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

温良剑
CIASH

Contribution

- derive the exact equivalence between infinitely wide deep net- works and GPs
- develop a computationally efficient pipeline to compute the covariance function for these GPs

主要内容

- GPS
- Bayesian Inference
- GPs AND SINGLE-LAYER NN
- GPs AND DEEP NEURAL NETWORKS
- EXPERIMENTAL RESULTS

What is a Gaussian Process?

A *Gaussian process* is a generalization of a multivariate Gaussian distribution to infinitely many variables.

Informally: infinitely long vector $\simeq$ function

> **Definition:** *a Gaussian process is a collection of random variables, any finite number of which have (consistent) Gaussian distributions.* $\square$

A Gaussian distribution is fully specified by a mean vector, μ, and covariance matrix Σ:

$$\mathbf{f} = (f_1, \ldots, f_n)^\top \sim \mathcal{N}(\mu, \Sigma), \quad \text{indexes } i = 1, \ldots, n$$

A Gaussian process is fully specified by a mean function $m(x)$ and covariance function $k(x, x')$:

$$f(x) \sim \mathcal{GP}\big(m(x), k(x, x')\big), \quad \text{indexes: } x$$

Random functions from a Gaussian Process

Example one dimensional Gaussian process:

$$p(f(x)) \sim \mathcal{GP}\big(m(x) = 0,\ k(x, x') = \exp(-\tfrac{1}{2}(x - x')^2)\big).$$

To get an indication of what this distribution over functions looks like, focus on a finite subset of function values $\mathbf{f} = (f(x_1), f(x_2), \ldots, f(x_n))^\top$, for which

$$\mathbf{f} \sim \mathcal{N}(0, \Sigma),$$

where $\Sigma_{ij} = k(x_i, x_j)$.

Then plot the coordinates of f as a function of the corresponding x values.

Bayesian Inference, parametric model

Supervised parametric learning:

- data: **x**, **y**
- model: $y = f_{\mathbf{w}}(x) + \varepsilon$

Gaussian likelihood:

$$p(\mathbf{y}|\mathbf{x}, \mathbf{w}, M_i) \propto \prod_c \exp(-\tfrac{1}{2}(y_c - f_{\mathbf{w}}(x_c))^2/\sigma^2_{\text{noise}})$$

Parameter prior:

$$p(\mathbf{w}|M_i)$$

Posterior parameter distribution by Bayes rule $p(a|b) = p(b|a)p(a)/p(b)$:

$$p(\mathbf{w}|\mathbf{x}, \mathbf{y}, M_i) = \frac{p(\mathbf{w}|M_i)p(\mathbf{y}|\mathbf{x}, \mathbf{w}, M_i)}{p(\mathbf{y}|\mathbf{x}, M_i)}$$

Bayesian Inference, parametric model, cont.

Making predictions:

$$p(y^*|x^*, \mathbf{x}, \mathbf{y}, M_i) = \int p(y^*|\mathbf{w}, x^*, M_i) p(\mathbf{w}|\mathbf{x}, \mathbf{y}, M_i) d\mathbf{w}$$

Marginal likelihood:

$$p(\mathbf{y}|\mathbf{x}, M_i) = \int p(\mathbf{w}|M_i) p(\mathbf{y}|\mathbf{x}, \mathbf{w}, M_i) d\mathbf{w}.$$

Model probability:

$$p(M_i|\mathbf{x}, \mathbf{y}) = \frac{p(M_i) p(\mathbf{y}|\mathbf{x}, M_i)}{p(\mathbf{y}|\mathbf{x})}$$

Problem: integrals are intractable for most interesting models!

Non-parametric Gaussian process models

In our non-parametric model, the "parameters" is the function itself!

