Ranks of elliptic curves and deep neural networks

Matija Kazalicki\textsuperscript{1*} and Domagoj Vlah\textsuperscript{2}

Abstract

Determining the rank of an elliptic curve $E/\mathbb{Q}$ is a difficult problem. In applications such as the search for curves of high rank, one often relies on heuristics to estimate the analytic rank (which is equal to the rank under the Birch and Swinnerton-Dyer conjecture). In this paper, we propose a novel rank classification method based on deep convolutional neural networks (CNNs). The method takes as input the conductor of $E$ and a sequence of normalized Frobenius traces $a_p$ for primes $p$ in a certain range ($p < 10^k$ for $k = 3, 4, 5$), and aims to predict the rank or detect curves of “high” rank. We compare our method with eight simple neural network models of the Mestre–Nagao sums, which are widely used heuristics for estimating the rank of elliptic curves. We evaluate our method on two datasets: the LMFDB and a custom dataset consisting of elliptic curves with trivial torsion, conductor up to $10^{30}$, and rank up to 10. Our experiments demonstrate that the CNNs outperform the Mestre–Nagao sums on the LMFDB dataset (remarkably, the neural network that took as an input all Mestre–Nagao sums performed much better than each sum individually). On the custom dataset, the performance of the CNNs and the Mestre–Nagao sums is comparable.

1 Introduction

Let $E$ be an elliptic curve over $\mathbb{Q}$ with discriminant $\Delta$ and conductor $N$. A celebrated theorem of Mordell states that the group of rational points $E(\mathbb{Q})$ is a finitely generated abelian group isomorphic to $E(\mathbb{Q})_{\text{tors}} \times \mathbb{Z}^r$, where $E(\mathbb{Q})_{\text{tors}}$ is the torsion subgroup of $E(\mathbb{Q})$ and $r$ is the rank of $E(\mathbb{Q})$. While the possible torsion subgroups were completely classified by Mazur [28] and easy to compute, the rank of the elliptic curve is a much more mysterious quantity. Not only is it not known which values can be obtained as ranks (although this question was already asked by Poincaré [36]), there is also no agreement on whether the rank is unbounded or not. Up until recently, the folklore conjecture was that the rank is unbounded which was challenged by the series of papers [34, 44, 45] that predict (based on different heuristic models) that there are only finitely many elliptic curves with rank greater than 21. The current rank record is 28 by Elkies [12]. For more information about the rank records for curves with fixed torsion subgroups as well for rank in families, see [11].
The search for curves of high rank is challenging, partly because determining the rank of an elliptic curve is a computationally expensive task. Also, there is no algorithm known to correctly compute the rank in all cases. This is mainly because finding rational points on elliptic curves is a difficult problem, and descent algorithms that are commonly used eventually reduce to a naive point search on some auxiliary curves. To overcome this difficulty, researchers use rank heuristics that are inspired by the Birch and Swinnerton-Dyer conjecture. These heuristics help in identifying probable candidates for elliptic curves of high rank, thereby reducing the computational burden of computing the rank of all curves.

For each prime of good reduction $p$, we define $a_p = p + 1 - \#E(F_p)$. For $p|N$, we set $a_p = 0, -1, 1$ if, respectively, $E$ has additive, split multiplicative or non-split multiplicative reduction at $p$. The $L$-function attached to $E/\mathbb{Q}$ is then defined as an Euler product

$$L_E(s) = \prod_{p|\Delta} \left(1 - \frac{a_p}{p^s}\right)^{-1} \prod_{p|\Delta} \left(1 - \frac{a_p}{p^s} + \frac{p^{1-2s}}{p^{2s}}\right)^{-1},$$

which converges absolutely for $\Re(s) > 3/2$ and extends to an entire function by the Modularity theorem [5, 46]. The Birch and Swinnerton-Dyer (BSD) conjecture states that the order of vanishing of $L_E(s)$ at $s = 1$ (the quantity known as analytic rank) is equal to the rank of $E(\mathbb{Q})$.

Mestre [29] and Nagao [31], and later others [1, 14], motivated by BSD conjecture, considered certain sums (see Sect. 2 for the list of sums examined in this paper) which heuristically should be able to detect curves of high analytic rank. In an abuse of terminology, we refer to all such sums as the Mestre–Nagao sums. For example, one of these sums (see Sect. 2 in [14])

$$\tilde{S}_5(B) = \sum_{p < B, \text{ good reduction}} \log \left(\frac{p + 1 - a_p}{p}\right)$$

has a property that $\exp(-\tilde{S}_5(B))$ is the partial product of $L_E(s)$

$$\prod_{p < B, \text{ good reduction}} \left(1 - a_p p^{-s} + p^{1-2s}\right)^{-1}, \quad (1.1)$$

evaluated at $s = 1$ (ignoring the primes of bad reduction). One expects that $\tilde{S}_5(B)$ should be large if $E$ has a large rank since then the partial product should rapidly approach zero. This sum was used in [14] as a first step in finding rank-record breaking curves with fixed cyclic torsion $\mathbb{Z}/n\mathbb{Z}$ for $n = 2, 3, \ldots, 7$.

Recently, some new fundamental results have been discovered with the assistance of deep neural networks in topology and representation theory [9], combinatorics [43], as well as some applications to problems in statistics [21, 22]. In number theory, the utility of machine learning methods was shown in [17, 18] where the authors, among other things, successfully used logistic regression for classifying elliptic curves of rank zero and one.

In this paper, we investigate a deep learning algorithm for rank classification based on convolutional neural networks (CNN). These networks take as an input the conductor of the elliptic curve together with the sequence of normalized $a_p$-s (i.e. $a_p/\sqrt{p}$) for $p$ in a fixed range and output the rank of the elliptic curve. We compare its performance to that of the Mestre–Nagao sums $\{S_0, S_1, \ldots, S_6, \Omega\}$ (defined in Sect. 2). A priori, it is not clear
how to decide on the rank of the elliptic curve based on the value of its Mestre–Nagao sum (see related question (1) in Sect. 7 of [14]), so we train a simple fully connected neural network to do that task for us. Since the answer critically depends on the conductor of the elliptic curve, these networks, besides the Mestre–Nagao sum, take the conductor of the elliptic curve as an input. Training these networks revealed the optimal cutoff of the specific Mestre–Nagao sum for rank classification (for cutoffs of $S_5$ see Fig. 3).

Architecture and the training process of our neural networks are described in Sect. 4. For training, we used two datasets. One is the LMFDB database [40] which contains 3,824,372 elliptic curves defined over $\mathbb{Q}$, divided into 2,917,287 isogeny classes, with rank between 0 and 5 and conductor less than 300,000,000. For more information about the structure of LMFDB database (it consists of three distinct datasets) see Sect. 3.1. The other one is a custom-made dataset that consists of 2,033,965 elliptic curves with trivial torsion, the conductor less than $10^{30}$, and the rank in the range between 0 and 10. Although this dataset is biased and does not represent typical elliptic curves from a given rank and conductor range (because its high-rank elliptic curves have low height generators, for discussion see Sect. 3), it is useful for comparing different classification methods.

For each neural network (the CNN or one of the Mestre–Nagao sums, in total 9) we have performed 24 tests by varying

(a) dataset—LMFDB or custom,
(b) range of $a_p$—we considered $(a_p)_{p \leq B}$ for $B = 10^3$, $10^4$, and $10^5$,
(c) test curves—uniformly selected (20% from the dataset) or all curves in the top conductor range (which is $[10^6, 10^9]$ for the LMFDB and $[10^{29}, 10^{30}]$ for the custom dataset),
(d) type of classification - binary or all ranks (for the LMFDB the rank range is from 0 to 5, and for the custom dataset from 0 to 10).

