Deep Learning for K3 Fibrations in Heterotic/Type IIA String Duality

Aaron Vermeersch*

(Dated: May 15, 2023)

The development of Large Language Models, such as the recently released GPT-4, has revolutionized the field of Machine Learning and opened new avenues for interdisciplinary research. Prompt Engineering, a methodology for designing effective input prompts to guide these AI models, will emerge as a powerful tool for accelerating research efforts. In this study, we leverage Prompt Engineering with GPT-4 to address the problem of predicting $K3$ Fibrations in Calabi-Yau manifolds embedded in toric varieties with a single weight system. Out of the 184,026 weights spaces previously discovered, 101,495 remained unclassified concerning the presence of a $K3$ projection. By utilizing an ensemble of Deep Neural Networks, we are able to predict the existence of the K3 fibration. Lastly, we evaluate the ability of these models to predict the complete spectrum of possible Hodge Numbers and other properties of reflexive polytopes. These results should be useful when studying the heterotic/Type IIA string duality in F-Theory.

I. INTRODUCTION

The emergence of Large Language Models (LLMs), exemplified by the groundbreaking GPT-4 [1], has revolutionized the field of Artificial Intelligence and unlocked new avenues for interdisciplinary research. A key technique that has evolved alongside these AI models is Prompt Engineering, which centers on crafting effective input prompts that guide LLMs towards specific outputs. Here, we investigate the application of GPT-4 in accelerating software development related to fundamental physics research, by using GPT-4 to construct Deep Neural Networks to investigate $K3$ fibered Calabi-Yau manifolds embedded in toric varieties with single weight systems.

$K3$ fibrations become significant when examining the dualities between the $E_8 \times E_8$ heterotic string compactified on $K3 \times T^2$ and the Type IIA string compactified on a Calabi-Yau manifold. The seminal work [2], brought to attention that the Type IIA string, when compactified on a Calabi-Yau manifold, can be conceptualized as a fiber bundle. This bundle has a $\mathbb{P}^1$ base and a $K3$ surface serving as the generic fiber. The heterotic string could then be compactified on $K3 \times T^2$ which can be constructed as a bundle with base $\mathbb{P}^1$ and with T^4 as its generic fiber.

It was demonstrated that K3 fibrations aligned with the properties of the dual heterotic string when compactified on $K3 \times T^2$ [3]. This duality was explicitly articulated in [4]. Under the assumption that the Type IIA dual of the weakly coupled heterotic string resides in the Calabi-Yau phase, it follows that the Calabi-Yau manifold upon which it was compactified must manifest as a K3 fibration.

The process of embedding Calabi-Yau manifolds in toric varieties was elaborated in [5]. Subsequently, the presence of K3 fibrations within these manifolds was investigated in [6]. For a more comprehensive exploration of K3 fibrations, readers can refer to [7, 8]. The role of K3 fibrations in the context of string theory is discussed in detail in [9].

In recent years, Machine Learning has become important in exploring the vast landscape of string theory vacua. Machine learning models have been employed to predict properties of Calabi-Yau manifolds, such as Hodge Numbers [10], leading to advancements in our understanding of these structures. Furthermore, Deep Learning has been applied to investigate a multitude of topics [11] pertaining to string theory including brane webs [12], topological properties of Calabi-Yau manifolds in F-theory [13], the string vacua in heterotic string theory [14],

* aaronvermeersch@gmail.com

and others [15].

In this study, we harness the power of GPT-4 for the development of our Deep Learning software, which is employed to identify K3 fibrations in Calabi-Yau manifolds embedded in toric varieties. We use an ensemble of Deep Neural Networks and the weight system of the space to predict K3 fibrations in 101,495 previously unclassified spaces. The remainder of this paper is organized as follows: Section 2 offers an overview of how from a reflexive polytope a Calabi-Yau manifold can be constructed and embedded in toric varieties and that this can produce a $K3$ fibration. Section 3 delves into the impact of Prompt Engineering in this work and the Deep Learning methodology employed in our analysis, while Section 4 presents our results. Finally, in Section 5, we suggest potential avenues for future research.

II. $K3$-FIBERED CY MANIFOLDS & REFLEXIVE POLYTOPES

In this section, we explore how a toric variety can be constructed from a polytope and if that polytope is reflexive then certain hypersurfaces of this space describe Calabi-Yau manifolds. By restricting ourselves to four dimensions, any reflexive polytope with a projection onto a 3 dimensional sublattice which produces a polyhedron will define a $K3$ fibration.

