



RECURRENT NEURAL NETWORKS

OUTLINE

1. Introduction
2. The Recurrent Neural Network
3. RNN applications
4. The RNN model
5. Gated RNN (LSTM and GRU)



1. INTRODUCTION

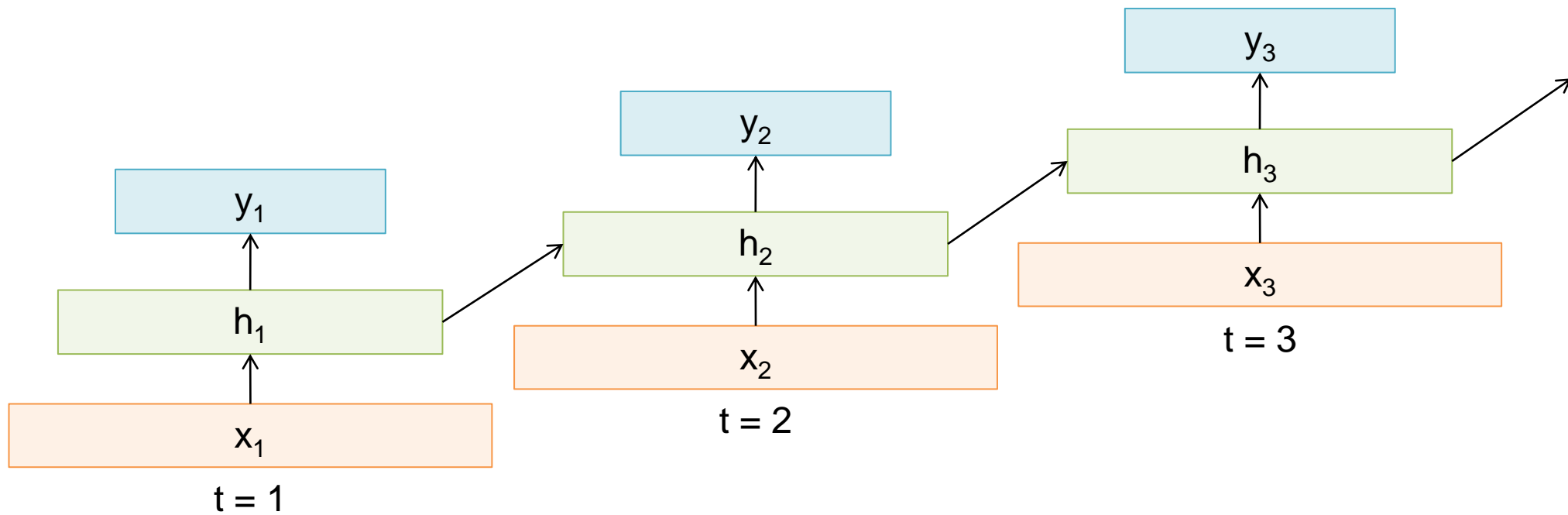




What is the next scene?



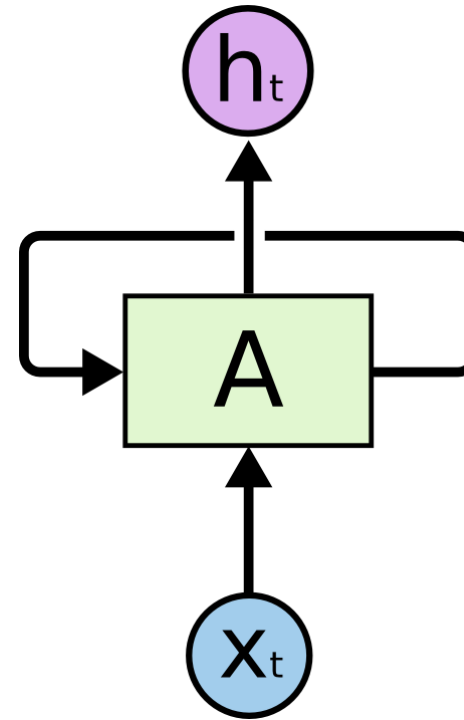
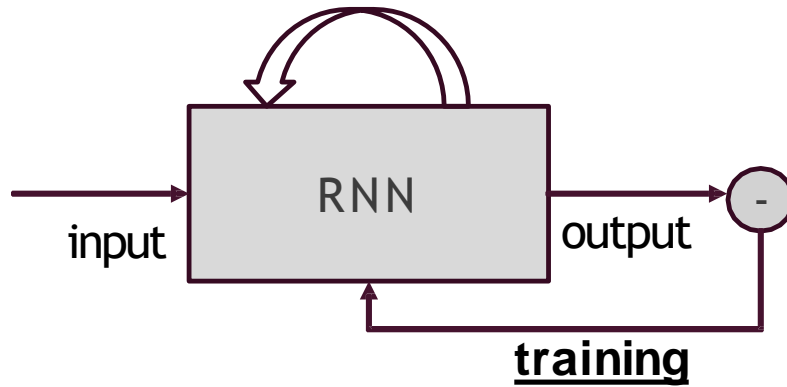
Sample RNN



Introduction

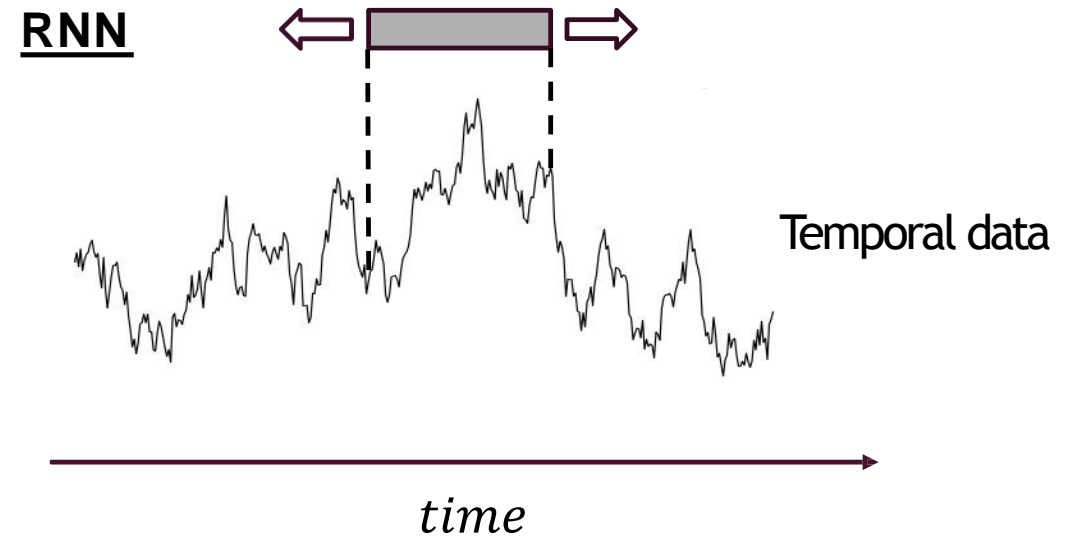
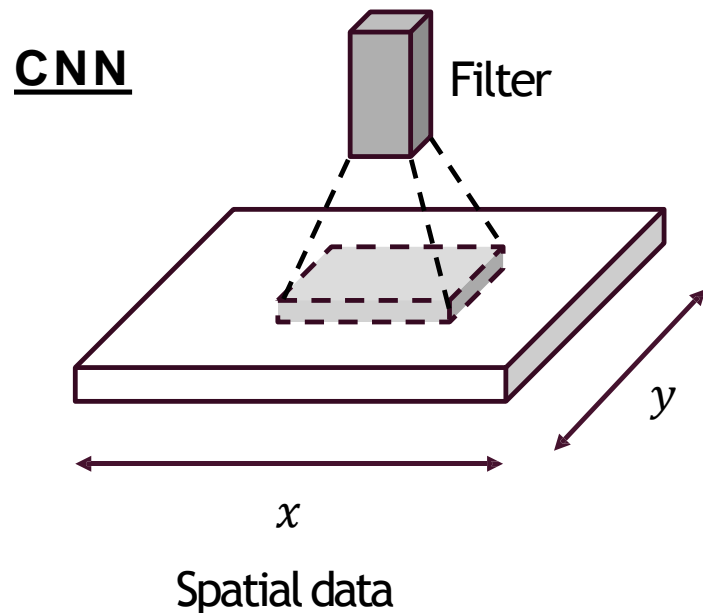
- RNN generalizes naturally to **new inputs** with any lengths.
- An RNN make use of **sequential information**, by modelling a **temporal dependencies** in the inputs.
 - **Example:** if you want to **predict the next word in a sentence** you need to know which words came before it
- The **output** of the network depends on the **current input** and on the value of the **previous internal state**.
- The **internal state** maintains a **(vanishing)** memory about **history** of all past inputs.
- RNNs can make use of information coming from **arbitrarily long** sequences, but in practice **they are limited to look back only a few time steps**.

- RNN can be trained to predict a **future value**, of the driving input.



DIFFERENCES WITH CNN

- Convolution in space (CNN) VS convolution in time (RNN) .
- CNN: models relationships in space. Filter slides along x and y dimensions.
- RNN: models relationships in time. Slides along time dimension.



Why RNN?

For example, imagine you want to classify what kind of event is happening at every point in a movie.

It's unclear how a traditional neural network could use its reasoning about previous events in the film to inform later ones.



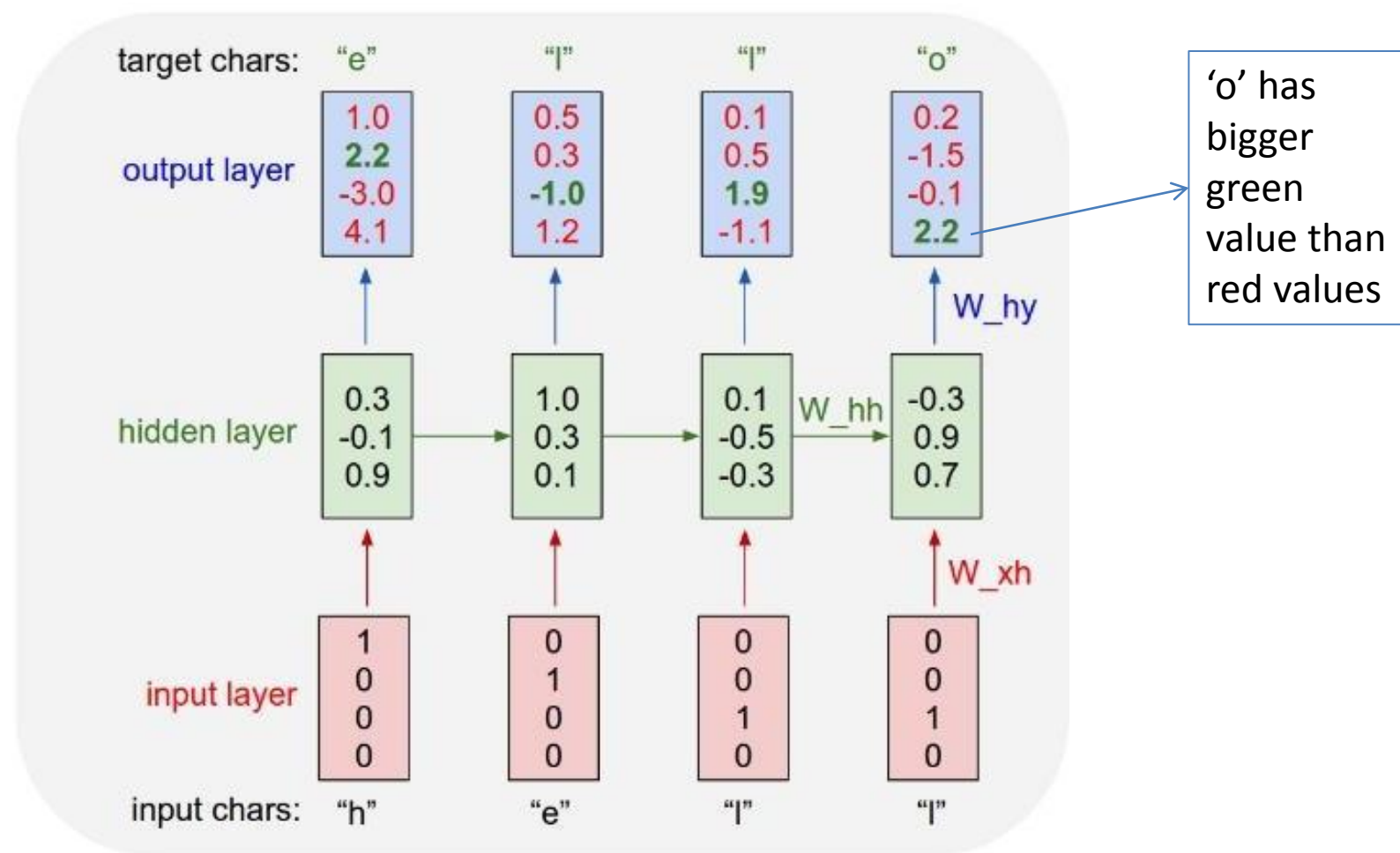
2. RNN APPLICATIONS



APPLICATION 1:

NATURAL LANGUAGE PROCESSING

- Given a sequence of words, RNN predicts the probability of next word given the previous ones.
- Input/output words are encoded as one-hot vector.
- We must provide the RNN all the dictionary of interest (usually, just the alphabet).
- In the output layer, we want the green numbers to be high and red numbers to be low.

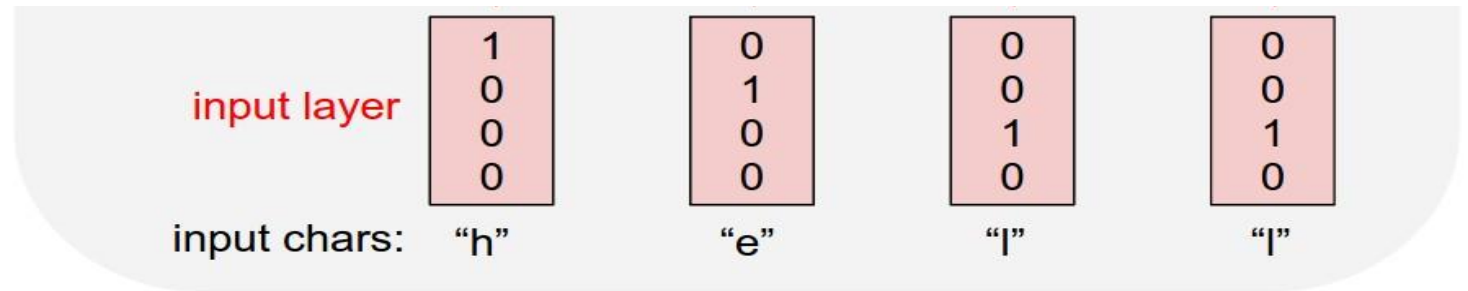


[Image:Andrej Karpathy]

Character-level language model example

Vocabulary:
[h,e,l,o]

Example training
sequence:
“hello”



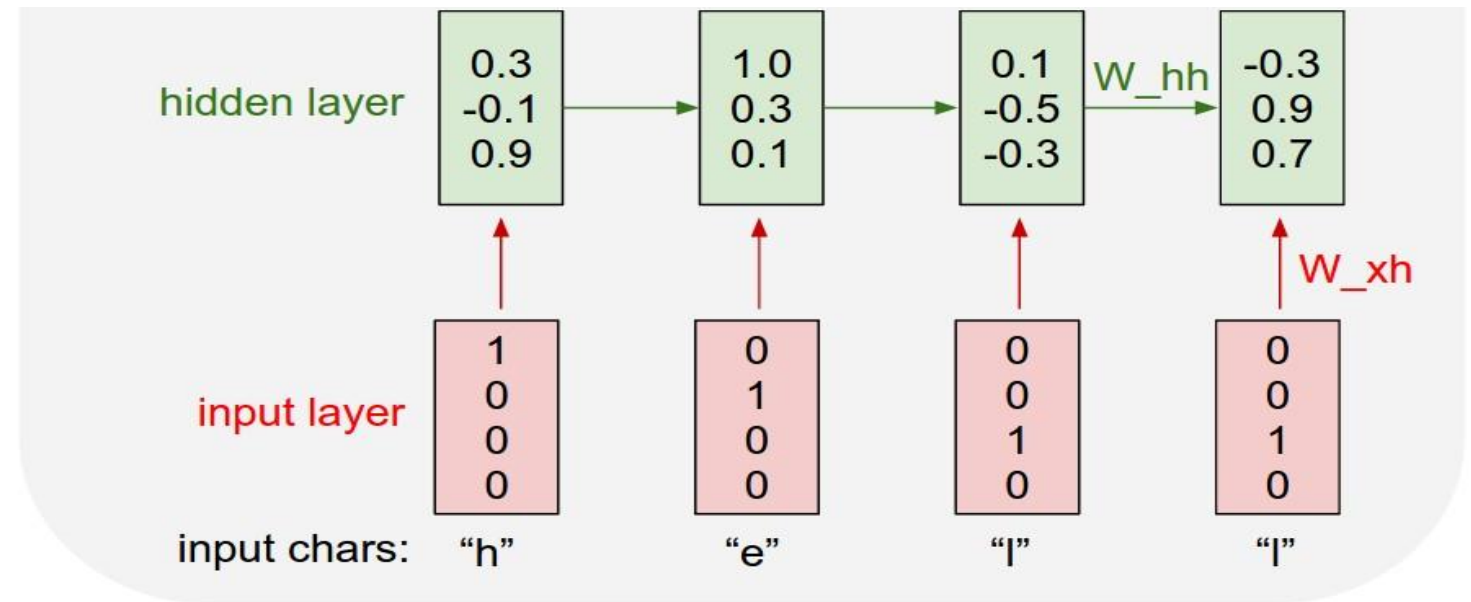
Character-level language model example

Vocabulary:

[h,e,l,o]

Example training
sequence:
“hello”