In binary classification, curves are labeled as either of low or high rank. For the LMFDB high rank means rank 4 (we did not consider 19 rank 5 curves), while for custom dataset high rank is 8, 9 or 10. Here, the idea is to test how good neural networks are in detecting elliptic curves of high rank.

The neural networks have been selected to maximize the Matthews correlation coefficient (MCC) or phi coefficient, which is generally regarded as a balanced measure of the quality of the classification even if the classes are of very different sizes [4], and thus prevents the classifier ignoring the small classes. This is relevant to our problem since elliptic curves of high rank are sparsely represented in our datasets (see Tables 2 and 4 for distribution of ranks in datasets). For binary classification, the MCC is computed using the following formula

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP) \cdot (TP + FN) \cdot (TN + EP) \cdot (TN + FN)}},$$

where $TP, FN, TN, FP$ denote, respectively, the number of true positives, false negatives, true negatives, and false positives. Note that $MCC$ lies in the segment $[-1, 1]$ and $MCC = 1$ only in the case of perfect classification.

For a comparison of the Mathews correlation coefficients of the CNN and $S_5$ classifiers for the uniform test set, see Table 1.

In all the tests performed on the LMFDB dataset, the CNN outperformed the Mestre–Nagao sums (this is especially true in the classification of all ranks). For example, in the all
Table 1  Comparison of the Matthews correlation coefficients of the CNN and $S_5$ all rank classifiers for the uniform test set

| Type of classifier | LMFDB | Custom dataset |
|--------------------|-------|----------------|
|                    |      |                |
| CNN                | 0.9507 | 0.9958  |
| $S_5$             | 0.6132 | 0.7774  |

ranks classification with $p < 10,000$ in the uniform range, the MCC of the CNN is 0.9958 (in particular, the CNN misclassified only 0.25% of the curves) while the best Mestre–Nagao sum $S_2$ has $MCC = 0.8697$ (it misclassified 8.1% of the curves). For more details see Table 6. Similarly, in the all ranks classification with $p < 10,000$ and top conductor range, the MCC of the CNN is 0.9289 (see confusion matrix in Fig. 2) while the best Mestre–Nagao sum $S_0$ has $MCC = 0.5057$ (see Fig. 4), which is remarkable!

We also trained a fully connected neural network whose inputs were all Mestre–Nagao sums {$S_0, \ldots, S_6$} (together with the conductor), and interestingly on the LMFDB dataset this network performed much better than any sum individually. For example, in the uniform range case above it attained $MCC = 0.9602$, and in the top range case $MCC = 0.7013$.

A binary classification on the LMFDB dataset (which tries to identify curves of rank 4) ended up being an easy task for both the CNN and the majority of the Mestre–Nagao sums—in a uniform mode they demonstrated the perfect classification even in $p < 1000$ range (see Tables 6 and 10).

Additionally, we evaluated trained models only on curves with prime conductors and curves with conductors lower than 500,000, see Tables 7, 8, and 9. Interestingly, we observed that curves with prime conductors are more challenging to classify.

The classification on the custom dataset was much more challenging for both the CNN and the Mestre–Nagao sums. For example, in all ranks classification with $p < 10,000$ and the uniform range, while the CNN with $MCC = 0.7218$ (it misclassified 23% of curves, while, for 3% of the curves prediction missed true rank for more than 1) still outperformed the best Mestre–Nagao sum $S_1$ with $MCC = 0.6890$ (it misclassified 26% of curves), it performed just a little bit better than the network $\Omega$ for which $MCC = 0.7069$ (see Table 11).

In the top conductor range (for all ranks classification on the custom dataset) all heuristics were much less efficient. This is expected, at least for the Mestre–Nagao sums, since the curves in this test set have large conductors (at least $10^{29}$) and thus we need more $a_p$’s to determine their $L$-function. The Mestre–Nagao sums slightly outperformed the CNN in this setting. For example, in $p < 10,000$ mode the MCC of the CNN was 0.3019 (it misclassified 61% of curves, while, for 12% of the curves prediction missed true rank for more than 1), while the best Mestre–Nagao sum $S_2$ had $MCC = 0.3291$. For complete information see Table 12.

In binary classification on the custom dataset (which tries to identify curves of rank 8, 9 or 10), the CNN outperformed each Mestre–Nagao sum individually, but it was a little bit worse than the neural network trained on all Mestre–Nagao sums. For example, in the top conductor range for $p < 10,000$ the MCC of the CNN was 0.6774 while the MCC of all sums was 0.7091.
Next, we explore how the relationship between conductor $N$ and the range of $a_p$ used in the algorithm (we restrict to the set of $a_p$ with $p < B$) influence the quality of classification. Understanding this relationship is important to properly assess the bound $B$ required, for example, to successfully classify elliptic curves with conductors larger than those in our dataset (or to improve the classification of the curves in the current dataset).

We note that for any particular choice of $a_p$ values with $p < B$, there is an infinite family of elliptic curves that realize them (as soon as there is one such curve). Since $a_p$ depends only on the residue classes of the Weierstrass coefficients modulo $p$, the Weierstrass coefficients of the curves in this family can be easily described using the Chinese remainder theorem. However, one expects that the average rank (when curves are ordered by height) in this family, since it is determined by congruence constraints, will be $1/2$ (with 50% of curves having rank 0 and 50% having rank 1). Thus, the values of $a_p$ in the range $p < B$ for a random curve from this family provide no useful information about its rank. In other words, if the conductor $N$ is much larger than $B$, we do not expect classification to work.

As a toy model for detecting rank-0 curves, we can numerically evaluate $L_E(1)$ using the approximate functional equation (see [8, Sect. 6]) given by

$$L_E(1) = 2 \sum_{n=1}^{\infty} \frac{a_n}{n} e^{-2\pi n/\sqrt{N}},$$

where $a_n$ are the coefficients of the Dirichlet series $L_E(s) = \sum_{n=1}^{\infty} \frac{a_n}{n^{s}}$. To ensure a small error of the approximation, we require $B = \sqrt{N}$ at a minimum. Likewise, the numerical tests for the modularity of $L$-functions [2] and the numerical methods for computing bad Euler factors of genus-two curves [3] necessitate at least $B = \sqrt{N}$ for proper functioning. Therefore, we investigate the dependence of the quality of classification of various models, as measured by the MCC, on the quantity $B/\sqrt{N}$.

Figure 7 illustrates that the quality of classification of the CNN, as a function of $B/\sqrt{N}$, is similar, although not identical, for all three trained models, where $B = 10^k$ for $k = 3, 4, 5$. This observation suggests that to maintain the quality of the classification when increasing the conductor $N$, the bound $B$ should be increased to maintain a constant $B/\sqrt{N}$ value. Since the conductors in the custom dataset reach up to $10^{30}$, it is not surprising that the model’s predictions are worse on the custom dataset than they are on the LMFDB. This is because the ratio of $B/\sqrt{N}$ is much smaller in the custom dataset than it is in the LMFDB, where the curves have conductors less than $3 \cdot 10^8$. Figure 8 further demonstrates that the CNN outperforms the other classifiers consistently in all the considered ranges of $B/\sqrt{N}$.

