



MIDDLE EAST TECHNICAL UNIVERSITY

# DI504

# Foundations of Deep Learning

## Sequence Models

# Welcome again!

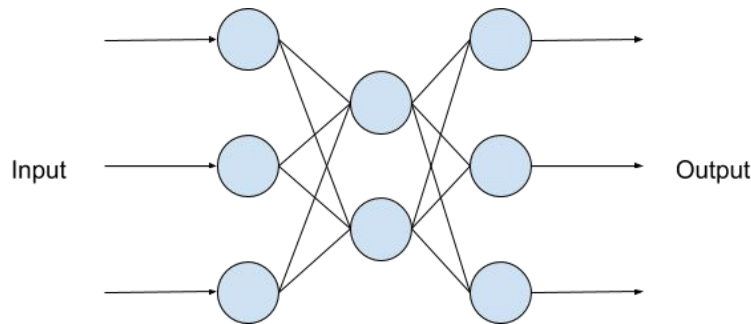
This is DI504, Foundations of Deep Learning

- We previously talked about feature spaces and score functions, which are basically the fundamentals concept of machine learning.
- Then we jumped into loss function. We learned what ANNs are and how to optimized them. And then to CNNs
- Now, we will talk about sequences. Varying length input!

# METU What is a Sequence?

(first, what is not a Sequence?)

- Consider the problem of predicting the health risk of a person based on multiple health parameters and we have decided to model the true relationship between the input and output using (e.g) a Feed-forward neural network (FNN).
- In FNNs the output of one data point is completely independent of the previous input. In other words, the health risk of the second person is not dependent on the health risk of the first person.

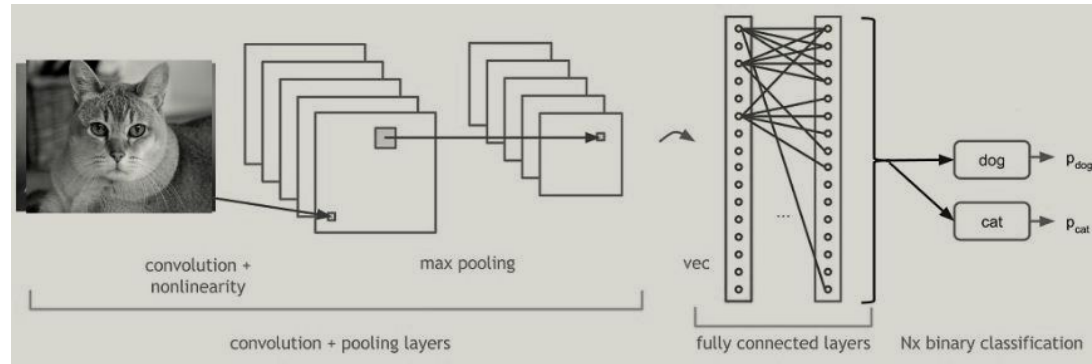


ANNs are a type of FNNs

# What is a Sequence?

(first, what is not a Sequence?)

- Similarly, in the case of Convolution Neural Networks (CNN), the output from the (e.g.) softmax layer in the context of image classification is entirely independent of the previous input image.



# Is human intelligence a type of an FNN?

- Ask someone «is this a cat image?» they would answer easily. Then we, as humans, must have FNN capabilities.
- Ask them the same image repeatedly. How will they answer?



# Is human intelligence a type of an FNN?

- Ask someone «is this a cat image?» they would answer easily. Then we, as humans, must have FNN capabilities.
- Ask them the same image repeatedly. How will they answer?
  1. «It is a cat image.» (similar to an FNN)
  2. «Again, cat image» (not quite like an FNN)
  3. «...» (definitely not an FNN, from this point on...)
  4. «I am a dog person.»
  5. «Is this one of your jokes?»
  6. «It isn't funny.»



# Is human intelligence a type of an FNN?

- Ask someone «is this a cat image?» they would answer easily. Then we, as humans, must have FNN capabilities.
- Ask them the same image repeatedly. How will they answer?
  1. «It is a cat image.» (similar to an FNN)
  2. «Again, cat image» (not quite like an FNN)
  3. «...» (definitely not an FNN, from this point on...)
  4. «I am a dog person.»
  5. «Is this one of your jokes?»
  6. «It isn't funny.»



# Is human intelligence a type of an FNN?

- Ask someone «is this a cat image?» they would answer easily. Then we, as humans, must have FNN capabilities.
- Ask them the same image repeatedly. How will they answer?
  1. «It is a cat image.» (similar to an FNN)
  2. «Again, cat image» (not quite like an FNN)
  3. «...» (definitely not an FNN, from this point on...)
  4. «I am a dog person.»
  5. «Is this one of your jokes?»
  6. «It isn't funny.»





# Is human intelligence a type of an FNN?

- Ask someone «is this a cat image?» they would answer easily. Then we, as humans, must have FNN capabilities.
- Ask them the same image repeatedly. How will they answer?
  1. «It is a cat image.» (similar to an FNN)
  2. «Again, cat image» (not quite like an FNN)
  3. «...» (**definitely not an FNN, from this point on...**)
  4. «I am a dog person.»
  5. «Is this one of your jokes?»
  6. «It isn't funny.»



# Is human intelligence a type of an FNN?

- Ask someone «is this a cat image?» they would answer easily. Then we, as humans, must have FNN capabilities.
- Ask them the same image repeatedly. How will they answer?
  1. «It is a cat image.» (similar to an FNN)
  2. «Again, cat image» (not quite like an FNN)
  3. «...» (**definitely not an FNN, from this point on...**)
  4. «I am a dog person.»
  5. «Is this one of your jokes?»
  6. «It isn't funny.»



# Is human intelligence a type of an FNN?

- Ask someone «is this a cat image?» they would answer easily. Then we, as humans, must have FNN capabilities.
- Ask them the same image repeatedly. How will they answer?
  1. «It is a cat image.» (similar to an FNN)
  2. «Again, cat image» (not quite like an FNN)
  3. «...» (**definitely not an FNN, from this point on...**)
  4. «I am a dog person.»
  5. «Is this one of your jokes?»
  6. «It isn't funny.»



# What is a Sequence?

(first, what is not a Sequence?)

- So, in humans (unlike FNNs), the output of one data point is **NOT independent** of the previous input.
- A **sequence** can be thought of as a list of elements with a **particular** order.
- Sequence Modeling is the task of modelling what word/letter/element comes next.
-

# What is a Sequence?

(first, what is not a Sequence?)

- So, in humans (unlike FNNs), the output of one data point is **NOT independent** of the previous input.
- A **sequence** can be thought of as a list of elements with a **particular** order.
- Sequence Modeling is the task of modelling what word/letter/element comes next.
- Unlike the FNN and CNN, in sequence modeling, the current output is dependent on the previous input and the length of the input is not fixed.



# Sequence Modelling

- Sequence modeling, put simply, is the process of modelling/predicting/generating a sequence of values by analyzing a series of previous input values.
- Unlike the FNN and CNN, in sequence modeling,
  - the current output is dependent on the previous input
  - and the length of the input is not fixed.



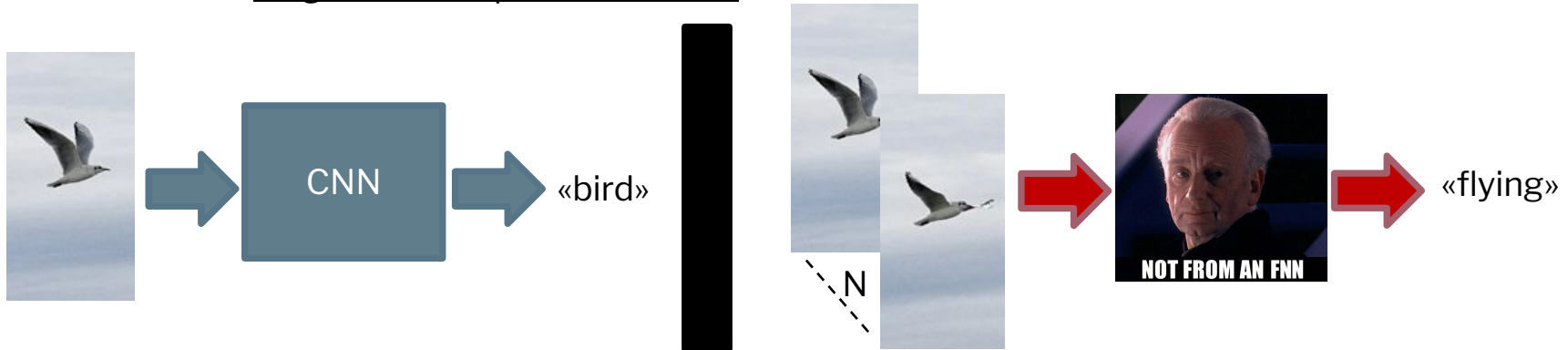
# Sequence Modelling

- Sequence modeling, put simply, is the process of modelling/predicting/generating a sequence of values by analyzing a series of previous input values.
- Unlike the FNN and CNN, in sequence modeling,
  - the current output is dependent on the previous input
  - and the length of the input is not fixed.



# Sequence Modelling

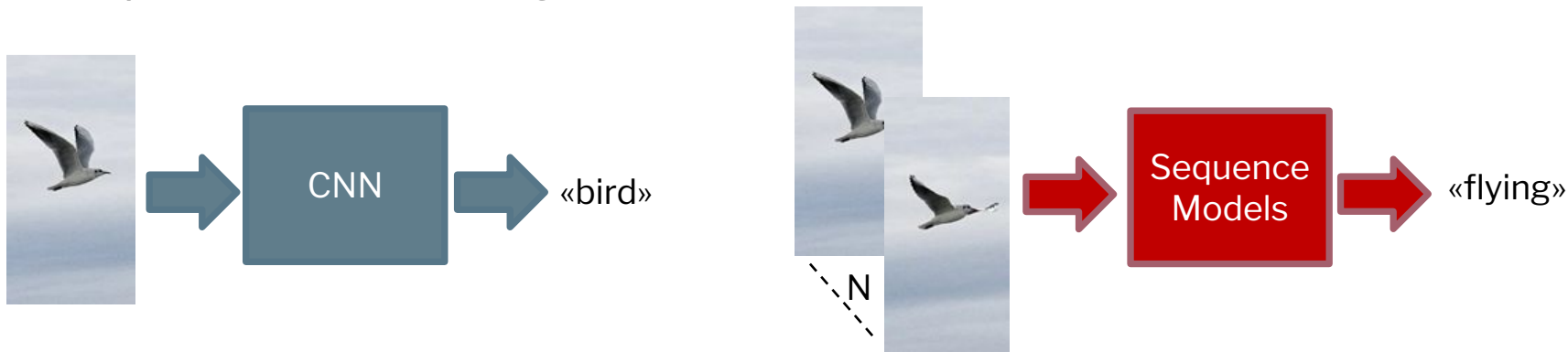
- Sequence modeling, put simply, is the process of modelling/predicting/generating a sequence of values by analyzing a series of previous input values.
- Unlike the FNN and CNN, in sequence modeling,
  - the current output is dependent on the previous input
  - and the length of the input is not fixed.





# What is it not an ANN?

- Sequence models in general are not ANNs (or vice versa).
- The first answer we will give to that question will be RNNs.
  - Recurrent Neural Networks (RNNs)
- However the answer is not limited to RNNs (which will be explore in the following weeks).



# What is a Sequence?

- Is an image a sequence?
  - A sequence of pixels maybe?
  - **NO, an image is NOT a sequence!**
    - *No history or dependence between pixels!*
      - That's why CNNs work good with them (i.e. small networks in a small pixel neighbourhood.)
    - *The image size (of a given «cat image») is (can be) fixed (i.e.  $M \times N$  image)*



History  
Varying Length Data

# What is a Sequence?

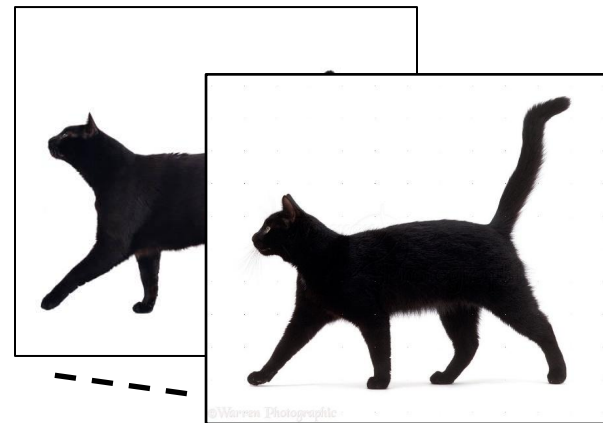
- Is an image a sequence?
  - A sequence of pixels maybe?
  - **NO, an image is NOT a sequence!**
    - *No history or dependence between pixels!*
      - That's why CNNs work good with them (i.e. small networks in a small pixel neighbourhood.)
    - *The image size (of a given «cat image») is (can be) fixed (i.e.  $M \times N$  image)*
  - Or is it ?  
*SPOILER ---- surprisingly they can also be treated as a sequence. ---- SPOILER*



History  
Varying Length Data

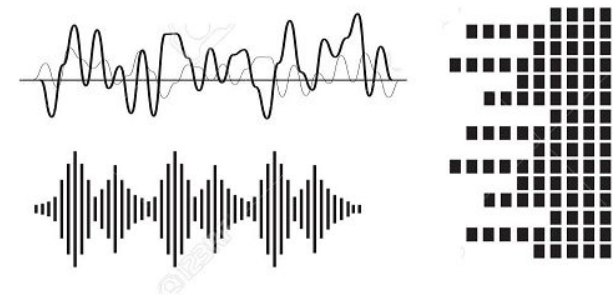
# What is a Sequence?

- Is video a sequence?
  - A sequence of images maybe?
  - **YES, video is a sequence of images!**
    - *There is history between frames (consequent images)*
    - *The video size (of a given «action») can be of different lengths*
      - 1 second catwalk (30 frames in a 30fps video)
      - 2 seconds catwalk (60 frames in a 30fps video)
      - Both are «walking cat» videos.
      - However we can always get a fixed-length piece of a video to be applied to CNNs (this is a common practice)



History  
Varying Length Data

# What is a Sequence?



- Is sound a sequence?
  - A sequence of signals maybe?
  - **YES, sound can be modelled as a sequence of sampled signals!**
    - *There is history between each sampled signal (in time dimension)*
    - *The size (of a given «sound») can be of different lengths*
      - 1 second of speech (44100 samples in a 44100 Hz sound)
      - 2 seconds of speech (88200 samples in a 44100 Hz sound)
      - Both are «human speech» sounds (categorically).
      - However we can always get a fixed-length piece of a sound to be applied to CNNs (in some occasion)

History

Varying Length Data

# What is a Sequence?

<sup>1</sup>This <sup>2</sup>is <sup>3</sup>a <sup>4</sup>meaningful <sup>5</sup>sentence.



History  
Varying Length Data

- Is text a sequence?
  - A sequence of words maybe?
  - **YES, meaningful text is purely a sequence!**
    - *There is history between each word.*
    - *The size (of a given «sentence/text») can be of different lengths*
      - 10 words sentence
      - 20 words sentence
      - Both are meaningful text

# What is a Sequence?

- Is an image a sequence?
  - A sequence of pixels maybe?

- **We said NO!**

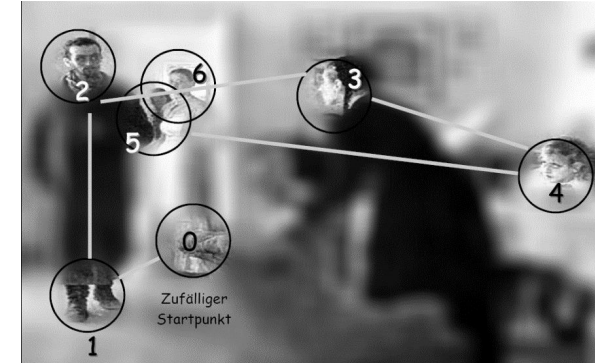
- *Well we lied! We lied.*
- *(We lied ask us why we lied!?)*
- *Because we wanted to give you a simple example of what is a sequence and what is not.*
- *However, the way humans understand images is not how FNNs do it. We treat images also as sequences.*
- *How? This is something we will learn in this course*
- *Let's make a brief introduction. Just to taste it.*



History  
Varying Length Data

# Image as a Sequence?

- Attention!
  - *(not to get your attention, but it is the title actually)*
  - The general idea is to take inspiration from how human eye works, i.e. retina,
  - Our eyes can capture a very wide view of the world, however, we tend to “glance” the overview and only pay «**attention**» to a particular region of the view, while the other part is kind of “blurred out”.
  - So we do not just scan the image, (like CNNs do)
    - First we make a very fast scan of the scene (like CNNs do), and detect particular areas in the image.
    - *Then appyle sequence of glances/jumps like sequence models do.*
    - *So we are a CNN+Sequence Model seer*
      - *Like a the CNN+RNN, you will learn in this course.*



History

Varying Length Data



# What to do with the Sequences?

- ✓ Classify a sequence?
- ✓ Predict a sequence?
- ✓ Transform a sequence?
- ✓ Compare two sequences?
- ✗ Generate a sequence?
  - But hopefully we will make a brief introduction in the final weeks.

$$\mathbf{y}(\mathbf{t}) = \mathbf{f}(\mathbf{x}(\mathbf{t}))$$

# For example: Sequence Prediction

- Sequence prediction consists of predicting the next sequence entry based on the previously observed sequence history.
- These sequence could be a number, an alphabet, a word, an event, a video, or an object like a webpage or product. For example:
  - A sequence of words or characters in a text
  - A sequence of products bought by a customer
  - A sequence of events observed on logs
  - A sequence of frames in a video clip

$$\mathbf{y}(\mathbf{t}) = \mathbf{f}(\mathbf{x}(\mathbf{t}))$$

# Sequence Prediction

- Sequence prediction is a common problem which finds real-life applications in various industries.
- (For the sake of simplicity), Let's introduce three principle types of sequence prediction problems:
  - Predicting the next value
  - Predicting a class label
  - Predicting a sequence

$$\mathbf{x}(t_{i+1}) = \mathbf{f}(\mathbf{x}(t_{i-0}))$$

# Sequence Prediction (next value)

- Being able to guess the next element of a sequence is an important question in many applications.
- A sequence prediction model learns to identify the pattern in the sequential input data and predict the next value.