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



Character-level language model example

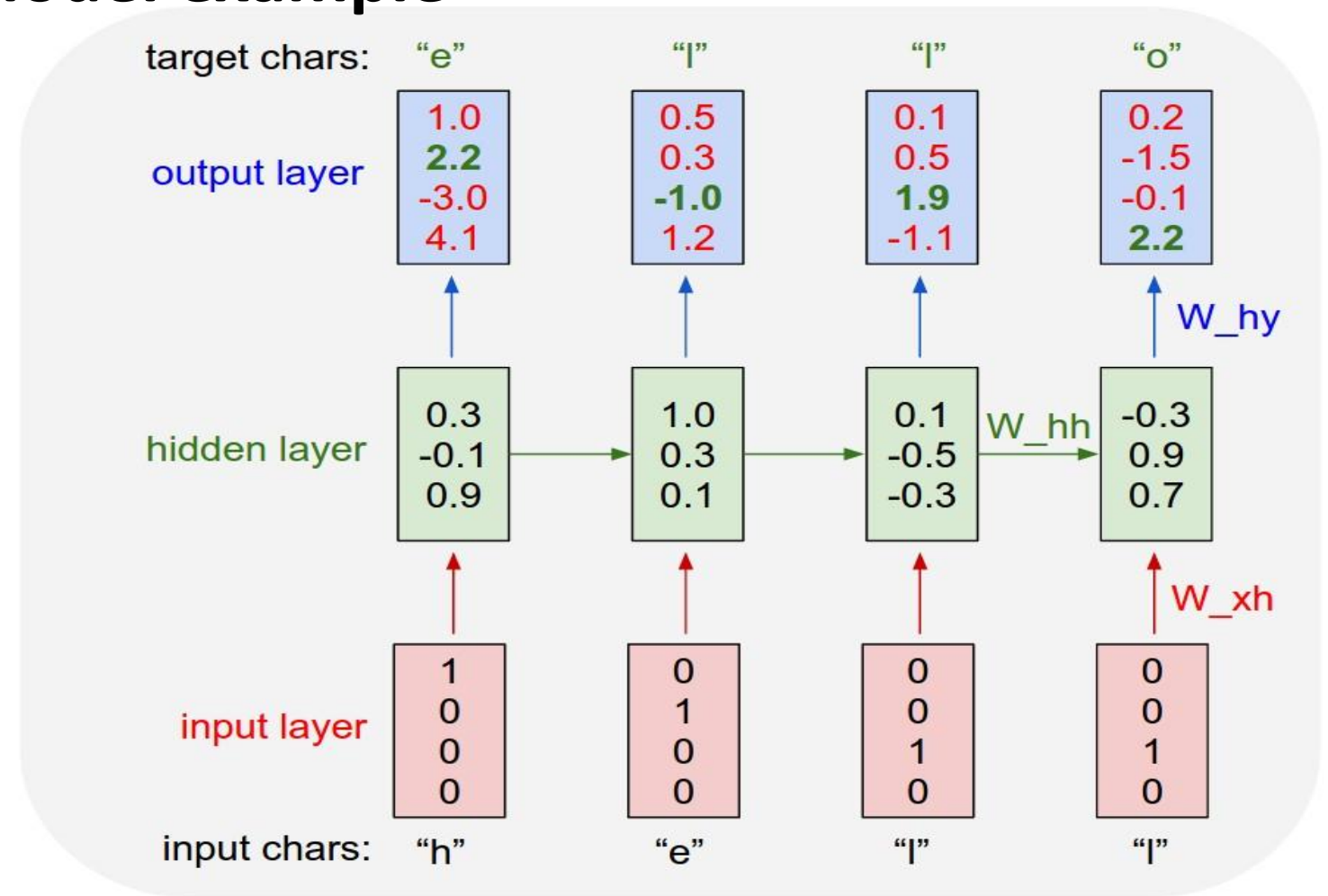
Vocabulary:

[h,e,l,o]

Example training

sequence:

“hello”



NATURAL LANGUAGE GENERATION: GENERATE SHAKESPEARE Poem

- Dataset: all the works of Shakespeare, concatenated them into a single (4.4MB) file.
- 3-layer RNN with 512 hidden nodes on each layer.
- Few hours of training.

VIOLA:

Why, Salisbury must find his flesh and thought
That which I am not apt, not a man and in fire,
To show the reining of the raven and the wars
To grace my hand reproach within, and not a fair are hand,
That Caesar and my goodly father's world;
When I was heaven of presence and our fleets,
We spare with hours, but cut thy council I am great,
Murdered and by thy master's ready there
My power to give thee but so much as hell:
Some service in the noble bondman here,
Would show him to her wine.

KING LEAR:

O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder'd at the deeds,
So drop upon your lordship's head, and your opinion
Shall be against your honour.

[Source:Andrej Karpathy]

TEXT GENERATION: WIKIPEDIA

- Hutter Prize 100MB dataset of raw Wikipedia.
- LSTM
- The link does not exist 😊

Naturalism and decision for the majority of Arab countries' capitalide was grounded by the Irish language by [[John Clair]], [[An Imperial Japanese Revolt]], associated with Guangzham's sovereignty. His generals were the powerful ruler of the Portugal in the [[Protestant Immineners]], which could be said to be directly in Cantonese Communication, which followed a ceremony and set inspired prison, training. The emperor travelled back to [[Antioch, Perth, October 25|21]] to note, the Kingdom of Costa Rica, unsuccessful fashioned the [[Thrales]], [[Cynth's Dajoard]], known in western [[Scotland]], near Italy to the conquest of India with the conflict. Copyright was the succession of independence in the slop of Syrian influence that was a famous German movement based on a more popular servicious, non-doctrinal and sexual power post. Many governments recognize the military housing of the [[Civil Liberalization and Infantry Resolution 265 National Party in Hungary]], that is sympathetic to be to the [[Punjab Resolution]] (PJS)[<http://www.humah.yahoo.com/guardian.cfm/7754800786d17551963s89.htm> Official economics Adjoint for the Nazism, Montgomery was swear to advance to the resources for those Socialism's rule, was starting to signing a major tripad of aid exile.]]

[Source:Andrej Karpathy]

TEXT GENERATION: SCIENTIFIC PAPER

- RNN trained on a book (LaTeX source code of 16MB).
- Multilayer LSTM

For $\bigoplus_{n=1,\dots,m}$ where $\mathcal{L}_{m\bullet} = 0$, hence we can find a closed subset \mathcal{H} in \mathcal{H} and any sets \mathcal{F} on X , U is a closed immersion of S , then $U \rightarrow T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \rightarrow V$. Consider the maps M along the set of points Sch_{fppf} and $U \rightarrow U$ is the fibre category of S in U in Section, ?? and the fact that any U affine, see Morphisms, Lemma ???. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \rightarrow S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S . We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \rightarrow \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. \square

To prove study we see that $\mathcal{F}|_U$ is a covering of \mathcal{X}' , and \mathcal{T}_i is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and \mathcal{F}_p exists and let \mathcal{F}_i be a presheaf of \mathcal{O}_X -modules on \mathcal{C} as a \mathcal{F} -module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1} \mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

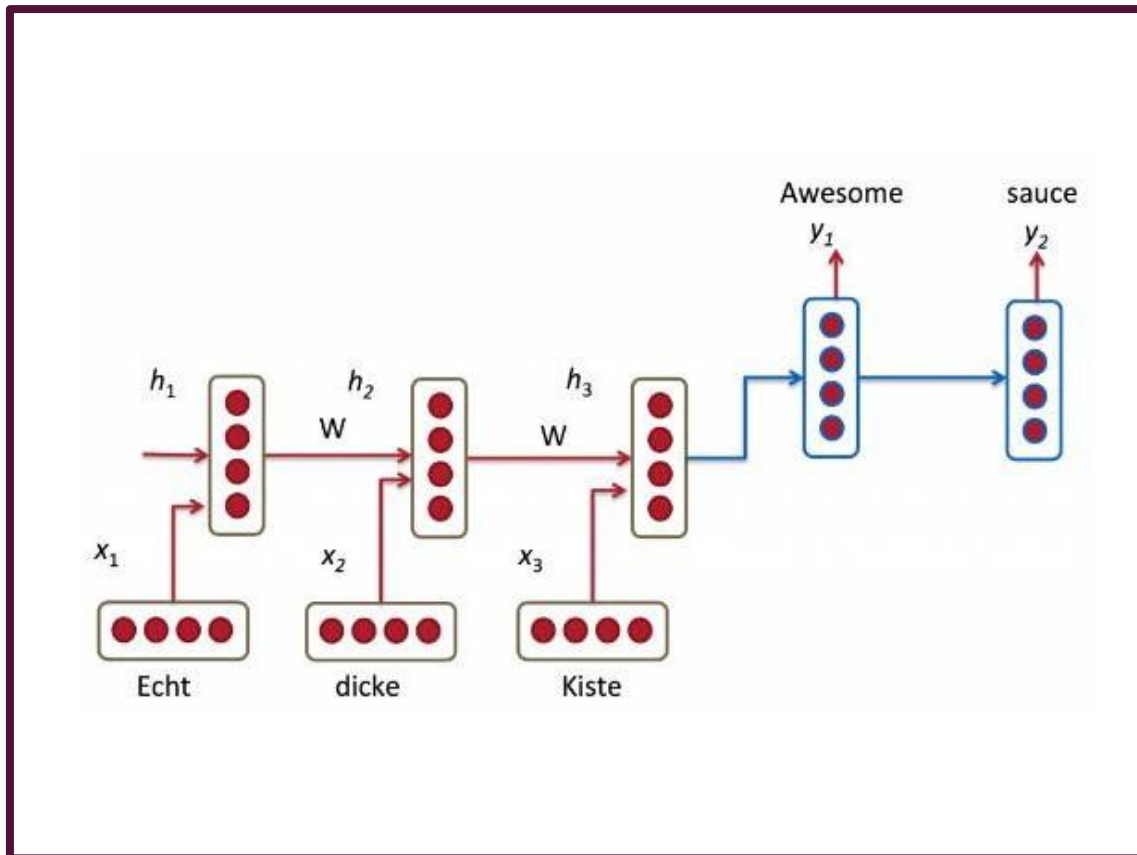
$$\mathrm{Arrows} = (Sch/S)_{fppf}^{opp}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is an open subset of X . Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S .

APPLICATION II: MACHINE TRANSLATION

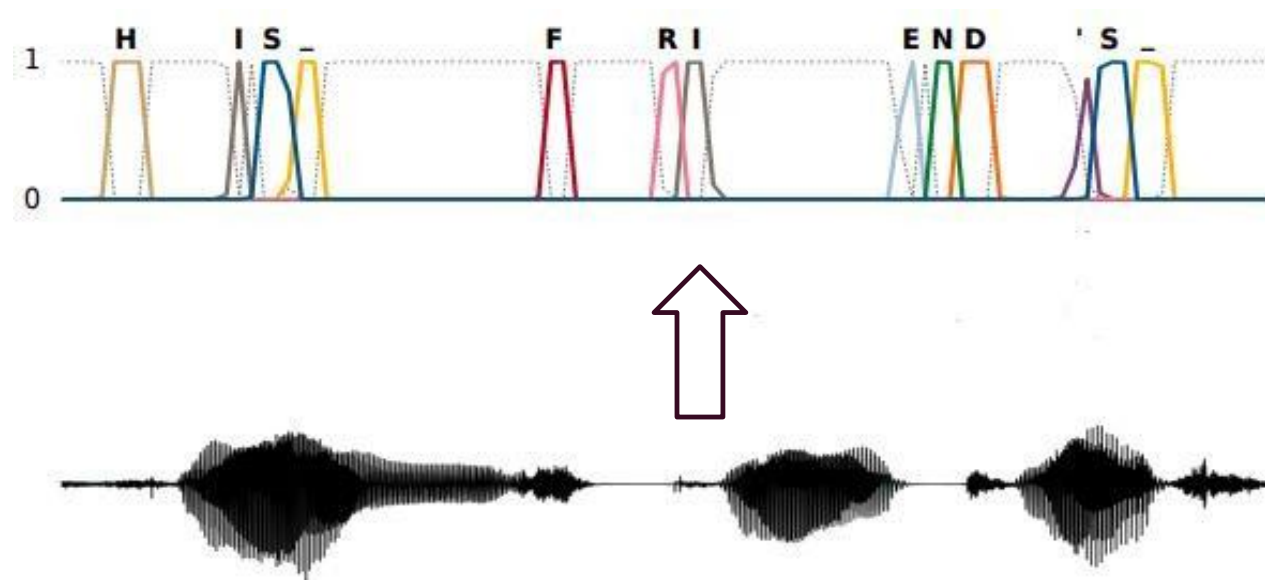


[Image: Richard Socher]

- Similar to language modeling.
- Train 2 different RNNs.
- **Input RNN**: trained on a source language (e.g. German).
- **Output RNN**: trained on a target language (e.g. English).
- The **second** RNN computes the output from the hidden layer of the **first** RNN.
- **Google translator**.

APPLICATION III: SPEECH RECOGNITION

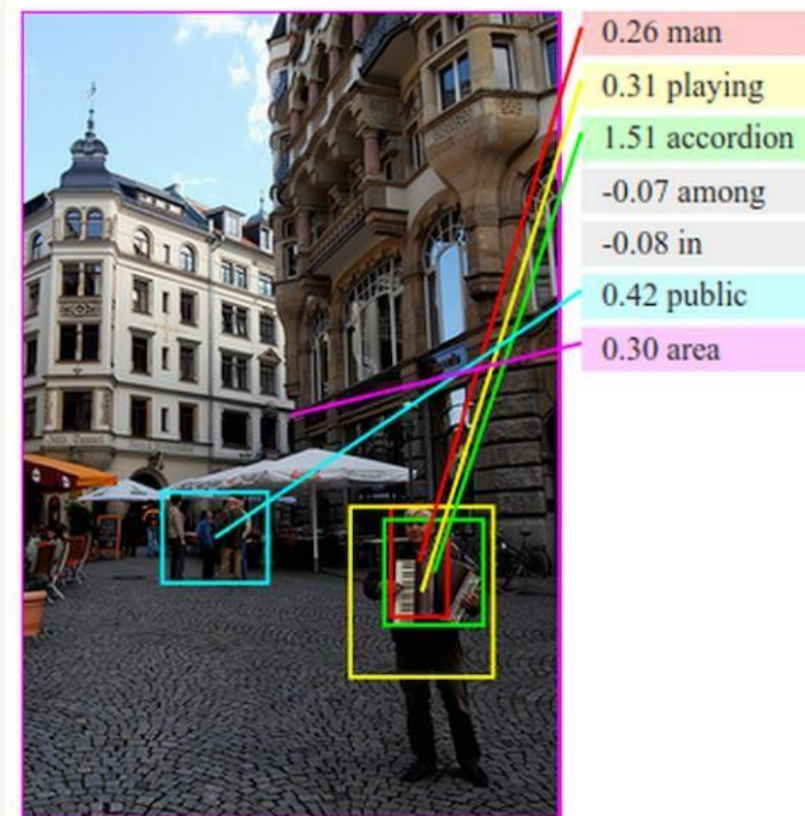
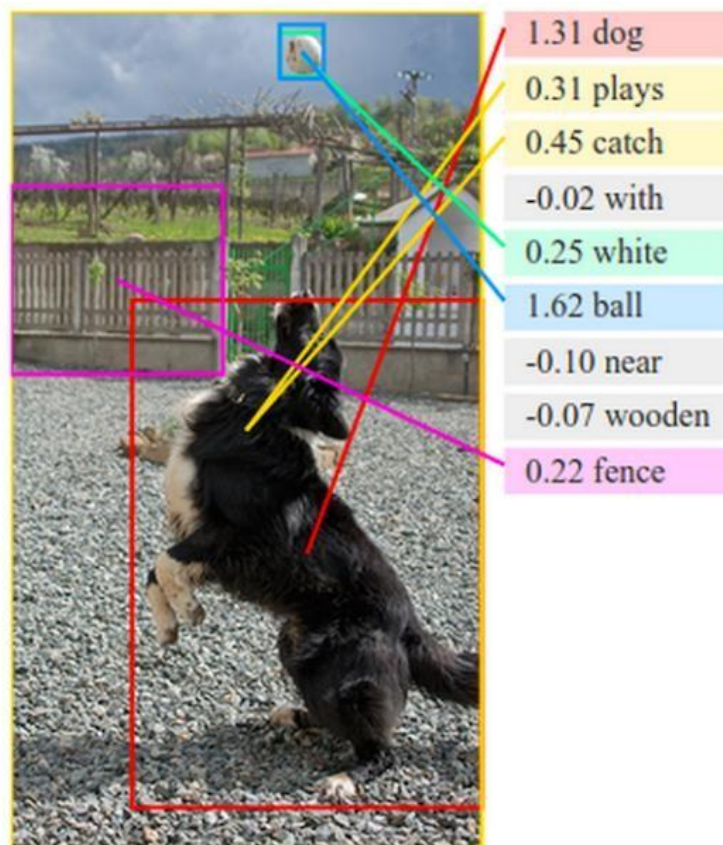
- **Input:** input sequence of acoustic signals.
- **Output:** phonetic features.
- Necessity of encoder/decoder to transit from digital/analogic domain.
- Graves, Alex, and Navdeep Jaitly. "Towards End-To-End Speech Recognition with Recurrent Neural Networks.", 2014.



Each speech sound can be analyzed in terms of its **phonetic** features.

