A Closer Look at Distribution Shifts and Out-of-Distribution Generalization on Graphs

Mucong Ding^{*1}**, Kezhi Kong**^{*1}**, Jiuhai Chen**^{*1,2}**, John Kirchenbauer** ^{1}**, Micah Goldblum** ^{1}**,**
David Wipf ^{2}**, Furong Huang** ^{1}**, Tom Goldstein** ^{1}

^{1}Department of Computer Science, University of Maryland
^{2}AWS Shanghai AI Lab
{mcding, kong, jchen169}@cs.umd.edu

Abstract

Distribution shifts, in which the training distribution differs from the testing distribution, can significantly degrade the performance of Graph Neural Networks (GNNs). We curate GDS, a benchmark of eight datasets reflecting a diverse range of distribution shifts across graphs. We observe that: (1) most domain generalization algorithms fail to work when applied to domain shifts on graphs; and (2) combinations of powerful GNN models and augmentation techniques usually achieve the best out-of-distribution performance. These emphasize the need for domain generalization algorithms tailored for graphs and further graph augmentation techniques that enhance the robustness of predictors.

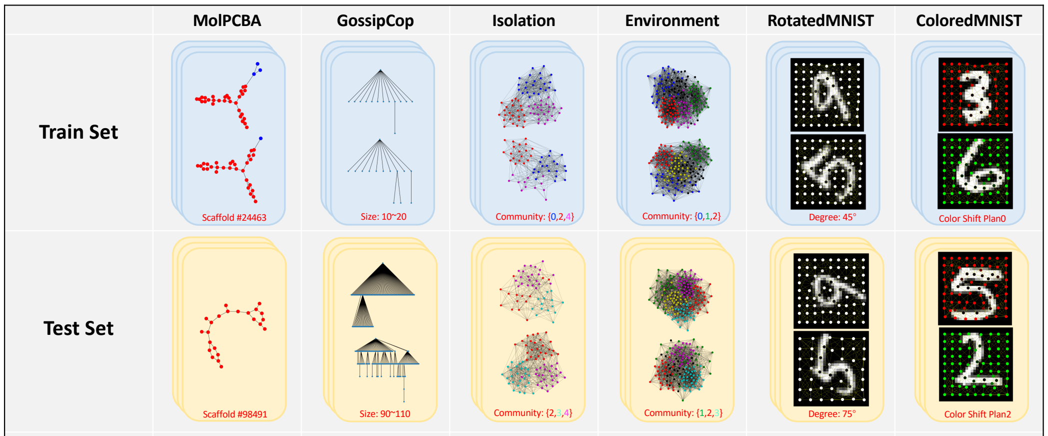
1 Introduction

Distribution shifts, in which training and testing distributions differ, often make machine learning systems fail in spectacular ways [1]. The over-reliance on the training distribution makes it challenging to apply systems in practical scenarios where distribution shifts are common, such as graph classification problems. Graphs are standard data structures that abstract complex systems of interacting objects, such as molecular graphs [2], biological networks [3], and social networks [4]. Distribution shifts arise naturally in many graph classification problems since it is infeasible to prepare a training set that covers all domains of interest. In problems such as drug discovery and social media fact-checking, molecular graph structures often differ at inference [5, 2], and news propagation graphs may grow with time [6].

In this paper, we consider *domain generalization* on graphs (see Appendix A for detailed definition), where training and test graphs are collected from related but different domains. Such domain shifts challenge the *out-of-distribution (OOD)* generalization abilities of graph neural networks (GNNs), as we often see that OOD test performance is usually significantly lower than *in-distribution (ID)* test performance (see Fig. 1). Despite the real-world prevalence of distribution shifts on graphs, they have not been thoroughly evaluated and discussed to the best of our knowledge. Graph learning benchmarks such as [7] note the broad existence of distribution shifts on graphs and have considered using domain information to split training and test sets. We further aim to (1) provide domain labels separately from features which allows the applications of domain generalization algorithms; (2) characterize the types of domain shifts exhibited in the datasets using statistical means; (3) and evaluate what challenges such domain shifts impose on GNN models.

We curate GDS (Graph Distribution Shift), a benchmark of eight datasets reflecting a diverse range of distribution shifts across graphs; see Fig. 1. To facilitate further research, we provide an open-source package that administers the GDS benchmarks with modular combinations of popular

*Equal contribution.



	MolPCBA	GossipCop	Isolation	Environment	RotatedMNIST	ColoredMNIST
ID Generalization	0.338±0.004 (AP)	0.842±0.001 (AUC)	0.648±0.002 (ACC)	0.789±0.001 (ACC)	0.798±0.003 (ACC)	0.688±0.002 (AUC)
OOD Generalization	0.247±0.002 (AP)	0.626±0.008 (AUC)	0.620±0.006 (ACC)	0.769±0.003 (ACC)	0.766±0.003 (ACC)	0.128±0.001 (AUC)

Figure 1: GDS datasets and types of domain shifts, along with the out-of-distribution and in-distribution generalization performance with Graph Isomorphism Network (GIN) and Empirical Risk Minimization (ERM).

Table 1: OOD generalization algorithms, augmentation techniques, and GNN models/techniques considered.

Types	Domain Generalization and Augmentation Algorithms
Baseline	ERM
Distributionally Robust Optimization	GroupDRO [8]
Invariant Risk Minimization	IRM [9]
Distribution Matching	DeepCORAL [10] DANN [11] and DANN-Graph
Meta Learning	MLDG [12]
Augmentation Techniques	FLAG [13] SA

Types	**GNN Models/Techniques**
Baselines	GIN [14] MLP GIN-Deep
Global-Context Techniques	Virtual Node [15]
Spectral Convolution	ChebNet [16]
WL-Isomorphism Hierarchy	3WL-GNN [17]
Structure-Aware Networks	GSN [18]

domain generalization algorithms, graph augmentation techniques, and GNN backbone models. This framework can streamline rigorous and reproducible experiments in the future.

Another major contribution of this work is a rigorous comparison of methods for improving OOD generalization on graphs with respect to GDS. The three high-level categories considered are (1) existing OOD generalization algorithms; (2) augmentations for graph data; and (3) GNN models and techniques that improve OOD generalization; see Table 1. The results of our comprehensive evaluation support two primary conclusions (demonstrated in Tables 2 to 6): (1) most OOD generalization algorithms fail to work when applied to domain shifts on graphs; and (2) combinations of best performing GNN models and augmentation techniques usually achieve the best OOD performance. These observations underscore the need for new OOD generalization algorithms tailored for domains shifts on graphs or further graph augmentation techniques that make the predictor robust to more types of spurious correlations.

2 Graph Distribution Shift (GDS) Datasets

We curate eight datasets reflecting a diverse range of distribution shifts across graphs; see Fig. 1 for illustrations, Table 7 for statistics, and Appendix B for details.