Gaussian likelihood:

$$\mathbf{y}|\mathbf{x}, f(x), M_i \sim \mathcal{N}(\mathbf{f}, \sigma^2_{\text{noise}} I)$$

(Zero mean) Gaussian process prior:

$$f(x)|M_i \sim \mathcal{GP}\big(m(x) \equiv 0, \; k(x, x')\big)$$

Leads to a Gaussian process posterior

$$f(x)|\mathbf{x}, \mathbf{y}, M_i \sim \mathcal{GP}\big(m_{\text{post}}(x) = k(x, \mathbf{x})[K(\mathbf{x}, \mathbf{x}) + \sigma^2_{\text{noise}} I]^{-1}\mathbf{y}, \\ k_{\text{post}}(x, x') = k(x, x') - k(x, \mathbf{x})[K(\mathbf{x}, \mathbf{x}) + \sigma^2_{\text{noise}} I]^{-1} k(\mathbf{x}, x')\big)$$

And a Gaussian predictive distribution:

$$y^*|x^*, \mathbf{x}, \mathbf{y}, M_i \sim \mathcal{N}\big(\mathbf{k}(x^*, \mathbf{x})^\top [K + \sigma^2_{\text{noise}} I]^{-1}\mathbf{y}, \\ k(x^*, x^*) + \sigma^2_{\text{noise}} - \mathbf{k}(x^*, \mathbf{x})^\top [K + \sigma^2_{\text{noise}} I]^{-1}\mathbf{k}(x^*, \mathbf{x})\big)$$

Some interpretation

Recall our main result:

$$\mathbf{f}_*|X_*, X, \mathbf{y} \sim \mathcal{N}\big(K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1}\mathbf{y}, \\ K(X_*, X_*) - K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1}K(X, X_*)\big).$$

The mean is linear in two ways:

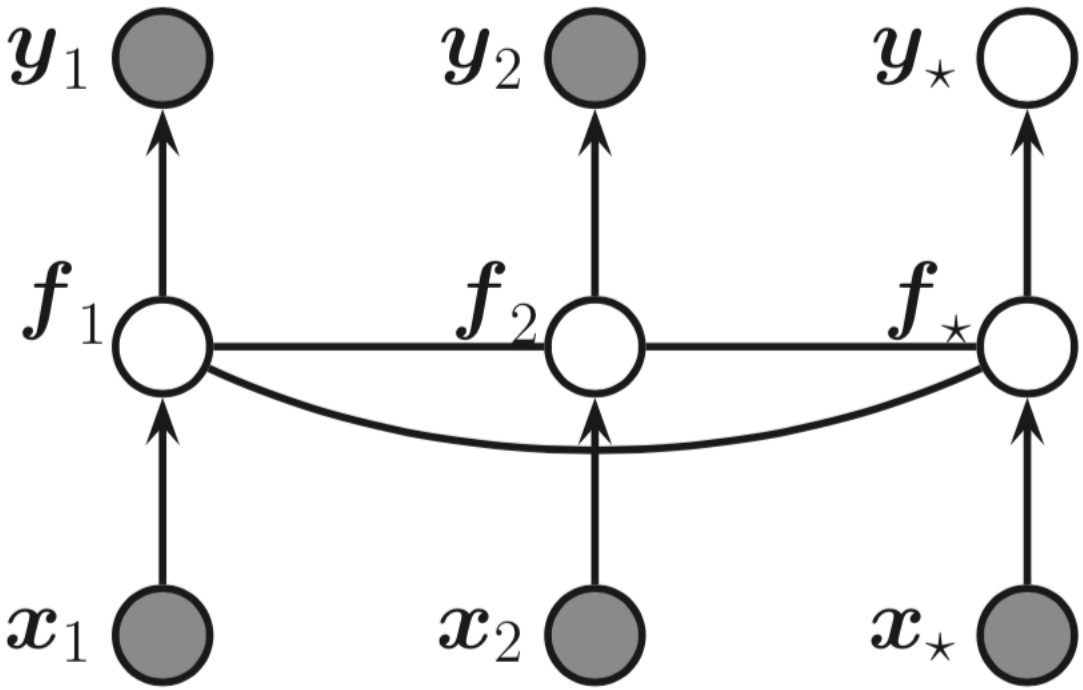
$$\mu(\mathbf{x}_*) \;=\; k(\mathbf{x}_*, X)[K(X, X) + \sigma_n^2]^{-1}\mathbf{y} \;=\; \sum_{c=1}^{n} \beta_c y^{(c)} \;=\; \sum_{c=1}^{n} \alpha_c k(\mathbf{x}_*, \mathbf{x}^{(c)}).$$

The last form is most commonly encountered in the kernel literature.

The variance is the difference between two terms:

$$V(\mathbf{x}_*) \;=\; k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}(\mathbf{x}_*, X)[K(X, X) + \sigma_n^2 I]^{-1}\mathbf{k}(X, \mathbf{x}_*),$$

the first term is the *prior variance*, from which we subtract a (positive) term, telling how much the data X has explained. Note, that the variance is independent of the observed outputs $\mathbf{y}$.


