

Learning nuclear cross sections across the chart of nuclides with graph neural networks

Hongjun Choi^{1,*}, Sinjini Mitra², Jason Brodksy¹, Ruben Glatt¹, Erika Holmbeck¹, Shusen Liu¹,
Nicolas Schunck¹, Andre Sieverding¹, and Kyle Wendt¹

¹*Lawrence Livermore National Laboratory, Livermore, California 94551, USA*

²*School of Electrical, Computer and Energy Engineering, Arizona State University, 1151 S Forest Avenue Tempe, Arizona 85281, USA*



(Received 2 April 2024; revised 28 April 2025; accepted 22 August 2025; published 1 October 2025)

We explore the use of deep learning techniques to learn how nuclear cross sections change as we add or remove protons and neutrons. As a proof of principle, we focus on the neutron-induced reactions in the fast energy regime. Our approach follows a two-stage learning framework. First, we apply representation learning to encode cross section data into a latent space using either variational autoencoders (VAEs) or implicit neural representations (INRs). Then, we train graph neural networks (GNNs) on the resulting embeddings to predict missing values across the nuclear chart by leveraging the topological structure of neighboring isotopes. We demonstrate accurate cross section predictions within a 9×9 block of missing nuclei. We also find that the optimal GNN training strategy depends on the type of latent representation used, with VAE embeddings performing best under end-to-end optimization in the original space, while INR embeddings achieve better results when the GNN is trained only in the latent space. Furthermore, using clustering algorithms, we map groups of latent vectors into regions of the nuclear chart and show that VAEs and INRs can discover some of the neutron magic numbers. These findings suggest that deep-learning models based on the representation encoding of cross sections combined with graph neural networks hold significant potential in augmenting nuclear theory models, e.g., by providing reliable estimates of covariances of cross sections, including cross-material covariances.

DOI: [10.1103/4jd4-bnyh](https://doi.org/10.1103/4jd4-bnyh)

I. INTRODUCTION

Nuclear cross sections encode the probability of a subatomic particle to interact with an atomic nucleus, defined by its proton and neutron numbers (Z, N) [1]. Each atomic nucleus can be involved in many different types of nuclear reactions often simply denoted as (*incoming process, outgoing process*), also called “channels.” For example, the (n, γ) channel involves the absorption of a neutron and the emission of one or several photons (γ rays) while the (n, f) channel involves the absorption of a neutron resulting in the fission of the nucleus. Nuclear reaction libraries are formed by the set of all known cross sections for all known nuclei [2–4]. Such libraries are essential inputs to nuclear technology applications [5], which include energy production [6], nuclear medicine [7], and nuclear forensics [8]. They also play a critical role in our attempts at understanding the formation of elements in the universe [9,10]. The importance of such libraries, and of nuclear data in general, in scientific research was recently underlined in the 2023 Long Range Plan for Nuclear Science

commissioned by the U.S. Department of Energy and the National Science Foundation [11].

Measuring nuclear cross sections requires complex and expensive experimental facilities and is often impossible for short-lived radioactive nuclei that cannot be made into targets or created in radioactive beams. While nuclear reaction theory has greatly improved over the past decades [12], its level of sophistication varies greatly depending on the nucleus and type of reaction. In very light nuclei, *ab initio* theory can successfully describe reactions at low energy in a fully quantum-mechanical framework [13]; in contrast, reactions of heavy nuclei with neutrons with energy of just a few sub-eV already lead to the formation of a highly excited nucleus with an extremely complex structure that can only be described within a statistical phenomenology framework [14,15]. In practice, such statistical reaction theory relies on additional theoretical inputs describing nuclear properties such as the nuclear level density; the probability of capturing incident particles; the rate of decay through the emission of photons, neutrons, and protons; etc. [16]. Today the best nuclear data libraries available largely rely on phenomenological models to determine such properties. These models are carefully calibrated to match realistic application requirements such as the criticality regimes of a nuclear reactor, at the price of a loss in generality and predictive power of the underlying theory [4,17]. As a result, current models fail at consistently predicting the complex relationships between different types of cross sections in a single nucleus, or between cross

*Contact author: choi22@llnl.gov

Published by the American Physical Society under the terms of the [Creative Commons Attribution 4.0 International](https://creativecommons.org/licenses/by/4.0/) license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

sections in different nuclei across the chart of isotopes. This lack of predictive power is a significant impediment to efforts at designing new, cleaner, or more efficient nuclear technologies. While research in fundamental reaction theory continues to progress, techniques originating from artificial intelligence and machine learning research offer an alternative viewpoint [18–21].

Indeed, each single cross section is a continuous one-dimensional function of the energy of the incident particle (akin to a time sequence) labeled by the number of protons Z and neutrons N . In neutron-induced reactions in the fast regime (when the energy E of the incident neutron is $E > 100$ keV), cross sections are rather smooth functions of neutron energy (notwithstanding threshold effects). Therefore, dimensionality reduction techniques could in principle be applied to compress information and learn from the data the physically relevant latent space that underpin the cross sections.

In addition, cross sections of nuclei with close (Z, N) numbers often tend to have similar forms, suggesting a correlation with the position on the two-dimensional (2D) grid structure spanned by Z and N . This observation suggests employing geometric deep learning for learning meaningful representations in the latent space, a technique that has recently emerged as an impactful subfield adept at learning non-Euclidean subspaces [22]. Graph neural networks (GNNs) [23] in particular provide an easy way to perform node-level, edge-level, and graph-level prediction tasks while being permutation invariant for structured gridlike data in such non-Euclidean subspaces, as has been shown for particle reconstruction [24].

The goal of this paper is to effectively learn the intricate relationships between cross sections of different nuclei by leveraging such state-of-the-art deep learning models. To this end, we employ two complementary techniques in succession. As the first step, we use either a variational autoencoder (VAE) [25] or an implicit neural representation (INR) to uncover the latent space of nuclear cross sections and reduce the dimensionality of the problem. In the second step, we use GNNs to account for the implicit 2D geometric nature of the nuclear chart, where the graph neighborhood is defined by similar Z and N numbers, and the learned latent spaces are used as graph node features. The resulting network provides an efficient and consistent framework to predict unknown cross sections not too far from the training region.

This paper is organized as follows: Section II gives a broad overview of the challenges facing nuclear theory when computing nuclear data. In Sec. III, we summarize the main aspects of our theoretical approach, including the architecture of the network and the characteristics of the datasets used for training. Section IV shows the results for different GNN architectures and encoder approaches. Section V discusses our results in the context of related work on nuclear data before we summarize our findings in Sec. VI.

II. THEORETICAL MODELS OF NUCLEAR DATA: CHALLENGES

As mentioned in Sec. I, nuclear reaction libraries are formed by the set of all known cross sections for all known

nuclei. Most of these data are not simply a set of experimental data points but the result of an *evaluation* of all available data by subject matter experts. This evaluation is itself the outcome of a complex, multistage, iterative process that involves (i) carefully analyzing available experimental results (vetting outliers, estimating uncertainties, etc.); (ii) adjusting model parameters of nuclear reaction codes such as TALYS [26] or EMPIRE [27] in such a way that the cross sections computed with such codes “match,” or are close to, experimental measurements; (iii) *processing* the data (energy binning, heating of cross sections to room temperature values, etc.); and (iv) verifying the nuclear reaction library through *benchmarks*, e.g., running a suite of codes simulating critical assemblies.

From a physics perspective, nuclear reactions are extremely complex processes. Ideally, one would like to describe any reaction as a quantum-mechanical evolution from an initial state, e.g., an incoming particle a with energy E_a , spin J_a , and parity π_a impinging on a target at rest A with energy E_A , spin J_A , and parity π_A , to a final state such as two reaction products b and B with respective energies E_b (E_B), spins J_b (J_B), and parities π_b (π_B). Such a description would require the perfect characterization of the quantum-mechanical state of all reaction products a , A , b , and B as well as the exact knowledge of the nuclear many-body Hamiltonian $\hat{H}$, since the unitary quantum-mechanical time evolution operator is $e^{-i\hat{H}t/\hbar}$. Both are out of reach of current nuclear theory.

In practice, the physics of a nuclear reaction is instead described by a suite of models with varying degrees of sophistication and phenomenology. These models play an essential role in the evaluation process and can be divided into several broad categories: (i) optical models describe the interaction of an incoming particle a with a target A ; (ii) level density models characterize the number of available energy levels $\rho(E, J, \pi)$ at a given excitation energy E , angular momentum J , and parity π ; (iii) gamma-strength functions $\gamma_{\text{SF}}(E)$ characterize the probabilities of emitting photons as a function of energy E ; (iv) fission transmission coefficients quantify the probability for a (heavy) nucleus to undergo fission as a function of energy, angular momentum, and parity; and (v) preequilibrium reaction models describe interaction mechanisms between projectile and target that involve specific nuclear excited states.

All these models are just different facets of the ideal quantum-mechanical evolution described above. This implies that they should all be related to one another, at the very least indirectly. For example, let us assume one were to describe the structure of an atomic nucleus with a self-consistent mean-field model based on some effective nucleon-nucleon potential [28]. How this mean field is computed or parametrized should directly impact the optical potential (the real part of the optical potential does represent the nuclear mean field), the nuclear level density (through the excited states associated with the mean-field potential), the gamma-strength function (which is typically related to functional derivatives of the mean field with respect to nuclear densities), fission transmission coefficients (which depend on how the mean-field potential changes with deformation), etc. Moreover, since the same effective nucleon-nucleon potential would be used to compute every single nucleus, nuclear properties would also

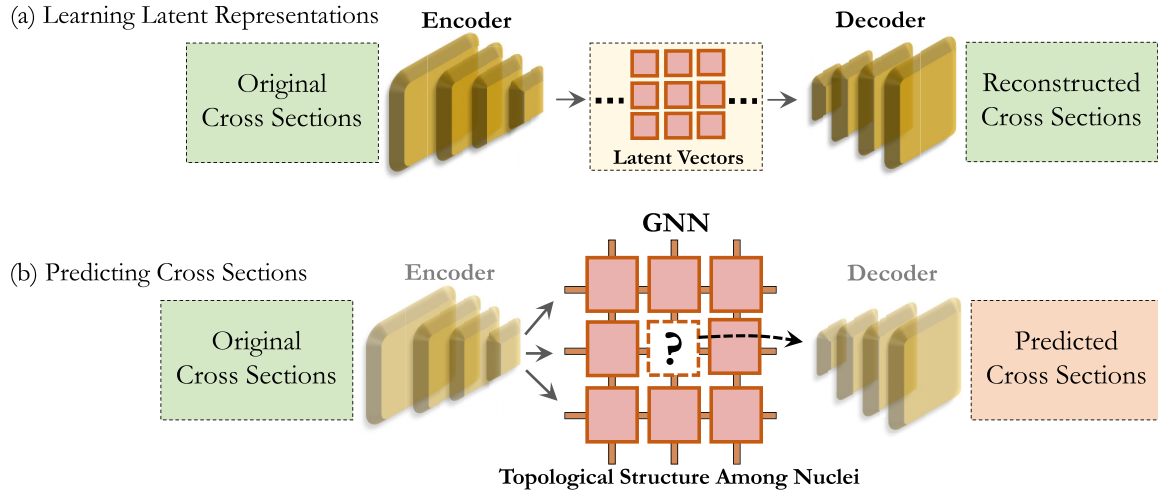


FIG. 1. Schematic representation of our framework. The initial data are nuclear cross sections represented as a vector indexed by the energy of the incident particle. (a) The data are encoded into a latent representation using either a variational autoencoder (VAE) or an implicit neural representation (INR) network (representation learning). (b) A graph neural network (GNN) then learns how the data in the latent space in nucleus (Z, N) transform into neighboring nuclei in order to make predictions in nucleus (Z', N') . The decoder then transforms the predicted vector in the latent space back into the space of nuclear cross sections (latent space predictions).

be heavily correlated as a function of proton and neutron number.

