Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Machine Learning Discovers New Champion Codes

Q Le

q.le@trinity.ox.ac.uk

University of Oxford https://orcid.org/0000-0002-8172-2464

Yang-Hui He

London Institute of Mathematical Sciences

Dmitrii Riabchenko

City St George's University of London

Alexander Kasprzyk

University of Warwick

Article

Keywords:

Posted Date: July 29th, 2025

DOI: https://doi.org/10.21203/rs.3.rs-7196769/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: There is **NO** Competing Interest.

Machine Learning Discovers New Champion Codes

Yang-Hui He[2,4,5], Alexander M. Kasprzyk[3], Q Le[1,*], Dmitrii Riabchenko[2]

Abstract Linear error-correcting codes form the mathematical backbone of modern digital communication and storage systems, but identifying champion linear codes (linear codes achieving or exceeding the best known minimum Hamming distance) remains challenging. By training a transformer to predict the minimum Hamming distance of a class of linear codes and pairing it with a genetic algorithm over the search space, we develop a novel method for discovering champion codes. This model effectively reduces the search space of linear codes needed to achieve a best minimum Hamming distance. Our results present the use of this method in the study and construction of error-correcting codes, applicable to codes such as generalised toric, Reed–Muller, Bose–Chaudhuri–Hocquenghem, Algebraic–Geometry, and potentially quantum codes.

Introduction

Coding theory is at the heart of modern communication. Error-correcting codes – used to detect and correct errors in data transmitted over Wi-Fi and LAN networks [1], under transatlantic waters [2], and across deep space [3, 4] – are a key part of coding theory. Thus, the search for more efficient error-correcting codes is of great importance.

A crucial parameter determining an error-correcting code's capabilities is its minimum Hamming distance. Codes with the best minimum distance among codes with the same parameters are called *champion codes*; they are recorded in the Code Tables database [5]. Finding champion codes is computationally challenging, and is an NP-hard problem [6].

In this paper, we present a novel, AI-driven method for discovering champion codes, summarised in Figure 1.

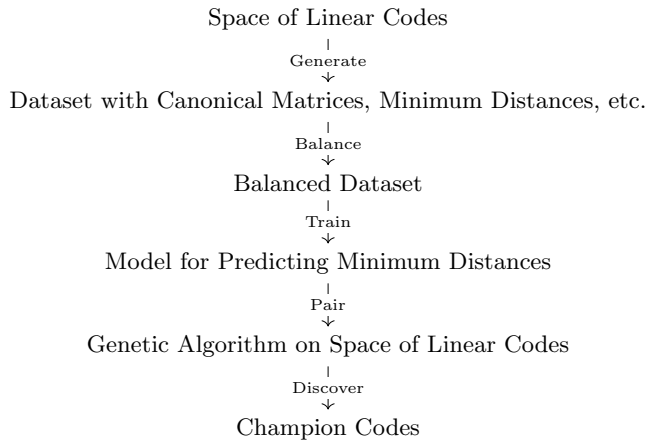


Figure 1: A general method for the discovery of champion error-correcting codes.

To show the efficacy of this method, we restrict ourselves to a class of error-correcting linear codes called generalised toric codes, introduced by Hansen [7], though there are many other classes that can adapt to this paradigm [8–10]. We use a transformer architecture, adapted from a simplified version of the OpenAI GPT-2 model, to train a predictive model for the minimum distance d_V of a generalised toric code over $\mathbb{F}_7$, achieving around 91.6% accuracy within a tolerance ($d_V \pm 3$) with mean absolute error 1.05 on the test set. Running this with a genetic algorithm, we rediscover champion codes initially found by [11]. Using a similar method over $\mathbb{F}_8$, on a similar-sized dataset, we find over 500 champion codes and can confirm at least 6 previously unknown champion codes.

Our method builds on previous work [11, 12], which classified all generalised toric codes over $\mathbb{F}_3$ to $\mathbb{F}_7$. Their methods failed to be extended to $\mathbb{F}_8$ due to the exponential computational cost of enumerating all candidate codes and calculating the minimum distance in each case.

Our work follows a tradition of applying the techniques of data science and machine learning to purely mathematical data. This tradition has origins in the study of string theory and algebraic geometry [13], and has developed to other areas of mathematics; for example [14–22]. Though machine learning does not guarantee optimal solutions to all problems, our investigation shows a successful application of machine learning to an NP-hard problem [23] and a new methodology to discover champion codes.

Methods

Generalised Toric Codes In this study, we focus on generalised toric codes, an extension of non-generalised toric codes [7, 24, 25] which had proven useful for identifying champion codes [26]. The reader is referred to Section 1 of the *Supplementary Information* for some rudiments of coding theory as well as the standard notation.

