A. Supplementary Materials

A.1. Theoretical Proof

In this paper, we use ∞-norm for both weight and gradient in our 1-Lipschitz neural network. For ease representation, we use $\|\cdot\|$ to replace $\|\cdot\|_\infty$ in our 1-Lipschitz neural network.

Lemma 1 *[Lagrange's Mean Value Theorem] For any continuous function f on the closed interval $[a,b]$ and differentiable on the open interval (a,b), then there exists a point c in (a,b) such that the tangent at c is parallel to the secant line through the endpoints $(a,f(a))$ and $(b,f(b))$.*

$$f'(c) = \frac{f(a)-f(b)}{a-b} \tag{14}$$

Proof. Please refer to the book (Sharma & Vasishtha, 2010) for detailed proof.

Theorem 1 *For any K_l-Lipschitz nonlinear activation function $\bar{f}:\mathbb{R}^N \mapsto \mathbb{R}^N$, if $\bar{f}$ is everywhere differentiable and $\bar{f}(\mathbf{x}) = \mathbf{0} \in \mathbb{R}^N$ at $\mathbf{x} = \mathbf{0} \in \mathbb{R}^N$, then there must exist a linear function g such that $\bar{f}(\mathbf{x}) \le g(\mathbf{x})$ for $\forall \mathbf{x}, \mathbf{x} \in \mathbb{R}^N$.*

Proof. The proof of this theorem can be divided into two cases: scalar and vector cases.

(Scalar case.) According to Definition 1 and Lemma 1, given large enough M, the domain $(-\infty,+\infty)$ can be partitioned into M subintervals $P = \{[x_0,x_1],[x_1,x_2],\cdots,[x_{m-1},x_m],\cdots,[x_{M-1},x_M]\}$, where $-\infty = x_0 < x_1 < x_2 < \cdots < x_{m-1} < x_m < \cdots < x_{M-1} < x_M = +\infty$, such that any K_l-Lipschitz nonlinear activation function $\bar{f}:\mathbb{R}\mapsto\mathbb{R}$ on $(-\infty,+\infty)$ become linear or near-linear on each subinterval. Therefore, $\bar{f}(x)$ can be rewrite as the piecewise function $\bar{f}'(\hat{x}_i)x$ w.r.t. variable x on M subintervals P, i.e., each piecewise function is approximately equivalent to a linear function with the slope of $\bar{f}'(\hat{x}_i)$ through the origin.

$$\bar{f}(x) \approx \sum_{i=1}^{M} \bar{f}'(\hat{x}_i)x \tag{15}$$

where $\hat{x}_i \in (x_{i-1}, x_i)$ and

$$\bar{f}'(\hat{x}_i) = \begin{cases} \dfrac{\bar{f}(x_i)-\bar{f}(x_{i-1})}{x_i - x_{i-1}}, & \text{if } x \in [x_{i-1}, x_i], \\ 0, & \text{if } x \notin [x_{i-1}, x_i]. \end{cases} \tag{16}$$

Since $\bar{f}$ is K_l-Lipschitz on $\mathbb{R}$, we have the following inequality for any $x_1, x_2 \in \mathbb{R}$.

$$\bar{f}' = \frac{\bar{f}(x_1)-\bar{f}(x_2)}{x_1-x_2} \le \frac{|\bar{f}(x_1)-\bar{f}(x_2)|}{|x_1-x_2|} \le K_l \tag{17}$$

Thus, there exist a linear function $g(x) = K_l x$ such that

$$\bar{f}(x) = \sum_{i=1}^{M} \bar{f}'(\hat{x}_i)x \le K_l x = g(x) \tag{18}$$

(Vector case.) For any K_l-Lipschitz nonlinear activation function $\bar{f}:\mathbb{R}^N \mapsto \mathbb{R}^N$ on $(-\infty,+\infty)$ w.r.t. infinity norm, we can follow the same strategy in the scalar case to partition the domain of each component in an N-dimensional vector $\mathbf{x}$ into M subintervals, such that $\bar{f}(\mathbf{x})$ become linear or near-linear on each subinterval of each component. Similarly, $\bar{f}(\mathbf{x})$ can be rewrite as the piecewise function $\nabla_{\hat{\mathbf{x}}_i}\bar{f}\cdot\mathbf{x}$ w.r.t. M subintervals of each component.

$$\bar{f}(\mathbf{x}) \approx \sum_{i=1}^{M} \nabla_{\hat{\mathbf{x}}_i}\bar{f}\cdot\mathbf{x} \tag{19}$$

Since $\bar{f}$ is gradient infinity-norm preservation, we have

$$\begin{aligned} \bar{f}(\mathbf{x}) &\approx \sum_{i=1}^{M} \nabla_{\hat{\mathbf{x}}_i}\bar{f}\cdot\mathbf{x} \\ &= \sum_{i=1}^{M} \Big[\frac{\partial\bar{f}}{\partial\hat{\mathbf{x}}_{i1}},\cdots,\frac{\partial\bar{f}}{\partial\hat{\mathbf{x}}_{iN}}\Big]\cdot\mathbf{x} \\ &\le \sum_{i=1}^{M} \Big[|\frac{\partial\bar{f}}{\partial\hat{\mathbf{x}}_{i1}}|,\cdots,|\frac{\partial\bar{f}}{\partial\hat{\mathbf{x}}_{iN}}|\Big]\cdot\mathbf{x} \\ &\le \sum_{i=1}^{M} \max\Big\{|\frac{\partial\bar{f}}{\partial\hat{\mathbf{x}}_{i1}}|,\cdots,|\frac{\partial\bar{f}}{\partial\hat{\mathbf{x}}_{iN}}|\Big\}\cdot\mathbf{x} \\ &= \sum_{i=1}^{M} \|\nabla_{\hat{\mathbf{x}}_i}\bar{f}\|_\infty\mathbf{x} \\ &= K_l\mathbf{x} = g(\mathbf{x}) \end{aligned} \tag{20}$$

Therefore, the proof is concluded.

Theorem 2 *By following the definition in Eq.(2), we build a $(L+1)$-layer 1-Lipschitz neural network $\bar{F}:\mathbb{R}^{N_0}\mapsto\mathbb{R}^{N_L}$, with a gradient norm preserving activation function $\|\nabla_{\bar{\mathbf{z}}_{l-1}}\bar{f}\| = K_l$ almost everywhere and a norm-constrained weight matrix $\|\bar{\mathbf{W}}_l\| = 1/K_l$ like the definitions in Eq.(7). If $K_l > 1$ for $\forall l, 2 \le l \le L$, our 1-Lipschitz neural network $\bar{F}$ achieves better expressive power than the neural network $\tilde{F}$ constructed by the GroupSort model (Anil et al., 2019; Cohen et al., 2019).*

Proof. Let $F:\mathbb{R}^{N_0}\mapsto\mathbb{R}^{N_L}$ be a regular $(L+1)$-layer fully connected neural network with unconstrained ReLU as the activation f and unrestricted weight matrix $\mathbf{W}_l$ at layer l, $\tilde{F}$ be a $(L+1)$-layer 1-Lipschitz neural network with

1-Lipschitz GroupSort activation $\tilde{f}$ and norm constrained weight matrix $\tilde{\mathbf{W}}_l$ proposed by the GroupSort model (Anil et al., 2019; Cohen et al., 2019), and $\bar{F} : \mathbb{R}^{N_0} \mapsto \mathbb{R}^{N_L}$ be our $(L+1)$-layer 1-Lipschitz neural network with K_l-Lipschitz Weibull activation $\bar{f}$ and norm-contained weight matrix $\bar{\mathbf{W}}_l$.

Namely, for F, $\|f(\mathbf{z}_l)\|_\infty = \|ReLU(\mathbf{z}_l)\|_\infty \leq \|\mathbf{z}_l\|_\infty$.

