IMSRG-Net: A machine-learning-based solver for the in-medium similarity renormalization group method

Sota Yoshida[*]
Institute for Promotion of Higher Academic Education, Utsunomiya University, Mine, Utsunomiya 321-8505, Japan

(Received 19 June 2023; revised 25 August 2023; accepted 18 September 2023; published 6 October 2023)

I present a method, IMSRG-Net, which utilizes machine-learning techniques as a solver for the in-medium similarity renormalization group (IMSRG). The primary objective of IMSRG-Net is to approximate the Magnus operators $\Omega(s)$ in the IMSRG flow equation, thereby offering an alternative to the computationally intensive part of IMSRG calculations. The key idea of IMSRG-Net is its design of the loss function inspired by physics-informed neural networks to encode the underlying *physics*, i.e., the IMSRG flow equation, into the model. Through training on a dataset comprising ten data points with flow parameters up to $s = 20$, capturing approximately one-eighth to one-quarter of the entire flow, IMSRG-Net exhibits remarkable accuracy in extrapolating the ground-state energies and charge radii of ^{16}O and ^{40}Ca. Furthermore, this model demonstrates effectiveness in deriving effective interactions for a valence space.

DOI: 10.1103/PhysRevC.108.044303

I. INTRODUCTION

The in-medium similarity renormalization group (IMSRG) method [1–5] is a highly powerful framework to study nuclear many-body systems. This method serves as an *ab initio* technique for investigating the properties of nuclei near subshell closures, while also enabling the systematic derivation of effective interactions and operators for a valence space. The IMSRG method is formulated by the unitary transformation of operators, such as the Hamiltonian, through the IMSRG flow equation. The objective is to decouple a target subspace from the rest of many-body Hilbert space. For ground-state calculations, this entails decoupling particle-hole excitations from the reference state, whereas for deriving effective interactions, the focus is on decoupling the valence space from the core and outside (excluded) space. Notably, recent studies have extended the application of the IMSRG to heavier nuclei, such as ^{132}Sn and ^{208}Pb [6,7].

Although the IMSRG method is a powerful approach, it is still computationally demanding to perform numerous calculations for different nuclei and input nuclear interactions. Consequently, it is crucial to develop efficient methods for conducting IMSRG calculations. The construction of such emulators or surrogate models has emerged as a prominent research topic within the nuclear physics community, providing as a key tool for comprehending and evaluating the uncertainties associated with nuclear many-body calculations and realistic nuclear potentials. A notable example of such emulators is the eigenvector continuation (EC) method [8–10], which has been extensively applied to diverse nuclear many-body problems [11–21]. Its significance has been recognized from a broader perspective as model order reduction [22,23]. However, applying the EC method to IMSRG calculations poses challenges since the EC method primarily operates on many-body wave functions (eigenvectors of a Hamiltonian), while IMSRG calculations are performed on many-body operators, such as the Hamiltonian.

In this work, I present an alternative approach to constructing a surrogate model for IMSRG, employing a data-driven technique based on machine learning. The nuclear physics community has witnessed diverse applications of machine-learning-based models to replicate or assist nuclear many-body calculations (see, e.g., Ref. [24]). I propose a machine-learning-based solver for the IMSRG flow equation, named IMSRG-Net, which is inspired by the physics-informed neural network (PINN) [25,26]. The neural network model, IMSRG-Net, is trained to approximate Magnus operators in IMSRG methods as a function of the flow parameter s. The primary objective of this study is to provide a proof of concept, exploring the potential of data-driven approaches in constructing surrogate models for IMSRG methods, rather than aiming to accelerate the cutting-edge IMSRG calculations on supercomputers. The test grounds for the proposed method are IMSRG calculations of ground-state energies and charge radii of ^{16}O and ^{40}Ca and derivation of effective interactions for sd- and pf-shell nuclei.

This work is organized as follows: In Sec. II, I briefly review the IMSRG method. Section III is devoted to the description of the proposed method, IMSRG-Net. The detailed procedure for training the neural network model and its computational cost are also discussed therein. In Sec. IV, I present the results of the IMSRG-Net calculations. Finally, I summarize this work in Sec. V. Throughout this work, I used NuclearToolkit.jl [27] for performing IMSRG calculations and PyTorch [28] for constructing and training the neural network model. One can reproduce the results with the codes and data on the author's GitHub repository [29].

[*]syoshida@cc.utsunomiya-u.ac.jp

II. METHODOLOGY

Here, let us briefly review the basics of the in-medium similarity renormalization group (IMSRG) method. In IMSRG methods, one starts from a normal-ordered Hamiltonian $H(s=0)$ on a reference state and then performs the unitary transformation $U(s)$ to decouple particle-hole excitations from the reference state

$$H(s) = U(s)H(0)U^\dagger(s), \tag{1}$$

where s is the flow parameter. This gives the following IMSRG-flow equation:

$$\frac{dH(s)}{ds} = [\eta(s), H(s)], \tag{2}$$

$$\eta(s) \equiv \frac{dU(s)}{ds}U^\dagger(s) = -\eta^\dagger(s). \tag{3}$$

In the last decade, significant progress has been made in solving the IMSRG flow equation employing the Magnus expansion [30]. Within the Magnus formulation of the IMSRG, the unitary transformation in Eq. (1) is explicitly evaluated by

$$U(s) = e^{\Omega(s)}, \tag{4}$$

with the anti-Hermitian Magnus operator $\Omega(s)$. By doing this, one can write down the transformations of any operators $O(s)$ including the Hamiltonian $H(s)$ as

