# New Insights into Graph Convolutional Networks using Neural Tangent Kernels

**Anonymous Author(s)**
Affiliation
Address
`email`

## Abstract

Graph Convolutional Networks (GCNs) have emerged as powerful tools for learning on network structured data. Although empirically successful, GCNs exhibit certain behaviour that has no rigorous explanation—for instance, the performance of GCNs significantly degrades with increasing network depth, whereas it improves marginally with depth using skip connections.

This paper focuses on semi-supervised learning on graphs, and explains the above observations through the lens of Neural Tangent Kernels (NTKs). We derive NTKs corresponding to infinitely wide GCNs (with and without skip connections). Subsequently, we use the derived NTKs to identify that, with suitable normalisation, network depth does not always drastically reduce the performance of GCNs—a fact that we also validate through extensive simulation. Furthermore, we propose NTK as an efficient 'surrogate model' for GCNs that does not suffer from performance fluctuations due to hyper-parameter tuning since it is a hyper-parameter free deterministic kernel. The efficacy of this idea is demonstrated through a comparison of different skip connections for GCNs using the surrogate NTKs.

## 1 Introduction

Graph structured data are ubiquitous in various domains, including social network analysis, bioinformatics, communications engineering among others. In recent years, graph neural networks have become an indisputable choice for various learning problems on graphs, and have been employed in a wide range of applications across domains. Several variants of graph neural networks have been proposed, including graph convolutional network [Kipf and Welling, 2017], graph recurrent network [Scarselli et al., 2008, Li et al., 2016], graph attention network [Velickovic et al., 2018], to name a few. The popularity of graph neural networks can be attributed to their ability to tackle two conceptually different learning problems on graphs. In *supervised learning on graphs*, each data instance is a graph and the goal is to predict a label for each graph (for example, a protein structure). In contrast, *semi-supervised learning on graphs* (also called *node classification* or *graph transduction*) refers to the problem of predicting the labels of nodes in a single graph. For instance, given the memberships of a few individuals in a social network, the goal is to predict affiliations of others.

This work focuses on the latter problem of semi-supervised learning. GCNs, along with its variants that locally aggregate information in the neighbourhood of each node, have proved to be superior methods in practice [Defferrard et al., 2016, Kipf and Welling, 2017, Chen et al., 2018a, Wu et al., 2019, Chen et al., 2020], outperforming classical, and well-studied, graph embedding based approaches. Among the different variants of GCNs, we focus on the methods based on approximations of spectral graph convolutions [Defferrard et al., 2016, Kipf and Welling, 2017], rather than spatial graph convolutions [Hamilton et al., 2017, Xu et al., 2019]. Surprisingly, these papers suggest shallow networks for the best performance, and unlike the standard neural networks that gain advantage with depth, the performance of GCN has been reported to decrease for deeper nets. This appears to be due to the over smoothing effect of applying many convolutions, that is, with repeated application of the graph diffusion operator in each layer, the feature information gets averaged out to a degree where it becomes uninformative. As a solution to this, Chen et al. [2020] and Kipf and Welling [2017] proposed different formulations of skip connections in GCNs that overcome the smoothing effect and thus outperform the vanilla GCN empirically. These networks achieve state-of-the-art results by directly operating on graphs which enables effective capturing of the complex structural information as well as the features associated with the entities. However, similar to standard neural networks, tuning the hyper-parameters is particularly hard due to the highly non-convex objective function and the over-parameterised setup making it computationally intense. As a result, there is no theoretical framework that supports rigorous analysis of graph neural networks. Furthermore, the graph convolutions increase the difficulty of analysis. Motivated by this, we are interested in a more formal approach to analyze GCNs and, specifically, to understand the influence of depth.

Explaining the empirical evidence of deep neural networks through mathematical rigour is an active area of research. In contrast, theoretical analysis of graph neural networks has been limited in the literature. From the perspective of learning theory, generalisation error bounds have been derived for graph neural networks using complexity measures like VC Dimension and Rademacher complexity [Scarselli et al., 2018, Garg et al., 2020]. However, it is often debated whether generalisation error bounds can explain the performance of deep neural networks [Neyshabur et al., 2017]. Another line of research relies on the connection between graph convolutions and belief propagation [Dai et al., 2016] to analyse the behaviour of graph neural networks in both supervised and semi-supervised settings using cavity methods and mean field approaches [Zhou et al., 2020b, Kawamoto et al., 2019, Chen et al., 2018b]. However, the above lines of research do not completely explain the empirical trends observed in GCNs, especially with regards to the aspects analysed in our work.

In this paper, we explain the empirically observed trends of GCNs using the recently introduced *Neural Tangent Kernel* (NTK) [Jacot et al., 2018]. NTK was proposed to describe the behaviour and generalisation properties of randomly initialised fully connected neural networks during training by gradient descent with infinitesimally small learning rate. Jacot et al. [2018] also showed that, as the network width increases, the change in the kernel during training decreases and hence, asymptotically, one may replace an infinitely wide neural network by a deterministic kernel machine, where the kernel (NTK) is defined by the gradient of the network with respect to its parameters as

$$\Theta(x, x') = \mathop{\mathbb{E}}_{W \sim \mathcal{N}} \left[ \left\langle \frac{\partial F(W, x)}{\partial W}, \frac{\partial F(W, x')}{\partial W} \right\rangle \right]. \tag{1}$$

Here $F(W, x)$ represents the output of the network at data point $x$ and the expectation is with respect to $W$, that is, all the parameters of the network randomly sampled from Gaussian distribution $\mathcal{N}$. There has been criticism of the 'infinite width' assumption being too strong to model real (finite width) neural networks, and empirical results show that NTK often performs worse than the practical networks [Arora et al., 2019, Lee et al., 2019]. Nevertheless, theoretical insights on neural network training gained from NTK have proved to be valuable, particularly in showing how gradient descent can achieve good generalisation properties [Du et al., 2019a]. Subsequent works have derived NTK to analyse different neural network architectures in infinite width limit, including convolutional networks, recurrent networks among others [Arora et al., 2019, Du et al., 2018, 2019a, Alemohammad et al., 2021]. The most relevant work in the context of our discussion is the work of Du et al. [2019b] that derived NTK for graph neural networks in the supervised setting (each graph is a data instance to be classified) and empirically showed that graph NTK outperforms most graph neural networks as well as other graph kernels for the problem of graph classification.

**Focus of this paper and contributions.** The focus of the present paper differs from existing work on graph NTK [Du et al., 2019b] in two key aspects—we derive NTK for semi-supervised node classification and, more importantly, we use the derived NTKs to rigorously analyse corresponding GCN architectures and demonstrate the cause for surprising trends observed empirically in GCNs, as opposed to standard deep neural networks. More precisely, we make the following contributions:

**1.** In Section 2, we derive the NTKs for GCNs used in semi-supervised node classification [Kipf and Welling, 2017, Wu et al., 2019] in infinite width limit. In contrast to simplifying assumptions in most NTKs derivations, we allow a non-linear (sigmoid) pooling in the last layer—a natural choice in practical networks for binary classification. Using the derived NTK and through extensive simulation, we show that the performance of GCN varies considerably for different hyper-parameters, but NTK captures the general trend of the best possible performance of GCN.

**2.** Due to the observation that NTK is a hyper-parameter free alternative to GCN that approximates the behaviour of GCNs, we suggest NTK as an efficient surrogate for GCN that could be used to identify the optimal network architecture. We demonstrate this idea in Section 3 by deriving the NTKs corresponding to GCNs with different skip connections [Chen et al., 2020, Kipf and Welling, 2017], and we make recommendation on the skip connection for improved performance through empirical studies of the NTKs. The NTK surrogate can be further used to assess the relative importance of structure and feature information in a graph dataset.

**3.** In Section 4, we use our NTK based analysis to investigate the popular belief that the performance of vanilla GCN degrades drastically with increasing network depth. We demonstrate that this observation is due to instabilities in the network training, which results in performance fluctuations of vanilla GCN, and that can be addressed by appropriate normalisation of the features at each level. The fluctuations can also be reduced by adding skip connections, even without appropriate normalisation.

**4.** In Section 5, we explain an empirical finding—unlike vanilla GCNs, the performance of NTK for certain skip connections converge with network depth. This is because the NTKs for skip connections converge with network depth, whereas this is less prominent in the case of NTK for vanilla GCNs.

We conclude in Section 6, and provide the NTK derivations and further experimental details in the appendix.

**Notation.** We represent the matrix Hadamard (entry-wise) product by $\odot$ and the scalar product by $\langle ., . \rangle$. We use $M^{\odot k}$ to denote Hadamard product of matrix $M$ with itself repeated $k$ times. Let $\mathcal{N}(\mu, \Sigma)$ be Gaussian distribution with mean $\mu$ and co-variance $\Sigma$. For a function $\sigma(.)$, we use $\dot{\sigma}(.)$ to represent its derivative. We use $\mathbf{1}_{n\times n}$ for the $n \times n$ matrix of ones, $I_n$ for identity matrix of size $n \times n$, $\mathbb{E}\left[.\right]$ for expectation, $\|.\|_F$ denotes Frobenius norm, and $[d] = \{1, 2, \ldots, d\}$.

## 2 NTK Captures the Behaviour of Vanilla GCN

We consider the problem of node classification in graphs in a semi-supervised setting,[1] where the labels are observed only for a subset of the nodes. We start with the formal setup and NTK derivation for the standard (vanilla) GCN proposed in Kipf and Welling [2017].

**Formal Setup.** Given a graph with $n$ nodes and a set of node features $\{x_i\}_{i=1}^n \subset \mathbb{R}^f$, we may assume without loss of generality that the set of observed labels $\{y_i\}_{i=1}^m$ correspond to first $m$ nodes. We consider a binary classification problem in this paper to simplify the NTK derivation, that is $y_i \in \{\pm 1\}$, but this could be extended to multi-class problems. The goal is to correctly predict the $n-m$ unknown labels $\{y_i\}_{i=m+1}^n$. We represent the observed labels of $m$ nodes as $Y \in \{\pm 1\}^{m\times 1}$, and the node features as $X \in \mathbb{R}^{n\times f}$ with the assumption that entire $X$ is available during training. We define $S$ to be the graph diffusion operator. The analysis holds for any diffusion $S$, but for simulations, we consider the symmetric degree normalized diffusion $S := (D+I_n)^{-\frac{1}{2}}(A+I_n)(D+I_n)^{-\frac{1}{2}}$ where $A$ is the adjacency matrix and $D$ is the degree matrix. We define the GCN of depth $d$ as,

$$F_W(X, S) := \Phi\left(\sqrt{\frac{c_\sigma}{h_d}} S \ldots \sigma\left(\sqrt{\frac{c_\sigma}{h_1}} S \sigma\left(SXW_1\right) W_2\right) \ldots W_{d+1}\right) \tag{2}$$

where $W := \{W_i \in \mathbb{R}^{h_{i-1}\times h_i}\}_{i=1}^{d+1}$ is the set of learnable weight matrices with $h_0 = f$ and $h_{d+1} = 1$, and $\Phi : \mathbb{R} \to (-1, +1)$ is re-scaled sigmoid since we consider binary node classification with labels in $\{\pm 1\}$, $h_i$ is the size of layer $i \in [d]$ and $\sigma : \mathbb{R} \to \mathbb{R}$ is the point-wise activation function. We initialise all the weights to be i.i.d $\mathcal{N}(0, 1)$ and optimise it using stochastic gradient descent. We study the limiting behavior of this network with respect to the width, that is, $h_1, \ldots, h_d \to \infty$.