Consider the planar lattice grid $[0, q-2]^2$ over the finite field $\mathbb{F}_q$ for a prime power q and primitive element $\xi \in \mathbb{F}_q$. For all $0 \leq i, j \leq q-2$, define $P_{i,j} = (\xi^i, \xi^j) \in (\mathbb{F}_q^*)^2$. For each $u = (u_1, u_2) \in [0, q-2]^2$, define the mapping

$$\begin{aligned} &e(u) : \mathbb{F}_q^* \times \mathbb{F}_q^* \longrightarrow \mathbb{F}_q \\ &P_{i,j} = (\xi^i, \xi^j) \longrightarrow e(u)(P_{i,j}) = (\xi^i)^{u_1} (\xi^j)^{u_2}. \end{aligned} \tag{1}$$

Definition 1. Let $V = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ be a set of vertices from $[0, q-2]^2$. The generalised toric code $C_V(\mathbb{F}_q)$ over the field $\mathbb{F}_q$ associated to V is the linear code of block length $n = (q-1)^2$, dimension $k = \dim C_V(\mathbb{F}_q)$, and minimum Hamming distance $d = d_V(\mathbb{F}_q)$ spanned by the vectors in

$$\left\{ (e(u)(P_{i,j}))_{0 \leq i,j \leq q-2} = (\xi^{iu_1 + ju_2})_{0 \leq i,j \leq q-2} \ : \ u \in V \right\}. \tag{2}$$

Where q is specified, we omit the field $\mathbb{F}_q$ from notation.

*Corresponding author: q.le@trinity.ox.ac.uk
[1]Mathematical Institute, University of Oxford, OX2 6GG, UK
[2]Dept. Math., City St George's, Univ. of London, EC1V 0HB, UK
[3]Mathematics Institute, University of Warwick, CV4 7AL, UK
[4]London Inst. for Math. Sci., Royal Institution, W1S 4BS, UK
[5]Merton College, University of Oxford, OX1 4JD, UK

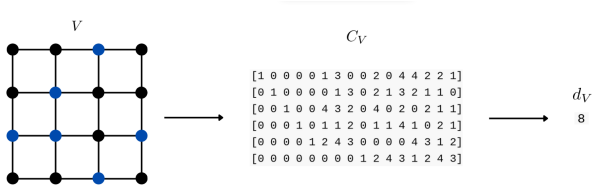


Figure 2: Example of generalised toric code C_V over $\mathbb{F}_5$, with $V = \big((0,1),(1,1),(1,2),(2,0),(2,3),(3,1)\big)$ and $d_V = 8$.

If we restrict ourselves to non-generalised toric codes, we can connect them to so-called toric varieties in mathematics [7]. In particular, studies have used toric varieties, their cohomology, and intersection theory to obtain bounds for the minimum distance of toric codes [7, 27, 28]. Although there are more tools to access by restricting to non-generalised toric codes, we consider generalised toric codes for two reasons: constructing datasets for generalised toric codes is computationally cheaper; generalised toric codes have been used to discover champion codes [11].

Here is the general method we employ to find champion codes over a given space:

1. Consider a space of linear codes that have an evolvable parameter space, as explained in the section 'Evolving champion codes'.
2. Generate a dataset of each linear code's canonical (generator and parity-check) matrices, minimum distance, and other useful properties worth training.
3. Balance this dataset to allow for training and testing datasets.
4. Train a model to predict the minimum distance of a code with this balanced dataset.
5. Pair this model with a genetic algorithm that evolves over the space of linear codes.
6. Run the genetic algorithm several times and discover new error-correcting codes, possibly finding champion codes.

Let us apply this to the space of generalised toric codes.

Machine Learning the Minimum Hamming Distance The problem of computing the minimum distance of a code is an NP-hard problem and is typically done using the Brouwer-Zimmermann (BZ) algorithm [29, 30]. Despite several improvements, the BZ algorithm remains computationally costly, having polynomial in k and exponential in q complexity [31]. This suggests the use of transformers [32], a powerful approach for sequence modelling tasks used in Natural Language Processing tasks, such as by OpenAI for ChatGPT [33]. Transformers have also been used for sentiment analysis [34], a conceptually similar task to what we consider here: given a sequence of elements (a sequence of text tokens or rows of the code[1]), a class needs to be predicted to it (the sentiment of a given text or a minimum Hamming distance).

To train this model[2], we constructed a dataset consisting of tuples of a code's canonical generator and parity-check matrices G and P, and corresponding minimum distance d.

[1] Linear codes can be represented by canonical generator or parity-check matrices, which can be viewed as sequences of rows.

[2] See GitHub for code, written using SageMath, Magma, and PyTorch. Further details of the models can be found in Section 2 of *Supplementary Information*

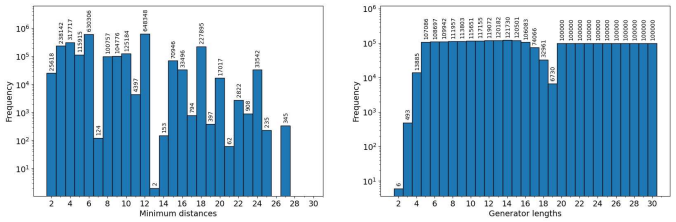


Figure 3: Histograms of the $\mathbb{F}_7$ codes dataset. The left histogram shows the distribution of minimum distances. The right histogram shows the distribution of generators' dimensions.

Toric Code Datasets The dataset generated for codes over $\mathbb{F}_7$ contains 2,700,000 codes. In Figure 3, observe the significant variation in the frequencies of minimum distances, ranging from two to more than 600,000. As the dataset is unbalanced, splitting the data into train and test sets randomly would be inadequate. Therefore, we split subsets with equal minimum distance into train and test sets separately and merged them into full train and test sets with oversampling to 500 or downsampling to 50,000 if necessary, and re-shuffling afterwards. The resulting dataset of size around 550,000 was used for training with 9:1 train/test split.

