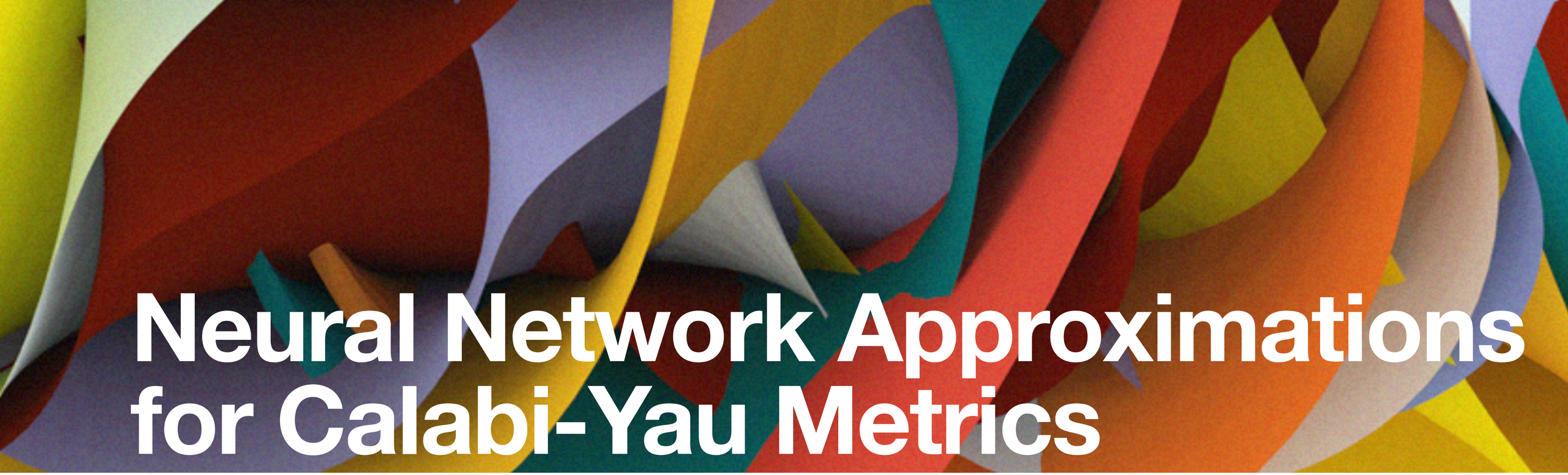


Neural Network Approximations for Calabi-Yau Metrics

Damián Mayorga Peña

Based on arXiv:2012.15821, in collaboration with Vishnu Jejjala and Challenger Mishra.

String Pheno Seminars 02.03.2021

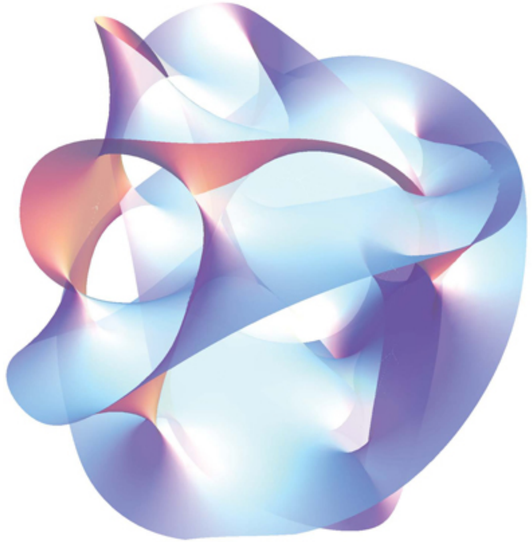
The Upshot

In this talk I discuss one of the recent **Machine Learning approaches to obtain numerical approximations to Ricci flat Calabi-Yau metrics.** Instead of approximating the Kähler potential, we approximate the metric directly by an array neural networks. We apply this approach to the quartic K3, the Dwork quintics and Tian-Yau manifold.

Calabi-Yau manifolds

In string compactifications one is interested in obtaining low energy effective field theories with some remnant supersymmetry

Total Space-time (10D)



Calabi-Yau manifold

Our usual 4D space-time

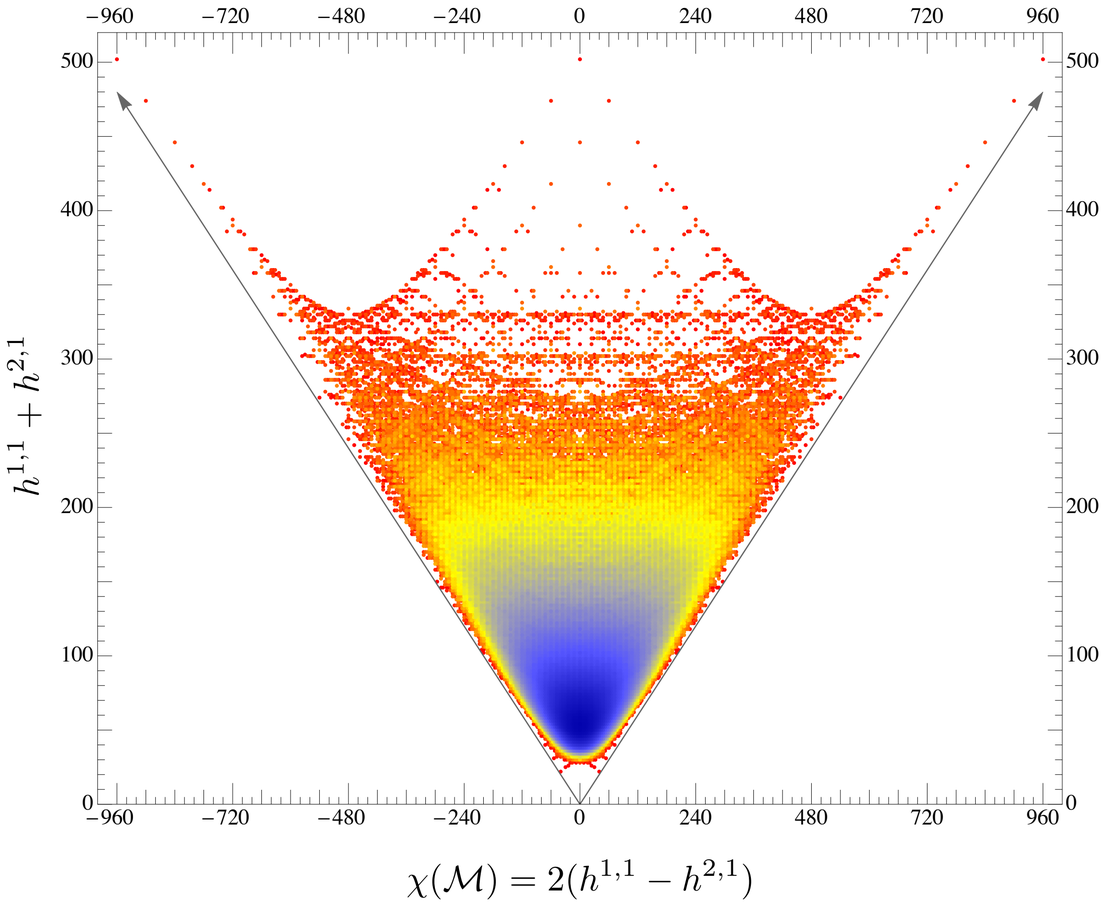
Calabi-Yau manifolds

The total parameter space of a CY threefold:

- $h^{1,1}(\mathcal{M})$ dimension of the Kähler moduli space.
- $h^{2,1}(\mathcal{M})$ dimension of the Complex structure moduli space.
- Calabi-Yau threefolds come in mirror pairs $(\mathcal{M}, \mathcal{M}')$, satisfying

$$h^{1,1}(\mathcal{M}) = h^{2,1}(\mathcal{M}') \qquad h^{2,1}(\mathcal{M}) = h^{1,1}(\mathcal{M}')$$

In simple terms, complex structure and Kähler structure get interchanged. This is the basic idea behind Mirror Symmetry.