Therefore, even if such a theoretical model still contains parameters fitted to experimental data—in this case, the parameters of the effective nucleon-nucleon force—all nuclear properties are directly related to that single set of parameters. In a sensitivity analysis, one would find that variations of the model parameters result in *correlated* variations of the model outputs, i.e., the nuclear data properties of interest. In current approaches based on separate models for each type of nuclear data, such correlated variations, which manifest relationships between different data, are impossible to capture with nuclear theory alone.

Instead, each nuclear model is treated as a separate black box independently parametrized by a handful of parameters. In a typical evaluation process, these parameters are adjusted in such a way that nuclear cross sections match experimental measurements and the resulting benchmarks are satisfied [2–4,29]. These fits will often take the form of a functional dependency of model parameters in terms of proton and neutron number (or atomic mass number A), excitation energy, or angular momentum. The nuclear reaction codes mentioned above typically implement a few different variants of such systematics. By construction, these systematics become highly questionable in nuclei or at energies that are beyond the range of their fit, which directly impacts the reliability of nuclear data libraries for extrapolations.

By employing a collection of independent nuclear models locally calibrated to experimental data, it thus becomes nearly impossible to (1) make reliable predictions outside the range of currently available measurements and (2) capture the relationships (or correlations) between nuclear data within the same nucleus, or between different nuclei. In this paper, we will show that deep-learning techniques provide an alternative solution to both points.

III. METHODOLOGY

The limited number of available data for individual cross sections together with the potentially complex relationships between cross sections of different nuclei can make the training of deep learning models challenging. To overcome such a challenge, we separate the overall learning problem into two interconnected components, namely a representation learning task and a prediction task, as shown in Fig. 1. In the representation learning task, we are learning the low-dimensional, latent space representation of cross sections; in the prediction task, we make predictions to unknown nuclei from such latent representations and use the decoder trained in the first task to bring them back into the original space of cross sections. In contrast to the space of cross sections, the latent space provides a more compact yet meaningful representation that allows the model to learn more rapidly and generalize better. Additionally, the separation of these two steps ensures a clear delineation between the process of learning the representation, which may require data augmentation, i.e., interpolation and scaling of the cross section data, and the actual prediction tasks, which should only involve the cross section data that physically exist, i.e., a cross section that is not augmented with additional, synthetic data or modified.

As schematically illustrated in Fig. 1(a), the representation learning component will focus on learning the underlying latent representation space of all known cross sections. In the following discussion, we denote $\sigma_\alpha(E)$ as the cross section of the nuclear reaction (or channel) α as a function of the energy E of the incident particle, where the energy is discretized on a finite grid $E \rightarrow \{E_o\}_{o=1,m}$, where m is the dimension of the original data vector. For the encoding of the cross section, we train either a 1D-convolutional VAE [30] or an encoding-decoding hypernetwork [31] that predicts the weights for the INR model [32] on data from s nuclei to learn a k -dimensional latent space for various cross sections. By design, $k < m$ so

that the size of the feature matrix is reduced to $\mathbf{X} \in \mathbb{R}^{s \times k}$ while keeping the implicit grid intact.

For the prediction task shown in Fig. 1(b), we rely on a GNN approach. The available cross section data in the latent space can be represented as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each nucleus is represented by a node $v_i \in \mathcal{V}$ (the set of total nodes) while the relationship between nodes v_i, v_j constitutes an edge $e_{ij} \in \mathcal{E}$ (the set of edges). Each node v_i with $i \in [0, s-1]$ consists of a k -dimensional feature vector $\mathbf{x}_i$, defining the feature matrix $\mathbf{X} \in \mathbb{R}^{s \times k}$.

A. Dataset

Our dataset is a subset of the TENDL dataset of nuclear cross sections [29]. In practice, we only consider the (n, el) (i.e., elastic), (n, n') (i.e., inelastic), (n, γ) , and $(n, 2n)$ cross sections for all nuclei included in TENDL. Since cross sections are probabilities per unit area, their values are always positive: $\sigma_\alpha(E) \geq 0$ for all channels α . The TENDL dataset used contains $s = 2171$ nuclei and can be represented in a 2D grid corresponding to the chart of isotopes and labeled as (Z, N) with Z the proton number and N the neutron number of the nucleus. For the elastic, inelastic, (n, γ) , and $(n, 2n)$ channels the original cross section is defined on an energy grid of $m = 256$ points regularly spaced between 0 and 20 MeV. The values of each type of cross section are normalized by dividing by their maximum across the dataset prior to network training. After training, the data are then embedded into a lower k -dimensional latent space with $k = 32$ using either a pretrained VAE or INR model, resulting in a final data matrix of size $(s \times k) = (2171 \times 32)$, which serves as input for the downstream prediction task.

The size of the TENDL dataset is rather small to train deep neural networks. This can be mitigated using data augmentation techniques. In the representation learning task, we only focus on the cross section data, i.e., describing curves rather than any underlying physical relationship and properties. The goal is simply to construct a continuous, multidimensional latent space of curve shapes. Since we do not directly assign any physical properties to such a space, we can adopt an interpolation-based data augmentation strategy, in which we produce new training examples by linearly interpolating between two randomly picked cross section samples with weights that add up to 1.0. Essentially the latent space thus generated provides a superset of all valid cross sections while at the same time allowing for a smooth transition between them for better optimization in the cross section prediction task with the GNN. For the latter, only actual cross sections in existing nuclei are used as training data. We found that this type of data augmentation improves the encoding performance of the VAE model, whereas the INR model does not benefit from it during training.

B. Representation learning

Representation learning aims to condense raw input data into a concise format that preserves crucial information. Ideally, the learned representation provides a lower-dimensional space which makes subsequent learning tasks

more efficient. In our setup, the goal of representation learning is to learn the latent space of all cross sections. By doing so, not only can all known cross sections of a nucleus be encoded in the latent space, we can also predict new cross sections for known and unknown nuclei. This prediction task then just translates into the task of finding the right vector in the learned representation space.

1. Variational autoencoder

One of the most commonly used approaches for representation learning is an autoencoder (AE) architecture. An AE is a neural network designed to encode input data into a lower-dimensional space (i.e., bottleneck) and then decode it back to the original data space. The autoencoding objective is centered around minimizing the discrepancy between the original input and the reconstructed output. For variational autoencoders (VAEs) [25], the standard autoencoding objective is augmented by the inclusion of variational inference [33], where the encoding process maps input data not to a fixed point but to a probability distribution in the latent space. This dual training objective involves both minimizing the reconstruction error and aligning the latent space distribution with a specified form, i.e., a standard Gaussian distribution (with parameters μ and σ), as illustrated in Fig. 2(a). Specifically, the VAE training loss can be defined by two key terms: The reconstruction loss and the Kullback-Leibler (KL) divergence,

$$\mathcal{L}_{\text{VAE}} = -\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] + \lambda D_{\text{KL}}(q_\phi(z|x)||p(z)). \quad (1)$$

The first term encourages the VAE to accurately reconstruct the original input data. It measures the negative log-likelihood of the decoder distribution $[p_\theta(x|z)]$ generating data that resembles the input (x) given a sampled latent code (z) from the encoder distribution $[q_\phi(z|x)]$. The second term acts as a regularizer, forcing the learned latent distribution $[q_\phi(z|x)]$ to be similar to a prior distribution $[p(z)]$, i.e., a standard normal Gaussian. The hyperparameter λ controls the balance between reconstruction accuracy and the regularity of the latent space. In this work, we set λ to 0.1.

The integration of variational inference ensures a continuous representation space and makes the network generative: we can generate diverse and realistic new data simply by sampling from the learned latent space distribution and applying the decoder. Overall, VAEs provide a powerful framework for unsupervised representation learning, while at the same time facilitating high-quality reconstruction and generation.

In this work, the encoder of the VAE model comprises a sequence of convolutional layers followed by rectified linear unit (ReLU) activation functions [34] and a flattening operation. This architecture is designed to progressively reduce the spatial dimensions of the inputs while increasing the number of feature channels. Our encoder includes four convolutional layers—denoted conv_1, conv_2, conv_3, and conv_4 in Fig. 2(a)—with increasing filters (32, 64, 128, 256), kernel size 4, and stride 2. We then convert the output of convolutional layers into a flattened vector of size k .

The decoder comprises a sequence of transposed convolutional layers aimed at reconstructing the original inputs from the latent space. It also employs ReLU activations and

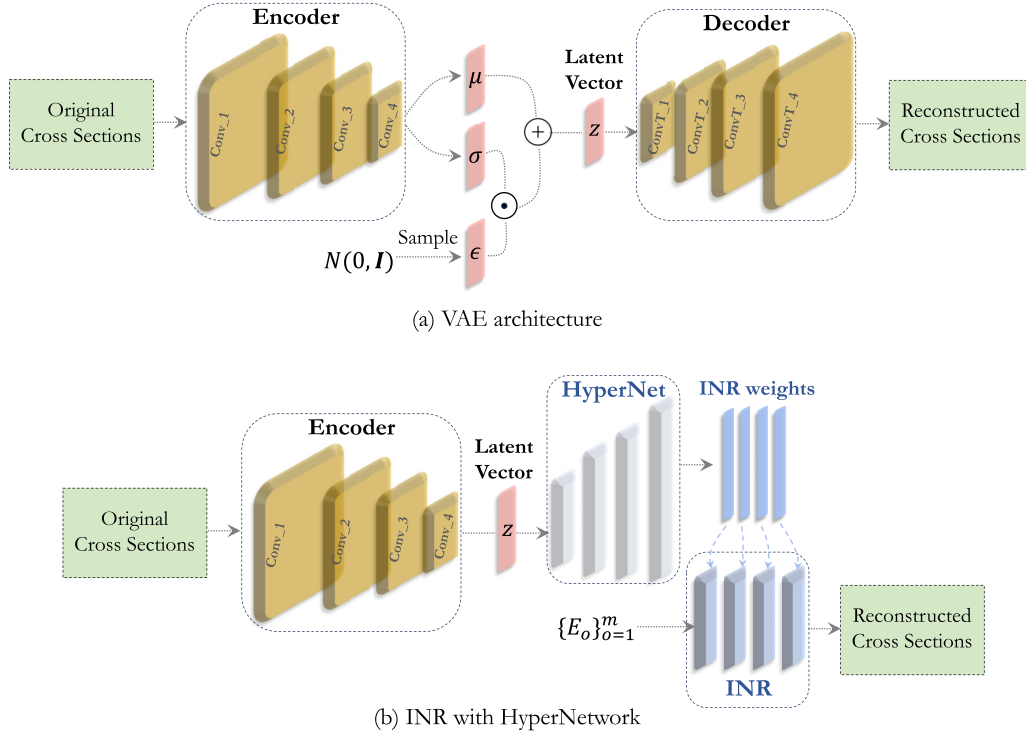


FIG. 2. (a) Variational autoencoder framework illustrating the encoding and decoding process with a latent space bottleneck. (b) Implicit neural representation (INR) architecture integrated with a hypernetwork (HyperNet). The hypernetwork generates weights for the INR, informed by a convolutional encoder that processes input data. Note that the INR takes input coordinates, $\{e_i\}$ (representing the energy index with a length of $m = 256$) and maps them to the corresponding cross sections.

concludes with a sigmoid activation for the reconstructed inputs. As the mirror opposite of the encoder, the decoder progressively refines the representation through four layers with specific filter sizes and strides: 128 filters with a kernel size of 5 and stride of 2, followed by 64 filters with a kernel size of 5 and stride of 2, then 32 filters with a kernel size of 6 and stride of 2, and finally, the output layer with a kernel size of 24 and a stride of 8. The output layer returns a vector of size m . By jointly optimizing the encoder and decoder, the VAE learns a compact representation of given inputs.

