Can We Count on Deep Learning: Characterizing Combinatorial Structures With Interpretability

Helen Jenne[1], **Herman Chau**[2], **Davis Brown**[1], **Jackson Warley**[1], **Tim Doster**[1], **Henry Kvinge**[1,2]
[1]Pacific Northwest National Laboratory
[2]Department of Mathematics, University of Washington
`{first.last}@pnnl.gov`, `hchau@uw.edu`

Abstract

With its exceptional pattern matching ability, deep learning has proven to be a powerful tool in a range of scientific domains. This is increasingly true in research mathematics, where recent work has demonstrated deep learning's ability to highlight subtle connections between mathematical objects that might escape a human expert. In this work we describe a simple method to help domain experts characterize a set of mathematical objects using deep learning. Such *characterization problems* often occur when some particular class of function, space, linear representation, etc. naturally emerges in calculations or other means but lacks a simple description. The goal is to find simple rules that also ideally shed light on the underlying mathematics. Our method, which we call *Feature Attribution Clustering for Exploration (FACE)*, clusters the feature attribution representations extracted from a trained model, arriving at a short list of prototype attributions that the domain expert can then try to convert into formal and rigorous rules. As a case study, we use our method to derive a new result in combinatorics by characterizing a subset of 0-1 matrices that corresponds to certain representations of permutations known as two-sided ordered words.

1 Introduction

The search for patterns in large (even infinite) sets of objects is a common task in mathematics. This is especially true in the field of combinatorics, an area of mathematics that counts, characterizes, and studies the properties of discrete structures. Since modern machine learning (ML) excels at extracting subtle, discriminative features from data, a natural question is whether ML can be leveraged to assist research mathematicians in these and other tasks. Preliminary evidence suggests an affirmative answer [11], but the most effective ways to integrate ML into a researcher's workflow remains a nascent area of research.

In this paper we describe a method of using a trained deep learning model as a tool to discover concise conditions that characterize a set of mathematical objects A, a problem type that we call a *characterization problem*. More specifically, we seek a method to help answer questions of the form:

> ***Characterization problem:*** *Given a subset of elements A in set B, find a simple method to determine whether an arbitrary element $b \in B$ belongs to A or not.*

These problems are ubiquitous throughout mathematics. Classical examples include:

- A matrix is *nonsingular* if and only if it has nonzero determinant.
- A subset of $\mathbb{R}^n$ is *compact* if and only if it is closed and bounded.
- A graph is *planar* if and only if it does not contain K_5 or $K_{3,3}$ as a minor.

- Primitive Pythagorean triples (x, y, z) are always of the form $x = r^2 - s^2$, $y = 2rs$, and $z = r^2 + s^2$ for positive coprime integers r and s.

The need for parsimony and human-intelligibility mean that high-level reasoning is especially useful for characterization problems. As such, they are non-trivial for ML which is good at detailed pattern recognition but generally poor at large-scale, comprehensive, or principled representations. To mitigate these limitations, in this paper we present a method to help with characterization problems that we call *feature attribution clustering for exploration (FACE)*. FACE uses a simple procedure where a model is trained in a supervised manner to identify positive examples of the set A out of a population of negative examples. Then explainability tools are employed to extract the features the model attends to when producing predictions, and finally these features are clustered and prototypes of each cluster are formed. When the approach is successful, the prototypes will point toward rules that define A. To illustrate this method, we describe how FACE was used to resolve a combinatorics question involving the form that a certain type of permutation representation, a two-sided ordered word, takes when it is realized as a weaving diagram array.

2 Background

2.1 Related Work

The use of computers to assist in mathematical data generation, guide conjecture formation, and verify proofs has a long and venerable history [1, 22]. Only recently though have deep learning and other modern ML frameworks been exploited for the purpose of advancing research-level mathematics. Examples span many of the major mathematical subfields, including representation theory [11, 6, 20, 8], knot invariants [11, 10], combinatorics and graph theory [25], differential and algebraic geometry [5, 4, 9], mathematical physics [2, 3], quantum algebra [19], and number theory [16].

Besides making progress in mathematics, many of these works are also notable in their development of methodologies by which deep learning can be used to advance mathematics. For example, both [11] and [9] attempt to identify connections between two sets of mathematical objects (knot invariants in the former and the quantum period and dimension of Fano varieties in the latter). On the other hand [25] uses a reinforcement learning framework to look for counterexamples to conjectures in combinatorics. The present work is unique, to our knowledge, in that the problem type that we look into is characterization.

2.2 Algebraic combinatorics

Algebraic combinatorics is the study of combinatorial objects such as permutations, partially ordered sets, and symmetric functions, using algebraic methods such as group theory and linear algebra. Tools and results from algebraic combinatorics underlie vast swathes of mathematics, such as algebraic geometry, commutative algebra, and statistical mechanics. Compared to other areas of pure mathematics, algebraic combinatorics is also more amendable to computational methods and the generation of data, making it a good choice for study with machine learning.

The central character of our work is the symmetric group, which is the group of permutations of n distinct elements. The symmetric group is ubiquitous throughout math and computer science with applications in cryptography, parallel and distributed computing, and geometric complexity theory, to name a few. It even makes an appearance in deep learning via the type of equivariance needed by graph neural networks and the symmetries intrinsic to deep learning models [13].

3 Feature Attribution Clustering for Exploration

In this section we describe our approach, *feature attribution clustering for exploration (FACE)*, which is designed to help a research mathematician arrive at a succinct characterization of a subset A of mathematical objects out of all B, where $A \subset B$. The method uses a performant deep learning model trained to classify whether $x \in B$ belongs to A or $B \setminus A$. The challenge of using standard explainability techniques in these settings is that most explainability methods produce explanations for each example in B and when B is large, the domain expert faces the task of consolidating many explanations into a few (hopefully simple) rules. FACE addresses this by clustering explanations with the goal of identifying a short list of prototype patterns.

Suppose that f_{θ_0} is a randomly initialized model from a chosen deep learning architecture. The user should choose some integer $k \leq 10$ (the number of feature attribution prototypes) as well as a metric with which to compare feature attribution representations (e.g., comparing saliency maps with respect to Euclidean distance). FACE proceeds as follows.

