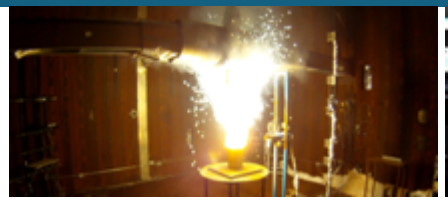
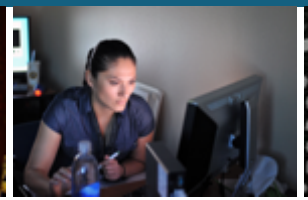
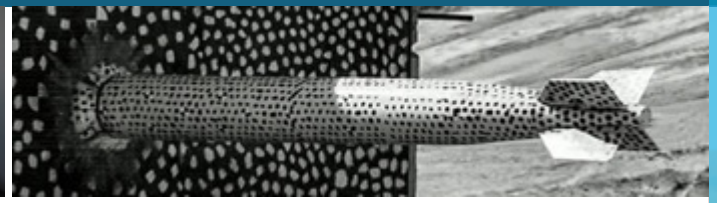
Analysis of Neural Networks as Random Dynamical Systems

Project # 21-0528







PI: Khachik Sargsyan, org. 8351
PM: Janine Bennett, org. 8759

Team: Joshua Hudson (8351),
Marta D'Elia (8754), Habib Najm (8300)

Continuation review: June 3, 2021

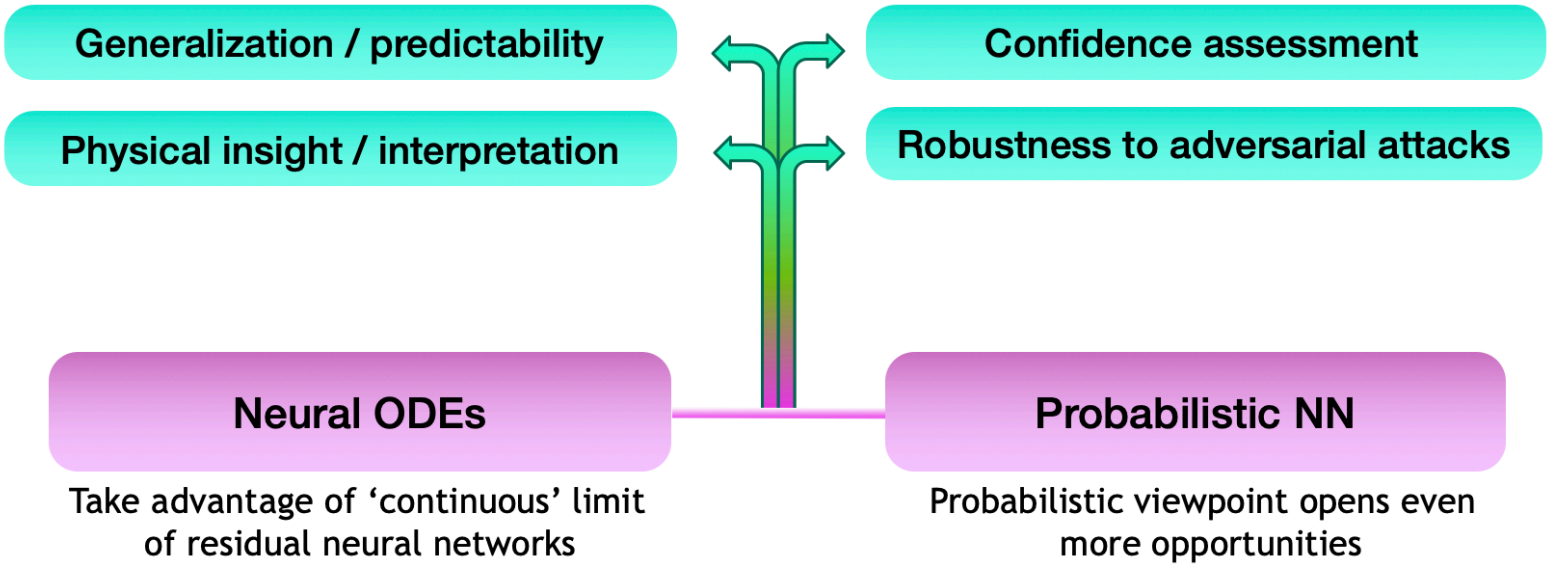
FY21-24, $500K/yr

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

IDEA SUMMARY: Overview

No deviations from the plan

What: Analyze the performance of neural networks (NNs)
[training, generalization, predictive confidence] from dynamical and probabilistic viewpoints.



