ADVANCED DATA ANALYSIS AND PROCESSING TECHNIQUES

Mathematical analysis of Feed-Forward Neural Network

- Michela Sammartino & Lorenzo Della Cioppa

IR0000032 – ITINERIS, Italian Integrated Environmental Research Infrastructures System
(D.D. n. 130/2022 - CUP B53C22002150006) Funded by EU - Next Generation EU PNRR-Mission 4 “Education and Research” - Component 2: “From research to business” - Investment 3.1: “Fund for the realisation of an integrated system of research and innovation infrastructures”

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani

Introduction to Feed Forward Neural Network

- **Feed forward neural networks** are supervised artificial neural networks in which nodes do not form loops. This type of neural network is also known as a **multi-layer perceptron** (MLP) as all information is only passed forward
- MLP are the most popular neural networks, applied to a wide scientific problems that can easily manage **non-linear system** and a huge amount of data
- They can learn the functioning of a scenario, running for a specific time, adapting to its internal structure.
- They are intended as non-linear and non-parametric models that look for a relationship among data.
- They can be used both for **regression** or **classification** problems
- A different approach from the modelling one, able to find the non-linear functional relationship among the variables, exploiting the numerical way and ignoring *a priori* assumption of suitable function or algorithm.
- It has to extract the input-output relation solely from the examples presented, which together are implicitly assumed to contain the information necessary for this relation. The relationship between problem (input) and solution (output) may be quite general

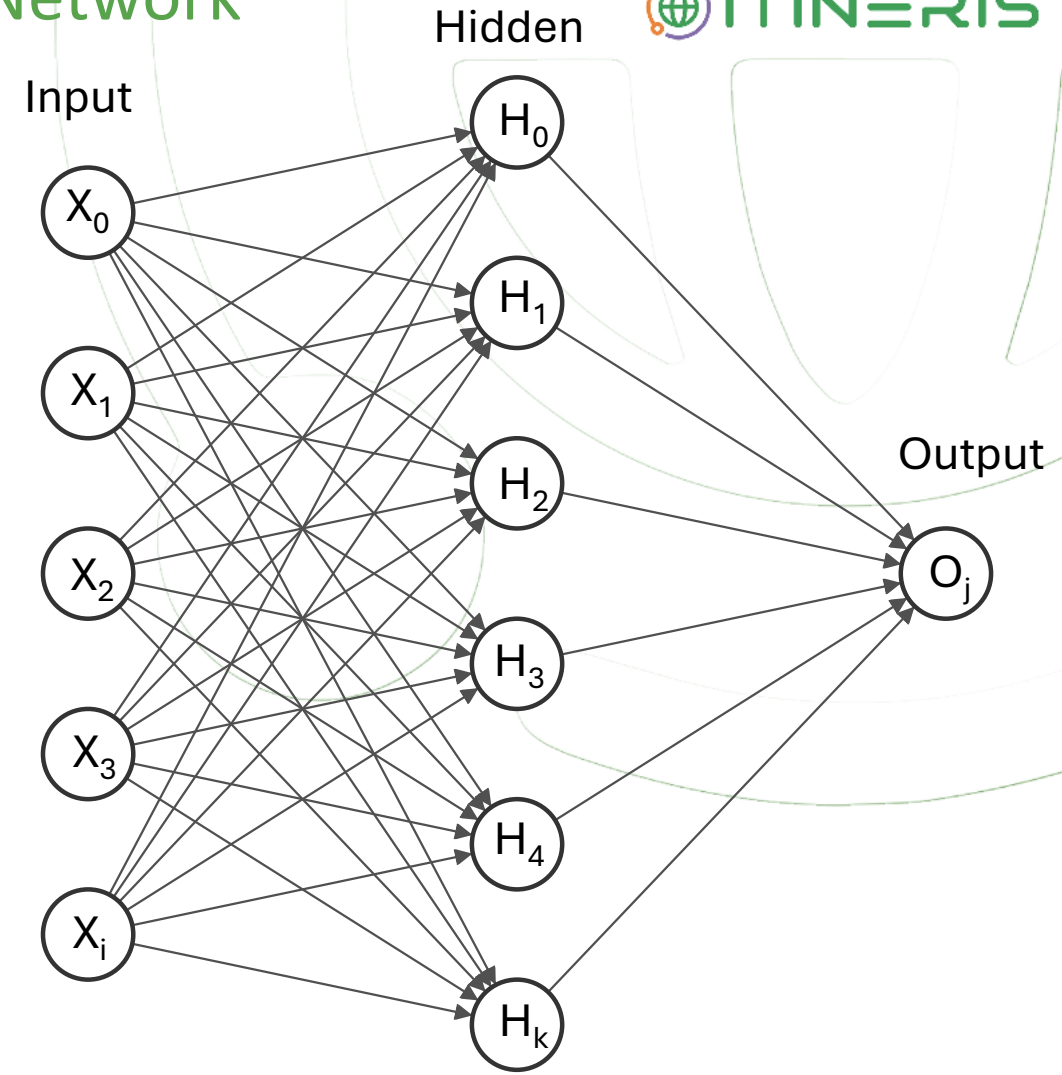
The base structure of a Feed-Forward Neural Network

- The base structure of a FFNN includes different layers divided in:

- **Input layer**: The **neurons** of this layer receive input and pass it on to the other layers of the network. The number of co-predictors used for the prediction match the number of neurons in the input layer.

- **Hidden layer**: Input and output layers get separated by hidden layers. Depending on the type of model, there may be several hidden layers.*

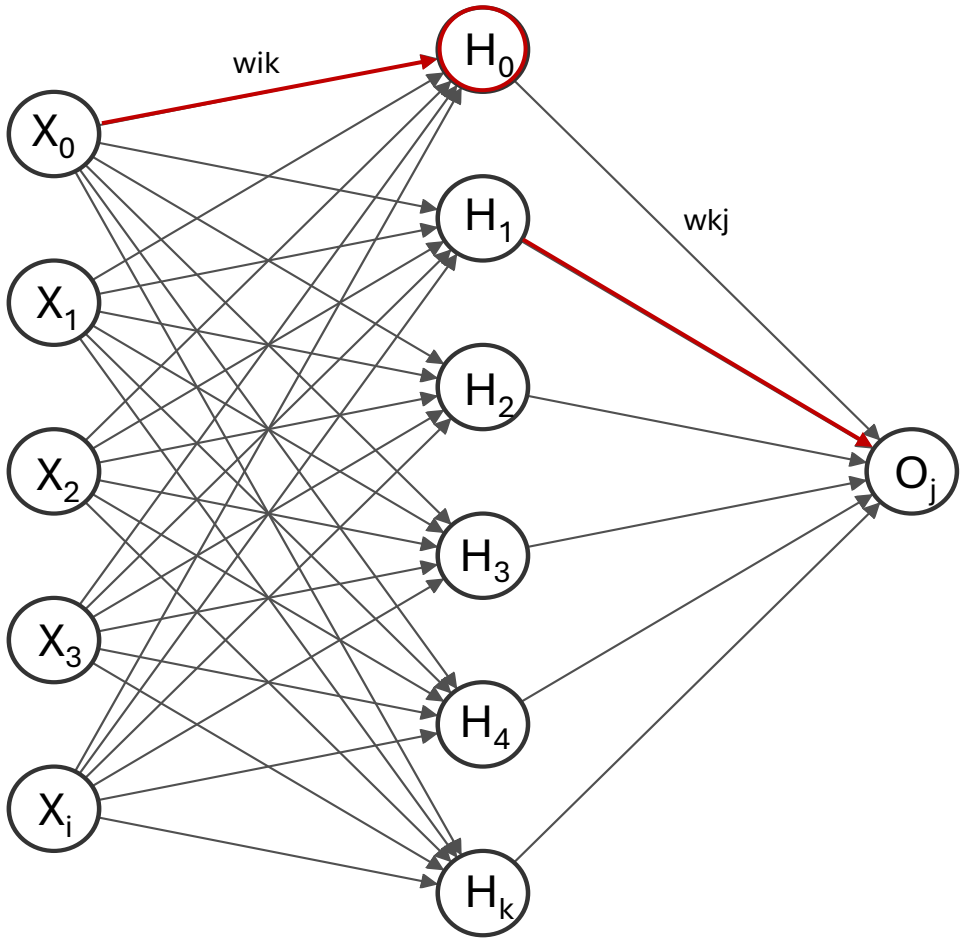
- **Output layer**: According to the type of model getting built, this layer represents the forecasted feature.



Network designed by using the tool at https://alexlenail.me/NN-SVG/

*(conventionally defined as Deep Neural Network when that are many layers deep, meaning there are many layers in between the input and output layer.