- **Maximum common subgraph: MolHIV and MolPCBA** are two real-world molecular graph datasets adopted from [2, 7]. The task is to predict molecular properties. The domains are defined by scaffolds, the core structures of small molecules [19]. In Fig. 1, we see that molecular graphs within a domain share a maximum common subgraph.

- **Multi-hop neighborhoods: PPA** is a real-world dataset of protein association networks adopted from [3, 7], and the task is to predict which taxonomic group (out of 37 groups) a graph originates from. Each graph in PPA is a subgraph sampled from the 2-hop neighborhood of a protein, where the species of the center protein (1,581 species in total) defines the domain.

- **Size growth: GossipCop** is a real-world dataset consisting of news propagation graphs adopted from [4, 20], which are abstractions of social engagement information (retweets between users).

Table 2: Out-of-distribution generalization performance of GNN models/techniques on GDS datasets.

Types		Baseline		Global	Spectral	WL Isomorphism	Structure
Models	GIN	MLP	GIN-Deep	Virtual Node	ChebNet	3WL-GNN	GSN
Algorithm				ERM			
MolPCBA	0.247±0.002	0.090±0.000	0.253±0.002	**0.285±0.001**	0.232±0.001	OOT	0.233±0.001
MolHIV	0.752±0.016	0.684±0.010	0.758±0.016	0.762±0.007	0.755±0.020	0.720±0.021	**0.778±0.013**
PPA	0.687±0.010	0.095±0.000	0.677±0.002	**0.710±0.004**	0.603±0.007	OOT	OOT
GossipCop	0.626±0.008	0.497±0.010	0.624±0.004	0.499±0.078	**0.831±0.004**	0.784±0.031	0.614±0.011
Isolation	0.620±0.006	0.256±0.003	0.686±0.002	0.690±0.007	0.708±0.004	**0.744±0.009**	OOT
Environment	0.769±0.003	0.335±0.000	0.849±0.012	**0.861±0.006**	0.786±0.008	0.629±0.015	OOT
RotatedMNIST	0.766±0.003	0.333±0.007	0.852±0.003	0.799±0.003	**0.854±0.001**	0.602±0.039	0.780±0.005
ColoredMNIST	0.128±0.001	0.103±0.001	0.129±0.000	0.128±0.000	**0.135±0.000**	0.102±0.000	OOT

Table 3: In-distribution generalization performance of GNN models/techniques on the GDS datasets (where the training, validation, and test sets are randomly sampled from all domains).

Types		Baseline		Global	Spectral	WL Isomorphism	Structure
Models	GIN	MLP	GIN-Deep	Virtual Node	ChebNet	3WL-GNN	GSN
Algorithm				ERM			
MolPCBA	0.338±0.004	0.094±0.002	0.366±0.001	**0.386±0.001**	0.283±0.003	OOT	0.313±0.002
MolHIV	0.787±0.012	0.694±0.002	0.767±0.005	0.795±0.004	**0.805±0.008**	0.769±0.000	0.788±0.004
PPA	0.923±0.007	0.101±0.000	0.912±0.006	**0.929±0.002**	Diverged	OOT	OOT
GossipCop	0.842±0.001	0.493±0.007	0.835±0.001	0.847±0.002	0.882±0.003	**0.893±0.001**	0.838±0.003
Isolation	0.648±0.002	0.253±0.002	0.706±0.000	0.723±0.005	0.724±0.001	**0.741±0.001**	OOT
Environment	0.789±0.001	0.335±0.001	0.878±0.001	**0.896±0.004**	0.810±0.002	OOT	OOT
RotatedMNIST	0.798±0.003	0.315±0.007	0.865±0.001	0.831±0.001	**0.876±0.001**	0.614±0.002	0.815±0.002
ColoredMNIST	0.688±0.002	0.613±0.001	0.697±0.000	0.693±0.002	**0.715±0.000**	0.609±0.004	OOT

Table 4: Performance of domain generalization algorithms and graph augmentation techniques with the Graph Isomorphism Network (GIN) backbone model on the GDS datasets.

Type	Baseline	DRO	IRM	Distribution-Matching			Meta-Learning	Augmentation	
Algorithms	ERM	GroupDRO	IRM	DeepCORAL	DANN	DANN-G	MLDG	FLAG	SA
Model					GIN				
MolPCBA	0.247±0.002	0.211±0.004	0.145±0.002	0.152±0.002	NA	NA	OOT	**0.251±0.004**	0.241±0.001
MolHIV	0.752±0.016	0.735±0.006	0.719±0.013	0.707±0.027	NA	NA	0.650±0.014	0.752±0.003	**0.758±0.005**
PPA	0.687±0.010	0.651±0.011	0.665±0014	0.681±0.016	NA	NA	OOT	**0.712±0.004**	0.702±0.009
GossipCop	0.626±0.008	0.633±0.006	0.636±0.005	0.636±0.012	0.632±0.002	0.633±0.004	**0.643±0.002**	0.635±0.006	0.623±0.005
Isolation	0.620±0.006	0.602±0.002	0.627±0.005	0.621±0.007	0.620±0.005	0.616±0.002	0.626±0.002	**0.633±0.007**	0.621±0.007
Environment	0.769±0.003	0.759±0.001	0.765±0.003	0.763±0.006	0.759±0.003	0.764±0.003	0.757±0.007	**0.778±0.006**	0.776±0.003
RotatedMNIST	0.766±0.003	0.761±0.004	0.771±0.001	0.756±0.002	0.764±0.002	0.765±0.005	0.773±0.005	0.703±0.001	**0.789±0.003**
ColoredMNIST	0.128±0.001	0.126±0.000	0.113±0.002	0.128±0.001	0.127±0.000	0.128±0.000	0.127±0.000	0.125±0.005	**0.129±0.000**

The task is to predict the credibility of news. We split the entire dataset into ten domains according to graph sizes, where each domain corresponds to a decile (every 10% percentile group).

- **Subgraph compositions: ISOLATION and ENVIRONMENT** are two artificial graph datasets generated with Stochastic Block Models (SBMs) [21], where each graph consists of communities, each with a specific intra-community connection probability. In both datasets, the domain is characterized by the composition of communities selected, but their prediction tasks differ. We predict the extra-community connection probabilities in ISOLATION, and the composition with respect to another set of communities in ENVIRONMENT.
- **Structural distortion: ROTATEDMNIST** is a collection of semi-artificial super-pixel graphs converted from the "image Rotated MNIST [22] dataset" using the SLIC [23] pipeline. The task is to classify digits, and the domain is defined by the rotation angle of the underlying image. The rotation of the underlying image leads to a distortion in the super-pixel graph; see Fig. 1.
- **Feature shift only: COLOREDMNIST** is another semi-artificial graph dataset converted from the "image Colored MNIST dataset [9]" using SLIC. The domain affects the true node-feature-to-label correlations while having nothing to do with the graph structures, fooling the algorithms that only learn from the node features.

