Google

Wide, Deep Neural Networks as Gaussian Processes

Towards an Understanding of Neural Networks
In the Regime of Large Width

Yasaman Bahri
Google Brain

KITP, “At the Crossroad of Physics and Machine Learning”

Feb 2019

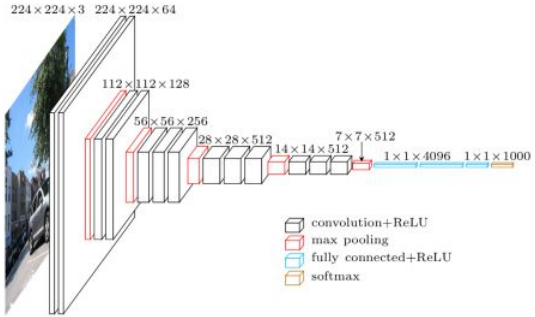
Introduction

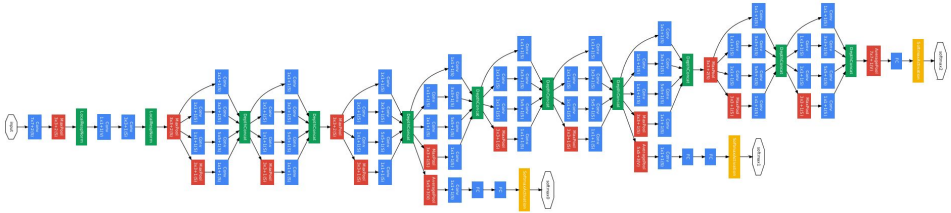
Many open questions in deep learning today.

“Vanilla” supervised learning setup:

--Training data drawn from some fixed distribution D
--Choose an architecture, unknown parameters.
--Run an algorithm for learning.
--Hyperparameters that can be further tuned.
--Ultimately: performance on test set (unseen, drawn from same distribution).

At present, not a lot is understood.





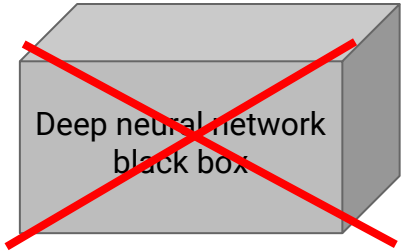
Google

Introduction

Just a sampling of open questions in deep learning theory:

--What are the conditions that yield good trainability and generalization (ultimate performance)?
--How to select an architecture (massive design space)? How to reduce hyperparameter search?
--Can we learn with less data, smaller networks?
--What role does the (complexity, structure of the) task itself play?

And ***many*** others….



Pieces of the problem are intimately entangled with one another.

Combination of a scientific, engineering, and design problem.

On the other hand, maybe not as experimentally limited as in the (biological, physical) sciences.

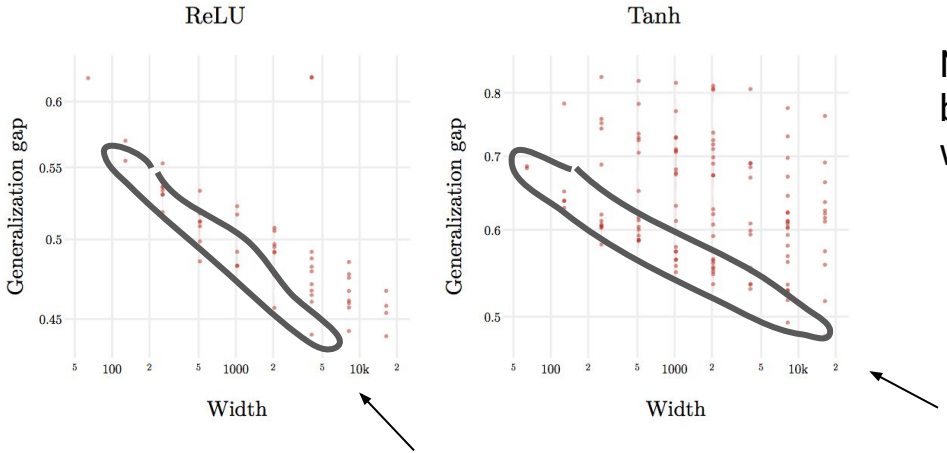
As always, a difficulty is to maintain balance between simplifying problem enough, but danger of making the problem too simple.

Where to begin?

Which regimes appear to be important to understand?

Observation: why do *"large"* networks generalize well? Why aren't you plagued with overfitting when you add more parameters?(*)

Unexplained by older, classical results in statistical learning theory.



Networks trained with stochastic optimization: best achievable test performance improves with width.

Generalization gap for 5 hidden layer ***fully-connected*** networks on CIFAR-10 (fixed depth, increasing width). Filtered for 100% classification training accuracy.

See also B. Neyshabur, et al. NeurIPS 2017.

What happens in limit of infinite width?

(*)Though won't be answering this question in this talk.

Talk Outline

I. Correspondence between infinitely wide **fully-connected** neural networks and Gaussian processes
   A. Focus on Bayesian inference
   B. Brief snapshot of recent results on evolution under gradient descent

II. Correspondence between infinitely wide **convolutional** neural networks and Gaussian processes
   A. Empirical results on Bayesian inference vs. gradient descent training

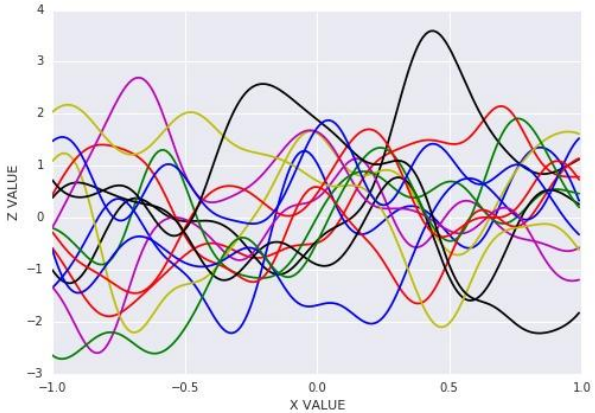
Review: Gaussian Processes (GPs)

Recall the definition:

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

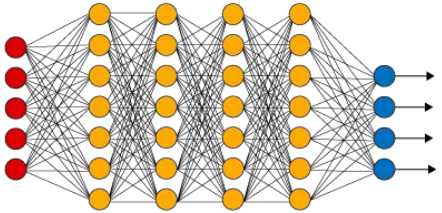
$$\vec{\mu} = \begin{bmatrix} \mu(x_1) \\ \vdots \\ \mu(x_n) \end{bmatrix}, \mathbf{K} = \begin{bmatrix} K(x_1, x_1) & \cdots & K(x_1, x_n) \\ \vdots & \ddots & \vdots \\ K(x_n, x_1) & \cdots & K(x_n, x_n) \end{bmatrix}$$

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



Samples from GP with RBF Kernel

Review: Approaches to Inference (+ Notation)



- Problem: Input, output pairs (x^μ, y^μ) constituting dataset $\mathcal{D}$, learn to predict for new inputs
- Model: Fully-connected deep neural network has recursion relation

$$z_i^l(x) = b_i^l + \sum_j W_{ij}^l \phi(z_j^{l-1}(x)) \qquad \textbf{Base Case:} \qquad z_i^0(x) = b_i^0 + \sum_j W_{ij}^0 x_j$$

- Optimization: feed into a loss function and e.g. do empirical risk minimization (with some initialization)

$$\mathcal{L}_\theta \equiv \frac{1}{M} \sum_{\mu=1}^{M} \ell(z_\theta(x^\mu), y^\mu) \rightarrow \text{solve: argmin } \mathcal{L}_\theta$$

- Bayesian approach: begin with prior belief about the model parameters (currently, model is fixed) and adjust belief based on data, in a manner consistent with Bayes rule

$$p(z_*|\mathcal{D}, x_*) = \int d\theta\, p(z_*|\theta, x_*)\, p(\theta|\mathcal{D}) = \frac{1}{p(\mathcal{D})} \int d\theta\, p(z_*|\theta, x_*)\, p(\mathcal{D}|\theta)\, p(\theta)$$

posterior

prior

A shift in perspective

Discussion in deep learning has mostly focused on parametric point of view.

$$p(z_*|\mathcal{D}, x_*) = \int d\theta \, p(z_*|\theta, x_*) \, p(\theta|\mathcal{D})$$

But parameters in a neural network might lack direct meaning. What we really care about, and possibly might have a better sense for, are the functions realized by the network (nonparametric perspective).

Instead, integrate over the distribution of functions that corresponds to this (e.g., see “Bayesian field theory”).

Radford Neal, “Priors for Infinite Networks,” 1994:
Given some distribution (in this case, the prior) over parameters of the neural network, what distribution over functions is induced by the network?

Shallow Neural Networks and GP Priors

Given some distribution (in this case, the prior) over parameters of the network, what is the distribution over functions computed by the network?

Neal answered this for single-hidden layer networks.

Specifically, consider a NN which:

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

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

Radford Neal, "Priors for Infinite Networks," 1994.

Shallow Neural Networks and GP Priors

Follows from the 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.

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

The parameters of the GP are: $\mu^1(x) = \mathbb{E}\left[z_i^1(x)\right] = 0$

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

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

Radford Neal, "Priors for Infinite Networks," 1994.

Deep Neural Networks and GP 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 recall that different units *j* are independent.

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

J. Lee, **YB***, R. Novak, S. Schoenholz, J. Pennington, J. Sohl-Dickstein. "Deep Neural Networks as Gaussian Processes." ICLR 2018.*

Deep Neural Networks and GP 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:

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

Base case in the recursion:

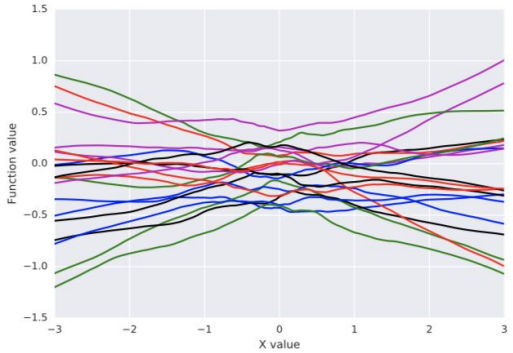
$$K^0(x,x') = \mathbb{E}\left[z_j^0(x) z_j^0(x')\right] = \sigma_b^2 + \sigma_w^2 \left(\frac{x \cdot x'}{d_{\text{in}}}\right)$$

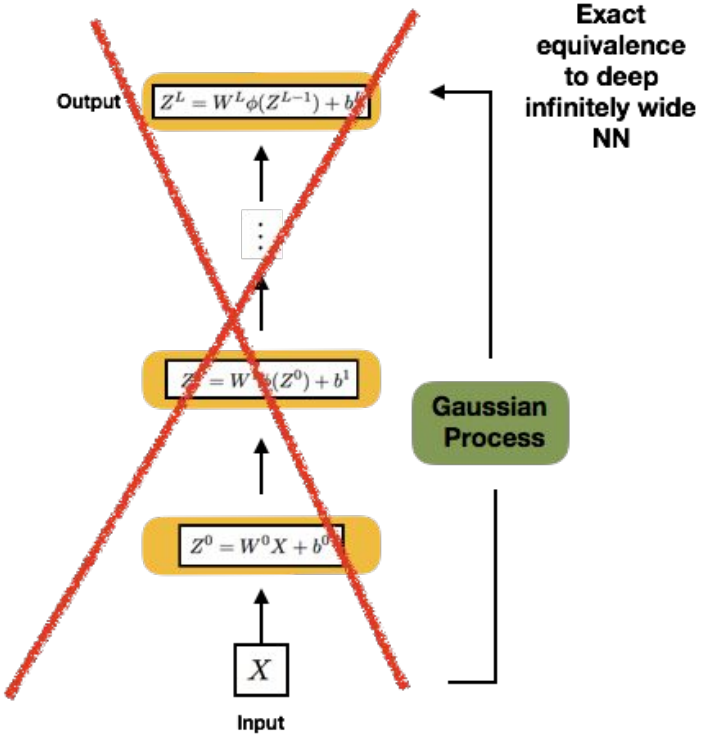
Deep Neural Networks and Gaussian Process Priors

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.

