OPEN ACCESS
IOP Publishing | Songshan Lake Materials Laboratory
AI FOR SCIENCE
AI Sci. 1 (2025) 025006 (12pp)
https://doi.org/10.1088/3050-287X/ae1d98

Learning to be simple

Yang-Hui He[1,2,*], Vishnu Jejjala[3], Mishra Challenger[4] and Em Sharnoff[5,6]

[1] London Institute for Mathematical Sciences, Royal Institution, London W1S 4BS, United Kingdom
[2] Merton College, University of Oxford, Oxford, OX1 4JD, United Kingdom
[3] Mandelstam Institute for Theoretical Physics, School of Physics, NITheCS, and CoE-MaSS, University of the Witwatersrand, Johannesburg, South Africa
[4] Department of Computer Science & Technology, Cambridge, CB3 0FD, United Kingdom
[5] Department of Computer Science, Oxford, OX1 3QG, United Kingdom
[6] Christ Church, University of Oxford, Oxford, OX1 1DP, United Kingdom

E-mail: hey@maths.ox.ac.uk

Received 5 June 2025, revised 5 November 2025
Accepted for publication 7 November 2025
Published 20 November 2025

Abstract

In this work we employ machine learning to understand structured mathematical data involving finite groups and derive a theorem about necessary properties of generators of finite simple groups. We create a database of all two-generated subgroups of the symmetric group on n-objects and conduct a classification of finite simple groups among them using shallow feed-forward neural networks. We show that this neural network classifier can decipher the property of simplicity with varying accuracies depending on the features. Our neural network model leads to a natural conjecture concerning the generators of a finite simple group. We subsequently prove this conjecture. This new toy theorem comments on the necessary properties of generators of finite simple groups. We show this explicitly for a class of sporadic groups for which the result holds. Our work further makes the case for a machine motivated study of algebraic structures in pure mathematics and highlights the possibility of generating new conjectures and theorems in mathematics with the aid of machine learning.

Keywords: AI for mathematics, simple finite groups, AI-assisted conjectures

(Some figures may appear in colour only in the online journal)

1. Introduction: machine learning and symmetries

Machine learning is an increasingly ubiquitous tool for studying a wide range of problems from self-driving cars and drug design to many electron systems in quantum chemistry and protein folding *in vivo*. However, thus far machine learning has played a smaller role in developing pure mathematics. Since the injection of machine learning into investigations of algebraic geometry in the context of theoretical physics [1–8], there has been an explosion of activity to machine learn various aspects of the topology and geometry of Calabi–Yau manifolds [9–20], algebra [21–24], knot theory [25–30], combinatorics [31, 32], and number theory [33–35], etc. The present work has a two-fold purpose. We investigate the following questions: (i) can one learn the structure of mathematics and let artificial intelligence help the intuition of a mathematician along [36, 37]? (ii) How does one develop machine learning architectures that can identify structures in mathematical datasets which are difficult to observe with the human eye? The aim is to develop new machine driven methodologies and architectures that can study such *synthetic* data, as opposed to *real world* data.

* Author to whom any correspondence should be addressed.

Original Content from this work may be used under the terms of the Creative Commons Attribution 4.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.

© 2025 The Author(s). Published by IOP Publishing Ltd on behalf of the Songshan Lake Materials Laboratory

One fundamental algebraic structure to examine through the lens of machine learning is a group. Groups provide a mathematical description of the symmetries of a system and are a guiding principle in our descriptions of Nature. Noether's theorems [38] establish that conserved quantities arise from the symmetries of a theory. For example, the conservation of energy is the consequence of the translational invariance of a system in time, the conservation of momentum is the consequence of the translational invariance of a system in space, and the conservation of angular momentum is the consequence of the invariance of a system under spatial rotations. Similarly, conserved currents in electromagnetism originate from the $U(1)$ gauge symmetry of quantum electrodynamics. The particles in the Standard Model are organized according to how they transform, namely in representations of certain Lie groups associated to the gauge symmetries. Identifying the underlying symmetries of a system and assessing their meaning is of paramount importance in understanding the physics [39, 40].

The analogous argument can be made for many machine learning endeavors. Knowledge of the symmetries of a dataset, real world or otherwise, is central to identifying correlations within the data. Indeed, making machine learning inferences using datasets is made more tractable by incorporating known invariances of the dataset into the models *ab initio*. This could, for instance, be accomplished by embedding the invariances into the architecture of a neural network [41, 42], or in the kernel of a Gaussian process. This way, various seemingly disconnected parts of the parameter space at play, albeit connected by these invariances, inform each other. This has the practical advantage of reducing computational costs, improving generalization, and has led to many real world applications. The abiding principle is that a cat is a cat regardless of how it is viewed and this equivalence should be in built where possible. These applications exploit the relationship between model invariances and the dynamics of optimization, leading to improved generalization.

Preliminary studies of machine learning the algebraic structure of groups and rings were initiated for finite groups [21, 36] and for Lie groups [43]. For specific algebraic structures, the reader is also referred to [13, 22–24, 31, 44, 45]. One key motivation of our present work is to advance these investigations to a deeper level. With a view towards building a machine driven detector of algebraic structures (and groups in particular), we ask if *interpretable* neural networks can study different properties of a group. Such properties can range from the order of a group (or group element), to more involved computations such as the invariant ring of a group. In this work, we concern ourselves with finite simple groups.

In section 2, we review some general results on finite simple groups, and describe the subclass of groups we study in this paper, i.e. two generated subgroups of the symmetric group S_n with examples. We also explain motivations behind our various representations for these groups, and their limitations. In section 3, we present our machine learning outcomes as well as a proposition that we were able to extract from our the machine learning investigations. Finally, we conclude with the discussion in section 4. Appendices A.1 and A.2 describe details of the datasets of groups we employ for our machine learning endeavors, while appendix A.3 presents the neural network architectures.

2. Machine learning simplicity

Recall that a group is *simple* if it does not admit any non-trivial normal subgroups. Following decades of effort, the finite simple groups are completely classified: they are cyclic groups of prime order $\mathbb{Z}_p$, or alternating groups A_n with $n>4$, or belong to one of 16 infinite families of groups of Lie type (plus the related Tits group), or are one of 26 exceptional cases called sporadic groups, the largest of which is the Fischer–Griess Monster, the source of moonshine [46]. Indeed, the classification of the finite simple groups initially relied on computational technology to establish the existence and uniqueness of certain sporadic groups.

In [21], relatively shallow neural networks as well as support vector machines were used to distinguish simple groups from non-simple one to 99% accuracy without the AI knowing anything about the usual techniques; this beckoned the question as to whether there is underlying new mathematics and constituted a motivation for the present study. In particular, the Cayley multiplication tables of all finite groups up to size 70 (there are 602) were taken. Next, random permutations were performed on each (since Cayley tables are only defined up to permutations) such that more permutations are included for the simple groups (since there are many more finite groups that are non-simple). This created a *balanced* database of 60 000 examples of 70×70 matrices (a group of size n would have all entries in $\{1,2,\ldots,n\}$ and all tables are padded with 0 where necessary) labeled as 'simple' or 'non-simple', with 50% each. Remarkably, when flattened and represented as points in $\mathbb{R}^{70^2}$, a support vector machine with Gaussian kernel was able to *separate* them to 99% accuracy. This led to a proto-conjecture that in the space of finite groups, the simple and non-simple groups are thus separated and can be so classified.