Finally, we evaluated the performance of the CNN model with $B = 10^5$ on a dataset of 808 elliptic curves with conductors less than $10^{29}$. These curves were sampled from the K3 elliptic surface with discriminant $-163$ given by equation (3.1), which was used by Elkies [13] to obtain the rank 12 elliptic curve with $\mathbb{Z}/4\mathbb{Z}$ torsion subgroup. This family has Mordell–Weil rank 4 and $\mathbb{Z}/4\mathbb{Z}$ torsion subgroup. Despite not being trained on curves with nontrivial torsion subgroups, the CNN model successfully classified curves of rank 6 and 7, as shown in the confusion matrix in Fig. 9. The MCC of the CNN model was 0.2492.

## 2 Mestre–Nagao sums

In this section, we define and motivate the Mestre–Nagao sums that we are going to use for rank classification.
Let
\[ \Lambda(n) = \begin{cases} 
\log p & \text{if } n = p^m, \\
0 & \text{otherwise},
\end{cases} \]
be the von Mangoldt function, and for elliptic curve \( E \) defined over \( \mathbb{Q} \) with conductor \( N_E \) let
\[ c_n = \begin{cases} 
\alpha_p + \beta_p & \text{if } n = p^m \text{ and } p \nmid N_E, \\
\alpha_p & \text{if } n = p^m \text{ and } p \mid N_E, \\
0 & \text{otherwise},
\end{cases} \]
where \( \alpha_p \) and \( \beta_p \), for \( p \nmid N_E \), are eigenvalues of the Frobenius morphism at \( p \) (thus \( \alpha_p = \alpha_p + \beta_p \) and \( \alpha_p \beta_p = p \)). Note that for \( m > 1 \) we have
\[ \alpha_p^m + \beta_p^m = (\alpha_p^{m-1} + \beta_p^{m-1})\alpha_p - p(\alpha_p^{m-2} + \beta_p^{m-2}), \]
hence for \( p \nmid N_E \) it follows
\[ c_{np} = c_{np} - (\alpha_{np} + \beta_{np}) \log p + \mathcal{O}(1). \]

We consider the following Mestre–Nagaosums.

\begin{align*}
S_0(B) &= \frac{1}{\log B} \sum_{p \leq B, \text{good reduction}} \frac{a_p(E) \log p}{p}, \\
S_1(B) &= S_0(B) - \frac{1}{B \log B} \sum_{n \leq B} c_n \Lambda(n), \\
S_2(B) &= \frac{1}{\log B} \sum_{n \leq B} \frac{c_n \Lambda(n)}{n} - \frac{1}{B \log B} \sum_{n \leq B} c_n \Lambda(n), \\
S_3(B) &= \sum_{p \leq B, \text{good reduction}} -a_p(E) + 2 \frac{\log p}{p + 1 - a_p(E)} \log p, \\
S_4(B) &= \frac{1}{B} \sum_{p \leq B, \text{good reduction}} -a_p(E) \log p, \\
S_5(B) &= \sum_{p \leq B, \text{good reduction}} \log \left( \frac{p + 1 - a_p(E)}{p} \right) + \sum_{p \leq B, \text{split mult. reduction}} \log \left( 3/2 : \frac{p - 1}{p} \right), \\
S_6(D) &= \frac{\log N_E}{2D \pi} - \frac{\log 2\pi}{D \pi} + \frac{1}{D \pi} \sum_{p \leq \exp(2\pi D)} \log p \sum_{k=1}^{\lfloor 2\pi D / \log p \rfloor} \frac{c_{pk}}{p^{k/2}} \left( 1 - k \log p \right) \\
& \quad + \frac{1}{\pi} \Re \left\{ \int_{-\infty}^{\infty} \frac{\Gamma'(1 + it)}{\Gamma(1 + it)} \left( \frac{\sin(D \pi t)}{D \pi t} \right)^2 \, dt \right\}. 
\end{align*}

The sum \( S_0 \) was analyzed in detail in [25], where it was shown that if the Riemann hypothesis for \( L_E(s) \) is true, and if the limit \( \lim_{B \to \infty} S_0(B) \) exists, then the limit is \( -\text{ran} + 1/2 \).
Table 2  Distribution of ranks in the LMFDB dataset

| Rank | 0     | 1     | 2     | 3     | 4     | 5     |
|------|-------|-------|-------|-------|-------|-------|
| Count| 1,404,510 | 1,887,132 | 493,291 | 37,334 | 2086  | 19    |
| Proportion | 36.73% | 49.34% | 12.90% | 0.98%  | 0.05%  | 0.00% |

where \( r_{an} \) is the analytic rank of \( E/\mathbb{Q} \). The relation between \( S_0 \) and the analytic rank comes from Perron’s formula applied to

\[
\frac{1}{2\pi i} \int -\frac{L_E'(s)}{L_E(s)} \frac{x^i}{s(s-1)} ds \quad x > 1, \; d > 3/2, \; a \in \mathbb{R}.
\]

From the proof of Theorem 6 in [25], by modifying the error term of \( S_0(B) \) one obtains sums \( S_1 \) and \( S_2 \) (Eqs. (3.11) and (3.12)) that have essentially the same limit as \( S_0 \), but faster convergence. Sums \( S_3 \) and \( S_4 \) were considered by Nagao [31]. Note that \( S_3(B) \) is the logarithmic derivative of the partial Euler product (1.1) evaluated at \( s = 1 \) (ignoring the primes of bad reduction). Since \( \frac{L_E'(s)}{L_E(s)} = \frac{r_{an}}{s-1} + O(1) \), it is reasonable to assume that for elliptic curve with large analytic rank \( r_{an} \), the corresponding \( S_3 \) sum will be large (see Sect. 1.3.3 in [6]). Sum \( S_5 \) is the modification of \( \tilde{S}_5 \) (from the introduction) at primes of split multiplicative reduction (see Sect. 7 in [14]).

Sum \( S_6 \) is described in [1]. It is based on the explicit formula for \( L_E(s) \) relating \( S_6(D) \) to the sum over nontrivial zeros (with multiplicities) \( \frac{1}{2} + iy \) of \( L_E(s) \), \( \sum_{\gamma} f(\gamma) \), where \( f(z) = f(z; D) = \left( \frac{\sin(D\pi z)}{D\pi z} \right)^2. \) Note that \( f(0) = 1 \) and \( f(x) \geq 0 \) for all real \( x \), thus, \( \sum_{\gamma} f(\gamma) \) gives an upper bound for \( r_{an} \). Assuming Riemann Hypothesis for \( L_E(s) \), one has \( \lim_{D \to \infty} S_6(D) = r_{an} \), but computing \( S_6(D) \) becomes infeasible once \( D \) gets a little larger than 4.4. For example, already to compute \( S_6(2) \) one needs \( a_p \)'s for all \( p < 286, 751 \).

For computing \( S_6(D) \) we used C version [24] of PSAGE [39] rankbound algorithm by Bober.

3 Datasets

We trained our models on two datasets, namely the LMFDB and the custom dataset. Furthermore, we evaluated the performance of the models on the dataset of elliptic curves sampled from the K3 elliptic surface given by Eq. (3.1).