Calabi-Yau awesomeness!

@mateofarinella

Calabi-Yau manifolds

A (compact) Calabi-Yau manifold of complex dimension n is a Kähler manifold $(\mathcal{M}, g, J)$ satisfying any of the following equivalent properties:

- The first Chern class of $\mathcal{M}$ is zero.
- $\mathcal{M}$ has a Kähler metric with vanishing Ricci curvature.
- $\mathcal{M}$ has a nowhere vanishing holomorphic n-form.
- $\mathcal{M}$ has a Kähler metric with local holonomy $SU(n)$

Calabi-Yau Manifolds

Some Examples: Calabi-Yaus constructed hypersufaces in projective spaces

-K3 (Fermat Quartic): $z_1^4 + z_2^4 + z_3^4 + z_4^4 = 0 \subset \mathbb{P}^3$,

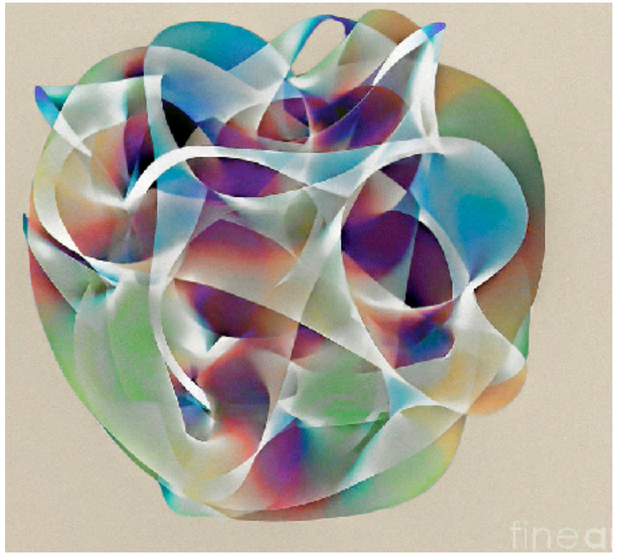
-Fermat Quintic $z_1^5 + z_2^5 + z_3^5 + z_4^5 + z_5^5 = 0 \subset \mathbb{P}^4$

-Dwork $z_1^5 + z_2^5 + z_3^5 + z_4^5 + z_5^5 - 5\psi z_1 z_2 z_3 z_4 z_5 = 0 \subset \mathbb{P}^4 \ , \quad \psi^5 \neq 1$

-Tian Yau $\left[\begin{array}{c|ccc} \mathbb{P}^3 & 3 & 0 & 1 \\ \mathbb{P}^3 & 0 & 3 & 1 \end{array} \right]^{14,\ 23}_{\chi=-18} \quad \Longleftrightarrow \quad \left\{ \begin{array}{ccc} \alpha^{ijk} z_i z_j z_k & = & 0 \ , \\ \beta^{ijk} w_i w_j w_k & = & 0 \ , \\ \gamma^{ij} z_i w_j & = & 0 \ . \end{array} \right.$

-Schoen $\left[\begin{array}{c|cc} \mathbb{P}^1 & 1 & 1 \\ \mathbb{P}^2 & 3 & 0 \\ \mathbb{P}^2 & 0 & 3 \end{array} \right]^{19,19}_{\chi=0} \quad \simeq \quad \left[\begin{array}{c|cccccc} \mathbb{P}^1 & 0 & 0 & 0 & 0 & 1 & 1 \\ \mathbb{P}^2 & 1 & 1 & 0 & 0 & 1 & 0 \\ \mathbb{P}^2 & 1 & 1 & 0 & 0 & 1 & 0 \\ \mathbb{P}^2 & 0 & 0 & 1 & 1 & 0 & 1 \\ \mathbb{P}^2 & 0 & 0 & 1 & 1 & 0 & 1 \end{array} \right]^{19,19}_{\chi=0} .$

Machine Learning Calabi-Yau Metrics



To date we do not have an analytic expression for a Ricci flat Calabi Yau metric.

-The metric can be accessed numerically.

Headrick, Wiseman'05 Anderson, Braun Karp, Ovrut'10
Headrick, Nassar'13, Cui, Gray'19

-More recently, Machine Learning techniques have been used for this endeavour.

Ashmore, Ovrut, He'19 Anderson, Gerdes, Gray, Krippendorf, Raghuram, Rühle'20
Douglas, Lakshminarasimhan, Qi'20 Jejjala, DM, Mishra'20

Being a Kähler manifold, the Hermitian metric g can be derived from a Kähler potential

$$g_{a\bar{b}} = \partial_a \partial_{\bar{b}} K(z^a, \bar{z}^{\bar{b}})$$

The Kähler form is given by:

$$J = \frac{\mathrm{i}}{2} g_{a\bar{b}}\, dz^a \wedge \bar{z}^{\bar{b}}$$

The Ricci tensor is obtained as

$$R_{a\bar{b}} = \partial_a \partial_{\bar{b}} \log \det g$$

Simplest case: The Fubini-Study metric in the ambient space

$$K_{FS} = \frac{1}{\pi} \log(z \cdot \bar{z})$$

restricted to the hypersurface (CICY).

Machine Learning Calabi-Yau Metrics

Donaldson's Algorithm

A valid Kähler potential can be obtained generalizing the Fubini-Study metric to polynomials of a higher degree

$$K^{(k)}(z,\bar{z}) = \frac{1}{k\pi}\log(h^{\alpha\bar{\beta}}s_\alpha \bar{s}_{\bar{\beta}})$$

where the s_α form a basis for holomorphic polynomials over $\mathcal{M}$ up to degree k. The task is to find a Hermitean matrix $h^{\alpha\bar{\beta}}$ for every k, that gives the best approximation to the Ricci flat metric.

