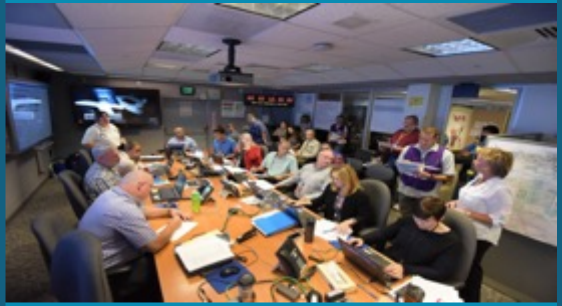
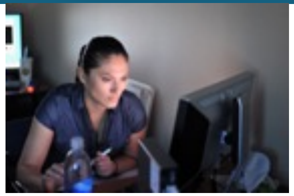
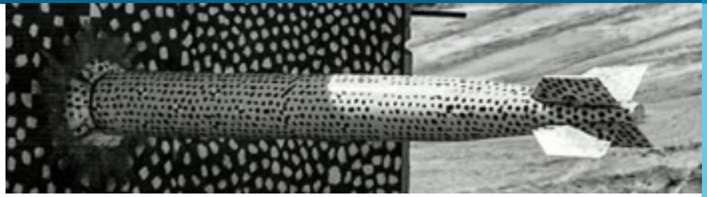
Quantifying Uncertainties in Residual Neural Networks and Neural ODEs



Khachik Sargsyan (Sandia National Laboratories, CA, USA)

Sandia-CA : Joshua Hudson, Oscar Diaz-Ibarra, Habib Najm
Pasteur Labs/Stanford U. : Marta D'Elia
Emory Univ. : Lars Ruthotto, Haley Rosso

June 12, 2023

UNCECOMP23, Athens, Greece

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.

Road to Trustworthy SciML

- Uncertainty quantification for NN
  - state of the art and challenges

Probabilistic NN

Confidence assessment

- How Residual NNs (ResNets) make UQ-for-NNs more tractable
  - weight-parameterization inspired by Neural ODE analogy

Neural ODEs / ResNets

Generalization

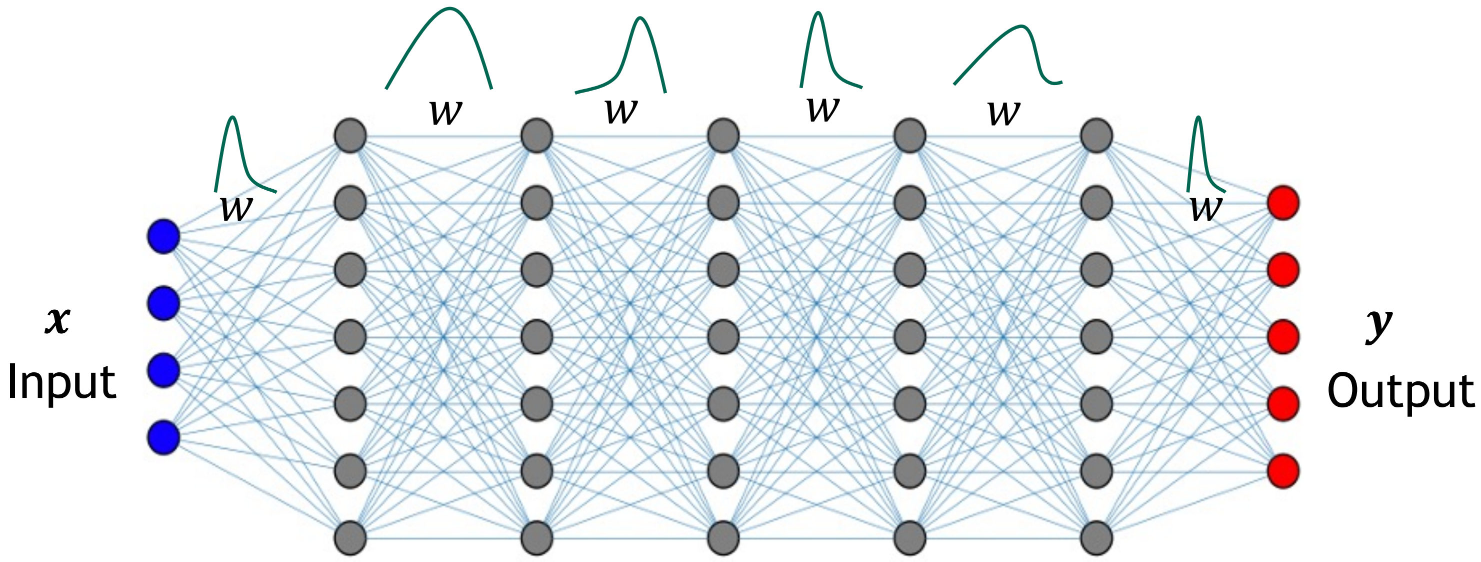
Probabilistic NN aka Bayesian NN

- *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"*

- Bayesian NN methods have been around since 90s [*MacKay, 1992*; *Neal, 1996*]
  - Full Bayesian treatment was infeasible back then….
    - … and still is, generally, not industry-standard by any means

UQ-for-NN: state of the art

- **True Bayesian:** Sampling methods with true posterior distribution



Posterior   Prior

$$p(w|y) \propto p(y|w)\, p(w)$$

Likelihood

$$\propto \exp\left(-\frac{||y - f_w(x)||^2}{2\sigma^2}\right)$$

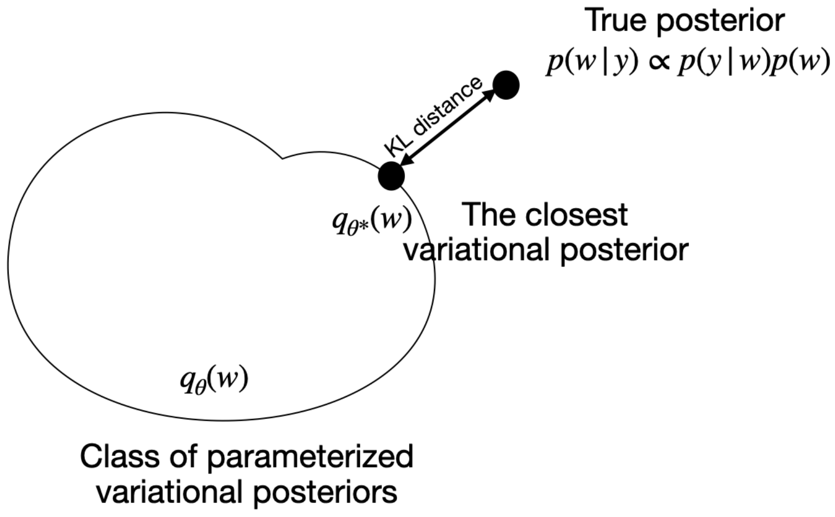
Negative log-posterior ⟷ Deterministic loss function

✓ Markov chain Monte Carlo (MCMC) sampling of posterior; Hamiltonian MC *[Levy, 2018]*
  ❑ Tuning is an art: essentially infeasible outside academic examples

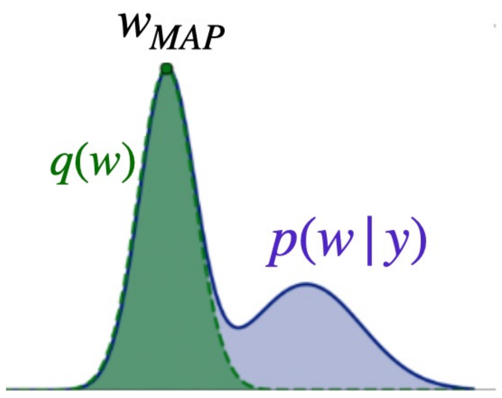
UQ-for-NN: state of the art, cont-d

- **Approximate Bayesian:**

  ✓Variational inference, many flavors;
  Bayes by Backprop [*Blundell, 2015*]
  Probabilistic backprop [*Hernandez-Lobato 2015*]
  SVI, BBVI, ADVI, ….
  - ❑Typically underestimates predictive uncertainty
  - ❑Restricted to variational class