One shortcoming of using the Cayley table is that these grow as the square of the group order and working in $\mathbb{R}^{n^2}$ limits the computational power. In the paper, we focus on building a multi-layer perceptron model that can classify whether finite groups are simple using a much more succinct representation. This gives a two-fold advantage: (i) it will allow the exploration of more groups; and (ii) it will cross-check whether the simple/non-simple separation is truly underlying some deep mathematics and not just an artifact of Cayley tables.

Now, in the literature there are some age old results regarding the simplicity of groups. Burnside's theorem, Sylow's test, and detecting zeroes from the character table are notable examples. Such classical results are helpful in assessing a small sample of possible finite simple groups. One naïve algorithm that would determine simplicity of a group would list out all possible non-trivial subgroups of a given group

and sequentially check if any of those are normal. State of the art deterministic algorithms that test the simplicity of a finite group are computationally non-trivial even though they might incorporate known classical theorems about finite simple groups[7].

Non-deterministic algorithms are faster at establishing simplicity. In this paper, we provide an example of a non-deterministic multi-layer perceptron classifier that works in polynomial time in the size of the inputs.

Conjecture (Dixon 1969) [48]: Two randomly chosen elements of a finite simple group G generate G with probability $\to 1$ as $|G| \to \infty$.

Noting that every finite group is a subgroup of the symmetric group S_n for some n and that every simple group is two-generated [49], Dixon's conjecture motivates our study of two-generated subgroups of S_n, and consider criterion about their simplicity [48, 50]. In the experiments, groups were randomly generated by two permutations drawn at random from S_n. We use these as generators of a finite group and query a neural network classifier with the question of simplicity of the resulting group. We conduct a number of experiments with different representations. In one set of experiments, we use the full permutation representation of both the generators (which are required to be unequal to each other and identity). In the second set of experiments, we only use traces and determinants of generators (as they are representation invariant quantities). In a final experiment we only use the orders of the group elements and the order of the group as features, which are the inputs to the neural network. This is motivated by theorem 1 (which we describe in section 3.2), which says that finite simple groups can be characterized by these integers. Further details about the experiments are in appendix A.

3. Learning outcomes and a conjecture

3.1. Machine learning experiments

Experiment 1

Our first experiment takes, as input, a pair of matrices as elements of the symmetric group S_n. We mark whether the group generated by the pair as simple or non-simple accordingly. All results described here were generated via the cross-validation process described in appendix A.2. Figure 1 shows the individual average validation accuracies at the end of training for each portion of dataset for n. For $n = 8$, due to computational restrictions from such a large dataset (over 10^8 elements), a single cross-validation experiment was done, at 4.6%, which achieved an accuracy of 75%.

In general, there seem to be two different patterns in the validation accuracies above: for $n = 5$ and 6, the validation accuracy appears to increase in a roughly linear fashion with the percent of the dataset given, ending between 85% and 95% accuracy. For $n = 7$, however—and this effect is still visible with $n = 8$ to some extent—the validation accuracy jumps up to 82% at only 5% of the full dataset. There is a slight decrease as the percent of dataset increases for $n = 7$ from 60% to 80%; this is likely due to the variability in the final accuracy for cross-validation runs at $n = 7$. We will discuss this further shortly.

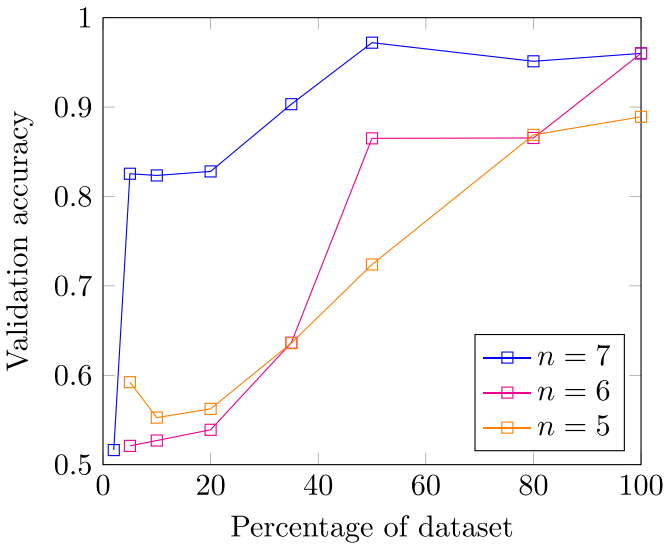


Figure 1. Validation accuracy for varied amount of datasets for each n. Percentages are of the total dataset, after balancing but before splitting to allow k-fold cross validation.

Putting these patterns aside, the models appear to become highly accurate when given greater portions of the dataset—finishing at 89%, 96%, and 96% validation accuracy for $n = 5$, 6, and 7 respectively. Excluding an outlier in the final result for $n = 7$ gives an average of 99% validation accuracy on the full dataset. Figure 2 displays the models' loss and accuracy on the training set during training, for each cross-validation run combined. The curves for $n = 5$ fit with the typically expected images; the other two sets merit further comments.

A particular effect discovered in the training runs for $n = 6$ is that the model appears to take some time to 'get off the ground' — i.e. some amount of training with slow progress is required before learning can accelerate. This effect was equally visible in the validation loss and accuracy. Multiple initializers were tried as replacements in an attempt to mitigate this effect, but it did not improve. Once training accelerated, the models appeared to behave normally.

Because epochs count the number of times the entire dataset has been used for training, there were also different requirements for each n in the number of epochs run for. For example, $n = 7$ only required six epochs, which partially explains why the per-epoch variability is more visible. Also present in $n = 7$ were occasional regressions during training, where one of the cross-validation runs would suddenly

[7] One method is to compute the character table and spotting the positions of 0s. We grant that these are polynomial complexity [47], but our main motivation is to uncover new structures in simple groups using AI rather than to find faster algorithms in their detection.