For $\mathbb{F}_8$ case, we generated a smaller dataset with 1,584,099 codes. The complexity for predicting codes over $\mathbb{F}_8$ is typically larger than over $\mathbb{F}_7$, and the amount of data we collected was marginally smaller than for $\mathbb{F}_7$, as seen in Figure 4. To mitigate this, we used a two-stage training procedure to train the model to extract useful features of codes from the common examples in the pre-training and generalise this knowledge to all the classes in the training. Pre-training was done on a dataset with around 300,000 examples, consisting of classes with more than 10,000 examples. In the training stage, we used all the classes, down-sampling or over-sampling all available classes to 600, with a total dataset of 20,000 examples. We used 9:1 train/test split in both stages.

Model architecture We cast prediction of the minimum distance as an NLP-style sequence-classification task: the input sequences comes from interpreting the canonical generator of a code as a sequence of rows $S \in \mathbb{F}_q^{k\times(q-1)^2}$ of length $(q-1)^2$, where the number of rows is the dimension k of the code; and the class label is the minimum Hamming distance $d \in \{0,\ldots,(q-1)^2\}$. Distances of $(q-1)^2$ occur only for trivial single-row codes and are negligible, so we limit classes to $\{0,\ldots,(q-1)^2-1\}$.

Starting from code available at [35], which is a stripped-down GPT-2 [36], we adapted it to work from sequence-to-sequence to

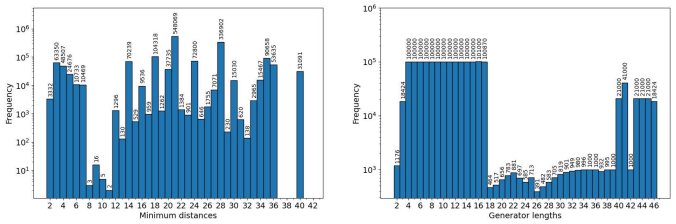


Figure 4: Histograms of the $\mathbb{F}_8$ codes dataset. The left histogram shows the distribution of minimum distances. The right histogram shows the distribution of generators' dimensions.

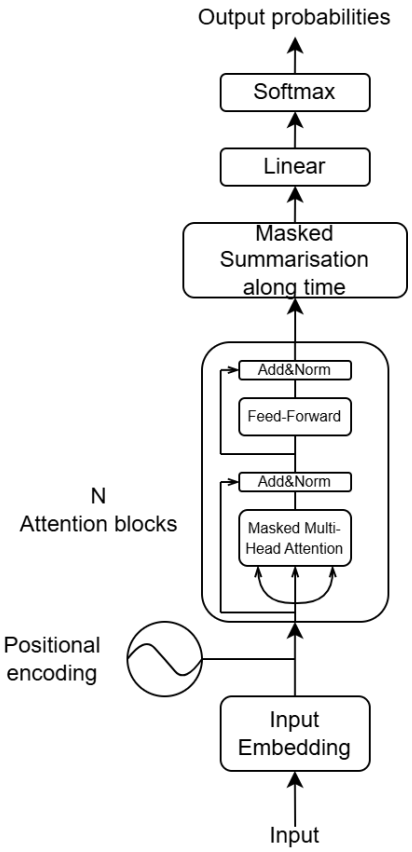


Figure 5: Toric Transformer - model architecture.

sequence classification. The adapted model uses: embeddings for field elements (none for $\mathbb{F}_7$, 4-dimensional vectors for $\mathbb{F}_8$); sinusoidal positional encodings of rows; two layers of attention blocks, each with masked self-attention and a feed-forward neural network; masked attention summarising information from each row; and a soft-max transformation to produce probabilities for minimum-distance classes[3].

Results

Training and Performance To measure performance, we report the proportion of estimates falling within a 3-unit range (± 3) of the actual minimum distance (which we will call accuracy for short), alongside the mean absolute error (MAE) and mean squared error (MSE) per distance class.

Over $\mathbb{F}_7$, our model's train set achieved overall 1.04 MAE and 91.9% accuracy. Per class MAE and MSE can be seen in Figure 6. On the test set, consisting of around 61,000 examples, our model achieved 1.05 MAE and 91.6% accuracy. Notice that for all classes, the MAE is less than 3 and most MSEs roughly equal the square of the MAEs, indicating only a small number of larger errors are present in the model's predictions.

The $\mathbb{F}_8$ model's post-train stage achieved overall MAE 1.09 and 93.4% accuracy on the train set and MAE 1.21 and accuracy 92.4% on the test set. Figure 7 shows per class MAE and MSE measurements. Although these numbers trail the $\mathbb{F}_7$ results slightly, they remain competitive, and most distance classes stay well within the ± 3-error target. Larger deviations are confined to a few classes ($d = 7, 14, 20, 21, 28$), showing opportunities for future improvement.

[3]See Supplementary Material for hyper-parameters used, the general structure of the network, and for more detailed discussion

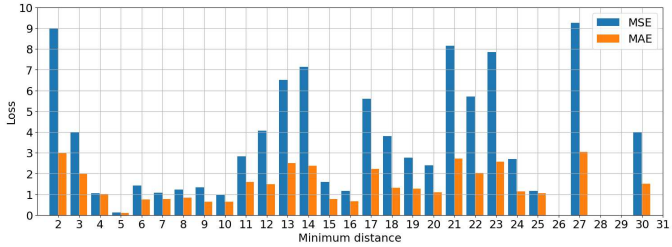


Figure 6: Losses on a test set for $\mathbb{F}_7$ codes.

