

Deep Learning

Recurrent Neural Network



Geoffroy Peeters

contact: geoffroy.peeters@telecom-paris.fr

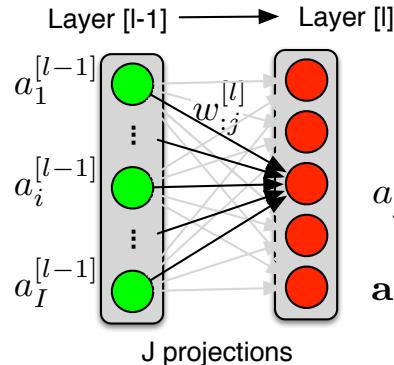
Télécom-Paris, IP-Paris, France

Roadmap

Architectures: *distinguish the way neurons are inter-connected*

Lecture 1

Fully Connected Neural Network

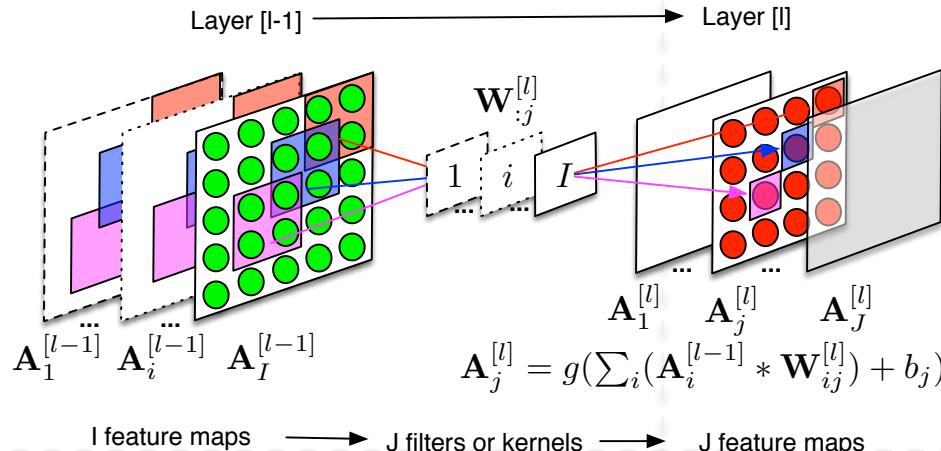


$$a_j^{[l]} = g(\sum_i a_i^{[l-1]} w_{ij} + b_j^{[l]})$$

$$\mathbf{a}^{[l]} = g(\mathbf{a}^{[l-1]} \mathbf{W}^{[l]} + \mathbf{b}^{[l]})$$

Lecture 2

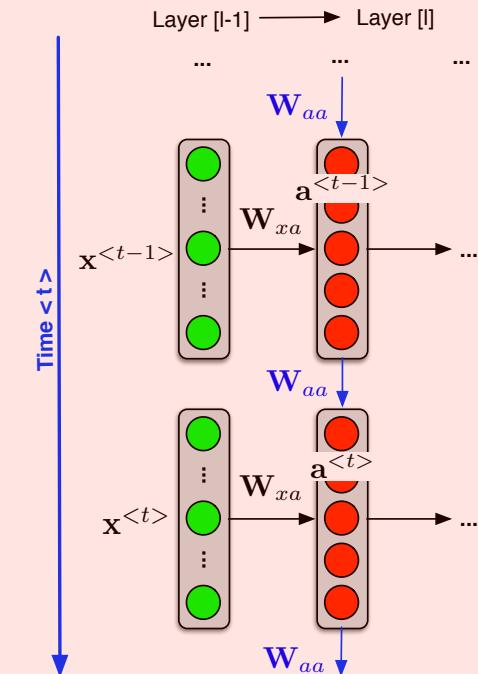
Convolutional Neural Network



Lecture 3

Recurrent Neural Network

$$\mathbf{a}^{<t>} = g(\mathbf{x}^{<t>} \mathbf{W}_{xa} + \mathbf{a}^{<t-1>} \mathbf{W}_{aa} + \mathbf{b}_a)$$

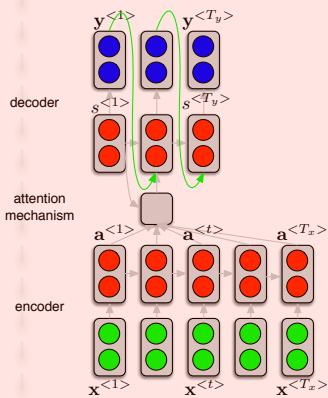


Roadmap

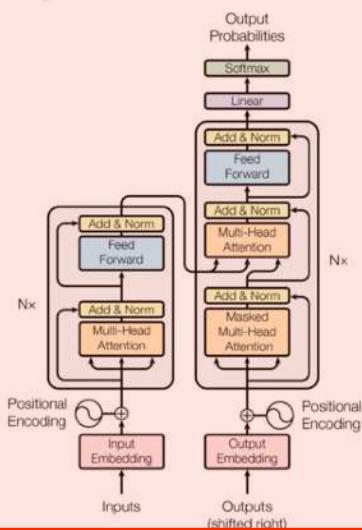
Meta-architectures: *the way architectures are plug to form a system*

Lecture 3

Encoder-Decoder with Attention

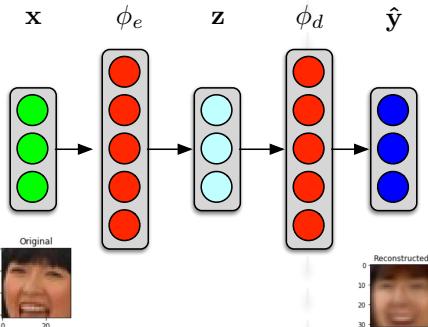


Transformer

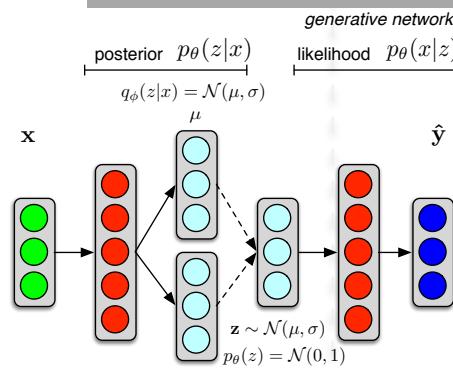


Lecture 4

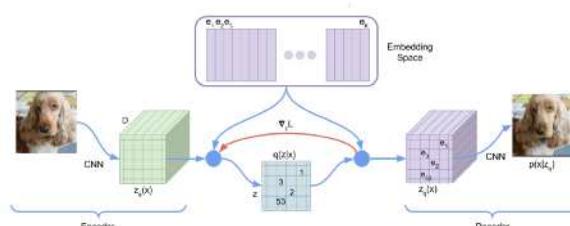
Auto-Encoder



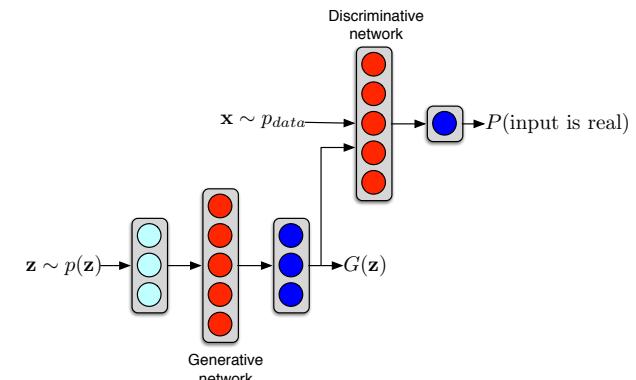
Variational-Auto-Encoder



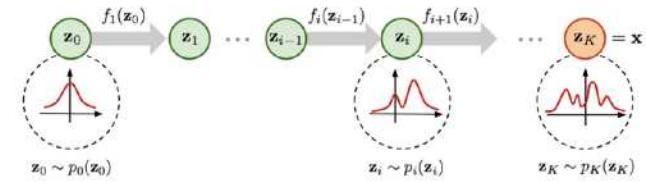
VQ-VAE



Generative Adversarial Network



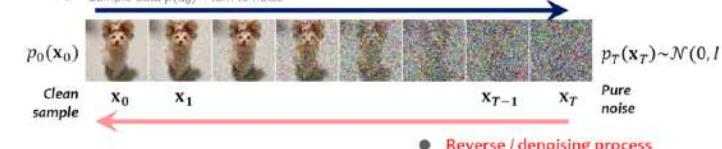
Normalizing Flows



Denoising Diffusion Models

Forward / noising process

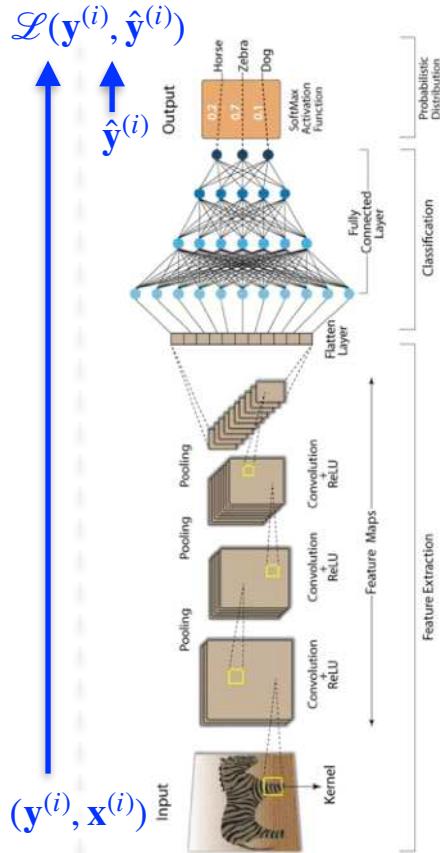
- Sample data $p(x_0)$ → turn to noise



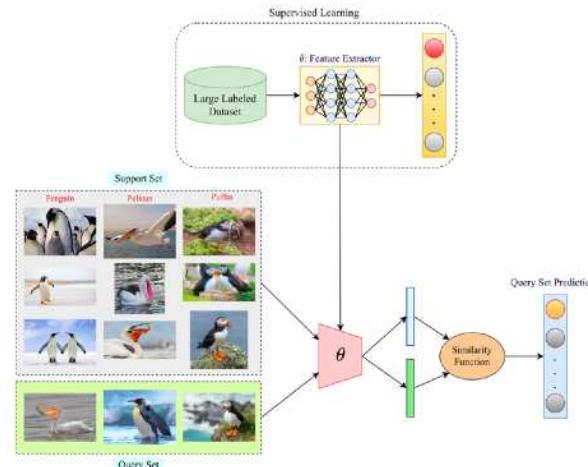
Roadmap

Training-paradigms: *the paradigm we use to train a system*

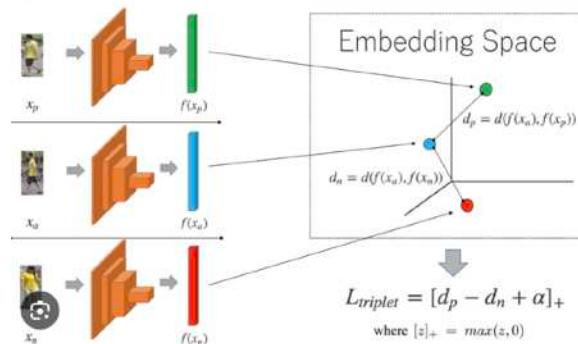
Supervised learning



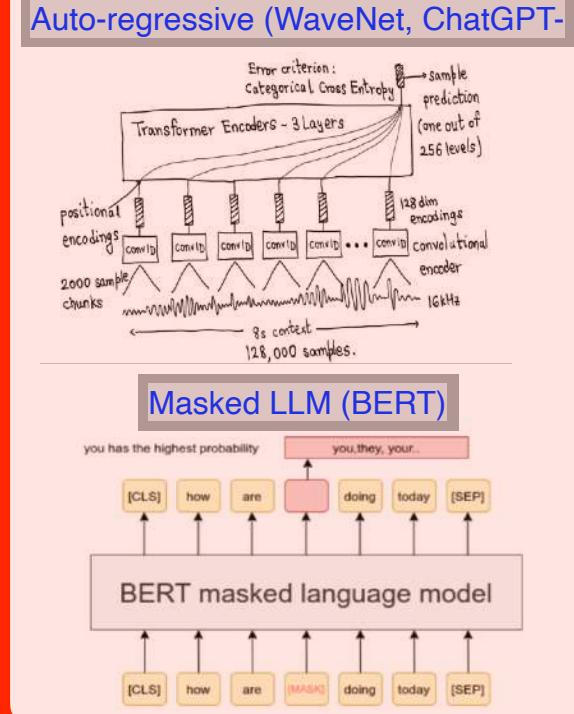
Zero/few Shot learning



Metric learning



Self Supervised learning



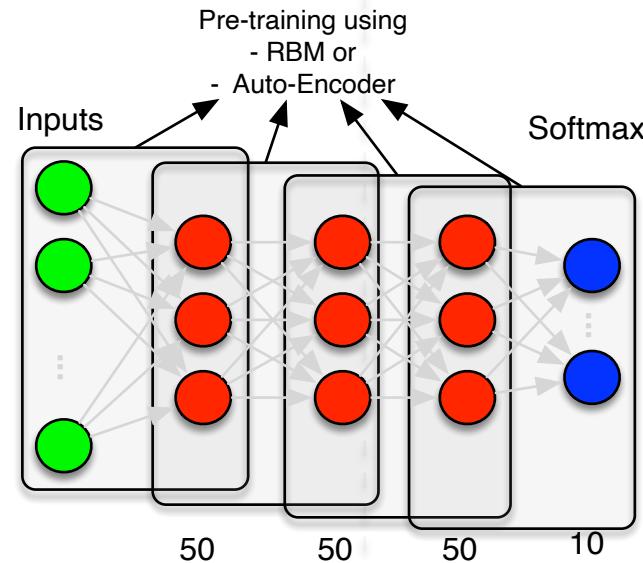
from <https://larbifarihi.medium.com/classifying-cats-and-dogs-using-convolutional-neural-networks-cnn-cc6bc7bad64d>

Three main types of Nets

Multi Layers Perceptron (MLP), Fully-Connected (FC), Feed-Forward

Multi Layers Perceptron (Fully Connected)

Specificity: Inner neurons are connected to all previous neurons

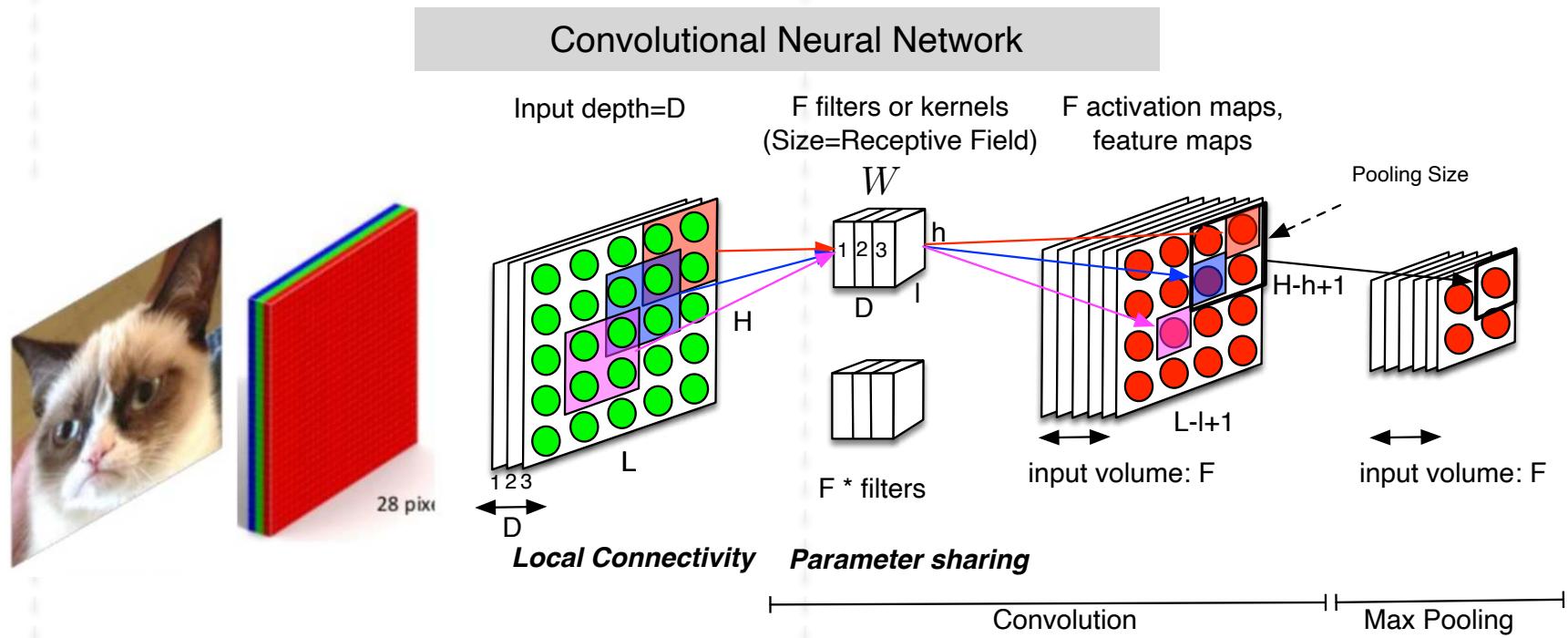


$$\begin{aligned}\underline{z}^{[l]} &= \underline{W}^{[l]} \underline{a}^{[l-1]} + \underline{b}^{[l]} \\ \underline{a}^{[l]} &= g^{[l]}(\underline{z}^{[l]})\end{aligned}$$

Three main types of Nets

Convolutional Neural Networks (CNN)

They are locally connected, not fully

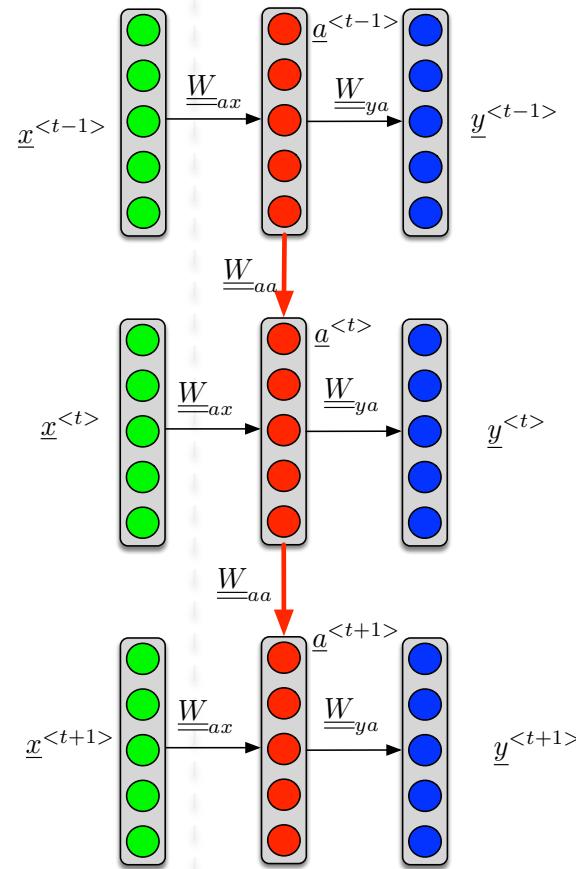


Three main types of Nets

Recurrent Neural Networks (RNN)

Network has “memory” from the past, like here it is the sequence of the images from the video

Recurrent Neural Network



Recurrent Neural Network for Sequential Data

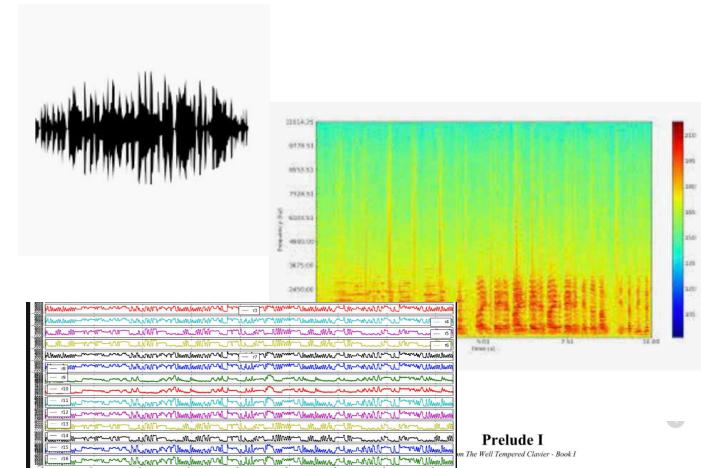
What are sequential data ?

- **Sequential data are any type of data for which the order matters**
- Examples:
 - **Time series:**
 - Audio (sequence of sound pressures or sequence of Fourier transforms)
 - Sequence of musical notes
 - Video (sequence of images)
 - Sensors (heart-beat, plane altitude)
 - Stock market
 - **Ordered:** (but not with time)
 - Text/Words
 - Genes/ADN

NN without non-linearity is
just a direct projection, not
a NN

– Common models:

- Vector linear autoregression (VAR)
- Markov model (Discrete-State HMMs)
- Kalman filters (Continuous-State HMMs)
- ...



ROMEO & JULIETTE WILLIAM SHAKESPEARE

Juliette : O Roméo ! Roméo ! Pourquoi es-tu Roméo ?
Renie ton père et abdique ton nom ; ou, si tu ne le veux
pas, jure de m'aimer, et je ne serai plus une Capulet.

Roméo, à part : Dois-je l'écouter encore ou lui répondre ?

Juliette : Ton nom est mon ennemi. Tu n'es pas un Montague, tu es moi-même. Qu'est-ce qu'un Montague ? Ce n'est ni une main, ni un pied, ni un bras, si un visage, ni rien qui fasse partie d'un homme... Oh ! Sois quelque autre nom ! Qu'y a-t-il dans un nom ? Ce que nous appelons une rose embaumerait autant sous un autre nom. Ainsi, quand Roméo ne s'appellerait plus Roméo, il conserverait encore les chères perfections qu'il

```
/* Simple HelloButton() method.
 * @version 1.0
 * @author John Doe <jdoe@example.com>
 */
HelloButton()
{
    JButton hello = new JButton( "Hello, world!" );
    hello.addActionListener( new HelloBtnListener()
    {
        // use the JFrame type until support for t
        // new component is finished
        JFrame frame = new JFrame( "Hello Button" );
        frame.setDefaultCloseOperation( EXIT_ON_CLOSE );
        frame.add( hello );
        frame.pack();
        frame.setVisible( true );
        // display the frame
    } );
}
```

A 10x10 grid of colored DNA bases (A, T, C, G) represented by different colors: A is green, T is orange, C is blue, and G is red. The grid shows a repeating pattern of these four bases across the rows and columns.

Recurrent Neural Network for Sequential Data

Notations

- Inputs:
 x is a vector, t is a position in the sequence
 - $\mathbf{x}^{(t)}$: input vector \mathbf{x} at time t (of dimension $n^{[0]}$)
 - $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(T_x)}\}$: a sequence of inputs of length T_x
e.g. it is the sequence of words
- Outputs:
 - $\mathbf{y}^{(t)}$: output vector \mathbf{y} at time t
 - $\{\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \dots, \mathbf{y}^{(T_y)}\}$: a sequence of outputs of length T_y
- T_x and T_y can be of different lengths e.g. it is the sentences
- $x^{(i)(t)}$ is the i^{th} example in the training-set
 - $T_x^{(i)}$ the length of the i^{th} input sequence
 - $T_y^{(i)}$ the length of the i^{th} output sequencee.g. in english the text will be small than in french

Recurrent Neural Network for Sequential Data

Notations

- **Example:** categorical inputs/ binary outputs

- "Named Entity Recognition"

$\{x\}$	$x^{<1>}$	$x^{<2>}$...				$x^{<7>}$
	Emmanuel	Macron	is	the	president	of	France
$\{y\}$	$y^{<1>}$	$y^{<2>}$...				$y^{<7>}$
	1	1	0	0	0	0	1

if it is the
named entity =
1

- Different types of **output $y^{(t)}$:**

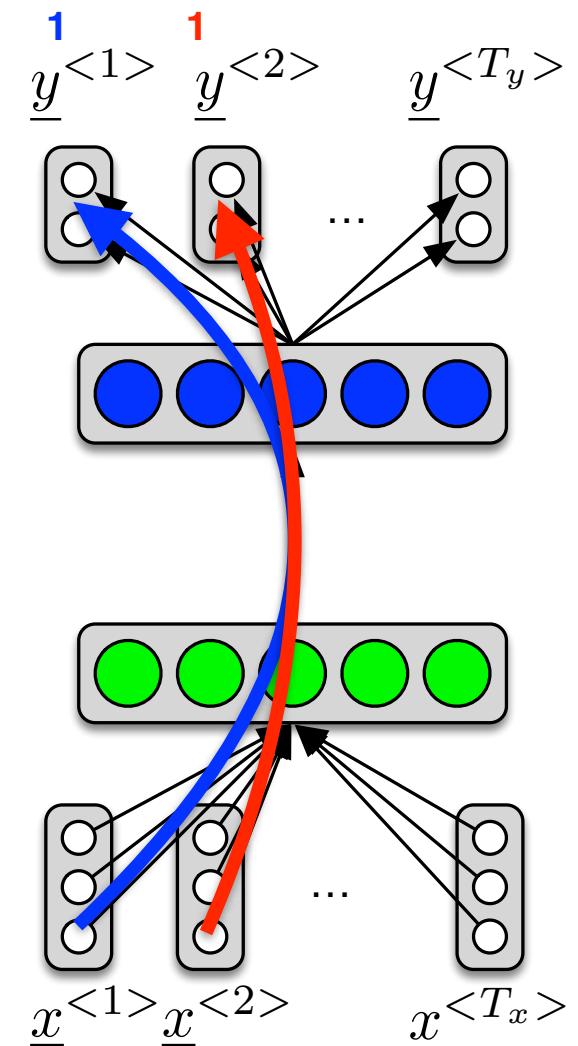
- Continuous values $\mathbf{y}^{(t)} \in \mathbb{R}^{n_y}$ (Activation: linear, Loss: MSE)
- Binary-classes $y^{(t)} \in \{0,1\}$ (Activation: sigmoid, Loss: Binary-Cross-Entropy)
- Multi-classes $y^{(t)} \in \{1\dots K\} \Rightarrow \mathbf{y}^{(t)} \in \{0,1\}^K$ (Activation: softmax, Loss: Cross-Entropy)

- Different types of **input $\mathbf{x}^{(t)}$** each word is considered as a class

- Continuous values $\mathbf{x}^{(t)} \in \mathbb{R}^{n_x}$
- Categorical values $x^{(t)} \in \{1\dots K\} \Rightarrow \text{how do we process categorical values as input ?}$
 - One-hot-encoding: $\mathbf{x}^{(t)} \in \{0,1\}^K$
 - Embedding: $\mathbf{x}^{(t)} \in \mathbb{R}^d$

Recurrent Neural Network for Sequential Data

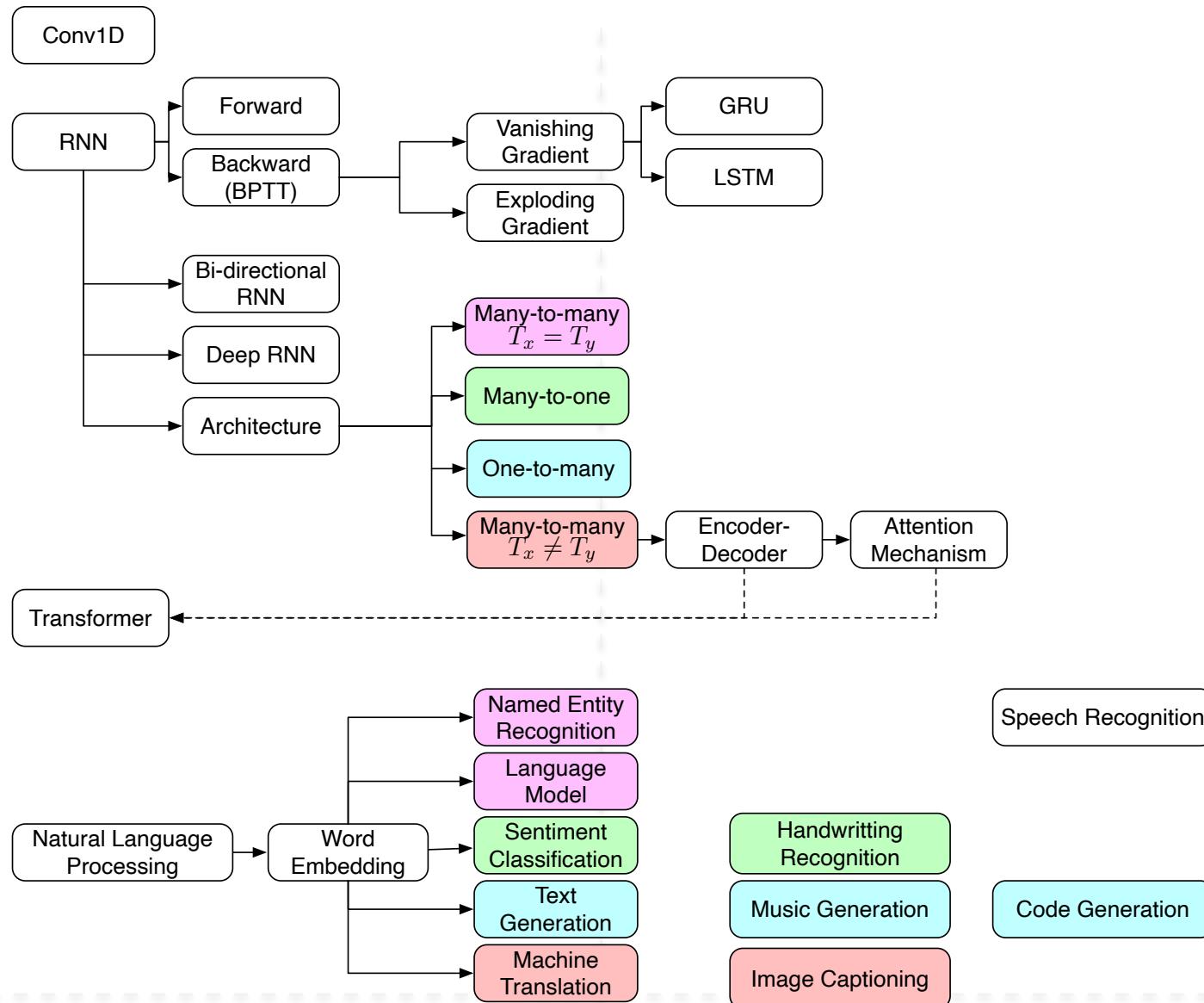
- **Why not using a Multi-Layer-Perceptron ?**
 - We could indeed represent an input sequence $\{\underline{x}^{(1)}, \dots, \underline{x}^{(T_x)}\}$ as a large-vector of dimension $(n^{[0]} T_x)$
 - it would require a huge input dimension !
- Problem:
 - $T_x^{(i)}$ can be different for each i ,
 \Rightarrow the dimension $(n^{[0]} T_x^{(i)})$ would be **different** for each i
 - Solution:
 - zero-padding ? $(n^{[0]} \max_i T_x^{(i)})$
 - it would still require a **huge input dimension** !
- Problem:
 - the network will have to learn different weights for the same dimension $n^{[0]}$ at various time $\langle t \rangle$ in the sequence
 \Rightarrow **no weight sharing** !
 - Solution:
 - Recurrent Neural Network
 - 1D ConvNet (convolution only over time)



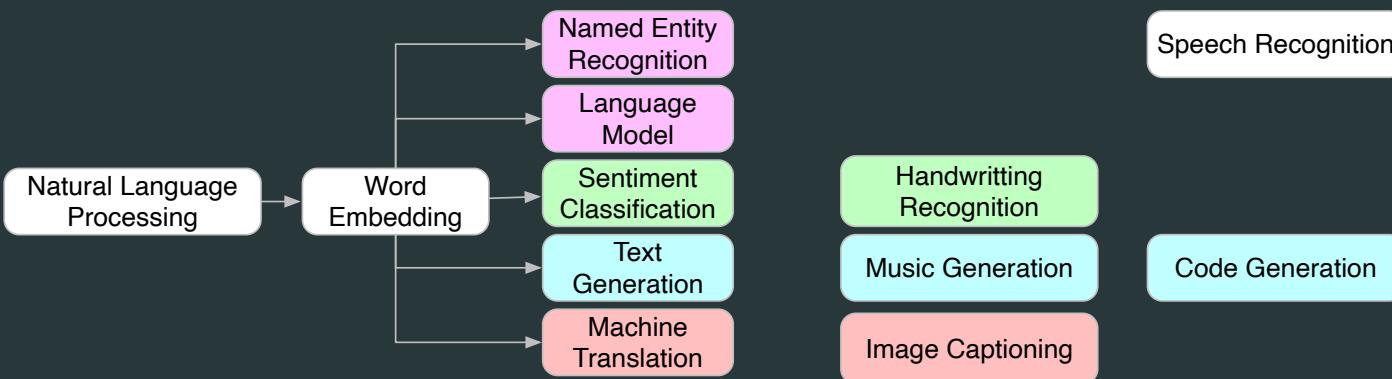
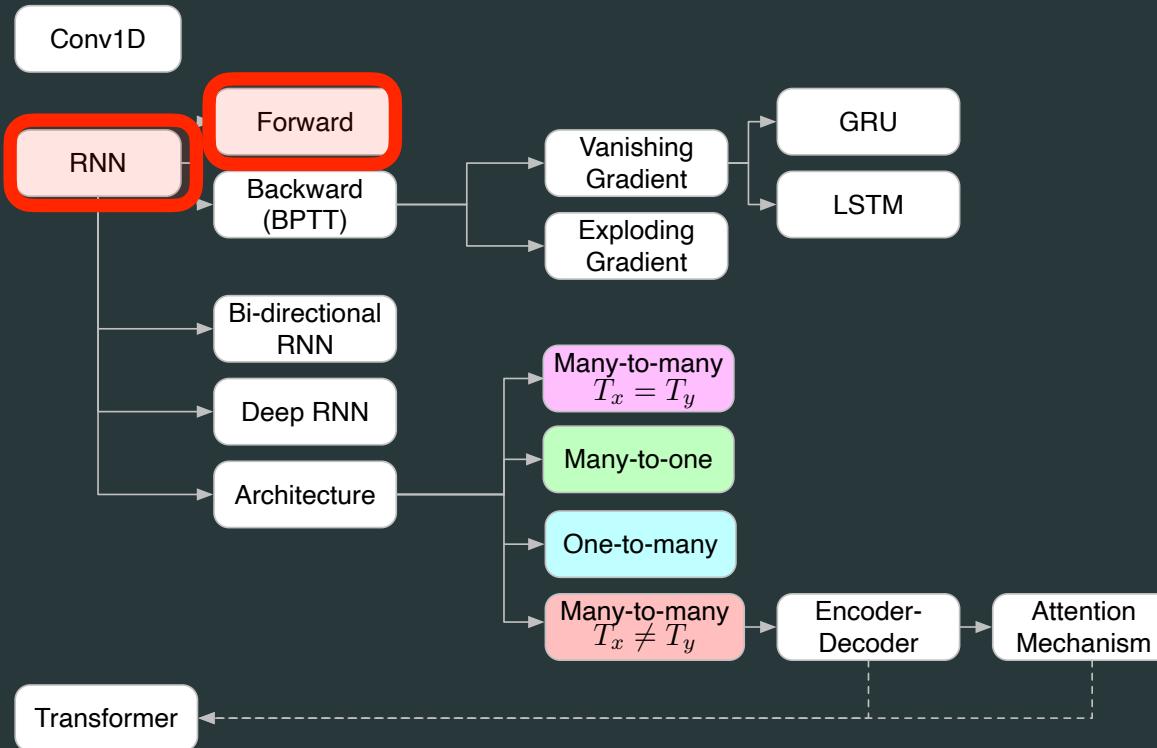
EmmanueEmmanuel

here we need to relearn from scratch for each neuron
for a word "emmanuel"

Overview

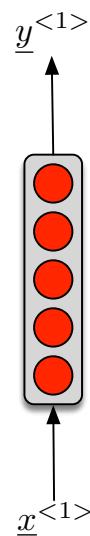


Recurrent Neural Network for Sequential Data



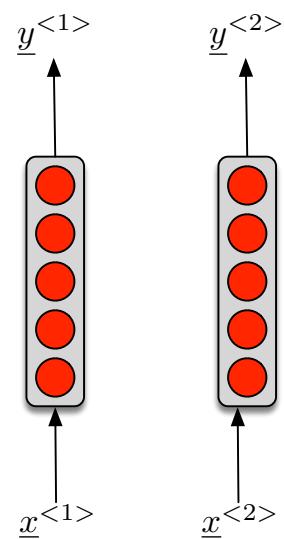
Recurrent Neural Network for Sequential Data

What is an RNN ?



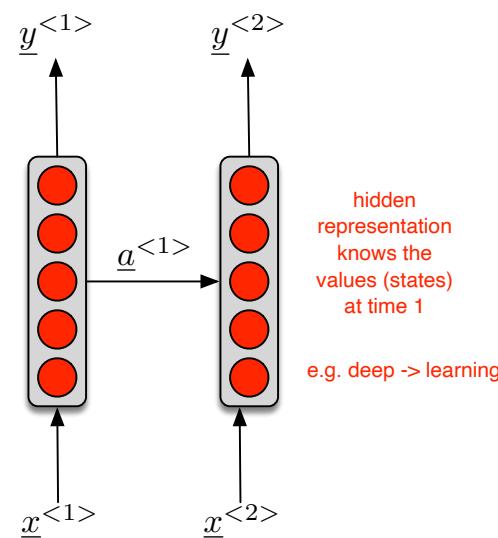
Recurrent Neural Network for Sequential Data

What is an RNN ?



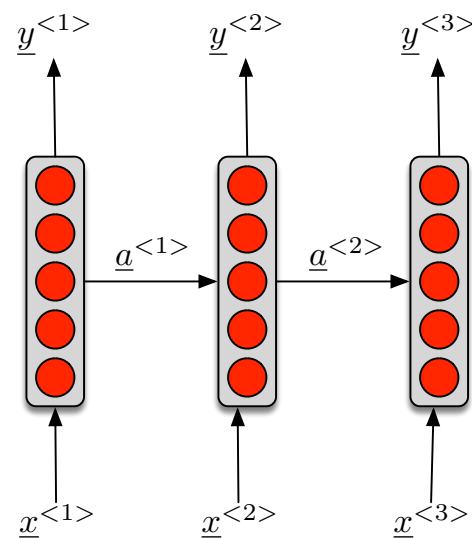
Recurrent Neural Network for Sequential Data

What is an RNN ?



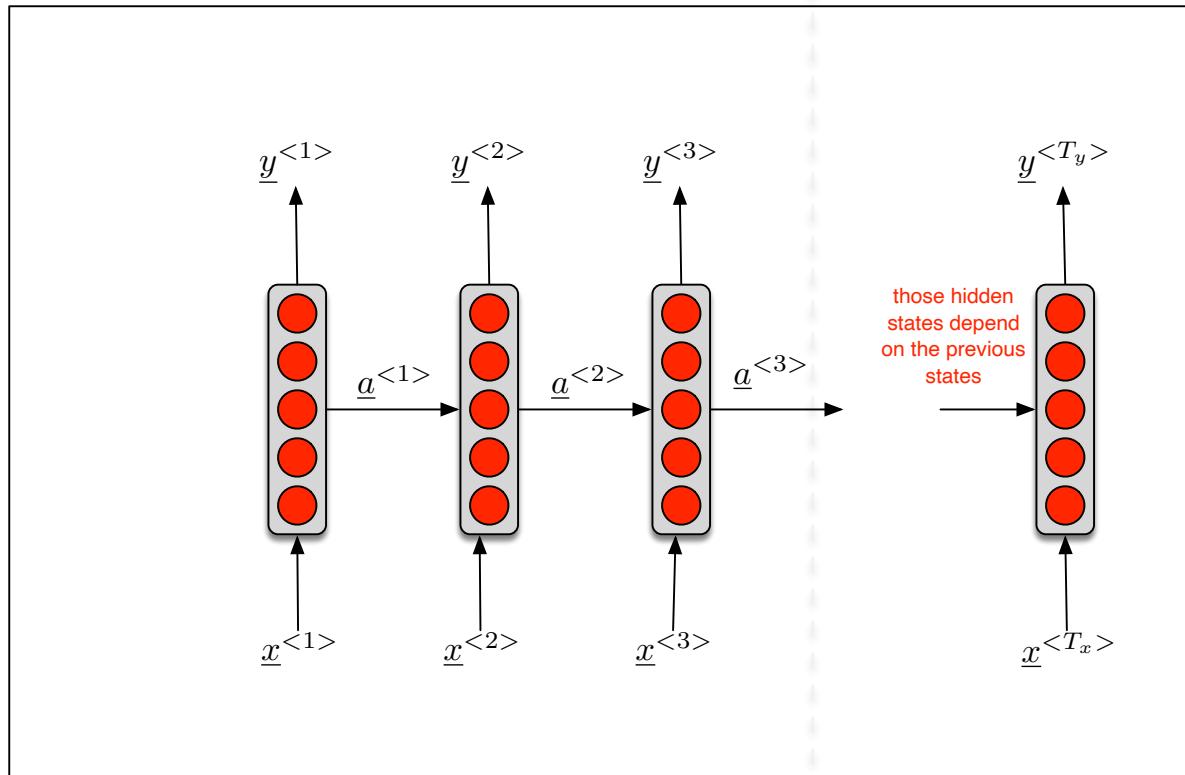
Recurrent Neural Network for Sequential Data

What is an RNN ?



Recurrent Neural Network for Sequential Data

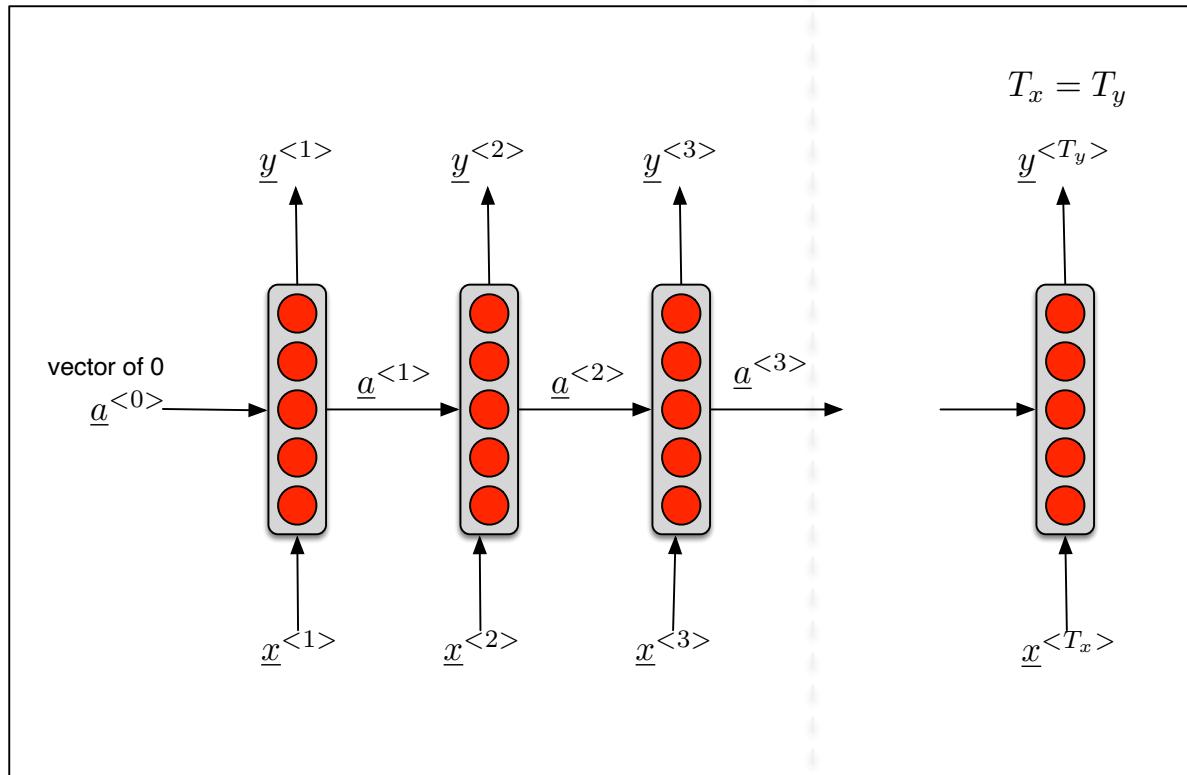
What is an RNN ?



