We thank all reviewers for their valuable feedback and appreciating our technical contributions. We first address some common concerns.

-*More discussion on experimental results*:

• First, we discuss more about the relation between GNTKs and GNNs. In Section 5.2, through ablation studies, we aimed to verify 1) whether performance of GNNs is similar to GNTKs and 2) whether various techniques used for improving performance of GNNs are transferable to GNTKs. For example, in Figure 2, while the observation that GNNs with more layers perform better on bioinformatics data is not new, we observed the same trend for GNTKs. In Figure 3, we also show that jumping knowledge, which has been shown to be effective in GNNs, is also effective in GNTKs. We will elaborate more on these connections in the final version.

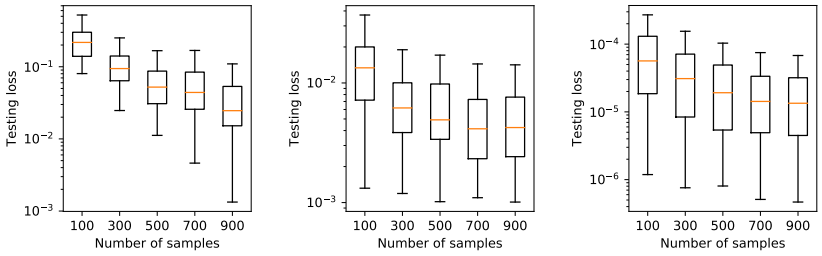
• Second, Reviewer #2 raised a great point about computational complexity. Indeed, running GNTK is much faster than running GNN. On IMDB-B dataset, running GIN with the default setup (official implementation of [26]) takes 19 minutes on a TITAN X GPU and running GNTK only takes 2 minutes. We will add more details.

-*Reproducibility*: We will open-source our code and datasets. We mistakenly chose "not applicable" for this question in Reproducibility Response.

To Reviewer #1:

-*Graph convolutional layer*: Our paper does contain a graph (spatial) convolutional layer which we call the aggregation step ($\sum_{v\in\mathcal{N}(u)\cup\{u\}} \mathbf{h}_v^{(\ell-1)}$). See line 89, 96 for GNNs and line 148 for GNTKs. We will clarify this more explicitly.

-*Lemma 4.2/4.3, conditions on line 193*: The aim of our theoretical analysis is to characterize the function class which can be efficiently learned by GNTK. This is a standard type of results in learning theory. The sample complexity depends on α, the coefficient of polynomials, and the norm of β. This kind of dependency is standard in learning polynomials and functions that can be expressed as summation of polynomials with fast decaying coefficients. If α or the norm of β is large, we do need more samples. See [1] and references therein. The tightness of the bound is an open problem. In general, it is impossible to verify these conditions on real datasets, as this is a quantity related to the underlying function which we cannot observe. Nevertheless, empirically we use simulations to show the applicability of these theoretical results, as given in Fig. 1. We will include these simulation results in the paper.



(a) $\ell = 1$. (b) $\ell = 2$. (c) $\ell = 4$.

Figure 1: We perform simulations using 1000 graphs in the IMDB-B dataset. For each graph, we generate the label according to $y = \sum_{u\in V}(\bar{h}_u^\top \beta)^\ell$ for $\ell \in \{1, 2, 4\}$, where we sample β from the uniform distribution over the unit ball (cf. Eq. (4) in our submission). By construction, these functions satisfy the condition in line 193. We use 100 samples for testing and use 100, 300, 500, 700 or 900 remaining samples for training. We repeat each simulation for 100 times and report the testing loss.

-*Theorem 4.1*: This is a classical result for kernel regression. The high-level idea is to use Rademacher complexity to bound the difference between empirical loss and population loss, and use $\mathbf{y}^\top \mathbf{\Theta^{-1}} \mathbf{y} \cdot \mathrm{tr}(\mathbf{\Theta})$ to bound the Rademacher complexity. The whole proof is given in [3] and we will provide a short outline in the final version.

To Reviewer #2:

-*Statistical test:* We will add a statistical test in the final version. Thanks for the suggestion.

-*Line 118 / typos / missing conclusions*: We will fix them accordingly. Thanks for pointing out.

To Reviewer #3:

-*Eq. (2)*: For the BLOCK operation with $R = 2$, we first apply the aggregation operation to gather local information, and then apply a two-layer MLP to create non-linearity. We will clarify this in the final version.

-*Experimental details:* For experiments, we use the same setup as in [26]. We have provided some descriptions of the experimental setup in Section B. We will add more detailed description in the final version.

-*Too many parameters*: GNTK do have some hyper-parameters. However, note that GNTK has strictly smaller number of hyper-parameters than GNN since we do not need to tune the learning rate, momentum, weight decay, batch size and the width of the MLP layers for GNTK. Furthermore, we found on bioinformatics datasets, we got consistently good results by setting the number of BLOCK operations to be 10, the number of MLP layers to be 1 and c_u to be $1/|\mathcal{N}(u)|$. We get 75.3% accuracy on PROTEINS, 67.9% on PTC, 83.6% on NCI1. For social network datasets, by setting the number of BLOCK operations to be 2, the number of MLP layers to be 2 and c_u to be 1, we get 76.7% accuracy on IMDB-B, 52.8% on IMDB-M, and 83.3% on COLLAB. We will discuss this point in the final version.