MAI4CAREU

Master programmes in Artificial Intelligence 4 Careers in Europe

University of Cyprus

University of Cyprus

MAI645 - Machine Learning for Graphics and Computer Vision

Andreas Aristidou, PhD

Spring Semester 2025

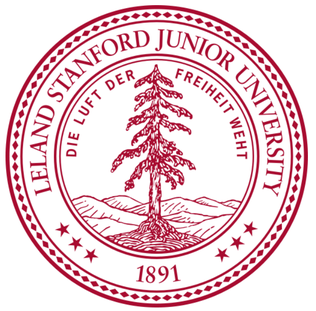
Co-financed by the European Union
Connecting Europe Facility

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Recurrent Neural Networks

These notes are based on the work of Fei-Fei Li, Jiajun Wu, Ruohan Gao,

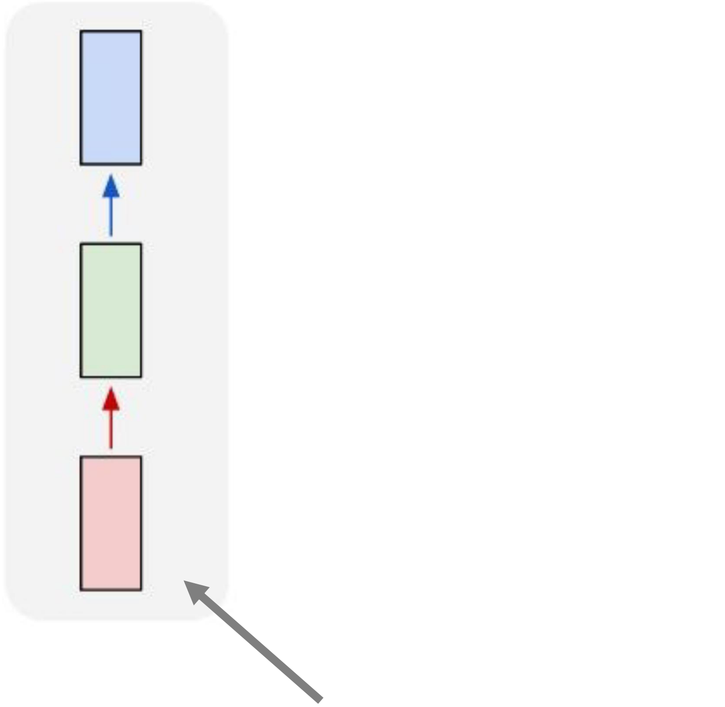
CS231 - Deep Learning for Computer Vision



This Master is run under the context of Action
No 2020-EU-IA-0087, co-financed by the EU CEF Telecom
under GA nr. INEA/CEF/ICT/A2020/2267423

Recurrent Neural Networks

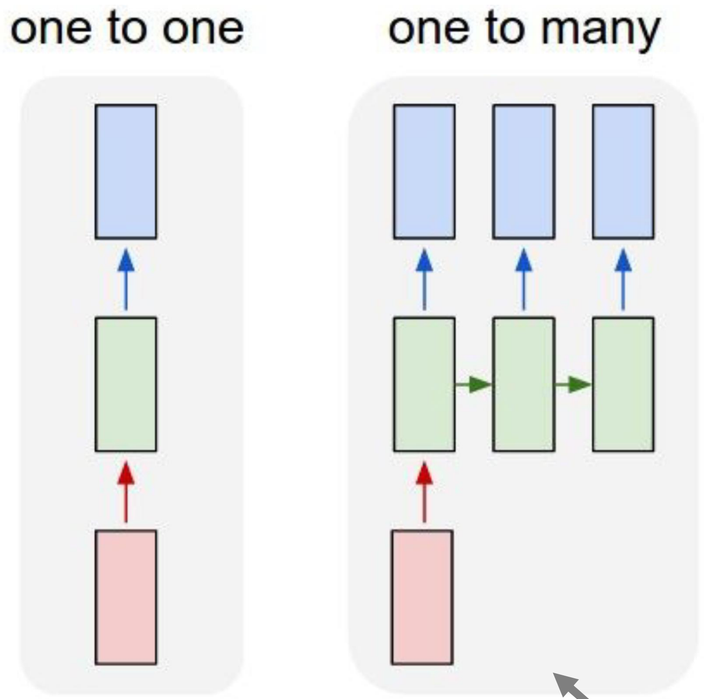
one to one



Vanilla Neural Networks

This Master is run under the context of Action
No 2020-EU-IA-0087, co-financed by the EU CEF Telecom
under GA nr. INEA/CEF/ICT/A2020/2267423

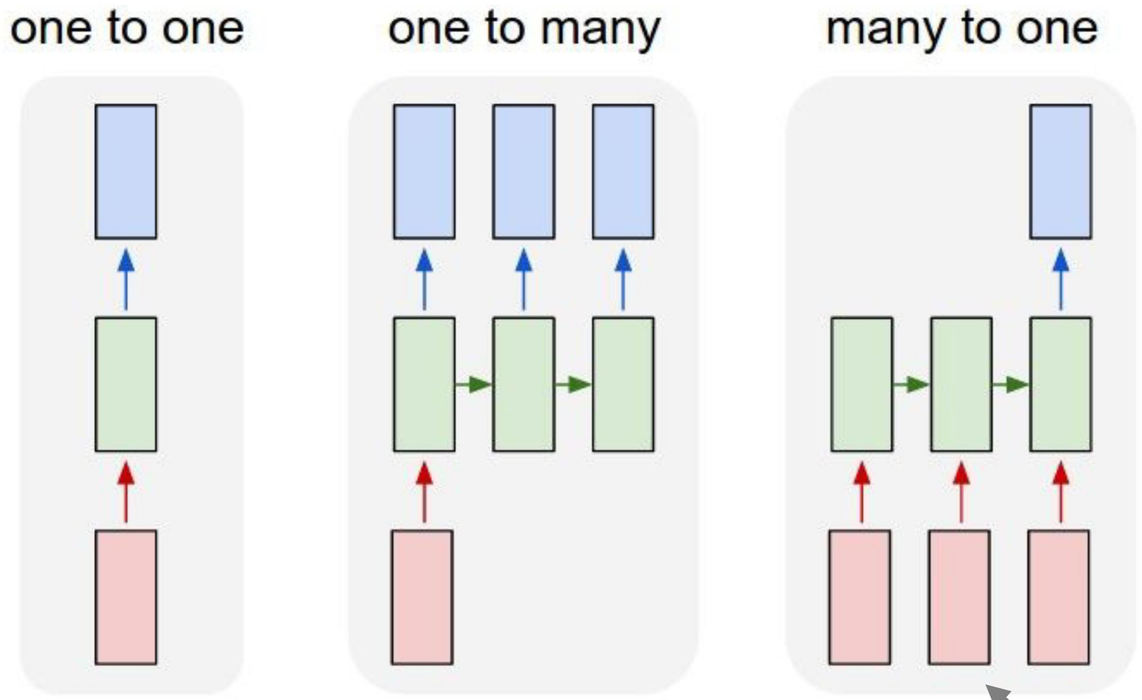
Recurrent Neural Networks



e.g., Image Captioning
image -> sequence of words

This Master is run under the context of Action
No 2020-EU-IA-0087, co-financed by the EU CEF Telecom
under GA nr. INEA/CEF/ICT/A2020/2267423

Recurrent Neural Networks

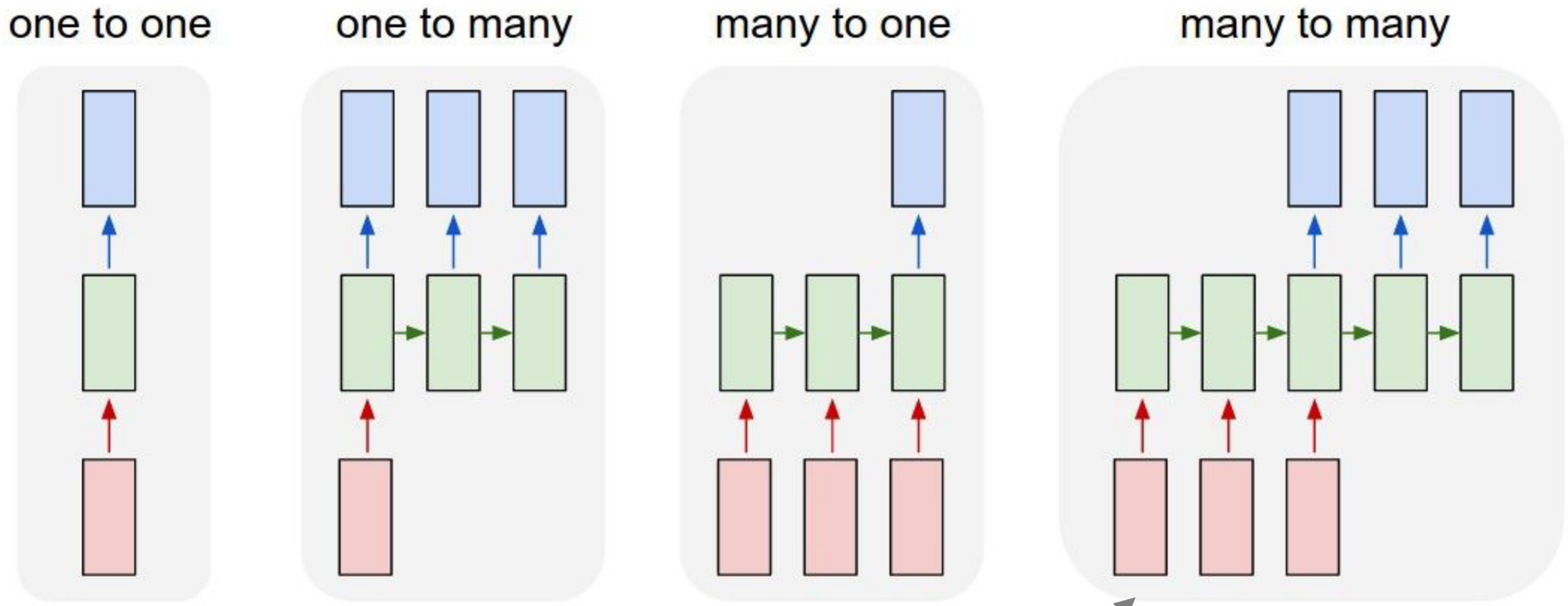


e.g. Action Prediction

sequence of video frames -> action class

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

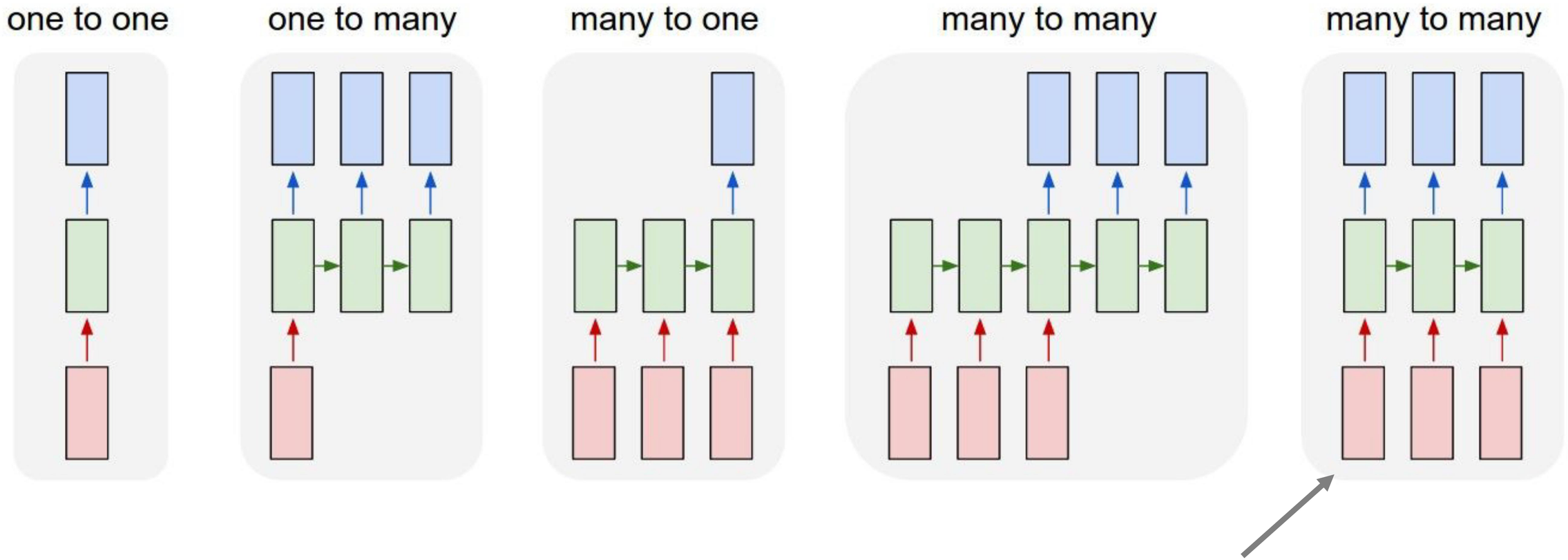
Recurrent Neural Networks



e.g. Video Captioning
Sequence of video frames -> caption

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

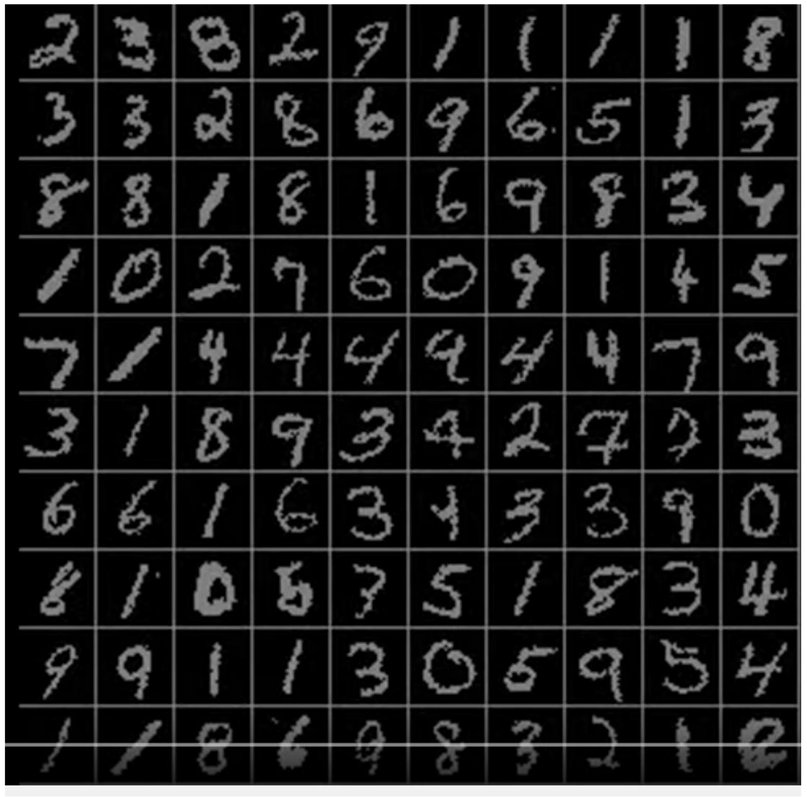
Recurrent Neural Networks

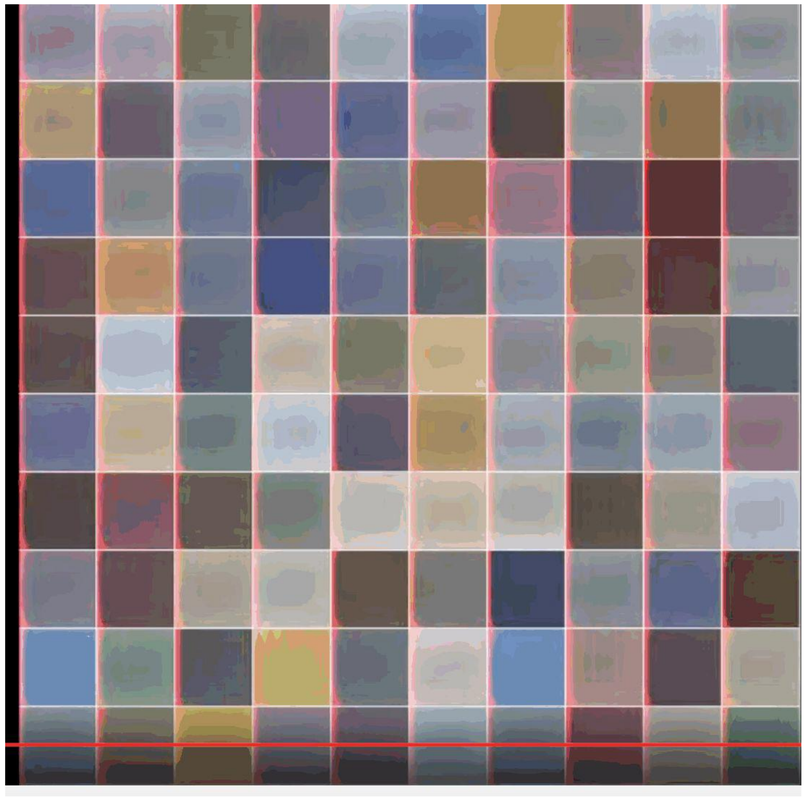


e.g. Video classification on frame level

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Sequential Processing of Non-Sequence Data





Sequential Processing of Non-Sequence Data

Generate images one piece at a time!

Ba, Mnih, and Kavukcuoglu, “Multiple Object Recognition with Visual Attention”, ICLR 2015.
Gregor et al, “DRAW: A Recurrent Neural Network For Image Generation”, ICML 2015

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

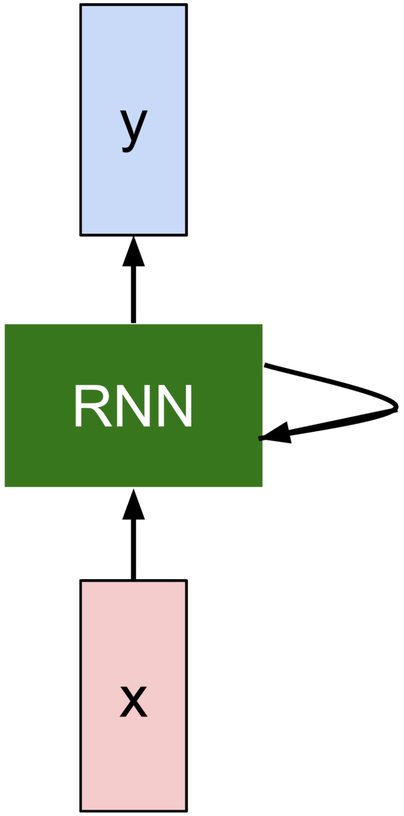
Recurrent Neural Network

A **Recurrent Neural Network (RNN)** is a type of neural network that is designed to work with sequential data. It is characterized by the ability to maintain an internal state, or memory, that captures information about the preceding sequence of inputs.

This allows the network to process input sequences of variable length and to generate output sequences of the same or different length. The internal state of the RNN is updated at each time step and is used to inform the processing of subsequent inputs in the sequence.

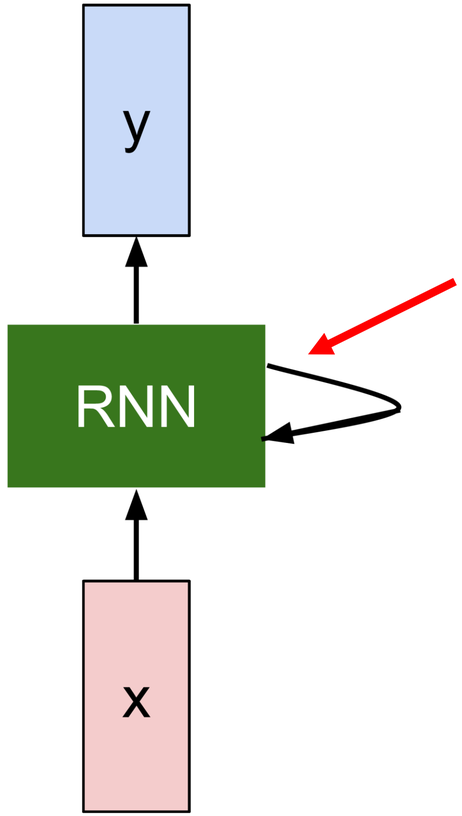
RNNs are widely used in natural language processing, speech recognition, time-series analysis, and other applications where sequential data is prevalent (e.g., animation). They have been shown to be effective in capturing dependencies and patterns in temporal data that would be difficult to model using other types of neural networks.

There are several variants of RNNs, including Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs), which are designed to address some of the limitations of standard RNNs in modeling long-term dependencies and avoiding the vanishing gradient problem.



This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

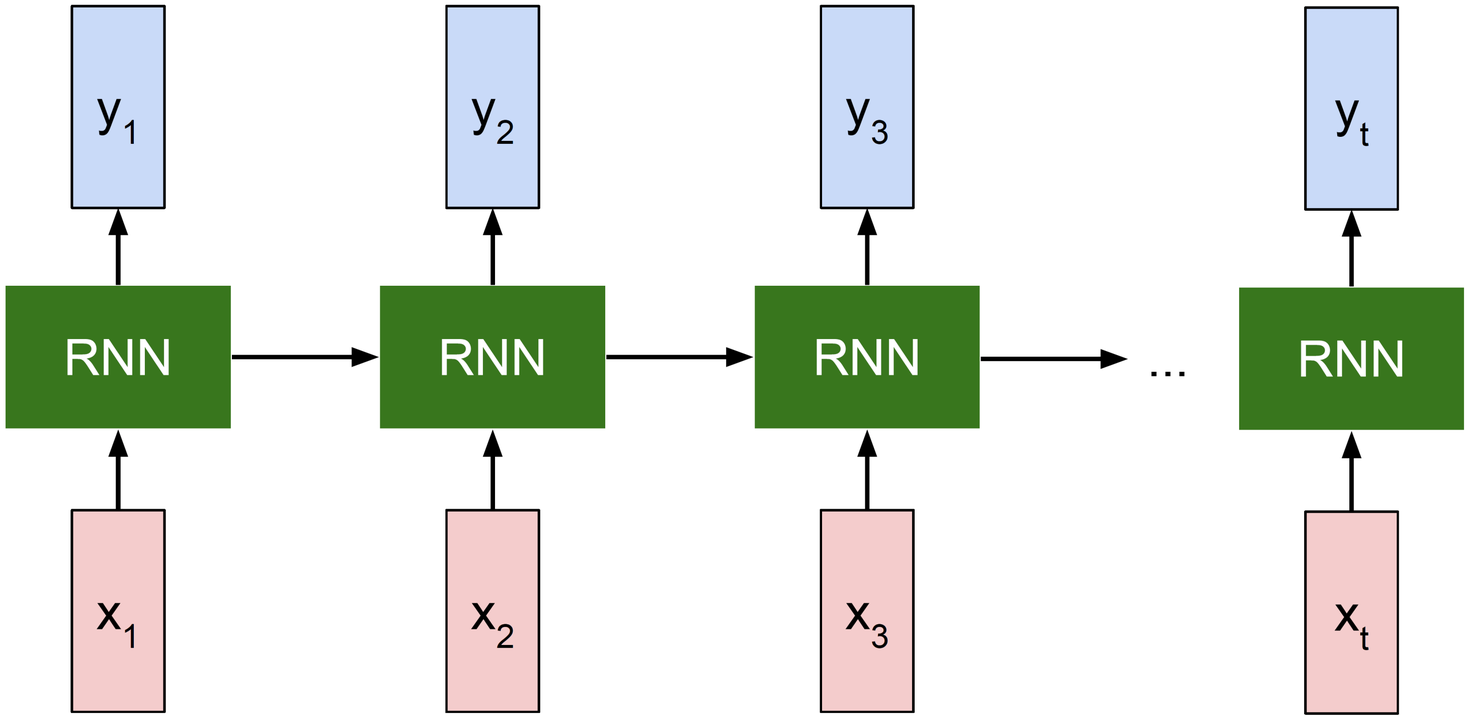
Recurrent Neural Network



Key idea: RNNs have an "internal state" that is updated as a sequence is processed

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Unrolled Recurrent Neural Network



An unrolled Recurrent Neural Network (RNN) is a visualization of an RNN that shows the network's architecture and operations across time steps. It is called "unrolled" because the connections between the network's hidden units are shown as a sequence of stacked layers, each representing one time step, instead of being shown as a loop

Each layer corresponds to a time step, and the input at each time step is fed into the corresponding layer. The output of each layer is then used as the input to the next layer, and so on. This allows the network to process input sequences of variable length and to generate output sequences of the same or different length.

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

RNN hidden state update

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

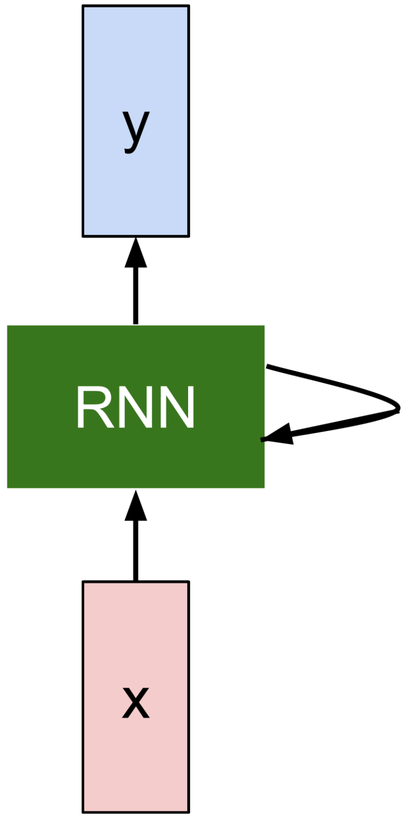
$$h_t = f_W(h_{t-1}, x_t)$$

new state

some function with parameters W

old state

input vector at some time step



This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

RNN hidden state update

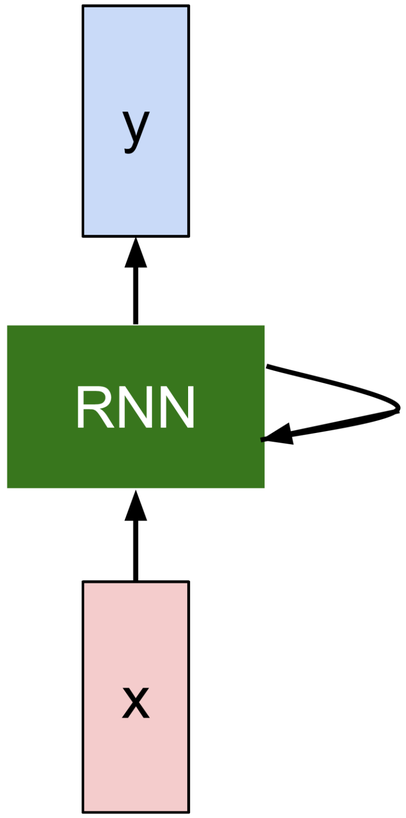
We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

$$y_t = f_{W_{hy}}(h_t)$$

output

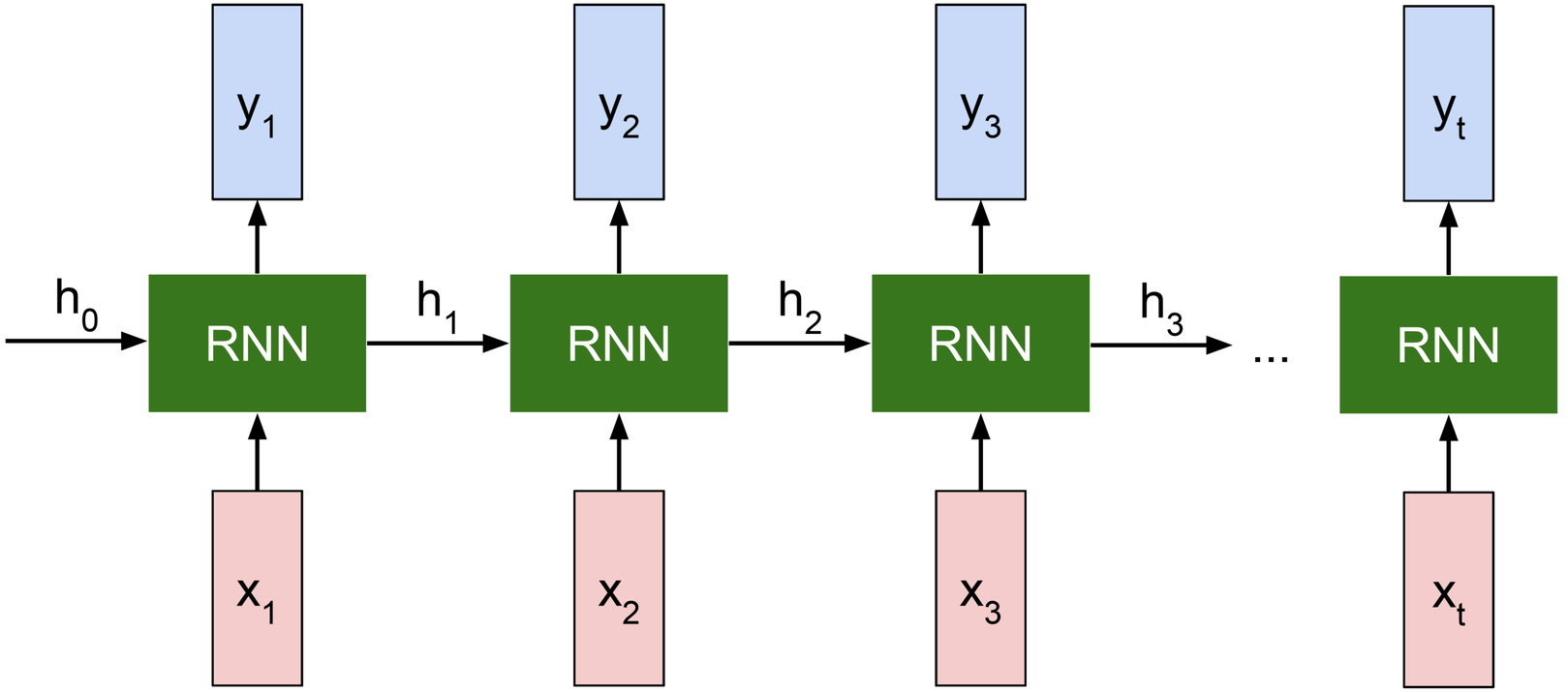
new state

another function with parameters W_o



This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Recurrent Neural Network



We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

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

Note: the same function and the same set of parameters are used at every time step.