Let M and N be two dual lattices, each of which are isomorphic to $\mathbb{Z}^n$, and with their respective real extensions as $M_{\mathbb{R}}$ and $N_{\mathbb{R}}$. A lattice polytope Δ, defined on $M_{\mathbb{R}}$, is a polytope such that the vertices of Δ lie on the lattice points. Δ is then considered reflexive if it, and its dual defined as

$$\Delta^* = \{y \in N_{\mathbb{R}} : \langle x, y\rangle \geq -1, \forall x \in \Delta\} \quad (1)$$

contain only one interior point at the origin. From here, we follow the holomorphic quotient approach [16] when constructing our toric variety V_Σ. Let us begin by defining the fan Σ on $N_{\mathbb{R}}$, which is based on a maximal triangulation of Δ^*. We then assign assign coordinates, $z_1, \ldots, z_N$, to every one-dimensional cone within Σ. The integer points of Δ^* represent the generators $\nu_1, \ldots, \nu_N$ of the one-dimensional cones. This process constructs a $\mathbb{C}^N$ space, from which we remove the exceptional set. The exceptional set is defined as

$$Z_\Sigma = \bigcup_I \{(z_1, \ldots, z_N) : z_i = 0,\ \forall i \in I\} \quad (2)$$

In this equation, the union extends over all sets I such that the generators do not reside within a cone in Σ. If we consider the product of a finite abelian group and $(\mathbb{C}^*)^{N-n}$, the toric variety is given by the quotient of $\mathbb{C}^N \backslash Z_\Sigma$, under the action of the equivalence relation

$$(z_1, \ldots, z_N) \sim (\lambda^{w_j^1} z_1, \ldots, \lambda^{w_j^N} z_N) \quad (3)$$

subject to $\sum_k w_j^k x^k = 0$ [17]. Consequently, a Calabi-Yau hypersurface in V_Σ is given by the zero locus of

$$P(z_1, \ldots, z_N) = \sum_{x \in \Delta \cap M} a_x \prod_{k=1}^{N} z_k^{<v_k, x>+1} \quad (4)$$

with the Hodge numbers h_{11} and h_{21}, for the above Calabi-Yau hypersurface from Batyrev [18]:

$$h_{11} = l(\Delta^*) - 5 - \sum_{codim\theta^*=1} l^*(\theta^*) + \sum_{codim\theta^*=2} l^*(\theta^*) l^*(\theta) \quad (5)$$

$$h_{21} = l(\Delta) - 5 - \sum_{codim\theta=1} l^*(\theta) + \sum_{codim\theta=2} l^*(\theta^*) l^*(\theta) \quad (6)$$

In this context, θ and θ^* correspond to faces of the polytopes Δ and Δ^* respectively. The functions l and l^* signify the counts of integer points and interior integer points contained within these polytopes. Batyrev's construction eloquently exhibits mirror symmetry, as it allows for the interchangeability of Δ with Δ^*

and h_{11} with h_{21} without altering the values of h_{11} with h_{21}.

In this study, we describe the fibration using a pair $(\Delta, \Delta^*) \subset M \times N$ of reflexive polytopes. We introduce a distinguished three-dimensional sub-lattice N_3 onto N, such that the polytope Δ_3^*, defined on the sub-lattice is a three dimensional reflexive polytope. This Δ_3^* acts as the fiber, splitting Δ^* into a top and a bottom. By projecting the fan of the fibration along the linear space spanned by Δ_3^*, we can derive the fan corresponding to the base space. Restricting to four dimensions, the base space invariably becomes $\mathbb{P}^1$ [19].

The generation of the $K3$-fibered toric Calabi-Yau data involved developing an algorithm that constructs a set of maximal polytopes, in such a way that any reflexive polytope can be represented as a subpolytope of a polytope in this set [20]. Each maximal polytope in the set was defined by a weight system comprising positive rational numbers $q_i = w_i/d$, where $\sum q_i = 1$ and $\sum w_i = d$. In this investigation, our focus is on the case of a single weight system that is defined by five weights.

Once this set was established, they checked for reflexivity by constructing all subpolyhedra of all polyhedra within the set [6]. The 184,026 weight systems pertaining to four-dimensional polytopes were originally computed in [21]. Later, [6] revisited these weight systems to ascertain the existence of the $K3$ fibrations.The dataset utilized in this study is accessible on Kreuzer and Sharke's website.

III. ARTIFICAL INTELLIGENCE

A. GPT-4

GPT-4 (Generative Pre-trained Transformer 4) is a state-of-the-art language model developed by OpenAI, building upon the success of its predecessor, GPT-3 [22], and offering substantial improvements in scale, capabilities, and performance [23]. The transformer architecture underpinning GPT-4 leverages self-attention mechanisms [24] to process input sequences in parallel rather than sequentially, enabling the model to effectively capture long-range dependencies and intricate patterns in text data.

