Applying machine learning to the Calabi-Yau orientifolds with string vacua

Xin Gao[1,*] and Hao Zou[2,3,†]

[1] *College of Physics, Sichuan University, Chengdu 610065, China*
[2] *Yau Mathematical Sciences Center, Tsinghua University, Beijing 100084, China*
[3] *Beijing Institute of Mathematical Sciences and Applications, Beijing 101408, China*

(Received 27 December 2021; accepted 9 February 2022; published 25 February 2022)

We use the machine learning technique to search the polytope which can result in an orientifold Calabi-Yau hypersurface and the "*naive type IIB string vacua*." We show that neural networks can be trained to give a high accuracy for classifying the orientifold property and vacua based on the newly generated orientifold Calabi-Yau database with $h^{1,1}(X) \leq 6$ [R. Altman, J. Carifio, X. Gao, and B. Nelson, Orientifold Calabi-Yau threefolds with divisor involutions and string landscape, arXiv:2111.03078]. This indicates the orientifold symmetry may already be encoded in the polytope structure. In the end, we try to use the trained neural networks model to go beyond the database and predict the orientifold signal of polytope for higher $h^{1,1}(X)$.

DOI: 10.1103/PhysRevD.105.046017

I. INTRODUCTION

Orientifold Calabi-Yau threefolds represent a rich phenomenological starting point for the construction of concrete string models for both particle physics and cosmology. There are lots of important properties like possible proper divisor exchange involutions, the classification and counts of orientifold planes under the involution, the nontrivial Hodge number splitting, which were systematically studied recently in [1].

The authors of [1] construct an orientifold Calabi-Yau threefold database in a systematic way for $h^{1,1}(X) \leq 6$ (www.rossealtman.com/tcy) by considering nontrivial $\mathbb{Z}_2$ divisor exchange involutions. The orientifold Calabi-Yau database is built on the threefolds database [2] constructed from the Kreuzer-Skarke list [3,4], and the earlier classificaion of divisor exchange involutions [5]. In [1], constructing orientifold Calabi-Yau involves several technical procedures which we summarized in Sec. II. This procedure include determining the topology for each individual divisor, identifying and classifying the proper nontrivial involutions for each unique Calabi-Yau hypersurface. Each of the proper involution will result in an new orientifold Calabi-Yau manifold with nontrivial odd equivariant cohomology $h^{1,1}_{-}(X/\sigma^*) \neq 0$. The authors clarified all possible fixed loci under the proper involution, i.e., the locations of $O3$, $O5$, and $O7$-planes in the type IIB projection. It was found that under the proper involutions one ends up with a majority of $O3/O7$-planes systems, most of which further admit aa "*naive type IIB string vacua*" by checking the D3 tadpole cancellation condition.

As we can see, constructing orientifold Calabi-Yau manifolds depends nontrivially on the underlying manifold data and it presents an interesting challenge for machine learning [6,7]. Machine learning has been a good implement in theoretical physics research and leads to fruitful results during the last couple of years. With the help of machine learning people are able to deal with problems with more computational efficiency, especially the problems involving big data, for example, study the landscape of string flux vacua [8–17] as well as F-theory compactifications [18–20]. This technique allows people to learn lots of quantities of Calabi-Yau manifolds, from its toric building blocks like the polytope structure [21,22] and triangulations [23,24], to the calculation of Hodge numbers [11,25–27], numerical metrics [28–31] and line bundle cohomologies [32,33]. Besides, machine learning has also been applied to study and find certain structures on Calabi-Yau for model building [34–41].

The newly established orientifold Calabi-Yau database [1] and its corresponding polytopes are the ideal data for machine learning in several aspects. First, the explicit formulas to determine an orientifold Calabi-Yau are not known and calculations rely on complicated and computationally intense algorithms [1]. It involves several computational algebraic geometry packages combined to work together [42–46]. So such topological properties are an interesting and challenging playground for machine

[*] xingao@scu.edu.cn
[†] hzou@bimsa.cn

learning. It is very interesting to see whether the machine learning can avoid these difficult calculations and is capable of learning this particular interesting property to get the desired polytope which can result in an orientifold Calabi-Yau and further the "*naive type IIB string vacua*".

Second, it was conjectured that the orientifold symmetry (or more precisely the involution symmetry) on the Calabi-Yau hypersurface is already encoded in the polytope structure. It is similar to the fact that when calculating the Hodge number of Calabi-Yau hypersurface X, one only need information of the reflexive polytope without desingularization [47], or triangulations in another word. So one may wonder whether one can determine a polytope can result in an orientifold Calabi-Yau hypersurface and naive string vacua or not, before a detail calculation. We expect by utilizing the power of the machine learning we may get closer to this question in this paper.

Third, it is very difficult to scan the Kreuzer-Skarke database to find the orientifold Calabi-Yau with higher $h^{1,1}$. This difficulty is threefold. One is due to the exponential increased size of the Kreuzer-Skarke database when $h^{1,1}$ goes higher [3,4]. For example, in [1] the authors considered 22974 favorable polytopes in total while only for $h^{1,1}(X) = 7$ itself, there are 50376 polytopes. Moreover, when $h^{1,1}$ increases it is more difficult to get all triangulations of the polytope due to the exponentially increased possible ways of doing that. Finally, the number of possible involutions also increase exponentially and for some of them it is extremely slow to get the fixed loci. Putting all these difficulties together, it is very unlikely to scan all the Kreuzer-Skarke database in a brute force way to get the orientifold Calabi-Yau with an accessible computer power. However, as shown in [1], the percentage of polytope which can result in an orientifold Calabi-Yau and the "*naive string vacua*" is very small (around 5% for $h^{1,1} \leq 6$). Moreover this percentage tends to decrease when $h^{1,1}$ goes higher. So the signal of orientifold is very rare in the full Kreuzer-Skarke database. This is exactly what we want to try to see whether the machine learning can help to pick out the "orientifold" signal in higher $h^{1,1}$. Considering there is no concrete generic orientifold Calabi-Yau with high $h^{1,1}$, our efforts to explore such possibility using machine learning would be very helpful.

Although there are many benefits for the machine learning to do the prediction, one should note machine learning cannot solve the problem once and for all. One has to check whether these prediction is correct or not. However, usually these check is hard to be done and the precision is not very high even the test accuracy is extremely high in training the neural network. This is due to the fact one has to use a subset of the database to learn something more complicated, just like in the Kreuzer-Skarke database, the larger the $h^{1,1}$ is, the more complicated of the polytopes is.

This paper is organized as follows. In Sec. II, we briefly summary the algorithm how to construct an orientifold Calabi-Yau manifold and the naive type IIB string vacua. In Sec. III we apply the machine learning method to study the orientifold Calabi-Yau database with $h^{1,1}(X) \leq 6$ [1]. Since it gives a very high accuracy, we will try to apply our network model to explore the higher $h^{1,1}(X)$ case. In Sec. IV we do the first step to predict the polytopes in $h^{1,1}(X) = 7$ which may give us the orientifold Calabi-Yau and vacua. We pick out some favorable cases to explicitly check whether our predictions give the right answer. We make a conclusion in Sec. V.

II. CONSTRUCT THE ORIENTIFOLD CALABI-YAU

Let us briefly recall some results from [1] in which the standard procedure to identify an orientifold Calabi-Yau threefolds is described in detail.

First we need a smooth description of our original Calabi-Yau hypersurface X from the Kreuzer-Skarke database [3,4]. In doing so, we must at least partially desingularize the ambient toric variety, denoted as $\mathcal{A}$, by blowing up enough of its singular points. A method for doing so is called maximal projective crepant partial (MPCP) desingularization, which involves the triangulation of the polar dual reflexive polytope Δ^*, containing at least one fine, star, regular triangulation (FSRT). We define the MPCP-desingularized ambient 4D toric variety as:

$$\mathcal{A} = \frac{\mathbb{C}^k \backslash Z}{(\mathbb{C}^*)^{k-4} \times G}, \tag{1}$$

where Z is the locus of points in $\mathbb{C}^k$ ruled out by the Stanley-Reisner ideal $\mathcal{I}_{\rm SR}(\mathcal{A})$, and G is the stringy fundamental group (trivial in most cases, there are only 14 polytopes in $h^{1,1} \leq 6$ contain nontrivial G). The geometry on this toric variety can be described by the projective coordinates $\{x_1, \ldots, x_k\}$ and their toric $\mathbb{C}^*$ equivalence classes

$$(x_1, \ldots, x_k) \sim (\lambda^{\mathbf{W}_{i1}} x_1, \ldots, \lambda^{\mathbf{W}_{ik}} x_k), \tag{2}$$

which define a projective weight matrix $\mathbf{W}$. However, there may exist two or more MPCP triangulations which result in the same Calabi-Yau hypersurface due to Wall's theorem [48]. This theorem shows the compact Calabi-Yau 3-folds are classified by their Hodge numbers, intersection numbers, and the second Chern Class. This leads to a "*geometry-wise description*" in which the various triangulations (phases of the complete Kähler cone) corresponding to a distinct Calabi-Yau threefold geometry were glued together. Furthermore, we restrict ourselves to the so-called "*favorable*" description, in which the toric divisor classes on the Calabi-Yau hypersurface X are all descended from ambient space $\mathcal{A}$.

