University of Antwerp

# Artificial Neural Networks

[2500WETANN]

José Oramas

# Modeling Sequences with Neural Networks

José Oramas

# Today's Lecture – Outline

- **Recap - Sequence Modeling**

  → [Algorithmically] How to approach the problem

- **Sequence Modeling with Recurrent Architectures**

  → RNNs, LSTMs, GRUs, etc.

- **Sequence Generation**

- **Transformers and Attention Mechanisms**

# Modeling Data Sequences [ in theory ]

## Some Foundations

- Supervised Learning

- Data

$$\{x, y\}_i$$

– Model

$$\hat{y} \approx f_\theta(x)$$

– Loss

$$L(\theta) = \sum_{i=1}^{N} l(f_\theta(x_i), y_i)$$

– Optimization

$$\theta^* = arg\ min_\theta\ L(\theta)$$

# Modeling Data Sequences [ in theory ]

## Some Foundations

- **Supervised Learning**

  - Data

    $$\{x, y\}_i$$

  – Model

    $$\hat{y} \approx f_\theta(x)$$

  – Loss

    $$L(\theta) = \sum_{i=1}^{N} l(f_\theta(x_i), y_i)$$

  – Optimization

    $$\theta^* = arg\ min_\theta\ L(\theta)$$

- **Modeling Sequences**

  - Data

    $$\{x\}_i$$

  – Model

    $$p(x) \approx f_\theta(x)$$

  – Loss

    $$L(\theta) = \sum_{i=1}^{N} log\ p(f_\theta(x_i))$$

  – Optimization

    $$\theta^* = arg\ max_\theta\ L(\theta)$$

# Modeling Sequences – classical approaches

## Summarizing

- + Good / Intuitive Models
- - Not Scalable

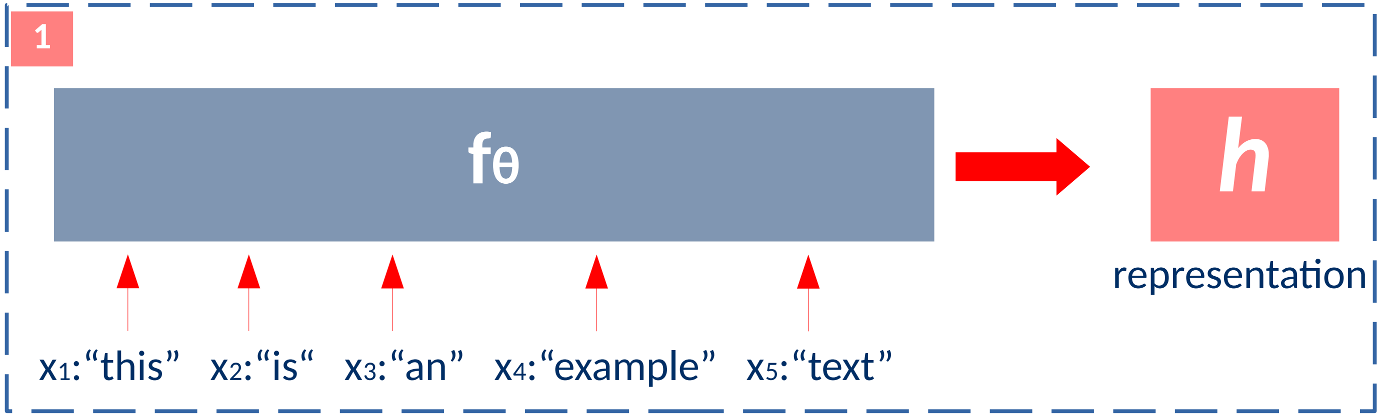
# Modeling Data Sequences

[ How to do it... in practice ]

# Modeling Data Sequences [ in practice ]

## 1. Describing/Vectorizing the Context

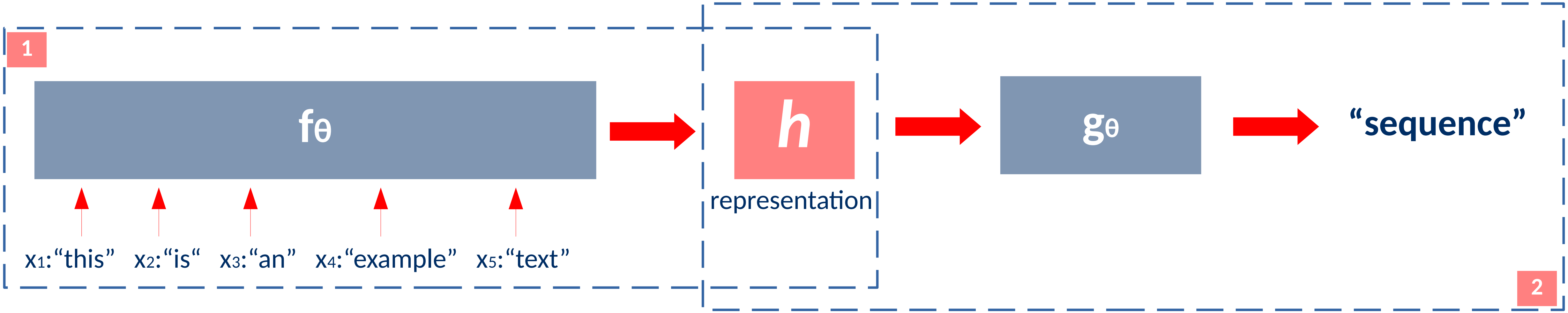
- **Idea:** Learn how to represent a sub-sequence



# Modeling Data Sequences [ in practice ]

## 1. Describing/Vectorizing the Context

- **Idea:** Learn how to represent a sub-sequence




## 2. Modeling Conditional Probabilities

- **Idea:** Predicting the next element given the context

# Recurrent Architectures

[ RNNs, LSTMs, GRUs, etc.]

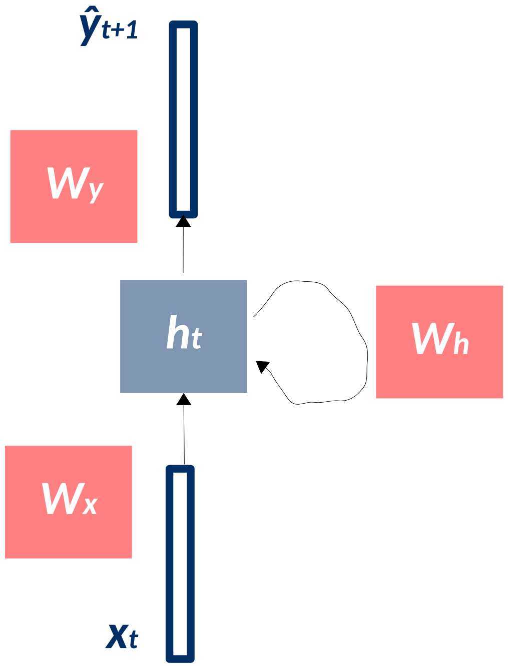
# Recurrent Neural Networks

[ The Most Popular Architecture ]

# Recurrent Neural Networks (RNNs) [ Elman, 1991 ]

## Provide Neural Networks with Memory

- **Idea:** Use a persistent state ***h*** that encodes past observations (context)



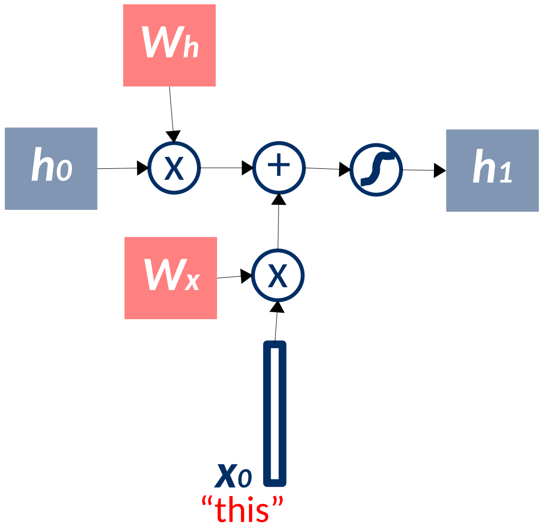

**Defined by three equations**

$$\boxed{\begin{aligned} h_t &= \tanh(\ W_h h_{t-1} + W_x x_t\ ) \\ p(y_{t+1}) &= softmax(\ W_y h_t\ ) \\ L_\theta(y, \hat{y})_t &= -y_t \log \hat{y}_t \end{aligned}}$$

# Recurrent Neural Networks (RNNs) [ Elman, 1991 ]

## Provide Neural Networks with Memory

- **Idea:** Use a persistent state ***h*** that encodes past observations (context)

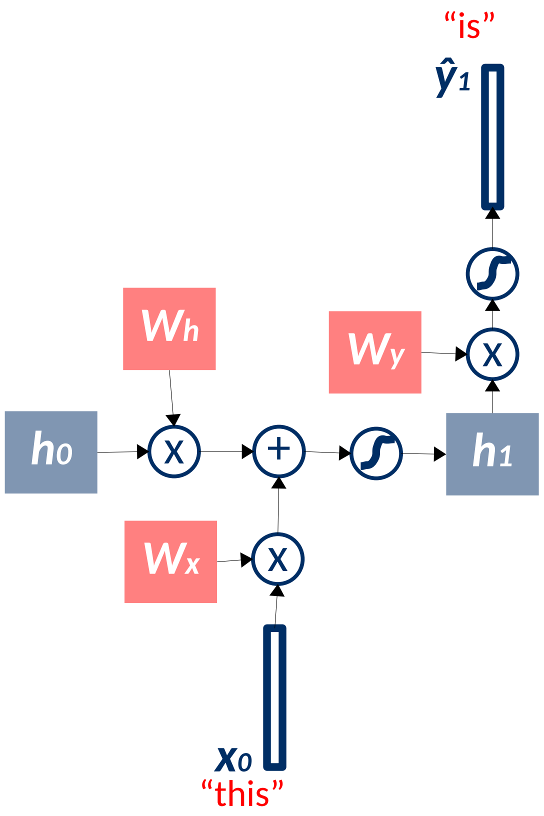



$$h_t = \tanh(\ W_h h_{t-1} + W_x x_t\ )$$

# Recurrent Neural Networks (RNNs) [ Elman, 1991 ]

## Provide Neural Networks with Memory

- **Idea:** Use a persistent state ***h*** that encodes past observations (context)



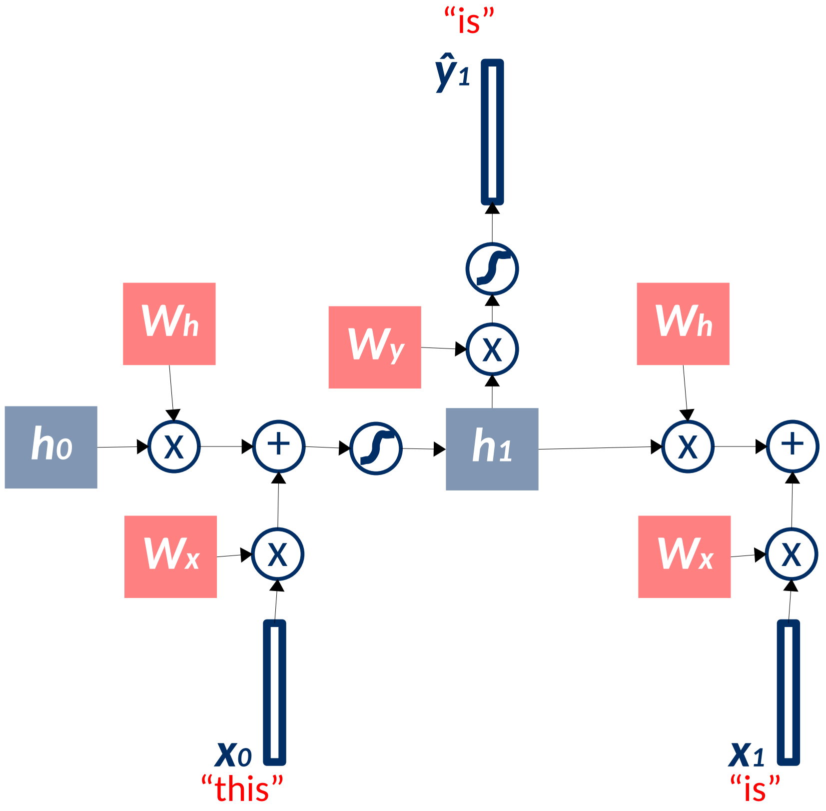

The probability of next element is obtained from the state ***h***

$$p(y_{t+1}) = softmax(\ W_y h_t\ )$$

# Recurrent Neural Networks (RNNs) [ Elman, 1991 ]

## Provide Neural Networks with Memory

- **Idea:** Use a persistent state ***h*** that encodes past observations (context)

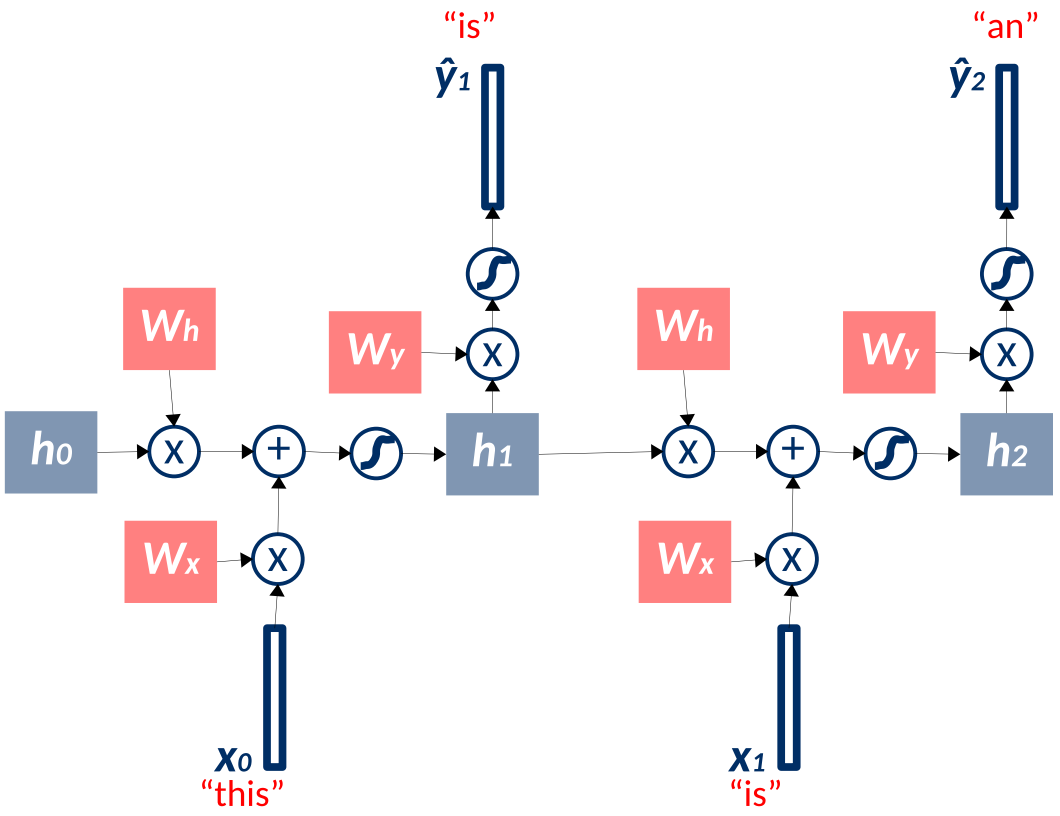



Input the next element $x_1$ from the sequence

# Recurrent Neural Networks (RNNs) [ Elman, 1991 ]

## Provide Neural Networks with Memory

- **Idea:** Use a persistent state ***h*** that encodes past observations (context)