**Remark 1** ($c_\sigma$) *While this setup is similar to Kipf and Welling [2017], it is important to note that we additionally consider the normalisation* $\sqrt{c_\sigma/h_i}$ *for layer* $i$ *to ensure that the input norm is approximately preserved. Here,* $c_\sigma$ *is a scaling factor to normalize the input in the initialization phase*

[1] More precisely, transductive setting as we assume all features are available during training at the same time.

*and* $c_\sigma = \left( \underset{u \sim \mathcal{N}(0,1)}{\mathbb{E}} \left[ (\sigma(u))^2 \right] \right)^{-1}$ *from Du et al. [2019a]. We discuss the role of this normalisation in Section 4.*

### 2.1 NTK for Vanilla GCN

We derive the NTK for vanilla GCN by first rewriting $F_W(X,S)$ as defined in (2) using the following recursive definitions:

$$g_1 := SX, \qquad g_i := \sqrt{\frac{c_\sigma}{h_{i-1}}} S\sigma(f_{i-1}) \, \forall i \in \{2, \dots, d+1\}, \qquad f_i := g_i W_i \, \forall i \in [d+1]$$

$$\text{Output:} \quad F_W(X,S) := \Phi(f_{d+1}), \quad \text{where } \Phi(x) := \frac{2}{1+\exp(-x)} - 1 \tag{3}$$

Using the definitions in (3), the gradient with respect to $W_i$ can be written as

$$\frac{\partial F_W(X,S)}{\partial W_i} := g_i^T b_i \quad \text{with} \quad b_{d+1} := \dot{\Phi}(f_{d+1}), \qquad b_i := \sqrt{\frac{c_\sigma}{h_i}} S^T b_{i+1} W_{i+1}^T \odot \dot{\sigma}(f_i) \tag{4}$$

We derive the NTK, as defined in (1), using the recursive definition of $F_W(X,S)$ in (3) and its derivative in (4). The following theorem defines the NTK between every pair of nodes, and the $n \times n$ NTK matrix can be computed at once, as shown below (proof in appendix).

**Theorem 1 (NTK for Vanilla GCN)** *For the vanilla GCN defined in* (2)*, the NTK* $\Theta$ *is given by*

$$\Theta = \left[ \sum_{i=1}^{d+1} \Sigma_i \odot \left(SS^T\right)^{\odot(d+1-i)} \odot \left( \bigodot_{j=i}^{d+1-i} \dot{E}_j \right) \right] \odot \underset{f \sim \mathcal{N}(0,\Sigma_d)}{\mathbb{E}} \left[ \dot{\Phi}\left(f\right) \dot{\Phi}\left(f\right)^T \right]. \tag{5}$$

*Here* $\Sigma_i \in \mathbb{R}^{n \times n}$ *is the co-variance between nodes of the layer* $f_i$*, and is given by* $\Sigma_1 := SXX^TS^T$*,* $\Sigma_i := SE_{i-1}S^T$ *with* $E_i := c_\sigma \underset{f \sim \mathcal{N}(0,\Sigma_i)}{\mathbb{E}} \left[ \sigma(f)\sigma(f)^T \right]$ *and* $\dot{E}_i := c_\sigma \underset{f \sim \mathcal{N}(0,\Sigma_i)}{\mathbb{E}} \left[ \dot{\sigma}(f)\dot{\sigma}(f)^T \right]$*.*

*Each entry of the expected matrix in* (5) *can be approximately computed as follows. For* $\Delta \in \mathbb{R}^{2\times 2}$*,*

$$\underset{(p,q) \sim \mathcal{N}(0,\Delta)}{\mathbb{E}} \left[ \dot{\Phi}\left(p\right) \dot{\Phi}\left(q\right) \right] = \frac{1}{4} - \frac{\Delta_{00} + \Delta_{11}}{16} + \frac{\Delta_{00}\Delta_{11} + 2\Delta_{01}^2}{64} + \frac{\Delta_{00}^2 + \Delta_{11}^2}{32} + \frac{\epsilon^3}{16}$$

*for* $|\epsilon| \leq \max\{\Delta_{00}, \Delta_{11}\}$*.*

**Inference using NTK.** The NTK matrix $\Theta \in \mathbb{R}^{n \times n}$ defines the pairwise kernel among all labeled and unlabeled nodes, where each entry $\Theta_{pq}$ represents the kernel between nodes (or features) $x_p$ and $x_q$. For inference, consider the sub-matrix $\Theta_l \in \mathbb{R}^{m \times m}$ that consists of the kernel computed between all pairs of labeled nodes, and $\Theta_u \in \mathbb{R}^{(n-m)\times m}$ that consists of the kernel computed between all pairs of unlabeled and labeled nodes. In the case of squared loss minimisation by stochastic gradient descent with infinitesimally small learning rate $\eta \to 0$, the training dynamics resemble kernel regression [Arora et al., 2019]. Hence, the labels for unlabeled nodes $Y_u$ can be inferred as

$$Y_u = \Theta_u \Theta_l^{-1} Y \ \in \mathbb{R}^{n-m} \tag{6}$$

which, when thresholded entry-wise at 0, yields the class prediction for unlabeled nodes.

The NTK derived in (5) holds for vanilla GCN with arbitrary activation function in (2). Since the focus of this work is explaining the empirical performance trends of GCNs, we focus on specific activation functions that fix the network architecture allowing the NTK to be evaluated exactly. We first consider a linear activation, that results in the SGC network [Wu et al., 2019], and derive the NTK as follows.

**Corollary 1 (Linear GCN)** *Consider* $\sigma(x) := x$ *in* $F_W(X,S)$*, then* $E_i = c_\sigma \Sigma_i$ *and* $\dot{E}_i = c_\sigma \mathbf{1}_{n\times n}$ *in Theorem 1, resulting in the following NTK*

$$\Theta = c_\sigma^d \left[ \sum_{i=1}^{d+1} \left( S^i XX^T \left(S^T\right)^i \right) \odot \left(SS^T\right)^{\odot(d+1-i)} \right] \odot \underset{f \sim \mathcal{N}(0,\Sigma_d)}{\mathbb{E}} \left[ \dot{\Phi}\left(f\right) \dot{\Phi}\left(f\right)^T \right].$$

*where the last expectation is approximated as in Theorem 1. The natural choice of normalisation constant* $c_\sigma$ *is* $c_\sigma = 1$ *based on Remark 1.*

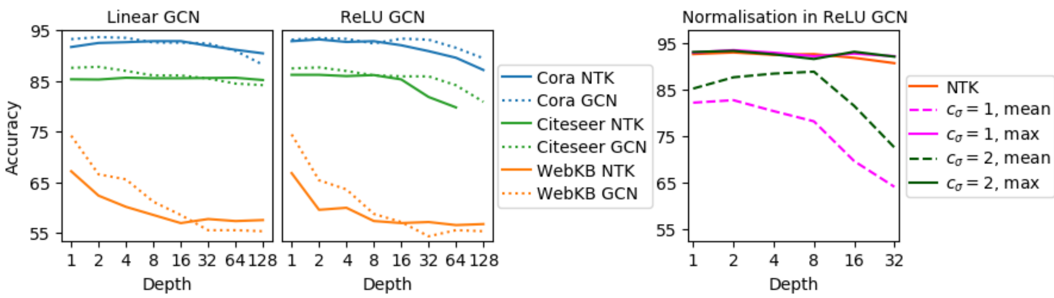


Figure 1: **(left/middle)** Performance of NTK vs GCN in linear and non-linear architectures. The performance trend of NTK matches the best performance of its corresponding GCN in both the architectures. **(right)** Impact of normalisation in ReLU GCN evaluated on Cora dataset. The correct choice of normalisation ($c_\sigma = 2$ in this case) stabilises the training of GCN even in higher depths, and enables identifying the hyper-parameters in lesser time compared to the unnormalised GCN.

Considering a non-linear network with ReLU activation, the NTK can be computed as shown below.

**Corollary 2 (ReLU GCN)** *Consider $\sigma(x) := ReLU(x)$ in $F_W(X, S)$. The NTK kernel is computed as in (5), where given $\Sigma_i$ at each layer, one can evaluate the entries of $E_i$ and $\dot{E}_i$ using a result from Bietti and Mairal [2019] as*

$$\left(E_i\right)_{pq} = \frac{c_\sigma}{2}\sqrt{\left(\Sigma_i\right)_{pp}\left(\Sigma_i\right)_{qq}}\,\kappa_1\left(\frac{\left(\Sigma_i\right)_{pq}}{\sqrt{\left(\Sigma_i\right)_{pp}\left(\Sigma_i\right)_{qq}}}\right) \text{ and } \left(\dot{E}_i\right)_{pq} = \frac{c_\sigma}{2}\kappa_0\left(\frac{\left(\Sigma_i\right)_{pq}}{\sqrt{\left(\Sigma_i\right)_{pp}\left(\Sigma_i\right)_{qq}}}\right), \tag{7}$$

*where $\kappa_0(x) := \frac{1}{\pi}\left(\pi - arccos\left(x\right)\right)$ and $\kappa_1(x) := \frac{1}{\pi}\left(x\left(\pi - arccos\left(x\right)\right) + \sqrt{1-x^2}\right)$. Based on Remark 1, the natural choice for normalisation constant $c_\sigma$ is $c_\sigma = 2$.*

## 2.2 Empirical Analysis of Depth

Many studies have shown that the performance of vanilla GCN drastically drops with depth due to the over smoothing effect of convolutional layers [Li et al., 2018, Kipf and Welling, 2017, Chen et al., 2020]. To validate it, we empirically study the performances of GCN and its NTK counterpart. We use Tesla K80 GPU with 12GB memory from Google Colab to obtain all our experimental results.

**Experimental Setup.** We evaluate the performances of linear and ReLU GCN as stated in Corollary 1 and 2, respectively, and their corresponding NTKs for different depths $d = \{1, 2, 4, 8, 16, 32, 64, 128\}$. We fix the size of hidden layers $h_i$ to be the same across all layers to reduce the number of hyper-parameters. We consider a range of learning rates $\eta = \{10^{-2}, 10^{-3}, 10^{-4}\}$, different size of the hidden layers $h_i = \{16, 64, 128, 256\}$ and report the best performance among the different $\eta$ and size $h_i$ over $10,000$ epochs. It is important to note that the chosen learning rates are in accordance to the theoretical analysis, that is, $\eta \to 0$. We conduct the experiments with three datasets, namely *Cora* [McCallum et al., 2000], *Citeseer* [Giles et al., 1998] and *WebKB* [Craven et al., 1998]. Since the datasets are for multi-class node classification, we combine the classes into two groups to fit our problem in focus – binary node classification. The choice of class grouping is decided by comparing the performances of different groupings and ensuring that the two groups are approximately equal sized. Appendix B includes detailed discussion on the datasets and grouping of the classes.

