SUPPLEMENTARY FILE
Graph Neural Networks Inspired by Classical Iterative Algorithms

A. Dataset and Experimental Setting Details

Standard Benchmarks In Section 5.1 of the main paper, we used four datasets, namely Cora, Citeseer, Pubmed and ogb-arxiv. These four are all citation datasets, i.e. their nodes represent papers and edges represents citation relationship. The node features of the former three are bag-of-words. Following (Yang et al., 2016), we use a fixed spitting for these three datasets in which there are 20 nodes per class for training, 500 nodes for validation and 1000 nodes for testing. For ogbn-arxiv, the features are word2vec vectors. We use the standard leaderboard splitting for ogbn-arxiv, i.e. papers published until 2017 for training, papers published in 2018 for validation, and papers published since 2019 for testing.

Adversarial Attack Experiments As mentioned in the main paper, we tested on Cora and Citerseer using Mettack. We use the DeepRobust library (Li et al., 2020) and apply the exact same non-targeted attack setting as in (Zhang & Zitnik, 2020). For all the baseline results in Table 4, we run the implementation in the DeepRobust library or the GNNGuard official code. Note the GCN-Jaccard results differ slightly from those reported in (Zhang & Zitnik, 2020), likely because of updates in the DeepRobust library and the fact that (Zhang & Zitnik, 2020) only report results from a single trial (as opposed to averaged results across multiple trails as we report).

Heterophily Experiments In Section 5.3, we use four datasets introduced in (Pei et al., 2019), among which Cornell, Texas, and Wisconsin are web networks datasets, where nodes correspond to web pages and edges correspond to hyperlinks. The node features are the bag-of-words representation of web pages. In contrast, the Actor dataset is induced from a film-director-actor-writer network (Tang et al., 2009), where nodes represent actors and edges denote co-occurrences on the same Wikipedia page. The node features represent some keywords in the Wikipedia pages. We used the data split, processed node features, and labels provided by (Pei et al., 2019), where for the former, the nodes of each class are randomly split into 60%, 20%, and 20% for train, dev and test set respectively.

Long-Range Dependency/Sparse Label Tests In Section 5.4, we adopt the Amazon Co-Purchase dataset, which has previously been used in (Gu et al., 2020) and (Dai et al., 2018) for evaluating performance involving long-range dependencies. We use the dataset provided by the IGNN repo (Gu et al., 2020), including the data-processing and evaluation code, in order to obtain a fair comparison. As for splitting, 10% of nodes are selected as the test set. And because there is no dev set, we directly report the test result of the last epoch. We also vary the fraction of training nodes from 5% to 9%. Additionally, because there are no node features, we learn a 128-dim feature vector for each node. All of these settings from above follow from (Gu et al., 2020).

Summary Statistics Table 5 summarizes the attributes of each dataset.

Table 5. Dataset statistics. The *FEATURES* column describes the dimensionality of node features. Note that the Amazon Co-Purchase dataset has no node features.

DATASET	NODES	EDGES	FEATURES	CLASSES
CORA	2,708	5,429	1,433	7
CITESEER	3,327	4,732	3,703	6
PUBMED	19,717	44,339	500	3
ARXIV	169,343	1,166,243	128	40
TEXAS	183	309	1,703	5
WISCONSIN	251	499	1,703	5
ACTOR	7,600	33,544	931	5
CORNELL	183	295	1,703	5
AMAZON	334,863	2,186,607	-	58

B. Model Specifications

B.1. Basic Architecture Design

The TWIRLS architecture is composed of the input module $f(X;W)$, followed by the unfolded linear propagation layers defined by (21) interleaved with attention given by (20), concluding with $g(\boldsymbol{y};\theta)$. Note that the attention only involves reweighting the edge weights of the graph (i.e., it does not alter the node embeddings at each layer), and when no attention is included we obtain TWIRLS$_{\text{base}}$.

The aggregate design is depicted in in Figure 3. For simplicity, we generally adopt a single attention layer sandwiched between equal numbers of propagation layers; however, for heterophily datasets we apply an extra attention layer before propagation. Additionally, for all experiments except ogbn-arxiv, we set $g(\boldsymbol{y};\theta) = \boldsymbol{y}$ (i.e., an identity mapping). For ogbn-arxiv, we instead set $f\,(X;W) = X$. Hence for every experiment, TWIRLS restricts all parameters to a single MLP module (or linear layer for some small datasets; see hyperparameter details below).

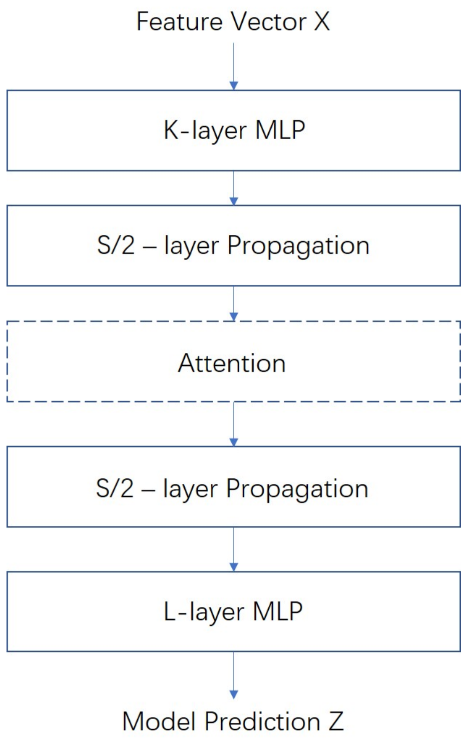

Figure 3. Model Architecture. S is the total number of propagation steps. K and L are number of MLP layers before and after the propagation respectively. While not a requirement, in all of our experiments, either K or L is set to zero, meaning that the MLP exists on only one side of the propagation layers.