  ✓Laplace approximation *[Daxberger, 2021]*
  - ❑Good only locally, fails to explore the full posterior



UQ-for-NN: state of the art, cont-d

- **Ensembling methods:** work surprisingly well!
  - ✓ Deep Ensembles *[Lakshminarayanan, 2017]*
  - ✓ Randomized MAP Sampling *[Pearce, 2020]*
  - ✓ MC-Dropout *[Gal, 2015]*
  - ✓ Stochastic Weight Averaging – Gaussian (SWAG) *[Maddox, 2019]*
  - ❑ Little theoretical backing
  - ❑ Too expensive, albeit parallelizable
  - ❑ Lots of recent work interpreting these from Bayesian persepective

- **Direct learning of predictive RV**
  - ✓ Delta-UQ *[Anirudh, 2021]*
  - ✓ Conformal UQ *[Hu, 2022]*
  - ✓ Information-bottleneck UQ *[Guo, 2023]*
  - ✓ ….

Bayesian UQ-for-NN: showstoppers

- Complicated posterior distribution (loss surface):
  invariances, multimodality, ‘ridges’

- Large number of weights:
  scales linearly with depth and quadratically with width

- Prior on weights hard to elicit/interpret/defend

Main message of the talk:

work with Weight-Parameterized ResNets to enable/facilitate UQ

Residual NNs (ResNets) and Neural ODEs

Neural Networks (NNs) layer-to-layer function

state weights

$$h_{t+1} = F(h_t, w)$$

Residual NNs (ResNets) and Neural ODEs

Neural Networks (NNs) layer-to-layer function

state weights

$$h_{t+1} = F(h_t, w)$$

Residual NN: learn the residual, not the state

Residual NNs (ResNets) and Neural ODEs

state weights

Neural Networks (NNs) layer-to-layer function

$$h_{t+1} = F(h_t, w)$$

Residual NN: learn the residual, not the state

$$h_{t+1} = h_t + \Delta t \, F(h_t, w)$$

Residual NNs (ResNets) and Neural ODEs

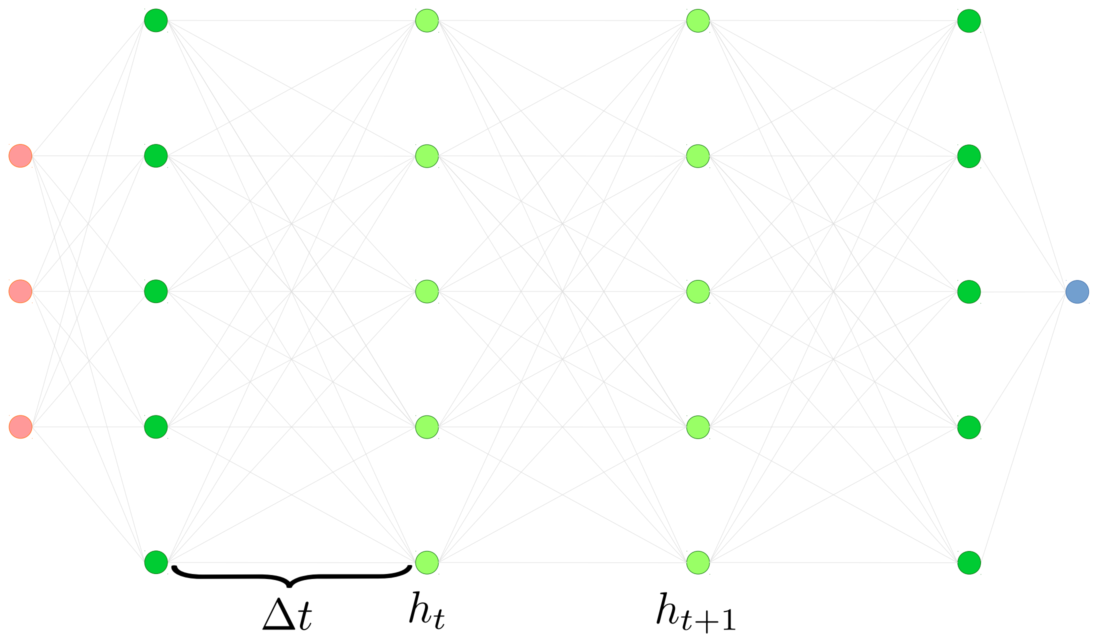
Neural Networks (NNs) layer-to-layer function

$$\underset{\text{state}}{h_{t+1}} = F(h_t, \underset{\text{weights}}{w})$$

Residual NN: learn the residual, not the state $h_{t+1} = h_t + \Delta t\, F(h_t, w)$

