Topic 3: String Theory Compactifications, Calabi-Yau Manifolds and Ricci-flat Metrics

Lecture 5: CY Metrics from NNs

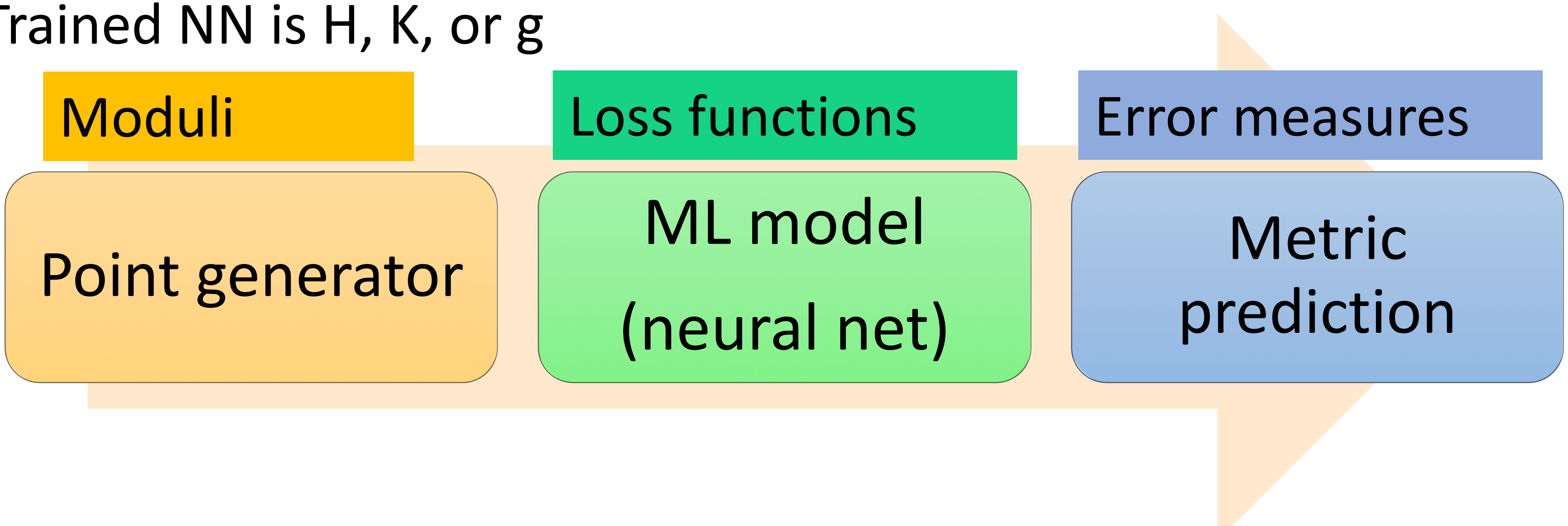
Package details & Advanced methods

Magdalena Larfors, Uppsala University
ML in Maths and Physics 2023

AI4Research

Summary of lecture 4

- ML use NNs to predict CY metrics
- Architecture ⇔ prediction: H, K, or g
- Loss functions ⇔ math/physics constraints (MA eq,…)
- Open-source packages using ML libraries
- Trained NN is H, K, or g



Outline: ML of CY metrics

Lecture 5: details

- Direct learning of CY metrics
  Details & tech comments
  Demo
- Advanced methods
  Goals and realizations

Tutorial

- Implementations & experiments

Main references
Anderson et al 2012.04656, Douglas et al 2012.04797, Larfors et al 2205.13408, Gerdes et al 2211.12520

Direct learning of CY metrics

Details on cymetric package

Package structure: Classes and libraries

Any package needs efficient code.

- cymetric structured in classes:
  1. Point generator
  2. ML models

  inherit from TensorFlow

  and helper functions
- Python libraries (numpy, sympy), Mathematica & SAGE routines used for computations
- Tricks e.g. function decorators, gradient tapes on input
- Python and Mathematica interface

```python
"""
Main PointGenerator module.

:Authors:
    Fabian Ruehle <fabian.ruehle@cern.ch> and
    Robin Schneider <robin.schneider@physics.uu.se>
"""
import numpy as np
import logging
import sympy as sp
from cymetric.pointgen.nphelper import prepare_basis_pickle, prepare_dataset,
from sympy.geometry.util import idiff
from joblib import Parallel, delayed
import itertools

logging.basicConfig(format='%(name)s:%(levelname)s:%(message)s')
logger = logging.getLogger('pointgen')


class PointGenerator:
    r"""The PointGenerator class.

    The numerics are entirely done in numpy; sympy is used for taking
    (implicit) derivatives.

    Use this one if you want to generate points and data on a CY given by
    one hypersurface.

    All other PointGenerators inherit from this class.

```

CY metric on Fermat quintic (using cymetric)

ψ

$\vec{z}$

$\longrightarrow$

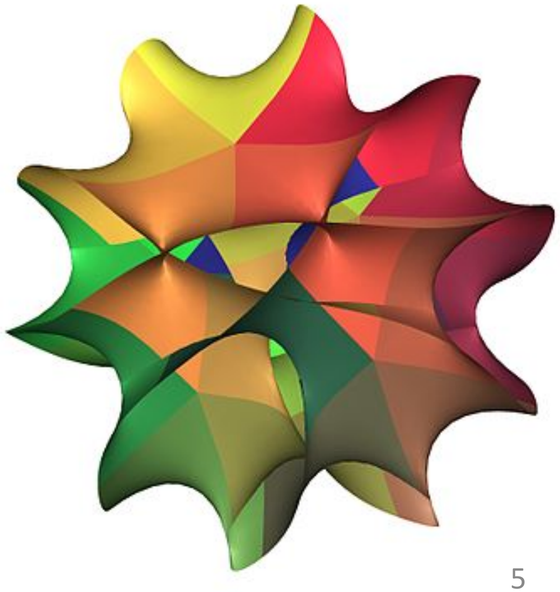
Model

learnable parameters θ

$\longrightarrow$

$g_{a\bar{b}}$

$$p = x_0^5 + x_1^5 + x_2^5 + x_3^5 + x_4^5$$



CY metric on Fermat quintic (using cymetric)

