Understanding Wide Neural Networks

Jaehoon Lee
Google Brain

HEP-AI Journal Club
Feb 5, 2019

Joint work with

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

Outline

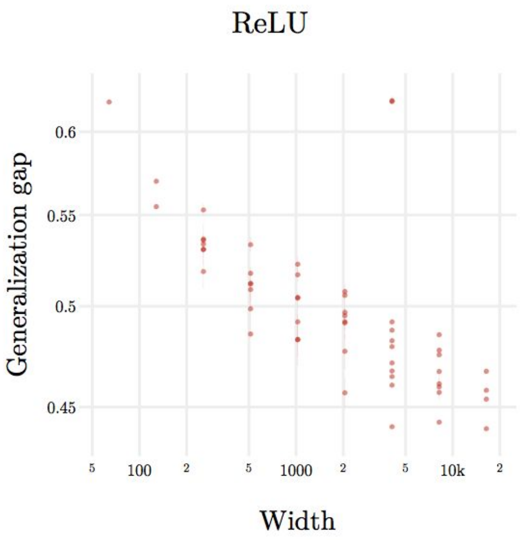
- Motivation
- Deep neural networks as Gaussian processes
  - Formulation / Experiments
- Gradient descent dynamics of wide networks
  - Formulation / Experiments

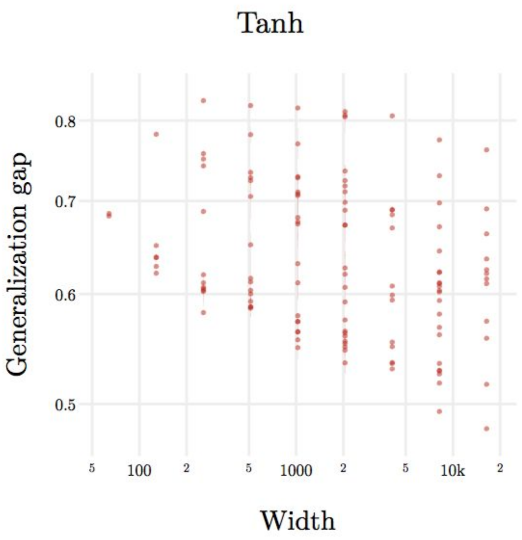
Why study wide neural networks?

- Understand effects of overparameterization
- Theoretically simplifying limits (thermodynamic?)
  - Signal propagation
  - Gaussian process correspondence
  - Gradient descent dynamics
- Think in function space (f) since parameters (w) in a neural network lack direct meaning
  - Random initialization p(w) induces prior over functions p(f)
  - Wide networks makes function space view more tractable
- Often wide networks perform better

Is the large width limit uninteresting?

In practice, find that larger width networks trained with stochastic optimization can generalize better.





Generalization gap for five-hidden layer fully-connected networks with variable widths on CIFAR-10. Filtered for 100% classification training accuracy.

Deep neural networks as Gaussian processes

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

- https://arxiv.org/abs/1711.00165
- Open source code : https://github.com/brain-research/nngp

*Slide credit: Yasaman Bahri

Motivations:

- To understand neural networks, can we connect them to objects we better understand?
- An algorithmic aspect: perform Bayesian inference with neural networks?

Our contributions:

- 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.

Bayesian treatment of neural networks

- Usual gradient based training of NN : maximum likelihood (or maximum posterior) estimate

- Bayesian deep learning : marginalize over parameter distribution
  - Uncertainty estimates
  - Principled model selection
  - Avoid overfitting (model averaging)

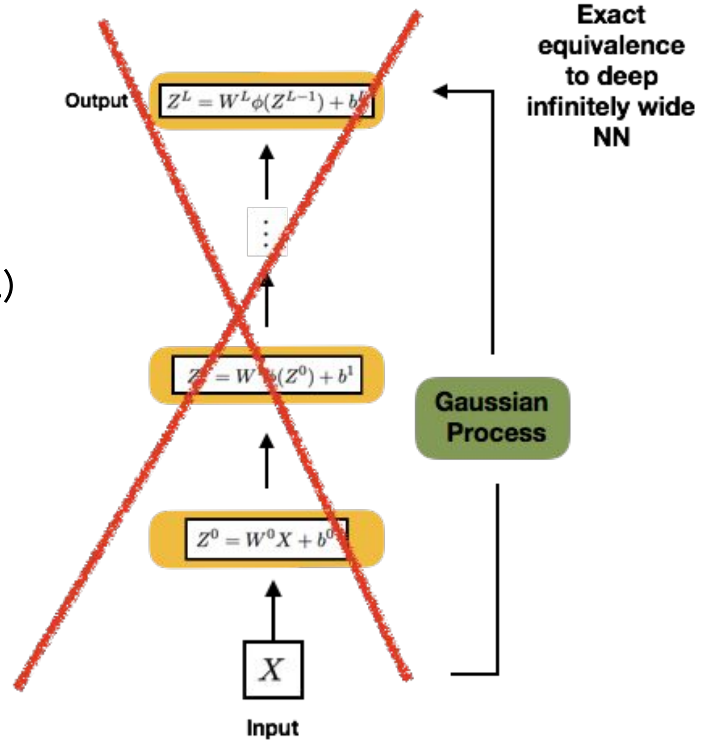
$$p(w|x,y) = \frac{p(x,y|w)p(w)}{\int p(y|x,w)p(w)dw}$$

- Why don't we use it then?