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

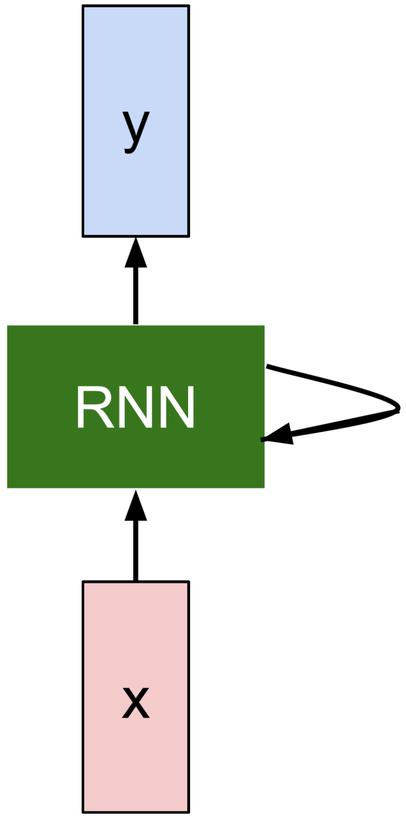
(Vanilla) Recurrent Neural Network

The state consists of a single *"hidden"* vector **h**:

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

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

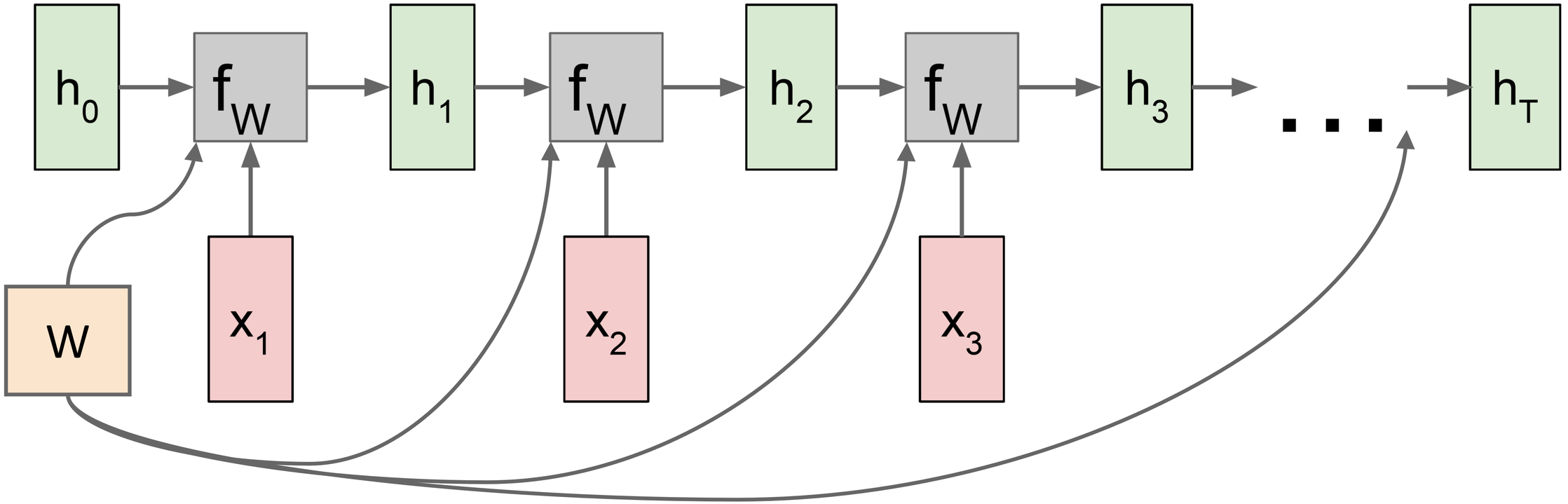


Sometimes called a "Vanilla RNN" or an "Elman RNN" after Prof. Jeffrey Elman

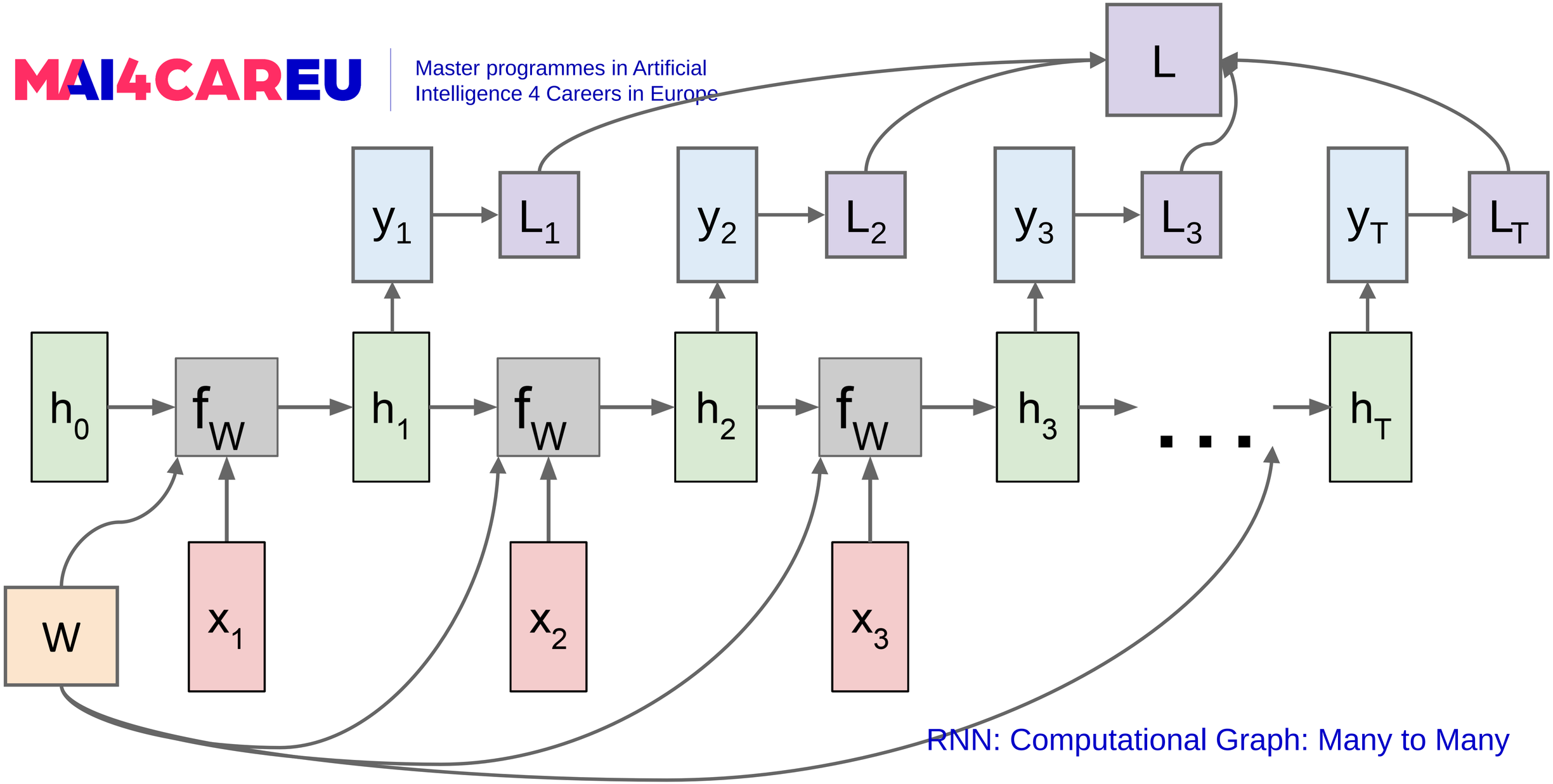
This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

RNN: *Computational Graph*

Re-use the same weight matrix at every time-step

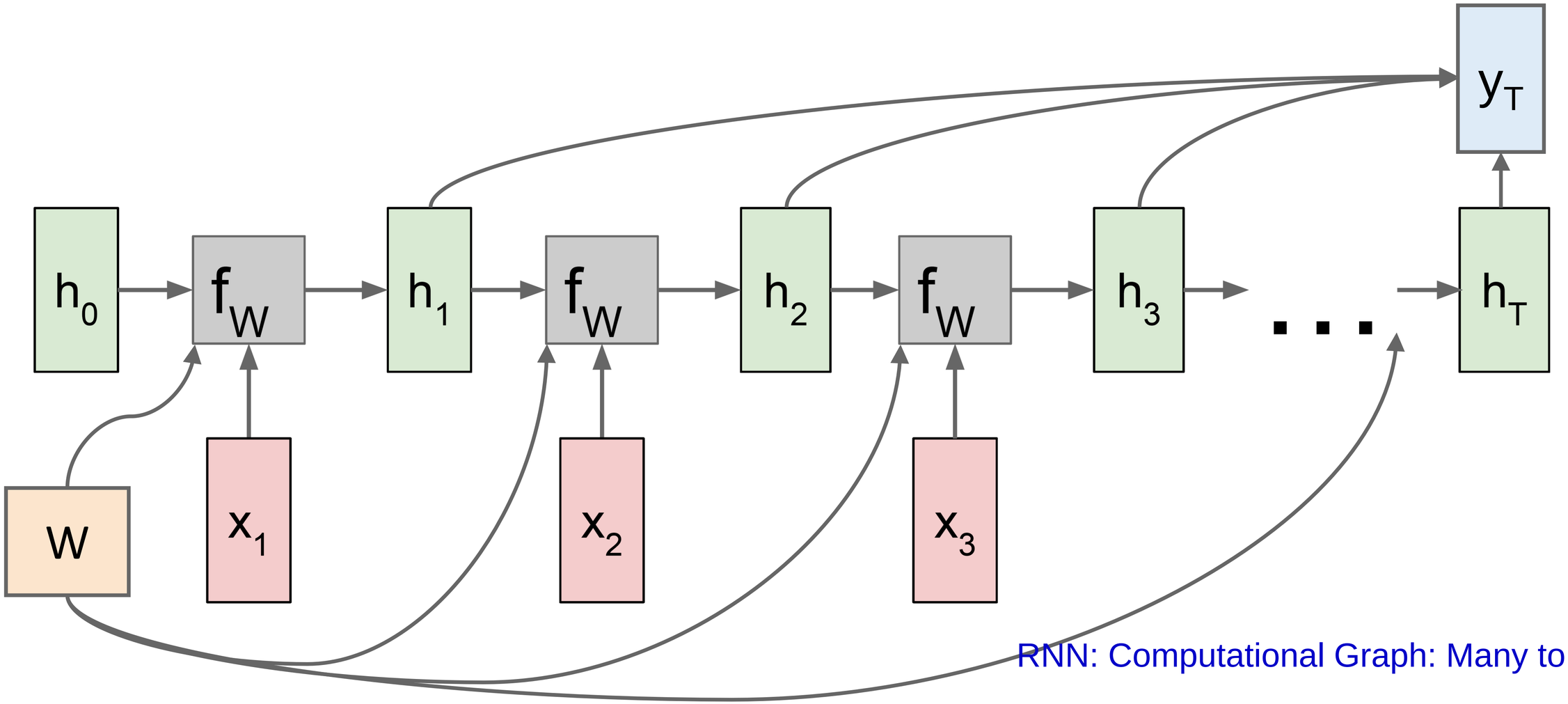


This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423



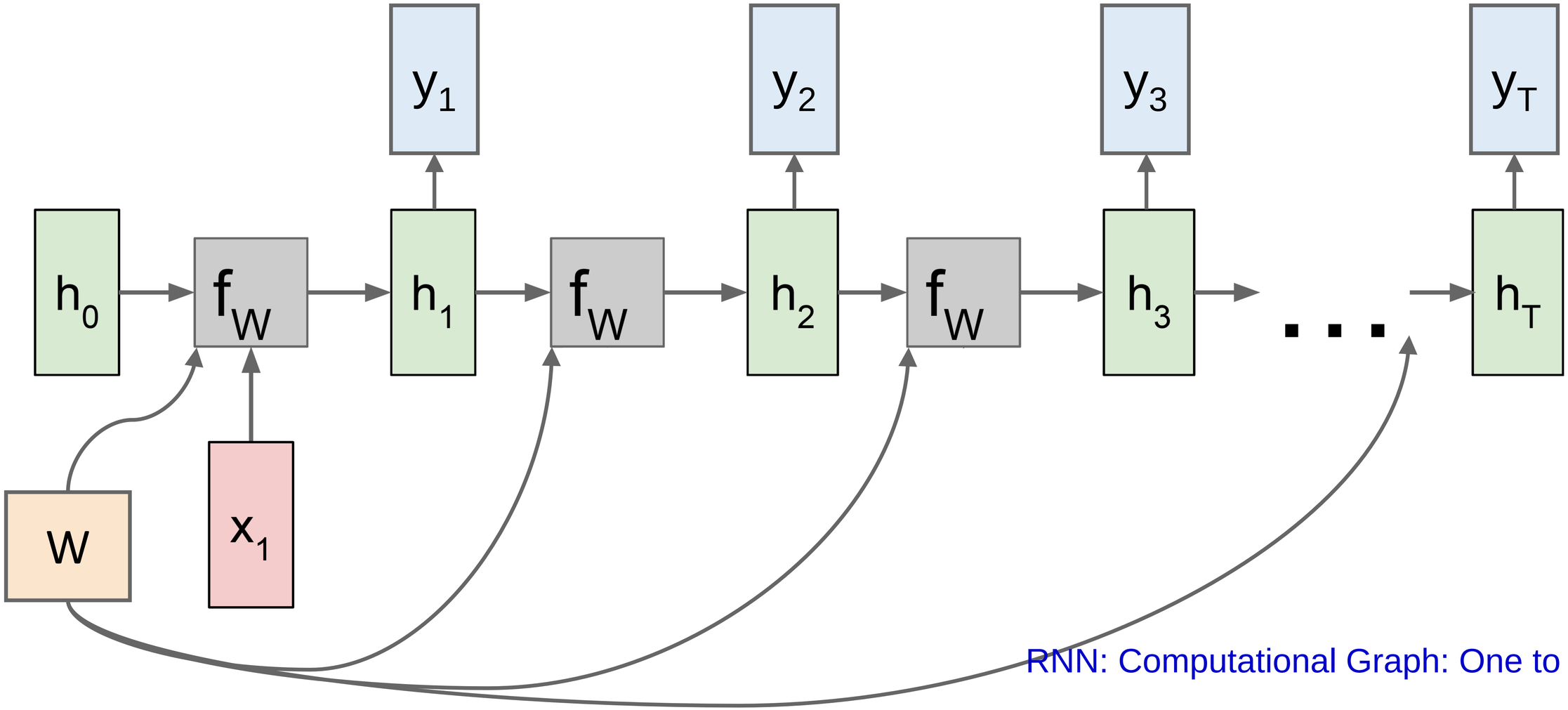
RNN: Computational Graph: Many to Many

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423



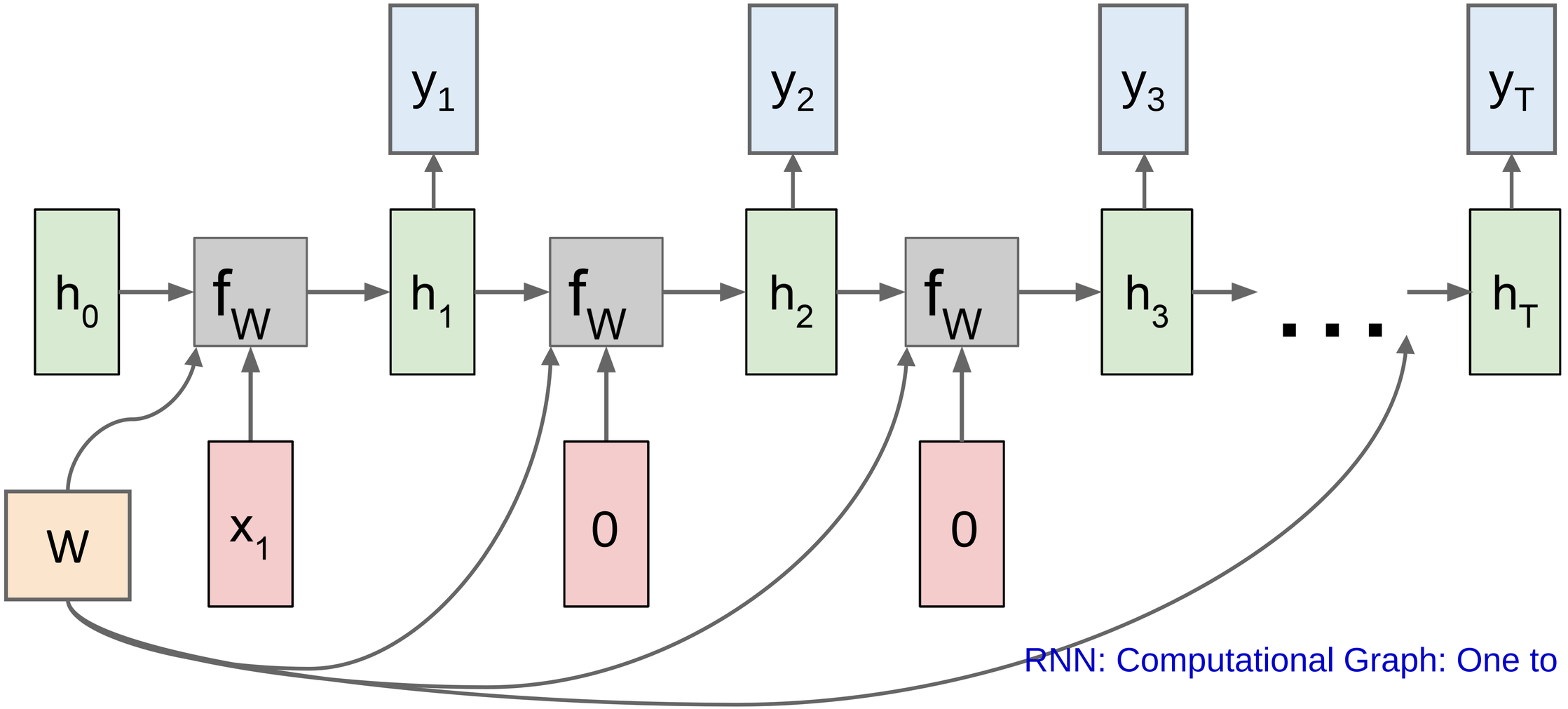
RNN: Computational Graph: Many to One

This Master is run under the context of Action
No 2020-EU-IA-0087, co-financed by the EU CEF Telecom
under GA nr. INEA/CEF/ICT/A2020/2267423



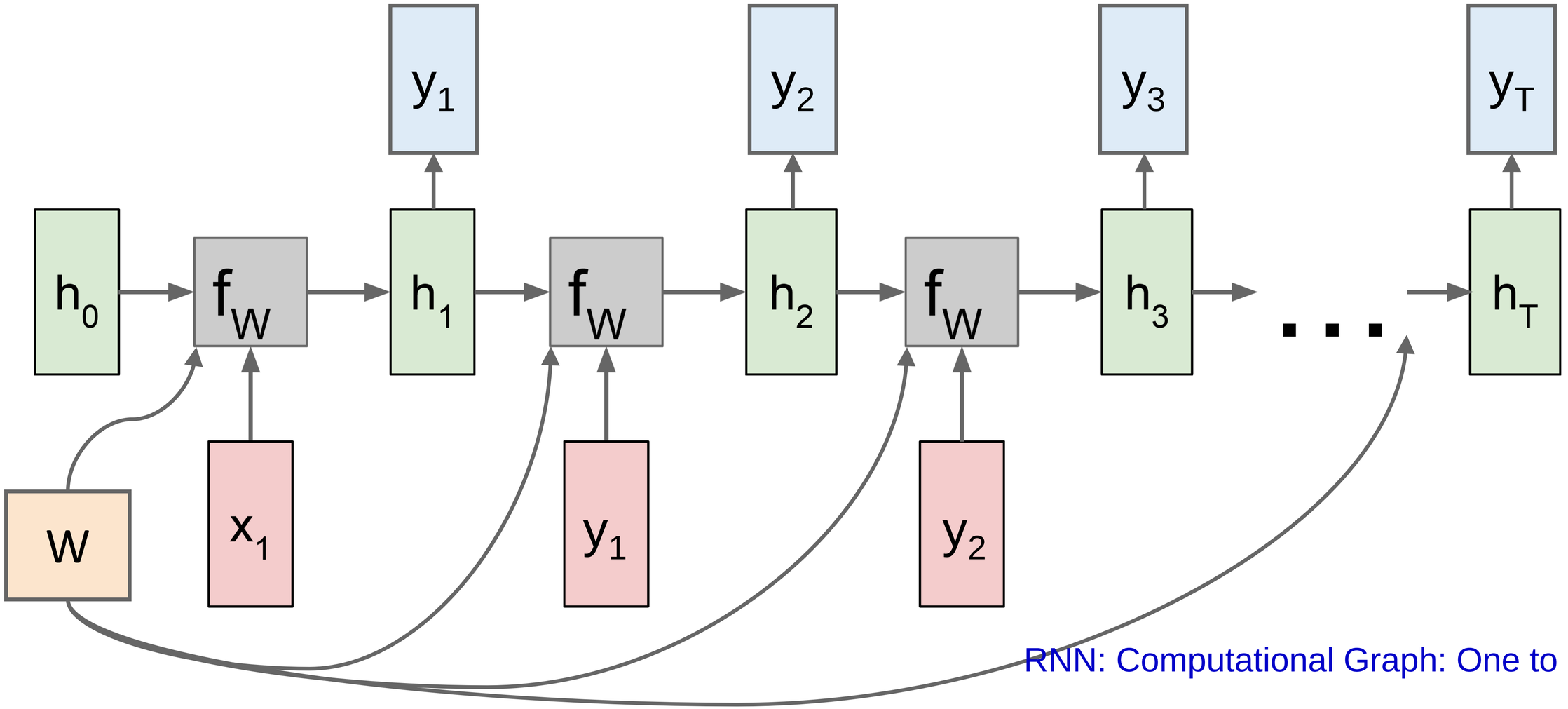
RNN: Computational Graph: One to Many

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423



RNN: Computational Graph: One to Many

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423



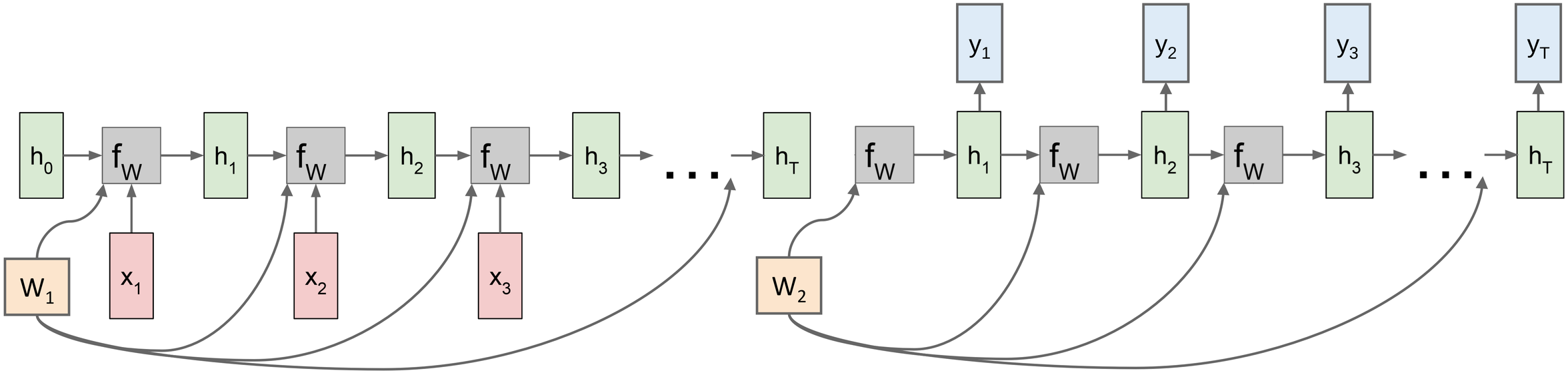
RNN: Computational Graph: One to Many

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Sequence to Sequence: *Many-to-one + one-to-many*

Many to one: Encode input sequence in a single vector

One to many: Produce output sequence from single input vector

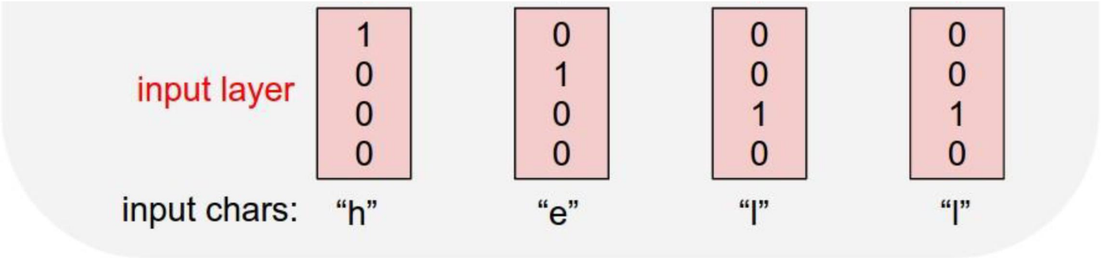


This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Example: *Character-level Language Model*