Starting from a favorable geometry-wise description, we need to identify the proper involution σ which involves exchanging one or more pairs of divisors. Those divisors should have the same topology and at the same time have different weights (nontrivial identical divisors (NID)):

$$\begin{aligned}&\sigma\colon x_i \leftrightarrow x_j \Rightarrow \sigma^*\colon D_i \leftrightarrow D_j.\\ &H^\bullet(D_i) \cong H^\bullet(D_j), \qquad \mathcal{O}(D_i) \neq \mathcal{O}(D_j)\end{aligned} \tag{3}$$

Furthermore, such involution should satisfy the symmetry of Stanley-Reisner Ideal $\mathcal{I}_{\mathrm{SR}}(\mathcal{A})$ and the symmetry of the linear ideal $\mathcal{I}_{\mathrm{lin}}(\mathcal{A})$. The first symmetry is to ensure the involution be an automorphism of $\mathcal{A}$, leaving invariant the exceptional divisors from resolved singularities. The later one ensures the defining polynomial of CY remains homogeneous under involution. Putting these two together, the involution should be a symmetry of the Chow-group:

$$A^\bullet(\mathcal{A}) \cong \frac{\mathbb{Z}(D_1,\ldots,D_k)}{\mathcal{I}_{\mathrm{lin}}(\mathcal{A}) + \mathcal{I}_{\mathrm{SR}}(\mathcal{A})}, \tag{4}$$

indicating the triple intersection form defined in the Chow-group is invariant under the involution σ. In this paper, we only consider the "*geometry-wise proper involution*" which are globally consistent across all disjoint phases of the Kähler cone for each unique Calabi-Yau geometry.

The next task is to check whether there exist any point-wise fixed loci for a given involution on the Calabi-Yau threefold. The first step is to fix the invariant Calabi-Yau hypersurface polynomial $P_{\mathrm{symm}} = \sigma(P_{\mathrm{symm}})$ and the minimal generators $\mathcal{G}$ generated by homogeneous polynomials that are (anti-)invariant under σ:

$$\mathcal{G} = \mathcal{G}_0 \cup \mathcal{G}_+ \cup \mathcal{G}_-. \tag{5}$$

The unexchanged coordinates in $\mathcal{G}_0$ are known from our choice of involution. To find the nontrivial even and odd parity generators in $\mathcal{G}_+$ and $\mathcal{G}_-$, we must consider not only σ, but all possible nontrivial "subinvolutions" $\rho \subseteq \sigma$ given by the nonempty subsets of $\{\sigma_1,\ldots,\sigma_n\}$ of size $1 \leq m \leq n$. Then we denote the new coordinate in $\mathcal{G} \equiv \{y_1,\ldots,y_{k'}\}$ as:

$$y_\pm(\mathbf{a}) = x_{i_1}^{a_1} x_{i_2}^{a_2} \ldots x_{i_m}^{a_m} \pm x_{j_1}^{a_1} x_{j_2}^{a_2} \ldots x_{j_m}^{a_m},$$

The condition for homogeneity, in terms of the columns $\mathbf{w}_{i_s}$ and $\mathbf{w}_{j_s}$ of the weight matrix $\mathbf{W}$ is given by:

$$a_1(\mathbf{w}_{i_1} - \mathbf{w}_{j_1}) + a_2(\mathbf{w}_{i_2} - \mathbf{w}_{j_2}) + \cdots + a_m(\mathbf{w}_{i_m} - \mathbf{w}_{j_m}) = 0. \tag{6}$$

The second step is to perform a Segre embedding transforming the projective coordinates into the (anti-) invariant generators $\{x_1,\ldots,x_k\} \mapsto \{y_1,\ldots,y_{k'}\}$ which constructs a new weight matrix $\tilde{\mathbf{W}}$ for $\{y_i\}$. Then we can find out the naive fixed point loci in the new weight matrix. In order for a codimension-1 subvariety $D \subset X$ to be point-wise fixed under the involution, the corresponding coordinate exchange must force its defining polynomial to vanish, i.e., $\sigma\colon y_i \mapsto -y_i$, so that $D_i = \{y_i = 0\}$ is fixed. For point-wise fixed point with codimension larger than one, one needs to check whether the involution forces a subset of generators $\mathcal{F} \subseteq \mathcal{G}$ to vanish simultaneously. Namely, one needs to check $\mathcal{F} \cap \mathcal{G}_- \neq \emptyset$. It is important to note that the torus $\mathbb{C}^*$ actions provide $r = \mathrm{rank}(\tilde{\mathbf{W}})$ additional degrees of freedom for the generators to avoid being forced to zero. In each subset of generators $\mathcal{F}$, we check for this by solving the system of equations

$$\lambda_1^{\tilde{W}_{1i}} \lambda_2^{\tilde{W}_{2i}} \ldots \lambda_r^{\tilde{W}_{ri}} = \sigma(y_i)/y_i, \quad i = 1,\ldots,k'. \tag{7}$$

By the construction of the generator y_i, the right-hand side is equal to ± 1. The set is point-wise fixed if this equation is solvable in the λ_i.

After finding out these naive fixed point loci, we need to check whether each point-wise fixed loci lies in Stanley-Reisner ideal $\mathcal{I}_{\mathrm{SR}}$. The definition of $\mathcal{I}_{\mathrm{SR}}$ leads $\mathcal{A}$ to be split into different patches $\{U_i\}$. For a given fixed set, we compute in each sector U_i the dimension of the ideal generated by the symmetry part of Calabi-Yau polynomial P_{symm} and the fixed set generators $\mathcal{F} \equiv \{y_1,\ldots,y_p\}$

$$\mathcal{I}_{ip}^{\mathrm{fixed}} = \langle U_i, P_{\mathrm{symm}}, y_1,\ldots,y_p \rangle. \tag{8}$$

If $\dim \mathcal{I}_{ip}^{\mathrm{fixed}} < 0$ for all U_i, then $\mathcal{F}$ does not intersect X. For each subset that is not discarded, we repeat this calculation for the ideal with one fixed set generator $\dim \mathcal{I}_{i1}^{\mathrm{fixed}}$, and then two $\dim \mathcal{I}_{i2}^{\mathrm{fixed}}$, etc., until $\dim \mathcal{I}_{i\ell}^{\mathrm{fixed}} = \dim \mathcal{I}_{ip}^{\mathrm{fixed}}$ when adding more generators to the ideal no longer changes the dimension for any region U_i. Then, the intersection $\{y_1 = \cdots = y_\ell = 0\}$ of these generators gives the final point-wise fixed locus, with redundancies eliminated. In the end, an $O3$-plane corresponds to a codimension-3 point-wise fixed subvariety, an $O5$- plane has codimension-2, and an $O7$-plane has codimension-1. If no O-planes exist and the invariant Calabi-Yau hypersurface is smooth, then the involution defines a $\mathbb{Z}_2$ free action on X.

Finally, one can check whether the orientifold Calabi-Yau manifold support the string vacua, we consider a simple case where the $D7$-brane tadpole cancellation condition is satisfied by simply placing eight $D7$-branes on top of the $O7$-plane. Then we only need to check the $D3$-brane tadpole condition which simplified to:

$$N_{D3} + \frac{N_{\mathrm{flux}}}{2} + N_{\mathrm{gauge}} = \frac{N_{O3}}{4} + \frac{\chi(D_{O7})}{4} \equiv -Q_{D3}^{\mathrm{loc}}. \tag{9}$$

with $N_{\mathrm{flux}} = \frac{1}{(2\pi)^4\alpha'^2}\int H_3 \wedge F_3$, $N_{\mathrm{gauge}} = -\sum_a \frac{1}{8\pi^2}\int_{D_a} \mathrm{tr}\mathcal{F}_a^2$, and N_{D3}, N_{O3} the number of $D3$-branes, $O3$-planes respectively. The $D3$-tadpole cancellation condition requires the total D3-brane charge $Q_{D3}^{\rm loc}$ of the seven-brane stacks and $O3$-planes to be an integer. If the involution passes this naive tadpole cancellation check, we will denote our geometry as a "*naive orientifold type IIB string vacuum*." One can further check the smoothness of the orientifold Calabi-Yau and the Hodge number splitting under the involutions.

III. MACHINE LEARNING FOR THE ORIENTIFOLD CALABI-YAU

A. Dataset and processing

The database of orientifold Calabi-Yau threefolds ($h^{1,1} \leq 6$) we use to train our model was recently published in [1] and we only explore the "*geometry-wise proper involution*" which exchange the so-called proper nontrivial identical divisor (NID). Therefore, we will share the same assumptions as in [1], i.e., only consider favorable polytopes (22974 in total, with 14 admitting a nontrivial fundamental group). Among these 22960 polytopes, there are 1401 out of them contain *geometry-wise proper involution* we are interested in and 996 out of the 1401 "orientifolds" polytopes admit "*naive type IIB string vacua*" (see [1] for more details).

First of all, we use vertices of the favorable dual polytopes as the input data, which are matrices after putting together. There are two types of dual polytopes in the database: unresolved toric dual polytopes and resolved toric dual polytopes (which are more refined data). For the $h^{1,1} \leq 6$ case, we will use both data for comparison. Unfortunately, for $h^{1,1} > 6$ cases, there are no database for resolved polytopes yet, therefore we can only predict for these cases using the model learning from the unresolved dual polytopes.[1]The input data has different sizes as matrices, ranging from 4×5 to 4×10. To resolve this issue, we embed all the matrices into larger ones by adding zeros columns, and for the purpose of this paper (to make predictions for $h^{1,1} = 7$ as example) we set the maximal size as 4×11. Below is one example how we embed a polytopy of Polyid#2 (in the database) into a 4×11 matrix by adding another six columns of zeros on the right:

$$\begin{bmatrix} -1 & -1 & -1 & -1 & 4 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 1 & 0 & 0 & -1 \end{bmatrix} \xrightarrow{\text{embedding}} \begin{bmatrix} -1 & -1 & -1 & -1 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Second, the size of the dataset we can exploit is relatively small (22960) for machine learning and too small if we want to make predictions for $h^{1,1} \geq 7$ since the number of polytopes goes exponential as $h^{1,1}$ increases. One notable fact is that the ordering of the vertices in one data is only a matter of labeling and in principle we can always reorder these vertices. Therefore, we choose 120 permutations to enlarge our dataset, which end up with 2755200 polytopes for machine learning. In practice, the permutations are realized by permuting columns of the original matrices.