For $\tilde{F}$, $\|\tilde{f}(\tilde{\mathbf{z}}_l)\|_\infty = \|\tilde{\mathbf{z}}_l\|_\infty$ as $\tilde{f}$ only sorts and permutates the elements in each $\tilde{\mathbf{z}}_l$ but do not change the component values. In addition, $\|\nabla_{\tilde{\mathbf{z}}_{l-1}} \tilde{f}\| = 1$ and $\|\tilde{\mathbf{W}}_l\| = 1$.

For $\bar{F}$, $\|\nabla_{\bar{\mathbf{z}}_{l-1}} \bar{f}\| = K_l$, $\|\bar{\mathbf{W}}_l\| = 1$ if $l = 1$, and $\|\bar{\mathbf{W}}_l\| = 1/K_l$ if $l > 1$.

By using the triangle inequality and submultiplicativity of matrix norms, the output difference between the 1-Lipschitz neural network $\tilde{F}$ by the GroupSort model and the regular fully connected neural network F is calculated below.

$$
\begin{aligned}
&\|\tilde{F}(\mathbf{x}) - F(\mathbf{x})\| \\
=&\|\tilde{\mathbf{W}}_L \tilde{f}(\tilde{\mathbf{z}}_{L-1}) + \mathbf{b}_L - (\mathbf{W}_L f(\mathbf{z}_{L-1}) + \mathbf{b}_L)\| \\
=&\|\tilde{\mathbf{W}}_L [\tilde{f}(\tilde{\mathbf{z}}_{L-1}) - ReLU(\mathbf{z}_{L-1})] + \\
&\tilde{\mathbf{W}}_L ReLU(\mathbf{z}_{L-1}) - \mathbf{W}_L f(\mathbf{z}_{L-1})\| \\
\leq&\|\tilde{\mathbf{W}}_L\| \|\tilde{f}(\tilde{\mathbf{z}}_{L-1}) - ReLU(\mathbf{z}_{L-1})\| + \\
&\|ReLU(\mathbf{z}_{L-1})\| \|\tilde{\mathbf{W}}_L - \mathbf{W}_L\| \\
=&\|\tilde{f}(\tilde{\mathbf{z}}_{L-1}) - ReLU(\mathbf{z}_{L-1})\| + \\
&\|ReLU(\mathbf{z}_{L-1})\| \|\tilde{\mathbf{W}}_L - \mathbf{W}_L\| \\
\leq&\|\tilde{f}(\tilde{\mathbf{z}}_{L-1})\| + \|ReLU(\mathbf{z}_{L-1})\| + \\
&\|ReLU(\mathbf{z}_{L-1})\| (\|\tilde{\mathbf{W}}_L\| + \|\mathbf{W}_L\|) \\
\leq&\|\tilde{\mathbf{z}}_{L-1}\| + \|\mathbf{z}_{L-1}\| + \|\mathbf{z}_{L-1}\| (\|\tilde{\mathbf{W}}_L\| + \|\mathbf{W}_L\|)
\end{aligned} \tag{21}
$$

By following the same strategy used for the K_l-Lipschitz nonlinear activation function $\bar{f}$ in Theorem 1, we have

$$
\begin{aligned}
&\|\tilde{F}(\mathbf{x}) - F(\mathbf{x})\| \\
\leq&\|\tilde{\mathbf{z}}_{L-1}\| + \|\mathbf{z}_{L-1}\| + \|\mathbf{z}_{L-1}\| (\|\tilde{\mathbf{W}}_L\| + \|\mathbf{W}_L\|) \\
\approx&\Big\| \sum_{i=1}^{M} \nabla_{\hat{\mathbf{z}}_{1i}} \tilde{\mathbf{z}}_{L-1} \cdot \tilde{\mathbf{z}}_1 \Big\| + \|\mathbf{z}_{L-1}\| + \\
&\|\mathbf{z}_{L-1}\| (\|\tilde{\mathbf{W}}_L\| + \|\mathbf{W}_L\|) \\
\leq&\Big\| \sum_{i=1}^{M} \nabla_{\hat{\mathbf{z}}_{1i}} \tilde{\mathbf{z}}_{L-1} \Big\| \|\tilde{\mathbf{z}}_1\| + \|\mathbf{z}_{L-1}\| + \\
&\|\mathbf{z}_{L-1}\| (\|\tilde{\mathbf{W}}_L\| + \|\mathbf{W}_L\|) \\
=&\|\tilde{\mathbf{z}}_1\| + \|\mathbf{z}_{L-1}\| + \|\mathbf{z}_{L-1}\| (\|\tilde{\mathbf{W}}_L\| + \|\mathbf{W}_L\|) \\
=&\|\tilde{\mathbf{W}}_1 \mathbf{x} + \mathbf{b}_1\| + \|\mathbf{z}_{L-1}\| + \\
&\|\mathbf{z}_{L-1}\| (\|\tilde{\mathbf{W}}_L\| + \|\mathbf{W}_L\|) \\
\leq&\|\tilde{\mathbf{W}}_1\| \|\mathbf{x}\| + \|\mathbf{b}_1\| + \|\mathbf{z}_{L-1}\| + \\
&\|\mathbf{z}_{L-1}\| (\|\tilde{\mathbf{W}}_L\| + \|\mathbf{W}_L\|) \\
=&\|\mathbf{x}\| + \|\mathbf{b}_1\| + \|\mathbf{z}_{L-1}\| + \|\mathbf{z}_{L-1}\| (\|\tilde{\mathbf{W}}_L\| + \|\mathbf{W}_L\|) \\
=&\tilde{\epsilon}
\end{aligned} \tag{22}
$$

Similarly, the output difference between the our 1-Lipschitz neural network $\bar{F}$ and the regular fully connected neural network F is computed as follows.

$$
\begin{aligned}
&\|\bar{F}(\mathbf{x}) - F(\mathbf{x})\| \\
=&\|\bar{\mathbf{W}}_L \bar{f}(\bar{\mathbf{z}}_{L-1}) + \mathbf{b}_L - (\mathbf{W}_L f(\mathbf{z}_{L-1}) + \mathbf{b}_L)\| \\
=&\|\bar{\mathbf{W}}_L [\bar{f}(\bar{\mathbf{z}}_{L-1}) - ReLU(\mathbf{z}_{L-1})] + \\
&\bar{\mathbf{W}}_L ReLU(\mathbf{z}_{L-1}) - \mathbf{W}_L f(\mathbf{z}_{L-1})\| \\
\leq&\|\bar{\mathbf{W}}_L\| \|\bar{f}(\bar{\mathbf{z}}_{L-1}) - ReLU(\mathbf{z}_{L-1})\| + \\
&\|ReLU(\mathbf{z}_{L-1})\| \|\bar{\mathbf{W}}_L - \mathbf{W}_L\| \\
=&\frac{1}{K_L} \|\bar{f}(\bar{\mathbf{z}}_{L-1}) - ReLU(\mathbf{z}_{L-1})\| + \\
&\|ReLU(\mathbf{z}_{L-1})\| \|\bar{\mathbf{W}}_L - \mathbf{W}_L\| \\
\leq&\frac{1}{K_L} (\|\bar{f}(\bar{\mathbf{z}}_{L-1})\| + \|ReLU(\mathbf{z}_{L-1})\|) + \\
&\|ReLU(\mathbf{z}_{L-1})\| (\|\bar{\mathbf{W}}_L\| + \|\mathbf{W}_L\|) \\
\leq&\frac{1}{K_L} (\|\bar{f}(\bar{\mathbf{z}}_{L-1})\| + \|\mathbf{z}_{L-1}\|) + \\
&\|\mathbf{z}_{L-1}\| (\|\bar{\mathbf{W}}_L\| + \|\mathbf{W}_L\|)
\end{aligned} \tag{23}
$$