- Input: point on CY
  Quintic: 10 nodes =
  $Re(x_I), Im(x_I)$
- Output: metric prediction -
  9 (or 1) node

- Experiment:
  1. Generate points
  2. Create NN
  3. Choose metric ansatz
  4. Select losses
  5. Run model

Building a ML model

```
import tensorflow as tf
```

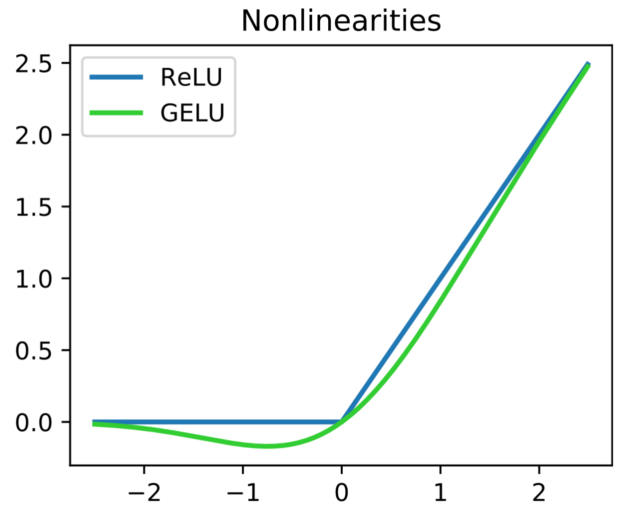
- Layer: takes input, gives output, as specified by weights

```
l=tf.keras.layers.Dense(64, activation='relu')
```

- Neural net: collection of layers, with activation functions

```
nn = tf.keras.Sequential()
nn.add(tf.keras.Input(shape=(n_in)))
nn.add(l)
nn.add(tf.keras.layers.Dense(n_out, use_bias=False))
```

- Model: neural net + loss functions + call backs; trainable on data



Metric ansatze in cymetric

- Different Ansatze possible for metric prediction g_{pr}
  Encode more/less of math knowledge

Name	Ansatz
Free	$g_{\mathrm{pr}} = g_{\mathrm{NN}}$
Additive	$g_{\mathrm{pr}} = g_{\mathrm{FS}} + g_{\mathrm{NN}}$
Multiplicative, element-wise	$g_{\mathrm{pr}} = g_{\mathrm{FS}} + g_{\mathrm{FS}} \odot g_{\mathrm{NN}}$
Multiplicative, matrix	$g_{\mathrm{pr}} = g_{\mathrm{FS}} + g_{\mathrm{FS}} \cdot g_{\mathrm{NN}}$
ϕ-model	$g_{\mathrm{pr}} = g_{\mathrm{FS}} + \partial\bar{\partial}\phi$

```

class AddFSModel(FreeModel):
    r"""AddFSModel inherits from :py:class:`FreeModel`.

    Example:
        Is identical to :py:class:`FreeModel`. Replace the model accordingly.
    """
    def __init__(self, *args, **kwargs):
        r"""AddFSModel is a tensorflow model predicting CY metrics.

        The output of this model has the following Ansatz

        .. math:: g_{\text{out}} = g_{\text{FS}} + g_{\text{NN}}

        and returns a hermitian (nfold, nfold)tensor.
        """
```

Loss functions in cymetric

$$\mathcal{L} = \alpha_1 \mathcal{L}_{\text{MA}} + \alpha_2 \mathcal{L}_{\text{dJ}} + \alpha_3 \mathcal{L}_{\text{transition}} + \alpha_4 \mathcal{L}_{\text{Ricci}} + \alpha_5 \mathcal{L}_{\text{Kclass}}$$

```
    def sigma_integrand_loss(y_true, y_pred):
        r"""Monge-Ampere integrand loss.

        l = |1 - det(g)/ (Omega \wedge \bar{Omega})|

        Args:
            y_true (tensor[(bsize, x), tf.float]): some tensor
                        with last value being (Omega \wedge \bar{Omega})
            y_pred (tensor[(bsize, 3, 3), tf.float]): NN prediction

        Returns:
            tensor[(bsize, 1), tf.float]: loss for each sample in batch
        """
        omega_squared = y_true[:, -1]
        det = tf.math.real(tf.linalg.det(y_pred))*factorial/det_factor
        return tf.abs(tf.ones(tf.shape(omega_squared), dtype=tf.float32) -
                      det/omega_squared/kappa)
```

- Once network, callbacks and loss functions are set up, call the relevant TF class for the model, then train
- e.g. in Jupyter notebook, do

```
In [16]: fmodel = MultFSModel(nn, BASIS, alpha=alpha)
```