The output data we are targeting would be: 1. Whether a polytope can result in an orientifold Calabi-Yau manifold with *geometry-wise proper involution* and 2. Whether an "orientifold" polytope from the first step can end up with a *naive (type IIB) string vacua*. In either case, essentially this is a binary classification problem, valued in `True` or `False`.

B. CNN classifier and learning results

1. Model building

One of the neural networks suitable for our classification problem is the convolutional neural network (CNN) (see for example [6, Chapter 6] for an introduction). A typical CNN model consists of the input layer, convolutional layers, pooling layer(s), flatten layer, fully-connected layer(s) and the output layer. Due to our input data is of quite "low resolution" (4×11), we drop the pooling layer in our model. We add two fully connected layers and each has 100 neurons so that they can provide enough free parameters (weights and biases). But to prevent overfitting, we also add a dropout layer before the output layer. Since we are dealing with the classification problem, we choose the rectified linear activation function (ReLU) as the activation function for most layers except for the output layer.

We construct the above model using the well-established platform `TensorFlow` [49]. In more specific, our model is defined as below:

(i) Layers (excluded the input layer):
 - one $2D$ convolution layer, with 25 filters, kernel size 3×3 and `ReLU` activation function,

[1] We will see in the next section, for $h^{1,1} \leq 6$ our results from learning both unresolved and resolved cases are almost identical, therefore we believe the predicted results for $h^{1,1} > 6$ based purely on unresolved dual polytopes are reliable.

TABLE I. Test results for $h^{1,1} \le 6$.

	Unresolved	Resolved
Orientifold	99.906%	99.907%
Naive type IIB string vacua	99.802%	99.897%

- one flatten layer, with default setup,
- two full-connected layers (dense layers), both with 100 neurons and ReLU activation functions,
- one dropout layer, with a dropout rate of 0.1,
- one output layer (dense layer), with 2 neurons and `Softmax` activation function.

(ii) Loss function: `Categorical Crossentropy`.

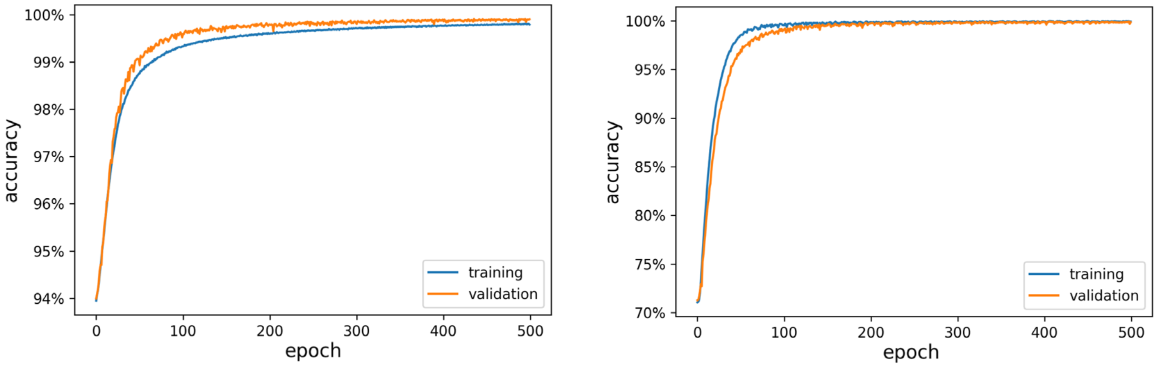
(iii) Optimizer: `Adam`, with default learning rate.

This model will be used to learn both orientifolds Calabi-Yau manifold and naive type IIB string vacua. Note that being orientifold is an necessary condition for a space to be a naive type IIB string vacuum, therefore we use the whole database ($h^{1,1} \le 6$) to train the model to identify an “orientifold” polytope while use only orientifold data ($h^{1,1} \le 6$ and `orientifold==True`) to learn whether it can end up with the string vacua. Since we have enlarged the database by permutations, the data size of purely orientifolds is also considerably large enough ($1401 \times 120 = 168120$) for machine learning. In practice, we can train the model for learning orientifold and string vacua separately as long as we use the restricted orientifolds database to train the vacua classifier.

2. Results

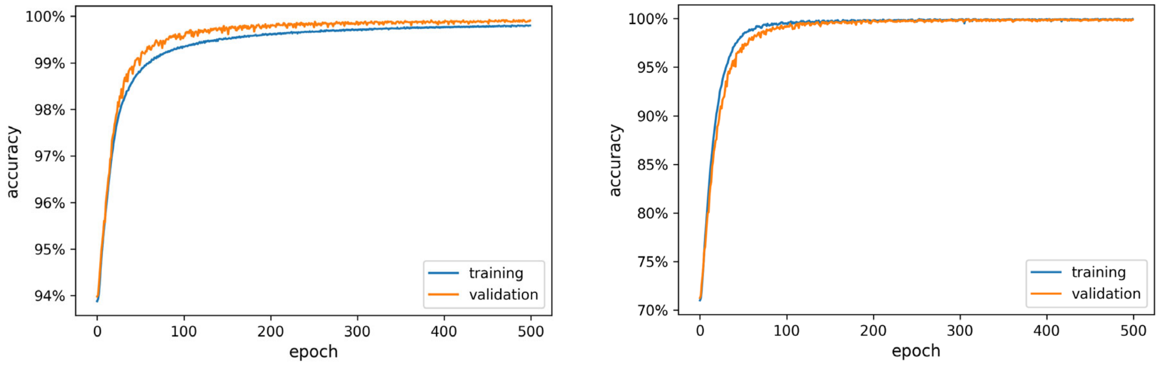
At this stage, we have feed the model with data of both resolved and unresolved toric dual polytopes. The test results are extremely accurate, $\gtrapprox 99.9\%$, as summarized in Table I, and they highly agree with the final training and validation accuracy (see Fig. 1 for training unresolved vertexes and Fig. 2 for training resolved vertexes). The learning curves in Fig. 1 also suggest that the high accuracy in our results is trustful and reliable, not obtained from overfitting. The learning curves for using resolved dual polytopes show the same features.

These test results also imply that, from the machine learning point of view, there would be almost no difference between using the resolved dataset and using the unresolved one when it classifies orientifolds or naive type IIB string vacua. It is in this sense that we can confidently use unresolved: confidently use an unresolved dataset to make



(a) (b)

FIG. 1. Learning curves for unresolved data. (a) Orientifold (b) Naive Type IIB string vacua.



(a) (b)

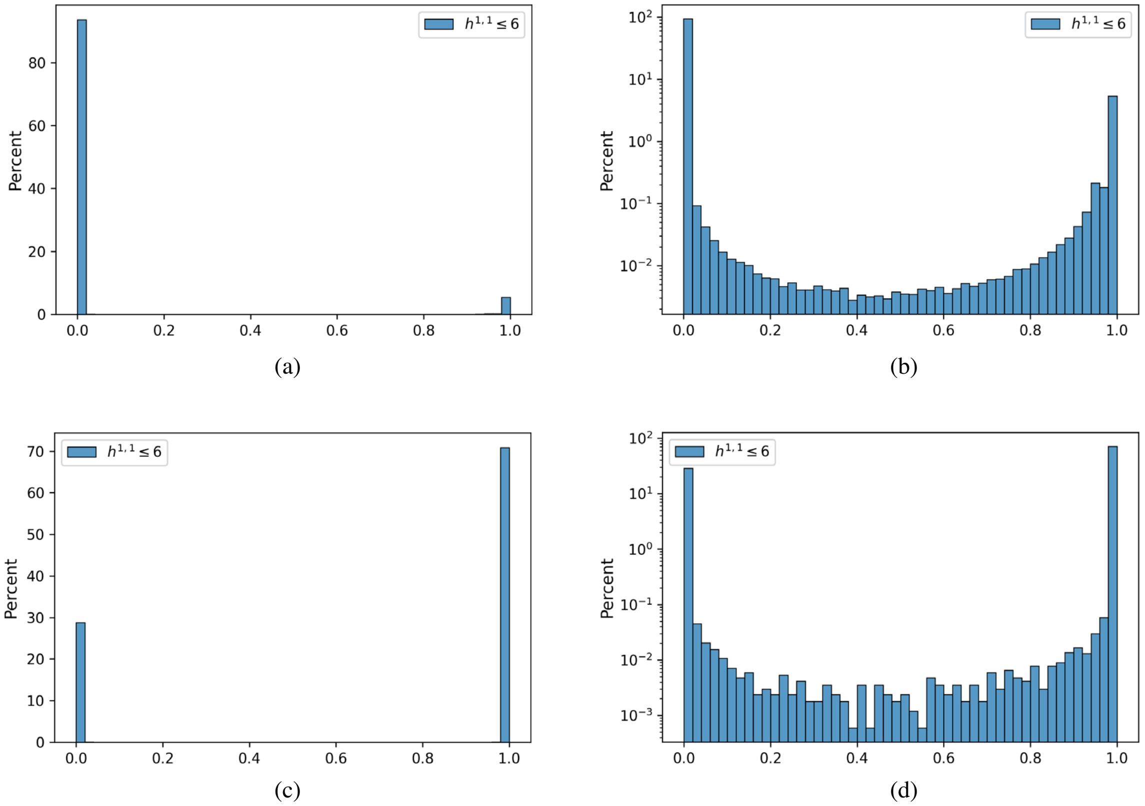
FIG. 2. Learning curves for resolved data. (a) Orientifold (b) Naive Type IIB string vacua

predictions for higher $h^{1,1} \geq 7$ in the next section. We leave a full machine learning of resolved dual-polytopes in a future work.