Vocabulary: [h,e,l,o]

Example training sequence: **“hello”**



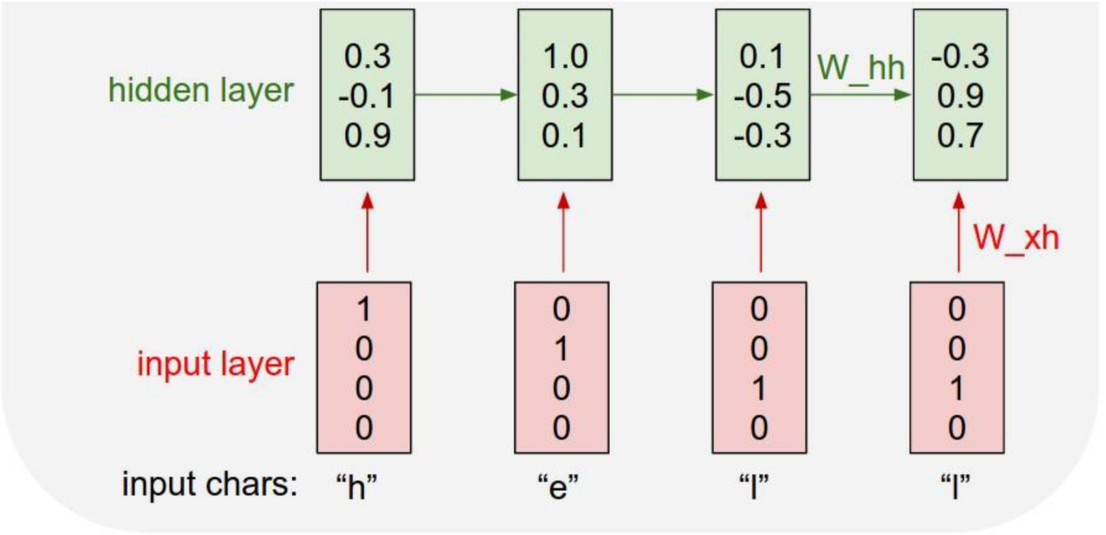
This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Example: *Character-level Language Model*

Vocabulary: [h,e,l,o]

Example training sequence: **“hello”**

$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$

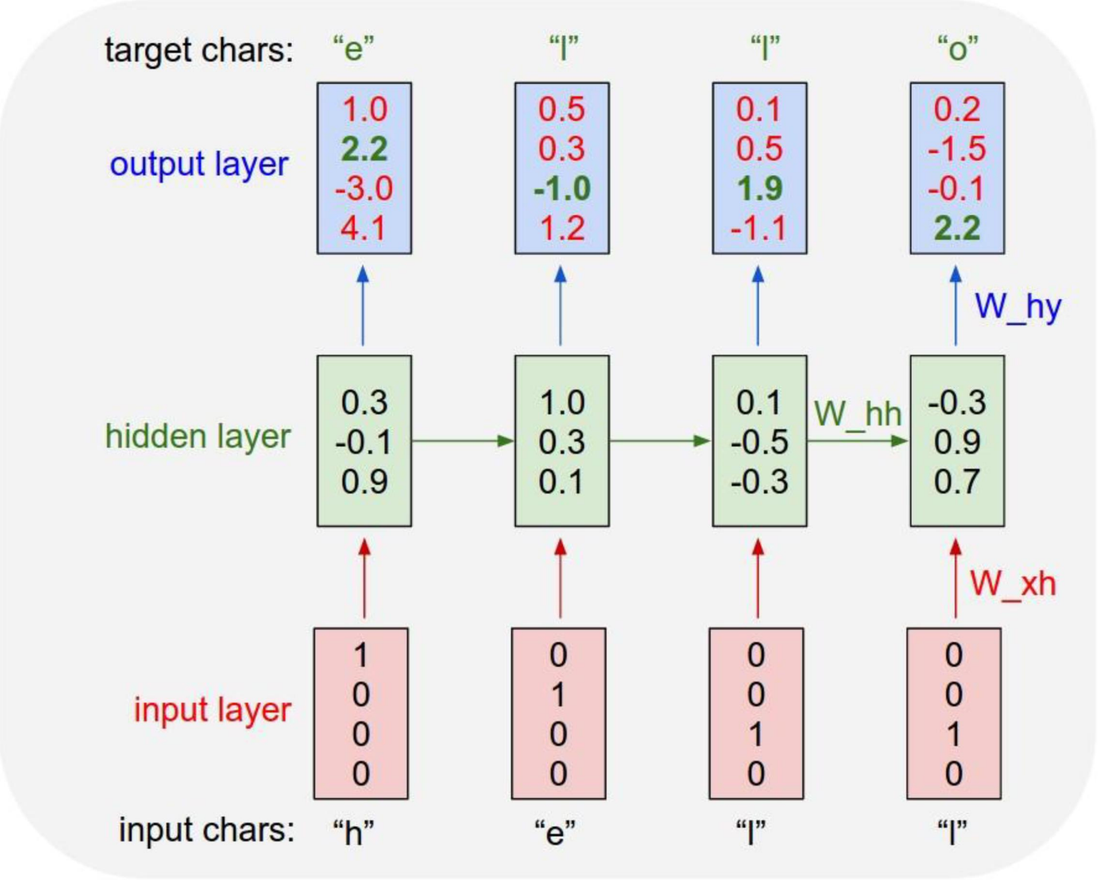


This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Example: *Character-level Language Model*

Vocabulary: [h,e,l,o]

Example training sequence: **“hello”**



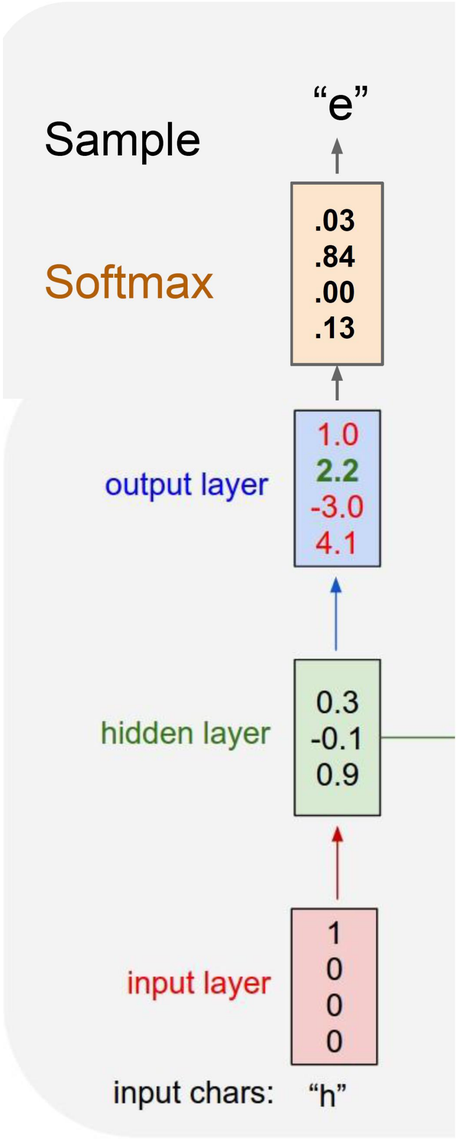
This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Example: *Character-level Language Model*

Vocabulary: [h,e,l,o]

Example training sequence: **“hello”**

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



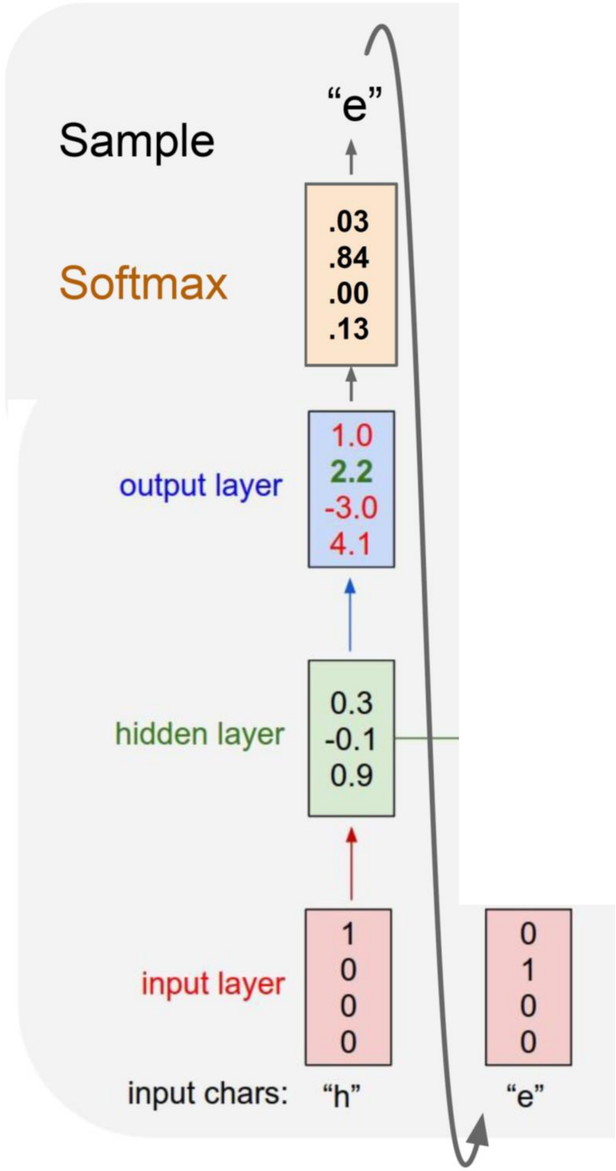
This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Example: *Character-level Language Model*

Vocabulary: [h,e,l,o]

Example training sequence: **“hello”**

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



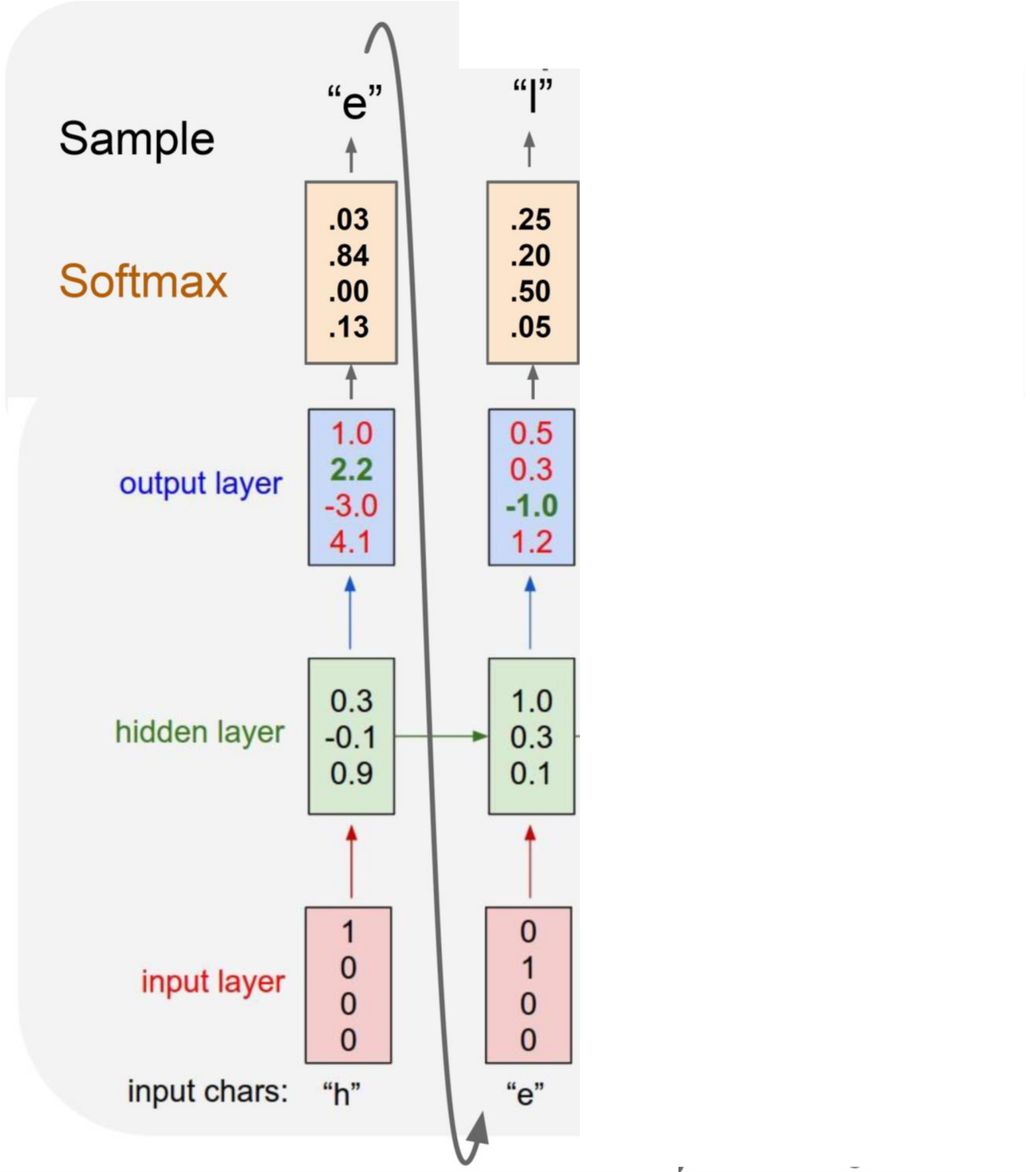
This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Example: *Character-level Language Model*

Vocabulary: [h,e,l,o]

Example training sequence: **“hello”**

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



Co-financed by the European Union
Connecting Europe Facility

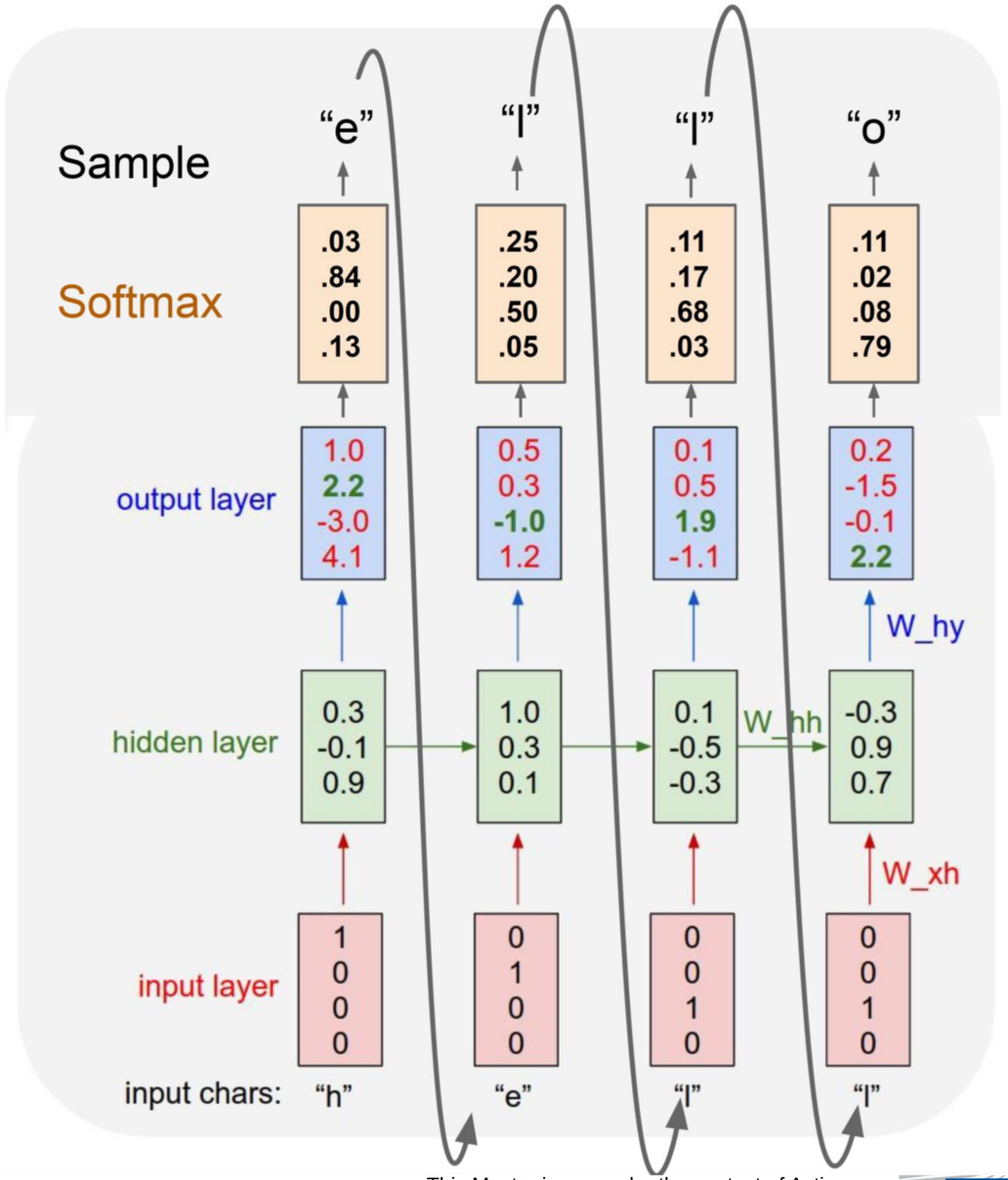
This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Example: *Character-level Language Model*

Vocabulary: [h,e,l,o]

Example training sequence: **“hello”**

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



This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

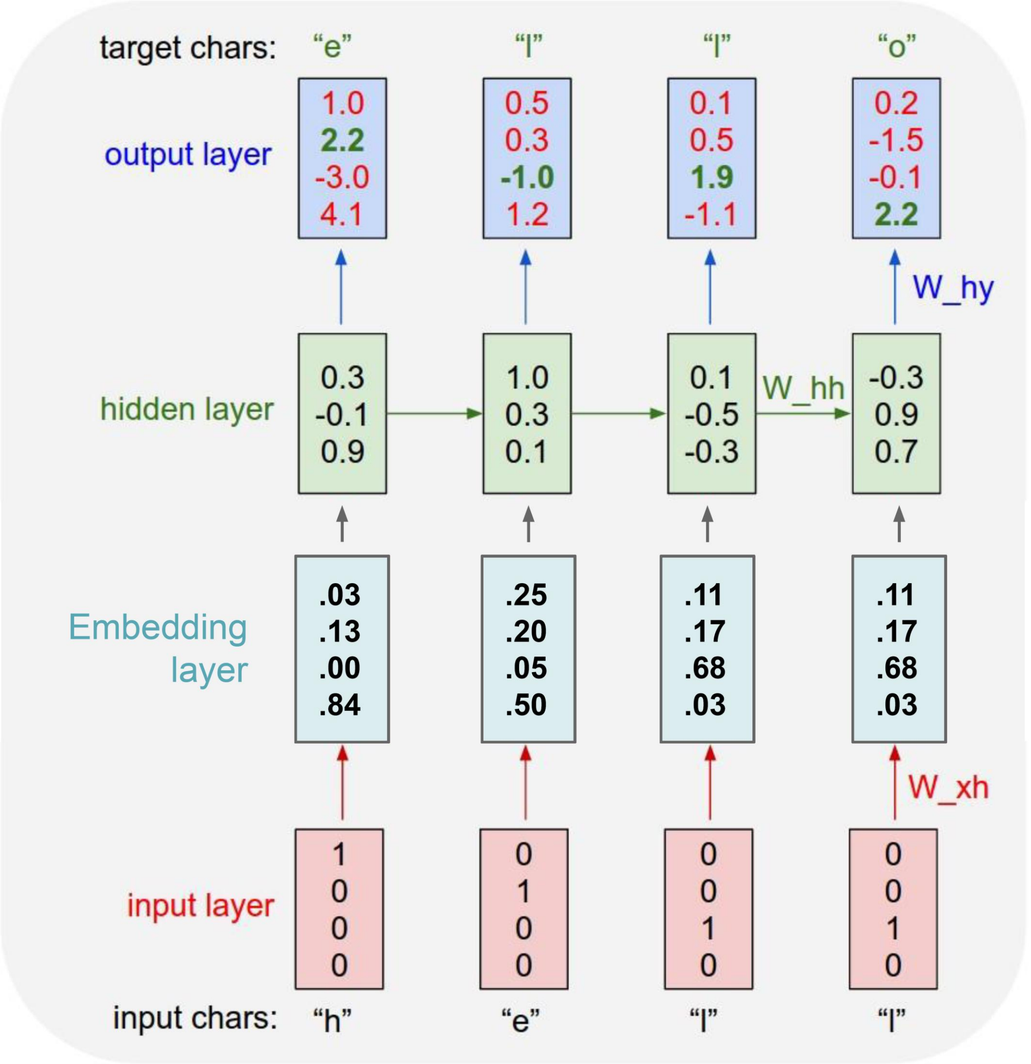
Example: *Character-level Language Model*

Vocabulary: [h,e,l,o]

Example training sequence: **“hello”**

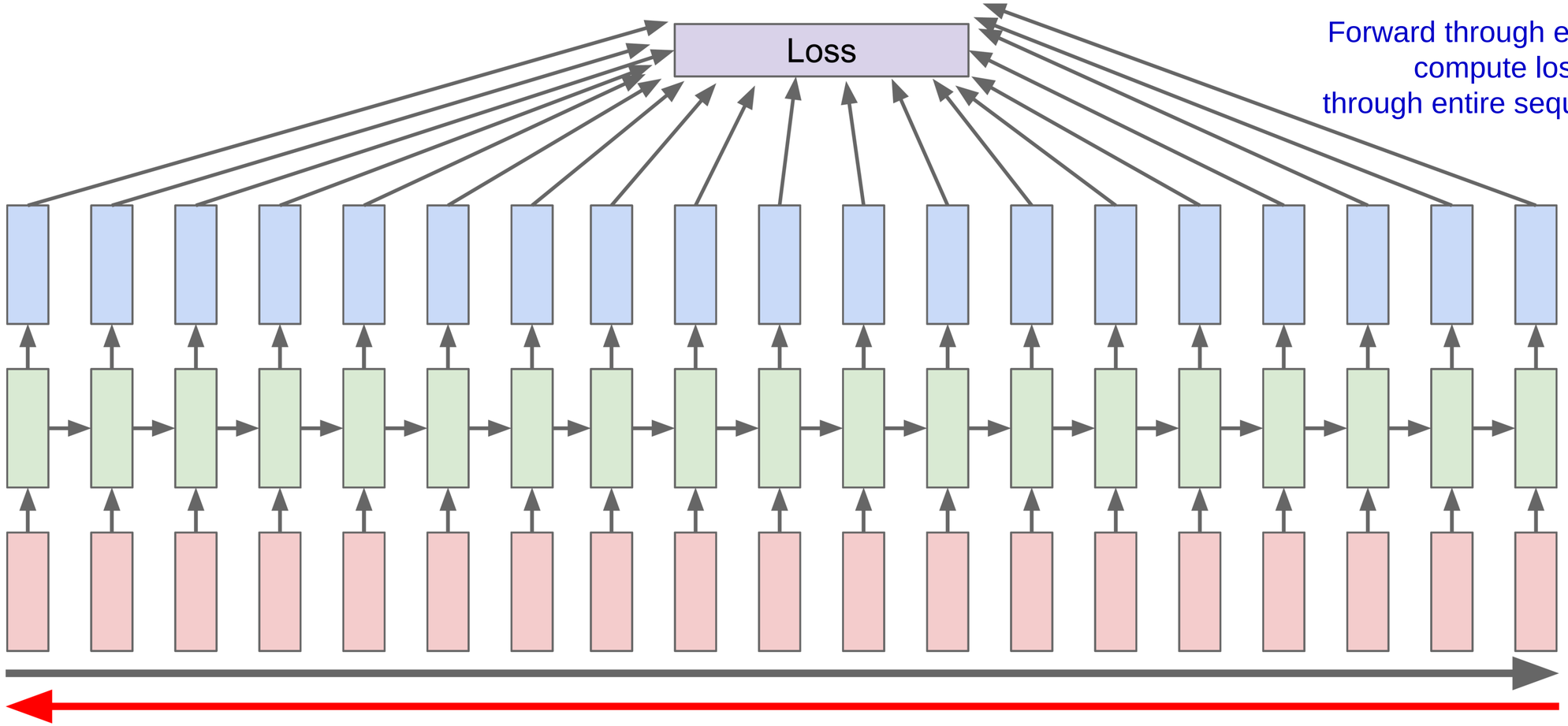
$$\begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{14} \\ w_{31} & w_{32} & w_{33} & w_{14} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} w_{11} \\ w_{21} \\ w_{31} \end{bmatrix}$$

Matrix multiply with a one-hot vector just extracts a column from the weight matrix. We often put a separate **embedding** layer between input and hidden layers.