we define some custom metrics to track our training progress and the optimizer

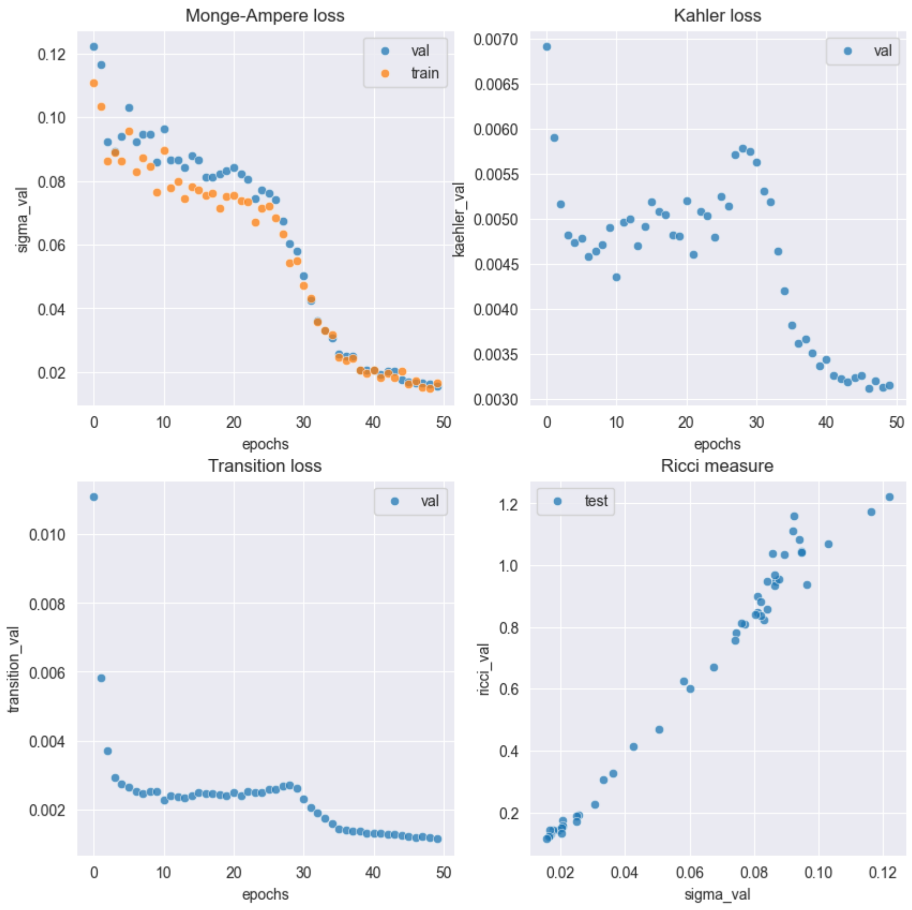
```
In [17]: cmetrics = [TotalLoss(), SigmaLoss(), KaehlerLoss(), TransitionLoss(), VolkLoss(), RicciLoss()]
         opt = tfk.optimizers.Adam()
```

compile and train the model

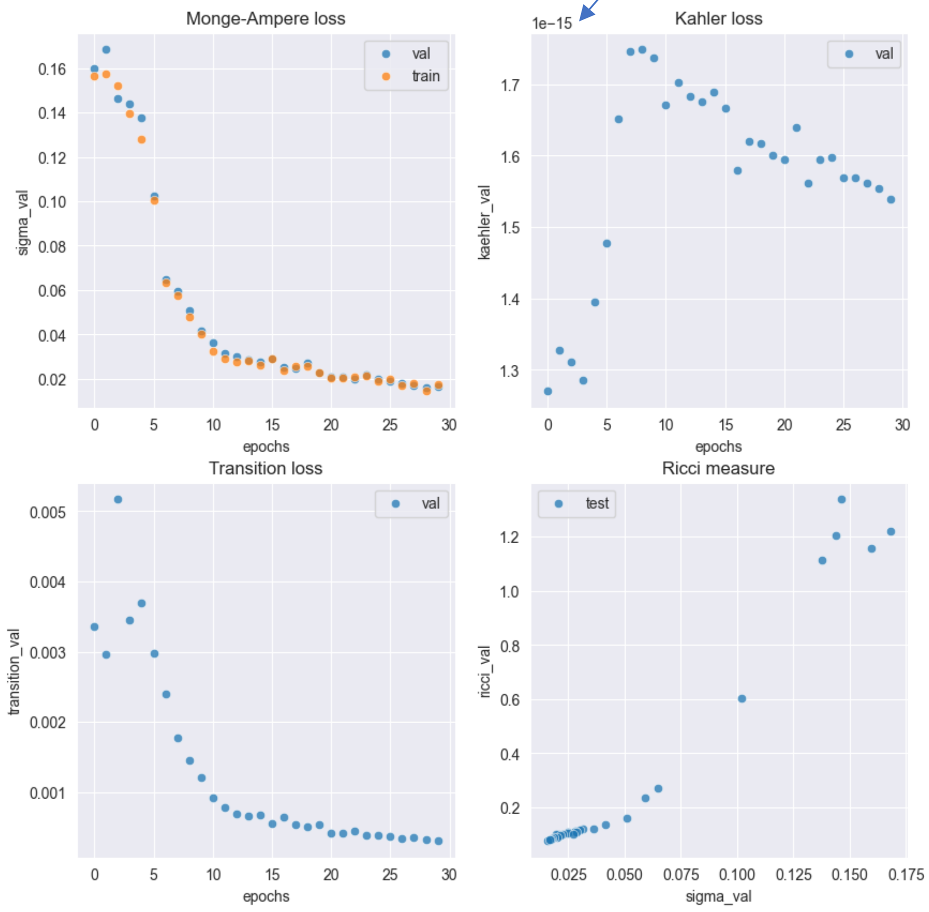
```
In [19]: fmodel, training_history = train_model(fmodel, data, optimizer=opt, epochs=nEpochs, batch_sizes=[64, 50000],
                                                verbose=1, custom_metrics=cmetrics, callbacks=cb_list)
```

Cymetric results

Really small; stays Kahler



MultFS model



PhiFS model

Can also run all of the above in Mathematica

Quintic

Example Jupyter and Mathematica notebooks on https://github.com/pythoncymetric/cymetric

Compute Points and Metric

To look at the parameters and options of a function, simply call ?<FunctionName> and Options[<-FunctionName>]

```
In[•]:= ? GeneratePoints
        Options[GeneratePoints]
```

Now we generate some points. We set the output Directory to “Quintic”

```
In[•]:= outDir = FileNameJoin[{NotebookDirectory[], "Quintic"}];
        ChangeSetting["Dir", outDir];
```

```
In[•]:= poly = {z0^5 + z1^5 + z2^5 + z3^5 + z4^5 + 10 z0 z1 z2 z3 z4};
        res = GeneratePoints[poly, {4}, "Points" → 100 000, "KahlerModuli" → {1}];
```

Now we train the NN. We choose the PhiFS model here. (Training can be made faster if one sets “EvaluateModel”->False).
We also set Verbose to 3 to see some more info during the training process.

```
In[•]:= history = TrainNN["Epochs" → 50, "EvaluateModel" → True, "Verbose" → 3];
```

Advanced methods

Goals and realizations

What problems do we want to solve?