B.2. Specific Attention Formula

While the proposed attention mechanism can in principle adopt any concave, non-decreasing function ρ, in this work we restrict ρ to a single functional form that is sufficiently flexible to effectively accommodate all experimental scenarios. Specifically, we adopt

$$\rho(z^2) = \begin{cases} \bar{\tau}^{p-2} z^2 & \text{if } z < \bar{\tau} \\ \frac{2}{p}\bar{T}^p - \rho_0 & \text{if } z > \bar{T} \\ \frac{2}{p} z^p - \rho_0 & \text{otherwise,} \end{cases} \tag{26}$$

where p, $\bar{T}$, and $\bar{\tau}$ are non-negative hyperparameters and $\rho_0 = \frac{2-p}{p}\bar{\tau}^p$ is a constant that ensures ρ is continuous. Additionally, the gradient of ρ produces the attention score function (akin to γ in the main paper) given by

$$s(z^2) \triangleq \frac{\partial \rho(z^2)}{\partial z^2} = \begin{cases} \bar{\tau}^{p-2} & \text{if } z < \bar{\tau} \\ 0 & \text{if } z > \bar{T} \\ z^{p-2} & \text{otherwise.} \end{cases} \tag{27}$$

And for convenience and visualization, we also adopt the reparameterizations $\tau = \bar{\tau}^{\frac{1}{2-p}}$ and $T = \bar{T}^{\frac{1}{2-p}}$, and plot $\rho(z^2)$ and $s(z^2)$ in Figure 4 using $p = 0.1, \tau = 0.2, T = 2$.

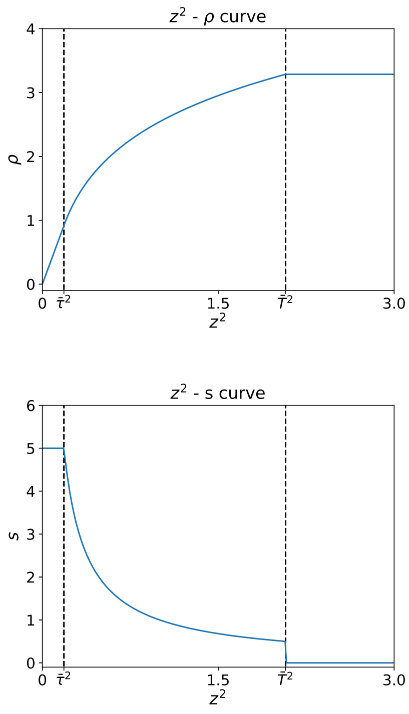

Figure 4. A visualization of attention functions.

Overall, this flexible choice has a natural interpretation in terms of its differing behavior between the intervals $[0, \bar{\tau}]$, $(\bar{\tau}, \bar{T})$, and $(\bar{T}, \infty)$. For example, in the $[0, \bar{\tau}]$ interval a quadratic penalty is applied, which leads to constant attention independent of z. This is exactly like TWIRLS$_{\text{base}}$. In contrast, within the $(\bar{T}, \infty)$ interval ρ is constant and the corresponding attention weight is set to zero (truncation), which is tantamount to edge removal. And finally, the middle interval provides a natural transition between these two extremes, with ρ becoming increasingly flat with larger z values.

Additionally, many familiar special cases emerge for certain parameter selections. For example $T = \infty$ corresponds with no explicit truncation, while $p = 2$ can instantiate no attention. And $T = \tau$ means we simply truncate those edges with large distance, effectively keeping the remaining edge attention weights at 1 (note that there is normalization during propagation, so setting edges to a constant is equivalent to setting them to 1).

B.3. Hyperparameters

All model structure-related hyperparameters for TWIRLS$_{\text{base}}$ can be found in Table 6. For experiments with attention, other hyperparameters of the model are the same as the base version, and additional hyperparameters introduced by attention can be found in Table 7. For training, we use the Adam optimizer for all experiments, with learning rate = 0.1 for Cora, Citeseer, attacked Cora, and Texas, and 0.5 for Pubmed and other heterophily datasets. For ogbn-arxiv and Amazon Co-Purchase, we use learning rate = 1e-3 and 1e-2 respectively.

Dataset	# prop layers	λ	α	MLP layers	hidden layer size
Cora	16	1	1	2	-
Citeseer	16	1	1	2	-
Pubmed	40	1	1	1	-
Arxiv	7	20	0.05	3	512
ATK-Cora	32	1	1	1	-
ATK-Cite	64	1	1	1	-
Wisconsin	4	0.001	1	2	64
Cornell	4	0.001	1	2	64
Texas	6	0.001	1	2	64
Actor	6	0.001	1	2	64
Amazon	32	10	0.1	1	128

Table 6. Model hyperparameters for TWIRLS$_{\text{base}}$.