1. **Dataset creation:** Make a dataset D out of B by labeling elements of B by 0 if they are in $B \setminus A$ and 1 if they are in A. Split D into training and testing sets D_{train}, D_{test}.
2. **Model training:** Train f_{θ_0} on D_{train} to obtain a model f_θ (with trained weights θ) that achieves reasonable accuracy on D_{test}.
3. **Calculate feature attribution representations:** For each $x \in D_{\text{test}}$, use a feature attribution method to generate representations of feature importance with respect to predictions by f_θ. Call this set U. Note that $|U| = |D|$.
4. **Find prototype feature attributions through clustering:** Cluster elements of U into k different clusters $\{U_i\}_{i \leq k}$ and calculate centroids (or some other type of consolidated representation) $u_1, \ldots, u_k$ for each cluster.
5. **Mathematician analysis of prototypes:** Analyze $u_1, \ldots, u_k$ with the goal of distilling the prediction patterns of f_θ into simple rules to characterize elements of A.

Though it is relatively simple, this approach has the virtue of reducing the burden of analyzing large numbers of feature attribution representations. Indeed, in the example we describe below, clustered feature attribution maps summarized model prediction patterns well enough for a mathematician to extract general conditions describing A. Finally, we note that we have sometimes found it useful to do a second round of clustering on a single cluster to further analyze the patterns captured by a prototype, but we leave this step out of the general algorithm.

4 Case Study: Two-Sided Ordered Words

To test the effectiveness of FACE, we applied it to an algebraic combinatorics question from one of our mathematician collaborators. Because of space constraints, we provide a more comprehensive (though still very brief) explanation of the combinatorial cast of characters in Section 6.1 of the Appendix. Here we provide surface level scene-setting, ignoring all of the mathematical motivation (and beauty) in favor of focusing on the ways FACE was used and the extent to which it was effective.

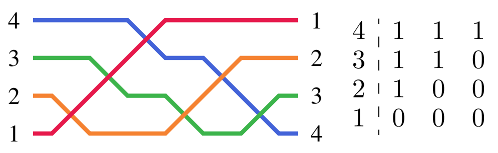

Figure 1: A wiring diagram and weaving pattern.

For $n \geq 1$, the weaving patterns of size n are a family of $n \times (n-1)$ $\{0,1\}$-matrices that correspond to representations of the permutation that reverses the order of $1, 2, \ldots, n$ (i.e., i gets sent to $n - i + 1$). Informally, such a permutation σ_0 can be realized by a diagram with n nodes on the left and n nodes on the right, where the ith node on the left is connected with the $(n - i + 1)$th node on the right by a curve. To form the ith row of the weaving pattern associated to a diagram D, one simply follows the curve from the ith node from left to right, adding a 0 every time the curve crosses another moving up and adding a 1 every time the curve crosses another moving down (see Figure 1). Weaving patterns were discovered independently by [12] and [7].

Two-sided ordered (TSO) words are representations of σ_0 that are defined by decomposing σ_0 into a product of adjacent transpositions (that is, the permutations that exchange i and j but leave all other elements of $\{1, 2, \ldots, n\}$ fixed). We provide more details in Section 6.1. The reader that is not interested in the mathematical background can think of them as a special subset of diagrams representing σ_0.

The question that we explored was: *Find a set of rules or an algorithm that defines the weaving patterns that correspond to two-sided ordered words.*

4.1 Characterizing TSO Weaving Patterns

To apply FACE, we set up a dataset consisting of all TSO words from the permutations of 10 elements converted into weaving patterns. This amounted to $48,896$ $\{0,1\}$-arrays of size 10×9. We also computed $48,896$ weaving patterns that did not correspond to TSO words. We split these into training and test sets and trained a convolutional neural network consisting of two convolutional layers followed by one linear layer to classify TSO words. Our model achieved an accuracy of 99.0% on the test set (Section 6.2 in the Appendix contains further details on data generation and model training). We then applied Shapley [21, 18] to 16,000 weaving patterns from the test set. Shapley values require a notion of 'missingness,' where the feature attribution is calculated by considering the increased output caused by (groups of) features over their missing feature baseline. For machine learning models, a missing feature is often not well-defined [15], here we use both the zero matrix and all-ones matrix as baselines. Following the FACE procedure we applied k-means clustering to the Shapley outputs. We experimented with several different values of k, finding that many of the same patterns appeared regardless of k. We highlight key insights that led us to a general algorithm below in the case $n = 9$ (so that both the weaving patterns and Shapley output were 10×9 binary matrices).

