Improving Detection of Hateful Users on Twitter using Attention and ReFex

Jestin Ma
jestinm@stanford.edu

Guillaume Nervo
gnervo@stanford.edu

Jason Zheng
jzzheng@stanford.edu

Abstract

We attempt to improve existing methods of detecting hateful Twitter users by exploring different graph neural networks (GNN) frameworks and recursive structural graph features. Specifically, we examine the performance of spatial convolutional graph neural networks (GCN, GAT, FeaStNet) and recurrent graph neural networks (Gated Graph Sequence Neural Networks) with respect to the baseline GraphSage network specified in Ribeiro et. al (2018) [5]. We demonstrate that FeaStNet, with and without ReFeX feature augmentation, is able to exceed the performance of the baseline GraphSage model, providing a $\sim 2.2\%$ increase in accuracy, $\sim 4.2\%$ increase in F1 score and $\sim 1\%$ increase in recall after 500 epochs. Moreover, including ReFeX features substantially improves the baseline GraphSage model, but offers negligible benefits for other models. These results are significant as they indicate that taking advantage of structural graph properties, whether through attention mechanisms or explicit features, can further improve GNN detection of hateful users.

1 Introduction

In hindsight of widespread domestic and international content integrity abuse on social media platforms during the 2016 United States presidential elections, managing elections integrity has become a high priority for the forthcoming 2020 presidential elections. At the same time, data usage integrity and user privacy violations have come under intense scrutiny by Congress and the general public, increasing the need to investigate and develop alternate methods of detecting fraudulent and harmful actors on social media platforms without interfacing directly with content-based data.

Our original motivation in our project was to detect election fraud on social media platforms like Twitter, but as we analyzed available Twitter datasets, such as the publicly available Twitter Election Integrity Dataset (2018) [6], we found that these networks exhibited strong bias, including only egonets of flagged fraudulent users. We initially planned to reduce this bias by gaining access to the Twitter Developer API, which would allow us to run k-depth fanouts of retweets, mentions and replies for each of these networks to get more connected users whom we may manually procure labels for. This would allow us to produce one network for each fraudulent user, with a mix of fraud, non-fraud and unknown users. However, our issues here were two-fold. First, we were unable to gain access to Twitter Developer API, and second, we realized that manual labels for fanout-nodes from fraudulent nodes on the scale of the election integrity dataset would be infeasible without additional human resources.

As such, we pivoted to detecting hateful users on Twitter, a tangentially related problem which has existing datasets with more labels and less bias. By improving on existing models in detection of hateful users, we are able to flag potentially problematic user interactions on the platform, which may be useful in detecting harmful actors during the 2020 election.

CS224W (2019), Stanford, CA.

2 Problem Statement

Using the same dataset as Ribeiro et. al [5] (whose work we explore below), we aim to show that the addition of ReFeX aggregated graph features and the use of different convolutional layers will allow us to improve prediction metrics on hateful Twitter users. To do this, we maintain Ribeiro et. al's dataset (retweet network, GloVe vectors, and node-level features extracted from Tweet language and user activity data) and alter only graph features and graph convolutional methods.

Specifically, our model takes in a single Tweet graph network with 100,386 users and 2,286,592 Tweet interactions. Each user has 320 user features by default, and 364 when aggregated with ReFeX features. We experiment with various convolutional architectures to perform a binary classification on whether each of 100,386 users is hateful or not.

3 Related Work

3.1 Ribeiro et. al Detecting Hateful users using GraphSage

Our project follows closely with the experiments of Ribeiro et. al (2018) [5] whose work focused on detecting hateful users using node-level and graph-level convolutions through GraphSage. Hateful speech online is often nuanced and may not be immediately obvious from only text-based features. As an example of nuanced hate speech, they consider the following tweet, which is in reply to a previous tweet about the Holocaust:

> *Timesup, yall getting w should have happened long ago*

As such, Ribeiro et. al propose an analysis of context-level structures, looking at user account details, activities and connections. In their analysis of hateful users, they found that hateful users have newer accounts, have short and strong bursts of activity, don't mass spam, are generally more central in the tweet interaction graph, use intense vocabulary, and are densely connected with one another.

