https://doi.org/10.1038/s44387-026-00077-3

Machine learning discovers new champion codes

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

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 champion codes. 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, algebrogeometric, and potentially quantum codes.

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, and the search for more efficient error-correcting codes is of great importance.

A crucial property when determining an error-correcting code's capabilities is its minimum Hamming distance. Finding new linear codes whose minimum Hamming distance equals or exceeds that of previously known codes (for fixed block length and dimension) is computationally challenging, and is an NP-hard problem[5]; these codes are called *champion codes*.

We present a new method for discovering champion linear codes, employing machine learning and genetic algorithms, as summarised in Fig. 1.

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[6], though there are many other classes that can adapt to this paradigm[7–9]. 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. Combining this model with a genetic algorithm, we rediscover champion codes initially found by[10]. 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. We compare our approach with random search and find it achieves up to a twofold reduction in BZ evaluations, improving the efficiency of champion code discovery. Here, a BZ evaluation is a use of the Brouwer-Zimmermann algorithm, the brute-force algorithm for determining the minimum Hamming distance of any linear code.

Our method builds on previous work[10,11], 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. Building on this work, we realised our method applies to any family of linear codes with an evolvable parameter space (see 'Evolving champion codes').

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[12], and has developed to other areas of mathematics; for example[13–22]. 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[6,24,25] 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 $0 \le i, j \le q-2$, 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

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

[1]Department Math City St George's, University of London, London, UK. [2]London Institute for Math Science, Royal Institution, London, UK. [3]Merton College, University of Oxford, Oxford, UK. [4]Mathematics Institute, University of Warwick, Warwick, UK. [5]Mathematical Institute, University of Oxford, Oxford, UK. ✉e-mail: qvietle2004@outlook.com

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

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

An example of this construction is illustrated in Fig. 2. Where q is specified, we omit the field $\mathbb{F}_q$ from notation.

If we restrict ourselves to non-generalised toric codes, we can connect them to toric varieties in mathematics[6]. In particular, studies have used toric varieties, their cohomology, and intersection theory to obtain bounds for the minimum distance of toric codes[6,27,28]. 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[10]. In general, we also focus on generalised toric codes for three other reasons: they underlie the work inspiring this study[10,11]; their simple parametrisation makes them suitable for genetic optimisation (see 'Genetic Algorithms'); and they provide a non-trivial test case for our general method.

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.

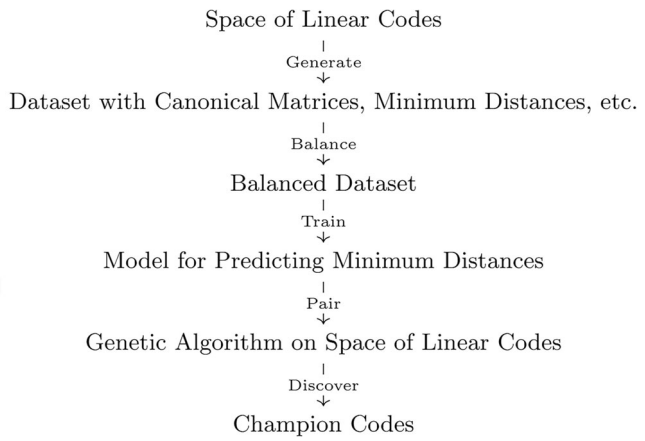

Fig. 1 | Method for ML-Assisted Discovery of Champion Codes. The flowchart shows the stages: generate a dataset of linear codes, with canonical matrices and associated minimum distances; balance the dataset for learning; train a model of minimum distance; pair that model with a genetic algorithm over a code family, and; discover champion 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].Thus, our method seeks to minimise the number of BZ evaluations required to identify new champion codes (see Table 1 and Table 2 for results).

A generalised toric code C, represented by a generator G or parity-check matrix P, can be viewed as a sequence of rows of length $(q-1)^2$, where the number of rows is the dimension k of the code. Thus, the problem can be formulated as the prediction of the Hamming minimum distance of that code, which can take integer values between 0 and $(q-1)^2$, from the input sequence S of length k (dimension of the code) of code rows, with each being a vector of length $(q-1)^2$.

This task belongs to *sequence modelling*, for which several architectures are commonly used: Convolutional Neural Networks, Recurrent Neural Networks, and Transformers. The latter two naturally handle variable-length input sequences (though transformers impose a maximum sequence length).

Since computing a code's minimum Hamming distance is NP-hard, we employ the most powerful sequence-modelling architecture currently available—*the transformer*[32]—which underlies many state-of-the-art natural language processing models such as ChatGPT[33] and sentiment analysis systems[34]. The latter task is conceptually analogous to ours: given a sequence of elements (text tokens or code rows), the goal is to predict a class label (text sentiment or minimum Hamming distance). Accordingly, we sometimes refer to codes of a given minimum distance as belonging to a class.

To train this model, 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. See GitHub for code, written using SageMath, MAGMA, and PyTorch. Further details of the models can be found in Section 2 of Supplementary Information.

Toric code datasets

The dataset generated for codes over $\mathbb{F}_7$ contains 2,700,000 codes. In Fig. 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.