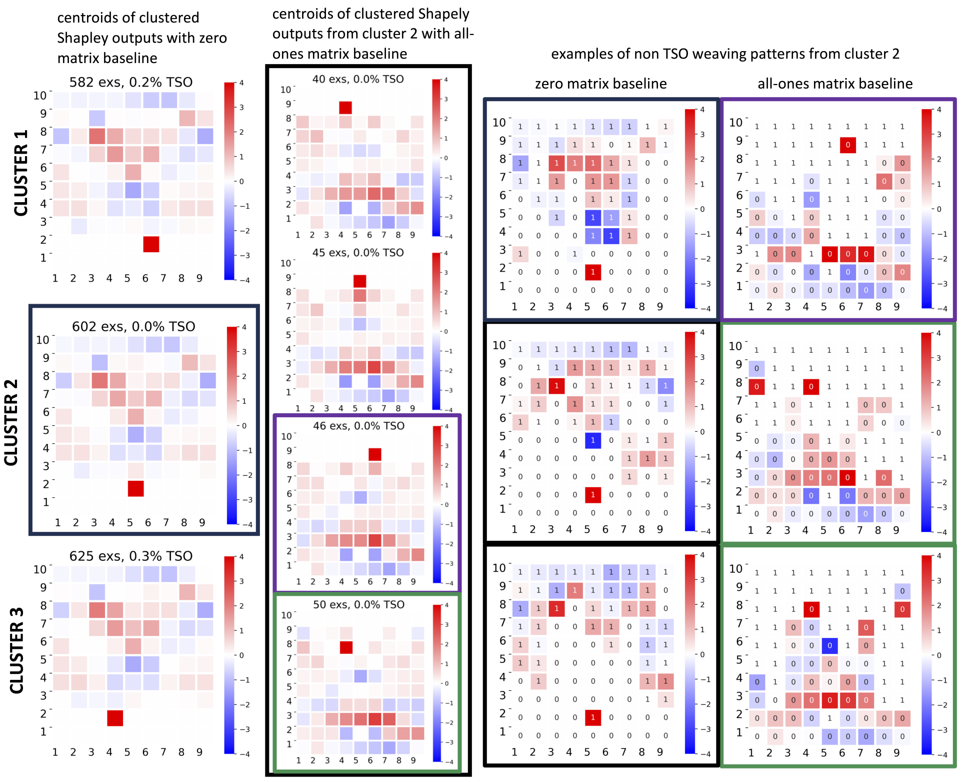

Figure 2: (**First column**) Three centroids of clustered Shapley outputs (with a zero matrix baseline). (**Second column**) Centroids from our second clustering of elements of cluster 2, computed with respect to the all-ones matrix as a baseline. (**Third and fourth column**) Three examples of non-TSO weaving patterns from cluster 2 are shown with superimposed Shapley outputs using a zero matrix baseline (left) and an all-ones matrix baseline (right).

Using FACE prototypes to identify a first necessary condition: In three of the FACE clusters the model strongly attended to a 1 in row 2. These clusters appeared for multiple choices of k and are visualized in Figure 2 (left column). This suggested that a 1 in this row related to a property characterizing those weaving patterns not corresponding to TSO words (though the simple presence of a 1 in that row was not enough as 3.6% of the TSO weaving patterns had a 1 in row 2, column 5).

To gain further insight we focused on cluster 2 (which was quite large) and applied k-means again to just these elements, this time using Shapley values relative to the all-1's matrix baseline (to focus on the effect of 0's). In three (out of eight) of the centroids from this second round of clustering, the model attended to the 0 in row 9 of the weaving pattern (top three examples in column two of Figure 2). Together these two observations eventually led to the following proposition.

Proposition 1. Let D be a $\{0, 1\}$-array of size $(n + 1) \times n$. Then if D has a 1 in one of the middle three positions of row 2 and a 0 in one of the three middle three positions of row n, it cannot correspond to a TSO word.

Identifying other cases: Unfortunately, this criteria does not cover all cases. For example, row 9 often contains 8 consecutive 1's, so the first criterion we developed does not apply. Looking for patterns that may include these other cases, we noticed that in cluster 4 the model highly attended to 0's that were placed far apart in row 8 (see the examples highlighted in green in Figure 2). This led to the following proposition.

Proposition 2. Let D be an $(n + 1) \times n$ weaving pattern which corresponds to a TSO weaving pattern and suppose it has a 1 in one of the middle three positions in row 2, the 0's in row $n - 1$ can have at most one 1 in between them.

Having identified these two conditions, we looked for examples that fell outside of these cases. In these examples, we noticed that the model appeared to attend more to cells in the rows less than 9 and greater than 2. That is, when the model was not able to find features in the outer rows (row 2 and row 9) that differentiated between TSO and non-TSO words, it moved inward (looking at rows 3 and 8 for instance). This suggested the general outline for the algorithm that we eventually developed (see Section 7.1 in the Appendix for the details) that also proceeds iteratively, comparing pairs of rows 2 and $n - 1$, then 3 and $n - 2$, then 4 and $n - 3$, etc. until a condition is found that can be used to determine whether a weaving pattern corresponds to a TSO word or not. While in most cases we are aware of, machine learning approaches to questions in mathematics have largely succeeded by identifying specific correlations between features, counterexamples, etc., we were surprised that in this instance FACE also suggested the structure in the desired algorithm.

5 Conclusion

There are now multiple examples that suggest that deep learning can be an effective tool to accelerate progress in research level mathematics. However, effective methodologies for applying deep learning to the specific types of problems that mathematicians work on are still needed. In this work we presented FACE, a method that can be used to help a mathematician find a solution to a characterization problem. As proof of the method's utility, we described how FACE was able to help a mathematician solve a combinatorics problem involving representations of a certain family of permutations.

Acknowledgments and Disclosure of Funding

This research was supported by the Mathematics for Artificial Reasoning in Science (MARS) initiative at Pacific Northwest National Laboratory. It was conducted under the Laboratory Directed Research and Development (LDRD) Program at at Pacific Northwest National Laboratory (PNNL), a multiprogram National Laboratory operated by Battelle Memorial Institute for the U.S. Department of Energy under Contract DE-AC05-76RL01830.

References

[1] David Bailey, Jonathan Borwein, Neil Calkin, Russell Luke, Roland Girgensohn, and Victor Moll. *Experimental mathematics in action*. CRC press, 2007.

[2] Jiakang Bao, Yang-Hui He, Elli Heyes, and Edward Hirst. Machine learning algebraic geometry for physics. *arXiv preprint arXiv:2204.10334*, 2022.

[3] Jiakang Bao, Yang-Hui He, Edward Hirst, Johannes Hofscheier, Alexander Kasprzyk, and Suvajit Majumder. Hilbert series, machine learning, and applications to physics. *Physics Letters B*, 827:136966, 2022.

[4] Per Berglund, Ben Campbell, and Vishnu Jejjala. Machine learning kreuzer–skarke calabi–yau threefolds. *arXiv preprint arXiv:2112.09117*, 2021.

[5] David S Berman, Yang-Hui He, and Edward Hirst. Machine learning calabi-yau hypersurfaces. *Physical Review D*, 105(6):066002, 2022.

[6] Charles Blundell, Lars Buesing, Alex Davies, Petar Veličković, and Geordie Williamson. Towards combinatorial invariance for kazhdan-lusztig polynomials. *Representation Theory of the American Mathematical Society*, 26(37):1145–1191, 2022.

[7] Herman Chau. Commutation classes of permutations. Seminar talk at University of Puget Sound, 2023.