Prompt Engineering [25] is a technique that optimizes input format and context for models like GPT-4, which are capable of generalizing from minimal input in one-shot or few-shot learning scenarios [22]. This approach facilitates more accurate and relevant responses to specific queries or tasks.

A key application of Prompt Engineering lies in expediting software development. GPT-4 can generate high-quality code, identify and rectify errors, and provide succinct explanations of complex code structures when given effective prompts. This process not only saves time but also alleviates cognitive burden on developers, enabling them to concentrate on more crucial aspects of software development.

Furthermore, Prompt Engineering in software development carries significant implications for fundamental research. Automating mundane and repetitive tasks allows researchers to dedicate more time and resources to exploring novel ideas and innovative methodologies. GPT-4's capacity to comprehend and generate human-like text empowers researchers to efficiently extract insights from copious volumes of literature, thereby streamlining knowledge discovery. Consequently, Prompt Engineering and GPT-4 hold the potential to substantially accelerate scientific progress and contribute to the broader advancement of human knowledge.

B. Feature Engineering

To optimize the performance of our machine learning models, we employed a variety of preprocessing techniques to both input and output variables in the dataset, with the aim of transforming the data into an appropriate format and scale that would facilitate the learning process and improve model accuracy.

For input variables to the models, specifically the weight system, standard scaling was applied. Standard scaling is a normalization tech-

nique whereby the features are made to have zero mean and unit variance by subtracting the mean value and dividing by the standard deviation for each feature. Employing standard scaling is particularly advantageous when dealing with features of differing scales, as it ensures that no single feature disproportionately influences the model's learning process due to its magnitude.

The formula for standard scaling is as follows:

$$z = \frac{x - \mu}{\sigma} \tag{7}$$

where x represents the original feature value, μ is the mean of the feature, σ is the standard deviation, and z is the standardized value.

In tasks involving the prediction of categorical variables, such as the number of K3 projections, one-hot encoding was employed to transform the output variable. One-hot encoding is a technique that converts a categorical variable into a binary vector in which each category is represented by a unique binary sequence with a single "1" in the position corresponding to the category and "0"s elsewhere. This representation enables the model to discern distinct patterns for each category without presupposing any ordinal relationship between them.

C. Deep Learning Models

Neural networks serve as powerful computational models designed to recognize patterns and representations within large, complex datasets. A common architecture for neural networks consists of densely connected layers, also known as fully connected or dense layers. In these layers, each neuron connects to every neuron in adjacent layers, enabling the network to learn intricate patterns from the input data[26].

Activation functions introduce non-linearity into neural networks, enabling them to model complex relationships between inputs and outputs. In our work, we utilize the Rectified Linear Unit (ReLU) as the activation function for the hidden layers. ReLU, defined as $f(x) = max(0, x)$ where x represents the input value, is computationally efficient and simple, making it popular in deep learning models.

In classification problems, the last layer of the neural network employs a softmax function. The softmax function, defined as

$$softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \tag{8}$$

for $i = 1, \ldots, K$, where x_i are the input values and K is the number of classes, converts the raw output values into probabilities representing the likelihood of each input belonging to a specific class. By using the softmax function, we can interpret the output as class probabilities, aiding decision-making and understanding the model's confidence level.

The Adam optimizer, a popular optimization algorithm for training deep learning models, combines the advantages of Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp) to compute adaptive learning rates for individual parameters based on the first and second moments of the gradients. The learning rate determines the step size for updating model parameters during each training iteration. Adam's ability to automatically adjust the learning rate for each parameter throughout the training process reduces manual hyperparameter tuning and makes it well-suited for handling noisy, sparse gradients which are common challenges in deep learning. In this work, all of the neural networks were trained using an Adam optimizer.

To further refine our model, we employed early stopping, a regularization technique that mitigates overfitting in deep learning and neural networks. Overfitting arises when a model captures noise or random fluctuations instead of the underlying patterns, impairing generalization to unseen data. Early stopping monitors the model's performance on both the training and validation sets, ceasing the training process when validation performance deteriorates, typically after a predetermined number of epochs without improvement. Early stopping curtails

overfitting, conserves computational resources, and streamlines model selection, culminating in superior models that generalize well to new data.

Dropout was also employed as a regularization technique to further mitigate overfitting in the neural networks. Dropout randomly "drops out" a certain percentage of neurons during training, preventing the model from relying too heavily on individual neurons and promoting the development of robust, generalizable features. In our work, dropout follows each densely connected layer, with varying dropout rates determined through hyperparameter tuning.

Training a neural network requires defining a loss function to measure the divergence between predicted outputs and true target values. Mean Squared Error (MSE) and Categorical Cross-Entropy (CCE) are two commonly used loss functions for regression tasks with discrete target values and classification tasks, respectively.