This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

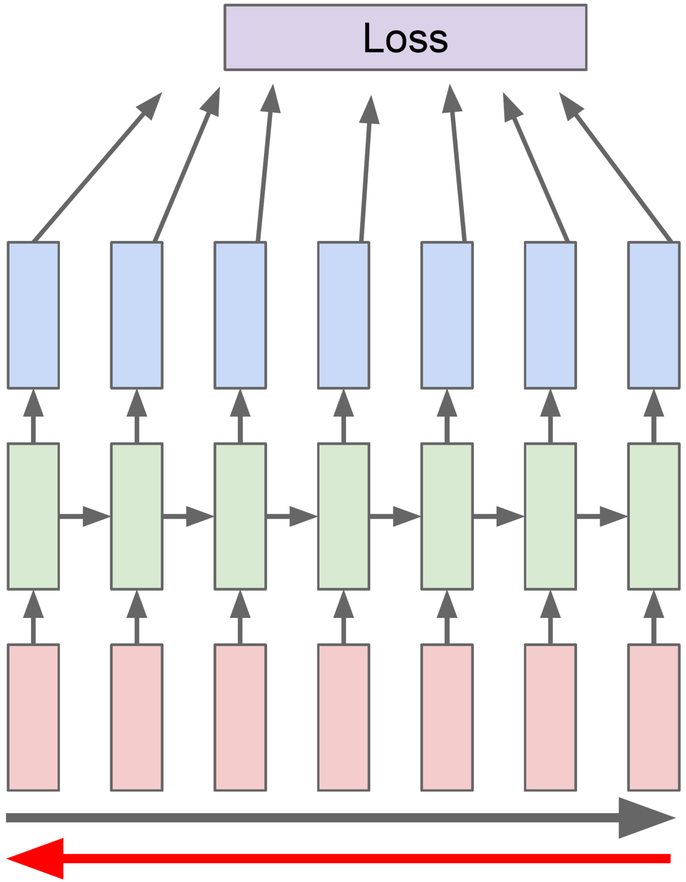
Backpropagation through time



Forward through entire sequence to compute loss, then backward through entire sequence to compute gradient

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

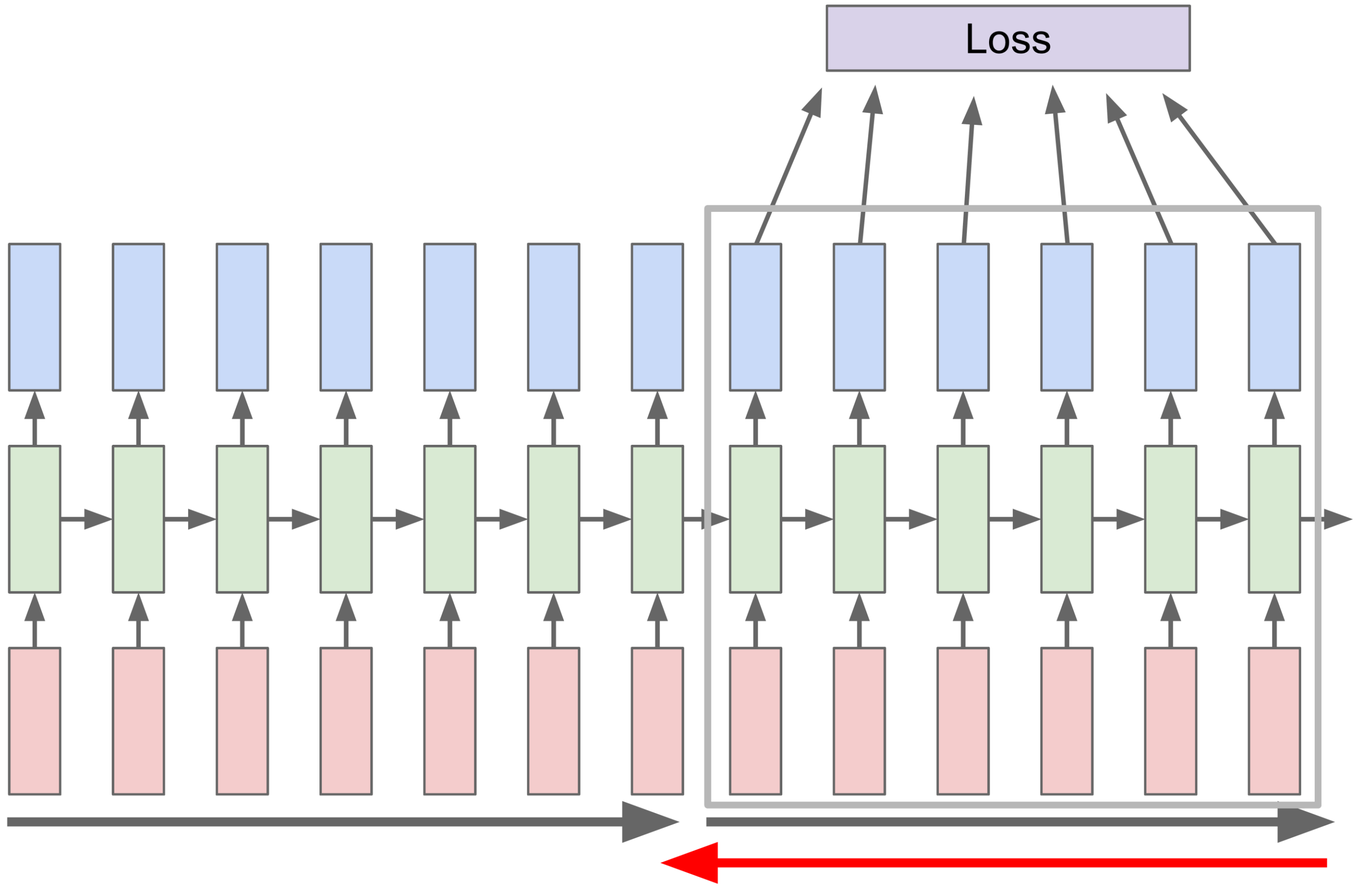
Truncated Backpropagation through time



Run forward and backward through chunks of the sequence instead of whole sequence

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

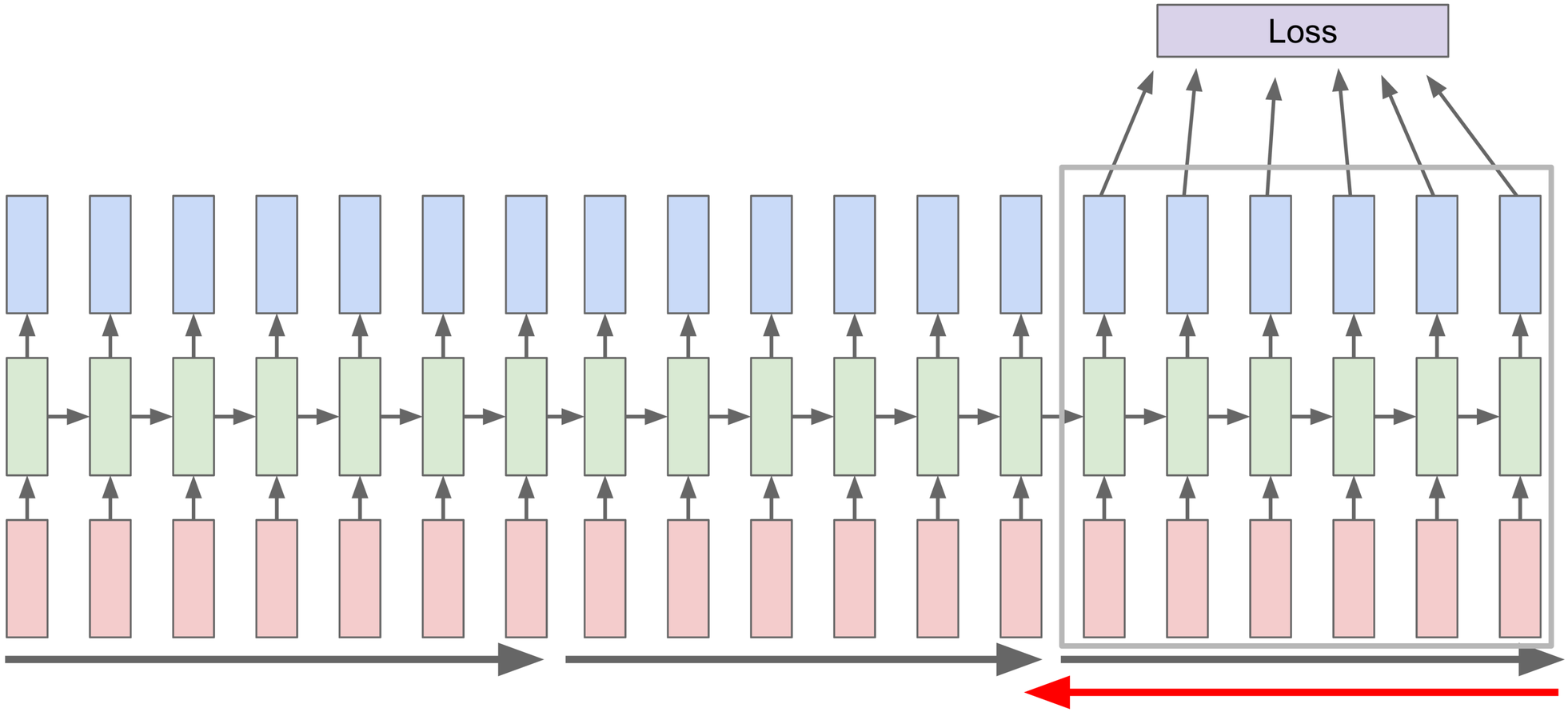
Truncated Backpropagation through time



Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Truncated Backpropagation through time



This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Searching for interpretable cells

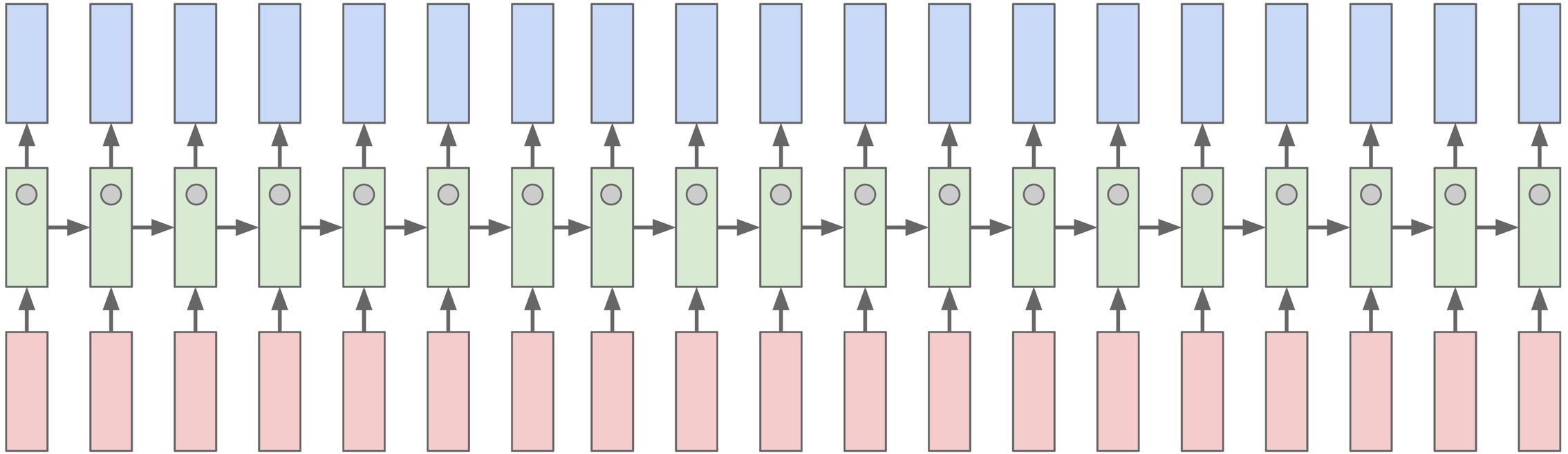
In the context of Recurrent Neural Networks (RNNs), **searching for interpretable cells** refers to the task of identifying which cells in the network are most relevant for generating its output. An RNN is a type of neural network that is designed to process sequential data, such as natural language text or time series data. It is composed of multiple interconnected nodes, or cells, that allow the network to store and retrieve information about the data it is processing.

One challenge with RNNs is that they can be difficult to interpret, meaning it can be hard to understand how the network is making its predictions or decisions. To address this issue, researchers have developed methods for identifying which cells in the network are most important for generating its output. These methods typically involve analyzing the activations of the cells during the network's processing of input data, and then selecting the cells that exhibit the strongest relationship with the network's output.

By identifying the most interpretable cells in an RNN, researchers can gain a better understanding of how the network is processing input data, and potentially use this information to improve the network's performance or make it more explainable to end-users.

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Searching for interpretable cells



This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Searching for interpretable cells

```
"You mean to imply that I have nothing to eat out of.... On the
contrary, I can supply you with everything even if you want to give
dinner parties," warmly replied Chichagov, who tried by every word he
spoke to prove his own rectitude and therefore imagined Kutuzov to be
animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating
smile: "I meant merely to say what I said."
```

quote detection cell

```
static int __dequeue_signal(struct sigpending *pending, sigset_t *mask,
        siginfo_t *info)
{
  int sig = next_signal(pending, mask);
  if (sig) {
    if (current->notifier) {
      if (sigismember(current->notifier_mask, sig)) {
        if (!(current->notifier)(current->notifier_data)) {
          clear_thread_flag(TIF_SIGPENDING);
          return 0;
        }
      }
    }
    collect_signal(sig, pending, info);
  }
  return sig;
}
```

if statement cell

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

This Master is run under the context of Action
No 2020-EU-IA-0087, co-financed by the EU CEF Telecom
under GA nr. INEA/CEF/ICT/A2020/2267423

RNN tradeoffs

RNN Advantages:

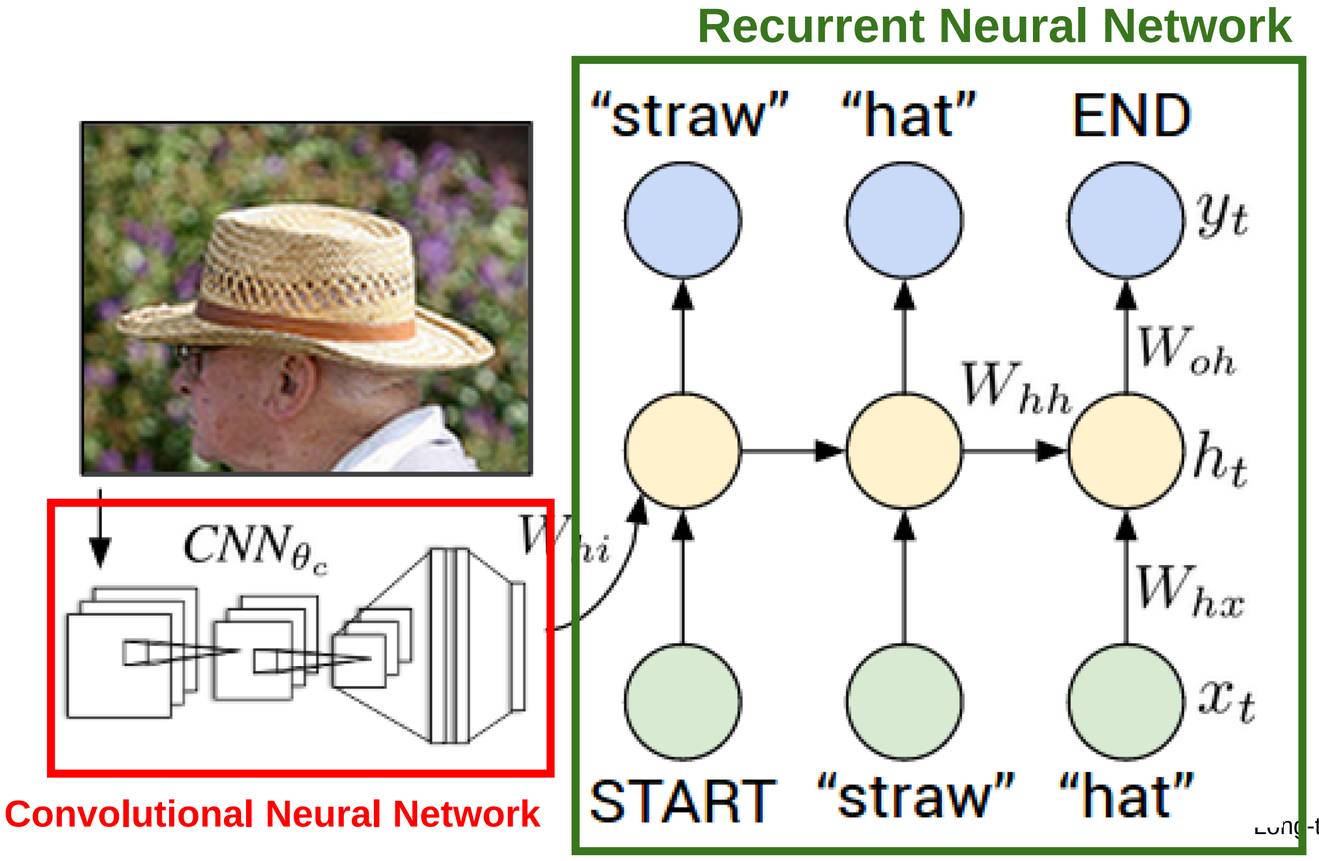
- Can process any length input
- Computation for step *t* can (in theory) use information from many steps back
- Model size doesn't increase for longer input
- Same weights applied on every timestep, so there is symmetry in how inputs are processed.

RNN Disadvantages:

- Recurrent computation is slow
- In practice, difficult to access information from many steps back

This Master is run under the context of Action
No 2020-EU-IA-0087, co-financed by the EU CEF Telecom
under GA nr. INEA/CEF/ICT/A2020/2267423

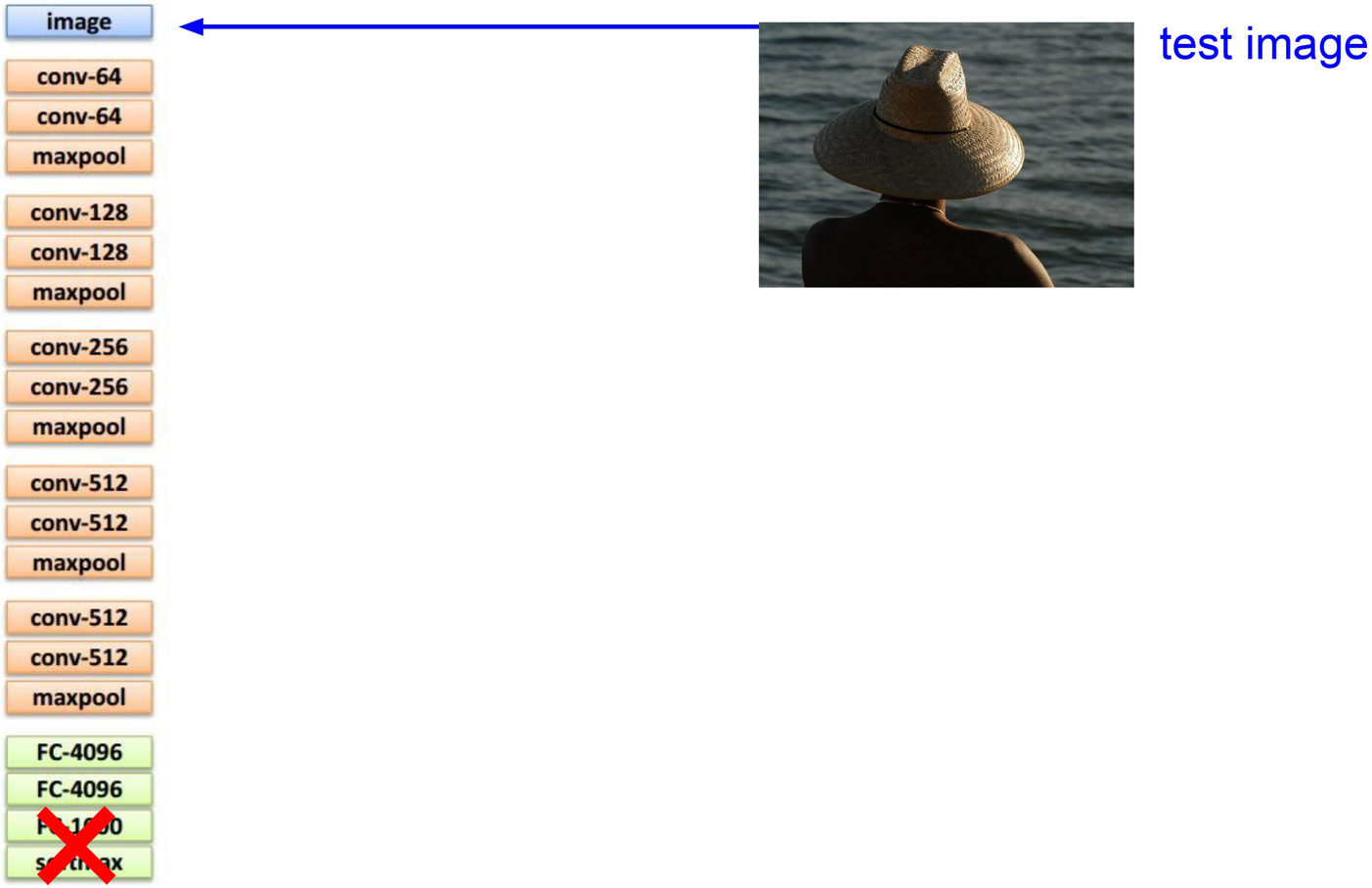
Image Captioning



Explain Images with Multimodal Recurrent Neural Networks, Mao et al.
Deep Visual-Semantic Alignments for Generating Image Descriptions, Karpathy and Fei-Fei
Show and Tell: A Neural Image Caption Generator, Vinyals et al.
Long-term Recurrent Convolutional Networks for Visual Recognition and Description, Donahue et al.
Learning a Recurrent Visual Representation for Image Caption Generation, Chen and Zitnick

This Master is run under the context of Action
No 2020-EU-IA-0087, co-financed by the EU CEF Telecom
under GA nr. INEA/CEF/ICT/A2020/2267423

Image Captioning

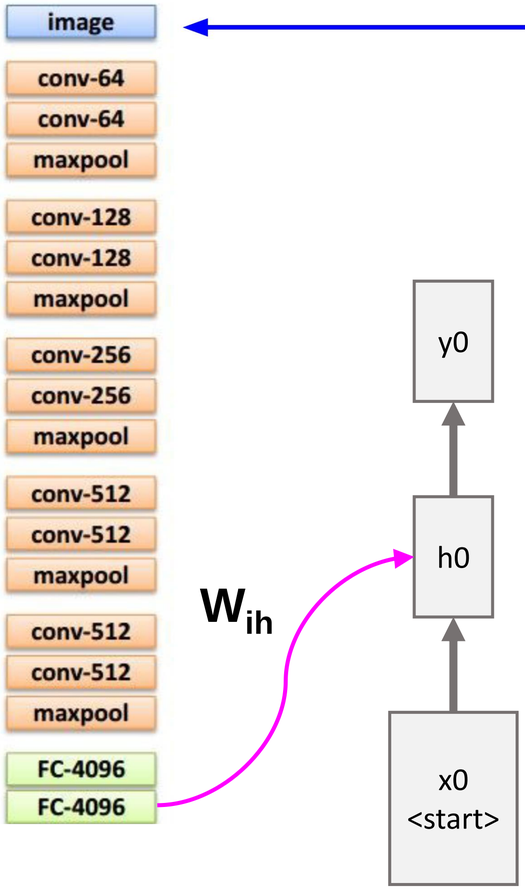


test image

Co-financed by the European Union
Connecting Europe Facility

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Image Captioning



test image

before:

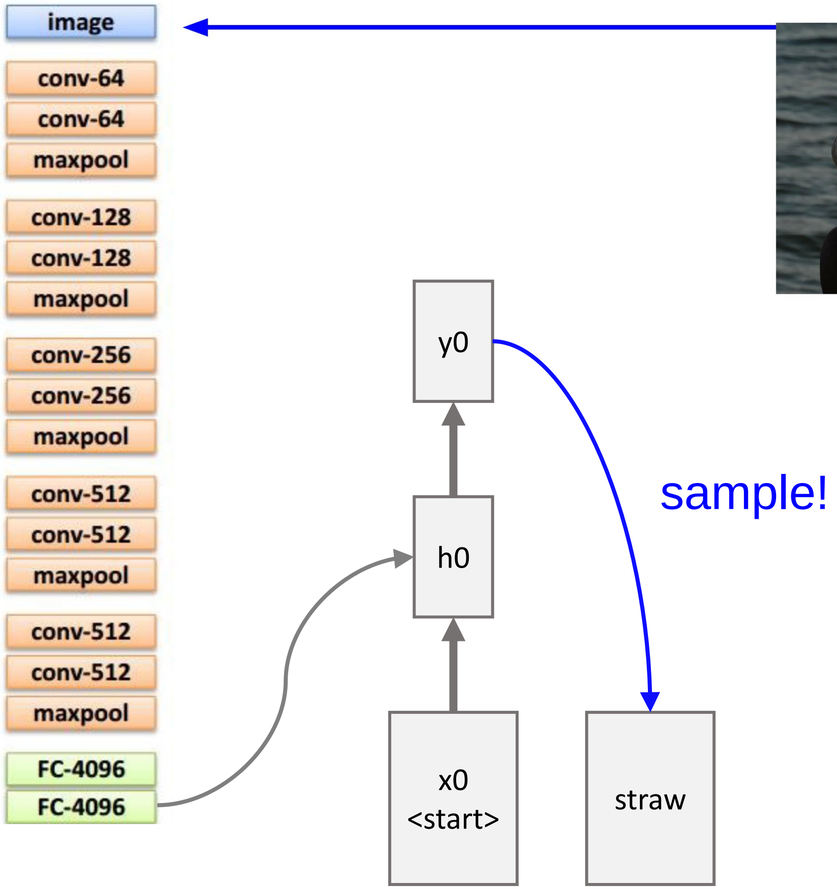
$h = \tanh(W_{xh} * x + W_{hh} * h)$

now:

$h = \tanh(W_{xh} * x + W_{hh} * h + \mathbf{W_{ih} * v})$

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

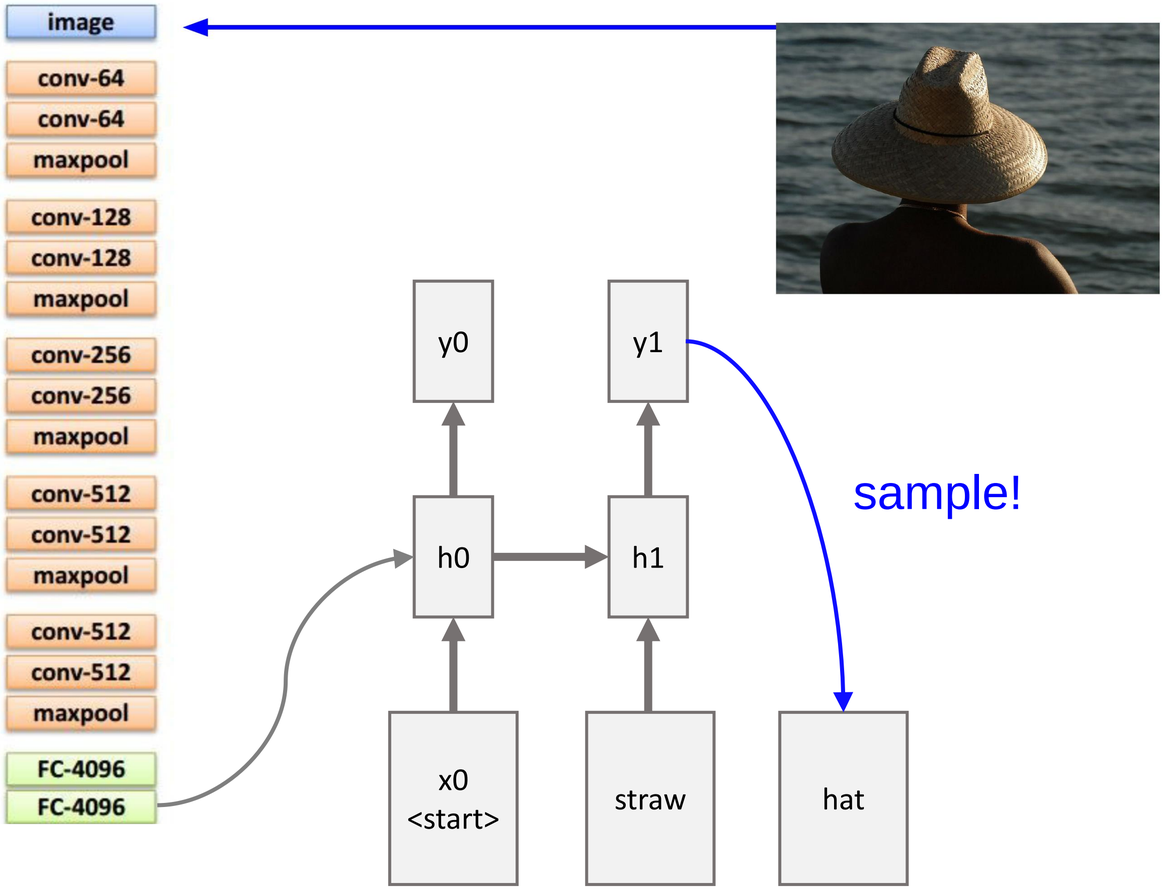
Image Captioning



test image

This Master is run under the context of Action
No 2020-EU-IA-0087, co-financed by the EU CEF Telecom
under GA nr. INEA/CEF/ICT/A2020/2267423

