# Machine Learning Light Hypernuclei

Isaac Vidaña, INFN Catania

EMMI Workshop "4th Workshop on Anti-Matter, Hypernuclei and Exotica Production at the LHC"

February 13th-17th 2023, Bologna (Italy)

# This work in few words

- We employ a feed-forward ANN to extrapolate at large model spaces the results of *ab-initio* hypernuclear NCSM calculations for the Λ separation energy $B_\Lambda$ of the lightest hypernuclei, obtained in accessible HO basis spaces using chiral NN, NNN & YN interactions

- Our results are in excellent agreement with those obtained using other extrapolation schemes of hypernuclear NCSM calculations, showing this that ANN is a reliable extrapolation method

Based on:

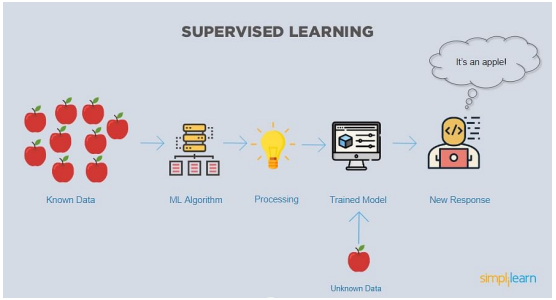
I.V. Accepted for publication in NPA (arXiv: 2203.11792v2)

# Machine Learning

**Machine Learning** is a branch of Artificial Intelligence whose scope is to *devise algorithms able to recognize patterns in previously unseen data without any explicit instructions by an external party*. Different types of ML include

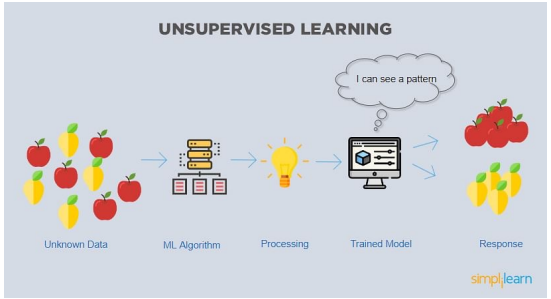
- **Supervised Learning**

  **Known input-output (feature-label) relations** are given to the machine learning algorithm to **trained** it and **infer a mapping therefrom**. Once the model is **trained based on the known data**, one can use unknown data into the model to get predictions. Used for **Classification** & **Regression** problems



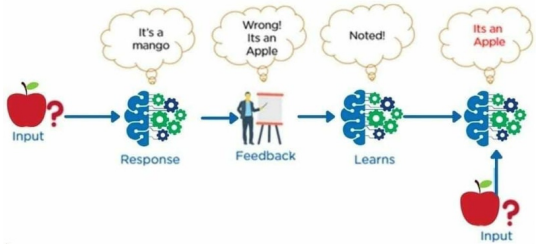

- **Unsupervised Learning**

  The output of the input training data is **unknown**. The input data is fed to the Machine Learning algorithm and is used to train the model which then is employed to **search for patterns in the data**. Used for **Clustering** & **Generation** problems




- **Reinforced learning**

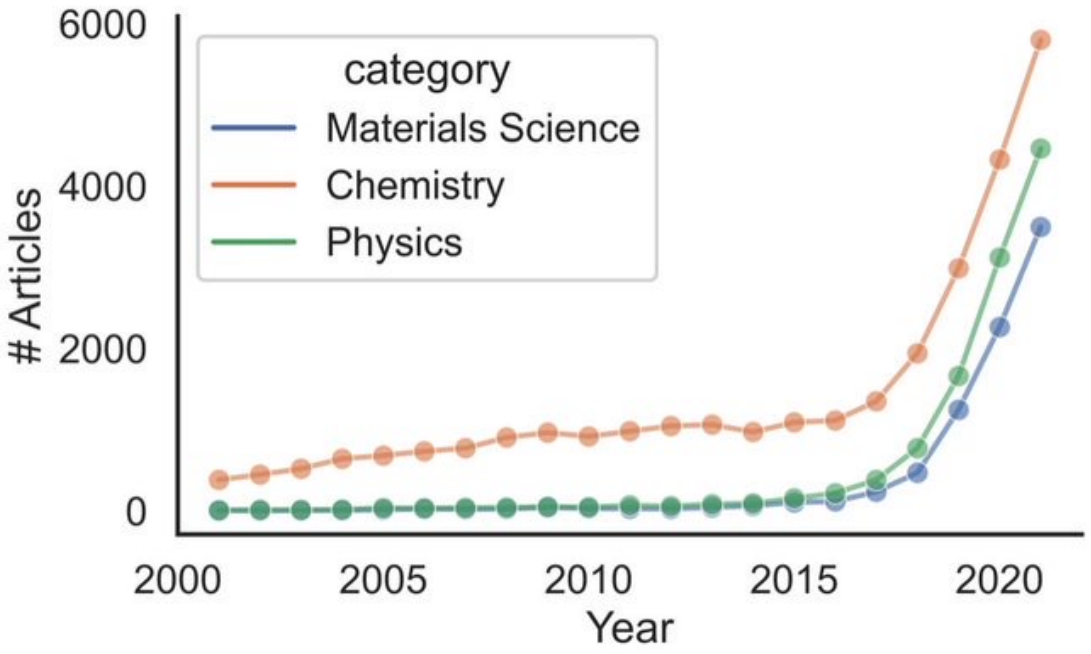
  Given a **framework of rules and goals**, an **agent** (algorithm) **learns in an interactive environment** by **trial and error** using **feedback from its own actions and experiences** and it gets **rewarded** or **punished** depending on which strategy it uses. Each **reward** reinforces the current strategy, while **punishment** leads to an adaptation of its policy. Example: **games** such as **Chess** or **Go**



# Machine Learning in Physics

Machine Learning has been applied in different areas of physics that include among others:

- Condense matter
- Statistical physics
- Cold atoms
- Quantum many-body theory
- Quantum computing
- Cosmology
- Particle physics
- Nuclear physics
- …

