On Exact Computation with an Infinitely Wide Neural Net

Sanjeev Arora[1,2] Simon S. Du[2] Wei Hu[1] Zhiyuan Li[1] Ruslan Salakhutdinov[3] Ruosong Wang[3]

[1]Princeton University [2]Institute For Advanced Study [3]Carnegie Mellon University

What happens when width (# of channels) is large?

- Recent papers [Li and Liang, Du et al., Allen-Zhu et al., Zou et al.] proved that NNs with sufficiently large width can achieve 0 training error via gradient descent.
- Over-parametrization doesn't hurt generalization much. [Zhang et al.,17]
- [Jacot et al.] showed that as one increases the width to infinity, a certain limiting behavior, called neural tangent kernel (NTK), can emerge.

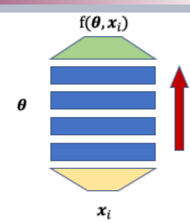
Main Questions:

1. **Can we formally show that the prediction of NNs is equivalent to that of NTKs when width is sufficiently large?**
2. **How does NTK of classic CNNs (VGG or AlexNet) perform on standard datasets, such CIFAR-10?**

Fully-connected (FC) networks and Neural Tangent Kernel

$$f(\boldsymbol{\theta}, \boldsymbol{x}) = \boldsymbol{W}^{(L+1)} \cdot \sqrt{\frac{c_\sigma}{d_L}} \sigma\left(\boldsymbol{W}^{(L)} \cdot \sqrt{\frac{c_\sigma}{d_{L-1}}} \sigma\left(\boldsymbol{W}^{(L-1)} \cdots \sqrt{\frac{c_\sigma}{d_1}} \sigma\left(\boldsymbol{W}^{(1)} \boldsymbol{x}\right)\right)\right)$$

where σ is activation, $c_\sigma = \left(\mathbb{E}_{z \sim \mathcal{N}(0,1)}\left[\sigma(z)^2\right]\right)^{-1} = 2$ for ReLU, $W_{ij}^{l} \sim N(0,1)$.



Square Loss: $\ell(\boldsymbol{\theta}) = \frac{1}{2} \sum_{i=1}^{n} \left(f(\boldsymbol{\theta}, \boldsymbol{x}_i) - y_i\right)^2$

Dynamics of Gradient Descent on ℓ: $\frac{\mathrm{d}\boldsymbol{u}(t)}{\mathrm{d}t} = -\boldsymbol{H}(t) \cdot (\boldsymbol{u}(t) - \boldsymbol{y})$

Here, $\boldsymbol{u}(t) = (f(\boldsymbol{\theta}(t), \boldsymbol{x}_i))_{i \in [n]} \in \mathbb{R}^n$ and $[\boldsymbol{H}(t)]_{i,j} = \left\langle \frac{\partial f(\boldsymbol{\theta}(t), \boldsymbol{x}_i)}{\partial \boldsymbol{\theta}}, \frac{\partial f(\boldsymbol{\theta}(t), \boldsymbol{x}_j)}{\partial \boldsymbol{\theta}} \right\rangle$

[Jacot et al.,18]: As $d_1, d_2, \ldots, d_L \to \infty$ sequentially, $\forall t, H(t) \to \Theta^{(L)}$.

Implication: GD Trajectory $\Longrightarrow$ ℓ_2 regression w.r.t. kernel $\Theta^{(L)}$.

