Machine learning for complete intersection Calabi-Yau manifolds

Harold Erbin, Riccardo Finotello, Mohamed Tamaazousti

Machine learning for complete intersection Calabi-Yau manifolds

Harold Erbin
Université Paris-Saclay, CEA-LIST
F-91120 Palaiseau, France
Center for Theoretical Physics, Massachusetts Institute of Technology
Cambridge, MA 02139, USA
NSF AI Institute for Artificial Intelligence and Fundamental Interactions
erbin@mit.edu

Riccardo Finotello
Université Paris-Saclay, CEA-LIST
F-91120 Palaiseau, France
riccardo.finotello@cea.fr

Mohamed Tamaazousti
Université Paris-Saclay, CEA-LIST
F-91120 Palaiseau, France
mohamed.tamaazousti@cea.fr

Abstract

We describe the recent developments in using machine learning techniques to compute Hodge numbers of complete intersection Calabi-Yau (CICY) 3- and 4-folds. The main motivation is to understand how to study data from algebraic geometry and solve problems relevant for string theory with machine learning. We describe the state-of-the art methods which reach near-perfect accuracy for several Hodge numbers, and discuss extrapolating from low to high Hodge numbers, and conversely.

1 Introduction

Calabi-Yau (CY) manifolds are a very active field of research in both mathematics (algebraic geometry) and theoretical physics (string theory). They correspond to Kähler manifolds (complex manifolds with a closed 2-form) whose holonomy group is a subgroup of $\mathrm{SU}(N)$ [1]. These manifolds have a special role in string theory because they are used to describe compactifications: indeed, superstring theory predicts that spacetime must be 10-dimensional, in apparent contradiction with the observed 4-dimensional universe. The solution is to compactify 6 dimensions into a tiny volume at each spacetime point, such that they cannot be observed at the energy scales from the current experiments; consistency of string theory requires the manifold describing these six dimensions to be CY [2, 3]. The shape of the manifold used for the compactification characterizes the low-energy effective action, which should match the existing Standard Model of particle physics. For this reason, classifying CY manifolds by computing their topological and geometrical properties is of the utmost importance in string theory.

String theory is a theory of quantum gravity which also unifies all force and matter fields. Hence, finding the appropriate compactification would allow, in principle, to completely determine the physics of our universe. However, there are some important challenges for pushing this program to its term [4]. First, there is an extremely large number of compactifications, which makes an exhaustive scan impossible. Second, a complete expression for the low-energy effective action is lacking because it requires knowing the CY metric. While existence has been proven by Yau, no metric has been constructed for compact CY until today (though recent progresses with machine learning have been achieved [5–10]). Third, topological properties (which largely constrain the general form of the

Machine Learning and the Physical Sciences workshop, NeurIPS 2022.

effective theory) can be difficult to compute with standard techniques of algebraic geometry, which lead to complicated algorithms, without closed-form solutions in general. For instance, computing cohomology group dimensions often requiring summing and subtracting huge integers at intermediate steps, to only get relatively small integers at the end.

Progress in string theory goes in hand with understanding better mathematical structures, and machine learning (ML) has been called to the rescue for all three problems [4, 11]. In this paper, we focus on computing Hodge numbers for complete intersection CY (CICY) manifolds [12–19]. This topic is interesting for fundamental mathematics and physics, as argued above, but also for the machine learning community as it showcases a new type of data which has not been properly studied until now. CICYs form a particularly appropriate set with which to begin our investigation since they correspond to the simplest category of CY, they have been completely classified, and their topological properties fully computed. As a consequence, this allows us to design and test new methods in a controlled environment before generalizing to new problems. Moreover, CICYs are often used in string theory model building due to their simple properties.

2 Complete intersection Calabi-Yau manifolds

CY N-folds are N-dimensional complex manifolds with $\mathrm{SU}(N)$ holonomy, or, equivalently, with a vanishing first Chern class [1]. They are characterized by their topological properties, such as the Hodge numbers. These features directly translate into properties of the 4-dimensional effective action, such as the number of chiral multiplets in heterotic compactifications, and the number of hyper and vector multiplets in type II compactifications. Ultimately, they are connected to the number of fermion generations, which could be used to test the effectiveness of the models.