3 Generalization Algorithms, Augmentation Techniques, and GNN Models

We consider a variety of (1) domain generalization algorithms; (2) graph augmentation techniques; and (3) GNN models/techniques; see Table 1 for the list of algorithms and models considered, Section 3 for discussions, and Appendix C for details

- **Domain generalization algorithms** aim to bias a model towards learning statistical invariances across the training distributions under the assumption that such invariances hold in unseen test

Table 5: Performance of domain generalization algorithms and graph augmentation techniques with the best performing GNN models/techniques on each of the GDS datasets.

Types		Baseline	DRO	IRM	Distribution-Matching			Meta-Learning	Augmentation	
Algorithms	Models	ERM	GroupDRO	IRM	DeepCORAL	DANN	DANN-G	MLDG	FLAG	SA
MolPCBA	Virtual Node	0.285±0.001	0.238±0.001	0.155±0.001	0.161±0.001	NA	NA	OOT	**0.293±0.003**	0.268±0.003
MolHIV	Virtual Node	0.762±0.007	0.729±0.014	0.715±0.005	0.722±0.025	NA	NA	0.693±0.019	0.769±0.017	**0.777±0.022**
PPA	Virtual Node	0.710±0.004	0.692±0.004	0.609±0.004	0.492±0.008	NA	NA	OOT	**0.724±0.002**	0.717±0.001
GossipCop	ChebNet	0.831±0.004	0.829±0.005	0.833±0.005	0.834±0.003	0.837±0.001	**0.838±0.002**	0.826±0.003	0.834±0.004	0.817±0.003
Isolation	ChebNet	0.708±0.004	0.704±0.007	0.713±0.004	0.713±0.006	0.704±0.007	0.704±0.007	0.716±0.005	0.718±0.003	**0.720±0.006**
Environment	Virtual Node	0.861±0.006	0.851±0.002	0.846±0.004	0.862±0.001	0.843±0.006	0.848±0.009	0.798±0.001	**0.887±0.003**	0.861±0.012
RotatedMNIST	ChebNet	0.854±0.001	0.851±0.001	0.857±0.002	0.855±0.002	0.852±0.001	0.852±0.001	**0.864±0.001**	0.854±0.002	0.855±0.001
ColoredMNIST	ChebNet	**0.135±0.000**	0.133±0.001	0.123±0.004	**0.135±0.000**	0.134±0.000	0.134±0.000	0.130±0.000	**0.135±0.001**	0.134±0.000

Table 6: In-distribution and out-of-distribution performance gaps of best combinations of algorithms & models.

Datasets	MolPCBA	MolHIV	PPA	GossipCop	Isolation	Environment	RotatedMNIST	ColoredMNIST
Models	Virtual Node	Virtual Node	Virtual Node	ChebNet	ChebNet	Virtual Node	ChebNet	ChebNet
Algorithms	FLAG	SA	FLAG	DANN-G	SA	FLAG	MLDG	FLAG
In-distribution generalization performance	0.384±0.001	0.814±0.005	0.936±0.001	0.885±0.000	0.726±0.007	0.909±0.001	0.874±0.03	0.706±0.000
Performance gaps (ID − OOD)	0.091±0.003	0.037±0.023	0.212±0.003	0.047±0.002	0.006±0.009	0.022±0.003	0.010±0.003	0.571±0.001

domains. These methods specify both a prior on the types of invariances desired and an algorithm for estimating them from training samples. Since the specific statistical invariances these algorithms learn may be highly dependent on the type of data and the network architectures to which they are applied, each technique's applicability to graph data and GNNs models is unclear.

- **Data augmentation methods** are a standard component in image classification pipelines, but augmentation methods for graph data are not as well studied or prevalent in practice. We consider two types of graph augmentations: (1) augmentations to node features following an adversarial-based augmentation algorithm [13]; (2) and augmentations to graph structures using the node dropping and edge perturbation operations proposed in [24].
- **GNN models and techniques.** The expressive power of the backbone model is also an important factor for generalizing to unseen test data [25]. Some GNN models or techniques may improve OOD generalization performance as they may be robust to certain domain shifts on graphs. For example, ChebNet [16], a spectral graph convolution model, may perform well in terms of OOD generalization when the domain shift is easier learned in the Fourier domain.

4 Experiments

OOD generalization performance of GNN models and techniques. Before evaluating generalization and augmentation algorithms, we want to understand the OOD generalization performance of off-the-shelf GNN models and techniques. Table 2 summarizes the out-of-distribution (OOD) generalization results of all GNN models, and Table 3 shows the corresponding in-distribution (ID) generalization performance. Here, OOT means out of the 24-hour quota of training time. The gap between the ID and OOD generalization performance will then be an indicator of the model's robustness to the domain shift.

- Virtual Node, ChebNet, 3WL-GNN, and GSN outperform baselines (GIN, MLP, and GIN-Deep) both ID and OOD. Virtual Node excels on MolPCBA, PPA, Environment, when long-range correlation is critical or graphs are large (see Table 7 for statistics).
- GSN shows robustness to the domain shifts in MolHIV: it is not the most performant in-distribution but beats other models on the OOD generalization task. Similarly, ChebNet is robust to the shifts in graph sizes on GossipCop.
- Finally, Virtual Node fails on GossipCop where it must generalize to test domains consisting of larger graphs. The reported performance 0.499 ± 0.078 AUC is no better than random guessing (0.5 AUC).

Performance of domain generalization and augmentation algorithms. We then evaluate the six domain generalization algorithms and the two graph augmentation methods; see Table 4.

- Augmentation methods, FLAG, and SA, generally work much better than domain generalization algorithms across the datasets (except for MLDG on GossipCop). This implies many of the domain generalization algorithms proposed for tensor data are not well-suited to graph data and GNNs out-of-the-box.
- On MolPCBA, MolHIV, and PPA, generalization algorithms (GroupDRO, IRM, DeepCORAL, and MLDG) fail consistently compared to the ERM baselines.

- IRM does not outperform the other algorithms on our graph COLOREDMNIST dataset because of the model selection criteria, similar to what was reported in [25].

Combinations of domain generalization algorithms or augmentation techniques with the best performing GNN models. We then evaluate the OOD generalization performance of algorithms combined with the best performing models selected from Table 2 for each dataset; see Table 5. We also report the performance of these combinations when tested in-distribution; see Table 6.

- The differences in performance between domain generalization algorithms in Table 5 are slightly larger than those in Table 4. And by comparing to Table 4, we see DANN-G outperforms MLDG on GOSSIPCOP, and MLDG outperforms SA on ROTATEDMNIST when using ChebNet as the backbone model.

Summary of observations. We summarize our observations from Tables 2 to 6 as follows:

- Most of the domain generalization algorithms fail when applied to graph distribution shift datasets; as shown in Tables 4 and 5. Their performance is usually close to the corresponding ERM baseline and can be even lower on the MOLHIV, MOLPCBA, and PPA datasets.
- The best performance on a dataset is usually achieved by the best performing GNN model with an augmentation technique; see Table 5. This implies that graph augmentations are promising research directions to improve OOD generalization on graphs further.
- Lastly, in Table 6, the gaps between in-distribution and out-of-distribution generalization performance of the best-performing pairs varies across datasets. Most of the gaps are large, especially on PPA, MOLPCBA, and COLOREDMNIST, addressing that there is still large room to improve the OOD generalization on those datasets by designing tailored algorithms to capture their specific statistical invariances.

