Understanding Neural Tangent Kernel: Key Theories and Experimental Insights

Samy Vilhes

Understanding Neural Tangent Kernel : Key Theories and Experimental Insights

Samy Vilhes

September 2024

Abstract

The Neural Tangent Kernel (NTK) has become a powerful framework for analyzing the behavior of deep neural networks in the infinite-width limit. This paper presents a concise overview of the key theoretical foundations of NTK, covering its origins, the proof of deterministic behavior at initialization, and its role in bounding the training loss for regression tasks. Additionally, we extend this analysis by establishing a bound for the training loss in classification problems. Each theoretical property of the NTK is validated through experiments on various datasets.

1 Introduction

Artificial Neural Networks (ANNs) are employed in a wide range of tasks, including market prediction, image classification, image generation, and anomaly detection. Understanding the training dynamics of ANNs is crucial for improving their performance and interpretability. The Neural Tangent Kernel (NTK) appeared in 2018 in [2]. The NTK provides a powerful framework for studying these dynamics. Notably, in the case of an infinitely wide ANN (when $H \to \infty$ in Fig. 1), the NTK becomes deterministic at initialization, remains constant during training, and the network's behavior converges to kernel regression, with the NTK serving as the kernel. The theoretical foundation of NTK relies on the understanding that an infinitely wide ANN behaves like a Gaussian process.

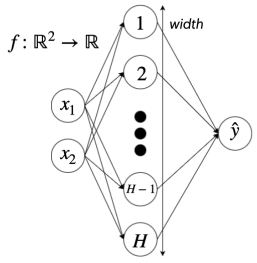

Figure 1: One hidden layer ANN with width H

2 Training an Artificial Neural Network

Designing the architecture of an ANN f is only one aspect of achieving optimal performance. The network must also be trained on a well-structured dataset, denoted as $(\mathcal{X}_{train}, \mathcal{Y}_{train})$, using an appropriate loss function $\mathcal{L}$ and optimized over several iterations t via gradient descent. This process iteratively updates the parameters θ_t of the ANN.
Let us define the following:

- (x, y_i): a data point consisting of input vector x and its corresponding label y_i (either a scalar or a class) from the training set $(\mathcal{X}_{train}, \mathcal{Y}_{train})$
- $f_{\theta_t}(x)$: the prediction made by the ANN f, parameterized by θ_t, for input x
- $l(f_{\theta_t}(x), y_i)$: the prediction error for sample i, where l is the individual loss function
- $\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} l(f_{\theta_t}(x), y_i)$: the overall loss function averaged over N training samples
- η: the learning rate, controlling the step size of each update
- $\nabla_f G$: the gradient of a function G with respect to f

Using gradient descent, the parameters update rule is expressed as:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}$$

This process allows the model to converge to a local minimum of the loss function.

Algorithm 1 Gradient Descent for Neural Network Training

1: **Input:** Dataset $(\mathcal{X}_{train}, \mathcal{Y}_{train})$, initial parameters θ_0, learning rate η, iterations T
2: **Output:** Parameters θ_T
3: **for** $t = 0$ to $T - 1$ **do**
4: Compute predictions $f_{\theta_t}(x)$
5: Calculate loss: $\mathcal{L} = \sum_{i=1}^{N} l(f_{\theta_t}(x), y_i)$
6: Update: $\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}$
7: **end for**
8: **Return** θ_T

3 Introduction to the Neural Tangent Kernel

3.1 Apparition of the Neural Tangent Kernel

We recall that the parameter update step is given by:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}$$

Rearranging the terms, we obtain:

$$\frac{\theta_{t+1} - \theta_t}{\eta} = -\nabla_\theta \mathcal{L}$$

In general, we use a small learning rate η to ensure stable convergence. Consequently, the update step can be viewed as gradient flow, which represents the time-continuous analog of gradient descent.

$$\frac{\theta_{t+1} - \theta_t}{\eta} \rightarrow \frac{\partial \theta_t}{\partial t}$$

Thus:

$$\frac{\partial \theta_t}{\partial t} = -\nabla_\theta \mathcal{L}$$

We are interested in the training dynamic of the ANN so the quantity $\frac{\partial f_{\theta_t}}{\partial t}$. Considering the chain-rule:

$$\frac{\partial f_{\theta_t}}{\partial t} = \frac{\partial f_{\theta_t}}{\partial \theta_t} \frac{\partial \theta_t}{\partial t}$$

Where $\frac{\partial f_{\theta_t}}{\partial \theta_t}$ represents a vector of derivatives. Assuming we have P parameters :

$$\frac{\partial f_{\theta_t}}{\partial \theta_t} = (\frac{\partial f_{\theta_t}}{\partial \theta_t^1}, \frac{\partial f_{\theta_t}}{\partial \theta_t^2}, \dots \frac{\partial f_{\theta_t}}{\partial \theta_t^P})$$

With the definition of $\mathcal{L}$ we have:

$$\nabla_\theta \mathcal{L} = \sum_{i=1}^{N} \frac{\partial f_{\theta_t}(x_i)}{\partial \theta_t} \frac{\partial l(f_{\theta_t}(x_i), y_i)}{\partial f}$$

Thus we have:

$$\frac{\partial f_{\theta_t}(x_j)}{\partial t} = -\sum_{i=1}^{N} \left(\frac{\partial f_{\theta_t}(x_j)}{\partial \theta_t} \right)^T \frac{\partial f_{\theta_t}(x_i)}{\partial \theta_t} \frac{\partial l(f_{\theta_t}(x_i), y_i)}{\partial f}$$

Where the quantity in red represents the value of the NTK between (x, x'). The NTK is a squared matrix $N \times N$, symmetric positive semidefinite (because the NTK can be seen as $\left(\nabla_{\theta_t} f_{\theta_t}(\mathcal{X}_{train})\right)^T \nabla_{\theta_t} f_{\theta_t}(\mathcal{X}_{train})$). Thus we write:

$$\Theta_{i,j}^t = \left(\frac{\partial f_{\theta_t}(x_i)}{\partial \theta_t} \right)^T \frac{\partial f_{\theta_t}(x_j)}{\partial \theta_t}$$

The NTK depends on the parameters at timestep t. Initializing the same architecture twice will yield two models with different parameters, resulting in distinct NTK values at initialization, denoted as Θ^0. Furthermore, as training progresses, the parameters are updated, causing the NTK to evolve over time, such that $\Theta^t \neq \Theta_{t'}$ for $t \neq t'$.