The simplest CYs are constructed as *complete intersections* of hypersurfaces in a product of complex projective spaces $\mathbb{P}^{n_1} \times \cdots \times \mathbb{P}^{n_m}$. They are defined by systems of homogeneous polynomial equations, whose solutions identify CY manifolds. When considering topological properties, it is sufficient to keep track only of the dimensions of the projective spaces and the degree of the equations. In the general case of m projective spaces and k equations, a CICY X is represented by a *configuration matrix*, which will be the input to the network:

$$X = \left[\begin{array}{c|ccc} \mathbb{P}^{n_1} & \alpha_1^1 & \cdots & \alpha_k^1 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{P}^{n_m} & \alpha_1^m & \cdots & \alpha_k^m \end{array}\right], \qquad \begin{array}{l} n_i + 1 = \sum_{r=1}^{k} \alpha_r^i, \\ \dim_{\mathbb{C}} X = \sum_{i=1}^{m} n_i - k = N, \end{array} \qquad \alpha_r^i \in \mathbb{N}_+. \tag{1}$$

The Hodge numbers $h^{(p,q)}$ are topological invariants which count the dimensions of the Dolbeault cohomology groups, whose elements are complex (p, q)-forms.

We will investigate the CICY 3- and 4-folds. The numbers, matrix size, and statistics (written as $\langle\cdot\rangle = \text{average}_{\text{min}}^{\text{max}}$) of their Hodge numbers are:

- $N = 3$: 7890 CICYs with 15×18 matrices, $\langle h^{(1,1)}\rangle = 7.4_1^{19}$, $\langle h^{(2,1)}\rangle = 29_{15}^{101}$ [20, 21].
- $N = 4$: 921 497 CICYs with 16×20 matrices, $\langle h^{(1,1)}\rangle = 10_1^{24}$, $\langle h^{(2,1)}\rangle = 0.8_0^{33}$, $\langle h^{(3,1)}\rangle = 40_{20}^{426}$, $\langle h^{(2,2)}\rangle = 240_{204}^{1752}$ [22, 23].

3 Methodology

In this section, we present briefly the CICYMiner architecture and explain its main characteristics.

The paper [17] performed a complete data analysis for the 3-folds dataset, including feature engineering and selection. It was found that no derived feature helped in computing the Hodge numbers, and that neural networks outperform any other tested algorithm. As a consequence, we will focus on the latter, and review the architecture with the highest performance.

CICYMiner [18] is built from a series of Inception modules [16, 17], starting with a common trunk which splits in branches, such that it computes the different Hodge numbers at the same time given a configuration matrix $m \times k$. Inception modules are inspired by GoogleNet [24] and perform

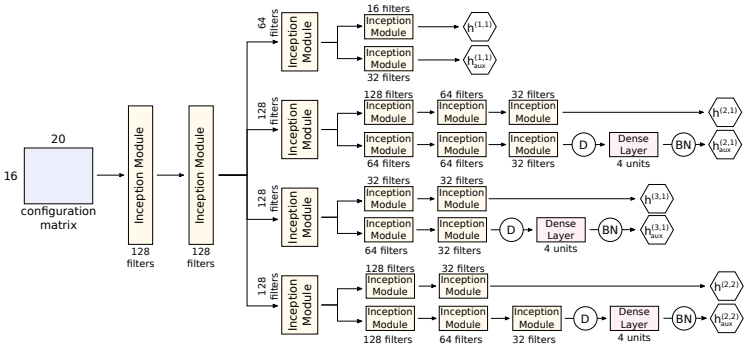


Figure 1: CICYMiner architecture, with dropout (D) with rate 0.2, and batch normalization (BN). The auxiliary branches replicate the output predictions and help to stabilize the learning process.

two parallel convolutions with $1d$ kernels $m \times 1$ and $1 \times k$, followed by concatenation and batch normalization. This structure is important 1) to put on an equal footing all the equations and projective spaces defining the configuration matrix (thanks to weight sharing), while also 2) taking into account the symmetries of the system. The ablation study performed in [16] displayed strikingly how both these ingredients are necessary to reach the highest accuracy. A neural network made of 3 Inception modules [16, 17] also achieved a near-perfect accuracy for the 3-folds $h^{(1,1)}$ while requiring much fewer parameters compared to earlier results [12, 13]. The architecture, inspired from [25], is shown in Figure 1 for the 4-folds; the network contains approximately 10^7 parameters. The same architecture can be used for the 3-folds which is achieved by removing the legs for $h^{(2,2)}$ and $h^{(3,1)}$ (3.3×10^6 parameters). Hyperparameter optimization is performed on the 4-folds dataset using a grid search over a reasonable portion of the hyperparameter space defined using insights from other architectures which used Bayesian or BOHB optimizations.