As a result of the realization that hateful users are densely connected, they also propose the use of a GraphSage model in the classification of hateful users, in order to potentially capture network connections between hateful users. Doing so, they were able to achieve large boosts in in their accuracy, F1 score and AUC for hateful user classification, with respect to conventional classification methods that do not utilize graph structures. This is especially relevant to us, since their initial work on utilizing tweet networks informs the extensions we propose in our own experiments. We extend on Ribeiro et. al's work by focusing on the ability of different neural graph techniques to better capture hateful user network connections.

3.2 Survey of graph neural networks

Various GNN model architectures have been proposed and subsequently organized into a taxonomy of model architectures and frameworks, such as recurrent graph neural networks (RecGNNs), convolutional graph neural networks (ConvGNNs), graph autoencoders (GAEs), and spatial-temporal graph neural networks (STGNNs) [10]. RecGNNs and ConvGNNs dominate the supervised learning and semi-supervised learning space of node-level and graph-level classification problems. GAEs act as an unsupervised learning framework of network embeddings and graph generative features. STGNNs aim to learn hidden patterns graphs in which spatial as well as temporal data is relevant (like forecasting). We focus on exploring RecGNNs and ConvGNNs, specifically spatially-based (graph convolutions as propagating information from N-hop neighbors) instead of spectral-based ones.

3.3 Detecting bots

Various systems and tools have been invented to detect bots. For example, Varol et al. present a system, Botometer [7], that trains a binary classifier on several per-user and network features. While detecting bots has been well researched, the proposed system likely does not work effectively for 'troll' accounts, i.e. actual users who may be assisted by software [3]. The proposed feature set likely does not discriminate between a troll and a genuine user well enough. However, the classes of features used (network-level, user-level, content, temporal, and friends) may be effective in detecting

hateful users. As such, we explore the use of aggregated network features as part of a local classifier used in conjunction with a message passing network.

3.4 Network Indicators of Bad Actors

Bad actors on a network perform sybil attacks, in which malicious nodes in networks disingenuously assume fake identities and undermine some notion of reputation, trust, and truth. Alvisi et al. characterized sybil attacks on networks by structural properties of social graphs: popularity, small world property, grouping coefficient, and conductance, the last of which is the only reliable characteristic [1]. However, this property fails to hold against denser and wider attacks, especially if the attack in question is from hateful users who will influence the amount of hate speech in the network and who might not be operating as part of an organized attack.

4 Methodology and Approach

4.1 Dataset

We utilize the dataset presented by Ribeiro et. al [5], which consists of 100,386 Twitter users with 2,286,592 retweet interactions. Specifically, we train on a annotated subgraph, which contains 544 hateful users, 4427 normal users, 3471 hateful neighbors (neighbors of hateful users) and 33564 normal neighbors (users who are not connected to hateful users). All users in the dataset take on a label $x \in 0, 1, 2$, where 2 denotes a hateful users, 0 denotes a normal user, and 1 denotes an unlabeled user.

Each user additionally has a 320-dimensional feature vector aggregated from GloVe text-based features from 200 recent tweets, as well as account-level information such as gender, location of content creators, and network centrality. In our experiments, we classify users as hateful ($x \in 2$) or non-hateful ($x \notin 2$), using the retweet graph from the dataset edges, as well as the user features. In subsequent tests, we also augment our dataset with features extracted from the retweet network using ReFeX, giving us a 364-dimensional per-user feature vector. This allows us to introduce additional network information directly into local classification in conjunction with message passing schemes described in 4.4.

4.2 Feature Extraction using RolX and ReFeX

Initially, we wanted to discover structural graph properties for each node through the RolX (Structural Role Extraction & Mining in Large Graphs) procedure [2].

As described by the RolX algorithm [Algorithm 1], we extract each node's local characteristics such as its degree, the number of edges in its egonet, and the number of edges leaving its egonet.

We then look at the node's 1-hop neighbours and aggregate new features as a function of the neighbourhood, and continue until a set iteration. These features are useful not only as additional features for our model, but also because they may give us insight into our dataset.

