



Machine learning the deuteron: new architectures and uncertainty quantification

J. Rozalén Sarmiento^{1,2,a} , J. W. T. Keeble³, A. Rios^{1,2,3}

¹ Departament de Física Quàntica i Astrofísica, Universitat de Barcelona (UB), c. Martí i Franquès 1, 08028 Barcelona, Spain

² Institut de Ciències del Cosmos (ICCUB), Universitat de Barcelona (UB), c. Martí i Franquès 1, 08028 Barcelona, Spain

³ Department of Physics, University of Surrey, Guildford GU2 7XH, UK

Received: 23 October 2023 / Accepted: 7 February 2024

© The Author(s) 2024

Abstract We solve the ground state of the deuteron using a variational neural network ansatz for the wavefunction in momentum space. This ansatz provides a flexible representation of both the S and the D states, with relative errors in the energy which are within fractions of a per cent of a full diagonalisation benchmark. We extend the previous work on this area in two directions. First, we study new architectures by adding more layers to the network and by exploring different connections between the states. Second, we provide a better estimate of the numerical uncertainty by taking into account the final oscillations at the end of the minimisation process. Overall, we find that the best performing architecture is the simple one-layer, state-connected network. Two-layer networks show indications of overfitting, in regions that are not probed by the fixed momentum basis where calculations are performed. In all cases, the errors associated to the model oscillations around the real minimum are larger than the stochastic initialization uncertainties.

1 Introduction

Artificial neural networks (ANNs) have become routine computational tools in a wide range of scientific domains [1, 2]. Nuclear physics is no exception, and the number of machine learning (ML) tools is increasing steadily [3–10]. An interesting and relatively recent use of ANNs is the solution of quantum mechanical problems, including in the many-body domain [11, 12]. These methods use ANNs as an ansatz for the many-body wavefunction, and employ ready-made ML tools to minimise the energy of the system [13, 14]. The first attempts in condensed matter [15, 16] were quickly followed by advances both within [17] and outside the field [5], including quantum chemistry [18, 19]. A good example in this direction is FermiNet [20], which uses an inherently antisymmetric wavefunction as an ansatz in a variational approach to solve the many-electron problem.

The methodology that we use to solve the deuteron here is based on the same variational philosophy and stems from Ref. [21], where we used a one-layer variational ANN ansatz to describe the wavefunction of the deuteron (the only two-body nuclear bound state) in momentum space. This yields excellent results in comparison with benchmark solutions for the energies and the wavefunctions, quantified through fidelity measures. Moreover, we estimate the out-of-sample uncertainty of the model by using several random initialisations in the minimisation problem.

The assessment of uncertainties in variational ANN (vANN) is particularly relevant to pinpoint the fundamental limitations of this methodology. There are several systematic errors that are important for vANN solutions to the Schrödinger equation. For instance, network architectures play a key role in determining the convergence properties of the network. Similarly, for a fixed architecture, vANN widths and depths change the expressivity of the network, and are thus fundamental ingredients in providing faithful representations of continuous wavefunctions. One should explore these systematic uncertainties extensively before reaching conclusions about any intrinsic shortcomings of the vANN method itself.

The deuteron is an excellent test bed for such studies. It is a relatively simple, one-body-like problem, which already includes the complexity of the strong force. Moreover, dealing with a wavefunction with two different angular momentum states adds an additional level of sophistication. In this manuscript, we extend the elementary approach of Ref. [21] in two different directions. First, we look at somewhat more complex ANN architectures to describe the wavefunction. This yields insight on the importance of the network architecture and should ultimately provide information on the limitations of the method beyond a specific architecture. Second, we introduce a novel way to compute the out-of-sample error. In particular, we look at oscillations in the energy, which is our cost function, around the global minimum. These oscillations are an additional source of uncertainty, associated to the minimisation process. The analysis of such errors provides a new insight into uncertainty quantification for variational ANN solvers.

^a e-mail: jrozalen@ub.edu (corresponding author)

This paper is structured in the following manner. In Sect. 2, we briefly go over the methods used to approach the deuteron problem with ANNs. Section 3 is devoted to the analysis of the numerical results, including their uncertainties. Finally, in Sect. 4, we draw a series of conclusions and comment on ways in which our work could be expanded and improved.

2 Methodology

The approach that we use to solve the problem is variational. The ANN plays the role of a wavefunction ansatz, which we denote $|\psi_{\text{ANN}}^{\mathcal{W}}\rangle$. $\mathcal{W} = \{\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \mathbf{W}^{(3)}, \mathbf{b}\}$ is a set formed by network weights, $\mathbf{W}^{(i)}$, and biases \mathbf{b} . In a variational setting, the total energy of the system is the loss function, which reads,

$$E^{\mathcal{W}} = \frac{\langle \psi_{\text{ANN}}^{\mathcal{W}} | \hat{H} | \psi_{\text{ANN}}^{\mathcal{W}} \rangle}{\langle \psi_{\text{ANN}}^{\mathcal{W}} | \psi_{\text{ANN}}^{\mathcal{W}} \rangle}. \quad (1)$$

The deuteron is the simplest nuclear two-body problem. We solve its structure in momentum space, where we can directly access nuclear interaction matrix elements. We further separate the centre of mass from the relative coordinates [22]. Consequently, the wavefunction of the system depends only on the relative momentum coordinate, \vec{q} . Working in momentum space allows us to skip numerically costly derivatives on the wavefunctions when computing the kinetic energy [9].

We can further reduce the dimensionality of the deuteron problem via a partial wave expansion, thus separating the dependence on the absolute value of the momentum q from its dependence on angles. For the deuteron, the tensor component of the strong interaction admixes the S ($L = 0$) and the D ($L = 2$) components of the ground-state wavefunction. The network will consequently have two different outputs, one for each state. The potential energy term used to compute Eq. (1) mixes these two states in a non-trivial way, so they are not completely independent from each other in the variational setting. We choose the N3LO Entem–Machleidt nucleon–nucleon force [23], which is easily implemented in our code as a momentum-dependent potential.

To implement the problem computationally, we use a one-dimensional grid in q with $N_q = 64$ points. This grid is used to compute the energy integrals in Eq. (1) via Gaussian quadrature. The loss function is obtained from a global integral, the energy and is always computed with the aforementioned set of momentum grid points. As we shall see later, this can create some issues in deeper models. To efficiently capture the low-momentum structure and the high-momentum tails of the wavefunction, we use a mesh that densely covers the region of low momenta but is sparse in regions of high momenta. We first distribute N_q Gauss–Legendre points x_i between 0 and 1, and then extend them tangentially using the transform $q_i = \frac{q_{\text{max}}}{\tan \frac{\pi}{2} x_N} \tan \frac{\pi}{2} x_i$, with $i = 1, \dots, N_q$ and

$q_{\text{max}} = 500 \text{ fm}^{-1}$. In this set-up, the overlap $\langle \psi_{\text{targ}}^L | \psi_{\text{ANN}}^L \rangle$, as well as all analogous integrated quantities, is discretised as follows:

$$\begin{aligned} \langle \psi_{\text{targ}}^L | \psi_{\text{ANN}}^L \rangle &= 4\pi \int_0^\infty dq q^2 \psi_{\text{targ}}^{L*}(q) \psi_{\text{ANN}}^L(q) \\ &\approx 4\pi \sum_{i=1}^{N_q} w_i q_i^2 \psi_{\text{targ}}^{L*}(q_i) \psi_{\text{ANN}}^L(q_i), \end{aligned} \quad (2)$$

where w_i are the integration weights associated to the tangentially transformed Gauss–Legendre mesh. Note that because the angular dependence has been removed via a partial wave expansion, all the integrals involve a radial q^2 term. Note also that all the wavefunctions are real-valued.

In order to estimate the errors in our quadratures introduced by the discretisation of the mesh, we have explored how different values of N_q can affect computed values. For instance, computing the energy, Eq. (1), with $N_q = 48, 64$ and 80 introduces variations of the order of keV. However, throughout the text we will use only $N_q = 64$ and perform all comparisons with this number of points.

In the following, we present results corresponding to different ANN architectures for the two wavefunctions, $\psi_{\text{ANN}}^L(q)$, with $L = 0$ and 2. Following Ref. [21], we take three identical steps to train all the networks. These three steps are common to most vANN problems in quantum mechanics [16]. First, we initialise network parameters to uniform random values in the ranges $\mathbf{W}^{(1)} \in [-1, 0)$, $\mathbf{W}^{(2)} \in [0, 1)$ and $\mathbf{b} \in [-1, 1)$. We choose a Sigmoid activation function, $\sigma(x)$, which provided marginally better results than the Softplus function in Ref. [21]. We also explore the use of different activation functions $\sigma(x)$ in Sect. 3.4.