Now, take the limit of infinite layers

$$\frac{dh(t)}{dt} = F(h(t), \theta)$$



Residual NNs (ResNets) and Neural ODEs

Neural Networks (NNs) layer-to-layer function

state weights

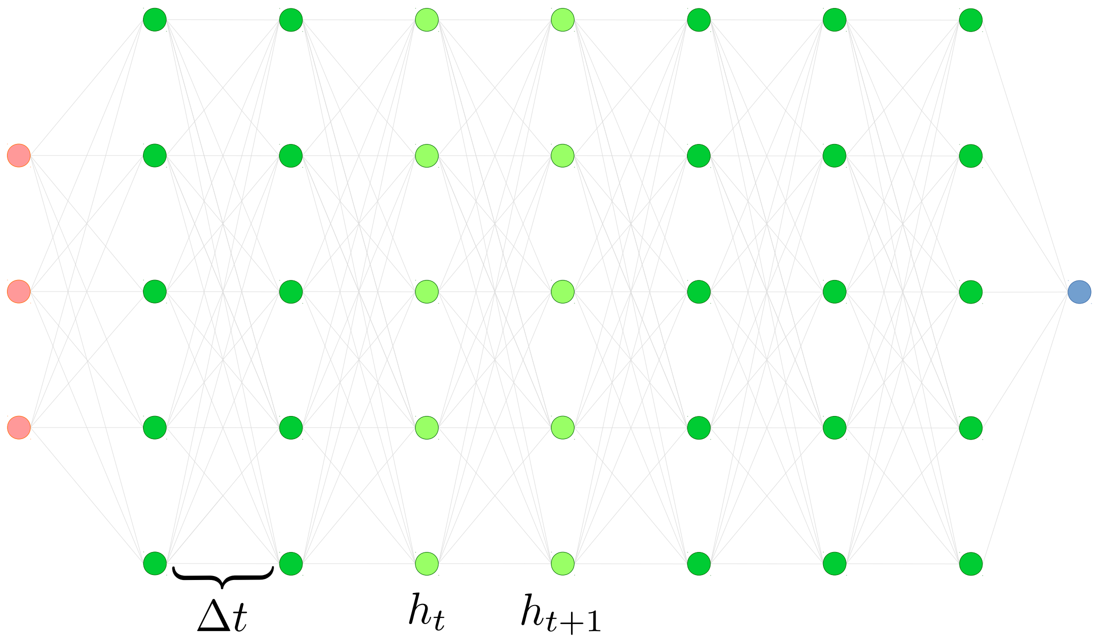
$$h_{t+1} = F(h_t, w)$$

Residual NN: learn the residual, not the state

$$h_{t+1} = h_t + \Delta t \, F(h_t, w)$$

Now, take the limit of infinite layers

$$\frac{dh(t)}{dt} = F(h(t), \theta)$$



Residual NNs (ResNets) and Neural ODEs

Neural Networks (NNs) layer-to-layer function

state weights

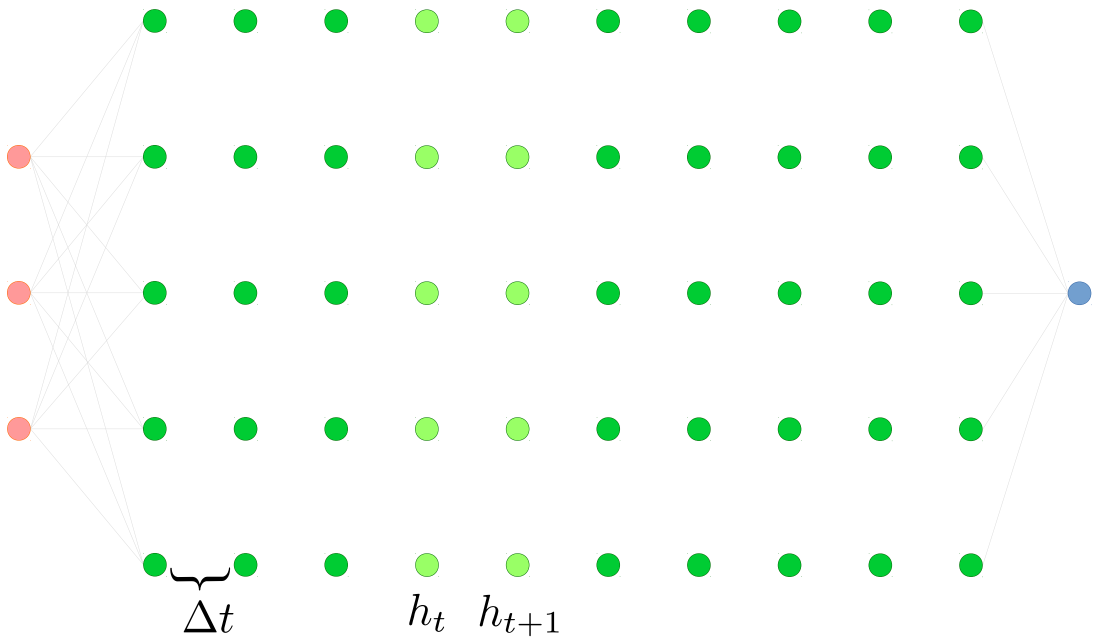
$$h_{t+1} = F(h_t, w)$$

Residual NN: learn the residual, not the state

$$h_{t+1} = h_t + \Delta t\, F(h_t, w)$$

Now, take the limit of infinite layers

$$\frac{dh(t)}{dt} = F(h(t), \theta)$$



Residual NNs (ResNets) and Neural ODEs

Neural Networks (NNs) layer-to-layer function

state weights

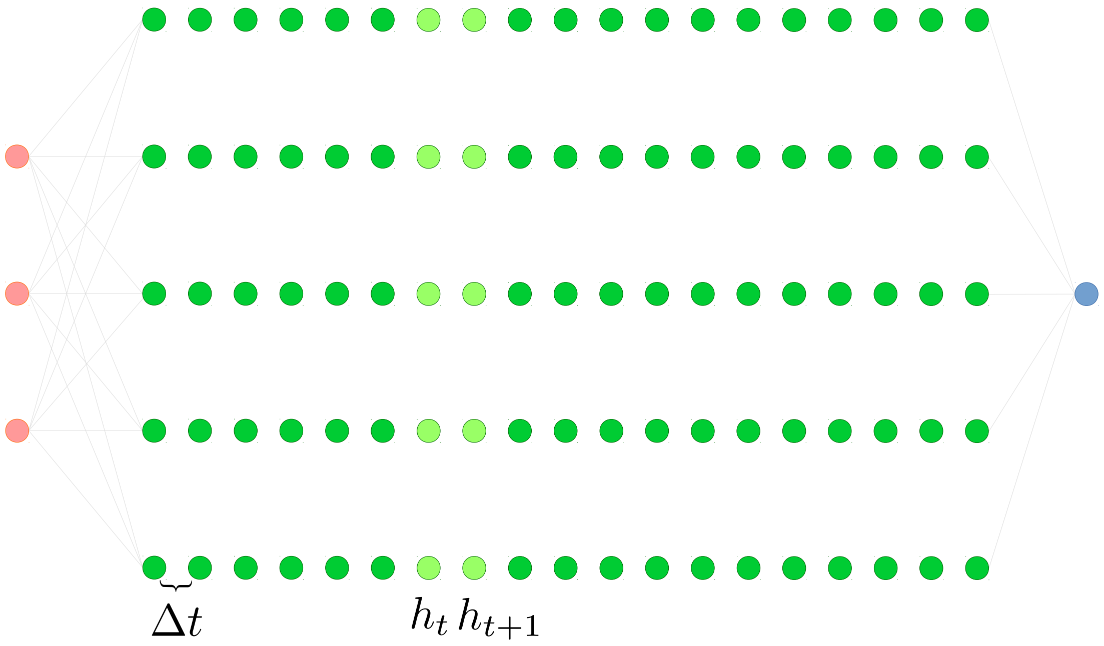
$$h_{t+1} = F(h_t, w)$$

Residual NN: learn the residual, not the state

$$h_{t+1} = h_t + \Delta t \, F(h_t, w)$$

Now, take the limit of infinite layers

$$\frac{dh(t)}{dt} = F(h(t), \theta)$$



Residual NNs (ResNets) and Neural ODEs

Neural Networks (NNs) layer-to-layer function

state weights

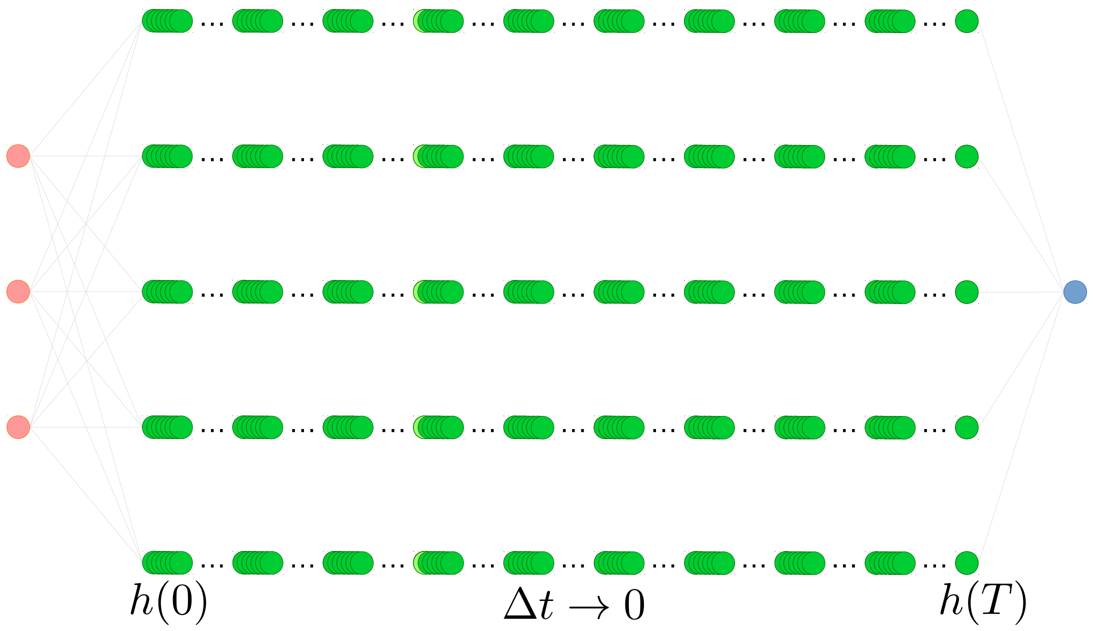
$$h_{t+1} = F(h_t, w)$$

Residual NN: learn the residual, not the state

$$h_{t+1} = h_t + \Delta t\, F(h_t, w)$$

Now, take the limit of infinite layers

$$\frac{dh(t)}{dt} = F(h(t), \theta)$$



Neural ODEs: state of the art

- **Neural ODEs** have 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 formulated as a control problem
- 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]*
- Plenty of challenges: active area of research, mix of optimism and skepticism in literature

Focus today: discrete counterpart of NODEs, **ResNets**, small change from MLPs, but huge gains.