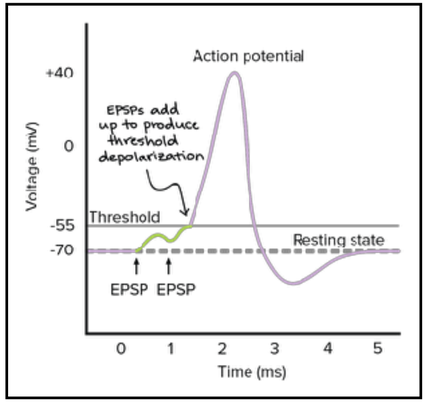
Take N_k to be the dimension of $\{s_\alpha\}$ and define

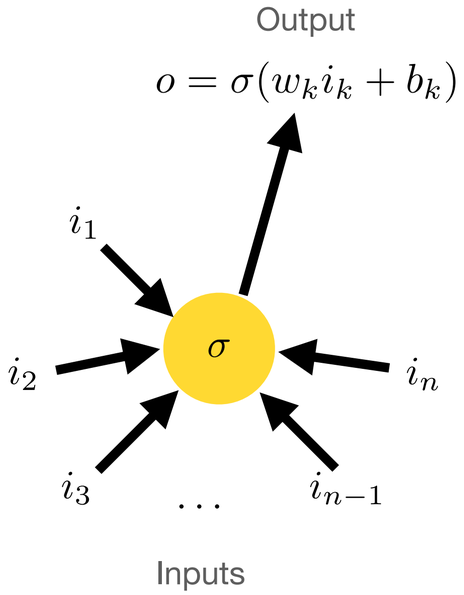
$$H_{\alpha\bar{\beta}} = \frac{N_k}{\mathrm{Vol}_\Omega}\int_{\mathcal{M}} d\mathrm{Vol}_\Omega \left(\frac{s_\alpha \bar{s}_{\bar{\beta}}}{h^{\alpha\bar{\beta}}s_\alpha \bar{s}_{\bar{\beta}}}\right) \qquad\qquad d\mathrm{Vol}_\Omega = \Omega\wedge\bar{\Omega}$$

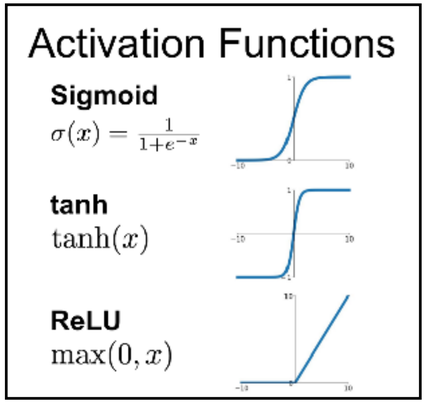
Now take $h^{\alpha\bar{\beta}} = (H_{\alpha\bar{\beta}})^{-1}$ and proceed iteratively until the metric stabilizes. In this manner one obtains the "balanced metric" at degree k.

As the polynomial degree increases the metric $g_{a\bar{b}}$ obtained from the Kähler potential with the balanced metric approaches the desired Ricci flat metric.

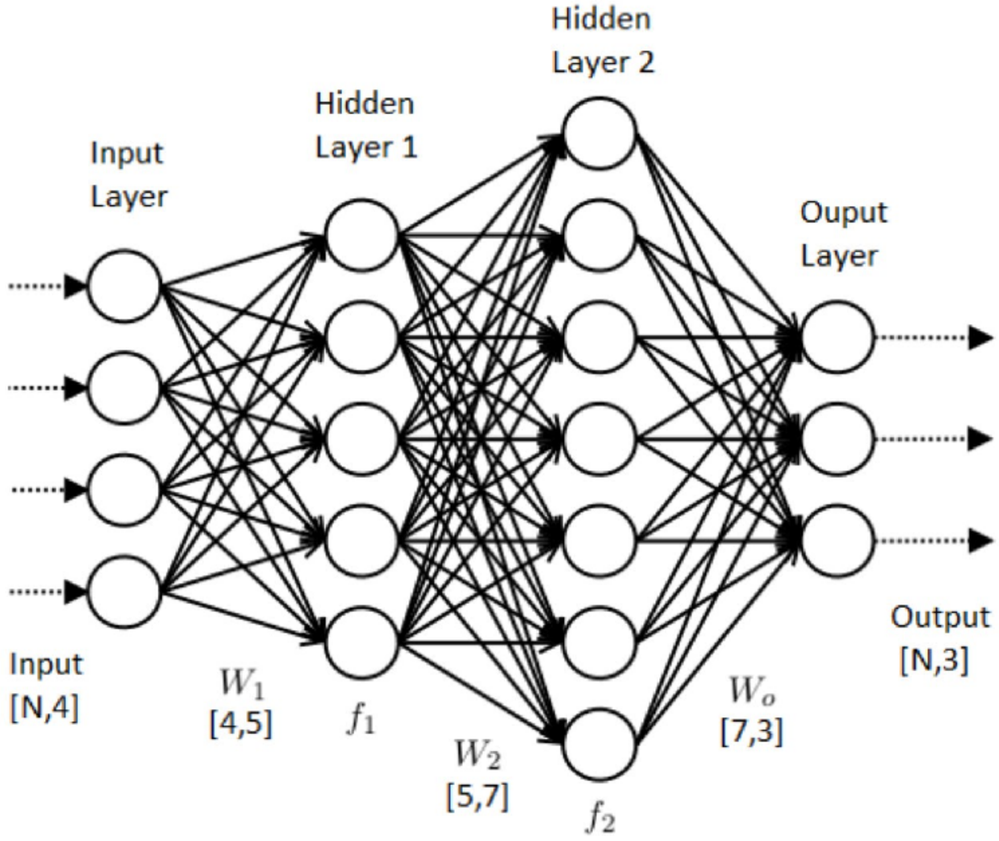
Machine Learning Calabi-Yau metrics







A paradigm in ML are **Artificial Neural Networks**: arrays of artificial neurons that emulate the human brain.



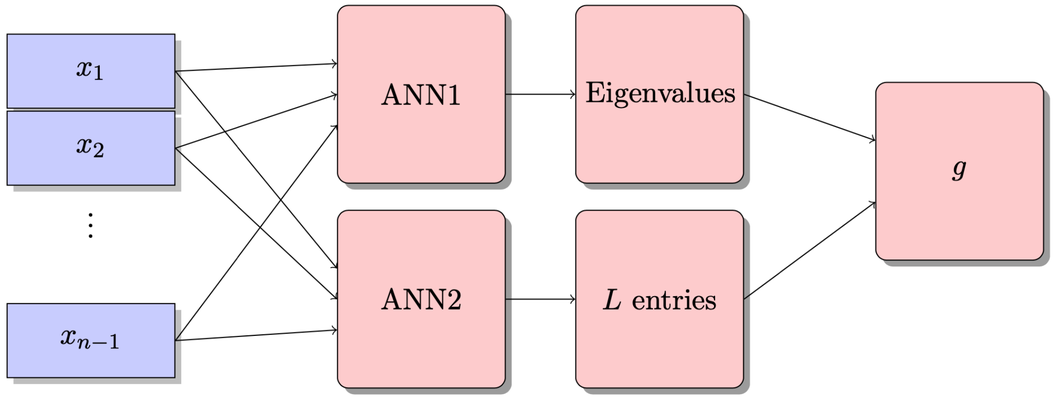
Machine Learning Calabi-Yau Metrics

A Hermitian metric can be written in the LDL decomposition.

$$g = L\,D\,L^{\dagger}$$

With L a lower triangular matrix with 1s in the diagonal and $D = \mathrm{diag}(e_1,\, e_2,\, ...,\, e_n) \quad e_i > 0$