Second, we train the ANN to mimic a target function that bears a certain degree of physical meaning. We choose a functional form $\psi_{\text{targ}}^L(q) = q^L e^{-\frac{\xi^2 q^2}{2}}$, which has the correct low-momentum asymptotic behaviour and a reasonable real-space width, $\xi = 1.5 \text{ fm}$. In this pretraining step, we use the overlap,

$$K^L = \frac{\langle \psi_{\text{targ}}^L | \psi_{\text{ANN}}^L \rangle^2}{\langle \psi_{\text{targ}}^L | \psi_{\text{targ}}^L \rangle \langle \psi_{\text{ANN}}^L | \psi_{\text{ANN}}^L \rangle}, \quad (3)$$

to compute a total cost function, C , defined as,

$$C = (K^S - 1)^2 + (K^D - 1)^2. \quad (4)$$

Equation (4) is zero if, and only if, the two overlaps $K^S = K^D = 1$. We choose the optimiser RMSprop [1], which dynamically adapts a global learning rate locally for all the network parameters. Training a medium-sized ANN with $N_{\text{hid}} \approx 20$ hidden nodes usually takes about 10^4 iterations. We point out that this pretraining step is not strictly necessary to achieve a correct energy minimisation, but it helps in guaranteeing a successful energy minimisation with fewer epochs.

In the third and final step, we define a new cost function: the energy, as given in Eq. (1). We compute it with the momentum quadrature described above. The network is then trained to minimise E^W for 2.5×10^5 epochs. The exact ground-state energy can be computed in the very same momentum quadrature via exact diagonalisation, and its value is $E_{\text{GS}} = -2.2267$ MeV. We use this value as a benchmark. We note that the kinetic energy has diagonal contributions in the $L = 0$ and 2 states, but the potential term mixes both states. We perform all calculations in double precision. We use the PyTorch library [24, 25], which is particularly useful due to its automatic differentiation capabilities [26]. A full version of the code is available on GitHub [27].

2.1 Architectures

The vANN procedure is relatively general, and can be applied to any feed-forward ANN. In the initial exploration of deuteron wavefunctions of Ref. [21], a minimal approach was chosen deliberately to explore the potential and the limitations of the method in the simplest possible model. We used a single-layer ANN with an increasing number of hidden nodes in order to assess systematics for a fixed architecture. Here, we take a step further and assess the impact of increasing the depth of the ANN which leads to different architectures.

To this end, we introduce four different ANN architectures, which we show in the four panels of Fig. 1. All our networks have a single input, q , and two outputs, $|\Psi_{\text{ANN}}^S\rangle$ and $|\Psi_{\text{ANN}}^D\rangle$. We distinguish between different configurations depending on how they connect to the two final states. On the one hand, in fully connected networks, which we dub “state-connected” (sc) networks, all the parameters contribute to both outputs (see the top panels, Fig. 1 a, b). On the other hand, “state-disconnected” (sd) networks (bottom panels, Fig. 1 c, d) provide independent parameters for each state. One may naively expect sd networks to provide more flexibility (and hence better variational properties) for the two wavefunctions.

In addition to the state dependence of the network, we also explore architecture systematics in terms of depth. We implement both sc and sd networks with one (left panels) and two (right panels) layers. In the following, we use the acronyms nsc and nsd ($n = 1, 2$), with n the number of layers, to refer to the four architectures explored in this work. We stress that in Ref. [21] we only looked at the 1sc architecture. The motivation to explore other architectures and, in particular further depths, is the widely held belief that deep ANNs outperform their shallow single-layered counterparts [28].

In the initial layer of all four architectures, we introduce both weights $\mathbf{W}^{(1)}$ and biases \mathbf{b} . In contrast, all the second (and/or third) layers only have weights, $\mathbf{W}^{(2)}$ ($\mathbf{W}^{(3)}$). For clarity and completeness, we now provide the functional form of the four networks. For the one-layer 1sd configurations, the wavefunction ansatz reads,

$$\psi_{\text{ANN}}^L(q) = \sum_{i=1}^{N_{\text{hid}}/2} W_{i,L}^{(2)} \sigma(W_{i,L}^{(1)} q + B_{i,L}). \tag{5}$$

The corresponding 1sc contribution is obtained by setting $N_{\text{hid}}/2 \rightarrow N_{\text{hid}}$ and using L -independent $\mathbf{W}^{(1)}$ and \mathbf{B} parameters. When 2 layers are included, in contrast, the sd ansatz is more complex, and becomes the sum of sums of two nested activation functions,

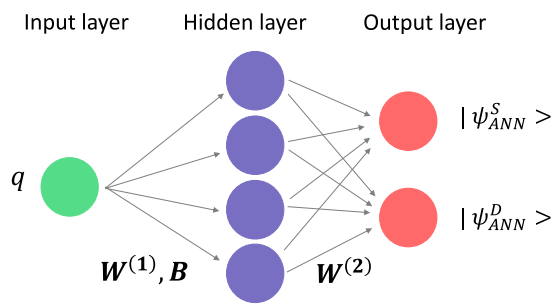
$$\psi_{\text{ANN}}^L(q) = \sum_{i=N_{\text{hid}}/2+1}^{N_{\text{hid}}} W_{i,L}^{(3)} \sigma \left(\sum_{j=1}^{N_{\text{hid}}/2} W_{j,L}^{(2)} \sigma(W_{j,L}^{(1)} q + B_{j,L}) \right). \tag{6}$$

One can again easily manipulate the previous expression to obtain a 2sd wavefunction.

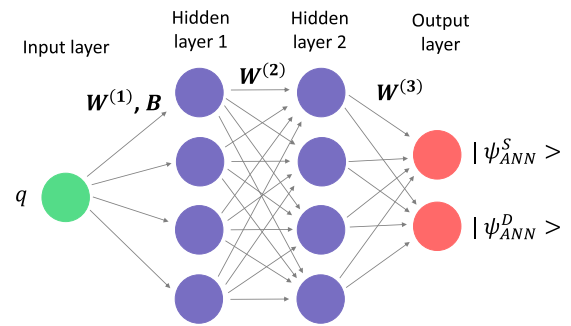
The relation between the total number of hidden neurons, N_{hid} , and the total number of variational parameters, N , is different for each architecture. The relations $N(N_{\text{hid}})$ are shown in the captions of Fig. 1 for each network configuration. One-layer networks have a linear $N(N_{\text{hid}})$ relation, whereas the relation for two-layer networks is quadratic. In other words, for the same N_{hid} , two-layer networks will usually involve a much larger number of parameters than one-layer models, and one may expect overfitting to become an issue. Whereas for the 1sc architecture the number of hidden nodes is unrestricted so long as $N_{\text{hid}} > 2$, the 2sc architecture must have $N_{\text{hid}} > 4$ with N_{hid} even. Likewise, the 1sd network must have an even N_{hid} . In contrast, for the 2sd network, N_{hid} is a multiple of 4.

2.2 Learning process

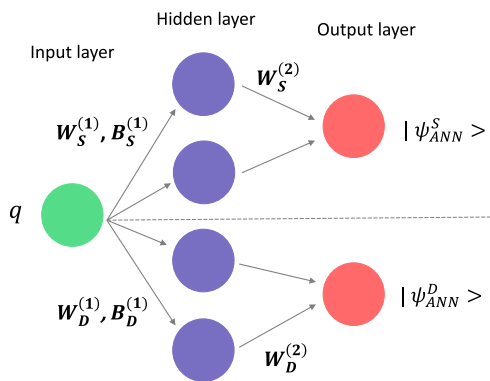
We minimise the energy cost function using RMSprop [1] with hyperparameters $\epsilon = 10^{-8}$ and a smoothing constant $\alpha = 0.9$. We set the learning rate to 10^{-2} and explore the N_{hid} dependence using the values $N_{\text{hid}} \in \{20, 30, 40, 60, 80, 100\}$ (in the 2sc architecture we change $N_{\text{hid}} = 30 \rightarrow N_{\text{hid}} = 32$). To explore the full flexibility of the wavefunction ansätze, we optimise the networks by minimising the overlap loss function for 2×10^3 epochs of pretraining, followed by 2.5×10^5 epochs of energy minimisation. Rather than doing this a single time, we use 20 different random initialisations and display mean values and (Bessel-corrected) standard



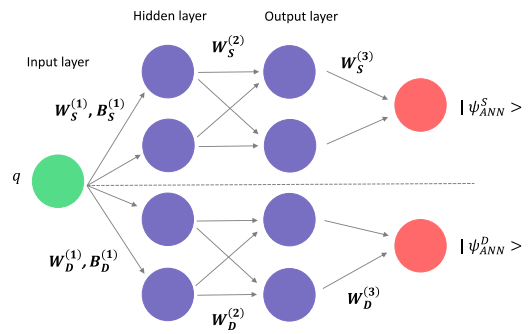
(a) State-connected network with a single hidden layer. For this network, the number of parameters is $N = 4N_{\text{hid}}$, with N_{hid} denoting the number of hidden nodes. Note that the first layer includes a bias.



(b) State-connected network with two hidden layers, with $N = \frac{1}{8}N_{\text{hid}}(N_{\text{hid}} + 12)$ parameters.



(c) State-disconnected network with a single hidden layer, with $N = 3N_{\text{hid}}$ parameters.