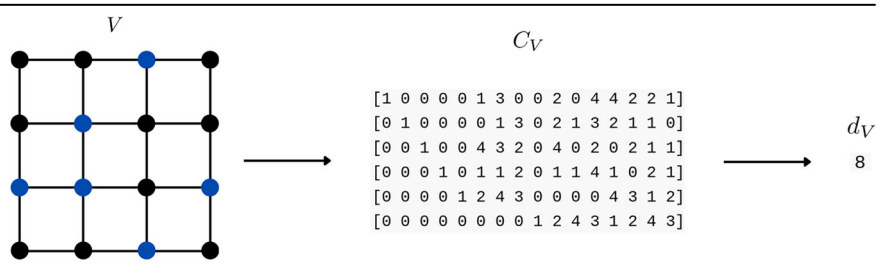

Fig. 2 | Constructing a generalised toric code from lattice data and reading off its distance. Example of generalised toric code C_V over $\mathbb{F}_5$, with $V = ((0,1),(1,1),(1,2),(2,0),(2,3),(3,1))$ and $d_V = 8$.

Table 1 | Search efficiency over $\mathbb{F}_7$ for generalised toric codes

ℓ	number of codes from ℓ vertices	number of codes of dimension $k=\ell$	number of champion codes	number of BZ evaluations per champion	number of GA runs	number of codes from GA	number of champions discovered from GA	number of BZ evaluations per champion for GA
5	107,086	86,867	19307	4.50	1	300	72	4.17
6	108,697	81,094	3126	25.94	1	300	21	14.29
7	109,942	74,451	82	907.94	2	600	1	600.0
8	111,957	67,578	368	183.64	1	300	2	150.0
9	113,803	60,257	20	3012.85	7	2100	1	2100.0
10	115,651	52,704	1210	43.56	1	300	14	21.43
11	117,155	45,315	60	755.25	3	900	1	900.0
12	119,072	38,301	0	≥38301	55	16,500	1	16500.0
13	120,182	31,447	125	251.58	1	300	1	300.0
14	121,730	25,748	2	12874.0	2	600	1	600.0
15	120,501	20,857	1	20857.0	82	24,600	1	24600.0
16	106,083	16,310	1859	8.77	1	300	32	9.38
17	74,066	12,458	350	35.59	1	300	7	42.86
18	32,961	9026	22	410.27	4	1200	1	1200.0
19	6730	6730	0	≥6730	84	25,200	0	≥25200
20	100,000	100,000	1094	91.41	1	300	4	75.0
21	100,000	100,000	39	2564.10	15	4500	1	4500.0
22	100,000	100,000	57	1754.39	18	5400	1	5400.0
23	100,000	100,000	336	297.62	1	300	1	300.0
24	100,000	100,000	5	20000.0	43	12,900	1	12900.0
25	100,000	100,000	45714	2.19	1	300	134	2.24
26	100,000	100,000	21432	4.67	1	300	59	5.08
27	100,000	100,000	4215	23.72	1	300	10	30.0
28	100,000	100,000	80	1250.0	6	1800	1	1800.0
29	100,000	100,000	2420	41.32	1	300	6	50.0
30	100,000	100,000	33302	3.00	1	300	110	2.73
31	100,000	100,000	10713	9.33	1	300	33	9.09

For each vertex count ℓ, the table reports the total generated codes, those with realised dimension $k=\ell$, the number of champion codes, and the average BZ evaluations per verified champion. The last four columns show the corresponding results for the GA-guided search, highlighting its reduced BZ evaluation cost compared with uniform or exhaustive searches in *Seven New Champion Linear Codes*[11]. When no champions were found, a lower bound n_{LB} is reported as $\geq n_{LB}$.

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 Fig. 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 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. This is illustrated in Fig. 5.

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

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 Fig. 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.

Table 2 | Search efficiency over $\mathbb{F}_8$ for generalised toric codes

ℓ	number of codes from ℓ vertices	number of champions	number of BZ evaluations per champion	maximum minimum Hamming distance bounds	number of codes from GA	number of champions from GA	number of BZ evaluations per champion for GA	maximum minimum Hamming distance bounds for GA
5	100,000	0	≥100,000	36/38	300	0	≥300	35/38
6	100,000	1034	96.71	36/36	300	3	100.0	36/36
7	100,000	216	462.96	35/35	300	1	300.0	35/35
8	100,000	28	3571.43	34/34	300	1	300.0	34/34
9	100,000	3	33333.33	31/31	300	0	≥300	16/31
10	100,000	11	9090.91	30/30	300	0	≥300	15/30
11	100,000	1	100000.00	29/29	300	0	≥300	13/29
12	100,000	144	694.44	28/28	300	0	≥300	12/28
13	100,000	1	100000.00	27/27	300	0	≥300	11/27
14	100,000	0	≥100000	24/26	300	0	≥300	12/26
15	100,000	146	684.93	24/24	300	0	≥300	16/24
16	101,000	1	101000.0	24/24	300	0	≥300	10/24
17	100,870	0	≥100870	22/23	300	0	≥300	8/23
18	464	0	≥464	17/21	300	4	75.0	21/21
19	517	0	≥517	15/21	300	0	≥300	8/21
20	656	0	≥656	15/20	300	0	≥300	9/20
21	783	0	≥783	15/19	300	0	≥300	8/19
22	881	0	≥881	14/18	300	0	≥300	11/18
23	697	0	≥697	14/17	300	0	≥300	9/17
24	585	0	≥585	13/16	300	0	≥300	10/16
25	713	0	≥713	12/16	300	0	≥300	7/16
26	391	0	≥391	10/15	300	0	≥300	5/15
27	482	0	≥482	8/14	300	0	≥300	4/14
28	583	0	≥583	7/14	300	0	≥300	4/14
29	705	0	≥705	7/13	300	0	≥300	4/13
30	819	0	≥819	7/12	0	0	≥0	0/12
31	901	0	≥901	7/12	300	0	≥300	4/12
32	949	0	≥949	7/11	300	0	≥300	4/11
33	980	0	≥980	7/11	300	0	≥300	3/11
34	996	0	≥996	7/10	300	0	≥300	4/10
35	1000	0	≥1000	7/9	300	0	≥300	4/9
36	1000	0	≥1000	7/8	300	0	≥300	5/8
37	932	0	≥932	6/8	300	0	≥300	4/8
38	995	0	≥995	6/7	300	1	300.0	7/7
39	1000	0	≥1000	6/7	300	0	≥300	3/7
40	21,000	6138	3.42	6/6	300	95	3.16	6/6
41	41,000	357	114.85	6/6	300	3	100.0	6/6
42	1000	0	≥1000	5/6	300	0	≥300	3/6
43	21,000	0	≥21,000	4/5	300	0	≥300	2/5
44	21,000	5177	4.06	4/4	300	101	2.97	4/4
45	21,000	0	≥21,000	3/4	300	0	≥300	2/4
46	18,424	16,464	1.12	3/3	300	236	1.27	3/3
47	1176	1176	≥1176	2/2	0	0	0	0/2

