Adversarial Filtering on Graph Data

Ayushi Agarwal
705496407
Department of Computer Science
ayushi15@ucla.edu

Abstract

Neural models of significant size have exhibited a level of performance on language and vision benchmarks comparable to humans. However, their performance suffers significantly when presented with adversarial or out-of-distribution samples, which prompts the inquiry of whether these models have learned to tackle a dataset instead of the underlying task by excessively fitting to irrelevant dataset biases. Some of the research has been done in removing spurious biases from NLP and vision datasets and prevent overestimatation of the performance of the models. To date, there has been no comparable exploration or mitigation efforts for biases in Graph Datasets. The interdependence among the points in these datasets presents a challenge when it comes to detecting spurious biases. Our objective in this project is to investigate Graph datasets and determine whether they contain any such biases. Subsequently, we will attempt to iteratively eliminate these biases and evaluate the resulting impact on model performance.

1 Introduction

Adversarial filters of dataset biases refer to techniques used to mitigate the impact of biases present in training data on machine learning models. These biases can cause models to make unfair or unjust decisions. The observed trend highlights that the exceptional performance of leading AI models is frequently limited to particular datasets, which implies a closed-world perspective. However, authentic task learning demands generalization, or an open-world perspective. The existence of unintended correlations between input and output, also known as spurious biases, within current datasets presents a significant obstacle to achieving generalization.

Adversarial training is a promising technique to eliminate biases from datasets. Adversarial examples are generated to test the model's ability to disregard biases in the training data. This process produces a more equitable and resilient model that is less prone to the impact of biased training data. Subsequently, the model's performance is evaluated on several benchmark datasets.

The authors of [4] present an algorithm that is predicated on the notion that peculiarities present in a dataset can significantly inflate model performance. Their goal is to eliminate spurious biases that exist at the instance level and are specific to the data. The objective, therefore, is to reduce the representation bias of the feature representation in a given dataset. This is accomplished by minimizing the predictability score of each instance. The predictability score is determined by how accurately a model can predict an instance when trained on a random subset of instances that does not include the target instance. In essence, the authors aim to identify a subset of instances that minimizes the predictability score of the entire dataset.

The use of graph representations in various applications is becoming more prevalent. Graph Neural Networks (GNNs) have emerged as a popular solution for processing graph data that feature intricate relationships and dependencies. However, the inherent complexities of graph data have introduced significant hurdles for existing algorithms to effectively learn

Preprint. Under review.

from them. In contrast to traditional adversarial training methods, this approach presupposes that the instances in the dataset are independent, which is not applicable to graph data.

2 Related Work

2.1 Adversarial Filters of Dataset Biases

AFLITE [4], is an iterative and greedy algorithm designed to filter out spurious biases from data as a preprocessing step, leading to more accurate benchmark estimations. Authors provide a theoretical foundation supporting AFLITE and demonstrate its efficacy in reducing biases on synthetic and real datasets through comprehensive analyses. They apply AFLITE to four datasets, including well-known benchmarks such as SNLI and ImageNet. They also present results on out-of-distribution and adversarial test sets designed for such benchmarks showing that models trained on AFLITE-filtered subsets perform better, highlighting their higher generalization abilities.

To remove points from the dataset they use the following algorithm. Give a dataset S, pretrained embeddings of all instances in S are taken. A random subset of these instances is taken of size say $(T - S)$. A Linear Classifier (M) is then trained on this random subset and predictions are collected for all the out of sample instances (T). The out of sample results are stored for each instance and the process is repeated m number of times.

2.2 Graph Convolution Networks

The paper [3] developed a method for classifying graph-structured data with partial supervision. The authors created a GCN model which employs a layer-wise propagation rule that efficiently approximates spectral convolutions on graphs. The experiments on various network datasets indicate that the GCN model can effectively capture both the node features and graph structure for semi-supervised classification. This model will be used to capture graph structure along with different features for removing spurious biases.

2.3 GOOD: A Graph Out-of-Distribution Benchmark

GOOD [2] is an OOD (Out-of-Distribution) benchmark designed specifically for graphs. It aims to distinguish between covariate and concept shifts and provides data splits that accurately reflect these shifts. The benchmark consists of 11 datasets with 17 domain selections, covering both graph and node prediction tasks. Graph prediction tasks involve predicting properties of entire graphs, while node prediction tasks involve predicting properties of individual nodes within a graph. The distinction between covariate and concept shifts is crucial for OOD benchmarking. Covariate shift occurs when the distribution of the input data changes, while the underlying concept remains the same. Concept shift, on the other hand, occurs when the underlying concept changes, while the input distribution remains the same.

3 Methodology

The AFLite algorithm is first applied to the Synthetic dataset given in the paper [4] and the github [7]. The aim is to replicate the outcomes reported in the paper. Subsequently, the algorithm is employed on the Graph dataset to evaluate its efficacy in dealing with related data points. We perform different variations of the algorithm to remove spurious data points in graphs.