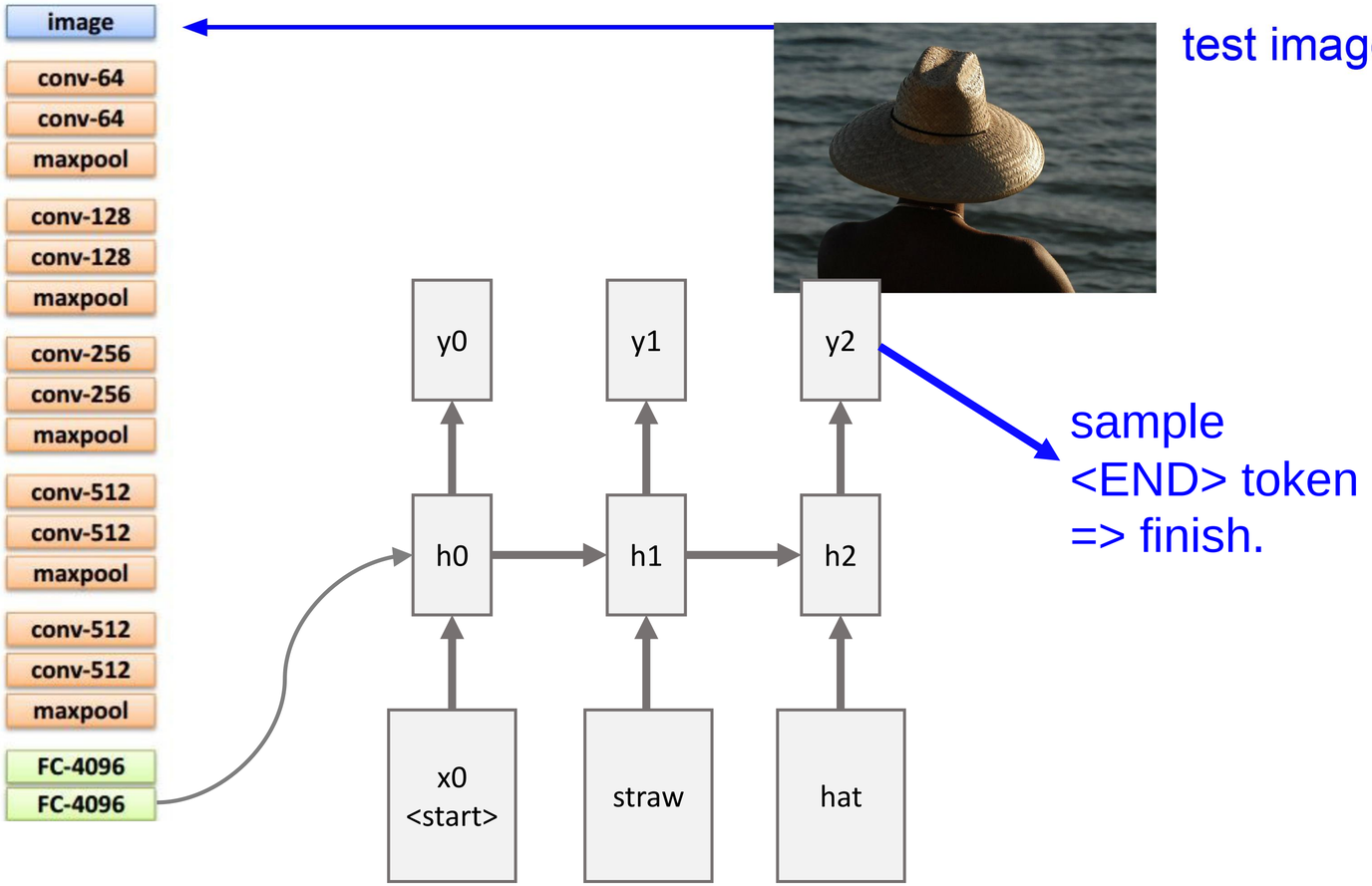
Image Captioning



Co-financed by the European Union
Connecting Europe Facility

This Master is run under the context of Action
No 2020-EU-IA-0087, co-financed by the EU CEF Telecom
under GA nr. INEA/CEF/ICT/A2020/2267423

Image Captioning



This Master is run under the context of Action
No 2020-EU-IA-0087, co-financed by the EU CEF Telecom
under GA nr. INEA/CEF/ICT/A2020/2267423

Image Captioning: *Examples*

A cat sitting on a suitcase on the floor

A cat is sitting on a tree branch

A dog is running in the grass with a frisbee

A white teddy bear sitting in the grass

Two people walking on the beach with surfboards

A tennis player in action on the court

Two giraffes standing in a grassy field

A man riding a dirt bike on a dirt track

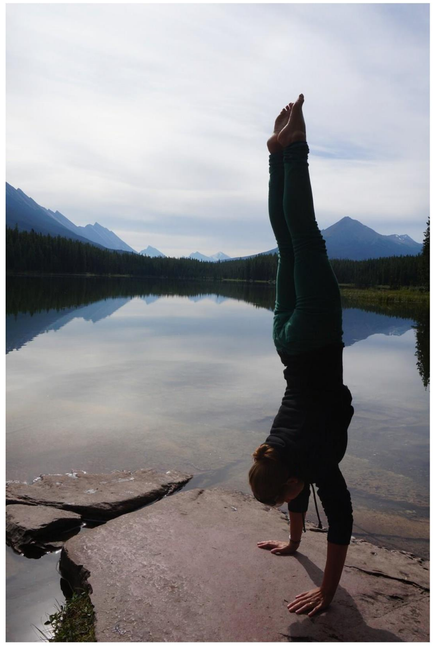
Co-financed by the European Union
Connecting Europe Facility

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Image Captioning: *Failure Cases*

A woman is holding a cat in her hand

A person holding a computer mouse on a desk



A woman standing on a beach holding a surfboard

A bird is perched on a tree branch

A man in a baseball uniform throwing a ball

Co-financed by the European Union
Connecting Europe Facility

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Visual Question Answering (VQA)

Q: What endangered animal is featured on the truck?

A: A bald eagle.
A: A sparrow.
A: A humming bird.
A: A raven.

Q: Where will the driver go if turning right?

A: Onto 24 ¾ Rd.
A: Onto 25 ¾ Rd.
A: Onto 23 ¾ Rd.
A: Onto Main Street.

Q: When was the picture taken?

A: During a wedding.
A: During a bar mitzvah.
A: During a funeral.
A: During a Sunday church service

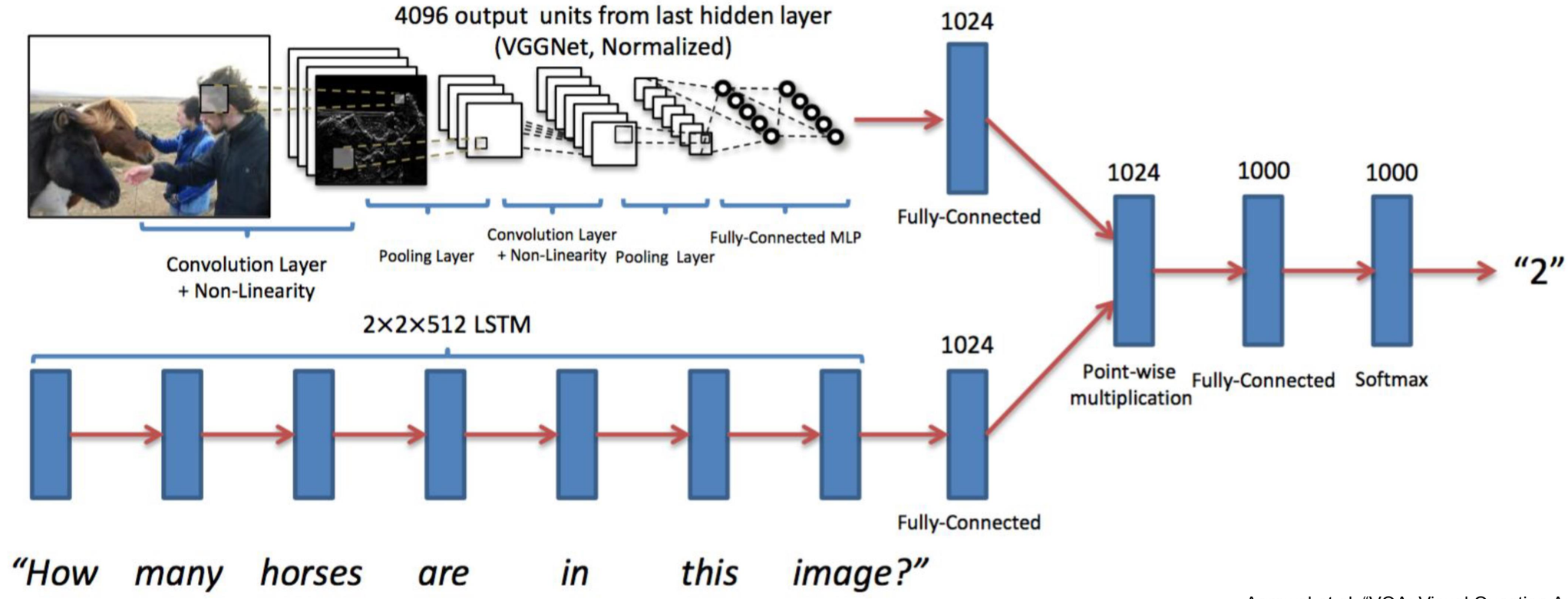
Q: Who is under the umbrella?

A: Two women.
A: A child.
A: An old man.
A: A husband and a wife.

Agrawal et al, “VQA: Visual Question Answering”, ICCV 2015
Zhu et al, “Visual 7W: Grounded Question Answering in Images”, CVPR 2016

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Visual Question Answering (VQA)



Agrawal et al, "VQA: Visual Question Answering", ICCV 2015
Zhu et al, "Visual 7W: Grounded Question Answering in Images", CVPR 2016

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Long Short Term Memory (LSTM)

Co-financed by the European Union
Connecting Europe Facility

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Long Short Term Memory (LSTM)

Vanilla RNN

$$h_t = \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

LSTM

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Long Short Term Memory (LSTM)

LSTM stands for **Long Short-Term Memory**, which is a type of Recurrent Neural Network (RNN) architecture that is designed to better handle the issue of vanishing gradients that can arise in traditional RNNs. The LSTM architecture was first introduced by Hochreiter and Schmidhuber in 1997.

The main difference between LSTM and traditional RNNs is that LSTM networks have a more complex cell structure that allows them to selectively remember or forget information from the past. This is achieved through the use of three gates within each LSTM cell: the input gate, the forget gate, and the output gate. These gates help control the flow of information through the cell, allowing it to selectively store or discard information over time, enabling it to capture long-term dependencies in data.

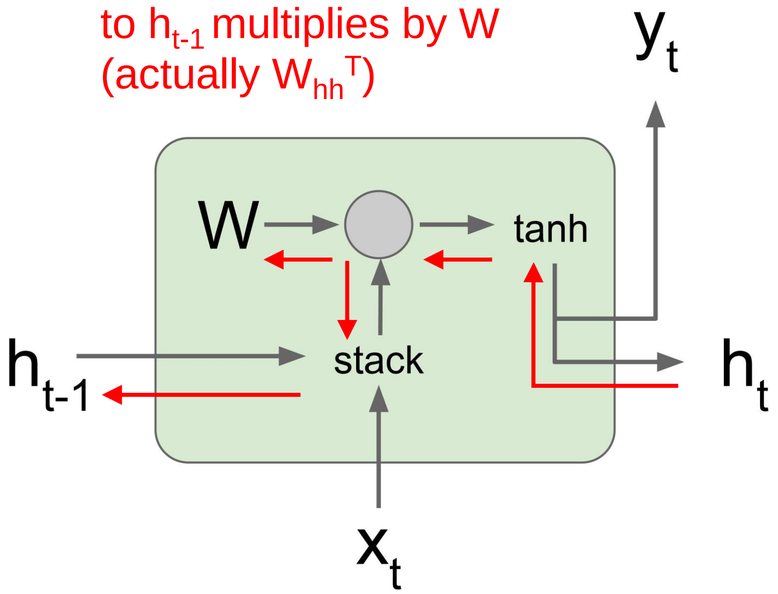
In contrast, a traditional RNN has a simpler architecture that consists of a single recurrent layer. While it is capable of modeling sequential data, it may struggle with long-term dependencies, as the gradients can vanish or explode during training, which makes it difficult to propagate information across long sequences.

Overall, LSTM is a powerful type of RNN architecture that is well-suited for modeling sequential data with long-term dependencies, while traditional RNNs may be better suited for simpler sequential data or applications where computational resources are limited.

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Vanilla RNN Gradient Flow

Backpropagation from h_t to h_{t-1} multiplies by W (actually W_{hh}^T)



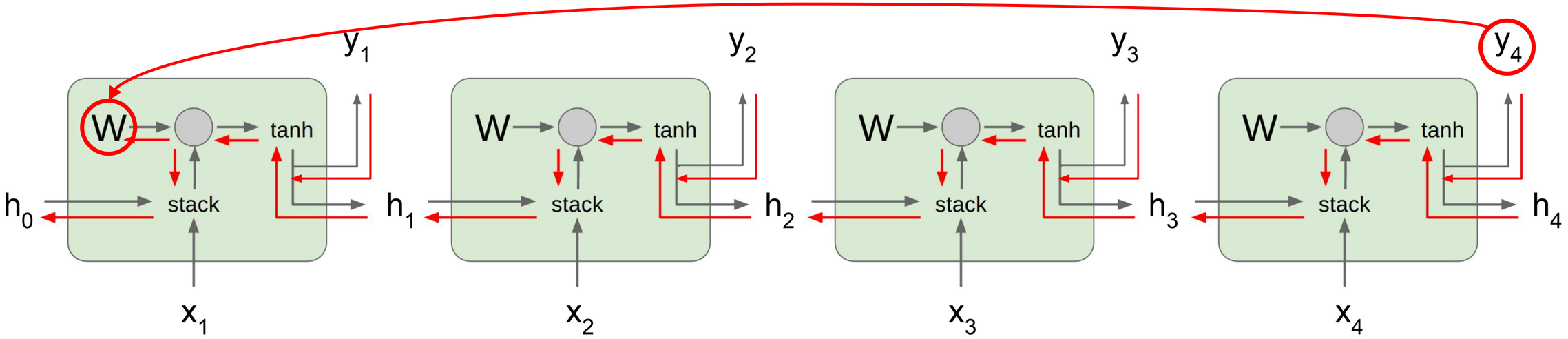
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$
$$= \tanh\left((W_{hh} \quad W_{hx})\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$
$$= \tanh\left(W\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

$$\frac{\partial h_t}{\partial h_{t-1}} = tanh'(W_{hh}h_{t-1} + W_{xh}x_t)W_{hh}$$

Bengio et al, “Learning long-term dependencies with gradient descent is difficult”, IEEE Transactions on Neural Networks, 1994
Pascanu et al, “On the difficulty of training recurrent neural networks”, ICML 2013

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Vanilla RNN Gradient Flow



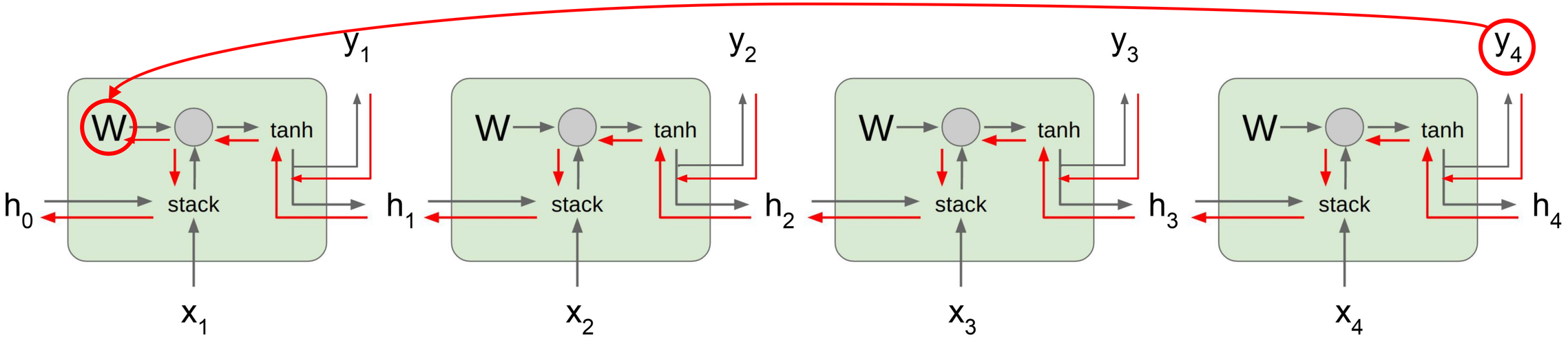
$$\frac{\partial L}{\partial W} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial W}$$

$$\frac{\partial h_t}{\partial h_{t-1}} = tanh'(W_{hh} h_{t-1} + W_{xh} x_t) W_{hh}$$

$$\frac{\partial L_T}{\partial W} = \frac{\partial L_T}{\partial h_T} \frac{\partial h_t}{\partial h_{t-1}} \cdots \frac{\partial h_1}{\partial W} = \frac{\partial L_T}{\partial h_T} \left(\prod_{t=2}^{T} \frac{\partial h_t}{\partial h_{t-1}}\right) \frac{\partial h_1}{\partial W}$$

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Vanilla RNN Gradient Flow



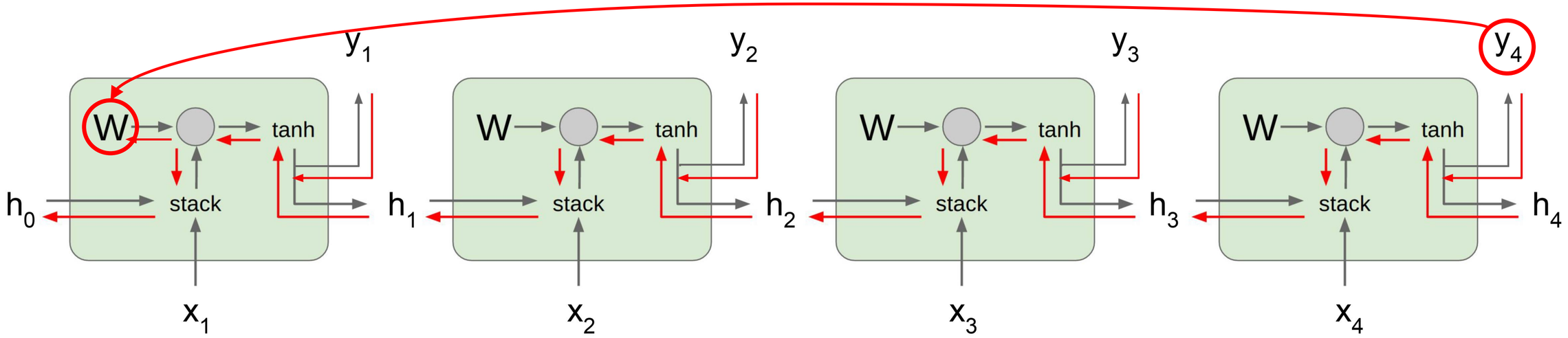
$$\frac{\partial L}{\partial W} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial W}$$

Almost always < 1
Vanishing gradients

$$\frac{\partial L_T}{\partial W} = \frac{\partial L_T}{\partial h_T} \left(\prod_{t=2}^{T} tanh'\left(W_{hh} h_{t-1} + W_{xh} x_t\right) \right) W_{hh}^{T-1} \frac{\partial h_1}{\partial W}$$

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Vanilla RNN Gradient Flow



$$\frac{\partial L}{\partial W} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial W}$$

What if we assumed no non-linearity?

$$\frac{\partial L_T}{\partial W} = \frac{\partial L_T}{\partial h_T} \boxed{W_{hh}^{T-1}} \frac{\partial h_1}{\partial W}$$

Largest singular value > 1:
Exploding gradients

Largest singular value < 1:
Vanishing gradients

Gradient clipping:
Scale gradient if its norm is too big

Change RNN architecture

This Master is run under the context of Action
No 2020-EU-IA-0087, co-financed by the EU CEF Telecom
under GA nr. INEA/CEF/ICT/A2020/2267423

Long Short Term Memory (LSTM)

Vanilla RNN

$$h_t = \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

LSTM

Four gates

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

Cell state

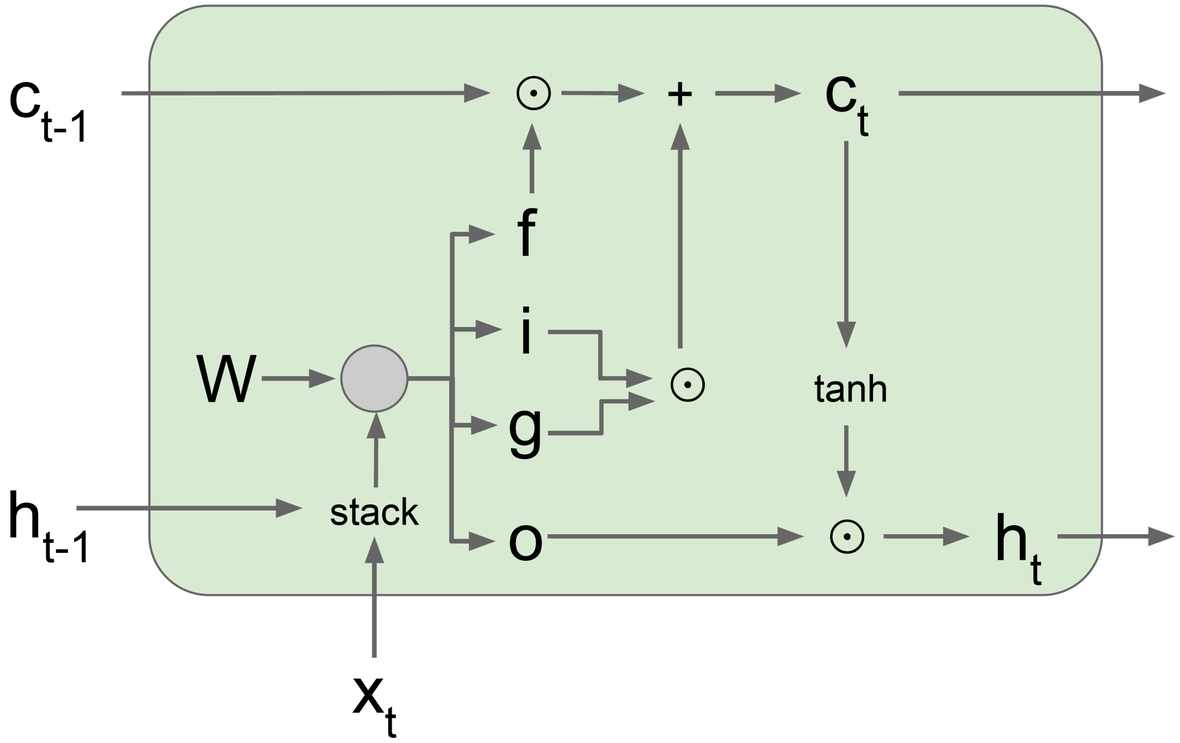
$$c_t = f \odot c_{t-1} + i \odot g$$

Hidden state

$$h_t = o \odot \tanh(c_t)$$

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Long Short Term Memory (LSTM)



i: Input gate, whether to write to cell

f: Forget gate, Whether to erase cell

o: Output gate, How much to reveal cell

g: Gate gate (?), How much to write to cell

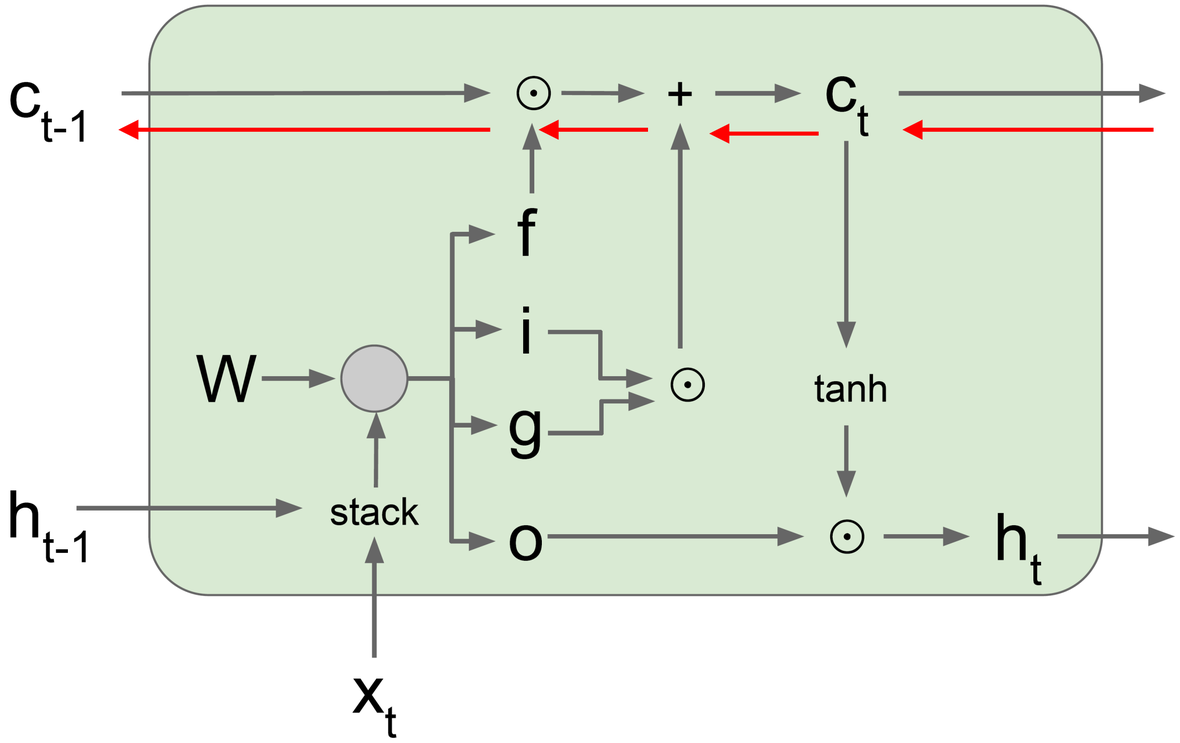
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Long Short Term Memory (LSTM)



Backpropagation from c_t to c_{t-1} only elementwise multiplication by f, no matrix multiply by W

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

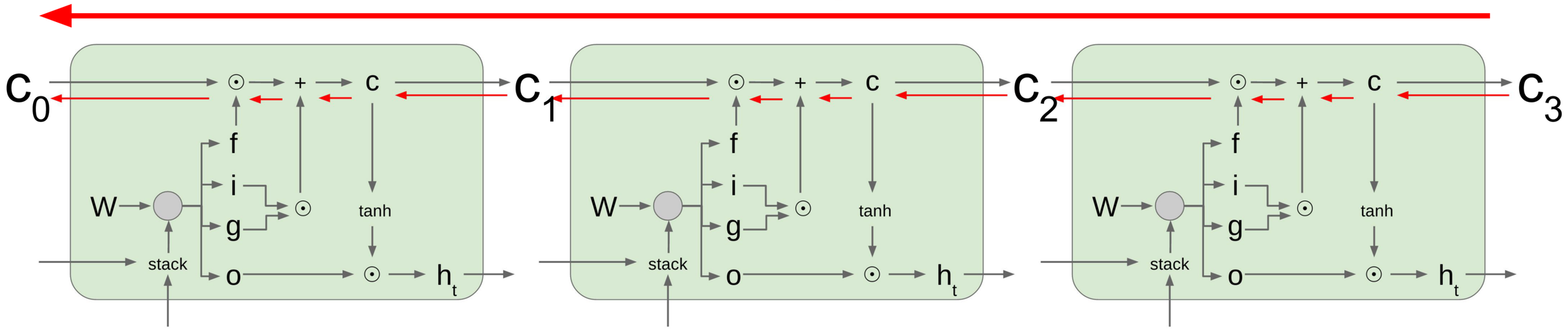
$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Long Short Term Memory (LSTM): *Gradient Flow*

Uninterrupted gradient flow!