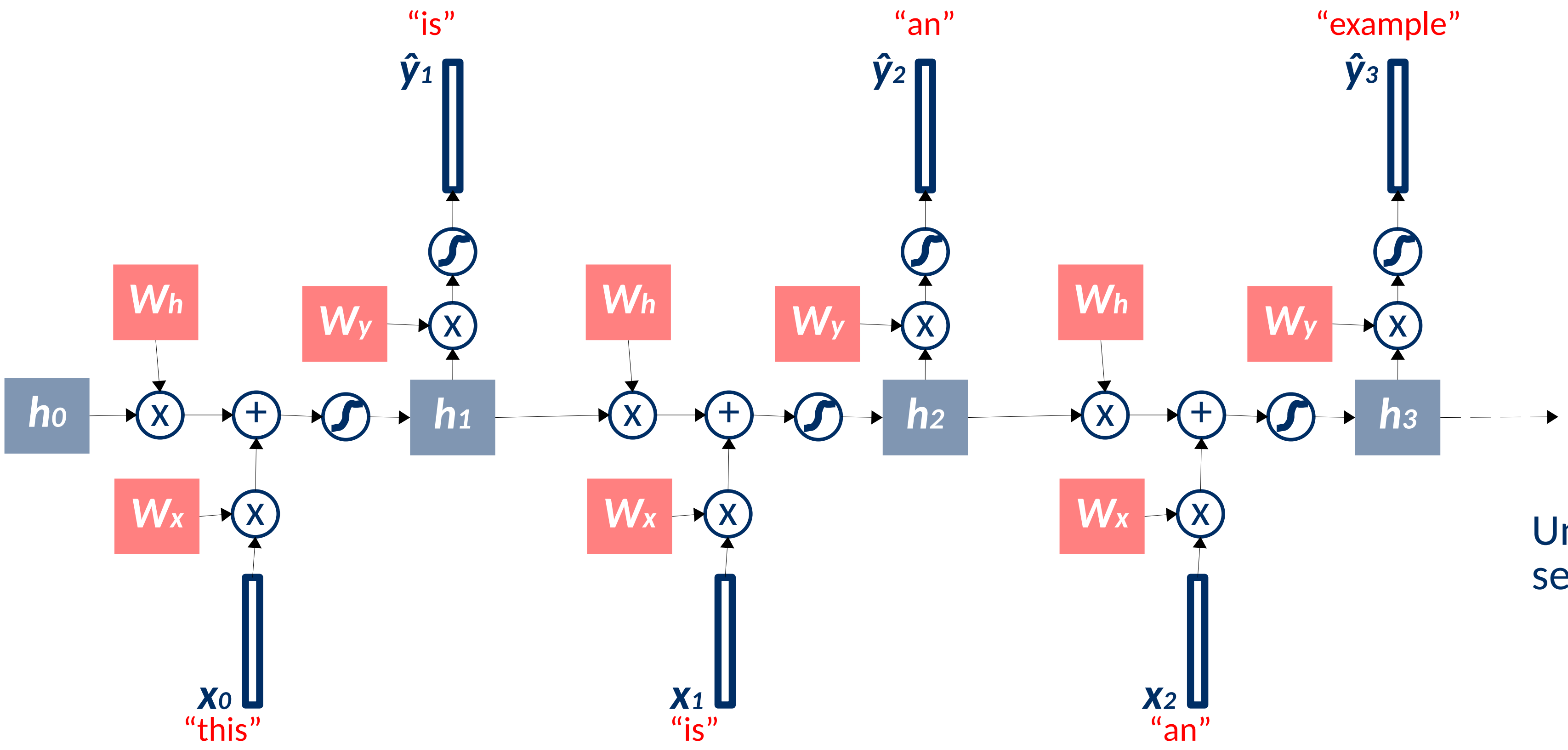



Keep going

# Recurrent Neural Networks (RNNs) [ Elman, 1991 ]

## Provide Neural Networks with Memory

- **Idea:** Use a persistent state ***h*** that encodes past observations (context)

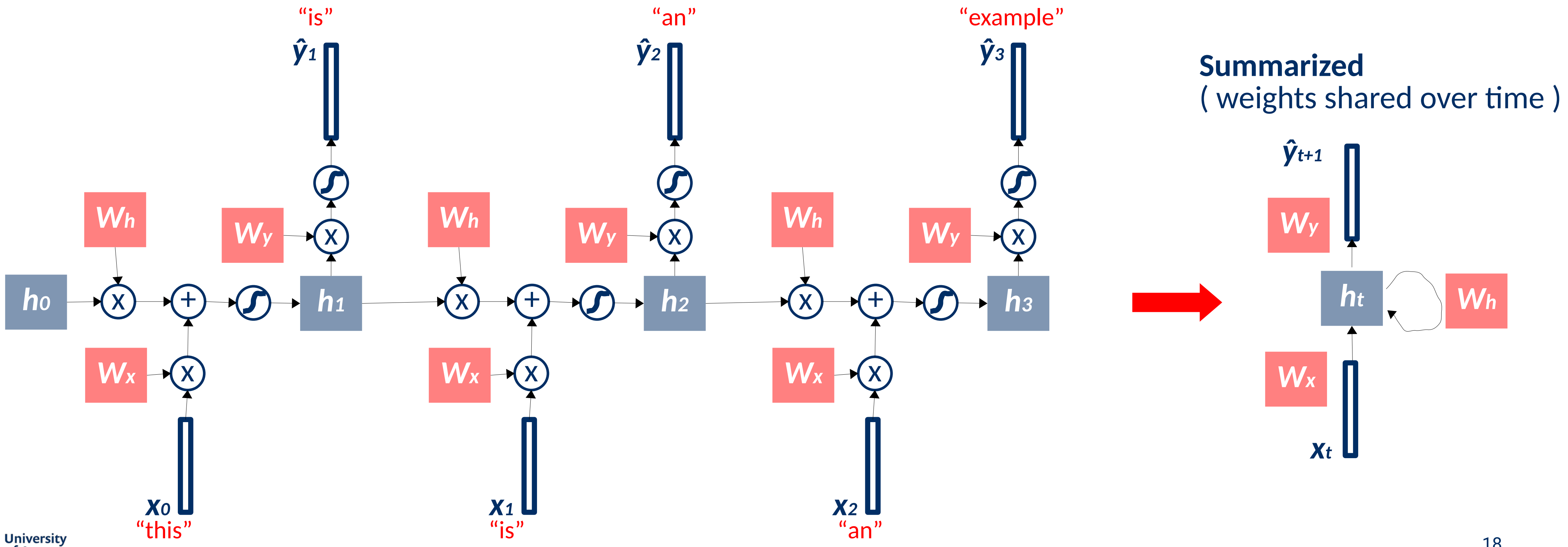



Until the whole sequence is pushed

# Recurrent Neural Networks (RNNs) [ Elman, 1991 ]

## Provide Neural Networks with Memory

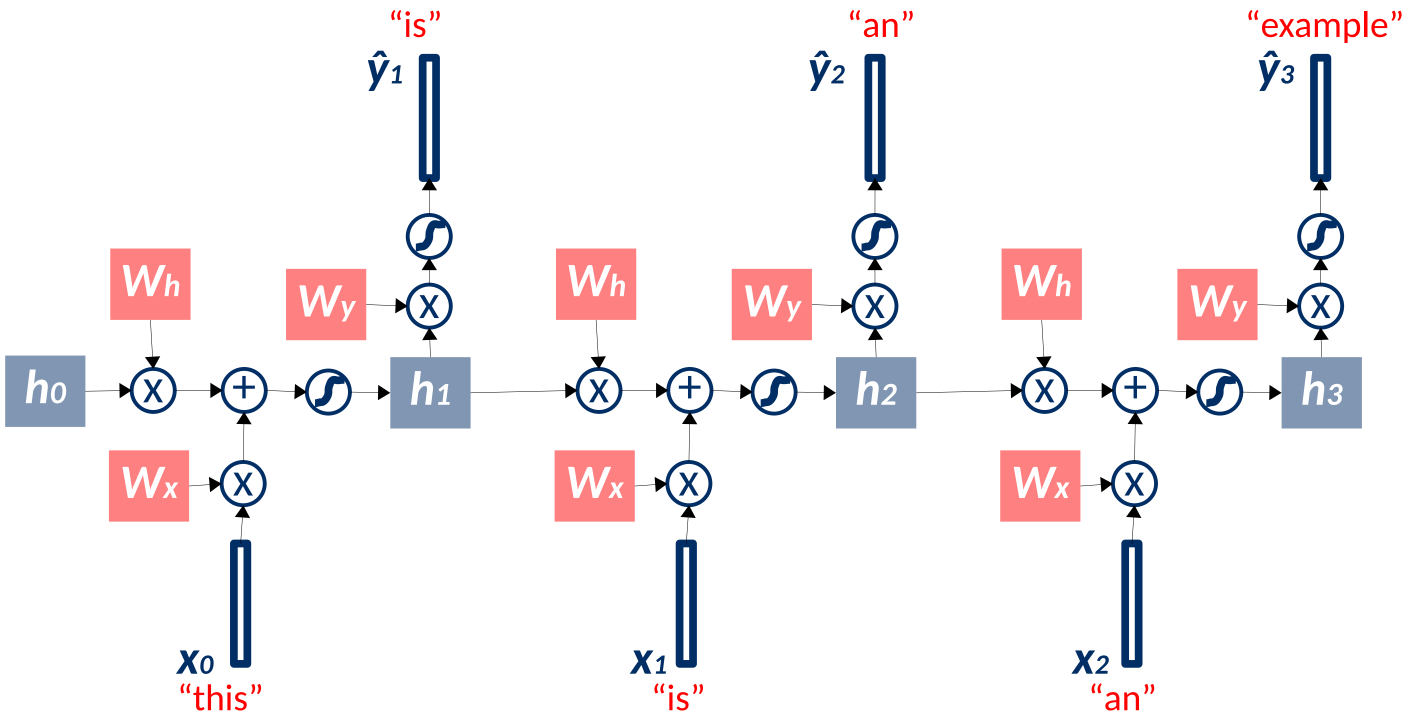
- **Idea:** Use a persistent state ***h*** that encodes past observations (context)



# Recurrent Neural Networks (RNNs) [ Elman, 1991 ]

## Learning | Training

- **Idea:** Formulate the next-word prediction as a classification problem
- Number of classes = vocabulary_size → Use the cross-entropy loss.




University of Antwerp

# Recurrent Neural Networks (RNNs) [ Elman, 1991 ]

## Learning | Training

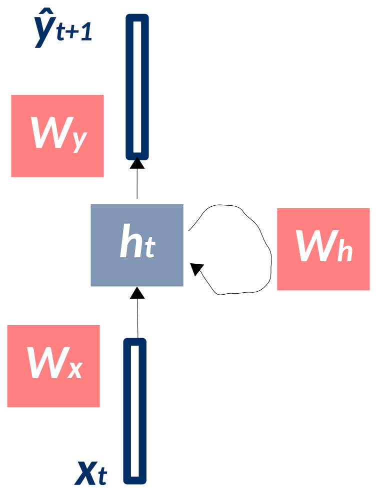
- **Idea:** Formulate the next-word prediction as a classification problem
- Number of classes = vocabulary_size → Use the cross-entropy loss.

Given a sequence of ***T*** elements:

For one element ***t*** → $L_\theta(y, \hat{y})_t = -y_t \log \hat{y}_t$

For the sequence → $L_\theta(y, \hat{y}) = -\sum_{t=1}^{T} y_t \log \hat{y}_t$

Trainable parameters → $\theta = \{W_x, W_h, W_y\}$



# Recurrent Neural Networks (RNNs) [ Elman, 1991 ]

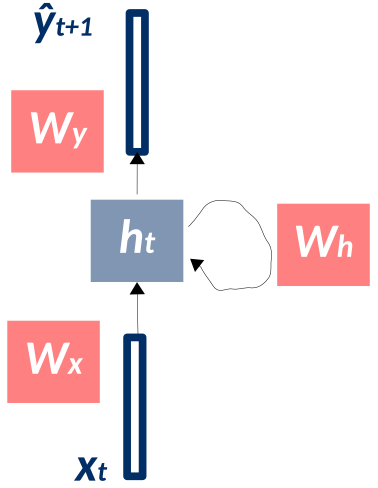
## Back-Prop. - Differentiation wrt. the parameters $\theta = \{W_x, W_h, W_y\}$

$$\boxed{\begin{aligned} h_t &= \tanh(\ W_h h_{t-1} + W_x x_t\ ) \\ p(y_{t+1}) &= softmax(\ W_y h_t\ ) \\ L_\theta(y, \hat{y})_t &= -y_t \log \hat{y}_t \end{aligned}}$$

### Differentiating wrt. $W_y$

$$\frac{\partial L_{\theta,t}}{\partial W_y} = \frac{\partial L_{\theta,t}}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial W_y}$$

$$= (y_t - \hat{y}_t) h_t$$



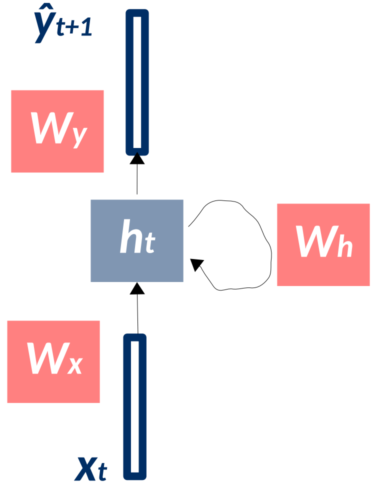
# Recurrent Neural Networks (RNNs) [ Elman, 1991 ]

## Back-Prop. - Differentiation wrt. the parameters $\theta = \{W_x, W_h, W_y\}$

$$\boxed{\begin{aligned} h_t &= \tanh(\ W_h h_{t-1} + W_x x_t\ ) \\ p(y_{t+1}) &= softmax(\ W_y h_t\ ) \\ L_\theta(y, \hat{y})_t &= -y_t \log \hat{y}_t \end{aligned}}$$

**Differentiating wrt. *$W_h$***

$$\frac{\partial L_{\theta,t}}{\partial W_h} = \frac{\partial L_{\theta,t}}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \frac{\partial h_t}{\partial W_h}$$



# Recurrent Neural Networks (RNNs) [ Elman, 1991 ]

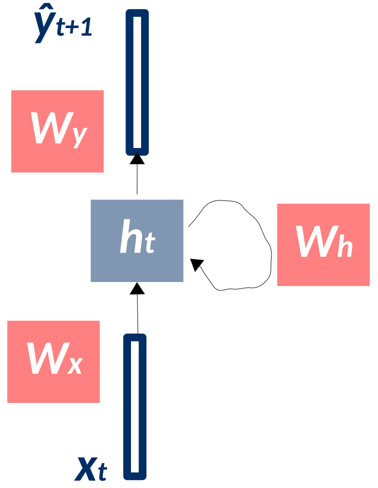
## Back-Prop. - Differentiation wrt. the parameters $\theta = \{W_x, W_h, W_y\}$

$$\boxed{\begin{aligned} h_t &= \tanh(\ W_h h_{t-1} + W_x x_t\ ) \\ p(y_{t+1}) &= softmax(\ W_y h_t\ ) \\ L_\theta(y, \hat{y})_t &= -y_t \log \hat{y}_t \end{aligned}}$$

### Differentiating wrt. *Wh*

$$\frac{\partial L_{\theta,t}}{\partial W_h} = \frac{\partial L_{\theta,t}}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \frac{\partial h_t}{\partial W_h}$$

$$\frac{\partial h_t}{\partial W_h} = \frac{\partial h_t}{\partial W_h} + \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial W_h}$$



# Recurrent Neural Networks (RNNs) [ Elman, 1991 ]

## Back-Prop. - Differentiation wrt. the parameters $\theta = \{W_x, W_h, W_y\}$

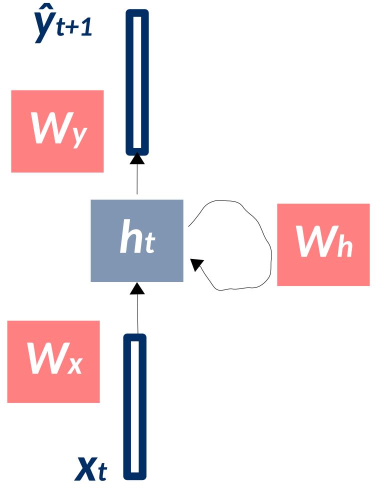
$$\boxed{\begin{aligned} h_t &= \tanh(\ W_h h_{t-1} + W_x x_t\ ) \\ p(y_{t+1}) &= softmax(\ W_y h_t\ ) \\ L_\theta(y, \hat{y})_t &= -y_t \log \hat{y}_t \end{aligned}}$$

**Differentiating wrt. *W*h**

$$\frac{\partial L_{\theta,t}}{\partial W_h} = \frac{\partial L_{\theta,t}}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \frac{\partial h_t}{\partial W_h}$$

$$\frac{\partial h_t}{\partial W_h} = \frac{\partial h_t}{\partial W_h} + \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial W_h}$$

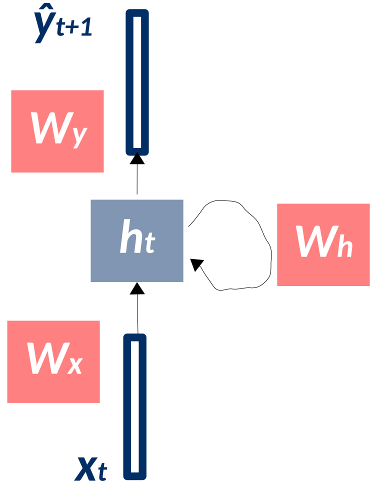
$$= \frac{\partial h_t}{\partial W_h} + \frac{\partial h_t}{\partial h_{t-1}} \left[ \frac{\partial h_{t-1}}{\partial W_h} + \frac{\partial h_{t-1}}{\partial h_{t-2}} \frac{\partial h_{t-2}}{\partial W_h} \right]$$



# Recurrent Neural Networks (RNNs) [ Elman, 1991 ]

## Back-Prop. - Differentiation wrt. the parameters $\theta = \{W_x, W_h, W_y\}$

$$\boxed{\begin{aligned} h_t &= \tanh(\ W_h h_{t-1} + W_x x_t\ ) \\ p(y_{t+1}) &= softmax(\ W_y h_t\ ) \\ L_\theta(y, \hat{y})_t &= -y_t \log \hat{y}_t \end{aligned}}$$




### Differentiating wrt. *Wh*

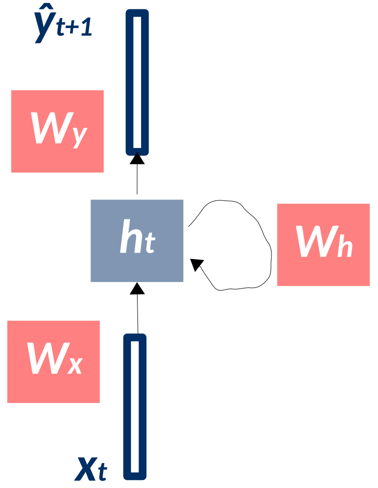
$$\frac{\partial L_{\theta,t}}{\partial W_h} = \frac{\partial L_{\theta,t}}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \frac{\partial h_t}{\partial W_h}$$

$$\begin{aligned} \frac{\partial h_t}{\partial W_h} &= \frac{\partial h_t}{\partial W_h} + \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial W_h} \\ &= \frac{\partial h_t}{\partial W_h} + \frac{\partial h_t}{\partial h_{t-1}} \left[ \frac{\partial h_{t-1}}{\partial W_h} + \frac{\partial h_{t-1}}{\partial h_{t-2}} \frac{\partial h_{t-2}}{\partial W_h} \right] \\ &= \sum_{k=1}^{t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W_h} \end{aligned}$$