[8] Man-Wai Cheung, Pierre-Philippe Dechant, Yang-Hui He, Elli Heyes, Edward Hirst, and Jian-Rong Li. Clustering cluster algebras with clusters. *arXiv preprint arXiv:2212.09771*, 2022.

[9] Tom Coates, Alexander M Kasprzyk, and Sara Veneziale. Machine learning the dimension of a fano variety. *Nature Communications*, 14(1):5526, 2023.

[10] Jessica Craven, Mark Hughes, Vishnu Jejjala, and Arjun Kar. Illuminating new and known relations between knot invariants. *arXiv preprint arXiv:2211.01404*, 2022.

[11] Alex Davies, Petar Veličković, Lars Buesing, Sam Blackwell, Daniel Zheng, Nenad Tomašev, Richard Tanburn, Peter Battaglia, Charles Blundell, András Juhász, et al. Advancing mathematics by guiding human intuition with ai. *Nature*, 600(7887):70–74, 2021.

[12] Stefan Felsner. On the number of arrangements of pseudolines. *Discrete Computational Geometry*, 18:257–267, 1997.

[13] Charles Godfrey, Davis Brown, Tegan Emerson, and Henry Kvinge. On the symmetries of deep learning models and their internal representations. *Advances in Neural Information Processing Systems*, 35:11893–11905, 2022.

[14] Gonçalo Gutierres, Ricardo Mamede, and José Luis Santos. Commutation classes of the reduced words for the longest element of S_n. *Electron. J. Combin.*, 27(2):Paper No. 2.21, 25, 2020.

[15] Saachi Jain, Hadi Salman, Eric Wong, Pengchuan Zhang, Vibhav Vineet, Sai Vemprala, and Aleksander Madry. Missingness bias in model debugging. *arXiv preprint arXiv:2204.08945*, 2022.

[16] Matija Kazalicki and Domagoj Vlah. Ranks of elliptic curves and deep neural networks. *Research in Number Theory*, 9(3):53, 2023.

[17] Donald E Knuth. *Axioms and hulls*. Springer, 1992.

[18] Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. Captum: A unified and generic model interpretability library for pytorch, 2020.

[19] Shailesh Lal, Suvajit Majumder, and Evgeny Sobko. The r-matrix net. *arXiv preprint arXiv:2304.07247*, 2023.

[20] Kyu-Hwan Lee. Machine-learning kronecker coefficients. *arXiv preprint arXiv:2306.04734*, 2023.

[21] Lloyd S Shapley et al. A value for n-person games. 1953.

[22] Carlos Simpson. Computer theorem proving in mathematics. *Letters in Mathematical Physics*, 69:287–315, 2004.

[23] Richard P Stanley. On the number of reduced decompositions of elements of coxeter groups. *European Journal of Combinatorics*, 5(4):359–372, 1984.

[24] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 10.1)*, 2023. `https://www.sagemath.org`.

[25] Adam Zsolt Wagner. Constructions in combinatorics via neural networks. *arXiv preprint arXiv:2104.14516*, 2021.

6 Supplementary Material

6.1 The Cast of Combinatorial Characters

Weaving patterns come from the combinatorial study of the symmetric group S_n, the group of permutations on $\{1, 2, \ldots, n\}$. We will write permutations $\pi \in S_n$ in one-line notation; for example, the permutation π that maps $1 \mapsto 4$, $2 \mapsto 3$, $3 \mapsto 2$, $4 \mapsto 1$ is written 4321. Of particular importance are the longest permutation $\pi_0 = n(n-1)(n-2)\cdots 1$ and the adjacent transpositions s_i, which swap i and $i+1$ while holding all other numbers fixed.

It is well known that any permutation can be written as a product of adjacent transpositions. For example, the permutation $\pi_0 = 4321$ can be written $s_1 s_2 s_3 s_1 s_2 s_1$, as visualized using the *wiring diagram* in Figure 1. We will abbreviate this product as 123121 and call it a *reduced word*. Reduced words for π_0 are equivalent to primitive sorting networks and of interest to both combinatorialists and computer scientists [17]. The word 'reduced' indicates that there is no way to write this permutation as a product of fewer transpositions.

In general, the permutation π_0 has many different reduced words. The combinatorics of reduced words for π_0 is well-studied [23]. On the other hand, one can define a equivalence relation on reduced words and partition them into classes called *commutation classes* that are less well understood. We are particularly interested in a subset of 0-1 matrices, weaving patterns, that are in bijection with commutation classes of π_0.

Definition 1. The *weaving pattern* of a reduced word of π_0 is an $n \times (n-1)$ matrix whose ith row is a binary sequence obtained by following the wire labeled i from left to right in the wiring diagram and recording a 0 for every crossing in which it goes *up* and a 1 for every crossing in which it goes *down*.

In this case study we characterize weaving patterns of a family of reduced words recently introduced by Gutierres et al. [14] called two-sided ordered (TSO) words. These words are a generalization of another family they introduced, that of *ordered words*. In order to develop intuition and to aid in the proofs of our TSO characterization we first discuss the family of ordered words.

6.1.1 Ordered weaving patterns

Gutierres et al. [14] study two families of reduced words. The first family that they study are called *ordered words*. An example of an ordered word for $\pi_0 \in S_5$ is: $w = 4321 \cdot 234 \cdot 32 \cdot 3$. In general an ordered word is defined by a binary vector of length $n-1$ as follows:

Definition 2 ([14]). Given a vector $\mathfrak{b} = (\mathfrak{b}_1, \mathfrak{b}_2, \ldots, \mathfrak{b}_{n-1})$ with $\mathfrak{b}_i \in \{\uparrow, \downarrow\}$ for all i, construct the word $w^{\mathfrak{b}}$ as a concatenation of monotone subwords $w_{n-\ell+1}$, $w^{\mathfrak{b}} = w_{n-1} w_{n-2} \ldots w_1$, which are defined recursively as follows. Let $w_n^{\uparrow} = 12\cdots n$ and $w_n^{\downarrow} = n \cdots 21$. Then for $\ell = 1, \ldots, n-1$, $w_{n-\ell}^{\mathfrak{b}_{\ell+1}}$ is obtained from $w_{n-\ell+1}^{\mathfrak{b}_\ell}$ by removing the last letter. If $\mathfrak{b}_{\ell+1} = \uparrow$, $w_{n-\ell}^{\mathfrak{b}_{\ell+1}}$ is increasing, otherwise $w_{n-\ell}^{\mathfrak{b}_{\ell+1}}$ is decreasing.