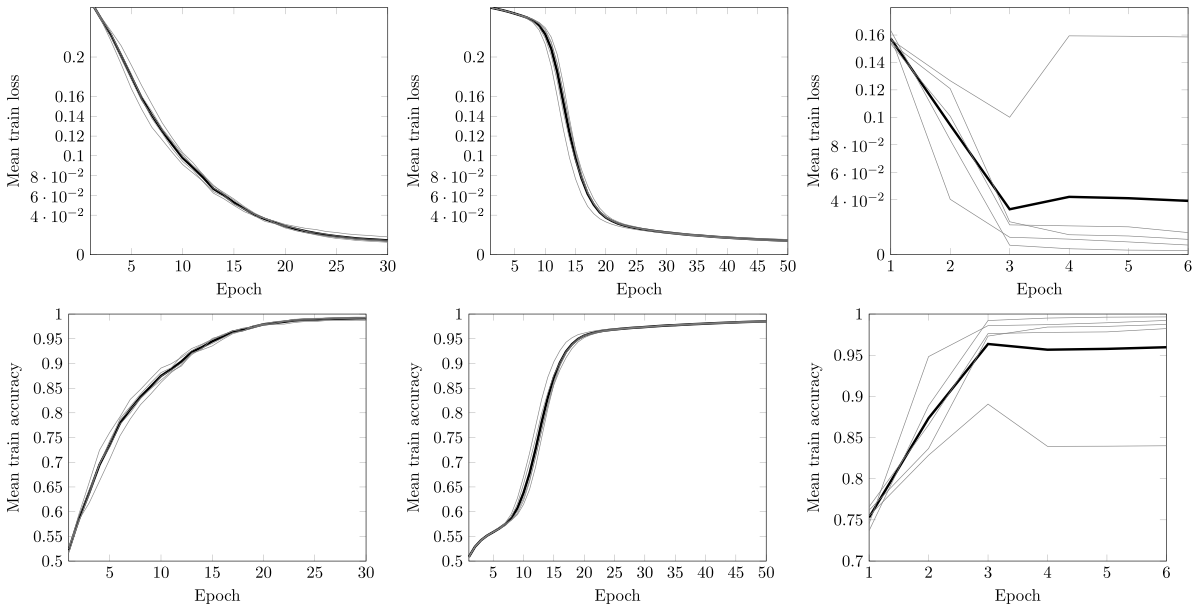


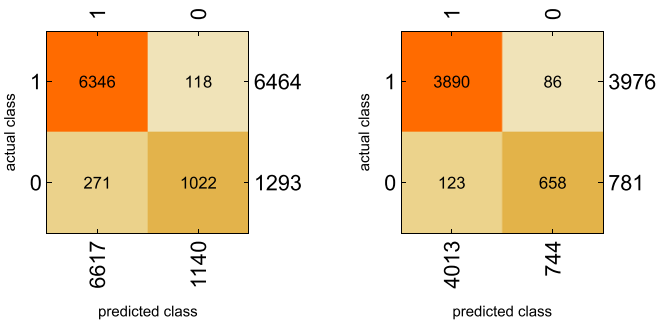
Figure 2. Loss and accuracy on training data while training on full datasets, for $n \in \{5,6,7\}$. Bold curves give the average value at that epoch across all cross-validation runs.

increase in loss and decrease in both training and validation accuracy. This consistently affected one or two cross-validation runs, entirely at random; even though the individual datasets used for each cross-validation run were kept consistent across repeated experimentation, the particular 'failing' runs were not consistent. In the particular case shown in figure 2, the 'failing' run finished with a heavy bias for negative results (i.e. indicating that a group is not simple). The class accuracies for that trial were 98% and 4%, respectively. We were unable to identify a cause, though it happens to be simply to manually identify when training a model, given knowledge of the previous behavior. There were insufficient data to determine whether this effect would occur at $n = 8$.

In order to gain further insights into understanding simplicity of finite groups, we now describe further machine learning experiments we conducted using alternate features.

Experiment 2

In this experiment we choose the features (inputs to the network) to be the traces and determinants of the two generators and the binary property of the group being Abelian. We used a multi-layer perceptron model with three hidden layers with 1000, 500, and 200 nodes respectively, and logistic sigmoid activation. We used an ADAM optimizer. The dataset of size ≈ 8500 was split into a training set of sizes size 1000, 4000. The remaining data was used for validation in both cases. The results are described in figure 3. Notably, the class accuracies on the training set is $\approx 97\%$ and the model generalizes to unseen test data with a high class accuracy of $\approx 85\%$. This seems to indicate that with only the traces and determinants as inputs to the network, as opposed to the entire generator description, the learning outcomes (class accuracies) of this Experiment is somewhat similar to that of Experiment 1. This



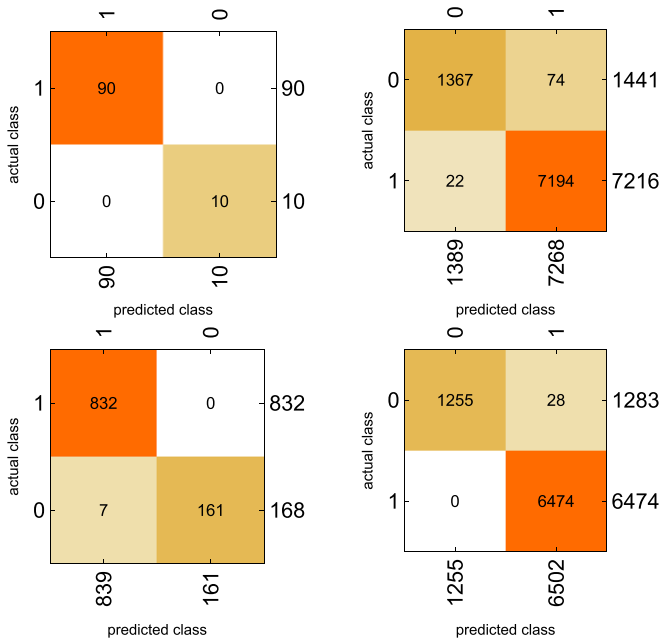
(L) Training performance; (R) Test performance

Figure 3. A neural network classifier predicts simplicity of a group based on orders and traces of group generators and the property of being Abelian. The figure shows a confusion plot where the class 0 stands for simple groups and 1 for nonsimple groups. Using a neural network model and 1000 training points, class accuracies were found to be $\approx 97\%$ and 85%, respectively, on the training set (left) and $\approx 97\%$ and 86% on the test set (right). We use 4000 points for training. We obtain similar results at a smaller dataset size of 1000.

seems to strongly indicate that these features could be important in determining the eligibility of a permutation matrix as a simple group's generator. We will also note further down in theorem 1 that a finite simple group is completely characterized by the order of group elements and the size of the group. This indicates that identifying a pertinent condition on generators for simplicity can be experimentally demonstrable as a favorable learning outcome by a simple learning model such as a neural network.

Experiment 3

In this experiment we chose the features to be the orders of the group elements and the order of the group. We used a



(L) Training performance; (R) Test performance

Figure 4. A classifier predicts simplicity of a group based on orders of group elements and order of the group. The plot shows a confusion plot where the class 0 stands for simple groups and 1 for nonsimple groups. Using a neural network model when using only 100 training points (top) and 1000 training points (bottom), class accuracies on the test set were found to be $\approx 99\%$.

multi-layer perceptron model with three hidden layers with 1000, 500, and 200 nodes respectively, and logistic sigmoid activation. We used an ADAM optimizer. The dataset of size $\sim$8500 was split into a training set of size 1000, and a test set of $\sim$7500. We repeated the experiment using a smaller architecture with 10 times fewer nodes and a training set of size 100. The results are described in figure 4.

The second and third experiments were particularly illuminating in terms of choice of features. The new features in these experiments were the traces and determinants of the generators as opposed to the full generators (Experiment 2); and the group order and the property of being Abelian (Experiment 3).

With these outcomes, it is interesting to consider a mathematical statement that might capture these experiments. We therefore consider necessary conditions on traces and determinants of generators of finite simple groups; hinted by our machine learning outcomes above. This manner of investigation has yielded different conjectures using an alternative approach [51].

3.2. A machine guided mathematical conjecture

In experiments 1–3 of section 3.1 we have built simple feed-forward multi-layer perceptron models to predict the simplicity of a two-generated group given merely the generators, or their properties. Our learning outcomes were modest with predictive accuracies $\sim$80%. In a traditional approach, one can conceive of a straightforward methodology for resolving the question of the simplicity of the resulting group. This could be seen to involve two key steps, the first of which is to freely generate a group from the two generators. Following this, one could list all non-trivial subgroups of this group, and check for normal subgroups. Both these key steps are computationally intensive. Therefore, it is impressive that a simple feed-forward multi-layer perceptron model should address this question to any reasonable degree of accuracy. We now ask the following question: can similar learning outcomes be attained using alternate representations for the generators? These could involve high level properties of the generators given a representation—traces and determinants are examples of such properties. If a similar or better learning outcome is obtained using the above properties as alternate features, this is perhaps indicative of an underlying mathematical relationship between such generator properties and the property of simplicity. This indeed turns out to be the case: figures 3 and 4 show learning outcomes using different features. In this section, we present such a mathematical relationship as a conjecture motivated by the above learning outcome, for which we then provide a proof. This AI-guided mathematical conjecture formulation is very much in the spirit of [6, 37, 51].

