City Research Online

City, University of London Institutional Repository

Citation: Chen, H-Y., He, Y., Lal, S. & Majumder, S. (2021). Machine learning Lie structures & applications to physics. Physics Letters B, 817, 136297. doi: 10.1016/j.physletb.2021.136297

This is the published version of the paper.

This version of the publication may differ from the final published version.

Permanent repository link: https://openaccess.city.ac.uk/id/eprint/26089/

Link to published version: https://doi.org/10.1016/j.physletb.2021.136297

Copyright: City Research Online aims to make research outputs of City, University of London available to a wider audience. Copyright and Moral Rights remain with the author(s) and/or copyright holders. URLs from City Research Online may be freely distributed and linked to.

Reuse: Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge. Provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way.

City Research Online: http://openaccess.city.ac.uk/ publications@city.ac.uk

Physics Letters B 817 (2021) 136297

Contents lists available at ScienceDirect

Physics Letters B

www.elsevier.com/locate/physletb

Machine learning Lie structures & applications to physics

Heng-Yu Chen[a], Yang-Hui He[b,c,d], Shailesh Lal[e], Suvajit Majumder[b,*]

[a] *Department of Physics, National Taiwan University, Taipei 10617, Taiwan*
[b] *Department of Mathematics, City, University of London, London EC1V0HB, UK*
[c] *Merton College, University of Oxford, OX1 4JD, UK*
[d] *School of Physics, NanKai University, Tianjin, 300071, PR China*
[e] *Faculdade de Ciencias, Universidade do Porto, 687 Rua do Campo Alegre, Porto 4169-007, Portugal*

ARTICLE INFO

Article history:
Received 10 November 2020
Received in revised form 23 March 2021
Accepted 14 April 2021
Available online 19 April 2021
Editor: N. Lambert

ABSTRACT

Classical and exceptional Lie algebras and their representations are among the most important tools in the analysis of symmetry in physical systems. In this letter we show how the computation of tensor products and branching rules of irreducible representations is machine-learnable, and can achieve relative speed-ups of orders of magnitude in comparison to the non-ML algorithms.

© 2021 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/). Funded by SCOAP^3.

1. Introduction & summary

Lie algebras are an integral part of modern mathematical physics. Their representation theory governs every field of physics from the fundamental structure of particles to the states of a quantum computer. Traditionally, an indispensable tool to the high energy physicist is the extensive tables of [1]. More contemporary usage, with the advent of computing power of the ordinary laptop, has relied on the likes of highly convenient software such as “LieART” [2]. Such computer algebra methods, especially in conjunction with the familiarity of the Wolfram programming language to the theoretical physicists, are clearly destined to play a helpful rôle.

In parallel, a recent programme of applying the techniques from machine-learning (ML) and data science to study various mathematical formulae and conjectures had been proposed [3–5]. Indeed, while the initial studies were inspired by and brought to string theory in timely and independent works in [3,4,6–8], experimentation of whether standard techniques in neural regressors and classifiers could be carried over to study diverse problems have taken a life of its own. These have ranged from finding bundle cohomology on varieties [7,9,10], to distinguishing elliptic fibrations [11] and invariants of Calabi-Yau threefolds [12], to machine-learning the Donaldson algorithm for numerical Calabi-Yau metrics [13], to the algebraic structures of discrete groups and rings [14], to the BSD conjecture & Langlands programme in number theory [15–17], to quiver gauge theories and cluster algebras [18], to patterns in particle masses [19], to knot invariants [20], to statistical predictions and model-building in string theory [21–23], to classifying combinatorial properties of finite graphs [24] etc. Moreover, the very structures of quantum field theory and holography [25–27] have also been proposed to be closely related to suitable neural networks.

In this letter, we continue this exciting programme and apply machine learning techniques to another indispensable concept for physicists, namely the ubiquitous continuous symmetries as encoded by Lie groups/algebras. Physicists have long used them to classify from the phases of matters to the spectrum of elementary particles. As listed earlier, machine learning techniques have provided us with a powerful new approach towards various classification problems of physical interests.[1] Here we would like to ask whether the essential structures of Lie group can also be learned by machine. Specifically, by this we mean whether neural network (NN) classifiers and regressors can, after having seen enough samples of typical calculations such as tensor decomposition or branching rules – both known to be heavily computationally expensive, as we will shortly see – predict the result more efficiently.