The base structure of a Feed-Forward Neural Network

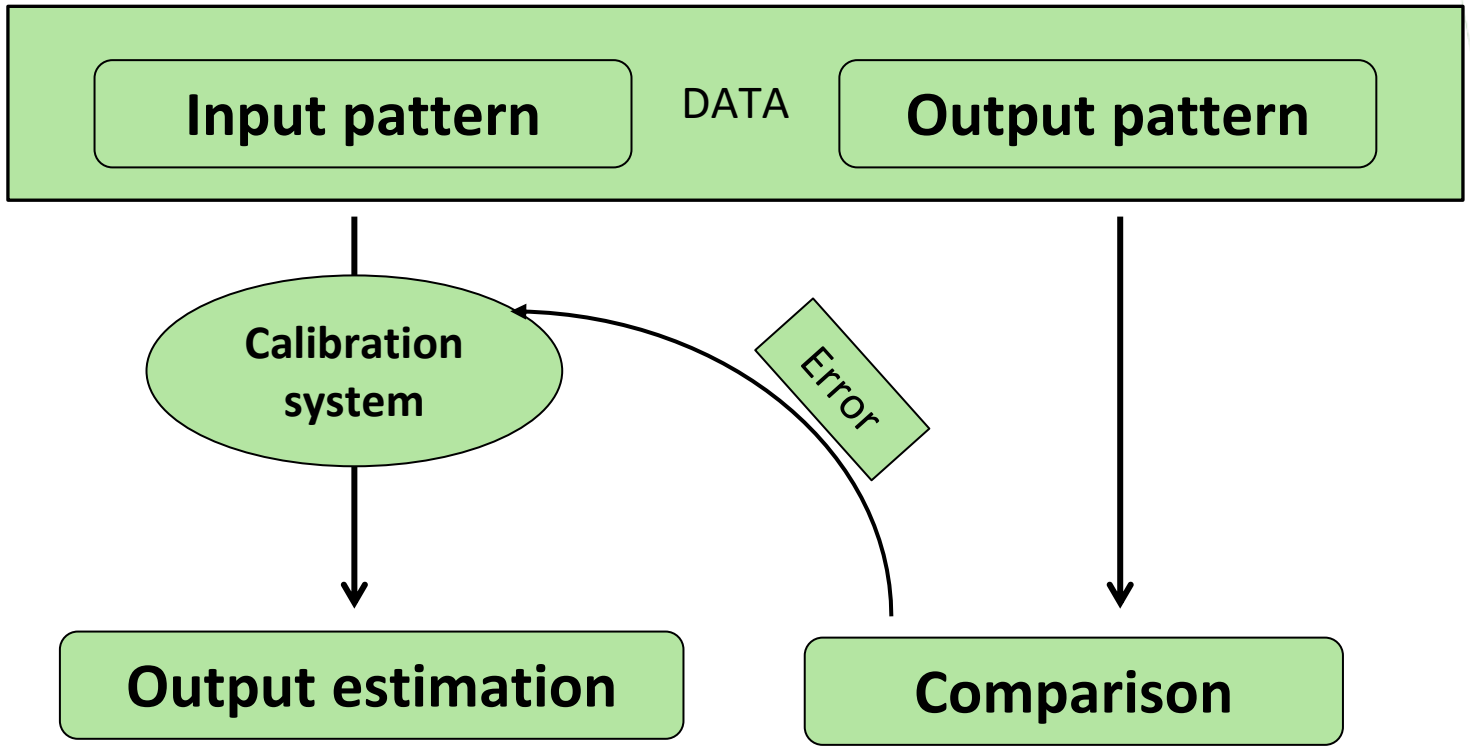


- **Neurons**: Neurons function in two ways: first, they create weighted input sums, and second, they activate the sums to make them normal.

- **Weights**: Neurons are connected by weights. No lateral connection between nodes within one layer, or feedback connection(s) are possible.

Principal goal: create a generalized system to retrieve the output value from input data, minimizing the error between ($Output_{estimated} - Output_{target}$)

How does it work? – Error Backpropagation algorithm



BPN Backpropagation Neural Network

adapted from Lek and Guègan, 2000

- **Backpropagation (Backward Propagation of Errors)** is the fundamental algorithm used to train artificial neural networks by adjusting the model's weights based on the error obtained from predictions. It uses **gradient descent** to minimize the loss function.
- The conceptual basis of the back propagation algorithm was first presented in 1974 by Webos, then reinvented by Rumelhart et al. (1986).
- Extract the **input-output relation** solely from the **examples** presented, which together implicity contain the information for this relationship.
- If the network gives the wrong answer, then the weights are corrected so that the error lessens.

Training phase

STEP **1**

Forward-propagation

Compute prediction

A set of input-output values is presented to the network iteratively. After a specific number of iterations, the training is stopped and the performance of network is tested.

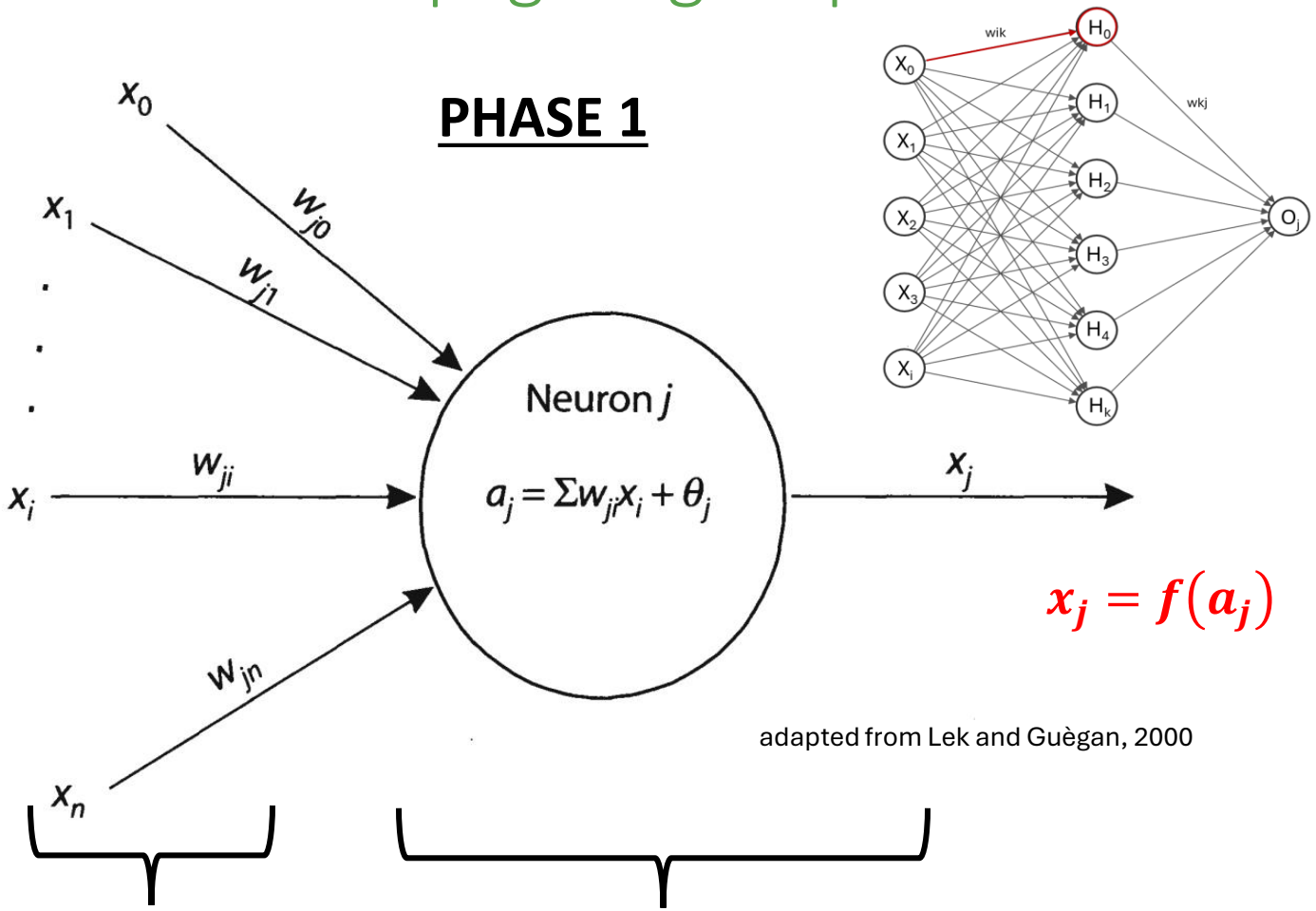
STEP **2**

Backward-propagation

Update the weights

The output is compared with the target value and the error is calculated. This led to change the weights moving from output layer backward to hidden layers.

Forward-Propagating Step



$$x_j = f(a_j)$$

adapted from Lek and Guègan, 2000

**weights contain the knowledge of the neuronal network about the problem ➡ solution relationship.*

1. Input data x_i is passed through the neural network.
2. Each neuron computes the weighted sum: a_j
3. Apply an activation function to a_j to compute the output

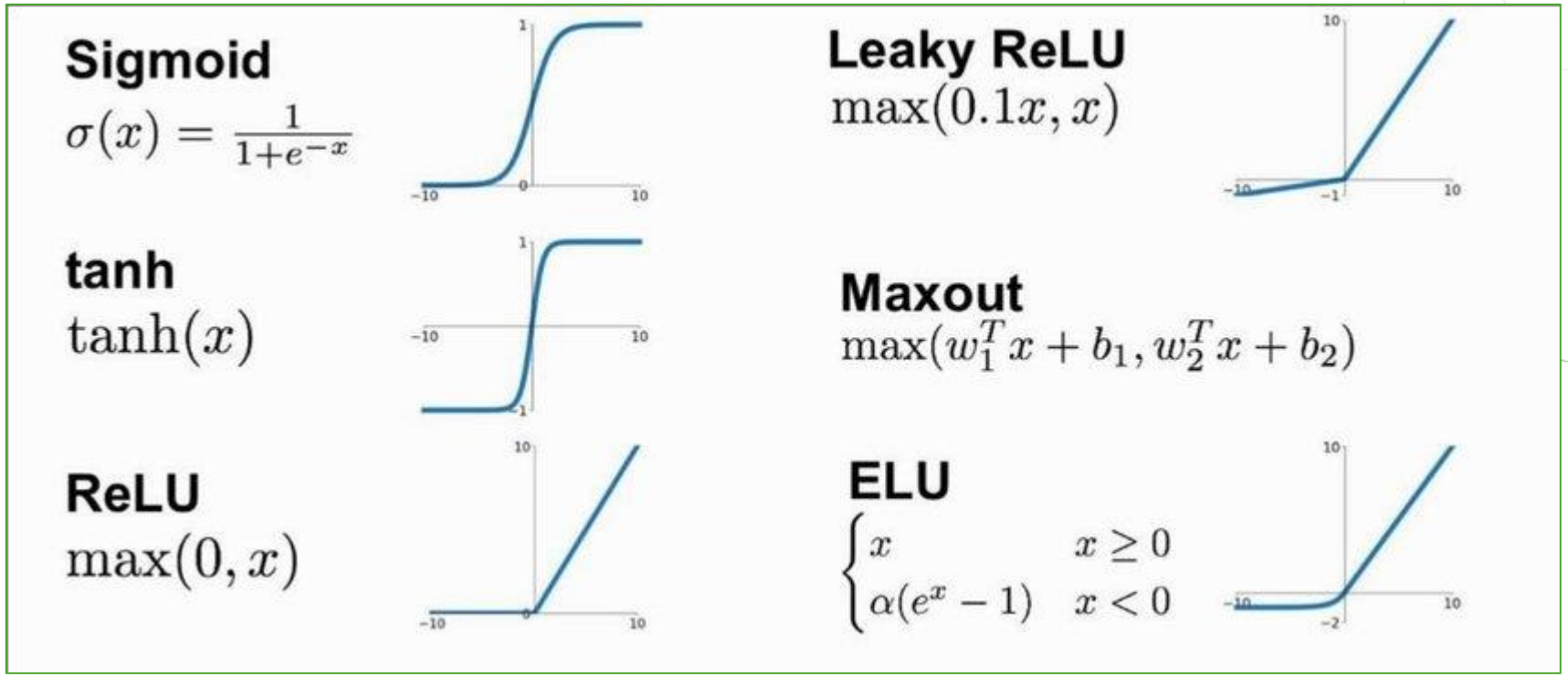
- INPUT $\mathbf{x_i}$
- WEIGHT $\mathbf{w_{ij}}$
- INPUT COMBINATION $\mathbf{a_j}$
- ACTIVATION FUNCTION $\mathbf{x_j}$
- BIAS $\boldsymbol{\theta_j}$

- The artificial neuron receives many inputs, but only a single output, which can stimulate many other neurons in successive layer.