**NTK captures the performance trend of GCN.** The best performance of GCN decreases with depth in both linear and non-linear architectures, as observed in other papers. This trend in the best performance is also confirmed in NTK and thus making it a suitable method to analyse finite width GCN, despite the fact that the actual performance of the NTK is usually worse than the corresponding GCN. The left plot of Figure 1[2] shows the best performance of both the GCN architectures with

[2]NTK for Citeseer faced out-of-memory issue for depth $d = 128$ in some cases (can also be seen in Figure 2).

its NTK counterpart. While there is a drop in the best performance in both the GCNs and the corresponding NTKs, the drop is not as drastic as it has been reported in other papers. This is due to two factors: first, unlike the previous works that evaluated the performance for a fixed network parameterisation, we allow the size of hidden layers to be chosen as a hyper-parameter. We found that increasing the network size $h_i$ and/or decreasing the learning rate $\eta$ can reduce the performance drop with depth. For instance, in *Cora* the best performing network of depth 2 is achieved with $h_i = 16$ and $\eta = 10^{-2}$, whereas, $h_i$ has to be increased to $256$ and $\eta$ has to be reduced to $10^{-4}$ for depth $128$ to achieve similar performance. Second, we identify that the normalisation constant $c_\sigma$ plays a crucial role in stabilising the GCN training. The right plot of Figure 1 shows the average and the best performance of unnormalised ($c_\sigma = 1$) and correctly normalised ($c_\sigma = 2$) ReLU GCN for a fixed parameterisation with $h_i = 16$ and trained with learning rate $\eta = \{10^{-2}, 10^{-3}, 10^{-4}\}$ over $10,000$ epochs. While the average performance of both unnormalised and normalised GCNs shows a drastic drop, correct normalisation enables the network to learn faster and achieve best results. Further detailed discussion on the role of normalisation constant $c_\sigma$ is provided in Section 4.

# 3 NTK - Surrogate for GCN to Analyse Skip Connections

Skip connections [Chen et al., 2020, Kipf and Welling, 2017] are one way to overcome the performance degradation with depth in GCNs, but little is known about the effectiveness of different forms of available skip connections. Inspired by the observation of the previous section that the NTK is a hyper-parameter free model that captures the trends of GCNs, we propose NTK as an efficient surrogate for GCN, and we investigate different skip connections for GCN in detail in this section. We consider two formulations of skip connections with two variants each that are described in subsequent sections. To facilitate skip connections, we need to enforce constant layer size, that is, $h_i = h_{i-1}$. Therefore, we transform the input layer to $H_0$ of size $n \times h$ where $h$ is the hidden layer size. This transformation is necessary as otherwise we would have to assume $h_i = f \ \forall i \in [d]$ and $h_i \to \infty$ would not be possible. For this work, we do not consider this transformation as a learnable parameter in the network. As we consider constant layer size, the NTKs are derived considering $h \to \infty$. We first define a skip connection related to the one in Kipf and Welling [2017], where the skip connection is added to the features before convolution (we refer to it as pre-convolution or Skip-PC).

**Definition 1 (Skip-PC)** *In a Skip-PC (pre-convolution) network, the transformed input $H_0$ is added to the hidden layers before applying the diffusion, leading to the changes in the recursive definition of* (3) *with $g_1 := SH_0$ and*

$$g_i := \sqrt{\frac{c_\sigma}{h}} S \left( \sigma \left( f_{i-1} \right) + \sigma_s \left( H_0 \right) \right) \ \forall i \in \{2, \ldots, d+1\}, \quad f_i := g_i W_i \ \forall i \in [d+1] \tag{8}$$

*where $\sigma_s(.)$ can be linear or ReLU accounting for two different skip connections.*

We refer to the network with linear $\sigma_s(.)$ and ReLU $\sigma_s(.)$ as Linear Skip-PC and ReLU Skip-PC, respectively. The above definition deviates from Kipf and Welling [2017] in the fact that we skip to the input layer instead of the previous layer. This particular change helps in evaluating the importance of graph information in a dataset which we discuss in the following section. We also consider a skip connection similar to the one described in Chen et al. [2020].

**Definition 2 (Skip-$\alpha$)** *Given an interpolation coefficient $\alpha \in (0, 1)$ and a function $\sigma_s(\cdot)$, a Skip-$\alpha$ network is defined such that the transformed input $H_0$ and the hidden layer are interpolated linearly, which changes the recursive definition in* (3) *as $g_1 := SH_0$ and*

$$g_i := \sqrt{\frac{c_\sigma}{h}} \left( (1-\alpha) S \sigma \left( f_{i-1} \right) + \alpha \, \sigma_s \left( H_0 \right) \right) \ \forall i \in \{2, \ldots, d+1\}, \quad f_i := g_i W_i \ \forall i \in [d+1] \tag{9}$$

Similar to Skip-PC, $\sigma_s(.)$ can be linear or ReLU accounting for two different skip connections. We refer to the network with linear $\sigma_s(.)$ and ReLU $\sigma_s(.)$ as Linear Skip-$\alpha$ and ReLU Skip-$\alpha$, respectively. Chen et al. [2020] recommends the choices for $\alpha$ as $0.1$ or $0.2$.

**Remark 2 (Change of the normalization factor $c_\sigma$ due to Skip connections)** *Note that the normalisation constant $c_\sigma$ for GCN with skip connections is not the same as defined in Remark 1 of vanilla GCN, since we add the transformed input to the hidden layers. Intuitively, $c_\sigma < 1$ as the norm of the hidden layers would increase otherwise due to the added term. We derived $c_\sigma$ specifically for non-linear GCN with $\sigma(x) :=$ ReLU$(x)$, and it is $\simeq 0.67$. Refer to Appendix A for the proof.*

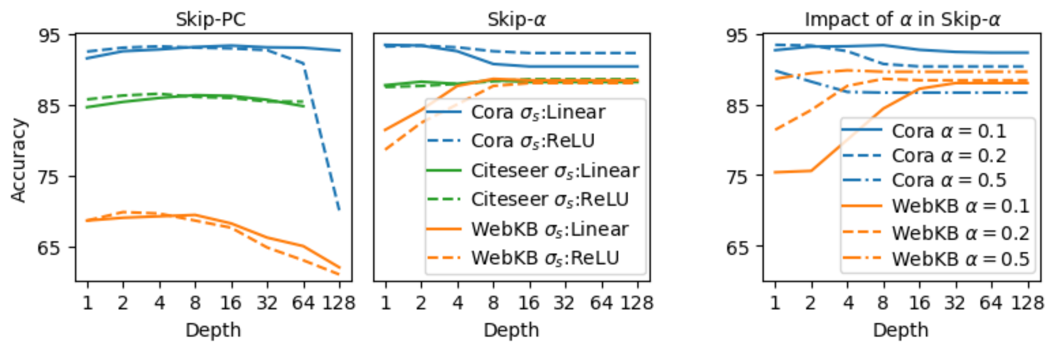


Figure 2: **(left/middle)** Performance of NTKs corresponding to the different skip connections where Skip-$\alpha$ is plotted for $\alpha = 0.2$. **(right)** Impact of $\alpha$ in Skip-$\alpha$ evaluated on Cora and WebKB datasets.

### 3.1 NTK for GCN with Skip Connections

We derive NTKs for the skip connections – Skip-PC and Skip-$\alpha$. Both the NTKs maintain the form presented in Theorem 1 with the following changes to the co-variance matrices.

**Corollary 3 (NTK for Skip-PC)** *The NTK for an infinitely wide Skip-PC network is as presented in Theorem 1 where $E_i$ is defined as in the theorem, but $\Sigma_i$ is defined as*

$$\Sigma_0 := XX^T, \qquad \Sigma_1 := SE_0S^T \qquad \textit{and} \qquad \Sigma_i := SE_{i-1}S^T + \Sigma_1. \tag{10}$$

**Corollary 4 (NTK for Skip-$\alpha$)** *The NTK for an infinitely wide Skip-$\alpha$ network is as presented in Theorem 1 where $E_i$ is defined as in the theorem, but $\Sigma_i$ is defined with $\Sigma_0 := XX^T$,*

$$\Sigma_1 := (1-\alpha)^2\, SE_0S^T + \alpha\,(1-\alpha)\left(SE_0 + E_0S^T\right) + \alpha^2 E_0 \ \textit{ and } \ \Sigma_i := SE_{i-1}S^T + E_0. \tag{11}$$

Both Corollary 1 and 2 for linear and ReLU activations, respectively, hold for the derived NTKs corresponding to Skip-PC and Skip-$\alpha$.

### 3.2 Empirical Analysis

Despite studies [Chen et al., 2020, Kipf and Welling, 2017] showing that having skip-connections gives a significant performance advantage, there is no clear way to choose one formulation of the skip connection over others. This practical problem can again be seen in the NTK setting as the derived NTKs have similar structure except the co-variance between the nodes, thus making it difficult to compare analytically. Therefore, we empirically study the performance of different NTKs in order to determine the preferred formulation, thereby avoiding computational intensive hyper-parameter tuning. In addition, we show that the NTK corresponding to Skip-$\alpha$ can be used for assessing the relevance of structure and feature information of graph in a dataset. We study the non-linear ReLU GCN with the discussed skip connections, that is, $\sigma(.) :=$ ReLU in (2) empirically.

**Experimental setup.** We evaluate the performance of NTKs corresponding to GCNs with skip connections for different depths $d = \{1, 2, 4, 8, 16, 32, 64, 128\}$ using non-linear activation $\sigma(x) :=$ ReLU$(x)$ for the GCNs. The linear transformation of the input $X$ is done by $H_0 = XT$ where $T$ is a $f \times h$ matrix and each entry is sampled from $\mathcal{N}(0, 1)$. The interpolation coefficient $\alpha$ in Skip-$\alpha$ is chosen to be $\{0.1, 0.2, 0.5\}$. NTKs for all the formulations of skip connections discussed in the previous section are evaluated on different datasets, namely *Cora*, *Citeseer* and *WebKB*. Figure 2 shows the empirical observations. Refer to Appendix B for more details.

We validate the expected performance advantage of GCN with skip connections over vanilla GCN, more precisely their NTK counterparts, and observe the following main findings.

**Non-linear $\sigma_s$ and shallow net for GCNs with skip connection.** Empirical analysis reveals a distinct behavior of skip connections with $\sigma_s(.)$ being linear and ReLU, which is illustrated in the left plot of Figure 2. We observe that the performance of both Skip-PC and Skip-$\alpha$ is not optimal at deeper depths, and hence we restrict our focus to shallow depths. In the case of shallow depths, we find that using non-linear ReLU $\sigma_s(.)$ in both Skip-PC and Skip-$\alpha$ produces the best performance. Although ReLU Skip-$\alpha$ initially falls short of its counterpart Linear Skip-$\alpha$, it eventually outperforms or performs as good as Linear Skip-$\alpha$, thus favoring ReLU $\sigma_s(.)$. In addition, this experiment also validates the general practice of using shallow nets for GCNs. Consequently, we propose skip connections with ReLU $\sigma_s(.)$ and using shallow nets to achieve the best performance in practice.