$$O(s) = e^{\Omega(s)}O(0)e^{-\Omega(s)}. \tag{5}$$

The flow equation (2) is now translated into the ordinary differential equation for $\Omega(s)$ and the *adjoint* of $\eta(s)$

$$\frac{d\Omega}{ds} = \sum_{k=0}^{\infty}\frac{B_k}{k!}ad_\Omega^{(k)}(\eta), \tag{6}$$

$$ad_\Omega^{(k)}(\eta) = \left[\Omega, ad_\Omega^{(k-1)}(\eta)\right], \tag{7}$$

$$ad_\Omega^{(0)}(\eta) = \eta, \tag{8}$$

where the explicit dependence of Ω and η on the flow parameter s is omitted, and B_k is the Bernoulli number. One can evaluate evolved operators, Eq. (5), through the Baker-Campbell-Hausdorff (BCH) formula.

Throughout this work, I employ the so-called IMSRG(2) truncation, where all operators are truncated up to the normal-ordered two-body (NO2B) level. Let us restrict ourselves to consider the IMSRG with a single spherical reference state and to use the so-called arctangent generator for $\eta(s)$. The interested readers are referred to Refs. [4,5,31] for additional information on extensions of the IMSRG method, such as different basis states, generator choices, higher-order contributions, and more.

In any solvers, the IMSRG flow is discretized using small finite step size ds, which is usually influential on the final results and thereby chosen adaptively. Utilizing the Magnus formulation of IMSRG, one can expect the results are insensitive to the choice of the step size [30]. I fixed the step size to $ds = 0.25$ in this work.

As a practical approach, it is common to divide IMSRG flows into multisteps by monitoring the norms of $\Omega(s)$ to prevent divergence and large computational time for evaluating deeply nested commutators. In other words, the transformed Hamiltonian is used as a reference point when the norm of $\Omega(s)$ exceeds a certain tolerance. At this point, the IMSRG flow is restarted from the pivot Hamiltonian. However, this partitioning introduces additional dependencies of the final results on the chosen tolerance. This is undesirable for the current purpose because it obfuscates the comparison between exact IMSRG results and their approximations using the proposed method. The issue is that the necessary number of such partitionings or splittings in exact IMSRG calculations cannot be predetermined. To circumvent this and to regard Magnus operators as smooth functions merely on the flow parameter s, I do not use such a multistep partitioning of the Magnus operators in this work. The only exception appears in the valence space problem (Sec. IV B), where one adopts a two-step process: First, the particle-hole decoupling, followed by the valence space decoupling for deriving effective interactions on a model space. For each step, any partitioning during the IMSRG flow is not used.

III. IMSRG-NET

Here, I describe the design of the neural network architecture and the training strategy for the proposed model, IMSRG-Net, and its computational cost.

A. Design of IMSRG-Net: Architecture and loss function

As shown in Fig. 1, the network architecture of IMSRG-Net is simple. The whole layers are the so-called fully connected layers (also referred to as Affine layers), which consist of the input layer, three hidden layers, and the output layer. The network is regarded as a function to give an approximation of the Magnus operators $\Omega^{\rm NN}(s)$ for larger values of s region giving converged results. The objective of the network is to predict not observables, but the Magnus operators. Hence, once the network is trained to predict $\Omega(s)$ accurately, it is guaranteed that the network reproduces the IMSRG results for any observables.

As summarized in Table I, the number of hidden layers is three, and the number of nodes for each hidden layer is 48, 48, and 16 from the input side to the output side, respectively. The adopted activation functions are the hyperbolic tangent for the first hidden layer and the softplus [32] for the remaining hidden layers.

To design the loss function to be minimized, let us define the following quantity:

$$\eta^{\rm NN}(s) = \frac{d\Omega^{\rm NN}(s)}{ds}, \tag{9}$$

where the superscript NN represents the values obtained through the neural network model. It should be noted that the above relation is an approximation in terms of the Magnus formulation of the IMSRG, taking the leading term of Eq. (6). This approximation is based on the fact that the derivatives of $\Omega^{\rm NN}$ can be evaluated easily and efficiently by automatic differentiation or numerical differentiation. Here I used the latter method, which is faster than the former one in our model having a large number of nodes in the output layer.

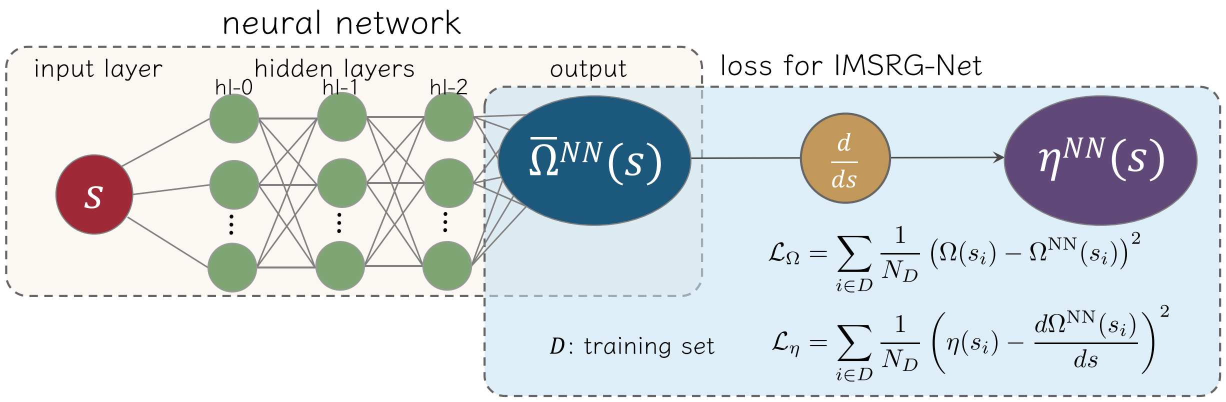
neural network
input layer
hidden layers
output
hl-0
hl-1
hl-2
s
$\bar{\Omega}^{NN}(s)$
loss for IMSRG-Net
$\frac{d}{ds}$
$\eta^{NN}(s)$
$\mathcal{L}_\Omega = \sum_{i\in D} \frac{1}{N_D}\left(\Omega(s_i) - \Omega^{\rm NN}(s_i)\right)^2$
D: training set
$\mathcal{L}_\eta = \sum_{i\in D} \frac{1}{N_D}\left(\eta(s_i) - \frac{d\Omega^{\rm NN}(s_i)}{ds}\right)^2$