- Positive weight= excitatory
- Negative weight= inhibitory

Types of Activation Function

Once the activation of the neuron is calculated, we can determine the output value (i.e. the response) by applying a transfer/activation function



from Shruti Jadon (Medium website)

$$x_j = f(a_j) = \frac{1}{1 + e^{-a_j}}$$

- The inputs are usually scaled between [0,1] or [-1,1] interval before feeding the network

Backward-Propagating Step (1)

OUTPUT LAYER → HIDDEN LAYER

- Compare the output pattern to the target value
- Calculate the error value: the difference between the predicted output and the actual output is calculated using a loss function.
- This error is passed backward through the network. Change the incoming weights staring from output toward the hidden layer.

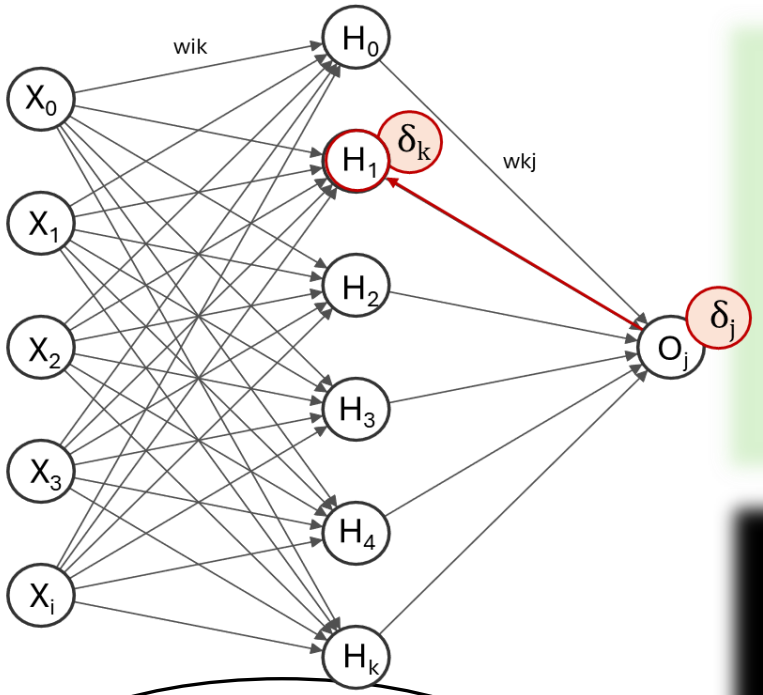
$$E_t = \frac{1}{2} * \sum_{j \in C} \left(out_{target} - out_{estimated}\right)^2$$

Amount of Output Error for each input

$$E = \frac{1}{m} * \sum_{p=0}^{m-1} E_t$$

- t= n. iteration
- C= n. of neurons of the output layer
- M=n. of pattern dataset

Backward-Propagating Step (1)



Output layer:
we have an $output_{target}$ and we compare it to the $output_{estimated}$ from the NN

Hidden layer:
we have not an output of reference to compare to the hidden layer output

OUTPUT LAYER ⟶ HIDDEN LAYER

Minimize the error following the **Optimizer*** method

This is where the "learning" occurs. Using this error, we calculate the gradients of the error with respect to our weights and biases.
This phase computes how much each weight contributed to the error and adjusts them accordingly using gradient descent.

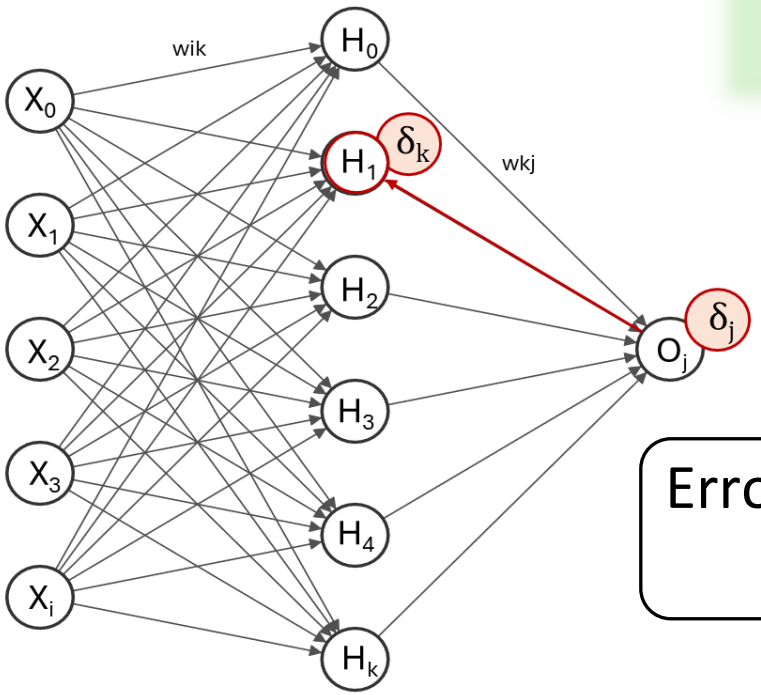
$$\frac{\partial E}{\partial w_{(o-h)}}$$

$$\delta_{out} = f'(a_o)(O_{target} - O_{estimated})$$

Error gradient for the output layer

* *Gradient descendent (GD), SGD, Adam)*

Backward-Propagating Step (1)



HIDDEN LAYER ⟶ **INPUT LAYER**

$$\delta_{hidden} = \left[\sum_{out} \delta_{out} w_{(out-hidden)}\right] f'(a_{hidden})$$

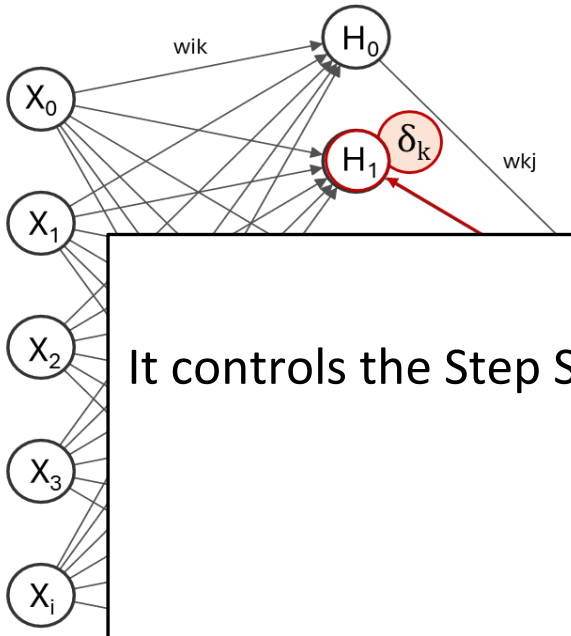
Error gradient for Hidden layer

Minimization of the Error

$$\Delta W_{(out-hidden)} = \eta \delta_{out} O_{hidden}$$
$$\Delta W_{(hidden-input)} = \eta \delta_{hidden} O_{input}$$

Variation of connection weights

Backward-Propagating Step (1)



HIDDEN LAYER ⟶ INPUT LAYER

Learning rate

It controls the Step Size in Weight Update, directly influencing how quickly or slowly the model learns.

$0<\eta<1$

High η ⟶ instability & unsatisfactory learning

Low η ⟶ excessively low learning

$$\Delta W_{(out-hidden)} = \eta \delta_{out} O_{hidden}$$

$$\Delta W_{(hidden-input)} = \eta \delta_{hidden} O_{input}$$

Variation of the connection weights

At each iteration…

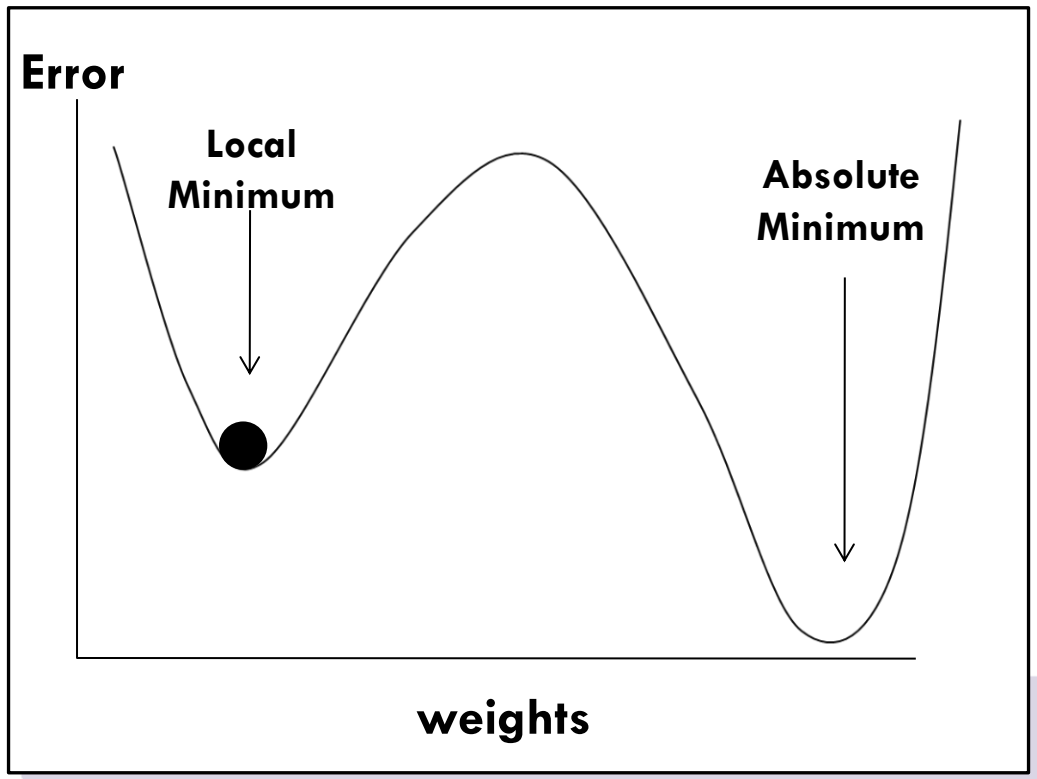
Update of weights

$$W_{(out-hidden)}(t+1) = W_{(out-hidden)}(t) + \Delta W_{(out-hidden)}(t)$$

$$W_{(hidden-input)}(t+1) = W_{(hidden-input)}(t) + \Delta W_{(hidden-input)}(t)$$

$$W_{(out-hidden)}(t+1) = W_{(out-hidden)}(t) + \Delta W_{(out-hidden)}(t) + \alpha \Delta W_{(out-hidden)}(t-1)$$

$$W_{(hidden-input)}(t+1) = W_{(hidden-input)}(t) + \Delta W_{(hidden-input)}(t) + \alpha \Delta W_{(hidden-input)}(t-1)$$



Momentum

0<α <1

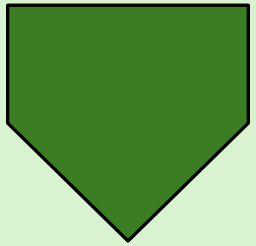
To get out of the local minima

The momentum term helps to incorporate a fraction of the previous weight update in the current weight update:

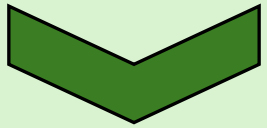
1. Accelerate convergence in the direction of consistent gradients
2. Reduce oscillations in the weight updates

Overtraining or Overfitting of Network

If overtrained, the network has a good memory in the detail of data



It loses the capacity to generalize



To avoid this

a) The **number of epochs**
b) The **number of hidden layers**
c) The **number of hidden neurons**
d) Regularization strategies **(earlystopping/dropout)**

How to avoid overfitting

- Early stopping is a regularization technique used in machine learning to prevent overfitting
  The training is stopped when the model's performance on a validation dataset starts to deteriorate

1.Monitor Performance: Track a validation metric (e.g., validation loss, accuracy) during training.