Algorithm 1: ReFeX recursive feature extraction

Data: G (Graph), K (number of steps)
Initialize an empty matrix M$\in \mathbb{M}_{|V|*n^{K+1}}$;
k$\leftarrow$ 1;
Compute: $\forall v \in V$:
$deg_v = deg(v)$, $deg_{v_{in}}$ = internal egonet edges, $deg_{v_{out}}$ = external egonet edges
for $v \in V$ **do**
  M[v,0:3]=$deg_v, deg_{v_{in}}, deg_{v_{out}}$
while *k<K* **do**
  for $v \in V$ **do**
    $M[v, 3^k : 3^{k+1}] = [M[v, 0 : 3^k]; \frac{1}{|N(u)|} \sum_{u \in N(v)} M[u, 0 : 3^k]; \sum_{u \in N(v)} M[u, 0 : 3^k]]$
Result: return

After extracting these features, we find the roles for each node. By first setting the number of roles to r, we want the dimensionality of our previous matrix to be $M \in \mathbb{R}^{|V|*r}$. We use Non-Negative Matrix Factorization to solve the following problem:

$$argmin_{G,F}||M - GF||_F \tag{1}$$

with constraints $G \in \mathbb{M}_{|V|*r}$, $F \in \mathbb{M}_{r*3^{K+1}}$, $G \geq 0$ and $F \geq 0$.

Finally, during RefeX extraction we use pruning to keep only relevant, unique features.

4.3 Model Specifications

For all specifications, we define the graph as $G(V, E)$ with input features $\{x_v, \forall v \in V\}$ and intermediary/output features h_v^k at layer k. Let σ be the nonlinear activation function *ReLU*.

4.3.1 Baseline GraphSage

We replicate the baseline GraphSage network from Ribeiro et. al [5], but with slight modifications. Define the features of node v, h_v^k as:

$$h_v^k = \sigma(W^k \cdot CONCAT(h_v^{k-1}, h_{N(v)}^k)$$

where $h_{N(v)}^k = \frac{1}{|N(v)|} \sum_{u \in N(v)} \sigma(A^k h_u + b)$, the mean aggregation of v's neighbors passed through a linear layer and ReLU activation. W^k, A^k, b are learnable parameters. Rather than sampling neighbors, we aggregate all neighbors, trading off efficiency in order to maximize information propagation and to create a strong baseline to improve from.

The use of GraphSage is motivated by the fact that it (a) allows us to use the neighborhood of the nodes even if neighbors aren't labeled as hateful or not; (b) it takes advantage of the retweet graph structure which indicates hateful users retweet other hateful users often.

4.3.2 Graph Convolutional Network

Another baseline model we selected was a GCN, in order to judge a very simple spatial convolutional GNN. We formulate the convolutional layer as

$$h^{k+1} = \sigma(D^{-1/2} A D^{-1/2} h^k W^k)$$

where $D^{-1/2} A D^{-1/2}$ is the normalized adjacency matrix of G. W^k is a learnable parameter.

4.3.3 Graph Attention Network

Unlike GraphSage and GCN, in which information contributions of $N(v) \to v$ are identical and predetermined respectively, Graph Attention Networks assumes that information contributions differ between neighbors and thus computes attention weights for $\{(v, u) : \forall u \in N(v)\}$ [8].

We formulate the attentional layer as

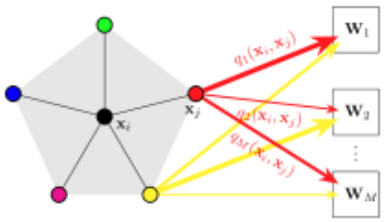
$$h_v^k = \sum_{u \in N(v)} \alpha_{uv} W^k h_v^k$$

The attention weights are $\alpha_{uv} = softmax_j(\sigma(a(CONCAT(W^k h_v, W^k h_u))))$, where W^k, a are learnable parameters.

4.3.4 FeaStNet

Furthermore, we test the application of attention mechanisms to traditional convolution neural networks. Specifically, we employ FeaStNet, which determines the correspondence between filter weights and neighbor features [9]. Instead of a typical 1-to-1 mapping from neighbor to a filter weight, FeaSt learns the importance of a neighbor as a function of the features in the previous layer (see Figure 1).

Figure 1: In FeaStNet, neighbors x_j use all weights; each weight's importance is learned as q.



The attentional layer is defined as

$$h_v^k = \frac{1}{|N(v)|} \sum_{u \in N(v)} \sum_{m=1}^{M} \alpha_{uv}^m W_m^k h_v^k$$

with m heads of attention. The attention weights are $\alpha_{uv}^m = softmax_j(\sigma(a_m(h_v^k - h_u^k) + b_m))$, where W_m^k, a_m, b_m are learnable parameters.