As a comparison, let us also mention a somewhat surprising result from [14], where some fundamental structures of algebra, viz., certain properties of finite groups and finite rings, seem to be machine-learnable. Difficult problems in representation theory such as recognizing whether a finite group is simple or not by “looking” at the Cayley multiplication table, or whether random permutation matrices (Sudoku) possess group structure, etc., can

---

* Corresponding author.
E-mail addresses: heng.yu.chen@phys.ntu.edu.tw (H.-Y. Chen), hey@maths.ox.ac.uk (Y.-H. He), shailesh.hri@gmail.com (S. Lal), Suvajit.Majumder@city.ac.uk (S. Majumder).

[1] There have been other interesting works on detecting physical symmetries using machine-learning [28,29].

https://doi.org/10.1016/j.physletb.2021.136297
0370-2693/© 2021 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/). Funded by SCOAP^3.

be classified by a support vector machine very efficiently without recourse to the likes of Sylow theorems which are computationally expensive.

In this letter we are motivated by the question of whether and how much one can machine-learn the essential information about classical, and exceptional Lie algebras as tabulated in standard texts such as Slansky [1]. Specifically, we address the two fundamental problems in the representation theory of Lie algebras that is crucial to physics – the tensor product decomposition and the branching rules to a sub-algebra – and show that these salient structures are machine learnable.

In particular we show that a relatively simple forward-feeding neural network can predict to high accuracy and confidence, the number of irreducible representations ("irreps") that appear in a tensor product decomposition, which we refer to as the *length* of the decomposition. Our findings for classical and exceptional algebras are summarized in Table 1. We subsequently show that a neural network can also predict with high accuracy, the presence or absence of a given irreducible representation of a maximal sub-algebra within an irreducible representation of a parent algebra. The neural network is capable of predicting, for example, the presence of bi-fundamentals in $SU(3) \times SU(2)$ for a given representation of $SU(5)$ to an accuracy of 88% and a confidence of 0.735.

We remark that our classification problems were also addressed with various standard classifiers, such as Naive Bayes, nearest neighbours and support vector machines. We found that the NN with the architecture shown below in Fig. 2 significantly outperformed them. For example, using Logistic Regression for the analysis of Table 2 for the A_m algebra yields a test accuracy of 0.823 and a confidence of 0.64. The results from support vector machines are similar. This is in line with previous observations where NNs with similar architectures perform well for a variety of problems, such as the computation of topological invariants of manifolds [3,21], and finite graph invariants [24].

2. Tensor products and branching rules learnt by a neural network

2.1. Tensor products

2.1.1. Predicting the length of generic tensor decompositions

Let us begin with a simple ML experiment. One of the most important computations for Lie groups/algebras is the decomposition of the tensor product of two representations into a direct sum of irreducible representations for a given group G: $R_1 \otimes R_2 = \bigoplus_{r \in \text{irreps}} a_r R_r$, where $a_r \in \mathbb{Z}_{\geq 0}$ are the multiplicity factors. To be concrete, let us first consider $A_m = SU(m+1)$. Every irreducible representation ("irrep") of A_m is specified by a highest-weight vector $\vec{v}$, which is a rank m vector of non-negative integer components. Throughout this letter, we will use

$$\vec{v} \equiv (v_1, \ldots, v_r) \tag{2.1}$$

to denote the weight vector for a Lie algebra of rank r. When the context is clear, an integer with the vector over-script is understood to be a vector of the same integer entry, e.g., $\vec{4} = (4, 4, \ldots, 4)$.

As the entries of $\vec{v}$ increase in magnitude, the dimension of the corresponding irrep $R_{\vec{v}}$ can grow dramatically. For instance, for $A_3 = SU(4)$, $\dim R_{\vec{v}=(a,b,c)} = \frac{1}{12}(a+1)(b+1)(c+1)(a+b+2)(b+c+2)(a+b+c+3)$. This makes the task of identifying the precise irreps contained in a tensor decomposition rather laborious.

