**Advances in
Applied Clifford Algebras**

Check for
updates

Machine Learning Discovers Invariants of Braids and Flat Braids

Alexei Lisitsa, Mateo Salles and Alexei Vernitski*

Abstract. We use machine learning to classify examples of braids (or flat braids) as trivial or non-trivial. Our machine learning takes the form of supervised learning, specifically multilayer perceptron neural networks. When they achieve good results in classification, we are able to interpret their structure as mathematical conjectures and then prove these conjectures as theorems. As a result, we find new invariants of braids and prove several theorems related to them. This work evolves from our experiments exploring how different types of AI cope with untangling braids with 3 strands, this is why we concentrate mostly on braids with 3 strands.

1. Introduction

The authors' recent research is in a type of artificial intelligence (AI) known as explainable AI; as applied to detecting trivial braids, which is the main topic of this article, this means that we used AI to produce a specific sequence of Reidemeister moves that untangles a braid [13,14]. We also applied explainable AI to knots and used more abstract algebraic constructions, instead of Reidemeister moves, to detect trivial knots, with AI producing a proof demonstrating, in all detail, that a knot is trivial [5,6,15]. Related recent research by other authors is [7,12]. In this study we use a different approach; let us describe it from the point of view of machine learning (ML), on the one hand, and from the point of view of mathematics, on the other hand.

From the ML perspective, interpretability of ML models is an important area of research. Our study is a positive example in this direction. We curate our ML models, keeping them small and simple; this enables us to manually inspect details of neural networks in our ML models; hence, we manage to

*Corresponding author.

🎂 **Birkhäuser**

notice some patterns in the trained ML models; based on them, we formulate mathematical conjectures and then prove them as theorems.

From the mathematical perspective, automated discovery of theorems or, in other words, AI-assisted conjectures is an increasingly important direction of research in recent years, see, for example, [3,4,19]. In this paper we describe how our experiments with applying neural networks to braids and flat braids have enabled us to form conjectures, which we then were able to prove as Theorems 1, 2, 3. The results of this study are an inspiration to use supervised learning to formulate new conjectures, as part of mathematical research.

2. Braids: Definitions and Encodings

Braids have connections with a number of other mathematical constructions, including knots, groups [10] and Clifford algebras [11].

This paper concentrates on braids with 3 strands (unless stated otherwise); an example of such a braid is shown in Fig. 1. We visualize a braid as stretched from the left to the right. Denote positions of strands in the braid by $1, 2, 3$ from the top to the bottom. Recall that the clockwise half-turn swapping the positions of two adjacent strands in positions $i, i+1$ in a braid is denoted by σ_i [10, Section 1.2.4]; thus, the braid in Fig. 1 is $\sigma_1\sigma_2\sigma_1^{-1}\sigma_2^{-1}$. Denote the number of crossings in a braid or, equivalently, the number of generators σ_i featuring in the word describing the braid, by k; for example, for the braid in Fig. 1 we have $k = 4$. In this study we use neural networks to work with braids, therefore, it is convenient to encode braids as matrices. We consider 4 encodings of braids as matrices, denoted below by EP2, EP1, ES2, ES1. In each of the encodings, the braid is represented by a matrix containing k columns. Each column in the matrix describes one crossing in the braid, in the same order (from the left to the right) as in the braid. In some of the encodings below, there are exactly 2 non-zero entries in each column, and in the others, there is exactly 1 non-zero entry in each column; the numbers in the notation of encodings EP2, EP1, ES2, ES1 correspond to these values 2 or 1.
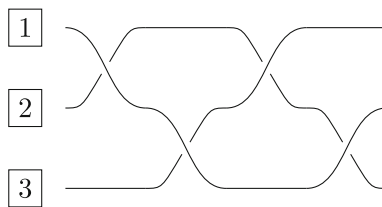

FIGURE 1. An example of a braid to illustrate encodings

Encodings Based on Positions (EP)

Encoding EP2 The matrix has 3 rows. Crossings of the type σ_1, σ_1^{-1}, σ_2, σ_2^{-1}, are denoted by columns $\begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}$, $\begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix}$, $\begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix}$, $\begin{pmatrix} 0 \\ -1 \\ 1 \end{pmatrix}$, respectively. The entry 1 (-1) in these columns indicates the position of the strand that passes over (under) the crossing. For example, the braid in Fig. 1 is encoded as $\begin{pmatrix} 1 & 0 & -1 & 0 \\ -1 & 1 & 1 & -1 \\ 0 & -1 & 0 & 1 \end{pmatrix}$.

Encoding EP1 The matrix has 2 rows. Crossings of the type σ_1, σ_1^{-1}, σ_2, σ_2^{-1}, are denoted by columns $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$, $\begin{pmatrix} -1 \\ 0 \end{pmatrix}$, $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$, $\begin{pmatrix} 0 \\ -1 \end{pmatrix}$, respectively. Thus, non-zero entries in the matrix indicate the positions of crossings and their signs. The braid in Fig. 1 is encoded as $\begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix}$.

Encodings Based on Strands (ES). Denote the strands of the braid by $1, 2, 3$, according to their positions on the left-hand side of the braid. For example, strand 1 in Fig. 1 is the curve that begins in position 1, then moves to position 2, then to position 3, and then back to position 2.

Encoding ES2 The matrix has 3 rows, corresponding to strands $1, 2, 3$, respectively. The column describing a crossing contains 1 in the position corresponding to the strand passing over the crossing and -1 in the position corresponding to the strand passing under the crossing. The braid in Fig. 1 is encoded as $\begin{pmatrix} 1 & 1 & 0 & 1 \\ -1 & 0 & -1 & -1 \\ 0 & -1 & 1 & 0 \end{pmatrix}$.

Encoding ES1 The matrix has 3 rows; row 1 (row 2, row 3) is used to record crossings of strands 1 and 2 (strands 2 and 3, strands 3 and 1). The entry 1 (or -1) indicates that the former strand passes above (or below) the latter strand. The braid in Fig. 1 is encoded as $\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix}$. Note that unlike the other three encodings, which can be easily applied to represent braids with any other number of strands, encoding ES1 does not easily generalize to braids with another number of strands.

If a permutation is represented as a product of adjacent transpositions, it can be shown using a braid-like diagram called a flat braid. For example, Fig. 2 shows a decomposition of permutation (2 3 1) into a product of adjacent transpositions (1 2)(2 3)(1 2)(2 3). All encodings of braids as matrices introduced above can be applied to encoding permutations as matrices, with the only difference that all non-zero entries in the matrices are 1 instead of either 1 or -1. Thus, for instance, the encoding ES2 of the flat braid in Fig. 2 is $\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$.
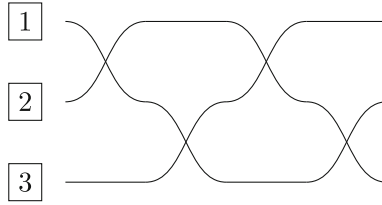
FIGURE 2. An example of a flat braid to illustrate encodings

For a given braid, we will denote the matrices encoding this braid in encodings EP2, EP1, ES2, ES1 by $[EP2]$, $[EP1]$, $[ES2]$, $[ES1]$, respectively.

As to encodings EP2 and EP1, they can be easily transformed into each other; namely, $[EP1] = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} [EP2]$ and $[EP2] = \begin{pmatrix} 1 & 0 \\ -1 & 1 \\ 0 & -1 \end{pmatrix} [EP1]$. Since EP2 and EP1 can be transformed into one another by matrix multiplication, it does not make a difference for an MLP which one to use. Out of EP2 and EP1, we prefer encoding EP1 because it has a slightly smaller matrix (with 2 rows) and more importantly, using it produces a slightly more readable version of Theorem 2. Thus, below we use only EP1 and not EP2.

As to encodings ES2 and ES1, it is possible to transform ES1 into ES2 using matrix multiplication, namely, $[ES2] = \begin{pmatrix} 1 & 0 & -1 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix} [ES1]$. However, it is not possible to transform ES2 into ES1 using matrix multiplication; accordingly, we will see, informally speaking, that ES1 can convey 'slightly more information' to the MLP than ES2. We will discuss this difference between ES2 and ES1 in Sects. 7 and 8. We find ES2 more readable than ES1, since ES2 explicitly refers to specific strands; therefore, when possible, we use ES2; when ES2 is 'too weak', we explicitly say that we switch to ES1.

