Metric Flows and CY Metrics with Infinite-Width Neural Networks

Jim Halverson

Northeastern University

Based on work in progress with F. Ruehle, 2305.xxxx.

A Specific Motivation: Neural Network Calabi-Yau Metrics

[Anderson, Gerdes, Krippendorf, Raghuram, Ruehle]
[Douglas, Lakshminarasimhan, Qi]
[Jejjala, Mayorga Pena, Mishra]

see also: [Ashmore, He, Ovrut]

“Let the neural network be the metric!”
 - above authors.

Neural network depends on parameters θ, which provide a variational ansatz:

$$g_{ij}(x) \mapsto g_{ij}(x,\theta)$$

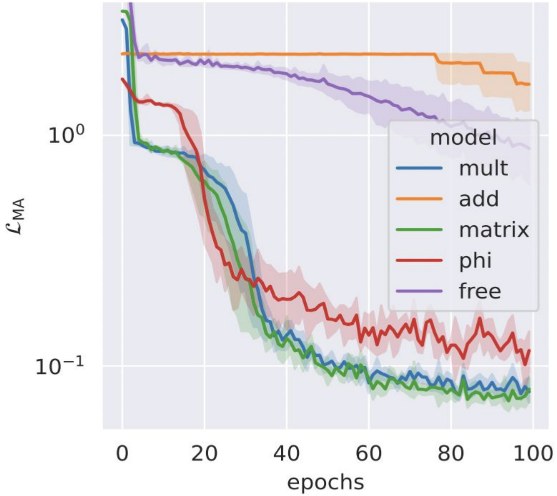
optimize parameters to minimize some objective, e.g. to drive the metric towards Ricci-flatness.

See also: NN as variational ansatz for quantum many-body wavefunction. Minimize energy, e.g.

[Carleo, Troyer] 2017 Infinite NN Context: [J.H., Luo]

Why this is a good idea: NN’s are powerful, universal approximation theorems, etc.

Evidence this is a good idea: [Larfors, Lukas, Ruehle, Schneider]



15 mins NN = 30 years w/ conventional techniques.

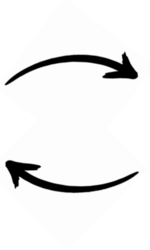
Broader Motivation: Playground for ML Modalities

- **Applied-ML-for-X, with error.**

  Typical applications you hear about.
  Sometimes error is ok.
  Other times we don't know how to avoid it.

- **ML-for-X, rigorous, no error.**
  there are examples!
  ask, e.g. smooth 4d Poincare,
  or DeepMind's knot theorem.

  How do we get rigor and understanding,
  hallmarks of math and theoretical physics,
  from techniques that are stochastic, error-prone, and blackbox?

- **ML Theory**  **Physics / Math Theory**

  This talk lives here.

Math and ML Theory Questions for NN Metrics Flows

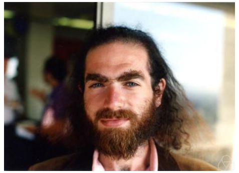
- Do there exist simplifying limits with increased understanding / tractability?

- Are those limits only simpler / more understandable,
  or also better, e.g. wrt CY metric approximation?

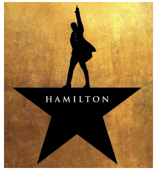
- Does increased understanding let us relate these to math literature?

- What about metric flows?