# Recurrent Neural Networks (RNNs) [ Elman, 1991 ]

## Back-Prop. - Differentiation wrt. the parameters $\theta = \{W_x, W_h, W_y\}$

$$\boxed{\begin{aligned} h_t &= \tanh(\ W_h h_{t-1} + W_x x_t\ ) \\ p(y_{t+1}) &= softmax(\ W_y h_t\ ) \\ L_\theta(y, \hat{y})_t &= -y_t \log \hat{y}_t \end{aligned}}$$




### Differentiating wrt. $W_h$

$$\frac{\partial L_{\theta,t}}{\partial W_h} = \frac{\partial L_{\theta,t}}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \frac{\partial h_t}{\partial W_h}$$

**Back-propagation through time**

$$\begin{aligned} \frac{\partial h_t}{\partial W_h} &= \frac{\partial h_t}{\partial W_h} + \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial W_h} \\ &= \frac{\partial h_t}{\partial W_h} + \frac{\partial h_t}{\partial h_{t-1}} \left[ \frac{\partial h_{t-1}}{\partial W_h} + \frac{\partial h_{t-1}}{\partial h_{t-2}} \frac{\partial h_{t-2}}{\partial W_h} \right] \\ &= \sum_{k=1}^{t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W_h} \end{aligned}$$

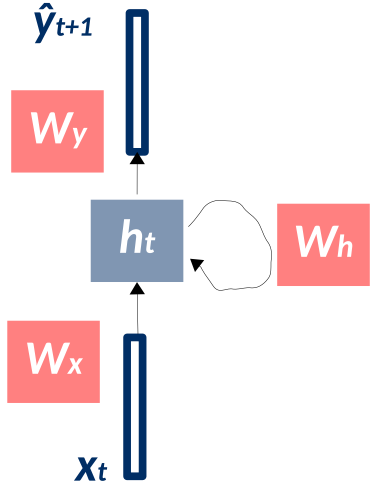
# Recurrent Neural Networks (RNNs) [ Elman, 1991 ]

## Back-Prop. - Differentiation wrt. the parameters $\theta = \{W_x, W_h, W_y\}$

$$\boxed{\begin{aligned} h_t &= \tanh(\ W_h h_{t-1} + W_x x_t\ ) \\ p(y_{t+1}) &= softmax(\ W_y h_t\ ) \\ L_\theta(y, \hat{y})_t &= -y_t \log \hat{y}_t \end{aligned}}$$

**Differentiating wrt. *Wh***

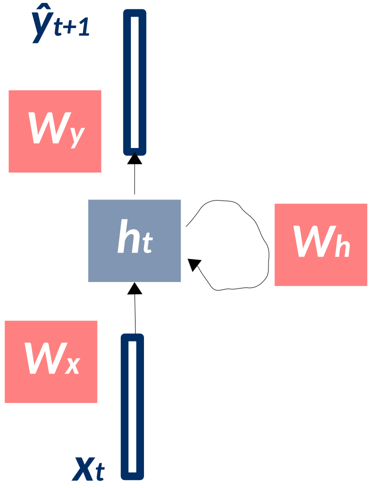
$$\frac{\partial L_{\theta,t}}{\partial W_h} = \frac{\partial L_{\theta,t}}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \frac{\partial h_t}{\partial W_h}$$



# Recurrent Neural Networks (RNNs) [ Elman, 1991 ]

## Back-Prop. - Differentiation wrt. the parameters $\theta = \{W_x, W_h, W_y\}$

$$\boxed{\begin{aligned} h_t &= \tanh(\ W_h h_{t-1} + W_x x_t\ ) \\ p(y_{t+1}) &= softmax(\ W_y h_t\ ) \\ L_\theta(y, \hat{y})_t &= -y_t \log \hat{y}_t \end{aligned}}$$




### Differentiating wrt. $W_h$

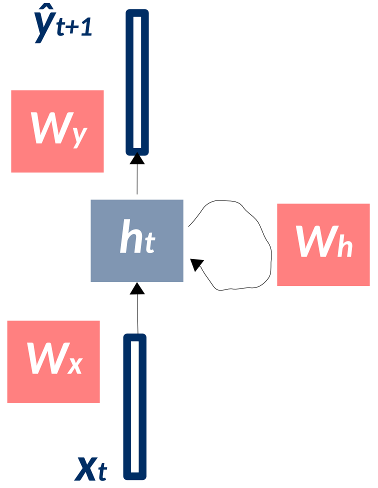
$$\frac{\partial L_{\theta,t}}{\partial W_h} = \frac{\partial L_{\theta,t}}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \frac{\partial h_t}{\partial W_h}$$

$$= \frac{\partial L_{\theta,t}}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \sum_{k=1}^{t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W_h}$$

# Recurrent Neural Networks (RNNs) [ Elman, 1991 ]

## Back-Prop. - Differentiation wrt. the parameters $\theta = \{W_x, W_h, W_y\}$

$$h_t = \tanh(\ W_h h_{t-1} + W_x x_t\ )$$
$$p(y_{t+1}) = softmax(\ W_y h_t\ )$$
$$L_\theta(y, \hat{y})_t = -y_t \log \hat{y}_t$$




**Differentiating wrt. *Wh***

$$\frac{\partial L_{\theta,t}}{\partial W_h} = \frac{\partial L_{\theta,t}}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \frac{\partial h_t}{\partial W_h}$$

$$= \frac{\partial L_{\theta,t}}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \sum_{k=1}^{t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W_h}$$

$$= \sum_{k=1}^{t} \frac{\partial L_{\theta,t}}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W_h}$$

# Recurrent Neural Networks (RNNs) [ Elman, 1991 ]

## Summarizing

- + Good at modeling sequences of variable length
- + Trainable via Back-Prop. → differentiable

- – Suffer from vanishing gradients for long sequences

# Long Short-Term Memory Networks

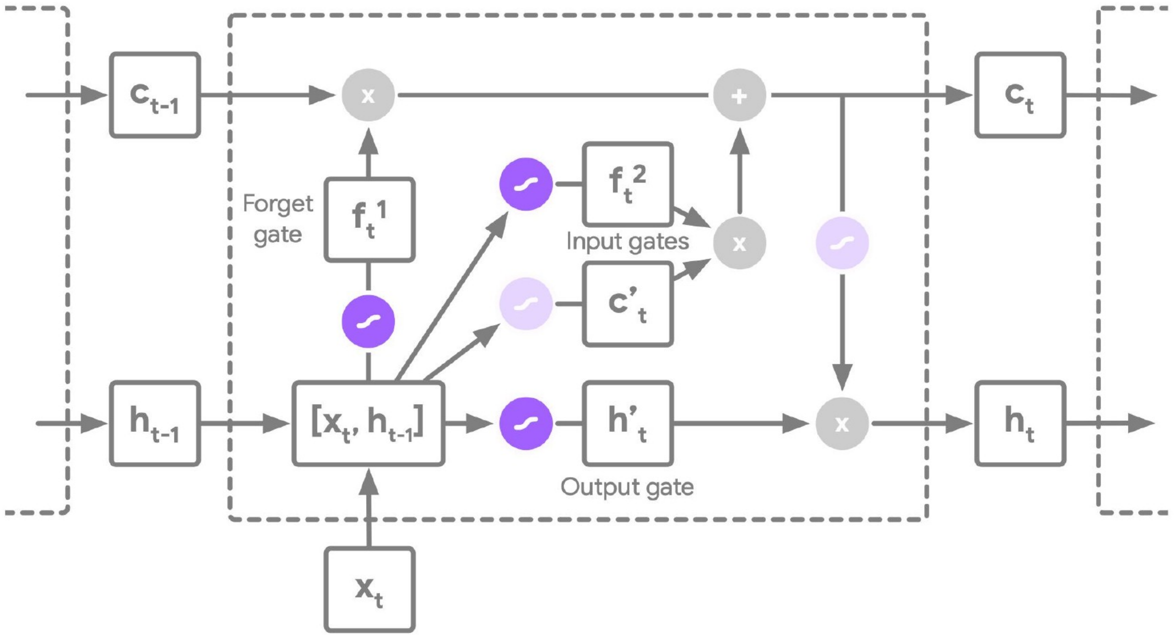
[ RNNs with Extra Memory ]

# Long Short-Term Memory Networks (LSTMs)

[ Hochreiter & Jurgen , 1997 ]

## Provide the capability of choosing what to remember/forget

- **Idea:** Provide special gates to control the flow of “memories”

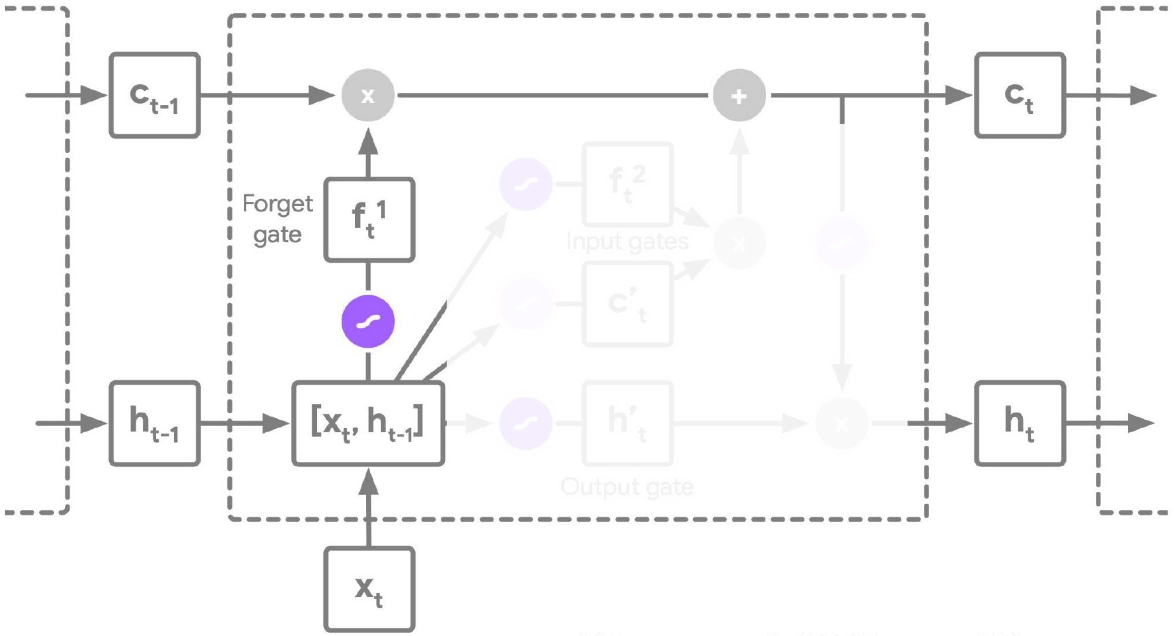


# Long Short-Term Memory Networks (LSTMs)

[ Hochreiter & Jurgen , 1997 ]

## Provide the capability of choosing what to remember/forget

- $f_1$: Forget Gate



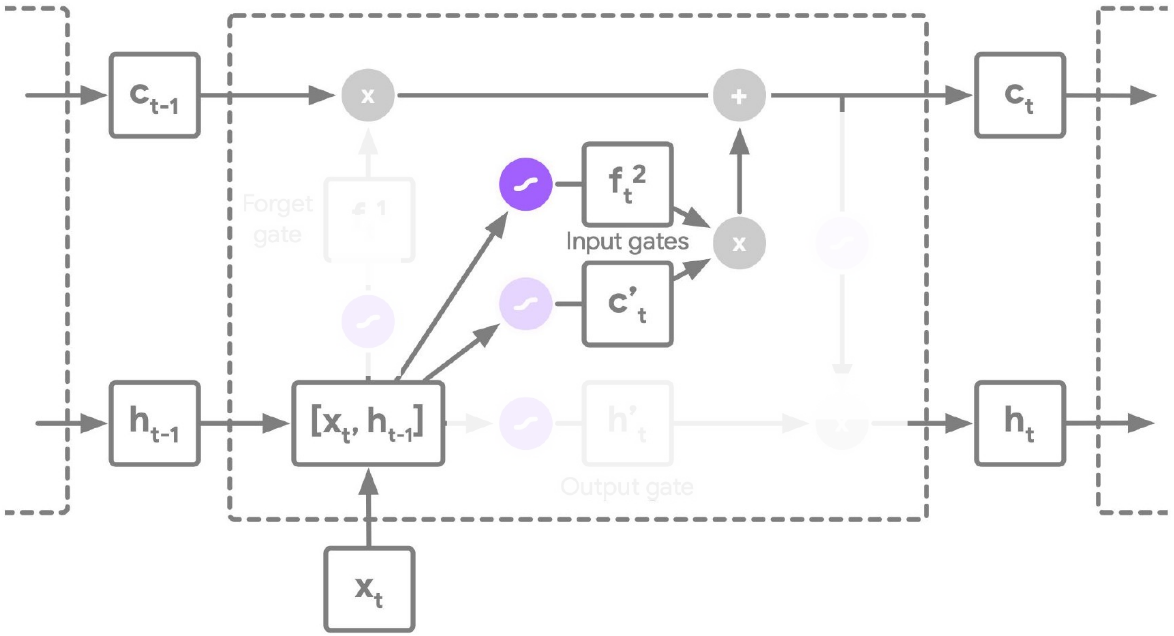

regulate what information to keep/ignore

$$f_t^1 = \sigma(\ W_{f^1} \cdot [h_{t-1}, x_t] + b_{f^1}\ )$$

# Long Short-Term Memory Networks (LSTMs)
[ Hochreiter & Jurgen , 1997 ]

## Provide the capability of choosing what to remember/forget

- $f_2$: Input Gate



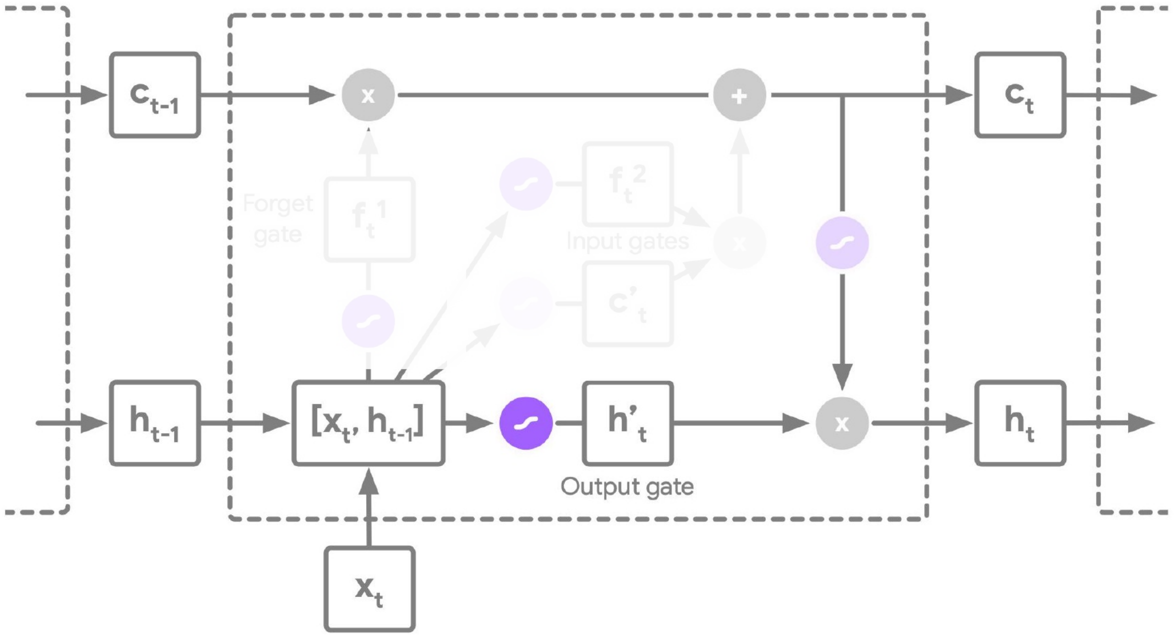

Decides what information to update

$$= \sigma(\ W_{f^2} \cdot [h_{t-1}, x_t] + b_{f^2}\ ) \odot \tanh(\ W_{c'}[h_{t-1}, x_t] + b_{c'}\ )$$