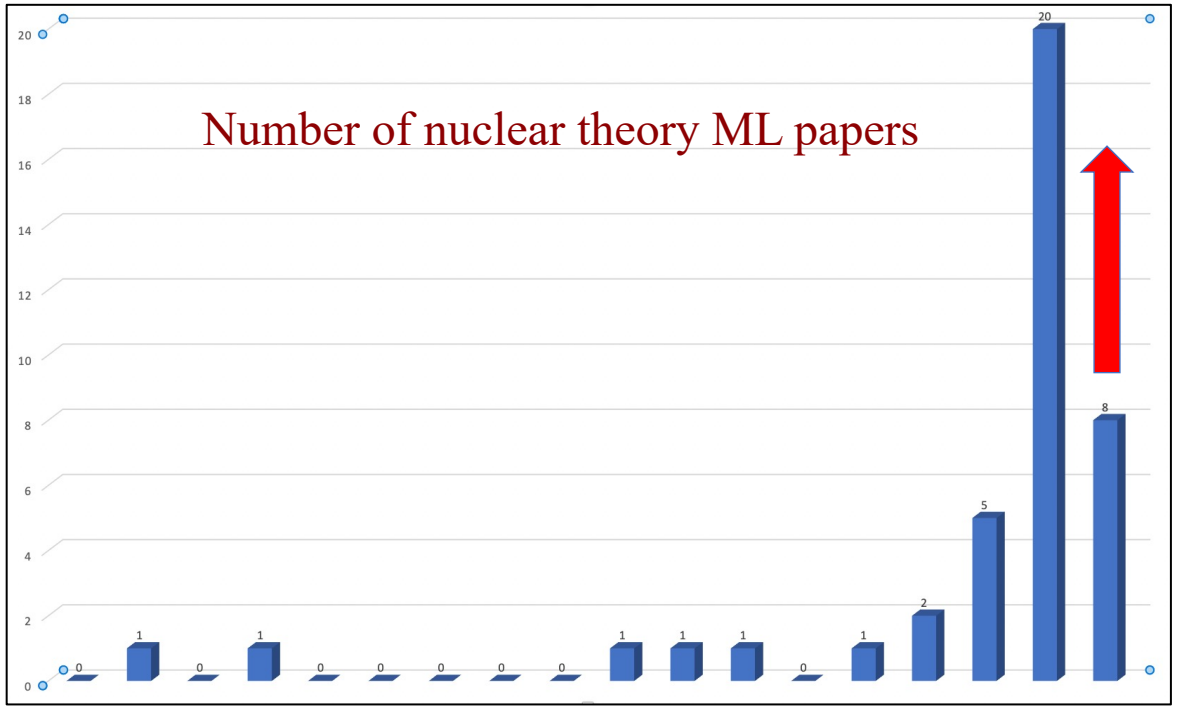



A **spectacular increase** of the number of publications related with AI or ML is observed in physical sciences in the last years

# Machine Learning Applications in Nuclear Theory

Since the pioneering work of *Gazula et al., NPA 540 1 (1992)*, who employed a **feed forward neural network to study global nuclear properties across the nuclear landscape**, Machine Learning has been used to predict

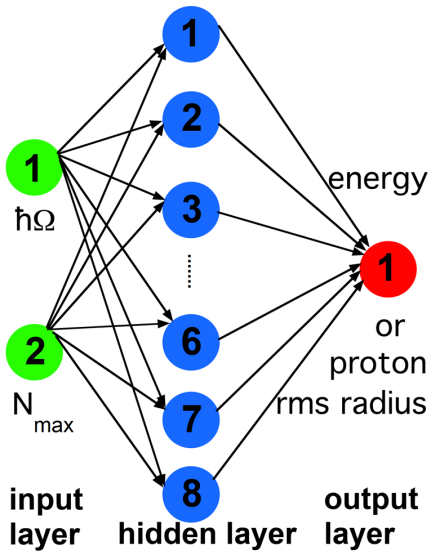
- Nuclear masses & charge radii
- α- & β-decay half-lives
- Fission yields
- Fusion reaction cross sections
- Isotropic cross-sections in proton-induced spallation reactions
- Ground and excited state energies
- Dripline locations
- The deuteron properties
- Proton radius
- Liquid-gas phase transition
- Nuclear energy density functionals
- Neutron star EoS
- The nucleon axial form factor from neutrino scattering
- Extrapolation of A-body results with ANN
- …

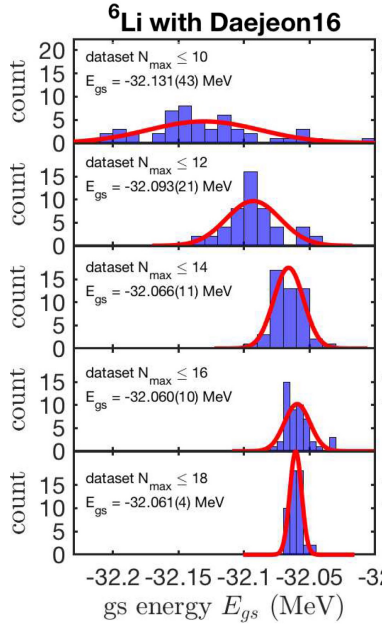


# Machine Learning Applications in Nuclear Theory

Recently, ANN have been employed to **extrapolate the results of *ab-initio* nuclear structure calculations in finite model spaces**. Particularly:

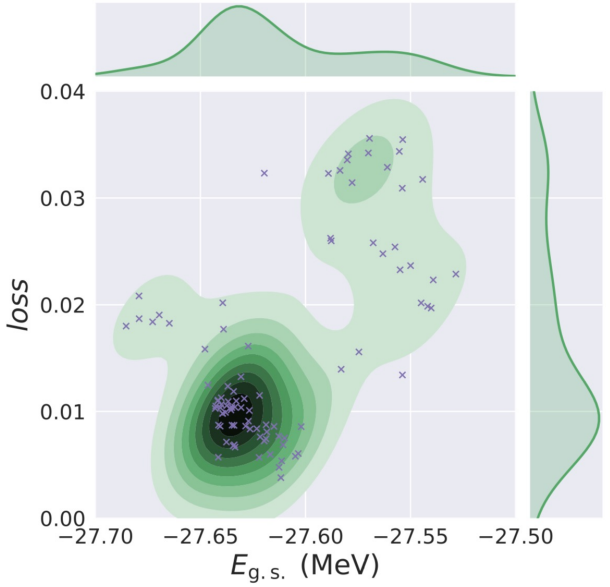
- *Negoita et al., PRC 99, 054308 (2019)* have used a feed-forward ANN method for predicting the **ground state energy and the ground state point proton root-mean-squared radius of $^{6}$Li** training the network with NCSM results, obtained in accessible harmonic oscillator (HO) basis spaces. They showed that an ANN is able to predict correctly extrapolations of the NCSM results to very large model spaces of size $N_{max} \sim 100$.



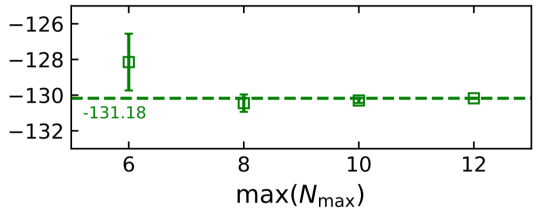




- Similarly, *Jiang et al., PRC 100, 054326 (2019)* have also employed an ANN to extrapolate the **ground state energy and radii of $^{4}$He, $^{6}$Li & $^{16}$O** computed with the NCSM and the coupled-cluster (CC) methods.