We preprocess the input data by simply rescaling the entries of the configuration matrices in the training set in the $[0, 1]$ range, but we don't rescale the labels. The computation is performed as a regression:[1] outputs are forced to be positive by adding a ReLU after each output layer, and predictions are rounded to the closest integer before computing the accuracy of the network. We train with the Huber loss with hyperparameter $\delta = 1.5$, and loss weights of 0.05, 0.3, 0.25 and 0.35 for $h^{(1,1)}$, $h^{(2,1)}$, $h^{(3,1)}$ and $h^{(2,2)}$, respectively, in the 4-folds case (determined using hyperparameter optimization as described above). In order to maintain a similar ratio, we use 0.17 and 0.83 for $h^{(1,1)}$ and $h^{(2,1)}$ for the CICY 3-folds. Performance of the network is assessed by computing the accuracy (agreement between the predicted and real integer values). Training has been performed on a single NVIDIA V100 GPU over a fixed amount of 300 epochs for CICY 4-folds, while we use 1500 epochs in the 3-folds case, due to limited cluster time. We use the Adam optimizer with an initial learning rate of 10^{-3} and a mini-batch size of 64 configuration matrices. The learning rate is reduced by a factor 0.3 after 75 epochs without improvement in the validation loss.

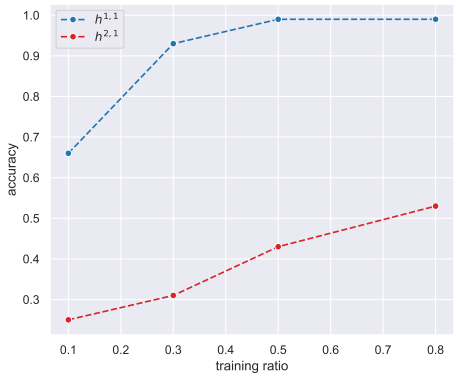
The code is available at: `https://github.com/thesfinox/ml-cicy-4folds`, `https://github.com/melsophos/cicy`.

4 Results

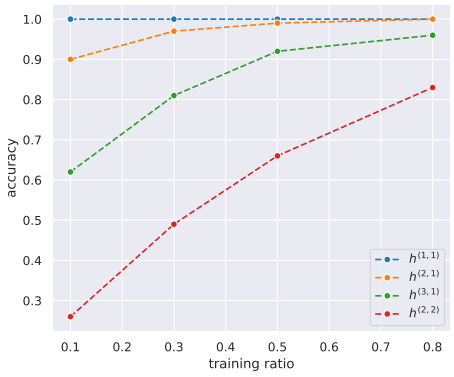
We provide three applications of the CICYMiner network: 1) Hodge numbers predictions, extrapolation after training only with 2) low and 3) high Hodge numbers. Except for the 4-folds predictions in 1), the results are new.

Predicting Hodge numbers We are interested in predicting the Hodge numbers, using the configuration matrix as input of the neural network. We will train with $x\%$ of the data selected by stratifying on $h^{(2,1)}$ in order to preserve the distribution of the samples, and varying x how the accuracy improves with the size of the training set. The validation set is chosen totally at random,

[1] Approaching the problem as a classification task does not make sense because it would assume an a priori knowledge of the value ranges, which is not desirable.



(a) 3-folds.



(b) 4-folds.

Figure 2: Accuracy for CICYMiner as a function of percentage of training data.

Table 1: Accuracy when training with an upper bound on $h^{(1,1)}$ or lower bound on $h^{(2,2)}$ (4-folds). As a reminder, extremal values for both Hodge numbers are $h^{(1,1)} \leq 24, h^{(2,2)} \geq 204$.

	ratio	$h^{(1,1)}$	$h^{(2,1)}$	$h^{(3,1)}$	$h^{(2,2)}$
$h^{(1,1)} \leq 5$	$2\,\%$	0.06	0.70	0.07	0.02
$h^{(1,1)} \leq 8$	$26\,\%$	0.53	0.83	0.27	0.11
$h^{(1,1)} \leq 10$	$59\,\%$	0.93	0.95	0.62	0.40
$h^{(2,2)} > 225$	$50\,\%$	1.00	0.88	0.39	0.04
$h^{(2,2)} > 250$	$27\,\%$	0.98	0.80	0.09	0.03
$h^{(2,2)} > 300$	$8\,\%$	0.95	0.72	0.05	0.03
$h^{(2,2)} > 400$	$1.6\,\%$	0.76	0.70	0.03	0.0

using $10\,\%$ of the samples. The remaining samples are included in the test set. The results for the 3- and 4-folds are given in Figures 2a and 2b.

