Infinitely Wide Neural Networks for Small Data Tasks

Sai Praneeth Donthu

Mentored by Prof. Shankar Prawesh

Abstract

Theoretical research has shown that an infinitely wide neural network trained under L2 loss by gradient descent with an infinitesimally small learning rate is equivalent to kernel regression with respect to a so called Neural Tangent Kernel (NTK). Since Kernel Methods are highly efficient on small data tasks, we attempt to test the Neural Tangent Kernel on the UCI dataset collection, since they have a large collection of small-data data sets and compare their performance to other tried and tested models such as Random Forests, Support Vector Machines (SVM), and (Deep) Neural Networks.

1 Introduction

In the realm of artificial intelligence and machine learning, neural networks have emerged as powerful models for tackling complex problems across various domains. Traditionally neural networks have excelled when large amounts of labelled data are readily available for training. However, when large quantities of data are not available, neural networks tend to struggle a lot.

Modern neural networks having way more parameters than training data points tend to achieve near zero training error. This encouraged study of over parameterized networks, including those neural networks whose widths tend to infinity.

A recent line of study has shown that in the limit of infinite width, training of neural networks with l2 loss and infinitesimally small learning rate converges to kernel regression with a special kernel called the Neural Tangent Kernel (NTK).

In this project we explore this line of study and compare the performance of the Neural Tangent Kernel (equivalently the infinite width neural network) with the traditional Neural Networks and other popular classifiers like Random Forests, Decision Trees and Support Vector Machines and also check how the NTK fares against data sets with small data points.

We use the UCI Dataset collection with around 100 datasets which contains a wide variety of datasets to train and test our models.

2 The Neural Tangent Kernel

To start off, consider a fully connected L layer network with parameters θ and let the network function be $f(.;\theta) : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ where each layer n_i has i neurons. Totally there are P parameters where $P = \sum_{l=0}^{L-1}(n_l + 1)n_{l+1}$ and hence we get $\theta \in \mathbb{R}^P$.

Let the training dataset contain N data points $\{x^{(i)}, y^{(i)}\}_{i=i}^{N}$ where $x^{(i)}$ are the inputs and $y^{(i)}$ is its label.

Now looking into the network function in detail, for each layer l define a transformation $A^{(l)}$ with a weight matrix $\mathbf{w}^{(l)} \in \mathbb{R}^{n_l \times n_{l+1}}$ and a biasing term $\mathbf{b}^{(l)} \in \mathbb{R}^{n_{l+1}}$. Also let $\sigma(.)$ be a lipschitz continuous function. Now we can define $A^{(l)}$ as:-

$$\begin{aligned}
A^{(0)} &= \mathbf{x} \\
\tilde{A}^{(l+1)}(\mathbf{x}) &= \frac{1}{\sqrt{n_l}}\mathbf{w}^{(l)\top} A^{(l)} + \beta \mathbf{b}^{(l)} \quad \in \mathbb{R}^{n_{l+1}} && \text{; pre-activations} \\
A^{(l+1)}(\mathbf{x}) &= \sigma(\tilde{A}^{(l+1)}(\mathbf{x})) \quad \in \mathbb{R}^{n_{l+1}} && \text{; post-activations}
\end{aligned}$$

Here the $1/\sqrt{n_l}$ rescaling makes sure that the transformation doesnt diverge from the infinite width network. The constant $\beta \geq 0$ controls the biasing term. Coming to the NTK, firstly the loss function $\mathcal{L} : \mathbb{R}^P \to \mathbb{R}_+$ is defined as

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \ell(f(\mathbf{x}^{(i)};\theta), y^{(i)})$$

Here l is the per-sample loss function $\ell : \mathbb{R}^{n_0} \times \mathbb{R}^{n_L} \to \mathbb{R}_+$. According to the chain rule the gradient of the loss would be

$$\nabla_\theta \mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta f(\mathbf{x}^{(i)};\theta) \nabla_f \ell(f, y^{(i)})$$