For each vertex count ℓ, the table lists the total generated codes, number of champions, and average BZ evaluations over champions under uniform sampling. The last three rows show corresponding results for GA-guided search. Counts are aggregated by realised dimension k after canonicalisation of the generator ($k \le \ell$ allowed). When no champions were found, a lower bound n_{LB} is reported as $\le n_{LB}$. Note that we excluded $\ell = 30$ due to the high computational cost of BZ evaluations at this dimension, and the champion code of dimension 22 is also omitted, as noted in *Champion Codes over* $\mathbb{F}_8$.

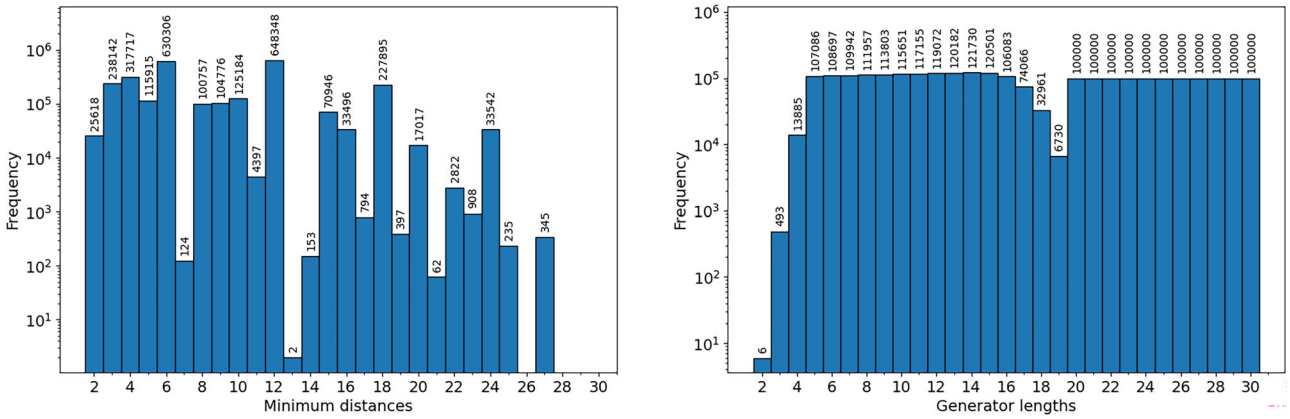

Fig. 3 | Histograms of the $\mathbb{F}_7$ codes dataset. The left histogram shows the distribution of minimum distances. The right histogram shows the distribution of generators' dimensions.

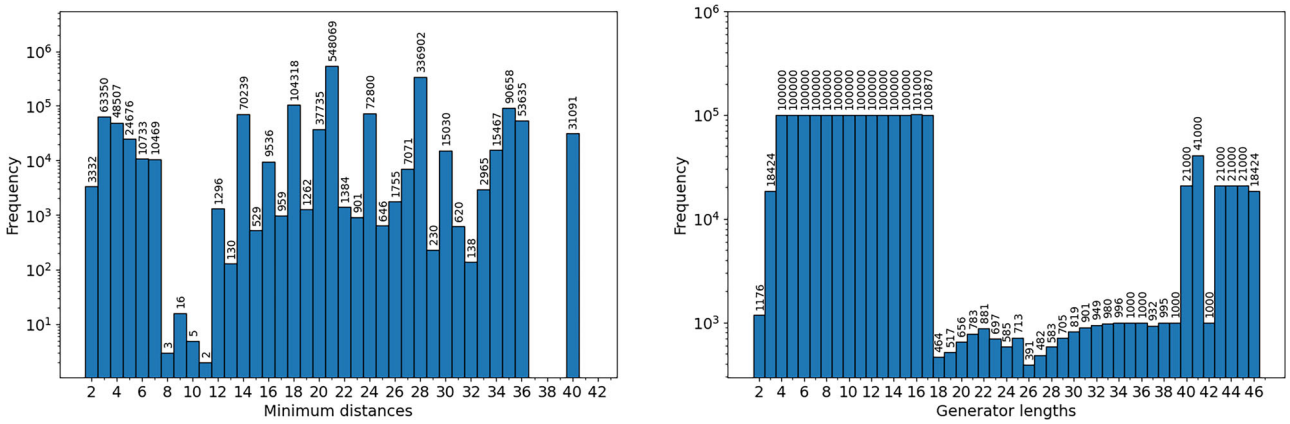

Fig. 4 | Histograms of the $\mathbb{F}_8$ codes dataset. The left histogram shows the distribution of minimum distances. The right histogram shows the distribution of generators' dimensions.