In computational experiments below, we frequently speak about generating random braids. By this we mean that the number of crossings k is fixed, and then a random braid, in encoding EP1, is produced as a matrix of size 2 by k, in which each column each chosen, at random, out of the following possibilities $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$, $\begin{pmatrix} -1 \\ 0 \end{pmatrix}$, $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$, $\begin{pmatrix} 0 \\ -1 \end{pmatrix}$.

Unlike EP, when one uses ES, one cannot just place randomly chosen columns next to each other to produce a braid, because not every matrix formed of arbitrarily chosen columns permitted in ES2 (or ES1) corresponds to a braid. Let us say that $[ES2]$ (or $[ES1]$) is *realizable* if it corresponds to a braid. For example, $[ES2] = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}$ does not correspond to any flat braid, so it is not realizable.

Recall [10, Theorem 1.6] that two braids are equal in the braid group if and only if they can be transformed into each other using two types of operations called the second Reidemeister move (R2) and the third Reidemeister move (R3). (For braids on more than 3 strands, other types of operations also need to be used in addition to R2 and R3.) Let us remind ourselves what R2 and R3 are. In terms of generators σ_i, R2 means removing from a braid or inserting into a braid two consecutive letters $\sigma_i \sigma_i^{-1}$ or $\sigma_i^{-1} \sigma_i$. The move R3 means locating in a braid three consecutive letters $\sigma_i^x \sigma_{i+1}^y \sigma_i^z$ and replacing them with $\sigma_{i+1}^z \sigma_i^y \sigma_{i+1}^x$, or vice versa. Note that we are allowed to apply R3 only if $x, -y, z$ are not all equal to each other.

In terms of EP1, R2 means removing from $[EP1]$ or inserting into $[EP1]$ two consecutive columns of the form $\begin{pmatrix} 1 & -1 \\ 0 & 0 \end{pmatrix}$ or $\begin{pmatrix} -1 & 1 \\ 0 & 0 \end{pmatrix}$ or $\begin{pmatrix} 0 & 0 \\ 1 & -1 \end{pmatrix}$ or $\begin{pmatrix} 0 & 0 \\ -1 & 1 \end{pmatrix}$. The move R3 means locating in $[EP1]$ three consecutive columns of the form $\begin{pmatrix} x & 0 & z \\ 0 & y & 0 \end{pmatrix}$ or $\begin{pmatrix} 0 & y & 0 \\ x & 0 & z \end{pmatrix}$ and changing these columns to $\begin{pmatrix} 0 & y & 0 \\ z & 0 & x \end{pmatrix}$ or $\begin{pmatrix} z & 0 & x \\ 0 & y & 0 \end{pmatrix}$, respectively. Note that we are allowed to apply R3 only if $x, -y, z$ are not all equal to each other.

In terms of ES2, R2 means removing from $[ES2]$ or inserting into $[ES2]$ two consecutive columns in which one of the rows is $(1\ 1)$, another is $(-1\ -1)$, and the third is $(0\ 0)$. The move R3 means locating in $[ES2]$ three consecutive columns in which one of the rows is $(x\ -y\ 0)$, another is $(-x\ 0\ z)$, and the third is $(0\ y\ -z)$ and swapping the positions of the first and the third of these columns. Note that we are allowed to apply R3 only if x, y, z are not all equal to each other.

Similarly, in terms of ES1, R2 means removing from $[ES1]$ or inserting into $[ES1]$ two consecutive columns in which one of the rows is $(1\ 1)$ or $(-1\ -1)$, and the other two are $(0\ 0)$. The move R3 means locating in $[ES1]$ three consecutive columns in which one of the rows is $(x\ 0\ 0)$, another is $(0\ 0\ z)$, and the third is $(0\ y\ 0)$ and swapping the positions of the first and the third of these columns. Note that we are allowed to apply R3 only if x, y, z are not all equal to each other.

For flat braids, R2 and R3 are defined in the same way, only, obviously, 1 is used everywhere instead of -1, and the condition on x, y, z not being equal to each other is dropped.

A braid is called *trivial* if it is equal to the empty braid (that is, the braid with $k = 0$) in the braid group or, equivalently, if it can be transformed into the empty braid using a sequence of applications of R2 and R3. A property of a braid is called an *invariant* if it is preserved by applications of R2 and R3; for example, being trivial is an invariant.

A flat braid is trivial if and only if the permutation expressed by the flat braid is the identity permutation. There is a natural mapping (actually, a group homomorphism [10, Section 1.2.4]) from the set of braids to the set of flat braids, acting by "forgetting" the signs of the crossings. For example,

the flat braid in Fig. 2 is the image of the braid in Fig. 1. If the flat braid corresponding to a braid is a trivial flat braid then the braid is called *pure*.

3. Details of AI Used

In this paper we consider a problem of *supervised learning* of the binary classifiers of some properties of braids and flat braids. We use the classical model of multilayer perceptron (MLP) [16,17] and its implementation in WEKA Workbench for Data Mining [21]. In terms of machine learning, a MLP is a kind of a feedforward neural network models which supports supervised learning using *backpropagation* [17]. It is known to be an *universal approximator* [2,9]. MLP implementation in WEKA supports *sigmoid* activation function [8]. More precisely, in our experiments we have used *single* intermediate (hidden) layer version of MLP for *binary* classification. The result of the training, that is learned model, can be defined in this particular case as follows. Let n be a length of the input (size of the input layer of neurons) and m be a size of the single intermediate layer. Then a trained model (a perceptron) is defined by 1) a matrix $W_1 \in \mathbb{R}^{n \times m}$ of *weights* w_{ij}^1 of links between every input neuron and every neuron in a hidden layer; 2) a matrix $W_2 \in \mathbb{R}^{m \times 2}$ of *weights* w_{ij}^2 of links between every neuron in hidden layer and each of two *output* neurons (labelled by 0 and 1 respectively); 3) a vector $T_1 \in \mathbb{R}^m$ of *threshold* values t_i^1, one for each neuron in hidden layer; 4) a vector $T_2 \in \mathbb{R}^2$ of *threshold* values t_i^2, one for each output neuron. Given a trained model $TM = \langle W_1, W_2, T_1, T_2 \rangle$ the *classification* function $cf_{TM} : \mathbb{R}^n \to \{0,1\}$ implemented by TM is defined as follows. For the input $\bar{x} \in \mathbb{R}^n$ the output functions of neurons in a hidden layer are defined as $o_j^1(\bar{x}) = sign(\sum_{i=1}^n w_{ij}^1 \cdot x_i + t_j^1)$, $j = 1, \ldots m$, where $sign(z) = \frac{1}{1+e^{-z}}$ is sigmoid function. Similarly, the output functions of output neurons are defined as $o_j^2(\bar{x}) = sign(\sum_{i=1}^m w_{ij}^2 \cdot o_i^1(\bar{x}) + t_j^2)$, $j = 1, 2$. Finally,

$$cf_{TM}(\bar{x}) = \begin{cases} 0 & \text{if } o_1^2(\bar{x}) \geq o_2^2(\bar{x}) \\ 1, & \text{otherwise} \end{cases}$$

The supervised learning is organized as follows. Given a (data) set $D \in \mathbb{R}^n \times \{0,1\}$. D is randomly split into disjoint subsets $D = D_{train} \cup D_{test}$. A training procedure using backpropagation is applied to D_{train} with the aim to obtain trained MLP model TM with a classification function cf_{TM}, which is a good approximation for a function f_D: $f_D(\bar{x}) = y \Leftrightarrow (\bar{x}, y) \in D$. An obtained model is evaluated by measuring its performance on the test set D_{test} using standard notions of *true positive, false positive, true negative* and *false negative* samples. As the measure of performance, we report in the tables *weighted average precision* for all experiments conducted. Weighted average precision (WAP) for the case of binary classification considered in this paper is defined as follows. For each class/label (0 and 1 in binary case) we calculate its recognition precision as `Pr =  TP/(TP+FP)` (TP is a number of true positive and FP is number of false positive cases). Then `WAP` $= w_0 \times$

PrO $+ w_1 \times$ Pr1; $w_0 = |S_0|/|S_0| + |S_1|$ and $w_1 = |S_1|/|S_0| + |S_1|$, where S_0 and S_1 are sets of testing samples with labels 0 and 1, respectively.

