Supplemental Material

This supplemental material accompanies the main paper and is divided into two principal sections. The first section provides a detailed proof of Theorem 3.3 as presented in the main paper. The second section presents further accuracy evaluation results for various GNN tasks and models, along with the hyperparameters used in these experiments.

1 PROOF OF THEOREM 3.3

We organize the proof as follows. First, we formally define the structure of graph neural network (GNN) in Section 1.1. Then we give the detailed proof throughout Section 1.2- 1.4: (1) we prove that the gradient is bounded and the loss function is almost convex within the neighborhood of random initialization in Section 1.2; (2) we prove that with the optimal weights of uncompressed features, quantized features produce a bounded loss compared to the uncompressed features in Section 1.3; and (3) with the help of the lemmas introduced in (1) and (2), we prove Theorem 3.3 in Section 1.4. Finally, we present an analysis of the factors that may affect the compression ratio in Section 1.5.

1.1 GNN Definitions

As Graph Convolutional Network (GCN) is one of the most widely-used GNN models, we use it to drive our theoretical proofs and analysis. Here, we follow the literature [11] to define the GCN structure as follows:

$$
\begin{aligned}
&g_{i,0} = \boldsymbol{A}x_i \\
&h_{i,0} = \phi(g_{i,0}) \\
&g_{i,l} = \boldsymbol{W}_l h_{i,l-1} && ,for\ i \in [n]\ and\ l \in [L] \\
&\overline{g}_{i,l} = \textstyle\sum_{j \in N(i)} \frac{1}{|N(i)|} g_{j,l} && ,for\ i \in [n]\ and\ l \in [L] \\
&h_{i,l} = \phi(\overline{g}_{i,l}) && ,for\ i \in [n]\ and\ l \in [L] \\
&f_i(W, X) = a^\top h_{i,L}
\end{aligned}
$$

Here x_i is the input feature, $g_{i,l}$ and $h_{i,l}$ are the feature vectors of node i before and after the ReLU activation function ϕ of layer l. Note that these notations and definitions are similar with those traditional DNN architecture studied in [1, 6]. However, the main difference lies in that GNN introduces an aggregation phase i.e. $\overline{g}_{i,l}$, where it gathers information by averaging neighbor features. It can also be written in a GCN style as $\overline{g}_{i,l} = \boldsymbol{G}_l \boldsymbol{L}_{:,i}$, where $\boldsymbol{G}_l = (g_{1,l}, \cdots, g_{n,l})$. We set $\boldsymbol{L} = \hat{\boldsymbol{D}}^{-1}\hat{\boldsymbol{A}}$, where $\hat{\boldsymbol{A}}$ and $\hat{\boldsymbol{D}}$ are the adjacency matrix and degree matrix including self loop. We use the diagonal matrix $\boldsymbol{D}$ to indicate the consequences of ReLU function, so $h_{i,l}$ can be written as $\boldsymbol{D}_{i,l}\overline{g}_{i,l}$.

With $\boldsymbol{S} = \{x \in \mathbb{R}^d : \|x\|_2 = 1\}$, we scale the input features to $\boldsymbol{X} = \{x_i \in \boldsymbol{S} : i \in [n]\}$ following [1]. We also follow the same initialization method as Definition 2.3 in [1] with a minor change on the output layer: $a_j \sim \mathcal{N}(0, 1/\overline{C}_L^f)$. This is intended to ensure the output scale fits the label, $\overline{C}_L^f$ is a factor we discuss in Section 1.5. We only train hidden layer weights $\boldsymbol{W}$ while keeping input and output layer weights $\boldsymbol{A}$ and a with random initialization as [1] does.

We denote $\mathcal{P}_{B(R)}$ as the Euclidean projection to the convex set $B(R)$ and $\boldsymbol{W}^{(t)}$ as the Weights after t iterations. We run projected gradient descent based on the constraint set $B(R)$ with step size α for T steps using the following update rule.

$$
\begin{aligned}
&\boldsymbol{V}^{(t+1)} = \boldsymbol{W}^{(t)} - \alpha \nabla_{\boldsymbol{W}} Loss(\boldsymbol{W}^{(t)}, \boldsymbol{X}), \\
&\boldsymbol{W}^{(t+1)} = \mathcal{P}_{B(R)}(\boldsymbol{V}^{(t+1)})
\end{aligned}
$$

To measure the effect of aggregation, we introduce two factors $C_{i,l}^e$ and $C_{i,l}^f$ to represent the expectation of scaling of $h_{i,l}$ caused by the aggregation of l layers. These factors have the following properties:

$$
C_{i,0}^e = 1, \quad C_{i,0}^f = 1
$$

$$
C_{i,l+1}^e = \sqrt{\sum_{j \in N(i)} \sum_{k \in N(i)} r_{l,j,k}^e \frac{1}{|N(i)|^2} C_{j,l}^e C_{k,l}^e}
$$

$$
C_{i,l+1}^f = \sqrt{\sum_{j \in N(i)} \sum_{k \in N(i)} r_{l,j,k}^f \frac{1}{|N(i)|^2} C_{j,l}^f C_{k,l}^f}
$$

$$
C_{i,l}^e \le 1, \quad C_{i,l}^f \le 1
$$

where $r_{l,j,k}^f$ and $r_{l,j,k}^e$ denotes the correlation of feature components and errors in layer l between the node j and node k, respectively. With Assumption 3.1, $r_{0,j,k}^e = 0$ $(j \neq k)$, $C_{i,l}^e$ is determined by the graph structure. $C_{i,l}^f$ is also relative to the graph structure, but it is also affected by the correlation of node features. $C_{i,l}^e \le C_{i,l}^f$ has a high probability as $r_{0,j,k}^f > 0$ has high probability. So we have $\frac{C_{i,l}^e}{C_{i,l}^f} \le 1$. $C_{i,l}^e$ and $C_{i,l}^f$ can also be seen as weighted average of l-hop neighbors, thus we have $C_{i,l}^e \ge \frac{1}{n}$ and $C_{i,l}^f \ge \frac{1}{n}$. Therefore, we expect the value of $C_{i,l}^e$, $C_{i,l}^f$, and $C_{i,l}^e$ / $C_{i,l}^f$ gets smaller as the layer l gets deeper. While the exact value of these factors could not be directly calculated with the statistics of graph, we can obtain them through quick profiling tests and derive the bound from the analysis. We conduct an analysis of these factors in Section 1.5.

1.2 Gradient Bounds and Almost Convex

In this subsection, we will prove that the gradient is bounded and the loss function is almost convex with regard to weights within the neighborhood of random initialization. We use the same proof sketch as [6], but our proof differs by having fixed input features.