3.1 AFLite on Graph Dataset

When utilizing AFLite for Graph Datasets, we opt for the node classification assignment. We begin by employing pre-trained node features, which are primarily derived from word-to-vector conversions using a dictionary that is exclusive to the dataset. Every edge associated with a particular node is eliminated when that node is removed.

3.2 Iterative AFLite

We tried to run the AFLite algorithm iteratively to see if it gives us better results. By employing this method, it would be possible to utilize iterative AFLite on GNN models like RGCN [6] and CompGCN [8], as well as conduct adversarial filtering on graph datasets that lack predetermined features, such as WordNet or FreeBase, or in situations where features are learned concurrently with downstream tasks. To so so we modify the AFLite Algorithm the following way:

The data is segregated into three distinct sets, namely train, validation, and test. n_{init} iterations are executed on the train data, which enables the network to attain a warm start. We set the hyperparameter 'i', and after each i iterations, k data points are eliminated from the graph (from both the train and validation sets). In order to determine which data point to remove, we maintain a table that records the count of correctly labeled datapoints during the forward pass. We also reduce the value of k by x percentage after each removal step.

3.3 Iterative AFLite without Features

As a follow-up to implementing iterative AFLite on predetermined features, we aim to learn features concurrently with downstream tasks via the employment of GCN. The features that are learned through this process are subsequently utilized to remove nodes using the Iterative AFLite algorithm. The learning process is then continued on the remaining features and the modified graph. The algorithm is given in 1.

Algorithm 1: Iterative AFLite Algorithm for a Graph with no features

Data: Dataset $D = (g, Y)$ where g is the graph with nodes N, labels Y, and edges E, warmup iterations n_{init}, number of datapoints to remove k after iterations i, table to store the prediction score E, percentage reduction in k, x, GCN model M, epochs e

Result: Reduced graph g, trained features X

X= Learnable features for nodes N;
E = array of 0 of size N;
train_mask = Train mask;
val_mask = Val mask;
remain_mask = total nodes;
epoch = 0;
while *epoch* < *e* **do**
    logits = $M(g, remain\_mask, X)$;
    correct_classification = argmax(logits, dim=1) == labels;
    correct_classification[∼(train_mask+val_mask)] = False;
    E += correct_classification;
    loss = nll_loss(logits[train_mask], Y[train_mask]);
    if $e \mod i == 0 \ \& \ e > n_{init}$ **then**
        topk = k nodes in E with maximum score;
        g = $remove\_nodes$(g, topk);
        $remain\_mask$[topk] = False;
        E = array of 0 of size $remain\_mask$;
        $k = int((1 - x) * k)$;
    epoch++;
    loss.backward();
return g, X

4 Datasets

4.1 Synthetic Data

The paper performs experiments on Synthetic dataset, NLP datasets and Computer Vision datasets. Since, the code to the algorithm is not available we try to reproduce the results using the synthetic dataset they provide. The dataset consists of two dimensional

Table 1: Test accuracies of Synthetic data using different models and separations

Separation	Model Type	Accuracy
0.4	Linear	0.8318 (+/-0.05)
0.4	SVM	0.9702 (+/-0.02)
0.6	Linear	0.7492 (+/-0.10)
0.6	SVM	0.9004 (+/-0.04)
0.7	Linear	0.7468 (+/-0.10)
0.7	SVM	0.8766 (+/-0.04)
0.8	Linear	0.7533 (+/-0.10)
0.8	SVM	0.8377 (+/-0.05)

points with concentric circles with different separation divided into two classes. Further Gaussian noise along with flipping of classes is added. The data is shown in Figure 1.

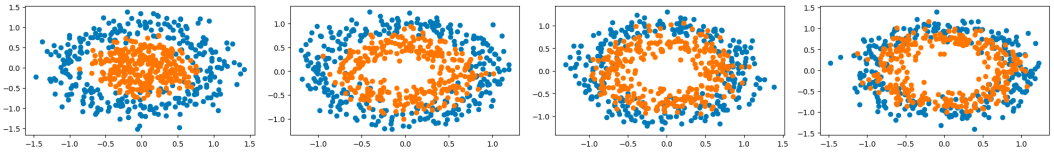


Figure 1: Original Synthetic data with different separations

A Linear and SVM classifier is trained on the dataset and the results are shown in table 1.

4.2 CORA

We use the Cora dataset, a scientific publication dataset, for our experiments. The dataset can be expressed as a citation graph where the nodes represent machine learning publications, and the edges represent the citations between pairs of publications. Note that by construction, the graph is directed since if a paper A cites paper B, it is impossible for paper B to cite paper A. However, an undirected version of the graph is still reasonable because we can consider the citations as a proxy for the similarity between the publications A and B. The task involved is document classification with the objective of categorizing each node into one of the seven classes determining the publication type, i.e., multi-class classification into seven independent classes. The dataset consists of 2,708 scientific publications with 5,429 citation edges. Each publication can be represented as a dense multi-hot bag-of-words feature vector of binary values representing the presence or absence of the corresponding words from a vocabulary of 1,433 unique words. Further, we divided the 2,708 training data points for our experiments into 500 training nodes and keep the remaining nodes for testing.

