Graph-Hist: Graph Classification from Latent Feature Histograms with Application to Bot Detection

Thomas Magelinski, David Beskow, Kathleen M. Carley
CASOS, School of Computer Science, Carnegie Mellon University
5000 Forbes Ave. Pittsburgh, PA, 15213
{tmagelin, dbeskow, kathleen.carley}@cs.cmu.edu

Abstract

Neural networks are increasingly used for graph classification in a variety of contexts. Social media is a critical application area in this space, however the characteristics of social media graphs differ from those seen in most popular benchmark datasets. Social networks tend to be large and sparse, while benchmarks are small and dense. Classically, large and sparse networks are analyzed by studying the distribution of local properties. Inspired by this, we introduce Graph-Hist: an end-to-end architecture that extracts a graph's latent local features, bins nodes together along 1-D cross sections of the feature space, and classifies the graph based on this multi-channel histogram. We show that Graph-Hist improves state of the art performance on true social media benchmark datasets, while still performing well on other benchmarks. Finally, we demonstrate Graph-Hist's performance by conducting bot detection in social media. While sophisticated bot and cyborg accounts increasingly evade traditional detection methods, they leave artificial artifacts in their conversational graph that are detected through graph classification. We apply Graph-Hist to classify these conversational graphs. In the process, we confirm that social media graphs are different than most baselines and that Graph-Hist outperforms existing bot-detection models.

Introduction

Given the success of traditional machine learning, interest in geometric learning has grown in recent years. Geometric learning seeks to extend machine learning models beyond Euclidean data to include objects such as graphs, point clouds, and manifolds. Non-Euclidean data structures are information-rich, as they can describe data that traditional structures cannot. For example, a traditional data structure may contain attributes of a group of individuals, while a graph or network can also encode the *relationships* between the individuals. Thus, new algorithms to leverage this type of information can bring new insights.

There are three main problems for graphs in particular: node classification, link prediction, and graph classification. Here, we focus on the last task, graph classification. In this problem, each input sample is a graph, which has a corresponding category or label. The goal is to create a model which takes an entire graph as an input, and assigns it to the correct class.

Graph classification is gaining interest in part due to the variety of domains it may be applied to. The same models that can classify proteins based on their structure may also be used to classify social media conversations. We identify a new application which is highly relevant in today's sociopolitical landscape: bot classification in social media. Automated accounts called bots are increasingly used in online information operations to manipulate both networks (virtual social links) and the narratives that transit these networks. Since bots operate *through* networks, their network structure can be used to identify them.

While many problems regarding social media data can be posed under a graph classification framework, few prior models focus on this domain. Much of the prior work in graph classification focuses on benchmark data that does not reflect the typical structure of social media data. Specifically, social media graphs or networks tend to have large nodesets and low density, while benchmark datasets tend to have less than 100 nodes and are quite dense.

In this work, we develop a new graph classification architecture inspired by classical network analysis. In analysis of large networks, it is common practice to calculate local features and study the distribution. Here, we use an end-to-end graph-convolutional architecture to extract local latent features and classify the graph based on the distribution of these features. Due to the high dimensionality of the feature space, we instead use one-dimensional cross sections of the distribution in the form of a multi-channel histogram. Since this procedure classifies graphs based on histograms of latent features, it has been named Graph-Hist.

In the following sections, we review prior work in graph classification, explain our architecture, demonstrate Graph-Hist's ability to achieve state of the art results on social media benchmarks, and finally demonstrate real-world application of our model through a case study of bot classification on Twitter data.

Related Work

The field of graph learning has expanded rapidly since the notation for Graph Neural Networks, or GNNs, was first introduced by Gori et al. (Gori, Monfardini, and Scarselli 2005). The work from then to 2018 is well summarized by Wu et. al (Wu et al. 2019).

GNNs for graph classification are typically based on a type of *graph convolution*, leading to their other name: Graph Convolutional Networks, or GCNs. Traditional convolutional networks have proved extremely successful at learning shape features for problems in the Euclidean domain, such as image classification. However, translating this operation to the graph domain is difficult due to irregularities in graph structure (Krizhevsky, Sutskever, and Hinton 2012). Graph convolutions usually fall into one of two approaches: spectral or spatial. Spectral methods stem from efforts to extend traditional signal processing techniques to graph signals, or Graph Signal Processing (Shuman et al. 2013). Spectral approaches typically use the symmetric normalized Laplacian, shown in Equation 1. Many spectral-based approaches like ChebNet relied on eigenvalue calculations, making them computationally costly (Defferrard, Bresson, and Vandergheynst 2016). Spatial methods on the other hand, operate on the local structure of the graph. In spatial methods such as GraphSage, nodes aggregate information from their neighbors (Hamilton, Ying, and Leskovec 2017).