MSE is a widely used loss function for regression tasks involving discrete target values. It calculates the average squared difference between predicted and actual values, as illustrated in the following equation:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_{\text{true}}^{(i)} - y_{\text{pred}}^{(i)})^2 \tag{9}$$

Here, n signifies the number of samples, y_{true} corresponds to the true target values, and y_{pred} denotes the predicted values.

For multi-class classification tasks, where the objective is to predict one of several potential categories, Categorical Cross-Entropy is a widely utilized loss function. It quantifies the dissimilarity between the true probability distribution and the predicted probability distribution across the classes, as demonstrated in the following formula:

$$CCE = -\sum_{i=1}^{n}(y_{\text{true}}^{i}\log(y_{\text{pred}}^{i})) \tag{10}$$

To evaluate the performance of a neural network, distinct metrics are employed for regression tasks with discrete target values and classification tasks. For regression tasks, which entail predicting discrete target values, Mean Absolute Error (MAE) serves as an evaluation metric, calculating the average absolute difference between predicted and actual values:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_{\text{true}}^{i} - y_{\text{pred}}^{i}| \tag{11}$$

In classification tasks, which involve predicting categorical variables, accuracy and F1-score serve as evaluation metrics. Accuracy represents the proportion of correctly classified instances out of the total instances. The F1-score, on the other hand, is the harmonic mean of precision and recall, calculated as follows:

$$F1\text{-}score = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{12}$$

Precision is defined as the proportion of true positive predictions out of the total predicted positives, while recall is the proportion of true positives out of the total actual positives.

A confusion matrix is another valuable tool for assessing a classifier's performance. This matrix provides a visual representation of a classifier's accuracy and robustness on labeled data. Each row corresponds to the true class labels, and each column represents the predicted class labels. The cell at the intersection of a specific row and column displays the number of instances belonging to the true class (row) and predicted as the corresponding class (column). The matrix's diagonal elements indicate the correctly classified instances, while the off-diagonal elements denote misclassifications.

To optimize our neural network's performance, we carried out hyperparameter optimization by identifying the most favorable combination of hyperparameters that minimized validation loss. We conducted 50 trials, each employing a distinct set of hyperparameters, and selected the configuration yielding the lowest validation loss. The chosen hyperparameters were subsequently utilized to train the final model. Hyperparameters considered in our

search encompassed the number of neurons in each layer, the degree of dropout preceding each layer, and the number of layers, which ranged from 1 to 4.

The hyperparameter space was explored using a random algorithm, whereby a random combination of hyperparameters was selected, the model trained, and subsequently evaluated. The best-performing model was then chosen for the ensemble method which would constitute our final result.

To further enhance the predictive accuracy and stability of our model, we incorporated Bootstrap Aggregating, an ensemble method commonly referred to as bagging, to amplify predictive accuracy and stability. Bagging entails training multiple base models (5 estimators in our case) on distinct subsets of the training data, sampled with replacement. The final prediction is achieved by averaging individual model predictions for regression tasks or by majority voting for classification tasks.

To rigorously evaluate our bagging ensemble, we employed cross-validation, a technique that partitions the dataset into k equal-sized folds, using k-1 folds for training and the remaining fold for validation. This process is repeated k times, with each fold serving as the validation set once. We implemented 3-fold cross-validation, ensuring that all data points contributed to both training and validation, thus providing a more reliable estimate of the model's performance.

Our neural network's training capitalized on an NVIDIA GeForce RTX 3060 GPU, which significantly outperforms traditional CPU-based training. GPUs excel at deep learning tasks due to their parallel computational capabilities. This parallelism efficiently offloads matrix and vector operations, prevalent in deep learning algorithms, from the CPU to the GPU. GPU offloading transfers computationally-intensive tasks to the GPU, allowing simultaneous CPU and GPU processing, thereby enhancing overall efficiency and expediting computation. In deep learning contexts, GPU offloading accelerates training, promotes rapid experimentation, and ultimately yields more accurate and robust models.

IV. RESULTS

In our study, we conducted an extensive training process involving 600 neural networks to identify the optimal hyperparameters for all the predictions we aimed to make. The outcomes of this hyperparameter search are presented in Table I. Utilizing the optimized hyperparameters, we assembled an ensemble of five neural networks for each quantity and provide a detailed analysis of their performance in the following sections.

