Deep Neural Networks as Gaussian Processes

Jaehoon Lee

Google Brain

Workshop on Accelerating the Search for Dark Matter with Machine Learning

April 10, 2019

Based on

Published as a conference paper at ICLR 2018

DEEP NEURAL NETWORKS AS GAUSSIAN PROCESSES

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

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

- Published in ICLR 2018, https://arxiv.org/abs/1711.00165
- Open source code : https://github.com/brain-research/nngp

Outline

- Motivation
- Review of Bayesian Neural Networks
- Review of Gaussian Process
- Deep Neural Networks as Gaussian Processes
- Experiment
- Conclusion

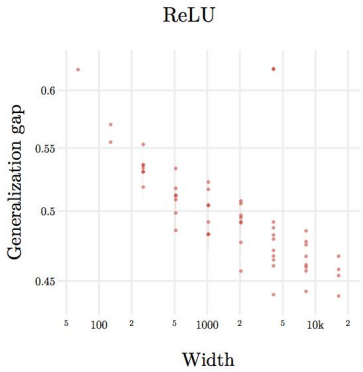
Motivation

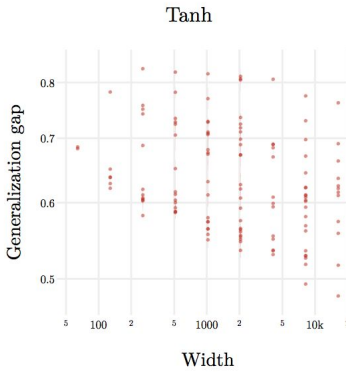
- Recent success with deep neural networks (DNN)
  - Speech recognition
  - Computer vision
  - Natural language processing
  - Machine translation
  - Game playing (Atari, Go, Dota2, ...)
- However, theoretical understanding is still far behind
  - Physicist way of approaching DNN: treat it as a complex `physical' system
  - Find simplifying limits that we could understand. Expand around (perturbation theory!)
  - We will consider overparameterized or infinitely wide limit
    - Other options (large depth, large data, small learning rate, ...)

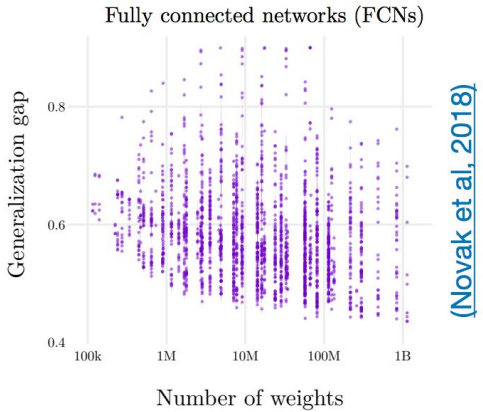
Why study overparameterized neural networks?

- Often wide networks generalize better!

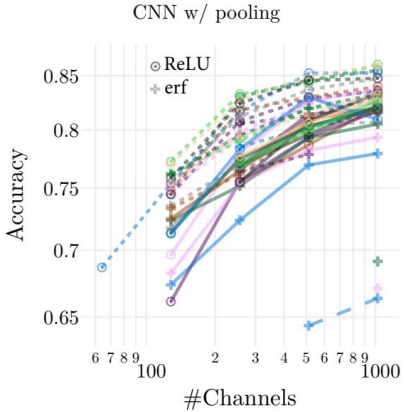
(CIFAR10, all 100% accurate NNs on training set)







(Novak et al, 2018)



(Novak & Xiao et al, 2019)

Why study overparameterized neural networks?

- Often larger networks generalize better!

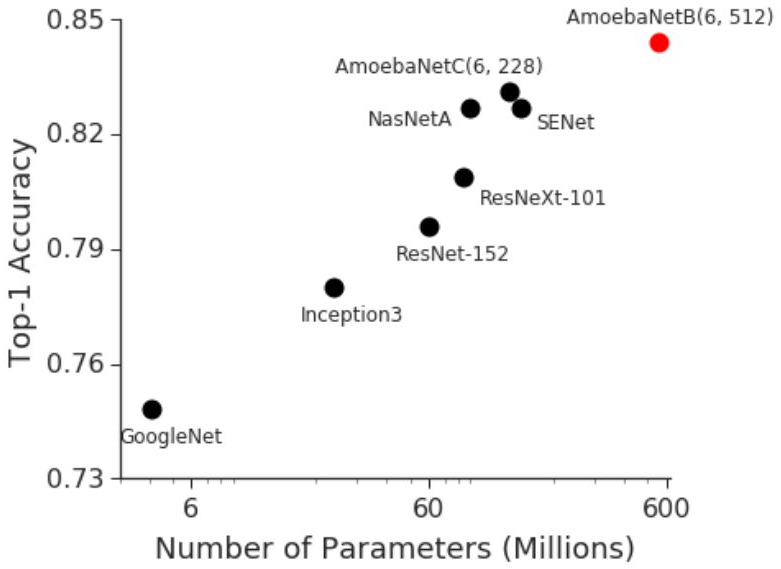
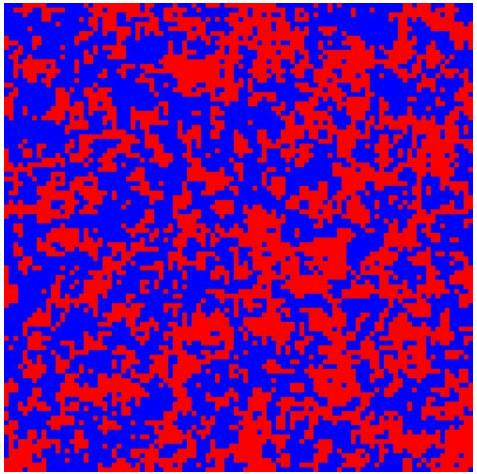