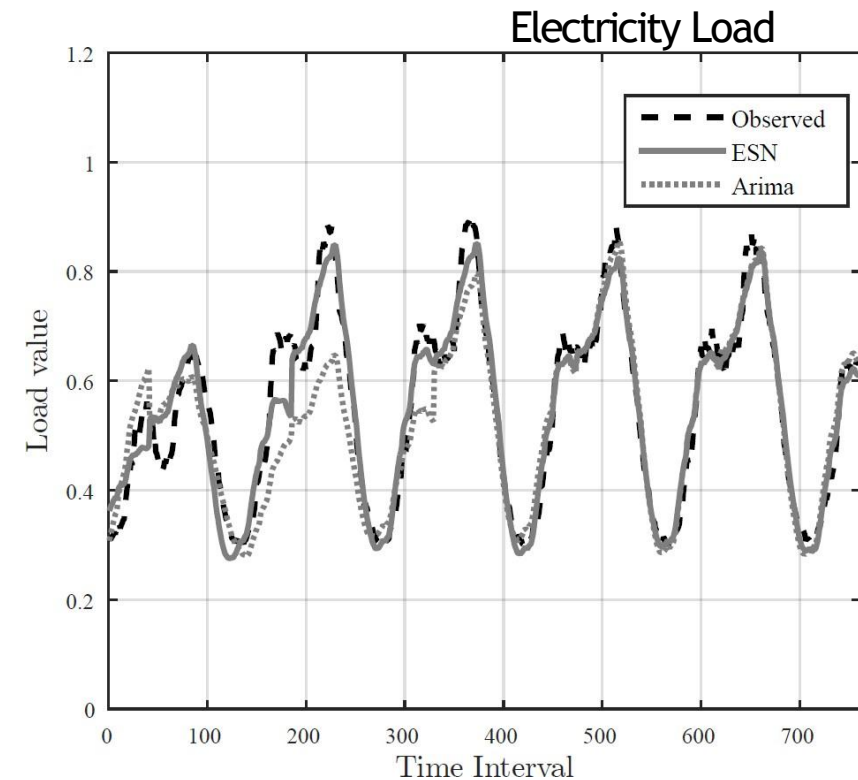
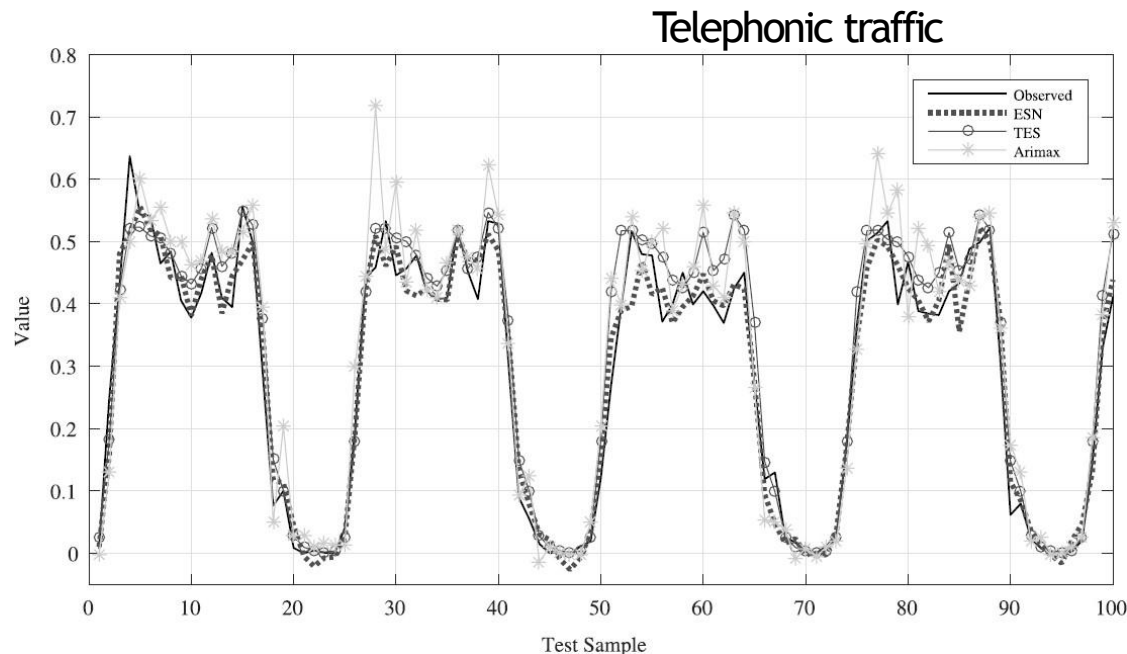
APPLICATION IV: IMAGE TAGGING

- **RNN + CNN** jointly trained.
- **CNN generates features** (hidden state representation).
- **RNN reads CNN features and produces output** (end-to-end training).
- Aligns the generated words with features found in the images
- *Karpathy, Andrej, and Li Fei-Fei. "Deep visual-semantic alignments for generating image descriptions.", 2015.*



APPLICATION V: TIME SERIES PREDICTION

- Forecast of future values in a time series, from past seen values.
- Many applications:
 - Weather forecast.
 - Load forecast.
 - Financial time series.

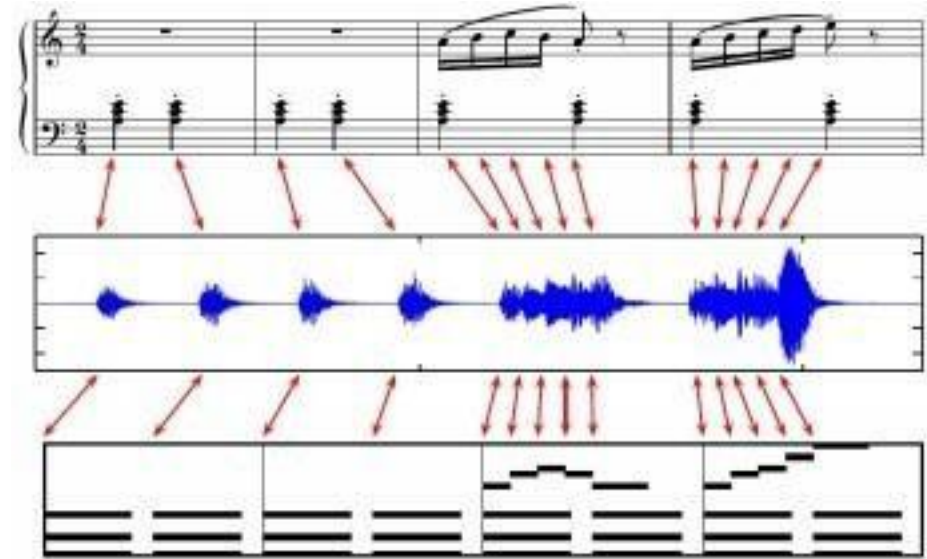


APPLICATION VI: MUSIC INFORMATION RETRIEVAL

○ Identification of songs/music

- Automatic categorization.
- Recommender systems.
- Track separation and instrument recognition.
- Music generation.

Music transcription example



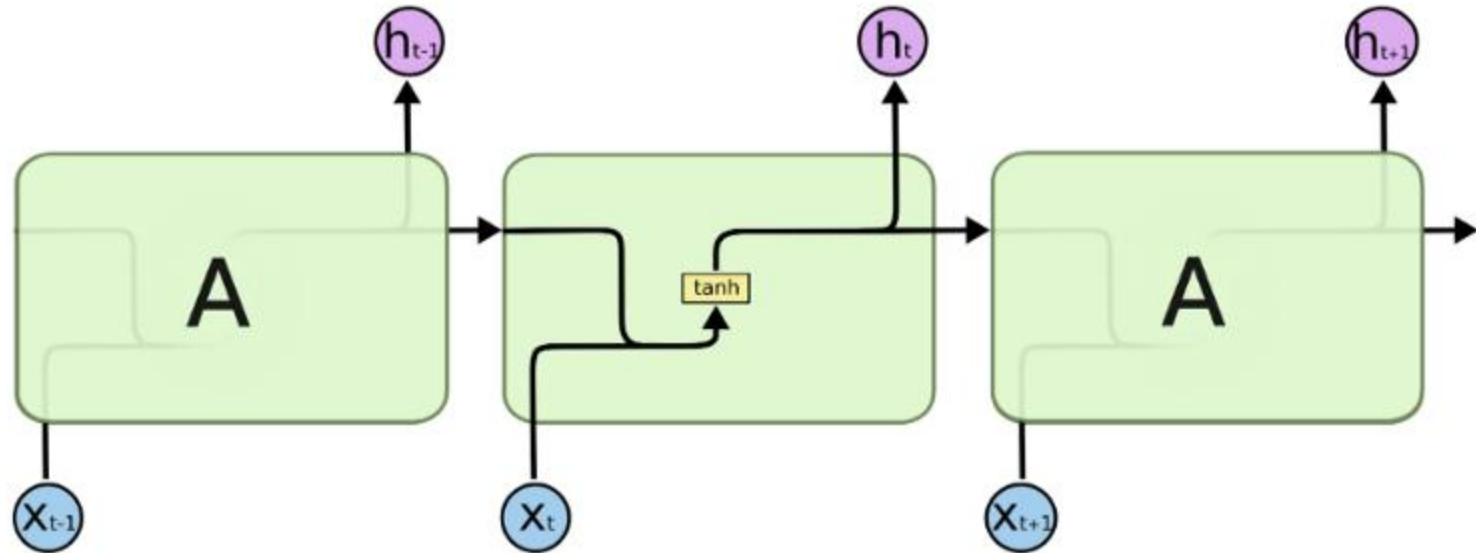
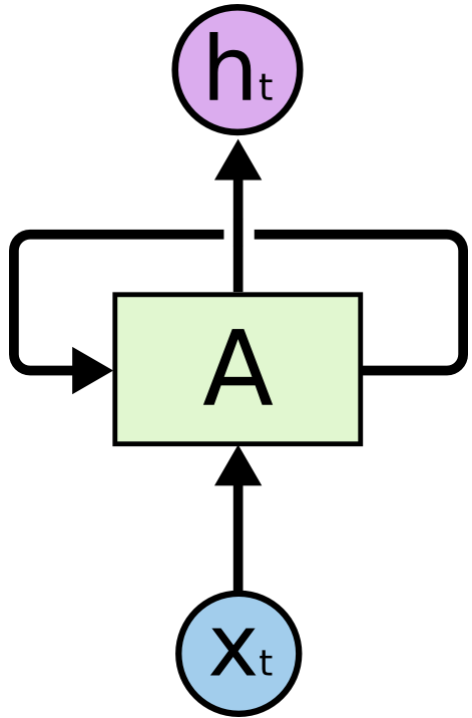
[Source: Meinard Müller]





3. Understanding R N N

RNN has loops in it and allows to persist information.

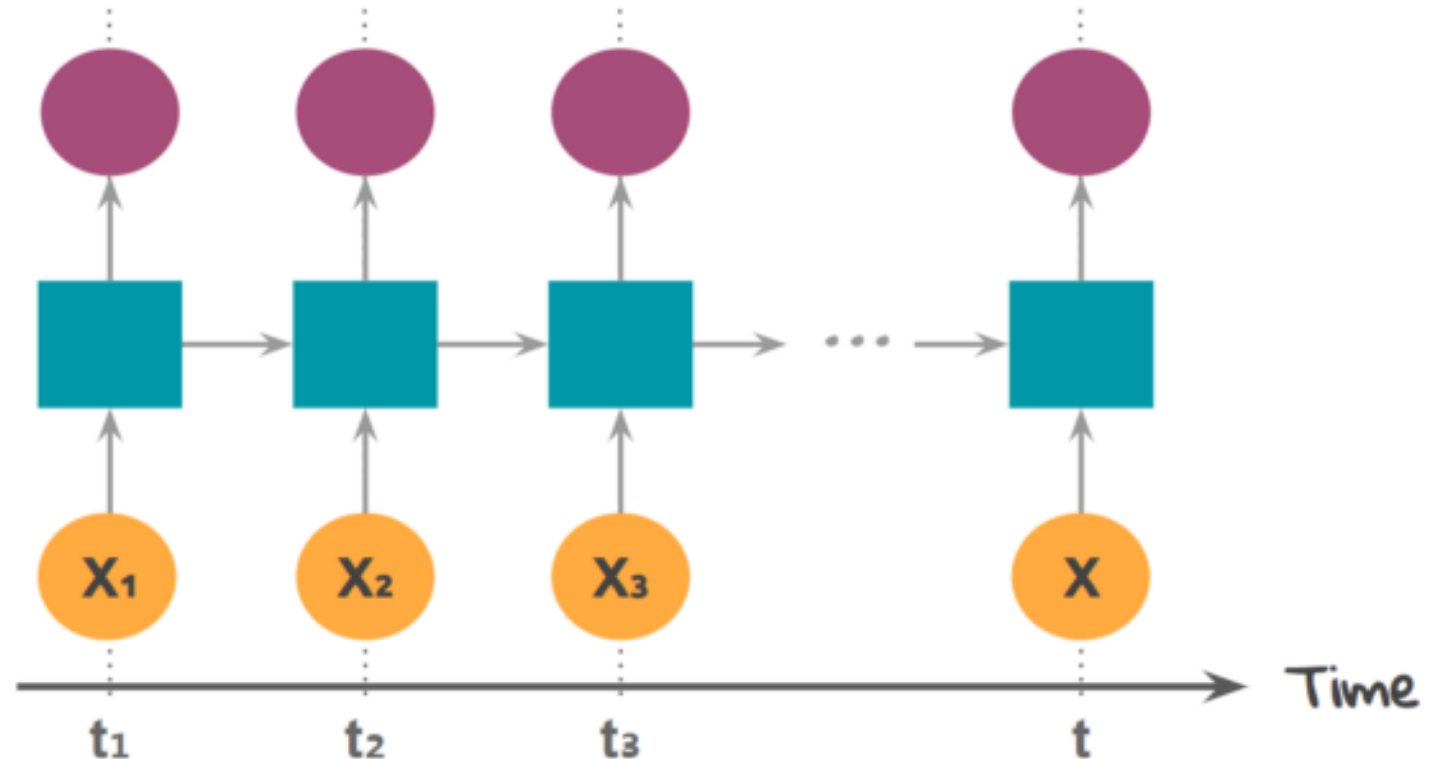


The repeating module in a standard RNN contains a single layer.

A chunk of neural network, A , looks at some input x_t and outputs a value h_t .
A loop allows information to be passed from one step of the network to the next.

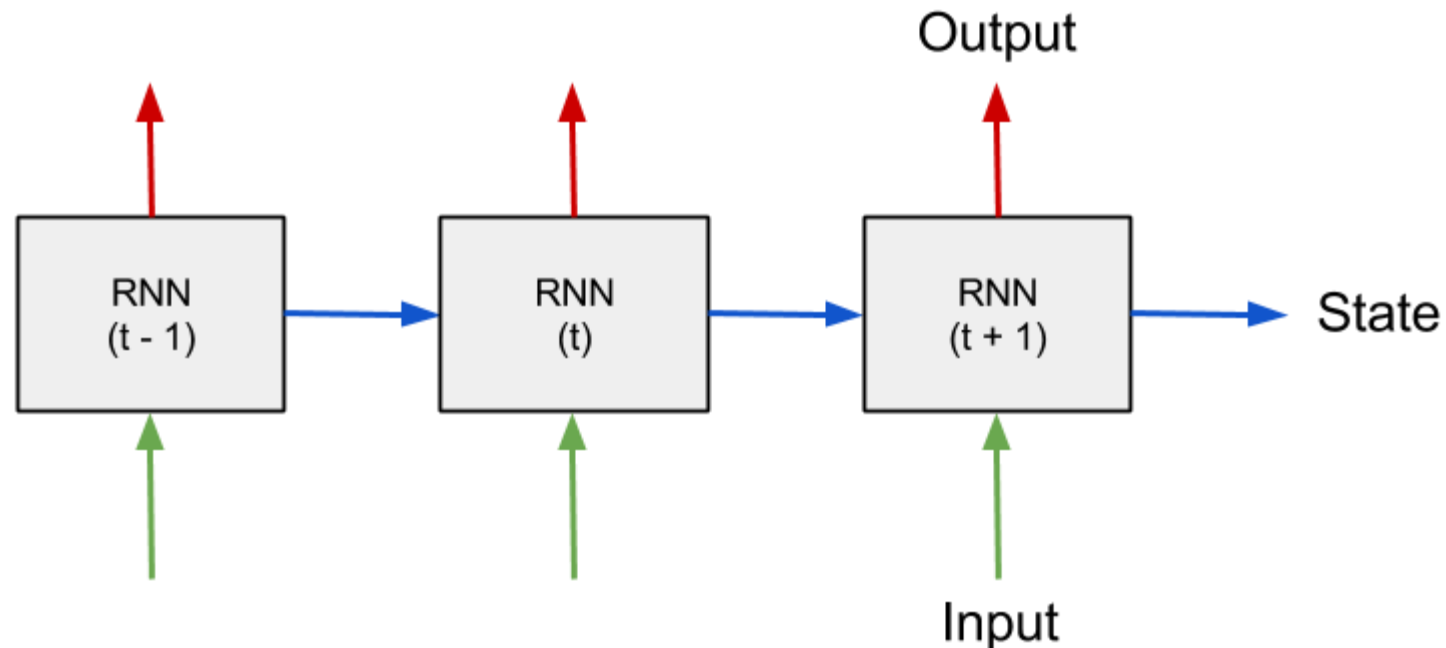
In the case of RNN, the data aren't inputted at the same time. As you can see the picture on the right, we will input **X1** first and then input **X2** to the result of **X1** computation. So in the same way, **X3** is computed with the result from **X2** computation stage.

- **Sequential** means we have an order in time between the data.
- if we change the order, it becomes significantly different.
A sentence will lose its meaning.