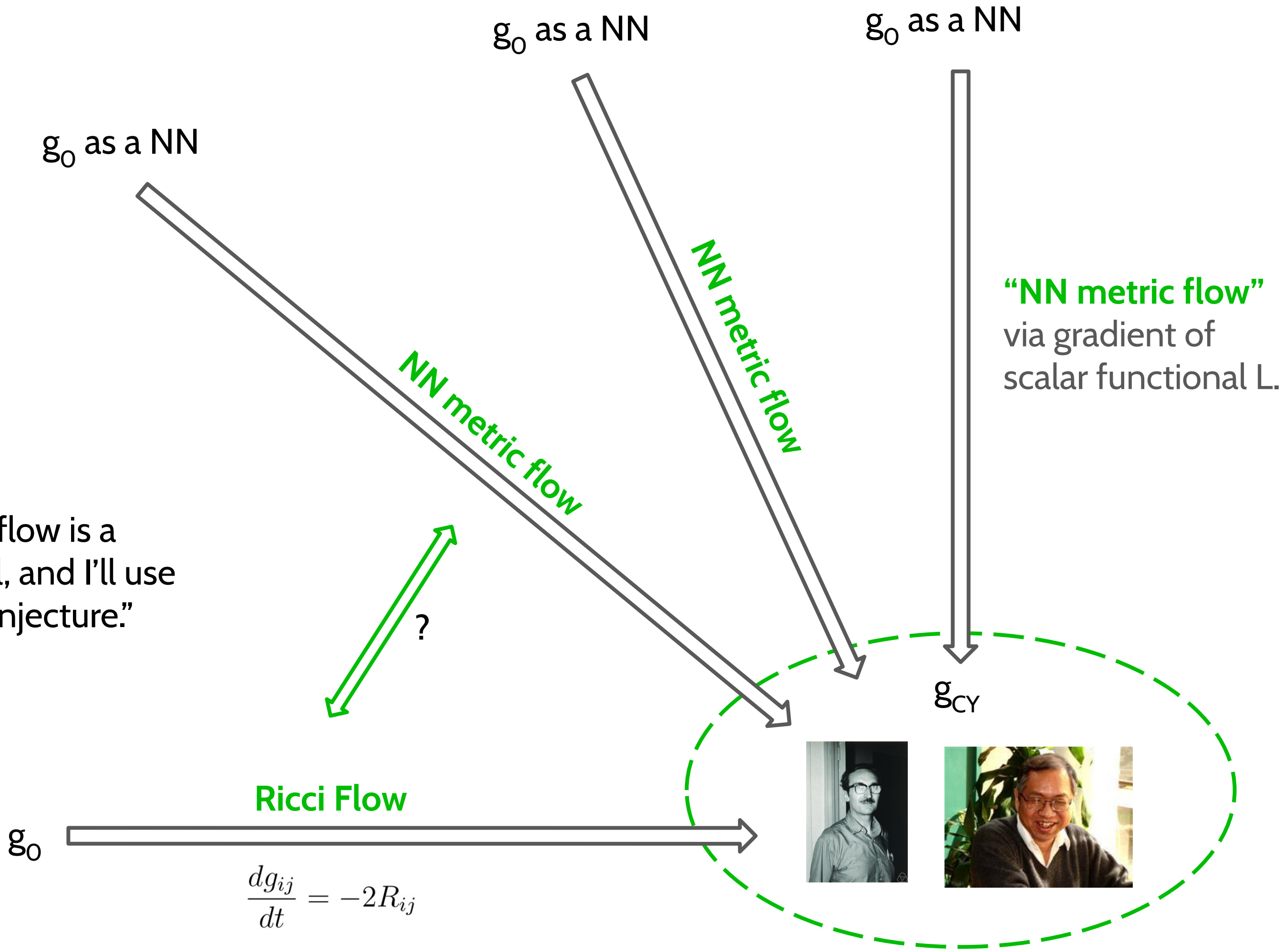
Space of Metrics



Perelman, 2000s: “Ricci flow is a gradient of a scalar functional, and I’ll use it to prove the Poincare conjecture.”



Hamilton, 1980s



g_0 as a NN

g_0 as a NN

g_0 as a NN

NN metric flow

NN metric flow

“NN metric flow” via gradient of scalar functional L.

?

