Meta Learning Assisted Graph Representation Learning for Downstream Learning Tasks

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

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

***Abstract*—Graph Transformer Network (GTN) is a state-of-the-art graph representation learning method that uses the transformer architecture to capture attention and long-range dependencies in a non-euclidean dataset such as graph. However, even with GTN in a downstream learning task, there are usually a few data points that are harder to classify than others and are at risk of being wrongly classified. Metric-based meta-earning, a family of meta-learning algorithms, can be useful in such scenarios. In this work, we propose a novel 2-stage pipeline, which combines a Graph Transformer Network with metric-based meta-learning methodologies in order to make downstream classification tasks easier. The work described in this paper investigates and evaluates this pipeline in two specific learning setups using two Metric Learning frameworks i.e. Siamese Network for binary classification of graphs and Prototypical Network for multi-class classification of nodes, both appended to a representation learning block of Graph Transformer Network. The experiments described in this paper conclusively prove the efficacy of such a pipeline both in full-shot and few-shot learning scenarios.**

***Index Terms*—graph transformer, meta-learning, siamese network, prototypical networks**

I. Introduction

Graph Transformers provide context-aware embedding from graphs by capturing local and global dependencies. They are motivated by successful natural language processing (NLP) designs. Metric-based meta-learning, a family of meta-learning algorithms, is a potent strategy in downstream machine-learning tasks as it moves similar things closer and different things apart. The work described in this paper 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. Existing Work

A. Graph Transformers

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 Transformers[1] 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 neighbors. The transformer architecture[2], [3] uses the common query, key, and vector concepts and extends them to the 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:

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

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:

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

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:

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

where alpha stands for the below:

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

Graph Transformer network has been recently used in few GNN based architectures i.e. document classification[4], hateful discussion detection[5] and fake news detection[6] but it has not been used in conjunction with metric-based meta-learning yet.

B. Metric based meta learning

Metric-based meta-learning is a type of meta-learning[7] that aims to learn a suitable distance metric in the feature space, allowing for improved discrimination between data samples. In the context of graph data, Metric Learning offers a compelling advantage by embedding nodes into a lower-dimensional space, where similarity relationships among nodes are better preserved.

1) Siamese Network: Siamese Networks[8] is a popular metric-based meta-learning architecture that leverages its ability to learn a similarity metric that distinguishes between similar and dissimilar pairs. This approach makes it particularly valuable for learning from limited labeled data. By incorporating Siamese Networks into graph representation learning[9], 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: Prototypical Networks[10] are similar to Siamese networks for a multi-class scenario. By creating representative prototypes or embeddings for each node class, it can address challenges related to data scarcity and generalization on unseen graph structures. By incorporating Prototypical Networks into state-of-the-art graph representation learning techniques[11], we can harness the strength of both metric learning and graph representation learning.

C. Meta learning using graph neural networks

Since both graph neural networks and meta-learning are relatively new areas of research, there is a limited amount of work combining graph neural techniques with meta-learning for graph few-shot learning[12] in areas such as node classification[13], link prediction[14], and graph classification[15]. Disinformation such as fake news on social media is inherently multi-contextual[16] in nature and a graph-based learning approach is a natural fit for take care of multiple contexts such as content, propagation, and user. Specifically for multi-modal fake news detection tasks, Yaqing Wang et. al.[17] have used meta-optimization as a meta-learning approach but this work does not use meta-learning with a graph neural network-based approach. To the best of our knowledge, the proposed methodology in this work has not been attempted so far for fake news detection tasks.

III. Datasets

Two datasets have been used in this work to evaluate the performance of the proposed pipeline.

1) **UPFD Dataset:** The User Preference-aware Fake News Detection (UPFD) dataset[18] is a dataset of graphs and is used for binary classification of graphs, where each graph refers to news propagation network obtained from Twitter and validated by Politifact. Each node in the graph represents a user. 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.
2) **Coauthor CS Dataset:** This dataset[19], used for node classification, consists of a single graph, where each node is represented by an author. 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 i.e. Machine Intelligence, Human-Computer Interaction, etc. 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.

IV. Methodology

As indicated earlier, the methodology adopted in this work is a two-step process i.e. in the first step, a Graph Transformer Network is used to obtain graph embeddings. In the second step, the learned embeddings are used as inputs to metric-learning-based meta-learning block to improve the accuracy of the downstream classification task.