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.

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[10], 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 algebro-geometric 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 (guided by a single parameter test) as our model's predicted minimum distance, each generation 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 algebrogeometric codes. It could be possible that certain classes of quantum codes also have an evolvable parameter space.

Note also that we do not store all codes generated during the genetic algorithm; only those produced for BZ evaluation will be stored in our datasets.

We target champion codes of dimension k by searching only k-vertex sets and penalising candidates whose dimension drifts from k.

To see why the dimension k should equal the number of vertices k_V, observe that: on the one hand, increasing k_V does not improve our chances of producing a champion code for dimension k; on the other hand, we must have $k \leq k_V$. Thus, optimal codes should be found when $k = k_V$.

Additional penalties prevent rediscovering codes found in earlier runs. The number 300 in the formula is simply a way to offset the other two terms

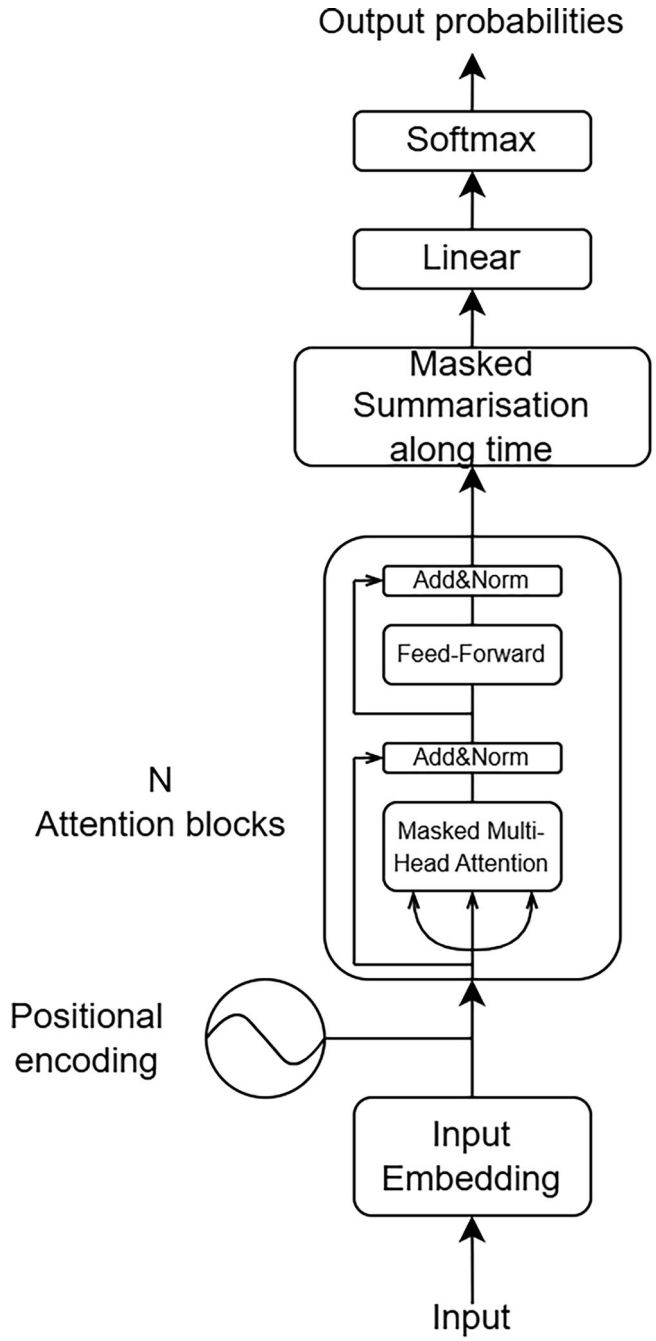

Fig. 5 | Transformer architecture for predicting minimum distance from a generator matrix. The single schematic shows the computation pipeline: input is a sequence of generator rows, input embedding maps field elements to vectors, positional encoding, a stack of attention blocks, masked summarisation along time, linear projections to logits, then softmax to output class probabilities.

and make the output of the formula positive as required by the PyGAD library we used[43]. 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[10]. To this aim, we use Algorithm 2.

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

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 $(3 \le i \le 34)$ are empty. For $3 \le i \le 34$, 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 $3 \le i \le 34$.
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 3), 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 3 and 4 show 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, 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. 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.

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

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 $(3 \le i \le 46)$ as empty sets.
2. For $3 \le i \le 46$, apply genetic algorithm to generate codes of dimension i.