By using the same strategy used in Theorem 1, we get

$$
\begin{aligned}
&\|\tilde{F}(\mathbf{x}) - F(\mathbf{x})\| \\
\leq &\frac{1}{K_L}(\|\bar{f}(\bar{\mathbf{z}}_{L-1})\| + \|\mathbf{z}_{L-1}\|) + \\
&\|\mathbf{z}_{L-1}\|(\|\bar{\mathbf{W}}_L\| + \|\mathbf{W}_L\|) \\
\approx &\frac{1}{K_L}\Big(\Big\|\sum_{i=1}^{M}\nabla_{\hat{\mathbf{z}}_{(L-1)i}}\bar{f}\cdot\bar{\mathbf{z}}_{L-1}\Big\| + \|\mathbf{z}_{L-1}\|\Big) + \\
&\|\mathbf{z}_{L-1}\|(\|\bar{\mathbf{W}}_L\| + \|\mathbf{W}_L\|) \\
\leq &\frac{1}{K_L}\Big(\Big\|\sum_{i=1}^{M}\nabla_{\hat{\mathbf{z}}_{(L-1)i}}\bar{f}\Big\|\|\bar{\mathbf{z}}_{L-1}\| + \|\mathbf{z}_{L-1}\|\Big) + \\
&\|\mathbf{z}_{L-1}\|(\|\bar{\mathbf{W}}_L\| + \|\mathbf{W}_L\|) \\
\leq &\frac{1}{K_L}(K_L\|\bar{\mathbf{z}}_{L-1}\| + \|\mathbf{z}_{L-1}\|) + \\
&\|\mathbf{z}_{L-1}\|(\|\bar{\mathbf{W}}_L\| + \|\mathbf{W}_L\|) \\
\approx &\Big\|\sum_{i=1}^{M}\nabla_{\hat{\mathbf{z}}_{1i}}\bar{\mathbf{z}}_{L-1}\cdot\bar{\mathbf{z}}_1\Big\| + \frac{\|\mathbf{z}_{L-1}\|}{K_L} + \\
&\|\mathbf{z}_{L-1}\|(\|\tilde{\mathbf{W}}_L\| + \|\mathbf{W}_L\|) \\
\leq &\Big\|\sum_{i=1}^{M}\nabla_{\hat{\mathbf{z}}_{1i}}\bar{\mathbf{z}}_{L-1}\Big\|\|\bar{\mathbf{z}}_1\| + \frac{\|\mathbf{z}_{L-1}\|}{K_L} + \\
&\|\mathbf{z}_{L-1}\|(\|\tilde{\mathbf{W}}_L\| + \|\mathbf{W}_L\|) \\
= &\|\bar{\mathbf{z}}_1\| + \frac{\|\mathbf{z}_{L-1}\|}{K_L} + \|\mathbf{z}_{L-1}\|(\|\bar{\mathbf{W}}_L\| + \|\mathbf{W}_L\|) \\
= &\|\bar{\mathbf{W}}_1\mathbf{x} + \mathbf{b}_1\| + \frac{\|\mathbf{z}_{L-1}\|}{K_L} + \\
&\|\mathbf{z}_{L-1}\|(\|\bar{\mathbf{W}}_L\| + \|\mathbf{W}_L\|) \\
\leq &\|\bar{\mathbf{W}}_1\|\|\mathbf{x}\| + \|\mathbf{b}_1\| + \frac{\|\mathbf{z}_{L-1}\|}{K_L} + \\
&\|\mathbf{z}_{L-1}\|(\|\bar{\mathbf{W}}_L\| + \|\mathbf{W}_L\|) \\
= &\|\mathbf{x}\| + \|\mathbf{b}_1\| + \frac{\|\mathbf{z}_{L-1}\|}{K_L} + \|\mathbf{z}_{L-1}\|(\|\bar{\mathbf{W}}_L\| + \|\mathbf{W}_L\|) \\
= &\bar{\epsilon}
\end{aligned}
\tag{24}
$$

So, we have

$$
\begin{aligned}
&\tilde{\epsilon} - \bar{\epsilon} \\
= &\|\mathbf{x}\| + \|\mathbf{b}_1\| + \|\mathbf{z}_{L-1}\| + \|\mathbf{z}_{L-1}\|(\|\tilde{\mathbf{W}}_L\| + \|\mathbf{W}_L\|) - \\
&\Big(\|\mathbf{x}\| + \|\mathbf{b}_1\| + \frac{\|\mathbf{z}_{L-1}\|}{K_L} + \|\mathbf{z}_{L-1}\|(\|\bar{\mathbf{W}}_L\| + \|\mathbf{W}_L\|)\Big) \\
= &2(1 - \frac{1}{K_L})\|\mathbf{z}_{L-1}\|
\end{aligned}
\tag{25}
$$

Therefore, $\tilde{\epsilon} > \bar{\epsilon}$ when $K_L > 1$, i.e., our $\bar{F}$ achieves closer approximation result to the regular F than the GroupSort model $\tilde{F}$.

In addition, it is straightforward to follow the above same proof process to derive $\bar{F}$ achieves better expressive power than $\bar{F}$ on each layer l when $K_l > 1$ for $\forall l, 2 \leq l < L$.

Theorem 3 *Given a regular $(L+1)$-layer fully connected neural network $F : \mathbb{R}^{N_0} \mapsto \mathbb{R}^{N_L}$ with unconstrained ReLU as the activation and unrestricted weight, and our $(L+1)$-layer 1-Lipschitz neural network $\bar{F}$ defined in Theorem 2, if an error budget between each layer of two neural networks is limited to ε, i.e., $\|\bar{\mathbf{z}}_l - \mathbf{z}_l\| \leq \varepsilon$, where $\mathbf{z}_l$ and $\bar{\mathbf{z}}_l$ are the representations at layer l in F and $\bar{F}$ respectively, then $K_l \leq \min\Big\{\frac{\|\bar{f}(\bar{\mathbf{z}}_{l-1})\|}{\|\mathbf{W}_l f(\mathbf{z}_{l-1})\| - \varepsilon}, \max\big\{-\min_j \frac{\partial \bar{f}}{\partial \bar{\mathbf{z}}_{(l-1)j}}, \max_j \frac{\partial \bar{f}}{\partial \bar{\mathbf{z}}_{(l-1)j}}\big\}\Big\}$.*

Proof. In order to control the approximation error at layer L within the error budget ε, we have

$$
\begin{aligned}
\varepsilon \geq &\|\bar{F}(\mathbf{x}) - F(\mathbf{x})\| \\
= &\|\bar{\mathbf{W}}_L\bar{f}(\bar{\mathbf{z}}_{L-1}) + \mathbf{b}_L - (\mathbf{W}_L f(\mathbf{z}_{L-1}) + \mathbf{b}_L)\| \\
= &\|\mathbf{W}_L f(\mathbf{z}_{L-1}) - \bar{\mathbf{W}}_L\bar{f}(\bar{\mathbf{z}}_{L-1})\| \\
\geq &\|\mathbf{W}_L f(\mathbf{z}_{L-1})\| - \|\bar{\mathbf{W}}_L\bar{f}(\bar{\mathbf{z}}_{L-1})\| \\
\geq &\|\mathbf{W}_L f(\mathbf{z}_{L-1})\| - \|\bar{\mathbf{W}}_L\|\|\bar{f}(\bar{\mathbf{z}}_{L-1})\|
\end{aligned}
\tag{26}
$$

Solving the above inequality, we get

$$
K_L = \frac{1}{\|\bar{\mathbf{W}}_L\|} \leq \frac{\|\bar{f}(\bar{\mathbf{z}}_{L-1})\|}{\|\mathbf{W}_L f(\mathbf{z}_{L-1})\| - \varepsilon} \tag{27}
$$