We start with two weight-vectors $\vec{v}_1, \vec{v}_2$. Their rank m is chosen randomly from {1,2,...,8}. Then, we randomly generate a pair quinary vectors $\vec{v}_1, \vec{v}_2$ of rank m, and compute their tensor decomposition into irreducible representations:

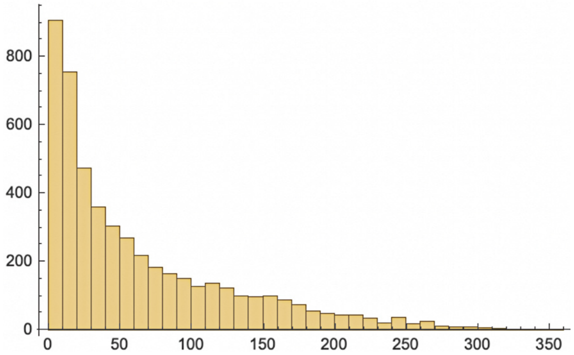

Fig. 1. Distribution of the number of distinct irreps in the tensor decomposition of $R_{\vec{v}_1} \otimes R_{\vec{v}_2}$ for A_m with randomized $1 \leq m \leq 8$ and randomized ternary weight vectors $\vec{v}_{1,2}$. The horizontal axis denotes the length of $R_{\vec{v}_1} \otimes R_{\vec{v}_2}$ and the vertical axis denotes the corresponding frequency.

$$R_{\vec{v}_1} \otimes R_{\vec{v}_2} = \bigoplus_{r \in \text{irreps}} a_r R_r \ . \tag{2.2}$$

This computation, although algorithmic, is non-trivial. Even the relatively simple question of *how many distinct irreps, along with their multiplicities, are there on the RHS* or what we call the *length* of a given tensor decomposition, is not immediately obvious just by looking at the vectors $\vec{v}_1$ and $\vec{v}_2$. For example,

$$\begin{aligned} R_{[0,1]} \otimes R_{[2,1]} &= \mathbf{8} \oplus \mathbf{10} \oplus \mathbf{27} \ ; \\ R_{[1,0,1]} \otimes R_{[0,2,0]} &= \overline{\mathbf{45}} \oplus \mathbf{20}' \oplus \mathbf{175} \oplus \mathbf{45} \oplus \mathbf{15} \ . \end{aligned} \tag{2.3}$$

It is difficult to see a priori that one decomposition would be of length 3 while the other would be of length 5; and one needs to actually compute the respective tensor decomposition to know the answer. It took several hours using LieART to perform five thousands decompositions.[2] To get an idea of their distributions, we show the histogram of the length: indeed there is a huge variation from 1 to over 350. A significant improvement in the running time (from hours to a few minutes) can be attained by capping off the maximum dimension of the irreps (say to 10,000). The distribution of the lengths of the decompositions vs frequency histogram, is depicted in Fig. 1.

Let us next consider a simple binary classification problem using the data generated by LieART: can ML distinguish tensor decompositions of length ≥ 70 and of length < 70? The length 70 is chosen since it splits the data rather evenly into around five thousands each. To uniformize the input vectors, for the rank $m < 8$, we also pad both $\vec{v}_{1,2}$ to the right with -1 (a meaningless number in this context) and stack them on top of each other. Thus, our input is a 2×8 matrix with integer entries for $1 \leq m \leq 8$. This step is essential for using a single NN for learning data for Lie algebras of varying ranks (it is for $A_m, 1 < m < 8$ here). For the majority of our experiments, we use a feed-forward neural network classifier built in `Mathematica` [32] with the architecture shown in Fig. 2. We also reproduced these results with a similar 2-layer architecture on `Keras` [30], with `selu` activated neurons to obtain similar accuracy and confidence. Finally, we need to ensure that the last softmax is rounded to 0 or 1 according to our binary categories.

---

[2] Care must be taken to find five thousands distinct pairs $(\vec{v}_1, \vec{v}_2)$ amidst the randomizations so as not to bias the input.

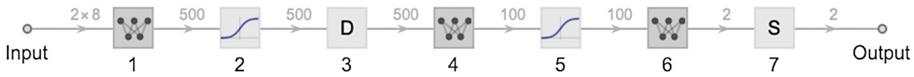

Fig. 2. The neural network architecture. S is the softmax activation and D is a dropout layer with probability 0.2. The hidden-layer neurons $(1, 2)$ and $(4, 5)$ are fully connected and sigmoid activated.