Consistent with prior work, we apply L2 regularization on model weights, with the corresponding weight decay rate set to 5e-4 for Cora, Pubmed, Wisconsin and Texas, 1e-3 for Citeseer, attacked citeseer, Cornell and Actor, 5e-5 for attacked Cora and 0 for others. Again, as in prior work, we also used dropout as regularization, with dropout rate set to 0.8 for Cora and Pubmed, 0.5 for Citeseer, ogb-arxiv, attcked Cora and attacked Citeseer, and 0 for other datasets.

Dataset	p	τ	T
ATK-Cora	0.1	0.2	2
ATK-Citeseer	0.1	0.2	2
wisconsin	1	0.1	$+\infty$
cornell	0	0.001	$+\infty$
texas	0	10	$+\infty$
geom-film	1	0.1	$+\infty$

Table 7. Attention Hyperparameters for TWIRLS

C. Model Variations

C.1. Alternative GCN-like Reparameterization

If we define the reparameterized embeddings $Z = \tilde{D}^{1/2}Y$ and left multiply (6) by $\tilde{D}^{1/2}$, we have

$$\begin{aligned} Z^{(k+1)} &= (1-\alpha)Z^{(k)} + \alpha\lambda\tilde{D}^{-1/2}AY^{(k)} + \alpha\tilde{D}^{-1/2}f(X;W) \\ &= (1-\alpha)Z^{(k)} + \alpha\lambda\tilde{D}^{-1/2}A\tilde{D}^{-1/2}Z^{(k)} + \alpha\tilde{D}^{-1}Z^{(0)}. \end{aligned} \tag{28}$$

From here, if we choose $\alpha = \lambda = 1$, for $Z^{(1)}$ we have that

$$\begin{aligned} Z^{(1)} &= \left(\tilde{D}^{-1/2}A\tilde{D}^{-1/2} + \tilde{D}^{-1}\right)Z^{(0)} \\ &= \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}Z^{(0)}, \end{aligned} \tag{29}$$

which gives the exact single-layer GCN formulation in Z-space with $Z^{(0)} = f(X;W)$.

C.2. Normalized Laplacian Unfolding

From another perspective, if we replace L in (1) with a normalized graph Laplacian, and then take gradients steps as before, there is no need to do preconditioning and reparameterizing. For example, following (5) with L changed to the symmetrically-normalized version $\tilde{L} = I - \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$, we get

$$Y^{(k+1)} = (1-\alpha-\alpha\lambda)Y^{(k)} + \alpha\lambda\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}Y^{(k)} + \alpha Y^{(0)}, \tag{30}$$

where we set $\tilde{D} = I + D$. This formula is essentially the same as (28). The main difference is that there is no $\tilde{D}^{-1}$ in front of X, which indicates an emphasis on the initial features. We found this version to be helpful on ogbn-arxiv and Amazon Co-Purchase data. Note however that all of our theoretical support from the main paper applies equally well to this normalized version, just with a redefinition of the gradient steps to include the normalized Laplacian.

C.3. Layer-Dependent Weights

It is also possible to seemlessly address the introduction of layer/iteration-dependent weights. Within the unfolding framework, this can be accomplished by simply changing the specification of the norms used to define $\ell_Y(Y)$. For example, if at each iteration we swap the stated parameter-free Frobenius norms with the reweighted alternative $\|U\|^2_{\Sigma^{(k)}} =$ trace $\left[U^\top\Sigma^{(k)}U\right]$, where $\Sigma^{(k)} = M^{(k)}(M^{(k)})^\top$ for some matrix $M^{(k)}$, then each term on the r.h.s. of (6) will be right multiplied by $\boldsymbol{M}^{(k)}$; similarly for (10). This can be viewed as applying a learnable warping metric to each iteration. While this additional flexibility may at times be useful, in the interest of simplicity, for the experiments presented herein we did not include them.

C.4. Sampling and Spectral Sparsification

The recursive neighborhood expansion across layers poses time and memory challenges for training large and dense graphs. In this regard, sparsifying the graph in each layer by random sampling is an effective technique, which significantly reduces training time and memory usage but still allows for competitive accuracy (Chen et al., 2018a;b). Edge sampling has also proved to be effective for relieving over-fitting and over-smoothing in deep GCNs (Rong et al., 2019).

More specifically, the normalized adjacency matrix $\tilde{A}$ can be sparsified via random sampling. Let $\tilde{A}'$ denote this sparse version. Then

$$Z'^{(k+1)} = \mathsf{ReLU}(\tilde{D}^{-1/2}\tilde{A}'\tilde{D}^{-1/2}Z'^{(k)}) \tag{31}$$

represents the corresponding GCN embedding update. Note that the sparse random matrix $\tilde{A}'$ used in each layer will generally be different i.i.d. samples. It should also be observed that nonlinear activation functions make the overall function rather complicated, and in particular, it is difficult to get unbiased estimators, i.e., $Z^{(k)} = \mathbb{E}\left[Z'^{(k)}\right]$ for all k. To address this issue, (Chen et al., 2018a) assume that there is a (possibly infinite) graph $\mathcal{G}'$ with the vertex set $\mathcal{V}'$ associated with a probability space $(\mathcal{V}', \mathcal{F}, \mathcal{P})$, such that for the given graph $\mathcal{G}$, it is an induced subgraph of $\mathcal{G}'$ and its vertices are i.i.d. samples of $\mathcal{V}'$ according to the probability measure $\mathcal{P}$. But even granted this strong assumption, (Chen et al., 2018a) were only be able to show that Z' is a consistent estimator of Z, but not necessarily unbiased. Consequently, it can be argued that the theoretical foundation of why random sampling does not significantly impact the accuracy still remains at least partially unclear.

