Metric learning inspired binary and multiclass classification on graphs

Nischal R Bhat
PES University
Dept. of CSE
Bengaluru, India
nischal108bhat@gmail.com

Prerana Sanjay Kulkarni
PES University
Dept. of CSE
Bengaluru, India
prer.kulk@gmail.com

Ananya Menon
PES University
Dept. of CSE
Bengaluru, India
ananya4am@gmail.com

Smruthika S Meda
PES University
Dept. of CSE
Bengaluru, India
2002smruthika@gmail.com

Prof. Bhaskarjyoti Das
PES University
Dept. of CSE
Bengaluru, India
bhaskarjyoti01@gmail.com

***Abstract*—In a task such as graph classification or node classification, there are usually a few data-points that are harder to classify than others and lie in a rather grey area. These are at the risk of being wrongly classified and bringing down the accuracy of our classification task. Metric learning is what comes in useful here. The Transformer architecture is used to capture attention and long-range dependencies in non-IID datasets, to get well-learned embeddings.**
We suggest a novel pipeline, which combines a Graph Transformer Network with Metric Learning methodologies in order to make classification tasks easier. In this paper, we focus on two metric learning frameworks - the Siamese Network (for binary classification of graphs) and the Prototypical Network (for multilabel classification of nodes). Through this paper, we aim to perform two tasks. The first task is binary graph classification using a Graph Transformer Network and a Siamese Network, and the second task is multiclass node classification using a Graph Transformer Network and a Prototypical Network.

I. Introduction

The requirement for efficient graph representation learning has drawn a lot of interest, owing to the development of graph-structured data in a variety of disciplines. Metric learning shows tremendous potential for improving tasks involving graphs, since it is a potent strategy that moves similar things closer and different things apart. Graph Transformers provide context-aware embeddings from graphs by capturing local and global dependencies. They are motivated by successful natural language processing (NLP) designs.

In order to overcome issues with graph data processing, this research article investigates the merging of metric learning and graph transformers. By combining these techniques, we hope to increase discriminative abilities, and enhance similarity-based retrieval and anomaly detection in graph-structured data. We further the practical applications of graph representation learning through theoretical analysis and experimental validation on benchmark datasets.

II. Background

A. Graph Representation Learning

Graph representation learning seeks to preserve important structural and semantic aspects of a graph, while converting complex graph-structured data into instructive and low-dimensional embeddings. Graph representation techniques enable effective and scalable processing of huge graphs in a variety of machine learning tasks, by mapping nodes and edges to continuous vector spaces. These embeddings effectively support tasks like node classification, link prediction, and graph clustering, by capturing the underlying linkages, neighbourhood structures, and local/global dependencies present in the graph. The difficulty lies in creating robust, scalable algorithms that can extract meaningful representations from heterogeneous, dynamic graphs for use in practical applications.

B. Graph Transformers

Graph Transformers are an extension of the Transformer architecture, which is widely used for NLP (Natural Language Processing) tasks. It is an add-on to the existing Graph Convolutional Network(GCN), due to its value addition in terms of capturing the attention of a node compared to its neighbours. The transformer architecture uses the common query, key and vector concepts and extends it to graph context. They are used to compute attention weights by capturing relationships between nodes in a graph.

The following equations represent the key components of the Graph Transformer:

1. Attention Mechanism:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \qquad (1)$$

where Q, K, and V are the query, key, and value matrices respectively, and d_k is the dimension of the key matrix.

2. Multi-Head Attention:

$$MultiHead(Q, K, V) = Concat(h_1, h_2, ..., h_n)W_O \quad (2)$$

where $h_i = Attention(QW_{Qi}, KW_{Ki}, VW_{Vi})$, n is the number of attention heads, and W_{Qi}, W_{Ki}, W_{Vi}, and W_O are learnable weight matrices.

