Calabi–Yau Metrics, Energy Functionals and Machine-Learning

Anthony Ashmore,1,2,∗ Lucille Calmon,3,4,† Yang-Hui He,5,6,7,8,† and Burt A. Ovrut9,§

1Enrico Fermi Institute & Kadanoff Center for Theoretical Physics, University of Chicago, IL 60637, USA
2Sorbonne Université et Kadanoff Center for Theoretical Physics, University of Chicago, F-75005 Paris, France
3Department of Physics, Imperial College London, Prince Consort Road, London, SW7 2AZ, UK
4Department of Mathematics, City, University of London, EC1V0HB, UK
5Department of Physics, Imperial College London, Prince Consort Road, London, SW7 2AZ, UK
6Department of Mathematics, City, University of London, EC1V0HB, UK
7Merton College, University of Oxford, OX1 4JD, UK
8School of Physics, NanKai University, Tianjin, 300071, P.R. China
9Department of Physics, University of Pennsylvania, Philadelphia, PA 19104, USA

We apply machine learning to the problem of finding numerical Calabi–Yau metrics. We extend previous work on learning approximate Ricci-flat metrics calculated using Donaldson’s algorithm to the much more accurate “optimal” metrics of Headrick and Nassar. We show that machine learning is able to predict the Kähler potential of a Calabi–Yau metric having seen only a small sample of training data.

I. INTRODUCTION & SUMMARY

Recent decades have seen great progress in trying to match string theory to experiment. There are now large classes of top-down string models with the potential to reproduce the Standard Model at low energies. Perhaps the most promising of these approaches is also the oldest, namely compactifying the $E_8 \times E_8$ heterotic string on a six-dimensional Calabi–Yau manifold (or on a threefold times an interval in heterotic M-theory [1–5]). Within this framework, there are vast numbers of models which give minimally supersymmetric extensions of the Standard Model, including three generations, the proper Higgs structure, and the correct gauge group [6–18].

Unfortunately, in nearly all of these models, we are currently unable to make sharp predictions for the couplings and masses that govern the resulting four-dimensional effective physics. This can be traced to our ignorance of the explicit Calabi–Yau metric on the compactification manifold (and an explicit hermitian Yang–Mills connection for the gauge fields). Without this metric, we are generally unable to compute the superpotential or the Kähler potential of the effective theories, nor fix a supersymmetry-breaking mechanism, and hence unable to compare predictions with experiment.

The recent years have seen a flurry of activity in tackling the problem of finding Calabi–Yau metrics by turning to numerical methods and, most recently, machine learning [19–23] (see [24–27] for some uses of ML in pure mathematics). There are now a variety of methods for computing these metrics numerically, including position space methods [28], spectral methods [29–33], and machine learning algorithms [34–39].

This note completes on the work started in [34]. There, three of the present authors attempted to understand the extent to which machine-learning techniques could be used to speed up or improve the accuracy of numerically calculated Calabi–Yau metrics. In that work, the focus was on Ricci-flat metrics computed using Donaldson’s balanced metrics [30]. A combination of point-wise extrapolation and gradient-boosted decision trees was used to improve the accuracy of these numerical metrics and to reduce the total time taken to calculate them. Here, we extend and complete this work by showing that a neural network can also learn the much more accurate “optimal” metrics, first discussed by Headrick and Nassar [33]. These metrics are exponentially more accurate than those found via balanced metrics.

For concreteness, we focus on perhaps the most famous Calabi–Yau manifold: the Fermat quintic. This threefold admits a large discrete symmetry group, $S_5 \times (Z_5)^4$, of order 150,000, which can be used to reduce the basis of polynomials and hence the space of Kähler potentials that are minimised over. We use a five-layer neural network to encode the Kähler potential of the approximate Ricci-flat metric. The network is trained via supervised learning, with inputs given by points on the Calabi–Yau hypersurface and the values of certain sections, and the outputs given by the numerical value of the exponential of the Kähler potential at those points. The outputs are calculated using the Mathematica package fermat.m available at [40].