4.3 GOOD-CORA

The GOOD-Cora dataset is a citation network provided by [2] that consists of nodes representing scientific publications and edges representing citation links adapted from the full Cora dataset [1]. The task is to classify publications into different classes based on their publication type. Two domain selections are used to generate splits in the dataset by the authors in [2]. The first domain selection is based on word diversity, which is defined by the selected-wordcount of a publication and is purely irrelevant to the publication's label. The second domain selection is based on the node degree in the graph, which implies that the popularity of a paper should not determine its class.

In our experiments we use the domain selection by word diversity and covariate shift. The dataset contains 19793 publications with 126842 citation edges. Each publication can be represented as a dense multi-hot bag-of-words feature vector of binary values from a vocabulary of 8,710 unique words. The data has 8,213 training points, 1,979 in distribution validation points, 1,979 in distribution test points, 3,841 out of distribution validation points and 3,781 out of distribution test points.

To conduct our experiments, we employed a subset of the GOOD-CORA dataset, which is a new graph comprising five classes and the same splits as the original dataset for a better visualization. We call this data **GOOD-CORA-SMALL**. The subset contains a total of 1786 nodes with a split of 423, 404, 395, 299 and 265 into each class. Note this distribution consits of train, ID validation and OOD validation sets. The Test ID set has 415 points and Test OOD set has 587 points.

5 Experiments and Results

5.1 AFLite

5.1.1 Synthetic Data

Algorithm AFLite is applied on the synthetic dataset to remove spurious biases. This is done to sanity check the working of our algorithm. The results as in the paper are given in Figure 2, where the reproduced results are shown in Figure 3.

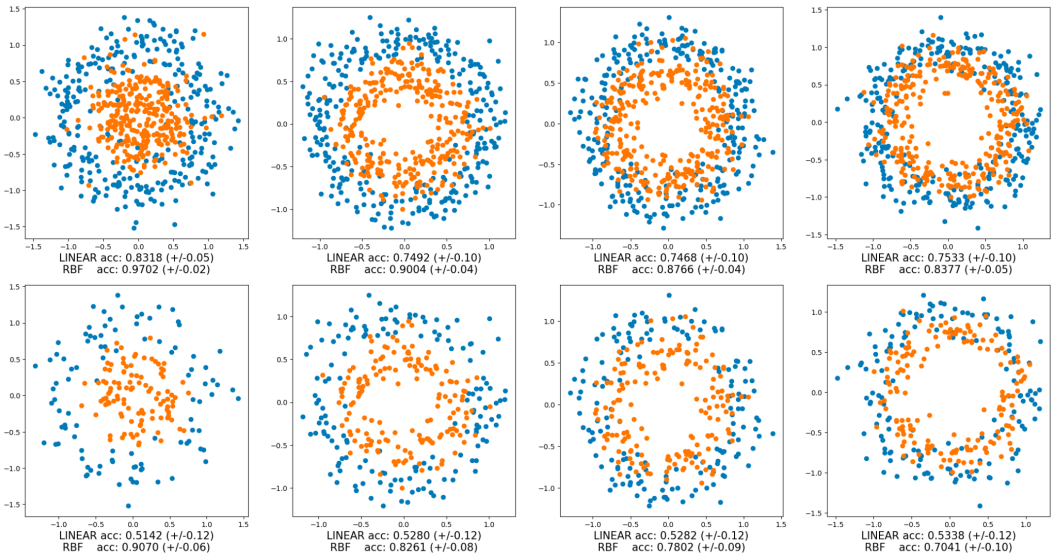


Figure 2: Performance after applying AFLite algorithm as stated in the paper.

5.1.2 Graph CORA dataset

We ran the AFlite algorithm on the CORA dataset over the nodes, using the node features provided by the dataset. We used the hyperparameters as follows m = 128, t = 400, k = 5, τ = 0.75. The algorithm used a Linear Classifier to calculate the predictability score of the node features. We removed a total of 755 nodes out of 2708 from the dataset using the algorithm.

To compare our results in the following experiments we divide the results into three parts:

1. Full data in which we do not remove any nodes, we call this Full Graph.
2. AFLite data in which we remove nodes using AFLite algorithm, we call this AFLite graph.
3. Random data, in which nodes are removed randomly (the same number as in AFLite graph), we call this Random graph.

The node features were visualized by reducing the number of dimensions to two using TSNE [5] algorithm. The projections of features of full graph and aflite graph random are given in the Figure 4a and 4b respectively. We also visualized the points removed by the AFLite graph in the Figure 5. Through the figure we can see that the removed points are tightly bound and thus easy to learn, which is what the AFLite algorithm intended to do.

