MATHEMATICAL FOUNDATIONS OF ML

Team Members: Prishita Ray

Term: Spring 2022

TOPIC: Reviewing Performance of Deep Neural Networks when combined with Neural Tangent Kernels (NTK) and Backward Feature Correction

I. Literature Survey

A kernel-based method, called Neural Tangent Kernel (NTK) is able to capture the behaviour of fully-connected deep nets in the infinite width limit trained by gradient descent. Therefore, combining Convolutional Neural Network architectures (CNNs) with NTK, as performed in [1] can be used to explore these questions and gain an intuition about the correlation of the network complexity with the classification accuracy. NTKs can be used to analyse the effect of neural networks' architecture, in terms of extent of overparameterization, on empirical performance on the classification tasks. This has been proved in Paper 1 [1] where the accuracy (number of correctly classified samples vs total samples in validation/test sets) of infinite width kernel versions of vanilla CNNs (CNTKs) is shown to be within 8-10% lower than the corresponding finite width CNNs in multiclass classification tasks. Network complexity bounds, in terms of number of parameters (width) in the layers can range from anywhere between $[\frac{W*H*n}{K}, \infty)$, where W is the output width, h is the output height, n is the number of output channels and K is the subsampling factor of a particular layer in the network.

In deep neural networks, a learner learns to represent a complicated target function by decomposing it into a sequence of simpler functions to reduce sample and time complexity. Deep neural networks can optimize well on such tasks using SGD. The training of higher-level layers in the networks can actually improve the features of lower-level ones. This is the concept of backward feature correction as explained in Paper 2 [2]. This apparently explains why neural networks generalize better due to their backpropagation algorithm, which accumulates partial derivatives of the outputs from the final layer to the internal layers using chain rule for hierarchically training the layers.

Based on the number of backpropagations performed, the value of the loss l can range from $[0, \infty)$, where with learning rate 0.1 for a sample 3-layer network having outputs z_1, z_2 and z_3 for the layers, having target function output $f = 0.1(z_1^4 + 2z_2^4 + z_3^2)^2$ and calculating $\frac{\partial l}{\partial w}$, which is the cumulative partial derivative for parameters w and loss l = func(f,f'), f' being the actual function output, higher complexity terms in the target function will gradually get discovered by the higher levels of the learner network and "subtracted" from the training objective. As a result, the lower-level features can get improved by removing over-fitting to higher-complexity signals.

II. Experiments

a. Motivation

To check how overparameterization affects generalization in neural networks, as proposed in Paper 1, Vanilla CNNs with increasing depth but finite width have been implemented along with their corresponding infinite width Convolutional Neural Tangent Kernel counterparts (CNTKs). The CNNs and CNTKs were trained on subsets of the MNIST, Fashion MNIST, and CIFAR 10 datasets due to resource constraints (exceeding available memory) on the local machine as well as on Google Colaboratory GPUs and TPUs. From each dataset, 500 training samples and 50 validation samples were used. Also, the Global Average Pooling versions of the networks (CNN-GAP) and the kernels (CNTK-GAP) could not be implemented due to resource constraints on the machines (exceeding available memory). The CNNs were trained till full training accuracy was reached and the validation accuracy plateaued before decreasing, to avoid overfitting. This will help draw conclusions about whether more parameters results in better performance (accuracy) over the validation/testing sets.

Since neural networks use backward feature correction while training (hierarchical learning) as explained in paper 2, to check whether the algorithm has any effect on the convergence or generalization, a Wide Residual Network Architecture with varying number of features per layer (WRN-L-10) has been implemented, along with a version of the WRN-L-10 where parameters of the first group of the network are fixed and only the latter 2 groups of the network are trained using backward feature correction (BFC-4). Additionally, finite width NTKs of the WRN-L-10 networks have been trained to correlate their accuracy with the neural networks. Experiments have been performed with varying depths of the networks. The networks were again trained on subsets of the MNIST, Fashion MNIST, and CIFAR10 datasets due to resource constraints (exceeding available memory) on the local machine as well as on Google Colaboratory GPUs and TPUs. From each dataset, 400 training samples and 40 validation samples were used. The networks were trained till full training accuracy was reached and the validation accuracy plateaued before decreasing, to avoid overfitting. This will help explain whether the algorithm used by deep learning can be attributed for its superior performance. All accuracy results have been recorded over the validation/testing sets.