# Long Short-Term Memory Networks (LSTMs)
[ Hochreiter & Jurgen , 1997 ]

## Provide the capability of choosing what to remember/forget

- h’t: Output Gate



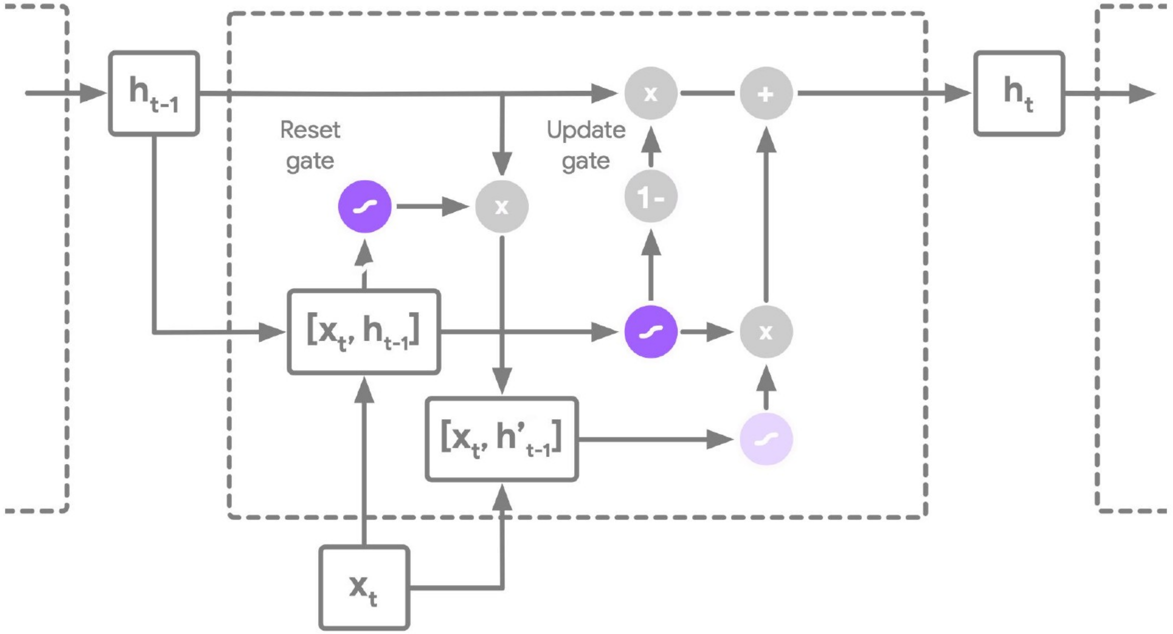

$$= \sigma(\ W_{h'_t} \cdot [h_{t-1}, x_t] + b_{h'_t}\ ) \odot \tanh(\ c_t\ )$$

University of Antwerp

# Gated Recurrent Units (GRUs) [ Cho et al. , 2014 ]

## A Simplifed LSTM Network

- **Idea:** Provide special gates to control the flow of “memories”



# LSTMs and GRUs

## Summarizing

- + Good at modeling sequences of variable length
- + Trainable via Back-Prop. → differentiable

- + Capable of handling long sequences

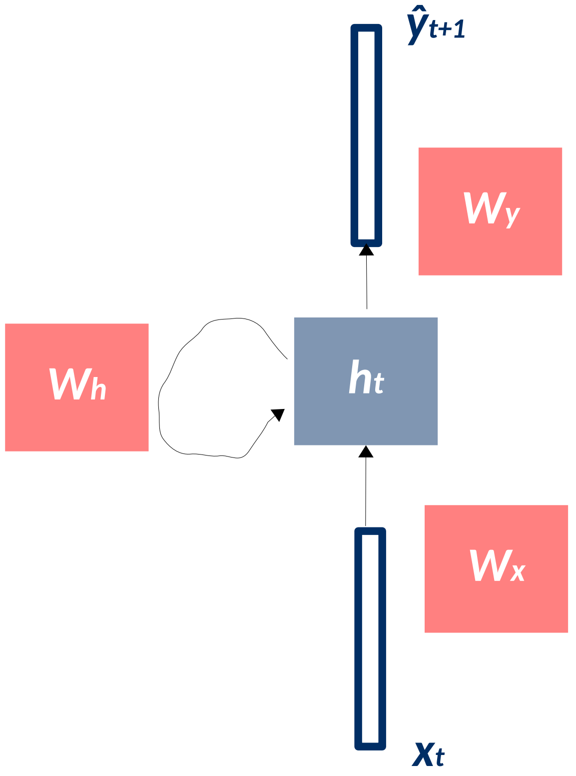
  ( robust to vanishing/exploding gradients )

# Predictions from Sequences

# Predictions from Sequences

## Training Classifiers/Regressors from Sequences

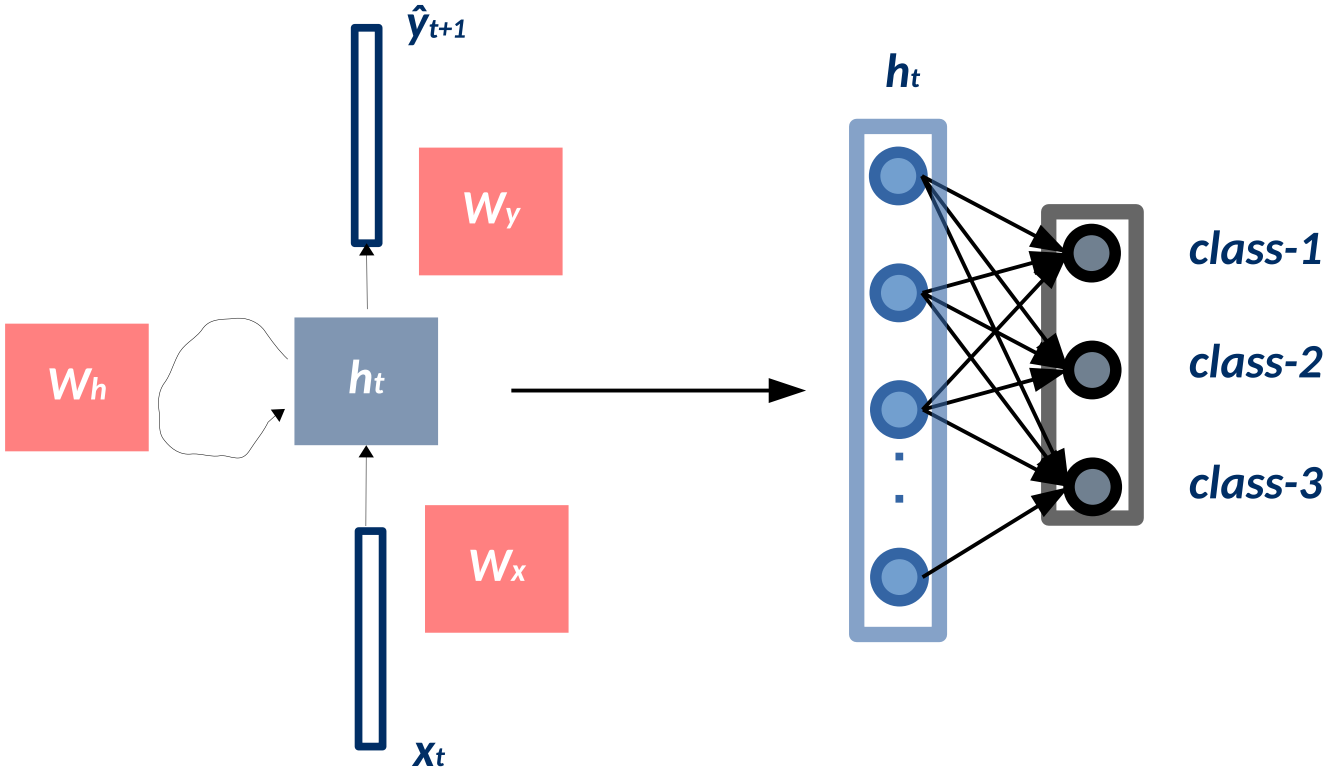
- 1) Attach a related head (classification, regression, etc.) to the *persistent* state
- 2) Measure the loss wrt. the prediction task



# Predictions from Sequences

## Training Classifiers/Regressors from Sequences

- 1) Attach a related head (classification, regression, etc.) to the *persistent* state
- 2) Measure the loss wrt. the prediction task

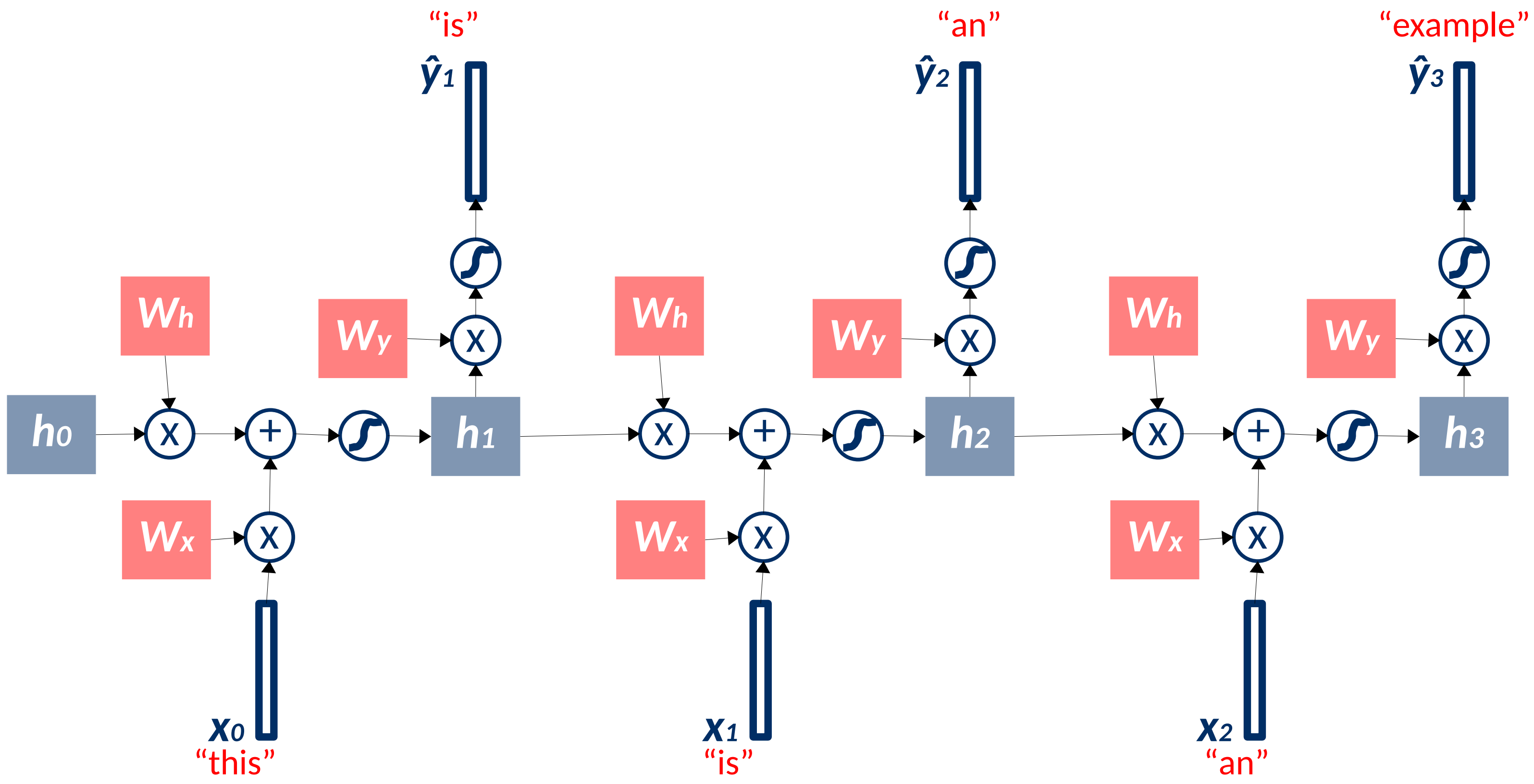


# Generating Sequences

# Generating Sequences

## Sample the next best element from the predicted distribution

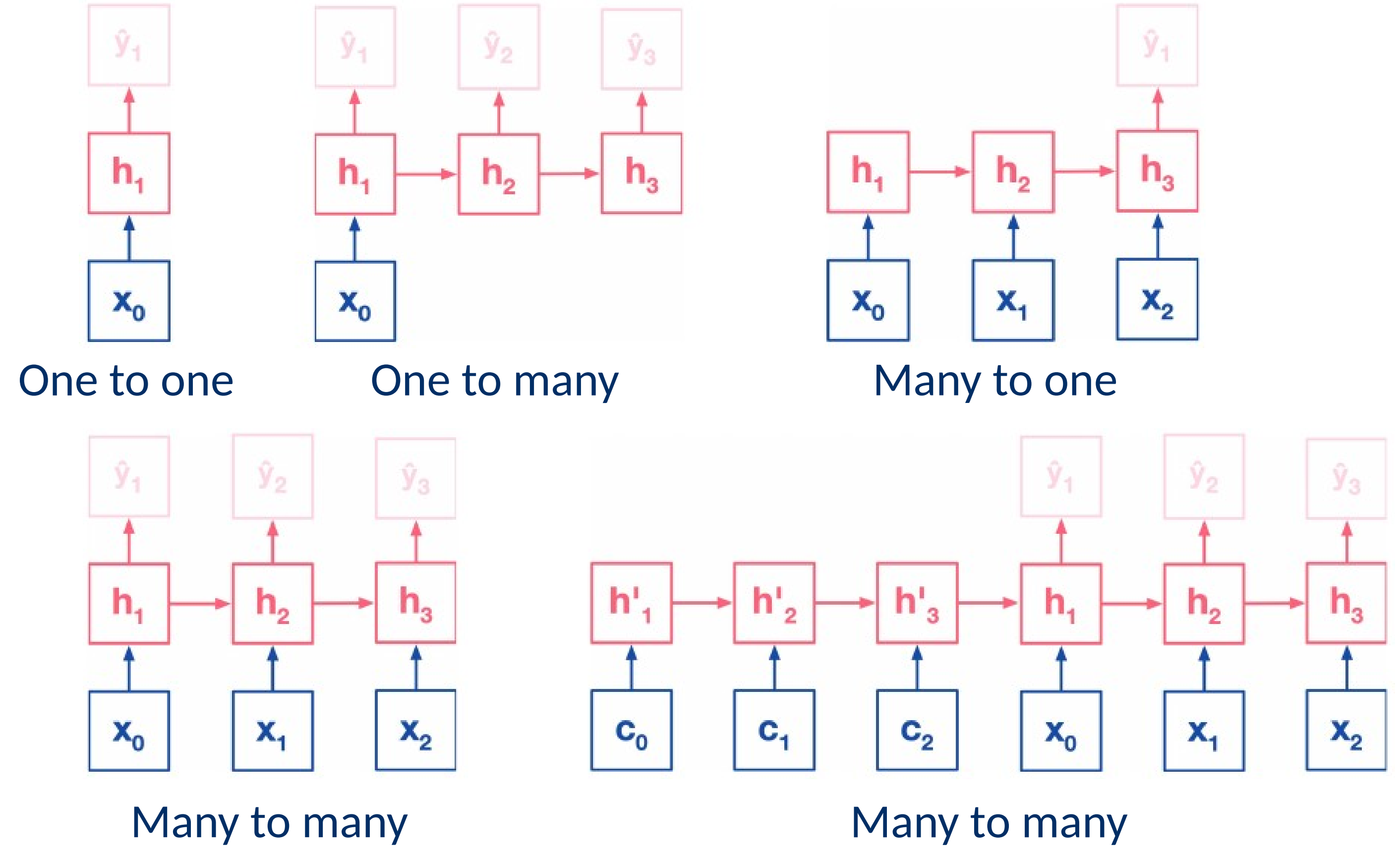
- **Idea:** Use the predicted element $\hat{y}_t$ as input in the next iteration



# Generating Sequences

## Several options are possible – beyond text sequences

- **Idea:** Different ways to define inputs, context and outputs




**Some Applications**

- Language Translation
- Speech-to-Text
- Contextual Search
- Image Captioning

# Break

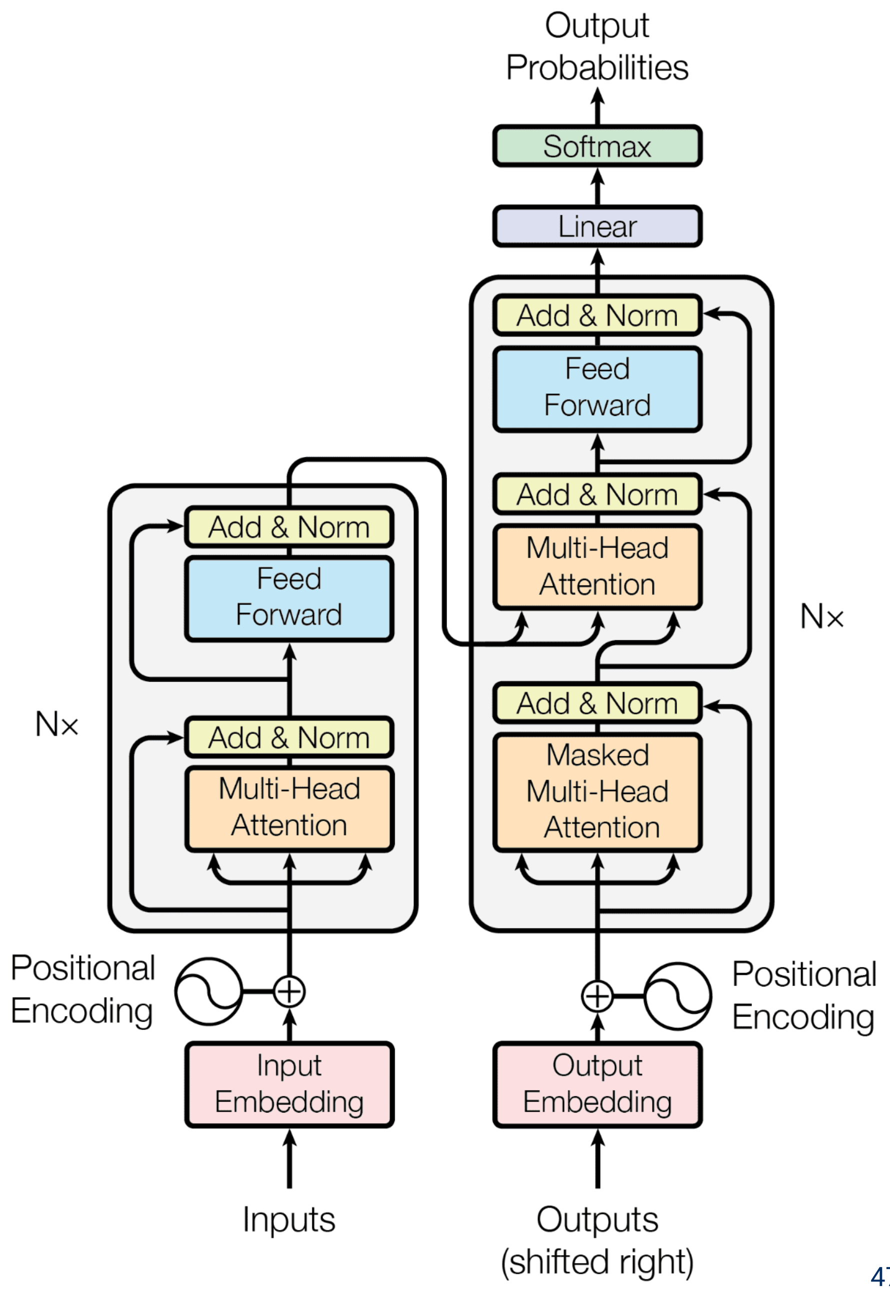
[ Let's meet again in 15 mins. ]