Figure 1: Strong correlation between top-1 accuracy on ImageNet 2012 validation dataset and model size for representative state-of-the-art image classification models in recent years [49, 50, 23, 54, 24, 57, 45]. Red dot shows 84.3% top-1 accuracy for a giant AmoebaNet model trained by GPipe.

Y. Huang et al., GPipe, 2018
arXiv: 1811.06965

Why study overparameterized neural networks?

- Allows theoretically simplifying limits (thermodynamic limit)
- Large neural networks with many parameters as statistical mechanical systems
- Apply obtained insights to finite models



Ising model simulation,
Credit: J. Sethna (Cornell)

Bayesian deep learning

- Usual gradient based training of NN : maximum likelihood (or maximum posterior) estimate
  - Point estimate
  - Does not provide posterior distribution

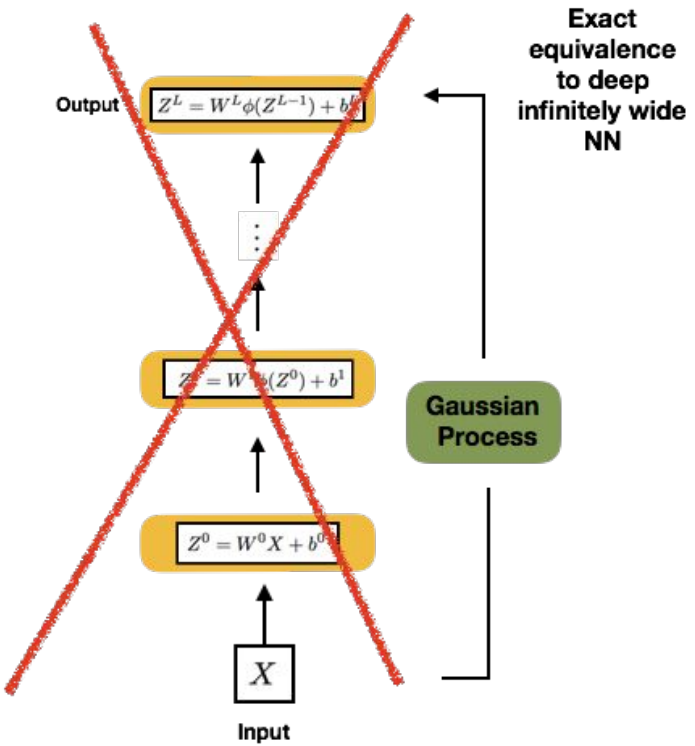
- Bayesian deep learning : marginalize over parameter distribution
  - Uncertainty estimates
  - Principled model selection
  - Robust against overfitting

$$p(\theta|D) = \frac{P(D|\theta)p(\theta)}{\int d\theta P(D|\theta)p(\theta)}$$

- Why don't we use it then?
  - High computational cost (estimating posterior weight dist)
  - Rely on approximate methods (variational / MCMC): does not provide enough benefit

Bayesian deep learning via GPs

- Benefits
  - Uncertainty estimates
  - Principled model selection
  - Robust against overfitting
- Problem
  - High computational cost (estimating posterior weight dist.)
  - Rely on approximate methods (variational / MCMC)
- **Our suggestion**
  - Exact GP equivalence to infinitely wide, deep networks
  - Works for any depth
  - Bayesian inference of DNN, without training!



Deep Neural Networks as GPs

Motivations:

- To understand neural networks, can we connect them to objects we better understand?
- Function space vs parameter space point of view
- An algorithmic aspect: perform Bayesian inference with neural networks?