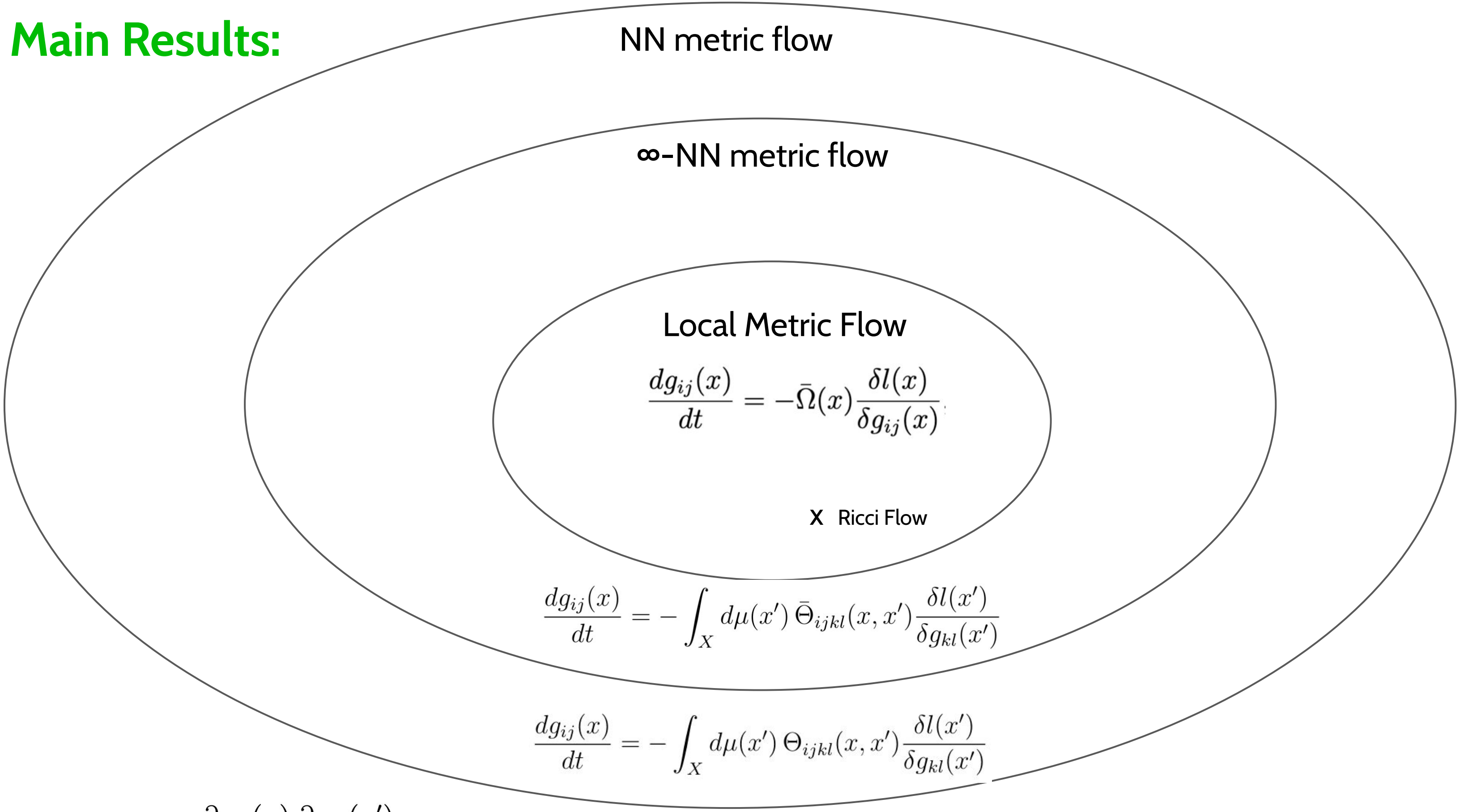
g_0

Ricci Flow

$$\frac{dg_{ij}}{dt} = -2R_{ij}$$

g_{CY}

Main Results:



NN metric flow

∞-NN metric flow

Local Metric Flow

$$\frac{dg_{ij}(x)}{dt} = -\bar{\Omega}(x)\frac{\delta l(x)}{\delta g_{ij}(x)}$$

X Ricci Flow

$$\frac{dg_{ij}(x)}{dt} = -\int_X d\mu(x')\,\bar{\Theta}_{ijkl}(x,x')\frac{\delta l(x')}{\delta g_{kl}(x')}$$

$$\frac{dg_{ij}(x)}{dt} = -\int_X d\mu(x')\,\Theta_{ijkl}(x,x')\frac{\delta l(x')}{\delta g_{kl}(x')}$$

$$\Theta_{ijkl}(x,x') := \frac{\partial g_{ij}(x)}{\partial \theta_I}\frac{\partial g_{kl}(x')}{\partial \theta_I}$$

$$\lim_{N\to\infty}\Theta_{ijkl}(x,x') = \mathbb{E}_\theta[\alpha_{ijkl}(x,x')] =: \bar{\Theta}_{ijkl}(x,x')$$

Θ_{ijkl} **difficult:** stochastic, t-dependent, hard to compute.

$\bar{\Theta}_{ijkl}$ **easier:** deterministic, t-indep, fixed function.

$\bar{\Omega}$ **easiest:** deterministic, t-indep, local

Outline

- Review: 1 Slide on Neural Networks
- Metric Flows with Neural Networks and Infinite-Width Limits
- Kahler Metric Flows and Calabi-Yau Metrics

Review: Neural Networks

Neural Networks

A neural network is a random function,
a composition of simpler functions (architecture),
with parameters drawn from some distribution.

Example: a neural network ϕ

$$\phi(x) = \frac{1}{\sqrt{N}} a_i \, \sigma(b_{ij} x_j + c_i)$$

$$a \sim P(a), \; b \sim P(b) \quad c \sim P(c) \qquad\qquad \theta = \{a, b, c\}$$

$$i = 1, \cdots, N$$

A width-N, depth-one, feedforward
network with (non-linear) activation function σ.

- **Neural networks are powerful.**

  The workhorse of the breakthroughs you've heard of in deep learning.

  Universal approximation theorems.
  Opposite of tameness?

- **Modern Neural Networks are BIG!**
  Hundreds of billions of parameters.

  e.g. chat-GPT is a “large language model” formed out of Transformers.

Metric Flows with Neural Networks

Metric Flows with Neural Networks

Q: how does the param-dependent metric change?

$$\frac{dg_{ij}(x)}{dt} = \frac{dg_{ij}(x)}{d\theta_I}\frac{d\theta_I}{dt}$$

under gradient descent, parameters update as

$$\frac{d\theta_I}{dt} = -\frac{\partial \mathcal{L}[g]}{\partial \theta_I}$$

according to scalar loss functional, which may be evaluated in the continuum or discretuum:

$$\mathcal{L}[g] = \int_X d\mu(x')\, l[g](x') \qquad \mathcal{L}[g] = \sum_{x' \in B} l[g](x')$$

Putting the pieces together, we have

$$\frac{dg_{ij}(x)}{dt} = -\int_X d\mu(x')\, \Theta_{ijkl}(x,x')\frac{\delta l(x')}{\delta g_{kl}(x')}$$

$$\frac{dg_{ij}(x)}{dt} = -\sum_{x' \in B} \Theta_{ijkl}(x,x')\frac{\delta l(x')}{\delta g_{kl}(x')}$$

in the continuous and discrete cases, where

$$\Theta_{ijkl}(x,x') := \frac{\partial g_{ij}(x)}{\partial \theta_I}\frac{\partial g_{kl}(x')}{\partial \theta_I}$$

is an object that we call the **metric neural tangent kernel** (metric-NTK).