The high accuracy in the learning results only can tell us that it is an accurate model. For an ideal binary classifying model, the output θ only take values in $\{0, 1\}$ (corresponding to `False` and `True` respectively), namely there will be only two bins located in 0 and 1 separately in the distribution histograms. In practice, the output θ lies in the interval [0, 1] and the distribution histograms will have bins in between. We use the model to go through the training data again ($h^{1,1} \leq 6$) and see how our model recognize it. See Figs. 3(a) and 3(c) and their corresponding distributions in log-scale. (For later comparison with dataset of different sizes, we draw the probability distribution histograms.) We should emphasis that all the orientifold distributions are drawn using the whole database, while all the (naive type IIB string) vacua distributions are drawn only using orientifolds database.

In order to see whether it is a good binary classifier, in principle we should check its receiver operating characteristic (ROC) curve [50]. To put it simply, the farther the distributions of "signal" and "background" are separated, the better classifier it would be. In our scenario, the "signal" would indicate the orientifold or string vacuum. However, it is unrealistic to separate the signals from the background and what we really obtained is a combined signal-background distribution. In the probability distribution histograms, Fig. 3, there are two peaks, one is located at 0 and the other one is located at 1, while the bins in between are at least two orders of magnitude less (see the log-scale distributions). With the high accuracy from the learning results, we can tell that the peak located around 1 is exactly where our "signals" are highly concentrated at. Meanwhile the "background" is highly concentrated around 0. The histograms inform that the overlapping between the "signal" and "background" distributions is extremely small and therefore, we are confident to claim that our neural network is an accurate and good classifier to pick out the polytopes which can result in an orientifold Calabi-Yau and string vacua.

This high accuracy indicates the orientifold symmetry, or more precisely the involution symmetry such as the Chow-group symmetry, may already encoded in the polytope structure with unknown formula. This is reasonable since at least for involution, we require it to be the symmetries of the graded Chow ring of the ambient space. It is very happy to see the machine learning seems to pick out this property quite efficiently.



(a)

(b)

(c)

(d)

FIG. 3. Probability histograms for training data, obtained by evaluating the model with data of $h^{1,1} \leq 6$. (*a*) and (*b*) are orientifold distributions of the whole data set, while (*c*) and (*d*) are (naive type IIB string) vacua distributions in all orientifolds (same meaning in Fig. 4).

IV. TOWARD PREDICTION FOR ORIENTIFOLD CALABI-YAU DATABASE WITH HIGHER $h^{1,1}$

Another motivation of this paper is to search for potential polytopes which may result in an orientifold Calabi-Yau, and further the string vacua with higher $h^{1,1}(X)$. As we discussed in the Introduction of this paper, the difficulties including too much amount of polytopes, too many possible triangulations and involutions, complicated and computationally intense algorithm, make us hard to do a brute force calculation to scan the orientifold Calabi-Yau in the Kreuzer-Skarke database. Due to the fact the orientifold signal is very rare in the full polytopes (around 5% [1], see also Table II), it would be great even if we just train our machine to narrow down the candidate pool and increase the successful rate by one order. Thus motivated by our learning result described in Sec. III B, we will try to predict the possible polytopes with the desired property using our trained model. Here, our approach to achieve this goal is to utilize the classifier learnt from data with $h^{1,1} \leq 6$ to make predictions for $h^{1,1} \geq 7$. In this paper, we will only apply it to the $h^{1,1} = 7$ case as an example.

A. $h^{1,1} = 7$ case

We evaluate the model using the (unresolved) dual polytopes with $h^{1,1} = 7$, which can be obtained from the database [3,4]. The number of these polytopes is 50376 and it is much less than the size of data used to train our model ($50376/2755200 \sim 1.83\%$), and thus the parameters set by our training result is reliable to make predictions for $h^{1,1} = 7$.

The distributions of predictions are summarized in the probability histograms in Fig. 4, and the shape of the corresponding probability histograms suggests it is still a very good classifier to pick out the signal of candidates of "orientifold" polytopes and string vacua. Compared with the histograms for $h^{1,1} \leq 6$, one can see that although those figures for $h^{1,1} = 7$ are still with great shape, it is a little bit flattened. This is due to the following reason: we trained our model using favorable data but directly applied the model on all data of $h^{1,1} = 7$ without excluding the nonfavorable ones since there is no such database available before a complicated calculation.

We choose the classifying threshold $\theta = 0.5$, which means the machine gives a orientifold (and further string vacua) whenever its output value is greater than 0.5. With this choice, the computer tells us that among the polytopes with $h^{1,1} = 7$, there would be 2086 of them may result in an orientifold Calabi-Yau manifold, 1399 out of these 2086 polytopes may admit the naive type IIB string vacua. This is summarized in Table II. It is interested to notice that the percentage of polytopes which may contains the orientifold property indeed decrease following the trend when $h^{1,1}$ increase.

We have tested our predictions on a few examples and will present one specific example in the next section. These are limited examples among the data with $h^{1,1} = 7$ that can actually be computed directly following the methods in [1]. The reason is that we only considered favorable toric polytope in [1] while for $h^{1,1} \geq 7$ many of them are not. Even the polytope is favorable, due to the large number of vertexes in $h^{1,1} = 7$, it is very hard to triangulate it to a smooth manifold in computing time. Nevertheless, after the exact computations of some examples they do lie in our predictions of "orientifold" labeled by machine learning. We have attached a list of some favorable examples in the Appendix which is classified as "orientifold."

B. Predicted example

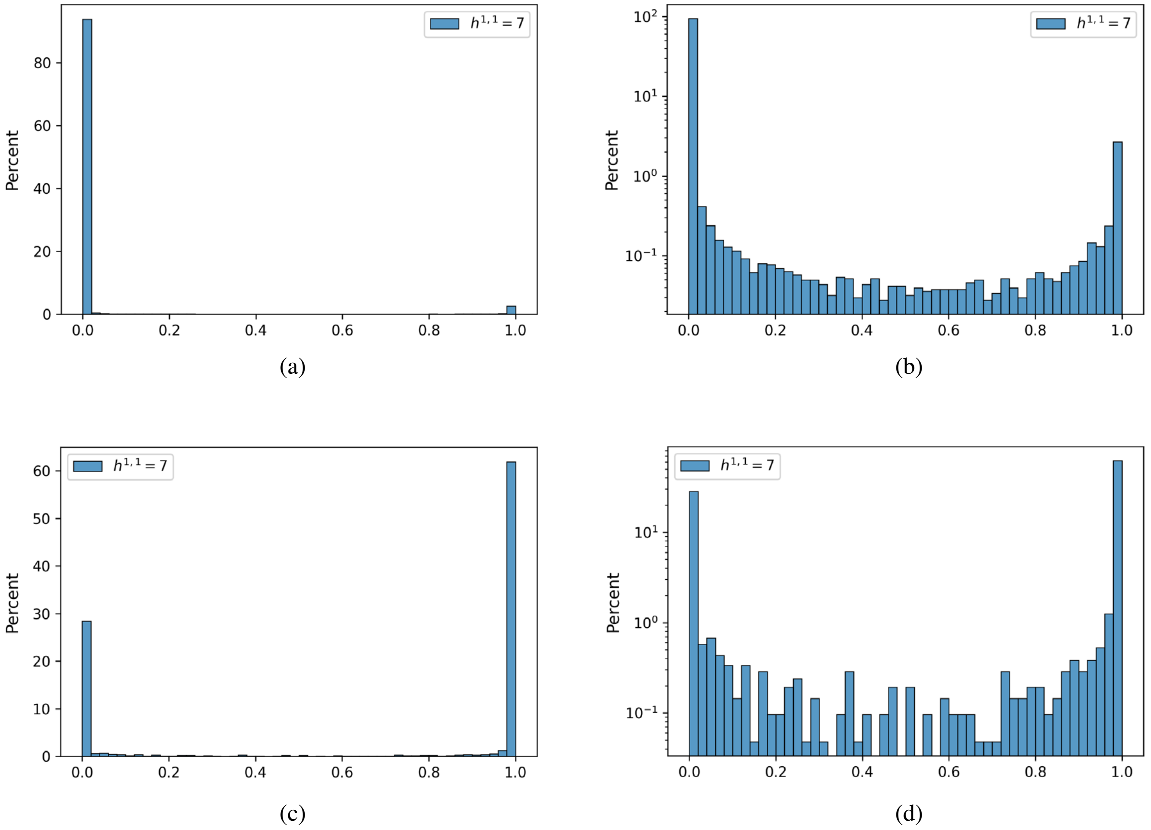
In this section, we present one particular example, which is labeled as both "orientifold" and "vacua" by machine learning. Let us check in detail whether it gives the right answer. The toric dual-polytope is given by the following 11 vertices with Hodge number $h^{1,1} = 7, h^{2,1} = 53$.

$$\begin{bmatrix} 0 & 1 & -1 & -1 & -1 & 0 & 0 & -1 & 0 & -1 & 1 \\ 0 & 1 & -1 & 0 & -1 & 0 & 0 & -1 & -1 & 0 & 0 \\ 0 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 1 \\ -1 & 1 & 0 & -1 & -1 & 0 & 1 & -1 & 0 & -1 & 1 \end{bmatrix}.$$

This example defines an MPCP desingularized ambient toric variety with weight matrix $\mathbf{W}$ given by

TABLE II. Statistic counting on the polytopes which can result in orientifold Calabi-Yau. The result for $h^{1,1} \leq 6$ comes from [1] while for $h^{1,1} = 7$ comes from our trained neural network.