Main Results:

- Correspondence between Gaussian processes and priors for *infinitely wide,* deep neural networks.
- We implement the GP (will refer to as NNGP) and use it to do Bayesian inference. We compare its performance to wide neural networks trained with stochastic optimization on MNIST & CIFAR-10.

Reminder: Gaussian Processes

GP provides a way to specify prior distribution over certain class of functions

Recall the **definition** of a Gaussian process:

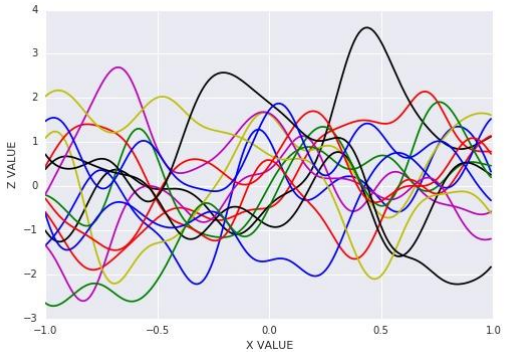
$z(x) \sim \mathcal{GP}(\mu, K)$, with mean and covariance functions $\mu(x), K(x, x')$, if any finite set of draws, $[z(x_1), ..., z(x_n)]^T$, follows $\mathcal{N}(\vec{\mu}, \mathbf{K})$ with

$$\vec{\mu} = \begin{bmatrix} \mu(x_1) \\ \vdots \\ \mu(x_n) \end{bmatrix}, \mathbf{K} = \begin{bmatrix} K(x_1, x_1) & \cdots & K(x_1, x_n) \\ \vdots & \ddots & \vdots \\ K(x_n, x_1) & \cdots & K(x_n, x_n) \end{bmatrix}$$

For instance, for the RBF(radial basis function) kernel,

$$K(x, x') = e^{-\frac{||x - x'||^2}{2\sigma^2}}$$

Samples from GP with RBF Kernel



Gaussian process Bayesian inference

Bayesian inference involves high-dimensional integration in general

For GP regression, can perform inference exactly because all the integrals are Gaussian

Conditional / Marginal distribution of a Gaussian is also a Gaussian

Result (Williams 97) is:

$$\text{Output } z^*|\mathcal{D}, x^* \sim \mathcal{N}(\bar{\mu}, \bar{K})$$

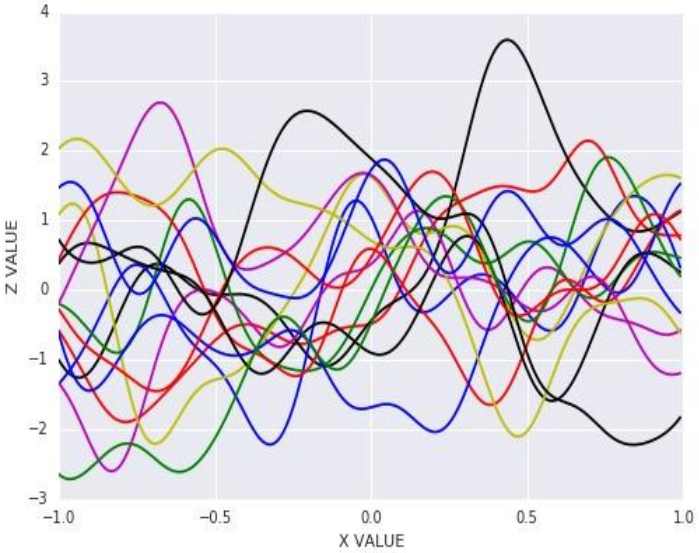
$$\bar{\mu} = K_{x^*,\mathcal{D}}(K_{\mathcal{D},\mathcal{D}} + \sigma_\epsilon^2 \mathbb{I}_n)^{-1}\boldsymbol{t}$$

$$\bar{K} = K_{x^*,x^*} - K_{x^*,\mathcal{D}}(K_{\mathcal{D},\mathcal{D}} + \sigma_\epsilon^2 \mathbb{I}_n)^{-1}K_{x^*,\mathcal{D}}^T$$

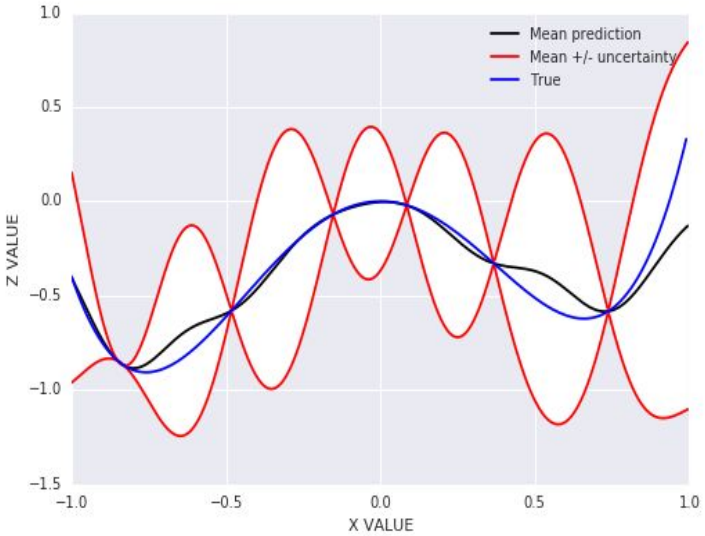
Reduces Bayesian inference to doing linear algebra. (Typically cubic cost in training samples)