Furthermore, we report the results after the application of the standard *k-fold cross validation procedure* (for $k = 2, 3, 4$), meaning that the whole data set D is split into k approximately equal in size disjoint subsets $D = D_1 \cup \ldots \cup D_k$. Then learning and testing procedure are repeated k times, by using each of $D \setminus D_i$ for training and D_i for testing. The performance measure (WAP) is then averaged over k cases. In the tables we indicate for each k an approximate size of a training set in % relative to the size of the whole dataset.

In our experiments for most of the WEKA settings for MLP model we have used default values (see them in Sect. 12), with the exception of the number of neurons in the hidden layer (denoted by H in WEKA); we needed to vary H to make the list of weights of the MLP reasonably small to enable us to interpret the numbers. For each of the reported experiments below, we will present the value of H, which include among others a default value denoted by a and defined as $a =$ [number of attributes + number classes]/2.

4. Discovering Theorems on Braids

4.1. Experiments

Recall that we work with braids on 3 strands. In the first series of experiments we considered datasets of random braids and studied learnability of the property of braids to be trivial using MLPs in standard setting, as described in Sect. 3. Note that all datasets we used were balanced, that is, they contained an equal number of trivial and non-trivial braids.

Table 1 presents the results of one such experiment. Weighted Average Precision is shown for various numbers H of neurons in a single hidden layer. The dataset of 8000 randomly generated braids (4000 trivial and 4000 non-trivial) on 3 strands and of lengths 12 is used for training. As one can see from the table, MLP can learn the property with very high precision, in excess of 99% even for small hidden layers containing only $H = 2$ neurons.

TABLE 1. Weighted average precision of classifying braids on 3 strands of length 12 as trivial or not, using an MLP with H hidden neurons

H	50% /2-fold	67%/3-fold	75% /4-fold
1	0.826	0.735	0.709
2	0.992	0.957	0.964
3	0.993	0.993	0.992
4	0.991	0.972	0.992
5	0.990	0.992	0.992
a=19	0.990	0.992	0.992

Encoding ES2. Dataset of 8000 braids

TABLE 2. Weighted average precision of classifying braids on 3 strands of length 12 as trivial or not, using an MLP with H hidden neurons

H	50%/2-fold	67%/3-fold	75%/4-fold
1	0.508	0.576	0.795
2	0.902	0.908	0.795
3	0.904	0.772	0.902
4	0.898	0.908	0.904
5	0.899	0.907	0.900
a=13	0.871	0.889	0.890

Encoding EP1. Dataset of 8000 braids

TABLE 3. Weighted average precision of classifying braids on 3 strands of length 12 as trivial or not, using an MLP with H hidden neurons

H	50% /2-fold	67%/3-fold	75% /4-fold
1	0.607	0.777	0.512
2	0.721	0.899	0.688
3	0.881	0.898	0.889
4	0.875	0.897	0.889
5	0.859	0.892	0.854
a=13	0.894	0.886	0.889

Encoding EP1. Dataset of 8000 braids satisfying CES2

We conducted similar experiments in encoding EP1. The property is also learnable with this encoding, but accuracy was not so high as can be seen from Table 2. (For generalising these observations to longer braids, see Table 8.)

Due to the very high accuracy of the MLPs working with encoding ES2, presented in Table 1, we manually inspected the weights of these MLPs trying to find useful mathematical patterns. As we will describe in Sect. 4.2, we were able to derive a conjecture from the weights of the MLPs classifying braids in encoding ES2. We proved this conjecture as Theorem 1. The condition which we extracted from the weights in the MLPs and which is a necessary condition for a braid to be trivial will be denoted by CES2.

In the second series of experiments we explored if in addition to the condition CES2 being satisfied, AI can suggest another condition which will improve classification of braids which are not necessarily classified correctly by CES2. We considered datasets of random braids satisfying condition CES2 in encoding EP1 and explored learnability of the property of braids to be trivial. Table 3 presents the results of one such experiment.

As we will describe in Sect. 4.2, we were able to derive a conjecture from inspecting the weights in MLPs classifying braids in datasets of random

braids satisfying condition CES2 in encoding EP1. We proved this conjecture as Theorem 2. The condition which we extracted from the weights in the MLPs and which is a necessary condition for a braid to be trivial will be denoted by CEP.

4.2. Proving the Conjectures Inspired by MLPs' Weights

When we considered weights of MLPs in experiments presented in Table 1, we saw that consistently they all have approximately the same absolute value, and their signs, $+$ or $-$, alternate. See an example shown in Table 11 in Sect. 13. Based on this pattern, we formulated a conjecture regarding trivial braids. Below we prove this conjecture as Theorem 1.

For a number x, by $[x]_{m \times n}$ we will denote a matrix of size m by n in which every entry is equal to x. By $[\pm]_{m \times n}$ we will denote a matrix of size m by n in which an entry at the position i, j is 1 (or -1) if $i + j$ is even (or odd).

Let $[ES]$ stand for $[ES2]$ or $[ES1]$. We will say that $[ES]$ satisfies condition CES if the alternating sum of each row is 0, that is, $[ES][\pm]_{k \times 1} = [0]_{3 \times 1}$. If we need to stress which one of $[ES2]$ or $[ES1]$ is tested, we will use notation CES2 or CES1.

Theorem 1. *Consider a braid on* 3 *strands. Let ES stand for* $[ES2]$ *or* $[ES1]$.

(1) *The list of alternating sums of each row, that is, the column* $[ES][\pm]_{k \times 1}$ *is a braid invariant.*

(2) *If a braid is trivial then condition CES is satisfied.*

Proof. 1. Considering how R2 and R3 act on $[ES2]$ (see Sect. 2), we can see that applying R2 or R3 does not affect the alternating sums of the rows. Note that applying R2 changes the value of k, making the braid 2 crossings shorter or 2 crossings longer, but the alternating sums of the rows are not affected.

2. If a braid is trivial then it can be transformed into the empty braid using R2 and R3. As we established in 1), applying R2 and R3 does not change the alternating sums of the rows, and the alternating sums for the empty braid are $[0]_{3 \times 1}$. Therefore, the alternating sums of the rows are $[0]_{3 \times 1}$ for each trivial braid.

The proof for $[ES1]$ can be constructed in the same way as 1), 2) above. □

Note that the invariant described in the theorem above cannot be easily generalised to braids with more than 3 strands. For a discussion, see the last paragraph of Sect. 6.

When we considered weights of MLPs in experiments presented in Table 3, we saw that consistently they all have approximately the same value. See an example shown in Table 12 in Sect. 13. Based on this pattern, we formulated another conjecture regarding trivial braids. Below we prove this conjecture as Theorem 2.

We will say that $[EP1]$ satisfies condition CEP if the sum of all entries in $[EP1]$ is equal to 0, that is, $[1]_{1 \times 2}[EP1][1]_{k \times 1} = 0$. Note that unlike

the previous theorem, the sum $[1]_{1\times2}[EP1][1]_{k\times1}$ featuring in the following theorem is not new; it is known in braid theory as the exponent sum of a braid, or the (signed) crossing number of a braid, or the writhe of a braid.

Theorem 2. *Consider a braid on 3 strands.*

(1) *The sum of all entries in* $[EP1]$, *that is,* $[1]_{1\times2}[EP1][1]_{k\times1}$ *is a braid invariant.*
(2) *If a braid is trivial then condition CEP is satisfied.*

*Proof.*The proof for $[EP1]$ can be constructed in the same way as the proof for $[ES2]$ in Theorem 1. □

Note that although we produced Theorems 1, 2 by inspecting relatively short braids with 12 crossings, in terms of classification of braids as trivial and non-trivial, they work well also for braids of much greater lengths; see Table 8.

5. Why Theorems 1, 2 are not "if and only if"

Conditions CES in Theorem 1 and CEP in Theorem 2 are necessary conditions for a braid being trivial, and for reasonably short braids, these are "almost sufficient" conditions. To give an example for braid length $k = 12$, Theorem 1 correctly classifies 99% of braids, and in addition to that, Theorem 2 correctly classifies 85% of the remaining 1% of braids.

