UNIVERSITÉ
CÔTE D'AZUR

Advanced Deep Learning

Lecture #9

(Stacked Denoising) AutoEncoder, Recurrent Neural Network , Biases

Frédéric Precioso

UNIVERSITÉ CÔTE D'AZUR

(DENOISING) STACKED AUTOENCODER

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Deep Autoencoders

- A deep Autoencoder is constructed by extending the encoder and decoder of autoencoder with multiple hidden layers.

- Gradient vanishing problem: the gradient becomes too small as it passes back through many layers

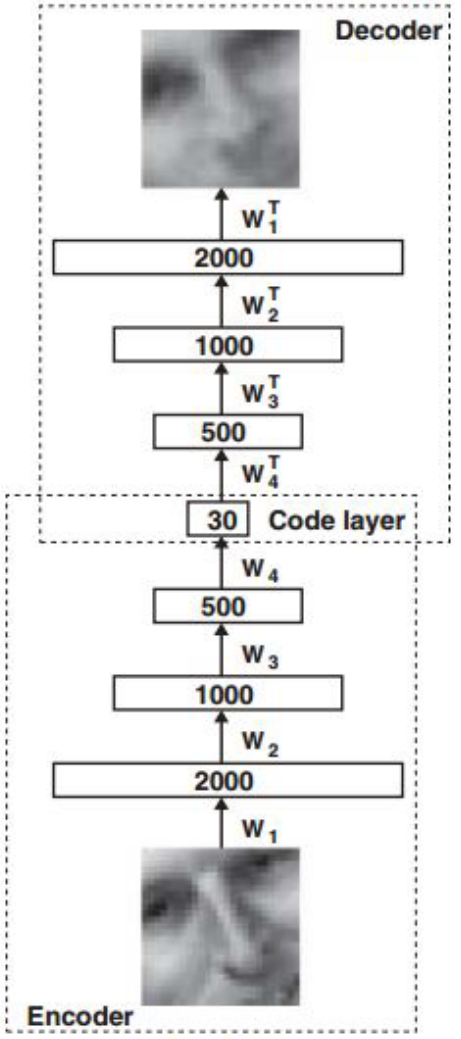


Diagram from (Hinton and Salakhutdinov, 2006)

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Training Deep Autoencoders

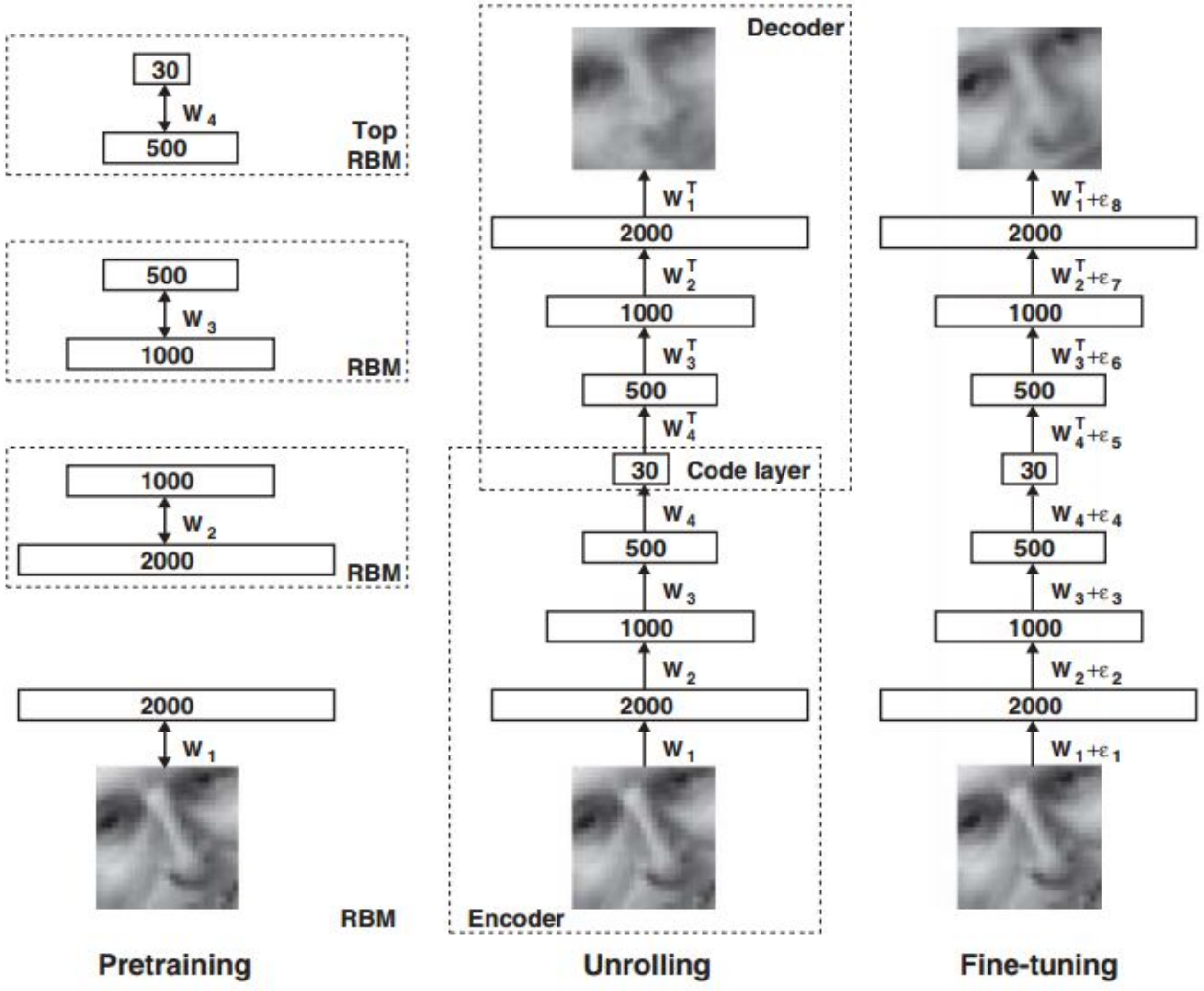
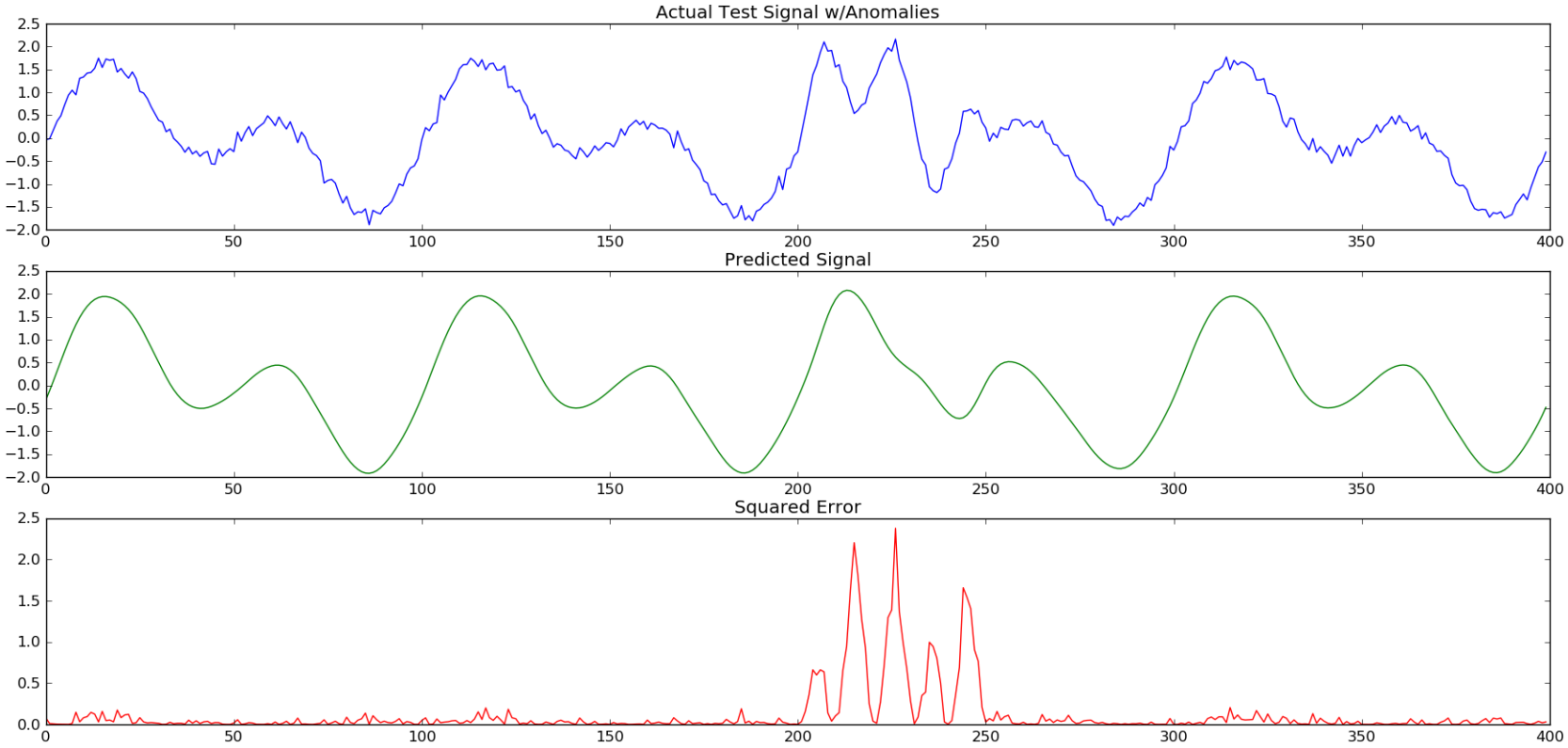


Diagram from (Hinton and Salakhutdinov, 2006)

UNIVERSITÉ CÔTE D'AZUR

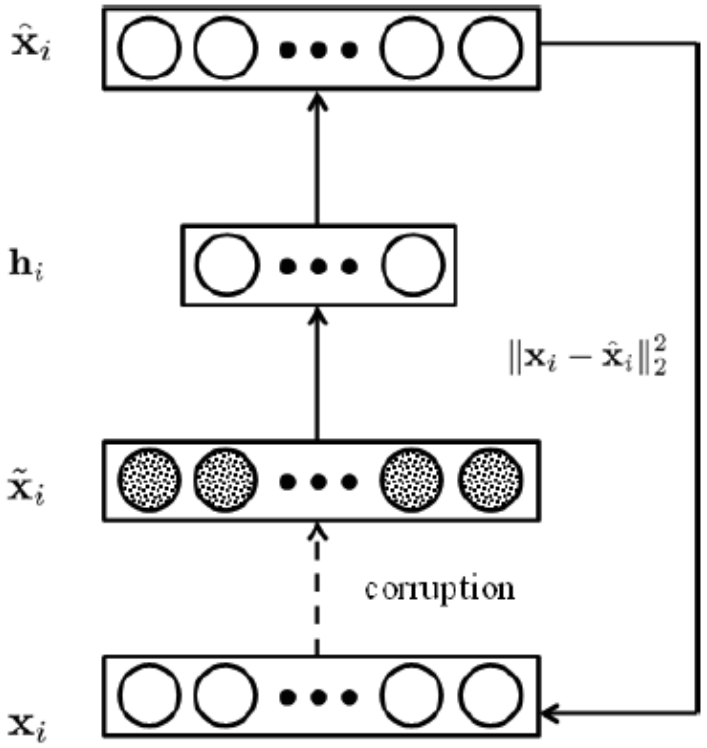
univ-cotedazur.fr

Time Series Anomaly Detection



Denoising Autoencoders

- By adding stochastic noise to the, it can force Autoencoder to learn more robust features.



UNIVERSITÉ CÔTE D'AZUR
univ-cotedazur.fr

Training Denoising Autoencoder

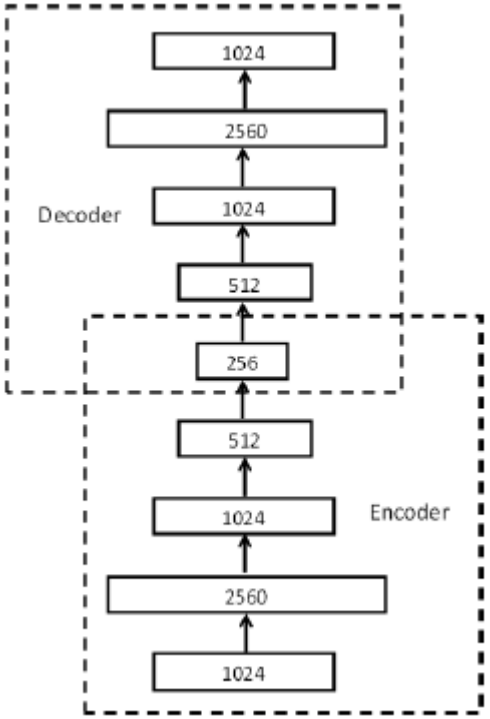
The loss function of Denoising autoencoder:

$$\min_{\mathbf{W}_1,\mathbf{W}_2,\mathbf{b}_1,\mathbf{b}_2} \sum_{\ell} \|\mathbf{x}^{(\ell)} - \hat{\mathbf{x}}^{(\ell)}\|_2^2 + \lambda\left(\|\mathbf{W}_1\|_F^2 + \|\mathbf{W}_2\|_F^2\right)$$

where

$$\mathbf{h}^{(\ell)} = \sigma(\mathbf{W}_1\tilde{\mathbf{x}}^{(\ell)} + \mathbf{b}_1)$$
$$\hat{\mathbf{x}}^{(\ell)} = \sigma(\mathbf{W}_2\mathbf{h}^{(\ell)} + \mathbf{b}_2)$$

Like deep Autoencoder, we can stack multiple denoising autoencoders layer-wisely to form a Stacked Denoising Autoencoder.