GP Bayesian inference

Prior with RBF Kernel



Posterior with RBF Kernel



Gaussian process

Non-parametric: models distribution over non-linear functions
    Covariance function (and mean function)

Probabilistic, Bayesian: uncertainty estimates, model comparison, robust against overfitting

Simple inference using linear algebra only (no sampling required)
    Exact posterior predictive distribution

Cubic time cost and quadratic memory cost in training samples

Few example of recent HEP papers utilizing GPs

Bertone et al., *Accelerating the BSM interpretation of LHC data with machine learning*, 1611.02704
Frate et al., Modeling Smooth Backgrounds & Generic Localized Signals with Gaussian Processes, 1709.05681
Bertone et al., Identifying WIMP dark matter from particle and astroparticle data, 1712.04793

Further read: A Visual Exploration of Gaussian Processes, Gortler et al., Distill, 2019

The single hidden layer case

Radford Neal, “Priors for Infinite Networks,” 1994.

Neal observed that given a neural network (NN) which:

- has **a single hidden layer**
- is **fully-connected**
- has **i.i.d. prior over parameters (such that it give a sensible limit)**

Then the distribution on its output converges to a Gaussian Process (GP) **in the limit of infinite layer width.**

The single hidden layer case

Inputs: $x_a \in \mathbb{R}^{N_0}$ $\Sigma^0_{ab} = \frac{1}{N_0}\sum_i x_{ia}x_{ib}$ Uncentered covariance

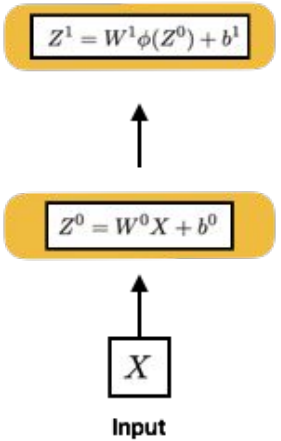
Parameters: $W^l \in \mathbb{R}^{N_l \times N_{l+1}}$ $b^l \in \mathbb{R}^{N_{l+1}}$

Priors over parameters: $W^l_{i,j} \sim \mathcal{N}(0, \frac{\sigma_w^2}{N_l})$ $b^l_i \sim \mathcal{N}(0, \sigma_b^2)$

Network:

$$z^0_{ia} = \sum_j^{N_0} W^0_{ij} x_{ja} + b^0_i$$

$$z^1_{ia} = \sum_j^{N_1} W^1_{ij} \phi(z^0_{ja}) + b^1_i$$



The single hidden layer case

Inputs: $x_a \in \mathbb{R}^{N_0}$ $\Sigma^0_{ab} = \frac{1}{N_0}\sum_i x_{ia}x_{ib}$ Uncentered covariance

Parameters: $W^l \in \mathbb{R}^{N_l \times N_{l+1}}$ $b^l \in \mathbb{R}^{N_{l+1}}$

Priors over parameters: $W^l_{i,j} \sim \mathcal{N}(0, \frac{\sigma_w^2}{N_l})$ $b^l_i \sim \mathcal{N}(0, \sigma_b^2)$

Network:

$$z^0_{ia} = \sum_j^{N_0} W^0_{ij} x_{ja} + b^0_i$$

$$\Sigma^1 = \sigma_w^2 \Sigma^0 + \sigma_b^2$$

$$(z^0_{ia}, z^0_{jb})^T \sim \mathcal{N}(0, \Sigma^1_{ab}\delta_{ij})$$

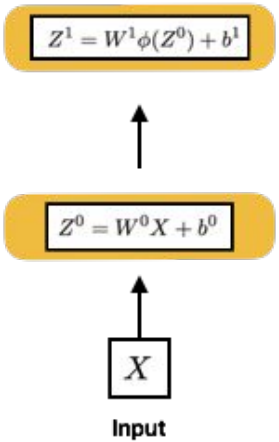
Sum of i.i.d. random variables

$$z^1_{ia} = \sum_j^{N_1} W^1_{ij} \phi(z^0_{ja}) + b^1_i$$

$\Sigma^2 = \sigma_w^2 \mathbb{E}_{z\sim\mathcal{N}(0,\Sigma^1)}[\phi(z)\phi(z)^T] + \sigma_b^2$

$$(z^1_{ia}, z^1_{jb})^T \xrightarrow[N_1 \to \infty]{} \mathcal{N}(0, \Sigma^2_{ab}\delta_{ij})$$

Multivariate C.L.T.