For the purpose of maintaining the property of infinity norm, we have

$$
\begin{aligned}
K_L &= \|\nabla_{\bar{\mathbf{z}}_{L-1}}\bar{f}\|_\infty \\
&= \Big\|\Big[\frac{\partial \bar{f}}{\partial \bar{\mathbf{z}}_{(L-1)1}}, \cdots, \frac{\partial \bar{f}}{\partial \bar{\mathbf{z}}_{(L-1)N_{L-1}}}\Big]\Big\|_\infty \\
&\leq \max\big\{-\min_j \frac{\partial \bar{f}}{\partial \bar{\mathbf{z}}_{(L-1)j}}, \max_j \frac{\partial \bar{f}}{\partial \bar{\mathbf{z}}_{(L-1)j}}\big\}
\end{aligned}
\tag{28}
$$

Therefore, we have

$$
\begin{aligned}
K_L \leq &\min\Big\{\frac{\|\bar{f}(\bar{\mathbf{z}}_{L-1})\|}{\|\mathbf{W}_L f(\mathbf{z}_{L-1})\| - \varepsilon}, \\
&\max\big\{-\min_j \frac{\partial \bar{f}}{\partial \bar{\mathbf{z}}_{(L-1)j}}, \max_j \frac{\partial \bar{f}}{\partial \bar{\mathbf{z}}_{(L-1)j}}\big\}\Big\}
\end{aligned}
\tag{29}
$$

Similarly, it is straightforward to extend the above conclusion to other layers l for $\forall l, 2 \leq l < L$ by utilizing the above same proof process.

Theorem 4 *Given an $N_l \times N_{l-1}$ matrix $\mathbf{A}$, the nearest orthonormal matrix $\mathbf{B}$ of $\mathbf{A}$ is unique. It is equal to $\hat{\mathbf{B}} = \mathbf{A}\mathbf{H}^{-1}$ by using the fast hybrid polar decomposition, where $\mathbf{H} = \sqrt{\mathbf{A}^T\mathbf{A}}$ is positive definite.*

Proof. Let $\mathbf{A} = \mathbf{U}\begin{bmatrix}\Sigma\\ \mathbf{O}\end{bmatrix}\mathbf{V}^T$ *be the singular value decomposition of* $\mathbf{A}$*, where* $\mathbf{U}^T = \mathbf{U}^{-1}$*,* $\mathbf{V}^T = \mathbf{V}^{-1}$*,* $\mathbf{O}$ *is the zero matrix, and* Σ *is a square positive diagonal matrix with the singular values of* $\mathbf{A}$ *on its diagonal. We will prove that the nearest orthogonal matrix is* $\hat{\mathbf{B}} = \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1/2} = \mathbf{U}\begin{bmatrix}\Sigma\\ \mathbf{O}\end{bmatrix}\mathbf{V}^T\left(\mathbf{V}[\Sigma^T\mathbf{O}]\mathbf{U}^T\mathbf{U}\begin{bmatrix}\Sigma\\ \mathbf{O}\end{bmatrix}\mathbf{V}^T\right)^{-1/2} = \mathbf{U}\begin{bmatrix}\mathbf{I}\\ \mathbf{O}\end{bmatrix}\mathbf{V}^T$.

Now we demonstrate that $\hat{\mathbf{B}}$ *is unique by verifying the difference* $\|\mathbf{A}-\mathbf{B}\|_F$ *for any* $N_{l-1}\times N_l$ *matrix* $\mathbf{B}$ *with* N_l *orthonormal columns (i.e.* $\mathbf{B}^T\mathbf{B} = \mathbf{I}$*).*

Frobenius norm has an important property of being invariant under rotations and unitary operations in general. That is, $\|\mathbf{X}\|_F = \|\mathbf{X}\mathbf{S}\|_F$ *for any* $N_l\times N_{l-1}$ *matrix* $\mathbf{X}$ *and any* $N_{l-1}\times N_{l-1}$ *unitary matrix* $\mathbf{S}$ *or* $\|\mathbf{X}\|_F = \|\mathbf{S}\mathbf{X}\|_F$ *for any* $N_l\times N_{l-1}$ *matrix* $\mathbf{X}$ *and any* $N_l\times N_l$ *unitary matrix* $\mathbf{S}$*. We change* $\|\mathbf{A}-\mathbf{B}\|_F$ *by unitary pre- or post-multiplication, such that* $\|\mathbf{A}-\mathbf{B}\|_F = \|\mathbf{U}^T(\mathbf{A}-\mathbf{B})\mathbf{V}\|_F$*. Now we have*

$$\begin{aligned}&\mathbf{U}^T\mathbf{A}\mathbf{V} = \begin{bmatrix}\Sigma\\ \mathbf{O}\end{bmatrix}, \mathbf{U}^T\hat{\mathbf{B}}\mathbf{V} = \begin{bmatrix}\mathbf{I}\\ \mathbf{O}\end{bmatrix}, \text{and}\\ &\mathbf{U}^T\mathbf{B}\mathbf{V} = \begin{bmatrix}\mathbf{P}\\ \mathbf{Q}\end{bmatrix} \text{ satisfying } \mathbf{P}^T\mathbf{P}+\mathbf{Q}^T\mathbf{Q} = \mathbf{I}\end{aligned} \tag{30}$$

Therefore, $\|\mathbf{A}-\mathbf{B}\|_F = \left\|\begin{bmatrix}\Sigma\\ \mathbf{O}\end{bmatrix} - \begin{bmatrix}\mathbf{I}\\ \mathbf{O}\end{bmatrix} + \begin{bmatrix}\mathbf{I}\\ \mathbf{O}\end{bmatrix} - \begin{bmatrix}\mathbf{P}\\ \mathbf{Q}\end{bmatrix}\right\|_F = \|\mathbf{T}\|_F$*, where* $\mathbf{T} = \begin{bmatrix}\Sigma-\mathbf{I}\\ \mathbf{O}\end{bmatrix} - \begin{bmatrix}\mathbf{P}-\mathbf{I}\\ \mathbf{Q}\end{bmatrix}$*. As a result,* $\mathbf{T}^T\mathbf{T} = (\Sigma-\mathbf{I})^2-(\Sigma-\mathbf{I})(\mathbf{P}-\mathbf{I})-(\mathbf{P}-\mathbf{I})^T(\Sigma-\mathbf{I})+(\mathbf{P}-\mathbf{I})^T(\mathbf{P}-\mathbf{I})+\mathbf{Q}^T\mathbf{Q} = (\Sigma-\mathbf{I})^2+\Sigma\cdot(\mathbf{I}-\mathbf{P})+(\mathbf{I}-\mathbf{P}^T)\Sigma$*. Now,* $\mathbf{I}-\mathbf{P} = \mathbf{O}$ *only when* $\mathbf{B} = \hat{\mathbf{B}}$*. Otherwise* $\mathrm{Diag}(\mathbf{I}-\mathbf{P}) \geq \mathbf{O}$ *because* $\mathrm{Diag}(\mathbf{P}^T\mathbf{P}) \leq \mathrm{I}$*. At least one element of* $\mathrm{Diag}(\mathbf{I}-\mathbf{P})$ *must be positive, and thus we find* $\|\mathbf{A}-\mathbf{B}\|_F^2 = \mathrm{trace}(\mathbf{T}^T\mathbf{T}) > \mathrm{trace}(\Sigma-\mathbf{I})^2 = \|\mathbf{A}-\hat{\mathbf{B}}\|_F^2$ *as was claimed for* $\mathbf{B}\neq\hat{\mathbf{B}}$*.*

Therefore, the proof is concluded.

Theorem 5 *Consider a 1-Lipschitz neural network* $\bar{F}: \mathbb{R}^{N_0}\mapsto\mathbb{R}$*, built with norm-constrained weights* ($\|\bar{\mathbf{W}}_l\| \leq 1$) *and 1-Lipschitz, element-wise, monotonic activation functions* $\|\nabla_{\bar{\mathbf{z}}_{l-1}} f\| = 1$*. If* $||\nabla_{\mathbf{x}}\bar{F}(\mathbf{x})|| = 1$ *almost everywhere, then* $\bar{F}$ *is linear (Anil et al., 2019).*

Proof. Please refer to Theorem 1 in the GroupSort paper (Anil et al., 2019) for detailed proof.