Current work: Despite all the success, there are many recognized challenges and unknowns in NN behavior.

Why now: There is a lot of accumulated knowledge from ODE and UQ;
prime time to build on these insights

TECHNICAL ACCOMPLISHMENTS: OVERVIEW

Task 0.0 — Ground work done: we have PyTorch codes developed from scratch for both ResNet (discrete) and Neural ODE (continuous)

Task 1.1 — Dynamical analysis, stiffness penalization

Task 1.2 — Weight parameterization by depth

Task 1.3 — Bayesian inference of weights: both Markov chain Monte Carlo and variational inference

Y1

Y2

Task 2.3 — Integral NODEs, nonlocal kernel networks

TECHNICAL ACCOMPLISHMENTS: OVERVIEW

		Task (T) / Milestone(M)	Period
FY21	T1.1	Dynamical analysis in deterministic setting†	10/2020-06/2021
	T1.2	Development of deterministic, sparse weight representations	01/2021-09/2021
	M1	Demonstrate reduced NODE in deterministic setting	by 09/2021
	T1.3	Formulation of Bayesian inference of weights	04/2021-09/2021

Task 0.0

Ground work done: we have PyTorch codes developed from scratch for both ResNet (discrete) and Neural ODE (continuous)

Task 1.1

- Dynamical analysis
- Stiffness penalization

Task 1.2

- Weight parameterization by depth

Task 1.3

- Bayesian inference of weights
- Both Markov chain Monte Carlo and variational inference

M 1

From next FY: Task 2.3 Integral NODEs

TA: We have started in-house codes for both discrete and continuous cases

Task 0.0

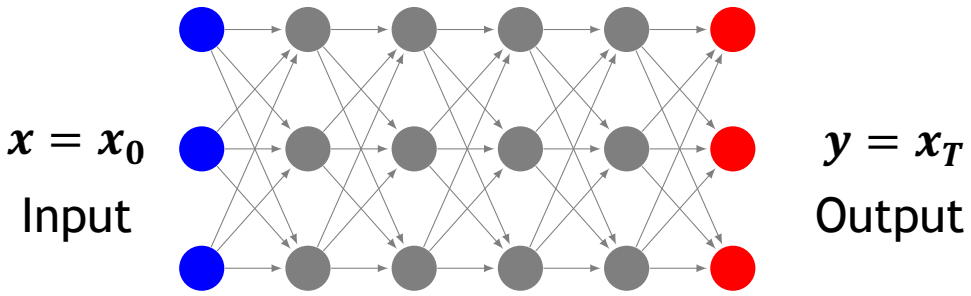
ResNet (discrete)

$$\begin{cases} \boldsymbol{x_1 = x + \alpha_0 \sigma(W_0 x_0 + b_0)} \\ \vdots \\ \boldsymbol{x_{n+1} = x_n + \alpha_n \sigma(W_n x_n + b_n)} \\ \vdots \\ \boldsymbol{y = x_{L-1} + \alpha_{L-1} \sigma(W_{L-1} x_{L-1} + b_{L-1})} \end{cases}$$

Neural ODE (continuous)

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{\sigma}(\boldsymbol{W}(t)\boldsymbol{x} + \boldsymbol{b}(t))$$

$$\boldsymbol{x}(0) = \boldsymbol{x} \qquad \boldsymbol{x}(T) = \boldsymbol{y}$$



Build-up complexity

Linear	$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{W}(t)\boldsymbol{x} + \boldsymbol{b}(t)$
No-bias	$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{W}(t)\boldsymbol{x}$
Const w	$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{\sigma}(\boldsymbol{W}\,\boldsymbol{x} + \boldsymbol{b})$

TA: Analytical work done to establishing equivalence

Task 0.0

ResNet

$$\boldsymbol{x_{n+1} = x_n + \alpha_n \sigma(W_n x_n + b_n)}$$

NODE

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{\sigma(W(t)x + b(t))}$$

Forward equivalence:

Neural ODE discretized using explicit Euler and ResNet produce identical outputs choosing time step: $\Delta t = \frac{T}{L}$, $\alpha_n := \Delta t$, $\boldsymbol{W_n} := \boldsymbol{W}(n\Delta t)$ and $\boldsymbol{b_n} := \boldsymbol{b}(n\Delta t)$ for all n.