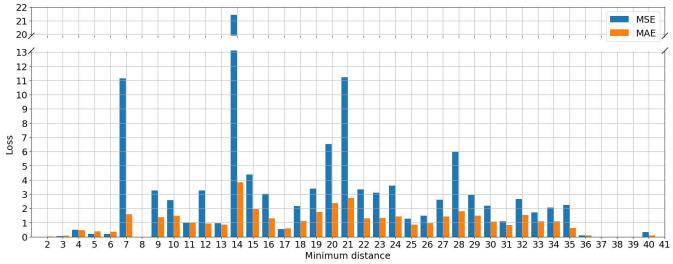


Figure 7: Losses on a test set for $\mathbb{F}_8$ codes.

Evolving Champion Codes Let us now couple our predictive model with a genetic algorithm to hunt for optimal generalised toric codes by dimension. Repeated runs over $\mathbb{F}_7$ rediscover champion codes seen in [11], and the same approach over $\mathbb{F}_8$ produces new champion linear codes.

Genetic Algorithms The following may be adapted to other classes of codes, including Reed–Muller codes [37, 38], Bose–Chaudhuri–Hocquenghem codes [39, 40], and Algebraic-Geometry codes [41, 42].

Fix a number n and a field $\mathbb{F}_q$, and consider a population of subsets of n vertices $\{V_i\}$, treating each n-vertex set V_i as a chromosome and each vertex (x_i, y_i) as a gene. With the fitness function of our genetic algorithm[4] as our model's predicted minimum distance, each generation[5] proceeds as follows:

1. Selection: Stochastic universal sampling picks 200 parents from a population of 300.

2. Crossover: Flatten each parent's vertex set $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ into a list $[x_1, y_1, \ldots, x_n, y_n]$. A random crossover split point made between vertex pairs swaps the tails; offspring with duplicate vertices are discarded and we repeat with other parents.

3. Mutation: With 10% probability, a vertex is replaced by a new, unused one, preserving n vertices.

4. Elitism: The best 30 solutions carry forward unchanged, with the rest discarded.

We repeat for 200 generations.

Note that we evolve vertex sets rather than generator matrices because there is no clear way to mutate the generator matrices and preserve the type of the linear code (in this case, generalised toric code), except to mutate the parameters from which they arise. If this can occur, then the space of linear codes is said to have an evolvable parameter space. Spaces of this sort include: generalised toric codes; Reed-Muller codes; Bose–Chaudhuri–Hocquenghem codes; and Algebraic-Geometry codes. It could be possible that certain classes of quantum codes also have an evolvable parameter space.

[4]The genetic algorithms we used can be found here, [43].
[5]Guided by a single parameter test with the models.

We target champion codes of dimension k by searching only k-vertex sets and penalising candidates whose dimension drifts from k. Additional penalties[6] prevent rediscovering codes found in earlier runs. This leads to the resulting fitness function, used both over $\mathbb{F}_7$ and $\mathbb{F}_8$:

Algorithm 1 Fitness Function

Require: Target dimension k, set of k vertices V, set of sets of k vertices S_k corresponding to codes found by the genetic algorithm so far.

Ensure: Fitness function f which implements the penalties described above.

1. If $V \in S_k$, return fitness score of $f = 10$.
2. Else, calculate the dimension $k_V = \dim C_V(\mathbb{F}_q)$.
3. Predict minimum distance of code d_{approx} using model.
4. Return fitness score of $f = 300 + d_{approx} - |k_V - k|$.

During GA search, we predict each code's distance with our model (d_{approx}), then confirm promising candidates with the expensive BZ brute-force algorithm. Since distance evaluation is the exploration's bottleneck, and verification is affordable over $\mathbb{F}_7$ but much costlier for $\mathbb{F}_8$, we apply the BZ check more selectively over $\mathbb{F}_8$.

Champion Codes over $\mathbb{F}_7$ We apply our $\mathbb{F}_7$ model to rediscover the best generalised toric codes over $\mathbb{F}_7$ from [11]. To this aim, we use Algorithm 2.

Algorithm 2 Discovering Best Codes over $\mathbb{F}_7$ for Dimensions $3 \leq k \leq 34$

Require: Total number of runs so far, T; list of best known minimum Hamming distances for generalised toric codes over $\mathbb{F}_7$, L_b; list of the best found distances discovered by genetic algorithm so far for each dimension, $L_d = (d_1, d_2, \ldots, d_{36})$; set of discovered codes of dimension i, S_i.

Ensure: Recurring algorithm to discover best generalised toric codes over $\mathbb{F}_7$.

1. If $T = 0$, then L_d and S_i $(3 \leq i \leq 34)$ are empty. For $3 \leq i \leq 34$, apply genetic algorithm to generate codes of dimension i. Place codes into S_i, calculate their Hamming distances, place best discovered minimum distances into L_k, and increase T by 1.
2. If $T \neq 0$, then consider $3 \leq i \leq 34$.
3. If $L_k[i] \neq L_b[i]$, then apply genetic algorithm to generate codes of dimension i. Place codes in S, calculate their minimum distances, and update L_d if necessary.
4. Increase T by 1.