3. Graph Transformer layer:

$$x' = W_1 x_i + \Sigma \alpha_{ij} W_2 x_j \quad (3)$$

where alpha stands for the below

$$\alpha_{ij} = softmax(\frac{(W_3 x_i)^T (W_4 x_j)}{\sqrt{d}}) \quad (4)$$

C. Metric Learning

Metric learning is a fundamental approach in machine learning that aims to learn a suitable distance metric in the feature space, allowing for improved discrimination between data samples. Traditional machine learning algorithms often assume that data points are independently and identically distributed (IID), which might not hold for complex data structures like graphs. In the context of graph data, metric learning offers a compelling solution by embedding nodes into a lower-dimensional space, where similarity relationships among nodes are better preserved. The learned embeddings capture the inherent structure of the graph, enhancing the performance of downstream tasks such as node classification, link prediction, and graph clustering.

1) Siamese Network: Building on the principles of metric learning, Siamese networks have emerged as a popular architecture, demonstrating remarkable success in various domains. Siamese networks leverage their ability to learn a similarity metric that distinguishes between similar and dissimilar pairs, making them particularly valuable for learning from limited labeled data. Extending this approach to graph data, Siamese networks enable the capture of pairwise similarities between graphs, facilitating the learning of meaningful embeddings that encode graph topology. By incorporating Siamese networks into graph representation learning, we can leverage their effectiveness in learning similarity metrics and enhance the performance of graph-related tasks, while efficiently handling scenarios where obtaining ground truth labels is challenging or expensive.

2) Prototypical Learning: On a parallel note, prototypical learning offers a powerful paradigm by learning prototypes or centroids of classes in the embedding space. By creating representative embeddings for each node class, prototypical learning serves as reference points for classification and similarity-based tasks, addressing challenges related to data scarcity and generalization on unseen graph structures. The efficiency of prototypical learning is particularly noteworthy, where classification can be performed with only a few labeled samples per class. By seamlessly integrating prototypical learning into metric learning on graphs, we harness its ability to learn compact and informative representations, culminating in improved efficiency and superior performance on a wide range of graph-related tasks. The combination of metric learning, Siamese networks, and prototypical learning represents a compelling frontier in graph representation learning, unlocking new potentials for real-world applications.

III. Datasets

A. Siamese Network Pipeline-The UPFD Dataset

The User Preference-aware Fake News Detection (UPFD) dataset is a dataset of graphs, where each graph refers news propagation network obtained from Twitter and validated by Politifact.

Each node in the graph represents a user (who is part of the news propagation). The root node is the source of the news. An edge between users represents a retweet. There are a total of 314 graphs in this dataset, and each graph can represent either real news or fake news. Hence, this is a case of binary classification of graphs.

B. Prototypical Network Pipeline-The CoAuthor Dataset

This dataset consists a single graph, where each node is represented by an author. An edge exists between 2 nodes means that the two authors have co-authored a paper.

We will be using CoAuthor CS dataset, from Pytorch Geometric, which is exclusively for Computer Science papers. Each node has can have 1 out of 15 labels, where each label represents a field of study. This dataset is suitable for node classification task.The dataset has around 18333 nodes and 1 graph with 163788 edges. The labels in the range 0-14 represent the different scientific domains that the authors can belong to, such as Machine Intelligence, Human Computer Interaction, etc.

IV. Methodology

A. Graph Transformer - Siamese Network Pipeline for Binary Classification of Graphs

In this section, we first focus on building a Graph Transformer Network to obtain graph embeddings. Next, we use the learned embeddings as inputs to a Siamese Network, which will improve the accuracy of the classification task.

1. The Graph Transformer Network - Obtaining Graph Embeddings:

The first stage of this task, as mentioned, is the Graph Transformer Network. We use a pre-defined split for the training, testing and validation datasets (61-32-221 respectively). The transformer network is constructed with 6 layers in total. First comes a regular graph convolutional layer, followed by a linear layer. This is followed by a transformer layer with 3 heads, a linear layer, another transformer layer with 3 heads, and a final

Fig. 1. Flowchart encapsulating the process.