# Transformers

# Transformers [Vaswani et al., 2017]

## Some specs

- Removed Recurrence components
- Based solely on the attention mechanism
- Originally addressed translation tasks
  [English-German | English-French]




University of Antwerp

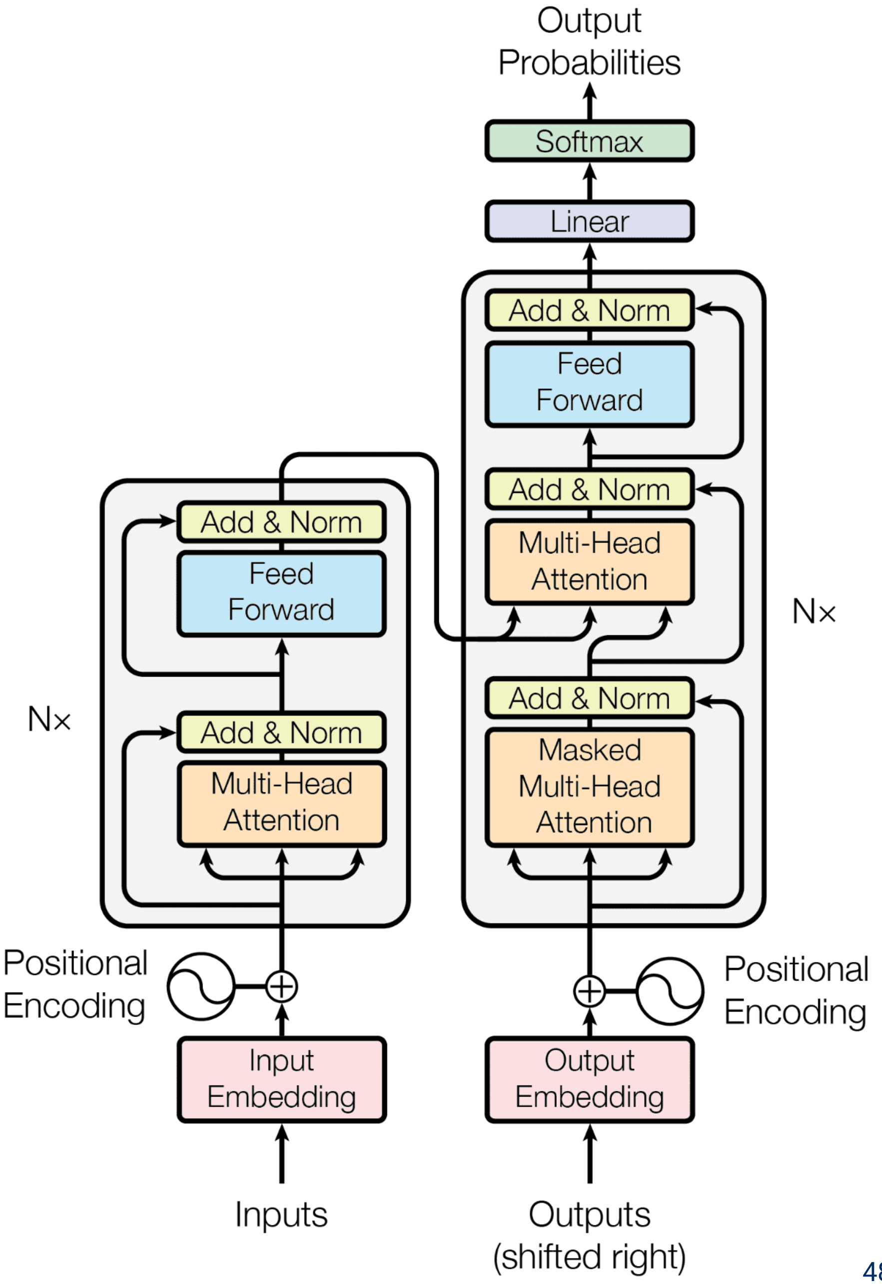
# Transformers [Vaswani et al., 2017]

## Some specs

- Removed Recurrence components
- Based solely on the attention mechanism
- Originally addressed translation tasks

  [English-German | English-French]

**Scary-looking yes, difficult not**
Let's follow Thomas' presentation



# Modeling Data Sequences with Transformers

## Lets consider the following sequence

**“I like chocolate”**

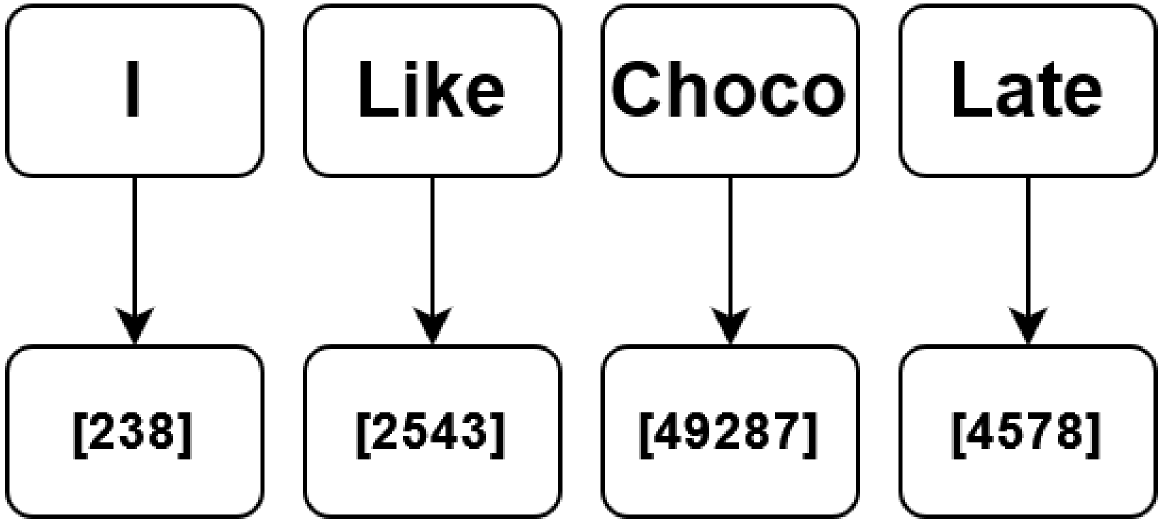


[Dooms, 2024]

# Modeling Data Sequences with Transformers

## Tokenization: defining granular unit of processing

- Break the input into smaller units (*tokens*)
- Different levels of codification possible (character, word, etc.)



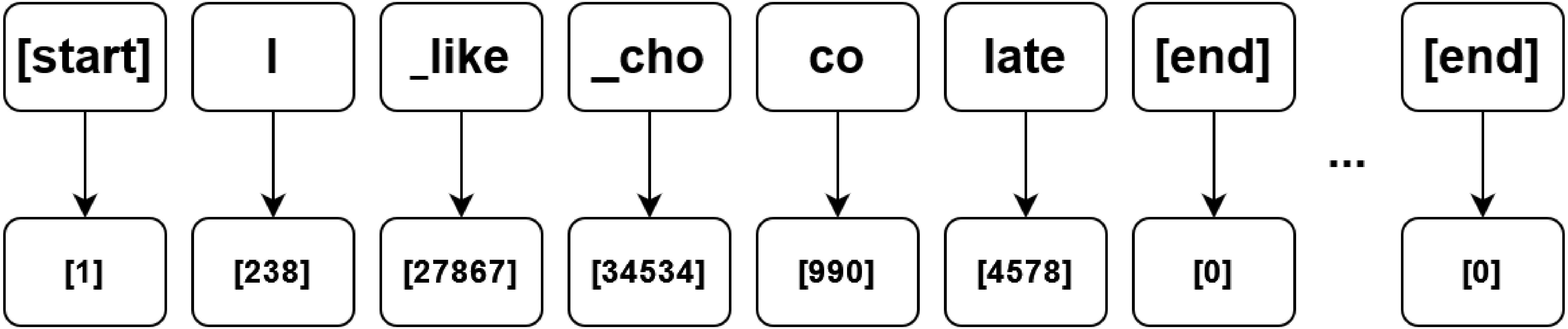

[Dooms, 2024]

# Modeling Data Sequences with Transformers

## Tokenization: defining granular unit of processing

- Break the input into smaller units (*tokens*)
- Different levels of codification possible (character, word, etc.)

**In reality it is not that straight-forward*

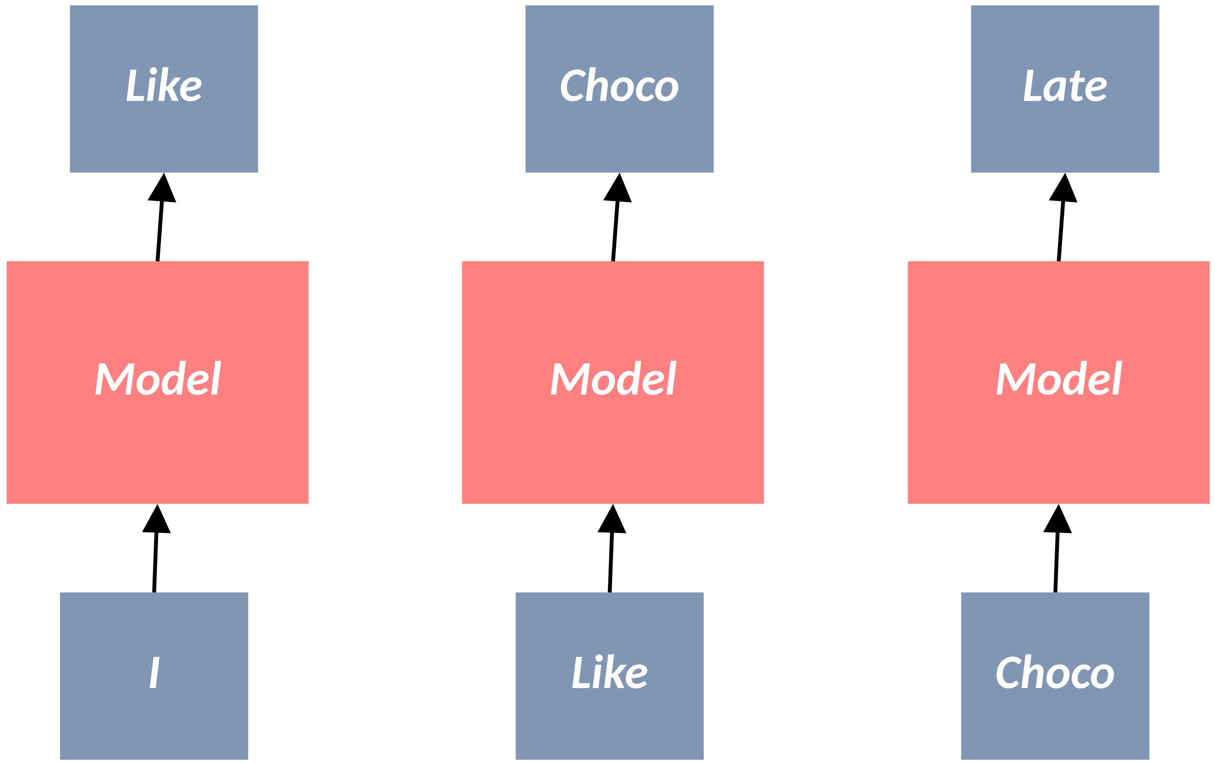





[Dooms, 2024]

# Modeling Data Sequences with Transformers

## Predicting the next element (token)

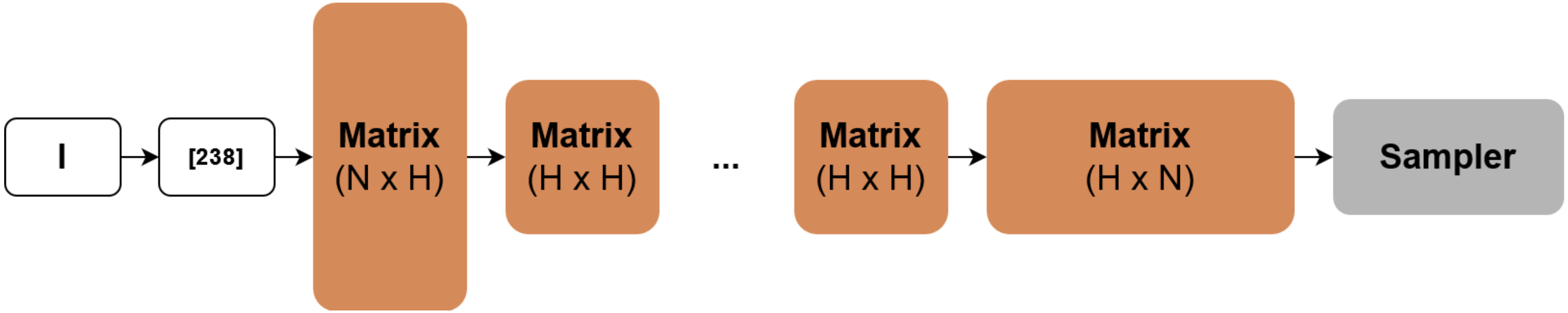



[Dooms, 2024]

# Sequence Modelling with Transformers

## A Very Popular Recipe

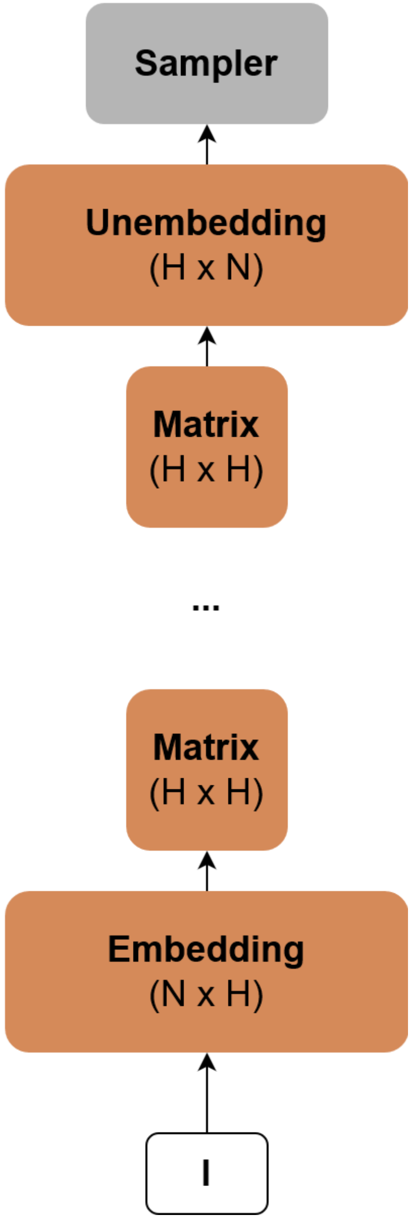
- Moving to a lower dimensional space (determined by ***H***)



# Sequence Modelling with Transformers

## A Very Popular Recipe

- Moving to a lower dimensional space (determined by ***H***)

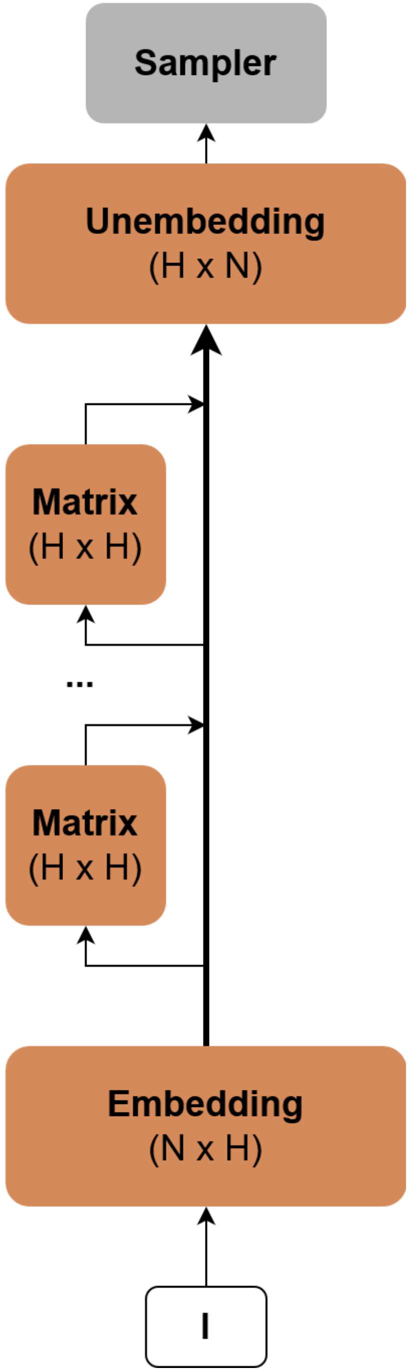



[Dooms, 2024]

# Sequence Modelling with Transformers

## A Very Popular Recipe

- Moving to a lower dimensional space (determined by ***H***)
- Using residual layers (why?)