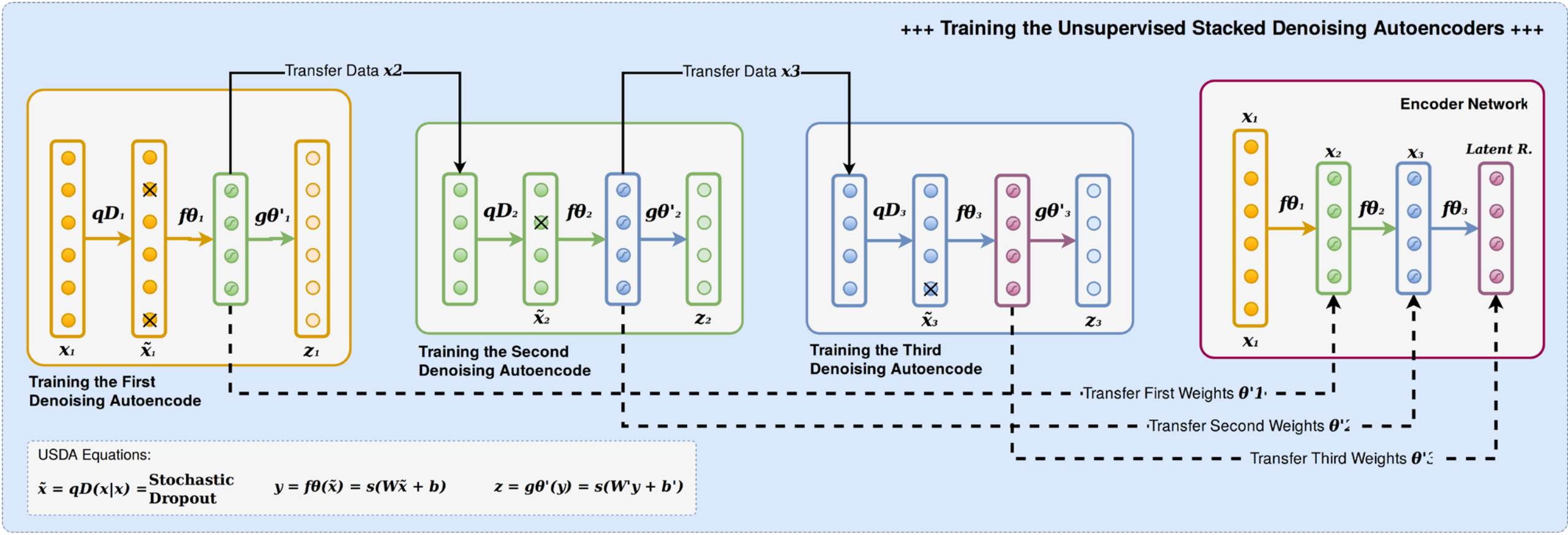


UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Denoising stacked Autoencoder: unsupervised

Result = a new latent representation



UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Denoising stacked Autoencoder: example

Model *Deep Patient*, Published in Nature, 2016

Stage 1.

Data-mining stage & Feature extraction:

Driving Electronic Health Records to build a binary phenotype representation.

Stage 2.

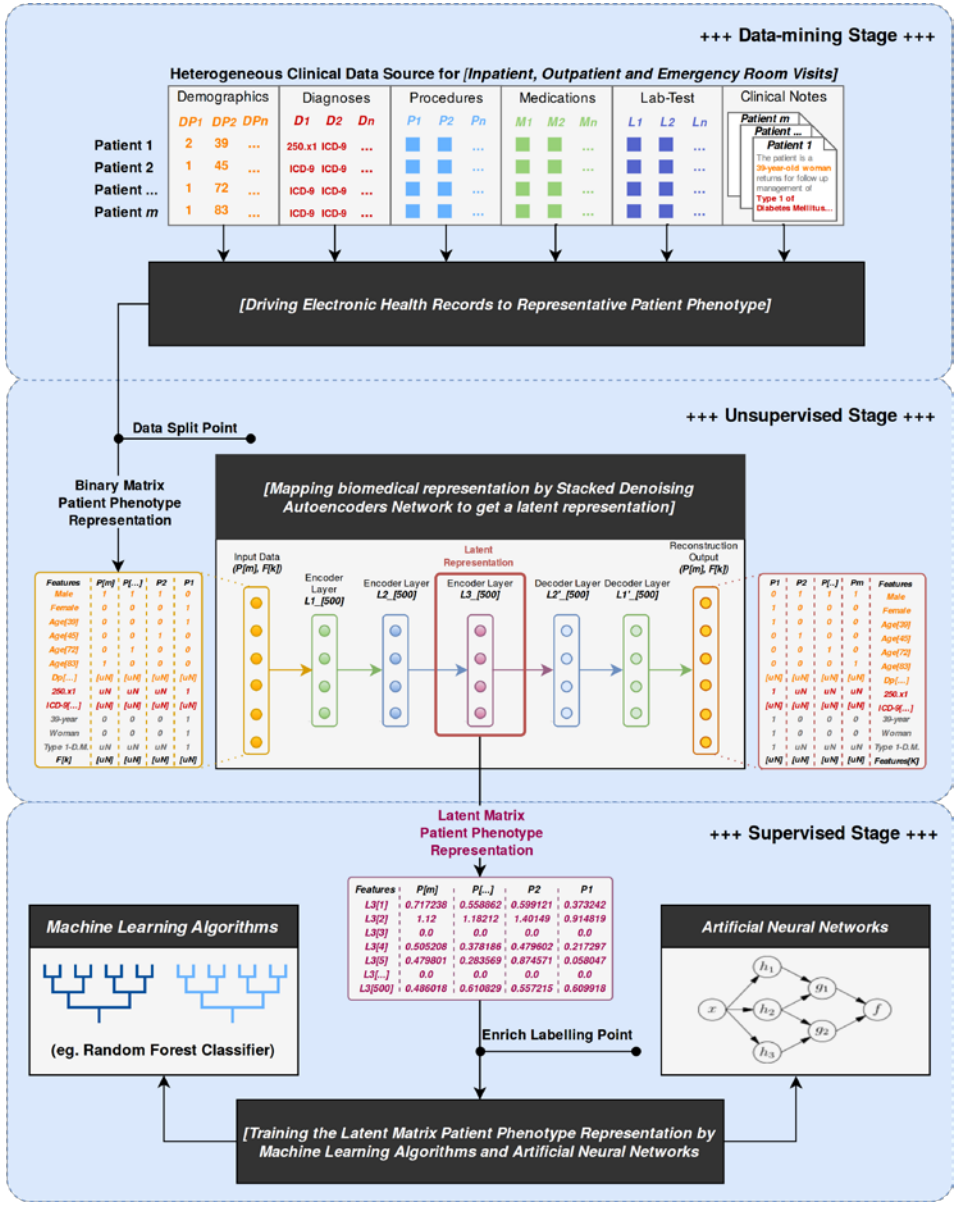
Unsupervised stage:

Mapping the Binary Patient Representation to get a new space call Deep Patient (or Latent Representation) Using Stacked Denoising Autoencoders.

Stage 3.

Supervised stage:

Labeling Medical Target and training the *Latent Representation* by Machine Learning algorithms for classification and prediction of patient's disease.



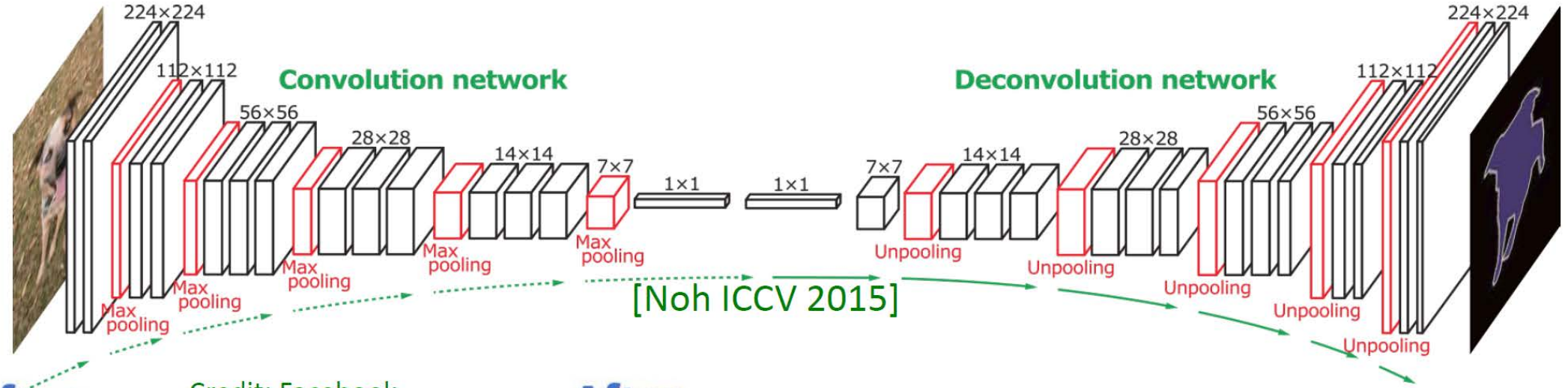
UNIVERSITÉ CÔTE D'AZUR

Combining CNN and Autoencoder

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Image Segmentation



Credit: Facebook

Before

After

MS COCO Detection Challenge!

Credits Matthieu Cord

Significantly borrowed from COLAH's Blog
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Completed with "Introduction to Deep Learning", by Professor Qiang Yang, from The Hong Kong University of Science and Technology

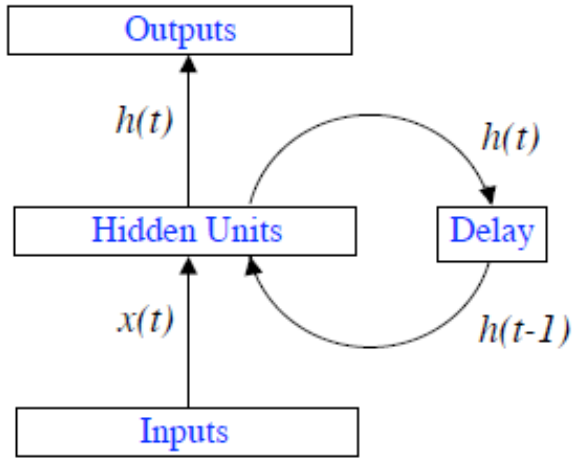
RECURRENT NEURAL NETWORKS (LSTM, GRU)

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

What are RNNs?

Recurrent neural networks (RNNs) are connectionist models with the ability to selectively pass information across sequence steps, while processing sequential data one element at a time.



The simplest form of ***fully recurrent neural network*** is an MLP with the previous set of hidden unit activations feeding back into the network along with the inputs

Allow a 'memory' of previous inputs to persist in the network's internal state, and thereby influence the network output

$$h(t) = f_H(W_{IH}x(t) + W_{HH}h(t-1))$$

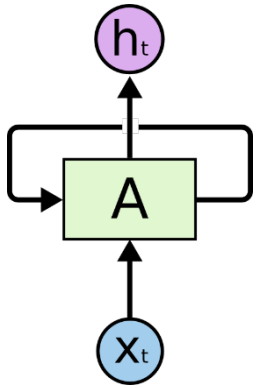
$$y(t) = f_O(W_{HO}h(t))$$

f_H and f_O are the activation function for hidden and output unit; W_{IH}, W_{HH}, and W_{HO} are connection weight matrices which are learnt by training

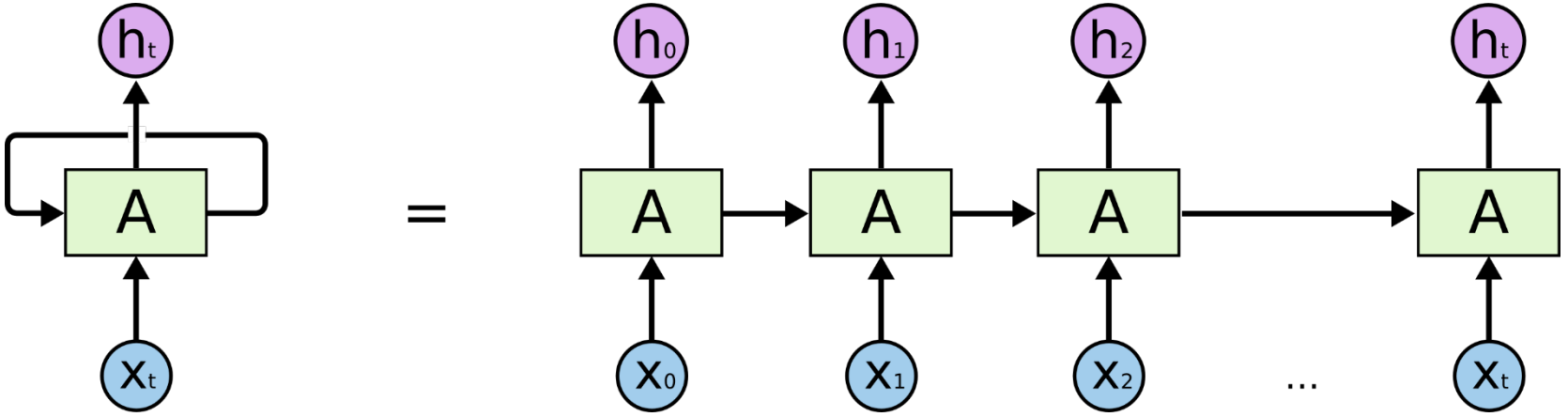
UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Recurrent Neural Networks have loops.



An unrolled recurrent neural network.



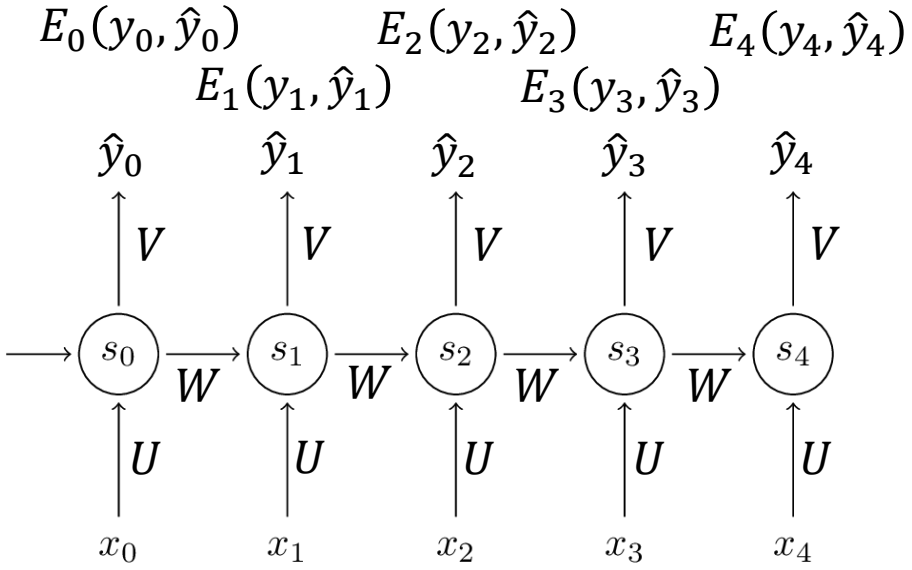
In the last few years, there have been incredible success applying RNNs to a variety of problems: speech recognition, language modeling, translation, image captioning…

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

What are RNNs?

- The recurrent network can be converted into a feed-forward network by ***unfolding over time***



An unfolded recurrent network. Each node represents a layer of network units at a single time step.
The weighted connections from the input layer to hidden layer are labelled 'W_{IH}', those from the hidden layer to itself (i.e. the recurrent weights) are labelled 'W_{HH}' and the hidden to output weights are labelled 'W_{HO}'.
Note that the same weights are reused at every time step. Bias weights are omitted for clarity.

