Theoretical Principles of Deep Learning
CentraleSupélec MDS 2025/2026 - Paper Reading Assignment
On Exact Computation with an Infinitely Wide Neural Net

Adonis JAMAL adonis.jamal@student-cs.fr

February 1st, 2026

1 Introduction and Contributions

The theoretical study of deep learning often relies on the limit of infinite width to provide tractable analytical tools for understanding optimization and generalization [1]. While the correspondence between infinite fully-connected networks and Gaussian Processes is well-established, the Neural Tangent Kernel (NTK) introduced by Jacot et al. [2] characterized the evolution of fully-trained networks in this regime. The paper *On Exact Computation with an Infinitely Wide Neural Net* extends this framework to Convolutional Neural Networks (CNNs), addressing the computational infeasibility of evaluating kernels for modern architectures with pooling layers. The authors pose a central question: can the "infinite limit" of classic architectures like VGG or AlexNet achieve competitive classification performance on standard datasets such as CIFAR-10? To answer this, they derive the Convolutional NTK (CNTK) and provide the first efficient, exact algorithm for its computation, proving that a fully-trained, sufficiently wide network is equivalent to the kernel regression predictor using CNTK.

In this report, we focus on the paper's algorithmic contribution: the exact and efficient computation of the CNTK. We first detail the theoretical recursive formulas that allow for the exact evaluation of convolutional kernels with and without Global Average Pooling (GAP). We then describe our reproduction of the paper's main experimental result on a subset of the data, implementing the exact dynamic programming algorithm to evaluate CNTK performance on the CIFAR-10 dataset. Our experiments confirm the significant performance gap between vanilla CNTK and CNTK-GAP architectures. Finally, we experimentally illustrate the authors' findings regarding network depth, verifying the distinct performance characteristics of infinite-width architectures compared to their finite counterparts across varying depths.

2 Paper Overview

2.1 The Setting: From Finite Networks to Kernel Limits

The paper operates in the regime of extreme over-parametrization, specifically the limit where the width of the neural network (number of channels in convolutional layers or nodes in fully-connected layers) tends to infinity.

Historically, the connection between infinite-width networks and kernel methods was studied using Gaussian Processes (GPs). In this "weakly-trained" regime, only the last layer is optimized

while hidden layers remain fixed at their random initialization. However, this view fails to capture the dynamics of modern deep learning, where all layers are trained simultaneously.

The authors build upon the "fully-trained" regime introduced in [2], known as the Neural Tangent Kernel (NTK). In this setting, gradient descent on an Infinitely wide network is equivalent to kernel regression with a deterministic kernel Θ, defined as:

$$\Theta(x, x') = \mathbb{E}_{\theta \sim \mathcal{W}} \left\langle \frac{\partial f(\theta, x)}{\partial \theta}, \frac{\partial f(\theta, x')}{\partial \theta} \right\rangle \tag{1}$$

where $f(\theta, x)$ is the network output and $\mathcal{W}$ is the initialization distribution. THis setting provides a theoretical bridge to understand why over-parametrized networks optimize easily and generalize well.

2.2 The Question: Feasibility and Performance of Convolutional Kernels

Prior to this work, the exact computation of the NTK for Convolutional Neural Networks (CNTK) was considered computationally infeasible for large datasets like CIFAR-10. Researchers resorted to Monte Carlo approximations or finite-width proxies, which introduced variance and degraded performance [3].

The authors seek to answer two primary questions:

- **Algorithmic Feasibility:** Can we derive an efficient, exact algorithm to compute the CNTK for standard architectures (including pooling layers) without relying on approximations?
- **Performance Gap:** How well do these "infinite" CNNs actually perform compared to their finite counterparts such as AlexNet and VGG-19? Does the infinite width limit capture the power of deep learning?

2.3 The Solution: CNTK and Exact Algorithms

The paper provides a three-fold solution:

- **Algorithmic:** They derive an exact dynamic programming algorithm to compute the CNTK for convolutional networks, utilizing the specific properties of ReLU activations to enable efficient GPU implementation.
- **Theoretical:** They provide a non-asymptotic proof that fully-trained, sufficiently wide networks are equivalent to kernel regression with the CNTK.
- **Empirical:** They establish a new benchmark for kernel methods, achieving $\approx 77\%$ accuracy on CIFAR-10 with an 11-layer architecture, narrowing the gap between kernel methods and finite deep networks.

