



HAL
open science

Label-GCN: An Effective Method for Adding Label Propagation to Graph Convolutional Networks

Claudio Bellei, Hussain Alattas, Nesrine Kaaniche

► **To cite this version:**

Claudio Bellei, Hussain Alattas, Nesrine Kaaniche. Label-GCN: An Effective Method for Adding Label Propagation to Graph Convolutional Networks. 2024. hal-03991083

HAL Id: hal-03991083

<https://hal.science/hal-03991083>

Preprint submitted on 18 Jun 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Label-GCN: An Effective Method for Adding Label Propagation to Graph Convolutional Networks

Claudio Bellei

Elliptic
London, UK
claudio@elliptic.co

Hussain Alattas

Department of Computer Science
The University of Sheffield, UK

Nesrine Kaaniche

Telecom SudParis
Polytechnic Institute of Paris

Abstract

We show that a modification of the first layer of a Graph Convolutional Network (GCN) can be used to effectively propagate label information across neighbor nodes, for binary and multi-class classification problems. This is done by selectively eliminating self-loops for the label features during the training phase of a GCN. The GCN architecture is otherwise unchanged, without any extra hyper-parameters, and can be used in both a transductive and inductive setting. We show through several experiments that, depending on how many labels are available during the inference phase, this strategy can lead to a substantial improvement in the model performance compared to a standard GCN approach, including with imbalanced datasets.

1 Introduction

In recent years, the emergence and development of graph neural networks (GNNs) has allowed the extension of deep learning methods to data that have an underlying non-Euclidian structure. Many variants of GNNs have been proposed over the years, with applications ranging from recommender systems, to chemistry, traffic control, physics, and more [32]. One of such variants, Graph Convolutional Networks (GCN) [13], can be considered a state-of-the-art model for learning graph representations [22]. GCNs are multi-layer feedforward neural networks where, at each layer, a first-order approximation of a spectral graph convolution is applied. GCNs can be applied to semi-supervised node classification problems, where the graph is made of both labeled and unlabeled data and the goal is to predict the label of a node, given the graph structure and a set of features associated with each node.

In GCN, labels are not used in the learning phase and therefore this information does not affect the prediction of an unknown node. However, in many applications, nearby nodes or nodes that belong to the same graph structure might share the same label. For such applications, one would expect that the ability to use label information should lead to an improvement of the model.

In this paper, we show that a simple modification of the first layer of a GCN can make the model learn information from neighbor labels. This leads to a measurable improvement in performance in various experiments of citation and Bitcoin networks, depending on the availability of neighbor labels during the inference phase. The model maintains the properties of the original GCN approach, i.e., it does not add extra hyper-parameters and can be used in both a transductive setting - where the whole graph structure is known during the training phase - and an inductive setting - where the model is tested on previously unseen nodes. Supporting code is available at <https://github.com/cbellei/LabelGCN>.

2 Related work

The notion that nearby nodes of a graph are likely to have the same label has been modeled by researchers over the years through various variants of an algorithm that goes under the name of Label Propagation (LP) [33, 11, 31, 26, 24, 29, 12, 7, 23]. LP is a semi-supervised learning algorithm that propagates known labels along the edges of a graph, for the purpose of classifying unlabelled nodes. Examples of real-world datasets where information from neighbor labels has been shown to be beneficial include social networks, web pages, protein-protein interactions, citation graphs and anti-money laundering in the Bitcoin network [5, 23, 18, 10].

The idea of combining graph convolutions with label propagation has only been explored recently [25, 4, 17]. In [25], a model is proposed that unifies LP and GCN. There is a duality between the two models, in that LP can be interpreted as propagating *labels* along edges in the graph and averaging them linearly, while GCN as propagating *features* along the edges and combining them non-linearly through the various layers that compose the neural network. The original model proposed in [25] works in a transductive setting, where all node features and graph structure is available during the training phase; the GCN model has to learn both the weight matrix and a weighted adjacency matrix, which can lead to overfitting. In [4], a two-step process is proposed, that includes a first step of label propagation that feeds into a second step of a neural network classifier. Similarly in [17], a general framework for composing networks is discussed, which includes the possibility of composing GCN with a label propagation network.