What Number Comes Next In This  
Number Sequence?

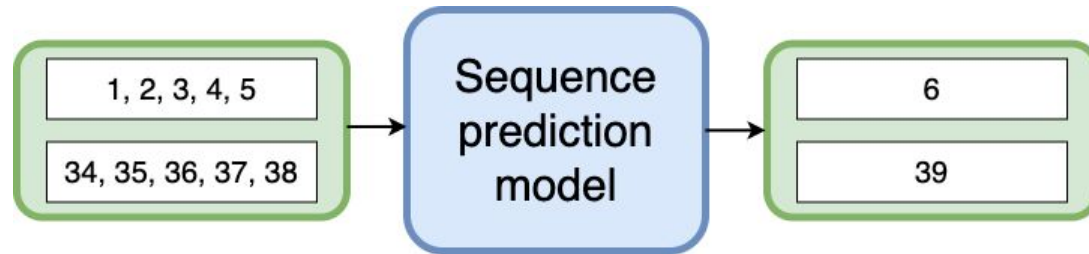
**1, 1, 2, 3, 5, 8, 13, 21, 34, 55...**

se)

$$\mathbf{x}(t_{i+1}) = \mathbf{f}(\mathbf{x}(t_{i-0}))$$

# Sequence Prediction (next value)

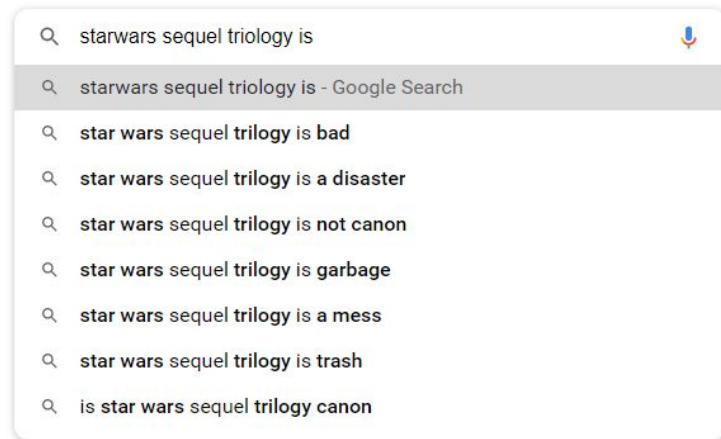
- Being able to guess the next element of a sequence is an important question in many applications.
- A sequence prediction model learns to identify the pattern in the sequential input data and predict the next value.



$$\mathbf{x}(t_{i+1}) = \mathbf{f}(\mathbf{x}(t_{i-0}))$$

# Sequence Prediction (next value)

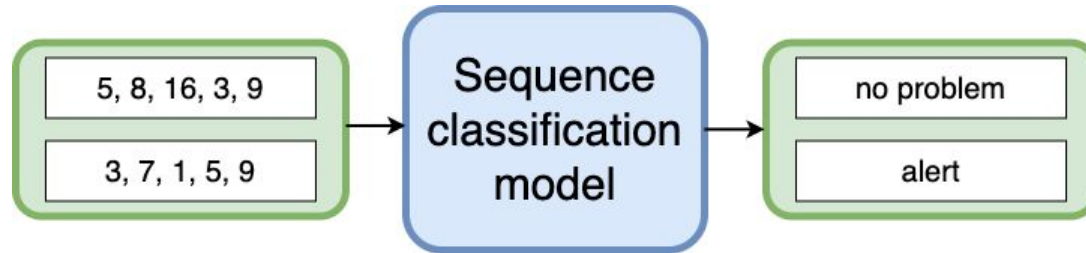
- Google's «Auto-Completion» is maybe the best real
- Next «word» (or words) is (are) predicted, according to your previous and other people's previous search
- There are countless real world examples:



$$k_j = \mathbf{f}(\mathbf{x}(t_{i-0}))$$

# Sequence Classification (class label)


- Sequence classification uses labeled datasets with some sequence inputs and class labels as outputs, to train a classification model which can be used to predict the class label of an unseen sequence.




$$k_j = \mathbf{f}(\mathbf{x}(t_{i-0}))$$

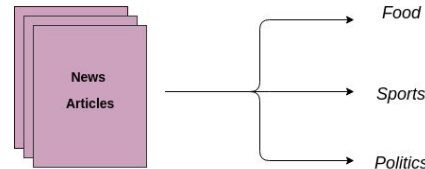
# Sequence Classification (class label)

- Some examples of sequence classification applications:
  - Text categorization.** Assigning labels to documents written in a natural language has numerous real-world applications including

■ sentiment classification "I love this movie.  
I've seen it many times  
and it's still awesome." → 

"This movie is bad.  
I don't like it it all.  
It's terrible." → 

■ topic categorization





$$k_j = \mathbf{f}(\mathbf{x}(t_{i-0}))$$

# Sequence Classification (class label)

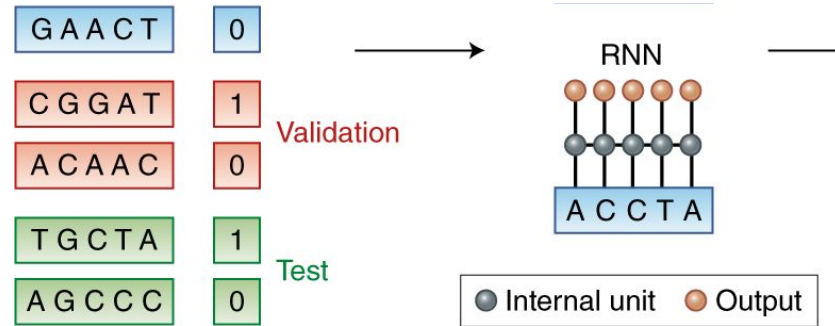
- Some examples of sequence classification applications:
  - **Anomaly detection.** Researchers explore detecting abnormal behaviors in many different time-series datasets,
    - electrocardiograms,
    - power demand, and engine sensors
    - etc.



$$k_j = \mathbf{f}(\mathbf{x}(t_{i-0}))$$

# Sequence Classification (class label)

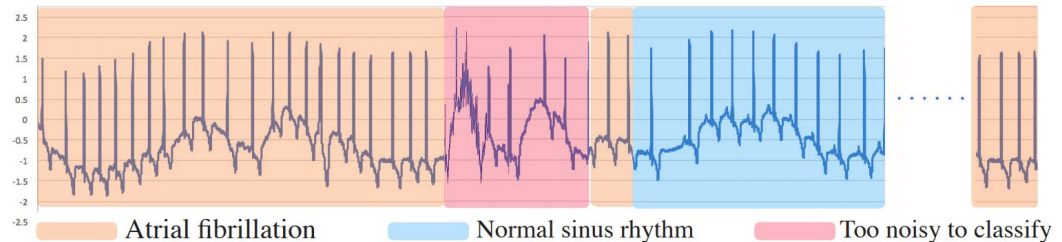
- Some examples of sequence classification applications:
  - Genomic research.** Researchers have been classifying protein sequences into categories. This work could be potentially useful for the discovery of a wide range of protein functions.



$$k_j = \mathbf{f}(\mathbf{x}(t_{i-0}))$$

# Sequence Classification (class label)

- Some examples of sequence classification applications:
  - **Health-informatics.**
    - Researchers use LSTM to classify electrocardiogram (ECG) signals into five different classes to identify the condition of a patient's heart. This allows end-to-end learning, extracting features related to ECG signals without any expert intervention.



$$\mathbf{y}(t_{j-0}) = \mathbf{f}(\mathbf{x}(t_{i-0}))$$

# Sequence Transformation (Seq2Seq)

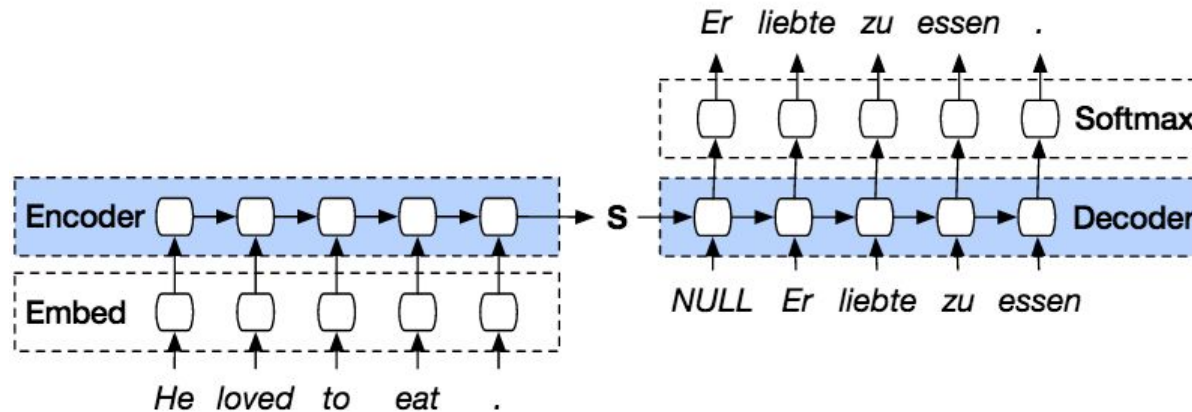
- Sequence-to-sequence learning is about training models to convert an input sequence and output another sequence.
- Like all supervised learning, Seq2Seq models are trained with a dataset of pairs, but the input sequences and output sequences can have different lengths.



$$\mathbf{y}(t_{j-0}) = \mathbf{f}(\mathbf{x}(t_{i-0}))$$

# Sequence Transformation (Seq2Seq)

- The most famous example is language translation:



# Vanilla RNN vs Standard FNN

- First, the standard Feed-forward Neural Network (FNN) or also called:
  - Artificial Neural Network (ANN)
  - Fully-connected (dense) Layers

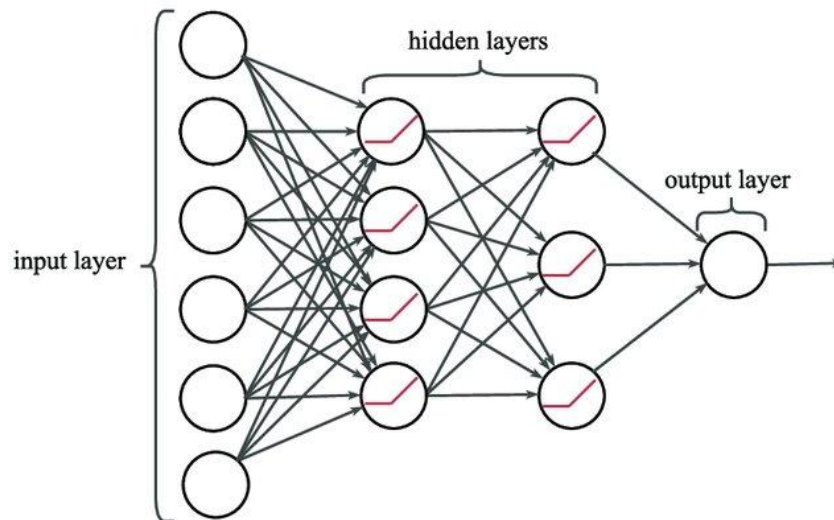
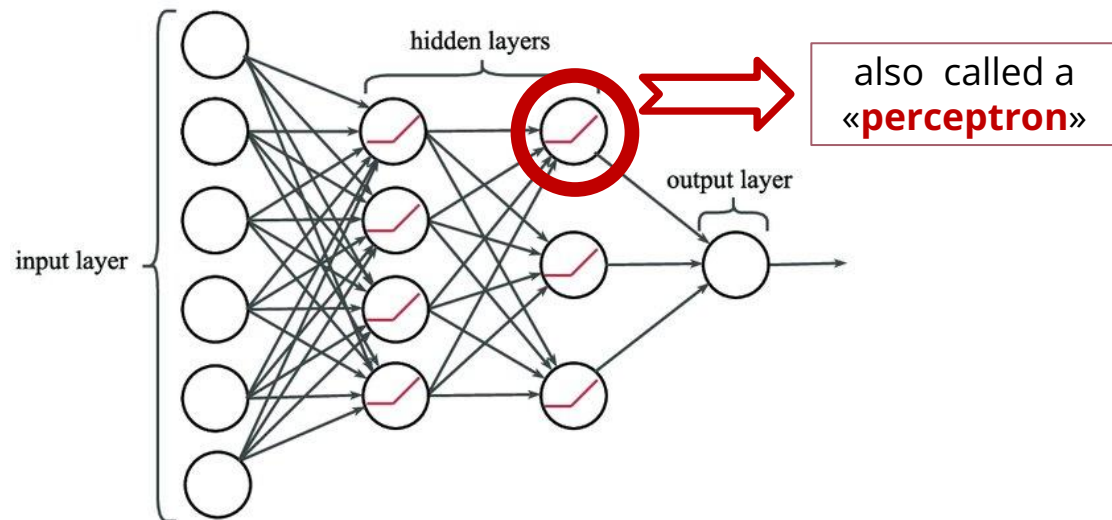


Figure from Cenek et al.,  
«[10.3390/app8050749](https://arxiv.org/abs/10.3390/app8050749)»



# Vanilla RNN vs Standard FNN

- First, the standard Feed-forward Neural Network (FNN) or also called:
  - Artificial Neural Network (ANN)
  - Fully-connected (dense) Layers



# Vanilla RNN vs Standard FNN

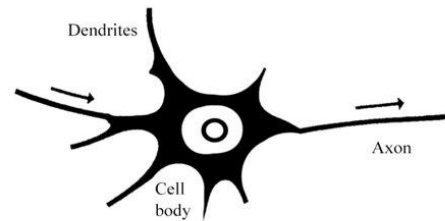
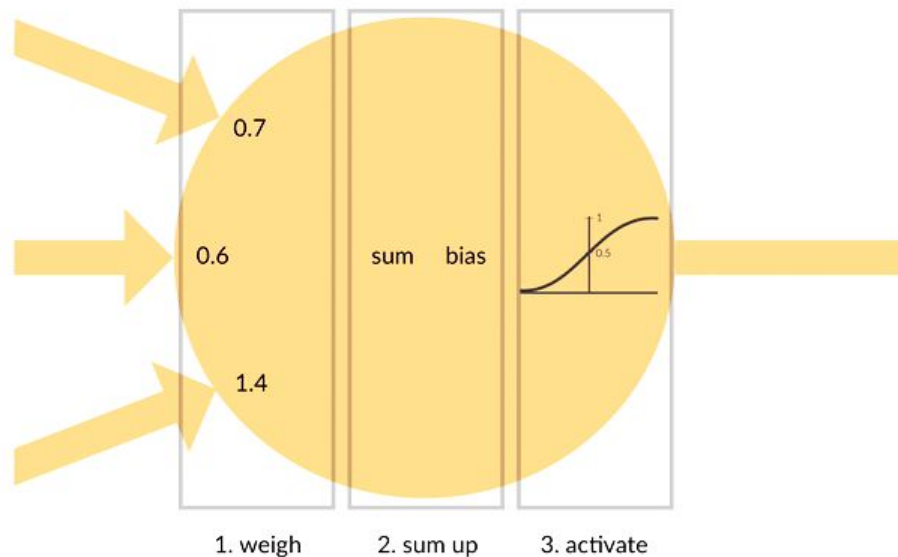


Figure from M.A. Arbib, «ISBN: 9780262011488»

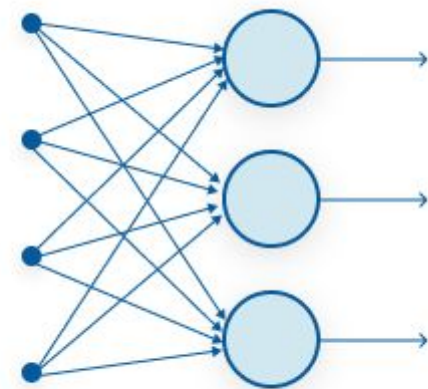
- Inside a perceptron (node), what you basically do is:
  - weigh (multiply)
  - accumulate & bias (sum)
  - activate (non-linear function)





# Vanilla RNN vs Standard FNN

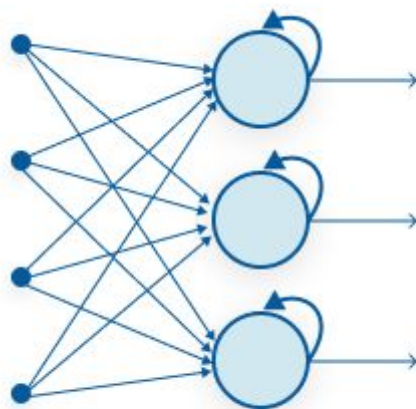
- So, FNN is simply a network of weighted inputs summed and activated by so-called «nodes»
- These set of nodes are referred to as a «layer»
- Layer are connected to construct deep networks. The output of a layer (also called an activation) is the input to the next «layer»



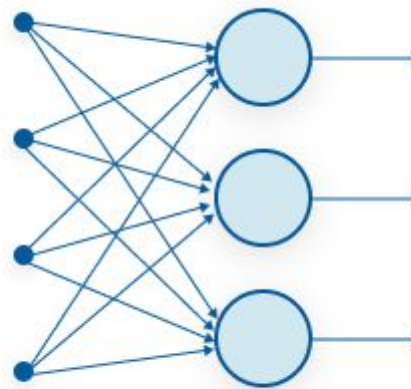
Feed-Forward Neural Network

# Vanilla RNN vs Standard FNN

- A vanilla RNN is a very similar architecture, but with an additional feedback connection



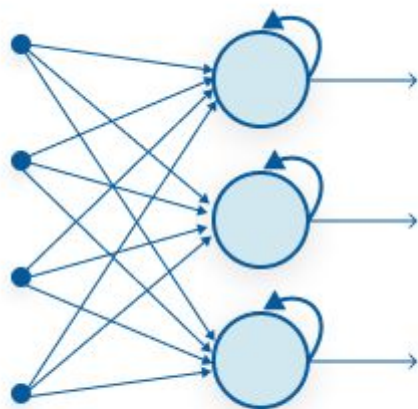
Recurrent Neural Network



Feed-Forward Neural Network

# Vanilla RNN vs Standard FNN

- A vanilla RNN is a very similar architecture, but with an additional feedback connection

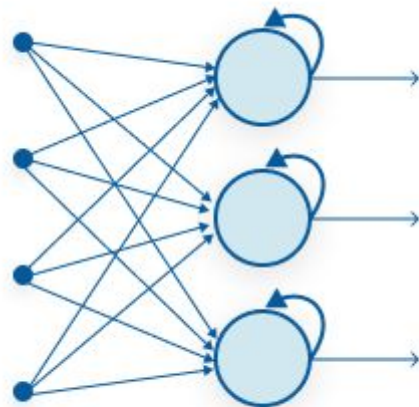


Recurrent Neural Network

- By «feedback», we mean a connection from the output to the perceptron's own input.
- What does this «mathematically» mean?