FIG. 1. Schematic of IMSRG-Net. A fully connected neural network is used to generate approximated Magnus operators $\Omega^{\rm NN}(s)$ and their derivatives $\eta^{\rm NN}(s)$ are used as indicators of how well a neural network model respects the underlying IMSRG flow equation. Both the loss terms on Ω and η are back-propagated to train the network. The bar on $\Omega^{\rm NN}$ indicates that the network output in actual calculations is *scaled residual* of the Magnus operator. See Sec. III C for more details.

Now the loss function of IMSRG-Net is defined as the sum of the mean-squared errors (MSEs) for $\Omega^{\rm NN}$ and $\eta^{\rm NN}$:

$$\mathcal{L} = \mathcal{L}_\Omega + \lambda_\eta \mathcal{L}_\eta, \tag{10}$$

$$\mathcal{L}_\Omega = \sum_{i\in D} \frac{1}{N_D}[\Omega(s_i) - \Omega^{\rm NN}(s_i)]^2, \tag{11}$$

$$\mathcal{L}_\eta = \sum_{i\in D} \frac{1}{N_D}[\eta(s_i) - \eta^{\rm NN}(s_i)]^2, \tag{12}$$

where D is the training data set and N_D is the number of data. The parameter λ_η is introduced to balance the two terms. The typical size of $\mathcal{L}_\eta$ can be different from that of $\mathcal{L}_\Omega$. From our investigations, a rule of thumb to make training stable is to set $\lambda_\eta = 10^1$–10^3, which makes the contribution of Eq. (12) comparable to Eq. (11). I use a fixed value, $\lambda_\eta = 10^2$, throughout this work.

Under this design of the loss functions, IMSRG-Net can be regarded as a special case of physics-informed neural networks (PINNs) [25,26]. Each component of η contains information regarding the channels to be decoupled through the IMSRG flow. Such physics information is encoded through the $\mathcal{L}_\eta$ term, serving as a soft constraint that guides the network to learn the underlying principle—the IMSRG flow equation.

B. Hyperparameters

I explored various combinations of hyperparameters for neural network models: (i) optimizers: Stochastic gradient descent (SGD), Adam [33], AdamW [34], L-BFGS [35]; (ii) activation functions: Tanh, softplus, ReLU, Leaky ReLU, ELU; (iii) number of hidden layers: 1, 2, 3, 4; (iv) number of nodes in hidden layers: 4, 8, 16, 32, 64; (v) parameters for optimizers (learning rate, weight decay, momentum, etc.).

Naturally, these trials do not cover all the possible combinations of hyperparameters. Thus, it is important to remark that the architecture and hyperparameters adopted in this work represent only one of the viable choices that can achieve the desired accuracy for the target problems.

Regarding activation functions, one of the popular choices today is the rectified linear unit (ReLU), particularly in deep neural network models, e.g., for image recognition tasks, due to its computational efficiency and ability to mitigate the vanishing gradient problem. However, in this work, I found that smooth activation functions perform better than ReLU. This may be attributed to the fact that $\Omega(s)$ is a smooth function of s. For this reason, I employed the hyperbolic tangent (Tanh) function for the first hidden layer. The Softplus function, employed in the other hidden layers, achieves both smooth outputs and avoiding the vanishing gradient problem.

I employed the AdamW optimizer [34] to train the network, updating the model parameters through the backpropagation based on the loss function (10). Empirically, I observed that the model exhibits superior extrapolation performance when using AdamW compared with the Adam optimizer with the same hyperparameters. This may originate from the fact that the weight decay term in AdamW prevents the overfitting. During each epoch, I did not use the entire dataset at once, but rather used a subset of the training data. This approach, known as mini-batch learning, is a common technique to accelerate the learning process and to prevent overfitting. Specifically, I set the batch size equal to the number of data points, which is also referred to as online learning. By combining weight decay in AdamW and mini-batch learning (online learning), the extrapolation is achieved without sacrificing generalization ability.

The number of nodes in the hidden layers is fixed to 48 for upstream layers and 16 for the downstream layer. While the former one is almost irrelevant to the generalization

TABLE I. The architecture of IMSRG-Net. The size of output layer is given by dimension of the vectorized operator, which depends on $e_{\max}$.

Layer	Activation	Number of nodes	Bias term
hl-0	Tanh	48	Yes
hl-1	Softplus	48	Yes
hl-2	Softplus	16	Yes
Output	Identity	Dim. Ω vector	No

ability of our model, I found that a larger value around 48 speeds up the training process. On the other hand, the number of nodes in the last hidden layer closest to the output layer significantly influences both accuracy and computational cost. Generally, a larger value yields better accuracy on the training, validation, and test sets, but a smaller value is preferable in terms of computational efficiency. I found that sixteen is a good compromise between accuracy and VRAM usage.