However, after that, MLPs were not able to generate more conjectures which would complement CES and CEP, producing an "if and only if" result describing trivial braids. In this section we offer an information-theoretical explanation for not being able to describe trivial braids using MLPs with one hidden layer.

Indeed, suppose a braid on 3 strands is represented by a matrix M. Assume that like in our encodings, the number of columns in M is k, and the entries in M are in the range $-1, 0, 1$. Suppose the number of rows in M is 3. Suppose that we can multiply the entries of M by some coefficients (as we do in Theorems 1 and 2), and these coefficients are in the range $-r, -r+1, \ldots, r$, and the value of this matrix product tells us if the braid is trivial. In other words, for some linear function f with coefficients $-r, -r+1, \ldots, r$, we have $f(M) = 0$ if and only if the braid is trivial. Since only braids with even values of k can be trivial, we will assume that k is even. A rule that can decide if a braid b of length k is trivial can also be used to decide if two braids c, d of length $k/2$ are equal in the braid group (by checking if cd^{-1} is trivial). Here, the entries of the matrix corresponding to c are the first $k/2$ columns (let us denote them by C), and the entries of the matrix corresponding to d are the last $k/2$ columns (let us denote them by D). If we put separately the terms in f referring to C and to D, we can rewrite the equality $f(M) = 0$ as $g(C) = h(D)$, for some linear functions g and h. Now let us consider the total range of possible values of g (and h); it is easy to see that the value of g (or h) lies between $-3rk/2$ and $3rk/2$. Assuming that r is fixed and does not depend on the value of k (as it is fixed in conditions CES and CEP), we

see that the total range of possible values of g (and h) grows linearly with k. However, the number of classes of equal braids of length $k/2$ (in other words, the growth rate of the braid group B_3, or, equivalently, the growth rate of the fundamental group of the trefoil knot) grows faster than linearly[1] with k. This is why for some k there are more classes of equal braids of length $k/2$ than the number of possible values of g (or h). Therefore, it is impossible to decide if two braids c and d of length $k/2$ are equal in the braid group by checking the equality $g(C) = h(D)$. Therefore, it is impossible to decide if a braid b on 3 strands of length k is trivial by checking the equality $f(M) = 0$.

The same argument can also be applied to the scenario of using several linear functions $f_1, \ldots, f_n$; thus, we can show that one cannot decide if a braid is trivial by checking a fixed number of equalities $f_1(M) = 0, \ldots, f_n(M) = 0$. Hence follows a fact that we summarized in the beginning of this section, that Theorems 1 and 2 used together cannot correctly describe all trivial braids. Also, from this it follows that if we discover, in addition to CES and CEP, finitely many new conditions of the form $f_1(M) = 0, \ldots, f_n(M) = 0$, all these conditions together will form a necessary condition for a braid to be trivial, but not a sufficient condition.

If the growth rate of a group is not faster than linear then one can hope to find a rule describing trivial elements using an MLP. For example, in this paper in Sects. 6 and 9 we demonstrate that MLPs successfully form patterns of weights which correspond to correct descriptions of trivial flat braids on 3 strands (the group $\mathfrak{S}_3$ is finite, the growth rate is constant) and trivial braids on 2 strands (the group is infinite cyclic, the growth rate is linear).

On a related topic, one might ask what is the smallest example of a braid which satisfies conditions CES and CEP but is not trivial. The answer is that such examples are the braids whose closure is the link known as Borromean rings (see Figure 3 in [11]); one such braid is $\sigma_1\sigma_2^{-1}\sigma_1\sigma_2^{-1}\sigma_1\sigma_2^{-1}$.

6. Discovering theorems on flat braids

Our success with Theorems 1 and 2 has inspired us to explore if MLPs can form patterns of weights from which one can derive conjectures regarding permutations represented as flat braids. As you will see, for permutations (of 3 elements), Theorem 1 can be reformulated as an "if and only if" result, which exactly describes those permutations which are equal to the identity permutation.

Tables 4 and 5 report on the results of ML applied to the problem of triviality of flat braids on 3 strands using ES2 and EP1 encodings, respectively.

When we considered weights of MLPs in experiments presented in Table 4, we saw that consistently they all have approximately the same absolute value, and their signs, $+$ or $-$, alternate. (The fact that these weights are

---

[1]A simple explicit example of an exponential-size family of braids of length k which are pairwise distinct in B_3 can be produced for even values of $k = 2m$ by considering words $\prod_{i=1}^m \sigma_{a(i)}^2$ for $a \in \{1, 2\}^m$.

TABLE 4. Weighted average precision of classifying flat braids on 3 strands of length 12 as trivial or not, using an MLP with H hidden neurons

H	50%/2-fold	67%/3-fold	75%/4-fold
0	1.000	1.000	1.000
1	1.000	1.000	1.000
2	1.000	1.000	1.000
3	1.000	1.000	1.000
4	1.000	1.000	1.000
5	1.000	1.000	1.000
a = 19	1.000	1.000	1.000

Encoding ES2. Dataset of 2000 braids

TABLE 5. Weighted average precision of classifying flat braids on 3 strands of length 12 as trivial or not, using an MLP with H hidden neurons

H	50%/2-fold	67%/3-fold	75%/4-fold
0	0.493	0.500	0.505
1	?	0.595	0.544
2	0.533	0.723	0.655
3	0.692	0.713	0.722
4	0.645	0.882	0.742
5	0.860	0.857	0.808
a=13	0.926	0.984	0.917

Encoding EP1. Dataset of 2000 braids. "?" denotes the result of an experiment in which precision for non-trivial braids cannot be computed because they all have been misclassified as trivial

the same as those produced by MLPs presented in Table 1 is an unexpected coincidence, since numbers in ES2 for braids and ES2 for flat braids are not the same; indeed, in ES2 for braids there is one entry equal to 1 and one entry equal to -1 in each column, whereas in ES2 for flat braids there are two entries equal to 1 in each column.) From this pattern, we derive a conjecture regarding trivial flat braids, which we formulate and prove as Theorem 3.

The following observation is useful.

Lemma 1. *Consider a flat braid on 3 strands. Suppose a column c features more than once in $[ES2]$, say, in positions i and j, where $i < j$. Suppose none of columns in positions between i and j is equal to c. Then the difference $j - i$ is odd.*

Proof Say, $c = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$. This column represents an intersection of strand 1 and strand 2. Strands 1 and 2 cross each other at columns with positions i and j,

but not between them. Each crossing in columns with positions between i and j is either strand 3 entering the gap between strands 1 and 2 (for example, see the second crossing in the flat braid in Fig. 2) or strand 3 leaving the gap between strands 1 and 2 (for example, see the third crossing in the flat braid in Fig. 2). In total, between column i and column j, the number of times strand 3 enters and leaves the gap between strands 1 and 2 must be even, to ensure that strands 1 and 2 are adjacent to each other both in column i and in column j. Hence, $j - i$ is odd. □

Theorem 3. *Consider a flat braid on 3 strands. Let ES stand for $[ES2]$ or $[ES1]$.*

(1) *The list of alternating sums of each row, that is, the column $[ES][\pm]_{k \times 1}$ is a flat braid invariant.*
(2) *A flat braid is trivial if and only if condition CES is satisfied.*

Proof. The proof is the same for ES2 and ES1; below we use ES2.

Using the same plan of the proof as in Theorems 1, 2, one can prove that $[ES2][\pm]_{k \times 1}$ is a flat braid invariant and in a trivial flat braid, condition CES is satisfied.

Conversely, suppose condition CES is satisfied. Consider a case when a column, say, $c = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$ features more than once in the matrix, say, in positions i and j, where $i < j$. We can assume that none of columns in positions between i and j is equal to c, and no two columns between i and j are equal to one another; otherwise, change the choice of i and j and c, if needed. Since the same two strands (strand 1 and strand 2) cross each other at position i and position j, the difference $j - i$ is odd, see Lemma 1. If $j - i > 1$, note that columns $i, i+1, i+2$ are pairwise distinct, so we can apply R3 to these three columns, thus shifting column c to position $i+2$. Continue applying R3 as needed to shift the two columns c, that is, those columns which are originally in positions i and j, towards each other until they are in consecutive positions. Then apply R2 to remove both these columns.