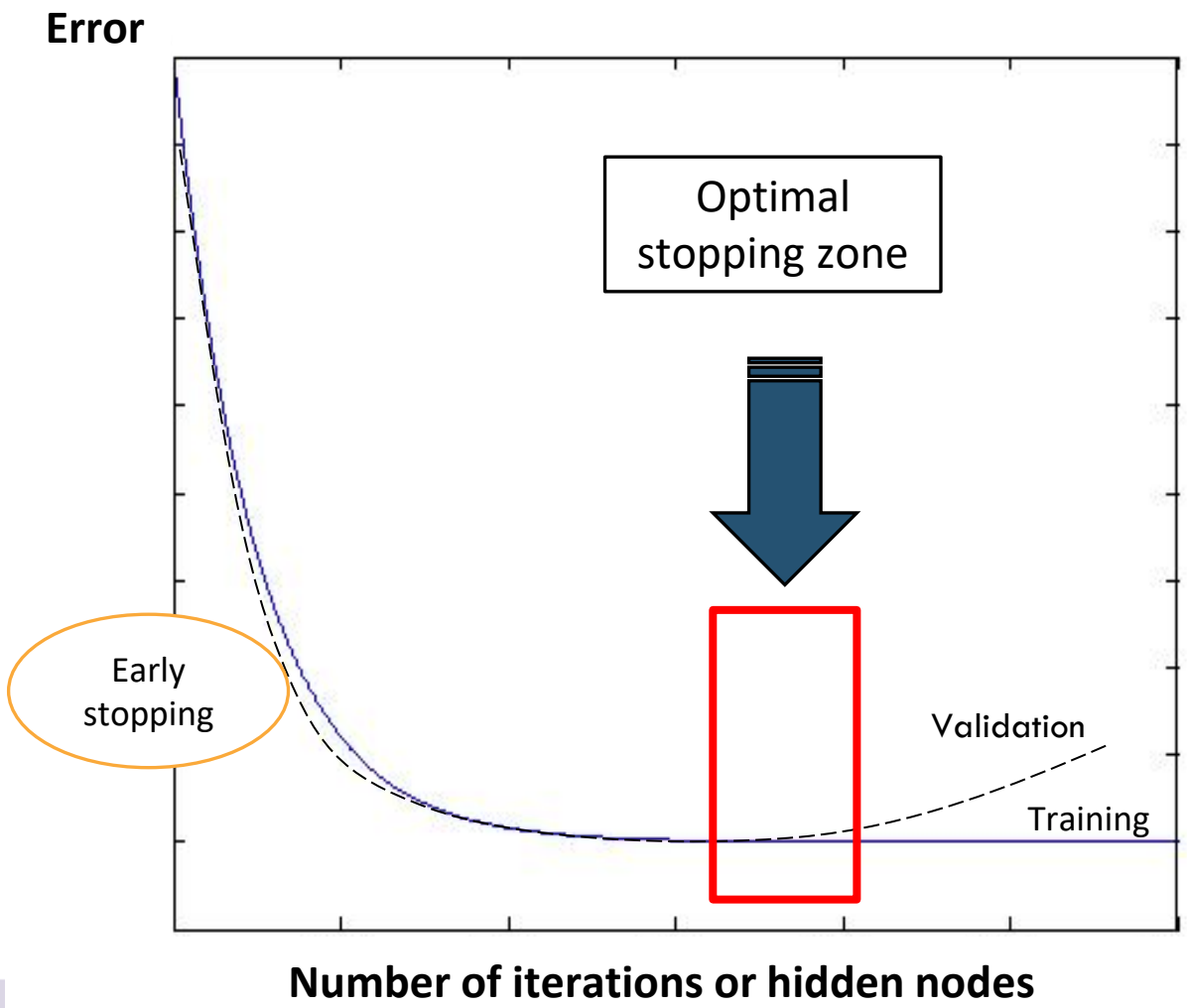
2.Define a Patience Parameter: Set a number of epochs to wait before stopping if no improvement is seen.

3.Stop Training: If the validation performance does not improve for a specified number of epochs, training is halted.

4.Restore Best Weights (Optional): Some implementations restore the model to the best-performing state before stopping.

Benefits of Early Stopping:
•Prevents overfitting by stopping before the model memorizes noise.
•Saves computational resources by reducing unnecessary training.
•Enhances generalization to unseen data.

Limitations:
•Requires careful tuning of patience and monitoring metrics.
•May stop too early if training dynamics are not well understood.



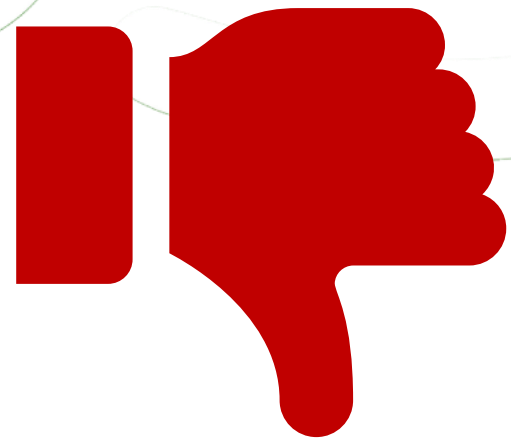
adapted from Lek and Guègan, 2000

Advantges in using Artificial Neural Network

- Able to generalize
- Dynamic behaviour
- Able to reproduce a non linear system
- Able to classify complex patterns
- Useful for both regression and classification problems

Disadvantges in using Artificial Neural Network

- The dimension of the dataset can have an impact on network performances
- We cannot know *a priori* the perfect NN configuration, therefore we must test it!
- Each run will give different weights with respect the previous one, at each run we have a different configuration
- Sometimes it is considered like a black box and therefore it moves away the eventual users



Some examples of ANN applications in marine sciences

- ANNs were applied to:
- Sea level prediction (Röske, 1997)
- Ocean salinity (Chen and Hu, 2017)
- Wave prediction (Ducournau and Fablet, 2016)
- Storm surges and extreme sea level heights (Sahoo and Bhaskaran, 2019).
- Surface currents (Sinha and Abernathey, 2021).
- Vertical profiles of chlorophyll-a from surface ocean data (Sammartino et al., 2020, Chen et al. 2022b)
- Primary production of phytoplankton (Scardi 1999, Mattei et al., 2018)
- Vertical bio-optical and bio-geochemical variables from Bio-Argo data combined with satellite measurements (Sauzède et al., 2015, 2016)
- Water-column nutrients from satellite and BCG-Argo measurement (Sauzède et al. 2017)

Most of these references are taken from Tao Song et al., 2023 (Review)

ADVANCED DATA ANALYSIS AND PROCESSING TECHNIQUES

Introduction to Long Short-term memory network (LSTM)

- Michela Sammartino & Lorenzo Della Cioppa

IR0000032 – ITINERIS, Italian Integrated Environmental Research Infrastructures System
(D.D. n. 130/2022 - CUP B53C22002150006) Funded by EU - Next Generation EU PNRR-Mission 4 “Education and Research” - Component 2: “From research to business” - Investment 3.1: “Fund for the realisation of an integrated system of research and innovation infrastructures”

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani

Introduction to Recurrent Neural Networks

- Recurrent Neural Networks (RNNs) are a type of neural network architecture which is mainly used to detect patterns in a sequence of data
- What differentiates Recurrent Neural Networks from FFNNs/MLPs is how information gets passed through the network.

  - FFNN pass information through the network without cycles/the RNN has cycles and transmits information back into itself.