- At each time step $\langle t \rangle$,
 - the RNN gets the activation $a^{(t-1)}$ from the previous time $\langle t - 1 \rangle$

Recurrent Neural Network for Sequential Data

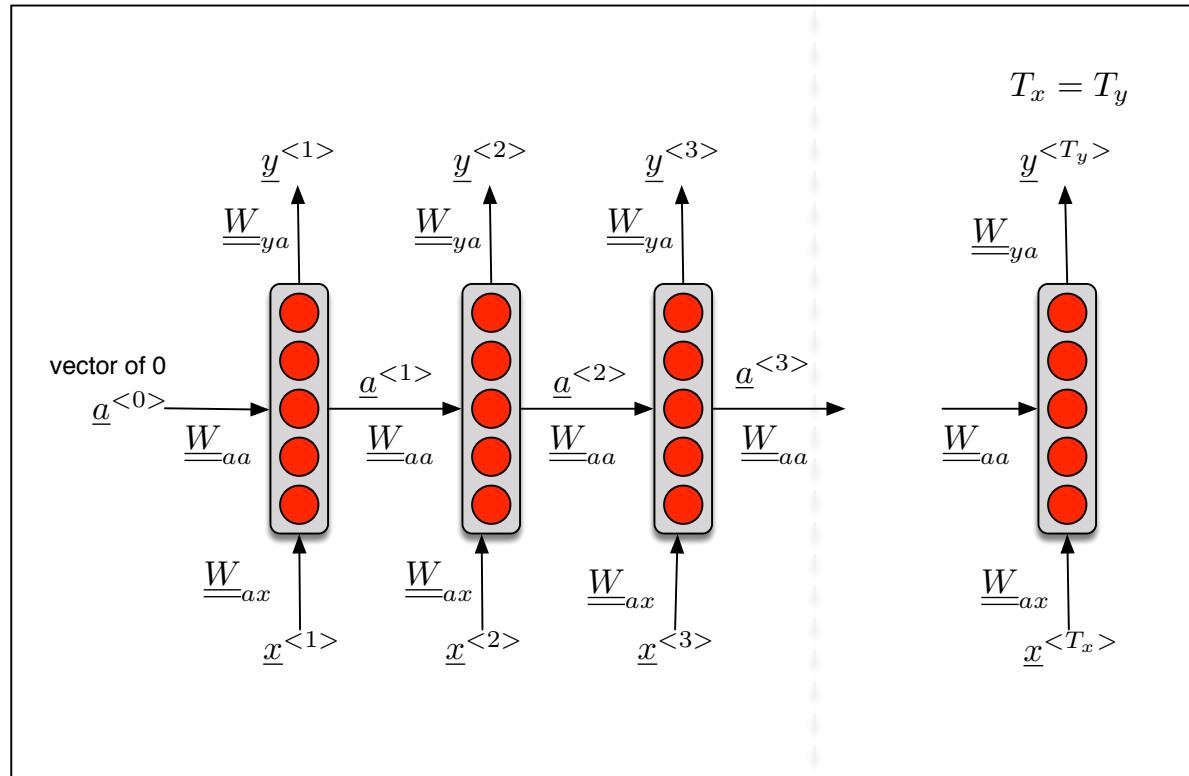
What is an RNN ?



- First activation ?
 - Initialise as $a^{<0>} = 0$

Recurrent Neural Network for Sequential Data

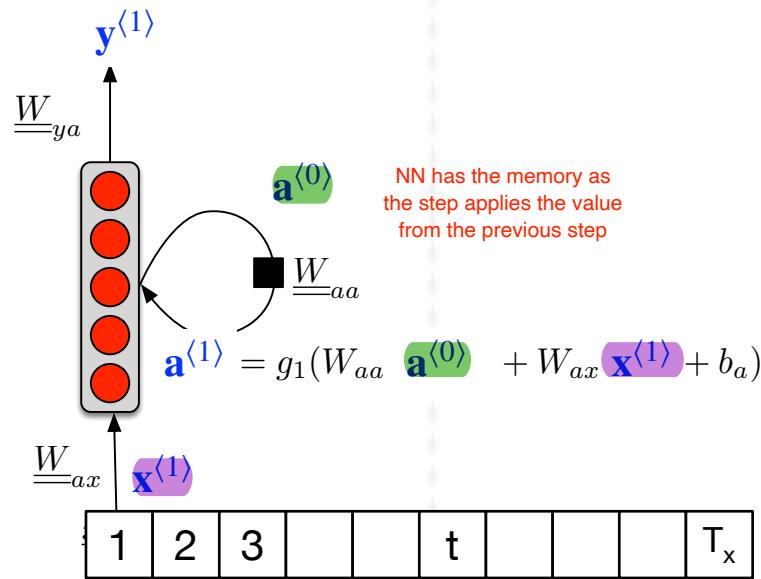
What is an RNN ?



- Parameters (shared from all time-steps):
 - \mathbf{W}_{ax} from $\mathbf{x}^{(t)}$ → $\mathbf{a}^{(t)}$,
 - \mathbf{W}_{aa} from $\mathbf{a}^{(t-1)}$ → $\mathbf{a}^{(t)}$,
 - \mathbf{W}_{ya} from $\mathbf{a}^{(t)}$ → $\mathbf{y}^{(t)}$

Recurrent Neural Network for Sequential Data

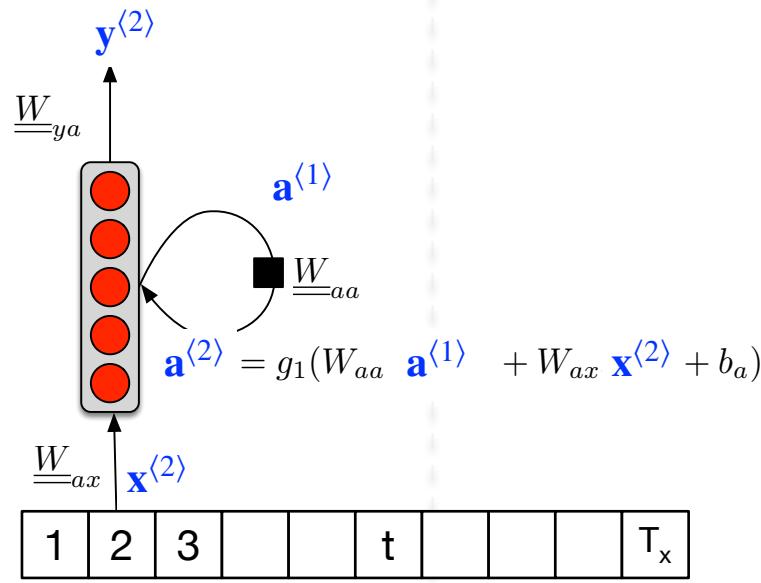
What is an RNN ?



- "**Rolled**" version of RNN (the real one, only one network)

Recurrent Neural Network for Sequential Data

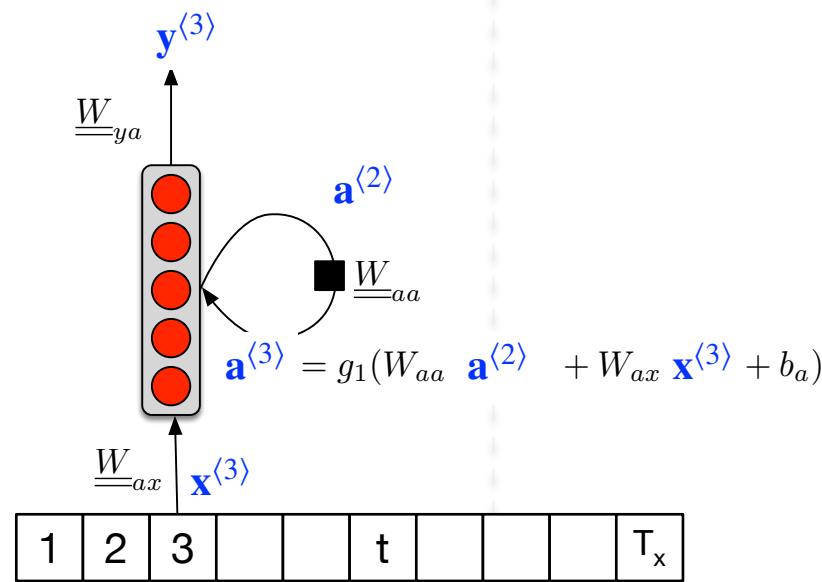
What is an RNN ?



- "**Rolled**" version of RNN (the real one, only one network)

Recurrent Neural Network for Sequential Data

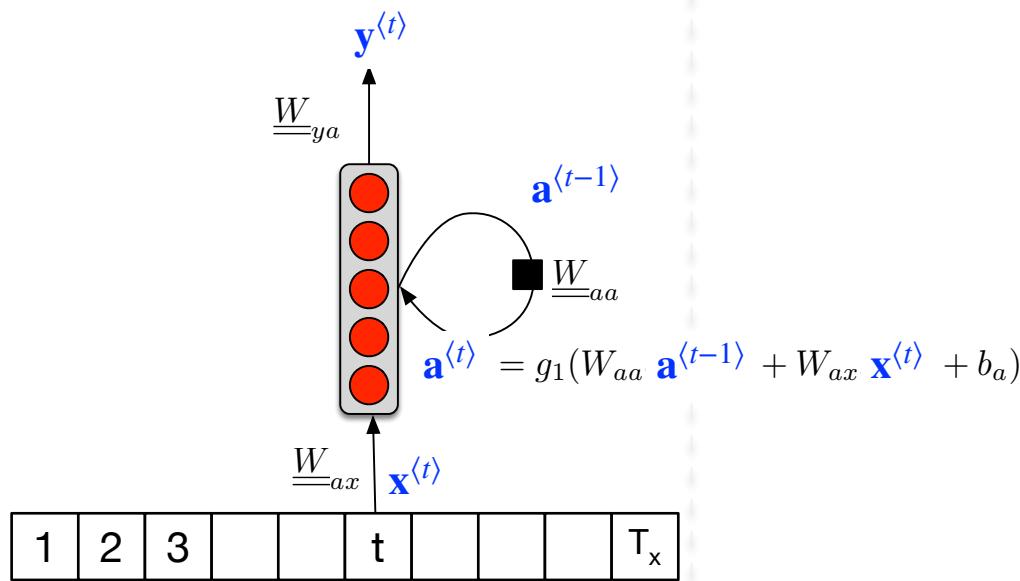
What is an RNN ?



- "**Rolled**" version of RNN (the real one, only one network)

Recurrent Neural Network for Sequential Data

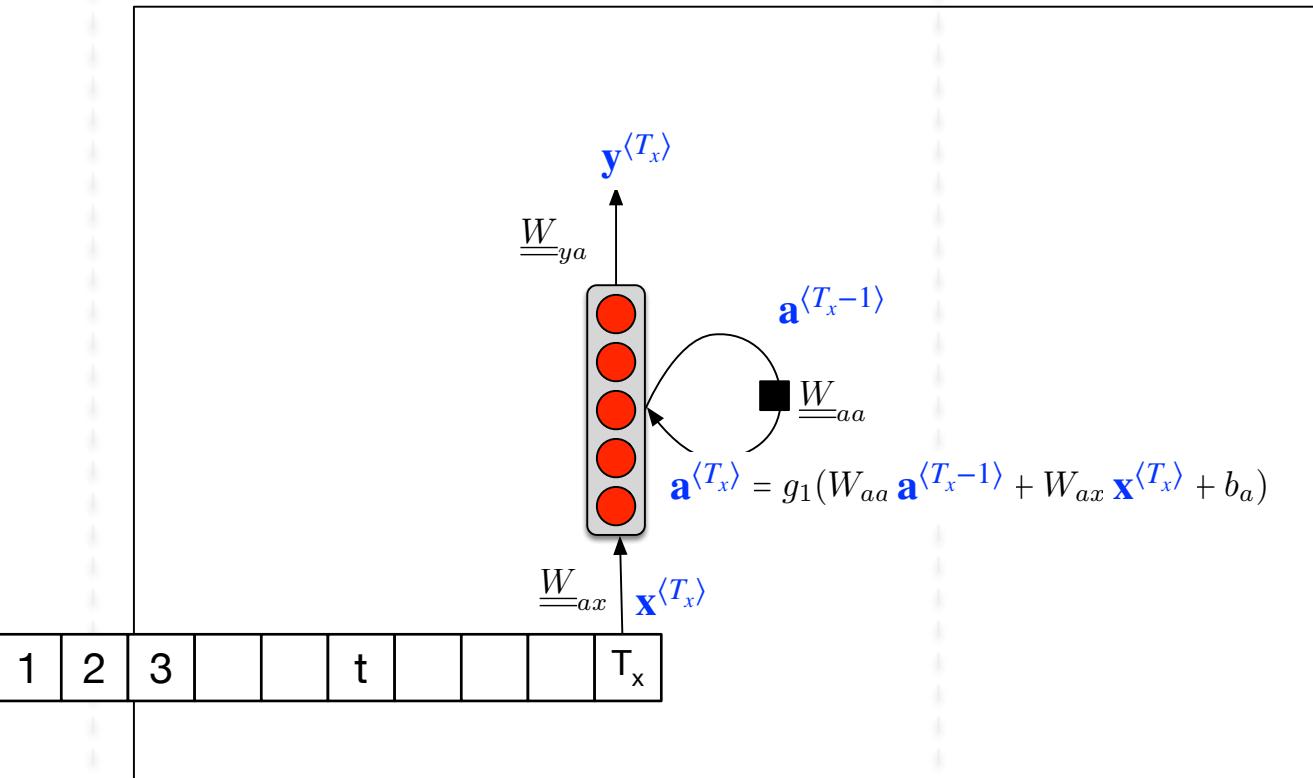
What is an RNN ?



- "**Rolled**" version of RNN (the real one, only one network)

Recurrent Neural Network for Sequential Data

What is an RNN ?



- "Rolled" version of RNN (the real one, only one network)

Recurrent Neural Network for Sequential Data

What is an RNN ?

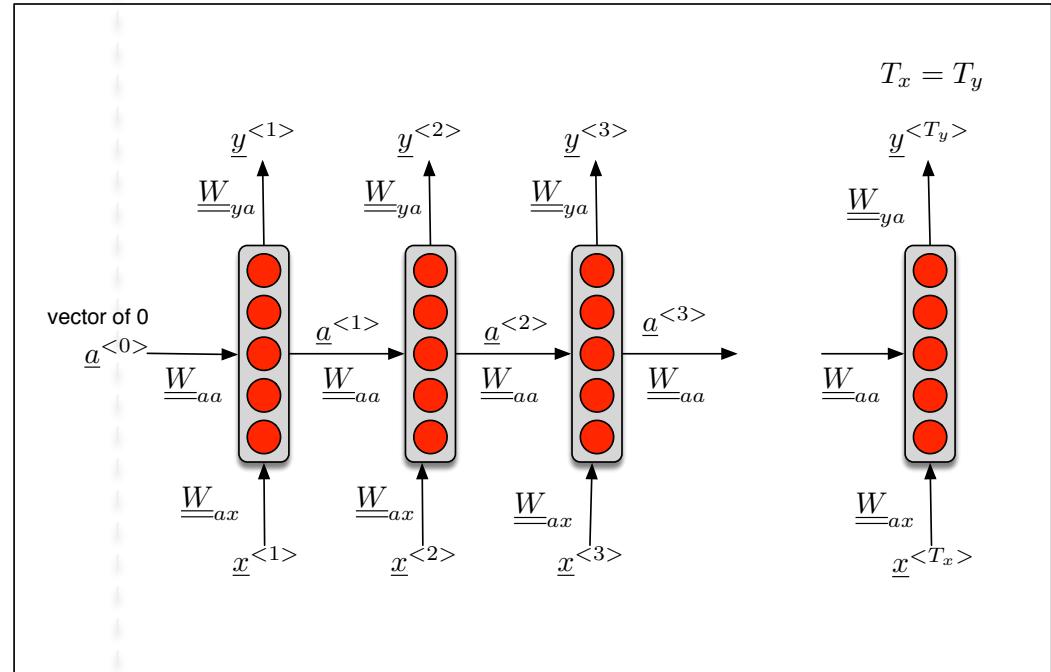
- **Forward propagation**

$$\mathbf{a}^{(0)} = 0 \quad \text{processing a value from the previous step}$$

$$\mathbf{a}^{(t)} = g_1 \left(\mathbf{W}_{aa} \mathbf{a}^{(t-1)} + \mathbf{W}_{ax} \mathbf{x}^{(t)} + \mathbf{b}_a \right)$$

$$\hat{\mathbf{y}}^{(t)} = g_2 \left(\mathbf{W}_{ya} \mathbf{a}^{(t)} + \mathbf{b}_y \right) \quad \text{no regression on the output}$$

...



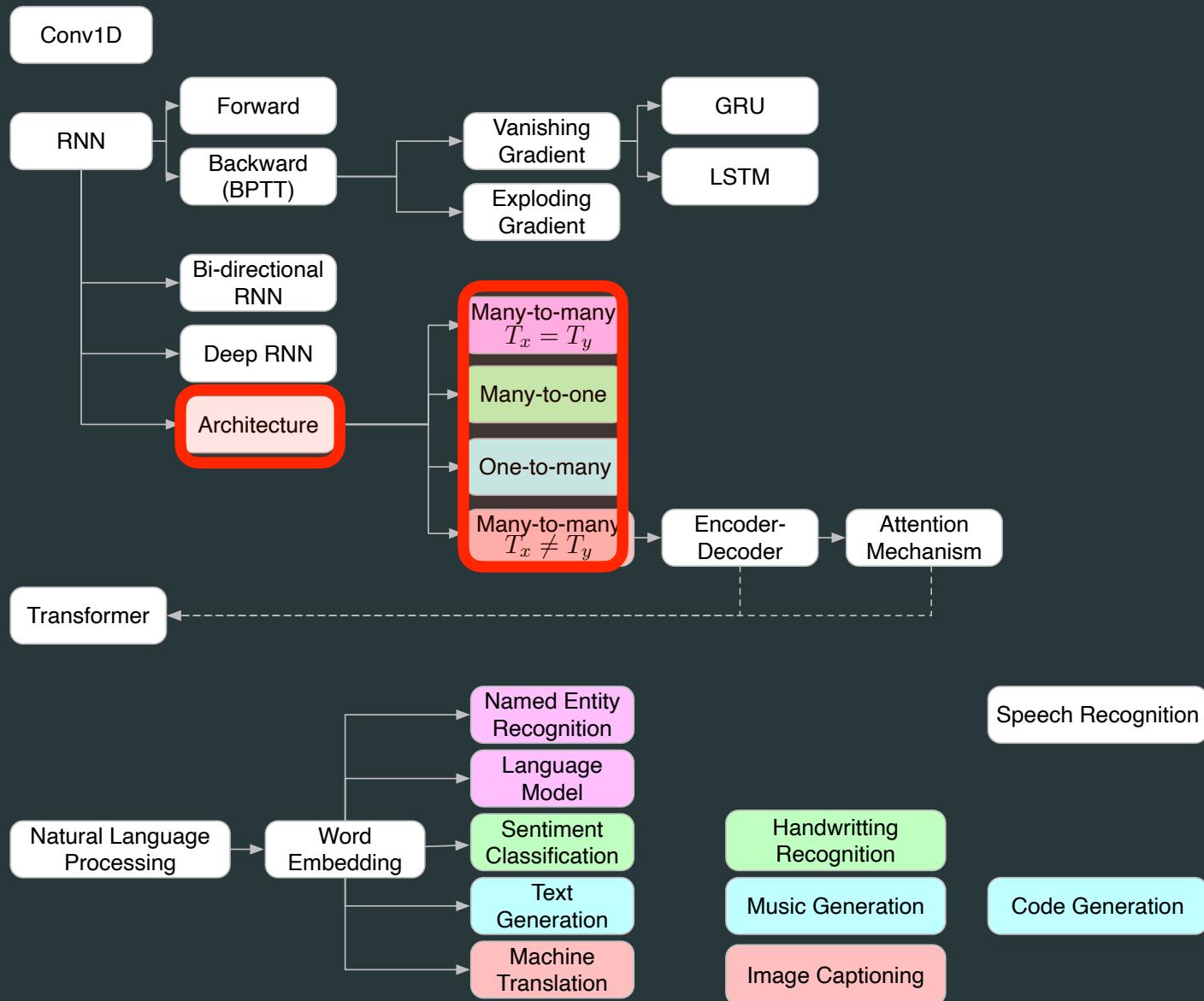
- Other possible notations

$$[\mathbf{W}_{aa} \mid \mathbf{W}_{ax}] \begin{bmatrix} \mathbf{a}^{(t-1)} \\ \mathbf{x}^{(t)} \end{bmatrix}$$

$$\mathbf{a}^{(t)} = g_1 \left(\mathbf{W}_a [\mathbf{a}^{(t-1)}; \mathbf{x}^{(t)}] + \mathbf{b}_a \right)$$

$$\hat{\mathbf{y}}^{(t)} = g_2 \left(\mathbf{W}_y \mathbf{a}^{(t)} + \mathbf{b}_y \right)$$

Various types of sequential data



Various types of sequential data

Many-to-Many : $T_x = T_y$.

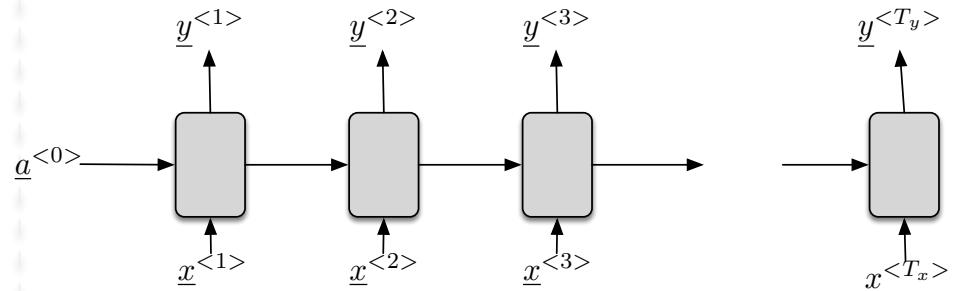
Input \underline{x}	Output \underline{y}	Type	Examples
sequence $T_x > 1$	sequence $T_y = T_x$	Many-To-Many	Named entity recognition

task is to assign each word to a class

In 1917, Einstein applied the general theory of relativity to model the large-scale structure of the universe. He was visiting the United States when Adolf Hitler came to power in 1933 and did not go back to Germany, where he had been a professor at the Berlin Academy of Sciences. He settled in the U.S., becoming an American citizen in 1940. On the eve of World War II, he endorsed a letter to President Franklin D. Roosevelt alerting him to the potential development of "extremely powerful bombs of a new type" and recommending that the U.S. begin similar research. This eventually led to what would become the Manhattan Project. Einstein supported defending the Allied forces, but largely denounced using the new discovery of nuclear fission as a weapon. Later, with the British philosopher Bertrand Russell, Einstein signed the Russell-Einstein Manifesto, which highlighted the danger of nuclear weapons. Einstein was affiliated with the Institute for Advanced Study in Princeton, New Jersey, until his death in 1955.

Tag colours:
LOCATION TIME PERSON ORGANIZATION MONEY PERCENT DATE

US is the location but united and states are not

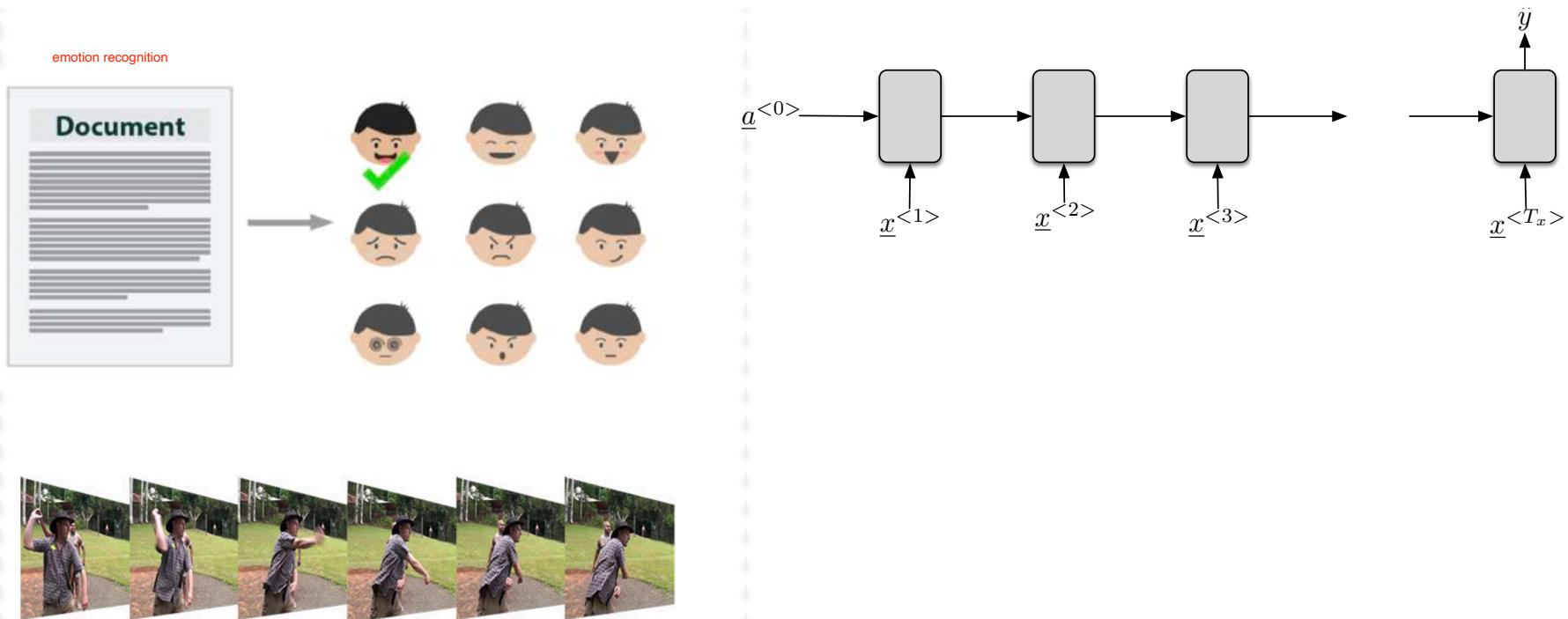


len of output = len of input

Various types of sequential data

Many-to-One : $T_x > 1, T_y = 1$.

Input x	Output y	Type	Examples
sequence $T_x > 1$	single $T_y = 1$	Many-To-One	Sentiment analysis, Video activity detection

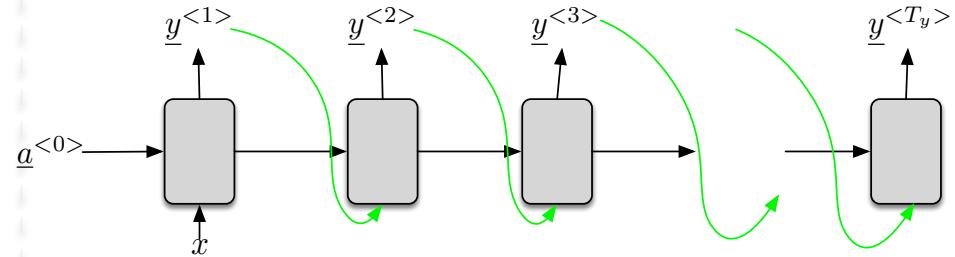


Various types of sequential data

One-to-Many : $T_x = 1, T_y > 1$.

Input x	Output y	Type	Examples
single $T_x = 1$	sequence, $T_y > 1$	One-To-Many	Text generation, music generation

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 Guangzhou's sovereignty. His generals were the powerful ruler of the Portugal in the [[Protestant Imminences]], 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 sloop of Syrian influence that was a famous German movement based on a more popular servitious, 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)[\(\[PJS\]\(http://www.humah.yahoo.com/guardian\)\)](http://www.humah.yahoo.com/guardian)
cmf/7754800786d17551963s89.htm Official economics Adjunct for the Nazism, Montgomery was swear to advance to the resources for those Socialism's rule, was starting to signing a major triad of aid exile.}}



AI-music, 192x96x576, 193 epochs, 805 seconds, 7 training sets

Russell E Glaue



Various types of sequential data

Many-to-Many : $T_x \neq T_y$.

Input \underline{x}	Output \underline{y}	Type	Examples
sequence $T_x > 1$	sequence $T_y > 1, T_y \neq T_x$	Many-To-Many	Automatic Speech Recognition, Machine translation

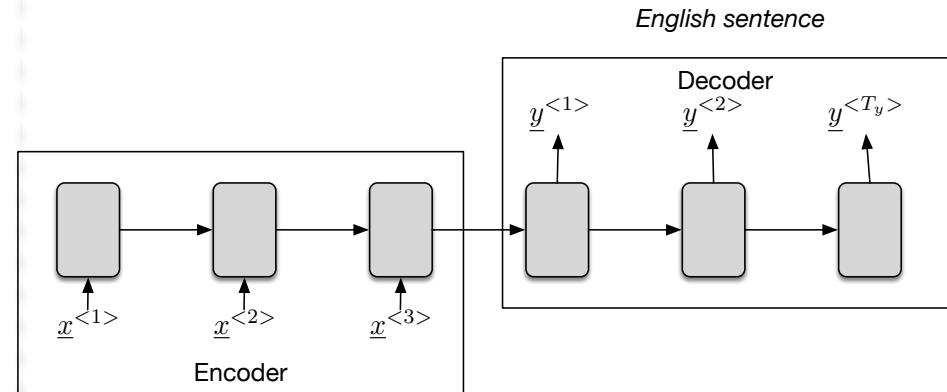
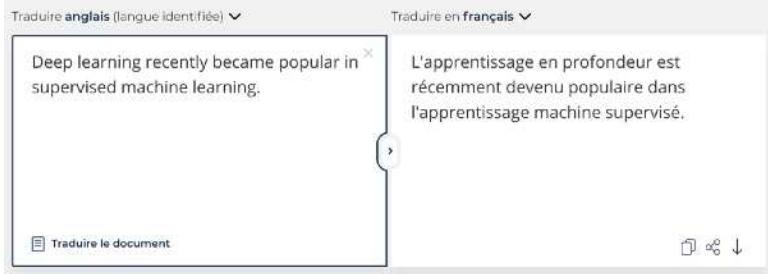
Traduire anglais (langue identifiée) ▾

Deep learning recently became popular in supervised machine learning.

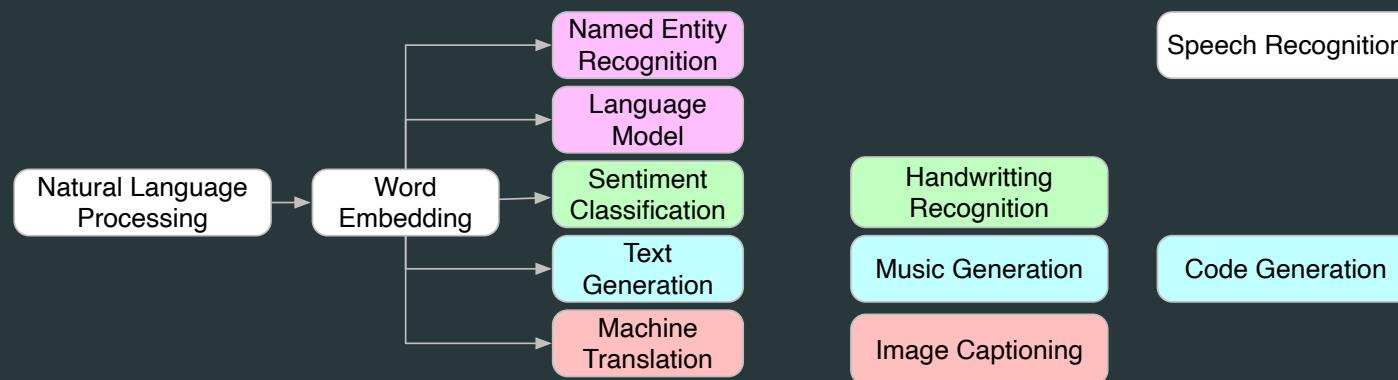
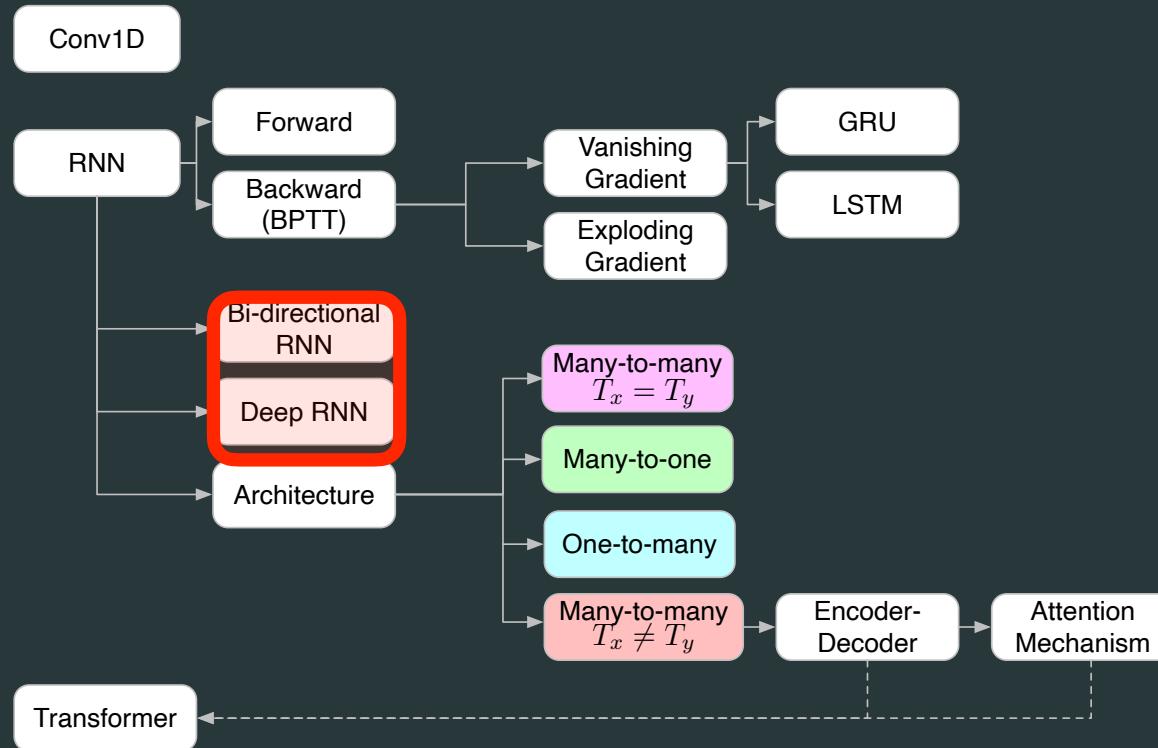
Traduire en français ▾

L'apprentissage en profondeur est récemment devenu populaire dans l'apprentissage machine supervisé.

Traduire le document

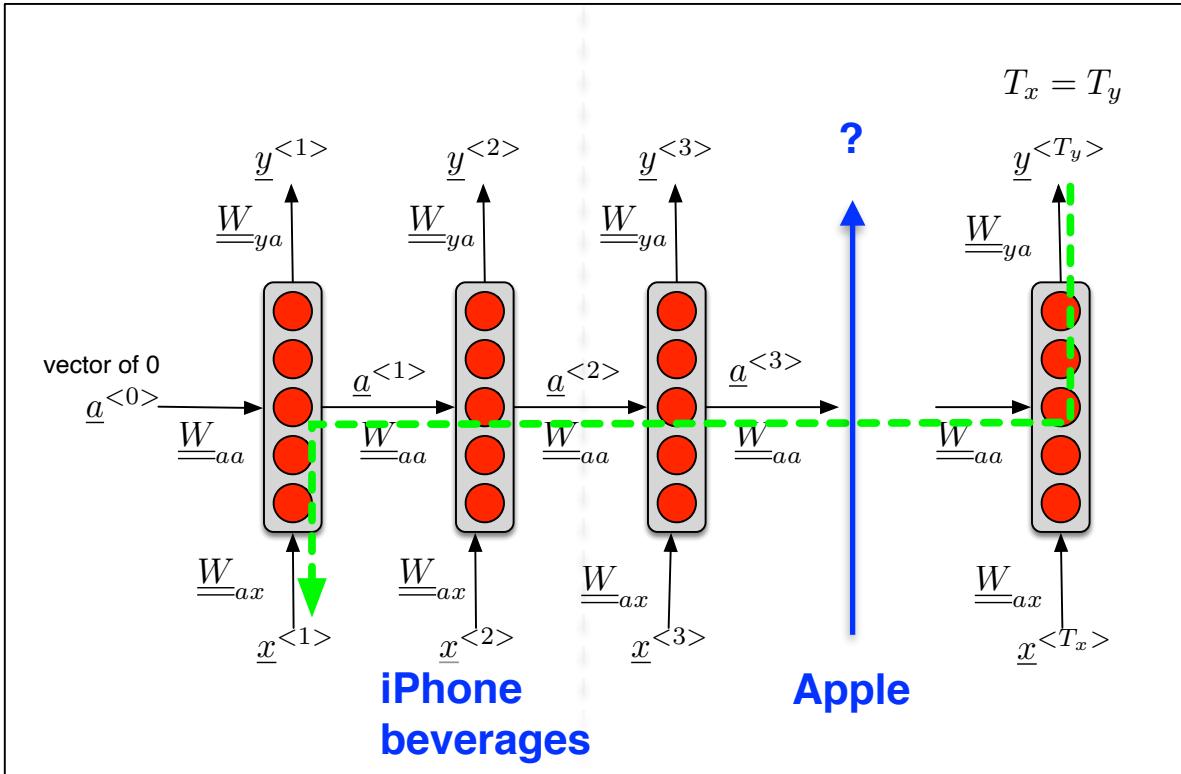


Different architectures



Different architectures

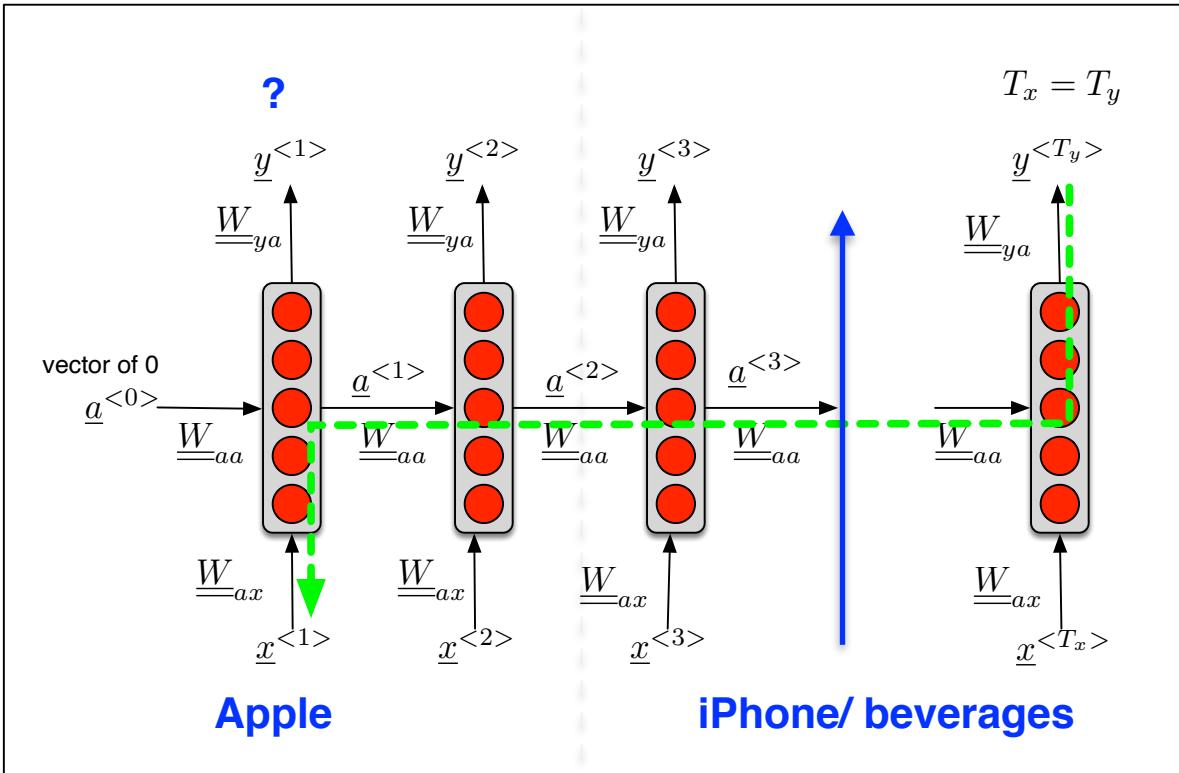
RNN



- RNN scans data from left to right \Rightarrow can do skip connections but only in the past
- Question: is "**Apple**" a named entity ?
 - "The new **iPhone** is sold in **Apple** stores."
 - "A refreshing and healthy **beverages** is an **Apple** juice."
- \Rightarrow we can predict (since RNN depends on the past)

Different architectures

RNN



- RNN scans data from left to right \Rightarrow can do skip connections but only in the past
- Question: is "**Apple**" a named entity ?
 - "**Apple** stores sale the new **iPhone**."
 - "**Apple** juice is a refreshing and healthy **beverages**."
- \Rightarrow we cannot predict (since RNN does not depend on the future)

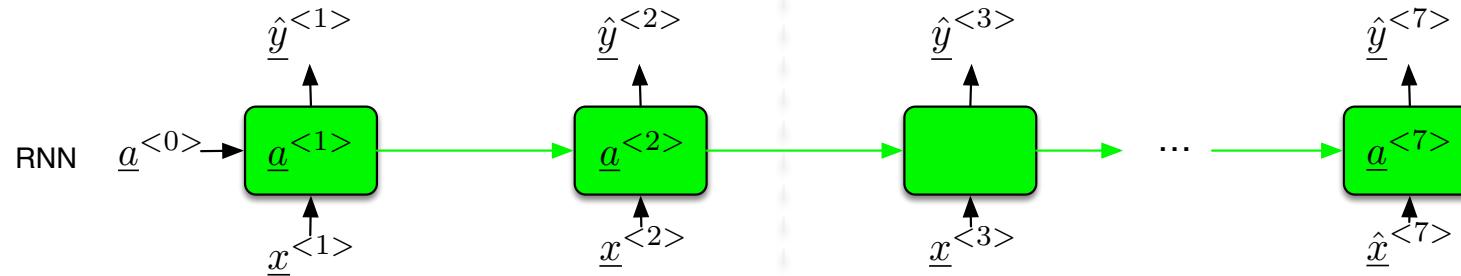
here Apple before its class --> RNN cannot predict

Different architectures

Bi-directional RNN

– Standard RNN:

- Left-Right hidden states: $a^{(t)}$

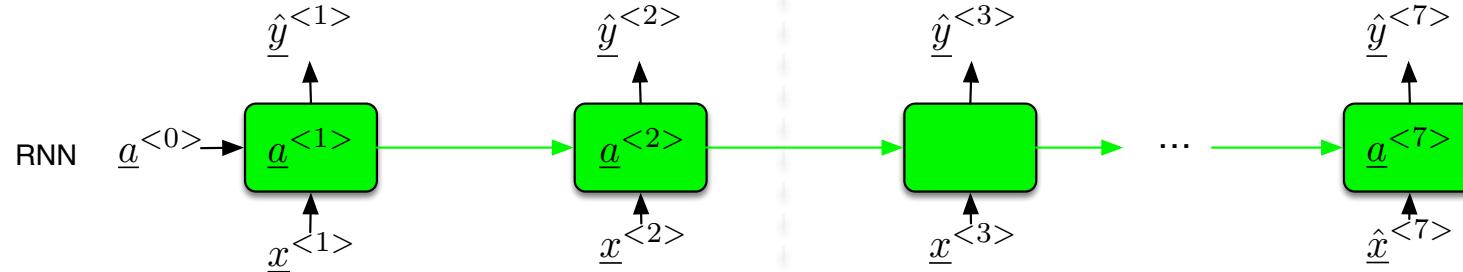


Different architectures

Bi-directional RNN

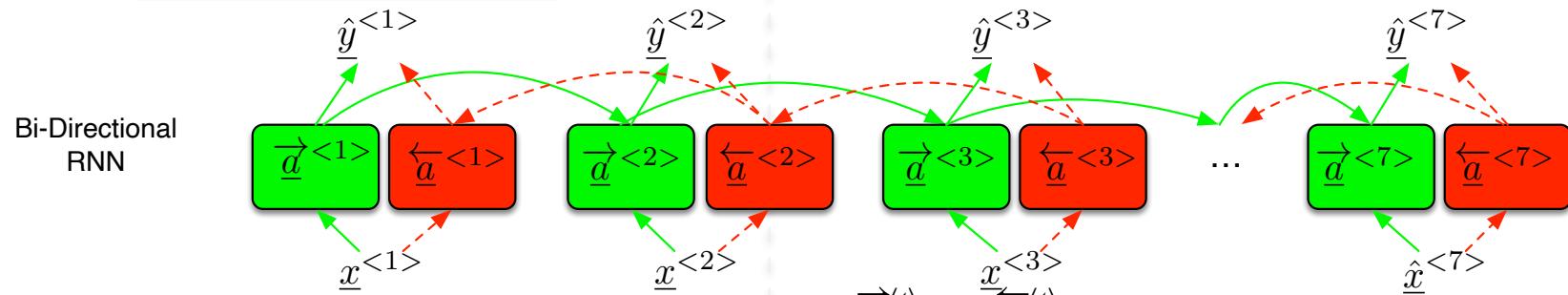
– Standard RNN:

- Left-Right hidden states: $\underline{a}^{(t)}$



– Bi-directional RNN:

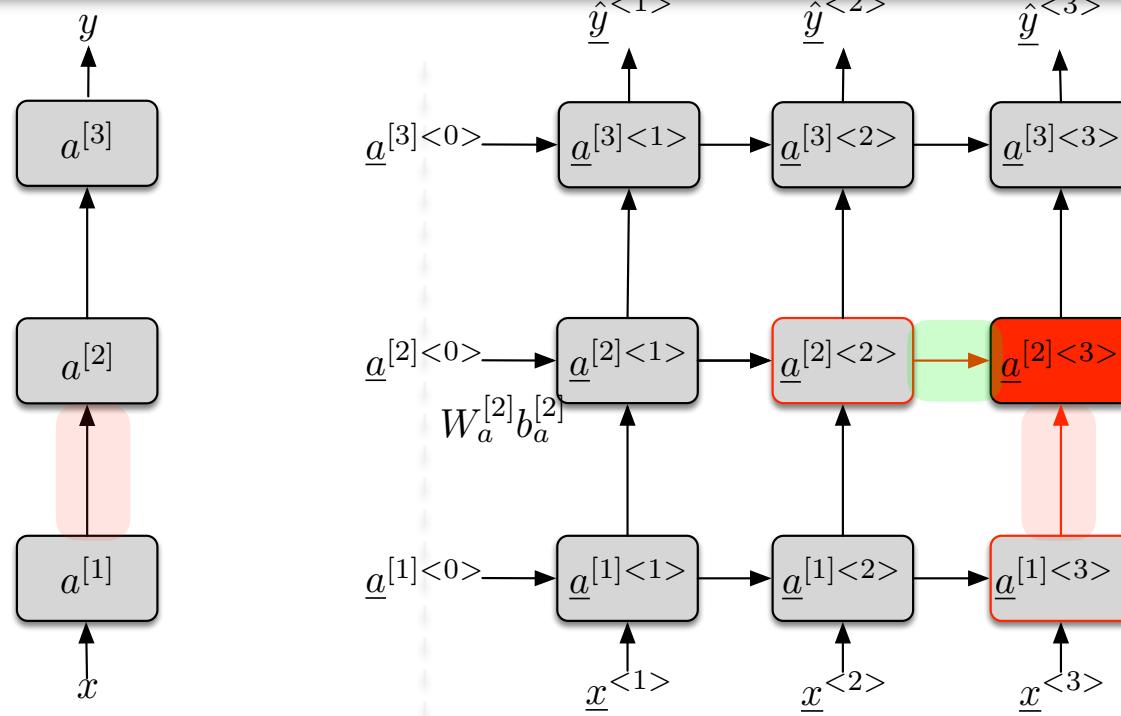
- Left-Right hidden states: $\overrightarrow{a}^{(t)}$
- Right-Left hidden states: $\overleftarrow{a}^{(t)}$



- The prediction is done from the concatenation of $\overrightarrow{a}^{(t)}$ and $\overleftarrow{a}^{(t)}$
- $\hat{y}^{(t)} = g(W_y[\overrightarrow{a}^{(t)}, \overleftarrow{a}^{(t)}] + b_y)$

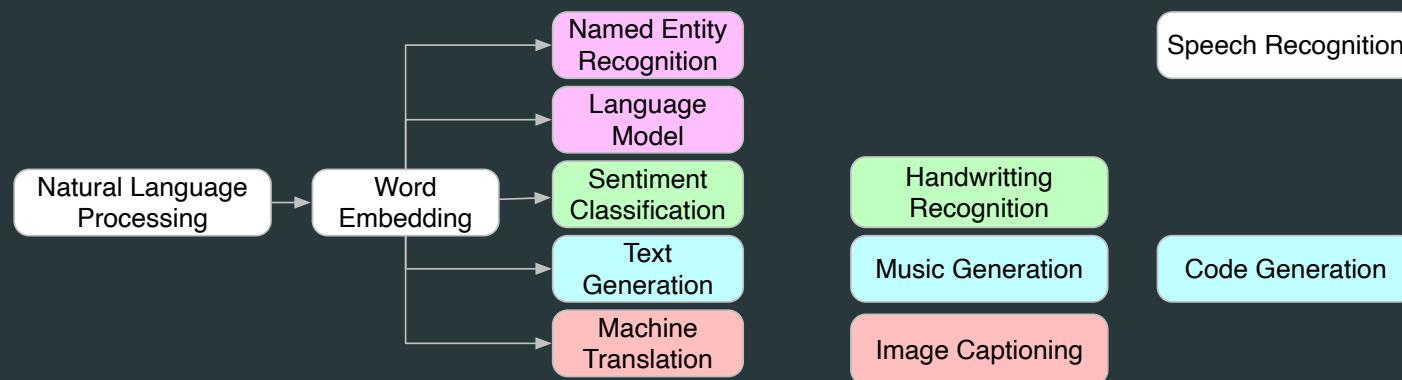
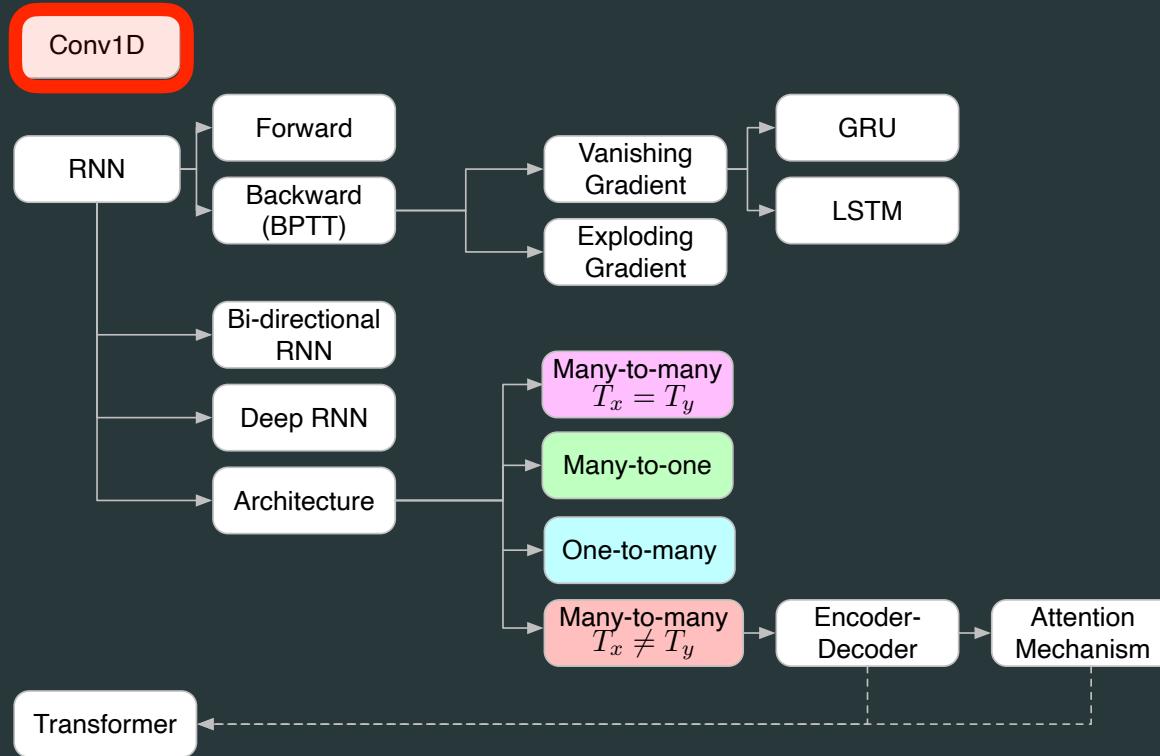
Different architectures

Deep-RNN



- For the layer $[1]$, the inputs of the cell at time $\langle t \rangle$ are
 - the input at the current time $\langle t \rangle$: $x^{<t>} = a^{[0]\langle t \rangle}$
 - the value of the cell at the previous time $\langle t - 1 \rangle$: $a^{[1]\langle t-1 \rangle}$
 - For the layer $[l]$, the inputs of the cell at time $\langle t \rangle$ are
 - the value of the cell of the previous layer $[l - 1]$ at the current time $\langle t \rangle$: $a^{[l-1]\langle t \rangle}$
 - the value of the cell of the current layer $[l]$ at the previous time $\langle t - 1 \rangle$: $a^{[l]\langle t-1 \rangle}$
- $$a^{[l]\langle t \rangle} = g \left(W_a^{[l]} [a^{[l]\langle t-1 \rangle}, a^{[l-1]\langle t \rangle}] + b_a^{[l]} \right)$$

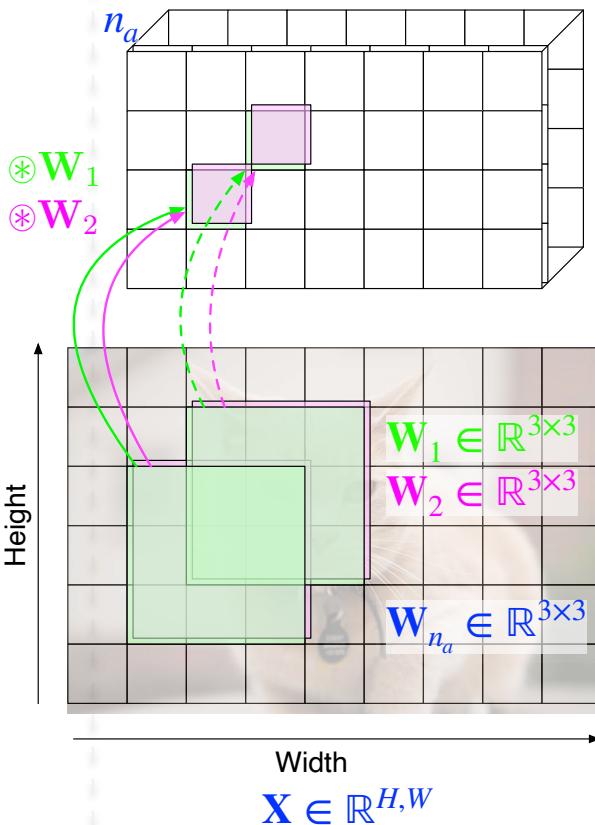
Alternative to RNN



Alternative to RNN

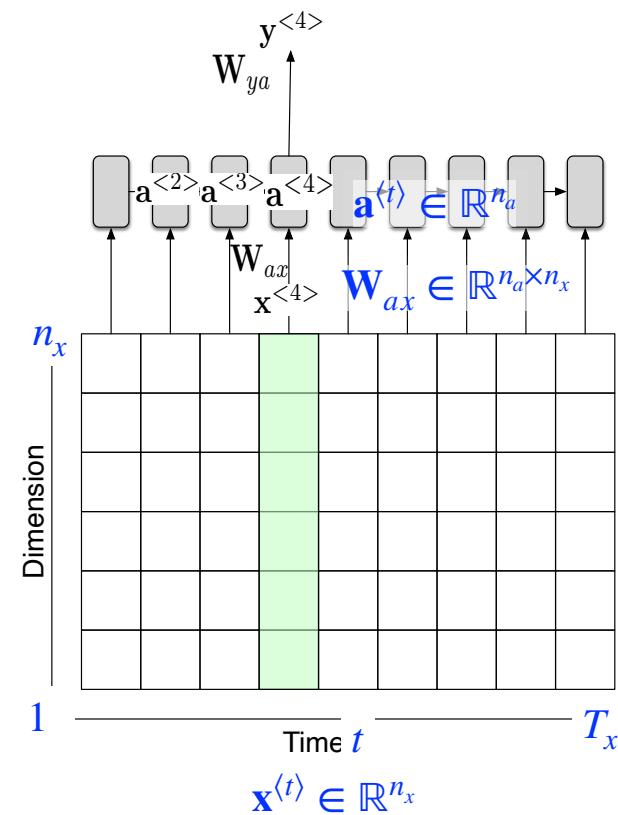
Using 1D convolution instead of RNN

Conv2D



?

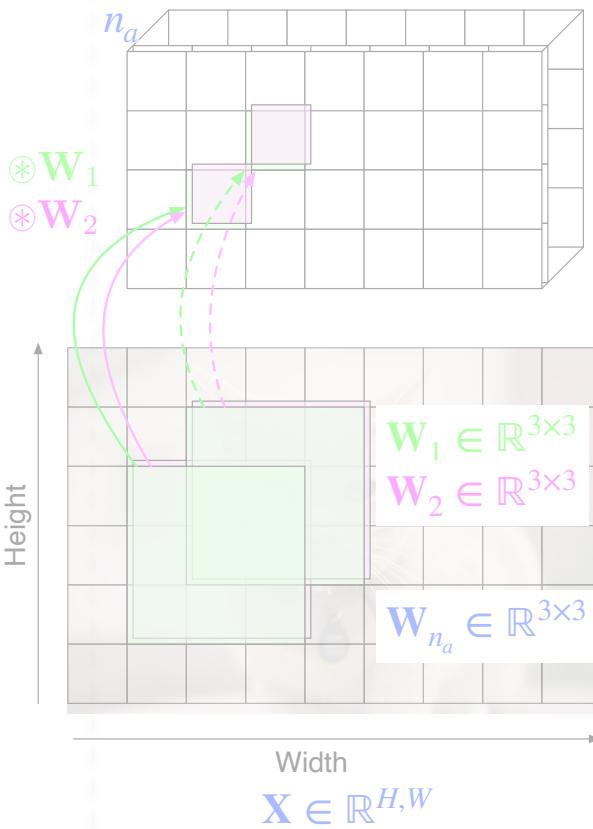
RNN



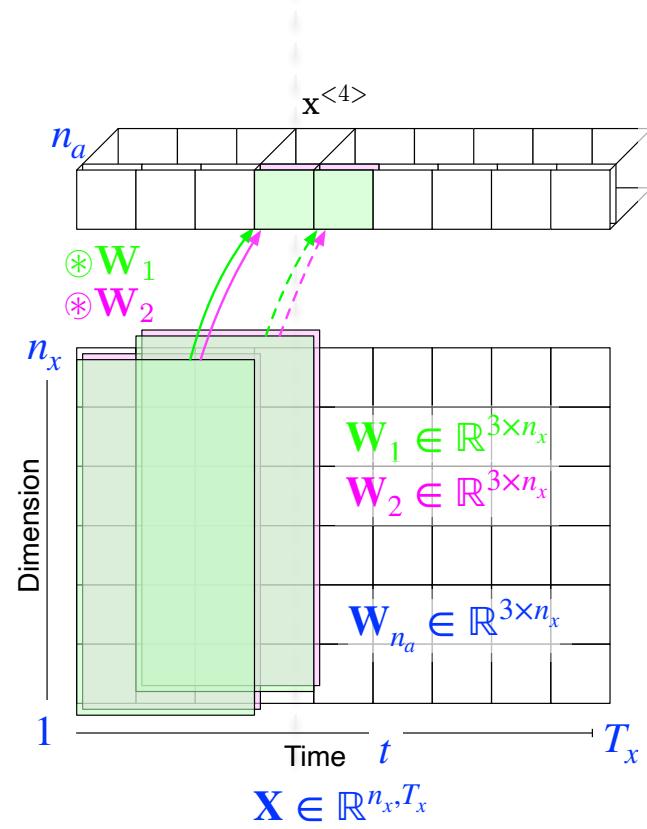
Alternative to RNN

Using 1D convolution instead of RNN

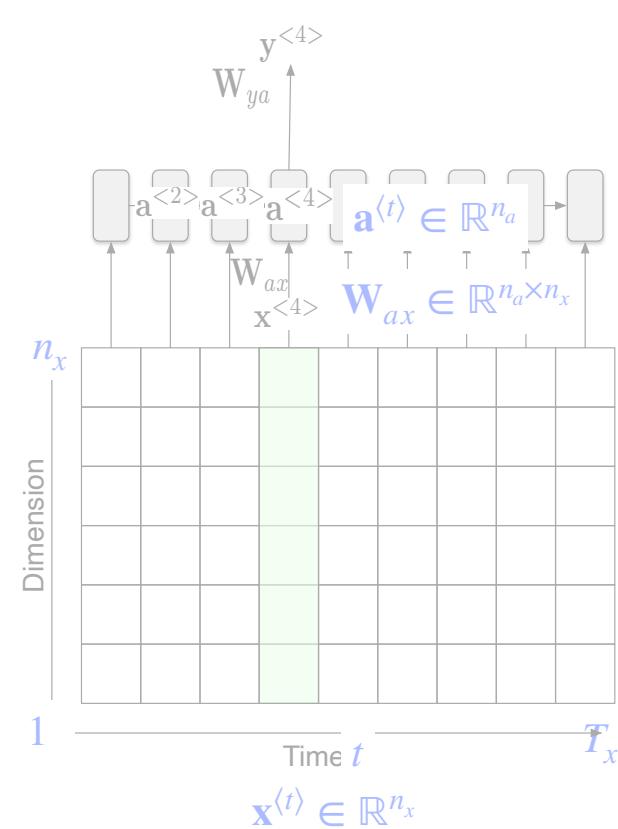
Conv2D



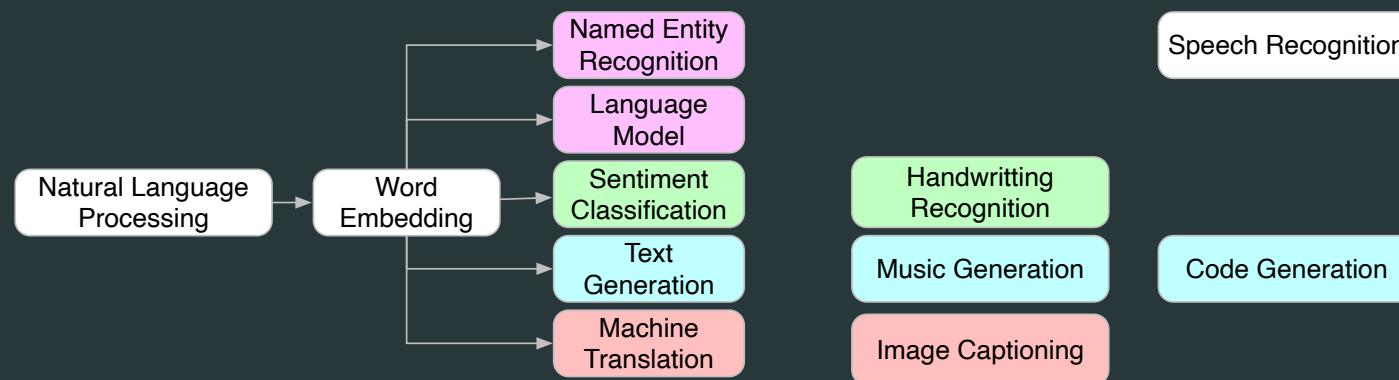
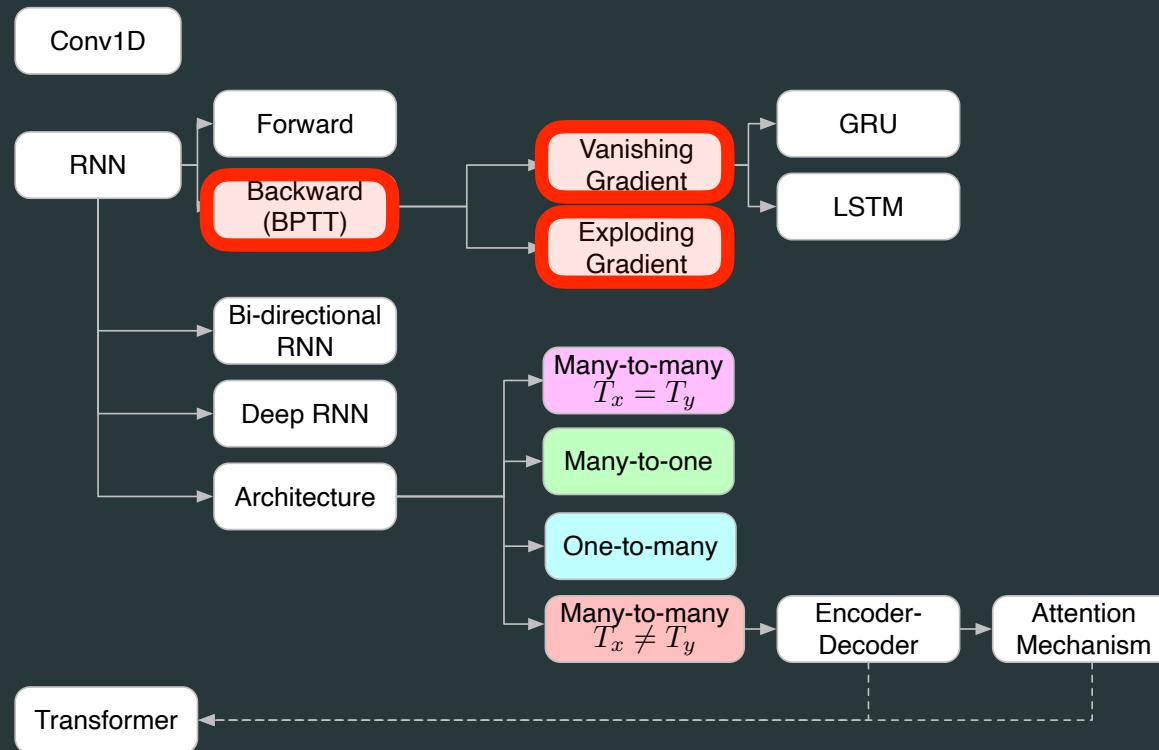
Conv1D



RNN



Back Propagation Through Time (BPTT)



Back Propagation Through Time (BPTT)

Forward pass

- Compute $\mathbf{a}^{(t)}, \hat{y}^{(t)}, \mathcal{L}^{(t)}, \mathcal{L}$

– Forward

$$\mathbf{a}^{(t)} = g_a \left(\mathbf{W}_{ax} \mathbf{x}^{(t)} + \mathbf{W}_{aa} \mathbf{a}^{(t-1)} + \mathbf{b}_a \right)$$

$$\hat{y}^{(t)} = g_y \left(\mathbf{W}_{ya} \mathbf{a}^{(t)} + \mathbf{b}_y \right)$$

– Loss

- Loss for time $\langle t \rangle$: $\mathcal{L}^{(t)}(y^{(t)}, \hat{y}^{(t)})$

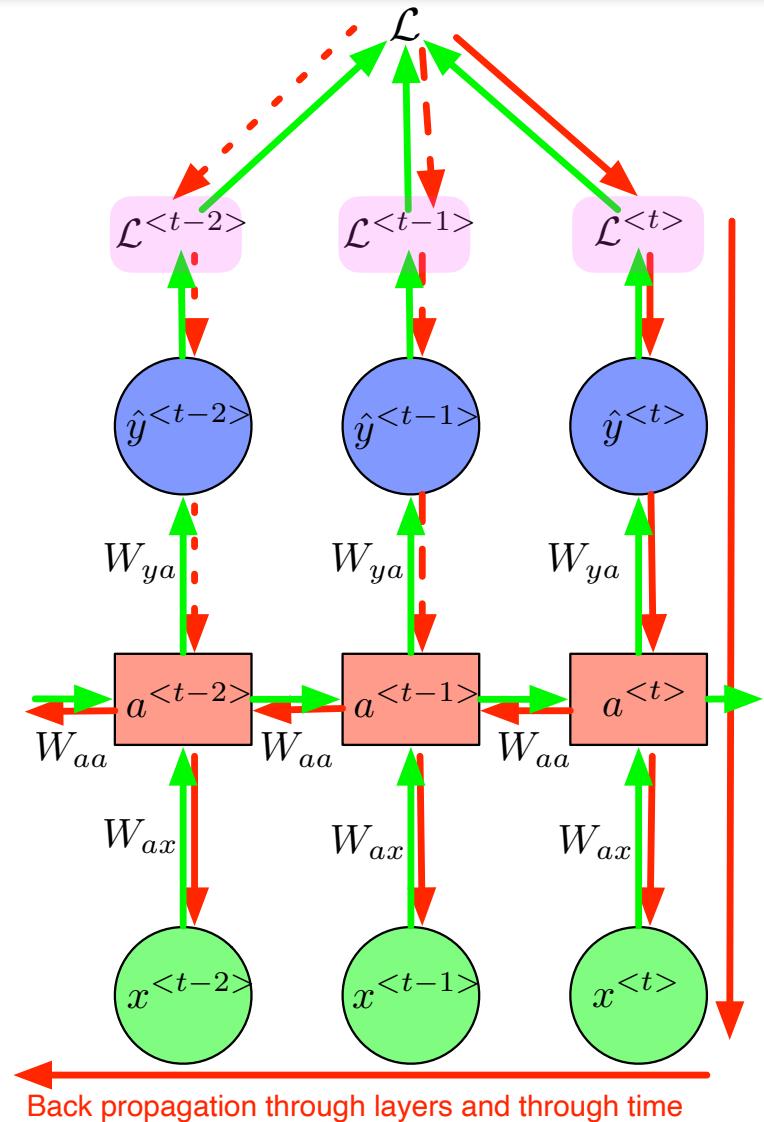
- $y^{(t)}$: ground-truth
- $\hat{y}^{(t)}$: prediction

- Total loss

$$\mathcal{L} = \sum_t \mathcal{L}^{(t)}(y^{(t)}, \hat{y}^{(t)})$$

– Gradients ?

- compute $\frac{\partial \mathcal{L}}{\partial W_{ya}}, \frac{\partial \mathcal{L}}{\partial W_{ax}}, \frac{\partial \mathcal{L}}{\partial W_{aa}}, \frac{\partial \mathcal{L}}{\partial b_y}, \frac{\partial \mathcal{L}}{\partial b_a}$
- All weights are shared across time steps !!!



Back Propagation Through Time (BPTT)

Backward: $\frac{\partial \mathcal{L}}{\partial W_{ya}}$

- Linearity

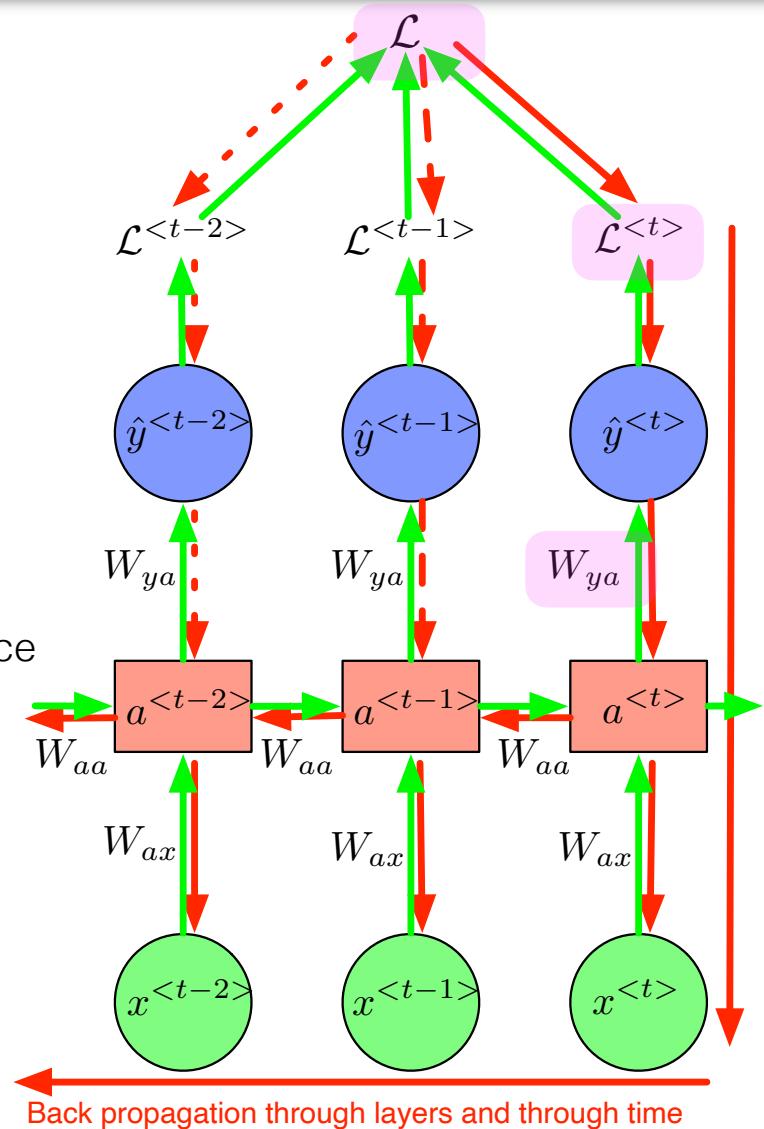
$$\frac{\partial \mathcal{L}}{\partial W_{ya}} = \sum_{t=1}^T \frac{\partial \mathcal{L}^{(t)}}{\partial W_{ya}}$$

- How much varying W_{ya} affect $\mathcal{L}^{(t)}$? : chain rule

$$\frac{\partial \mathcal{L}^{(t)}}{\partial W_{ya}} = \frac{\partial \mathcal{L}^{(t)}}{\partial \hat{y}^{(t)}} \frac{\partial \hat{y}^{(t)}}{\partial W_{ya}}$$

- Very simple since $\hat{y}^{(t)}$ depends on W_{ya} on only place
(indeed $a^{(t)}$ does not depend on W_{ya})

$$\hat{y}^{(t)} = g_y \left(\mathbf{W}_{ya} \mathbf{a}^{(t)} + \mathbf{b}_y \right)$$



Back Propagation Through Time (BPTT)

Backward: $\frac{\partial \mathcal{L}}{\partial W_{aa}} .$

- Linearity

$$\frac{\partial \mathcal{L}}{\partial W_{aa}} = \sum_{t=1}^T \frac{\partial \mathcal{L}^{(t)}}{\partial W_{aa}}$$

- How much varying W_{aa} affect $\mathcal{L}^{(t)}$? no chain rule

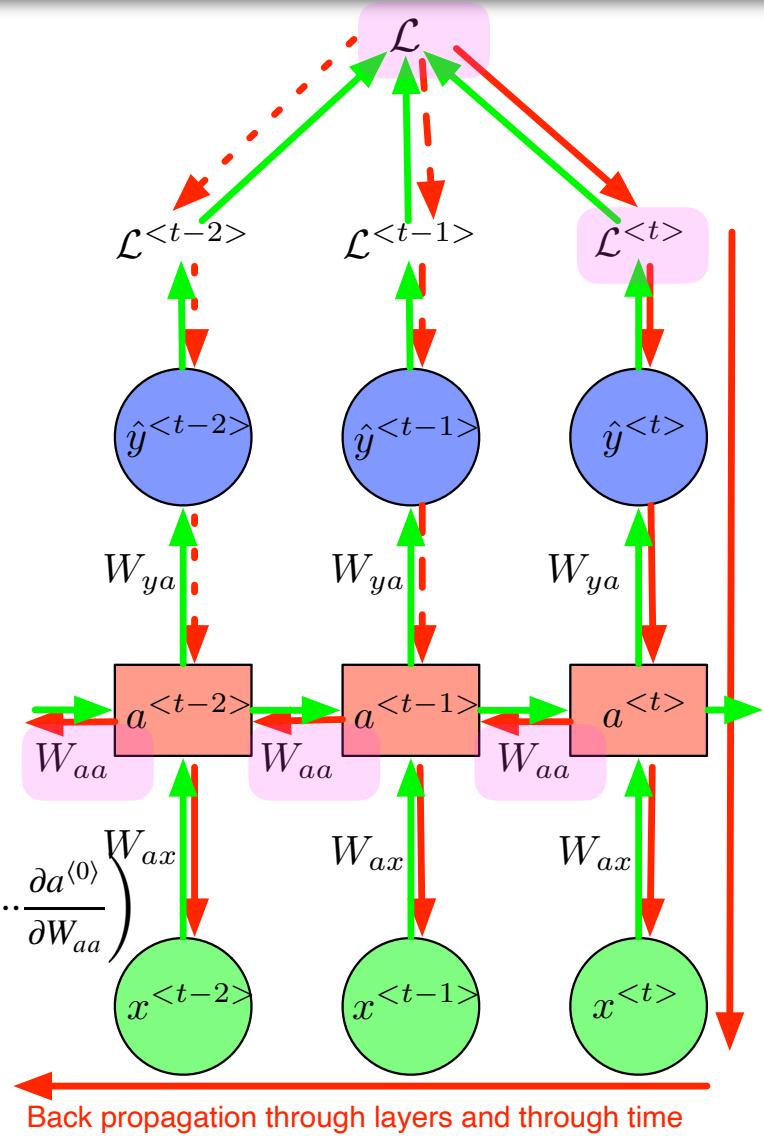
$$\frac{\partial \mathcal{L}^{(t)}}{\partial W_{aa}} \neq \frac{\partial \mathcal{L}^{(t)}}{\partial \hat{y}^{(t)}} \frac{\partial \hat{y}^{(t)}}{\partial a^{(t)}} \frac{\partial a^{(t)}}{\partial W_{aa}}$$

- because $a^{(t)}$ also depends on W_{aa} through $a^{(t-1)}$ which itself depends on ...

$$\mathbf{a}^{(t)} = g_a \left(\mathbf{W}_{ax} \mathbf{x}^{(t)} + \mathbf{W}_{aa} \mathbf{a}^{(t-1)} + \mathbf{b}_a \right)$$

- We use a formula of total derivative

$$\begin{aligned} \frac{\partial \mathcal{L}^{(t)}}{\partial W_{aa}} &= \frac{\partial \mathcal{L}^{(t)}}{\partial \hat{y}^{(t)}} \frac{\partial \hat{y}^{(t)}}{\partial a^{(t)}} \left(\frac{\partial a^{(t)}}{\partial W_{aa}} + \frac{\partial a^{(t)}}{\partial a^{(t-1)}} \frac{\partial a^{(t-1)}}{\partial W_{aa}} + \dots + \frac{\partial a^{(t)}}{\partial a^{(t-1)}} \dots \frac{\partial a^{(0)}}{\partial W_{aa}} \right) \\ &\quad \sum_{k=0}^t \frac{\partial a^{(t)}}{\partial a^{(t-1)}} \dots \frac{\partial a^{(k+1)}}{\partial a^{(k)}} \frac{\partial a^{(k)}}{\partial W_{aa}} \\ &= \sum_{k=0}^t \left(\prod_{i=k+1}^t \frac{\partial a^{(i)}}{\partial a^{(i-1)}} \right) \frac{\partial a^{(k)}}{\partial W_{aa}} \end{aligned}$$



Back Propagation Through Time (BPTT)

Backward: $\frac{\partial \mathcal{L}}{\partial W_{ax}}$.

- Linearity

$$\frac{\partial \mathcal{L}}{\partial W_{ax}} = \sum_{t=1}^T \frac{\partial \mathcal{L}^{(t)}}{\partial W_{ax}}$$

- How much varying W_{ax} affect $\mathcal{L}^{(t)}$? no chain rule

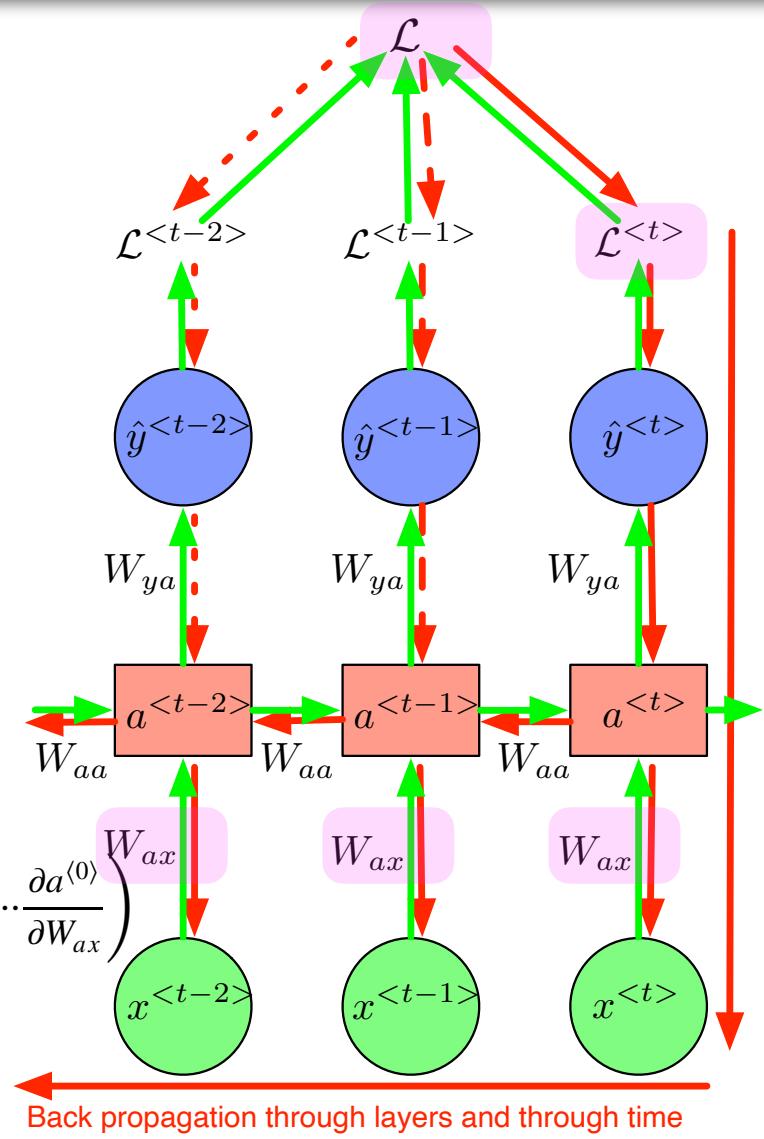
$$\frac{\partial \mathcal{L}^{(t)}}{\partial W_{ax}} \neq \frac{\partial \mathcal{L}^{(t)}}{\partial \hat{y}^{(t)}} \frac{\partial \hat{y}^{(t)}}{\partial a^{(t)}} \frac{\partial a^{(t)}}{\partial W_{ax}}$$

- because $a^{(t)}$ also depends on W_{ax} through $a^{(t-1)}$ which itself depends on ...

$$\mathbf{a}^{(t)} = g_a \left(\mathbf{W}_{ax} \mathbf{x}^{(t)} + \mathbf{W}_{aa} \mathbf{a}^{(t-1)} + \mathbf{b}_a \right)$$

- We use a formula of total derivative

$$\begin{aligned} \frac{\partial \mathcal{L}^{(t)}}{\partial W_{ax}} &= \frac{\partial \mathcal{L}^{(t)}}{\partial \hat{y}^{(t)}} \frac{\partial \hat{y}^{(t)}}{\partial a^{(t)}} \left(\frac{\partial a^{(t)}}{\partial W_{ax}} + \frac{\partial a^{(t)}}{\partial a^{(t-1)}} \frac{\partial a^{(t-1)}}{\partial W_{ax}} + \dots + \frac{\partial a^{(t)}}{\partial a^{(t-1)}} \dots \frac{\partial a^{(0)}}{\partial W_{ax}} \right) \\ &\quad \sum_{k=0}^t \frac{\partial a^{(t)}}{\partial a^{(t-1)}} \dots \frac{\partial a^{(k+1)}}{\partial a^{(k)}} \frac{\partial a^{(k)}}{\partial W_{ax}} \\ &= \sum_{k=0}^t \left(\prod_{i=k+1}^t \frac{\partial a^{(i)}}{\partial a^{(i-1)}} \right) \frac{\partial a^{(k)}}{\partial W_{ax}} \end{aligned}$$



Back propagation through layers and through time

Vanishing and exploding gradients

- Vanishing gradient
 - In MLP: very deep NN
 - very difficult to propagate back the gradient to affect the weights of the **early** Layers
 - In RNN: **very difficult to learn long-term dependency**
 - The student, which already followed 6 hours of lessons, was tired.
 - The students, which already followed 6 hours of lessons, were tired.

Back Propagation Through Time (BPTT)

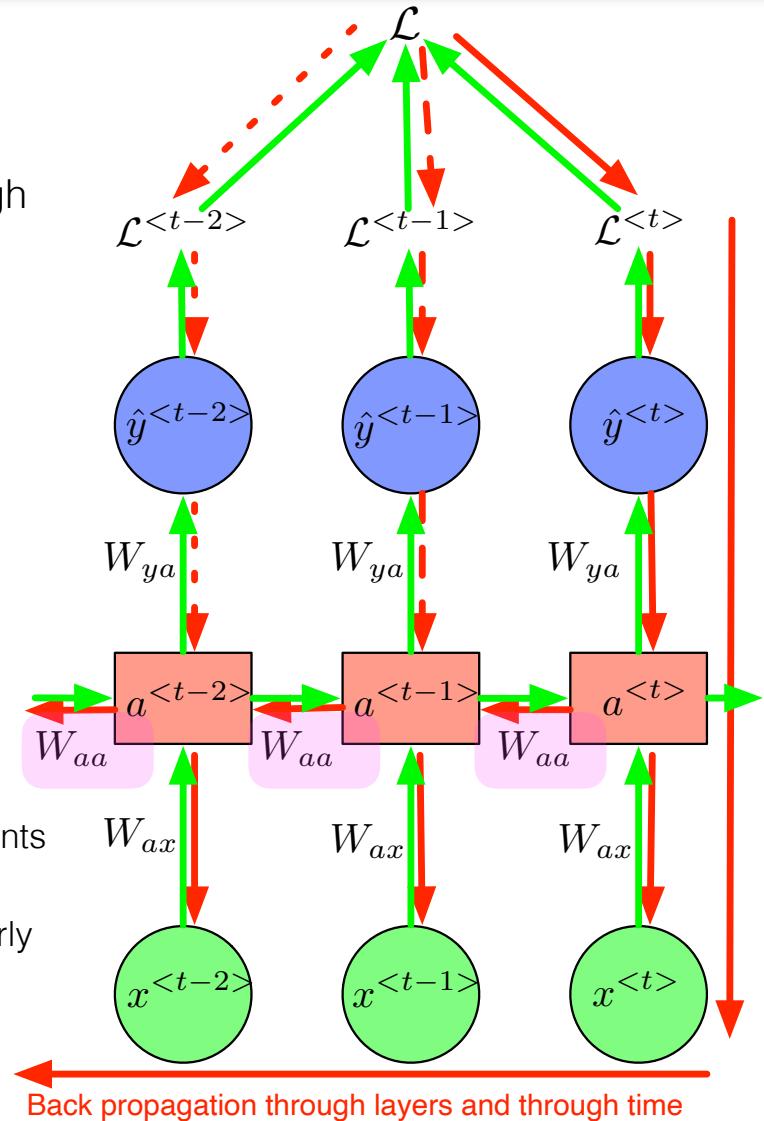
Vanishing and exploding gradients

- To train an RNN we need to back-propagate through layers and through time

$$\frac{\partial \mathcal{L}}{\partial W_{aa}} = \sum_{t=1}^T \frac{\partial \mathcal{L}^{(t)}}{\partial W_{aa}}$$

$$\frac{\partial \mathcal{L}^{(t)}}{\partial W_{aa}} = \sum_{k=0}^t \left(\prod_{i=k+1}^t \frac{\partial a^{(i)}}{\partial a^{(i-1)}} \right) \frac{\partial a^{(k)}}{\partial W_{aa}}$$

- each term in the sum is the contribution of
 - a state at time $\langle k \rangle$
 - to the gradient of the loss at time step $\langle t \rangle$
- the more steps between $\langle k \rangle$ and $\langle t \rangle$, the more elements in this product
- the values of these Jacobian matrices have particularly severe impact on the contributions of faraway steps



Back Propagation Through Time (BPTT)

Vanishing and exploding gradients

$$\frac{\partial \mathcal{L}}{\partial W_{aa}} = \sum_{t=1}^T \frac{\partial \mathcal{L}^{(t)}}{\partial W_{aa}}$$

$$\frac{\partial \mathcal{L}^{(t)}}{\partial W_{aa}} = \sum_{k=0}^t \left(\prod_{i=k+1}^t \frac{\partial a^{(i)}}{\partial a^{(i-1)}} \right) \frac{\partial a^{(k)}}{\partial W_{aa}}$$

_ Suppose only one hidden units, then $a^{(t)}$ is a scalar and consequently $\frac{\partial a^{(t)}}{\partial a^{(t-1)}}$ is also a scalar

- if $\left| \frac{\partial a^{(t)}}{\partial a^{(t-1)}} \right| < 1$, then the product goes to **0** exponentially fast
- if $\left| \frac{\partial a^{(t)}}{\partial a^{(t-1)}} \right| > 1$, then the product goes to **∞** exponentially fast

_ **Vanishing gradients:** $\left| \frac{\partial a^{(t)}}{\partial a^{(t-1)}} \right| < 1$

- contributions from **faraway steps** vanish and **don't affect the training**
- difficult to learn **long-range** dependencies

_ **Exploding gradients:** $\left| \frac{\partial a^{(t)}}{\partial a^{(t-1)}} \right| > 1$

- make the learning process **unstable**
- **gradient** could even become **NaN**

Back Propagation Through Time (BPTT)

Dealing with the **exploding gradient**

– Solution 1: **gradient clipping**

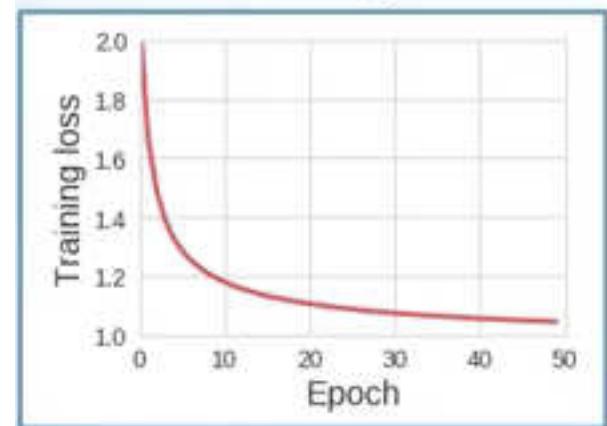
– gradient $d\theta = \frac{\partial \mathcal{L}}{\partial \theta}$ where θ are all the parameters

- if $\|\theta\| > \text{threshold}$

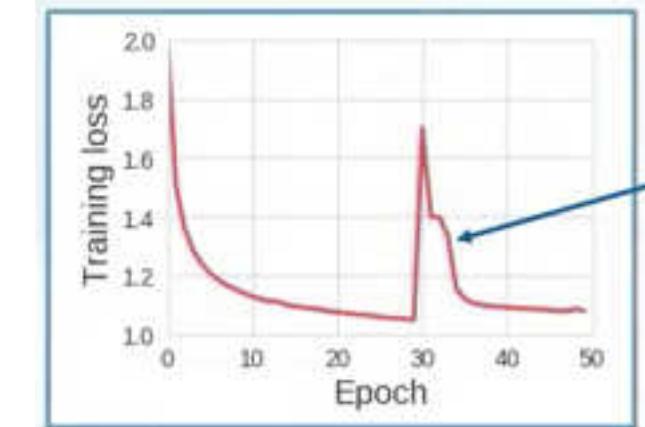
$$d\theta \leftarrow \frac{\text{threshold}}{\|\theta\|} d\theta$$

- clipping **doesn't change the direction** of the gradient but change its **length**
- we can clip only the norm of the part which causes the problem
- we choose the threshold manually: start with a large threshold and then reduce it