The hyperparameters discussed above are kept fixed throughout the study to demonstrate the robustness of the proposed model to different target nuclei and input potentials.

C. Data and normalization

The operators of interests, Ω and η, are vectorized and divided into the three distinct data sets: Ten training data $\{s = 17.75, 18.0, \dots, 20.0\}$, five validation data $\{s = 16.5, \dots, 17.5\}$, and test data $\{s = \infty\}$. Here, $s = \infty$ corresponds to the point at which IMSRG(2) converged, with $||\eta(s)|| < \epsilon_{\rm norm}$ as the convergence criterion 10^{-6}, which is one employed in IMSRG codes [27,36]. The training set is used to train the network, while the test data is used to assess the accuracy of the extrapolation by the model. Using the validation data, I select the model that minimizes the sum of the training loss (10) and MSE of the validation set across all epochs as the best model. The number of training data points was determined by compromise between the number of data and the memory cost. As expected, using data points closer to $s = \infty$ naturally leads to higher accuracy and vice versa. However, in this study, the maximum value of s used as training data was set to 20. This value typically corresponds to the first 1/8 to 1/4 of the entire IMSRG flow. In this case, the results are not significantly affected by a particular number of data points as long as the number of training data exceeds about five.

In our numerical calculations, Ω values in network predictions are defined as residuals relative to a reference value $\Omega(s_t)$ such that s_t represents the largest value in the training set, which is $s_t = 20.0$ in the present study. This works as a normalization between different problems and justifies the consideration of multiple tasks using the same architecture. Additionally, the residuals of the Ω are multiplied by a global scaling factor. Normalizing high-dimensional vectors solely by the standard deviation of the data can sometimes lead to overflow issues. To address this, I have determined that the scaling factor, chosen as $C_\Omega = \min\{1/{\rm std}(\Omega), 10^3\}$, is suitable for the current model and target nuclei. It is important to note that the optimal scaling factor may vary depending on the specific problem, and an inappropriate choice can result in both overflow and underflow during calculations. To summarize, the network approximates the scaled residuals $\overline{\Omega}^{\rm NN}(s) \approx C_\Omega(\Omega(s) - \Omega(s_t))$, and those are rescaled and added the offset through

$$\Omega^{\rm NN}(s) = \overline{\Omega}^{\rm NN}(s)/C_\Omega + \Omega(s_t = 20), \tag{13}$$

to be compared with exact IMSRG data.

TABLE II. The number of elements and size of operator as a function of $e_{\rm max}$.

$e_{\rm max}$	Number of elements	Size (Float32) [MB]
4	34 334	0.1
6	467 076	1.8
8	3 526 724	13
10	18 345 624	70
12	73 842 358	282
14	246 457 592	940

D. Computational cost for IMSRG and IMSRG-Net

Here, I discuss the computational time for full IMSRG(2) calculation and IMSRG-Net, and estimate potential benefits of surrogating some part of IMSRG-flows by IMSRG-Net.

The computational cost of IMSRG-Net is primarily determined by the size of the output layer, which depends on the size of model space specified by $e_{\rm max}$ truncation. While it is desirable to use GPU accelerators for training neural network models, the VRAM usage often exceeds the capacity of standard GPGPUs for larger $e_{\rm max} > 10$. The dimension of the operator vector is primarily determined by the number of elements in the two-body part $\{\Omega^{(2)}_{ijkl}\}$. More precisely, one does not need to store all matrix elements of $\{\Omega^{(2)}_{ijkl}\}$ to vectorize it because of its anti-Hermitian nature. Table II summarizes the size of the vectorized operator as a function of $e_{\rm max}$. To construct fully connected neural network, these numbers are multiplied by the number of nodes in the hidden layer closest to the output layer, which governs the size of the weight parameters. In addition to storing those weight parameters, the VRAM is also used to store computational graphs, gradients, and other temporary variables needed in forward and backward propagations in neural network models. Despite the total size of operator vectors is depending on the choice of optimizers, VRAM usage easily exceeds 20 GB for $e_{\rm max}$ larger than 10. Consequently, the presented results are limited to smaller $e_{\rm max}$ values: 4, 6, 8, and 10.

To simplify the discussion on computational time, let us consider the $e_{\rm max} = 10$ cases. For full IMSRG(2) calculations for ^{16}O and ^{40}Ca with the EM500 interaction, it takes about six hours on an i9-10940X and four hours on a Xeon Platinum 8360Y. On the other hand, computation time for IMSRG-Net is independent of target nuclei and input interactions as long as the number of epochs is fixed. It takes 2.3 hours on a NVIDIA RTX A4000 and 1.2 hours on a NVIDIA A100. A reduction factor of computational time by IMSRG-Net, $(T_{s\leqslant 20} + T_{\rm IMSRGNet})/T_{\rm full}$, is around 0.4–0.6 in above cases, where $T_{s\leqslant 20}$ denotes computational time for IMSRG(2) calculations up to $s = 20$ to generate training data. It should be noted that the reduction factor for the proposed method can vary, because both the current IMSRG code in NuclearToolkit.jl [27] and the training strategy for neural network models may have rooms for improvement.