Kipf and Welling introduced a model that bridged the gap between the two: it is an approximation of a spectral convolution, but it is localized in space (Kipf and Welling 2016). This model uses the propagation rule shown in Equation 2, which is discussed in the following section. Many architectures for graph classification now build upon this convolutional structure, including two works we draw from here: SortPool and Capsule Graph Networks (Zhang et al. 2018; Xinyi and Chen 2019).

Zhang et al, replaces the normalized Laplacian with the random-walk Laplacian, and draws parallels to the Weisfeiler-Lehman subtree kernel (Shervashidze et al. 2011). This effectively gives node embeddings, which they then sort and either truncate or pad to a fixed size, hence the name SortPool. While the sorting procedure gives some spatial relationship to the nodes, the truncating/padding procedure either drops important information, or adds erroneous data. This is especially problematic when datasets have high variance in graph size, which is often the case in social media datasets. Xinyi and Chen have also used this GNN structure, but applied attention to handle the differences in graph size (Xinyi and Chen 2019).

Tixier et al. take a different approach (Tixier et al. 2017). They first assume that node embeddings are given. Node embeddings can be obtained in a number of unsupervised ways, most of which attempt to preserve the network-based distance between nodes in the embedded space through operations like random walks (Perozzi, Al-Rfou, and Skiena 2014). They then compress this high-dimensional embedding into a multi-channel image by looking at cross sections of consecutive principle components from principle component analysis (PCA). Finally, they use a standard image classifier architecture to classify the graphs. This approach achieved good results, but has two shortcomings. First, the lack of end-to-end architecture results in embeddings that may work well for spatial preservation, but poorly for graph classification. Second, their pairing of PCA dimensions is somewhat arbitrary.

Here, we effectively combine and apply a powerful CNN architecture like that used by Tixier et al to the expressive node embeddings from GNNs, as in Kipf, Zhang, and Xinyi. The previously missing piece that can attach these two methods is a differentiable operation that converts node embeddings into a format that CNNs can leverage. To define such an operation, we draw from classical network science. Networks are typically analyzed by studying the distribution of their local features (Wasserman and Faust 1994). The most common of such analyses is performed on the degree distribution, which has been used to classify networks of different types, such as scale-free or small-world networks. This individual analysis of feature histograms inspired the binning mechanism introduced here. Our binning operation approximates the full node embedding distribution into a multi-channel histogram, which is easily inputted to a standard CNN.

Graph-Hist

A graph can be represented by $\mathcal{G} = (\mathcal{V}, X, A)$, where $\mathcal{V} = \{v_1, v_2, ..., v_n\}$, is the set of nodes, $A \in \{0,1\}^{n \times n}$, is the adjacency matrix, and $X \in \mathbb{R}^{n \times d}$, is the feature matrix. If there is a link from v_i to v_j, then $A_{i,j} = 1$; otherwise, $A_{i,j} = 0$. We will assume graphs contain self-loops, that is $A_{i,i} = 1$. The number of nodes is denoted by n, and the number of node features is denoted by d.

Typically, the graph Laplacian is operated on rather than the adjacency matrix itself. First, the degree matrix is calculated. The degree matrix, D, contains the node degrees along its diagonal, $D_{i,i} = \sum_j A_{i,j}$, and is zero elsewhere. Then, the symmetric normalized Laplacian is calculated using Equation 1.

$$L = D^{-\frac{1}{2}}(D - A)D^{-\frac{1}{2}} \tag{1}$$

This definition of the Laplacian is widely used in spectral graph theory. Thus, GCNs that work with this form of the Laplacian are referred to as *spectral* GCNs. Spectral GCNs were popularized by Kipf and Welling when they provided a local approximation to ChebNet, greatly reducing the computational cost (Kipf and Welling 2016). Spectral GCNs serve as a local approximation to the more general convolutional framework that was originally proposed by Bruna et al., and have strong underpinnings in graph signal processing (Bruna et al. 2013; Shuman et al. 2013). Another possibility would have been the random walk Laplacian, $L^{rw} = I - D^{-1}A$, which encodes the probability of transitioning from node to node. The random walk Laplacian has been used in a variety of works, both for graph classification and node embedding (Scarselli et al. 2009; Zhang et al. 2018; Hamilton, Ying, and Leskovec 2017). However, given the underpinnings in graph signal processing and the success of recent spectral models, we move forward with this definition.

In graph classification we are given a set of N labeled graphs, $\{(\mathcal{G}_1, y_1), (\mathcal{G}_2, y_2), ..., (\mathcal{G}_N, y_N)\}$. Using a one-hot encoding for the m classes, gives $y \in \{0, 1\}^{N \times m}$, where $y_{i,j} = 1$ if $\mathcal{G}_i$ belongs to class j, and $y_{i,j} = 0$ otherwise. We learn a classifier that outputs probabilities p, of each class by minimizing the cross entropy loss though stochastic gradient descent. The loss function is given by $\ell = -\sum_{i=1}^{N} \sum_{j=1}^{m} y_{i,j} \log(p_{i,j})$

A diagram of our proposed architecture is illustrated in Figure 1.