Stable learning curve



Unstable learning curve



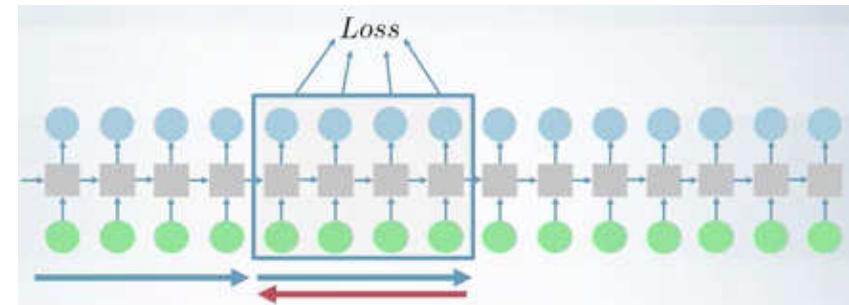
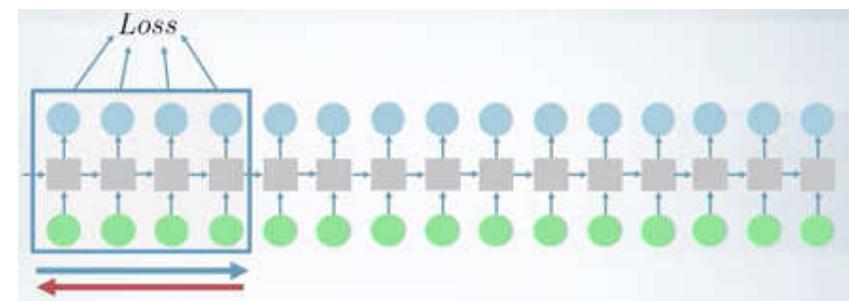
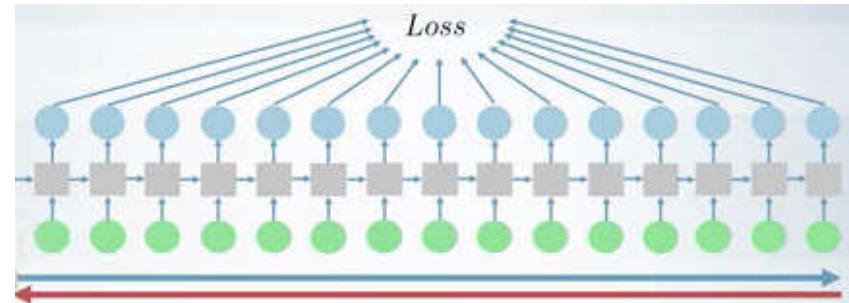
Back Propagation Through Time (BPTT)

Dealing with the **exploding** gradient

– Solution 1: **truncated** BPTT

- Training very long sequences
 - time consuming !
 - exploding gradients !
- Truncated BPTT:
 - run forward and backward passes through the **chunks of the sequence** (instead of the whole sequence)
- Forward
 - We carry hidden states forward in time forever
- Backward
 - only back-propagate in the chunks (small number of steps)
- Much faster but dependencies longer than the chunk size don't affect the training (at least they still work at Forward pass)

we don't learn any long term dependencies



Back Propagation Through Time (BPTT)

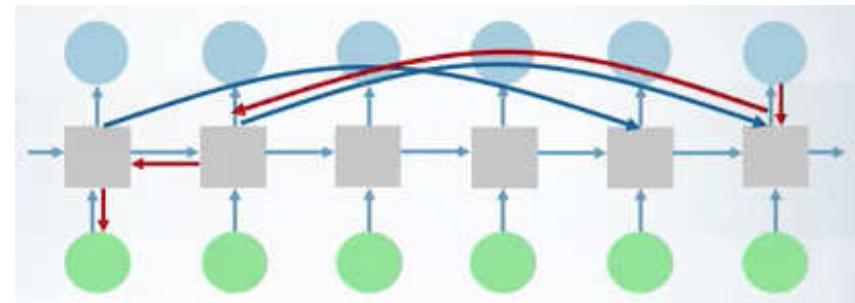
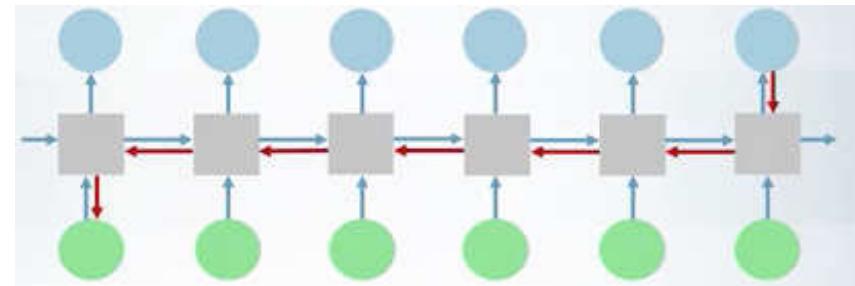
Dealing with the **vanishing** gradient

– Solution 2: use **skip-connections**

- in RNN: because at each step we multiply the Jacobian matrices → vanishing gradient
 - we cannot learn long-term dependencies
- Solution: add **short-cuts** between hidden states that are separated by more than one time step
 - are usual connections (with their own parameter matrices)
 - create much shorter paths between far away time-steps
- Back-propagation through the shortcuts: the gradients vanish slower
 - we can learn long-term dependencies
- not exclusive to RNN (see ResNet)

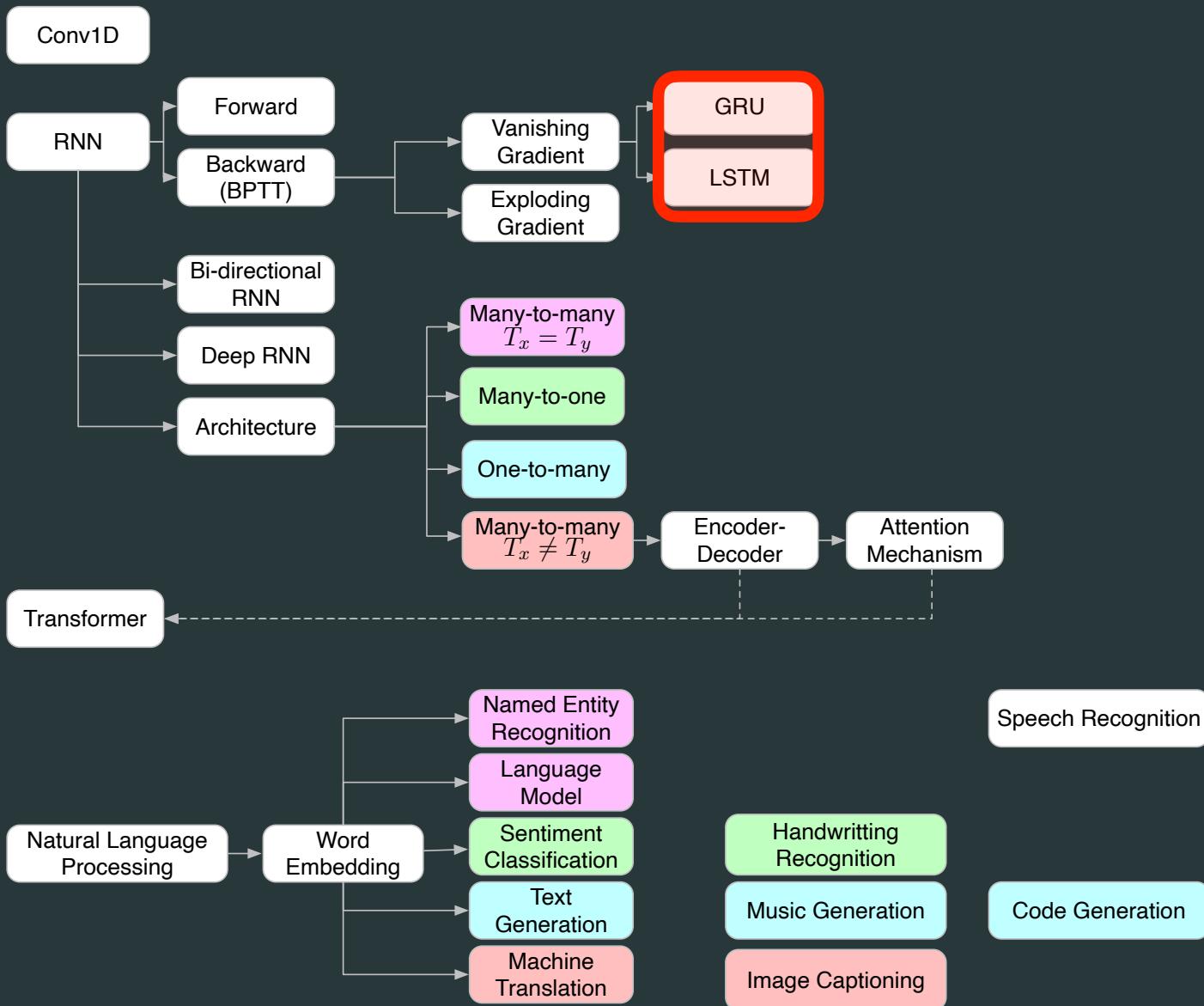
ResNet skips
connections

NN learns by itself where to put connections



the path is shorter

Gated units (LSTM and GRU)

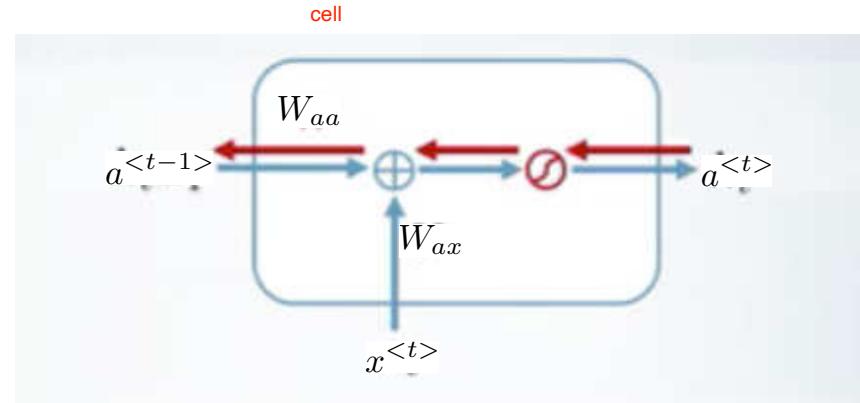


Gated units (LSTM and GRU)

Main ideas : Recurrent Neural Network (RNN)

$$\mathbf{a}^{(t)} = g_a \left(\mathbf{W}_{ax} \mathbf{x}^{(t)} + \mathbf{W}_{aa} \mathbf{a}^{(t-1)} + \mathbf{b}_a \right)$$

- non-linearities and/or multiplication creates the vanishing gradient problem
- Solution ?
 - create a short-way for back-propagation without any non-linearities or multiplication



Gated units (LSTM and GRU)

Main ideas : Long Short Term Memory Units (LSTM)

- Add a new path: the **memory cell** $c^{(t)}$
 - same dimension as the hidden layer $a^{(t)}$
- **Simplified LSTM** (no possibility to erase)
 - Candidate cell value

$$\tilde{c}^{(t)} = g \left(W_{ax}^c x^{(t)} + W_{aa}^c a^{(t-1)} + b^c \right)$$

- Gates (update, output)

$$\Gamma_u = \sigma \left(W_{ax}^u x^{(t)} + W_{aa}^u a^{(t-1)} + b^u \right)$$

$$\Gamma_o = \sigma \left(W_{ax}^o x^{(t)} + W_{aa}^o a^{(t-1)} + b^o \right)$$

- Output memory cell

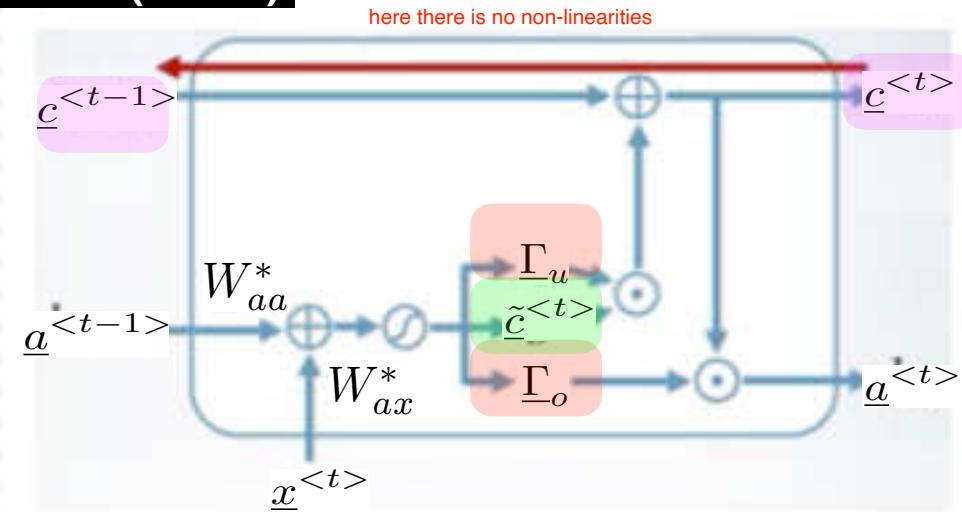
$$c^{(t)} = c^{(t-1)} + \Gamma_u \odot \tilde{c}^{(t)}$$

- Output hidden state

$$a^{(t)} = \Gamma_o \odot f(c^{(t)})$$

- no non-linearity or multiplication in the memory cell-path

- $\frac{\partial c^{(t)}}{\partial c^{(t-1)}} = \text{diag}(1) \Rightarrow$ no vanishing gradients



Gated units (LSTM and GRU)

Main ideas : Long Short Term Memory Units (LSTM)

– Forget Gate LSTM (no possibility to erase)

- Gates (forget)

$$\Gamma_f = \sigma(W_{ax}^f x^{(t)} + W_{aa}^f a^{(t-1)} + b^f)$$

- Output memory cell

$$c^{(t)} = \Gamma_f \odot c^{(t-1)} + \Gamma_u \odot \tilde{c}^{(t)}$$

- Output hidden state

$$a^{(t)} = \Gamma_o \odot f(c^{(t)})$$

– We now have a multiplication on the short-way

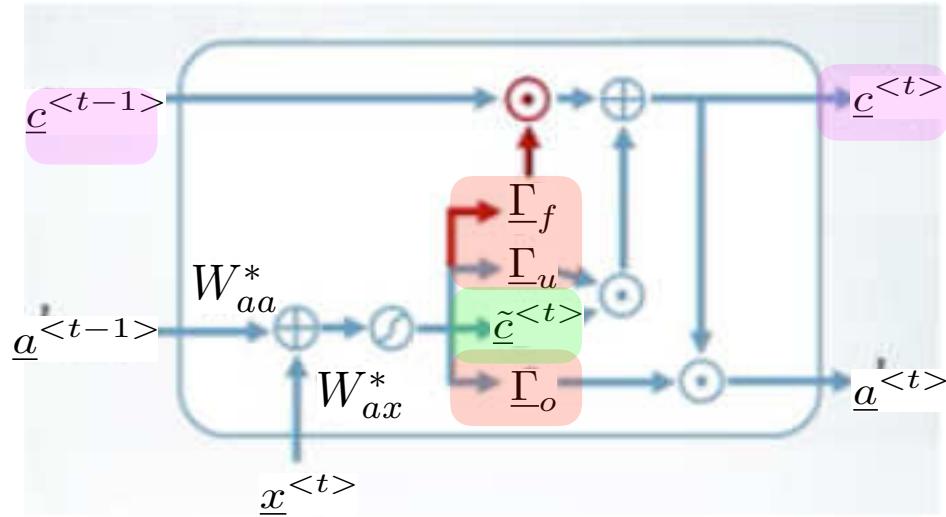
$$\bullet \frac{\partial c^{(t)}}{\partial c^{(t-1)}} = \text{diag}(\Gamma_f)$$

$$\bullet \text{with } \Gamma_f \in [0,1] \text{ the forget gate: } \Gamma_f = \sigma(W_{ax}^f x^{(t)} + W_{aa}^f a^{(t-1)} + b^f)$$

- set a high initial value for $b^f \Rightarrow$ at the beginning of training the gate is open, the LSTM doesn't forget and can learn long-term dependencies in the data

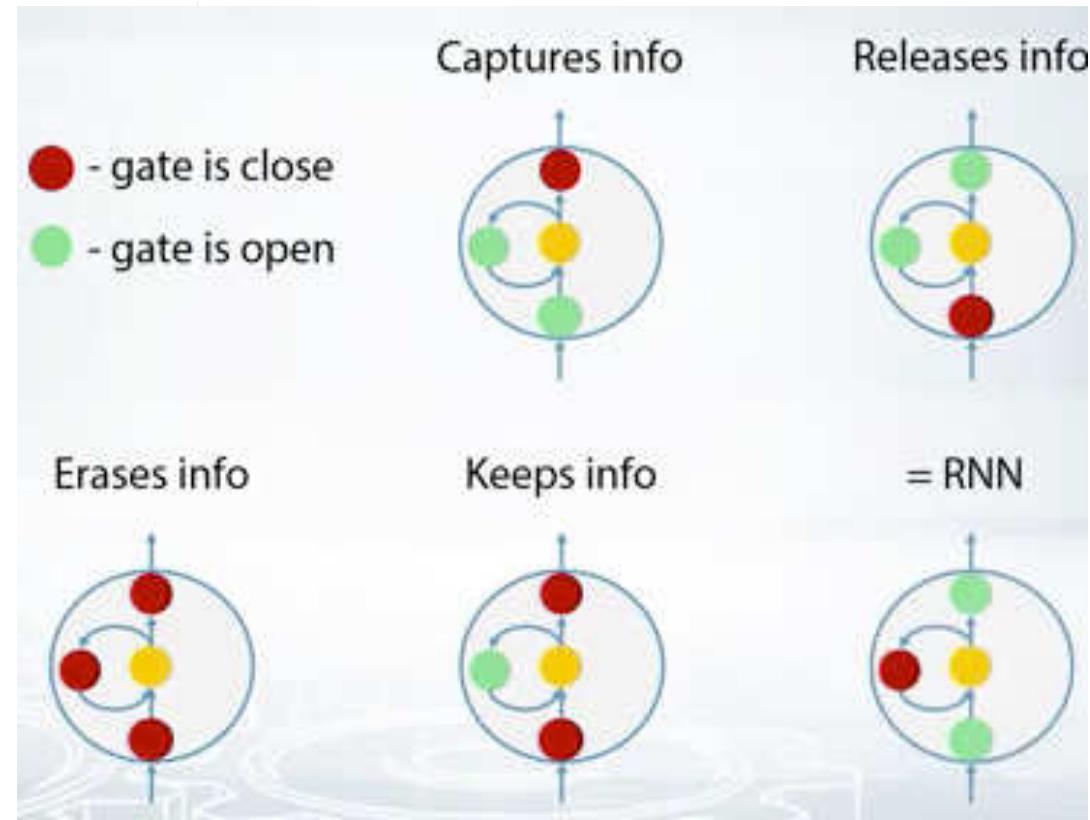
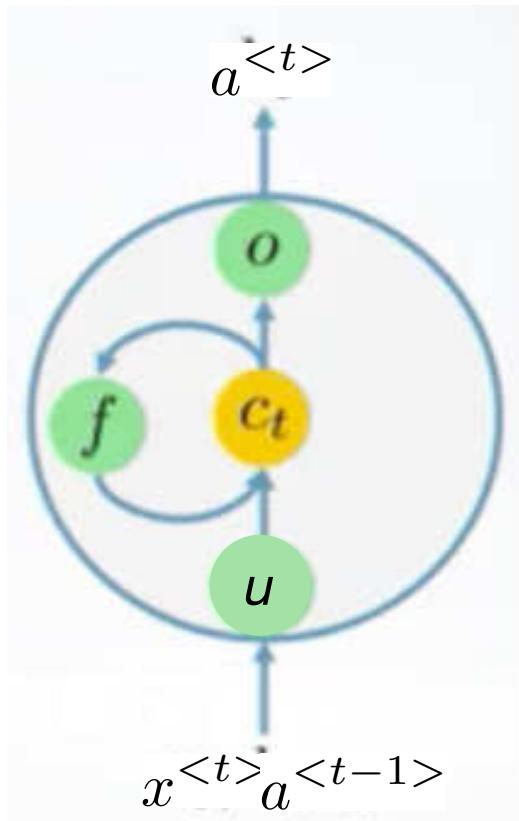
– Note: LSTM has four times more parameters to be trained than RNN

- $W_{ax}^f, W_{aa}^f, W_{ax}^u, W_{aa}^u, W_{ax}^o, W_{aa}^o, W_{ax}^c, W_{aa}^c, b^f, b^u, b^o, b^c$

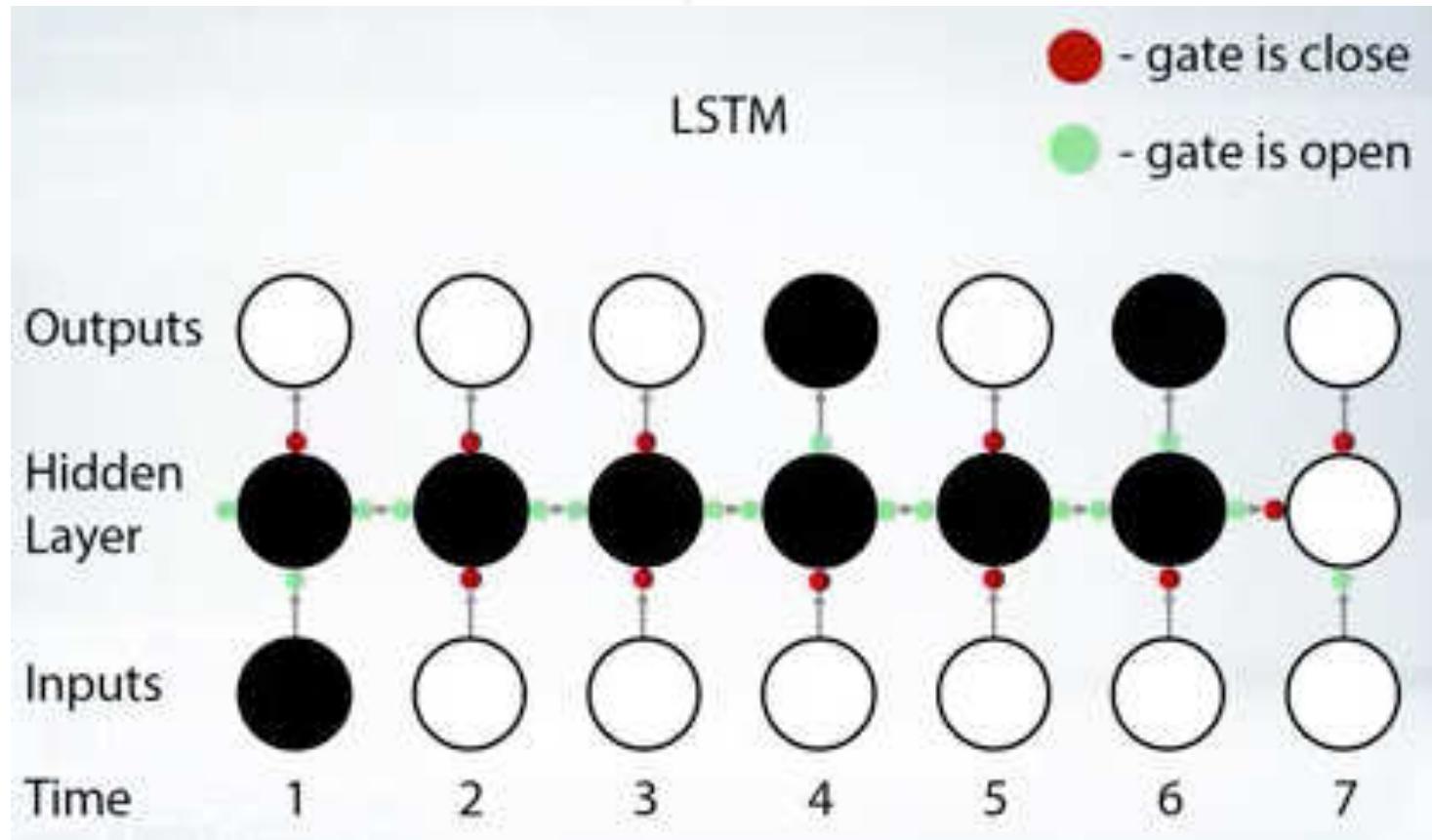


Gated units (LSTM and GRU)

LSTM regimes



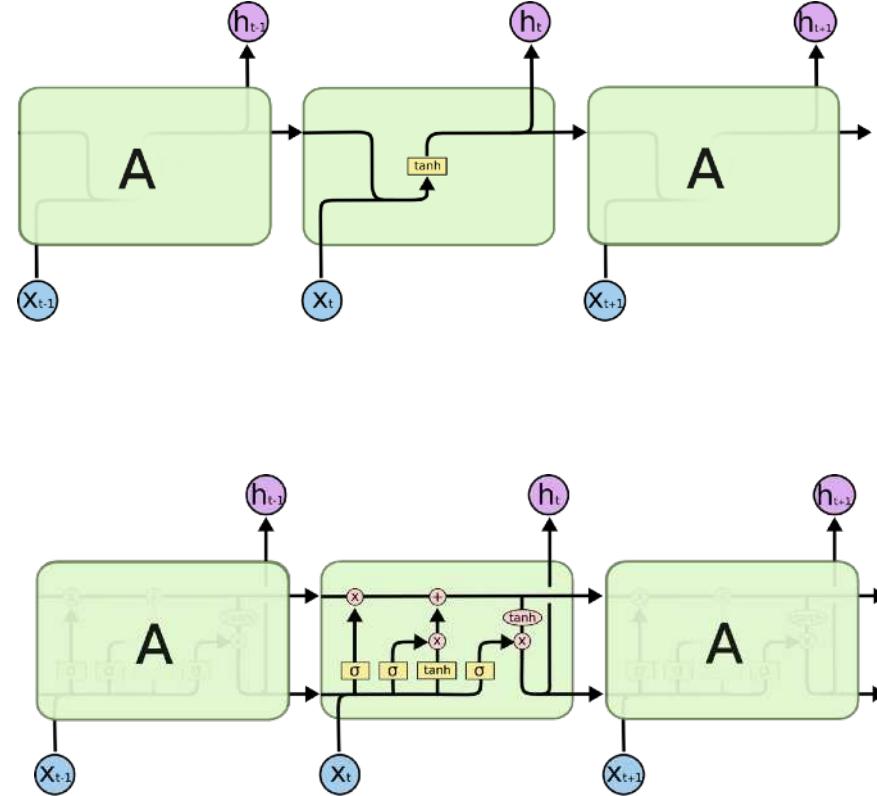
LSTM regimes



Gated units (LSTM and GRU)

LSTM example usage

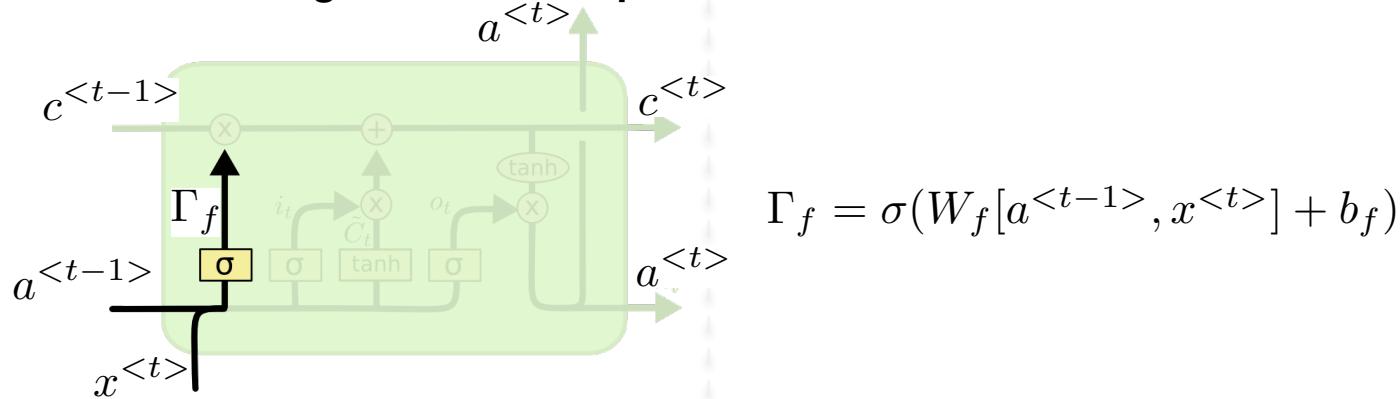
- Instead of having a single neural network layer, there are four, interacting in a very special way.
 - Lines carry a vector
 - A recurrent net transmits its hidden states
 - LSTM introduces a second channel : the cell state
 - It acts as a memory
 - Gates control the memory



Gated units (LSTM and GRU)

LSTM example usage

- What should be forgotten from the previous cell state ?



- Action

- The sigmoid (forget gate) answers for each component :
 - 1 : to keep it,
 - 0 to forget it, or
 - a value in-between to mitigate its influence

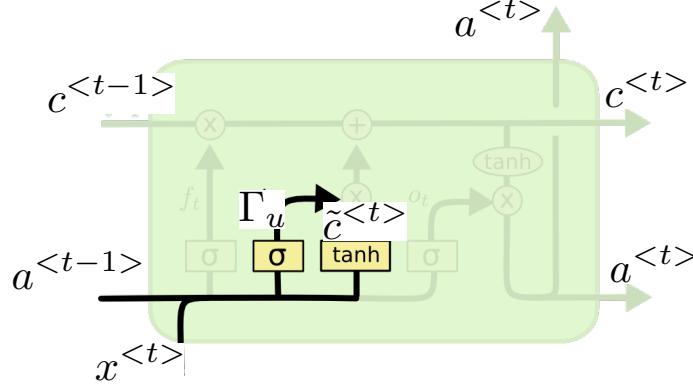
- Intuition for language modelling

- The cell state might embed the gender of the present subject :
 - keep it to predict the correct pronouns,
 - or forget it, when a new subject appears

Gated units (LSTM and GRU)

LSTM example usage

- What should be taken into account ?



$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u)$$

- Action

- Create the update $\tilde{c}^{(t)}$ of the cell state
- and its contribution Γ_u (the update gate with a sigmoid activation)

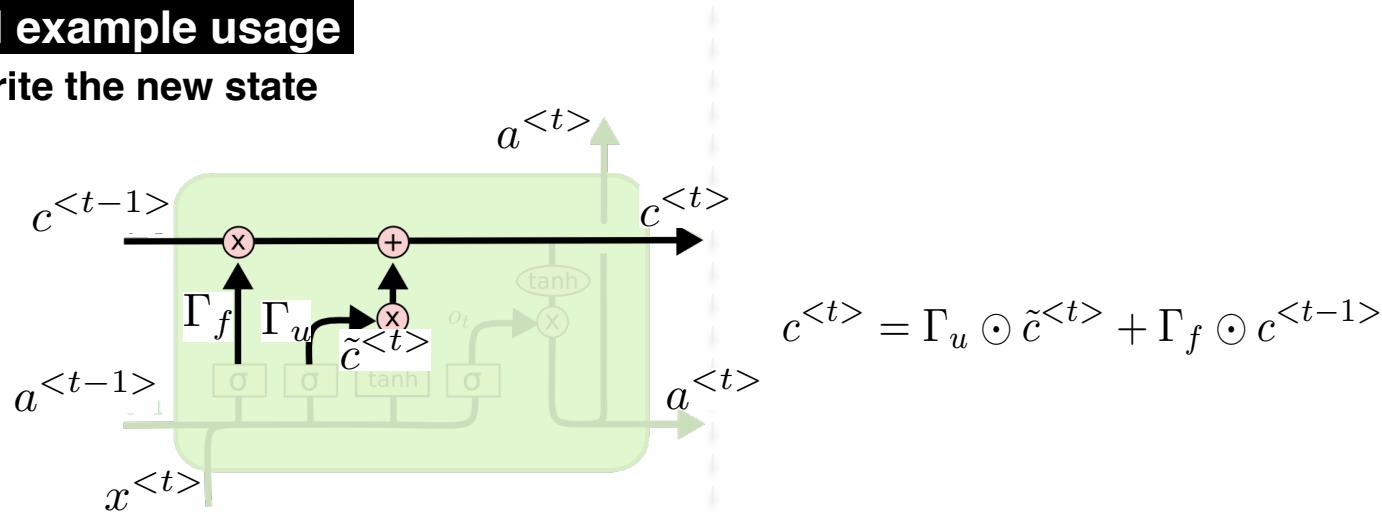
- Intuition for language modelling

- Add the gender of the new subject to the cell state,
- to replace the old one we're forgetting.

Gated units (LSTM and GRU)

LSTM example usage

- Write the new state



- Action

- Merge the old cell state modified by the forget gate with the new input

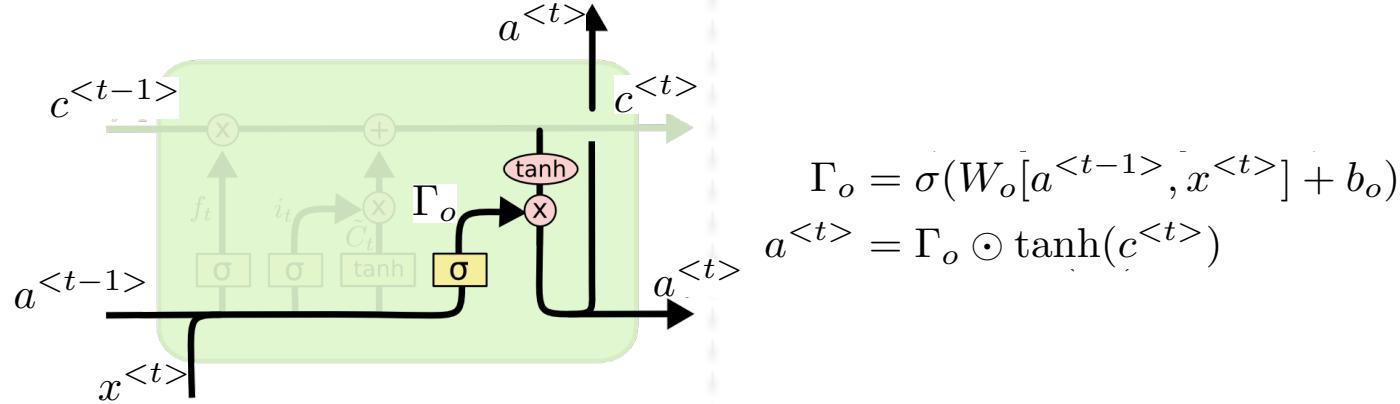
- Intuition for language modelling

- Decide to drop the information about the old subject
- Refresh the memory

Gated units (LSTM and GRU)

LSTM example usage

- Write the new hidden state



- Action

- Decide what parts of the (filtered) cell state to output $a^{<t>}$
- Compute the hidden state

- Intuition for language modelling

- Since we just saw a subject,
- output the relevant information for the future (gender, number)

Gated units (LSTM and GRU)

Gated Recurrent Unit (GRU)

– Simple GRU

- Candidate cell value

$$\tilde{c}^{(t)} = \tanh \left(W^c [c^{(t-1)}, x^{(t)}] + b^c \right)$$

$$c^{(t-1)} = a^{(t-1)}$$

- Gates (update)

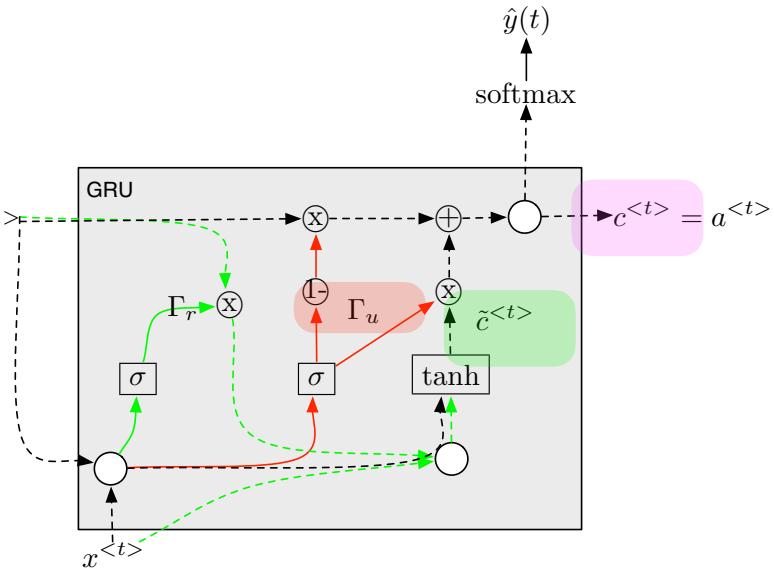
$$\Gamma_u = \sigma \left(W^u [c^{(t-1)}, x^{(t)}] + b^u \right)$$

- Output memory cell

$$c^{(t)} = (1 - \Gamma_u) \odot c^{(t-1)} + \Gamma_u \odot \tilde{c}^{(t)}$$

- Output hidden state

$$a^{(t)} = c^{(t)}$$



Gated units (LSTM and GRU)

Gated Recurrent Unit (GRU)

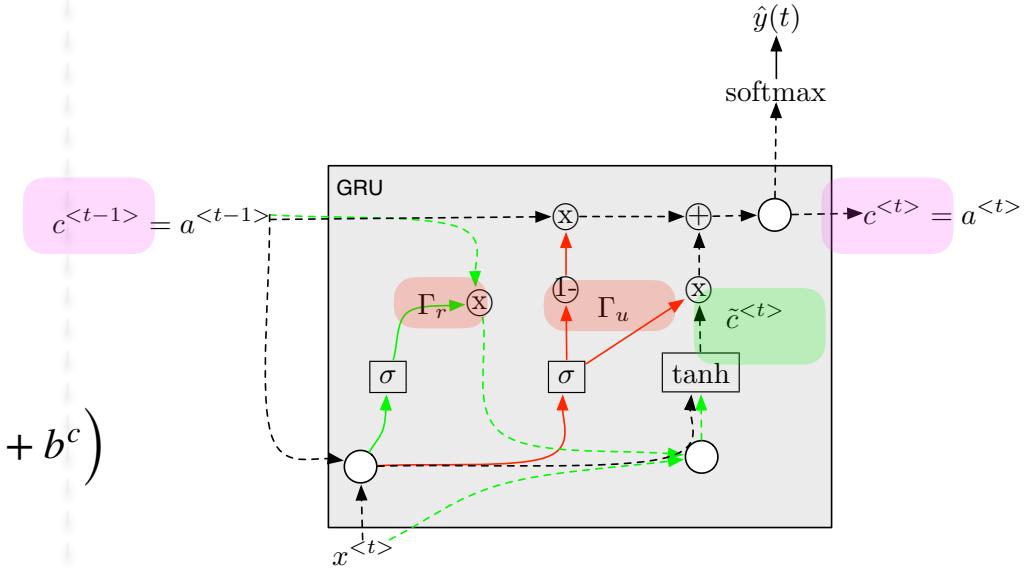
– Full GRU

- add a relevance gate Γ_r
- instead of using $c^{(t-1)}$ to compute $\tilde{c}^{(t)}$ we first filter it using a relevance gate Γ_r

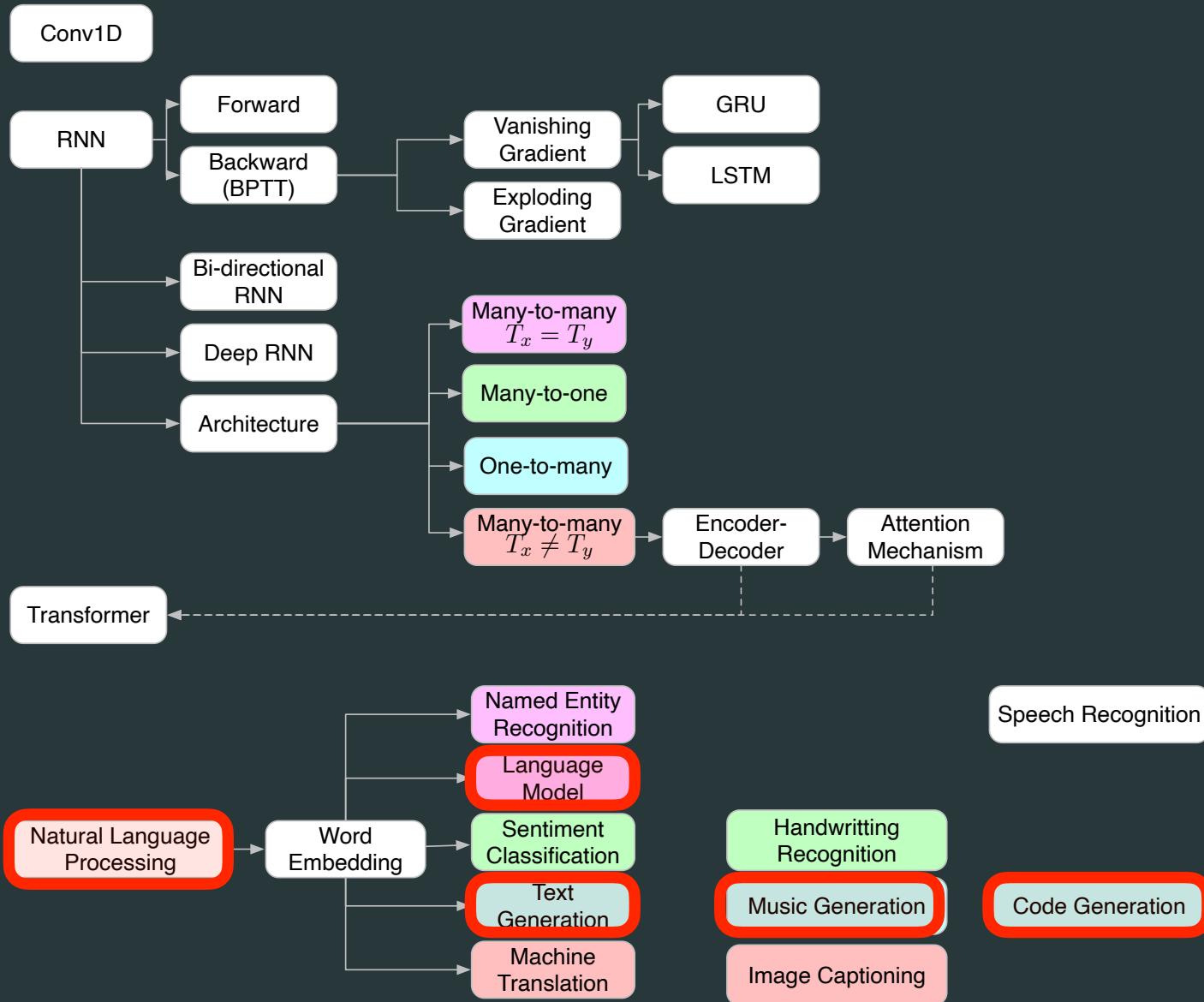
$$\Gamma_r = \sigma \left(W^r [c^{(t-1)}, x^{(t)}] + b^r \right)$$

- Candidate cell value

$$\tilde{c}^{(t)} = \tanh \left(W^c [\Gamma_r \odot c^{(t-1)}, x^{(t)}] + b^c \right)$$



Natural Language Processing



Language model

"you don't really listen to me, you are predicting on what I'm going to say"

– Applications

- Automatic Speech Recognition, Machine Translation, Optical Character Recognition, ...

– Automatic Speech Recognition

- We need a language model to disambiguate what the acoustic model has eared
- Choice between "This is an **example** of an homophone" or "This is an **egg-sample** of an homophone"
 - $p(\text{"example"}) = 5.7 \cdot 10^{-10} \Rightarrow$ is more likely in this context
 - $p(\text{"egg-sample"}) = 3.2 \cdot 10^{-13}$

– Language model: $p(\text{sentence})$

- $p(y^{(1)}, y^{(2)}, \dots, y^{(T_y)})$

Language model

– Notation:

- $\mathbf{w} := w_1^L := w_1, w_2, \dots, w_L$

– Goal:

- Estimate the (non-zero) probability of a word sequence for a given vocabulary

$$\begin{aligned} P(w_1^L) &= P(w_1, w_2, \dots, w_L) \\ &= \prod_{i=1}^L P(w_i | w_1^{i-1}) \quad \forall i, w_i \in V \\ &= P(w_1)P(w_2 | w_1)P(w_3 | w_1, w_2)P(w_4 | w_1, w_2, w_3)\dots P(w_L | w_1\dots w_{L-1}) \end{aligned}$$

- Simplification: the **N-gram assumption** :

$$P(w_1^L) = \prod_{i=1}^L P(w_i | w_{i-N+1}^{i-1}), \quad \forall i, w_i \in V$$

– Bi-gram (N=2) assumption

$$P(w_1^L) = P(w_1)P(w_2 | w_1)P(w_3 | w_2)P(w_4 | w_3)\dots P(w_L | w_{L-1})$$

- using **Recurrent Neural Network (RNN)**

$$P(w_i | w_1^{i-1})$$

Language model

– How to build a language model ?