1) **Graph Transformer - Siamese Network Pipeline for Binary Classification of Graphs:** The Figure 1 and Figure 2 shows the pipeline used.
   - In 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 linear layer. The graph's embeddings are obtained through a global max-pooling function applied to 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.
   - These graph embeddings are used as input in the Siamese Network. For this, 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

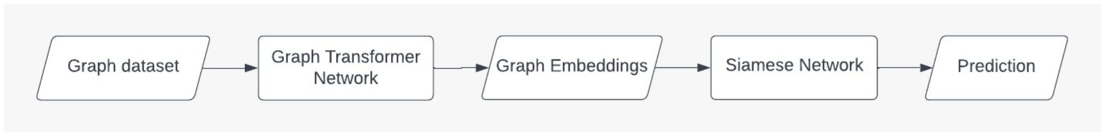

Fig. 1. Flowchart encapsulating the process.

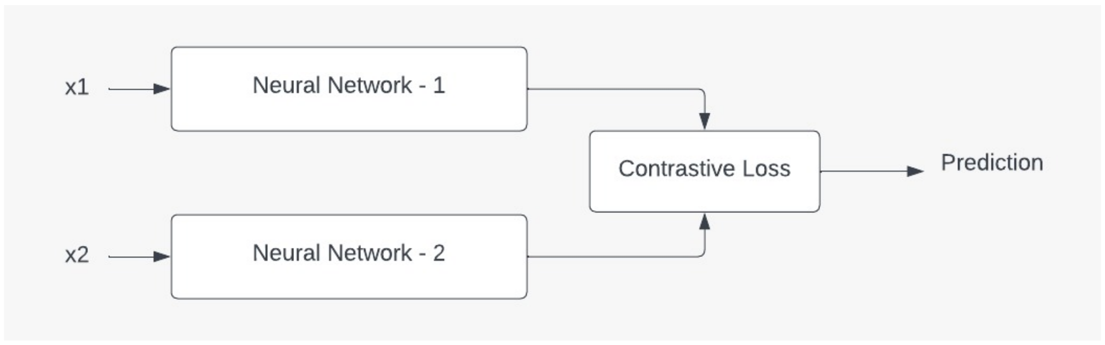

Fig. 2. Siamese Network.

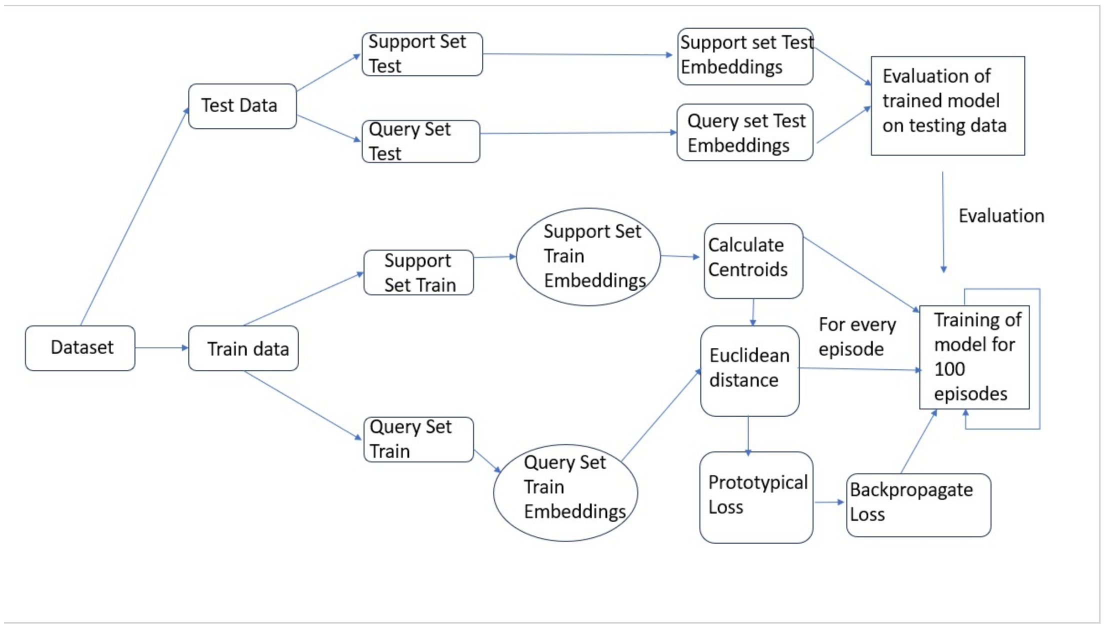

Fig. 3. Flow chart for prototypical learning.