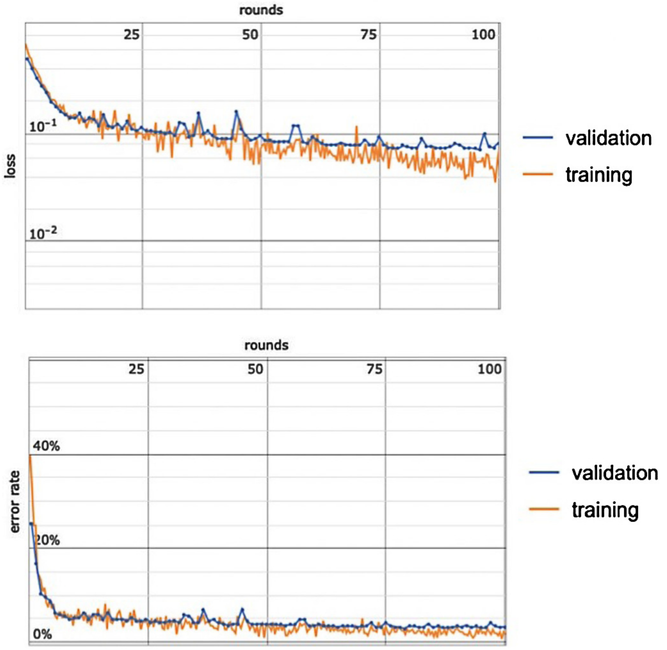

Fig. 3. Loss-function (top), and error-rate (below) for training and validation for A_m are plotted against number of epochs.

The results of our training and learning for A_m are depicted in Fig. 3. The data was partitioned into 64% training, 16% validation, and 20% test splits. The network was trained on the training and validation sets and the test set was used purely for evaluating the trained network. The plots show a steady lowering in both the error-rate and loss-function as we increase the number of rounds of training and validation. We achieved accuracy 0.969, confidence 0.930, 5% error rate, and 0.1 loss function within one minute by training for 100 epochs using learning rate 10^{-3}, ADAM optimizer; which is excellent indeed. Throughout this letter, we will use "accuracy" to mean percentage agreement of predicted and actual values. In addition, in discrete classification problems it is also important to have a measure of "confidence" so that false positives/negatives can be noted. A widely used one is the so-called Matthews' Phi-coefficient ϕ (essentially a signed square-root of the chi-squared of the contingency table) [31], which is $\lesssim 1$ for predictions with good confidence.

The above experiment was also carried out with other classical, as well as exceptional Lie algebras with comparable success. The results are given in Table 1. We generated the same data size as in the A_m case, i.e. 5000, and used the same cap on the maximum dimension of the irreps (10,000). In contrast, though the dimension cap for exceptional groups was set to 120,000, it yielded far fewer data points. The lengths we split the data-sets on were chosen to generate a balanced data-set in each case. The accuracy of ML prediction was above .95 for each of these cases. The relatively lower accuracy for E_6 is caused by the low number of points available at low dimensions due to its relatively high rank: 903 data points below dimension of 120,000. Raising the dimension cap would improve the machine-learning, bringing it up to par with others, however the corresponding data generation using LieART would take days.

Table 1
The binary classification of product decomposition lengths. The splitting lengths yield a balanced dataset.

Group	Data size	Splitting length	Accuracy	Confidence
A_m	5000	70	0.969	0.930
B_m	5000	40	0.959	0.878
C_m	5000	40	0.969	0.921
D_m	5000	35	0.965	0.908
G_2	1275	110	0.946	0.891
E_6	903	30	0.898	0.795

Table 2
Training on low dimensional irreps, and testing on high dimensional ones.

Group	Train/Val accuracy	Test accuracy	Confidence
A_m	0.974/0.957	0.961	0.907
B_m	0.972/0.963	0.957	0.845
C_m	0.969/0.970	0.892	0.792
D_m	0.971/0.940	0.956	0.817
G_2	0.969/0.963	0.968	0.922
E_6	0.963/0.947	0.875	0.751

We also note that partitioning the data-sets at the 'midpoint' to generate balanced data-sets as we have done above is by no means necessary. As an example, we explored this classification problem for the A_m algebras but now organizing the data into partitions of varying lengths, *viz.* 20/80 through to 80/20. Here by a partition of length 20/80 we mean that a 'cutoff' decomposition length was chosen such that 20% of the decomposition lengths in the dataset are below this length, i.e. are denoted by the target variable $\mathrm{Y} = 0$ and the remaining 80% are above, and hence denoted by $\mathrm{Y} = 1$. In every case the Matthews' Phi-coefficient remains close to 1. In particular, for the 20/80 and 80/20 partition it is 0.98.