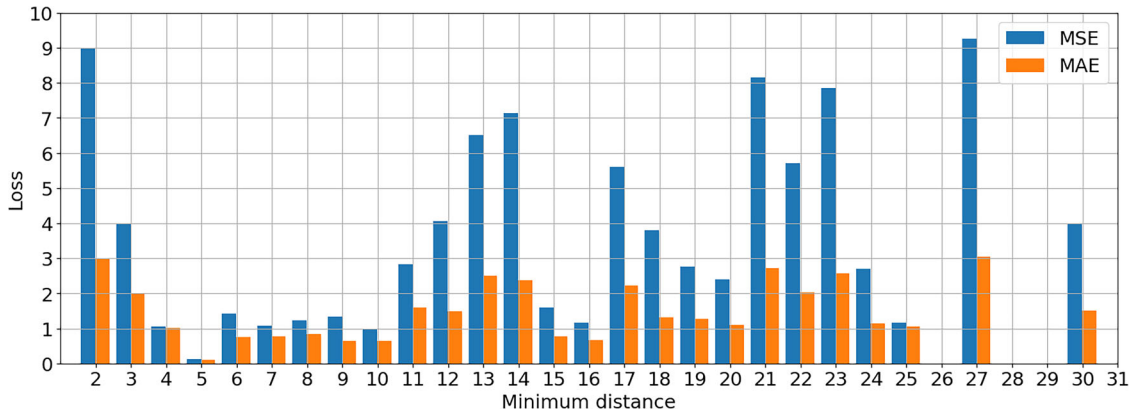

Fig. 6 | Per-class prediction errors on the F_7 test set. Losses on a test set for $\mathbb{F}_7$ codes.

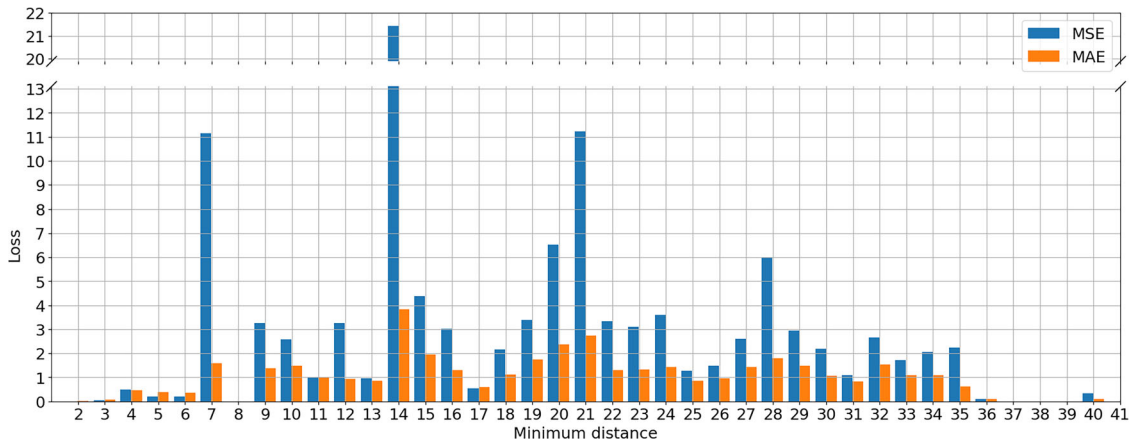

Fig. 7 | Per-class prediction errors on the F_8 test set. Losses on a test set for $\mathbb{F}_8$ codes.

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.

Due to time and computational constraints, we performed one complete run of the above algorithm, excluding dimension 30 due to the high computational cost of running the BZ algorithm at this dimension. Hence, there is no GA run for $l = 30$ in Table 2. Despite this, we found over 700 possibly new champion codes from this complete run.

It should be noted that we had previously run the algorithm for vertices between 20 and 29. In this search, we discovered and saved a champion code of dimension 22, as seen in Table 5. Due to a file error, we were unable to save the corresponding dataset that included this code. As a result, this code (and all codes from this run of the algorithm) will *not* be noted in Table 2.

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 dimensions that had no champion codes in their datasets: dimensions 18, 22, and 38. Through explicit calculation compared with the known best codes in, we see that all codes of dimension 18, 22, and 38 that have been discovered are in fact novel: see Table 6.

For any linear code, a standard upper bound on the minimum distance is given by the Singleton bound, $d \leq n - k + 1$. A comparison with our results (for $n = 49$) appears in Table 7. For sharper bounds based on additional code information, see[25,28,44,45].

Comparing our method with random search

Recall that our method aims to minimise the number of BZ evaluations required to identify new champion codes.

To illustrate our method's advantage over random search, we first detail the size of the search space. Over $\mathbb{F}_q$, there are $2^{(q-1)^2}$ possible vertex sets for constructing generalised toric codes. However, distinct vertex sets V_1, V_2 may yield equivalent linear codes. Following the notions of monomial and lattice equivalence from[46], we approximate the number of inequivalent codes as

$$\left|\frac{X}{\mathrm{AGL}_2(\mathbb{F}_q)}\right| \approx \frac{2^{(q-1)^2}}{q^2(q^2-1)(q^2-q)} \tag{3}$$

where $\mathrm{AGL}_2(\mathbb{F}_q)$ is the affine general linear group of $\mathbb{F}_q$ in two dimensions. Approximate values for the number of unique generalised toric codes over $\mathbb{F}_q$ $(q = 7, 8, 9, 11, 13)$ are shown in Table 8.

This estimate is less accurate for small q, where fixed configurations under affine transformations are non-negligible, but becomes effective for large q, where exact enumeration is infeasible.