$h^{1,1}(X)$	1	2	3	4	5	6	7
Number of trianed polytopes	5	36	243	1185	4897	16608	50376
Number of "orientifold" polytopes	0	1	16	96	330	958	2086
% of "orientifold" polytopes	0	2.78	6.58	8.10	6.74	5.77	4.14



(a)

(b)

(c)

(d)

FIG. 4. Predicted probability histograms for data with $h^{1,1}=7$. (a) Orientifold (b) Orientifold (log-scale) (c) Vacua (d) Vacua (log-scale).

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}
1	1	1	0	0	0	0	0	0	1	1
1	0	0	0	0	0	1	0	0	0	0
0	1	0	0	1	0	0	0	0	1	1
0	1	0	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	0	1	1	0
0	0	0	1	0	0	0	0	0	0	1
0	0	0	0	0	1	0	0	0	1	1

(10)

There are 12 different MPCP triangulations and we check the first one with Stanley-Reisner (SR) ideal as:

$$\mathcal{I}_{\mathrm{SR}}=\langle x_1x_3,x_1x_7,x_2x_3,x_2x_5,x_2x_8,x_2x_9,x_4x_8,x_4x_9,x_4x_{11},x_6x_8,x_6x_{10},x_9x_{10},x_3x_{11},x_5x_{11},x_5x_7\rangle.$$

The Hodge numbers of the corresponding individual toric divisors $D_i\equiv\{x_i=0\}$ can be calculate by the `cohomCalg` package [44,45]:

$$\begin{aligned}&h^{\bullet}(D_1)=h^{\bullet}(D_2)=\{1,0,1,20\},\qquad h^{\bullet}(D_{10})=h^{\bullet}(D_{11})=\{1,0,1,22\}\\&h^{\bullet}(D_4)=h^{\bullet}(D_8)=h^{\bullet}(D_9)=\{1,0,0,8\}\\&h^{\bullet}(D_3)=\{1,0,0,7\},\qquad h^{\bullet}(D_5)=\{1,0,0,5\},\qquad h^{\bullet}(D_7)=\{1,0,0,13\}\end{aligned}\tag{11}$$

So there are several possible proper involutions. We first consider the involution as follows:

$$\sigma\colon\ D_1 \leftrightarrow D_2, \qquad D_4 \leftrightarrow D_9, \qquad D_{10} \leftrightarrow D_{11}. \tag{12}$$

In order to be a consistent orientifold, the volume form Ω_3 should have a definite parity under σ. And, indeed we find $\sigma^*\Omega_3 = -\Omega_3$ and we would expect if there are any fix points under the involution, it should be $O3/O7$-planes. After a detailed calculation as described in Sec. II and [1], we determined that there indeed are four $O7$-planes under the involution without any $O3$-plane.

$$O7_{F_1}: x_4x_{10} - x_9x_{11}, \quad O7_{F_2}: x_3, \quad O7_{F_3}: x_5, \quad O7_{F_4}: x_6. \tag{13}$$

Then by placing eight $D7$-branes on top of the $O7$-plane, we only need to check the D3-tadpole cancellation condition:

$$\begin{aligned} N_{D3} + \frac{N_{\text{flux}}}{2} + N_{\text{gauge}} &= \frac{N_{O3}}{4} + \frac{\chi(D_{O7})}{4} \\ &= \frac{36+9+7+12}{4} \\ &= 16 \end{aligned} \tag{14}$$

So we indeed get an Orientifold Calabi-Yau three folds and naive string vacua.

However, not all the involutions will end up with orientifold and string vacua. For example, if we consider other involutions like $\{D_4 \leftrightarrow D_9, D_{10} \leftrightarrow D_{11}\}$, we can get the fixed locus as three O7-plane $\{\{x_4x_{10} - x_9x_{11}\}, \{x_6\}, \{x_5\}\}$ and one O3-plane located at $\{x_3, x_7, x_4x_{10} + x_9x_{10}\}$. But they can not satisfy the D3 tadpole cancellation condition.

One should note that this example is a favorable one. To check whether one vertex is favorable or not, one should desingularize the vertex up to determinant singularity. For those toric Calabi-Yau which is not favorable, one could possible favor it following a similar method introduced in [51] where all complete intersection Calabi-Yau 3-folds (CICYs) [52] can have a favorable description. Then for favorable Calabi-Yau, one should exhaust all triangulations to see whether there exist possible involutions and end up with fixed locus which may time consuming. Then one can check whether the D3 tadpole cancellation condition can be satisfied.

One has to notice that the list presented in Appendix are just predicted by the machine learning, and one has to check whether these predictions is correct or not following the method described in this subsection. Usually this check is hard to be done and the precision is not very high even though the test accuracy is extremely high (for $h^{1,1} \leq 6$ it is around 99.9%). This is usually the case when one use some simple training data to predict a more complicated one, like in the Kreuzer-Skarke database when increasing the $h^{1,1}$. We expect more than half of them will give the correct answer, i.e, we successfully narrow down the candidate pool and increase the successful rate by one order, from 5% to 60%. This value is sensitive to the amount of training data if it is not large. On the other hand, one may missing some orientifold polytope due to the classifying threshold θ one choose. We have checked if we train our neural network for the database of $h^{1,1} \leq 5$ and do a prediction for $h^{1,1} = 6$, it shows a similar picture. However, once we include more than 10% of the $h^{1,1} \leq 6$ data as training data, we would end up with a relatively very high accuracy of prediction (see Table III). For training the network with 10% of the whole $h^{1,1} \leq 6$ database, the validation accuracy does not increase but fluctuate around 91%, which indicates the initial setting of training ratio is too small. So one way to improve our prediction for $h^{1,1} = 7$ is to provide relatively small amount of data (>10%) for training. Another way to improve the prediction is to perform a in principle unsupervised machine learning, such as the generative adversarial network (GAN) [53] or variational autoencoder (VAE) [54]. All of these difficulties need a more detailed and systemic study of the neural network, we will leave it for a further study.

TABLE III. The test accuracy varies according to the ratio of training data in $h^{1,1} \leq 6$.

Ratio of training data	30%	20%	10%
Training accuracy	99.70%	99.64%	99.22%
Validation accuracy	99.75%	99.16%	91.90%
Test accuracy	99.76%	99.14%	91.64%

V. CONCLUSION

In this paper, we use the machine learning to clarify the polytope which can result in an orientifold Calabi-Yau hypersurface and together with the *naive type IIB string vacua* We show that indeed neural networks can be trained to give a high accuracy (around 99.9%) for classifying the orientifold property and vacua. This high accuracy indicates the orientifold symmetry, or more precisely the involution symmetry like the Chow-group structure, may already encoded in the polytope structure with unknown formula. In the end, we tried to use the trained neural networks model to go beyond the database and predict the orientifold property of polytope for higher $h^{1,1}(X)$. Again, as being conservative, we should emphasize that some checks on the predictions still need to be done even though the machine learning has an extremely high accuracy on training data. In fact, our work is just a starting point for machine learning on the new orientifold Calabi-Yau database. There are several ways we can improve it and do the machine learning in a more systematic way.

First, it would be very interesting to improve and generalize our work to have a more systemic machine learning for higher $h^{1,1}$. One may try other neural networks like generative adversarial network (GAN) or variational autoencoder (VAE) to improve the prediction. Beside these in principle unsupervised training, we can still have a supervised training. For example, to combine our method of finding orientifold signal and the method of triangulation [55] to generate enough training data for higher $h^{1,1}$ ($\gtrsim 10\%$ of the target data). On the other hand, it would also be interested to see whether learning the resolved dual-polytope vertex will give a more precise prediction in the higher $h^{1,1}$ case. For the machine learning for higher $h^{1,1}$ case, it would be great if the computer can pick out the polytopes which can result in a favorable Calabi-Yau and search for the orientifold structure there, since we only know the algorithm to do the brute force calculate the orientifold Calabi-Yau in the favorable case.

Second, in the context of CICY, a landscape of orientifold vacua has been constructed [56,57] from the most favorable description of the CICY 3-folds database [51]. More general free quotients have been classified and studied in the case of CICYs [58–61]. Applying the machine learning technique to these geometry would also be great. In fact, a methodological study of machine learning on such kind of CICYs has been done in [27]. In the coming paper[62], we will show that such kind of machine learning can also be done on the so-called "generalized complete intersection Calabi-Yau" (gCICYs) [63].

Third, a lot of work has been made in understanding the statistical structure of the moduli in many classes of Calabi-Yau threefolds, without considering the orientifold involution explicitly, such as the axion landscape or Swiss cheese structure [24,55,64–69]. Moreover, the study of the landscape of Calabi-Yau manifold with $h_-^{1,1} \neq 0$ under the exchange involution is also very interesting [57,70,71]. It would be great to combine our work to study the string model building in a real orientifold Calabi-Yau using machine learning technique. All of these works contain several technique problems and is important and worthwhile for a further study.