We can take this experiment one step further and train the neural net on low dimensional tensor decomposition data, then test its performance on higher dimensional cases. If successful this would immensely reduce the computation time. For example, obtaining the length of decomposition for two A_6 weight vectors $\vec{v}_1 = \vec{v}_2 = (2, 2, 2, 2, 2, 2)$ by brute force takes over 15 minutes on LieART while machine learning should estimate the length in a matter of seconds.

We retrained the NN in Fig. 2 on the same data for the classical and G_2 algebras generated by LieART for the previous experiment. However, the training set is now restricted to have both input weight vectors of dimension less than certain cut-off value, here taken to be 2,000. The *trained* neural network was subsequently evaluated on the test dataset consisting of input weight vectors of dimension ranging between 2,000 and 10,000. Our results are presented in Table 2 and Fig. 4.

2.1.2. Beyond binary classification

We now move beyond the simpler binary classification experiments done previously to a multi-class classification task, with the aim of predicting a range for the length of the product decomposition as opposed to the over/under estimates obtained above. For definiteness, let us take the A_m data and classify it into five classes, depending on whether the length of the product decomposition lies in the ranges 0 to 10, 10 to 25, 25 to 55, 55 to 115 and greater than 115. Fig. 5 shows a histogram with the class populations, and the training curves are displayed in Fig. 6. The neural

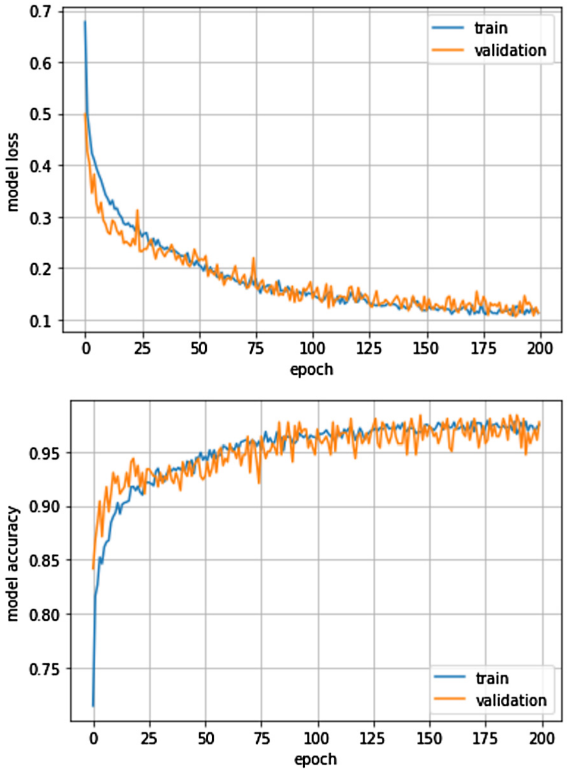

Fig. 4. Loss-function (top), and model accuracy (below) for training on A_m irreps, plotted against number of epochs.

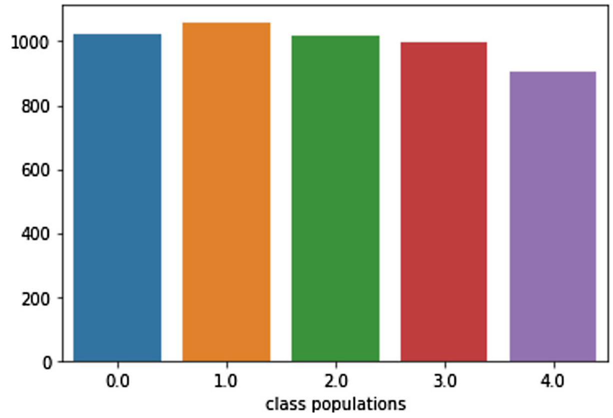

Fig. 5. Class Populations of length bins for A_m irreps.