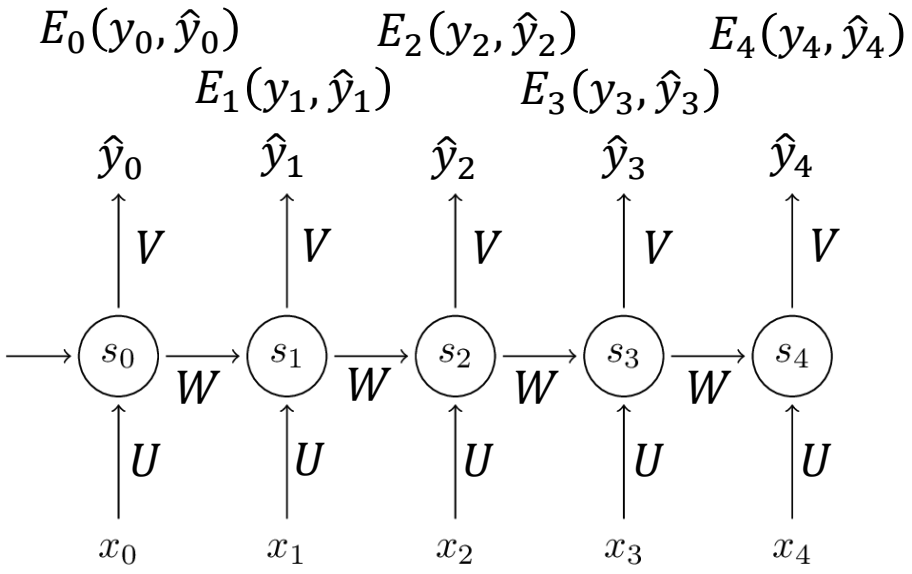
UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

What are RNNs?

- Training RNNs (determine the parameters)

  Back Propagation Through Time (BPTT) is often used to learn the RNN
  BPTT is an extension of the back-propagation (BP)



- The output of this RNN is $\hat{y}_t$

$$s_t = \tanh(Ux_t + Ws_{t-1})$$
$$\hat{y}_t = softmax(Vs_t)$$

- The loss/error function of this network is

$$E_t(y_t, \hat{y}_t) = -y_t \log \hat{y}_t$$ → The error at each time step

$$E(y, \hat{y}) = \sum_t E_t(y_t, \hat{y}_t)$$ → the total loss is the sum of the errors at each time step

UNIVERSITÉ CÔTE D'AZUR

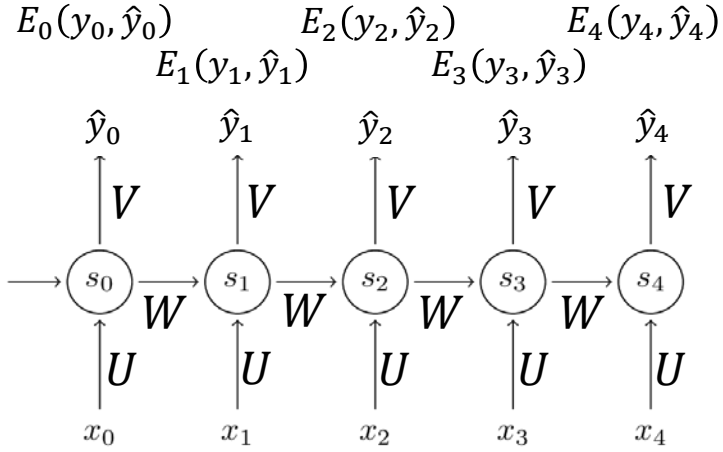
univ-cotedazur.fr

What are RNNs?

- Training RNNs (determine the parameters)
  - ✓ The gradients of the error with respect to our parameters

Just like we sum up the errors, we also sum up the gradients at each time step for one training example. For parameter W, the gradient is

$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W}$$



  - ✓ The gradient at each time step

we use time 3 as an example

$$\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_3}\frac{\partial s_3}{\partial W}$$ → Chain Rule

$$s_3 = \tanh(Ux_3 + Ws_2)$$ → s_3 depends on W and s_2, we cannot simply consider s_2 a constant

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_3}\frac{\partial s_3}{\partial s_k}\frac{\partial s_k}{\partial W}$$ → Apply Chain Rule again on s_k

UNIVERSITÉ CÔTE D'AZUR

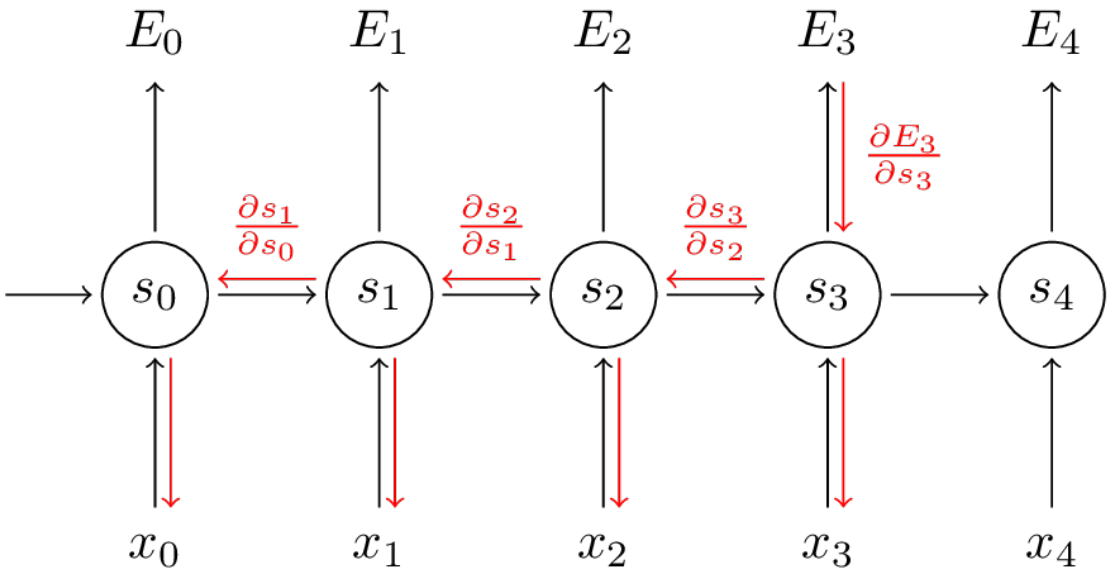
univ-cotedazur.fr

What are RNNs?

- Training RNNs (determine the parameters)

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial s_k} \frac{\partial s_k}{\partial W} \longrightarrow \frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \prod_{j=k+1}^{3} \frac{\partial s_j}{\partial s_{j-1}} \frac{\partial s_k}{\partial W}$$

Because W is used in every step up to the output we care about, we need to backpropagate gradients from $t = 3$ through the network all the way to $t = 0$



UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

What are RNNs?

- The vanishing gradient problem

To understand why, let's take a closer look at the gradient we calculated above:

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \underbrace{\prod_{j=k+1}^{3} \frac{\partial s_j}{\partial s_{j-1}}} \frac{\partial s_k}{\partial W}$$

Because the layers and time steps of deep neural networks relate to each other through multiplication, derivatives are susceptible to vanishing

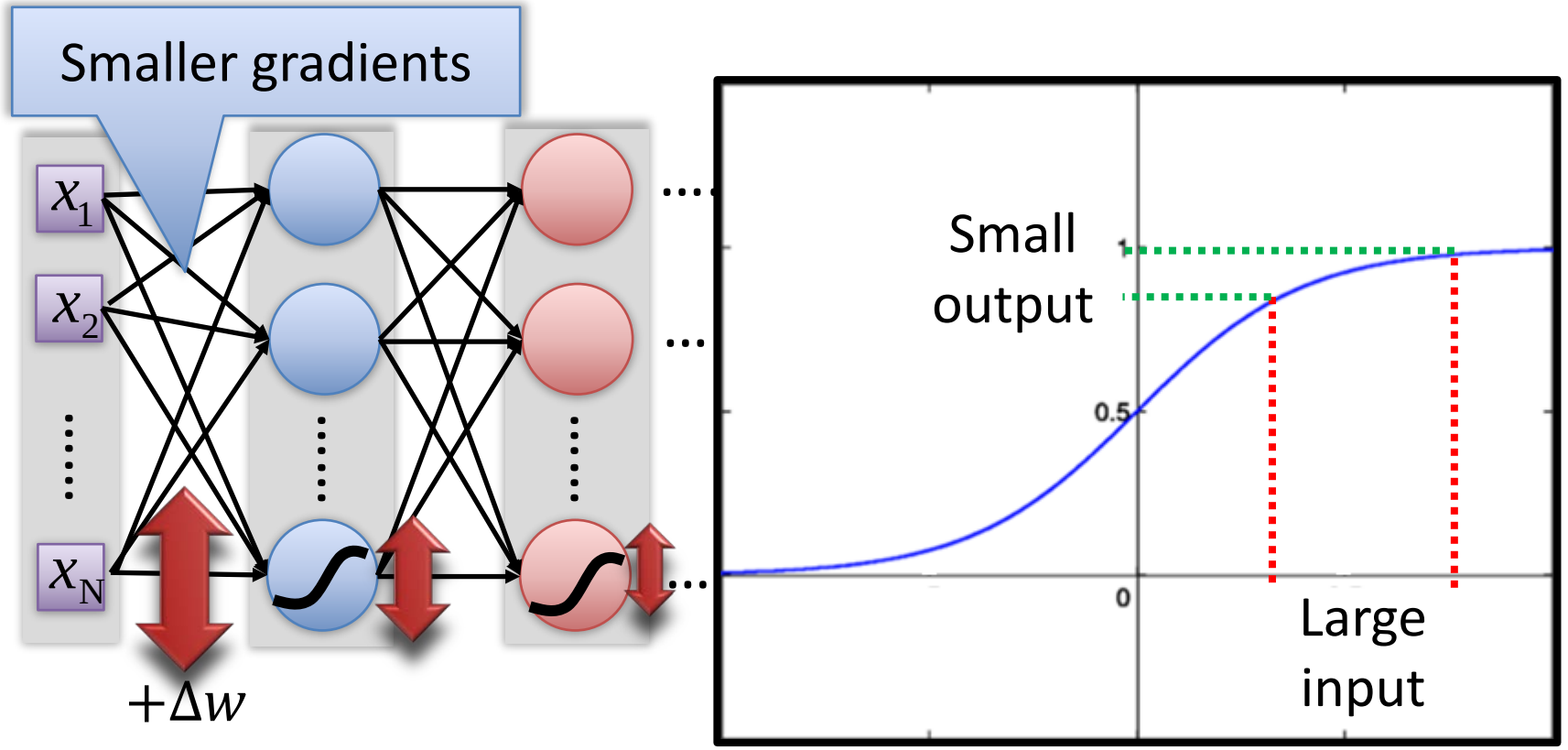
Gradient contributions from "far away" steps become zero, and the state at those steps **does not contribute** to what you are learning: You end up not learning long-range dependencies.

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Recipe of Deep Learning

- ***Vanishing Gradient Problem***



Intuitive way to compute the derivatives …

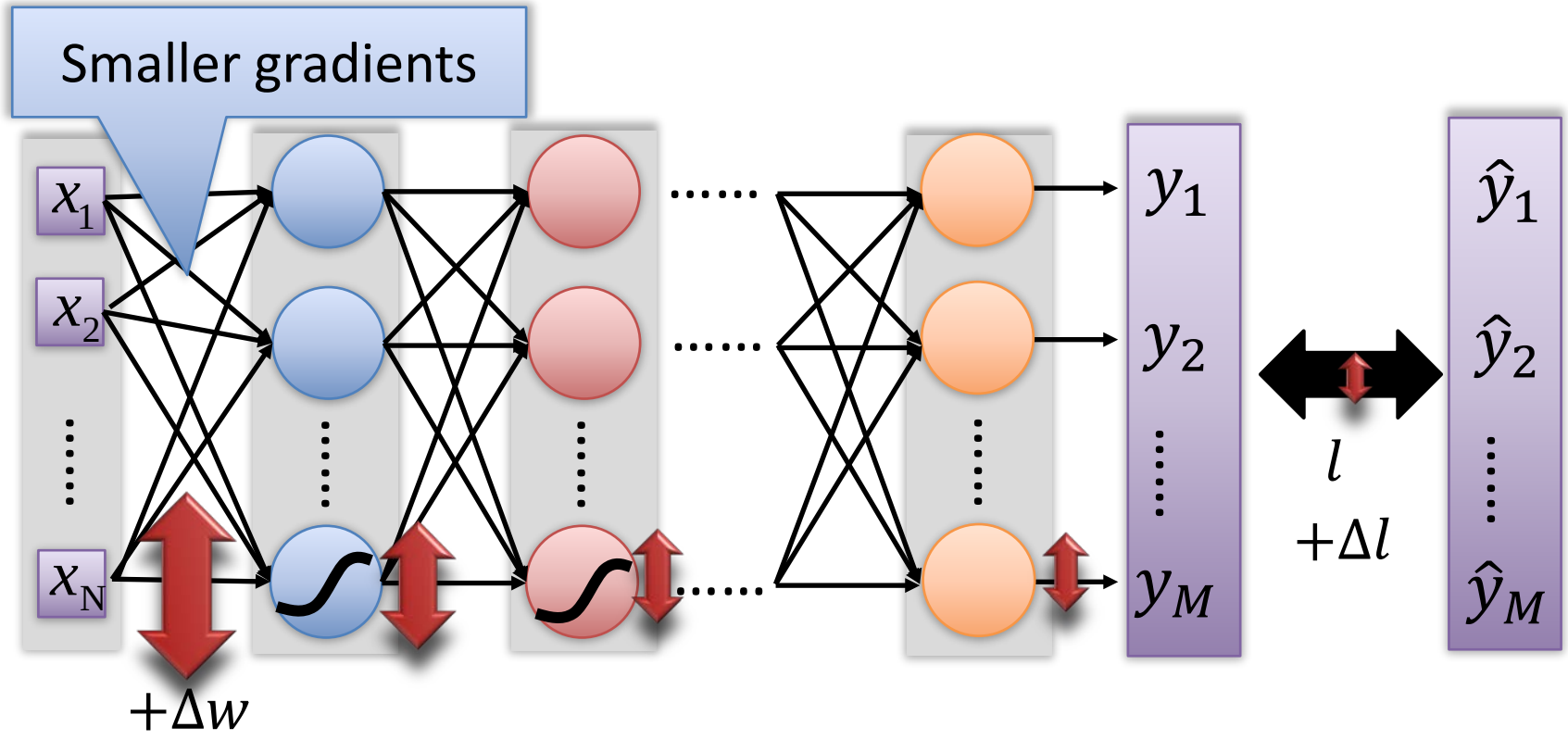
$$\frac{\partial l}{\partial w} =? \frac{\Delta l}{\Delta w}$$

