Graph Neural Networks for Tensor Product Decompositions of Lie Algebra Representations

Max Vargas [1] Helen Jenne [1] Davis Brown [1 2] Henry Kvinge [1 3]

Abstract

Advances in AI promise to accelerate progress in mathematics by automating the process of pattern recognition within large mathematically-motivated datasets. In this extended abstract, we report on work-in-progress using graph neural networks (GNNs) to predict properties of tensor products of Lie algebra representations. First, we impose a graph structure on the weight lattice associated to a finite-dimensional semisimple Lie algebra $\mathfrak{g}$. This structure is used to generate datasets for predicting decomposition factors of tensor products between finite-dimensional irreducible representations of $\mathfrak{g}$. We find that while this problem quickly grows in complexity, GNNs have the potential to learn algorithmic rules for predicting the structure of tensor products.

1. Introduction

In mathematics, the exhaustive analysis of large collections of examples is a key step in extracting new mathematical insights. At the same time, innovation in machine learning has increased the ability of domain scientists to uncover complex patterns in large repositories of data. This includes datasets that have been curated for the purpose of AI-assisted mathematical discovery (Chau et al., 2025). Further, graph-based techniques have proven capable of advancing these frontiers in cases where individual instances in the dataset can have complex pairwise relationships, and have proven useful in several problems in discrete math (He et al.; Cappart et al., 2023).

In this work we describe how graph neural networks (GNNs) can be trained to predict the decomposition factors appearing in tensor products of Lie algebra representations (a fundamental problem in the field of representation theory). Similar problems abound in representation theory, several of which are open such as calculation of Kronecker coefficients and tensor decompositions of symmetric group representations (Lee, 2023; Butbaia et al., 2025).

[1]Pacific Northwest National Laboratory, Seattle, USA [2]University of Washington, Seattle, USA [3]University of Pennsylvania, Philadelphia, USA. Correspondence to: Max Vargas <max.vargas@pnnl.gov>.

The second AI for MATH Workshop at the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. Copyright 2025 by the author(s).

2. Background

Unless otherwise noted, all vector spaces considered throughout will be finite-dimensional over the complex numbers. More detail can be found in the appendix.

2.1. Lie Algebras and Representations

Lie algebras play a foundational role in representation theory with connections in (but not limited to) combinatorics, number theory, and particle physics (Ray, 2006; Georgi, 2019). Stated briefly, they are vector spaces $\mathfrak{g}$ which come equipped with a special kind of *noncommutative*, *nonassociative* multiplication rule often written with bracket notation $[\cdot,\cdot] : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$.

Rather than directly study $\mathfrak{g}$ itself, it is standard to study representations of $\mathfrak{g}$; these are vector spaces V together with the data of an operation $\mathfrak{g} \times V \to V, (x, v) \mapsto x \cdot v$ satisfying certain properties. While it may seem circuitous to study representations rather than $\mathfrak{g}$ itself, it is well-known how to recover the Lie algebra from its representations.

Among the first major achievements in a course on (semisimple) Lie algebras is the classification of *irreducible representations*. These are the representations, V, of $\mathfrak{g}$ for which there exists no proper linear subspace $W \subsetneq V$ which is also a representation of $\mathfrak{g}$. The classic result identifies a correspondence between irreducible representations and a set, Λ^+, known as the *dominant weights* of $\mathfrak{g}$. Elements of Λ^+ can be written as nonnegative integral combinations of distinguished *fundamental weights*. We will denote by $L(\lambda)$ the irreducible representation associated to $\lambda \in \Lambda^+$.

2.2. Tensor Products

Given two representations V and W of a Lie algebra $\mathfrak{g}$, the vector space $V \otimes W$ naturally inherits the structure of a representation of $\mathfrak{g}$. If $L(\lambda)$ and $L(\mu)$ are two irreducible

Table 1. Classification accuracies for GNNs trained for binary classification on tensor product decompositions.