ResNet and Neural ODE in a regression setting (supervised ML)

ResNet (discrete)

$$
\begin{cases}
\boldsymbol{x}_1 = \color{blue}{\boldsymbol{x}} + \boldsymbol{\alpha}_0 \boldsymbol{\sigma}(\boldsymbol{W}_0 \boldsymbol{x}_0 + \boldsymbol{b}_0) \\
\vdots \\
\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \boldsymbol{\alpha}_n \boldsymbol{\sigma}(\boldsymbol{W}_n \boldsymbol{x}_n + \boldsymbol{b}_n) \\
\vdots \\
\color{red}{\boldsymbol{y}} = \boldsymbol{x}_{L-1} + \boldsymbol{\alpha}_{L-1} \boldsymbol{\sigma}(\boldsymbol{W}_{L-1} \boldsymbol{x}_{L-1} + \boldsymbol{b}_{L-1})
\end{cases}
$$

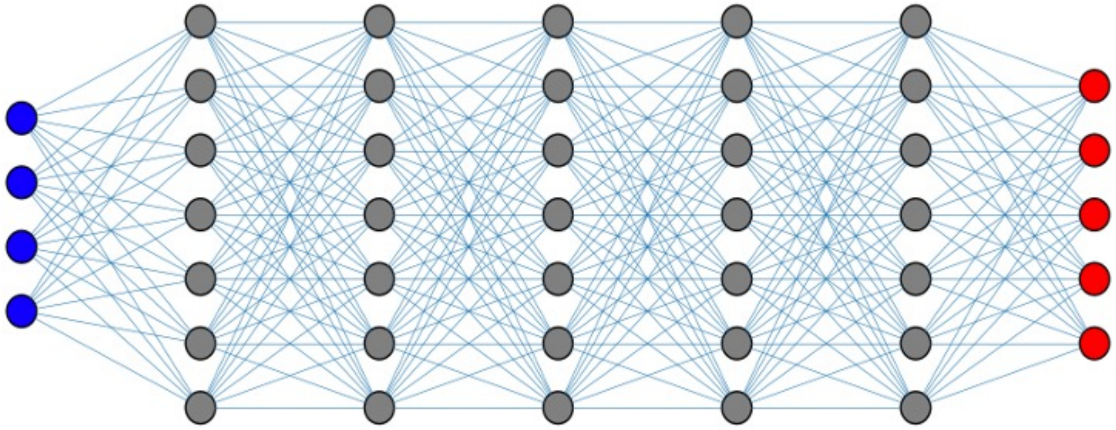
Neural ODE (continuous)

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

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

$\boldsymbol{x} = \boldsymbol{x}_0$

Input



$\boldsymbol{y} = \boldsymbol{x}_T$

Output

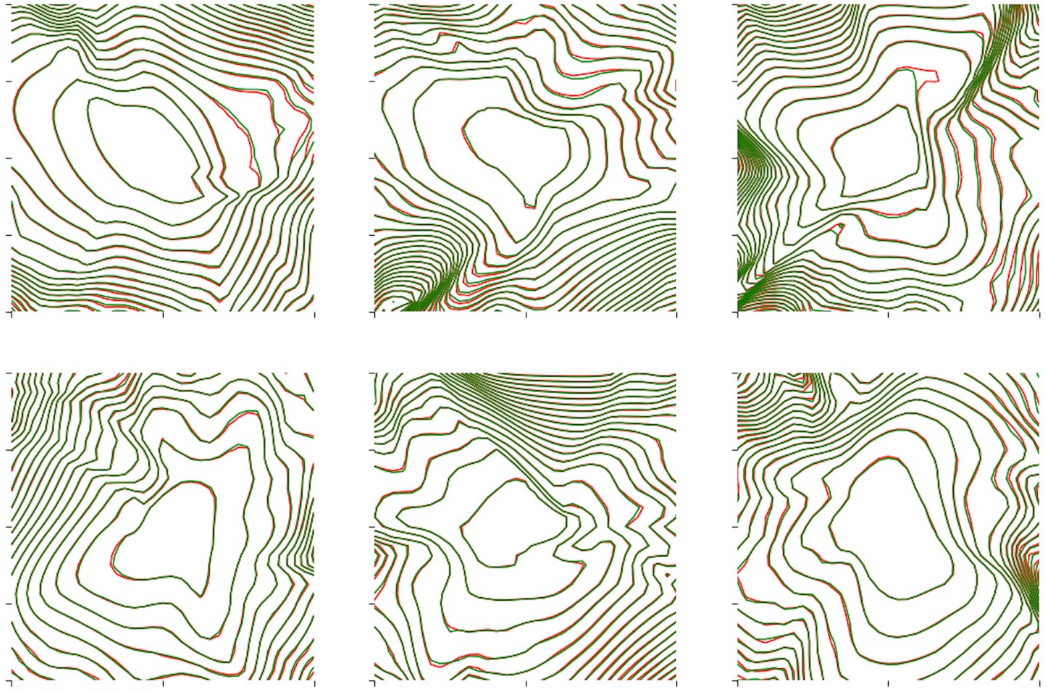
ResNets regularize loss landscape compared to MLPs

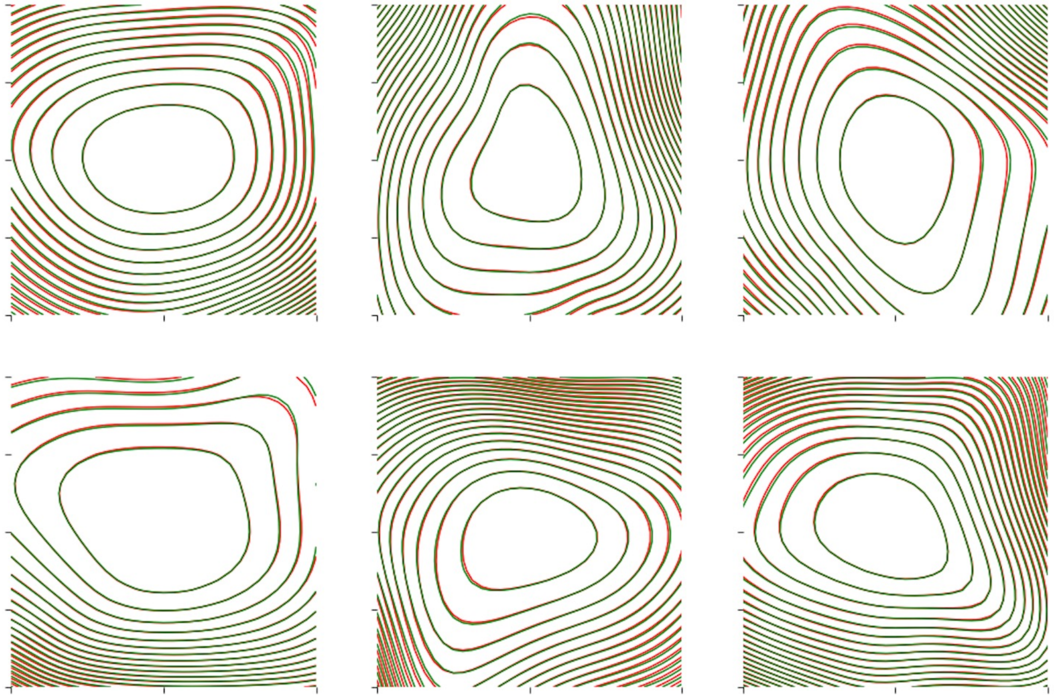
MLP NN: $x_{n+1} = \sigma(W_n x_n + b_n)$

Multilayer Perceptron (learning the layer)

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

ResNets (learning the layer diff.)