**NTK as a model to assess relevance of structure and feature information of graphs.** In the left plot of Figure 2, we notice that the performance of Skip-$\alpha$ on WebKB improved significantly as compared to Skip-PC and moreover, its performance continued to improve with depth, which is in contrast to other datasets. We further investigate this by analysing the interpolation coefficient $\alpha$, and the corresponding results on Cora and WebKB datasets are shown in the right plot of Figure 2. Large value of $\alpha$ in Skip-$\alpha$ implies that more importance is given to feature information than the structural information of the graph. Therefore, from the figure, we infer that the structural information is not as important as the feature information for WebKB which is in contrast to Cora. Besides, NTK is a ready-to-use model without the need for hyper-parameter tuning. As a result, we propose NTK corresponding to Skip-$\alpha$ as a stand-alone model to determine the relative importance of structure and feature information in tasks where GCNs are employed.

# 4 Role of Normalisation in GCN

In Section 2, we discussed that the performance drop with depth in vanilla GCN can be reduced by varying the size of the hidden layers $h_i$ rather than fixing the network parameterisation as done in other works. The main difference between our theoretical setup for infinite width GCN and the practical finite width GCN is the normalisation $\sqrt{c_\sigma/h_i}$. Practical networks generally ignore the normalisation factor and rely on weight initialisation and optimisation algorithm to stabilise the training. Intrigued by this, we investigate the role of normalisation applied to each layer by fixing the network parameterisation in vanilla GCN and Skip-PC empirically. Figure 3 illustrates this for different $c_\sigma = \{0.67, 1, 2\}$ and depths $d = \{8, 16, 32\}$ on *Cora* dataset. The considered architectures have non-linear activation, that is, $\sigma(x) := \mathrm{ReLU}(x)$ and we fix the network parameterisation in both the cases. Different colors in the plot represent the epoch at which the performance is achieved. The correct choice of $c_\sigma$ is 2 for ReLU in vanilla GCN (Corollary 2) and $0.67$ for Skip-PC (Remark 2).

We make the following observation. In the case of vanilla GCN, it is clear that the best performance is achieved in almost the same number of epochs across all the depths for the correct choice of $c_\sigma = 2$. Moreover, the decrease in the performance for deeper networks is not significant. Also we need to train the network longer for $c_\sigma = 1$ to achieve similar performance of the network with correct normalisation ($c_\sigma = 2$) as we increase the depth. Thus, normalisation plays a crucial role in stabilising the training of vanilla GCN especially in higher depths. In Skip-PC, the performance of the network is not significantly affected by $c_\sigma$. This is because the residual connection ensures that the hidden layer norm is approximately equal to the input norm, and thus $c_\sigma$ is not as relevant as it is in vanilla GCN case. Therefore, in practice, the absence of this normalisation in vanilla GCN explains the reported drastic degradation in performance with depth in the existing literature.

# 5 Convergence of NTK with depth

In Figure 2, we observe that the performance of NTKs corresponding to GCNs with skip connections does not change significantly beyond a certain depth. We investigate this behaviour of the NTK further by measuring the amount of change between NTKs of different depths. To this end, we consider the alignment between the NTKs in the eigenspace following Fowlkes et al. [2004, Section 4.2]. Formally, let $\Theta_i$ and $\Theta_j$ be the NTK of depth $i$ and $j$, respectively, and $U_i^{(k)}$ and $U_j^{(k)}$ be the matrix of $k$ leading eigenvectors of $\Theta_i$ and $\Theta_j$, respectively, then the alignment between $\Theta_i$ and $\Theta_j$ is computed by $a = \frac{1}{k}\left\|{U_i^{(k)}}^T U_j^{(k)}\right\|_F^2$, where $a \in [0,1]$ with $a = 1$ indicating perfect alignment. Figure 4 shows the alignment of the NTKs for the discussed non-linear ReLU architectures ($\sigma(.) := \mathrm{ReLU}$ in (2)), evaluated on *Cora* dataset. Similar pattern is observed in other datasets as well (Appendix B).

**The learning happens in shallow depth.** The different alignment plots illustrate the general influence of depth in GCN. We observe significant changes in the alignment between NTKs of shallow depths indicating that this is the important part where learning happens. Since the NTKs for both

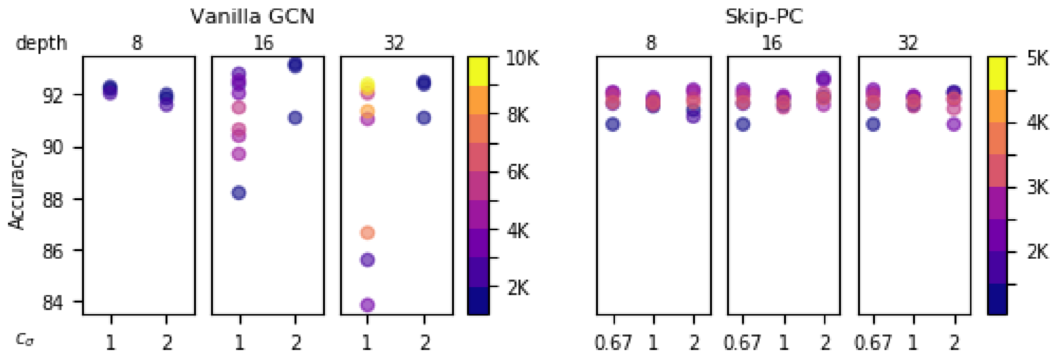


Figure 3: Role of normalisation $c_\sigma$ in vanilla GCN and Skip-PC as defined in 8. The colorbar represents the number of epochs. The correct choice of $c_\sigma$ stabilises the training of GCN even in higher depths in vanilla GCN.

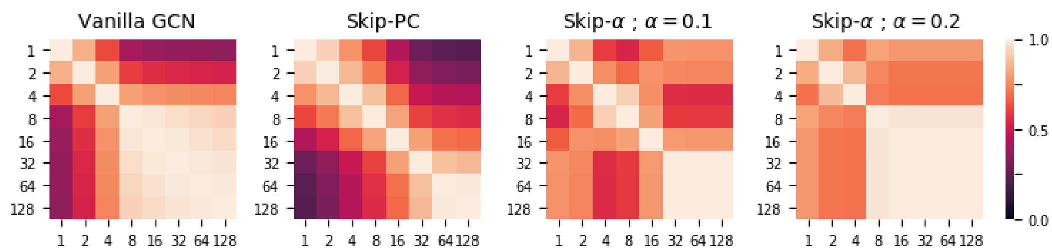


Figure 4: Convergence of NTK with depth for all the discussed ReLU architectures, evaluated on Cora dataset. The plots show perfect alignment of NTKs for higher depths in GCNs with skip connections.

vanilla GCN and GCN with skip connections converge with depth, it is clear that deep GCNs have no advantage or in other words, no new information is learned at deeper depths.

**Influence of Skip connection.** In addition, we observe that the NTKs reach almost perfect alignment with depth for GCNs with skip connection, suggesting that the networks reached saturation in learning as well. We can further distinguish the presented skip connections: overall Skip-PC has slow convergence most likely because the skip connection facilitates learning; Skip-$\alpha$ converges fast and as discussed in Section 3, we observe the influence of $\alpha$ in the learning depending on the dataset.

## 6 Conclusion

In this work, we derive NTKs for semi-supervised GCNs, including different formulations of skip connections. The deterministic hyper-parameter free nature of NTK makes it preferable over its neural network counterpart since it captures the behaviour of the networks very well, as demonstrated in our experiments. With the support of our empirical results and the findings from Du et al. [2019b] that the NTK for supervised GCN outperforms the neural network, we expect the NTKs for semi-supervised models to perform competitively against the respective neural networks. Nonetheless, the primary goal of our work is to use NTK to advance our understanding of GCN, particularly on the impact of depth. In addition, we suggest NTK as a surrogate to study variants of GCNs. From our surrogate analysis, we propose the NTK corresponding to the skip connection Skip-$\alpha$ as an efficient ready-to-use off-the-shelf model to determine the relative importance of structure and feature information in graphs, which we believe to be of great practical value. There is a possibility of expanding the usage of NTK surrogates to analyse robustness or explainability of GCNs, or other contexts that involve repeated training of networks. Another direction of research is to incorporate practical considerations of network architecture in the NTK derivation. The present paper allows sigmoid functions in the output layer, which is included through a Taylor expansion. It would be also interesting to derive NTKs considering approximations for softmax, max-pooling, dropout or batch normalisation, and use the NTKs to analyse the impact of these techniques on network performance.

# References


Sina Alemohammad, Zichao Wang, Randall Balestriero, and Richard Baraniuk. The recurrent neural tangent kernel. In *International Conference on Learning Representations*, 2021.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In *Annual Conference on Neural Information Processing Systems, NeurIPS*, 2019.

Alberto Bietti and Julien Mairal. On the inductive bias of neural tangent kernels. In *NeurIPS 2019-Thirty-third Conference on Neural Information Processing Systems*, volume 32, pages 12873–12884, 2019.

Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: Fast learning with graph convolutional networks via importance sampling. In *International Conference on Learning Representations*, 2018a.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, pages 1725–1735. PMLR, 2020.

Zhengdao Chen, Lisha Li, and Joan Bruna. Supervised community detection with line graph neural networks. In *International Conference on Learning Representations*, 2018b.

Mark Craven, Andrew McCallum, Dan PiPasquo, Tom Mitchell, and Dayne Freitag. Learning to extract symbolic knowledge from the world wide web. Technical report, Carnegie-mellon univ pittsburgh pa school of computer Science, 1998.

Hanjun Dai, Bo Dai, and Le Song. Discriminative embeddings of latent variable models for structured data. In *International conference on machine learning*, pages 2702–2711. PMLR, 2016.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, 2016.

Pedro Domingos. Every model learned by gradient descent is approximately a kernel machine. *arXiv preprint arXiv:2012.00152*, 2020.

Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *International Conference on Machine Learning*, pages 1675–1685. PMLR, 2019a.

Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2018.

Simon S Du, Kangcheng Hou, Barnabás Póczos, Ruslan Salakhutdinov, Ruosong Wang, and Keyulu Xu. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019b.

Charless Fowlkes, Serge Belongie, Fan Chung, and Jitendra Malik. Spectral grouping using the nystrom method. *IEEE transactions on pattern analysis and machine intelligence*, 26(2):214–225, 2004.

Vikas Garg, Stefanie Jegelka, and Tommi Jaakkola. Generalization and representational limits of graph neural networks. In *International Conference on Machine Learning*, pages 3419–3430. PMLR, 2020.

C Lee Giles, Kurt D Bollacker, and Steve Lawrence. Citeseer: An automatic citation indexing system. In *Proceedings of the third ACM conference on Digital libraries*, pages 89–98, 1998.

William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035, 2017.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: convergence and generalization in neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 8580–8589, 2018.

Tatsuro Kawamoto, Masashi Tsubaki, and Tomoyuki Obuchi. Mean-field theory of graph neural networks in graph partitioning. *Journal of Statistical Mechanics: Theory and Experiment*, 2019 (12):124007, dec 2019. doi: 10.1088/1742-5468/ab3456.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes. In *International Conference on Learning Representations*, 2018.