We find that this neural network can mimic the behaviour of the Kähler potential to high accuracy, having been trained on approximately 2000 input-output pairs. We show that the performance of the network is robust by calculating a ten-run averaged loss measure and showing that the accuracy does not vary significantly between each run. We also examine how the accuracy of the network varies as a function of training set size, showing that the mean absolute percentage error between the network outputs and the target Kähler potential decreases as the training fraction increases.

II. BACKGROUND AND NOTATION

For the purposes of this work, a Calabi–Yau (CY) manifold $X$ is a compact, Kähler manifold which admits a Ricci-flat metric. Thanks to Yau’s proof [41] of the Calabi
We now review some of the important steps. \(\psi\) where

\[
Q = z_0^5 + z_1^5 + z_2^5 + z_3^5 + z_4^5 = 0,
\]

where the \(z_i, i = 0, \ldots, 4\) are homogeneous coordinates on projective space. Since a Calabi–Yau manifold is Kähler and the holomorphic (3, 0)-form is determined exactly by \(Q\) via a residue theorem, the problem of finding the Ricci-flat metric reduces to finding a suitable Kähler potential, \(\mathcal{K}\).

### II.1. The energy functional method

The energy functional method was developed by Headrick and Nassar in [33] with the aim of computing numerical Ricci-flat metrics on Calabi–Yau manifolds given by hypersurfaces in projective space (or products thereof).\(^1\) We now review some of the important steps.

First, one parametrises the space of Kähler potentials using the “algebraic metrics” ansatz of Tian [29] and Donaldson [30]. In our case, \(\mathcal{K}\) is written as an expansion in the eigenfunctions of the Laplacian on \(\mathbb{P}^4\). This provides a controlled expansion in terms of a parameter \(k\) when one includes eigenfunctions for the first \(k + 1\) eigenspaces. Explicitly, the first \(k\) eigenspaces of the Laplacian are spanned by

\[
\frac{s_a \bar{s}_\beta}{(\delta^{ij} z_i \bar{z}_j)^k},
\]

where the \(s_a(z)\) are homogeneous polynomials of degree \(k\) in the \(z_i\), and \(\delta^{ij}\) is the hermitian form which gives the Fubini–Study metric on \(\mathbb{P}^4\).

Since \(Q = 0\) on the hypersurface, one should quotient the set of homogeneous polynomials by the ideal generated by \(Q\). This gives a reduced basis, which we denote by \(p_A\). Any Kähler metric with potential \(\mathcal{K}\) in the same Kähler class as the Fubini–Study metric differs from \(\mathcal{K}_{FS}\) by a globally defined function. Introducing coordinates \(u\) on the patch \(O(c) \subset X\) (defined by \(z_c \neq 0\)), one can expand the Kähler potential in the reduced \(p_A\) basis as

\[
\mathcal{K} = \frac{1}{k} \ln \left(h^{AB} p_A \bar{p}_B\right) = \frac{1}{k} \ln \psi,
\]

where \(\psi = h^{AB} p_A \bar{p}_B\) and the explicit form of \(\mathcal{K}_{FS}\) was used. The resulting metrics, which depend on the parameters \(h^{AB}\), are known as “algebraic metrics”.

If the zero locus of \(Q\) is invariant under the action of a group, the Ricci-flat Kähler potential and the metric itself must also be invariant, and so the group gives an isometry. In practice, this means we can restrict the combinations of \(p_A \bar{p}_B\) that appear in \(\mathcal{K}\) to be those invariant under the isometry group. The Fermat quintic equation (1) admits a discrete \(S_5 \times (\mathbb{Z}_3)^5\) symmetry, which can be thought of as permutations of the \(z_i\) combined with phase rotations.\(^2\)

