Physics Letters B 785 (2018) 65–72

Contents lists available at ScienceDirect

Physics Letters B

www.elsevier.com/locate/physletb

Machine learning CICY threefolds

Kieran Bull [a,*], Yang-Hui He [b,c,d], Vishnu Jejjala [e], Challenger Mishra [f]

[a] Department of Physics, University of Oxford, UK
[b] Department of Mathematics, City University, London, UK
[c] School of Physics, NanKai University, Tianjin, PR China
[d] Merton College, University of Oxford, UK
[e] Mandelstam Institute for Theoretical Physics, NITheP, CoE-MaSS, and School of Physics, University of the Witwatersrand, South Africa
[f] Rudolf Peierls Centre for Theoretical Physics and Christ Church, University of Oxford, UK

ARTICLE INFO

Article history:
Received 2 July 2018
Received in revised form 1 August 2018
Accepted 7 August 2018
Available online 16 August 2018
Editor: M. Cvetič

Keywords:
Machine learning
Neural network
Support Vector Machine
Calabi–Yau
String compactifications

ABSTRACT

The latest techniques from Neural Networks and Support Vector Machines (SVM) are used to investigate geometric properties of Complete Intersection Calabi–Yau (CICY) threefolds, a class of manifolds that facilitate string model building. An advanced neural network classifier and SVM are employed to (1) learn Hodge numbers and report a remarkable improvement over previous efforts, (2) query for favourability, and (3) predict discrete symmetries, a highly imbalanced problem to which both Synthetic Minority Oversampling Technique (SMOTE) and permutations of the CICY matrix are used to decrease the class imbalance and improve performance. In each case study, we employ a genetic algorithm to optimise the hyperparameters of the neural network. We demonstrate that our approach provides quick diagnostic tools capable of shortlisting quasi-realistic string models based on compactification over smooth CICYs and further supports the paradigm that classes of problems in algebraic geometry can be machine learned.

1. Introduction

String theory supplies a framework for quantum gravity. Finding our universe among the myriad of possible, consistent realisations of a four dimensional low-energy limit of string theory constitutes the vacuum selection problem. Most of the vacua that populate the string landscape are *false* in that they lead to physics vastly different from what we observe in Nature. We have so far been unable to construct even one solution that reproduces all of the known features of particle physics and cosmology in detail. The challenge of identifying suitable string vacua is a problem in big data that invites a machine learning approach.

The use of machine learning to study the landscape of vacua is a relatively recent development. Several avenues have already yielded promising results. These include Neural Networks [1–4], Linear Regression [5], Logistic Regression [5,4], Linear Discriminant Analysis, k-Nearest Neighbours, Classification and Regression Tree, Naive Bayes [5], Support Vector Machines [5,4], Evolving Neural Networks [6], Genetic Algorithms [7], Decision Trees and Random Forest [4], Network Theory [8].

Calabi–Yau threefolds occupy a central rôle in the study of the string landscape. In particular, Standard Model like theories can be engineered from compactification on these geometries. As such, Calabi–Yau manifolds have been the subject of extensive study over the past three decades. Vast datasets of their properties have been constructed, warranting a deep-learning approach [1,2], wherein a paradigm of machine learning computational algebraic geometry has been advocated. In this paper, we employ feedforward neural networks and support vector machines to probe a subclass of these manifolds to extract topological quantities. We summarise these techniques below.

- Inspired by their biological counterparts, artificial **Neural Networks** constitute a class of machine learning techniques capable of dealing with both classification and regression problems. In practice, they can be thought of as highly complex functions acting on an input vector to produce an output vector. There are several types of neural networks, but in this work we employ feedforward neural networks, wherein information moves

* Corresponding author.
E-mail addresses: kieran.bull@seh.ox.ac.uk (K. Bull), hey@maths.ox.ac.uk (Y.-H. He), vishnu@neo.phys.wits.ac.za (V. Jejjala), challenger.mishra@gmail.com (C. Mishra).

https://doi.org/10.1016/j.physletb.2018.08.008

in the forward direction from the input nodes to the output nodes via hidden layers.

- **Support Vector Machines (SVMs)**, in contrast to neural networks, take a more geometric approach to machine learning. SVMs work by constructing hyperplanes that partition the feature space and can be adapted to act as both classifiers and regressors.

The manifolds of interest to us are the Complete Intersection Calabi–Yau threefolds (CICYs), which we review in the following section. The CICYs generalise the famous quintic as well as Yau's construction of the Calabi–Yau threefold embedded in $\mathbb{P}^3\times\mathbb{P}^3$ [9]. The simplicity of their description makes this class of geometries particularly amenable to the tools of machine learning. The choice of CICYs is however mainly guided by other considerations. First, the CICYs constitute a sizeable collection of Calabi–Yau manifolds and are in fact the first such large dataset in algebraic geometry. Second, many properties of the CICYs have already been computed over the years, like their Hodge numbers [9,10] and discrete isometries [11–14]. The Hodge numbers of their quotients by freely acting discrete isometries have also been computed [11,15–18]. In addition, the CICYs provide a playground for string model building. The construction of stable holomorphic vector [19–23] and monad bundles [21] over smooth favourable CICYs has produced several quasi-realistic heterotic string derived Standard Models through intermediate GUTs. These constitute another large dataset based on these manifolds.

Furthermore, the Hodge numbers of CICYs were recently shown to be machine learnable to a reasonable degree of accuracy using a primitive neural network of the multi-layer perceptron type [1]. In this paper, we consider whether a more powerful machine learning tool (like a more complex neural network or an SVM) yields significantly better results. We wish to learn the extent to which such topological properties of CICYs are machine learnable, with the foresight that machine learning techniques can become a powerful tool in constructing ever more realistic string models, as well as helping understand Calabi–Yau manifolds in their own right. Expanding upon the primitive neural network in [1], and with the foresight that large datasets will likely benefit from deep learning, we turn to neural networks. We choose to contrast this technique with SVMs, which are particularly effective for smaller datasets with high dimensional data, such as the dataset of CICY threefolds.