Graph Convolution

Kipf and Welling's propagation rule for GCNs is shown in Equation 2, where σ is a non-linear activation function, $W^{(l)} \in \mathbb{R}^{d \times c}$, are trainable weights for layer l, and where c is the number of channels selected for node embedding. Initially, $Z^{(0)} = X$. From there, the GCNs are layered such that the features from the first GCN are the new input features to the second GCN, which then outputs for the third, and so forth. This scheme has been widely adopted, and is seen in architectures like SortPool and Capsule Graph. This form of layering allows the features to be passed beyond direct neighbors. An h-level layering allows nodes to aggregate features within its h-hop neighborhood. Additionally, Zhang et al. show that layered GCNs provide a continuous analog to the graph coloring problem and the Weisfeiler-Lehman subtree kernel (Zhang et al. 2018; Shervashidze et al. 2011).

$$Z^{(l+1)} = \sigma(LZ^{(l)}W^{(l)}) \tag{2}$$

Despite this, GCN layering necessitates that feature aggregation from a node's extended neighborhood is reliant on aggregation closer to the source node. Further, a layered architecture is not parallelizable, and is harder to train. Thus, we propose a slight variant of the GCN, given in Equation 3.

$$Z^{(l)} = \sigma(L^l XW^{(l)} + b^{(l)}) \tag{3}$$

Instead of layering the graph convolutions, we compute them with powers of the Laplacian, thus embedding the original features for each hop, rather than the learned features. Experiments with our approach show slight improvements in performance while allowing for parallelization if memory allows. Additionally, independence of GCNs allows for the use of a bias term, which is not natural in layered-GCN architectures, since they would be multiplied back into the Laplacian at the next level. Lastly, we include $Z^{(0)}$, which reduces the Laplacian to the identity matrix, thus providing a standard fully connected sub-module from the node features.

Fully Connected Combination Layers

As in SortPool, Graph Capsules, and others, the node embeddings obtained from $Z^{(0)}, ..., Z^{(h)}$ are concatenated to give node embeddings of dimension $(h + 1) * c$. In the succeeding step, the inter-dimension properties are lost. More details on why are given in the following section.

To minimize the information loss, the embeddings are passed through two fully connected layers, which can capture these nonlinear inter-dimension properties. For simplicity, we've chosen layers that have the same dimension as the initial embeddings, $(h + 1) * c$. We denote these final embeddings as $C \in \mathbb{R}^{n \times (h+1)*c}$. This step is similar to the initial convolution applied to the final embeddings in SortPool. The two differences are the size of the output, and that we are applying the transformation twice.

Node Binning

One of the fundamental challenges in graph classification is that each input graph can have a different number of nodes, n, in its nodeset. Thus, graph classification algorithms which rely on node embeddings must find some method of transitioning from a variable size n to a fixed size k.

SortPool, for example, re-shapes the data by setting a threshold, k. After sorting, the top k nodes are selected, and the rest are dropped. If the graph has less than k nodes, zeros are appended to the nodeset until it is size k. As previously discussed, this harms the integrity of the data.

We solve this problem through a binning procedure. The input space is discretized, and the number of nodes falling into each discrete bin is counted. Then, a standard convolutional architecture can be applied to the obtained density function. However, effective node embeddings are typically high in dimension, making discretization over the full space intractable. The 2D-CNN approach taken by Tixier et al. approximates the distribution using principle components (Tixier et al. 2017). They take 2-D cross sections of the ascending principle components, stacking them as channels of an image. A standard image classifier can then be applied. While this achieved good results, it relies on preprocessing and cannot update node embeddings to improve performance.

Here we provide a differentiable alternative that does not rely on a dimension-pairing scheme. Instead, we bin the data along one-dimensional cross sections of each dimension, resulting in a histogram with k bins and $(h + 1) * c$ channels. The number of bins is a tunable parameter.

The activation function used to obtain C must be bounded. Without a bounded activation function, bin boundaries could not be predefined to capture all of the nodes placements. For all experiments, we have selected $tanh$. With $tanh$, all output values will be between -1, and 1, so if $k = 10$, evenly spaced non-overlapping bins will be of length 0.2, and will capture all potential node placements.

The derivative of the loss function, ℓ, can then be propagated through the binning layer using a weighted average of the bin gradients, as shown in Equation 4. First, the distances, r, to the bin centers, B, are calculated, making $r \in \mathbb{R}^{k \times n \times (h+1)*c}$. Then, weighted averages of the bin gradients are taken, allowing bins closer to nodes to have more pull than bins further away. Thus, each bin pulls nodes towards it if its gradient is positive, and pushes nodes away if its gradient is negative. The amount of pull is proportionate to its distance from the nodes and is controlled by γ, which we have set to $\gamma = 20$, giving low weight to bins far away.