Theorem 6 *Given* T *Weibull activation functions with the definition in Eq.(11), there must exist solutions of parameters* α_t*,* λ_t*, and* μ_t *to guarantee* $\|\nabla_{\bar{\mathbf{z}}_l}\bar{f}\| = K_l$*.*

Proof. In this paper, we choose a value of $\alpha_t < 1$ *to stimulate the attack failure rate decreasing with the perturbation diffusion. When* $\alpha_t < 1$*,* $\bar{f}'(z)$ *is monotonically decreasing function. We have*

$$\lim_{z\to\mu_t}\bar{f}'(z) = \sum_{t=1}^{T}\frac{\alpha_t}{\lambda_t}0^{\alpha_t-1}e^{0^{\alpha_t}} = \infty > K_l \text{ if } z\geq\mu_t \tag{31}$$

On the other hand,

$$\lim_{z\to\infty}\bar{f}'(z) = \sum_{t=1}^{T}\frac{\alpha_t}{\lambda_t}\infty^{\alpha_t-1}e^{-\infty^{\alpha_t}} = 0 < K_l \tag{32}$$

Notice that $\|\nabla_{\bar{\mathbf{z}}_l}\bar{f}\|_\infty = \bar{f}'(\bar{\mathbf{z}}_{lU}) = \frac{\partial\bar{f}}{\partial\bar{\mathbf{z}}_{lU}}$*, where* $\frac{\partial\bar{f}}{\partial\bar{\mathbf{z}}_{lU}} = \max\{|\frac{\partial\bar{f}}{\partial\bar{\mathbf{z}}_{l1}}|,\cdots,|\frac{\partial\bar{f}}{\partial\bar{\mathbf{z}}_{lN_l}}|\}$*. Therefore, we can always find the appropriate parameters* α_t*,* λ_t*, and* μ_t *to make* $\bar{f}'(\bar{\mathbf{z}}_{lU}) = K_l$*.*

Lemma 2 *[Restricted Stone-Weierstrass Theorem] Suppose that* (X, d_X) *is a compact metric space with at least two points and* $\mathcal{F}$ *is a lattice in* $C_{\mathcal{F}}(X,\mathbb{R})$ *with the property that for any two distinct elements* $x, y\in X$ *and any two real numbers* a *and* b *such that* $|a-b|\leq d_X(x,y)$ *there exists a function* $f\in\mathcal{F}$ *such that* $f(x) = a$ *and* $f(y) = b$*. Then* $\mathcal{F}$ *is dense in* $C_{\mathcal{F}}(X,\mathbb{R})$ *(Anil et al., 2019).*

Proof. Please refer to Lemma 1 in the GroupSort paper (Anil et al., 2019) for detailed proof.

Theorem 7 *Let* $\mathcal{LN}_\infty^{N_L}: \mathbb{R}^{N_0}\mapsto\mathbb{R}^{N_L}$ *denote the class of* $(L+1)$*-layer 1-Lipschitz neural networks* $\bar{F}$ *with norm-contained weight matrices* $\|\bar{\mathbf{W}}_l\|_\infty = 1$ $(l=1)$ *and* $\|\bar{\mathbf{W}}_l\|_\infty = 1/K_l$ $(l>1)$*, and gradient norm preserving activation function* $\|\nabla_{\bar{\mathbf{z}}_{l-1}}\bar{f}\|_\infty = K_l$*, by following the definitions in Eqs.(2) and (7). Let input* $\mathcal{X}$ *be a closed and bounded subset of* $\mathbb{R}^{N_0}$ *with the* L_∞ *metric. Then the closure of* $\mathcal{LN}_\infty^{N_L}$ *is dense in* $C_{\mathcal{F}}(\mathcal{X},\mathbb{R})$*.*

Proof. For any two input $\mathbf{x}_1 = [\mathbf{x}_{11},\cdots,\mathbf{x}_{1N_0}], \mathbf{x}_2 = [\mathbf{x}_{21},\cdots,\mathbf{x}_{2N_0}]\in\mathbb{R}^{N_0}$*,* $||\mathbf{x}_1-\mathbf{x}_2||_\infty = \max_{j=1}^{N_0}|\mathbf{x}_{1j}-\mathbf{x}_{2j}|$*. Based on Definition 1 and Lemma 1, by using the same strategy used in Theorem 1, we have the outputs* $\bar{F}(\mathbf{x}_1)$ *and* $\bar{F}(\mathbf{x}_2)$ *through the above* $(L+1)$*-layer 1-Lipschitz neural*

networks $\bar{F}$ as follows.

$$
\begin{aligned}
&\bar{F}(\mathbf{x}_1)\\
&= \bar{\mathbf{W}}_L \bar{\mathbf{h}}_{L-1} + \mathbf{b}_L = \bar{\mathbf{W}}_L \bar{f}(\bar{\mathbf{z}}_{L-1}) + \mathbf{b}_L\\
&\approx \bar{\mathbf{W}}_L \sum_{i=1}^{M} \nabla_{\hat{\mathbf{z}}_{(L-1)i}} \bar{f} \cdot \bar{\mathbf{z}}_{L-1} + \mathbf{b}_L\\
&= \bar{\mathbf{W}}_L \sum_{i=1}^{M} \big[\frac{\partial \bar{f}}{\partial \hat{\mathbf{z}}_{(L-1)i1}}, \cdots, \frac{\partial \bar{f}}{\partial \hat{\mathbf{z}}_{(L-1)iN_{L-1}}}\big] \cdot \bar{\mathbf{z}}_{L-1} + \mathbf{b}_L\\
&\leq \bar{\mathbf{W}}_L \sum_{i=1}^{M} \Big[\big|\frac{\partial \bar{f}}{\partial \hat{\mathbf{z}}_{(L-1)i1}}\big|, \cdots, \big|\frac{\partial \bar{f}}{\partial \hat{\mathbf{z}}_{(L-1)iN_{L-1}}}\big|\Big] \cdot \bar{\mathbf{z}}_{L-1} + \mathbf{b}_L\\
&\leq \bar{\mathbf{W}}_L \sum_{i=1}^{M} \max\Big\{\big|\frac{\partial \bar{f}}{\partial \hat{\mathbf{z}}_{(L-1)i1}}\big|, \cdots, \big|\frac{\partial \bar{f}}{\partial \hat{\mathbf{z}}_{(L-1)iN_{L-1}}}\big|\Big\}\\
&\quad \cdot \bar{\mathbf{z}}_{L-1} + \mathbf{b}_L\\
&= \bar{\mathbf{W}}_L \sum_{i=1}^{M} \|\nabla_{\hat{\mathbf{z}}_{(L-1)i}} \bar{f}\|_\infty \bar{\mathbf{z}}_{L-1} + \mathbf{b}_L\\
&= K_L \bar{\mathbf{W}}_L \bar{\mathbf{z}}_{L-1} + \mathbf{b}_L\\
&\approx K_L \bar{\mathbf{W}}_L \sum_{i=1}^{M} \nabla_{\hat{\mathbf{x}}_{1i}} \bar{\mathbf{z}}_{L-1} \cdot \mathbf{x}_1 + \mathbf{b}_L\\
&\leq K_L \bar{\mathbf{W}}_L \sum_{i=1}^{M} \max\Big\{\big|\frac{\partial \bar{\mathbf{z}}_{L-1}}{\partial \hat{\mathbf{x}}_{1i1}}\big|, \cdots, \big|\frac{\partial \bar{\mathbf{z}}_{L-1}}{\partial \hat{\mathbf{x}}_{1iN_0}}\big|\Big\} \cdot \mathbf{x}_1 + \mathbf{b}_L\\
&= K_L \bar{\mathbf{W}}_L \sum_{i=1}^{M} \|\nabla_{\hat{\mathbf{x}}_{1i}} \bar{\mathbf{z}}_{L-1}\|_\infty \mathbf{x}_1 + \mathbf{b}_l = K_L \bar{\mathbf{W}}_L \mathbf{x}_1 + \mathbf{b}_L
\end{aligned}
\tag{33}
$$