An example of an ordered word for $\pi_0 \in S_5$ is: $w = 4321 \cdot 234 \cdot 32 \cdot 3$.

Gutierres et al. show that any ordered word is a reduced word for π_0 [14, Prop. 19], and that the set of ordered words $\mathcal{O}(n)$ contains 2^{n-1} distinct words which each belong to a different commutation class [14, Prop. 20].

We found that a binary matrix is a weaving pattern for a commutation classes containing an ordered word if and only if it has a decomposition into a sequence of decreasing nested "L"s (see Figure 3). This characterization did not require any ML techniques but resulted in a key observation that was important for our work on the TSO characterization: in ordered words, the subword $w_{n-\ell}$ corresponds to a row containing a sequence of $n-\ell$ consecutive 1's or 0's, depending on whether it is increasing or decreasing.

In order to characterize the weaving patterns associated with ordered words (which we call *ordered weaving patterns*), we introduce one more definition. We label the cells of the weaving diagram so that the cell in the jth column of the row labeled i is cell (i, j).

Definition 3. In a weaving pattern, an L centered at (i, j) of length $2k$ is either

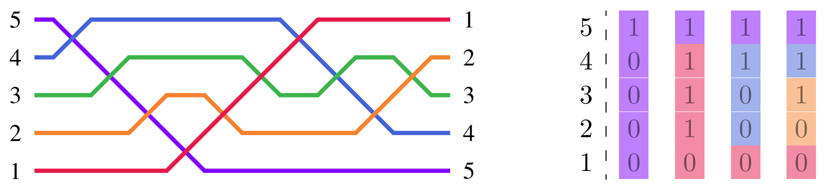

Figure 3: A wiring diagram for the reduced word $4321 \cdot 234 \cdot 32 \cdot 3$ (left) and the associated weaving pattern (right).

(1) a sequence of consecutive 1's in the cells $(i, j), (i, j + 1), \ldots, (i, j + k)$ and a sequence of consecutive 0's in the cells $(i - 1, j), (i - 2, j), \ldots, (i - k, j)$, or

(2) a sequence of consecutive 0's in the cells $(i, j), (i, j + 1), \ldots, (i, j + k)$ and a sequence of consecutive 1's in the cells $(i + 1, j), (i - 2, j), \ldots, (i + k, j)$.

In the first case, we call the L upward-facing, and in the second case we call the L downward-facing

In the examples in Figure 3, the weaving patterns can be decomposed into Ls of decreasing length, each corresponding to a subword. The L is upward facing when the corresponding subword is increasing, and downward facing when the corresponding subword is decreasing. This is the case in general, and follows from the wiring diagram representation.

Definition 4. If a weaving pattern for a reduced word in S_{n+1} can be decomposed into adjacent Ls that decrease in length from left to right so that the lengths of the Ls in order are $2n, 2(n - 2), \ldots, 2$, we say that the weaving pattern has a *decreasing nested L-decomposition*.

Proposition 5. A binary matrix is an ordered weaving pattern if and only if it has a decreasing nested L-decomposition.

Proof. To see that each ordered weaving pattern has a decreasing nested L-decomposition, it suffices to observe that:

(1) An ordered word can be decomposed into subwords of lengths n, $n - 1$, . . ., 1 (reading left to right)

(2) A decreasing subword of length k gives rise to an L of length $2k$ in the weaving pattern

Although we have not proven the second claim rigorously, it can be proven by induction.

For the reverse direction, we count the number of weaving patterns with decreasing nested L-decompositions. Observe that there are 2^{n-1} decreasing nested L decompositions for a word that uses the letters $1, \ldots, n$, since we can choose whether the Ls are upward or downward facing for the first $n - 1$ Ls (the last L is determined once the first $n - 1$ are chosen). Two ordered words cannot have the same weaving pattern since Gutierres et al. established that there is at most one ordered word in each conjugacy class. So the facts that there are 2^{n-1} ordered words ([14, Proposition 20]) and the map from ordered weaving patterns to nested L decompositions is an injection establish the claim. □

Proposition 5 is an example of the type of characterization described in Section 1. We conclude our discussion of ordered words with an explicit construction of the weaving pattern for a particular ordered word.

Proposition 6. To construct the weaving pattern for the ordered word $w_n^{b_1} w_{n-1}^{b_2} \cdots w_{n-\ell}^{b_{\ell+1}}$, for $\ell \geq 0$, let $k = |\{b_i : b_i = b_{\ell+1} \text{ and } i < \ell + 1\}|$. Then

- If $b_{\ell+1} = -$, the cells $\{(n + 1 - k, j) : \ell + 1 \leq j \leq n\}$ are 1's and the cells $\{(n + 1 - k - i, \ell + 1) : 1 \leq i \leq n - \ell\}$ are 0's.
- If $b_{\ell+1} = +$, the cells $\{(1 + k, j) : \ell + 1 \leq j \leq n\}$ are 0's and the cells $\{(1 + k + i, \ell + 1) : 1 \leq i \leq n - \ell\}$ are 1's.

Example 7. In Figure 3,

- $b_1 = -$ and indeed when $\ell = 0$, the cells $(n+1, j)$ are 1's for $1 \leq j \leq n$ and $(n+1-i, 1)$ are 0's for $1 \leq i \leq n$.
- $b_2 = +$; when $\ell = 1$, the cells $(1, j)$ for $2 \leq j \leq n$ are 0's and the cells $(1+i, 2)$ for $1 \leq i \leq 3$ are 1's
- $b_3 = -$; when $\ell = 2$, the cells $(n+1-1, j)$ for $3 \leq j \leq n$ are 1's and $(n-i, 3)$ for $1 \leq i \leq 2$ are 0's.
- Considering $b_4 = +$ (b_4 can be either $+$ or $-$), when $\ell = 3$, the cell $(2, 4)$ is 0 and the cell $(3, 4)$ is 1.