  - High computational cost (estimating posterior weight dist)
  - Rely on approximate methods (variational / MCMC)

Bayesian treatment of deep neural networks by GPs

- Benefits
  - Uncertainty estimates
  - Principled model selection
  - Avoid overfitting (model averaging)
- 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 NN, without training!



Reminder: Gaussian Processes

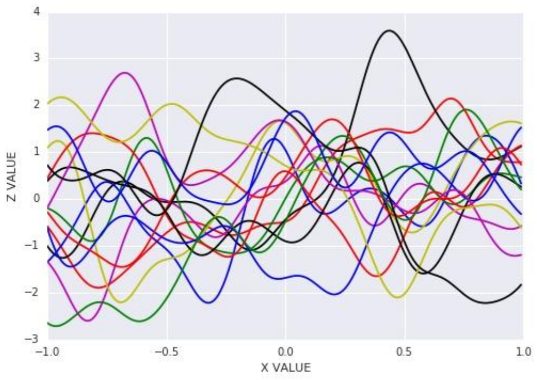
Recall the definition of a Gaussian process:

$z(x) \sim \mathcal{GP}(\mu, K)$, 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

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

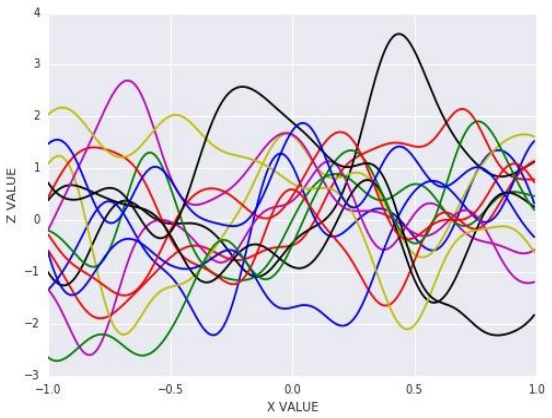
For instance, for the RBF kernel, $K(x, x') = e^{-\frac{||x-x'||^2}{2\sigma^2}}$

Samples from GP with RBF Kernel

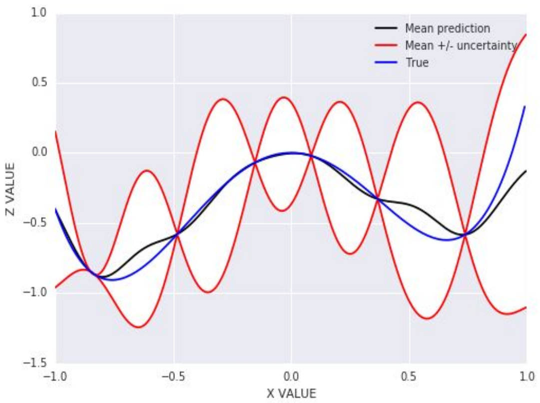


Bayesian inference using a GP prior

Prior with RBF Kernel



Posterior with RBF Kernel



GP: Bayesian inference

- Bayesian inference involves high-dimensional integration in general.

- For regression, can perform inference exactly because all the integrals are 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 inference to doing linear algebra.

Shallow Neural Networks and Gaussian Process Priors

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.**

Shallow Neural Networks and Gaussian Process Priors

Justification: Central Limit Theorem

$$z_i^1(x) = b_i^1 + \sum_{j=1}^{N_1} W_{ij}^1 x_j^1(x), \quad x_j^1(x) = \phi\left(b_j^0 + \sum_{k=1}^{d_{in}} W_{jk}^0 x_k\right)$$

In the infinite width limit, every finite collection of $\{z_i^1(x^\mu)\}_{i,\mu}$ will have a joint multivariate Normal distribution: definition of GP.

Let's suppose e.g.: $W_{i,j}^1 \sim \mathcal{N}(0, \sigma_w^2/N_1), b_i^1 \sim \mathcal{N}(0, \sigma_b^2)$

$$\mu^1(x) = \mathbb{E}\left[z_i^1(x)\right] = 0$$

$$K^1(x, x') \equiv \mathbb{E}\left[z_i^1(x) z_i^1(x')\right] = \sigma_b^2 + \sigma_w^2 \, \mathbb{E}\left[x_i^1(x) x_i^1(x')\right] \equiv \sigma_b^2 + \sigma_w^2 C(x, x')$$

(Note that outputs are independent because they have Normal joint and zero covariance.)