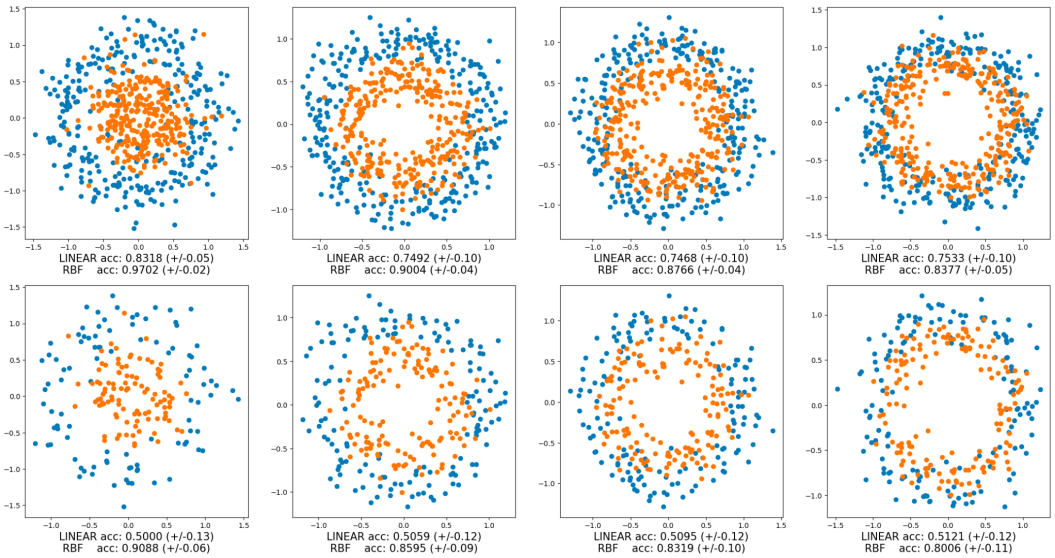
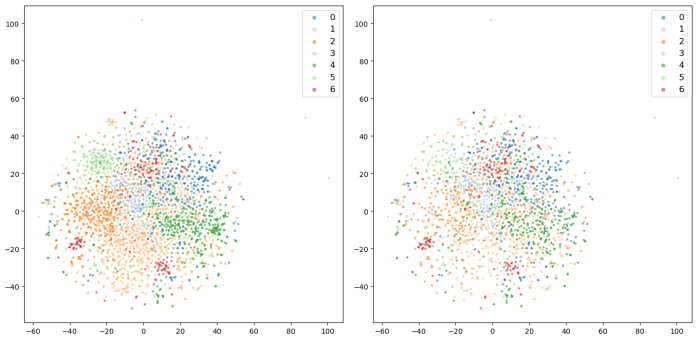


Figure 3: Our results after applying AFLite algorithm to synthetic dataset.



(a) Full Graph (b) AFLite Graph

Figure 4: Visualization of data features of CORA dataset.

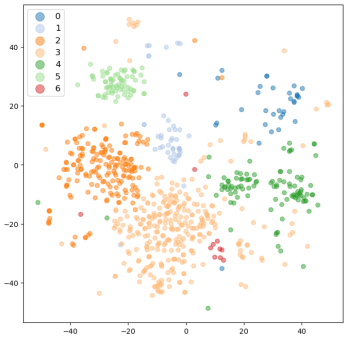
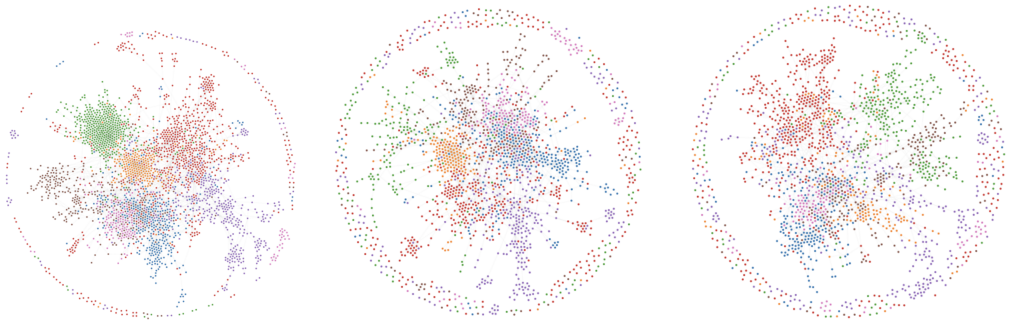


Figure 5: Points removed using AFLite Algorithm on CORA dataset. The removed points are tightly bound (easy to learn examples).

6a, 6b and 6 respectively displays the structural visualization of the Full, AFLite, and Random graphs that we examined. The visualization indicates that the AFLite graph caused significant damage to the well-clustered points.



(a) Ground truth Graph (b) AFLite Graph (c) Random Graph

Figure 6: Visualization of different subsets of CORA dataset Graph. This shows how AFLite has impacted the graph structure.

1. **Performance using Linear Classifier**

   The AFLite algorithm's effectiveness was assessed using a Linear classifier on three types of graphs: the full graph, the AFLite graph, and a random graph. To train the linear classifier, node features were utilized with a maximum iteration count of 1000, and the test data was split into 33%. Table 2 presents the outcomes. As anticipated, the results indicate that the Random Graph performs similarly to the full graph, whereas the AFLite graph has very poor performance. This demonstrates how spurious data points can inflate the model's effectiveness even in graph data, without use of graph structure.

