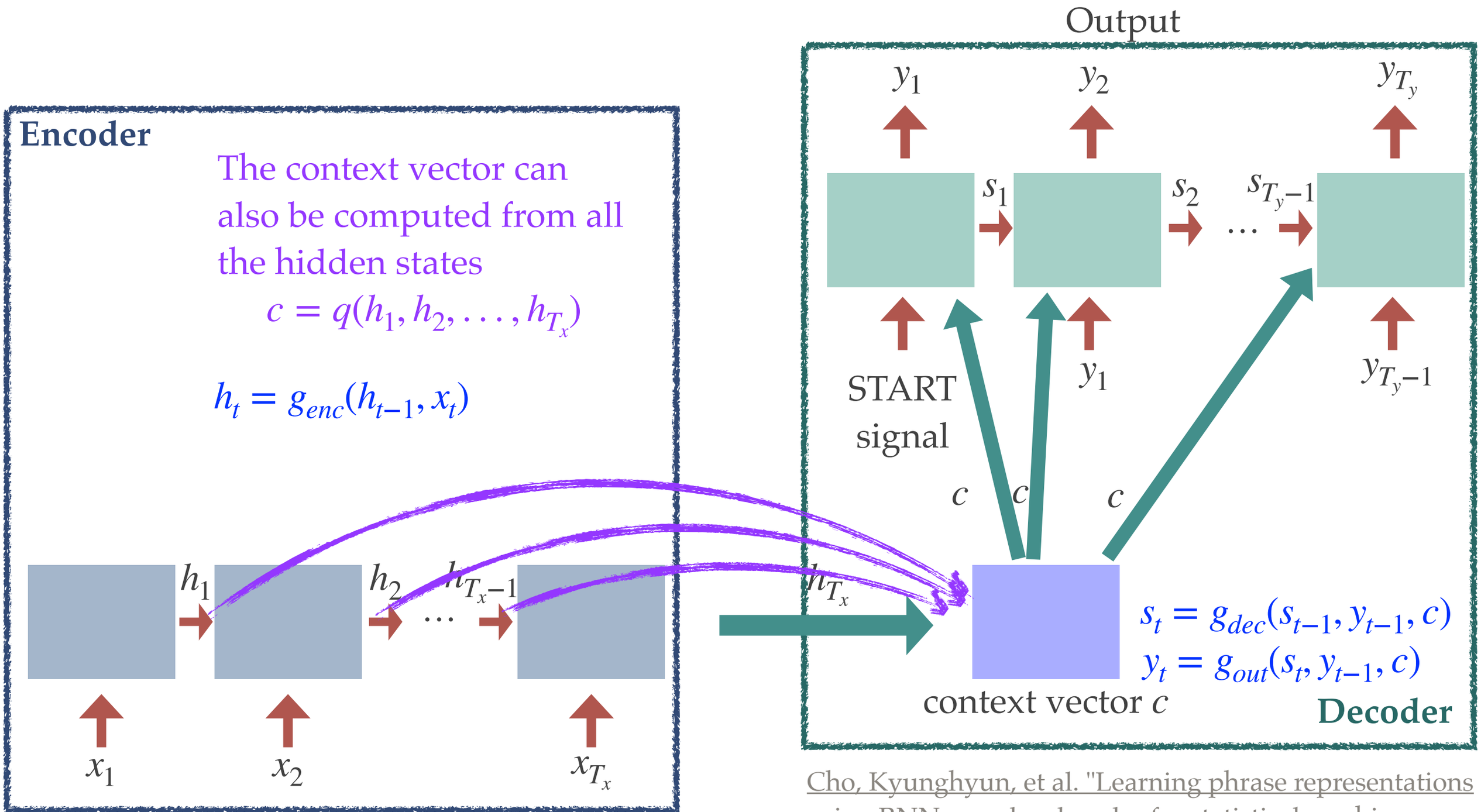
Sequence-to-Sequence Model



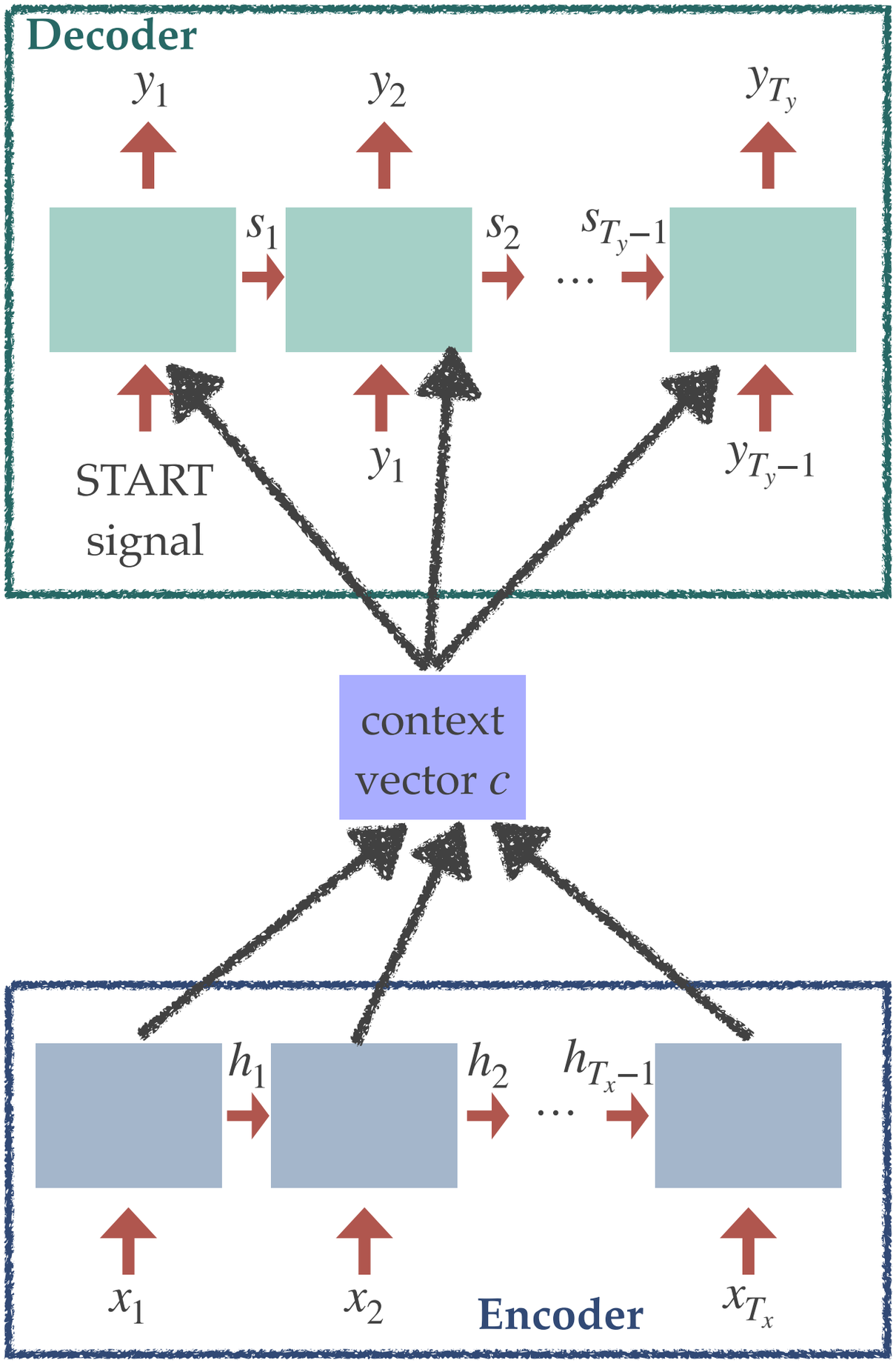
Cho, Kyunghyun, et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." arXiv preprint arXiv:1406.1078 (2014).

Question: Is a single context vector enough to capture all important information on the input?

Answer: Most likely not → **Attention Mechanism** where each step of the decoder can access information from all hidden states of the encoder



Cho, Kyunghyun, et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." arXiv preprint arXiv:1406.1078 (2014).



Without attention mechanism

$$s_t = g_{dec}(s_{t-1}, y_{t-1}, c)$$
$$y_t = g_{out}(s_t, y_{t-1}, c)$$
$$c = q(h_1, h_2, \ldots, h_{T_x})$$

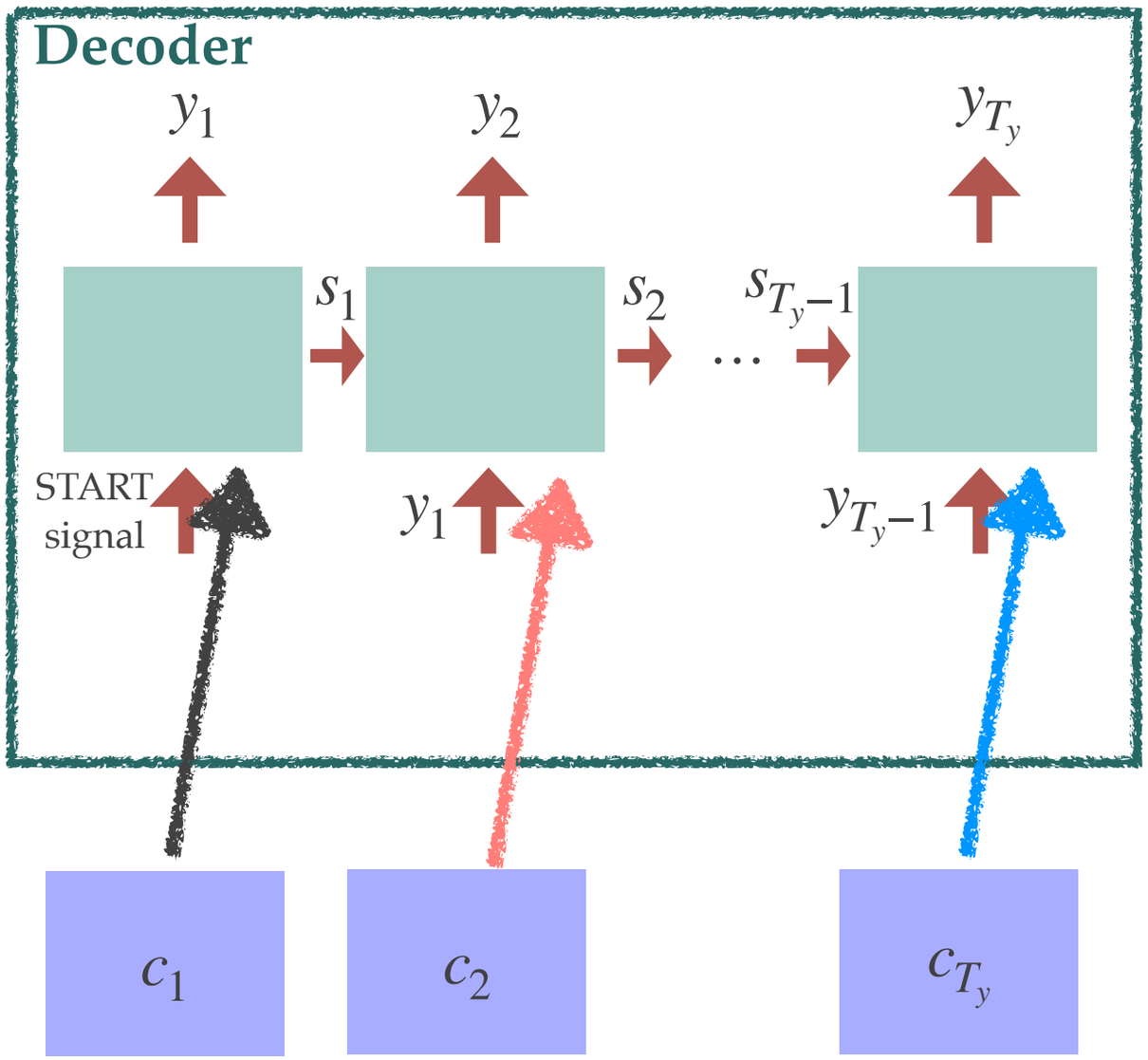
With attention mechanism

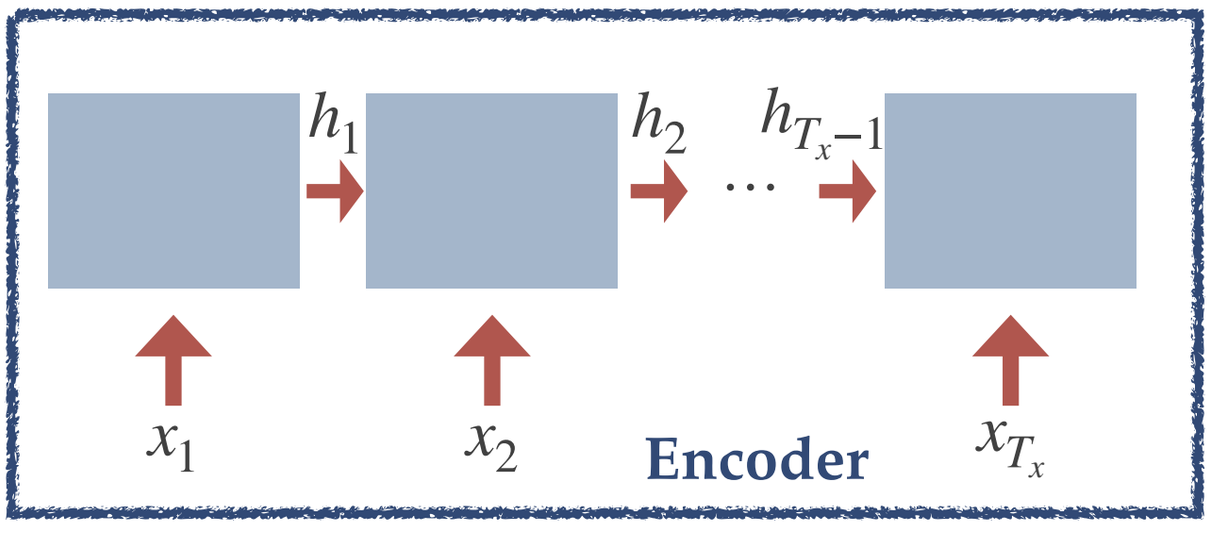
$$s_t = g_{dec}(s_{t-1}, y_{t-1}, c_t)$$
$$y_t = g_{out}(s_t, y_{t-1}, c_t)$$
$$c_t = q_t(h_1, h_2, \ldots, h_{T_x}, s_{t-1})$$

This is just one of many ways to include attention mechanisms (simplified from Bahdanau et al. ICLR 2015)

$$h_t = g_{enc}(h_{t-1}, x_t)$$

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).





Without attention mechanism

$$s_t = g_{dec}(s_{t-1}, y_{t-1}, c)$$
$$y_t = g_{out}(s_t, y_{t-1}, c)$$
$$c = q(h_1, h_2, \ldots, h_{T_x})$$

With attention mechanism

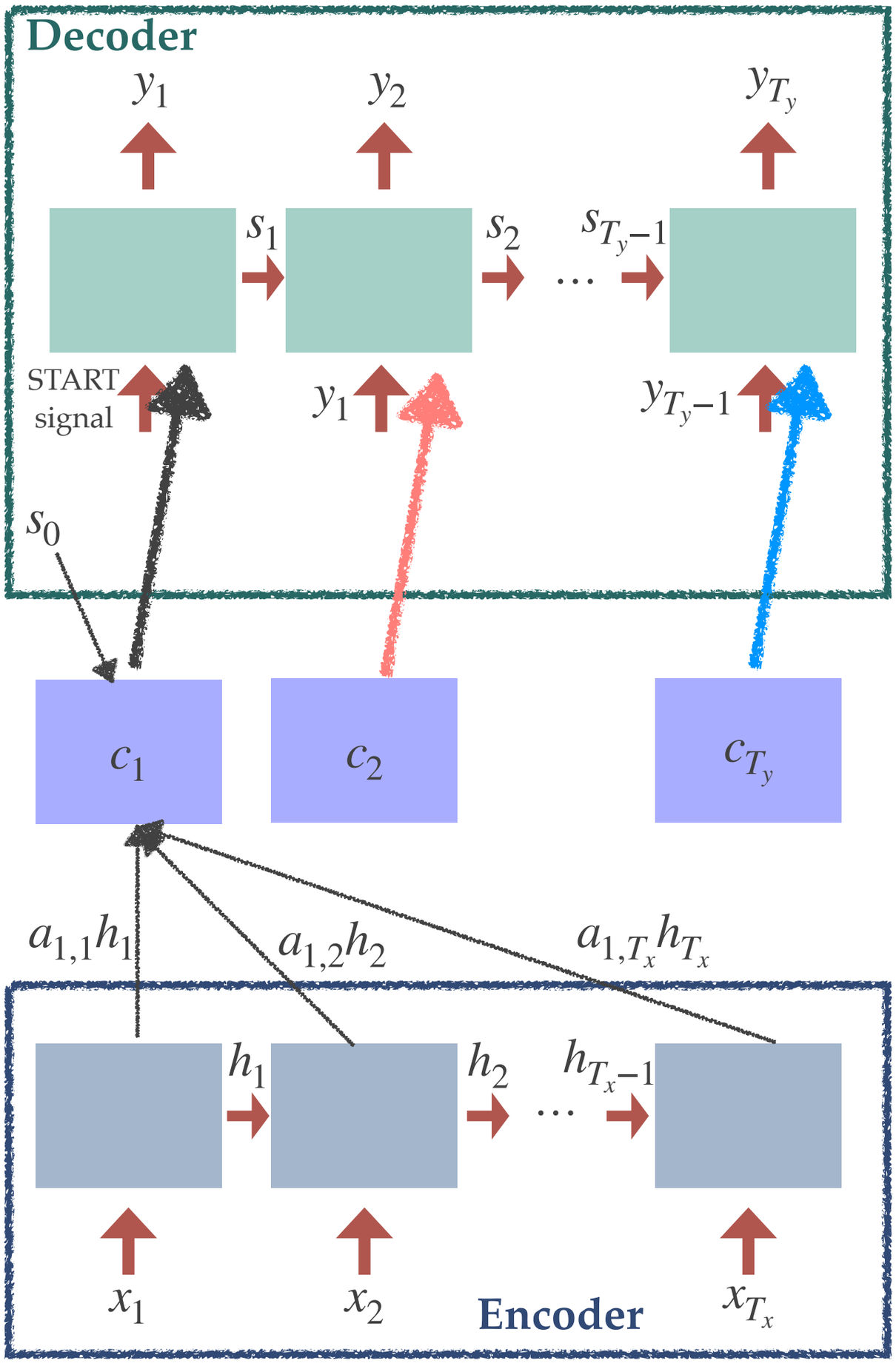
$$s_t = g_{dec}(s_{t-1}, y_{t-1}, c_t)$$
$$y_t = g_{out}(s_t, y_{t-1}, c_t)$$
$$c_t = q_t(h_1, h_2, \ldots, h_{T_x}, s_{t-1})$$

Different c_t's for different time points

The context vector is a weighted sum of the encoder hidden states

$$h_t = g_{enc}(h_{t-1}, x_t)$$

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).



Without attention mechanism

$$s_t = g_{dec}(s_{t-1}, y_{t-1}, c)$$
$$y_t = g_{out}(s_t, y_{t-1}, c)$$
$$c = q(h_1, h_2, \ldots, h_{T_x})$$

With attention mechanism

$$s_t = g_{dec}(s_{t-1}, y_{t-1}, c_t)$$
$$y_t = g_{out}(s_t, y_{t-1}, c_t)$$
$$c_t = q_t(h_1, h_2, \ldots, h_{T_x}, s_{t-1})$$

Different c_t's for different time points

The context vector is a weighted sum of the encoder hidden states

$$c_1 = a_{1,1}h_1 + a_{1,2}h_2 + \ldots + a_{1,T_x}h_{T_x}$$

$$1 = a_{1,1} + a_{1,2} + \ldots + a_{1,T_x}$$
$$0 \le a_{1,i}$$

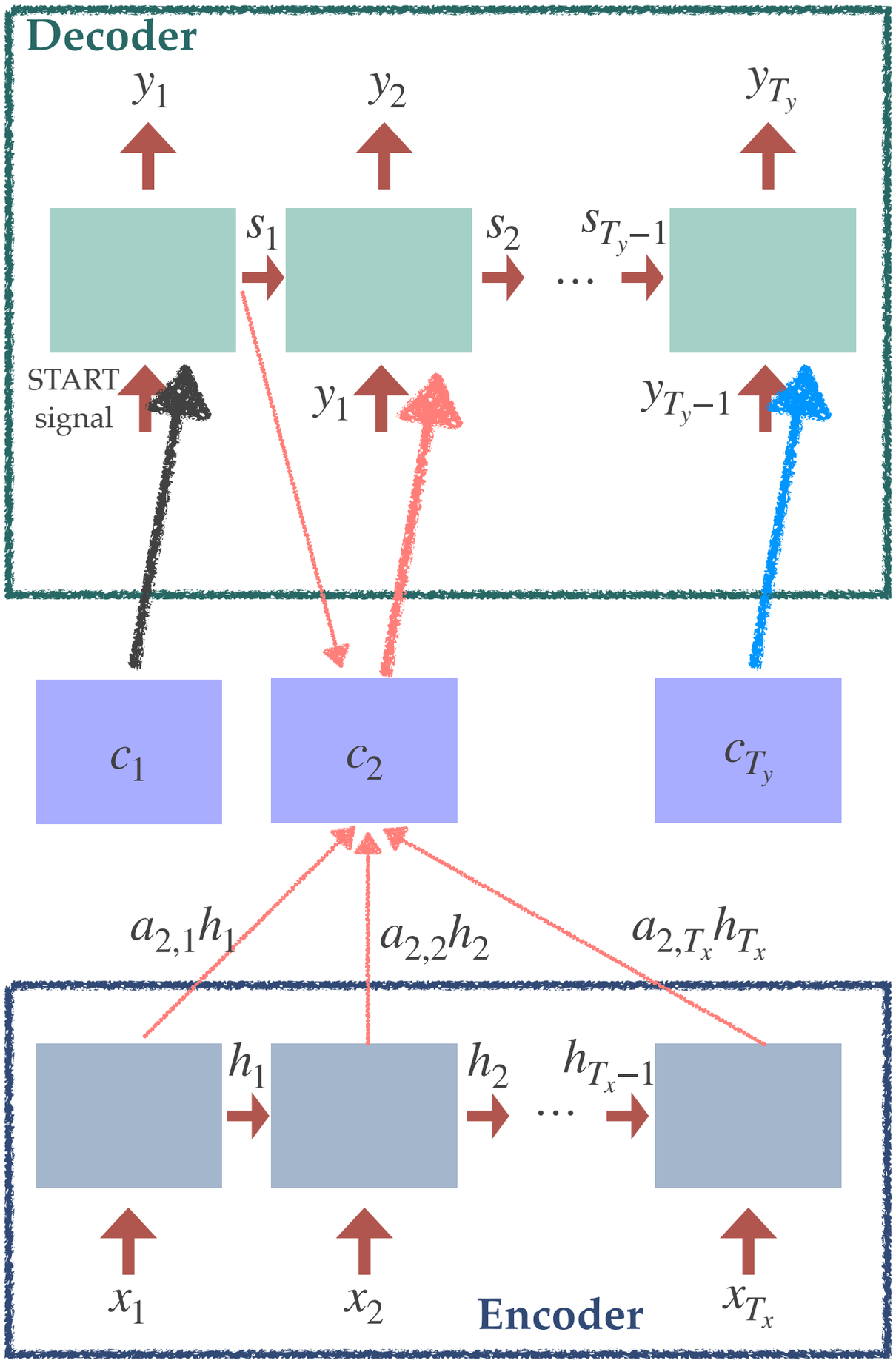
These constraints can be enforced using softmax

$$a_{1,i} = score(s_0, h_i)$$

A function that computes a "similarity" score between the two inputs (e.g., inner product)

$$h_t = g_{enc}(h_{t-1}, x_t)$$

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).