Deep Neural Networks and Gaussian Process Priors

What is the prior over functions implied by the prior over parameters, for ***deep neural networks?***

Consider a network which:
- is **deep (L layers)**
- is **fully-connected**
- has **i.i.d. prior over parameters (such that it give a sensible limit)**

Then the distribution on its output is also a GP **in the limit of infinite layer width.**

$$z_i^l(x) = b_i^l + \sum_{j=1}^{N_l} W_{ij}^l x_j^l(x), \quad x_j^l(x) = \phi(z_j^{l-1}(x)).$$

Suppose (from induction), that $z_j^{l-1} \sim \mathcal{GP}(0, K^{l-1})$, and different units *j* are independent.

Then similarly, from Central Limit Theorem: $z_i^l \sim \mathcal{GP}(0, K^l)$

NNGP covariance function

Recursion relation is:

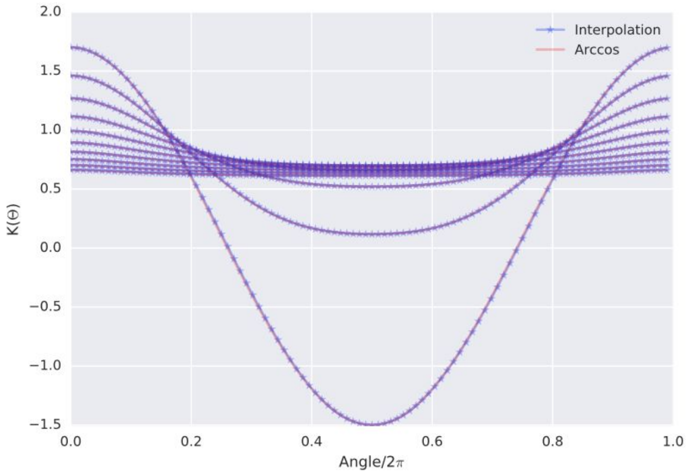
$$K^l(x, x') = \sigma_b^2 + \sigma_w^2 \, F_\phi\Big(K^{l-1}(x, x'),\ K^{l-1}(x, x),\ K^{l-1}(x', x')\Big)$$

For some non-linearities, can compute F_ϕ exactly (e.g. see Cho and Saul, '09; A. Daniely, et al. '16).

For ReLU:

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

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

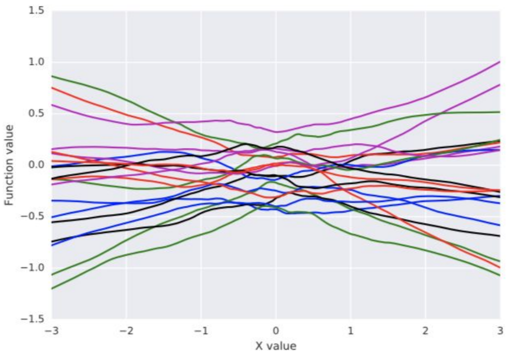


ReLU kernel for various depths (larger depth gives flatter curves).

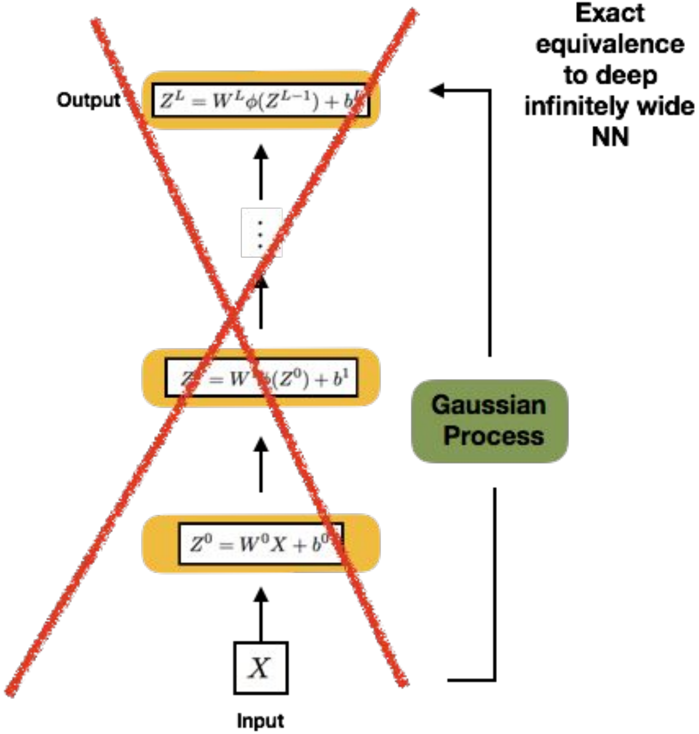
Deep Neural Networks and Gaussian Process Priors

Altogether, for a depth L network, we summarize this:

$$z^L \sim \mathcal{GP}(0, K^L)$$
$$K^L = \sigma_b^2 + \sigma_w^2 F_\phi(K^{L-1})$$



Samples from a GP neural network prior with depth 10.