2. Implicit neural representation

INR is a method for representing and modeling input data without explicitly defining the geometry or structure of objects [32,35]. Traditional data encoding methods rely on discrete representations, such as data grids, which are constrained by their spatial resolution and are susceptible to inherent discretization artifacts. In contrast, INR represents input data (including sound, time series, images, and 3D scenes) as a continuous function of its coordinate locations, where each value is generated independently. This function is approximated by a deep neural network. INR offers flexibility and adaptability for encoding data independently of resolution through the use of coordinate grids, making them highly effective with numerous potential applications [35–37].

While INRs offer advantages in practical scenarios, they face the limitation of often struggling to generalize beyond the single input on which they were trained. This can hinder

their effectiveness in deep learning applications, where robust generalization across diverse training datasets is crucial. To address this issue, hypernetworks have been introduced [31]. Hypernetworks act as meta neural networks, generating weights for a primary neural network, the desired INR. Sitzmann *et al.* [32] successfully employed a convolutional encoder and hypernetwork combination to guide the INR in learning more generalizable representations of training data. This unified framework helps the INR overcome the “single-fit” limitation by learning priors over the input space, enhancing its representational capabilities while maintaining its accurate reconstruction abilities.

As shown in Fig. 2(b), our framework integrates a hypernetwork with an INR. The convolutional encoder leverages the same configuration as in Fig. 2(a), featuring a series of convolutional layers. It processes cross section data and compresses it into a 32-dimensional latent vector. This vector is then passed to the hypernetwork decoder, which consists of standard multilayer perceptron layers with ReLU activations. The hypernetwork’s output provides the weights for the INR. Finally, the INR takes coordinates as input and reconstructs the original input data. This pipeline enables networks to be trained in an end-to-end manner with the following loss function:

$$\mathcal{L}_{\text{INR}} = \mathcal{L}_{\text{recon}} + \alpha \mathcal{L}_{\text{latent}} + \beta \mathcal{L}_{\text{weights}}, \quad (2)$$

where $\mathcal{L}_{\text{recon}}$, $\mathcal{L}_{\text{latent}}$, and $\mathcal{L}_{\text{weights}}$ represent the reconstruction loss, regularization terms for latent vectors, and weights in the hypernetwork, respectively. The hyperparameters α and β are

utilized to balance the contributions of the different losses. Specifically, each loss term is computed as follows:

$$\mathcal{L}_{\text{recon}} = \frac{1}{N_{\text{batch}}} \sum_{i=1}^{N_{\text{batch}}} \|\mathbf{y}_i - \hat{\mathbf{x}}_i\|_2, \quad (3a)$$

$$\mathcal{L}_{\text{latent}} = \frac{1}{N_{\text{batch}}} \sum_{i=1}^{N_{\text{batch}}} \|\mathbf{z}_i\|_2^2, \quad (3b)$$

$$\mathcal{L}_{\text{weights}} = \frac{1}{N_{\text{batch}}} \frac{1}{N_{\mathcal{W}}} \sum_{i=1}^{N_{\text{batch}}} \sum_{\mathbf{w} \in \mathcal{W}} \|\mathbf{w}_i\|_2^2, \quad (3c)$$

where $\|\cdots\|_2$ is the usual Euclidean norm. Here, $\mathbf{y}_i$ represents the i th ground truth input and $\hat{\mathbf{x}}_i$ its corresponding reconstruction. N_{batch} represents the batch size, indicating the number of samples processed during each training iteration. $\mathbf{z}_i$ refers to the latent vector with k dimensions produced by the encoder for the i th sample in the batch. $N_{\mathcal{W}}$ represents the total number of parameters in the INR model, and $\mathbf{w}_i$ are the individual weights. Note that $\mathcal{L}_{\text{latent}}$ and $\mathcal{L}_{\text{weights}}$ are regularization terms that aim to decrease the overall magnitude of feature and weight representations, making them less prone to overfitting by penalizing large magnitude. Through empirical evaluation, we determined that setting $\alpha = 0.01$ and $\beta = 1.0$ results in optimal reconstruction performance.

C. Graph neural networks

GNNs are optimizable transforms that operate on all attributes of a graph (nodes, edges, global context, inherent relationships, etc.) while preserving symmetries and the underlying structure of the data. Hence, GNNs are well suited to expose and explore the relationships between nuclear cross sections across a two-dimensional grid: the nuclear chart. GNNs accept a graph as input (for example, the encoded 2D TENDL grid) with the information contained in the nodes (feature vectors) and perform transformations without changing the connectivity of the input. GNNs also produce a graph as the output.

In this study, we are interested in graph-based imputation, where the goal is to predict missing node features by leveraging the structural and feature information from neighboring nodes. Two primary learning paradigms are commonly considered, transductive and inductive settings. In the transductive setting, the model is trained and evaluated on the same graph, where the node and edge structure remain fixed. In the inductive setting a subset of the graph is used for training while testing is performed on a disjoint subset or entirely new graph. Such a paradigm generally requires much stronger generalization power of the network. In this initial work, we focus on the transductive setting, as our primary interest lies in improving predictions by fully exploiting structural correlations across neighboring nuclei within the given nuclear chart.

1. Graph connectivity and message passing

The graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is constructed by representing each nucleus as a node. The node features are the learned latent representation of the nuclear cross sections obtained from the encoding module (either VAE and INR), i.e., vectors of

dimension $k = 32$. The edges $\mathcal{E}$ define the structural relationship between nuclei based on their positions in the (Z, N) chart. Here, we employ two-skip edge connectivity, where each node skips its one-hop and two-hop neighbors and connects to nodes located three steps away along the horizontal and vertical directions. This design choice is motivated by our goal to impute a relatively large contiguous missing region (e.g., a 9×9 test window). Using only local connectivity would limit the model's ability to propagate information into the interior of the masked test region. The designed skip connectivity enables longer-range message passing and allows the model to capture broader trends in the nuclear chart. We identified a performance tradeoff between local dense connectivity and skip connectivity, particularly with respect to the size of the test window. For example, denser edge connections tend to yield better performance for smaller test windows, whereas a noticeable performance drop is observed when applied to larger test windows (e.g., 9×9), where longer-range dependencies become more critical.

To effectively model such neighbor-dependent interactions, our GNN architecture $\chi(\cdot)$ leverages the *message passing neural network* (MPNN) framework [38], where the hidden state of each node is iteratively updated by aggregating information from its neighbors. Specifically, for the regular grid graph considered in this paper, the forward pass through the GNN has two phases: a message passing phase and a readout phase. In the message passing phase for a particular node, the message function M_t aggregates the information from its neighbors and updates the node according to some update function U_t . At any time t , the hidden state h_i^t at node v_i is updated as

$$m_i^{t+1} = \sum_{j \in N(v_i)} M_t(h_i^t, h_j^t, e_{ij}), \quad (4)$$

$$h_i^{t+1} = U_t(h_i^t, m_i^{t+1}), \quad (5)$$

where $N(v_i)$ denotes the neighbors of node v_i in the graph $\mathcal{G}$, and e_{ij} represents optional edge attributes.

2. GNN training

In our GNN-based imputation framework, training is performed by randomly masking a subset of nodes at each iteration and replacing their features with random placeholders. The GNN is then tasked with predicting the original values in the masked regions using the features of neighboring nodes and the graph structure. Importantly, the loss is computed only over the masked nodes. The test nodes [held-out 9×9 region centered at $(Z_0, N_0) = (42, 60)$] are also replaced with fixed random placeholders, using a consistent random seed, and are excluded from training loss computations.

We adopt two different training strategies depending on the underlying representation model, VAE or INR, used to generate node features. For VAE-based embeddings, we found that training the GNN to minimize reconstruction loss (mean square error) in the original cross section space, i.e., between decoded predictions and ground truth, works well, particularly when the decoder is fine tuned jointly with the GNN. This end-to-end training allows the decoder to adapt to the GNN's latent predictions, resulting in more accurate predictions. By

ALGORITHM 1. Graph-Based Imputation on Embeddings.

Require: Node feature extractor E (VAE or INR encoder), decoder ϕ (VAE or INR), GNN χ , training set $\mathcal{T}$, test region $\mathcal{S}$ (e.g., 9×9 window), mask probability $p = 10\%$, number of epochs T

Ensure: Predicted node features for $\mathcal{S}$

- 1: Encode raw cross section data: $\mathbf{z}_i \leftarrow E(\mathbf{y}_i)$ for all $v_i \in \mathcal{T} \cup \mathcal{S}$
- 2: Build graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with chosen edge connectivity (e.g., two-skip edge connectivity)
- 3: Replace node features of all $v_j \in \mathcal{S}$ with fixed random noise: $\mathbf{z}_j \leftarrow \eta_j$ with fixed seed
- 4: **for** $t = 1$ to T **do**
- 5: Sample a random subset $\mathcal{M} \subset \mathcal{T}$ with $|\mathcal{M}| = p|\mathcal{T}|$
- 6: Replace node features of $\mathcal{M}$ with random noise η_i
- 7: $\hat{\mathbf{z}}_i \leftarrow \chi(\mathcal{G})_i$ for all $v_i \in \mathcal{V}$
- 8: **if** using VAE **then**
- 9: $\hat{\mathbf{y}}_i \leftarrow \phi(\hat{\mathbf{z}}_i)$ ▷ Decode latent
- 10: $\mathcal{L} \leftarrow \frac{1}{|\mathcal{M}|} \sum_{v_i \in \mathcal{M}} \|\hat{\mathbf{y}}_i - \mathbf{y}_i\|^2$ ▷ Eq. (6)
- 11: **else**
- 12: $\mathcal{L} \leftarrow \frac{1}{|\mathcal{M}|} \sum_{v_i \in \mathcal{M}} \|\hat{\mathbf{z}}_i - \mathbf{z}_i\|^2$ ▷ Eq. (7)
- 13: **end if**
- 14: Backpropagate and update parameters of χ (and ϕ if decoder is fine tuned)
- 15: **end for**
- 16: Use trained χ (and ϕ) to predict missing features for all $v_j \in \mathcal{S}$
- 17: Return $\hat{\mathbf{y}}_j \leftarrow \phi(\chi(\mathcal{G})_j)$

contrast, for INR-based embeddings, we observe that freezing the INR decoder and minimizing the loss directly in the latent space leads to better performance. This might be because the INR decoder with hypernetwork that generates the weights of an implicit neural function makes it difficult to fine-tune effectively. Formally, the two training objectives read

$$\mathcal{L}_{\text{GNN/VAE}} = \frac{1}{|\mathcal{M}|} \sum_{v_i \in \mathcal{M}} \|\phi(\chi(\mathcal{G})_i) - \mathbf{y}_i\|^2, \quad (6)$$

$$\mathcal{L}_{\text{GNN/INR}} = \frac{1}{|\mathcal{M}|} \sum_{v_i \in \mathcal{M}} \|\chi(\mathcal{G})_i - \mathbf{z}_i\|^2. \quad (7)$$

Here, $\mathcal{M}$ denotes the set of masked training nodes. $\chi(\mathcal{G})_i$ denotes the GNN-predicted latent vector for node v_i , and ϕ represents the decoder that maps the latent vectors to the original cross section domain. $\mathbf{y}_i$ is the ground-truth cross section, while $\mathbf{z}_i$ refers to the corresponding ground-truth latent representation. The overall imputation pipeline is summarized in Algorithm 1. Furthermore, we refer the reader to Appendix A3 for a detailed comparisons for training strategies under different loss spaces and decoder fine-tuning options.