b. Codes

The link for the code implementation for Paper 1 can be accessed here:
https://colab.research.google.com/drive/1dwG_pEBjeS35gTUGkr6TQBnSK65haps3?usp=sharing

The link for the code implementation for Paper 2 can be accessed here:
https://colab.research.google.com/drive/1SRVAIAwFKBCq89zP4GGTColutB0HCQKO?usp=sharing

c. Hyperparameters and Trials

All experiments were performed using GPUs and TPUs in Google Colaboratory notebooks.

***Note:**

i. To run the code for paper 1 (MFM1.ipynb), please change the runtime device in the Colab notebook by going to the top menu, select Runtime->Change Runtime Type-> From None, to GPU. This should be followed for both the CNNs and the infinite width CNTKs.

ii. To run the code for paper 2 (MFM2.ipynb), please change the runtime device in the Colab notebook by going to the top menu, select Runtime->Change Runtime Type-> From None, to GPU for the networks and to TPU for the NTKs.

iii. Please check comments asking to modify input features based on dataset used, or parameters to vary depth and width in the experiments.

For paper 1, Vanilla CNNs and their infinite width CNTKs have been implemented with increasing depth (number of layers= 5,10,15,20). Different learning rates of 0.1,1,10 as in the original paper were also experimented with. SGD was used for convergence for the CNNs. The accuracy results reported are an average of 3 trials for each network or kernel.

For paper 2, two feature scaling factors (K=1,2) for each layer have been tried to vary the width of the networks and their kernels. Depth of the networks and kernels were increased by varying the number of blocks (N=1,2,3). Learning rate of 0.1 and momentum of 0.9 has been used for convergence as in the original paper. SGD was used for convergence for the networks and Adam with batch size 5 was used for the NTKs. The accuracy results reported are an average of 3 trials for each network or kernel.

No other tuning has been performed unless specifically mentioned.

III. Results and Discussion

a. Tables and Figures

<u>Paper 1 (Overparameterization):</u>

Classification Accuracy of the vanilla CNNs and their infinite-width CNTKs over the validation/testing sets of the 3 datasets have been recorded in Table 1 and 2 respectively. Comparison of the accuracy of the CNNs with learning rate of 0.1 and the infinite-width CNTKs can be observed in Figure 1.

Table 1: Classification Accuracy (%) of Vanilla CNNs based on depths of CNN (n) and learning rates (α)

Dataset	CNNs (α=0.1, Number of Layers n)				CNNs (α=1, Number of Layers n)				CNNs (α=10, Number of Layers n)			
	n=5	n=10	n=15	n=20	n=5	n=10	n=15	n=20	n=5	n=10	n=15	n=20
MNIST	98	100	96	96	10	10	10	18	10	10	10	12
Fashion MNIST	74	72	72	70	6	6	10	6	6	6	8	6
CIFAR10	34	40	40	30	10	10	6	10	10	10	16	10

Table 2: Classification Accuracy (%) of Infinite Width CNTK based on varying depths (n)

Dataset	Infinite Width CNTK (Number of Layers n)			
	n=5	n=10	n=15	n=20
MNIST	96	97.99	94	91.99
Fashion MNIST	74	74	70	70
CIFAR10	26	34	35.99	35.99

