ACTIVE GRAPH ANOMALY DETECTION

BY

ISHIKA AGARWAL

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois Urbana-Champaign, 2024

Urbana, Illinois

Advisor:

Associate Professor Hanghang Tong

ABSTRACT

Recently, detecting anomalies in attributed networks has gained a lot of attention from research communities due to the numerous real-world use cases in the financial, social media, medical, and agricultural domains. This thesis aims to explore node anomaly detection in two different aspects: soft-labeling, and multi-armed bandits. The environment in both settings is constrained to an active learning scenario where there is no direct access to ground truth labels but access to an oracle. This thesis comprises of three works: one using soft-labeling, another with multi-armed bandits, and a third that explores a combination of both. We present experimental results for each work to justify the algorithmic decisions that were made. Future work is also discussed to build on top of these methods.

To all the people who believe in me...

ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Hanghang Tong for his guidance and mentorship. I'm utterly grateful to him for giving me the opportunity to join his esteemed lab. Next, I would like to thank my family for supporting me throughout my academic career. I would not be here today without their encouragement and confidence in me. I would also like to thank my friends for the great memories and keeping me sane.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

Graph data is ubiquitous and can represent complex systems of relationships in diverse domains such as social media [1], citation networks [2], and product reviews [3]. Unfortunately, these networks can contain malicious components whose behavior deviates from that of the general population. These are called anomalies, as opposed to benignities. Graph anomaly detection (GAD) aims to learn a function that can detect anomalous entities in a graph. Detecting such entities is important because anomalies indicate fraudulent activities and untrustworthy information [4]. These entities can be nodes, edges, or subgraphs.

Example of anomalies in graphs. Consider a social network. The nodes represent users and the edges represent connections between the users - for example, a user following another. Bots represent anomalies in this scenario, and we can map the different anomalous entities to a bot's behavior.

Firstly, anomalous nodes would be bot accounts where the attributes differ significantly from those of their followers [5]. A few examples of account attributes are their post content, frequency of posts, and post engagements (likes, reposts, etc.). Next, anomalous edges could represent strange connections that bot accounts have with irrelevant accounts [4]. An example of a strange connection would be between a bot that posts spam advertisements and a user who posts about their academic research. Finally, bot networks could represent anomalous subgraphs [4]. Bot networks are usually heavily dense since each user follows all the other accounts in the network [6]; this simply is not normal with real users, even within a large group of friends.

Problem Statement. Graph anomaly detection is an important task, and one that has been heavily researched so far [7, 8, 9, 10, 11, 12]. Although graph data is prevalent, ground truth labels are hard to acquire. Due to the lack of annotate data, researchers and industry practitioners have turned to using active learning to enable low-resource (specifically, few-shot) model training.

Following this trend, we constrain the environment in this research by assuming we also do not have direct access to ground truth labels. Instead, we have an oracle (a human expert, a teacher model, etc.) that can provide labeled information at a cost. We impose a querying budget, allowing us to leverage active learning techniques to ensure the high informativeness of the queries made to the oracle.

1.1 OUTLINE

In the rest of this chapter, we provide some background on the preliminary concepts used in this thesis. Chapter 2 provides an overview of the related works regarding this area. It also highlights the gap in the previous works that we aim to address.

The next three chapters – 3, 4, and 5 – discuss novel works done towards this thesis, namely, AMUGraph, NeurONAL, and AMUBandits. A brief description of each is provided below. Finally, Chapter 6 contains a brief summary of the thesis, and elaborates on some future works that can be made in this area.

AMUGraph. This method is a GAD framework that uses mixup to enrich the information from the labeled queries obtained through active learning. This framework uses representation learning to quantify uncertainty. The learning process is separate for anomalies and benignities to improve the accuracy. Details are in Chapter 3.

NeurONAL. This method is useful for node classification, which is a broader category of anomaly detection. We use multi-armed bandits with principled exploration and exploitation to make efficient and accurate predictions. Details are in Chapter 4. *This work was accepted into ICLR 2024 [13].*

AMUBandits. This method is a combination of the above two methods. It uses multi-armed bandits with algorithmic details borrowed from the previous works. We explore variants of the algorithm as well. Details are in Chapter 5.

1.2 PRELIMINARIES

1.2.1 Multi-Armed Bandits

Multi-armed bandits (MABs) are powerful agents that can learn patterns in data via reinforcement learning [14]. A common example of an MAB is an agent learning to maximize their reward on slot machines. Given input data about the state of an environment, an MAB can pull one of k arms, after which it will receive a reward.

We can model classification as an MAB problem where each arm is a label in the label set. The reward can be any standard label entropy loss. The replay buffer will contain tuples (X, y) where X is a data sample and y is the true label.

Exploration vs exploitation. One common consideration in multi-armed bandits, and reinforcement learning in general, is balancing exploration and exploitation. Recall that the goal of reinforcement learning agents is to learn a policy π that will accurately map states to actions. During training, the agent receives positive rewards for correct actions. While learning the policy π, the agent could simply choose the action with the highest reward, thereby exploiting the action. However, that might leave many areas of the state-action space unexplored: areas that might have higher rewards [15].

For example, consider a reinforcement learning agent that is learning to recommend content to a user. If the user enjoys cat videos, the agent might exploit that knowledge and continuously recommend cat videos. However, suppose the user enjoys dog videos even more, and generates more engagement with such videos (which translates to a higher reward). If the agent never experiments with and explores the user's interests, it will never find out that the user enjoys dog videos. Hence, while training, an agent must learn to balance exploration and exploitation in order to maximize the rewards and learn an optimal policy.

1.2.2 Active Learning

Labeled data can be quite difficult and expensive to obtain. Instead of labeling the entire dataset, active learning involves an oracle (human expert-in-the-loop or a teacher model) that provides labels to given data. In order to avoid abusing the oracle by querying for the label of every data point, a query budget is imposed. The goal of active learning is to learn a model that achieves comparable performance using only a subset of data, as opposed to using the entire labeled dataset [16]. We can do this by calculating the uncertainty of a prediction to maximize the information gained by the labeled data.

Active learning algorithm design usually answers the following three questions[1]:

1. How does one calculate the uncertainty of a prediction?
2. How does one query for a label? Is the oracle a teacher model, or a human expert? Are the labels hard or soft?
3. Once the labels are acquired, how does one maximize the information learned by the queried data?

There are three flavors of active learning sampling: membership query synthesis, stream-based, and pool-based [16]. In membership query synthesis, the model can generate queries that potentially yield the maximum amount of information. Unfortunately, this is not always

[1]Since they use active learning, all methodologies in this thesis will answer these three questions.

feasible for human annotators since the generated queries are sometimes indecipherable. For example, [17] used query synthesis to ask humans to annotate handwritten digits, but they found the generated digits were not recognizable. For such reasons, stream-based and pool-based sampling are more favorable since they use the available data, which ensures interpretability.

In stream-based active learning, the model is provided with data as a stream. The model needs to decide on-the-fly whether it wants to query for the label of a certain data point. On the other hand, in pool-based active learning, the model assumes access to all the data beforehand. This way, the model can choose a small subset of data points on which to query. Usually, stream-based learners have a larger query budget than pool-based learners.

1.2.3 Mixup

Mixup is a popular strategy originated in computer vision, used to augment data for classification models. It generates new images by linearly interpolating existing images, i.e., taking a weighted sum of two images and their labels [18]. Consider two data instances (image, label): (x_i, y_i) and (x_j, y_j). Mixup creates a new data instance (x_k, y_k) using the below formulation:

$$x_k = \lambda \cdot x_i + (1 - \lambda) \cdot x_j \tag{1.1}$$

$$y_k = \lambda \cdot y_i + (1 - \lambda) \cdot y_j \tag{1.2}$$

where λ is a parameter in $(0, 1)$.

Mixup is used to make learning more robust and generalizable. It works particularly well with soft labels because we can replace λ and $(1 - \lambda)$ with the soft labels. Using mixup with soft labels is a common technique in a variety of domains such as image anomaly detection [19] and intent classification in dialogue systems [20]. Soft labels not only express uncertainty, but are useful to teach the model to be uncertain for wrong predictions and reduce the chance of overconfident predictions. Mixup also helps to refine the decision boundaries, which, in turn, improves prediction accuracy.

CHAPTER 2: LITERATURE SURVEY

This chapter provides a background of the current literature in the surrounding areas. We point out many works and the key differences and limitations when compared to our work. We also identify the gap that our research aims to bridge.

2.1 GRAPH REPRESENTATION LEARNING

There are many works that perform representation learning on graphs for anomaly detection. Representation learning can be useful for anomaly detection as it provides a classification framework based on the latent space representation [21, 22]: specifically, the reconstruction loss [23].

While [21] introduces a new algorithm, it builds upon a well-established concept of constructing hyperspheres in the latent representation to detect anomalies. Embeddings of benign nodes are pushed into the hypersphere while embeddings of anomalies are pushed away from the hypersphere. As a result, when performing inference on such a model, a node is classified as an anomaly if the embedding is outside of the hypersphere.

Works like [22, 24, 25] learn a classifier on top of the latent space. These methods optimize the reconstruction and the classifier at the same time. The joint optimization enables a stronger and robust learning process. In addition, this method is quite popular with domain-specific applications – dementia diagnosis on neuroimages [26], anomaly detection in industrial inspection [27], pest and disease detection in crops [28], and many more. Essentially, the representational module will act as the feature extractor and the classification module will perform the task.

Another common technique in representation learning is categorizing anomalies based on the reconstruction loss [23]. Since anomalies deviate from the normal behavior, it is expected that models cannot learn the distribution or structure of anomalies well enough while reconstructing them. Therefore, we can use the reconstruction loss as a score to detect anomalies. Nodes with high reconstruction loss are considered anomalous, while nodes with low reconstruction loss are considered benign. An important work following this notion, and one we borrow ideas from, uses label information to train a variational autoencoder [29], a traditionally unsupervised model. They also perform anomaly detection, but in tabular time series applications, where they modify the ELBO training objective based on whether the node is an anomaly or benignity. As mentioned later on in Chapter 3, because there are only a few anomalous data points, using a standard training objective will result in a model that knows how to reconstruct an anomaly, which reduces the reconstruction loss and the ability to detect anomalies. Having separate training objectives for anomalies and benignities enhances and improves detection, as our experiments also show.

2.2 REINFORCEMENT LEARNING

Recent works have demonstrated the power of reinforcement learning and multi-armed bandits for graphs [30] and even for anomaly detection (on graph and tabular data) [10, 11]. Reinforcement learning is able to combine the representational power of deep learning with the inherent decision-making framework [31].

Among the taxonomy of reinforcement learning, actor-critic is a framework that consists of two networks: an actor that makes decisions and a critic that criticizes the decisions. This is not unlike the roles of the generator and discriminator in a Generative Adversarial Network (GAN) [32]. [33] employs the actor-critic framework to detect anomalies in wireless sensor networks. They train the agent to choose one of the N sensors that might be anomalous, and assign a reward proportional to how high the confidence score is. The actor is used to explore a policy to map states to actions, and the critic is used to estimate the value of a state. We use a similar principle in the formulation of the method laid out in Chapter 4.