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 training process when repeated multiple times with all the pairs created, tends to move the similar instances closer together, and dissimilar instances farther apart.

2) **Graph Transformer - Prototypical Network Pipeline for Multiclass Classification of nodes:** The Figure 3 shows the pipeline used in detail.

- First the Graph Transformer Network is used to obtain node embeddings in a graph. Here one Graph Convolutional Layer and two Graph Transformer Layers,with three convolutional layers, are used. 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 the few-shot multiclass node classification in the second step.

- Construction of a Prototypical network involves splitting 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 are then projected onto this metric space containing the centroids. The distance metric used 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 an evaluation on the query set of the testing data. The episodic training accuracy and the episodic testing accuracy are noted.

V. Experimentation

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

1) 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 a model exceeding three are likely to result in over-smoothing, where all embeddings seem to converge to extremely similar values.
2) 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) 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) 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 Multi-class Classification of nodes

1) 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 labeled 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 labeled nodes are experimented with 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) The criterion used is the negative log-likelihood loss (NLLLoss), which is commonly used in classification tasks. The optimizer used is Adam. The scheduler with step size 50 and gamma value 0.5 adjusts the learning rate every 50 episodes.
3) A training loop is performed for 100 episodes-In each episode,
   - It computes the centroids/prototypes for the support set using pre-computed support set embeddings.
   - Calculates the distances between query set embeddings and the class centroids.
   - Calculates prototypical loss using negative log likelihood loss, from softmax probabilities.
   - Uses this loss for back propagation and parameter updates.
   - Episodic train accuracy is computed for the support set, and episodic test accuracy is computed for the query set.
   - 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) 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

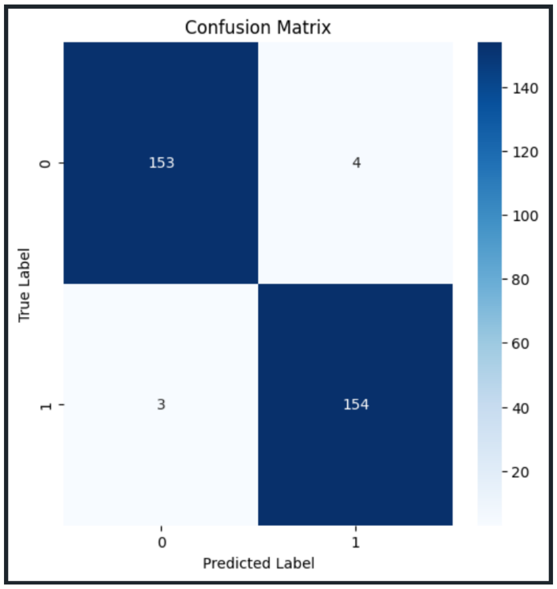

Fig. 4. Confusion matrix for 50-50 split.

TABLE I
GRAPH TRANSFORMER NETWORK WITH 3 HEADS

Train accuracy	Validation accuracy	Test accuracy
1.0000	0.7742	0.8552

TABLE II
SIAMESE NETWORK TEST PERFORMANCE

Accuracy	Precision	Recall	F1-score	Area under curve
0.9777	0.9747	0.9809	0.9777	0.9777

episodic test accuracy gives one a measure of how well a model can generalize to unseen,unlabelled, untrained samples.

VI. Results and Discussion

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

The initial training on the transformer network model gives us a training accuracy of 1, with a 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. As shown in the confusion matrix in Figure 4, 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 was good to begin with after passing it through the transformer network architecture

As observed here in Table I, we get an accuracy of 0.978. Here 0 means negative pair and 1 means positive pair in the context of the confusion matrix. The true positives and true negatives are very low compared to the correct predictions.

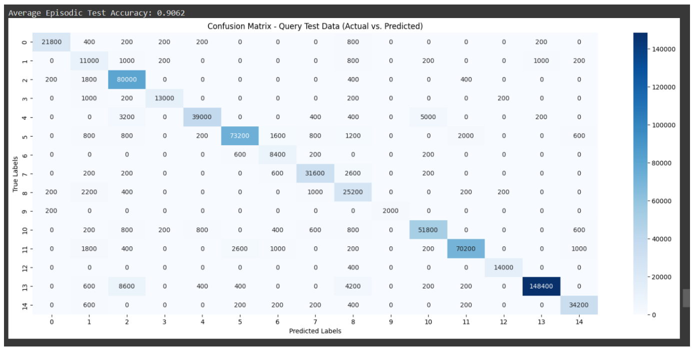
```
Average Episodic Test Accuracy: 0.9062
```

Fig. 5. Confusion matrix for Query Train Data.