First, we recall a theorem for characterizing finite simple groups [52]. Let us define the set of orders of group elements as

$$\pi(G) := \{\mathrm{Ord}(g) : g \in G\} \ . \tag{1}$$

Theorem 1. *A finite simple group G is completely characterized by the order of group elements $\pi(G)$ and the order (size) of the group G.*

Theorem 1 implies that for G and H simple finite groups, $G \sim eq H$ iff $|G| = |H|$ and $\pi(G) = \pi(H)$ (as sets). This is indicative that perhaps determinants of the generator could play a crucial role in discovering a new mathematical relationship between the generators and the property of simplicity. In fact, this also provides an alternate representation for our classification problem. Computing $\pi(G)$ for a group two-generated G requires further computation. When our ML models are trained using these alternate features, it is not surprising that the learning outcomes are nearly perfect. Figure 4 shows the learning outcome. However, this is at additional computational cost, since generating such representations involves computing $\pi(G)$ and computing $|G|$ requires.

Furthermore, we make an observation from studying our dataset. Before doing this, we introduce a preliminary notion [53]. Let G be a subgroup of a symmetric group acting as permutation of a set Ω, as is with the case with all groups we consider here. We recall the standard definitions that for an element $\alpha \in \Omega$, the stabilizer G_α of α is the set of all group elements that fix α. On the contrary, the set of all fixed points

for a group element $x \in G$ is denoted $C_\Omega(x)$. In short,

$$G_\alpha := \{x \in G : x(\alpha) = \alpha\} \ , \qquad (2)$$
$$C_\Omega(x) := \{\alpha \in \Omega : x(\alpha) = \alpha\} \ .$$

Then,

Definition 1. The fixed point ratio of $x \in G$, denoted by $\mathrm{fpr}(x)$ (which is of course implicitly dependent upon Ω), is the proportion of points in Ω fixed by x, i.e.

$$\mathrm{fpr}(x) = \frac{|C_\Omega(x)|}{|\Omega|} \ .$$

For our dataset of all the two-generated subgroups of S_n up to $n \leqslant 10$, we notice that whenever a resulting group is simple, the number of fixed points of either of its two generators is never $n-2$ or $n-4$ for $n \geqslant 5$. Now, all our group elements are $n \times n$ permutation matrices (in particular matrices with only 0 and 1) acting on $\{1,2,\ldots,n\}$. Hence, the number of fixed points is counted by the number of 1 s on the diagonal. Hence, this observation hints towards the existence of a conjecture regarding traces (or equivalently, number of fixed points) and signs of permutations.

Proposition 1. (conjectured from machine learning). *Consider two-generated subgroups $\mathbb{H} \subset S_n$ ($n \geqslant 5$) with distinct non-trivial generators in the permutation representation. That is, consider $\sigma_{i=1,2}(\neq e) \in S_n$ as $n \times n$ permutation matrices. If $\mathbb{H}$ is a simple group, then*

1. $\det(\sigma_i) = 1$, *and,*
2. $\mathrm{tr}(\sigma_i) \in \{1,2,\ldots,n\} \setminus \{n-4, n-2, n-1, n\}$.

In particular, if $\mathrm{tr}(\sigma_i) = n-4$, then $\mathbb{H} = \mathbb{D}_{2n} \subseteq \mathbb{A}_n \lhd \mathbb{S}_n, n \geqslant 5$. That is $\mathbb{H}$ is the dihedral group, and thus not simple.

We can now prove this conjecture which was guided by our machine learning experiments.

Proof. First, since $\mathbb{H}$ is a subgroup of S_n as represented by permutation matrices, the determinant of all group elements is equal to ± 1. Suppose, without loss of generality, $\det(\sigma_1) = -1$. It is straightforward to check that the following defines a homomorphism from $\mathbb{H}$ to $(\mathbb{Z}/(2\mathbb{Z}), +\mathrm{mod}\,2)$.

$$\psi : \mathbb{H} \longrightarrow \{0,\ 1\}$$
$$\psi(\sigma) = \begin{cases} 0, \text{ if } \det(\sigma) = 1 \ , \\ 1, \text{ if } \det(\sigma) = -1 \ . \end{cases}$$

By the first isomorphism theorem, $\mathbb{H}/\ker(\psi) \sim_{eq} \mathrm{im}(\psi)$, with $\ker(\psi) \lhd \mathbb{H}$. Clearly, $\psi(e) = 0$, and $\psi(\sigma_1) = 1$ (by assumption). Therefore, $\mathrm{im}(\psi) = \{0,\ 1\} = \mathbb{Z}/(2\mathbb{Z})$. As such, $\ker(\psi)$ is an index two subgroup of $\mathbb{H}$, and therefore normal. Hence, $\mathbb{H}$ is not simple. Thus the determinant of both σ_1 should by 1. This proves the first part of the proposition.

To prove the second part, let us assume $\mathbb{H}$ is simple. We begin by observing that $\mathrm{tr}(\sigma_i) \neq n$ since e is the only element that has trace n and $\sigma_i \neq e$ by assumption. Progressing further, $\mathrm{tr}(\sigma_i) \neq n-1$ as that would require a single 0 and 1s everywhere else on a diagonal, which is a singular matrix. Finally, we in fact have that $\mathrm{tr}(\sigma_i) \neq n-2$ since then $\det(\sigma_i) = -1$, coming from the $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ block.

Now consider the case of $\mathrm{tr}(\sigma_i) = n-4$ and $\det(\sigma_i) = 1$. We will show that $\sigma_i^2 = e$. If this were to be the case, $\mathbb{H}$ would be a group generated by involutions. This would imply that $\mathbb{H}$ is the dihedral group of order $2m$, $\mathbb{D}_{2m}$ (where for $m = 1, 2$ they are, $\mathbb{Z}/(2\mathbb{Z})$ and $\mathbb{Z}/(2\mathbb{Z}) \times \mathbb{Z}/(2\mathbb{Z})$). In any case, $\mathbb{H}$ would not simple (except for the trivial case of $m = 1$, which we excluded by having $n \geqslant 5$).

To finish the proof, it remains show that

$$P(n) := \big((\det(\sigma) = 1) \wedge (\mathrm{tr}(\sigma) = n-4) \implies \sigma_i = \sigma^T\big)$$

holds. We do this by induction. We can enumerate to see that $P(4)$ holds. Now assume $P(n)$ holds. Let $\widehat{\sigma} \in \mathbb{A}_{n+1}$, so that $\det(\widehat{\sigma}) = 1$ (by definition of the alternating group), and $\mathrm{tr}(\widehat{\sigma}) = (n+1) - 4$. We must show that $P(n+1)$ holds by showing $\widehat{\sigma} = \widehat{\sigma}^T$. Note that each $\widehat{\sigma}$ can be obtained from a σ with the introduction of a 1 in the diagonal at one of $(n+1)$ possible positions (reflecting the fact that $|\mathbb{A}_{n+1}| = (n+1)\,|\mathbb{A}_n|$). Note that this does not alter the determinant but increases the trace by 1. It therefore follows that $\widehat{\sigma} = \widehat{\sigma}^T$ since $\sigma = \sigma^T$, establishing that $P(n+1)$ holds, completing the proof. □

The following corollary is a restatement of the above written in terms of fixed point ratios.

Corollary 1. *Let $\mathbb{H}$ be a finite simple group with the generating set containing only two distinct group elements in a permutation representation of degree n. Then, the fixed point ratio (*fpr*) of any such generator cannot equal $2^i/n$, $\forall i \in \{0,1,2\}$.*

Fixed point ratios quantify the proportion of points fixed by an element in a permutation action, and sharp bounds have been instrumental in results on random generation, base sizes, and derangements [53]. Our trace/determinant criterion is analogous in spirit, imposing structural restrictions that reduce the space of potential generators. A similar discovery process may inform new computational approaches to generation problems.