The metric-NTK is a metric-oriented version of the NTK appearing in deep learning theory.

[Jacot, Gabriel, Hongler] NeurIPS 2018

$$\Theta_{ijkl}(x, x') := \frac{\partial g_{ij}(x)}{\partial \theta_I} \frac{\partial g_{kl}(x')}{\partial \theta_I}$$

It is a fundamental object that governs metric flows induced by neural network gradient descent.

In general, it depends heavily on initial parameters and evolves during training, during the metric flow.

The parameter sum and many chain rules make it very tedious to compute.

Simplified Flows in Infinite Limit

Deterministic NTK in Infinite Limit:

there is very often an N (e.g., N = width), such that

$$\lim_{N\to\infty}\Theta_{ijkl}(x,x') = \mathbb{E}[\alpha_{ijkl}(x,x')] =: \bar{\Theta}_{ijkl}(x,x')$$

i.e., via law-of-large-numbers, metric-NTK depends only on P(θ), not specific θ draw.

Linearized Models:

$$g^L_{ij}(x) := g_{ij}(x)\Big|_{\theta=\theta_0} + (\theta_I - \theta_{0,I})\frac{\partial g_{ij}(x)}{\partial\theta_I}\Big|_{\theta=\theta_0}$$

$$\Theta^L_{ijkl}(x,x') = \frac{\partial g_{ij}(x)}{\partial\theta_I}\Big|_{\theta=\theta_0}\frac{\partial g_{kl}(x')}{\partial\theta_I}\Big|_{\theta=\theta_0} = \Theta_{ijkl}(x,x')\Big|_{\theta=\theta_0}$$

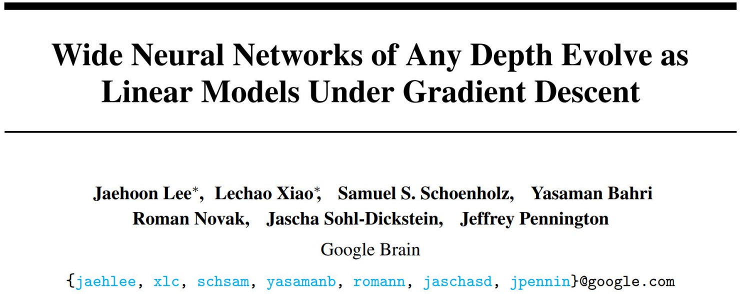
i.e., metric-NTK doesn't evolve in t.

Frozen NTK-limit:

take both deterministic limit and linearized model, gradient descent governed by a "frozen" NTK, which is a ***deterministic***, ***t-independent*** function.

Linear models?

Infinite neural nets evolve as linear models.



Wide Neural Networks of Any Depth Evolve as Linear Models Under Gradient Descent

Jaehoon Lee*, Lechao Xiao*, Samuel S. Schoenholz, Yasaman Bahri
Roman Novak, Jascha Sohl-Dickstein, Jeffrey Pennington

Google Brain

{jaehlee, xlc, schsam, yasamanb, romann, jaschasd, jpennin}@google.com

Metric Flows with Infinite Neural Networks

Recapping: in appropriate limits, our metric-NTK

$$\Theta_{ijkl}(x,x') := \frac{\partial g_{ij}(x)}{\partial \theta_I}\frac{\partial g_{kl}(x')}{\partial \theta_I}$$

becomes *deterministic* and *t-independent*, henceforth denoted as

$$\bar{\Theta}_{ijkl}(x,x')$$

a function of two variables that may be computed.

In this frozen-NTK limit, the NN metric flow is

$$\frac{dg_{ij}(x)}{dt} = -\int_X d\mu(x')\,\bar{\Theta}_{ijkl}(x,x')\frac{\delta l(x')}{\delta g_{kl}(x')}$$

the metric flow depends on *choice of architecture*, which determines the metric-NTK, and *choice of loss l*, e.g. Ricci-flatness itself, which defines the optimization.

Note: metric-NTK is a smearing function that defines g-update at x according to the loss and g at x'.

Local Metric Flows and Ricci Flow with Infinite Neural Networks

NN Metric Flow, Local Flows, and Ricci Flow

Infinite NN Metric Flow:

$$\frac{dg_{ij}(x)}{dt} = -\int_X d\mu(x')\,\bar{\Theta}_{ijkl}(x,x')\,\frac{\delta l(x')}{\delta g_{kl}(x')}$$

non-local metric updates with non-trivial component mixing, depends on arch. and l.

Ricci Flow a la Hamilton:

$$\frac{dg_{ij}(x)}{dt} = -2R_{ij}(x)$$

local metric updates, no component mixing, no scalar functional entering at all.

Ricci Flow a la Perelman:

$$\frac{dg_{ij}}{dt} = \frac{\delta \mathcal{F}[\phi, g]}{\delta g_{ij}(x)} \qquad \mathcal{F}[\phi, g] = \int_X \left(R + |\nabla\phi|^2\right)\, e^{-\phi}\, dV$$

still a local metric update w/o mixing,
but now we have a scalar functional formulation.