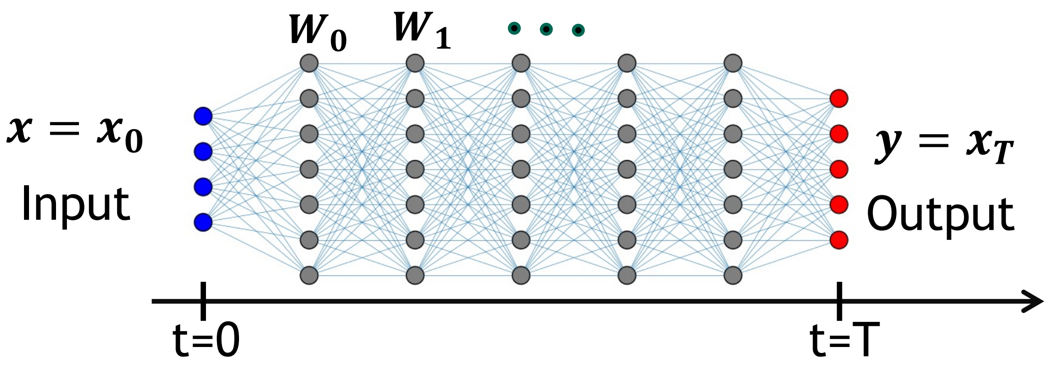


See *[Lee, 2017]* for a more comprehensive study

Weight parameterization inspired by ODEs

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

ResNet: $\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \boldsymbol{\alpha}_n \boldsymbol{\sigma}(\boldsymbol{W}_n \boldsymbol{x}_n + \boldsymbol{b}_n)$

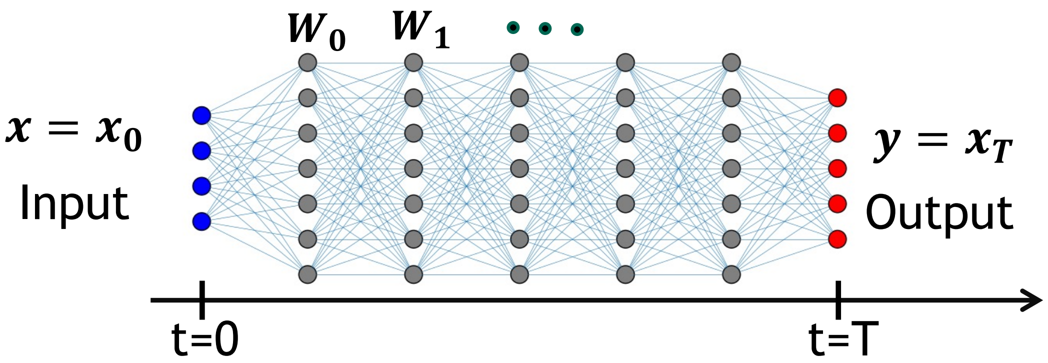


Parameterize weight matrices with respect to time (aka depth)