Table 1. Properties of the generators of the Janko Group J_1. We use a 266 dimensional permutation representation for the group. The two group generators (permutation actions) g_1 and g_2 are presented in the appendix A.4. Both the generators are of positive signature and the number of fixed points is never $n - 2^k$, $k \in \{0,1,2\}$, ala proposition 1, with $n = 266$ for J_1.

Sporadic group	generator	trace	sign
J_1	g_1	10	1
	g_2	5	1

4. Discussion

In this work, we demonstrate that standard off the shelf machine learning tools such as neural networks can help

Table 2. Properties of Mathieu Group generators. All the generators are of positive signature and the number of fixed points is never $n-2^k$, $k\in\{0,1,2\}$, ala proposition 1.

Sporadic group	generators	trace	$\{n-2^k\}_{k=0}^2$
M_9	(1,4,9,8)(2,5,3,6), (1,6,5,2)(3,7,9,8)	1, 1	5, 7, 8
M_{10}	(1, 9, 6, 7, 5)(2,10, 3, 8, 4), (1,10, 7, 8)(2, 9, 4, 6)	0, 2	6,8,9
M_{11}	(1, 2, 3, 4, 5, 6, 7, 8, 9,10,11), (3,7,11,8)(4,10,5,6)	0, 3	7,9,10
M_{12}	(1,2,3,4,5,6,7,8,9,10,11), (3,7,11,8)(4,10,5,6), (1,12)(2,11)(3,6)(4,8)(5,9)(7,10)	1, 4, 0	8,10,11
M_{21}	(1, 4, 5, 9, 3)(2, 8,10, 7, 6) (12,15,16,20,14)(13,19,21,18,17), (1,21, 5,12,20)(2,16, 3, 4,17) (6,18, 7,19,15)(8,13, 9,14,11)	0, 3	17,19,20
M_{22}	(1, 2, 3, 4, 5, 6, 7, 8, 9,10,11) (12,13,14,15,16,17,18,19,20,21,22), (1, 4, 5, 9, 3)(2, 8,10, 7, 6) (12,15,16,20,14)(13,19,21,18,17), (1,21)(2,10, 8, 6)(3,13, 4,17) (5,19, 9,18)(11,22)(12,14,16,20)	1, 1, 1	18,20,21
M_{23}	(1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12, 13,14,15,16,17,18,19,20,21,22,23), (3,17,10, 7, 9)(4,13,14,19, 5) (8,18,11,12,23)(15,20,22,21,16)	1, 1	19, 21,22
M_{24}	(1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13, 14,15,16,17,18,19,20,21,22,23), (3,17,10, 7, 9)(4,13,14,19, 5) (8,18,11,12,23)(15,20,22,21,16), (1,24)(2,23)(3,12)(4,16) (5,18)(6,10)(7,20)(8,14) (9,21)(11,17)(13,22)(15,19)	1, 4, 0	20, 22, 23

determine the simplicity of a group with relatively high accuracies and produce novel insights. We conduct a number of machine learning experiments with different mathematical features to determine simplicity. When using permutation representations of group generators, we find modest learning outcomes. In another experiment, using element orders as features alongside the order of the group, we demonstrate remarkable class accuracies of close to 99% in figure 4, reflective of a known result for finite simple groups, *viz.*, theorem 1. The success our experiments gives further confidence that the ability of machine-learning (in particular support vector machines) to distinguish simple/non-simple groups when trained on Cayley tables [21] was not a mere artifact of data representation, but truly underlies interesting mathematics. From these experiments, we distill proposition 1, which we then prove using properties of finite simple groups. The result restricts choices of two-generating sets for finite simple groups.

The pipeline for generating this result was a multi-step process of (1) conjecture generation (2) ML experimentation and (3) theorem proving. The conjecture formulation step was itself guided by the learning outcomes in the neural network experiments using the features hypothesized by the conjecture. Future experiments may involve standard interpretability tools such as relevance scores, similar to [37, 54]. We noted that the learning outcomes using the traces and determinants were favorable, and the structure of theorem 1 was indicative of the hypothesis. As such our conjecture was guided by a combination of classical results and ML experimentation. We then prove the conjecture using known results. We recast this result in terms of fixed point ratios of group generators in corollary 1. Bounds on fixed point ratios are an important topic of consideration. The above observation feeds directly into studies of fixed point ratios of groups, which have been studied extensively over many decades [53]. The result places restrictions on the traces of generators of any finite simple group. The computational advantage from this observation is $\approx 10^{-4}\%$ when looking for a generator of the Monster group in the smallest faithful irreducible representation (which occurs at $n = 196\,883$). In absolute terms this is a very small reduction. While the computational advantage is small now, the fact that one can prove such constraints (via ML guidance) is of interest. A more specific interesting point is that the determinant and trace of the generators conspire in some cases to form a dihedral group and therefore can never be simple.

Corollary 1 holds for all finite simple groups in the permutation representation. As such it holds for sporadic groups in their permutation representations. In tables 1 and 2, we check consistency of our results for the Mathieu group and the Janko

group J_1. For this, we note that the traces and determinants of known generators of these sporadic groups are consistent with the allowed values in corollary 1. We can as well study other presentations of finite groups, and we leave this to future work.

The prospect of finding novel mathematical results is tantalizing in the age of AI. There are a number of emerging pathways for mathematical research in light of machine learning. In this work, we exploit one such pathway which involves learning a mathematical property (in this case simplicity) using standard machine learning architectures, and exploiting insights from the learning process and known results to distill a theorem. Often, tools from machine interpretability such as relevance scores can aid in this process resulting in new mathematical insights or theorems [6, 28, 36, 37]. Complementary approaches have been proposed in the recent literature, relying on an organizational principle for mathematical statements, resulting in novel conjectures which could often be proved using domain expertise [51]. We emphasize that the novelty here lies less in the technical difficulty of the proof and more in the route by which the conjecture was uncovered. Once formulated, the statement could indeed be proved using standard techniques, but the formulation itself emerged through machine-learning experimentation with a variety of input features, and known classical results and proof technique. This highlights the value of ML guided mathematical discovery.

Recent proposals such as the Birch test [55] highlight criteria for when AI should be credited with autonomous discovery; while our contribution does not meet that standard, it illustrates how collaborative workflows can still yield genuinely new and nontrivial mathematics. In this sense, the work points toward a broader research programme: leveraging ML not to replace classical reasoning, but to expand the landscape of conjectures that mathematicians can explore and establish.

Table 3. Examples of permutation generators of some small groups.