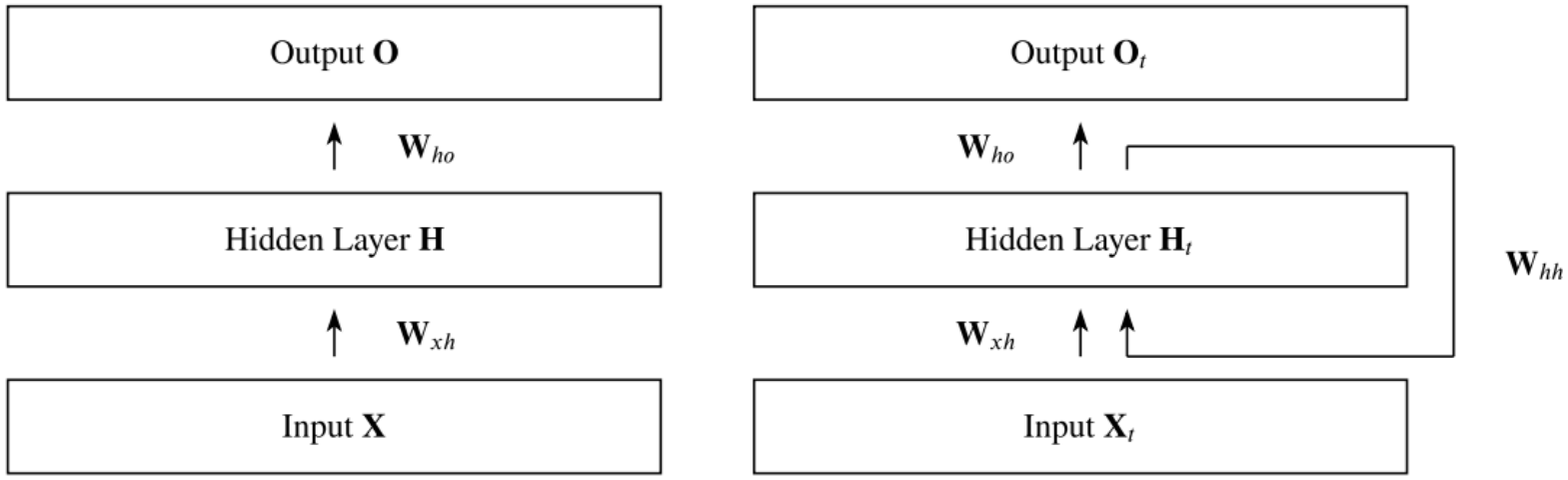


Figure 1: Visualisation of differences between Feedfoward NNs und Recurrent NNs

From Robin M. Schmidt, 2019

Disadvantages of RNNs

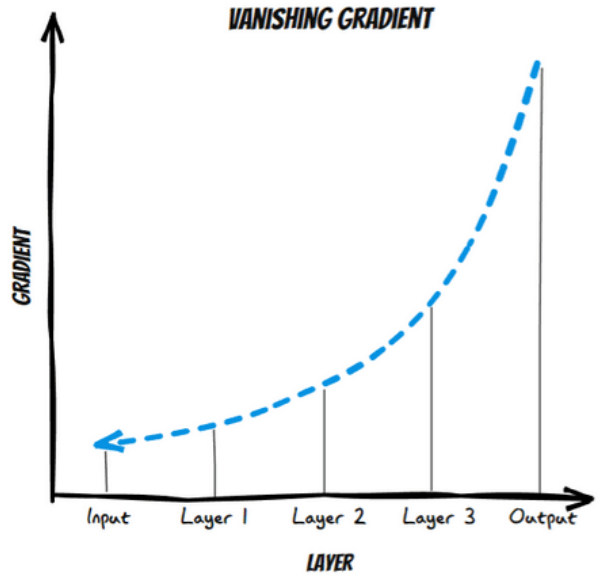
Vanishing & Exploding Gradients

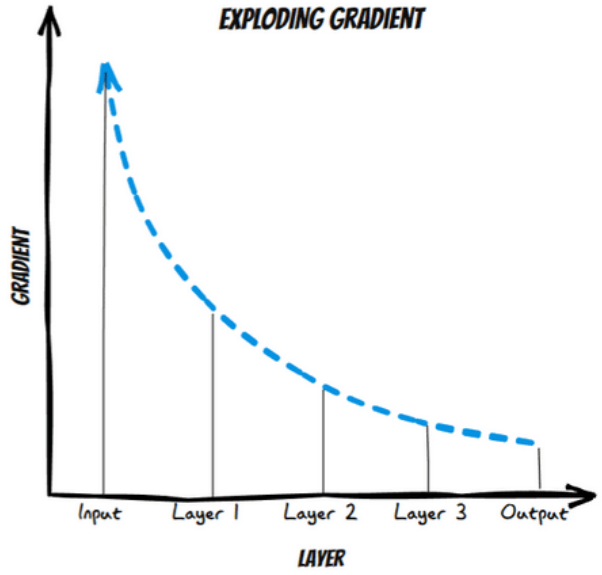
- The vanishing gradient problem in RNNs occurs because, as the gradients are propagated backwards through time, they can become very small due to the repeated multiplication of gradients in each time step.

- If there are small values (< 1) in the matrix multiplication this causes the gradient to decrease with each layer (or time step) and finally vanish → low/zero contribution of states from earlier time

- If there are large values (> 1) during matrix multiplication → the gradient explodes

One solution to avoid the Vanishing was the introduction of Long Short-term Memory units (LSTMs)

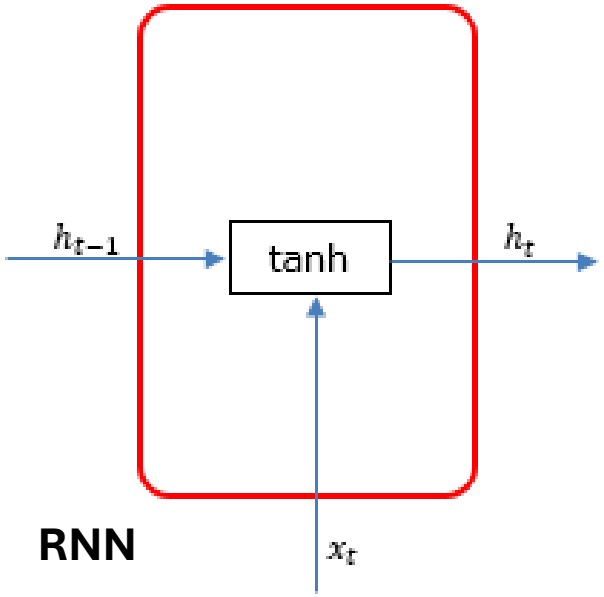


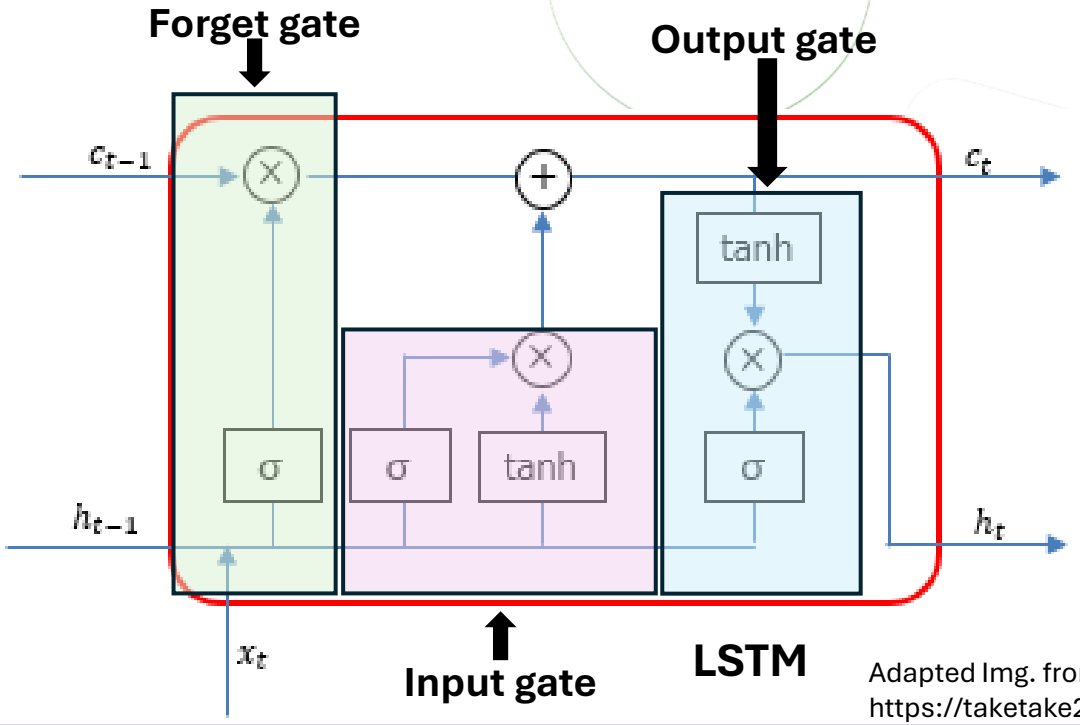


https://medium.com/@muradatcorvit23/unlocking-the-power-of-lstm-gru-and-the-struggles-of-rnn-in-managing-extended-sequences-05879b6899d3

Introduction to LSTMs

- LSTM is a type of RNNs able to process and analyse sequential data such as time series, text and speech (tasks where the step t is dependent on what happens in step t-1)
- LSTM are used in several application, e.g. for speech recognition or time series forecasting
- They were introduced by Hochreiter & Schmidhuber (1997)
- LSTM solves the vanishing gradient problem that traditional RNNs face by using special memory cells and gating mechanisms



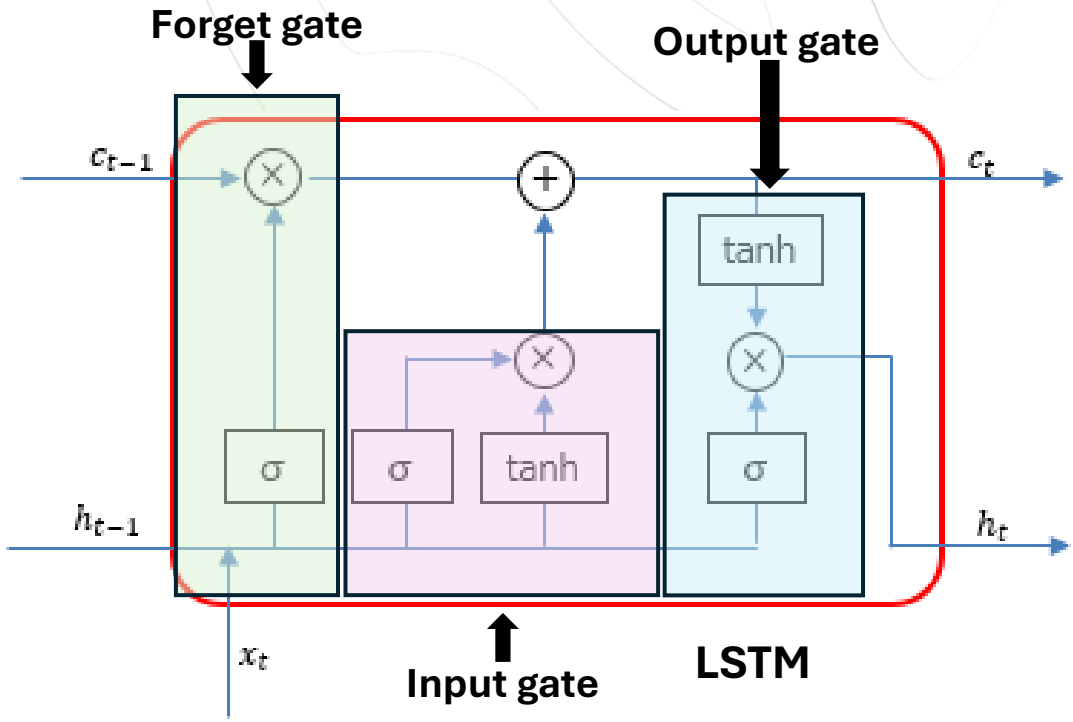


Adapted Img. from https://taketake2.com/P56_en.html

LSTM memory units

- An LSTM unit consists of three main gates, that control the flow of information through the LSTM cell, allowing it to decide what to remember, what to forget, and what to output and allow to control the flow of gradients through time addressing the vanishing gradient issue.