Lemma 1.1. *If $m \ge d$, with probability $1 - O(nL)e^{-\Omega(m)}$ at initialization, we have $\|\boldsymbol{A}\|_2 = O(1)$, $\|\boldsymbol{W}_L\|_2 = O(1), \forall l \in [L]$, and $\|a\|_2 = O(\frac{\sqrt{m}}{\overline{C}_L^f})$.*

Lemma 1.2. *For any fixed input $\boldsymbol{X} = \{x_i \in \boldsymbol{S} : i \in [n]\}$, with probability $1 - O(L)e^{-\Omega(m/L)}$ over the randomness of initialization, we have $\forall i \in [n]$ and $l \in \{0, ..., L\}$, $\|h_{i,l}\|_2 \in [\frac{2}{3}C_{i,l}^f, \frac{4}{3}C_{i,l}^f]$.*

Table 1: Summary of notations.

Notation	Description
n	Number of nodes in the graph.
d	Input features dimension.
m	Hidden layer dimension, also known as the width of network.
$N(i)$	Set of nodes adjacent to i.
L	Number of layers.
$[n]$	$\{1, 2, \cdots, n\}$.
$\boldsymbol{X}$	Input features of all nodes, with shape $n \times d$. Note that we also use x_i to denote features of node i.
$\boldsymbol{X'}$	Quantized input features of all nodes with the same shape as $\boldsymbol{X}$.
$\boldsymbol{A}$	Input layer weights with the shape of $m \times d$.
$\boldsymbol{W}$	Hidden layer weights $\boldsymbol{W} = (\boldsymbol{W}_1, \cdots, \boldsymbol{W}_L)$, where weights of each layer has the shape of $m \times d$.
a	Output layer weights with the shape of $m \times 1$.
$\boldsymbol{D}$	Diagonal matrix, $\boldsymbol{D}_{i,l} = diag(I(\overline{g}_{i,l} \geq 0))$, where I is the element-wise indicator function.
$f_i(\boldsymbol{W}, \boldsymbol{X})$	Output of node i with weights $\boldsymbol{W}$ and input $\boldsymbol{X}$
l	Loss function
$Loss(\boldsymbol{W}, \boldsymbol{X})$	Loss with weights $\boldsymbol{W}$ and inputs $\boldsymbol{X}$
$B(R)$	The neighborhood of initial weights, $B(R) = \{\boldsymbol{W} : \|\boldsymbol{W}_l - \boldsymbol{W}_l^{(0)}\|_F \leq R/\sqrt{m}, l \in [L]\}$
$\mathcal{P}_{B(R)}$	Euclidean projection to the convex set $B(R)$.
α	Step size of projected gradient descent.
T	Number of steps.
$C_{i,l}^f$	The expected scale of node i's features after the aggregation of l layers.
$C_{i,l}^e$	The expected scale of node i's error caused by quantization after the aggregation of l layers.
$\overline{C}_L^f$	The mean value of $C_{i,L}^f$.
$\overline{C}_L^e$	The mean value of $C_{i,L}^e$.
$\hat{C}_L$	$C_{i,L}^e / C_{i,L}^f$.

Lemma 1.3. *If $m = \Omega(LlogL)$, for any fixed input $\boldsymbol{X} = \{x_i \in \boldsymbol{S} : i \in [n]\}$, with probability $1 - e^{-\Omega(m/L)}$ over the randomness of initialization, we have for every $l \in [L]$ and every $i \in [n]$,$\|a^\top \boldsymbol{D}_{i,L}\boldsymbol{W}_{i,L} \cdots \boldsymbol{D}_{i,l}\boldsymbol{W}_{i,l}\|_2 = O(\frac{\sqrt{mL}}{\overline{C}_L^f})$.*

Proof of lemma1.1, lemma1.2, lemma1.3. These are restatements of Lemma A.1, Lemma A.2 and Lemma A.3 in [6]. Since our GNN architecture shares the same weight initialization method as regular DNN, we can directly apply these results over the randomness of initialization. In Lemma 1.2 the value is different from Lemma A.2 in [6] due to aggregation, but with the same random weight initialization, the probability and relative bound is consistent.

We prove the perturbation of gradients caused by perturbation of weights is small following [1].

Lemma 1.4. *Given any fixed inputs $\boldsymbol{X} = \{x_i \in \boldsymbol{S} : i \in [n]\}$. If $m \geq max(d, \Omega(LlogL))$, $\frac{R}{\sqrt{m}} \leq \frac{1}{SL^6(\log m)^3}$ for some sufficiently large constant S, with probability $1 - O(L)e^{-\Omega((mR)^{2/3}L)}$ over the randomness of initialization, we have for any $\boldsymbol{W} \in B(R)$, any $l \in [L]$ and every $i \in [n]$,*

$$\|\frac{\partial f_i(\boldsymbol{W}, X)}{\partial \boldsymbol{W}_l} - \frac{\partial f_i(\boldsymbol{W}^{(0)}, X)}{\partial \boldsymbol{W}_l}\|_F = O(\frac{R^{\frac{1}{3}} m^{\frac{1}{3}} L^2 \sqrt{\log m}}{\overline{C}_L^f})$$

$$\|\frac{\partial f_i(\boldsymbol{W}, X)}{\partial \boldsymbol{W}_l}\|_F = O(\frac{\sqrt{mL}}{\overline{C}_L^f})$$

Proof. We denote $\boldsymbol{D'}$, $\boldsymbol{W'}$, and h' as perturbed $\boldsymbol{D}$, $\boldsymbol{W}$, and h, respectively. With Claim 8.3 and Lemma 8.2(c) of [1], we have that when $\frac{R}{\sqrt{m}} \leq \frac{1}{SL^6(\log m)^3}$, with probability $1 - e^{-\Omega(m(R/\sqrt{m})^{2/3}L)}$,

$$\|\boldsymbol{D}'_{i,l} - \boldsymbol{D}_{i,l}\|_0 = O(m(\frac{R}{\sqrt{m}})^{2/3}L) \tag{1}$$

and

$$\|h'_{i,l} - h_{i,l}\|_0 = O(C_{i,l}^f \frac{R}{\sqrt{m}} L^{5/2}\sqrt{\log m}) \tag{2}$$