Table 2: Test accuracies of different subsets of CORA on Linear Classifier

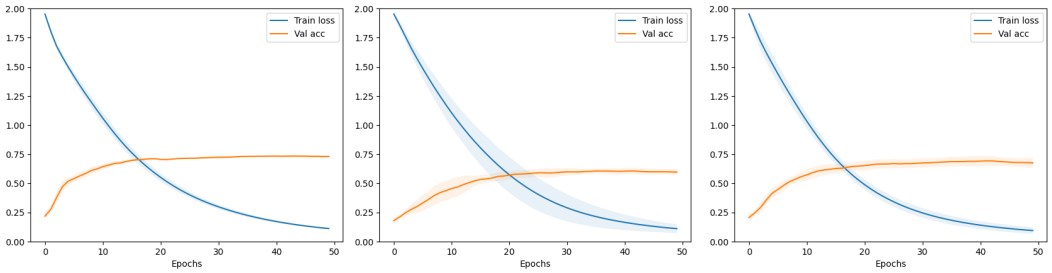
Dataset	Test Accuracy
Full Graph	0.7444 (+/-0.02)
AFLite Graph	0.5994 (+/-0.03)
Random Graph	0.7136 (+/-0.02)

2. **Performance using Graph Convolution Networks**

   To examine how the training process is impacted by graph structure, we utilized the full graph, AFLite graph, and Random graph on the CORA dataset to train a two-layer Graph Convolution Network. We then plotted the training loss and validation accuracy curves for each of the three graphs. The results are shown as in Figure 10a. The testing accuracies are given in Table 3. The outcomes reveal a significant decrease in both the test accuracies and validation accuracies when the AFLite algorithm is utilized to eliminate data points, confirming that the graph structure or interdependence between points has little impact on spurious data.

Table 3: Test accuracies of different subsets of CORA on GCN network

Dataset	Test Accuracy
Full Graph	0.724 (+/-0.008)
AFLite Graph	0.601 (+/-0.013)
Random Graph	0.698 (+/-0.037)



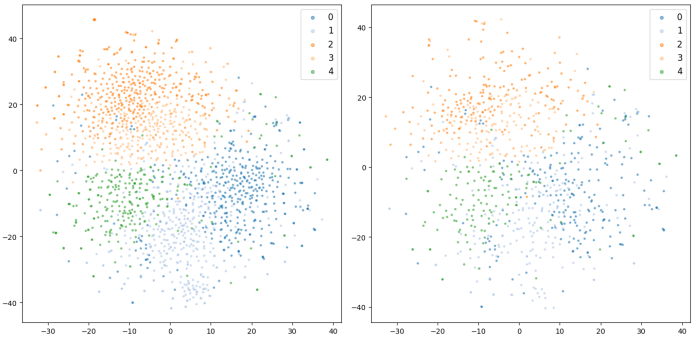
(a) Full Data (b) AFLite Data (c) Random Data

Figure 7: Testing accuracy and training loss curves of CORA dataset. The testing accuracy is lower in AFLite graph as compared to full and random graph.

5.1.3 GOOD-CORA-SMALL dataset

To test the Out of Distribution performance of AFLite algorithm on Graph Data, we use the GOOD-CORA Dataset. We ran the AFlite algorithm on the GOOD-CORA-SMALL dataset over the nodes, using the node features provided by the dataset (This time only on test and validation sets to have consistency in the test set results). We used the hyperparameters as follows m = 128, t = 400, k = 5, τ = 0.75. The algorithm used a Linear Classifier to calculate the predictability score of the node features. Using the algorithm, 860 nodes out of 1786 were eliminated from the dataset. This resulted in a shift in the distribution of points for each class, with 221, 212, 184, 168, and 141 points remaining.

The node features were visualized by reducing the number of dimensions to two using TSNE [5] algorithm. The projections of features of full graph and aflite graph random for GOOD-CORA-SMALL are given in the Figure 4a and 4b respectively. We also visualized the points removed by the AFLite graph in the Figure 9. Through the figure we can see that the removed points are tightly bound and thus easy to learn similar to CORA dataset, which is what the AFLite algorithm intended to do.



(a) Full Graph (b) Aflite Graph

Figure 8: Visualization of data features for GOOD-CORA-SMALL train and validation set

1. **Performance using Linear Classifier**

   The AFLite algorithm's effectiveness was assessed using a Linear classifier on three types of graphs: the full graph, the AFLite graph, and a random graph on GOOD-CORA-SMALL. To train the linear classifier, node features were utilized with a maximum iteration count of 1000, and 33% of data comprising of train and validation sets was split into test set for the classifier. Results are shown in Table 4 presents the outcomes. As anticipated, the results indicate that the Random Graph performs similarly to the full graph, whereas the AFLite graph has very poor performance similar to CORA dataset.