Properties of the GP corresponding to a deep network

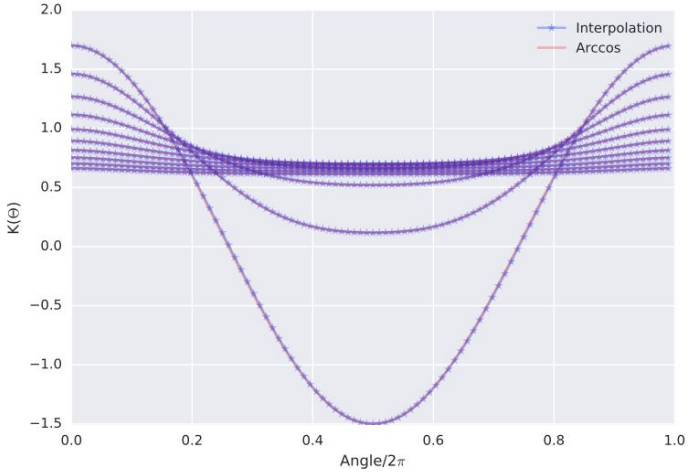
Recall the defining recursion relation for the covariance function:

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

For some nonlinearities, can compute F_ϕ exactly, such as for ReLU:[1]

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

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



ReLU kernel for various depths (flatter curves with larger depths).

[1]. Cho and Saul, “Kernel Methods for Deep Learning,” 2009.

Bayesian inference with a GP prior

$$\text{Data } \mathcal{D} = \{(x^1, t^1), ..., (x^n, t^n)\}$$
$$\text{Training inputs } \boldsymbol{x} \equiv (x^1, ..., x^n)$$
$$\text{Training targets } \boldsymbol{t} \equiv (t^1, ..., t^n)$$
$$\text{Function values } \boldsymbol{z} \equiv (z^1, ..., z^n)$$
$$\text{Test point } x^*$$

- Bayesian inference in function space:

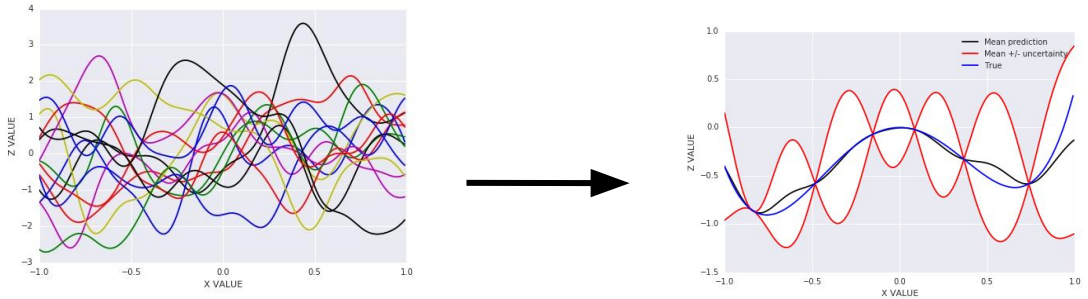
$$P(z^*|\mathcal{D}, x^*) = \int d\boldsymbol{z}\, P(z^*|\boldsymbol{z}, \boldsymbol{x}, x^*)\, P(\boldsymbol{z}|\mathcal{D})$$

For regression (specialize to this case in this talk), can do Gaussian integrals exactly. Obtain result:

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

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

Reduces inference to linear algebra.



[1]. C. K. Williams, “Computing with Infinite Networks,” 1997.

Experiments

Comparison of:

- Bayesian inference using this GP (corresponding to a fully-connected NN)
- Finite-width deep networks trained with (some variant of) gradient descent

Treating classification as a regression task on one-hot targets, on MNIST and CIFAR-10.

- Fully-connected architecture, ReLU/Tanh nonlinearities
- Hyperparameter optimized

Kernel computation: either analytic form or using numerical interpolation with some tricks for speed-up.

Bayesian inference requires matrix inversion (cubic time): we do exactly up to full dataset size.

Performance comparison

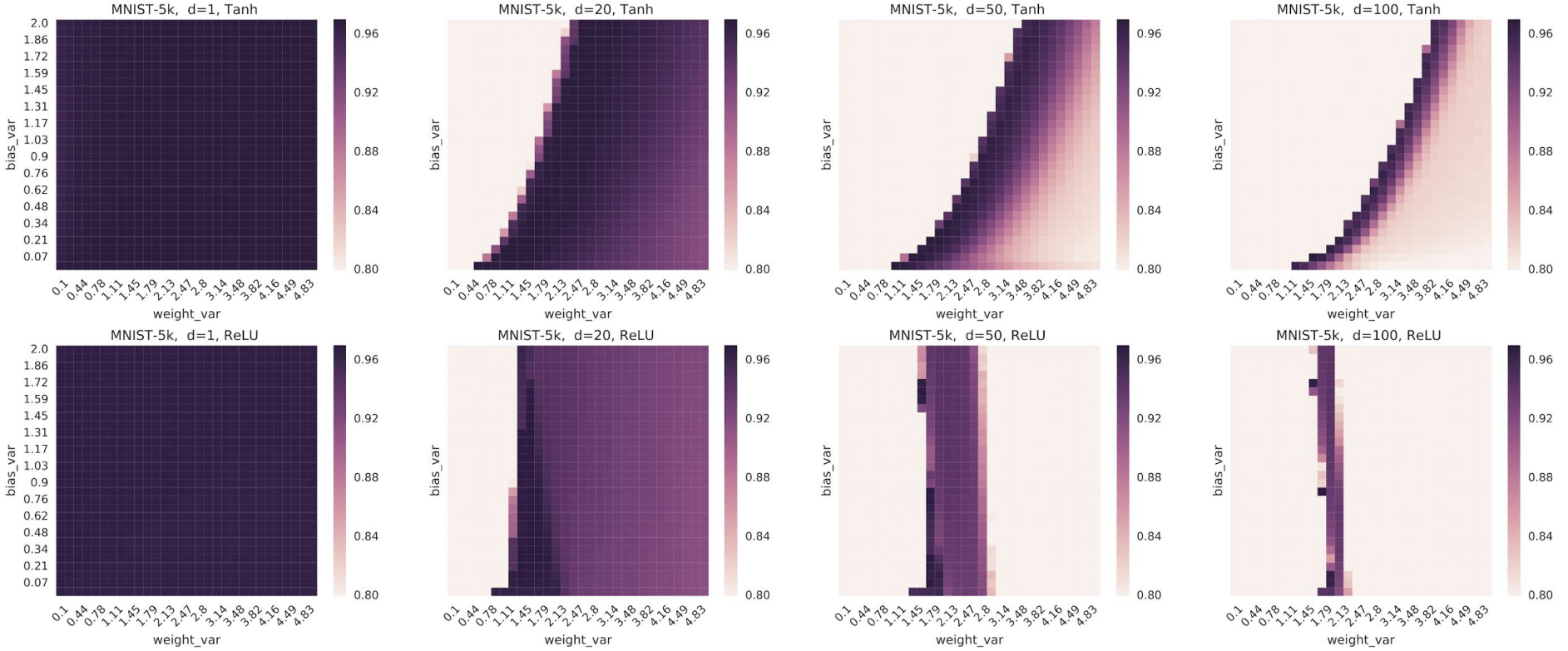
Num training	Model (ReLU)	Test accuracy	Model (tanh)	Test accuracy
MNIST:1k	NN-2-5000-3.19-0.00	0.9252	NN-2-1000-0.60-0.00	0.9254
	GP-20-1.45-0.28	**0.9279**	GP-20-1.96-0.62	0.9266
MNIST:10k	NN-2-2000-0.42-0.16	0.9771	NN-2-2000-2.41-1.84	0.9745
	GP-7-0.61-0.07	0.9765	GP-2-1.62-0.28	**0.9773**
MNIST:50k	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