Specifically, since the task of anomaly detection does not have any state transitions or a diverse action space, anomalies can be detected using a multi-armed bandit. [30] uses a multi-armed bandit to learn an optimal policy to maximize rewards on graph data. They experiment with a toy example of a drone that aims to provide internet access to a rural area. The drone receives rewards proportional to the amount of communication traffic the drone is able to handle in a certain area. In this scenario, the drone does not have access to the reward distribution and therefore, must use guided exploration and exploitation to maximize rewards.

[11] uses reinforcement learning to jointly learn to detect anomalies with a partially labeled dataset. In their semi-supervised setting, the agent samples a new point on which to classify and train. The agent receives an intrinsic reward if it identifies data points that are abnormal (can be determined by a weak learning - iForest [34], for example). It will also receive an extrinsic reward if a correct classification is made.

[10] also uses multi-armed bandits and human-machine collaboration to rank the abnormality of nodes in graphs. Using clustering techniques to group nodes, the bandit is tasked to choose a cluster that has suspicious nodes, query the human expert on one of the nodes, and learn from the feedback. This process is repeated until a certain query budget is exhausted.

In these works, it is clear that multi-armed bandits are very useful to balance exploration and exploitation. The common setting in these works is that they are off-policy, which means they learn from past experience. Off-policy bandits can be expensive to learn because of the costs to constantly train on the replay buffer. An anomaly detection algorithm could benefit from learning from a few data points, where the size of the replay buffer is controlled.

2.3 ACTIVE LEARNING

Works such as [23] and [11] assume partially labeled data. While this setting is quite practical, we can obtain higher quality data with the use of human-machine collaboration in the form of active learning. Active learning has become an area of interest for researchers because of the practice implications and ease of implementation. It is much easier for machine learning model developers to use human expertise to provide a few labels than to collect labeled data.

Usually, unsupervised anomaly detection methods make certain assumptions on the anomaly distribution - active learning can alleviate this [25]. In this particular work, they develop a model-agnostic neural network layer, called the UAI layer, that can be used to output an anomaly score given some latent representation. In batches, they rank and select a few data points to query to improve the original model. In this thesis, two of our works use a similar principle of performing classification on the latent space representation.

As mentioned before, [10] uses active learning to alleviate the cost of requiring labeled data. Here, pool-based active learning is employed and the human labeler provides hard labels.

[35] employs active learning and multi-armed bandits to find erroneous triples in knowledge graphs. Using some off-the-shelf anomaly detectors, they identify a few suspicious triples, query a human expert, and assign a reward based on the correct identification of an anomalous triple. They assign a hard reward of 0 or 1, which can be useful for training, but can be hard to distinguish the uncertainty of two data points. Rewards assigned in a range of [0, 1] are useful for ranking the uncertainty of data points to ensure the queries are most informative.

In [29], a variational autoencoder-based network is employed. Therefore, they define uncertainty based on the evidence lower bound (ELBO) of a certain data point. Recalling that the ELBO contains a component for the reconstruction loss, they find that benign points have a low ELBO ($< 5e - 3$) while anomalous points have a large ELBO (> 20). In their model SemiVAE, they formulate a querying strategy based on this finding. We borrow the concept of the SemiVAE model architecture.

While it makes anomaly detection algorithms much more practical with industry requirements and scale, active learning algorithms do not fully utilize the rich information of labeled data. Data augmentation strategies can be used to take full advantage of the valuable labels.

2.4 MIXUP

Due to the limited amount of data present in active learning algorithms, there is a risk of overfitting to the data, especially with deep learning models [36]. To address, this, either we can acquire more data, which is contradicts our goals, or we can artificially create more data. To ensure our artificial data "looks like"[2] our current data, we can use data augmentation strategies like mixup. It is widely used in the field of computer vision, but can also be applicable to other forms of data, such as tabular data and graphs.

Particularly, [37] uses mixup for graph data augmentation. They map nodes from one input graph to the nodes of another input graph, align the two graphs based on their node features and network topology, and mix the graph to generate a new one with soft edges (i.e., weighted edges). In the context of node classification, works like [38] mix the node features and the network topology. This enhances predictions by expanding the training distribution into areas that are reachable by linear interpolation – the core principle of mixup – from areas in which the current data lies [18]. Since graphs are information-rich, mixup can result in a high volume of informative data without overfitting.

In the context of anomaly detection, researchers have found that such tasks can be improved upon with mixup. For example, [39] uses mixup, for graph anomaly detection, to augment anomalies by mixing known anomalies with high-confidence unlabeled anomalies. They use the Euclidean distance between node embeddings to determine confidence. [40] performs image anomaly detection by learning a model to reject out-of-distribution samples. They mix in-distribution images with each other to learn a model.

In the specialized field of graph anomaly detection and data augmentation, a very recent work on learnable data augmentation [41] has shown that adding a small amount of noise to the existing homophilic graph data[3] can improve anomaly detection performance by 3-5%. They argue that using data augmentation is beneficial in low-resource settings, which fits this thesis' setting of active learning.

Moreover, an implicit benefit of mixup is its ease of use with soft labeling. In practice, hard labels are harder to obtain because they require an expert to determine [42], whereas soft labels can be acquired from knowledgeable, but not expert sources. [43] use mixup

[2]The new data should not be out-of-distribution with respect to the current data.

[3]Graphs that exhibit homophily, i.e., the tendency of edges to connect similar nodes.

and soft labeling for text classification. They augment data samples by mixing two random sets of text embeddings and their corresponding soft labels. They find that soft labels have inherent uncertainty, which mixup can alleviate, thereby improving the classification.

In summary, mixup has shown good performance when applied to graphs, anomaly detection, and with soft labeling. The primary advantage of using mixup is to improve the generalizability and robustness of the model. [44] shows that the classification decision-making boundaries are smoother, because it avoids overfitting to a small set of data, which boosts the robustness against neural network attacks.

This thesis explores the narrow but interesting intersection among these works: performing anomaly detection on graphs, particularly on graph nodes, with active multi-armed bandits that augment data using mixup.

CHAPTER 3: ACTIVE GAD USING MIXUP

In this chapter, we describe **A**ctive **M**ix **U**p for **Graph** Anomaly Detection (AMUGraph), an active graph anomaly detection algorithm. Graph data is prevalent, but ground truth labels are hard to acquire. Hence, active learning can be used to obtain soft labels for efficient supervised learning. Active learning assumes no initial access to labeled data but relies on access to an oracle that can provide ground truth data. In order to avoid overuse of the oracle, we impose a small query budget.

Here, the oracle represents a human expert. Because of the data complexity[4], it can be infeasible to expect human labelers to be fully confident in their predictions. Therefore, we assume the oracle will provide soft labels. Intuitively, a soft label of $(0.3, 0.7)$ means the oracle is 70% confident the node is anomalous. We employ pool-based active learning where in each of the R rounds, we query r points.

To make the labeled data more informative, we can use data augmentation strategies to simulate more labeled data. Hence, we use mixup, a popular computer vision technique.

3.1 PROBLEM FORMULATION

Input: (1) a graph $G = (V, E, X)$ with nodes V - with attributes X - and edges E, (2) an oracle $\mathbb{O}$ that provides soft labels.

Goal: to learn a function $f(x; \theta)$ that outputs a label {benign, anomalous} for a given node x, where θ represents the function's parameters.

To determine which nodes to query, we use the normalized reconstruction loss, L2 norm, between the original and reconstructed node embeddings. We assume that nodes with a reconstruction loss near 0.5 exhibit high uncertainty. Therefore, the model randomly chooses r points with reconstruction losses in the range of $[0.4, 0.8]$ for training[5].

3.2 METHODOLOGY

Figure 3.1 shows the model training architecture for AMUGraph. The key components of the AMUGraph framework are explained as follows:

SemiVAE. In anomaly detection, as previously mentioned, a common technique is to measure "anomalousness" by using the reconstruction loss. Anomalies are expected to have

[4]Graph data can be complex to interpret, requiring an expert to be able to label.

[5]This range was treated as a hyperparameter (see Section 3.3.5 for details on hyperparameter tuning).

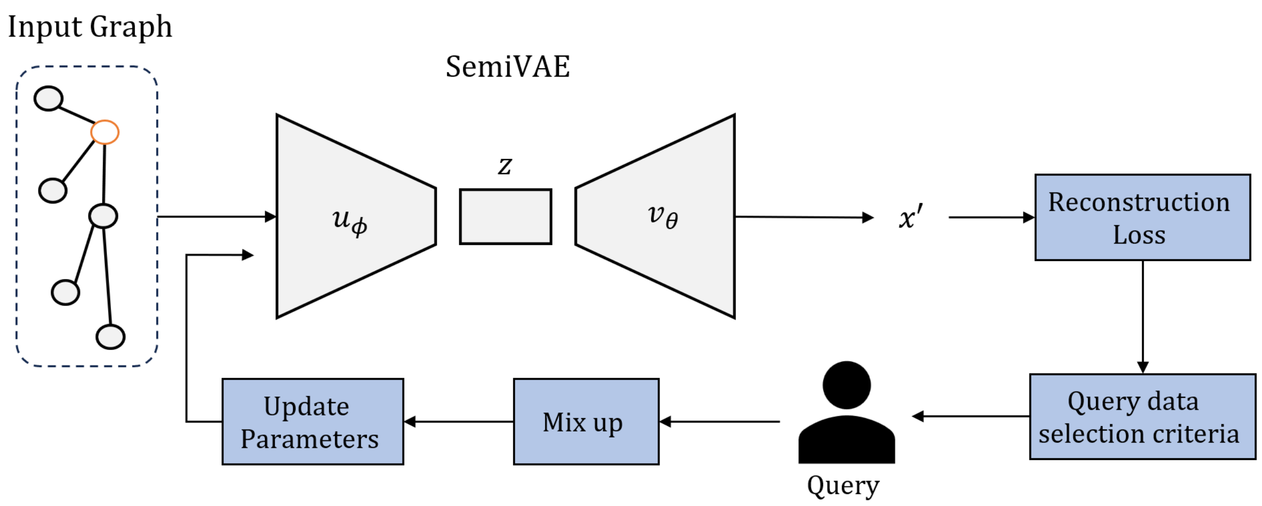

Figure 3.1: The model training architecture for AMUGraph. The orange node is anomalous. Here, z represents the latent space, and x' represents the reconstructed node x.

higher reconstruction losses. However, if a model is trained to reconstruct anomalies, it might get better at reducing the reconstruction loss over time [29]. Therefore, to ensure the reconstruction loss is high for anomalies, we must separate the learning of benign nodes and anomalous nodes.

Hence, we borrow the concept of the SemiVAE [29] - the difference from the standard VAE lies in the loss function. The loss function of a VAE has two components: the reconstruction loss and the KL-divergence. [29] splits the training set into benign points and anomalies. For benign points, they train the network with the original VAE loss function. For anomalies, however, they train the network to only maximize the reconstruction loss.