Note that z_i and z_j are independent because they have Normal joint and zero covariance

The single hidden layer case

Infinitely wide neural networks are Gaussian processes:

$$z^0_{ia} = \sum_j^{N_0} W^0_{ij} x_{ja} + b^0_i \qquad (z^0_{ia}, z^0_{jb})^T \sim \mathcal{N}(0, \Sigma^1_{ab}\delta_{ij})$$

$$z^1_{ia} = \sum_j^{N_1} W^1_{ij} \phi(z^0_{ja}) + b^1_i \qquad (z^1_{ia}, z^1_{jb})^T \xrightarrow[N_1 \to \infty]{} \mathcal{N}(0, \Sigma^2_{ab}\delta_{ij})$$

Completely defined by compositional kernel

$$\Sigma^1 = \sigma_w^2 \Sigma^0 + \sigma_b^2$$

$$\Sigma^2 = \sigma_w^2 \mathbb{E}_{z \sim \mathcal{N}(0, \Sigma^1)}[\phi(z)\phi(z)^T] + \sigma_b^2$$

Extension to deep networks

$$z^0_{ia} = \sum_j^{N_0} W^0_{ij} x_{ja} + b^0_i \qquad \longrightarrow \qquad \Sigma^1 = \sigma_w^2 \Sigma^0 + \sigma_b^2$$

$$(z^0_{ia}, z^0_{jb})^T \sim \mathcal{N}(0, \Sigma^1_{ab}\delta_{ij})$$

Extension to deep networks

$$z^0_{ia} = \sum_j^{N_0} W^0_{ij} x_{ja} + b^0_i \longrightarrow \Sigma^1 = \sigma_w^2 \Sigma^0 + \sigma_b^2$$

$$(z^0_{ia}, z^0_{jb})^T \sim \mathcal{N}(0, \Sigma^1_{ab}\delta_{ij})$$

$$\downarrow$$

$$z^1_{ia} = \sum_j^{N_1} W^1_{ij} \phi(z^0_{ja}) + b^1_i \xrightarrow{N_1 \to \infty} \Sigma^2 = \sigma_w^2 \mathbb{E}_{z\sim\mathcal{N}(0,\Sigma^1)}[\phi(z)\phi(z)^T] + \sigma_b^2$$

$$(z^1_{ia}, z^1_{jb})^T \sim \mathcal{N}(0, \Sigma^2_{ab}\delta_{ij})$$

Extension to deep networks

$$z^0_{ia} = \sum_j^{N_0} W^0_{ij} x_{ja} + b^0_i \longrightarrow \Sigma^1 = \sigma_w^2 \Sigma^0 + \sigma_b^2$$

$$(z^0_{ia}, z^0_{jb})^T \sim \mathcal{N}(0, \Sigma^1_{ab}\delta_{ij})$$

$$\downarrow$$

$$z^1_{ia} = \sum_j^{N_1} W^1_{ij} \phi(z^0_{ja}) + b^1_i \xrightarrow{N_1 \to \infty} \Sigma^2 = \sigma_w^2 \mathbb{E}_{z\sim\mathcal{N}(0,\Sigma^1)}[\phi(z)\phi(z)^T] + \sigma_b^2$$

$$(z^1_{ia}, z^1_{jb})^T \sim \mathcal{N}(0, \Sigma^2_{ab}\delta_{ij})$$

$$\vdots$$

$$z^l_{ia} = \sum_j^{N_l} W^l_{ij} \phi(z^{l-1}_{ja}) + b^l_i \xrightarrow{N_l \to \infty} \Sigma^{l+1} = \sigma_w^2 \mathbb{E}_{z\sim\mathcal{N}(0,\Sigma^l)}[\phi(z)\phi(z)^T] + \sigma_b^2$$

$$(z^l_{ia}, z^l_{jb})^T \sim \mathcal{N}(0, \Sigma^{l+1}_{ab}\delta_{ij})$$

Reference for more formal treatments

- A. Matthews et al., ICLR 2018
  - Gaussian Process Behaviour in Wide Deep Neural Networks
  - https://arxiv.org/abs/1804.11271

- R. Novak et al., ICLR 2019
  - Bayesian Deep Convolutional Networks with Many Channels are Gaussian Processes
  - https://arxiv.org/abs/1810.05148
  - Appendix E