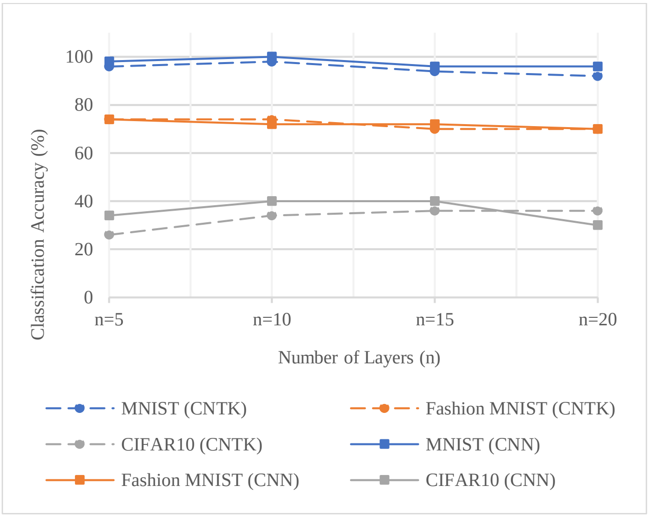

Figure 1: Accuracy Comparison of Infinite Width CNTK and Finite Width Vanilla CNNs based on Depth (Number of Layers n)

Paper 2 (Backward Feature Correction):

Classification Accuracy of the WRN-L-10 networks with complete training (Full), partial training (BFC-4) and their finite-width NTKs over the validation/testing sets of the 3 datasets have been recorded in Table 3. Comparison of the accuracy of the WRN-L-10 networks with complete training (Full) and partial training (BFC-4) for MNIST, Fashion MNIST and CIFAR 10 can be observed in figures 2,3 and 4 respectively.

Table 3: Classification Accuracy (%) of WRN-L-10 and its Finite NTK based on blocks N (depth), feature scaling K(width), and complete (Full) or hierarchical training (BFC-4), over validation sets of the datasets

Dataset	Number of Blocks N, Features Scaling Factor K																	
	N=1						N=2						N=3					
	K=1			K=2			K=1			K=2			K=1			K=2		
	Full	BFC-4	NTK	Full	BFC-4	NTK	Full	BFC-4	NTK	Full	BFC-4	NTK	Full	BFC-4	NTK	Full	BFC-4	NTK
MNIST	80	77.5	17.5	77.5	77.5	10	92.5	97.5	7.5	92.5	92.5	5	95	92.5	5	90	92.5	10
Fashion MNIST	75	82.5	12.5	70	80	7.5	70	77.5	15	72.5	72.5	15	80	82.5	2.5	72.5	75	5
CIFAR10	32.5	35	12.5	30	35	15	35	35	7.5	32.5	32.5	7.5	35	40	10	35	40	10

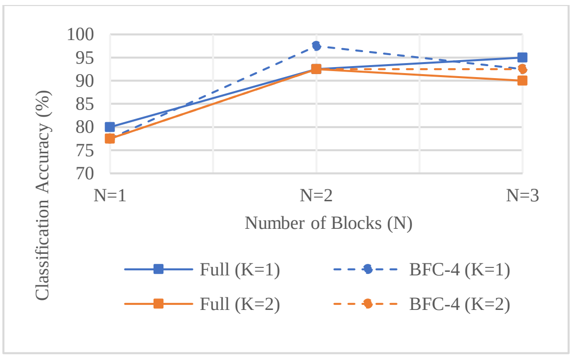

Figure 2: Accuracy Comparison on MNIST based on varying depth (N) and width (K) of WRN-L-10 architecture, with complete (Full) and partial (BFC-4) backward feature correction

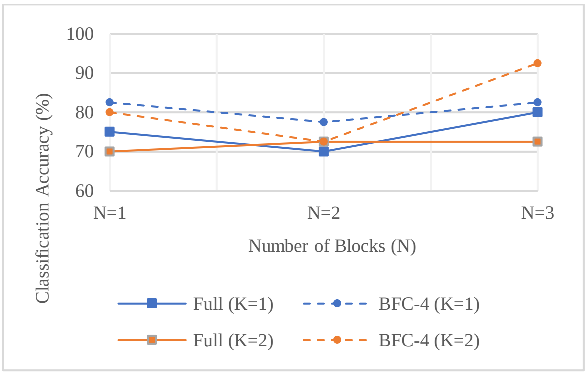

Figure 3: Accuracy Comparison on Fashion-MNIST based on varying depth (N) and width (K) of WRN-L-10 architecture, with complete (Full) and partial (BFC-4) backward feature correction