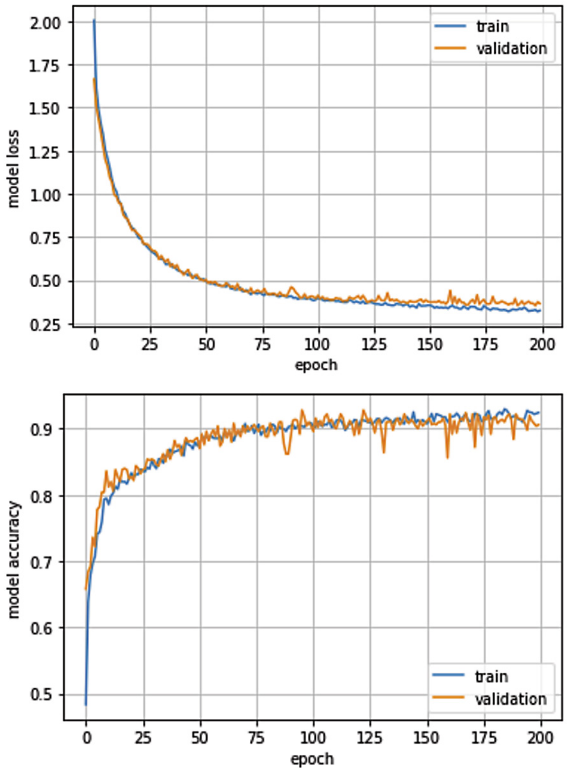

Fig. 6. Training curves for the quinary classification problem for A_m, plotted against number of epochs.

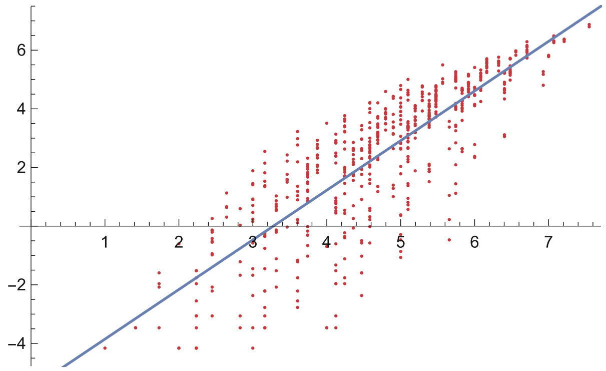

Fig. 7. Plot and fit of log of the time in seconds of the branching of irreps of $SU(5)$ versus the length of the weight-vector.

network reaches a ϕ-coefficient of 0.917 on the test set, and the confusion matrix is given by

$$\begin{pmatrix} 96 & 3 & 0 & 0 & 0 \\ 1 & 92 & 3 & 0 & 0 \\ 0 & 2 & 109 & 4 & 0 \\ 0 & 0 & 7 & 86 & 7 \\ 0 & 0 & 0 & 6 & 84 \end{pmatrix} \tag{2.4}$$

2.2. Branching rules

The next task on which we train our neural network of Fig. 2 is to learn about the branching rules for Lie algebras. Suppose we take a weight-vector of $SU(5)$, and restrict its entries from 0 to 4 (i.e., as quinary 4-vectors). Even though this may look rather harmless, the dimension of the corresponding irrep ranges from 1 for $\vec{0}$, to 9765625 for $\vec{4}$. When we decomposed these irreps of $SU(5)$ to those of its maximal sub-algebra $SU(3) \times SU(2) \times U(1)$, and found their explicit branching products, the time taken on LieART was easily seen to be exponential.[3] In Fig. 7, we plot the log of the time taken in seconds, versus the length of the weight-vector. The best fit is the line $-5.54361 + 1.69186x$. By extrapolation, the single irrep of $SU(5)$ corresponding to weight vector $\overrightarrow{10}$ would take over 20 years just to compute its branching products into $SU(3) \times SU(2) \times U(1)$.

In the rest of this section, we shall show the efficacy of using ML to predict presence/absence of any given representation of the maximal sub-algebra in a given irrep of $SU(5)$ and G_2 algebras. For concreteness, we look for bi-fundamental representations of $SU(3) \times SU(2)$ (with arbitrary values of $U(1)$ charges) in any given $SU(5)$ irrep. In the G_2 case, we restrict ourselves to bi-fundamental representation of $SU(2) \times SU(2)$ maximal subalgebra.

---

[3] Notice that as LieART is only capable of generating branching rule data for maximal subgroups, here we will focus on this simplest set of branching training data to illustrate the capability of neural network.