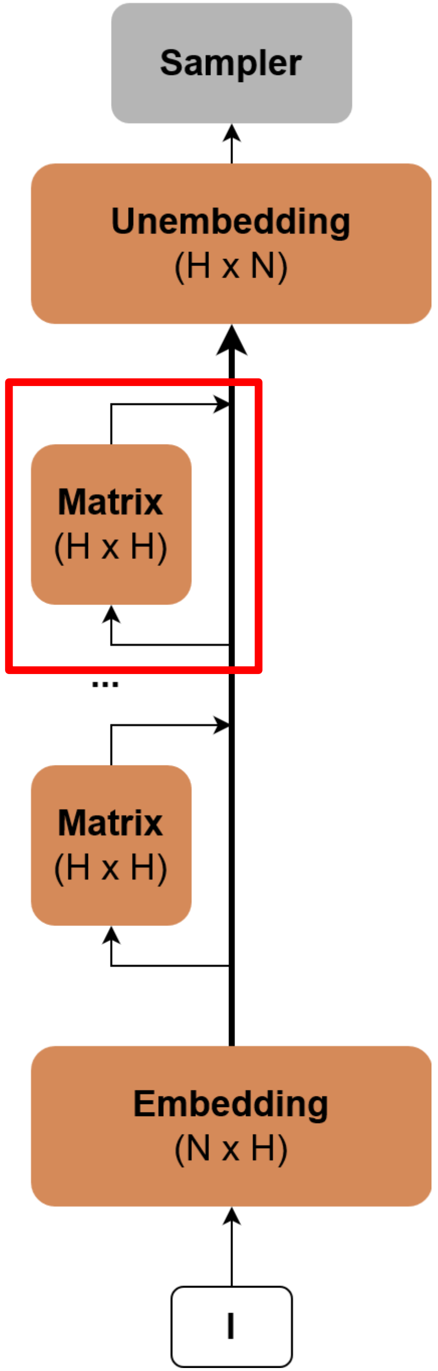



[Dooms, 2024]

# Sequence Modelling with Transformers

## A Very Popular Recipe

- Moving to a lower dimensional space (determined by ***H***)
- Using residual layers (why?)

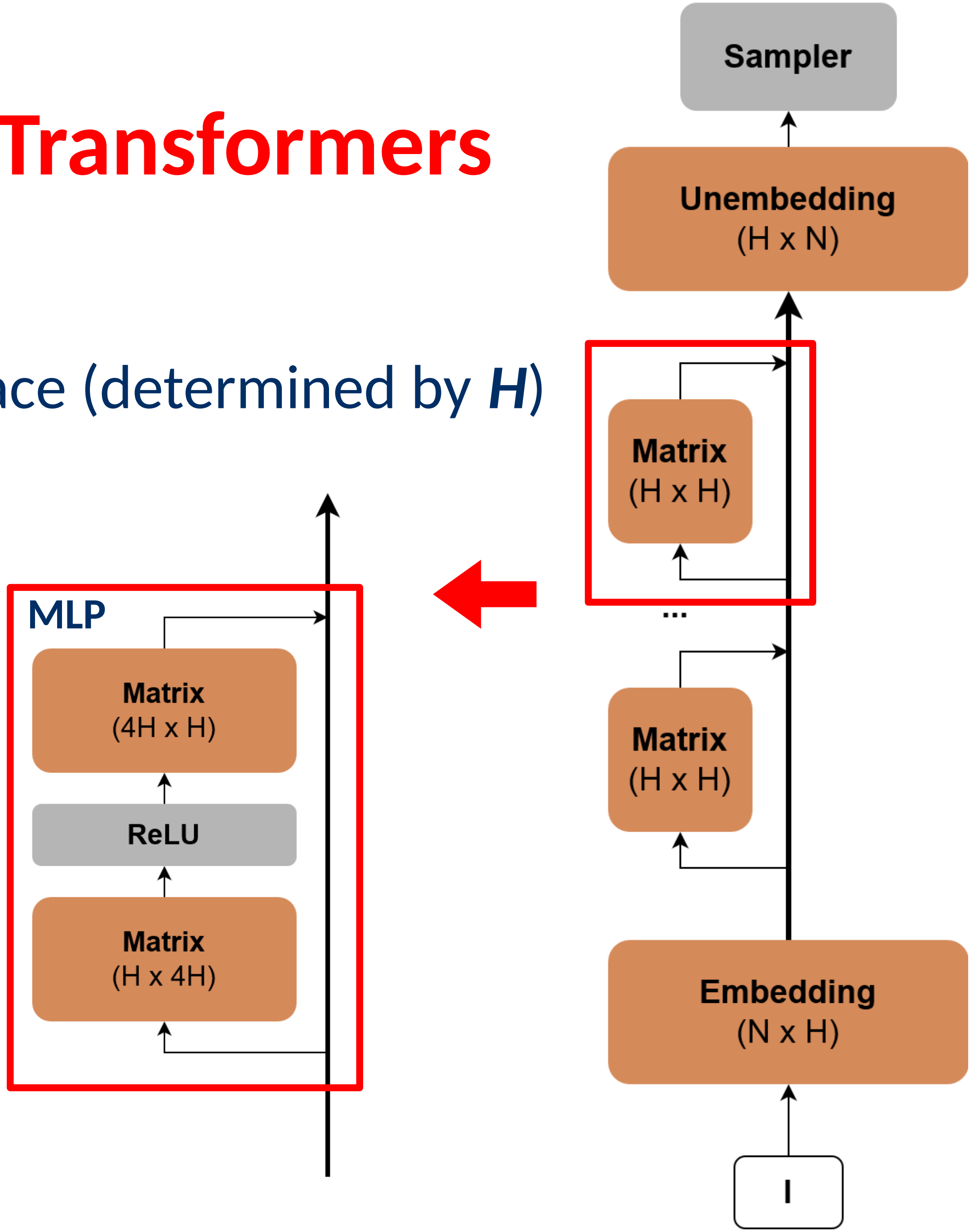



[Dooms, 2024]

# Sequence Modelling with Transformers

## A Very Popular Recipe

- Moving to a lower dimensional space (determined by ***H***)
- Using residual layers (why?) - MLP

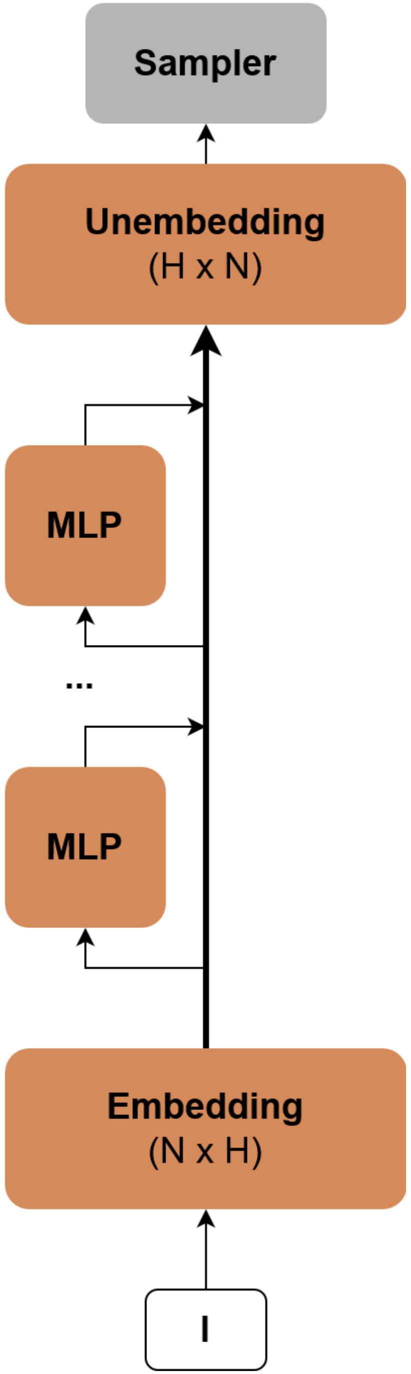



[Dooms, 2024]

# Sequence Modelling with Transformers

## A Very Popular Recipe

- Moving to a lower dimensional space (determined by ***H***)
- Using residual layers (why?) - MLP

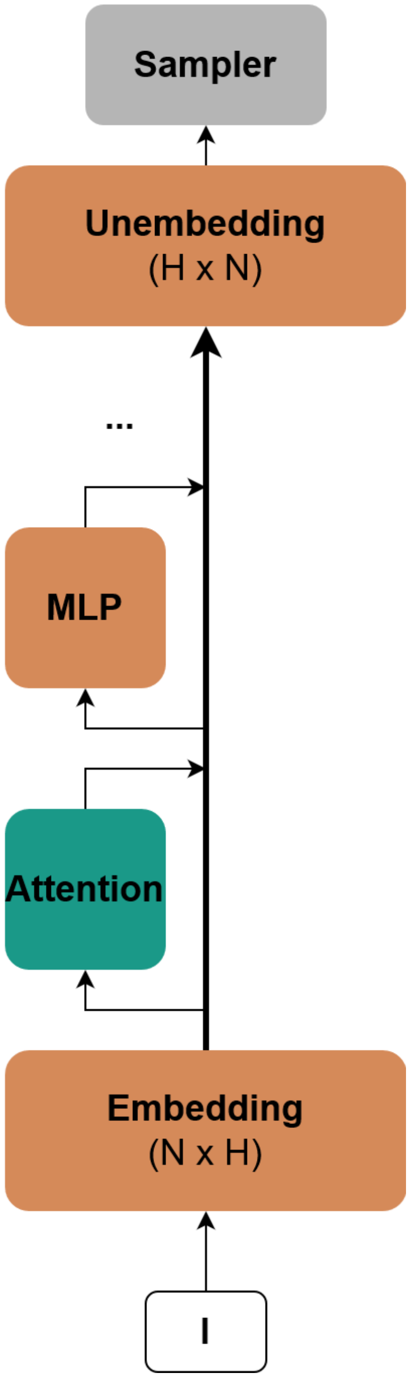



[Dooms, 2024]

# Sequence Modelling with Transformers

## A Very Popular Recipe

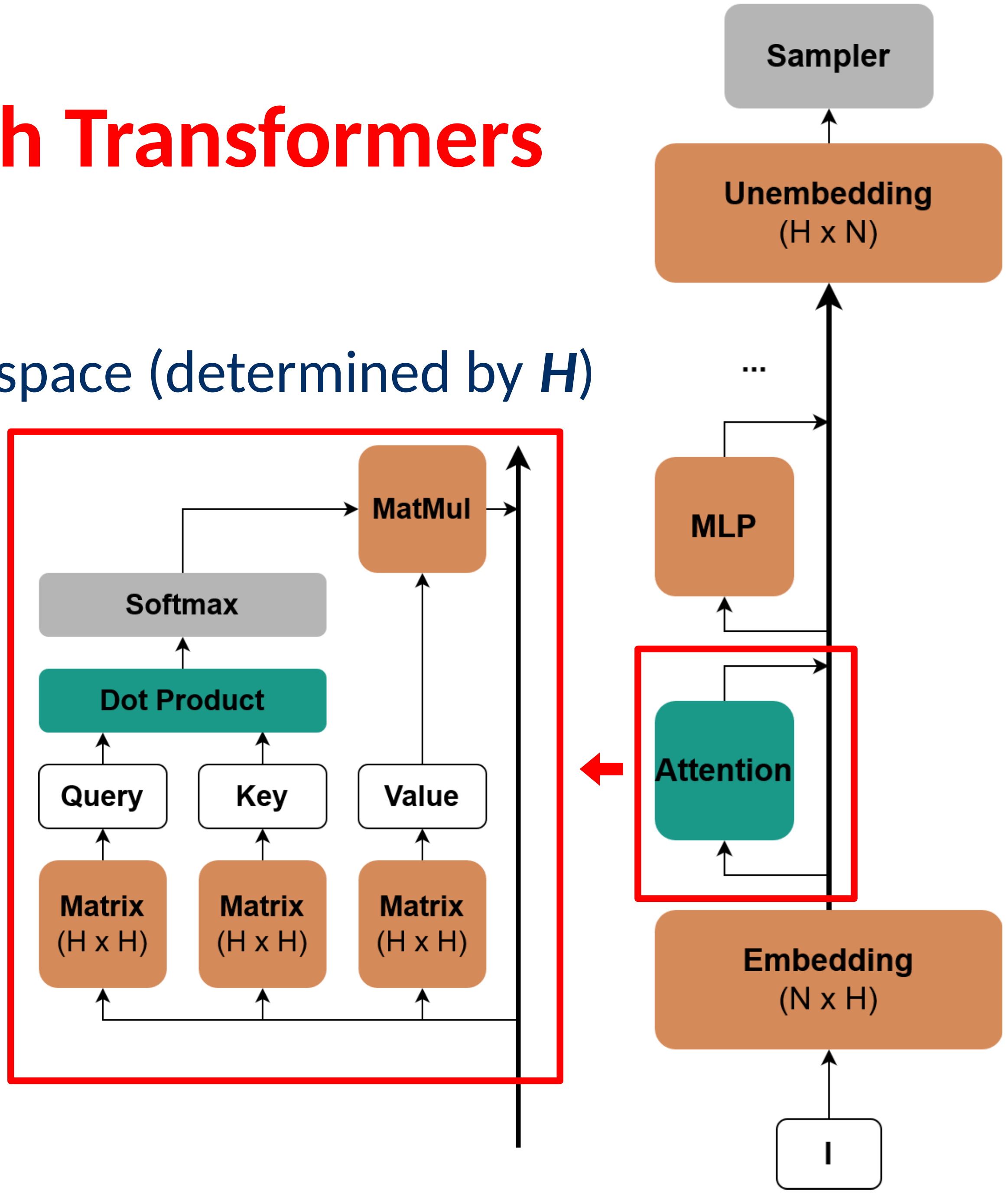
- Moving to a lower dimensional space (determined by ***H***)
- Using residual layers (why?) - MLP
- Add an [self] attention layer



# Sequence Modelling with Transformers

## A Very Popular Recipe

- Moving to a lower dimensional space (determined by ***H***)
- Using residual layers (why?)
- Add an [self] attention layer



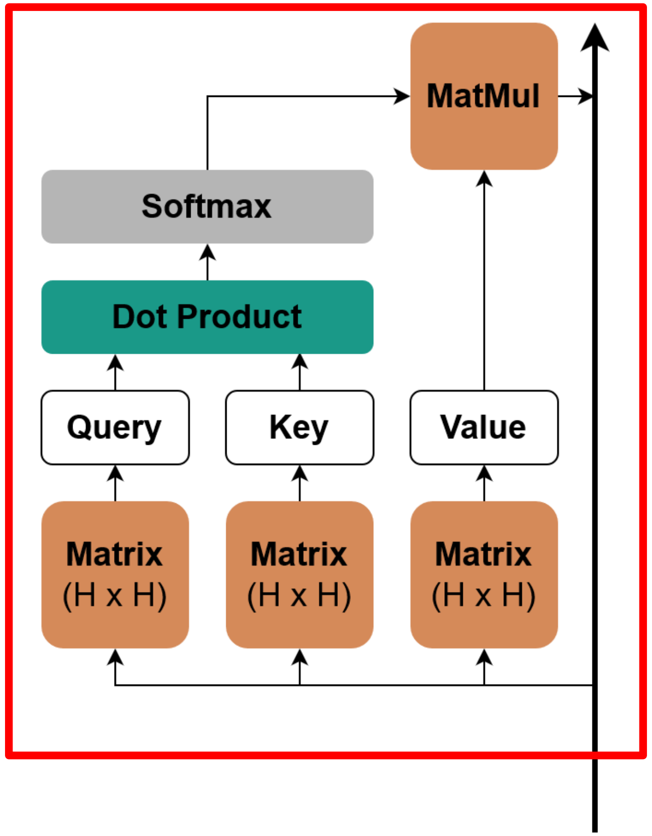

[Dooms, 2024]