3.2 Properties of the Neural Tangent Kernel

In the case of an infinitely wide ANN, the NTK exhibits the following properties:

- It is deterministic at initialization.
- It remains constant during training.

In this discussion, we will focus on proving the NTK's determinism at initialization. To understand this proof, a solid grasp of the theory linking infinitely-wide neural networks to Gaussian processes is essential.

4 Neural Networks and Gaussian Processes

An equivalence between infinitely wide single-hidden-layer ANN and Gaussian processes was established by [4] in 1994. This foundational work demonstrated how the behavior of such neural networks can be understood in the context of Gaussian processes. Later, in 2017, [3] extended this result, showing that the same equivalence holds for each scalar output of an infinitely wide deep ANN. This extension highlights the broader implications of the relationship between neural networks and Gaussian processes, particularly in the context of deep learning architectures.

4.1 Introduction to Gaussian Processes

A Gaussian process (GP) is defined as a collection of random variables, any finite number of which have a joint Gaussian distribution. A GP is specified by a mean function $m(x)$ and a covariance function (or kernel) $k(x, x')$:

$$f(x) \sim \mathcal{GP}(m(x), k(x, x'))$$

where

$$m(x) = \mathbb{E}[f(x)]$$

and

$$k(x, x') = \mathrm{Cov}(f(x), f(x'))$$

The mean function $m(x)$ captures the average behavior of the process, while the covariance function $k(x, x')$ encodes assumptions about the function's smoothness and variability.

4.2 Infinitely wide single-hidden-layer ANN as GP

In this section, we will provide a mathematical demonstration of the connection between wide single-hidden-layer ANN. We will derive how, in the limit of infinite width, the distribution of the outputs of ANN can be described by Gaussian processes.

We consider a single-hidden-layer ANN f_{θ_0} at initialization of width H. For simplicity, we will denote this network as f. The function f maps inputs from $\mathbb{R}^I$ to outputs in $\mathbb{R}^K$:

$$f : \mathbb{R}^I \to \mathbb{R}^K$$

where:

- I represents the dimension of the input space
- K denotes the dimension of the output space.

To simplify the calculations, we will consider the ANN f without bias terms. However, it is important to note that the proof can be extended to include bias terms.
In fact :

$$f(x) = \sqrt{\frac{c_\sigma}{H}} W^1 \sigma(\sqrt{\frac{1}{I}} W^0 x)$$

where:

- $W^1 \in \mathbb{R}^{K \times H}$ with $W^1_{i,j} \sim \mathcal{N}(0,1)$
- $W^0 \in \mathbb{R}^{H \times I}$ with $W^0_{i,j} \sim \mathcal{N}(0,1)$
- σ the activation function
- $c_\sigma = (\mathbb{E}[\sigma(z)^2])^{-1}$ a scaling factor preserving norms

We can represent the k^{th} output by:

$$f_k(x) = \sqrt{\frac{c_\sigma}{H}} \sum_{j=1}^{H} W^1_{k,j} h_j(x)$$

with:

$$h_j(x) = \sigma\left(\sqrt{\frac{1}{I}} \sum_{i=1}^{I} W^0_{j,l} x_l\right)$$

Firstly we have:

$\mathbb{E}[W^1_{k,j} h_j(x)] = \mathbb{E}[W^1_{k,j}] \cdot \mathbb{E}[h_j(x)] = 0 \quad (\text{because } \mathbb{E}[W^1_{k,j}] = 0) \quad \forall\, (i,j) \in [I] \times [H]$

Secondly:

$$\begin{aligned}
\text{Var}(W^1_{k,j} h_j(x)) &= \mathbb{E}\left[\left(W^1_{k,j} h_j(x)\right)^2\right] - \mathbb{E}[W^1_{k,j}] \cdot \mathbb{E}[h_j(x)] \\
&= \mathbb{E}\left[\left(W^1_{k,j} h_j(x)\right)^2\right] \\
&= \mathbb{E}\left[\left(W^1_{k,j}\right)^2\right] \cdot \mathbb{E}\left[\left(h_j(x)\right)^2\right] \\
&= \mathbb{E}\left[\left(h_j(x)\right)^2\right]
\end{aligned}$$

Using the Central Limit Theorem we have:

$$\frac{\frac{1}{H}\sum_{j=1}^{H} W_{k,j}^1 h_j(x) - \mathbb{E}[W_{k,j}^1 h_j(x)]}{\frac{\sqrt{\mathrm{Var}(W_{k,j}^1 h_j(x))}}{\sqrt{H}}} \underset{H\to\infty}{\longrightarrow} \mathcal{N}(0,1)$$

$$\frac{\frac{1}{H}\sum_{j=1}^{H} W_{k,j}^1 h_j(x)}{\frac{\sqrt{\mathbb{E}\left[(h_j(x))^2\right]}}{\sqrt{H}}} \underset{H\to\infty}{\longrightarrow} \mathcal{N}(0,1)$$

$$\frac{1}{\sqrt{H}}\sum_{j=1}^{H} W_{k,j}^1 h_j(x) \underset{H\to\infty}{\longrightarrow} \mathcal{N}(0, \mathbb{E}\left[(h_j(x))^2\right])$$

$$\sqrt{\frac{c_\sigma}{H}}\sum_{j=1}^{H} W_{k,j}^1 h_j(x) \underset{H\to\infty}{\longrightarrow} \mathcal{N}(0, c_\sigma \mathbb{E}\left[(h_j(x))^2\right])$$