The present paper proposes an alternative method to adding label propagation to GCN, through a simple variant of the original GCN approach. This has three main advantages: 1. it allows the model to work in an inductive setting, where the testing nodes are not part of the graph structure during the training phase; 2. it does not result in the addition of new hyper-parameters, which often increases the risk of overfitting the training data; 3. it does not add any significant algorithmic complexity to the original GCN approach.

3 Label-GCN

3.1 Graph Convolutional Networks

Graph Convolutional Networks [13] are a class of multi-layer neural network architectures designed to take advantage of the graph structure of a given dataset. The graph structure is specified by using an undirected adjacency matrix $A \in \mathbb{R}^{n \times n}$, where n is the number of nodes in the graph, along with the input matrix $X \in \mathbb{R}^{n \times d}$ containing the features for each node, where d is the dimension of the feature vector. The output of every new hidden layer in GCN combines a normalized adjacency matrix \hat{A} , the output of the previous hidden layer $H^{(l)} \in \mathbb{R}^{n \times d_l}$ and a weight matrix $W^{(l)} \in \mathbb{R}^{d_l \times d_{l+1}}$,

$$H^{(l+1)} = \sigma \left(\hat{A} H^{(l)} W^{(l)} \right) \quad (1)$$

$$\hat{A} = \tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2}, \quad \tilde{A} = A + I, \quad \tilde{D}_{ii} = \sum_j \tilde{A}_{ij} \quad (2)$$

The first hidden layer corresponds to the input matrix, $H^{(0)} := X$. The σ operator in eq. (1) represents a non-linear activation function (typically ReLU). The normalized adjacency matrix \hat{A} differs from the standard adjacency matrix A in that self-loops are added and the matrix elements are rescaled through the diagonal node degree matrix \tilde{D} .

Over time, many variants of the original GCN approach have been proposed. Notable ones have been directed at improving the applicability of graph-based models for large scale applications [8, 2, 28]. A recent survey on the state of the research can be found in [30].

3.2 Label-GCN

For many applications, it is reasonable to assume that knowledge of the labels surrounding an unknown center node can help predict the label of the center node itself. This suggests adding a new

feature during learning of the network: the label of each node. However in the canonical formulation of GCN, eqs. (1) and (2), the convolution operation would include a self-loop for each center node. Simply adding the label as a new "feature" during the training phase of the model would clearly not work: the model would simply learn to use the label for the prediction, which would not be available during the testing phase. What is instead desirable is to train the network through the neighbors' labels, without addition of the self-loops.

In order to achieve this, it is possible to modify the first layer of a GCN to include the labels of the neighbors, for each center node, but not the label of the center node itself. Labels can then become part of the feature vector, as long as the model is only allowed to see the labels of the neighbors. For a classification problem with K classes, assuming one-hot encoding the dimensionality of the input matrix therefore increases by a factor K , such that $X \in \mathbb{R}^{n \times (d+K)}$. The modified GCN network that achieves the required behavior is then

$$H^{(1)} = \sigma \left[\left(\widehat{A}X - \text{diag}(\widehat{A})X \sum_{j=1}^K e_j e_j^T \right) W^{(0)} \right] \quad (3)$$

$$H^{(l+1)} = \sigma \left(\widehat{A}H^{(l)}W^{(l)} \right) \quad l \geq 1 \quad (4)$$

In eq. (3), e_j is a unit vector that has all its components equal to zero, except for the index associated with the position of the one-hot encoded class within the feature vector, $e_j = [0, \dots, 0, 1, 0, \dots, 0]^T$. The product $e_j e_j^T$ is a single-entry matrix J^{jj} and the term

$$- \text{diag}(\widehat{A})X \sum_{j=1}^K e_j e_j^T$$

selectively eliminates the self-loops for the components of the feature vector corresponding to the labels. For all the other components, the standard convolution operation is applied.

Training the network using eq. (3) corresponds to masking the label of a center node. For a visual explanation, consider node A in Figure 1a. The standard convolution, eq. (4), aggregates all the values around the center node, including the value of the center node itself (Figure 1b). The modified convolution of eq. (3) computes the same aggregation, except for the components of the feature vector corresponding to the labels: in this case it only aggregates the neighbors, avoiding the self-loop (Figure 1c).