3.1 LMFDB

The LMFDB database [40] contains 3,824,372 elliptic curves defined over \( \mathbb{Q} \), distributed in 2,917,287 isogeny classes. The LMFDB database is a union of three distinct datasets: all curves of conductor less than 500,000, all curves whose conductor is 7-smooth, and all curves of prime conductor \( p \leq 200,000,000 \). We selected one representative per isogeny class, as curves in the same isogeny class have the same \( L \)-functions and hence the same analytic rank (but not necessarily the same torsion subgroup). Since there are only 19 curves with a rank equal to 5, we didn’t take any of them into consideration. See Table 2 for rank distribution. Moreover, 63.71% of curves have trivial torsion, while 31.18% have \( \mathbb{Z}/2\mathbb{Z} \) torsion.
### Table 3  Distribution of conductors in the custom dataset

| Range       | Proportion |
|-------------|------------|
| \([1,10^1]\) | 0.94%      |
| \([10^1,10^{10}]\) | 21.60%    |
| \([10^{10},10^{15}]\) | 26.97%    |
| \([10^{15},10^{20}]\) | 21.27%    |
| \([10^{20},10^{25}]\) | 17.02%    |
| \([10^{25},10^{30}]\) | 10.19%    |
| \([10^{30},10^{30}]\) | 1.97%      |

### Table 4  Distribution of ranks in the custom dataset

| Rank | Count  | Proportion |
|------|--------|------------|
| 0    | 78,755 | 3.79%      |
| 1    | 121,966| 5.87%      |
| 2    | 179,593| 8.65%      |
| 3    | 435,195| 20.97%     |
| 4    | 543,713| 26.20%     |
| 5    | 405,129| 19.52%     |

| Rank | Count  | Proportion |
|------|--------|------------|
| 6    | 210,439| 10.14%     |
| 7    | 77,923 | 3.75%      |
| 8    | 19,272 | 0.92%      |
| 9    | 2670   | 0.12%      |
| 10   | 201    | 0.01%      |

### 3.2 Custom dataset

Our custom dataset contains 2,074,863 elliptic curves defined over \(\mathbb{Q}\) with trivial torsion and conductor less than \(10^{30}\). See Table 3 for the number of curves in a specific conductor range. Our curves have rank between 0 and 10. See Table 4 for the distribution of curves by rank.

We used two methods for generating the dataset. First, we produced curves with random Weierstrass coefficients. These were primarily curves of rank 0 and 1.

The curves of higher rank were obtained as random specializations of pencils of cubics through randomly selected \(k\) rational points in the plane, for \(k = 2, 3, \ldots, 8\). Every such cubic (if nonsingular) was transformed to Weierstrass form by the change of variables described in Chapter 8 of [7]. We removed curves with nontrivial torsion subgroups.

Next, using PARI/GP [33] function ellrank we tried to compute ranks of all previously generated curves (assuming the Parity conjecture). Ultimately, we discarded those for which PARI/GP couldn’t find the rank. Note that ellrank in computing the upper bound for the rank of the elliptic curve \(E\) approximates the rank of \(\text{III}(E/\mathbb{Q})[2]\) from below with the rank of \(\text{III}(E/\mathbb{Q})[2]/2\text{III}(E/\mathbb{Q})[4]\) (which it computes via Cassels pairing), thus it can not determine the rank of \(E\) if \(\text{III}(E)[4]\) is nontrivial. Consequently, such curves are missing from our dataset.

It is important to note that this dataset suffers from bias as it is constructed by sampling the rational points of small height in the pencil of cubics. Consequently, it contains many elliptic curves with small canonical height generators and hence small regulators. This presents a problem, particularly for curves of small rank and large conductor, since the regulator is typically expected to be large under standard conjectures. The size of the regulator is significant since the Mestre–Nagao sums ultimately approximate a term whose size depends on both the rank and the regulator. For instance, from the original Birch and Swinnerton-Dyer conjecture,

\[
\prod_{p \text{ good reduction}, p < B_{\text{good reduction}}} \frac{1 - a_p}{p} \approx A \log(B)^r,
\]

it follows (by taking logarithms) that \(\hat{S}_5(B) = \log A + r \log \log B + o(1)\), where \(A\) is a constant that conjecturally depends on the regulator of \(E\) (e.g. see [15]).

While this dataset does not represent typical elliptic curves of a given rank and conductor range, it is still helpful for comparing different classification methods.
Table 5  Distribution of conductors in the K3 elliptic surface dataset

| Range             | (1, 10^2) | (10^2, 10^5) | (10^5, 10^10) | (10^10, 10^15) | (10^15, 10^20) | (10^20, 10^25) | (10^25, 10^29) |
|-------------------|-----------|--------------|---------------|----------------|----------------|----------------|----------------|
| Proportion        | 0.12%     | 0.49%        | 1.61%         | 7.30%          | 26.86%         | 63.61%         |                |

3.3 K3 elliptic surface
Consider the K3 elliptic surface with discriminant $-163$ (see [10, 13])

$$
y^2 = x^3 + (65536t^4 - 17472t^3 - 10176t^2 + 18672t - 3535)x^2 + 1024(t + 1)^2(15t - 8)^2(31t - 7)^2x,
$$

of Mordell–Weil rank 4 (over $\mathbb{Q}(t)$) with $\mathbb{Z}/4\mathbb{Z}$ torsion subgroup. Generators of the infinite part of the Mordell–Weil group are points with $x$-coordinates

$$
x_1 = -361(t + 1)(31t - 7),
x_2 = -4(t + 1)(15t - 8)(16t - 7)^2,
x_3 = -16(t + 1)(8t + 7)^2(15t - 8),
x_4 = 4(15t - 8)(16t + 1)^2(31t - 7),
$$

while the point of order 4 has $x$-coordinate equal to $32(t + 1)(15t - 8)(31t - 7)$.

To generate a sample of curves from this family, we substituted $t$ with an integer between $-2000$ and $2000$, resulting in 808 curves with conductors less than $10^{29}$. However, we had to discard 37 curves because we were unable to compute their rank. The distribution of conductors in the dataset is given in Table 5.

4 Neural network architectures and training procedure
The models we employ are convolutional neural networks (CNN) [16, II.9]. The CNN could be regarded as the composition of functions, so-called layers. Each layer is in principle composition of a fittable affine mapping and an element-wise nonlinear function. To detect and utilize any correlations in $a_p$ sequence inherent to our data, for the affine mapping we use 1D matrix convolutional operations, inspired by the success of 2D matrix convolutions in image classification task [27]. A very similar CNN architecture is used in [42], where it is explained in more detail. The point is that all Mestre–Nagao sums inherently employ the same computation “rule” for each $a_p$, which we expect could be efficiently modeled by a matrix convolution operation. Thus, we can regard the training of a CNN as an optimization in the highly dimensional parameterized space of Mestre–Nagao-like heuristics.

Additionally, we use smaller fully connected neural networks (FCNN) for the classification of curves using only precomputed values of the selected Mestre–Nagao sums and conductors.

All described models are implemented in Python using the PyTorch library [35]. Our models’ training and evaluation code is available online [23]. Models were trained using a server with dual Xeon 5218R CPUs (40 cores) equipped with 6 Nvidia Quadro RTX 5000 GPUs, each with 16 GB of RAM.
4.1 Architecture of CNNs

For each elliptic curve $E$, the CNN takes as an input a matrix $M(E)$ of shape $3 \times \pi(B)$, where $B = 10^3, 10^4, 10^5$ and the columns are indexed with primes less than $B$. The first row consists of $a_p$-s divided by $\sqrt{p}$ (by Hasse theorem $a_p/\sqrt{p} \in [-2, 2]$). The second row contains a constant sequence of normalized conductors, $\log N_E/\log N_{\text{max}}$, where $N_{\text{max}}$ is the maximal conductor present in the dataset. The third row consists of a sequence of values $r_n = -1 + \frac{2n}{\pi(N)}$ (a linear sweep from $-1$ to $1$). Each $r_n$ represents the relative position of the $n$-th prime $p_n$ in a sequence of primes less than $B$.