Need CY metric to compute physics (interactions & masses)

- Generality (methods that work on different CY)
- Fast and robust prediction
- Moduli dependence
- Very accurate (e.g. at stable points in moduli space)
- Metrics on submanifolds

…

What problems do we want to solve?

1. Moduli dependent metric
   Anderson et al 2012.04656, Gerdes et al 2211.12520, cyjax
2. Efficiency and accuracy
   (aka make the most of what we know: symmetries & Kahler geometry)
   Douglas et al 2012.04797, Gerdes et al 2211.12520, Berglund et al 2211.09801, holomorphic & bihomogeneous NN, cyjax
3. Generality
   point generator + architecture for CICYs and KS CYs
   Larfors et al 2205.13408, cymetric
4. Learn metric of SU(3) structure manifolds
   Anderson et al 2012.04656
5. Use moduli-dependent metric to compute spectra etc
   e.g. Swampland checks Ashmore-Ruehle 2103.07472, Ahmed-Ruehle 2304.00027

Moduli dependence

Learning moduli dependence of metric

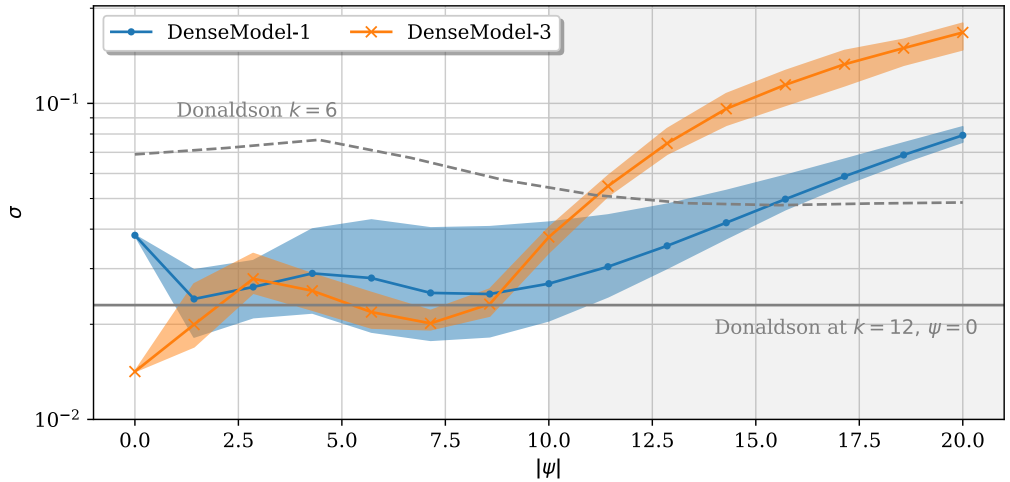
- The quintic CY is really a family of CY manifolds
  - 1 Kähler modulus
  - 101 complex structure moduli
    (previously, we have set most to zero by hand; also required for symmetry)

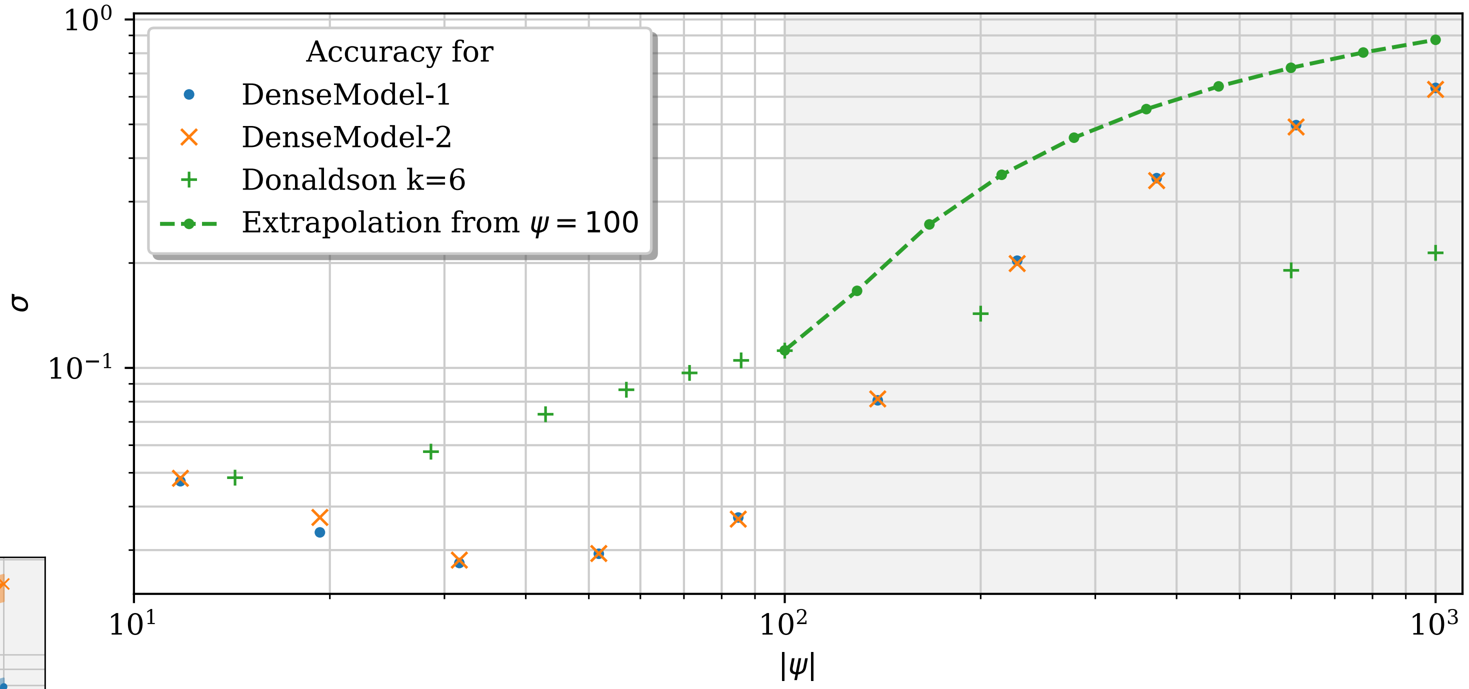
- With ML can learn the moduli-dependent metric
- Challenging, since moduli enters in subtle ways
  - Point sampling (restricting to CY, computing weights)
  - Loss functions