(d) State-disconnected network with two hidden layers, with $N = \frac{1}{4}N_{\text{hid}}(N_{\text{hid}} + 8)$ parameters.

Fig. 1 Neural network architectures used in this work. Panels (a) and (b) show state-connected architectures. Panels (c) and (d) show state-disconnected architectures

deviations obtained with all these runs.¹ With this, we explore the out-of-sample bias of the network and we attempt to draw generic conclusions for the network architecture, rather than for a single, specific model.

In fact, we perform 150 minimisations for each architecture (other than for a specific 2sd model, as we discuss in the following). Not all of these runs converge or, if they do, they may not converge to meaningful values, close enough to the minimum. Figure 2 shows the rate of convergence of different architectures as a function of N_{hid} . This is obtained as the ratio of the number of converged models, N_{con} , to the total set of model initialisations, $N_{\text{tot}}=150$, $r = N_{\text{con}}/N_{\text{tot}}$. As a selection criterion to compute N_{con} , we define converged models as those that provide a final energy within the range $E \in (-2.220, -2.227)$ MeV. We set $N_{\text{tot}} = 150$ to guarantee that all configurations have a minimum of $N_{\text{con}} = 20$ converged models. The only exception is the 2sd architecture with $N_{\text{hid}} = 20$, which has a very small convergence rate. This architecture requires $N_{\text{tot}} = 1100$ runs to get $N_{\text{con}} = 20$ converged models. Whenever more than 20 initialisations lead to converged results, we randomly pick 20 states to have representative, but also homogeneous, statistics across the N_{hid} domain.

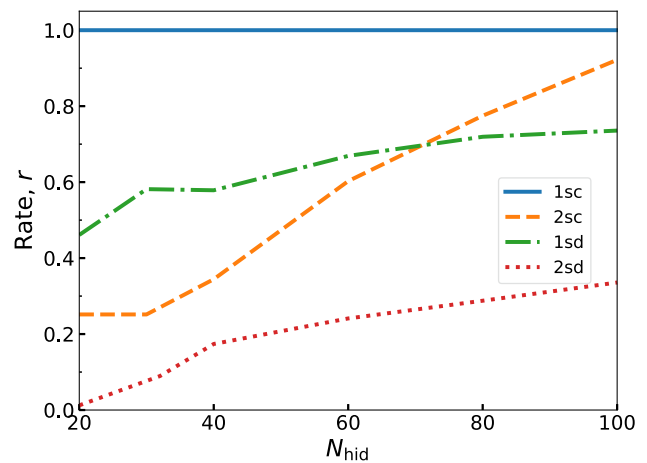
We stress the fact that these rates are obtained from initialisations with a pretraining step. With no pretraining, all models show slower convergence rates.² We take this difference as an indication of the fact that pretraining is effective in providing a physical representation of the ANN wavefunction. Initially non-pretrained networks may occasionally lead to accurate results, but are likely to require many more energy minimisation epochs.

Going back to Fig. 2, we find that the one-layer configurations provide better convergence rates than their two-layer counterparts up to $N_{\text{hid}} \approx 60$, at which point the 2sc architecture is marginally better than 1sd. The 1sc network (solid line) has a perfect

¹ This is to be compared to the 50 runs shown for the 1sc architecture in Ref. [21].

² This includes the 1sc configuration, which leads to a perfect convergence rate if pretraining is included.

Fig. 2 Convergence rate of the four network architectures as a function of the number of hidden neurons. The rate (vertical axis) is defined as the ratio between the number of models that converge and the total number of trained models, N_{con}/N_{tot}



convergence rate. The 1sd architecture (dash-dotted line) provides a convergence rate of $r > 50\%$ relatively independent of the value of N_{hid} . We attribute these similarities to the fact that both pretraining and training are more easily done with a single-layer ANN.

In contrast, two-layer networks start with a relatively low convergence rate. In architecture 2sc, we have $r \approx 25\%$ for $N_{hid} = 20$, and the rate increases in a relatively steady and linear fashion up to $r \approx 80 - 90\%$ for $N_{hid} = 100$. In architecture 2sd, we find even lower convergence rates, $r \approx 1\%$ for $N_{hid} = 20$, and a relatively slow increasing tendency in terms of N_{hid} up to $r \approx 20\%$. The stark differences in convergence rates between one and two-layer ANNs may be due to a variety of factors. On the one hand, network parameter initialisation may be an issue. Network parameters in different domains than the ones we prescribe at the moment may provide better starting points. On the other hand, we fix the total number of energy minimisation epochs and the current rate may just be an indication that smaller two-layer networks simply take, with RMSprop, a larger number of epochs to converge. Finally, two-layer networks may be problematic in terms of the Sigmoid activation function, which is prone to a vanishing gradients issue [29, 30]. This should be accentuated in deeper (as opposed to shallow one-layer) networks. Along these lines, having few neurons increases the probability that all of them become frozen, which may explain why low N_{hid} models have a harder time converging. These arguments suggest that an activation function like Softplus or ReLU may work better with two-layer architectures. We have carried out tests with different activation functions that show an improvement in convergence rates for two-layer networks, as shown in Table 3.

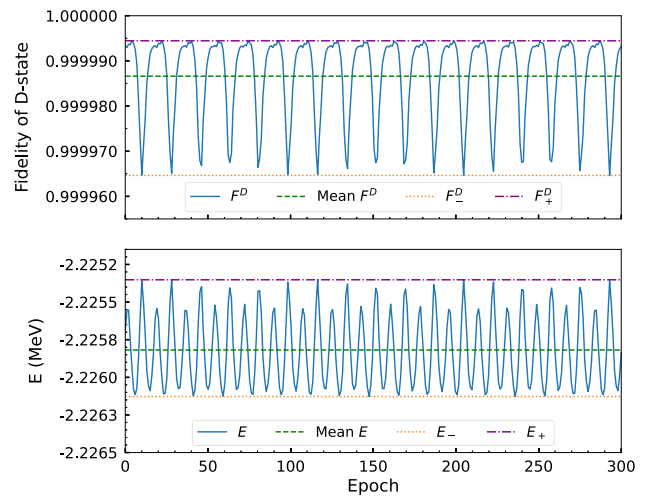
2.3 Error analysis

In addition to different network architectures, we examine two types of uncertainties. First, as we have just explained, we run 20 minimisations for different N_{hid} values and network architectures. We take the standard deviation of these 20 results as a measurement of the uncertainty related to the stochastic minimisation process. This out-of-sample error, which was also explored in Ref. [21], is represented in terms of dark bands in the figures of the following section. It tends to be a relatively small error, of the order of a fraction of a keV in energy.

Second, even after a full minimisation, our results still show some residual oscillations. These oscillations are typically within a few keV of the total energy. Rather than keeping only the lowest values of the energy as our final values, we instead adopt a more conservative stance and we average over the oscillations to obtain a central value. The oscillation amplitude can then be used to quantify an additional source of uncertainty, associated to the minimiser. To determine this error, we take a trained model and let it evolve for 300 additional epochs—a process which we call post-evolution. The number of post-evolution epochs is small enough so that the mean value does not change during the process, but also large enough to observe periodicity in the oscillations.

We illustrate this behaviour in Fig. 3. The top panel shows the evolution of the D -state fidelity F^D over the 300 post-evolution epochs for a 1sc network with $N_{hid} = 100$. Here, we define F^L as the overlap between the ANN and benchmark (obtained via exact diagonalisation) wavefunctions in analogy to Eq. (3). The solid blue line, corresponding to the fidelity at every epoch, displays clear oscillations that are asymmetric with respect to a mean value of $F^D \approx 0.999987$. To account for the asymmetry in the oscillating behaviour, we assign an upper and lower error estimate. The upper value corresponds to the maximum value across the post-evolution phase— $\delta F_+^D = 0.000008$ (dash-dotted line), in the case shown in the figure. In contrast, the lower value is significantly lower for this specific example, leading to $\delta F_-^D = 0.00002$ (dotted line). The bottom panel of Fig. 3 shows the energy (solid line) as a function of the post-evolution epoch for the same simulation. The energy oscillates around the mean value (dashed line), and the top and bottom bounds of these oscillations are shown in dash-dotted and dotted lines respectively. In this particular example, the mean value of the energy is $E^W \approx -2.2258$ MeV, with oscillation errors $\delta E_+^W = 0.5$ keV and $\delta E_-^W = 0.3$ keV. We note that these values are typical for almost any number of neurons, and are extremely small, less than 0.1% of the total energy value.

Fig. 3 Top panel: D-state fidelity post-evolution of a trained model with architecture 1sc and $N_{\text{hid}} = 100$ hidden neurons. Bottom panel: total energy evolution of the same model as in the top panel



In order to take into account the stochastic out-of-sample uncertainty in these oscillations, we repeat the process above several times for each network configuration (N_{hid}) and keep 20 random samples of the networks. As explained earlier, we extract central values for the different physical quantities by taking the average over these 20 mean values. We quote post-evolution oscillation errors that also include this stochastic element. In other words, both upper and lower bounds are calculated by averaging over 20 individual post-evolution runs. In a cost function with no near multiple minimums of similar heights, all oscillations should happen around the same minimum, and the mean values of $E^{\mathcal{W}}$ should all be similar. This, in tandem with the fact that we estimate the stochastic $\delta E^{\mathcal{W}}$ as the standard deviation, leads us to expect small stochastic errors and comparatively bigger oscillation errors. We confirm this expectation in the following section.