References

[1] Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *CVPR 2011*, pages 1521–1528. IEEE, 2011.

[2] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

[3] Damian Szklarczyk, Annika L Gable, David Lyon, Alexander Junge, Stefan Wyder, Jaime Huerta-Cepas, Milan Simonovic, Nadezhda T Doncheva, John H Morris, Peer Bork, et al. String v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. *Nucleic acids research*, 47(D1):D607–D613, 2019.

[4] Yingtong Dou, Kai Shu, Congying Xia, Philip S. Yu, and Lichao Sun. User preference-aware fake news detection. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021.

[5] Ricardo Macarron, Martyn N Banks, Dejan Bojanic, David J Burns, Dragan A Cirovic, Tina Garyantes, Darren VS Green, Robert P Hertzberg, William P Janzen, Jeff W Paslay, et al. Impact of high-throughput screening in biomedical research. *Nature reviews Drug discovery*, 10(3):188–195, 2011.

[6] Naeemul Hassan, Gensheng Zhang, Fatma Arslan, Josue Caraballo, Damian Jimenez, Siddhant Gawsane, Shohedul Hasan, Minumol Joseph, Aaditya Kulkarni, Anil Kumar Nayak, et al. Claimbuster: The first-ever end-to-end fact-checking system. *Proceedings of the VLDB Endowment*, 10(12):1945–1948, 2017.

[7] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.

[8] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv preprint arXiv:1911.08731*, 2019.

[9] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

[10] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European conference on computer vision*, pages 443–450. Springer, 2016.

[11] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

[12] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Learning to generalize: Meta-learning for domain generalization. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[13] Kezhi Kong, Guohao Li, Mucong Ding, Zuxuan Wu, Chen Zhu, Bernard Ghanem, Gavin Taylor, and Tom Goldstein. Flag: Adversarial data augmentation for graph neural networks. *arXiv preprint arXiv:2010.09891*, 2020.

[14] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[15] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[16] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29:3844–3852, 2016.

[17] Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. *arXiv preprint arXiv:1905.11136*, 2019.

[18] Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *arXiv preprint arXiv:2006.09252*, 2020.

[19] Guy W Bemis and Mark A Murcko. The properties of known drugs. 1. molecular frameworks. *Journal of medicinal chemistry*, 39(15):2887–2893, 1996.

[20] Kai Shu, Deepak Mahudeswaran, Suhang Wang, Dongwon Lee, and Huan Liu. Fakenewsnet: A data repository with news content, social context, and spatiotemporal information for studying fake news on social media. *Big data*, 8(3):171–188, 2020.

[21] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

[22] Muhammad Ghifary, W Bastiaan Kleijn, Mengjie Zhang, and David Balduzzi. Domain generalization for object recognition with multi-task autoencoders. In *Proceedings of the IEEE international conference on computer vision*, pages 2551–2559, 2015.

[23] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.

[24] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33:5812–5823, 2020.

[25] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. In *International Conference on Learning Representations*, 2020.

[26] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[27] Alberto Sanfeliu and King-Sun Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE transactions on systems, man, and cybernetics*, pages 353–362, 1983.

[28] Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

[29] Pang Wei Koh, Shiori Sagawa, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, pages 5637–5664. PMLR, 2021.

[30] Gilad Yehudai, Ethan Fetaya, Eli Meirom, Gal Chechik, and Haggai Maron. From local structures to size generalization in graph neural networks. In *International Conference on Machine Learning*, pages 11975–11986. PMLR, 2021.

[31] Lirong Wu, Haitao Lin, Zhangyang Gao, Cheng Tan, Stan Li, et al. Graphmixup: Improving class-imbalanced node classification on graphs by self-supervised context prediction. *arXiv preprint arXiv:2106.11133*, 2021.

[32] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

[33] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[34] Junying Li, Deng Cai, and Xiaofei He. Learning graph-level representation for drug discovery. *arXiv preprint arXiv:1709.03741*, 2017.

Supplementary Material

A Problem Definitions

Supervised learning on graphs and graph classification. The goal of supervised graph learning is to train a featurizer $\phi : \mathcal{G} \rightarrow \mathcal{H}$ which maps a graph $G = (V, E)$ with node attributes X_v for $v \in V$ and edge attributes $e_{u,v}$ for $(u, v) \in E$ to a representation vector h_G, which can then be used to predict the label y of graph G through a classifier $\omega : \mathcal{H} \rightarrow \mathcal{Y}$ in graph classification problems. In the supervised learning problem of graph classification, the training dataset $S = \{(G_i, y_i)\}_{i=1}^{n}$ contains i.i.d. samples from the joint probability distribution $P(G, y)$. We choose a predictor $f = \omega \circ \phi$ that minimizes the empirical risk $\frac{1}{n}\sum_{i=1}^{n} \ell(f(G_i), y_i)$ [26], where $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow [0, \infty)$ is the loss function. The ubiquitous choice of hypothesis class for learning graph representations are Graph Neural Networks (GNNs), which apply permutation-equivariant operations, e.g. message passing between connected nodes, and a permutation-invariant graph-level pooling function to all nodes representations to obtain h_G.

Domain generalization on graphs. The problem of domain generalization on graphs is an extension of supervised learning where the data distribution is a mixture of D domains $\{1, \cdots, D\}$, each characterized by a dataset $S^d = \{(G_i^d, y_i^d)\}_{i=1}^{n_d}$ consisting of i.i.d. samples from distribution $P(G^d, y^d)$. We train on the training domains $\{1, \cdots, D_{\text{train}}\}$, and the goal is *out-of-distribution (OOD) generalization* to D_{test} test domains $\{D_{\text{train}} + 1, \cdots, D_{\text{train}} + D_{\text{test}}\}$. The test domains are not accessible during training, which differs from unsupervised domain adaptation, where unlabeled data from test domains are available during training. The sample space of graph structure is high dimensional and discrete, exhibiting a different nature than the Euclidean space, $\mathbb{R}^d$, of conventional vector features. An OOD generalization algorithm that learns the statistical invariances of the training distributions on graphs should also respect permutation symmetries, making the challenging OOD generalization problem even harder on graphs. In the OOD generalization setting, the highly dimensional and discrete nature of graph structure introduces added complexity as compared to structures like 2D grids of continuous pixel values used to represent images. If we consider connected undirected graphs without loops of size k, there are $O(2^{\binom{k}{2}})$ different representations of adjacency matrices $A \in \{0, 1\}^{\binom{n}{2}}$, while there are only $O(2^{\binom{k}{2}}/k!)$ distinct graphs. The fact that a graph structure can be encoded differently implies that the functions acting on graphs should satisfy the permutation invariance given by the permutation group $\mathfrak{G} = \Sigma_k$, i.e., the outcomes of a function on any two isomorphic graphs should be identical. If not mentioned otherwise in this work, we assumed the featurizer ϕ was a permutation-invariant function (i.e., GNN with graph pooling), and the classifier ω a Multi-Layer Perceptron (MLP). However, in other cases, the featurizer ϕ is permutation-equivariant, and the classifier ω is permutation-invariant, where embeddings $\phi(G)$ are also attribute graphs.