Backward not so much:

Gradient computations differ!

Consider $W(t) \equiv W$ and $b \equiv 0$:

Discretized Neural ODE with adjoint method:

$$\nabla loss = 2\big((1 + \delta t\, W)^L x - y\big)(1 + \delta t\, W)^{L} x$$

ResNet with backpropagation:

$$\nabla loss = 2\big((1 + \delta t\, W)^L\, x - y\big)(1 + \delta t\, W)^{L-1} x$$

- Gradients converge as $L \to \infty$ but differences can be large for small L,
- Optimize then discretize (adjoint method) ≠ discretize then optimize (backpropagation).

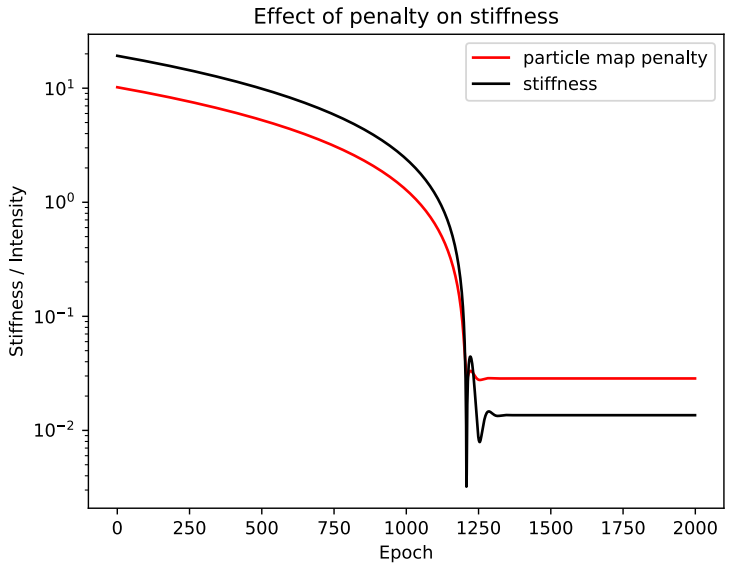
TA: Regularization path 1: We developed a Particle Kernel Map as a stiffness surrogate

Task 1.1

- A kernel density evaluation to 'interpolate' the stiffness metric

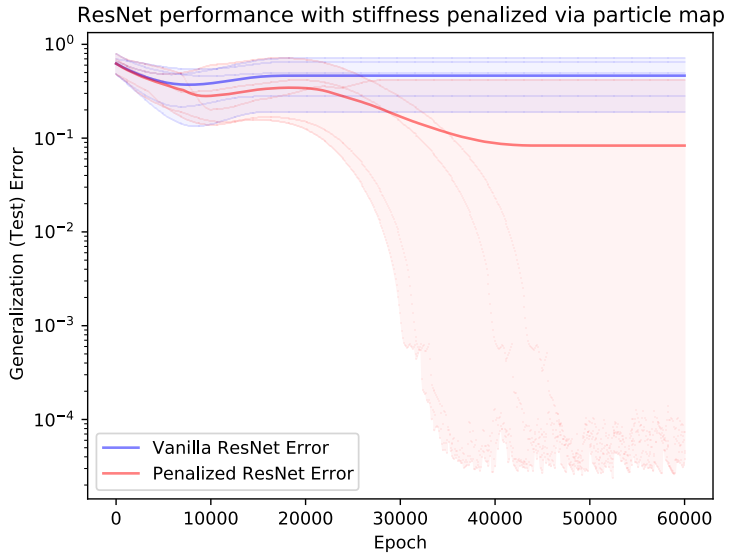
$$S(W) \approx \sum_p S\left(M_p\right) e^{-1/\gamma^2 \|W - M_p\|^2}$$

estimated — known



- Sum of kernel contributions form a differentiable approximation for stiffness of target weight

- List of particles is updated at every optimization step with nearest neighbors of the current weight matrix