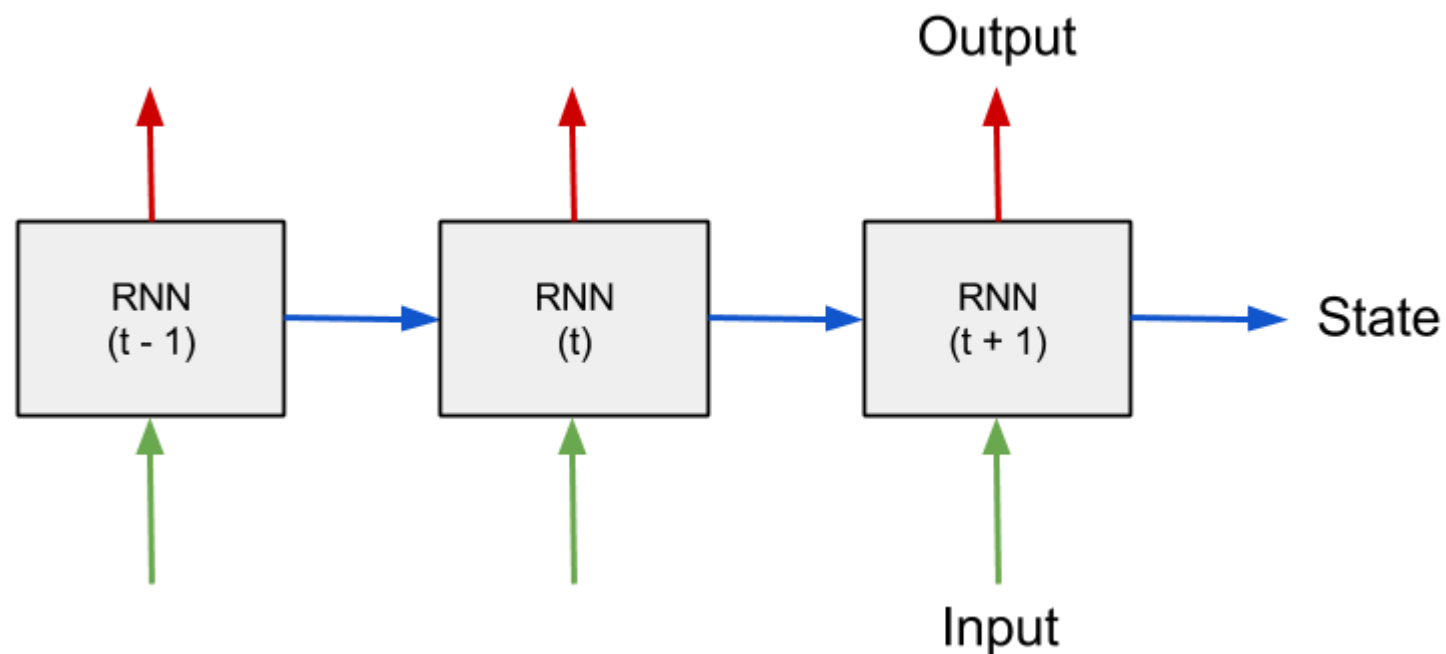
RNN...

- In traditional Networks, all inputs are independent of each other.
- RNN will make use of sequential information.
- If you want to predict the next word in a sentence you better know which words came before it (A sequence of words → Predict the next word).

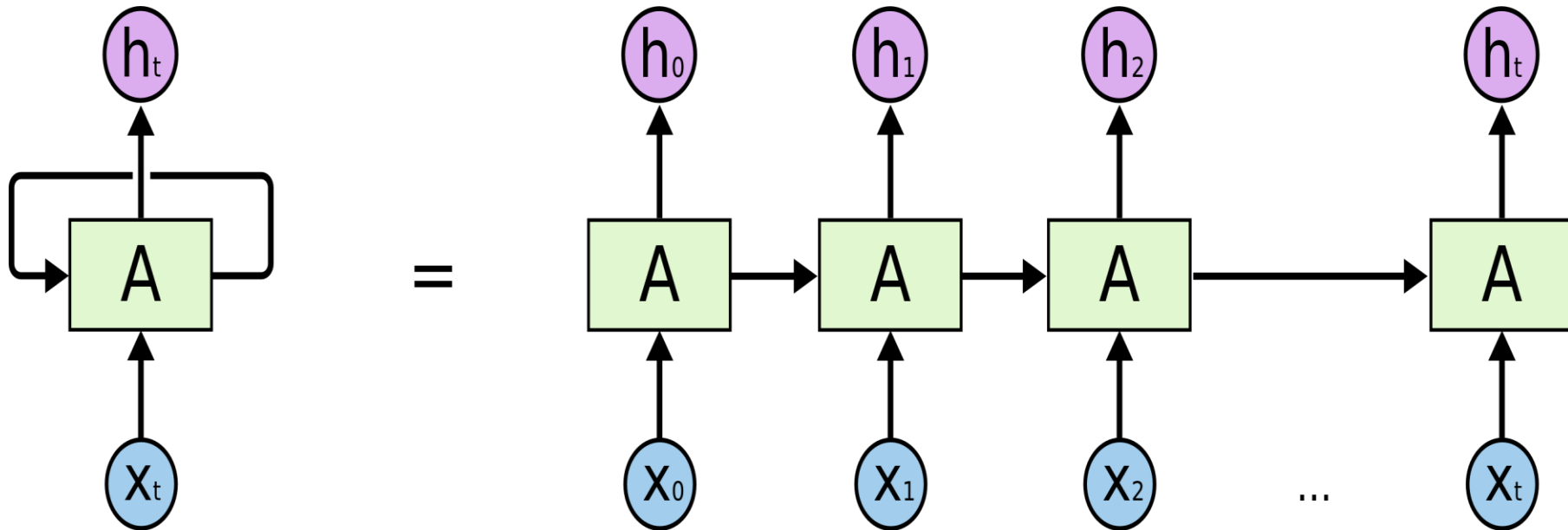


RNN...

- The most straight-forward example is perhaps **a time-series of numbers**, where the task is to **predict the next value** given previous values.
- The input to the RNN at every time-step is the *current value as well as a state vector* which represent what the network has “seen” at time-steps before.
- This state-vector is the encoded memory of the RNN, initially set to zero.




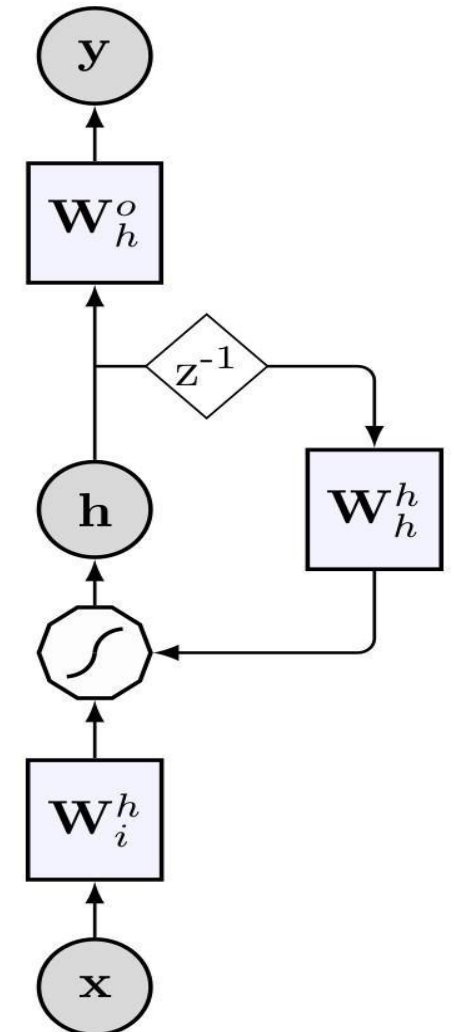
- A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor.



ARCHITECTURE COMPONENTS

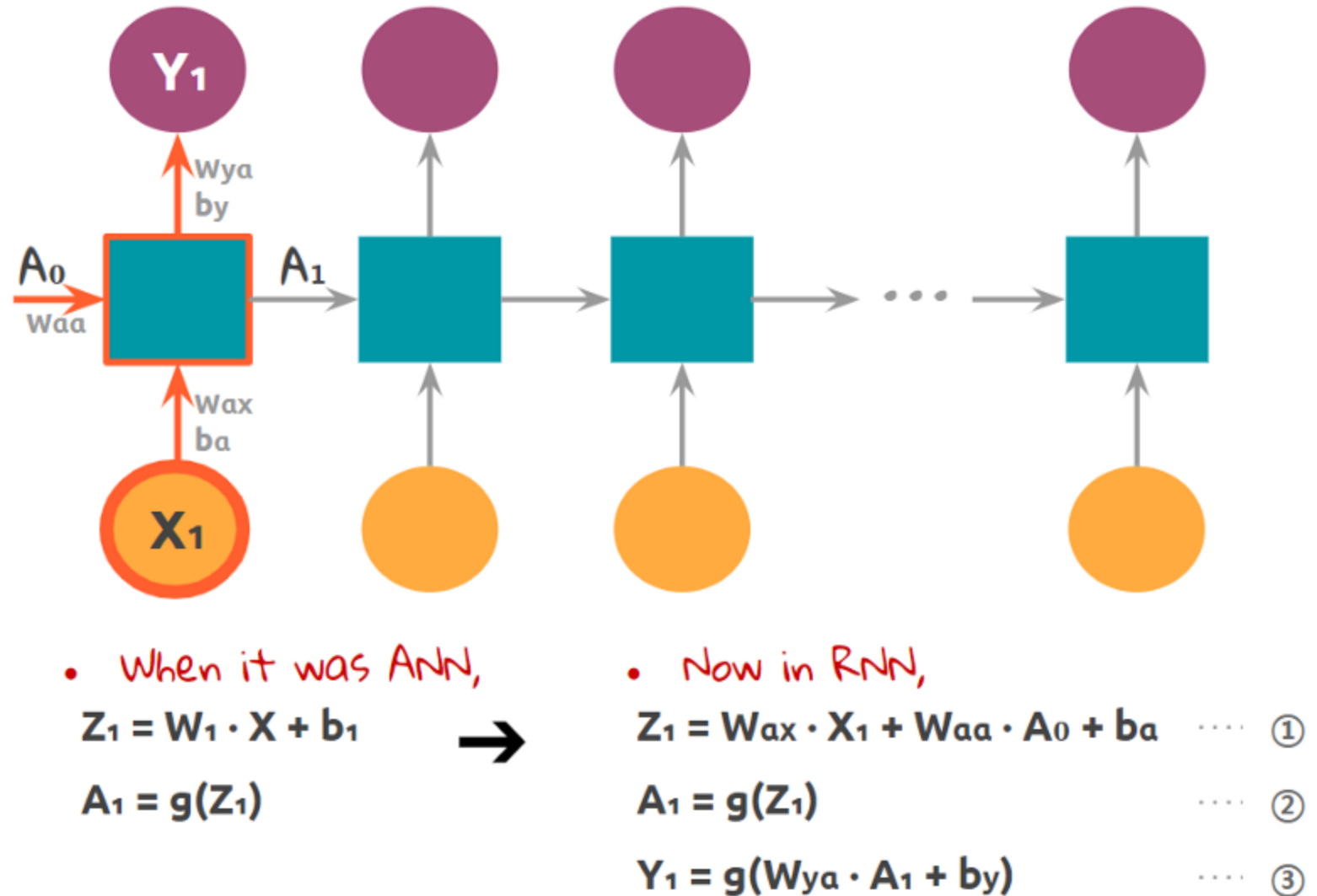
- x : input
- y : output
- h : internal state (memory of the network)
- W_i^h : input weights

- W_h^h : recurrent layer weights
- W_h^o : output weights
- z^{-1} : time-delay unit
-  : Activation function

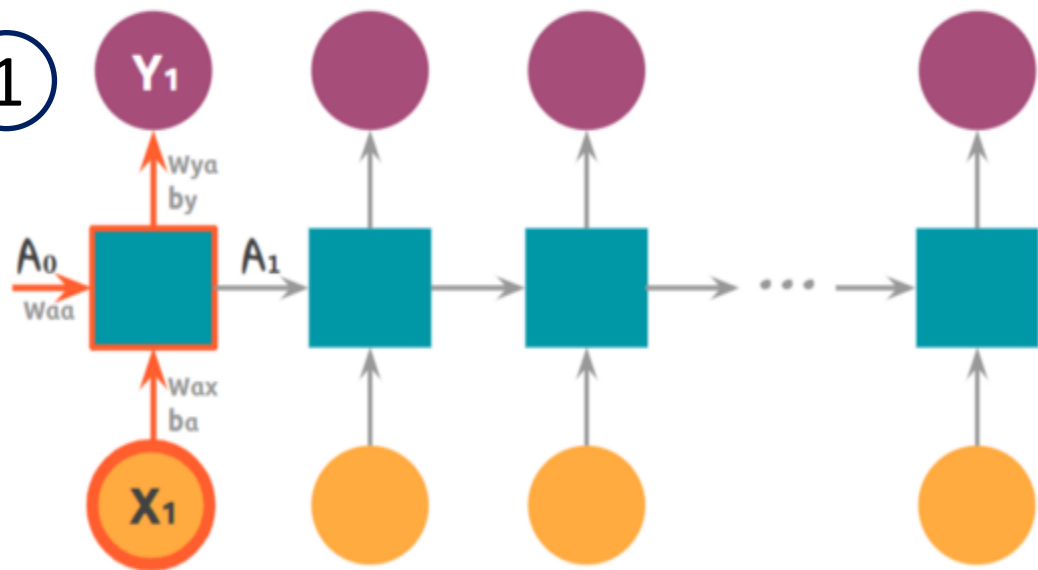


ARCHITECTURE COMPONENTS...

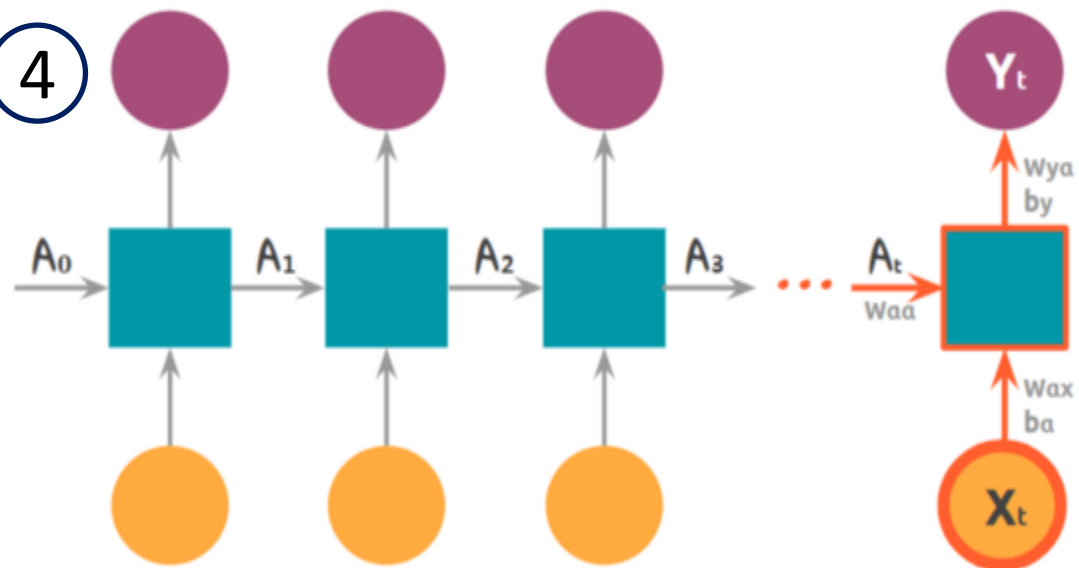
- Equation ① - There are weights and bias as we did with simple NN. It's just adding one more input value **A0**.
- And there are two different outputs from the cell. The output of **A1** which will go to the next unit(②) and the final output **Y1** of the unit cell(③).