$\mathfrak{g}$	λ	#Layers	Test Acc.	Baseline
$\mathfrak{sl}_3$	(3,2)	3	99.7	75.8
$\mathfrak{sl}_3$	(4,4)	9	99.8	85.7
$\mathfrak{sl}_3$	(11,8)	18	93.0	82.7
$\mathfrak{sl}_4$	(3,2,0)	3	99.9	85.4
$\mathfrak{sl}_4$	(3,2,2)	6	99.0	93.4
$\mathfrak{sp}_6$	(3,2,0)	6	96.4	51.6

representations associated to the dominant weights $\lambda, \mu \in \Lambda^+$, then $L(\lambda) \otimes L(\mu)$ will typically not be irreducible. We can, however, decompose this tensor product as a direct sum of irreducible representations:

$$L(\lambda) \otimes L(\mu) \simeq \bigoplus_{\nu \in \Lambda^+} L(\nu)^{\oplus c^{\nu}_{\lambda\mu}},$$

where the coefficients $c^{\nu}_{\lambda\mu}$ are the multiplicities of $L(\nu)$. As we are dealing with finite-dimensional representations, only finitely many $c^{\nu}_{\lambda\mu}$ are nonzero for any pair λ, μ.

3. Experiments and Results

In our experiments, we fix a Lie algebra $\mathfrak{g}$ and dominant weight $\lambda \in \Lambda^+$. Our goal is to train a neural network to predict which coefficients $c^{\nu}_{\lambda\mu}$ are nonzero as μ and ν vary. To take advantage of structural properties of Λ^+, we construct a graph using fundamental weights to define edges.

3.1. Graph Structures

We build a directed, weighted, graph Q whose vertices are the dominant weights Λ^+. Two vertices μ, ν are connected if there exists a fundamental weight, λ_*, so that $\lambda_* = \mu - \nu$. The edge will be weighted by λ_* itself. There will also be an edge in the opposite direction weighted by $-\lambda_*$. Sample data will consist of finite subgraphs $Q_\mu \subset Q$ for each $\mu \in \Lambda^+$, corresponding to the product $L(\lambda) \otimes L(\mu)$. The vertices are given by

$$\mathrm{V}(Q_\mu) = \{\nu \in \Lambda^+ \mid ||\mu - \nu|| \leq ||\lambda||\} \cup \{\text{src}\}.$$

We introduce a source vertex connected to μ to indicate where the tensor product is taken. A GNN is then trained on the graphs Q_μ to predict, for each vertex $\nu \in Q_\mu$, whether or not the coefficient $c^{\nu}_{\lambda\nu}$ is nonzero.

3.2. Results

GNNs are able to learn features related to tensor product decompositions, but the classification task quickly grows in complexity. Results in several cases are shown in Table 1. We learned that choices of successful architecture will depend on properties of λ. In particular, the depth of the network should grow with $||\lambda||$. For example, with $\mathfrak{sl}_3$, a 3-layer GNN trained on a dataset for $\lambda = (3, 2)$ succeeded in learning which coefficients $c^{\nu}_{\lambda\mu}$ are nonzero, but even a 18-layer network struggled with the largest weight.

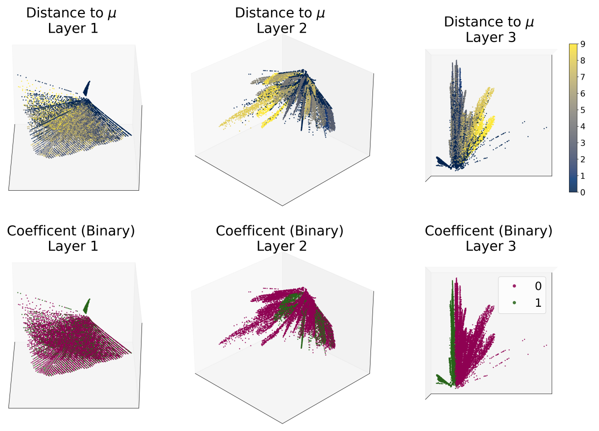

Figure 1. Principal Component embeddings of graph vertices. **Top:** Colored by graph distance from μ. **Bottom:** Colored whether the coefficient $c^{\nu}_{\lambda\mu}$ is zero.