This prompts the definition of a new basis spanned by polynomials of \(u\) and \(v\):

\[
\mathcal{P}_l = c^{IJ}_l \rho_l \bar{\rho}_J,
\]

where the \(\rho_l\) are, like the \(p_A\), functions of \(u\). Here, the coefficients \(c^{IJ}_l\) pick out the polynomials that are linearly independent on the hypersurface \(Q = 0\) and invariant under the discrete isometry group. In this basis, the function \(\psi\) is simply \(\psi = h^l \mathcal{P}_l\), with the resulting Kähler potential \(\mathcal{K}[h]\) given as a function of \(h^l\). The coefficients \(h^l\) are the adjustable parameters that one can tweak to find the best approximation to the Ricci-flat metric for a choice of degree \(k\).

In [33], the problem of finding an approximate Ricci-flat metric on a Calabi–Yau manifold was rephrased as the minimisation of an appropriate functional. Recall that there are two natural volume forms on a Calabi–Yau threefold, defined by the holomorphic (3, 0)-form and the Kähler metric:

\[
\text{vol}_\Omega = i \Omega \wedge \bar{\Omega}, \quad \text{vol}_\omega = \omega \wedge \bar{\omega} \wedge \omega.
\]

The first, \(\text{vol}_\Omega\), depends only on the defining equation \(Q = 0\), while the second depends on the explicit choice of Kähler potential. Defining the ratio of these as

\[
v_\omega = \frac{\text{vol}_\omega}{\text{vol}_\Omega},
\]

the Ricci tensor of the Kähler metric is given by

\[
\mathcal{R}_{ab} = -\partial_a \bar{\partial}_b \ln v_\omega.
\]

Yau’s theorem then ensures that there is a unique choice of \(\omega\) for which the Ricci tensor vanishes, with the Ricci-flat metric lying in the same Kähler class as the original Fubini–Study metric. From above, the vanishing of the Ricci tensor is equivalent to \(v_\omega = 1\) constant. Without loss of generality, the coefficients in (5) can be chosen so that \(v_\omega = 1\) for the Ricci-flat representative.

The functional chosen in [33] was

\[
E[\omega] = \int_X \text{vol}_\Omega (1 - v_\omega)^2.
\]

Importantly, \(E[\omega]\) is non-negative, has a unique minimum on the Ricci-flat metric, and has no other critical points. Thinking of the functional as \(E[h]\), i.e. a functional of

\(^1\) Further details can be found in their unpublished notes available at [40].

\(^2\) A discussion of the generators of this group can be found in [31, 45, 44].
the parameters of the underlying Kähler potential, an approximate Ricci-flat Kähler potential can be found by minimising $E[h]$ over the parameters $h^i$.

The ratio $v_\omega$ can be rewritten to make its dependence on the parameters $h^i$ explicit and allow efficient numerical computation. Upon defining the four-component object

$$Q_i = \frac{\partial Q}{\partial u^i},$$

where $\partial$ is taken on $\mathbb{P}^4$, the $(3, 0)$-form $\Omega$ can be written as

$$\Omega = Q_\delta^{-1} \prod_{i \neq \delta} du^i,$$

where the coordinates on $X$ are taken as $u^i$ for $i \neq \delta$. The volume form defined by $\Omega$ is then given by

$$\text{vol}_\Omega = (-i)^3 |Q_\delta|^{-2} \prod_{i \neq \delta} du^i \wedge \prod_{j \neq \delta} \bar{d}u^j,$$

where $|Q|^2 \equiv \hat{g}^{ij} Q_i \bar{Q}_j$. Consequently, the ratio $v_\omega$ can be written as

$$v_\omega = |Q|^2 \det \hat{g}^{\bar{\delta} \bar{\delta}}.$$

This is now expressed fully in terms of quantities computed on $\mathbb{P}^4$. The determinant of the inverse metric $g_{ab}$ on $X$, defined by $K[h]$, is then related to the metric $g_{ij}$ on $\mathbb{P}^4$ as