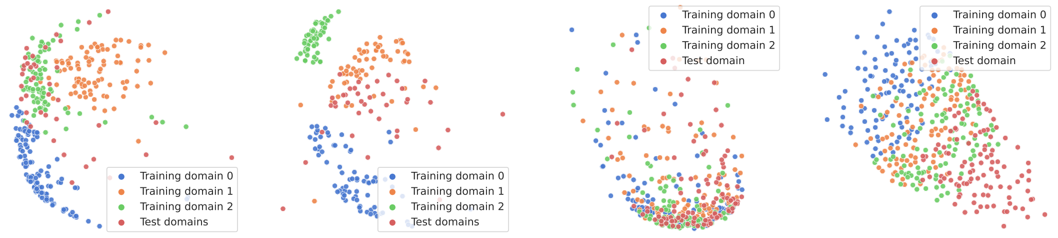
B GDS Datasets

At the heart of our experiments is the design of graph datasets with rich distribution shifts; see Table 7 for more information and Fig. 1 for illustrations. For each dataset, we either prepare or recover the domain information, and as an OOD generalization benchmark, the domain labels d for each sample are included in the datasets.

Table 7: Statistics of the GDS datasets.

Dataset	MolHIV	MolPCBA	PPA	GossipCop	Isolation	Environment	RotatedMNIST	ColoredMNIST
Input	mol-graph	mol-graph	protein-nets	propagation-nets	SBM graphs	SBM graphs	super-pixels	super-pixels
Prediction	mol-properties	mol-properties	protein-prop	fake-news	edge-prob	composition	digits	digits
Domain	scaffold	scaffold	species	size	composition	composition	rotation	color
Metric	AUC	AP	Accuracy	AUC	Accuracy	Accuracy	Accuracy	AUC
Avg. Graph Size	25.5	26.0	243.4	57.5	58.5	147.7	66.0	67.0
# of Domains	19,089	120,084	1,581	10	10	4	6	3
# of Samples	41,127	437,929	158,100	5,464	20,000	60,000	70,000	70,000
Type	real-world Hu et al. [7]	real-world Hu et al. [7]	real-world Hu et al. [7]	semi-artificial Dou et al. [4]	artificial	artificial	semi-artificial Ghifary et al. [22]	semi-artificial Arjovsky et al. [9]

Maximum common subgraph: OGB-MolHIV and OGB-MolPCBA are two real-world molecular graph datasets adopted from Wu et al. [2], Hu et al. [7]. Each graph is an abstraction of a molecule, where nodes are atoms and edges are chemical bonds. The task is to predict molecular properties, e.g., whether the given molecule inhibits HIV virus replication in OGB-MolHIV. The domains are defined by scaffolds, the core structures of small molecules [19], which can be translated



(a) OGB-MolHIV **(b)** OGB-MolPCBA **(c)** OGB-PPA **(d)** UPFD-GossipCop

Figure 2: Scatter plots visualizing the training and test domain distributions of some GDS datasets, using the embeddings learned by a Weisfeiler-Lehman (WL) subtree kernel.

into the language of graph theory as the maximal isomorphic subgraph. In Fig. 1, we see that molecular graphs within a domain share a maximum common subgraph (i.e., the scaffold). They differ from each other only on a few atoms added to different locations of the scaffold. The graph edit distances [27] between graphs in the same domain are very small but can be unbounded across domains. We have many domains in the training, validation, and test splits, as most of the domains contain few samples. Each of the test domains in the two datasets has only a single sample, which imposes yet another challenge to some OOD generalization algorithms such as DANN [11]; We adopt the two molecule datasets from the OGB paper [7]. As the original paper already splits the train, validation, and test sets using domain information (scaffold), we follow this practice to leave the split unchanged.

Multi-hop neighborhoods: OGB-PPA is a real-world dataset of protein association networks adopted from [3, 7]. The graphs in OGB-PPA are abstractions of protein-protein association relations of $1,581$ species, and the task is to predict which taxonomic group (out of 37 groups) a graph originates from. Each graph in OGB-PPA is a subgraph sampled from the 2-hop neighborhood of a protein, where the species of the center protein defines the domain. Thus, graphs within a domain share a common type of center node (center proteins are identical species). Although the center node is always removed in sub-sampling, we believe the sampled neighborhoods around proteins of a specific species exhibit sufficient similarity to be considered as a coherent domain. Similarly, as the OGB-PPAdataset is also adopted from the OGB datasets, we follow the original practice to split the train, validation, and test sets using the species information provided by the dataset.

Size growth: UPFD-GossipCop is a real-world dataset consisting of news propagation graphs adopted from [4, 20]. The propagation networks are abstractions of social engagement information (retweets between users) collected from Twitter, built according to fact-check information from Gossipcop and extracted by FakeNewsNet [20]. Given this propagation network, the task is to predict the credibility of news, either real or fake. We split the entire dataset into ten domains according to graph sizes, where each domain corresponds to a decile (every 10% percentile group). The training and validation sets are randomly selected from the 80% smallest graphs with a split ratio of 6:2, while the test set is the 20% largest graphs. Domains constructed to simulate the scenario of training on propagation graphs filtered to a specific range of sizes but then deploying to real-time fact-checking systems facing an environment with propagation graphs of growing sizes. On the original dataset, we find a multi-layer perceptron (MLP) can achieve 0.948 accuracy, which is even higher than the accuracy of a Graph Isomorphism Network (GIN) with the same number of layers and parameters, 0.930. Thus, we replace all node features (which are user profile information) prepared in Dou et al. [4] with random integers uniformly selected from a vocabulary of size 8, i.e., $\{0, \cdots, 7\}$. We then use the ROC-AUC metric to reflect false positives.

Subgraph compositions: SBM-Isolation and SBM-Environment are two artificial graph datasets generated with Stochastic Block Models (SBMs) [21]. A SBM defines a random graph which modulates the intra- and extra-community connection probabilities. In SBM-Isolation, we randomly select 3 from the 5 total communities $C_1, \cdots, C_5$ for each graph, with intra-edge probability $p_1 = 0.5, \cdots, p_5 = 0.9$ respectively, and we predict the number of communities with relatively smaller extra-edge probability $q' = 0.1$ (with default $q = 0.3$). The domain is characterized by the composition of communities selected, for example, $\{C_1, C_3, C_5\}$. While in SBM-Environment, we define 7 communities $\{C_1, \cdots, C_7\}$ with intra-edge probabilities $p_1 = 0.3, p_2 = 0.4, \cdots, p_7 = 0.9$ and set the extra-edge probability q to 0.1. For each graph, we select 3 communities from $\{C_1, C_3, C_5, C_7\}$ and 2 communities from $\{C_2, C_4, C_6\}$, and we use the former selections to define the domain and the later selections as the prediction task. In one sentence, SBM-ISOLATION and SBM-ENVIRONMENT are similar in their distribution shifts: graphs in different domains consist of distinct community subgraphs, although their prediction tasks differ. When generalizing to unseen test graphs, we have already seen all communities, but we have never seen them appear in the same graph.