Each CNN consists of the following sequence of convolutional layers:

1. Preparatory layer: increasing the number of channels (rows in the input matrix) from $3$ to $64$.
2. Several input layers $I_1, \ldots, I_{L_1}$: preserving the number of channels and the length of the input sequence.
3. Several reducing layers $R_1, \ldots, R_{L_2}$: preserving the number of channels and reducing the length of the input sequence by a factor of $2$ (convolution kernel stride is equal to $2$).
4. Several output layers $O_1, \ldots, O_{L_3}$: preserving the number of channels and the length of the input sequence.

Each convolution layer is followed by an activation function and a batch normalization layer, improving the convergence of the network training and network generalization properties [20]. For the activation function, we decided to use ReLU. Kernel size $K$ is the same in each layer. The number of reducing layers $L_2$ is chosen such that the length of the output of the final reducing layer $R_{L_2}$ (and consequently the length of the output of every output layer) is equal to $1$.

Following convolutional layers, we have one final fully connected layer reducing dimensionality from $64$ to the number of classification labels (ranks). Thus, our CNN architecture is fully determined with hyperparameters $L_1, L_2, L_3,$ and $K$. In contrast to model parameters, which are learned in the training process (see Sect. 4.3), hyperparameters are the parameters that determine the model architecture and the model optimization process.

For instance, in the case of uniform range for the LMFDB model with $p < 10,000$ we used the following values of hyperparameters: $L_1 = 0, L_2 = 11, L_3 = 3, K = 17$.

To find optimal values of hyperparameters we conducted Bayesian optimization [30] in the space of hyperparameters by training altogether $500$ different variants of networks. Similar hyperparameters were acquired for LMFDB in the case of $p < 1000$ and for $p < 100,000$, due to lengthy computations, we just reused the hyperparameters for $p < 10,000$, by setting $L_2 = 14$. For each custom dataset CNN model, we took the same values of hyperparameters as for the corresponding LMFDB model.

4.2 Architecture of FCNNs

Here the input to the network for each curve $E$ consists of one or more precomputed values of the Mestre–Nagao sums for $a_p$ in a certain range and the normalized conductor $N_E$, as in the case of CNNs. In every case, our FCNN consists of an input linear layer having the size of the number of input features (the Mestre–Nagao sums and the normalized conductor), four hidden linear layers of size $128$, and an output linear layer of the size
of the number of classification labels (ranks). In between each of the linear layers, we use dropout [32], which reduces overfitting during training, and the ReLU activation function. In our experience, the capacity of this network is more than enough to learn the dependency of rank on the input.

4.3 Training procedure

Training a neural network means optimizing network parameters (weights in convolutional and linear layers) to minimize a chosen loss function over a given dataset.

Dataset

For neural network training, as customary the whole dataset is divided into three disjoint subsets called training, validation, and test dataset. Only the training and validation datasets are used in network training. The training set is used directly by the gradient descent-based optimizer for the gradient backpropagation. The validation set is only used to avoid model overfitting or underfitting during training. The test dataset is not "seen" by the network during the training and is only used at the end to test the performance of already trained models. We used the 4 : 1 ratio between the training and validation datasets.

Loss function

As we are training classifier networks, we decided to use the cross entropy loss function as implemented in PyTorch library [35]. Additionally, we supply weights for the loss function to reflect on disproportional sizes of classes, which is considered standard practice.

Optimizer hyperparameters

Our models are trained using the fast.ai library [19]. For the optimizer, we use the usual choice of Adam algorithm [26]. For governing learning rate and momentum scheduling we used a one-cycle policy with cosine annealing [38], which ensures better convergence of model parameters to a broad optimum and allows for better generalization of the trained model [37]. For the optimizer hyperparameters, we used different ranges of the maximum learning rate for CNNs and FCNNs.

For CNNs the range was between $3.5 \times 10^{-4}$ and $1.5 \times 10^{-3}$. Number of training epochs was equal to 40, and batch size equal to 1024 ($p < 10^5$) or 2048 ($p < 10^3$ and $p < 10^4$). For Adam-specific hyperparameters, we have used: $\beta_2 = 0.99$, $\epsilon = 10^{-5}$, and weight decay of $10^{-3}$.

For FCNNs, we employed Bayesian optimization for three hyperparameters: dropout, maximum learning rate, and weight decay, by training 200 different models (networks) for each of the 7 Mestre–Nagao sums $\{S_0, \ldots, S_6\}$ and also for $\Omega$, in each of the 24 test variants. Finally, in each of these 192 tests performed, we took the model having the best MCC on the test set. The number of training epochs for every model was equal to 10 and the rest of the Adam-specific hyperparameters were the same as in the CNN training.

5 Results

We present rank classification results using the Mathews correlation coefficient in each of the 216 tests. We use two different datasets (LMFDB and custom, see Sect. 3), three $a_p$-s
ranges \((p < 10^3, 10^4, 10^5)\), two modes for selecting test curves (20\% uniformly selected or top conductor range), and two types of classification (binary or all ranks).

For binary classification we always trained our networks for all ranks classification and to obtain binary classification we merged all rank labels into two categories, “low” and “high” ranks—effectively only reinterpreting all ranks classification, but rerunning Bayesian optimization for hyperparameters. For the LMFDB low ranks are 0, 1, 2, 3 and high ranks are 4, and for the custom dataset low ranks are 0, \ldots, 7 and high ranks are 8, 9, 10. Interestingly, this procedure produced much better binary classification results than training directly a binary classifier model.

Regarding the CNN models, for each out of 12 test variants (varying two datasets, three ranges, and two modes for selecting test curves) we trained only 10 different networks due to high computational expense. For each of these 12 test variants, we computed the best MCC out of 10 different trained models, separately for all ranges classification and binary classification. For instance training of models in the worst case of \(p < 10^5\) took over 14 h per model on a single GPU. On the other hand, training of FCNNs for the Mestre–Nagao sums was much less computationally demanding, so for each of the 192 test cases, we trained at least 200 models (which took several minutes per model on a single GPU), using Bayesian optimization in the space of hyperparameters, as described in Sect. 4.3. For each of the test cases, we selected the best-performing model out of all trained models.

At the end of the section, we present the performance of the CNN model with \(B = 10^5\) on a K3 elliptic surface dataset (see Sect. 3).

5.1 LMFDB

Test results for the complete LMFDB dataset are presented in Tables 6 and 10. As the LMFDB dataset is heterogeneous, we also separately present in Tables 7, 8, and 9 partial test results for all curves of prime conductor, all curves of conductor less than 500,000, and all curves of prime conductor less than 500,000, respectively.

In Table 6 are maximal values of the MCC obtained for different classifiers. Classifiers are trained using 80\% of randomly selected curves from the LMFDB dataset, having a conductor less than 10^8. The MCC is computed on the other 20\% of the dataset, not seen during the training. Classifiers (the CNN, \(\Omega\), and the Mestre–Nagao sums from Sect. 2) are trained using values of \(a_p\)-s for primes \(p\) less than 10^3, 10^4 or 10^5, and the value of the elliptic curve conductor.

For Tables 7, 8, and 9 we did not train additional classifiers. We evaluated already trained classifiers from Table 6 on the test dataset restricted to appropriate subsets.