6.1.2 TSO words

Like an ordered word, a TSO word contains exactly one subword of each length (denoted $w_1, \ldots, w_n$) and is defined as follows:

Definition 8. Given an ordered word $w^{\mathfrak{b}}$ and a partition $I \cup J$ of $\{1, 2, \ldots, n-1\}$, the *two-sided ordered word* $w_I^{\mathfrak{b}}$ is the concatenation $\prod_{i \in I} \underline{w}_i^{\mathfrak{b}_{n-i+1}} w_n^{\mathfrak{b}_1} \prod_{j \in J} w_j^{\mathfrak{b}_{n-j+1}}$, where the word $\underline{w}_i^{\mathfrak{b}_{n-i+1}}$ is the word obtained from $w_i^{\mathfrak{b}_{n-i+1}}$ by adding $\sum_{k=i+1}^{n} \mathfrak{b}_k$ to each letter, considering $\uparrow$ and $\downarrow$ as 1 and -1, respectively. In the product on the left hand (resp. right hand) side, the factors are written in increasing (resp. decreasing) orders of their lengths.

Note that instead of requiring the subwords be in decreasing order by length, they can be on either side of w_n, as long as the subwords on the left are in increasing order and the subwords on the right are in decreasing order.

All TSO words are reduced words for π_0, since they are obtained from an ordered word by applying braid relations. In total there are $3 \cdot 4^{n-2}$ TSO words [14, Lemma 23]. For the remainder of this paper, we will focus on the $3 \cdot 4^{n-2} - 2^{n-1}$ *non-alternating* TSO words, which each belong to a distinct commutation class [14, Prop. 46]. We call the weaving patterns for these commutation classes TSO weaving patterns.

6.2 Experimental details

Dataset generation: We generated all TSO words on $n = 9$ letters and then generated their weaving patterns using SageMath's Pseudolines module [24]. When $n = 9$ there are 48,896 TSO weaving patterns, and we generated the same number of non-TSO weaving patterns by permuting the rows of the TSO weaving patterns, and checking whether the resulting matrix was a weaving pattern that was not TSO.

Model: We split the data set into 60% training samples, 15% validation samples, and 25% test samples. The model was trained using Adam for 50 epochs with early stopping if the validation loss did not improve for five consecutive epochs. Our model achieved an accuracy of 99.0% on the test set.

k-means clustering details: After applying k-means clustering to Shapley outputs with baseline the zero matrix we found that the clusters were either clusters of 99.7-100% non-TSO examples or 98-98.5% TSO examples, for $k \in \{2, 3, \ldots, 16\}$. Table 1 in the Appendix shows the sizes of the clusters; we found that many of the non-TSO clusters were unchanged when k changed. Figure 4 shows the centroids of the clusters of Shapley outputs obtained when $k = 8$.

7 Proofs of Propositions 1 and 2

Proof of Proposition 1. Since, like ordered words, TSO words contain one subword of each of the lengths $n, n-1, n-2, \ldots, 1$, the statements about sequences of consecutive 1's and 0's in the rows from Proposition 6 still hold. If a weaving pattern is TSO and has a 0 in one of the middle three positions of row n and a 1 in one of the middle three positions in row 2, it must be the case that w_n and w_{n-1} correspond to the top and bottom rows. But then since there is not a sequence of $n-2$ consecutive 1's in row n or a sequence of $n-2$ consecutive 0's in row 2, there cannot be a subword of length $n-2$, contradicting the assumption that the weaving pattern is TSO. $\square$

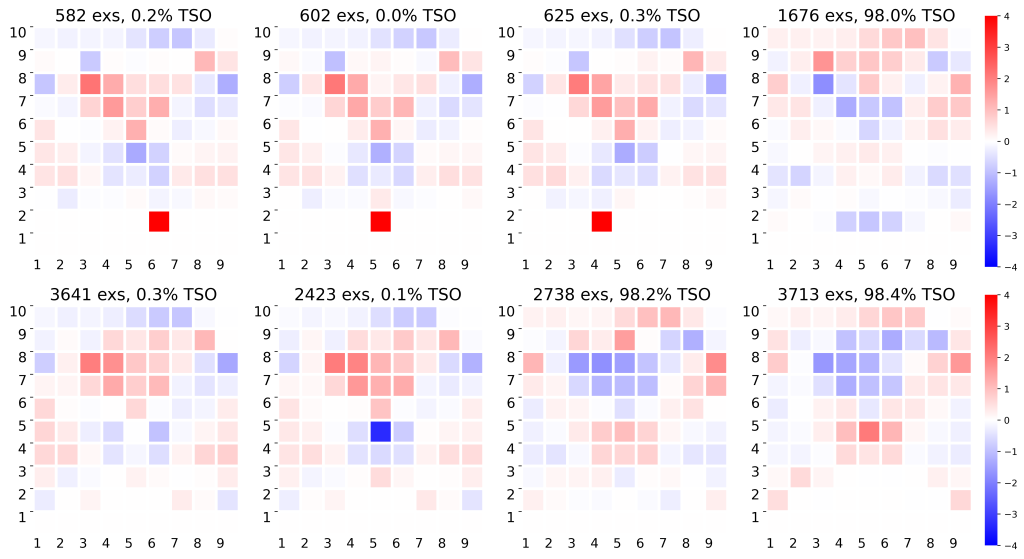

Figure 4: Cluster centroids after k-means clustering was applied to Shapley outputs using the zero matrix baseline with $k = 8$.