In this context, the perspective on GCNs from Sections 2.2 (main paper) and C.1 (supplementary) provides a simple alternative explanation. Let L' be the Laplacian matrix of the subsampled graph with appropriate scaling such that we have $L = \mathbb{E}[L']$. And let $\ell'_Y(Y) = \|Y - f(X;W)\|^2_{\mathcal{F}} + \lambda \text{tr}\left(Y^T L' Y\right)$. It is then easy to check that $\ell_Y(Y) = \mathbb{E}[\ell'_Y(Y)]$ for all Y. And giving the embeddings

$$\begin{aligned} Z &= \arg\min_Y \ell_Y(Y), \text{ s. t. } Y \geq 0 \text{ and} \\ Z' &= \arg\min_Y \ell'_Y(Y), \text{ s. t. } Y \geq 0, \end{aligned} \tag{32}$$

we observe that, even though Z' is not an unbiased estimator of Z, it is nonetheless the optima of an objective function $\ell'_Y(Y)$ that is an unbiased estimator of the corresponding objective for Z.

Additionally, per this interpretation, we can also apply spectral sparsification results to get strong theoretical guarantees on graph sparsification for GCNs. For dense graphs with n vertices and m edges, we have that $m = \Omega(n^2)$, which is huge for moderately large graphs. However, it has been proven that there exists a sparse graph $\mathcal{G}'$, with the same set of vertices and with its edge set a reweighted subset of $\mathcal{E}$, satisfying

$$x^T L' x = (1 \pm \varepsilon) x^T L x \quad \text{for all } x, \tag{33}$$

where L' is the Laplacian of $\mathcal{G}'$, and the number of edges in $\mathcal{G}'$ is $O\left(\frac{n}{\varepsilon^2}\right)$ (Batson et al., 2012). Moreover, $\mathcal{G}'$ can be computed in near-linear time (Lee & Sun, 2017). From this result and our interpretation of GCNs, we can always obtain a sparse GCN with constant number of edges per vertex, which approximates the GCN defined by the original graph well. This provides nontrivial theoretical guarantees for graph sparsification for graph neural networks.

D. Ablation Study

D.1. Varying α and # of Propagation Steps

In the main paper, we interpret α as the gradient step size. From this viewpoint, if the step size becomes smaller, we might naturally expect that more steps are needed for the model to obtain good performance. To verify this interpretation, we vary α and the number of propagation steps S on Citeseer and observe the performance of TWIRLS$_{\text{base}}$. The results are shown in Table 8. In general, the best results are arranged on a counter diagonal, which indicates a matching of α and the number of propagation steps gives the best result as expected.

S \ α	0.1	0.25	0.5	1
8	66.00	67.25	69.51	72.35
16	66.80	69.23	72.56	**74.07**
32	68.71	72.55	**74.00**	73.78
64	**71.66**	**73.98**	73.84	72.58

Table 8. TWIRLS$_{\text{base}}$ performance on Citeseer as the step size α and the number of propagation steps S are varied. Note that for computational efficiency, the number of repeated experiments here is lower than that of the main paper, so the results are slightly different (we also omit standard deviations for compactness).

D.2. Varying Truncation Parameter T

We also show how results change when using a different truncating hyperparameter T on attacked Cora and Citeseer. The results are reported in Table 9. Overall, the model performance is relatively stable with respect to T, with the best performance occurring with $T = 2$.

T \ DATA	ATK-CORA	ATK-CITESEER
τ	70.10 ± 1.03	69.56 ± 1.32
2	**70.23 ± 1.09**	**70.63 ± 0.93**
$+\infty$	69.77 ± 1.26	70.51 ± 1.09

Table 9. TWIRLS performance under adversarial attacks with different T.

D.3. Varying MLP Layers

In Table 10 we demonstrate that even with only one MLP layer (which is linear), the performance of TWIRLS$_{\text{base}}$ is still competitive.

K \ DATA	CORA	CITESEER	PUBMED
1	83.3 ± 0.3	74.1 ± 0.5	**80.7 ± 0.5**
2	**84.1 ± 0.5**	**74.2 ± 0.45**	**80.7 ± 0.4**

Table 10. TWIRLS$_{\text{base}}$ performance with different number of MLP layers before propagetion. Here K denotes the number of MLP layers before propagation.

E. Additional Empirical Results

E.1. Running Time

The time complexity of our model is $O(mdS+Nd^2)$, where m is the number of edges, S is the number of propagation steps, N and d are the number of MLP layers and hidden size respectively. By contrast, the time complexity of a GCN is $O(mdN + Nd^2)$. Moreover, if the MLP layers are all after propagation (i.e., no parameters before propagation), the time complexity can be reduced to $O(Nd^2)$ by precomputing the propagation (i.e., the same as an MLP).