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Recipe of Deep Learning

- ***Vanishing Gradient Problem***



Intuitive way to compute the derivatives …

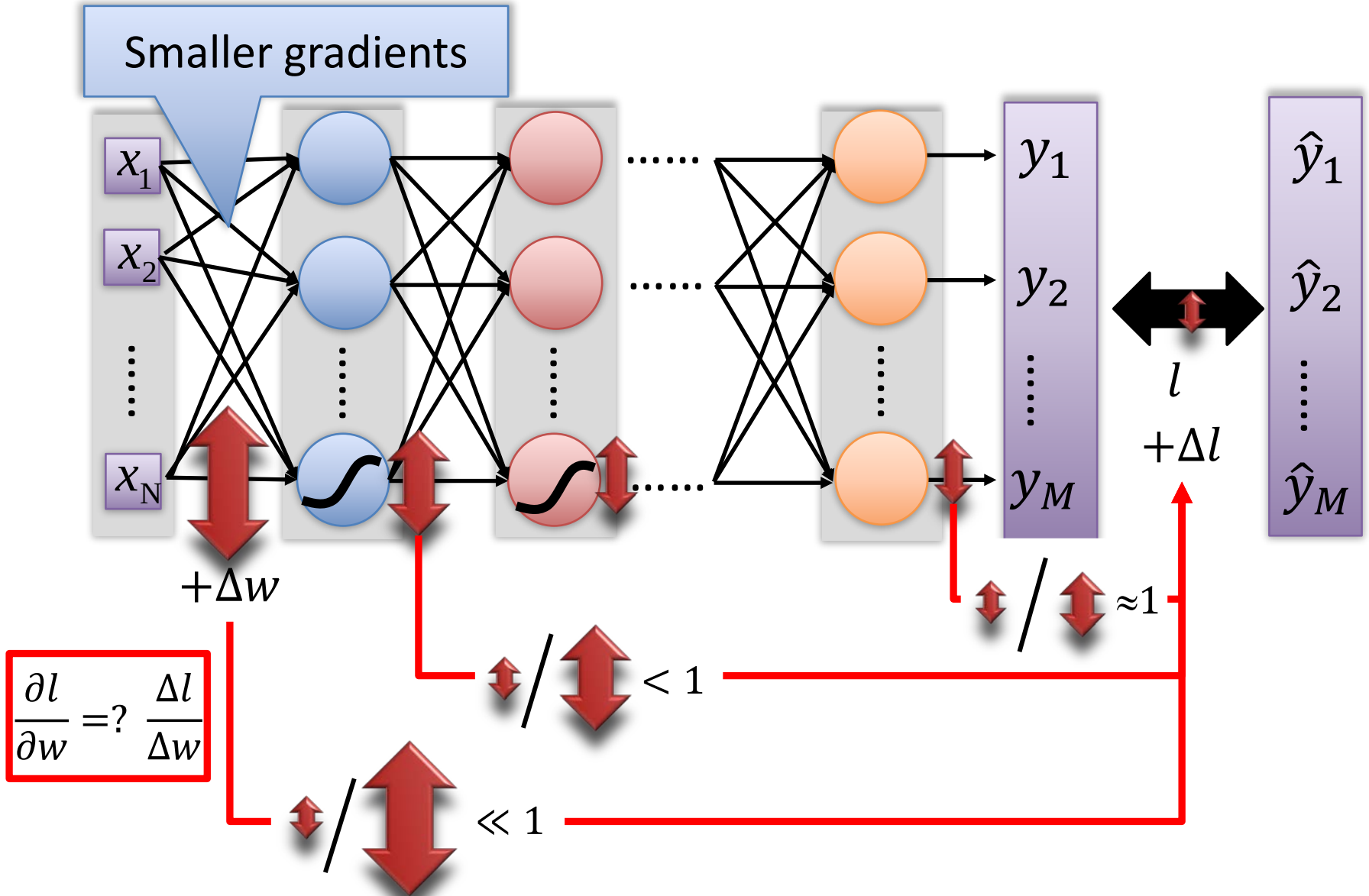
$$\frac{\partial l}{\partial w} =? \frac{\Delta l}{\Delta w}$$

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Recipe of Deep Learning

- ***Vanishing Gradient Problem***



UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

What are RNNs?

- The vanishing gradient problem

To understand why, let's take a closer look at the gradient we calculated above:

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \underbrace{\prod_{j=k+1}^{3} \frac{\partial s_j}{\partial s_{j-1}}} \frac{\partial s_k}{\partial W}$$

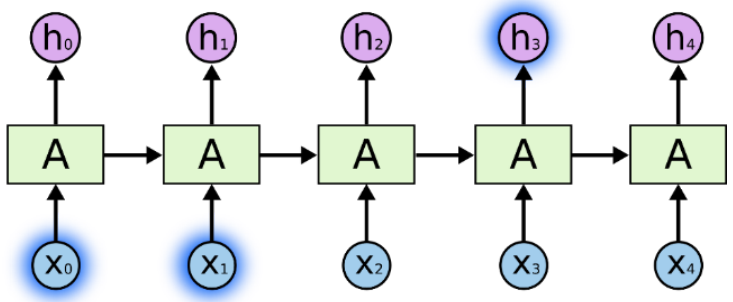
Because the layers and time steps of deep neural networks relate to each other through multiplication, derivatives are susceptible to vanishing

Gradient contributions from "far away" steps become zero, and the state at those steps **does not contribute** to what you are learning: You end up not learning long-range dependencies.

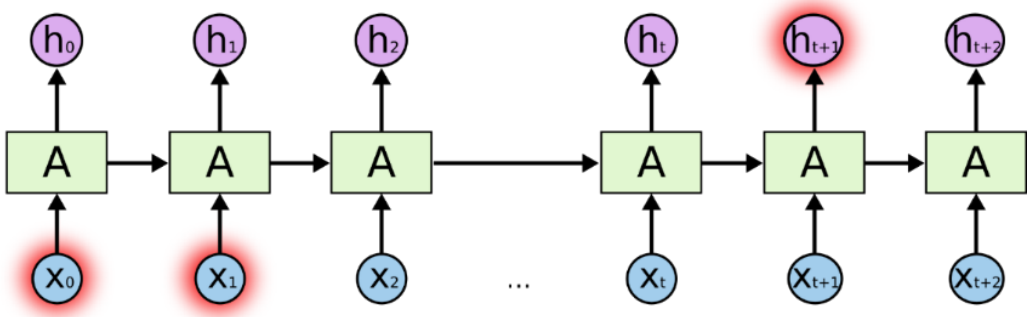
UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

The Problem of Long-Term Dependencies



*If we are trying to predict the last word in "the clouds are in the **(sky)**," we don't need any further context – it's pretty obvious the next word is going to be sky.*

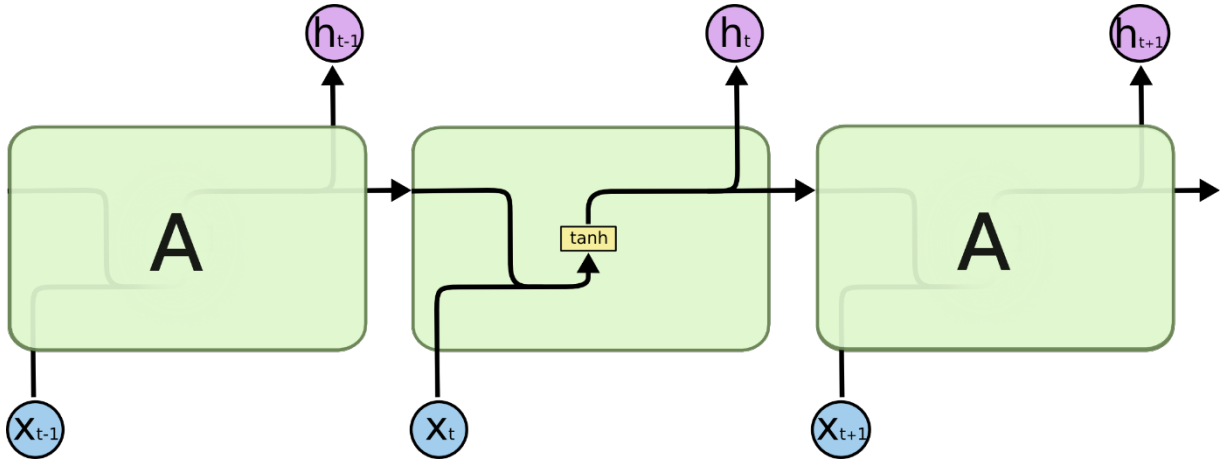


*Consider trying to predict the last word in the text "I grew up in France… I speak fluent **(French)**." Recent information suggests that the next word is probably the name of a language, but if we want to narrow down which language, we need the context of France, from further back.*

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

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



Unfortunately, as that gap grows, RNNs become unable to learn to connect the information (cf. vanishing gradients)

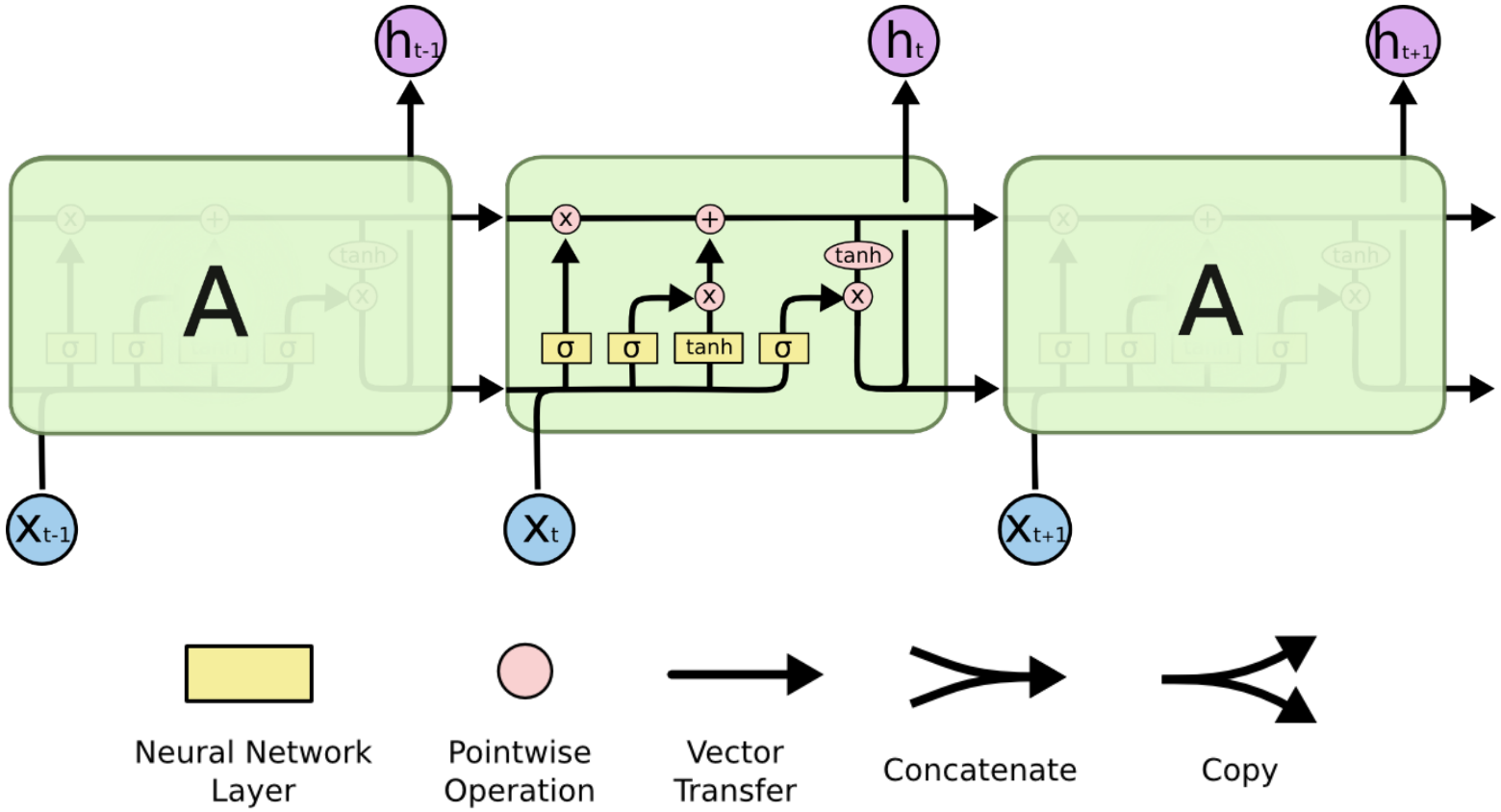
The problem was explored in depth by Hochreiter (1991) and Bengio, et al. (1994), who found some pretty fundamental reasons why it might be difficult.

Thankfully, LSTMs do not have this problem! (Hochreiter & Schmidhuber, 1997)

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

The repeating module in an LSTM contains four interacting layers.

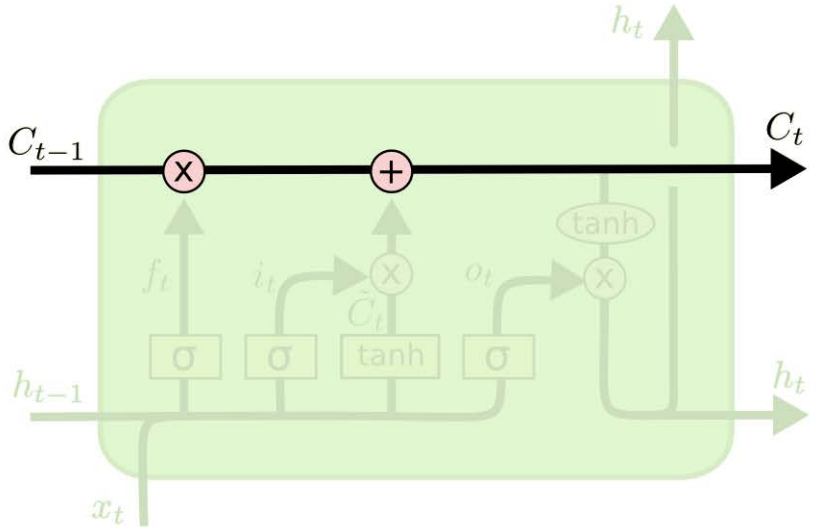


UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

The Core Idea Behind LSTMs

The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It's very easy for information to just flow along it unchanged.

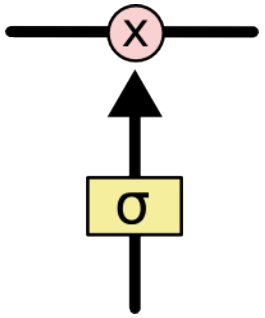


The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates.

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Gates are a way to optionally let information through.

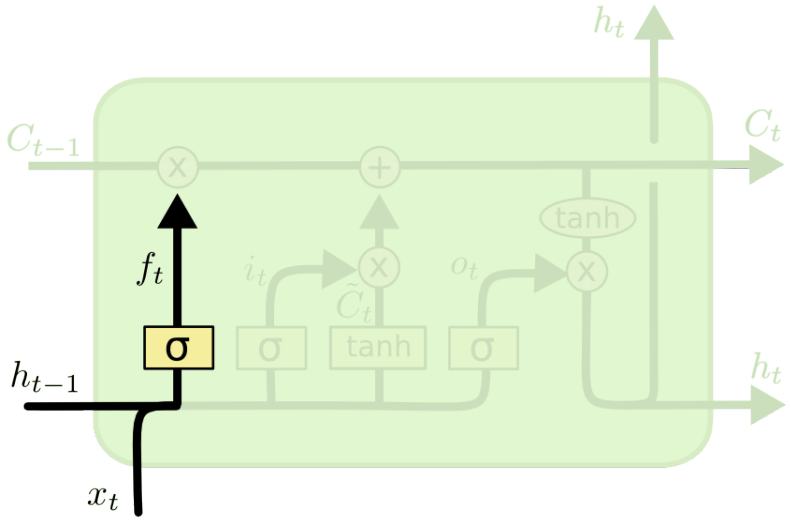


The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through.