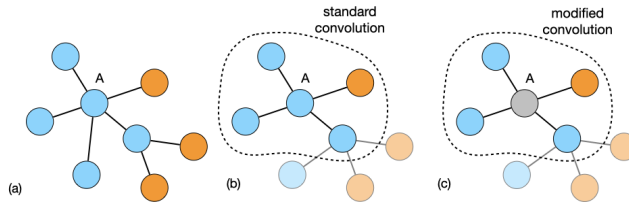


Figure 1: The difference between standard convolution, eq. (4), and the modified convolution, eq. (3), when self-loops are removed.

4 Experiments

4.1 Datasets

For our experiments, we test our model on four different datasets, treating the graphs as undirected. The datasets are the well-known citation graphs (CORa, CiteSeer [21] and PubMed [15]) and the Elliptic dataset, a Bitcoin transaction graph [27]. The statistics on the datasets are shown in Table 1. Given that the Bitcoin network graph is more recent and less studied, we give below more details about it and motivate why label propagation can work for such problems.

Dataset	Type	Nodes	Edges	Classes	Features	Training/Validation/Test/Support
Cora	Citation network	2,708	5,429	7	1,433	140/140/273/2,155
Citeseer	Citation network	3,312	4,715	6	3,703	120/120/332/2,740
PubMed	Citation network	19,717	44,338	3	500	60/60/1973/17,624
Elliptic	Bitcoin network	203,769	234,355	2	166	4,656/4,656/9,314/27,938

Table 1: Dataset statistics and splits for the experiments (transductive setting). The concept of "support" set is explained in Subsection 4.2.

4.1.1 The Elliptic Dataset

The Elliptic dataset is a publicly available dataset that is made of 203,769 nodes and 234,355 edges, divided into 49 distinct time steps [6]. Each node represents a transaction; an edge represents the flow of bitcoins between two transactions, as in Figure 2b. There are 166 features associated with each node. The classes in the dataset are grouped by entities belonging to the "licit", "illicit" or "unknown" categories; the proportion of nodes in the dataset are 21%, 2% and 77%, respectively. Therefore, this is an imbalanced dataset and only a fraction of it (23% or 46,564 nodes) is labelled.

4.1.2 Label propagation in the context of Bitcoin

The use of labels for the purpose of detecting illicit transactions in the Bitcoin network has been investigated in [10, 16]. In particular, in [16] it is shown that this information can lead to a significant improvement of the results on the Elliptic dataset. Here, we want to motivate the reason why this is the case.

One of the main features of Bitcoin (as well as of other blockchains such as Zcash, Litecoin, Bitcoin Cash, Bitcoin SV, ...) is that it is unspent-transaction-output (UTXO) based: each input of a given transaction unlocks the funds associated with an output from a previous transaction. The transaction generates new (unspent) outputs that can be spent by the private-key holders of those outputs in future transactions. Typically, among those private-key holders, one of them corresponds to the same entity that has generated the transaction itself. The reason is that 1. normally the inputs of a transaction are controlled by the same entity and 2. what is left of a transaction goes back as one of the outputs to the same entity generating it (the "change" address of the transaction). These two observations have implications with respect to the privacy aspects of the Bitcoin blockchain [19, 1, 20, 14, 9].

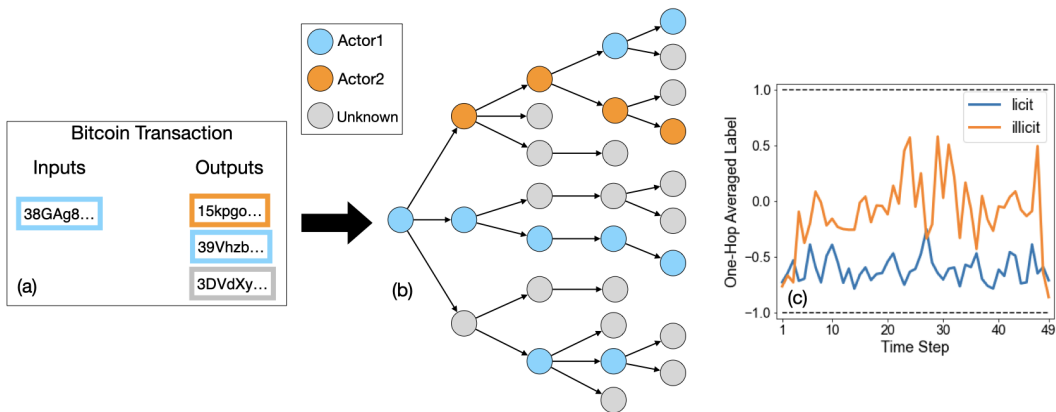


Figure 2: A Bitcoin transaction is made of inputs and outputs. Typically, among the outputs there is one that is controlled by the same entity that controls the inputs (a). This has the effect of producing chains of transactions, each of which has been created and then broadcasted to the Bitcoin network by the same entity (b). Licit and illicit nodes in the Elliptic dataset show a different behavior in the one-hop averaged labels.