Few comments about the NNGP Covariance Kernel

At layer L, kernel is fully deterministic given the kernel at layer L-1

$$K^l(x,x') = \sigma_b^2 + \sigma_w^2\, \mathcal{Z}^{-1} \int du_1 du_2\, \phi(u_1)\phi(u_2) \exp\left(-\tfrac{1}{2} [u_1, u_2] \begin{bmatrix} K^{l-1}(x,x) & K^{l-1}(x,x') \\ K^{l-1}(x,x') & K^{l-1}(x',x') \end{bmatrix}^{-1} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \right)$$

$$K^l(x,x') = \sigma_b^2 + \sigma_w^2\, F_\phi\Big(K^{l-1}(x,x'),\, K^{l-1}(x,x),\, K^{l-1}(x',x') \Big)$$

For ReLU / Erf (+ few more), closed form solution exists

$$K^l(x,x') = \sigma_b^2 + \frac{\sigma_w^2}{2\pi} \sqrt{K^{l-1}(x,x) K^{l-1}(x',x')} \left(\sin\theta^{l-1}_{x,x'} + (\pi - \theta^{l-1}_{x,x'}) \cos\theta^{l-1}_{x,x'} \right)$$

$$\theta^l_{x,x'} = \cos^{-1}\left(\frac{K^l(x,x')}{\sqrt{K^l(x,x) K^l(x',x')}} \right).$$

ReLU: ArcCos Kernel (Cho & Saul 2009)

For general activation function, numerical 2d Gaussian integration can be done efficiently

Also, empirical Monte Carlo estimates works for complicated architectures!

Experimental setup

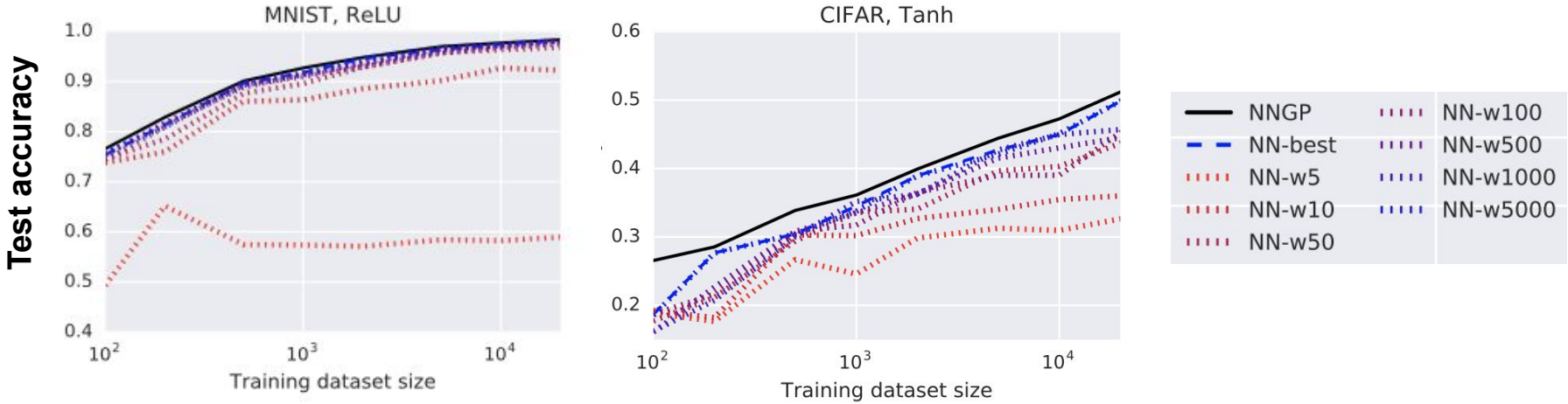
- Datasets: MNIST, CIFAR10
- Permutation invariant, fully-connected model, ReLU/Tanh activation function
- Trained on mean squared loss
- Targets are one-hot encoded, zero-mean and treated as regression target
  - incorrect class -0.1, correct class 0.9
- Hyperparameter optimized
  - Weight/bias variance, optimization hyperparameters (for NN)
- NN: `SGD' trained opposed to Bayesian training.
- NNGP: standard exact Gaussian process regression, 10 independent outputs