Below is the mathematical formulation. u is an encoder neural network with parameters ϕ. It takes in input x and outputs a hidden representation z. v is a decoder neural network with parameters θ. It takes in input z and outputs a reconstruction.

$$L(x) = \begin{cases} E_{u_\phi}[\log(v_\theta(x|z))] - KL(u_\phi(z|x)||v_\theta(z)) & \text{if } x \text{ is benign} \\ -E_{u_\phi}[\log(v_\theta(x|z))] & \text{if } x \text{ is anomalous} \end{cases} \tag{3.1}$$

We use the same principle in our architecture because it allows us to create a scoring function based on the reconstruction loss. If the reconstruction loss is low, the point is classified as benign; if the reconstruction loss is high, the point is classified as anomalous.

Querying and Mixup. The oracle represents a human expert. As mentioned before, the oracle will provide soft labels. Due to the nature of soft labels, each node can be thought to have a benign component and an anomalous component. This structure can be particularly useful in mixup. Consider two nodes x_p and x_q with soft labels (p_b, p_a) and (q_b, q_a), respectively[6]. To mix two nodes, we calculate a weighted sum of the anomalous component of one node and the benign component of another node. Using mathematical notation, we calculate the new embedding and the soft label as[7]:

$$x_{new} = p_b \cdot x_p + q_a \cdot x_q \tag{3.2}$$

$$y_{new} = (p_b, q_a) \tag{3.3}$$

Overall, we employ this strategy to augment every pair of queried nodes to obtain a larger training set. A node x_p is anomalous if $p_a > p_b$ and benign otherwise. All of the new augmented data are added to the training set and are used for training, including forward and backpropagation.

Classification. For the final classification, we use the latent representation (we treat the latent dimension size as a hyperparameter). First, we use the soft labels to split the points into anomalies and benign nodes. Again, a point is anomalous if the anomalous component has a larger confidence than the benign component and benign otherwise.

Next, we treat the values in each dimension as a distribution (see Figure 3.2). For a new testing node w, the latent representation is extracted. Then, for each dimension, we calculate the probability density function (PDF) of the value in that dimension according to both the distributions (anomalous and benign). We average the probabilities across the dimensions and retrieve a new soft label (w_b, w_a). If $w_a > w_b$, the node w is classified as anomalous and benign otherwise.

3.2.1 Active Learning.

Here, we answer the three questions relevant to active learning algorithm design outlined in Section 1.2.2. The uncertainty of a prediction is calculated using the reconstruction loss. A query is made to a human labeler who will return a soft label. Once the labels are acquired, data augmentation will maximize the information inherent in the queried data.

[6]The benign component is denoted with the subscript b and the anomalous component with subscript a.

[7]We have empirically tested that there is no difference between $p_b \cdot x_p + q_a \cdot x_q$ and $p_a \cdot x_p + q_b \cdot x_q$, which is expected.

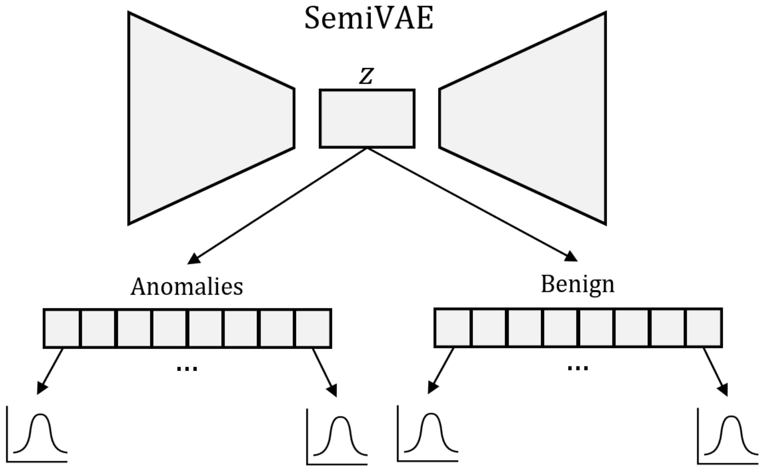

Figure 3.2: Creating distributions for the latent representations for classification.

3.3 EMPIRICAL RESULTS

3.3.1 Datasets

In this thesis, we use two datasets: Pubmed [45] and Yelp [46]. The Pubmed graph dataset is a citation network dataset where nodes are publications and edges indicate citation relationships. This is a node classification dataset where the goal is to predict the category of a certain publication. The Yelp dataset (or Fraud Yelp Dataset) contains restaurant and hotel reviews where nodes are reviews and edges connect reviews to users, reviews to products, and reviews to reviews. This is another node classification dataset where the goal is to identify fraudulent reviews. Table 3.1 show the statistics of these datasets.

	Pubmed	Yelp
# Nodes	19,717	45,954
# Edges	88,651	8,051,348
% Training	74.9	74.9
% Anomalous	20.81	14.53

Table 3.1: Statistics of the Pubmed and Yelp graph datasets.

3.3.2 Experimental Setup

Since Pubmed does not contain anomalous data, we randomly treat one the classes as anomalous, and the other classes as benign. We run our training algorithm for 5 rounds with 0.69% labels for each round (total of 3.5% of the dataset). Because we do not have access to a human labeler, we train a Graph Convolutional Network (GCN) [47] on the

	Pubmed	Yelp
DOMINANT	60.6	57.4
GDN	61.7	67.8
AMUGraph-SVAE	50.7	51.9
AMUGraph-NoAugment	52.3	51.9
AMUGraph-Random	54.7	65.7
AMUGraph	**74.1**	**72.6**

Table 3.2: ROC-AUC (%) on the selected datasets and baselines compared to AMUGraph. Higher is better.

training set of each dataset. We train the GCN for 40 epochs with a learning rate of 1e-3. To gauge performance, we use the ROC-AUC score.

3.3.3 Baselines

We use a few baselines. Firstly, we use the Graph Deviation Network (GDN) from [12]. It uses few-shot learning to encode node representations and learn an abnormality scoring function. Secondly, we compare our method with [23]'s DOMINANT method that works on the latent representations from an autoencoder. They first encode the graph structure with the attributes. Then, they split the decoding into a structural reconstruction and an attribute reconstruction. Finally, they compute a score over the reconstructions which is then ranked to find anomalies.

Finally, we report the performance of our method without the mixup augmentation strategy (called AMUGraph-NoAugment), without the active learning strategy - where we choose a random subset of points on which to query - (called AMUGraph-Random), and without either (called AMUGraph-SVAE).

3.3.4 Results

Table 3.2 shows the results for our experiments. As shown, our method is competitive against both baselines and shows improvement in ROC-AUC.

The objective of the SemiVAE allows the model to learn to reconstruct benign points, guiding this towards a one-class classification problem, which is an easier task in representation learning. Also, using active learning querying strategies helps to identify points that provide the most information, which improves the learning process. Furthermore, mixup works well with active learning, because mixup will create points that are close to the decision boundaries, ultimately enriching the information provided by the querying.

3.3.5 Hyperparameter Tuning

Here, we provide a few details on the hyperparameter tuning. We provide details on the human labeler simulator, the baselines, and the various parameters used in AMUGraph, including the number of rounds, the latent dimension size, and the query uncertainty criteria. While most of these studies are performed on the Yelp dataset, we find that the trends observed here are consistent on the Pubmed dataset as well.

Human labeler simulator. As mentioned before, we simulate a human labeler by training a GCN on the data. We use the exact same training and testing split across the GCN and AMUGraph to ensure there is no label leakage. We train the GCN by performing a grid search across {20, 40, 60, 80, 100} for epochs and {1e-3, 2e-3, 5e-3, 1e-2, 2e-2} for the learning rate.

Baseline hyperparameters. For the baselines DOMINANT and GDN, we use the hyperparameters provided in the source code found on Github. We find that the performance of both algorithms match the performance in the original papers.

AMUGraph hyperparameter studies. In Section 3.3.2, we mention that we run our algorithm for 5 rounds with a query budget 0.69% and that our querying criteria was a reconstruction loss in the range of $[0.4, 0.8]$. Additionally, we use a latent dimension size of 32. Below, we provide empirical results from the hyperparameter tuning we performed.

Table 3.3 demonstrates the performance with varying round-query budget values for the Yelp dataset. The first row indicates AMUGraph achieves an ROC-AUC of 72.6% with 5 rounds where each round has a query budget of 103 data points. It seems that with fewer rounds and more query budget in each round, the performance improves. This could be due to the larger training set in the settings with fewer rounds, which allows for more robust learning.

Rounds	Query Budget (%)	ROC-AUC(%)
5	103 (0.69)	72.6
10	51 (0.35)	70.5
16	32 (0.21)	68.3
20	25 (0.17)	67.1

Table 3.3: ROC-AUC wrt varying round-query budget values on the **Yelp** dataset. Note that each round-query budget pair adds up to 3.5% of the entire dataset.

Table 3.4 demonstrates the performance across query uncertainty criteria, with varying lower and upper threshold values, for the Yelp dataset. AMUGraph seems to suffer with thresholds that are too large (i.e., $[0.2, 0.8]$) and too small (i.e., $[0.2, 0.4]$, $[0.4, 0.6]$, and $[0.6, 0.8]$). Larger thresholds might include points with high certainty, therefore not gain much information during querying. Next, while it is our assumption that uncertainty results in reconstruction losses near 0.5, that might not be true for all cases. Therefore, smaller thresholds, even if they are near 0.5, would not be flexible enough to fully capture uncertainty. For these reasons, $[0.4, 0.8]$ seems like a good compromise.

	0.2	0.4	0.6
0.4	67.3	x	x
0.6	67.1	65.1	x
0.8	66.4	72.6	64.0

Table 3.4: ROC-AUC wrt varying lower and upper threshold values on the **Yelp** dataset. The column labels indicate the lower threshold while the rows indicate the upper threshold.

Table 3.5 demonstrates the performance with varying latent dimension sizes on both datasets. There does not seem to be a clear reason why a latent dimension size of 32 works the best with both datasets. However, one can notice a stark performance improvement with a latent dimension size of 32 compared to that of 16 or 64.

	Pubmed	Yelp
8	52.2	65.4
16	66.6	57.9
32	74.1	72.6
64	51.6	67.5

Table 3.5: ROC-ARC wrt varying latent dimension sizes.

3.3.6 Unsupervised Methods

To put this method in perspective and make the emprical results more comprehensive, we add a short evaluation involving an unsupervised methods. In the below table, we compare our method with the unsupervised method AnomalyDAE [48]. We find that AnomalyDAE performs slightly better than our method. Still, the settings of the two problems are different in which our work aims to generate good soft labels for anomaly detection.

	Pubmed	Yelp
DOMINANT	60.6	57.4
GDN	61.7	67.8
AMUGraph-SVAE	50.7	51.9
AMUGraph-NoAugment	52.3	51.9
AMUGraph-Random	54.7	65.7
AnomalyDAE	**74.3**	**74.0**
AMUGraph	74.1	72.6

Table 3.6: ROC-AUC (%) on the selected datasets and baselines (in particular, compared with an unsupervised method). Higher is better.

3.4 SUMMARY

In this chapter, we described a method that uses mixup to augment and enhance labeled data that was received by an oracle using an active learning training algorithm. As a downstream task, this framework was applied on node anomaly detection in graphs. We also borrowed a label-specific representation learning strategy where only benign data points are learned to be represented and reconstructed. Through evaluation, we were able to show how our method outperforms a few seminal algorithms regarding node anomaly detection by a 4.8-12.4% performance improvement, and how the performance varies with respect to different components of our algorithm.