By following the change, long chains of transactions can be built - where each transaction has been broadcasted to the Bitcoin network by a single entity. Building a chain of transactions, starting from a given labelled transaction, can be seen as propagating the label along a specific direction as shown in Figure 2a,b. In the Elliptic dataset, the labels of the various known entities are grouped together

Table 2: Model architectures for the experiments (transductive setting).

Layer	# Neurons	Activation Function	Parameters
Graph Convolution	16 (100 Elliptic)	RELU	-
Dropout	-	-	rate=0.5
Graph Convolution	16 (100 Elliptic)	RELU	-
Dropout	-	-	rate=0.5
Dense (output)	# classes	Softmax	-
learning rate	-	-	0.01
early stopping - patience	-	-	10 (30 Elliptic)

under the labels of "licit" and "illicit" instead of being specific to an actor. However, the logic around the propagation of labels through the change address still applies. Figure 2c, obtained after mapping "licit" \rightarrow -1, "unknown" \rightarrow 0, "illicit" \rightarrow 1 shows that the average label at a distance of one-hop from a center node can indeed be a useful feature for the classification problem.

4.2 Setup

The Label-GCN model was trained after modifying the Stellargraph library [3] to include a layer implementing eq. (3).

The experiments were performed in both a transductive (CORA, PubMed, CitSeer, Elliptic) and inductive (Elliptic) setting. The models architectures for the transductive setting are shown in Table 2. The CORA, CiteSeer and PubMed datasets were trained with two hidden layers with 16 neurons each, the Elliptic dataset was trained with two hidden layers of 100 neurons each. For each model, the number of epochs for training were chosen using early stopping with a set patience. The convolution in the first layer was either calculated using eq. (1), for the standard GCN, or the modified convolution, eq. (3), for Label-GCN.

For a fair evaluation of the Label-GCN approach, the graphs were split in four different sets: a training, validation, test and support set. During the training phase of Label-GCN, the model can only see the labels associated with the training and validation set. Through these labels, the model learns how much it can rely on the neighbors' labels to make a prediction. During the inference phase, the model is also able to see a given fraction of the labels from the support set (keeping the test set always unseen). As the fraction of labels that are available through the support set increases, our results show that the model performance always improves. A summary of this "workflow" is shown in Figure 3.

For the Elliptic dataset in the inductive setting, following [27] the model was trained for 1000 epochs using a learning rate of 0.001, still using 100 neurons for the hidden layers. Finally, the label features were one-hot encoded for the CORA, CiteSeer and PubMed datasets. For the Elliptic dataset, the mapping "licit" \rightarrow -1, "unknown" \rightarrow 0, "illicit" \rightarrow 1 was applied.



Figure 3: Summary of the setup used for evaluating the Label-GCN approach.

4.3 Results

4.3.1 Transductive setting

Results for the CORA, CiteSeer and PubMed datasets are summarised in Table 3. The accuracy and standard deviation for the CORA and CiteSeer models were calculated using 100 random train/validation/test/support splits and 10 random initializations for each. The accuracy and standard deviation for the PubMed dataset were calculated using 20 splits and 10 random initializations.

	Label % (total)	Label % (support)	CORA	CiteSeer	PubMed
GCN	-	-	79.3 ± 2.9	64.8 ± 3.4	77.2 ± 3.2
Label-GCN	10/7/1	0	79.9 ± 2.8	64.5 ± 3.5	77.5 ± 3.0
	15	6/9/16	80.4 ± 2.8	65.1 ± 3.4	81.4 ± 1.9
	30	25/27/33	82.1 ± 2.7	66.4 ± 3.4	82.9 ± 1.5
	60	62/64/66	84.7 ± 2.4	68.4 ± 3.2	84.0 ± 1.3
	90	100	86.4 ± 2.2	70.3 ± 3.1	84.3 ± 1.3

Table 3: Mean test set accuracy and standard deviation for the CORA, CiteSeer and PubMed datasets. For Label-GCN, the total fraction of labels that are available during inference is also shown. Given the different splits of Table 1, when the support set is fully unlabelled (0%) the datasets are 10%/7%/1% labelled for the CORA, CiteSeer and PubMed datasets respectively.