Combining (1) and Lemma 8.7 of [1], we get:

$$\begin{aligned}&\|(a^\top \boldsymbol{D}'_{i,L}\boldsymbol{W}'_L \cdots \boldsymbol{D}'_{i_{l+1},l+1}\boldsymbol{W}'_{l+1}\boldsymbol{D}'_{i_L,l}) \\ &\quad - (a^\top \boldsymbol{D}_{i,L}\boldsymbol{W}_L \cdots \boldsymbol{D}_{i_{l+1},l+1}\boldsymbol{W}_{l+1}\boldsymbol{D}_{i_L,l})\|_2 \\ &\leq O(\frac{1}{\overline{C}_L^f}(\frac{R}{\sqrt{m}})^{1/3} L^2 \sqrt{m \log m})\end{aligned}$$

During backward propagation, we have:

$$\frac{\partial f_i(\boldsymbol{W},X)}{\partial \boldsymbol{W}_L} = \underbrace{\sum_{i_{L-1}\in N(i)} \frac{1}{|N(i)|} \cdots \sum_{i_L \in N(i_{l+1})} \frac{1}{|N(i_{l+1})|}}_{L-l \text{ sums}} h_{i_L,l-1} a^\top \boldsymbol{D}_{i,L} \boldsymbol{W}_L \cdots \boldsymbol{D}_{i_{l+1},l+1} \boldsymbol{W}_{l+1} \boldsymbol{D}_{i_L,l}$$

Then, using Lemma1.2, Lemma1.3, and the above results, we have:

$$\begin{aligned}
&\|\frac{\partial f_i(\boldsymbol{W},X)}{\partial \boldsymbol{W}_l} - \frac{\partial f_i(\boldsymbol{W}^{(0)},X)}{\partial \boldsymbol{W}_l}\|_F = \\
&\|\underbrace{\sum_{i_{L-1}\in N(i)} \frac{1}{|N(i)|} \cdots \sum_{i_L \in N(i_{l+1})} \frac{1}{|N(i_{l+1})|}}_{L-l \text{ sums}} \\
&h'_{i_L,l-1} a^\top \boldsymbol{D}'_{i,L} \boldsymbol{W}'_L \cdots \boldsymbol{D}'_{i_{l+1},l+1} \boldsymbol{W}'_{l+1} \boldsymbol{D}'_{i_L,l} \\
&- h_{i_L,l-1} a^\top \boldsymbol{D}_{i,L} \boldsymbol{W}_L \cdots \boldsymbol{D}_{i_{l+1},l+1} \boldsymbol{W}_{l+1} \boldsymbol{D}_{i_L,l}\|_F \\
\le &\underbrace{\sum_{i_{L-1}\in N(i)} \frac{1}{|N(i)|} \cdots \sum_{i_L \in N(i_{l+1})} \frac{1}{|N(i_{l+1})|}}_{L-l \text{ sums}} \\
&[\|h'_{i_L,l-1} - h_{i_L,l-1}\|_2 \cdot \|a^\top \boldsymbol{D}_{i,L} \boldsymbol{W}_L \cdots \boldsymbol{W}_{l+1} \boldsymbol{D}_{i_L,l}\|_2 \\
&+ \|h_{i_L,l-1}\|_2 \cdot \|(a^\top \boldsymbol{D}'_{i,L} \boldsymbol{W}'_L \cdots \boldsymbol{W}'_{l+1} \boldsymbol{D}'_{i_L,l}) \\
&- (a^\top \boldsymbol{D}_{i,L} \boldsymbol{W}_L \cdots \boldsymbol{W}_{l+1} \boldsymbol{D}_{i_L,l})\|_2 \\
\le &\underbrace{\sum_{i_{L-1}\in N(i)} \frac{1}{|N(i)|} \cdots \sum_{i_L \in N(i_{l+1})} \frac{1}{|N(i_{l+1})|}}_{L-l \text{ sums}} \\
&[O(C^f_{i,l-1} \frac{R}{\sqrt{m}} L^{5/2}\sqrt{\log m}) \cdot O(\frac{\sqrt{mL}}{\overline{C}^f_L}) \\
&+ O(C^f_{i,l-1}) \cdot O(\frac{1}{\overline{C}^f_L} (\frac{R}{\sqrt{m}})^{1/3} L^2 \sqrt{m \log m})] \\
= &\underbrace{\sum_{i_{L-1}\in N(i)} \frac{1}{|N(i)|} \cdots \sum_{i_L \in N(i_{l+1})} \frac{1}{|N(i_{l+1})|}}_{L-l \text{ sums}} \\
&O(\frac{C^f_{i,l-1}}{\overline{C}^f_L} R^{1/3} m^{1/3} L^2 \sqrt{\log m}) \\
=&O(\frac{R^{1/3} m^{1/3} L^2 \sqrt{\log m}}{\overline{C}^f_L})
\end{aligned}$$

where $C^f_{i,l-1} \le 1$.

With Lemma1.2, Lemma1.3, (2), and $\frac{R}{\sqrt{m}} \le \frac{1}{SL^6(\log m)^3}$, we have

$$\begin{aligned}
\|\frac{\partial f_i(\boldsymbol{W},X)}{\partial \boldsymbol{W}_l}\|_F &= O(\frac{\sqrt{mL}}{\overline{C}^f_L}) \cdot (O(C^f_{i,l}) + O(C^f_{i,l} \frac{R}{\sqrt{m}} L^{5/2}\sqrt{\log m})) \\
&= O(\frac{C^f_{i,l}\sqrt{mL}}{\overline{C}^f_L}) \\
&= O(\frac{\sqrt{mL}}{\overline{C}^f_L})
\end{aligned}$$

Then we are able to prove the loss function is almost convex within the neighborhood $B(R)$ for any fixed X.

LEMMA 1.5. *If $R = O(\frac{\sqrt{m}}{L^6(\log m)^3})$, with probability at least $1 - O(nL)e^{-\Omega((mR)^{2/3}L)}$ over random initialization, we have for any $\boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)} \in B(R)$, any $\boldsymbol{X} = \{x_i \in \boldsymbol{S} : i \in [n]\}$, and any $Y = (y_1, \cdots, y_n) \in \mathbb{R}^n$,*

$$\begin{aligned}
l(f_i(\boldsymbol{W}^{(2)},X), y_i) \ge\ & l(f_i(\boldsymbol{W}^{(1)},X), y_i) \\
&+ \left\langle \nabla_{\boldsymbol{W}} l(f_i(\boldsymbol{W},X), y_i), \boldsymbol{W}^{(2)} - \boldsymbol{W}^{(1)} \right\rangle \\
&- \|\boldsymbol{W}^{(2)} - \boldsymbol{W}^{(1)}\|_F O(\frac{R^{1/3} m^{1/3} L^{5/2} \sqrt{\log m}}{\overline{C}^f_L})
\end{aligned}$$