CHAPTER 4: ACTIVE MULTI-ARMED BANDITS FOR CLASSIFICATION

In this chapter, we describe a work that was accepted to ICLR 2024 [13] called **Neur**al **On**line **A**ctive **L**earning (NeurONAL). We solve multi-class classification using active multi-armed bandits. We provide two new efficient learning algorithms using stream-based and pool-based active learning, respectively.

In stream-based active learning, the agent receives data one point at a time and needs to decide on-the-fly whether to query it. Alternatively, in pool-based active learning, the agent receives all the data at once and in each round, can query a pool of data points. In either setting, a small query budget is imposed and queries are made until the budget is exhausted. The goal of active learning is to exploit the knowledge from labeled data points and explore new knowledge from unlabeled data.

Recent research has shown that multi-armed bandits are good agents for using active learning as they achieve good theoretical and empirical performance. We use neural networks to represent and balance exploration and exploitation. The exploitation network aims to improve the classification while the exploration network aims to model the uncertainty in the exploitation network.

In previous works [49, 50], in an effort to turn k-classification into a bandit problem, the data instance x was transformed into a long vector that is k times the original data size:

$$x_{t,1} = [x_t^\top, \mathbf{0}^\top, \cdots, \mathbf{0}^\top], x_{t,2} = [\mathbf{0}^\top, x_t^\top, \cdots, \mathbf{0}^\top], x_{t,k} = [\mathbf{0}^\top, \mathbf{0}^\top, \cdots, x_t^\top] \tag{4.1}$$

This is an effective method, but if k is large, it can be quite costly and cannot scale well. See Figure 4.2 for a depiction of such a method. Setting aside the increased memory requirements, calculating class scores for each class requires one forward pass through the

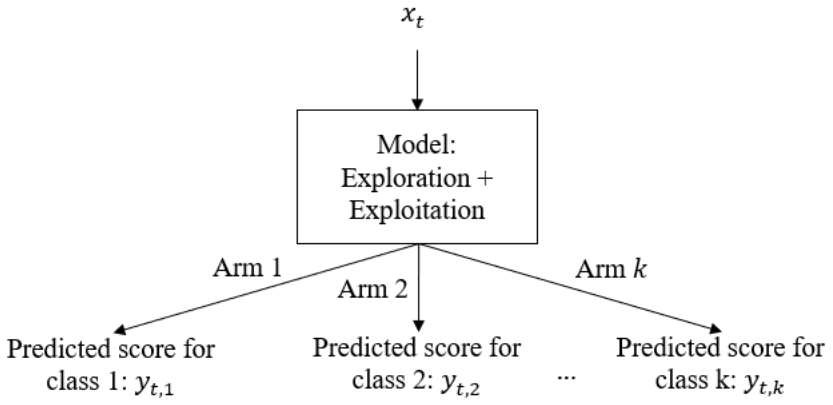
x_t
Model: Exploration + Exploitation
Arm 1
Arm 2
Arm k
Predicted score for class 1: $y_{t,1}$
Predicted score for class 2: $y_{t,2}$
...
Predicted score for class k: $y_{t,k}$

Figure 4.1: Synchronous, and more efficient, nature of our method, NeurONAL.

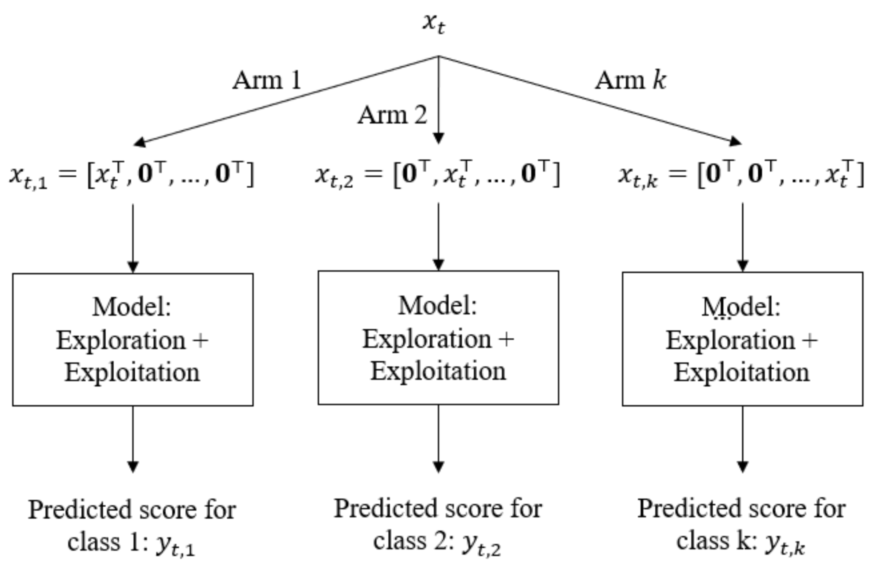
x_t
Arm 1
Arm 2
Arm k
$x_{t,1} = [x_t^\top, \mathbf{0}^\top, \ldots, \mathbf{0}^\top]$
$x_{t,2} = [\mathbf{0}^\top, x_t^\top, \ldots, \mathbf{0}^\top]$
$x_{t,k} = [\mathbf{0}^\top, \mathbf{0}^\top, \ldots, x_t^\top]$
Model: Exploration + Exploitation
Model: Exploration + Exploitation
Model: Exploration + Exploitation
Predicted score for class 1: $y_{t,1}$
Predicted score for class 2: $y_{t,2}$
Predicted score for class k: $y_{t,k}$

Figure 4.2: Sequential, and sub-optimal, nature of previous bandit-based algorithms.

model.

Thus, we propose two new algorithms that balance exploration and exploitation to make learning bandits more efficient. Figure 4.1 illustrates our method. Comparing it to Figure 4.2, we can see that the efficiency of NeurONAL comes from the ability to calculate all k class scores with one pass of the model versus making multiple passes through the model for class scores.

4.1 PROBLEM FORMULATION

Input: (1) data $X \in \mathbb{R}^d$ with corresponding labels in the space $\{0,1\}^k$ where k is the length of the label set, (2) T rounds, and (3) oracle $\mathbb{O}$.

Goal: to learn a bandit that learns to assign a given data instance x a label from the label set $\{1, \cdots, k\}$.

4.2 METHODOLOGY

Figure 4.3 illustrates the architecture of the neural networks used for exploitation and exploration.

The exploitation network, f_1, learns to classify data instances and receives rewards proportional to the correctness of the network's prediction. On the other hand, the exploration network, f_2, learns to model the uncertainty of the exploitation network. It does so by

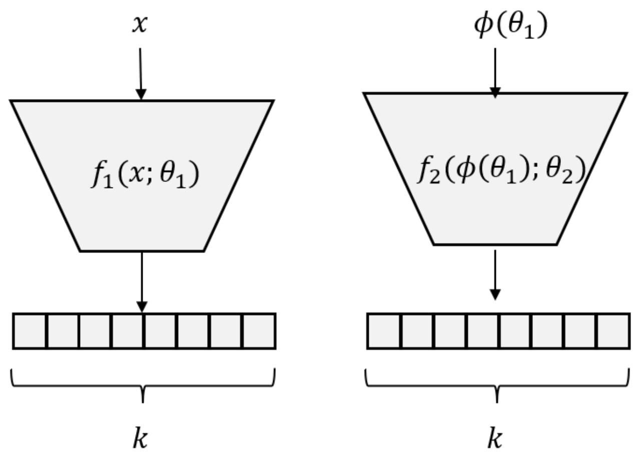

Figure 4.3: The two neural networks for exploitation (left) and exploration (right) used in NeurONAL.

receiving as input an embedding of f_1's parameters, denoted by $\phi(\theta_1)$ in the figure. The exploration network receives rewards proportional to the uncertainty in f_1's prediction. The rewards are calculated using the information gained from querying the oracle.

Training. The stream-based and pool-based settings use the same backbone algorithm for training [13]:

1. A data instance is randomly sampled from the dataset.
2. The data instance is fed into f_1, where the network partial derivative of f_1 is fed into f_2, resulting in two k-dimensional vectors. These vectors represent a probability distribution over the label space.
3. By adding them, we obtain a final distribution in which we can calculate the top-2 Bayes optimal classes.
4. If the difference in confidence between these two classes is small[8], the corresponding data instance will be queried.
5. From the queried information, the training sets for f_1 and f_2 will be constructed, and
6. A batch gradient descent will be performed to update the parameters of both networks.

These steps are repeated until the query budget is exhausted.

[8]This is determined by a hyperparameter. In this work, we query if the difference is smaller than 0.6.

Streaming versus pooling. In the streaming setting, the algorithm encounters each data instance once. Using the criteria mentioned above in step (4), the learner will decide on-the-fly whether to query an instance, and the parameters of f_1 and f_2 will be updated as soon as the data instance is queried. In the pooling setting, the algorithm will encounter each data instance multiple times. Using the same criteria, the learner will look over all the data instances, and choose a pool of instances to query. After each round[9], the pool will be labeled and the network parameters will be updated.

4.2.1 Active Learning

Here, we answer the three questions relevant to active learning algorithm design outlined in Section 1.2.2. A prediction is uncertain if the difference in confidence between the first- and second-most Bayes optimal class is very small (see Step 4 of the training procedure). Queries are answered by a human expert that provides a hard label. Once the labels are acquired, a higher reward is given to the exploitation network for learning the correct class label and to the exploration network for reducing the uncertainty in predicting the correct class label. The joint cooperation between the exploration and exploitation networks is how information is maximized from the queried data.

4.3 EMPIRICAL RESULTS

4.3.1 Datasets

We extensively evaluate NeurONAL in the stream-based (NeurONAL-S) and pool-based (NeurONAL-P) settings on six public, tabular classification datasets: Adult, Covertype (CT), MagicTelescope (MT), Shuttle, Fashion, and Letter. Table 4.1 displays the statistics of the datasets. We use 10,000 instances for training and another 10,000 instances for testing (except MT, where we use 9,000 for testing).

4.3.2 Baselines

Streaming. We use three baselines: (1) NeurAL-NTK [49], (2) I-NeurAL [50], and (3) ALPS [51]. Given an instance, NeurAL-NTK is a method that predicts k scores for k classes

[9]In this work, we use 10 rounds. From the previous work, we see that 10 rounds with 100 queries per round is a standard limit. Since we cap training datasets to 10,000 points, we are querying for 10% of the dataset.

	# Instances	# Features	k
Adult	48,842	105	2
CT	581,012	98	7
MT	19,020	10	2
Shuttle	58,000	9	7
Fashion	70,000	784	10
Letter	124,800	784	2

Table 4.1: Statistics of the datasets used to evaluate NeurONAL.

sequentially, only based on the exploitation network classifier with an Upper-Confidence-Bound (UCB). On the contrary, I-NeurAL predicts k scores for k classes sequentially, based on both the exploitation and exploration network classifiers. NeurAL-NTK and I-NeurAL query for a label when the model cannot differentiate the Bayes-optimal class from other classes. Finally, ALPS makes a prediction by choosing a hypothesis (from a set of pre-trained hypotheses) that minimizes the loss of labeled and pseudo-labeled data, and queries based on the difference between hypotheses.