TABLE III
TABLE FOR DIFFERENT VALUES OF K.

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

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. The higher intensity in color signifies a high count and represents that the particular label/class has been accurately classified with the 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.

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.

Experimentation is done on values of k i.e. 1,5,10,15,20 that represent 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.

Table III represents the average episodic train and test accuracy 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 and next steps

In this paper, we proposed and demonstrated the efficacy of a two-step pipeline for Metric based meta-learning using Siamese Networks and Prototypical Networks with Graph Transformers. Specifically on the Coauthor CS dataset, the pipeline is evaluated for a few-shot learning scenario from limited support samples during training. Thus the proposed pipeline not only showcases the high accuracy that can result from two factors i.e. the quality of embeddings obtained from graph transformers and the metric learning-based meta-learning used by us. The proposed pipeline generalizes well to new classes by learning a metric space where similar samples are close to their corresponding class prototype.

As a possible next step, the findings in this work encourage further exploration and application of these techniques in real-world scenarios with limited labelled data.

References

[1] E. Min, R. Chen, Y. Bian, T. Xu, K. Zhao, W. Huang, P. Zhao, J. Huang, S. Ananiadou, and Y. Rong, "Transformer for graphs: An overview from architecture perspective," *arXiv preprint arXiv:2202.08455*, 2022.

[2] L. Guo, Q. Zhang, and H. Chen, "Unleashing the power of transformer for graphs," *arXiv preprint arXiv:2202.10581*, 2022.

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

[4] H. Zhang and J. Zhang, "Text graph transformer for document classification," in *Conference on empirical methods in natural language processing (EMNLP)*, 2020.

[5] L. Hebert, L. Golab, and R. Cohen, "Predicting hateful discussions on reddit using graph transformer networks and communal context," in *2022 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, pp. 9–17, IEEE, 2022.

[6] H. Matsumoto, S. Yoshida, and M. Muneyasu, "Propagation-based fake news detection using graph neural networks with transformer," in *2021 IEEE 10th Global Conference on Consumer Electronics (GCCE)*, pp. 19–20, IEEE, 2021.

[7] M. Huisman, J. N. Van Rijn, and A. Plaat, "A survey of deep meta-learning," *Artificial Intelligence Review*, vol. 54, no. 6, pp. 4483–4541, 2021.

[8] G. Koch, R. Zemel, R. Salakhutdinov, *et al.*, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, vol. 2, Lille, 2015.

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

[10] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," *Advances in neural information processing systems*, vol. 30, 2017.

[11] K. Ding, J. Wang, J. Li, K. Shu, C. Liu, and H. Liu, "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, 2020.

[12] V. Garcia and J. Bruna, "Few-shot learning with graph neural networks," *arXiv preprint arXiv:1711.04043*, 2017.

[13] F. Zhou, C. Cao, K. Zhang, G. Trajcevski, T. Zhong, and J. Geng, "Meta-gnn: On few-shot node classification in graph meta-learning," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 2357–2360, 2019.

[14] A. J. Bose, A. Jain, P. Molino, and W. L. Hamilton, "Meta-graph: Few shot link prediction via meta learning," *arXiv preprint arXiv:1912.09867*, 2019.

[15] N. Ma, J. Bu, J. Yang, Z. Zhang, C. Yao, Z. Yu, S. Zhou, and X. Yan, "Adaptive-step graph meta-learner for few-shot graph classification," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 1055–1064, 2020.

[16] B. Das *et al.*, "Multi-contextual learning in disinformation research: A review of challenges, approaches, and opportunities," *Online Social Networks and Media*, vol. 34, p. 100247, 2023.

[17] Y. Wang, F. Ma, H. Wang, K. Jha, and J. Gao, "Multimodal emergent fake news detection via meta neural process networks," in *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pp. 3708–3716, 2021.

[18] Y. Dou, K. Shu, C. Xia, P. S. Yu, and L. Sun, "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, 2021.

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