Notice that the gradient contains the **f** gate's vector of activations
- allows better control of gradients values, using suitable parameter updates of the forget gate.

Also notice that are added through the **f, i, g,** and **o** gates
- better balancing of gradient values

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Long Short Term Memory (LSTM): *Gradient Flow*

Do LSTMs solve the vanishing gradient problem?

The LSTM architecture makes it easier for the RNN to preserve information over many timesteps

- e.g. **if the f = 1 and the i = 0**, then the information of that cell is preserved indefinitely.
- By contrast, it's harder for vanilla RNN to learn a recurrent weight matrix W_h that preserves info in hidden state

LSTM **doesn't guarantee** that there is no vanishing/exploding gradient, but it does provide an easier way for the model to learn long-distance dependencies

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Long Short Term Memory (LSTM): *Summary*

RNNs allow a lot of flexibility in architecture design

- Vanilla RNNs are simple but don’t work very well
- Common to use LSTM or GRU: their additive interactions improve gradient flow
- Backward flow of gradients in RNN can explode or vanish. Exploding is controlled with gradient clipping. Vanishing is controlled with additive interactions (LSTM)
- Better/simpler architectures are a hot topic of current research, as well as new paradigms for reasoning over sequences
- Better understanding (both theoretical and empirical) is needed.

Co-financed by the European Union
Connecting Europe Facility

This Master is run under the context of Action
No 2020-EU-IA-0087, co-financed by the EU CEF Telecom
under GA nr. INEA/CEF/ICT/A2020/2267423

Attention and Transformers

Co-financed by the European Union
Connecting Europe Facility

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Sequence to Sequence with RNNs

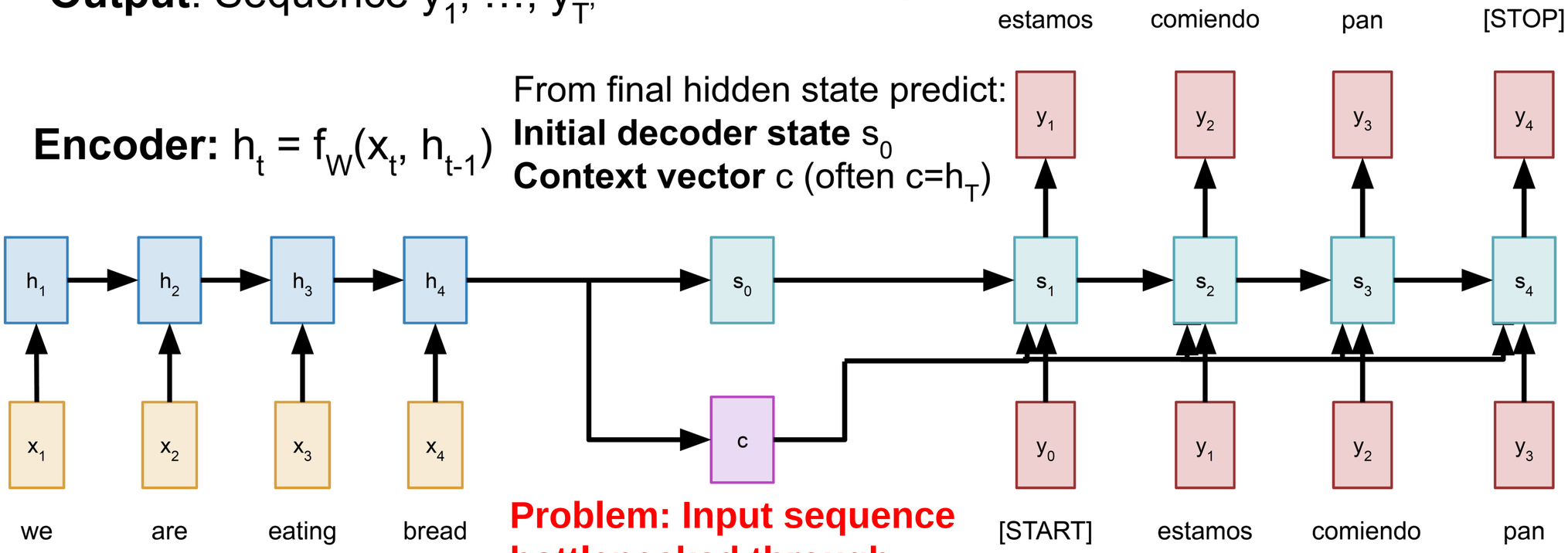
Input: Sequence $x_1, \ldots x_T$
Output: Sequence $y_1, \ldots, y_{T'}$

Idea: use new context vector at each step of decoder!

Decoder: $s_t = g_U(y_{t-1}, s_{t-1}, c)$

Encoder: $h_t = f_W(x_t, h_{t-1})$

From final hidden state predict:
Initial decoder state s_0
Context vector c (often $c=h_T$)



Problem: Input sequence bottlenecked through fixed-sized vector. What if T=1000?

Co-financed by the European Union
Connecting Europe Facility

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Sequence to Sequence with RNNs and Attention

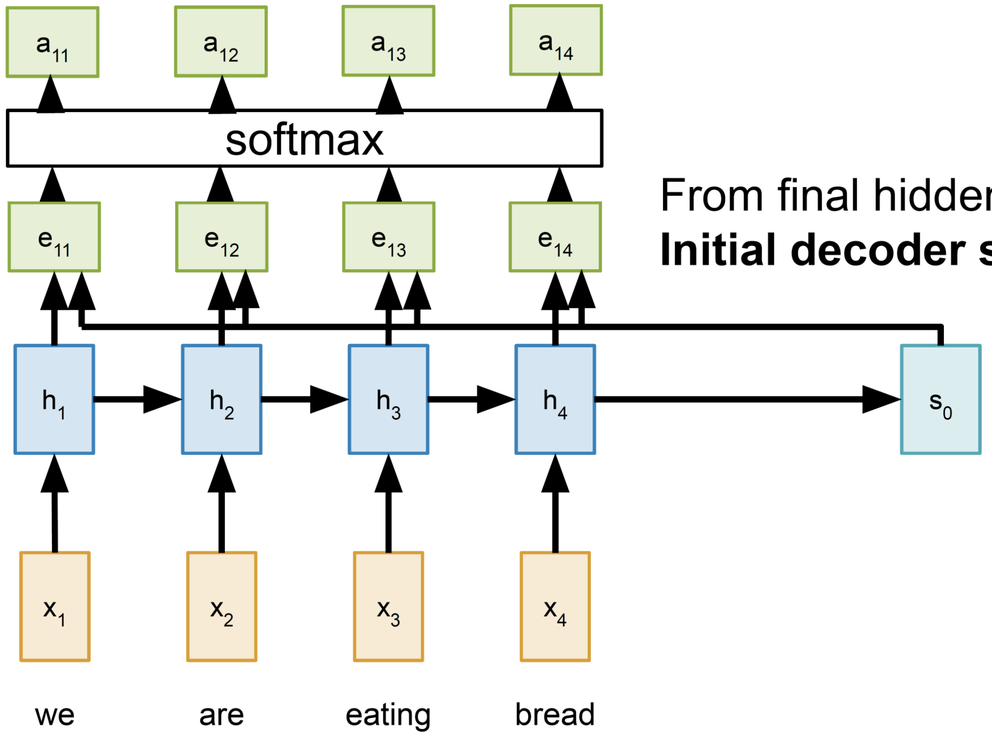
In Recurrent Neural Networks (RNNs), **attention** is a mechanism that enables the network to focus on specific parts of the input sequence when processing the output at a given time step.

In a traditional RNN, the network processes the entire input sequence and generates an output at each time step. However, attention allows the network to selectively attend to certain parts of the input sequence while ignoring others. This can be useful in tasks such as machine translation, where the network may need to focus on specific words in the source sentence to accurately translate them into the target language.

The attention mechanism works by assigning a weight to each element in the input sequence based on how relevant it is to the current output. These weights are then used to calculate a weighted sum of the input elements, which is used as input to the current time step of the network. The weights are learned during training and can be interpreted as a measure of how much attention the network is paying to each element in the input sequence.

This Master is run under the context of Action
No 2020-EU-IA-0087, co-financed by the EU CEF Telecom
under GA nr. INEA/CEF/ICT/A2020/2267423

Sequence to Sequence with RNNs and Attention



From final hidden state:
Initial decoder state s_0

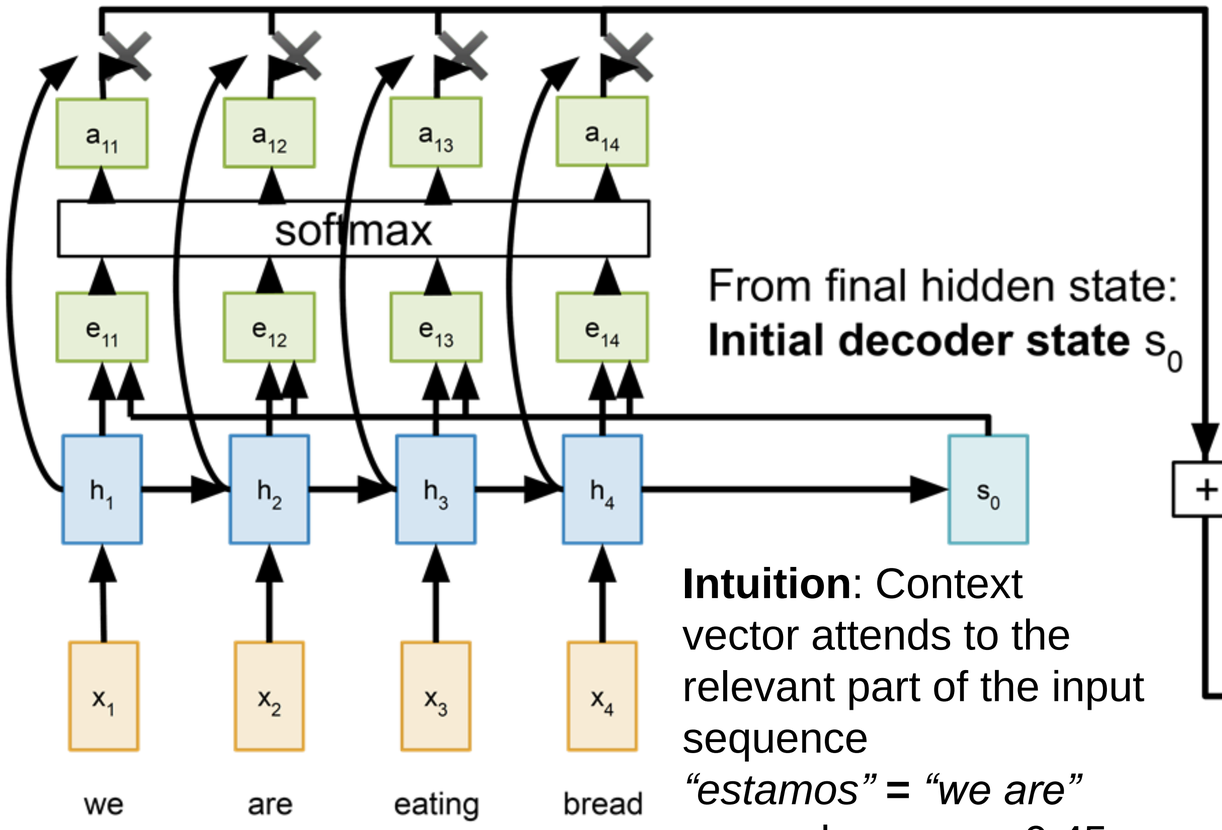
Compute (scalar) **alignment scores**
$e_{t,i} = f_{att}(s_{t-1}, h_i)$ (f_{att} is an MLP)

Normalize alignment scores to get **attention weights** $0 < a_{t,i} < 1$ $\Sigma_i a_{t,i} = 1$

Bahdanau et al, “Neural machine translation by jointly learning to align and translate”, ICLR 2015

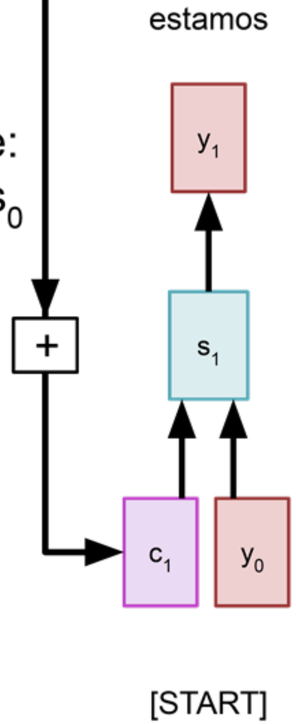
This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Sequence to Sequence with RNNs and Attention



From final hidden state:
Initial decoder state s_0

Intuition: Context vector attends to the relevant part of the input sequence
"estamos" = *"we are"*
so maybe $a_{11}=a_{12}=0.45$, $a_{13}=a_{14}=0.05$



Compute (scalar) **alignment scores**
$e_{t,i} = f_{att}(s_{t-1}, h_i)$ (f_{att} is an MLP)

Normalize alignment scores to get **attention weights**
$0 < a_{t,i} < 1 \quad \sum_i a_{t,i} = 1$

Compute context vector as linear combination of hidden states
$c_t = \sum_i a_{t,i} h_i$

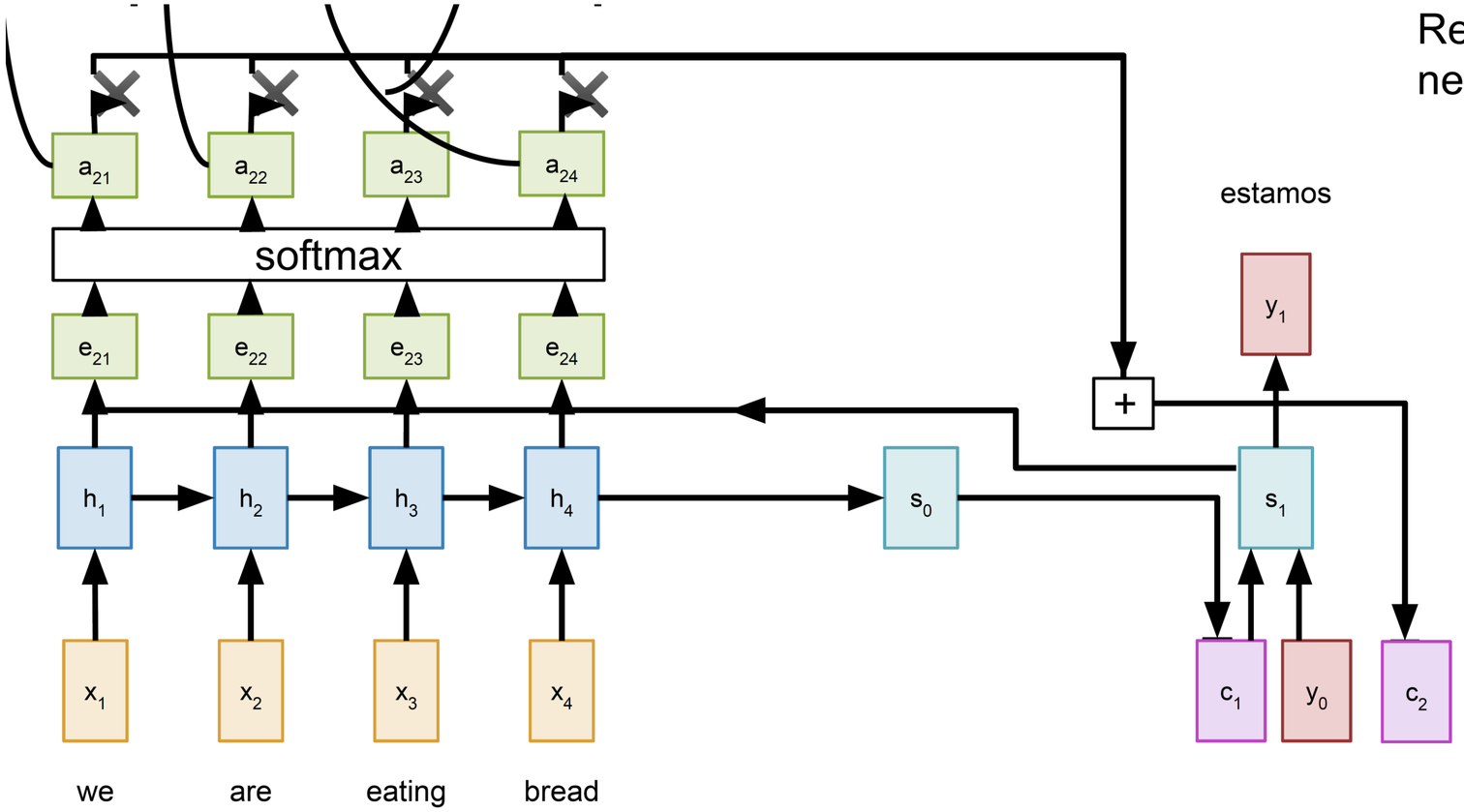
Use context vector in decoder: $s_t = g_U(y_{t-1}, s_{t-1}, c_t)$

This is all differentiable! No supervision on attention weights – backprop through everything

Bahdanau et al, "Neural machine translation by jointly learning to align and translate", ICLR 2015

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Sequence to Sequence with RNNs and Attention

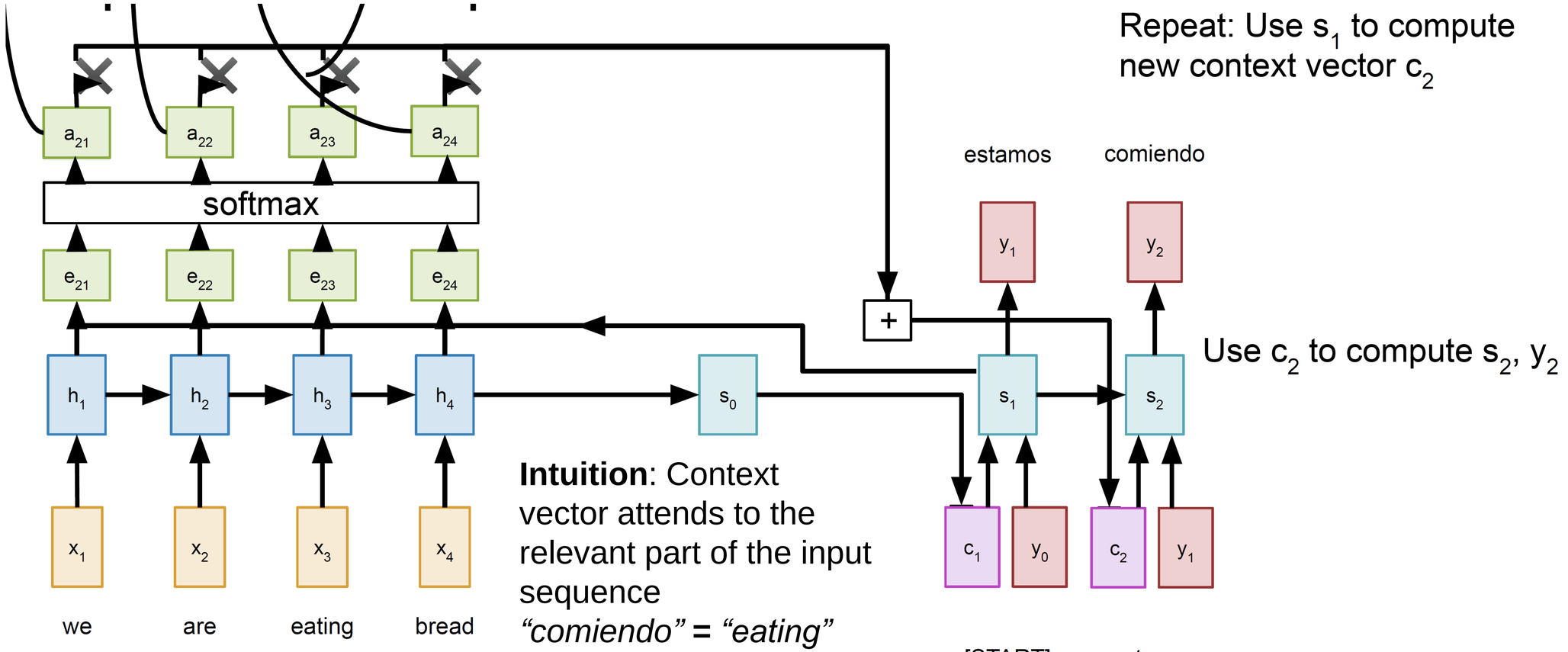


Repeat: Use s_1 to compute new context vector c_2

Bahdanau et al, “Neural machine translation by jointly learning to align and translate”, ICLR 2015

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Sequence to Sequence with RNNs and Attention



Intuition: Context vector attends to the relevant part of the input sequence

"comiendo" **=** *"eating"*

so maybe a_{21}=a_{24}=0.05, a_{22}=0.1, a_{23}=0.8

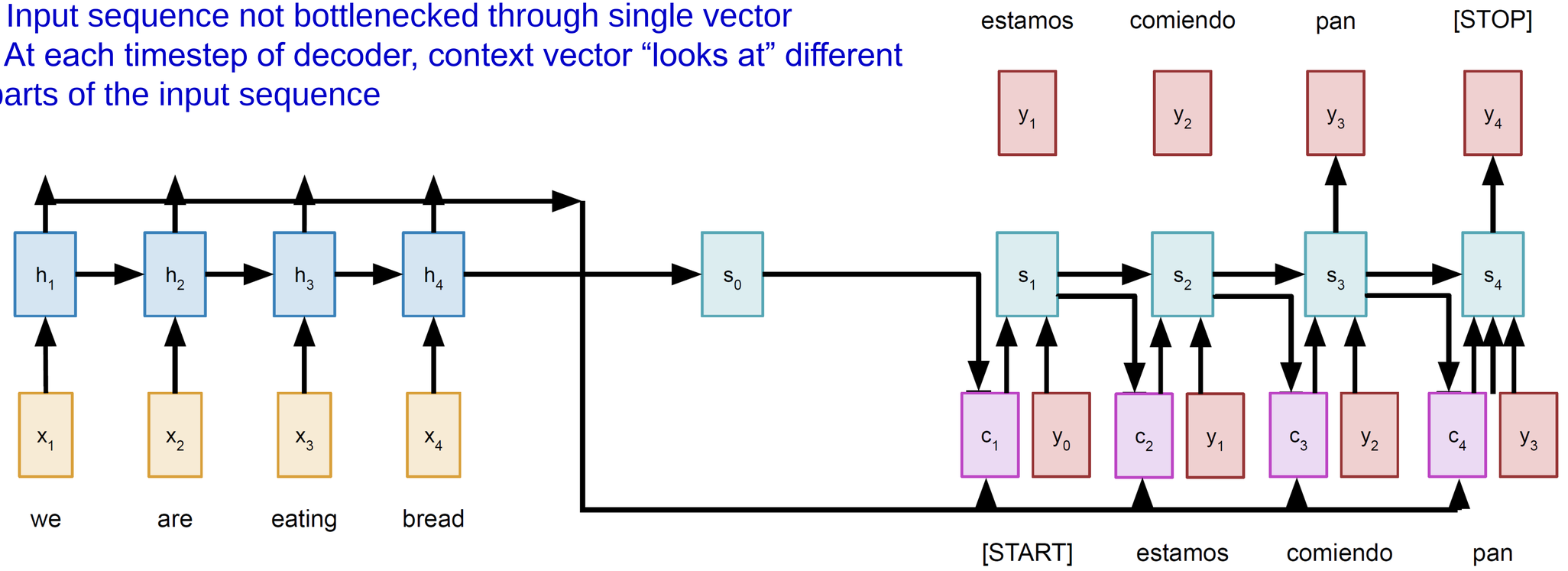
Bahdanau et al, "Neural machine translation by jointly learning to align and translate", ICLR 2015

Co-financed by the European Union
Connecting Europe Facility

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Sequence to Sequence with RNNs and Attention

Use a different context vector in each timestep of decoder
- Input sequence not bottlenecked through single vector
- At each timestep of decoder, context vector “looks at” different parts of the input sequence



Bahdanau et al, “Neural machine translation by jointly learning to align and translate”, ICLR 2015

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

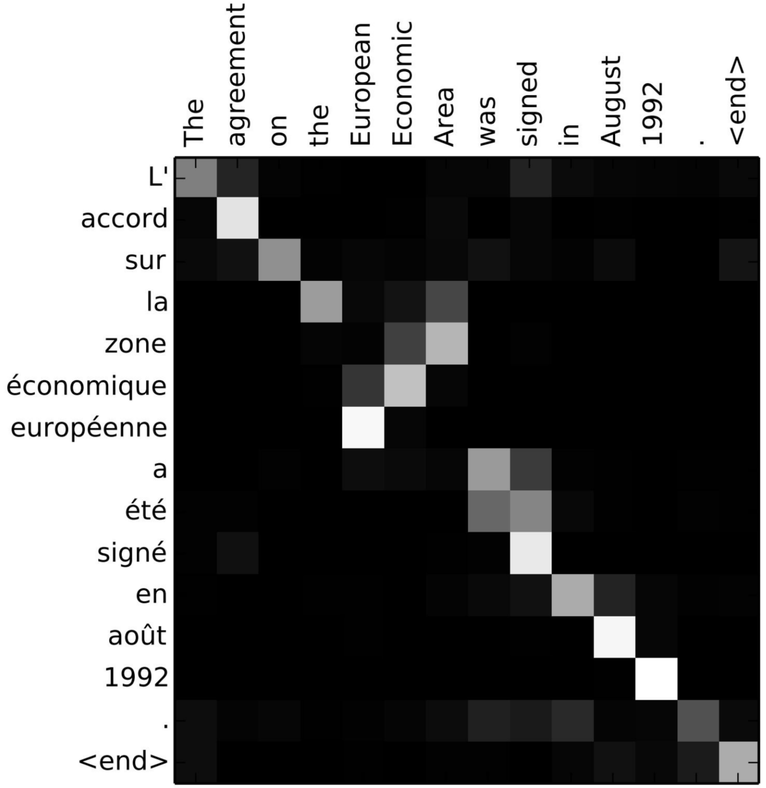
Sequence to Sequence with RNNs and Attention

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.”

Visualize attention weights $a_{t,i}$



Bahdanau et al, “Neural machine translation by jointly learning to align and translate”, ICLR 2015

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

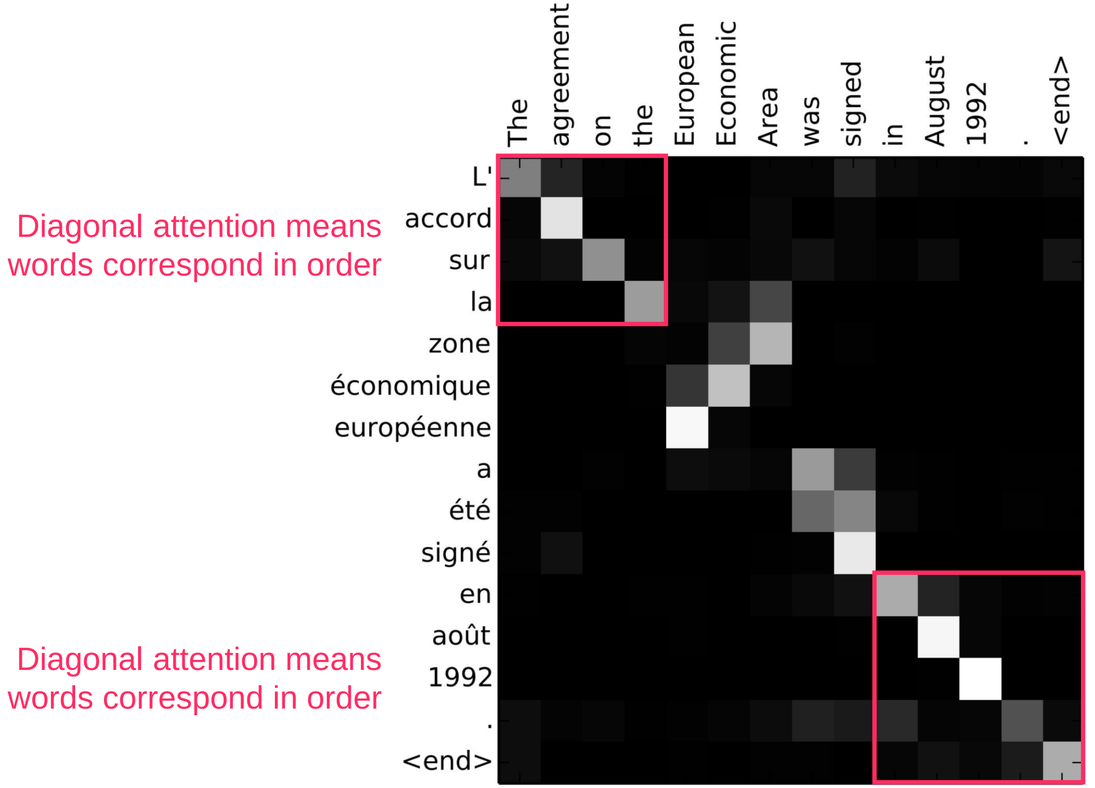
Sequence to Sequence with RNNs and Attention

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**."