3 Results

3.1 Energy

The two quantities that we use as indicators of the quality of our final trained models are the energy and the fidelity, represented in the top and bottom panels of Fig. 4, respectively. The four columns correspond to the different network architectures of Fig. 1. The solid dark lines in Fig. 4 indicate the central values. We show two different types of uncertainties for each quantity. First, stochastic uncertainties associated to 20 different initialisations are shown using dark colour bands. Second, post-evolution oscillation uncertainties are displayed with pale colours. A key finding of our work is that post-evolution uncertainties are always larger than stochastic uncertainties.

Before discussing uncertainties, we want to stress the quality of the results. As discussed in the context of Fig. 2, the ANNs used to generate these results have been preselected to lie in the range $E \in (-2.220, -2.227)$ MeV. Therefore, all these models already lie within 0.3% of the final energy result. For some architectures, the rate of convergence to this energy window is not 100%. Yet, converged results provide energy values which, according to the stochastic uncertainty, are well within 1 keV (or 0.04%) of each other. Post-evolution uncertainties are much larger than stochastic uncertainties, but are typically smaller than 4 keV (or 0.2%). We take this as an indication that all these ANNs provide faithful, high-quality representations of the deuteron wavefunction. While none of the models shown here are able to reach the real minimum even when the post-evolution lower bounds are considered, the distance between the lower bound and the real minimum is always less than 2 keV. Our analysis indicates that this limitation is due to the fact that the network cannot always distinguish between the S - and D -state contributions in high-momentum regions, where the two wavefunctions are equally small. Ultimately, this limitation only has minor consequences, of a fraction of a per cent, in the total energy. An important takeaway of this analysis is that the simplest architecture, 1sc, provides the most stable and overall best results: see panel (a) of Fig. 4.

An interesting conclusion of Fig. 4 is that state-disconnected architectures (panels (c) and (d)) perform seemingly worse than state-connected networks. One could have expected the opposite, on the basis that disconnected architectures may provide more independent flexibility for the ψ^S and ψ^D wavefunctions. One possible explanation for this behaviour is as follows. Within a single minimisation epoch, the optimiser proposes steps based on the backpropagation of gradients. Changes in parameters associated to the S -state are thus immediately propagated to the D state in a fully connected configuration. In contrast, in the disconnected case, these changes do not necessarily affect the D state. In spite of having fewer parameters than connected architectures, the training may take longer due to this effect. In a minimisation protocol with a fixed number of epochs like ours, this may lead to worse results.

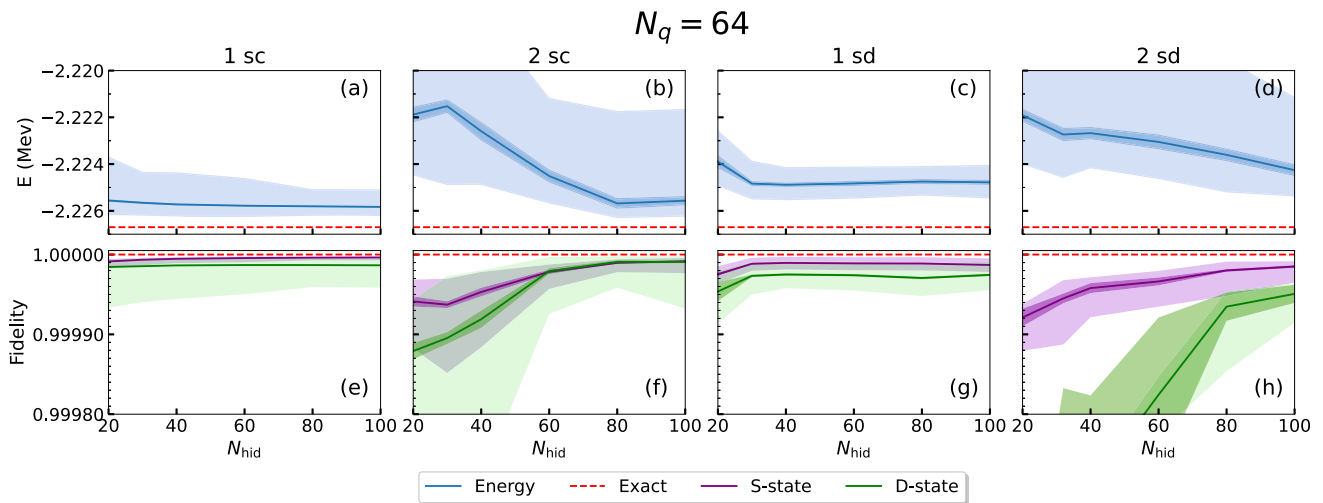


Fig. 4 Panels (a)–(d): energy as a function of the number of hidden nodes. Different panels correspond to the network architectures of Fig. 1. Lines represent central values obtained with 20 stochastic initialisations. Dark bands show the associated standard deviations (stochastic uncertainty), whereas light bands show the post-evolution oscillation uncertainty. The dashed red lines indicate the benchmark value. Panels (e)–(h): the same for the fidelity of the ANN wavefunction with respect to the benchmark for the S (purple) and D (green) states

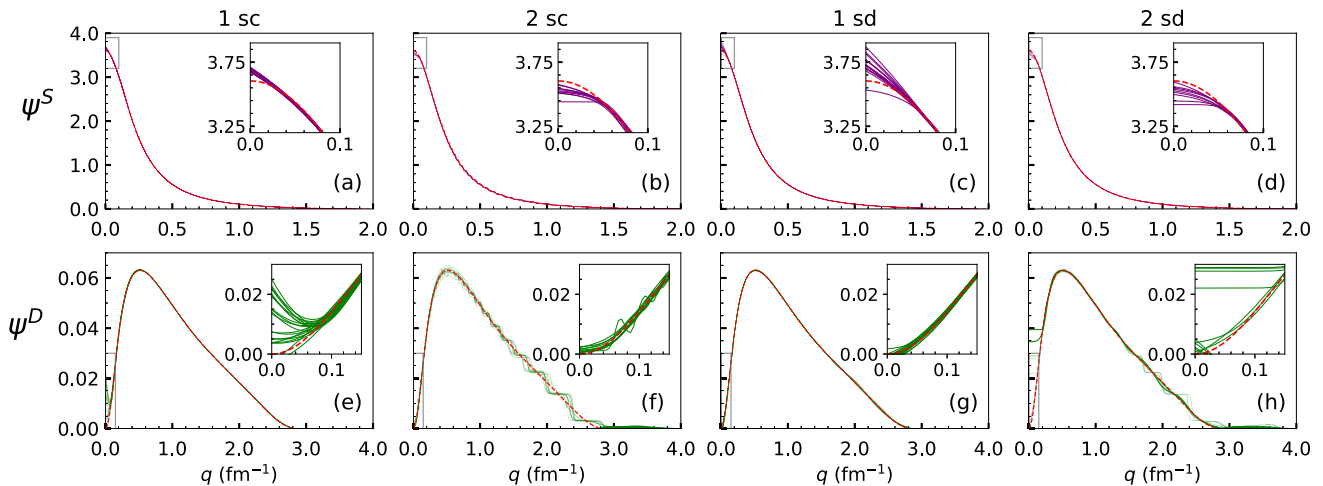


Fig. 5 Panels (a)–(d): 20 instances of the S -state wavefunction as a function of momentum, for each of the 4 architectures considered here for $N_{hid} = 100$. The dashed line represents the benchmark wavefunction. The insets focus on the region around the origin, $q \approx 0$. Panels (e)–(h): the same for the D -state wavefunction. Note the difference in scales of the momentum in both rows

We now proceed to discuss the N_{hid} dependence of the results. We find two very different trends depending on the depth of the network. One-layer network results are relatively independent on the network width. In other words, in terms of energy (but also in terms of fidelity), models with $N_{hid} \approx 30$ are as good as models with $N_{hid} = 100$, as can be seen in panels (a) and (b) of Fig. 4. In contrast, the two-layer models display a clear improvement in energy. This suggests that two-layer networks with small values of N_{hid} have not managed to reach the full minimum after 2.5×10^5 epochs. Single-layer models have notably fewer parameters and can thereby be trained faster than their two-layer counterparts. In fact, snapshots of the minimisation process indicate that two-layer models with few neurons are still learning at the end of the 2.5×10^5 epochs. The decrease in energy as N_{hid} increases extends to the whole range of N_{hid} in architectures 2sc and 2sd (panels (b) and (d) of Fig. 4). However, the post-evolution uncertainties for these two-layer models are relatively big and almost compatible with a constant value of energy.