L-layer recursion, encoding NN's architecture

$$\Sigma^{(0)}(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^\top \boldsymbol{x}',$$

$$\boldsymbol{\Lambda}^{(h)}(\boldsymbol{x}, \boldsymbol{x}') = \begin{pmatrix} \Sigma^{(h-1)}(\boldsymbol{x}, \boldsymbol{x}) & \Sigma^{(h-1)}(\boldsymbol{x}, \boldsymbol{x}') \\ \Sigma^{(h-1)}(\boldsymbol{x}', \boldsymbol{x}) & \Sigma^{(h-1)}(\boldsymbol{x}', \boldsymbol{x}') \end{pmatrix} \in \mathbb{R}^{2 \times 2},$$

$$\Sigma^{(h)}(\boldsymbol{x}, \boldsymbol{x}') = c_\sigma \mathop{\mathbb{E}}_{(u,v) \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{\Lambda}^{(h)}\right)} \left[\sigma(u)\sigma(v)\right],$$

Dependency on the derivative of non-linearity

$$\dot{\Sigma}^{(h)}(\boldsymbol{x}, \boldsymbol{x}') = c_\sigma \mathop{\mathbb{E}}_{(u,v) \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{\Lambda}^{(h)}\right)} \left[\dot{\sigma}(u)\dot{\sigma}(v)\right].$$

Final output:

$$\Theta^{(L)}(\boldsymbol{x}, \boldsymbol{x}') = \sum_{h=1}^{L+1} \left(\Sigma^{(h-1)}(\boldsymbol{x}, \boldsymbol{x}') \cdot \prod_{h'=h}^{L+1} \dot{\Sigma}^{(h')}(\boldsymbol{x}, \boldsymbol{x}')\right)$$

In what sense does an ultra wide NN converge to NTK?

Existing results on **asymptotic** convergence:

- [Jacot et al.'18] **sequential limit**($d_1 \to \infty, \ldots, d_L \to \infty$),
- [Yang'19] **simultaneous limit** ($d_1 = \cdots = d_L \to \infty$)

In practice, change during training $\approx O(m^{-1} \cdot \mathrm{poly}(n, L))$. [Lee et al.'19]

Theorem (this work): first **non-asymptotic** convergence result (m = width, n =# training data)

- At initialization: Finite-width NTK converges to Infinite-width NTK, i.e. $H(0) \to \Theta^{(L)}$, at the rate of $O(m^{-0.25} L^{1.5} \log n)$ for ReLU activation.
  (for smooth activation, the rate could be in principle improved to $O(m^{-0.5} L^2 \log n)$)
- During training: The change of NTK over training, i.e. $\left\|H(t) - H(0)\right\|_F$ is bounded by $O(m^{-1/6} \cdot \mathrm{poly}(n, L))$. (Using Lemma from [Allen-Zhu, Li, Song])

Convolutional Neural Tangent Kernel (CNTK)

CNN with L Conv layers and one FC layer:

Weights $W_{(\alpha),(\beta)}^{(h)} \in \mathbb{R}^{q \times q}$, $W_{(\alpha)}^{(L+1)} \in \mathbb{R}^{P \times Q}$ are initialized by i.i.d. $\mathcal{N}(0,1)$.

Let $\boldsymbol{x}^{(0)} = \boldsymbol{x} \in \mathbb{R}^{P \times Q \times C^{(0)}}$ be the input image where $C^{(0)}$ is the number of channels.
For $h = 1, \ldots, L$, $\beta = 1, \ldots, C^{(h)}$, the intermediate outputs are defined as

$$\tilde{\boldsymbol{x}}_{(\beta)}^{(h)} = \sum_{\alpha=1}^{C^{(h-1)}} \boldsymbol{W}_{(\alpha),(\beta)}^{(h)} * \boldsymbol{x}_{(\alpha)}^{(h-1)}, \quad \boldsymbol{x}_{(\beta)}^{(h)} = \sqrt{\frac{c_\sigma}{C^{(h)} \times q \times q}} \sigma\left(\tilde{\boldsymbol{x}}_{(\beta)}^{(h)}\right),$$