Pooling. Here, we use four baselines: (1) CoreSet [52], (2) BADGE [53], (3) DynamicAL [54], and (4) ALBL [55]. Coreset is an approach that aims to create a set cover of data points that can be used to effectively train a model. Points are selected by optimizing the generalization error over the dataset, the training error, and the empirical loss between the labeled and unlabeled data points. BADGE uses the k-means++ algorithm to select points based on the predicted label and the gradient embedding of the network. DynamicAL creates a query set by comparing the change in training dynamics over the unlabeled dataset and chooses the points with larger changes. Finally, ALBL is a method that learns a number of models and iteratively selects models to improve based on their past performances.

4.3.3 NeurONAL-S Results

To gauge performance, we use testing accuracy and running time. The results are compiled in Table 4.2. As shown, NeurONAL-S consistently outperforms the baselines on all datasets in terms of testing accuracy and running time. This is because traditional bandit-based approaches calculate scores sequentially, while our neural network approach calculates scores synchronously.

Figure 4.4 also compares the cumulative training regret at each round of NeurONAL-S compared to the baselines. Regret is calculated by a mistake in a prediction; it is a binary value of 0 (incorrect) or 1 (correct). As shown, NeurONAL-S' regret climbs the slowest compared to most baselines. Intuitively, the training regret for NeurONAL-S is slow growing because of the balanced exploration and exploitation.

	Adult	MT	Letter	Covertype	Shuttle	Fashion
	Accuracy					
NeurAL-NTK	80.3 ± 0.12	76.9 ± 0.15	79.3 ± 0.21	61.9 ± 0.08	95.3 ± 0.20	64.5 ± 0.16
I-NeurAL	84.2 ± 0.22	79.4 ± 0.16	82.9 ± 0.06	65.2 ± 0.19	99.3 ± 0.12	73.5 ± 0.28
ALPS	75.6 ± 0.19	35.9 ± 0.76	73.0 ± 0.41	36.2 ± 0.25	78.5 ± 0.21	74.3 ± 0.01
NeurONAL-S	**84.8 ± 0.51**	**83.7 ± 0.17**	**86.5 ± 0.16**	**74.4 ± 0.19**	**99.5 ± 0.09**	**83.2 ± 0.38**
	Running Time					
NeurAL-NTK	163.2 ± 1.31	259.4 ± 2.48	134.0 ± 3.44	461.2 ± 1.26	384.7 ± 1.86	1819.4 ± 10.84
I-NeurAL	102.4 ± 7.53	46.2 ± 5.58	232.2 ± 3.80	1051.7 ± 5.85	503.1 ± 9.66	1712.7 ± 12.8
ALPS	403.27 ± 4.99	801.51 ± 7.31	239.49 ± 4.10	720.54 ± 5.50	780.08 ± 7.52	839.94 ± 3.13
NeurONAL-S	**54.7 ± 3.21**	**10.5 ± 0.39**	**92.1 ± 3.4**	**166.4 ± 0.59**	**101.2 ± 2.32**	**116.3 ± 3.39**

Table 4.2: Test accuracy and running time compared to bandit-based methods in **stream-based** setting.

4.3.4 NeurONAL-P Results

Similar to the streaming setting, we use testing accuracy and running time to measure the performance. Table 4.3 shows the results. As shown, NeurONAL-P also consistently outperforms the baselines on all datasets in term of testing accuracy. This is because the baselines that do not have explicit exploration tend to be trapped in sub-optimal solutions. In most cases, NeurONAL-P experiences a significant time speed-up.

Figure 4.5 compares the testing accuracy at each query round for NeurONAL-P and the baselines. In most cases, NeurONAL-P has a higher testing accuracy across the query rounds. However, for a few datasets - such as MagicTelescope and Letter - Coreset has a better testing accuracy in the early query rounds. Still, the final testing accuracy at 10 rounds for NeurONAL-P is the highest compared to the baselines.

4.3.5 Parameter Studies

Here, we perform ablation studies for a few important parameters. In particular, we perform ablation studies on μ and γ for NeurONAL-P and on the label budget percentage for NeurONAL-S. We do not perform a label budget ablation on NeurONAL-P because the rounds and budget per round is fixed by previous work to ensure comparability.

μ and γ. Table **??** shows NeurONAL-P with varying μ and γ values (500, 1000, 2000) on four datasets. Intuitively, if μ and γ are too small, NeurONAL-P will place more weight

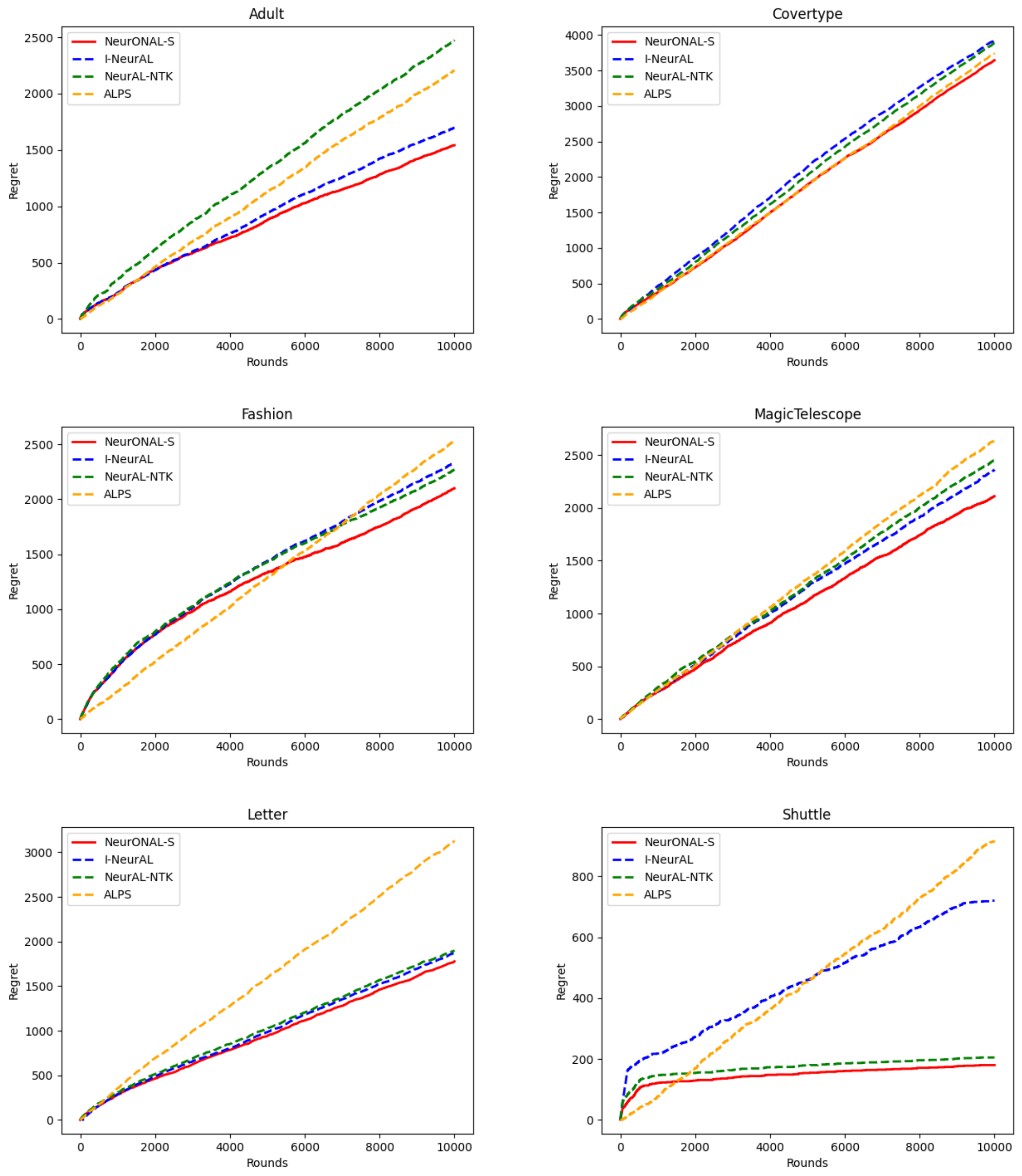

Figure 4.4: Regret comparison on six datasets in the **stream-based** setting. Note: Margin and NeurONAL-S perform similarly, so it might be hard to see both lines.

on the tail of the distribution while sampling. Otherwise, NeurONAL-P will focus more on the head of the sampling distribution. From the results in the table, it seems that different datasets respond to different values of μ and γ. This sensitivity study roughly shows good values for μ and γ.

	Adult	MT	Letter	Covertype	Shuttle	Fashion
	Accuracy					
CoreSet	76.7 ± 1.13	75.1 ± 0.79	80.6 ± 0.63	62.6 ± 3.11	97.7 ± 0.41	80.4 ± 0.08
BADGE	76.6 ± 0.49	71.6 ± 0.81	81.7 ± 0.57	64.8 ± 1.02	98.6 ± 0.39	76.1 ± 0.21
DynamicAL	72.4 ± 0.14	67.8 ± 1.01	63.2 ± 0.31	54.1 ± 0.12	78.7 ± 0.05	54.5 ± 0.19
ALBL	78.1 ± 0.45	73.9 ± 0.71	81.9 ± 0.47	65.3 ± 0.14	98.6 ± 0.37	77.6 ± 0.32
NeurONAL-P	**79.1 ± 0.04**	**81.3 ± 0.12**	**83.7 ± 0.07**	**67.6 ± 0.21**	**99.5 ± 0.01**	**81.1 ± 0.13**
	Running Time					
CoreSet	43.1 ± 7.65	119.3 ± 4.42	228.3 ± 6.51	32.5 ± 10.94	10.9 ± 2.22	33.1 ± 6.32
BADGE	159.5 ± 4.61	212.5 ± 9.32	484.8 ± 7.04	545.7 ± 9.32	222.9 ± 5.13	437.8 ± 5.32
DynamicAL	24.3 ± 5.21	821.5 ± 6.14	382.3 ± 3.13	621.6 ± 3.21	483.4 ± 9.78	413.2 ± 7.14
ALBL	315.8 ± 4.31	343.5 ± 6.24	271.3 ± 6.32	481.3 ± 5.21	63.2 ± 2.16	92.1 ± 3.42
NeurONAL-P	**17.2 ± 3.24**	140.1 ± 3.69	**133.7 ± 12.8**	**14.1 ± 5.81**	15.6 ± 8.03	**25.5 ± 7.80**

Table 4.3: Testing accuracy (%) and running time on all methods in **pool-based** Setting

Letter

		γ 500	1000	2000
μ	500	80.9%	81.7%	80.5%
	1000	77.9%	**83.9**%	78.9%
	2000	81.7%	81.8%	80.1%

Adult

		γ 500	1000	2000
μ	500	**79.9**%	79.4%	78.9%
	1000	79.1%	79.7%	79.0%
	2000	79.4%	79.4%	79.7%

Fashion

		γ 500	1000	2000
μ	500	80.3%	80.5%	79.5%
	1000	80.5%	80.6%	80.4%
	2000	80.8%	**80.9**%	80.7%

MT

		γ 500	1000	2000
μ	500	79.5%	80.9%	80.6%
	1000	80.2%	80.9%	80.1%
	2000	80.5%	80.6%	**81.3**%

Table 4.4: Testing Accuracy on four datasets (Letter, Adult, Fashion, and MT) with varying μ and γ in pool-based setting

Label budget. Tables 4.5, 4.6, and 4.7 show NeurONAL-S' performance in active learning with different budget percentages: 3%, 10%, 50%. NeurONAL-S achieves the best performance in most of the experiments. With 3% label budget, almost all neural network models are not well trained. Thus, NeurONAL-S does not perform stably. With a 10% and 50% label budget, NeurONAL-S achieves better performance, because the advantages of neural network models can better exploit and explore this labeled information.