Proof. For any fixed X and any fixed i, with probability $1 - O(nL)e^{-\Omega((mR)^{2/3}L)}$, we have:

$$\begin{aligned}
&l(f_i(\boldsymbol{W}^{(2)},X), y_i) - l(f_i(\boldsymbol{W}^{(1)},X), y_i) \\
&\quad - \left\langle \nabla_{\boldsymbol{W}} l(f_i(\boldsymbol{W},X), y_i), \boldsymbol{W}^{(2)} - \boldsymbol{W}^{(1)} \right\rangle \\
&\ge \frac{\partial}{\partial f} l(f_i(\boldsymbol{W}^{(1)},X), y_i)[f_i(\boldsymbol{W}^{(2)},X) \\
&\quad - f_i(\boldsymbol{W}^{(1)},X) - \left\langle \nabla_{\boldsymbol{W}} f_i(\boldsymbol{W}^{(1)},X), \boldsymbol{W}^{(2)} - \boldsymbol{W}^{(1)} \right\rangle] \\
&= \frac{\partial}{\partial f} l(f_i(\boldsymbol{W}^{(1)},X), y_i) \left\langle \int_0^1 (\nabla_{\boldsymbol{W}} f_i(t\boldsymbol{W}^{(2)} + (1-t)\boldsymbol{W}^{(1)},X) \right. \\
&\quad \left. - \nabla_{\boldsymbol{W}} f_i(\boldsymbol{W}^{(1)},X)) dt, \boldsymbol{W}^{(2)} - \boldsymbol{W}^{(1)} \right\rangle \\
&\ge -\|\boldsymbol{W}^{(2)} - \boldsymbol{W}^{(1)}\|_F O(\frac{R^{1/3} m^{1/3} L^{5/2} \sqrt{\log m}}{\overline{C}^f_L})
\end{aligned}$$

The first inequality is due to convexity of l with regard to f and the second inequality is due to Lemma 1.4. $\frac{\partial l}{\partial f}$ is bounded and $\frac{\partial f}{\partial \boldsymbol{W}} = (\frac{\partial f}{\partial \boldsymbol{W}_1}, \cdots, \frac{\partial f}{\partial \boldsymbol{W}_L})$.

1.3 Loss after Quantization

In this subsection we prove with the optimal weights of uncompressed features, the difference between losses using quantized features and original features as input is small.

LEMMA 1.6. *Let $\boldsymbol{W}^{(*)} = \arg\min_{\boldsymbol{W}\in B(R)} Loss(\boldsymbol{W}, X)$ be the optimal weights of GNN trained using original features X, with loss function satisfying Assumption 3.1, use the same weights on quantized features X' satisfying $\|x'_i - x_i\|_2 \le \delta$ and Assumption 3.1, with*

high probability over the randomness of initialization of $a^\top$ *we have*

$$Loss(\boldsymbol{W}^{(*)}, X') - Loss(\boldsymbol{W}^{(*)}, X) = O(\hat{C}_L\delta)$$

Proof. We denote $\boldsymbol{W}^{(*)}$ as the optimal weights without feature compression, so the optimal loss is $Loss(\boldsymbol{W}^{(*)}, X)$. We use $h_{i,l}(X)$ to indicate $h_{i,l}$ with regard to input feature X, we can get

$$\begin{aligned}
&Loss(\boldsymbol{W}^{(*)}, X') - Loss(\boldsymbol{W}^{(*)}, X)\\
&= \frac{1}{n}\sum_{i=1}^{n}[l(f_i(\boldsymbol{W}^{(*)}, X'), y_i) - l(f_i(\boldsymbol{W}^{(*)}, X), y_i)]\\
&= \frac{1}{n}\sum_{i=1}^{n}\frac{\partial l}{\partial f}(f_i(\boldsymbol{W}^{(*)}, X') - f_i(\boldsymbol{W}^{(*)}, X))\\
&= \frac{1}{n}\sum_{i=1}^{n}\frac{\partial l}{\partial f}(a^\top(h_{i,L}(X') - h_{i,L}(X)))\\
&= \frac{1}{n}\sum_{i=1}^{n}O(\frac{1}{\overline{C}_L^f})\cdot O(\delta C_{i,L}^e)\\
&= O(\frac{\delta\overline{C}_L^e}{\overline{C}_L^f})\\
&= O(\hat{C}_L\delta)
\end{aligned}$$

where we define $\hat{C}_L = \frac{\overline{C}_L^e}{\overline{C}_L^f}$ as the ratio of the impact of error to feature on the final loss.

1.4 Proof of Theorem 3.3

With Lemma1.4, Lemma1.5, and Lemma1.6, we are ready to prove Theorem 3.3.

Proof of Theorem 3.3. For a projected gradient descent with in total T steps starting from initialization $\boldsymbol{W}^{(0)}$, we denote $\boldsymbol{W}^{(t)}$ as the weights after t steps with step size α. $\boldsymbol{W}^{(t)} \in B(R)$ holds for all $t = 0, 1, \cdots, T$.

The update rule of projected gradient descent is $\boldsymbol{W}^{(t+1)} = \mathcal{P}_{B(R)}(\boldsymbol{V}^{(t+1)})$, $\boldsymbol{V}^{(t+1)} = \boldsymbol{W}^{(t)} - \alpha\nabla_{\boldsymbol{W}}Loss(\boldsymbol{W}^{(t)})$. Let $d_t = \|\boldsymbol{W}^{(t)} - \boldsymbol{W}^{(*)}\|_F$. If $R = O(\frac{\sqrt{m}}{L^6(\log m)^3})$ and loss function satisfies Assumption 3.2, with probability at least $1 - O(nL)e^{-\Omega((mR)^{2/3}L)}$ over random initialization, we have