Guided by these considerations, we conduct three case studies over the class of CICYs. We first apply SVMs and neural networks to machine learn the Hodge number $h^{1,1}$ of CICYs. We then attempt to learn whether a CICY is favourably embedded in a product of projective spaces, and whether a given CICY admits a quotient by a freely acting discrete symmetry.

The paper is structured as follows. In Section 2, we provide a brief overview of CICYs and the datasets over them relevant to this work. In Section 3, we discuss the metrics for our machine learning paradigms. Finally, in Section 4, we present our results.

2. The CICY dataset

A CICY threefold is a Calabi–Yau manifold embedded in a product of complex projective spaces, referred to as the ambient space. The embedding is given by the zero locus of a set of homogeneous polynomials over the combined set of homogeneous coordinates of the projective spaces. The deformation class of a CICY is then captured by a *configuration matrix* (1), which collects the multi-degrees of the polynomials:

$$X = \begin{matrix}\mathbb{P}^{n_1}\\ \vdots \\ \mathbb{P}^{n_m}\end{matrix}\left[\begin{matrix} q_1^1 & \cdots & q_K^1 \\ \vdots & \ddots & \vdots \\ q_1^m & \cdots & q_K^m \end{matrix}\right], \quad q_a^r \in \mathbb{Z}_{\geq 0}. \tag{1}$$

In order for the configuration matrix in (1) to describe a CICY threefold, we require that $\sum_r n_r - K = 3$. In addition, the vanishing of the first Chern class is accomplished by demanding that $\sum_a q_a^r = n_r + 1$, for each $r \in \{1,\ldots,m\}$. There are 7890 CICY configuration matrices in the CICY list (available online at [24]). At least 2590 of these are known to be distinct as classical manifolds.

The Hodge numbers $h^{p,q}$ of a Calabi–Yau manifold are the dimensions of its Dolbeault cohomology classes $H^{p,q}$. A related topological quantity is the Euler characteristic χ. We define these quantities below:

$$h^{p,q} = \dim H^{p,q}, \quad \chi = \sum_{p,q=0}^{3} (-1)^{p+q} h^{p,q}, \quad p,q \in \{0,1,2,3\} \tag{2}$$

For a smooth and connected Calabi–Yau threefold with holonomy group $SU(3)$, the only unspecified Hodge numbers are $h^{1,1}$ and $h^{2,1}$. These are topological invariants that capture the dimensions of the Kähler and the complex structure moduli spaces, respectively. The Hodge numbers of all CICYs are readily accessible [24]. There are 266 distinct Hodge pairs $(h^{1,1}, h^{2,1})$ of the CICYs, with $0 \leq h^{1,1} \leq 19$ and $0 \leq h^{2,1} \leq 101$. From a model building perspective, knowledge of the Hodge numbers is imperative to the construction of a string derived Standard Model.

If the entire second cohomology class of the CICY descends from that of the ambient space $\mathbb{A} = \mathbb{P}^{n_1}\times\ldots\times\mathbb{P}^{n_m}$, then we identify the CICY as *favourable*. There are 4874 favourable CICYs [24]. As an aside, we note that it was shown recently that all but 48 CICY configuration matrices can be brought to a favourable form through *ineffective splittings* [25]. The remaining can be seen to be favourably embedded in a product of del Pezzo surfaces. The favourable CICY list is also available online [26], although in this paper we will not be concerned with this new list of CICY configuration matrices. The favourable CICYs have been especially amenable to the construction of stable holomorphic vector and monad bundles, leading to several quasi-realistic heterotic string models.

Discrete symmetries are one of the key components of string model building. The breaking of the GUT group to the Standard Model gauge group proceeds via discrete Wilson lines, and as such requires a non-simply connected compactification space. Prior to the classification efforts [11,12], almost all known Calabi–Yau manifolds were simply connected. The classification resulted in identifying all CICYs that admit a quotient by a freely acting symmetry, totalling 195 in number, 2.5% of the total, creating a highly unbalanced dataset. 31 distinct symmetry groups were found, the largest being of order 32. Counting inequivalent projective representations of the various groups acting on the CICYs, a total of 1695 CICY quotients were obtained [24].

A CICY quotient might admit further discrete symmetries that survive the breaking of the string gauge group to the Standard Model gauge group. These in particular are phenomenologically interesting since they may address questions related to the stability of the proton via R-symmetries and the structure of the mixing matrices via non-Abelian discrete symmetries. A classification of the remnant symmetries of the 1695 CICY quotients found that 381 of them had nontrivial remnant symmetry groups [13], leading to a more balanced dataset based on symmetry. We will however focus on the first symmetry dataset available at [24] purely on the

grounds that the size of the dataset is itself much larger than the latter dataset.

3. Benchmarking models

In order to benchmark and compare the performance of each machine learning approach we adapt in this work, we use cross validation and a range of other statistical measures. Cross validation means we take our entire data set and split it into training and validation sets. The training set is used to train models whereas the validation set remains entirely unseen by the machine learning algorithm. Accuracy measures computed on the training set thus give an indication of the model's performance in recalling what it has learned. More importantly, accuracy measures computed against the validation set give an indication of the models performance as a predictor.

For regression problems we make use of both root mean square error (RMS) and the coefficient of determination (R^2) to assess performance:

$$\begin{aligned} \text{RMS} &:= \left(\frac{1}{N} \sum_{i=1}^{N} (y_i^{pred} - y_i)^2) \right)^{1/2} , \\ R^2 &:= 1 - \frac{\sum_i (y_i - y_i^{pred})^2}{\sum_i (y_i - \bar{y})^2} , \end{aligned} \tag{3}$$

where y_i and y_i^{pred} stand for actual and predicted values, with i taking values in 1 to N, and $\bar{y}$ stands for the average of all y_i. A rudimentary binary accuracy is also computed by rounding the predicted value and counting the results in agreement with the data. As this accuracy is a binary success or failure, we can use this measure to calculate a Wilson confidence interval. Define

$$\omega_{\pm} := \frac{p + \frac{z^2}{2n}}{1 + \frac{z^2}{n}} \pm \frac{z}{1 + \frac{z^2}{n}} \left(\frac{p(1-p)}{n} + \frac{z^2}{4n^2} \right)^{1/2} , \tag{4}$$

where p is the probability of a successful prediction, n the number of entries in the dataset, and z the *probit* of the normal distribution (*e.g.*, for a 99% confidence interval, $z = 2.575829$). The upper and the lower bounds of this interval are denoted by WUB and WLB respectively.