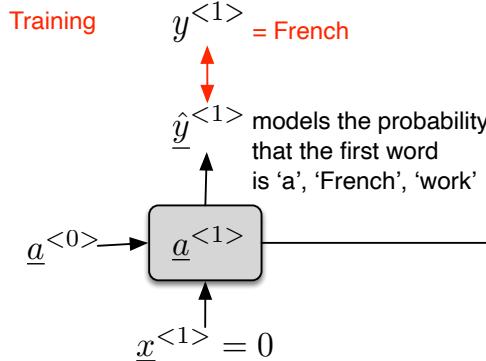
- need a large corpus of English text
- define a dictionary
- tokenise the text
- convert to one-hot-vector
 - + <EOS> (end of sentence)
 - + <UNK> (unknown word, very uncommon vocabulary): Gif-sur-Yvette

Language model: (1) Training using a RNN

– Objective:

- we want to train the parameters θ of the RNN model to maximise the probability of observing the sequence $P(w_1^L) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)\dots P(w_L|w_1\dots w_{L-1})$
- **Example** for $P(w_1^L)$ = "French people work an average of 35 hours a week."
 - Maximize $P(w_1)$

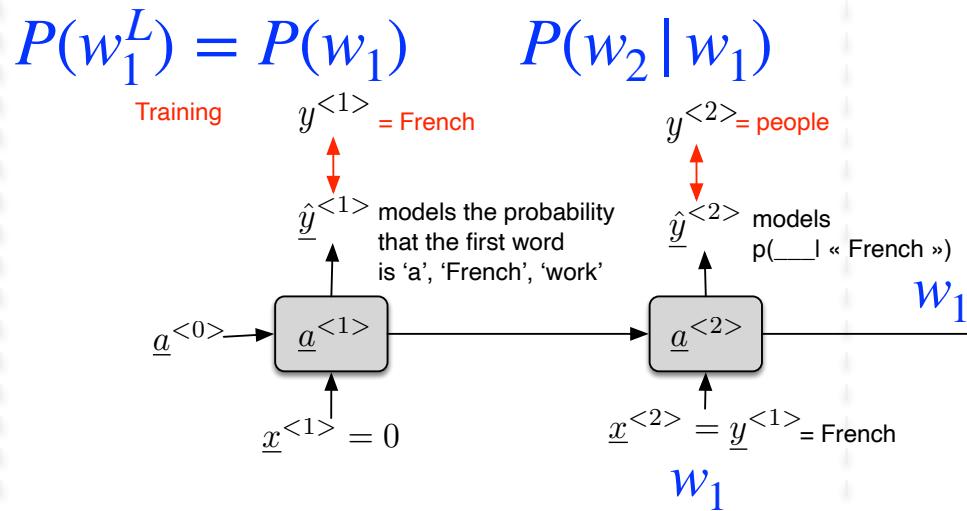
$$P(w_1^L) = P(w_1)$$



Language model: (1) Training using a RNN

– Objective:

- we want to train the parameters θ of the RNN model to maximise the probability of observing the sequence $P(w_1^L) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)\dots P(w_L|w_1\dots w_{L-1})$
- **Example** for $P(w_1^L)$ = "French people work an average of 35 hours a week."
 - Maximize $P(w_1)$
 - Maximize $P(w_2|w_1)$

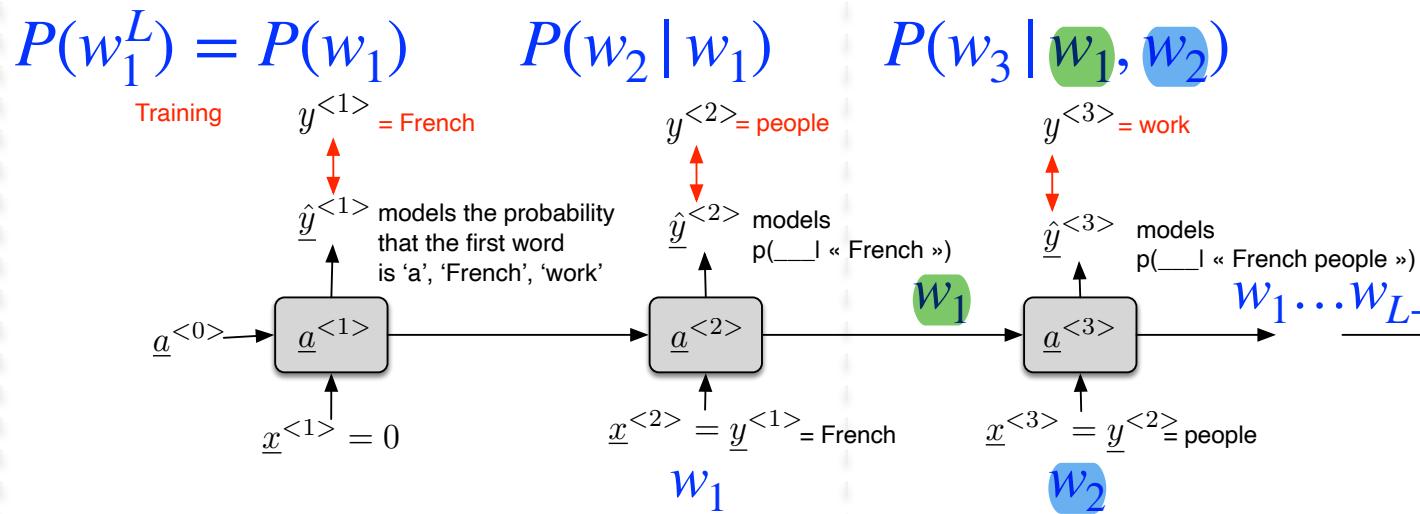


Natural Language Processing

Language model: (1) Training using a RNN

– Objective:

- we want to train the parameters θ of the RNN model to maximise the probability of observing the sequence $P(w_1^L) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)\dots P(w_L|w_1\dots w_{L-1})$
- **Example** for $P(w_1^L)$ = "French people work an average of 35 hours a week."
 - Maximize $P(w_1)$
 - Maximize $P(w_2|w_1)$
 - Maximise $P(w_3|w_1, w_2)$

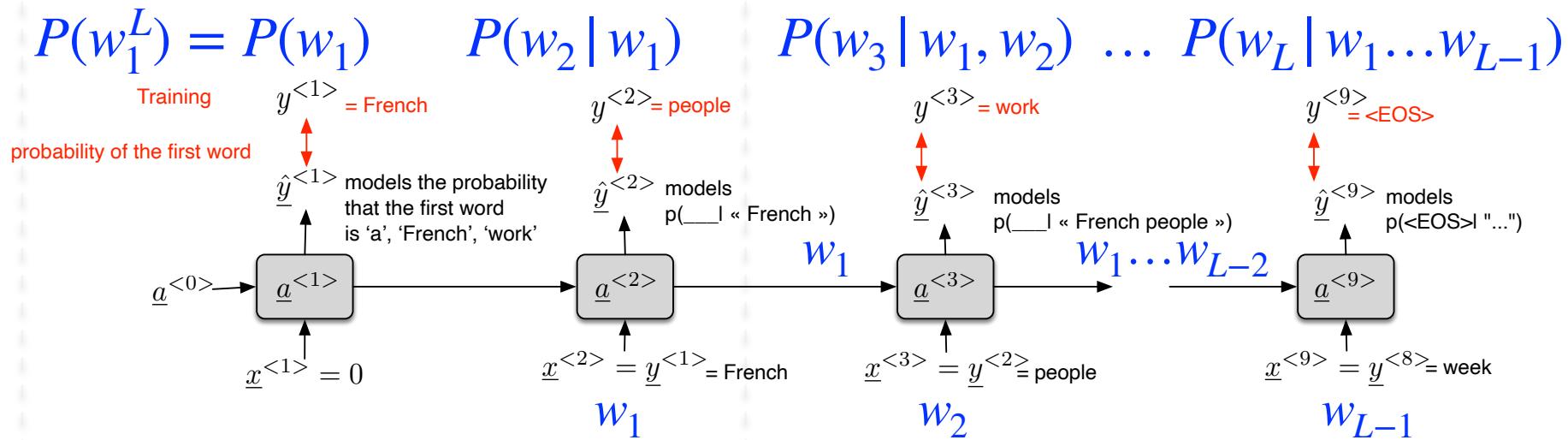


Natural Language Processing

Language model: (1) Training using a RNN

– Objective:

- we want to train the parameters θ of the RNN model to maximise the probability of observing the sequence $P(w_1^L) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)\dots P(w_L|w_1\dots w_{L-1})$
- **Example** for $P(w_1^L)$ = "French people work an average of 35 hours a week."

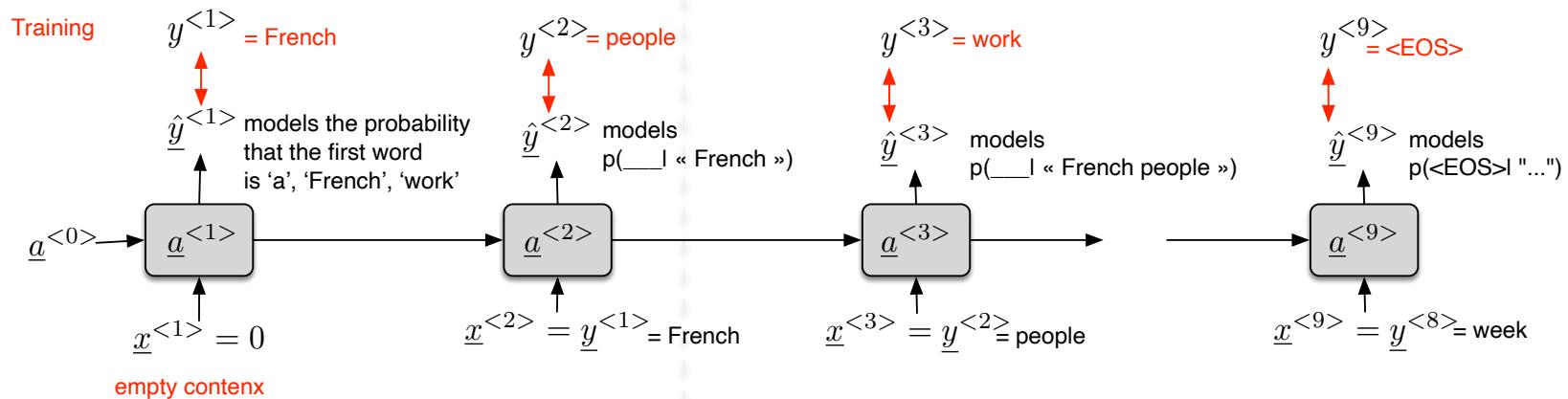


Natural Language Processing

Language model: (1) Training using a RNN

– Algorithm:

- given the previous word as input $x^{(t)}$ and the context $x^{(1,\dots,t-1)}$ (given by the hidden cell $a^{(t)}$) **maximise the probability** to observe the next word $x^{(t+1)}$ (we want the output $\hat{y}^{(t)}$ to be $y^{(t)} = x^{(t+1)}$)
- start with empty context $a^{(0)} = 0$, empty input word $x^{(1)} = 0$



– Training:

- $\hat{y}^{(t)}$: output with a softmax with K neurons, gives the probability of each of the K words in the dictionary
- $y^{(t)}$: ground-truth, one-hot-encoding with K classes

minimise cross entropy

$$\text{Minimise the cross-entropy loss } \mathcal{L} = \sum_t \mathcal{L}(\hat{y}^{(t)}, y^{(t)}) \quad \text{with} \quad \mathcal{L}(\hat{y}^{(t)}, y^{(t)}) = - \sum_{c=1}^K y_c^{(t)} \log(\hat{y}_c^{(t)})$$

Natural Language Processing

Language model: (2) Test using a RNN

- Given a new sentence of three words $\{y^{(1)}, y^{(2)}, y^{(3)}\}$ what is its probability ?

$$p(y^{(1)}, y^{(2)}, y^{(3)}) = p(y^{(1)})$$

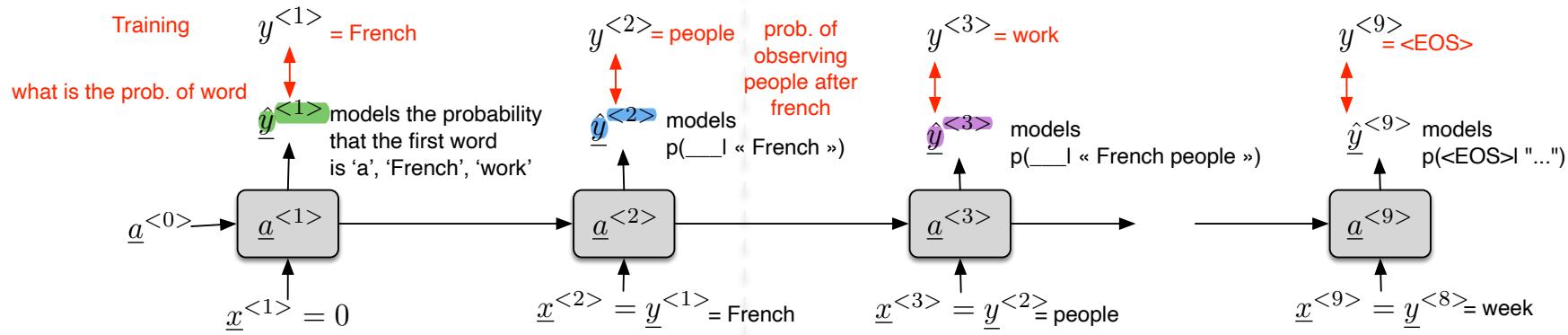
$$p(y^{(2)} | y^{(1)})$$

$$p(y^{(3)} | y^{(1)}, y^{(2)})$$

given by the first softmax

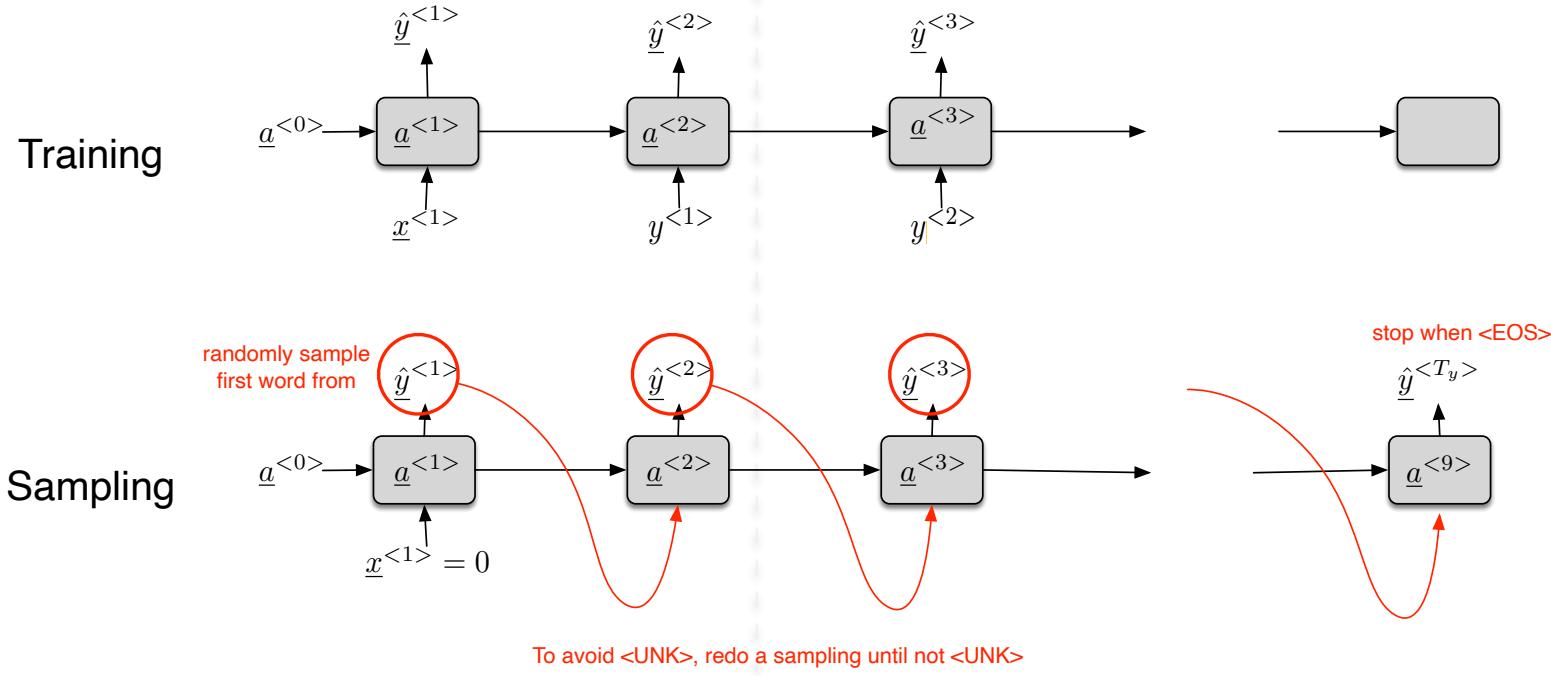
given by the second softmax

given by the third softmax



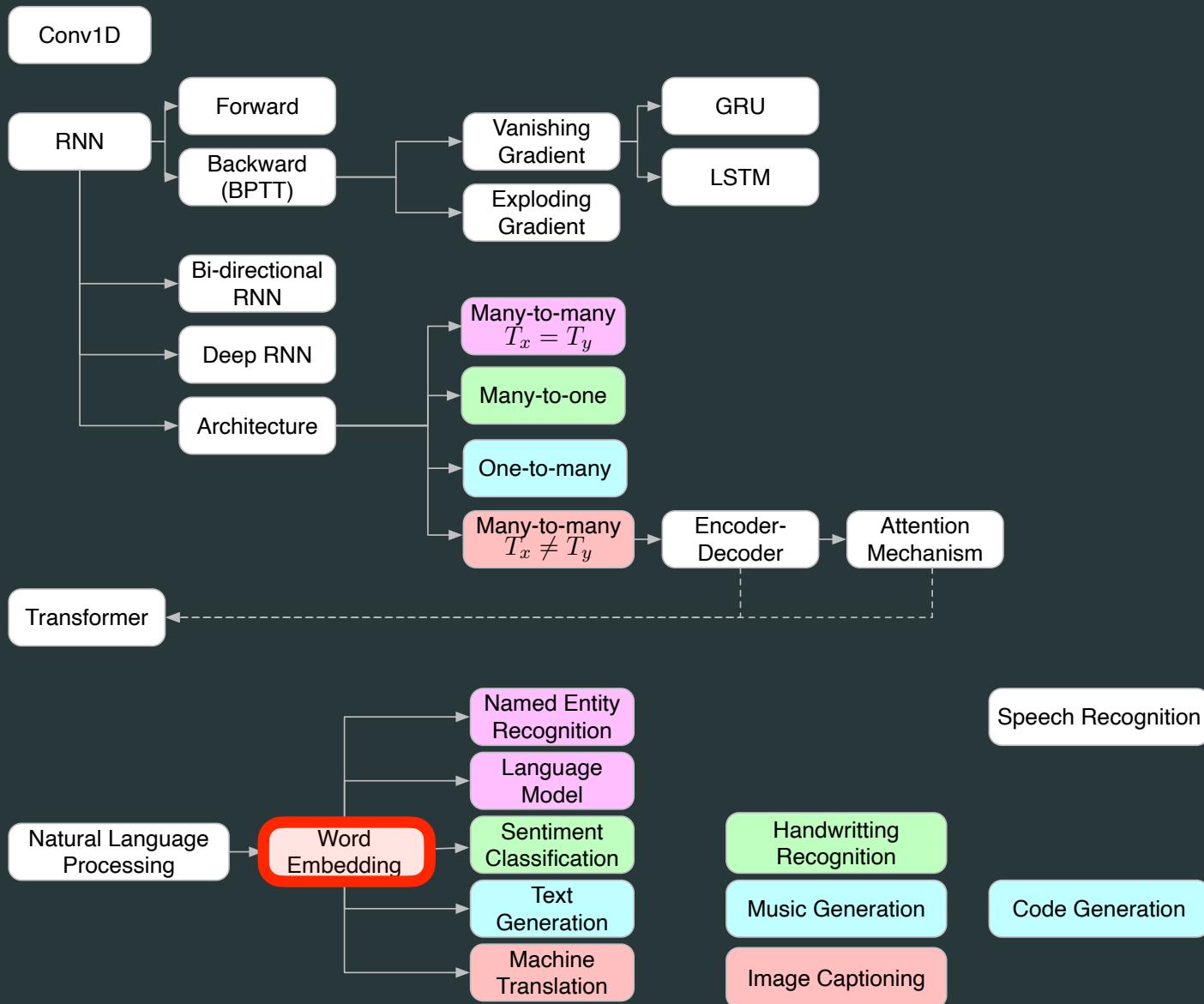
Natural Language Processing

Language model: (3) Sampling novel sequences using RNN



- Word-level RNN
- Character-level RNN:
 - Vocabulary= ['a', 'b', 'c', ..., ' ', ',', '.', '!', 'A', 'B', 'C', ...]
- Examples:
 - Shakespeare, wikipedia, source-code generation
 - <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>

One-hot-encoding



One-hot-encoding

- How to represent the individual words in a sentence ?
 - What will be $x^{(t)}$?
- Construct the list of the N most used categories (words) in the corpus
 - **"vocabulary/dictionary"**
 - N is usually in the range 10.000 words - 50.000.
 - add a <UNK> (unknown word) for the words that are not in vocabulary/dictionary
- Each word is now associated with an ID

1	Angela
...	Boris
...	chancelor
367	Emmanuel
...	France
...	Germany
4075	is
...	Johnson
6830	Macron
...	Merkel
...	of
...	president
...	prime-minister
...	the
10.000	UK

One-hot-encoding

- **One-hot-encoding:**
 - the one-hot vector of an ID is a vector filled with 0s, except for a 1 at the position associated with the ID
 - each word w is associated with a one-hot-vector vector o_{ID}
 - If $N = 10.000$, $x^{(t)}$ has dimension 10.000
- a one-hot encoding makes no assumption about word similarity
 - all words are equally different from each other

{x}	x ^{<1>}	x ^{<2>}	x ^{<3>}	x ^{<4>}	x ^{<5>}	x ^{<6>}	x ^{<7>}
	Emmanuel	Macron	is	the	president	of	France
1	0	0	0				
2	0	0	0				
...				
367	1	0	0				
...				
4075	0	0	1				
...				
6830	0	1	0				
...				
10.000	0	0	0				

One-hot-encoding

– Weaknesses of one-hot-encoding

- it is very high-dimensional
 - vulnerability to overfitting, computationally expensive
- all words are equally different from each other
 - the inner product between any two one-hot vectors is zero
 - as a consequence, we cannot generalise

– Example:

- we have trained a language model to complete the sentence
 - The president is on a **boat** ____? ____ ⇒ "trip"
- since there is no specific relationship between "boat" and "plane", the algorithm cannot complete the sentence
 - The president is on a **plane** ____? ____ ⇒ ?

Natural Language Processing

Word embedding

represent the words in the semantic space

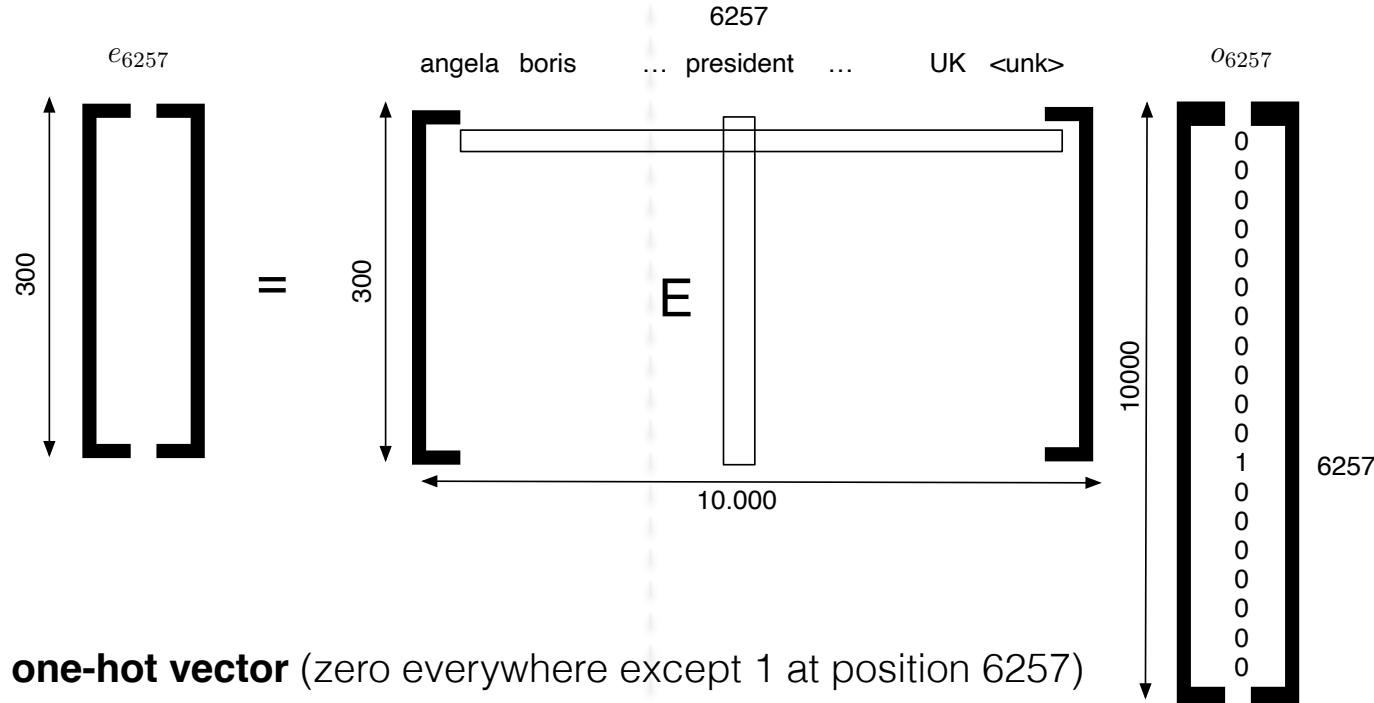
- learn a continuous representation of words
- each word w is associated with a real-valued vector e_{ID}
- if embedding size is 300, e_{5391} is a 300 dimensional vector
- we would like the distance $\|e_{ID1} - e_{ID2}\|$ to reflect the meaningful similarities between words

	Man 5391	Woman 9853	King 4914	Queen 7157	Uncle 456	Aunt 6257	President 7124	Chancellor 3212	France 6789	Germany 1234	Boat 5923	Plane 8871
Gender	-1	1	-1	1	-1	1	-0,7	-0,3	0	0	0	0
Royal	0	0	1	1	0	0	0,5	0,5	0	0	0	0
Age	0	0	0,7	0,7	0,5	0,5	0,7	0,7	0	0	0	0
Country	0	0	0,5	0,5	0	0	0,5	0,2	1	1	0	0
Transportation	0	0	0	0	0	0	0	0	0	0	1	1
...

- In this representation "boat" and "plane" are quite similar
 - some of the features may differ but most features are similar
- Therefore the algorithm can find (by generalisation) that
 - The president is on a **boat** ____?____ \Rightarrow "trip"
 - The president is on a **plane** ____?____ \Rightarrow "trip"

Natural Language Processing

Word embedding/ Matrix



- \mathbf{o}_{6257} = **one-hot vector** (zero everywhere except 1 at position 6257)
 - dimension 10,000
- $\mathbf{e}_{6257} = \frac{\mathbf{E}}{(300, 10,000)} \cdot \frac{\mathbf{o}_{6257}}{(10,000, 1)}$ = **embedding** of \mathbf{o}_{6257}
- $\mathbf{e}_j = \mathbf{E} \cdot \mathbf{o}_j$ = embedding for word j
- In practice, use **look-up table** (take the column vector of \mathbf{E} corresponding to 6257)

Word embedding/ Learning

- Various existing methods to learn word embedding:
 - Classic neural language model [Bengio et al., 2003]
 - Collobert and Weston model [Collobert, Weston, 2008]
 - Word2Vec [Mikolov et al. 2013]
 - Continuous bag-of-words (CBOW)
 - Skip-gram
 - Glove model [Pennington et al. 2014]

[Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.]
[R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.]

[T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.]

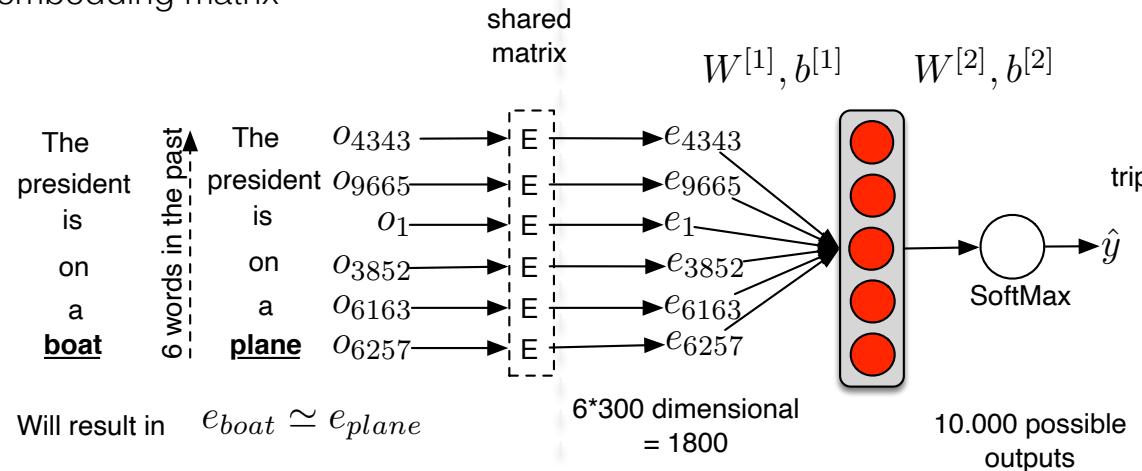
[T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.]

[J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.]

Natural Language Processing

Word embedding/ Classic neural language model

- **Method:** build a language model (learn to predict the following word) using a MLP
 - Use back-propagation to learn the parameters: \mathbf{E} , $\mathbf{W}^{[1]}$, $\mathbf{b}^{[1]}$, $\mathbf{W}^{[2]}$, $\mathbf{b}^{[2]}$
 - \mathbf{E} is the embedding matrix



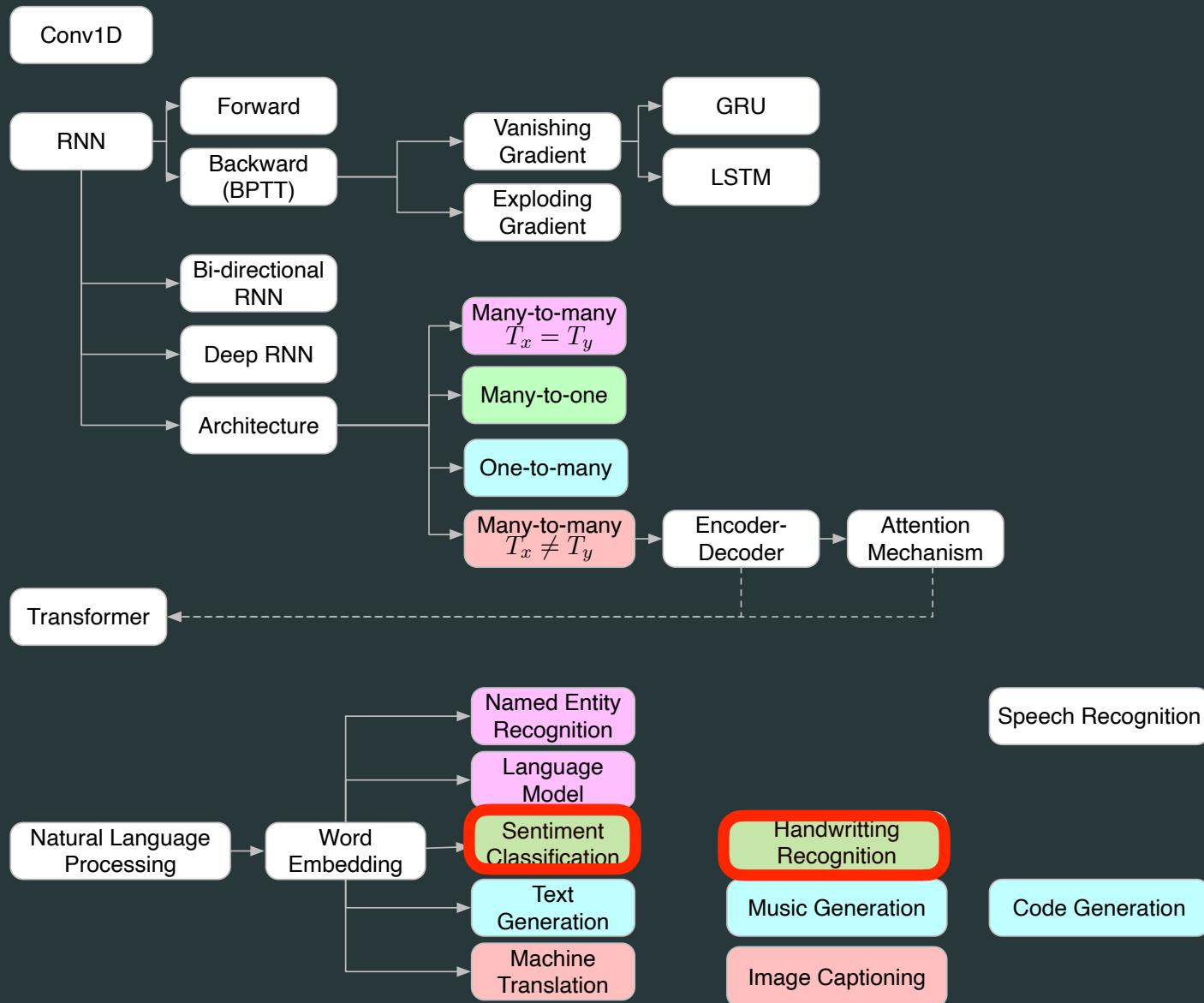
– Example:

- "The president is on a boat trip to the United States."
 target

– Other possible definition of the context

- last 4 words
- 4 words on the left, 4 words on the right: "is on a boat ____?____ to the United States"
- last 1 words: "boat ____?____"
- nearby 1 word (skip-gram): "president ____?____"

Application: Sentiment Classification



Application: Sentiment Classification

x

y

This movie is fantastic ! I really like it because it is so good !



Not to my taste, will skip and watch another movie.

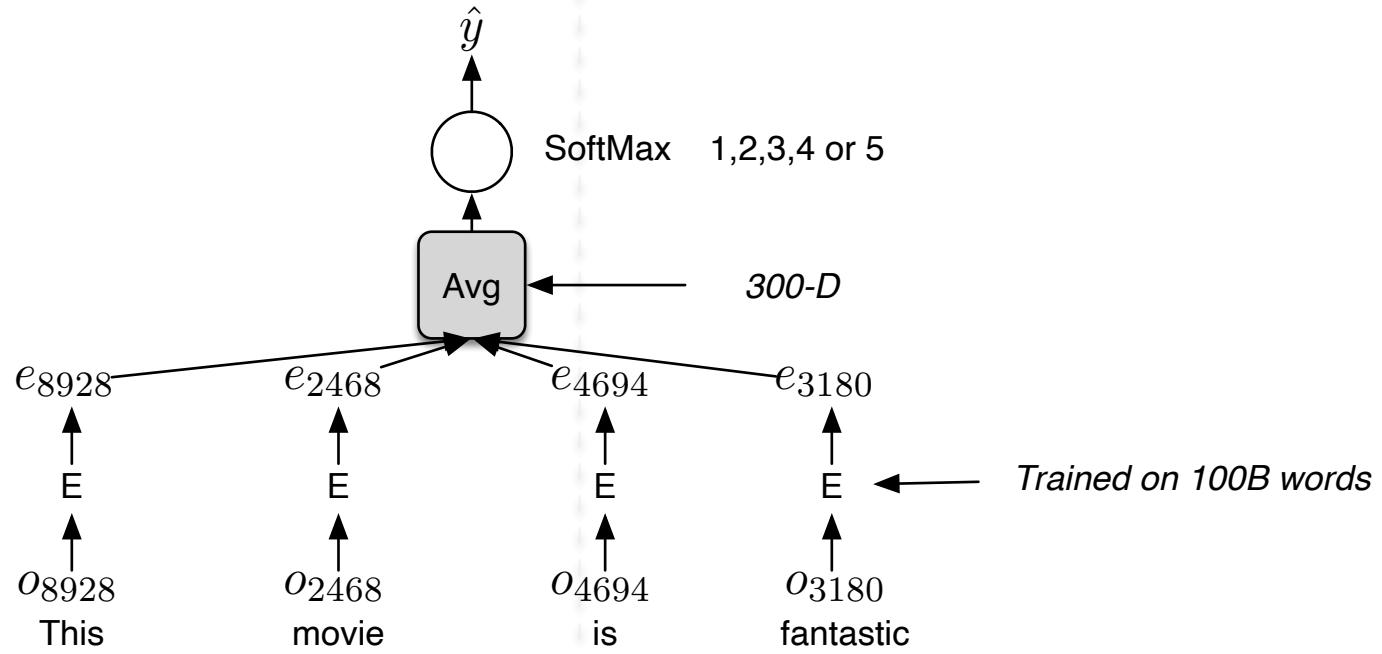


This movie was completely lacking a good script,
good actors and a good director.



Natural Language Processing

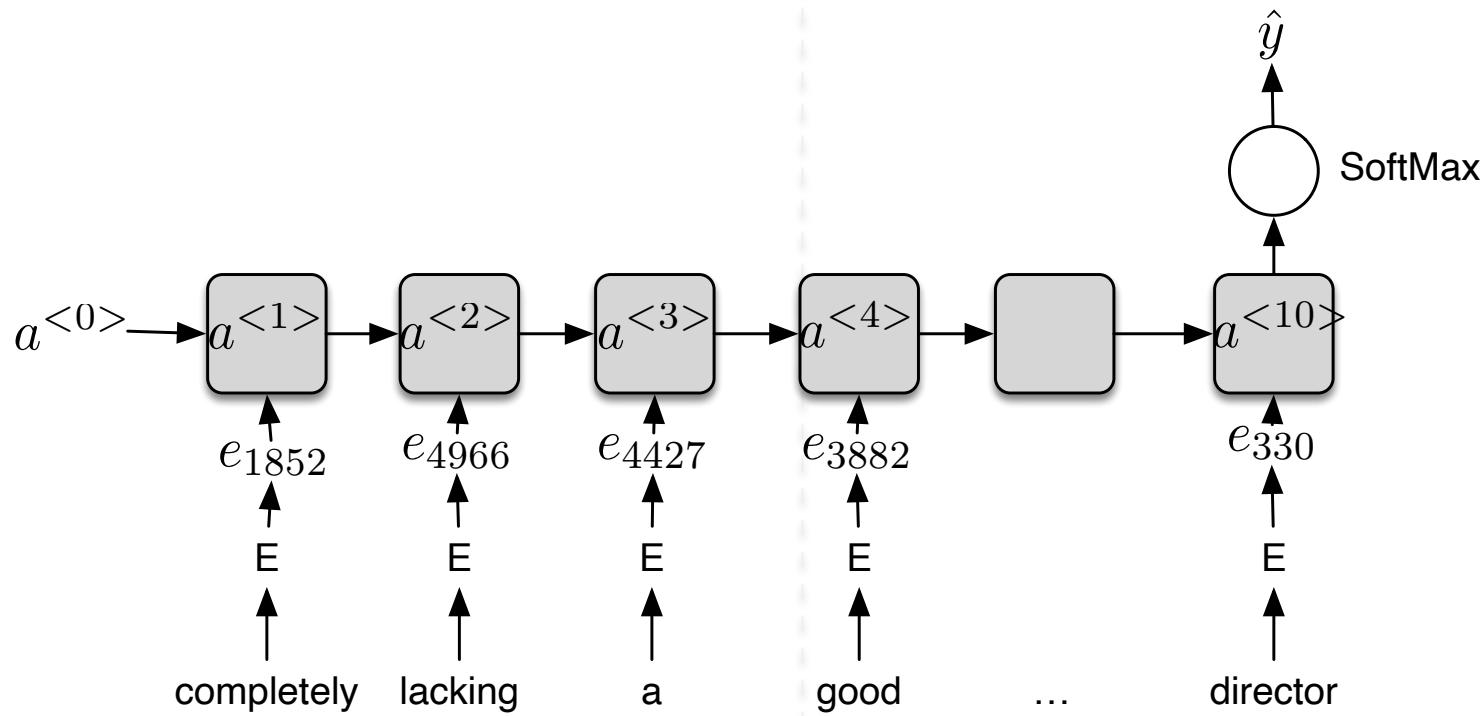
Application: Sentiment Classification / using a simple model



- Does not work for
"This movie was completely lacking a **good** script, **good** actors and a **good** director"
 - This is because the Avg of "good" is positive
 - The model can not understand that we means the opposite ("lacking")
 - We need a Sequence model \Rightarrow RNN

Application: Sentiment Classification / using a RNN

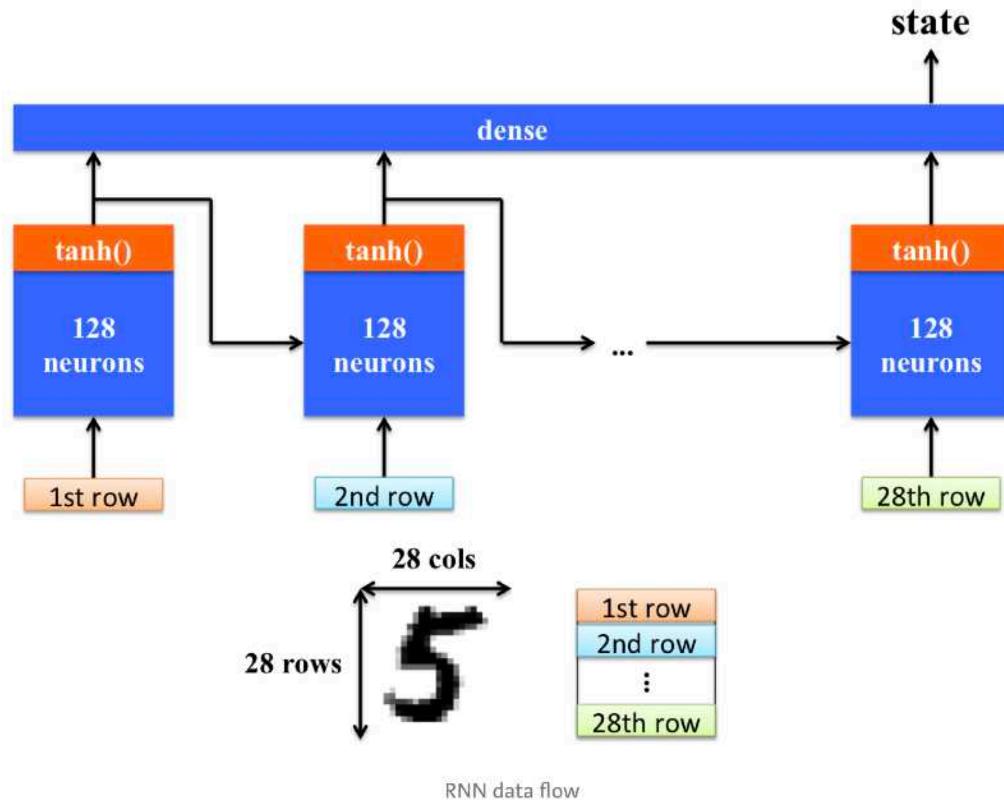
- **Many-to-one** architecture



Natural Language Processing

Application: Handwritten Character Recognition

- **Many-to-one** architecture

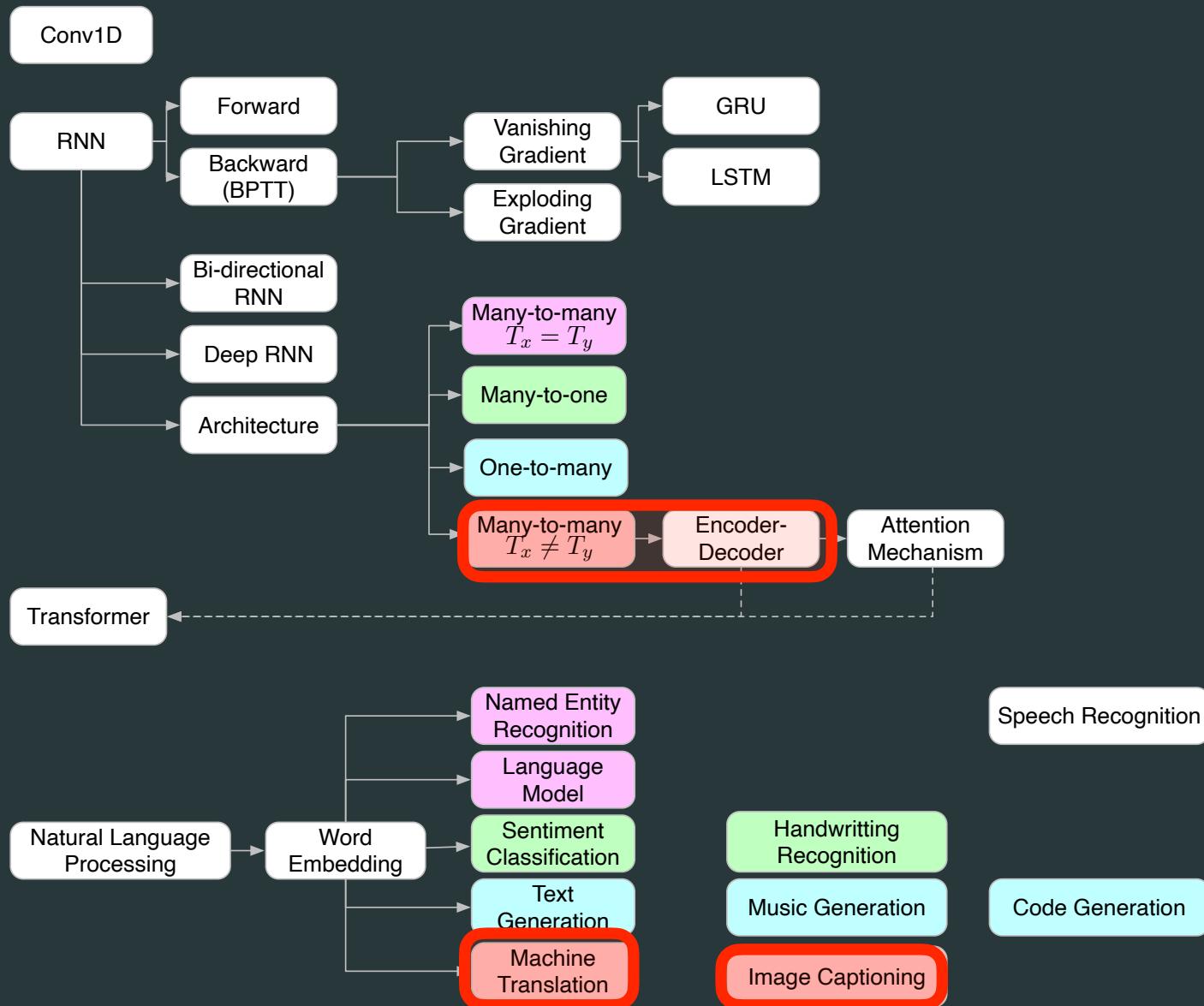


source <https://medium.com/machine-learning-algorithms/mnist-using-recurrent-neural-network-2d070a5915a2>

[Graves et al. 2008 "Unconstrained online handwriting recognition with recurrent neural networks"] [LINK](#)

[Graves et al. 2009 "Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks"] [LINK](#)

Sequence to Sequence



Applications

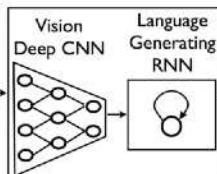
Automatic Speech Recognition



Automatic picture captioning



A group shopping at an outdoor market.
There are many vegetables at the fruit stand.