2. **Performance using Graph Convolution Networks**

   We trained a two-layer Graph Convolution Network on the CORA dataset using

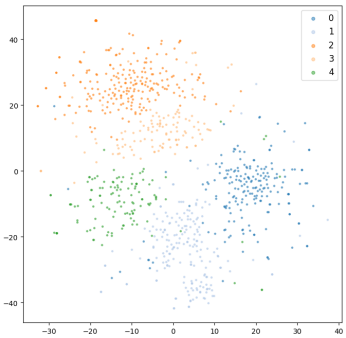


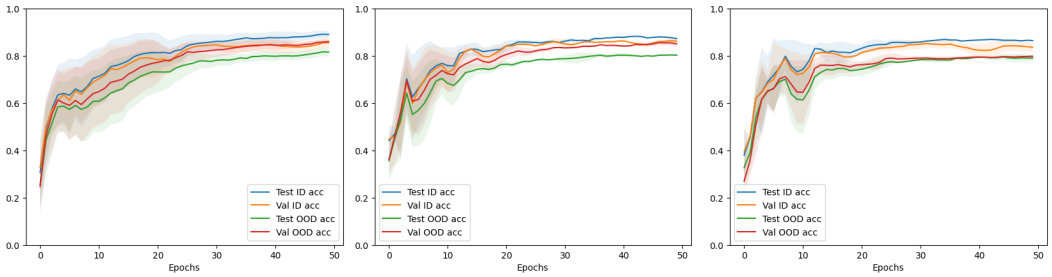
Figure 9: Points removed using AFLite Algorithm on GOOD-CORA-SMALL train and validation set. The removed points are tightly bound (easy to learn examples)

Table 4: Test accuracies of different subsets of GOOD-CORA-SMALL on Linear Classifier

Dataset	Test Accuracy
Full Graph	0.8814
AFLite Graph	0.69
Random Graph	0.85

the full graph, AFLite graph, and Random graph to investigate the influence of graph structure on the training process and the effects of in-distribution and out-of-distribution test sets. All the splits were similar to what was provided by GOOD [2] datasets.

The model was trained using the Train set, and its performance was evaluated on reduced ID and OOD validation sets. The performance was also assessed on the original ID and OOD test sets. Figure 10 illustrates the results obtained, while Table 5 displays the ID and OOD Test Accuracies. The graph depicted in the figure highlights that the Full Graph exhibits high accuracies, while there is a slight decrease in accuracies observed in the AFLite graph, with a reduced gap between ID and OOD accuracies. In contrast, the Random Graph displays greater discrepancies in accuracies. Furthermore, the AFLite Graph demonstrates a smaller gap between ID and OOD as compared to the Random and Full Graphs as seen in Table 5. In terms of OOD Test Accuracy, the AFLite Graph outperforms the Random Graph.



(a) Full Graph (b) AFLite Graph (c) Random Graph

Figure 10: Testing and Validation accuracy curves of GOOD-CORA-SMALL dataset using GCN on different graphs.

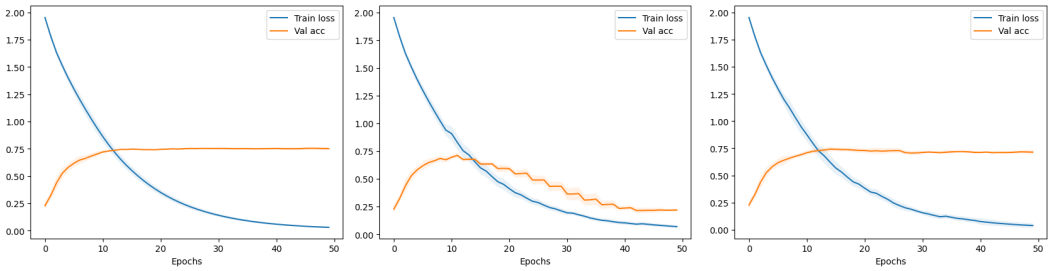
Table 5: ID and OOD Test Accuracies of different subsets of GOOD-CORA-SMALL on GCN

Dataset	OOD Test Accuracy	ID Test Accuracy
Full Graph	0.816 (+/-0.019)	0.890 (+/-0.014)
AFLite Graph	0.803 (+/-0.004)	0.873 (+/-0.003)
Random Graph	0.797 (+/-0.018)	0.871 (+/-0.006)

5.2 Iterative AFLite

5.2.1 CORA

We run the experiments on CORA dataset. The train, val and test sets contain 140, 500 and 1000 points respectively. We set the value if n_{init} as 5, i as 3, k as 60 and x as 0.01. We only used the similar two layer GCN model to remove the data points. The results are shown in Figure 11. The AFLite algorithm removes a total of 52 training points out of 140 and 361 validation points out of 500 on an average. We can see the random and full graph have similar performances in terms of validation accuracy, whereas when we remove points using AFLite algorithm, the AFLite graph has a significant reduction in validation accuracy.



(a) Full Graph (b) AFLite Graph (c) Random Graph

Figure 11: Validation accuracy and training loss curves of CORA dataset using Iterative AFLite. The validation accuracy reduces significantly when points are removed using AFLite algorithm.