For classifiers, we have addressed two distinct types of problems in this paper, namely balanced and imbalanced problems. Balanced problems are where the number of elements in the true and false classes are comparable in size. Imbalanced problems, or the so called *needle in a haystack*, are the opposite case. It is important to make this distinction, since models trained on imbalanced problems can easily achieve a high accuracy, but accuracy would be a meaningless metric in this context. For example, consider the case where only $\sim 0.1\%$ of the data is classified as true. In minimising its cost function on training, a neural network could naively train a model which just predicts false for any input. Such a model would achieve a 99.9% accuracy, but it is useless in finding the special few cases that we are interested in. A different measure is needed in these cases. For classifiers, the possible outcomes are summarised by the *confusion matrix* of Table 1, whose elements we use to define several performance metrics:

$$\begin{aligned} \text{TPR} &:= \frac{tp}{tp + fn} , & \text{FPR} &:= \frac{fp}{fp + tn} , \\ \text{Accuracy} &:= \frac{tp + tn}{tp + tn + fp + fn} , & \text{Precision} &:= \frac{tp}{tp + fp} , \end{aligned} \tag{5}$$

Table 1
Confusion matrix.

		Actual	
		True	False
Predicted Classification	True	True Positive (tp)	False Positive (fp)
	False	False Negative (fn)	True Negative (tn)

where, TPR (FPR) stand for True (False) Positive Rate, the former also known as *recall*. For balanced problems, accuracy is the go-to performance metric, along with its associated Wilson confidence interval. However, for imbalanced problems, we use F-values and AUC. We define,

$$F := \frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}} , \tag{6}$$

while AUC is the area under the *receiver operating characteristic* (ROC) curve that plots TPR against FPR. F-values vary from 0 to 1, whereas AUC ranges from 0.5 to 1. We will discuss these in greater detail in Section 4.3.2.

4. Case studies

We conduct three case studies over the CICY threefolds. Given a CICY threefold X, we explicitly try to learn the topological quantity $h^{1,1}(X)$, the Hodge number that captures the dimension of the Kähler structure moduli space of X. We then attempt a (balanced) binary query, asking whether a given manifold is favourable. Finally, we attempt an (imbalanced) binary query about whether a CICY threefold X, admits a quotient X/G by a freely acting discrete isometry group G. In all the case studies that follow, neural networks were implemented using the Keras Python package with TensorFlow backend. SVMs were implemented by using the quadratic programming Python package Cvxopt to solve the SVM optimisation problem.

4.1. Machine learning Hodge numbers

As noted in Section 2, the only independent Hodge numbers of a Calabi–Yau threefold are $h^{1,1}$ and $h^{2,1}$. We attempt to machine learn these. For a given configuration matrix (1) describing a CICY, the Euler characteristic $\chi = 2(h^{1,1} - h^{2,1})$ can be computed from a simple combinatorial formula [27]. Thus, it is sufficient to learn only one of the Hodge numbers. We choose to learn $h^{1,1}$ since it takes values in a smaller range of integers than $h^{2,1}$.

4.1.1. Architectures

To determine the Hodge numbers we use regression machine learning techniques to predict a continuous output with the CICY configuration matrix (1) as the input. The optimal SVM hyperparameters were found by hand to be a Gaussian kernel with $\sigma = 2.74$, $C = 10$, and $\epsilon = 0.01$. Optimal neural network hyperparameters were found with a genetic algorithm, leading to an overall architecture of five hidden layers with 876, 461, 437, 929, and 404 neurons, respectively. The algorithm also found that a ReLU (rectified linear unit) activation layer and dropout layer of dropout 0.2072 between each neuron layer give optimal results.

A neural network classifier was also used. To achieve this, rather than using one output layer as is the case for a binary classifier or regressor, we use an output layer with 20 neurons (since $h^{1,1} \in (0, 19)$) with each neuron mapping to 0/1, the location of the 1 corresponding to a unique $h^{1,1}$ value. Note this is effectively adding extra information to the input as we are explicitly fixing the range of allowed $h^{1,1}$ s. For a large enough training data size this is not an issue, as we could extract this information from the

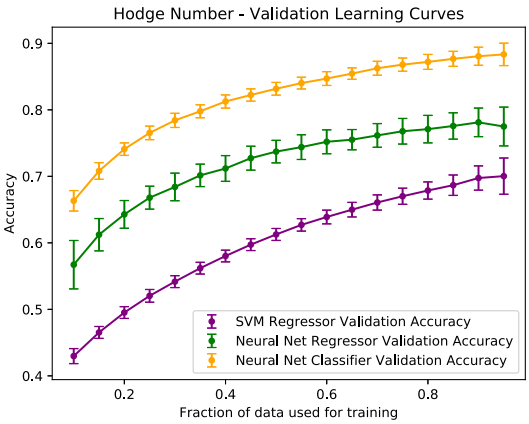

Fig. 1. Hodge learning curves generated by averaging over 100 different random cross validation splits using a cluster. The accuracy quoted for the 20 channel (since $h^{1,1} \in [0, 19]$) neural network classifier is for complete agreement across all 20 channels.

training data (choose the output to be the largest $h^{1,1}$ from the training data — for a large enough sample it is likely to contain $h^{1,1} = 19$). Moreover, for a small training data size, if only $h^{1,1}$ values less than a given number are present in the data, the model will not be able to learn these $h^{1,1}$ values anyway — this would happen with a continuous output regression model as well.