$$\begin{aligned}
&d_{t+1}^2 = \|\boldsymbol{W}^{(t+1)} - \boldsymbol{W}^{(*)}\|_F^2\\
&\leq \|\boldsymbol{V}^{(t+1)} - \boldsymbol{W}^{(*)}\|_F^2\\
&= \|\boldsymbol{W}^{(t)} - \boldsymbol{W}^{(*)}\|_F^2 + 2\left\langle V_{t+1} - \boldsymbol{W}^{(t)}, \boldsymbol{W}^{(t)} - \boldsymbol{W}^{(*)}\right\rangle\\
&\quad + \|V_{t+1} - \boldsymbol{W}^{(t)}\|_F^2\\
&= d_t^2 + 2\alpha\left\langle\nabla_{\boldsymbol{W}}L'(\boldsymbol{W}^{(t)}), \boldsymbol{W}^{(*)} - \boldsymbol{W}^{(t)}\right\rangle + \alpha^2\|\nabla_{\boldsymbol{W}}L'(\boldsymbol{W}^{(t)})\|_F^2\\
&= d_t^2 + \frac{2\alpha}{n}\sum_{i=1}^{n}\left\langle\nabla_{\boldsymbol{W}}l(f_i(\boldsymbol{W}^{(t)}, X'), y_i), \boldsymbol{W}^{(*)} - \boldsymbol{W}^{(t)}\right\rangle\\
&\quad + \alpha^2\|\frac{1}{n}\sum_{i=1}^{n}\frac{\partial l}{\partial f}\nabla_{\boldsymbol{W}}f_i(\boldsymbol{W}^{(t)}, X')\|_F^2\\
&\leq d_t^2 + \frac{2\alpha}{n}\sum_{i=1}^{n}[l(f_i(\boldsymbol{W}^{(*)}, X'), y_i) - l(f_i(\boldsymbol{W}^{(t)}, X'), y_i)\\
&\quad + \|\boldsymbol{W}^{(*)} - \boldsymbol{W}^{(t)}\|_F O(\frac{R^{1/3}m^{1/3}L^{5/2}\sqrt{\log m}}{\overline{C}_L^f})]\\
&\quad + \alpha^2 O(\frac{mL^2}{(\overline{C}_L^f)^2})\\
&\leq d_t^2 + 2\alpha[Loss(\boldsymbol{W}^{(*)}, X) + O(\hat{C}_L\delta) - Loss(\boldsymbol{W}^{(t)}, X')]\\
&\quad + O(\alpha\frac{R^{4/3}m^{-1/6}L^{5/2}\sqrt{\log m}}{\overline{C}_L^f} + \alpha^2\frac{mL^2}{(\overline{C}_L^f)^2})\\
&\leq d_t^2 + 2\alpha[Loss(\boldsymbol{W}^{(*)}, X) - Loss(\boldsymbol{W}^{(t)}, X')]\\
&\quad + O(\alpha\frac{R^{4/3}m^{-1/6}L^{5/2}\sqrt{\log m}}{\overline{C}_L^f} + \alpha^2\frac{mL^2}{(\overline{C}_L^f)^2} + \alpha\hat{C}_L\delta)
\end{aligned}$$

where the second inequality is for Lemma1.5 and Lemma1.4, the third inequality is due to Lemma1.6. Note that Lemma1.5 and Lemma1.4 can be satisfied with $m = max(\Omega(\frac{L^{16}R^9}{(\delta\overline{C}_L^e)^7}), \Omega(d^2))$.

Using induction, we have

$$\begin{aligned}
d_T^2 \leq& d_0^2 + 2\alpha\sum_{t=0}^{T+1}[Loss(\boldsymbol{W}^{(*)}, X) - Loss(\boldsymbol{W}^{(t)}, X')]\\
&+ O(T(\alpha\frac{R^{4/3}m^{-1/6}L^{5/2}\sqrt{\log m}}{\overline{C}_L^f} + \alpha^2\frac{mL^2}{(\overline{C}_L^f)^2} + \alpha\hat{C}_L\delta))
\end{aligned}$$

which implies that

$$\begin{aligned}
&\min_{0\leq t\leq T}(Loss(\boldsymbol{W}^{(t)}, X') - Loss(\boldsymbol{W}^{(*)}, X))\\
&\leq \frac{d_0^2 - d_T^2}{\alpha T} + O(\frac{R^{4/3}m^{-1/6}L^{5/2}\sqrt{\log m}}{\overline{C}_L^f} + \alpha\frac{mL^2}{(\overline{C}_L^f)^2} + \hat{C}_L\delta)\\
&\leq \frac{R^2}{m\alpha T} + O(\frac{R^{4/3}m^{-1/6}L^{5/2}\sqrt{\log m}}{\overline{C}_L^f} + \alpha\frac{mL^2}{(\overline{C}_L^f)^2} + \hat{C}_L\delta)\\
&\leq O(\hat{C}_L\delta)
\end{aligned}$$

Where the last inequality is a result of parameter selection, specifically $\alpha = O(\frac{\delta \overline{C}_L^e \overline{C}_L^f}{mL^2})$, $T = \Theta(\frac{R^2}{m\alpha\delta\hat{C}_L})$ and $m = \Omega(\frac{L^{16}R^9}{(\delta\overline{C}_L^e)^7})$.

Given $\epsilon > 0$, suppose $R = \Omega(1)$, $m = max(\Omega(\frac{L^{16}}{(\delta\overline{C}_L^e)^7}), \Omega(d^2))$ and $\delta \leq \frac{\epsilon}{\hat{C}_L}$. Let the loss function satisfy Assumption 3.2, if we run projected gradient descent based on the convex constraint set $B(R)$ with step size $\alpha = O(\frac{\epsilon(\overline{C}_L^f)^2}{mL^2})$ for $T = \Theta(\frac{R^2}{m\alpha\epsilon})$ steps, with high probability we have

$$\min_{0\leq t\leq T}(Loss(\boldsymbol{W}^{(t)}, X') - Loss(\boldsymbol{W}^{(*)}, X)) \leq O(\hat{C}_L\delta) \leq O(\epsilon)$$

where $\boldsymbol{W}^{(*)} = \arg\min_{\boldsymbol{W}\in B(R)} Loss(\boldsymbol{W}, X)$.

The selection of m, α, and δ within the above analysis relies on factors decided by graph properties. For simplicity we can directly use the bounds of these factors so that $m = max(\Omega(\frac{L^{16}R^9n^7}{\epsilon^7}), \Omega(d^2))$ and $\alpha = O(\frac{\epsilon}{mL^2n^2})$. The detailed analysis of these factors are in Section 1.5.

1.5 Impacts of Graph Structure and Model Depth on Compression Ratios

To train a GNN model over a large graph data set, the key to using feature quantization is to choose a proper compression ratio to strike a balance between the saved memory and PCIe bandwidth consumption and the accuracy. With the previous analysis of loss bound, combined with our experiments, we identify that the major contributor to the loss bound (see Theorem 3.3) is $\delta\hat{C}_L$. In particular, given an expected loss bound, the factor $\hat{C}_L$ is inversely proportional to δ, where δ is roughly $\Theta(2^{-32/CR})$. Therefore, we would expect the compression ratio to be negatively correlated to the $\hat{C}_L$. *Since $\hat{C}_L$ is relevant to both model depth, i.e., number of layers L, and the structure of the graph data set, we further study the impacts of the two factors on the selection of compression ratios.*

Model depth. We begin our analysis with understanding the impact of number of GNN layers. We test the values of $\overline{C}_L^f$ and $\overline{C}_L^e$ with the different numbers of layers on Reddit dataset. As expected, the values of $\overline{C}_L^f$, $\overline{C}_L^e$, and $\hat{C}_L$ decrease when we increase the number of layers, as shown in Table2. The decrease in $\hat{C}_L$ is mainly because $\overline{C}_L^f$ decreases much slower than $\overline{C}_L^e$.