Reference for more formal treatment

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

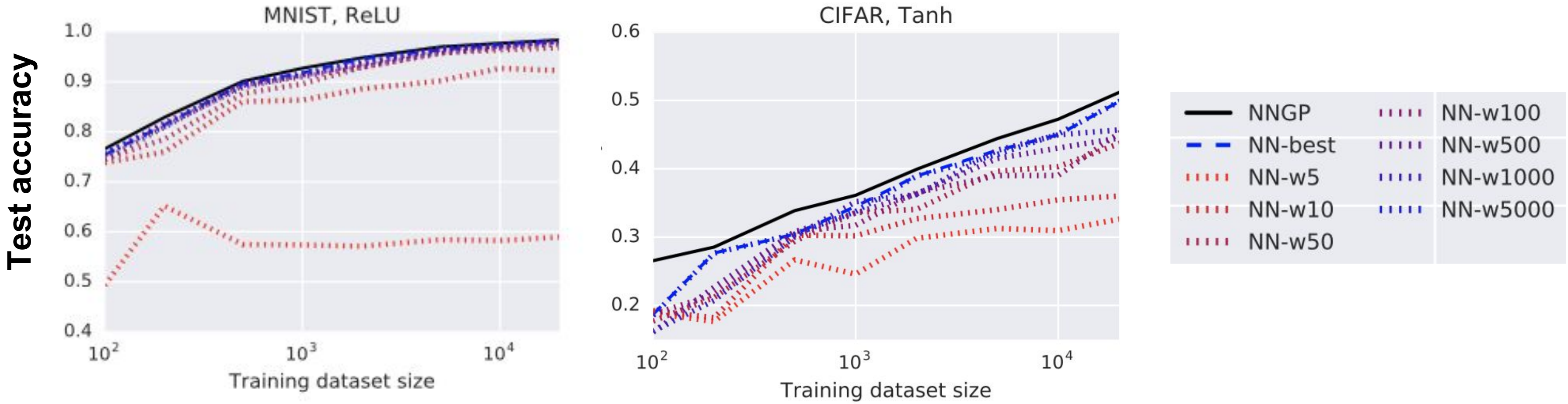
- R. Novak et al., ICLR 2019
  - Bayesian Deep Convolutional Networks with Many Channels are Gaussian Processes
  - https://arxiv.org/abs/1810.05148
  - Appendix E

Experiments

Experimental setup

- Datasets: MNIST, CIFAR-10
- 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 using random / grid search
  - Weight / bias variances, optimization hyperparameters (for NN)
- NN: `SGD’ trained opposed to Bayesian training. In practice, Adam optimizer was used (qualitatively similar).
- NNGP: standard exact Gaussian process regression, 10 independent outputs

Performance of wide networks approaches NNGP



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

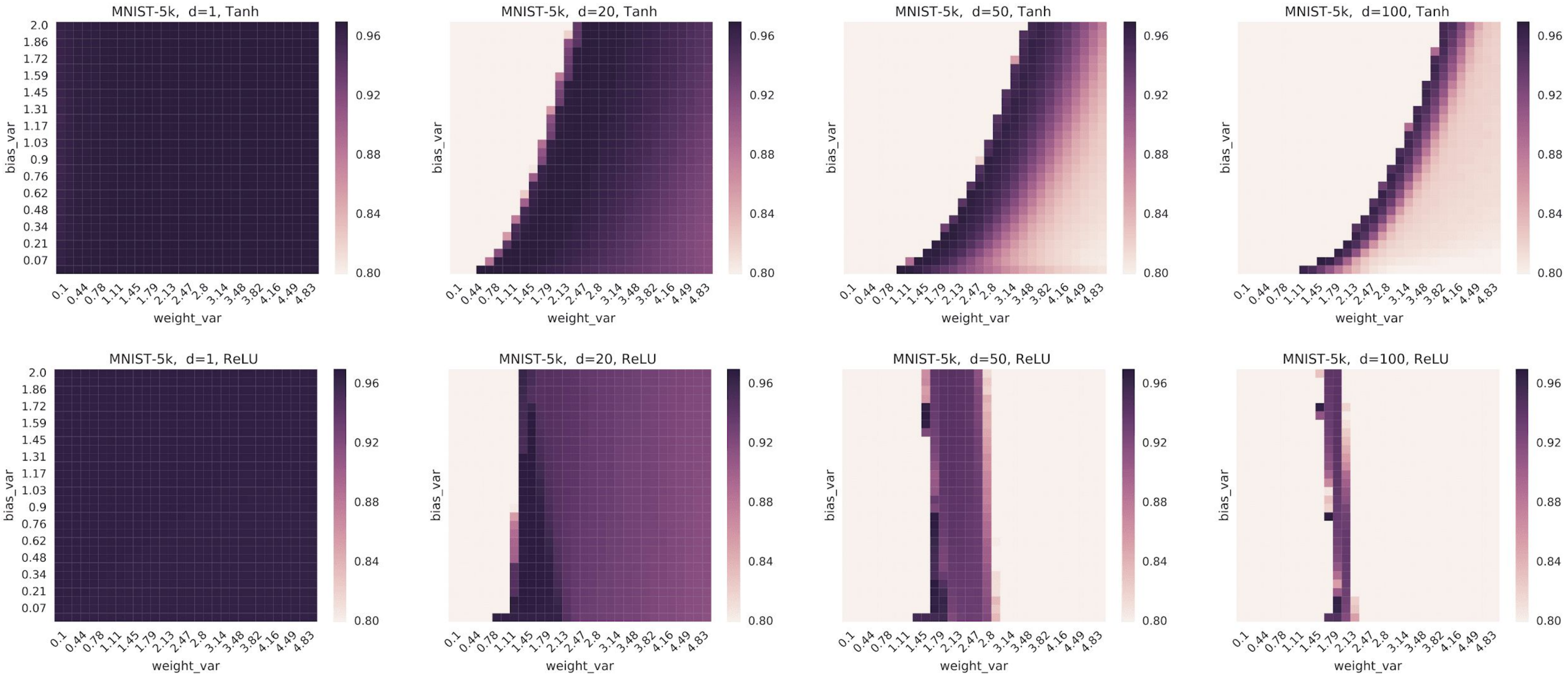
Finite width networks trained with SGD vs NNGP