$^{4}$He with NCSM+NNLO$_{opt}$




$^{16}$O with CC+NNLO$_{opt}$

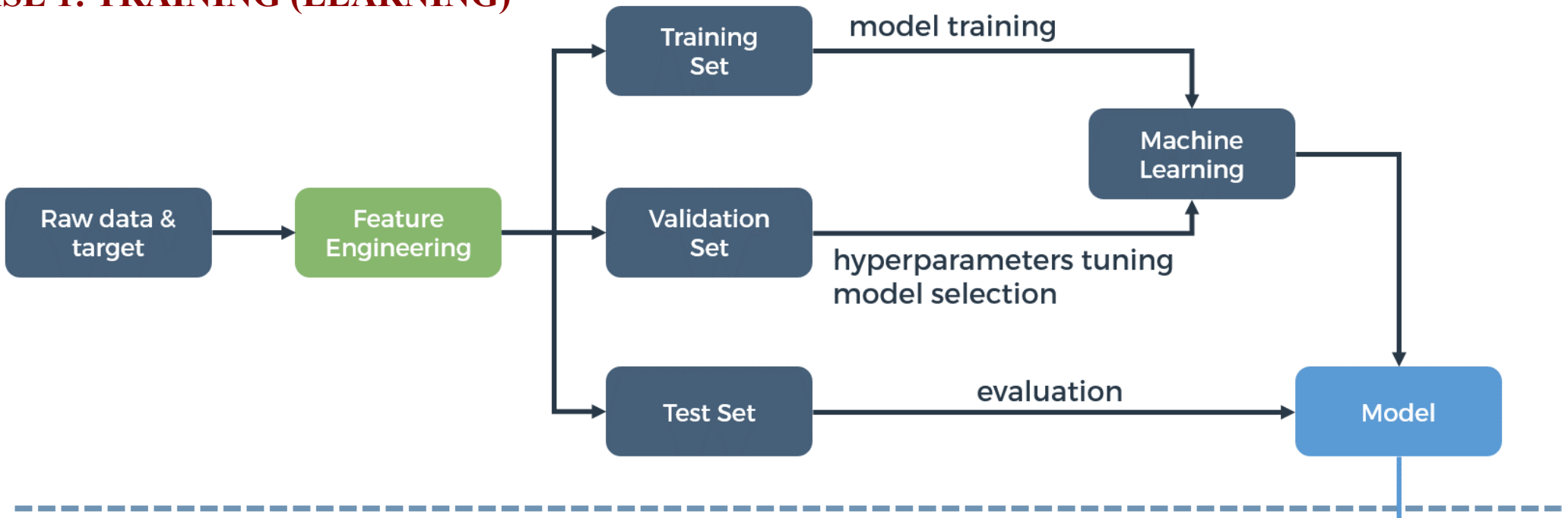



Here we follow the work of these authors, to extrapolate at large model spaces the results of *ab-initio* hypernuclear NCSM calculations for the Λ separation energy $B_{\Lambda}$ of the lightest hypernuclei

# Machine Learning Process: General Scheme

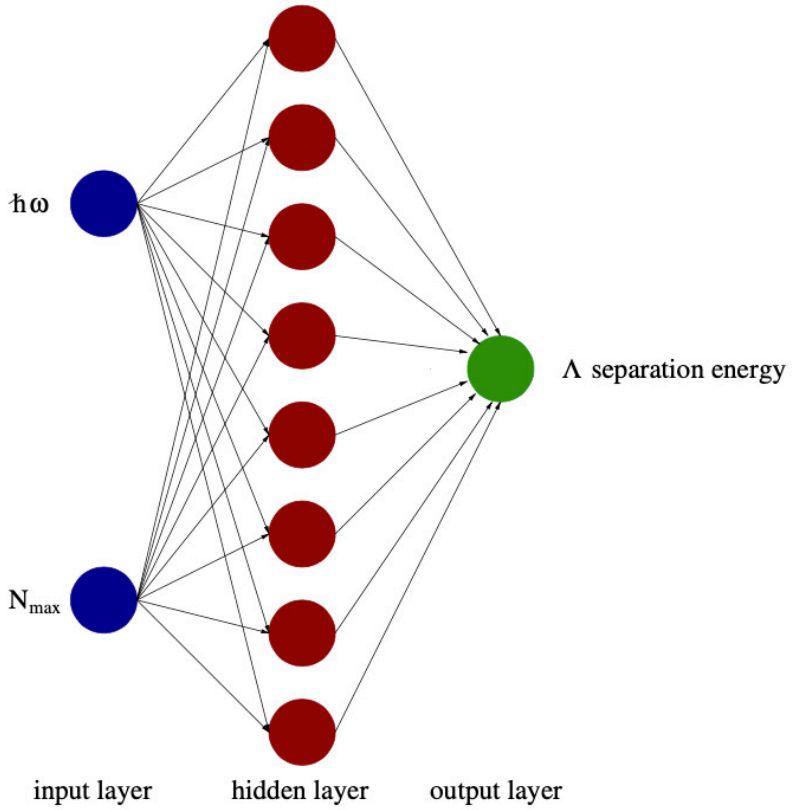
The task of **making a machine to learn** is made of **2 phases**

## PHASE 1: TRAINING (LEARNING)




## PHASE 2: PREDICTION

# Feed-forward Artificial Neural Networks




Architecture of our ANN. The input data are the HO spacings ħω and the maximum number of basis states $N_{max}$ employed in hypernuclear NCSM calculations, whereas the output is the Λ separation energy

Numerical implementation with Python libraries Scikit-learn & Keras using a TensorFlow backend

- ANNs consist of a series of layers (input, hidden & output) each one contained a certain number of interconnected neurons
- In a feed-forward ANN, neurons do not form a cycle and the data propagates sequentially from the input to the output layer through all the hidden layers
- At each one of the $N_k$ neurons $i$ of a given layer $k$, the set of input data $\{a_j^{(k-1)}\}$ from the $N_{k-1}$ neurons $j$ of the layer *k-1* is transformed into

$$a_i^{(k)} = f\left(\sum_{j=1}^{N_{k-1}} \omega_{ij}^{(k)} a_j^{(k-1)} + b_i^{(k)}\right)$$

  - $f(z)$: activation function, introduces non-linearities on the neural network that enable it to capture complex non-linear relationships in the dataset. In this work we use a sigmoid activation function $f(z)=(e^z+1)^{-1}$
  - $\omega_{ij}^{(k)}, b_i^{(k)}$: fitting parameters of the ANN. Are the *weights* of the connections between the neurons of the two adjacent layers *k-1* & *k*, and the activation offset (*bias*) of each neuron of the layer *k*. The total number of fitting parameters $n_p$ is

$$n_p = \sum_{k=0}^{L-2} (N_k + 1) N_{k+1}$$

# The Learning Process of an ANN

The **learning (or training) process of an ANN** involves the **minimization of a loss** (also called **cost** or **error**) **function** (which compares the desired ouput (target) and the predicted one by the ANN) in order to **obtain the optimal set of fitting parameters** (**weights** and **biases**) $(\boldsymbol{W}, \boldsymbol{b}) \equiv \left\{\boldsymbol{\omega}_{\boldsymbol{ij}}^{(\boldsymbol{k})}, \boldsymbol{b}_{\boldsymbol{i}}^{(\boldsymbol{k})}\right\}$ of the network.

**Choice of the loss function**

In general, the **choice of the loss function depends on the type of problem** one is solving with a neural network. In this work we are solving a regression-type problem and we chose the *mean squared error* (MSE), a common choice for this kind of problems

$$L(\boldsymbol{W}, \boldsymbol{b}) = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i(\boldsymbol{W}, \boldsymbol{b}) - y_i)^2$$

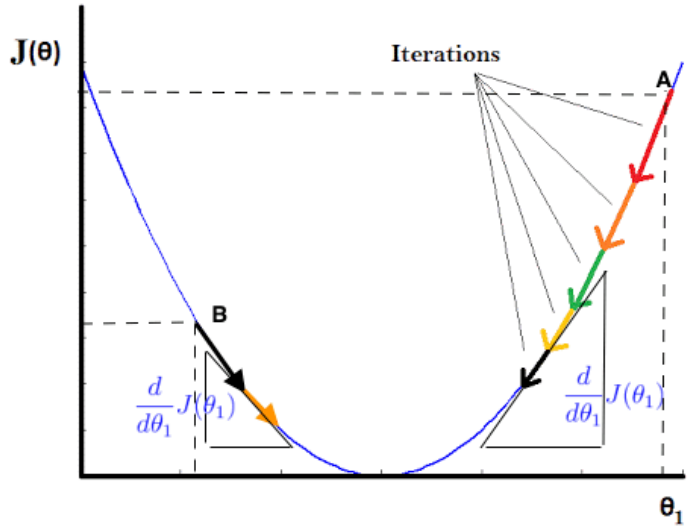
- $N$ : number of points used in the minimization procedure
- $\hat{y}_i(\boldsymbol{W}, \boldsymbol{b}) \equiv \boldsymbol{a}_{\boldsymbol{i}}^{(\boldsymbol{L-1})}$ : prediction of the ANN
- $y_i$ : actual output oof the input data