Thus we have in the infinite-width context:

$$f_k(x) \sim \mathcal{N}(0, c_\sigma \mathbb{E}\left[(h_j(x))^2\right])$$

Thus:

$$\{f_k(x)\}_{i=1}^t \sim \mathcal{GP}(0, \Sigma^1)$$

We have $m(x) = 0$ because $\mathbb{E}[f_k(x)] = 0$.
Furthermore:

$$\begin{aligned}
\Sigma^1(x,x') &= \mathrm{Cov}(f_k(x), f_k(x')) \\
&= \mathbb{E}[f_k(x) f_k(x')] - \mathbb{E}[f_k(x)]\mathbb{E}[f_k(x')] \\
&= \mathbb{E}[f_k(x) f_k(x')] \\
&= \mathbb{E}[\Big(\sqrt{\frac{c_\sigma}{H}}\sum_{j=1}^{H} W_{k,j}^1 h_j(x)\Big)\Big(\sqrt{\frac{c_\sigma}{H}}\sum_{m=1}^{H} W_{k,m}^1 h_m(x')\Big)] \\
&= \frac{c_\sigma}{H}\sum_{j=1}^{H}\sum_{m=1}^{H} \mathbb{E}[W_{k,j}^1 W_{k,m}^1]\mathbb{E}[h_j(x) h_m(x')] \\
&= \frac{c_\sigma}{H}\sum_{j=1}^{H} \mathbb{E}[\left(W_{k,j}^1\right)^2]\mathbb{E}[h_j(x) h_j(x')] \\
&= \frac{c_\sigma}{H}\sum_{j=1}^{H} \mathbb{E}[h_j(x) h_j(x')] \\
&\underset{H\to\infty}{\longrightarrow} c_\sigma \mathbb{E}[h_1(x) h_1(x')]
\end{aligned}$$

Because:

$$\mathbb{E}[W_{k,j}^1 W_{k,m}^1] = \begin{cases} 0 & \text{if } j \neq m, \\ \mathrm{Var}(W_{k,j}^1) = 1 & \text{if } j = m. \end{cases}$$

And:

$$\mathbb{E}[h_j(x)h_j(x')] = \mathbb{E}[h_m(x)h_m(x')] \quad \forall (j,m) \in [H] \times [H]$$

Thus we have:

$$\{f_k(x)\}_{i=1}^{t} \sim \mathcal{GP}(0, \Sigma^1)$$

Where:

$$\Sigma^1(x,x') = c_\sigma \mathbb{E}[h_1(x)h_1(x')]$$

We saw that:

$$h_1(x) = \sigma\left(\frac{1}{\sqrt{I}} \sum_{l=1}^{I} W_{1,l}^0 x_l\right)$$

Since $\{W_{1,l}^0\}_{l=1}^I$ are iid and follows a Gaussian distribution, $g(x) = \frac{1}{\sqrt{I}} \sum_{l=1}^I W_{1,l}^0 x_l$ is Gaussian distributed with :

$$\mathbb{E}[g(x)] = 0$$

and:

$$\begin{aligned}
\mathrm{Var}(g(x)) &= \mathbb{E}[\big(\frac{1}{\sqrt{I}} \sum_{l=1}^{I} W_{1,l}^0 x_l\big)\big(\frac{1}{\sqrt{I}} \sum_{m=1}^{I} W_{1,m}^0 x_m\big)] \\
&= \frac{1}{I} \mathbb{E}[\sum_{l=1}^{I} \sum_{m=1}^{I} W_{1,l}^0 W_{1,m}^0 x_l x_m] \\
&= \frac{1}{I} \sum_{l=1}^{I} \sum_{m=1}^{I} \mathbb{E}[W_{1,l}^0 W_{1,m}^0] x_l x_m \\
&= \frac{1}{I} \sum_{l=1}^{I} \mathbb{E}[(W_{1,l}^0)^2] x_l x_l \\
&= \frac{1}{I} \sum_{l=1}^{I} x_l x_l = \frac{x^T x}{I}
\end{aligned}$$

Because:

$$\mathbb{E}[W_{k,j}^0 W_{k,m}^0] = \begin{cases} 0 & \text{if } j \neq m, \\ \mathrm{Var}(W_{k,j}^0) = 1 & \text{if } j = m. \end{cases}$$

Then $\{g(x)\}_{i=1}^{t} \sim \mathcal{GP}(0, \Sigma^0)$ with :

$$\begin{aligned}
\Sigma^0(x, x') &= \text{Cov}(g(x), g(x')) \\
&= \mathbb{E}[\big(\frac{1}{\sqrt{I}} \sum_{l=1}^{I} W_{1,l}^0 x_l\big)\big(\frac{1}{\sqrt{I}} \sum_{m=1}^{I} W_{1,m}^0 x'_m\big)] \\
&= \frac{1}{I} \mathbb{E}[\sum_{l=1}^{I} \sum_{m=1}^{I} W_{1,l}^0 W_{1,m}^0 x_l x'_m] \\
&= \frac{1}{I} \sum_{l=1}^{I} \sum_{m=1}^{I} \mathbb{E}[W_{1,l}^0 W_{1,m}^0] x_l x'_m \\
&= \frac{1}{I} \sum_{l=1}^{I} \mathbb{E}[(W_{1,l}^0)^2] x_l x'_l \\
&= \frac{1}{I} \sum_{l=1}^{I} x_l x'_l = \frac{x^T x'}{I}
\end{aligned}$$

In final we have:

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

With:

$$\Lambda^1(x, x') = \begin{pmatrix} \Sigma^0(x, x) & \Sigma^0(x, x') \\ \Sigma^0(x', x) & \Sigma^0(x', x') \end{pmatrix}$$

4.3 Infinitely wide L-layer ANN as GP

In the previous section we saw how an infinitely wide single-hidden-layer ANN acts as a GP. We will extend this property for an infinitely wide L-hidden-layers ANN.
Let's denote:

$$f_k(x) = f_k^{L+1}(x)$$

Where:

$$\begin{cases} f_k^l(x) = \sqrt{\frac{c_\sigma}{N_{l-1}}} \sum_{j=1}^{N_{l-1}} W_{i,j}^l h_j^{l-1}(x), \quad f_k^0(x) = \sqrt{\frac{1}{I}} \sum_{j=1}^{I} W_{i,j}^0 x_j \\ h_j^l(x) = \sigma(f_j^l(x)) \end{cases}$$

With:

- N_l is the width of the l^{th} layer
- $W^l \in \mathbb{R}^{N_l \times N_{l-1}}$ the weigth matrix of the l^{th} layer
- $W_{i,j}^l \sim \mathcal{N}(0, 1)$

Given the demonstration from the previous section, we easily see that:

$$f_j^l(x) \sim \mathcal{N}(0, c_\sigma \mathbb{E}[(h_j^{l-1}(x))^2]$$

Then:

$$\big(f_j^l(x), f_j^l(x')\big) \sim \mathcal{N}(0, \Lambda^l(x, x'))$$

Where:

$$\Lambda^l(x, x') = \begin{pmatrix} \Sigma^{l-1}(x, x) & \Sigma^{l-1}(x, x') \\ \Sigma^{l-1}(x', x) & \Sigma^{l-1}(x', x') \end{pmatrix}$$