By repeating the process described above as many times as needed, we end up with a flat braid in which each column features at most once. What flat braid is it? Recall that both the flat braid $[ES2]$ we started from and each of the flat braids we transformed it to satisfy the condition that the alternating sum of each row is 0. There is only a small number of flat braids in which each column features at most once; they are listed below (in the ES2 encoding).

$$\left(\right), \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}.$$

By inspecting all these flat braids, we can see that there is only one among them in which the alternating sum of each row is 0; this is the empty braid. Thus, $[ES2]$ can be transformed into the empty braid using R2 and R3, therefore, $[ES2]$ is trivial. □

From Theorem 3 it follows that the list of alternating sums of rows is not only an invariant, but a *complete* invariant of a flat braid, that is, two flat braids are equal as group elements if and only if they have the same list of alternating sums of rows. To say it explicitly, all 6 possible values of the list of alternating sums of rows in $[ES2]$ representing a flat braid on 3 strands are as follows (for convenience, we list them in the order corresponding to the list of flat braids above).

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix}, \begin{pmatrix} 0 \\ 2 \\ 0 \end{pmatrix}.$$

Based on this observation, we can formulate a simple algorithm for testing if a matrix $[ES2]$ is realizable as a flat braid. Let us denote by $[ES2]_m$ the matrix produced from $[ES2]$ by keeping only the leftmost m columns and discarding other columns.

Proposition 1. *A matrix $[ES2]$ is realizable as a flat braid if and only if for each $m = 1, \ldots, k$ the list of alternating sums of $[ES2]_m$ is equal to one of the 6 columns above.*

Proof If $[ES2]$ is realizable then each $[ES2]_m$ is realizable. Conversely, if $[ES2]$ is not realizable then there is m such that $[ES2]_{m-1}$ is realizable but $[ES2]_m$ is not realizable, and it can be checked directly that the fact that $[ES2]_m$ is not realizable due to having a forbidden last column must be reflected in a "faulty" value of the invariant of $[ES2]_m$ which is not one of the 6 listed above. Hence the result follows. □

In keeping with the spirit of this article, we have also checked if MLPs can detect realizable flat braids (in ES2 or ES1). In our experiment, MLPs learned surprisingly well to detect realizable flat braids, but we were not able to produce a meaningful interpretation of their weights, so we could not formulate a mathematical conjecture based on them. See more details in Sect. 13.

We have also attempted to experiment with flat braids on 4 strands (in other words, permutations on 4 elements), and MLPs do not produce any patterns from which we can derive conjectures and theorems. To remind the reader, in flat braids on 4 elements, apart from moves R2 and R3, one also is expected to use the move looking as $(12)(34) = (34)(12)$ in the notation of adjacent transpositions; in ES2, this is the move that replaces 2 consecutive columns $\begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}$ by $\begin{pmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \end{pmatrix}$ or vice versa. Obviously, the alternating sums of rows are not preserved by this move. Thus, if there are invariants of flat braids on 4 strands, they will have a form completely different to that in Theorem 3.

7. Pure Braids

Recall that we work with braids on 3 strands. In Theorem 1 condition CES is shown to be a necessary condition for a braid to be trivial, and in Theorem 3 condition CES is shown to be equivalent to a flat braid being trivial. Recall that a braid is called pure if the flat braid corresponding to it is a trivial flat braid. If you compare Theorems 1 and 3, you might ask if a braid is pure if and only if it satisfies CES. We discuss and answer this question in this section.

It is not true that every pure braid satisfies CES. Actually, a majority of pure braids do not satisfy CES; that is, if you fix an even braid length, approximately a third of these braids are pure[2], but only a tiny minority of these braids satisfy CES. The simplest specific counterexample is σ_1^2.

It is not true that if a braid satisfies CES2 then it is pure. The braid $\sigma_1\sigma_2\sigma_1$, with $[ES2] = \begin{pmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & -1 \end{pmatrix}$ is a counterexample.

However, it is true that if a braid satisfies CES1 then it is pure. The proof of this little fact is unexpectedly involved; we prove it in the following theorem.

Theorem 4. *If a braid satisfies condition CES1 then it is a pure braid.*

Proof For a braid $[ES1]$, let us denote the corresponding flat braid by $[ESf1]$; that is, the difference between $[ESf1]$ and $[ES1]$ is that in $[ESf1]$ all entries which are equal to -1 in $[ES1]$ are replaced by 1.

The proof will proceed as an expanded version of the proof of Theorem 3. We will simplify the braid applying the same Reidemeister moves (that is, Reidemeister moves affecting the same columns) to $[ES1]$ and $[ESf1]$. The type of Reidemeister moves we use will be slightly wider than valid braid Reidemeister moves on $[ES1]$ and slightly narrower than valid flat braid Reidemeister moves on $[ESf1]$. Namely, we will use Reidemeister moves which, on the one hand, preserve the alternating sums of columns of $[ES1]$ and, on the other hand, enable us to simplify $[ESf1]$ to obtain the empty flat braid. By successfully reducing $[ESf1]$ to the empty flat braid we will demonstrate that $[ESf1]$ is a trivial flat braid and, equivalently, $[ES1]$ is pure.

The result is true for the empty braid with $k = 0$. Now assume that $k \geq 1$. Then $[ESf1]$ contains a column, say, $c = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$. Since the alternating sum of row 1 in $[ES1]$ is 0, there must be at least one other column in $[ESf1]$ which is also equal to c. Let us list all column positions of columns which are equal to c in $[ESf1]$; they are $p_1, \ldots, p_m$. By Lemma 1, the difference $p_{i+1} - p_i$ is odd for each i. What would happen if in $[ES1]$ every column at position p_i for odd values of i was c and every column at position p_i for even values of i was $-c$; or what would happen if every column at position

---

[2]To be precise, there are 4^k braids and $\frac{2^k(2^k+2)}{3}$ pure braids of length k.

p_i for odd values of i was $-c$ and every column at position p_i for even values of i was c? Then the alternating sum of row 1 in $[ES1]$ would be m or $-m$ and not 0. Therefore, there are an odd i and an even j such that columns in positions p_i, p_j are both equal to c or both equal to $-c$. Therefore, for some i, columns in positions p_i, p_{i+1} are both equal to c or both equal to $-c$. Use the move R3 to shift these columns towards each other until they are in consecutive positions; perform the Reidemeister moves synchronously in $[ES1]$ and $[ESf1]$, and ignore the condition in the definition of R3 for braids which does not allow one to use it in a braid when x, y, z are equal to each other. Then apply R2 to remove the columns. Note that condition CES1 remains true for the modified $[ES1]$.

Applying the same process as many times as needed, we find a sequence of applications of R3 and R2 which transforms $[ESf1]$ into the empty flat braid. □

Corollary 1. *CES1 is a necessary condition for a braid to be trivial and a sufficient condition for a braid to be pure.*

8. The Gap Between CES2 and CES1

The braid $\sigma_1 \sigma_2 \sigma_1$, which we saw in Section 7, has $[ES2] = \begin{pmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & -1 \end{pmatrix}$ and $[ES1] = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix}$. Therefore, it is an example of a braid which satisfies condition CES2 but not CES1 (and not CEP).

Proposition 2. *If a braid satisfies condition CES1 then is satisfies condition CES2.*

Proof. This follows from our observation that $[ES2] = \begin{pmatrix} 1 & 0 & -1 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix} [ES1]$. □

Lemma 2. *The sum of all entries in $[EP1]$ is equal to the sum of alternating sums of rows of $[ES1]$. In other words, $[1]_{1 \times 2}[EP1][1]_{k \times 1} = [1]_{1 \times 3}[ES1][\pm]_{k \times 1}$*

Proof. Let us denote the columns that might feature in [ES1] by c_1, $-c_1$, c_2, $-c_2$, c_3, $-c_3$, where $c_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$, $c_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$, $c_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$. It is easy to prove by induction on the length of the braid that the leftmost column position in which $\pm c_1 (\pm c_2, \pm c_3)$ can feature has an odd (odd, even) position in the braid. As stated in Lemma 1 and discussed in the proof of Theorem 4, each next column featuring $\pm c_1 (\pm c_2, \pm c_3)$ appears an odd number of positions after the previous column featuring $\pm c_1 (\pm c_2, \pm c_3)$.