Approximate the metric as a combination of neural networks



Since the eigenvalues have to be positive. we take exponentiate the outputs of ANN1 $e_i = \mathrm{Exp}(o_i^{(1)})$ and use the outputs of ANN2 to construct L in K3 for example,

$$L = \begin{pmatrix} 1 & 0 \\ o_1^{(2)} + \mathrm{i}o_2^{(2)} & 1 \end{pmatrix}$$

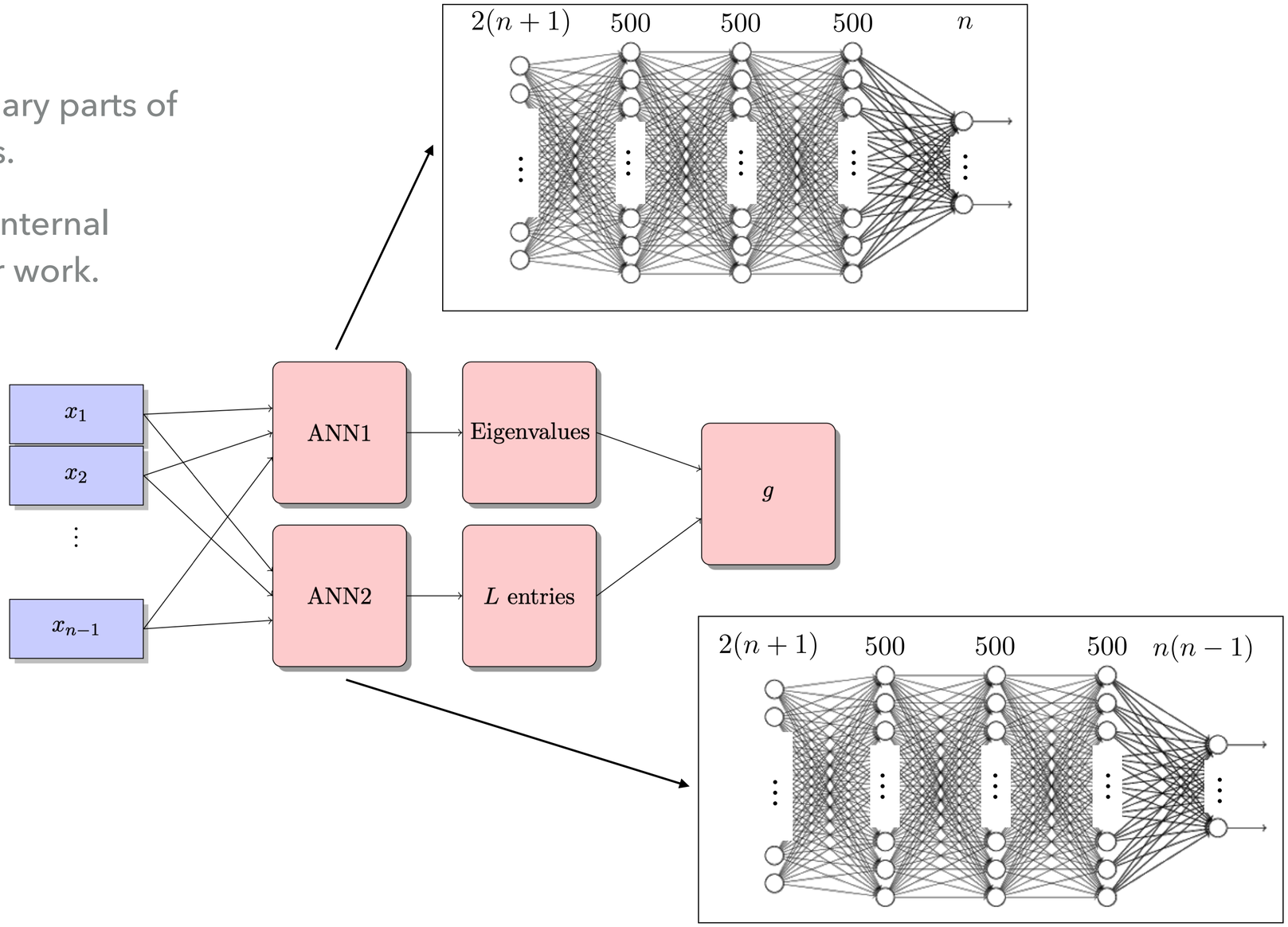
Machine Learning Calabi-Yau Metrics

We have taken real and imaginary parts of the affine coordinates as inputs.

The number of neurons in the internal layers was kept throughout our work.

In all of the experiments three activation functions were used: Logistic Sigmoid, ReLU and Tanh.

The data was prepared in Mathematica and the neural networks were implemented in PyTorch.



Machine Learning Calabi-Yau Metrics

The loss function is constructed for the full network ensemble, and it is minimized for g approaching the Flat metric. It is constructed based on three properties

-Local Flatness

$$\sigma = \frac{1}{\mathrm{Vol}_\Omega} \int_{\mathcal{M}} d\mathrm{Vol}_\Omega \left| 1 - \frac{\mathrm{Vol}_\Omega}{\mathrm{Vol}_J} \cdot \frac{J^n}{\Omega \wedge \bar{\Omega}} \right| .$$

-Kählerity

$$\kappa = \frac{\mathrm{Vol}_J^{1/n}}{\mathrm{Vol}_\Omega} \int_{\mathcal{M}} d\mathrm{Vol}_J \, |k|^2, \quad |k|^2 = \sum_{a,b,\bar{c}} |k_{ab\bar{c}}|^2, \quad k_{ab\bar{c}} = \partial_a g_{b\bar{c}} - \partial_b g_{a\bar{c}} \ .$$

-Patch Matching

$$\mu = \frac{1}{N_p!} \sum_{m',l'} \sum_{m.l \neq m',l'} \frac{1}{\mathrm{Vol}_\Omega} \int_{\mathcal{M}} d\mathrm{Vol}_J \, |M(m',l';m,l)|^2,$$

With the total Loss function being

$$\mathrm{Loss} = \alpha_\sigma \sigma + \alpha_\kappa \kappa + \alpha_\mu \mu \, .$$

Point Sampling

Braun, Brelidze, Douglas, Ovrut'07
Anderson, Braun Karp, Ovrut'10
Ashmore, He, Ovrut'19

Building Lines in $\mathbb{P}^n$

- Start with selecting random points in $[-1,1]^{2(n+1)}$ and use each point to build a complex vector $v \in \mathbb{C}^{n+1}$. Discard those vectors with $|v| > 1$