Local Metric Flow and Ricci Flow with Infinite NN:

get Perelman's formulation when

$$\bar{\Theta}_{ijkl}(x) = \bar{\Omega}(x)\,\delta_{ik}\delta_{jl}\,\delta(x - x')$$

$$l(x) = -\frac{\mathcal{F}[\phi, g]}{\bar{\Omega}(x)}$$

We recover Perelman's formulation of
Ricci Flow as Infinite NN gradient descent when

$$\bar{\Theta}_{ijkl}(x) = \bar{\Omega}(x)\,\delta_{ik}\delta_{jl}\,\delta(x - x')$$

$$l(x) = -\frac{\mathcal{F}[\phi, g]}{\bar{\Omega}(x)}$$

we are free to choose l(x),
but can this form of metric-NTK be realized?

i.e., are there architectures with this metric-NTK?

Architectures for Local Metric Flows and Ricci Flow

$$\bar{\Theta}_{ijkl}(x) = \bar{\Omega}(x)\,\delta_{ik}\delta_{jl}\,\delta(x - x')$$

requires 3 properties: *frozen*, *local*, and *no mixing*.

By choosing *any* architecture (see lit.) with frozen NTK limit, and choosing metric components to each be an independent NN, we achieve:

$$\bar{\Theta}_{ijkl}(x, x') = \delta_{ik}\delta_{jl}\,\bar{\Theta}(x, x')$$

a frozen, *non*-local metric-NTK without mixing.

The **non-trivial step** is to get locality!

Technical, but sufficient choices are:

1) NTK of an architecture only last layer weights evolving frozen is a NN Gaussian Process kernel K(x, x').

2) There are architectures with

$$K(x, x') \propto \exp\left(-\frac{1}{2}\frac{|x - x'|^2}{\sigma^2}\right)$$

which when normalized give the locality-inducing δ-function in the $\sigma \to 0$ limit.

Example Architectures

Cos-net: [J.H.]

$$\phi(x) = \sum_{i=1}^{N} a_i l_i(x), \qquad \ell_i(x) = \sum_j \cos(b_{ij} x_j + c_i)$$

$$a \sim \mathcal{N}(0, \frac{\sigma_a^2}{N}), \;\; b \sim \mathcal{N}(0, \frac{\sigma_b^2}{d}), \;\; c \sim \mathcal{U}[-\pi, \pi]$$

Gauss-net: [J.H., Maiti, Stoner]

$$\phi(x) = \frac{\sum_{i=1}^{N} a_i g_i(x)}{\sqrt{\exp[2(\sigma_c^2 + \sigma_b^2 x^2/d)]}} \qquad g_i(x) = \sum_j \exp(b_{ij} x_j + c)$$

$$a \sim \mathcal{N}(0, \frac{\sigma_a^2}{2N}) \;\; b \sim \mathcal{N}(0, \frac{\sigma_b^2}{d}) \;\; c \sim \mathcal{N}(0, \sigma_c^2),$$