Self Driving Cars



DEEP LEARNING HAS MASTERED GO Google Alpha Go

nature International weekly journal of science
Home News & Comment Research Careers & Jobs Current Issue Archives About & Write Profile
AlphaGo > Home Site > News Now > Now > Article
ARTICLE | NOW

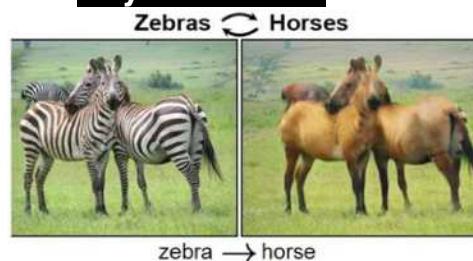
Google reveals secret test of AI bot to beat top Go players
Updated version of DeepMind's AlphaGo program beaten legendary online competitor.

David Silver, Jasper Snoek-Johnson, Oriol Vinyals, George van den Driessche, Julian Schrittwieser, Nal Kalchbrenner, Marc Lanctot, Karen Simonyan, Demis Hassabis, John Mnih, Noa Kischka, Ilya Sutskever, Timothy Lillicrap, Mandarake Leach, Koen Kavvounoglou, Thore Graepel & Demis Hassabis

Nature 529, 484–489 (28 January 2016) | doi:10.1038/nature18861
Received: 11 November 2015 | Accepted: 05 January 2016 | Published online: 27 January 2016



Style Transfer



zebra → horse



horse → zebra



Neural Machine Translation

DeepL Traducteur DeepL Pro Connexion
Anglais (langue détectée) ▾ Fr... ▾ forme/informel ▾ Glossaire

Deep learning (also known as deep structured learning) is part of a broader family of machine learning methods based on artificial neural networks with representation learning. Learning can be supervised, semi-supervised or unsupervised.

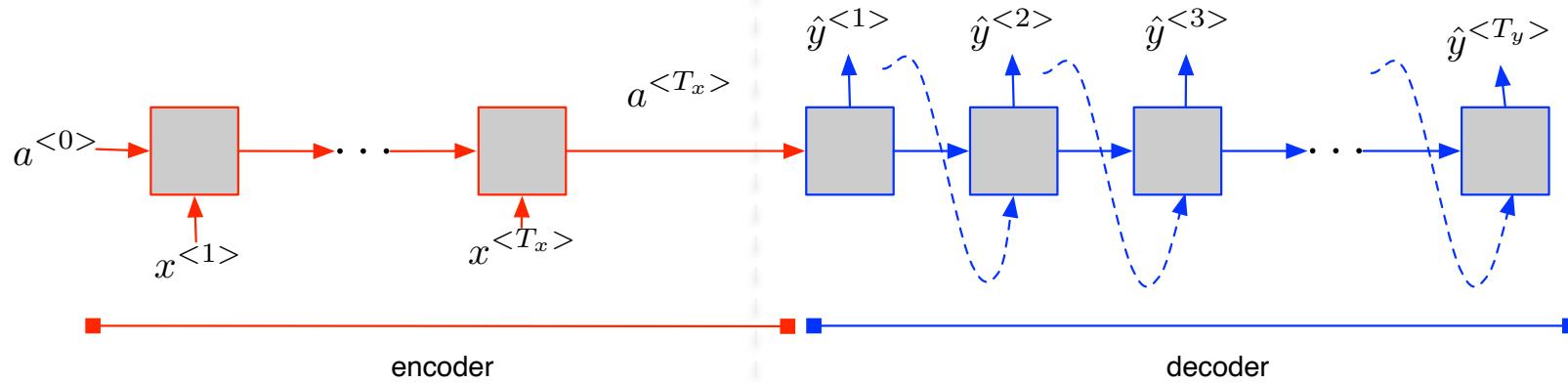
L'apprentissage profond (également appelé apprentissage structuré profond) fait partie d'une famille plus large de méthodes d'apprentissage automatique basées sur les réseaux neuronaux artificiels avec apprentissage par représentation. L'apprentissage peut être supervisé, semi-supervisé ou non supervisé.

Sequence to Sequence

- What happens if $T_x \neq T_y$ (many-to-many architecture but with different lengths)
 - Example: Machine Translation, Automatic Speech Recognition

{x}	x ^{<1>}	x ^{<2>}	...			x ^{<7>}		
	Deep	Learning	is	part	of	machine	learning	
{y}	y ^{<1>}	y ^{<2>}	...					y ^{<9>}
	L'	apprentissage	profond	fait	partie	de	l'	apprentissage machine

- Solution:
 - **Sequence to sequence** [Sutskever et al., 2014] or **Encoder-Decoder** [Cho et al., 2014] architecture



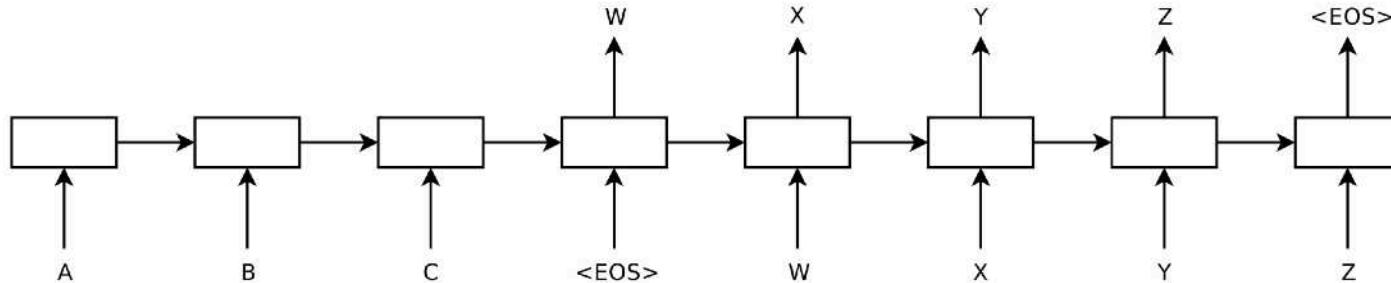
[Sutskever et al. 2014, "Sequence to sequence learning with neural networks"] [LINK](#)

[Cho et al. 2014 "Learning phrase representations using RNN encoder-decoder for statistical machine translation"] [LINK](#)

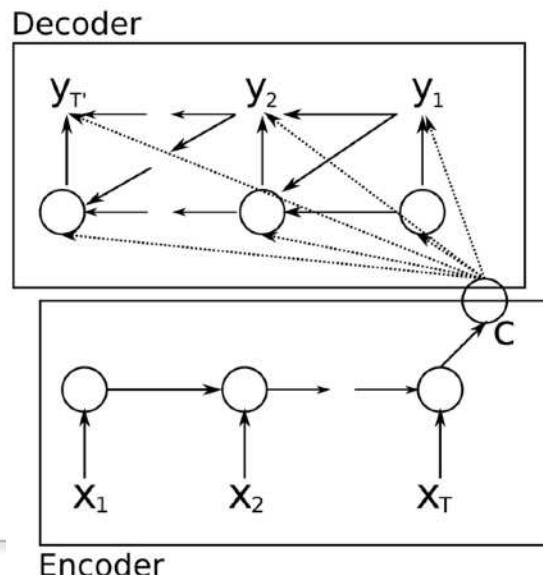
Sequence to Sequence

Introduction

[Sutskever et al. 2014, "Sequence to sequence learning with neural networks"] [LINK](#)



[Cho et al. 2014 "Learning phrase representations using RNN encoder-decoder for statistical machine translation"] [LINK](#)

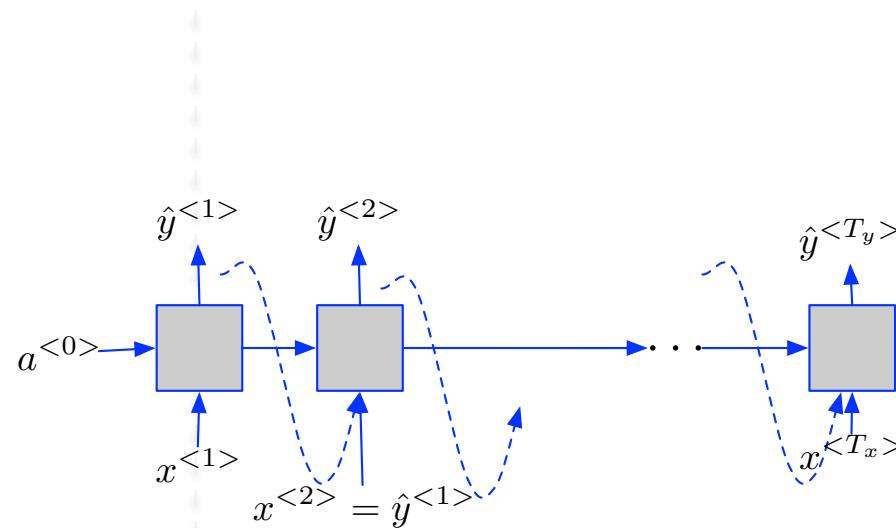


Sequence to Sequence

Machine Translation

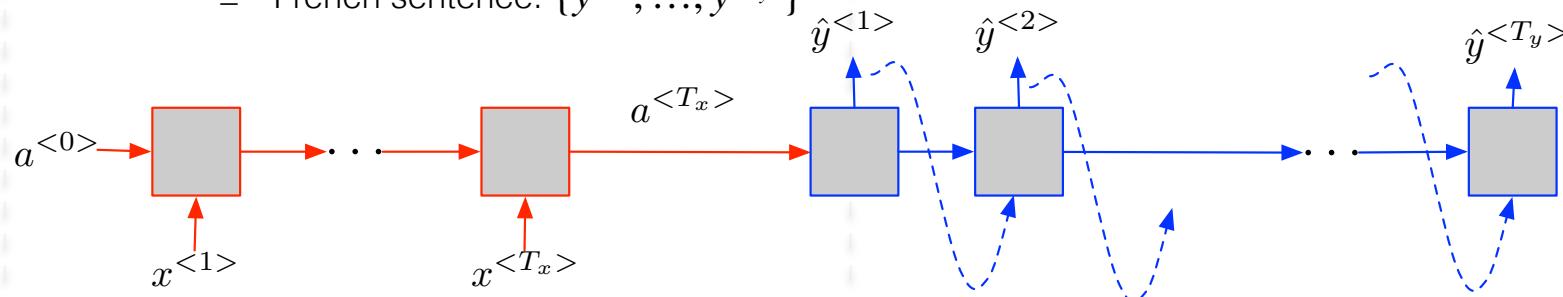
– Language model

- $p(y^{(1)}, y^{(2)}, \dots, y^{(T_y)})$

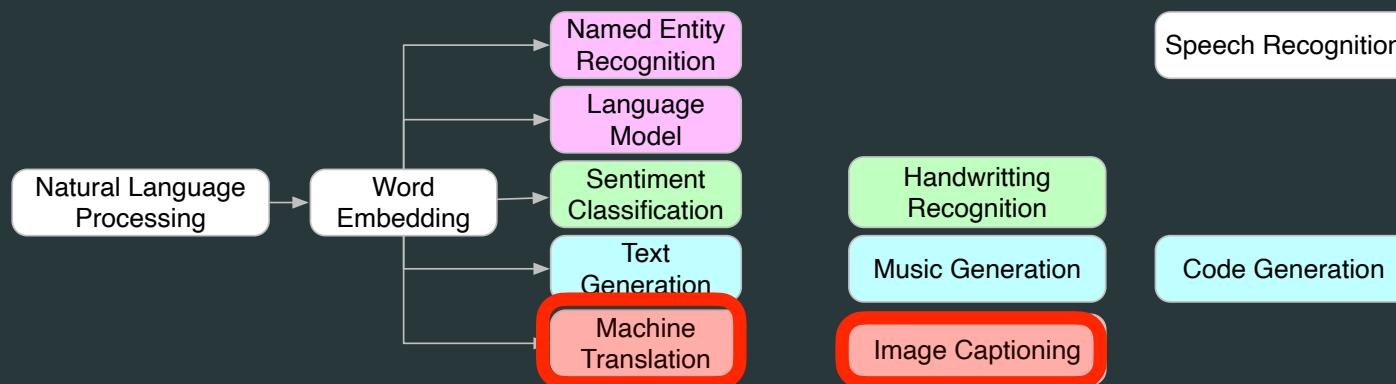
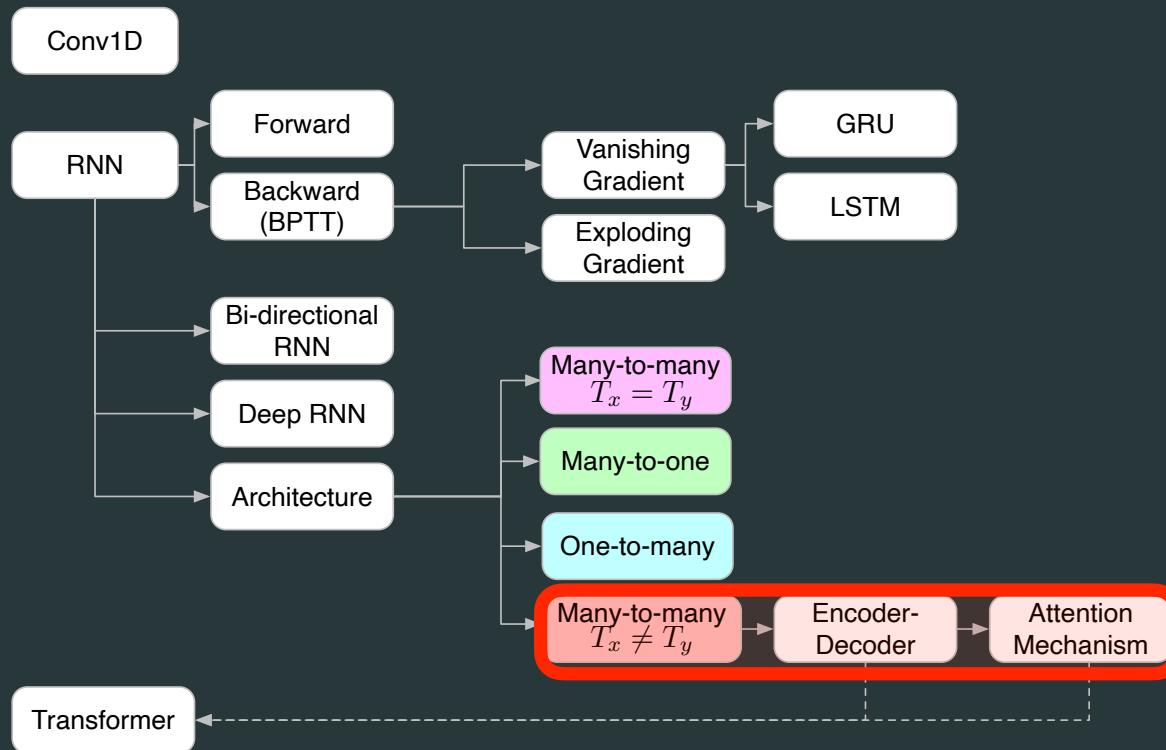


– Machine Translation:

- Conditional language model: $p(y^{(1)}, \dots, y^{(T_y)} | x^{(1)}, \dots, x^{(T_x)})$
 - English sentence: $\{x^{(1)}, \dots, x^{(T_x)}\}$
 - French sentence: $\{y^{(1)}, \dots, y^{(T_y)}\}$



Attention model



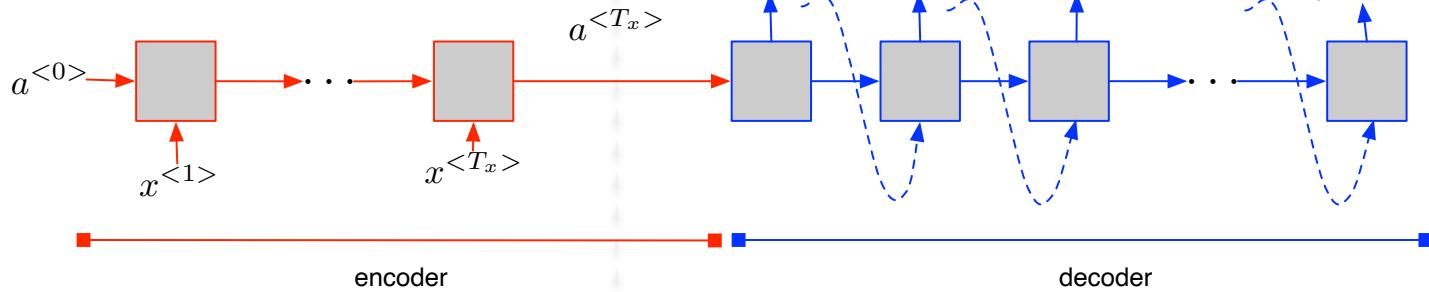
Sequence to sequence

Attention model

– The problem of long sequences

This kind of experience is part of Disney's efforts to "extend the lifetime of its series and build new relationships with audiences via digital platforms that are becoming ever more important," he added.

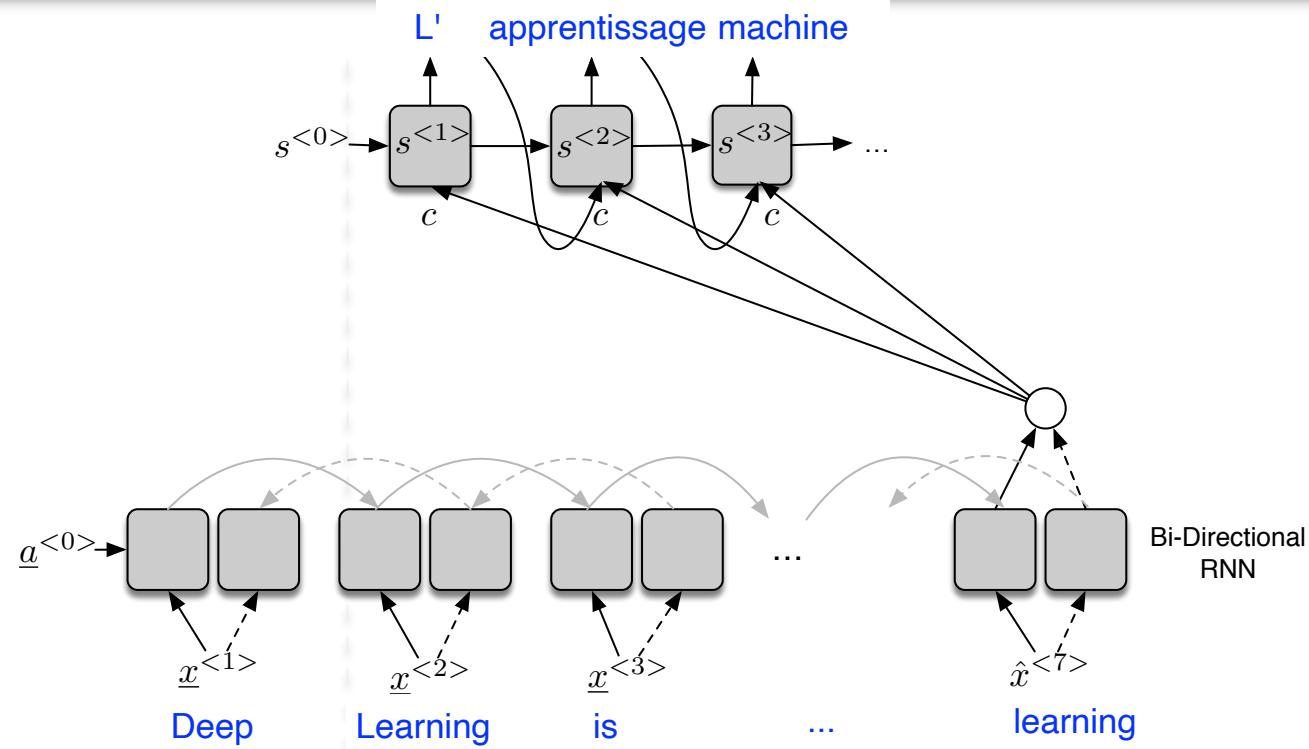
Ce type d'expérience fait partie des initiatives du Disney pour "prolonger la durée de vie de ses nouvelles et de développer des liens avec les lecteurs numériques qui deviennent plus complexes.



- Encoder/Decoder:
 - $a^{<T_x>}$ is supposed to memorise the whole sentence then translate it;
 - human way: translate each part of a sentence at a time
- **Attention model?** (modification of the Encoder/Decoder)
 - $a^{<T_x>}$ is replaced by a local version in the encoder \Rightarrow we pay attention on specific encoding

Sequence to sequence

Attention model

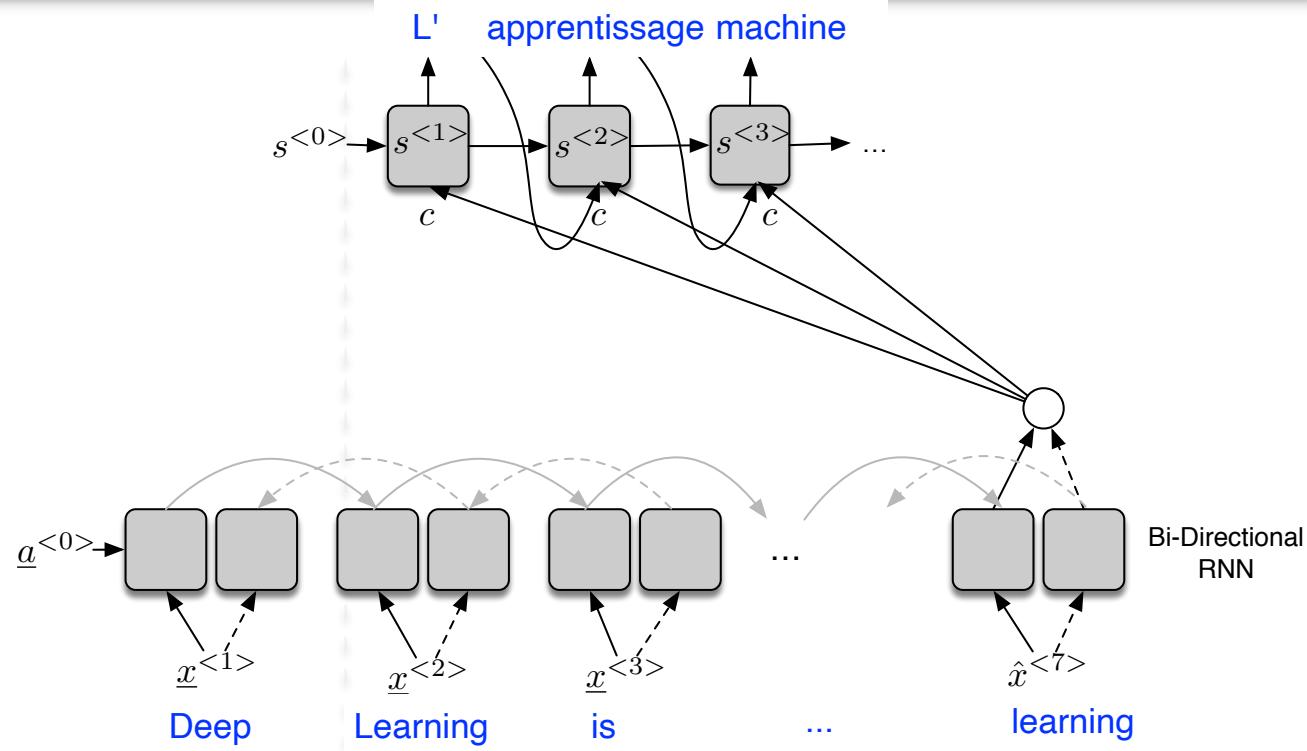


– Notation:

- $a^{<t>}$: encoder hidden states
- $s^{<t>}$: decoder hidden states

Sequence to sequence

Attention model



– In **usual encoder-decoder**:

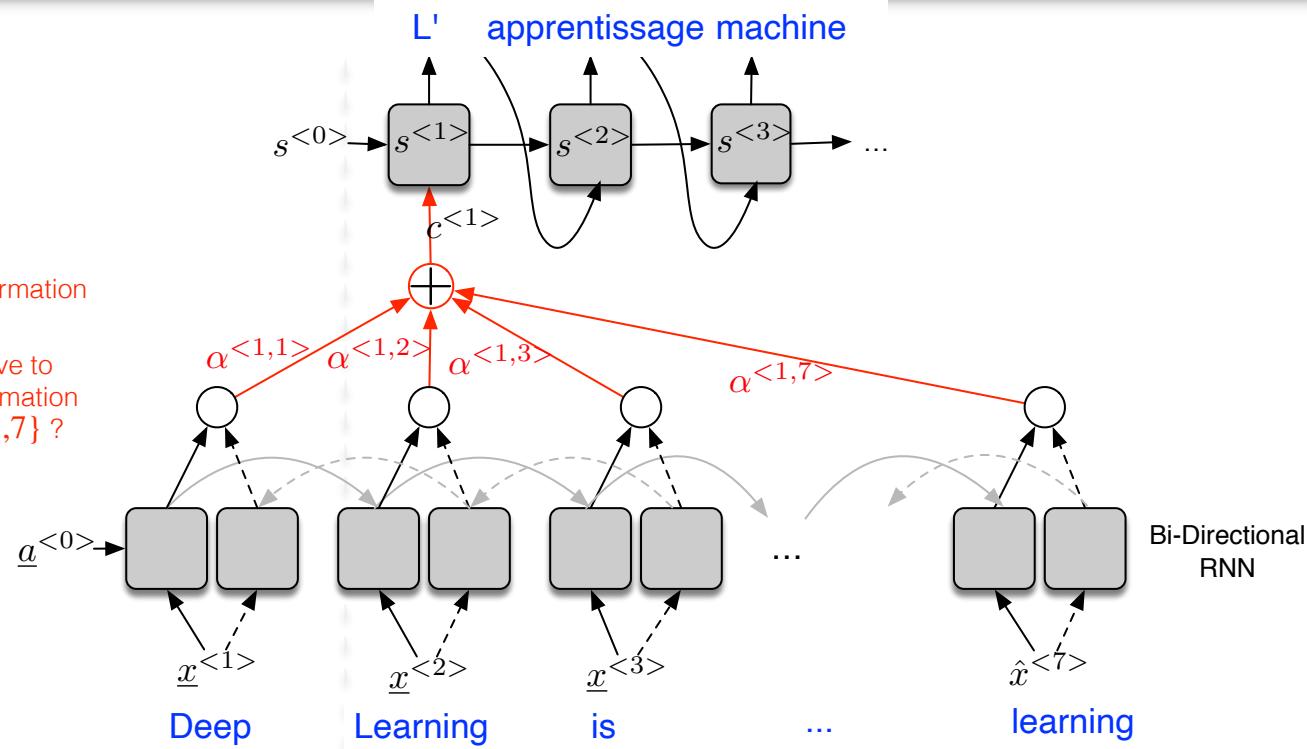
- we use the same context for all decoding time τ : $c^{(\tau)} = c$
- the context c corresponds to the encoding hidden state at time T_x : $c = a^{(T_x)}$

Sequence to sequence

Attention model

$\alpha^{(1,t)}$:

when generating information at time $\tau = 1$,
how much, do we have to pay attention on information at time $t = \{1,2,3,\dots,7\}$?



- In attention model:

- each decoding time τ has its own context: $c^{(\tau)}$
- the context $c^{(\tau)}$ is computed as a weighted sum of the encoding hidden states $a^{(t)}$

- weights $\alpha^{(\tau,t)}$:

- attention weights
- when generating information at time τ , how much, do we have to pay attention on information at time $t = \{1,2,3,\dots,7\}$

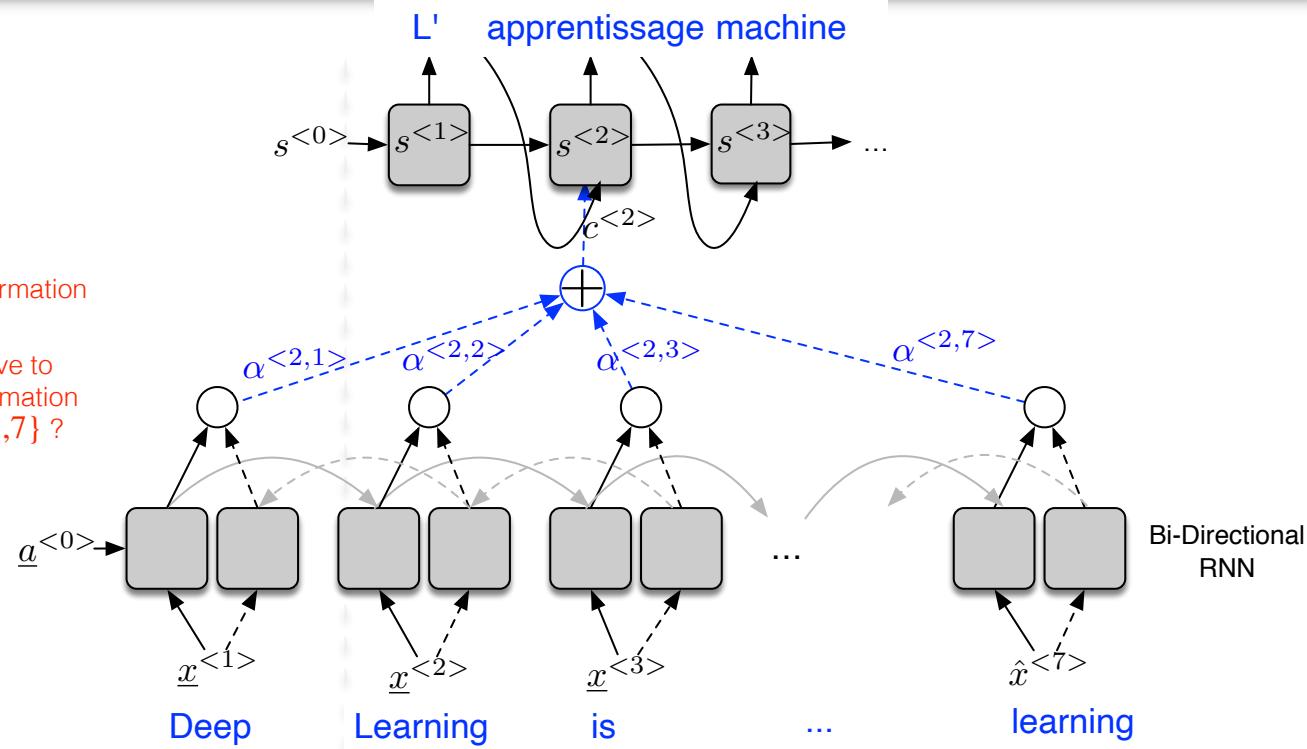
Sequence to sequence

Attention model

$\alpha^{(2,t)}$:

when generating information at time $\tau = 2$,

how much, do we have to pay attention on information at time $t = \{1,2,3,\dots,7\}$?



- In attention model:

- each decoding time τ has its own context: $c^{(\tau)}$
- the context $c^{(\tau)}$ is computed as a weighted sum of the encoding hidden states $a^{(t)}$

- weights $\alpha^{(\tau,t)}$:

- attention weights
- when generating information at time τ , how much, do we have to pay attention on information at time $t = \{1,2,3,\dots,7\}$

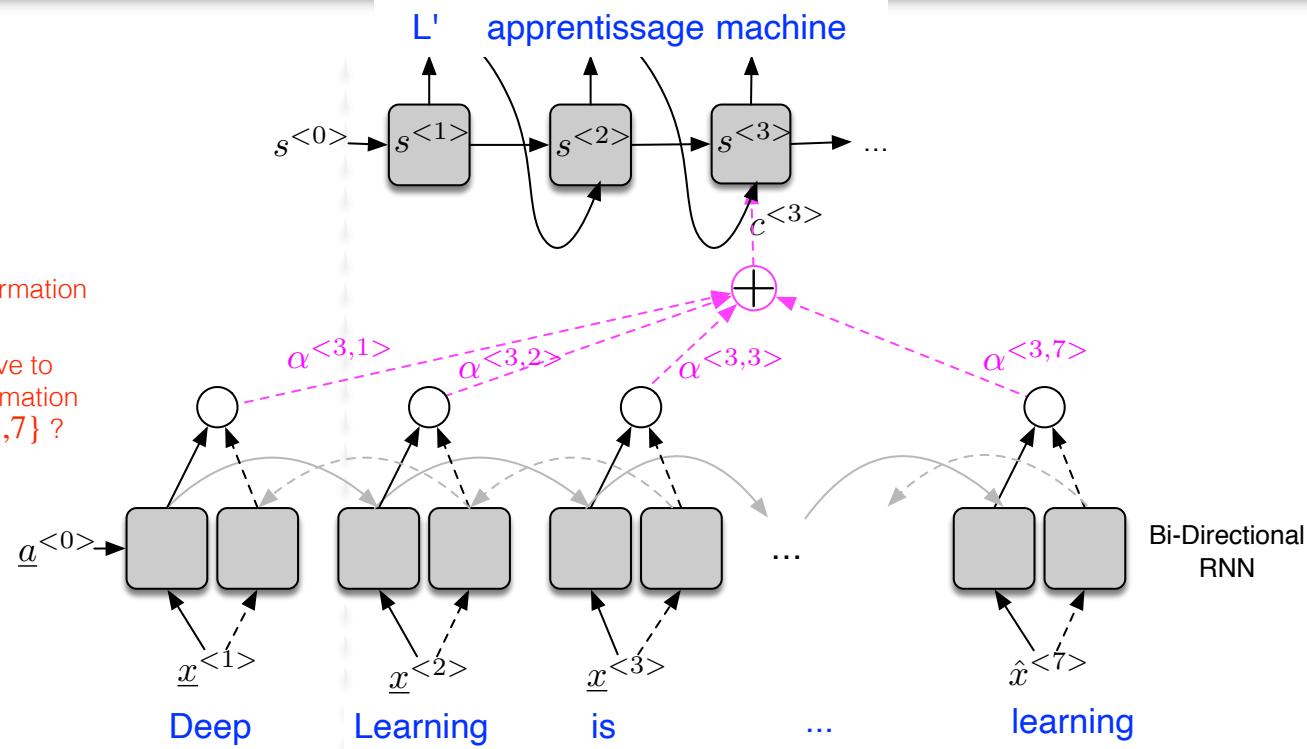
Sequence to sequence

Attention model

$\alpha^{(3,t)}$:

when generating information at time $\tau = 3$,

how much, do we have to pay attention on information at time $t = \{1,2,3,\dots,7\}$?



- In attention model:

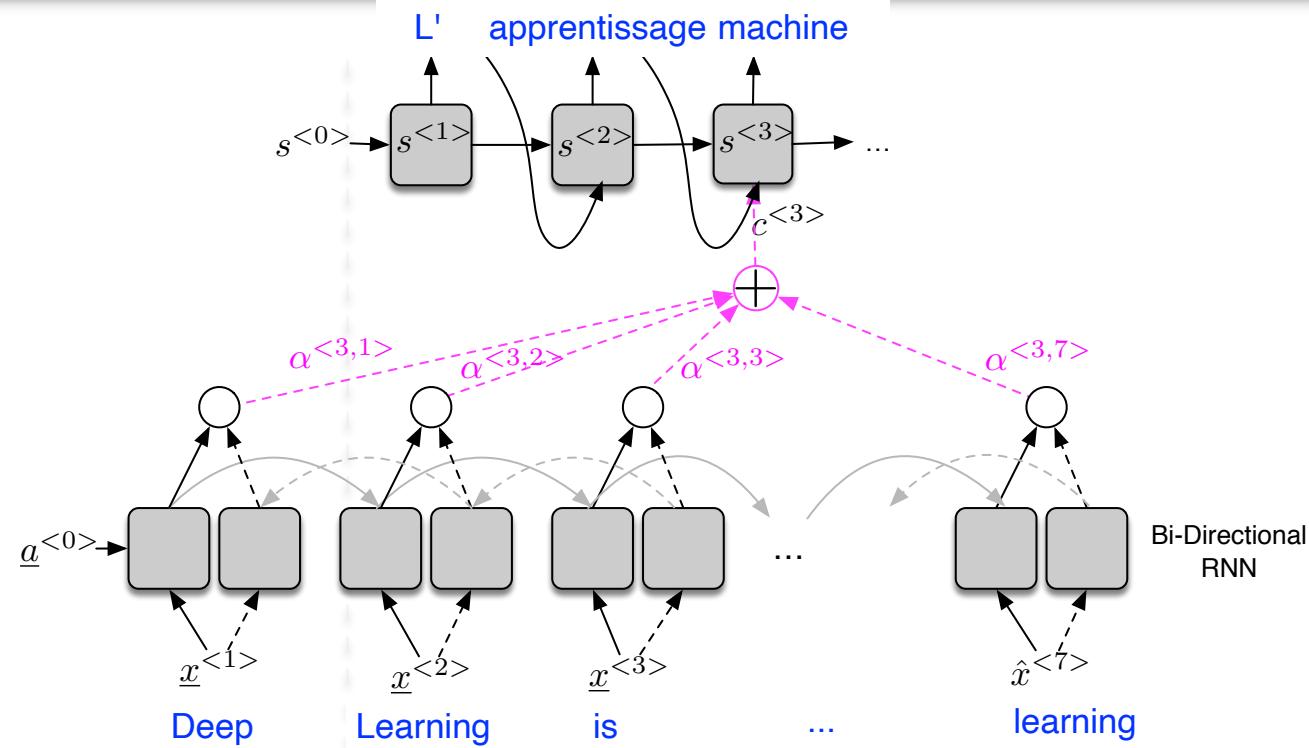
- each decoding time τ has its own context: $c^{(\tau)}$
- the context $c^{(\tau)}$ is computed as a weighted sum of the encoding hidden states $a^{(t)}$

- weights $\alpha^{(\tau,t)}$:

- attention weights
- when generating information at time τ , how much, do we have to pay attention on information at time $t = \{1,2,3,\dots,7\}$

Sequence to sequence

Attention model



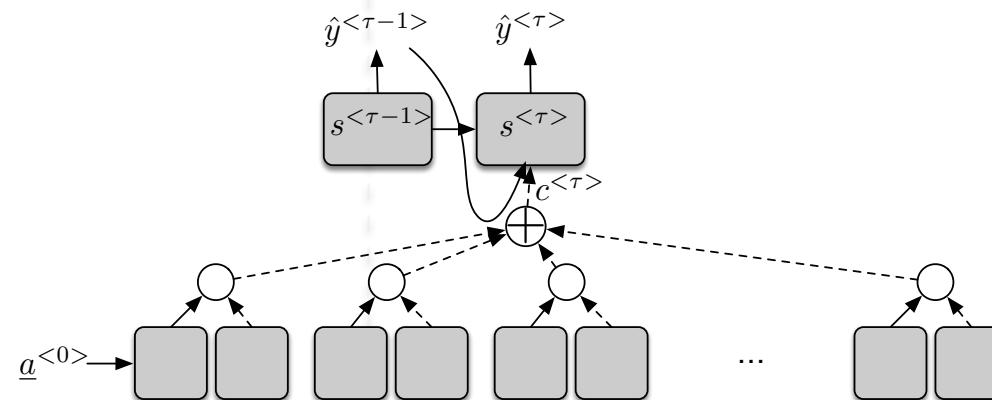
- Attention weights $\alpha^{(\tau=3,t)}$ describe an "alignment" between information at
 - **encoding time t :**
 - computed using $\vec{a}^{(t)}$ and $\overleftarrow{a}^{(t)}$; we note $a^{(t)} = [\vec{a}^{(t)}, \overleftarrow{a}^{(t)}]$
 - **decoding time $\tau = 3$:**
 - computed using $s^{(\tau=2)}$ (we do not yet observe $\tau = 3$)

Sequence to sequence

Attention model

- The **context vector** $c^{(\tau)}$ at decoding time τ
 - computed as the sum of the annotations $a^{(t)}$ weighted by their **attention weights** $\alpha^{(\tau,t)}$

$$c^{(\tau)} = \sum_t \alpha^{(\tau,t)} a^{(t)}$$



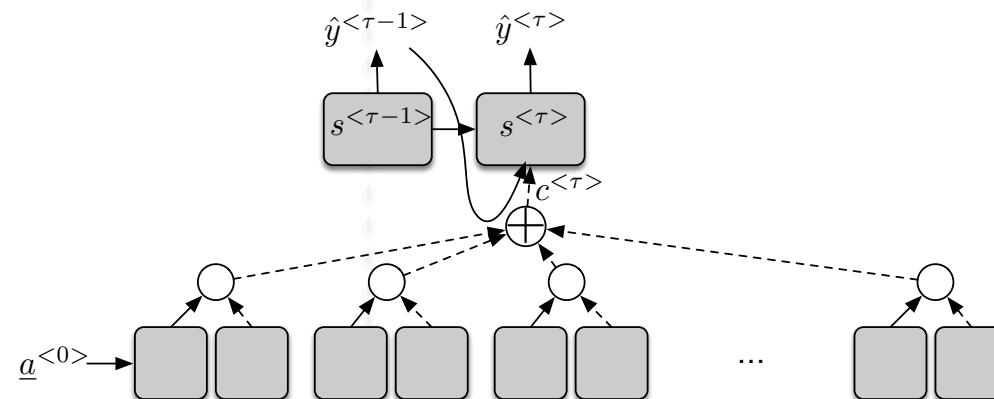
Sequence to sequence

Attention model

– The **attention weight** $\alpha^{(\tau,t)}$

- $\alpha^{(\tau,t)}$ represents the amount of attention $y^{(\tau)}$ should pay to $a^{(t)}$
- computed using a softmax on the **alignment model** $e^{(\tau,t)}$

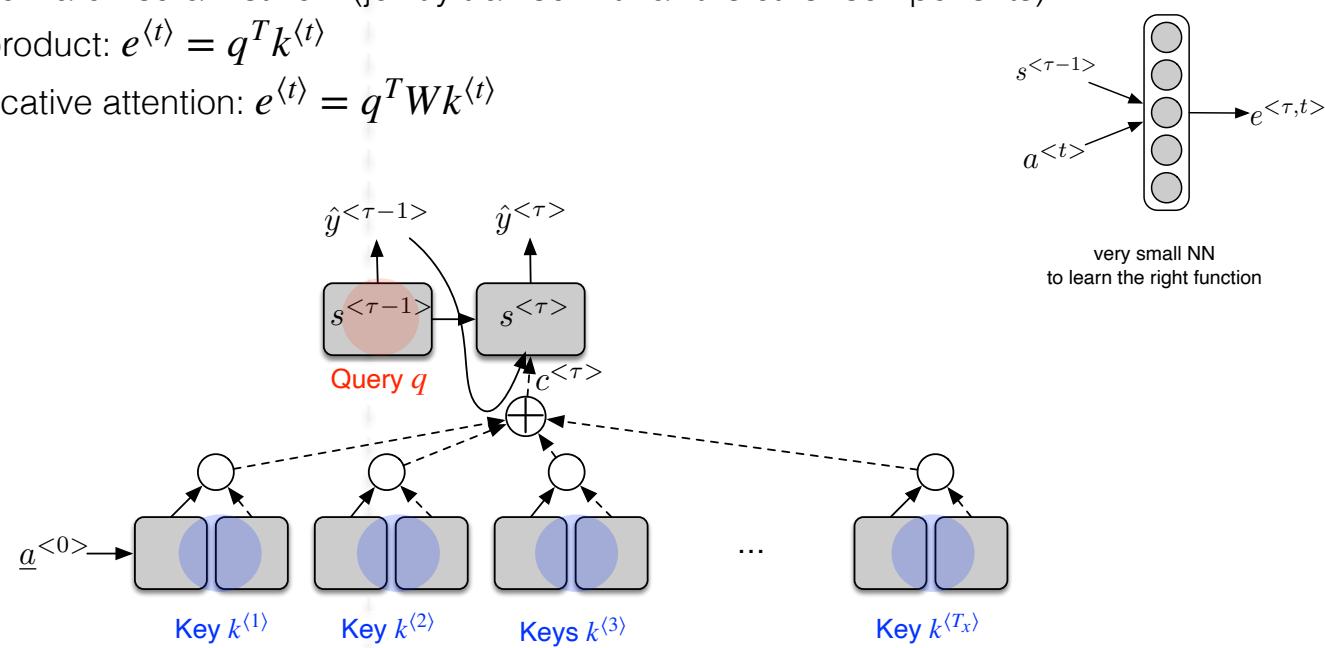
$$\alpha^{(\tau,t)} = \frac{\exp(e^{(\tau,t)})}{\sum_{t'=1}^{T_x} \exp(e^{(\tau,t')})} \text{ with } \sum_t \alpha^{(\tau,t)} = 1$$



Sequence to sequence

Attention model

- The **alignment model** $e^{(\tau,t)}$ scores how well the inputs around position t match the output at position τ
 - is computed using two inputs
 - **query**: the RNN hidden state $s^{(\tau-1)}$ of the output sequence (just before emitting $y^{(\tau)}$)
 - **keys**: the RNN hidden states $a^{(t)}$ of the input sentence.
 - $e^{(\tau,t)}$ is computed using $f(s^{(\tau-1)}, a^{(t)})$
 - can be a feedforward neural network (jointly trained with all the other components)
 - can be a dot-product: $e^{(\tau,t)} = q^T k^{(t)}$
 - can be multiplicative attention: $e^{(\tau,t)} = q^T W k^{(t)}$



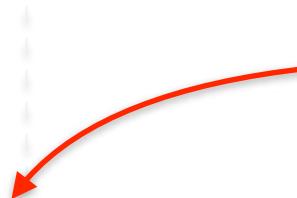
Sequence to sequence

Attention model

- Different **alignment models**

Name	Alignment score function	Citation
Content-base attention	$\text{score}(\mathbf{s}_t, \mathbf{h}_i) = \text{cosine}[\mathbf{s}_t, \mathbf{h}_i]$	Graves2014
Additive(*)	$\text{score}(\mathbf{s}_t, \mathbf{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\mathbf{s}_{t-1}; \mathbf{h}_i])$	Bahdanau2015
Location-Base	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \mathbf{s}_t)$ Note: This simplifies the softmax alignment to only depend on the target position.	Luong2015
General	$\text{score}(\mathbf{s}_t, \mathbf{h}_i) = \mathbf{s}_t^\top \mathbf{W}_a \mathbf{h}_i$ where \mathbf{W}_a is a trainable weight matrix in the attention layer.	Luong2015
Dot-Product	$\text{score}(\mathbf{s}_t, \mathbf{h}_i) = \mathbf{s}_t^\top \mathbf{h}_i$	Luong2015
Scaled Dot-Product(^)	$\text{score}(\mathbf{s}_t, \mathbf{h}_i) = \frac{\mathbf{s}_t^\top \mathbf{h}_i}{\sqrt{n}}$ Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.	Vaswani2017

Here
 \mathbf{s}_t corresponds to $\mathbf{s}^{(t)}$
 \mathbf{h}_i corresponds to $\mathbf{a}^{(i)}$



Sequence to sequence

Attention model

- Original sentence

This kind of experience is part of Disney's efforts to "extend the lifetime of its series and build new relationships with audiences via digital platforms that are becoming ever more important," he added.