Example: learning moduli dependence

Anderson et al 2012.04656

- Direct learning of H
- Input cpl str modulus ψ
- 1 or 2 hidden layers
- Sigmoid act functions
- Output H

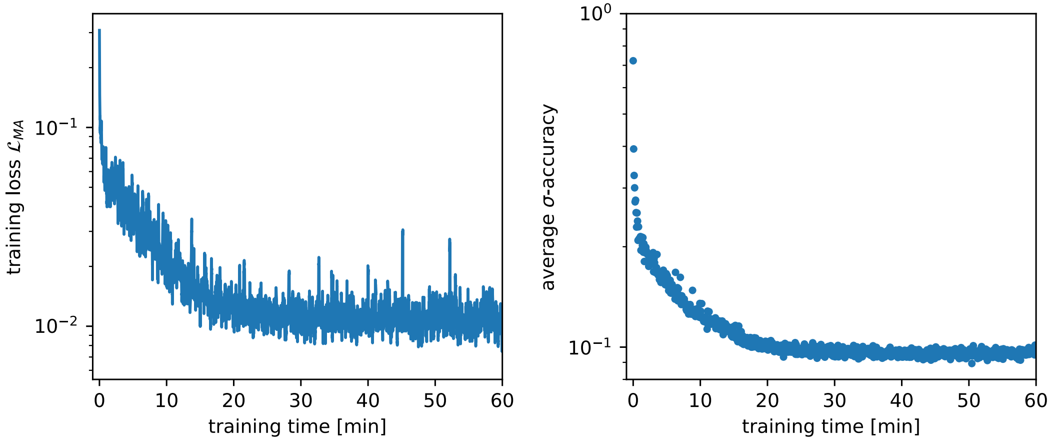


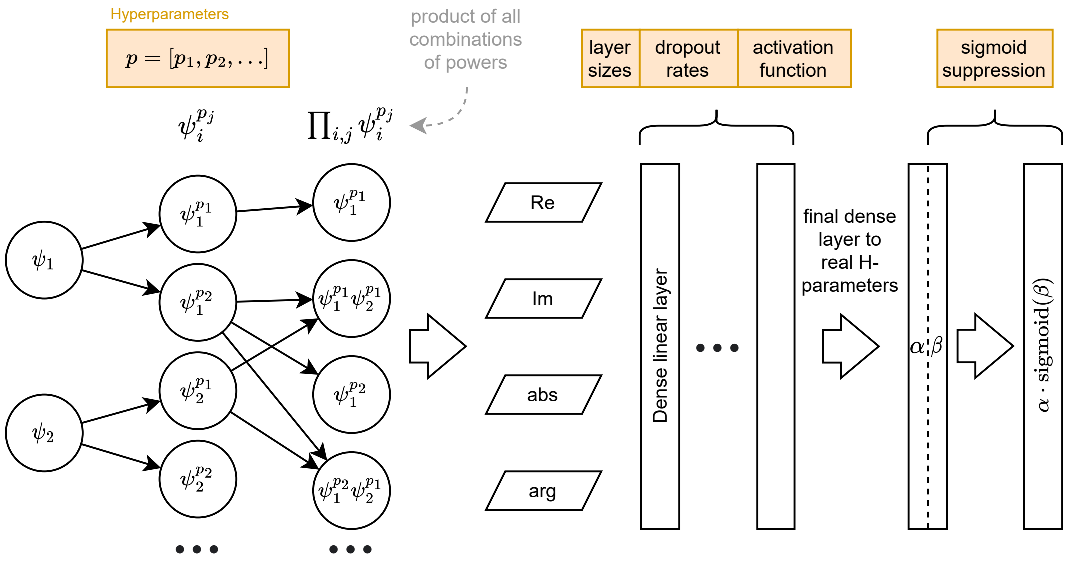


Example: learning moduli dependence

[Gerdes et al 2211.12520](https://arxiv.org/abs/2211.12520)

- Learning H-matrix with cyjax
- MA loss
- Quintic, 2 cpl str moduli





Accuracy and symmetries

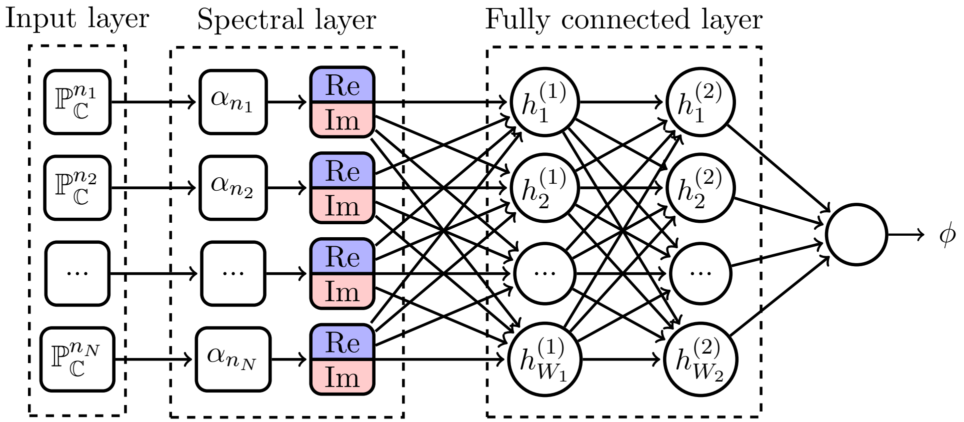
Example: accuracy and symmetries

- The accuracy of the learned metric varies with complex structure
- Metrics harder to learn at/near singularities
- Can check performance by computing topological invariants (will also depend on point sampling!)
- Note: can compute such invariants with any metric, e.g. FS
- Embedding/spectral layers may improve things