Thus, c_1 in an odd position contributes 1 to the alternating sums of $[ES1]$ and, being a clockwise half-turn, also 1 to $[EP1]$, whereas c_1 in an even position contributes -1 to the alternating sums of $[ES1]$ and, being an anticlockwise half-turn, also -1 to $[EP1]$. As to $-c_1$, $-c_1$ in an odd position contributes -1 to the alternating sums of $[ES1]$ and, being an anticlockwise half-turn, also -1 to $[EP1]$, whereas $-c_1$ in an even position contributes 1 to the alternating sums of $[ES1]$ and, being a clockwise half-turn, also -1 to $[EP1]$. The same argument also applies to c_2 and $-c_2$.

As to c_3 $(-c_3)$, recall that it is strand 3 crossing over (under) strand 1. Thus, $-c_3$ in an even position (see Fig. 1 for an example where $-c_3$ features in an even position) contributes 1 to the alternating sums of $[ES1]$ and, being a clockwise half-turn, also 1 to $[EP1]$, whereas $-c_3$ in an odd position contributes -1 to the alternating sums of $[ES1]$ and, being an anticlockwise half-turn, also -1 to $[EP1]$. Likewise, c_3 in an even position contributes -1 to the alternating sums of $[ES1]$ and, being an anticlockwise half-turn, also -1 to $[EP1]$, whereas c_3 in an odd position contributes 1 to the alternating sums of $[ES1]$ and, being a clockwise half-turn, also 1 to $[EP1]$.

Hence, the sum of all entries in $[EP1]$ is equal to the sum of alternating sums of rows of $[ES1]$. $\square$

Corollary 2. *If a braid satisfies CES1 then it satisfies CEP.*

Theorem 5. *If a braid satisfies CEP and CES2 then it satisfies CES1.*

Proof Suppose $[EP2]$ satisfies CEP. Considering the formula recalculating $[EP1]$ into $[EP2]$ presented in the proof of Proposition 2, we conclude that the alternating sums of rows in $[EP1]$ are equal to each other, that is, have a form $\begin{pmatrix} a \\ a \\ a \end{pmatrix}$ for some number a. Hence, using Lemma 2, we conclude that the sum of entries in $[EP1]$ is equal to $3a$. Since condition CEP is satisfied, $3a = 0$, hence, $a = 0$, hence, the alternating sums of rows in $[EP1]$ are all equal to 0. $\square$

9. Discovering Theorems on Braids on 2 Strands

The group B_2 of braids on 2 strands is isomorphic to the infinite cyclic group, so due to reasons described in Sect. 5 we were certain that MLPs can easily learn to classify braids on 2 strands as trivial or non-trivial both in ES and EP1. As expected, we were able to train ML models to achieve 100% precision in many settings. See Tables 6, 7 for the results on machine learning using ES2 and EP1 encodings, respectively.

10. An Application: An Algorithm for Finding Trivial Braids

The authors conduct research in applying deep learning to mathematical reasoning, and untangling trivial braids using Reidemeister moves is one of our favorite problems in this area [13,14]. This is why we regularly generate

TABLE 6. Weighted average precision of classifying flat braids on 2 strands of length 12 as trivial or not, using an MLP with H hidden neurons

H	50%/2-fold	67%/3-fold	75%/4-fold
1	0.830	0.609	0.685
2	1.000	0.723	0.881
3	1.000	1.000	0.950
4	1.000	1.000	1.000
5	1.000	1.000	1.000
a=13	1.000	1.000	1.000

Encoding ES2. Dataset of 1000 braids

TABLE 7. Weighted average precision of classifying flat braids on 2 strands of length 12 as trivial or not, using an MLP with H hidden neurons

H	50% /2-fold	67%/3-fold	75% /4-fold
1	0.631	0.635	0.595
2	0.820	1.000	0.881
3	1.000	1.000	0.899
4	1.000	1.000	1.000
5	1.000	1.000	1.000
a=7	1.000	1.000	1.000

Encoding EP1. Dataset of 1000 braids

datasets of random trivial braids to use for training or testing our AI agents. We use an implementation of an algorithm for checking triviality which we will denote by AUT; the details of AUT are of minor importance in this paper, but we will briefly present them in the next section.[3]

The following paragraphs summarize the experiments presented in more detail further in this section.

The easiest way to produce a dataset of uniformly sampled trivial braids is by generating random braids, one after another, and checking if they are trivial, until you have found as many trivial braids as you need. Since only a minority of braids are trivial, the task feels like finding the proverbial needle in a haystack. Theorems 1 and 2 provide new necessary conditions for a braid on 3 strands to be trivial, and these simple conditions can be checked very fast. Therefore, Theorems 1 and 2 can be used to quickly discard a braid if it is non-trivial, before the slower algorithm AUT is applied in those cases when Theorems 1 and 2 do not give a definite answer.

---

[3]When we were writing the section, we searched for other implementations of triviality-checking algorithms and could not find any which we could run easily. This is why in this section we show how the results of this paper can be applied to our algorithm AUT.

TABLE 8. How many braids, out of 10,000, satisfy conditions CEP, CES2, CES1, CEP & CES2, and how many are trivial (checked by algorithm AUT)

Length	CEP	CES2	CES1	CEP & CES2	AUT	Time (s)
10	2475	379	301	301	262	1.76
20	1731	168	103	103	64	2.91
30	1530	110	47	47	15	5.93
40	1273	90	36	36	11	25.19
50	1102	69	24	24	5	147.68

The column 'Time' shows the total time in seconds

Accuracy of Theorem 1 in detecting trivial braids is much higher than Theorem 2; this is why the MLPs have found the patterns expressed in Theorem 1 before those expressed in Theorem 2. However, encoding ES used in the condition of Theorem 1 takes longer to construct than EP, so in terms of time efficiency, it is slightly better to check the condition CEP from Theorem 2 first, and only if it is satisfied, check the condition CES from Theorem 1. This algorithm, which we will call CEP-CES-AUT, is faster than the combination CES-CEP-AUT which the MLPs seem to suggest.

Corollary 2 suggests that we can use the combination CES1-AUT, since CES1 implies CEP. However, discarding most non-trivial braids using condition CEP in combination CEP-CES-AUT is faster than building encoding ES1 for all braids and using combination CES1-AUT.

10.1. Many Braids can be Classified Using CES or CEP Instead of AUT

Recall that for each braid of length k, the majority of braids are non-trivial. As Table 8 shows, most of the non-trivial braids can be classified as non-trivial using conditions CEP or CES2 or CES1. Looking at this table, it is useful to remember that we proved that CES1 $\Rightarrow$ CES2, CES1 $\Rightarrow$ CEP, CEP & CES2 $\Leftrightarrow$ CES1, and the triviality implies all CEP, CES2 and CES1, see Theorems 1, 2, Proposition 2, Corollary 2 and Theorem 5.

For this experiment we generated 10,000 random braids of the same length on 3 strands and checked conditions CEP, CES2, CES1, CEP & CES2, and also triviality (using algorithm AUT) We repeated the experiment for lengths 10, . . . , 50.

10.2. Using CES and CEP Makes Checking Faster

In this experiment we generate 10,000 random braids and then for each braid we check if condition CEP is satisfied. If CEP is satisfied, condition CES2 is checked for this braid. Then, if CES2 is also satisfied, the braid is further checked using AUT. In Table 9, the columns CEP (CES, AUT) refer to the number of braids that satisfy CEP (satisfy both CEP and CES, are found to be trivial by AUT). As you can see, only a minority of braids, about 0.2–3% of braids need to be checked using AUT; indeed, we have already seen similar numbers in Table 8. The columns T CEP (T CES2, T AUT) correspond to

TABLE 9. The number of braids considered and time spent
when checking triviality using combination CEP-CES-AUT

Length	CEP	CES2	AUT	T CEP	T CES2	T AUT	TT
10	2456	279	248	0.029	0.057	0.011	0.097
20	1764	95	55	0.051	0.076	0.011	0.138
30	1456	54	20	0.073	0.092	0.022	0.186
40	1252	34	8	0.094	0.103	0.064	0.262
50	1122	24	3	0.116	0.115	0.741	0.972

the total time taken to check CEP (or CES2, or run AUT) for those braids
for which these checks were conducted. The column TT contains the total
time taken to check the triviality of the 10,000 braids.