Jaehoon Lee, Lechao Xiao, Samuel S Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *NeurIPS*, 2019.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. In *International Conference on Learning Representations (ICLR)*, 2016.

Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000.

Behnam Neyshabur, Srinadh Bhojanapalli, David Mcallester, and Nati Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

Franco Scarselli, Ah Chung Tsoi, and Markus Hagenbuchner. The vapnik–chervonenkis dimension of graph and recursive neural networks. *Neural Networks*, 108:248 – 259, 2018.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *stat*, 1050:4, 2018.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020a.

Pengfei Zhou, Tianyi Li, and Pan Zhang. Phase transitions and optimal algorithms for semisupervised classifications on graphs: From belief propagation to graph convolution network. *Physical Review Research*, 2(3):033325, 2020b.

# Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] Since our work is theoretical and attempts to explain the existing models.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 2.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Included in the supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Explained in each experiment section and in appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] NTK is deterministic and the seed for GCN is fixed to ensure that the results are reproducible.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Included GPU specifications.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

# A Proofs of NTKs for GCN and GCN with Skip Connections

We provide proofs of Theorem 1 and all corollaries with additional empirical results in this section.

## A.1 Proof of NTK for Vanilla GCN (Theorem 1)

**Co-variance between Nodes.** We will first derive the co-variance matrix of size $n \times n$ for each layer comprising of co-variance between any two nodes $p$ and $q$. The co-variance between $p$ and $q$ in $f_1$ and $f_i$ are derived below. We denote $p$-th row of matrix $M$ as $M_{p.}$ throughout our proofs.

$$\mathbb{E}\left[(f_1)_{pk}(f_1)_{qk'}\right] = \mathbb{E}\left[(g_1W_1)_{pk}(g_1W_1)_{qk'}\right]$$
$$= \mathbb{E}\left[\sum_{r=1}^{h_0}(g_1)_{pr}(W_1)_{rk}\sum_{s=1}^{h_0}(g_1)_{qs}(W_1)_{sk'}\right] \overset{(W_1)_{xy}\sim\mathcal{N}(0,1)}{=} 0 \quad ; \text{if } r \neq s \text{ or } k \neq k'$$
$$\mathbb{E}\left[(f_1)_{pk}(f_1)_{qk}\right] \underset{k=k'}{\overset{r=s}{=}} \mathbb{E}\left[\sum_{r=1}^{h_0}(g_1)_{pr}(g_1)_{qr}(W_1)^2_{rk}\right]$$
$$\overset{(W_1)_{xy}\sim\mathcal{N}(0,1)}{=} \sum_{r=1}^{h_0}(g_1)_{pr}(g_1)_{qr} = \left\langle (g_1)_{p.}, (g_1)_{q.}\right\rangle \tag{12}$$
$$\mathbb{E}\left[(f_i)_{pk}(f_i)_{qk}\right] \underset{k=k'}{\overset{r=s}{=}} \mathbb{E}\left[\sum_{r=1}^{h_{i-1}}(g_i)_{pr}(g_i)_{qr}(W_i)^2_{rk}\right]$$
$$\overset{(W_i)_{xy}\sim\mathcal{N}(0,1)}{=} \sum_{r=1}^{h_{i-1}}(g_i)_{pr}(g_i)_{qr} = \left\langle (g_i)_{p.}, (g_i)_{q.}\right\rangle \tag{13}$$

$$(12): \quad \left\langle (g_1)_{p.}, (g_1)_{q.}\right\rangle = \left\langle (SX)_{p.}, (SX)_{q.}\right\rangle = S_{p.}XX^TS^T_{.q} = (\Sigma_1)_{pq} \tag{14}$$
$$(13): \quad \left\langle (g_i)_{p.}, (g_i)_{q.}\right\rangle = \frac{c_\sigma}{h_{i-1}}\left\langle (S\sigma(f_{i-1}))_{p.}, (S\sigma(f_{i-1}))_{q.}\right\rangle$$
$$= \frac{c_\sigma}{h_{i-1}}\sum_{k=1}^{h_{i-1}}(S\sigma(f_{i-1}))_{pk}(S\sigma(f_{i-1}))_{qk}$$
$$\overset{h_{i-1}\to\infty}{=} c_\sigma\mathbb{E}\left[(S\sigma(f_{i-1}))_{pk}(S\sigma(f_{i-1}))_{qk}\right] \qquad \text{; law of large numbers}$$
$$= c_\sigma\mathbb{E}\left[\left(\sum_{r=1}^{n}S_{pr}\sigma\left(f_{i-1}\right)_{rk}\right)\left(\sum_{s=1}^{n}S_{qs}\sigma\left(f_{i-1}\right)_{sk}\right)\right]$$
$$= c_\sigma\mathbb{E}\left[\sum_{r=1}^{n}\sum_{s=1}^{n}S_{pr}S_{qs}\sigma\left(f_{i-1}\right)_{rk}\sigma\left(f_{i-1}\right)_{sk}\right]$$
$$\overset{(a)}{=} \sum_{r=1}^{n}\sum_{s=1}^{n}S_{pr}\left(E_{i-1}\right)_{rs}S^T_{sq} = S_{p.}E_{i-1}S^T_{.q} = (\Sigma_i)_{pq} \tag{15}$$

$(a)$: using $\mathbb{E}\left[(f_{i-1})_{rk}(f_{i-1})_{sk}\right] = (\Sigma_{i-1})_{rs}$ and the definition of $E_{i-1}$ in Theorem 1.

**NTK for Vanilla GCN.** Let us first evaluate the tangent kernel component from $W_i$ respective to nodes $p$ and $q$. The following two results are needed to derive it.

**Result 1 (Inner Product of Matrices).** Let $a$ and $b$ be vectors of size $d_1 \times 1$ and $d_2 \times 1$, then

$$\left\langle ab^T, ab^T\right\rangle = \operatorname{tr}\left(ab^T\left(ab^T\right)^T\right)$$
$$= \operatorname{tr}\left(ab^Tba^T\right) = \operatorname{tr}\left(a^Tab^Tb\right) = \left(a^Ta\right)\odot\left(b^Tb\right) = \langle a,a\rangle\odot\langle b,b\rangle \tag{16}$$

**Result 2** $\left\langle (b_r)_{p.}, (b_r)_{q.} \right\rangle$**.** We evaluate $\left\langle (b_r)_{p.}, (b_r)_{q.} \right\rangle = \left(b_r b_r^T\right)_{pq}$ which appears in the gradient.

$$
\begin{aligned}
\left(b_r b_r^T\right)_{pq} &= \frac{c_\sigma}{h_r} \sum_{k=1}^{h_r} \left(S^T b_{r+1} W_{r+1}^T\right)_{pk} \dot{\sigma}(f_r)_{pk} \left(S^T b_{r+1} W_{r+1}^T\right)_{qk} \dot{\sigma}(f_r)_{qk} \\
&= \frac{c_\sigma}{h_r} \sum_{k=1}^{h_r} \sum_{i,j}^{n,h_{r+1}} S_{ip} \left(b_{r+1}\right)_{ij} \left(W_{r+1}\right)_{kj} \dot{\sigma}(f_r)_{pk} \dot{\sigma}(f_r)_{qk} \sum_{i',j'}^{n,h_{r+1}} S_{i'q} \left(b_{r+1}\right)_{i'j'} \left(W_{r+1}\right)_{kj'} \\
&= \frac{c_\sigma}{h_r} \sum_{i,j}^{n,h_{r+1}} \sum_{i',j'}^{n,h_{r+1}} \left(b_{r+1}\right)_{ij} \left(b_{r+1}\right)_{i'j'} S_{ip} S_{i'q} \sum_{k=1}^{h_r} \left(W_{r+1}\right)_{kj} \dot{\sigma}(f_r)_{pk} \dot{\sigma}(f_r)_{qk} \left(W_{r+1}\right)_{kj'} \\
&= \sum_{j,j'}^{h_{r+1},h_{r+1}} \left(S^T b_{r+1}\right)_{pj} \left(S^T b_{r+1}\right)_{qj'} \frac{c_\sigma}{h_r} \sum_{k=1}^{h_r} \left(W_{r+1}\right)_{kj} \dot{\sigma}(f_r)_{pk} \dot{\sigma}(f_r)_{qk} \left(W_{r+1}\right)_{kj'} \\
&\overset{h_r \to \infty}{=} \sum_{j}^{h_{r+1}} \left(S^T b_{r+1}\right)_{pj} \left(S^T b_{r+1}\right)_{qj} c_\sigma \mathbb{E}\left[\left(W_{r+1}^2\right)_{kj} \dot{\sigma}(f_r)_{pk} \dot{\sigma}(f_r)_{qk}\right] \qquad ; 0 \text{ for } j \neq j' \\
&\overset{(b)}{=} \left\langle \left(S^T b_{r+1}\right)_{p.} \left(S^T b_{r+1}\right)_{q.} \right\rangle c_\sigma \mathbb{E}\left[\dot{\sigma}(f_r)_{pk} \dot{\sigma}(f_r)_{qk}\right] \\
&\overset{(16)}{=} \left(SS^T\right)_{pq} \left\langle b_{r+1}, b_{r+1} \right\rangle_{pq} c_\sigma \mathbb{E}\left[\dot{\sigma}(f_r)_{pk} \dot{\sigma}(f_r)_{qk}\right] \\
&= \left(SS^T\right)_{pq} \left\langle b_{r+1}, b_{r+1} \right\rangle_{pq} \left(\dot{E}_r\right)_{pq}
\end{aligned}
\tag{17}
$$

$(b)$: $\left(W_{r+1}\right)_{kj}$ is independent and $\mathbb{E}\left[\left(W_{r+1}^2\right)_{kj} = 1\right]$.

Now, lets derive $\left\langle \left(\frac{\partial F}{\partial W_i}\right)_p, \left(\frac{\partial F}{\partial W_i}\right)_q \right\rangle$ and $\left\langle \left(\frac{\partial F}{\partial W_1}\right)_p, \left(\frac{\partial F}{\partial W_1}\right)_q \right\rangle$ using the above results.

$$
\begin{aligned}
\left\langle \left(\frac{\partial F}{\partial W_i}\right)_p, \left(\frac{\partial F}{\partial W_i}\right)_q \right\rangle &= \left\langle (g_i)_{p.}^T (b_i)_{p.}, (g_i)_{q.}^T (b_i)_{q.} \right\rangle \\
&\overset{(16)}{=} \left\langle (g_i)_{p.}, (g_i)_{q.} \right\rangle \odot \left\langle (b_i)_{p.}, (b_i)_{q.} \right\rangle \\
&\overset{(15),(17)}{=} \left(\Sigma_i\right)_{pq} \left(SS^T\right)_{pq} \left\langle b_{r+1}, b_{r+1} \right\rangle_{pq} \left(\dot{E}_r\right)_{pq} \\
&\overset{(c)}{=} \left(\Sigma_i\right)_{pq} \left(\left(SS^T\right)_{pq}\right)^{d+1-i} \left(\prod_{j=i}^{d+1-i} \left(\dot{E}_j\right)_{pq}\right) \left\langle b_{d+1}, b_{d+1} \right\rangle_{pq} \\
&\overset{(d)}{=} \left(\Sigma_i\right)_{pq} \left(\left(SS^T\right)_{pq}\right)^{d+1-i} \left(\prod_{j=i}^{d+1-i} \left(\dot{E}_j\right)_{pq}\right) \left(\dot{\Phi}\left(f_{d+1}\right) \dot{\Phi}\left(f_{d+1}\right)^T\right)_{pq}
\end{aligned}
\tag{18}
$$

$(c)$: repeated application of (17).
$(b)$: definition of $b_{d+1}$.