Empirical comparison: best models

Num training	Model (ReLU)	Test accuracy	Model (tanh)	Test accuracy
MNIST:1k	NN-2-5000-3.19-0.00	0.9252	NN-2-1000-0.60-0.00	0.9254
	GP-20-1.45-0.28	**0.9279**	GP-20-1.96-0.62	0.9266
MNIST:10k	NN-2-2000-0.42-0.16	0.9771	NN-2-2000-2.41-1.84	0.9745
	GP-7-0.61-0.07	0.9765	GP-2-1.62-0.28	**0.9773**
MNIST:50k	NN-2-2000-0.60-0.44	0.9864	NN-2-5000-0.28-0.34	0.9857
	GP-1-0.10-0.48	0.9875	GP-1-1.28-0.00	**0.9879**
CIFAR:1k	NN-5-500-1.29-0.28	0.3225	NN-1-200-1.45-0.12	0.3378
	GP-7-1.28-0.00	0.3608	GP-50-2.97-0.97	**0.3702**
CIFAR:10k	NN-5-2000-1.60-1.07	0.4545	NN-1-500-1.48-1.59	0.4429
	GP-5-2.97-0.28	**0.4780**	GP-7-3.48-2.00	0.4766
CIFAR:45k	NN-3-5000-0.53-0.01	0.5313	NN-2-2000-1.05-2.08	0.5034
	GP-3-3.31-1.86	**0.5566**	GP-3-3.48-1.52	0.5558

NN-depth-width-σ_w^2-σ_b^2 GP-depth-σ_w^2-σ_b^2

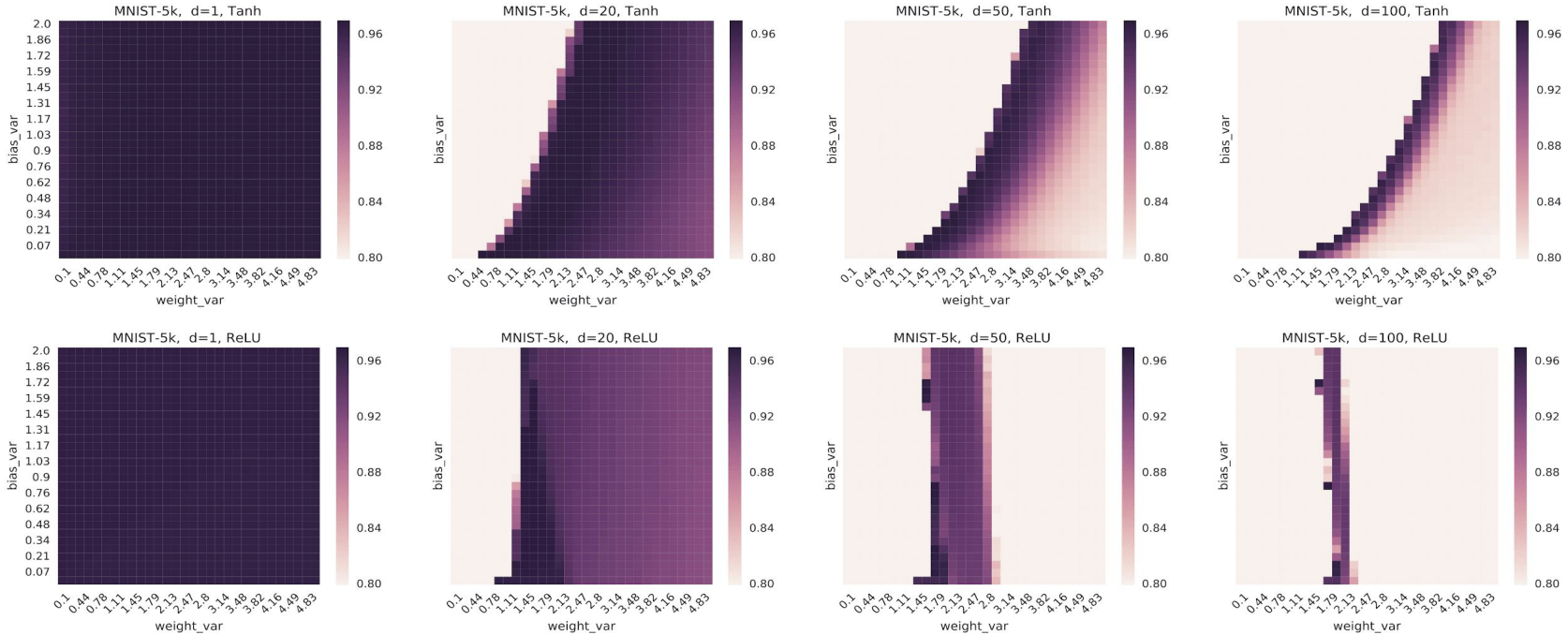
Performance of wide networks approaches NNGP



Performance of finite-width, fully-connected deep NN + SGD → NNGP with exact Bayesian inference