Without attention mechanism

$$s_t = g_{dec}(s_{t-1}, y_{t-1}, c)$$
$$y_t = g_{out}(s_t, y_{t-1}, c)$$
$$c = q(h_1, h_2, \ldots, h_{T_x})$$

With attention mechanism

$$s_t = g_{dec}(s_{t-1}, y_{t-1}, c_t)$$
$$y_t = g_{out}(s_t, y_{t-1}, c_t)$$
$$c_t = q_t(h_1, h_2, \ldots, h_{T_x}, s_{t-1})$$

Different c_t's for different time points

The context vector is a weighted sum of the encoder hidden states

$$c_2 = a_{2,1}h_1 + a_{2,2}h_2 + \ldots + a_{2,T_x}h_{T_x}$$

$$1 = a_{2,1} + a_{2,2} + \ldots + a_{2,T_x}$$
$$0 \leq a_{2,i}$$

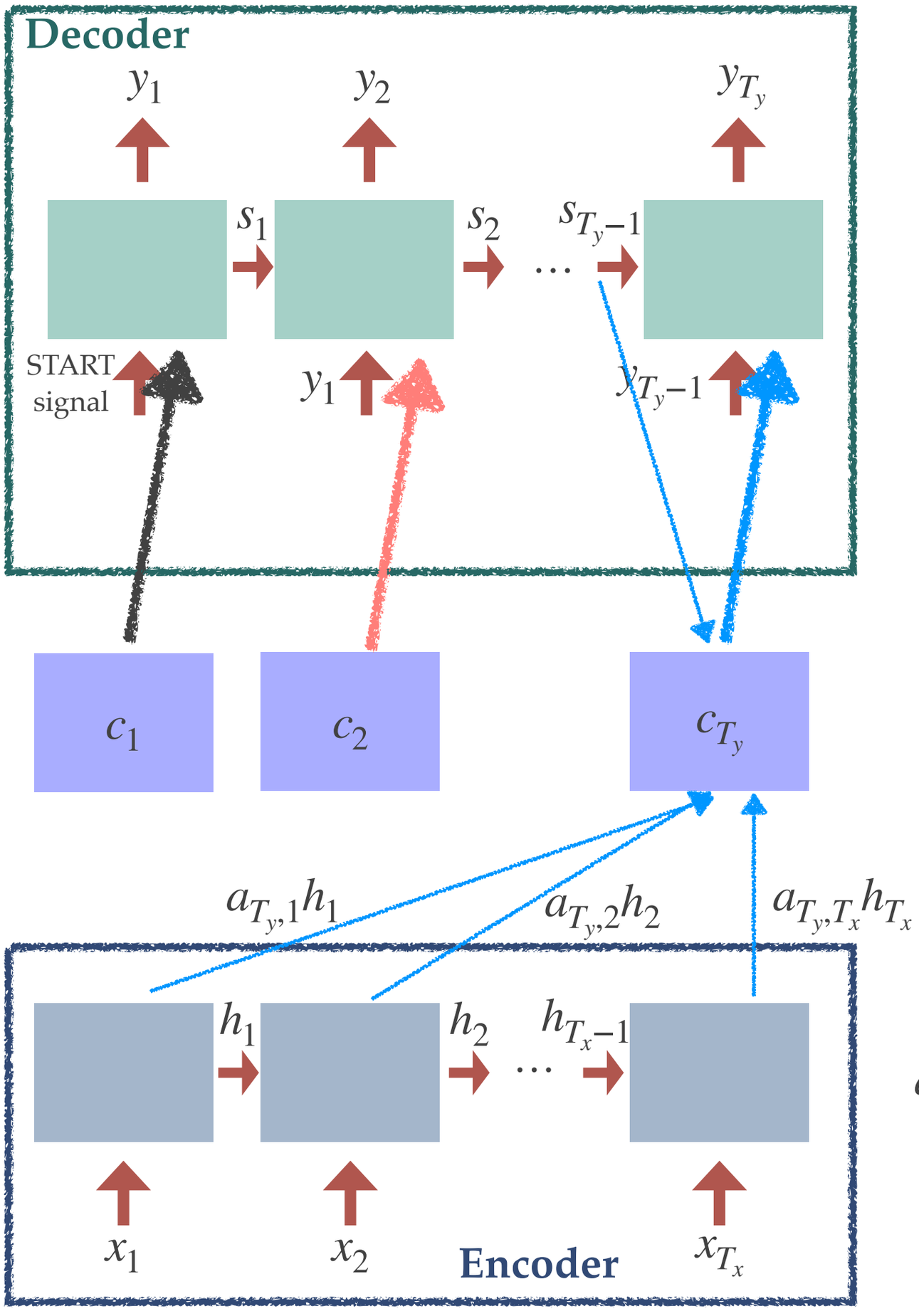
These constraints can be enforced using softmax

$$a_{2,i} = score(s_1, h_i)$$

A function that computes a "similarity" score between the two inputs (e.g., inner product)

$$h_t = g_{enc}(h_{t-1}, x_t)$$

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).



Without attention mechanism

$$s_t = g_{dec}(s_{t-1}, y_{t-1}, c)$$
$$y_t = g_{out}(s_t, y_{t-1}, c)$$
$$c = q(h_1, h_2, \ldots, h_{T_x})$$

With attention mechanism

$$s_t = g_{dec}(s_{t-1}, y_{t-1}, c_t)$$
$$y_t = g_{out}(s_t, y_{t-1}, c_t)$$
$$c_t = q_t(h_1, h_2, \ldots, h_{T_x}, s_{t-1})$$

Different c_t's for different time points

The context vector is a weighted sum of the encoder hidden states

$$c_{T_y} = a_{T_y,1}h_1 + a_{T_y,2}h_2 + \ldots + a_{T_y,T_x}h_{T_x}$$

$$1 = a_{T_y,1} + a_{T_y,2} + \ldots + a_{T_y,T_x}$$
$$0 \le a_{T_y,i}$$

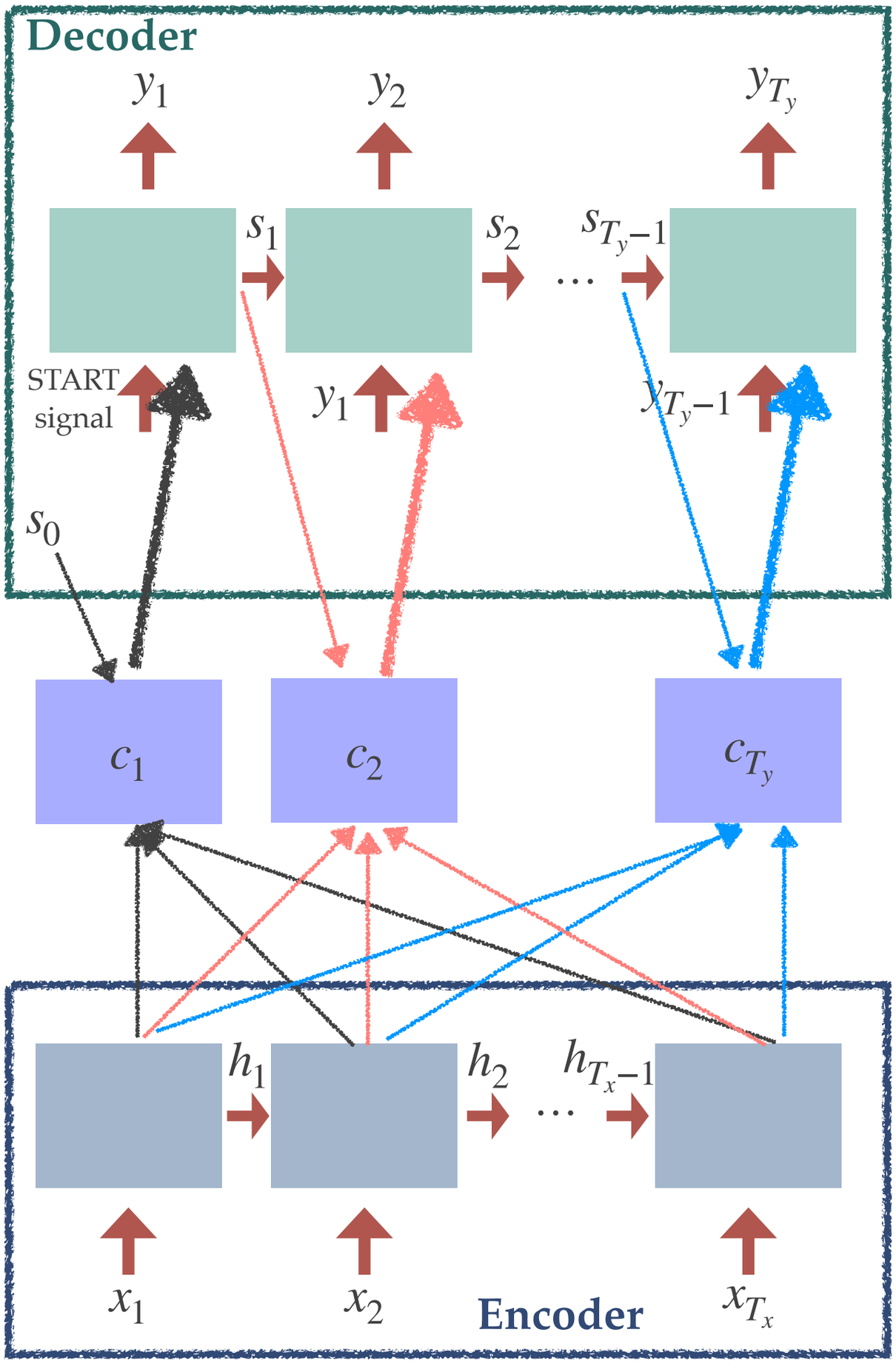
These constraints can be enforced using softmax

$$a_{T_y,i} = score(s_{T_y-1}, h_i)$$

A function that computes a "similarity" score between the two inputs (e.g., inner product)

$$h_t = g_{enc}(h_{t-1}, x_t)$$

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).



Without attention mechanism

$$s_t = g_{dec}(s_{t-1}, y_{t-1}, c)$$
$$y_t = g_{out}(s_t, y_{t-1}, c)$$
$$c = q(h_1, h_2, \ldots, h_{T_x})$$

With attention mechanism

$$s_t = g_{dec}(s_{t-1}, y_{t-1}, c_t)$$
$$y_t = g_{out}(s_t, y_{t-1}, c_t)$$
$$c_t = q_t(h_1, h_2, \ldots, h_{T_x}, s_{t-1})$$

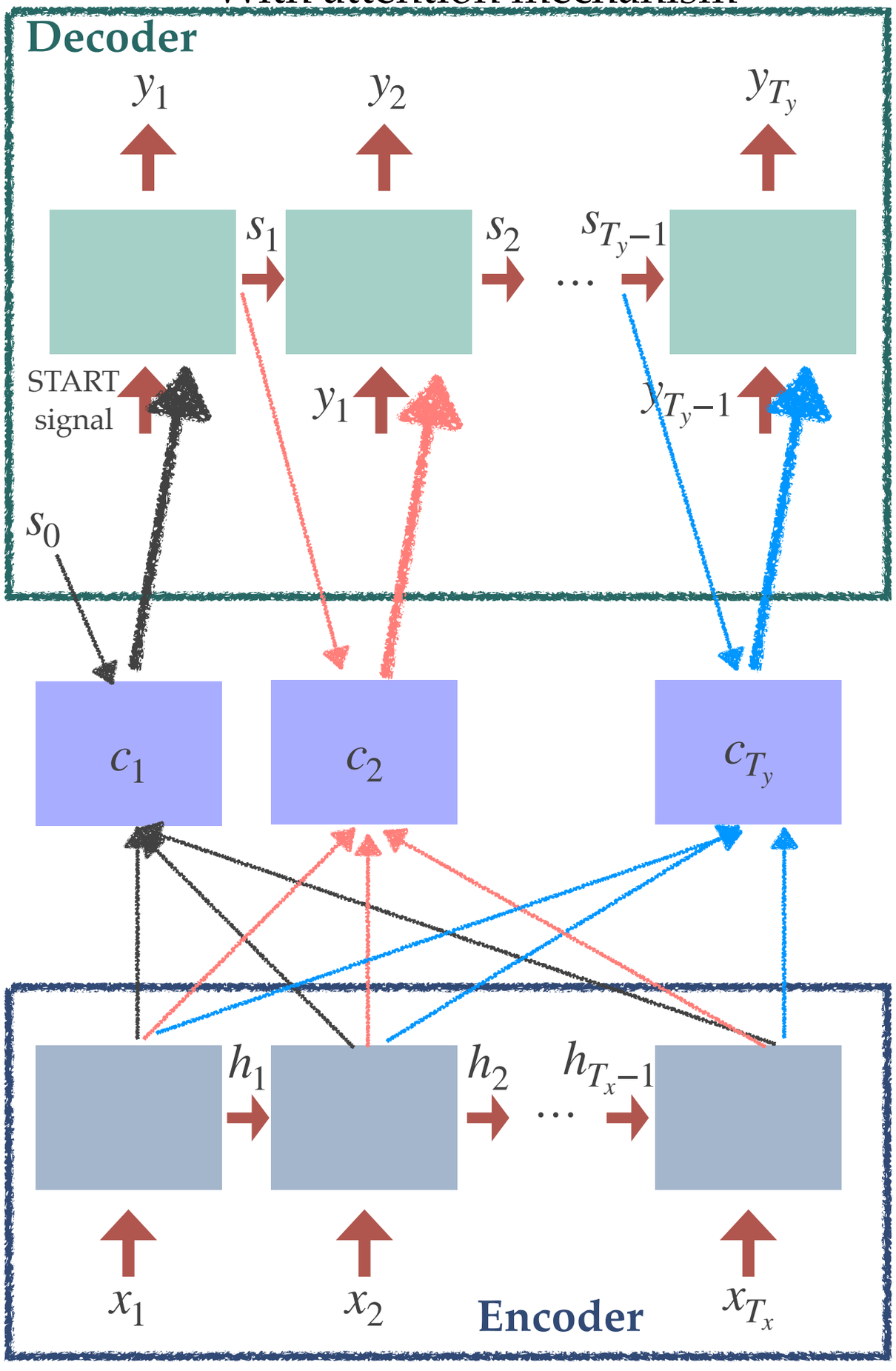
Different c_t's for different time points

The context vector is a weighted sum of the encoder hidden states

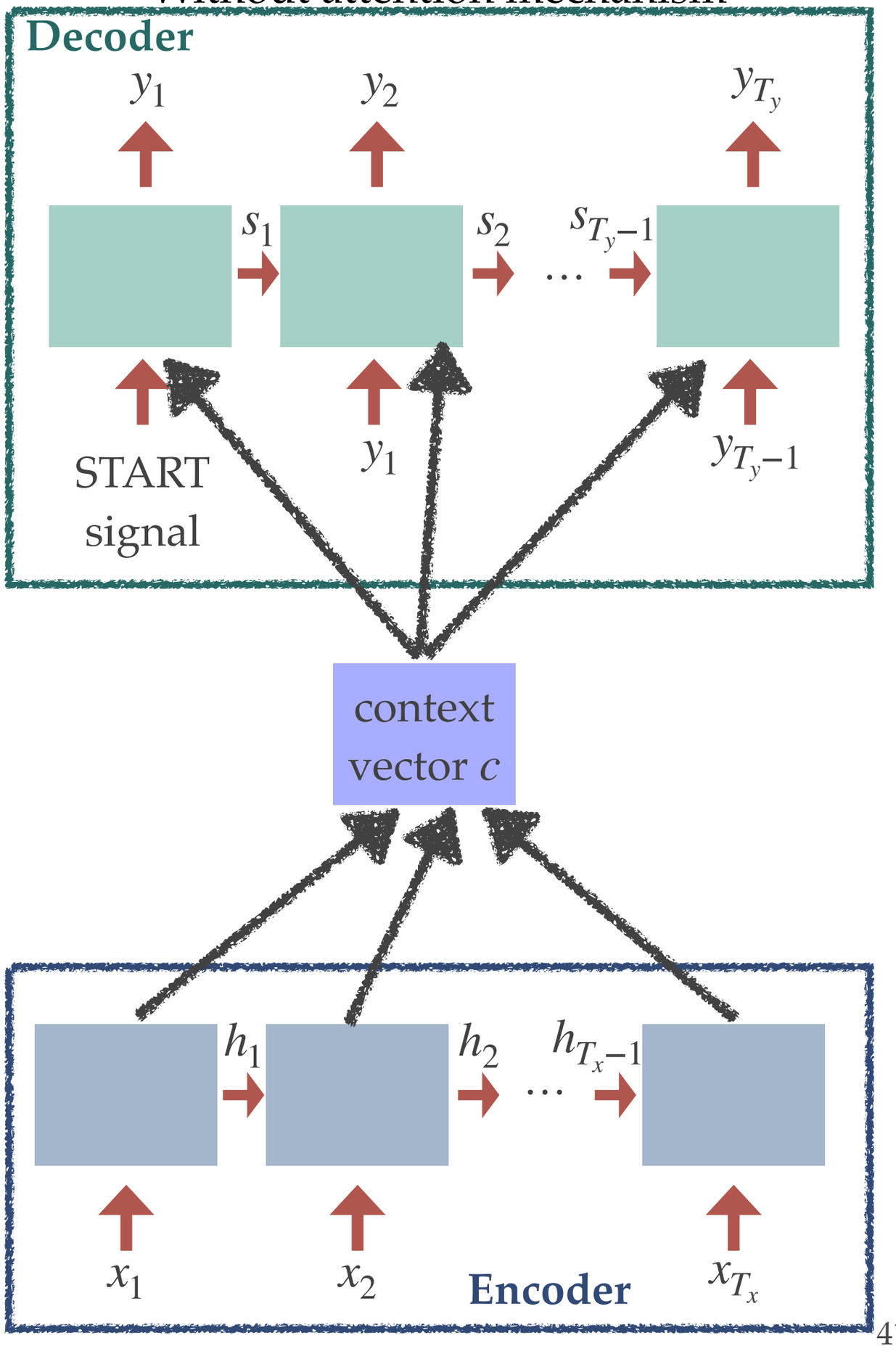
$$h_t = g_{enc}(h_{t-1}, x_t)$$

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).

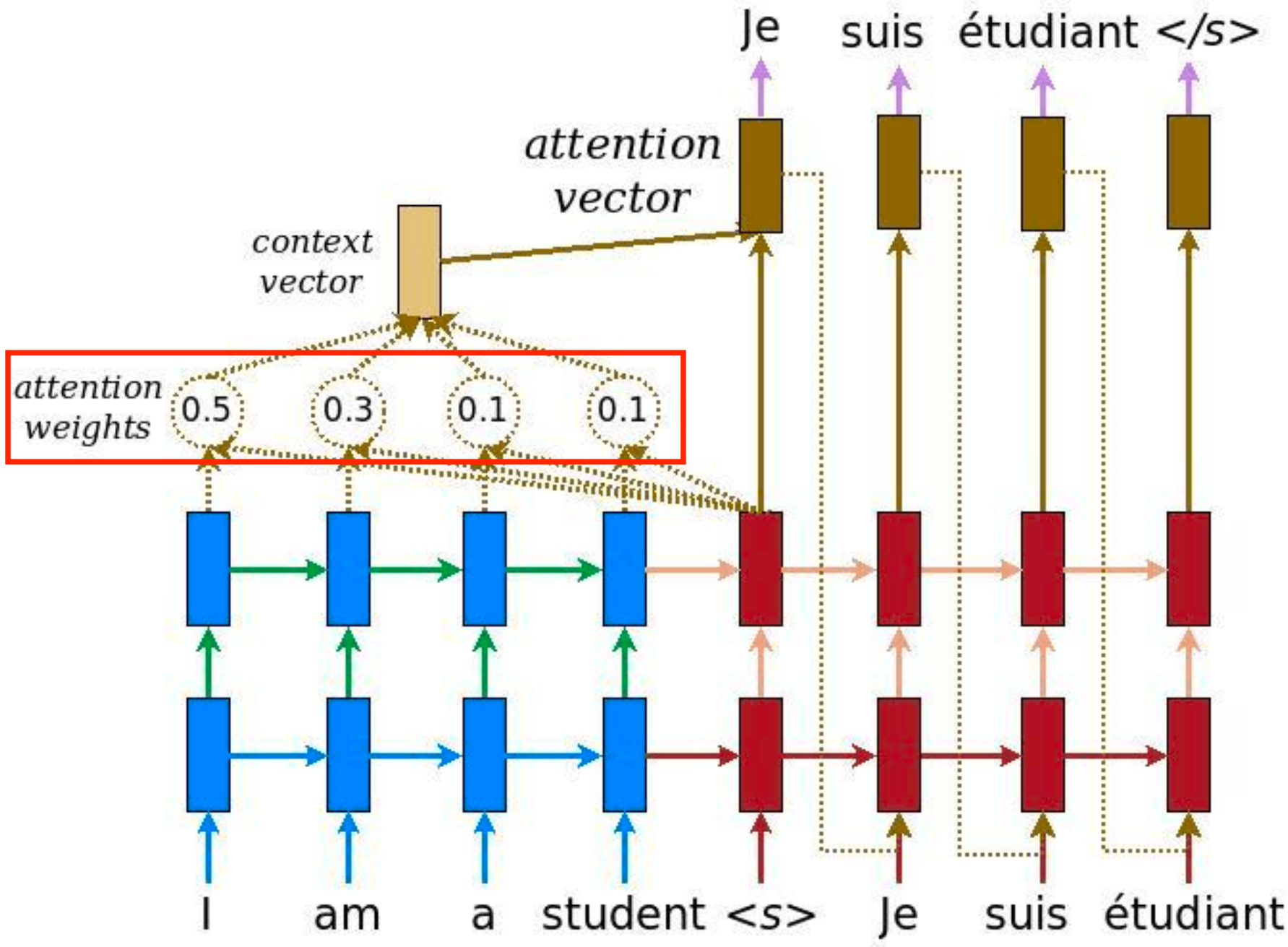
With attention mechanism



Without attention mechanism



Example: English to French with an Attention Mechanism



Tato, Ange, and Roger Nkambou. "Infusing Expert Knowledge Into a Deep Neural Network Using Attention Mechanism for Personalized Learning Environments." Frontiers in Artificial Intelligence (2022): 128.