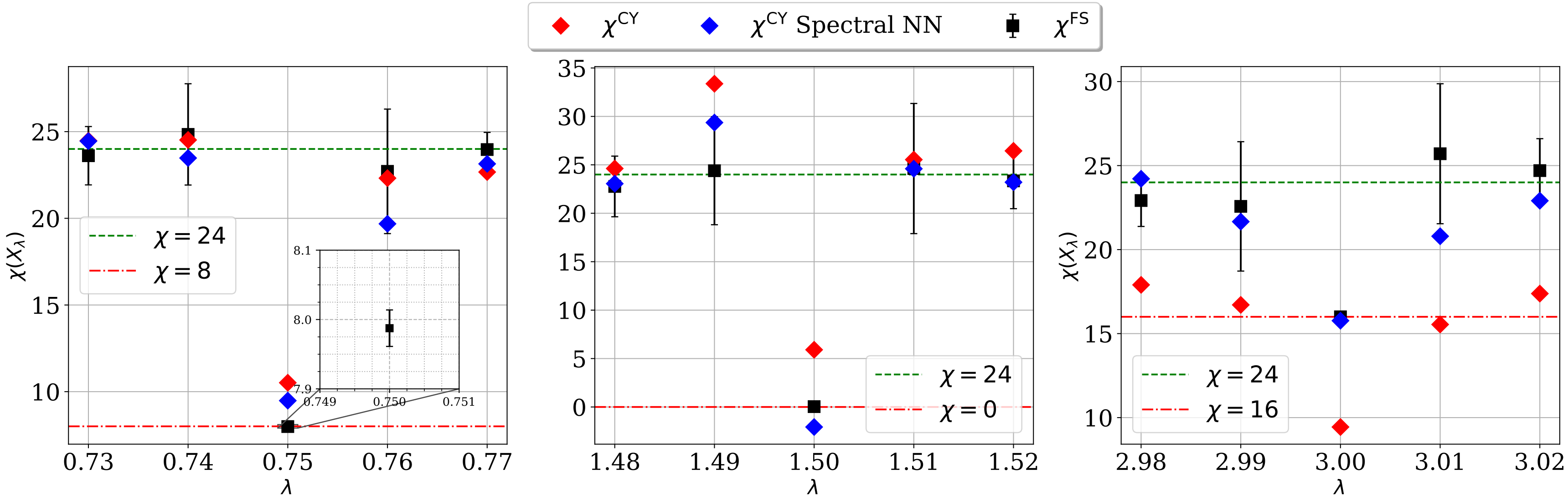
cf. holomorphic/bihom. NNs

Example: accuracy and symmetries

Berglund et al 2211.09801



$$\mathbb{P}^3 \supset X_\lambda := \{p_\lambda(z) = 0\} : \quad p_\lambda(z) := \sum_{i=0}^{3} z_i^4 - \frac{\lambda}{3}\left(\sum_{i=0}^{3} z_i^2\right)^2 .$$



Beyond the quintic

CICY and Kreuzer-Skarke CY

- Quintic CY: $X = \left[\mathbb{P}^4 | 5\right]$
- We can change ambient space, and polynomial eq and still get a CY
- CICY

$$X = \left[\begin{array}{c|ccc} \mathbb{P}^{n_1} & q_1^1 & \cdots & q_K^1 \\ \vdots & \vdots & \vdots & \vdots \\ \mathbb{P}^{n_m} & q_1^m & \cdots & q_K^m \end{array}\right]$$

- Hypersurfaces in toric ambient spaces (Kreuzer-Skarke list)

New challenge: $h^{(1,1)} > 1$

- Many Kahler classes
  How do we guarantee we stay in the same during training?

New challenge: $h^{(1,1)} > 1$

- Many Kahler classes
  How do we guarantee we stay in the same during training?

- Loss function preserving $[\,J\,]$!
- Take basis of line bundles and keep track of their slopes (topological)

$$\mu_J := \int_X J \wedge J \wedge c_1(\mathcal{O}_X(k)) = -\frac{i}{2\pi}\int_X J \wedge J \wedge F = d_{\alpha\beta\gamma} t^\alpha t^\beta k^\gamma$$

New challenge: $h^{(1,1)} > 1$

- Many Kahler classes
  How do we guarantee we stay in the same during training?

- Loss function: for h^{11}-dim basis of line bundles with $k^1 = (1, 0, 0, \ldots)$ etc. compute
$$\mathcal{L}_{\text{K-class}} = \frac{1}{h^{11}} \sum_{i=1}^{h^{11}} \left|\left|\mu_{J_{\text{FS}}}(L_i) - \int_X J_{\text{pr}} \wedge J_{\text{pr}} \wedge F_i\right|\right|_n$$
- Requires more points than contained in mini-batch; NN code more involved.
- Cross-check after training: compute volume and line bundle slopes from intersection numbers, from FS metric and from CY metric.