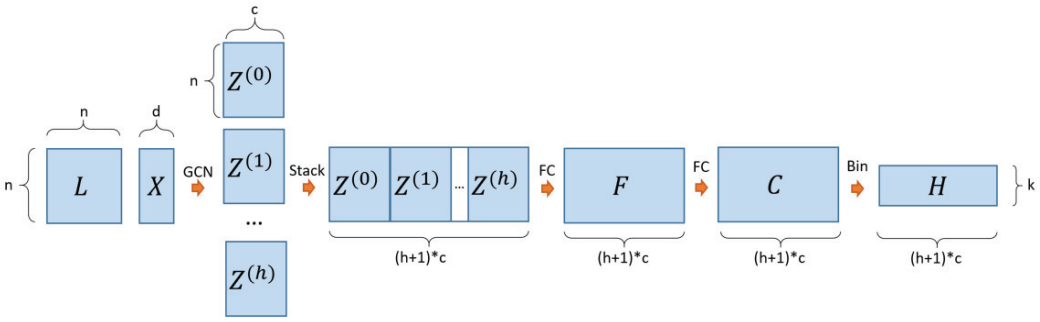

Figure 1: The general architecture used in this work. The output, H, is the multi-channel histogram that is then classified using a 1-dimensional variant of LeNet-5, as described in the Histogram Classification section.

$$\frac{\partial \ell}{\partial C_{i,j}} = \frac{1}{\sum_{b=1}^{k} e^{-\gamma |r_{b,i,j}|}} \sum_{b=1}^{k} e^{-\gamma |r_{b,i,j}|} \text{sign}(r_{b,i,j}) \frac{\partial \ell}{\partial H_{b,j}} \tag{4}$$

$$r = B - C \tag{5}$$

Again, this process does lose the co-variance relationship between the dimensions of the distribution. However, the combination layers overcome this simplification. Nodes will be pushed along these 1-D cross sections during back-propagation such that a classification can be made. The effectiveness of this approach is demonstrated on standard benchmark data in the Experiments section and on bot detection data in the Case Study section.

Histogram Classification

Finally, the multi-channel histogram can be classified using a traditional convolutional architecture. Tixier et al. used a variant of LeNet-5 to classify graphs based on 2-D cross sections of predefined node embeddings (Tixier et al. 2017). The CNN achieved 99.45% accuracy on the MNIST handwritten digit classification task. We slightly modify the architecture to suit the 1-dimensional data that we have obtained from the previous steps.

As in (Tixier et al. 2017), H is passed to 4 sub-modules, with filter sizes of $f = 3, 4, 5,$ and 6, respectively. A sub-module is performed as follows. The input data is convolved over with filter size f and a stride of 1, to 64 output channels. Then, max pooling is performed with size and stride 2. The convolution is performed again, but with 96 output channels.

Simultaneously, H is passed to a convolution with $f = (h+1) * c$, thus capturing the entire histogram with 96 output channels. The full-histogram convolution and the sub-module outputs are concatenated and connected to a fully connected layer of size 256. Lastly, the 256 unit layer is connected to a softmax output that classifies the graph. Dropout layers were placed before all fully connected layers in the histogram classifier with probability ρ. The activation function used was ReLU. The three changes from the original classifier given are: the whole-histogram convolution is added, the 128 hidden unit layer was changed to 256 units, and the model was adapted to its 1-dimensional analog.

The entire model, then, can be trained in an end-to-end manner.

Experiments

Benchmark Datasets and Methods

There are many potential benchmark datasets for graph classification, however few of them are social networks, and even fewer resemble the type of networks seen in real world social media data. Real world social media networks are typically large and sparse, while most benchmark datasets are relatively dense and have nodesets with less than 100 nodes (Wasserman and Faust 1994; Kersting et al. 2016).

With this in mind, we have selected 6 popular benchmark datasets, displayed in Table 1. The datasets have been obtained from Kersting et al's collection, but were created by Yanardag and Vishwanathan (Kersting et al. 2016; Yanardag and Vishwanathan 2015).

The IMDB datasets are movie collaboration datasets. Nodes represent actors or actresses, and links represent co-appearance in a movie. The graphs are ego networks, and the task is to classify the genre that an ego network belongs to. This dataset is somewhat challenging because movies may belong to more than one genre, but may only be given one label.

COLLAB was derived from scientific collaboration data in three fields: High Energy Physics, Condensed Matter Physics, and Astro Physics. Each graph is an author's ego network, and the task is to identify which field they work in.

All three of the Reddit datasets were scraped from the social media platform Reddit, using their API. Nodes in the graph are Reddit users, and links are created by direct replies in the discussion. In the binary dataset, the graphs either come from question-and-answer subreddits, or discussion-based subreddits. The task is to identify which type of subreddit the conversational graph comes from. In the 5k and 12k, datasets, the task is to identify the specific subreddit that the graph belongs to. We place greater emphasis on these datasets, as they are the only social media classification tasks. Table 1 illustrates the importance of this distinction. The graphs in the Reddit datasets tend to have ten times more nodes, and tend to be 100 times less dense.