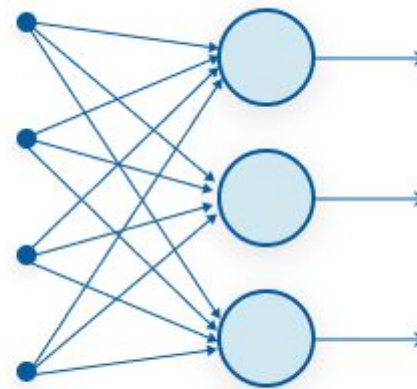
# Vanilla RNN vs Standard FNN

- Let us begin with the mathematics of FNN:



Recurrent Neural Network

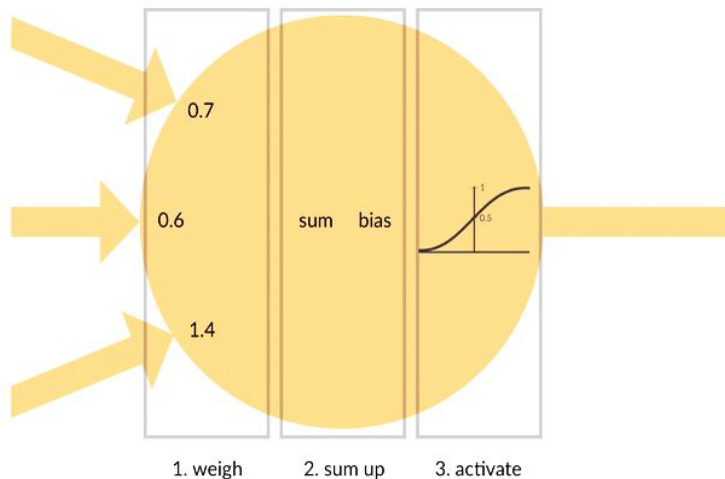
VS



Feed-Forward Neural Network

# Vanilla RNN vs Standard FNN

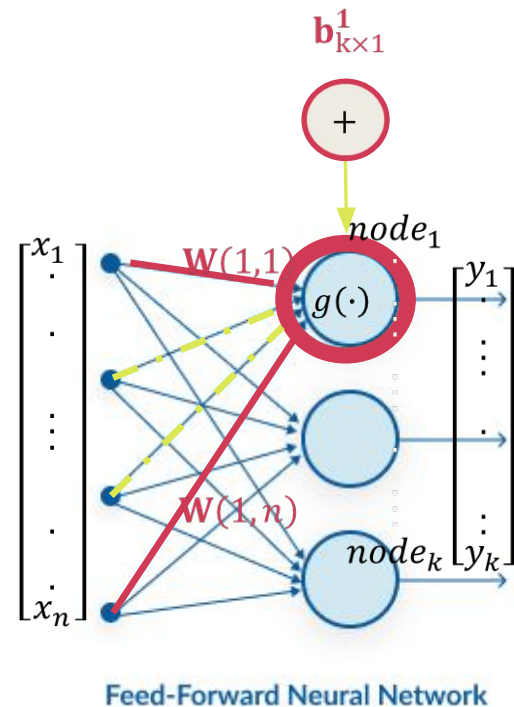
- Let us begin with the mathematics of FNN:



$$\mathbf{y}_{k \times 1} = g(\mathbf{W}_{k \times n} \cdot \mathbf{x}_{n \times 1} + \mathbf{b}_{k \times 1})$$

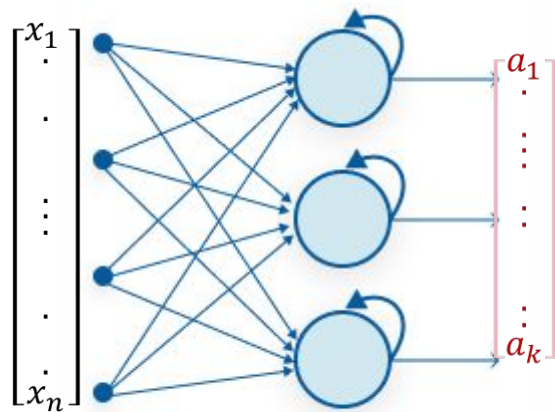
$$y(i) = g(\mathbf{w}_{1 \times n}^i \cdot \mathbf{x}_{n \times 1} + b_1)$$

$g(\cdot)$  being any activation function



# Vanilla RNN vs Standard FNN

■ Now, what is different in RNNs?

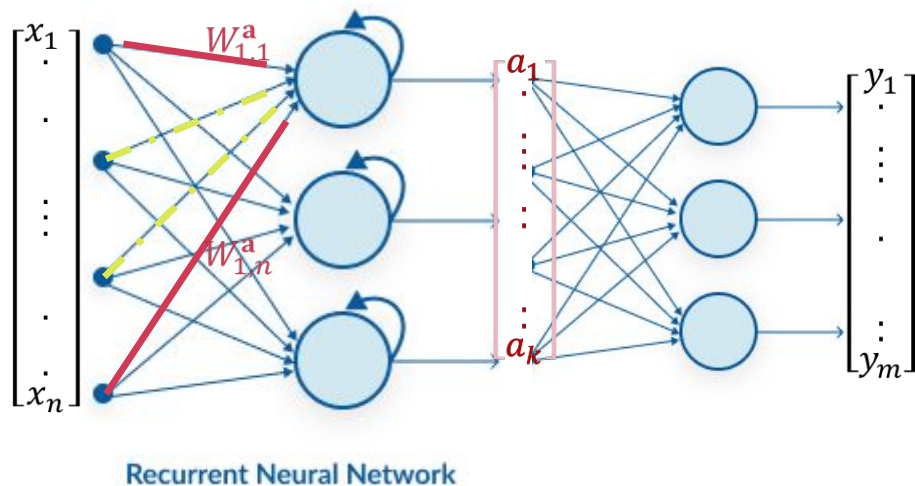


- In addition to FNN's connections, there is also a feedback (the red arc). How to model it?
- It is slightly different. Actually there's an additional set of weights (hence connections), in the conventionally accepted vanilla RNN.

# Vanilla RNN vs Standard FNN

- Now, what is different in RNNs?

This is the general model of a so-called “Vanilla RNN”



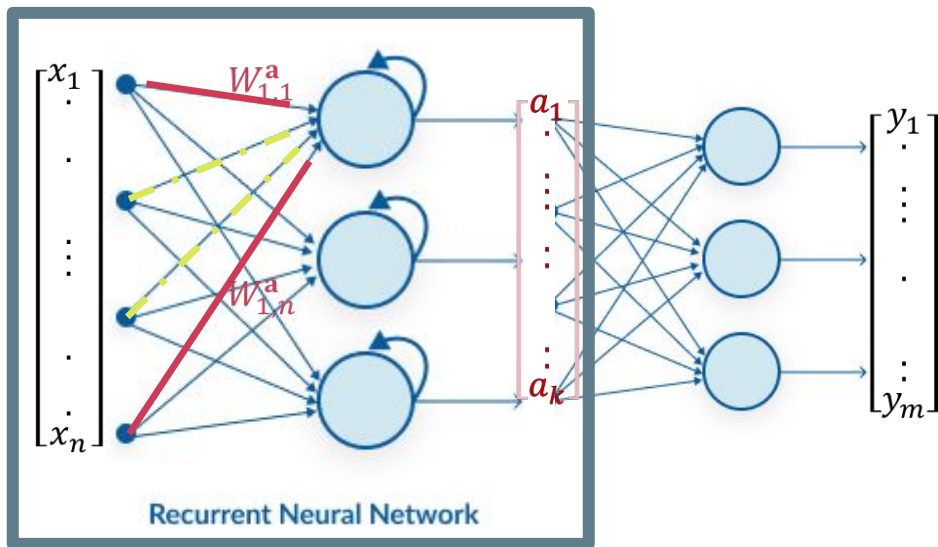
$$\mathbf{a}_{k \times 1}^{next} = g \left( \mathbf{W}_{k \times (k+n)}^a \cdot \begin{bmatrix} \mathbf{x} \\ \mathbf{a}_{k \times 1}^{prev} \end{bmatrix} \right)$$

$$\mathbf{y}_{m \times 1} = g(\mathbf{W}_{m \times k}^y \cdot [\mathbf{a}_{k \times 1}^{next}])$$

# Vanilla RNN vs Standard FNN

- Now, what is different in RNNs?

This is the general model of a so-called “Vanilla RNN”



Core RNN:

$$\mathbf{a}_{k \times 1}^{next} = g \left( \mathbf{W}_{k \times (k+n)}^a \cdot \begin{bmatrix} \mathbf{x} \\ \mathbf{a}_{k \times 1}^{prev} \end{bmatrix} \right)$$

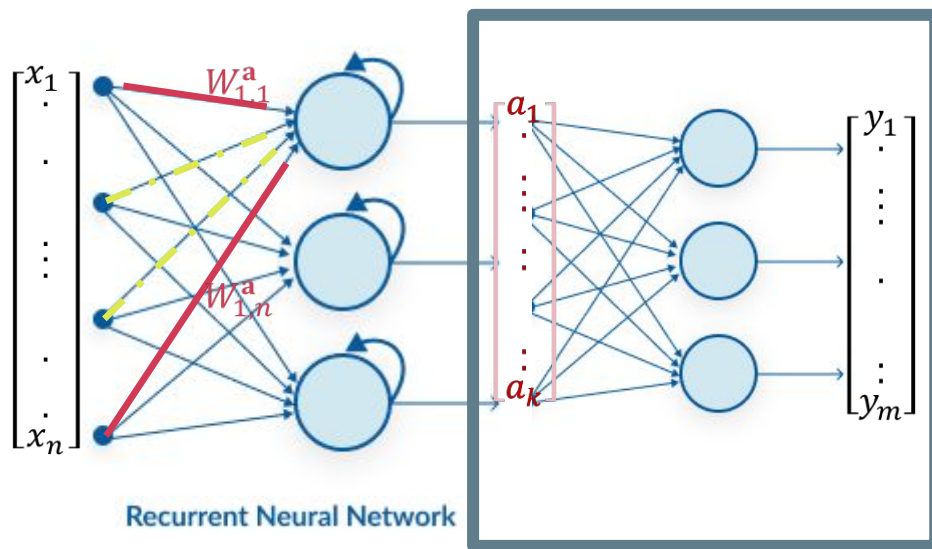
$$\mathbf{y}_{m \times 1} = g(\mathbf{W}_{m \times k}^y \cdot [\mathbf{a}_{k \times 1}^{next}])$$



# Vanilla RNN vs Standard FNN

- Now, what is different in RNNs?

This is the general model of a so-called “Vanilla RNN”



$$\mathbf{a}_{k \times 1}^{next} = g \left( \mathbf{W}_{k \times (k+n)}^a \cdot \begin{bmatrix} \mathbf{x} \\ \mathbf{a}_{k \times 1}^{prev} \end{bmatrix} \right)$$

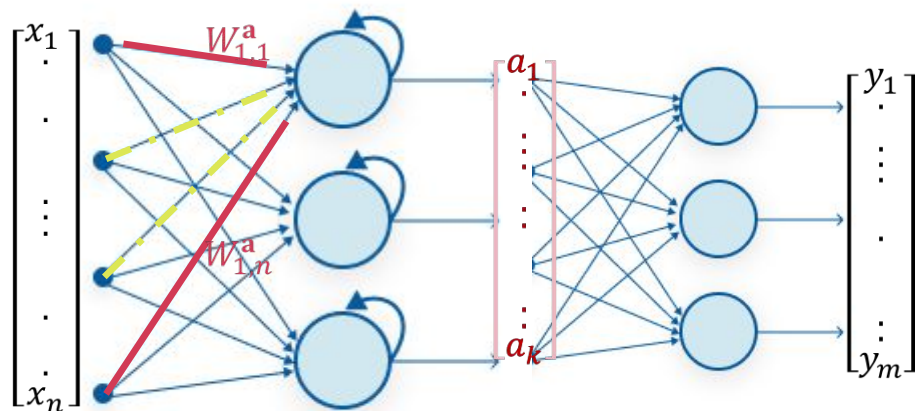
The Output:

$$\mathbf{y}_{m \times 1} = g(\mathbf{W}_{m \times k}^y \cdot [\mathbf{a}_{k \times 1}^{next}])$$

# Vanilla RNN vs Standard FNN

- Now, what is different in RNNs?

This is the general model of a so-called “Vanilla RNN”



Recurrent Neural Network

$$\mathbf{a}_{k \times 1}^{next} = g \left( \mathbf{W}_{k \times (k+n)}^a \cdot \begin{bmatrix} \mathbf{x}_{n \times 1}^{prev} \\ \mathbf{a}_{k \times 1}^{prev} \end{bmatrix} + \mathbf{b}_{k \times 1}^a \right)$$

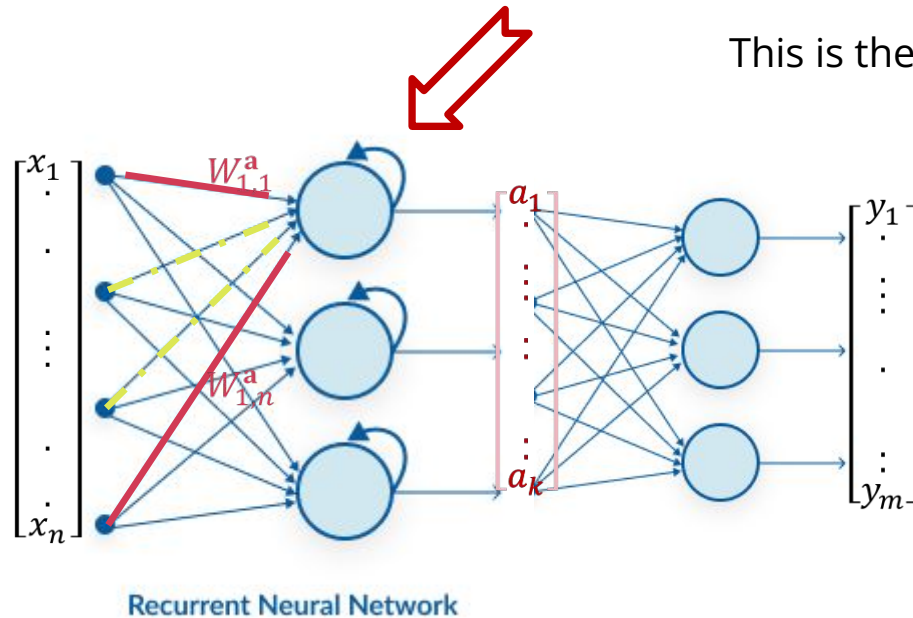
$$\mathbf{W}_{k \times (k+n)}^a = [\mathbf{W}_{k \times n}^{ax} \quad \mathbf{W}_{k \times k}^{aa}]$$

$$\mathbf{a}_{k \times 1}^{next} = g(\mathbf{W}_{k \times n}^{ax} \cdot \mathbf{x}_{n \times 1}^{prev} + \mathbf{W}_{k \times k}^{aa} \cdot \mathbf{a}_{k \times 1}^{prev} + \mathbf{b}_{k \times 1}^a)$$

$$\mathbf{y}_{m \times 1} = g(\mathbf{W}_{m \times k}^y \cdot [\mathbf{a}_{k \times 1}^{next}] + \mathbf{b}_{m \times 1}^y)$$

# Vanilla RNN vs Standard FNN

- Now, what is different in RNNs?