Group Id	Name	Simple?	Generators	
[1, 1]	1	No	$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$	$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$
[2, 1]	$\mathbb{Z}_2$	Yes	$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$	$\begin{bmatrix}0&1&0&0\\1&0&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$
[3, 1]	$\mathbb{Z}_3$	Yes	$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$	$\begin{bmatrix}0&0&1&0\\1&0&0&0\\0&1&0&0\\0&0&0&1\end{bmatrix}$
[4, 1]	$\mathbb{Z}_4$	No	$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$	$\begin{bmatrix}0&0&0&1\\1&0&0&0\\0&1&0&0\\0&0&1&0\end{bmatrix}$
[6, 1]	S_3	No	$\begin{bmatrix}0&1&0&0\\1&0&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$	$\begin{bmatrix}0&0&1&0\\1&0&0&0\\0&1&0&0\\0&0&0&1\end{bmatrix}$
[24, 12]	S_4	No	$\begin{bmatrix}0&1&0&0\\1&0&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$	$\begin{bmatrix}0&0&0&1\\1&0&0&0\\0&1&0&0\\0&0&1&0\end{bmatrix}$
[8, 3]	$\mathbb{D}_8$	No	$\begin{bmatrix}0&1&0&0\\1&0&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$	$\begin{bmatrix}0&0&1&0\\0&0&0&1\\1&0&0&0\\0&1&0&0\end{bmatrix}$
[4, 2]	$\mathbb{Z}_2\times\mathbb{Z}_2$	No	$\begin{bmatrix}0&1&0&0\\1&0&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$	$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}$
[12, 3]	A_4	No	$\begin{bmatrix}0&0&1&0\\1&0&0&0\\0&1&0&0\\0&0&0&1\end{bmatrix}$	$\begin{bmatrix}0&0&0&1\\0&1&0&0\\1&0&0&0\\0&0&1&0\end{bmatrix}$

Acknowledgments

Y H H would like to thank STFC for Grant ST/J00037X/2, the Leverhulme Trust for a project grant, as well as Joseph Chuang and Radha Kessar for many helpful discussions. V J is supported by the South African Research Chairs Initiative of the Department of Science and Innovation and the National Research Foundation. C M is supported by the Accelerate Programme for Scientific Discovery, at the Computer Laboratory, University of Cambridge. C M would like to thank Damián Kaloni Mayorga Peñã, Aditya Ravuri, Subhayan Roy Moulik for helpful discussions. The authors would like to thank the Isaac Newton Institute for Mathematical Sciences for support and hospitality during the program 'Black holes: bridges between number theory and holographic quantum information' when work on this paper was undertaken; this work was supported by EPSRC Grant Number EP/R014604/1.

Appendix A. Machine learning experiments

There were therefore many entries in the dataset for groups equivalent under isomorphism, and the number of entries per-group was not changed to be less imbalanced. This was intentional. To give a sense of scale, the number of subgroups of S_n for $n = [4, 7]$ are 11, 19, 56, 96. The number of entries generated by our methods for each S_n would be 560, 14375, 518 364, 25 401 551.

A.1. Two-generated subgroups of S_n

The input data to our models were the permutation matrices corresponding to the generators of each group. We found this to be the most effective representation of the data, as it allowed

Table 4. Subgroups of S_3.

Group Id	Name	Simple?	Generators
[1, 1]	1	No	$\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}$ $\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}$
[2, 1]	$\mathbb{Z}_2$	Yes	$\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}$ $\begin{bmatrix}0&1&0\\1&0&0\\0&0&1\end{bmatrix}$
[3, 1]	$\mathbb{Z}_3$	Yes	$\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}$ $\begin{bmatrix}0&0&1\\1&0&0\\0&1&0\end{bmatrix}$
[6, 1]	S_3	No	$\begin{bmatrix}0&1&0\\1&0&0\\0&0&1\end{bmatrix}$ $\begin{bmatrix}0&0&1\\1&0&0\\0&1&0\end{bmatrix}$

Table 5. Number of each subgroup of S_4 that would be present in the dataset.

Group Id	Name	Simple?	Count (filtered)	Count (unfiltered)
[2, 1]	C2	Yes	18	27
[3, 1]	C3	Yes	24	32
[4, 1]	C4	No	30	36
[4, 2]	C2 × C2	No	24	24
[6, 1]	S3	No	72	72
[8, 3]	D8	No	72	72
[12, 3]	A4	No	96	96
[24, 12]	S4	No	216	216

for fixed-size inputs and produced better performing models than encoding the generators as permutation vectors.

Example: two-generated subgroups of S_3

As illustration, we tablulate the two-generated subgroups of S_3 in table 4. Moreover, in table 3, we give some small groups in table 3 and their permutation structure. The ID are given in GAP notation. Model training and validation were performed on subsets of the *balanced* datasets for five-fold cross-validation. The mean final validation accuracy for models trained on different portions of the balanced dataset for each n is displayed in figure 1, from 'Results'.

A.2. Datasets

The datasets are given by the generating permutation matrices for each subgroup of some S_n. Only the two-generated subgroups were selected, but all permutations were included, giving a total $(n!)^2 - n^2$ individual entries in each dataset. The orders of the groups in these datasets were therefore not evenly distributed; the number of each group included in the dataset for S_4 and S_5 is in tables 5 and 6. It was computationally infeasible to include the same information for S_6 and larger.

For all datasets, we used the `SageMath` interface to `GAP` to analyze the groups generated by each pair of permutation matrices. The categorization of whether a group is simple was paired with the generating matrices to produce the entry for that group.

Table 6. Number of each subgroup of S_5 present in the dataset.

Group Id	Name	Simple?	Count (filtered)	Count (unfiltered)
[2, 1]	C2	Yes	50	75
[3, 1]	C3	Yes	60	80
[4, 1]	C4	No	150	180
[4, 2]	C2 × C2	No	120	120
[5, 1]	C5	Yes	120	144
[6, 2]	C6	No	220	240
[6, 1]	S3	No	360	360
[8, 3]	D8	No	360	360
[10, 1]	D10	No	360	360
[12, 3]	A4	No	480	480
[12, 4]	D12	No	360	360
[20, 3]	C5 : C4	No	1440	1440
[24, 12]	S4	No	1080	1080
[60, 5]	A5	Yes	2280	2280
[120, 34]	S5	No	6840	6840

Algorithm 1. Conversion from $k \in [0, n!)$ to a permutation of n elements.

Input: An integer $k \in [0, n!)$
Output: A permutation of the elements $0, 1, ..n$
vs := [] // **The values 0, .. n**
ps := [] // **The final permutation**
for $i \leftarrow 0$ **to** $n-1$ **do**
  vs.append(i)
end
for $i \leftarrow n$ **to** 1 **do**
  r:= remainder of $\frac{k}{i}$
  $k \leftarrow k \mod i$
  v := vs[r]
  Remove vs[r] and shift remaining elements to the left
  ps.append(v)
end
return *ps*

With most of the symmetric groups examined here, it was feasible to produce the categorization from `GAP` for all permutation pairs, which was done by simple iteration. Experiments on S_8 used a random subset of all possible permutation pairs by taking a random sample of integers 1 to $(8!)^2$, mapping these values to permutation matrices, and filtering out pairs with the same permutation. The component of this mapping that produces the permutation[8] is specified in algorithm 1.

A single sample in one of the datasets could be constructed as follows. Suppose we are generating data for the subgroups of S_4, with the permutations given by the integer pair (6, 19). The corresponding permutations, as generated by the aforementioned algorithm, are:

$$(2,1,0,3) \text{ and } (3,1,2,0) \tag{3}$$

[8] Constructing the permutation matrix from the corresponding permutation of n elements is trivial; we leave this undiscussed.

which generate the following permutation matrices, respectively:

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{ and } \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}. \tag{4}$$

The inputs for the training sample are just the concatenation of each row in the two matrices into a single vector, as shown below:

$$\langle 0,0,1,0,\ 0,1,0,0,\ 1,0,0,0,\ 0,0,0,1,\ 0,0,0,1,\\ 0,1,0,0,\ 0,0,1,0,\ 1,0,0,0\rangle. \tag{5}$$

The particular group corresponding to these inputs S_3, which is not simple.

Training datasets

The generated datasets were unbalanced; only approximately 20% of the entries correspond to simple groups in each dataset[9]. From here on, we refer only to the *balanced* datasets, which are chosen by a random selection of the non-simple entries in equal number to the set of simple ones.

All experiments were done with five-fold cross-validation, with each train/validation split taking from the same balanced subset of the unbalanced dataset. For experimentation on differing amounts of data (e.g. using 50% of the 'full', balanced dataset), all balanced datasets were subsets of the 'full' balanced dataset.