Best performing models selected on validation set.

How to read entries in table:

$\text{NN-depth-width-}\sigma_w^2\text{-}\sigma_b^2$ $\qquad$ $\text{GP-depth-}\sigma_w^2\text{-}\sigma_b^2$

Performance for changing hyperparameters



depth →

Higher test accuracy = **purple color**.

Phase Diagram for Signal Propagation

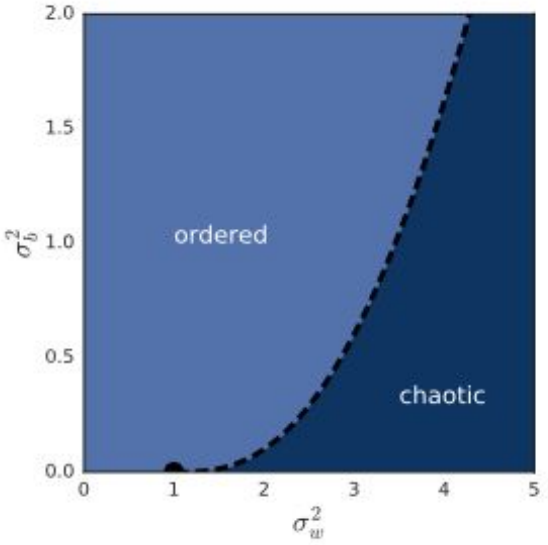
In fact, iterating the recursion relation results in fixed points -- “phases” and transitions between them as a function of hyperparameters.

$$q_x^l \equiv K^l(x,x), \;\; |q_x^l - q^*| \sim e^{-l/\xi_q}$$

$$c_{x,x'}^l \equiv \frac{K^l(x,x')}{\sqrt{K^l(x,x)K^l(x',x')}}, \;\; |c_{x,x'}^l - c^*| \sim e^{-l/\xi_c}$$

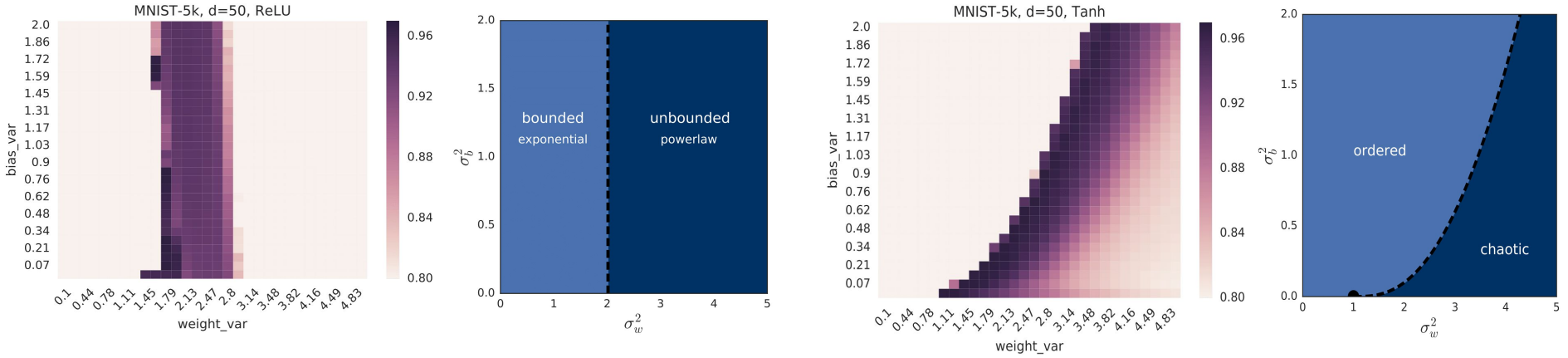
(ξ_q, ξ_c are length scales whose form is known.)