This is the general model of a so-called "Vanilla RNN"

$$\mathbf{a}_{k \times 1}^{next} = g \left( \mathbf{W}_{k \times (k+n)}^a \cdot \begin{bmatrix} \mathbf{x}_{n \times 1}^{prev} \\ \mathbf{a}_{k \times 1}^{prev} \end{bmatrix} + \mathbf{b}_{k \times 1}^a \right)$$

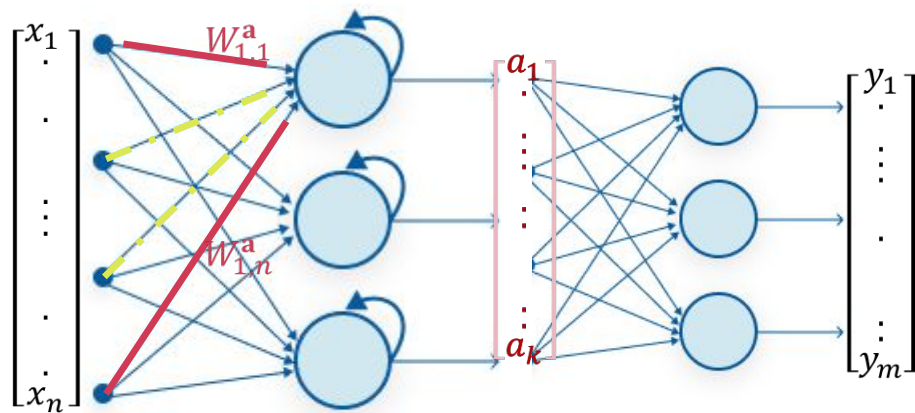
$$\mathbf{W}_{k \times (k+n)}^a = [\mathbf{W}_{k \times n}^{ax} \quad \mathbf{W}_{k \times k}^{aa}]$$

$$\mathbf{a}_{k \times 1}^{next} = g(\mathbf{W}_{k \times n}^{ax} \cdot \mathbf{x}_{n \times 1}^{prev} + \mathbf{W}_{k \times k}^{aa} \cdot \mathbf{a}_{k \times 1}^{prev} + \mathbf{b}_{k \times 1}^a)$$

$$\mathbf{y}_{m \times 1} = g(\mathbf{W}_{m \times k}^y \cdot [\mathbf{a}_{k \times 1}^{next}] + \mathbf{b}_{m \times 1}^y)$$

# Vanilla RNN vs Standard FNN

- This is a confusing diagram, which (almost) shows every connection (hence multiplication) that exists in a vanilla RNN.



Recurrent Neural Network

$$\mathbf{a}_{k \times 1}^{next} = g \left( \mathbf{W}_{k \times (k+n)}^a \cdot \begin{bmatrix} \mathbf{x}_{n \times 1}^{prev} \\ \mathbf{a}_{k \times 1}^{prev} \end{bmatrix} + \mathbf{b}_{k \times 1}^a \right)$$

$$\mathbf{W}_{k \times (k+n)}^a = [\mathbf{W}_{k \times n}^{ax} \quad \mathbf{W}_{k \times k}^{aa}]$$

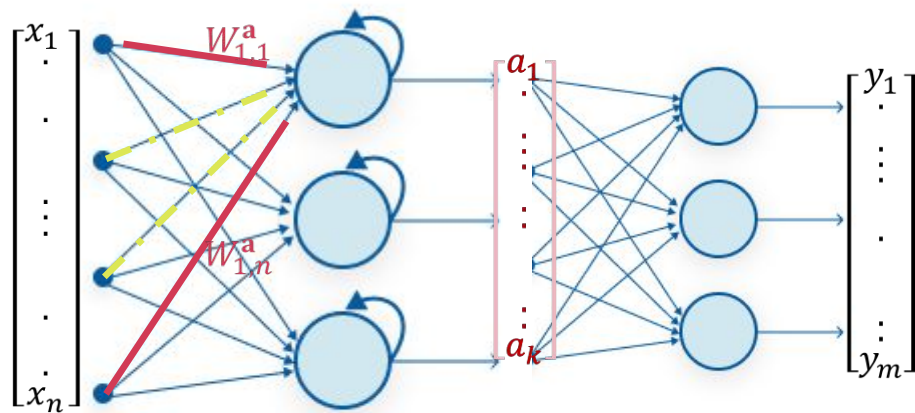
$$\mathbf{a}_{k \times 1}^{next} = g(\mathbf{W}_{k \times n}^{ax} \cdot \mathbf{x}_{n \times 1}^{prev} + \mathbf{W}_{k \times k}^{aa} \cdot \mathbf{a}_{k \times 1}^{prev} + \mathbf{b}_{k \times 1}^a)$$

$$\mathbf{y}_{m \times 1} = g(\mathbf{W}_{m \times k}^y \cdot [\mathbf{a}_{k \times 1}^{next}] + \mathbf{b}_{m \times 1}^y)$$



# Vanilla RNN vs Standard FNN

- For the sake of simplicity, I am going to switch to a simpler depiction, keeping the same formulas.



Recurrent Neural Network

$$\mathbf{a}_{k \times 1}^{next} = g \left( \mathbf{W}_{k \times (k+n)}^a \cdot \begin{bmatrix} \mathbf{x}_{n \times 1}^{prev} \\ \mathbf{a}_{k \times 1}^{prev} \end{bmatrix} + \mathbf{b}_{k \times 1}^a \right)$$

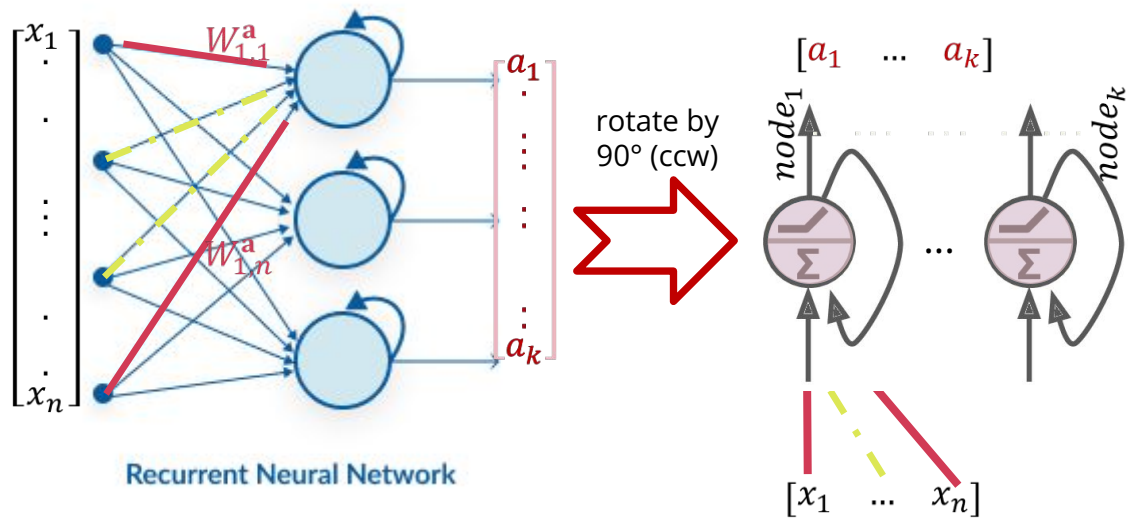
$$\mathbf{W}_{k \times (k+n)}^a = [\mathbf{W}_{k \times n}^{ax} \quad \mathbf{W}_{k \times k}^{aa}]$$

$$\mathbf{a}_{k \times 1}^{next} = g(\mathbf{W}_{k \times n}^{ax} \cdot \mathbf{x}_{n \times 1}^{prev} + \mathbf{W}_{k \times k}^{aa} \cdot \mathbf{a}_{k \times 1}^{prev} + \mathbf{b}_{k \times 1}^a)$$

$$\mathbf{y}_{m \times 1} = g(\mathbf{W}_{m \times k}^y \cdot [\mathbf{a}_{k \times 1}^{next}] + \mathbf{b}_{m \times 1}^y)$$

# RNN «unrolled in time»

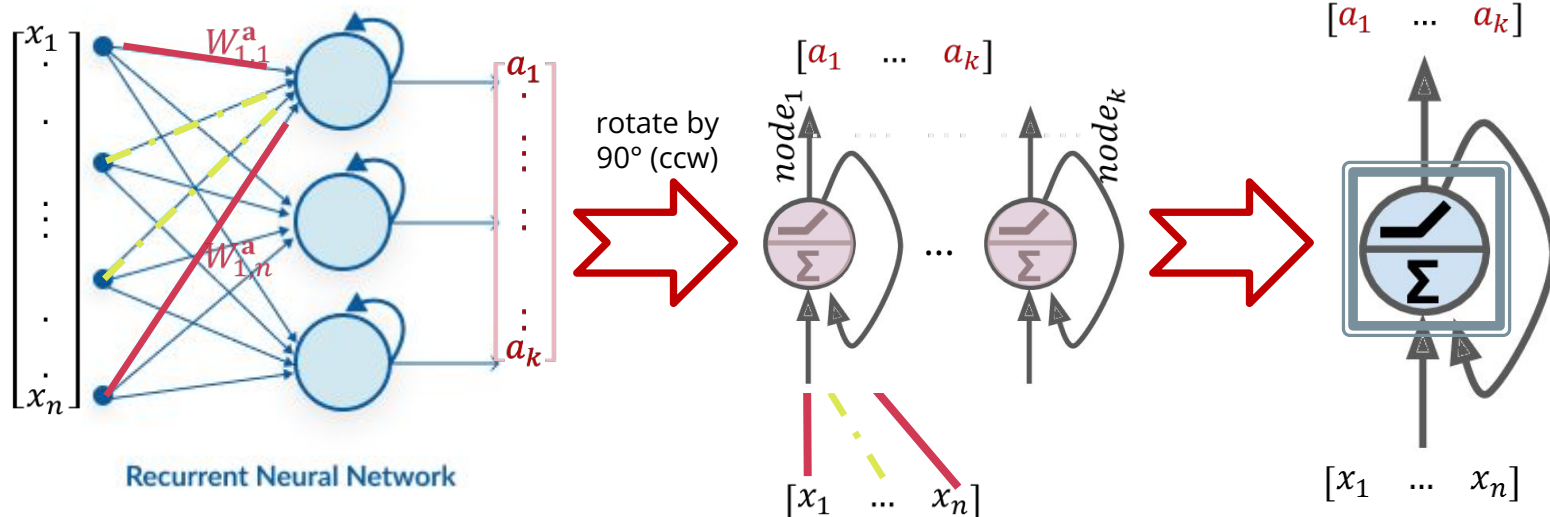
- We will first focus only on the core RNN (output layer is not shown!)



# RNN «super-node representation»

- We will first focus only on the core RNN (output layer is not shown!)

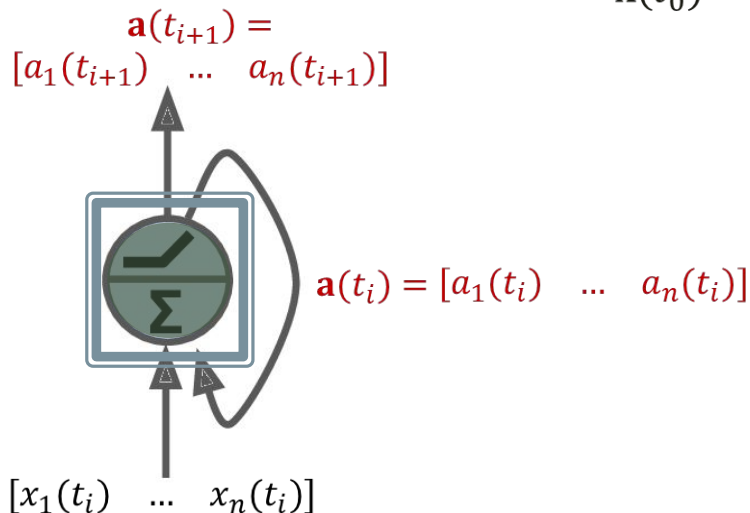
all recurrent nodes in a  
«single super-node» representation



# RNN «super-node representation»

- $\mathbf{x} = [x_1 \quad \dots \quad x_n]$  is a single vector at time say « $t_0$ ». Let's denote it as:

$$\mathbf{x}(t_0) = [x_1(t_0) \quad \dots \quad x_n(t_0)]$$

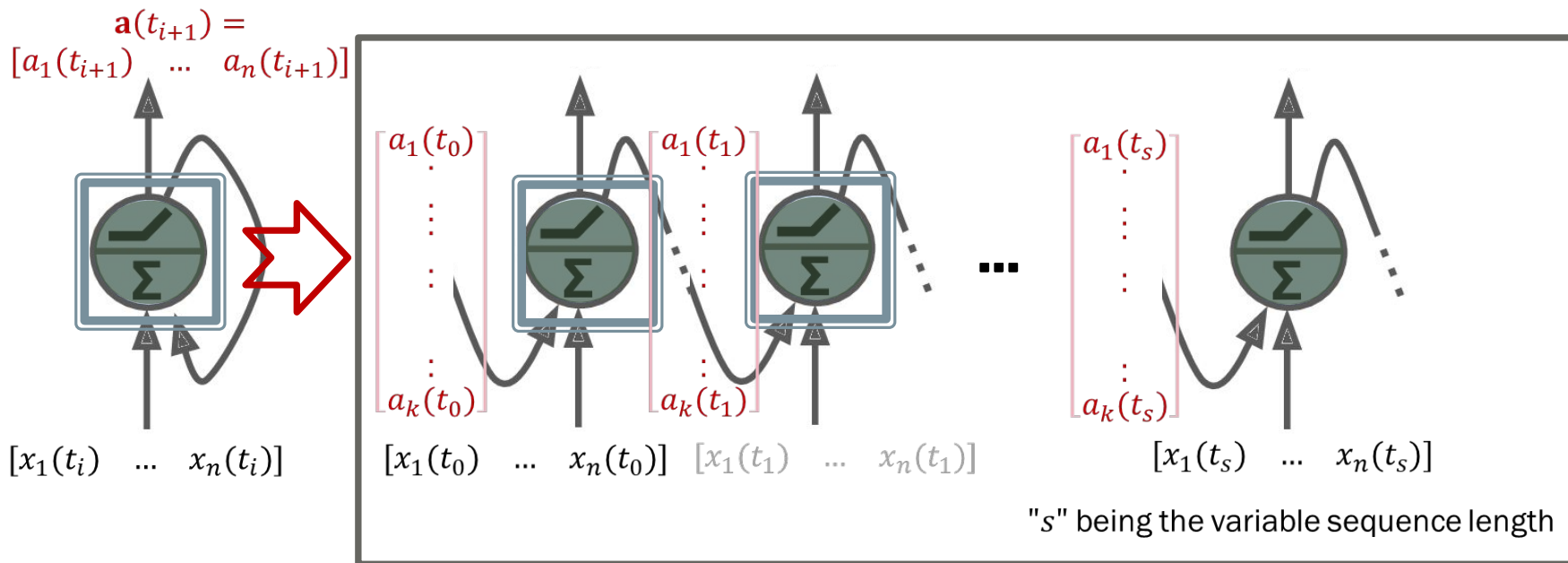


$$\mathbf{a}_{k \times 1}^{next} = g \left( \mathbf{W}_{k \times (k+n)}^a \cdot \begin{bmatrix} \mathbf{x}_{n \times 1}^{prev} \\ \mathbf{a}_{k \times 1}^{prev} \end{bmatrix} + \mathbf{b}_{k \times 1}^a \right)$$

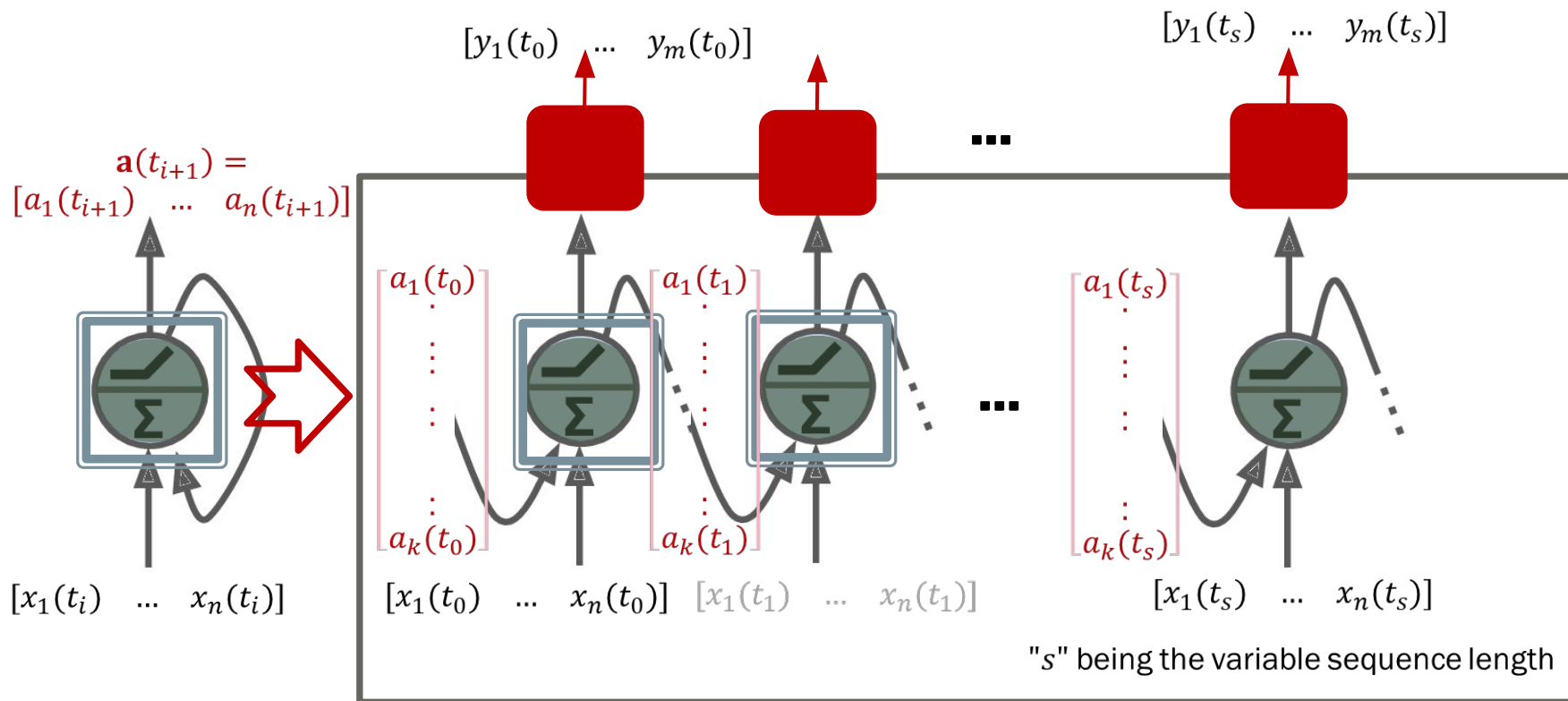
$$\mathbf{a}_{k \times 1}^{next} = g(\mathbf{W}_{k \times n}^{ax} \cdot \mathbf{x}_{n \times 1}^{prev} + \mathbf{W}_{k \times k}^{aa} \cdot \mathbf{a}_{k \times 1}^{prev} + \mathbf{b}_{k \times 1}^a)$$