Table 1: A summary of the datasets studied. Numbers for nodes and edges are averages.

Dataset	IMDB-B	IMDB-M	COLLAB	REDDIT-B	REDDIT-5K	REDDIT-12K	BOTS
Graphs	1000	1500	5000	2000	4999	11929	14962
Classes	2	3	3	2	5	11	2
Nodes	19.77	13.00	74.49	429.63	508.52	391.41	7294
Edges	96.53	65.94	2457.78	497.75	594.87	456.89	11034

We have selected 5 different methods to compare our results against, namely Anonymous Walk Embeddings, SortPool, DiffPool, CapsGNN, and 2D CNN, (Ivanov and Burnaev 2018; Zhang et al. 2018; Ying et al. 2018; Tixier et al. 2017). These methods were selected to reflect state-of-the-art classification results, and to compare our results against methods from which we have built upon. To the best of our knowledge, the current state-of-the-art performances are shown for every dataset. The accuracies and standard deviations are reported in Table 3 based on the values reported in initial publication. Because of this, not every dataset has a value for every method. Fey and Lenssen have introduced PyTorch Geometric, a library with implementations of many geometric learning algorithms (Fey and Lenssen 2019). Some gaps are filled by using values reported from their implementations. Anonymous Walk Embeddings is the only kernel approach shown, so it is separated in Table 3.

Experimental Setting

The general architecture used for all experiments is illustrated in Figure 1. Graph-Hist was implemented in PyTorch. The hyperbolic tangent function was used for all activation functions leading up to LeNet. We used the ReduceLROnPlateau scheduler with an initial learning rate of $\alpha = 1e-4$, a factor of 0.5, a patience of 2, a cooldown of 0, and a minimum learning rate of $1e-7$. We used stochastic gradient descent with a mini-batch size of 32. We terminated training after 9 consecutive epochs without progress in the testing loss.

We then tuned parameters to each dataset in the search space $h \in [2, 4, 6, 8]$, $c \in [32, 64, 128, 256]$, $\rho \in [0.2, 0.8]$. Parameters were selected based on their performance on the test set. The final parameters are shown in Table 2.

Finally, we performed 10-fold cross-validation on each of the datasets using the parameters in Table 2. The mean accuracy and its standard deviation is reported for each dataset in Table 3. Graph-Hist advances state-of-the-art classification in all 3 of the social media benchmarks. It also beats state-of-the-art results for IMDB-B, and obtains second place results for the remaining two datasets.

We recognize that there are many more hyperparameters that could be tuned, like the batch size, and that even the size of the fully connected layers could be tuned. Exploring these possibilities is left for future work, but could result in even better results than those demonstrated here.

Table 2: The parameters used on each dataset.

Dataset	k	h	c	ρ
IMDB- B	50	2	128	0.8
IMDB-M	25	4	128	0.8
COLLAB	25	2	256	0.2
REDDIT-B	25	6	64	0.8
REDDIT-5K	25	8	64	0.8
REDDIT-12K	25	2	64	0.8
TWITTER BOTS	25	2	8	0.5

Case Study

Automated accounts called bots are increasingly used in online information operations to manipulate networks and the narratives that transit these networks. In doing so, state and non-state actors can artificially manipulate the online marketplace of belief and ideas. The growing field of social cybersecurity seeks to protect this marketplace from information campaigns and disinformation (Carley et al. 2018). Bot detection plays a critical role in characterizing information campaigns, though bend maneuvers, and in understanding the spread of disinformation (Beskow and Carley 2019; Babcock, Beskow, and Carley 2018). Thus, researchers in industry, government, and academia have developed increasingly sophisticated algorithms to detect these nefarious accounts. These research efforts have led to a "cat and mouse" cycle in which increasingly sophisticated algorithms are required to detect increasingly sophisticated bots. Early detection models identified tell-tale indicators of automated activity such as stolen identities, lack of circadian rhythms, anonymous attributes (lack of profile picture, random string screen name, etc), and a low follower/followee ratio. These features, however, are relatively easy for a bot "puppetmaster" to manipulate in order to remain undetected.

It is much harder for these same bot "puppet-masters" to change the artificial features of the social and communication networks that they inhabit. These social and communication networks (following, retweeting, mentioning, replying) lack the overlapping social integration of human social and communication links. Thus, we exploit the structure of these communication networks directly using Graph-Hist. We find that this approach generalizes to new datasets better than current alternative approaches.

Building Networks

We built the conversational network that a Twitter account inhabits in the same manner as Beskow and Carley, resulting in ego networks with 21 node features (Beskow and Carley 2018a). This approach combines the timelines of the target account and their followers to build the larger conversation. This method was selected because it creates a comprehensive ego network while overcoming API rate limiting con-

Table 3: Benchmark dataset accuracies from 10-fold cross-validation. Top-2 scores are emboldened, the state-of-the-art score is marked with an asterisk. Scores are shown as reported in publication, so not all datasets are represented or are shown with their standard deviation for every method.