- Encoder/Decoder without attention model

Ce type d'expérience fait partie des initiatives du Disney pour "prolonger la durée de vie de ses nouvelles et de développer des liens avec les lecteurs numériques qui deviennent plus complexes.

- Encoder/Decoder with attention model

Ce genre d'expérience fait partie des efforts de Disney pour "prolonger la durée de vie de ses séries et créer de nouvelles relations avec des publics via des plateformes numériques de plus en plus importantes", a-t-il ajouté.

Visualisation of $\alpha^{(\tau,t)}$ (English to French)

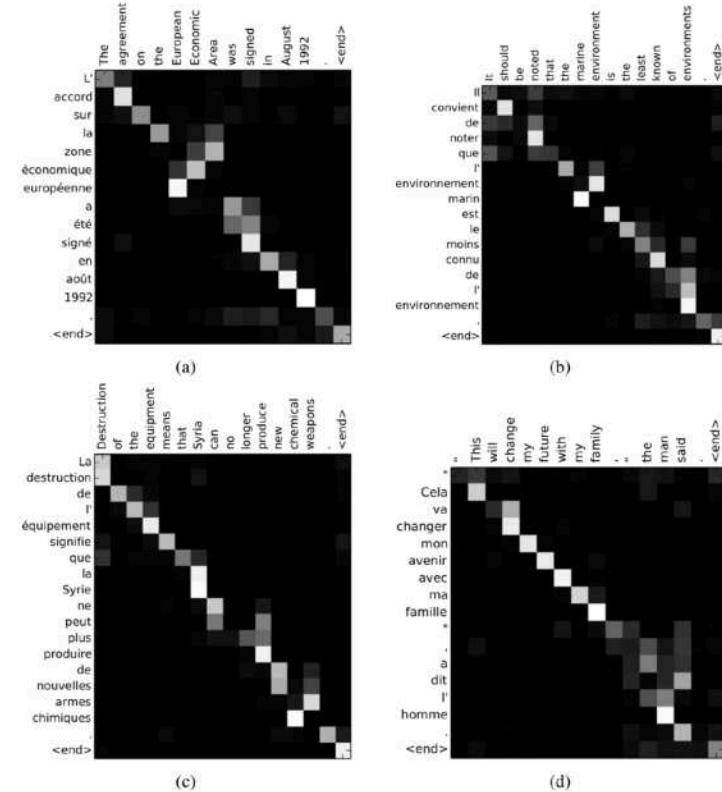
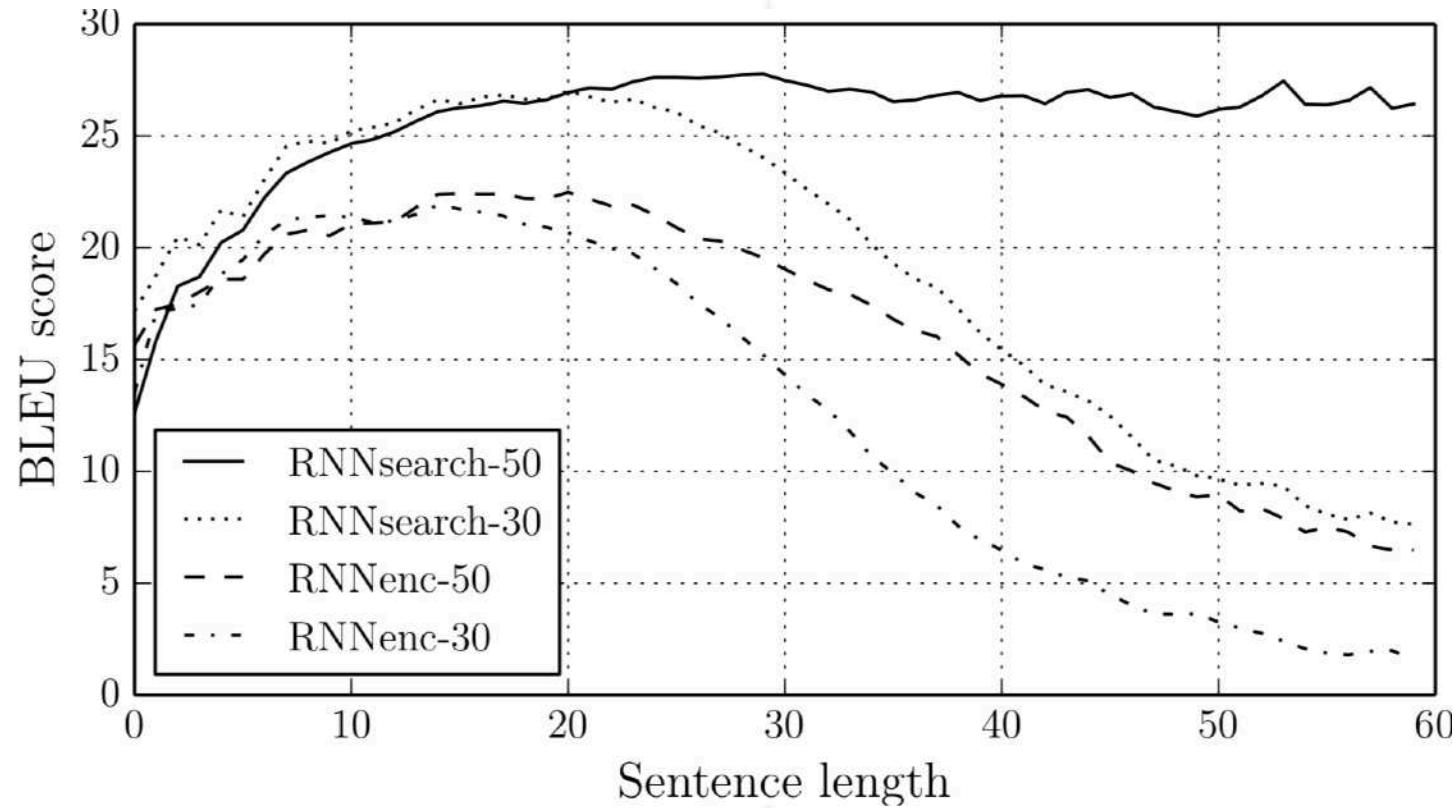


Figure 3: Four sample alignments found by RNNsearch-50. The x-axis and y-axis of each plot correspond to the words in the source sentence (English) and the generated translation (French), respectively. Each pixel shows the weight α_{ij} of the annotation of the j -th source word for the i -th target word (see Eq. (6)), in grayscale (0: black, 1: white). (a) an arbitrary sentence. (b-d) three randomly selected samples among the sentences without any unknown words and of length between 10 and 20 words from the test set.

Sequence to sequence

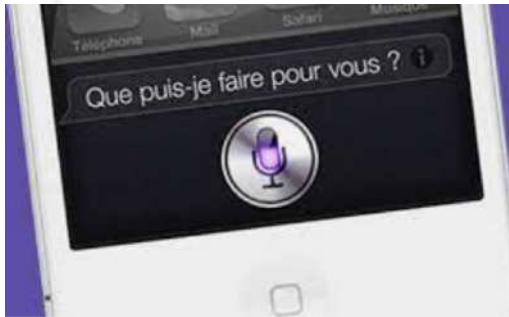
Attention model

- Bleu score for long sentences

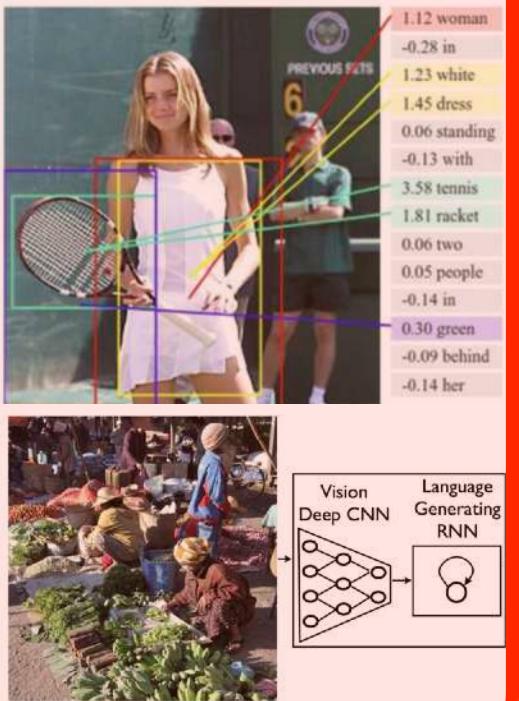


Applications

Automatic Speech Recognition



Automatic picture captioning



Self Driving Cars



DALL-E



DEEP LEARNING HAS MASTERED GO Google Alpha Go

nature International weekly journal of science

Home News & Comment Research Careers & Jobs Current Issue Archives About & Write Profile Article > Science > Nature News > News > Article

AlphaGo reveals secret test of AI bot to beat top Go players

Updated version of DeepMind's AlphaGo program beaten regularly online competitor.

Mastering the game of Go with deep neural networks and tree search

David Silver, Aljaž Mazzega, Chris J. Maddison, Arthur O'Dea, Laurent Bégin, George van den Driessche, Jürgen Schmidhuber, Murray Amrullagau, Vedat Paravancıoğlu, Miro Lázović, Hamid Hassani, Dominik Grewe, John Mnih, Isao Kubota, Ryo Saito, Timothy Lillicrap, Mandarake Leach, Koen Koenigsmann, Thore Graepel & Demis Hassabis

Nature 529, 484–489 (28 January 2016) | doi:10.1038/nature18861
Received: 11 November 2015 | Accepted: 05 January 2016 | Published online: 27 January 2016



Style Transfer



A group shopping at an outdoor market.

There are many vegetables at the fruit stand.

Neural Machine Translation

DeepL Traducteur DeepL Pro Découvrir DeepL Pro Connexion

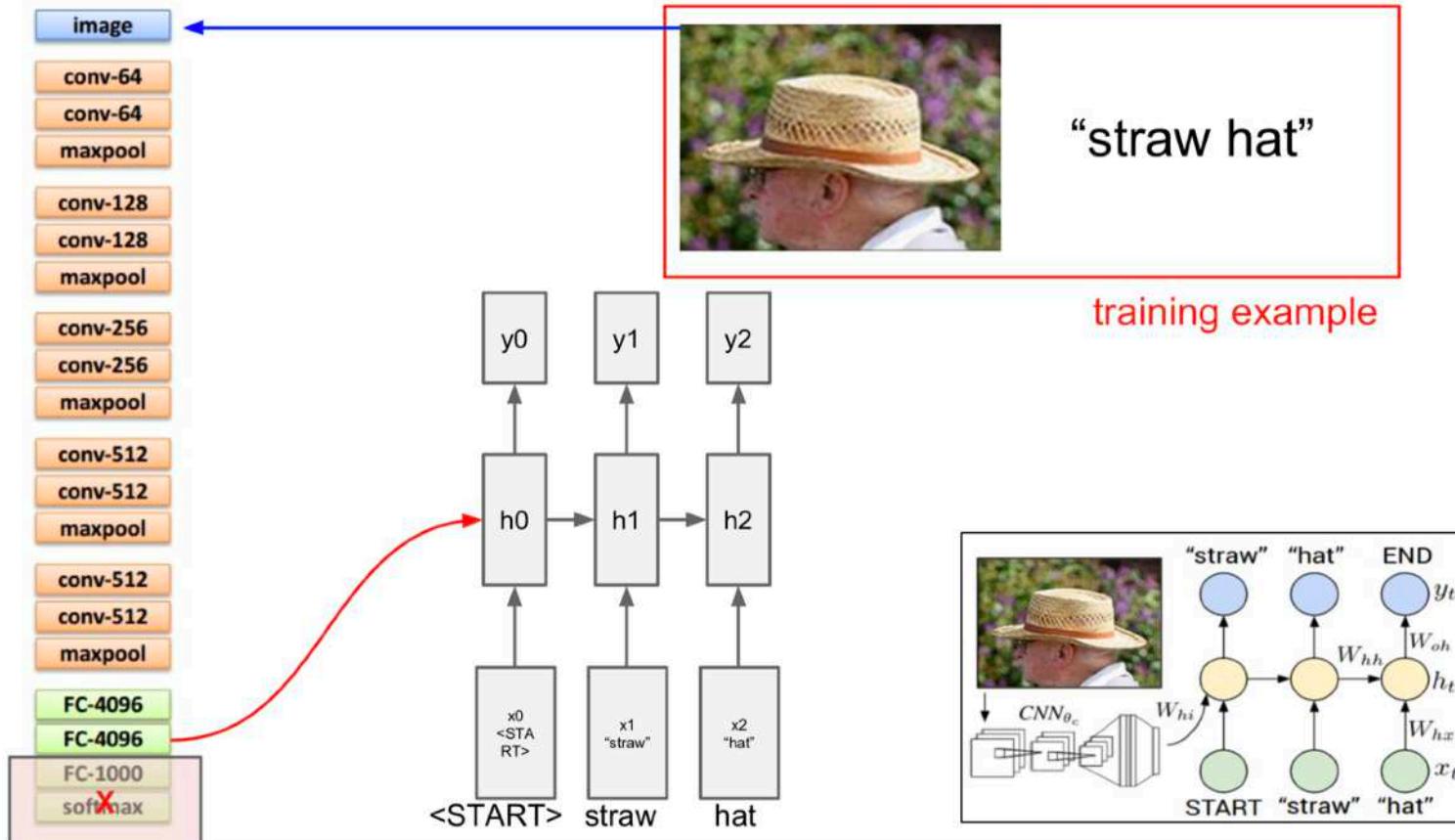
Anglais (langue détectée) ▾ Fr... ▾ forme/informel ▾ Glossaire

Deep learning (also known as deep structured learning) is part of a broader family of machine learning methods based on artificial neural networks with representation learning. Learning can be supervised, semi-supervised or unsupervised.

L'apprentissage profond (également appelé apprentissage structuré profond) fait partie d'une famille plus large de méthodes d'apprentissage automatique basées sur les réseaux neuronaux artificiels avec apprentissage par représentation. L'apprentissage peut être supervisé, semi-supervisé ou non supervisé.

Sequence to sequence

Application: Image captioning



SOURCE: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lec10.pdf

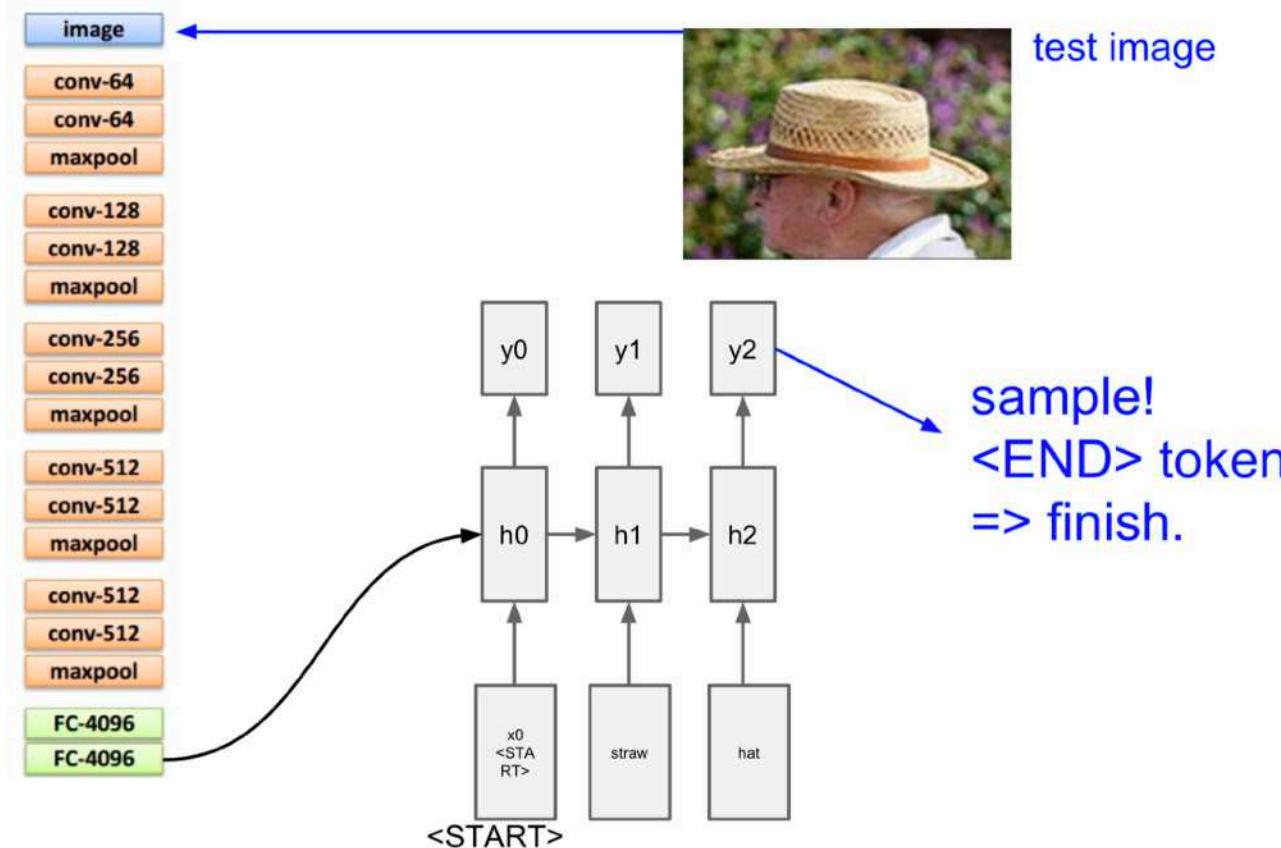
[Mao et al. 2014 "Deep captioning with multimodal recurrent neural networks (m-RNN)"] [LINK](#)

[Vinyals et al. 2015 "Show and tell: A neural image caption generator"] [LINK](#)

[Karpathy et al. 2015 "Deep visual-semantic alignments for generating image descriptions"] [LINK](#)

Sequence to sequence

Application: Image captioning



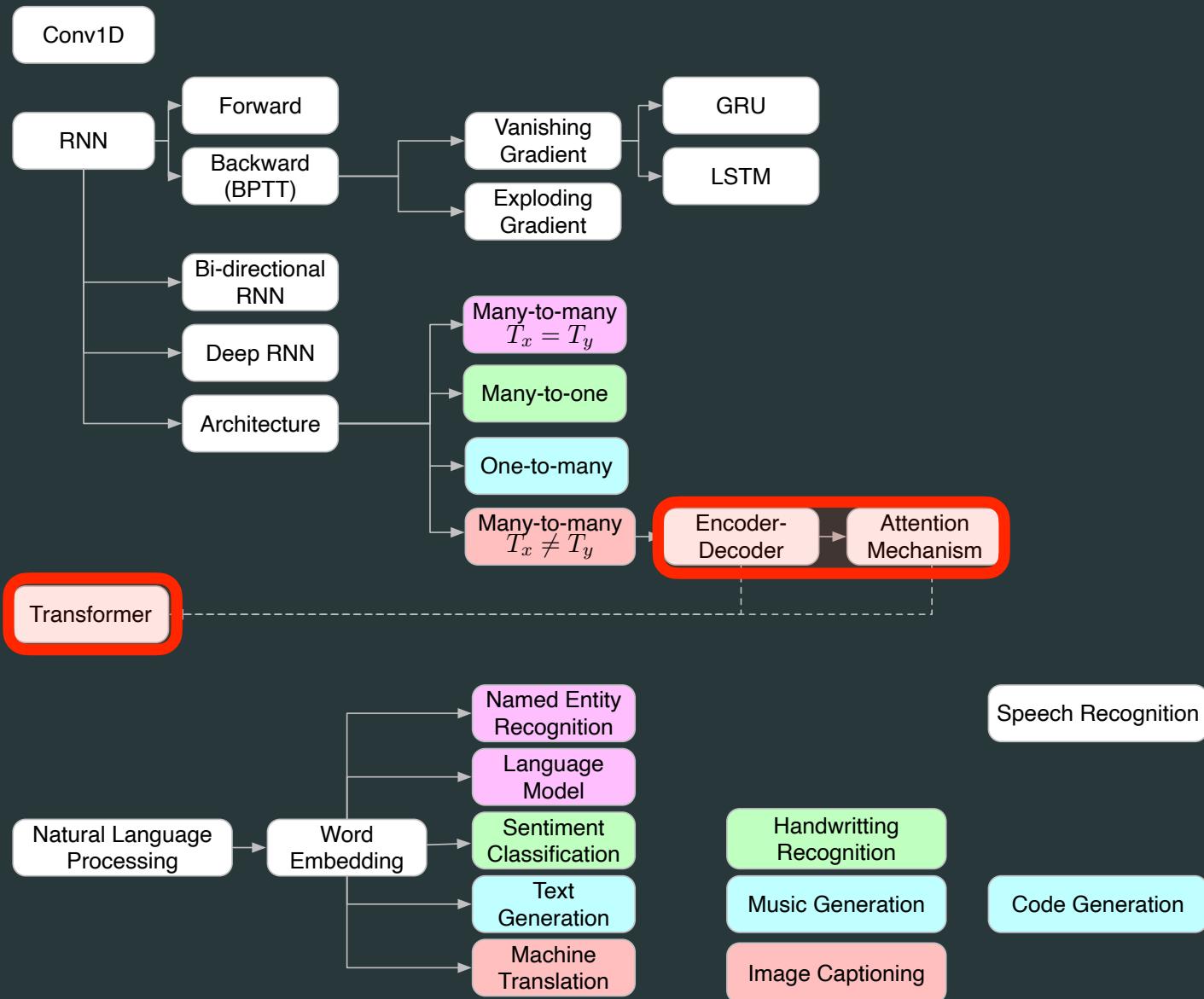
SOURCE: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

[Mao et al. 2014 "Deep captioning with multimodal recurrent neural networks (m-RNN)"] [LINK](#)

[Vinyals et al. 2015 "Show and tell: A neural image caption generator"] [LINK](#)

[Karpathy et al. 2015 "Deep visual-semantic alignments for generating image descriptions"] [LINK](#)

Transformer

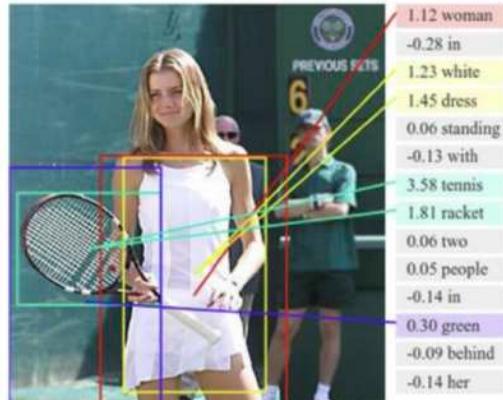


Applications

Automatic Speech Recognition



Automatic picture captioning

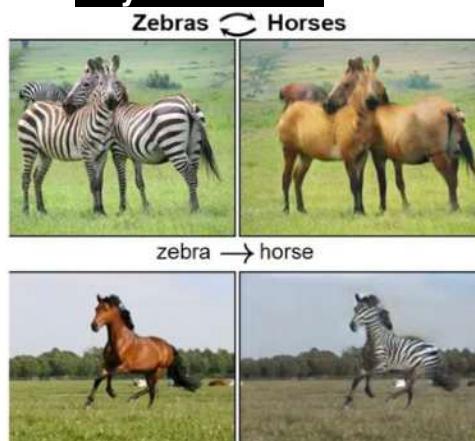


A group shopping at an outdoor market.
There are many vegetables at the fruit stand.

Self Driving Cars



Style Transfer



horse → zebra



DALL-E



DEEP LEARNING HAS MASTERED GO

Google Alpha Go

The screenshot shows a news article from the journal 'nature' (Volume 508, Number 7501, 28 January 2014). The title reads 'Google reveals secret test of AI bot to beat top Go players'. The article discusses the AlphaGo program developed by Google DeepMind, which defeated Lee Sedol, a professional Go player. The page includes a sidebar with links to other news stories and a navigation bar for the journal.



Neural Machine Translation

The screenshot shows the DeepL Pro interface. A French sentence 'L'apprentissage profond (également appelé apprentissage structuré profond) fait partie d'une famille plus large de méthodes d'apprentissage automatique basées sur les réseaux neuronaux artificiels avec apprentissage par représentation. L'apprentissage peut être supervisé, semi-supervisé ou non supervisé.' is being translated into English. The English translation is displayed below the original text, along with options for 'Anglais (langue détectée)' and 'Connexion'.

Applications Generative AI (GenAI)

NLP

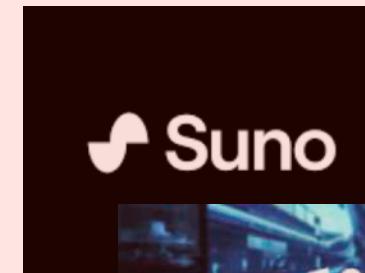


Video



<https://www.youtube.com/watch?v=8LjfxFQ5iNk>

Music



Computer Vision



Speech (deep fake)



Voice dubbing



Transformer

- **Goal ?**

- get rid of complex recurrent or convolutional neural networks

- **How ?**

- simple network architecture (still an encoder decoder) based on
 - (1) attention mechanisms \Rightarrow self-attention
 - (2) encoder blocks
 - (3) decoder blocks
 - (4) positional encoding

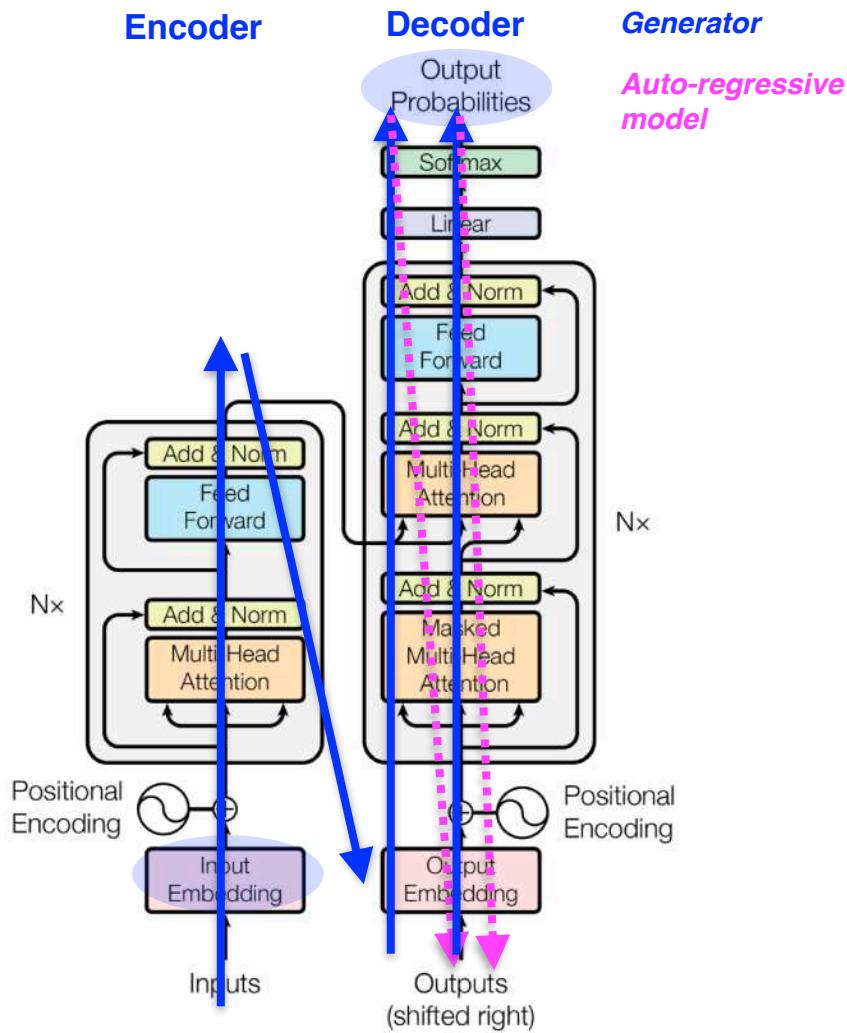


Figure 1: The Transformer - model architecture.

Transformer

(1) Self-Attention

Self-Attention ?

– **Example:** we want to translate the following input sequence

- "The animal didn't cross the street because it was too tired"



- **Problem:** What does "it" in this sentence refer to ? Is it referring to "street" or "animal" ?

– Self-Attention ?

- As the model processes each word (each position in the input sequence), self attention allows the process to look at other positions in the sequence for clues that can help lead to a better encoding for this word
- **Example:** when the model is processing the word "it", self-attention allows to associate "it" with "animal"

Transformer

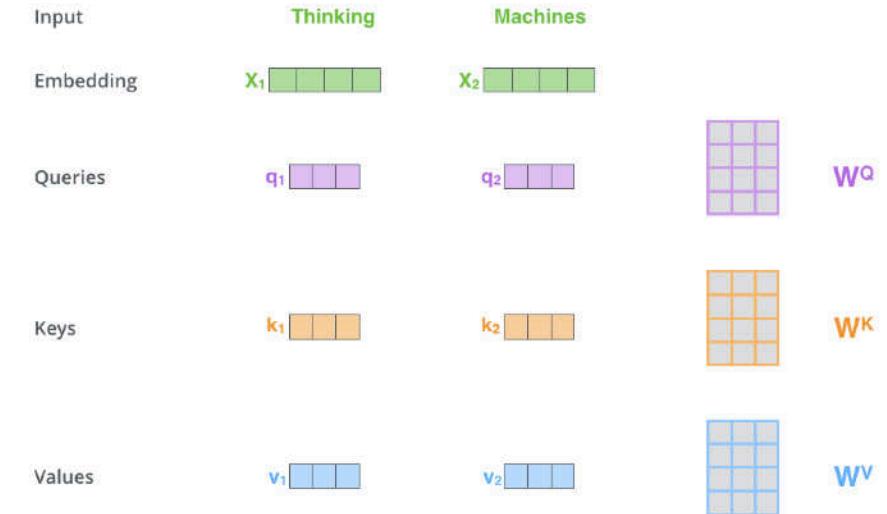
(1) Self-Attention

Implementation

– First step :

- for each input vectors $\mathbf{x}_i \in \mathbb{R}^d$: create three vectors
 - a Query vector $\mathbf{q}_i \in \mathbb{R}^{d'}$
 - a Key vector $\mathbf{k}_i \in \mathbb{R}^{d'}$
 - a Value vector $\mathbf{v}_i \in \mathbb{R}^{d'}$
- the vectors are created by multiplying \mathbf{x}_i by three matrices $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V$
 - $\mathbf{q}_i = \mathbf{x}_i \mathbf{W}^Q \quad (1,d') = (1,d)(d,d')$
 - $\mathbf{k}_i = \mathbf{x}_i \mathbf{W}^K$
 - $\mathbf{v}_i = \mathbf{x}_i \mathbf{W}^V$

- $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V$ are trained during the training process
- $d' < d$: the new vectors are smaller in dimension than the embedding vector



Multiplying \mathbf{x}_1 by the \mathbf{W}^Q weight matrix produces \mathbf{q}_1 , the "query" vector associated with that word. We end up creating a "query", a "key", and a "value" projection of each word in the input sentence.

Transformer

(1) Self-Attention

Implementation

– Second step : calculate a score, an "attention"

- α_{ij} = while encoding position i (query), how much should we pay attention on position j (key)

- **Example** : calculate self-attention for "Thinking"

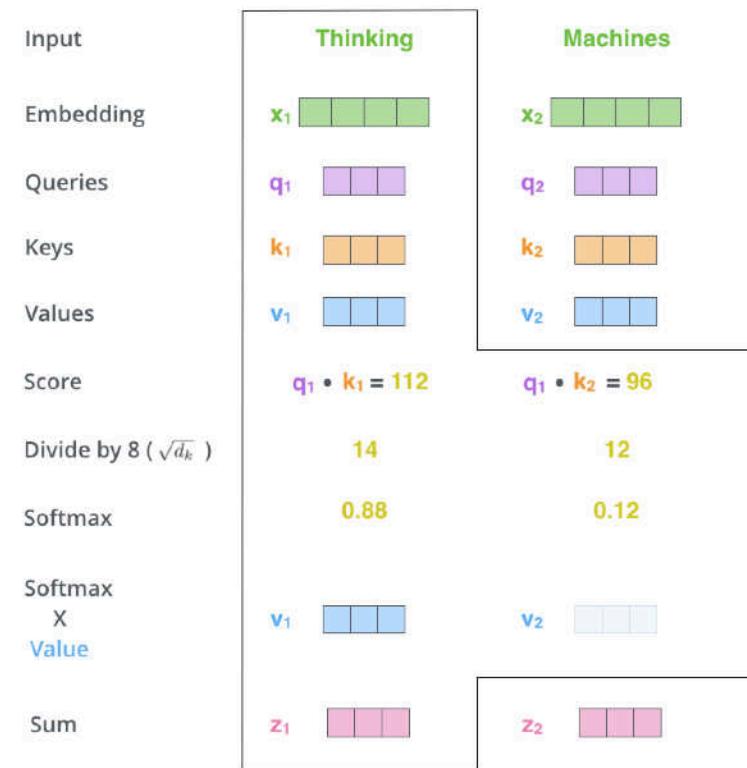
- α_{ij} ?

- dot product of query q_i with key k_j
- divide by $\sqrt{d'}$ (square root of the dimension of the key)

$$e_{ij} = \frac{\mathbf{q}_i \mathbf{k}_j^T}{\sqrt{d'}} \quad (1, d')(d', 1)$$

- normalise using softmax

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=0}^{T_k} \exp(e_{ik})}$$



Transformer

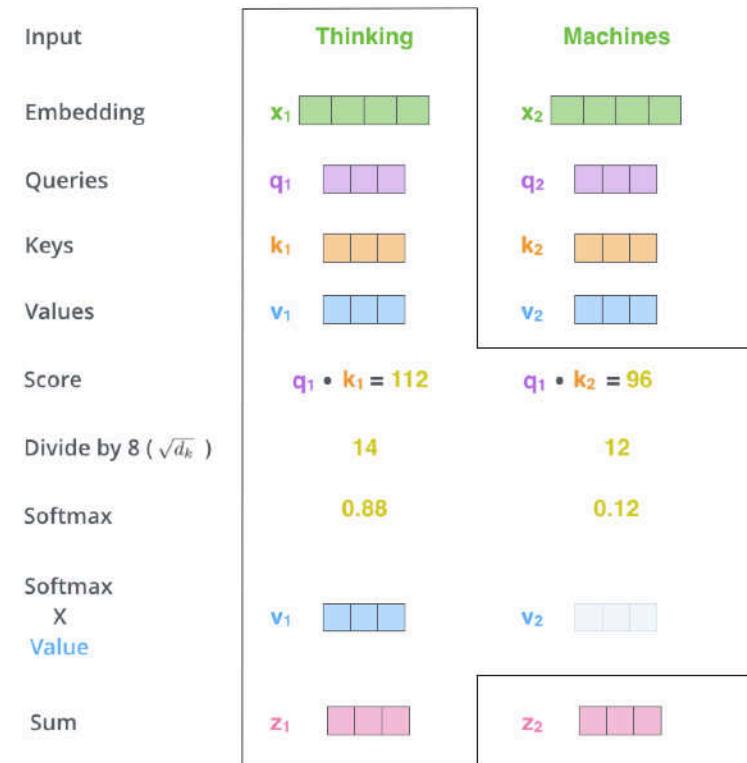
(1) Self-Attention

Implementation

– Third step :

- weighted value: multiply each **value** by the **score**
 - keep intact the values of the word(s) we want to focus on, and drown-out irrelevant words
- final: **sum up the weighted value vectors**
 - produces the output \mathbf{z}_i of the self-attention layer at this position

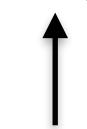
$$\mathbf{z}_i = \sum_{j=0}^{T_k} \alpha_{ij} \mathbf{v}_j$$



Transformer

(1) Self-Attention

$$z_1 = \sum_t \alpha_{1,t} v_t$$



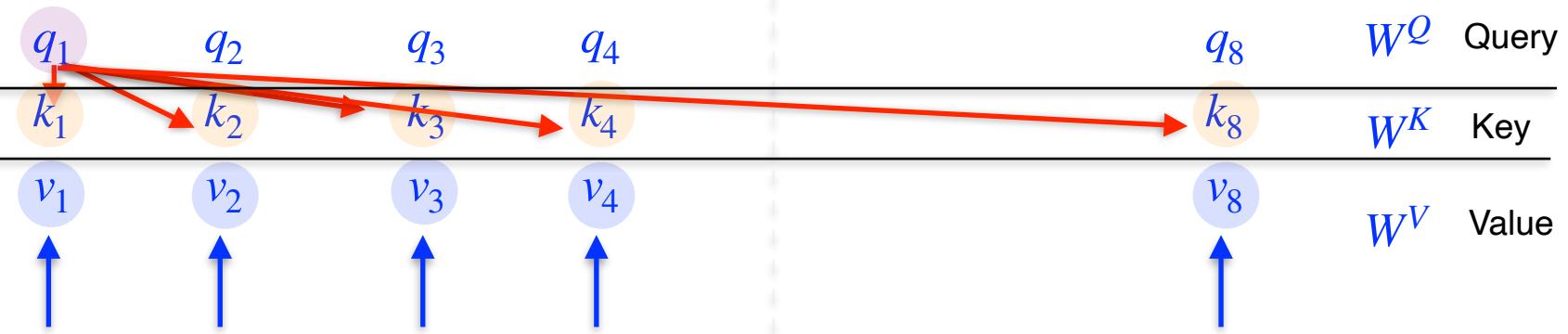
$$\alpha_{1,1} = q_1 \cdot k_1$$

$$\alpha_{1,2} = q_1 \cdot k_2$$

$$\alpha_{1,3} = q_1 \cdot k_3$$

$$\alpha_{1,4} = q_1 \cdot k_4$$

$$\alpha_{1,8} = q_1 \cdot k_8$$



The animal didn't cross the street because it was too tired

x_1

x_2

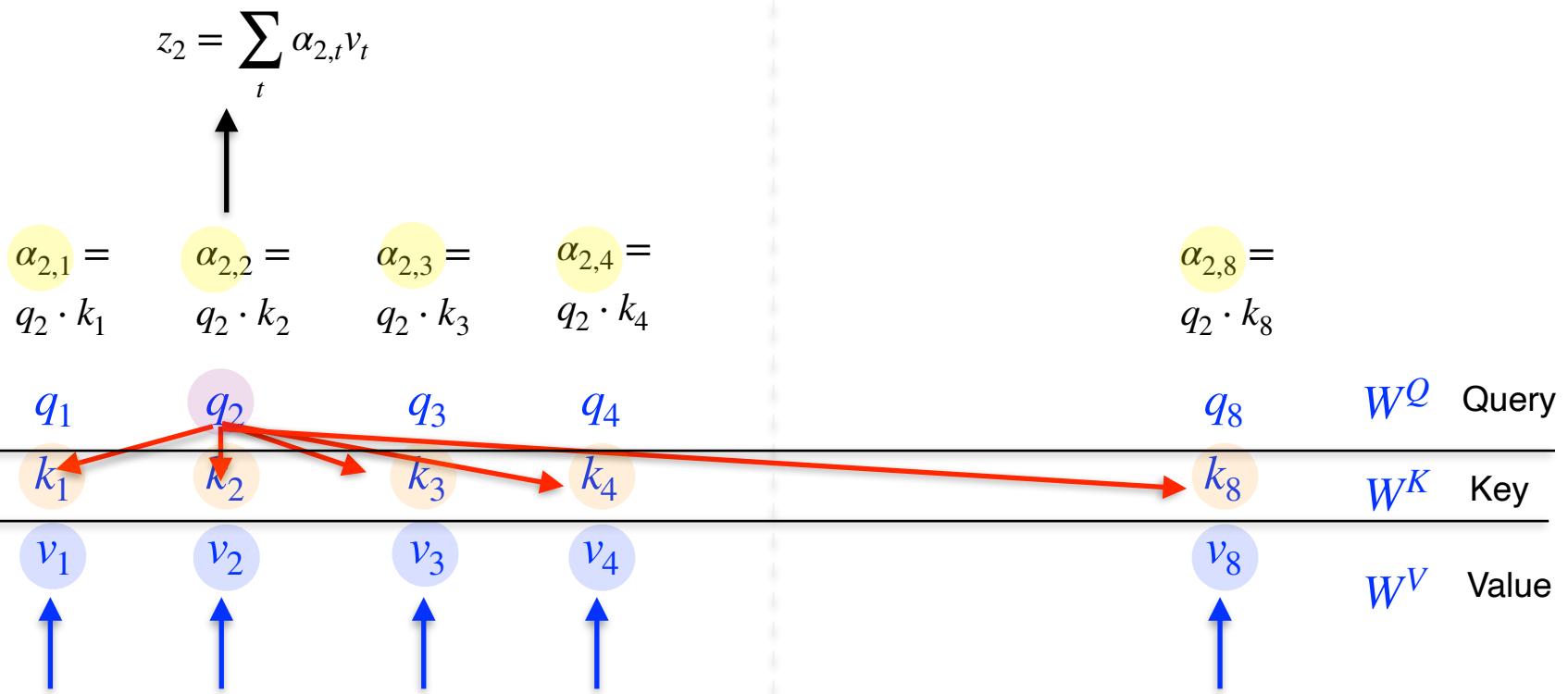
x_3

x_4

x_8

Transformer

(1) Self-Attention



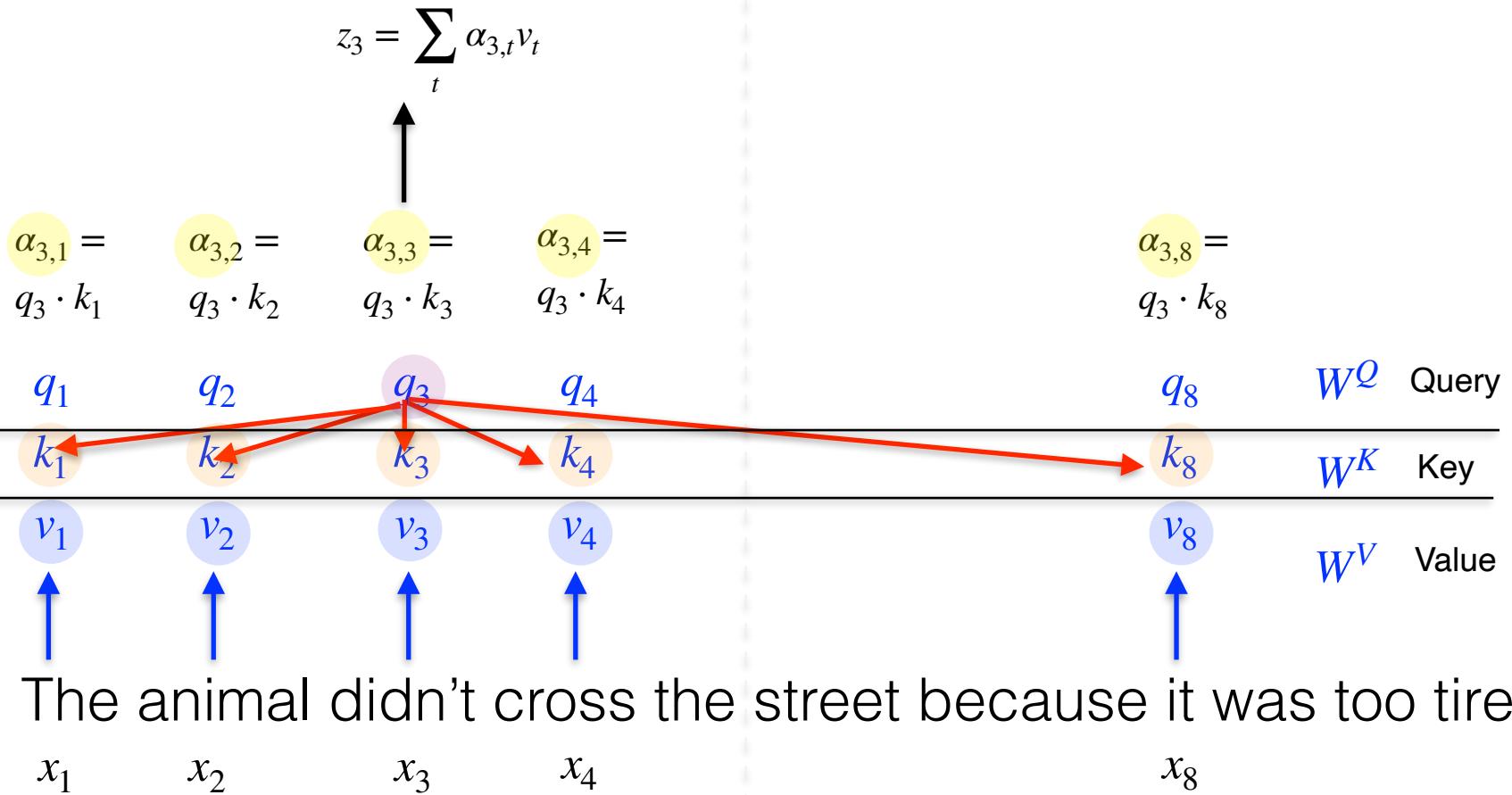
The animal didn't cross the street because it was too tired

$x_1 \quad x_2 \quad x_3 \quad x_4$

x_8

Transformer

(1) Self-Attention



Transformer

(1) Self-Attention

Implementation

- We can express all computation in **Matrix form** (for all time steps i jointly)

- $\mathbf{Q} = \mathbf{X} \mathbf{W}^Q$ $(T, d') = (T, d)(d, d')$

- $\mathbf{K} = \mathbf{X} \mathbf{W}^K$

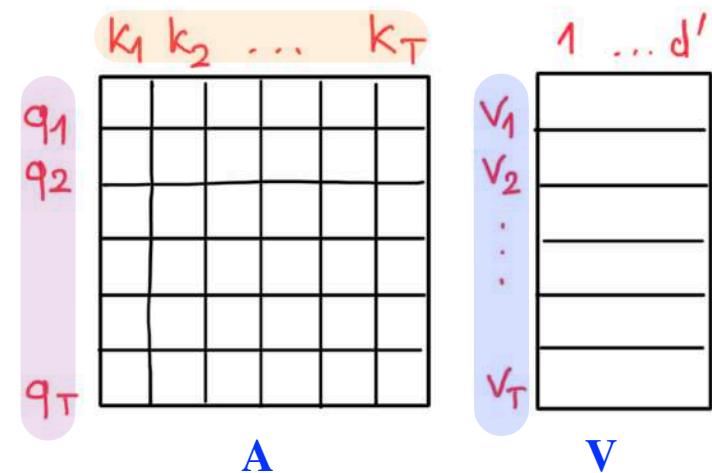
- $\mathbf{V} = \mathbf{X} \mathbf{W}^V$

$$\mathbf{E} = \frac{\mathbf{Q} \mathbf{K}^T}{\sqrt{d'}} \quad (T_q, T_k) = (T_q, d')(d', T_k)$$

$$\alpha_{ij} = \frac{\exp(\mathbf{E}_{ij})}{\sum_k \exp(\mathbf{E}_{ik})}$$

$$\mathbf{Z} = \mathbf{A} \mathbf{V}$$

$$(T_q, d') = (T_q, T_k)(T_k, d')$$



Transformer

(1) Self-Attention

Multi-head Self-Attention

- Several attention heads (often 8)

- $\mathbf{Q}_0 = \mathbf{XW}_0^Q, \mathbf{K}_0 = \mathbf{XW}_0^K, \mathbf{V}_0 = \mathbf{XW}_0^V$
- $\mathbf{Q}_1 = \mathbf{XW}_1^Q, \mathbf{K}_1 = \mathbf{XW}_1^K, \mathbf{V}_1 = \mathbf{XW}_1^V$
- ...