Example: English to French with an Attention Mechanism

Example: English to French translation

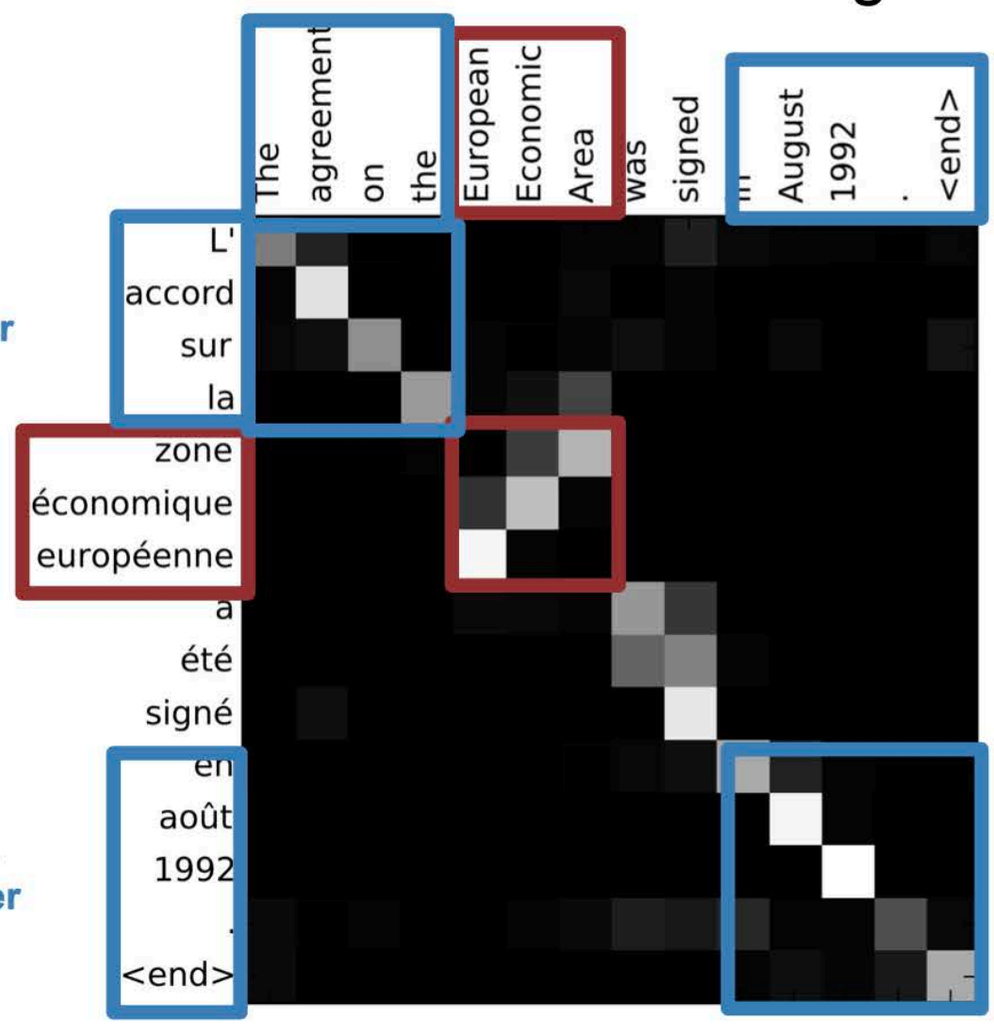
Input: "**The agreement on the European Economic Area** was signed **in August 1992**."

Output: "**L'accord sur la zone économique européenne** a été signé **en août 1992**."

Diagonal attention means words correspond in order

Attention figures out different word orders

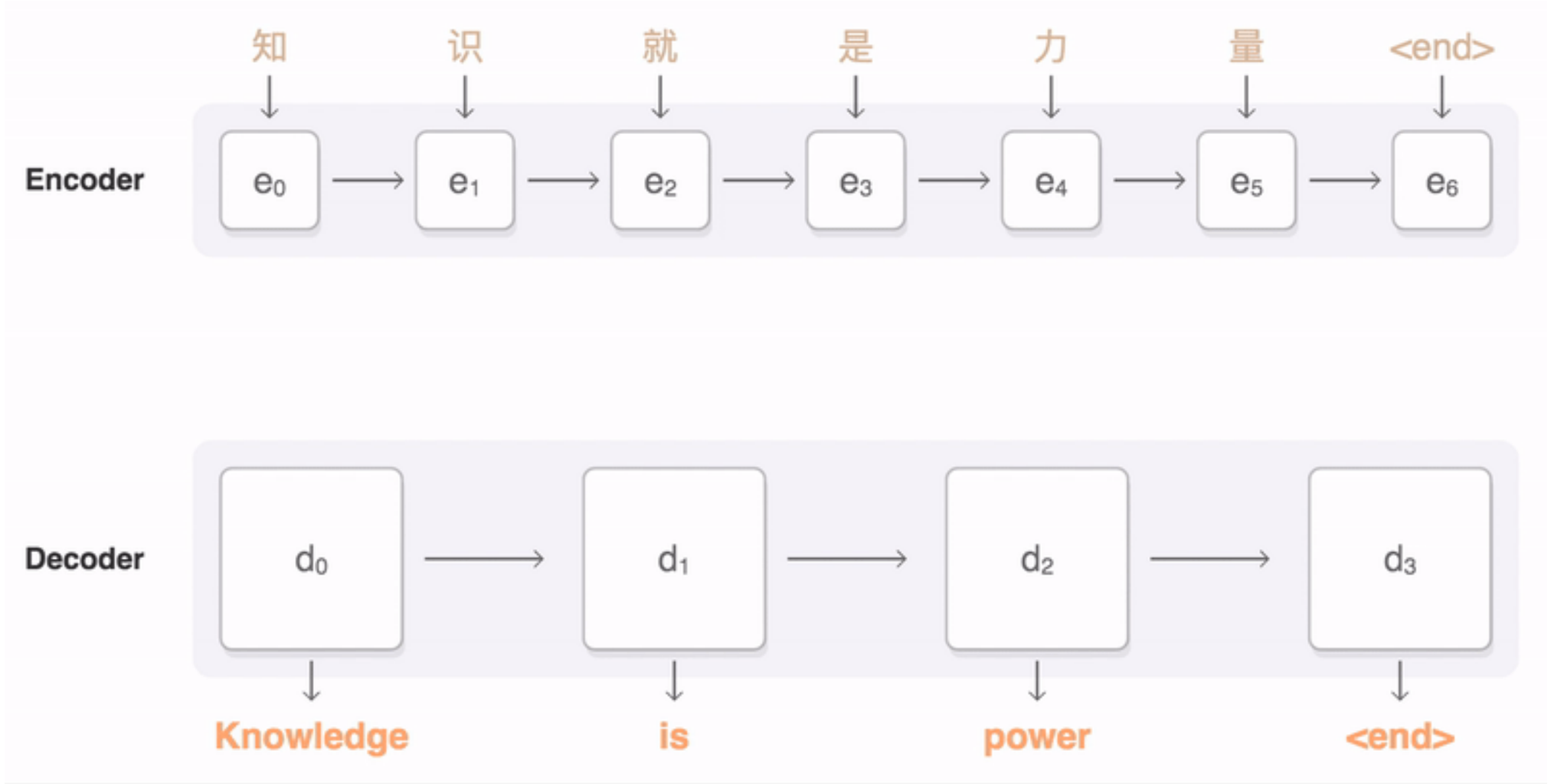
Diagonal attention means words correspond in order



Stanford CS231n (Spring 2022): Lecture 11.

Tato, Ange, and Roger Nkambou. "Infusing Expert Knowledge Into a Deep Neural Network Using Attention Mechanism for Personalized Learning Environments." Frontiers in Artificial Intelligence (2022): 128.

Example: Chinese to English with an Attention Mechanism



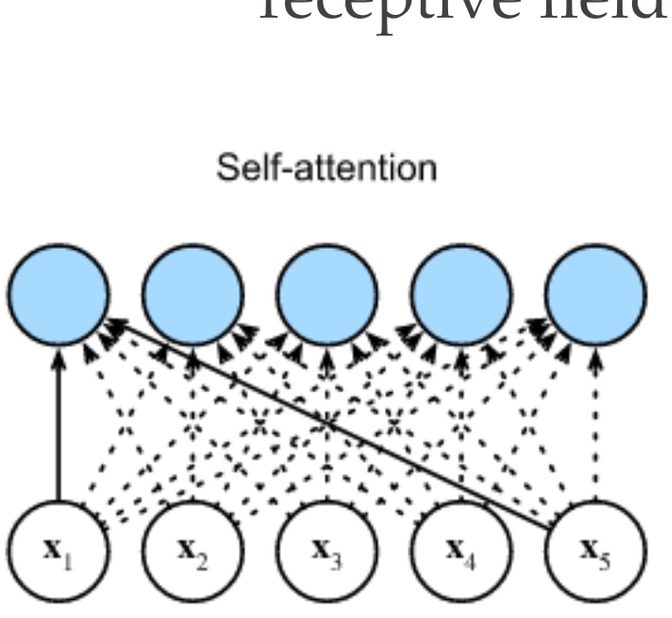
Google's tf-seq2seq

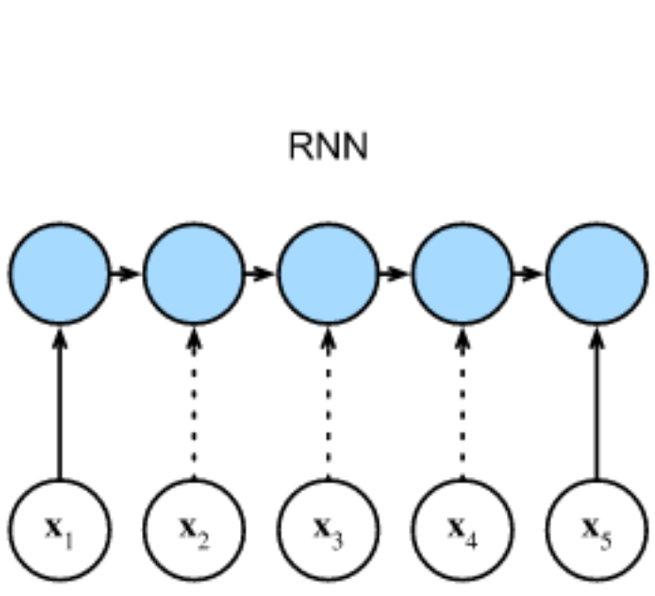
Outline

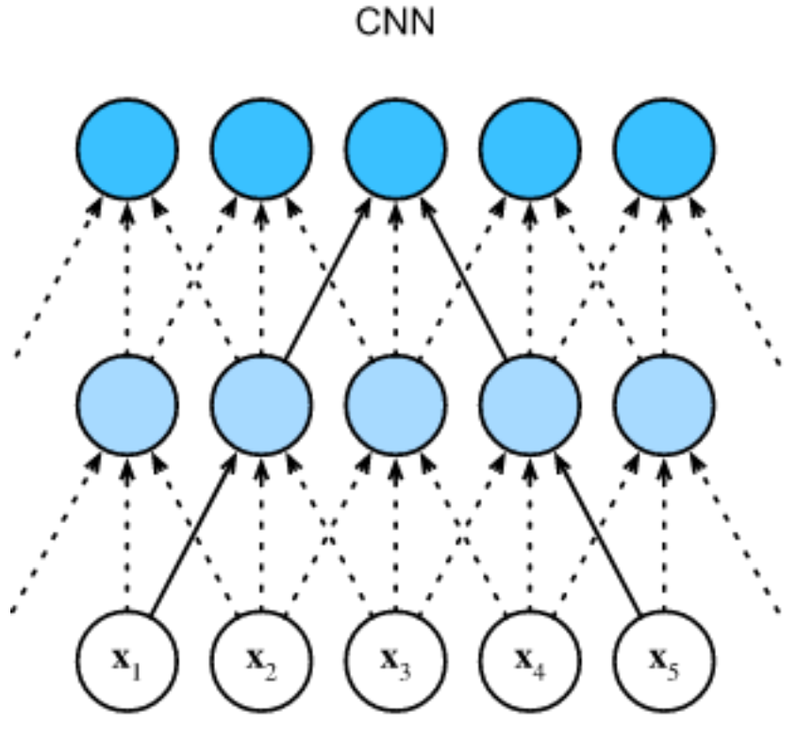
- Recurrent Neural Network (RNN)
- Sequence-to-Sequence Model
  - Attention Mechanism
- Self-Attention and Transformer
- Graph Neural Network

Attention vs RNN/CNN

- We have seen how the attention mechanism helps with neural machine translation
  - Allow for both short and long range dependencies
    - RNN has no explicit modeling of short-/long-range dependencies
    - CNN exploits short-range dependencies and the long-range dependencies can be captured by combining many layers sequentially to get larger receptive field







Dive Into Deep Learning: Self-Attention and Positional Encoding

Attention vs RNN/CNN

- We have seen how the attention mechanism helps with neural machine translation
  - Allow for both short and long range dependencies
    - RNN has no explicit modeling of short-/long-range dependencies
    - CNN exploits short-range dependencies and the long-range dependencies can be captured by combining many layers sequentially to get larger receptive field
  - Allow parallelization (per layer)
    - RNN relies on sequential computations (wait for previous time point)
    - CNN can also be parallelized
- We can use attention for data representation as well

Self-Attention

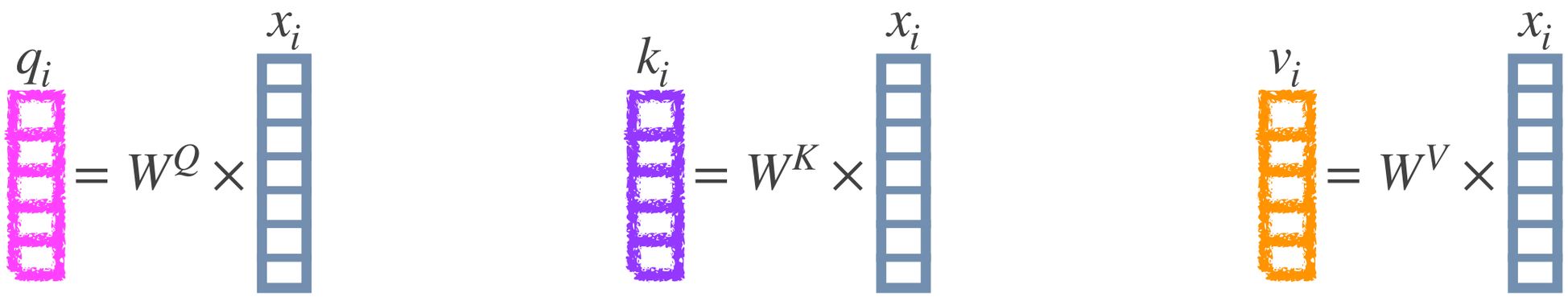
x_1

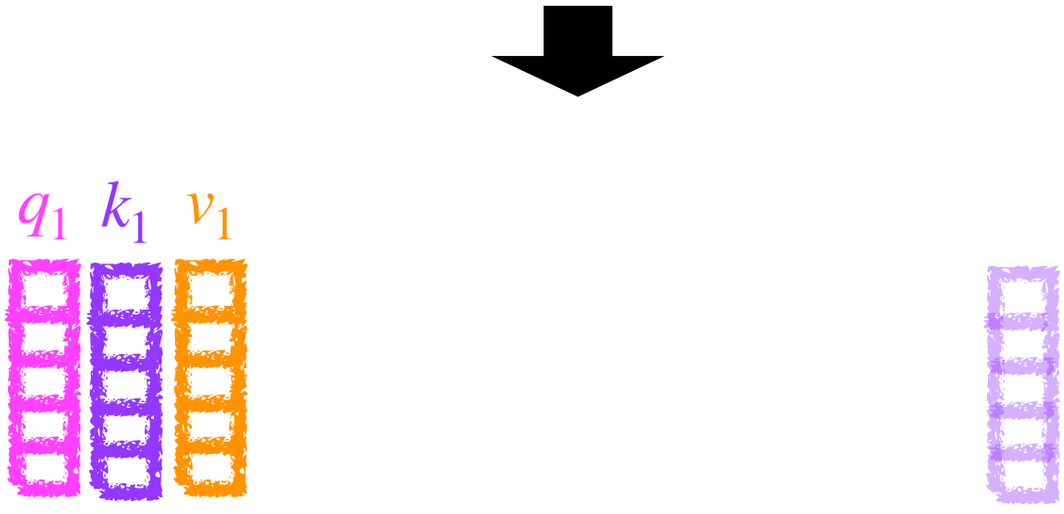
Self-Attention

For each input x_i $(i = 1,2,...,T_x)$

1. Calculate its query, key, and value

$$q_i = W^Q x_i \qquad k_i = W^K x_i \qquad v_i = W^V x_i$$





Self-Attention

For each input x_i $(i = 1,2,...,T_x)$

q_1 k_1 v_1

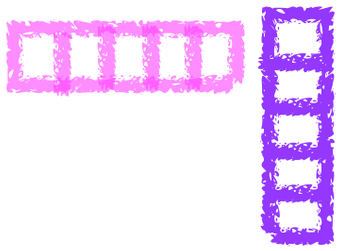
1. Calculate its query, key, and value

2. Calculate attention scores e's between a query and keys, and turn them into the corresponding attention distribution a using softmax

$$e_{ij} = q_i^T k_j$$

The inner product between the query and key
Just one way to compute an attention score

Self-Attention



Self-Attention

q_1 k_1 v_1

For each input x_i $(i = 1,2,...,T_x)$

1. Calculate its query, key, and value

2. Calculate attention scores e's between a query and keys, and turn them into the corresponding attention distribution a using softmax

$$e_{ij} = q_i^T k_j$$

3. Compute the weighted sum of the values

$$h_i = \sum_j a_{ij} v_j$$

Self-Attention

For each input x_i $(i = 1,2,...,T_x)$

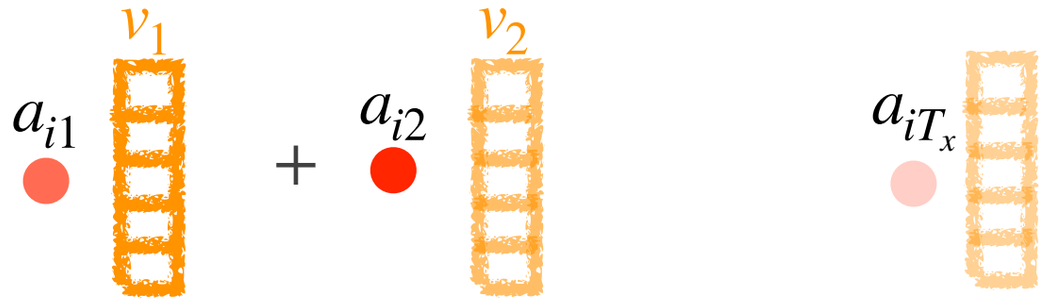
1. Calculate its query, key, and value

2. Calculate attention scores e's between a query and keys, and turn them into the corresponding attention distribution a using softmax