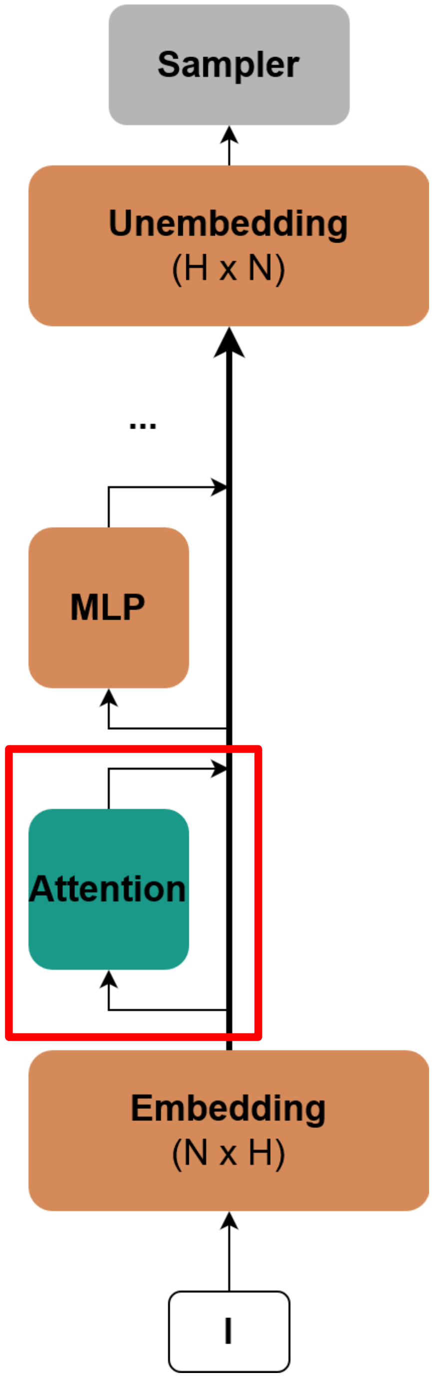
# Sequence Modelling with Transformers

## A Very Popular Recipe

- Moving to a lower dimensional space (determined by ***H***)
- Using residual layers (why?)
- Add an [self] attention layer

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

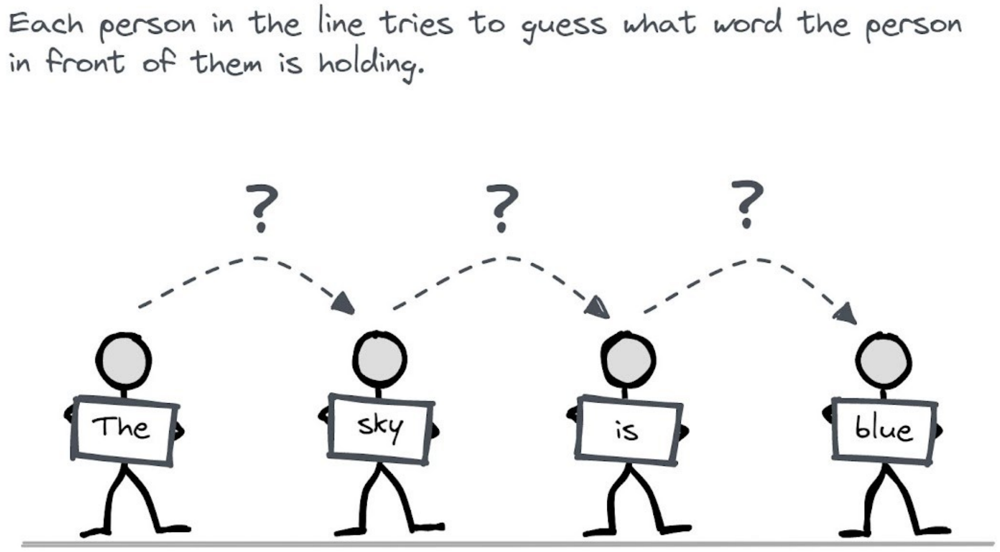



[Dooms, 2024]

# Sequence Modelling with Transformers

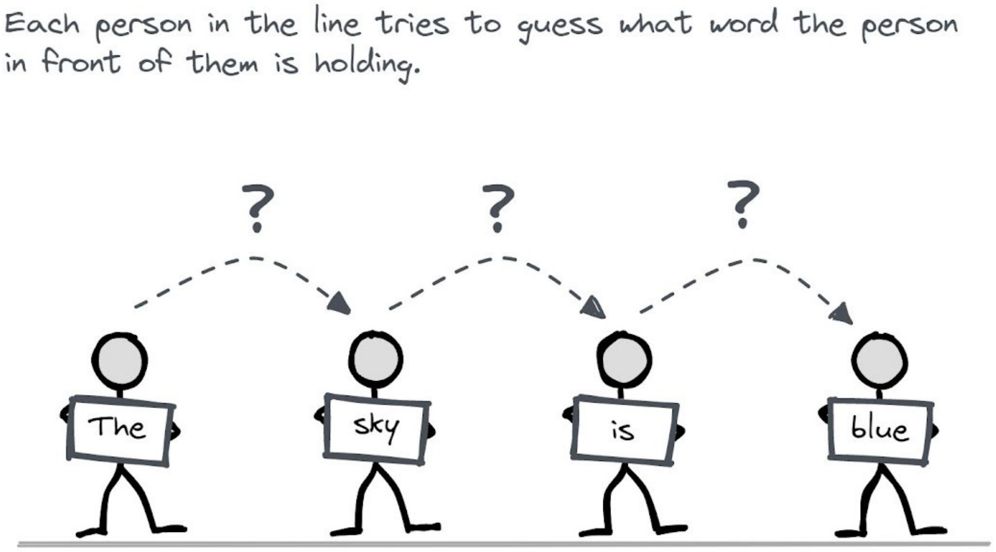
## Attention Mechanism – An Intuition

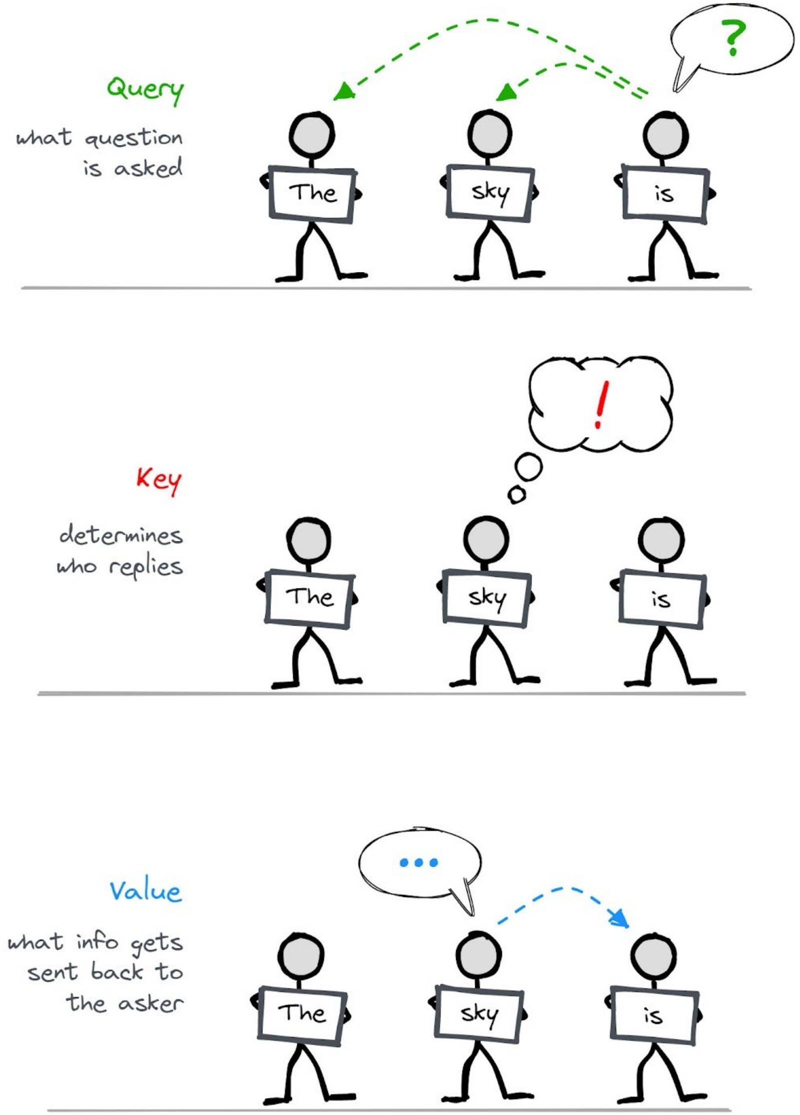



[Dooms, 2024]

# Sequence Modelling with Transformers

## Attention Mechanism – An Intuition



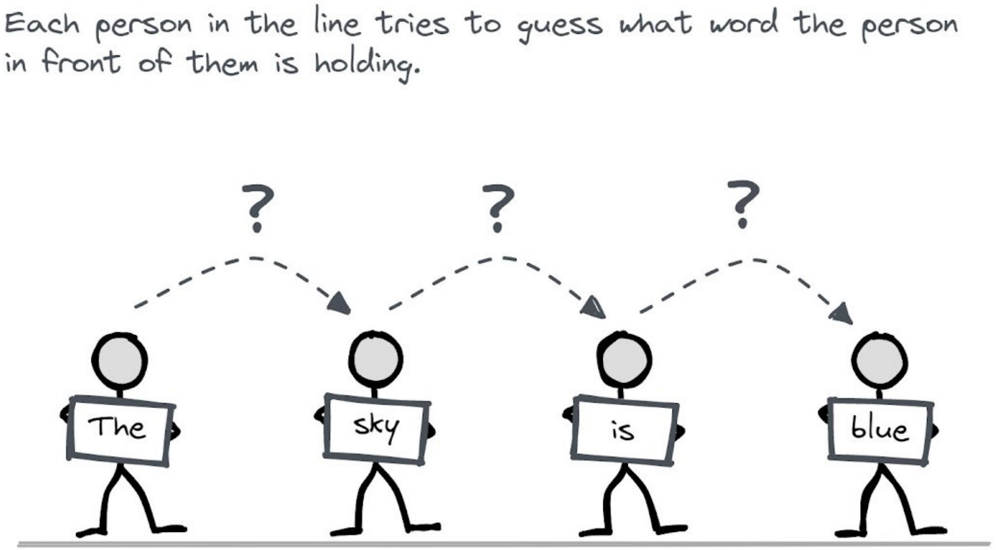



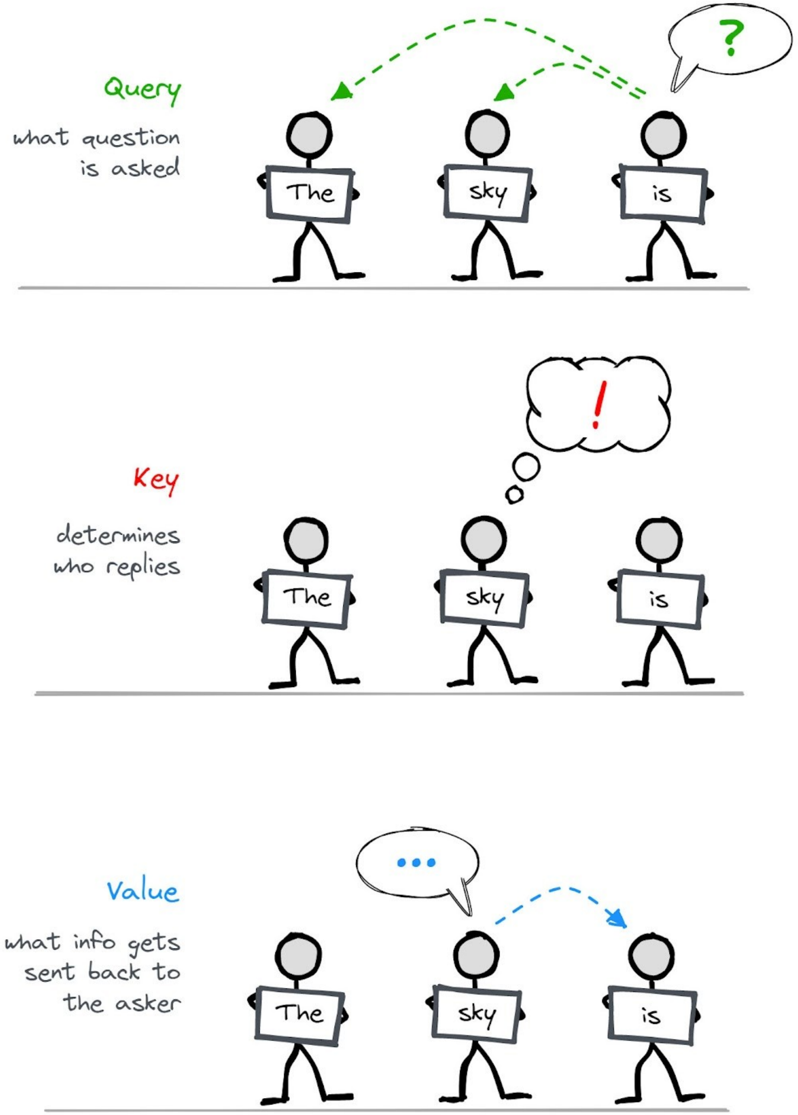

[Dooms, 2024]

# Sequence Modelling with Transformers

## Attention Mechanism – An Intuition

- Q → the question
- K → critical element to focus on
- V → the information to be forwarded




[Dooms, 2024]

# Sequence Modelling with Transformers

## Attention Mechanism – An Intuition

- Q → the question
- K → critical element to focus on
- V → the information to be forwarded

**A database Analogy**

| | Query = 5 | Values | Output |
|---|---|---|---|
| Key = 1 | 0 | 1 | 0 |
| Key = 5 | 1 | 2 | 2 |
| Key = 3 | 0 | 3 | 0 |
| Key = 4 | 0 | 4 | 0 |
| Key = 5 | 1 | 5 | 5 |
| | | | 7 |

[Dooms, 2024]

# Sequence Modelling with Transformers

## Attention Mechanism – An Intuition

- Q → the question
- K → critical element to focus on
- V → the information to be forwarded

**A database Analogy**

| | Query = 5 | Values | Output |
|---|---|---|---|
| Key = 1 | 0.125 | 1 | 0.125 |
| Key = 5 | 1 | 2 | 2 |
| Key = 3 | 0.25 | 3 | 0.75 |
| Key = 4 | 0.5 | 4 | 2 |
| Key = 5 | 1 | 5 | 5 |
| | | | 9.875 |

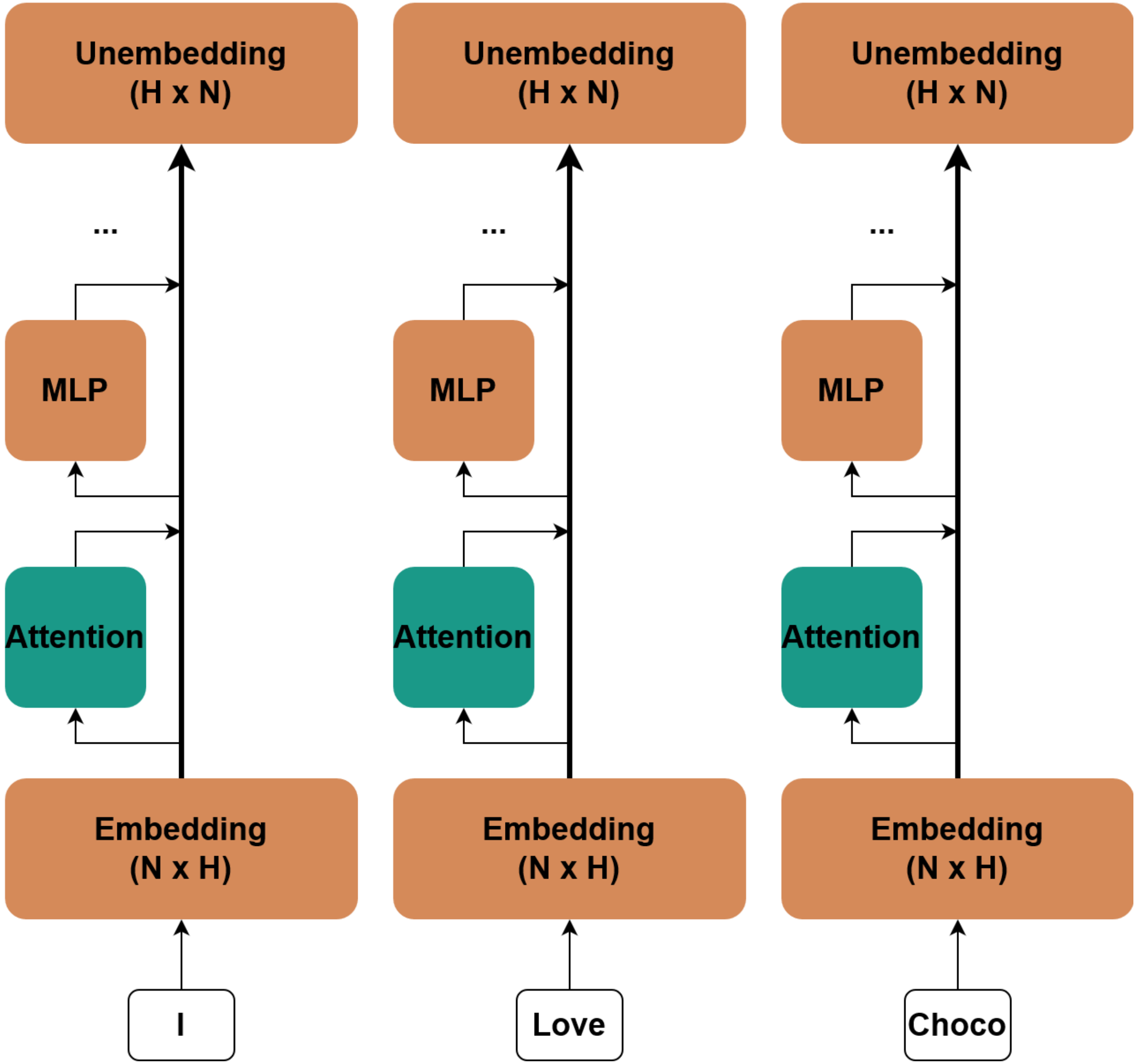
**Like convolution but with dynamic weight (query-key similarity)**