- Project v onto the surface of the unitary sphere S^{2n+1}

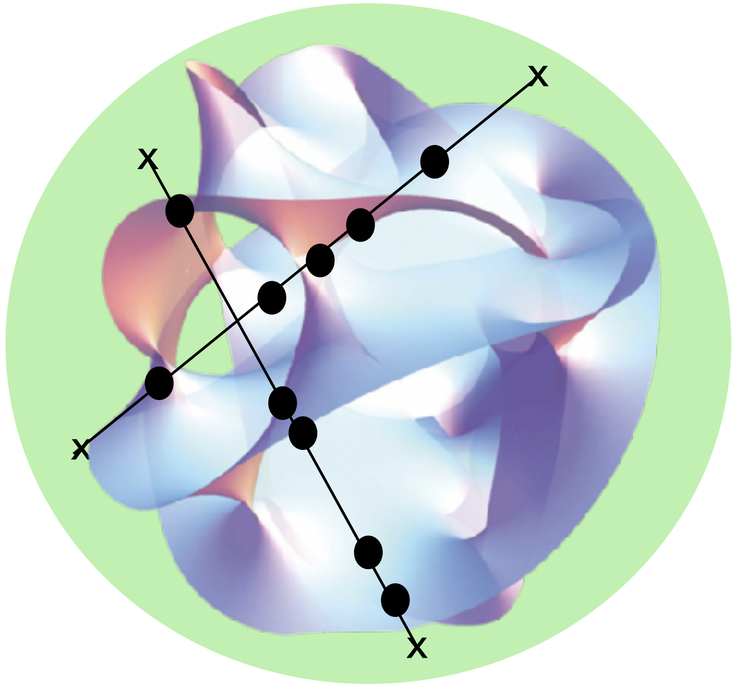
- As points in $\mathbb{P}^n$, the rescaled v's are uniformly distributed with respect to the $SU(n+1)$ symmetry of the Fubini-Study metric in $\mathbb{P}^n$

- Use any two unitary 's to construct a line

$$L_{ij} = \{v_i + \lambda v_j \,|\, \lambda \in \mathbb{C}\}$$

- The sample points in the hypersurface of interest are obtained as

$$L_{ij} \cap \{p = 0\}$$



Point Sampling

Slight Modification for Tian-Yau Manifold

-Sample points in each $\mathbb{P}^3$ identically as before.

-Use the points to obtain a line and a plane $L_{ij} = \{v_i + \lambda v_j \,|\, \lambda \in \mathbb{C}\} \subset \mathbb{P}^3_1$
$P_{ijk} = \{v_i + \alpha\ v_j + \beta v_k \,|\, \alpha\,, \beta \in \mathbb{C}\} \subset \mathbb{P}^3_2$

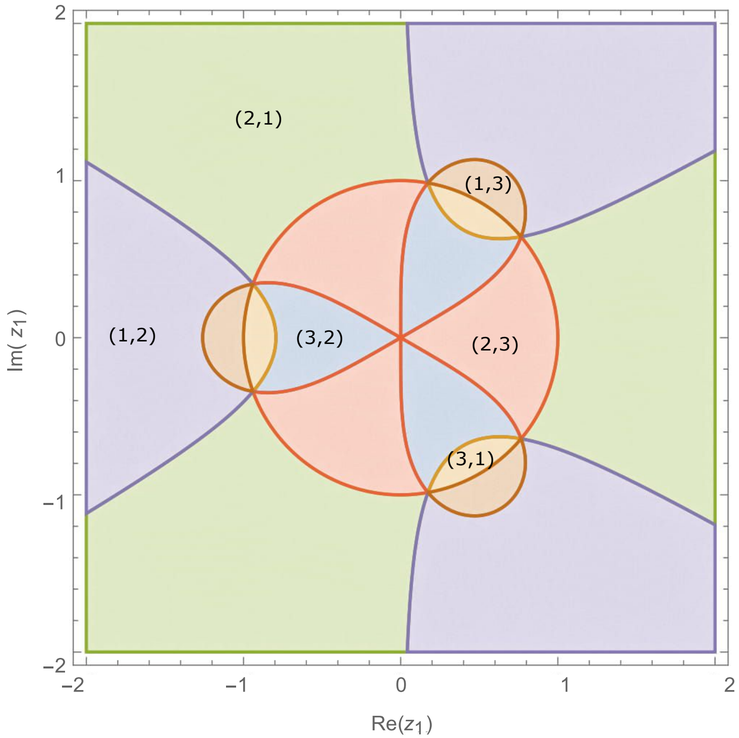
-Take the points in the Tian-Yau manifold as $(L_{ij} \cup P_{klm}) \cap \{p_1 = p_2 = p_3 = 0\}$

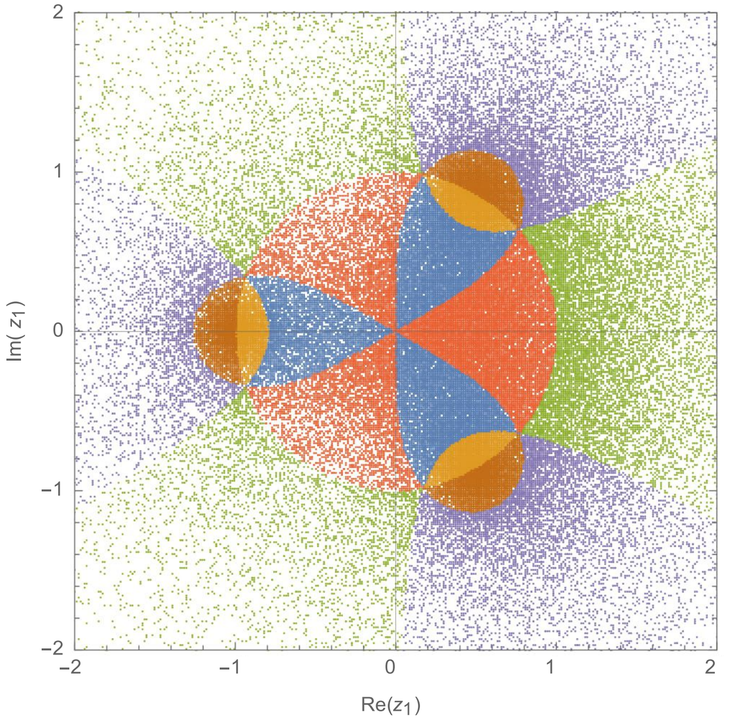
(plus $\mathbb{P}^3_1 \leftrightarrow \mathbb{P}^3_2$)

Sampling points in the torus

$$z_1^3 + z_2^3 + z_3^3 = 0 \subset \mathbb{P}^2$$

For the torus we have six patches fixed upon choice of the affine coordinate and the dependent coordinate. For each point in the torus there is a preferred patch.





Point Sampling

Numerical Integration

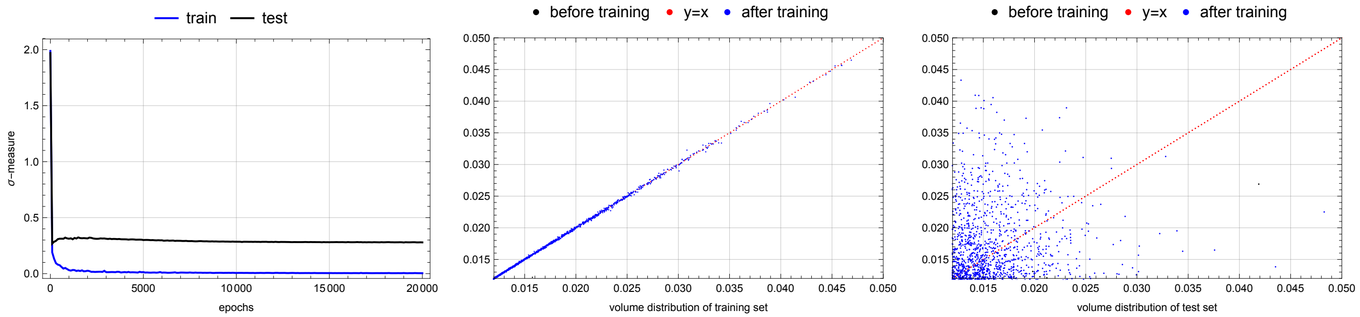
- Patches intersect over zero measure sets, hence numerical integration would be a sum over points in different patches if these would be uniformly distributed with respect to the Calabi-Yau metric.
- The sampling method provides points uniformly distributed with respect to the Fubini-Study metric restricted to the Calabi-Yau.
- Numerical integration requires to weight the sample points in order to obtain meaningful quantities

$$\int_{\mathcal{M}} d\mathrm{Vol}_\Omega \, f(z,\bar{z}) = \int_{\mathcal{M}} d\mathrm{Vol}_{FS} \left(\frac{d\mathrm{Vol}_\Omega}{d\mathrm{Vol}_{FS}} \right) \, f(z,\bar{z})$$