3 Methodology: Exact Recursive Computation

In this section, we detail the algorithmic contribution of the paper: the exact computation of the Convolutional Neural Tangent Kernel (CNTK).

The core contribution of the paper is an efficient algorithm to compute the CNTK kernel matrix $\Theta^{(L)}(x, x')$ for two inputs x and x'. For a network of depth L, the computation relies on a dynamic programming approach that recursively propagates covariance matrices through the layers.

3.1 Covariance Propagation

For a convolutional network, the kernel computation requires tracking two quantities layer-by-layer:

1. $\Sigma^{(h)}(x, x')$: The covariance of the pre-activations at layer h.
2. $\dot{\Sigma}^{(h)}(x, x')$: The derivative covariance, which captures the backward pass dynamics.

For ReLU activation $\sigma(z) = \max(0, z)$, the paper leverages closed-form arc-cosine kernels to compute these expectations efficiently without explicit integration. Given the covariance matrix Λ from the previous layer with correlation ρ, the key recursive formulas are:

$$\mathbb{E}[\sigma(u)\sigma(v)] = \frac{\sqrt{\Sigma_{11}\Sigma_{22}}}{2\pi}\left(\rho(\pi - \arccos\rho) + \sqrt{1-\rho^2}\right) \tag{2}$$

$$\mathbb{E}[\sigma'(u)\sigma'(v)] = \frac{1}{2\pi}(\pi - \arccos\rho) \tag{3}$$

These formulas allow the algorithm to compute the kernel for layer h directly from the covariance of layer $h-1$ via entry-wise matrix operations, making the process GPU-efficient.

3.2 CNTK-Vanilla vs. CNTK-GAP

The paper describes two variants of the kernel:

- **CNT-Vanilla (CNTK-V):** Corresponds to a standard convolutional network ending with a fully connected layer.
- **CNTK-GAP:** Includes a Global Average Pooling (GAP) operation before the final classification. This involves taking the mean over spatial dimensions, which significantly alters the final kernel $\Theta^{(L)}$ by accounting for cross-variances between all spatial patches.

4 Data and Experimental Setup

4.1 Dataset and Preprocessing

We utilized the CIFAR-10 dataset, a standard benchmark for image classification containing over $50,000$ training and $10,000$ test images across 10 classes. Consistent with the paper's protocol, we applied no data augmentation to ensure a fair comparison between kernel methods and finite networks.

The labels were processed using a specific encoding scheme to improve regression performance: the target label y_c is set to 0.9 for the correct class, and -0.1 for incorrect classes. This zero-mean centering helps in the context of Mean Squared Error (MSE) regression.

4.2 Experimental Reproduction Setup

While the original paper scales the exact CNTK computation to the full CIFAR-10 using optimized CUDA kernels on NVIDIA Testa V100 GPUs, reproducing the full-scale experiment is computationally prohibitive for this project. We therefore adapted the experimental scale while strictly preserving the algorithmic implementation.

- **Data Subset:** We restricted the dataset to a subset of $N_{train} = 200$ and $N_{test} = 100$ images. This allows us to verify the correctness of the CNTK implementation and observe depth-dependent behaviors without requiring extensive computational resources.
- **Kernel Regression:** Classification is performed via exact kernel regression. For a test point x_{test}, the prediction $f^*(x_{test})$ is computed as:

$$f^*(x_{test}) = \ker(x_{test}, X_{train})(H^*)^{-1}Y_{train} \tag{4}$$

  where H^* is the kernel matrix on the training data. The class with the highest predicted value is selected.
- **Implementation:** We implemented the recursive CNTK formulas using PyTorch with GPU acceleration to handle the tensor operations required for the covariance propagation.

5 Results

6 Conclusion and Discussion

References

[1] Sanjeev Arora, Simon S. Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net, 2019. URL https://arxiv.org/abs/1904.11955.

[2] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks, 2020. URL https://arxiv.org/abs/1806.07572.

[3] Roman Novak, Lechao Xiao, Yasaman Bahri, Jaehoon Lee, Greg Yang, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-dickstein. Bayesian deep convolutional networks with many channels are gaussian processes. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=B1g30j0qF7.

A Appendix: Appendix section