$$\det g_{ab} = \frac{|Q|^2}{|Q_\delta|^2} \det \hat{g}_{ij},$$

where $|Q|^2 \equiv \hat{g}^{ij} Q_i \bar{Q}_j$. This yields

$$v_\omega = k^{-5} \psi^{-4} (\bar{Q}_\beta \Psi^{\bar{\alpha} \bar{\beta}} Q_\alpha) \det \Psi_{\alpha \bar{\beta}},$$

where the indices run over $\alpha = (i, c)$, and we have defined

$$\Psi_{\alpha \bar{\beta}} = h^i (Q_i)_{\alpha \beta},$$

with

$$Q'_\alpha = \rho^i, \quad Q'_i = \hat{\partial}_i \rho^i, \quad (Q_i)_{\alpha \beta} = c^l_{I J} q^{l}_{i} \bar{q}^{J}_{\beta},$$

and $Q_\alpha = (Q_i, 0)$. This form of $v_\omega$ can be calculated at randomly sampled points on $X$, inserted into (8), and then integrated. One then minimises $E[h]$ over the parameters $h^i$ in order to find the best approximation to the honest Ricci-flat metric. In summary, the algorithm consists of the following steps:

1. Calculate the basis. For a given $k$, find the basis of invariant polynomials $P_l$.

2. Generate points. Generate random points on $\mathbb{P}^4$ distributed according to vol$_\Omega$. This yields $N_p$ tuples of four homogeneous coordinates $u^i$ that label the points.

3. Calculate data. Calculate $Q_i$, $Q'_\alpha \bar{\beta}$ and the matrix $\Psi_{\alpha \bar{\beta}}$ at each point. Using (14), compute the value of $v_\omega$ at each point.

4. Integrate. Compute $E[h]$ by summing the point-wise values of $(1 - v_\omega)^2$.

5. Minimise. Minimise $E[h]$ as a function of $h^i$. The values of $h^i$ obtained define the “best” approximate Ricci-flat metric for the chosen value of $k$.

This algorithm was implemented in Mathematica and available in the Fermat package at [40]. The resulting metrics have come to be known as “optimal metrics”, since they give the best possible approximation to the exact Ricci-flat metric within the family of algebraic metrics. As investigated in detail in [33], the accuracy of these metrics scales exponentially with $k$, rather than as a polynomial of $k$ as is the case for Donaldson’s balanced metric approach.

In phenomenologically realistic models, the relevant Calabi–Yau spaces, such as the Schoen manifold used in [8], generally admit much smaller discrete symmetry groups. The basis of invariant polynomials is then much larger, leading to a minimisation problem for a very large number of parameters.

### II.2. Data set

Our data set $D = \{ u^i, Q_i, Q'_\alpha \bar{\beta} \rightarrow \psi \}$ consists of a set of 5000 inputs and outputs calculated using [40]. The inputs are given by the point-wise quantities that enter the calculation of the functional $E[h]$. The outputs are given by the point-wise values of $\psi$ that correspond to an “optimal” approximate Ricci-flat metric. The set $D$ is then split into a training set $T$ and a validation set $V$, with the fraction of $D$ used for the training set denoted by $\gamma \in [0, 1]$.

---

3 Note that for arbitrary Kähler potentials, $E[K]$ does not have a unique minimum since Kähler transformations change $K$ but leave $\omega$ (and hence $E[K]$) unchanged. For Kähler potentials defined via $\psi = h^i P_l$, changing the $h^i$ never corresponds to a Kähler transformation, and so $E[h]$ does have a unique minimum.

4 The $u^i$ give four coordinates on $\mathbb{P}^4$. One of these, $i = \delta$, is singled out so that $u^\delta$ is defined implicitly by $Q = 0$, with the remaining $u^i$, $i \neq \delta$, giving three coordinates on the hypersurface.