- More work needed to make it scalable to high dimensional problems

TA: Regularization path 1: penalize for stiffness of weight matrices

Task 1.1

- Measure of stiffness: $\max_i|\lambda_i| - \min_i|\lambda_i|$
  - For non-symmetric weights, eigenvalue computation is not readily differentiable with PyTorch
  - Explicitly enforcing symmetry on weights may reduce expressiveness (reduction is severe in simple one layer case)
  - **New development**, thanks to Kyungjoo Kim (org. 1465):
    We have *Sacado* code for computation and automatic differentiation of the largest-magnitude e-value
    - relies on *Kokkos*-based math library *Tines*
    - use active time scale, estimated from the ODE, as a reference, instead of the min e-value

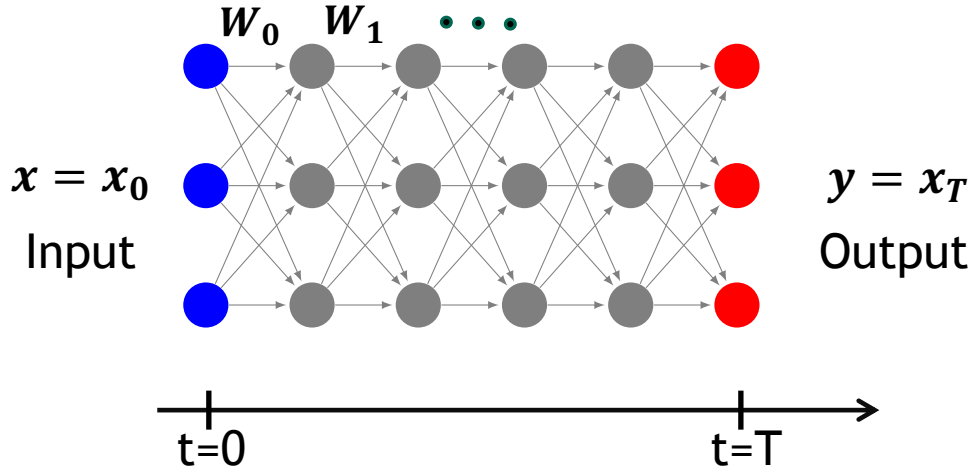
- We experimented with other proxies for stiffness
  - via singular values $\sigma_1 > \cdots > \sigma_k > 0$
  - e.g. $\sigma_1 - \sigma_k$ ($\sigma_1 > \max_i|\lambda_i|, \min_i|\lambda_i| > \sigma_k$)
    - Experiments indicate may be too loose to provide a tight control on stiffness.

TA: Regularization path 2: weight parameterization allows dimensionality reduction

Task 1.2

$\boldsymbol{W}(nT/L;\boldsymbol{\alpha})$ $\boldsymbol{b}(nT/L;\boldsymbol{\alpha})$

ResNet: $\boldsymbol{x_{n+1}} = \boldsymbol{x_n} + \boldsymbol{\alpha_n}\boldsymbol{\sigma}(\boldsymbol{W_n}\boldsymbol{x_n} + \boldsymbol{b_n})$



NODE: $\frac{d\boldsymbol{x}}{dt} = \boldsymbol{\sigma}(\boldsymbol{W}(t;\boldsymbol{\alpha})\boldsymbol{x} + \boldsymbol{b}(t;\boldsymbol{\alpha}))$

Instead of training for weight matrices $\boldsymbol{W_0}, \boldsymbol{W_1}, \ldots$ parameterize $\boldsymbol{W}(t;\boldsymbol{\alpha})$ and train for $\boldsymbol{\alpha}$'s.

For example:

- Constant parameterization: $\boldsymbol{W}(t;\boldsymbol{\alpha}) = \boldsymbol{\alpha}$
- Linear parameterization: $\boldsymbol{W}(t;\boldsymbol{\alpha}) = \boldsymbol{\alpha}t + \boldsymbol{\beta}$

…

- Business-as-usual: $\boldsymbol{W}(t;\boldsymbol{\alpha}) = \boldsymbol{W_{tL/T}}$

Pending work:

- Analysis of parameterized network capacity and dimensionality metrics
- How does the weight parameterization affect loss surface?