# Gradient Descent

Most of the **minimization algorithms** employed in ANN are based on the so-called **gradient descent algorithm**

Idea: Take repeated steps in the opposite direction of the gradient since the **gradient of a multi-variable function $J(\vec{\theta})$ defines the direction of its maximum increase.** One starts with a guess $\vec{\theta}_o$ and considers the sequence $\vec{\theta}_1, \vec{\theta}_2, \vec{\theta}_3, \cdots$ according to

$$\vec{\theta}_{n+1} = \vec{\theta}_n - \eta \vec{\nabla} J(\vec{\theta}_n), \text{ with } \eta > 0$$



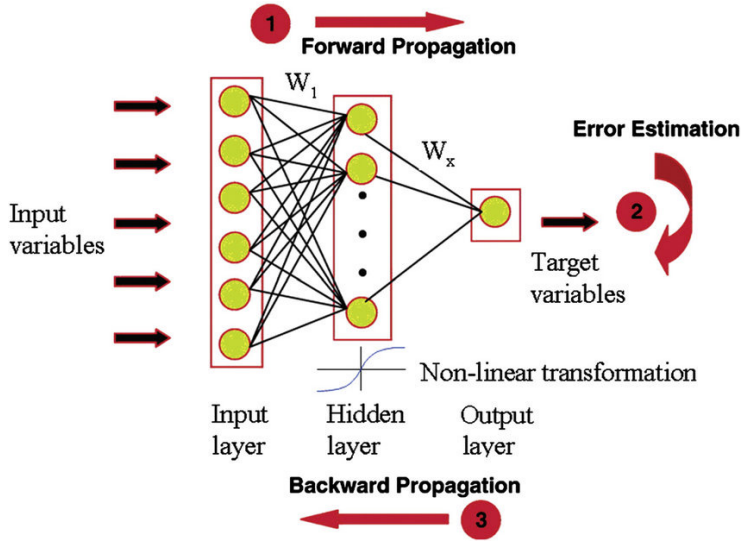

With this idea in mind the **weights $\omega_{jk}^l$ & biases $b_j^l$ of the network are updated at each iteration** according to:

$$\omega_{jk}^l \to \omega_{jk}^l - \eta \frac{\partial L}{\partial \omega_{jk}^l}, \qquad b_j^l \to b_j^l - \eta \frac{\partial L}{\partial b_j^l}$$

where $\eta$ is the so-called **learning rate,** one of the **hyperparameters** of the newtwork, that controls how fast or how slow the network parameters are updated

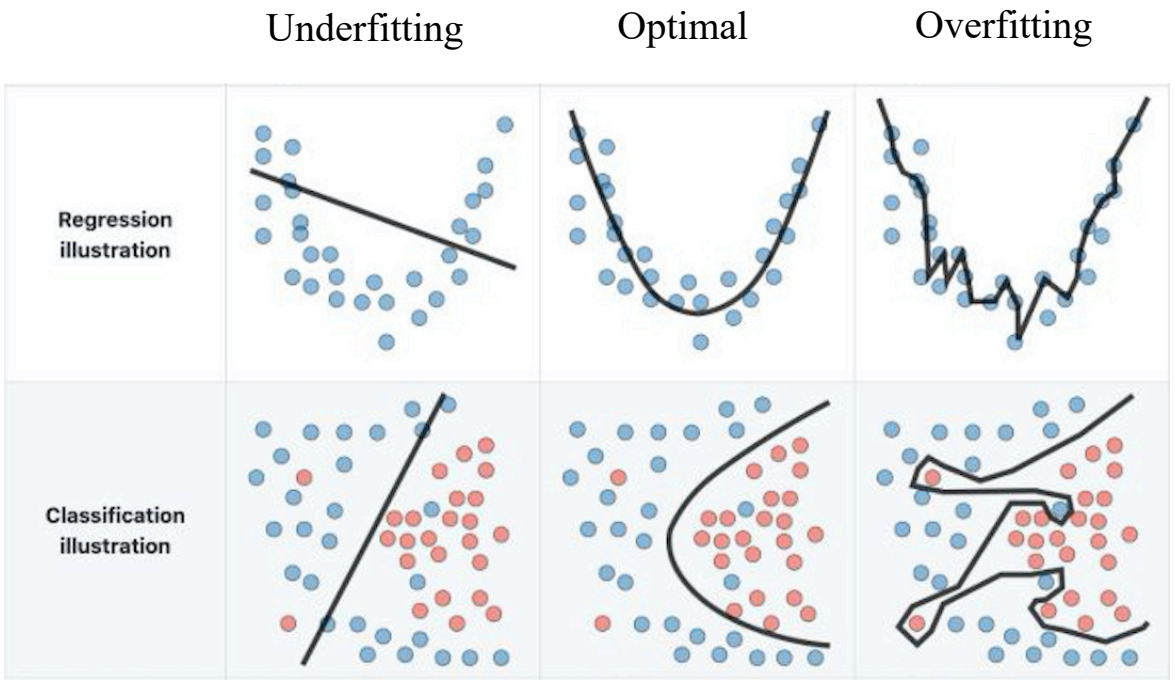
# The Backpropagation Algorithm

Backpropagation is a method used to **calculate in an efficient way** the **gradient** $\frac{\partial L}{\partial \omega_{jk}^{l}}$, $\frac{\partial L}{\partial b_{j}^{l}}$ of the **loss function** and **adjust the connection weights & the biases** to reduce the error during the learning process.




- **Input** $x$: set the corresponding activation $a_j^1 = x_j$ for each neuron *j-th* of the input layer
- **Feedforward**: for each layer $l = 2, 3, \cdots, L$ compute $z_j^l = \sum_k \omega_{jk}^l \, a_k^{l-1} + b_j^l$ and $a_j^l = f(z_j^l)$
- **Output error** $\delta_j^L$: compute the error of each neutron of the last layer L, $\delta_j^L = \frac{\partial L}{\partial a_j^L} f'(z_j^L) = \frac{\partial L}{\partial \hat{y}_j} f'(z_j^L)$
- **Backpropagate the error**: for each layer $l = L-1, L-2, \cdots, 2$ compute $\delta_j^l = \sum_k \delta_k^{l+1} \omega_{kj}^{l+1} f'(z_j^l)$
- **Gradient of the loss function**: $\frac{\partial L}{\partial \omega_{jk}^l} = \delta_j^l a_k^{l-1}$ , $\frac{\partial L}{\partial b_j^l} = \delta_j^l$
- **Update the weights & biases**: $\omega_{jk}^l \to \omega_{jk}^l - \eta \frac{\partial L}{\partial \omega_{jk}^l}$ , $b_j^l \to b_j^l - \frac{\partial L}{\partial b_j^l}$
- **Repeat till convergence is achieved**

# Overfitting of an ANN




- A major issue in the development of an ANN is overfitting (also known as overtraining), which basically means that the network, due to its high flexibility to approximate complex non-linear functions, tries to fit the data entirely and ends up memorizing all the data patterns.