5 Points on $X$ should be sampled according to the exact Calabi–Yau measure vol$_\Omega$. In practice, this can be achieved either by rejection sampling [33] or by sampling according to a known, auxiliary distribution and then weighting the points appropriately to recover vol$_\Omega$ [31].
The outputs are calculated using the \( k = 10 \) optimal metric. At \( k = 10 \), there are 38 invariant polynomials \( P_l \) for the Fermat quintic. For each sample of the data set, \( D \) contains 958 complex input values (or 1916 real input values) and 1 real output. The neural network takes the input and produces a real number, which can then be compared with the target value of \( \psi \). In what follows, the target values of \( \psi \) corresponding to the optimal metric are denoted by \( \hat{\psi} \), while the values computed by the neural network are labelled \( \hat{\psi} \).

II.3. Neural network

In order to predict point-wise values of \( \hat{\psi} \) from \( \{u', Q_i, Q_{\alpha\beta}'\} \), we use a five-layer neural network with widths (1916, 150, 70, 10, 1) implemented in Python using the TensorFlow library [45]. At each node of the network, the weights and biases together with the activation functions of each layer determine the flow of information through the network. We use the Rectified Linear Unit (ReLU) activation function for all layers but the last one. The network then essentially acts as a function, mapping 1916 arguments onto the reals. The power and flexibility of the network comes from the non-linearity of the ReLu activation function, which allows it to capture the complex relation between the input and output data; in our case an integration over many points and the minimisation of a functional. In addition, we found a 0.01 drop-out rate in the first hidden layer was useful to ensure the network learned global structures instead of the specific features of the training data (which might not generalise to unseen inputs). This randomly deactivates nodes during each pass over the data and helps to avoid overfitting, a phenomenon where the network’s performance improves with more training data but underperforms when tested on validation data.

During the training phase, the internal parameters of the network (the weights and biases of each node) are adjusted to minimise a loss function, which we chose as the mean absolute percentage error (MAPE), given by

\[
\text{MAPE} = 100 \cdot \frac{1}{n} \sum_{p=1}^{n} \frac{|\psi - \hat{\psi}|_p}{|\psi|_p},
\]

where \( n \) is the number of points considered in each training round, and the index \( p \) on \( \psi|_p \) indicates the value of \( \psi \) at the point \( p \).

The training proceeds as follows: given a batch of samples drawn randomly from the training set \( T \), the predicted outputs \( \hat{\psi} \) are computed by the network and compared to target values of \( \psi \) using the MAPE. The parameters of the network are then adjusted via gradient descent to reduce the MAPE for all points in the batch. This is repeated with new batches until all data in \( T \) has been presented to the network. This forms one “epoch”. The network is then trained over multiple epochs, adjusting its parameters each time until the MAPE is reduced to an acceptable value. We trained our network for 100 epochs, with batches of 70 points.

III. MACHINE LEARNING THE KÄHLER POTENTIAL

III.1. A single training

Training the network with \( \gamma = 0.7 \) (i.e. 3500 points for training and 2500 for validation), we calculate the value of the loss function (MAPE) after each epoch and plot this in Fig. 2 for both the training and validation sets. The loss clearly decreases as training proceeds. After 100 epochs, the discrepancy between the predictions of the network, \( \psi \), and the values corresponding to the optimal \( k = 10 \) metric, \( \hat{\psi} \), drops to 1.6%, showing that the network can indeed learn an accurate representation of the approximately Ricci-flat Kähler potential.

Curiously, the loss on the training set remains 1% above the loss on the validation set. This behaviour can be traced to the 0.01 drop-out rate, where, on average, about 1.5 nodes are removed during each pass. This highlights...
the subtlety of neural networks, where the removal of a single node can significantly alter or improve performance. This result confirms that our neural network is not overfitting to features in the training set and can extrapolate well to unseen data.

Using this trained neural network on a new set of 10000 input and output samples (akin to an unseen validation set), one can obtain $\psi$ from the trained neural network for these 10000 points in roughly one second. In Fig. 3, we present the predicted outputs $\hat{\psi}$ for these 10000 points against the corresponding values of $\psi$ calculated using the optimal $k = 10$ metric. Using a linear fit, we find a slope of $1.012 \pm 0.002$, confirming the high accuracy of the predicted values of $\hat{\psi}$. This single training helps to verify that our architecture does not overfit local features of the training set.