An LSTM has three of these gates, to protect and control the cell state.

Step-by-Step LSTM Walk Through: *forget gate layer*

It looks at h_{t-1} and x_t, and outputs a number between 0 and 1 for each number in the cell state C_{t-1}. A 1 represents "completely keep this" while a 0 represents "completely get rid of this."



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

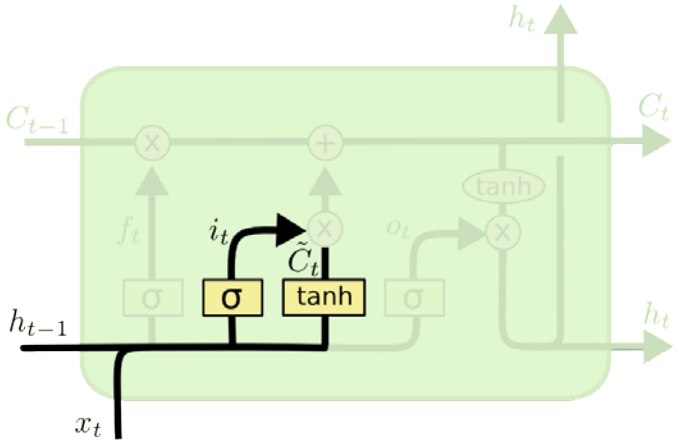
Let's go back to our example of a language model trying to predict the next word based on all the previous ones. In such a problem, the cell state might include the gender of the present subject, so that the correct pronouns can be used.
When we see a new subject, we want to forget the gender of the old subject.

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Step-by-Step LSTM Walk Through: *input gate layer*

The next step is to decide what new information we are going to store in the cell state.
This has two parts.
First, a *sigmoid* layer called the *"input gate layer"* decides which values we will update.
Next, a *tanh* layer creates a vector of new candidate values, $\tilde{C}_t$, that could be added to the state.
In the next step, we will combine these two to create an update to the state.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

In the example of our language model, we would want to add the gender of the new subject to the cell state, to replace the old one we are forgetting.

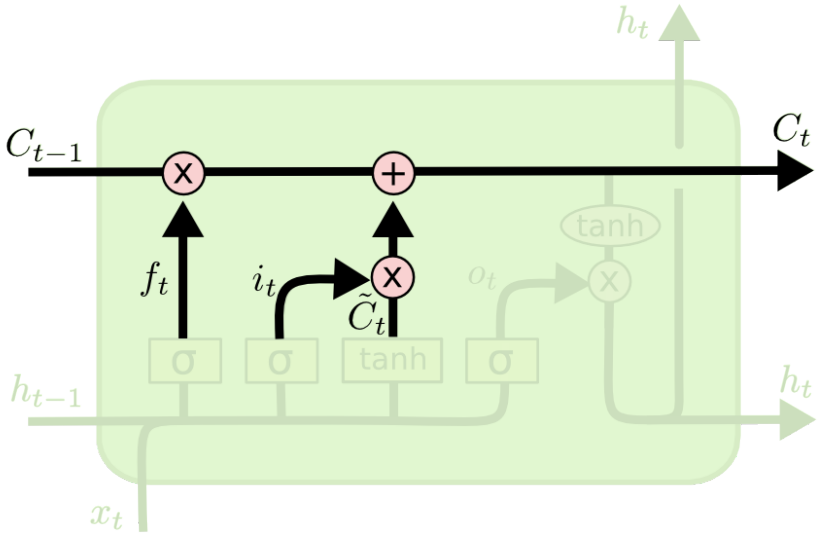
UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Step-by-Step LSTM Walk Through: *update gate layer*

It's now time to update the old cell state, C_{t-1}, into the new cell state C_t.

The previous steps already decided what to do, we just need to actually do it.
We multiply the old state by f_t, forgetting the things we decided to forget earlier. Then we add $i_t * \tilde{C}_t$.
This is the new candidate values, scaled by how much we decided to update each state value.

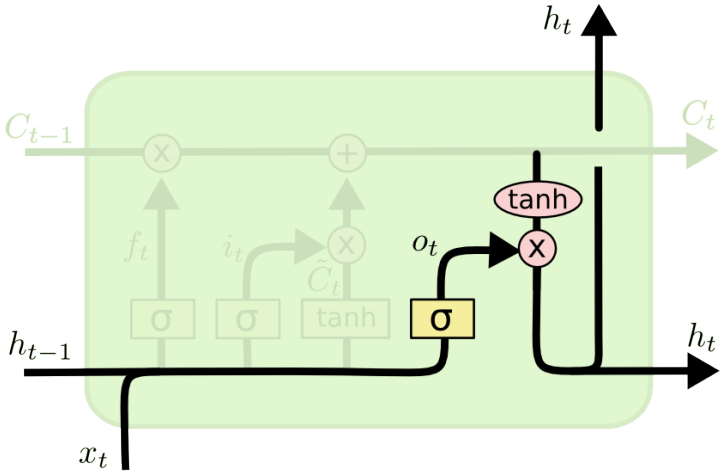


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

In the case of the language model, this is where we would actually drop the information about the old subject's gender and add the new information, as we decided in the previous steps.

Step-by-Step LSTM Walk Through

The output will be based on the cell state, but will be a filtered version.
First, we run a sigmoid layer which decides what parts of the cell state we are going to output.
Then, we put the cell state through *tanh* (to push the values to be between −1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

For the language model example, since it just saw a subject, it might want to output information relevant to a verb, in case that's what is coming next. For example, it might output whether the subject is singular or plural, so that we know what form a verb should be conjugated into if that's what follows next.

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Long Short-term Memory

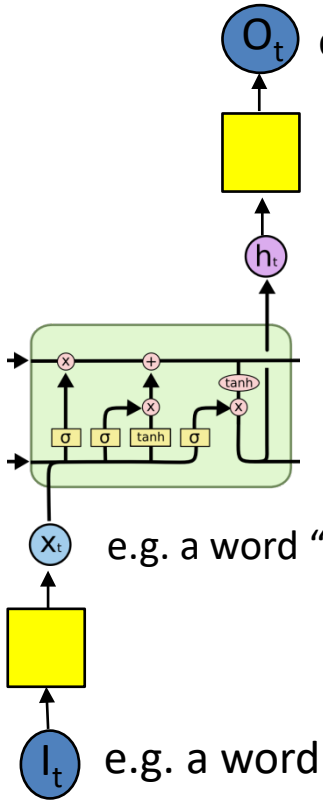
Why LSTM can combat the vanish gradient problem?

LSTMs help preserve the error that can be backpropagated through time and layers, by the *conveyor belt*.
By maintaining a more constant error, they allow recurrent nets to continue to learn over many time steps (over 1000), thereby opening a channel to link causes and effects remotely

UNIVERSITÉ CÔTE D'AZUR
univ-cotedazur.fr

Overall Network Architecture

- Single or multilayer networks can compute LSTM inputs from problem inputs and problem outputs from LSTM outputs.



e.g. a POS tag as a "one hot" vector

e.g. a word "embedding" with reduced dimensionality

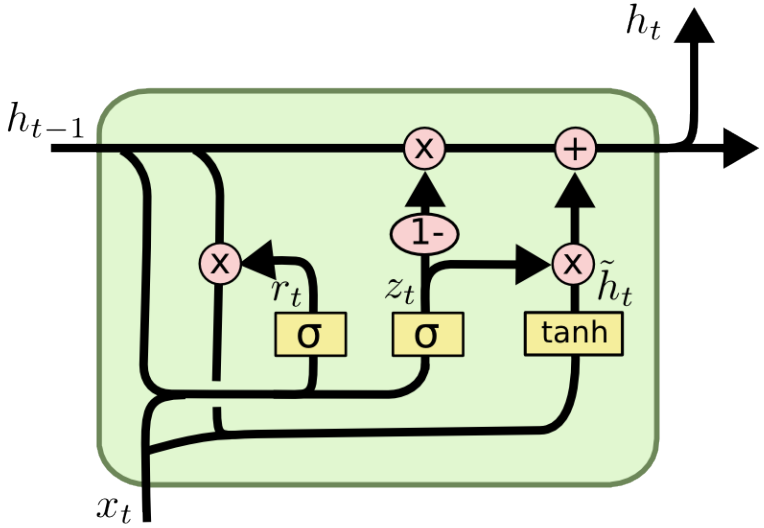
e.g. a word as a "one hot" vector

LSTM Training

- Trainable with backprop derivatives such as:
  - Stochastic gradient descent (randomize order of examples in each epoch) with momentum (bias weight changes to continue in same direction as last update).
  - ADAM optimizer (Kingma & Ma, 2015)
- Each cell has many parameters (W_f, W_i, W_C, W_o)
  - Generally, requires lots of training data.
  - Requires lots of compute time that exploits GPU clusters.

Gated Recurrent Unit (GRU)

- Alternative RNN to LSTM that uses fewer gates (Cho, et al., 2014)
  - Combines forget and input gates into "update" gate.
  - Eliminates cell state vector



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

GRU vs. LSTM

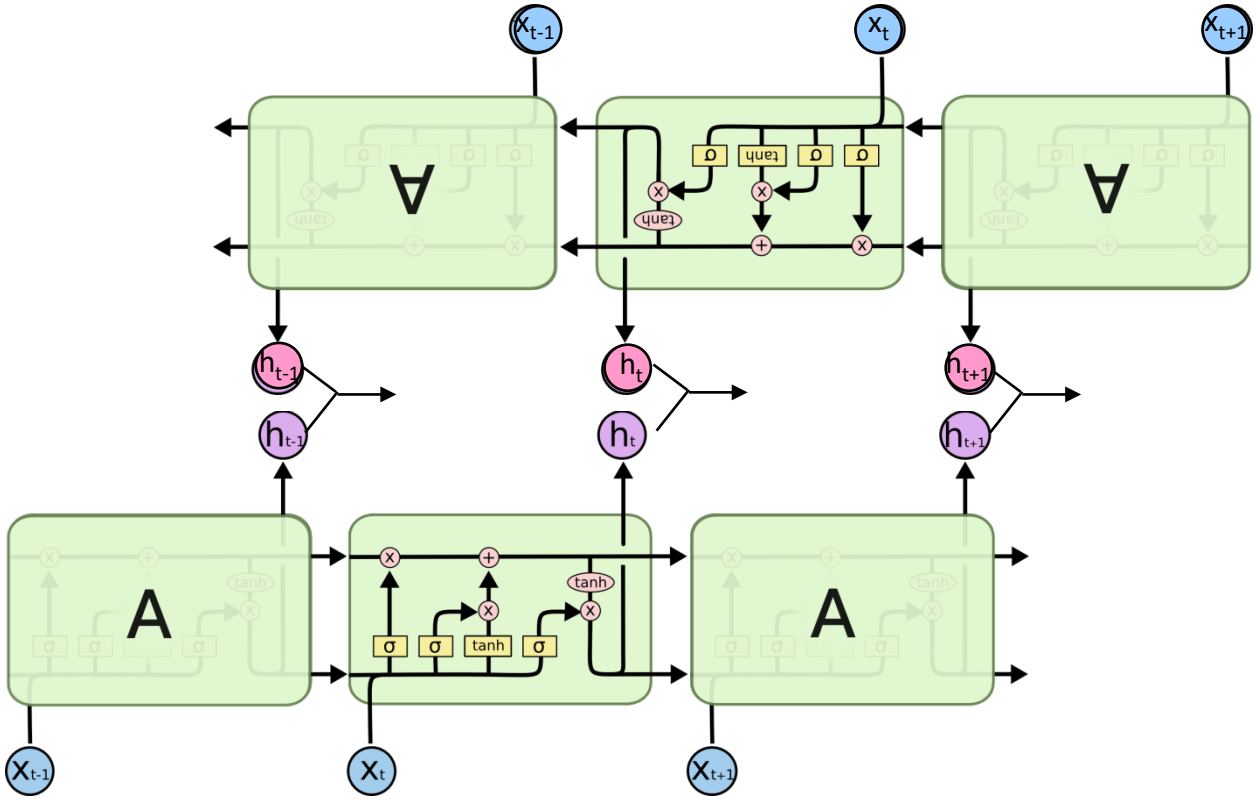
- GRU has significantly fewer parameters and trains faster.
- Experimental results comparing the two are still inconclusive, many problems they perform the same, but each has problems on which they work better.

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Bi-directional LSTM (Bi-LSTM)

- Separate LSTMs process sequence forward and backward and hidden layers at each time step are concatenated to form the cell output.



UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Conclusions

- By adding “gates” to an RNN, we can prevent the vanishing/exploding gradient problem.
- Trained LSTMs/GRUs can retain state information longer and handle long-distance dependencies.

UNIVERSITÉ CÔTE D'AZUR

RNN examples

RNN Examples

- One to Many
- Many to One
- Many to Many
- Parallel Many to Many

-> RNNs can be easily adapted to model different sequential structure

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Summary of LSTM Application Architectures

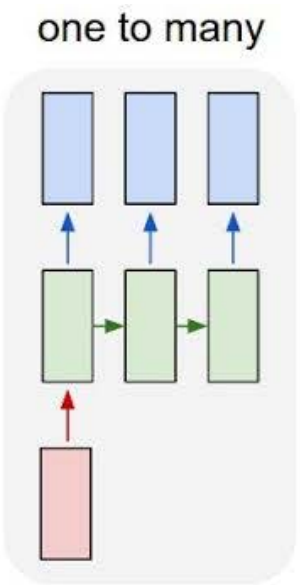
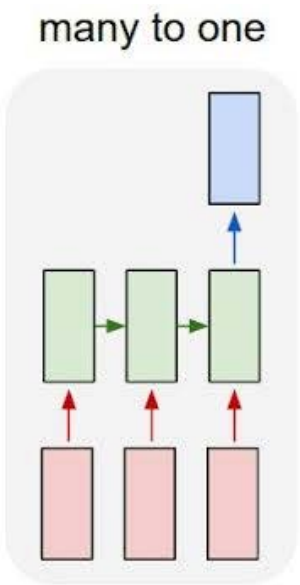
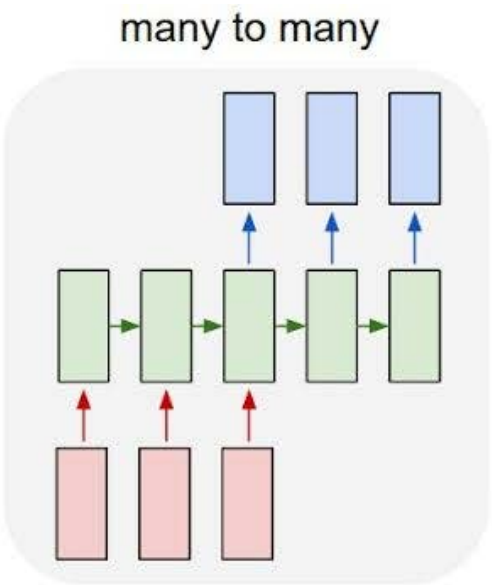