In addition to central values, the dependence of uncertainties in N_{hid} is informative. The stochastic uncertainties are relatively constant across all values, but are marginally larger for two-layer models. Post-evolution errors for one-layer networks are on the order of ≈ 2 keV, with a mild decreasing dependence on network width. In contrast, two-layer network post-evolution uncertainties are as large as 6 (4) keV for low (high) N_{hid} values. Following the same argument about the energies of such models, this can be understood assuming that the updates in \mathcal{W} from RMSprop affect both ψ^S and ψ^D in the state-connected case (as opposed to the state-disconnected case). One expects that this may lead to larger energy variations and, hence, bigger oscillations.

3.2 Fidelity

The panels (e)–(h) at the bottom of Fig. 4 also provide an insight on the model quality after minimisation. Again, we highlight that overall the fidelities are extremely close to benchmark values, within a fraction of a per cent in all cases. We find general conclusions in terms of network width that are in line with those we found for the energy. Across all architectures and network depths, the fidelity of the S state is better than that associated to the D state. Not only are the central values of F^S closer to one, but post-evolution uncertainties are significantly smaller for this state. This is in contrast to the much smaller stochastic uncertainty, which is of the same size for both states.³

When it comes to the N_{hid} dependence of the results, we find again that one-layer networks are relatively independent of the network width. Overall, networks perform better with 1sc than with 2sc architectures as measured by overlaps that are closer to one, although they have relatively similar uncertainties. The fidelities of the 1sd architecture are relatively constant as the width increases. The behaviour of the 2sd networks shown in panel (h) is slightly different. While the S –state fidelity is relatively close to the benchmark throughout the whole range of N_{hid} , the D –state fidelity is much lower for small $N_{\text{hid}} = 80$, approaching the benchmark only at $N_{\text{hid}} = 80$ or higher. Finally, we stress again that there is no observable bias-variance trade-off in the fidelities [1].

3.3 Wavefunctions

We now turn our attention to analysing directly the ANN outputs: the wavefunctions. This is an instructive exercise that allows us to have local indicators of model quality, in addition to the information provided by integrated quantities such as the energy or the fidelity. We show 20 different instances of the wavefunctions of the S – (top panels, (a)–(d)) and D –states (bottom panels, (e)–(h)) in Fig. 5. To show the “best” possible wavefunctions, we focus on the number of hidden nodes N_{hid} that provide the lowest (central value of the) energies. For all four architectures considered, this corresponds to $N_{\text{hid}} = 100$. By looking at these instances directly, one gets an idea of the possible spread of variationally evolved models. Overall, we find that all networks provide very similar, high-quality representations of the benchmark wavefunction, which is shown in dashed lines.

In most cases, the largest discrepancies occur towards the origin. As explained in Ref. [21], the presence of a q^2 factor in the energy integrals allows the networks to push a large amount of variance towards the low- q region. In other words, changing the wavefunction near the origin has no reflection on the global cost function. The large variance can be seen in the insets of all panels. For the S –state, one-layer models have relatively linear behaviours as $q \rightarrow 0$, whereas two-layer networks saturate close to the origin. None of these behaviours matches the benchmark, showed in dashed lines, in this limit. We note that the 20 realisations provide a relatively similar amount of deviations around a central value. This is in contrast to the D –state wavefunction of the 1sc architecture, panel (e) of Fig. 5, which has a large variation around the origin, in spite of providing the best overall energies.

Unlike in Ref. [21], we display the wavefunctions of Fig. 5 in a linear and denser grid of points than the one used to compute energy quadratures. This allows us to investigate whether the network is able to learn efficiently not only the properties at the mesh points or the origin, but also the continuity of the wavefunction across a set of points at finite momentum. Indeed, we observe a step-like behaviour in the two-layer D –state wavefunctions (panels (f) and (h) of Fig. 5). These steps occur precisely in the vicinity of each quadrature mesh point and they are more easily seen at high momenta where mesh points are relatively spaced apart. Clearly, two-layer networks have enough flexibility to generate horizontal steps around these points. In other words, the networks learn the properties of the wavefunction only locally around the meshpoints and interpolate in a stepwise fashion between them. We take this as another, different sign of out-of-sample uncertainty. We stress that the steps do not occur in simpler, less flexible one-layer models. We speculate that the lack of flexibility of such models forces them to behave in a more continuous fashion. Further, these effects are not detected in integrated quantities, since the quadrature meshpoint values are well reproduced by all the networks. In other words, in our set-up, the regions between mesh points do not contribute to the loss functions. Therefore, the network can push the variance to these regions with no observable effect. We also note that all wavefunctions go immediately to zero for values of q larger than the ones shown in Fig. 5.

Figure 6 provides further insight into the structure of the wavefunction variance in our models. The variance here is estimated as the standard deviation associated to the 20 model initialisations of our networks. The different line styles in the figure correspond to different values of N_{hid} . Top (bottom) panels show S (D) state variances. It is quite clear from these figures that the overall variance at finite momentum is relatively independent of the network width. Moreover, most network models have maximum variances towards the origin, which is in line with Fig. 5. 1–layer networks have relatively flat declines with momentum for the S state, and saturate at values of the order of $\sigma \approx 10^{-3}$, as shown in panels (a) and (c).

In contrast, two-layer networks have pronounced oscillatory values. The oscillation positions have a one-to-one correspondence with the plateaus in Fig. 5. Variance minima occur at the quadrature mesh points, and maxima happen in between. The oscillation minima (e.g. the smallest uncertainties) have variances which are similar to the 1–layer case. This is a clear indication that the network has learned locally the wavefunctions of the system at the quadrature mesh grids, but is overfitting the values in between, which are not penalised by the cost function. We discuss potential mitigation strategies to this problem in the following sections.

³ The only exception is the D –state fidelity of the 2sd architecture with $N_{\text{hid}} = 60$, shown in panel (h) of Fig. 4, which has a larger oscillation uncertainty.

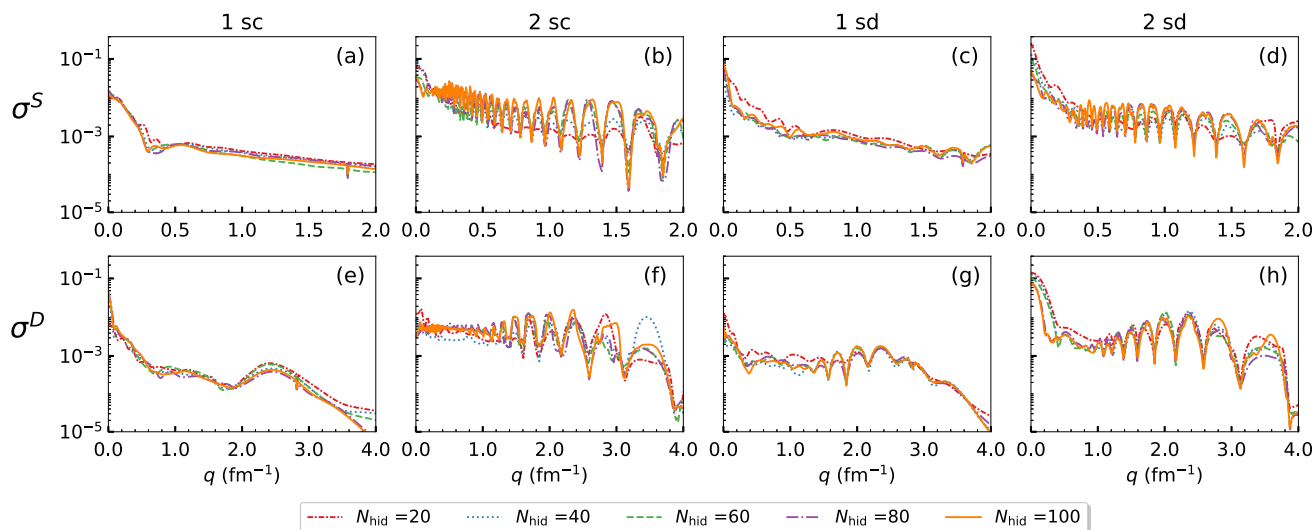


Fig. 6 Panels (a)–(d): standard deviation associated to 20 instances of the S –state wavefunction as a function of momentum, for each of the 4 architectures considered here. Different line styles correspond to different N_{hid} values. Panels (e)–(h): the same for the D –state wavefunction. Note the difference in scales of the momentum in both rows

Table 1 Fidelity and energies of the 1sd and 2sd architectures with $N_{\text{hid}} = 100$ hidden neurons in a uniform mesh, where the overfitting is captured

	F^S	F^D	E (MeV)
1sd	0.99997 ± 0.00001	0.9999 ± 0.0001	-2.216 ± 0.003
2sd	0.9997 ± 0.0001	0.99 ± 0.01	-2.1 ± 0.1
Benchmark	1	1	-2.22486

Each value is the mean over the set of 20 initialisations, and the errors are the associated standard deviations

Before discussing solutions to this problem, we want to quantify its relevance. To this end, we generate wavefunction models in a new uniform 10^3 -point mesh in the interval $[0, 5] \text{ fm}^{-1}$. We use this new linear mesh to compute, in quadrature, the overlaps of the network models to the benchmark wavefunction as well as the total energy. To summarise our findings, we show in Table 1 the results of this exercise for two different models. On the one hand, the 1sd model with $N_{\text{hid}} = 100$ (top row) shows almost no signs of overfitting. In contrast, the middle row shows results for the 2sd architecture with $N_{\text{hid}} = 100$, which has significant overfitting. The bottom row quotes the values obtained with a benchmark wavefunction, obtained in the very same uniform mesh.