Visualize attention weights $a_{t,i}$



Bahdanau et al, "Neural machine translation by jointly learning to align and translate", ICLR 2015

Co-financed by the European Union
Connecting Europe Facility

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

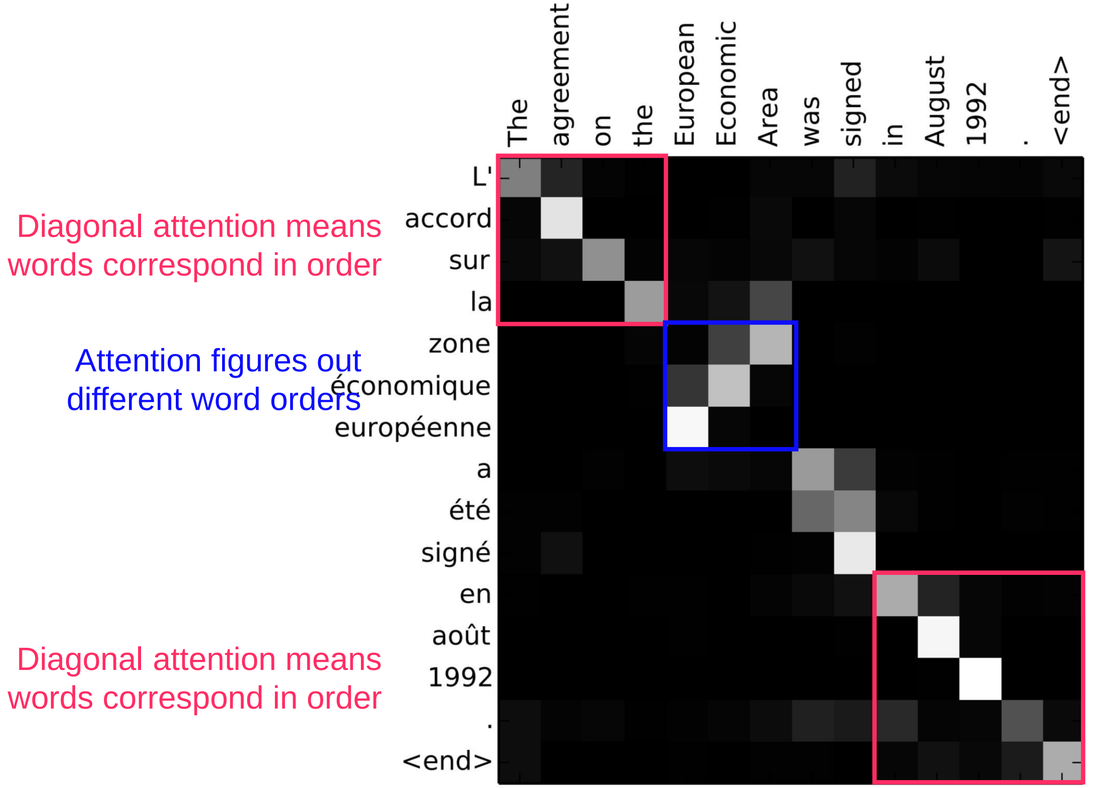
Sequence to Sequence with RNNs and Attention

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**."

Visualize attention weights $a_{t,i}$



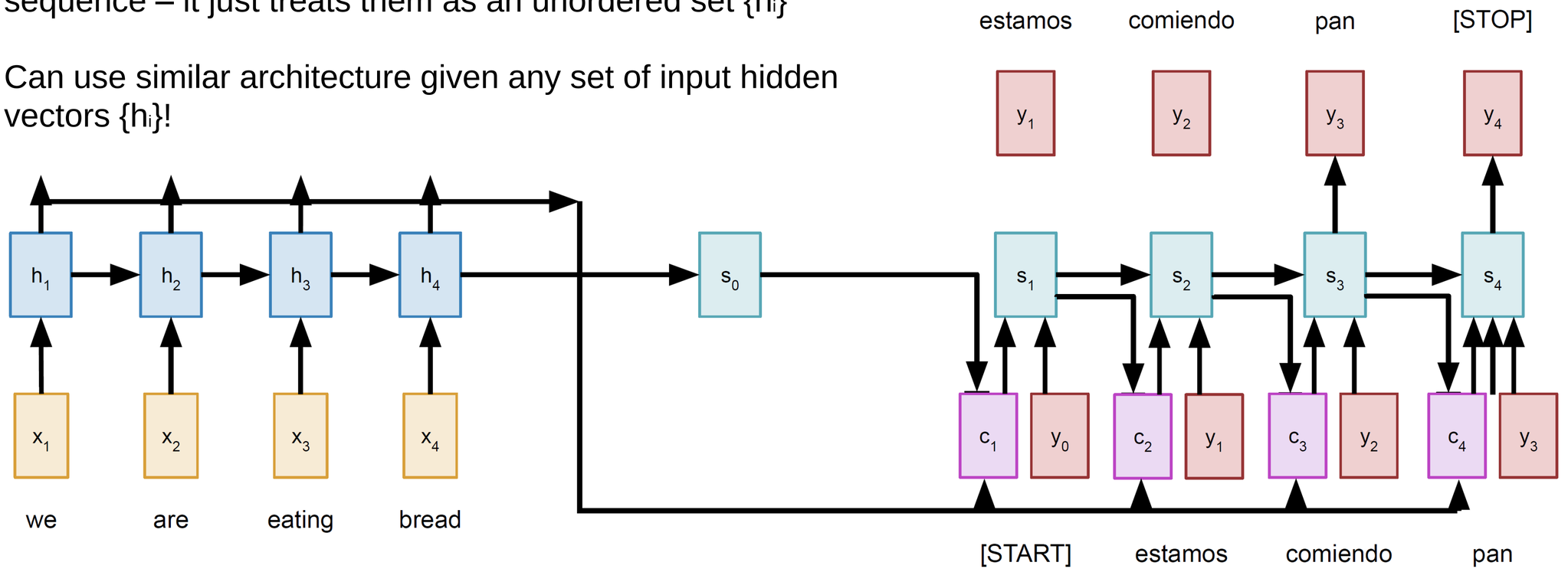
Bahdanau et al, "Neural machine translation by jointly learning to align and translate", ICLR 2015

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Sequence to Sequence with RNNs and Attention

The decoder doesn't use the fact that h_i form an ordered sequence – it just treats them as an unordered set $\{h_i\}$

Can use similar architecture given any set of input hidden vectors $\{h_i\}$!



Bahdanau et al, "Neural machine translation by jointly learning to align and translate", ICLR 2015

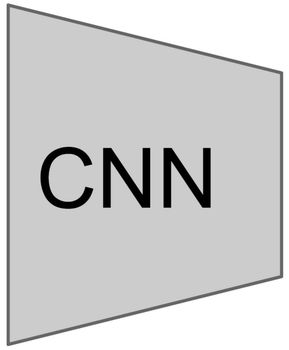
This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

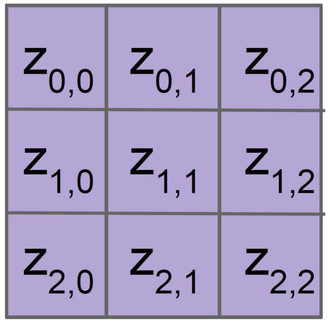
Image Captioning using spatial features

Input: Image **I**
Output: Sequence **y** = $y_1, y_2, ..., y_T$

Encoder: $h_0 = f_w(\mathbf{z})$
where **z** is spatial CNN features
f_w**(.)** is an MLP

Extract spatial features from a pretrained CNN

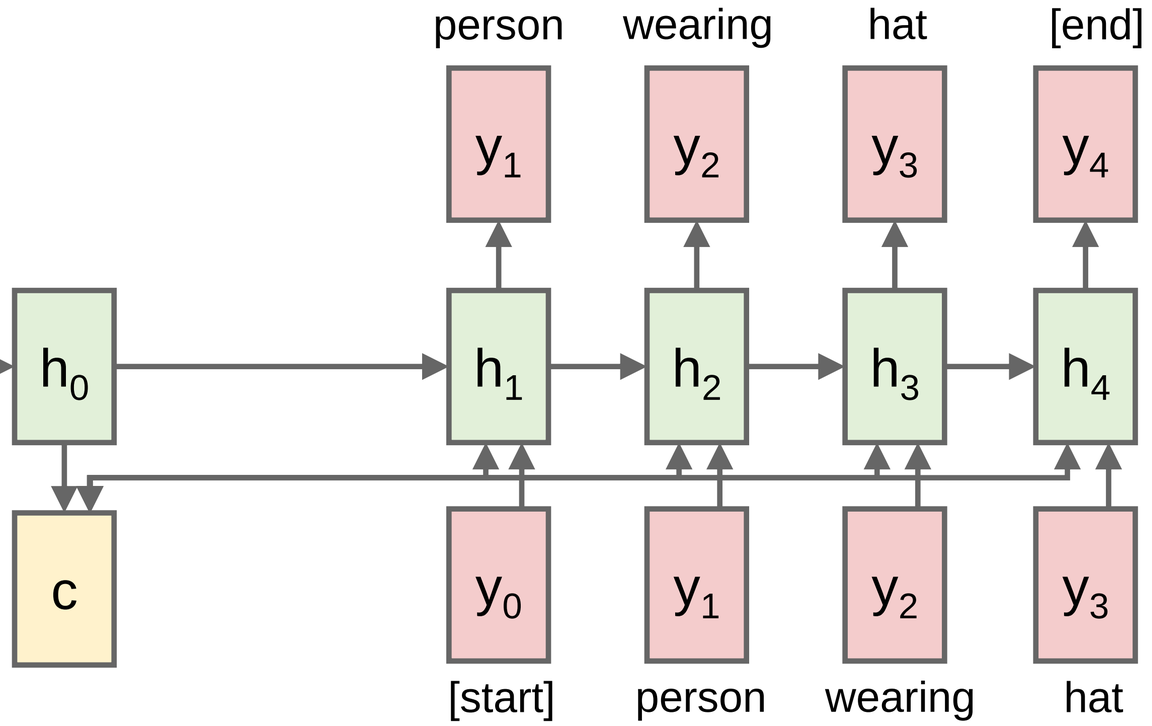




Features:
H x W x D

MLP

Decoder: $y_t = g_v(y_{t-1}, h_{t-1}, c)$
where context vector c is often $c = h_0$



Xu et al, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, ICML 2015

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Image Captioning using spatial features

Problem: Input is "bottlenecked" through c
- Model needs to encode everything it wants to say within c

This is a problem if we want to generate really long descriptions? 100s of words long

Extract spatial features from a pretrained CNN

CNN

$z_{0,0}$ $z_{0,1}$ $z_{0,2}$
$z_{1,0}$ $z_{1,1}$ $z_{1,2}$
$z_{2,0}$ $z_{2,1}$ $z_{2,2}$

Features: H x W x D

MLP

h_0 h_1 h_2 h_3 h_4

c

y_0 y_1 y_2 y_3

[start] person wearing hat

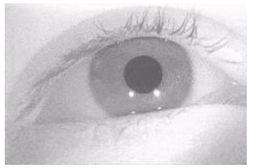
y_1 y_2 y_3 y_4

person wearing hat [end]

Xu et al, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, ICML 2015

Co-financed by the European Union
Connecting Europe Facility

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423



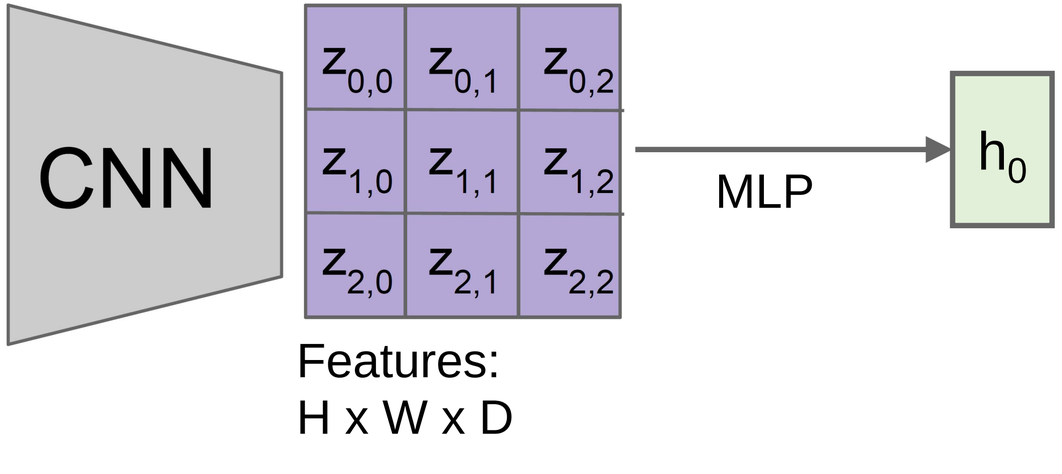
Attention Saccades in humans

Image Captioning with RNNs and Attention

Attention idea: New context vector at every time step.

Each context vector will attend to different image regions

Extract spatial features from a pretrained CNN



Features: H x W x D

Xu et al, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, ICML 2015

Co-financed by the European Union
Connecting Europe Facility

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

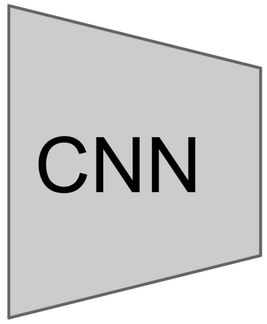
Image Captioning with RNNs and Attention

Compute alignments scores (scalars):

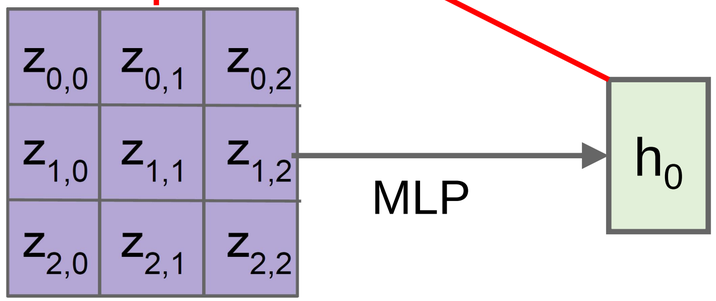
$$e_{t,i,j} = f_{att}(h_{t-1}, z_{i,j})$$

$f_{att}(.)$ is an MLP

Extract spatial features from a pretrained CNN

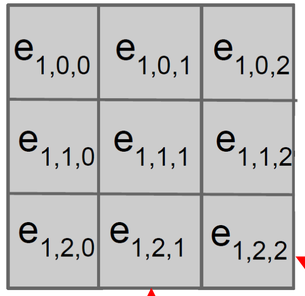


Features: H x W x D



MLP

Alignment scores: H x W



Xu et al, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, ICML 2015

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Image Captioning with RNNs and Attention

Compute alignments scores (scalars):

$$e_{t,i,j} = f_{att}(h_{t-1}, z_{i,j})$$

$f_{att}(.)$ is an MLP

Alignment scores: H x W

$e_{1,0,0}$ $e_{1,0,1}$ $e_{1,0,2}$
$e_{1,1,0}$ $e_{1,1,1}$ $e_{1,1,2}$
$e_{1,2,0}$ $e_{1,2,1}$ $e_{1,2,2}$

Attention: H x W

$a_{1,0,0}$ $a_{1,0,1}$ $a_{1,0,2}$
$a_{1,1,0}$ $a_{1,1,1}$ $a_{1,1,2}$
$a_{1,2,0}$ $a_{1,2,1}$ $a_{1,2,2}$

Normalize to get attention weights:

$$a_{t,:,:} = softmax(e_{t,:,:})$$

$0 < a_{t,i,j} < 1$,
attention values sum to 1

Compute context vector:

$$c_t = \sum_{i,j} a_{t,i,j} z_{t,i,j}$$

Extract spatial features from a pretrained CNN

CNN

$z_{0,0}$ $z_{0,1}$ $z_{0,2}$
$z_{1,0}$ $z_{1,1}$ $z_{1,2}$
$z_{2,0}$ $z_{2,1}$ $z_{2,2}$

Features: H x W x D

MLP

h_0

c_1

x

Co-financed by the European Union
Connecting Europe Facility

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

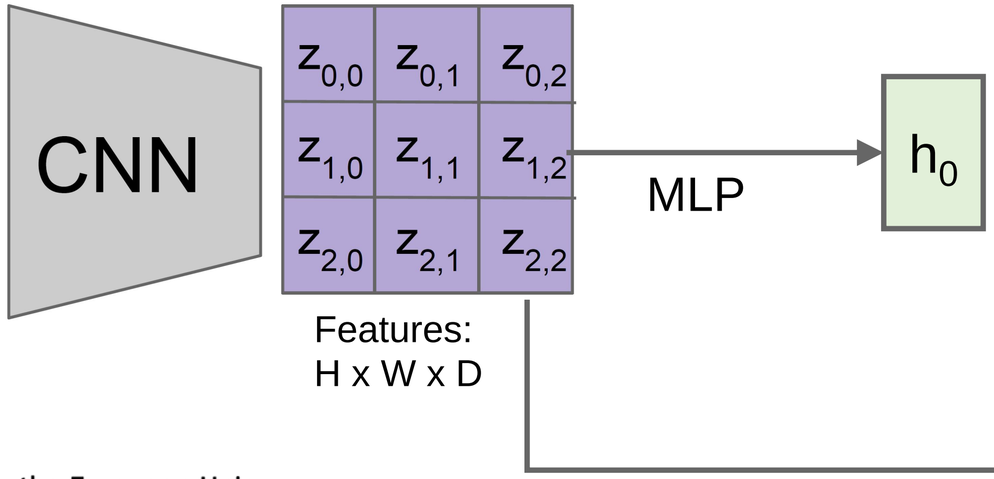
Image Captioning with RNNs and Attention

Each timestep of decoder uses a different context vector that looks at different parts of the input image

$$e_{t,i,j} = f_{att}(h_{t-1}, z_{i,j})$$
$$a_{t,:,:} = softmax(e_{t,:,:})$$
$$c_t = \sum_{i,j} a_{t,i,j} z_{t,i,j}$$

Decoder: $y_t = g_V(y_{t-1}, h_{t-1}, c_t)$
New context vector at every time step

Extract spatial features from a pretrained CNN



This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Image Captioning with RNNs and Attention

Decoder: $y_t = g_V(y_{t-1}, h_{t-1}, c_t)$
New context vector at every time step

$$e_{t,i,j} = f_{att}(h_{t-1}, z_{i,j})$$
$$a_{t,:,:} = softmax(e_{t,:,:})$$
$$c_t = \sum_{i,j} a_{t,i,j} z_{t,i,j}$$

Extract spatial features from a pretrained CNN

CNN

$z_{0,0}$ $z_{0,1}$ $z_{0,2}$
$z_{1,0}$ $z_{1,1}$ $z_{1,2}$
$z_{2,0}$ $z_{2,1}$ $z_{2,2}$

Features: H x W x D

MLP

h_0

y_1

h_1

c_1 y_0

[START]

Co-financed by the European Union
Connecting Europe Facility

Image Captioning with RNNs and Attention

$$e_{t,i,j} = f_{att}(h_{t-1}, z_{i,j})$$
$$a_{t,:,:} = softmax(e_{t,:,:})$$
$$c_t = \sum_{i,j} a_{t,i,j} z_{t,i,j}$$

Alignment scores: H x W

Attention: H x W

Decoder: $y_t = g_v(y_{t-1}, h_{t-1}, c_t)$
New context vector at every time step

Extract spatial features from a pretrained CNN

CNN

Features: H x W x D

MLP

person

[START]

Co-financed by the European Union
Connecting Europe Facility

Image Captioning with RNNs and Attention

Each timestep of decoder uses a different context vector that looks at different parts of the input image

Decoder: $y_t = g_V(y_{t-1}, h_{t-1}, c_t)$
New context vector at every time step

$$e_{t,i,j} = f_{att}(h_{t-1}, z_{i,j})$$
$$a_{t,:,:} = softmax(e_{t,:,:})$$
$$c_t = \sum_{i,j} a_{t,i,j} z_{t,i,j}$$

Extract spatial features from a pretrained CNN

CNN

$z_{0,0}$ $z_{0,1}$ $z_{0,2}$
$z_{1,0}$ $z_{1,1}$ $z_{1,2}$
$z_{2,0}$ $z_{2,1}$ $z_{2,2}$

Features: H x W x D

MLP

h_0 h_1 h_2 h_3 …

y_1 y_2 y_3

person wearing hat

c_1 y_0 c_2 y_1 c_3 y_2

[START] person wearing

Co-financed by the European Union
Connecting Europe Facility

Image Captioning with RNNs and Attention

This entire process is differentiable.
- model chooses its own attention weights. No attention supervision is required

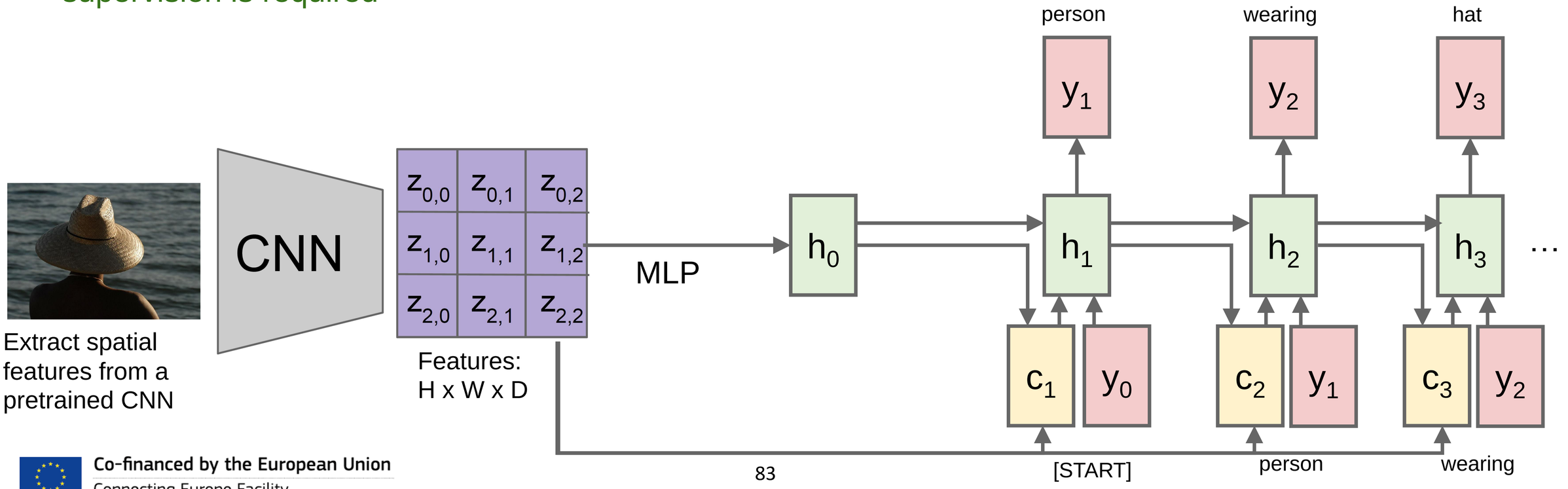
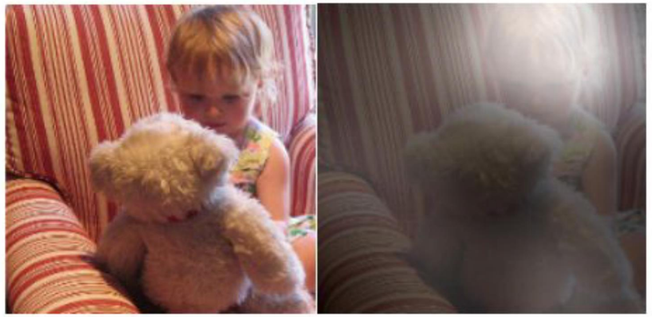


Image Captioning with RNNs and Attention

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

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



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

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

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

Xu et al, “Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention”, ICML 2015

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

General and self attention layers

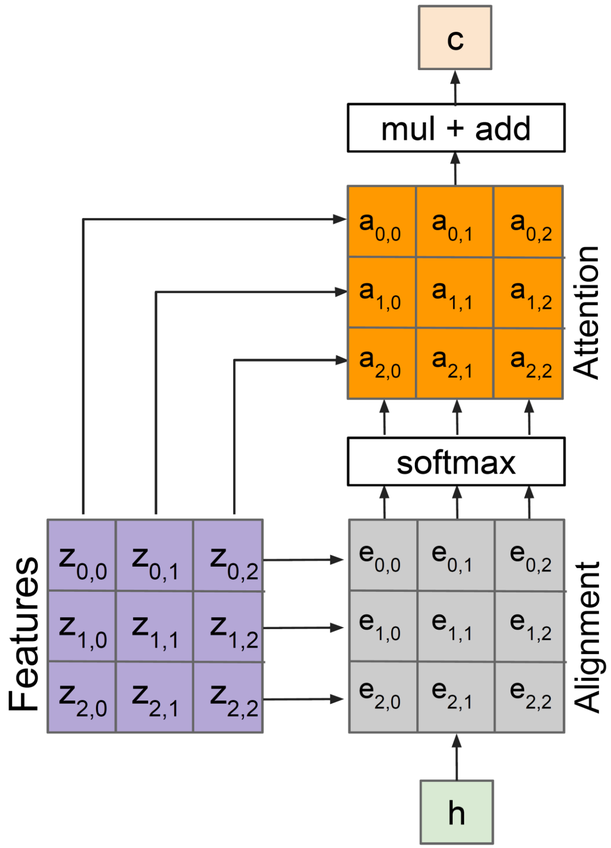
A **general attention layer** is a mechanism that learns how much attention should be paid to each element in a sequence of inputs. It works by calculating a weight for each input element based on its relevance to the current context, and then combining the elements using a weighted sum. The weights are learned through backpropagation during training.

On the other hand, **self-attention**, also known as intra-attention or multi-head attention, is a special type of attention mechanism that is used to capture the relationships between different elements of the input sequence itself. Self-attention layers calculate the attention weights by comparing every pair of elements in a sequence with each other. In other words, the self-attention mechanism allows each element in the sequence to "attend to" (or pay attention to) all the other elements in the sequence, and then use this information to compute a weighted sum.