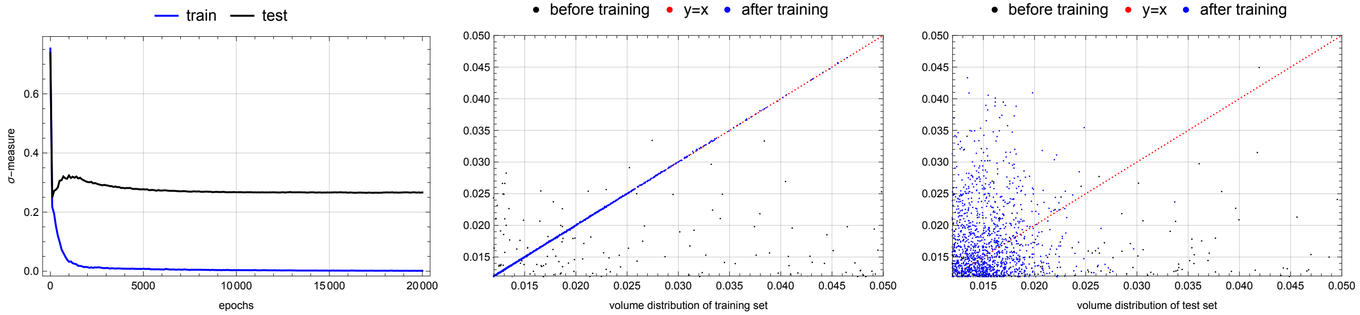
$$\int_{\mathcal{M}} d\mathrm{Vol}_\Omega \, f(z,\bar{z}) = \frac{1}{N} \sum_{l=1}^{M} w(p_l) f(p_l) \qquad\qquad w(p_l) = \frac{d\mathrm{Vol}_\Omega(p_l)}{d\mathrm{Vol}_{FS}(p_l)}$$

Calabi-Yau Metrics (Results)

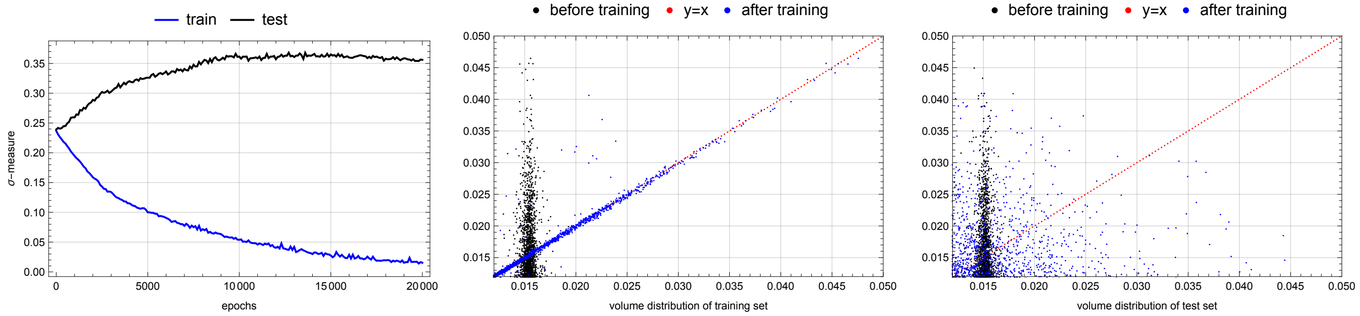
K3: Training and testing with 2000 points



ReLU



Tanh



Sigmoid

Calabi-Yau Metrics (Results)

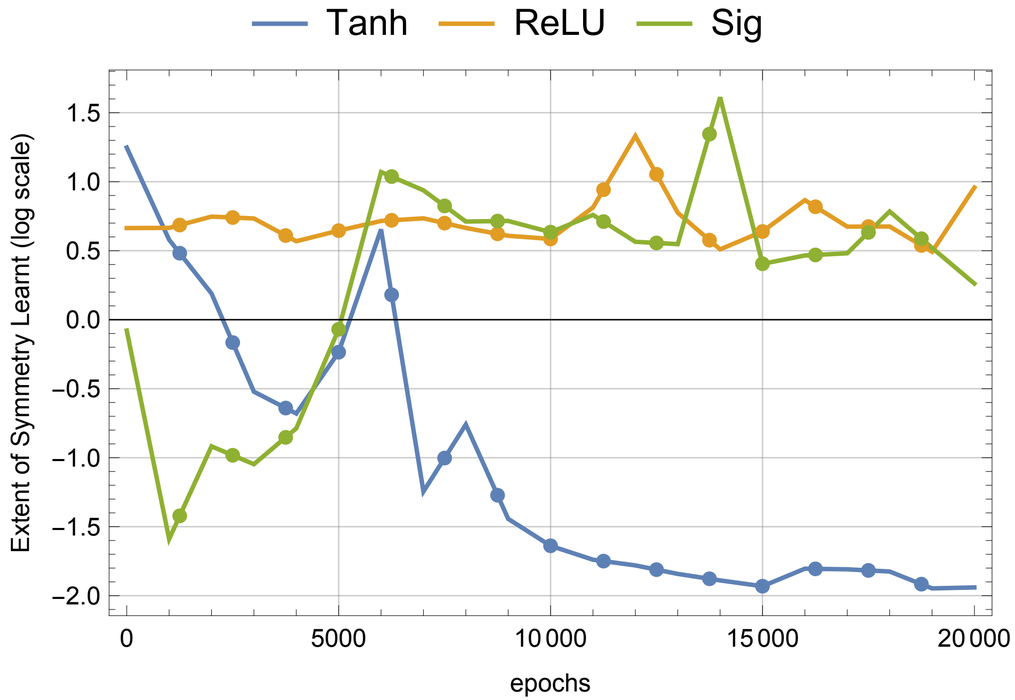
K3: Test of Symmetry Learning

$$\text{K3}: \quad z_1^4 + z_2^4 + z_3^4 + z_4^4 = 0 \subset \mathbb{P}^3$$

$$z_p \to \omega_p \, z_p \; ; \quad \text{with} \quad p \in \{1,2,3,4\} \quad \text{and} \quad \omega_p \in \mathbb{Z}_4$$

$$\tau: \quad z_1 \mapsto i \, z_1 \; , \; z_2 \mapsto - z_2 \; , \; z_3 \mapsto -i \, z_3 \; .$$