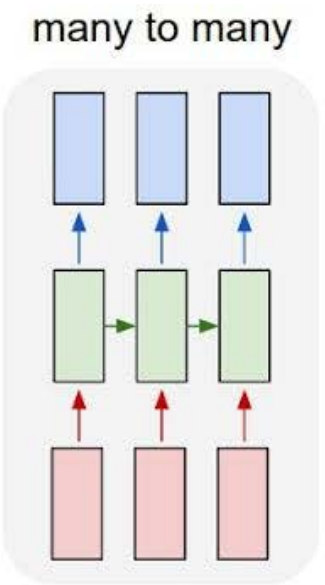
Image Captioning



Video Activity Recog
Text Classification



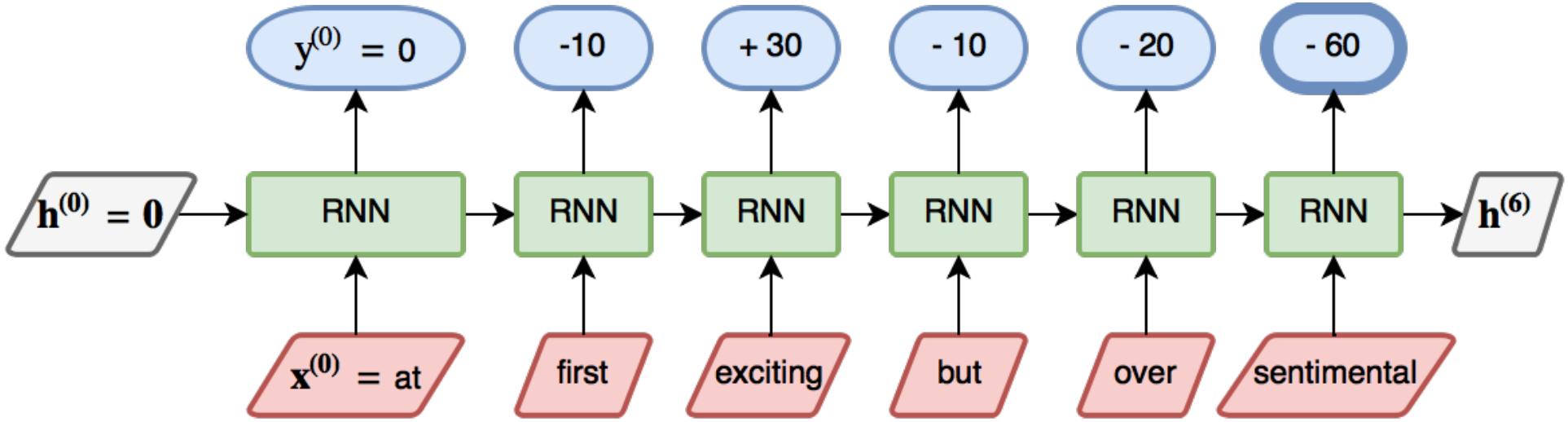
Video Captioning
Machine Translation



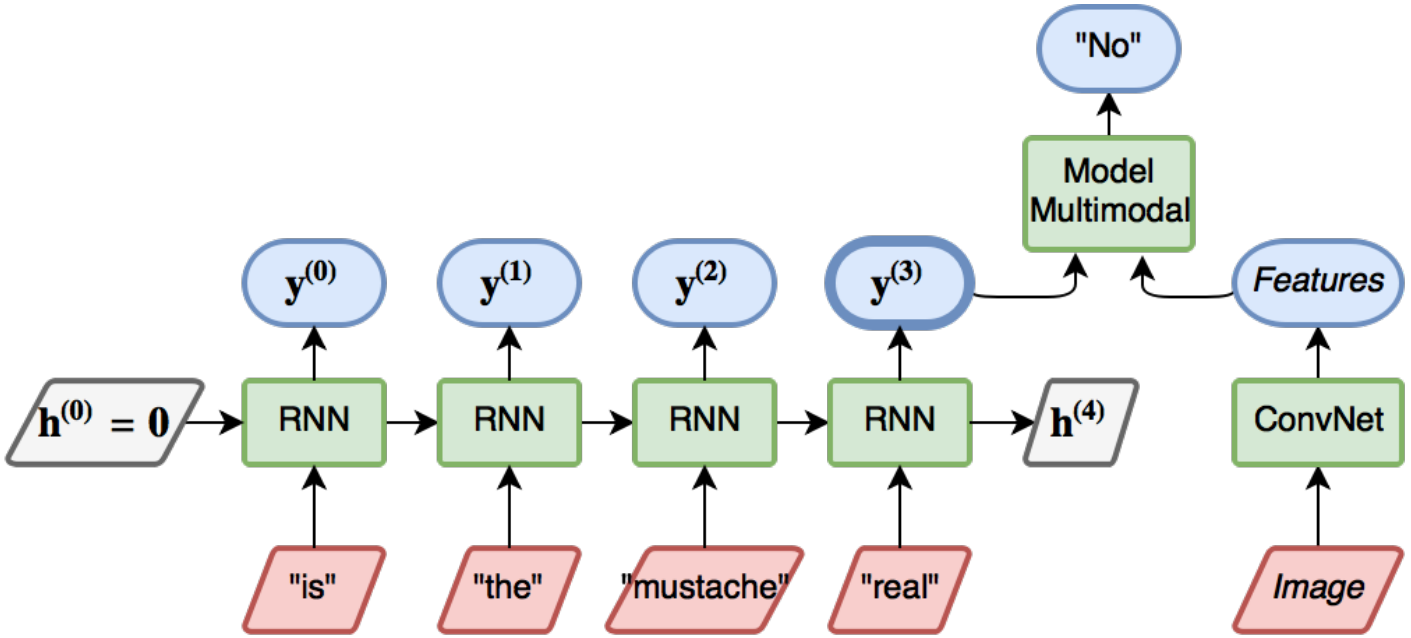
POS Tagging
Language Modeling

Many to One - Sentiment classification

- Input : "At first exciting but over sentimental“
- Output : -60 = Bad review



Many to One - Visual Question Answering

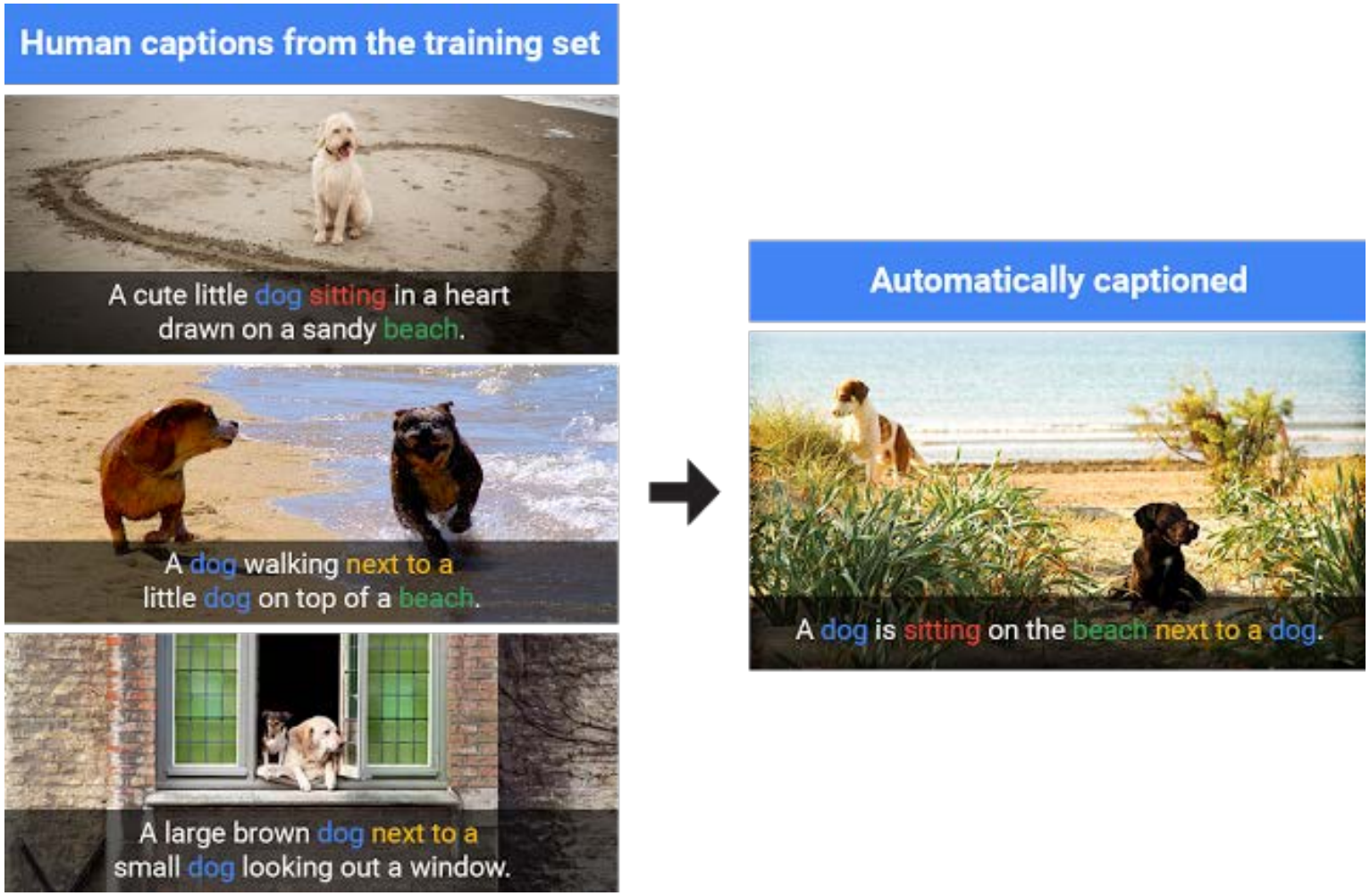


- Output $\mathbf{y}^{(i)}$ is a vector of 2400 dimensions.
- Consider the last output $\mathbf{y}^{(4)}$ to be the final output.

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

One to Many - Image captioning

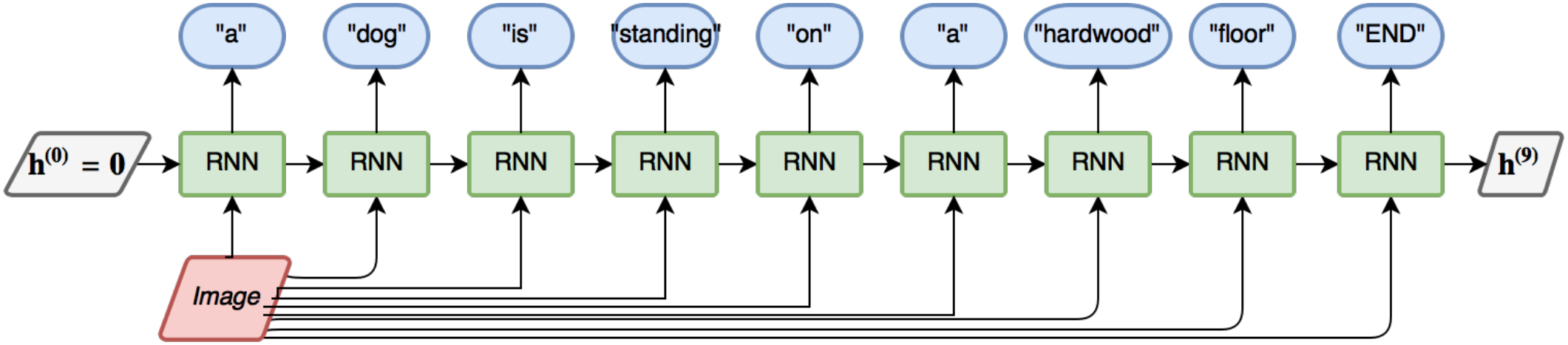


[Xu et al. 2015]

UNIVERSITÉ CÔTE D'AZUR
univ-cotedazur.fr

One to Many - Image captioning

- Input : Image
- Output : "A dog is standing on a hardwood floor."

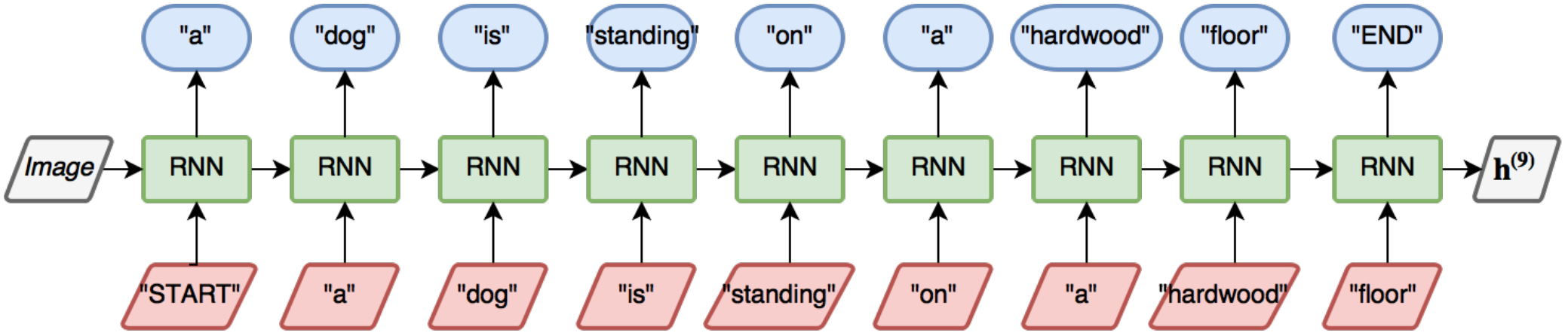


UNIVERSITÉ CÔTE D'AZUR
univ-cotedazur.fr

One to Many - Image captioning

- Input : Image
- Output : "A dog is standing on a hardwood floor."

Different way exists. (Even both way : Image and predicted word as input)



[Karpathy et Fei-Fei 2015]

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Many to Many Parallel - Char-nn

Proof. Omitted. □

Lemma 0.1. *Let $\mathcal{C}$ be a set of the construction.*

Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

.

Proof. This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{\acute{e}tale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

Lemma 0.2. *This is an integer $\mathcal{Z}$ is injective.*

Proof. See Spaces, Lemma ??. □

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

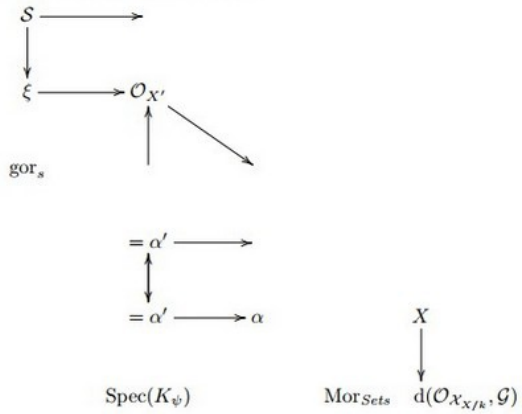
be a morphism of algebraic spaces over S and Y.