Across 757 GA runs (see Table 1), most dimensions yielded a best-distance code on the first attempt. Dimensions 12 and 15 needed more iterations, indicating a tougher search landscape. Dimension 19 remains elusive after 501 runs, pointing to the rarity of $[36, 19, 12]$ generalised toric codes likely coupled with a tougher search landscape.

Comparing Tables 1 and 2 shows a loose correlation: dimensions that contain more best codes in the dataset usually need fewer GA runs to rediscover them. This pattern fades at the smallest and largest dimensions, where optimal codes are naturally easier to locate than those in the 12–28 range. Thus,

[6] The number 300 in the formula is simply a way to offset the other two terms and make the output of the formula positive as required by the PyGAD library we used [44].

Dimension	No. of Runs
3	1
4	1
5	1
6	1
7	2
8	1
9	20
10	1
11	3
12	54
13	1
14	2
15	81
16	1
17	1
18	20
19	N/A
20	1
21	22
22	18
23	1
24	30
25	1
26	1
27	1
28	43
29	1
30	1
31	1
32	1
33	1
34	1

Table 1: Dimension of code vs. No. of runs to find best minimum Hamming distance

dimensions 12, 15, 19, and 28 converged more slowly because the dataset held few optimal-code examples at those sizes (this is amplified for dimensions 19 and 28 by the rarity of examples in the training set). Notably, though there were no best codes of dimension 12 in the training and testing datasets, our method still found a best generalised toric code of dimension 12. We see this more apparently over $\mathbb{F}_8$, where we discover new champion codes.

Champion Codes over $\mathbb{F}_8$ Because champion distances over $\mathbb{F}_8$ are largely unknown, we screen candidates against the best-known lower bounds b_k from [5]. For a code C of dimension k, we call Magma's `VerifyMinimumDistanceLowerBound`(C, b_k). If the BZ routine shows $d_C < b_k$, we discard C immediately; otherwise we keep it for a full distance check. This filter saves considerable computation by focusing effort only on codes that *could* beat the current records. Thus, we use Algorithm 3.

We performed one run[7][8] of the above algorithm. Despite this, we found over 700 possibly new champion codes.

As with $\mathbb{F}_7$, we compare these results to the number of champion codes found in the dataset on which the $\mathbb{F}_8$ model trained.

Once again, champion codes were discovered at the lowest and highest dimensions for similar reasons as over $\mathbb{F}_7$. Rather surprisingly, in only one run, we achieved champion codes in

[7] We had run the algorithm twice for vertices between 20 and 29. In this search, we discovered a champion code of dimension 22, as seen in Table 4. Upon discovery, we saved this code. Due to a file error, we were unable to save the corresponding dataset that included this code.

[8] This is due to time and computational constraints.

Dimension	No. of Best
3	98
4	332
5	23446
6	4023
7	115
8	564
9	27
10	2482
11	124
12	0
13	411
14	16
15	1
16	11838
17	1951
18	66
19	0
20	1094
21	39
22	57
23	336
24	5
25	45714
26	21432
27	4215
28	80
29	2420
30	33302
31	10713
32	0
33	0
34	0

Table 2: Dimension of code vs. No. of best codes in the dataset over $\mathbb{F}_7$

dimensions that had no champion codes in their datasets: dimensions 18, 22, and 38. Through explicit calculation compared with the known best codes in [5], we see that all codes of dimension 18, 22, and 38 that have been discovered are in fact novel.

Discussion

Our method provides a new tool for the construction of error-correcting champion codes, and can grow stronger on several fronts. Larger and more balanced datasets would give predictive models richer coverage, especially at rare distances. Broader hyper-parameter searches (optimisers, mixed cross-entropy and Wasserstein losses, alternative row/field encodings) and systematic embedding design could further improve accuracy.

The GA itself invites tuning: a wider sweep of population sizes, crossover strategies, and penalty weights, plus multi-week runs over $\mathbb{F}_8$ with staged Magma [45] checks, should surface additional champions. It may also be fruitful to consider other ways to investigate the parameter space [46].

Beyond $\mathbb{F}_7$ and $\mathbb{F}_8$, the same framework can progress over higher fields of prime powers ($\mathbb{F}_9$, $\mathbb{F}_{11}$, $\mathbb{F}_{13}$), expand to non-generalised toric codes that can exploit algebraic and geometric properties arising from their underlying toric varieties [7, 28, 47], or even construct possible champion codes from other families (BCH, Reed–Muller, AG, and more), once suitable GA "genes" are defined[9].

[9] Vaessens et al. [48] give a general GA scheme for building large codes, but this would be unsuitable for searching for codes of a fixed dimension as the fitness function would require a dimension penalty, as done in Algorithm 1.

Algorithm 3 Discovering New Champion Codes over $\mathbb{F}_8$ for Dimensions $3 \leq k \leq 46$

Require: Total number of runs so far, T; list of lower bounds for minimum Hamming distances distances of champion codes over $\mathbb{F}_8$, L_b; set of discovered codes of dimension i, S_i.

Ensure: Recurring algorithm to search for champion generalised toric codes over $\mathbb{F}_8$.