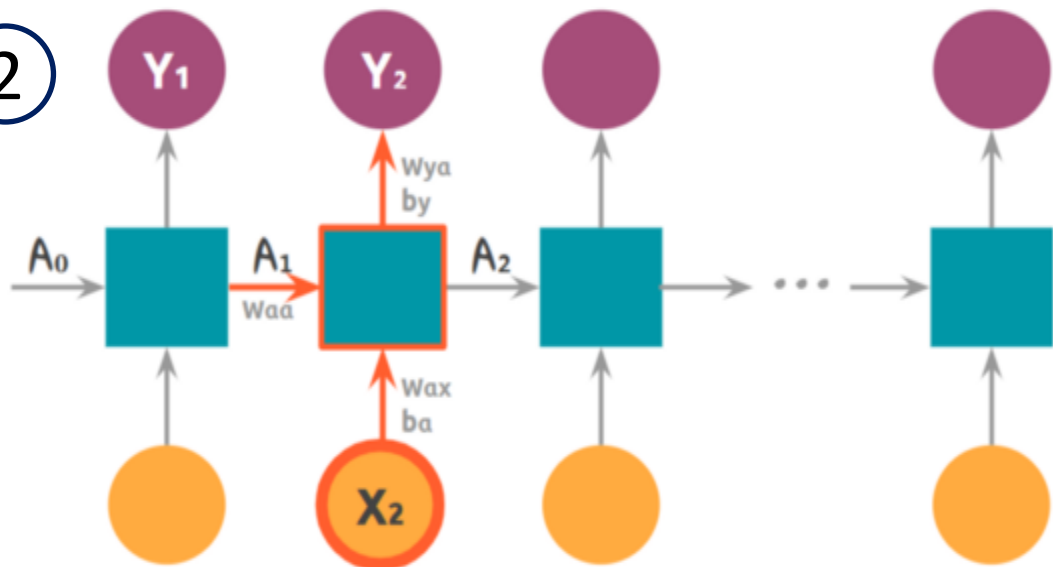
1



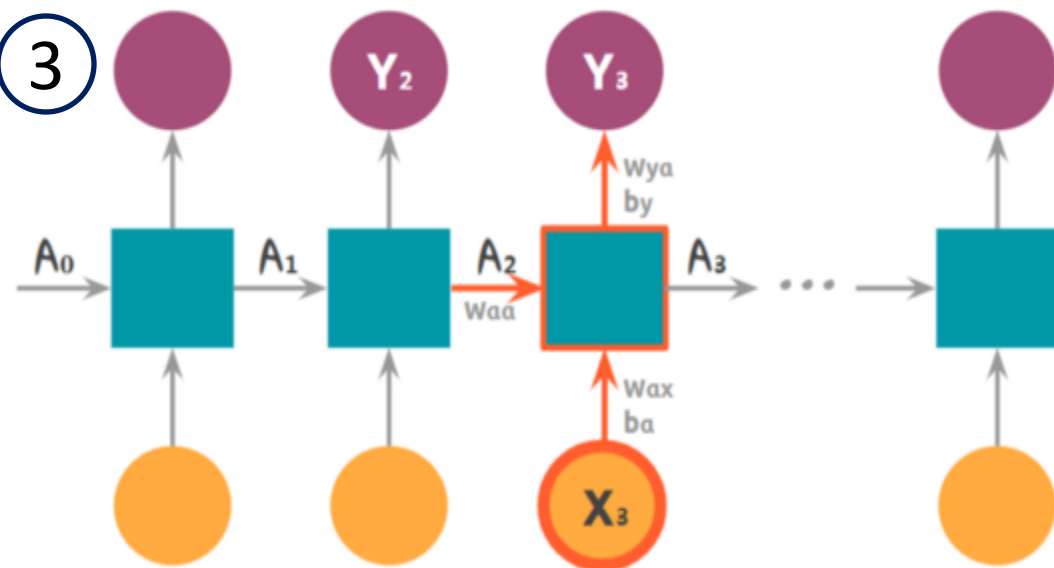
4



2



3

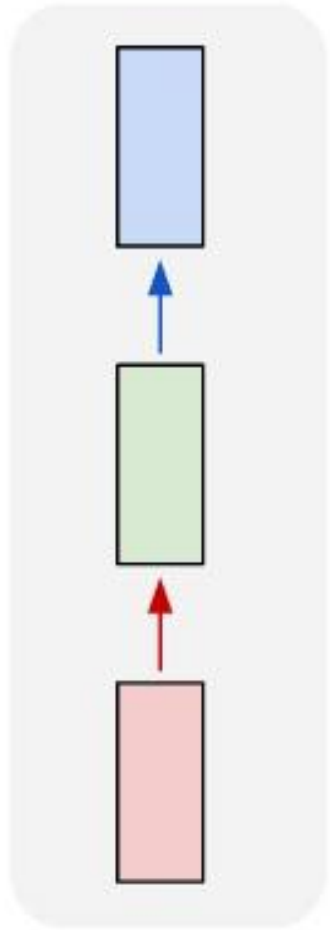




Various types of RNN

RNN - Various types

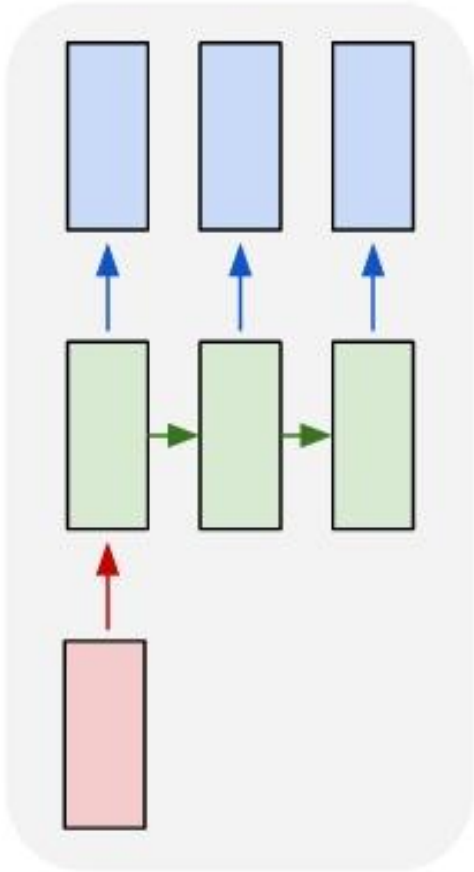
one to one



(1) Processing without RNN, from fixed-sized input to fixed-sized output

Example: Image classification

one to many



(2) Sequence output

Example: Image captioning

Takes an image and outputs a sentence of words.

Single Image → Sequence of words (Caption)

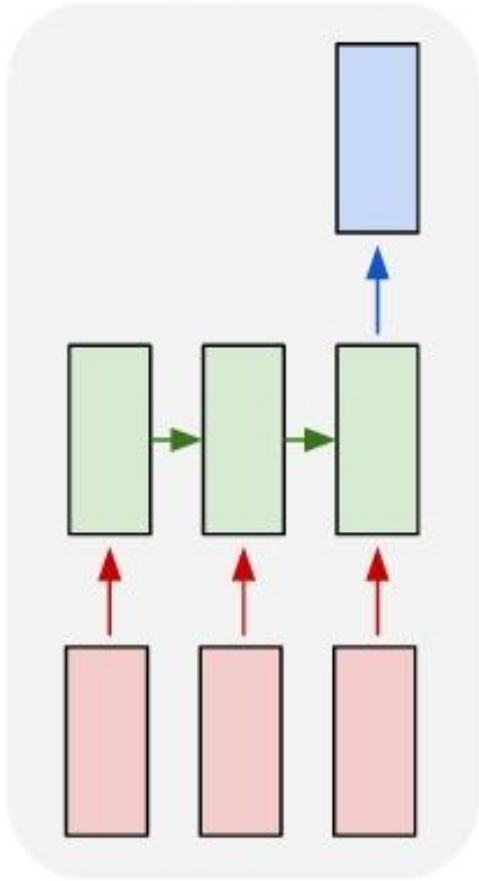


You have beautiful eyes!



IMAGE CAPTIONING

many to one



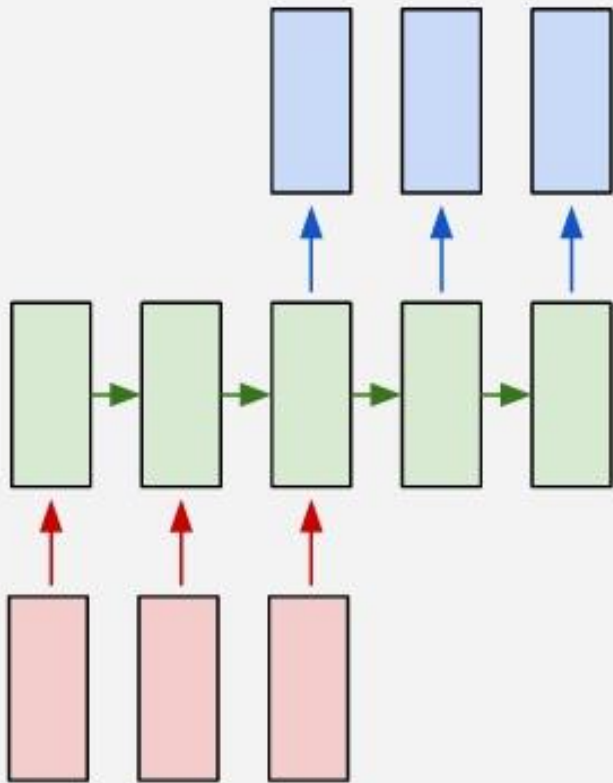
(3) Sequence of input to a single output

Example: Sentiment Prediction

Takes a sequence of words (Sentence) and outputs a word (Sentiment).

Sentence (sequence of words) → Sentiment (word)

many to many



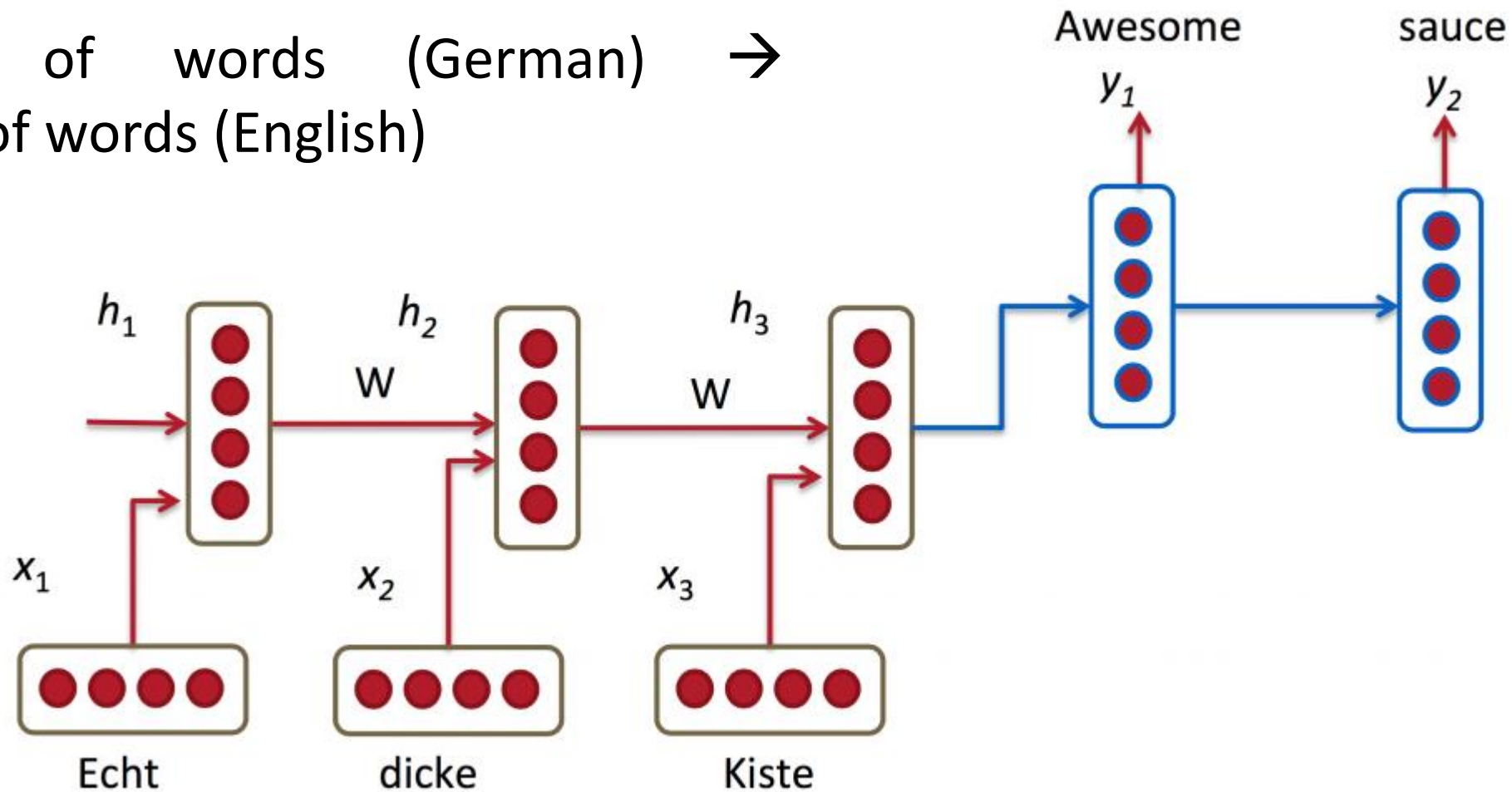
(4) Sequence input and sequence output

Example: Machine Translation

An RNN reads a sentence in English and then outputs a sentence in French.

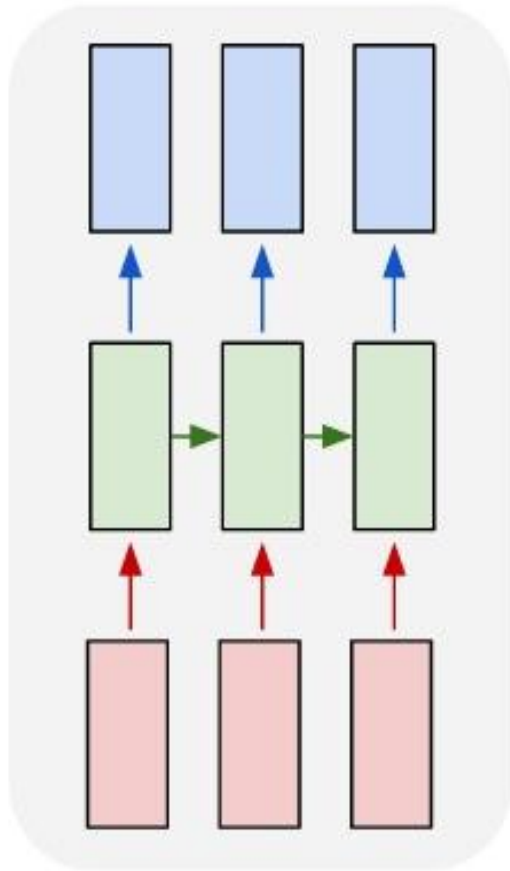
RNN - Machine translation

Sequence of words (German) →
Sequence of words (English)



output only starts after we have seen the complete input, because the first word of our translated sentences may require information captured from the complete input sequence.

many to many



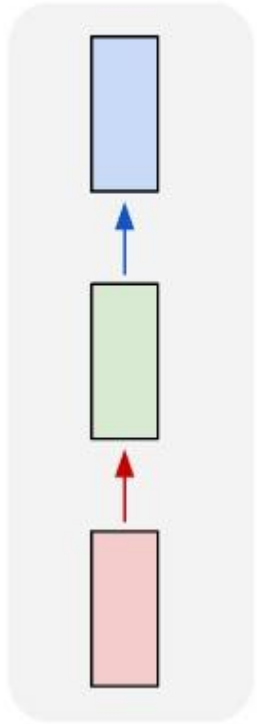
(5) Synced Sequence input and output

Example: Video Classification

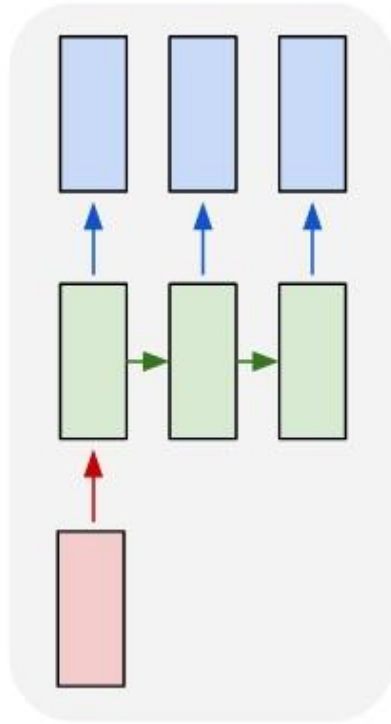
Label each frame of the video

RNN - Various types

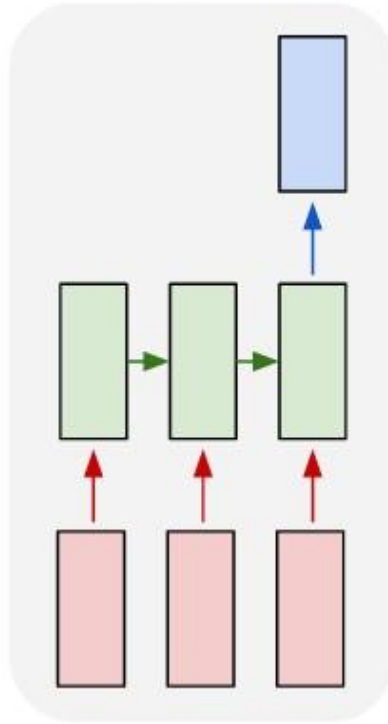
one to one



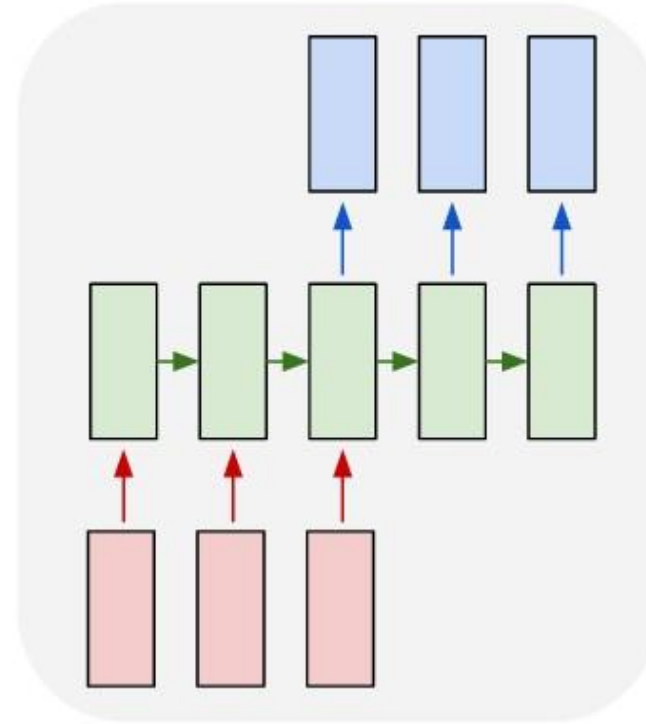
one to many



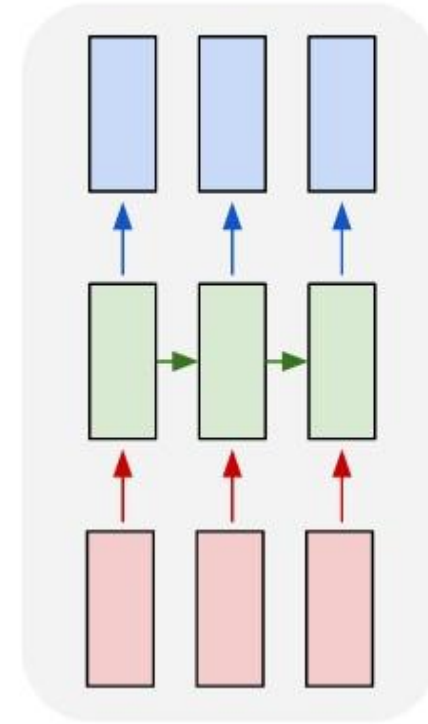
many to one



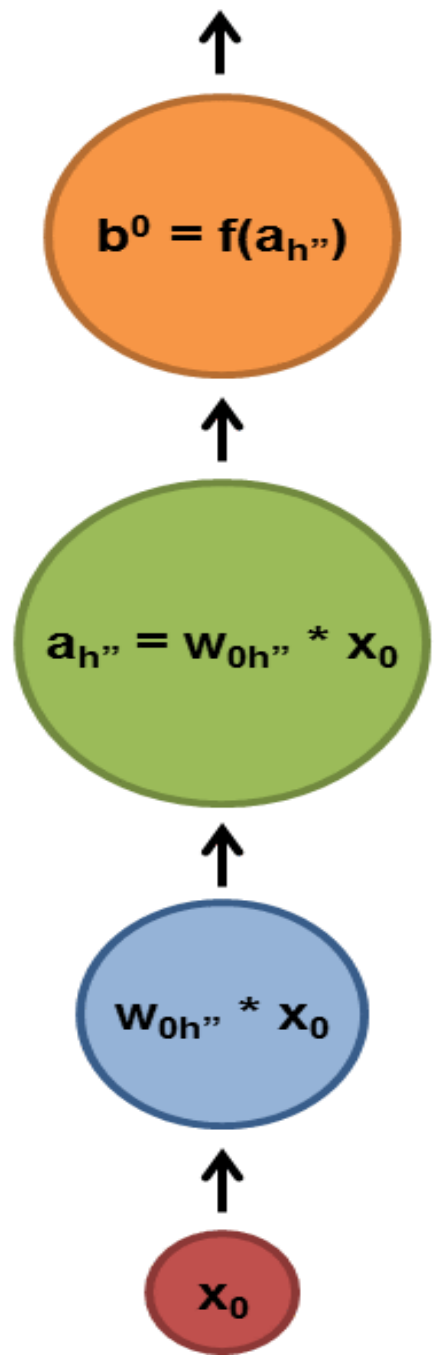
many to many



many to many



b^0 is fed to next layer



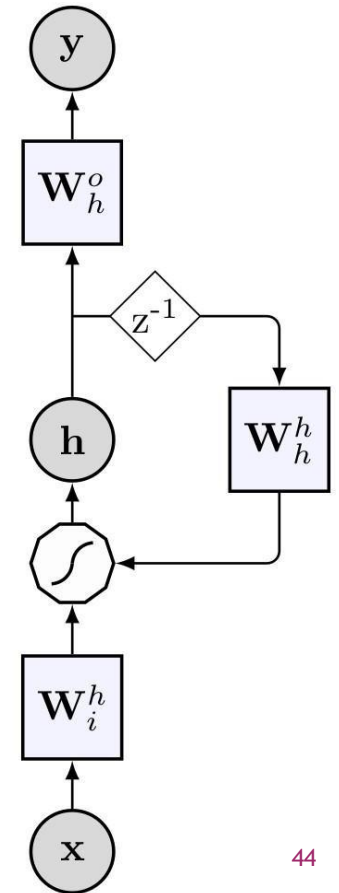
x - input
w - weights
a - activation
b - output



How RNN Learns?