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

NNGP hyperparameter dependence

Test accuracy



Uncertainty

- Neural networks are good at making predictions, but does not naturally provide uncertainty estimates

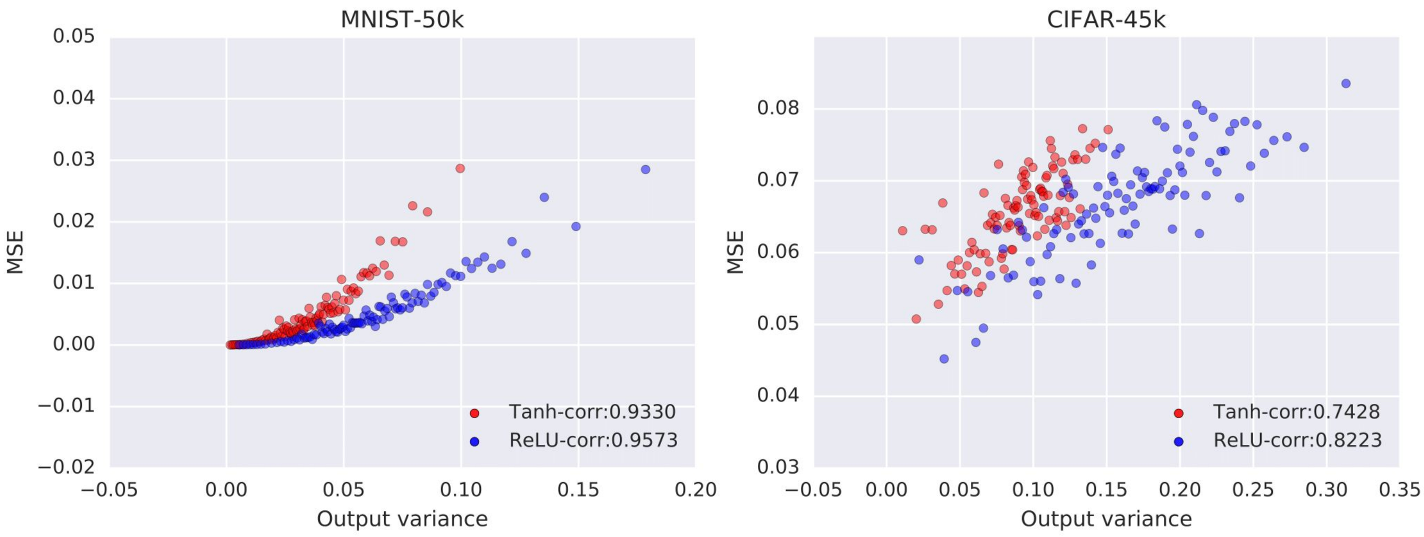
- Bayesian methods incorporates uncertainty

- In domains where uncertainty of prediction is important, GP has been useful

- 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: how good are the estimates?



X: predicted uncertainty

Y: realized MSE

* averaged over 100 points binned by predicted uncertainty

Empirical error is well correlated with uncertainty predictions

Log marginal likelihood (model selection)

$$\log p(t|\theta) = -\frac{1}{2}t^T\left(K_{\mathcal{D}\mathcal{D}}(\theta) + \sigma_\epsilon^2\mathbb{I}\right)^{-1}t - \frac{1}{2}\log\det\left(K_{\mathcal{D}\mathcal{D}}(\theta) + \sigma_\epsilon^2\mathbb{I}\right) + \text{const}$$

- Neural network hyperparameters: depth, weight / bias variance, non-linearity
- No validation set is required to select model hyperparameters. Evaluate on train data.
- K_{DD} is deterministic and differentiable, implemented in Tensorflow. Can backprop!

Future works

NNGP correspondence opens up interesting angles to further analyze deep neural networks.