And:

$$\{f_k^l(x)\}_{i=1}^t \sim \mathcal{GP}(0, \Sigma^{l-1})$$

Where:

$$\begin{aligned} \Sigma_{i,j}^{l-1} &= c_\sigma \mathbb{E}[h_1^{l-1}(x) h_1^{l-1}(x')] \\ &= c_\sigma \mathbb{E}[\sigma(f_1^{l-1}(x)\sigma(f_1^{l-1}(x')] \\ &= c_\sigma \mathbb{E}_{(u,v)\sim\mathcal{N}(0,\Lambda^{l-1}(x,x'))}[\sigma(u)\sigma(v)] \end{aligned}$$

To conclude we have:

$$\{f_k^l(x)\}_{i=1}^t \sim \mathcal{GP}(0, \Sigma^{l-1})$$

Where:

$$\Sigma_{i,j}^{l-1} = c_\sigma \mathbb{E}_{(u,v)\sim\mathcal{N}(0,\Lambda^{l-1}(x,x'))}[\sigma(u)\sigma(v)]$$

and:

$$\Sigma^0(x, x') = \frac{x^T x'}{I}$$

We have demonstrated that each node k in layer l, denoted as $f_k^l(\cdot)$, follows a Gaussian process. In the next section, we will explore how the theory behind NNGP is connected to the theory of the NTK.

5 Infinitely wide L-layer ANN and NTK

5.1 The deterministic fashion of the NTK

To prove the deterministic nature of the NTK at initialization for an infinitely wide L-layer ANN, we must first rely on the theory behind NNGP. This is essential because both frameworks apply to infinitely wide L-layer ANN. We will consider the model to output a scalar (a regression problem for example). Changing a little bit the notations from previously, we recall that:

$$f_{\theta_0}(x) = W^{(L+1)} \sqrt{\frac{c_\sigma}{N_L}} \sigma(W^{(L)} \sqrt{\frac{c_\sigma}{N_{L-1}}} \sigma(... \sqrt{\frac{c_\sigma}{N_1}} \sigma(W^{(1)} \sqrt{\frac{1}{I}} x)))$$

Which can be seen as :

$$f_{\theta_0}(x) = f^{(L+1)}(x) \in \mathbb{R}$$

Where:

$$f^l(x) = W^{(l)} h^{(l-1)}(x) \in \mathbb{R}^{N_l}$$

$$f_i^l(x) = \sum_{j=1}^{N_{l-1}} W_{ij}^{(l)} h_j^{(l-1)}(x), \quad \mathbb{E}[f_i^l(x)] = 0$$

and:

$$h^{(l)}(x) = \sqrt{\frac{c_\sigma}{N_l}} \sigma(f^{(l)}(x)) \in \mathbb{R}^{N_l}, \quad h^{(0)}(x) = \sqrt{\frac{1}{I}} x$$

The NTK at initialization between 2 samples (x, x') is the following quantity:

$$\begin{aligned} \Theta^0(x, x') &= \big(\frac{\partial f_{\theta_0}(x)}{\partial \theta_0}\big)^T \frac{\partial f_{\theta_0}(x')}{\partial \theta_0} \\ &= \langle \frac{\partial f_{\theta_0}(x)}{\partial \theta_0}, \frac{\partial f_{\theta_0}(x')}{\partial \theta_0} \rangle \end{aligned}$$

Since we have:

$$\Theta_0 = \left(W^{(1)}, W^{(2)}, \ldots, W^{(L+1)} \right)$$

The NTK can be computed this way:

$$\Theta^0(x, x') = \sum_{l=1}^{L+1} \langle \frac{\partial f_{\theta_0}(x)}{\partial W^l}, \frac{\partial f_{\theta_0}(x')}{\partial W^l} \rangle \tag{1}$$

Let $b^{(l)}(x) \in \mathbb{R}^{N_l}$:

$$b^{(l)}(x) = \begin{cases} 1 & \text{if } l \neq L+1, \\ \sqrt{\frac{c_\sigma}{N_l}} \text{diag}(\dot{\sigma}(f^{(l)}(x))) {W^{(l+1)}}^T b^{(l+1)}(x) & \text{else.} \end{cases}$$

This way we have:

$$\frac{\partial f_{\theta_0}(x)}{\partial W^l} = b^{(l)}(x) h^{(l-1)}(x)^T \in \mathbb{R}^{N_l \times N_{l-1}} \tag{2}$$

$$\big[b^{(l)}(x) h^{(l-1)}(x)^T\big]_{m,n} = \big[b^{(l)}(x)\big]_m \cdot \big[h^{(l-1)}(x)\big]_n$$

We have:

$$\begin{aligned} \langle b^{(l)}(x) h^{(l-1)}(x)^T, b^{(l)}(x') h^{(l-1)}(x')^T \rangle &= \sum_{m=1}^{N_l} \sum_{n=1}^{N_{l-1}} \big[b^{(l)}(x) h^{(l-1)}(x)^T\big]_{m,n} \big[b^{(l)}(x') h^{(l-1)}(x')^T\big]_{m,n} \\ &= \sum_{m=1}^{N_l} \sum_{n=1}^{N_{l-1}} \big[b^{(l)}(x)\big]_m \cdot \big[h^{(l-1)}(x)\big]_n \cdot \big[b^{(l)}(x')\big]_m \cdot \big[h^{(l-1)}(x')\big]_n \\ &= \sum_{m=1}^{N_l} \big[b^{(l)}(x)\big]_m \cdot \big[b^{(l)}(x')\big]_m \cdot \sum_{n=1}^{N_{l-1}} \big[h^{(l-1)}(x)\big]_n \cdot \big[h^{(l-1)}(x')\big]_n \\ &= \langle b^{(l)}(x), b^{(l)}(x') \rangle \cdot \langle h^{(l-1)}(x), h^{(l-1)}(x') \rangle \end{aligned}$$

Then using equality 1 and 2:

$$\Theta^0(x,x') = \sum_{l=1}^{L+1} \langle b^{(l)}(x) \| b^{(l)}(x') \rangle \langle h^{(l-1)}(x) \| h^{(l-1)}(x') \rangle \tag{3}$$