TA: Regularization path 3: We developed PyTorch wrapper codes for Bayesian ResNets and Neural ODEs

Task 1.3

Major targets next year rely on probabilistic reformulation

```
class MCMCRegr():
    def __init__(self, nnmodule, nmcmc=10000, verbose=False):
        self.nnmodel = nnmodule
        self.verbose = verbose
        self.nens = nens

        if self.verbose:
```

- Markov chain Monte Carlo inference of NN weights: usually infeasible, but can be done with weight reparameterization!

```
class VIRegr():
    def __init__(self, nnmodule, verbose=False):
        self.bmodel = BNet(nnmodule)
        self.verbose = verbose

        if self.verbose:
```

- Variational approximation as an alternative for high-dimensional and complex loss surfaces

TA: Integral (resolution-independent) NODEs: Nonlocal kernel networks for PDE learning & image classification

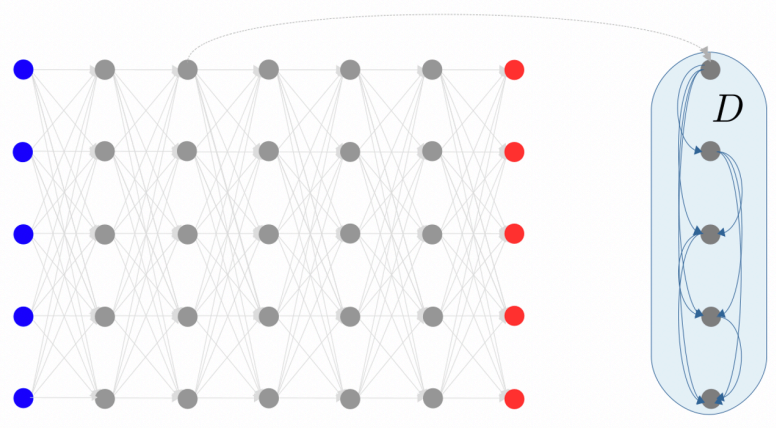
Task 2.3

$$\frac{d\mathbf{h}}{dt} = \sigma(W_t\mathbf{h}_t + \beta_t)$$

NODE

$$\frac{\partial \mathbf{h}(\mathbf{x},t)}{\partial t} = \int_D k_{NN}(\mathbf{x},\mathbf{y};\mathbf{v})(\mathbf{h}(\mathbf{y},t) - \mathbf{h}(\mathbf{x},t))d\mathbf{y} - R_{NN}(\mathbf{x})\mathbf{h}(\mathbf{x},t)$$

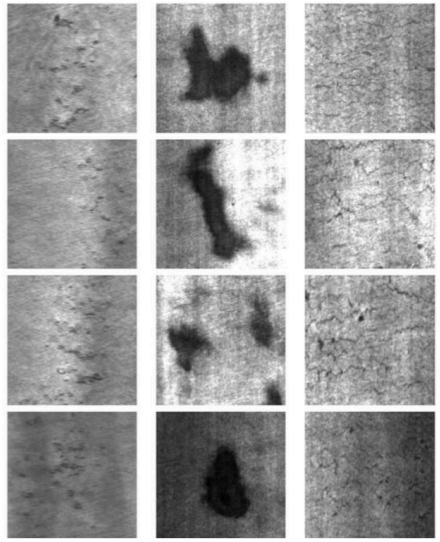
Nonlocal kernel network, NKN



Nonlocal diffusion-reaction equation whose **stability** follows from integral calculus

- **generalization with respect to width**, i.e. resolution.
- ability to **capture interactions in the feature space**, i.e. node-to-node interactions – useful in image proc. and PDE learning

Application in material design and anomaly detection



*You, Yu, **D'Elia**, Gao, Silling,*
Nonlocal Kernel Network (NKN): a Stable and Resolution-Independent Deep Neural Network.
Submitted to NeurIPS 2021

PROJECT STATUS: No adjustments needed

Technical: we are on target (and more!) to meet the milestone, all tasks good progress

		Task (T) / Milestone(M)	Period
FY21	T1.1	Dynamical analysis in deterministic setting†	10/2020-06/2021
	T1.2	Development of deterministic, sparse weight representations	01/2021-09/2021
	M1	Demonstrate reduced NODE in deterministic setting	by 09/2021
	T1.3	Formulation of Bayesian inference of weights	04/2021-09/2021

Programmatic / Spend plan:

65 %
% of Project Costed

69%
% of FY Elapsed

- Hired Joshua Hudson in November 2020 via CSRI postdoc position announcement, and received the promised funds bump-up
- Hiring challenge: need a second postdoc. A few interviews. A couple of close calls. One offer pending now.
- The team has increased the effort to advance the tasks while searching for a second postdoc