$\boldsymbol{W}(t;\boldsymbol{\theta})$ and train for $\boldsymbol{\theta}$’s

Weight parameterization as a regularization tool

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

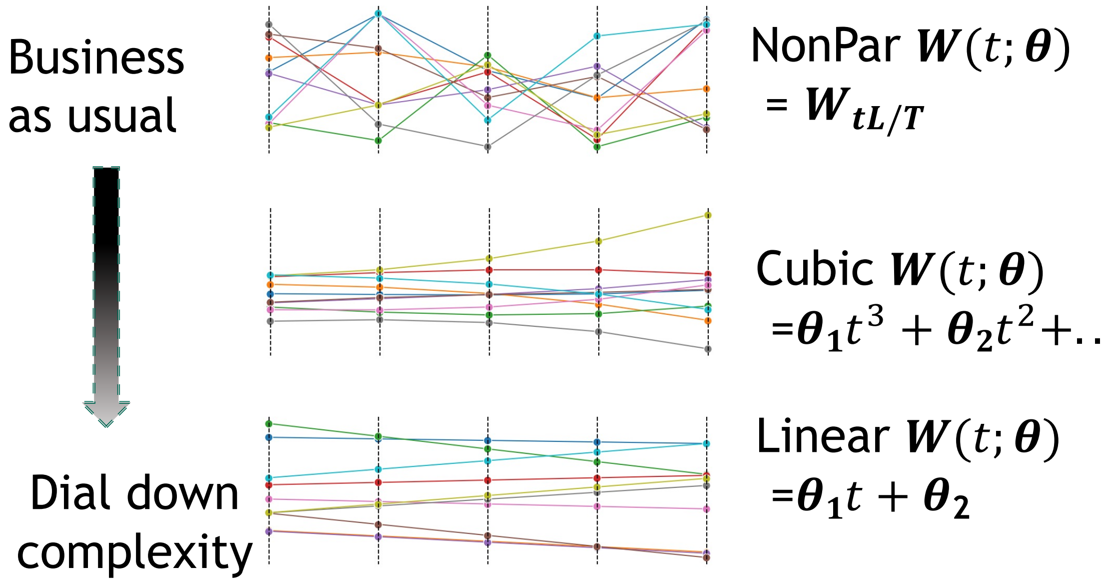


Training for weight matrices $W_0, W_1, \ldots$

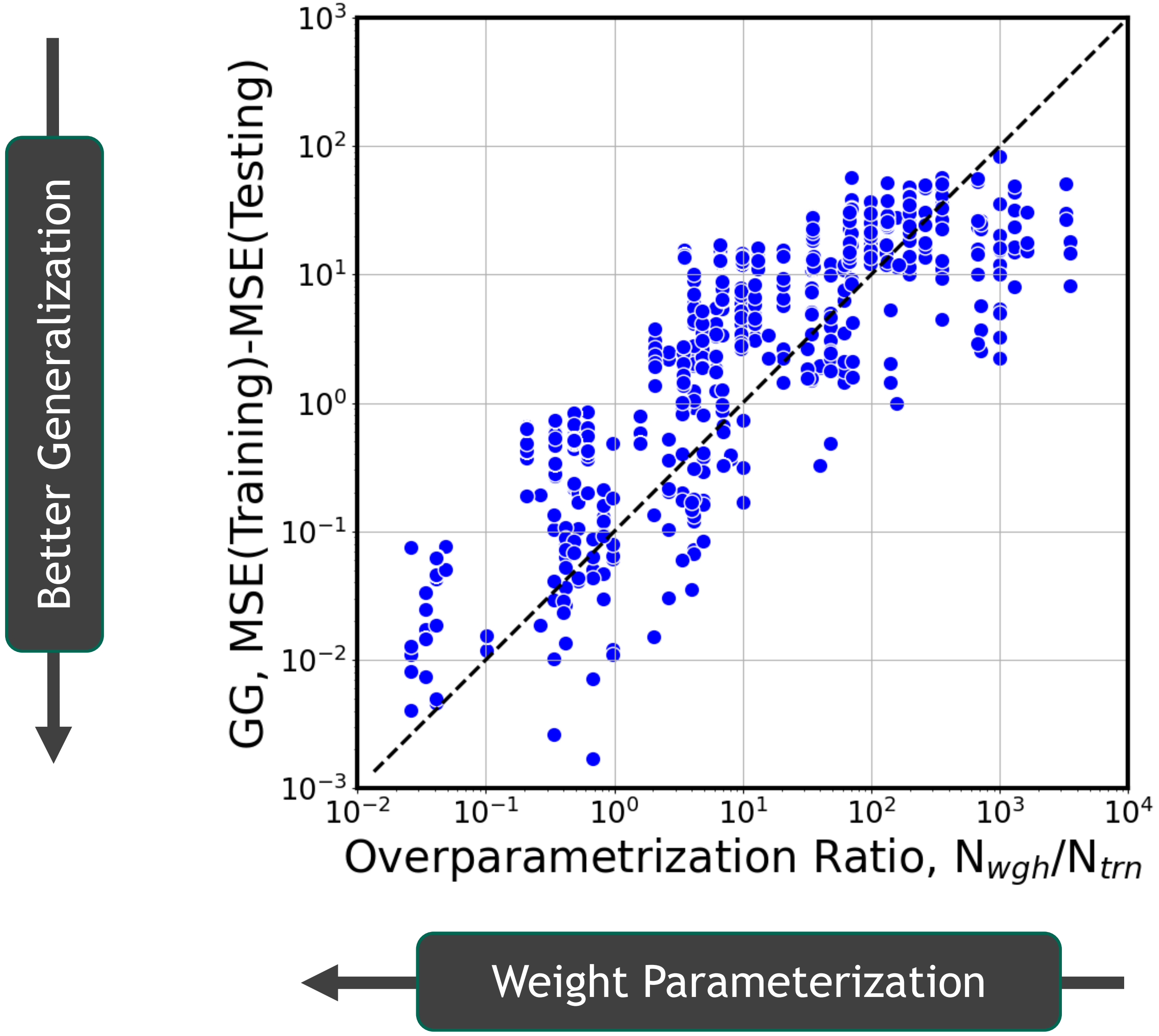
Heavily overparameterized, does not generalize well

Parameterize $W(t;\theta)$ and train for θ's.

Parameterization of weight functions reduces capacity and improves generalization