In Table 10 classifiers are trained using all curves from the LMFDB dataset having a conductor less than 10^9. The MCC is computed on all curves from the LMFDB dataset having a conductor between 10^8 and 10^9, not seen during the training.

5.2 Custom dataset

Test results are presented in Tables 11 and 12.

In Table 11 are maximal values of the MCC obtained for different classifiers. Classifiers are trained using 80\% of randomly selected curves from the custom dataset with a con-
### Table 6  LMFDB with the uniform test dataset

| Type of classifier | Number of $a_p$-s used | Classify all ranks | Binary classification |
|--------------------|------------------------|--------------------|-----------------------|
|                    |                        | $p < 10^3$ | $p < 10^4$ | $p < 10^5$ | $p < 10^6$ | $p < 10^7$ |
| CNN                | 0.9507                 | 0.9958        | 0.9992        | 1          | 1          | 1          |
| $S_0$              | 0.6823                 | 0.8435        | 0.9068        | 0.9877     | 0.9758     | 0.9729     |
| $S_1$              | 0.6848                 | 0.8507        | 0.9301        | 0.9968     | 1          | 1          |
| $S_2$              | 0.7277                 | **0.8697**    | 0.9359        | 0.9938     | 1          | 1          |
| $S_3$              | 0.6933                 | 0.8499        | 0.9142        | 0.9420     | 0.9842     | 0.9726     |
| $S_4$              | 0.2678                 | 0.3015        | 0.1525        | 0.2188     | 0.1103     | 0.0744     |
| $S_5$              | 0.6132                 | 0.7774        | 0.8463        | 0.9968     | 0.9968     | 1          |
| $S_6$              | **0.6969**             | 0.8647        | **0.9381**    | 1          | 1          | 1          |
| $\Omega$           | 0.8685                 | **0.9602**    | **0.9826**    | 1          | 1          | 1          |

![Confusion matrix of the CNN for the LMFDB and $p < 10^5$ with the uniform test dataset](image)

**Fig. 1** Confusion matrix of the CNN for the LMFDB and $p < 10^5$ with the uniform test dataset

### Table 7  LMFDB with the uniform test dataset and only prime conductors

| Type of classifier | Number of $a_p$-s used | Classify all ranks | $p < 10^3$ | $p < 10^4$ | $p < 10^5$ |
|--------------------|------------------------|--------------------|------------|------------|------------|
| CNN                | 0.7564                 | **0.9802**        | 0.9961     | 0.7552     |
| $S_1$              | 0.3498                 | 0.6017            | 0.9130     |
| $\Omega$           | 0.5920                 | 0.8203            | 0.9130     |

### Table 8  LMFDB with the uniform test dataset and conductors less than 500,000

| Type of classifier | Number of $a_p$-s used | Classify all ranks | $p < 10^3$ | $p < 10^4$ | $p < 10^5$ |
|--------------------|------------------------|--------------------|------------|------------|------------|
| CNN                | **0.9838**             | **0.9987**        | **0.9998** | 0.9602     |
| $S_1$              | 0.7413                 | 0.8934            | 0.9936     |
| $\Omega$           | 0.9172                 | 0.9831            | 0.9936     |
Table 9  LMFDB with the uniform test dataset and prime conductors less than 500,000

| Type of classifier | Number of \( q_p \)-s used |
|--------------------|----------------------------|
| CNN                |                           |
| \( p < 10^3 \)     | 0.9459                    |
| \( p < 10^4 \)     | 0.9985                    |
| \( p < 10^5 \)     | 1                         |
| \( S_1 \)          | 0.6535                    |
| \( S_2 \)          | 0.8455                    |
| \( S_3 \)          | 0.9392                    |
| \( \Omega \)       | 0.9241                    |
| \( p < 10^4 \)     | 0.9633                    |
| \( p < 10^5 \)     | 0.9664                    |

Table 10  LMFDB with the top conductor range

| Type of classifier | Number of \( q_p \)-s used |
|--------------------|----------------------------|
| CNN                |                           |
| \( p < 10^3 \)     | 0.5631                    |
| \( p < 10^4 \)     | 0.9289                    |
| \( p < 10^5 \)     | 0.9846                    |
| \( p < 10^6 \)     | 0.9811                    |
| \( p < 10^7 \)     | 1                         |
| \( S_0 \)          | 0.2880                    |
| \( S_1 \)          | 0.4983                    |
| \( S_2 \)          | 0.5057                    |
| \( S_3 \)          | 0.6545                    |
| \( S_4 \)          | 0.9470                    |
| \( S_5 \)          | 0.9750                    |
| \( S_6 \)          | 0.9756                    |
| \( \Omega \)       | 0.4433                    |
| \( p < 10^4 \)     | 0.7013                    |
| \( p < 10^5 \)     | 0.8530                    |
| \( p < 10^6 \)     | 0.9796                    |
| \( p < 10^7 \)     | 1                         |

Fig. 2  Confusion matrix of the CNN for the LMFDB and \( p < 10^4 \) with the top conductor range

ductor less than \( 10^{29} \). As in the previous section, the MCC is computed on the other 20% of the dataset, not seen during the training.

In Table 12 classifiers are trained using all curves from the custom dataset with conductor less than \( 10^{29} \). The MCC is computed on all curves from the custom dataset with conductor between \( 10^{29} \) and \( 10^{30} \), not seen during the training.

Figures 7 and 8 depict the relationship between the MCC and \( \log_{10}(B \sqrt{N}) \) for 82 subsets of curves corresponding to conductor intervals of \((1, 50, 000), (50, 000 \cdot 2^k, 50, 000 \cdot 2^{k+1})\)
Fig. 3  Rank cutoffs of the classifier $S_5$ as a function of a conductor, trained on the LMFDB with the uniform test set and $p < 10^4$. On the x-axis are $\log_{10}$ values of conductors and on the y-axis are values of the sum $S_5$. The unexpected shape of the cutoff between ranks 3 and 4 is the consequence of a small number of rank 4 curves with a small conductor, which are present in the dataset.

Fig. 4  Confusion matrix of $S_5$ for the LMFDB and $p < 10^4$ with the top conductor range for $k = 0, \ldots, 80$. In Fig. 7, we present the results of the same CNN classifier for three different values of $B$ ($10^3$, $10^4$, and $10^5$), while in Fig. 8, we compare the performance of three different classifiers (the CNN, $\Omega$, and $S_1$) for $B = 10^5$.

5.3 K3 elliptic surface

The confusion matrix of the CNN with $p < 10^5$ for this dataset can be found in Fig. 9. The MCC of the CNN is equal to 0.2492.
Table 11 Custom dataset with the uniform test set

| Type of classifier | Classify all ranks | binary classification |
|--------------------|--------------------|------------------------|
|                    | $p < 10^3$         | $p < 10^4$         | $p < 10^5$         | $p < 10^3$         | $p < 10^4$         | $p < 10^5$         |
| CNN                | 0.6129             | 0.7218             | 0.7958             | 0.6695             | 0.8335             | 0.9425             |
| $S_0$              | 0.5738             | 0.6782             | 0.7462             | 0.6761             | 0.8275             | 0.9175             |
| $S_1$              | 0.5780             | 0.6890             | 0.7592             | 0.6484             | 0.8317             | 0.9309             |
| $S_2$              | 0.5649             | 0.6761             | 0.7521             | 0.6407             | 0.8252             | 0.9206             |
| $S_3$              | 0.5551             | 0.6616             | 0.7361             | 0.6515             | 0.8118             | 0.9074             |
| $S_4$              | 0.2893             | 0.2472             | 0.2251             | 0.4271             | 0.4179             | 0.3874             |
| $S_5$              | 0.4987             | 0.5990             | 0.6696             | 0.5230             | 0.6919             | 0.7956             |
| $S_6$              | 0.5230             | 0.6509             | 0.7361             | 0.5179             | 0.7554             | 0.8961             |
| $\Omega$           | 0.5999             | 0.7069             | 0.7807             | 0.6821             | 0.8527             | 0.9412             |