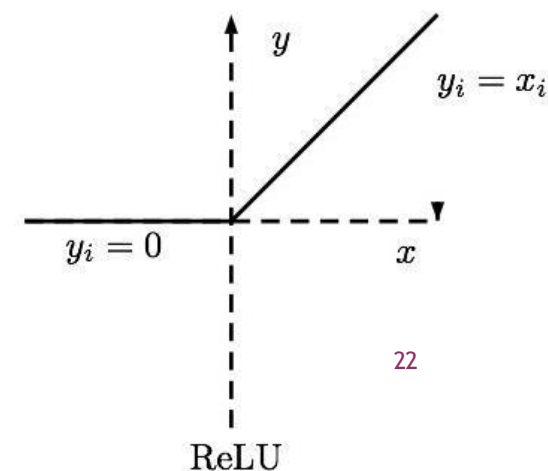
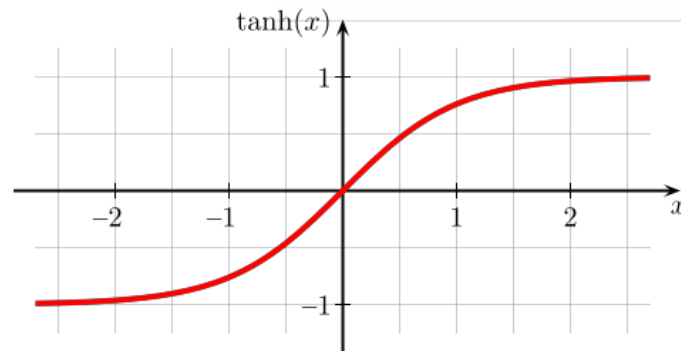
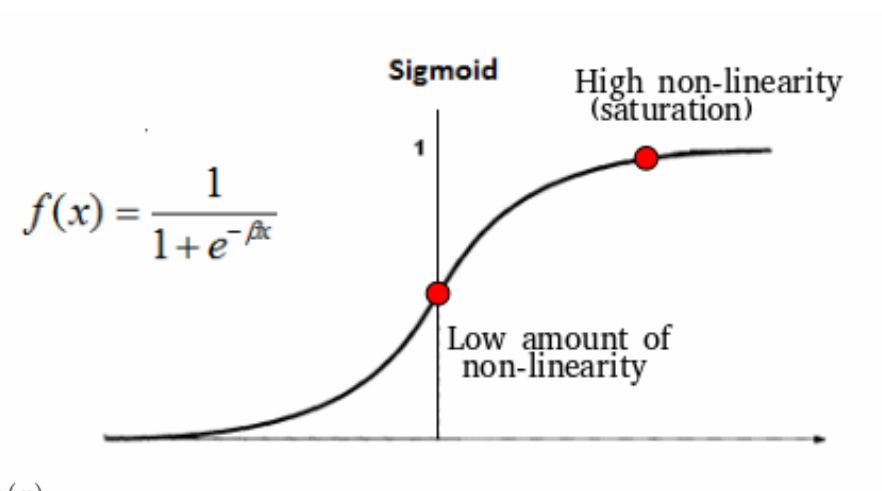
STATE UPDATE AND OUTPUT GENERATION

- An RNN selectively **summarize an input sequence** in a fixed-size state vector via a **recursive update**.
- Discrete, time-independent **difference equations** of RNN state and output:
$$h[t + 1] = f(W_h^h h[t] + W^h x[t + 1] + b_h),$$
$$y[t + 1] = g(W_h^o h[t + 1] + b_o).$$
- $f()$ is the **transfer function** implemented by each neuron (usually the same non-linear function for all neurons).
- $g()$ is the **readout** of the RNN. Usually is the identity function - all the non-linearity is provided by the internal processing units (neurons) - or the **softmax function**.



NEURON TRANSFER FUNCTION

- The activation function in a RNN is traditionally implemented by a sigmoid.
 - Saturation causes vanishing gradient.
 - Non-zero centering produces only positive outputs, which lead to zig-zagging dynamics in the gradient updates.
- Another common choice is the tanh.
 - Saturation causes vanishing gradient.
- ReLU
 - Greatly accelerate gradient convergence and it has low demanding computational time.
 - No vanishing gradient.
 - Large gradient flowing through a ReLU neuron could cause the its “death”.

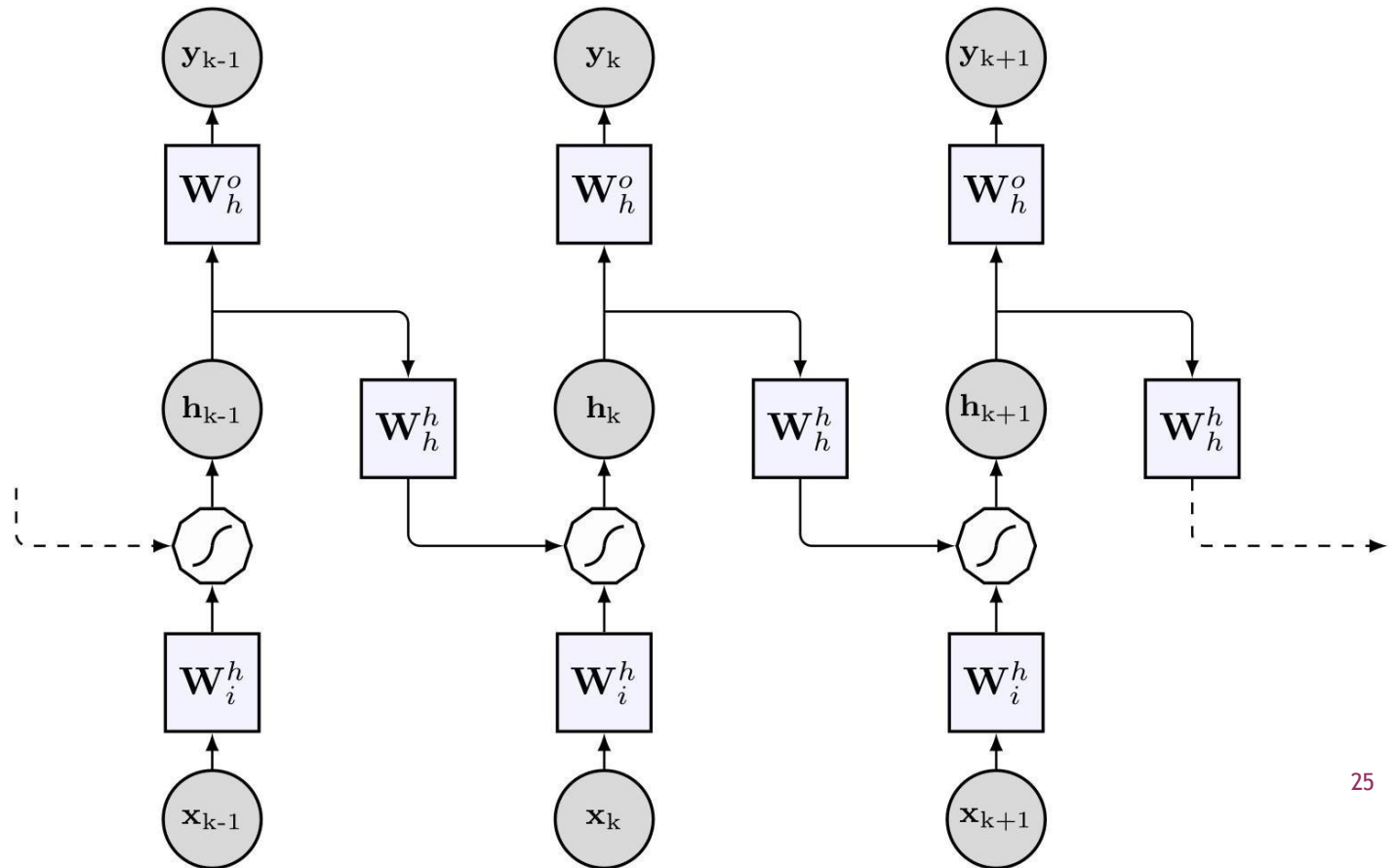


TRAINING

- Model's parameters are trained with gradient descent.
- A loss function is evaluated on the error performed by the network on the training set and, usually, also a regularization term.
- The derivative of the loss function, with respect to the model parameters, is backpropagated through the network.
- Weights are adjusted until a stop criterion is met:
 - Maximum number of epochs is reached.
 - Loss function stop decreasing.

RNN UNFOLDING

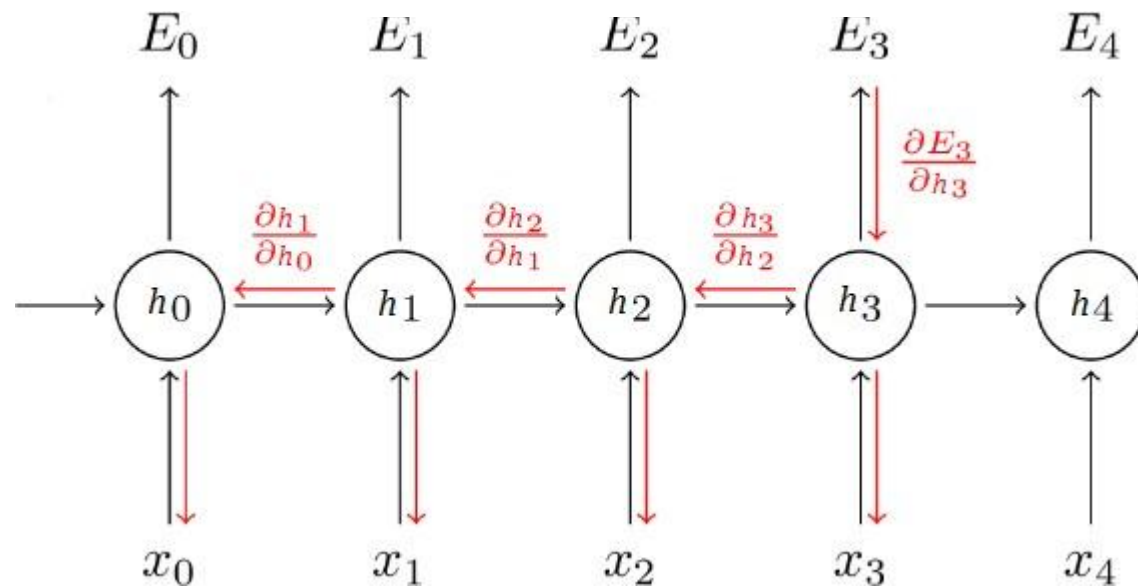
- In order to train the network with gradient descent, the RNN must be **unfolded**.
- Each **replica** of the network is relative to a **different time** interval.
- Now, the architecture of the network become **very deep**, even starting from a shallow RNN.
- The **weights** are constrained to be the **same**.
- **Less parameters** than in other deep architectures.



BACK PROPAGATION THROUGH TIME

- In the example, we need to backpropagate the gradient $\frac{\partial E_3}{\partial W}$ from current time (t_3) to initial time (t_0) → **chain rule** (eq. on the right).
- We **sum** up the **contributions** of **each time** step to the gradient.
- With of **very long** sequence (possibly infinite) we have **untreatable depth**.
- Repeat the procedure only up to a given time (**truncate** BPPT).
- Why it works? Because each **state** carries a little bit of **information** on each **previous input**.
- Once the network is **unfolded**, the procedure is analogue to **standard backpropagation** used in deep Feedforward Neural Networks.

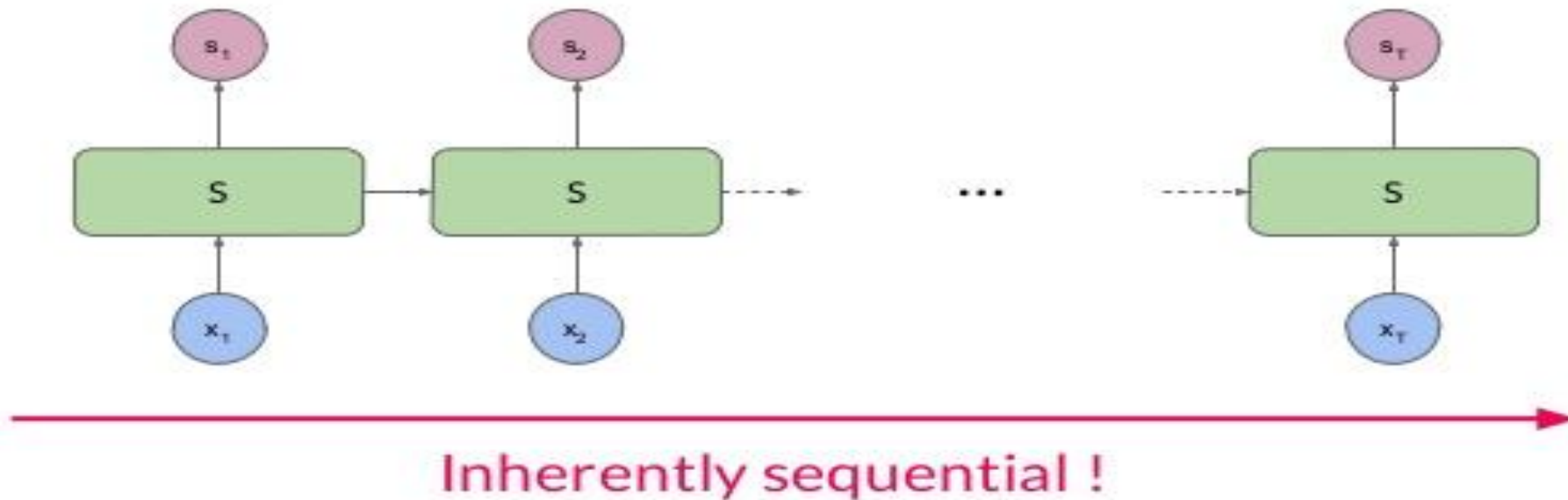
$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^3 \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial h_3} \frac{\partial h_3}{\partial h_k} \frac{\partial h_k}{\partial W}$$



RNN - Problems & Disadvantages

RNN - Disadvantages

Recurrent Neural Networks

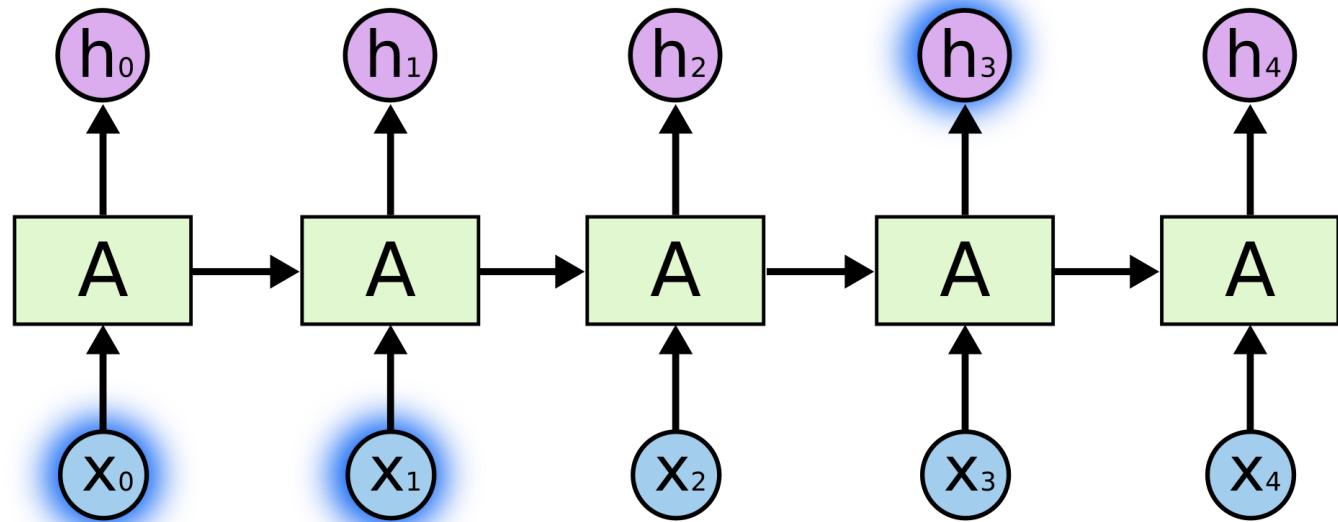


Inherently sequential

RNN - Disadvantages

“the clouds are in the *sky*,”

we don't need any further context – it's pretty obvious the next word is going to be sky. In such cases, where the gap between the relevant information and the place that it's needed is small, RNNs can learn to use the past information.