3.3. Exploring Embeddings

In order to glean information about the algorithmic process underpinning a neural network trained on this task, we provide a brief exploration of the embeddings in a simple case in Figure 1. In the first layer, the network is able to separate a cluster of vertices ν with uniform values of $c^{\nu}_{\lambda\mu}$. This cluster corresponds to some of the nearest vertices to μ for each graph Q_μ. Layer 2 shows improved clustering with respect to the binary classification task with striations corresponding to the distance of a vertices from μ, suggesting that this is a quick proxy for determining if $c^{\nu}_{\lambda\mu}$ is nonzero.

4. Limitations and Future Work

Our experiments show the potential of using GNNs to learn tensor product decompositions, but currently require us to train a network for each weight λ. That said, there is value in learning the rules for specific families of λ. Another limitation is training efficiency and stability; a single model can take over 12 hours to converge on an H100 GPU. Optimizations to accelerate this process will let us explore the algorithmic reasoning of GNNs that much quicker.

Structures that are strikingly analogous to those in this investigation persist throughout mathematics. It will be interesting to apply this technique to new cases where the combinatorics is less familiar.

References

Butbaia, G., Lee, K.-H., and Ruehle, F. Interpretable machine learning for kronecker coefficients, 2025. URL https://arxiv.org/abs/2502.11774.

Cappart, Q., Chételat, D., Khalil, E. B., Lodi, A., Morris, C., and Veličković, P. Combinatorial optimization and reasoning with graph neural networks. *Journal of Machine Learning Research*, 24(130):1–61, 2023.

Chau, H., Jenne, H., Brown, D., He, J., Raugas, M., Billey, S., and Kvinge, H. Machine learning meets algebraic combinatorics: A suite of datasets capturing research-level conjecturing ability in pure mathematics, 2025. URL https://arxiv.org/abs/2503.06366.

Georgi, H. *Lie Algebras in Particle Physics: From Isospin to Unified Theory*. CRC Press, Boca Raton, FL, 2019.

He, J., Jenne, H., Chau, H., Brown, D., Raugas, M., Billey, S., and Kvinge, H. Machines and mathematical mutations: Using GNNs to characterize quiver mutation classes. Preprint. Accepted to ICML 2025.

Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., and Leskovec, J. Strategies for pre-training graph neural networks, 2020. URL https://arxiv.org/abs/1905.12265.

Humphreys, J. E. *Introduction to Lie Algebras and Representation Theory*. Springer, New York, NY, 1972.

Lee, K.-H. Machine-learning kronecker coefficients, 2023. URL https://arxiv.org/abs/2306.04734.

Ray, U. *Automorphic Forms and Lie Superalgebras*. Springer Dordrecht, 2006.

The Sage Developers. *SageMath, the Sage Mathematics Software System (Version x.y.z)*, 2025. https://www.sagemath.org.

A. Extra Mathematical Background

Here, we provide basic information about Lie algebras. A standard reference on the subject is (Humphreys, 1972). Throughout this report, a *Lie algebra* $\mathfrak{g}$ is a vector space over the complex numbers, $\mathbb{C}$, with a binary multiplication operation $[\cdot,\cdot] : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$ that satisfies the following properties for all $a, b \in \mathbb{C}$ and $x, y, z \in \mathfrak{g}$.

- (Bilinearity) $[ax + by, z] = a[x, y] + b[y, z]$
- (Skew-symmetric) $[x, x] = 0$
- (Jacobi Identity) $[x, [y, z]] + [z, [x, y]] + [y, [z, x]] = 0$

Notice that a Lie algebra does not satisfy the traditional properties one might expect from a multiplication rule. In particular, the Lie bracket *does not* have the associative property ($[x, [y, z]] \neq [[x, y], z]$) nor the commutative property ($[x, y] \neq [y, x]$). In fact, $\mathfrak{g}$ is guaranteed to always be anti-commutative in that $[x, y] = -[y, x]$ for all $x, y \in \mathfrak{g}$.