# RNN «unrolled in time»

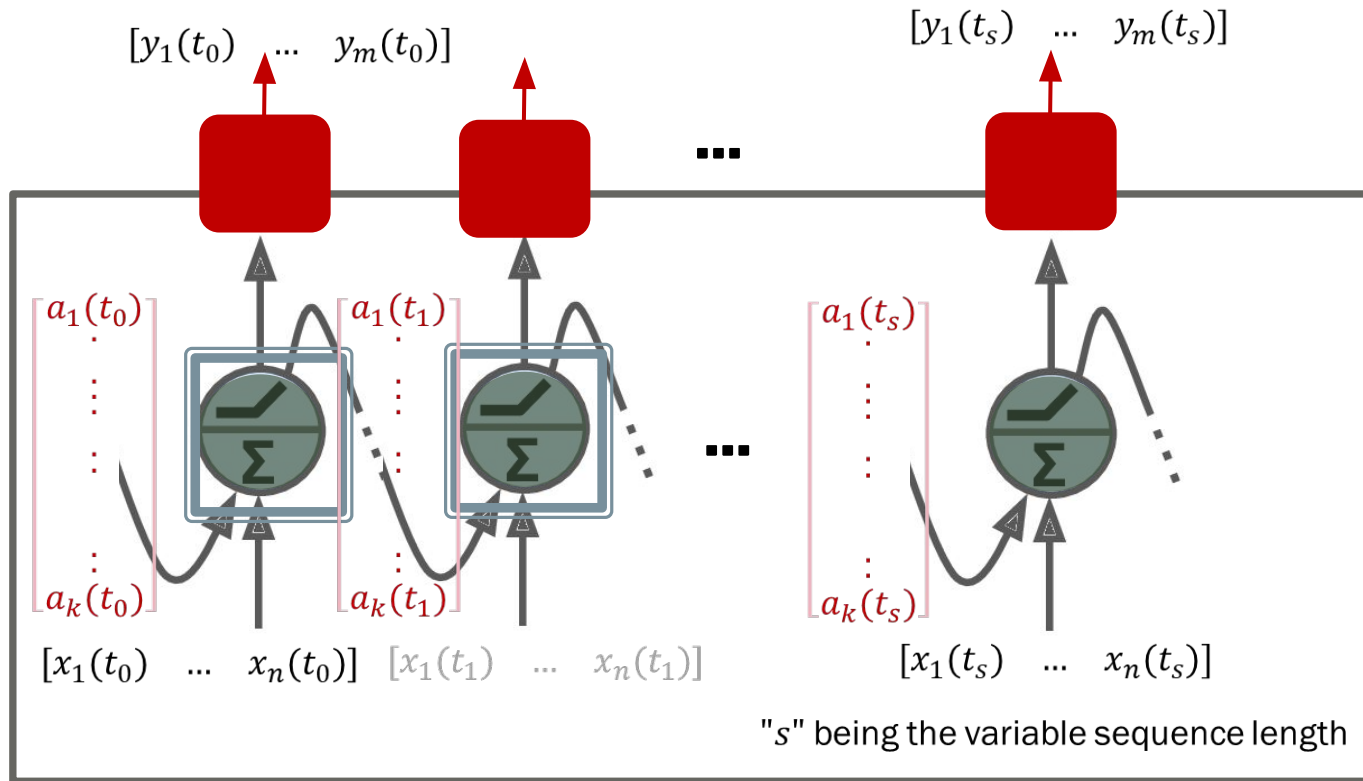


# Vanilla RNN «unrolled in time» with output



# Vanilla RNN «unrolled in time» with output

- k hidden nodes,
- s long sequence
- n long features
- m long output



# Vanilla RNN «unrolled in time» with output

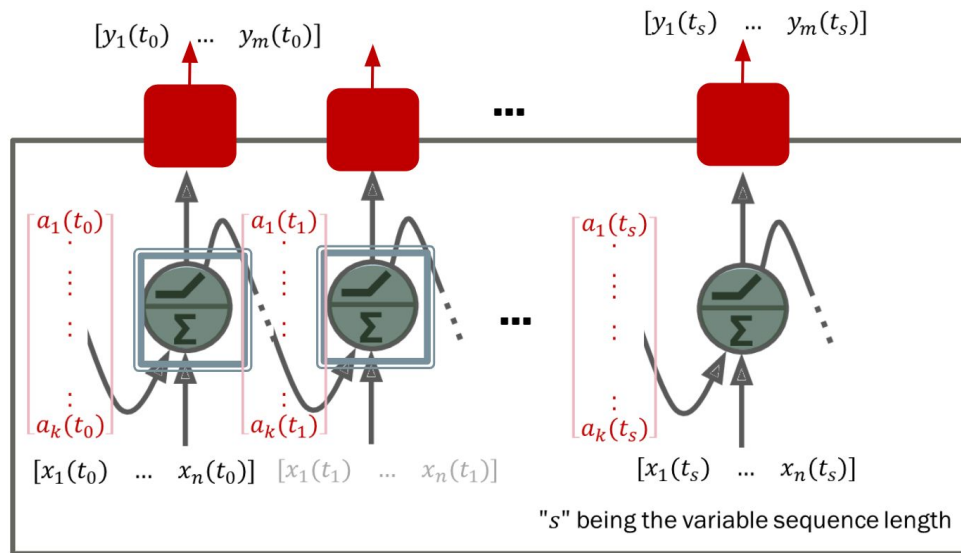
$$\mathbf{a}_{k \times 1}^{next} = g \left( \mathbf{W}_{k \times (k+n)}^a \cdot \begin{bmatrix} \mathbf{x}_{n \times 1}^{prev} \\ \mathbf{a}_{k \times 1}^{prev} \end{bmatrix} + \mathbf{b}_{k \times 1}^a \right)$$

$$\mathbf{W}_{k \times (k+n)}^a = [\mathbf{W}_{k \times n}^{ax} \quad \mathbf{W}_{k \times k}^{aa}]$$

$$\mathbf{a}_{k \times 1}^{next} = g(\mathbf{W}_{k \times n}^{ax} \cdot \mathbf{x}_{n \times 1}^{prev} + \mathbf{W}_{k \times k}^{aa} \cdot \mathbf{a}_{k \times 1}^{prev} + \mathbf{b}_{k \times 1}^a)$$

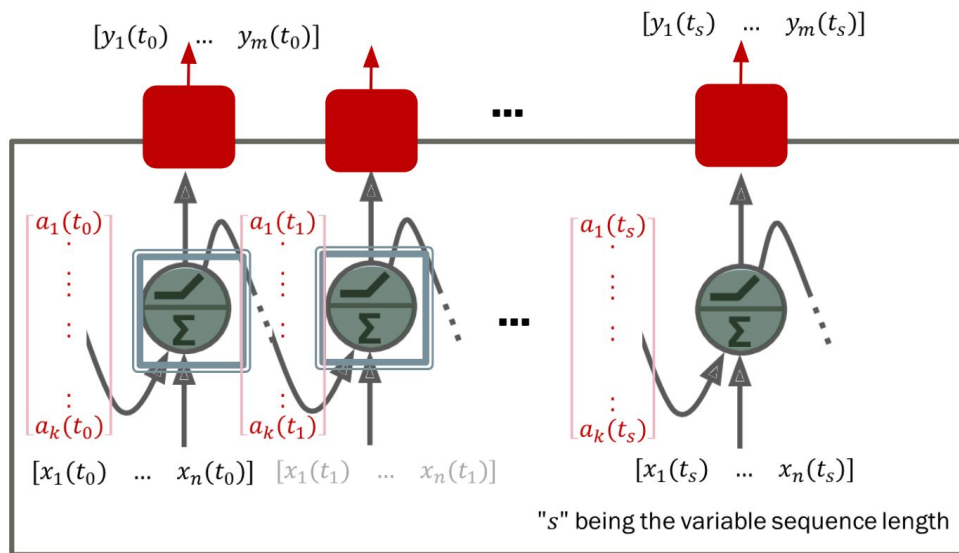
$$\mathbf{y}_{m \times 1} = g(\mathbf{W}_{m \times k}^y \cdot [\mathbf{a}_{k \times 1}^{next}] + \mathbf{b}_{m \times 1}^y)$$

Let's go over the formulation:



# Vanilla RNNs I/O Types

- The basic RNN structure defined previous corresponds to a **many-to-many** model.
  - It gets an «s» length sequence as **input**
  - and provides an «s» length sequence as **output**



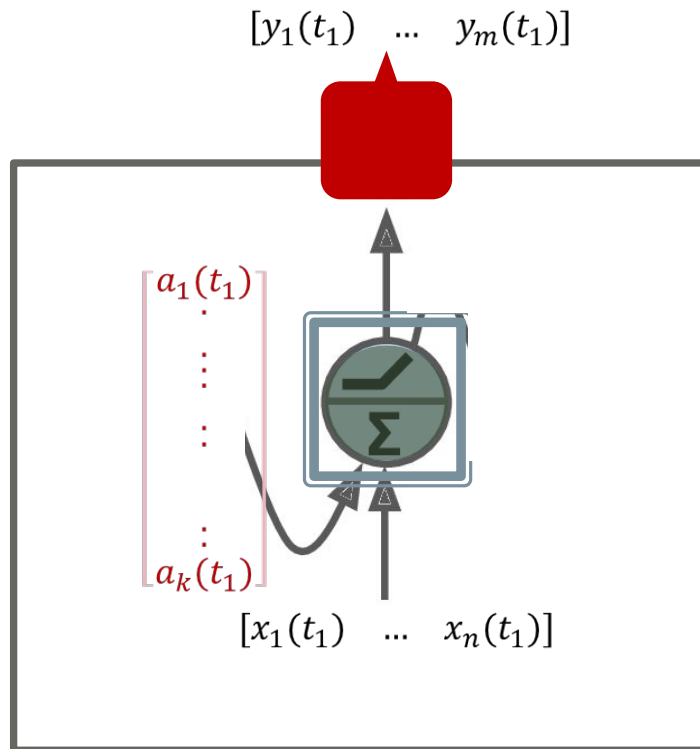
# Vanilla RNNs I/O Types

- With simple modifications on this «*many-to-many*» model, we may be able to create:
  - one-to-one
  - one-to-many
  - one-to-one (with-delay)
  - many-to-one
  - many-to-many (with delay)

models, as well!

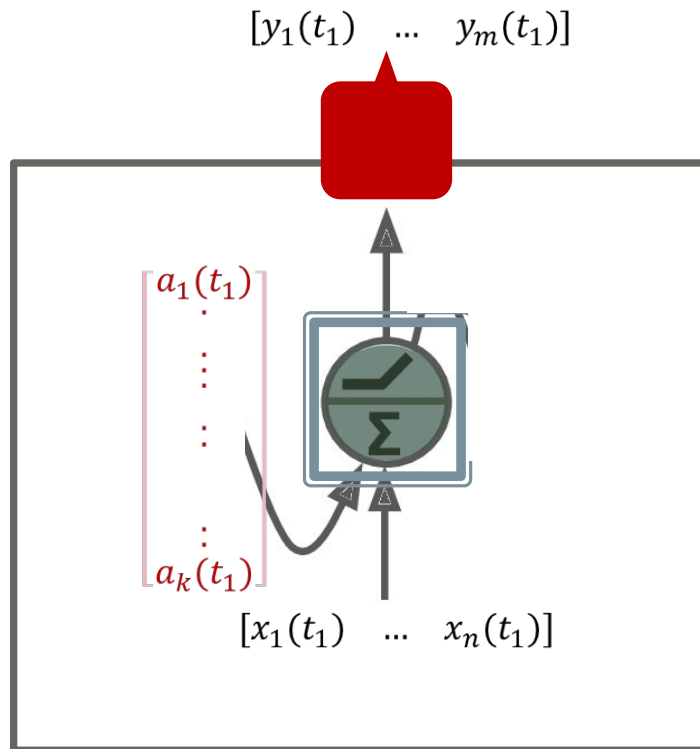
# One-to-One RNN

- This is like, calling the vanilla RNN for a fixed-sized vector (as input). You get a fixed-sized output.
- Is this an FNN?



# One-to-One RNN

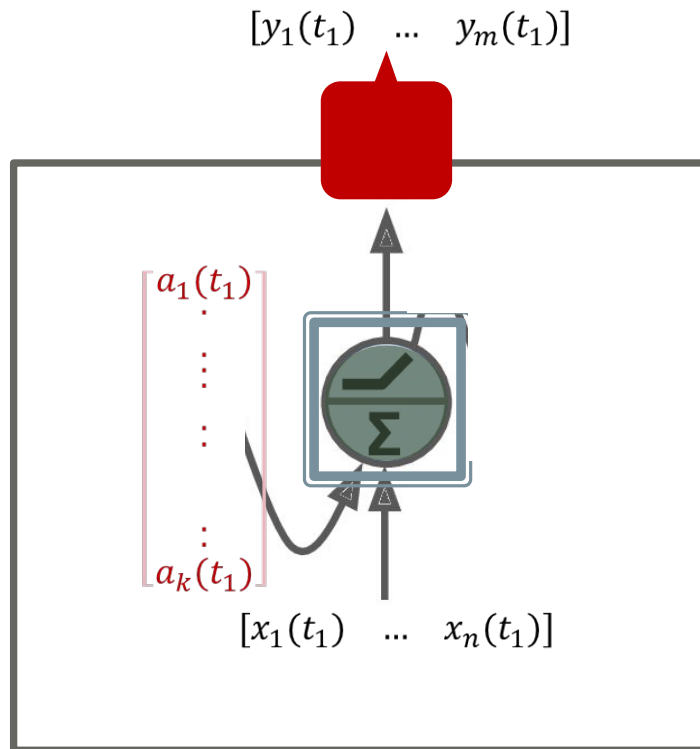
- This is like, calling the vanilla RNN for a fixed-sized vector (as input). You get a fixed-sized output.
- **No**, it is different from an FFN, because each new call will correlate with previous calls.
- If you do not want it to correlate to previous calls, that means you do not want a «memory»
- Then use an FFN.





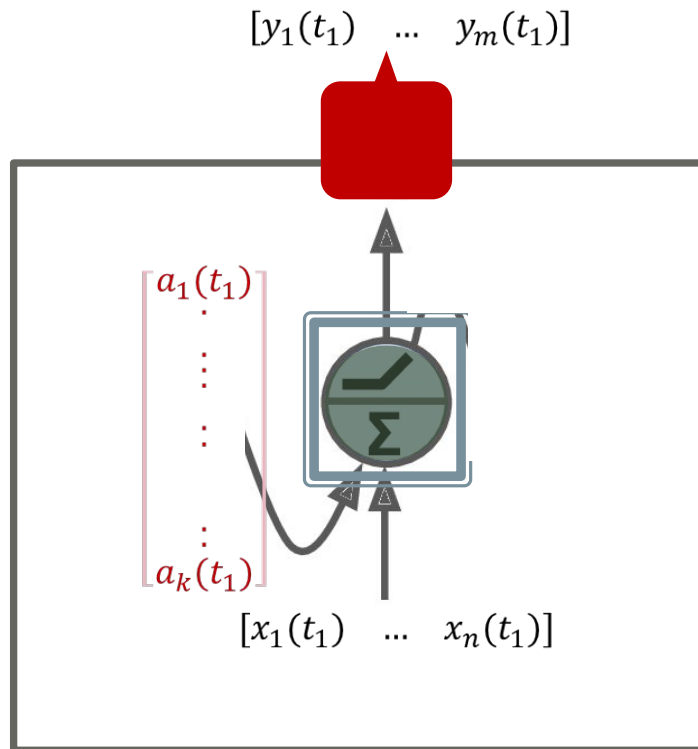
# One-to-One RNN

- This is like, calling the vanilla RNN for a fixed-sized vector (as input). You get a fixed-sized output.
- **No**, it is different from an FFN, because each new call will correlate with previous calls.
- If you do not want it to correlate to previous calls, that means you do not want a «memory»
- Example: ?