For training our machine learning models, we utilize the $K3$ fibered toric Calabi-Yau dataset as follows: the input features solely comprise of the weight systems in [6]. To predict each quantity, we must train a new set of models on the weight system and the specific quantity in question. Initially, we examine how well our models can predict the number of reflexive projections Π corresponding to the $K3$ fibrations, as well as the number of projections onto reflexive facets. We also convert the latter problem into a binary classification task and predict the existence of the 1st and 2nd projections onto reflexive facets. Next, we make predictions on the Hodge numbers as outlined by Equations 3 and 4. Due to the sparsity of h_{21} at its highest values, we have had to limit h_{21} to only those values below 200; otherwise, there are insufficient instances within each value to adequately train the model. Lastly, we will make predictions on the unclassified weight spaces concerning the existence of a $K3$ fibration.

The results of our regression models are presented in Table II. The table displays the Mean Absolute Errors for each quantity, providing an insight into the performance of our regressors. For predicting the number of K3 projections, the regression model yielded an MAE of 0.1848 ± 0.0021, indicating an effective prediction of this particular quantity. The regressors for the number of projections onto $K3$ Facets and predicting the existence of the projection onto the 1st K3 Facet regressors

Property	Neurons/Layer	Dropout/Layer	Learning Rate
R: # of $K3$ Projections	128, 128, 192, 224	0.4, 0.1, 0.4, 0.2	3.168E-4
C: # of $K3$ Projections	256, 192, 64, 256	0.5, 0.1, 0.2, 0.3	7.371E-4
R: # of Π onto $K3$ Facet	64, 224, 256, 192	0.1, 0.5, 0.1, 0.4	1.966E-4
C: # of Π onto $K3$ Facet	160, 224, 32, 256	0.1, 0.3, 0.3, 0.2	1.915E-4
R: Existence of 1st Facet	192, 160, 192, 64	0.1, 0.3, 0.5, 0.5	1.350E-4
C: Existence of 1st Facet	192, 256, 128, 224	0.1, 0.1, 0.2, 0.3	7.396E-5
R: Existence of 2nd Facet	128, 256, 128	0.2, 0.3, 0.3	1.257E-4
C: Existence of 2nd Facet	192, 224, 32	0.2, 0.5, 0.3	1.172E-4
R: h_{11}	96, 128	0.2, 0.2	7.196E-3
C: h_{11}	128, 160, 192, 224	0.3, 0.3, 0.2, 0.4	3.040E-4
R: $h_{21} \leq 200$	224, 224	0.3, 0.3	2.723E-4
C: $h_{21} \leq 200$	128, 224, 224	0.2, 0.3, 0.1	7.018E-4

TABLE I: Optimal hyperparameters for each neural network model. The table presents the best-performing configuration of neurons per layer, dropout rate per layer, and learning rate for each model, identified from a total of 600 trained networks. Regression models are indicated by "R" and classification models by "C".

had MAEs of 0.3169 ± 0.0002 and 0.2978 ± 0.0012, respectively, demonstrating satisfactory predictive performance. Remarkably, predicting the existence of the projection onto the 2nd K3 Facet regressor outperformed the others, achieving an MAE of 0.0294 ± 0.0004, which suggests excellent predictive accuracy.

In contrast, the h_{11} and h_{21} regressors had higher MAEs of 23.89 ± 0.2698 and 2.512 ± 0.049, respectively. These results imply that there is room for improvement in the prediction of these quantities. Overall, our model demonstrated strong predictive capabilities for most of the considered quantities, though further optimization may be necessary for h_{11} and h_{21}.

The performance of our classifiers for various quantities is summarized in Table III, which presents both the accuracy and F1-scores for each quantity. Our classifier for predicting the number of K3 projections achieved an accuracy of 87.84 ± 0.19% and an F1-score of 0.8758 ± 0.0026, demonstrating strong predictive performance. The classifiers for predicting the number of projections onto $K3$ Facets and the existence of the projection onto the 1st K3 Facet also exhibited satisfactory results, with accuracies of 76.70 ± 0.001% and 79.463 ± 0.007%, respectively, and corresponding F1-scores of 0.7609 ± 0.0004 and 0.7946 ± 0.0002. The existence of the projection onto the 2nd K3 Facet classifier outperformed the others, achieving an impressive accuracy of 98.128 ± 0.001%

Property	MAE
# of $K3$ Projections	0.1848 ± 0.0021
# of Π onto Facets	0.3169 ± 0.0002
Existence of 1st Facet	0.2978 ± 0.0012
Existence of 2nd Facet	0.0294 ± 0.0004
h_{11}	23.89 ± 0.2698
$h_{21} \leq 200$	2.512 ± 0.049

TABLE II: Mean Absolute Errors for the regression models associated with each property. The table presents the average prediction error, along with the standard error, for each model in predicting the corresponding property values.

and an F1-score of 0.9727 ± 0.0001, indicating exceptional classification capabilities.