Example: Bicubic

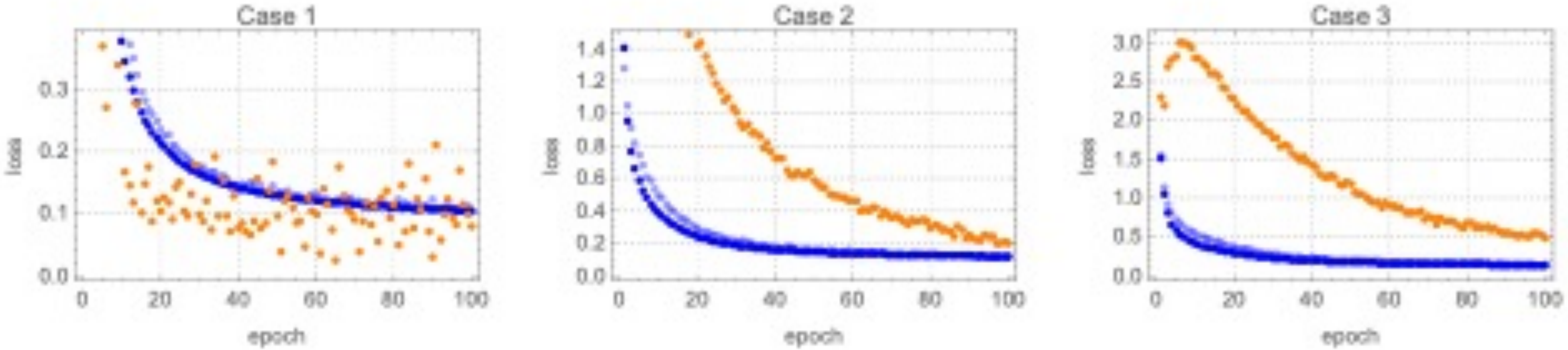
Given by a homogeneous degree (3,3) polynomial in $\mathcal{A} = \mathbb{P}^2 \times \mathbb{P}^2$.
Has 2 Kähler moduli and 83 complex structure moduli.

Choose complex structure moduli, i.e. specify the (3,3) polynomial.
Choose several Kähler moduli paired with line bundles of vanishing slope.

`cymetric` point generation and training

- Generate 100 000 points for each choice of Kähler parameters
- Train ϕ-model for 100 epochs (width 64, depth 3, GELU activation functions, batch size of 64, learning rate of $1/1000$).
  Training has been carried out on a single CPU in about two hours.

Example: Bicubic



orange: $\mathcal{L}_{\text{Kclass}}$, blue: $4 \times \mathcal{L}_{\text{MA}}$, both on training data, light-blue: $4 \times \sigma$ measure on validation data

Check volumes and slopes agree

case	1	2	3	4	5	6	7
Vol_{int}	8.49	4.97	2.93	2.02	6.87	7.59	6.16
Vol_{FS}	8.49	4.50	2.94	2.03	6.91	7.58	6.26
error	$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$	$\sim 2\%$
Vol_{CY}	8.56	5.03	2.96	2.03	6.86	7.58	6.28
error	$< 1\%$	$\sim 1\%$	$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$	$\sim 2\%$

New challenge: toric ambient space

- Requires new point generator
  Atill want points with known distribution
- Idea:
  - embed toric space into projective space (redundant description)
  - sample points, again using Shiffman-Zelditch theorem
  - sort out some subtleties with redundancies
  - re-express in toric coordinates
- Use functionality of Sage for toric geometry

`cymetric`: Point generators

Creating a point sample on KS CY 3-fold

Can we relate ambient toric variety $\mathcal{A}$ to projective spaces?
$\Longrightarrow$ and so apply Shiffman–Zelditch theorem, and generalise the CICY algorithm.

- Sections $s_j^{(\alpha)}$ of the toric Kähler cone generators $J_\alpha \sim$ coordinates of $\mathbb{P}^{r_\alpha}$
- Use Shiffman–Zelditch on $\mathbb{P}^{r_\alpha}$
- Express CY 3-fold as non-complete intersection in $\hat{\mathcal{A}} \cong \bigotimes_{\alpha=1}^{h^{1,1}} \mathbb{P}^{r_\alpha}$
- Intersect $\rightsquigarrow$ sample of random points on CY distributed wrt FS measure.
- $\boxed{\text{Implemented in } \texttt{cymetric} \text{ as } \texttt{ToricPointGeneratorMathematica}}$

`cymetric`: Point generators

Creating a point sample on KS CY 3-fold, part 1

Can relate ambient toric variety $\mathcal{A}$ to projective spaces
$\implies$ can apply Shiffman–Zelditch theorem, and generalise the CICY algorithm.

- Sections of the toric Kähler cone generators $J_\alpha \sim$ coordinates of $\mathbb{P}^{r_\alpha}$

$$\Phi_\alpha : \quad [x_0 : x_1 : \ldots] \quad \rightarrow \quad [s_0^{(\alpha)} : s_1^{(\alpha)} : \ldots : s_{r_\alpha}^{(\alpha)}]$$

- FS metrics on $\mathbb{P}^r \longrightarrow$ (non-FS) Kähler metric on $\mathcal{A}$.
- Can build random sections

$$S = \sum_{j=0}^{r_\alpha} a_j^{(\alpha)} s_j^\alpha$$

  using i.i.d. Gaussian coefficients $a_j^{(\alpha)} \sim \mathcal{N}(0,1)$
- **Theorem**[Shiffman and Zelditch]:
  Zeros of random sections are distributed according to the FS measure.

`cymetric`: Point generators

Creating a point sample on KS CY 3-fold, part 2

Got map $\Phi_\alpha : \quad [x_0 : x_1 : \ldots] \quad \to \quad [s_0^{(\alpha)} : s_1^{(\alpha)} : \ldots : s_{r_\alpha}^{(\alpha)}]$ and
Know zeros of random sections $S = \sum_{j=0}^{r_\alpha} a_j^{(\alpha)} s_j^\alpha$ have good distribution.