A.1. Representations and Weight Spaces

Rather than directly study $\mathfrak{g}$ itself, it is standard to study representations of $\mathfrak{g}$. A *representation* of $\mathfrak{g}$ is the data of a vector space V along with an operation $\mathfrak{g} \times V \to V, (x, v) \mapsto x \cdot v$ that satisfies the following properties.

- For any $a, b \in \mathbb{C}$, $x, y \in \mathfrak{g}$ and $v \in V$:
$$(ax + by) \cdot v = a(x \cdot v) + b(y \cdot v)$$
- For any $a, b \in \mathbb{C}$, $x \in \mathfrak{g}$ and $v, w \in V$:
$$x \cdot (av + bw) = a(x \cdot v) + b(y \cdot v)$$
- For any $x, y \in \mathfrak{g}$ and $v \in V$:
$$[x, y] \cdot v = x \cdot v - y \cdot v$$

Given a representation V, there is a well-known decomposition of V into *weight spaces*, $V = \bigoplus_\lambda V_\lambda$. While the linear subspaces V_λ are generally not themselves proper representations of $\mathfrak{g}$, they still carry a rich linear-algebraic structure. The values of λ are called *weights*, and for any $\mathfrak{g}$ the permissible values of λ turn out to form a lattice inside of some ambient Euclidean space.

A.2. Examples

We briefly present some introductory examples to illustrate the type of combinatorial and graph-based data that we use in our experiments. Many classical examples of Lie algebras can be realized as vector subspaces of matrix algebras — a vector space consisting of square matrices whose multiplication rule is given by matrix multiplication. We stress, however, that the Lie bracket is not given by matrix multiplication. If we let Mat_n denote the set of $n \times n$ matrices with coefficients in $\mathbb{C}$, we can always view Mat_n as a Lie algebra with a Lie bracket defined by

$$[X, Y] = XY - YX \tag{1}$$

for all matrices $X, Y \in \mathrm{Mat}_n$. In the following examples, the Lie bracket will be defined through this formula.

$\mathfrak{sl}_2$:

The simplest Lie algebra with nontrivial structure is $\mathfrak{g} = \mathfrak{sl}_2$, the smallest in the family of special linear Lie algebras. This Lie algebra can be explicitly described as a 3-dimensional subspace of 2×2 matrices with trace zero. It has a basis given by $e = \left(\begin{smallmatrix} 0 & 1 \\ 0 & 0 \end{smallmatrix}\right)$, $f = \left(\begin{smallmatrix} 0 & 0 \\ 1 & 0 \end{smallmatrix}\right)$, and $h = \left(\begin{smallmatrix} 1 & 0 \\ 0 & -1 \end{smallmatrix}\right)$. The Lie bracket $[\cdot,\cdot] : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$ is given by Equation 1.

The weight lattice of $\mathfrak{sl}_2$ is given by $\mathbb{Z}$, with dominant weights being those elements of $\mathbb{Z}^+$. There is a single fundamental weight corresponding to 1. Given two $m, n \in \mathbb{Z}^+$, the tensor product $L(m) \otimes L(n)$ can be decomposed using the *Clebsch-Gordon* rule:

$$L(m) \otimes L(n) \simeq \bigoplus_{i=0}^{\min(m,n)} L(m + n - 2i).$$

$\mathfrak{sl}_3$:

The next simplest case is $\mathfrak{g} = \mathfrak{sl}_3$. As before, this comprises the linear subspace consisting of matrices whose trace is zero. It is 8-dimensional with a basis given by the following matrices.

$$e_1 = \left(\begin{smallmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{smallmatrix}\right),\ f_1 = \left(\begin{smallmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{smallmatrix}\right),\ h_1 = \left(\begin{smallmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{smallmatrix}\right)$$
$$e_2 = \left(\begin{smallmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{smallmatrix}\right),\ f_2 = \left(\begin{smallmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{smallmatrix}\right),\ h_2 = \left(\begin{smallmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{smallmatrix}\right)$$
$$[e_1, e_2] = \left(\begin{smallmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{smallmatrix}\right),\ [f_1, f_2] = \left(\begin{smallmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{smallmatrix}\right)$$