For the $SU(5)$ branching, we use first 800 irreps of smallest dimension as input vectors with binary output depending on presence/absence of a bi-fundamental rep of $SU(3) \times SU(2)$. The data was split into train/validation/test sets in the ratio 80/10/10. The neural network reached a test accuracy of 0.899 and a confidence of 0.813. The next best results were arrived at by a support vector classifier which reached an test accuracy of 0.838 and confidence of 0.677.

For the G_2 branching, we used 400 weight input vectors with dimensions below 4.7 million. Analogous to the $SU(5)$ case, the output was binary, depending on presence/absence of a bi-fundamental rep of $SU(2) \times SU(2)$. All classifiers, neural nets and otherwise, performed at the level of blind guessing in this case, which is possibly due to the relatively fewer input data as well as smaller number of features in the data.

3. Outlook

Given the ubiquity of Lie algebras and groups in physics, let us end this letter with some comments about the vast possibilities in applications to physics of our results, exemplifying with two which immediately come to mind.

In scattering processes, given a pair of incoming particles transforming under the irreps of certain global symmetry group, the outgoing particles can be classified via their tensor decompositions. The tensor decomposition prediction and extrapolation results in section 2.1 thus allow us to efficiently estimate the number of distinct outgoing particles. It would also be exciting to see if the NN upper bound estimate of the length of a given decomposition can help LieART package to work out its explicit terms within significantly shorter period.

Our choice of studying the branching of $SU(5)$ into its maximal subgroup $SU(3) \times SU(2) \times U(1)$ in section 2.2 was phenomenologically motivated. This hopefully can lead to an useful algorithm for testing whether a field transforming under $SU(5)$ GUT gauge group can yield descendants transforming under standard model gauge groups upon spontaneous symmetry breaking. We hope this will be useful for particle physics model building purposes.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

HYC is supported in part by Ministry of Science and Technology (MOST) through the grant 108-2112-M-002 -004 -. YHH is indebted to STFC UK, for grant ST/J00037X/1. SL's work was supported by the Simons Foundation grant 488637 (Simons Collaboration on the Non-perturbative bootstrap) and the project CERN/FIS-PAR/0019/2017. Centro de Fisica do Porto is partially funded by the Foundation for Science and Technology of Portugal (FCT) under the grant UID-04650-FCUP. SM's work is supported by SMCSE Doctoral Studentship at City, University of London.

References

[1] R. Slansky, Group theory for unified model building, Phys. Rep. 79 (1981) 1–128.

[2] R. Feger, T.W. Kephart, R.J. Saskowski, LieART 2.0 – a mathematica application for Lie algebras and representation theory, Comput. Phys. Commun. 257 (2020) 107490, arXiv:1912.10969 [hep-th], https://lieart.hepforge.org/.

[3] Y.H. He, Deep-learning the landscape, Science 365 (6452) (2019), arXiv:1706.02714 [hep-th].

[4] Y.H. He, Machine-learning the String Landscape, Phys. Lett. B 774 (2017) 564–568.

[5] Y.H. He, The Calabi-Yau landscape: from geometry, to physics, to machine-learning, arXiv:1812.02893 [hep-th], Springer, in press.

[6] D. Krefl, R.K. Seong, Machine learning of Calabi-Yau volumes, Phys. Rev. D 96 (6) (2017) 066014, arXiv:1706.03346 [hep-th].

[7] F. Ruehle, Evolving neural networks with genetic algorithms to study the String Landscape, J. High Energy Phys. 1708 (2017) 038, https://doi.org/10.1007/JHEP08(2017)038, arXiv:1706.07024 [hep-th].

[8] J. Carifio, J. Halverson, D. Krioukov, B.D. Nelson, Machine learning in the String Landscape, J. High Energy Phys. 1709 (2017) 157, https://doi.org/10.1007/JHEP09(2017)157, arXiv:1707.00655 [hep-th].

[9] C.R. Brodie, A. Constantin, R. Deen, A. Lukas, Machine learning line bundle cohomology, Fortschr. Phys. 68 (1) (2020) 1900087, https://doi.org/10.1002/prop.201900087, arXiv:1906.08730 [hep-th].

[10] M. Larfors, R. Schneider, Explore and exploit with heterotic line bundle models, Fortschr. Phys. 68 (5) (2020) 2000034, https://doi.org/10.1002/prop.202000034, arXiv:2003.04817 [hep-th].