To examine empirically, we pick three ogbn-arxiv SOTA models, as well as common baselines GCN, GAT, and MLP (no graph). We train each for 100 epochs on ogbn-arxiv with a single Tesla T4 and report the average time per epoch in Table 11 (in seconds). For consistency, all models have three hidden layers and three propagation layers, and we adjust the hidden size so that the total number of parameters is roughly equivalent for all models. TWIRLS* denotes that MLP layers are after propagation; otherwise they are before propagation. Note that for the ogbn-arxiv experiment from Table 2 we use TWIRLS$^*_{\text{base}}$, so the computational cost is negligibly different from an MLP, i.e., both are $O(Nd^2)$.

Table 11. Running time.

MODEL	TRAIN	TEST	# PARAMETERS
MLP	0.672	0.124	351,272
GCN	0.775	0.192	351,272
GCNII	0.839	0.304	317,776
JKNET	0.998	0.285	362,920
DAGNN	0.769	0.154	351,313
TWIRLS$_{\text{BASE}}$	0.746	0.169	351,272
TWIRLS$^*_{\text{BASE}}$	0.679	0.124	351,272
GAT	1.486	0.266	352,336
TWIRLS	0.814	0.189	351,272
TWIRLS*	0.775	0.223	351,272

E.2. Amazon Co-Purchase

In Figure 2 from the main paper, we show the Micro F1 performance of our model on the Amazon Co-Purchase dataset. Here in Figure 5 we present the corresponding Macro F1 curve to provide a more detailed picture of our model's ability to capture long-range dependencies.

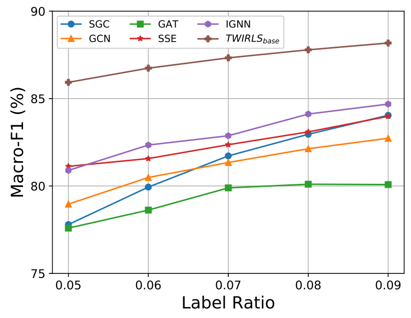

Figure 5. Amazon Co-Purchase Macro-F1 results.

E.3. Chains Long-Range Dependency Dataset

To further showcase the ability of our model to capture long-range dependencies, we tested TWIRLS using the Chains dataset introduced by (Gu et al., 2020). Note that this data has been explicitly synthesized to introduce long-range dependencies of controllable length. This is accomplished by constructing a graph formed from several uncrossed chains, each randomly labeled 0 or 1. There is also a 100-dim feature for each node. And for the node at one end of the chain, the first dimension of its feature vector is the label of this chain; for other nodes the feature vector is a zero vector. See (Gu et al., 2020) for further details. Figure 6 reveals that our model can achieve 100% accuracy on this data, unlike several of the baselines reported in (Gu et al., 2020). Again, TWIRLS was not designed for this task, but nonetheless performs well.

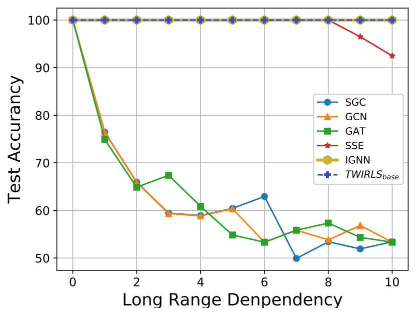

Figure 6. Results on Chains dataset. TWIRLS$_{\text{base}}$ achieves 100% accuracy as the long-range dependency is increased by varying chain lengths.

E.4. Chinese Word Segmentation

Finally, we also observed that TWIRLS$_{\text{base}}$ is even able to capture long-range dependencies contained within sentences when we model each sentence as a chain. To demonstrate this capability, we test our model on a Chinese word segmentation (CWS) dataset, namely the so-called PKU dataset (Emerson, 2005). Here each sample is a Chinese sentence with a label for each character, indicating whether this character is the start of a word, middle of a word, end of a word, or itself forms a word. The model then needs to predict the correct labels.

By viewing each Chinese character as a node, and connecting an edge between neighboring characters, we construct a graph from a sentence in a relatively naive way. We use a static character embedding trained by FastNLP[1] and run TWIRLS$_{\text{base}}$ on this graph. We simply set $\alpha = 0.5$, $\lambda = 1$, and apply 8 propagation steps, and use a 2-layer MLP after propagation with a hidden size of 512. As baselines, we train a GCN and MLP with the same number of layers and hidden size. We also include an 8-layer GCN (denoted by GCN-8), to allow the GCN to have more propagation steps. And as an additional baseline explicitly designed for modeling dependencies within text sequences, we include a bilateral LSTM that includes one LSTM layer and one linear transform layer.

Table 12 shows the resulting Macro F1 scores, which demonstrates the ability of TWIRLS$_{\text{base}}$ to handle sentences reasonably well, while the similar performance of GCN and MLP shows that the former does not have similar ability. Surprisingly, TWIRLS$_{\text{base}}$ performance is even comparable with the Bi-LSTM sequence model despite not being designed for this task. And note also, as a quick preliminary test, we did not tune the hyperparameters, nor finely design the model structure and graph construction for this task, so there are space to further boost performance on this type of task.