$\boldsymbol{y}_1$
$\boldsymbol{y}_2$
$\boldsymbol{y}_\star$
$\boldsymbol{f}_1$
$\boldsymbol{f}_2$
$\boldsymbol{f}_\star$
$\boldsymbol{x}_1$
$\boldsymbol{x}_2$
$\boldsymbol{x}_\star$

GPs AND SINGLE-LAYER NN

- The ith component of the network output, z_i^1,:

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

W_{ij}^l, b_i^l : weight, bia$l$ and : number of layers

- the weight and bias parameters are taken to be i.i.d.
- the post-activations $x_j^1, x_{j'}^1$ are independent for $j \neq j'$.
- According to the Central Limit Theorem :

  if hiddenwidt $N_1 \to \infty$, $z_i^1(x)$ will be Gaussian distributed

GPs AND SINGLE-LAYER NN

- Multidimensional Central limit theorem:

  Then the sum of the random vectors will be:

$$\begin{bmatrix} X_{1(1)} \\ \vdots \\ X_{1(k)} \end{bmatrix} + \begin{bmatrix} X_{2(1)} \\ \vdots \\ X_{2(k)} \end{bmatrix} + \cdots + \begin{bmatrix} X_{n(1)} \\ \vdots \\ X_{n(k)} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} \left[X_{i(1)}\right] \\ \vdots \\ \sum_{i=1}^{n} \left[X_{i(k)}\right] \end{bmatrix} = \sum_{i=1}^{n} \mathbf{X}_i$$

- $\{z_i^1(x^{\alpha=1}), ..., z_i^1(x^{\alpha=k})\} \longrightarrow$ joint m $\Rightarrow$ $z_i^1 \sim \mathcal{GP}(\mu^1, K^1)$,

- the NN paprameters have zero mean $\Rightarrow$ $\mu^1(x) = \mathbb{E}\left[z_i^1(x)\right] = 0$

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

GPs AND DEEP NN

- Suppose that z_j^{l-1} is a GP, identical and independent for every j
- After l - 1 steps, the network computes:

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

- $z_i^l \sim \mathcal{GP}(0, K^l).$

- The covariance is:

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

GPs AND DEEP NN

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

integrating against the joint distribution of only $z_i^{l-1}(x)$ and $z_i^{l-1}(x')$

- the joint distribution :

  zero mean

  two - dimensional covariance matrix $K^{l-1}(x, x')$, $K^{l-1}(x, x)$, $K^{l-1}(x', x')$
- the shorthand of covariance

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

- analytically computiation:

  the ReLU non- linearity → arccosine kernel
- numerically computiation

GPs AND DEEP NN

- recursive process

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

- compution complexity

$$\mathcal{O}\left(n_g^2 L(n_{\text{train}}^2 + n_{\text{train}} n_{\text{test}})\right)$$

- pipeline to compute the covariance function:

$$\mathcal{O}\left(n_g^2 n_v n_c + L(n_{\text{train}}^2 + n_{\text{train}} n_{\text{test}})\right)$$

EXPERIMENTAL RESULTS

dataset: MNIST, CIFAR-10

setup:

(1) Formulating classification as regression

(2) no dropout

(3) nonlinearities: ReLU and Tanh

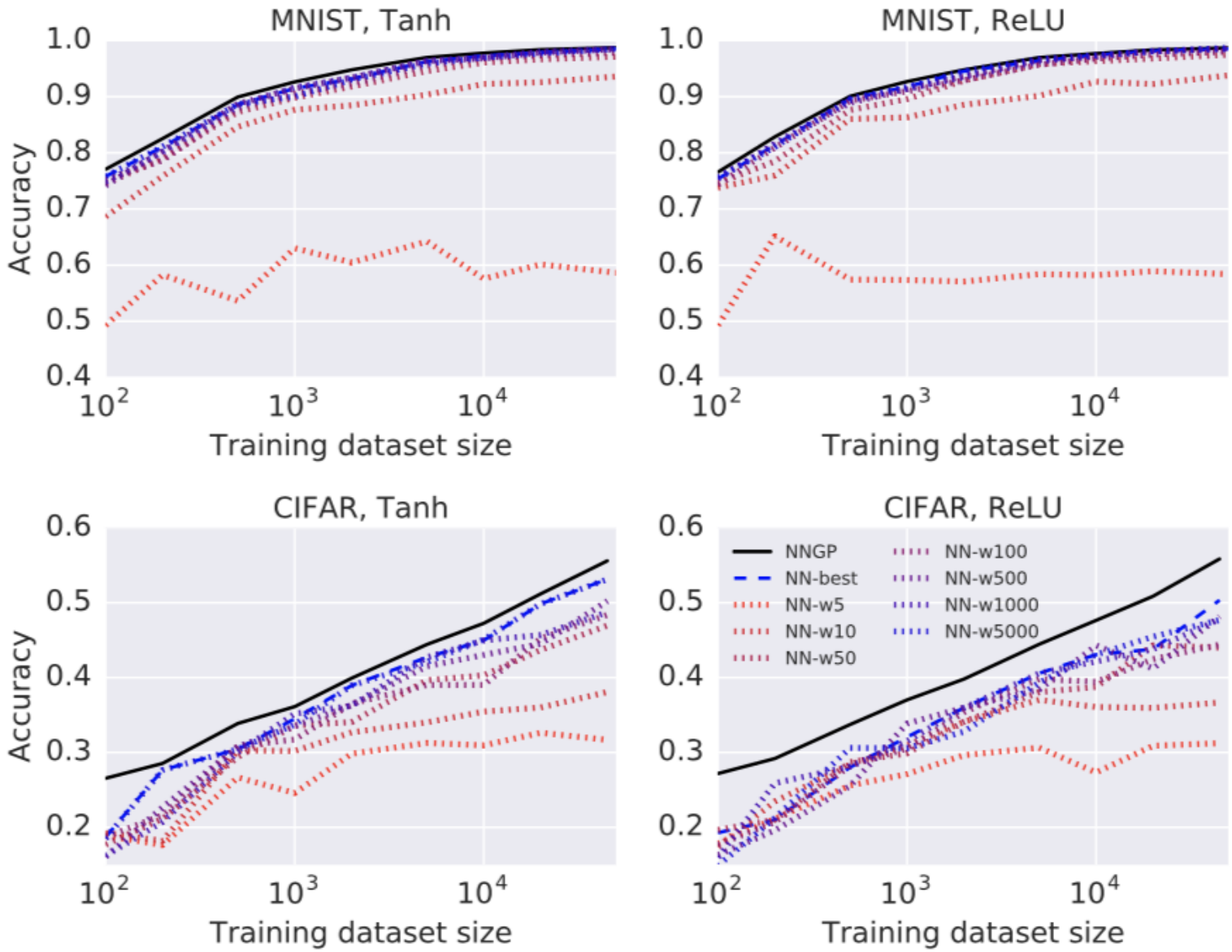
(4) numerically computiation

performace:

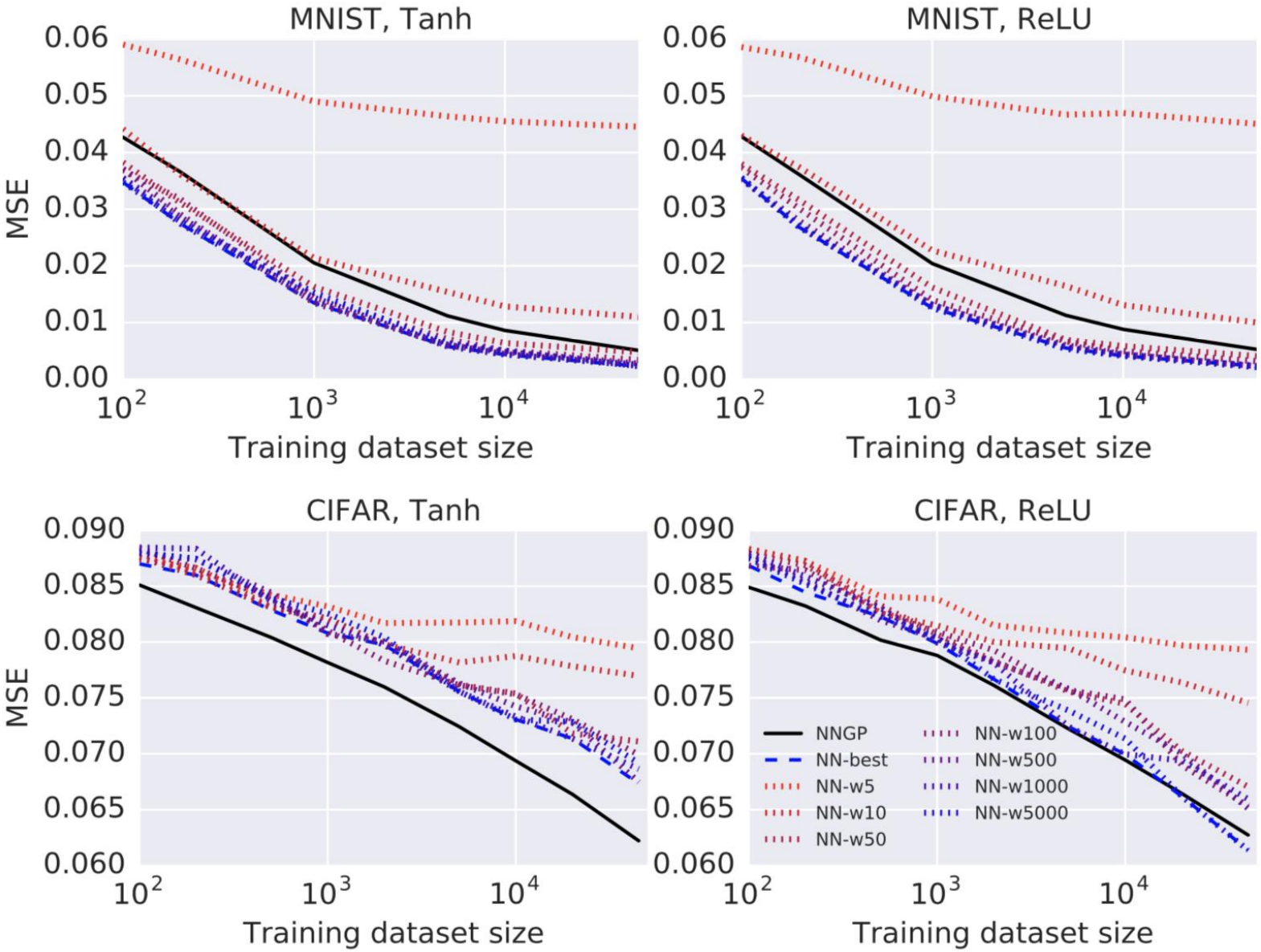
the NNGP often outperforms trained finite width networks

Table 1: The NNGP often outperforms finite width networks. Test accuracy on MNIST and CIFAR-10 datasets. The reported NNGP results correspond to the best performing depth, σ_w^2, and σ_b^2 values on the validation set. The traditional NN results correspond to the best performing depth, width and optimization hyperparameters. Best models for a given training set size are specified by (depth-width-σ_w^2-σ_b^2) for NNs and (depth–σ_w^2-σ_b^2) for GPs. More results are in Appendix Table 2.

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


MNIST, Tanh
MNIST, ReLU
CIFAR, Tanh
CIFAR, ReLU
Accuracy
Training dataset size
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
10^2
10^3
10^4
NNGP
NN-best
NN-w5
NN-w10
NN-w50
NN-w100
NN-w500
NN-w1000
NN-w5000