ACKNOWLEDGMENTS

The authors would like to thank Wei Cui, Jun Guo, Arthur Hebecker, Jinmian Li, Andreas Schachner, Juntao Wang for helpful discussions and correspondence. X. G. was supported in part by NSFC under Grant No. 12005250.

APPENDIX: SOME PREDICTED "ORIENTIFOLD" POLYTOPES ($h^{1,1} = 7$)

Vertices	Vacua
$[[-1,-1,0,2],[-1,-1,2,1],[-1,-1,2,0],[-1,-1,0,1],[-1,2,0,0],[1,-1,-1,0],[-1,-1,5,-1]]$	✓
$[[0,0,0,-1],[1,1,0,1],[-1,-1,-1,0],[-1,0,-1,-1],[-1,-1,-1,-1],[0,0,-1,0],[0,0,0,1],[-1,-1,0,-1],[0,-1,0,0],[-1,0,0,-1],[1,0,1,1]]$	✓
$[[-1,0,0,-1],[-1,0,-1,0],[1,1,0,0],[0,-1,0,0],[-1,-1,-1,-1],[-1,-1,0,-1],[-1,-1,-1,0],[0,0,0,1],[0,0,-1,0],[0,0,0,-1],[0,0,1,0]]$	✓
$[[-1,-1,0,-1],[-1,-1,0,0],[-1,-1,-1,0],[-1,0,0,0],[-1,0,0,-1],[0,0,1,0],[-1,-1,-1,-1],[0,0,0,-1],[-1,0,-1,-1],[0,-1,-1,1],[1,1,0,1]]$	✓
$[[-1,0,-1,1],[-1,0,-1,0],[-1,-1,0,-1],[-1,-1,-1,0],[1,0,0,0],[-1,-1,0,0],[-1,-1,-1,-1],[-1,0,0,-1],[0,-1,-1,-1],[0,0,0,-1],[1,1,1,0]]$	
$[[-1,1,-1,1],[0,1,-1,1],[1,-1,1,-1],[-1,-1,-1,1],[-1,-1,1,0],[-1,-1,-1,0],[0,0,-1,0],[1,0,1,-1]]$	✓
$[[0,1,1,1],[-1,-1,-1,-1],[-1,-1,-1,0],[-1,-1,0,-1],[0,-1,0,-1],[-1,0,-1,1],[-1,0,0,-1],[-1,1,1,1],[0,0,-1,1],[1,0,0,-1]]$	✓
$[[1,1,1,1],[0,0,-1,1],[-1,-1,-1,-1],[-1,0,0,-1],[-1,-1,0,-1],[-1,0,-1,1],[-1,1,1,1],[0,0,0,-1],[-1,-1,-1,0],[0,-1,0,-1]]$	✓
$[[-1,0,-1,0],[1,0,0,0],[0,-1,0,-1],[-1,-1,0,-1],[-1,-1,-1,-1],[0,0,-1,0],[1,0,0,-1],[0,1,0,1],[-1,-1,-1,0],[-1,0,0,0],[0,0,1,0]]$	✓
$[[0,0,-1,0],[-1,-1,-1,0],[-1,-1,-1,-1],[-1,-1,0,-1],[-1,0,-1,0],[-1,0,0,-1],[-1,0,0,0],[-1,-1,0,0],[0,-1,-1,0],[0,-1,0,-1],[1,1,1,1]]$	
$[[-1,-1,-1,0],[0,-1,0,0],[-1,0,-1,0],[0,1,1,0],[-1,-1,0,-1],[0,0,0,-1],[-1,0,0,-1],[-1,-1,-1,-1],[1,0,0,0],[-1,0,0,0],[0,0,-1,1]]$	✓
$[[0,0,-1,0],[-1,-1,-1,0],[-1,-1,-1,-1],[-1,-1,0,-1],[-1,0,-1,0],[-1,0,0,-1],[-1,0,0,0],[-1,-1,0,0],[0,-1,-1,0],[0,-1,0,-1],[1,1,1,1]]$	
$[[-1,-1,-1,0],[0,-1,0,0],[-1,0,-1,0],[0,1,1,0],[-1,-1,0,-1],[0,0,0,-1],[-1,0,0,-1],[-1,-1,-1,-1],[1,0,0,0],[-1,0,0,0],[0,0,-1,1]]$	✓

(Table continued)

(Continued)

Vertices	Vacua
[[0, −1, 1, −1], [−1, 0, 0, −1], [0, −1, 0, −1], [0, 0, −1, 0], [−1, −1, −1, −1], [0, −1, −1, 0], [0, 1, 0, 1], [−1, −1, −1, 0], [−1, −1, 0, −1], [−1, 0, −1, 0], [1, 1, 0, 1]]	✓
[[0, −1, 0, 0], [0, 0, −1, −1], [−1, 0, −1, −1], [−1, 0, −1, 0], [0, 0, 0, 1], [−1, −1, 0, 0], [−1, −1, −1, −1], [−1, −1, −1, 0], [1, 1, 0, 0], [−1, −1, 0, −1], [0, 0, 1, 0]]	✓
[[0, 0, −1, 0], [−1, 0, −1, −1], [−1, −1, 0, 0], [−1, −1, −1, 0], [1, 1, 0, 1], [0, −1, 0, 1], [0, 0, −1, −1], [−1, 0, 0, −1], [0, 0, 0, −1], [−1, −1, −1, −1], [0, −1, 1, 1]]	✓
[[−1, −1, 0, −1], [0, 0, −1, 0], [−1, −1, −1, 0], [−1, −1, −1, −1], [0, −1, 0, 0], [0, −1, −1, 0], [−1, 0, 0, 0], [−1, 0, −1, −1], [0, 0, −1, −1], [1, 0, 1, 1], [0, 1, 1, 1]]	✓
[[−1, 0, −1, −1], [0, −1, 0, 1], [−1, −1, 0, −1], [−1, 0, 0, −1], [−1, 0, 0, 0], [−1, −1, −1, −1], [0, 0, 0, −1], [−1, −1, 0, 0], [−1, −1, −1, 0], [0, −1, −1, 0], [1, 1, 1, 1]]	
[[−1, 0, −1, −1], [1, 1, 1, 0], [0, −1, 0, 0], [0, 0, 0, 1], [−1, 0, −1, 0], [−1, −1, −1, 0], [−1, −1, −1, −1], [−1, −1, 0, 0], [0, 0, 0, −1], [0, 0, −1, 0], [−1, −1, 0, −1]]	✓
[[−1, 0, −1, 2], [−1, −1, −1, 2], [0, 0, −1, 2], [−1, −1, −1, −1], [0, −1, 0, −1], [−1, −1, 0, −1], [−1, 0, 0, −1], [1, 1, 1, −1]]	✓
[[−1, −1, 0, 2], [−1, −1, 1, 0], [0, −1, 0, 0], [−1, 2, −1, −1], [−1, −1, 0, 0], [1, 2, −1, −1], [0, −1, 1, 0], [−1, 1, 0, −1], [0, 1, 0, −1]]	
[[−1, 0, 0, −1], [−1, −1, −1, 1], [−1, −1, 0, −1], [−1, 0, −1, 0], [−1, 0, 0, 0], [−1, −1, −1, −1], [0, −1, 0, −1], [0, 0, −1, 0], [1, 1, 1, 1]]	✓
[[0, 0, 0, −1], [−1, −1, −1, 0], [−1, −1, 0, −1], [−1, −1, −1, −1], [−1, 0, 0, −1], [−1, 0, 0, 0], [0, −1, 0, 0], [−1, −1, 0, 0], [0, −1, −1, 0], [−1, 0, −1, 0], [1, 1, 1, 1]]	
[[0, −1, 0, −1], [−1, −1, −1, 0], [−1, 0, −1, 0], [−1, 0, 0, 0], [−1, −1, −1, −1], [−1, 0, 0, −1], [0, −1, −1, 0], [−1, −1, 0, −1], [0, 0, −1, 0], [1, 0, 0, 1], [0, 1, 1, 0]]	
[[−1, 0, −1, 0], [0, 0, −1, 0], [−1, 0, 0, 0], [0, 0, 1, 0], [0, −1, 0, −1], [−1, −1, 0, −1], [0, −1, −1, 0], [−1, −1, −1, −1], [−1, −1, −1, 0], [−1, 0, 0, −1], [1, 1, 0, 1]]	
[[−1, 0, −1, −1], [−1, 0, 0, −1], [−1, 0, 0, 0], [−1, −1, −1, 0], [−1, −1, 0, −1], [−1, −1, −1, −1], [0, −1, 0, −1], [−1, −1, 0, 0], [0, −1, −1, 0], [−1, 0, −1, 0], [1, 1, 1, 1]]	
[[−1, −1, −1, 0], [−1, −1, −1, −1], [−1, 0, 0, −1], [−1, −1, 0, −1], [−1, −1, 0, 0], [0, −1, −1, 1], [−1, 0, −1, −1], [0, 0, 1, −1], [0, −1, −1, 0], [0, 1, 0, 0], [1, 0, 0, 1]]	
[[−1, −1, 0, −1], [0, 0, 0, 1], [0, −1, 0, 0], [−1, −1, −1, 0], [0, 0, −1, 0], [−1, 0, −1, −1], [−1, −1, −1, −1], [0, 0, −1, −1], [−1, −1, 0, 0], [0, 1, 0, 0], [1, 0, 1, 1]]	✓
[[0, 0, −1, 0], [−1, 0, 0, −1], [−1, 0, −1, 0], [−1, −1, −1, 0], [0, −1, 0, 0], [−1, −1, −1, −1], [−1, −1, 0, −1], [0, 0, 0, −1], [1, 1, 0, 1], [−1, 0, −1, 1], [0, −1, 1, −1]]	✓
[[−1, 1, 1, −1], [−1, −1, 0, 0], [0, −1, −1, 1], [−1, 0, −1, 0], [−1, −1, −1, 1], [−1, −1, −1, 0], [0, 0, −1, 0], [−1, 1, 1, 0], [0, −1, 0, 0], [1, 1, 1, 0]]	
[[0, 1, 0, 0], [−1, −1, −1, 0], [0, −1, 1, 0], [0, 0, −1, −1], [−1, −1, −1, −1], [−1, −1, 0, 0], [−1, 0, −1, −1], [−1, −1, 0, −1], [−1, 0, −1, 0], [−1, 0, 0, 0], [1, 1, 1, 1]]	✓
[[0, 0, −1, 0], [−1, −1, 0, −1], [0, −1, 0, 0], [−1, −1, −1, −1], [−1, 0, −1, −1], [−1, −1, −1, 0], [0, 0, −1, 1], [0, 1, −1, 0], [−1, 0, 0, −1], [1, 0, 1, 1], [0, −1, 1, −1]]	✓
[[−1, 0, −1, 1], [−1, 0, −1, 0], [−1, 0, 0, −1], [−1, −1, 0, −1], [0, −1, 0, 0], [−1, −1, −1, 0], [−1, −1, 0, 0], [−1, −1, −1, −1], [0, 0, 0, −1], [0, −1, −1, −1], [1, 1, 1, 0]]	
[[0, 0, −1, −1], [0, −1, 0, 0], [0, 0, −1, 0], [0, 0, 0, 1], [−1, −1, −1, 0], [−1, −1, −1, −1], [−1, −1, 0, −1], [−1, 0, −1, −1], [1, 1, 0, 1], [−1, 0, 0, −1], [0, −1, 1, 0]]	✓
[[0, 0, −1, 0], [−1, −1, −1, 0], [−1, −1, 0, −1], [−1, 0, −1, 0], [−1, 1, 1, 1], [1, 1, 1, 1], [−1, 0, 0, 1], [−1, −1, −1, −1], [−1, 0, 0, −1], [0, −1, 0, −1]]	✓
[[−1, 0, 0, −1], [−1, −1, 0, −1], [−1, −1, −1, −1], [−1, −1, −1, 0], [0, −1, −1, 0], [0, 0, 1, 0], [−1, −1, 0, 0], [−1, 0, 0, 0], [0, −1, −1, −1], [−1, 0, −1, 0], [1, 1, 1, 1]]	
[[−1, 0, −1, −1], [0, 0, −1, 0], [−1, −1, −1, 0], [−1, 0, −1, 0], [0, −1, 0, 0], [−1, −1, −1, −1], [−1, 0, 0, 0], [−1, −1, 0, 0], [−1, −1, 0, −1], [0, −1, −1, −1], [1, 1, 1, 1]]	