The final output is defined as $f(\boldsymbol{\theta}, \boldsymbol{x}) = \sum_{\alpha=1}^{C^{(L)}} \left\langle \boldsymbol{W}_{(\alpha)}^{(L+1)}, \boldsymbol{x}_{(\alpha)}^{(L)} \right\rangle$ **or** $\sum_{\alpha=1}^{C^{(L)}} W_{(\alpha)}^{(L+1)} \left(\frac{1}{PQ} \sum_{(i,j) \in [P] \times [Q]} \left[\boldsymbol{x}_{(\alpha)}^{(L)}\right]_{ij}\right)$

Without Global Average Pooling — With Global Average Pooling

CNTK formula: (1) Covariance $\Sigma^{(L)}$ in NN-GP (Gaussian Process)

For $\alpha = 1, \ldots, C^{(0)}$, $(i, j, i', j') \in [P] \times [Q] \times [P] \times [Q]$, define

$$\boldsymbol{K}_{(\alpha)}^{(0)}(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}_{(\alpha)} \otimes \boldsymbol{x}'_{(\alpha)} \text{ and } \left[\boldsymbol{\Sigma}^{(0)}(\boldsymbol{x}, \boldsymbol{x}')\right]_{ij,i'j'} = \sum_{\alpha=1}^{C^{(0)}} \mathrm{tr}\left(\left[\boldsymbol{K}_{(\alpha)}^{(0)}(\boldsymbol{x}, \boldsymbol{x}')\right]_{\mathcal{D}_{ij,i'j'}}\right).$$

For $h \in [L]$,
- For $(i, j, i', j') \in [P] \times [Q] \times [P] \times [Q]$, define

$$\boldsymbol{\Lambda}_{ij,i'j'}^{(h)}(\boldsymbol{x}, \boldsymbol{x}') = \begin{pmatrix} \left[\boldsymbol{\Sigma}^{(h-1)}(\boldsymbol{x}, \boldsymbol{x})\right]_{ij,ij} & \left[\boldsymbol{\Sigma}^{(h-1)}(\boldsymbol{x}, \boldsymbol{x}')\right]_{ij,i'j'} \\ \left[\boldsymbol{\Sigma}^{(h-1)}(\boldsymbol{x}', \boldsymbol{x})\right]_{i'j',ij} & \left[\boldsymbol{\Sigma}^{(h-1)}(\boldsymbol{x}', \boldsymbol{x}')\right]_{i'j',i'j'} \end{pmatrix} \in \mathbb{R}^{2 \times 2}.$$

- Define $\boldsymbol{K}^{(h)}(\boldsymbol{x}, \boldsymbol{x}'), \dot{\boldsymbol{K}}^{(h)}(\boldsymbol{x}, \boldsymbol{x}') \in \mathbb{R}^{P \times Q \times P \times Q}$: for $(i, j, i', j') \in [P] \times [Q] \times [P] \times [Q]$,

$$\left[\boldsymbol{K}^{(h)}(\boldsymbol{x}, \boldsymbol{x}')\right]_{ij,i'j'} = \frac{c_\sigma}{q^2} \cdot \mathop{\mathbb{E}}_{(u,v) \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{\Lambda}_{ij,i'j'}^{(h)}(\boldsymbol{x}, \boldsymbol{x}')\right)} \left[\sigma(u)\sigma(v)\right],$$

$$\left[\dot{\boldsymbol{K}}^{(h)}(\boldsymbol{x}, \boldsymbol{x}')\right]_{ij,i'j'} = \frac{c_\sigma}{q^2} \cdot \mathop{\mathbb{E}}_{(u,v) \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{\Lambda}_{ij,i'j'}^{(h)}(\boldsymbol{x}, \boldsymbol{x}')\right)} \left[\dot{\sigma}(u)\dot{\sigma}(v)\right].$$

- Define $\boldsymbol{\Sigma}^{(h)}(\boldsymbol{x}, \boldsymbol{x}') \in \mathbb{R}^{P \times Q \times P \times Q}$: for $(i, j, i', j') \in [P] \times [Q] \times [P] \times [Q]$,

$$\left[\boldsymbol{\Sigma}^{(h)}(\boldsymbol{x}, \boldsymbol{x}')\right]_{ij,i'j'} = \mathrm{tr}\left(\left[\boldsymbol{K}^{(h)}(\boldsymbol{x}, \boldsymbol{x}')\right]_{D_{ij,i'j'}}\right).$$