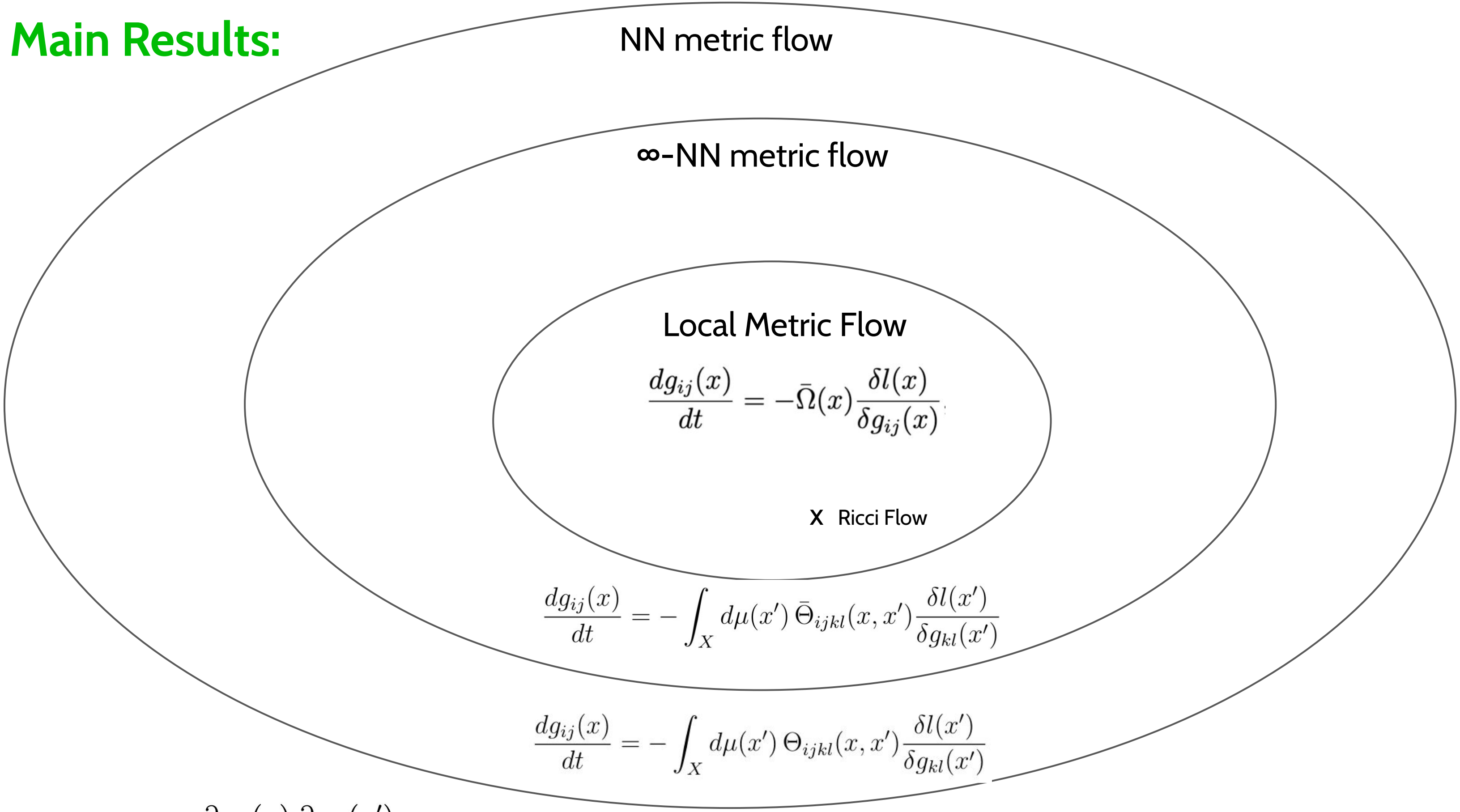
both architectures have

$$G^{(2)}(p) = \frac{\sigma_a^2}{2Z_b} e^{-\frac{1}{2}\frac{d}{\sigma_b^2} p^2}$$

which of course has Gaussian Fourier transform,
gives locality-inducing δ-function in the $\sigma \to 0$ limit.

(interestingly:
as field theories, exhibit duality at infinite-N, but at finite-N, theories and symmetries are different!)

Main Results:



NN metric flow

∞-NN metric flow

Local Metric Flow

$$\frac{dg_{ij}(x)}{dt} = -\bar{\Omega}(x)\frac{\delta l(x)}{\delta g_{ij}(x)}$$

X Ricci Flow

$$\frac{dg_{ij}(x)}{dt} = -\int_X d\mu(x')\,\bar{\Theta}_{ijkl}(x,x')\frac{\delta l(x')}{\delta g_{kl}(x')}$$

$$\frac{dg_{ij}(x)}{dt} = -\int_X d\mu(x')\,\Theta_{ijkl}(x,x')\frac{\delta l(x')}{\delta g_{kl}(x')}$$

$$\Theta_{ijkl}(x,x') := \frac{\partial g_{ij}(x)}{\partial \theta_I}\frac{\partial g_{kl}(x')}{\partial \theta_I}$$

$$\lim_{N\to\infty}\Theta_{ijkl}(x,x') = \mathbb{E}_\theta[\alpha_{ijkl}(x,x')] =: \bar{\Theta}_{ijkl}(x,x')$$

Θ_{ijkl} **difficult:** stochastic, t-dependent, hard to compute.

$\bar{\Theta}_{ijkl}$ **easier:** deterministic, t-indep, fixed function.

$\bar{\Omega}$ **easiest:** deterministic, t-indep, local

Kahler Metric Flows and Calabi-Yau Metrics

If we want numerical CY metrics via infinite-width and NTK,
it’s useful to specify the story further.

Kahler Potential Flows

Naive Kahler Potential Flow:

$$\frac{dK(z,\bar{z})}{dt} = -\int_X d\mu(z',\bar{z}')\;\Theta(z,\bar{z},z',\bar{z}')\,\frac{\delta l(z',\bar{z}')}{\delta\,K(z',\bar{z}')}$$

$$\Theta(z,\bar{z},z',\bar{z}') = \frac{dK(z,\bar{z})}{d\theta_I}\,\frac{dK(z,\bar{z})}{d\theta_I}$$

Kahler Potential Flow

$$\frac{dK(z,\bar{z})}{dt} = -\int_X d\mu(z',\bar{z}')\;\Theta_{i\bar{j}}(z,\bar{z},z',\bar{z}')\,\frac{\delta l(z',\bar{z}')}{\delta\,\partial_i\bar{\partial}_{\bar{j}}K(z',\bar{z}')}$$

where we have varied with respect to the Kahler metric to avoid Kahler transformation reduns.

Relationship between better and naive NTK
makes the former tractable.

$$\Theta_{i\bar{j}}(z,\bar{z},z',\bar{z}') = \frac{dK(z,\bar{z})}{d\theta_I}\,\frac{d\,\partial_i\bar{\partial}_{\bar{j}}K(z',\bar{z}')}{d\theta_I} = \partial_i\bar{\partial}_{\bar{j}}\Theta(z,\bar{z},z',\bar{z}')$$

Both can have frozen NTK limits.