The fundamental weights of $\mathfrak{sl}_3$ are given by $\lambda_1 = (2/3, -1/3, -1/3)$ and $\lambda_2 = (2/3, -1/3, -1/3)$ and are conventionally projected into two dimensions as shown in Figure 2. However, here it is more convenient to identify them with $\lambda_1 = (1, 0)$ and $\lambda_2 = (1, 1)$ using the SageMath package (The Sage Developers, 2025). This identification can be made through an intimate connection between $\mathfrak{sl}_3$ and $\mathfrak{gl}_3$, the *general linear Lie algebra*. The weight lattice is those points of the form $\{a\lambda_1 + b\lambda_2 \mid a, b \in \mathbb{Z}\} \subset \mathbb{C}^2$. In Figure 2, we illustrate a standard depiction of the weight lattice and dominant weights for $\mathfrak{sl}_3$, as well as the nonzero summands of the tensor product $L((4.2)) \otimes L((3, 2))$.

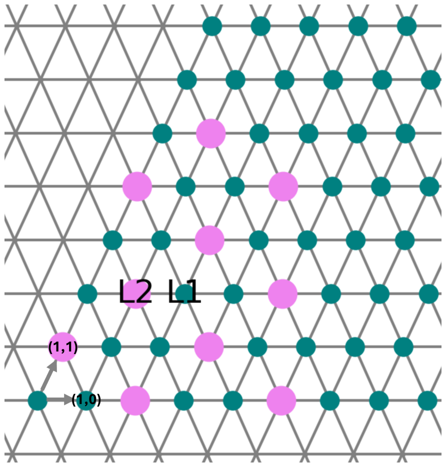

Figure 2. Illustration of the $\mathfrak{sl}_3$ weight lattice and nonzero summands for the tensor product of representations $L1 = (4, 2)$ and $L2 = (3, 2)$. Dominant weights are illustrated in teal, and nonzero summands are in pink.

B. Neural Network Training

B.1. Architecture

Our experiments use the DirGINE architecture for message passing (He et al.). This is a modified version of the Graph Isomorphism Network (GIN) (Hu et al., 2020) to support edge features that is built to operate with a message passing scheme. The features for vertex v in the ℓ-th layer of the network is given by

$$x_v^{(\ell)} = \mathrm{ReLU}\Bigg(W^{(\ell)} x_v^{(\ell-1)} + \sum_{(u,v)} \varphi_{\text{in}}^{(\ell)}(x_v^{(\ell-1)}, e_{uv}) + \sum_{(v,w)} \varphi_{\text{out}}^{(\ell)}(x_v^{(\ell-1)}, e_{vw}) \Bigg),$$

where $W^{(\ell)}$, $\varphi_{\text{in}}^{(\ell)}$ and $\varphi_{\text{out}}^{(\ell)}$ are learnable parameters and functions.

B.2. Hyperparameters

We found that batch size did not play a critical role in the convergence of our networks. Networks reported herein were trained with a batch size of 512.

We swept over learning rates within [1e-1, 5e-2, 1e-2, 5e-3, 1e-3, 5e-4, 1e-4], in many cases finding that choices between 1e-3 and 1e-2 performed well. In cases where a GNN was not able to considerably improve upon a baseline, we found that the learning rate still had minor effect.

Other parameters that we experimented with included the depth of the model and the dimension of the hidden layers. As the size of λ grew, it was critical to also increase the number of layers. This is likely due to the message passing mechanism providing a limit on the rate that information can propagate as the graph passes through the network. We did not observe a significant effect from changing the hidden dimension and found that 32 dimensions worked well.

Table 2. Dataset sizes for various Lie algebras and λ.