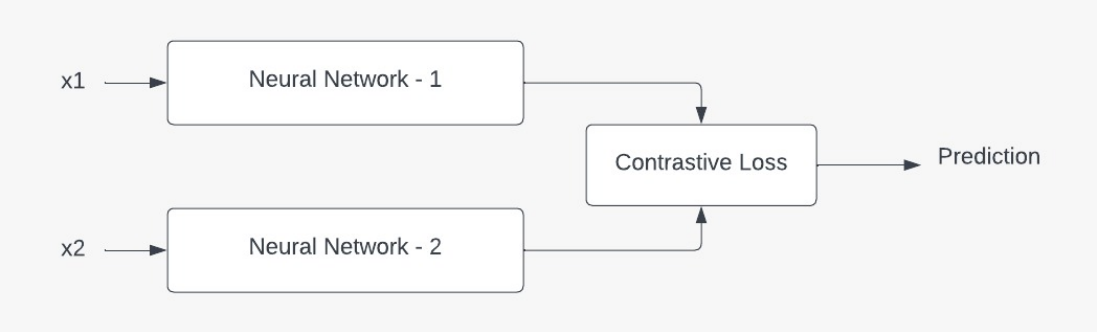

Fig. 2. Siamese network.

linear layer. The graph's embeddings are obtained through a global max-pooling function applied on the node embeddings. It is a pooling operation that arrives at a single embedding for the entire graph. In our case, each graph has a 16-dimensional embedding.

2. The Siamese Network - Using the Graph Embeddings for the Binary Classification Task:
We now use these graph embeddings in our Siamese Network. For this, positive and negative pairs of embeddings must be created. Positive pairs are created from two instances of the same class (both real news or both fake news) and negative pairs are created from two instances of different classes (one real news and the other fake). The Siamese network comprises two identical neural networks, which are trained using the pairs we create. The instances of a pair are passed through the twin neural networks in the training process. A similarity function is used to compare the outputs of the neural networks. We use a Contrastive Loss function based on Euclidean distance, as the similarity function.

The lesser the distance between two instances in the metric space, the higher the similarity between them, and the higher the probability of them belonging to the same class. Using this similarity function and a certain threshold value to compare the calculated similarities, positive pairs are moved closer together in the metric space, and negative pairs are moved further apart. This process when repeated multiple times with all the pairs created, tends to move the similar instances closer together, and dissimilar instances farther apart.

B. Graph Transformer - A Prototypical Network Pipeline for Multiclass Classification of nodes in a graph

1) The Graph Transformer Network-Obtaining node embeddings: In this section, we first focus on building a Graph Transformer Network to obtain node embeddings in a graph.Here we use one Graph Convolutional Layer and two Graph Transformer Layers- with three convolutional layers in total. These layers are followed by batch normalization, non-linear 'ReLU' activation and dropout. The second Transformer convolutional layer for multiheaded attention uses 2 heads and the third Transformer convolutional layer subsequently uses 4 heads. The final output from the third layer is fed into the linear output layer to obtain the logits for each class.The model also returns embeddings, which holds the learned node embeddings obtained after the last convolutional layer. These embeddings are used for few-shot multiclass node classification methodology.

2) Working of a Prototypical network: Construction of a Prototypical network involves splitting of training and testing data into support and query sets. Computation or training is done on the support sets,and evaluation of the model is done on the query sets.The support set contains a few samples for every class/label that the model is trained upon. This trained model is then evaluated on the query set that contains untrained/unseen examples. The entire objective is to generalize to this query set,by training with a few examples on the support set. During Prototypical Training, the support set of the train data is used to compute centroids or prototypes, which are representations of every label that we are classifying to. The entire idea is to create a metric space, containing these centroids. Examples from the query set of the train data is then projected on to this metric space containing the centroids. The distance metric we use here is "Euclidean distance". The Euclidean distance is measured between every query point in the query set and all the prototypes in the metric space. The least euclidean distance between the query point and the prototype is the prototype and subsequently, the label that the query point is classified into. During Prototypical Testing, the model is trained for a certain number of episodes and the trained model is used to do evaluation on the query set of the testing data.The episodic training accuracy and episodic testing accuracy is noted.