Structural distortion: SUPERPIXEL-ROTATEDMNIST is a collection of semi-artificial graphs constructed from images in Rotated MNIST [22] using the super-pixel [23] pipeline. In Rotated MNIST, handwritten-digit images are split into 6 equal folds and rotated counter-clockwisely by $0°, 15°, 30°, 45°, 60°$, and $75°$, respectively. Because super-pixels sampled near image boundaries and on digits are not significantly affected by rotations, the super-pixel graphs converted from images rotated with different angles can be thought of as graphs that undergo some "structure distortion"; see Fig. 1 for visualizations. The task is to classify digits with 1-dimensional node features as the super-pixel intensities. We train and validate on graphs converted from $0°, \cdots, 60°$-rotated images and generalize to unseen $75°$ ones. For the generation process of the dataset, after we attain the images which are rotated for certain degrees, we use the SuperPixel [23] image segmentation algorithm to extract nodes that will represent small regions of homogeneous intensity. Then we construct edges using k-nearest neighbor algorithm (we set k as 8 here), where distances are computed by the 2D coordinates of each node. We set the upper bound of the number of super-pixels to 75 and the compactness of the SuperPixel [23] image segmentation algorithm to 0.25. We adopt the same setting of the SuperPixel algorithm from Dwivedi et al. [28].

Feature shift only: SUPERPIXEL-COLOREDMNIST is another semi-artificial graph dataset constructed using Colored MNIST images [9] using the same super-pixel method as above. In Colored MNIST, the domain affects the true image-to-label correlations, fooling the algorithms that try to learn this mapping directly. When converted to graphs, the super-pixel colors are used as node features, and GNNs which only use the node features will fail on the test graphs, where the feature-to-label correlation is reversed. However, GNNs that only learn the structure-to-label mapping will generalize as if the test graphs are in-distribution. Domain shifts on SUPERPIXEL-COLOREDMNIST only affect node features, making it a feature-shift-only dataset. For the Colored MNIST dataset, we follow the original paper [9] to generate images with color shifts. Totally there are three domains, where the first two form the train and validation sets, and the last one is the test set. Firstly a temporary binary label $\tilde{y}$ is assigned to each image. For digits 0-4, $\tilde{y} = 0$; for digits 5-9, $\tilde{y} = 1$. Secondly, $\tilde{y}$ is randomly flipped with the probability 0.25 to get the label y. Then, we paint red color to images with label $y = 1$ and green to whose $y = 0$. Lastly, the colors are randomly flipped by the probability of 0.1, 0.2, and 0.9 for each domain, respectively. For graph construction, the pipeline is exactly the same as that for the Rotated MNIST. The dataset is constructed this way so that the models that rely heavily on the node features will fail.

In-distribution vs. out-of distribution validation set. An important fact we want to note is that, among the eight datasets we curated, MOLHIV, MOLPCBA, and PPA's validation sets are out-of-distribution (i.e., not part of the training domains), while the rest five datasets' validation set are just randomly sampled from the training domains. We understand that both setups are acceptable. Sampling the validation set from the training domains is just the standard model selection method discussed in [25], while using an out-of-distribution validation set can sometimes improve the OOD test performance as shown in [29]. We did a simple ablation study on MOLHIVand found that using an in-distribution validation set, the OOD test performance of empirical risk minimization with a Graph Isomorphism Network (GIN) is 0.767, a bit higher than the baseline model 0.752 selected using the original out-of-distribution validation set.

Data analytics and visualization. To gain a qualitative understanding of the distribution shifts defined for some of the datasets above, we visualize the domain distributions via scatter plots of embeddings learned with a Weisfeiler-Lehman (WL) subtree kernel of 3 iterations; see Appendix D for details. We see form Fig. 2, the graphs (dots) from different domains (represented by colors) in MOLHIV, MOLPCBA are clearly separated. However, on PPA and GOSSIPCOP, there are still large overlaps between the distributions of graphs from different domains.

C Baseline Algorithms and Models

We implement a collection of OOD generalization algorithms, graph augmentation methods, and GNN models and techniques as shown in Table 1.

OOD generalization algorithms. We implement the following six[2] OOD generalization algorithms:

- Empirical Risk Minimization (**ERM, [26]**) means simultaneously minimizing the loss over all D_{train} datasets ${S^d}_{d=1}^{D_{\text{train}}}$ combined, such that the objective becomes $\frac{1}{D_{\text{train}}} \frac{1}{n_d} \sum_{d=1}^{D_{\text{train}}} \sum_{t=1}^{n_d} \ell(f(G_i^d), y_i^d)$.
- Group Distributionally Robust Optimization (**GroupDRO**, [8]) minimizes the empirical risk on the training set while more heavily weighting domains with larger errors. Let $\mathcal{R}_d = \frac{1}{n_d} \sum_i^{n_d} \ell(f((G_i^d), y_i^d))$ be the empirical risk over a single domain. Group Distributionally Robust Optimization (GroupDRO) for domain generalization then solves $\min\{\max_{d \in D_{\text{train}}} \mathcal{R}_d\}$, minimizing the worst case expected risk for all domains [8].
- Invariant Risk Minimization (**IRM**, [9]) seeks to learn a common parameterization of the linear classifier ω that minimizes the empirical risks in every domain. It seeks to learn a featurizer $\omega : \mathcal{G} \to \mathcal{H}$ and classifier $\phi : \mathcal{H} \to \mathcal{Y}$, composed as predictor $\phi \circ \omega$, such that a single parametrization for ϕ minimizes the empirical loss in each domain, $\mathcal{R}_d(\phi \circ \omega)$ for all domains, $\phi \in \arg\min_\phi \mathcal{R}_d(\phi \circ \omega)$ for all $d \in D_{\text{train}}$
- Correlation Alignment for Deep Domain Adaptation (**DeepCORAL**, [10]) matches the mean and covariance statistics of graph representations $h_G = \phi(G)$ (after graph pooling) from different domains. It penalizes the difference between each domain. Let C_i and C_j denote the feature covariance matrices of domain i and j, that is $C_i = \frac{1}{n_i - 1}(D_i^\top D_i - \frac{1}{n_i}(\mathbf{1}^\top D_i)^\top(\mathbf{1}^\top D_i))$, where n_i is the number of samples and D_i indicates the data example. The CORAL loss is defined as $l = \frac{1}{4d^2}\|C_i - C_j\|_F^2$, d is the number of dimension of data and F is Frobenius norm.
- Domain-Adversarial Neural Networks: (**DANN**, [11]) use an adversarial network to match the distributions of graph representations. During the training procedure, DANN encourages the appearance of features, which are discriminative for the main learning task on the source domains and indiscriminate with respect to the transfer between the domains. As mentioned in Appendix A, we also consider a modified version, **DANN-Graph**, where the featurizer ϕ is a permutation-equivariant GNN without graph pooling, and the classifier ω is a permutation-invariant GNN with graph pooling. We investigate whether performing distribution matching on the graph embeddings before pooling leads to an increase in performance.
- Meta-Learning for Domain Generalization (**MLDG**, [12]) learns how to generalize across domains using the framework of MAML [32]. Meta-learning for Domain Generalization splits the D_{train} domains datasets S^d into subsets *meta-train* and *meta-test*. During training, the loss is first computed over the *meta-train* sets and a parameter update determined, and then the loss is computed over the *meta-test* subsets with respect to model parameters after *meta-train* update is applied. This bi-level optimization is performed using gradient descent, and the final parameters are evaluated on the true test domains.