III.2. Training over multiple $\gamma$

In order to assess how the performance of the network is affected by the size of the training set, we compute the learning curve (the value of the MAPE) for varying training set fractions, $\gamma$. For each $\gamma$, the network is trained from scratch using the dataset $D$ described above. The learning curve, averaged over ten independent realisations of the network, is shown in Fig. 4 together with the corresponding standard error on the mean.

The mean value of the loss (itself a mean percentage error) clearly decreases as the amount of data seen during training increases, i.e. the network is learning from the data given. The error on the averaged loss decreases as $\gamma$ increases, showing that the network is learning reliably from one training to the next. Specifically, when learning from $\gamma = 0.4$ of the data (i.e. 2000 points), the MAPE is already below 3%. It reduces to $2.5\% \pm 0.1\%$ at $\gamma = 0.46$ (i.e. having seen 2300 points). For $\gamma = 0.78$ (3000 points), the error drops to $1.5\% \pm 0.07\%$. This demonstrates that our relatively simple network architecture is able to learn significant features of our data set, encoding the optimal $k = 10$ metric to an excellent accuracy that varies negligibly between different trainings. To verify this, we compute the ten-run average of the R-squared coefficient,

$$ R^2(\psi, \hat{\psi}) \equiv 1 - \frac{\sum_p (\psi_p - \hat{\psi}_p)^2}{\sum_p (\psi_p - \overline{\psi_p})^2}, \quad (18) $$

where $\overline{\psi_p}$ is the statistical mean of all $\psi_p$ [24]. This is plotted in Fig. 5 for the validation set for each choice of training fraction, $\gamma$. The $R^2$ coefficient reaches 1 around $\gamma = 0.4$, further confirming that the network is able to learn significant underlying features of the data with only 2000 points. The exact values are $R^2 = 0.996 \pm 0.001$ for $\gamma = 0.46$ and $R^2 = 0.998 \pm 0.0003$ for $\gamma = 0.78$, again confirming the high accuracy of the results produced with the neural network.

In summary, the error on the ten-run averaged MAPE and $R^2$ coefficients shown in Figs. 4 and 5 decreases quickly with increasing $\gamma$: predictions are poor for $\gamma \lesssim 0.2$, where the standard error for ten-run averages are largest; after stabilising at $\gamma \gtrsim 0.2$, the values of $\psi$ produced are of approximately constant accuracy.

On a personal laptop, generating the data set $D$ using the fermat.m package took 16 minutes, of which 10 minutes were needed to calculate the input data and 6 minutes to perform the minimisation and obtain the values of $\psi$. In comparison, our neural network architecture completes...
training in under 2 minutes and predicts values of $\hat{\psi}$ with approximately 1.5% error in under a second. The network is capable of learning the features underlying the relationship between $\psi$ and the input data with great accuracy. In particular, it predicts $\hat{\psi}$ with less than 2.5% error after having seen only 2300 data points. Training the network on 3500 points (with error around 1.5%) takes 2 minutes and obtaining the predictions for 10000 additional points takes under 2 seconds. This is a considerable gain from having seen only 2300 data points. Training the network and obtaining the predictions for 10000 additional points on 3500 points (with error around 1.5%) takes under 2 minutes and predicts values of $\hat{\psi}$ with $\psi$ and the minimisation. The simple network structure detailed in Section II.3 is thus capable of reproducing the relationship between $\psi$ with the minimum. The simple network structure $\{w^i, Q_i, Q^i_{\alpha\bar{j}}\}$, i.e. the coordinates and invariant sections, and $\psi$, i.e. the exponential of the Kähler potential.