Table 2: Impacts of model depth (the number of layers). $\hat{C}_L$ is the factor deciding acceptable compression ratio (smaller is better).

	Number of layers (L)				
Factor	2	3	5	10	20
$\overline{C}_L^f$	0.1216	0.1095	0.0983	0.0804	0.0691
$\overline{C}_L^e$	0.0152	0.0086	0.0047	0.0026	0.0018
$\hat{C}_L$	0.1248	0.0788	0.0482	0.0325	0.0258

Graph structure. Then, to examine the impact of graph structure, we artificially create graph data sets with different structures based on the Reddit data set. Basically, we keep the Reddit nodes but delete some of the edges with three different selection methods to simulate graphs with varied sparsity, resulting in three types of data sets, namely, RANDOM, CENTRALIZED, and UNIFORM. In more detail, RANDOM deletes randomly selected edges, and thus keeps the power-law distribution of node degrees. CENTRALIZED deletes edges between low-degree nodes with high priority, leading the edges in the revised graph to be more centralized on high-degree nodes than that of the original graph. In contrast, UNIFORM prioritizes to delete edges between high-degree nodes, and make the resulting graph to follow an almost uniform degree distribution.

Table 3 shows the values of $\hat{C}_L$ with different graph structures under varied sparsity. Here, sparsity relates to the number of edges, i.e., more edges, denser graph, and vice versa. We also list the values of $\overline{C}_L^f$ and $\overline{C}_L^e$, as $\hat{C}_L = \overline{C}_L^e/\overline{C}_L^f$. Furthermore, *CR* denotes the maximum compression ratio with less than 0.1% accuracy degradation. Note that, here, we only apply scalar quantization for the analysis, where the maximum CR is 32.

We draw the following conclusions from Table 3. First, across the three data sets, denser graphs (higher percentage of remaining edges, in comparison to the original Reddit data set) always have smaller $\hat{C}_L$. Therefore, their compression ratios can be higher. With the same sparsity (the same percentage of remaining edges), the graphs created by the CENTRALIZED method have the following properties: some nodes are isolated, and the remaining nodes are connected by several super nodes. The direct consequence is to have a higher $\hat{C}_L$ and low acceptable compression ratio. Contrary, UNIFORM has a lower $\hat{C}_L$ and higher compression ratio. The reasons are as follows. The existence of super nodes makes the quantization errors spread to a large amount of nodes, which however can hardly be cancelled out during the aggregation, because neighbors of each node mainly have this component. But, the UNIFORM method addresses this limitation by eliminating super nodes and keeping each node's neighborhoods less overlapped.

In summary, the above results indicate that $\mathsf{F}^2\mathsf{CGT}$ training is especially useful to handle deeper GNNs and graphs with degree distribution that is not too centralized.

2 ACCURACY EVALUATION

In this section, we present additional accuracy results about $\mathsf{F}^2\mathsf{CGT}$ for mainstream GNN tasks, encompassing node classification, link prediction, and graph classification.

2.1 Experiment Setup

We conducted our accuracy validation experiments on a multi-GPU server hosted on an AWS g4dn.metal instance. This server is equipped with eight NVIDIA T4 GPUs, each having 16 GB of memory, along with 96-core vCPU with clock speed of 2.5 GHz (Cascade Lake 24C) and 384 GB of RAM. Our experimental setup utilizes PyTorch 1.12.1 [12] and DGL 0.9.1 [5].

We train five GNN models to fulfill three tasks, where we use GraphSAGE[7], GAT[15], and ClusterGCN[3] for node classification, GraphSAGE and GCN[8] for link prediction, and GCN and GIN[16] for graph classification. We exercise nine graphs, from MUTAG[4] to MAG240M[9], the largest open-source graph dataset. Table 4 shows the statistics of all the nine graph datasets.

We use the original DGL GNN training with no feature quantization as the natural baseline, denoted as "Full", with an optimal

Table 3: Impact of graph structures. CR means maximum acceptable compression ratio.

Method to extract graph		Percentage of remaining edges. 10%	30%	50%	70%	90%	100%
RANDOM	$\overline{C}_L^f$	0.1329	0.1141	0.1147	0.1116	0.1050	0.1098
	$\overline{C}_L^e$	0.0356	0.0171	0.0127	0.0105	0.0092	0.0088
	$\hat{C}_L$	0.2677	0.1500	0.1109	0.0940	0.0875	0.0799
	CR	16	16	32	32	32	32
CENTRALIZED	$\overline{C}_L^f$	0.4013	0.3789	0.3427	0.2836	0.1854	
	$\overline{C}_L^e$	0.3873	0.3423	0.2924	0.2166	0.0913	
	$\hat{C}_L$	0.9650	0.9034	0.8548	0.7638	0.4923	
	CR	4	4	8	16	16	
UNIFORM	$\overline{C}_L^f$	0.1109	0.1111	0.1113	0.1022	0.1134	
	$\overline{C}_L^e$	0.0169	0.0104	0.0089	0.0086	0.0087	
	$\hat{C}_L$	0.1523	0.0938	0.0802	0.0845	0.0770	
	CR	32	32	32	32	32	

Table 4: Statistics of graph datasets. K, M, and B stand for thousand, million, and billion, respectively.

Datasets	Task	#Graphs	#Avg Nodes	#Avg Edges	#Labels	#Features
Reddit[7]	Node	1	233.0K	11.6M	41	602
Papers100M[10]	Node	1	111.1M	1.6B	172	128
MAG240M [13]	Node	1	244.2M	3.4B	153	768
OGBL-COLLAB [10]	Link	1	235.9K	1.3M	-	128
OGBL-PPA [10]	Link	1	576.3K	30.3M	-	58
MUTAG [4]	Graph	188	17.9	57.5	2	7
PTC [14]	Graph	344	25.6	77.5	2	19
PROTEINS [2]	Graph	1113	39.1	184.7	2	3
COLLAB [17]	Graph	5000	74.5	4989.5	3	367

Table 5: Training accuracy and performance of node classification tasks. CR is compression ratio, Acc is the test set accuracy. ClusterGCN raises out of CPU memory error when partitioning and normalizing MAG240M.