- **FORGET GATE**: Determines which information to discard from the previous hidden state and the current input. By using a sigmoid activation function, it generates values between 0 and 1, representing the proportion of data retained in the cell state.
- **INPUT GATE**: Determines which new information should be incorporated into the cell state. It consists of two components: a sigmoid activation function that selects the values to update and a tanh activation function that generates new candidate values for the cell state.
- **OUTPUT GATE**: Determines the data to be presented as the LSTM cell's current output. It processes the updated cell state along with the current hidden state to produce the final output.

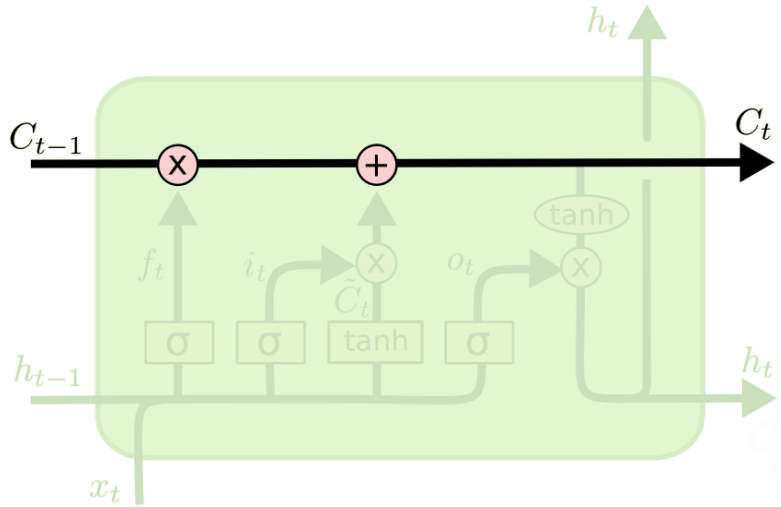


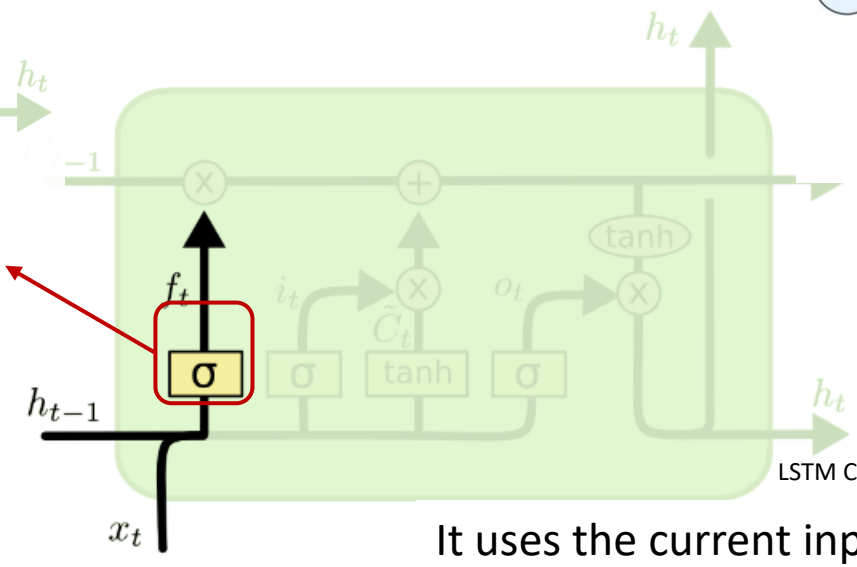
Adapted from Img. from https://taketake2.com/P56_en.html

LSTM memory units functioning

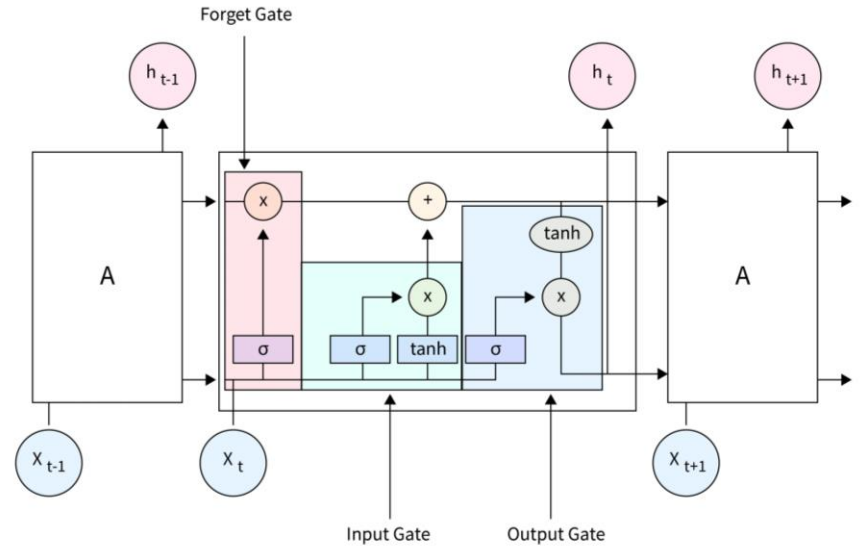
STEP 1: What information to throw away from the cell state (Forget gate).

what information from the previous cell state (C_{t-1}) should be discarded, retaining the fraction to be combined with the cell state.





- **0** means "completely forget" the corresponding information in the cell state.
- **1** means "completely retain" the corresponding information in the cell state.
- Values in between represent partial retention.



By Steve Jerome Lawrence

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

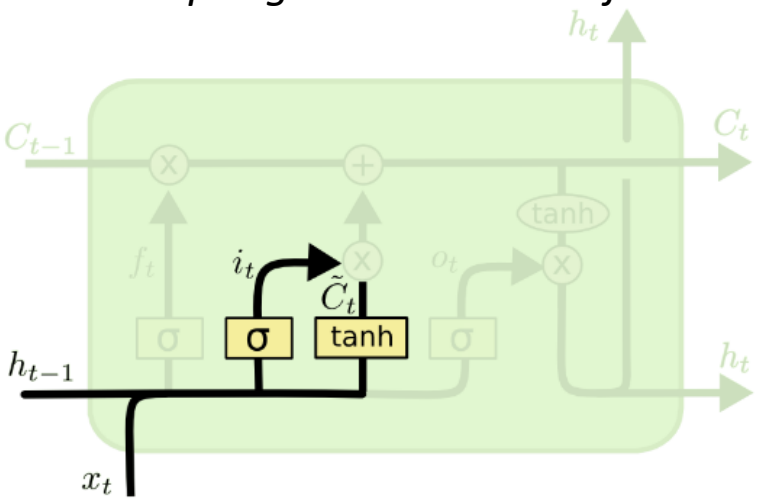
LSTM Cell (Source — http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

It uses the current input (x_t) and the previous hidden state (h_{t-1}) to compute a forget gate activation vector (f_t).

LSTM memory units functioning

STEP 2: What new information to store in the cell state (Input gate).

*The input gate works in conjunction with the **candidate cell state** ($\widetilde{C}_t$) to update the cell state.*



1. Input Gate Activation

Decides which parts of the candidate cell state ($\widetilde{C}_t$) should be added to the cell state

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

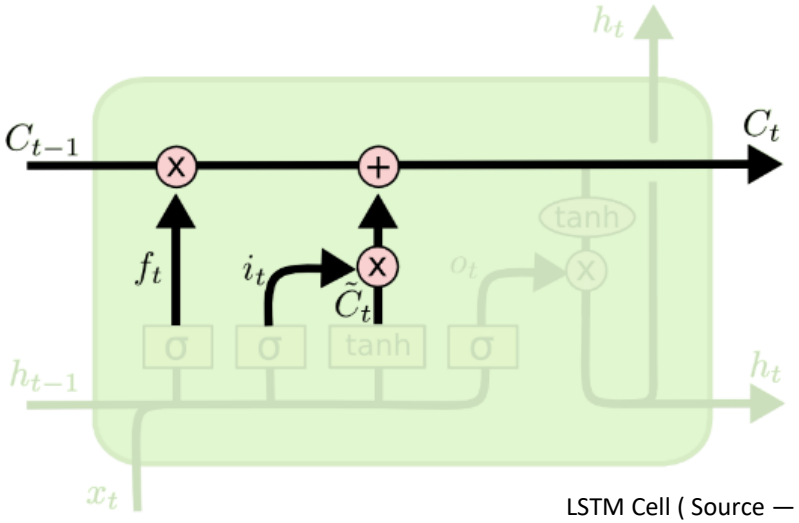
2. Candidate Cell State

Represents the new information that could potentially be added to the cell state.

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

**tanh layer creates a vector of new candidate values $\widetilde{C}_t$, that could be added to the state.*

STEP 3: Update the old cell state into the new cell state C_t



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

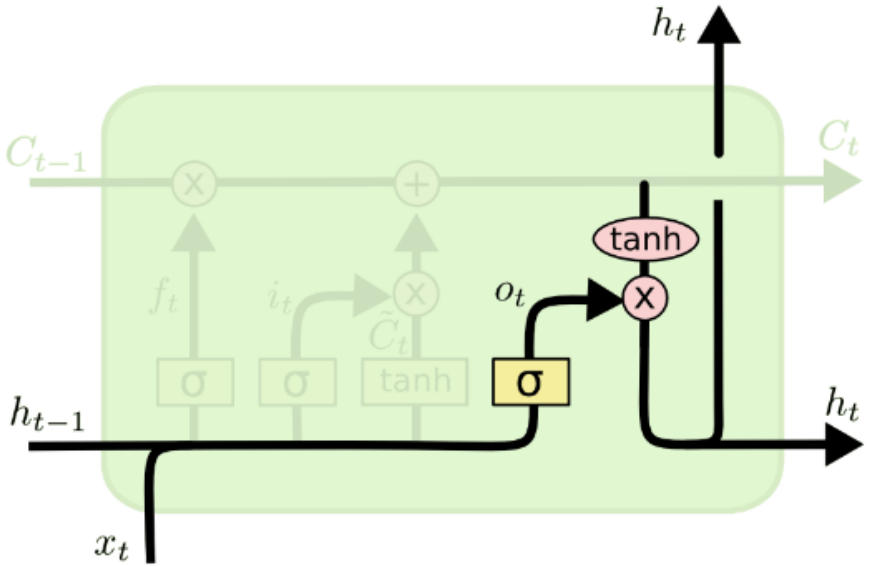
The input gate is crucial for incorporating new information into the cell state while preserving relevant information from the past.