[1]https://github.com/fastnlp/fastNLP

MODEL	TEST ACCURACY
MLP	56.68 ± 2.81
GCN	61.95 ± 0.52
GCN-8	37.25 ± 0.62
TWIRLS$_{\text{BASE}}$	**84.86 ± 0.39**
BI-LSTM	90.75 ± 0.52

Table 12. Performance of different models on the CWS task. TWIRLS$_{\text{base}}$ outperforms other graph-based models, and is even competitive with a bilateral LSTM that is explicitly designed to handle long-range dependencies within sequences.

F. Proof of Technical Results

Lemma 3.1 *For any $p(Y)$ expressible via (13), we have*

$$-\log p(Y) = \pi(Y;\rho) \triangleq \sum_{\{i,j\}\in\mathcal{E}} \rho\left(\left\|\boldsymbol{y}_i - \boldsymbol{y}_j\right\|_2^2\right)$$

excluding irrelevant constants, where $\rho : \mathbb{R}^+ \to \mathbb{R}$ is a concave non-decreasing function that depends on μ.

***Proof*:** A function $f : \mathbb{R}_+ \to \mathbb{R}_+$ is said to be *totally monotone* (Widder, 2015) if it is continuous on $[0,\infty)$ and infinitely differentiable on $(0,\infty)$, while also satisfying

$$(-1)^n \frac{\partial^n}{\partial u^n} f(z) \;\geq\; 0, \;\; \forall\, n = 1,2,\ldots. \tag{34}$$

for all $z > 0$. Furthermore, a non-negative symmetric function $p_z(z)$ can be expressed as a Gaussian scale mixture, i.e.,

$$p_z(z) = \int \mathcal{N}\left(z|0,\gamma^{-1}I\right) d\mu(\gamma), \tag{35}$$

for some positive measure μ, iff $p_z(\sqrt{z})$ is a totally monotone function on $[0,\infty)$ (Andrews & Mallows, 1974). However, as shown in (Palmer et al., 2006), any such totally monotone function can be expressed as $p_z(\sqrt{z}) = \exp[-\rho(z)]$, where ρ is a concave, non-decreasing function. From these results, and the assignment $z_{ij} \triangleq \sqrt{\boldsymbol{u}_{ij}^\top \boldsymbol{u}_{ij}} =$

$\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2$, we can then infer that

$$\begin{aligned}
&-\log p(Y)\\
&\equiv -\sum_{\{i,j\}\in\mathcal{E}} \log \int \mathcal{N}\left(\boldsymbol{u}_{ij}|0,\gamma_{ij}^{-1}I\right) d\mu\left(\gamma_{ij}\right)\\
&= -\sum_{\{i,j\}\in\mathcal{E}} \log \int \left(\tfrac{\gamma_{ij}}{2\pi}\right)^{d/2} \exp\left[-\tfrac{\gamma_{ij}}{2}\boldsymbol{u}_{ij}^\top \boldsymbol{u}_{ij}\right] d\mu\left(\gamma_{ij}\right)\\
&= -\sum_{\{i,j\}\in\mathcal{E}} \log \int \left(\tfrac{\gamma_{ij}}{2\pi}\right)^{1/2} \exp\left[-\tfrac{\gamma_{ij}}{2}z_{ij}^2\right] d\mu'\left(\gamma_{ij}\right)\\
&= -\sum_{\{i,j\}\in\mathcal{E}} \log \int \mathcal{N}\left(z_{ij}|0,\gamma_{ij}^{-1}I\right) d\mu'\left(\gamma_{ij}\right)\\
&= -\sum_{\{i,j\}\in\mathcal{E}} \log p_z\left(z_{ij}\right)\\
&= -\sum_{\{i,j\}\in\mathcal{E}} \log p_z\left(\sqrt{\|\boldsymbol{y}_i-\boldsymbol{y}_j\|_2^2}\right)\\
&= \sum_{\{i,j\}\in\mathcal{E}} \rho\left(\|\boldsymbol{y}_i-\boldsymbol{y}_j\|_2^2\right), \qquad (36)
\end{aligned}$$

where the positive measure μ' is defined such that $d\mu'(\gamma_{ij}) = \left(\frac{\gamma_{ij}}{2\pi}\right)^{(d-1)/2} d\mu(\gamma_{ij})$, noting that prior results apply equally well to this updated version for some concave non-decreasing ρ. Hence Lemma 3.1 directly follows. ■

Lemma 3.2 *For all $\{\gamma_{ij}\}_{i,j\in\mathcal{E}}$,*

$$\hat{\ell}_Y(Y;\Gamma,\widetilde{\rho}) \;\geq\; \ell_Y(Y;\rho),$$

with equality[2] iff

$$\begin{aligned}
\gamma_{ij} &= \arg\min_{\{\gamma_{ij}>0\}} \widetilde{\pi}\left(Y;\widetilde{\rho},\{\gamma_{ij}\}\right)\\
&= \left.\frac{\partial\rho\left(z^2\right)}{\partial z^2}\right|_{z=\|\boldsymbol{y}_i-\boldsymbol{y}_j\|_2}.
\end{aligned}$$