In a first part, we will focus on $\langle h^{(l-1)}(x) \| h^{(l-1)}(x') \rangle$:

$$\begin{aligned}
\mathbb{E}[f_i^{(l)}(x).f_i^{(l)}(x')] &= \mathbb{E}[\big(\sum_{j=1}^{N_{l-1}} W_{ij}^{(l)} h_j^{(l-1)}(x)\big)\big(\sum_{k=1}^{N_{l-1}} W_{ik}^{(l)} h_k^{(l-1)}(x')\big)] \\
&= \mathbb{E}[\sum_{j=1}^{N_{l-1}} \sum_{k=1}^{N_{l-1}} W_{ij}^{(l)} W_{ik}^{(l)} h_j^{(l-1)}(x) h_k^{(l-1)}(x')] \\
&= \sum_{j=1}^{N_{l-1}} \sum_{k=1}^{N_{l-1}} \mathbb{E}[W_{ij}^{(l)} W_{ik}^{(l)}] h_j^{(l-1)}(x) h_k^{(l-1)}(x') \\
&= \sum_{j=1}^{N_{l-1}} h_j^{(l-1)}(x) h_j^{(l-1)}(x') \\
&= \langle h^{(l-1)}(x) \| h^{(l-1)}(x') \rangle
\end{aligned}$$

Furthermore we know from Section 4.3:

$$\mathbb{E}[f_i^{(l)}(x).f_i^{(l)}(x')] \underset{N_{l-1}\to\infty}{\longrightarrow} \Sigma^{l-1}(x,x')$$

Then:

$$\langle h^{(l-1)}(x) \| h^{(l-1)}(x') \rangle \underset{N_{l-1}\to\infty}{\longrightarrow} \Sigma^{(l-1)}(x,x') \tag{4}$$

From now, let's focus on:

$$\langle b^{(l)}(x), b^{(l)}(x') \rangle$$

For simplicity, we denote $D^{(l)}(x) = diag(\dot{\sigma}(f^l(x)) \in \mathbb{R}^{N_l \times N_l}$

$$b^{(l)}(x) = \sqrt{\frac{c_\sigma}{N_l}} D^{(l)}(x) {W^{(l+1)}}^T b^{(l+1)}(x)$$

$$\big\langle b^{(l)}(x), b^{(l)}(x') \big\rangle = \frac{c_\sigma}{N_l} \big\langle D^{(l)}(x) {W^{(l+1)}}^T b^{(l+1)}(x), D^{(l)}(x') {W^{(l+1)}}^T b^{(l+1)}(x') \big\rangle$$

Since $W^{(l+1)}$ are not independent to $b^{(l+1)}(x)$. Changing $W^{(l+1)}$ by $\tilde{W}^{(l+1)}$, followings the same distribution as $W^{(l+1)}$ and independant to $b^{(l+1)}(x)$ won't change the limits.
Then:

$$\begin{aligned}\left\langle b^{(l)}(x), b^{(l)}(x')\right\rangle &= \frac{c_\sigma}{N_l}\left\langle D^{(l)}(x)W^{(l+1)^T}b^{(l+1)}(x), D^{(l)}(x')W^{(l+1)^T}b^{(l+1)}(x')\right\rangle \\ &\simeq \frac{c_\sigma}{N_l}\left\langle D^{(l)}(x)\tilde{W}^{(l+1)^T}b^{(l+1)}(x), D^{(l)}(x')\tilde{W}^{(l+1)^T}b^{(l+1)}(x')\right\rangle \\ &= \frac{c_\sigma}{N_l}\sum_{i=1}^{N_l}\dot{\sigma}(f_i^{(l)}(x))\dot{\sigma}(f_i^{(l)}(x'))\sum_{k=1}^{N_{l+1}}\sum_{m=1}^{N_{l+1}}\tilde{W}_{ki}^{(l+1)}\tilde{W}_{mi}^{(l+1)}b_k^{(l+1)}(x)b_m^{(l+1)}(x')\end{aligned}$$

$$\begin{aligned}&\xrightarrow[N_l\to\infty]{} c_\sigma\mathbb{E}_{u,v\sim\mathcal{N}(0,\Lambda^{(l)});w_z\sim\mathcal{N}(0,1)\forall z\in[N_{l+1}]}[\dot{\sigma}(u)\dot{\sigma}(v)\sum_{k=1}^{N_{l+1}}\sum_{m=1}^{N_{l+1}}w_kw_mb_k^{(l+1)}(x)b_m^{(l+1)}(x')] \\ &= c_\sigma\mathbb{E}_{u,v\sim\mathcal{N}(0,\Lambda^{(l)})}[\dot{\sigma}(u)\dot{\sigma}(v)]\mathbb{E}_{w_z\sim\mathcal{N}(0,1)\forall z\in[N_{l+1}]}[\sum_{k=1}^{N_{l+1}}\sum_{m=1}^{N_{l+1}}w_kw_mb_k^{(l+1)}(x)b_m^{(l+1)}(x')] \\ &= \dot{\Sigma}^{(l)}(x,x')\sum_{k=1}^{N_{l+1}}\sum_{m=1}^{N_{L+1}}b_k^{(l+1)}(x)b_m^{(l+1)}(x')\mathbb{E}_{w_z\sim\mathcal{N}(0,1)\forall z\in[N_{l+1}]}[w_kw_m] \\ &= \dot{\Sigma}^{(l)}(x,x')\sum_{k=1}^{N_{l+1}}b_k^{(l+1)}(x)b_k^{(l+1)}(x') \\ &= \dot{\Sigma}^{(l)}(x,x')\left\langle b^{(l+1)}(x), b^{(l+1)}(x')\right\rangle \\ &= \prod_{l'=l}^{L}\dot{\Sigma}^{(l')}(x,x')\end{aligned}$$

So:

$$\left\langle b^{(l)}(x), b^{(l)}(x')\right\rangle \xrightarrow[N_l\to\infty]{} \prod_{l'=l}^{L}\dot{\Sigma}^{(l')}(x,x') \tag{5}$$

Then using equality 3, 4 and 5:

$$\Theta^0(x,x') \xrightarrow[N_l\to\infty]{\forall l\in[L]} \sum_{l=1}^{L+1}\Sigma^{(l-1)}(x,x')\prod_{l'=l}^{L}\dot{\Sigma}^{(l')}(x,x') \tag{6}$$

That ends the demonstration of the NTK being deterministic at initialization.