		Elliptic				
	Label % (total)	Label % (support)	Precision	Illicit Recall	F ₁	Accuracy
GCN	-	-	86.5 ± 1.7	72.8 ± 2.2	79.0 ± 1.1	96.2 ± 0.2
Label-GCN	20	0	86.3 ± 1.4	74.6 ± 2.1	80.0 ± 1.6	96.4 ± 0.3
	50	50	89.1 ± 1.4	74.7 ± 2.3	81.3 ± 1.7	96.6 ± 0.3
	60	67	89.6 ± 1.6	74.9 ± 2.1	81.6 ± 1.6	96.7 ± 0.3
	70	83	90.4 ± 1.4	75.2 ± 2.5	82.0 ± 1.7	96.8 ± 0.3
	80	100	90.7 ± 1.5	75.2 ± 2.4	82.2 ± 1.7	96.8 ± 0.3

Table 4: Results of the experiment for the Elliptic dataset (transductive setting).

The results for the Elliptic dataset are summarised in Table 4. For this experiment, the accuracy and standard deviation was calculated using 5 random train/validation/test/support splits and 2 random initializations. The experiment was treated as an anomaly detection problem on the "illicit" class.

Overall, both tables show that the performance of Label-GCN is superior to the performance of the GCN model alone, with the trend improving in favour of Label-GCN as more labels become available during the inference phase.

4.3.2 Inductive setting

Label-GCN was tested in an inductive setting on the Elliptic dataset. Following [27], various models were compared against each other. In particular, Label-GCN was compared against standard GCN, logistic regression (LR) and random forest (RF). All models were trained using the first 34 time steps $t_1 \dots t_{34}$ of the dataset and tested against the remaining 15 time steps $t_{35} \dots t_{49}$. Default parameters were used for LR, the RF model was run with $n_estimators=50$ and $max_features=50$ (sklearn library, version 0.23.2). In order to mitigate the imbalance in the "illicit" class, for GCN and Label-GCN the illicit nodes were over-sampled by a factor x6.

Method	Illicit				Accuracy
	Precision	Recall	F ₁	$F_{1,t \geq t_{43}}$	
Logistic Regr ^{AF}	29.5 ± 0.0	65.8 ± 0.0	40.8 ± 0.0	5.0 ± 0.0	87.6 ± 0.0
Logistic Regr ^{AF+NE-GCN}	83.8 ± 2.7	39.0 ± 2.6	53.1 ± 2.5	1.7 ± 0.9	95.5 ± 0.2
Logistic Regr ^{AF+NE-Label-GCN}	92.9 ± 1.2	57.9 ± 3.0	71.3 ± 2.2	22.2 ± 5.2	97.0 ± 0.2
RandomForest ^{AF}	92.7 ± 1.3	72.1 ± 0.2	81.1 ± 0.5	3.0 ± 0.1	97.8 ± 0.1
RandomForest ^{AF+NE-GCN}	94.1 ± 1.7	67.7 ± 0.6	78.7 ± 0.7	3.0 ± 0.5	97.6 ± 0.1
RandomForest ^{AF+NE-Label-GCN}	97.4 ± 1.0	61.5 ± 2.7	75.4 ± 2.2	8.3 ± 4.0	97.4 ± 0.2
GCN	61.6 ± 3.8	52.2 ± 1.6	56.4 ± 1.4	1.5 ± 0.6	94.7 ± 0.3
Label-GCN	83.3 ± 3.0	69.2 ± 2.2	75.5 ± 0.3	36.4 ± 3.2	97.1 ± 0.1

Table 5: Results of the experiment for the Elliptic dataset (inductive setting). Following [27], *AF* refers to all features available in the dataset. *NE-GCN* denotes the node embeddings computed through GCN, and *NE-Label-GCN* the node embeddings computed through Label-GCN. The bottom part of the table shows the performance of the GCN and Label-GCN models alone. The column $F_{1,t \geq t_{43}}$ shows the model performance after the dark market shutdown.