- Leads to several values

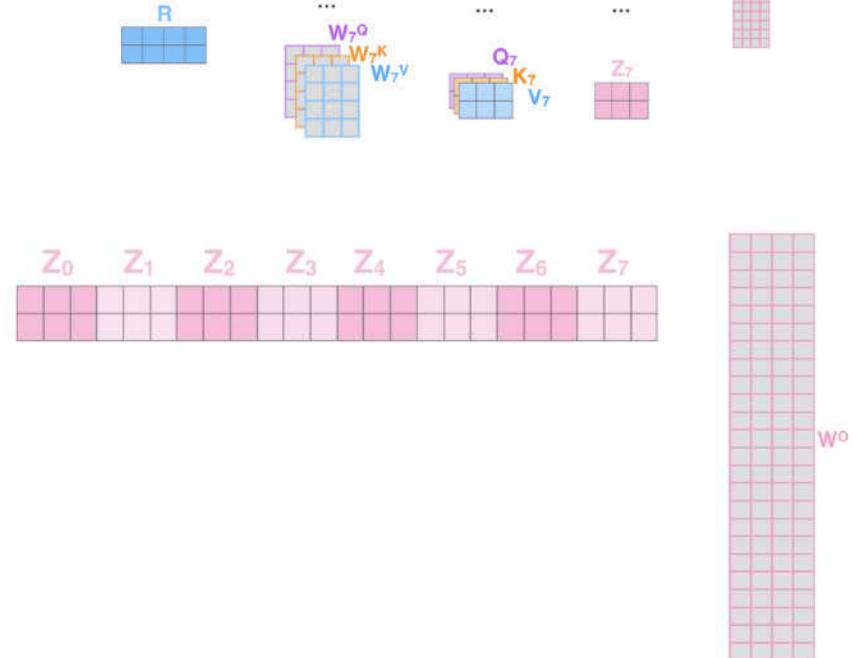
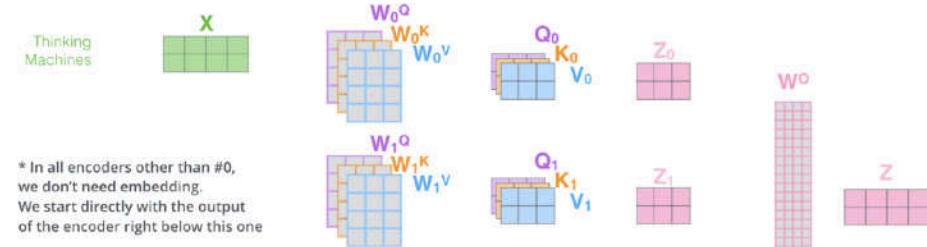
- $\mathbf{Z}_0 (T_q, d')$
- $\mathbf{Z}_1 (T_q, d')$
- ...

- Which are concatenated

- $\mathbf{Z} (T_q, 8 * d')$

- Then projected again using $\mathbf{W}^O (8 * d', d)$

- 1) This is our input sentence*
- 2) We embed each word*
- 3) Split into 8 heads. We multiply \mathbf{X} or \mathbf{R} with weight matrices
- 4) Calculate attention using the resulting $\mathbf{Q}/\mathbf{K}/\mathbf{V}$ matrices
- 5) Concatenate the resulting \mathbf{Z} matrices, then multiply with weight matrix \mathbf{W}^O to produce the output of the layer



Transformer

(2) Encoder blocks

One encoder block

– Input $\{\mathbf{x}_i\}_{i \in \{0, \dots, T-1\}}$:

- words turned into vectors using an **embedding algorithm** (only for bottom-most encoder)

– Two sub-layers

- (1) **Self-Attention** layer,

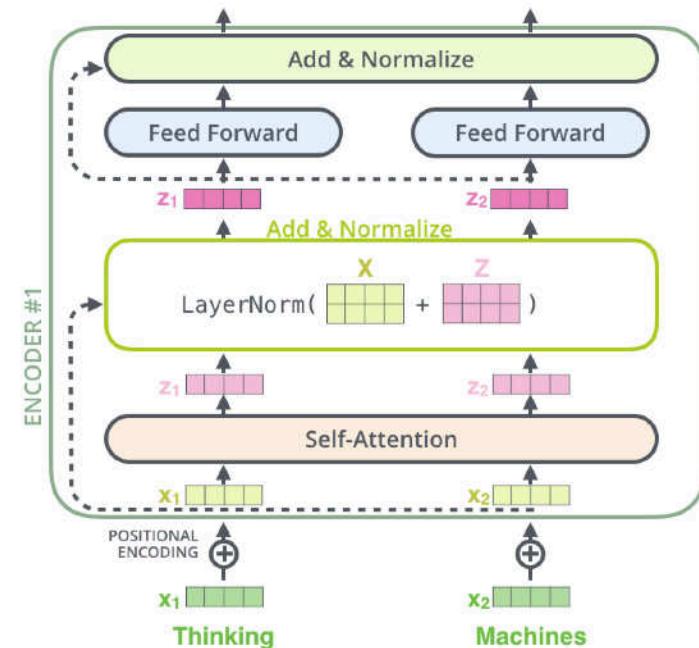
- compute dependencies between the inputs
 - output: $\{\mathbf{z}_i\}_{i \in \{0, \dots, T-1\}}$

- (2) **Feed Forward** neural network,

- $FFN(\mathbf{z}_i) = \max(0, \mathbf{z}_i W_1 + b_1) W_2 + b_2$
 - process each element \mathbf{z}_i **independently**
 - the exact same network is applied to each position: can be process in parallel

– Each sub-layer has

- a **residual** connection around it
 - a **layer-normalisation** step
 - normalises the inputs across the features dimension (instead of the batch dimension)



Transformer

(2) Encoder blocks

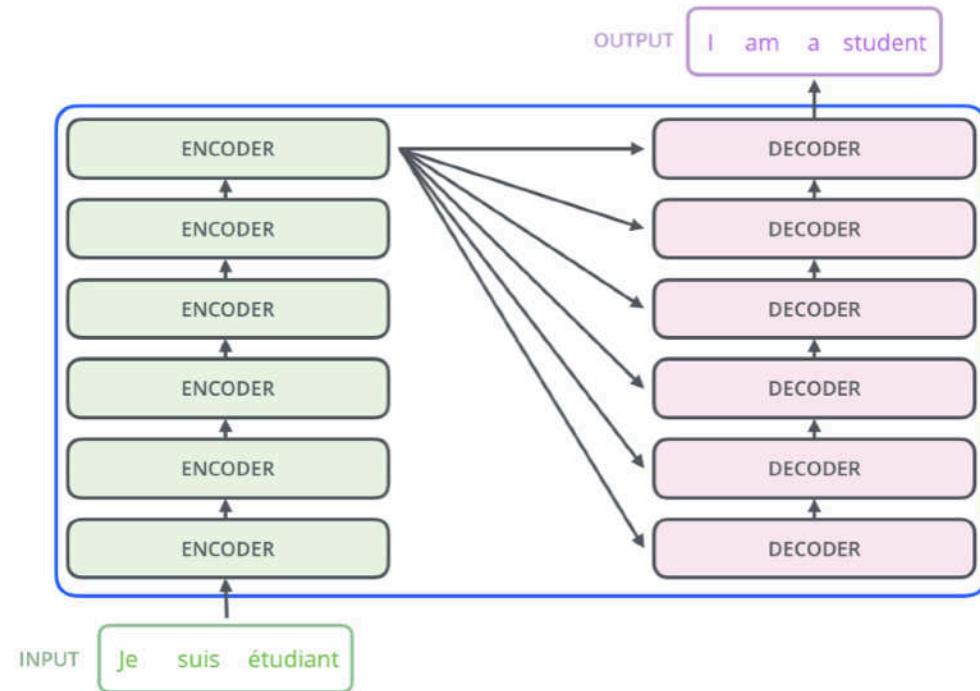
The whole encoder

– Encoding component

- stack of 6 encoder blocks on top of each other

– Decoding component

- stack of 6 decoders blocks on top of each other



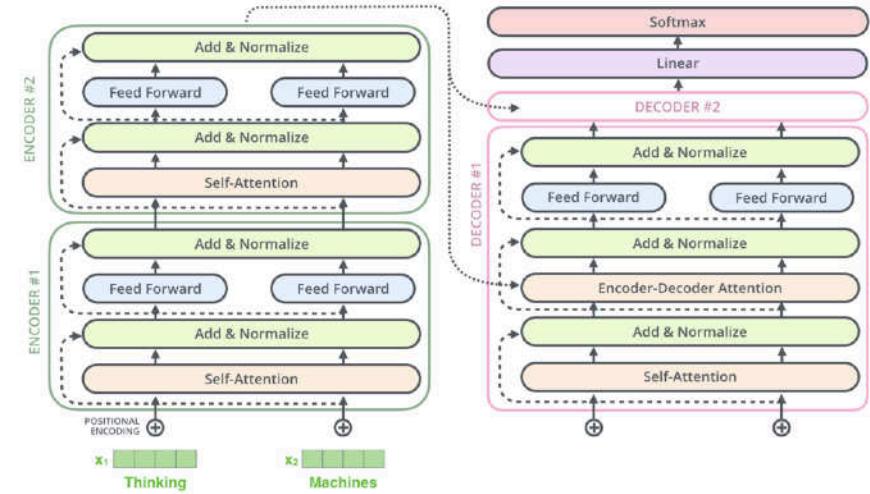
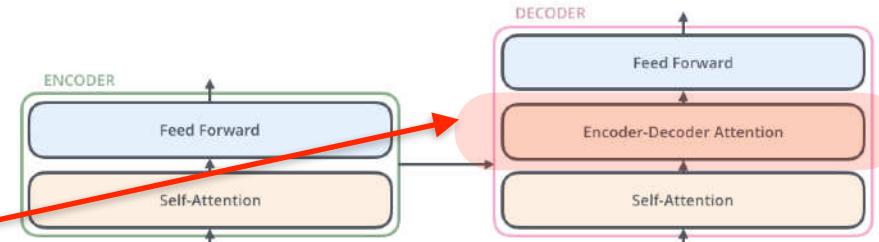
Transformer

(3) Decoder block

One decoder block

– Three sub-layers

- (1) Self-Attention (as the encoder)
 - but auto-regressive (masking)
 - (2) Encoder-Decoder Attention
 - helps the decoder focus on relevant parts of the input sentence
 - similar to what attention does in Seq2Seq models
 - (3) Feed Forward layers (as the encoder)
- Each sub-layer has
- a **residual** connection around it
 - a **layer-normalisation** step



Transformer

(3) Decoder block

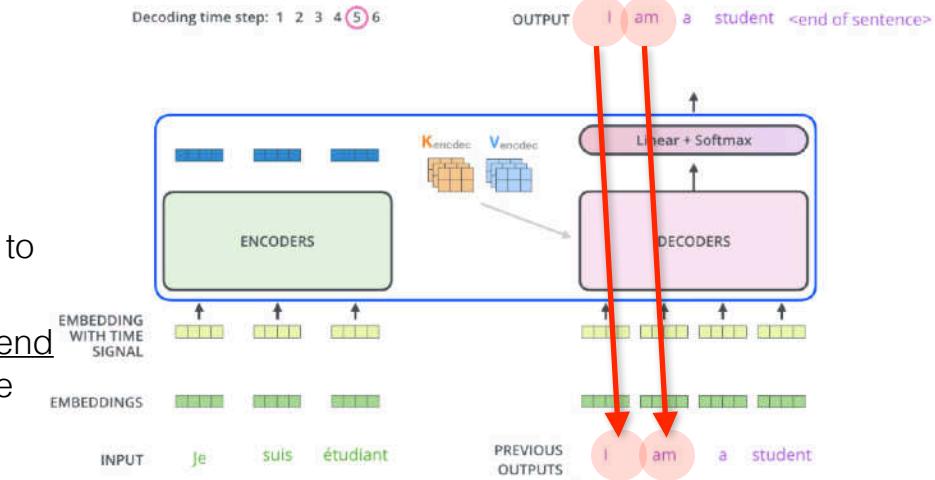
One decoder block

– Decoder Self-Attention:

- is auto-regressive !!!
 - output of the decoder of each step is fed to the bottom in the next step
 - the self-attention layer only allowed to attend to earlier positions in the output sequence
 - masking (see next slide)

– Encoder-Decoder Attention ?

- works just like multi-head self-attention
- helps the decoder focus on appropriate places in the input sequence
- K_{encdec} and V_{encdec} comes from transformed output of the top encoder
 - used by each decoder in its EDA layer
- Queries come from the layer below it
- Embed and add **positional encoding** to the decoder inputs (as for the encoder)



Transformer

(1) Self-Attention

Masked Self-Attention (decoder)

– **Matrix form** (for all time steps i)

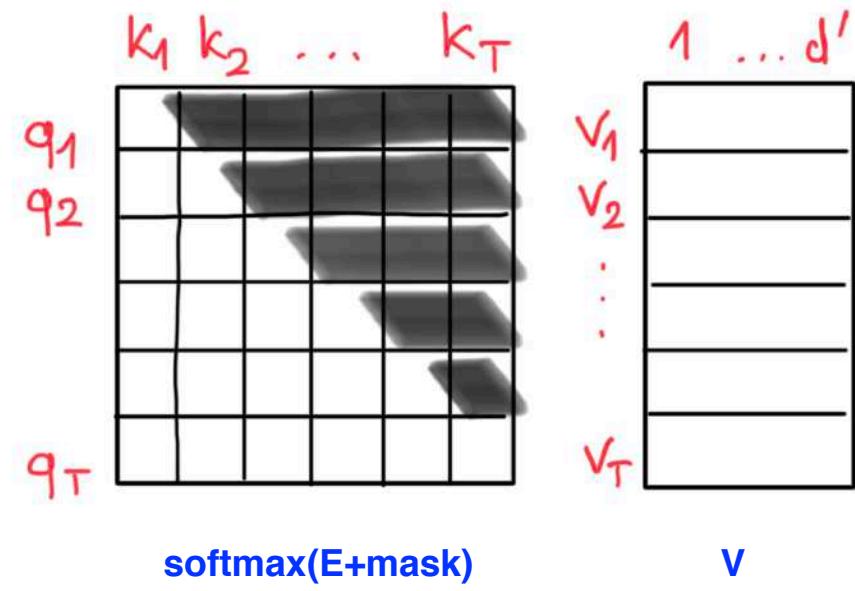
- the decoder is an auto-regressive model
- it cannot see the future
- use masking

$$\mathbf{E} = \frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d'}} \quad [(T_q, d')(d', T_k)](T, d')$$

$$\mathbf{Z} = \text{softmax}(\mathbf{E} + \text{mask}) \mathbf{V}$$

$$\text{mask} = \begin{pmatrix} 0 & -\inf & -\inf & \dots & -\inf \\ 0 & 0 & -\inf & \dots & -\inf \\ 0 & 0 & 0 & \dots & -\inf \\ \vdots & & & & \\ 0 & 0 & 0 & \dots & 0 \end{pmatrix}$$

$$e^{-\inf} = 0$$



Transformer

(4) Positional encoding

- **Problem:**

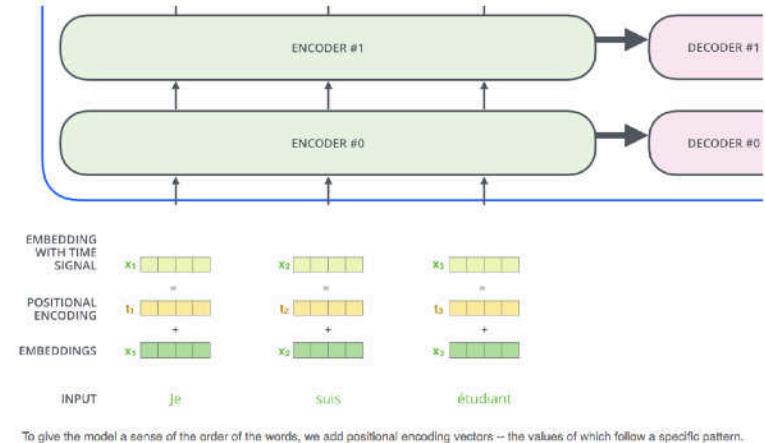
- Transformer doesn't have an inherent sense of word order or position;
- It does not know that \mathbf{x}_i is before or after \mathbf{x}_j

- **Solution:**

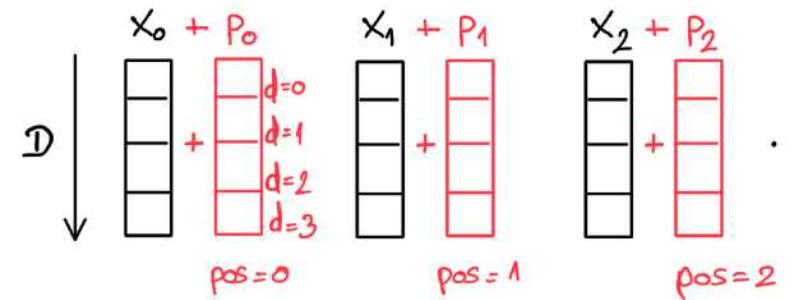
- We encode the order/position of each element of the sequence by **adding** a **Positional Encoding**

- **Which Positional Encoding ?**

- Define
 - $p_0 = [0,0,\dots,0]$, $p_1 = [1,1,\dots,1]$, $p_2 = [2,2,\dots,2]$
 - PE add too much distortion to \mathbf{x}_i !
- Normalise by the length of the sequence
 - $p_0 = [0,0,\dots,0]$, $p_1 = [1/3,\dots,1/3]$, $p_2 = [2/3,\dots,2/3]$
 - PE will change depending on the length !



To give the model a sense of the order of the words, we add positional encoding vectors -- the values of which follow a specific pattern.



Transformer

(4) Positional encoding

Which positional encoding ?

- [Vaswani et al 2017]

$$PE(pos, 2d) = \sin\left(\frac{pos}{10000^{\frac{2d}{D}}}\right)$$

$$PE(pos, 2d + 1) = \cos\left(\frac{pos}{10000^{\frac{2d}{D}}}\right)$$

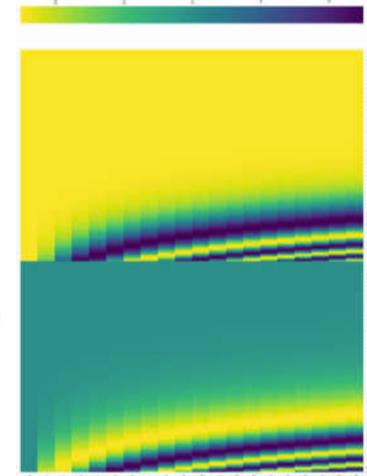
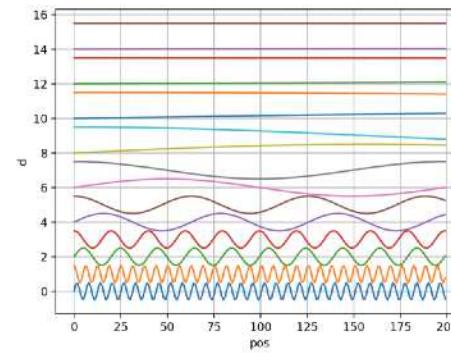
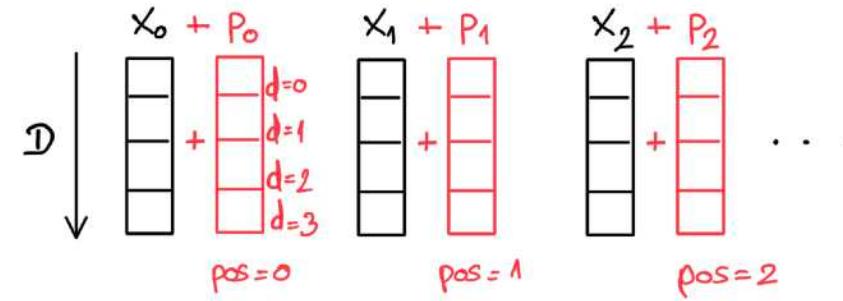
- Example: dimensions $d \in \{0 \dots, D\}$

$$d = 0 \rightarrow 10000^{0/4} = 1 \quad \rightarrow PE(pos,0) = \sin(pos)$$

$$\rightarrow PE(pos,1) = \cos(pos)$$

$$d = 1 \rightarrow 10000^{2/4} = 100 \quad \rightarrow PE(pos,2) = \sin(pos/100)$$

$$\rightarrow PE(pos,3) = \cos(pos/100)$$



```
pos=np.arange(0,200)
D=16
clf()
for d in range(int(D/2)):
    plot(pos, 2*d+0.5*sin(pos/(10000**((2*d/D)))));
    plot(pos, 2*d+1+0.5*cos(pos/(10000**((2*d/D)))));
plt.xlabel('pos'), plt.ylabel('d'); plt.grid(True)
```

Transformer

(4) Positional encoding

Which positional encoding ?

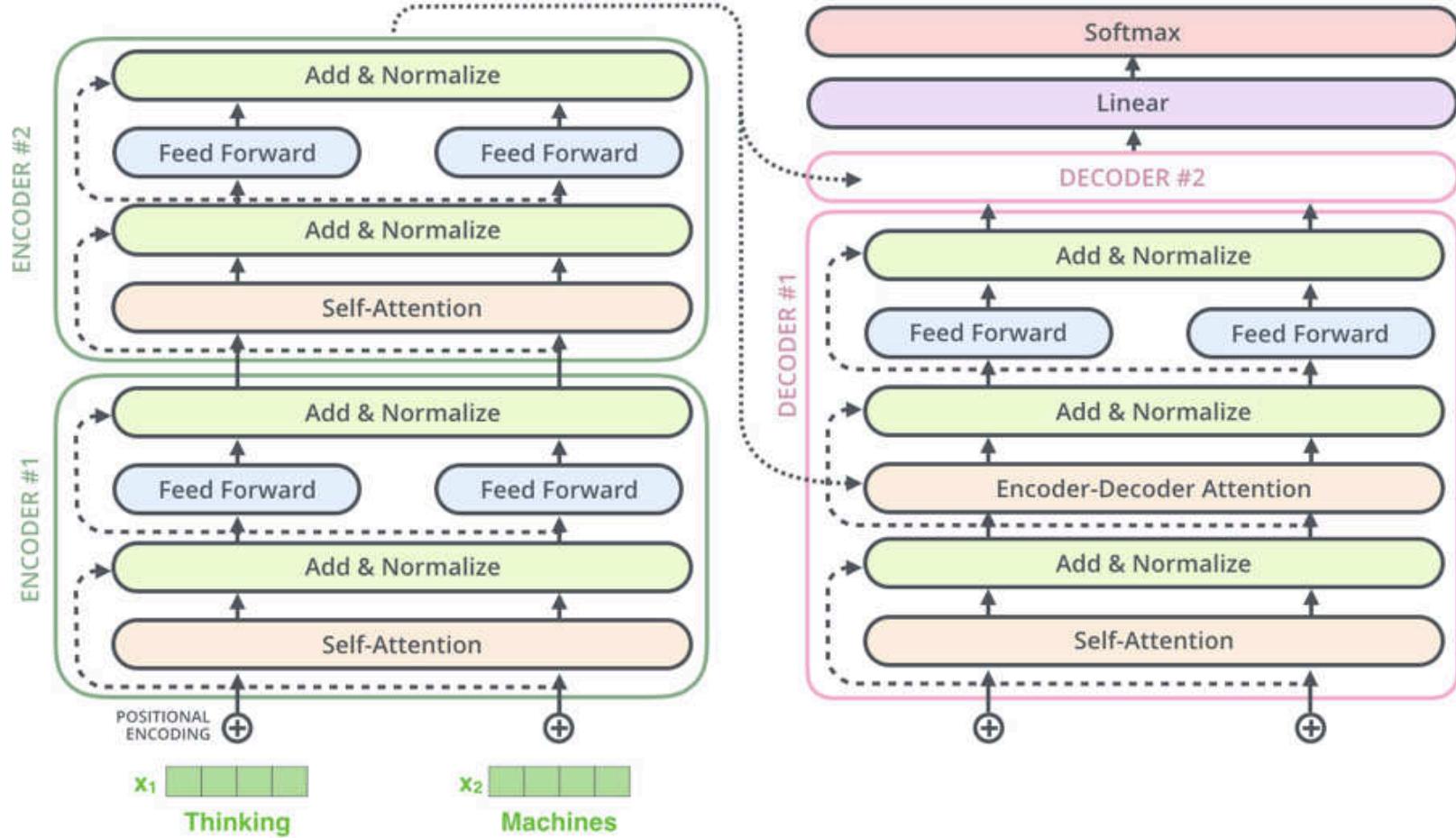
– Absolute positional encoding

$$z_i = \sum_{j=1}^n \alpha_{ij} (x_j W^V)$$
$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^n \exp(e_{ik})}$$
$$e_{ij} = \frac{(x_i W^Q)(x_j W^K)^T}{\sqrt{d}}$$
$$x_i = x_i + p_i$$

– Relative positional encoding

$$z_i = \sum_{j=1}^n \alpha_{ij} (x_j W^V + p_{ij}^V)$$
$$e_{ij} = \frac{(x_i W^Q + p_{ij}^Q)(x_j W^K + p_{ij}^K)^T}{\sqrt{d}}$$

Transformer

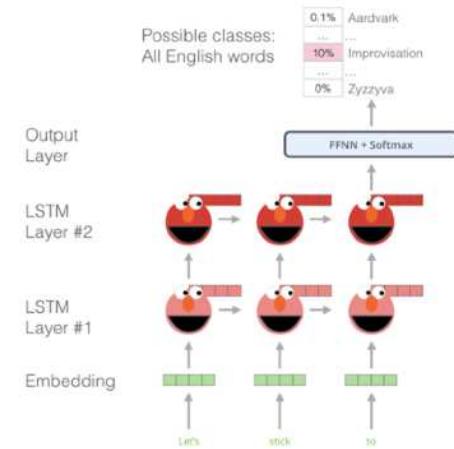


Transformer usage

Transformer usage in NLP

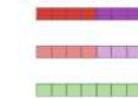
ELMo (Embeddings from Language Models)

- Context Matters:
 - takes into the context when computing the embedding
- How ?
 - bi-directional LSTM
- Contextualised embedding :
 - grouping together the hidden states (and initial embedding) in a certain way (concatenation followed by weighted summation)

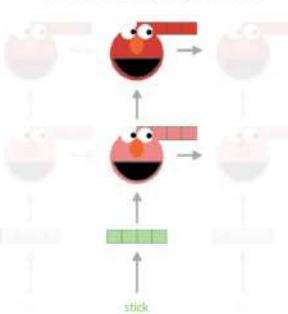


Embedding of "stick" in "Let's stick to" - Step #2

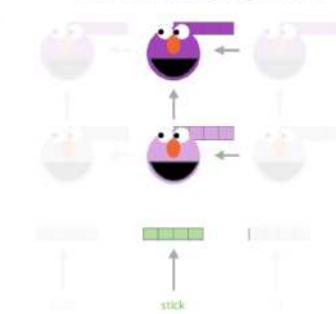
1- Concatenate hidden layers



Forward Language Model



Backward Language Model



2- Multiply each vector by a weight based on the task



3- Sum the (now weighted) vectors



ELMo embedding of "stick" for this task in this context

Transformer usage in NLP

OpenAI GPT

- Use Transformer (instead of LSTM)
- Only use the decoder (predicting the next word) since it's built to mask future tokens
- Pre-training:
 - predict the next word using massive (unlabelled) datasets: 7,000 books
- Transfer-learning (downstream tasks)

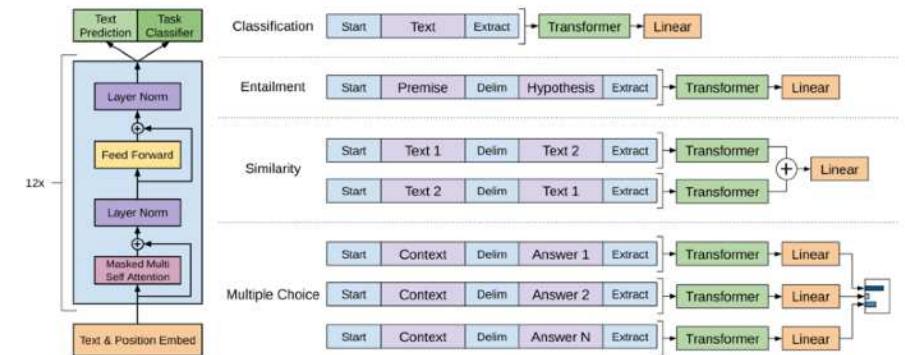
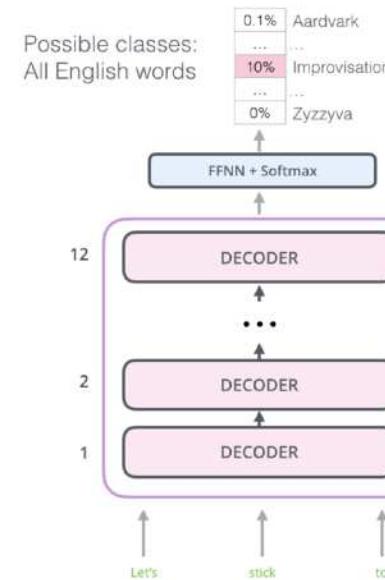
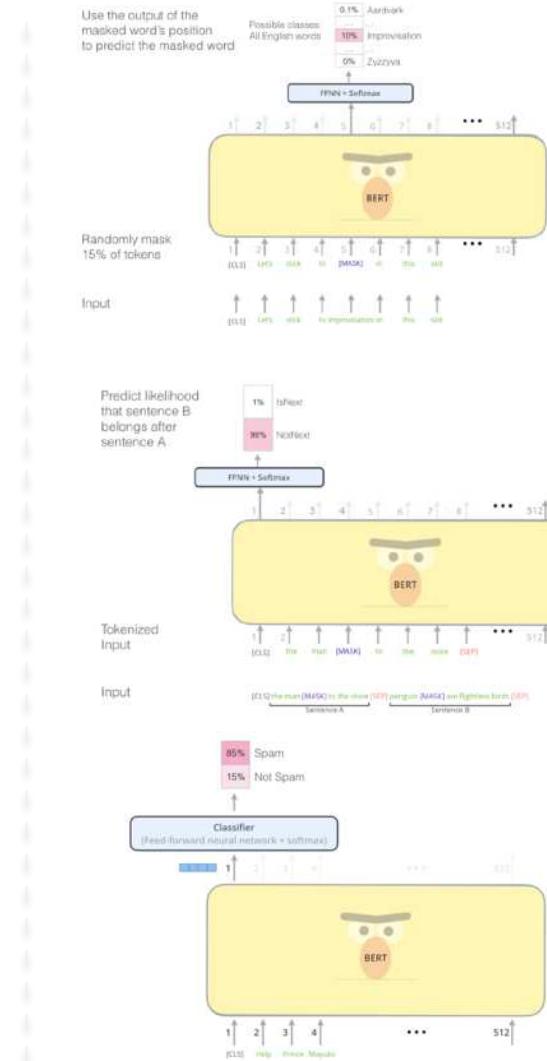


Figure 1: (left) Transformer architecture and training objectives used in this work. (right) Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

Transformer usage in NLP

Google BERT

- OpenAI GPT is Transformer Decoder
 - this is great but Decoder is not bi-directional (LSTM ELMo was !)
- BERT
 - Use the Transformer Encoder
 - the first input token is supplied with a special [CLS] token. CLS here stands for Classification
- Training on pretext tasks:
 - Masked Language Model
 - Two-sentence classification task
- Use directly the CLS output for downstream tasks (Spam, not Spam)

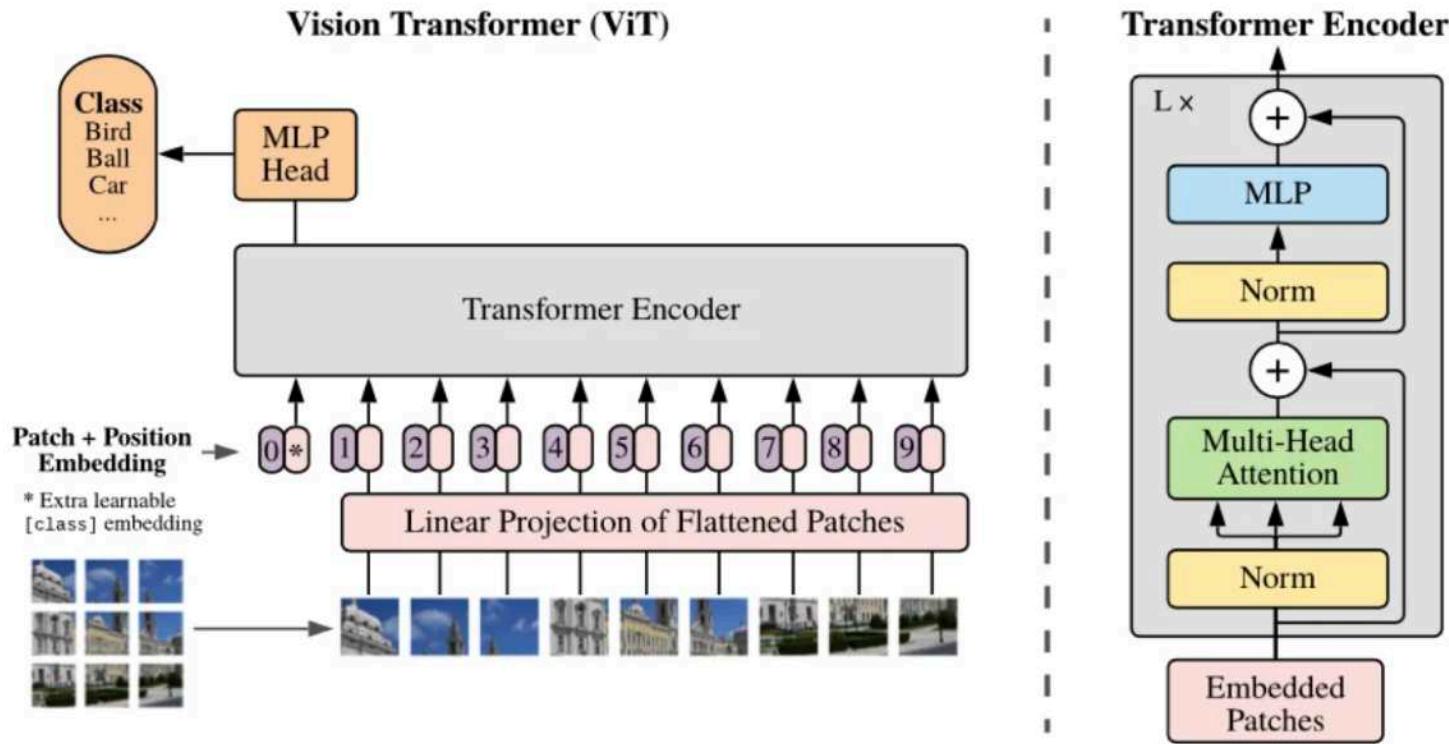


Masked Language Model

Two-sentence
classification task

Transformer usage in Computer Vision

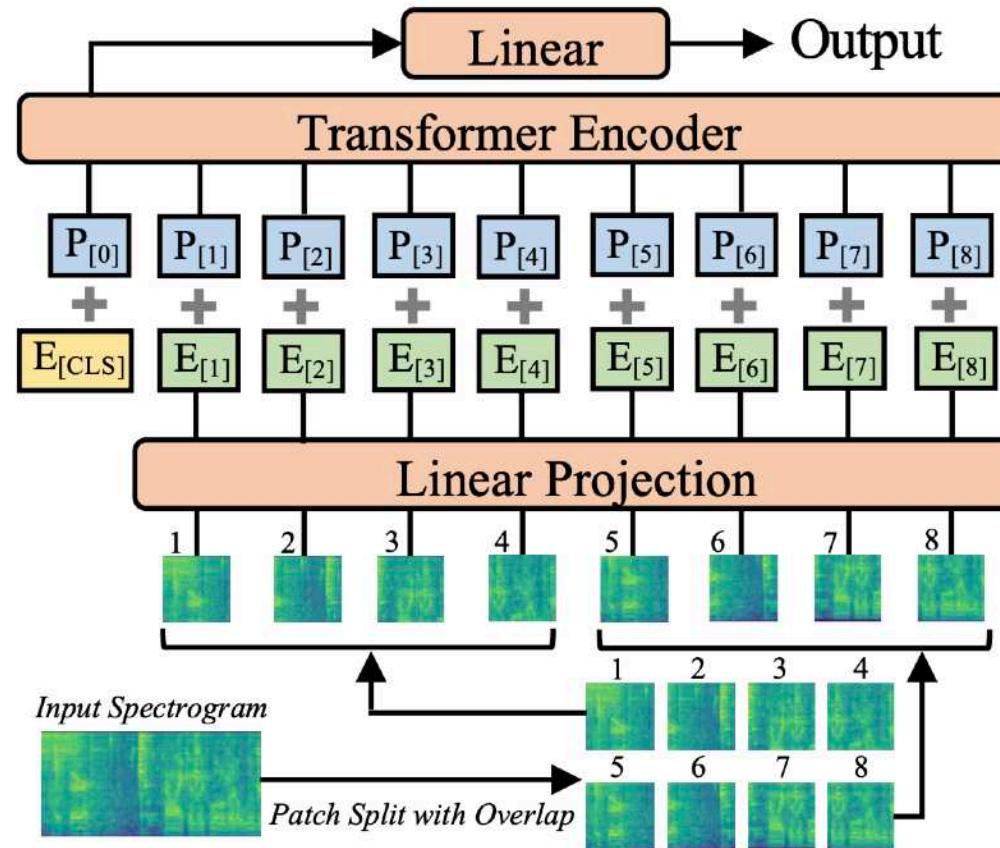
Vision Transformer



Vision Transformer ViT Architecture – [Source](#)

Transformer usage in Audio

Audio Spectrogram Transformer



[Gong et al. "AST: Audio Spectrogram Transformer"] [LINK](#)

Evolutionary Tree

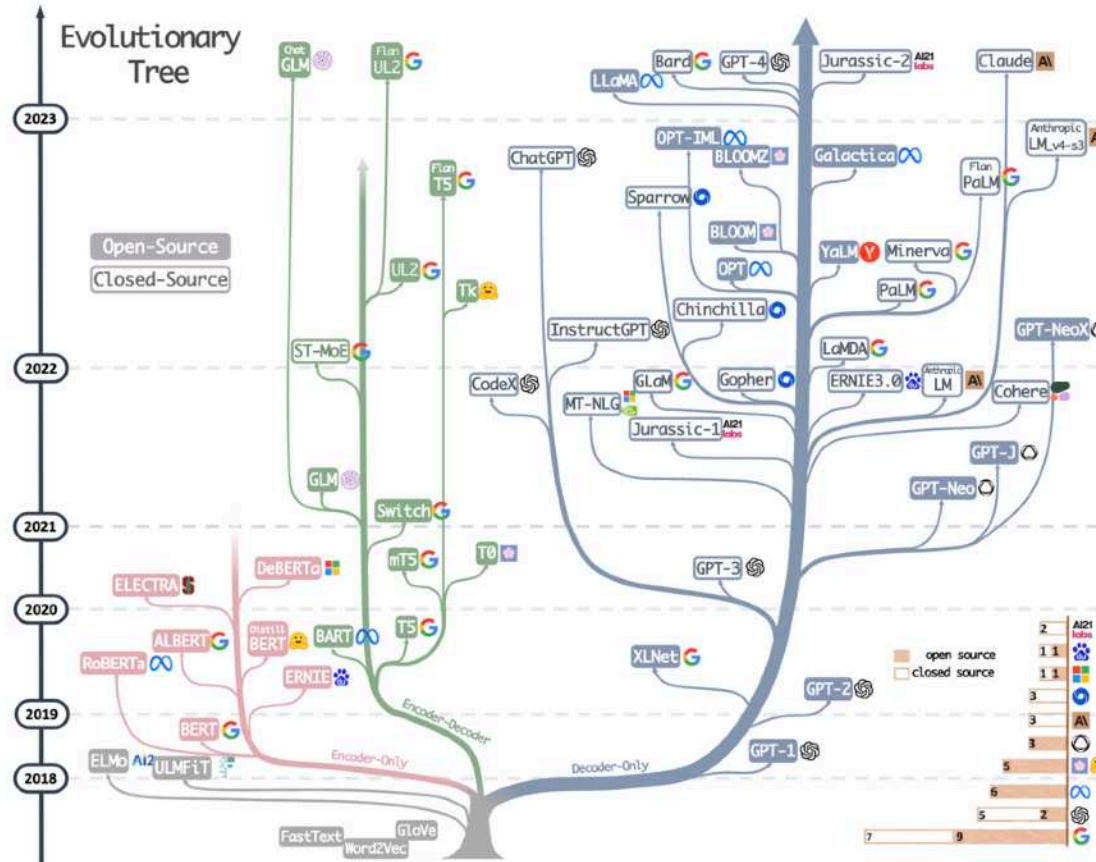


Fig. 1. The evolutionary tree of modern LLMs traces the development of language models in recent years and highlights some of the most well-known models. Models on the same branch have closer relationships. Transformer-based models are shown in non-grey colors: decoder-only models in the blue branch, encoder-only models in the pink branch, and encoder-decoder models in the green branch. The vertical position of the models on the timeline represents their release dates. Open-source models are represented by solid squares, while closed-source models are represented by hollow ones. The stacked bar plot in the bottom right corner shows the number of models from various companies and institutions.

Overview