3. GNN architectures

We evaluate three types of message-passing neural network architectures, ARMAConv [39], GatedConv [40], and GraphSAGE [41]. The ARMA convolutional layer simulates graph convolution via approximated rational filters using a stack of recurrent units. It supports deeper feature propagation and has shown stability in capturing long-range dependencies. With shared weights across layers, it balances model expressiveness and parameter efficiency. GatedConv integrates recurrent updates with gated mechanisms, allowing information to flow

adaptively over multiple time steps. The gates help control feature propagation, which can be useful for graphs where information needs to be selectively passed or retained. GraphSAGE generates node embeddings by aggregating features from a node's neighborhood. It supports scalable learning and provides robust performance across a variety of graph learning tasks. In our setup, it achieves the highest imputation performance with VAE embeddings while maintaining parameter efficiency. All these architectures considered use the MPNN backbone with slight variations either in terms of the type of filter used or the node update method, and each architecture is trained under the same data and masking setup.

IV. RESULTS

We recall that our proposed framework consists of two main stages. First, we learn low-dimensional latent representations of nuclear cross sections using dimensionality reduction techniques. Second, we train GNN to impute the missing node features (i.e., latent vectors) in the nuclear chart. Since the effectiveness of GNNs in this task depends on the quality of embedding space, we begin by analyzing the two types of encoders used in this study, the VAE model of Fig. 2(a) and the INR model of Fig. 2(b). We then evaluate the performance of several GNN architectures for the downstream imputation task, where the goal is to predict unknown cross sections in the test region using information learned from known regions in the training set.

A. Encoding performance

To train models for latent representations, we randomly split the dataset containing 2171 samples into a training set ($\approx 90\%$) and a testing set ($\approx 10\%$). To ensure reproducibility and fair comparisons, we employed a fixed random seed, guaranteeing both models were trained on identical datasets. Prior to model training, we normalized the input data by dividing each cross section by its corresponding maximum value across the entire dataset. This ensures that each cross section falls within the range of $[0, 1]$. Then, the VAE model was trained on a multi-channel cross section dataset for 5000 epochs with a batch size of 32. We utilized the Adam optimizer (initial learning rate 1×10^{-3}) for optimization. The INR model, trained on the same dataset, also used the Adam optimizer (learning rate 5×10^{-5}) but for 600 epochs. Notably, the INR model demonstrated significantly faster convergence compared to the VAE training while maintaining high quality of latent representations.

Figure 3 illustrates the performance of both the VAE (top) and INR (bottom) for a nucleus within the TENDL dataset. Each color represents one of the four cross section channels considered in this work. The curves with symbols (square, circle, $\times$, inverted triangle) represent the reconstructed inputs generated by either the VAE or the INR, whereas plain curves depict the ground truth inputs. As seen in this figure, both models effectively reconstruct given test samples, closely aligning with the ground truth. These results demonstrate that both models are able to encode the multichannel cross

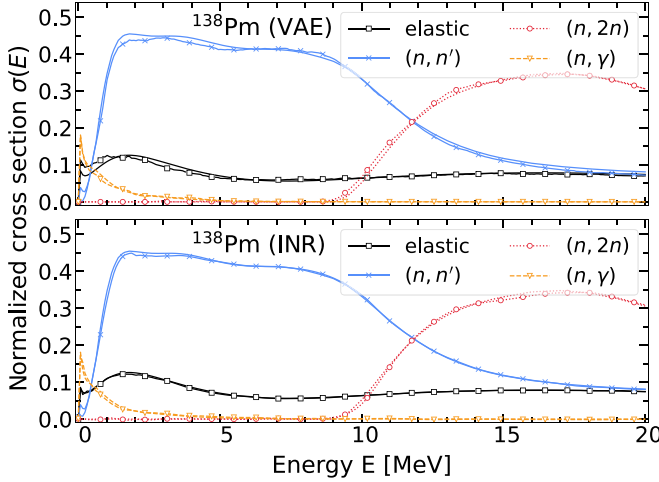


FIG. 3. Comparison of the encoding performance between VAE (top) and INR (bottom) for all four cross sections in the ^{138}Pm nucleus. Ground-truth results are shown by plain lines and full symbols while reconstructions are shown by dashed lines and open symbols. Performance is typical of all nuclei in the TENDL dataset.

section input into a compact, 32-dimensional latent vector with minimal loss in reconstruction accuracy.

To quantify reconstruction performance, we use the peak signal-to-noise ratio (PSNR). The PSNR is defined as follows

$$\text{PSNR} = 20 \log_{10} \left(\frac{\text{MAX}}{\sqrt{\text{MSE}}} \right), \quad (8)$$

where $\text{MSE} = \frac{1}{n} \sum_{i=1}^n \|\mathbf{y}_i - \hat{\mathbf{x}}_i\|_2^2$, $\hat{\mathbf{x}}_i$ represents the reconstructed input generated by either the VAE or the INR, while $\mathbf{y}_i$ denotes the original cross section inputs. Here, n is the total number of testing samples. MAX represents the maximum possible value of the signal, which is equal to 1. We evaluate each model using this metric on the entire test set. The VAE achieves a score of 46.68, while the INR achieves a higher score of 51.39. Overall, the performance of the INR is significantly better as this model is better able to precisely capture high-frequency information such as thresholds, for example the fact that the cross section for the $(n, 2n)$ cross section is identically 0 unless the energy of the incident neutron exceeds the neutron separation energy.

B. GNN performance

To evaluate the GNN performance, we divide the entire TENDL dataset into a training and testing set based on a windowing function. The windowed test set is created using a square window centered around a specific nucleus (Z_0, N_0) in the chart of isotopes and containing all nuclei with $Z_0 - \Delta Z \leq Z_0 \leq Z_0 + \Delta Z$ and $N_0 - \Delta N \leq N_0 \leq N_0 + \Delta N$. All results presented in this section were obtained using a test window centered at $(Z_0, N_0) = (42, 60)$ with $\Delta Z = \Delta N = 4$, resulting in a 9×9 test region on the nuclear chart. This window contains a total of 81 test nuclei, 67 of which are included in the original TENDL dataset. Since that original dataset contains 2171 points on the 2D plane, this means that about 3.0% of the total data is allocated to the test set. In

TABLE I. Test PSNR values of the different GNN architectures with VAE and INR embeddings. Bold entries indicate the highest PSNR score for each embedding.

GNN Arch.	ARMAConv	GatedConv	GraphSAGE
INR embedding	35.31	35.48	35.14
VAE embedding	35.87	36.40	36.65

Appendix A 2, we also present the GNN performance across varying window sizes and different center locations.

To train the GNNs, we use the AdamW optimizer [42] with a base learning rate of $r_l = 0.001$. To promote faster convergence and stabilize training dynamics, we adopt a one-cycle learning rate scheduler [43], which linearly increases the rate up to $10 \times r_l$ before annealing it back down. Specifically, we set the maximum learning rate to 0.01, with a total of 100 000 training epochs. This setup was applied consistently across both VAE-based and INR-based experiments. For each epoch, the entire training set is passed through the GNN and we do not create batches of the training graph since breaking the edge relationships between the nodes would yield inaccurate results. The prediction error of the decoded cross sections is computed as

$$\epsilon = \frac{1}{n} \sum_{i=1}^n \|\mathbf{y}_i - \hat{\mathbf{x}}_i\|_2^2, \quad (9)$$

where $\hat{\mathbf{x}}_i$ and $\mathbf{y}_i$ are the predicted node features and the ground truth in the original space, respectively, and n is the actual number of nuclei in the set (either training or test). The quantity $(\mathbf{y}_i - \hat{\mathbf{x}}_i)$ computes the pointwise difference, which is then summed for each position i in the test set. We calculate the prediction error in the 256-dimension space using Eq. (9), and present the corresponding error distribution as a heat map in Fig. 4.

In Fig. 4, we observe notable differences in the error distribution between VAE and INR embeddings across the nuclear chart. In the VAE case (left column), prediction errors are mostly concentrated in the test region, while the rest of the chart (training region) shows low errors. Conversely, INR-based models (right column) display more widespread errors, including in the training region, especially in the upper part of the chart. This aligns with INR's relatively compact latent space as well as lower diversity across samples, which lead to a smaller training-testing gap. We further examine and visualize the characteristics of these latent spaces in Sec. V A. To provide a more quantitative comparison, Table I reports the average test PSNR of Eq. (8) over all test samples for each GNN architecture using both VAE and INR embeddings.

We also visualize the prediction performance of the GraphSAGE network using both VAE encoding (left column) and INR encoding (right column) in Fig. 5. The figure shows the normalized cross sections for three nuclei located within in the test window: ^{95}Y near the lower-left corner, ^{102}Mo at the center (hence the furthest away from the training set), and ^{108}Ru near the upper-right corner of the window. With the VAE encoding, the predicted curves closely match the ground truth, with only a slight deviation in the $(n, 2n)$ channel. The

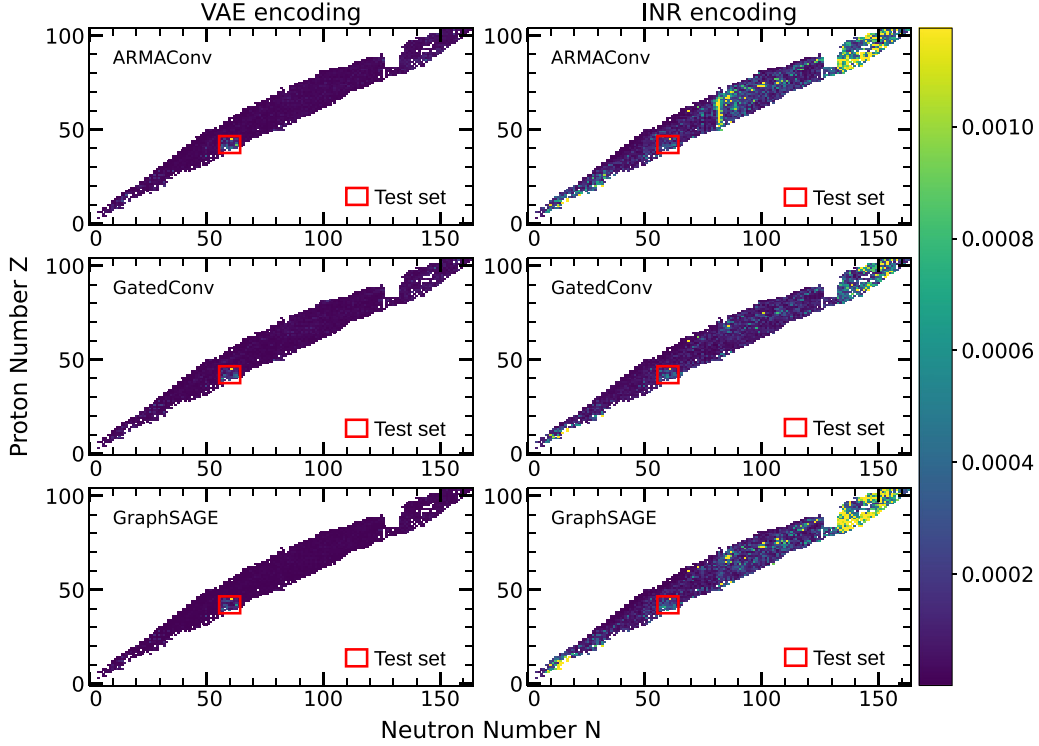


FIG. 4. Mean prediction errors calculated using Eq. (9) of the GNN for each nucleus in the TENDL dataset. For better visualization, all plots are normalized between $\epsilon_{\min}$ and $0.03 \times \epsilon_{\max}$, where $\epsilon_{\min}$ ($\epsilon_{\max}$) is the minimum (maximum) error over all plots. The three panels on the left side are based on VAE encoding while the three on the right on INR encoding. ARMAConv, GatedConv, and GraphSAGE refer to the three types of GNNs considered in this work. The red square corresponds to the windowed test data: the GNN was not trained on them.