Don't have memory cells,
 x_0 , x_1 are lost when we reached x_3

Can't persist long term dependencies

“I grew up in Bangalore... I speak fluent ? ”

Ans: Kannada

Recent information (=I speak) suggests that the next word is probably the name of a language,

But, Which language?

we need the context of Bangalore, from further back.

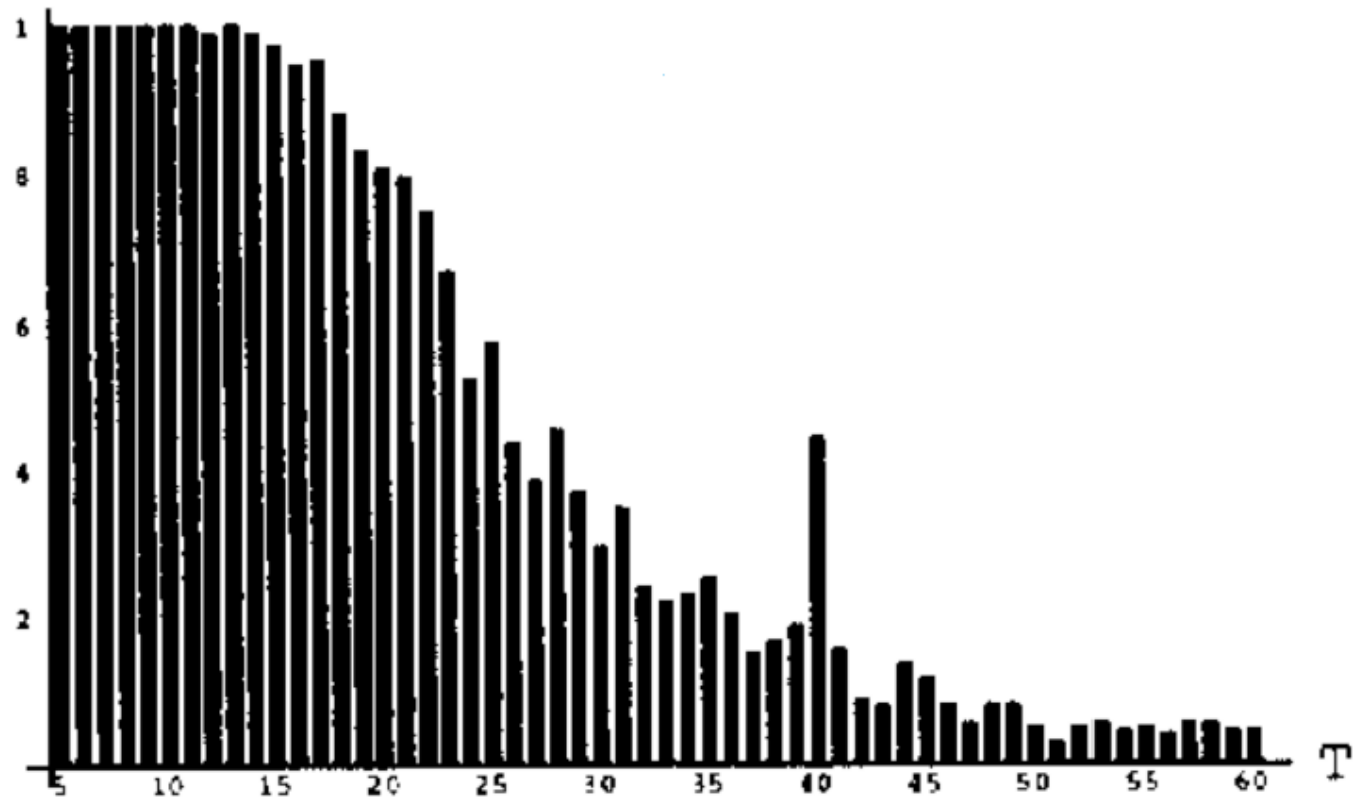
It's entirely possible for the gap between the relevant information and the point where it is needed to become very large.

If the gap grows, RNNs cannot learn to connect the information.

Can't persist long term dependencies

VANISHING GRADIENT: SIMPLE EXPERIMENT

- Bengio, 1991.
- A simple RNN is trained to keep 1 bit of information for T time steps.
- $P(\text{success}|T)$ decreases exponentially as T increases.



[Image:Yoshua Bengio]

VANISHING GRADIENT: TOO MANY PRODUCTS!

- In order to have (local) **stability**, the **spectral radius** of the matrix W_h^h must be **lower than 1**.
- Consider **state update** equation $h[t+1] = f(h[t], u[t+1])$. We can see it is a **recursive** equation.
- When input sequence is given, the previous equation can be rewritten **explicitly** as:

$$h[t+1] = f_t(h[t]) = f_t(f_{t-1}(\dots f_0(h[0]))). \quad (1)$$

- The resulting gradient, relative to the loss at time t will be:

$$\frac{\partial L_t}{\partial W} = \sum_{\tau} \frac{\partial L_t}{\partial h_t} \frac{\partial h_t}{\partial h_{\tau}} \frac{\partial h_{\tau}}{\partial W}. \quad (2)$$

- The Jacobian of matrix **derivatives** $\frac{\partial h_t}{\partial h_{\tau}}$ can be **factorized** as follows

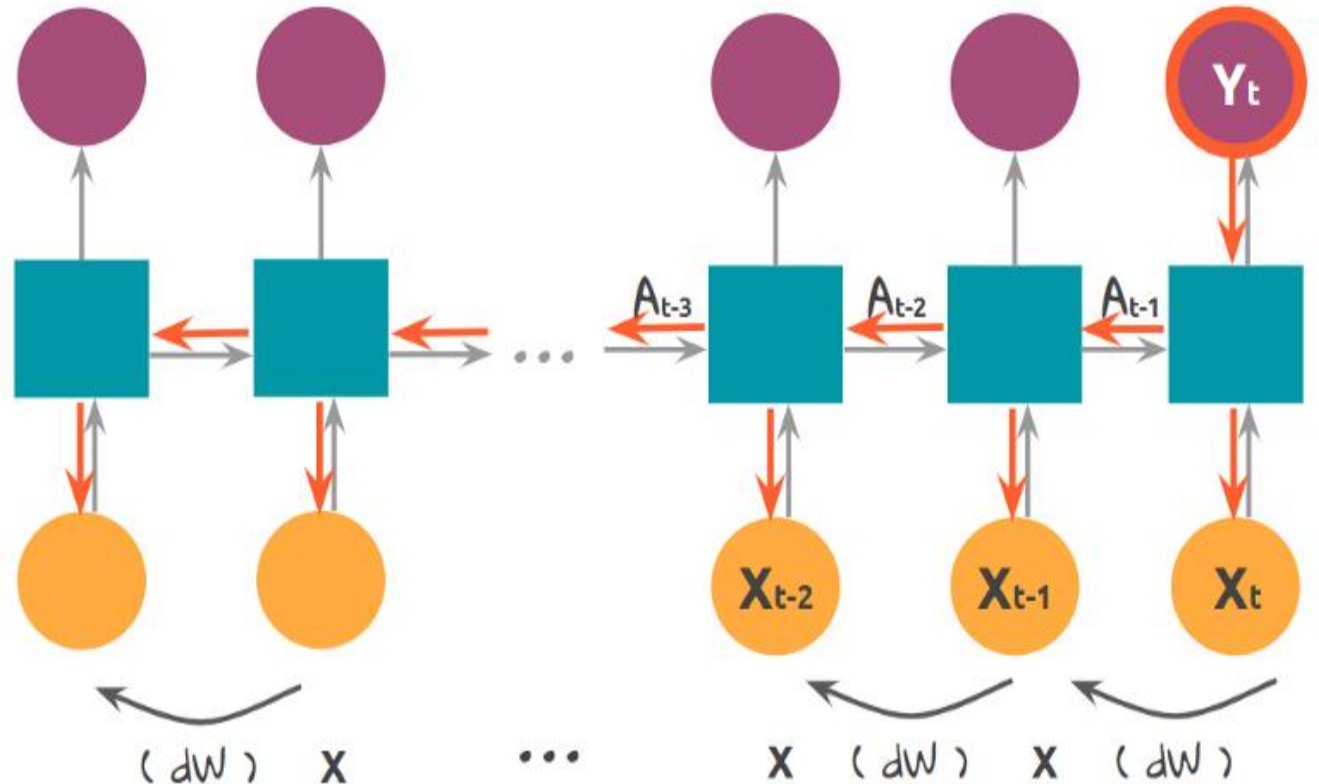
$$\frac{\partial h_t}{\partial h_{\tau}} = \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial h_{t-2}} \dots \frac{\partial h_{\tau+1}}{\partial h_{\tau}} = f'_t f'_{t-1} \dots f'_{\tau+1} \quad (3)$$

Let us demystify this in the next slides

VANISHING GRADIENT: TOO MANY PRODUCTS!

Imagine when the gradients are bigger than 1. The updated values become so big to use it for optimizing. This is called **exploding gradients**.

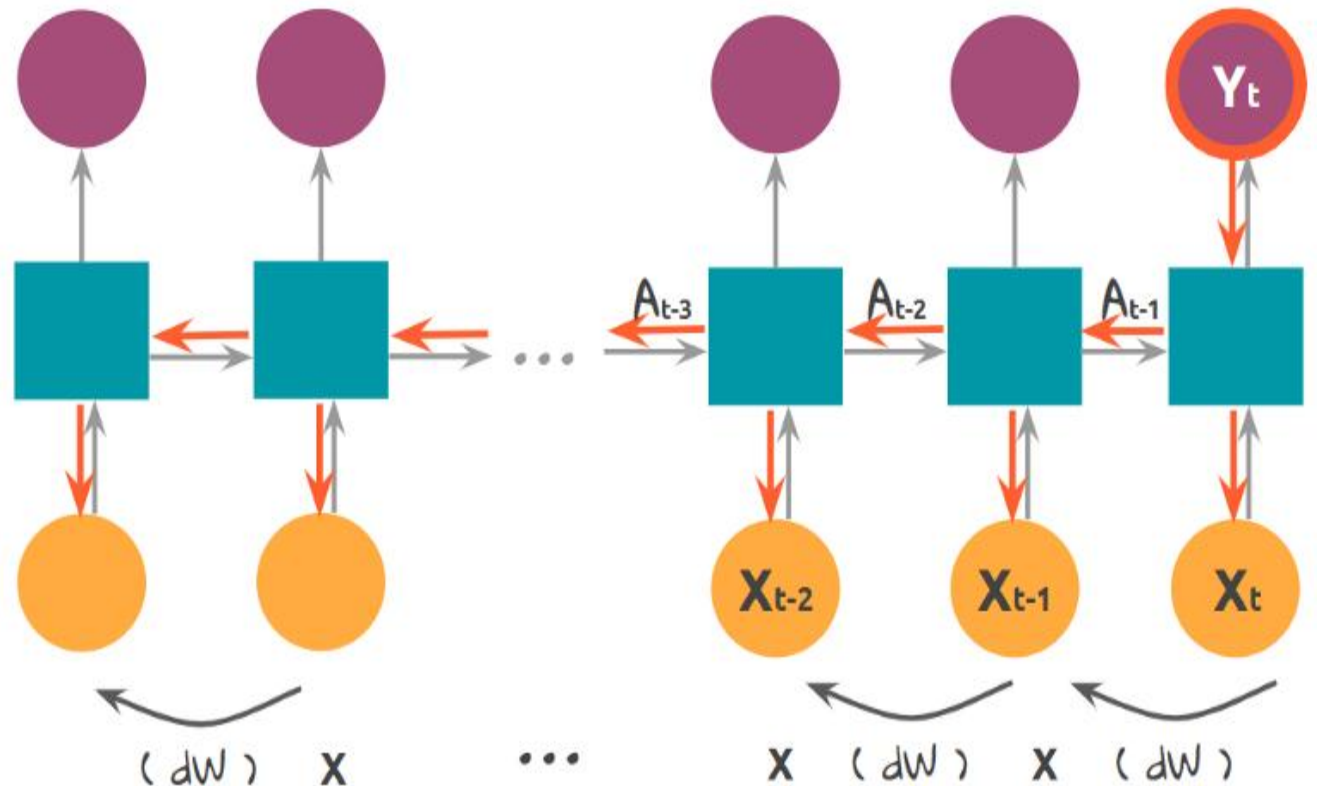
Solution: Using proper learning rate and activation function we can fix the range that gradients can't go over.



VANISHING GRADIENT: TOO MANY PRODUCTS!

When the gradients are smaller than 1, If we keep multiplying the values lower than 1, the result becomes smaller and smaller.

After some steps, there will be no significant difference in outcome, and it can't make any update in weights. It is called **vanishing gradients**.



HOW TO LIMIT VANISHING GRADIENT ISSUE?

- Use ReLU activations (in RNN however, they cause the “dying neurons” problem).
- Use **LSTM** or **GRU** architectures (discussed later).
- Use a proper **initialization** of the weights in W .



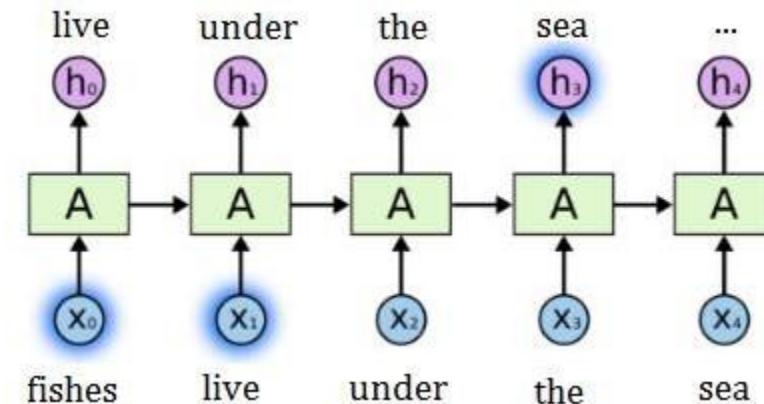
4. Long Short Term Memory (LSTM)



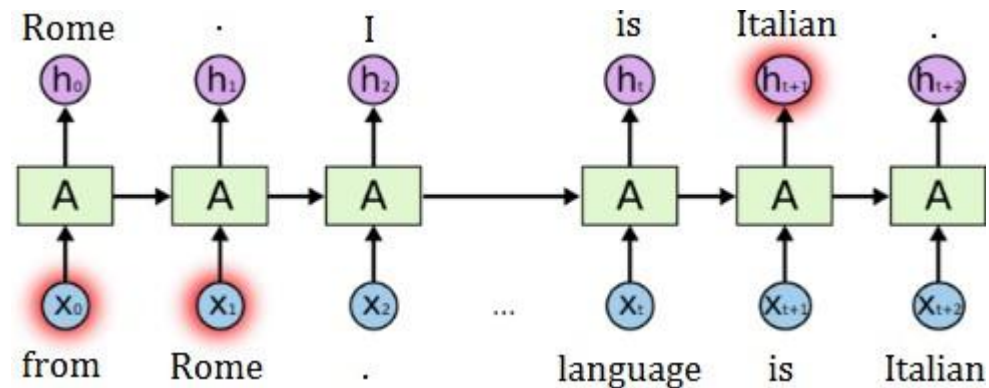
LONG-TERM DEPENDENCIES

- Due to vanishing gradient, RNN are **uncapable** of learning **long-term** dependencies.
- Some applications require **both short** and **long** term dependencies.
- Example: Natural Language Processing (**NLP**).
- In some cases, **short-term** dependencies are **sufficient** to make predictions.

- Consider to train the RNN to make 1-step ahead prediction.
- To predict the word '**sea**' it is sufficient to look only **3** step back in time.
- In this case, it is sufficient to **backpropagate** the **gradient** **3** step back to successfully learn this task.



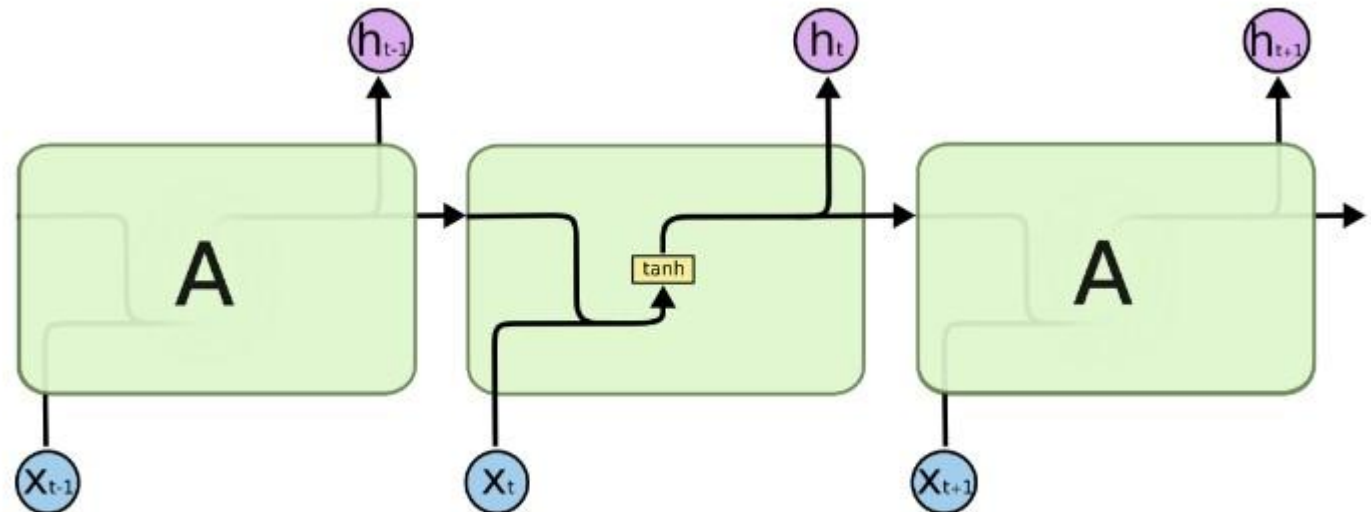
- Let's stick to 1-step ahead prediction.
- Consider the sentence: *I am from Rome. I was born 30 years ago. I like eating good food and riding my bike. My native language is Italian.*
- When we want to predict the word *Italian*, we have to look back *several time steps*, up to the word *Rome*.
- In this case, the short-term memory of the RNN would not do the trick.