The testing phase was achieved in the following manner: at each time step t_j ($j > 34$), the adjacency matrix was updated to reflect the graph corresponding to all times $t \leq t_j$. Then, for each test node at time $t = t_j$, full information of the node labels in the graph ($t \leq t_j$) was assumed. In other words, for each test node 100% of the labels were made available, except for the test node itself.

Table 5 presents the results of the experiment. They are overall in line with [27]. The performance of the standard GCN model is significantly improved by the addition of the neighbors’ label as a learning feature, through Label-GCN. However, the use of node embeddings from Label-GCN only leads to a considerable improvement of the performance for LR, while no such improvement is observed for RF, in terms of F1-score. In general, it is observed that using embeddings from GCN results in a higher precision score, at the expense of recall. Interestingly, whilst all models suffer from the dark market shutdown at $t = t_{43}$ [27], label propagation aids the robustness of the models against occurrences such as this one. This is shown through the column $F_{1,t \geq t_{43}}$ of Table 5, where the F_1 -score after $t = t_{43}$ shows a measurable improvement when using Label-GCN, despite still being low in absolute values. Although feature engineering and further optimizations could lead to better results, these are outside the scope of the present paper.

5 Conclusion

We have developed Label-GCN, a modification of GCN that allows the model to learn from available labels. We have shown that this can lead to substantial improvements of the model performance, when labels are available during both the training and inference phases. Implementation of the Label-GCN model requires minor changes compared to the standard GCN implementation, without the addition of hyper-parameters, and works in both a transductive and inductive setting. We expect Label-GCN to be also applicable to the many variants of GCN that have been proposed over time, with potential benefits especially in terms of applicability to large scale applications.

Acknowledgements

We thank useful discussions with Jonty Page.

References

- [1] ANDROULAKI, E., KARAME, G. O., ROESCHLIN, M., SCHERER, T., AND CAPKUN, S. Evaluating user privacy in bitcoin. In *International Conference on Financial Cryptography and Data Security* (2013), Springer, pp. 34–51.
- [2] CHEN, J., MA, T., AND XIAO, C. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247* (2018).
- [3] CSIRO’S DATA61. Stellargraph machine learning library. <https://github.com/stellargraph/stellargraph>, 2018.
- [4] DONG, H., CHEN, J., FENG, F., HE, X., BI, S., DING, Z., AND CUI, P. On the equivalence of decoupled graph convolution network and label propagation. *arXiv preprint arXiv:2010.12408* (2020).
- [5] DU, J., ZHU, F., AND LIM, E.-P. Dynamic label propagation in social networks. In *Database Systems for Advanced Applications* (Berlin, Heidelberg, 2013), W. Meng, L. Feng, S. Bressan, W. Winiwarter, and W. Song, Eds., Springer Berlin Heidelberg, pp. 194–209.
- [6] ELLIPTIC. Elliptic Data Set. <https://www.kaggle.com/ellipticco/elliptic-data-set>, 2019.
- [7] GONG, C., TAO, D., LIU, W., LIU, L., AND YANG, J. Label propagation via teaching-to-learn and learning-to-teach. *IEEE transactions on neural networks and learning systems* 28, 6 (2016), 1452–1465.
- [8] HAMILTON, W. L., YING, R., AND LESKOVEC, J. Inductive representation learning on large graphs. *arXiv preprint arXiv:1706.02216* (2017).
- [9] HARRIGAN, M., AND FRETTER, C. The unreasonable effectiveness of address clustering. In *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress* (2016), IEEE, pp. 368–373.
- [10] HU, Y., SENEVIRATNE, S., THILAKARATHNA, K., FUKUDA, K., AND SENEVIRATNE, A. Characterizing and detecting money laundering activities on the bitcoin network. *arXiv preprint arXiv:1912.12060* (2019).