For the 4-folds and $h^{(1,1)}$ of the 3-folds, the CICYMiner network is capable of learning accurately the discreteness of the Hodge numbers: for most of them, the regression metrics show values which indicate confidently integer numbers (MAE $\ll 0.50$ and MSE $\ll 0.25$). Note that the network is still underfitting the training set: longer training or different choices of the learning rate may be needed to investigate this aspect.

Extrapolation from low Hodge numbers We focus on the CICY 4-folds dataset. Following [14], we test the ability to predict the entire range of Hodge numbers when training only with samples whose $h^{(1,1)}$ is below a fixed value. Then, we test how well the network performs in predicting Hodge numbers outside the training range. The motivation is that, for other datasets, CYs with low $h^{(1,1)}$ are easier to understand, such that one can be interested in training for those instances and extrapolate the predictions. We reuse the same hyperparameters as before, but train CICYMiner on the reduced training set. Results are summarized in Table 1, where the ratio of the data refers to the size of the training set. As we can see, the performance increases quickly for the first two Hodge numbers, but remain lower than in the previous study for the last two.

Extrapolation from high Hodge numbers The same experiment can be run by imposing a lower bound on $h^{(2,2)}$ (largest number of outliers and largest interval of variation). As previously, no hyperparameter optimization is run in this case. Results are shown in Table 1. As in the previous case, good results on $h^{(1,1)}$ and $h^{(2,1)}$ can be recovered starting from approximately $50\,\%$ of the training samples, but $h^{(2,2)}$ performance stays low in any case. In this and previous cases, it is not clear why the network behaves this way.

5 Conclusion

This paper described the state-of-the-art in predicting CICY Hodge numbers with neural networks. This shows that machine learning techniques, and especially deep learning, are completely adapted to manipulate data in algebraic geometry. While the other topological quantities of the CICY are relatively easy to compute, it would still be interesting to explore if neural networks can predict them accurately. These different studies open the stage for attacking more difficult problems in algebraic geometry, such as studying CY built from the Kreuzer-Skarke polytopes [26, 27].

Computing accurately $h^{(2,1)}$ for the 3-folds is still an open problem. While the CICYMiner architecture performs slightly better than the pure Inception model from [16, 17], the drawback is a much slower training time and many more parameters. Moreover, while results are more precise for the 4-folds, it would be desirable to increase the accuracy of $h^{(2,2)}$. Work in progress explores if the graph representation of the data [1, 28, 29] can help.

Finally, the last step is extracting analytical formulas in order to interpret the neural network and improve our knowledge. The simplest approach is to perform symbolic regression on the neural network following [30, 31]. It can be expected that the result contains piecewise functions, given the structure of the analytic expressions found for line bundle cohomologies [32–36].

Broader impacts

Historically, codes used to compute properties of CY for string theory have not been made open source, which made the barrier to enter the field high. One goal of developing ML algorithms and make them open source is to help people get involved in the field.

Acknowledgments and Disclosure of Funding

This project has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No 891169. This work is supported by the National Science Foundation under Cooperative Agreement PHY-2019786 (The NSF AI Institute for Artificial Intelligence and Fundamental Interactions, `http://iaifi.org/`). This publication was made possible by the use of the *FactoryIA* supercomputer, financially supported by the *Île-De-France Regional Council*.

References

[1] T. Hübsch. *Calabi-Yau Manifolds: A Bestiary For Physicists*. World Scientific, Singapore, March 1992. ISBN 978-981-02-1927-7.

[2] Mariana Graña. Flux compactifications in string theory: A comprehensive review. *Physics Reports*, 423 (3):91–158, January 2006. ISSN 03701573. doi: 10.1016/j.physrep.2005.10.008.

[3] Luis E. Ibáñez and Angel M. Uranga. *String Theory and Particle Physics: An Introduction to String Phenomenology*. Cambridge University Press, Cambridge ; New York, 1st edition edition, March 2012. ISBN 978-0-521-51752-2.

[4] Fabian Ruehle. Data science applications to string theory. *Physics Reports*, 839:1–117, January 2020. ISSN 0370-1573. doi: 10.1016/j.physrep.2019.09.005.