IV. RESULTS OF IMSRG-NET

A. Ground-state properties

In Fig. 2, the results of IMSRG-Net for ^{16}O and ^{40}Ca using two different input interactions are shown. The first

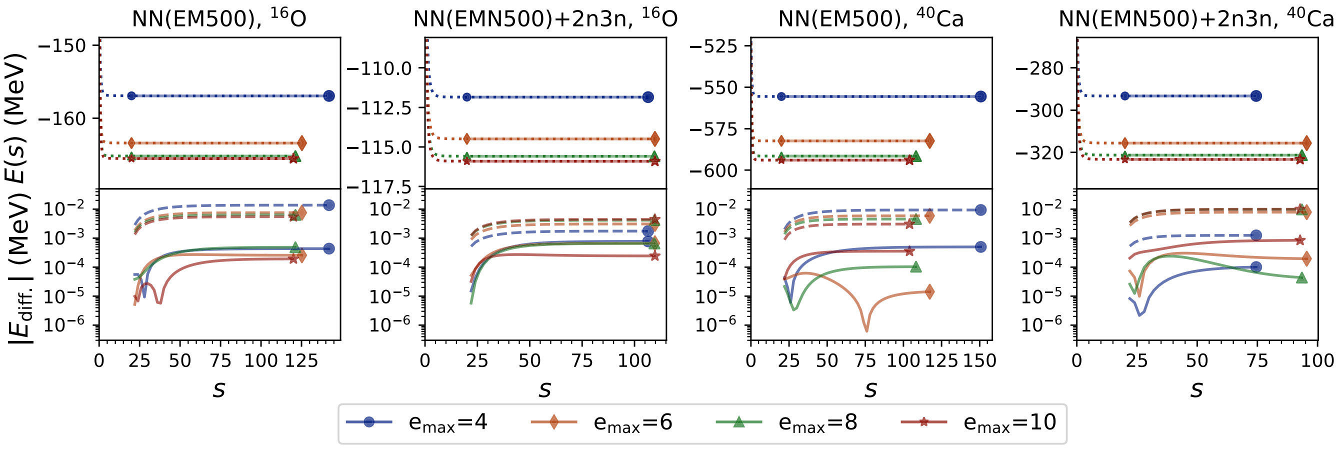

FIG. 2. Ground-state energies of ^{16}O and ^{40}Ca obtained using IMSRG(2) and IMSRG-Net. In the upper panels, the dashed lines correspond to the results of IMSRG(2). While the solid lines are the results of IMSRG-Net (for $s > 20.0$). The markers are placed on the right edge of the lines to represent the points where IMSRG(2) calculations converged. The lower panels illustrate the energy differences between IMSRG(2) and IMSRG-Net. The solid lines represent the results of IMSRG-Net with $\lambda_\eta = 10^2$, and the dashed ones represent results with $\lambda_\eta = 0$. Further information regarding the input potentials, NN(EM500) and NN(EMN500)+2n3n, can be found in the main text.

interaction, denoted as EM500, consists of the so-called EM nucleon-nucleon (NN) interaction [37] regularized by a cutoff 500 MeV and softened by the similarity renormalization group (SRG) with $\lambda = 2.0$ fm^{-1}. The other one, denoted as EMN500+2n3n, comprises the SRG evolved EMN interaction [38,39] up to N4LO. The "2n3n" is representing that a density-dependent three-nucleon force [40,41] is added to the softened EMN500 interaction. Each different symbol corresponds to results with different model spaces, specified by the $e_{\max}$.

The upper panels in Fig. 2 display the results obtained by IMSRG(2) (dashed lines) and IMSRG-Net (solid lines for $s > 20.0$) for different $e_{\max}$ truncations. Since it is difficult to discern the differences between the two models in this scale, the energy differences between IMSRG(2) and IMSRG-Net are shown in the lower panels. The solid lines in the lower panels are the results of IMSRG-Net, and the dashed ones represents cases where $\lambda_\eta = 0$. It is evident that incorporating the loss term on η is crucial for training the network and achieving better extrapolation in the larger s region. The extrapolations exhibit an accuracy of less than 1 keV.

Table III summarizes the ground-state energies and charge radii of ^{16}O and ^{40}Ca evaluated by IMSRG(2) and IMSRG-Net. The digits that differ between IMSRG(2) and IMSRG-Net are highlighted in bold. Since IMSRG-Net is designed to extrapolate not the energies but the Magnus operators Ω to achieve unitary transformations, Eqs. (4) and (5), one can compute the charge radii in a straightforward manner. From the Table III, it is apparent that the results of IMSRG-Net are in good agreement with those of IMSRG(2). In both cases, energies and charge radii, one can see that the deviation between IMSRG(2) and IMSRG-Net are much smaller than the residuals which are to be obtained through the rest IMSRG flow from $s = 20$ to $s = \infty$. This level of accuracy is considered sufficient to replace the remaining flow for $s > 20$ with the proposed model.

In Fig. 3, the learning curves of IMSRG-Net are plotted for ground-state calculations of ^{16}O and ^{40}Ca. Because of using online learning, MSE losses are inevitably shaggy, more like a band rather than a line. The corresponding moving averages of 100 epochs are plotted by the lines to see the trends of the learning curves.

As a whole, the learning curves show similar trends for different nuclei and input potentials. It occasionally encounters a plateau, and, after a while, the MSE suddenly drops. This trend can be understood as the effect of the AdamW optimizer. The weight decay term gradually reduces redundant degrees of freedom in network parameters that are ineffective at reducing the loss. Then, it leads to escaping the plateau and a subsequent drop in the training loss. While test errors on Magnus operators are generally one to two orders of magnitude worse than those for the training sets, the results of the observables in Table III indicate that the extrapolation is sufficiently accurate. Looking at the dashed lines in Fig. 3, we can see that the $\mathcal{L}_\eta$ term is also well suppressed after training and the approximation introduced in Eq. (9) works well to inform the network of information of η.