As you can see, the total time in Table 9 is considerably smaller than
the total time in Table 8, because AUT is only run for a minority of braids
and not all braids.

As you scan the columns of Table 9, you can see that the proportion of
braids that need to be checked using AUT (column CES2) becomes smaller
as the braid length k grows, but the time taken to run AUT on these longer
braids becomes larger (column T AUT).

10.3. CEP-CES-AUT is the Fastest Combination

Theorems 1, 2, Proposition 2, Corollary 2 and Theorem 5 suggest several
sequences in which CEP, CES1, CES2 and AUT can be combined to check the
triviality, including AUT on its own, CEP followed by AUT, CES1 followed
by AUT, CES2 followed by CEP followed by AUT, and CEP followed by
CES2 followed by AUT. In the graph in Fig. 3 we compare how fast these
implementations are, and we see that combination CEP-CES2-AUT is the
fastest.

For this experiment, we use each of the algorithms from the list above,
for each value of braid length k in the range $10, 14, \ldots, 34$, to produce a
dataset consisting of 1000 trivial braids and 1000 non-trivial braids.

The line corresponding to CES1-AUT is only marginally higher than the
line corresponding to CES2-CEP-AUT. This is because CES1 is equivalent
to CES2 & CEP, the time needed for producing encoding ES1 is practically
the same (just slightly longer, in our implementation) as the time needed for
producing encoding ES2, and hardly any braids need to be checked for CEP
after CES2 has been checked.

10.4. Comparing AUT and CEP-CES-AUT on Individual Braids

In this experiment we generated 10,000 random braids of length 20 on 3
strands. We checked condition AUT on every braid and recorded the time
taken by AUT on this individual braid. These times are presented by the
green bar chart in Fig. 4, with the time (the horizontal axis) on the log scale;
to present the data more clearly, we show the same bar chart two times on
different scales, with a smaller size in (A) and with a magnified size in (B).
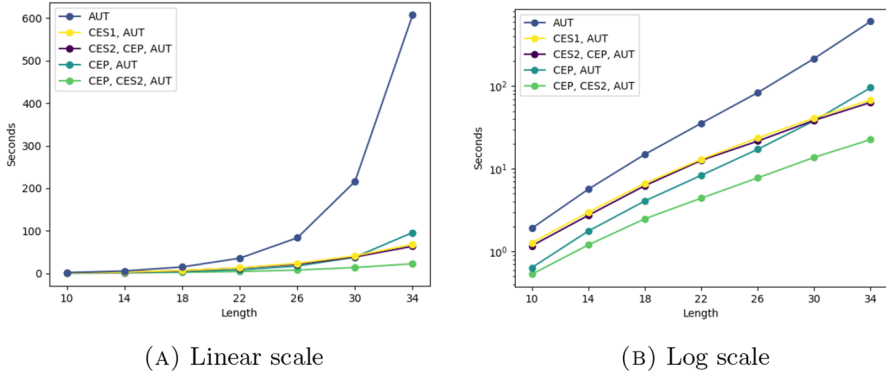
(A) Linear scale

(B) Log scale

FIGURE 3. Out of several ways of checking triviality, CEP-CES2-AUT is the fastest

(A) A plot showing all the data
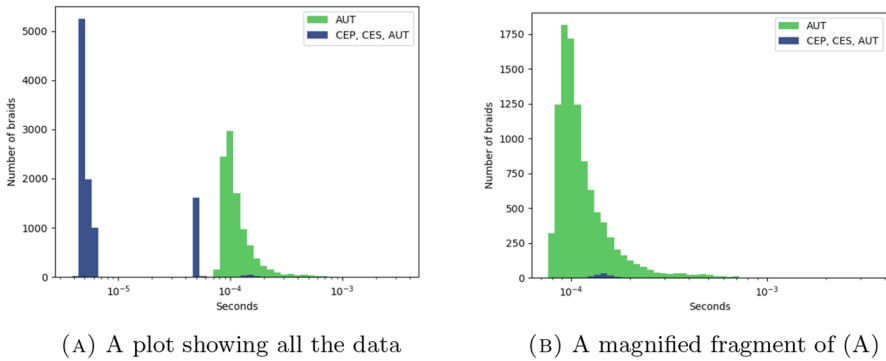
(B) A magnified fragment of (A)

FIGURE 4. Time to check the triviality taken by AUT (in green) and by CEP-CES-AUT (in blue)

On the other hand, we tested the triviality by checking conditions CEP, CES, AUT on the same braids and recorded the time for each braid. These times are presented on the same bar chart in blue.

Thus, the green chart shows a typical time it takes to check if a braid is trivial using AUT, which is about 10^{-4}. The blue chart shows a typical time it takes to check if a braid is trivial using CEP-CES-AUT. If you look at part (A), the high blue spike on the left-hand side, that is, at about $5 \cdot 10^{-6}$ on the horizontal coordinate, correspond to braids classified by CEP. The next blue spike, at about $5 \cdot 10^{-5}$, corresponds to braids classified by CES. Then the small blue hump at about 10^{-4} (it is easier to spot it in part (B)) corresponds to the braids which CEP and CES could not classify; to this minority of braids, AUT is applied.

10.5. How Much Search it Takes to Find Trivial Braids

In this experiment we produce a dataset consisting of 1000 pairwise distinct trivial braids. The aim of this exercise is to show how many braids one had

TABLE 10. How many braids need to be explored to find 1000 trivial braids

Length	EXP	CEP	CES	AUT	Time
10	42,999	10,515	1126	1021	0.574
20	167,694	29,596	1677	1000	3.306
30	515,663	74,347	2737	1000	13.689
40	1,241,628	155,568	4394	1000	43.899
50	2,796,519	313,277	6941	1000	155.987

to consider to find 1000 trivial braids, and how well our new algorithm CEP-CES-AUT copes with this task in terms of time. Table 10 shows how many braids we had to explore in total (column EXP) before we found 1000 trivial braids, how many of these braids satisfied condition CEP (column CEP), how many of these braids satisfied both CEP and CES and were then checked using AUT (column CES), and how many of these braids were found trivial by AUT (column AUT). For example, in order to produce 1000 pairwise distinct trivial braids of length 10 we had to consider $42,999$ random braids, out of which $10,515$ satisfied CEP, 1126 satisfied CEP and CES, and 1021 were found trivial by AUT; out of these 1021 braids, 21 braids were discarded because their duplicates had already been added to the dataset.

11. Details of Algorithm AUT

Algorithm AUT is based on Artin's solution of the word problem in the braid group via automorphisms of a free group (this is why we denote this algorithm by AUT). Algorithm AUT is exponential in the worst case, but it is fast in practice; for example, we used AUT to produce all words representing trivial braids on 3 strands up to length $k = 16$, see https://oeis.org/A354602. We are aware that there are algorithms for checking if a braid is trivial which have a better algorithmic complexity, polynomial instead of exponential, for example [1], but in our research practice, AUT works sufficiently fast, and in the context of this study, our conclusions regarding combining AUT with CEP and CES can be applied to any other algorithm instead of AUT.

The differences between our algorithm AUT and a naive implementation of Artin's solution of the word problem are as follows.

We do not use Artin's automorphisms of the free group, but automorphisms introduced in Theorem 1 (automorphism type 2) in [18]. The benefit of using these automorphisms is that in each word considered, each letter at an odd-numbered position is one of generators of the free group, and each letter at an even-numbered position is the inverse of one of generators of the free group; this useful property enables us to write especially simple computer code for simplifying words to reduced words. The way this type of automorphisms is used in AUT can be further improved using algebraic structures known as *keis* or *involutory quandles*. In the past, the authors of this paper

studied keis [5, 6, 15, 20]; keis can be informally described as groups considered with the operation $x \triangleright y = yx^{-1}y$ instead of the usual group operation. The automorphisms from [18] can be usefully represented as kei terms instead of group words, and these kei terms are considerably shorter than the corresponding group words (although they are still exponentially long in the worst case; the length of the kei terms grows, in the worst case, as Fibonacci numbers instead of 2^k for group words).