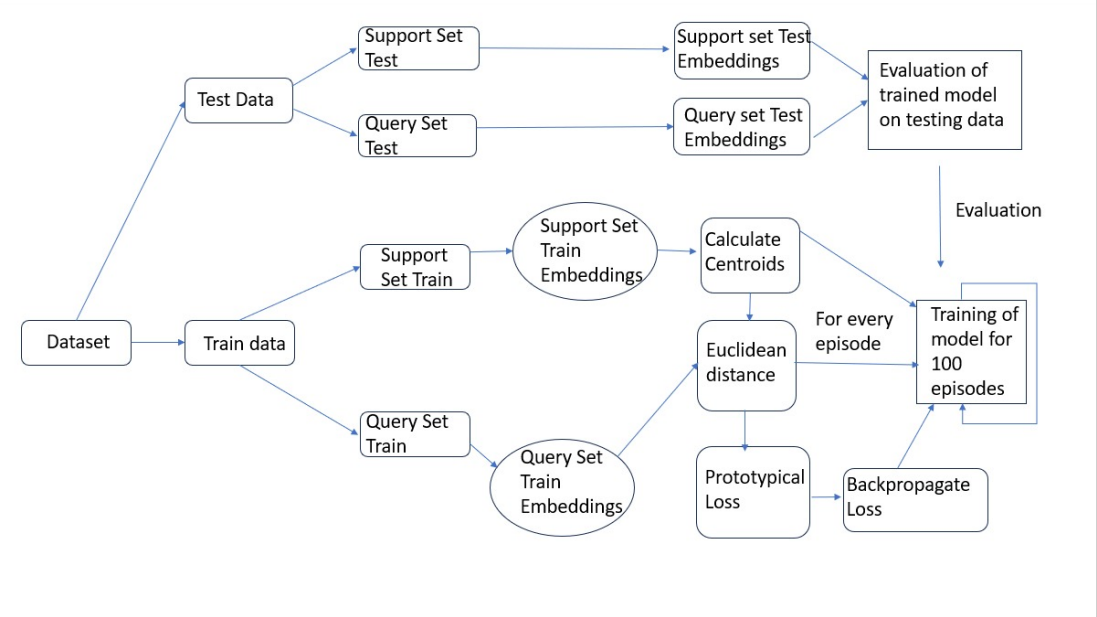

Fig. 3. Flow chart for prototypical learning.

V. Experimentation

A. Graph Transformer - Siamese Network Pipeline for Binary Classification of Graphs

1. Training data and the model:

The training data consists of 61 graphs, which is the default one provided by the dataset functionality. We use a combination of 1 GCN and 2 Transformer layers (each with 3 heads). The idea behind using a combination of different layers is based on observations from various configurations of layers. Any layers used in model exceeding three is likely to result in over-smoothing, where all embeddings seem to converge to extremely similiar values.

2. Model configuration and training:

The model calculates node representations and then performs global max pooling to obtain the representations for a graph. We train the model, using NLLLoss (Negative Log Likelihood Loss), using and ADAM (Adaptive Moment Estimation). Passing all the graphs through the trained model provides us with embeddings for all graphs in the dataset. The dimension of embedding per graph in our case is 16 dim.

3. Construction of pairs:

These embeddings are then split in multiple train test ratios. We construct the required pairs for each anchor in the training data i.e. one positive and one negative pair. Our objective as a metric learning is to exploit the similarity to differentiate between the 2 possible classes. The Siamese network model is defined and we use Contrastive Loss function and ADAM optimizer for this purpose. The similarity metric used here is Euclidean Distance, which needs to be greater to differentiate dissimilar classes.