Augmentation methods. We implement two augmentation methods tailored for graphs:

- Adversarial Augmentation on Graphs (**FLAG**, [13]) iteratively augments node features with gradient-based adversarial perturbations. With image data, rotations and crops are standard examples of semantics preserving transformations, and these types of augmentations are not considered to be domain-specific. However, the graphs we consider are abstracted from diverse fields, and individual nodes, edges, and features are much more semantically relevant than individual pixels in an image - i.e., nodes might represent atoms, and edges represent chemical bonds. Thus, it is unclear what types of augmentation (in features and connective structure) work well for which types of graphs and the domain shifts they are subject to. FLAG is one type of graph adversarial-based augmentation algorithm that happens in the node feature space. Our experimental results show that FLAG is effective, especially when the input node features are discrete categorical features. In this work, we follow the original paper to do three steps of adversarial training. We only carry out a rough hyperparameter search on the step size of gradient ascent from $\{0.01, 0.001\}$, which can make the results suboptimal compared with the original paper.
- Structural Augmentation (**SA**) performs randomized node dropping and edge perturbation to increase the structural diversity of the graphs sampled from the training domains. You et al. [24]

[2]We note the recent development of graph-specific algorithms such as those by Yehudai et al. [30] (generalizing across sizes using a self-supervised learning task) and Wu et al. [31] (a mix-up framework tailored for class-imbalanced node classification). We do not implement them in our initial release of GDS because they are recent papers, and the source code has not been made public.

proposes a graph contrastive learning framework (GCL) for self-supervised pre-training of GNNs based on minimizing the distance in latent space between two augmented views of the same graph according to a contrastive loss (a formulation of the loss proposed in SimCLR [33], but for graph data). The self-supervised pre-training plus supervised fine-tuning scheme they implemented for solving transfer learning tasks was the closest application to the domain generalization present in their evaluation. Inspired by the graph contrastive learning framework (GCL) for self-supervised pre-training of GNNs proposed by You et al. [24], we select two of the *structural augmentation* (SA) types that they use to generate contrastive pairs and simply apply them directly to the training data while performing ERM. We consider *Node dropping* - randomly discarding a certain portion of vertices V along with their connections, and *Edge perturbation* - perturbing the structure of G by randomly adding or dropping a certain ratio of edges under the assumption that the graph semantics are preserved under these transformations and that a model should be robust to variance in the edge connectivity. The choice described in section Appendix C to omit this contrastive pre-training in the main evaluation is motivated by empirical results suggesting that SA with ERM alone is competitive and that GCL does not provide a performance benefit in the domain generalization setting compared to the added computational cost of processing $2N$ augmented graphs per epoch of pre-training.

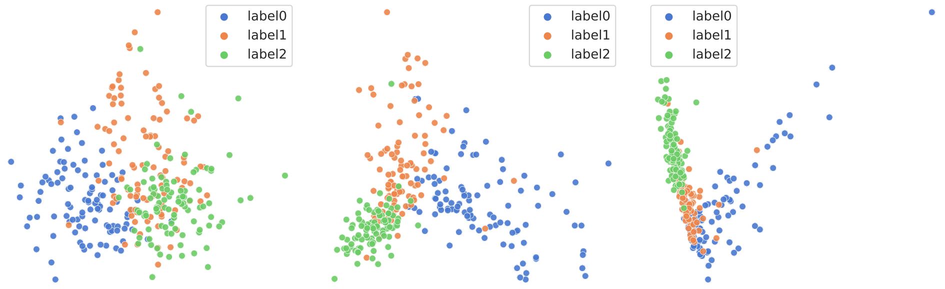
GNN models and techniques. We implement eight GNN models and techniques reflecting a variety of architectural features and inductive biases:

- Standard GNNs: Graph Isomorphism Network (**GIN**, [14]) and a 10-layer version **GIN-Deep**, along with Multi-Layer Perceptron (**MLP**) as the ablation setup without structure learning.
- Global-Context Techniques: **Virtual Node** [15, 34] introduces a "virtual node" that is connected to all the nodes in the graph, which helps to learn correlations at a distance and improves the complexity of graph aggregations.
- Spectral Convolution Methods: **ChebNet** [16] applies spectral filtering in the Fourier domain representations of node features.
- Weisfeiler-Lehman (WL) Expressive Power Hierarchy: **3WL-GNN** [17] enjoys guaranteed 3-WL expressiveness, which is strictly stronger than message passing GNNs.
- Structure-Aware GNNs: **GSN** [18] counts graph substructures isomorphic to some small query graph and enhances the expressive power of message passing GNNs.

D More Experimental Details

D.1 Embedding Visualization

We provide some visualization results on the OOD generalization behavior of the DeepCORAL, DANN, and DANN-G algorithms. The visualization shows the clustering behavior of the final predictions with respect to the predictive labels on the test set. Results are shown in Fig.3. We can see that DANN and DANN-G have better performances than DeepCORAL.



(a) DeepCORAL **(b)** DANN **(c)** DANN-G

Figure 3: Visualization of the projected features by PCA of DeepCORAL, DANN, and DANN-G algorithms on the SBM-ENVIRONMENT dataset.

D.2 Domain Distributions Visualization of GossipCop

ChebNet is robust to the shifts in graph sizes on GossipCop. Considering scatter plots of the GossipCop domain distributions like in Appendix B, but altered where the embeddings are extracted from the graph Laplacian spectrum instead of the WL sub-tree kernel, the comparison between Fig. 4 and Fig. 2d, we see that domain shifts involving graph size may be easier learned in the Fourier representation than the WL kernel space.

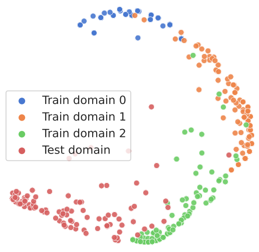


Figure 4: UPFD-GossipCop clustered using embeddings learned from Laplacian spectrums.