Corollary 3.2.1 *For any ρ, there exists a set of attention weights $\Gamma^* \equiv \{\gamma_{ij}^*\}_{i,j\in\mathcal{E}}$ such that*

$$\arg\min_Y \ell_Y(Y;\rho) \;=\; \arg\min_Y \hat{\ell}_Y(Y;\Gamma^*,\widetilde{\rho}).$$

***Proof*:** Both Lemma 3.2 and Corollary 3.2.1 follow directly from principles of convex analysis and Fenchel duality (Rockafellar, 1970). In particular, any concave, non-decreasing function $\rho:\mathbb{R}_+\to\mathbb{R}$ can be expressed via the variational decomposition

$$\begin{aligned}
\rho\left(z^2\right) &= \min_{\gamma>0}\left[\gamma z^2 - \rho^*\left(\gamma\right)\right]\\
&\geq \gamma z^2 - \widetilde{\rho}\left(\gamma\right), \qquad (37)
\end{aligned}$$

where γ is a variational parameter whose optimization defines the decomposition, and $\widetilde{\rho}$ is the concave conjugate of ρ. From a visual perspective, (37) can be viewed as constructing $\rho\left(z^2\right)$ as the minimal envelope of a series of quadratic upper bounds, each defined by a different value of γ. And for any fixed γ, we obtain a fixed upper bound once we remove the minimization operator. By adopting $z = \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2$ for all $i,j\in\mathcal{E}$ we obtain (16), which by construction satisfies (17). And (18) follows by noting that at any optimal γ^*, the upper bound satisfies

$$\gamma^* z^2 - \widetilde{\rho}(\gamma^*) = \rho\left(z^2\right), \qquad (38)$$

i.e., it is tangent to ρ at z^2, in which case γ^* must be equal to the stated gradient (or subgradient).

And finally, in terms of Corollary 3.2.1, let $Y^* = \arg\min_Y \ell_Y(Y;\rho)$. We may then simply apply Lemma 3.2 to form the bound

$$\hat{\ell}_Y(Y^*;\Gamma,\widetilde{\rho}) \;\geq\; \hat{\ell}_Y(Y^*;\Gamma^*,\widetilde{\rho}) \;=\; \ell_y(Y^*;\rho), \qquad (39)$$

where Γ^* denotes a diagonal matrix with optimized γ_{ij}^* values along the diagonal. Therefore $\hat{\ell}_Y(Y;\Gamma^*,\widetilde{\rho})$ so-defined achieves the stated result. ■

Lemma 3.3 *Provided that $\alpha \leq \frac{1}{2}\left\|\lambda B^\top\Gamma^{(k)}B + I\right\|_2^{-1}$, the iterations (20) and (21) are such that*

$$\ell_Y(Y^{(k)};\rho) \;\geq\; \ell_Y(Y^{(k+1)};\rho).$$

***Proof*:** We will first assume that no Jacobi preconditioning is used (i.e., $\tilde{D}\equiv I$); later we will address the general case. Based on Lemma 3.2 and (18), as well as the analogous update rule for $\Gamma^{(k+1)}$ from (20), it follows that

$$\hat{\ell}_Y\left(Y^{(k)};\Gamma^{(k+1)},\widetilde{\rho}\right) \;=\; \ell_Y(Y^{(k)};\rho). \qquad (40)$$

Now define $\Psi(Y) \triangleq \hat{\ell}_Y\left(Y;\Gamma^{(k+1)},\widetilde{\rho}\right)$ and

$$\begin{aligned}
\hat{\Psi}(Y) \;\triangleq\; & \Psi\left(Y^{(k)}\right) + \qquad (41)\\
& \nabla\Psi\left(Y^{(k)}\right)^\top\left(Y - Y^{(k)}\right) + \tfrac{\mathcal{L}}{2}\left\|Y - Y^{(k)}\right\|_{\mathcal{F}}^2,
\end{aligned}$$

where Ψ has Lipschitz continuous gradients with Lipschitz constant $\mathcal{L}$ satisfying

$$\|\nabla\Psi(Y_1) - \nabla\Psi(Y_2)\|_{\mathcal{F}} \leq \mathcal{L}\,\|\Psi(Y_1) - \Psi(Y_2)\|_{\mathcal{F}} \qquad (42)$$

[2] If ρ is not differentiable, then the equality holds for any γ_{ij} which is an element of the subdifferential of $-\rho(z^2)$ evaluated at $z = \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2$.

for all Y_1 and Y_2. We may then conclude that

$$\hat{\Psi}(Y) \;\geq\; \Psi(Y) \;\geq\; \ell_Y(Y;\rho), \tag{43}$$

with equality at the point $Y = Y^{(k)}$. Note that the first inequality in the above expression follows from basic results in convex analysis (e.g., see (Bubeck, 2014)[Lemma 3.4]), while the second comes from (17). Consequently, we have that

$$\min_Y \hat{\Psi}(Y) \;=\; \hat{\Psi}(Y^*) \;\leq\; \hat{\Psi}(Y^{(k)}) \;=\; \ell_Y\left(Y^{(k)};\rho\right), \tag{44}$$

where

$$Y^* \;\triangleq\; \arg\min_Y \hat{\Psi}(Y) \;=\; Y^{(k)} - \tfrac{1}{\mathcal{L}}\nabla\Psi\left(Y^{(k)}\right), \tag{45}$$

noting that the optimal solution can be obtained by simply differentiating with respect to Y and equating to zero. We may therefore choose

$$Y^{(k+1)} = Y^{(k)} - \tfrac{1}{\mathcal{L}}\nabla\Psi\left(Y^{(k)}\right) \tag{46}$$

to guarantee that

$$\ell_Y\left(Y^{(k)};\rho\right) \;\geq\; \hat{\Psi}\left(Y^{(k+1)}\right) \;\geq\; \ell_Y\left(Y^{(k+1)};\rho\right). \tag{47}$$