Dataset	IMDB-B	IMDB-M	COLLAB	REDDIT-B	REDDIT-5K	REDDIT-12K
AWE	$\mathbf{74.5 \pm 5.8}$	$\mathbf{51.5 \pm 3.6}^*$	73.9 ± 1.9	87.9 ± 2.5	50.5 ± 1.9	39.2 ± 2.1
SortPool	72.4 ± 3.8	47.8 ± 0.8	77.7 ± 3.1	74.9 ± 6.7	-	-
DiffPool	72.6 ± 3.9	-	78.9 ± 2.3	$\mathbf{92.1 \pm 2.6}$	-	47.1
CapsGNN	73.1 ± 4.8	$\mathbf{50.3 \pm 2.6}$	$\mathbf{79.6 \pm 0.9}^*$	-	$\mathbf{52.9 \pm 1.5}$	46.6 ± 1.9
2D CNN	70.4 ± 3.8	-	71.3 ± 2.0	89.1 ± 1.7	52.1 ± 2.2	$\mathbf{48.1 \pm 1.5}$
Graph-Hist	$\mathbf{74.7 \pm 3.9}^*$	$\mathbf{50.3 \pm 3.6}$	$\mathbf{79.2 \pm 2.0}$	$\mathbf{92.2 \pm 2.2}^*$	$\mathbf{55.0 \pm 1.7}^*$	$\mathbf{49.2 \pm 1.0}^*$

Table 4: Bot detection F1, Precision, and Recall scores. All models but Botometer trained on Debot data. Top-2 F1 scores are emboldened, the state-of-the-art score is marked with an asterisk.

Model	F1	Precision	Recall
Botometer	0.524	0.858	0.377
Debot	0.012	1.00	0.006
Bot-Hunter Tier1	0.656	0.821	0.546
Bot-Hunter Tier2	**0.687**	0.691	0.683
Bot-Hunter Tier3	0.599	0.837	0.466
Graph-Hist	$\mathbf{0.740}^*$	0.683	0.807

straints and expediting the time it would take to collect the data (target collection is 5 minutes per account). While 5 minutes per account seems long, this is trivial compared to the hours or days that it would take to build a single ego network based on friends/followers connections. The properties of these networks are summarized in Table 1.

Again, the differences between social media networks and standard benchmarks are pronounced. The Twitter networks are 2 orders of magnitude larger than the non-social media benchmarks in terms of nodeset size. The Twitter network densities are also 3 orders of magnitude smaller than those of the standard benchmarks.

Previous Work in Bot Detection

For the past decade, increasing numbers of researchers have worked on developing algorithms to detect increasingly sophisticated bots. These models can be broadly separated into supervised machine learning models, unsupervised models, and graph-based models. Several of the models have become prominent tools that are used in social cybersecurity workflows, including the Botometer, Bot-Hunter, Debot, and Botwalk algorithms (Davis et al. 2016; Beskow and Carley 2018b; Chavoshi, Hamooni, and Mueen 2016; Minnich et al. 2017).

Most of the graph and community detection methods have been conducted on Facebook, where these bots are at times called Sybils. These include random walk approaches like Sybil-Guard (Yu et al. 2006), Sybil-Resist (Ma et al. 2014) and Sybil-Rank (Cao et al. 2012). Other models relax some of the assumptions and use trust propagation approaches such as the Sybil-Fence method (Cao and Yang 2013).

Supervised models include traditional machine learning with SVM (Lee and Kim 2014), Naïve Bayes (Chen, Guan, and Su 2014), and Random Forest (Ferrara et al. 2016) models trained on features extracted from Twitter's tweet and user objects. Other methods have attempted to classify accounts based only on their text (Kudugunta and Ferrara 2018) or their screen name (Beskow and Carley 2018c). Several of the available models like Botometer (Davis et al. 2016) and Bot-Hunter (Beskow and Carley 2018b) are classic supervised machine learning models.

Several unsupervised methods have also emerged, largely focused on identifying underlying patterns produced by certain types of bots. These include clustering algorithms (Benigni, Joseph, and Carley 2017) and anomaly detection algorithms like the BotWalk algorithm (Minnich et al. 2017).

Most of these models leverage account data and account history while graph-based models are focused on finding patterns in the conversation and connections. Not many models focus on the larger conversational ego-network surrounding the account. Only one supervised machine learning model has attempted to bring network science metrics (centrality, simmelian ties, triadic census, etc) from these ego networks into their feature space (Beskow and Carley 2018a). Rather than using network metrics as proxies for the network itself, we approached this same problem with geometric learning over the entire graph.