$$e_{ij} = q_i^T k_j$$

3. Compute the weighted sum of the values

$$h_i = \sum_j a_{ij} v_j$$



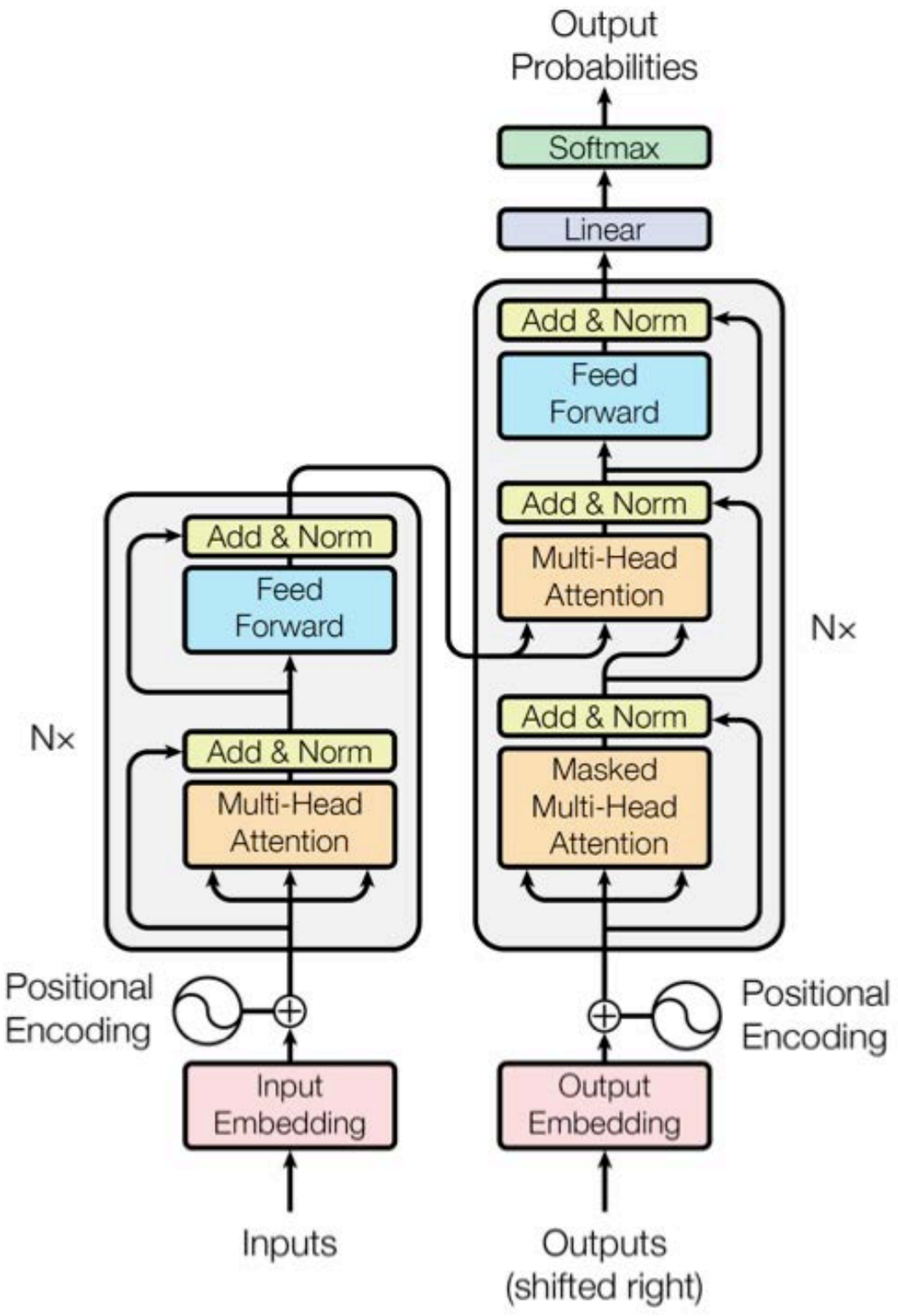
Self-Attention

x_1

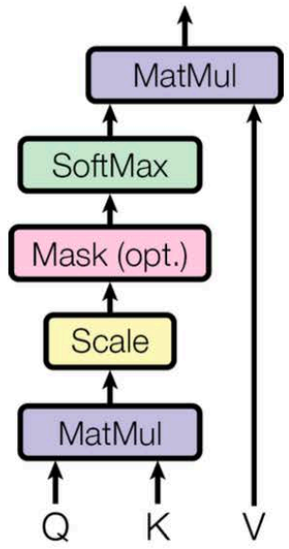
Self-Attention

x_1

Transformer



Scaled Dot-Product Attention

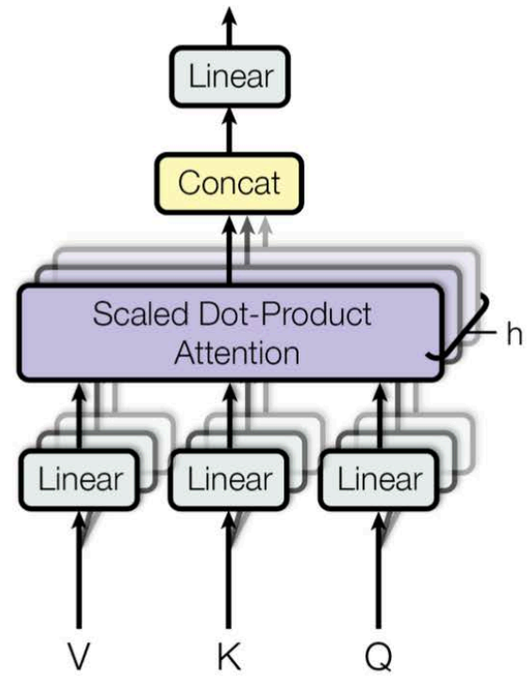


$$softmax(QK^T)V$$

$$\downarrow$$

$$softmax(QK^T/\sqrt{d_k})V$$

Multi-Head Attention



Each attention head performs attention independently

Feed Forward Add more expressive power

Add & Norm Add residual connections & Normalize the data within each layer

Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

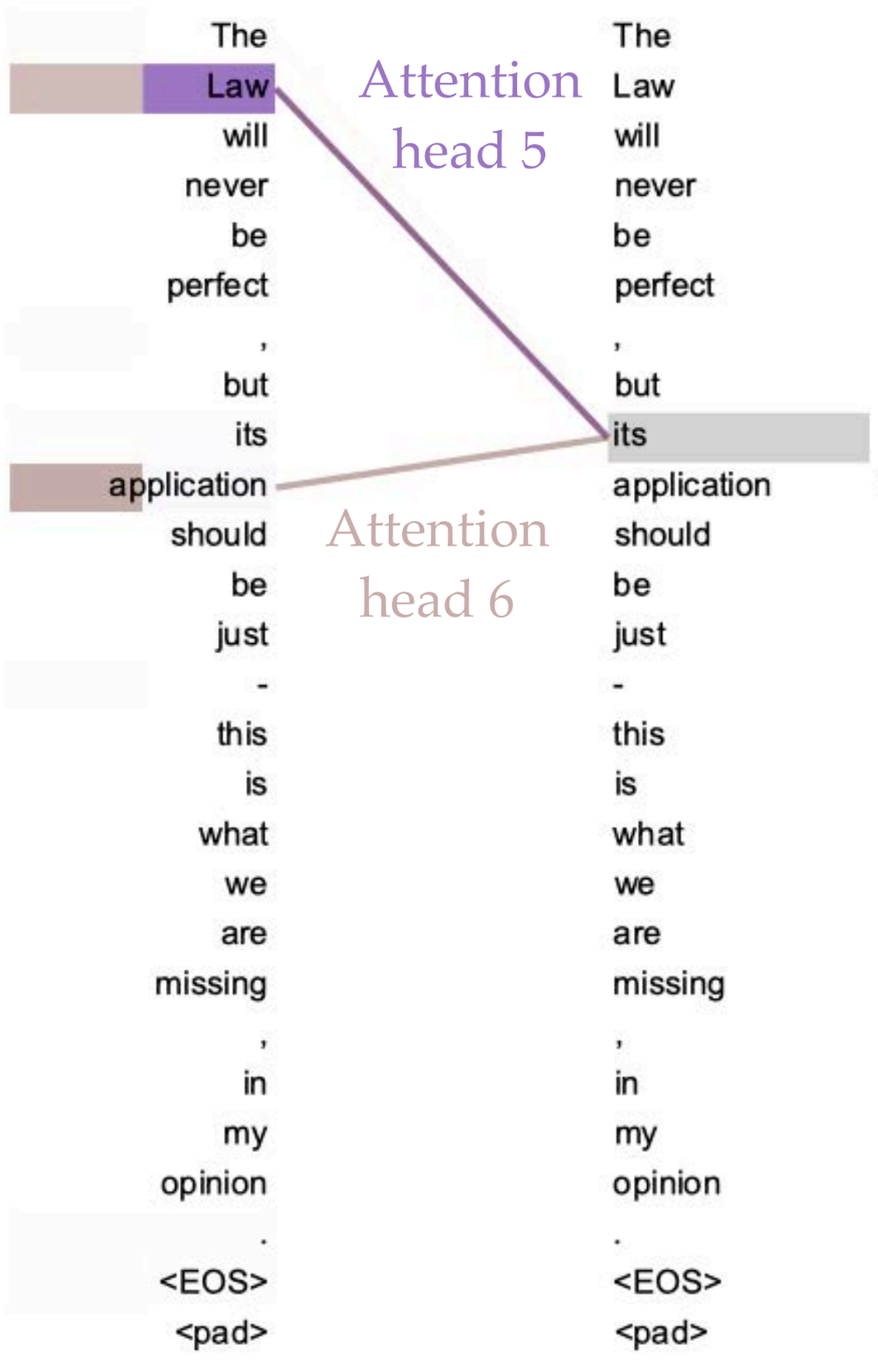
Transformer

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

Transformer

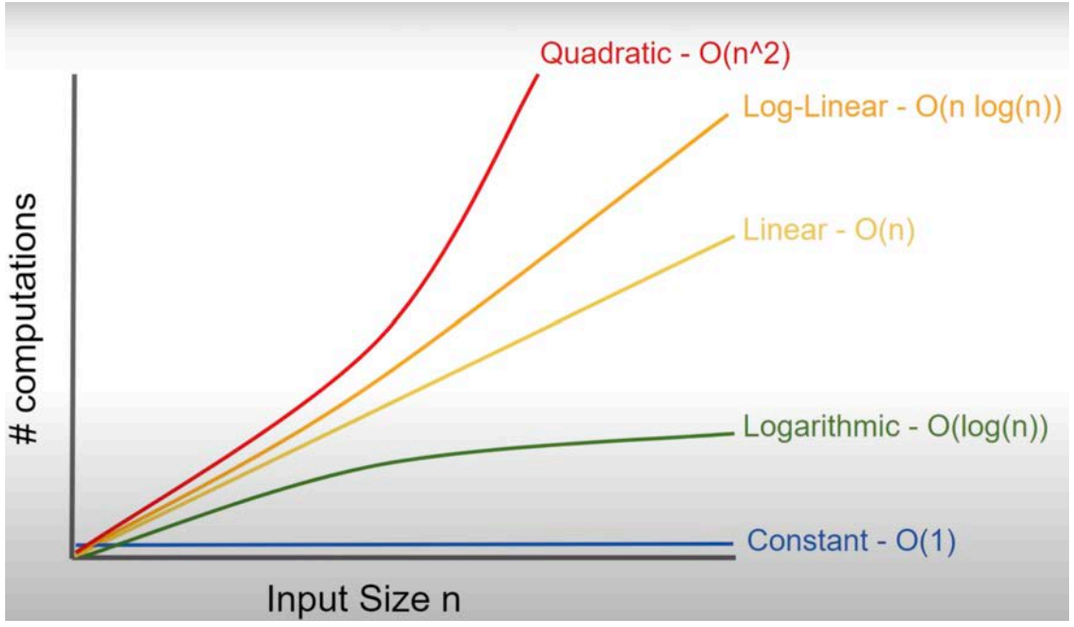


Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

Transformer

- The computation of self-attention grows quadratically with the sequence length
  - For RNNs, the computation grows linearly
  - Recent models can alleviate the problem
    - Set Transformer (2019)
    - Linformer (2020)
    - Big Bird (2020)
    - Perceiver (2021)/Perceiver IO (2021)

Algorithms Explained: Computational Complexity



- Stanford CS224n (Winter 2022): Lecture 9.
- Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).
- Lee, Juho, et al. "Set transformer: A framework for attention-based permutation-invariant neural networks." International conference on machine learning. PMLR, 2019.
- Wang, Sinong, et al. "Linformer: Self-attention with linear complexity." arXiv preprint arXiv:2006.04768 (2020).
- Zaheer, Manzil, et al. "Big bird: Transformers for longer sequences." Advances in Neural Information Processing Systems 33 (2020): 17283-17297.
- Jaegle, Andrew, et al. "Perceiver io: A general architecture for structured inputs & outputs." arXiv preprint arXiv:2107.14795 (2021).

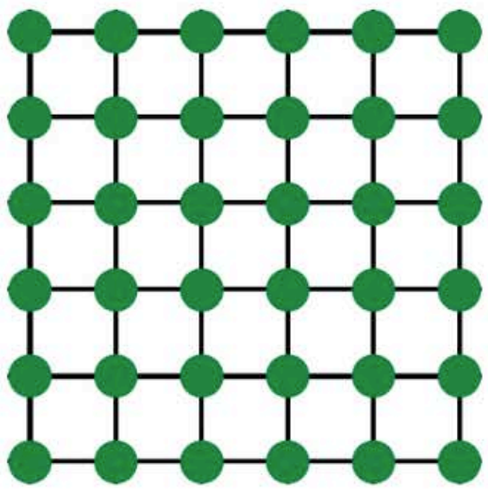
SYSTEM PROMPT (HUMAN-WRITTEN)	*In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.*
MODEL COMPLETION (MACHINE-WRITTEN, 10 TRIES)	The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science. Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved. Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow. Pérez and the others then ventured further into the valley. "By the time we reached the top of one peak, the water looked blue, with some crystals on top," said Pérez. Pérez and his friends were astonished to see the unicorn herd. These creatures could be seen from the air without having to move too much to see them – they were so close they could touch their horns. While examining these bizarre creatures the scientists discovered that the creatures also spoke some fairly regular English. Pérez stated, "We can see, for example, that they have a common 'language,' something like a dialect or dialectic."

GPT-2: A large transformer-based language model with 1.5 billion parameters

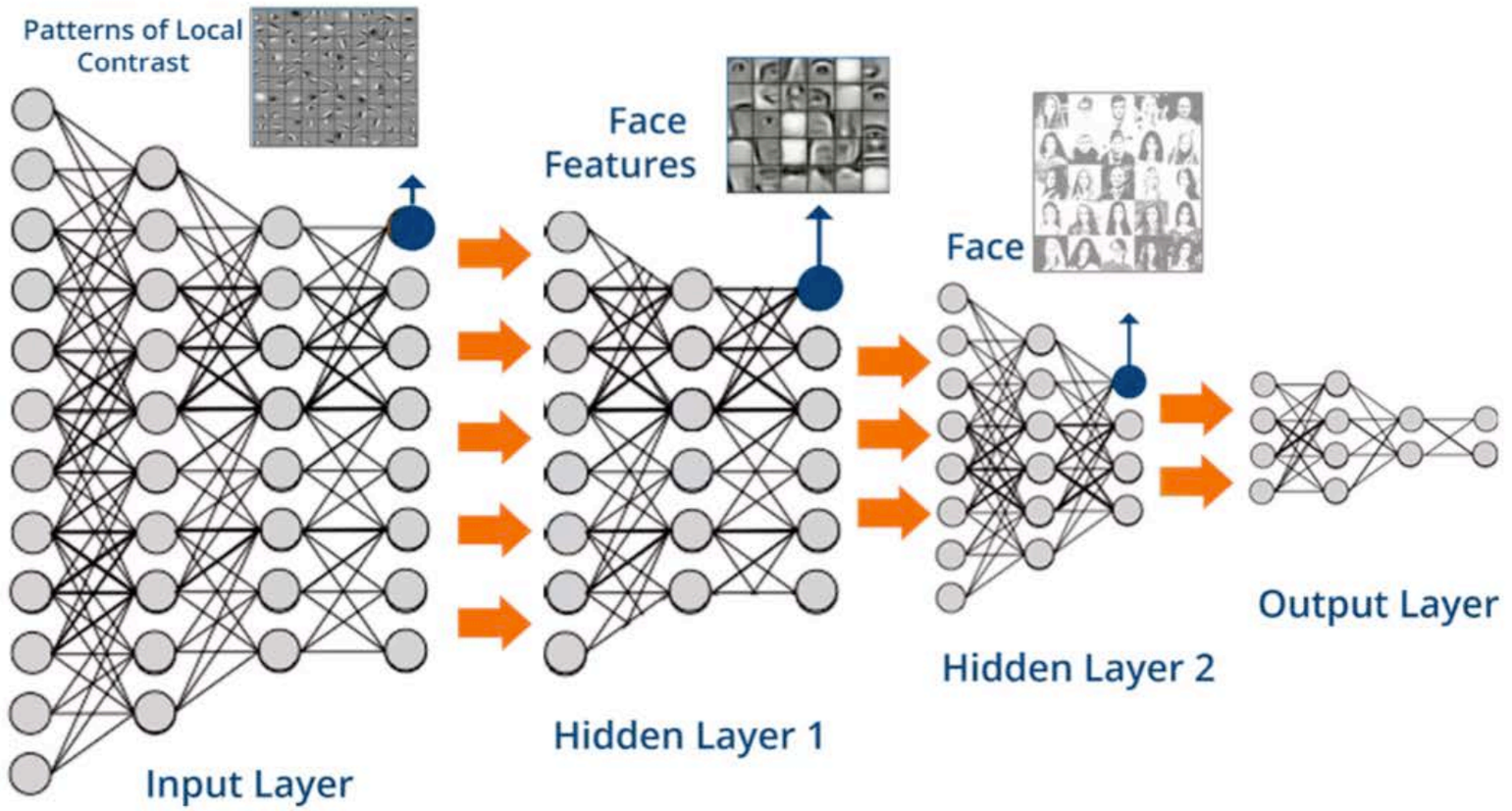
Outline

- Recurrent Neural Network (RNN)
- Sequence-to-Sequence Model
  - Attention Mechanism
- Self-Attention and Transformer
- Graph Neural Network

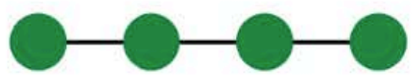
Graphs

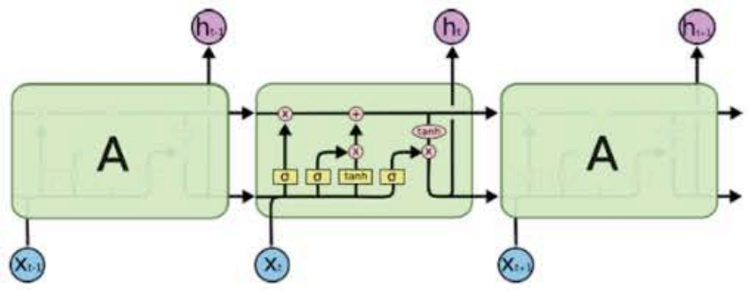


Image



Sequence

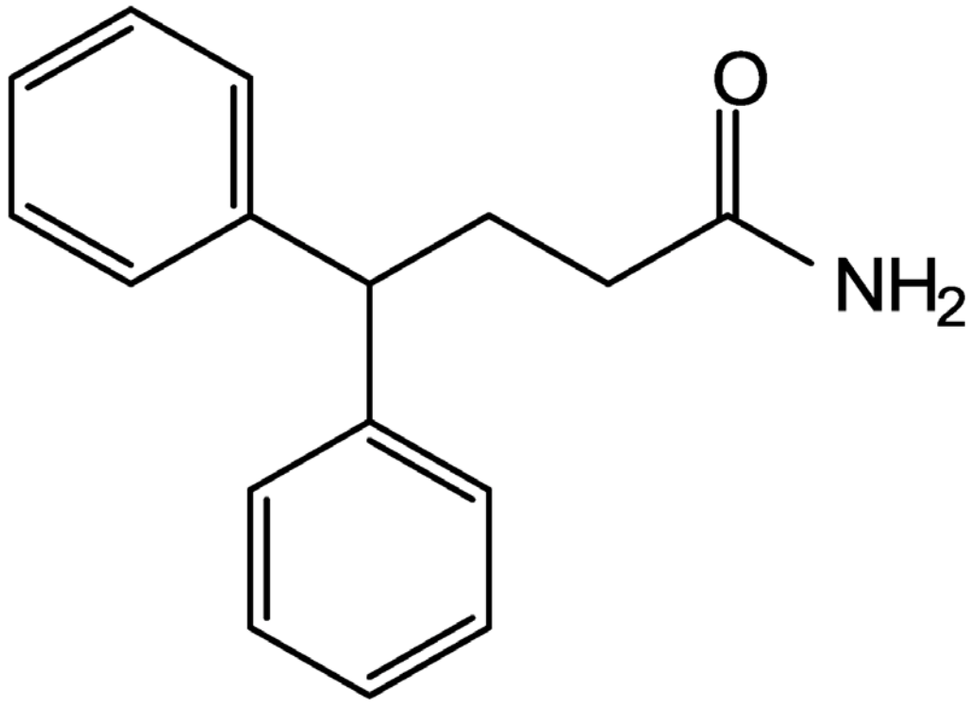




Graphs

3,3-Diphenylpropylamine

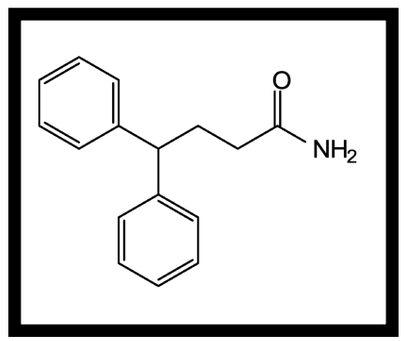
$C_{15}H_{17}N$

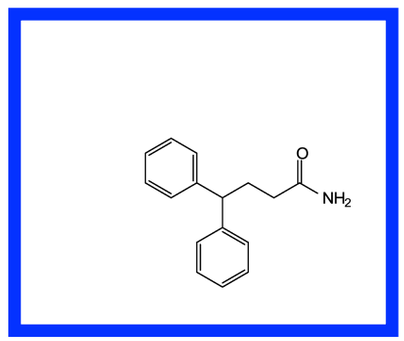


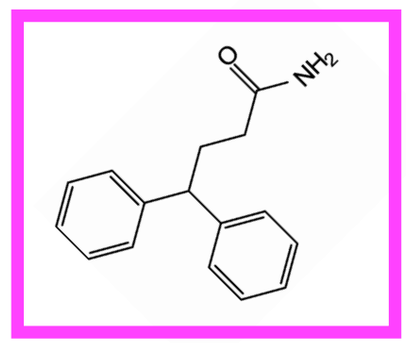
How should we process this type of data?

Can we treat it as image and use a CNN to analyze it?

If we zoom / translate / shift the molecule, will our CNN understand that? More importantly, rotate?







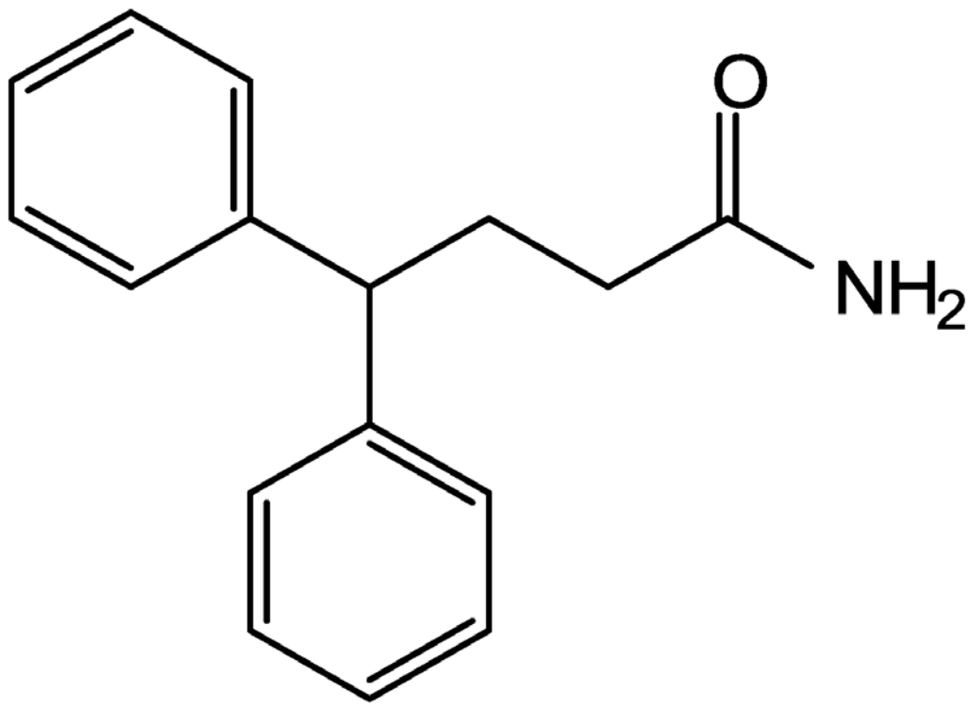
Vogt, Martin, Anne Mai Wassermann, and Jürgen Bajorath. "Application of information—Theoretic concepts in chemoinformatics." Information 1.2 (2010): 60-73.

keywords: Simplified molecular-input line-entry system (SMILES)

Graphs

3,3-Diphenylpropylamine

$C_{15}H_{17}N$



How should we process this type of data?

Can we treat it as a string of characters and use a RNN to analyze it?

If we swap the order of the characters, will our model understand it?

CCC…CHH…HN

NCCC…CHH…HN

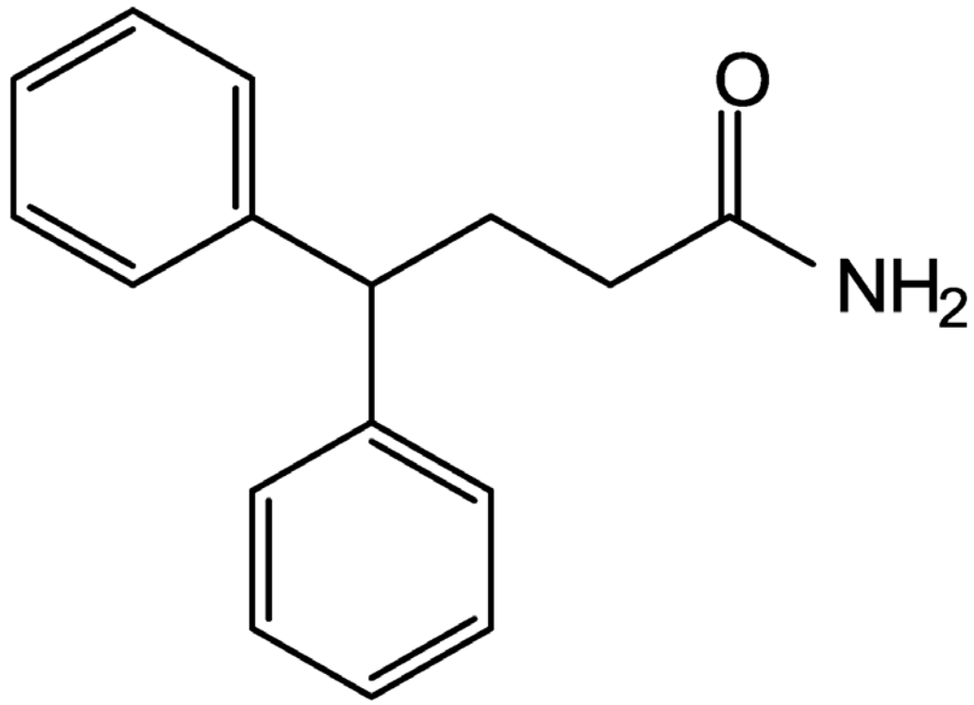
NCHCH…CHHH

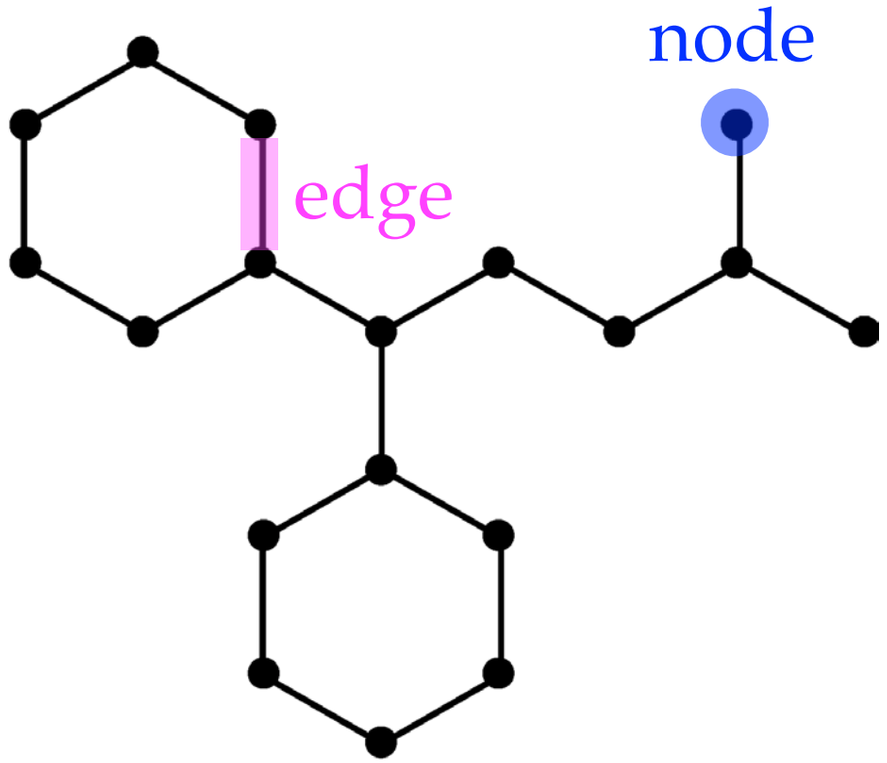
Vogt, Martin, Anne Mai Wassermann, and Jürgen Bajorath. "Application of information —Theoretic concepts in chemoinformatics." Information 1.2 (2010): 60-73.