The genetic algorithm is used to find the optimal classifier architecture. Surprisingly, it finds that adding several convolution layers led to the best performance. This is unexpected as convolution layers look for features which are translationally or rotationally invariant (for example, in number recognition they may learn to detect rounded edges and associate this with a zero). Our CICY configurations matrices do not exhibit these symmetries, and this is the only result in the paper where convolution layers lead to better results rather than worse. The optimal architecture was found to be be four convolution layers with 57, 56, 55, and 43 feature maps, respectively, all with a kernel size of 3×3. These layers were followed by two hidden fully connected layers and the output layer, the hidden layers containing 169 and 491 neurons. ReLU activations and a dropout of 0.5 were included between every layer, with the last layer using a sigmoid activation. Training with a laptop computer's[1] CPU took less than 10 minutes and execution on the validation set after training takes seconds.

4.1.2. Outcomes

Our results are summarised in Figs. 1 and 2 and in Table 2. Clearly, the validation accuracy improves as the training set increases in size. The histograms in Fig. 2 show that the model slightly overpredicts at larger values of $h^{1,1}$.

We contrast our findings with the preliminary results of a previous case study by one of the authors [1,2] in which a Mathematica implemented neural network of the multi-layer perceptron type was used to machine learn $h^{1,1}$. In this work, a training data size of 0.63 (5000) was used, and a test accuracy of 77% was obtained. Note this accuracy is against the entire dataset after seeing only the training set, whereas we compute validation accuracies against only the unseen portion after training. In [1] there were a total of 1808 errors, so assuming the training set was perfectly learned (reasonable as training accuracy can be arbitrarily high with overfitting), this translates to a validation accuracy of 0.37. For the same sized cross validation split, we obtain a validation accuracy of 0.81 ± 0.01, a significant enhancement. Moreover, it should be emphasized that whereas [1,2] did a binary classification of large vs. small Hodge numbers, here the *actual* Hodge number $h^{1,1}$ is learned, which is a much more sophisticated task.

[1] Laptop specs: Lenovo Y50, i7-4700HQ, 2.4 GHz quad core; 16 GB RAM.

4.2. Machine learning favourable embeddings

Following from the discussion in Section 2, we now study the binary query: given a CICY threefold configuration matrix (1), can we deduce if the CICY is favourably embedded in the product of projective spaces? Already we could attempt to predict if a configuration is favourable with the results of Section 4.1 by predicting $h^{1,1}$ explicitly and comparing it to the number of components of $\mathbb{A}$. However, we rephrase the problem as a binary query, taking the CICY configuration matrix as the input and return 0 or 1 as the output.

An optimal SVM architecture was found by hand to use a Gaussian kernel with $\sigma = 3$ and $C = 0$. Neural network architecture was also found by hand, as a simple one hidden layer neural network with 985 neurons, ReLU activation, dropout of 0.46, and sigmoid activation at the output layer gave best results.

Results are summarised in Fig. 3 and Table 3. Remarkably, after seeing only 5% of the training data (400 entries), the models are capable of extrapolating to the full dataset with an accuracy $\sim 80\%$. This analysis took less than a minute on a laptop computer. Since computing the Hodge numbers directly was a time consuming and nontrivial problem [27], this is a prime example of how applying machine learning could shortlist different configurations for further study in the hypothetical situation of an incomplete dataset.

4.3. Machine learning discrete symmetries

The symmetry data resulting from the classifications [11,12] presents various properties that we can try to machine learn. An ideal machine learning model would be able to replicate the classification algorithm, giving us a list of every symmetry group which is a quotient for a given manifold. However, this is a highly imbalanced problem, as only a tiny fraction of the 7890 CICYs would admit a specific symmetry group. Thus, we first try a more basic question, given a CICY configuration, can we predict if the CICY admits any freely acting group. This is still most definitely a *needle in a haystack* problem as only 2.5% of the data belongs to the true class. In an effort to overcome this large class imbalance, we generate new synthetic data belonging to the positive class. We try two separate methods to achieve this — **sampling techniques** and **permutations of the CICY matrix**.

Sampling techniques preprocess the data to reduce the class imbalance. For example, downsampling drops entries randomly from the false class, increasing the fraction of true entries at the cost of lost information. Upsampling clones entries from the true class to achieve the same effect. This is effectively the same as associating a larger penalty (cost) to misclassifying entries in the minority class. Here, we use Synthetic Minority Oversampling Technique (SMOTE) [28] to boost performance.

4.3.1. SMOTE

SMOTE is similar to upsampling as it increases the entries in the minority class as opposed to downsampling. However, rather than purely cloning entries, new *synthetic* entries are created from the coordinates of entries in the feature space. Thus the technique is ignorant to the actual input data and generalises to any machine learning problem. We refer to different amounts of SMOTE by a integer multiple of 100. In this notation, SMOTE 100 refers to doubling the minority class (100% increase), SMOTE 200 refers to tripling the minority class and so on:

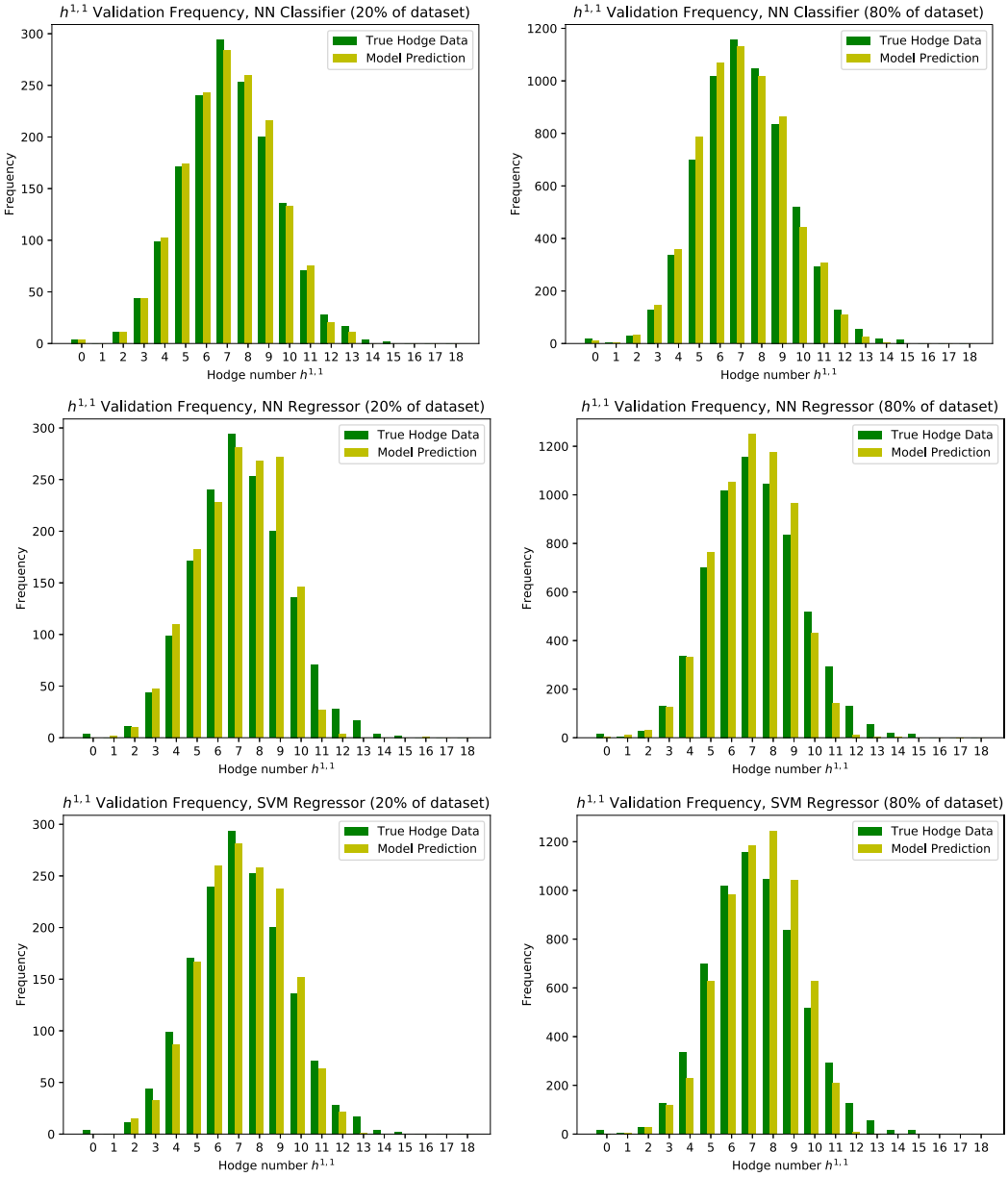

Fig. 2. The frequencies of $h^{1,1}$ (validation sets of size 20% and 80% respectively of the total data), for the neural network classifier (top row) and regressor (middle row) and the SVM regressor (bottom row).

Table 2
Summary of the highest validation accuracy achieved for predicting the Hodge numbers. WLB (WUB) stands for Wilson Upper (Lower) Bound. The dashes are because the NN classifier returns a binary 0/1 but RMS and R^2 are defined for continuous outputs. We also include 99% Wilson confidence interval evaluated with a validation size of 0.25 the total data (1972). Errors were obtained by averaging over 100 different random cross validation splits using a cluster.

	Accuracy	RMS	R^2	WLB	WUB
SVM Reg	0.70 ± 0.02	**0.53± 0.06**	**0.78 ± 0.08**	0.642	0.697
NN Reg	0.78 ± 0.02	0.46 ± 0.05	0.72 ± 0.06	0.742	0.791
NN Class	**0.88 ± 0.02**	–	–	**0.847**	**0.886**

SMOTE algorithm

1. For each entry in the minority class $\mathbf{x_i}$, calculate its k nearest neighbours $\mathbf{y_k}$ in the feature space (*i.e.*, reshape the 12×15, zero padded CICY configuration matrix into a vector $\mathbf{x_i}$, and find the nearest neighbours in the resulting 180 dimensional vector space).
2. Calculate the difference vectors $\mathbf{x_i} - \mathbf{y_k}$ and rescale these by a random number $n_k \in (0, 1)$.
3. Pick at random one point $\mathbf{x_i} + n_k(\mathbf{x_i} - \mathbf{y_k})$ and keep this as a new synthetic point.
4. Repeat the above steps $N/100$ times for each entry, where N is the amount of SMOTE desired.

4.3.2. SMOTE threshold, ROC, and F-values

The results obtained here all trivially obtained validation accuracies $\sim 99\%$. As noted in Section 3, this is meaningless and instead we should use AUC and F-values as our metrics. However, after processing the data with a sampling technique and training the model, we would only obtain one point (FPR, TPR) to plot on a ROC curve. Thus, to generate the full ROC curve, we vary the output threshold of the model to sweep through the entire range of values. Fig. 4 shows the profile of a good ROC curve.

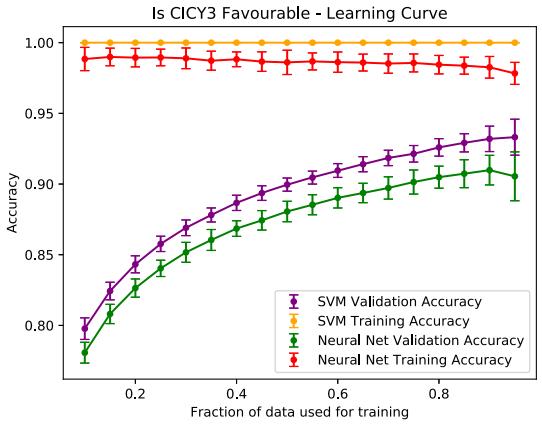

Fig. 3. Learning curves for testing favourability of a CICY.

Table 3
Summary of the best validation accuracy observed and 99% Wilson confidence boundaries. WLB (WUB) stands for Wilson Upper (Lower) Bound. Errors were obtained by averaging over 100 random cross validation splits using a cluster.