Published as a conference paper at ICLR 2019

BAYESIAN DEEP CONVOLUTIONAL NETWORKS WITH MANY CHANNELS ARE GAUSSIAN PROCESSES

Roman Novak †, **Lechao Xiao** † *, **Jaehoon Lee** ‡ *, **Yasaman Bahri** ‡ *, **Greg Yang**°,
Daniel A. Abolafia, **Jeffrey Pennington**, **Jascha Sohl-Dickstein**
Google Brain, °Microsoft Research AI
{romann, xlc, jaehlee, yasamanb, gregyang@microsoft.com,
danabo, jpennin, jaschasd}@google.com

- Practical usage of NNGP
- Extension to other network architectures
  - **Convolutional** / Residual [Novak et al., ICLR 2019, Garriga-Alonso et al., ICLR 2019]
  - Batch normalization, self-attention, recurrent, ...
- Systematic finite width correction

Gradient descent dynamics of wide networks

Gaussian Predictions from Gradient Descent Training of Wide Neural Networks

Jaehoon Lee*, **Lechao Xiao***, **Jascha Sohl-Dickstein**, **Jeffrey Pennington**
Google Brain
{jaehlee, xlc, jaschasd, jpennin}@google.com

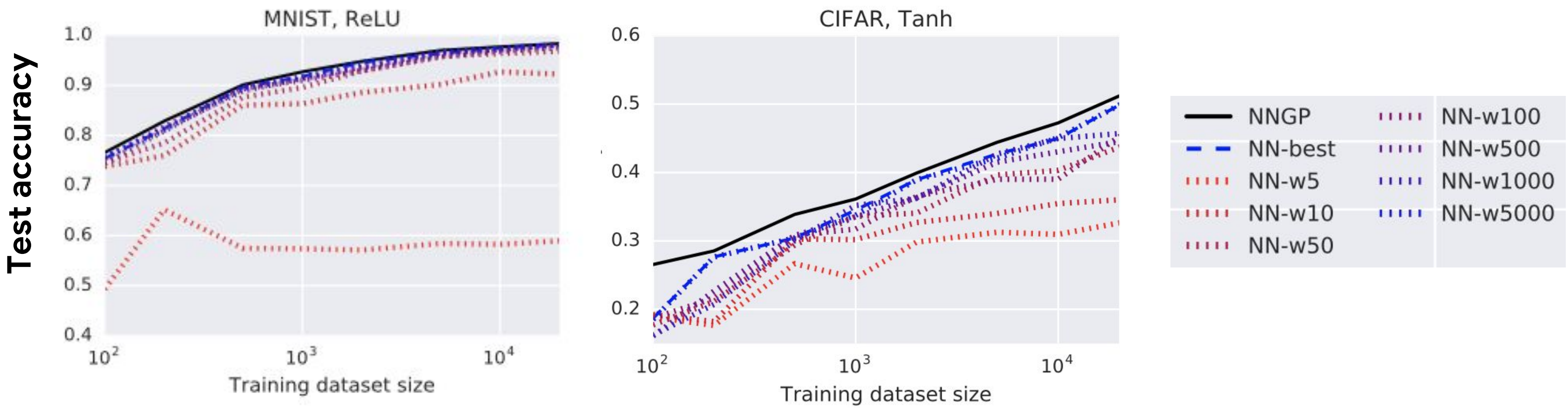
NeurIPS Bayesian Deep Learning Workshop 2019

Wide neural networks of any depth evolve as linear models under gradient descent

Jaehoon Lee *[1] **Lechao Xiao** *[1] **Sam Schoenholz** [1]
Yasaman Bahri [1] **Jascha Sohl-Dickstein** [1] **Jeffrey Pennington** [1]

Available at arXiv soon

Recall : empirical observations



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

How similar is gradient descent based training to the Bayesian inference?

Motivations:

- Bayesian inference VS gradient descent training
- Tractable learning dynamics of deep neural networks

Our contributions:

- Wide neural networks' training dynamics under gradient descent become surprisingly simple
  - Effectively replace NN by its first-order Taylor expansion around init parameters
  - Linear model captures the NN training dynamics
- Analytic dynamics for MSE loss, simple generalization to xent loss / momentum optimizer / practical networks (wide residual network)
- Analytic output distribution dynamics for MSE loss: not equal to NNGP posterior

Gradient descent dynamics (continuous time)

$$\mathcal{L} = \sum_{(x,y)\in\mathcal{D}} \ell(f_t(x,\theta), y).$$

$$\dot{\theta}_t = -\eta \nabla_\theta f_t(\mathcal{X})^T \nabla_{f_t(\mathcal{X})}\mathcal{L}$$

$$\dot{f}_t(\mathcal{X}) = \nabla_\theta f_t(\mathcal{X})\,\dot{\theta}_t = -\eta\,\hat{\Theta}_t(\mathcal{X},\mathcal{X})\nabla_{f_t(\mathcal{X})}\mathcal{L}$$