It should be noted that the numbers (energies and charge radii) presented in this work are based on a single run specifying a random seed. The results can vary in different environments and with different random seeds, they can either be better or worse than the values reported here. The uncertainty resulting from such optimization process is small, on the same order of magnitude as the values in the boldface in Table III. To reproduce the results, one can refer to the provided repository on GitHub [29], which contains the necessary information and code.

B. Valence space IMSRG

As a natural expectation, one may consider applications of IMSRG-Net to valence-space IMSRG (VS-IMSRG) for deriving valence space effective interactions and operators. To this end, let us consider the VS-IMSRG utilizing IMSRG-Net with the same architecture above. The dataset is constructed simultaneously as the IMSRG case, utilizing $\Omega(s)$ up to $s =$

TABLE III. The ground-state energies and charge radii of ^{16}O and ^{40}Ca by IMSRG(2) and IMSRG-Net. The $s = \infty$ corresponds to the one giving converged value of IMSRG(2).

			Energy (MeV)				R_{ch} (fm)			
			$s = 20$		$s = \infty$		$s = 20$		$s = \infty$	
Target	Interaction	e_{max}	IMSRG(2)	IMSRG-Net	IMSRG(2)	IMSRG-Net	IMSRG(2)	IMSRG-Net	IMSRG(2)	IMSRG-Net
^{16}O	EM500	4	−156.9474	−156.9474	−156.96**11**	−156.96**07**	2.2578	2.2578	2.261**2**	2.261**0**
		6	−163.4079	−163.4079	−163.415**3**	−163.415**0**	2.2526	2.2526	2.254**7**	2.254**6**
		8	−165.1876	−165.187**5**	−165.19**32**	−165.19**27**	2.2482	2.2482	2.249**9**	2.249**7**
		10	−165.5309	−165.5309	−165.535**9**	−165.535**7**	2.2469	2.2469	2.248**5**	2.248**4**
	EMN500+2n3n	4	−111.8453	−111.8453	−111.84**70**	−111.84**62**	2.3600	2.3600	2.360**7**	2.360**5**
		6	−114.4895	−114.4895	−114.49**25**	−114.49**18**	2.3681	2.3681	2.369**2**	2.369**0**
		8	−115.5894	−115.5894	−115.59**30**	−115.59**24**	2.3735	2.3735	2.374**8**	2.374**6**
		10	−115.9040	−115.9040	−115.90**79**	−115.90**82**	2.3751	2.3751	2.3765	2.3765
^{40}Ca	EM500	4	−555.6791	−555.6791	−555.68**84**	−555.68**79**	2.5947	2.5947	2.595**9**	2.595**8**
		6	−582.4293	−582.4293	−582.435**0**	−582.435**1**	2.5960	2.5960	2.5967	2.5967
		8	−591.5783	−591.578**2**	−591.582**2**	−591.582**1**	2.5915	2.5915	2.5920	2.5920
		10	−594.0215	−594.0215	−594.024**2**	−594.024**6**	2.5890	2.5890	2.589**4**	2.589**5**
	EMN500+2n3n	4	−293.3474	−293.3474	−293.348**6**	−293.348**7**	2.8579	2.8579	2.8581	2.8581
		6	−315.6334	−315.6334	−315.64**11**	−315.64**09**	2.9082	2.9082	2.908**9**	2.908**8**
		8	−321.323**3**	−321.323**2**	−321.33**20**	−321.33**19**	2.920**1**	2.920**0**	2.920**9**	2.920**8**
		10	−323.35**19**	−321.35**20**	−323.36**05**	−323.36**13**	2.9252	2.9252	2.9260	2.9260

20 as the training and validation set. Note that the $s = \infty$ point for IMSRG(2) now serves as the starting point for the VS-IMSRG flow, i.e., $s = 0$.

Figure 4 illustrates the correlation plot between the effective interactions derived by VS-IMSRG(2) and IMSRG-Net. These are effective interactions on the sd- and pf-shell model spaces employing the same input interactions as mentioned above. The reference states are chosen as the nearest doubly magic nuclei ^{16}O and ^{40}Ca. As a whole, the deviation is small, which is typically less than 10^{-8} in mean squared error, with a few exceptions for $e_{max} \geqslant 8$ corresponding the symbols located far from the diagonal lines in Fig. 4. These exceptions correspond to the two-body matrix elements relevant to higher orbits in the valence space, $0d_{3/2}$ for the sd shell and $0f_{5/2}$ for the pf shell. It is found that the calculation of the commutator in the BCH formula exhibits nonconvergent behavior during the VS-IMSRG flow for these cases. For this reason, the failure in some components is attributed to not IMSRG-Net, but the numerical instability in VS-IMSRG. When one employs the multistep partitioning or pivoting mentioned in Sec. II, such a numerical instability can be mitigated. However, as mentioned earlier, the question remains as to how many times partitioning can occur up to a certain flow parameter, i.e., at what point can the Magnus operator be regarded as a continuous function of s and the IMSRG-Net be used to replace subsequent flows?