Extending (18) to all $n$ nodes which will result in $n \times n$ matrix,

$$
\begin{aligned}
\left\langle \frac{\partial F}{\partial W_i}, \frac{\partial F}{\partial W_i} \right\rangle &= \Sigma_i \odot \left(SS^T\right)^{\odot d+1-i} \bigodot_{j=i}^{d+1-i} \dot{E}_j \odot \dot{\Phi}\left(f_{d+1}\right) \dot{\Phi}\left(f_{d+1}\right)^T \\
\mathop{\mathbb{E}}_{W_i}\left[\left\langle \frac{\partial F}{\partial W_i}, \frac{\partial F}{\partial W_i} \right\rangle\right] &= \Sigma_i \odot \left(SS^T\right)^{\odot d+1-i} \bigodot_{j=i}^{d+1-i} \dot{E}_j \odot \mathop{\mathbb{E}}_{f \sim \mathcal{N}(0,\Sigma_d)}\left[\dot{\Phi}(f) \dot{\Phi}(f)^T\right]
\end{aligned}
\tag{19}
$$

Finally, NTK $\Theta$ is,

$$\begin{aligned}\Theta &= \sum_{i=1}^{d+1} \mathop{\mathbb{E}}_{W_i}\left[\left\langle \frac{\partial F}{\partial W_i}, \frac{\partial F}{\partial W_i}\right\rangle\right] \\ &= \left[\sum_{i=1}^{d+1} \Sigma_i \odot \left(SS^T\right)^{\odot(d+1-i)} \odot \left(\bigodot_{j=i}^{d+1-i} \dot{E}_j\right)\right] \odot \mathop{\mathbb{E}}_{f\sim\mathcal{N}(0,\Sigma_d)}\left[\dot{\Phi}\left(f\right)\dot{\Phi}\left(f\right)^T\right] \end{aligned} \tag{20}$$

We will now compute $\mathop{\mathbb{E}}_{f\sim\mathcal{N}(0,\Sigma_d)}\left[\dot{\Phi}\left(f\right)\dot{\Phi}\left(f\right)^T\right]$. We use Lagrange form of the remainder to approximate the Taylor's expansion for the re-scaled sigmoid function $\Phi(.)$ which gives better bound.

$$\begin{aligned}\Phi(x) &= \frac{2}{1+\exp^{-x}} - 1 = \frac{x}{2} - \frac{x^3}{24} + \frac{x^5}{240} + \cdots \\ \dot{\Phi}(x) &= \frac{1}{2} - \frac{x^2}{8} + \frac{x^4}{48} + \frac{x^6\dot{\Phi}^6(\xi)}{6!} \qquad \text{; last term is the Lagrange form of the remainder.}\end{aligned} \tag{21}$$

To evaluate the expectation of an entry $i,j$ in the matrix $\dot{\Phi}\left(f\right)\dot{\Phi}\left(f\right)^T$, let us define $\Delta$ as a $2\times 2$ co-variance matrix as follows, $\Delta = \begin{bmatrix}\left(\Sigma_{d+1}\right)_{ii} & \left(\Sigma_{d+1}\right)_{ij} \\ \left(\Sigma_{d+1}\right)_{ji} & \left(\Sigma_{d+1}\right)_{jj}\end{bmatrix}$

$$\begin{aligned}\mathop{\mathbb{E}}_{(x,y)\sim\Delta}\left[\dot{\Phi}\left(x\right)\dot{\Phi}\left(y\right)\right] &\overset{(21)}{=} \mathop{\mathbb{E}}_{(x,y)\sim\Delta}\left[\left(\frac{1}{2} - \frac{x^2}{8} + \frac{x^4}{48} + \frac{x^6\dot{\Phi}^6(\xi)}{6!}\right)\left(\frac{1}{2} - \frac{y^2}{8} + \frac{y^4}{48} + \frac{y^6\dot{\Phi}^6(\xi)}{6!}\right)\right] \\ &= \frac{1}{4}\mathop{\mathbb{E}}_{(x,y)\sim\Delta}\Big[1 - \frac{x^2}{4} - \frac{y^2}{4} + \frac{x^4}{24} + \frac{y^4}{24} + \frac{x^2y^2}{16} - \frac{x^4y^2}{96} - \frac{x^2y^4}{96} \\ &\qquad + \frac{x^4y^4}{576} + \frac{x^6\dot{\Phi}^6(\xi)}{6!}\left(\frac{1}{2} - \frac{y^2}{8} + \frac{y^4}{48} + \frac{y^6\dot{\Phi}^6(\xi)}{6!}\right)\Big]\end{aligned} \tag{22}$$

**Compute $\mathop{\mathbb{E}}_{x\sim\mathcal{N}(0,\lambda^2)}\left[x^k\right]$ and $\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{N}(0,\Delta)}\left[x^iy^j\right]$.**

$$\begin{aligned}\mathop{\mathbb{E}}_{x\sim\mathcal{N}(0,\lambda^2)}\left[x^k\right] &= \frac{2}{\sqrt{2\pi}\lambda}\int_0^\infty x^k \exp\left(\frac{-x^2}{2\lambda^2}\right)\mathrm{d}x \\ &= \frac{2\lambda^k}{\sqrt{2\pi}}\int_0^\infty t^k\exp\left(\frac{-t^2}{2}\right)\mathrm{d}t \qquad ; x = \lambda t \implies \mathrm{d}x = \lambda \mathrm{d}t \\ &= \frac{2\lambda^k}{\sqrt{2\pi}}(k-1)\int_0^\infty t^{k-2}\exp\left(\frac{-t^2}{2}\right)\mathrm{d}t \\ \text{Thus, } \mathop{\mathbb{E}}_{x\sim\mathcal{N}(0,\lambda^2)}\left[x^k\right] &= (k-1)\lambda^2 \mathop{\mathbb{E}}_{x\sim\mathcal{N}(0,\lambda^2)}\left[x^{k-2}\right]\end{aligned} \tag{23}$$

$$\begin{aligned}\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{N}(0,\Delta)}\left[x^iy^j\right] &= \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{N}(0,\Delta)}\left[x^i\left(y\pm\alpha x\right)^j\right] \qquad ; \alpha = \frac{\mathbb{E}\left[xy\right]}{\mathbb{E}\left[x^2\right]} \text{ then } x, y-\alpha x \text{ are independent} \\ &= \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{N}(0,\Delta)}\left[x^i\left(\sum_{k=0}^{j} {}^{j}C_k\left(y-\alpha x\right)^j\left(\alpha x\right)^k\right)\right] \\ &= \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{N}(0,\Delta)}\left[\sum_{k=0}^{j} {}^{j}C_k\alpha^k\left(y-\alpha x\right)^j x^{k+i}\right] \\ &\overset{(e)}{=} \sum_{k=0}^{j} {}^{j}C_k\alpha^k \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{N}(0,\Delta)}\left[x^{k+i}\right]\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{N}(0,\Delta)}\left[\left(y-\alpha x\right)^j\right]\end{aligned} \tag{24}$$

$(e)$: $x, (y-\alpha x)$ are independent then $x^a, (y-\alpha x)^b$ are also independent.

Now, we evalute (22) using (23) and (24) as follows.

$$
\begin{aligned}
(22) &\overset{(23),(24)}{=} \frac{1}{4} - \frac{1}{16}\left(\Sigma_{2_{ii}} + \Sigma_{2_{jj}}\right) + \frac{1}{64}\left(\Sigma_{2_{ii}}\Sigma_{2_{jj}} + 2\Sigma_{2_{ij}}^2\right) \\
&\quad + \frac{1}{32}\left(\Sigma_{2_{ii}}^2 + \Sigma_{2_{jj}}^2\right) - \frac{1}{128}\left(\Sigma_{2_{ii}}^2\Sigma_{2_{jj}} + \Sigma_{2_{ii}}\Sigma_{2_{jj}}^2 + 4\Sigma_{2_{ij}}^2\Sigma_{2_{ii}} + 4\Sigma_{2_{ij}}^2\Sigma_{2_{jj}}\right) \\
&\quad + \frac{1}{768}\left(3\Sigma_{2_{ii}}^2\Sigma_{2_{jj}}^2 + 8\Sigma_{2_{ij}}^4 + 24\Sigma_{2_{ij}}^2\Sigma_{2_{ii}}\Sigma_{2_{jj}}\right) \\
&\quad + \mathop{\mathbb{E}}_{(x,y)\sim\Delta}\left[\frac{x^6\dot{\Phi}^6(\xi)}{6!}\left(\frac{1}{2} - \frac{y^2}{8} + \frac{y^4}{48} + \frac{y^6\dot{\Phi}^6(\xi)}{6!}\right)\right] \\
&\leq \frac{1}{4} - \frac{1}{16}\left(\Sigma_{2_{ii}} + \Sigma_{2_{jj}}\right) + \frac{1}{64}\left(\Sigma_{2_{ii}}\Sigma_{2_{jj}} + 2\Sigma_{2_{ij}}^2\right) + \frac{1}{32}\left(\Sigma_{2_{ii}}^2 + \Sigma_{2_{jj}}^2\right) \\
&\quad - \frac{10}{128}\epsilon^3 + \frac{35}{768}\epsilon^4 + \frac{15}{720}\epsilon^3 \qquad ; |\epsilon| \leq \max\{\Delta_{00}, \Delta_{11}\}, \mathbb{E}\left[x^6\right] = 15\Delta_{00} \\
&\leq \frac{1}{4} - \frac{1}{16}\left(\Sigma_{2_{ii}} + \Sigma_{2_{jj}}\right) + \frac{1}{64}\left(\Sigma_{2_{ii}}\Sigma_{2_{jj}} + 2\Sigma_{2_{ij}}^2\right) + \frac{1}{32}\left(\Sigma_{2_{ii}}^2 + \Sigma_{2_{jj}}^2\right) + \frac{1}{16}\epsilon^3
\end{aligned} \tag{25}
$$

where $|\epsilon| \leq \max\{\Delta_{00}, \Delta_{11}\}$.

We get the NTK in Theorem 1 by putting together (25) and (20).

**Corollary 1 (Linear GCN).** In this case, $\sigma(x) := x$ and so derivative $\dot{\sigma}(x) = 1$. Consequently, one can derive $\dot{E}_i = c_\sigma \mathbf{1}_{n\times n}$ from its definition. Therefore, we get NTK for linear GCN in Corollary 1 by substituting $\dot{E}_i$ in general NTK equation in (20).

**Corollary 2 (ReLU GCN).** NTK for ReLU GCN is derived by substituting (7) in general NTK equation in (20) as discussed in the corollary.