A.3. Neural network architectures

The models used for experimentation were fully-connected neural networks with the two input permutation matrices flattened into a single vector of $2n^2$ Boolean values for each S_n. Predictions are chosen by the greater of two output values, scaled by the softmax activation function. Hidden layers for all final models consist of 256 nodes, with all nodes using the ReLU activation function. See figure 5 above for an illustration.

Optimization was done with stochastic gradient descent (SGD) with Nesterov-accelerated momentum and Mean Squared Error as the loss function. The hyperparameters for SGD (both learning-rate and momentum), the number of hidden layers for each n, and the size of the hidden layers were selected by manual hyperparameter search. The results with the best mean validation accuracy across all five validation datasets at the end of training. A hidden layer size of 256 was chosen for all n as a common best size for $n=5$ and $n=6$.

For $n=7$ and $n=8$, rigorous hyperparameter search was infeasible due to the size of the datasets and the resulting computational restrictions—less so with the former than the latter.

[9] For each value of $n \in \{5,..8\}$, these fractions are 0.1758, 0.2027, 0.2315, and 0.2133, respectively.

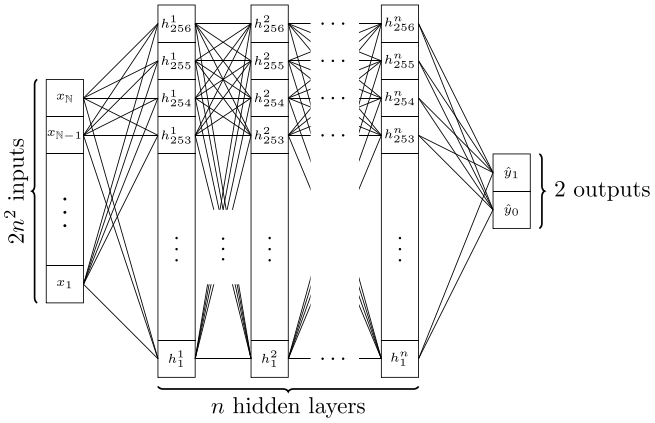


Figure 5. Generic architecture of the models used. The generating permutation matrices are flattened and provided as input. Softmax is applied to the two outputs, which give probabilities for whether the group is simple or not.

Table 7. Hyperparameters used for each dataset.

n	Learning rate	Momentum	Num. hidden layers	γ
5	0.05	0	1	0.1
6	0.001	0.01	3	0.05
7*	0.01	0.1	9	0.05
8*	0.01	0.1	9	0.05

For these datasets, the hyperparameters were initially chosen by extrapolation from those used with the lower two values for n, without attempting to produce optimal results. Adjustments were made only until the model was successfully better than random chance. This less rigorous iteration was required as, for some sets of hyperparameters, the models completely fail to perform any meaningful learning (often skewing entirely towards one output or another). This occurred most prominently at $n=7$, when testing with five or fewer hidden layers. This effect was avoided by increasing the number of hidden layers, which did not prove to be additionally necessary for $n=8$ (i.e. there was no increase from $n=7$ to 8).

In addition to the hyperparameters previously mentioned, learning rate decay was also provided, with the learning rate at each epoch calculated from the previous by:

$$\mathrm{lr}_{i+1} = (1-\gamma)\,\mathrm{lr}_i \tag{6}$$

with γ as the hyperparameter controlling the rate of decay. The hyperparameters used for each n are given above, in table 7.

A.4. Sporadic group generator

In the paper we refer to the two generators of the Janko group J_1. Using GAP 4 [56], we list a 266−dimensional permutation representation of J_1.

LoadPackage('atlasrep');; G: = SimpleGroup('J1');; gens: = GeneratorsOfGroup(G);;

g_1:=(1,17) (2,67) (3,135) (4,83) (5,115) (6,8) (7,210) (9,114) (10,214) (11,149) (12,59) (13,53) (14,119) (15,179) (16,139) (18,169) (19,50) (20,232) (21,109) (22,128) (23,202) (24,82) (25,56) (26,134) (27,196) (28,234) (29,133) (30,102) (31,256) (32,204) (33,123) (34,148) (35,248) (36,164) (37,185) (38,137) (39,177) (40,165) (41,230) (42,113) (44,93) (45,127) (46,198) (47,180) (48,197) (49,239) (51,112) (52,265) (54,88) (55,66) (57,263) (58,62) (60,89) (61,238) (63,258) (64,97) (65,219) (68,186) (69,120) (70,124) (71,242) (72,211) (73,246) (74,175) (75,92) (76,192) (77,153) (78,215) (79,162) (80,183) (81,244) (84,85) (86,132) (87,264) (90,152) (91,103) (94,220) (95,201) (96,108) (98,171) (99,228) (100,178) (101,187) (104,253) (105,141) (106,147) (107,174) (110,247) (111,184) (116,146) (117,154) (118,158) (121,226) (122,157) (125,161) (126,212) (129,155) (130,159) (131,249) (136,235) (138,218) (140,205) (142,176) (143,227) (145,189) (156,181) (160,236) (163,172) (166,217) (167,251) (170,190) (173,237) (182,262) (188,209) (191,243) (193,259) (194,240) (195,224) (199,229) (200,208) (203,255) (213,225) (216,252) (221,257) (223,254) (231,260) (233,245) (250,261).

g_2:=(1,154,190) (2,78,249) (3,139,165) (4,265,119) (5,201,258) (6,187,237) (7,99,124) (8,246,240) (9,96,180) (10,75,36) (11,141,16) (12,174,215) (13,164,58) (14,242,191) (15,140,26) (17,37,118) (18,59,170) (19,65,189) (20,162,172) (22,223,104) (23,117,45) (24,62,256) (25,152,232) (27,72,128) (28,54,196) (29,40,126) (30,238,255) (31,87,251) (32,142,106) (33,202,261) (34,193,205) (35,171,183) (38,57,254) (39,70,143) (41,123,55) (42,145,110) (43,64,134) (44,195,231) (46,56,248) (47,51,184) (48,66,79) (49,50,207) (52,100,83) (53,158,121) (60,211,88) (61,113,132) (63,98,218) (67,136,81) (68,227,213) (69,192,138) (71,175,93) (73,122,160) (74,137,148) (76,263,125) (77,108,101) (80,264,97) (82,178,163) (84,103,167) (85,262,252) (86,216,166) (89,243,244) (91,241,120) (92,235,214) (94,206,102) (95,144,173) (105,159,157) (107,188,197) (109,156,199) (111,135,253) (112,245,150) (114,247,220) (115,198,194) (116,239,177) (130,230,260) (131,161,200) (133,146,176) (147,204,168) (149,222,181) (151,236,212) (153,225,233) (155,217,226) (169,210,219) (179,182,203) (185,209,224) (186,257,234) (208,221,259) (228,229,266).