LSTM Cell (Source — http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

LSTM memory units functioning

STEP 4: Compute the output. It will be based on a filtered version of the cell state

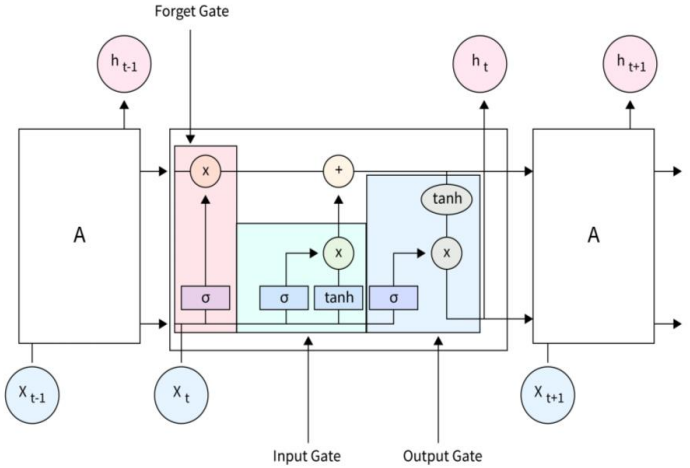
The output gate ensures that the LSTM cell provide only the relevant information to the next layer or time step.



LSTM Cell (Source — http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

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

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



SCALER Topics

By Steve Jerome Lawrence

1. Output Gate Activation: Decides which parts of the updated cell state (C_t) should be output as the hidden state (h_t).

2. Hidden State: The final output of the LSTM cell for the current time step.

Limits of LSTM networks

Gradient Explosion

- Even if LSTM can avoid the vanishing gradient, in specific cases, this network can suffer from gradient explosion due to a large gradients that can occur during backpropagation (e.g. very long input sequences/initialization of large weights)

Limited Contextual Information

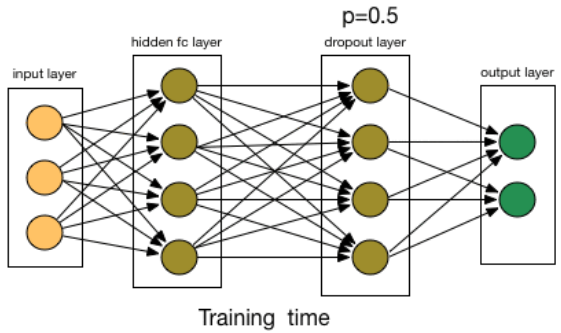
- LSTMs can capture dependencies over longer sequences than simple RNNs, but they still have a limited context window.

Complexity and Training Time

- A higher complexity can lead to longer training time and computing cost

Overfitting

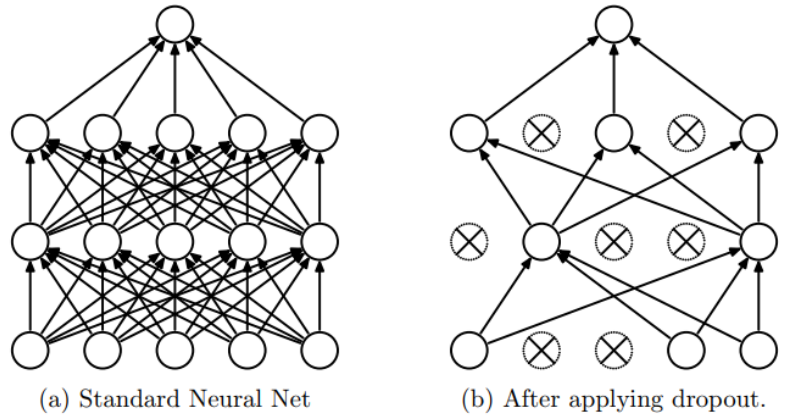
- LSTMs are prone to overfitting when training data is limited, but regularization methods like DROPOUT can help reduce this risk



https://primo.ai/index.php?title=Dropout

Source: chatbotslife

From Srivastava et al., 2014



(a) Standard Neural Net (b) After applying dropout.

DROPOUT: random exclusion of a specific % of nodes during the training

Some examples of LSTM applications in marine sciences

- Sea Surface Height (SSH) prediction (Song et al., 2021; 2020)
- Currents, sea level, and even ENSO phenomenon (Broni-Bedaiko et al., 2019; Ishida et al., 2020; Zulfa et al., 2021)
- Chlorophyll-a concentrations (Cho and Park, 2019; Rostam et al., 2021; Cen et al., 2022)
- 3D Temperature, Salinity and Steric Height from satellite observations (Buongiorno Nardelli, 2020)
- Sea Surface Temperature prediction from LSTM integrated with MLP (Jahanbakht et al., 2021)

Most of these references are taken from Tao Song et al., 2023 (Review)

THANKS!

IR0000032 – ITINERIS, Italian Integrated Environmental Research Infrastructures System
(D.D. n. 130/2022 - CUP B53C22002150006) Funded by EU - Next Generation EU PNRR-Mission 4 “Education and Research” - Component 2: “From research to business” - Investment 3.1: “Fund for the realisation of an integrated system of research and innovation infrastructures”

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani

Practice: Application of the studied networks on a real case in the Mediterranean Sea

Michela Sammartino

IR0000032 – ITINERIS, Italian Integrated Environmental Research Infrastructures System
(D.D. n. 130/2022 - CUP B53C22002150006) Funded by EU - Next Generation EU PNRR-Mission 4 “Education and Research” - Component 2: “From research to business” - Investment 3.1: “Fund for the realisation of an integrated system of research and innovation infrastructures”

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani

Two reference works

FFNN

remote sensing

MDPI

Article

An Artificial Neural Network to Infer the Mediterranean 3D Chlorophyll-*a* and Temperature Fields from Remote Sensing Observations

Michela Sammartino [1,*], Bruno Buongiorno Nardelli [2], Salvatore Marullo [1,3] and Rosalia Santoleri [1]

1 Istituto di Scienze Marine, Consiglio Nazionale delle Ricerche (ISMAR-CNR), 00133 Rome, Italy; salvatore.marullo@enea.it (S.M.); rosalia.santoleri@cnr.it (R.S.)
2 Istituto di Scienze Marine, Consiglio Nazionale delle Ricerche (ISMAR-CNR), 80133 Naples, Italy; bruno.buongiornonardelli@cnr.it
3 Agenzia Nazionale per le Nuove Tecnologie, l'Energia e lo Sviluppo Economico Sostenibile (ENEA), Centro Ricerche Frascati, 00044 Frascati, Italy
* Correspondence: michela.sammartino@artov.ismar.cnr.it; Tel.: +39-064993-4505

Received: 26 October 2020; Accepted: 11 December 2020; Published: 17 December 2020

Abstract: Remote sensing data provide a huge number of sea surface observations, but ca give direct information on deeper ocean layers, which can only be provided by sparse in situ The combination of measurements collected by satellite and in situ sensors represents one o most effective strategies to improve our knowledge of the interior structure of the ocean ecosys In this work, we describe a Multi-Layer-Perceptron (MLP) network designed to reconstruct th fields of ocean temperature and chlorophyll-*a* concentration, two variables of primary importanc many upper-ocean bio-physical processes. Artificial neural networks can efficiently model ever non-linear relationships among input variables, and the choice of the predictors is thus cruci build an accurate model. Here, concurrent temperature and chlorophyll-*a* in situ profiles and se different combinations of satellite-derived surface predictors are used to identify the optimal m configuration, focusing on the Mediterranean Sea. The lowest errors are obtained when taki

LSTM

remote sensing

MDPI

Letter

A Deep Learning Network to Retrieve Ocean Hydrographic Profiles from Combined Satellite and In Situ Measurements

Bruno Buongiorno Nardelli

Consiglio Nazionale delle Ricerche, Istituto di Scienze Marine (CNR-ISMAR), 80133 Naples, Italy; bruno.buongiornonardelli@cnr.it

Received: 5 September 2020; Accepted: 24 September 2020; Published: 25 September 2020

Abstract: An efficient combination of remotely-sensed data and in situ measurements is needed to obtain accurate 3D ocean state estimates, representing a fundamental step to describe ocean dynamics and its role in the Earth climate system and marine ecosystems. Observations can either be assimilated in ocean general circulation models or used to feed data-driven reconstructions and diagnostic models. Here we describe an innovative deep learning algorithm that projects sea surface satellite data at depth after training with sparse co-located in situ vertical profiles. The technique is based on a stacked Long Short-Term Memory neural network, coupled to a Monte-Carlo dropout approach, and is applied here to the measurements collected between 2010 and 2018 over the North Atlantic Ocean. The model provides hydrographic vertical profiles and associated uncertainties from corresponding remotely sensed surface estimates, outperforming similar reconstructions from simpler statistical algorithms and feed-forward networks.

Keywords: artificial intelligence; machine learning; deep learning; neural networks; Earth observations; ocean dynamics; sea surface temperature; sea surface salinity; altimetry; hydrography

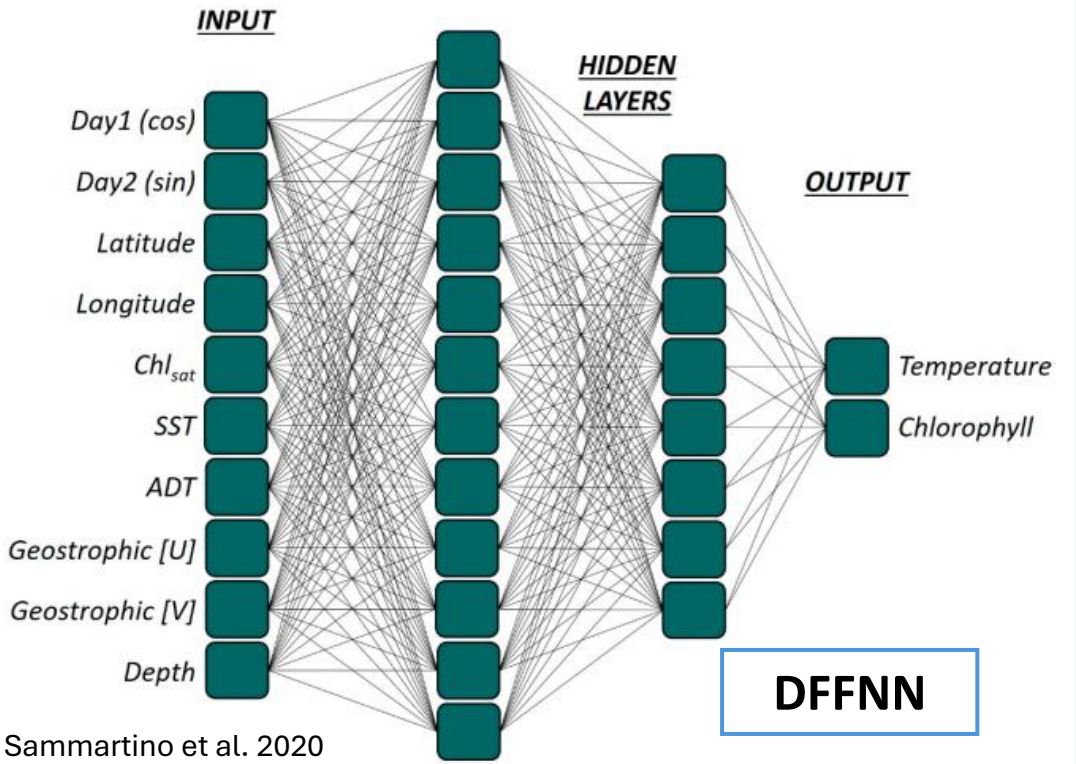
Two reference works

remote sensing

MDPI

Article

An Artificial Neural Network to Infer the Mediterranean 3D Chlorophyll-*a* and Temperature