predictions from the INR encoding exhibit similar trends and capture the overall structure of the cross sections, despite more noticeable differences in $(n, 2n)$ channel, especially at the center region. Numerical experiments show that the model's epistemic error stemming from the training of both VAE/INR and GNN is small and have no effect on these conclusions. Note that the (n, γ) channels is not identically zero but the normalized cross section is several order smaller than the other ones for these particular nuclei.

V. DISCUSSION

The results presented in Section IV pose a few interesting questions, from both data science and nuclear physics perspectives, which we discuss in this section. In particular, we focus on a brief analysis of the structure of the latent space to analyze whether specific patterns can be matched to nuclear properties.

A. Representation learning and structure of the latent space

When training GNNs with different latent representations, the choice of the loss function (6) or (7) significantly impacts performance. Our experiments reveal a clear contrast: GNNs trained on VAE-based embeddings achieve better performance when optimizing in the original cross section space, whereas GNNs trained on INR-based embeddings perform better when optimized directly in the latent space. This discrepancy can be attributed to two key factors:

Decoder sensitivity in INR models. The INR decoder is parameterized by a hypernetwork which maps latent vectors not directly to outputs but to the weights of an implicit neural function. This architectural design introduces an additional sensitivity when it is used as downstream prediction task. Small perturbations in the predicted weights of the neural representation can drastically affect the predicted cross sections. This makes finetuning the decoder in an end-to-end fashion less effective compared to the one with VAE-based embedding.

Embedding space expressivity. As shown in Fig. 6, VAE latent spaces tend to have higher variance and broader spread across samples, which provide the downstream GNN with richer signals to learn from, whereas, INR latent spaces are more compact and concentrated in narrow regions of the latent space. This could limit the discriminative power of the GNN. Consequently, optimizing directly in the latent space for INR-based embedding circumvents the difficulty of optimization through the sensitive decoder.

In Fig. 6, we show a direct visualization of the latent spaces for the training dataset of both the VAE and INR models. In both panels, we perform a principal component analysis (PCA) of the k -dimensional latent vectors ($k = 32$) and project each of them onto the two-dimensional plane spanned by the two largest PCA components. Panel (a) corresponds to the VAE latent space while panel (b) corresponds

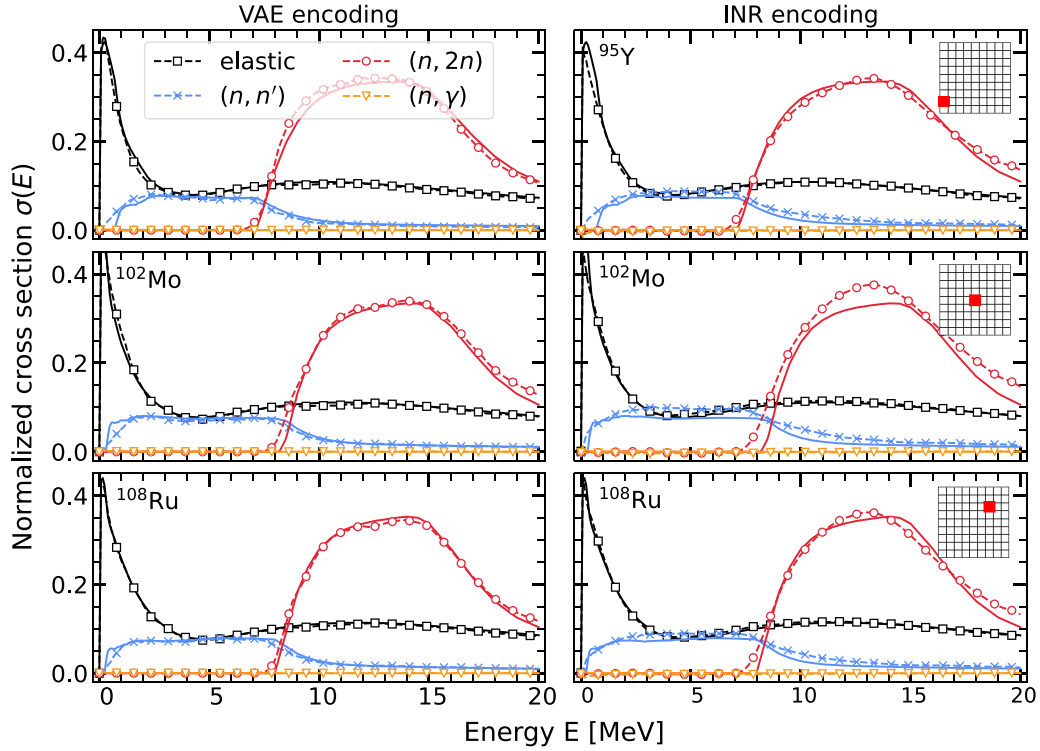


FIG. 5. Performance of the VAE (left) and INR (right) model for the GraphSAGE GNN architecture in predicting nuclear cross sections within the test set for the central nucleus, ^{102}Mo , and the ones near the lower left and upper right of the window, ^{95}Y and ^{108}Ru respectively. Plain curves without symbols correspond to the ground truth while dashed curves with open symbols correspond to GNN predictions. All cross sections are normalized independently in each channel across the entire dataset. The small grid in the upper right corner of the INR figures shows the location of each nucleus within the test set.

to the INR latent space. We recall that each point in such a figure corresponds to a given nucleus; its color is the value of the difference $N - Z$ of the neutron number N and proton number Z in the corresponding nucleus. The only noticeable difference is that the samples' variability scale for the latent space in the INR model is significantly smaller than for its VAE counterpart: the INR hypernet latent space is much more “compact” than the VAE one.

To better understand the structural differences between the two encoding techniques, we compute the t-distributed stochastic neighbor (t-SNE) embedding [44]. Let us quickly recall the basics of the t-SNE method. Assume we have a set $\mathcal{P}$ of points P in a n_p -dimensional space, and characterize the “similarity” of two points P and P' in $\mathcal{P}$ by some joint probability distribution function p . Let us now assume that $\mathcal{P}$ can be mapped onto a set $\mathcal{Q}$ of points Q in a n_q -dimensional space ($n_q \ll n_p$) that is characterized by another probability distribution q . The t-SNE technique minimizes the Kullback-Leibler divergence $\mathcal{L}_{PQ}$ between the joint probabilities p in the high-dimensional space and the joint probabilities q in the low-dimensional space. In simple terms: two points that are “neighbors” in the high-dimensional space should also be neighbors in the low-dimensional one.

Figure 7 shows the two-dimensional representation of this t-SNE embedding for both the VAE and INR encodings as obtained with the scikit-learn implementation. Like Fig. 6, the color of a point is the difference $N - Z$ of the neutron number N and proton number Z in the corresponding nucleus. In

practice, t-SNE algorithms require setting a few hyperparameters, most notably one called “perplexity” or Perp , which can be interpreted as “a smooth measure of the effective number of neighbors” in the high-dimensional space [44]. Although traditional applications of t-SNE suggest values of the perplexity between 5 and 50, this is highly dependent on the dataset and its dimensionality. In our case, the dataset has 2090 points—training nuclei represent 97% of the whole TENDL dataset of 2171 nuclei—and is encoded in a latent space of dimension $k = 32$. We found that in order to converge the Kullback-Leibler loss $\mathcal{L}_{PQ}$, the perplexity parameter had to be significantly larger: Fig. 7 was obtained for $\text{Perp} = 200$. Note that the spatial arrangement of points and the variability scale in each latent dimension are very dependent on the value of Perp . In particular, small values of Perp (corresponding to large values of $\mathcal{L}_{PQ}$) lead to excessive visible clustering while failing to converge the projection.

To better analyze clustering structures, we used the scikit-learn implementation of the spectral clustering algorithm to identify potential clusters of points directly in the VAE or INR dataset. Out of all available clustering techniques, spectral clustering and the DBScan (density-based spatial clustering of applications with noise) algorithms are the naive best choices to identify clusters based on existing documentation and literature. However, calibrating the maximum distance between any two points to be considered in the neighborhood of each other in DBScan requires a deeper understanding of the latent space than we currently have. In contrast, spectral clustering

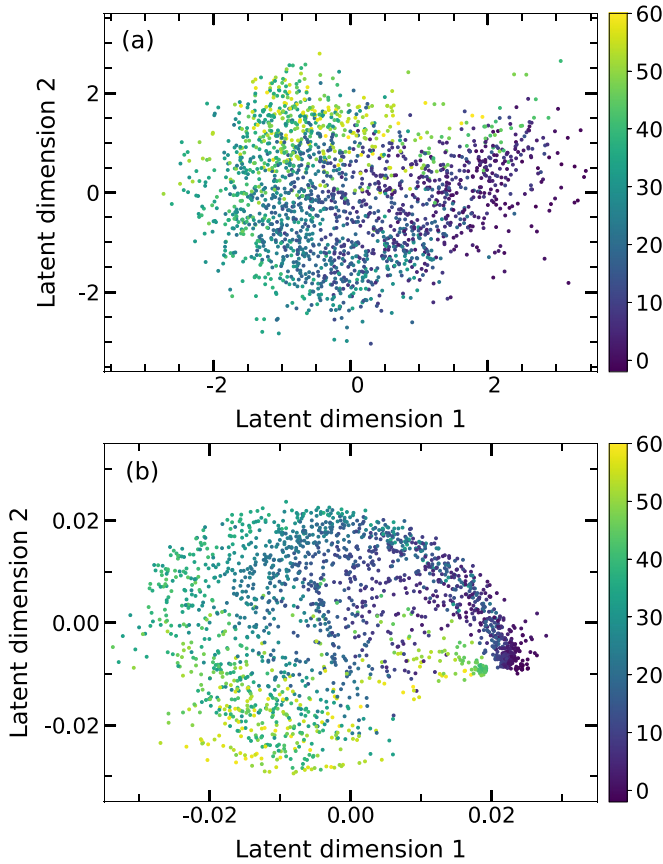


FIG. 6. PCA visualization of the VAE and INR hypernet latent spaces. (a) Projections of each k -dimensional VAE latent vector onto a two-dimensional plane spanned by the two largest components of the PCA. (b) Same figure for the INR latent vector.

does not involve any prior knowledge of the latent space. We first determined a rough estimate of the optimal number n_{clus} of clusters based on the silhouette method. In practice, we found $n_{\text{clus}} = 4$. Figure 8 shows the position in the nuclear chart of each nucleus from the INR encoding training dataset, color coded by the cluster the nucleus belongs to. The most striking feature of this plot is the fact that nuclei are organized in diagonal bands loosely following the valley of stability rather than along isotopic, isotonic, or isobaric sequences. In addition, some of the clusters seem to be related to neutron magic numbers (but, interestingly, not proton magic numbers), especially $N = 82$ and $N = 126$. Both observations are quite robust. In Appendix A 4, we show that different clustering scenarios lead to similar qualitative results.