1. Begin with S_i $(3 \leq i \leq 46)$ as empty sets.
2. For $3 \leq i \leq 46$, apply genetic algorithm to generate codes of dimension i.
3. Place codes into S_i, verify whether they are possible champion codes or not, tag the possible champion codes, and increase T by 1.
4. Repeat until you wish to finish.
5. Once finished, for tagged possible champion codes, apply BZ algorithm to compute their minimum distances.
6. Verify possible champion codes against known bounds.

Finally, recasting the task as image classification with padded matrices or applying representation-learning techniques from NLP [49] offer alternative modelling avenues worth exploring.

Data availability Our datasets are available from GitHub.

Code availability All code required to replicate the results in this paper is available from GitHub under an MIT license.

Correspondence Correspondence and requests for materials should be addressed to Q Le.

References

1. Kadel, R., Paudel, K., Guruge, D. B. & Halder, S. J. Opportunities and challenges for error control schemes for wireless sensor networks: A review. *Electronics* **9,** 504. doi:10.3390/electronics9030504 (2020).
2. Pamart, J. *et al.* Forward error correction in a 5 Gbit/s 6400 km EDFA based system. *Electronics Letters* **30,** 342–343. doi:10.1049/el:19940242 (1994).
3. Fredricksen, H. *Error correction for deep space network teletype circuits (Technical Report 32-1275)* tech. rep. (1968).
4. Heller, J. & Jacobs, I. Viterbi decoding for satellite and space communication. *IEEE Transactions on Communication Technology* **19,** 835–848. doi:10.1109/TCOM.1971.1090711 (1971).
5. Grassl, M. *Bounds on the minimum distance of linear codes and quantum codes* Online, available at https://www.codetables.de.
6. Vardy, A. The intractability of computing the minimum distance of a code. *IEEE Trans. Inform. Theory* **43,** 1757–1766. ISSN: 0018-9448. doi:10.1109/18.641542 (1997).
7. Hansen, J. P. Toric varieties Hirzebruch surfaces and error-correcting codes. *Appl. Algebra Engrg. Comm. Comput.* **13,** 289–300. ISSN: 0938-1279. doi:10.1007/s00200-002-0106-0 (2002).