### A.2 Proof of NTK for GCN with Skip Connections (Corollary 3 and 4)

We derive the NTKs for GCNs with different skip connections, Skip-PC and Skip-$\alpha$ in this section. Before we present the proofs, we note that there are typographical errors in Definitions 1 and 2, and Corollary 4. The corrections in each are listed as follows,

1. $g_1$ in Definition 1 should be $g_1 := \sqrt{\frac{c_\sigma}{h}} S\sigma_s(H_0)$.
2. $g_1$ in Definition 2 should be $g_1 := \sqrt{\frac{c_\sigma}{h}} ((1-\alpha)S\sigma_s(H_0) + \alpha\sigma_s(H_0))$.
3. $\Sigma_i$ in Corollary 4 should be $\Sigma_i := (1-\alpha)^2 SE_{i-1}S^T + \alpha^2\tilde{E}_0$ where $\tilde{E}_0 = \mathop{\mathbb{E}}_{f\sim\mathcal{N}(0,\Sigma_0)}\left[\sigma_s(f)\sigma_s(f)^T\right]$. We replace $E_0$ with $\tilde{E}_0$ in both Corollary 3 and 4 to be clear.

We clarify that the errors are only typographical and it did not carry forward to the experiments, thus leaving the empirical results and discussions unaffected. The above mentioned errors will be corrected in the final version of the paper. We derive the NTKs for Skip-PC and Skip-$\alpha$ using these definitions.

We observe that the definitions of $g_i \,\forall i \in [1, d+1]$ are different for GCN with skip connections from the vanilla GCN. Despite the difference, the definition of gradient with respect to $W_i$ in (4) does not change as $g_i$ in the gradient accounts for the change and moreover, there is no new learnable parameter since the input transformation $H_0 = XT$ where $T_{ij}$ is sampled from $\mathcal{N}(0,1)$ is not learnable in our setting. Given the fact that the gradient definition holds for GCN with skip connection, the NTK will retain the form from NTK for vanilla GCN as evident from the above derivation. The change in $g_i$ will only affect the co-variance between nodes. Hence, we will derive the co-variance matrix for the discussed skip connections, Skip-PC and Skip-$\alpha$ in the following sections.

**Skip-PC: Co-variance between nodes.** The co-variance between nodes $p$ and $q$ in $f_1$ and $f_i$ are derived below.

$$\begin{aligned}
\mathbb{E}\left[(f_1)_{pk}(f_1)_{qk}\right] &= \left\langle (g_1)_{p.}, (g_1)_{q.} \right\rangle \\
&= \frac{c_\sigma}{h}\left\langle (S\sigma_s(H_0))_{p.}, (S\sigma_s(H_0))_{q.} \right\rangle \\
&= \frac{c_\sigma}{h}\sum_{k=1}^{h} (S\sigma_s(H_0))_{pk}(S\sigma_s(H_0))_{qk} \\
&\overset{h\to\infty}{=} c_\sigma \mathbb{E}\left[(S\sigma_s(H_0))_{pk}(S\sigma_s(H_0))_{qk}\right] \qquad \text{; law of large numbers} \\
&= S_{p.}\tilde{E}_0 S_{.q}^T \qquad ; \tilde{E}_0 = c_\sigma \mathop{\mathbb{E}}_{f\sim\mathcal{N}(0,XX^T)}\left[\sigma_s(f)\sigma_s(f)^T\right] \\
&= (\Sigma_1)_{pq}
\end{aligned} \tag{26}$$

$$\begin{aligned}
\mathbb{E}\left[(f_i)_{pk}(f_i)_{qk}\right] &= \left\langle (g_i)_{p.}, (g_i)_{q.} \right\rangle \\
&= \frac{c_\sigma}{h}\left\langle (S(\sigma(f_{i-1})+\sigma_s(H_0)))_{p.}, (S(\sigma(f_{i-1})+\sigma_s(H_0)))_{q.} \right\rangle \\
&= \frac{c_\sigma}{h}\sum_{k=1}^{h} (S\sigma(f_{i-1})+S\sigma_s(H_0))_{pk}(S\sigma(f_{i-1})+S\sigma_s(H_0))_{qk} \\
&\overset{h\to\infty}{=} c_\sigma \mathbb{E}\left[(S\sigma(f_{i-1})+S\sigma_s(H_0))_{pk}(S\sigma(f_{i-1})+S\sigma_s(H_0))_{qk}\right] \qquad \text{; law of large numbers} \\
&= c_\sigma \Big[\mathbb{E}\left[(S\sigma(f_{i-1}))_{pk}(S\sigma(f_{i-1}))_{qk}\right] + \mathbb{E}\left[(S\sigma(f_{i-1}))_{pk}(S\sigma_s(H_0))_{qk}\right] \\
&\qquad + \mathbb{E}\left[(S\sigma_s(H_0))_{pk}(S\sigma(f_{i-1}))_{qk}\right] + \mathbb{E}\left[(S\sigma_s(H_0))_{pk}(S\sigma_s(H_0))_{qk}\right]\Big] \\
&= S_{p.}E_{i-1}S_{.q}^T + c_\sigma \mathbb{E}\left[(S\sigma(f_{i-1}))_{pk}(S\sigma_s(XW_0))_{qk}\right] \\
&\qquad + c_\sigma \mathbb{E}\left[(S\sigma_s(XW_0))_{pk}(S\sigma(f_{i-1}))_{qk}\right] \\
&\qquad + c_\sigma \mathbb{E}\left[\sum_{r=1}^{n}\sum_{s=1}^{n} S_{pr}S_{qs}\sigma_s(XW_0)_{rk}\,\sigma_s(XW_0)_{sk}\right] \\
&\overset{(f)}{=} S_{p.}E_{i-1}S_{.q}^T + c_\sigma S_{p.}\mathbb{E}\left[\sigma_s(XW_0)_{rk}\,\sigma_s(XW_0)_{sk}\right]S_{.q}^T \\
&= S_{p.}E_{i-1}S_{.q}^T + S_{p.}\tilde{E}_0 S_{.q}^T \\
&= S_{p.}E_{i-1}S_{.q}^T + (\Sigma_1)_{pq} \\
&= (\Sigma_i)_{pq}
\end{aligned} \tag{27}$$

$(f)$: $\mathbb{E}\left[(S\sigma(f_{i-1}))_{pk}(S\sigma_s(XW_0))_{qk}\right]$ and $\mathbb{E}\left[(S\sigma_s(XW_0))_{pk}(S\sigma(f_{i-1}))_{qk}\right]$ evaluate to 0 by conditioning on $W_0$ first and rewriting the expectation based on this conditioning. The terms within expectation are independent when conditioned on $W_0$, and hence it is $\mathop{\mathbb{E}}_{W_0}\left[\mathop{\mathbb{E}}_{\Sigma_{i-1}|W_0}\left[(S\sigma(f_{i-1}))_{pk}|W_0\right]\mathop{\mathbb{E}}_{\Sigma_{i-1}|W_0}\left[(S\sigma_s(XW_0))_{qk}|W_0\right]\right]$ by taking $h$ in $W_0$ going to infinity first. Here, $\mathop{\mathbb{E}}_{\Sigma_{i-1}|W_0}\left[(S\sigma_s(XW_0))_{qk}|W_0\right] = 0$.

We get the co-variance matrix for all pairs of nodes $\Sigma_1 = S\tilde{E}_0S^T$ and $\Sigma_i = SE_{i-1}S^T + \Sigma_1$ from (26) and (27).

**Skip-$\alpha$: Co-variance between nodes.** Let $p$ and $q$ be two nodes and the co-variance between $p$ and $q$ in $f_1$ and $f_i$ are derived below.

$$
\begin{aligned}
\mathbb{E}\left[(f_1)_{pk}(f_1)_{qk}\right] &= \left\langle (g_1)_{p.}, (g_1)_{q.}\right\rangle \\
&= \frac{c_\sigma}{h}\sum_{k=1}^{h}\left((1-\alpha)S\sigma_s(H_0)+\alpha\sigma_s(H_0)\right)_{pk}\left((1-\alpha)S\sigma_s(H_0)+\alpha\sigma_s(H_0)\right)_{qk} \\
&\overset{h\to\infty}{=} c_\sigma\mathbb{E}\left[\left((1-\alpha)S\sigma_s(H_0)+\alpha\sigma_s(H_0)\right)_{pk}\left((1-\alpha)S\sigma_s(H_0)+\alpha\sigma_s(H_0)\right)_{qk}\right] \\
&= c_\sigma\Big[(1-\alpha)^2\mathbb{E}\left[(S\sigma_s(H_0))_{pk}(S\sigma_s(H_0))_{qk}\right] \\
&\quad + (1-\alpha)\alpha\left(\mathbb{E}\left[(S\sigma_s(H_0))_{pk}(\sigma_s(H_0))_{qk}\right]+\mathbb{E}\left[(S\sigma_s(H_0))_{qk}(\sigma_s(H_0))_{pk}\right]\right) \\
&\quad + \alpha^2\mathbb{E}\left[(\sigma_s(H_0))_{pk}(\sigma_s(H_0))_{qk}\right] \\
&= (1-\alpha)^2 S_{p.}\tilde{E}_0 S^T_{.q} + (1-\alpha)\alpha\left(S_{p.}\left(\tilde{E}_0\right)_{.q}+\left(\tilde{E}_0\right)_{p.}S^T_{.q}\right)+\alpha^2\left(\tilde{E}_0\right)_{pq} \\
&= (\Sigma_1)_{pq} && (28)
\end{aligned}
$$

$$
\begin{aligned}
\mathbb{E}\left[(f_i)_{pk}(f_i)_{qk}\right] &= \left\langle (g_i)_{p.}, (g_i)_{q.}\right\rangle \\
&= \frac{c_\sigma}{h}\sum_{k=1}^{h}\left((1-\alpha)S\sigma(f_{i-1})+\alpha\sigma_s(H_0)\right)_{pk}\left((1-\alpha)S\sigma(f_{i-1})+\alpha\sigma_s(H_0)\right)_{qk} \\
&\overset{h\to\infty}{=} c_\sigma\mathbb{E}\left[\left((1-\alpha)S\sigma(f_{i-1})+\alpha\sigma_s(H_0)\right)_{pk}\left((1-\alpha)S\sigma(f_{i-1})+\alpha\sigma_s(H_0)\right)_{qk}\right] \\
&= c_\sigma\Big[(1-\alpha)^2\mathbb{E}\left[(S\sigma(f_{i-1}))_{pk}(S\sigma(f_{i-1}))_{qk}\right]+\alpha^2\mathbb{E}\left[(\sigma_s(H_0))_{pk}(\sigma_s(H_0))_{qk}\right] \\
&\quad + (1-\alpha)\alpha\left(\mathbb{E}\left[(S\sigma(f_{i-1}))_{pk}(\sigma_s(H_0))_{qk}\right]+\mathbb{E}\left[(\sigma_s(H_0))_{pk}(S\sigma(f_{i-1}))_{qk}\right]\right)\Big] \\
&\overset{(g)}{=} (1-\alpha)^2 S_{p.}E_{i-1}S^T_{.q}+\alpha^2\left(\tilde{E}_0\right)_{pq} = (\Sigma_i)_{pq} && (29)
\end{aligned}
$$

$(g)$: same argument as $(f)$ in derivation of $\Sigma_i$ in Skip-PC.