Tanh



S. Schoenholz, et al. “Deep Information Propagation.” ICLR 2017.

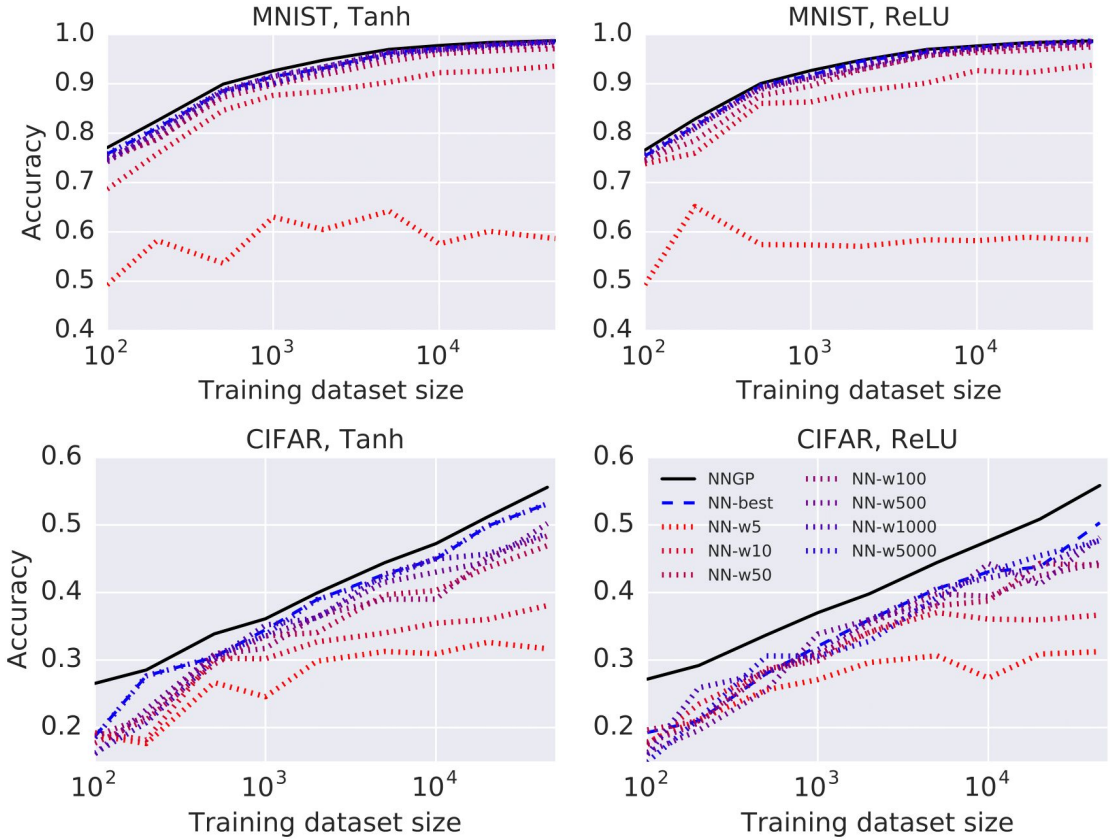
Phase diagrams: experiments vs. theory



Test accuracy as a function of weight, bias variance hyperparameters. Depth = 50 here.
Higher accuracy = **purple color**.

At large depths, structure of covariance function is eroded.

Performance comparison with width and dataset size



Midway Summary

As a method: more powerful class of GPs (due to the kernel), which correspond to NNs.

- Usefulness for scientific applications? Smaller datasets, principled inference, model selection.

So far: a compact description of deep networks (arbitrary depth) in the infinite width limit, at the level of

- Prior
- Posterior from Bayesian inference

For **fully-connected networks**, what we found empirically:

- GP performs competitively, often better, than networks trained with stochastic optimization.
- Performance of the finite-width networks → GP with increasing width.

Reference:

J. Lee*, **YB***, R. Novak, S. Schoenholz, J. Pennington, J. Sohl-Dickstein. “Deep Neural Networks as Gaussian Processes.” ICLR 2018.

Also see: A G. de G. Matthews, et al. ICLR 2018.

Midway Summary

What about evolution with gradient descent?

Subsequent related work:
Relationship between gradient descent and Bayesian inference when you have a linear model:

- see for instance AGG Matthews, “Sample-then-optimize posterior sampling for Bayesian linear models.” NeurIPS Workshop 2017.

In function space, behavior of gradient descent in the infinite width limit recently solved (in a certain operational regime): *evolution with a fixed kernel.*
A. Jacot, et al. “Neural Tangent Kernel.” NeurIPS 2018

Evolution with gradient descent in infinite width

Exact gradient descent equations (parameters and functions):

$$\dot{\theta}_t = -\eta \nabla_\theta z_t(\mathcal{X})^T \nabla_{z_t(\chi)} \mathcal{L}$$
$$\dot{z}_t(\chi) = \nabla_\theta z_t(\chi) \dot{\theta}_t = -\eta \hat{\Theta}_t(\chi, \chi) \nabla_{z_t(\chi)} \mathcal{L}$$
$$\hat{\Theta}_t(x, x') = \nabla_\theta z_t(x) \nabla_\theta z_t(x')^T$$

Kernel stays constant in the limit of infinite width [1]:

$$\lim_{\text{width} \to \infty} \hat{\Theta}_t(x, x') = \hat{\Theta}_0(x, x') = \Theta(x, x')$$

[1]. A. Jacot, et al. “Neural Tangent Kernel.” NeurIPS 2018.

Evolution with gradient descent in infinite width

Mapping back:
Given this function space solution, what dynamics occurs in parameter space?
These dynamics are equivalent to linearizing the network about the initial point [1]. (Sneak peek)

$$z_t^{\text{lin}}(x) \equiv z_0(x) + \nabla_\theta z_0(x)\,\omega_t \qquad \omega_t \equiv \theta_t - \theta_0$$

Substitute

$$\dot{\omega}_t = -\eta \nabla_\theta z_0(\chi)^T \nabla_{z_t^{\text{lin}}}(\chi)\mathcal{L}$$
$$\dot{z}_t^{\text{lin}}(x) = -\eta \hat{\Theta}_0(x,\chi)\nabla_{z_t^{\text{lin}}(\chi)}\mathcal{L}$$

For instance, for squared loss can obtain closed-form solution to training dynamics in parameter space:

$$\omega_t = -\nabla_\theta z_0(\chi)^T \Theta^{-1}(I - e^{-\eta\Theta t})\Big(z_0(\chi) - \mathcal{Y}\Big)$$
$$z_t(\chi) = (I - e^{-\eta\Theta t})\mathcal{Y} + e^{-\eta\Theta t} z_0(\chi)$$

[1]. J. Lee*, L. Xiao*, S. Schoenholz, **YB**, J. Sohl-Dickstein, J. Pennington. “Wide neural networks of any depth evolve as linear models under gradient descent.” To appear.

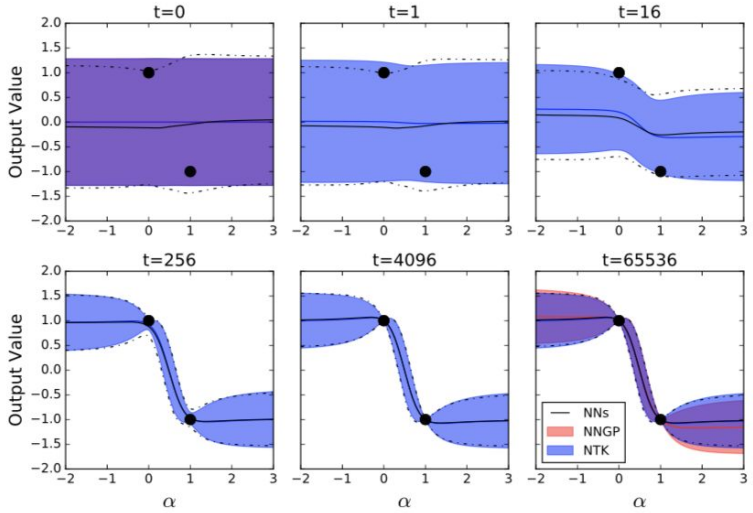
Evolution with gradient descent in infinite width