Table 3 | Dimension of code vs. No. of runs to find best minimum Hamming distance

Dimension	No. of Runs
3	1
4	1
5	1
6	1
7	2
8	1
9	7
10	1
11	3
12	55
13	1
14	2
15	82
16	1
17	1
18	20
19	N/A
20	1
21	15
22	18
23	1
24	43
25	1
26	1
27	1
28	6
29	1
30	1
31	1
32	1
33	1
34	1

Table 4 | Dimension of code vs. Number of best codes in the dataset over $\mathbb{F}_7$

Dimension	No. of best codes
3	98
4	332
5	23,446
6	4023
7	115
8	564
9	27
10	2482
11	124
12	0
13	411
14	16
15	1
16	11,838
17	1951
18	66
19	0
20	1094
21	39
22	57
23	336
24	5
25	45,714
26	21,432
27	4215
28	80
29	2420
30	33,302
31	10,713
32	0
33	0
34	0

One may derive this formula as follows. By Burnside's Lemma,

$$\left|\frac{X}{\mathrm{AGL}_2(\mathbb{F}_q)}\right| = \frac{1}{|\mathrm{AGL}_2(\mathbb{F}_q)|}\sum_{g\in \mathrm{AGL}_2(\mathbb{F}_q)} |X^g|,$$

where $X^g = \{x \in X: g \cdot x = x\}$. Now, $|\mathrm{AGL}_2(\mathbb{F}_q)| = |\mathbb{F}_q^2| \cdot |\mathrm{GL}_2(\mathbb{F}_q)| = q^2(q^2-1)(q^2-q)$. Consider $|X^g|$ for each $g \in \mathrm{AGL}_2(\mathbb{F}_q)$: if g is the identity, then all configurations are fixed, so $|X^g| = 2^{(q-1)^2}$; if g is not the identity, then we have significantly fewer fixed configurations, which we may approximate as negligible as q increases.

To our knowledge, no prior work has quantified the proportion of generalised toric codes with a given Hamming distance among codes of fixed dimension or vertex count. Such a quantitative result is left for future work.

One way to compare the efficiency of the GA over a random search is to estimate the expected number of BZ evaluations required to find a champion code. Let $N_k^{(\ell)}$ denote the number of distinct codes of dimension k generated from ℓ-vertex sets, and $C_k^{(\ell)}$ the number of champions among them. Then, the expected number of evaluations in a random search is

$$\mathbb{E}[\text{ steps before champion}] = \frac{C_k^{(\ell)}}{N^{(\ell)}}, \tag{4}$$

where $N^{(\ell)} = \sum_k N_k^{(\ell)}$ is the total number of distinct codes generated from ℓ-vertex sets.

For the GA, this expectation can be similarly estimated using the number of codes evaluated and champions found, $\widetilde{C}_k^{(\ell)}$:

$$\mathbb{E}[\text{ steps before champion for GA}] = \frac{\widetilde{C}_k^{(\ell)}}{N^{(\ell)}}. \tag{5}$$

Tables 1 and 2 summarise these statistics for both random search and our GA search for the $\mathbb{F}_7$ and $\mathbb{F}_8$ datasets, respectively.

For smaller and larger vertex counts, where champions are common, using the GA provides little advantage. However, these cases are not computationally difficult.

In contrast, for intermediate vertex counts, the GA achieves up to a twofold reduction in BZ evaluations relative to random search.

Table 5 | 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)

Table 6 | All champion codes found over $\mathbb{F}_8$ with dimension $k = 18, 22, 38$ with block length $n = 49$ and best minimum distance found d

k	d	Vertices
18	21	(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)
18	21	(0, 1), (0, 5), (1, 3), (1, 4), (1, 6), (2, 1), (2, 5), (2, 6), (2, 7), (3, 0), (3, 1), (3, 4), (3, 6), (4, 5), (5, 1), (6, 0), (6, 1), (6, 5)
18	21	(0, 0), (0, 1), (0, 4), (1, 1), (1, 2), (2, 6), (3, 4), (3, 6), (4, 1), (4, 2), (4, 5), (4, 6), (5, 2), (5, 5), (6, 0), (6, 4), (6, 5), (7, 6)
18	21	(0, 1), (0, 2), (1, 3), (1, 4), (1, 6), (2, 4), (3, 2), (3, 3), (3, 4), (3, 5), (3, 6), (4, 1), (4, 2), (4, 6), (5, 1), (5, 3), (5, 5), (6, 4)
22	18	(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	(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)

Table 7 | Code parameters with $n = 49$ and corresponding Singleton bounds

Dimension	Min. Dist.	Singleton bound
3	42	47
4	40	46
6	36	44
7	35	43
8	34	42
18	21	32
22	18	28
38	7	12
40	6	10
41	6	9
44	4	6
46	3	4

Table 8 | Size of $|X/\mathrm{AGL}_2(\mathbb{F}_q)|$

q	Approximate size
7	$6.96 \cdot 10^5$
8	$2.49 \cdot 10^9$
9	$3.95 \cdot 10^{13}$
11	$7.94 \cdot 10^{23}$
13	$5.03 \cdot 10^{36}$

While smaller than the orders-of-magnitude improvement seen in[20], this can be attributed to two factors:

1. in certain dimensions (such as dimension 16 over $\mathbb{F}_7$ and dimension 40 over $\mathbb{F}_8$), the high density of champions causes the GA to saturate once it retains 30 elite codes per generation, and
2. in sparsely represented dimensions (as in dimensions 16 to 39 over $\mathbb{F}_8$), few or no champions appear in the training data, limiting the ML model's predictive accuracy so that champion codes receive low approximations from the model.