ACKNOWLEDGMENTS

AA is supported by the European Union’s Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No. 838776. LC thanks Adam Calabrese for useful discussions. YH would like to thank STFC for grant ST/J00037X/1. BAO is supported in part by both the research grant DOE No. DESC0007901 and SAS Account 020-0188-2-010202-6603-0338.
Learning meets Number Theory: The Data Science of Birch-Swinnerton-Dyer”, arXiv:1911.02008 [math.NT].

[27] Y.-H. He, “Machine-Learning Mathematical Structures”, arXiv:2101.06317 [cs.LG].

[28] M. Headrick and T. Wiseman, “Numerical Ricci-flat metrics on K3”, Class. Quant. Grav. 22 (2005)4931–4960, arXiv:hep-th/0506129.

[29] G. Tian, “On a set of polarized Kähler metrics on algebraic manifolds”, Journal of Differential Geometry 32 1, (1990)99 – 130.

[30] S. K. Donaldson, “Some numerical results in complex differential geometry”, arXiv:math/0512625 [math.DG].

[31] M. R. Douglas, R. L. Karp, S. Lukic, and R. Reinbacher, “Numerical Calabi-Yau metrics”, J. Math. Phys. 49 (2008)032302, arXiv:hep-th/0612075.

[32] V. Braun, T. Brelidze, M. R. Douglas, and B. A. Ovrut, “Calabi-Yau Metrics for Quotients and Complete Intersections”, JHEP 05 (2008)080, arXiv:0712.3563 [hep-th].

[33] M. Headrick and A. Nassar, “Energy functionals for Calabi-Yau metrics”, Adv. Theor. Math. Phys. 17 5, (2013)867–902, arXiv:0908.2635 [hep-th].

[34] A. Ashmore, Y.-H. He, and B. A. Ovrut, “Machine Learning Calabi–Yau Metrics”. Fortsch. Phys. 68 9, (2020)2000068, arXiv:1910.08605 [hep-th].

[35] L. B. Anderson, M. Gerdes, J. Gray, S. Krippendorf, N. Raghuram, and F. Ruehle, “Moduli-dependent Calabi-Yau and SU(3)-structure metrics from Machine Learning”, JHEP 05 (2021)013, arXiv:2012.04656 [hep-th].

[36] M. R. Douglas, S. Lakshminarasimhan, and Y. Qi, “Numerical Calabi-Yau metrics from holomorphic networks”, arXiv:2012.04797 [hep-th].

[37] V. Jejjala, D. K. Mayorga Pena, and C. Mishra, “Neural Network Approximations for Calabi-Yau Metrics”, arXiv:2012.15821 [hep-th].

[38] M. R. Douglas, “Holomorphic feedforward networks”, arXiv:2105.03991 [math.CV].

[39] M. Larfors, A. Lukas, F. Ruehle, and R. Schneider, “Learning Size and Shape of Calabi-Yau Spaces”, arXiv:2111.01436 [hep-th].

[40] M. Headrick and A. Nassar, fermat.m, 2009. https://people.brandeis.edu/~headrick/Mathematica/index.html.

[41] S. T. Yau, “On the Ricci curvature of a compact Kähler manifold and the complex Monge-Ampère equation.1.”, Commun. Pure Appl. Math. 31 3, (1978)339–411. http://cds.cern.ch/record/420951.

[42] E. Calabi and K. On, “On kähler manifolds with vanishing canonical class”, Princeton Mathematical Series 12 (1957)78–89.

[43] V. Braun, T. Brelidze, M. R. Douglas, and B. A. Ovrut, “Eigenvalues and Eigenfunctions of the Scalar Laplace Operator on Calabi-Yau Manifolds”, JHEP 07 (2008)120, arXiv:0805.3689 [hep-th].

[44] A. Ashmore and F. Ruehle, “Moduli-dependent KK towers and the swampland distance conjecture on the quintic Calabi-Yau manifold”, Phys. Rev. D 103 10, (2021)106028, arXiv:2103.07472 [hep-th].

[45] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems”, 2015. http://tensorflow.org/. Software available from tensorflow.org.