V. SUMMARY AND OUTLOOK

This study explores the possibility of employing a machine-learning-based model as an alternative solver for in-medium similarity renormalization group (IMSRG). By utilizing the proposed neural network, IMSRG-Net, inspired by physics-informed neural networks (PINNs), the extrapola-

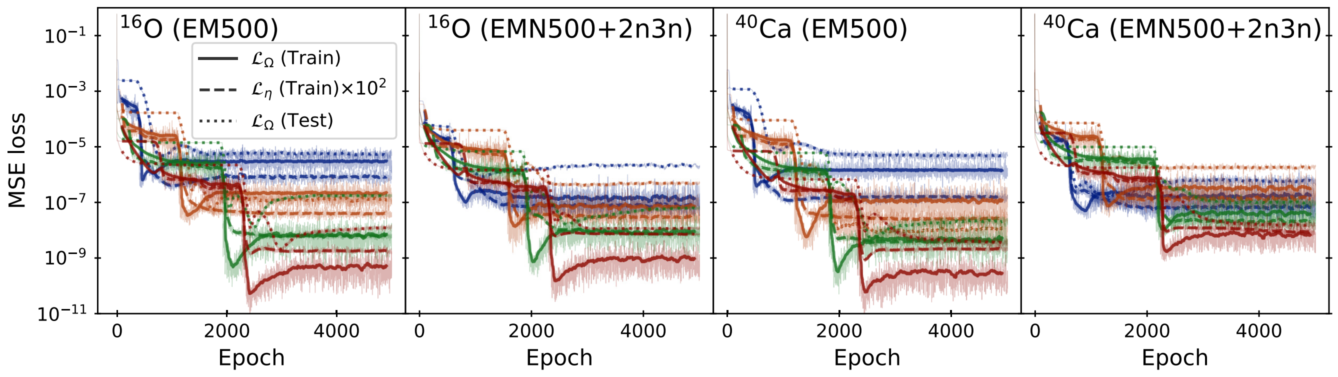

FIG. 3. Learning curve of IMSRG-Net calculations. The lines represent moving average over 100 epochs to see the trend. The colors for different e_{max} truncation and abbreviations of input potentials are similar to Fig. 2.

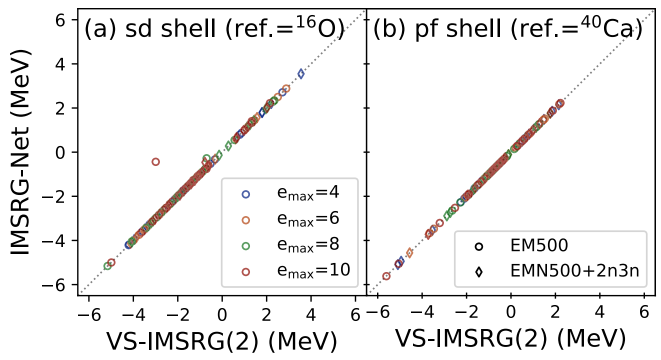

FIG. 4. Correlation plot of effective interactions derived by VS-IMSRG(2) and IMSRG-Net, (a) *sd* shell and (b) *pf* shell. This includes both single-particle energies and two-body matrix elements, and some are randomly omitted since the number of parameters are too large to plot them all. The circle and diamond symbols represent the results of EM500 and EMN5002n3n, respectively.

tion is achieved with a satisfactory level of accuracy. Energy extrapolation was accomplished with an accuracy of less than 1 keV, and the charge radii were also well reproduced with errors of approximately 0.0001 fm. Once accurate approximations of the Magnus operators $\Omega(s)$ are obtained, any desired operators of interest can be evolved accurately using the Magnus formulation of IMSRG.

While I have empirically investigated the effectiveness of IMSRG-Net on various datasets, it is important to acknowledge that there may always be exceptions and cases where this network architecture or learning strategy encounters some limitations. One example is the issue of numerical instability within IMSRG itself, as demonstrated by the valence space results. To further enhance the accuracy of extrapolated results and improve robustness against different target interactions and nuclei, additional inductive biases may need to be introduced.

This work may herald new approaches for IMSRG methods and offer numerous possibilities for applying PINNs-like techniques to other nuclear many-body machinery. On the other hand, from a practical perspective on computational time, the speeding-up effect of IMSRG calculations through IMSRG-Net is limited, as discussed in Sec. III D. It would be crucial to develop more computationally efficient versions of IMSRG-Net. Since the current IMSRG-Net directly employs $\Omega(s)$ with large dimensions as the output layer of the neural network, it becomes challenging to store and manipulate all of them on GPGPU for cases with $e_{\max}$ larger than 10. This limitation prevents the application of IMSRG-Net for cutting-edge IMSRG calculations having larger $e_{\max}$. For instance, working within a latent space using encoder and decoder (e.g., Ref. [42]) or constructing low-rank operator expressions while retaining maximal information through techniques like tensor decomposition could be effective.

ACKNOWLEDGMENTS

This work was supported by JSPS KAKENHI (Grant No. 22K14030) and JGC-Saneyoshi Scholarship Foundation. This research was partly conducted using the FUJITSU Supercomputer PRIMEHPC FX1000 and FUJITSU Server PRIMERGY GX2570 (Wisteria/BDEC-01) at the Information Technology Center, The University of Tokyo.