Similarly, we have $\bar{F}(\mathbf{x}_2) \leq K_L \bar{\mathbf{W}}_L \mathbf{x}_2 + \mathbf{b}_L$. Therefore,

$$
\begin{aligned}
||\bar{F}(\mathbf{x}_1) - \bar{F}(\mathbf{x}_2)||_\infty &\leq \|K_L \bar{\mathbf{W}}_L \mathbf{x}_1 - K_L \bar{\mathbf{W}}_L \mathbf{x}_2\|_\infty\\
&\leq K_L \|\bar{\mathbf{W}}_L\|_\infty \|\mathbf{x}_1 - \mathbf{x}_2\|_\infty\\
&\leq K_L \frac{1}{K_L} \|\mathbf{x}_1 - \mathbf{x}_2\|_\infty\\
&= \|\mathbf{x}_1 - \mathbf{x}_2\|_\infty
\end{aligned}
\tag{34}
$$

Based on the Restricted Stone-Weierstrass Theorem, the closure of $\mathcal{LN}_\infty^{N_L}$ is dense in $C_\mathcal{F}(\mathcal{X}, \mathbb{R})$.

Therefore, for any input $\mathbf{x} \in \mathbb{R}^{N_0}$, our 1-Lipschitz neural network $\bar{F}(\mathbf{x}) \in \mathcal{LN}_\infty^{N_L}$ can be used to approximate any function $\mathbb{R}^{N_0} \mapsto \mathbb{R}^{N_L}$ in continuous function space $C_\mathcal{F}(\mathcal{X}, \mathbb{R})$. Namely, our 1-Lipschitz neural network $\bar{F}(\mathbf{x})$ with K_l-Lipschitz Weibull activation is very expressive, compared with the norm-constrained neural networks with common 1-Lipschitz activation functions (e.g. ReLU, Leaky ReLU, Sigmoid, SoftPlus, or tanh).

A.2. Supplementary Experiments on Transferability

Table 6: Experiment Datasets

Dataset	Cora	Citeseer	BLOGCATALOG
#Nodes	2,708	3,327	10,312
#Edges	5,429	4,732	333,983
#Classes	7	6	39

In this section, we use three popular real graph datasets in network embedding and node classification for transferability experiments (Sen et al., 2008; Xu et al., 2019b; Zhu et al., 2019; Hasanzadeh et al., 2019; Zheng et al., 2020; Jin et al., 2020b; Feng et al., 2020; Perozzi et al., 2014; Wang et al., 2016; Dai et al., 2019; Wu et al., 2020b) , as shown in Table 6.

Transferability study. We explore whether our proposed our expressive and robust 1-Lipschitz neural network can be applied to other graph learning tasks. In this vein, we select two representative graph applications for evaluation, i.e., node classification and network embedding. In the network embedding, we use a node clustering algorithm of K-Means (Lloyd, 1982) to partition nodes into Y clusters and adopt $Dunn$ index (Liu et al., 2010; Zhou et al., 2015a) to evaluate the clustering quality. A larger Dunn value denotes a better clustering result.

$$
\begin{aligned}
&Dunn(G^1, \cdots, G^S)\\
&= \sum_{s=1}^{S} \sum_{y=1}^{Y} \frac{\min_{1 \leq x < y \leq Y} \|\mathbf{c}_x - \mathbf{c}_y\|_2^2}{\max_{1 \leq y \leq Y} \frac{1}{|C_y|} \sum_{\mathbf{v}_i^s \in C_y} \|\mathbf{v}_i^s - \mathbf{c}_y\|_2^2}
\end{aligned}
\tag{35}
$$

where C_y represents the y^{th} cluster among total Y clusters and $\mathbf{c}_y$ is the center of cluster C_y. $|C_y|$ denotes the size of C_y. $\|\mathbf{c}_x - \mathbf{c}_y\|_2^2$ represents the inter-cluster distance between clusters C_x and C_y, and $\|\mathbf{v}_i^s - \mathbf{c}_y\|_2^2$ measures the intra-cluster distance of cluster C_y. A cluster with a large $Dunn$ index indicates that the cluster has high inter-cluster distance and low intra-cluster distance. The algorithms that produce clusters with high Dunn index are more desirable.

Node classification baselines. We compare our ERNN model with two recent general graph denoising algorithms, two state-of-the-art robust node classification models against adversarial attacks, and two representative Lipschitz-bound neural architectures for restricting the perturbation propagation. **GCN-Jaccard** (Wu et al., 2019a) eliminates edges that connect nodes with Jaccard similarity of features smaller than a threshold τ. Here we use structural features of nodes to calculate the Jaccard similarity. **GCN-SVD** (Entezari et al., 2020) learns a low-rank approximation of the graph to resist high-rank perturbations. Both GCN-Jaccard and

Table 7: *Accuracy* of node classification with 10% perturbed edges

Dataset	**Cora**			**BLOGCATALOG**		
Attack Model	**Clean**	**RND Attack**	**GC-RWCS Attack**	**Clean**	**RND Attack**	**GC-RWCS Attack**
GCN-Jaccard	35.6	33.1	19.9	32.7	29.3	22.6
GCN-SVD	36.2	33.6	25.9	30.6	12.5	28.3
Pro-GNN	32.3	29.2	31.9	31.9	17.6	17.6
GRAND	31.9	30.4	30.1	23.0	23.0	23.1
GroupSort	56.6	55.8	55.9	47.2	29.4	29.3
BCOP	58.9	54.9	58.1	46.9	29.6	29.2
ERNN	**59.2**	**57.6**	**58.5**	**66.9**	**61.5**	**64.0**

Table 8: *Accuracy* of node classification of ERNN variants with 10% perturbed edges

Dataset	**Cora**			**BLOGCATALOG**		
Attack Model	**Clean**	**RND Attack**	**GC-RWCS Attack**	**Clean**	**RND Attack**	**GC-RWCS Attack**
ERNN-1	56.9	54.9	55.2	63.9	57.2	59.8
ERNN-R	52.1	50.2	52.0	59.1	55.8	56.2
ERNN-N	48.6	45.9	48.2	55.1	53.9	52.1
ERNN	**59.2**	**57.6**	**58.5**	**66.9**	**61.5**	**64.0**

GCN-SVD are general perturbation elimination models irrelevant to specific graph learning tasks and architectures. **Pro-GNN** (Jin et al., 2020b) jointly learns a clean graph and a robust GNN model for defending node classification. **GRAND** (Entezari et al., 2020) is a graph random neural network with random propagation and data augmentation to increase the robustness of node classification. **GroupSort** (Anil et al., 2019; Cohen et al., 2019) is a 1-Lipschitz fully-connected neural network that restricts the perturbation propagation by imposing a Lipschitz constraint on each layer. **BCOP** (Li et al., 2019b) is a Lipschitz-constrained convolutional network with expressive parameterization of orthogonal convolution operations. For GCN-SVD and Pro-GNN, we utilize a GCN on the denoised graphs generated by these two models to learn the node classification. Since some comparison methods do not use attributes for the tasks of node classification and network embedding, in order to make a fair comparison, we use only the graph structures without attributes in all the experiments of node classification and network embedding.

GC-RWCS (Ma et al., 2020a) is a novel black-box attack model for node classification on GNNs with a constraint of limited node access, by exploiting the structural inductive biases of GNNs. We will utilize Random attack **(RND)** and GC-RWCS to attack the node classification algorithms.