Kahler Metric Flows

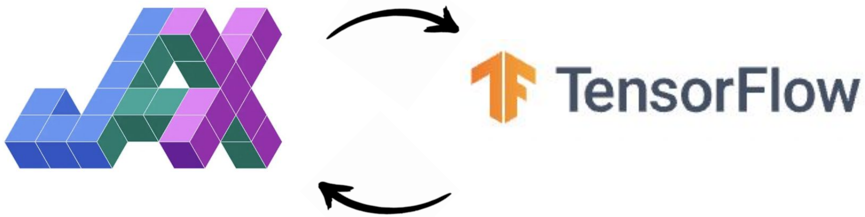
$$\frac{dg_{i\bar{j}}(z,\bar{z})}{dt} = -\int_X d\mu(z',\bar{z}') \; \left[\Theta_{i\bar{j}k\bar{l}}(z,\bar{z},z',\bar{z}') \; \frac{\delta l[g](z',\bar{z}')}{\delta g_{k\bar{l}}(z',\bar{z}')}\right]$$

$$\Theta_{i\bar{j}k\bar{l}}(z,\bar{z},z',\bar{z}') = \frac{dg_{i\bar{j}}(z,\bar{z})}{d\theta_I}\frac{dg_{k\bar{l}}(z',\bar{z}')}{d\theta_I}$$

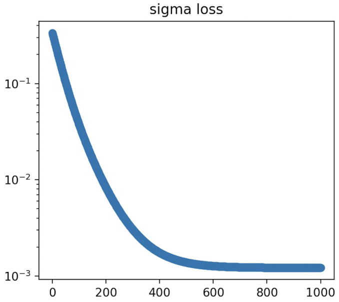
Implementation: Infinite-Width Metric Flows to CY

- explicitly compute metric updates from
  our kernel regression equations,
  no NN parameters b/c of infinite-width limit.

- use great ML packages from Google teams,
  neural-tangents, JAX, TensorFlow.
  and cymetric from our community.



Preliminary result:



$$l_\sigma = \left(1 - \frac{J^3}{\Omega \wedge \bar{\Omega}}\right)$$

Simple metric flow with Gaussian kernel,
infinite width limit. 10000 pts on the Fermat quintic.

Caveat: this is train loss from last week,
currently overtraining, stay tuned!s

Interesting TBD: better or worse than finite NN?
Interesting ML Q related to **large-scale Google study.**

[Lee et. al.]

Conclusions

I’ve explained how ML theory helps us understand and
characterize NN metric flows by opening up
a new duality frame at infinite width.

I’ve specialized to Kahler metrics,
for the purpose of approximating CY metrics at infinite width,
and showed preliminary results.

Interested? Here are seminars at this interface.

Institute for Artificial Intelligence and Fundamental Interactions (IAIFI)

one of five original NSF AI research institutes, this one at the interface with physics!

MIT, Northeastern, Harvard, Tufts.

ML for physics / math discoveries?
Can physics / math help ML?

Sign up for our mailing list: www.iaifi.org.

Physics ∩ ML

Physics Meets ML

virtual seminar series, "continuation" of 2019 meeting at Microsoft Research.

Bi-weekly seminars from physicists and CS, academia and industry.

Organizers: Bahri (Google), Krippendorf (LMU Munich), J.H., Paganini (DeepMind), Ruehle (CERN), Shiu (Madison), Yang (MSR)

Sign up at www.physicsmeetsml.org.

Interested? Upcoming Meetings at this Interface

- **IAIFI Summer School 2023**
  ML + Physics education for students, postdocs, and faculty.

- **IAIFI Summer Workshop 2023**
  great talks across physics, industry and academia.

- **Mathematics and Machine Learning 2023** Caltech, December 10-12, 2023.
- **String Data 2023** Caltech, December 13-15, 2023.

Thanks!

Questions?

Or get in touch after:

e-mail: jhh@neu.edu
Twitter: @jhhalverson
web: www.jhhalverson.com

Ricci Flow

Hamilton’s Formulation:

$$\frac{dg_{ij}}{dt} = -2R_{ij}$$

In the 1980’s, Richard Hamilton introduced a flow in the space of metrics known as Ricci flow.

Proved numerous critical theorems, e.g. uniqueness and existence for arbitrary init. cond.

Perelman’s Formulation:

$$\frac{dg_{ij}}{dt} = \frac{\delta \mathcal{F}[\phi, g]}{\delta g_{ij}(x)} = -2[R_{ij} + \nabla_i \nabla_j \phi]$$

$$\mathcal{F}[\phi, g] = \int_X \left(R + |\nabla \phi|^2\right) e^{-\phi}\, dV$$

In the 2000’s, Perelman famously showed that Ricci flow is a gradient of a scalar functional, via t-dependent diff. relating above to Hamilton’s.

Introduced “Ricci flow with Surgery” to deal with singularities, prove Poincare conjecture.

A Generalized Ricci-Flow

$$\bar{\Theta}_{ijkl}(x) = \bar{\Omega}(x)\,\delta_{ik}\delta_{jl}\,\delta(x-x')$$

recall 3 properties: *frozen*, *local*, and *no mixing*, and that it was locality that was harder.