[1] K. Tsukiyama, S. K. Bogner, and A. Schwenk, Phys. Rev. Lett. **106**, 222502 (2011).
[2] K. Tsukiyama, S. K. Bogner, and A. Schwenk, Phys. Rev. C **85**, 061304(R) (2012).
[3] H. Hergert, S. Bogner, T. Morris, A. Schwenk, and K. Tsukiyama, Phys. Rep. **621**, 165 (2016).
[4] S. R. Stroberg, H. Hergert, S. K. Bogner, and J. D. Holt, Annu. Rev. Nucl. Part. Sci. **69**, 307 (2019).
[5] H. Hergert, Front. Phys. **8**, 379 (2020).
[6] T. Miyagi, S. R. Stroberg, P. Navrátil, K. Hebeler, and J. D. Holt, Phys. Rev. C **105**, 014302 (2022).
[7] B. Hu, W. Jiang, T. Miyagi, Z. Sun, A. Ekström, C. Forssén, G. Hagen, J. D. Holt, T. Papenbrock, S. R. Stroberg, and I. Vernon, Nat. Phys. **18**, 1196 (2022).
[8] D. Frame, R. He, I. Ipsen, D. Lee, D. Lee, and E. Rrapaj, Phys. Rev. Lett. **121**, 032501 (2018).
[9] A. Sarkar and D. Lee, Phys. Rev. Lett. **126**, 032501 (2021).
[10] A. Sarkar and D. Lee, Phys. Rev. Res. **4**, 023214 (2022).
[11] S. König, A. Ekström, K. Hebeler, D. Lee, and A. Schwenk, Phys. Lett. B **810**, 135814 (2020).
[12] P. Demol, T. Duguet, A. Ekström, M. Frosini, K. Hebeler, S. König, D. Lee, A. Schwenk, V. Somà, and A. Tichai, Phys. Rev. C **101**, 041302(R) (2020).
[13] P. Demol, M. Frosini, A. Tichai, V. Somà, and T. Duguet, Ann. Phys. (NY) **424**, 168358 (2021).
[14] R. Furnstahl, A. Garcia, P. Millican, and X. Zhang, Phys. Lett. B **809**, 135719 (2020).
[15] J. Melendez, C. Drischler, A. Garcia, R. Furnstahl, and X. Zhang, Phys. Lett. B **821**, 136608 (2021).
[16] C. Drischler, M. Quinonez, P. Giuliani, A. Lovell, and F. Nunes, Phys. Lett. B **823**, 136777 (2021).
[17] S. Wesolowski, I. Svensson, A. Ekström, C. Forssén, R. J. Furnstahl, J. A. Melendez, and D. R. Phillips, Phys. Rev. C **104**, 064001 (2021).
[18] D. Bai and Z. Ren, Phys. Rev. C **103**, 014612 (2021).
[19] M. Companys Franzke, A. Tichai, K. Hebeler, and A. Schwenk, Phys. Lett. B **830**, 137101 (2022).
[20] S. Yoshida and N. Shimizu, Prog. Theor. Exp. Phys. **2022**, 053D02 (2022).
[21] N. Yapa, K. Fossez, and S. König, Phys. Rev. C **107**, 064316 (2023).
[22] J. A. Melendez, C. Drischler, R. J. Furnstahl, A. J. Garcia, and X. Zhang, J. Phys. G **49**, 102001 (2022).
[23] E. Bonilla, P. Giuliani, K. Godbey, and D. Lee, Phys. Rev. C **106**, 054322 (2022).
[24] 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, and L.-G. Pang, Rev. Mod. Phys. **94**, 031003 (2022).

[25] M. Raissi, P. Perdikaris, and G. Karniadakis, J. Comput. Phys. **378**, 686 (2019).

[26] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, Nat. Rev. Phys. **3**, 422 (2021).

[27] S. Yoshida, J. Open Source Software **7**, 4694 (2022); https://github.com/SotaYoshida/NuclearToolkit.jl.

[28] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai *et al.*, *Advances in Neural Information Processing Systems 32* (Curran Associates, Red Hook, New York, 2019), pp. 8024–8035.

[29] S. Yoshida, https://github.com/SotaYoshida/IMSRGNet.

[30] T. D. Morris, N. M. Parzuchowski, and S. K. Bogner, Phys. Rev. C **92**, 034331 (2015).

[31] M. Heinz, A. Tichai, J. Hoppe, K. Hebeler, and A. Schwenk, Phys. Rev. C **103**, 044318 (2021).

[32] H. Zheng, Z. Yang, W. Liu, J. Liang, and Y. Li, *Proceedings of the 2015 Inter-national Joint Conference on Neural Networks (IJCNN)* (IEEE, Piscataway, NJ, 2015), pp. 1–4.

[33] D. P. Kingma and J. Ba, in *Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings*, edited by Y. Bengio and Y. LeCun (2015).

[34] I. Loshchilov and F. Hutter, arXiv:1711.05101.

[35] D. C. Liu and J. Nocedal, Math. Program. **45**, 503 (1989).

[36] S. R. Stroberg, https://github.com/ragnarstroberg/imsrg.

[37] D. R. Entem and R. Machleidt, Phys. Rev. C **68**, 041001(R) (2003).

[38] D. R. Entem, N. Kaiser, R. Machleidt, and Y. Nosyk, Phys. Rev. C **91**, 014002 (2015).

[39] D. R. Entem, R. Machleidt, and Y. Nosyk, Phys. Rev. C **96**, 024004 (2017).

[40] M. Kohno, Phys. Rev. C **88**, 064005 (2013); **96**, 059903(E) (2017).

[41] S. Yoshida, arXiv:2208.02464.

[42] D. Di Sante, M. Medvidović, A. Toschi, G. Sangiovanni, C. Franchini, A. M. Sengupta, and A. J. Millis, Phys. Rev. Lett. **129**, 136402 (2022).