Model	Method	Reddit CR	Reddit Acc (%)	OGBN-Papers100M CR	OGBN-Papers100M Acc (%)	MAG240M CR	MAG240M Acc (%)
GraphSAGE[7]	Full	-	96.3	-	66.1	-	68.6
	SQ	32	96.4	32	65.4	32	68.5
	VQ	163.2	96.1	46.5	65.4	46.5	68.3
	ML	520.0	96.3	62.2	65.8	63.0	68.4
GAT[15]	Full	-	95.3	-	65.8	-	67.8
	SQ	32	95.4	32	64.9	32	67.7
	VQ	163.2	95.3	46.5	65.3	46.5	67.7
	ML	520.0	95.3	62.2	65.6	63.0	67.6
ClusterGCN[3]	Full	-	96.0	-	63.3	OOM	
	SQ	32	96.0	32	62.8		
	VQ	148	95.5	46.5	62.7		
	ML	69.8	95.8	37.4	62.9		

hyperparameter settings. We run $\texttt{F}^2\texttt{CGT}$ with single scalar quantization denoted as "SQ", $\texttt{F}^2\texttt{CGT}$ with single vector quantization as "VQ" and $\texttt{F}^2\texttt{CGT}$ with two-level compression as "ML". All baselines

adopts the same model configuration. The compression ratio for F^{2}CGT-SQ is set at 32, the maximum achievable value. However, for F^{2}CGT-VQ, the compression ratio may vary depending on the models and datasets. Therefore, we report the maximum compression ratios with marginal accuracy drop compared to the "Full" baseline. For F^{2}CGT-ML, the VQ setup follows the same rule and is typically more aggressive. We obtain the accuracy results by running three times and averaging the best test accuracy of each run.

2.2 Node Property Prediction

We assess the influence of F^{2}CGT on node property prediction tasks using three datasets: Reddit, OGBN-Papers100M, and MAG240M. Despite Reddit's relatively small size, it serves as a commonly used benchmark for accuracy validation. The other two datasets are characterized by large-scale graphs. For the MAG240M heterogeneous graph, we convert it into an undirected homogeneous graph. Additionally, we enhance its feature information by averaging features from neighboring nodes for author and institution nodes. The MAG240M dataset, after this transformation, amounts to approximately 400GB.

Table 5 provides an overview of the accuracy results and compression ratios achieved by the three variants of F^{2}CGT, in comparison to the "Full" baseline. We have identified several key findings: First, all three compression methods yield comparable accuracy to the "Full" baseline, with the maximum accuracy drop remaining below 1%. Second, F^{2}CGT-SQ consistently achieves a maximum compression ratio of 32, as it binarizes 32-bit float features. In contrast, F^{2}CGT-VQ can achieve significantly higher compression ratios, reaching up to 163.2. Third, For models such as GraphSAGE and GAT, F^{2}CGT-ML outperforms both simple compression methods. F^{2}CGT-ML not only achieves higher compression ratios than F^{2}CGT-VQ but also exhibits superior accuracy performance in some cases. In the case of ClusterGCN, F^{2}CGT-ML exhibits a relatively lower compression ratio due to ClusterGCN's subgraph sampling approach, which includes a larger proportion of seed nodes. Nevertheless, it still achieves higher accuracy performance compared to F^{2}CGT-VQ. Lastly, the same compression setup can perform well across different models. Therefore, compressed features can be efficiently reused in multiple training runs without the need for repeated compression setup tuning.

These results, combined with the theoretical analysis, demonstrate that GNN training can typically maintain the expected accuracy even with aggressive compression of input data.

Comparison to non-GNN model. We additionally conducted an evaluation to gauge the impact of F^{2}CGT on a non-GNN model. Specifically, we employed a 3-layer MLP with a structure and number of parameters identical to the GraphSAGE model used in the tests but without any aggregation operations. According to previous theoretical analysis, the factor $\hat{C}_L$ is vital to the canceling out of compression errors, and MLP has $\hat{C}_L$ of exactly 1, meaning the impact of error is fully passed to the output of the model. As expected, the MLP model exhibited a notable accuracy degradation ranging from 6% to 11%. When compared with GNN results, this suggests that F^{2}CGT is specifically tailored for GNNs, and only models with aggregation mechanisms like GNNs are compatible with F^{2}CGT.

Table 6: Link prediction accuracy

Model	Method	OGBL-PPA CR	OGBL-PPA Hits@100(%)	OGBL-COLLAB CR	OGBL-COLLAB Hits@50(%)
GCN[11]	Full	-	18.4	-	50.6
	SQ	32	18.5	32	49.0
	VQ	46.5	18.9	46.5	49.2
GraphSAGE	Full	-	16.4	-	49.4
	SQ	32	16.3	32	50.4
	VQ	46.5	16.5	46.5	49.3

2.3 Link Property Prediction.

For link property prediction, we run link prediction tasks, which are to predict existence of edges (i.e. pairs of nodes). We utilized two graph datasets, namely, OGBL-PPA and OGBL-COLLAB. OGBL-PPA represents a protein-protein association network, structured as an undirected, unweighted graph. Nodes in this graph represent different proteins from 58 species, and edges indicate biological associations between proteins. OGBL-COLLAB, on the other hand, is an author collaboration network, also organized as an undirected graph. Our training involved GCN [8] and GraphSAGE models on these datasets, and the prediction results are presented in Table 6.

Similar to the node property prediction tasks, using both F^{2}CGT-SQ and F^{2}CGT-VQ yielded comparable link prediction accuracy (Hits@100 in OGBL-PPA, Hits@50 in OGBL-COLLAB) as the full precision baseline. Once again, F^{2}CGT-VQ demonstrated a superior compression ratio, maintaining a constant value of 46.5 (after experimenting with various VQ setups, this configuration consistently performed well for all tasks), representing a 45% increase over F^{2}CGT-SQ.

2.4 Graph Property Prediction.

Concerning the graph property prediction tasks, we trained GCN and GIN [16] on four datasets: PTC, MUTAG, PROTEINS, and COLLAB. The first three datasets are bio-informatics datasets, while the last one is a social network dataset. It's noteworthy that the features of these datasets are one-hot encoded, meaning each feature vector can be represented by a single integer, indicating the index of the only non-zero element in the vector. Clearly, when applied to such features, F^{2}CGT-SQ is lossless and behaves exactly the same as the full precision baseline. Therefore, we omit its results from Table 7.

On the other hand, since one-hot codes theoretically have a high lossless compression ratio, to test the accuracy impact of vector quantization while stressing F^{2}CGT, we set a higher compression ratio than the lossless setting. Doing so forces F^{2}CGT-VQ to lose some information during quantization. Table 7 demonstrates impressively that F^{2}CGT-VQ can actually enhance training accuracy. Based on our observations, vector quantization extracts valuable information and filters out noise, thereby improving model robustness.