4. Testing and evaluation:

We then test by creating pairs from the test data. Evaluation is done, in which accuracy is used to tell the quality of the pipeline

B. Graph Transformer - A Prototypical Network Pipeline for Multiclass Classification of nodes in a graph

1. Splitting of training and testing data into support and query sets:

The train and test data have an 80-20 split. They are further split into support and query sets. In a prototypical learning scenario,the support set contains a few labelled nodes per class used for training, and the query set contains the remaining labeled nodes used for testing during training.The idea is to allow the model to adapt and learn from a limited number of labeled nodes per class. Here, the few labelled nodes are experimented for 1,5,10,15 and 20. These nodes for the support set are chosen in a random fashion in order for the model to generalize better on query sets. The support and query sets for the test data, are split in a similar fashion.

2.Prototypical Training-Data and Model setup:

The criterion used is nn.NLLLoss() defines the negative log-likelihood loss, which is commonly used in classification tasks. The optimizer used is optim.Adam(model.parameters(), lr=0.01) initializes the Adam optimizer to update the model parameters during training. The scheduler StepLR with step size 50 and gamma value 0.5 adjusts the learning rate every 50 episodes The model is then set to training mode using model.train()

3.Prototypical Training- Training phase:

A training loop is performed for 100 episodes-In each episode, 1)It computes the centroids/prototypes for the support set using pre-computed support set embeddings. 2)Calculates the distances between query set embeddings and the class centroids. 3)Calculates prototypical loss using negative log

likelihood loss, from softmax probabilities. 4)Uses this loss for back propagation and parameter updates. 5)Episodic train accuracy is computed for the support set, and episodic test accuracy is computed for the query set. 4)The average episodic train accuracy is finally computed for over a 100 episodes,as the sum of train accuracies divided by the total number of episodes.

4.Prototypical Testing-Testing Phase:
Evaluation of the trained model is done on the testing set.It samples a few labeled examples per class to create support and query sets, then computes class centroids and uses them to classify the query nodes.The average episodic test accuracy gives one a measure of how well a model can generalize to unseen,unlabelled,untrained samples.

VI. Results

A. Graph Transformer - Siamese Network Pipeline for Binary Classification of Graphs

The initial training on transformer network model gives us a training accuracy of 1, with testing accuracy of 0.855. This seems to be a good result, which can generate the required 16 dimensional embeddings.

The whole Siamese Network model training and testing was done with different train-test splits. Based on the below output, in the confusion matrix, we observe that for 50-50 split, the model gives a good accuracy of 0.978. The choice of 50-50 split has to do with that the quality of embeddings obtained were good to begin with after passing it through transformer network architecture

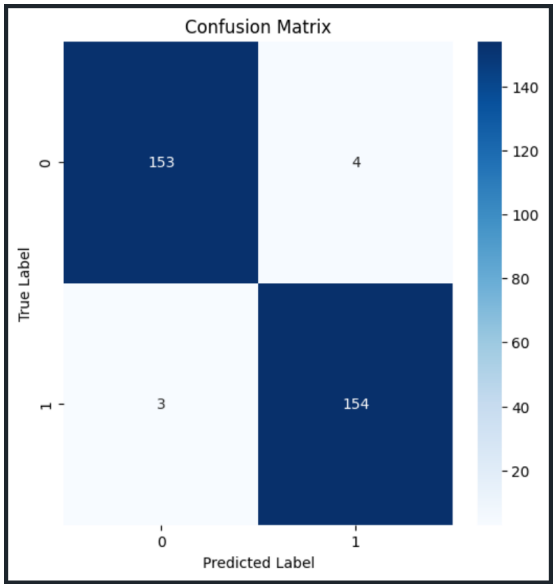

Fig. 4. Confusion matrix for 50-50 split.