D.3 Loss Curves of Virtual Node with ERM on GossipCop

Virtual node fails on GossipCop where it must generalize to test domains consisting of larger graphs. The reported performance 0.499 ± 0.078 AUC is no better than random guessing (0.5 AUC); see the learning curves below in Fig. 5.

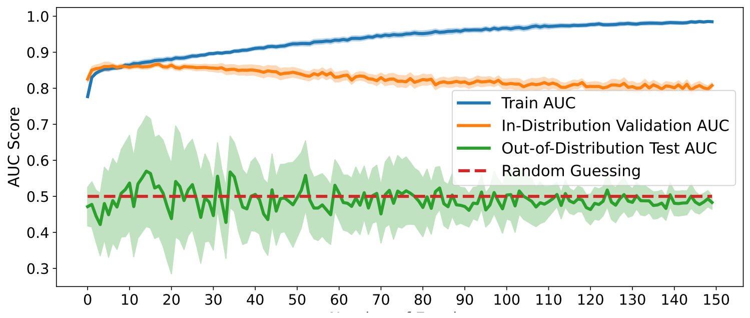


Figure 5: Virtual Node fails catastrophically to generalize OOD on UPFD-GossipCop.

D.4 Hyperparameter Search

We search from the following hyperparameter sets for different algorithms. For each algorithm, we choose at most one algorithm-specific hyperparameter to tune:

- IRM: the weight for IRM penalty loss is chosen from $\{1.0, 100.0\}$, IRM penalty anneals per 500 iterations.
- MLDG: the hyper-parameter beta is chosen from $\{0.1, 1.0, 10.0\}$.
- FLAG: the inner gradient ascent step size for FLAG is chosen from $\{0.01, 0.001\}$.
- SA: the data augmentation ratio for SA is chosen from $\{0.1, 0.2, 0.3, 0.4\}$, the type of data augmentations are randomly chosen from node drop and edge permutation.
- DeepCORAL: Coral penalty weight is chosen from $\{1.0, 10.0\}$.
- DANN: the DANN lambda value is chosen from $\{0.1, 1.0, 10.0\}$.
- DANN-G: similarly, the DANN lambda is chosen from $\{0.1, 1.0, 10.0\}$.
- GSN: the type of subgraph are cycles, and the maximum substructure size is 6.

Tables 8 to 10 present the results for different types of algorithms across different hyper-parameters. We select the top-performing hyper-parameters from Tables 8 to 10. We summarize all the important hyper-parameter selections in Table 11.

D.5 Other experiment details

For all the implementations, we use the Adam optimizer, with a weight decay of 0. We also show the batch size, the number of epochs, learning rate, the number of groups for each batch for every dataset in Table 12. We use GNN base model GIN with five convolutional layers, dropout of 0.5, and the RELU activation function. The dimension of the hidden layer is 300.

Table 8: The results for IRM and Meta-Learning type algorithms with different hyperparameters. Top results are boldfaced.

	IRM		Meta-Learning		
Alogorithms	IRM	IRM	MLDG	MLDG	MLDG
Parameters	1.0	100.0	0.1	1.0	10.0
GIN					
MolPCBA	**0.112**	0.066	NA	NA	NA
MolHIV	**0.689**	0.643	**0.666**	0.615	0.653
PPA	**0.581**	0.443	NA	NA	NA
GossipCop	**0.642**	0.599	**0.64**	0.639	0.638
Isolation	**0.616**	0.547	0.604	**0.638**	0.625
Environment	**0.757**	0.599	0.759	0.751	**0.767**
RotatedMNIST	**0.766**	0.377	**0.779**	0.761	0.768
ColoredMNIST	**0.126**	0.106	**0.128**	0.127	0.128

Table 9: The results for Data Augmentation type algorithms with different hyperparameters. Top results are boldfaced.

Augmentation						
Algorithms	FLAG	FLAG	SA	SA	SA	SA
Parameters	0.001	0.01	0.1	0.2	0.3	0.4
GIN						
MolPCBA	0.251	**0.257**	**0.244**	0.232	0.227	0.227
MolHIV	**0.76**	0.747	0.732	0.78	**0.786**	0.766
PPA	**0.699**	0.695	**0.704**	0.699	0.675	0.688
GossipCop	0.63	**0.632**	**0.62**	0.613	**0.62**	0.614
Isolation	0.616	**0.633**	0.623	0.631	**0.637**	0.616
Environment	0.77	**0.794**	0.772	**0.774**	0.771	0.671
RotatedMNIST	0.695	**0.698**	**0.787**	0.781	0.784	0.782
ColoredMNIST	0.127	**0.126**	**0.129**	0.129	0.129	0.13

Table 10: The results for Distribution Matching type algorithms with different hyperparameters. Top results are boldfaced.

Distribution-Matching								
Algorithms	DeepCORAL	DeepCORAL	DANN	DANN	DANN	DANN-G	DANN-G	DANN-G
Paramerters	1.0	10.0	0.1	1.0	10.0	0.1	1.0	10.0
GIN								
MolPCBA	**0.152**	0.145	NA	NA	NA	NA	NA	NA
MolHIV	**0.722**	0.674	NA	NA	NA	NA	NA	NA
PPA	**0.704**	0.694	NA	NA	NA	NA	NA	NA
GossipCop	**0.639**	0.627	**0.635**	0.631	0.637	**0.638**	0.631	0.578
Isolation	**0.639**	0.613	0.608	**0.611**	0.602	0.605	**0.606**	0.576
Environment	**0.766**	0.761	0.75	**0.755**	0.754	0.755	**0.759**	0.74
RotatedMNIST	**0.755**	0.759	**0.745**	0.74	0.681	**0.742**	0.729	0.684
ColoredMNIST	**0.128**	0.129	**0.126**	0.125	0.125	**0.127**	0.125	0.125

Table 11: The summary of the hyperparameters for different algorithms.

Datasets	MolPCBA	MolHIV	PPA	GossipCop	Isolation	Environment	RotatedMNIST	ColoredMNIST
GroupDRO	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
IRM	1.0	1.0	1.0	1.0	1.0	1.0	1.0	100.0
MLDG	NA	0.1	NA	0.1	1.0	10.0	0.1	0.1
FLAG	0.01	0.001	0.001	0.01	0.01	0.01	0.01	0.01
SA	0.1	0.3	0.1	0.1	0.3	0.2	0.1	0.1
DeepCORAL	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
DANN	NA	NA	NA	0.1	1.0	1.0	0.1	0.1
DANN-G	NA	NA	NA	0.1	1.0	1.0	0.1	0.1

Table 12: More hyperparameters for different datasets.

Datasets	MolPCBA	MolHIV	PPA	GossipCop	Isolation	Environment	RotatedMNIST	ColoredMNIST
Batch Size	128	128	32	128	128	128	128	128
# Epochs	250	200	150	150	150	200	150	100
Learning Rate	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
# Groups Per Batch	4	4	4	4	4	2	4	2