4.4 CONNECTION TO GAD.

At first glance, this method does not seem to directly relate to graph anomaly detection. However, GAD can be thought of as a downstream task for this model training algorithm.

	Adult	Covertype	Fashion	MagicTelescope	Letter	Shuttle
I-NeurAL	79.4%	52.8%	51.9%	72.3%	74.6%	93.0%
NeurAL-NTK	23.9%	1.56%	11.9%	32.9%	42.8%	70.6%
ALPS	24.2%	36.8%	10.0%	64.9%	72.7%	79.4%
NeurONAL-S	**79.9%**	**65.6%**	69.7%	**77.3%**	74.2%	**99.8%**

Table 4.5: Test accuracy with 3% budget in stream-based setting

	Adult	Covertype	Fashion	MagicTelescope	Letter	Shuttle
I-NeurAL	80.5%	55.4%	71.4%	77.9%	81.8%	99.2%
NeurAL-NTK	70.5%	59.9%	38.7%	34.3%	53.8%	75.9%
ALPS	24.2%	36.8%	10.0%	35.1%	79.9%	79.4%
NeurONAL-S	**79.5%**	**71.3%**	81.3%	**82.1%**	**81.8%**	**99.8%**

Table 4.6: Test accuracy with 10% budget in stream-based setting

	Adult	Covertype	Fashion	MagicTelescope	Letter	Shuttle
I-NeurAL	83.4%	65.9%	82.5%	77.9%	85.8%	99.7%
NeurAL-NTK	76.9%	73.1%	56.8%	81.6%	79.3%	97.1%
ALPS	75.8%	36.8%	10.0%	64.9%	81.5%	79.4%
NeurONAL-S	**84.6%**	**75.9%**	**85.4%**	**86.4%**	**86.9%**	**99.8%**

Table 4.7: Test accuracy with 50% budget in stream-based setting

Anomaly detection is a flavor of classification where the two classes are "anomalous" and "benign". One key difference is that anomaly detection is an imbalanced classification problem, whereas k-classification assumes balance in the dataset. For this reason, anomaly detection methods need to be designed carefully to offset the imbalance in the dataset.

Since NeurONAL is designed to work with tabular data, we can create node embeddings using a graph neural network (or a variant), and treat the embeddings as tabular data. Experimentation with this result will be performed in Chapter 5.

4.5 SUMMARY

In this chapter, we described an active learning algorithm for training a multi-armed bandit on the downstream task of multi-class classification. The dual neural network architecture represents the exploitation and exploration goals accordingly. The exploitation network aims to assign labels to data points while the exploration network aims to estimate the uncertainty of the exploitation network. As shown in the evaluation, using both networks allowed us to outperform the baselines - a few of the established algorithms that solve this problem - by a margin of 0.6-47.8%.

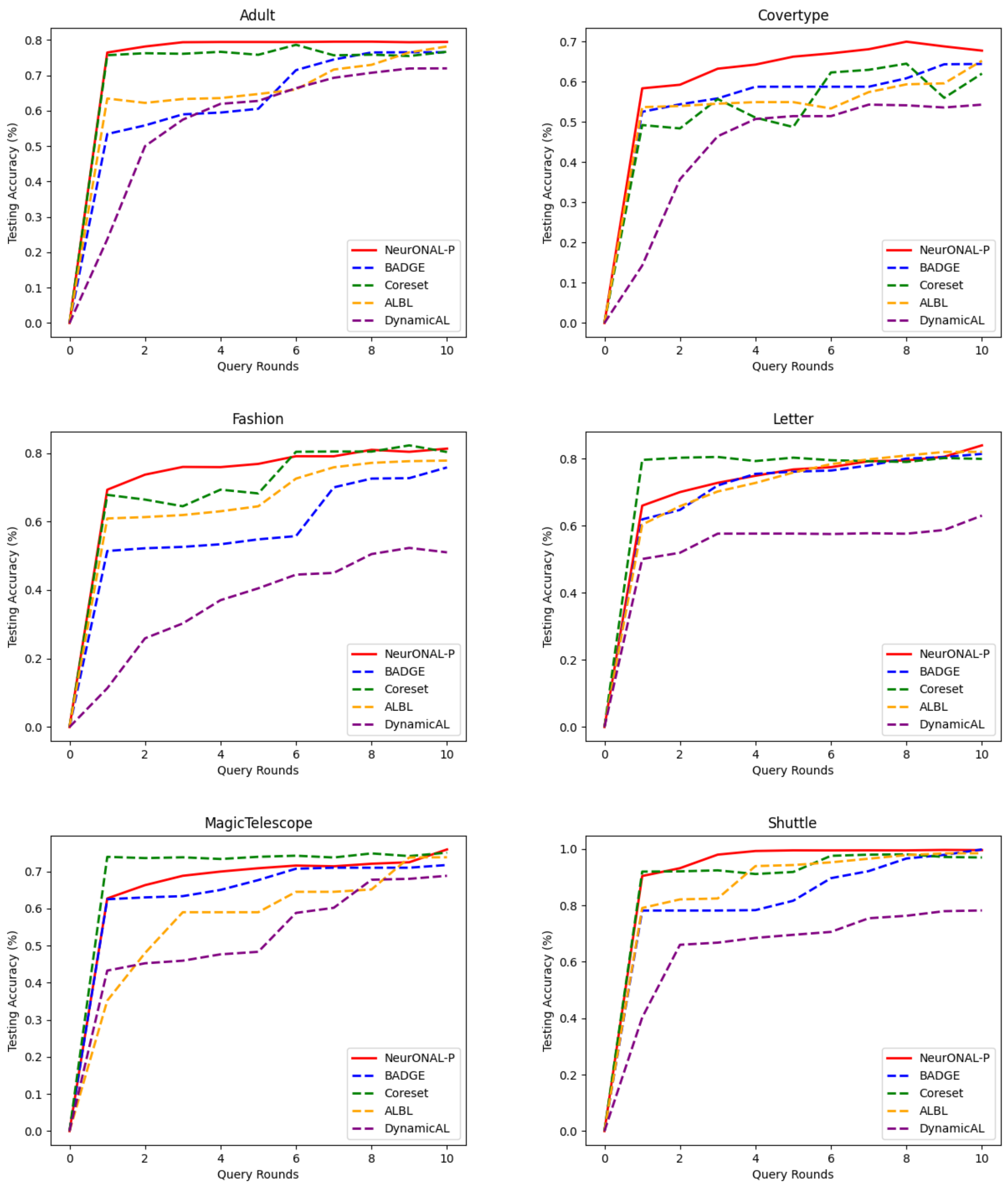

Figure 4.5: Test accuracy versus the number of query rounds in **pool-based** setting on six datasets. NeurONAL-P outperforms baselines on all datasets.

CHAPTER 5: GAD USING BANDITS

In this chapter, we describe a solution for anomaly detection using active bandit-based approaches. This method is a combination of the previous two methods mentioned in this thesis (AMUGraph and NeurONAL). From AMUGraph, we are able to generate soft labels for node classification. From NeurONAL, we are able to balance exploitation and exploration with the use of neural networks. Combining the two frameworks will enable us to leverage the strong points of both methods. We call this combined method **A**ctive **M**ix **U**p for GAD using **Bandits** (AMUBandits).

5.1 PROBLEM FORMULATION

Input: (1) a graph $G = (V, E, X)$ with nodes V - with attributes X - and edges E, (2) an oracle $\mathbb{O}$ that provides soft labels.

Goal: to learn a bandit that learns to assign a given data instance x a label from the label set $\{\text{anomalous}, \text{benign}\}$.

5.2 METHODOLOGY

Recalling the exploitation and exploration networks illustrated in Figure 4.3, and the SemiVAE model from Figure 3.1, we can combine these diagrams to illustrate the neural network architecture for this new model, as shown in Figure 5.1. Here, the exploitation network still learns to classify data instances as anomalous or benign hence, AMUGraph behaves as our exploitation network. Moreover, the classification technique illustrated in Figure 3.2 remains the same (it is represented by the box labeled 'Classification'). Same as in NeurONAL, the exploration network still models the uncertainty in the exploitation network.

We replace the neural network with AMUGraph because, while the neural network will use uncertainty in prediction as a querying metric, AMUGraph uses the reconstruction loss. Ultimately, using the reconstruction loss makes more sense in the task of anomaly detection.

Active Learning. Here, we answer the three questions relevant to active learning algorithm design outlined in Section 1.2.2. Calculating uncertainty is similar to that of AMUGraph: based on the reconstruction loss. The querying is also similar to AMUGraph where a query is made to a human expert who will return a soft label. Query information max-

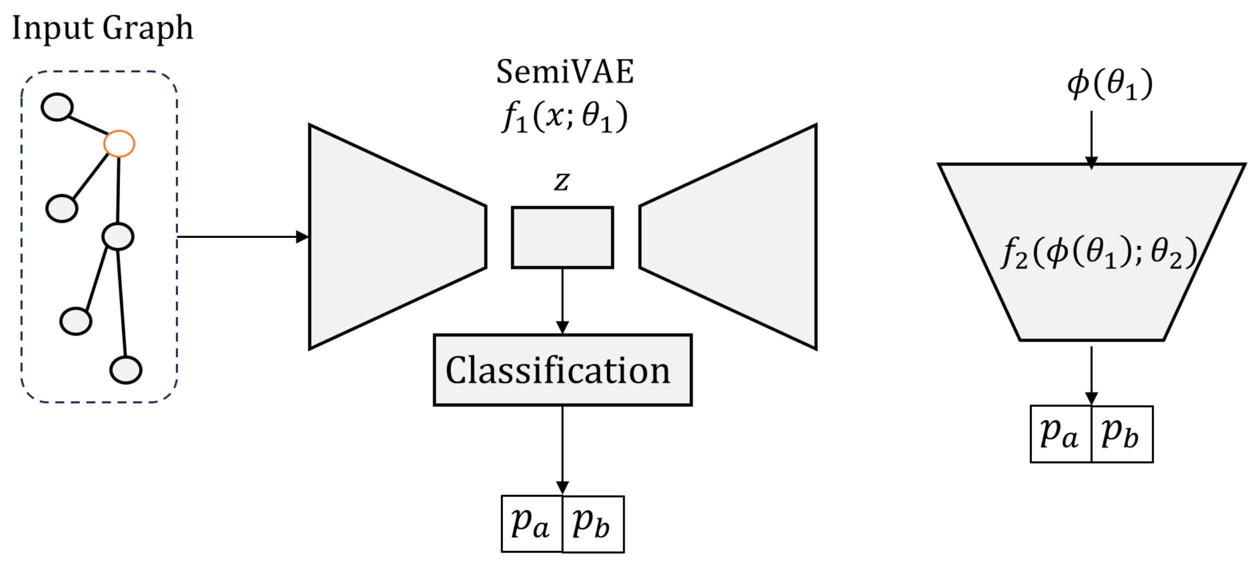

Figure 5.1: The two neural networks for exploitation (left) using AMUGraph's SemiVAE architecture and exploration (right) using f_2 from NeurONAL. Here, p_a and p_b are the probabilities of a node being anomalous and benign, respectively where $p_a + p_b = 1$.

imization has two parts: (1) data augmentation and (2) a joint cooperation between the exploration and exploitation networks to learn to predict a correct class label with high certainty.