PROJECT PLAN FOR REMAINDER OF FY21 and FY22

		Task (T) / Milestone(M)	Period
FY21	T1.1	Dynamical analysis in deterministic setting†	10/2020-06/2021
	T1.2	Development of deterministic, sparse weight representations	01/2021-09/2021
	M1	Demonstrate reduced NODE in deterministic setting	by 09/2021
	T1.3	Formulation of Bayesian inference of weights	04/2021-09/2021

Work to do on all fronts

Regularization/NN reduction:

- Dynamical analysis/stiffness → E-value autodiff w Sacado
- Weight parameterization → Incorporate sparsity
- Probabilistic NODEs → Posterior/loss analysis
- Integral/fractional NODEs → Formulate probabilistically

FY22	T2.1	Formulation of stability conditions under uncertainty†	10/2021-03/2022
	T2.2	Extension of dynamical analysis under uncertainty	10/2021-06/2022
	T2.3	Formulation of fractional PNODE construction*	01/2022-09/2022
	T2.4	Regularization via weight representations in PNODEs	01/2022-06/2022
	T2.5	Inference/training of weight functions in PNODEs	04/2022-09/2022
	M2	Demonstrate training of regularized PNODEs	by 09/2022

ANTICIPATED OUTPUTS AND TIE TO INVESTMENT AREA CALL

Output: tools

- Dynamical Analysis
- Regularization
- Stability

NN Challenges

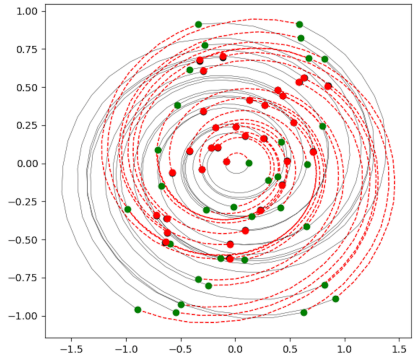
- Generalization / predictability
- Confidence assessment
- Physical insight / interpretation
- Robustness to adversarial attacks

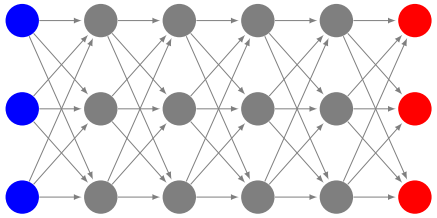
- **Main output:** fundamental mathematical analysis tools to impact the four challenges above
- All five areas under **Trusted AI** strategic initiative are relevant
- Strong potential to mark our territory in ML/UQ landscape
- Targeting both SIAM conferences and ML conferences
- 1 paper submitted to NeurIPS. We expect more beginning this Fall

Analysis of Neural Networks as Random Dynamical Systems
PI: Khachik Sargsyan (8351) PM: Janine Bennett (8759)

Project Goal(s)

- Develop methods for analysis and regularization of neural networks
- Merge probabilistic and ODE viewpoints to improve NN performance
- Demonstrate on exemplar applications: climate, material science





FY21 Technical Milestones

All tasks are in progress and on track

- T1.1 Dyn. analysis in deterministic setting
- T1.2 Develop sparse weight representations
- M1 Demo reduced NODE in determ. setting
- T1.3 Formulate Bayesian inference of weights

Mission Impact

- Bring together theory, modeling, computation, and data, under potentially noisy and adversarial conditions
- Improved NN performance can be key to many mission apps
- Probabilistic NODEs will be a unique capability and will remain mission-relevant for years to come

Transition Plan

- Unique and risky capability: if successful, can lead to further ASCR funding
- Follow-on funding (BER, FES) for applications of interest highly likely
- Software not a direct target but a bi-product which will serve us well for future funding