We get the co-variance matrix for all pairs of nodes $\Sigma_1 = (1-\alpha)^2 S\tilde{E}_0S^T + \alpha(1-\alpha)\left(S\tilde{E}_0+\tilde{E}_0S^T\right)+\alpha^2\tilde{E}_0$ and $\Sigma_i = (1-\alpha)^2SE_{i-1}S^T+\alpha^2\tilde{E}_0$ from (28) and (29).

### A.3 Normalisation constant $c_\sigma$ (Remark 1 and 2).

We derive the normalisation constant $c_\sigma$ loosely, as the purpose of $c_\sigma$ is to preserve the input norm approximately. We focus on general form of a network with skip connection (not GCN in particular), where the output vector of size $h$ from any hidden layer $l$ with weight matrix $W \in \mathbb{R}^{h\times h}$ and transformed input vector $X_0$ of size $h$ can be written as $g_l := \sqrt{\frac{c_\sigma}{h}}\left(\sigma(Wg_{l-1})+X_0\right) \in \mathbb{R}^{h\times 1}$. The role of the normalisation constant $c_\sigma$ is to maintain $\|g_l\|_2 \simeq \|X_0\|_2$ and is derived as follows.

$$
\begin{aligned}
\|X_0\|_2^2 &= \|g_l\|_2^2 = \frac{c_\sigma}{h}\sum_{k=1}^{h}\left(\sigma(Wg_{l-1})+X_0\right)_k^2 \\
\|X_0\|_2^2 &= c_\sigma\mathbb{E}\left[\left(\sigma(Wg_{l-1})_k\right)^2+(X_0)_k^2+2\sigma(Wg_{l-1})_k(X_0)_k\right] \qquad ; h\to\infty \\
\|X_0\|_2^2 &= c_\sigma \mathop{\mathbb{E}}_{u\sim\mathcal{N}(0,\|X_0\|^2)}\left[(\sigma(u))^2\right]+\|X_0\|_2^2 \qquad ; \mathbb{E}\left[\sigma(Wg_{l-1})_k(X_0)_k\right]=0 \\
c_\sigma &= \left(\mathop{\mathbb{E}}_{u\sim\mathcal{N}(0,1)}\left[(\sigma(u))^2\right]+1\right)^{-1} \qquad ; \text{normalised } X_0 && (30)
\end{aligned}
$$

We use this $c_\sigma$ for GCN with skip connection in our work and it evaluates to $2/3$ for $\sigma(x) :=$ ReLU$(x)$ in GCN as stated in Remark 2. The evident change for a network without skip connection is to not add $X_0$ in $g_l := \sqrt{\frac{c_\sigma}{h}}\sigma(W g_{l-1})$ and consequently by following the proof, we get $c_\sigma = \left(\underset{u\sim\mathcal{N}(0,1)}{\mathbb{E}}\left[(\sigma(u))^2\right]\right)^{-1}$ as mentioned in Remark 1.

# B Additional Experimental Results

## B.1 Datasets for binary node classification

Since the considered datasets *Cora*, *Citeseer* and *WebKB* are for multi-class node classification, we converted the datasets to have binary class by grouping the classes into two sets. Table 1 shows the label grouping for each dataset and total number of nodes with the grouped labels respectively. The classes in all the datasets are well balanced and sensible to learn for binary classification problem which is proved from the performance of a simple graph neural network like linear vanilla GCN. The train-test split for each dataset is 708 and 2000 nodes for Cora, 312 and 2000 for Citeseer, and 377 and 500 for WebKB for all the experiments.

| | **Cora** | | **Citeseer** | | **WebKB** | |
|---|---|---|---|---|---|---|
| | Class Groups | #nodes | Class Groups | #nodes | Class Groups | #nodes |
| **Class 1** | Neural_Networks<br>Theory<br>Probabilistic_Methods | 1595 | Agents<br>AI<br>ML | 1435 | student | 415 |
| **Class 2** | Case_Based<br>Rule_Learning<br>Reinforcement<br>Genetic_Algorithms | 1103 | DB<br>IR<br>HCI | 1877 | faculty<br>staff<br>course<br>project | 462 |
| **Total** | | 2708 | | 3312 | | 877 |

Table 1: Class grouping in datasets for binary node classification.

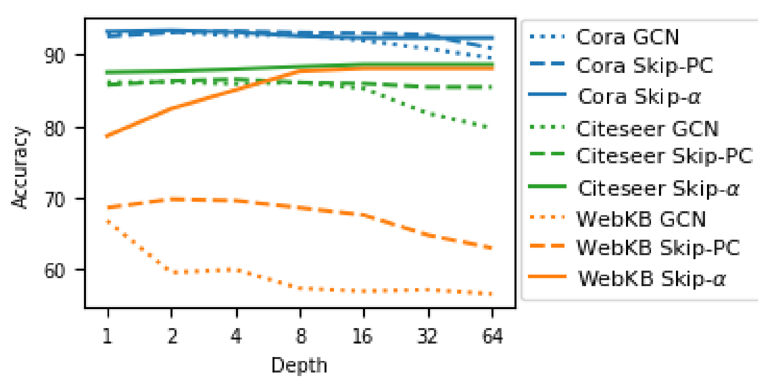


Figure 5: Performance validation of vanilla GCN, Skip-PC and Skip-$\alpha$ with $\sigma(.) :=$ ReLU, $\sigma_s(.) :=$ ReLU and $\alpha = 0.2$ using the respective NTKs.

## B.2 Vanilla GCN vs GCN with Skip Connections

We established ReLU GCN is preferred over linear in Section 2 and ReLU for the input transformation in Section 3. Hence, we focus on $\sigma(.) := \text{ReLU}$ and $\sigma_s(.) := \text{ReLU}$ with $\alpha = 0.2$ for Skip-$\alpha$ to validate the performance of vanilla GCN and GCN with skip connections, Skip-PC and Skip-$\alpha$. We use the respective NTKs to validate the performance. Figure 5 shows that GCN with skip connection outperforms vanilla GCN even in deeper depths, and Skip-$\alpha$ gives better performance than Skip-PC with depth.

**Note.** In Figure 2, the performance of Skip-PC with ReLU $\sigma_s(.)$ evaluated on Citeseer when depth $= 32$ is different from what is plotted due to some numerical precision error. We evaluated the performance at depth $30, 31$ and used it to plot.

## B.3 Convergence of NTK with depth - Cora, Citeseer, WebKB

We presented the convergence of NTK with depth for ReLU GCN with and without skip connections evaluated on Cora dataset in Figure 4. Here, we present the convergence plot for Linear GCN evaluated on Cora and all discussed linear and ReLU networks evaluated Citeseer and WebKB. The observation is similar to the discussion in Section 5. Figures 6, 7 and 8 show the convergence plots for linear GCN evaluated on Cora, ReLU and linear GCNs with and without skip connections for Citeseer and WebKB, respectively.

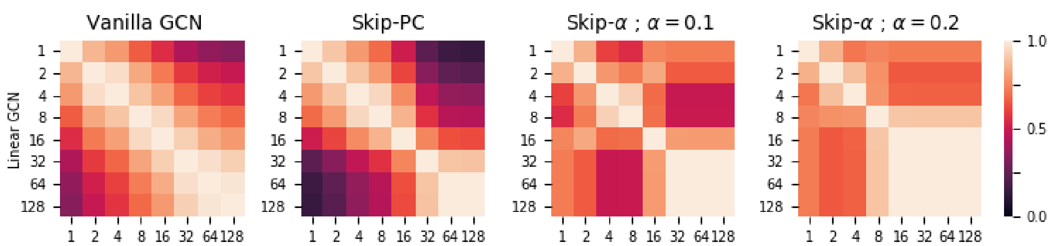


Figure 6: Convergence of NTK with depth for all the discussed linear architectures evaluated on Cora dataset.

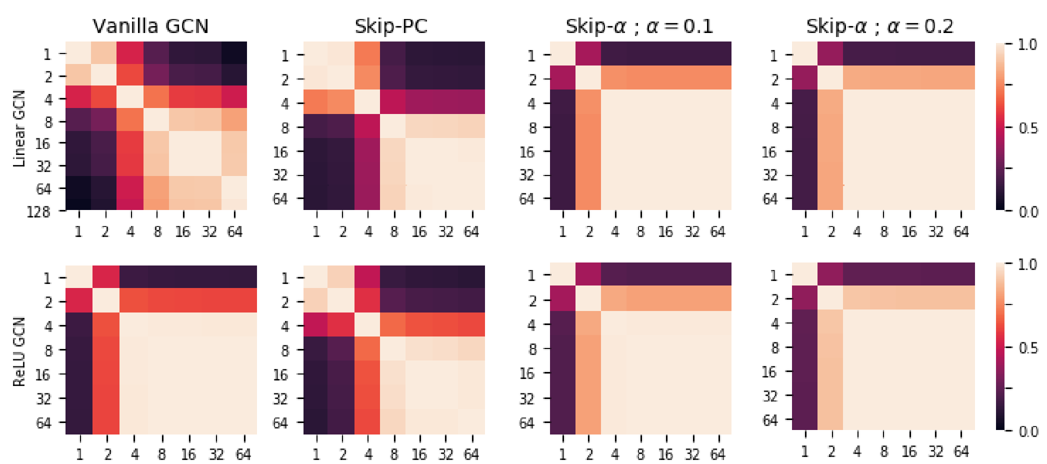


Figure 7: Convergence of NTK with depth for all the discussed linear and ReLU architectures evaluated on Citeseer dataset.

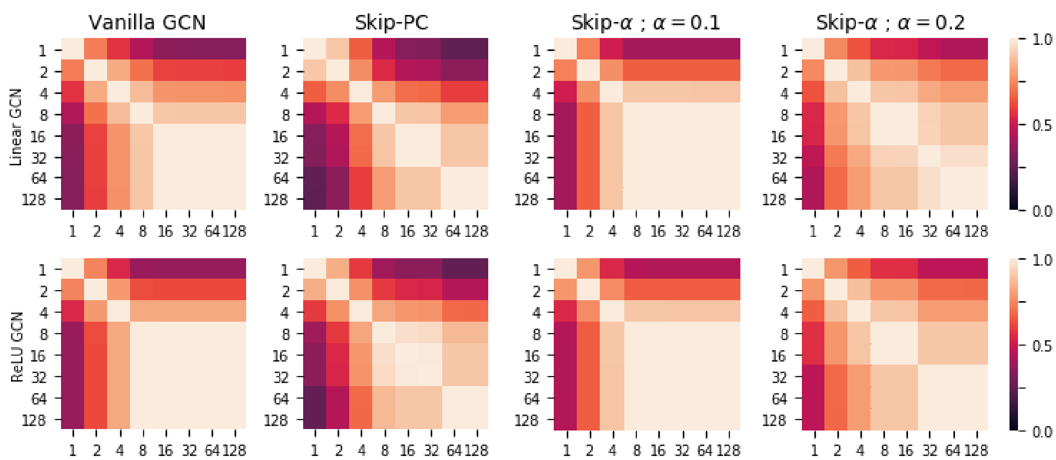


Figure 8: Convergence of NTK with depth for all the discussed linear and ReLU architectures evaluated on WebKB dataset.