NNGP hyperparameter dependence

Test accuracy



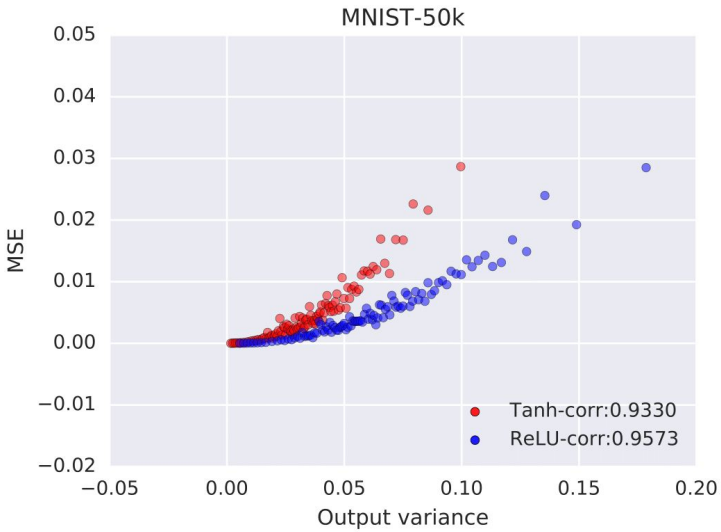
Good agreement with signal propagation study (Schoenholz et al., ICLR 2017)
: interesting structure remains at the “critical” line for very deep networks

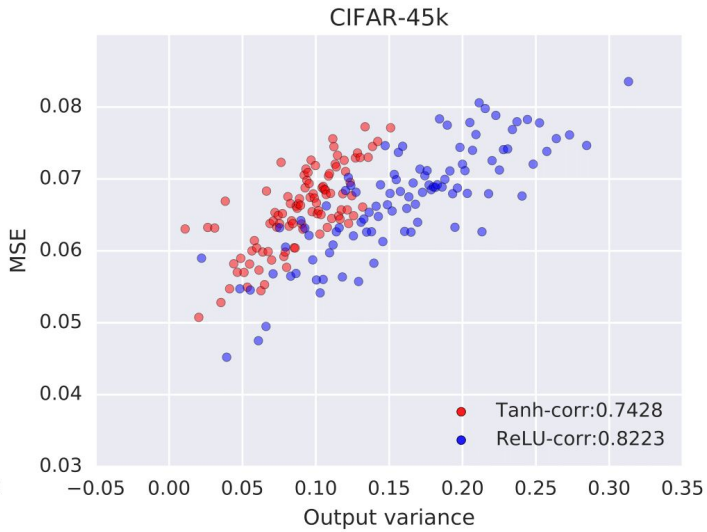
Uncertainty

- Neural networks are good at making predictions, but does not naturally provide uncertainty estimates
- Bayesian methods naturally incorporates uncertainty
- In NNGP, uncertainty of NN's prediction is captured by variance in output

$$\bar{K} = K_{x^*,x^*} - K_{x^*,\mathcal{D}}(K_{\mathcal{D},\mathcal{D}} + \sigma_\epsilon^2 \mathbb{I}_n)^{-1} K_{x^*,\mathcal{D}}^T$$

Uncertainty: empirical comparison





X: predicted uncertainty

Y: realized MSE

* averaged over 100 points binned by predicted uncertainty

Empirical error is well correlated with uncertainty predictions

Next steps

Overparameterization limit opens up interesting angles to further analyze deep neural networks

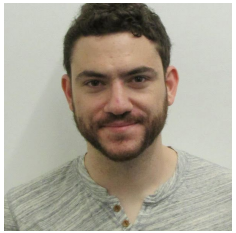
- Practical usage of NNGP
- Extensions to other network architectures
- Systematic finite width corrections

Tractable learning dynamics of overparameterized deep neural networks

- *Wide Deep Neural Networks evolve as Linear Models*, arXiv 1902.06720
- Bayesian inference VS gradient descent training
- Replace a deep neural network by its first-order Taylor expansion around initial parameters

Thanks to the amazing collaborators




Yasaman Bahri, Roman Novak, Jeffrey Pennington, Sam Schoenholz, Jascha Sohl-Dickstein, Lechao Xiao, Greg Yang (MSR)

ICML Workshop: Call for Papers

- 2019 ICML Workshop on Theoretical Physics for Deep Learning

- Location: Long Beach, CA, USA
- Date: June 14 or 15, 2019
- Website: https://sites.google.com/view/icml2019phys4dl
- Submission: 4 pages short paper until 4/30

- Invited speakers: Sanjeev Arora(Princeton), Kyle Cranmer(NYU), David Duvenaud (Toronto, TBC), Michael Mahoney(Berkeley), Andrea Montanari(Stanford), Jascha Sohl-Dickstein(Google Brain), Lenka Zdeborova(CEA/Saclay)
- Organizers: Jaehoon Lee(Google Brain), Jeffrey Pennington(Google Brain), Yasaman Bahri(Google Brain), Max Welling(Amsterdam), Surya Ganguli(Stanford), Joan Bruna(NYU)

Thank you for your attention!