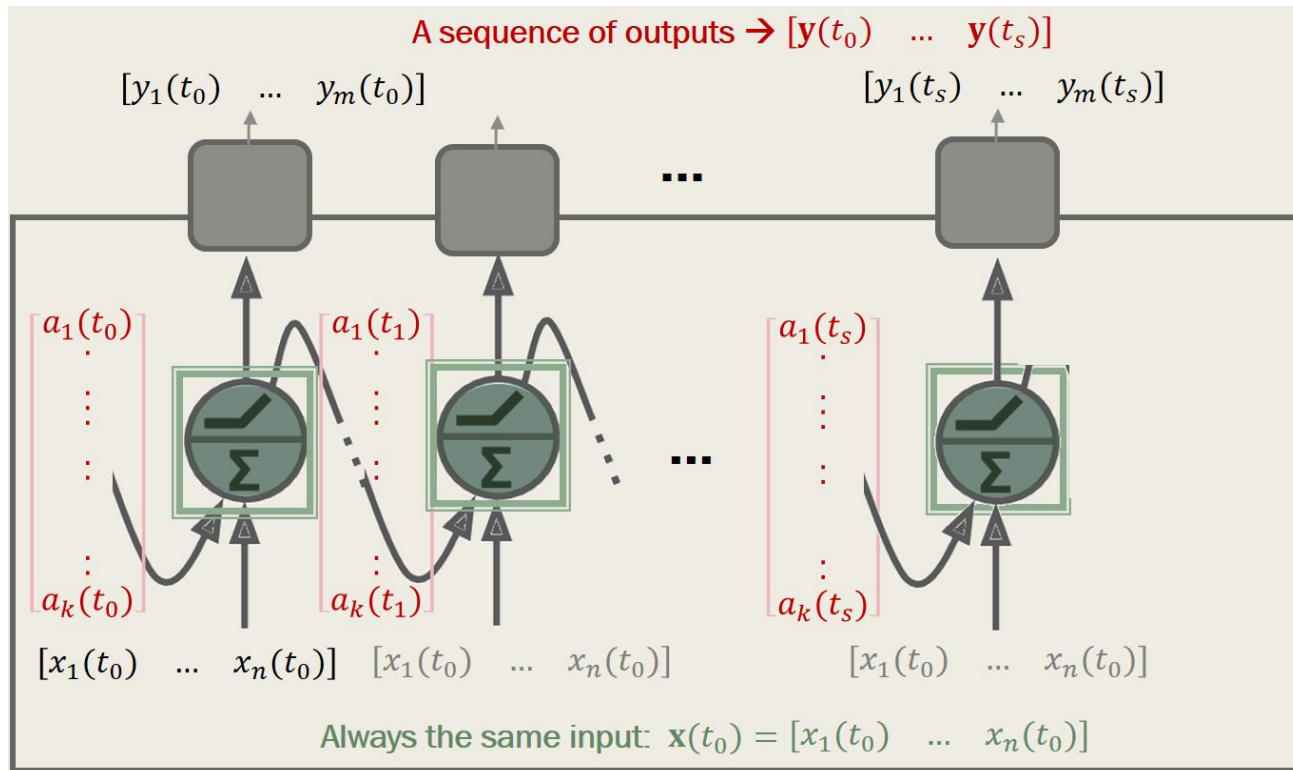
# One-to-One RNN

- This is like, calling the vanilla RNN for a fixed-sized vector (as input). You get a fixed-sized output.
- **No**, it is different from an FFN, because each new call will correlate with previous calls.
- If you do not want it to correlate to previous calls, that means you do not want a «memory»
- Example: *Random Sample Generator with a specific distribution.*
  - «like drawing “a card” from a deck»



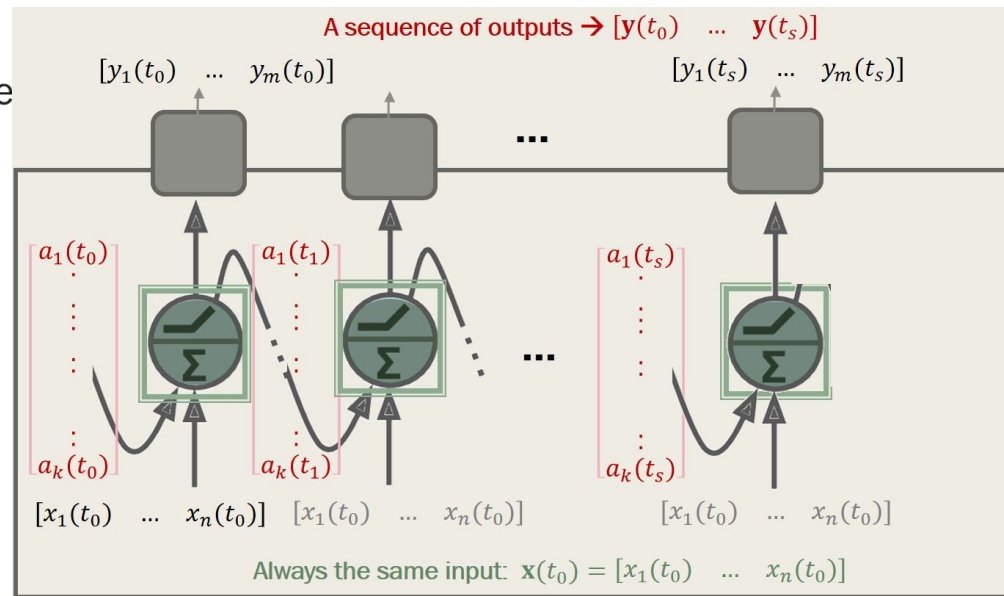
# One-to-Many RNN

- This is like, calling the vanilla RNN for multiple times using the same fixed size vector (as input), and getting a sequence of outputs.
- The output sequence can be a varying or fixed length sequence of vectors.



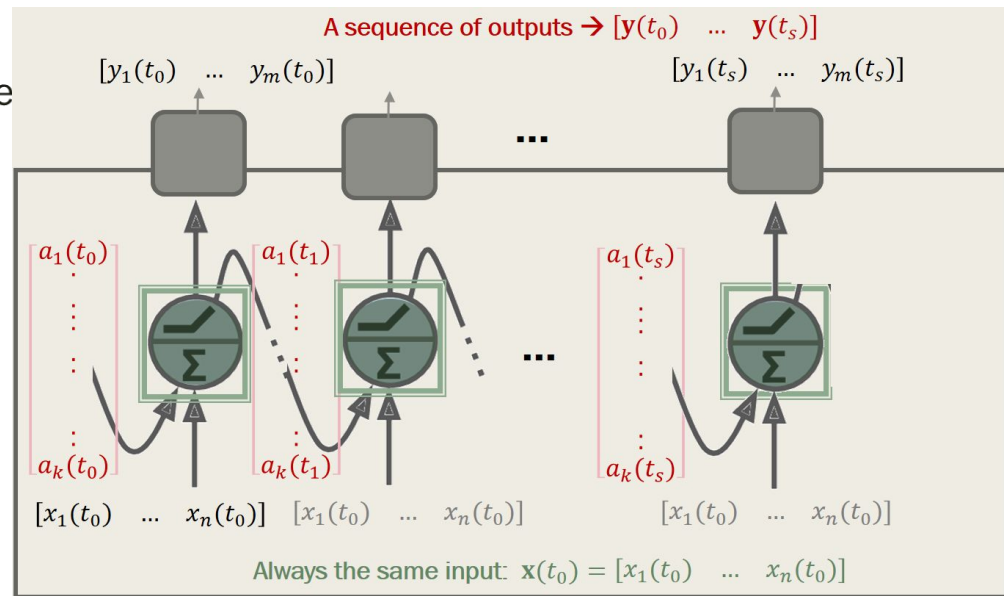
# One-to-Many RNN

- This looks like a fixed length output of length «s»
- How to make it variable length
- Maybe we could keep going (i.e. Calling the RNN with the same input) until a special  $\tilde{y} = [y_1 \dots y_m]$  arrives.
  - Like the end of a sentence (dot).
- Example: ?



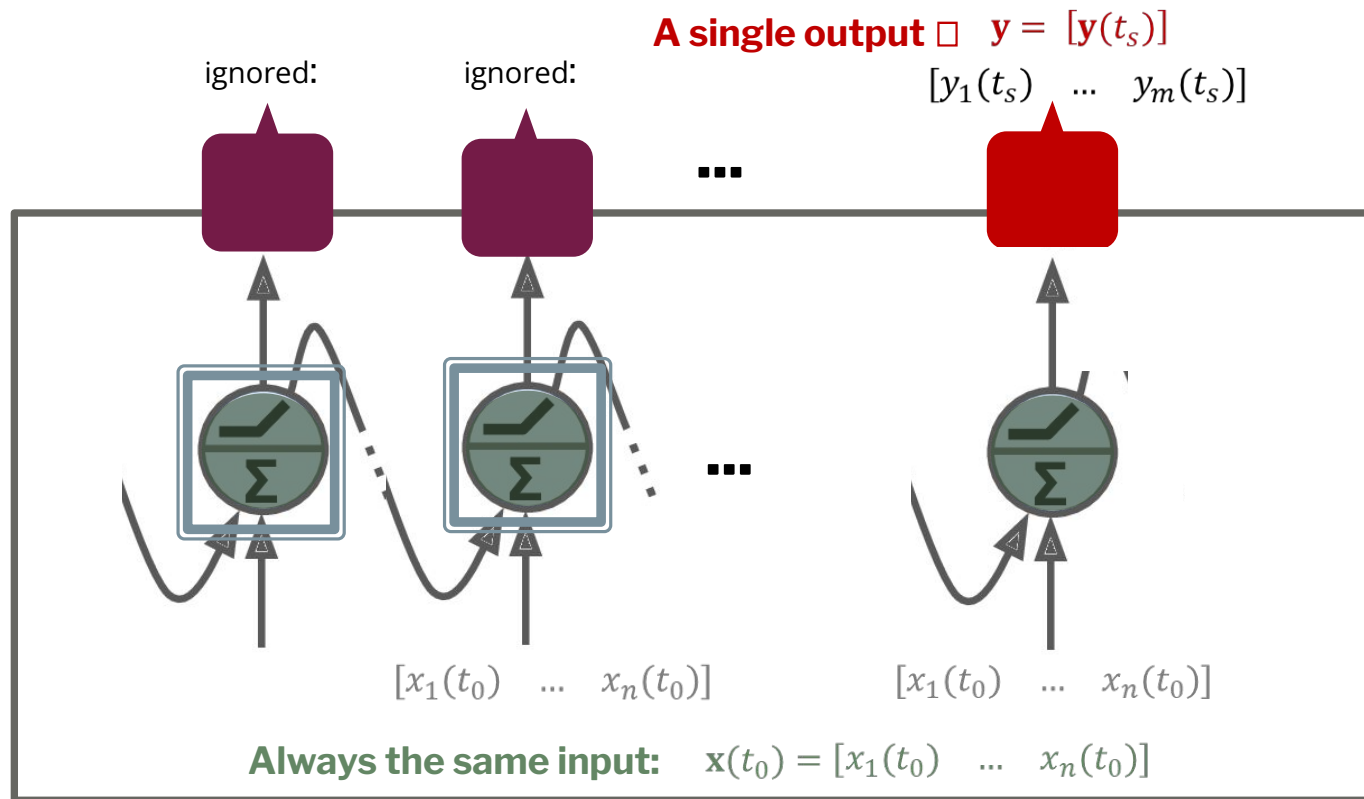
# One-to-Many RNN

- This looks like a fixed length output of length «s»
- How to make it variable length
- Maybe we could keep going (i.e. Calling the RNN with the same input) until a special  $\tilde{y} = [y_1 \dots y_m]$  arrives.
  - Like the end of a sentence (dot).
- Example: Music Generation
  - input = «A harmonic minor»



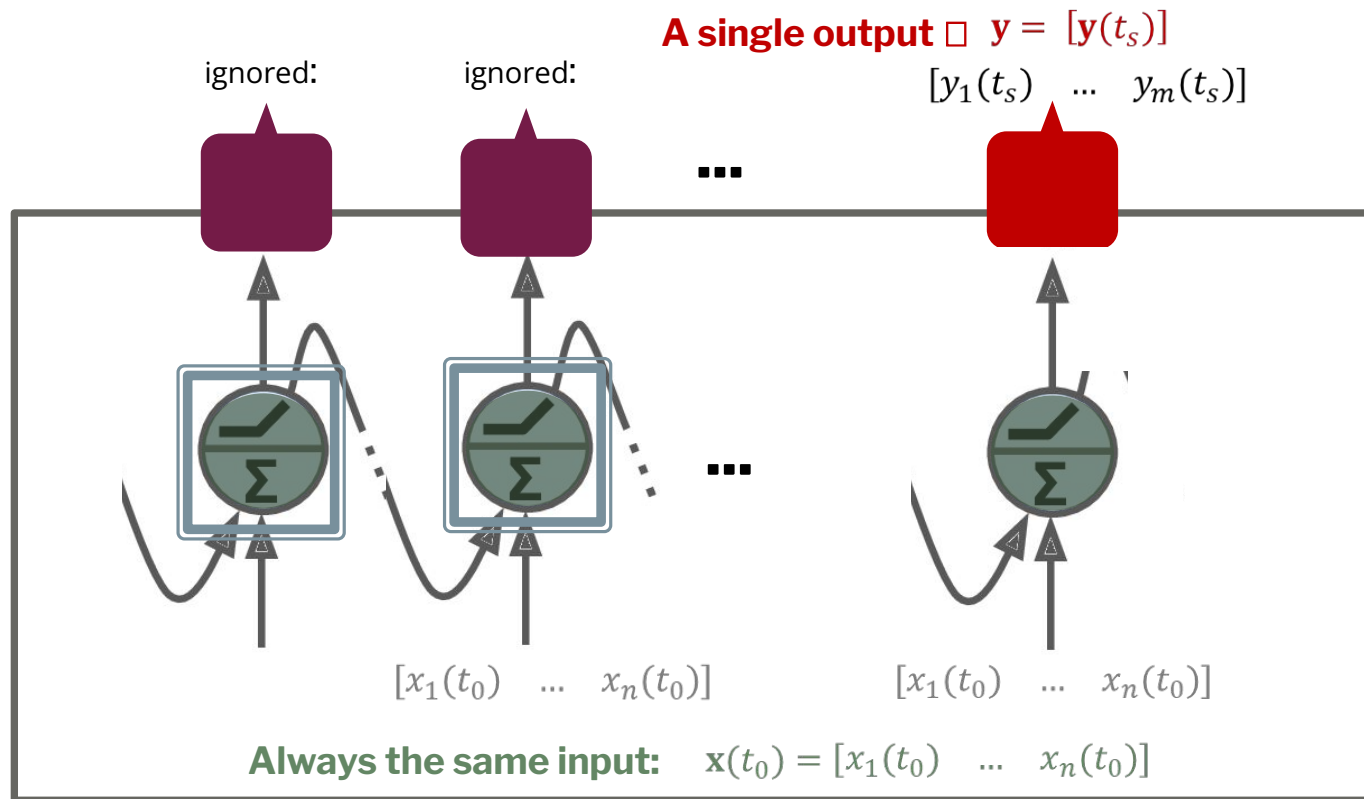
# One-to-One RNN (with delay)

- This is like, calling the vanilla RNN for multiple times using the same fixed size vector (as input), and getting a sequence of output.
- The output sequence can be a varying or fixed length sequence of vectors.



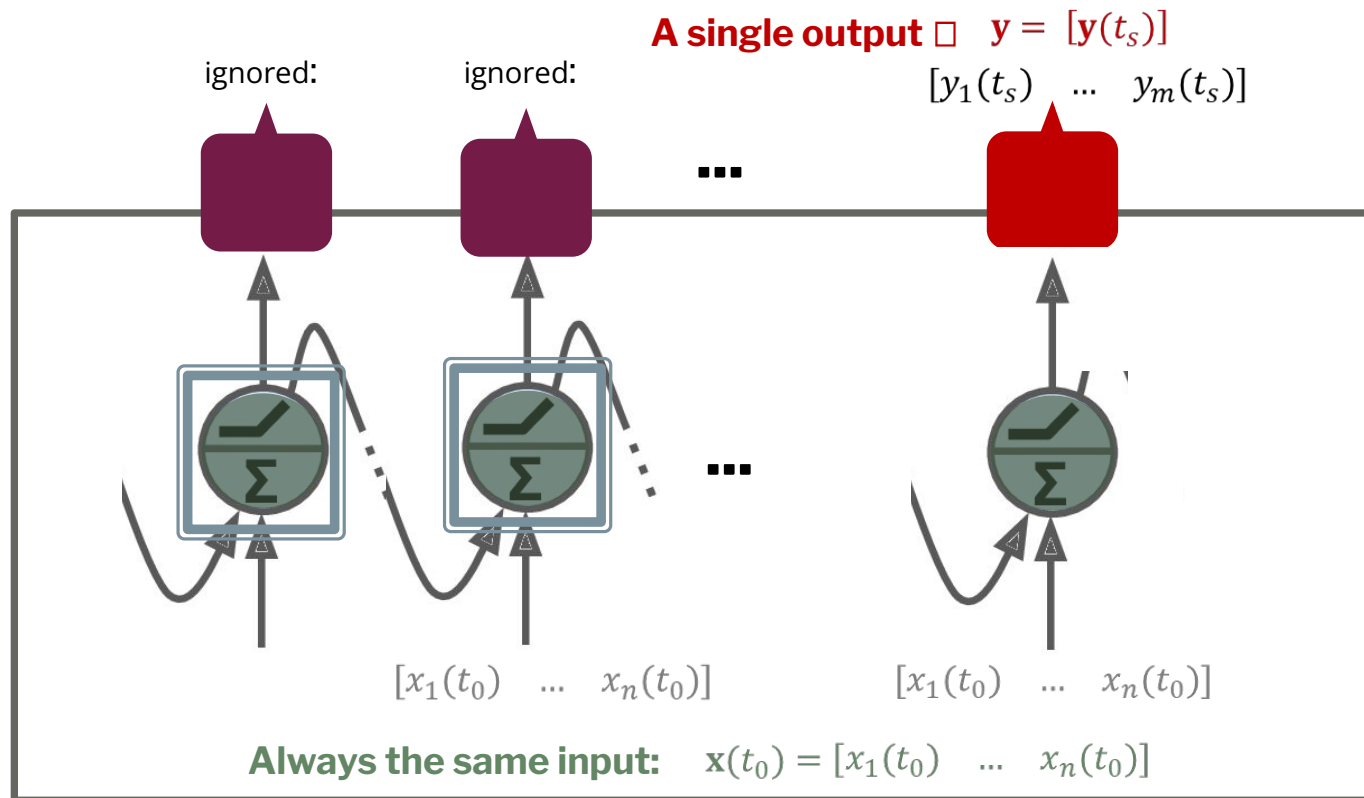
# One-to-One RNN (with delay)

- It is almost the same architecture as one-to-many, but only the last output is considered.
- Simply ignore the intermediate outputs.