While performing back propagation and updation of θ each update is of infinitesimal size hence it can be approximately viewed as a time derivative $\frac{d\theta}{dt} = -\nabla_\theta \mathcal{L}(\theta)$. Further the network function $f(.;x)$ gets updates accordingly by the chain rule:

$$\frac{df(\mathbf{x};\theta)}{dt} = \frac{df(\mathbf{x};\theta)}{d\theta}\frac{d\theta}{dt} = -\frac{1}{N}\sum_{i=1}^{N} \nabla_\theta f(\mathbf{x};\theta)^\top \nabla_\theta f(\mathbf{x}^{(i)};\theta) \nabla_f \ell(f, y^{(i)})$$

Here in the above formula, the NTK is equal to $\nabla_\theta f(\mathbf{x};\theta)^\top \nabla_\theta f(\mathbf{x}^{(i)};\theta)$. Formally the NTK is defined as $K : \mathbb{R}^{n_0} \times \mathbb{R}^{n_0} \to \mathbb{R}^{n_L \times n_L}$

$$K(\mathbf{x}, \mathbf{x}';\theta) = \nabla_\theta f(\mathbf{x};\theta)^\top \nabla_\theta f(\mathbf{x}';\theta)$$

Now that the NTK has been defined, we can see what happens to infinite width networks. When $n_1, \ldots, n_L \to \infty$ (Infinite width neural network), the NTK converges to be deterministic at initialization, meaning that the kernel is irrelevant to the initialization values and only determined by the model architecture and also stays constant during training. The proof of the above is lengthy and uses induction.

Another aspect is that for ultra wide neural networks, θ at initialisation is close to the actual values of θ hence by taylor expanding the network we can see that the model becomes a linearized mapping (Lee & Xiao, et al. 2019). Hence we can also conclude that when $n_1, \ldots, n_L \to \infty$ the network function $f(\theta;x)$ converges to a fixed kernel i.e, the Neural tangent Kernel.

Thus the Neural Tangent Kernel is an good approximation for large models. And since kernel regression models work brilliantly for small data tasks we can further examine the performance of the NTK (and equivalently the theoretical infinite width neural network) on data sets with small sizes.

3 Materials and Methods

3.1 Dataset Collection

We used the UCI Machine Learning Repository to collect around 100 datasets. A brief overview of the datasets chosen is given in the table below to give an idea on the variety in the datasets.

Dataset	Size	Classes
abalone	4177	3
acute-inflammation	120	2
acute-nephritis	120	2
arrhythmia	b452	13
balance-scale	625	3
bank	45211	2
blood	748	2
breast-cancer-wd	569	2
breast-cancer-wp	198	2
breast-cancer-w	699	2
breast-cancer	286	2
breast-tissue	106	6
car	1728	4
cardiography-10	2126	10
cardiography-3	2126	3
chess-krvk	28056	18
chess-krvkp	3196	2
congressional-voting	435	2
conn-bench	208	2
connect-4	67557	2
contrac	1473	3
credit-approval	690	2
cylinder-bands	512	2
dermatology	366	6
echocardiogram	131	2
ecoli	336	8
energy-y1	768	3
energy-y2	768	3
fertility	100	2
flags	194	8
glass	214	6
haberman-survival	306	2
heart-cleveland	303	5
heart-hungarian	294	2
heart-switzerland	123	2
heart-va	200	5
hepatitis	155	2
ilpd-indian-liver	583	2
ionosphere	351	2
iris	150	3
led-display	1000	10
lenses	24	3
letter	20000	26
libras	360	15
low-res-spect	531	9
lung-cancer	32	3
lymphography	148	4
magic	19020	2
mammographic	961	2
miniboone	130064	2
molec-biol-promoter	106	2
molec-biol-promoter	106	2
molec-biol-splice	3190	3
mushroom	8124	2
musk-1	476	2
musk-2	6598	2
nursery	12960	5
oocytes-merluccius-nucleus-4d	1022	2
oocytes-merluccius-nucleus-2f	1022	3
oocytes-trisopterus-nucleus-2f	912	2
oocytes-trisopterus-states-5b	912	2
ozone	2536	2
page-blocks	5473	5
parkinsons	195	2
pima	768	2
pittsburg-bridges-MATERIAL	106	3
pittsburg-bridges-REL-L	103	3
pittsburg-bridges-SPAN	92	3
pittsburg-bridges-T-OR-D	102	2
pittsburg-bridges-TYPE	105	6
planning	182	2
plant-margin	1600	100
plant-shape	1600	100
plant-texture	1600	100
post-operative	90	3
primary-tumor	330	15
ringnorm	7400	2
seeds	210	3
semeion	1593	10
spambase	4601	2
statlog-australian-credit	690	2
statlog-german-credit	1000	2
statlog-heart	270	2
statlog-image	2310	7
statlog-vehicle	846	4
steel-plates	1941	7
synthetic-control	600	6
teaching	151	3
tic-tac-toe	958	2
titanic	2201	2
trains	10	2
twonorm	7400	2
vertebral-column-2	310	2
vertebral-column-3	310	3
wall-following	5456	4
waveform-noise	5000	3
waveform	5000	3
wine-quality-red	1599	6
wine-quality-white	4898	7
wine	179	3
yeast	1484	10
zoo	101	7