[Dooms, 2024]

# Sequence Modelling with Transformers

## Strengths: Communication

- Depth: more specialized and structured
- Through attention: across tokens





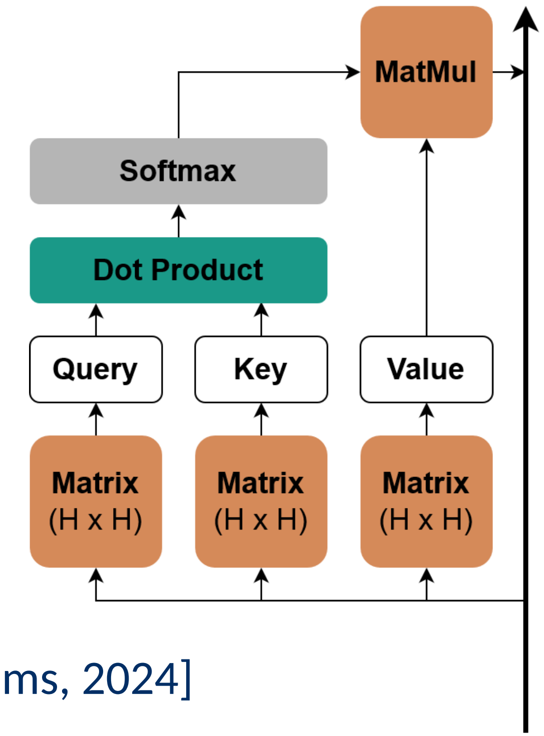

[Dooms, 2024]

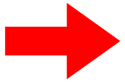
# Sequence Modelling with Transformers

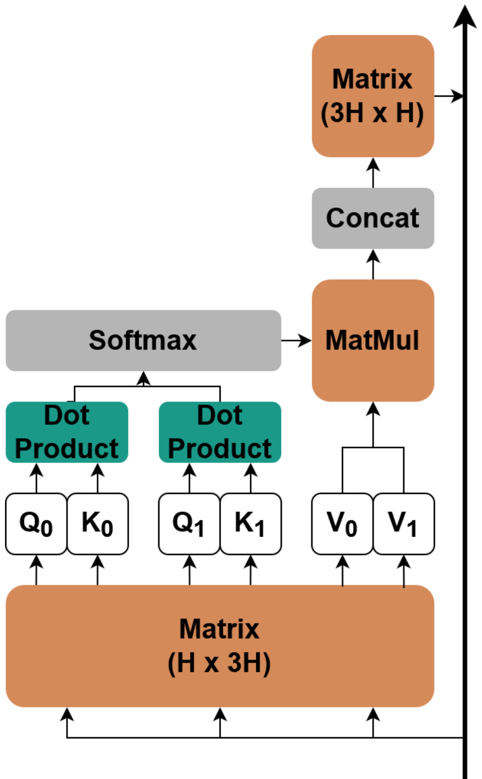
## Single vs. Multi-head Attention

- Single: Each layer ask one question via the single  QKV matrices.
- Multi: Split the QKV matrices into smaller ones → ask more/simpler questions

Single-head Attention






Multi-head Attention



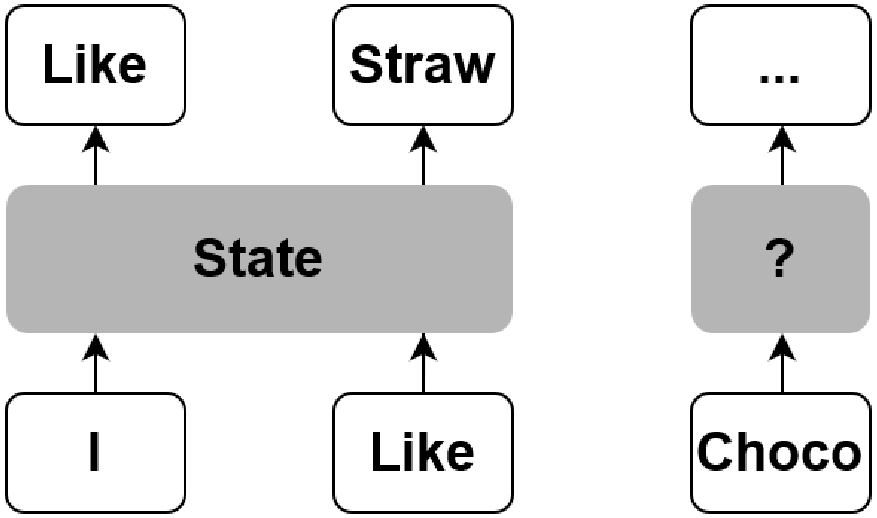
[Dooms, 2024]

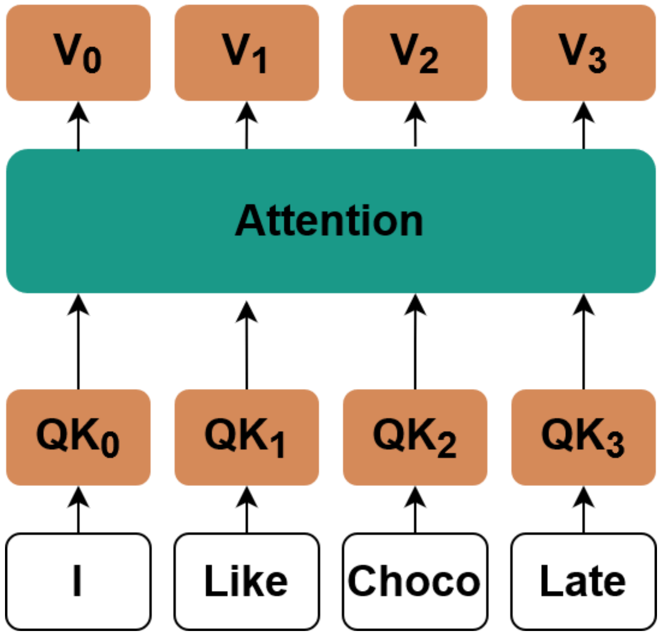
# RNNs VS Transfomers

## Context Sources

- Information from memory  vs. tokens (error propagation)

  → Sentence vs. Word -based



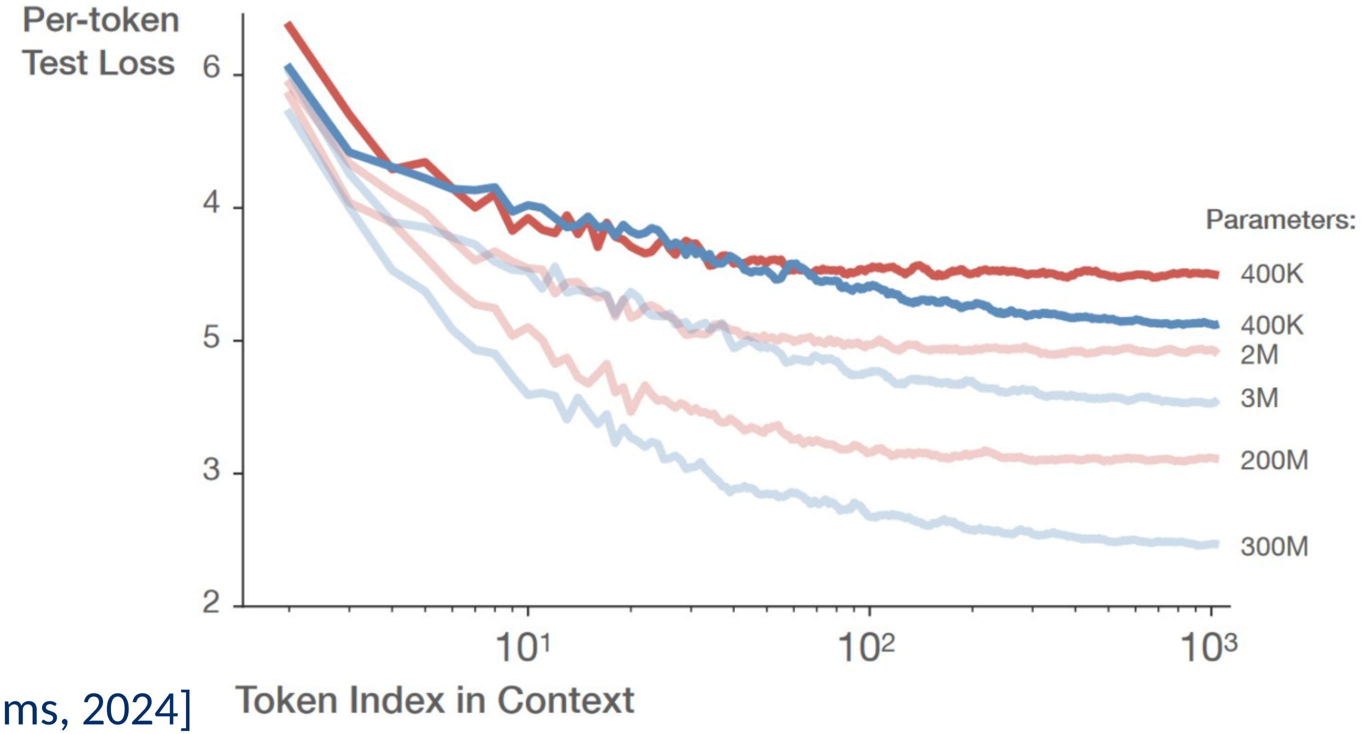

University of Antwerp

[Dooms, 2024]

# RNNs VS Transfomers

## Performance / size trade-off

- Higher performance as # of considered tokens increases
- Better use of parameters




[Kaplan, 2020]

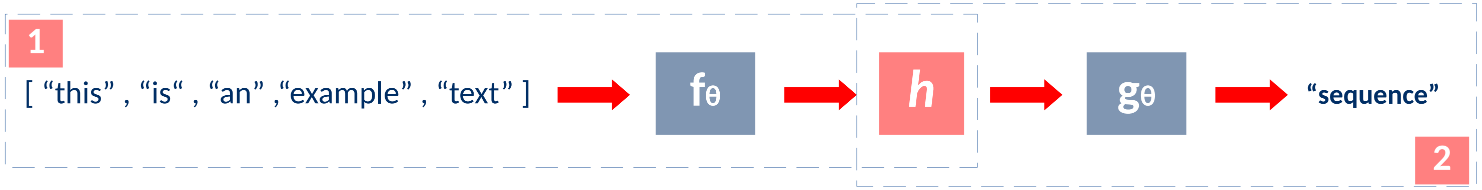
[Dooms, 2024]

# Summarizing

[ Finally :D ]

# Summarizing

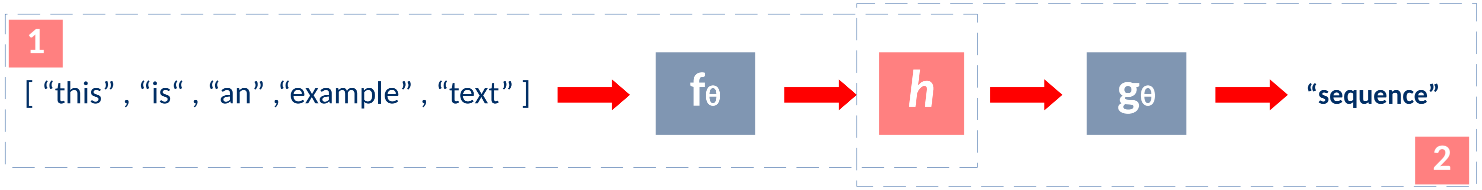
- **Two Step-Approach**
  1. **Model Context**
  2. **Predict Next Element given Context**



# Summarizing

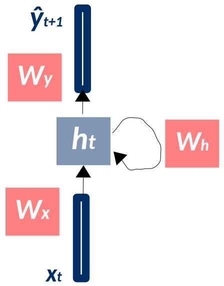
- **Two Step-Approach**

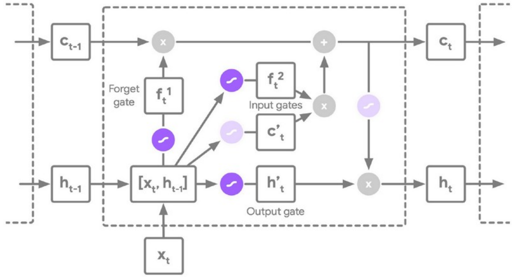
  1. Model Context
  2. Predict Next Element given Context

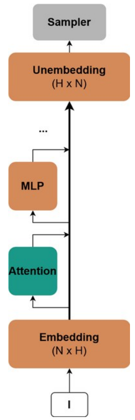



- **Serveral architectures with different capabilities**

  RNNs | LSTMs | GRUs | Transformers

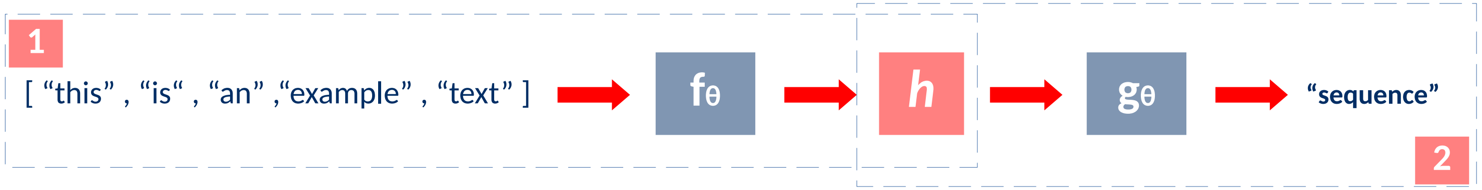


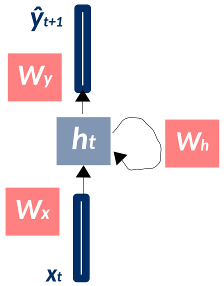
# Summarizing

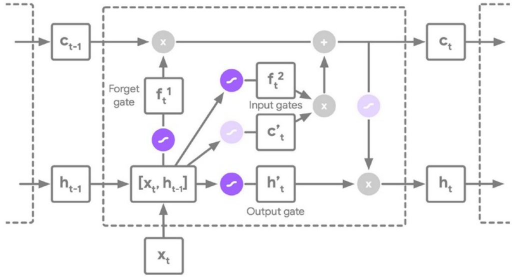
- **Two Step-Approach**
  1. Model Context
  2. Predict Next Element given Context

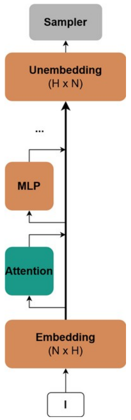



- **Serveral architectures with different capabilities**

  RNNs | LSTMs | GRUs | Transformers

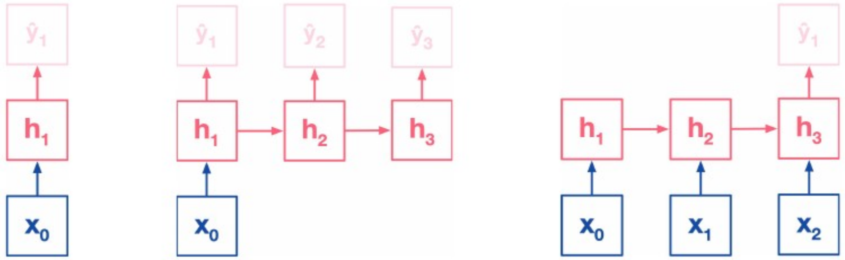



- **High Flexibility towards different problems**

  one-to-one | many-to-one | many-to-many …



# References


- K. Cho, B. van Merrienboer, Gulcehre, Caglar, D. Bahdanau; F. Bougares, H. Schwenk; Y. Bengio (2014). "**Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation"**. arXiv:1406.1078.
- Elman, J. L. (1990). **Finding Structure in Time. Cognitive Science,** 14(2), 179–211. https://doi.org/10.1207/s15516709cog1402_1
- S. Hochreiter and J. Schmidhuber. 1997. **Long Short-Term Memory**. Neural Comput. 9, 8 (November 15, 1997), 1735–1780. DOI:https://doi.org/10.1162/neco.1997.9.8.1735
- **The Recurrent Neural Network - Theory and Implementation of the Elman Network and LSTM**

  https://pabloinsente.github.io/the-recurrent-net
- **Understanding LSTM Networks** - COlah's Blog

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

# References


- K. Cho, B. van Merrienboer, Gulcehre, Caglar, D. Bahdanau; F. Bougares, H. Schwenk; Y. Bengio (2014). "**Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation"**. arXiv:1406.1078.
- Elman, J. L. (1990). **Finding Structure in Time. Cognitive Science,** 14(2), 179–211. https://doi.org/10.1207/s15516709cog1402_1
- S. Hochreiter and J. Schmidhuber. 1997. **Long Short-Term Memory**. Neural Comput. 9, 8 (November 15, 1997), 1735–1780. DOI:https://doi.org/10.1162/neco.1997.9.8.1735
- J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, D. Amodei. **Scaling Laws for Neural Language Models.** arxiv:2001.08361, 2020.
- A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, Aidan N. Gomez, L. Kaiser, I. Polosukhin. **Attention is All you Need**. NeurIPS 2017.

- **The Recurrent Neural Network - Theory and Implementation of the Elman Network and LSTM** https://pabloinsente.github.io/the-recurrent-net
- **Understanding LSTM Networks** - COlah's Blog http://colah.github.io/posts/2015-08-Understanding-LSTMs/

# Modeling Sequences with Neural Networks

José Oramas

University of Antwerp