5.2 The NTK is constant during training

During the training of an ANN, the weights are continuously updated, leading to changes in the network's parameters. Consequently, the NTK, which depends on these parameters, typically evolves throughout the training. However, in the case of an infinitely wide ANN, the NTK remains constant during training, as demonstrated in [1].

$$\Theta^t = \Theta_0$$

To verify this property, we will examine the evolution of the relative norm of the NTK across several architectures with varying widths. We expect that as the architecture's width increases, the relative norm will remain closer to zero, indicating greater constancy during training. These initial experiments are conducted on the Boston dataset, a regression task where the goal is to predict the median home value in a neighborhood based on various features related to that neighborhood.

$$\text{relative norm } = \frac{\|\Theta^t - \Theta^0\|^2}{\|\Theta^0\|^2}$$

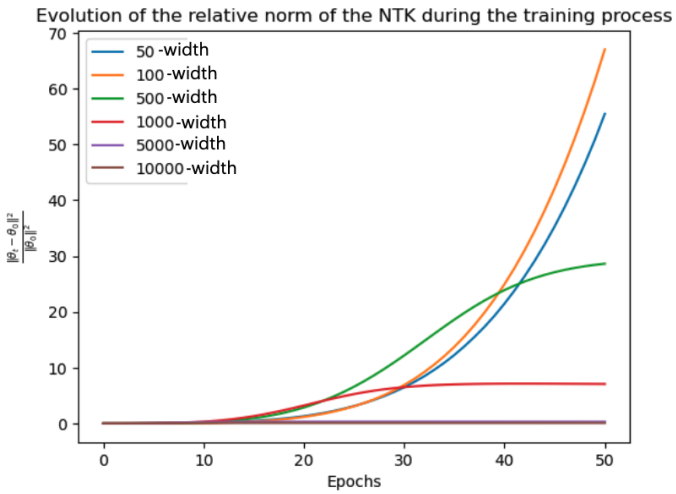

Figure 2: Evolution of the relative norm of the NTK during the training process for a regression problem

The results align with our expectations: as the width of the network increases, the evolution of the relative norm becomes more constant and remains closer to zero. However, there is an exception with models of width 50 and 100, where the smaller model appears more stable than the width-100 model in our experiments. Despite this anomaly, the results for the larger models remain consistent with our hypothesis.
While the NTK theory we explored was originally developed for regression problems, its properties have been extended to ANN with convolutional layers, as shown in [1], and remain valid for classification tasks. In the case of convolutional layers, the network width is defined by the number of output channels in each convolution.
We will conduct a similar study on a classification problem using the MNIST dataset. The first part will focus on a standard ANN, investigating the con-

stancy of the NTK during classification with dense layers. The second part will extend the analysis to ANNs with convolutional layers.

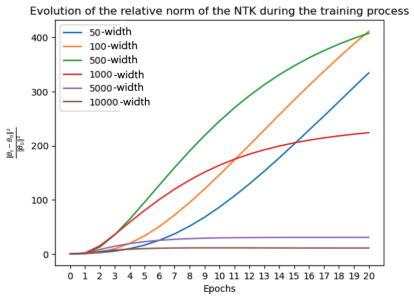

Figure 3: Evolution of the relative norm of the NTK during the training process for a classification problem using classic ANN

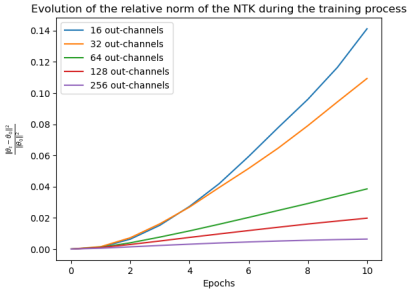

Figure 4: Evolution of the relative norm of the NTK during the training process for a classification problem using Convolutional CNN

Figure 3 illustrates the evolution of the relative norm of the NTK during training for a classification problem using classic ANNs. The results are consistent with previous findings: as the model size increases, the NTK becomes more constant and remains closer to zero throughout the training process. However, for smaller models, this relationship does not hold; smaller models can sometimes exhibit greater constancy and remain closer to zero compared to larger models. However, for very large models, the inequality holds true.
Figure 4, which shows the evolution of the relative norm of the NTK for a convolutional CNN, demonstrates similar results to the classic ANN. In fact, the results are more clear for the convolutional layers: the larger the number of out channels, the more consistent the NTK norm becomes, staying close to zero throughout the training process. This reinforces the idea that increasing the number of out channels improves the constancy close to 0 of the NTK during training.

6 Bounding the Training Loss with the NTK

Having a bound on the evolution of the training loss at initialization allows us to assess whether an architecture is well-positioned to progress effectively in the loss space. This can be particularly useful in Neural Architecture Search, where comparing numerous architectures at initialization is critical for selecting promising candidates.

6.1 Bound for a regression problem

Now, let's revisit regression problems. We consider we train using:

$$\mathcal{L}_t = \frac{1}{2}\|\mathcal{Y} - f_{\theta_t}(\mathcal{X})\|_2^2$$

Then:

$$\nabla_{\theta_t}\mathcal{L}_t = (f_{\theta_t}(\mathcal{X}) - \mathcal{Y})$$

We will demonstrate that:

$$\|\mathcal{Y} - f_{\theta_t}(\mathcal{X})\|_2^2 \leq e^{-\lambda_{min}^t t}\|\mathcal{Y} - f_{\theta_0}(\mathcal{X})\|_2^2 \tag{7}$$

Here, $\lambda_{\min}^t$ represents the smallest eigenvalue of the NTK at time t. In the case of an infinitely wide ANN, we can substitute $\lambda_{\min}^t$ with $\lambda_{\min}^0$, which is the smallest eigenvalue of the NTK at initialization. Therefore, we obtain:

$$\|\mathcal{Y} - f_{\theta_t}(\mathcal{X})\|_2^2 \leq e^{-\lambda_{min}^0 t}\|\mathcal{Y} - f_{\theta_0}(\mathcal{X})\|_2^2 \tag{8}$$