B. Predictions of unknown cross sections

Predicting nuclear cross sections across the entire chart in a consistent manner and with well-defined uncertainties is particularly challenging due to the lack of experimental data on radioactive isotopes and the limited predictive power of existing models. In Sec. IV we have shown that both VAE and INR networks can successfully compress four-channel cross sections with full energy index into compact, low-dimensional representations. This approach exhibits excellent

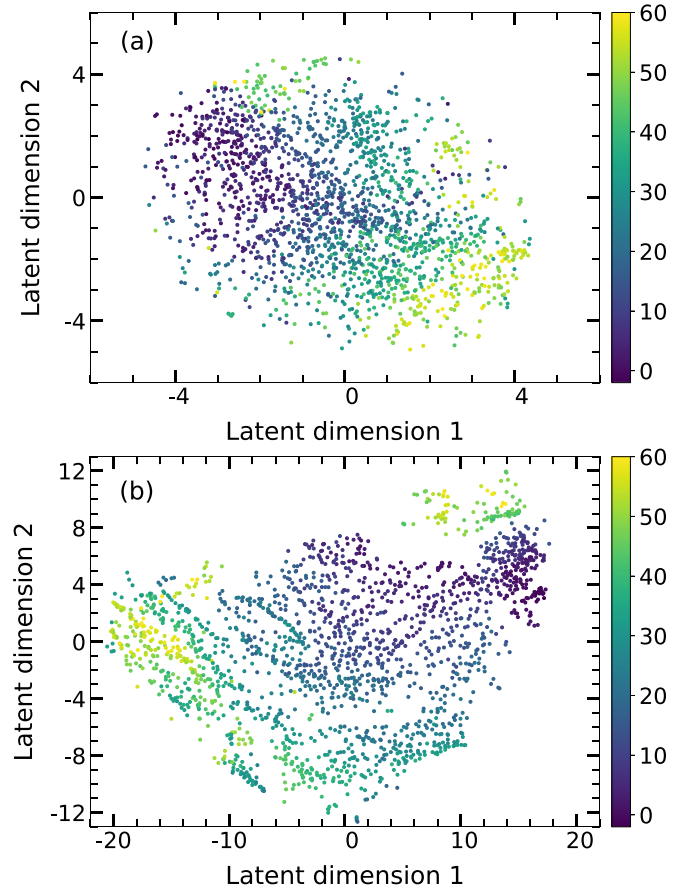


FIG. 7. t-SNE visualization of the VAE and INR hypernet latent spaces. (a) Two-dimensional t-SNE embedding of k -dimensional VAE latent vectors obtained for a perplexity value of 200. (b) Same figure for the INR latent vector.

encoding performance. This nontrivial result suggests that the “effective” space of nuclear cross sections is low-dimensional indeed.

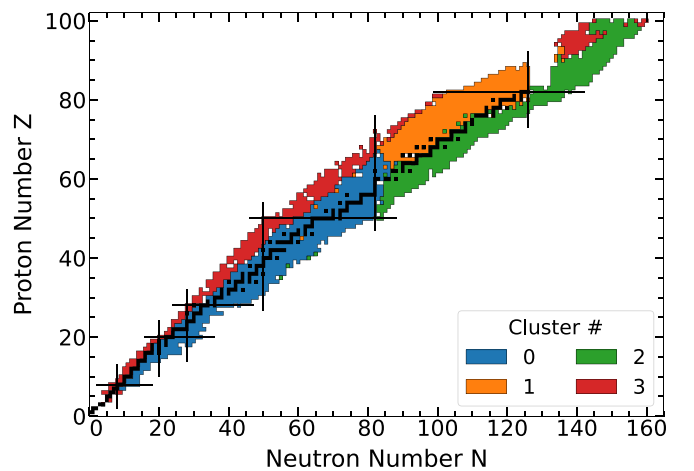


FIG. 8. Location of each nucleus in the training set with color code based on which of four clusters the nucleus belong too. Black horizontal and vertical lines represent magic numbers. Black squares are stable nuclei.

The predictive performance of GNNs in our imputation task is influenced by several interconnected factors. One of the most critical elements is the test window size, which directly impacts the number of available training nodes in the transductive setting where the model is trained and evaluated on a fixed global graph. As the size of the test window increases, fewer nodes remain for training, thereby limiting the opportunity for the model to learn from surrounding neighborhoods. This degradation in performance with larger test windows is confirmed in Fig. 11. Furthermore, nuclei occupy a diagonal band in the (Z, N) chart rather than a uniform 2D grid. This results in regions of sparsity and disconnection that challenge the GNN's ability to propagate information effectively. As a result, the location of the test window also significantly affects performance, for example, regions with denser nuclear neighborhood exploit more effective message passing, whereas sparse areas hinder it, as verified in Fig. 10. In terms of training strategy, our results show that GNN performance depends not only on the structure of the latent space learned through VAEs or INRs but also on how the GNN is optimized. Specifically, the GNN with VAE embeddings benefit from end-to-end training in the original cross section space, while INR embeddings perform better when optimization is restricted to the latent space as reported in Table II. Our analysis from multiple perspectives, including test PSNR, error heatmaps, and the structure of the learned latent space, highlights the critical importance of the embedding in determining the effectiveness of downstream GNN predictions. These findings suggest that developing latent representations that are well aligned with the needs of graph-based prediction tasks is a promising direction for future research.

The obvious limitation of this initial study is that the training of all networks is based on the TENDL dataset, which is itself a tabulation of predictions from a finite collection of nuclear models. However, this could be turned into an advantage: since the deep learning approach provides a single unifying representation of all cross sections in that set, cross sections that the GNN fails to reproduce and/or predict well may be used as signals for outliers in TENDL. Furthermore, our methodology can easily be extended by augmenting TENDL with additional datasets, either from evaluated libraries such as ENDF, JEFF, or JENDL, or from pure experimental data such as EXFOR.

Most importantly, the learned relationships between cross sections of different types within a single nucleus, or between different nuclei can pave the way to much more ambitious studies. In the short term, short-distance extrapolation of cross sections of the type $\sigma_\alpha(N, Z) \rightarrow \sigma_\alpha(N + i, Z + j)$ for channel α should be relatively reliable, especially if i and j are small, i.e., $i, j \leq 4$. Such extrapolations would be especially useful in the context of nucleosynthesis simulations where precision matters less than accuracy. In the longer term, it should be possible to exploit these relationships to estimate covariances. Current libraries are severely limited in covariance data and in many cases, these data are unreliable. Furthermore, cross-material covariances, that is, how uncertainties in a cross section of one nucleus are related to another type of cross section in another nucleus, have never been estimated. By providing a unified framework to compute cross sections,

our methodology offers a viable path forward to estimate all such covariances and properly propagate uncertainties from nuclear data into applications. In the next section, we show how the current trained networks can already give us valuable information how uncertainties propagate across the nuclear chart.

VI. CONCLUSION

In this work, we used variational autoencoders (VAEs) and implicit neural representations (INRs) to learn the latent space of nuclear cross sections, specifically the elastic, inelastic, $(n, 2n)$ and (n, γ) cross sections in the fast regime. Both types of architecture were trained on the synthetic data from the TENDL nuclear dataset. We then used graph neural networks (GNNs) on the latent representations to predict the missing cross sections across the nuclear chart by leveraging the topological structure. We showed that we could predict cross sections of a rather decent-sized test set with good accuracy using a physics-free methodology: the networks learn from the data, whether the data come from measurements, calculations, or evaluations. The fact that this proof-of-concept study used the TENDL dataset was motivated exclusively by the need to have enough data to train the networks.

Our initial results suggest that modern deep learning methods are capable of accurately predicting, in a single, unifying framework, different types of nuclear cross sections. This opens the way to a more rigorous quantification of uncertainties, especially covariances and cross-material covariances. Since experimental data on nuclear cross sections are rather scarce (at the scale of the chart of isotopes), it should be possible to augment datasets with other sources of data coming, e.g., from large-scale simulations of ground-state properties, β -decay rates and γ -strength functions, or level densities. Microscopic theories such as, e.g., nuclear density functional theory, should have enough predictive power to predict accurately the trends of these various observables at the scale of the mass table, which could in turn help constrain deep learning models during the training phase.

ACKNOWLEDGMENTS

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract No. DE-AC52-07NA27344 with Lawrence Livermore National Security, LLC. This work was supported by the Laboratory Directed Research and Development (LDRD) program under project tracking code 23-SI-004. Computing support came from the Lawrence Livermore National Laboratory (LLNL) Institutional Computing Grand Challenge program.

APPENDIX

1. Train-test performance trends for VAE

To ensure the model does not overfit during the representation learning phase, we analyze the evolution of the loss and PSNR for both the training and testing set during the VAE training. These metrics are presented as a function

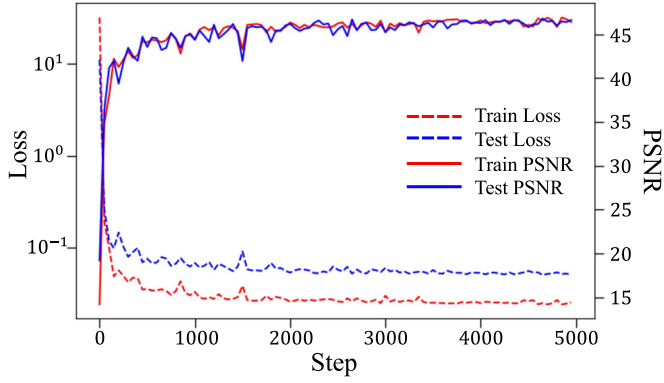


FIG. 9. Loss (dashed curves) and PSNR (plain curves) for the training set (red) and testing set (blue) as a function of epoch during VAE training. Note that the y axis for the loss is on a logarithmic scale.

of epochs in Fig. 9. Both training and testing loss curves steadily decrease and converge to similar values by the end of training. Also, the PSNR values for both training and testing increase consistently, improving reconstruction quality over time. The consistent convergence of both metrics indicates minimal overfitting and robust generalization of the learned latent representations.