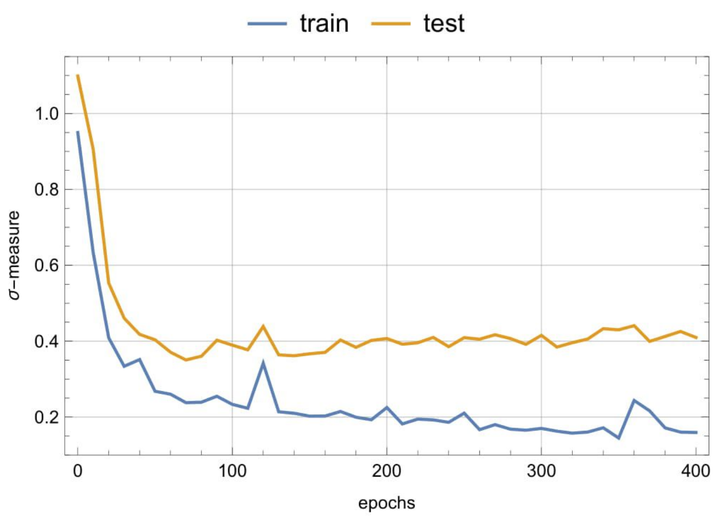
$$\delta_n(\tau) := \frac{1}{N} \sum_z \text{abs}\left(\frac{g_{NN}(\theta_n; z) - g_{NN}(\theta_n; \tau.z)}{g_{NN}(\theta_n; z)} \right)$$



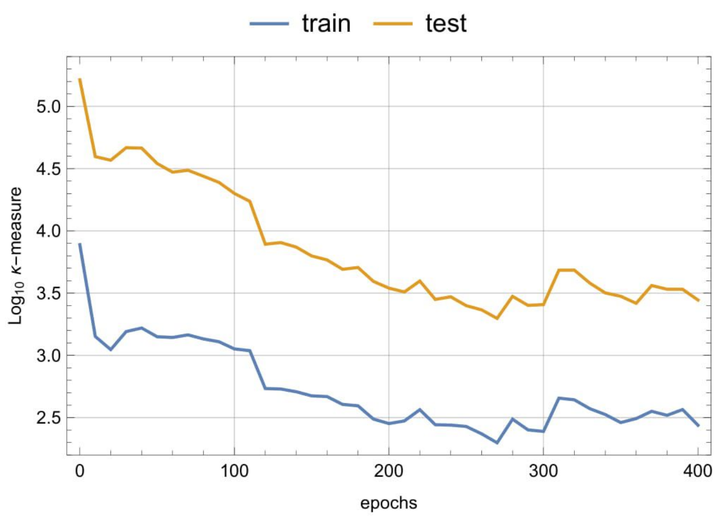
Calabi-Yau Metrics (Results)

General Network Architecture: 3 hidden layers each with different activation functions: ReLU, Logistic Sigmoid, Tanh.

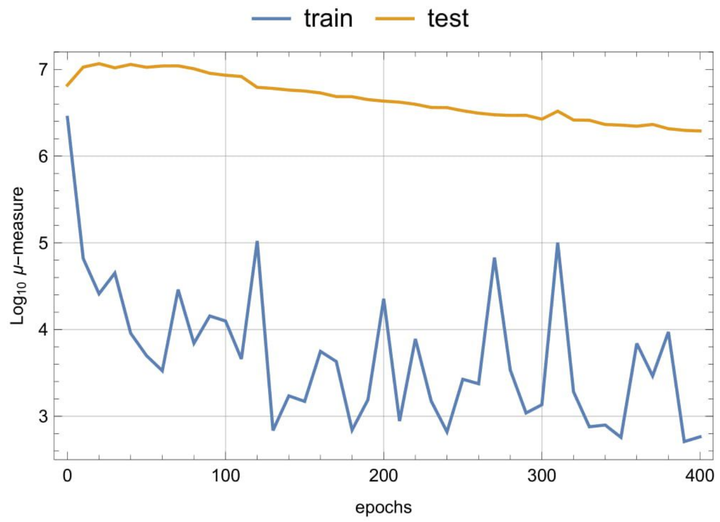
The Fermat Quintic: Training and testing with 2000 points



sigma



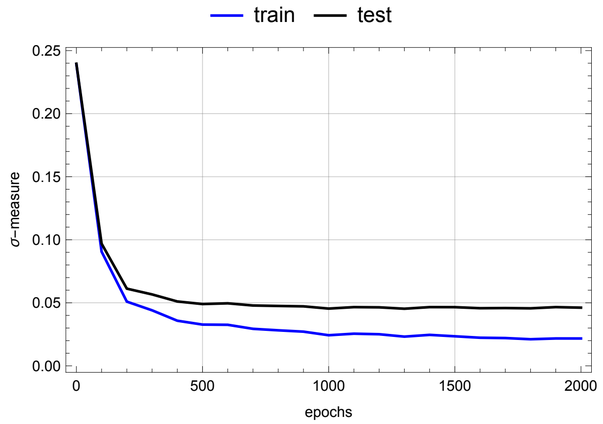
kappa

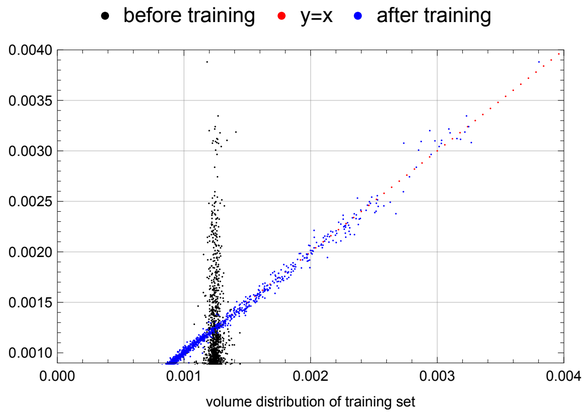


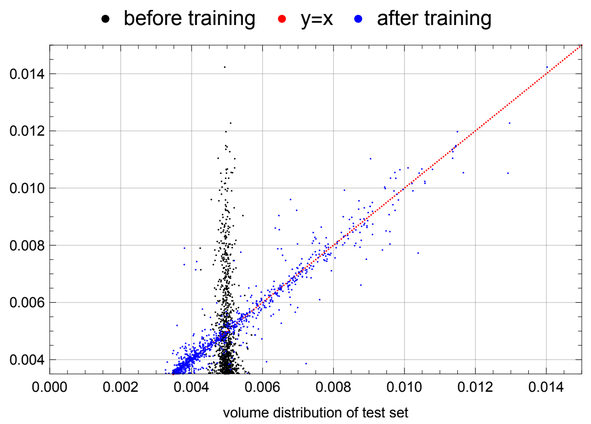
mu

Calabi-Yau Metrics (Results)