Further implication: predictive “posterior” distribution in this case is still a GP governed by mean, covariance that can be explicitly written.

$$p(z^*|x^*, \mathcal{D}) \sim \mathcal{N}(\mu(x^*, t), \Sigma(x^*, t))$$

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

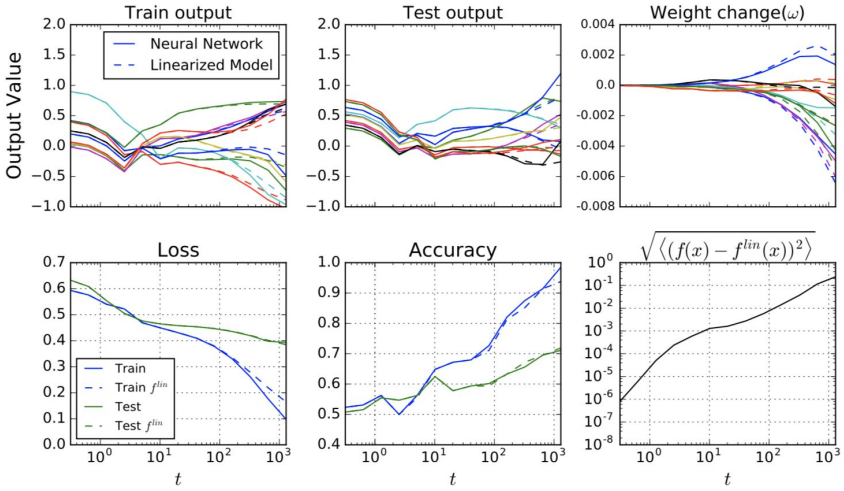
$$\Sigma(x^*) = K(x^*, x^*) - 2\Theta(x, \chi)\Theta^{-1}(I - e^{-\eta\Theta t})K(x, \mathcal{X})^T + \Theta(x, \chi)\Theta^{-1}(I - e^{-\eta\Theta t})K\Theta^{-1}(I - e^{-\eta\Theta t})\Theta(x, \chi)^T$$



Comparison of predictive “posterior” distribution of NNGP, NTK, and ensemble (100 networks) of full-batch gradient descent trained finite-width networks. MSE, depth = 3, width = 8912, tanh fully-connected NN, training points = 128.

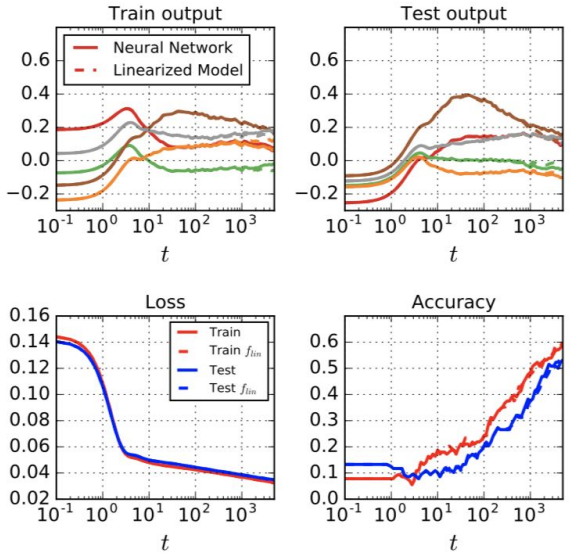
Wide, deep neural networks evolve as linear models

Can describe the dynamics of some real networks well.



NN vs linearized dynamics. Binary CIFAR-10 classification task with MSE loss. Tanh conv network with 3 hidden layers, channels = 512, global average pooling, 128 training points, momentum optimizer.

J. Lee*, L. Xiao*, et al. To appear.



NN vs linearized dynamics, trained with SGD. A type of wide residual network with MSE loss and momentum. BN-Relu-Conv ordering. Channels = 1024, full CIFAR-10 dataset (50k), 10-class output.

Remarks

Gaussian process prior was a useful starting point.

- Characterizes the Bayesian description.
- Plays a role in the evolution with gradient descent.

Function space/nonparameteric perspective was useful for both of these, arguably easier for exposing the simplification.

Mapping back to parameter space corresponds to a linear model using particular random features (the gradients).

Empirically this can describe real-world networks in a certain operating regime well.

Deep Convolutional Networks with (Infinitely) Many Channels

Wide, deep convolutional neural networks

Can play the same game with other models where some relevant dimension is taken to be infinite.

Signal propagation in ultra-deep CNNs (covariance function of pure CNN-GP and the fixed points of the recursion relation with depth)

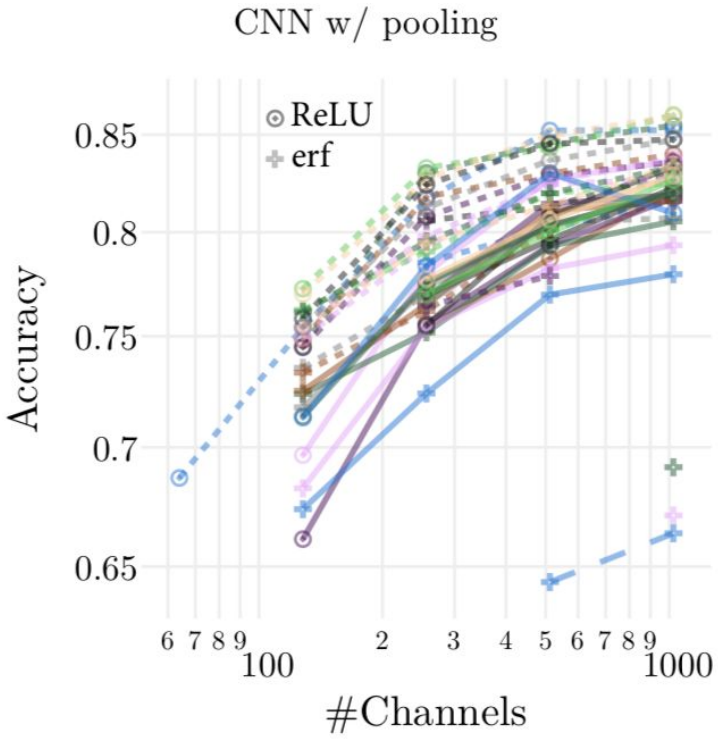
L. Xiao, **YB**, J. Sohl-Dickstein, S. Schoenholz, J. Pennington. “Dynamical Isometry and a Mean Field Theory for CNNs: How to Train 10k-Layer Vanilla CNNs.” ICML 2018.