2. Generalization analysis in GNN

Choice of test regions. To investigate the model performance in imputation across different regions of the nuclear chart, we evaluate the test PSNR scores at eight distinct locations (Z_0, N_0) for the center of the test region. These locations vary in their local neighborhood density. For example, test regions centered on medium-mass nuclei such as $(Z_0, N_0) = (45, 55)$, $(Z_0, N_0) = (42, 60)$, and $(Z_0, N_0) = (57, 71)$ are densely surrounded by known neighbors, while test regions centered on heavy nuclei such as $(Z_0, N_0) = (81, 133)$, $(Z_0, N_0) = (85, 134)$, or $(Z_0, N_0) = (91, 148)$ are closer to the boundaries of the domain and are surrounded by fewer neighbors; see Fig. 8 for example. As shown in Fig. 10, the imputation performance is strongly correlated with local neighborhood density. Test regions with dense neighboring support yield higher PSNR scores, indicating effective message passage and generalization. In contrast, edge regions with limited topological context yield noticeably lower PSNR, reflecting the increased difficulty of inference. This observation highlights the importance of neighborhood context in graph-based models and the advantages of transductive message passing when applied within densely populated nuclear regions.

Choice of test window lengths. To further investigate how the imputation performance is influenced by the spatial extent of the test region, we analyze the GNN model's performance across various test window sizes denoted by Δ . In this setup, a test window of size $\Delta \times \Delta$ is held out and masked during training, while all other surrounding data points are used for learning. All windows are centered on our default $(Z_0, N_0) = (40, 62)$ nucleus. Since our graph is built based on spatial nuclear chart connectivity, increasing Δ reduces the number of

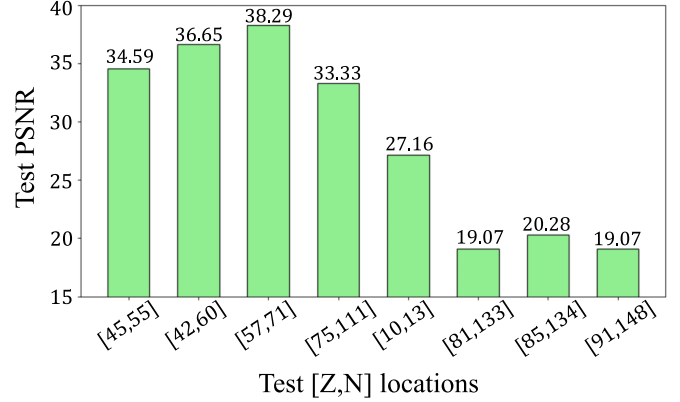


FIG. 10. Test PSNR scores for GNN imputation in test regions centered at different (Z_0, N_0) across the nuclear chart. The performance trend reflects the influence of local structural support on message-passing effectiveness and prediction quality. All test regions have the same size characterized by $\Delta Z = \Delta N = 4$ (i.e., 9×9 test window); only the center nucleus changes.

available training nodes, making the imputation task increasingly challenging. As shown in Fig. 11, both GraphSAGE and GatedConv exhibit a clear degradation in test PSNR as the window length increases from $\Delta = 2$ to $\Delta = 8$. This trend aligns with the intuition that accurate imputation demands more effective long-range message passing for larger masked

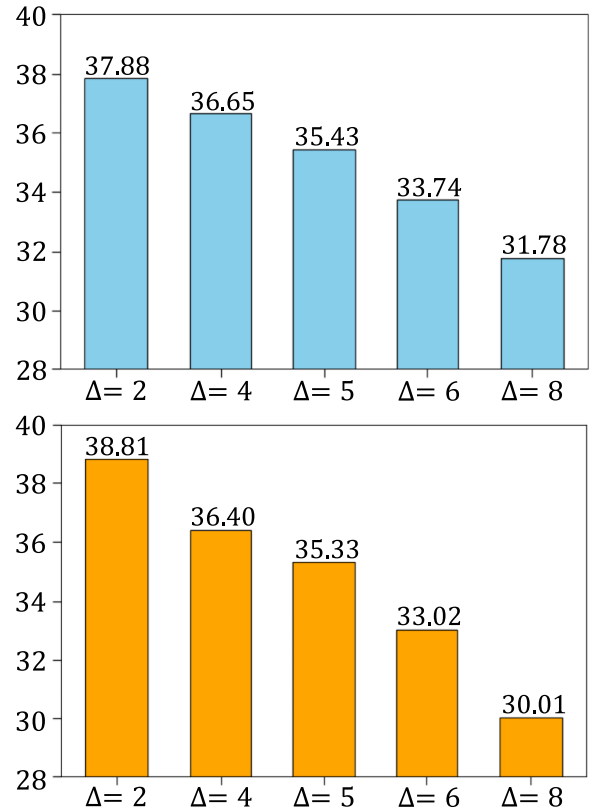


FIG. 11. Test PSNR values across varying window sizes for GraphSAGE (top) and GatedConv (bottom) approaches, with the center of the test window fixed at $(Z_0, N_0) = (42, 60)$. The two-skip edge connectivity is used for all cases.

TABLE II. Comparison of test PSNR scores under different training strategies with GatedConv. $\mathcal{L}_{\text{GNN}/256}$: MSE in original cross section space; $\mathcal{L}_{\text{GNN}/32}$: MSE in latent space. Note that for $\mathcal{L}_{\text{GNN}/32}$ (Decoder ON), the decoder does not affect the optimization process as the loss is computed before decoding. Bold entries indicate the highest PSNR score among latent space size.

Loss type	Decoder OFF		Decoder ON	
	INR	VAE	INR	VAE
$\mathcal{L}_{\text{GNN}/256}$	25.85	35.78	34.52	36.40
$\mathcal{L}_{\text{GNN}/32}$	35.48	33.21		

regions. This result highlights the importance of carefully balancing test window size and graph connectivity design. This is especially relevant in nuclear physics applications where the finite extent of the nuclear chart combined with the limited availability of actual data to train poses practical constraints.

3. Loss space and decoder fine-tuning effects

In this section, building on the training discussion in Sec. III C 2, we present an ablation study comparing different optimization strategies for VAE-based embeddings and INR-based embeddings. Specifically, we analyze the effect of minimizing the loss in either the latent space (dimension $k = 32$) or the original space (dimension $m = 256$), with and without decoder fine tuning. Let $\mathcal{L}_{\text{GNN}/32}$ denote the MSE loss computed between predicted latent vectors and corresponding ground-truth latent vectors, and let $\mathcal{L}_{\text{GNN}/256}$ denote the MSE loss computed between predicted and ground-truth cross sections in the original input space. The result in Table II highlights the distinct behaviors between the VAE and INR embeddings under different training regimes. When minimizing the loss in the original input space, the VAE-based embedding achieves the highest PSNR score when the decoder is fine tuned jointly with the GNN. This suggests that the VAE's

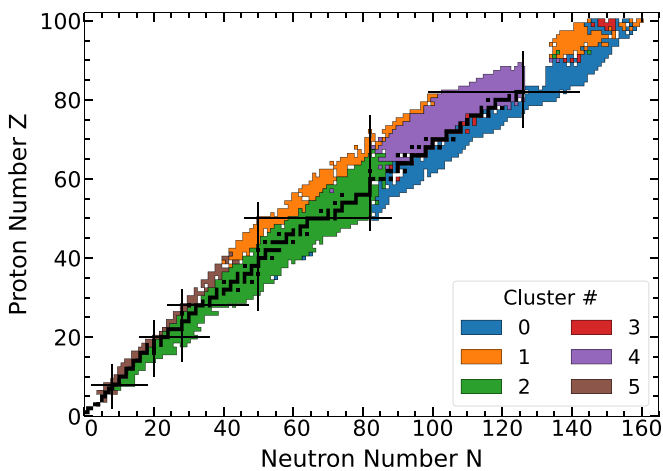


FIG. 12. Location of nuclei in the training set with color code based on which cluster the nucleus belongs too. Black horizontal and vertical lines represent magic numbers. The figure corresponds to the INR latent vectors and $n_{\text{clus}} = 6$.

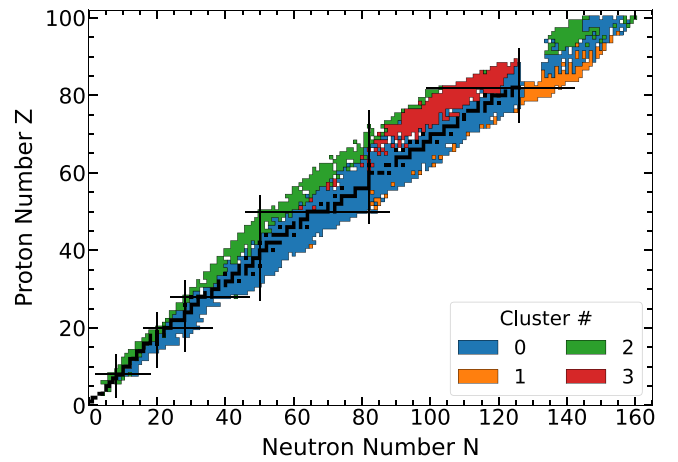


FIG. 13. Same as Fig. 12. The figure corresponds to the VAE latent vectors and $n_{\text{clus}} = 4$.

decoder is flexible enough to adapt to latent predictions during training, allowing for better end-to-end optimization. Interestingly, INR embeddings perform best when trained with loss computed directly in the latent space without involving the decoder in the optimization loop. This setup bypasses the complications associated with fine tuning the INR hypernetwork. These observations justify the different training strategies outlined in Algorithm 1 for each type of embedding.

4. Location of latent clusters

In this section, we show a few more examples of the location of clusters of nuclei identified directly from the VAE or INR latent vectors. Figure 12 quantifies the impact of changing the numbers of clusters to $n_{\text{clus}} = 6$. The main change is the splitting of the cluster of proton-rich nuclei into two subsets and the appearance of a small cluster of very heavy nuclei.

Figure 13 is the exact same figure as Fig. 8 in the main text, only it was constructed from the spectral clustering of the

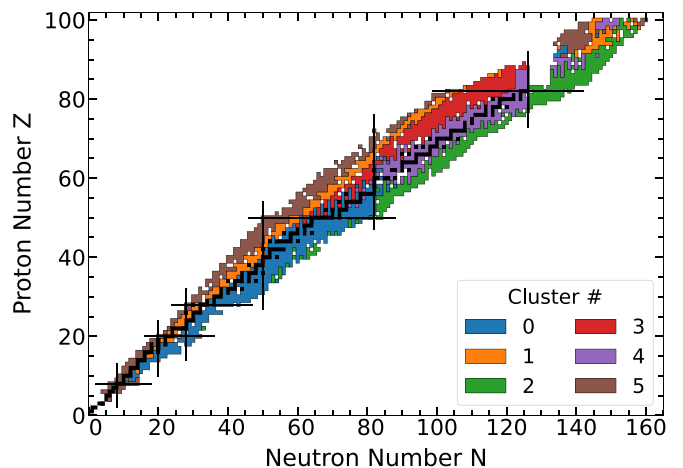


FIG. 14. Same as Fig. 12. The figure corresponds to the INR latent vectors and is obtained by the k -means algorithm instead of spectral clustering and with $n_{\text{clus}} = 6$.

VAE latent space (with $n_{\text{clus}} = 4$). We notice that the overall “diagonal” grouping of nuclei remains, although clusters are a little less compact. In addition, the main cluster that surrounds the valley of stability is not split in two at $N = 82$ as in Fig. 8 but instead at $N = 126$. It is still interesting that neutron magic numbers seem to be identified as markers of clustering in both methods.

Finally, Fig. 14 illustrates another clustering method based on the k -means algorithm. It is constructed from the clustering of the INR latent vectors and corresponds to $n_{\text{clus}} = 6$. Compared to Fig. 12, it illustrates the impact of a different clustering method. The k -means method is a little more naive

than spectral clustering and makes additional hypotheses about the clusters, in particular that they should be spherical. The main difference between the two methods is that k -means produces “thinner” clusters. However, the diagonal nature of the clusters and the role of magic neutron numbers in identifying clusters remains.