8. Reed, I. S. & Solomon, G. Polynomial codes over certain finite fields. *J. Soc. Indust. Appl. Math.* **8,** 300–304. ISSN: 0368-4245 (1960).
9. Gallager, R. G. Low-density parity-check codes. *IRE Trans.* **IT-8,** 21–28. doi:10.1109/tit.1962.1057683 (1962).
10. Arıkan, E. Channel polarization: a method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *IEEE Trans. Inform. Theory* **55,** 3051–3073. ISSN: 0018-9448. doi:10.1109/TIT.2009.2021379 (2009).
11. Brown, G. & Kasprzyk, A. M. Seven new champion linear codes. *LMS J. Comput. Math.* **16,** 109–117. doi:10.1112/S1461157013000041 (2013).
12. Brown, G. & Kasprzyk, A. M. Small polygons and toric codes. *J. Symbolic Comput.* **51,** 55–62. ISSN: 0747-7171. doi:10.1016/j.jsc.2012.07.001 (2013).
13. He, Y.-H. Machine-learning the string landscape. *Phys. Lett. B* **774,** 564–568. doi:10.1016/j.physletb.2017.10.024 (2017).
14. Davies, A. *et al.* Advancing mathematics by guiding human intuition with AI. *Nature* **600,** 70–74. doi:10.1038/s41586-021-04086-x (2021).
15. Wu, Y. & De Loera, J. A. *Turning Mathematics Problems into Games: Reinforcement Learning and Gröbner bases together solve Integer Feasibility Problems* 2022. arXiv: 2208.12191 [cs.LG].
16. Coates, T., Kasprzyk, A. M. & Veneziale, S. Machine learning the dimension of a Fano variety. *Nat. Commun.* **14.** doi:10.1038/s41467-023-41157-1 (2023).
17. Coates, T., Kasprzyk, A. M. & Veneziale, S. *Machine learning detects terminal singularities* in *Advances in Neural Information Processing Systems* (eds Oh, A. *et al.*) **36** (2023), 67183–67194.
18. He, Y.-H. AI-driven research in pure mathematics and theoretical physics. *Nature Reviews Physics* **6,** 546–553. doi:10.1038/s42254-024-00740-1 (2024).
19. Gukov, S., Halverson, J. & Ruehle, F. Rigor with machine learning from field theory to the Poincaré conjecture. *Nature Rev. Phys.* **6,** 310–319. doi:10.1038/s42254-024-00709-0. arXiv: 2402.13321 [hep-th] (2024).
20. Yau, M., Karalias, N., Lu, E., Xu, J. & Jegelka, S. *Are Graph Neural Networks Optimal Approximation Algorithms?* in *Advances in Neural Information Processing Systems* (eds Globerson, A. *et al.*) **37** (2024), 73124–73181.
21. Berglund, P. *et al.* New Calabi–Yau manifolds from genetic algorithms. *Phys. Lett. B* **850,** Paper No. 138504, 10. ISSN: 0370-2693. doi:10.1016/j.physletb.2024.138504 (2024).
22. Abramson, J. *et al.* Accurate structure prediction of biomolecular interactions with AlphaFold 3. *Nature* **630,** 493–500. doi:10.1038/s41586-024-07487-w (2024).
23. Ardon, L. *Reinforcement learning to solve NP-hard problems: an application to the CVRP* 2022. arXiv: 2201.05393 [cs.AI].
24. Hansen, J. P. in *Coding theory, cryptography and related areas (Guanajuato, 1998)* 132–142 (Springer, Berlin, 2000).
25. Little, J. B. Remarks on generalized toric codes. *Finite Fields Appl.* **24,** 1–14. ISSN: 1071-5797. doi:10.1016/j.ffa.2013.05.004 (2013).
26. Ruano, D. On the structure of generalized toric codes. *J. Symbolic Comput.* **44,** 499–506. ISSN: 0747-7171. doi:10.1016/j.jsc.2007.07.018 (2009).
27. Joyner, D. Toric codes over finite fields. *Appl. Algebra Engrg. Comm. Comput.* **15,** 63–79. ISSN: 0938-1279. doi:10.1007/s00200-004-0152-x (2004).
28. Soprunov, I. & Soprunova, J. Toric surface codes and Minkowski length of polygons. *SIAM J. Discrete Math.* **23,** 384–400. ISSN: 0895-4801. doi:10.1137/080716554 (2008).
29. Zimmermann, K.-H. *Integral hecke modules, integral generalized Reed–Muller codes, and linear codes* tech. rep. (Techn. Univ. Hamburg-Harburg, 1996).
30. Grassl, M. in *Discovering mathematics with Magma* 287–313 (Springer, Berlin, 2006). doi:10.1007/978-3-540-37634-7_13.
31. Hernando, F., Igual, F. D. & Quintana-Ortí, G. Algorithm 994: fast implementations of the Brouwer-Zimmermann algorithm for the computation of the minimum distance of a random linear code. *ACM Trans. Math. Software* **45,** Art. 23, 28. ISSN: 0098-3500. doi:10.1145/3302389 (2019).
32. Vaswani, A. *et al. Attention is all you need* in *Advances in Neural Information Processing Systems* (eds Guyon, I. *et al.*) **30** (2017).
33. Brown, T. *et al. Language models are few-shot learners* in *Advances in Neural Information Processing Systems* (eds Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. & Lin, H.) **33** (2020), 1877–1901.
34. Barbieri, F., Camacho-Collados, J., Neves, L. & Espinosa-Anke, L. *TweetEval: Unified benchmark and comparative evaluation for tweet classification* 2020. arXiv: 2010.12421 [cs.CL].
35. Karpathy, A. *minGPT* https://github.com/karpathy/minGPT. 2022.
36. Radford, A. *et al.* Language Models are Unsupervised Multitask Learners (2019).
37. Muller, D. E. Application of Boolean algebra to switching circuit design and to error detection. *Transactions of the I.R.E. Professional Group on Electronic Computers* **EC-3,** 6–12. doi:10.1109/IREPGELC.1954.6499441 (1954).
38. Reed, I. S. A class of multiple-error-correcting codes and the decoding scheme. *Trans. IRE,* 38–49 (1954).
39. Hocquenghem, A. Codes correcteurs d'erreurs. *Chiffres* **2,** 147–156. ISSN: 0245-9922 (1959).
40. Bose, R. C. & Ray-Chaudhuri, D. K. On a class of error correcting binary group codes. *Information and Control* **3,** 68–79. ISSN: 0019-9958 (1960).
41. Couvreur, A. & Randriambololona, H. in *Concise encyclopedia of coding theory* 307–361 (CRC Press, Boca Raton, FL, 2021).
42. Goppa, V. D. Algebraico–geometric codes. *Mathematics of the USSR-Izvestiya* **21,** 75–91. doi:10.1070/IM1983v021n01ABEH001641 (1983).
43. Le, Q. & Riabchenko, D. *Genetic Algorithms* https://github.com/QTVLe/MachineLearningGeneralisedToricCodes. 2025.

44. Gad, A. F. PyGAD: An intuitive genetic algorithm Python library. *Multimed. Tools Appl.* **83,** 58029–58042. doi:10.1007/s11042-023-17167-y (2024).

45. Bosma, W., Cannon, J. & Playoust, C. The Magma algebra system. I. The user language. *J. Symbolic Comput.* **24.** Computational algebra and number theory (London, 1993), 235–265. doi:10.1006/jsco.1996.0125 (1997).

46. Sipper, M., Fu, W., Ahuja, K. & Moore, J. H. Investigating the parameter space of evolutionary algorithms. *BioData Mining* **11,** 2. ISSN: 1756-0381. doi:10.1186/s13040-018-0164-x (2018).

47. Little, J. & Schenck, H. Toric surface codes and Minkowski sums. *SIAM J. Discrete Math.* **20,** 999–1014. ISSN: 0895-4801. doi:10.1137/050637054 (2006).

48. Vaessens, R. J. M., Aarts, E. H. L. & van Lint, J. H. Genetic algorithms in coding theory—a table for $A_3(n, d)$. *Discrete Appl. Math.* **45,** 71–87. ISSN: 0166-218X. doi:10.1016/0166-218X(93)90141-A (1993).

49. Bengio, Y., Courville, A. & Vincent, P. Representation learning: A review and new perspectives. *IEEIEEE Trans. Pattern Anal. Mach. Intell.* **35,** 1798–1828. doi:10.1109/TPAMI.2013.50 (2013).