(a) Accuracy



(b) Mean squared error

Uncertainty

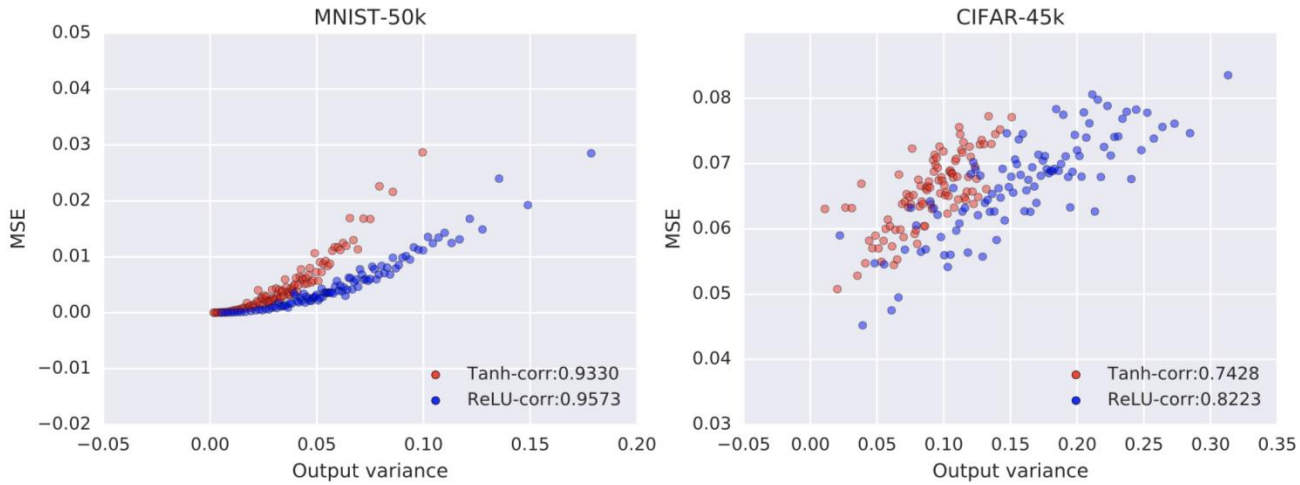


Figure 3: The Bayesian nature of NNGP allows it to assign a prediction uncertainty to each test point. This prediction uncertainty is highly correlated with the empirical error on test points. The $x-$axis shows the predicted MSE for test points, while the $y-$axis shows the realized MSE. To allow comparison of *mean* squared error, each plotted point is an average over 100 test points, binned by predicted MSE. The hyperparameters for the NNGP are depth$=3$, $\sigma_w^2 = 2.0$, and $\sigma_b^2 = 0.2$. See Appendix Figure 8 for dependence on training set size.

Generalization gap

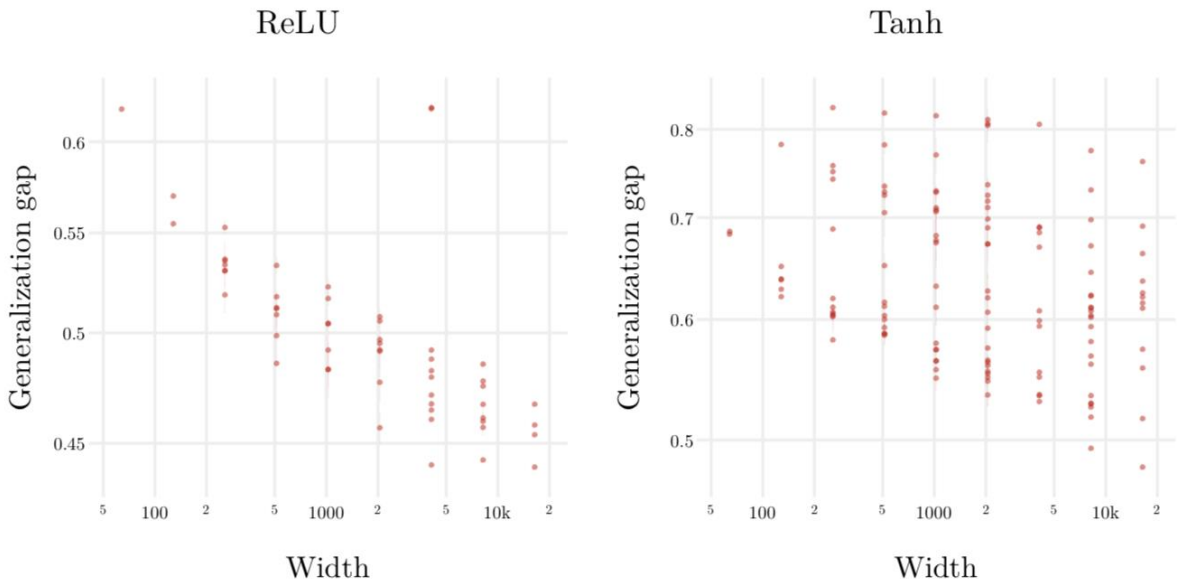


Figure 2: Generalization gap for five hidden layer fully-connected networks with variable widths, using ReLU and Tanh nonlinearities on CIFAR-10. Random optimization and initialization hyperparameters were used and results were filtered for networks with 100% classification training accuracy, resulting in a total of 125 Tanh and 55 ReLU networks. The best generalizing networks are consistently the widest.