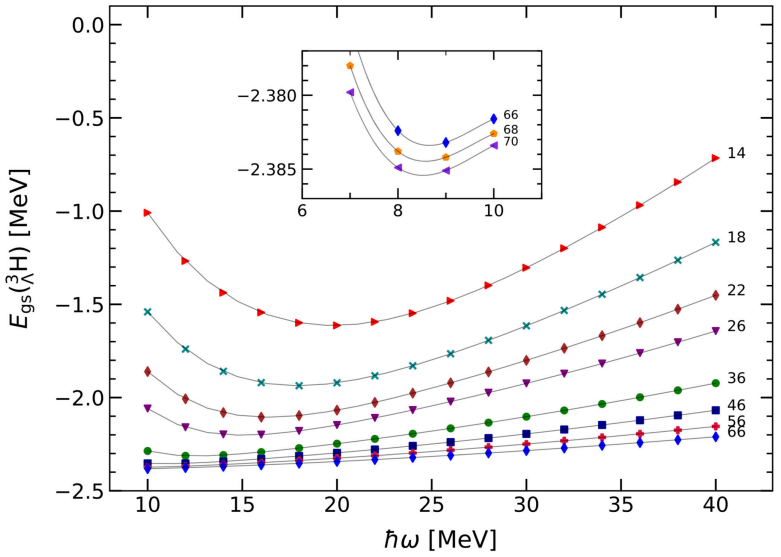
- Due to overfitting the predictability of the network on testing data becomes questionable

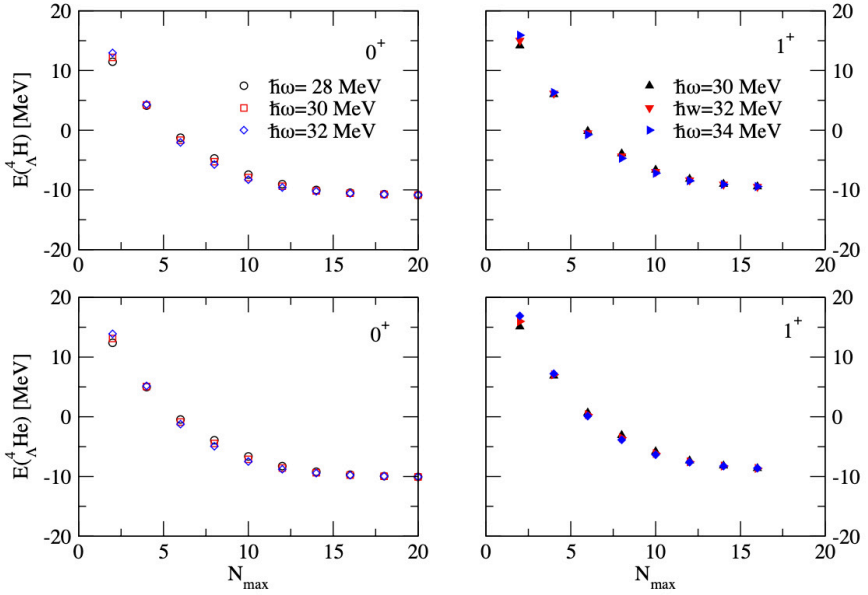
- Strategies to avoid overfitting include among others:
  - early stopping of the training: stops the training process once the model performance stops improving on the validation dataset
  - dropout: reduce overfitting by dropping randomly neurons from the neural network during training in each iteration

- In addition to these which can be used together, overfitting can be reduced by:
  - enlarging the input dataset (specially in those case where the input dataset is not large enough)
  - adding noise to the input dataset making the network less able to memorize data patterns since they change randomly during the training

# Input Dataset

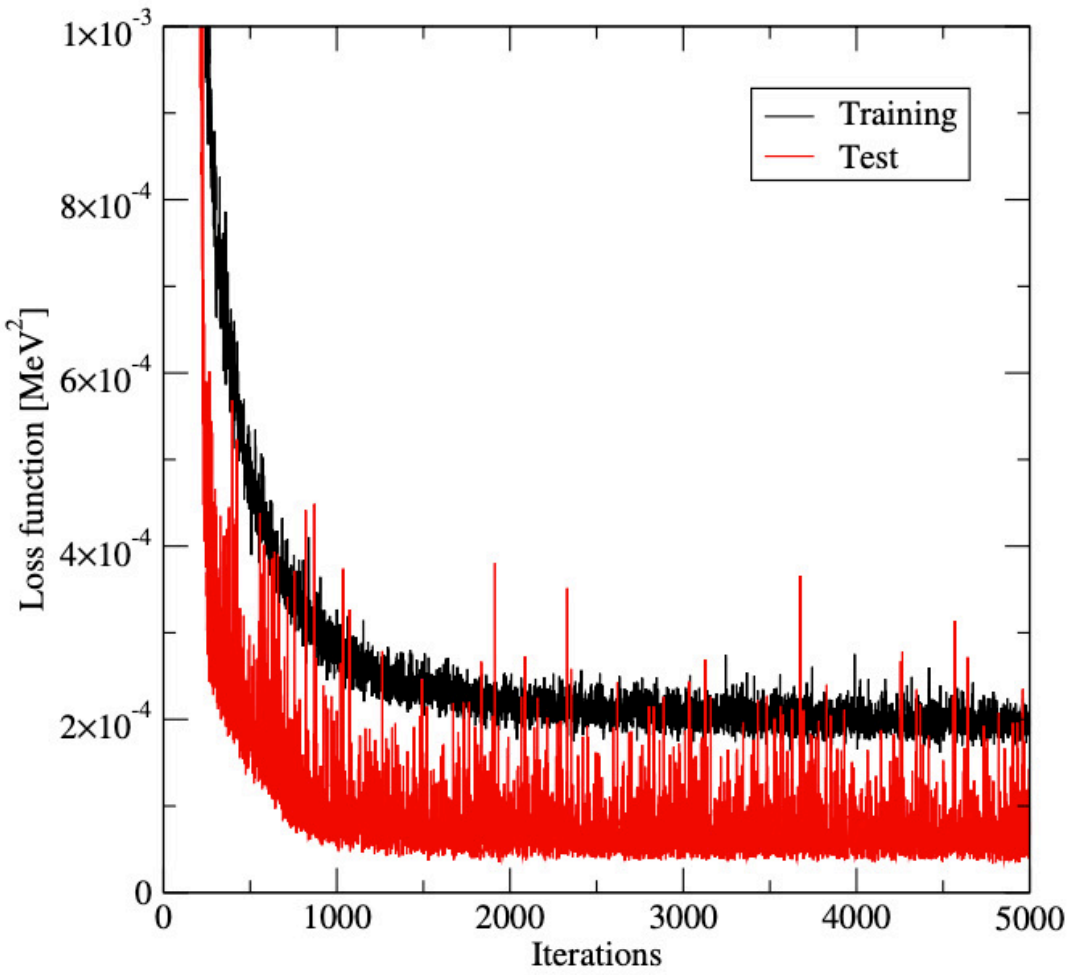
We employ as input dataset the hypernuclear NCSM results of Gazda *et al.* (PRC 97 (2018) 064315, Few-Body Syst. 62 (2021) 94) for the Λ separation energy of ${}^{3}_{\Lambda}$H, ${}^{4}_{\Lambda}$H & ${}^{4}_{\Lambda}$He obtained with chiral NN & NNN interactions at N$^3$LO and N$^2$LO, respectively both with a regulator cutoff of 500 MeV, and YN potentials at LO with a cutoff of 600 MeV




- Due to the small size of the original input dataset to avoid overfitting we have:
  - enlarged it by performing a cubic interpolation in the HO spacing $\hbar\omega$ at each given value of $N_{max}$
  - introduced a Gaussian noise in the enlarged input dataset during the training of the network
- We use the 80% (10 % of it used for validation) of the enlarged input dataset to train the network and leave the 20% of it for testing