Convolutional NN:GP correspondence

R. Novak*, L. Xiao*, J. Lee†, **YB**†, G. Yang♦, J. Hron♦, D. Abolafia, J. Pennington, J. Sohl-Dickstein. “Bayesian Deep Convolutional Neural Networks with Many Channels are Gaussian Processes.” ICLR 2019.

Deep Convolutional Networks

Is it reasonable to consider this limit? Overparameterization appears to help CNNs as well.



For *theoretical* setup, consider purely convolutional layers (1D PBC for simplicity). Network performs iterative computation:

$$z^l_{i,\alpha}(x) = b^l_i + \sum_{j=1}^{N} \sum_{\beta=-k}^{k} \omega^l_{ij,\beta}\, \phi(z^{l-1}_{j,\alpha+\beta}(x))$$

N filters, each of size (2k+1), spatial dimensionality d.

Note features of CNNs: locality and weight sharing.

(Full training set, CIFAR-10. Each line corresponds to particular choice of architecture and init hyperparameters, selected over best learning rate/weight decay.)

Deep Convolutional Networks with Infinitely Many Channels

Consider parameters drawn i.i.d: $p(\omega_{ij,\beta}^l) = \mathcal{N}\left(0, \frac{\sigma_\omega^2}{(2k+1)N}\right),\ p(b_i^l) = \mathcal{N}(0, \sigma_b^2)$

Then as the # of channels (filter) becomes infinite → Gaussian Process
(see paper for rigorous proof).

Computation of covariance function:

$$z_{i,\alpha}^l(x) = b_i^l + \sum_{j=1}^{N} \sum_{\beta=-k}^{k} \omega_{ij,\beta}^l\, \phi(z_{j,\alpha+\beta}^{l-1}(x))$$

$$K_{\alpha,\alpha'}^l(x,x') = \mathbb{E}\left[z_{i,\alpha}^l(x) z_{i,\alpha'}^l(x')\right] = \sigma_b^2 + \frac{\sigma_\omega^2}{(2k+1)} \sum_\beta \mathbb{E}\left[\phi(z_{j,\alpha+\beta}^{l-1}(x))\phi(z_{j,\alpha'+\beta}^{l-1}(x'))\right]$$

$$= \sigma_b^2 + \frac{\sigma_\omega^2}{(2k+1)} \sum_\beta \mathcal{C}_{\alpha+\beta,\alpha'+\beta}(x,x') \text{ where}$$

$$\left[\mathcal{C}(K^{l-1})\right]_{\alpha,\alpha'}(x,x') \equiv \mathbb{E}_{z\sim\mathcal{N}(0,K^{l-1})}\left[\phi(z_\alpha(x))\phi(z_{\alpha'}(x'))\right]$$

Transforming GP over spatial locations into GP over classes

CNNs have more architectural design choices:

- Consider pure convolutions until choice of end aggregation into 10 classes.

Involve various transformations of the pure CNN covariance function.

Case 1: Vectorization

- Flatten vector over channels and spatial dimensions → pass through ϕ → FC layer

$$z_i^{\text{final}}(x) = b_i^{\text{final}} + \sum_{j=1}^{N \cdot d} W_{ij}^{\text{final}} \phi(\text{vec}\left[z^L(x)\right])_j$$

$$K_{\text{vect}}^{final}(x, x') = \mathbb{E}\left[z_i^{\text{final}}(x) z_i^{\text{final}}(x')\right] = \sigma_b^2 + \frac{\sigma_\omega^2}{d} \sum_\alpha \left[\mathcal{C}(K^L)\right]_{\alpha,\alpha}(x, x')$$

In this case, you have lost covariances between different spatial locations.

Observation from previous slide: spatial diagonal maps to spatial diagonal (all you need to keep).

Actually, in this case, didn't need weight sharing -- could have gotten same result from locality, by considering locally connected network.

Transforming GP over spatial locations into GP over classes

Case 2: Projection

- Pass through $\phi \rightarrow$ project spatial components onto some vector$\rightarrow$ FC layer

Let $h \in \mathbb{R}^d$ be some fixed projection vector.

$$z_i^{\text{final}}(x) = b_i^{\text{final}} + \sum_{j=1}^{N} W_{ij}^{\text{final}} \left(\sum_{\alpha=1}^{d} \phi(z^L(x))_{j,\alpha} h_\alpha \right)$$

$$K_h^{final}(x, x') = \mathbb{E}\left[z_i^{\text{final}}(x) z_i^{\text{final}}(x') \right] = \sigma_b^2 + \sigma_\omega^2 \sum_{\alpha,\alpha'} h_\alpha h_{\alpha'} \left[\mathcal{C}(K^L) \right]_{\alpha,\alpha'} (x, x')$$

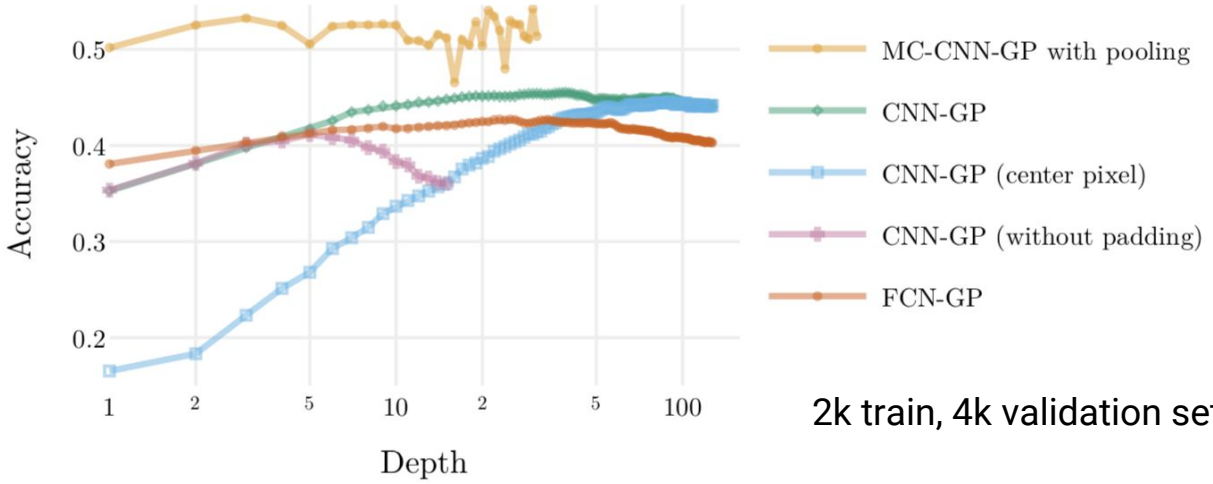
Example: **global average pooling**. Take $h = \frac{1}{d}\mathbf{1}_d$.