Notice that this is very similar to the Graph Attention Network layer definition. One difference is that the attention to neighbors is translation invariant, which is important if the difference/relationship between nodes' features is more important than the actual values. (e.g. coordinates, time, GloVe vectors, sentiment) [9].

4.3.5 Other Convolutional Methods

We experimented with several other models that were readily available, such as Gated Graph Sequence Neural Networks [4], convolutional ARMA filters, and approximate personalized propagation of neural features. However, as they did not perform exceptionally well or poorly, we neither analyze their results nor elaborate on their architectures.

5 Results and Analysis

5.1 RolX Analysis of Dataset

Through our analysis of RolX features in our dataset, we demonstrate some of the key characteristics of our dataset. The first important fact is that hate users tend to be strongly connected to each other, an observation which confirms previous analysis done in Ribeiro et. al [5]. This is evident in a representation of the egonet of hateful users [Figure 2].

Figure 2: Egonet of hateful users in our Tweet dataset. Notice that hateful users tend to be densely connected.

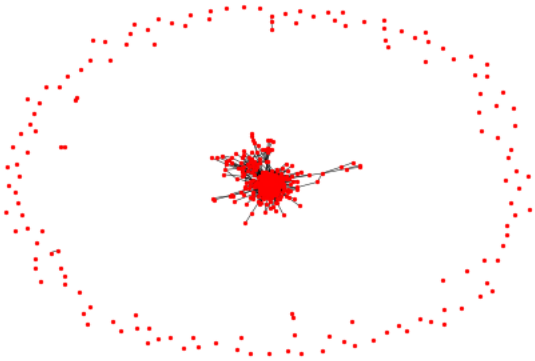
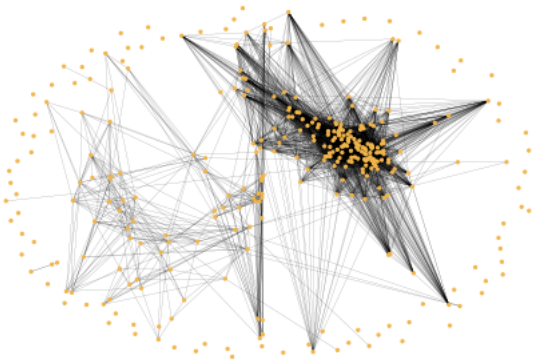


Figure 3: Ego net of the role with the least number of representatives, each of the roles are densely connected.



What we find particularly interesting here, is that there exists a key cluster of hateful users who connect with similar unlabeled users, and each other. We believe that disconnected nodes here are a result of (a) our visualization sampling only a small part of the dataset and (b) the fact that the dataset creators did not completely annotate the dataset, so there may be edges that do not exist when visualizing the dataset. Another possibility is that these disconnected nodes may be outliers, that are not related to other hateful users. However, we cannot fully confirm this as we only visualize the 1-hop egonet. It is possible that these nodes are still connected, but only at a greater hop distance, with intermediary unlabeled hateful nodes connecting them (which may be any of 3 classes in the dataset "normal", "hateful" and "undefined"). While the mean degree of hateful users (≈ 45), and the mean degree over all users (≈ 43) are pretty close, the centrality in hateful nodes can be characterized by the eigenvector centrality of each node. A node's eigenvector centrality represents the influence of a node in the network. We obtain a mean eigenvector centrality of 0.0014 for users in general while getting 0.00175 for hateful users, indicating that hateful users tend to have more influence.

As for role detection in the graph, analysis is limited. Our approach detected seven roles in our graph structure. Hateful users are not concentrated in few roles, but are rather proportionally (in respect to the number of nodes in each role) divided in the different roles. There doesn't seem to be a direct explanation to each roles, but one thing to be mentioned is that all the egonet of each roles are densely connected, which is demonstrated in figure 3.

5.2 Experiments

5.2.1 Experimental settings and methodology

We follow the evaluation framework presented by Ribeiro et. al [5], attempting to label both hateful and non-hateful users. For each model architecture we describe in section 4, we train the model on the retweet network and user features for 500 epochs. We repeat the same methodology using the same retweet network but with features augmented by our ReFeX output. When training, we account for the class imbalance in our dataset (90% less hateful labels than non-hateful labels) by taking a class-wise weighted negative loss. Finally, during decision time, our binary classifier's discrimination threshold is whether our network ranks/scores a user higher as a hateful or non-hateful user. We perform a 5-fold cross validation over the dataset, and report loss, and the mean and standard deviation of accuracy, F1-score, and recall for each epoch of training.