Proof. Let X be a nonzero scheme of X. Let X be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

(1) $\mathcal{F}$ is an algebraic space over S.
(2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram



is a limit. Then $\mathcal{G}$ is a finite type and assume S is a flat and $\mathcal{F}$ and $\mathcal{G}$ is a finite type f_*. This is of finite type diagrams, and

- the composition of $\mathcal{G}$ is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

□

Proof. We have see that $X = \mathrm{Spec}(R)$ and $\mathcal{F}$ is a finite type representable by algebraic space. The property $\mathcal{F}$ is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U. □

Proof. This is clear that $\mathcal{G}$ is a finite presentation, see Lemmas ??.
A *reduced above* we conclude that U is an open covering of $\mathcal{C}$. The functor $\mathcal{F}$ is a "field

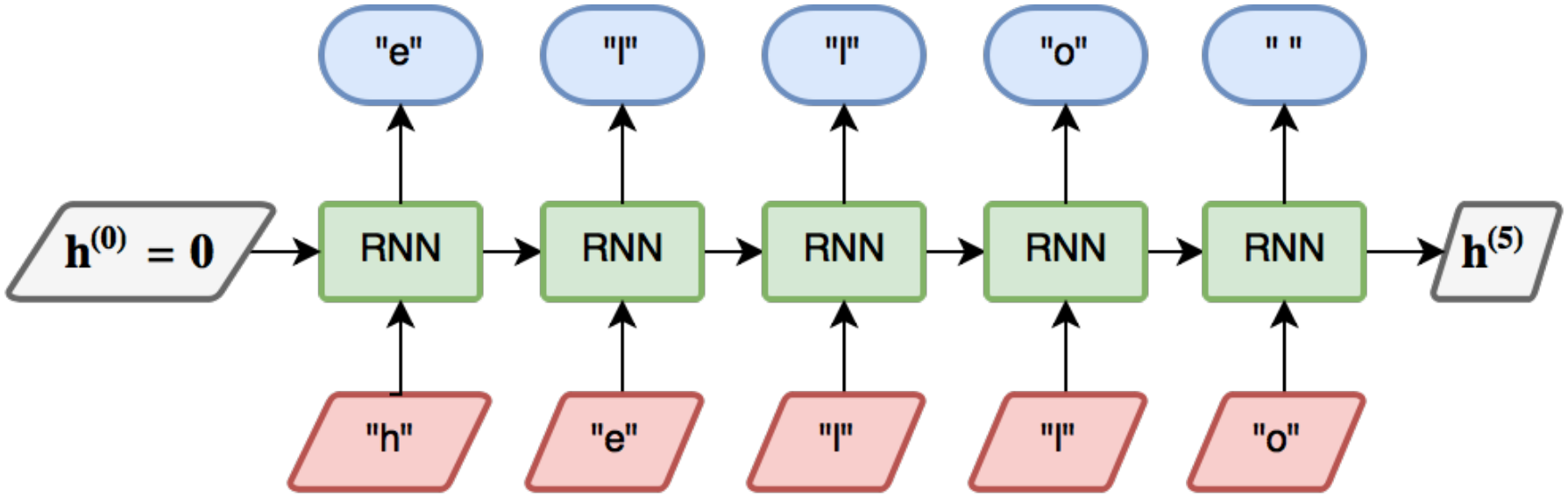
$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\overline{x}} \quad \text{-1}(\mathcal{O}_{X_{\acute{e}tale}}) \longrightarrow \mathcal{O}_{X_\ell}^{-1}\mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^{\overline{v}})$$

is an isomorphism of covering of $\mathcal{O}_{X_i}$. If $\mathcal{F}$ is the unique element of $\mathcal{F}$ such that X is an isomorphism.
The property $\mathcal{F}$ is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme $\mathcal{O}_X$-algebra with $\mathcal{F}$ are opens of finite type over S.
If $\mathcal{F}$ is a scheme theoretic image points. □

If $\mathcal{F}$ is a finite direct sum $\mathcal{O}_{X_\lambda}$ is a closed immersion, see Lemma ??. This is a sequence of $\mathcal{F}$ is a similar morphism.

UNIVERSITÉ CÔTE D'AZUR
univ-cotedazur.fr

Many to Many Parallel - Char-nn



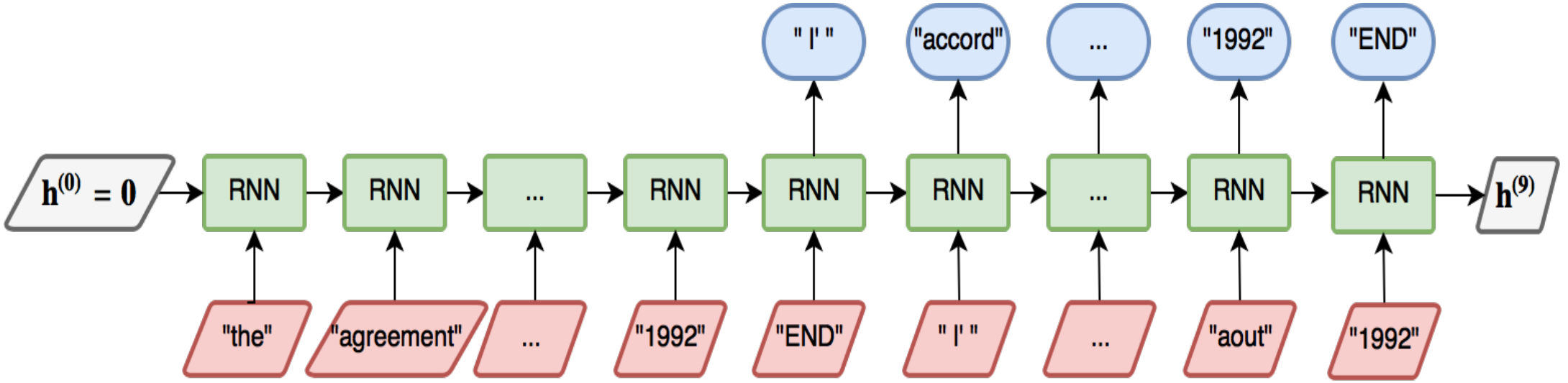
[Karpathy 2015]

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Many to Many - Machine translation text2text

- Input : "The agreement on the European Economic Area was signed in August 1992."
- Output : "L'accord sur la zone économique européenne a été signé en août 1992."



[Bahdanau, Cho et Bengio 2014] [Olah et Carter 2016]

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Named Entity Recognition (NER) in Clinical Notes

NER: classify entities into categories, e.g. drug names and diseases in the medical context

Recognition of Disorder and Medication in the sentence:

"the patient has internal bleeding (A) secondary to warfarin (B)"

Disorder Medication

Label for **A** is strongly related to the label prediction of **B**.

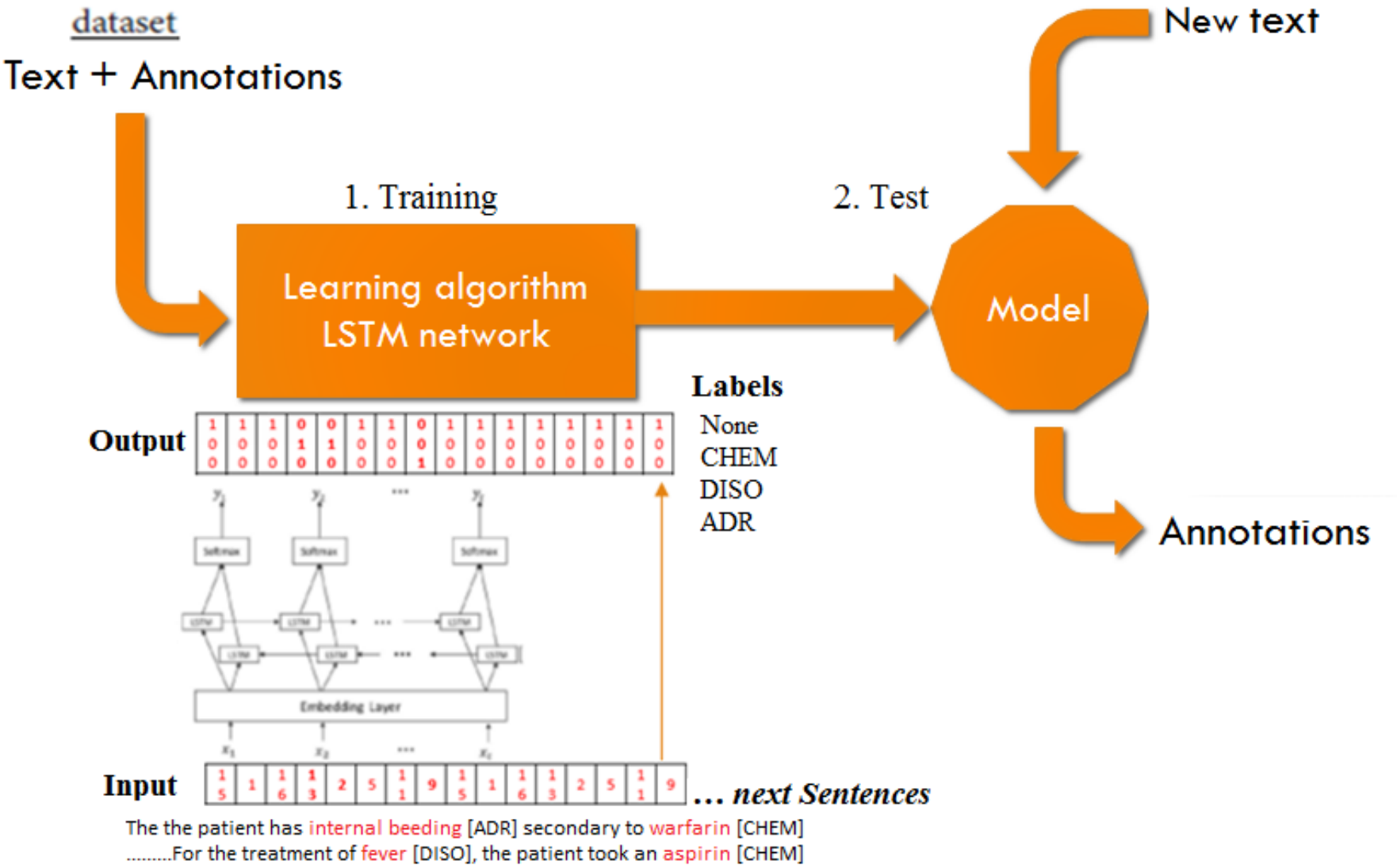
RNN (Recurrent Neural Network) models as **LSTM (Long Short Term Memory)** can model long term label dependencies [1].

[1] **Jagannatha**, Abhyuday et al. "Structured prediction models for RNN based sequence labeling in clinical text." *Proceedings of the Conference on Empirical Methods in Natural Language Processing.* 2016.

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

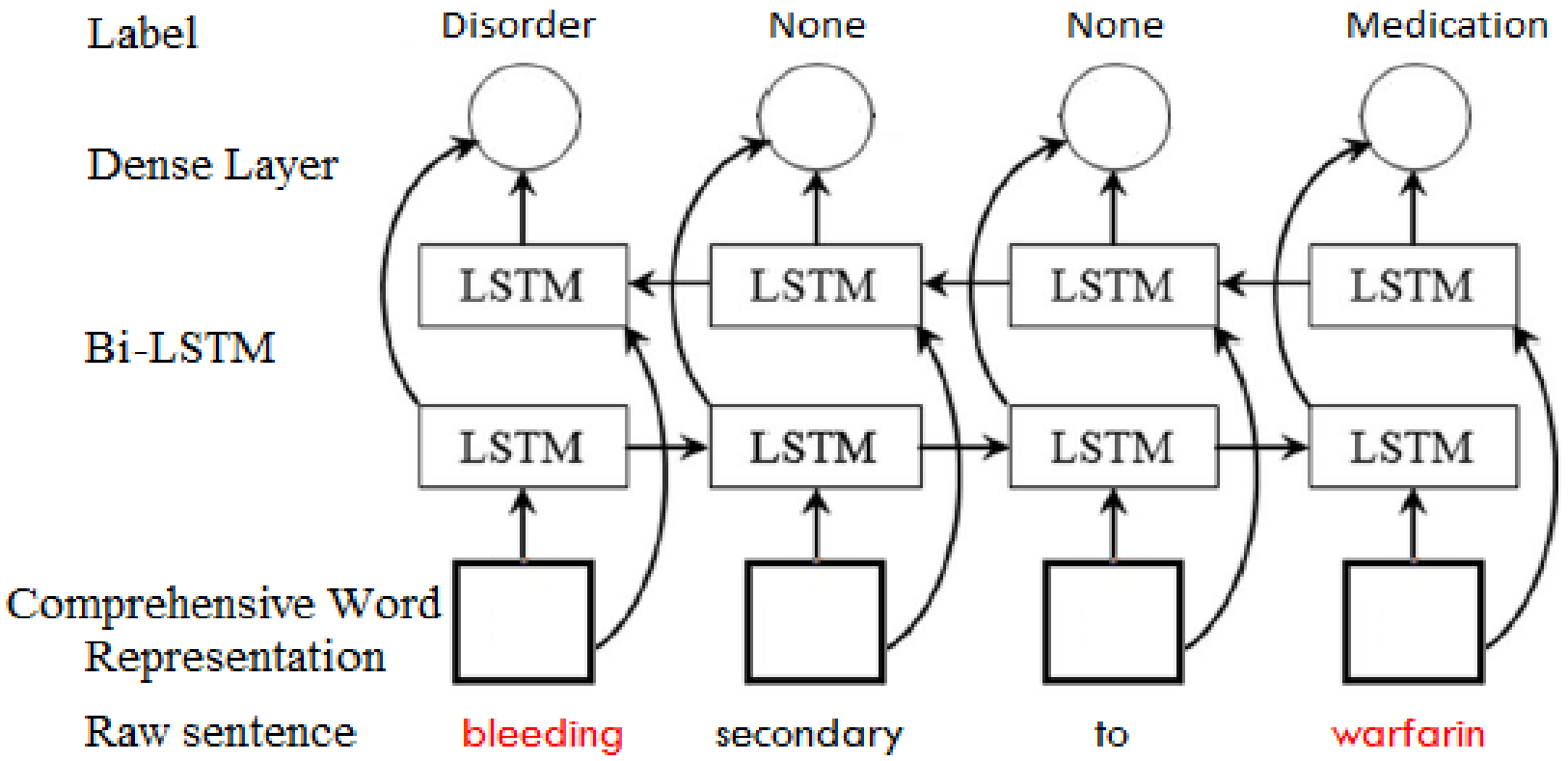
Named-Entity Recognition in Clinical Notes with LSTM



UNIVERSITÉ CÔTE D'AZUR
univ-cotedazur.fr

LSTM network for tagging

Features for the Bi-LSTM layer: Comprehensive word representation



[11] Liu, Z., Yang, M., Wang, X., Chen, Q., Tang, B., Wang, Z., & Xu, H. (2017). Entity recognition from clinical texts via recurrent neural network. *BMC medical informatics and decision making*, *17*(2), 67.