2.5 Hyperparameter Setups

In this section, we show the detailed hyperparameter setups corresponding to experiments presented

Table 7: Graph classification accuracy

		PTC		MUTAG		PROTEINS		COLLAB	
Model	Method	CR	Acc(%)	CR	Acc(%)	CR	Acc(%)	CR	Acc(%)
GCN	Full	-	61.3	-	85.6	-	74.4	-	82.9
	VQ	152	63.4	112	87.3	136	74.0	1468	83.1
GIN	Full	-	64.1	-	87.7	-	73.6	-	81.8
	VQ	152	65.1	112	87.8	136	73.6	1468	82.2

Table 8: Experimental setups for the node property prediction tasks

		Datasets		
Model	Hyperparameters	Reddit	OGBN-Papers100M	MAG240M
GraphSAGE	#layer	2	3	3
	#hidden	64	256	256
	fan out	10,25	5,10,15	5,10,15
	SQ	1	1	1
	VQ	100-16384	16-2048	16-2048
	ML	2,301-8192	2,301-8192	4,151-8192
GAT	#layer	2	3	3
	#hidden	64	256	256
	fan out	10,25	5,10,15	5,10,15
	drop out	0.2	0.25	0.25
	lr	0.003	0.005	0.001
	SQ	1	1	1
	VQ	51-1024	16-16384	16-2048
	ML	8,16-256	8,16-256	8,16-256
ClusterGCN	#hidden	128	256	-
	psize	1000	1000	-
	batch size	100	20	-
	dropout	0.25	0.1	-
	lr	0.05	0.001	-
	SQ	1	1	-
	VQ	37-256	16-2048	-
	ML	4,16-256	4,16-256	-

Table 9: Experimental setups for the link property prediction tasks

		Datasets	
Model	Hyperparameters	OGBL-COLLAB	OGBL-PPA
GCN	#layer	3	3
	#hidden	256	256
	lr	0.001	0.01
	SQ	1	1
	VQ	16-2048	16-2048
GraphSAGE	#layer	3	3
	#hidden	256	256
	lr	0.001	0.01
	SQ	1	1
	VQ	16-2048	16-2048

Table 10: Experimental setups for the graph property prediction tasks

Model	Hyperparameters	Datasets PTC	MUTAG	PROTEINS	COLLAB
GCN	#GNN layer	5	5	5	5
	#MLP layer	2	2	2	2
	#hidden	64	64	64	64
	degree as label	false	false	false	true
	graph pooling type	sum	sum	sum	mean
	VQ	19-16	7-4	17-16	367-256
GIN[16]	#GNN layer	5	5	5	5
	#MLP layer	2	2	2	2
	#hidden	64	64	64	64
	degree as label	false	false	false	true
	graph pooling type	sum	sum	sum	mean
	VQ	19-16	7-4	17-16	367-256

Node Property Prediction. For GraphSAGE, GAT, and ClusterGCN, we use the example code from DGL[5]. We changed GAT implementation to make it support neighbor sampling and able to train on giant graphs.

ClusterGCN's default implementation uses the subgraph of train set nodes for training, which has poor performance in OGBN-Papers100M. We change this implementation by using the subgraph including train set nodes' 1-hop neighbors.

For MLP model, we use a three-layer MLP with ReLU activation and dropout, where its hidden layer dimension is 256.

The detailed hyperparameters, including F^2CGT-SQ and F^2CGT-VQ setups, are listed in Table 8. Not listed hyperparameters use the default value.

The listed SQ parameter means the F^2CGT-SQ setup used in the test. Here all tests are done with this parameter as 1, meaning we only use the sign bit to indicate the original float32 number. The VQ parameter includes two numbers, *width* and *length*, are shown with the format *width* - *length*. These parameters mean the number of dimensions of each codebook entry and the number of entries in each codebook.

Link Property Prediction. We test GCN and GraphSAGE with the OGB example code. Two models both use the default setting. The detailed setups are shown in Table9 and Table10, with the same manner as node property prediction setups.

Graph Property Prediction. We used the DGL example code of GIN and added GCN implementation. In our tests, GCN uses the same hyperparameter setup as GIN. The detailed setups are shown in Table10. Due to the original features being one-hot codes, we didn't test F^2CGT-SQ, and we used a much more aggressive F^2CGT-VQ setup.

REFERENCES

[1] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. 2019. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*. PMLR, 242–252.

[2] Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. 2005. Protein function prediction via graph kernels. *Bioinformatics* 21, suppl_1 (2005), i47–i56.

[3] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 257–266.

[4] Asim Kumar Debnath, Rosa L Lopez de Compadre, Gargi Debnath, Alan J Shusterman, and Corwin Hansch. 1991. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry* 34, 2 (1991), 786–797.

[5] Team DGL. 2023. DGL Homepage. https://www.dgl.ai/. [Online; accessed Dec-2023].

[6] Ruiqi Gao, Tianle Cai, Haochuan Li, Cho-Jui Hsieh, Liwei Wang, and Jason D Lee. 2019. Convergence of adversarial training in overparametrized neural networks. *Advances in Neural Information Processing Systems* 32 (2019).

[7] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[8] Negar Heidari and Alexandros Iosifidis. 2021. Progressive Graph Convolutional Networks for Semi-Supervised Node Classification. *IEEE Access* 9 (2021), 81957–81968. https://doi.org/10.1109/ACCESS.2021.3085905

[9] Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. 2021. OGB-LSC: A Large-Scale Challenge for Machine Learning on Graphs. *arXiv preprint arXiv:2103.09430* (2021).

[10] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems* 33 (2020), 22118–22133.

[11] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[12] Team PyTorch. 2023. Pytorch Homepage. https://pytorch.org/. [Online; accessed Dec-2023].

[13] OGB Team. 2023. MAG240M. https://ogb.stanford.edu/docs/lsc/mag240m/. accessed, Dec-2023.

[14] Hannu Toivonen, Ashwin Srinivasan, Ross D King, Stefan Kramer, and Christoph Helma. 2003. Statistical evaluation of the predictive toxicology challenge 2000–2001. *Bioinformatics* 19, 10 (2003), 1183–1193.

[15] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[16] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *International Conference on Learning Representations*. https://openreview.net/forum?id=ryGs6iA5Km

[17] Pinar Yanardag and SVN Vishwanathan. 2015. A structural smoothing framework for robust graph comparison. *Advances in neural information processing systems* 28 (2015).