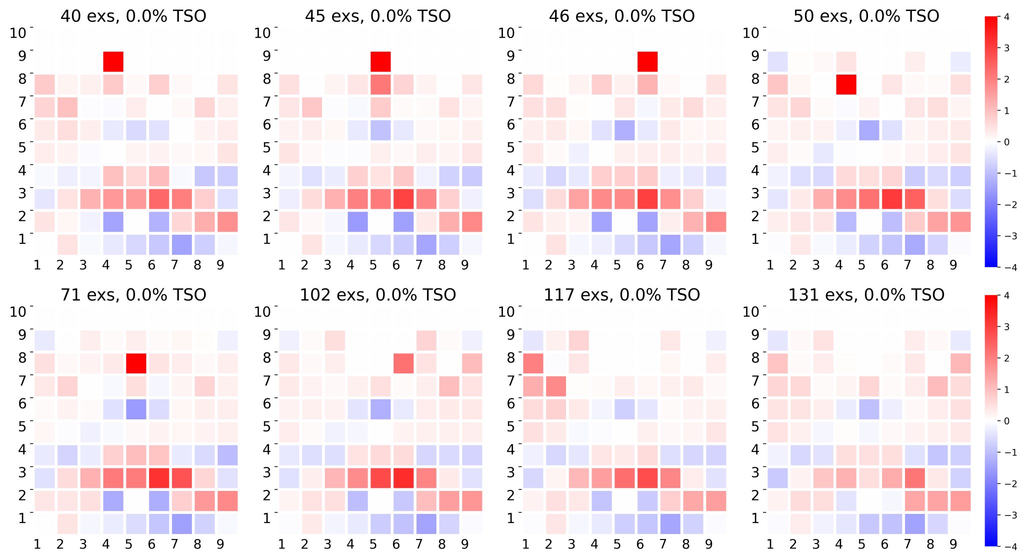

Figure 5: The cluster centroids after applying k-means clustering to cluster 1 from Figure 4, this time using the all 1's baseline.

Proof of Proposition 2. As in Proposition 6, every decreasing word creates a vertical sequence of 0's equal to the length of the word (but unlike in the ordered case, these 0's do not necessarily have to be in the same column). Consequently, if a 1 is in one of the middle three positions in row 2, two of w_n, w_{n-1}, and w_{n-2} must be decreasing, which will result in two 0's that are separated by at most a single 1 in row $n - 1$. $\square$

7.1 Algorithm to recover TSO word from the weaving pattern

The input to the algorithm is a binary matrix with rows labeled $1, \dots, n+1$. At each step we modify a list of possible subwords and the associated patterns. More precisely, each element in our list is of the form (subword, [list of (required pattern, rows of required pattern)]), and at each step we either remove an element of the list or extend the subword and list of required patterns. The algorithm

k	non-TSO cluster sizes	TSO cluster sizes
2	7873	8127
3	$602, 7271$	8127
4	$602, 7271$	$3585, 4542$
5	$602, 1215, 6056$	$3585, 4542$
6	$602, 624, 2671, 3976$	$3573, 4554$
7	$585, 602, 627, 6059$	$2227, 2862, 3038$
8	$582, 602, 625, 2423$	$1676, 2738, 3641, 3713$
9	$582, 602, 625, 1567, 2423$	$1578, 2464, 2518, 3641$
10	$581, 602, 622, 1528, 2406, 2134$	$1565, 1604, 2449, 2509$
11	$581, 602, 622, 1528, 2135, 2405$	$1304, 1461, 1517, 1673, 2172$
12	$578, 581, 602, 623, 1435, 1986, 2068$	$853, 1503, 1504, 1838, 2429$
13	$578, 581, 602, 623, 1435, 1986, 2068$	$284, 1183, 1233, 1492, 1569, 2366$
14	$460, 539, 581, 601, 624, 1332, 1816, 1920$	$310, 1167, 1209, 1493, 1576, 2372$
15	$525, 581, 602, 622, 974, 1077, 1432, 2060$	$285, 310, 1018, 1157, 1476, 1522, 2359$

Table 1: Table of cluster sizes after applying k-means to Shapley outputs with the zero matrix as a baseline. Across different choices for k, we consistently observe smaller clusters with similar sizes, as is shown by coloring these cluster sizes red, blue, and purple.

terminates when this list is empty or we have a word that contains monotone subwords of length $1, 2, \dots, n$.

At the beginning of the algorithm, the list is $[(w_n^{\downarrow}, [(0, \text{rows } 1 - n)]), (w_n^{\uparrow}, [(1, \text{rows } 2 - n + 1)]$, where $w_n^{\downarrow} := n \cdots 21$ and $w_n^{\uparrow} := 12 \cdots n$, because at the outset it is possible for w_n to be increasing or decreasing. In the first case it is associated to the n consecutive 1's in row $n + 1$, and in the second case it is associated to the n consecutive 0's in row 1.

In general, assume we have a sequence that contains the factors $w_n, w_{n-1}, \dots$ and a list of the required patterns associated to those factors. At step i, for each (subword, [list of (required pattern, rows of required pattern)]) in the list we need to determine whether it is possible for w_{n-i} to be increasing and/or decreasing, and whether it can appears to the left and/or the right of the sequence we've established so far.

(1) We start with the highest row that is not already associated to one of the subwords. If this row has i consecutive 1's, this is a necessary but not sufficient condition for the subword w_{n-i} to be decreasing. If this condition is satisfied, go to step (a). Otherwise, go to step (2).

(a) If the required pattern associated to our sequence occurs in the $n - i$ rows below the current row with a 0 to the right (resp. to the left), then the subword w_{n-i} can be decreasing and can appear to the left (resp. to the right) of the current sequence.

(2) Next check if w_{n-i} can be increasing. This is analogous to the check for decreasing: we start with the lowest row that is not already associated to one of the subwords, and check for appearances of the required pattern with a 1 appended to the right or left.

7.2 Example of TSO algorithm

Example 1. We wish to construct the TSO word corresponding to the weaving pattern:

$$\begin{array}{c|cccccc}
7 & 1 & 1 & 1 & 1 & 1 & 1 \\
6 & 0 & 1 & 1 & 1 & 1 & 1 \\
5 & 0 & 1 & 1 & 0 & 1 & 1 \\
4 & 0 & 1 & 1 & 0 & 0 & 1 \\
3 & 0 & 1 & 1 & 0 & 0 & 0 \\
2 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}$$

We end step i with a list of all possible subwords of length i and the associated required patterns.

Step 1: The list of possible (subwords, required patterns) is ($w_6^{\downarrow}$, (0, rows 1-6)) and ($w_6^{\uparrow}$, (1, rows 2-7)).