	Accuracy	WLB	WUB
SVM Class	**0.933 ± 0.013**	0.867	0.893
NN Class	0.905 ± 0.017	**0.886**	**0.911**

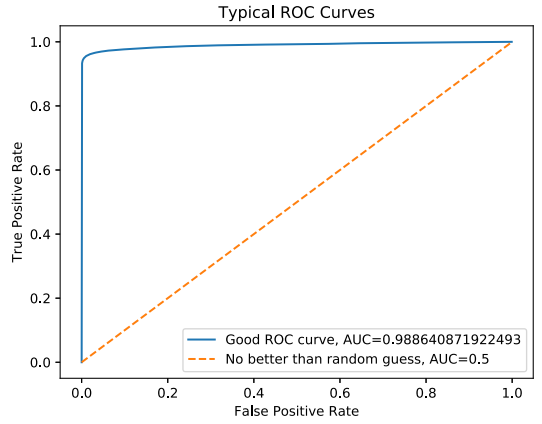

Fig. 4. Typical ROC curves. The points above the diagonal represent classification results which are better than random.

4.3.3. Permutations

From the definition of the CICY *configuration matrix* (1), we note that row and column permutations of this matrix will represent the same CICY. Thus we can reduce the class imbalance by simply including these permutations in the training data set. In this paper we use the same scheme for different amounts of PERM as we do for SMOTE, that is, PERM 100 doubles the entries in the minority class, thus one new permuted matrix is generated for each entry belonging to the positive class. PERM 200 creates two new permuted matrices for each entry in the positive class. Whether a row or column permutation is used is decided randomly.

4.3.4. Outcomes

Optimal SVM hyperparameters were found by hand to be a Gaussian kernel with $\sigma = 7.5, C = 0$. A genetic algorithm found the optimal neural network architecture to be three hidden layers with 287, 503, and 886 neurons, with ReLU activations and a dropout of 0.4914 in between each layer.

SMOTE results are summarised in Table 4 and Fig. 5. As we sweep the output threshold, we sweep through the extremes of

Table 4
Metrics for predicting freely acting symmetries. Errors were obtained by averaging over 100 random cross validation splits using a cluster.

SMOTE	SVM AUC	SVM max F	NN AUC	NN max F
0	0.77 ± 0.03	0.26 ± 0.03	0.60 ± 0.05	0.10 ± 0.03
100	0.75 ± 0.03	0.24 ± 0.02	0.59 ± 0.04	0.10 ± 0.05
200	0.74 ± 0.03	0.24 ± 0.03	0.71 ± 0.05	0.22 ± 0.03
300	0.73 ± 0.04	0.23 ± 0.03	0.80 ± 0.03	0.25 ± 0.03
400	0.73 ± 0.03	0.23 ± 0.03	0.80 ± 0.03	0.26 ± 0.03
500	0.72 ± 0.04	0.23 ± 0.03	0.81 ± 0.03	0.26 ± 0.03

classifying everything as true or false, giving the ROC curve its characteristic shape. This also explains the shapes of the F-value graphs. For everything classified false, $tp, fp \to 0$, implying the F-value blows up, hence the diverging errors on the right side of the F-curves. For everything classified true, $fp \gg tp$ (as we only go up to SMOTE 500 with 195 true entries and use 80% of the training data, this approximation holds). Using a Taylor expansion $F \approx 2tp/fp = 2 \times 195/7890 = 0.049$. This is observed on the left side of the F-curves. In the intermediate stage of sweeping, there will be an optimal ratio of true and false positives, leading to a maximum of the F-value. We found that SMOTE did not affect the performance of the SVM. Both F-value and ROC curves for various SMOTE values are all identical within one standard deviation. As the cost variable for the SVM $C = 0$ (ensuring training leads to a global minimum), this suggests that the synthetic entries are having no effect on the generated hypersurface. The distribution of points in feature space is likely too strongly limiting the possible regions synthetic points can be generated. However, SMOTE did lead to a slight performance boost with the neural network. We see that SMOTEs larger than 300 lead to diminishing returns, and again the results were quite poor, with the largest F-value obtained being only 0.26. To put this into perspective, the optimal confusion matrix values in one run for this particular model (NN, SMOTE 500) were $tp = 30, tn = 1127, fp = 417, fn = 10$. Indeed, this model could be used to shortlist 447 out of the 1584 for further study, but 417 of them are falsely predicted to have a symmetry and worse still this model misses a quarter of the actual CICYs with a symmetry.

PERM results are summarized in Table 5 and Fig. 6. Note these results are not averaged over several runs and are thus noisy. We see that for 80% of the training data used (the same training size as used for SMOTE runs) that the F-values are of the order 0.3–0.4. This is a slight improvement over SMOTE, but we note from the PERM 100, 000 results in Table 5 there is a limit to the improvement permutations can give.

Identifying the existence and the form of freely acting discrete symmetries on a Calabi–Yau geometry is a difficult mathematical problem. It is therefore unsurprising that the machine learning algorithms also struggle when confronted with the challenge of finding a rare feature in the dataset.

5. Discussion

In this study, continuing with the paradigm in and improving upon the results of [1,2], we utilise neural networks and Support Vector Machines (SVMs) to machine learn various geometric properties of CICY threefolds. We note that the SVM performed fractionally better than the neural network at predicting favourability of a CICY threefold. This could perhaps be attributed to the fact that SVMs are natural binary classifiers. Moreover, the simple architecture of the neural network needed for classifying favourability implies that the classifying function (or correspondingly, the hypersurface for the SVM) must be relatively simple. Since the training algorithm for SVMs assure a global minimum, we expect