Graphs

3,3-Diphenylpropylamine

$C_{15}H_{17}N$





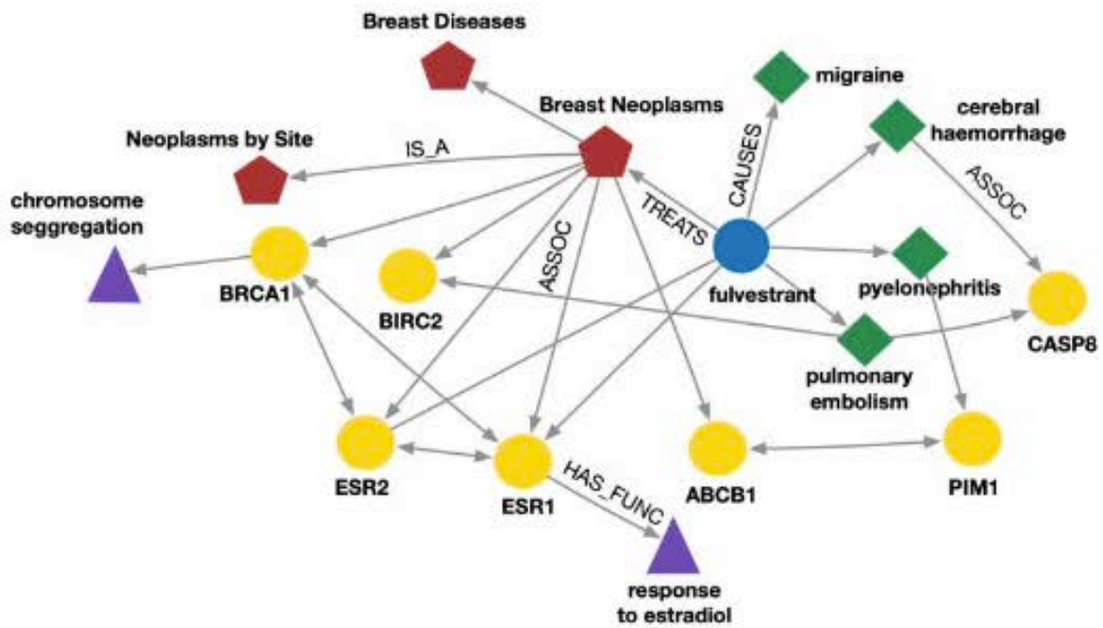
Graph

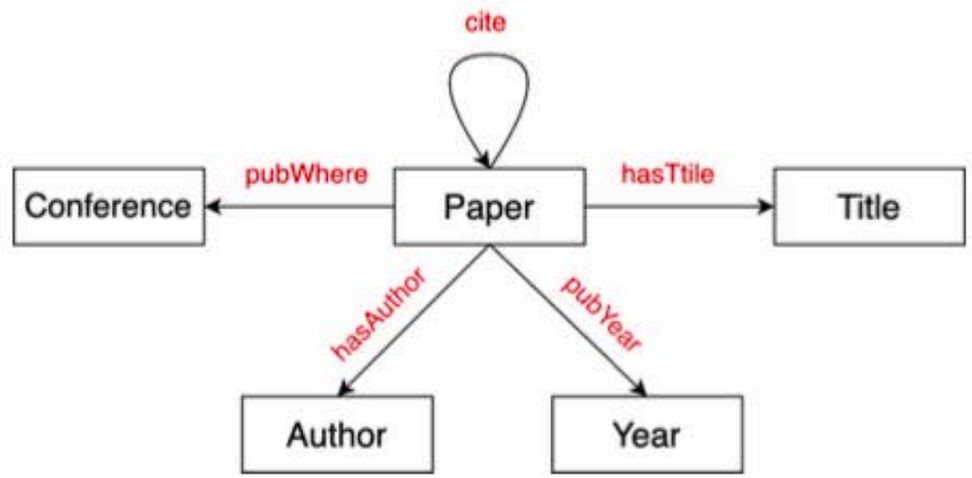
nodes: atoms / molecules

edges: chemical bonds

Vogt, Martin, Anne Mai Wassermann, and Jürgen Bajorath. "Application of information—Theoretic concepts in chemoinformatics." Information 1.2 (2010): 60-73.

Examples: Knowledge/Academic Graphs





Biomedical Knowledge Graphs

Example node: Migraine
Example edge: (fulvestrant, Treats, Breast Neoplasms)
Example node type: Protein
Example edge type (relation): Causes

Academic Graphs

Example node: ICML
Example edge: (GraphSAGE, NeurIPS)
Example node type: Author
Example edge type (relation): pubYear

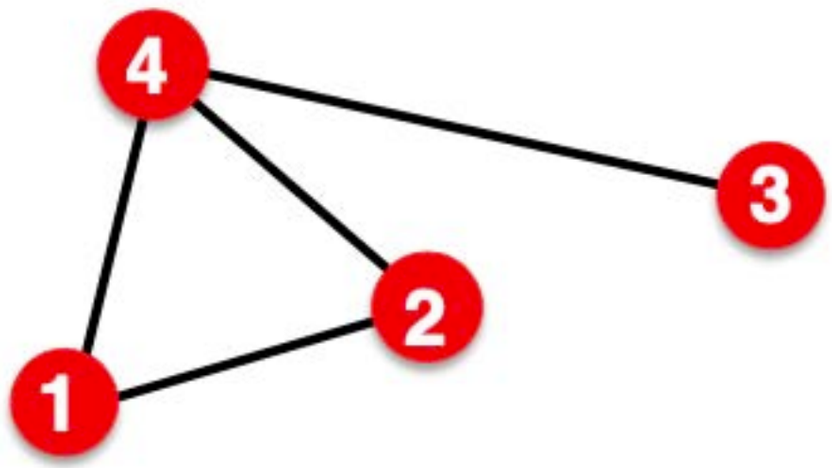
Example: Physical Simulation

nodes: particles

edges: interaction between particles

Sanchez-Gonzalez, Alvaro, et al. "Learning to simulate complex physics with graph networks." International Conference on Machine Learning. PMLR, 2020.

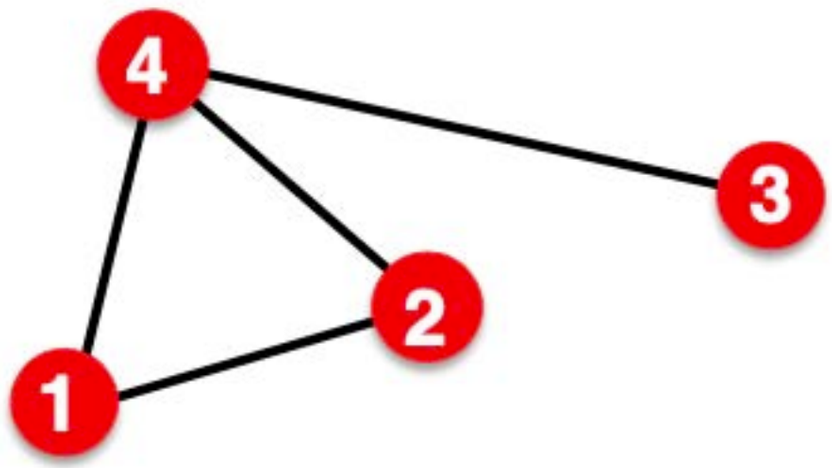
Graphs: Representation



Adjacency matrix, A

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

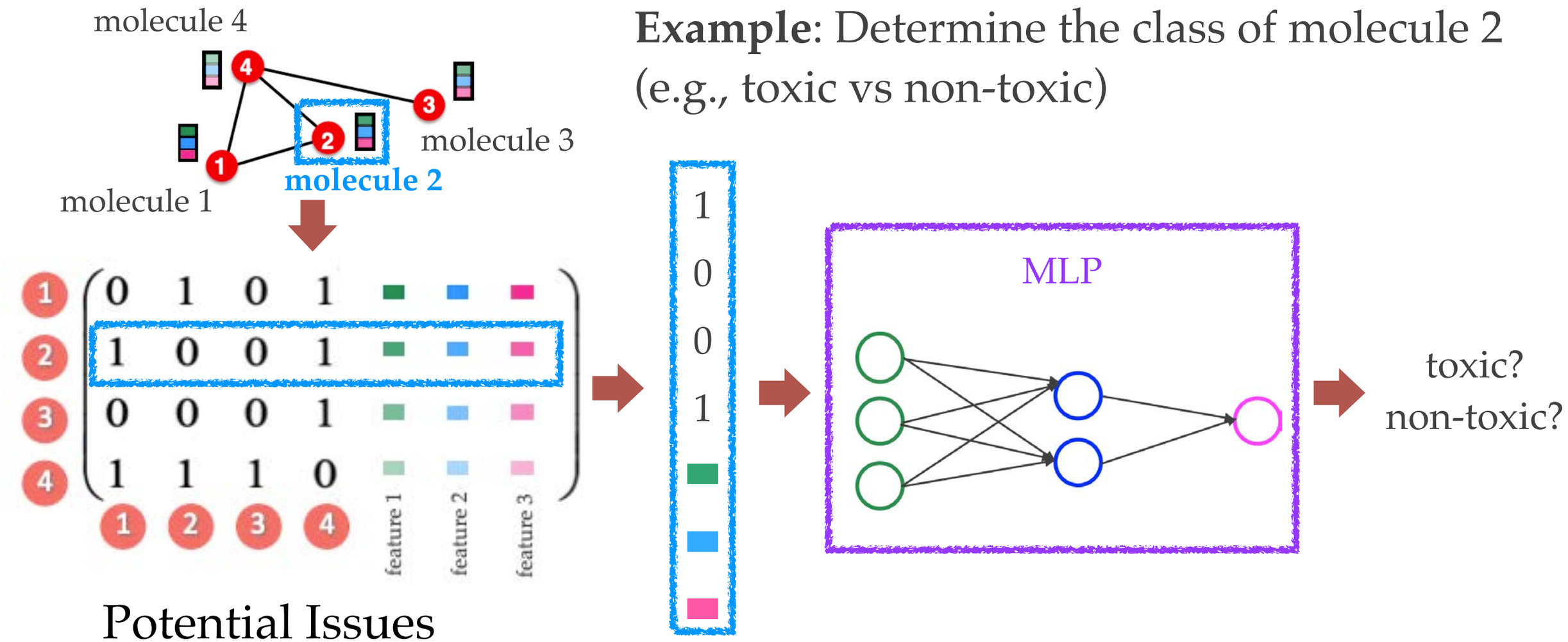
Graphs: Representation



Adjacency matrix, A

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Node-Level Classification

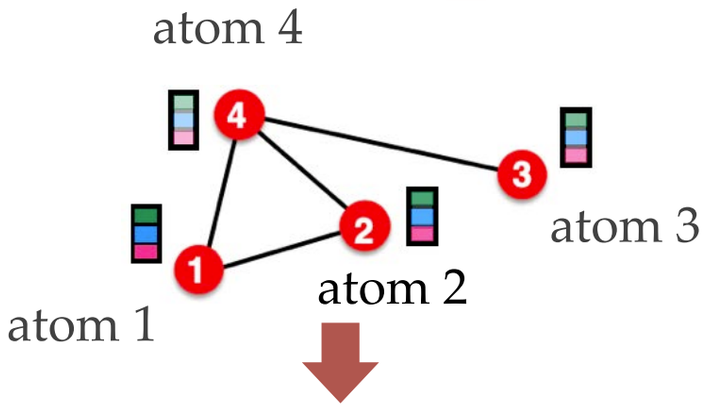


Example: Determine the class of molecule 2 (e.g., toxic vs non-toxic)

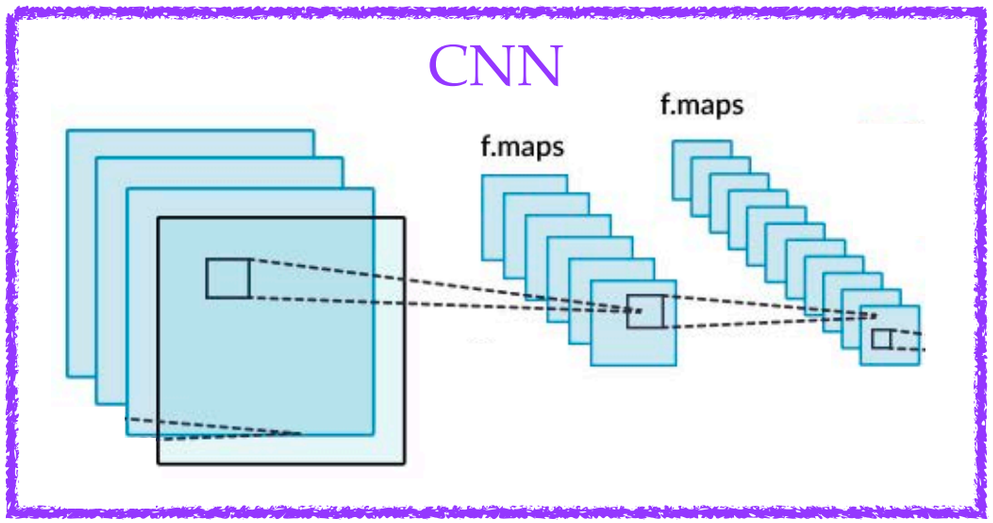
Potential Issues

- Once trained, it is not applicable to graphs of different sizes
- It is sensitive to node ordering
- It becomes computationally intensive when we have lots of nodes

Graph-Level Classification



Example: Determine if the molecule is a potential drug



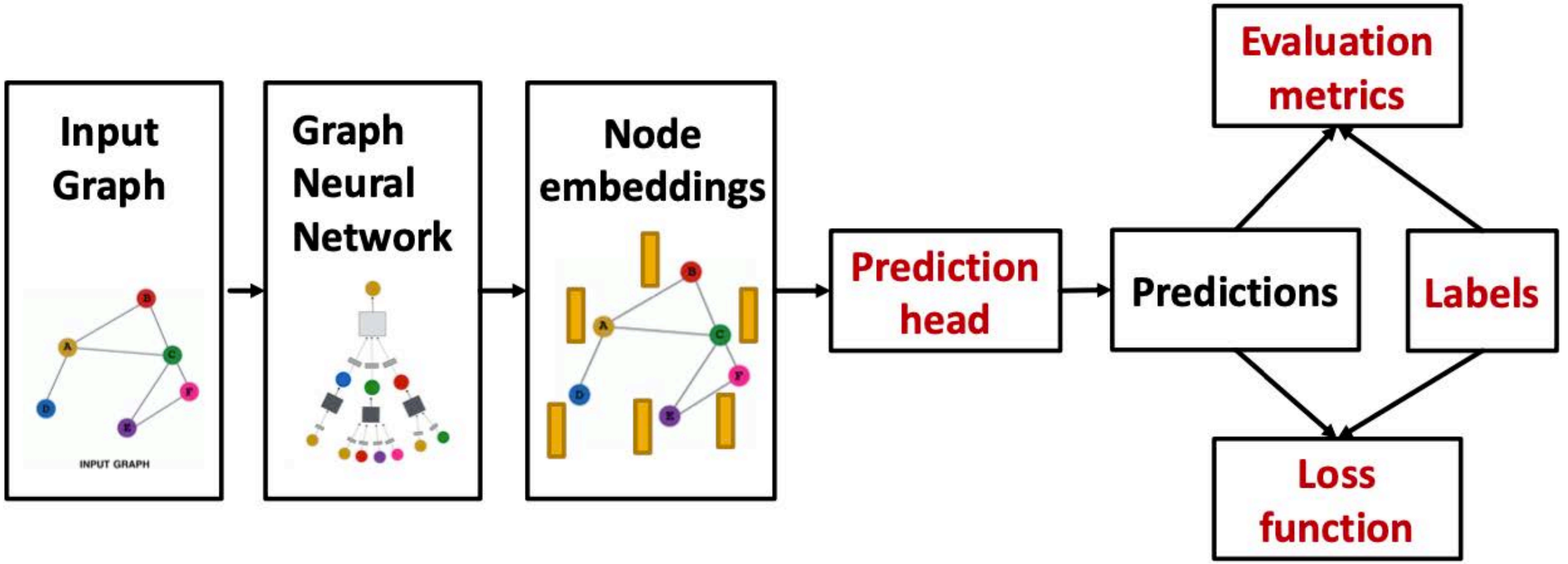
drug?
non-drug?

Potential Issues

- It is sensitive to node ordering
- It becomes computationally intensive when we have lots of nodes
- Does it really make sense to apply 2D convolution to the extended adjacency matrix?

Machine Learning on Graphs

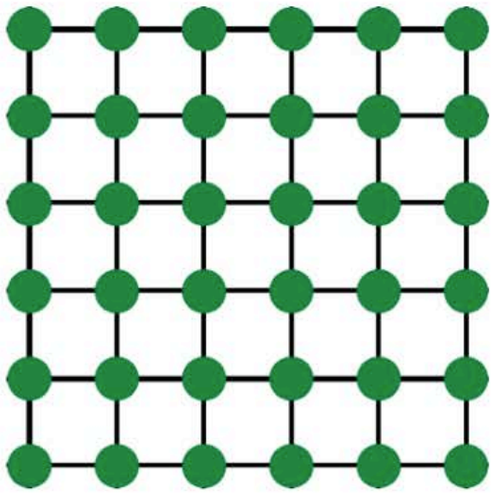
- Consider the node-level classification task as an example



- Obtaining a good node embedding ("features") for each node that takes into account not only its own information, but also other nodes' information will be beneficial for this task

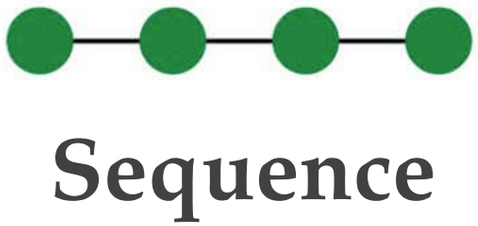
  Idea: In most graphs, similar nodes are likely to connect to each other, so we can come up with operations that exploit local information (small neighborhood in a graph)

Machine Learning on Graphs



Image

CNN



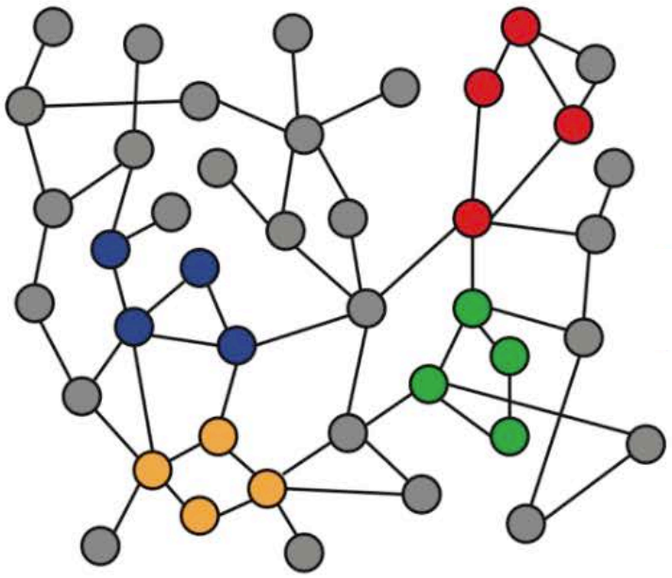
Sequence

1D CNN

MLP RNN

LSTM GRU

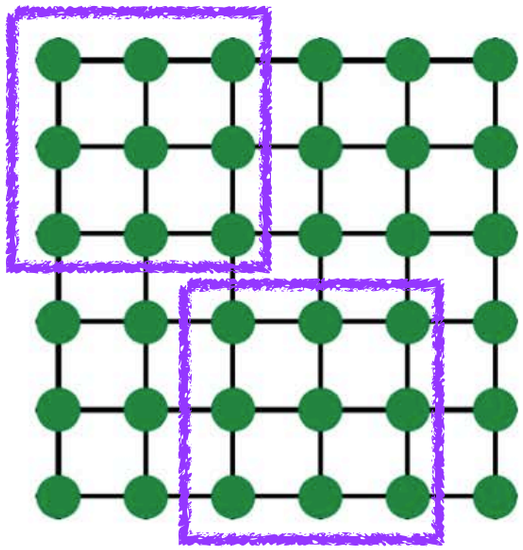
Transformer



Graph

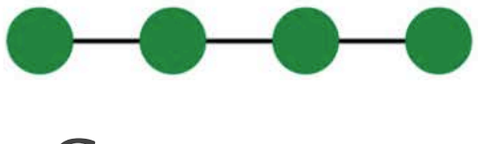
Machine Learning on Graphs

Will convolution work?



Image

CNN



Sequence

1D CNN

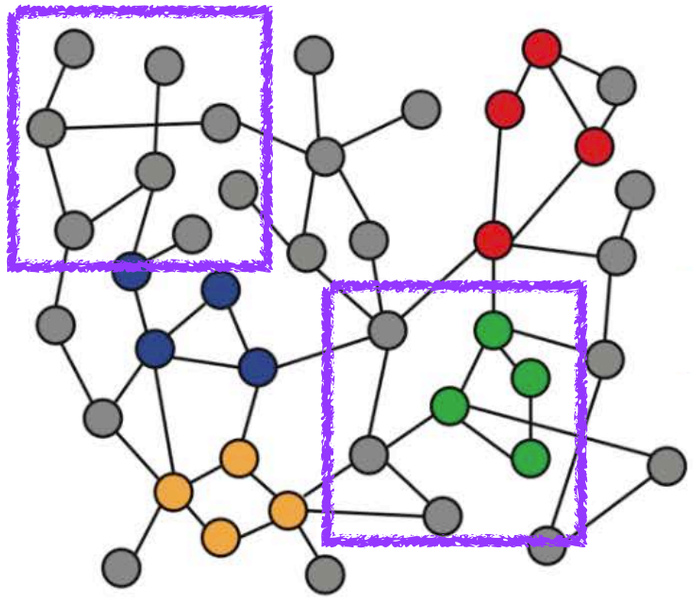
MLP RNN

LSTM GRU

Transformer

No spatial locality like grids

No fixed node ordering



Graph

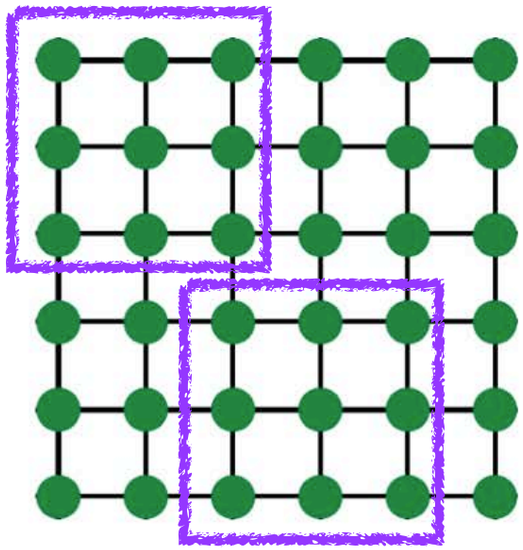
Graph Neural Network

Machine Learning on Graphs

Will convolution work?