Fig. 5 Confusion matrix of the CNN for the custom dataset and $p < 10^4$ with the uniform test set

Fig. 6 Confusion matrix of $\Omega$ for the custom dataset and $p < 10^4$ with the uniform test set
Table 12  Custom dataset with the top conductor range

| Type of classifier | Number of $a_p$-s used | classify all ranks | binary classification |
|--------------------|-------------------------|-------------------|-----------------------|
|                    | $p < 10^3$ | $p < 10^4$ | $p < 10^3$ | $p < 10^4$ | $p < 10^5$ | $p < 10^5$ |
| CNN                | 0.2147 | 0.3019 | 0.3655 | 0.5449 | 0.6774 | 0.8082 |
| $S_0$              | 0.2533 | **0.3233** | 0.3719 | 0.5866 | 0.6978 | 0.7900 |
| $S_1$              | **0.2573** | 0.3291 | **0.3834** | 0.5649 | 0.6925 | 0.7946 |
| $S_2$              | 0.2340 | 0.3118 | 0.3688 | 0.5472 | 0.6814 | 0.7856 |
| $S_3$              | **0.2556** | 0.3189 | 0.3645 | 0.5688 | 0.6830 | 0.7807 |
| $S_4$              | 0.1234 | 0.1228 | 0.1024 | 0.3966 | 0.3935 | 0.3842 |
| $S_5$              | 0.2081 | 0.2858 | 0.3380 | 0.4860 | 0.6071 | 0.6880 |
| $S_6$              | 0.1803 | 0.2757 | 0.3527 | 0.4321 | 0.6217 | 0.7326 |
| $\Omega$           | **0.2622** | **0.3246** | **0.3905** | 0.5931 | 0.7091 | 0.8118 |

Fig. 7  MCC of the CNN as a function of $\log_{10}(B/\sqrt{N})$, for $B = 10^3, 10^4, 10^5$ in red, green, and blue, respectively

5.4 Discussion

As expected, classifiers based on $S_0$, $S_1$, and $S_2$ have very similar performance across all tested regimes (improved convergence of sums $S_1$ and $S_2$ does not seem to improve the quality of the classification). They have the best performance of all considered Mestre–Nagao sums. Classifiers based on $S_3$ have an overall slightly worse performance than the first three sums. Peculiarity, $S_4$ underperforms in all tests, and its performance, in contrast to every other sum, decreases as the number of used $a_p$-s increases. Classifier $S_5$ performs well but compared to the first four sums is in most cases significantly lesser. While classifier $S_6$ is among the best on the curves having small conductors, its performance fades on larger conductors. Classifier $\Omega$, which is based on all Mestre–Nagao sums simultaneously, is practically always better than any other sum-based classifier, especially on curves of a small conductor.

Finally, CNN-based classifiers on the LMFDB dataset were significantly better than $\Omega$ (and consequently any other sum-based classifier). On custom datasets with a uniform test set, they were slightly better than $\Omega$ and other classifiers, but on the custom dataset with
top range conductor, they performed worse. Notice that every CNN-based classifier was selected as the best one out of only a 10 trained model, while each Mestre–Nagao sum-based classifier was selected out of at least 200 trained models, whose hyperparameters were additionally optimized using the Bayesian optimization technique. We expect that the quality of a CNN-based classification would increase with more trained models and possibly surpass other models. As the CNN model training is much more computationally expensive, more training was unfeasible for us. It is fascinating how the CNN almost perfectly classifies LMDFB with a uniform test database (see Fig. 1)—it misclassified only 0.045% of all curves! On the other hand for the custom dataset, both the CNN and \( \Omega \) classifiers perform similarly—their confusion matrices look almost identical (see Figs. 5 and 6).
We noticed that all classifiers had the highest error rates when classifying curves of rank 0 and 1 (see Fig. 5). Through the process of training Mestre–Nagao sum models, we were able to learn the optimal cutoffs for rank classification (see Fig. 3). This information is particularly useful in searching for curves of high rank (as discussed in Sect. 7 of [14]).

Regarding the results presented in Tables 8 and 9, we noticed that classifying curves with prime conductors is more challenging than classifying all curves. This finding is surprising given our observation from the previous paragraph since the proportion of curves of rank 0 and 1 with prime conductors is lower than the proportion of curves of rank 0 and 1 in the entire LMFDB dataset. Notably, MCC values in Table 7 are significantly lower than the corresponding values for the entire LMFDB dataset, as the average conductor in the family of curves with a prime conductor is much higher than that in the LMFDB database.

As shown in Fig. 8, the CNN outperforms the other two classifiers ($S_1$ and $\Omega$) across the entire range of $B/\sqrt{N}$. For instance, to achieve an MCC level of 0.7, the CNN requires a $B$ value that is approximately 6.5 times smaller than that required by the $S_1$ classifier for curves in the same conductor range.

6 Future research

This paper left us with some open questions which may be addressed in future projects:

(1) The CNN and $\Omega$ models could be used as a substitution for the current use of the Mestre–Nagao sums in the search for elliptic curves of high rank. Also, these models could be trained on other elliptic curves datasets, such as curves with non-trivial torsion, and with additional input of the root number (to employ the Parity conjecture).

(2) It is unclear why the CNN works much better than all of the other Mestre–Nagao sum-based models on the curves from the LMFDB (or in general on curves of a small conductor). Did the CNN discover some new mathematics?

(3) Suboptimal choices in the CNN model architecture and optimizer hyperparameters (see Sect. 4) may lead to underperforming classifiers. One could perform more extensive optimization of hyperparameters, especially in the case of the custom dataset.

(4) Using some of the more advanced neural network architectures, originally developed for natural language processing tasks (see [41]) may lead to better-performing models. The idea is that such architectures can more easily extract distant correlations in the $a_p$-s sequence.

Acknowledgements

We would like to express our sincere gratitude to the anonymous referees whose insightful comments and suggestions greatly improved the quality of the paper. The first author was supported by the Croatian Science Foundation under the Project No. IP-2018-01-1313, and by the QuantiXLie Center of Excellence, a project co-financed by the Croatian Government and European Union through the European Regional Development Fund - the Competitiveness and Cohesion Operational Programme (Grant KK.01.1.1.01.0004). The second author was supported by Croatian Science Foundation (HRZZ) Grant PZS-2019-02-3055 from the “Research Cooperability” program funded by the European Social Fund.

Data availability The data that support the findings of this study are available on request from the corresponding author M.K. The data are not publicly available due to the large size. The
computer code that supports the findings of this study have been deposited at https://github.com/domagojvlah/deepellrank.

Author details
1Department of Mathematics, University of Zagreb, Bijenička cesta 30, 10000 Zagreb, Croatia, 2Department of Applied Mathematics, Faculty of Electrical Engineering and Computing, University of Zagreb, Unska 3, 10000 Zagreb, Croatia.