5.3 EMPIRICAL RESULTS

5.3.1 Experimental Setup

We use the same datasets as in Chapter 3. We add the Amazon dataset [56] where the nodes are users and edges connect users to products, products to reviews, and reviews to users. To calculate performance, we use ROC-AUC. We use GDN (from Chapter 3) and our previous methods as baselines - namely, AMUGraph, NeurONAL-S, and NeurONAL-P.

To apply NeurONAL-S and NeurONAL-P on graph datasets, we create node embeddings using Node2Vec [57] that will encode the structural information. We concatenate the node attributes to the embedding and treat this as tabular data, with which the NeurONAL algorithms are equipped to handle.

5.3.2 Results

The results are gathered in Table 5.1. The comparison between the results from AMUGraph and AMUBandits shows that AMUGraph benefits from the principled exploration and exploitation, with a 3-20% performance increase. AMUBandits and NeurONAL-P tend to perform differently for different datasets. NeurONAL-P chooses points to query by estimating the prediction uncertainty of the data, creating a probabilistic distribution where higher uncertainty yields higher sampling probability, and sampling from the distribution. AMUBandits chooses points based on the reconstruction loss criteria from AMUGraph. According to the results in Table 5.1, there is no clear indication why a certain querying strategy works over another.

NeurONAL-S outperforms AMUGraph, NeurONAL-P, and AMUBandits on two out of three datasets. However, that could be due to the method existing in a different setting. Overall, NeurONAL-S uses much more labeled data than any of the pool-based settings do.

5.3.3 Hyperparameter Tuning

Tables 5.2, 5.3, and 5.4 display the performance of AMUBandits on the Amazon, Pubmed and Yelp datasets, respectively, with varying training epoch rounds and varying latent dimension sizes. Unfortunately, there is no clear pattern that can help interpret the range of values our method performs well on. Each dataset exhibits different trends.

5.4 SUMMARY

In this chapter, we describe a work that combines the two algorithms AMUGraph and NeurONAL into one algorithm AMUBandits: using active learning to train a multi-armed bandit for node anomaly detection using soft labels. We keep the principle of the exploitation and exploration network, where the exploitation network is essentially AMUGraph. We show through our evaluation that it can outperform a few of the baselines, but still requires work to reliably outperform the stronger baselines.

	Pubmed	Yelp	Amazon
GDN	61.7	67.8	88.5
AMUGraph	74.1	**72.6**	53.6
NeurONAL-P	69.7	55.9	<u>89.1</u>
NeurONAL-S	**78.9**	67.8	**89.5**
AMUBandits-NoAug	73.2	57.9	71.3
AMUBandits	<u>77.8</u>	<u>69.4</u>	73.2

Table 5.1: ROC-AUC (%) on selected datasets and baselines compared with AMUBandits. Higher is better.

E/L.D.	8	16	32
5	50.9	59.6	64.4
10	52.5	54.7	54.4
20	52.3	**73.2**	55.0

Table 5.2: ROC-AUC (%) for varying number of training epochs (denoted by E, values change by rows) and latent dimension sizes (denoted by L.D., values change by columns) on the **Amazon** dataset.

E /L.D.	8	16	32
5	63.7	75.2	**77.8**
10	61.2	77.7	62.7
20	51.8	51.4	53.0

Table 5.3: ROC-AUC (%) for varying number of training epochs (denoted by E, values change by rows) and latent dimension sizes (denoted by L.D., values change by columns) on the **Pubmed** dataset.

E /L.D.	8	16	32
5	59.3	54.0	**69.4**
10	58.4	53.2	65.7
20	53.3	57.5	51.1

Table 5.4: ROC-AUC (%) for varying number of training epochs (denoted by E, values change by rows) and latent dimension sizes (denoted by L.D., values change by columns) on the **Yelp** dataset.

CHAPTER 6: CONCLUSION

6.1 SUMMARY

This thesis explores many different methods for graph anomaly detection using representation learning with data augmentation, and with multi-armed bandits. Overall, all methods employed active learning in the problem settings. Each method worked alongside with a human simulator to learn a model that is comparable to a model with full access to the labeled information. One significant advantage of these works is the simplicity and contained nature - future research with these methods as a backbone can be expected without hangups due to the implementations.

From the AMUGraph work, we were able to show that augmenting queried data provided significant performance boosts to the task of graph anomaly detection. Next, from the NeurONAL work, we showed that quantifying exploration and exploitation led to better data point selection for querying to the task of classification. Finally, from the AMUBandits work, we began to show the benefits of using data augmentation with exploration-exploitation-aware data selection strategies.

6.2 FUTURE WORK

6.2.1 Using Temporal Signals

Many GAD applications contain anomalies that progress over time. Our current framework cannot handle temporal information, but it can be modified to do so. As a naive baseline, we can run any of the three methods on each timestamp of the graph and keep track of the confidence of the predictions. Once the difference in confidences changes beyond a threshold, we can label such a node anomalous.

6.2.2 Unreliable Soft Labeling

In real life applications of soft labeling, adversarial approaches are most likely necessary to handle labelers that provide wrong information (intentionally or accidentally), or vague information (uncertain labels, i.e., confidence near 50%). In such situations, an additional teacher model for knowledge distillation can come in handy. We can also make use of anti-bias learning techniques from the Fair AI community to learn robust models despite biased soft labels.

6.2.3 Unified GAD

There is much work to be done in this area, and in the general area of graph anomaly detection. First off, there are other kinds of anomalies in graphs besides nodes - edges and subgraphs. Depending on the use case, detecting anomalous edges and subgraphs could be useful in learning good graph models.

One such use case could be **data cleaning**. Graph data are expensive to collect, and therefore might not be entirely clean and error-free. Being able to detect such mistakes made in data collection will be valuable. Another use case could be **detecting anomalies in complex physical representations**, such as river networks [58] and power grid systems [59]. Identifying anomalies is crucial in such systems, especially considering the various types of anomalies. For instance, in the power grid example, suspicious activity centered around a specific tower could correspond to an anomalous node while that for a power line could correspond to an anomalous edge. Being able to identify anomalies with that level of granularity would prove very helpful.

The methods mentioned in this thesis can be furthered to handle these other kinds of anomalies as well. Currently, there is not a lot of work on learning a singular model to classify nodes *and* edges, separately. A few reasons why researchers have not tapped into this area could be: (1) there are little to no relevant datasets available, despite the practical use cases, (2) such one-size-fits-all algorithms are quite challenging to train, and (3) these algorithms might need to be domain-specific, making it difficult to generalize to other domains easily.

We could broaden the scope of these works by handling the graph using different views, and using principles from multi-scale anomaly detection [60, 61, 62]. These works use the structural information along with the node attribute information to learn robust anomaly detection models. Some of these works use contrastive learning to differentiate a node and an ego-net. We could also borrow the idea of task adapters from the natural language processing community [63] to jointly learn but separately infer on anomalous nodes, edges, and/or subgraphs.

6.2.4 Generative Modeling for Active Learning

As mentioned in Section 1.2.2, a third (and unexplored in this thesis) strategy for active learning is membership query synthesis (MQS). In this strategy, the queries are not made on the available data, but on manipulated samples of fake data that can yield the most possible information. This is a very niche area of active learning because the queries made by MQS techniques are mostly not interpretable by humans. However, MQS is a very powerful framework that can reduce the number of queries needed by a significant amount because each query contains more information (due to the nature of the query creation) compared to queries made with stream-based or pool-based active learning.

Since these queries are not human readable, [64] has become a standard solution to using MQS with representation learning-based approaches. Using a binary-search like approach, the decision boundary is approximated. One point above and one point below the decision boundary are found where the midpoint m_p between those is queried. Next, they find the nearest neighbor of the point m_p, decode it, and query the label for that point. Technically, queries are still made on the existing data, but the learner is able to guarantee good label information by ensuring the data is human readable.

REFERENCES

[1] R. Zafarani, M. A. Abbasi, and H. Liu, *Social media mining: an introduction.* Cambridge University Press, 2014.

[2] J. Xu, S. Kim, M. Song, M. Jeong, D. Kim, J. Kang, J. F. Rousseau, X. Li, W. Xu, V. I. Torvik et al., "Building a pubmed knowledge graph," *Scientific data*, vol. 7, no. 1, p. 205, 2020.

[3] S. Zhang, H. Yin, T. Chen, Q. V. N. Hung, Z. Huang, and L. Cui, "Gcn-based user representation learning for unifying robust recommendation and fraudster detection," in *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, 2020, pp. 689–698.

[4] X. Ma, J. Wu, S. Xue, J. Yang, C. Zhou, Q. Z. Sheng, H. Xiong, and L. Akoglu, "A comprehensive survey on graph anomaly detection with deep learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 12, pp. 12 012–12 038, 2023.

[5] R. Yu, H. Qiu, Z. Wen, C. Lin, and Y. Liu, "A survey on social media anomaly detection," *ACM SIGKDD Explorations Newsletter*, vol. 18, no. 1, p. 1–14, Aug. 2016.

[6] M. Orabi, D. Mouheb, Z. Al Aghbari, and I. Kamel, "Detection of bots in social media: a systematic review," *Information Processing & Management*, vol. 57, no. 4, p. 102250, 2020.

[7] J. Tang, F. Hua, Z. Gao, P. Zhao, and J. Li, "Gadbench: Revisiting and benchmarking supervised graph anomaly detection," in *Advances in Neural Information Processing Systems*, A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36. Curran Associates, Inc., 2023. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/file/5eaafd67434a4cfb1cf829722c65f184-Paper-Datasets_and_Benchmarks.pdf p. 29628–29653.

[8] J. Chen, Q. Chen, F. Jiang, X. Guo, K. Sha, and Y. Wang, "Scn_gnn: A gnn-based fraud detection algorithm combining strong node and graph topology information," *Expert Systems with Applications*, vol. 237, p. 121643, 2024.

[9] Z. Liu, C. Cao, F. Tao, and J. Sun, "Revisiting graph contrastive learning for anomaly detection," no. arXiv:2305.02496, May 2023, arXiv:2305.02496 [cs]. [Online]. Available: http://arxiv.org/abs/2305.02496

[10] K. Ding, J. Li, and H. Liu, "Interactive Anomaly Detection on Attributed Networks," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining.* Melbourne VIC Australia: ACM, Jan. 2019. [Online]. Available: https://dl.acm.org/doi/10.1145/3289600.3290964 pp. 357–365.