- As the *gap in time grows*, the *harder* for an RNN become to *handle* the problem.
- Let's see how Long-Short Term Memory (**LSTM**) can handle this difficulty.

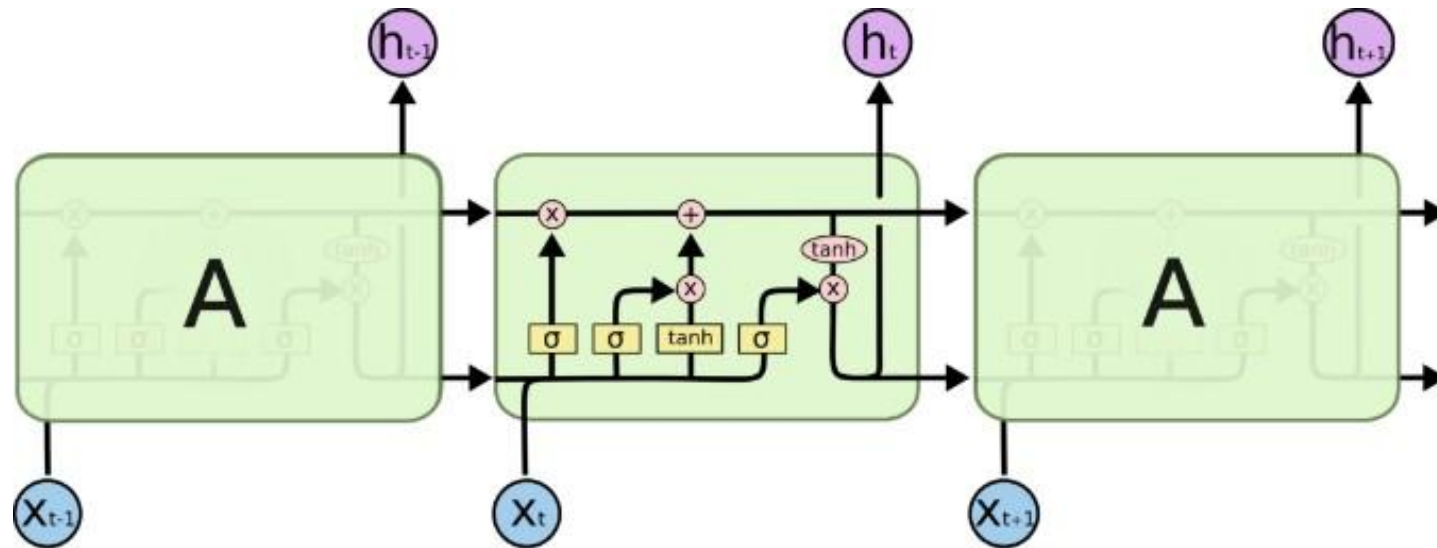
LSTM OVERVIEW

- Introduced by Hochreiter & Schmidhuber (1997).
- Work very well on many different problems and are widely used nowadays.
- Like RNN, they must be unfolded in time to be trained and understood.
- Let's recall the unfolded version of a RNN.
- A very simple processing unit is repeated each time.

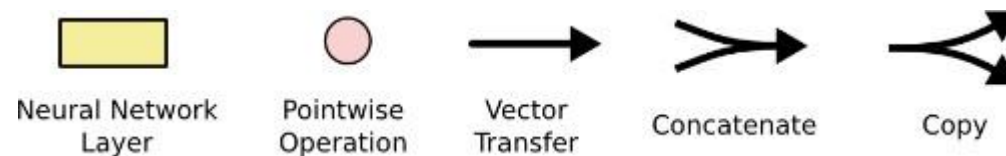


The repeating module in a standard RNN contains a single layer.

- The processing unit of the LSTM is more complex and is called *cell*.
- An LSTM cell is composed of 4 layers, interacting with each other in a special way.

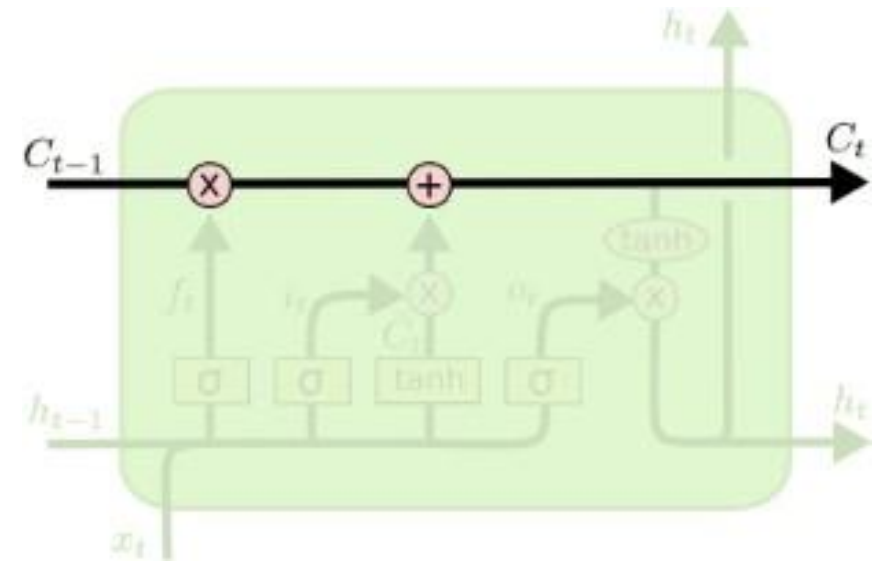
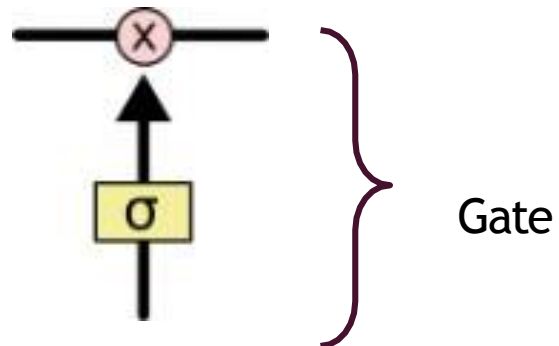


The repeating module in an LSTM contains four interacting layers.



CELL STATE AND GATES

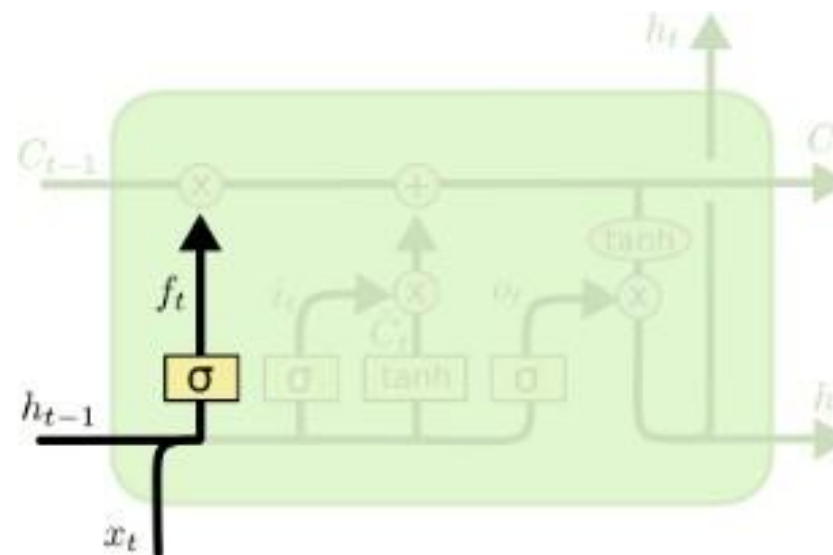
- The **state** of a cell at time t is C_t .
- The LSTM modify the state only through **linear interactions**: information flows **smoothly** across time.
- LSTM **protect** and **control** the information in the cell through **3 gates**.
- **Gates are implemented by a sigmoid and a pointwise multiplication.**



Cell state

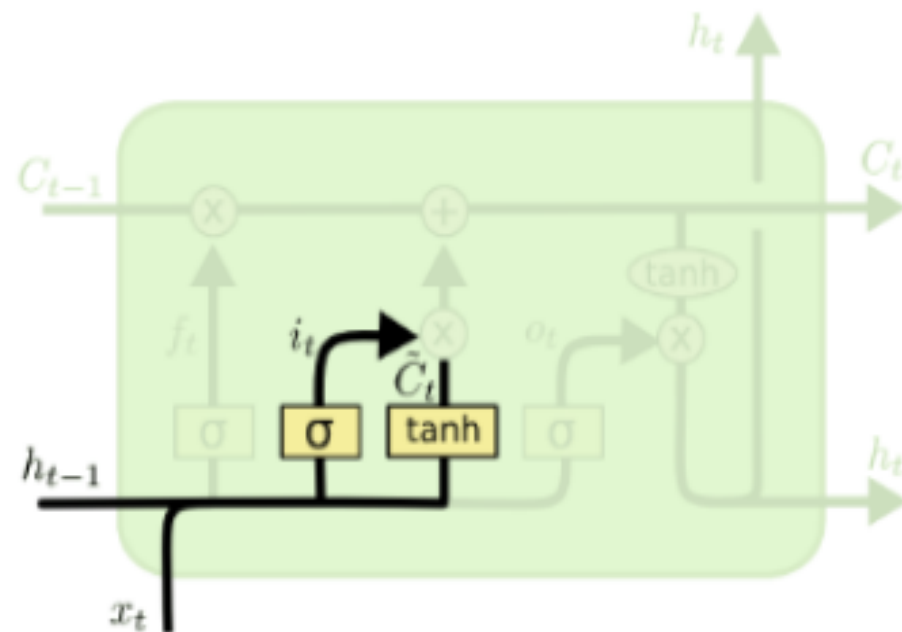
1. FORGET GATE

- Decide what information should be **discarded** from the cell state.
- Gate controlled by current input x_t and past cell output h_{t-1} .
- NLP example: cell state keep the gender of the present subject to use correct pronouns.
- When sees a **new subject**, **forget the gender** of the old subject.



2. UPDATE GATE

- With forget gate we decided wheter or not to forget cell content.
- After, with update gate we decide **how much** to **update** the old state C_{t-1} with a **new candidate** \tilde{C}_t .
- Update gate: $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) = \begin{cases} 0 \rightarrow \text{no update} \\ 1 \rightarrow \text{completely update} \end{cases}$
- New candidate: $\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$.
- New state: $C_t = C_{t-1} + i_t * \tilde{C}_t$.
- In the NLP example, we **update** the cell state as we **see** a **new subject**.
- Note that the new candidate is computed exactly like the **state in traditional RNN** (same difference equation).

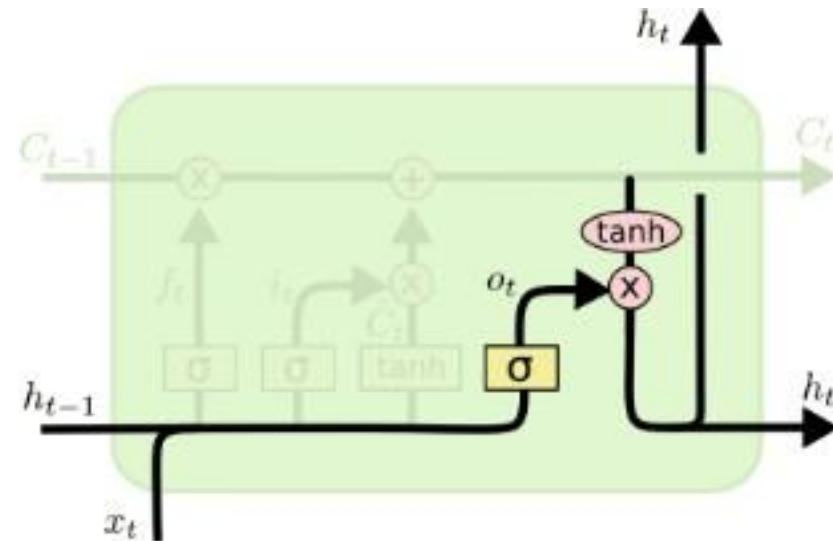


3. OUTPUT GATE

- The output is a filtered version of the cell state.
- Cell state is fed into a tanh, which squashed its values between -1 and 1.
- Then, the gate **select** the **part** to be returned as **output**.

Cell output: $h_t = o_t * \tanh(C_t)$

- In NLP example, after having seen a subject, if a verb comes next, the cell outputs information about being singular or plural.



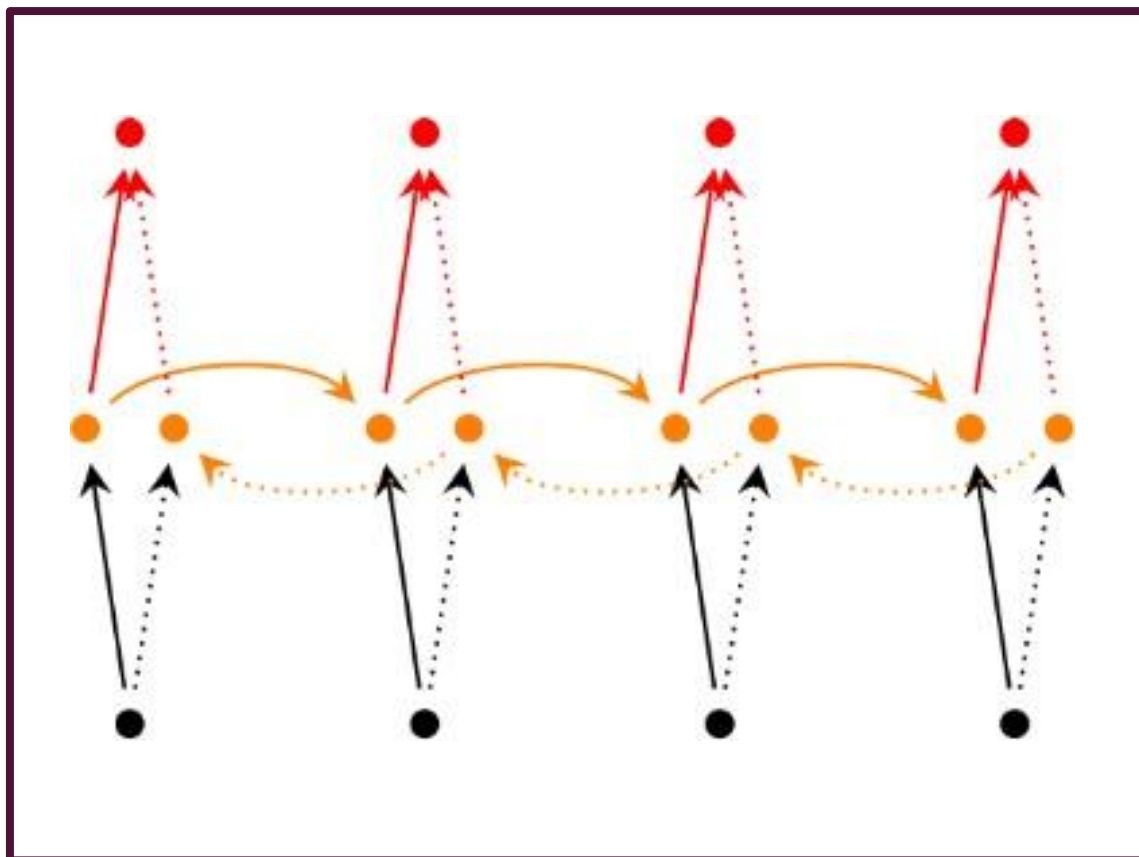
LSTM DOWNSIDES

- Even if LSTM provides a huge improvement w.r.t. RNN, it still struggles with very long time dependencies.
- Number of parameters: $4 \cdot (N_i + 1) \cdot N_o + N_o^2$.
- Example: input = time series of 100 elements, LSTM units = 256 \rightarrow 168960 parameters.
- Scales up quickly!! Lot of parameters = **lot of training data**.
- **Memory** problems when dealing with lot of data.
- **Long training** time (use GPU computing).

GATED RECURRENT UNITS (GRU)

- Several LSTM variants exists.
- GRU is one of the most famous alternative architectures (Cho, et al. (2014).
- **It combines the forget and input gates into a single update gate.**
- It also **merges** the **cell state** and **hidden state**, and makes some other changes.
- The cell is characterize by **fewer** parameters.

BIDIRECTIONAL RNN



- The output at time t may not only depend on the previous elements in the sequence, but also future elements.
- **Example:** to predict a missing word in a sequence you want to look at both the left and the right context.
- Two RNNs stacked on top of each other.
- Output computed based on the hidden state of both RNNs.
- Huang Zhiheng, Xu Wei, Yu Kai. *Bidirectional LSTM Conditional Random Field Models for Sequence Tagging*.



Thank You