(2) Tangent Kernel $\Theta^{(L)}$ by Dynamic Programming

$$\boldsymbol{\Theta}^{(0)}(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\Sigma}^{(0)}(\boldsymbol{x}, \boldsymbol{x}')$$

For h=1,2,…,L-1,

$$\left[\boldsymbol{\Theta}^{(h)}(\boldsymbol{x}, \boldsymbol{x}')\right]_{ij,i'j'} = \mathrm{tr}\left(\left[\dot{\boldsymbol{K}}^{(h)}(\boldsymbol{x}, \boldsymbol{x}') \odot \boldsymbol{\Theta}^{(h-1)}(\boldsymbol{x}, \boldsymbol{x}') + \boldsymbol{K}^{(h)}(\boldsymbol{x}, \boldsymbol{x}')\right]_{D_{ij,i'j'}}\right)$$

$$\boldsymbol{\Theta}^{(L)}(\boldsymbol{x}, \boldsymbol{x}') = \dot{\boldsymbol{K}}^{(L)}(\boldsymbol{x}, \boldsymbol{x}') \odot \boldsymbol{\Theta}^{(L-1)}(\boldsymbol{x}, \boldsymbol{x}') + \boldsymbol{K}^{(L)}(\boldsymbol{x}, \boldsymbol{x}')$$

Final output (no GAP): $\mathrm{tr}\left(\boldsymbol{\Theta}^{(L)}(\boldsymbol{x}, \boldsymbol{x}')\right)$

(with GAP) $\sum_{(i,j,i',j') \in [P] \times [Q] \times [P] \times [Q]} \left[\boldsymbol{\Theta}^{(L)}(\boldsymbol{x}, \boldsymbol{x}')\right]_{ij,i'j'}$

Depth	CNN-V	CNTK-V	CNTK-V-2K	CNN-GAP	CNTK-GAP	CNTK-GAP-2K
3	59.97%	64.47%	40.94%	63.81%	70.47%	49.71%
4	60.20%	65.52%	42.54%	80.93%	75.93%	51.06%
6	64.11%	66.03%	43.43%	83.75%	76.73%	51.73%
11	69.48%	65.90%	43.42%	82.92%	**77.43%**	51.92%
21	75.57%	64.09%	42.53%	83.30%	77.08%	52.22%

Table 1: Classification accuracies of CNNs and CNTKs on the CIFAR-10 dataset. CNN-V represents vanilla CNN and CNTK-V represents the kernel corresponding to CNN-V. CNN-GAP represents CNN with GAP and CNTK-GAP represents the kernel correspondong to CNN-GAP. CNTK-V-2K and CNTK-GAP-2K represent training CNTKs with only 2,000 training data.

Take-aways:

(1) CNTK are very powerful kernels

(2) GAP significantly improves the test accuracy for both CNNs and CNTKs by 8%-10% in accuracy.

(3) There's still a 5%-6% performance gap between CNTKs and CNNs.

- Can we explain the effect of Global Average Pooling?

  Enhanced Convolutional Neural Tangent Kernels **(GAP ≈ Data Augmentation!!)**

  Zhiyuan Li, Ruosong Wang, Dingli Yu, Simon S. Du, Wei Hu, Ruslan Salakhutdinov, Sanjeev Arora

- How well does NTK perform on non-image tasks, compared to other standard ML methods?

  Harnessing the Power of Infinitely Wide Deep Nets on Small-data Tasks **(NTK beats random forests, NN and GP!!)**

  Sanjeev Arora, Simon S. Du, Zhiyuan Li, Ruslan Salakhutdinov, Ruosong Wang, Dingli Yu>