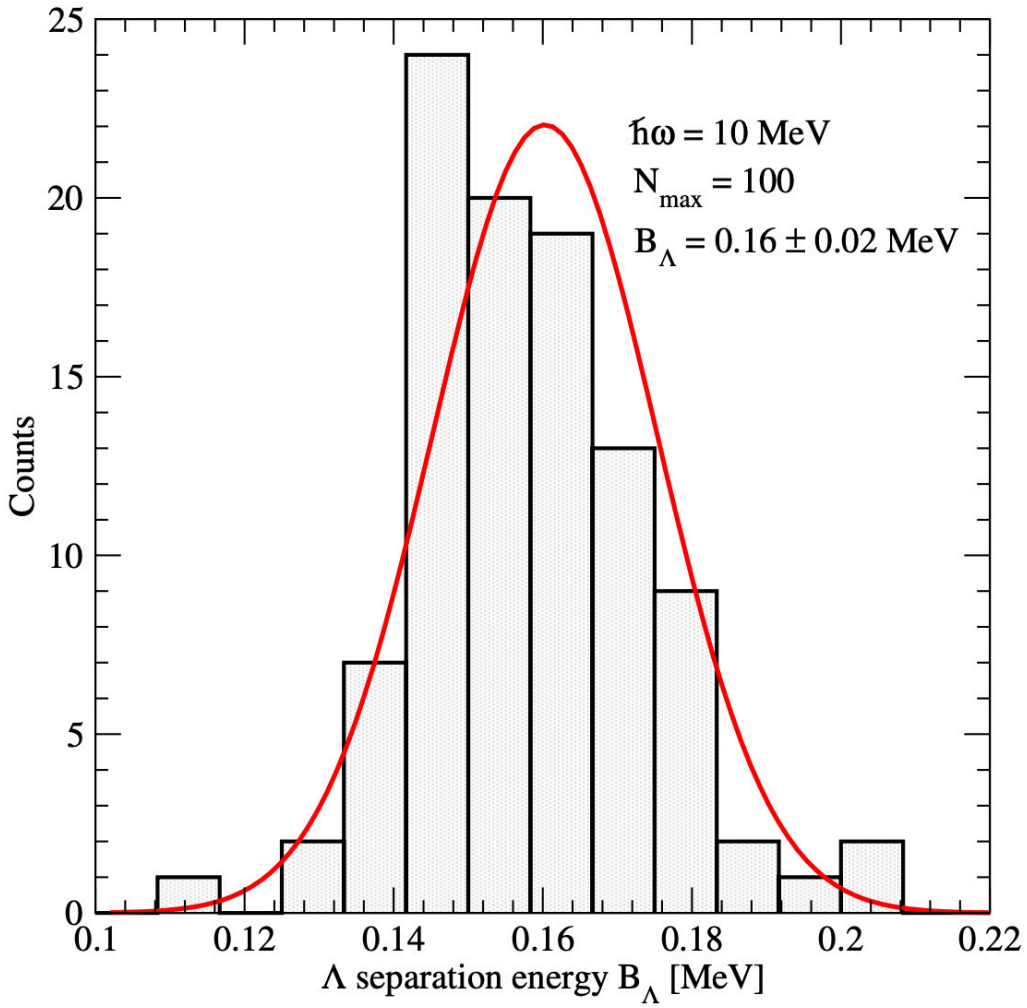
# Performance of the ANN




To illustrate the performance of the network we show the loss function $L(\boldsymbol{W}, \boldsymbol{b})$ of the training & test datasets as a function of the number of iterations in the calculation of the $\Lambda$ separation energy of the ground state of ${}^{3}_{\Lambda}\mathrm{H}$

- Very fast decrease during the first 500 iterations becoming (on average) essentially constant at about 1000 iterations and above it
- The loss function of the test dataset is smaller that that of the training one, indicating that overfitting has been substantially reduced
- Similar good performance for ${}^{4}_{\Lambda}\mathrm{H}$ and ${}^{4}_{\Lambda}\mathrm{He}$

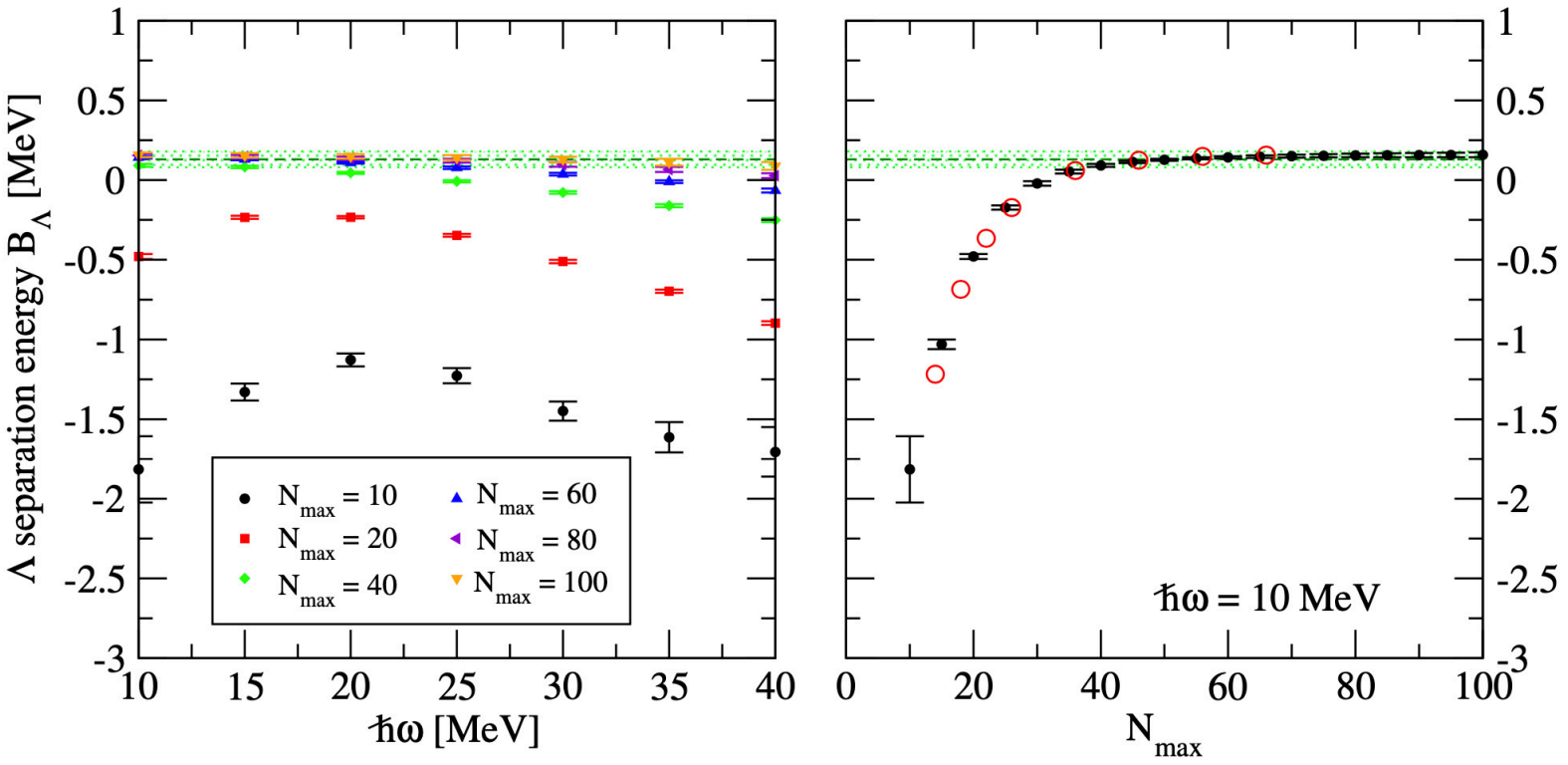
# A General Comment




- A typical run of an ANN starts with random values of the weights & biases of the network.
- The random inizialization of the weights & biases is not accidental but an important feature of the network training that introduces a certain degree of stochasticity that reduces the risk that during the optimization process of the network parameters it gets stuck in a local minimum
- Consequently, different runs of the ANN lead to slightly different results as seen in the figure
- Because of this, for each hypernucleus, we have performed 100 independent runs of the ANN and taken the average and the standard deviation of all these runs ad the predictions of the network and their corresponding error, respectively