And if we adopt the step-size $\alpha = \frac{1}{\mathcal{L}}$, then (46) is equivalent to (21), excluding the Jacobi preconditioner. Hence we must only enforce the Lipschitz constraint (42) to guarantee monotonicity. This is equivalent to the requirement that $\mathcal{L}I - \nabla^2\Psi\left(Y^{(k)}\right) \succeq 0$ for all k, which computes to $\mathcal{L}I \succeq 2\left(I + \lambda B^\top\Gamma^{(k)}B\right)$. Setting $\mathcal{L}$ to be greater than or equal to the maximum singular value of $2\left(I + \lambda B^\top\Gamma^{(k)}B\right)$ satisfies this objective, which then leads to the step-size bound $\alpha \leq \frac{1}{2}\left\|\lambda B^\top\Gamma^{(k)}B + I\right\|_2^{-1}$.

And finally, if we reintroduce the non-trivial Jacobi preconditioner $\tilde{D}^{-1} = (\lambda D + I)^{-1} \neq I$, we need only redefine the bound from (41) as

$$\begin{aligned}\hat{\Psi}(Y) \;\;\triangleq\;\; & \Psi\left(Y^{(k)}\right) + \nabla\Psi\left(Y^{(k)}\right)^\top\left(Y - Y^{(k)}\right) \\ & + \tfrac{\mathcal{L}}{2}\left(Y - Y^{(k)}\right)^\top\left(\tilde{D}^{(k+1)}\right)^2\left(Y - Y^{(k)}\right).\end{aligned} \tag{48}$$

And because $\left(\tilde{D}^{(k+1)}\right)^2 \succeq I$, (43) still holds and the same conclusions follow through as before. The only major difference is that now

$$Y^* = \arg\min_Y \hat{\Psi}(Y) = Y^{(k)} - \tfrac{1}{\mathcal{L}}\left(\tilde{D}^{(k+1)}\right)^{-1}\nabla\Psi\left(Y^{(k)}\right) \tag{49}$$

leading to the exact preconditioned update rule from (21) once we adopt $\alpha = \frac{1}{\mathcal{L}}$ and rearrange terms. And in fact, a larger step-size range is actually possible in this situation since the upper bound from (48) holds for smaller values of $\mathcal{L}$ (note that all diagonal values of $\tilde{D}^{(k+1)}$ are greater than one). ■

References

Andrews, D. and Mallows, C. Scale mixtures of normal distributions. *J. Royal Statistical Society: Series B*, 36(1): 99–102, 1974.

Batson, J., Spielman, D. A., and Srivastava, N. Twice-ramanujan sparsifiers. *SIAM Journal on Computing*, 41 (6):1704–1721, 2012.

Bubeck, S. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, 2014.

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

Chen, J., Zhu, J., and Song, L. Stochastic training of graph convolutional networks with variance reduction. In *International Conference on Machine Learning*, pp. 942–950, 2018b.

Dai, H., Kozareva, Z., Dai, B., Smola, A., and Song, L. Learning steady-states of iterative algorithms over graphs. In *International conference on machine learning*, pp. 1106–1114, 2018.

Emerson, T. The second international chinese word segmentation bakeoff. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing, SIGHAN@IJCNLP*, 2005.

Gu, F., Chang, H., Zhu, W., Sojoudi, S., and Ghaoui, L. E. Implicit graph neural networks. In *Advances in Neural Information Processing Systems, NeurIPS*, 2020.

Lee, Y. T. and Sun, H. An sdp-based algorithm for linear-sized spectral sparsification. In *Proc. ACM Symposium on Theory of Computing*, pp. 678–687, 2017.

Li, Y., Jin, W., Xu, H., and Tang, J. Deeprobust: A pytorch library for adversarial attacks and defenses. *arXiv preprint arXiv:2005.06149*, 2020.

Palmer, J., Wipf, D., Kreutz-Delgado, K., and Rao, B. Variational EM algorithms for non-Gaussian latent variable models. *Advances in Neural Information Processing Systems*, 2006.

Pei, H., Wei, B., Chang, K. C.-C., Lei, Y., and Yang, B. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2019.

Rockafellar, R. T. *Convex Analysis*. Princeton University Press, 1970.

Rong, Y., Huang, W., Xu, T., and Huang, J. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2019.

Tang, J., Sun, J., Wang, C., and Yang, Z. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 807–816, 2009.

Widder, D. V. *The Laplace Transform*. Princeton university press, 2015.

Yang, Z., Cohen, W., and Salakhudinov, R. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pp. 40–48, 2016.

Zhang, X. and Zitnik, M. Gnnguard: Defending graph neural networks against adversarial attacks. *Advances in Neural Information Processing Systems*, 33, 2020.