An ideal case over $\mathbb{F}_7$ is dimension 14, where the dataset contained few champion codes, making random discovery difficult; nonetheless, the GA located a champion within two runs.

For $\mathbb{F}_8$, the mid-range $\ell \in [18, 39]$ is especially sparse of codes. Given the exponentially larger search space compared to $\mathbb{F}_7$, one might expect the discovery of champions to require substantially more BZ evaluations, making the discovery of champions in dimensions 18, 22, and 38 particularly striking. Despite limited representation in the dataset (see Table 9), the ML model successfully generalised from other dimensions to poorly represented ones, allowing our method to evolve new champions.

Table 9 | Dimension of code vs. number of champion codes in training and testing datasets over $\mathbb{F}_8$

Dim	No. of champs
1	0
2	0
3	16,464
4	31,091
5	0
6	1034
7	216
8	28
9	3
10	11
11	1
12	144
13	1
14	0
15	146
16	1
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
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	16,464
47	0
48	1176

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. Improvements to the BZ algorithm[47] make the method easier to apply (datasets can be collected quicker) and accelerate the discovery of champion codes by reducing the computational cost of determining the minimum Hamming distance.

As described before, we wish to quantitatively characterise the distribution of generalised toric codes with respect to their Hamming distances across fixed dimensions or vertex counts: a problem that, to our knowledge, has not yet been explored.

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[48] checks, should surface additional champions. It may also be fruitful to consider other ways to investigate the parameter space[49].

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[6,28,45], or even construct possible champion codes from other families (BCH, Reed-Muller, AG, and more), once suitable GA "genes" are defined. Notably, Vaessens et al.[50] provide a general GA scheme for building large codes, but this is unsuitable for searching for codes of a fixed dimension as the fitness function would require a dimension penalty, as done in Algorithm Algorithm 1.

Finally, recasting the task as image classification with padded matrices or applying representation-learning techniques from NLP[51] offer alternative modelling avenues worth exploring.

Data availability

The datasets generated for the training of our models are available in the folder Datasets at the following URL: https://github.com/QTVLe/MachineLearningDiscoversNewChampionCodes. The datasets of the codes generated from the genetic algorithm are available in the folders F_7_Runs and *F_8_Runs* at the following URL: https://github.com/QTVLe/MachineLearningDiscoversNewChampionCodes.

Code availability

All code required to replicate the results in this paper is available in the folders F_7_GA, F_8_GA, Model_training, and Magma_code at the following URL under an MIT license: https://github.com/QTVLe/MachineLearningDiscoversNewChampionCodes.

Received: 6 August 2025; Accepted: 29 January 2026;
Published online: 25 March 2026

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 (2020).
2. Pamart, J. et al. Forward error correction in a 5 Gbit/s 6400 km EDFA based system. *Electron. Lett.* **30**, 342–343 (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 Trans. Commun. Technol.* **19**, 835–848 (1971).
5. Vardy, A. The intractability of computing the minimum distance of a code. *IEEE Trans. Inform. Theory* **43**, 1757–1766 (1997).
6. Hansen, J. P. Toric varieties Hirzebruch surfaces and error-correcting codes. *Appl. Algebra Eng. Comm. Comput.* **13**, 289–300 (2002).
7. Reed, I. S. & Solomon, G. Polynomial codes over certain finite fields. *J. Soc. Indust. Appl. Math.* **8**, 300–304 (1960).