Statistical distribution of the results for the Λ separation energy $B_\Lambda$ of the ground state of ${}^{3}_{\Lambda}\mathrm{H}$ predicted by 100 independent runs of the ANN for an HO spacing $\hbar\omega$= 10 MeV and a model space size $N_{max}$ = 100. The continuous line shows a Gaussian fit of the histogram.

# Λ separation energy of the ground state of ${}^{3}_{\Lambda}H$



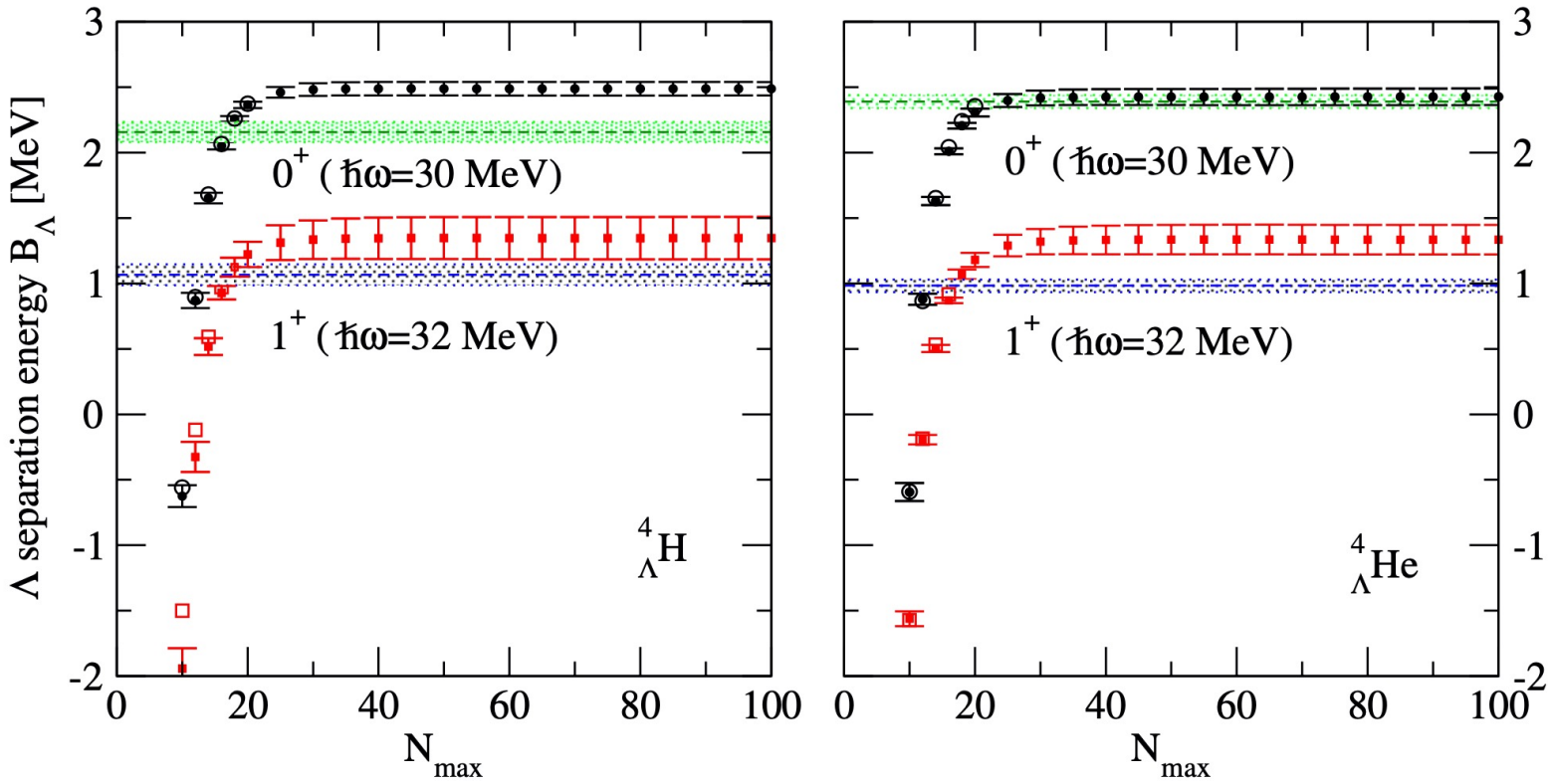

Open circles in the right panel show the NCSM results used for the training of the ANN

- Slow convergence due to the extremely weak binding energy of ${}^{3}_{\Lambda}H$
- Considerably reduction of the $B_\Lambda$ dependence with $\hbar\omega$ with the increase of $N_{max}$
- Good extrapolation to the experimental result for large values of the model space size $N_{max}$