[5] Anthony Ashmore, Yang-Hui He, and Burt Ovrut. Machine learning Calabi-Yau metrics. *Fortschritte der Physik*, 68(9):2000068, September 2020. ISSN 0015-8208, 1521-3978. doi: 10.1002/prop.202000068.

[6] Michael R. Douglas, Subramanian Lakshminarasimhan, and Yidi Qi. Numerical Calabi-Yau metrics from holomorphic networks. *arXiv:2012.04797 [hep-th, physics:physics]*, May 2021. URL `http://arxiv.org/abs/2012.04797`.

[7] Vishnu Jejjala, Damian Kaloni Mayorga Pena, and Challenger Mishra. Neural Network Approximations for Calabi-Yau Metrics. *arXiv:2012.15821 [hep-th]*, January 2021. URL `http://arxiv.org/abs/2012.15821`.

[8] Lara B. Anderson, Mathis Gerdes, James Gray, Sven Krippendorf, Nikhil Raghuram, and Fabian Ruehle. Moduli-dependent Calabi-Yau and SU(3)-structure metrics from Machine Learning. May 2021. URL http://arxiv.org/abs/2012.04656.

[9] Magdalena Larfors, Andre Lukas, Fabian Ruehle, and Robin Schneider. Learning Size and Shape of Calabi-Yau Spaces. *arXiv:2111.01436*, November 2021. URL http://arxiv.org/abs/2111.01436.

[10] Magdalena Larfors, Andre Lukas, Fabian Ruehle, and Robin Schneider. Numerical Metrics for Complete Intersection and Kreuzer-Skarke Calabi-Yau Manifolds. May 2022. doi: 10.48550/arXiv.2205.13408.

[11] Yang-Hui He. *The Calabi–Yau Landscape: From Geometry, to Physics, to Machine Learning*. Springer, 1st edition, August 2021. ISBN 978-3-030-77561-2. URL http://arxiv.org/abs/1812.02893.

[12] Yang-Hui He. Machine-learning the string landscape. *Physics Letters B*, 774:564–568, November 2017. ISSN 0370-2693. doi: 10.1016/j.physletb.2017.10.024.

[13] Kieran Bull, Yang-Hui He, Vishnu Jejjala, and Challenger Mishra. Machine Learning CICY Threefolds. *Physics Letters B*, 785:65–72, October 2018. ISSN 03702693. doi: 10.1016/j.physletb.2018.08.008.

[14] Kieran Bull, Yang-Hui He, Vishnu Jejjala, and Challenger Mishra. Getting CICY High. *Physics Letters B*, 795:700–706, August 2019. ISSN 03702693. doi: 10.1016/j.physletb.2019.06.067.

[15] Yang-Hui He and Andre Lukas. Machine Learning Calabi-Yau Four-folds. *Physics Letters B*, 815:136139, April 2021. ISSN 03702693. doi: 10.1016/j.physletb.2021.136139.

[16] Harold Erbin and Riccardo Finotello. Inception Neural Network for Complete Intersection Calabi-Yau 3-folds. *Machine Learning: Science and Technology*, 2(2):02LT03, March 2021. ISSN 2632-2153. doi: 10.1088/2632-2153/abda61.

[17] Harold Erbin and Riccardo Finotello. Machine learning for complete intersection Calabi-Yau manifolds: A methodological study. *Physical Review D*, 103(12):126014, June 2021. ISSN 2470-0010, 2470-0029. doi: 10.1103/PhysRevD.103.126014.

[18] Harold Erbin, Riccardo Finotello, Robin Schneider, and Mohamed Tamaazousti. Deep multi-task mining Calabi-Yau four-folds. *Machine Learning: Science and Technology*, 3(1):015006, November 2021. ISSN 2632-2153. doi: 10.1088/2632-2153/ac37f7.

[19] Harold Erbin and Riccardo Finotello. Deep learning complete intersection Calabi-Yau manifolds. In Yang-Hui He, editor, *Machine-Learning in Theoretical Physics & Pure Mathematics*. World Scientific, 2023.

[20] P. Candelas, A. M. Dale, C. A. Lütken, and R. Schimmrigk. Complete intersection Calabi-Yau manifolds. *Nuclear Physics B*, 298(3):493–525, March 1988. ISSN 0550-3213. doi: 10.1016/0550-3213(88)90352-5.

[21] P. S. Green, T. Hübsch, and C. A. Lütken. All the Hodge numbers for all Calabi-Yau complete intersections. *Classical and Quantum Gravity*, 6(2):105, 1989. ISSN 0264-9381. doi: 10.1088/0264-9381/6/2/006.