Neural Tangent Kernel (NTK) [Jacot et al. 2018]

$$\hat{\Theta}_t = \nabla_\theta f_t(\mathcal{X})\nabla_\theta f_t(\mathcal{X})^T = \sum_{l=1}^{L+1} \nabla_{\theta^l} f_t(\mathcal{X})\nabla_{\theta^l} f_t(\mathcal{X})^T.$$

Linearized networks

$$f_t^{\text{lin}}(x) \equiv f_0(x) + \nabla_\theta f_0(x)\,\omega_t$$

$$\omega_t \equiv \theta_t - \theta_0$$

$$\dot{\omega}_t = -\eta \nabla_\theta f_0(\mathcal{X})^T \nabla_{f_t^{\text{lin}}(\mathcal{X})} \mathcal{L}$$

$$\dot{f}_t^{\text{lin}}(x) = -\eta\,\hat{\Theta}_0(x, \mathcal{X}) \nabla_{f_t^{\text{lin}}(\mathcal{X})} \mathcal{L}\,.$$

Dynamics fully determined by initialization objects: **simple ODE**

Tractable dynamics for wide networks

- Remarkably Jacot et al. 2018 showed that

$$\sup_{t\in[0,T]} \|\hat{\Theta}_t - \hat{\Theta}_0\|_F = O(\min\{n_1, \ldots, n_L\}^{-1/2})$$

- For MSE loss, we also show that

$$\sup_{t\in[0,T]} \|f_t(\mathcal{X}) - f_t^{\text{lin}}(\mathcal{X})\|_2 = O(\sup_{t\in[0,T]} \|\hat{\Theta}_t - \hat{\Theta}_0\|_F),$$

- Linearized networks training dynamics converges to that of original network as width increases

Predictive output distribution

- Sample-then-optimize posterior sampling (Matthews et al., 2017)
  - Randomly initialize networks
  - Optimize (via GD) using training data
  - Predictive output distribution over ensemble of different initialization

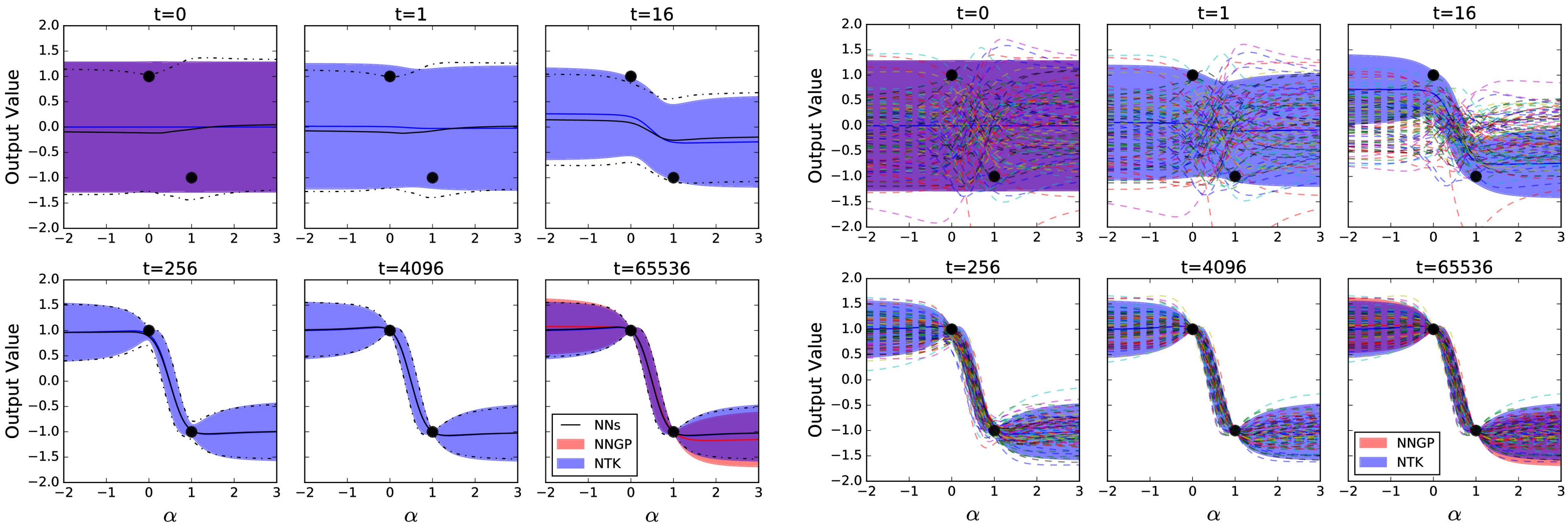
- For wide networks
  - Only optimize readout weights : interpolation between prior and posterior of NNGP
  - Optimize all the weights: As width increases, **ensembles** of random wide neural networks trained with (stochastic) gradient descent converges to a Gaussian process

$$\mu(x) = \Theta(x, \mathcal{X})\Theta^{-1}(I - e^{-\eta\Theta t})\mathcal{Y} \tag{15}$$

$$\begin{aligned}\Sigma(x) = {} & \mathcal{K}(x, x) - 2\Theta(x, \mathcal{X})\Theta^{-1}(I - e^{-\eta\Theta t})\mathcal{K}(x, \mathcal{X})^T \\ & + \Theta(x, \mathcal{X})\Theta^{-1}(I - e^{-\eta\Theta t})\mathcal{K}\Theta^{-1}(I - e^{-\eta\Theta t})\Theta(x, \mathcal{X})^T .\end{aligned} \tag{16}$$