8. Gallager, R. G. Low-density parity-check codes. *IRE Trans.* **IT-8**, 21–28 (1962).
9. Arıkan, E. Channel polarization: a method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *IEEE Trans. Inform. Theory* **55**, 3051–3073 (2009).
10. Brown, G. & Kasprzyk, A. M. Seven new champion linear codes. *LMS J. Comput. Math.* **16**, 109–117 (2013).
11. Brown, G. & Kasprzyk, A. M. Small polygons and toric codes. *J. Symb. Comput.* **51**, 55–62 (2013).
12. He, Y.-H. Machine-learning the string landscape. *Phys. Lett. B* **774**, 564–568 (2017).
13. Davies, A. et al. Advancing mathematics by guiding human intuition with AI. *Nature* **600**, 70–74 (2021).
14. Wu, Y. & De Loera, J. A. Turning mathematics problems into games: Reinforcement learning and Gröbner bases together solve integer feasibility problems (2022). 2208.12191.
15. Coates, T., Kasprzyk, A. M. & Veneziale, S. Machine learning the dimension of a Fano variety. *Nat. Commun.* **14**, 1–11 (2023).
16. Coates, T., Kasprzyk, A. M. & Veneziale, S. Machine learning detects terminal singularities. In Oh, A. et al. (eds.) Advances in Neural Information Processing Systems, vol. 36, 67183-67194 (2023).
17. He, Y.-H. AI-driven research in pure mathematics and theoretical physics. *Nat. Rev. Phys.* **6**, 546–553 (2024).
18. Gukov, S., Halverson, J. & Ruehle, F. Rigor with machine learning from field theory to the Poincaré conjecture. *Nat. Rev. Phys.* **6**, 310–319 (2024).
19. Yau, M., Karalias, N., Lu, E., Xu, J. & Jegelka, S. Are graph neural networks optimal approximation algorithms? In Globerson, A. et al. (eds.) *Advances in Neural Information Processing Systems*, vol. 37, 73124-73181 (2024).
20. Berglund, P. et al. New Calabi–Yau manifolds from genetic algorithms. Phys. Lett. B 850, Paper No. 138504, 10 (2024).
21. Abramson, J. et al. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature* **630**, 493—500 (2024).
22. Hashemi, B., Corominas, R. G. & Giacchetto, A. Can transformers do enumerative geometry? In *The Thirteenth International Conference on Learning Representations* https://openreview.net/forum?id=4X9RpKH4Ls (2025).
23. Ardon, L. Reinforcement learning to solve NP-hard problems: an application to the CVRP (2022). 2201.05393.
24. Hansen, J. P. Toric surfaces and error-correcting codes. 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 (2013).
26. Ruano, D. On the structure of generalized toric codes. *J. Symb. Comput.* **44**, 499–506 (2009).
27. Joyner, D. Toric codes over finite fields. *Appl. Algebra Eng. Comm. Comput.* **15**, 63–79 (2004).
28. Soprunov, I. & Soprunova, J. Toric surface codes and Minkowski length of polygons. *SIAM J. Discret. Math.* **23**, 384–400 (2008).
29. Zimmermann, K.-H. Integral hecke modules, integral generalized Reed–Muller codes, and linear codes. Tech. Rep., (1996).
30. Grassl, M. Searching for linear codes with large minimum distance. In *Discovering mathematics with Magma*, vol. 19 of *Algorithms Comput. Math.*, 287-313 (Springer, Berlin, 2006).
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. Softw.* **45**, Art. 23, 28 (2019).
32. Vaswani, A. et al. Attention is all you need. In Guyon, I. et al. (eds.) *Advances in Neural Information Processing Systems*, vol. 30 (2017).
33. Brown, T. et al. Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. & Lin, H. (eds.) *Advances in Neural Information Processing Systems*, vol. 33, 1877-1901 (2020).
34. Barbieri, F., Camacho-Collados, J., Espinosa Anke, L. & Neves, L. TweetEval: Unified benchmark and comparative evaluation for tweet classification. In Cohn, T., He, Y. & Liu, Y. (eds.) *Findings of the Association for Computational Linguistics: EMNLP 2020*, 1644-1650 (Association for Computational Linguistics, Online, 2020). https://aclanthology.org/2020.findings-emnlp.148/.
35. Karpathy, A. minGPT. https://github.com/karpathy/minGPT (2022).
36. Radford, A. et al. Language models are unsupervised multitask learners. Technical Report 1-24 (OpenAI, 2019).
37. Muller, D. E. Application of Boolean algebra to switching circuit design and to error detection. *Trans. I. R. E. Prof. Group Electron. Comput.* **EC-3**, 6–12 (1954).
38. Reed, I. S. A class of multiple-error-correcting codes and the decoding scheme. Trans. IRE38-49 (1954).
39. Hocquenghem, A. Codes correcteurs d'erreurs. *Chiffres* **2**, 147–156 (1959).
40. Bose, R. C. & Ray-Chaudhuri, D. K. On a class of error correcting binary group codes. *Inf. Control* **3**, 68–79 (1960).
41. Couvreur, A. & Randriambololona, H. Algebraic geometry codes and some applications. 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 (1983).
43. Gad, A. F. PyGAD: An intuitive genetic algorithm Python library. *Multimed. Tools Appl.* **83**, 58029–58042 (2024).
44. Beelen, P. & Ruano, D. The order bound for toric codes. In Bras-Amorós, M. & Høholdt, T. (eds.) *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, 1-10 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2009).
45. Little, J. & Schenck, H. Toric surface codes and Minkowski sums. *SIAM J. Discret. Math.* **20**, 999–1014 (2006).
46. Little, J. & Schwarz, R. On toric codes and multivariate vandermonde matrices. *Appl. Algebra Eng., Commun. Comput.* **18**, 349–367 (2007).
47. Bouyuklieva, S. & Bouyukliev, I. An extension of the brouwer-zimmermann algorithm for calculating the minimum weight of a linear code. *Mathematics* **9**. https://www.mdpi.com/2227-7390/9/19/2354 (2021).
48. Bosma, W., Cannon, J. & Playoust, C. The Magma algebra system. I. The user language. J. Symbolic Comput. 24, 235-265 (1997). Computational algebra and number theory (London, 1993).
49. Sipper, M., Fu, W., Ahuja, K. & Moore, J. H. Investigating the parameter space of evolutionary algorithms. *BioData Min.* **11**, 2 (2018).
50. Vaessens, R. J. M., Aarts, E. H. L. & van Lint, J. H. Genetic algorithms in coding theory—a table for $A_3(n, d)$. *Discret. Appl. Math.* **45**, 71–87 (1993).
51. Bengio, Y., Courville, A. & Vincent, P. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* **35**, 1798–1828 (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

Q.L. developed the MAGMA and SageMath algorithms for running experimentation and data collection, and was a major contributor in writing the manuscript. D.R. developed the transformer-based model and genetic algorithms, analysed the performance of the $\mathbb{F}_7$ and $\mathbb{F}_8$ models, and was a major contributor in writing the manuscript. Both Y.H. and A.K. were major contributors in writing the manuscript and developing the initial research problem that led to the conception of this paper. All authors reviewed the manuscript.

Competing interests
The authors declare no competing interests.

Additional information
Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s44387-026-00077-3.

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

Peer review information *Nature Communications* thanks Keyu Xia and the other, anonymous, reviewers for their contribution to the peer review of this work. A peer review file is available.