Extra Materials

Analysis of Neural Networks as Random Dynamical Systems

FY21-0528: K. Sargsyan (8351), J. Bennett (8759), 3Yr, Total $1600K

Prior Work

First ingredient: Neural ODEs

- **Neural ODEs** been around a while (few papers in 90's), but revived in ML community recently
  - *Chen, Duvenaud, 2018+*: clever trick with adjoints; *Ruthotto et al, 2018+*: more fundamental, discovery; *Weinan E, 2017*: dynamical system context; training as control
- Many extensions followed
  - SDE context [*Liu et al, 2019; Tzen et al, 2019*]; PDE context: [*Ruthotto et al, 2018; Long et al, 2018*]
  - Inspires new NN architectures [*Lu et al, 2018*]
  - Fractional/nonlocal DNN [*Antil, 2020; Pang, 2020; D'Elia, 2020*]
- Challenges with NODEs, active area of research: we will inherit some of the issues, but also enhancements
- Not restricted to dynamical, time-resolved physical models
- Good balance of optimism and skepticism in literature

Analysis of Neural Networks as Random Dynamical Systems

FY21-0528: K. Sargsyan (8351), J. Bennett (8759), 3Yr, Total $1600K

Prior Work

Second ingredient: Probabilistic formulation

- **Probabilistic NN** have been around since 90s [*MacKay, 1992; Neal, 1997*]
  - Full probabilistic treatment was infeasible back then (and still is, generally)
  - Recent work showed avenues via variational methods with clever tricks:

    Bayes by Backprop [*Blundell, 2015*]; Probabilistic backprop [*Hernandez-Lobato 2015*]
  - *Ghahramani, "Probabilistic Machine Learning and Artificial Intelligence". Nature, 2015*
    - *"Nearly all approaches to probabilistic programming are Bayesian since*

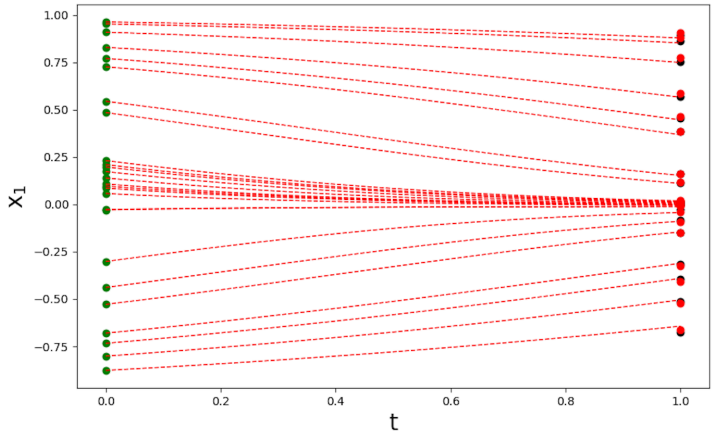
      it is hard to create other coherent frameworks for automated reasoning about uncertainty"
  - Industry ***is*** catching up: Bayesflow at Google, infer.NET at Microsoft, Uber has shown interest
  - Still not industry-standard: expensive, not well understood.

TA: Simple 1d and 2d demos highlight challenges ahead

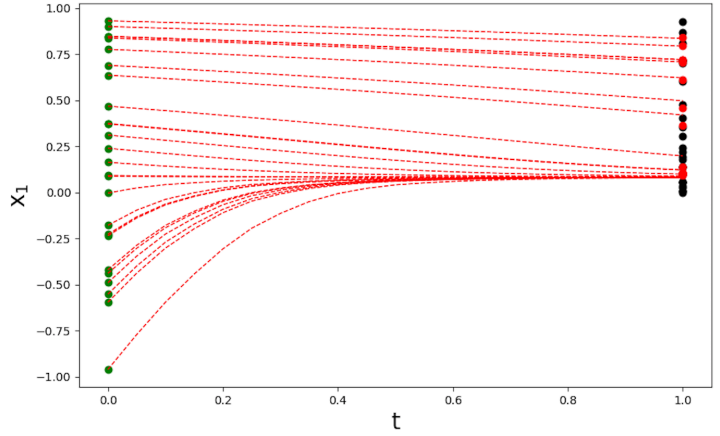
Task 0.0

Path crossing issue,
best seen in 1d

Function $y = x^3$



Function $y = x^2$



Linear, autonomous NODE $\frac{d\boldsymbol{x}}{dt} = \boldsymbol{W}\boldsymbol{x}$

Exact solution $\boldsymbol{x}(T) = \boldsymbol{x}(0)e^{\boldsymbol{W}T}$

Matrix logarithm is not unique!

$e^{\boldsymbol{W_1}} = e^{\boldsymbol{W_2}}$ but $\boldsymbol{W_1} \neq \boldsymbol{W_2}$