Bot Classification Results

For the case study, we built training data of bot accounts that have been labeled by the Debot unsupervised algorithm. The Debot algorithm uses warped correlation to identify bots that post the same content at roughly the same time (Chavoshi, Hamooni, and Mueen 2016). Debot has demonstrated high precision identifying this special class of bots, and has been used to train classic supervised bot detection models with strong results; thus, it was used to label bot data for training. Non-bot "human" data was randomly sampled from the Twitter 1% Stream. Our training data consisted of 8,842 bots and 6,120 human accounts and their associated conversational networks.

We developed a separate test dataset to compare against other state of the art algorithms as well as measure generalizability. The final test data was created by manually annotating 337 bot accounts focused on propaganda and other manipulation. Emphasis was made to ensure this test data did not overlap with any training data. The test dataset was balanced with 337 bot accounts and 337 human accounts.

For an evaluation metric we used the F1-score, defined as the harmonic mean of precision and recall. Many early bot-detection models had relatively high precision but low recall, inflating accuracy metrics. With low recall, these models underestimate the scale of the bot infestation and disinformation problem in general. We found the F1 score as an adequate balance emphasizing both precision and recall. F1 score for all models is provided in Table 4.

We see that the Botometer model has the highest precision of all comparison models, but lower recall and therefore lower relative F1 score. The Debot algorithm was able to identify two of the bots, has perfect precision, but very low recall and F1 score. The Bot-Hunter algorithms improve recall at a slight cost in precision, resulting in slightly higher F1 scores compared to other models.

Graph-Hist was hand-tuned after the grid search used on the benchmark datasets, resulting in the final configuration given in Table 2. The training environment was the same as that used for the benchmark dataset experiments, except random over-sampling of the data was performed for balanced training. The new stopping threshold was given by F1 score in the validation set. Graph-Hist has recall higher than all other models and precision slightly below Bot-Hunter, resulting in the highest F1 score of all models tested.

Conclusions

In this paper, we have proposed a neural network architecture, Graph-Hist, for graph classification. Graph-Hist creates expressive node embeddings from GCNs in a similar manner to previous successful models, and uses a powerful CNN architecture to classify these embeddings in an end-to-end manner. While each aspect of the model has not appeared in its exact form in prior literature, the most significant innovation is the binning module. It allows node embedding distributions to be approximated in a differential manner, such that convolutional architectures are then applicable. The binning procedure was inspired by the analysis of large social networks, and as such has been applied to social network classification tasks. Graph-Hist advances the state-of-the-art performance on 4 out of the 6 tested benchmark datasets, including all 3 of the social media benchmarks.

Lastly, Graph-Hist was applied to a new graph classification domain: bot detection. Graph-Hist demonstrated better generalization in this task than the current leading bot detection models. Graph classification methods have another huge advantage to classic approaches when it comes to bot detection: they are hard to guard against. These models are highly non-linear, so it is not obvious what types of graphs a "puppet-master" should try to construct to avoid detection. Even if an inconspicuous structure was known, the communication graph is far more challenging to manipulate than simple features like tweet frequency. While communication networks are more costly to collect, the popularization of graph classification approaches to bot detection should slow down the ongoing "cat and mouse" cycle.

Future extensions of this work may involve attaching binning modules to different embeddings schemes, classifying the resulting histograms with new methods, or experimenting with non-uniform binning modules. It could also include improvements in the bot domain, specifically by classifying other types of entities, such as trolls, which may have communication graphs differing from both normal actors and bots. Currently, deployment of graph classification algorithms for bot detection is inhibited by social media API bottlenecks, so work on more scalable social media graph collection schemes could have a large impact. Finally, future work may advocate for increased attention to social media datasets though the release of new social media benchmark datasets which reflect the scale and sparsity of networks seen in the wild.

Acknowledgments

This work was supported in part by the Office of Naval Research (ONR) Award N00014182106 Group Polarization in Social Media and Bot-Hunter Award N000141812108, and the Center for Computational Analysis of Social and Organization Systems (CASOS). Thomas Magelinski was also supported by an ARCS Foundation scholarship. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the ONR or the U.S. Government.

References

Babcock, M.; Beskow, D. M.; and Carley, K. M. 2018. Beaten up on twitter? exploring fake news and satirical responses during the black panther movie event. In Thomson, R.; Dancy, C.; Hyder, A.; and Bisgin, H., eds., *Social, Cultural, and Behavioral Modeling*, 97–103. Cham: Springer International Publishing.

Benigni, M. C.; Joseph, K.; and Carley, K. M. 2017. Online extremism and the communities that sustain it: Detecting the isis supporting community on twitter. *PloS one* 12(12):e0181405.

Beskow, D., and Carley, K. M. 2018a. Bot conversations are different: Leveraging network metrics for bot detection in twitter. In *Advances in Social Networks Analysis and Mining (ASONAM)*, 176–183. IEEE.

Beskow, D., and Carley, K. M. 2018b. Introducing bothunter: A tiered approach to detection and characterizing automated activity on twitter. In *International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation*. Springer.