# One-to-One RNN (with delay)

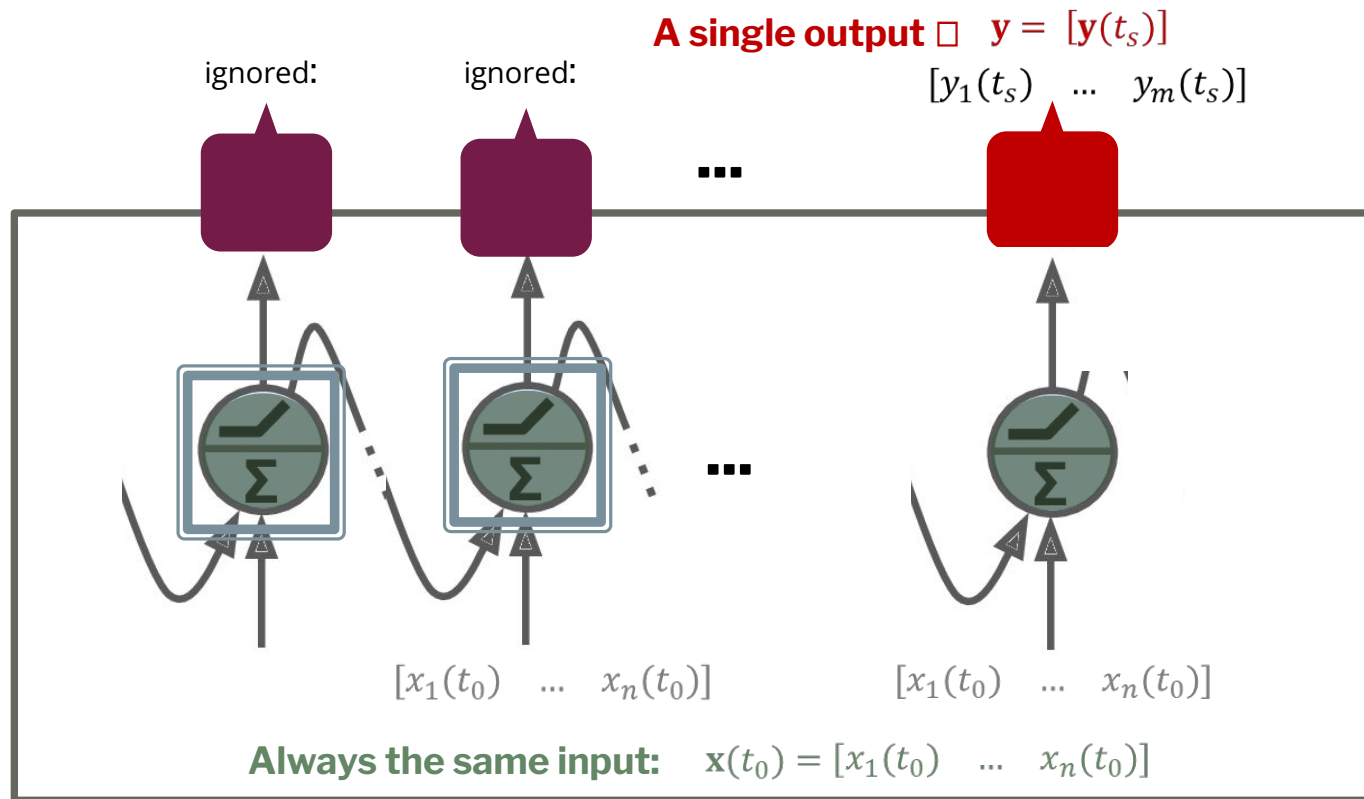
- So what is it with the delay?
- Why wait (repeat)?
- How many times to wait/repeat?
  - Fixed / varying?
- How do I know this is a one-to-one with delay, or one-to-many model?
- Who is ignoring the intermediate outputs? Is it only me?





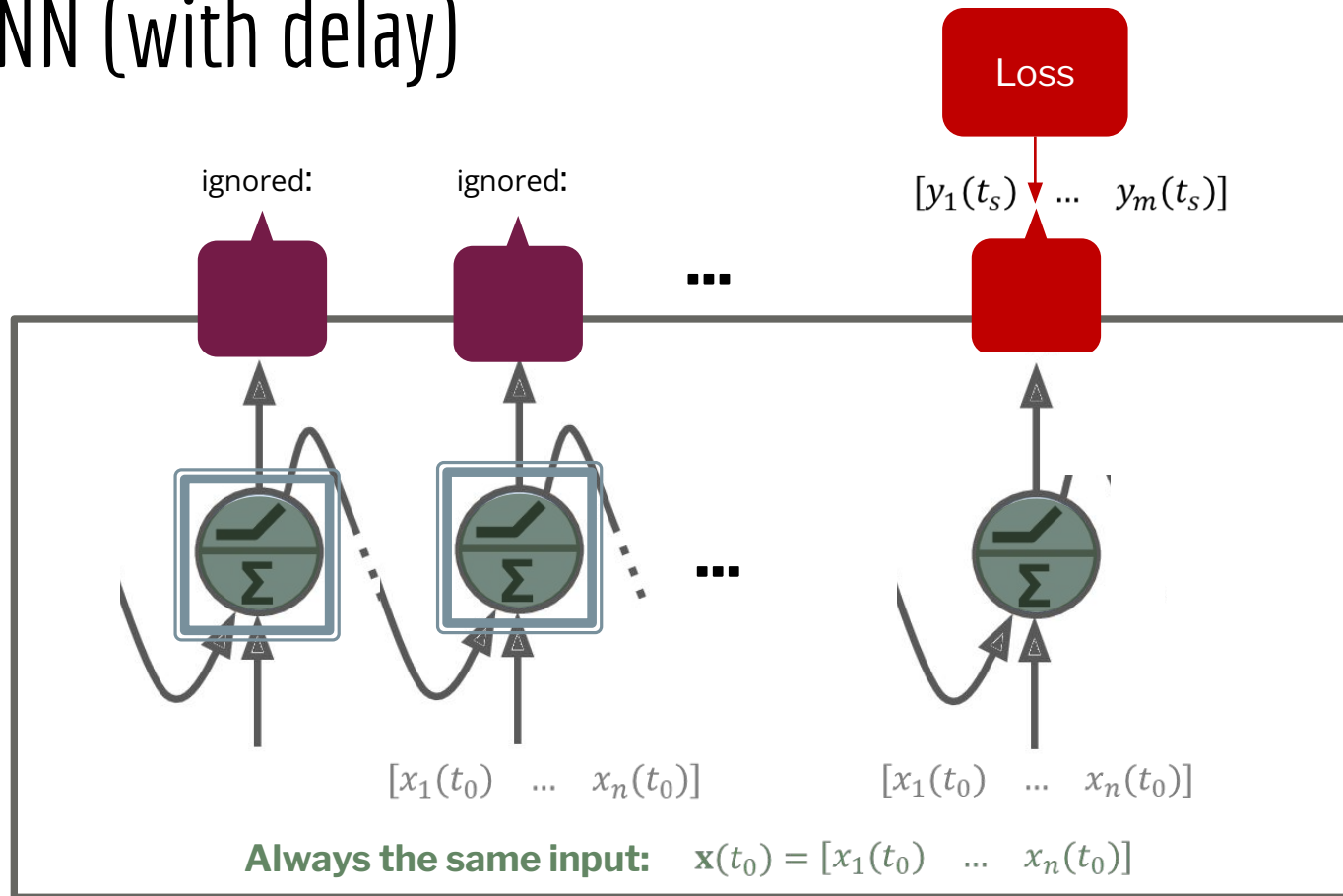
# One-to-One RNN (with delay)

- So what is it with the delay?
- Why wait (repeat)?
- How many times to wait/repeat?
  - Fixed / varying?
- How do I know this is a one-to-one with delay, or one-to-many model?
- Who is ignoring the intermediate outputs?
  - No, it is the training.



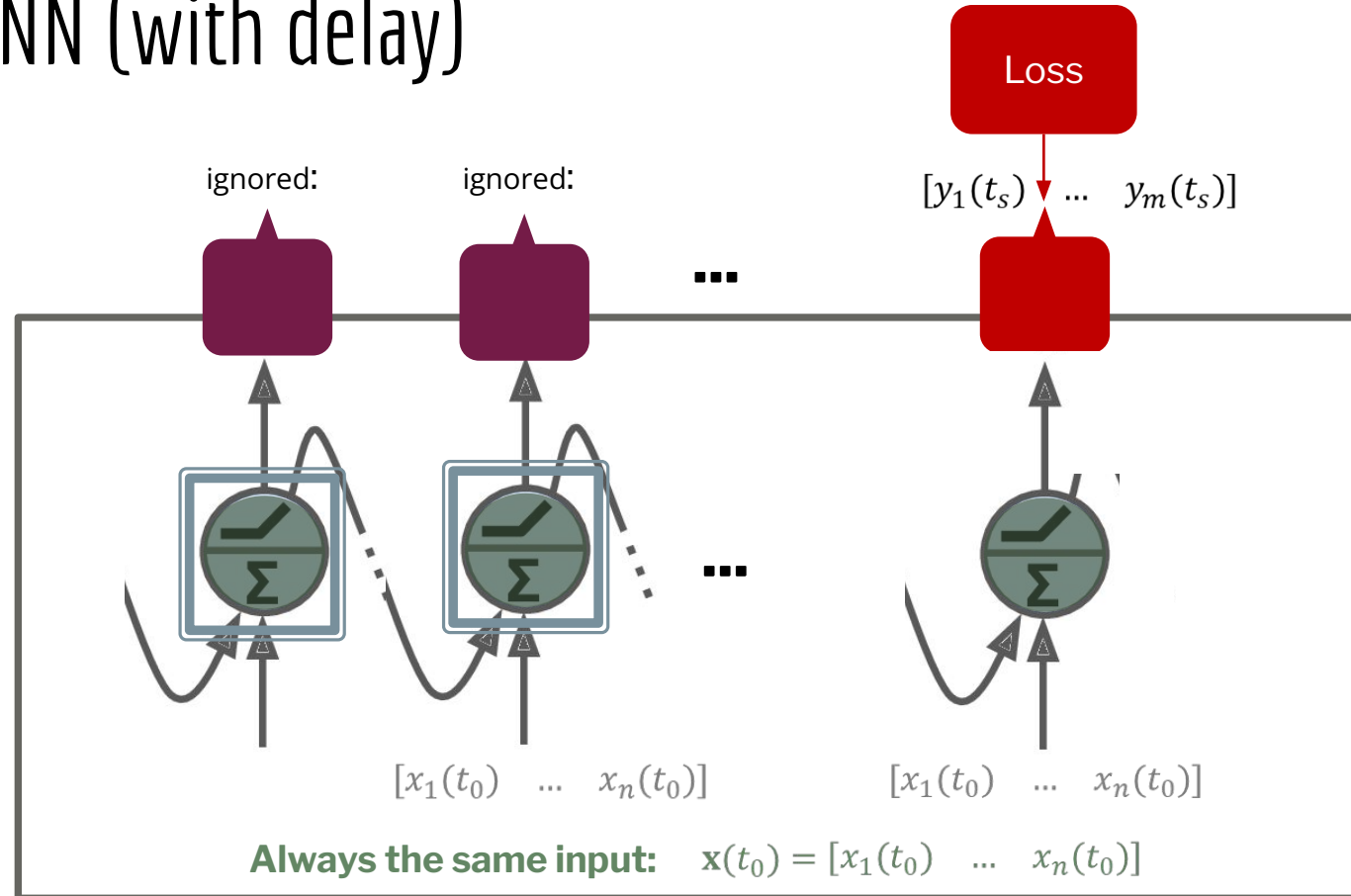
# One-to-One RNN (with delay)

- Example: ?
  - Let's think about why there is a need for a delay/repeat.



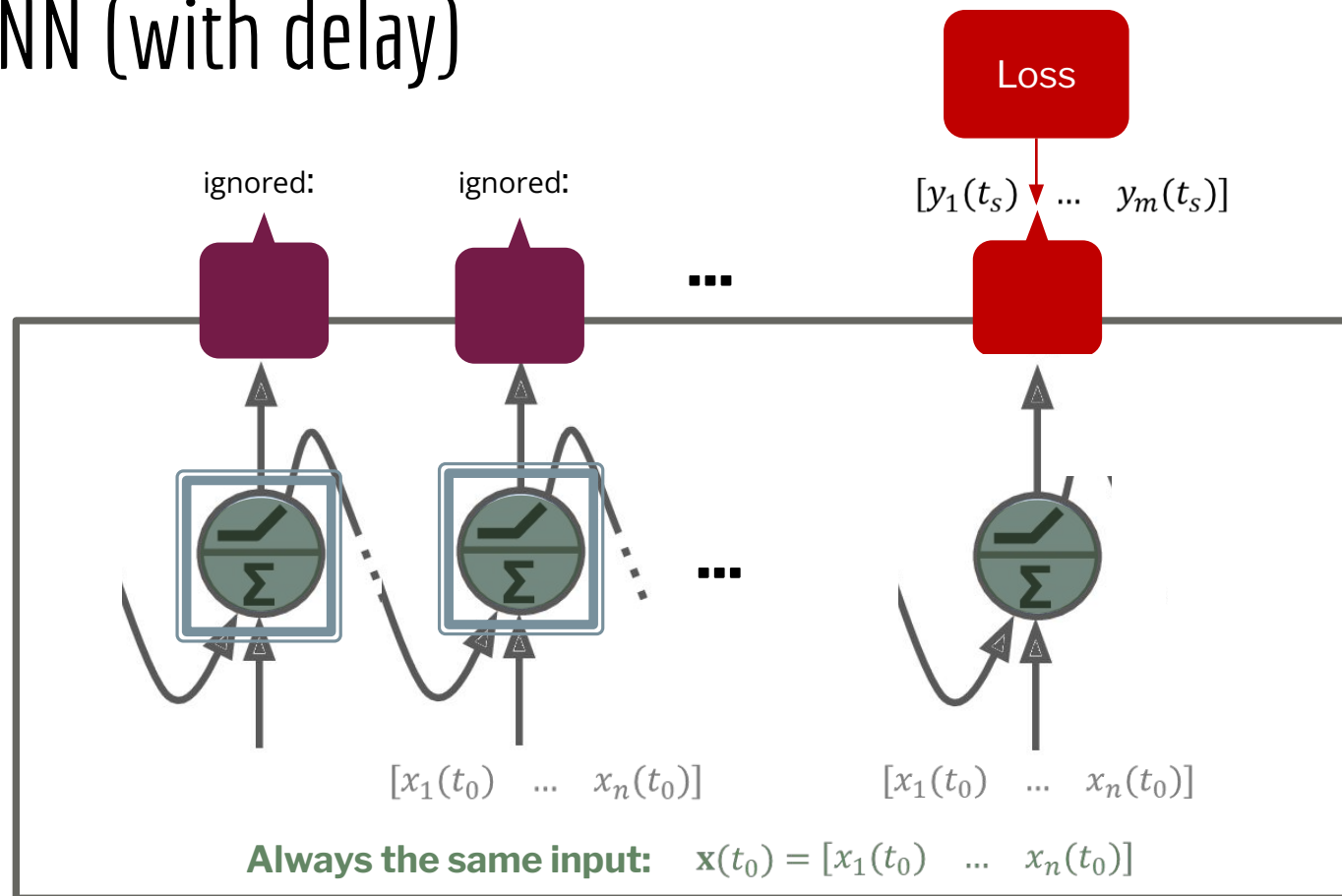
# One-to-One RNN (with delay)

- Example: ?
  - Let's think about why there is a need for a delay/repeat.



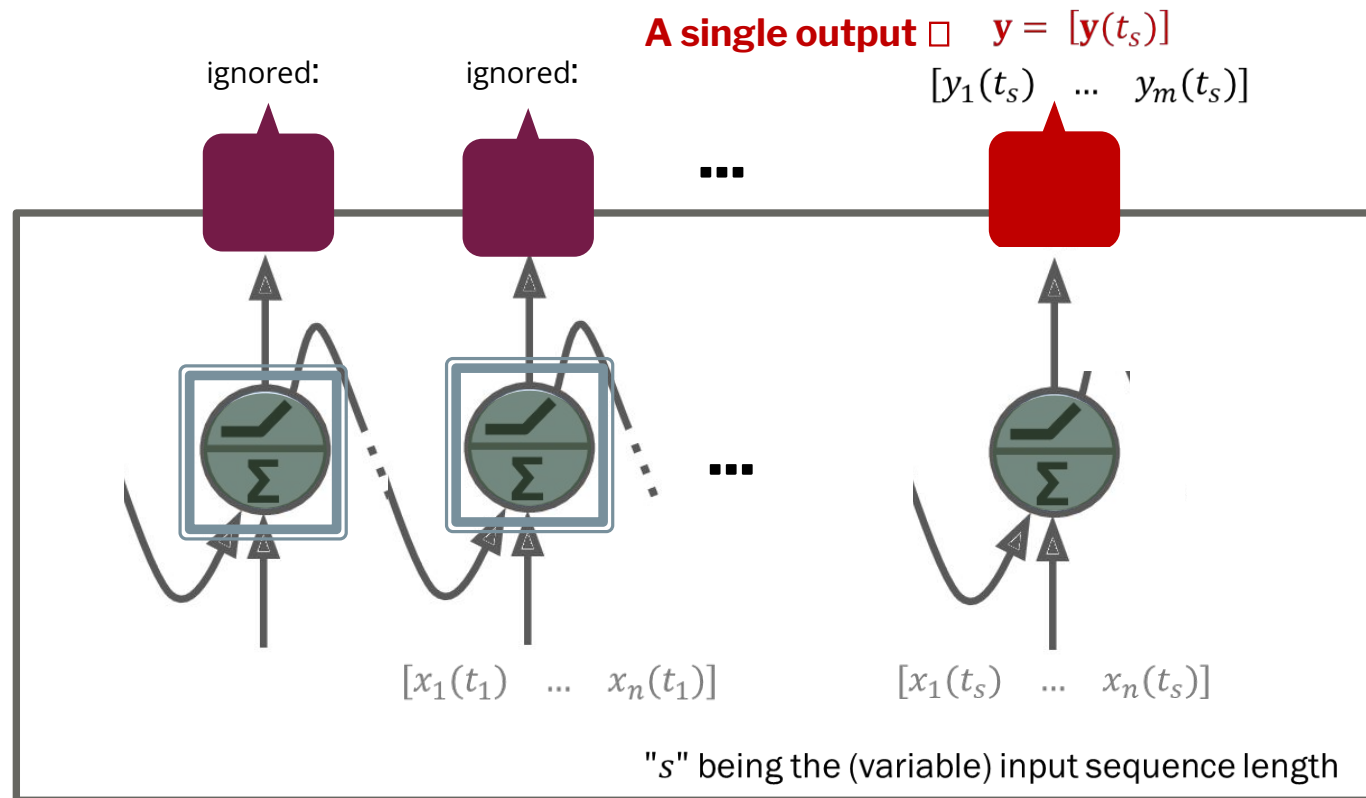
# One-to-One RNN (with delay)

- Example:  
*Dynamic System Simulation*
  - Like a second order control system.
  - Because these systems have a natural delay of their own.
  - But they can also be modelled as many-to-many (with delay).