[11] Y.H. He, S.J. Lee, Distinguishing elliptic fibrations with AI, Phys. Lett. B 798 (2019) 134889, arXiv:1904.08530 [hep-th].

[12] K. Bull, Y.H. He, V. Jejjala, C. Mishra, Machine learning CICY threefolds, Phys. Lett. B 785 (2018) 65, arXiv:1806.03121 [hep-th];
K. Bull, Y.H. He, V. Jejjala, C. Mishra, Getting CICY high, Phys. Lett. B 795 (2019) 700, arXiv:1903.03113 [hep-th].

[13] A. Ashmore, Y.H. He, B.A. Ovrut, Machine learning Calabi-Yau metrics, Fortschr. Phys. 68 (9) (2020) 2000068, arXiv:1910.08605 [hep-th].

[14] Y.H. He, M. Kim, Learning algebraic structures: preliminary investigations, arXiv:1905.02263 [cs.LG].

[15] L. Alessandretti, A. Baronchelli, Y.H. He, Machine learning meets number theory: the data science of Birch-Swinnerton-Dyer, arXiv:1911.02008 [math.NT].

[16] Y.H. He, E. Hirst, T. Peterken, Machine-learning dessins d'enfants: explorations via modular and Seiberg-Witten curves, J. Phys. A A54 (2021) 7, https://doi.org/10.1088/1751-8121/abbc4f, arXiv:2004.05218 [hep-th].

[17] Y.H. He, K.H. Lee, T. Oliver, Machine-learning the Sato–Tate conjecture, arXiv:2010.01213 [math.NT].

[18] J. Bao, S. Franco, Y.H. He, E. Hirst, G. Musiker, Y. Xiao, Quiver mutations, Seiberg duality and machine learning, Phys. Rev. D, arXiv:2006.10783 [hep-th].

[19] Y. Gal, V. Jejjala, D.K. Mayorga Pena, C. Mishra, Baryons from mesons: a machine learning perspective, arXiv:2003.10445 [hep-ph].

[20] V. Jejjala, A. Kar, O. Parrikar, Deep learning the hyperbolic volume of a knot, Phys. Lett. B 799 (2019) 135033, arXiv:1902.05547 [hep-th];
S. Gukov, J. Halverson, F. Ruehle, P. Sułkowski, Learning to unknot, arXiv:2010.16263 [math.GT].

[21] R. Deen, Y.H. He, S.J. Lee, A. Lukas, ML string standard models, arXiv:2003.13339 [hep-th].

[22] J. Halverson, B. Nelson, F. Ruehle, Branes with brains: exploring string vacua with deep RL, J. High Energy Phys. 06 (2019) 003, arXiv:1903.11616 [hep-th].

[23] J. Halverson, C. Long, Statistical predictions in string theory and deep generative models, Fortschr. Phys. 68 (5) (2020) 2000005, arXiv:2001.00555 [hep-th].

[24] Y.H. He, S.T. Yau, Graph Laplacians, Riemannian manifolds and their machine-learning, arXiv:2006.16619 [math.CO].

[25] T. Akutagawa, K. Hashimoto, T. Sumimoto, Phys. Rev. D 102 (2) (2020) 026020, arXiv:2005.02636 [hep-th].

[26] E.d. Koch, R. de Mello Koch, L. Cheng, Is deep learning a renormalization group flow?, arXiv:1906.05212 [cs.LG].

[27] J. Halverson, A. Maiti, K. Stoner, Neural networks and quantum field theory, arXiv:2008.08601 [cs.LG].

[28] S. Krippendorf, M. Syvaeri, Detecting symmetries with neural networks, arXiv:2003.13679 [physics.comp-ph].

[29] H.Y. Chen, Y.H. He, S. Lal, M.Z. Zaz, Machine learning etudes in conformal field theories, arXiv:2006.16114 [hep-th].

[30] F. Chollet, Keras, Github repository, https://github.com/fchollet/keras, 2015.

[31] B.M. Matthews, Comparison of the predicted and observed secondary structure of T4 phage lysozyme, Biochim. Biophys. Acta, Protein Struct. 405 (2) (1975) 442–451.

[32] Wolfram Research, Inc., Mathematica version 12.1, Champaign, Il, 2020, https://www.wolfram.com/mathematica.