ANN prediction for $N_{max} = 100$

$$B_\Lambda\left({}^{3}_{\Lambda}H\right) = 0.16 \pm 0.02 \text{ MeV}$$

# Λ separation energy of the $0^+$ & $1^+$ states of ${}^4_\Lambda$H & ${}^4_\Lambda$He



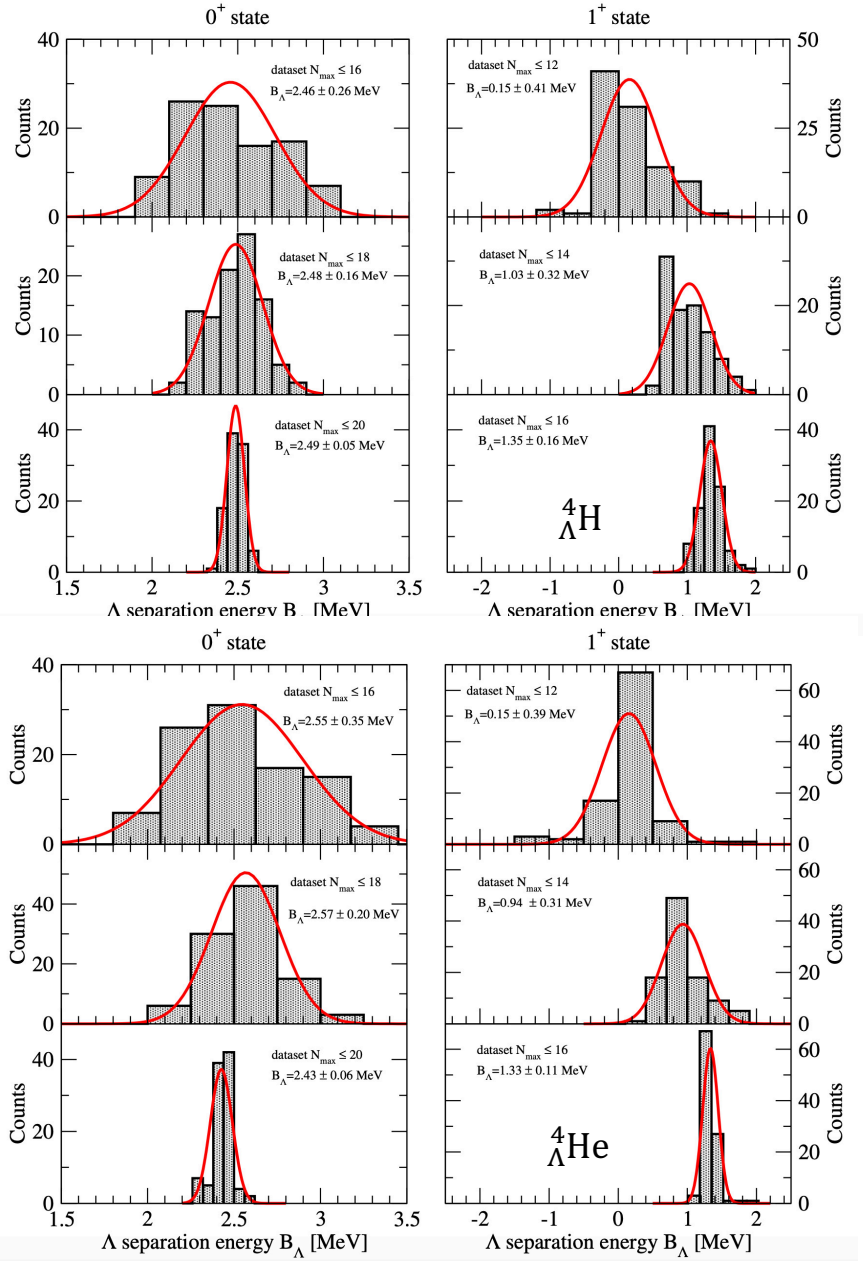

ANN prediction for $N_{max} = 100$

$B_\Lambda\left({}^4_\Lambda H(0^+)\right) = 2.49 \pm 0.05$ MeV  $B_\Lambda\left({}^4_\Lambda He(0^+)\right) = 2.43 \pm 0.06$ MeV

$B_\Lambda\left({}^4_\Lambda H(1^+)\right) = 1.35 \pm 0.16$ MeV  $B_\Lambda\left({}^4_\Lambda He(1^+)\right) = 1.33 \pm 0.11$ MeV

- Faster convergence than in the ${}^3_\Lambda$H case. Good convergence already for $N_{max} \gtrsim 25$
- Well extrapolation of the ANN prediction for the $0^+$ state of ${}^4_\Lambda$He to the experimental value
- ANN prediction for the $0^+$ & $1^+$ states of ${}^4_\Lambda$H & $1^+$ of ${}^4_\Lambda$He off of the experiment by about 0.3 MeV. This should not be attributed to the performance of the ANN but to the Hamiltonian used & the symmetries assumed in the NCSM calculations
- Charge symmetry breaking (CSB) in these two A=4 mirror hypernuclei not explained because CSB effects are not included in the NCSM calculations used to train the ANN. Therefore, the ANN cannot account for them

# Convergence of the extrapolated results of the $0^+$ & $1^+$ states at $N_{max}$=100




To further check the convergence of the extrapolated value of $B_\Lambda$ for the $0^+$ & $1^+$ states of ${}^{4}_{\Lambda}$H & ${}^{4}_{\Lambda}$He at $N_{max}$=100 we show the statistical distributions of the results of 100 independent runs of the ANN for several choices of the maximal value of $N_{max}$ taken into account in the training dataset

We observe that:

- The dispersion of the ANN predictions becomes narrower & narrower when the maximal value of $N_{max}$ included in the training dataset increases
- Successive extrapolates are consistent consistent with the previous ones within the given uncertainties

From this one can conclude that the extrapolated values of $B_\Lambda$ for the two states of both hypernuclei show a rather well convergence in terms of the maximal value of $N_{max}$ included in the training dataset, being this an indication that ANN are a reliable method to extrapolate the results of hypernuclear NCSM calculations to large model spaces

# Comparison with other extrapolation schemes

The main goal of this work is mainly focused on discerning whether ANN is a reliable scheme to extrapolate NCSM results at larger model spaces rather than its accuracy on reproducing the experimental results. To such end we compare our results with the extrapolated ones of Htun *et al.*, FBS 62 (2021) 94 [1] & Wirth *et al.*, PRC 97 (2018) 064315 [2]

| Hypernucleus | ANN Prediction | Extrapolated results of [1] & [2] | Experimental Vaue |
|---|---|---|---|
| ${}^{3}_{\Lambda}\mathrm{H}$ (g.s.) | $0.16 \pm 0.02$ | 0.158 [1] | $0.13 \pm 0.05$ |
| ${}^{4}_{\Lambda}\mathrm{H}(0^{+})$ | $2.49 \pm 0.05$ | $2.48 \pm 0.04$ [2] | $2.157 \pm 0.077$ |
| ${}^{4}_{\Lambda}\mathrm{H}(1^{+})$ | $1.35 \pm 0.16$ | $1.40 \pm 0.28$ [2] | $1.067 \pm 0.08$ |
| ${}^{4}_{\Lambda}\mathrm{He}(0^{+})$ | $2.43 \pm 0.06$ | $2.45 \pm 0.04$ [2] | $2.39 \pm 0.05$ |
| ${}^{4}_{\Lambda}\mathrm{He}(1^{+})$ | $1.33 \pm 0.11$ | $1.34 \pm 0.28$ [2] | $0.984 \pm 0.05$ |

As it is seen our results are in excellent agreement with those of Htun *et al.* & With *et al.*, confirming this that ANN is a reliable method to extrapolate results of hypernuclear NCSM calculations to large model spaces

# Take home message

**Summary & Conclusions**

- We employ a feed-forward ANN to extrapolate at large model spaces the results of *ab-initio* hypernuclear NCSM calculations for the Λ separation energy $B_\Lambda$ of the lightest hypernuclei, obtained in accessible HO basis spaces using chiral NN, NNN & YN interactions
- The overfitting problem is avoided by enlarging the size of the input dataset & by introducing a Gaussian noise during the training process of the neural network
- We find that a network with a single hidden layer of eight neurons is enough to extrapolate correctly the value of $B_\Lambda$ to model spaces of size $N_{max}$=100
- Our results are in excellent agreement with those obtained using other extrapolation schemes of hypernuclear NCSM calculations, showing this that ANN is a reliable extrapolation method

**Future Perspectives**

- Compare the ANN results with those obtained with other extrapolation schemes such that of infrared extrapolation (IR) where the model space parameter $\hbar\omega$ & $N_{max}$ are translated into an IR length scale $L_{eff}$ and an ultraviolet (UV) cutoff scale $\Lambda_{UV}$
- Analyze the ANN performance, particularly the convergence of the ANN results, on heavier hypernuclei

- ✧ You for your time & attention
- ✧ Benjamin, Francesca, Jürgen & Stefania for their kind inviation to participate in this workshop & their support
- ✧ Specially Daniel Gazda for providing me with the NCSM results used to train the ANN


"This project has received funding from the Helmholtz Institute Mainz and the European Union's Horizon 2020 research and innovation programme under grant agreement No 824093"