Property	Accuracy (%)	f1-score
# of $K3$ Projections	87.84 ± 0.19	0.8758 ± 0.0026
# of Π onto Facets	76.70 ± 0.001	0.7609 ± 0.0004
Existence of 1st Facet	79.463 ± 0.007	0.7946 ± 0.0002
Existence of 2nd Facet	98.128 ± 0.001	0.9727 ± 0.0001
h_{11}	11.83 ± 0.01	0.0114 ± 0.0004
$h_{21} \leq 200$	14.96 ± 0.42	0.1655 ± 0.0084

TABLE III: Performance metrics for the classification models associated with each property. The table presents the accuracy (in percentage) and F1-scores, along with their respective standard errors, for each model in classifying the corresponding property values.

In the context of the h_{11} and h_{21} classifiers, the accuracy of 11.83 ± 0.01% and F1-score of 0.0114 ± 0.0004 for h_{11}, and accuracy of 14.96 ± 0.42% and F1-score of 0.1655 ± 0.0084 for h_{21}, reveal that these classifiers faced difficulties predicting all Hodge Numbers. The relatively poor performance can be attributed to two factors: the classification problem for Hodge Numbers has significantly more classes than the other properties we investigated (352 and 200 classes for h_{11} and h_{21}, respectively), and the scarcity of training instances for the largest Hodge Numbers. This issue was particularly prominent for the h_{21} classifier, which was limited to values below 200 due to insufficient examples. Comparing the results to a naive classifier that randomly selects h_{11} or h_{21} values, we would achieve accuracies of 0.28% and 0.5%, respectively. In this context, our classifiers did learn something about Hodge Numbers as they outperformed the naive classifier, but predicting the entire range is challenging.

We now focus on predicting the number $K3$ projections (denoted #Π), which corresponds to the $K3$ fibrations, using a classifier. Before presenting the final prediction, we will examine the classifier's performance in greater detail by analyzing its confusion matrix 1. To evaluate the classifier's performance, we can calculate the F1-score for each class. For spaces

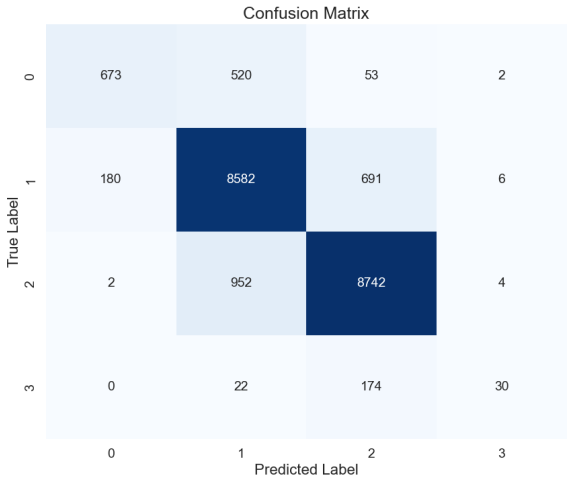

FIG. 1: Confusion matrix for a 4-class classification problem. The rows represent the true class labels, and the columns represent the predicted class labels. The diagonal elements represent the correctly classified instances for each class, while the off-diagonal elements represent the misclassifications. Overall classification performance metrics such as accuracy, precision, recall, and F1-score can be derived from this matrix.

with #Π = 0, the F1-score is 0.53, indicating that the classifier's performance in predicting #Π = 0 could be improved. Spaces with #Π = 1 has an F1-score of 0.92, demonstrating that the classifier performs well for this class. Similarly, spaces with #Π = 2 have a relatively high F1-score of 0.91, suggesting that the classifier's performance in predicting #Π = 2 instances are strong. However, spaces #Π = 3 has the lowest F1-score at 0.20, indicating that the classifier's performance for #Π = 3 is weak and likely stems from the small proportion of spaces which have #Π = 3 making training for this class difficult. Overall, the classifier exhibits strong performance in #Π = 1 or #Π = 2, while for the spaces with #Π = 0 and #Π = 3 the classifier under performs.

Our ensemble of deep neural networks, combined with the weight system of the space, en-

abled us to classify the previously unclassified Calabi-Yau spaces regarding the presence of reflexive polytopes. Out of the 184,026 spaces discovered, 101,495 remained unclassified. After applying our ensemble model, we predicted the number of $K3$ projections for these unclassified spaces, as shown in Table IV. The predictions revealed that 16,531 of these spaces possess a single projection, while 83,498 spaces contain two projections. Furthermore, 1,466 spaces were found to have three projections.

#Π	Previously Known	ML Predictions	Total
0	5,130	0	5,130
1	37,215	16,531	53,746
2	39,175	83,498	122,673
3	1,011	1,466	2,477

TABLE IV: Predicted Number of K3 Projections for Unclassified Spaces. The predictions were obtained using an ensemble of classifiers, and the final class assignment was determined by selecting the class with the highest output value (argmax) from the neural network.