ORCID iD

Yang-Hui He 0000-0002-0787-8380

References

[1] He Y-H 2017 Deep-learning the landscape (arXiv:1706.02714)

[2] Carifio J, Halverson J, Krioukov D and Nelson B D 2017 Machine learning in the string landscape *J. High Energy Phys.* JHEP09(2017)157

[3] Krefl D and Seong R-K 2017 Machine learning of Calabi-Yau volumes *Phys. Rev.* **D96** 066014

[4] Ruehle F 2017 Evolving neural networks with genetic algorithms to study the string landscape *J. High Energy Phys.* JHEP08(2017)038

[5] He Y-H 2017 Machine-learning the string landscape *Phys. Lett. B* **774** 564–8

[6] He Y-H 2018 *The Calabi-Yau Landscape: From Geometry, to Physics, to Machine Learning* (*Lecture Notes in Mathematics* vol 5) (Springer) (https://doi.org/10.1007/978-3-030-77562-9)

[7] Ruehle F 2020 Data science applications to string theory *Phys. Rep.* **839** 1–117

[8] Jain A, Mishra C and Liò P 2022 A physics-informed search for metric solutions to ricci flow, their embeddings, and visualisation (arXiv:2212.05892)

[9] Bull K, He Y-H, Jejjala V and Mishra C 2018 Machine learning CICY threefolds *Phys. Lett. B* **785** 65–72

[10] Bull K, He Y-H, Jejjala V, and Mishra C 2019 Getting CICY high *Phys. Lett. B* **795** 700–6

[11] Berglund P, Campbell B and Jejjala V 2021 Machine learning Kreuzer-Skarke Calabi-Yau threefolds (arXiv:2112.09117)

[12] Ashmore A, He Y-H and Ovrut B A 2020 Machine learning Calabi-Yau metrics *Fortsch. Phys.* **68** 2000068

[13] Peifer D, Stillman M and Halpern-Leistner D 2020 Learning selection strategies in Buchberger's algorithm *Int. Conf. on Machine Learning* (PMLR) pp 7575–85

[14] Anderson L B, Gerdes M, Gray J, Krippendorf S, Raghuram N and Ruehle F 2020 Moduli-dependent Calabi-Yau and SU(3)-structure metrics from machine learning (arXiv:2012.04656)

[15] Douglas M R, Lakshminarasimhan S and Qi Y 2020 Numerical Calabi-Yau metrics from holomorphic networks (arXiv:2012.04797)

[16] Jejjala V, Pena D K M and Mishra C 2020 Neural network approximations for Calabi-Yau metrics (arXiv:2012.15821)

[17] Larfors M, Lukas A, Ruehle F and Schneider R 2021 Learning size and shape of Calabi-Yau spaces (arXiv:2111.01436)

[18] Ashmore A, Calmon L, He Y-H and Ovrut B A, Metrics C-Y 2021 Energy functionals and machine-learning (arXiv:2112.10872)

[19] Larfors M, Lukas A, Ruehle F and Schneider R 2022 Numerical metrics for complete intersection and Kreuzer-Skarke Calabi-Yau manifolds *Mach. Learn. Sci. Tech.* **3** 035014

[20] Berglund P, Butbaia G, Hübsch T, Jejjala V, Peña D M and Mishra C 2022 Machine learned Calabi–Yau metrics and curvature (arXiv:2211.09801)

[21] He Y-H and Kim M 2019 Learning algebraic structures: preliminary investigations (arXiv:1905.02263)

[22] Bao J, Franco S, He Y-H, Hirst E, Musiker G, Xiao Y and Mutations Q 2020 Seiberg duality and machine learning *Phys. Rev. D* **102** 086013

[23] Amorós L, Gasanova O and Jakobsson L 2021 A machine learning approach to commutative algebra: Distinguishing table vs non-table ideals (arXiv:2109.11417)

[24] Dechant P-P, He Y-H, Heyes E and Hirst E 2022 Cluster algebras: network science and machine Learning (arXiv:2203.13847)

[25] Hughes M C 2020 A neural network approach to predicting and computing knot invariants *J. Knot Theory Ramif.* **29** 2050005

[26] Jejjala V, Kar A and Parrikar O 2019 Deep learning the hyperbolic volume of a knot *Phys. Lett. B* **799** 135033

[27] Gukov S, Halverson J, Ruehle F and Sułkowski P 2020 Learning to unknot (arXiv:2010.16263)

[28] Craven J, Jejjala V and Kar A 2020 Disentangling a deep learned volume formula (arXiv:2012.03955)

[29] Craven J, Hughes M, Jejjala V and Kar A 2021 Learning knot invariants across dimensions, (arXiv:2112.00016)

[30] Craven J, Hughes M, Jejjala V and Kar A 2022 Illuminating new and known relations between knot invariants (arXiv:2211.01404)
[31] He Y-H and Yau S-T 2020 Graph Laplacians, Riemannian Manifolds and their machine-learning (arXiv:2006.16619)
[32] Bao J, He Y-H, Hirst E, Hofscheier J, Kasprzyk A and Majumder S 2021 Polytopes and machine learning (arXiv:2109.09602)
[33] Alessandretti L, Baronchelli A and He Y-H 2019 *Machine Learning Meets Number Theory* (The Data Science of Birch-Swinnerton-Dyer) (arXiv:1911.02008)
[34] He Y-H, Hirst E and Peterken T 2021 Machine-learning dessins d-enfants: explorations via modular and Seiberg-Witten curves *J. Phys. A: Math Theor.* **54** 075401
[35] He Y-H, Lee K-H and Oliver T 2020 Machine-learning the Sato–Tate conjecture (arXiv:2010.01213)
[36] He Y-H 2021 Machine-learning mathematical structures (arXiv:2101.06317)
[37] Davies A, Veličković P, Buesing L, Blackwell S, Zheng D and Tomašev N *et al* 2021 Advancing mathematics by guiding human intuition with ai *Nature* **600** 70–74
[38] Noether E 1918 Invariant Variation Problems *Gott. Nachr.* **1918** 235–57
[39] Arnol'd V I 2013 *Mathematical Methods of Classical Mechanics* vol 60 (Springer)
[40] Weinberg S 1995 *The Quantum Theory of Fields* vol 1 (Cambridge University Press)
[41] Cohen T S and Welling M 2016 Group equivariant convolutional networks *Proc. 33rd Int. Conf. on Machine Learning, Proc. Machine Learning Research* vol 48 pp 2990–9
[42] Kondor R and Trivedi S 2018 On the generalization of equivariance and convolution in neural networks to the action of compact groups *Int. Conf. on Machine Learning* (PMLR) pp 2747–55
[43] Chen H-Y, He Y-H, Lal S and Majumder S 2021 Machine learning Lie structures & applications to physics *Phys. Lett. B* **817** 136297
[44] Cheung M-W, Dechant P-P, He Y-H, Heyes E, Hirst E and Li J-R 2022 Clustering cluster algebras with clusters (arXiv:2212.09771)
[45] Lee K-H 2023 Data-scientific study of Kronecker coefficients
[46] Gorenstein D 2007 *Finite Groups* vol 301 (American Mathematical Soc.)
[47] Bernstein D 2004 The computational complexity of rules for the character table of sn *J. Symb. Comput.* **37** 727–48
[48] Dixon J D 1969 The probability of generating the symmetric group *Mathemat. Z.* **110** 199–205
[49] Aschbacher M and Guralnick R 1984 Some applications of the first cohomology group *J. Algebra* **90** 446–60
[50] Shalev A 2001 Asymptotic group theory *Not. AMS* **48** 383–9
[51] Mishra C, Moulik S R and Sarkar R 2023 Mathematical conjecture generation using machine intelligence (arXiv:2306.07277)
[52] Vasil'ev A V, Grechkoseeva M A and Mazurov V D 2009 Characterization of the finite simple groups by spectrum and order *Algebra Logic* **48** 385–409
[53] Burness T C 2017 Simple groups, fixed point ratios and applications (arXiv:1707.03564)
[54] Craven J, Hughes M, Jejjala V and Kar A 2024 Illuminating new and known relations between knot invariants *Mach. Learn.: Sci. Technol.* **5** 045061
[55] He Y-H and Burtsev M 2024 Can AI make genuine theoretical discoveries? *Nature* **625** 241–241
[56] The GAP Group, 2024 GAP – Groups, algorithms, and programming, version 4.14.0