The reported test accuracies after this training are given in Table 6. After observing a substantial decrease in validation accuracies, it is noticeable that AFLite Graph's test accuracy surpasses that of a Random Graph. Therefore, it can be concluded that AFLite has eliminated easily learnable spurious points, which were responsible for inflating the model's performance.

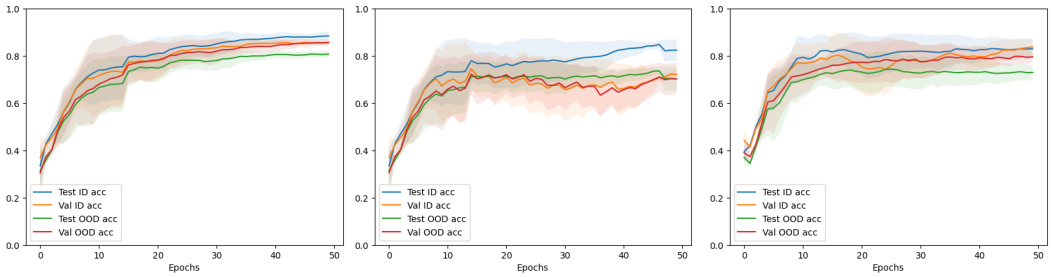
Table 6: Test Accuracies on different algorithms to remove points interatively from CORA

Dataset	Test Accuracy
Full Graph	0.740 (+/-0.011)
AFLite Graph	0.700 (+/-0.016)
Random Graph	0.689 (+/-0.012)

5.2.2 GOOD-CORA-SMALL

In the experiments conducted on the GOOD-CORA-SMALL dataset, there were 1786 points in the training set, 407 points in the ID validation set, and 415 points in the ID test set. In addition, there were 587 points in the OOD validation set and 684 points in the OOD test set. The values of n_{init}, i, k, and x were set to 6, 3, 250, and 0.01, respectively. Only a two-layer GCN

model was used to remove the data points, and the results are presented in Figure 12. On average, the AFLite algorithm eliminated a total of 1146 training points out of 640, 345 OOD validation points out of 587, and 249 ID validation points out of 407. The test performance appears to be quite comparable between the graphs, whereas the validation performance experiences a substantial drop in the AFLite Graph, which is not observed in the Random Graph.



(a) Full Graph (b) AFLite Graph (c) Random Graph

Figure 12: ID and OOD Validation and testing accuracy curves of GOOD-CORA-SMALL dataset using Iterative AFLite.

Table 7 presents the test accuracies reported following this training. The results indicate that AFLite outperforms Random in terms of OOD test sets, which is in line with the findings of earlier results.

Table 7: Test Accuracies on different datasets to remove points iteratively from GOOD-CORA-SMALL

Dataset	OOD Test Accuracy	ID Test Accuracy
Full Graph	0.783 (+/-0.106)	0.857 (+/-0.112)
AFLite Graph	0.751 (+/-0.098)	0.823 (+/-0.111)
Random Graph	0.741 (+/-0.099)	0.841 (+/-0.110)

5.2.3 GOOD-CORA

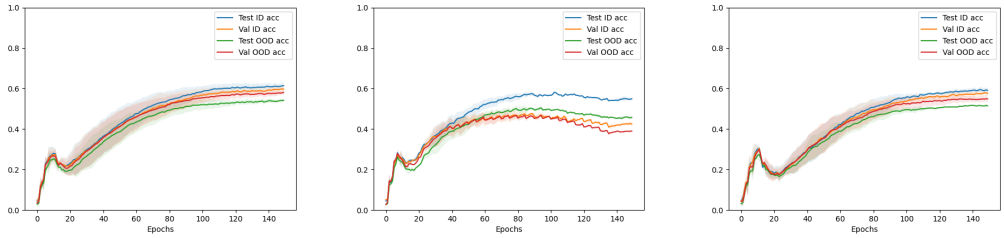
In addition to the experiments conducted on the GOOD-CORA-SMALL dataset, we also performed experiments on the larger GOOD-CORA dataset using the Iterative AFLite algorithm. The train, ID validation, and ID test sets in this dataset consisted of 8213, 1979, and 1979 points, respectively, while the OOD validation and OOD test sets contained 3841 and 3781 points, respectively. For this experiment, we set the values of n_{init}, i, k, and x to 40, 5, 300, and 0.01, respectively, and utilized the same two-layer GCN model to remove the data points. The results of this experiment are presented in Figure 13. On average, the AFLite algorithm removed 3955 training points, 1378 OOD validation points, and 1308 ID validation points.

The graph illustrates that the results obtained are consistent with those obtained on the GOOD-CORA-SMALL dataset. Nevertheless, the decrease in validation accuracies in relation to the AFLite data is more apparent in this case.

After this training, the test accuracies are reported in Table 8. The results demonstrate that, consistent with earlier findings, AFLite outperforms Random regarding OOD test sets. Moreover, in the AFLite graph, there is a noteworthy decrease in the gaps between the ID and OOD test accuracies.