For the node classification task, the following loss $\mathcal{L}$ is utilized and minimized based on the cross-entropy loss for GCN-Jaccard, GCN-SVD, GroupSort, and BCOP, while Pro-GNN and GRAND use the default loss function in their papers.

$$\mathcal{L} = -\sum_{\mathbf{v}_i \in D} \sum_{y=1}^{Y} \mathbf{Y}_{\mathbf{v}_i y} \log \tilde{\mathbf{Y}}_{\mathbf{v}_i y} \tag{36}$$

where Y is the number of classes, D is the training data, $\mathbf{Y}$ is the ground-truth label matrix and $\tilde{\mathbf{Y}} = softmax(\mathbf{z}_L)$ are predictions of the GCNs or the Lipschitz-bounded neural networks by passing the hidden representation $\mathbf{z}_L$ in the final layer to a softmax function.

Network embedding baselines. We compare the ERNN model with two state-of-the-art robust network embedding models against adversarial attacks, GCN-SVD, and GroupSort. **DWNS** (Dai et al., 2019) proposes adversarial training method for network embedding models which can improve both model robustness and generalization ability. **GIB** (Wu et al., 2020b) presents an information-theoretic principle that optimally balances expressiveness and robustness of the learned representation of graph-structured data against adversarial attacks.

The following loss is used to maximize the similarities between connected nodes while minimizing the similarities between isolated nodes for optimizing the network embedding.

$$\mathcal{L} = \sum_{\mathbf{v}_i \in D} \Bigg(\sum_{\mathbf{v}_j \in N(\mathbf{v}_i)} \Big(-\log \sigma\big(E(\mathbf{v}_i)^T \cdot E(\mathbf{v}_j)\big)\Big) + \sum_{k=1}^{K} \mathbb{E}_{\mathbf{v}_k \sim p(\mathbf{v}_k)} \Big(\log \sigma\big(E(\mathbf{v}_i)^T \cdot E(\mathbf{v}_k)\big)\Big)\Bigg) \tag{37}$$

Table 9: $Dunn$ of node clustering of with 10% perturbed edges

Dataset	**Cora**		**Citeseer**	
Attack Model	**RND Attack**	**NEA Attack**	**RND Attack**	**NEA Attack**
GCN-SVD	30.6	33.0	30.1	28.9
DWNS	52.3	60.3	60.0	65.3
GIB	55.0	63.5	56.6	62.6
GroupSort	49.5	45.9	43.8	45.0
ERNN-1	61.8	61.0	61.3	54.9
ERNN-R	58.2	59.6	58.9	55.7
ERNN-N	55.1	55.2	55.1	50.4
ERNN	**64.1**	**64.2**	**63.6**	**69.3**

where $E(\cdot)$ represents the node embeddings by various network embedding models. $E(\mathbf{v}_i)^T$ is the transpose of $E(\mathbf{v}_i)$. $N(\mathbf{v}_i)$ is the set of neighbors of node $\mathbf{v}_i$. $p(\mathbf{v}_k)$ denotes the distribution for sampling K negative nodes $\mathbf{v}_k \neq \mathbf{v}_j$ through the negative sampling method (Mikolov et al., 2013). $\sigma(\cdot)$ is the sigmoid function. The inner product $\cdot$ represents the similarity degree between two embedding vectors. The above loss is equivalent to a cross-entropy loss with $(\mathbf{v}_i, \mathbf{v}_j)$ as positive samples and $(\mathbf{v}_i, \mathbf{v}_k)$ as negative ones.

For the node classification experiments, by following the similar setting in (Zügner et al., 2018; Chang et al., 2020), we split the graph into labeled (30%) and unlabeled nodes (70%). Further, the labeled nodes are splitted into equal parts for training and validation. The defense performance is evaluated with the node classification accuracy under adversarial attacks. $Accuracy$ is used to quantify the quality of node classification. In the network embedding tests, we run the K-Means algorithm (Lloyd, 1982) on the node embeddings generated by different network embedding methods to partition nodes into Y clusters. We use $Dunn$ to validate the performance of node clustering. As we can see from Tables 7-9, while all the node classification and network embedding methods are effective, our method achieves the best robustness in most experiments. It validates the generalization ability of our expressive and robust 1-Lipschitz neural network on other graph learning models.

A.3. Supplementary Experiments on Parameter Analysis

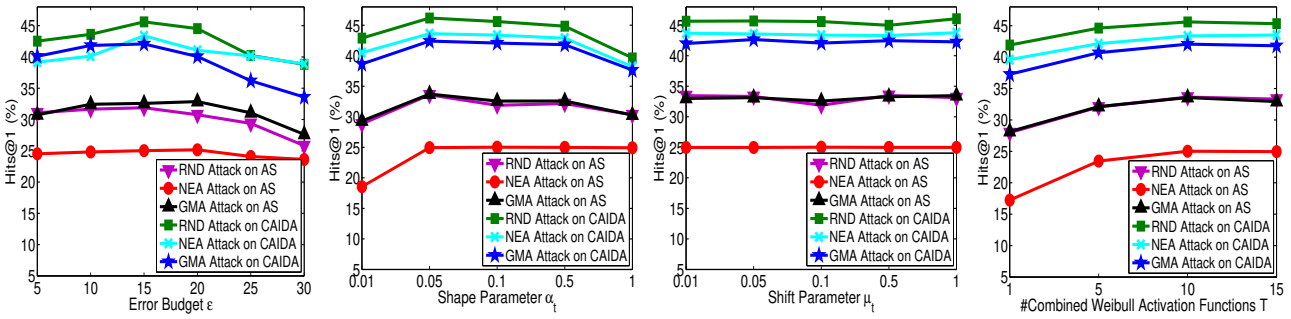
In this section, we conduct more experiments to validate the sensitivity of various parameters in our expressive and robust 1-Lipschitz neural network for the graph matching task.

Impact of error budget ε. In Theorem 3, we use error budget ε to control the approximation error between our 1-Lipschitz neural network and regular fully connected neural network. Based on a given ε, we derive the lower and upper bounds of feasible K_l. Figure 7 (a) shows the impact of ε in our 1-Lipschitz neural network model over two groups of datasets by varying ε from 5 to 30. We have witnessed the performance curves initially stable and then drop quickly when ε continuously increases. This demonstrates that our 1-Lipschitz neural network can approximate the regular neural network well and produce an expressive learning result with reasonable error budget. We have observed that the $Hits@1$ scores oscillate within the range of 27.4% and 10.6% on AS and CAIDA respectively.

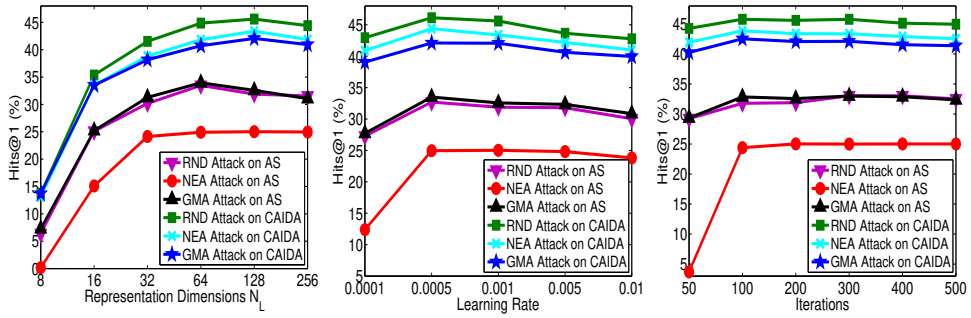
Sensitivity of shape parameter α_t. In our composite Weibull activation function, there are four parameters of shape parameter α_t, scale parameter λ_t, shift parameter μ_t, and number T of combined Weibull activation functions. Figure 7 (b) exhibits the impact of α_t in the Weibull activation function by varying α_t from 0.01 to 1. the $Hits@1$ values have concave curves when increasing α_t. This demonstrates that a smaller $\alpha_t < 1$ can effectively model the relationship between the perturbation diffusion and the attack failure, while an appropriate α_t approaching 1 indicates that the attack failure rate has less correlation with the perturbation diffusion, especially the attack failure rate becomes constant over the perturbation diffusion when $\alpha_t = 1$. Therefore, a smaller $\alpha_t < 1$ can help the Weibull activation function robust to adversarial attacks.