In our results, we present the predictions of the ensemble-based classifier for the number of $K3$ projections in the first 20 spaces. The classifier, constructed using an ensemble of neural networks, provides a confidence level for each prediction, reflecting the level of certainty in the model's assessment. These confidence levels, as shown in Table V, are derived from the output layer of a neural network with a softmax activation function and range from 0 to 1. The sum of confidence levels across all four classes equals one, indicating a normalized probability distribution over the classes. It is evident from the table that the classifier is generally more confident in its predictions for certain spaces, while displaying a lower level of confidence in others. This variation in confidence levels highlights the complexity of the classification task and the importance of considering multiple factors when assessing the performance of the ensemble-based classifier. [27]

w1	w2	w3	w4	w5	Predicted #Π	Confidence
3	4	5	6	6	1.0	0.4281
3	4	5	6	7	1.0	0.4286
3	4	5	6	8	1.0	0.4291
3	4	5	7	7	1.0	0.4428
4	4	5	6	7	1.0	0.4479
3	4	5	6	9	1.0	0.4298
3	4	5	7	8	1.0	0.4434
3	5	6	6	7	1.0	0.3945
4	5	5	6	7	2.0	0.4093
3	4	5	6	10	1.0	0.4306
3	4	5	7	9	1.0	0.4441
3	4	5	8	8	1.0	0.4666
4	5	6	6	7	2.0	0.4105
3	4	5	6	11	1.0	0.4314
3	4	5	7	10	1.0	0.4448
3	4	5	8	9	1.0	0.4676
3	5	6	7	8	1.0	0.4028
3	4	5	6	12	1.0	0.4323
3	4	5	7	11	1.0	0.4456
3	4	5	8	10	1.0	0.4685

TABLE V: Predictions of the first 20 spaces from the ensemble-based classifier constructed using neural networks. The table presents the weight system of the space, the predicted number of $K3$ projections, and the confidence level for each prediction for the first 20 spaces.

V. CONCLUSIONS

In this paper, we have presented a study on the application of machine learning techniques to explore the properties of Calabi-Yau manifolds embedded in toric varieties with single weight systems. Our primary focus was on identifying the presence of$K3$ fibrations in these manifolds. After showing the connection between Calabi-Yau spaces and reflexive polytopes, we demonstrated the effectiveness of ma-

chine learning in predicting the properties of reflexive polytopes with remarkable accuracy.

Our research significantly enriches the understanding of heterotic/Type IIA string duality, as evidenced by the identification of more than twice the previously known quantity of K3 fibrations. This accomplishment does more than expand our comprehension of the geometric landscape in string theory; it also carries potential implications for the understanding of the possible vacua in string phenomenology.

Moreover, our results showcase the power of machine learning as an invaluable tool in the study of complex geometric structures in the context of string theory and mathematical physics. By harnessing the capabilities of Deep Neural Networks, we have opened up new avenues for exploring the rich geometric structures underlying fundamental aspects of theoretical physics. Furthermore, the incorporation of Prompt Engineering with GPT-4 illustrates the potential of Large Language Models to accelerate scientific advancements across various fields. In this paper, GPT-4 contributed all of the software given the prompts that we engineered. The use of these powerful AI tools to expedite the discovery process and optimize computational approaches holds great promise for accelerating innovation in the sciences.

Emboldened by the successful application of machine learning to reflexive polytopes, our future endeavor is to tackle the vast dataset of 470 million reflexive polytopes documented in [28]. Our previous investigations have indicated that reflexive polytopes are particularly amenable to deep learning methods. However, the enormous size of the dataset presents a unique challenge. To overcome this, we anticipate leveraging architectures specifically designed for parallelization across multiple GPUs, along with training techniques traditionally utilized in the development of large language models and high-performance computer vision models.

DECLARATION OF GENERATIVE AI AND AI-ASSISTED TECHNOLOGIES IN THE WRITING PROCESS

Statement: During the preparation of this work the author(s) used GPT-4 in order to expedite the software development. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.

ACKNOWLEDGMENTS

A.V. would like to express gratitude to Lauren Vermeersch for her insightful feedback and valuable suggestions during the manuscript editing process.

[1] OpenAI. GPT-4 Technical Report, 2023. [arXiv:2303.08774].

[2] C. Vafa and E. Witten. Dual string pairs with N = 1 and N = 2 supersymmetry in four dimensions. *Nuclear Physics B - Proceedings Supplements*, 46(1-3):225–247, Mar 1996. [arXiv:hep-th/9507050].