- [11] JAAKKOLA, M. S. T., AND SZUMMER, M. Partially labeled classification with markov random walks. *Advances in neural information processing systems (NIPS) 14* (2002), 945–952.
- [12] KARASUYAMA, M., AND MAMITSUKA, H. Manifold-based similarity adaptation for label propagation. *Advances in neural information processing systems 26* (2013), 1547–1555.
- [13] KIPF, T. N., AND WELLING, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
- [14] MEIKLEJOHN, S., POMAROLE, M., JORDAN, G., LEVCHENKO, K., MCCOY, D., VOELKER, G. M., AND SAVAGE, S. A fistful of bitcoins: characterizing payments among men with no names. In *Proceedings of the 2013 conference on Internet measurement conference* (2013), pp. 127–140.
- [15] NAMATA, G., LONDON, B., GETOOR, L., HUANG, B., AND EDU, U. Query-driven active surveying for collective classification. In *10th International Workshop on Mining and Learning with Graphs* (2012), vol. 8.
- [16] OLIVEIRA, C., TORRES, J., SILVA, M. I., APARÍCIO, D., ASCENSÃO, J. T., AND BIZARRO, P. Guiltywalker: Distance to illicit nodes in the bitcoin network. *arXiv preprint arXiv:2102.05373* (2021).
- [17] RAGESH, R., SELLAMANICKAM, S., LINGAM, V., AND IYER, A. A graph convolutional network composition framework for semi-supervised classification. *arXiv preprint arXiv:2004.03994* (2020).
- [18] RAGHAVAN, U. N., ALBERT, R., AND KUMARA, S. Near linear time algorithm to detect community structures in large-scale networks. *Physical review E* 76, 3 (2007), 036106.
- [19] REID, F., AND HARRIGAN, M. An analysis of anonymity in the bitcoin system. In *Security and privacy in social networks*. Springer, 2013, pp. 197–223.
- [20] RON, D., AND SHAMIR, A. Quantitative analysis of the full bitcoin transaction graph. In *International Conference on Financial Cryptography and Data Security* (2013), Springer, pp. 6–24.
- [21] SEN, P., NAMATA, G., BILGIC, M., GETOOR, L., GALLIGHER, B., AND ELIASSI-RAD, T. Collective classification in network data. *AI magazine* 29, 3 (2008), 93–93.
- [22] SHCHUR, O., MUMME, M., BOJCHEVSKI, A., AND GÜNNEMANN, S. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868* (2018).
- [23] VISWANATHAN, K., SACHDEVA, S., TOMKINS, A., AND RAVI, S. Improved semi-supervised learning with multiple graphs. In *The 22nd International Conference on Artificial Intelligence and Statistics* (2019), PMLR, pp. 3032–3041.
- [24] WANG, B., TU, Z., AND TSOTSOS, J. K. Dynamic label propagation for semi-supervised multi-class multi-label classification. In *Proceedings of the IEEE international conference on computer vision* (2013), pp. 425–432.
- [25] WANG, H., AND LESKOVEC, J. Unifying graph convolutional neural networks and label propagation. *arXiv preprint arXiv:2002.06755* (2020).
- [26] WANG, J., WANG, F., ZHANG, C., SHEN, H. C., AND QUAN, L. Linear neighborhood propagation and its applications. *IEEE transactions on pattern analysis and machine intelligence* 31, 9 (2008), 1600–1615.
- [27] WEBER, M., DOMENICONI, G., CHEN, J., WEIDELE, D. K. I., BELLEI, C., ROBINSON, T., AND LEISERSON, C. E. Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. *arXiv preprint arXiv:1908.02591* (2019).
- [28] WU, F., SOUZA, A., ZHANG, T., FIFTY, C., YU, T., AND WEINBERGER, K. Simplifying graph convolutional networks. In *International conference on machine learning* (2019), PMLR, pp. 6861–6871.
- [29] WU, X.-M., LI, Z., SO, A. M.-C., WRIGHT, J., AND CHANG, S.-F. Learning with partially absorbing random walks. In *NIPS* (2012), vol. 25, pp. 3077–3085.
- [30] ZHANG, Z., CUI, P., AND ZHU, W. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering* (2020).
- [31] ZHOU, D., BOUSQUET, O., LAL, T. N., WESTON, J., AND SCHÖLKOPF, B. Learning with local and global consistency. *Advances in neural information processing systems 16*, 16 (2004), 321–328.
- [32] ZHOU, J., CUI, G., ZHANG, Z., YANG, C., LIU, Z., WANG, L., LI, C., AND SUN, M. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434* (2018).
- [33] ZHU, X., AND GHARAMANI, Z. Learning from labeled and unlabeled data with label propagation.