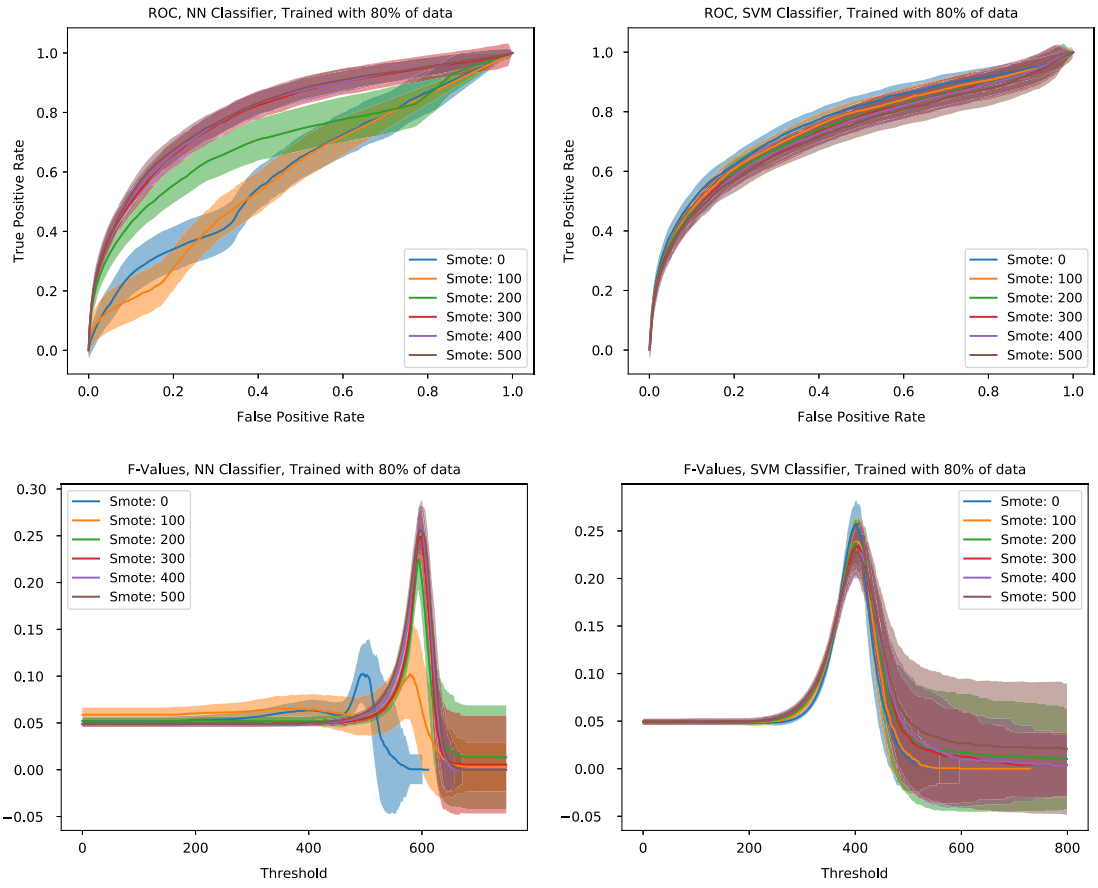

Fig. 5. ROC and F-curves generated for both SVM and neural network for several SMOTE values by sweeping thresholds and averaging over 100 different random cross validation splits. Here we present results after the models have been trained on 80% of the training data. Shading represents possible values to within one standard deviation of measurements.

Table 5
F-values obtained for different amounts of PERMS for one run. Dashes correspond to experiments which couldn't be run due to memory errors.

	30% Training Data		80% Training Data	
PERM	NN F-Value	SVM F-Value	NN F-Value	SVM F-Value
100000	0.2857	–	0.3453	–
10000	0.3034	0.2989	0.3488	–
2000	0.2831	0.3306	0.3956	0.3585
1800	0.2820	0.3120	0.4096	0.3486
1600	0.2837	0.3093	0.3409	0.3333
1400	0.2881	0.3018	0.4103	0.3364
1200	0.2857	0.3164	0.3944	0.3636
800	0.2919	0.3067	0.3750	0.3093
600	0.2953	0.2754	0.3951	0.2887
500	0.2619	0.2676	0.4110	0.2727
400	0.2702	0.2970	**0.4500**	0.3218
300	0.2181	0.2672	0.3607	0.2558
200	0.2331	0.2759	0.2597	0.2954

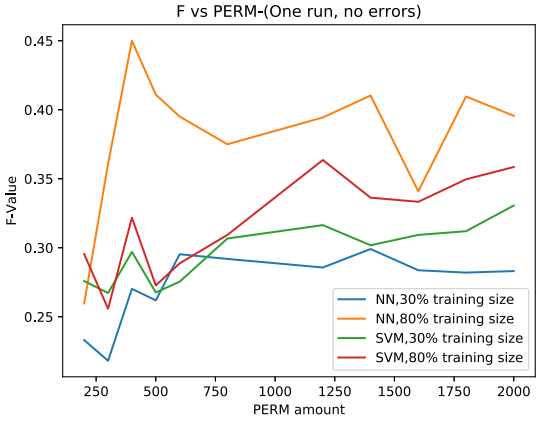

Fig. 6. Plot of permutation F-values up to PERM 2000.

the SVM to perform better for such architectures. The neural network however, was better at predicting Hodge numbers than the SVM. Here the neural network architecture is far from trivial, and its success over SVMs is likely due to its greater flexibility with non-linear data. To benchmark the performance of each model we use cross validation and take a variety of statistical measures where appropriate, including accuracy, Wilson confidence interval, F-values, and the area under the receiving operator characteristic (ROC) curve (AUC). Errors were obtained by averaging over a large sample of cross validation splits and taking standard deviations. Models are optimised by maximising the appropriate statistical measure. This is achieved either by varying the model by hand or by implementing a genetic algorithm. Remarkable accuracies can be achieved, even when, for instance, trying to predict the exact values of Hodge numbers.

This work serves as a proof of concept for exploring the geometric features of Calabi–Yau manifolds using machine learning beyond binary classifiers and feedforward neural networks. In future work, we intend to apply the same techniques to study the Kreuzer–Skarke [29] list of half a billion reflexive polytopes and the toric Calabi–Yau threefolds obtained from this dataset [30]. Work in progress extends the investigations in this paper to the CICY fourfolds [31] and cohomology of bundles over CICY threefolds.

Acknowledgements

This paper is based on the Masters of Physics project of KB with YHH, both of whom would like to thank the Rudolf Peierls Centre for Theoretical Physics, University of Oxford for the provision of resources. YHH would also like to thank the Science and Technology Facilities Council, UK, for grant ST/J00037X/1, the Chinese Ministry of Education, for a Chang-Jiang Chair Professorship at NanKai University and the City of Tian-Jin for a Qian–Ren Scholarship, as well as Merton College, University of Oxford, for her enduring support. VJ is supported by the South African Research Chairs Initiative of the DST/NRF grant No. 78554. He thanks Tsinghua University, Beijing, for generous hospitality during the completion of this work. We thank participants at the "Tsinghua Workshop on Machine Learning in Geometry and Physics 2018" for comments. CM would like to thank Google DeepMind for facilitating helpful discussions with its various members.