UNIVERSITÉ CÔTE D'AZUR

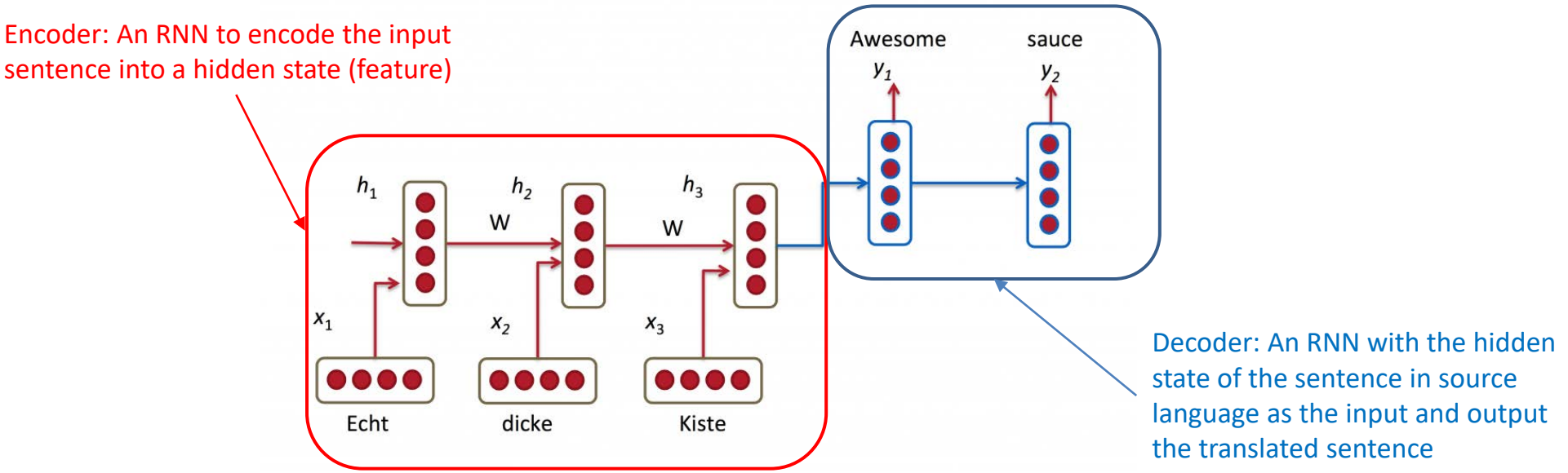
Combining RNN and Autoencoder

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Machine Translation

In machine translation, the input is a sequence of words in source language, and the output is a sequence of words in target language.



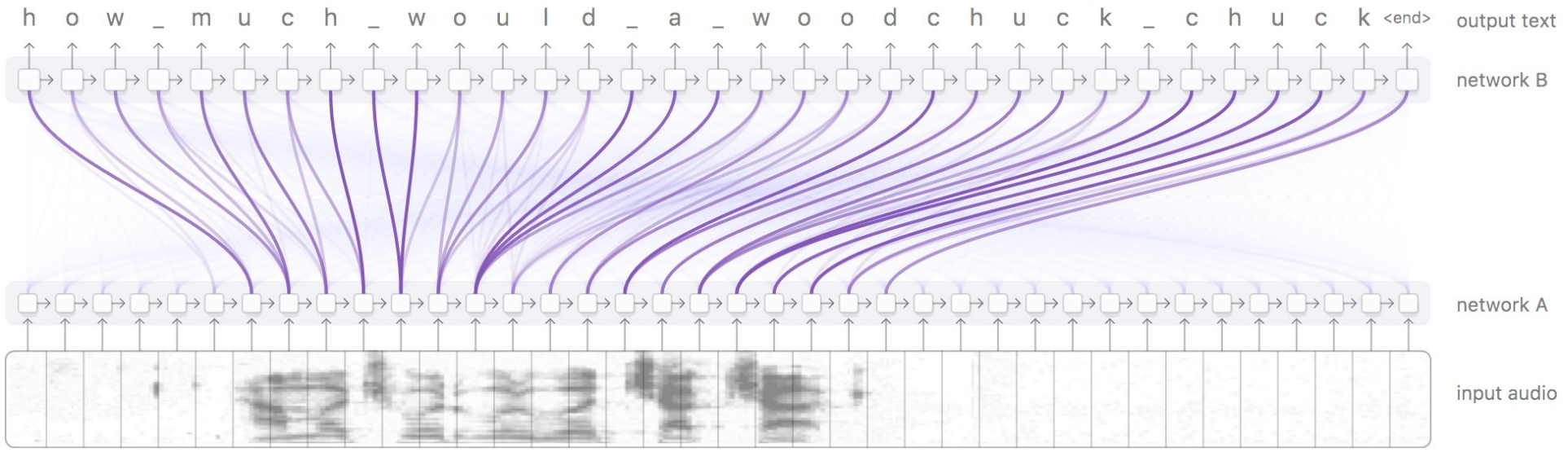
Encoder-decoder architecture for machine translation

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

RNN + Autoencoder + Attention Speech2text

- Input : Audio mp3 (natural language)
- Output : "How much would a woodchuck chuck"



[Chan et al. 2015] [Olah et Carter 2016]

AMAZING BUT BEWARE THE BIASES

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Beware the biases!

Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings

Tolga Bolukbasi[1], Kai-Wei Chang[2], James Zou[2], Venkatesh Saligrama[1,2], Adam Kalai[2]

[1]Boston University, 8 Saint Mary's Street, Boston, MA

[2]Microsoft Research New England, 1 Memorial Drive, Cambridge, MA

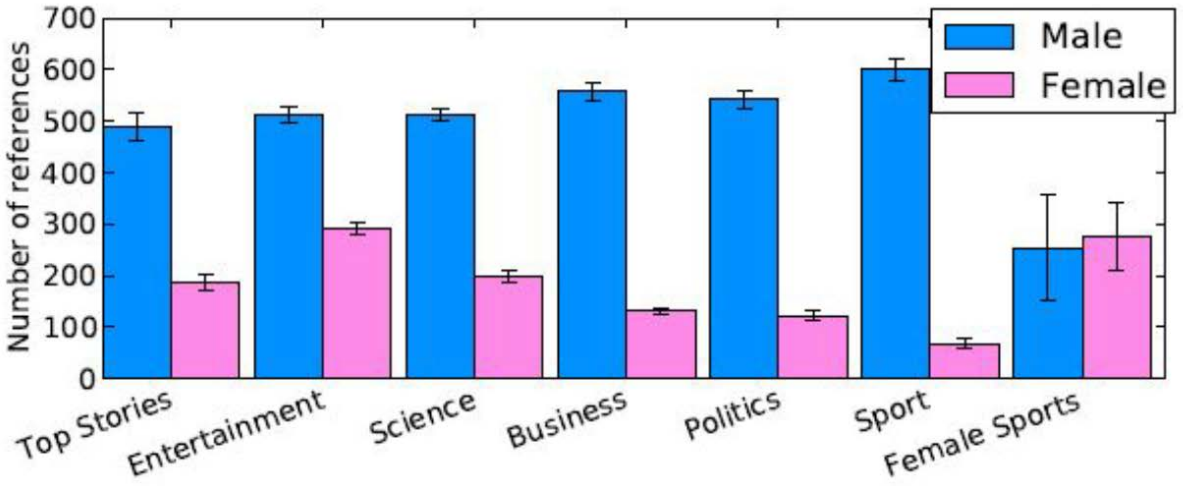
tolgab@bu.edu, kw@kwchang.net, jamesyzou@gmail.com, srv@bu.edu, adam.kalai@microsoft.com

Abstract

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Beware the biases!



https://translate.google.com/

TABLE I: List of the top 10 occupations per gender by their association with gender.

Gender	Occupations most associated with a gender
Male	Manager, Engineer, Coach, Executive, Surveyor, Secretary, Architect, Driver, Police, Caretaker, Director
Female	Housekeeper, Nurse, Therapist, Bartender, Psychologist, Designer, Pharmacist, Supervisor, Radiographer, Underwriter

From Nello Cristianini, at at *Frontier Research and Artificial Intelligence Conference*: https://erc.europa.eu/sites/default/files/events/docs/Nello_Cristianini-ThinkBIG-Patterns-in-Big-Data.pdf

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Beware the biases!

BUSINESS NEWS OCTOBER 10, 2018 / 5:12 AM / 7 MONTHS AGO

Amazon scraps secret AI recruiting tool that showed bias against women

Jeffrey Dastin 8 MIN READ

SAN FRANCISCO (Reuters) - Amazon.com Inc's (AMZN.O) machine-learning specialists uncovered a big problem: their new recruiting engine did not like women.

Forget Killer Robots—Bias Is the Real AI Danger

John Giannandrea.
GETTY

Artificial Intelligence / Robots

Forget Killer Robots—Bias Is the Real AI Danger

John Giannandrea, who leads AI at Google, is worried about intelligent systems learning human prejudices.

by **Will Knight** Oct 3, 2017

Google's AI chief isn't fretting about super-intelligent killer robots. Instead, John Giannandrea is concerned about the danger that may be lurking inside the machine-learning algorithms used to make millions of decisions every minute.

"The real safety question, if you want to call it that, is that if we give these systems biased data, they will be biased," Giannandrea said before a recent Google conference on the relationship between humans and AI systems.

The problem of bias in machine learning is likely to become more significant as the technology spreads to critical areas like medicine and law, and as more people without a deep technical understanding are tasked with deploying it.

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Beware the biases!

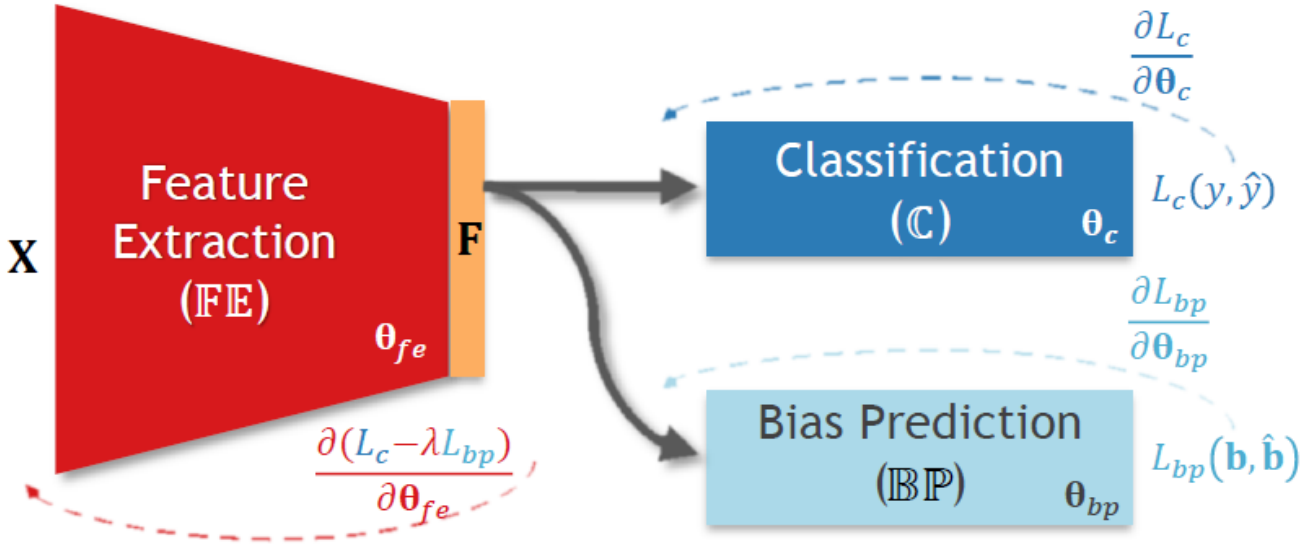


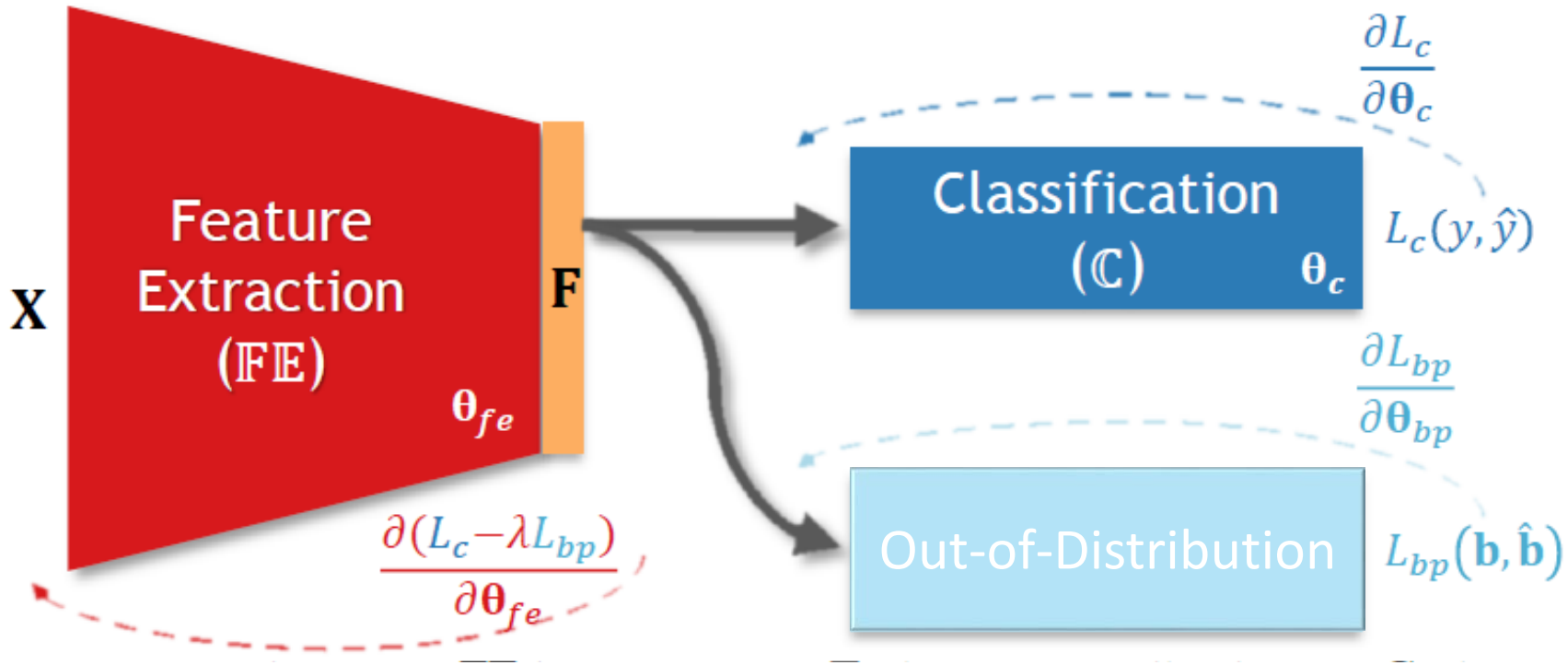
Figure 2: BR-Net architecture: $\mathbb{FE}$ learns features, $\mathbf{F}$, that successfully classify ($\mathbb{C}$) the input while being invariant (not correlated) to the bias variable(s), $\mathbf{b}$, using $\mathbb{BP}$ and the adversarial loss component, $-\lambda L_{bp}$, which is based on correlation coefficient. Forward arrows show forward paths while the backward dashed ones indicate back-propagation with the respective gradient values.

https://arxiv.org/pdf/1910.03676.pdf

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Available Internship



You have to like Maths (at least statistics) and possibly Quebec…

UNIVERSITÉ
CÔTE D'AZUR