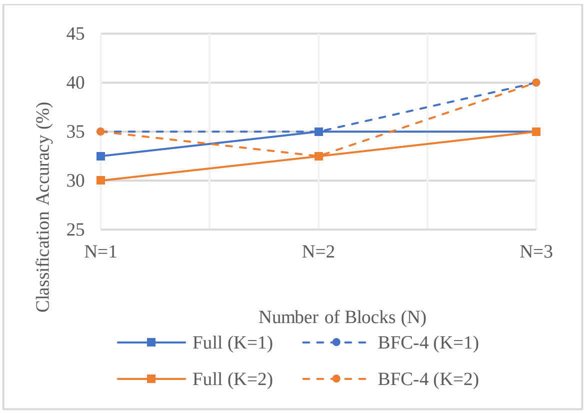

Figure 4: Accuracy Comparison on CIFAR10 based on varying depth (N) and width (K) of WRN-L-10 architecture, with complete (Full) and partial (BFC-4) backward feature correction

b. Discussion

Results for Paper 1:

From Tables 1 and 2, and Figure 1 it can be observed that increasing the number of parameters (network width) infinitely as in the infinite CNTKs does not necessarily result in higher accuracy than its finite width CNNs, even though it is largely within a -5% range of the CNNs accuracy.

There is also a drop in accuracy for both the CNNs and CNTKs as depth of the network (number of layers) is increased beyond a point (around 10-15 layers).

Additionally, increasing learning rate from 0.1 to 1 or even 10 affects network performance drastically giving bogus accuracy results.

Thus overparameterization, does not yield better performance and cannot explain the superiority of deep learning. These results are consistent with the conclusions of the original paper.

Results for Paper 2:

From Table 3 and Figures 2,3, and 4 it can be observed that the performance of the WRN-L-10 networks with full backward feature correction versus partial backward feature correction (keeping first group fixed) does not provide evidence about whether an extra backward feature correction between one pair of groups in the network necessarily results in higher accuracy. Even in the original paper, that trained the networks on the entire datasets, the differences in the performance of the Full and BFC-4 versions were within +/- 0.3% of each other, irrespective of increasing depth.

However, the accuracy of all the networks did increase with the depth (number of blocks), notwithstanding the amount of backward feature correction in all cases. This can be attributed to the fact that deeper layers with subsampling does capture larger patterns within the input training data that can improve its accuracy and generalization capability.

Adding more features in the layers, increases the performance till a point and then decreases due to the overparameterization problem found from the previous paper.

Additionally, the performance of NTKs compared with the networks drops drastically with increasing depth.

Therefore, the backward feature correction algorithm of deep learning may not be a strong basis for explaining its superiority as suggested in paper 2, though the depth of the network can give some insight to improvement in accuracy, which is consistent with paper.

IV. Code Overview

The code implementations provide necessary insights for the expected behaviour presented in the original papers. The experiments performed are similar to those in the papers with varying depth, width, hyperparameters and deep learning models and kernels as required. Overall code results seem consistent in behaviour, but have differences in values due to the amount of data used, in comparison to the papers. Convergence over training data is obtained in all the experiments.

V. Conclusion

From the results, neither overparameterization, nor the backward feature correction algorithm can properly explain the success of deep learning. The actual superiority in the performance can be attributed to the overall network architecture of neural networks, designed to obtain more abstract representations from the input data with each successive layer or block, via subsampling (pruning of quality inputs to each layer). However, the innermost abstraction should be just sufficient to capture the larger patterns and should not be too simple so as to hurt the basic discrimination ability. Therefore, the actual depth and width suitable for different applications requires experimentation and thus leads to development of new models.

References:

[1] Arora, Sanjeev, et al. "On exact computation with an infinitely wide neural net." Advances in Neural Information Processing Systems 32 (2019).

[2] Allen-Zhu, Zeyuan, and Yuanzhi Li. "Backward feature correction: How deep learning performs deep learning." arXiv preprint arXiv:2001.04413 (2020).