- Express the CY 3-fold in terms of Kähler cone sections $s_j^{(\alpha)}$
  - Problem 1: too many sections! Problem 2: relations among sections!
- First find relations among sections ...
  - Groebner basis analysis using Singular (access via Sage)
  - Linear algebra routine (faster, requires generic points in section space)

$$\prod_I s_I^{f_I} = \prod_J s_J^{g_J} \Leftrightarrow \prod_I s_I^{h_I} = 1 \ , \ s_J = \prod_a x_a^{M_{a,J}} \implies \sum_I M_{a,I} h_I = \vec{0}_a$$

- ... then combine relations + hypersurface eq:
  CY 3-fold as non-complete intersection in $\hat{\mathcal{A}} \cong \bigotimes_{\alpha=1}^{h^{1,1}} \mathbb{P}^{r_\alpha}$.
- Intersect: random point sample on CY distributed wrt FS measure.

Point generation on KS CY: example

- Toric ambient space $\mathbb{P}^1 \hookrightarrow \mathcal{A} \rightarrow \mathbb{P}^3$, coordinates $x_0, \ldots x_5$.
  Two generators of Kähler cone: J_1, J_2
- CY hypersurface specified by $p(x_0, \ldots, x_5) = 0$ polynomial with 80 terms.
- Sections s^α

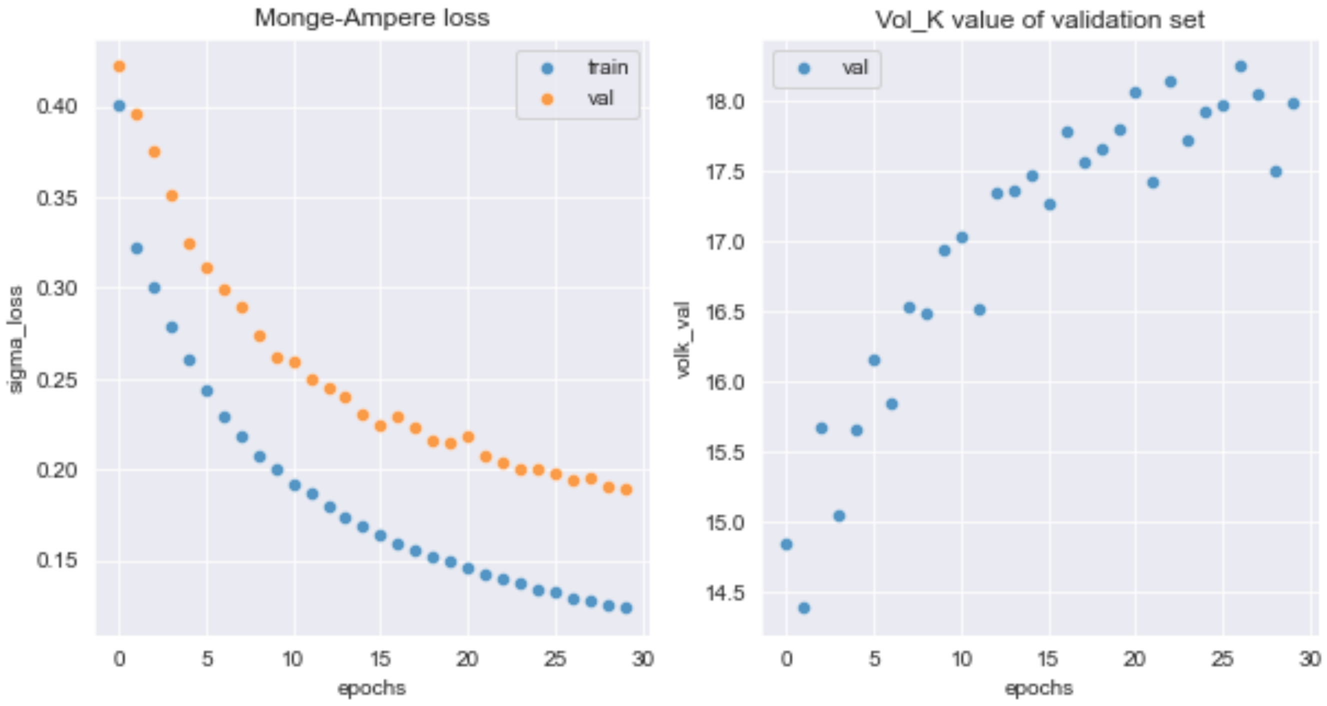
$$H^0(J_1) = (x_1, x_4)\,, \qquad H^0(J_2) = (x_0, x_2, x_3, x_1^2 x_5, x_4^2 x_5, x_1 x_4 x_5)\,,$$

  define the morphisms $\Phi_{1,2}$ into $\mathbb{P}^1$ and $\mathbb{P}^5$.
- Point generation ~ 1 hour (generic cpl structure moduli, and $t_1 = t_2 = 1$).

Example: CY in toric ambient space

Toric ϕ-model on 50 000 points, 30 epochs
(width 64, depth 3, GELU activation, batch size of 64, learning rate of 1/1000).
Point generation: about 1 hour, Training: about three hours (single CPU).

MA loss and volume (exact 20; more points/epochs needed)



Summary of this lecture

With available packages, we can

- Use ML to predict Ricci-flat metrics on CY manifolds (of interest in e.g. string theory)
- Most work done on the quintic (with 1-2 cpl moduli)
  cyjax, holomorphic and bihomogeneous NNs
- Utilize symmetries, Kählerity for speed and accuracy
- cymetric provides general methods that
  work on large databases of CYs
  generalize to non-Kähler setting

Outlook and open problems

- Improving all of the above!
- Compute other types of metrics (non-Kähler, SU(3), G2, gen CY, …)
- Use metric prediction in computations in physics & math
  - Compute spectra (swampland program)
    cf Ashmore 2011.13939, Ashmore-Ruehle 2103.07472, Ahmed-Ruehle 2304.00027
  - Compute gauge connections (SUSY heterotic standard models)
    cf Anderson et al 1004.4399, 1103.3041, Ashmore et al 2110.12483
  - Compute spectrum of bundle-valued diffrential forms
    cf Ashmore He Heyes Ovrut 2305.08901
  - Volumes of subcycles

- Here NN = complicated function (CY metric).
  Can we use same idea in other areas of science?