Weight parameterization (WP) improves generalization

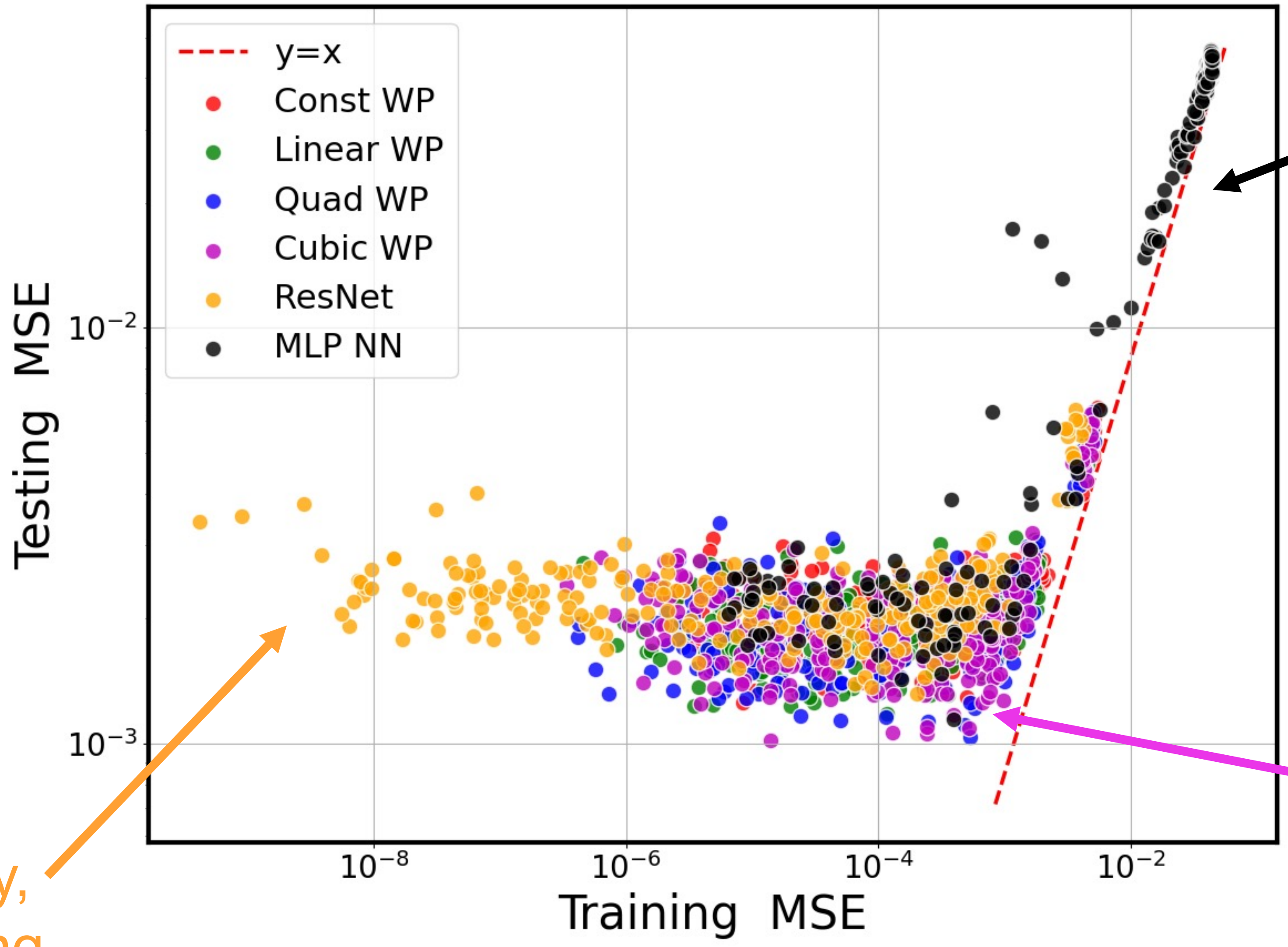


- Generalization Gap correlates with overparameterization

- Weight-parameterized ResNets reduce Generalization Gap

Each dot is a training run with varying weight parameterization functions

ResNet + WP improves accuracy



MLP:
bad accuracy

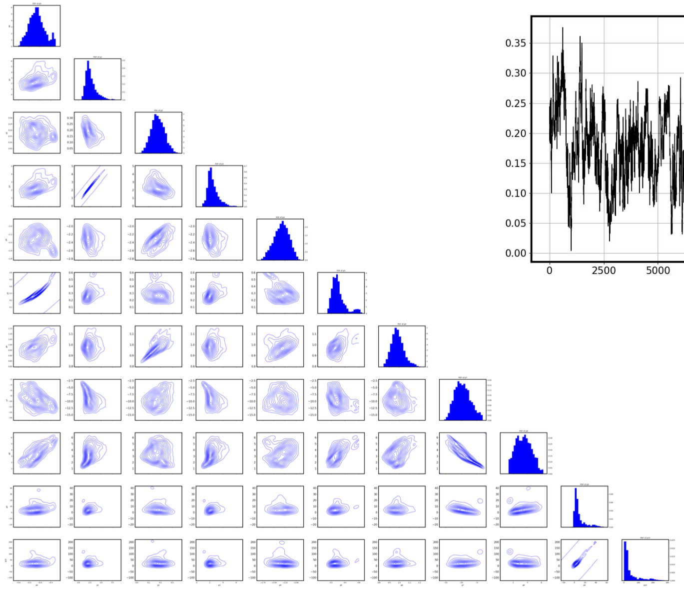
ResNet:
ok accuracy,
but overfitting

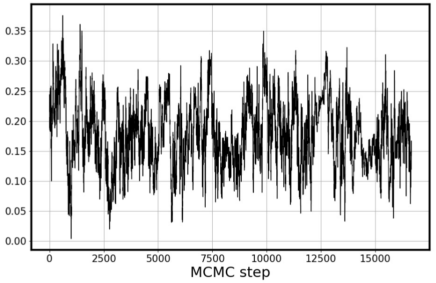
WP:
perfect tradeoff

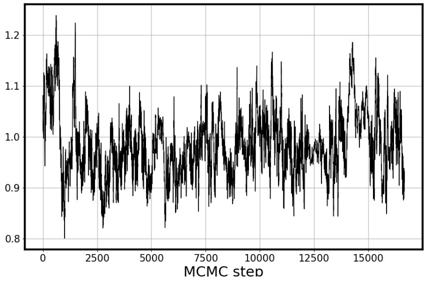
ResNet + WP enables UQ

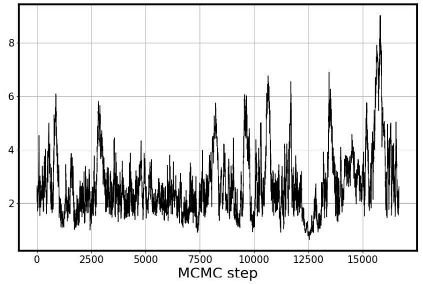
- Number of parameters in ResNets, as well as MLPs, **grows with linearly depth**.
- Number of parameters in weight-parameterized ResNets is **independent of depth**.
- We can easily achieve regimes with manageable MCMC dimensionality and

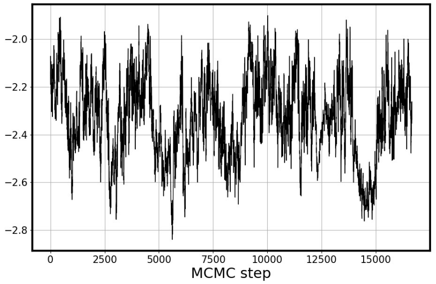
  posterior PDFs that out-of-box MCMC methods can easily sample.

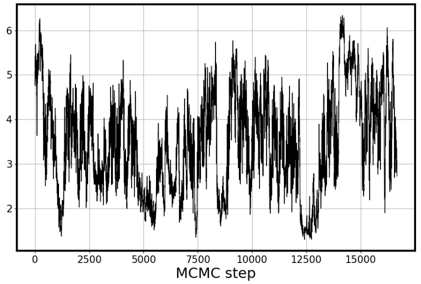






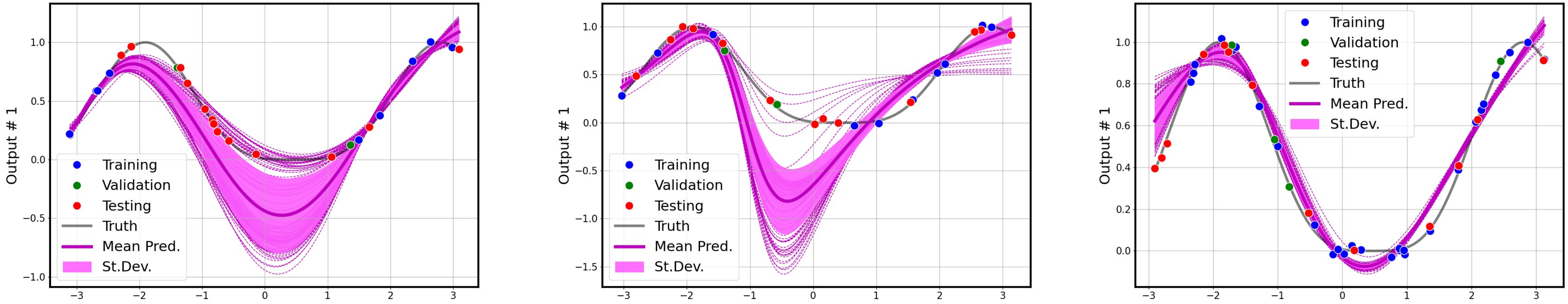






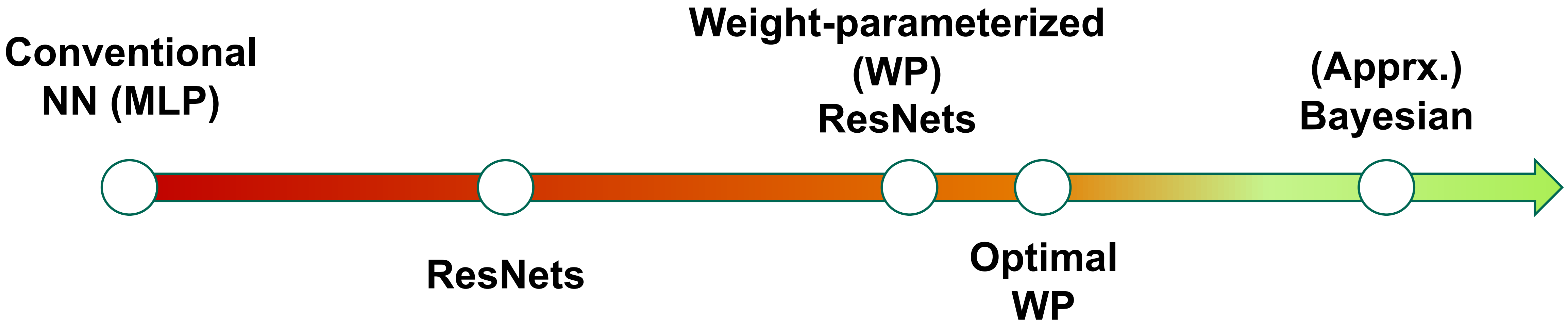
ResNet + WP enables full Bayesian treatment

- Number of parameters in ResNet, as well as MLP, **grows with linearly depth**.
- Number of parameters in weight-parameterized ResNets is **independent of depth**.
- We can easily achieve regimes with manageable MCMC dimensionality and posterior PDFs that out-of-box MCMC methods can easily sample.