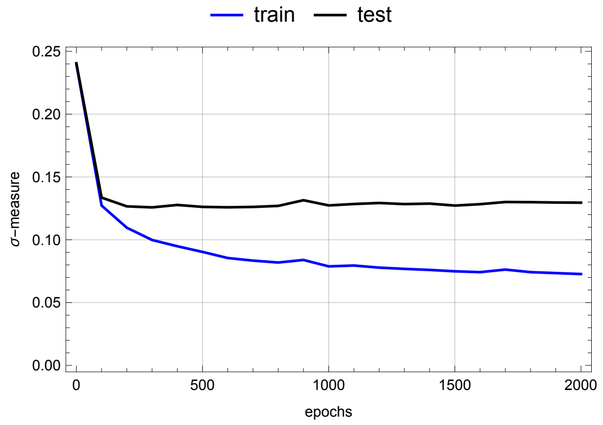
Fermat Quintic, sigma training: 0.5m points, testing: 0.125 m points

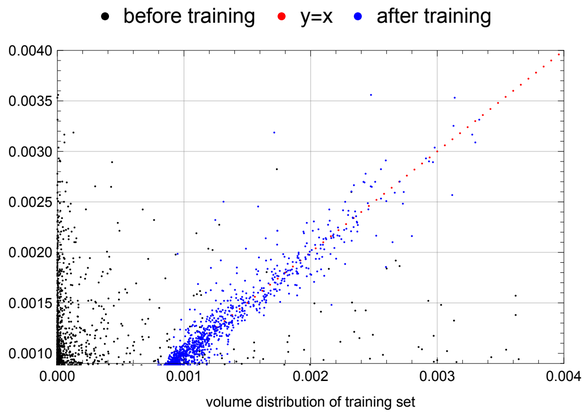


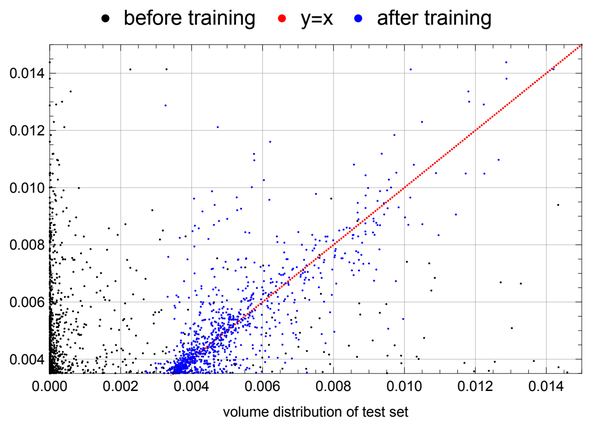




Logistic Sigmoid





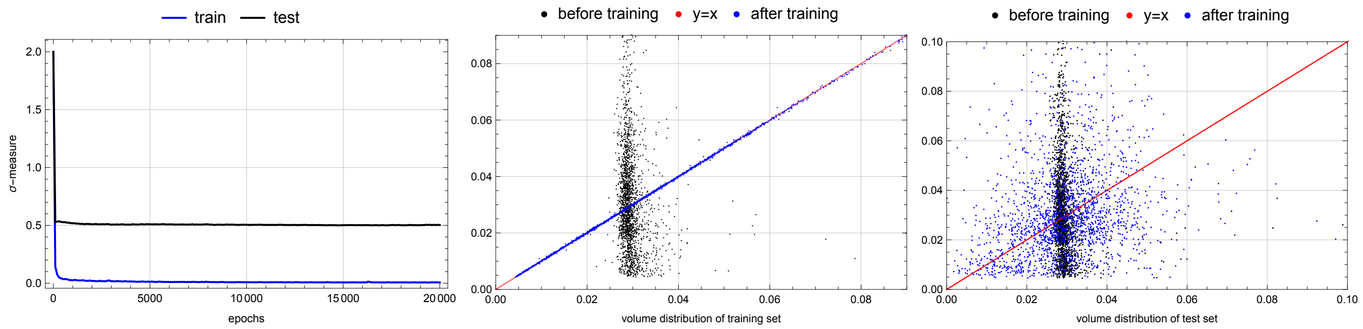


Tanh

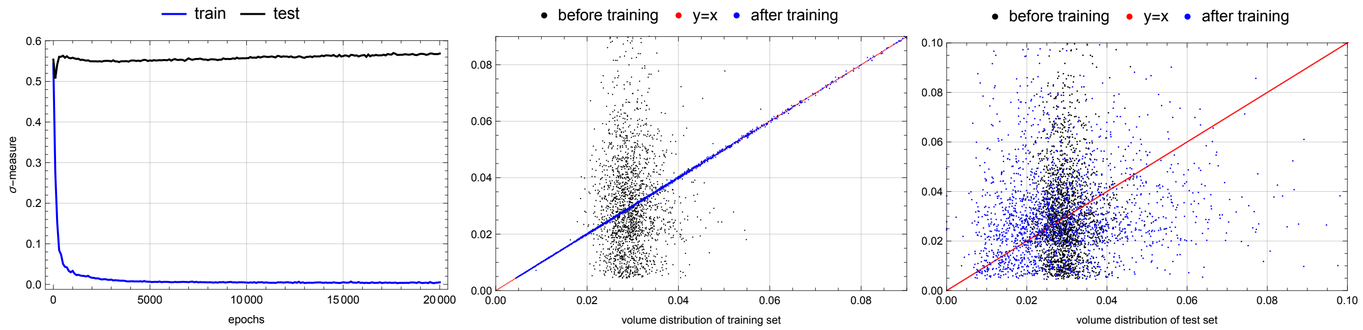
sigma **Training** **Test**

Calabi-Yau Metrics (Results)

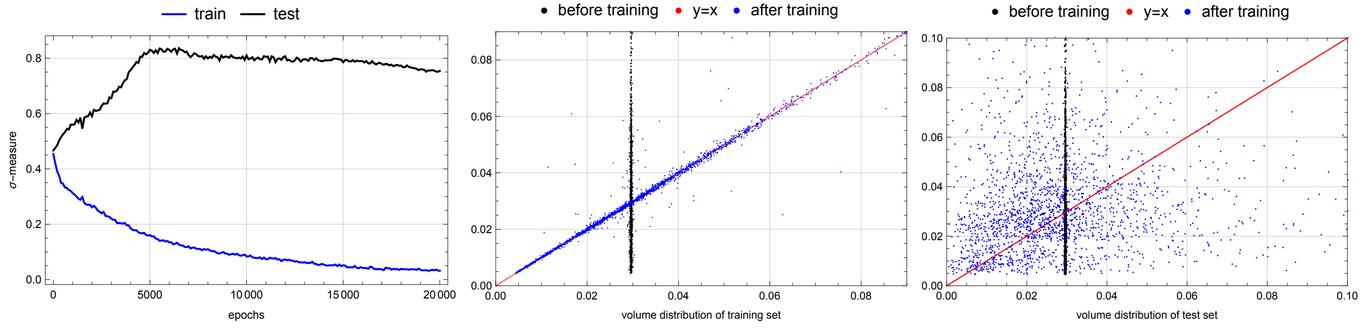
Tian-Yau: Training and Testing 2000 points



ReLU



Tanh



Sigmoid

Final Remarks

-A need for interpretability: Can we deduce analytic expressions for the Calabi-Yau metrics?

-As symmetry learning hints to the best architecture/activation function. Can we use symmetries for architecture selection? Can we develop an AI symmetry classifier?

-The structure of our approach suggests that it can be extended to other Calabi-Yau manifolds, in particular: Complete Intersections.

Thank you!