No spatial locality like grids

No fixed node ordering



Image

CNN

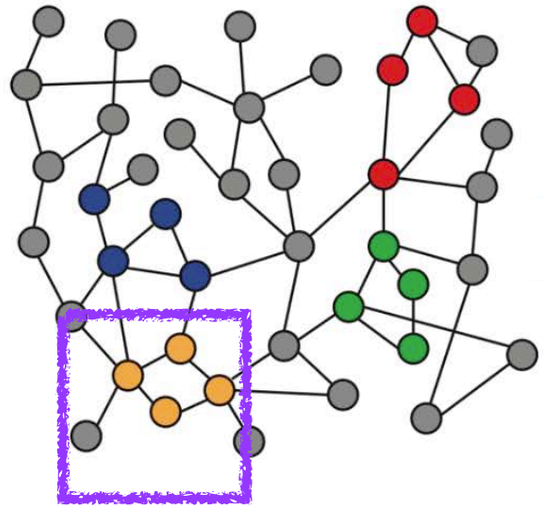
Sequence

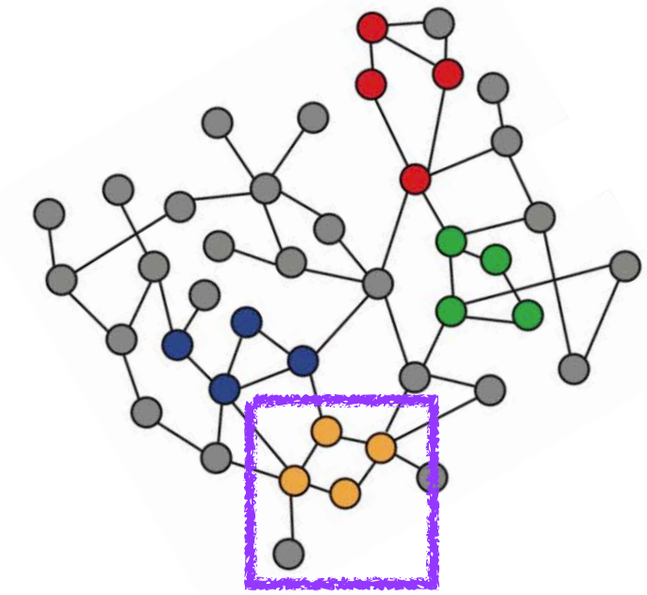
1D CNN

MLP RNN

LSTM GRU

Transformer



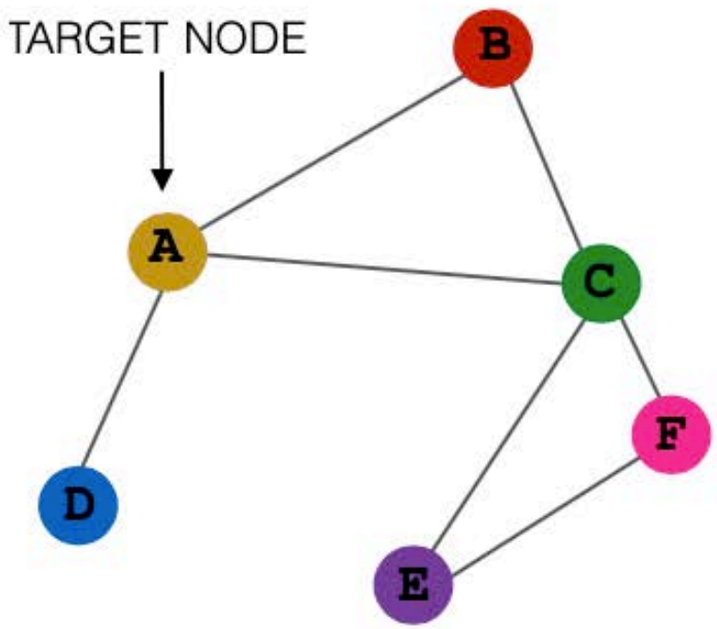


Graph

Graph Neural Network

Graph Neural Network (GNN)

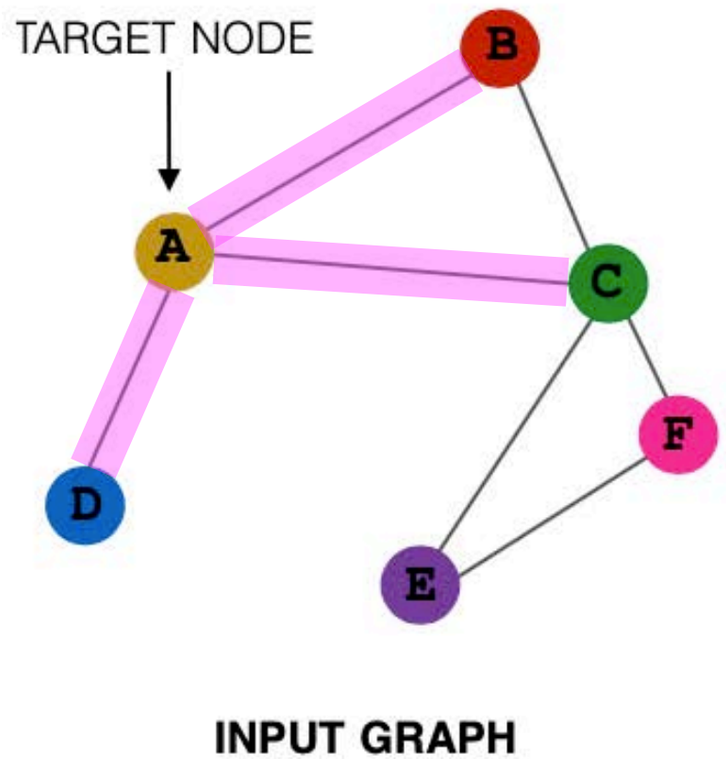
Idea: For each node, generate its node embedding ("features") based on its local neighborhood



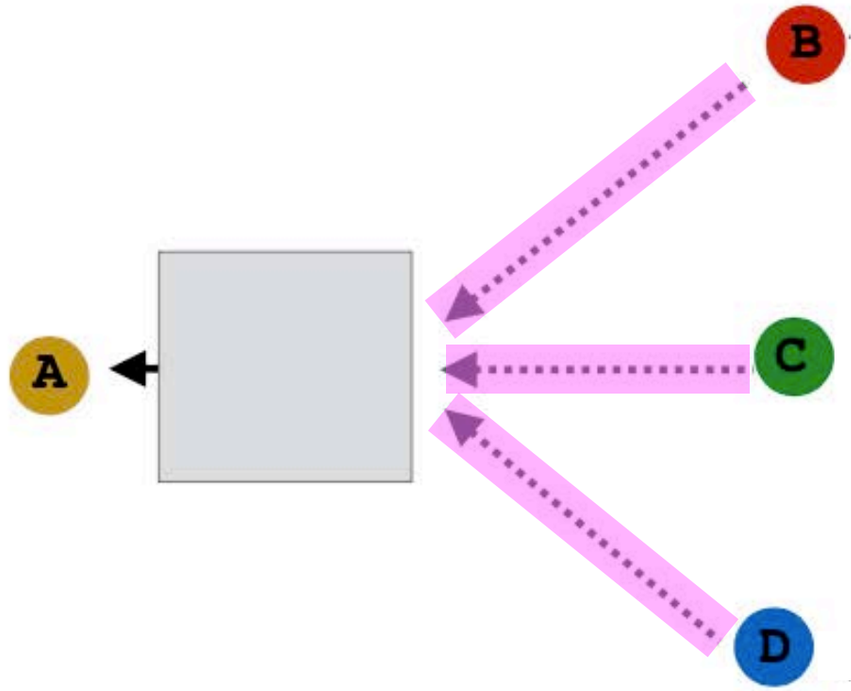
INPUT GRAPH

Graph Neural Network (GNN)

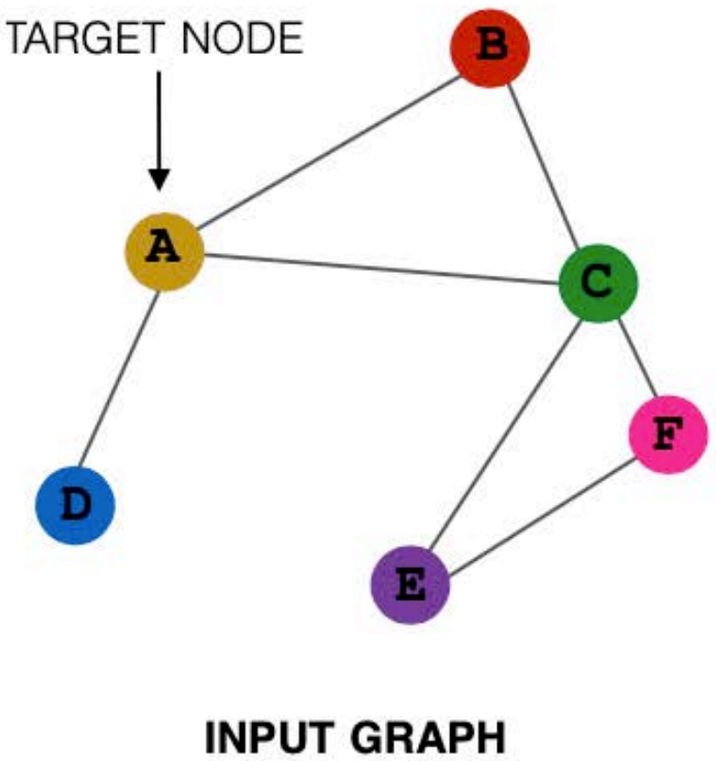
Idea: For each node, generate its node embedding ("features") based on its local neighborhood



INPUT GRAPH



Graph Neural Network (GNN)



INPUT GRAPH

Layer 2

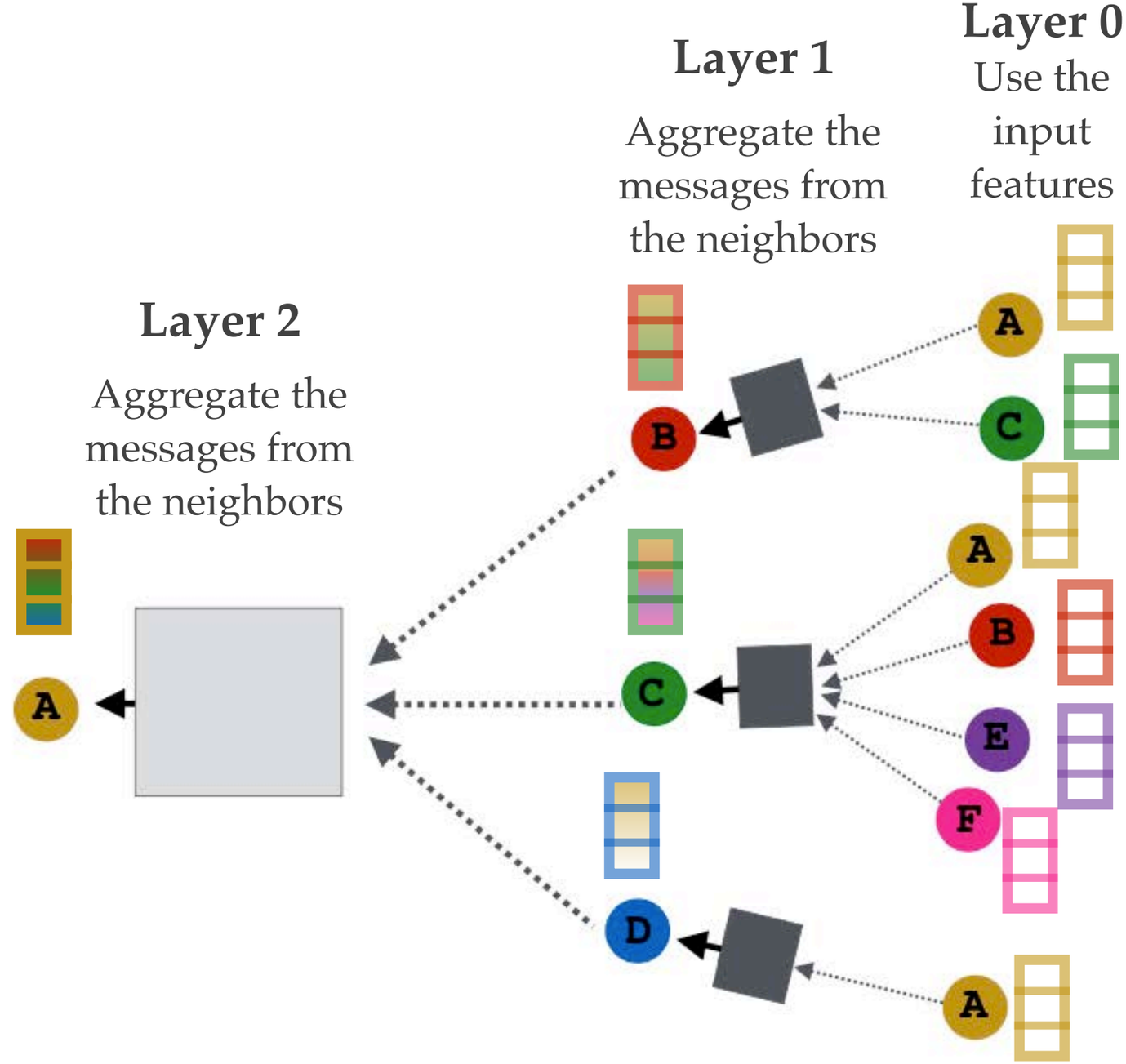
Aggregate the messages from the neighbors

Layer 1

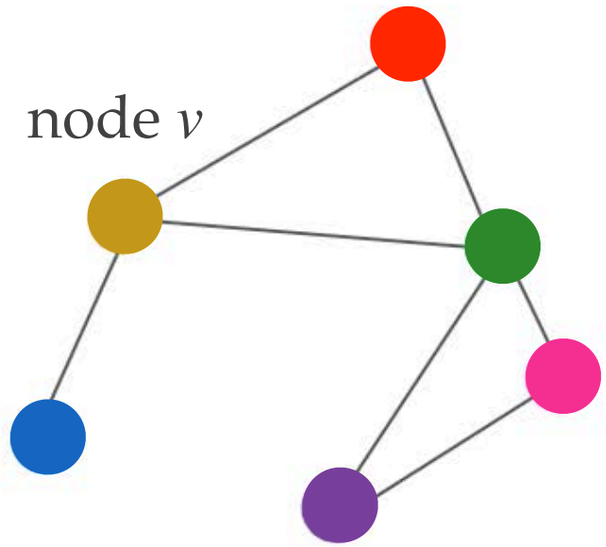
Aggregate the messages from the neighbors

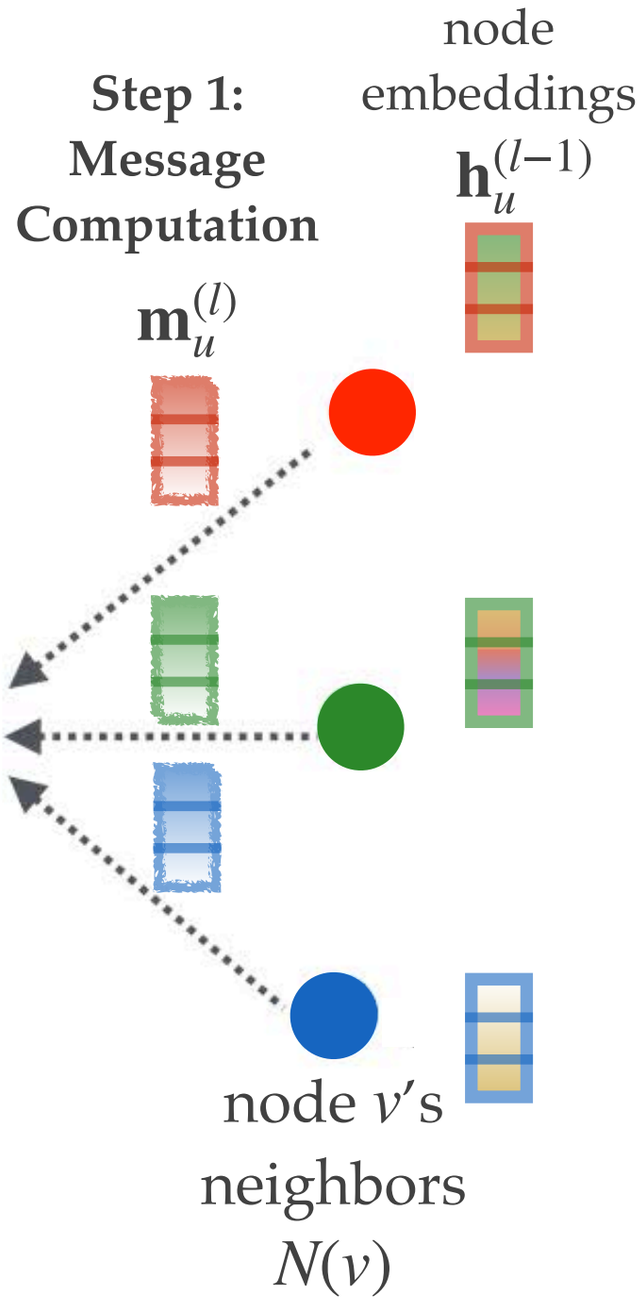
Layer 0

Use the input features



A Single GNN Layer



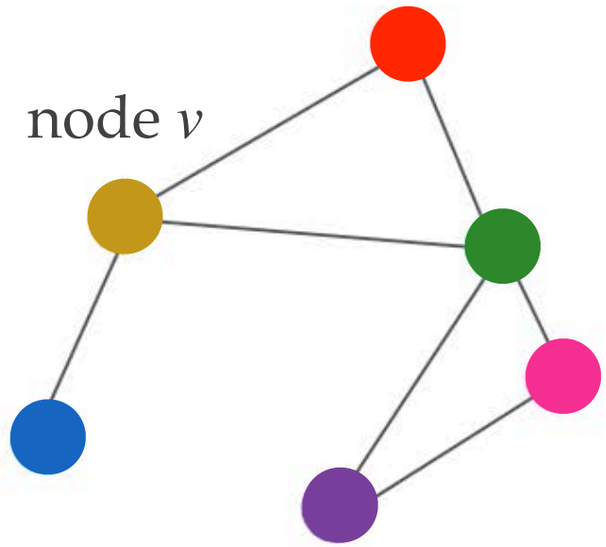


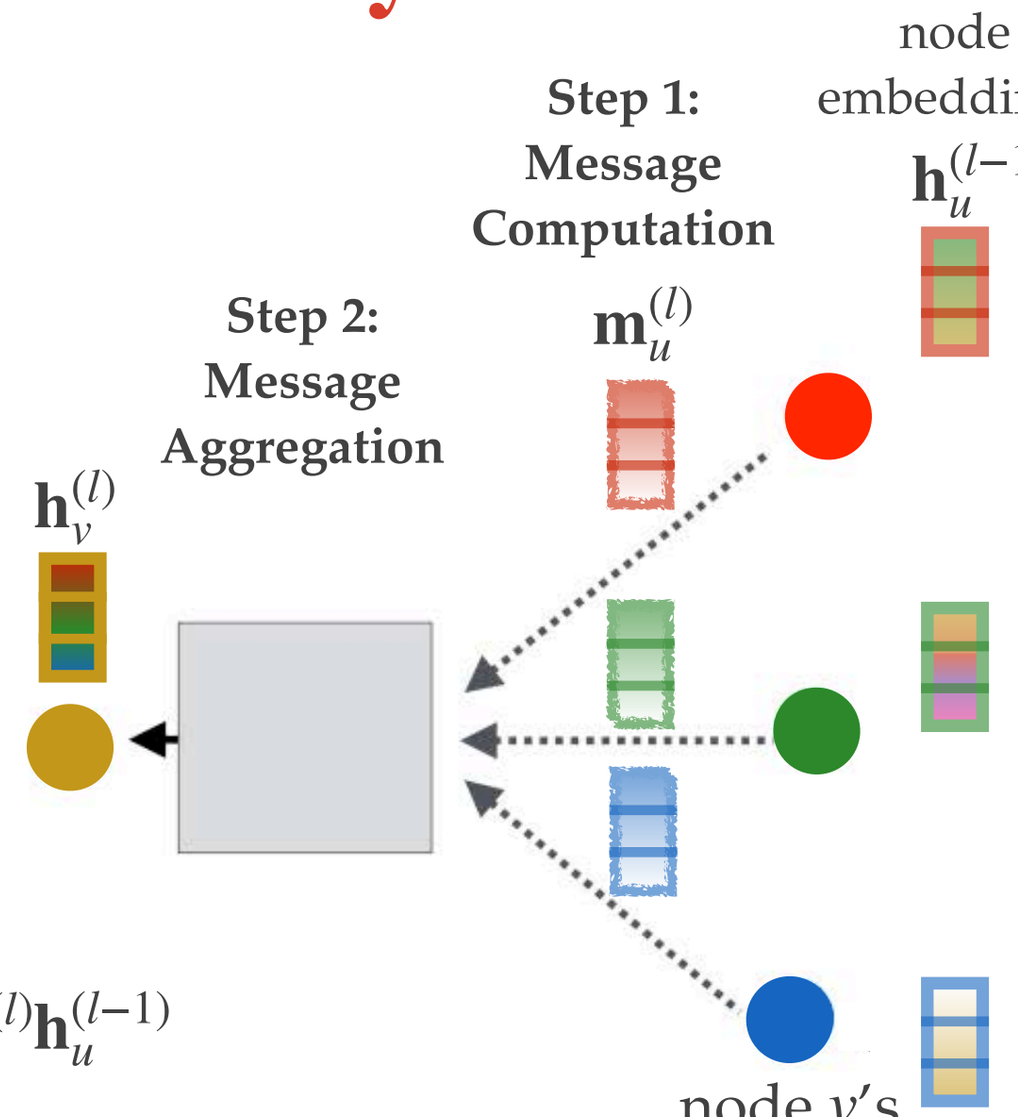
1. Message computation

$$\mathbf{m}_u^{(l)} = MSG^{(l)}(\mathbf{h}_u^{(l-1)}), u \in \{N(v) \cup v\}$$

Ex. Applying a linear layer $\mathbf{m}_u^{(l)} = W^{(l)}\mathbf{h}_u^{(l-1)}$

A Single GNN Layer





1. Message computation

$$\mathbf{m}_u^{(l)} = MSG^{(l)}(\mathbf{h}_u^{(l-1)}), u \in \{N(v) \cup v\}$$

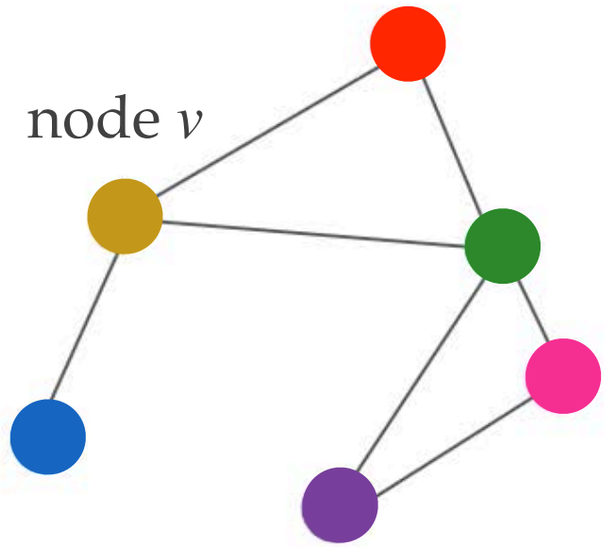
Ex. Applying a linear layer $\mathbf{m}_u^{(l)} = W^{(l)}\mathbf{h}_u^{(l-1)}$