Acknowledgements

AK is funded by EPSRC Fellowship EP/N022513/1. QL is funded by a Mitchell Fund award from Trinity College, Oxford.

Author contributions

All authors contributed equally to this work.

Competing interests

The authors declare no competing interests.

Dim	No. of Champs
1	0
2	0
3	16464
4	31091
5	0
6	1034
7	216
8	28
9	3
10	11
11	1
12	144
13	1
14	0
15	146
16	1

Dim	No. of Champs
17	0
18	0
19	0
20	0
21	0
22	0
23	0
24	0
25	0
26	0
27	0
28	0
29	0
30	0
31	0
32	0

Dim	No. of Champs
33	0
34	0
35	0
36	0
37	0
38	0
39	0
40	6138
41	357
42	0
43	0
44	5177
45	0
46	16464
47	0
48	1176

Table 3: Dimension of Code vs. Number of Champion Codes in Training and Testing Datasets over $\mathbb{F}_8$.

Table 4: Champion codes found over $\mathbb{F}_8$, with block length $n = 49$, dimension k, and best minimum distance found d, and number of such codes discovered N.

k	d	N	**Example Vertices**
3	42	231	(1, 0), (2, 3), (6, 3)
4	40	66	(0, 5), (2, 6), (6, 0), (6, 4)
6	36	3	(0, 0), (0, 6), (2, 4), (2, 5), (4, 6), (6, 4)
7	35	1	(0, 3), (2, 5), (4, 6), (5, 0), (5, 1), (6, 0), (6, 3)
8	34	1	(0, 4), (0, 6), (2, 3), (3, 4), (4, 0), (5, 1), (5, 3), (6, 0)
18	21	4	(0, 1), (0, 2), (0, 3), (0, 5), (1, 0), (1, 1), (1, 5), (1, 6), (3, 2), (3, 3), (3, 5), (3, 6), (4, 3), (4, 4), (5, 1), (5, 3), (5, 5), (6, 4)
22	18	1	(0, 0), (0, 6), (1, 0), (1, 3), (1, 4), (1, 5), (2, 0), (2, 5), (2, 6), (3, 2), (3, 3), (3, 5), (3, 6), (4, 2), (5, 0), (5, 3), (5, 4), (6, 0), (6, 1), (6, 3), (6, 4), (6, 5)
38	7	1	(0, 1), (0, 3), (0, 4), (0, 6), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (2, 0), (2, 1), (2, 3), (2, 4), (2, 6), (3, 0), (3, 1), (3, 3), (3, 4), (3, 5), (3, 6), (4, 0), (4, 1), (4, 2), (4, 3), (4, 4), (4, 5), (4, 6), (5, 0), (5, 1), (5, 3), (5, 4), (5, 5), (5, 6), (6, 0), (6, 1), (6, 2), (6, 5)
40	6	94	(0, 0), (0, 2), (0, 3), (0, 4), (0, 5), (0, 6), (1, 1), (1, 2), (1, 5), (1, 6), (2, 0), (2, 1), (2, 2), (2, 3), (2, 4), (2, 5), (2, 6), (3, 0), (3, 1), (3, 2), (3, 3), (3, 4), (3, 5), (3, 6), (4, 1), (4, 2), (4, 4), (4, 5), (4, 6), (5, 1), (5, 2), (5, 3), (5, 4), (5, 5), (5, 6), (6, 0), (6, 1), (6, 4), (6, 5), (6, 6)
41	6	3	(0, 0), (0, 1), (0, 2), (0, 4), (0, 5), (0, 6), (1, 0), (1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (2, 0), (2, 1), (2, 2), (2, 4), (2, 6), (3, 0), (3, 1), (3, 2), (3, 3), (3, 4), (3, 5), (3, 6), (4, 0), (4, 1), (4, 4), (4, 6), (5, 0), (5, 1), (5, 2), (5, 3), (5, 5), (5, 6), (6, 0), (6, 1), (6, 2), (6, 3), (6, 4), (6, 5), (6, 6)
44	4	100	(0, 0), (0, 1), (0, 2), (0, 3), (0, 4), (0, 5), (0, 6), (1, 0), (1, 1), (1, 3), (1, 4), (1, 5), (1, 6), (2, 0), (2, 1), (2, 2), (2, 3), (2, 4), (2, 5), (2, 6), (3, 0), (3, 1), (3, 4), (3, 5), (3, 6), (4, 0), (4, 1), (4, 2), (4, 3), (4, 4), (4, 6), (5, 0), (5, 1), (5, 2), (5, 3), (5, 4), (5, 5), (6, 0), (6, 1), (6, 2), (6, 3), (6, 4), (6, 5), (6, 6)
46	3	235	(0, 0), (0, 1), (0, 2), (0, 3), (0, 4), (0, 5), (0, 6), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (2, 1), (2, 2), (2, 3), (2, 4), (2, 5), (2, 6), (3, 0), (3, 1), (3, 2), (3, 3), (3, 4), (3, 6), (4, 0), (4, 1), (4, 2), (4, 3), (4, 4), (4, 5), (4, 6), (5, 1), (5, 2), (5, 3), (5, 4), (5, 5), (5, 6), (6, 0), (6, 1), (6, 2), (6, 3), (6, 4), (6, 5), (6, 6)

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- SupplementaryMaterial.pdf