In summary, while general attention layers learn to attend to all input elements based on their relevance to the current context, self-attention layers enable each element in the input sequence to attend to all the other elements in the sequence itself, and then use this information to compute a weighted sum.

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Attention we just saw in image captioning



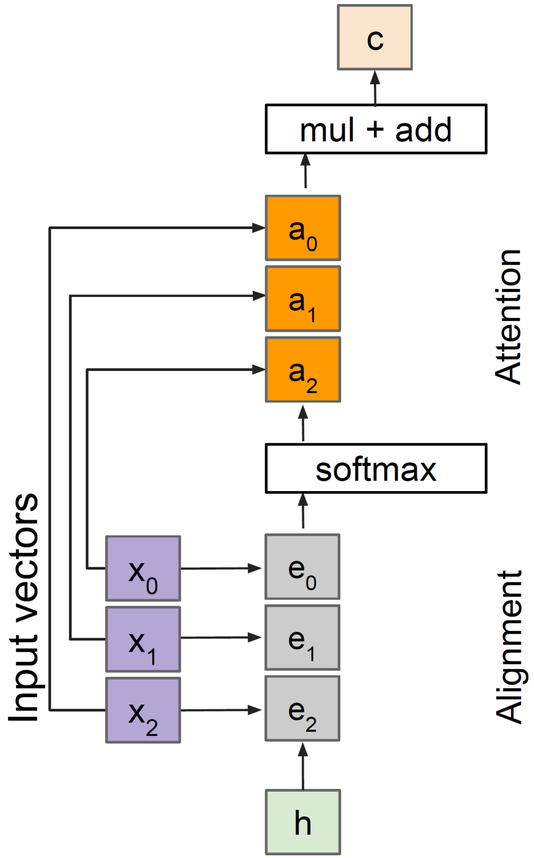
Outputs:
context vector: **c** (shape: D)

Operations:
Alignment: $e_{i,j} = f_{att}(h, z_{i,j})$
Attention: **a** = softmax(**e**)
Output: $\mathbf{c} = \sum_{i,j} a_{i,j} z_{i,j}$

Inputs:
Features: **z** (shape: H x W x D)
Query: **h** (shape: D)

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

General attention layer



Outputs:
context vector: **c** (shape: D)

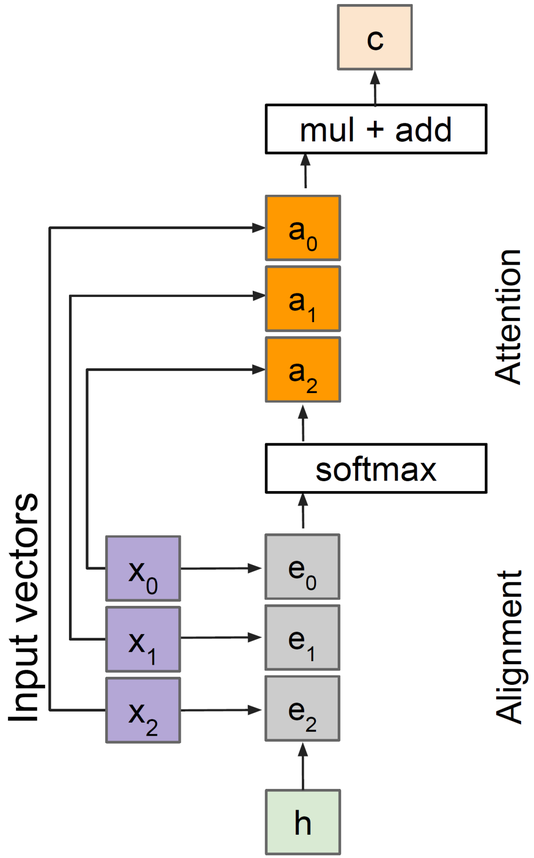
Operations:
Alignment: $e_i = f_{att}(h, x_i)$
Attention: **a** = softmax(**e**)
Output: $\mathbf{c} = \sum_i a_i x_i$

Inputs:
Input vectors: **x** (shape: N x D)
Query: **h** (shape: D)

Attention operation is **permutation invariant.**
- Doesn't care about ordering of the features
- Stretch H x W = N into N vectors

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

General attention layer



Outputs:
context vector: **c** (shape: D)

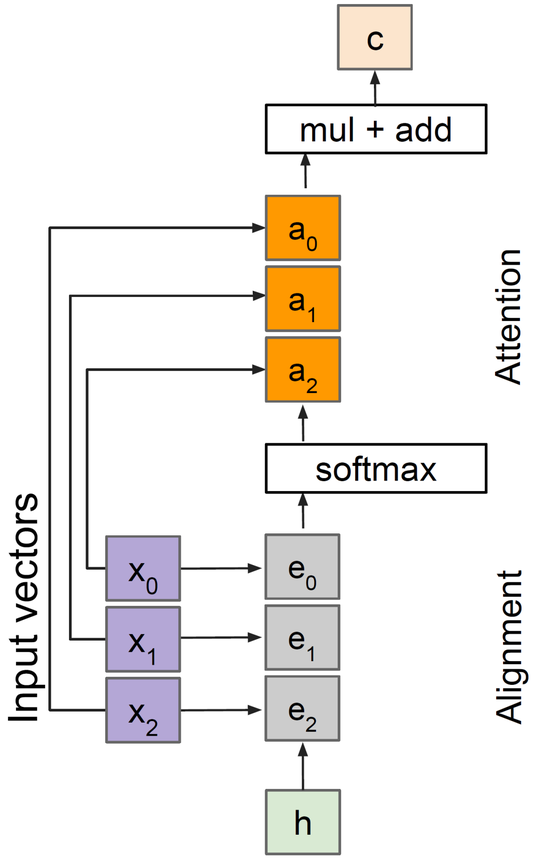
Operations:
Alignment: $e_i = h \cdot x_i$
Attention: **a** = softmax(**e**)
Output: **c** = $\sum_i a_i x_i$

Change $f_{att}(.)$ to a simple dot product
- only works well with key & value transformation trick (will mention in a few slides)

Inputs:
Input vectors: **x** (shape: N x D)
Query: **h** (shape: D)

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

General attention layer



Outputs:
context vector: **c** (shape: D)

Operations:
Alignment: $e_i = h \cdot x_i / \sqrt{D}$
Attention: **a** = softmax(**e**)
Output: **c** = $\sum_i a_i x_i$

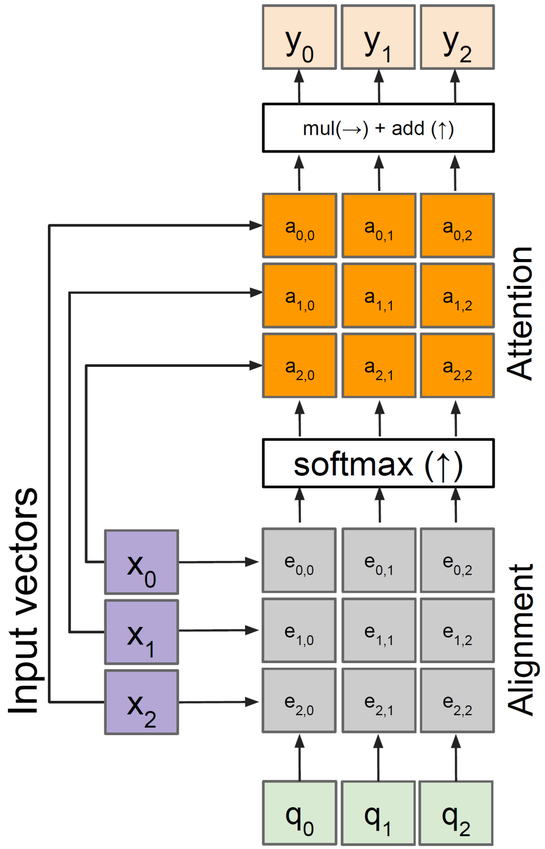
Change $f_{att}(.)$ to a scaled simple dot product

- Larger dimensions means more terms in the dot product sum.
- So, the variance of the logits is higher. Large magnitude vectors will produce much higher logits.
- So, the post-softmax distribution has lower-entropy, assuming logits are IID.
- Ultimately, these large magnitude vectors will cause softmax to peak and assign very little weight to all others
- Divide by $\sqrt{D}$ to reduce effect of large magnitude vectors

Inputs:
Input vectors: **x** (shape: N x D)
Query: **h** (shape: D)

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

General attention layer



Outputs:
context vectors: **y** (shape: D)

Multiple query vectors
- each query creates a new output context vector

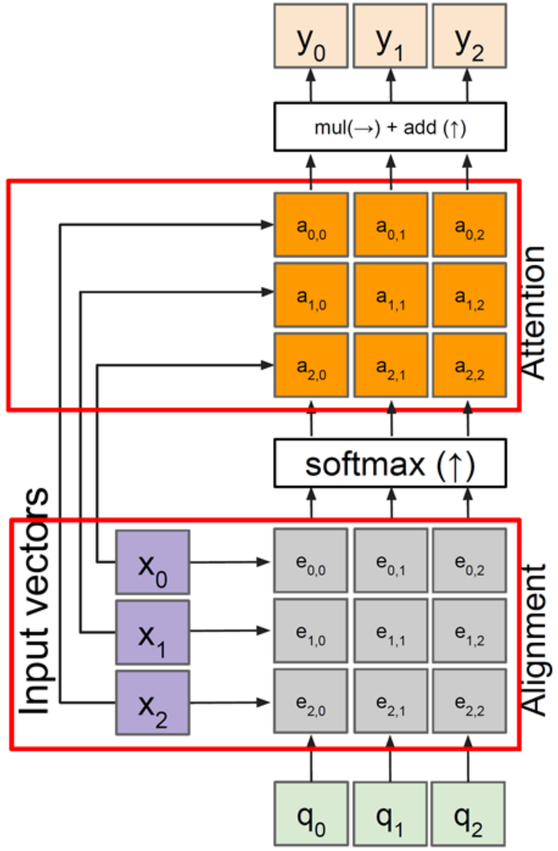
Operations:
Alignment: $e_{i,j} = q_j \cdot x_i / \sqrt{D}$
Attention: **a** = softmax(**e**)
Output: $y_j = \sum_i a_{i,j} x_i$

Multiple query vectors

Inputs:
Input vectors: **x** (shape: N x D)
Queries: **q** (shape: M x D)

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

General attention layer



Outputs:
context vectors: **y** (shape: D)

Operations:
Alignment: $e_{i,j} = q_j \cdot x_i / \sqrt{D}$
Attention: **a** = softmax(**e**)
Output: $y_j = \sum_i a_{i,j} x_i$

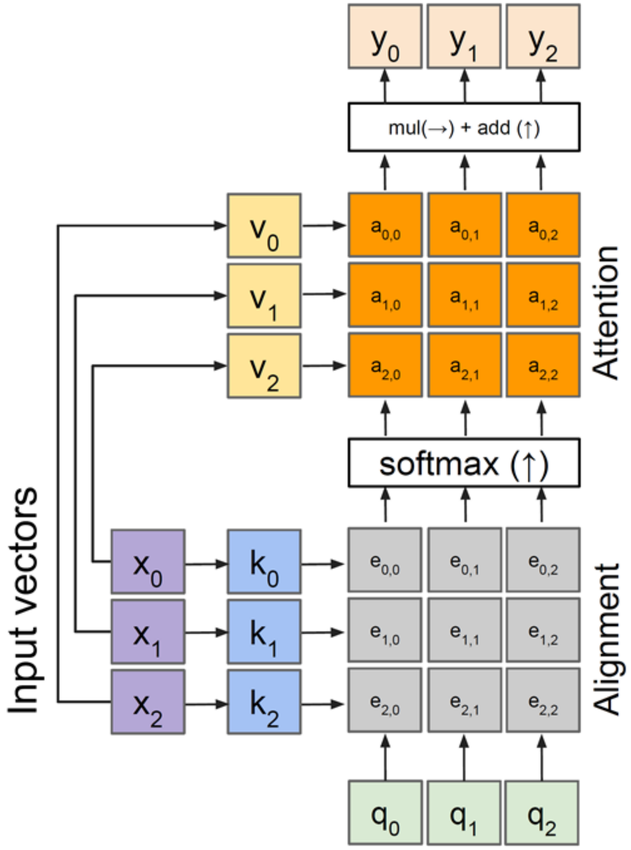
Inputs:
Input vectors: **x** (shape: N x D)
Queries: **q** (shape: M x D)

Notice that the input vectors are used for both the alignment as well as the attention calculations.

- We can add more expressivity to the layer by adding a different FC layer before each of the two steps.

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

General attention layer



Outputs:
context vectors: **y** (shape: D_v)

Operations:
Key vectors: $\mathbf{k} = \mathbf{x}W_k$
Value vectors: $\mathbf{v} = \mathbf{x}W_v$
Alignment: $e_{i,j} = q_j \cdot k_i / \sqrt{D}$
Attention: $\mathbf{a} = \text{softmax}(\mathbf{e})$
Output: $y_j = \sum_i a_{i,j} v_i$

Inputs:
Input vectors: **x** (shape: N x D)
Queries: **q** (shape: M x D_k)

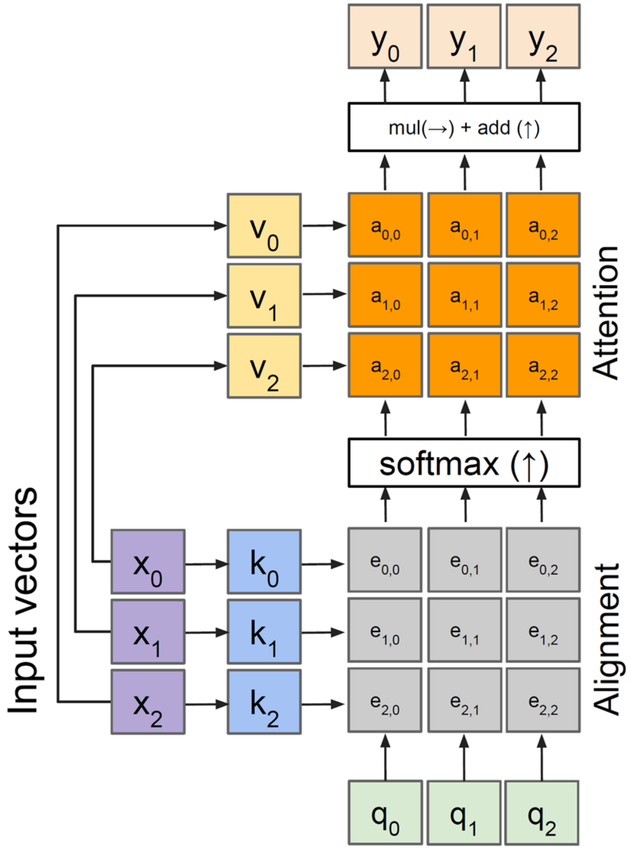
The input and output dimensions can now change depending on the key and value FC layers

Notice that the input vectors are used for both the alignment as well as the attention calculations.

- We can add more expressivity to the layer by adding a different FC layer before each of the two steps.

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

General attention layer



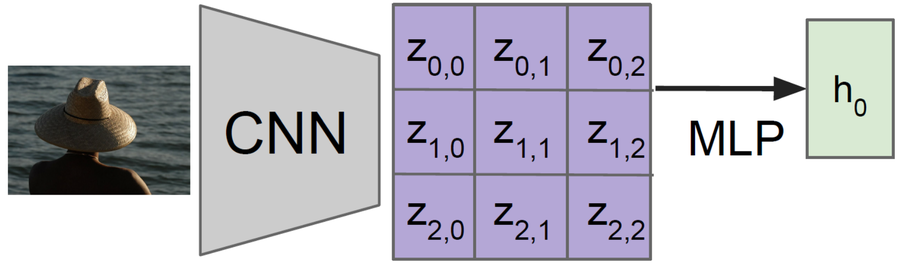
Outputs:
context vectors: **y** (shape: D_v)

Operations:
Key vectors: $\mathbf{k} = \mathbf{x}W_k$
Value vectors: $\mathbf{v} = \mathbf{x}W_v$
Alignment: $e_{i,j} = q_j \cdot k_i / \sqrt{D}$
Attention: $\mathbf{a} = \text{softmax}(\mathbf{e})$
Output: $y_j = \sum_i a_{i,j} v_i$

Inputs:
Input vectors: **x** (shape: N x D)
Queries: **q** (shape: M x D_k)

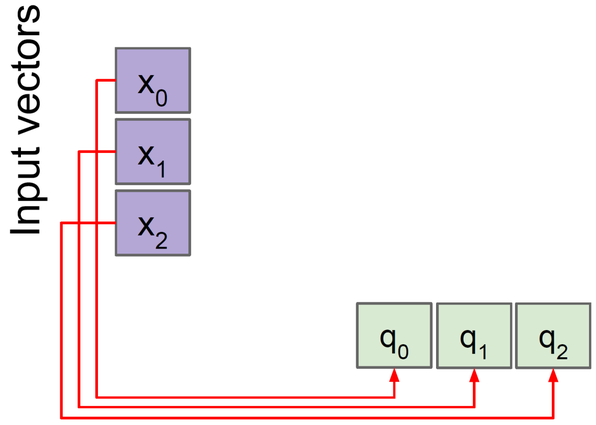
Recall that the query vector was a function of the input vectors

Encoder: $h_0 = f_W(\mathbf{z})$
where **z** is spatial CNN features
$f_W(.)$ is an MLP



This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Self attention layer



Operations:
Key vectors: $k = xW_k$
Value vectors: $v = xW_v$
Query vectors: $q = xW_q$
Alignment: $e_{i,j} = q_j \cdot k_i / \sqrt{D}$
Attention: $a = \text{softmax}(e)$
Output: $y_j = \sum_i a_{i,j} v_i$

We can calculate the query vectors from the input vectors, therefore, defining a "self-attention" layer.

Instead, query vectors are calculated using a FC layer.

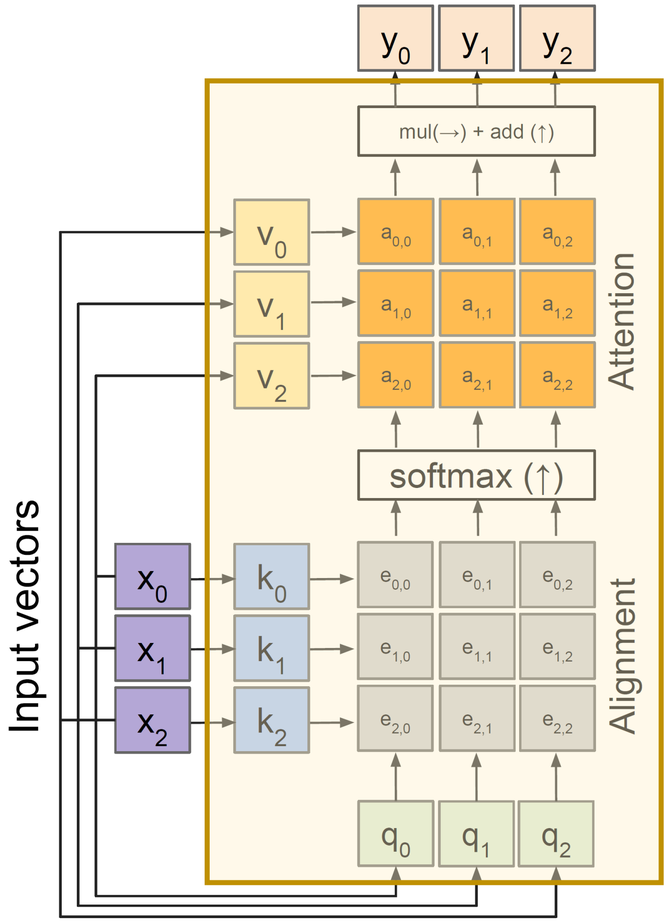
Inputs:
Input vectors: x (shape: N x D)
~~Queries: q (shape: M x D_k)~~

No input query vectors anymore

Co-financed by the European Union
Connecting Europe Facility

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

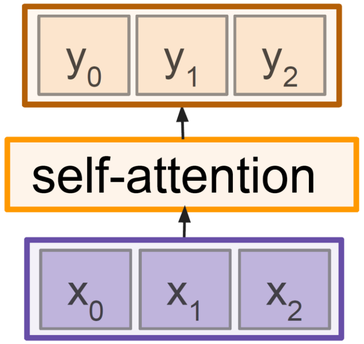
Self attention layer - *attends over sets of inputs*



Outputs:
context vectors: **y** (shape: D_v)

Operations:
Key vectors: $\mathbf{k} = \mathbf{x}\mathbf{W}_k$
Value vectors: $\mathbf{v} = \mathbf{x}\mathbf{W}_v$
Query vectors: $\mathbf{q} = \mathbf{x}\mathbf{W}_q$
Alignment: $e_{i,j} = q_j \cdot k_i / \sqrt{D}$
Attention: $\mathbf{a} = \text{softmax}(\mathbf{e})$
Output: $y_j = \sum_i a_{i,j} v_i$

Inputs:
Input vectors: **x** (shape: N x D)



This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Other attention layers

Masked self-attention layer: A masked self-attention layer is a type of self-attention layer that prevents the attention mechanism from looking ahead in the sequence during training. This is achieved by masking out the entries in the attention matrix that correspond to future elements in the sequence. The purpose of the mask is to ensure that the model can only attend to information that was previously seen, and not rely on future information during training. This is particularly useful in tasks such as language modeling, where the model needs to predict the next word in a sentence based on the preceding words.

Multi-head self attention layer: A multi-head self-attention layer is a type of self-attention layer that allows the model to attend to multiple parts of the input sequence simultaneously. This is done by splitting the input sequence into multiple "heads," and computing self-attention independently for each head. Each head learns a different representation of the input sequence, allowing the model to capture different relationships between the input elements. The output of each head is then concatenated and passed through a linear layer to produce the final output. Multi-head self-attention is particularly useful in tasks such as machine translation, where the model needs to attend to both the source and target language simultaneously

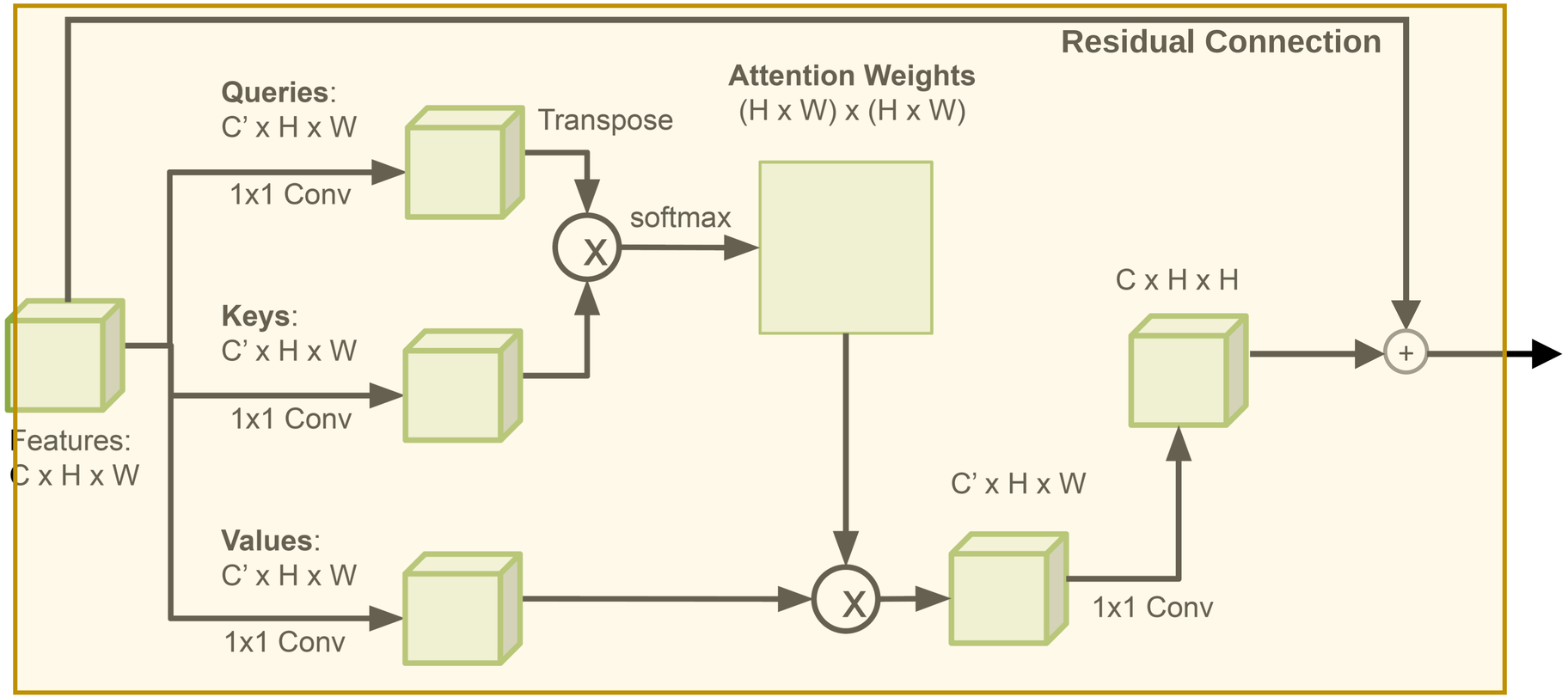
This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Example: CNN with Self-Attention

Input Image

Cat image is free to use under the Pixabay License

CNN



Self-Attention Module

Zhang et al, "Self-Attention Generative Adversarial Networks", ICML 2018

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Comparing RNNs to Transformer

RNNs

(+) LSTMs work reasonably well for long sequences.

(-) Expects an ordered sequences of inputs

(-) Sequential computation: subsequent hidden states can only be computed after the previous ones are done.

Transformer:

(+) Good at long sequences. Each attention calculation looks at all inputs.

(+) Can operate over unordered sets or ordered sequences with positional encodings.

(+) Parallel computation: All alignment and attention scores for all inputs can be done in parallel.

(-) Requires a lot of memory: N x M alignment and attention scalers need to be calculated and stored for a single self-attention head. (but GPUs are getting bigger and better)

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Transformers

Transformers are a type of neural network architecture that was introduced in the paper "Attention is All You Need" in 2017. They were designed to improve the performance of sequence-to-sequence models in natural language processing tasks such as machine translation, summarization, and question answering.

The key innovation of the transformer architecture is the self-attention mechanism, which allows the model to attend to all the input elements in a sequence simultaneously, rather than relying on recurrent or convolutional layers that process the sequence one element at a time. This allows transformers to capture long-range dependencies and better represent the relationships between different elements in the input sequence.

Transformers consist of two main components: an **encoder** and a **decoder**. The encoder takes an input sequence and generates a sequence of hidden representations, while the decoder takes the encoder output and generates a target sequence. Both the encoder and decoder are composed of multiple layers of self-attention and feed-forward neural networks.

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Transformers

During training, the model is fed pairs of input and output sequences, and the parameters of the model are updated to minimize the difference between the model's predictions and the ground truth output. The self-attention mechanism allows the model to learn to focus on different parts of the input sequence depending on the current state of the decoder, while the feed-forward layers help to transform the input representations into a form that is suitable for the task.

Transformers have achieved state-of-the-art performance on a wide range of natural language processing tasks and have become a popular choice for developing language models. They have also been adapted for other types of data, such as images and audio.

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Summary

- Adding attention to RNNs allows them to "attend" to different parts of the input at every time step
- The general attention layer is a new type of layer that can be used to design new neural network architectures
- Transformers are a type of layer that uses self-attention and layer norm.
  - It is highly scalable and highly parallelizable
  - Faster training, larger models, better performance across vision and language tasks
  - They are quickly replacing RNNs, LSTMs, and may(?) even replace convolutions.

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Thank you!

See you next week

Co-financed by the European Union
Connecting Europe Facility

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423