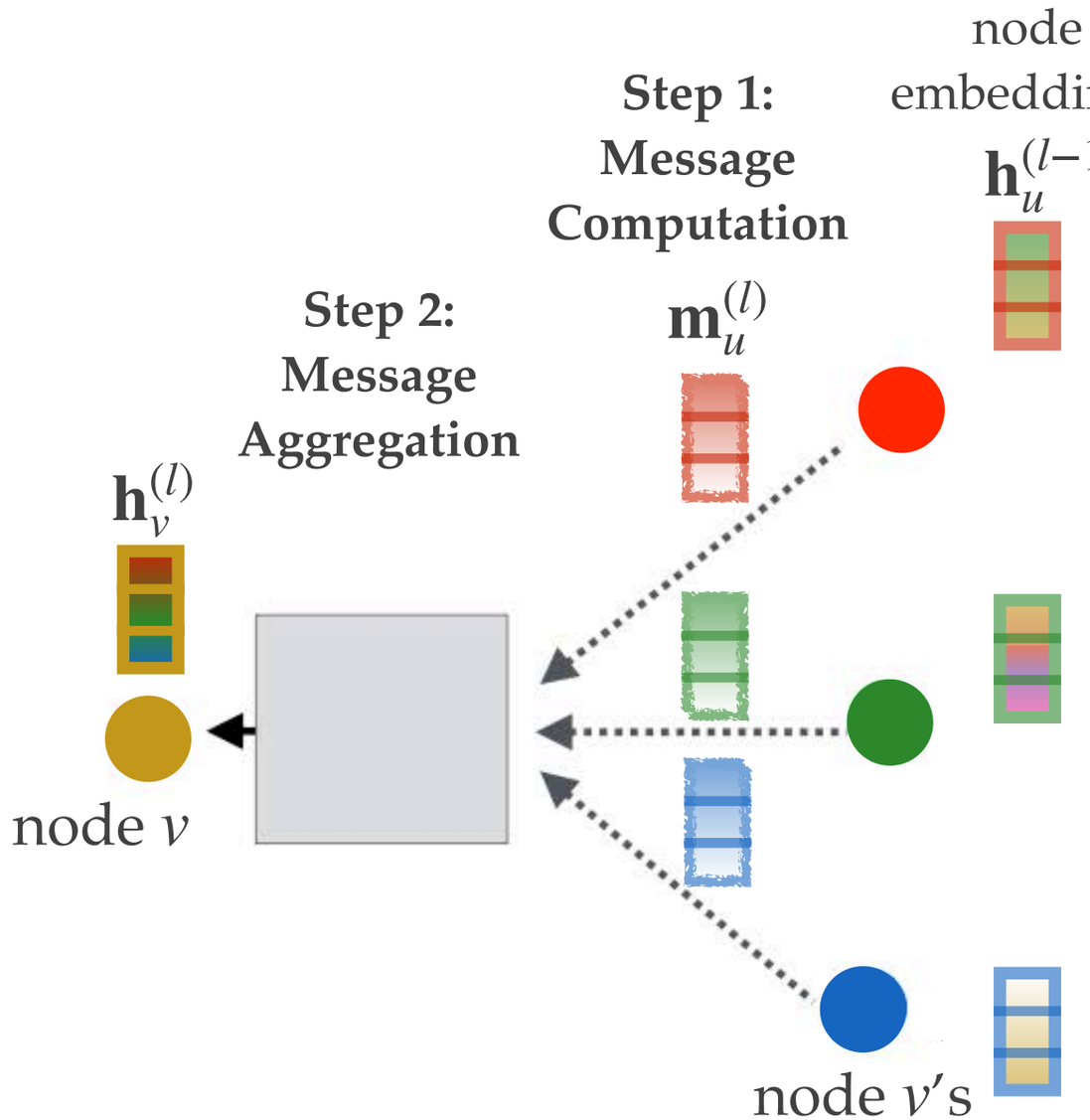
2. Message aggregation possibly with its own message

$$\mathbf{h}_v^{(l)} = AGG^{(l)}(\{\mathbf{m}_u^{(l)}, u \in N(v)\}), \mathbf{m}_v^{(l)})$$

Ex. Summing over the messages $\mathbf{h}_v^{(l)} = \sum_{u \in \{N(v) \cup v\}} \mathbf{m}_u^{(l)}$

Graph Convolutional Network (GCN)





1. Message computation

$$\mathbf{m}_u^{(l)} = \frac{W^{(l)}\mathbf{h}_u^{(l-1)}}{|N(v)|}$$

2. Message aggregation

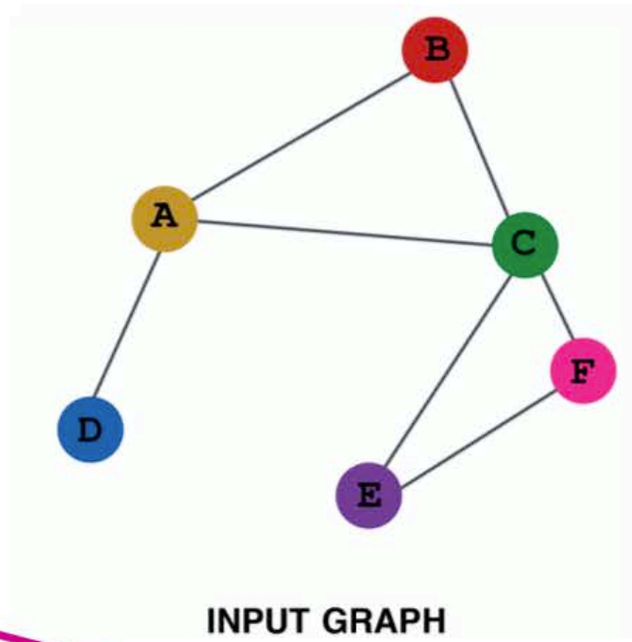
$$\mathbf{h}_v^{(l)} = ReLU\left(\sum_{u \in N(v)} \mathbf{m}_u^{(l)}\right)$$

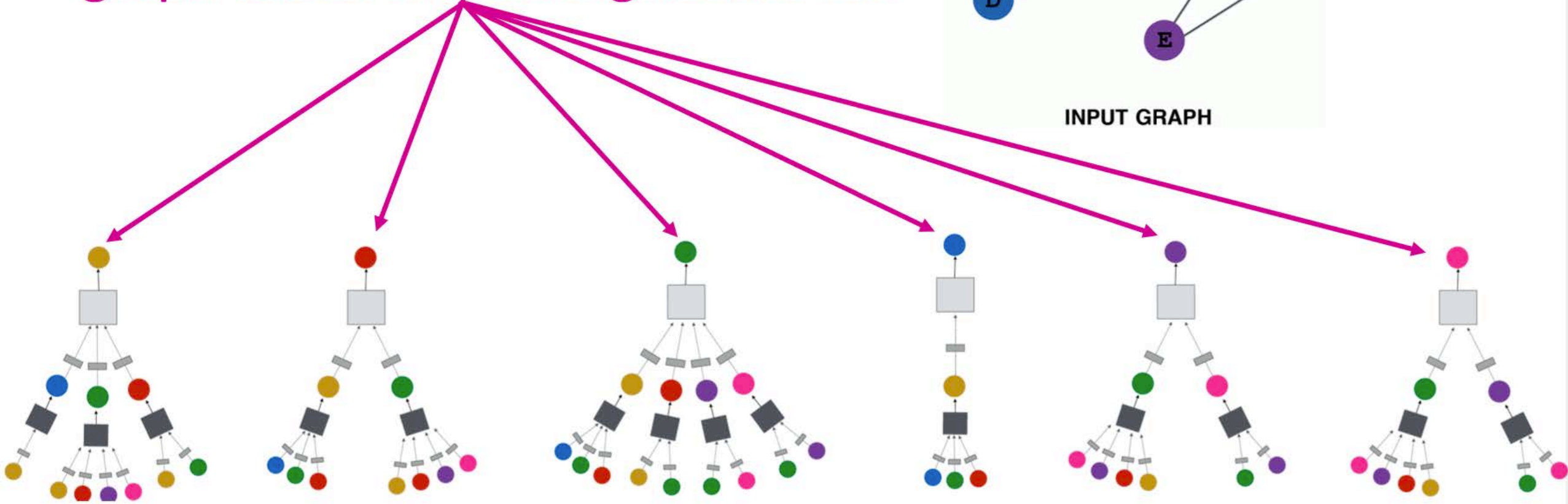
matrix form

$$\mathbf{H}^{(l)} = ReLU(\tilde{\mathbf{A}}\mathbf{H}^{(l-1)}\mathbf{W}^{(l)})$$

Graph Neural Network (GNN)

Every node defines a computation graph based on its neighborhood!



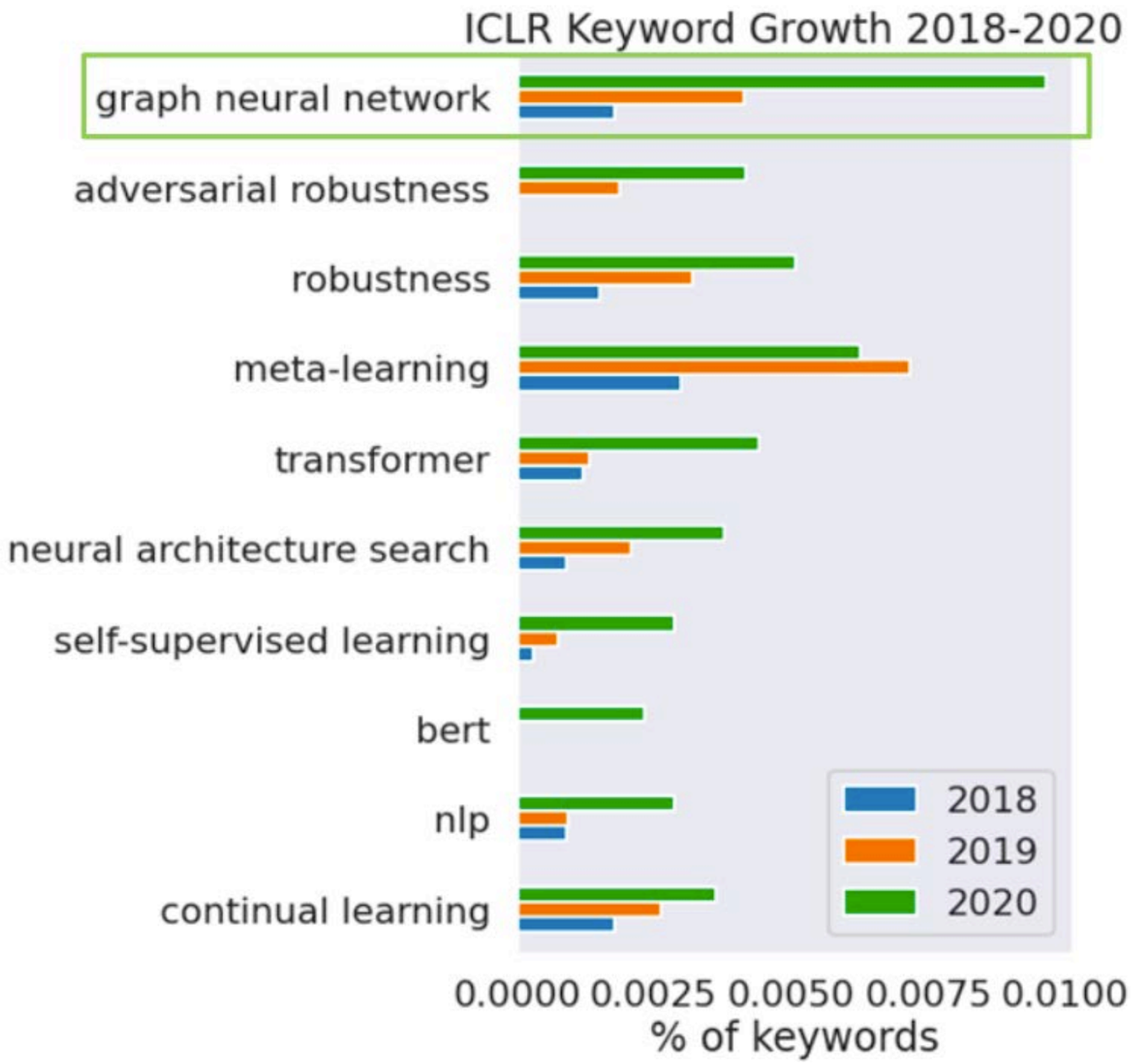


keywords: graph isomorphism network (GIN), graphSAGE, graph attention network (GAT), Cluster-GCN

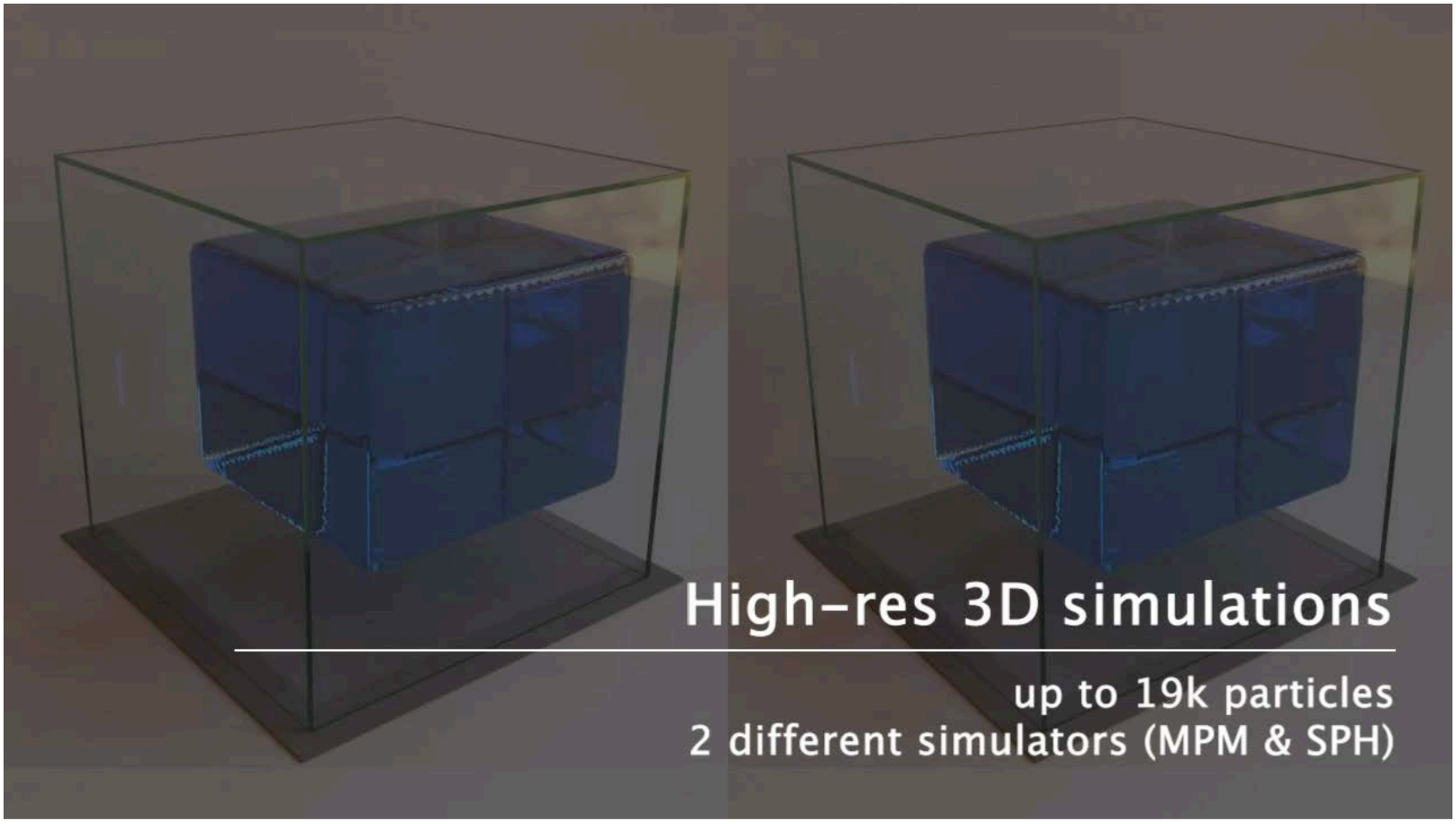
Graph Neural Network (GNN)

- There are many more techniques that are related to graph neural networks
  - Different ways to perform message computation and aggregation
  - Feature augmentation
  - Neighbor sampling
  - Position- and identity-aware GNNs
- Many deep learning components can also be incorporated such as
  - Attention mechanism
  - Gating
  - Skip/residual connections
  - Batch normalization
  - Dropout
  - Data augmentation

Graph Neural Network (GNN)



Applications: Physical Simulation



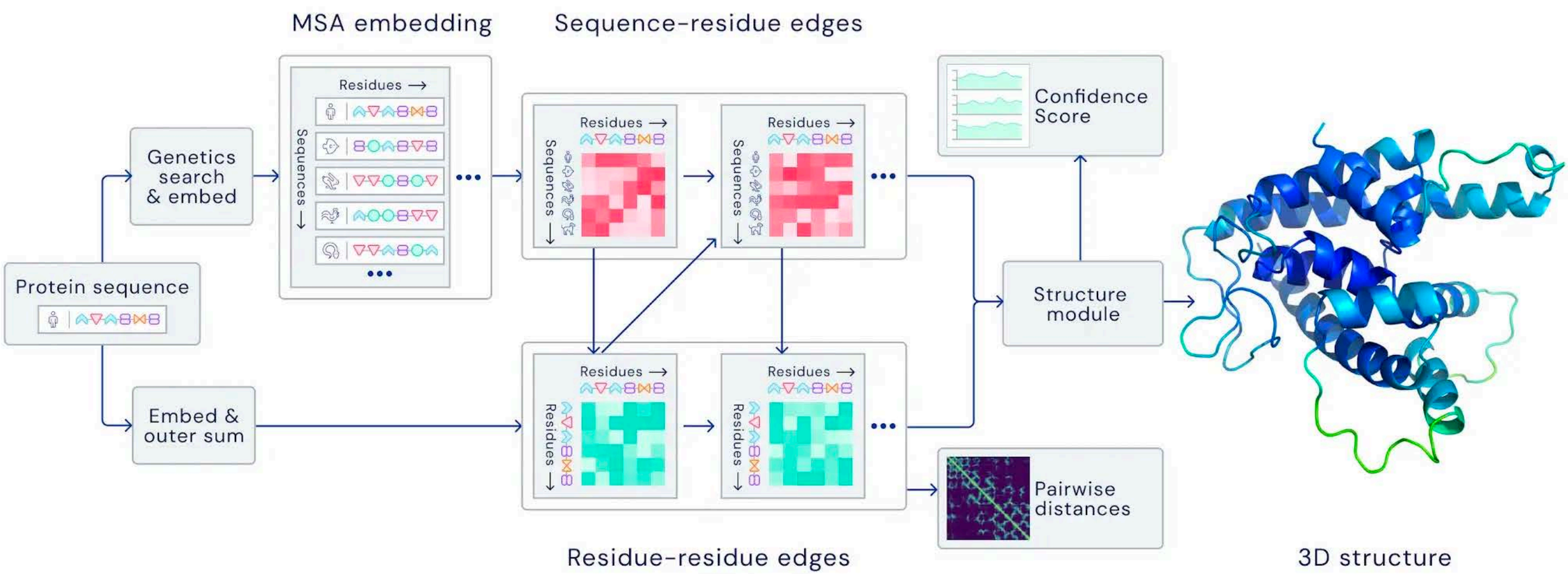
nodes: particles

edges: interaction between particles

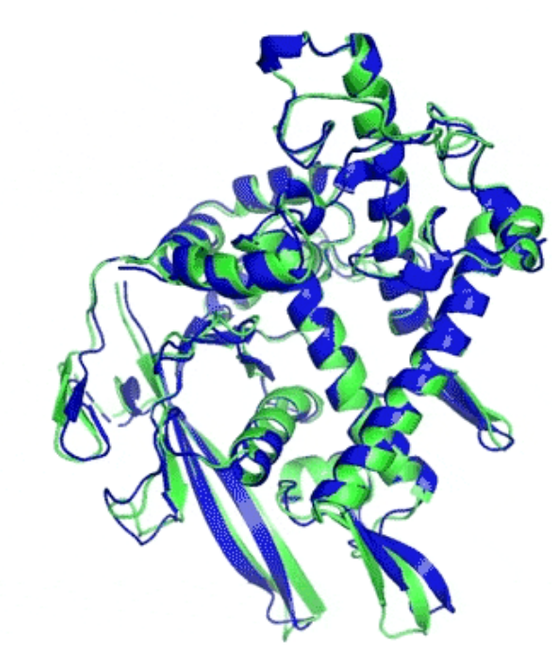
Sanchez-Gonzalez, Alvaro, et al. "Learning to simulate complex physics with graph networks." International Conference on Machine Learning. PMLR, 2020.

Applications: Protein Folding

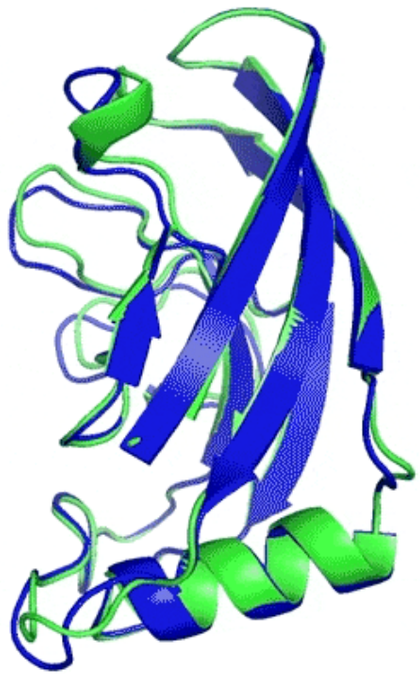
an amino acid sequence → the corresponding protein's 3D structure



Applications: Protein Folding



T1037 / 6vr4
90.7 GDT
(RNA polymerase domain)



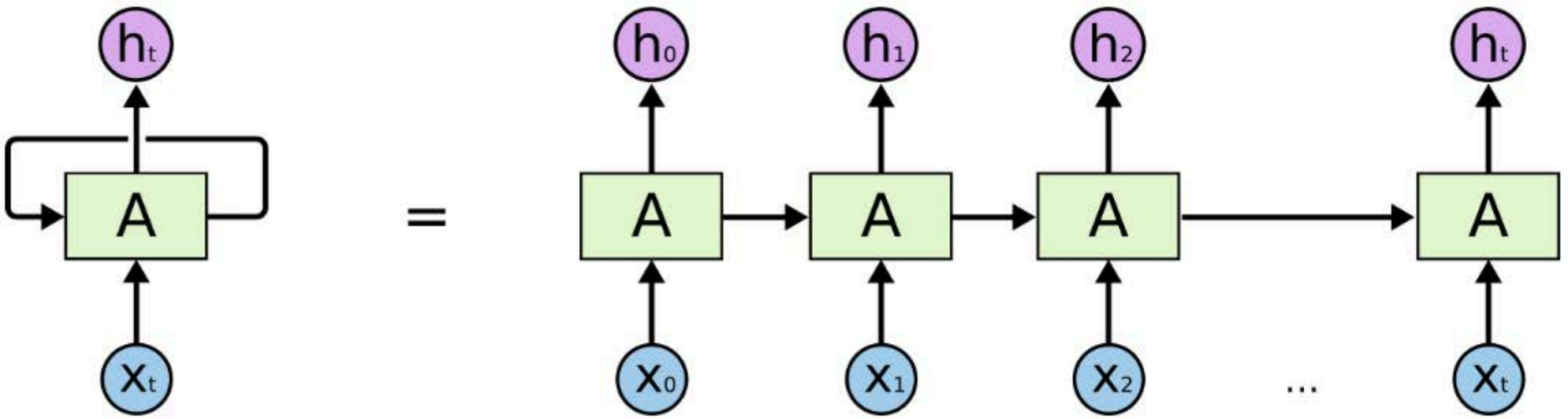
T1049 / 6y4f
93.3 GDT
(adhesin tip)