Beskow, D., and Carley, K. M. 2018c. Using random string classification to filter and annotate automated accounts. In *International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation*. Springer.

Beskow, D., and Carley, K. 2019. Social cybersecurity: An emerging national security requirement. *Military review* 99:117.

Bruna, J.; Zaremba, W.; Szlam, A.; and LeCun, Y. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*.

Cao, Q., and Yang, X. 2013. Sybilfence: Improving social-graph-based sybil defenses with user negative feedback. *arXiv preprint arXiv:1304.3819*.

Cao, Q.; Sirivianos, M.; Yang, X.; and Pregueiro, T. 2012. Aiding the detection of fake accounts in large scale social online services. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, 15–15. USENIX Association.

Carley, K. M.; Cervone, G.; Agarwal, N.; and Liu, H. 2018. Social cyber-security. In *International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation*. Springer.

Chavoshi, N.; Hamooni, H.; and Mueen, A. 2016. Debot: Twitter bot detection via warped correlation. In *ICDM*, 817–822.

Chen, C.-M.; Guan, D.; and Su, Q.-K. 2014. Feature set identification for detecting suspicious urls using bayesian classification in social networks. *Information Sciences* 289:133–147.

Davis, C. A.; Varol, O.; Ferrara, E.; Flammini, A.; and Menczer, F. 2016. Botornot: A system to evaluate social bots. In *Proceedings of the 25th International Conference Companion on World Wide Web*, 273–274. International World Wide Web Conferences Steering Committee.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, 3844–3852.

Ferrara, E.; Varol, O.; Davis, C.; Menczer, F.; and Flammini, A. 2016. The rise of social bots. *Communications of the ACM* 59(7):96–104.

Fey, M., and Lenssen, J. E. 2019. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*.

Gori, M.; Monfardini, G.; and Scarselli, F. 2005. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, 729–734. IEEE.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, 1024–1034.

Ivanov, S., and Burnaev, E. 2018. Anonymous walk embeddings. *arXiv preprint arXiv:1805.11921*.

Kersting, K.; Kriege, N. M.; Morris, C.; Mutzel, P.; and Neumann, M. 2016. Benchmark data sets for graph kernels.

Kipf, T. N., and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Kudugunta, S., and Ferrara, E. 2018. Deep neural networks for bot detection. *Information Sciences* 467:312–322.

Lee, S., and Kim, J. 2014. Early filtering of ephemeral malicious accounts on twitter. *Computer Communications* 54:48–57.

Ma, W.; Hu, S.-Z.; Dai, Q.; Wang, T.-T.; and Huang, Y.-F. 2014. Sybil-resist: A new protocol for sybil attack defense in social network. In *International Conference on Applications and Techniques in Information Security*, 219–230. Springer.

Minnich, A.; Chavoshi, N.; Koutra, D.; and Mueen, A. 2017. Botwalk: Efficient adaptive exploration of twitter bot networks. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, 467–474. ACM.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 701–710. ACM.

Scarselli, F.; Gori, M.; Ah Chung Tsoi; Hagenbuchner, M.; and Monfardini, G. 2009. The Graph Neural Network Model. *IEEE Transactions on Neural Networks* 20(1):61–80.

Shervashidze, N.; Schweitzer, P.; Leeuwen, E. J. v.; Mehlhorn, K.; and Borgwardt, K. M. 2011. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research* 12:2539–2561.

Shuman, D. I.; Narang, S. K.; Frossard, P.; Ortega, A.; and Vandergheynst, P. 2013. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine* 30(3):83–98.

Tixier, A. J.-P.; Nikolentzos, G.; Meladianos, P.; and Vazirgiannis, M. 2017. Graph Classification with 2d Convolutional Neural Networks. *arXiv:1708.02218 [cs]*. arXiv: 1708.02218.

Wasserman, S., and Faust, K. 1994. *Social network analysis: Methods and applications*, volume 8. Cambridge university press.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Yu, P. S. 2019. A Comprehensive Survey on Graph Neural Networks. *arXiv:1901.00596 [cs, stat]*. arXiv: 1901.00596.

Xinyi, Z., and Chen, L. 2019. Capsule graph neural network. In *International Conference on Learning Representations*.

Yanardag, P., and Vishwanathan, S. 2015. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1365–1374. ACM.

Ying, Z.; You, J.; Morris, C.; Ren, X.; Hamilton, W.; and Leskovec, J. 2018. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*, 4800–4810.

Yu, H.; Kaminsky, M.; Gibbons, P. B.; and Flaxman, A. 2006. Sybilguard: defending against sybil attacks via social networks. *ACM SIGCOMM Computer Communication Review* 36(4):267–278.

Zhang, M.; Cui, Z.; Neumann, M.; and Chen, Y. 2018. An End-to-End Deep Learning Architecture for Graph Classification. In *AAAI*.