Table 1: Datasets Chosen

3.2 Models

We built 8 classifiers, the details of which are given below.

1. **The Neural Tangent Classifier:-**

   The NTK implementation used has been borrowed from the implementation used in the paper "Harnessing the power of infinitely wide deep nets on small-data tasks."

2. **Neural Network (He Init):-**

   The neural network model was built in python using the Tensorflow. The architecture of the model is given below

   - 10 hidden layers
   - 100 nodes per layer
   - 200 epochs of gradient descent using the Adam optimizer with a learning rate of 0.01
   - He initialization for each hidden layer
   - Elu activation function (Relu was also used)

3. **Decision Tree Classifier:-**

   The decision tree classifier was built in python using the scikit learn library. The minimum number of leafs at each node was set to be 2 leafs. Nothing else was changed.

4. **Random Forest:-**

   The Random Forest Classifier was built in python using the scikit learn library. The Random forest trains an ensemble of 500 decision trees and outputs the output of majority of the trees.

5. **Support Vector Machine:-**

   The support vector machines was built in python using the scikit learn library. In total 4 SVM classifiers were built:-

   - SVM with linear kernel
   - SVM with polynomial kernel of degree 2
   - SVM with polynomial kernel of degree 3
   - SVM with RBF kernel

4 Results

Table 2 below lists the performance of the 8 classifiers on the datasets we collected under various parameters.

Models	Average Accuracy	P90	P95
Neural Tangent Kernel	83.24% $\pm$ 13.53%	34.65%	21.78%
Neural Network	80.01% $\pm$ 17.64%	34.65%	21.78%
Decision Tree	75.34% $\pm$ 16.74%	22.77%	14.85%
Random Forest	77.44% $\pm$ 17.40%	24.75%	14.85%
SVM (Linear)	76.76% $\pm$ 18.02%	23.76%	17.82%
SVM (2nd deg Polynomial)	79.98% $\pm$ 16.21%	28.71%	19.8%
SVM (3rd deg Polynomial)	80.44% $\pm$ 16.10%	32.67%	23.76%
SVM (Gaussian)	58.93% $\pm$ 22.40%	10.89%	9.91%

Table 2: Comparision Of Different Classifiers on UCI Datasets. P90/P95 here refers to the percentage of datasets on which the model gave an accuracy of more than 90/95 than the maximum accuracy

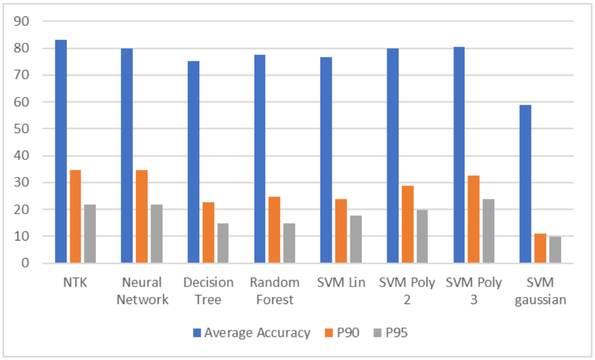

Figure 1: Results Of the models tested on the UCI Dataset

Based on the average accuracy over all the datasets, it is very clear that the Neural Tangent Kernel (Average Accuracy of 83.24%) very clearly outperforms all the other models with a clear margin of around 3%. The Neural Tangent Kernel is followed by the SVM with polynomial kernel (Average Accuracy of 80.44%), the neural network(Average Accuracy of 80.01%) and then the random forest (Average Accuracy of 77.44%).