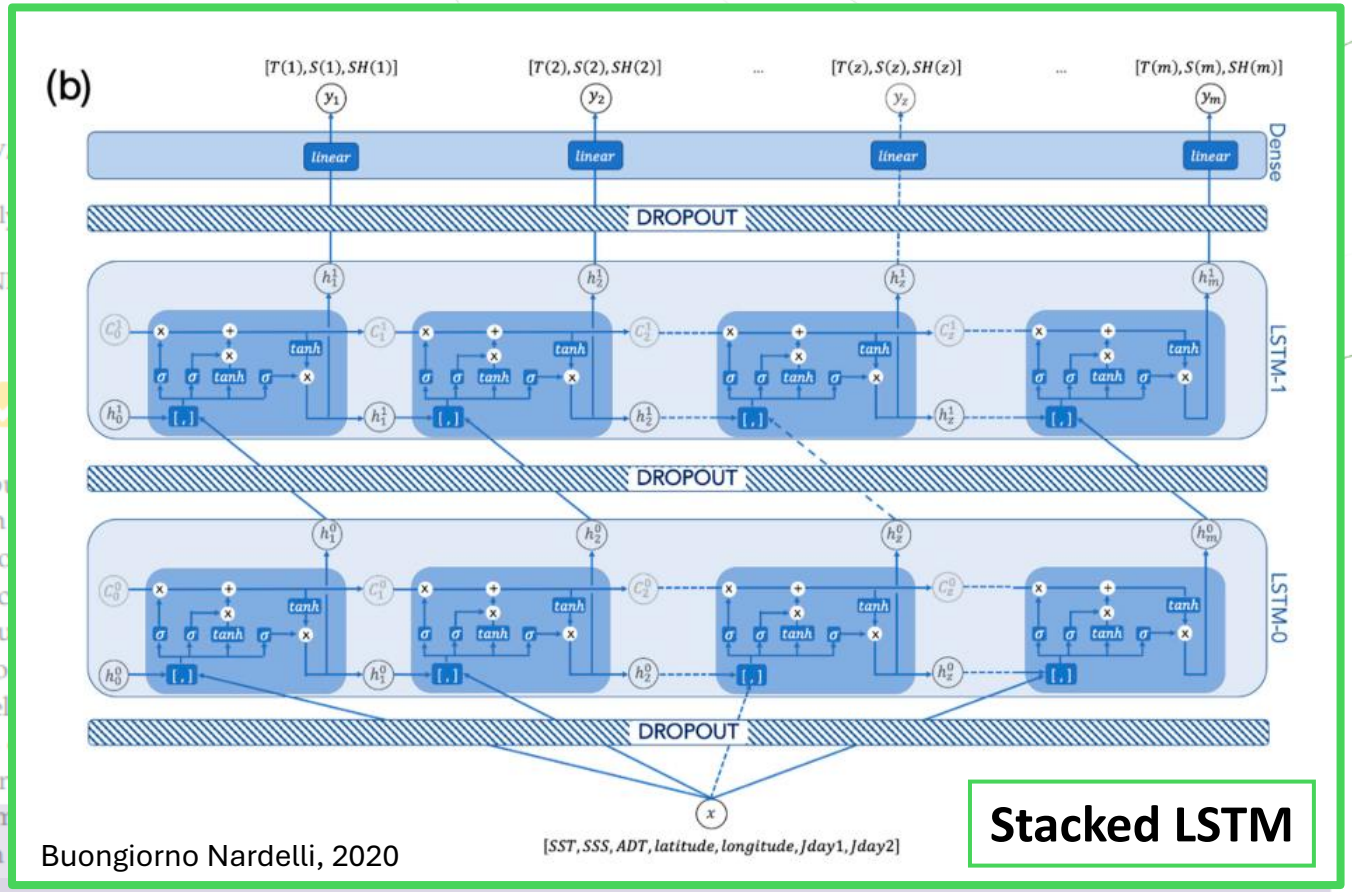
Sammartino et al. 2020

remote sensing

MDPI

Letter

A Deep Learning Network to Retrieve Ocean Hydrographic Profiles from Combined Satellite and In Situ Measurements



Buongiorno Nardelli, 2020

Creation of a dataset

Two main factors affect the prediction performance of the models:

1) dataset optimization
2) parameter optimization.

The creation of a reference quality-controlled database ensures a correct training/validation of the network,

The introduction of strange behavior in example time sequences or patterns can lead to a wrong answer.

https://blog.devgenius.io/data-quality-implementation-a-comprehensive-guide-243b6e8e4bda

Exercise

INPUTS

- Day of the year (sin/cos transformed)
- Latitude
- Longitude
- Surface Temperature, T(0)
- Surface Salintiy, S(0)
- Surface Chlorophyll, Chl(0)
- Absolute Dynamic Topography (CMEMS product)
- Geostrophic velocities (U and V)

AI model

OUTPUTS

- Vertical profile of Temperature
- Vertical profile of Chlorophyll

How to implement a neural network

1. **Data Preprocessing**
   - Split Data: Divide data into training, validation, and test sets.
   - Normalize Inputs: Scale input features to a common range (e.g., [0, 1] or [-1, 1]) to improve convergence.

2. **Define the Neural Network Architecture**
   - Choose the number of layers and neurons.
   - Select activation functions (e.g., ReLU, Sigmoid, Softmax).
   - Define input and output dimensions.

3. **Compile the Model**
   - Select a loss function (e.g., MSE for regression, Cross-Entropy for classification).
   - Choose an optimizer (e.g., Adam, SGD).
   - Define evaluation metrics.

4. **Train the Model**
   - Specify batch size and number of epochs.
   - Implement **early stopping** to prevent overfitting.

5. **Evaluate the Model**
   - Use test data to assess generalization performance.
   - Compute accuracy, loss, or other metrics.

THANKS!

IR0000032 – ITINERIS, Italian Integrated Environmental Research Infrastructures System
(D.D. n. 130/2022 - CUP B53C22002150006) Funded by EU - Next Generation EU PNRR-Mission 4 “Education and Research” - Component 2: “From research to business” - Investment 3.1: “Fund for the realisation of an integrated system of research and innovation infrastructures”

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani

References

- M. Scardi, (1996). Artificial Neural networks as empirical models for estimating phytoplankton procution. Marine Ecology Progress Series 139, 289-299.
- Scardi, M.; Harding, L.W. Developing an empirical model of phytoplankton primary production: A neural network case study. Ecol. Model. 1999, 120, 13–223.
- Mattei, F.; Franceschini, S.; Scardi, M. A depth-resolved artificial neural network model of marine phytoplankton primary production. Ecol. Model. 2018, 382, 51–62.
- S. Lek, J.L. Giraudel, J.F. Guègan (2000). Artificial Neural Networks, Springer Edition.
- Schmidt, Robin M. "Recurrent neural networks (rnns): A gentle introduction and overview." *arXiv preprint arXiv:1912.05911* (2019).
- Song, Tao, et al. "A review of artificial intelligence in marine science." *Frontiers in Earth Science* 11 (2023): 1090185.
- Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." *The journal of machine learning research* 15.1 (2014): 1929-1958.
- Sammartino, Michela, et al. "An artificial neural network to infer the Mediterranean 3D chlorophyll-a and temperature fields from remote sensing observations." *Remote Sensing* 12.24 (2020): 4123.
- Buongiorno Nardelli, Bruno. "A deep learning network to retrieve ocean hydrographic profiles from combined satellite and in situ measurements." *Remote sensing* 12.19 (2020): 3151.
- Sauzède, R.; Claustre, H.; Uitz, J.; Jamet, C.; Dall'Olmo, G.; D'Ortenzio, F.; Gentili, B.; Poteau, A.; Schmechtig, C. A neural network-based method for merging ocean color and Argo data to extend surface bio-optical properties to depth: Retrieval of the particulate backscattering coefficient. J. Geophys. Res. Oceans 2016, 121, 2552–2571.
- Sauzède, R.; Claustre, H.; Jamet, C.; Uitz, J.; Ras, J.; Mignot, A.; D'Ortenzio, F. Retrieving the vertical distribution of chlorophyll a concentration and phytoplankton community composition from in situ fluorescence profiles: A method based on a neural network with potential for global-scale applications. J. Geophys. Res. Ocean. 2015, 120, 451–470.
- Sauzède, R.; Bittig, H.C.; Claustre, H.; De Fommervault, O.P.; Gattuso, J.-P.; Legendre, L.; Johnson, K.S. Estimates of Water-Column Nutrient Concentrations and Carbonate System Parameters in the Global Ocean: A Novel Approach Based on Neural Networks. Front. Mar. Sci. 2017, 4, 128.

Web references

- https://www.baeldung.com/cs/lstm-vanishing-gradient-prevention
- https://colah.github.io/posts/2015-08-Understanding-LSTMs/
- https://alexlenail.me/NN-SVG/
- https://medium.com/@shrutijadon/survey-on-activation-functions-for-deep-learning-9689331ba092
- https://medium.com/@muradatcorvit23/unlocking-the-power-of-lstm-gru-and-the-struggles-of-rnn-in-managing-extended-sequences-05879b6899d3
- https://taketake2.com/P56_en.html
- https://www.scaler.com/topics/deep-learning/lstm/
- https://primo.ai/index.php?title=Dropout