(Table continued)

(Continued)

Vertices	Vacua
$[[-1,-1,-1,0],[-1,-1,0,0],[-1,0,0,0],[0,-1,0,0],[-1,-1,0,-1],[-1,0,-1,0],[-1,-1,-1,-1],[0,-1,-1,-1],[0,0,-1,0],[-1,0,1,-1],[1,1,0,1]]$	
$[[-1,0,-1,0],[-1,0,-1,-1],[-1,-1,0,-1],[-1,-1,0,0],[0,-1,0,1],[-1,-1,-1,0],[-1,0,0,-1],[-1,-1,-1,-1],[0,0,0,-1],[0,-1,-1,0],[1,1,1,0]]$	
$[[-1,0,-1,1],[-1,0,-1,0],[-1,-1,0,0],[-1,-1,0,-1],[-1,-1,-1,0],[-1,0,0,-1],[0,0,0,-1],[-1,-1,-1,-1],[0,-1,0,-1],[0,-1,-1,0],[1,1,1,0]]$	
$[[-1,0,0,-1],[-1,-1,-1,0],[-1,-1,0,0],[0,-1,-1,0],[-1,-1,0,-1],[0,-1,0,0],[0,-1,-1,-1],[-1,0,0,0],[-1,-1,-1,-1],[-1,0,-1,0],[1,1,1,1]]$	
$[[-1,-1,0,0],[-1,-1,-1,0],[-1,0,0,-1],[-1,-1,-1,-1],[0,0,0,-1],[-1,-1,0,-1],[0,-1,-1,-1],[-1,0,0,0],[-1,0,-1,0],[0,0,-1,1],[1,1,1,0]]$	
$[[-1,0,-1,-1],[-1,0,0,-1],[-1,-1,0,-1],[-1,-1,0,0],[0,-1,-1,0],[-1,-1,-1,-1],[-1,-1,-1,0],[-1,0,0,0],[0,-1,-1,-1],[0,1,1,0],[1,0,0,1]]$	✓
$[[-1,0,-1,1],[-1,-1,-1,0],[-1,0,0,-1],[-1,-1,-1,-1],[-1,1,1,1],[0,-1,0,-1],[1,1,1,0],[-1,-1,0,-1],[-1,1,1,0],[0,0,-1,1]]$	✓
$[[-1,-1,1,1],[-1,-1,1,0],[1,1,0,-1],[-1,1,0,-1],[-1,1,-1,0],[-1,1,-1,-1],[0,-1,0,0],[0,-1,0,1],[-1,-1,0,0],[-1,-1,0,1]]$	✓
$[[-1,1,0,-1],[0,-1,1,0],[0,-1,0,1],[-1,1,-1,0],[-1,-1,0,1],[0,1,-1,-1],[-1,1,-1,-1],[-1,-1,0,0],[-1,-1,1,0],[-1,-1,1,1],[1,0,0,0]]$	✓
$[[-1,0,-1,0],[-1,0,-1,-1],[0,-1,-1,1],[-1,-1,0,0],[-1,-1,0,-1],[-1,0,0,-1],[0,1,1,-1],[0,0,0,-1],[-1,-1,-1,-1],[-1,-1,-1,0],[1,0,0,1]]$	✓
$[[0,-1,1,0],[0,-1,0,0],[0,-1,0,1],[-1,-1,0,0],[-1,-1,0,1],[-1,1,-1,0],[-1,-1,1,0],[-1,1,-1,-1],[-1,-1,1,1],[1,0,0,0],[-1,1,0,-1]]$	✓
$[[-1,-1,0,-1],[-1,-1,-1,0],[-1,0,-1,-1],[-1,-1,-1,-1],[-1,0,0,0],[1,1,-1,-1],[1,0,1,1],[0,-1,1,1],[-1,-1,1,1]]$	✓
$[[-1,-1,0,3],[-1,-1,0,2],[-1,-1,2,0],[-1,-1,1,0],[1,-1,0,0],[-1,2,-1,0],[1,-1,1,-1]]$	✓
$[[-1,-1,0,-1],[0,0,-1,0],[-1,-1,-1,0],[-1,0,-1,0],[-1,1,0,1],[-1,0,0,-1],[0,-1,0,-1],[1,0,0,0],[-1,-1,-1,-1],[0,-1,-1,0],[-1,0,1,0]]$	✓

[1] R. Altman, J. Carifio, X. Gao, and B. Nelson, Orientifold Calabi-Yau threefolds with divisor involutions and string landscape, arXiv:2111.03078.

[2] R. Altman, J. Gray, Y.-H. He, V. Jejjala, and B. D. Nelson, A Calabi-Yau database: Threefolds constructed from the Kreuzer-Skarke list, J. High Energy Phys. 02 (2015) 158.

[3] M. Kreuzer and H. Skarke, Complete classification of reflexive polyhedra in four-dimensions, Adv. Theor. Math. Phys. **4**, 1209 (2000).

[4] Kreuzer-Skarke database, http://hep.itp.tuwien.ac.at/kreuzer/CY.

[5] X. Gao and P. Shukla, On classifying the divisor involutions in Calabi-Yau threefolds, J. High Energy Phys. 11 (2013) 170.

[6] M. A. Nielsen, *Neural Networks and Deep Learning* (Determination Press, 2015).

[7] F. Ruehle, Data science applications to string theory, Phys. Rep. **839**, 1 (2020).

[8] A. Cole and G. Shiu, Topological data analysis for the string landscape, J. High Energy Phys. 03 (2019) 054.

[9] A. Cole, A. Schachner, and G. Shiu, Searching the landscape of flux vacua with genetic algorithms, J. High Energy Phys. 11 (2019) 045.

[10] S. Krippendorf, R. Kroepsch, and M. Syvaeri, Revealing systematics in phenomenologically viable flux vacua with reinforcement learning, arXiv:2107.04039.

[11] A. Cole, S. Krippendorf, A. Schachner, and G. Shiu, Probing the structure of string theory vacua with genetic algorithms and reinforcement learning, in *Proceedings of the 35th Conference on Neural Information Processing Systems* [arXiv:2111.11466].

[12] Y.-H. He, *The Calabi-Yau Landscape* (Springer, New York, 2021).

[13] F. Ruehle, Evolving neural networks with genetic algorithms to study the string landscape, J. High Energy Phys. 08 (2017) 038.

[14] J. Halverson, B. Nelson, and F. Ruehle, Branes with brains: Exploring string vacua with deep reinforcement learning, J. High Energy Phys. 06 (2019) 003.

[15] Y.-H. He, Universes as big data, Int. J. Mod. Phys. A **36**, 2130017 (2021).