Impact of shift parameter μ_t. Figure 7 (c) presents the impact of shift parameter μ_t in the Weibull activation function with α_t between 0.01 and 1. It is observed that the $Hits@1$ values are very stable with varying μ_t. Namely, our composite Weibull activation function is insensitive to μ_t. A reasonable explanation is that the shift parameter corresponds to the time of the attack failure. This demonstrates that our 1-Lipschitz neural network can always result in the attack failure after enough perturbation diffusion.

Influence of number T of combined Weibull activation functions. Figure 7 (d) shows the sensitivity of T in the Weibull activation function with T between 1 and 15. The



(a) Error Budget ε (b) Shape Parameter α_t (c) Shift Parameter μ_t (d) #Combined Activations T



(e) Dimensions N_L (f) Learning Rate (g) Iterations

Figure 7: $Hits@1$ (%) with varying parameters

performance curves continuously increase with increasing T. This is consistent with the fact that non-saturating non-linear activation functions can achieve faster training than the saturating ones. A large T help the Weibull activation function achieve the advantage of non-saturating nonlinearity.

Impact of representation dimensions N_L at final layer L. Figure 7 (e) exhibits the impact of the number of node representation dimensions in the ERNN model over two dataset. We have observed that the performance initially raises when the number of dimension increases. Intuitively, the final node representations with more dimensions can introduce enough information for sparse graphs, and thus help improve the quality of graph matching running on the latent representations. Later on, the performance curves keep relatively stable or even decrease when the number of dimensions continuously increases. A reasonable explanation is that the additional dimensions are unnecessary for future prediction if we already have enough information for graph matching analysis. In addition, less dimensions can help improve the efficiency of both representation learning and graph matching. Thus, it is important to determine the optimal number of dimensions for the graph matching.

Sensitivity of learning rate. Figure 7 (f) shows the impact of learning rate in our ERNN model by varying it from 0.0001 to 0.01. the $Hits@1$ values have concave curves when increasing learning rate. A too small learning rate may result in a long training process that could get stuck, whereas a too large learning rate may result in learning a sub-optimal set of weights too fast or an unstable training process. Thus, this demonstrates that there must exist the optimal learning rate that makes the performance of our 1-Lipschitz neural network be maximally optimized.

Convergence study. Figure 7 (g) presents the convergence of our 1-Lipschitz neural network for the graph matching tasks. As we can see, the $Hits@1$ values keep increasing when we iteratively perform the defense task. The method converges when the numbers of iterations go beyond some thresholds. We have observed that the curves on two datasets converge very quickly in 100 iterations. This verifies the efficiency and potential of our 1-Lipschitz neural network to combat with perturbations for any kind of adversarial attacks.

A.4. Experimental Details

Environment. Our experiments were conducted on a compute server running on Red Hat Enterprise Linux 7.2 with 2 CPUs of Intel Xeon E5-2650 v4 (at 2.66 GHz) and 8 GPUs of NVIDIA GeForce GTX 2080 Ti (with 11GB of GDDR6 on a 352-bit memory bus and memory bandwidth in the neighborhood of 620GB/s), 256GB of RAM, and 1TB of HDD. Overall, our experiments took about 4 days in a shared resource setting. We expect that a consumer-grade

single-GPU machine (e.g., with a 1080 Ti GPU) could complete our full set of experiments in around 8 days, if its full resources were dedicated. The codes were implemented in Python 3.7.3 and PyTorch 1.0.14. We also employ Numpy 1.16.4 and Scipy 1.3.0 in the implementation. Since the datasets used are all public datasets and the hyperparameter settings are explicitly described, our experiments can be easily reproduced on top of a GPU server.

Implementation. For random attack model, we add the noisy edges to the datasets with different levels of noisy data, say 5% (i.e., 0.05), by randomly adding or removing edges with the half noise level respectively, say 2.5%. For other four attack models of NEA [1], GMA [2], RL-S2V [3], and GC-RWCS [4], we used the open-source implementation and default parameter settings by the original authors for our experiments. For two general graph denoising methods of GCN-Jaccard [5] and GCN-SVD [6], two node classification algorithms of Pro-GNN [7] and GRAND [8], three graph classification approaches of PAN [9], RoboGraph [10], and GraphCL [11], two Lipschitz-bound neural architectures of GroupSort [12] and BCOP [13], six graph matching methods of FINAL [14], REGAL [15], MOANA [16], DGMC [17], CONE-Align [18], and G-CREWE [19], two network embedding baselines of DWNS [20] and GIB [21], we also utilized the same model architecture as the official implementation provided by the original authors and used the same perturbed graphs to validate the robustness of these graph learning models in all experiments. For the GCN model [22], we also use the default parameters in the authors' implementation. We used the public TensorFlow implementation of GCN and pass the hidden representation in the final layer by the GCN to a softmax function as the node classification results. Since some comparison methods do not use attributes for the tasks of node classification and network embedding, in order to make a fair comparison, we use only the graph structures without attributes in all the experiments of node classification and network embedding.

For our expressive and robust 1-Lipschitz neural network, we performed hyperparameter selection by performing a parameter sweep on Lipschitz constant $K_l \in \{0.5, 1, 1.5, 2, 2.5\}$, error budget $\varepsilon \in \{5, 10, 15, 20, 25, 30\}$ between our 1-Lipschitz neural network and regular fully connected neural network, shape parameter $\alpha_t \in \{0.01, 0.05, 0.1, 0.5, 1\}$, shift parameter $\mu_t \in \{0.01, 0.05, 0.1, 0.5, 1\}$, number $T \in \{1, 5, 10, 15, 20\}$ of combined Weibull activation functions, dimensions $N_L \in \{8, 16, 32, 64, 128, 256\}$ of final layer L, and learning rate $\in \{0.0001, 0.0005, 0.001, 0.005, 0.01\}$. We select the best parameters over 50 iterations of training and evaluate the model at test time. After the hyperparameter selection, the model was trained for 500 iterations, with a batch size of 512, and a learning rate of 0.001.

[1] https://www.in.tum.de/daml/node-embedding-attack/
[2] https://github.com/DMML-AU/GMA
[3] https://github.com/Hanjun-Dai/graph_adversarial_attack
[4] https://github.com/Mark12Ding/GNN-Practical-Attack
[5] https://github.com/DSE-MSU/DeepRobust/
[6] https://github.com/DSE-MSU/DeepRobust/
[7] https://github.com/ChandlerBang/Pro-GNN
[8] https://github.com/THUDM/GRAND
[9] https://github.com/YuGuangWang/PAN
[10] https://github.com/RobustGraph/RoboGraph
[11] https://github.com/Shen-Lab/GraphCL
[12] https://github.com/cemanil/LNets
[13] https://github.com/ColinQiyangLi/LConvNet
[14] https://github.com/sizhang92/FINAL-network-alignment-KDD16
[15] https://github.com/GemsLab/REGAL
[16] https://github.com/sizhang92/Multilevel-network-alignment-Moana-
[17] https://github.com/rusty1s/deep-graph-matching-consensus
[18] https://github.com/GemsLab/CONE-Align
[19] https://github.com/cruiseresearchgroup/G-CREWE
[20] https://github.com/wonniu/AdvT4NE_WWW2019
[21] http://snap.stanford.edu/gib/
[22] https://github.com/tkipf/gcn