First, we discuss the fidelities reported in the first and second columns of Table 1. These are computed in the uniform mesh, as opposed to the results presented in Fig. 4 for the original quadrature. For the 1sd model, the fidelities are practically the same, close to 1 up to the fourth or fifth significant digit. This is due to the fact that the model has barely any overfitting. The fidelities of the 2sd model are, however, one and two orders of magnitude further away from one than those reported in Fig. 4. We take this as an indication of potential overfitting.

A second, stronger indication comes from the energies in the final column. These values should be compared to the benchmarks in the same uniform mesh, reported in the bottom row. Two conclusions can be drawn here. First, the energy values obtained in this mesh are significantly worse than the exact ones. Whereas for the 1sd value the energy lies within a fraction of a per cent of the benchmark, for the 2sd model the model is only accurate up to 10% of the total energy. Second, the stochastic uncertainties are significantly larger than those reported in Fig. 4. For the 1sd network, the values are relatively competitive, of the order of 1 keV. The 2sd model has stochastic uncertainties of 200 keV, several orders of magnitude higher.

This substantial increase in stochastic uncertainty, as well as the important change in energy central values, suggests that overfitting is particularly problematic for two-layer networks. Among other things, this indicates that the results shown in Fig. 4 are not fully representative of the true values of two-layer networks. The general behaviour of these quantities as a function of N_{hid} is, however, expected to remain the same.

Having quantified the importance of overfitting, we now discuss its causes. The differences between the exact and predicted wavefunctions appear almost exclusively in two-layer models. These models have comparatively more parameters, which are presumably redundant. Two-layer networks can “hide” this redundancy in regions with little or no contributions to the cost functions, either towards the origin or in regions in-between fixed quadrature mesh points. We stress that having too many parameters does not restrict the ANN outputs, but it may increase the difficulty of the task assigned to the optimiser. The easy path out in this case is to overfit the mesh points.

Table 2 Energy (E) and convergence rate (r) for different activation functions (Act. fun.), learning rates (lr), smoothing constants (α) and momentum (μ)

		1sd		
		Hyperparam	E (MeV)	r
The errors are displayed in the format: (stoch · osc), where stoch and osc are the stochastic and oscillation errors, respectively. All these results correspond to the 1sd architecture and $N_{\text{hid}} = 60$	Act. fun.	Sigmoid	$-2.22473^{+(0.00006, 0.0007)}_{-(0.00006, 0.0006)}$	66.9%
		Softplus	$-2.2220^{+(0.0004, 0.004)}_{-(0.0004, 0.002)}$	12.2%(*)
		ReLU	$-2.2223^{+(0.0003, 0.004)}_{-(0.0003, 0.002)}$	13.2%(*)
	lr	0.005	$-2.22598^{+(0.00002, 0.0002)}_{-(0.00002, 0.0002)}$	90.7%
		0.01	$-2.22473^{+(0.00006, 0.0007)}_{-(0.00006, 0.0006)}$	66.9%
		0.05	$-2.2231^{+(0.0004, 0.003)}_{-(0.0004, 0.001)}$	5.3%(*)
	α	0.7	$-2.22503^{+(0.00006, 0.0009)}_{-(0.00006, 0.0006)}$	59.6%
		0.8	$-2.22488^{+(0.00008, 0.0008)}_{-(0.00008, 0.0006)}$	65.6%
	μ	0.9	$-2.22473^{+(0.00006, 0.0007)}_{-(0.00006, 0.0006)}$	66.9%
0.0		$-2.22473^{+(0.00006, 0.0007)}_{-(0.00006, 0.0006)}$	66.9%	
		0.9	$-2.2253^{+(0.0003, 0.002)}_{-(0.0003, 0.0009)}$	28.5%

Moreover, as one increases the number of parameters, the network may find multiple ways of fitting the data set evaluated on the same fixed N_q mesh points. In other words, larger networks can represent the same wavefunctions in many different ways, and as the number of parameters grow, so do the number of ways in which a network can represent the wavefunctions. This is usually identified as a bias-variance trade-off.

We now speculate on how to mitigate the overfitting effects on the wavefunctions. An obvious alternative is to try and increase the mesh density. One expects that in regions which are densely covered by the mesh, the networks may not have enough freedom to develop the artificial plateaus observed in Fig. 5. Just as in traditional variational models, improving the quality of the (momentum-space) basis by, say, increasing N_q , may also be beneficial in terms of the optimisation. Alternatively, one can envisage a set-up in which meshes change from epoch to epoch. If such changes are entirely stochastic, one could rephrase them as traditional variational quantum Monte Carlo implementations, although these are not usually formulated with non-local interactions like the Entem–Machleidt potential. Alternatively, one could work with deterministic meshes but change the total number of meshpoints or the meshpoint positions at each epoch. In the most favourable scenario, one would expect to find a network that does not memorise specific data mesh points, but rather learns the wavefunction structure. In the end, any such strategy would try to enforce a network learning process that is independent of the specific quadrature that is used to compute the global cost energy function.

3.4 Hyperparameter exploration

In the analysis performed throughout the previous subsections, we explored the performance of four ANN architectures with different numbers of hidden neurons, N_{hid} , with the rest of the hyperparameters kept fixed. Now, for the sake of completeness, we present some results for the ground-state energy computed with different hyperparameters. We discuss here the 1sd architecture and show a qualitative discussion for the other architectures in the “Appendix”. All the hyperparameter exploration has been performed for networks with $N_{\text{hid}} = 60$.

In this exploration, we look at the effect of the activation function and the learning rate. We use one of the best-behaved optimisers in the literature (RMSprop), and let some of its parameters vary to explore additional systematics. Specifically, we modify the smoothing constant (α) and the momentum (μ) of RMSprop. To explore the systematic dependence of the hyperparameters, we change one specific hyperparameter at a time, keeping the values of all other hyperparameters fixed to those used in Sect. 3. Specifically, our baseline uses a Sigmoid activation function, a learning rate of 0.01 and $\alpha = 0.9$ and $\mu = 0.0$.

Table 2 shows numerical results of both the energy and the convergence rate for different combinations of the aforementioned hyperparameters, all of them sharing the same network architecture, 1sd. Notice that for each hyperparameter class there is a row which is repeated: this corresponds to the baseline hyperparameter configuration, which is also the same one used in the previous sections, and displayed here to facilitate the comparison. In a similar spirit to that in Sect. 2, we use $N_{\text{tot}} = 150$ trained models to compute all the convergence rates presented in Table 2. Nevertheless, not all hyperparameter configurations achieve $N_{\text{con}} = 20$ converged models with this N_{tot} . In Table 2, we include an asterisk (*) next to the convergence rate of the models which need more runs. For all such models, the convergence rate is computed using $N_{\text{tot}} = 1100$.

Concerning the changes in the activation function shown at the top of the table, we observe that our choice, the Sigmoid, has the highest convergence rate of $r = 66.9\%$ for this architecture. This is contrast to $r < 15\%$ for both the ReLU and Softplus functions,

indicating that models have a harder time converging for these functions. Not only this, but Sigmoids also yield the lowest (and hence best) energy value. In contrast, ReLU and Softplus come at considerably higher energies.

Models with lower learning rates are inherently slower to train. Because we work with a fixed number of iterations, a change in the learning can thus have a large effect in the results. To attenuate such artifices, we train models with a learning rate that is half that of our benchmark, $lr = 0.005$, for 5×10^5 epochs instead of 2.5×10^5 epochs. We also explore models with larger learning rates ($lr = 0.05$) for the same number of epochs than our baseline. We observe that both the energy and the convergence rate improve as the learning rate decreases. The associated energy uncertainties also decrease, and we note that the best results for $lr = 0.005$ are incompatible with the baseline of $lr = 0.01$, which is half as cheap in epochs.

Finally, we explore the hyperparameter space associated to the optimiser in the bottom rows of Table 2. Our results indicate that variations in the smoothing constant α and the momentum μ do not have a significant impact on the metrics studied here. Perhaps the most robust conclusion is that adding momentum to RMSprop reduces the convergence rate of our simulations. The interested reader can find a similar table for the 1sc, 2sc and 2sd configurations in the “Appendix”.

4 Conclusions and future outlook

In this work, we use vANNs to compute the ground-state properties of the deuteron with a realistic Hamiltonian. To this end, we discretise the problem in a fixed mesh on the relative momentum coordinate. We use standard ML tools, based on PyTorch, to pretrain our models and subsequently minimise the energy for a fixed number of epochs.