$$K_{global}^{final}(x, x') = \sigma_b^2 + \frac{\sigma_\omega^2}{d^2} \sum_{\alpha,\alpha'} \left[\mathcal{C}(K^L) \right]_{\alpha,\alpha'} (x, x')$$

Example: **subsampling one particular pixel**. Take $h = \mathbf{e}_\alpha$.

$$K_{\mathbf{e}_\alpha}^{final}(x, x') = \sigma_b^2 + \sigma_\omega^2 \left[\mathcal{C}(K^L) \right]_{\alpha,\alpha} (x, x')$$

Some empirical results: comparison of various GPs

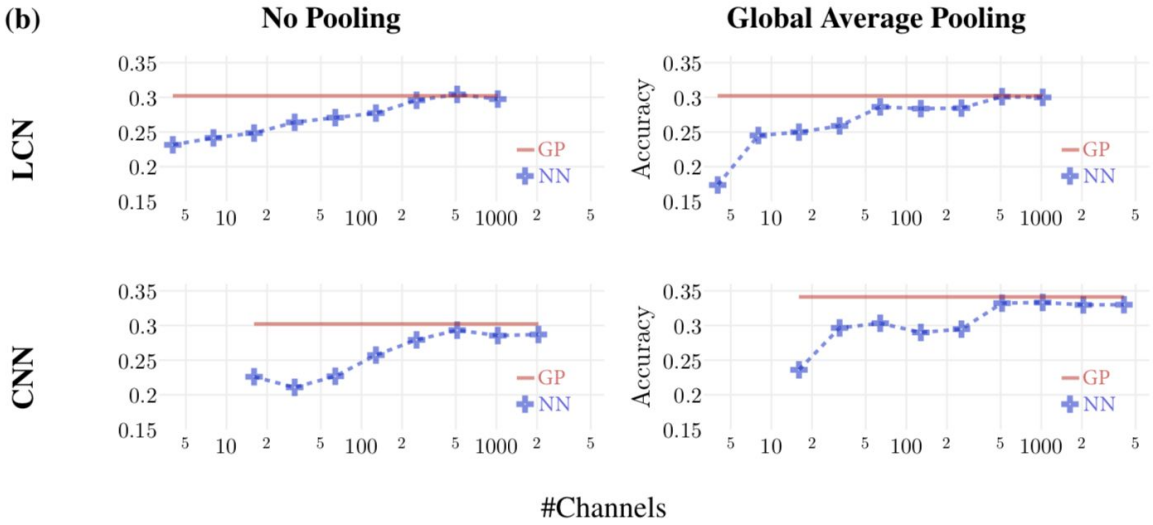


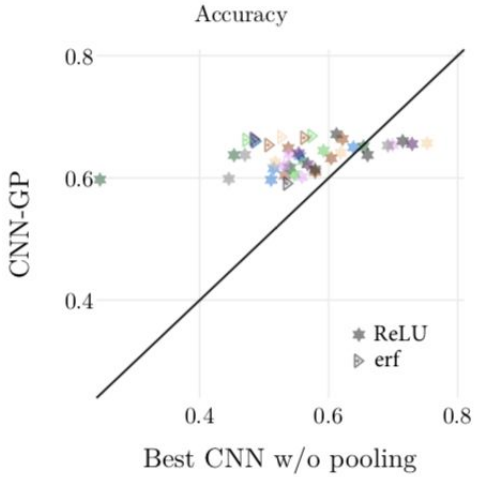
2k train, 4k validation set on CIFAR-10

- CNN-GP (which uses zero padding) better than FCN-GP for most depths (apart from shallow): benefit of local connectivity
- CNN-GP with global pooling best (experiments only available for “empirical”/MC case, not exact)

Empirical results

For same prior (initialization), gradient-descent trained NN accuracy improves and gets closer to the corresponding CNN-GP.





Full CIFAR-10

Each point selected over learning rate, weight decay, batch size. CIFAR-10 downsampled to 8x8, 500 train/4K validation set. Depth=3, erf nonlinearity.

Empirical results: comparison between GPs and GD-NNs

Model	CIFAR10	MNIST	Fashion-MNIST
CNN with pooling	**14.85 (15.65)**	–	–
CNN with ReLU and large learning rate	24.76 (17.64)	–	–
CNN-GP	32.86	0.88	**7.40**
CNN with small learning rate	33.31(22.89)	–	–
CNN with erf (any learning rate)	33.31(22.17)	–	–
FCN-GP	41.06	1.22	8.22
FCN-GP (Lee et al., 2018)	44.34	1.21	–
FCN	45.52 (44.73)	–	–

Hyperparameters: Erf/Relu, learning rate, weight decay, channels/width up to O(1000). 2^{18} gradient steps, batch size 128, zero padding.

Test error (percentage) for the best model within each family (maximizing validation accuracy over hyperparameters)

Full training set

- CNN-GPs are state-of-the-art on CIFAR-10 for GPs without trainable kernels. Outperforms CNN with small learning rate.
- SGD-training of CNN with Relu and large learning rate seems to have beneficial interplay, for absence of pooling
- These differences between CNNs and CNN-GPs are not really observed/not as strong as FC case

State-of-the-art result on GPs without trainable kernels.

Remains to fully disentangle the roles of the various ingredients

Collaborators (Google Brain)

Jaehoon Lee

Roman Novak

Jeffrey Pennington

Sam Schoenholz

Jascha Sohl-Dickstein

Lechao Xiao

Summary

Utility of thinking about functions realized by network.

Compact way of looking at neural networks in a certain limit: through their kernel

- Can characterize Bayesian training
- Can characterize gradient descent evolution

Kernels/GPs as a way of distinguishing architectures.

For many networks in the wild (particularly in science -- small datasets, not a lot of hyperparameter tuning): might be close to this operating point.

(As a side benefit: estimation of uncertainty.)

Thank you!