Graph Transformer Network:			
	Train	Validation	Test
3 heads	1.0000	0.7742	0.8552
Siamese Network:			
Testing accuracy	0.9777		
Precision	0.9747		
Recall	0.9809		
F1 score	0.9777		
Area Under Curve	0.9777		

Fig. 5. Results of above configuration.

As we observe here, we get an accuracy of 0.978. The 0 means negative pair and 1 means positive pair in the context of above diagram. The true positives and true negatives are very low compared to the correct predictions.

B. Graph Transformer - A Prototypical Network Pipeline for Multiclass Classification of nodes in a graph

The trained model achieved an impressive average episodic test accuracy of 0.9062 during the testing/evaluation stage for 15 samples every label of the support set, over 100 episodes. This outcome demonstrates the model's effectiveness in generalizing to novel classes based on a limited number of support samples. The model accurately classified query samples from previously unseen classes, indicating its potential applicability in scenarios where data from novel classes is scarce. During the training process, the model achieved an average episodic train accuracy of 0.9679 for 15 samples of the support set during the training phase over 100 episodes. This high accuracy on the training set illustrates the model's ability to learn effectively from support samples during training episodes. The model successfully leveraged the limited support samples to learn class prototypes, leading to better generalization to unseen classes. To gain deeper insights into the model's performance, we plotted the confusion matrix for the query test data. The confusion matrix visually represents the model's ability to distinguish between different research areas. The matrix showcases how well the model performed in predicting the true labels of the query samples. The figure below presents the confusion matrix. The confusion matrix represents a comparison between actual and predicted labels by the model during the evaluation stage on query test data. Higher the intensity in colour signifies high count and represents that the particular label/class has been accurately classified with most number of nodes. In the figure, it is found that class or label '13' is the most accurately classified on the query test data with 148400 query nodes. Class or label '3' is the second most accurately classified with 80000 nodes.

Fig. 6. Confusion matrix for 50-50 split.

The diagonal elements of the confusion matrix represent the correctly classified samples, while the off-diagonal elements correspond to misclassifications. The high values on the diagonal indicate that our model performed well in classifying query samples from different research areas. Similarly, the figure below represents another confusion matrix on 15 samples of the support set, but on query train data during the training phase over 100 episodes.

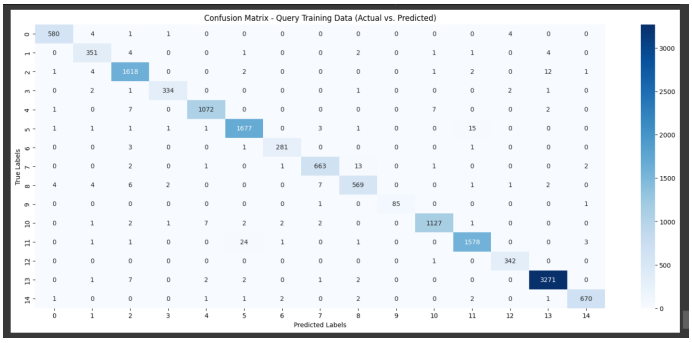

Fig. 7. Confusion matrix for k = 15.

Experimentation is done on the following values of k-1,5,10,15,20 which represents the number of samples taken in the support set during training. For 100 episodes, the value of k=15 is found to have the highest episodic test accuracy. It is imperative to choose an optimal value of 'k' during training in order for the model not to overfit or underfit.

The table below represents the average episodic train and test accuracy for different values of 'k'.

Column 1	Column 2	Column 3
Number of support samples	AE Train Accuracy	AE Test Accuracy
1	0.9247	0.7615
5	0.9558	0.8939
10	0.9746	0.8993
15	0.9679	0.9062
20	0.9727	0.9038

TABLE I
TABLE FOR DIFFERENT VALUES OF K.