Instead of using kei terms, we obtain approximately the same effect in AUT by judiciously using dichotomy to ensure that we get an opportunity to simplify group words whenever possible, before they become very long. Namely, when dealing with a word b in the braid group, we decompose it into a product of words $b = cd$, where c, d have an equal (or almost equal) length, compute the automorphisms induced by c and d, inspect the description of these automorphisms and simplify the words in the free group to reduced words, and then multiply the automorphisms corresponding to c and d to produce the automorphism corresponding to b.

Acknowledgements

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

12. Appendix: Default settings of MLP in WEKA Workbench (version 3.8.3)

L = 0.3 *(learning rate)*

M = 0.2 *(momentum rate)*

N = 500 *(number of epochs to train)*

V = 0 *(percentage size of validation set to use to terminate training)*

S = 0 *(seed for Random Number Generator)*

E = 20 *(threshold for number of consecutive errors to use to terminate training)*

TABLE 11. Typical weights of a hidden-layer neuron in experiments presented in Table 1

-5.9	5.6	-5.7	5.6	-5.5	6.0	-5.7	5.9	-5.7	5.7	-5.7	6.1
6.6	-6.7	6.6	-6.7	6.6	-7.2	6.9	-6.7	6.7	-6.5	6.4	-6.9
-0.8	1.1	-0.8	1.0	-1.1	1.3	-1.2	0.7	-1.1	0.8	-0.9	0.9

TABLE 12. Typical weights of a hidden-layer neuron in experiments presented in Table 3

8.7	11.4	9.7	9.3	3.3	4.5	6.4	4.7	5.8	8.6	7.8	11.2
1.9	2.9	8.0	8.1	7.2	5.6	6.2	7.0	5.0	5.5	4.7	2.8

TABLE 13. An example of weights of a hidden-layer neuron in experiments aiming to classify flat braids in encoding ES2 as realizable or not

3.0	1.3	-4.2	1.6	-4.2	2.1	-4.9	3.2	-4.0	1.7	-4.4	1.5
-6.4	-11.1	8.2	-11.0	8.1	-10.1	7.5	-9.6	4.1	-6.9	4.2	-7.1
-0.7	5.7	-8.1	5.3	-8.0	4.0	-6.7	2.4	-4.1	1.0	-3.9	1.4

13. Appendix: Examples of weights in MLPs

It would be tedious and repetitive to list all weights, so we present examples of typical values of weights, namely, the weights of the links leading to a node in the hidden layer that is connected to the output layer with a link whose weight has the largest absolute value. In notation of Sect. 3 this means that we select a j-th column from matrix W_1, where j is the row in matrix W_2 where the largest absolute value features.

From the point of view of the MLP, matrix $[ES2]$ or $[EP1]$ is 'flattened' into a list of numbers $\bar{x}$. For better readability, we perform the opposite change and arrange the list of numbers which is a column of matrix W_1 in the shape of a matrix, thus showing exactly which weight value corresponds to each entry in the matrix $[ES2]$ or $[EP1]$.

We have chosen one experiment from Table 1, the one with $H = 5$ and a 50% training set; its weights are shown in Table 11.

We have chosen one experiment from Table 3, the one with $H = 4$ and a 50% training set; its weights are shown in Table 12.

We have chosen one experiment in which an MLP learned to classify flat braids in encoding ES2 as realizable or not, with $H = 5$ and 50% training set, on a dataset of 2000 flat braids of length 12; weighted average precision in this experiment is 0.969. The weights are shown in Table 13.

References

[1] Birman, J., Ko, K.H., Lee, S.J.: A new approach to the word and conjugacy problems in the braid groups. Adv. Math. **139**(2), 322–353 (1998)

[2] Cybenko, G.: Approximation by superpositions of a sigmoidal function. Math. Control Signals Syst. (MCSS) **2**(4), 303–314 (1989)

[3] Davies, A., Veličković, P., Buesing, L., Blackwell, S., Zheng, D., Tomašev, N., Tanburn, R., Battaglia, P., Blundell, C., Juhász, A., et al.: Advancing mathematics by guiding human intuition with ai. Nature **600**(7887), 70–74 (2021)

[4] Fawzi, A., Balog, M., Huang, A., Hubert, T., Romera-Paredes, B., Barekatain, M., Novikov, A., Ruiz, F.J.R., Schrittwieser, J., Swirszcz, G., et al.: Discovering faster matrix multiplication algorithms with reinforcement learning. Nature **610**(7930), 47–53 (2022)

[5] Fish, A., Lisitsa, A., Stanovskỳ, D.: A combinatorial approach to knot recognition. In: Embracing Global Computing in Emerging Economies: First Workshop, EGC 2015, Almaty, Kazakhstan, February 26–28, 2015. Proceedings 1, pp. 64–78. Springer (2015)

[6] Fish, A., Lisitsa, A.: Detecting unknots via equational reasoning, I: exploration. In: Intelligent Computer Mathematics: International Conference, CICM 2014, Coimbra, Portugal, July 7-11, 2014. Proceedings, pp. 76–91. Springer (2014)

[7] Gukov, S., Halverson, J., Ruehle, F., Sułkowski, P.: Learning to unknot. Mach. Learn. Sci. Technol. **2**(2), 025035 (2021)

[8] Han, J., Moraga, C.: The influence of the sigmoid function parameters on the speed of backpropagation learning. In: Mira, J., Hernández, F.S. (Eds.) IWANN, volume 930 of Lecture Notes in Computer Science, pp. 195–201. Springer (1995)

[9] Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Netw. **2**(5), 359–366 (1989)

[10] Kassel, C., Turaev, V.: Braid Groups, vol. 247. Springer Science & Business Media, Berlin (2008)

[11] Kauffman, L.H., Lomonaco Jr, S.J.: Braiding with majorana fermions. In: Quantum Information and Computation IX, vol. 9873, pp. 81–95. SPIE (2016)

[12] Kauffman, L., Russkikh, N., Taimanov, I.: Rectangular knot diagrams classification with deep learning. J. Knot Theory Ramif. **31**(11), 2250067 (2022)

[13] Khan, A., Vernitski, A., Lisitsa, A.: Untangling braids with multi-agent q-learning. In: 2021 23rd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), pp. 135–139. IEEE (2021)

[14] Lisitsa, A., Salles, M., Vernitski, A.: Supervised learning for untangling braids. In: Proceedings of the 15th International Conference on Agents and Artificial Intelligence—Volume 3: ICAART, pp. 784–789. INSTICC, SciTePress (2023)

[15] Lisitsa, A., Vernitski, A.: Automated reasoning for knot semigroups and π-orbifold groups of knots. In: Mathematical Aspects of Computer and Information Sciences: 7th International Conference, MACIS 2017, Vienna, Austria, November 15–17, 2017, Proceedings, pp. 3–18. Springer (2017)

[16] Rosenblatt, F.: Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan Books, Richmond (1961)

[17] Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. In: Rumelhart, D.E., Mcclelland, J.L. (eds.) Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations, pp. 318–362. MIT Press, Cambridge (1986)

[18] Shpilrain, V.: Representing braids by automorphisms. Int. J. Algebra Comput. **11**(06), 773–777 (2001)

[19] Vasylenko, A., Gamon, J., Duff, B.B., Gusev, V.V., Daniels, L.M., Zanella, M., Shin, J.F., Sharp, P.M., Morscher, A., Chen, R., et al.: Element selection for crystalline inorganic solid discovery guided by unsupervised machine learning of experimentally explored chemistry. Nat. Commun. **12**(1), 5561 (2021)

[20] Vernitski, A., Tunsi, L., Ponchel, C., Lisitsa, A.: Dihedral semigroups, their defining relations and an application to describing knot semigroups of rational links. Semigroup Forum **97**(1), 75–86 (2018)

[21] Witten, I.H., Frank, E., Hall, M.A., Pal, C.J.: The WEKA Workbench. Online Appendix for Data Mining, Fourth Edition: Practical Machine Learning Tools and Techniques. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2016)

Alexei Lisitsa
University of Liverpool
Liverpool
UK
e-mail: `lisitsa@liverpool.ac.uk`

Mateo Salles and Alexei Vernitski
University of Essex
Colchester
UK
e-mail: `asvern@essex.ac.uk`

Mateo Salles
e-mail: `mateosi98@gmail.com`