[22] James Gray, Alexander S. Haupt, and Andre Lukas. All Complete Intersection Calabi-Yau Four-Folds. *Journal of High Energy Physics*, 2013(7), July 2013. ISSN 1029-8479. doi: 10.1007/JHEP07(2013)070.

[23] James Gray, Alexander S. Haupt, and Andre Lukas. Topological Invariants and Fibration Structure of Complete Intersection Calabi-Yau Four-Folds. *Journal of High Energy Physics*, 2014(9), September 2014. ISSN 1029-8479. doi: 10.1007/JHEP09(2014)093.

[24] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, June 2015. doi: 10.1109/CVPR.2015.7298594.

[25] Abdallah Benzine, Mohamed El Amine Seddik, and Julien Desmarais. Deep Miner: A Deep and Multi-branch Network which Mines Rich and Diverse Features for Person Re-identification. *arXiv:2102.09321 [cs]*, February 2021. URL http://arxiv.org/abs/2102.09321.

[26] Maximilian Kreuzer and Harald Skarke. Complete classification of reflexive polyhedra in four dimensions. *Advances in Theoretical and Mathematical Physics*, 4(6):1209–1230, 2000. doi: 10.4310/ATMP.2000.v4.n6.a2.

[27] Mehmet Demirtas, Liam McAllister, and Andres Rios-Tascon. Bounding the Kreuzer-Skarke Landscape. *arXiv:2008.01730 [hep-th]*, August 2020. URL http://arxiv.org/abs/2008.01730.

[28] Paul Green and Tristan Hübsch. Polynomial deformations and cohomology of Calabi-Yau manifolds. *Communications in Mathematical Physics*, 113(3):505–528, September 1987. ISSN 1432-0916. doi: 10.1007/BF01221257.

[29] Sven Krippendorf and Marc Syvaeri. Detecting Symmetries with Neural Networks. *Machine Learning: Science and Technology*, 2(1):015010, December 2020. doi: 10.1088/2632-2153/abbd2d.

[30] Miles D. Cranmer, Rui Xu, Peter Battaglia, and Shirley Ho. Learning Symbolic Physics with Graph Networks. *arXiv:1909.05862 [astro-ph, physics:physics, stat]*, November 2019. URL `http://arxiv.org/abs/1909.05862`.

[31] Miles Cranmer, Alvaro Sanchez-Gonzalez, Peter Battaglia, Rui Xu, Kyle Cranmer, David Spergel, and Shirley Ho. Discovering Symbolic Models from Deep Learning with Inductive Biases. *arXiv:2006.11287 [astro-ph, physics:physics, stat]*, November 2020. URL `http://arxiv.org/abs/2006.11287`.

[32] Andrei Constantin and Andre Lukas. Formulae for Line Bundle Cohomology on Calabi-Yau Threefolds. *Fortschritte der Physik*, 67(12):1900084, December 2019. ISSN 0015-8208, 1521-3978. doi: 10.1002/prop.201900084.

[33] Magdalena Larfors and Robin Schneider. Line bundle cohomologies on CICYs with Picard number two. *Fortschritte der Physik*, 67(12):1900083, December 2019. ISSN 0015-8208, 1521-3978. doi: 10.1002/prop.201900083.

[34] Daniel Klaewer and Lorenz Schlechter. Machine Learning Line Bundle Cohomologies of Hypersurfaces in Toric Varieties. *Physics Letters B*, 789:438–443, February 2019. ISSN 03702693. doi: 10.1016/j.physletb.2019.01.002.

[35] Callum R. Brodie, Andrei Constantin, Rehan Deen, and Andre Lukas. Index Formulae for Line Bundle Cohomology on Complex Surfaces. *Fortschritte der Physik*, 68(2):1900086, February 2020. ISSN 0015-8208, 1521-3978. doi: 10.1002/prop.201900086.

[36] Callum R. Brodie, Andrei Constantin, Rehan Deen, and Andre Lukas. Machine Learning Line Bundle Cohomology. *Fortschritte der Physik*, 68(1):1900087, January 2020. ISSN 0015-8208, 1521-3978. doi: 10.1002/prop.201900087.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [N/A]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Sections 3 and 4.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] Because of computational cost.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See end of Section 3.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]

(b) Did you mention the license of the assets? [N/A] The data we used was not licensed by the authors.

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]