This appendix contains supplementary analyses and ablation studies to support the findings presented in the main paper. We first begin by investigating the representation learning phase, followed by additional analyses of the GNN-based prediction task, including variations in the test region location, test window size, and different training strategies.

-
- [1] J. Blatt and V. Weisskopf, *Theoretical Nuclear Physics* (Springer-Verlag, New York, 1979).
 - [2] O. Iwamoto, N. Iwamoto, S. Kunieda, F. Minato, S. Nakayama, Y. Abe, K. Tsubakihara, S. Okumura, C. Ishizuka, T. Yoshida, S. Chiba, N. Otuka, J.-C. Sublet, H. Iwamoto, K. Yamamoto, Y. Nagaya, K. Tada, C. Konno, N. Matsuda, K. Yokoyama *et al.*, Japanese evaluated nuclear data library version 5: JENDL-5, *J. Nucl. Sci. Technol.* **60**, 1 (2023).
 - [3] A. J. M. Plompen, O. Cabellos, C. De Saint Jean, M. Fleming, A. Algora, M. Angelone, P. Archier, E. Bauge, O. Bersillon, A. Blokhin, F. Cantargi, A. Chebboubi, C. Diez, H. Duarte, E. Dupont, J. Dyrda, B. Erasmus, L. Fiorito, U. Fischer *et al.*, The joint evaluated fission and fusion nuclear data library, JEFF-3.3, *Eur. Phys. J. A* **56**, 181 (2020).
 - [4] D. A. Brown, M. B. Chadwick, R. Capote, A. C. Kahler, A. Trkov, M. W. Herman, A. A. Sonzogni, Y. Danon, A. D. Carlson, M. Dunn, D. L. Smith, G. M. Hale, G. Arbanas, R. Arcilla, C. R. Bates, B. Beck, B. Becker, F. Brown, R. J. Casperson, J. Conlin *et al.*, ENDF/B-VIII.0: The 8th major release of the nuclear reaction data library with CIELO-project cross sections, new standards and thermal scattering data, *Nucl. Data Sheets* **148**, 1 (2018).
 - [5] L. Bernstein, F. Bostelmann, M. Carpenter, M. Chadwick, M. Fratoni, A. Hawari, L. Heilbronn, C. Howell, J. Ressler, C. Keppel, A. Koning, K. LaBel, T. Turflinger, C. Nesaraja, S. Qaim, C. Romano, A. Spyrou, S. Siem, C. Vermeulen, and R. Vogt, *First Report of the Nuclear Data Subcommittee of the Nuclear Science Advisory Committee*, Technical Report No. LLNL-TR-840465, 1890079, 1061532 (Lawrence Livermore National Laboratory, Livermore, CA, 2022).
 - [6] F. Bostelmann, G. Ilas, C. Celik, A. Holcomb, and W. Wiesequist, *Nuclear Data Assessment for Advanced Reactors*, Technical Report No. UREG/CR-7289, ORNL/TM-2021/2002 (United States Nuclear Regulatory Commission, Washington, DC, 2022).
 - [7] M. U. Khandaker, D. A. Bradley, H. Osman, M. I. Sayyed, A. Sulieman, M. R. I. Faruque, K. A. Naseer, and A. M. Idris, The significance of nuclear data in the production of radionuclides for theranostic/therapeutic applications, *Radiat. Phys. Chem.* **200**, 110342 (2022), special issue, The 15th International Symposium on Radiation Physics (ISRP15), 6–10 December 2021, Sunway University, Malaysia.
 - [8] A. C. Hayes, Applications of nuclear physics, *Rep. Prog. Phys.* **80**, 026301 (2017).
 - [9] I. U. Roederer, N. Vassh, E. M. Holmbeck, M. R. Mumpower, R. Surman, J. J. Cowan, T. C. Beers, R. Ezzeddine, A. Frebel, T. T. Hansen, V. M. Placco, and C. M. Sakari, Element abundance patterns in stars indicate fission of nuclei heavier than uranium, *Science* **382**, 1177 (2023).
 - [10] M. R. Mumpower, R. Surman, G. C. McLaughlin, and A. Aprahamian, The impact of individual nuclear properties on r -process nucleosynthesis, *Prog. Part. Nucl. Phys.* **86**, 86 (2016).
 - [11] G. Dodge, *A New Era of Discovery: The 2023 Long-Range Plan for Nuclear Science*, Technical Report (U.S. Department of Energy and National Science Foundation, Washington, DC, 2023).
 - [12] I. J. Thompson and F. M. Nunes, *Nuclear Reactions for Astrophysics: Principles, Calculation and Applications of Low-Energy Reactions* (Cambridge University Press, Cambridge, 2009).
 - [13] P. Navrátil and S. Quaglioni, *Ab initio* nuclear reaction theory with applications to astrophysics, in *Handbook of Nuclear Physics*, edited by I. Tanihata, H. Toki, and T. Kajino (Springer Nature, Singapore, 2020), p. 1.
 - [14] P. E. Hodgson, Compound nucleus reactions, *Rep. Prog. Phys.* **50**, 1171 (1987).
 - [15] B. V. Carlson, J. E. Escher, and M. S. Hussein, Theoretical descriptions of compound-nuclear reactions: Open problems and challenges, *J. Phys. G: Nucl. Part. Phys.* **41**, 094003 (2014).
 - [16] R. Capote, M. Herman, P. Obložinský, P. G. Young, S. Goriely, T. Belgia, A. V. Ignatyuk, A. J. Koning, S. Hilaire, V. A. Plujko, M. Avrigeanu, O. Bersillon, M. B. Chadwick, T. Fukahori, Z. Ge, Y. Han, S. Kailas, J. Kopecky, V. M. Maslov, G. Reffo, M. Sin, E. Sh. Soukhovitskii, and P. Talou, RIPL—reference input parameter library for calculation of nuclear reactions and nuclear data evaluations, *Nucl. Data Sheets* **110**, 3107 (2009).
 - [17] S. Goriely, P. Dimitriou, M. Wiedeking, T. Belgia, R. Firestone, J. Kopecky, M. Krtička, V. Plujko, R. Schwengner, S. Siem, H. Utsunomiya, S. Hilaire, S. Péru, Y. S. Cho, D. M. Filipescu, N. Iwamoto, T. Kawano, V. Varlamov, and R. Xu, Reference database for photon strength functions, *Eur. Phys. J. A* **55**, 172 (2019).
 - [18] K. Kolos, V. Sobes, R. Vogt, C. E. Romano, M. S. Smith, L. A. Bernstein, D. A. Brown, M. T. Burkey, Y. Danon, M. A. Elsayi, B. L. Goldblum, L. H. Heilbronn, S. L. Hogle, J. Hutchinson, B. Loer, E. A. McCutchan, M. R. Mumpower, E. M. O’Brien, C. Percher, P. N. Peplowski *et al.*, Current nuclear data needs for applications, *Phys. Rev. Res.* **4**, 021001 (2022).

- [19] M. D. Schwartz, Modern machine learning and particle physics, *Harvard Data Sci. Rev.* **3**, 13 (2021).
- [20] M. Feickert and B. Nachman, A living review of machine learning for particle physics, [arXiv:2102.02770](#).
- [21] J. M. R. Fox and K. A. Wendt, Illuminating systematic trends in nuclear data with generative machine learning models, [arXiv:2403.16389](#).
- [22] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, Geometric deep learning: going beyond Euclidean data, *IEEE Signal Process. Mag.* **34**, 18 (2017).
- [23] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, The graph neural network model, *IEEE Trans. Neural Netw.* **20**, 61 (2008).
- [24] X. Ju, S. Farrell, P. Calafiura, D. Murnane, L. Gray, T. Klijsma, K. Pedro, G. Cerati, J. Kowalkowski, G. Perdue *et al.*, Graph neural networks for particle reconstruction in high energy physics detectors, [arXiv:2003.11603](#).
- [25] C. Doersch, Tutorial on variational autoencoders, [arXiv:1606.05908](#).
- [26] A. Koning, S. Hilaire, and S. Goriely, Talys: modeling of nuclear reactions, *Eur. Phys. J. A* **59**, 131 (2023).
- [27] M. Herman, R. Capote, B. V. Carlson, P. Obložinský, M. Sin, A. Trkov, H. Wienke, and V. Zerkin, EMPIRE: Nuclear reaction model code system for data evaluation, *Nucl. Data Sheets* **108**, 2655 (2007), Special Issue on Evaluations of Neutron Cross Sections.
- [28] M. Bender, P.-H. Heenen, and P.-G. Reinhard, Self-consistent mean-field models for nuclear structure, *Rev. Mod. Phys.* **75**, 121 (2003).
- [29] A. J. Koning, D. Rochman, J. C. Sublet, N. Dzysiuk, M. Fleming, and S. van der Marck, TENDL: Complete nuclear data library for innovative nuclear science and technology, *Nucl. Data Sheets* **155**, 1 (2019), Special Issue on Nuclear Reaction Data.
- [30] D. P. Kingma and M. Welling, Auto-encoding variational bayes, in *Proceeding of the 2nd International Conference on Learning Representations* (ICLR, 2014).
- [31] D. Ha, A. M. Dai, and Q. V. Le, Hypernetworks, [arXiv:1609.09106](#).
- [32] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, Implicit neural representations with periodic activation functions, in *Advances in Neural Information Processing Systems*, edited by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Curran Associates, Inc., 2020), Vol. 33, pp. 7462–7473.
- [33] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, An introduction to variational methods for graphical models, *Mach. Learn.* **37**, 183 (1999).
- [34] A. F. Agarap, Deep learning using rectified linear units (ReLU), [arXiv:1803.08375](#).
- [35] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, NeRF: Representing scenes as neural radiance fields for view synthesis, *Commun. ACM* **65**, 99 (2022).
- [36] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, Occupancy networks: Learning 3D reconstruction in function space, in *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (IEEE, Piscataway, NJ, 2019), pp. 4460–4470.
- [37] F. Szatkowski, K. J. Piczak, P. Spurek, J. Tabor, and T. Trzciński, Hypernetworks build implicit neural representations of sounds, in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (Springer, Berlin, 2023), pp. 661–676.
- [38] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, Neural message passing for quantum chemistry, in *International Conference on Machine Learning* (PMLR, 2017), Vol. 70, pp. 1263–1272.
- [39] F. M. Bianchi, D. Grattarola, L. Livi, and C. Alippi, Graph neural networks with convolutional ARMA filters, *IEEE Trans. Pattern Anal. Mach. Intell.* **44**, 3496 (2022).
- [40] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, Gated graph sequence neural networks, [arXiv:1511.05493](#).
- [41] W. Hamilton, Z. Ying, and J. Leskovec, Inductive representation learning on large graphs, in *NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems* (Curran Associates, Reed Hook, NY, 2017).
- [42] I. Loshchilov and F. Hutter, Decoupled weight decay regularization, in *Proceedings of the 7th International Conference on Learning Representations* (ICLR, 2019).
- [43] L. N. Smith, A disciplined approach to neural network hyperparameters: Part 1—learning rate, batch size, momentum, and weight decay, [arXiv:1803.09820](#).
- [44] L. van der Maaten and G. Hinton, Visualizing data using t-SNE, *J. Mach. Learn. Res.* **9**, 2579 (2008).