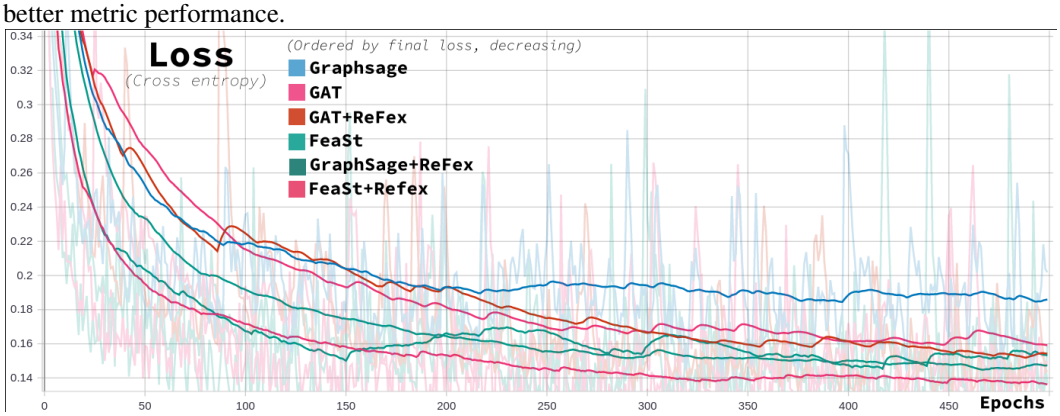
5.2.2 Experiment results and discussion

Our results are shown in Table 1, with each row representing a different model with possibly additional RefEx features. The corresponding loss of tested models (excluding GCN and Gate) are shown in Figure 4. We find that the FeaSt network without extra ReFex features performs better than other tested models across all metrics, achieving a F1-score of 73.9 versus the baseline GraphSage's 69.7 (a $\sim 4.2\%$ increase). Moreover, FeaStNet performed the best consistently epoch after epoch with relatively low standard deviation.

Table 1: Accuracy, F1, and recall results and standard deviations for predicting hateful vs non-hateful users. FeaSt without additional ReFex features performs the best across the board, suggesting the importance of translational invariance and attention mechanisms.

Model	With RefEx?	Accuracy	F1	Recall	500 epoch runtime
GraphSage (baseline)		90.1 ±2.4	69.7 ±5.0	97.7 ±2.0	18m
GraphSage (baseline)	x	91.8 ±2.0	73.5 ±4.3	97.5 ±2.0	17m
GCN		79.8 ±2.4	51.6 ±2.8	96.8 ±2.1	**9m**
Gate		75.8 ±7.6	46.9 ±5.9	88.3 ±6.4	1h5m
GAT		90.0 ±1.5	69.1 ±2.8	**98.7 ±1.2**	25m
GAT	x	90.2 ±1.5	69.7 ±2.9	**98.7 ±1.2**	25m
FeaSt		**92.03 ±1.6**	**73.9 ±3.7**	**98.7 ±1.4**	20m
FeaSt	x	91.92 ±1.7	73.4 ±3.9	98.0 ±2.0	41m

Figure 4: We display the loss of each model configuration per epoch over 500 epochs of training, excluding GCN and Gated GCN (loss was much higher). Note that lower loss does not correspond to better metric performance.



We believe that this result is because FeaStNet uses a more complex attention mechanism in weighing neighbors during aggregation. GraphSage weighs each neighbor identically when aggregating their messages. An even more extreme example is GCN, in which each neighbor has a fixed, non-parametric weight; its performance clearly suffers (although it is the simplest and thus fastest model).

On the other hand, while Graph Attention Networks and FeaStNet are comparably similar, GAT performs at around the same level as the GraphSage baseline and performs as well as FeaSt on only recall, even when the GAT network is augmented with ReFex features. This indicates that GAT had lower precision than FeaSt, i.e. it identified many more false positive/hateful users. One explanation can be found in the difference between FeaStNet's and GAT's attention mechanism. Recall that while the FeaStNet and GAT layers were architecturally very similar, the FeaStNet attention mechanism we used preserves translational invariance of the attention values. This is particularly important if the relationship between neighbors' features, such as GloVe vectors and tweet sentiment, is better than their actual values as discriminators of whether a user is hateful.