Step 2 (a): First we consider the possible subword $w_6^{\downarrow}$ and its required pattern (0, rows 1-6). Assuming this subword, is w_5 increasing or decreasing, and does it appear to the right or to the left of w_6? If w_5 is decreasing it corresponds to the 5 consecutive 1's in row 6 (see Figure 6, left). But since row 5 does not have two consecutive 0's (highlighted in red), this is not possible.

If instead w_5 is increasing this corresponds to five of the 0's in row 1. Because $w_6^{\downarrow}w_5^{\uparrow}$ would require the pattern 01 in rows 2-6 and $w_5^{\uparrow}w_6^{\downarrow}$ would require the pattern 10 in rows 2-6, this is not possible (see Figure 6, center). So there are not any possible subwords of length 2 where w_6 is decreasing.

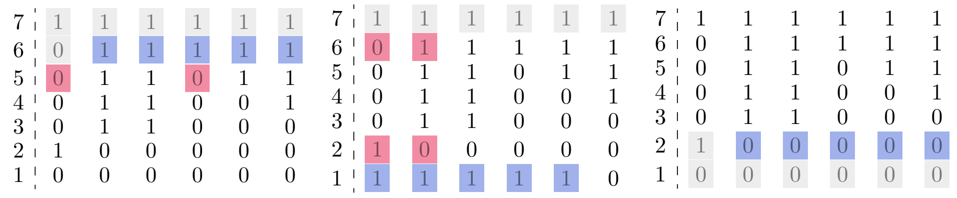

Figure 6: Illustration of the different cases in Step 2

(b) Next we consider the subword $w_6^{\uparrow}$ and its required pattern (1, rows 2-7)). Since row 2 has five consecutive 0's and rows 3-7 have two consecutive 1's, it is possible for w_5 to be increasing. Since the 1 in row 2 is in the leftmost position, $w_5^{\uparrow}$ must appear to the right of $w_6^{\uparrow}$ (see Figure 6, right). So at the end of step 2 our list will include ($w_6^{\uparrow}w_5^{\uparrow}$, [(1, rows 2-7), (11, rows 3-7)]). If w_5 is decreasing, either the pattern 10 must appear in rows 2-6 or the pattern 01 must appear in rows 2-6. Since we saw in (a) that neither of these patterns appears in all of these rows, this is not possible. So at the end of this step, our list of possible (subwords, required patterns) is ($w_6^{\uparrow}w_5^{\uparrow}$, [(1, rows 2-7), (11, rows 3-7)]).

Step 3: Starting with the subword $w_6^{\uparrow}w_5^{\uparrow}$ and the required patterns [(1, rows 2-7), (11, rows 3-7)]), we see that w_4 cannot be increasing because there are not four consecutive 0's in row 3. If w_4 is decreasing, this will correspond to 4 of the 1's in row 7, and either the pattern 011 or the pattern 110 is required in rows 3-6. As shown in Figure 7 (left), 011 is a pattern in all of these rows but 110 is not a pattern in row 6, our list of possible subwords of length 3 and corresponding required patterns is ($w_4^{\downarrow}w_6^{\uparrow}w_5^{\uparrow}$, [(1, rows 2-7), (11, rows 3-7), (011, rows 3-6)]).

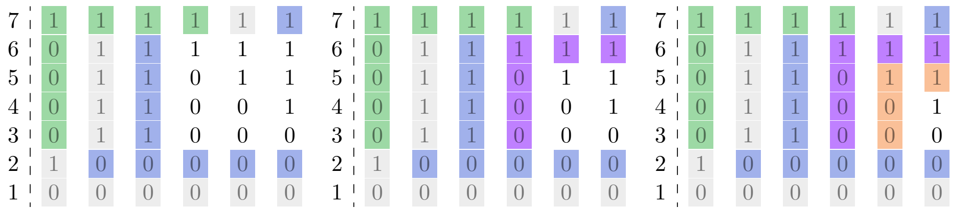

Figure 7: Illustration of the cases in steps 3 and 4.

Step 4: Beginning with the subword $w_4^{\downarrow}w_6^{\uparrow}w_5^{\uparrow}$ and the required patterns [(1, rows 2-7), (11, rows 3-7), (011, rows 3-6)], if w_3 is decreasing it corresponds to 3 of the consecutive 1's in row 6 and we must grow the pattern 011 to either 0110 or 0011 in rows 3-5. The pattern 0110 exists in all of these rows, as shown in Figure 7 (center), so the subword ($w_4^{\downarrow}w_6^{\uparrow}w_5^{\uparrow}w_3^{\downarrow}$, [(1, rows 2-7), (11, rows 3-7), (011, rows 3-6), (0110, rows 3-5)]) is a possible subword. We see that w_3 cannot be increasing because although there are three consecutive 0's in row 3, neither of the patterns 1011 or 0111 exist in row 4.

Step 5: Beginning with the subword $w_4^{\downarrow}w_6^{\uparrow}w_5^{\uparrow}w_3^{\downarrow}$ and required patterns [(1, rows 2-7), (11, rows 3-7), (011, rows 3-6), (0110, rows 3-5)]), we see that there are two consecutive 1's in row 5 and two consecutive 0's in row 3. But then we see that the only way to extend the pattern 0110 in rows 3 and 4 is to add a 0: 01100, so w_2 must be decreasing and appear to the right of the existing subword. (There is not a way to extend the pattern 0110 that is consistent in rows 4 and 5.) So we conclude this step with the subword ($w_4^{\downarrow} w_6^{\uparrow} w_5^{\uparrow} w_3^{\downarrow} w_2^{\downarrow}$, [(1, rows 2-7), (11, rows 3-7), (011, rows 3-6), (0110, rows 3-5), (01100, rows 3-4)]). This also determines w_1, which appears to the right of w_2, see Figure 7.

Thus the TSO word corresponding to this weaving diagram is

$$6543 \cdot 123456 \cdot 12345 \cdot 432 \cdot 43 \cdot 4,$$

where the word 6543 is obtained from 4321 by incrementing each letter by 2, as is required by Definition 8.