

On Exact Computation with an Infinitely Wide Neural Net by Arora et al. (2019)

Review by Adonis Jamal

CentraleSupélec

Context and Motivation

The Question: How does a Deep Neural Network behave when its width (number of channels/neurons) goes to infinity?

- **Infinite Width Limit:** Connects Deep Learning to Gaussian Processes (GPs), allowing for exact analytical study.
- **Neural Tangent Kernel (NTK):** Describes the training dynamics of infinite networks under gradient descent [1].
- **The Gap:** Previous works handled Fully Connected layers (FC). This paper provides the first exact, efficient algorithm for **Convolutional Neural Networks (CNTK)**.

Theoretical Guarantee (Proof Sketch)

Theorem: A fully-trained infinite-width net is equivalent to Kernel Regression using the CNTK.

Proof Idea (Lazy Training Regime): The proof relies on analyzing the trajectory of weights $W(t)$ during gradient descent.

- ① **Dynamics:** The network output $f(x)$ evolves according to:

$$\frac{df(x)}{dt} = -\eta H(t) \cdot (f(x) - y)$$

where $H(t)$ is the Neural Tangent Kernel.

- ② **Stability:** As width $m \rightarrow \infty$, the authors prove that weights stay infinitesimally close to initialization:

$$\|W(t) - W(0)\|_F \rightarrow 0$$

- ③ **Linearization:** Consequently, the kernel remains constant $H(t) \approx H(0)$. The complex non-linear training dynamics simplify to **linear regression** on the kernel features fixed at initialization.

The CNTK Algorithm

Computing the kernel exactly requires propagating covariance matrices through the network layers.

Recursive Convolution Step

For patches of images x and x' , the covariance $\Sigma^{(h)}$ at layer h is computed from layer $h-1$:

$$\Sigma^{(h)}(x, x') = c_\sigma \cdot \mathbb{E}_{(u,v) \sim \mathcal{N}(0, K^{(h-1)})} [\sigma(u)\sigma(v)]$$

Efficient ReLU Implementation

Instead of naïve sampling (slow), the paper uses a closed-form solution for the ReLU activation.

Let λ be the correlation between two inputs:

$$\lambda = \frac{\Sigma_{12}}{\sqrt{\Sigma_{11}\Sigma_{22}}}$$

The covariance after ReLU is exact:

$$V_{\text{ReLU}} = \frac{\sqrt{\Sigma_{11}\Sigma_{22}}}{2\pi} (\lambda(\pi - \arccos \lambda) + \sqrt{1 - \lambda^2})$$

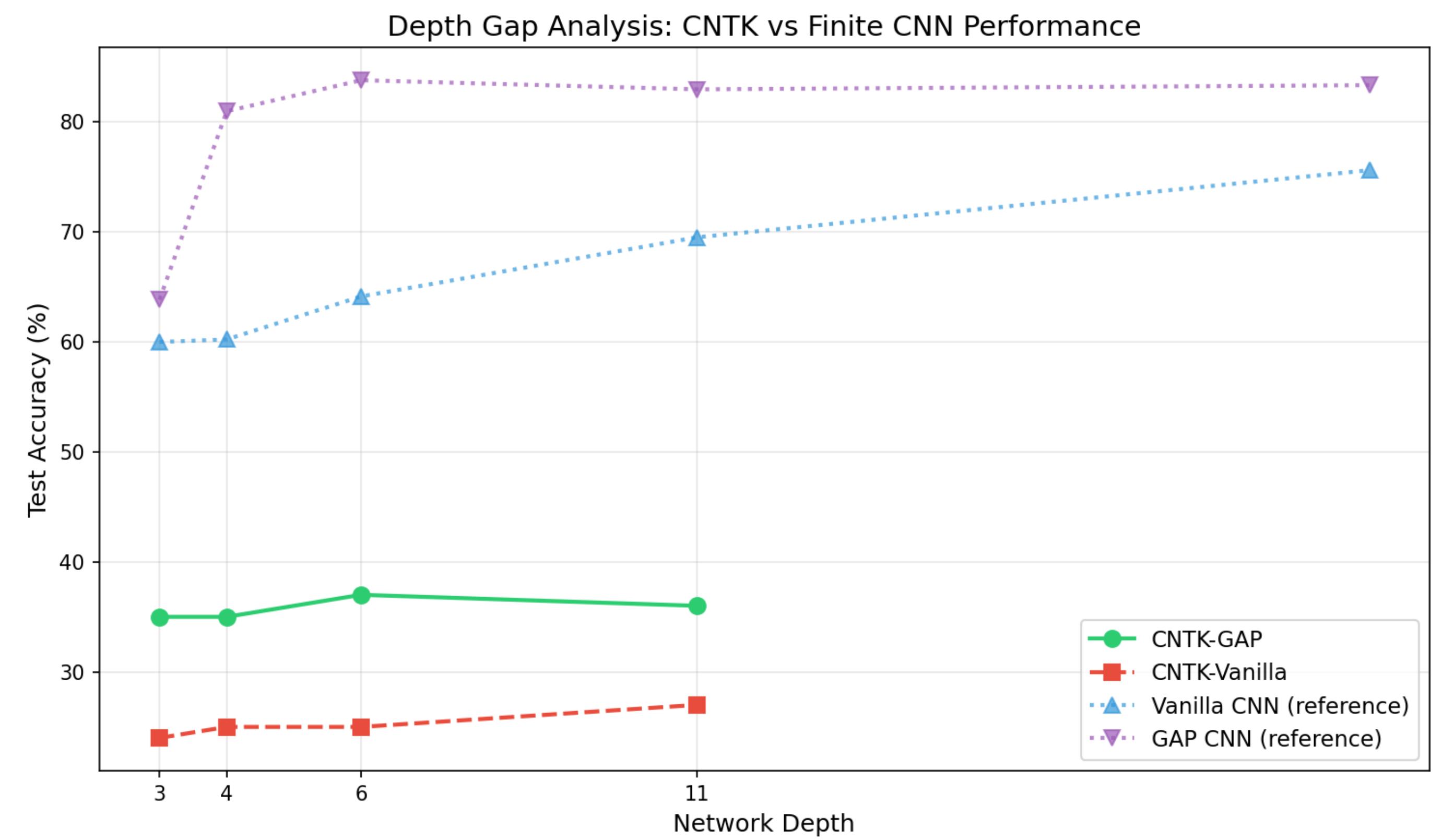
This allows for exact, vectorized computation on GPUs without Monte Carlo approximation.

Reproduction Setup

- **Code:** Implemented in PyTorch (Custom CNTK Module).
- **Data:** CIFAR-10 subset ($N = 200$ images) due to $O(N^2)$ complexity.

Experimental Results

The Depth Gap Phenomenon: While Finite CNNs improve with depth, the CNTK performance saturates.



- **Computational Cost:** The method scales as $O(N^2)$, making it intractable for full ImageNet without approximations.
- **Feature Learning:** The success of the "lazy" CNTK regime implies that feature learning is not strictly necessary for small data.
- **Limitations:** The performance gap on large data (Finite Nets > CNTK) suggests that feature learning (weights moving far from initialization) remains crucial for state-of-the-art performance.

References

- [1] Arthur Jacot, Franck Gabriel, and Clément Hongler.
Neural tangent kernel: Convergence and generalization in neural networks, 2020.