Support Set Size of 1: This represents an extreme type of learning scenario where the model must learn from just one labelled example per class. This setting is challenging and tests the model's ability to generalize from very limited information.

Support Set Size of 5: This is a common choice for prototypical learning tasks. It provides a small but more diverse set of examples for each class, allowing the model to learn better representations and potentially improve its performance.

Support Set Size of 10-20: As we increase the support set size, the model has more labelled examples to learn from, leading to better representations and potentially higher performance on a wide range of tasks.

Overall, the model shows consistent and strong performance both on the training and testing sets. The high average episodic test accuracy demonstrates the model's ability to generalize to novel classes with limited labelled data, which is a significant advantage in learning scenarios where the labelled data may be scarce or limited.

VII. Conclusion

In this paper, we proposed a pipeline for metric learning using Siamese Networks and Graph Transformers on UPFD dataset. Our model achieved higher accuracy, indicating the ability of models to discriminate the negative pairs properly and also identifying the positive pairs. This high accuracy is mainly due to two reasons: the quality of embeddings obtained from graph transformers, and the nature of similarity metric used by us.

Also, we presented a learning approach utilizing Prototypical Networks on the Coauthor dataset. Our model achieved high average episodic test accuracy, demonstrating its ability to generalize to novel classes. The high episodic train accuracy reflects the model's proficiency in learning from limited support samples during training. The experimental results highlight the effectiveness of Prototypical Networks in addressing learning tasks. Our findings encourage further exploration and application of these techniques in real-world scenarios with limited labelled data.

In conclusion, metric based prototypical learning mechanism is better than traditional machine learning techniques as not only does it tend to have fewer parameters compared to some traditional deep learning architectures, it is also scalable to a large number of classes especially when data is scarce, and it enables the model to generalize well to new classes by learning a metric space where similar samples are close to their corresponding class prototype.

References

[1] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P. and Bengio, Y., 2017. Graph attention networks. arXiv preprint arXiv:1710.10903.

[2] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need. Advances in neural information processing systems, 30.

[3] Dwivedi, V.P. and Bresson, X., 2020. A generalization of transformer networks to graphs. arXiv preprint arXiv:2012.09699.

[4] Ding, K., Wang, J., Li, J., Shu, K., Liu, C. and Liu, H., 2020, October. Graph prototypical networks for few-shot learning on attributed networks. In Proceedings of the 29th ACM International Conference on Information Knowledge Management (pp. 295-304).

[5] Jin, M., Zheng, Y., Li, Y.F., Gong, C., Zhou, C. and Pan, S., 2021. Multi-scale contrastive siamese networks for self-supervised graph representation learning. arXiv preprint arXiv:2105.05682.

[6] Melekhov, I., Kannala, J. and Rahtu, E., 2016, December. Siamese network features for image matching. In 2016 23rd international conference on pattern recognition (ICPR) (pp. 378-383). IEEE.

[7] Shorfuzzaman, M. and Hossain, M.S., 2021. MetaCOVID: A Siamese neural network framework with contrastive loss for n-shot diagnosis of COVID-19 patients. Pattern recognition, 113, p.107700.

[8] Dou, Y., Shu, K., Xia, C., Yu, P.S. and Sun, L., 2021, July. User preference-aware fake news detection. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 2051-2055).

[9] Shchur, O., Mumme, M., Bojchevski, A. and Günnemann, S., 2018. Pitfalls of graph neural network evaluation. arXiv preprint arXiv:1811.05868.

[10] Yao, H., Zhang, C., Wei, Y., Jiang, M., Wang, S., Huang, J., Chawla, N. and Li, Z., 2020, April. Graph few-shot learning via knowledge transfer. In Proceedings of the AAAI conference on artificial intelligence (Vol. 34, No. 04, pp. 6656-6663). red IEEE conference templates contain

guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove the template text from your paper may result in your paper not being published.