References

[1] Y.-H. He, Deep-learning the landscape, arXiv preprint, arXiv:1706.02714.
[2] Y.-H. He, Machine-learning the string landscape, Phys. Lett. B 774 (2017) 564–568, https://doi.org/10.1016/j.physletb.2017.10.024.
[3] D. Krefl, R.-K. Seong, Machine learning of Calabi–Yau volumes, Phys. Rev. D 96 (6) (2017) 066014.
[4] Y.-N. Wang, Z. Zhang, Learning non-higgsable gauge groups in 4d f-theory, arXiv preprint, arXiv:1804.07296.
[5] J. Carifio, J. Halverson, D. Krioukov, B.D. Nelson, Machine learning in the string landscape, J. High Energy Phys. 2017 (9) (2017) 157.
[6] F. Ruehle, Evolving neural networks with genetic algorithms to study the string landscape, J. High Energy Phys. 2017 (8) (2017) 38.
[7] S. Abel, J. Rizos, Genetic algorithms and the search for viable string vacua, J. High Energy Phys. 2014 (8) (2014) 10.
[8] J. Carifio, W.J. Cunningham, J. Halverson, D. Krioukov, C. Long, B.D. Nelson, Vacuum selection from cosmology on networks of string geometries, arXiv: 1711.06685.
[9] P. Green, T. Hübsch, Calabi–Yau manifolds as complete intersections in products of complex projective spaces, Commun. Math. Phys. 109 (1) (1987) 99–108.
[10] P. Candelas, A.M. Dale, C. Lütken, R. Schimmrigk, Complete intersection Calabi–Yau manifolds, Nucl. Phys. B 298 (3) (1988) 493–525.
[11] P. Candelas, R. Davies, New Calabi–Yau manifolds with small Hodge numbers, Fortschr. Phys. 58 (2010) 383–466, https://doi.org/10.1002/prop.200900105, arXiv:0809.4681.
[12] V. Braun, On free quotients of complete intersection Calabi–Yau manifolds, J. High Energy Phys. 04 (2011) 005.
[13] A. Lukas, C. Mishra, Discrete symmetries of complete intersection Calabi–Yau manifolds, arXiv preprint, arXiv:1708.08943.
[14] P. Candelas, C. Mishra, Highly symmetric quintic quotients, Fortschr. Phys. 66 (4) (2018) 1800017.
[15] P. Candelas, A. Constantin, Completing the web of Z_3 – quotients of complete intersection Calabi–Yau manifolds, Fortschr. Phys. 60 (2012) 345–369, https://doi.org/10.1002/prop.201200044, arXiv:1010.1878.
[16] P. Candelas, A. Constantin, C. Mishra, Hodge numbers for CICYs with symmetries of order divisible by 4, Fortschr. Phys. 64 (6–7) (2016) 463–509.
[17] A. Constantin, J. Gray, A. Lukas, Hodge numbers for all CICY quotients, J. High Energy Phys. 2017 (1) (2017) 1.
[18] P. Candelas, A. Constantin, C. Mishra, Calabi–Yau threefolds with small hodge numbers, arXiv preprint, arXiv:1602.06303.
[19] L.B. Anderson, Y.-H. He, A. Lukas, Heterotic compactification, an algorithmic approach, J. High Energy Phys. 0707 (2007) 049, https://doi.org/10.1088/1126-6708/2007/07/049, arXiv:hep-th/0702210.
[20] L.B. Anderson, Y.-H. He, A. Lukas, Monad bundles in heterotic string compactifications, J. High Energy Phys. 0807 (2008) 104, https://doi.org/10.1088/1126-6708/2008/07/104, arXiv:0805.2875.
[21] L.B. Anderson, J. Gray, Y.-H. He, A. Lukas, Exploring positive monad bundles and a new heterotic standard model, J. High Energy Phys. 2010 (2) (2010) 1.
[22] L.B. Anderson, J. Gray, A. Lukas, E. Palti, Two hundred heterotic standard models on smooth Calabi–Yau threefolds, Phys. Rev. D 84 (2011) 106005, https://doi.org/10.1103/PhysRevD.84.106005, arXiv:1106.4804.
[23] L.B. Anderson, J. Gray, A. Lukas, E. Palti, Heterotic line bundle standard models, J. High Energy Phys. 1206 (2012) 113, https://doi.org/10.1007/JHEP06(2012)113, arXiv:1202.1757.
[24] A. Lukas, L. Anderson, J. Gray, Y.-H. He, S.-J.L. Lee, CICY list including Hodge numbers and freely-acting discrete symmetries, Data available online at http://www-thphys.physics.ox.ac.uk/projects/CalabiYau/cicylist/index.html, 2007.
[25] L.B. Anderson, X. Gao, J. Gray, S.-J. Lee, Fibrations in CICY threefolds, J. High Energy Phys. 2017 (10) (2017) 77.
[26] L. Anderson, X. Gao, J. Gray, S.J. Lee, The favorable CICY list and its fibrations, http://www1.phys.vt.edu/cicydata/, 2017.
[27] T. Hubsch, Calabi–Yau Manifolds, a Bestiary for Physicists, World Scientific, 1992.
[28] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, Smote: synthetic minority over-sampling technique, J. Artif. Intell. Res. 16 (2002).
[29] M. Kreuzer, H. Skarke, Complete classification of reflexive polyhedra in four-dimensions, Adv. Theor. Math. Phys. 4 (2002) 1209–1230.
[30] R. Altman, J. Gray, Y.-H. He, V. Jejjala, B.D. Nelson, A Calabi–Yau database: threefolds constructed from the Kreuzer–Skarke list, J. High Energy Phys. 02 (2015) 158, https://doi.org/10.1007/JHEP02(2015)158, arXiv:1411.1418.
[31] J. Gray, A.S. Haupt, A. Lukas, All complete intersection Calabi–Yau four-folds, J. High Energy Phys. 07 (2013).