[16] I. Bena, J. Blabäck, M. Graña, and S. Lüst, Algorithmically solving the Tadpole Problem, Adv. Appl. Clifford Algebras **32**, 7 (2022).

[17] Y.-H. He, S. Lal, and M. Z. Zaz, The world in a grain of sand: Condensing the string vacuum degeneracy, arXiv:2111.04761.

[18] J. Carifio, J. Halverson, D. Krioukov, and B. D. Nelson, Machine learning in the string landscape, J. High Energy Phys. 09 (2017) 157.

[19] Y.-N. Wang and Z. Zhang, Learning non-Higgsable gauge groups in 4D F-theory, J. High Energy Phys. 08 (2018) 009.

[20] M. Bies, M. Cvetič, R. Donagi, L. Lin, M. Liu, and F. Ruehle, Machine learning and algebraic approaches towards complete matter spectra in 4d F-theory, J. High Energy Phys. 01 (2021) 196.

[21] D. Krefl and R.-K. Seong, Machine learning of Calabi-Yau volumes, Phys. Rev. D **96**, 066014 (2017).

[22] J. Bao, Y.-H. He, E. Hirst, J. Hofscheier, A. Kasprzyk, and S. Majumder, Polytopes and machine learning, arXiv:2109.09602.

[23] R. Altman, J. Carifio, J. Halverson, and B. D. Nelson, Estimating Calabi-Yau hypersurface and triangulation counts with equation learners, J. High Energy Phys. 03 (2019) 186.

[24] M. Demirtas, L. McAllister, and A. Rios-Tascon, Bounding the Kreuzer-Skarke landscape, Fortschr. Phys. **68**, 2000086 (2020).

[25] K. Bull, Y.-H. He, V. Jejjala, and C. Mishra, Machine learning CICY threefolds, Phys. Lett. B **785**, 65 (2018).

[26] Y.-H. He and A. Lukas, Machine learning Calabi-Yau fourfolds, Phys. Lett. B **815**, 136139 (2021).

[27] H. Erbin and R. Finotello, Machine learning for complete intersection Calabi-Yau manifolds: A methodological study, Phys. Rev. D **103**, 126014 (2021).

[28] L. B. Anderson, M. Gerdes, J. Gray, S. Krippendorf, N. Raghuram, and F. Ruehle, Moduli-dependent Calabi-Yau and SU(3)-structure metrics from machine learning, J. High Energy Phys. 05 (2021) 013.

[29] V. Jejjala, D. K. M. Pena, and C. Mishra, Neural network approximations for Calabi-Yau metrics, arXiv:2012.15821.

[30] M. R. Douglas, S. Lakshminarasimhan, and Y. Qi, Numerical Calabi-Yau metrics from holomorphic networks, arXiv:2012.04797.

[31] M. Larfors, A. Lukas, F. Ruehle, and R. Schneider, Learning size and shape of Calabi-Yau spaces, arXiv:2111.01436.

[32] D. Klaewer and L. Schlechter, Machine learning line bundle cohomologies of hypersurfaces in toric varieties, Phys. Lett. B **789**, 438 (2019).

[33] C. R. Brodie, A. Constantin, R. Deen, and A. Lukas, Machine learning line bundle cohomology, Fortschr. Phys. **68**, 1900087 (2020).

[34] S. Krippendorf and M. Syvaeri, Detecting symmetries with neural networks, arXiv:2003.13679.

[35] R. Deen, Y.-H. He, S.-J. Lee, and A. Lukas, Machine learning string standard models, Phys. Rev. D **105**, 046001 (2022).

[36] H. Otsuka and K. Takemoto, Deep learning and k-means clustering in heterotic string vacua with line bundles, J. High Energy Phys. 05 (2020) 047.

[37] A. Ashmore, R. Deen, Y.-H. He, and B. A. Ovrut, Machine learning line bundle connections, arXiv:2110.12483.

[38] S. Abel, A. Constantin, T. R. Harvey, and A. Lukas, String model building, reinforcement learning and genetic algorithms, in *Nankai Symposium on Mathematical Dialogues: In Celebration of S. S. Chern's 110th Anniversary* [arXiv:2111.07333].

[39] S. Abel, A. Constantin, T. R. Harvey, and A. Lukas, Evolving heterotic gauge backgrounds: Genetic algorithms versus reinforcement learning, arXiv:2110.14029.

[40] A. Constantin, T. R. Harvey, and A. Lukas, Heterotic string model building with monad bundles and reinforcement learning, arXiv:2108.07316.

[41] T. R. Harvey and A. Lukas, Quark mass models and reinforcement learning, J. High Energy Phys. 08 (2021) 161.

[42] W. Stein *et al.*, Sage Mathematics Software (Version 9.1), The Sage Development Team, 2020, http://www.sagemath.org.

[43] W. Decker, G.-M. Greuel, G. Pfister, and H. Schönemann, Singular 3-1-6—A computer algebra system for polynomial computations, http://www.singular.uni-kl.de.

[44] R. Blumenhagen, B. Jurke, T. Rahn, and H. Roschy, Cohomology of line bundles: A computational algorithm, J. Math. Phys. (N.Y.) **51**, 103525 (2010).

[45] cohomCalg package, Download link: http://wwwth.mppmu.mpg.de/members/blumenha/cohomcalg/, 2010. High-performance line bundle cohomology computation based on [45].

[46] M. Kreuzer and H. Skarke, PALP: A package for analyzing lattice polytopes with applications to toric geometry, Comput. Phys. Commun. **157**, 87 (2004).

[47] V. V. Batyrev, Dual polyhedra and mirror symmetry for Calabi-Yau hypersurfaces in toric varieties, J. Algebraic Geom. **3**, 493 (1993).

[48] C. Wall, Classification problems in differential topology. V, Inventiones Mathematicae **1**, 355 (1966).

[49] M. Abadi *et al.*, TensorFlow: Large-scale machine learning on heterogeneous distributed systems, arXiv:1603.04467.

[50] Receiver operating characteristic, https://en.wikipedia.org/wiki/Receiver_operating_characteristic.

[51] L. B. Anderson, X. Gao, J. Gray, and S.-J. Lee, Fibrations in CICY threefolds, J. High Energy Phys. 10 (2017) 077.

[52] P. Candelas, A. Dale, C. Lutken, and R. Schimmrigk, Complete Intersection Calabi-Yau manifolds, Nucl. Phys. **B298**, 493 (1988).

[53] I. Goodfellow, J. Pouget-Abadie, M. Mehdi, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, Generative adversarial nets, in *Proceedings of the*

International Conference on Neural Information Processing Systems NIPS 2014 (2014), pp. 2672–2680.

[54] D. P. Kingma and M. Welling, Auto-encoding variational bayes, arXiv:1312.6114.

[55] C. Long, L. McAllister, and P. McGuirk, Heavy tails in Calabi-Yau moduli spaces, J. High Energy Phys. 10 (2014) 187.

[56] F. Carta, J. Moritz, and A. Westphal, A landscape of orientifold vacua, J. High Energy Phys. 05 (2020) 107.

[57] F. Carta, A. Mininno, N. Righi, and A. Westphal, Thraxions: Towards full string models, J. High Energy Phys. 01 (2022) 082.

[58] V. Braun, On free quotients of complete intersection Calabi-Yau manifolds, J. High Energy Phys. 04 (2011) 005.

[59] J. Gray, A. S. Haupt, and A. Lukas, All complete intersection Calabi-Yau four-folds, J. High Energy Phys. 07 (2013) 070.

[60] P. Candelas, A. Constantin, and C. Mishra, Hodge numbers for CICYs with symmetries of order divisible by 4, Fortschr. Phys. **64**, 463 (2016).

[61] A. Constantin, J. Gray, and A. Lukas, Hodge numbers for all CICY quotients, J. High Energy Phys. 01 (2017) 001.

[62] W. Cui, X. Gao, and J. Wang, Machine learning on generalized complete intersection Calabi-Yau (gCICY) (to be published).

[63] L. B. Anderson, F. Apruzzi, X. Gao, J. Gray, and S.-J. Lee, A new construction of Calabi–Yau manifolds: Generalized CICYs, Nucl. Phys. **B906**, 441 (2016).

[64] J. Gray, Y.-H. He, V. Jejjala, B. Jurke, B. D. Nelson, and J. Simon, Calabi-Yau manifolds with large volume vacua, Phys. Rev. D **86**, 101901 (2012).

[65] R. Galvez, Kahler moduli inflation in type IIB compactifications: A random tumble through the Calabi-Yau landscape, Phys. Rev. D **94**, 103521 (2016).

[66] C. Long, L. McAllister, and J. Stout, Systematics of axion inflation in Calabi-Yau hypersurfaces, J. High Energy Phys. 02 (2017) 014.

[67] R. Altman, Y.-H. He, V. Jejjala, and B. D. Nelson, New large volume Calabi-Yau threefolds, Phys. Rev. D **97**, 046003 (2018).

[68] M. Demirtas, C. Long, L. McAllister, and M. Stillman, The Kreuzer-Skarke axiverse, J. High Energy Phys. 04 (2020) 138.

[69] J. Halverson, C. Long, B. Nelson, and G. Salinas, Towards string theory expectations for photon couplings to axionlike particles, Phys. Rev. D **100**, 106010 (2019).

[70] X. Gao and P. Shukla, F-term stabilization of odd axions in LARGE volume scenario, Nucl. Phys. **B878**, 269 (2014).

[71] M. Cicoli, A. Schachner, and P. Shukla, Systematics of type IIB moduli stabilisation with odd axions, arXiv:2109.14624.