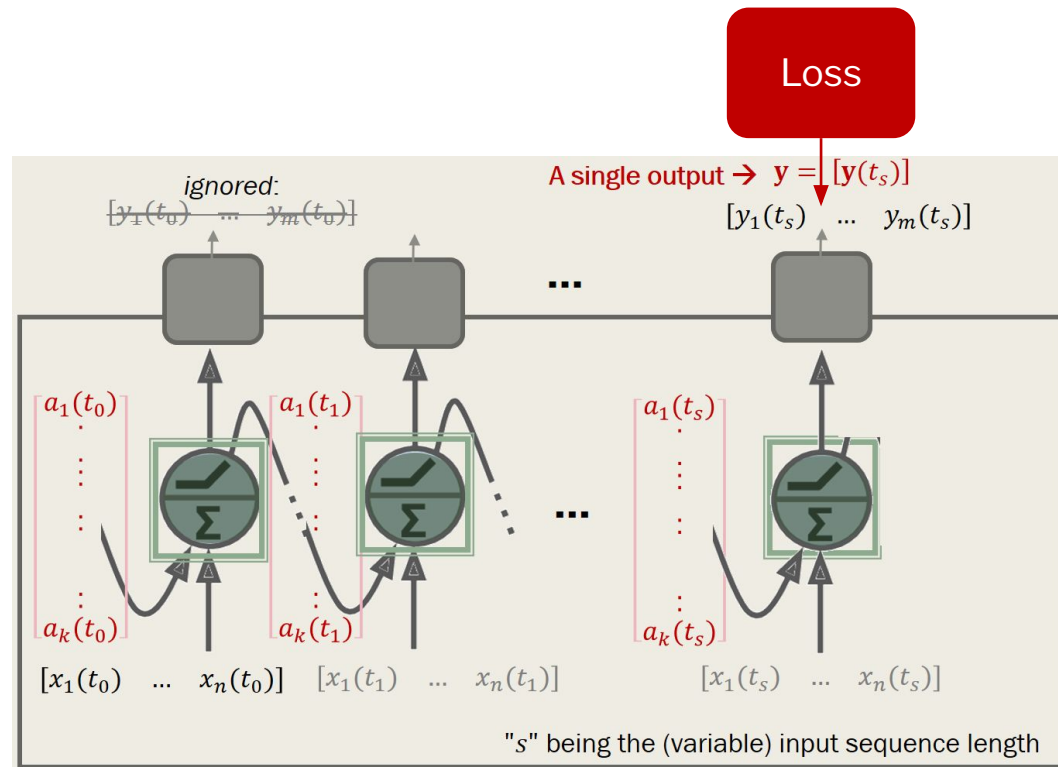
# Many-to-one RNN

- It is almost the same architecture as «one-to-one with delay».
- The only difference is, the input feed is a sequence.



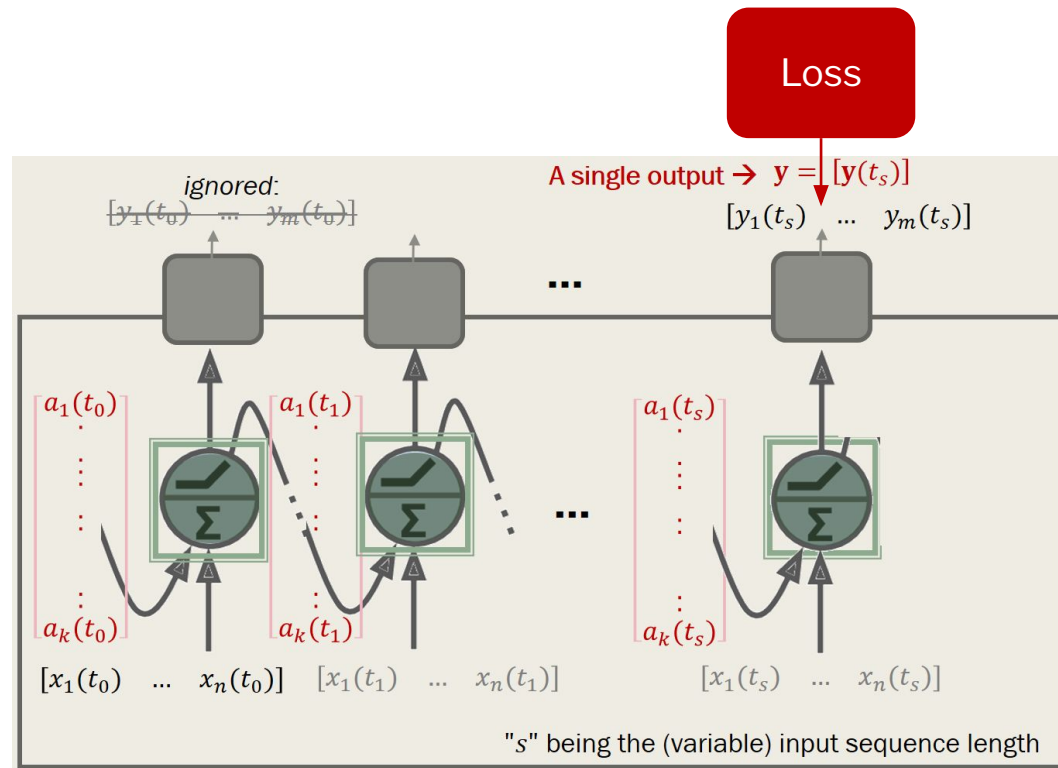
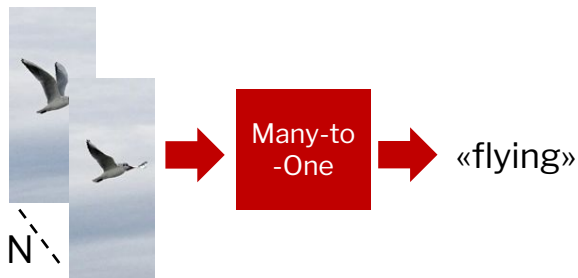
# Many-to-one RNN

- So training them will be the same as well.
- We simply use a different input set.
- Example ?



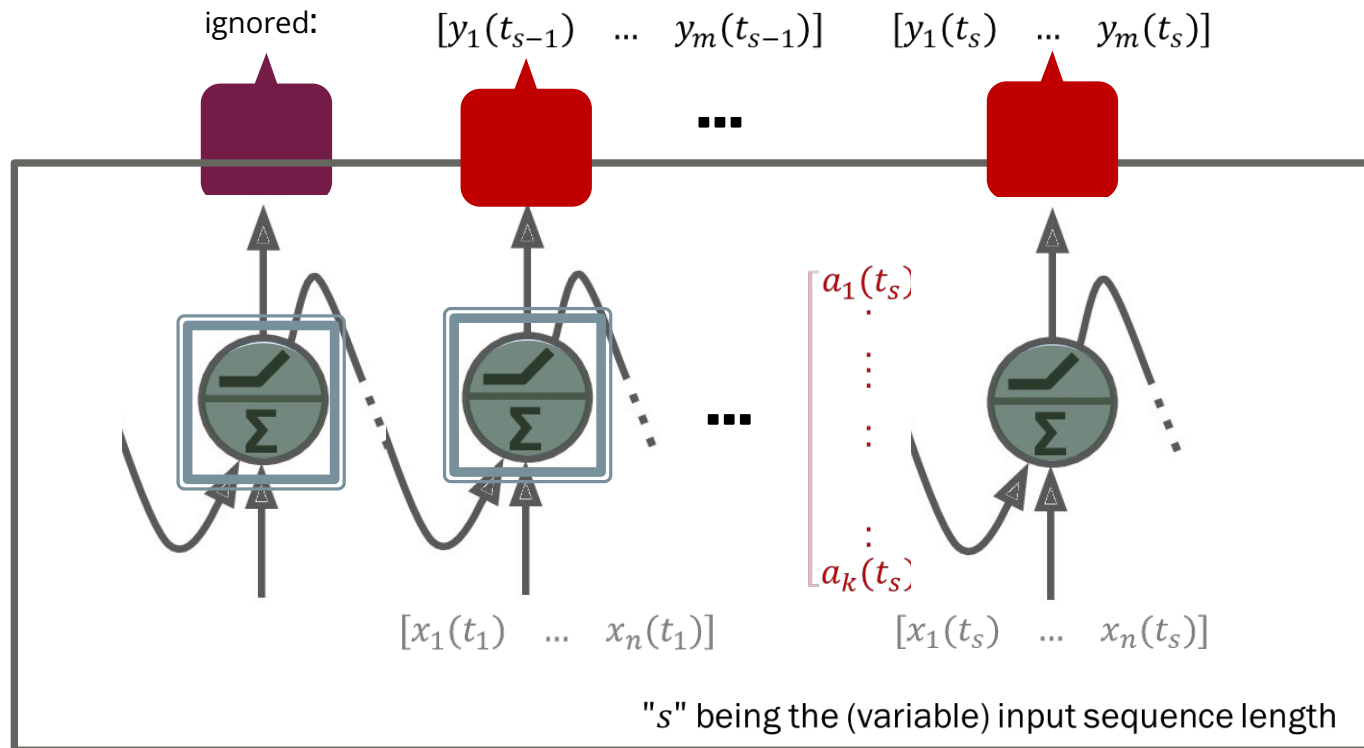
# Many-to-one RNN

- So training them will be the same as well.
- We simply use a different input set.
- Example : **action recognition from a video**



# Many-to-Many RNN (with delay)

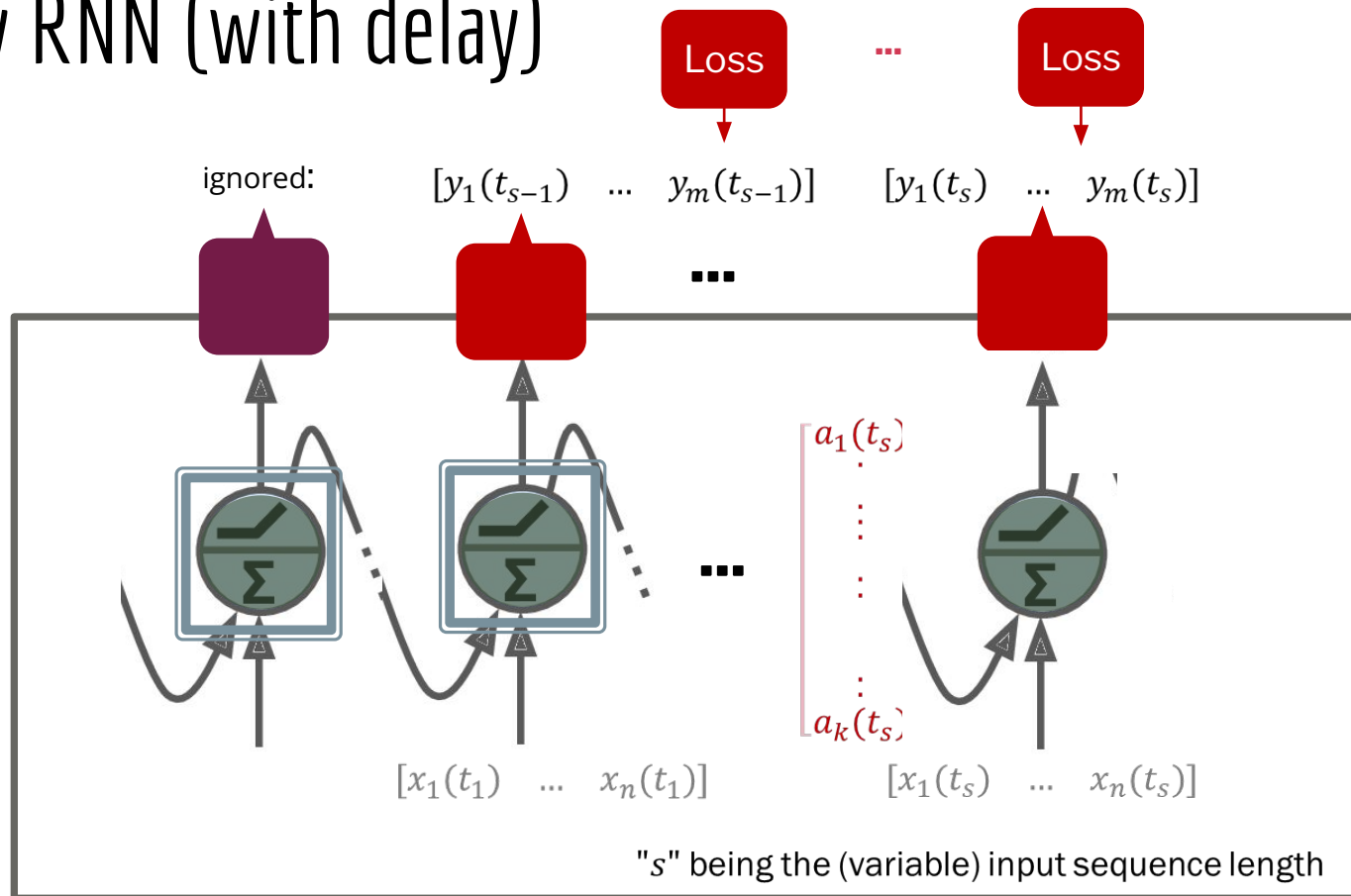
- This is like many-to-many architecture, but you just ignore some of the output.
- Which means...





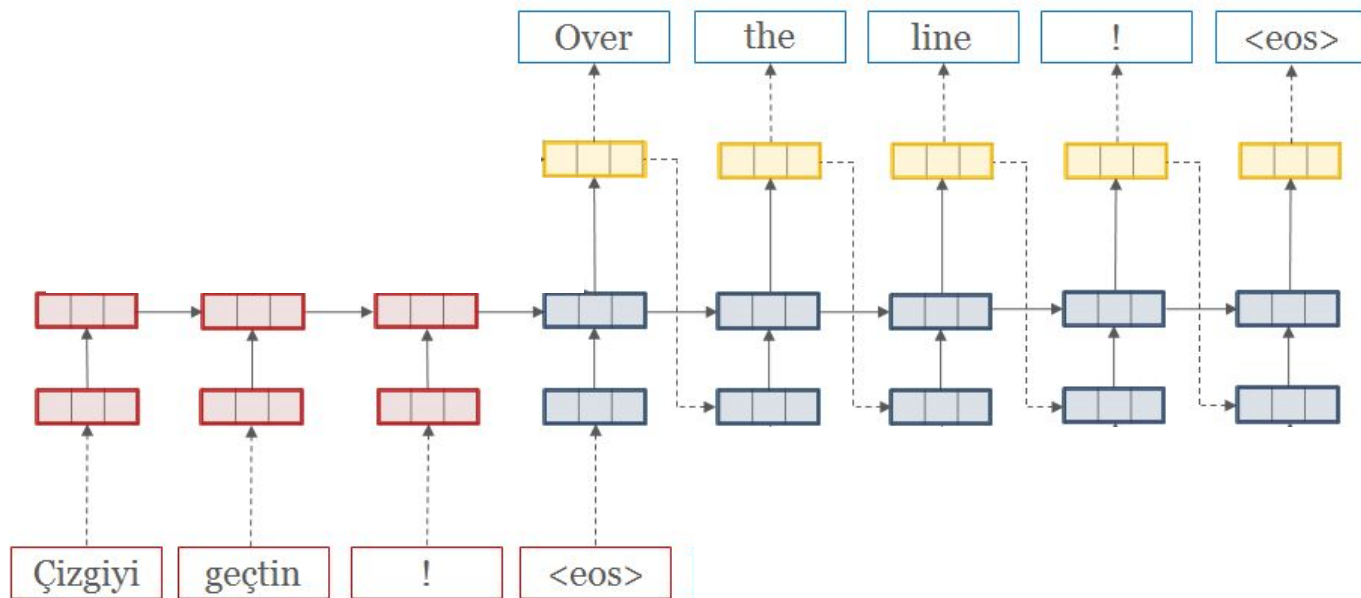
# Many-to-Many RNN (with delay)

- Not only the last, but some other recent output is also used in training.
- Example: ?



# Many-to-Many RNN (with delay)

- Not only the last, but some other recent output is also used in training.
- Example: **language translation**



# What will we do next week?

- LSTMs and Introduction to NLP ...

# Additional Reading & References

- <https://arxiv.org/ftp/arxiv/papers/2001/2001.07092.pdf>
- <https://embryo.asu.edu/pages/david-h-hubel-and-torsten-n-wiesel-s-research-optical-development-kittens>
- <https://becominghuman.ai/from-human-vision-to-computer-vision-convolutional-neural-network-part3-4-24b55ffa7045>
- <https://www.cns.nyu.edu/~david/courses/perception/lecturenotes/ganglion/ganglion.html>
- <https://christophm.github.io/interpretable-ml-book/cnn-features.html>
- <https://www.mathworks.com/help/deeplearning/ug/visualize-activations-of-a-convolutional-neural-network.html>
- <https://www.mathworks.com/help/deeplearning/ug/visualize-activations-of-a-convolutional-neural-network.html>
- <https://github.com/utkuozbulak/pytorch-cnn-visualizations>
- <https://www.youtube.com/watch?v=AgkflQ4IGaM>