High-quality solutions for the variational wavefunction were already obtained in Ref. [21] with a similar set-up. We extend this work here in several directions, aimed at identifying fundamental limitations of the vANN approach. First, we look at different ANN architectures, increasing the number of layers and treating the connection to the output states in a connected or disconnected fashion. Second, we identify a new source of uncertainty associated to the oscillations around the final energy minimum. Third, by carefully analysing the wavefunction outputs, we identify conditions in which finite-momentum overfitting arises.

All vANNs models provide excellent results for the energies and fidelities, when compared to benchmark wavefunctions. By looking at the rate of model convergence for different architectures, we find a first qualitative sign that two-layer networks have a harder time minimising the energy than their one-layer counterparts. In terms of model uncertainties, the post-evolution oscillation errors dominate over the stochastic initialisation errors, but they remain relatively small (of the order of 6 to 8 keV) across a wide range of network widths. When it comes to the structure of the network, state-connected networks, with output nodes that are connected to the internal hidden layer, provide marginally better results than state-disconnected architectures.

Overall, we find that two-layer networks provide worse results than one-layer models. This may have been expected since the input layer is based on a single, positive-defined degree of freedom (the magnitude of the relative momentum, q). Central values of the energy are less attractive, and the stochastic and post-evolution uncertainties are larger. We also identify a dependence on the number of hidden nodes, which is absent in the one-layer case. Representing the wavefunctions of these models at grid points that are not used in the minimisation process, we find anomalous horizontal steps between mesh points for the two-layer models. These are clear signs of network overfitting. It seems that, during the minimisation process, the network is able to push some of the redundant dependence of parameters into these regions, which do not contribute to the energy cost function. This is analogous to the observation of a large variance (in terms of wavefunction values) around the origin, where the spherical q^2 factor allows the network to change values arbitrarily around $q \approx 0$ without affecting the total energy. Unlike the situation at the origin, however, a change of integration mesh for the energy can easily detect the degradation of the model associated to overfitting at finite momentum. We find that, in the new mesh, the fidelities with respect to benchmark wavefunctions become worse. The energy values in a different mesh are also substantially less attractive. The associated stochastic uncertainty increases, reaching up to 300 keV in some cases.

While unveiling the internal behaviour of the ANN is hard, the comparison between different architectures certainly sheds some light on the fundamental limitations of these variational methods. When it comes to presenting wavefunctions that depend on a single continuous variable, our results indicate that one-layer networks provide a better starting point than the more complex two-layer approaches. This effect may be due to the fixed (if long) total number of epochs, but other observations indicate that overfitting arises much more easily in deeper networks. We have explored the hyperparameter and activation function dependence of our results. We find that our baseline model, for Sigmoid functions and moderate learning rates, is almost optimal. The only significant dependence is on the learning rate, which unfortunately requires running simulations for a higher number of epochs.

Similarly, our results suggest that the network can tackle states with different quantum numbers in a fully connected configuration. In training ANNs to represent continuous quantum systems, non-fixed grid methods may provide superior learning capabilities. Overall, this experience provides useful ideas to build more sophisticated nuclear models and tackle more difficult problems, including those of a many-body nature.

Acknowledgements We thank Bruno Juliá-Díaz for fruitful discussions. This work is supported by STFC, through Grants Nos ST/L005743/1 and ST/P005314/1; by Grant PID2020-118758GB-I00 funded by MCIN/AEI/10.13039/501100011033; by the “Ramón y Cajal” Grant RYC2018-026072 funded by MCIN/AEI/10.13039/501100011033 and FSE “El FSE invierte en tu futuro”; and by the “Unit of Excellence María de Maeztu 2020-2023” award to the Institute of Cosmos Sciences, funded by MCIN/AEI/10.13039/501100011033.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

Data Availability Statement The code used to produce all the results in this paper, including the figures appearing in the text, is available at the GitHub repository <https://github.com/javier-rozalen/deuteron>. There are instructions and figures to facilitate the understanding and usage of the code. The manuscript has associated data in a data repository.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Appendix A: Hyperparameter dependence exploration

In Sect. 3.4, we briefly commented on the various hyperparameters in our models, and in Table 2, we showed the effects of these variations upon the energy and the convergence rate for the 1sd architecture. Here, we provide a similar analysis for the architectures 1sc, 2sc and 2sd.

Table 3 shows numerical results of both the energy and the convergence rate for different values of the hyperparameters. We explore this dependence by changing one parameter at a time and keeping the remaining terms fixed. Here, we find the same issue with convergence rates discussed in Sect. 3.4: for some hyperparameter configurations, using $N_{\text{tot}} = 150$ is insufficient to guarantee a minimum of $N_{\text{con}} = 20$ converged models. For such models, we use $N_{\text{tot}} = 1100$ trained models instead, and we include an asterisk (*) next to their convergence rate in Table 3. We extract two relevant pieces of information from this table. First, we find that 1sc is the most stable architecture. Both the convergence rates and the energy values of this network are robust against changes of the hyperparameters. Second, we also find that a high convergence rate does not always entail a low ground-state energy. This can be easily understood realising that, while the trend for convergence rates appears to be linked almost exclusively to the architecture, the energy values depend instead on a wider range of factors. This is certainly something to bear in mind in future vANN calculations, where ANN architectures may have to be explored in detail.

We also find a less direct but significant dependence on the learning rate. Just as for the 1sd architecture, we find that lowering the learning rate by a factor of 2 (and doubling the number of epochs) seems to lead to better variational minima. These also show slightly smaller uncertainties and improved convergence rates.

For the 1sc and 2sd architectures, the numbers in Table 3 also suggest that adding momentum results in models with lower (and hence better) energies. We also notice that two-layer architectures have much worse convergence rates than their one-layer counterparts for $\mu = 0.9$. This may be understood with similar arguments as those presented in the main text. The 2sc and 2sd networks present overfitting, which suggests that there is an excessive number of hidden neurons. Therefore, parameter updates in these models are prone to having a greater impact on the overall wavefunction, which makes training less predictable, and increasing the momentum value is yet another step in that same direction.

Table 3 The same as Table 2, but for the ANN architectures 1sc, 2sc and 2sd

	Hyperp.	1sc		2sc		2sd	
		E (MeV)	r	E (MeV)	r	E (MeV)	r
A. F.	Sigmoid	$-2.225796^{+(0.000007, 0.001)}_{-(0.000007, 0.0004)}$	100.0%	$-2.2248^{+(0.0002, 0.004)}_{-(0.0002, 0.001)}$	60.3%	$-2.2231^{+(0.0003, 0.004)}_{-(0.0003, 0.001)}$	23.2%
	Softplus	$-2.22534^{+(0.00001, 0.002)}_{-(0.00001, 0.0009)}$	100.0%	$-2.22490^{+(0.00009, 0.002)}_{-(0.00009, 0.0009)}$	98.0%	$-2.22512^{+(0.00006, 0.003)}_{-(0.00006, 0.0009)}$	51.7%
	ReLU	$-2.2235^{+(0.0002, 0.006)}_{-(0.0002, 0.002)}$	45.0%	$-2.2247^{+(0.0002, 0.003)}_{-(0.0002, 0.001)}$	68.2%	$-2.2232^{+(0.0003, 0.004)}_{-(0.0003, 0.001)}$	15.2%
lr	0.005	$-2.226305^{+(0.000009, 0.0002)}_{-(0.000009, 0.0001)}$	100.0%	$-2.2255^{+(0.0003, 0.002)}_{-(0.0003, 0.0006)}$	72.9%	$-2.2245^{+(0.0002, 0.0007)}_{-(0.0002, 0.0004)}$	49.0%
	0.01	$-2.225796^{+(0.000007, 0.001)}_{-(0.000007, 0.0004)}$	100.0%	$-2.2248^{+(0.0002, 0.004)}_{-(0.0002, 0.001)}$	60.3%	$-2.2231^{+(0.0003, 0.004)}_{-(0.0003, 0.001)}$	23.2%
	0.05	$-2.226111^{+(0.000004, 0.0005)}_{-(0.000004, 0.0002)}$	86.5%	$-2.2249^{+(0.0004, 0.01)}_{-(0.0004, 0.001)}$	13.2%	$-2.2215^{+(0.0002, 0.008)}_{-(0.0002, 0.002)}$	3.0%(*)
α	0.7	$-2.22567^{+(0.00002, 0.0002)}_{-(0.00002, 0.0001)}$	100.0%	$-2.2228^{+(0.0004, 0.003)}_{-(0.0004, 0.002)}$	55.0%	$-2.2229^{+(0.0002, 0.0009)}_{-(0.0002, 0.0007)}$	38.4%
	0.8	$-2.225742^{+(0.000008, 0.0003)}_{-(0.000008, 0.0002)}$	100.0%	$-2.2239^{+(0.0004, 0.002)}_{-(0.0004, 0.001)}$	49.0%	$-2.2234^{+(0.0003, 0.002)}_{-(0.0003, 0.0008)}$	46.4%
	0.9	$-2.225796^{+(0.000007, 0.001)}_{-(0.000007, 0.0004)}$	100.0%	$-2.2248^{+(0.0002, 0.004)}_{-(0.0002, 0.001)}$	60.3%	$-2.2231^{+(0.0003, 0.004)}_{-(0.0003, 0.001)}$	23.2%
μ	0.0	$-2.225796^{+(0.000007, 0.001)}_{-(0.000007, 0.0004)}$	100.0%	$-2.2248^{+(0.0002, 0.004)}_{-(0.0002, 0.001)}$	60.3%	$-2.2231^{+(0.0003, 0.004)}_{-(0.0003, 0.001)}$	23.2%
	0.9	$-2.226614^{+(0.000007, 0.0002)}_{-(0.000007, 0.00007)}$	100.0%		0.0%	$-2.22612^{+(0.00008, 0.001)}_{-(0.00008, 0.0005)}$	3.2%(*)

Table 3 also confirms some of the results we discussed in the main body of this manuscript. We observe that two-layer architectures have lower convergence rates than one-layer networks throughout the entire range of hyperparameter values explored here. The same thing occurs for disconnected architectures, which have lower rates and somewhat worse energies than their counterparts. We take this as an indication that the tendencies are robust and reflect features linked to the network architecture itself, and not the training process details.