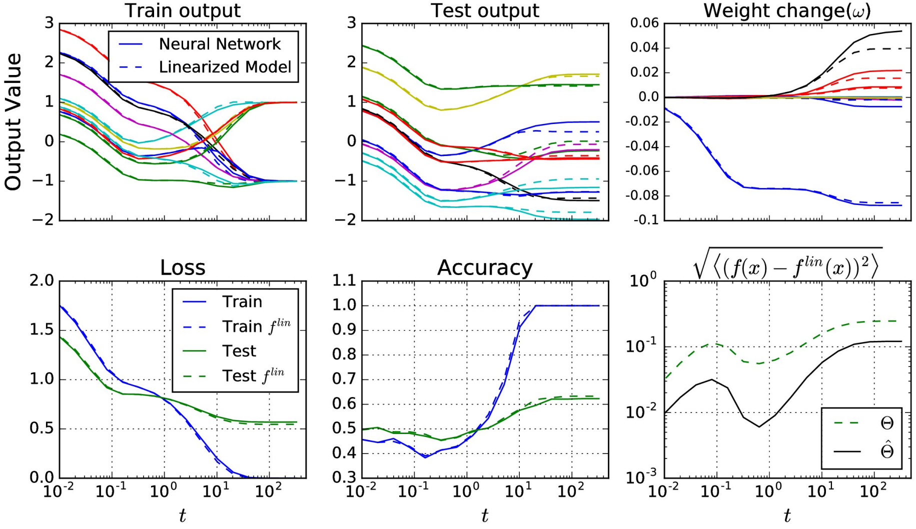
Experiments

NN posterior vs GP posterior

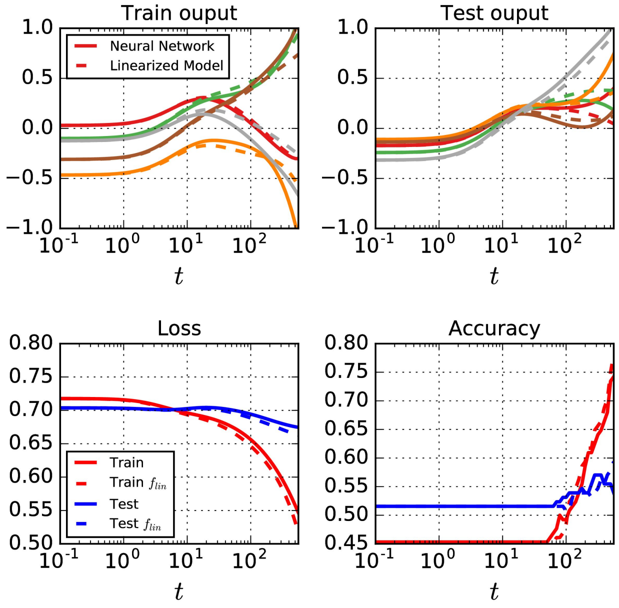


Comparison of training dynamics linearized network vs original network

FC / MSE / GD



WResNet* / xent / momentum



CIFAR binary classification with 128 samples

Thank you! Questions?

NTK parameterization of NN

Conventional

$$z^l = W^l x^l + b^l$$

$$W_{ij}^l \sim \mathcal{N}(0, \sigma_w^2 / n^l)$$

NTK [Jacot et al 2018]

$$z^l = \frac{1}{\sqrt{n^l}} \tilde{W}^l x^l + b^l$$

$$\tilde{W}_{ij}^l \sim \mathcal{N}(0, \sigma_w^2)$$

Computes the same functions / modifies dynamics / universal learning rates (absorb 1/n)

Deep Neural Networks and Gaussian Process Priors

$$z_i^l(x) = b_i^l + \sum_{j=1}^{N_l} W_{ij}^l x_j^l(x), \quad x_j^l(x) = \phi(z_j^{l-1}(x)).$$

$$K^l(x, x') \equiv \mathbb{E}\left[z_i^l(x) z_i^l(x')\right] = \sigma_b^2 + \sigma_w^2\, \mathbb{E}_{z_i^{l-1} \sim \mathcal{GP}(0, K^{l-1})}\left[\phi(z_i^{l-1}(x))\phi(z_i^{l-1}(x'))\right]$$

The calculation of the expectation is a 2D Gaussian integral:

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

As a result:

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

Base case in the recursion:

$$K^0(x, x') = \mathbb{E}\left[z_j^0(x) z_j^0(x')\right] = \sigma_b^2 + \sigma_w^2 \left(\frac{x \cdot x'}{d_{\text{in}}}\right)$$