On the topic of augmenting the input data with ReFex features of each node, we hypothesized that including more structural graph features (that might characterize hateful users) would help GNNs learn more accurately. ReFex features interestingly had mixed results. ReFeX feature augmentation did not improve the performance of FeaStNet, but GAT and GraphSage benefited (with GraphSage's F1-score increasing from 69.7% to 73.5%). We postulate that this difference in gain is due to the naive neighbor aggregation GraphSage uses compared to the more complex, attention-based weighing done by GAT and FeaStNet. ReFex features can help GraphSage gain a subtler view of a node's neighborhood, but won't benefit models like GAT and FeaStNet that already account for neighbors via attention. This difference and explanation implies that including nodes' structural graph features and attending to a nodes' neighbors are potentially functionally equivalent. That is, they both aim to better capture the structure and importance of subnetworks of each node.

6 Conclusion

We have demonstrated that a learned FeaStNet network consistently and noticeably improves upon the existing GraphSage network (as well as other networks) on the Hateful Twitter users dataset across all metrics. In particular, we note a 4.2% F1-Score increase. Regarding the use of ReFeX features, they provide a 4.1% reduction in loss to FeaStNet, but negligible effects on all other metrics. On the other hand, the addition of ReFeX was able to improve the performance of the baseline GraphSage model, indicating that the addition of these features likely helps models which do not already have an attention-like mechanism to learn the importance of a node's neighbors.

Moving forward, we would like to continue our experimentation by testing different aggregation methods in message passing, such as LSTM aggregation and max pooling. With access to the Twitter API and original tweets, we would also like to augment the dataset by taking into account other kinds of networks and edges, such as mention, reply and follower graphs. This would allow us to deepen our model's understanding of hateful users on Twitter, potentially build out new networks 'in the wild', and determine the performance of our model on such new networks, such as the 2020 election Tweet livestreams.

7 Contributions

All members of our group contributed equally.
Code contributions can be found at `https://github.com/jestjest/cs224w-project/`

8 Acknowledgments

We would like to thank Michele for initial guidance and comments on our initial incomplete dataset; Alexis for providing checkpoint feedback and pointing us to another more practical and complete Twitter dataset; and Prof. Leskovec and the rest of the CAs for managing the course.

References

[1] ALVISI, L., CLEMENT, A., EPASTO, A., LATTANZI, S., AND PANCONESI, A. Sok: The evolution of sybil defense via social networks. In *2013 IEEE Symposium on Security and Privacy, SP 2013* (2013).

[2] HENDERSON, K., GALLAGHER, B., ELIASSI-RAD, T., TONG, H., BASU, S., AKOGLU, L., KOUTRA, D., FALOUTSOS, C., AND LI, L. Rolx: Structural role extraction mining in large graphs. *EECS* (2012).

[3] IM, J., CHANDRASEKHARAN, E., SARGENT, J., LIGHTHAMMER, P., DENBY, T., BHARGAVA, A., HEMPHILL, L., JURGENS, D., AND GILBERT, E. Still out there: Modeling and identifying russian troll accounts on twitter. *CoRR abs/1901.11162* (2019).

[4] LI, Y., ZEMEL, R., BROCKSCHMIDT, M., AND TARLOW, D. Gated graph sequence neural networks. In *Proceedings of ICLR'16* (April 2016).

[5] RIBEIRO, M. H., CALAIS, P. H., SANTOS, Y. A., ALMEIDA, V. A. F., AND JR., W. M. Characterizing and detecting hateful users on twitter. *CoRR abs/1803.08977* (2018).

[6] TWITTER. Information operations.

[7] VAROL, O., FERRARA, E., DAVIS, C., MENCZER, F., AND FLAMMINI, A. Online human-bot interactions: Detection, estimation, and characterization, 2017.

[8] VELIČKOVIĆ, P., CUCURULL, G., CASANOVA, A., ROMERO, A., LIÒ, P., AND BENGIO, Y. Graph Attention Networks. *International Conference on Learning Representations* (2018).

[9] VERMA, N., BOYER, E., AND VERBEEK, J. Dynamic filters in graph convolutional networks. *CoRR abs/1706.05206* (2017).

[10] WU, Z., PAN, S., CHEN, F., LONG, G., ZHANG, C., AND YU, P. S. A comprehensive survey on graph neural networks. *CoRR abs/1901.00596* (2019).