Received: 9 October 2022   Accepted: 19 June 2023   Published online: 28 June 2023

References
1. Bober, J.W.: Conditionally bounding analytic ranks of elliptic curves. In: ANTS X—Proceedings of the Tenth Algorithmic Number Theory Symposium, vol. 1 of Open Book Ser., pp. 135–144. Math. Sci. Publ., Berkeley, CA (2013)
2. Booker, A.R.: Numerical tests of modularity. J. Ramanujan Math. Soc. 20(4), 283–339 (2005)
3. Booker, A.R., Sijling, J., Sutherland, A.V., Voight, J., Yasaki, D.: A database of genus-2 curves over the rational numbers. LMS J. Comput. Math. 19(suppl. A), 235–254 (2016)
4. Brougherbel, S., Jaray, F., El-Anbari, M.: Optimal classifier for imbalanced data using Matthews correlation coefficient metric. PLoS ONE 12(6), 1–17 (2017).
5. Breuil, C., Conrad, B., Diamond, F., Taylor, R.: On the modularity of elliptic curves over \( \mathbb{Q} \): wild 3-adic exercises. J. Am. Math. Soc. 14(4), 843–939 (2001)
6. Campbell, G.: Finding elliptic curves and families of elliptic curves over \( \mathbb{Q} \) of large rank. PhD thesis, Rutgers University (1999)
7. Cassels, J.W.S.: Lectures on Elliptic Curves. London Mathematical Society Student Texts, vol. 24. Cambridge University Press, Cambridge (1991)
8. Cohen, H.: Computing \( \zeta \)-functions: a survey. J. Théor. Nombres Bordeaux 27(3), 699–726 (2015)
9. Davis, A., Velčić, P., Buesing, L., et al.: Advancing mathematics by guiding human intuition with AI. Nature 310, 600–70–74 (2021)
10. Dujella, A., Peral, J.C.: Construction of high rank elliptic curves. J. Geom. Anal. 31(7), 6698–6724 (2021)
11. Dujella, A.: Infinite families of elliptic curves with high rank and prescribed torsion. https://web.math.pmf.unizg.hr/~duje/tors/generic.html
12. Elkies, N.D.: \( \text{Z}^2 \) in \( \mathbb{F}(Q) \) Number Theory Listserver (2006)
13. Elkies, N.D.: Three lectures on elliptic surfaces and curves of high rank. arXiv:0709.2908 (2007)
14. Elkies, N.D., Klagsbrun, Z.: New rank records for elliptic curves having rational torsion. In: ANTS XIV. Proceedings of the Fourteenth Algorithmic Number Theory Symposium, Auckland, New Zealand, virtual event, June 29–July 4, 2020, pp. 233–250. Mathematical Sciences Publishers (MSP), Berkeley, CA (2020)
15. Golofeld, D.: Sur les parties pursles eulériens attachés aux courbes elliptiques. C. R. Acad. Sci. Paris Sér. I Math. 294(14), 471–474 (1982)
16. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016)
17. He, Y.-H., Lee, K.-H., Oliver, T., Pozdnyakov, A.: Murmurations of elliptic curves. arXiv:2204.10140 (2022)
18. He, Y.-H., Lee, K.-H., Oliver, T.: Machine learning invariants of arithmetic curves. J. Symb. Comput. 115, 478–491 (2023)
19. Howard, J., et al. fastai. https://github.com/fastai/fastai (2018)
20. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. arXiv:1502.03167 (2015)
21. Ivezk, T., Vlah, D.: BlackBox: generalizable reconstruction of extremal values from incomplete spatio-temporal data. Extremes 24(1), 145–162 (2021)
22. Ivezk, T., Vlah, D.: Reconstruction of incomplete wildfire data using deep generative models. Extremes (2023)
23. Kazalicki, M., Vlah, D.: deepellrank. https://github.com/domagojvlah/deepellrank (2022)
24. Kedlaya, K.S., Sutherland, A.V.: Computing \( \zeta \)-series of hyperelliptic curves. In: Algorithmic number theory, volume 5011 of Lecture Notes in Comput. Sci., pp. 312–326. Springer, Berlin (2008)
25. Kim, S., Murty, M.R.: From the Birch and Swinnerton-Dyer conjecture to Nagao’s conjecture. Math. Comput. 92(339), 385–408 (2022)
26. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization (2017)
27. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Adv. Neural. Inf. Process. Syst. 25, 1106–1114 (2012)
28. Mazur, B.: Modular curves and the Eisenstein ideal. Publ. Math. Inst. Hautes Étud. Sci. 47, 33–186 (1977)
29. Mestre, J-F.: Construction d’une courbe elliptique de rang \( \geq 12 \) C. R. Acad. Sci. Paris Sér. I Math. 295, 643–644 (1982)
30. Molčuk, J.: On Bayesian Methods for Seeking the Extremum, pp. 400–404. Springer, Berlin (1975)
31. Nagao, K.: Examples of elliptic curves over \( \mathbb{Q} \) with rank \( \geq 17 \). Proc. Jpn. Acad. Ser. A 68(8), 287–289 (1992)
32. Nitish, S., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. 15(56), 1929–1958 (2014)
33. PARI/GP, version 2.11.1. Bordeaux 2019. http://pari.math.u-bordeaux.fr/
34. Park, J., Poonen, B., Voight, J., Wood, M.M.: A heuristic for boundedness of ranks of elliptic curves. J. Eur. Math. Soc. (JEMS) 21(9), 2859–2903 (2019)
35. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Benoit Steiner, L., Fang, J.B., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d’Alché-Buc, F., Fox, E.B., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8–14 December 2019, pp. 8024–8035. Canada, Vancouver, BC (2019)
36. Poincaré, H.: Sur les propriétés arithmétiques des courbes algébriques. J. de Math. 5(7), 161–233 (1901)
37. Smith, L.N.: A disciplined approach to neural network hyper-parameters: part 1—learning rate, batch size, momentum, and weight decay (2018)
38. Smith, L.N., Topin, N.: Very fast training of neural networks using large learning rates. Super-convergence (2018)
39. Stein, W.A. et al.: Purple SAGE. http://purple.sagemath.org/ (2011)
40. The LMFDB Collaboration. The L-functions and modular forms database. http://www.lmfdb.org (2022). Accessed 4 July 2022
41. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Guyon, I., Von Luxburg, U., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc., Red Hook (2017)
42. Vlah, D., Šepetanc, K., Pandžić, H.: Solving bilevel optimal bidding problems using deep convolutional neural networks. IEEE Syst. J. (2023). https://doi.org/10.1109/JSYST.2022.3232942
43. Wagner, A.Z.: Constructions in combinatorics via neural networks. arXiv:2104.14516 (2021)
44. Watkins, M.: A discursus on 21 as a bound for ranks of elliptic curves over q, and sundry related topics. http://magma.maths.usyd.edu.au/~watkins/papers/DISCURSUS.pdf (2015)
45. Watkins, M., Donnelly, S., Elkies, N.D., Fisher, T., Granville, A., Rogers, N.F.: Ranks of quadratic twists of elliptic curves. In: Numéro consacré au trimestre "Méthodes arithmétiques et applications", automne 2013, pp. 63–98. Besançon: Presses Universitaires de Franche-Comté (2014)
46. Wiles, A.: Modular forms, elliptic curves, and Fermat’s Last Theorem. In: Proceedings of the International Congress of Mathematicians, ICM ’94, August 3–11, 1994, Zürich, Switzerland. Vol. I, pp. 243–248. Basel: Birkhäuser (1995)

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

Springer Nature or its licensor (e.g. a society or other partner) Holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.