[3] A. Klemm, W. Lerche, and P. Mayr. K3-fibrations and heterotic-type II string duality. *Physics Letters B*, 357(3):313–322, Sep 1995. [arXiv:hep-th/9506112].

[4] P.S. Aspinwall and J. Louis. On the ubiquity of K3 fibrations in string duality. *Physics Letters B*, 369(3-4):233–242, Feb 1996. [hep-th/9510234].

[5] M. Audin. *The Topology of Torus Actions on Symplectic Manifolds*, volume 93 of *Progress in Math.* Birkhauser, 1991.

[6] A.C. Avram, M. Kreuzer, M. Mandelberg, and H. Skarke. Searching for K3 fibrations. *Nuclear Physics B*, 494(3):567–589, Jun 1997. [hep-th/9610154].

[7] B.H. Lian and S.T. Yau. Mirror Maps, Modular Relations and Hypergeometric Series I, 1995. [arXiv:hep-th/9507151].

[8] G Aldazabal, A Font, L.E Ibáñez, and F Quevedo. Chains of N = 2, D = 4 heterotic/type II duals. *Nuclear Physics B*, 461(1-

2):85–100, Feb 1996. [arXiv:hep-th/9510093].

[9] P.S. Aspinwall. K3 Surfaces and String Duality, 1999. [arXiv:9611137].

[10] K. Bull, Y.H He, V. Jejjala, and C. Mishra. Machine learning CICY threefolds. *Physics Letters B*, 785:65–72, Oct 2018. [arXiv:1806.03121].

[11] Y.H. He, E. Heyes, and E. Hirst. Machine Learning in Physics and Geometry, 2023. [arXiv:2303.12626].

[12] G. Arias-Tamargo, Y.H. He, E. Heyes, E. Hirst, and D. Rodriguez-Gomez. Brain webs for brane webs. *Physics Letters B*, 833:137376, Oct 2022. [arXiv:2202.05845].

[13] Y.H. He and A. Lukas. Machine learning Calabi-Yau four-folds. *Physics Letters B*, 815:136139, Apr 2021. [arXiv:2009.02544].

[14] R. Deen, Y.H. He, S.J. Lee, and A. Lukas. Machine Learning String Standard Models, 2020. [arXiv:2003.13339].

[15] J. Bao, Y.H. He, E. Heyes, and E. Hirst. Machine Learning Algebraic Geometry for Physics, 2022. [arXiv:2204.10334].

[16] D. Cox. Erratum to "The Homogeneous Coordinate Ring of a Toric Variety", along with the original paper, 2014. [arXiv:alg-geom/9210008].

[17] M. Kreuzer and H. Skarke. Calabi-Yau 4-folds and toric fibrations. *Journal of Geometry and Physics*, 26(3-4):272–290, Jul 1998. [arXiv:9701175].

[18] V.V. Batyrev. Dual Polyhedra and Mirror Symmetry for Calabi-Yau Hypersurfaces in Toric Varieties, 1993. [arXiv:alg-geom/9310003].

[19] P. Candelas, A. Constantin, and H. Skarke. An Abundance of K3 Fibrations from Polyhedra with Interchangeable Parts. *Communications in Mathematical Physics*, 324(3):937–959, Oct 2013. [arXiv:1207.4792].

[20] M. Kreuzer and H. Skarke. On the Classification of Reflexive Polyhedra. *Communications in Mathematical Physics*, 185(2):495–508, May 1997. [hep-th/9512204].

[21] H. Skarke. Weight Systems for Toric Calabi-Yau Varieties and Reflexivity of Newton Polyhedra,. *Modern Physics Letters A*, 11(20):1637–1652, Jun 1996. [arXiv:9603007].

[22] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language Models are Few-Shot Learners, 2020. [arXiv:2005.14165].

[23] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, H. Nori, H. Palangi, M. T. Ribeiro, and Y. Zhang. Sparks of Artificial General Intelligence: Early experiments with GPT-4, 2023. [arXiv:2303.12712].

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention Is All You Need, 2017. [arXiv:1706.03762].

[25] To obtain useful guidance on enhancing the quality of your prompts, please refer to the documentation provided by OpenAI onTechniques to Improve Reliability.

[26] For additional reading, see [29–31].

[27] A CSV file containing all predictions and their respective confidence levels can be accessed on `https://github.com/avermeersch/k3-fibrations`Github. Additionally, the Jupyter Notebook utilized in this study can be found at the same link.

[28] M. Kreuzer and H. Skarke. Complete classification of reflexive polyhedra in four dimensions, 2000. [arXiv:hep-th/0002240].

[29] F. Chollet. *Deep Learning with Python*. Manning, November 2017.

[30] A. Géron. *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, Sebastopol, CA, 2017.

[31] I.J. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. `http://www.deeplearningbook.org`.