Experimental result

Computational prediction

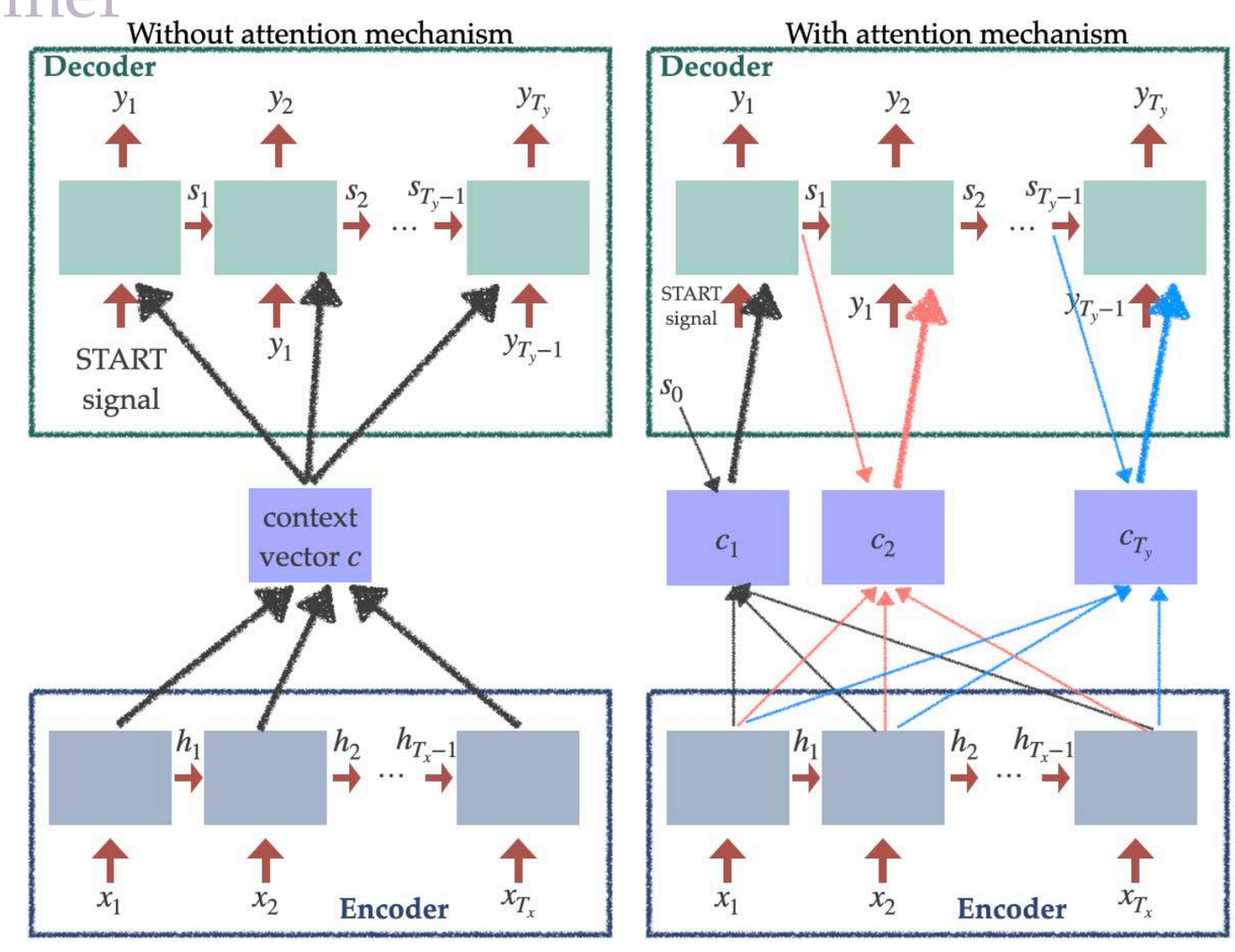
Outline

- Recurrent Neural Network (RNN)
- Sequence-to-Sequence Model
  - Attention Mechanism
- Self-Attention and Transformer
- Graph Neural Network



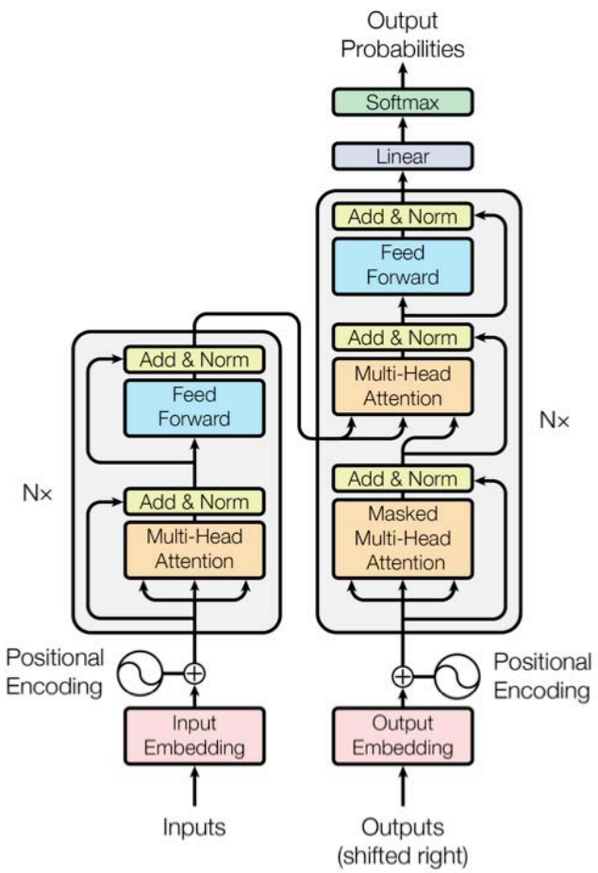
Outline

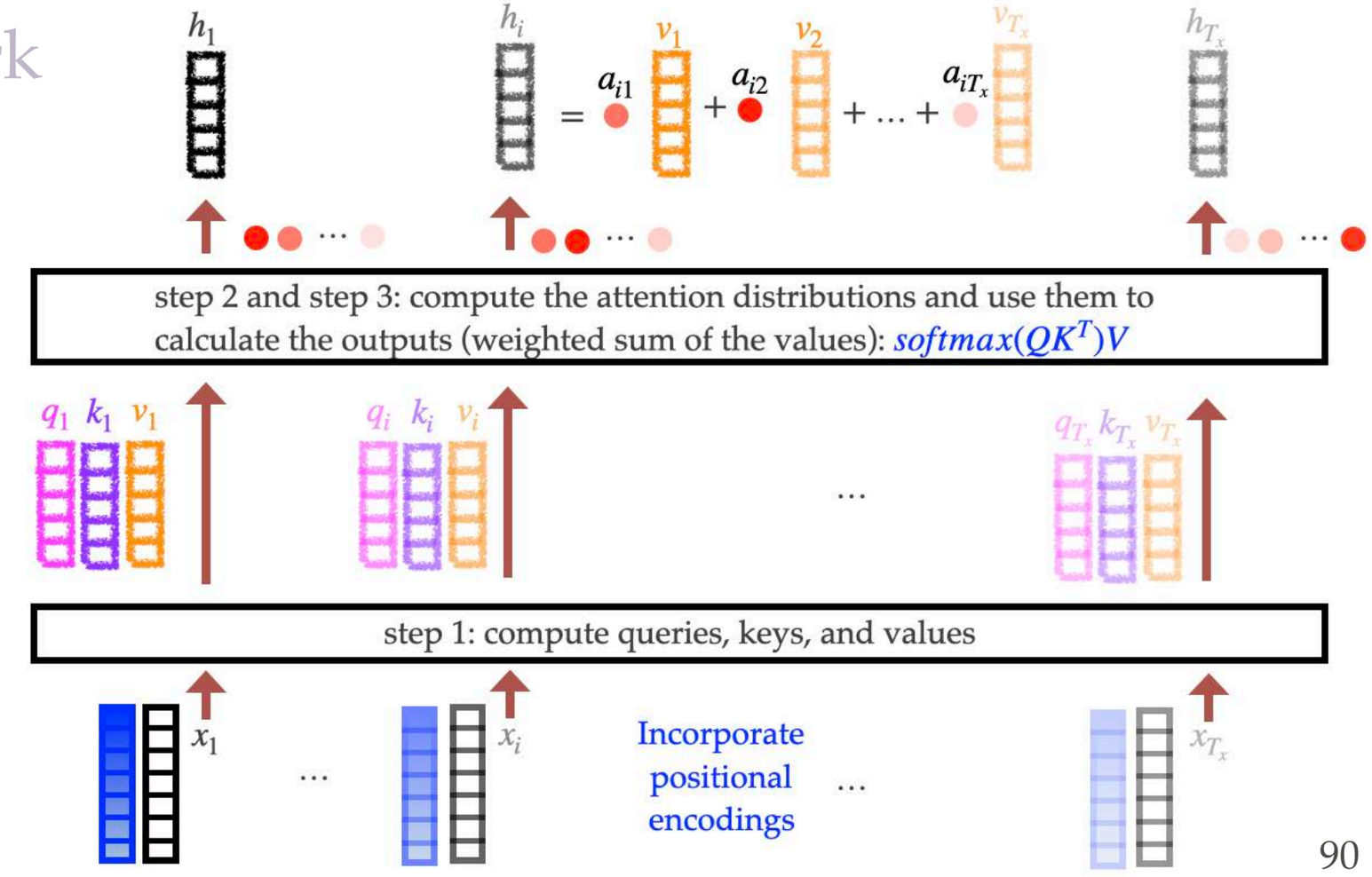
- Recurrent Neural Network (RNN)
- Sequence-to-Sequence Model
  - Attention Mechanism
- Self-Attention and Transformer
- Graph Neural Network



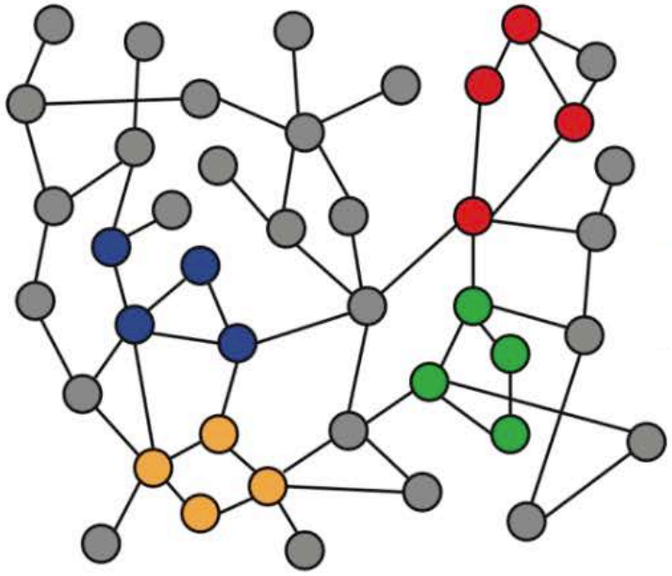
Outline

- Recurrent Neural Network (RNN)
- Sequence-to-Sequence Model
  - Attention Mechanism
- Self-Attention and Transformer
- Graph Neural Network

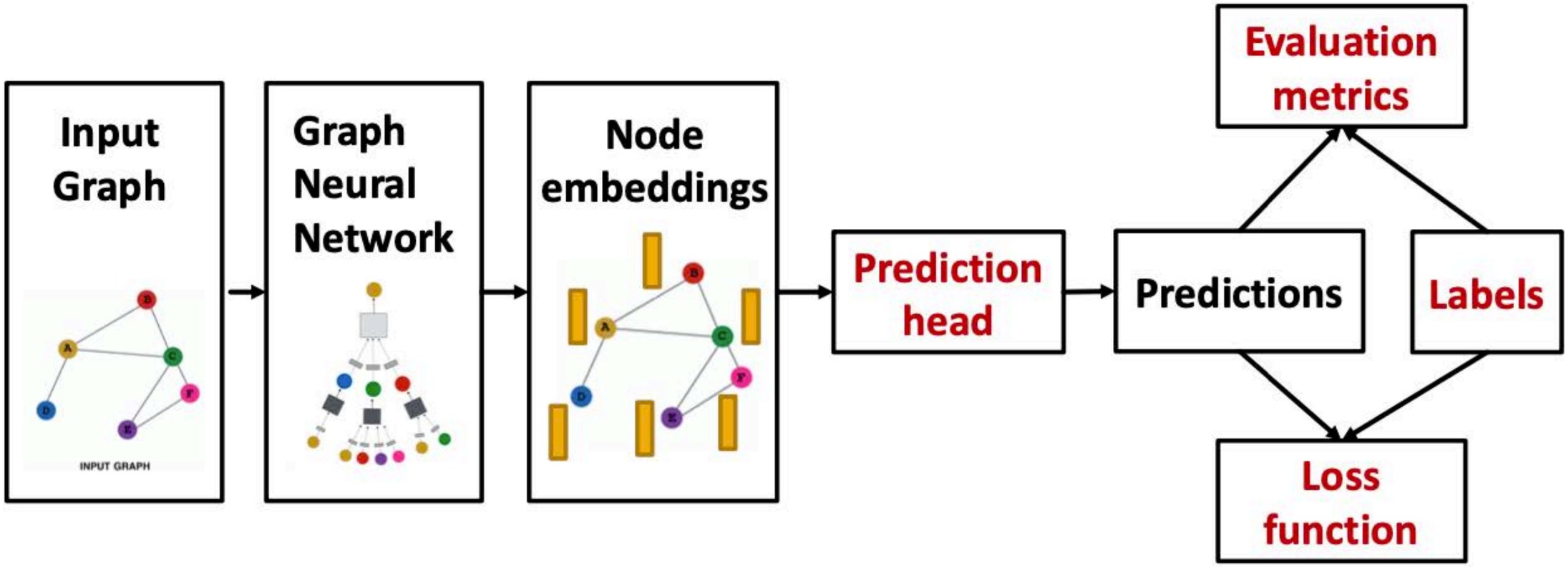




Outline



- Recurrent Neural Network (RNN)
- Sequence-to-Sequence Model
  - Attention Mechanism
- Self-Attention and Transformer
- Graph Neural Network



Resources

Conferences

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature.

1 Introduction

Recurrent neural networks, long short-term memory [12] and gated recurrent [7] neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and transduction problems such as language modeling and machine translation [29, 2, 5]. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures [31, 21, 13].

*Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

†Work performed while at Google Brain.

‡Work performed while at Google Research.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

This CVPR paper is the Open Access version, provided by the Computer Vision Foundation. Except for this watermark, it is identical to the version available on IEEE Xplore.

Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

Abstract

Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset we evaluate residual nets with a depth of up to 152 layers—8× deeper than VGG nets [40] but still having lower complexity. An ensemble of these residual nets achieves 3.57% error on the ImageNet test set. This result won the 1st place on the ILSVRC 2015 classification task. We also present analysis on CIFAR-10 with 100 and 1000 layers.

The depth of representations is of central importance for many visual recognition tasks. Solely due to our extremely deep representations, we obtain a 28% relative improvement on the COCO object detection dataset. Deep residual nets are foundations of our submissions to ILSVRC & COCO 2015 competitions[1], where we also won the 1st places on the tasks of ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation.

1. Introduction

Deep convolutional neural networks [22, 21] have led to a series of breakthroughs for image classification [21, 49, 39]. Deep networks naturally integrate low/mid/high-level features [49] and classifiers in an end-to-end multi-layer fashion, and the "levels" of features can be enriched by the number of stacked layers (depth). Recent evidence [40, 43] reveals that network depth is of crucial importance, and the leading results [40, 43, 12, 16] on the challenging ImageNet dataset [35] all exploit "very deep" [40] models, with a depth of sixteen [40] to thirty [16]. Many other non-trivial visual recognition tasks [7, 11, 6, 32, 27] have also

[1]http://image-net.org/challenges/LSVRC/2015/ and http://mscoco.org/dataset/#detections-challenge2015.

Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

greatly benefited from very deep models.

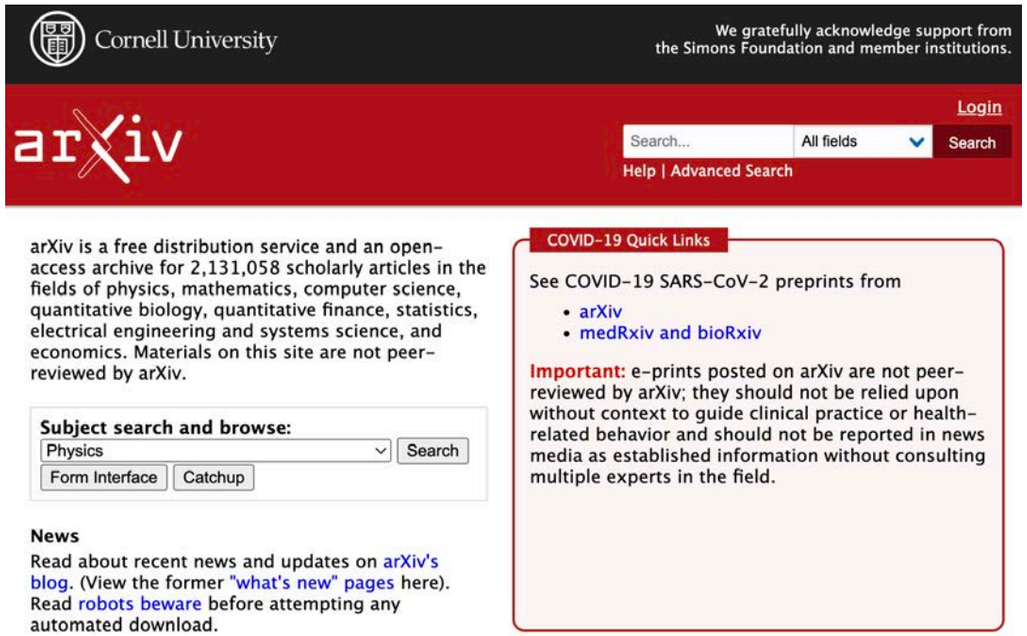
Driven by the significance of depth, a question arises: *Is learning better networks as easy as stacking more layers?* An obstacle to answering this question was the notorious problem of vanishing/exploding gradients [14, 1, 8], which hamper convergence from the beginning. This problem, however, has been largely addressed by normalized initialization [23, 8, 36, 12] and intermediate normalization layers [16], which enable networks with tens of layers to start converging for stochastic gradient descent (SGD) with back-propagation [22].

When deeper networks are able to start converging, a *degradation* problem has been exposed: with the network depth increasing, accuracy gets saturated (which might be unsurprising) and then degrades rapidly. Unexpectedly, such degradation is *not caused by overfitting*, and adding more layers to a suitably deep model leads to *higher training error*, as reported in [10, 41] and thoroughly verified by our experiments. Fig. 1 shows a typical example.

The degradation (of training accuracy) indicates that not all systems are similarly easy to optimize. Let us consider a shallower architecture and its deeper counterpart that adds more layers onto it. There exists a solution *by construction* to the deeper model: the added layers are *identity* mapping, and the other layers are copied from the learned shallower model. The existence of this constructed solution indicates that a deeper model should produce no higher training error than its shallower counterpart. But experiments show that our current solvers on hand are unable to find solutions that

770

arXiv



Cornell University
We gratefully acknowledge support from the Simons Foundation and member institutions.
arXiv
Login
Search... All fields Search
Help | Advanced Search

arXiv is a free distribution service and an open-access archive for 2,131,058 scholarly articles in the fields of physics, mathematics, computer science, quantitative biology, quantitative finance, statistics, electrical engineering and systems science, and economics. Materials on this site are not peer-reviewed by arXiv.

Subject search and browse:
Physics Search
Form Interface Catchup

News
Read about recent news and updates on arXiv's blog. (View the former "what's new" pages here).
Read robots beware before attempting any automated download.

COVID-19 Quick Links
See COVID-19 SARS-CoV-2 preprints from
- arXiv
- medRxiv and bioRxiv

Important: e-prints posted on arXiv are not peer-reviewed by arXiv; they should not be relied upon without context to guide clinical practice or health-related behavior and should not be reported in news media as established information without consulting multiple experts in the field.

Courses

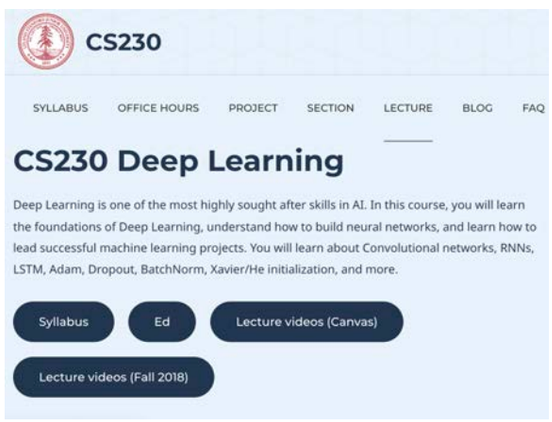
CS229

CS229: Machine Learning

Instructor

Anand Avati

Course Description This course provides a broad introduction to machine learning and statistical pattern recognition. Topics include: supervised learning (generative/discriminative learning, parametric/non-parametric learning, neural networks, support vector machines); unsupervised learning (clustering, dimensionality reduction, kernel methods); learning theory (bias/variance tradeoffs, practical advice); reinforcement learning and adaptive control. The course will also discuss recent applications of machine learning, such as to robotic control, data mining, autonomous navigation, bioinformatics, speech recognition, and text and web data processing.



CS230

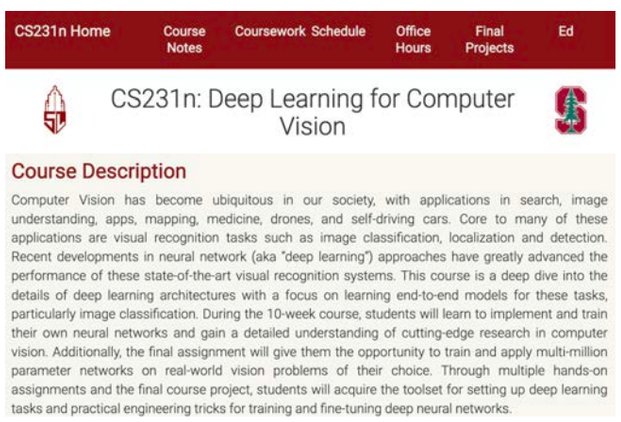
SYLLABUS OFFICE HOURS PROJECT SECTION LECTURE BLOG FAQ

CS230 Deep Learning

Deep Learning is one of the most highly sought after skills in AI. In this course, you will learn the foundations of Deep Learning, understand how to build neural networks, and learn how to lead successful machine learning projects. You will learn about Convolutional networks, RNNs, LSTM, Adam, Dropout, BatchNorm, Xavier/He initialization, and more.

Syllabus Ed Lecture videos (Canvas)

Lecture videos (Fall 2018)



CS231n Home Course Notes Coursework Schedule Office Hours Final Projects Ed

CS231n: Deep Learning for Computer Vision

Course Description

Computer Vision has become ubiquitous in our society, with applications in search, image understanding, apps, mapping, medicine, drones, and self-driving cars. Core to many of these applications are visual recognition tasks such as image classification, localization and detection. Recent developments in neural network (aka "deep learning") approaches have greatly advanced the performance of these state-of-the-art visual recognition systems. This course is a deep dive into the details of deep learning architectures with a focus on learning end-to-end models for these tasks, particularly image classification. During the 10-week course, students will learn to implement and train their own neural networks and gain a detailed understanding of cutting-edge research in computer vision. Additionally, the final assignment will give them the opportunity to train and apply multi-million parameter networks on real-world vision problems of their choice. Through multiple hands-on assignments and the final course project, students will acquire the toolset for setting up deep learning tasks and practical engineering tricks for training and fine-tuning deep neural networks.

INTRODUCTION

2110573 Pattern Recognition

Pattern Recognition 2022: L01-Introduction

1,163 views Streamed live on Jan 13, 2022 0:00 pre-stream
18:00 introduction & course logistics
45:07 Technology trends and machine learning
1:21:27 AI&ML&PR Terminologies
1:51:19 !!!!
1:55:11 Introduction to supervised learning
2:06:18 Typical workflow in machine learning
2:12:49 Feature extraction
2:17:29 Evaluation & metrics
2:33:02 Metrics in detection problems (Precision, recall, accuracy, ...)
3:04:35 In class exercise
3:06:39 In class solution

Textbooks

Journals

Tutorials

Blog posts

YouTube videos