5.3 Iterative AFLite without Features

To execute the Iterative AFLite without Features, we eliminated the features from the CORA dataset. The train, validation, and test sets comprised 140, 500, and 1000 points, respectively, and remained the same as in the previous algorithms. For this experiment, we set the values of n_{init}, i, k, and x to 10, 3, 50, and 0.01, respectively, and utilized the same two-layer GCN model to



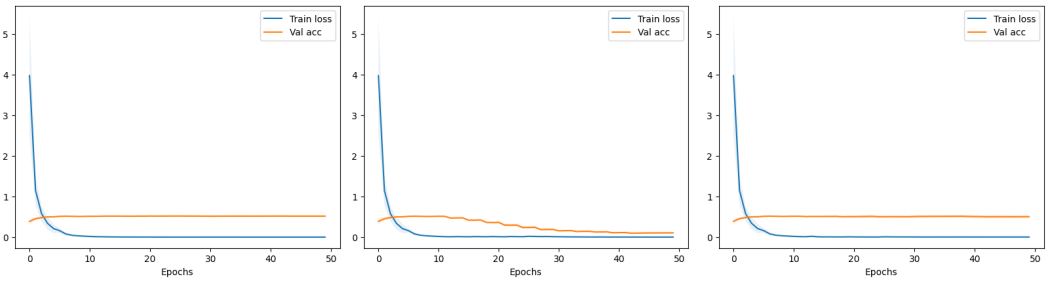
(a) Full Graph (b) AFLite Graph (c) Random Graph

Figure 13: ID and OOD Validation and testing accuracy curves of GOOD-CORA dataset using Iterative AFLite.

Table 8: Test Accuracies on different datasets to remove points iteratively from GOOD-CORA

Dataset	OOD Test Accuracy	ID Test Accuracy
Full Graph	0.545 (+/-0.071)	0.618 (+/-0.005)
AFLite Graph	0.517 (+/-0.091)	0.563 (+/-0.010)
Random Graph	0.519 (+/-0.011)	0.596 (+/-0.009)

remove the data points. The results of this experiment are illustrated in Figure 14. On average, the AFLite algorithm removed 86 training points and 246 validation points. As depicted in the figure, we can observe that the results remain consistent even when learning the features iteratively. Nonetheless, there is a considerable reduction in validation accuracies in the AFLite graph.



(a) Full Graph (b) AFLite Graph (c) Random Graph

Figure 14: Training Loss and Validation accuracy curves of CORA dataset without features using GCN and AFLite iteratively on different graphs. The validation accuracy still decreases if we learn the node features along with removing points from graph data.

The test accuracies are displayed in Table 9. The results reveal that, in line with the previous findings, AFLite still outperforms Random by a small margin.

Table 9: Test Accuracies on different graphs to remove points iteratively from CORA without features

Dataset	Test Accuracy
Full Graph	0.529 (+/-0.001)
AFLite Graph	0.496 (+/-0.026)
Random Graph	0.491 (+/-0.002)

6 Conclusion

In our work, we applied the AFLite algorithm to graph data and observed that it can be used iteratively with state-of-the-art GNN models to remove spurious data points from the data. Our experiments on several datasets showed that removing such points can lead to a significant reduction in accuracies, indicating that these datasets contain spurious biases. We also observed that the AFLite algorithm can be applied iteratively during training, even if we do not have pretrained features.

While we applied our iterative AFLite algorithm to the CORA dataset in this work, we would like to apply it to other datasets in the future to test its generalizability. Additionally, we would like to see how this algorithm performs on link classification tasks, where instead of dropping nodes, we drop the relations in relational graphs. Overall, our work highlights the importance of detecting and removing spurious biases in graph data to ensure the robustness and reliability of GNN models.

7 Acknowledgements

I would like to thank my advisor Yu Yang and Professor Baharan Mirzasoleiman for their invaluable guidance and support throughout this project, without whom this work would not have been possible.

References

[1] A. Bojchevski and S. Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. *arXiv preprint arXiv:1707.03815*, 2017.

[2] S. Gui, X. Li, L. Wang, and S. Ji. Good: A graph out-of-distribution benchmark. *arXiv preprint arXiv:2206.08452*, 2022.

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

[4] R. Le Bras, S. Swayamdipta, C. Bhagavatula, R. Zellers, M. Peters, A. Sabharwal, and Y. Choi. Adversarial filters of dataset biases. In *International Conference on Machine Learning*, pages 1078–1088. PMLR, 2020.

[5] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. In *Journal of Machine Learning Research*, volume 9, pages 2579–2605, 2008.

[6] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*, pages 593–607. Springer, 2018.

[7] S. Swayamdipta. notebooks_for_aflite. `https://github.com/swabhs/notebooks_for_aflite`.

[8] S. Vashishth, S. Sanyal, V. Nitin, and P. Talukdar. Composition-based multi-relational graph convolutional networks. *arXiv preprint arXiv:1911.03082*, 2019.