[11] G. Pang, "Toward Deep Supervised Anomaly Detection: Reinforcement Learning from Partially Labeled Anomaly Data," 2022. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/3447548.3467417

[12] K. Ding, Q. Zhou, H. Tong, and H. Liu, "Few-shot network anomaly detection via cross-network meta-learning," 2021.

[13] Y. Ban, I. Agarwal, Z. Wu, Y. Zhu, K. Weldemariam, H. Tong, and J. He, "Neural active learning beyond bandits," in *International Conference on Learning Representations*, 2024.

[14] D. Bouneffouf, I. Rish, and C. Aggarwal, "Survey on applications of multi-armed and contextual bandits," in *2020 IEEE Congress on Evolutionary Computation (CEC)*, 2020, pp. 1–8.

[15] M. Coggan, "Exploration and exploitation in reinforcement learning."

[16] B. Settles, "Active learning literature survey," 2010.

[17] E. B. Baum and K. Lang, "Query learning can work poorly when a human oracle is used," in *International joint conference on neural networks*, vol. 8. Beijing China, 1992, p. 8.

[18] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," no. arXiv:1710.09412, Apr. 2018, arXiv:1710.09412 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1710.09412

[19] F. Zhu, Z. Cheng, X.-Y. Zhang, and C.-L. Liu, "OpenMix: Exploring Outlier Samples for Misclassification Detection," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* Vancouver, BC, Canada: IEEE, June 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10205216/ pp. 12 074–12 083.

[20] Z. Cheng, Z. Jiang, Y. Yin, C. Wang, and Q. Gu, "Learning to classify open intent via soft labeling and manifold mixup," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 635–645, 2022.

[21] X. Wang, B. Jin, Y. Du, P. Cui, and Y. Yang, "One-Class Graph Neural Networks for Anomaly Detection in Attributed Networks," *Neural Computing and Applications*, vol. 33, pp. 12 073–12 085, Sep. 2021, arXiv:2002.09594 [cs, stat]. [Online]. Available: http://arxiv.org/abs/2002.09594

[22] H. Xu, Y. Wang, G. Pang, S. Jian, N. Liu, and Y. Wang, "RoSAS: Deep Semi-Supervised Anomaly Detection with Contamination-Resilient Continuous Supervision," July 2023, arXiv:2307.13239 [cs]. [Online]. Available: http://arxiv.org/abs/2307.13239

[23] K. Ding, J. Li, R. Bhanushali, and H. Liu, “Deep Anomaly Detection on Attributed Networks,” in *Proceedings of the 2019 SIAM International Conference on Data Mining (SDM)*, ser. Proceedings. Society for Industrial and Applied Mathematics, May 2019, pp. 594–602. [Online]. Available: https://epubs.siam.org/doi/10.1137/1.9781611975673.67

[24] J. Lin, Y. He, W. Xu, J. Guan, J. Zhang, and S. Zhou, “Latent feature reconstruction for unsupervised anomaly detection,” *Applied Intelligence*, vol. 53, no. 20, pp. 23 628–23 640, Oct. 2023. [Online]. Available: https://doi.org/10.1007/s10489-023-04767-2

[25] T. Pimentel, M. Monteiro, A. Veloso, and N. Ziviani, “Deep active learning for anomaly detection,” in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–8.

[26] T. Zhou, K.-H. Thung, M. Liu, F. Shi, C. Zhang, and D. Shen, “Multi-modal latent space inducing ensemble svm classifier for early dementia diagnosis with neuroimaging data,” *Medical image analysis*, vol. 60, p. 101630, 2020.

[27] J. Li, X. Xu, L. Gao, Z. Wang, and J. Shao, “Cognitive visual anomaly detection with constrained latent representations for industrial inspection robot,” *Applied Soft Computing*, vol. 95, p. 106539, 2020.

[28] H. Liu, Y. Zhan, H. Xia, Q. Mao, and Y. Tan, “Self-supervised transformer-based pre-training method using latent semantic masking auto-encoder for pest and disease classification,” *Computers and Electronics in Agriculture*, vol. 203, p. 107448, 2022.

[29] T. Huang, P. Chen, and R. Li, “A Semi-Supervised VAE Based Active Anomaly Detection Framework in Multivariate Time Series for Online Systems,” in *Proceedings of the ACM Web Conference 2022*. Virtual Event, Lyon France: ACM, Apr. 2022. [Online]. Available: https://dl.acm.org/doi/10.1145/3485447.3511984 pp. 1797–1806.

[30] T. Zhang, K. Johansson, and N. Li, “Multi-armed Bandit Learning on a Graph,” in *2023 57th Annual Conference on Information Sciences and Systems (CISS)*. Baltimore, MD, USA: IEEE, Mar. 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10089744/ pp. 1–6.

[31] K. Arshad, R. F. Ali, A. Muneer, I. A. Aziz, S. Naseer, N. S. Khan, and S. M. Taib, “Deep reinforcement learning for anomaly detection: A systematic review,” *IEEE Access*, vol. 10, pp. 124 017–124 035, 2022.

[32] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks,” 2014.

[33] C. Zhong, M. C. Gursoy, and S. Velipasalar, “Deep actor-critic reinforcement learning for anomaly detection,” in *2019 IEEE global communications conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[34] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 413–422.

[35] J. Dong, Q. Zhang, X. Huang, Q. Tan, D. Zha, and Z. Zihao, "Active Ensemble Learning for Knowledge Graph Error Detection," in *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining.* Singapore Singapore: ACM, Feb. 2023. [Online]. Available: https://dl.acm.org/doi/10.1145/3539597.3570368 pp. 877–885.

[36] X. Ying, "An overview of overfitting and its solutions," in *Journal of physics: Conference series*, vol. 1168. IOP Publishing, 2019, p. 022022.

[37] H. Ling, Z. Jiang, M. Liu, S. Ji, and N. Zou, "Graph mixup with soft alignments," in *Proceedings of the 40th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., vol. 202. PMLR, 23–29 Jul 2023. [Online]. Available: https://proceedings.mlr.press/v202/ling23a.html pp. 21 335–21 349.

[38] Y. Wang, W. Wang, Y. Liang, Y. Cai, and B. Hooi, "Mixup for node and graph classification," in *Proceedings of the Web Conference 2021*, 2021, pp. 3663–3674.

[39] S. Zhou, X. Huang, N. Liu, H. Zhou, F.-L. Chung, and L.-K. Huang, "Improving Generalizability of Graph Anomaly Detection Models via Data Augmentation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 12, pp. 12 721–12 735, Dec. 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10119211/

[40] T. Yang, Y. Huang, Y. Xie, J. Liu, and S. Wang, "MixOOD: Improving Out-of-distribution Detection with Enhanced Data Mixup," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 19, no. 5, pp. 1–18, Sep. 2022. [Online]. Available: https://dl.acm.org/doi/10.1145/3578935

[41] N. Chen, Z. Liu, B. Hooi, B. He, R. Fathony, J. Hu, and J. Chen, "Consistency training with learnable data augmentation for graph anomaly detection with limited supervision," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=elMKXvhhQ9

[42] Q. Nguyen, H. Valizadegan, and M. Hauskrecht, "Learning classification models with soft-label information," *Journal of the American Medical Informatics Association*, vol. 21, no. 3, pp. 501–508, 2014.

[43] J. Wang, H. Xie, F. L. Wang, and L.-K. Lee, "Improving text classification via a soft dynamical label strategy," *International Journal of Machine Learning and Cybernetics*, vol. 14, no. 7, pp. 2395–2405, 2023.

[44] D. Liang, F. Yang, T. Zhang, and P. Yang, "Understanding mixup training methods," *IEEE Access*, vol. 6, pp. 58 774–58 783, 2018.

[45] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, pp. 93–93, 2008.

[46] S. Rayana and L. Akoglu, "Collective opinion spam detection: Bridging review networks and metadata," in *Proceedings of the 21th acm sigkdd international conference on knowledge discovery and data mining*, 2015, pp. 985–994.

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

[48] F. Z. Haoyi Fan and Z. Li, "Anomalydae: Dual autoencoder for anomaly detection on attributed networks," in *45th International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 2020.

[49] Z. Wang, P. Awasthi, C. Dann, A. Sekhari, and C. Gentile, "Neural active learning with performance guarantees," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[50] Y. Ban, Y. Zhang, H. Tong, A. Banerjee, and J. He, "Improved algorithms for neural active learning," in *Advances in Neural Information Processing Systems*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., 2022. [Online]. Available: https://openreview.net/forum?id=riIaC2ivcYA

[51] G. DeSalvo, C. Gentile, and T. S. Thune, "Online active learning with surrogate loss functions," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[52] O. Sener and S. Savarese, "Active learning for convolutional neural networks: A core-set approach," *arXiv preprint arXiv:1708.00489*, 2017.

[53] J. T. Ash, C. Zhang, A. Krishnamurthy, J. Langford, and A. Agarwal, "Deep batch active learning by diverse, uncertain gradient lower bounds," *arXiv preprint arXiv:1906.03671*, 2019.

[54] H. Wang, W. Huang, Z. Wu, H. Tong, A. J. Margenot, and J. He, "Deep active learning by leveraging training dynamics," *Advances in Neural Information Processing Systems*, vol. 35, pp. 25 171–25 184, 2022.

[55] W.-N. Hsu and H.-T. Lin, "Active learning by learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, 2015.

[56] Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, and P. S. Yu, "Enhancing graph neural network-based fraud detectors against camouflaged fraudsters," in *Proceedings of the 29th ACM international conference on information & knowledge management*, 2020, pp. 315–324.

[57] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[58] K. Buchhorn, E. Santos-Fernandez, K. Mengersen, and R. Salomone, "Graph neural network-based anomaly detection for river network systems," *arXiv preprint arXiv:2304.09367*, 2023.

[59] S. Li, A. Pandey, B. Hooi, C. Faloutsos, and L. Pileggi, "Dynamic graph-based anomaly detection in the electrical grid," *IEEE Transactions on Power Systems*, vol. 37, no. 5, pp. 3408–3422, 2021.

[60] M. Jin, Y. Liu, Y. Zheng, L. Chi, Y.-F. Li, and S. Pan, "Anemone: Graph anomaly detection with multi-scale contrastive learning," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 3122–3126.

[61] J. Duan, S. Wang, P. Zhang, E. Zhu, J. Hu, H. Jin, Y. Liu, and Z. Dong, "Graph anomaly detection via multi-scale contrastive learning networks with augmented view," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 6, 2023, pp. 7459–7467.

[62] L. Gutiérrez-Gómez, A. Bovet, and J.-C. Delvenne, "Multi-scale anomaly detection on attributed networks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 01, 2020, pp. 678–685.

[63] N. Bang, J. Lee, and M.-W. Koo, "Task-optimized adapters for an end-to-end task-oriented dialogue system," *arXiv preprint arXiv:2305.02468*, 2023.

[64] L. Wang, X. Hu, B. Yuan, and J. Lu, "Active learning via query synthesis and nearest neighbour search," *Neurocomputing*, vol. 147, pp. 426–434, 2015.