Hence, The larger λ_{min}^0, the lower $\mathcal{L}_t$ is susceptible to be.
Let's proove 7:
We recall that :

$$\frac{\partial f_{\theta_t}(x_j)}{\partial t} = -\sum_{i=1}^{N}\left(\frac{\partial f_{\theta_t}(x_j)}{\partial \theta_t}\right)^T \frac{\partial f_{\theta_t}(x_i)}{\partial \theta_t}\frac{\partial l(f_{\theta_t}(x_i), y_i)}{\partial f}$$

Then:

$$\frac{\partial f_{\theta_t}(\mathcal{X})}{\partial t} = \frac{\partial\big(f_{\theta_t}(\mathcal{X}) - \mathcal{Y}\big)}{\partial t} = -\Theta^t(f_{\theta_t}(\mathcal{X}) - \mathcal{Y}) = \Theta^t(\mathcal{Y} - f_{\theta_t}(\mathcal{X}))$$

Considering the spectral decomposition of Θ^t, we have:

$$\Theta^t = P_t D_t P_t^T$$

Where:

- P_t is orthogonal,
- D_t is diagonal with each entry being eigenvalue of Θ^t.

Then we have:

$$\begin{aligned}
\min_{\|x\|=1} x^T\Theta^t x &= \min_{\|x\|=1} x^T P_t D_t P_t^T x \\
&= \min_{\|x\|=1} \big(P_t^T x\big)^T D_t\big(P_t^T x\big) \\
&= \min_{\|y\|=1} y^T D_t y \\
&= \lambda_{\min}^t \quad \text{(the minimal eigenvalue of } \Theta^t\text{)}
\end{aligned} \tag{9}$$

Furthermore:

$$\begin{aligned}\frac{\partial\|\mathcal{Y}-f_{\theta_t}(\mathcal{X})\|_2^2}{\partial t} &= -\big(\mathcal{Y}-f_{\theta_t}(\mathcal{X})\big)\frac{\partial f_{\theta_t}(\mathcal{X})}{\partial t}\\ &= -\big(\mathcal{Y}-f_{\theta_t}(\mathcal{X})\big)\Theta^t(\mathcal{Y}-f_{\theta_t}(\mathcal{X}))^T\\ &= -\|\mathcal{Y}-f_{\theta_t}(\mathcal{X})\|_2^2\frac{\big(\mathcal{Y}-f_{\theta_t}(\mathcal{X})\big)}{\|\mathcal{Y}-f_{\theta_t}(\mathcal{X})\|}\Theta^t\frac{\big(\mathcal{Y}-f_{\theta_t}(\mathcal{X})\big)^T}{\|\mathcal{Y}-f_{\theta_t}(\mathcal{X})\|}\\ &\leq -\lambda_{min}^t\|\mathcal{Y}-f_{\theta_t}(\mathcal{X})\|_2^2 \quad \text{(using 9)} \end{aligned} \tag{10}$$

We will focus on the variation of $\exp(\lambda_{min}^t t)\|\mathcal{Y}-f_{\theta_t}(\mathcal{X})\|_2^2$

$$\begin{aligned}&\frac{\partial}{\partial t}\exp(\lambda_{min}^t t)\|\mathcal{Y}-f_{\theta_t}(\mathcal{X})\|_2^2\\ &= \lambda_{min}^t\exp(\lambda_{min}^t t))\|\mathcal{Y}-f_{\theta_t}(\mathcal{X})\|_2^2+\exp(\lambda_{min}^t t))\frac{\partial}{\partial t}\|\mathcal{Y}-f_{\theta_t}(\mathcal{X})\|_2^2\\ &\leq \lambda_{min}^t\exp(\lambda_{min}^t t))\|\mathcal{Y}-f_{\theta_t}(\mathcal{X})\|_2^2-\exp(\lambda_{min}^t t))\lambda_{min}^t\|\mathcal{Y}-f_{\theta_t}(\mathcal{X})\|_2^2\\ &= 0\end{aligned}$$

Thus $\exp(\lambda_{min}^t t)\|\mathcal{Y}-f_{\theta_t}(\mathcal{X})\|_2^2$ is decreasing over time:

$$\begin{aligned}\exp(\lambda_{min}^t t)\|\mathcal{Y}-f_{\theta_t}(\mathcal{X})\|_2^2 &\leq \exp(\lambda_{min}^0\times 0)\|\mathcal{Y}-f_{\theta_0}(\mathcal{X})\|_2^2\\ &= \|\mathcal{Y}-f_{\theta_0}(\mathcal{X})\|_2^2\end{aligned}$$

We thus have:

$$\|\mathcal{Y}-f_{\theta_t}(\mathcal{X})\|_2^2 \leq \exp(-\lambda_{min}^t t)\|\mathcal{Y}-f_{\theta_0}(\mathcal{X})\|_2^2$$

Then:

$$\mathcal{L}_t \leq \exp(-\lambda_{min}^t t)\mathcal{L}_0$$

To verify this property, we continue using the Boston dataset. At each iteration of the training process, we will compare the value of the loss function $\mathcal{L}_t$ with its theoretical bounds.

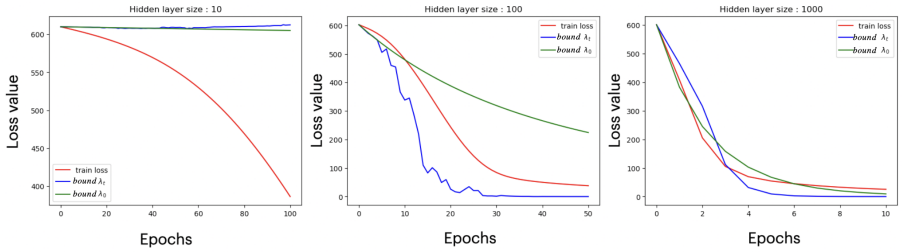

Figure 5: Evolution of the training loss and its theoretical bounds

Examining figure 5, we observe that as the width of our dense network increases, the theoretical bounds (in red and blue) become closer to the training error. For

the network with width 10, the theoretical bounds correctly upper-bound the training error but remain relatively loose. Indeed, they stay constant, equal to the error observed at initialization, which can be explained by very small eigenvalues. In contrast, for wider networks, the bounds better follow the trend of the error curve, although they sometimes no longer upper-bound it. This could be due to the fact that the NTK is not computed on the entire training set but only on a subsample. We conclude that the inequality becomes more relevant in the context of wide networks.