References

- P. Mehta, M. Bukov, C. Wang et al., A high-bias, low-variance introduction to machine learning for physicists. *Phys. Rep.* **810**, 1–124 (2019). <https://doi.org/10.1016/j.physrep.2019.03.001>. arXiv:1803.08823
- G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, L. Zdeborová, Machine learning and the physical sciences. *Rev. Mod. Phys.* **91**, 045002 (2019). <https://doi.org/10.1103/RevModPhys.91.045002>. arXiv:1903.10563
- A. Boehnlein, M. Diefenthaler, N. Sato, M. Schram, V. Ziegler, C. Fanelli, M. Hjorth-Jensen, T. Horn, M.P. Kuchera, D. Lee, W. Nazarewicz, P. Ostroumov, K. Orginos, A. Poon, X.-N. Wang, A. Scheinker, M.S. Smith, L.-G. Pang, Colloquium: Machine learning in nuclear physics. *Rev. Mod. Phys.* **94**, 031003 (2022). <https://doi.org/10.1103/RevModPhys.94.031003>. arXiv:2112.02309
- R. Utama, J. Piekarewicz, H.B. Prosper, Nuclear mass predictions for the crustal composition of neutron stars: a Bayesian neural network approach. *Phys. Rev. C* **93**(1), 014311 (2016). <https://doi.org/10.1103/PhysRevC.93.014311>. arXiv:1508.06263
- X. Gao, L.M. Duan, Efficient representation of quantum many-body states with deep neural networks. *Nat. Commun.* **8**(1), 1 (2017). <https://doi.org/10.1038/s41467-017-00705-2>. arXiv:1701.05039
- R.-D. Lasserri, D. Regnier, J.-P. Ebran, A. Penon, Taming nuclear complexity with a committee of multilayer neural networks. *Phys. Rev. Lett.* **124**, 162502 (2020). <https://doi.org/10.1103/PhysRevLett.124.162502>. arXiv:1910.04132
- Z.M. Niu, H.Z. Liang, B.H. Sun, W.H. Long, Y.F. Niu, Predictions of nuclear β -decay half-lives with machine learning and their impact on r -process nucleosynthesis. *Phys. Rev. C* **99**, 064307 (2019). <https://doi.org/10.1103/PhysRevC.99.064307>
- Z.-A. Wang, J. Pei, Y. Liu, Y. Qiang, Bayesian evaluation of incomplete fission yields. *Phys. Rev. Lett.* **123**(12), 122501 (2019). <https://doi.org/10.1103/PhysRevLett.123.122501>. arXiv:1906.04485
- K. Raghavan, P. Balaprakash, A. Lovato, N. Rocco, S.M. Wild, Machine-learning-based inversion of nuclear responses. *Phys. Rev. C* **103**, 035502 (2021). <https://doi.org/10.1103/PhysRevC.103.035502>. arXiv:2010.12703
- X.-X. Dong, R. An, J.-X. Lu, L.-S. Geng, Novel Bayesian neural network based approach for nuclear charge radii. *Phys. Rev. C* **105**, 014308 (2022). <https://doi.org/10.1103/PhysRevC.105.014308>. arXiv:2109.09626
- C. Adams, G. Carleo, A. Lovato, N. Rocco, Variational Monte Carlo calculations of $a \leq 4$ nuclei with an artificial neural-network correlator ansatz. *Phys. Rev. Lett.* **127**, 022502 (2021). <https://doi.org/10.1103/PhysRevLett.127.022502>. arXiv:2007.14282
- A. Gnech, C. Adams, N. Brawand, G. Carleo, A. Lovato, N. Rocco, Nuclei with up to $A = 6$ nucleons with artificial neural network wave functions. *Few-Body Syst.* **63**(1), 7 (2021). <https://doi.org/10.1007/s00601-021-01706-0>. arXiv:2108.06836
- G. Carleo, K. Choo, D. Hofmann, J.E.T. Smith, T. Westerhout, F. Alet, E.J. Davis, S. Efthymiou, I. Glasser, S.-H. Lin, M. Mauri, G. Mazzola, C.B. Mendl, E. van Nieuwenburg, O. O'Reilly, H. Théveniaut, G. Torlai, F. Vicentini, A. Wietek, Netket: a machine learning toolkit for many-body quantum systems. *SoftwareX* (2019). <https://doi.org/10.1016/j.softx.2019.100311>. arXiv:1904.00031
- F. Vicentini, D. Hofmann, A. Szabó, D. Wu, C. Roth, C. Giuliani, G. Pescia, J. Nys, V. Vargas-Calderón, N. Astrakhantsev, G. Carleo, NetKet 3: machine learning toolbox for many-body quantum systems. *SciPost Phys. Codebases* (2022). <https://doi.org/10.21468/SciPostPhysCodeb.7>
- G. Carleo, M. Troyer, Solving the quantum many-body problem with artificial neural networks. *Science* (2017). <https://doi.org/10.1126/science.aag2302>. arXiv:1606.02318
- H. Saito, Method to solve quantum few-body problems with artificial neural networks. *J. Phys. Soc. Jpn.* (2018). <https://doi.org/10.7566/JPSJ.87.074002>. arXiv:1804.06521
- T. Vieijra, C. Caser, J. Nys, W. De Neve, J. Haegeman, J. Ryckebusch, F. Verstraete, Restricted Boltzmann machines for quantum states with non-abelian or anyonic symmetries. *Phys. Rev. Lett.* **124**, 097201 (2020). <https://doi.org/10.1103/PhysRevLett.124.097201>. arXiv:1905.06034
- K. Choo, A. Mezzacapo, G. Carleo, Fermionic neural-network states for ab-initio electronic structure. *Nat. Commun.* **11**(1), 2368 (2020). <https://doi.org/10.1038/s41467-020-15724-9>. arXiv:1909.12852
- J. Hermann, Z. Schätzle, F. Noé, Deep-neural-network solution of the electronic schrödinger equation. *Nat. Chem.* **12**(10), 891–897 (2020). <https://doi.org/10.1038/s41557-020-0544-y>. arXiv:1909.08423
- D. Pfau, J. Spencer, A. de G. Matthews, et al., Ab-initio solution of the many-electron Schrödinger equation with deep neural networks. *Phys. Rev. Res.* (2020). <https://doi.org/10.1103/PhysRevResearch.2.033429>. arXiv:1909.02487
- J. Keeble, A. Rios, Machine learning the deuteron. *Phys. Lett. B* (2020). <https://doi.org/10.1016/j.physletb.2020.135743>. arXiv:1911.13092
- J.M. Eisenberg, W. Greiner, *Nuclear Theory. Microscopic Theory of the Nucleus. Vol. 3 of Nuclear Theory* (North-Holland Publishing Company, Amsterdam, 1975)
- D. R. Entem, R. Machleidt, Accurate charge-dependent nucleon–nucleon potential at fourth order of chiral perturbation theory. *Phys. Rev. C* (2003). <https://doi.org/10.1103/PhysRevC.68.041001>. arXiv:nucl-th/0304018
- A. Paszke, S. Gross, F. Massa, et al., Pytorch: an imperative style, high-performance deep learning library, in *Advances in Neural Information Processing Systems 32* (Curran Associates, Inc., 2019). <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>
- Pytorch tutorials. <https://pytorch.org/tutorials/beginner/basics/intro.html>. Accessed 27 Apr 2022
- A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in PyTorch, in *NIPS Autodiff Workshop* (2017). <https://pytorch.org/>
- Project github repository. <https://github.com/javier-rozalen/deuteron.git>. Accessed 15 Apr 2022
- D.J.C. MacKay, *Information Theory, Inference, and Learning Algorithms*, 1st edn. (Cambridge University Press, Cambridge, 2003)
- S. Hochreiter, The vanishing gradient problem during learning recurrent neural nets and problem solutions. *Int. J. Uncertain. Fuzz.* **6**(2), 107–116 (1998). <https://doi.org/10.1142/S0218488598000094>
- Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw.* **5**(2), 157–166 (1994). <https://doi.org/10.1109/72.279181>