The difference in average accuracies between NTK and Neural Network and NTK and Random Forests is quite significant (+3.24% and +5.8% in average accuracy respectively). This shows that the Neural Tangent Kernel performs much better than even robust models like neural networks and random forests on all type of data sets.

It is also important to note that amongst all 8 models, the Neural Tangent Kernel has a much smaller standard deviation (13.53%) as compared to the next smaller.

Coming over to the P90 and P95 statistics, firstly we note that both the Neural Tangent Kernel and the Neural Network have the same values, indicating the slight similarity of the two models. Comparing the Neural Tangent Kernel and the Neural Network with the other models we see that these two have a higher P90 value than the rest by a large margin (The SVM with polynomial kernel however outperforms these two in its P95 value).

4.1 Pairwise Comparision

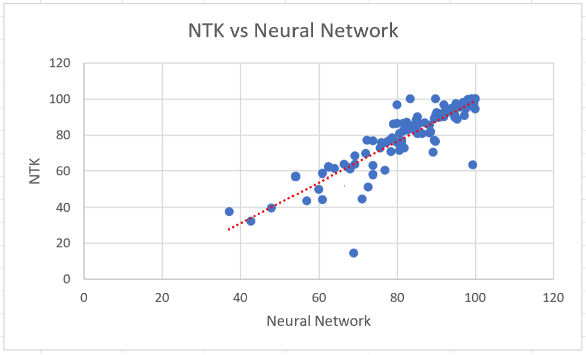

Figure 2: NTK vs Neural Network

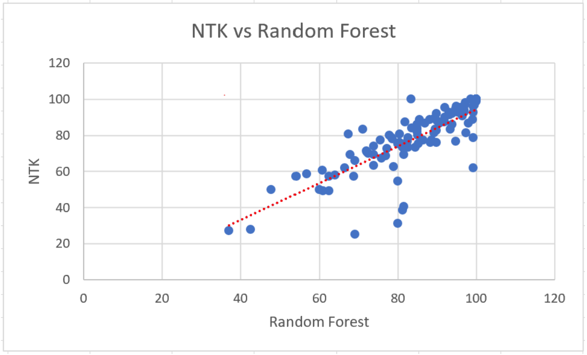

Figure 3: NTK vs Random Forests

NTK vs NN: We can also see how much better the Neural Tangent Kernel performs against the Neural Network, as it outperforms the Neural Network in 61 datasets while the Neural Network performs better in only 38 datasets. From Figure 2 we can also see that the NTK performs much more consistently and has fewer outliers than the Neural Network. The similarity of the two models can also be seen.

NTK vs RF: Similarly, the NTK performs even better against random forests than against the neural network, beating the random forest in 70 dataset whereas the random forest manages to beat the NTK in only 28 datasets. In the remaining 4 datasets, both achieved same 100% accuracy. From figure 3 it is also evident that the NTK is also consistent than random forest which again has a lot of outliers.

5 Conclusion

From the above results we can conclude that the Neural Tangent Kernel is a good approximation of neural networks at infinite width. It gives much better results than other popular classifiers like Neural Networks, Random Forests and Support Vector Machines. It is also found to give good results for small sized data sets, beating other excessively tuned conventional models. Also since NTKs are easy to compute they can be easily and quickly used.

6 References and Acknowledgments

1 Arora, S., Du, S. S., Li, Z., Salakhutdinov, R., Wang, R., & Yu, D. (2019). Harnessing the power of infinitely wide deep nets on small-data tasks. arXiv preprint arXiv:1910.01663.

2 Fernández-Delgado, M., Cernadas, E., Barro, S., & Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems?. The journal of machine learning research, 15(1), 3133-3181.

3 Olson, M., Wyner, A., & Berk, R. (2018). Modern neural networks generalize on small data sets. Advances in neural information processing systems (NIPS), 31.

4 Jacot et al. "Neural Tangent Kernel: Convergence and Generalization in Neural Networks." NeuriPS 2018.

5 Weng, Lilian. (Sep 2022). Some math behind neural tangent kernel. Lil'Log. https://lilianweng.github.io/posts/2022-09-08-ntk/.