6.2 Bound for a classification problem

The previous inequality applies to regression problems. However, in classification problems, the NTK takes the form of a tensor with shape $(N, N, n_{\text{classe}}, n_{\text{classe}})$. The training loss differs, and calculating the minimal eigenvalue of the NTK becomes non-trivial due to its complex structure. As a result, the previous inequality does not hold for classification problems. In this paper, we introduce a new inequality tailored to classification tasks, which provides a bound on the training loss for such problems.

In multi-class classification problems, the output of our ANN (typically a CNN) is a vector of size n_{classe}. For the purpose of this demonstration, we assume the ANN uses a softmax function as the final activation. Each coordinate of the output vector represents the model's predicted probability that a given sample belongs to a specific class. In this demonstration, we will consider the squared cross-entropy loss as the training loss. We denote by $\tilde{f}_{\theta_t}(x) = f_{\theta_t}(x)_y$ the probability to belong to the true class given the model parameters. We expect each $\tilde{f}_{\theta_t}(x)$ to be close to 1. We denote by $\mathcal{L}_t$ our theorical loss function:

$$\mathcal{L}_t = \frac{1}{2}\|\mathbb{1} - \tilde{f}_{\theta_t}(\mathcal{X})\|_2^2$$

To obtain our new inequality:

$$\|\mathbb{1} - f_{\theta_t}(\mathcal{X})\|_2^2 \leq \exp(-\lambda_{min}^t t)\|\mathbb{1} - f_{\theta_0}(\mathcal{X})\|_2^2 \tag{11}$$

It is enough to replace $\mathcal{Y}$ by $\mathbb{1}$ in the proof for regression in 6.1. Using these $\tilde{f}_{\theta_t}(x)$ which outputs a single scalar being the probability given the model to belong to the true class y, the NTK Θ^t is now a squared matrix with:

$$\Theta_{i,j}^t = \langle \frac{\partial \tilde{f}_{\theta_t}(x_i)}{\partial \theta_t}; \frac{\partial \tilde{f}_{\theta_t}(x_j)}{\partial \theta_t} \rangle$$

And computing its minimal eigenvalue λ_{min}^t is now trivial.

In Figure 6, we test our theoretical bounds on classification tasks using classic ANNs (without convolutions). The results are similar to those observed in regression problems: as the width of the ANN increases, our bounds more accurately approximate and constrain the actual value of the loss function during training. However, for the model with a width of 1000, the bound using

$\lambda_{\min}^t$ fails to upper bound the loss, which is contrary to our expectations and inconsistent with the theory.

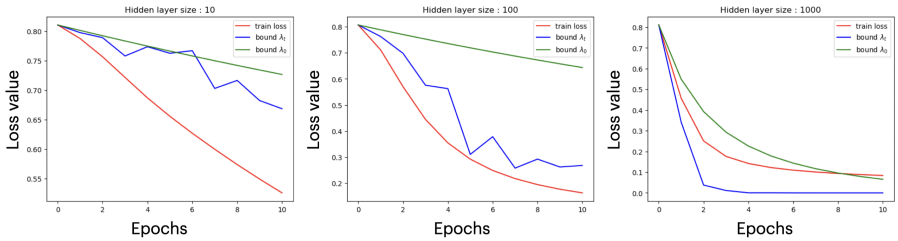

Figure 6: Evolution of the training loss and its theoretical bounds using classic ANN

We performed a comparable analysis for CNNs, as depicted in Figure 7. In our exploration of classification tasks using CNNs, we found results that closely resembled those observed in regression scenarios. Specifically, as the width of the CNN increased, our theoretical bounds offered a more accurate approximation and tighter constraints on the actual loss function during training. Notably, in these experiments, the training loss was well-bounded. However, due to memory limitations, we were unable to test CNNs with output channels exceeding 100.

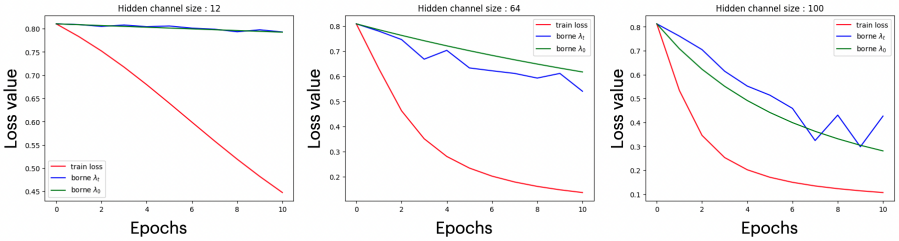

Figure 7: Evolution of the training loss and its theoretical bounds using CNN

7 Conclusion

The Neural Tangent Kernel provides valuable insights into the training dynamics of wide neural networks. It exhibits several properties and offers interesting inequalities that enhance our understanding of these systems. Furthermore, the NTK has practical applications, such as in Physics-Informed Neural Networks (PINNs) [5] and Neural Architecture Search (NAS) [6], demonstrating its versatility and potential for advancing research in the field.

Références

[1] Sanjeev Arora et al. "On Exact Computation with an Infinitely Wide Neural Net". In: *CoRR* abs/1904.11955 (2019). arXiv: 1904.11955. URL: http://arxiv.org/abs/1904.11955.

[2] Arthur Jacot, Franck Gabriel, and Clément Hongler. "Neural Tangent Kernel: Convergence and Generalization in Neural Networks". In: *CoRR* abs/1806.07572 (2018). arXiv: 1806.07572. URL: http://arxiv.org/abs/1806.07572.

[3] Jaehoon Lee et al. *Deep Neural Networks as Gaussian Processes*. 2018. arXiv: 1711.00165 [stat.ML]. URL: https://arxiv.org/abs/1711.00165.

[4] Radford M. Neal. "Priors for Infinite Networks". In: *NIPS 1994* (1994). URL: https://glizen.com/radfordneal/ftp/pin.pdf.

[5] Sifan Wang, Xinling Yu, and Paris Perdikaris. "When and why PINNs fail to train: A neural tangent kernel perspective". In: *CoRR* abs/2007.14527 (2020). arXiv: 2007.14527. URL: https://arxiv.org/abs/2007.14527.

[6] Jingjing Xu et al. "KNAS: Green Neural Architecture Search". In: *CoRR* abs/2111.13293 (2021). arXiv: 2111.13293. URL: https://arxiv.org/abs/2111.13293.