Recall that frozen and no-mixing gives us:

$$\bar{\Theta}_{ijkl}(x,x') = \delta_{ik}\delta_{jl}\,\bar{\Theta}(x,x')$$

and that our in our examples

$$\bar{\Theta}(x,x') \propto \exp\left(-\frac{1}{2}\frac{|x-x'|^2}{\sigma^2}\right)$$

gives the locality-inducing δ-function as $\sigma \to 0$.

Generalized Ricci Flow:
Take σ small but finite,
sets scale of non-locality for the generalized Ricci flow that may be made parametrically small.

Examples: (Cos-net, e.g.)

$$\phi(x) = \sum_{i=1}^{N} a_i l_i(x), \qquad \ell_i(x) = \sum_j \cos(b_{ij}x_j + c_i)$$

$$a \sim \mathcal{N}(0, \frac{\sigma_a^2}{N}), \;\; b \sim \mathcal{N}(0, \frac{\sigma_b^2}{d}), \;\; c \sim \mathcal{U}[-\pi,\pi]$$

σ small limit is σ_b large limit.

Implementation: easy to take into account this generalized Ricci flow.

Remarks: Algebraic Geometry ∩ NTK

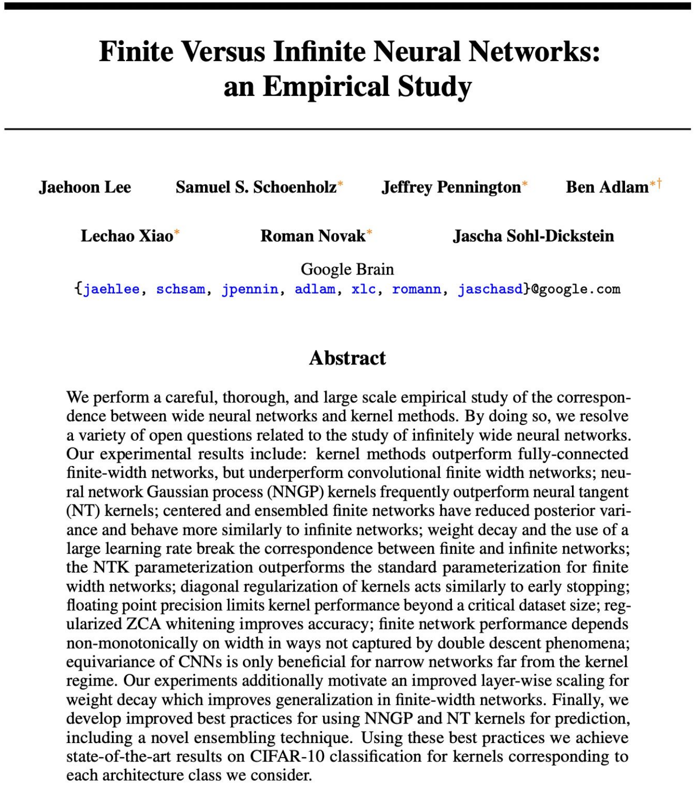
Upshot: we cast the story of *NN reps of CY metrics*, and especially their learning process,
in the language of *∞-NN and NTK theory*.

This introduces a connection between algebraic geometry and deep learning theory.

Interesting connections are immediate by simple thought and relating to literature.

- flat directions of loss function
  = CY moduli space,
  embedded in infinite dim. parameter space.
- ∞-NN trained with mean square error
  will memorize (learn perfectly) target metric.
- develop theory of NTK spectrum to facilitate learning CY metrics? (c.f. computer vision).

Finite vs. Infinite: Large-Scale Study by Google



Finite Versus Infinite Neural Networks: an Empirical Study

Jaehoon Lee Samuel S. Schoenholz[*] Jeffrey Pennington[*] Ben Adlam[*†]

Lechao Xiao[*] Roman Novak[*] Jascha Sohl-Dickstein

Google Brain
{jaehlee, schsam, jpennin, adlam, xlc, romann, jaschasd}@google.com

Abstract

We perform a careful, thorough, and large scale empirical study of the correspondence between wide neural networks and kernel methods. By doing so, we resolve a variety of open questions related to the study of infinitely wide neural networks. Our experimental results include: kernel methods outperform fully-connected finite-width networks, but underperform convolutional finite width networks; neural network Gaussian process (NNGP) kernels frequently outperform neural tangent (NT) kernels; centered and ensembled finite networks have reduced posterior variance and behave more similarly to infinite networks; weight decay and the use of a large learning rate break the correspondence between finite and infinite networks; the NTK parameterization outperforms the standard parameterization for finite width networks; diagonal regularization of kernels acts similarly to early stopping; floating point precision limits kernel performance beyond a critical dataset size; regularized ZCA whitening improves accuracy; finite network performance depends non-monotonically on width in ways not captured by double descent phenomena; equivariance of CNNs is only beneficial for narrow networks far from the kernel regime. Our experiments additionally motivate an improved layer-wise scaling for weight decay which improves generalization in finite-width networks. Finally, we develop improved best practices for using NNGP and NT kernels for prediction, including a novel ensembling technique. Using these best practices we achieve state-of-the-art results on CIFAR-10 classification for kernels corresponding to each architecture class we consider.