$\mathfrak{g}$	λ	# GRAPHS	MAX VERTICES	COEFFS.
$\mathfrak{sl}_3$	(3,2)	2271	49	107K
$\mathfrak{sl}_3$	(4,4)	2271	113	237K
$\mathfrak{sl}_3$	(11,8)	2271	613	1.1M
$\mathfrak{sl}_4$	(3,2,0)	4343	257	852K
$\mathfrak{sl}_4$	(3,2,2)	4343	515	1.6M
$\mathfrak{sp}_6$	(3,2,0)	4340	257	852K

B.3. Datasets

Table 2 presents some basic statistics regarding the datasets generated for each Lie algebra and each dominant weight λ. We focus on several examples where $\mathfrak{g}$ is a Lie algebra of classical type, specifically $\mathfrak{sl}_2$, $\mathfrak{sl}_3$, and $\mathfrak{sp}_3$ corresponding to special linear and symplectic Lie algebras. Fixing a dominant weight $\lambda \in \Lambda^+$, we use the SageMath software package to compute a dataset of coefficients $c_{\lambda\mu}^{\nu}$ for a set $\mu \in \Lambda^+$ bounded by some fixed constant; $|\mu| \leq C$. We then store this data using the graph structure outlined above. After first constructing Q_μ and then populating vertices ν with nonzero coefficients $c_{\lambda\mu}^{\nu}$, the remaining vertices in Q_μ are stored with a null coefficient value.

B.4. Training Efficiency

Training the GNNs takes a surprising amount of wall-clock time, and it would be a useful challenge to optimize the training efficiency of our method. Given that training efficiency is not our main objective, a single training epoch could take longer than 23 seconds on an H100 GPU for a 9-layer network with a batch size of 512, a training set of 1580 graphs (50 vertices each), and a batch size of 512. In the worst case, training a GNN with $\mathfrak{g} = \mathfrak{sp}_6$ and $\lambda = (3, 2, 0)$ took 1.5 min/epoch and took 12 hours to complete 480 epochs training epochs. Another artifact seen during training is that it is common to see collapses. For example, in the training metrics recorded in Figure 3, we see that the training loss will suddenly and drastically shoot up and performance will drop. The network then needs time to recover the lost gains.

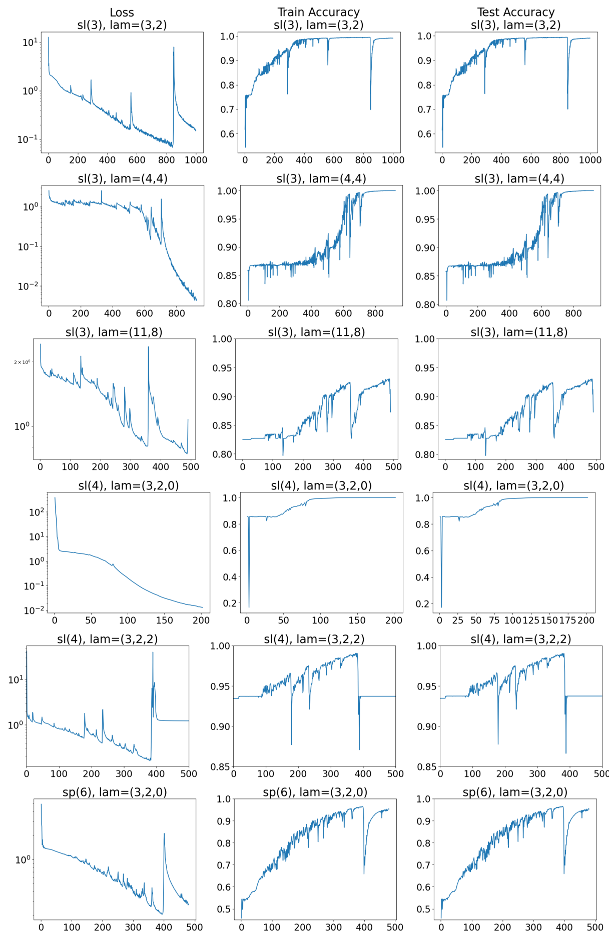

Figure 3. Training metrics for GNN training. In order of rows, GNNs were trained with 3, 6, 18, 3, 6, and 6 layers.