Architectural regularization allows UQ

path toward better generalization and confidence assessment



- [Work-in-progress with Lars Ruthotto, Emory U]

  orthogonal expansions for WP

  work better than monomials

QUiNN: Quantifying Uncertainty in NN
github.com/sandialabs/quinn

Deterministic → **Probabilistic**

torch.nn.module → **wrapper(torch.nn.module)**

Usage:

```
uqnet = MCMC_NN(nnet)
```

```
class MCMC_NN(QUiNNBase):
    def __init__(self, nnmodule, verbose=True):
        super(MCMC_NN, self).__init__(nnmodule)
        self.verbose = verbose
```

Option 1: MCMC

```
uqnet = VI_NN(nnet)
```

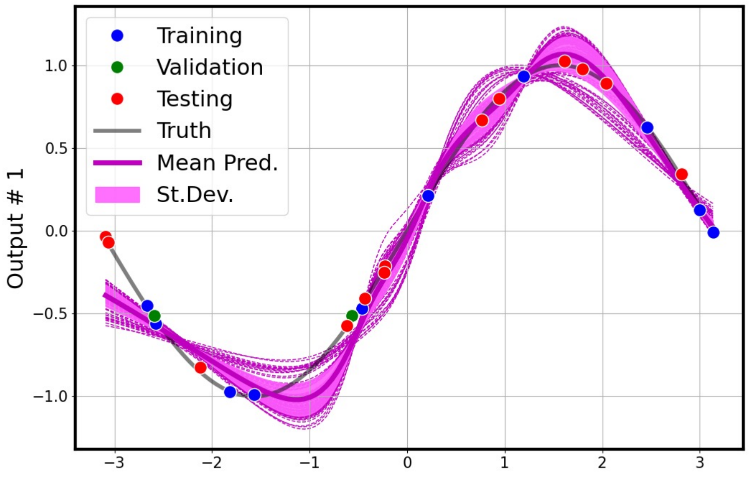
```
class VI_NN(QUiNNBase):
    def __init__(self, nnmodule, verbose=False):
        super(VI_NN, self).__init__(nnmodule)
        self.bmodel = BNet(nnmodule)
        self.verbose = verbose
```

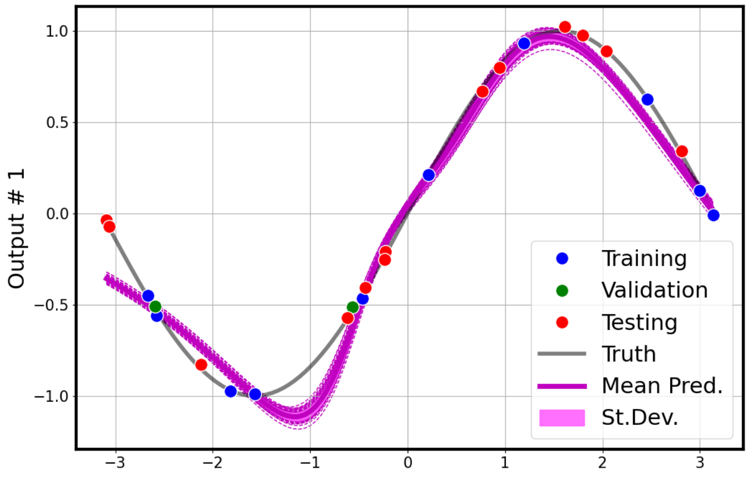
Option 2: Variational Inference

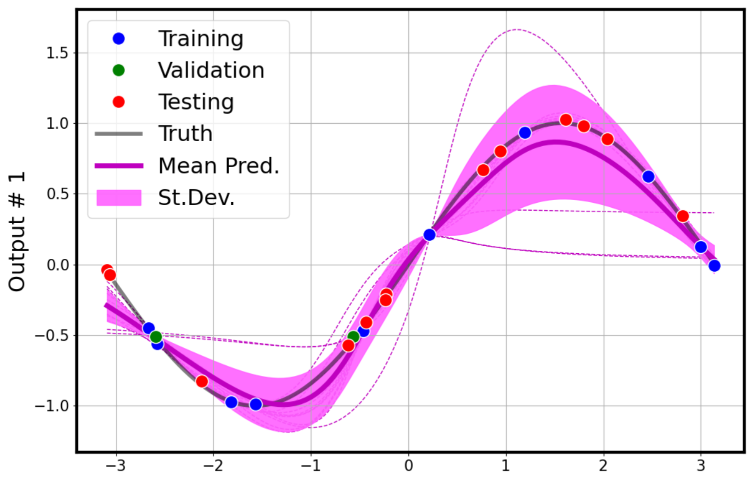
```
uqnet = Ens_NN(nnet, nens=nmc)
```

```
class Ens_NN(QUiNNBase):
    def __init__(self, nnmodule, nens=1, verbose=False):
        super(Ens_NN, self).__init__(nnmodule)
        self.verbose = verbose
        self.nens = nens
```

Option 3: Ensembling







Summary

- **UQ for NN** challenged by many factors
- Draw inspiration from ODE and infinite depth limit
- ResNets regularize the learning problem, smoother loss/log-posterior surface
- **Weight parameterization** allows regularization without losing much expressivity
- Full Bayesian UQ treatment made more feasible with weight-parameterized residual NNs (WP ResNets)
- *In progress:* optimal (e.g., ortho basis) WP for better training and more accuracy
- *In progress:* extention to infinite-depth limit, Neural ODEs
- Implemented in QUiNN: github.com/sandialabs/quinn modular code as a wrapper to three base categories of methods (MCMC, VI, Ens)

Literature

Thank you!

- Z. Ghahramani, "Probabilistic machine learning and artificial intelligence". Nature 521, 452–459 (2015)
- D. J. C. MacKay, "A practical Bayesian framework for backpropagation networks". Neural Computation 4 448–472 (1992)
- R. M. Neal, "Bayesian Learning for Neural Networks". Springer, New York (1996)
- D. Lévy, M. D. Hoffman, and J. Sohl-Dickstein, "Generalizing Hamiltonian Monte Carlo with Neural Networks". ICLR (2018)
- C. Blundell, J. Cornebise, K. Kavukcuoglu, D. Wierstra, "Weight uncertainty in neural networks". arXiv:1505.05424 (2015)
- J.M. Hernández-Lobato, R. Adams, "Probabilistic backpropagation for scalable learning of Bayesian neural networks". ICML (2015)
- E. Daxberger, A. Kristiadi, A. Immer, R. Eschenhagen, M. Bauer, P. Hennig, "Laplace Redux-Effortless Bayesian Deep Learning" Advances in neural inf. proc. systems 34 (2021)
- Y. Gal, Z. Ghahramani, "Dropout as a Bayesian approximation: representing model uncertainty in deep learning". ICML (2016)
- B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles". NIPS'17. 6405–6416 (2017)
- T. Pearce, F. Leibfried, A. Brintrup, "Uncertainty in Neural Networks: Approximately Bayesian Ensembling". Artificial Intelligence and Statistics, 108:234-244 (2020)
- W.J. Maddox, P Izmailov, T. Garipov, D.P. Vetrov, A. G. Wilson, "A simple baseline for Bayesian uncertainty in deep learning". NIPS (2019)
- R. Anirudh, J. J. Thiagarajan. "Delta-UQ: Accurate Uncertainty Quantification via Anchor Marginalization", arxiv.org/abs/2110.02197 (2021)
- Y. Hu, J. Musielewicz, Z. W. Ulissi and A. J. Medford, "Robust and scalable uncertainty estimation with conformal prediction for machine-learned interatomic potentials" Machine Learning: Science and Technology, 3-4 (2022)
- L. Guo, H. Wu, W. Zhou, Y. Wang, T. Zhou, "IB-UQ: Information bottleneck based uncertainty quantification for neural function regression and neural operator learning", https://arxiv.org/abs/2302.03271 (2023)
- H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein, "Visualizing the Loss Landscape of Neural Nets, NIPS (2018)
- R. T. Q. Chen, Y. Rubanova, J. Bettencourt, D. Duvenaud, "Neural ordinary differential equations". NIPS'18 (2018).
- L. Ruthotto, E. Haber, "Deep neural networks motivated by partial differential equations". arXiv preprint arXiv:1804.04272 (2018)
- W. E, "A Proposal on Machine Learning via Dynamical Systems". Commun. Math. Stat. 5, 1–11 (2017)