A APPENDIX

A.1 PSEUDOCODE FOR G-ΔUQ

```
class GNN:
    def __init__():
        self.make_layers()
        self.MPNN1 = MPNN(InputDim,HDim1)
        self.MPNN2 = MPNN(HDim1,HDim2)
        self.MPNN3 = MPNN(HDim2, HDim3)
        self.READOUT = READOUTLayer()
        self.classifier = Linear(HDim3, NumClass)

    def forward(self, X,Adj):
        h = self.MPNN1(X,Adj) # Out: (B,N,HDim1)
        h = self.MPNN2(h,Adj) # Out: (B, N,HDim2)
        h = self.MPNN3(h,Adj) # Out: (B,N,HDim3)
        g = self.READOUT(h,Batch) # Out: (B, HDim3)
        pred = self.classifier(g) # Out: (B, NumClass)
        return pred
```

(a) Vanilla GNN

```
class GNN_NodeFeatureAnc:
    def __init__(GNN):
        self.MPNN1 = MPNN(InputDim * 2,HDim1)
        self.MPNN2 = MPNN(HDim1,HDim2)
        self.MPNN3 = MPNN(HDim2, HDim3)
        self.READOUT = READOUTLayer()
        self.classifier = Linear(HDim3, NumClass)
        self.make_anchor_dist(AggNodeFeats)

    def make_anchor_dist(NodeFeats):
        mu = MEAN(AggNodeFeats,dim=0)
        std = STD(AggNodeFeats,dim=0)
        #d-dim Gaussian
        self.AnchorDist = Normal(mean=mu, std=std)

    def forward(X,Adj):
        # (N,InputDim)
        C = self.AnchorDist.sample(X.shape[0])
        # (N,InputDim x 2)
        AncRep = CONCAT([X - C, C], dim=1)
        h = self.MPNN1(AncRep,Adj) #Out: (N,HDim1)
        h = self.MPNN2(h,Adj)
        h = self.MPNN3(h,Adj)
        g = self.READOUT(h,Batch)
        pred = self.classifier(g)
        return pred
```

(b) G-ΔUQ with Node Feature Anchoring

```
class GNN_HiddnRepAnc:
    def __init__(GNN):
        self.MPNN1 = MPNN(InputDim,HDim1)
        # Corres. to GDUQLayer!
        self.MPNN2 = MPNN(HDim1 * 2,HDim2)
        self.MPNN3 = MPNN(HDim2, HDim3)
        self.READOUT = READOUTLayer()
        self.classifier = Linear(HDim3, NumClass)

    def forward(X,Adj):
        h = self.MPNN1(X,Adj)
        C = SHUFFLE(h, dim=0)
        hAnc = CONCAT([h-C, C],dim=1)
        h = self.MPNN2(hAnc,Adj) #Out: (N, HDim2)
        h = self.MPNN3(h,Adj)
        g = self.READOUT(h,Batch)
        pred = self.classifier(g)
        return pred
```

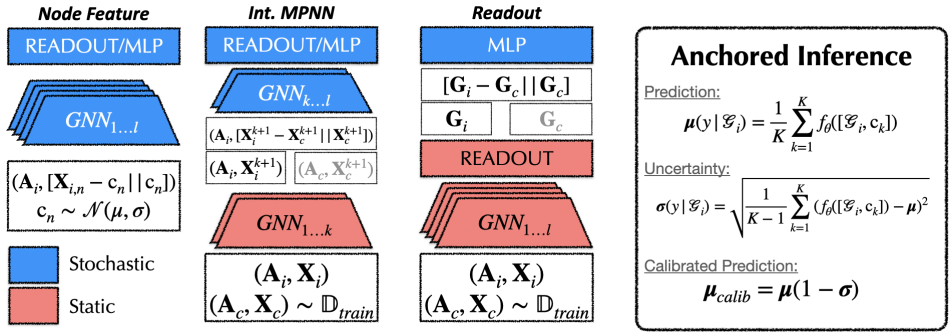
(c) G-ΔUQ with Hidden Rep Anchoring

```
class GNN_ReadOutAnc:
    def __init__(GNN):
        self.MPNN1 = MPNN(InputDim,HDim1)
        self.MPNN2 = MPNN(HDim1, HDim2)
        self.MPNN3 = MPNN(HDim2, HDim3)
        self.READOUT = READOUTLayer()
        #Corres. to GDUQLayer!
        self.classifier = Linear(HDim3 * 2, NumClass)

    def forward(X,Adj):
        h = self.MPNN1(X,Adj)
        h = self.MPNN2(h,Adj)
        h = self.MPNN3(h,Adj)
        g = self.READOUT(h,Batch) #OUT: (B, HDim3)
        C = SHUFFLE(g, dim=0) #(B, HDim3)
        gAnc = CONCAT([g-C,C],dim=1)#(B, HDim3 x 2)
        pred = self.classifier(gAnc) #Out: (B, NumClass)
        return pred
```

(d) G-ΔUQ with READOUT Anchoring

Figure 5: **PseudoCode for G-ΔUQ**. We provide simplified pseudo-code to demonstrate how anchoring can be performed. We assume PyTorchGeometric style mini-batching. Changes with respect to the vanilla GNN are shown in bold. Unchanged lines are grayed out.



Overview of G-ΔUQ models. Here, we present a conceptual overview of how G-ΔUQ induces partially stochastic models. This figure is complementary to 5.

A.2 Ethics Statement

This work proposes a method to improve uncertainty estimation in graph neural networks, which has potential broader societal impacts. As graph learning models are increasingly deployed in real-world applications like healthcare, finance, and transportation, it becomes crucial to ensure these models make reliable predictions and know when they may be wrong. Unreliable models can lead to harmful outcomes if deployed carelessly. By improving uncertainty quantification, our work contributes towards trustworthy graph AI systems.

We also consider several additional safety-critical tasks, including generalization gap prediction for graph classification (to the best of our knowledge, we are the first to report results on this task) and OOD detection. We hope our work will encourage further study in these important areas.

However, there are some limitations. Our method requires (modest) additional computation during training and inference, which increases resource usage. Although G-ΔUQ, unlike post-hoc methods, does not need to be fit on a validation dataset, evaluation of its benefits also also relies on having some out-of-distribution or shifted data available, which may not always be feasible. We have seen in Table 1 that there are tasks for which G-ΔUQfails to improve accuracy and/or calibration of some post-hoc methods, further emphasizing the need to perform appropriate model selection and the risks if shifted validation data is not available. Finally, there are open questions around how much enhancement in uncertainty calibration translates to real-world safety and performance gains.

Looking ahead, we believe improving uncertainty estimates is an important direction for graph neural networks and deep learning more broadly. This will enable the development safe, reliable AI that benefits society. We hope our work inspires more research in the graph domain that focuses on uncertainty quantification and techniques that provide guarantees about model behavior, especially for safety-critical applications. Continued progress will require interdisciplinary collaboration between graph machine learning researchers and domain experts in areas where models are deployed.

A.3 Reproducibility

For reproducing our experiments, we have made our code available at this anonymous repository. In the remainder of this appendix (specifically App. A.6, A.7), and A.10), we also provide additional details about the benchmarks and experimental setup.

A.4 Details on Super-pixel Experiments

We provide an example of the rotated images and corresponding super-pixel graphs in Fig. 6. (Note that classes "6" and "9" may be confused under severe distribution shift, i.e. 90 degrees rotation or more. Hence, to avoid harming class information, our experiments only consider distribution shift from rotation up to 40 degrees.)

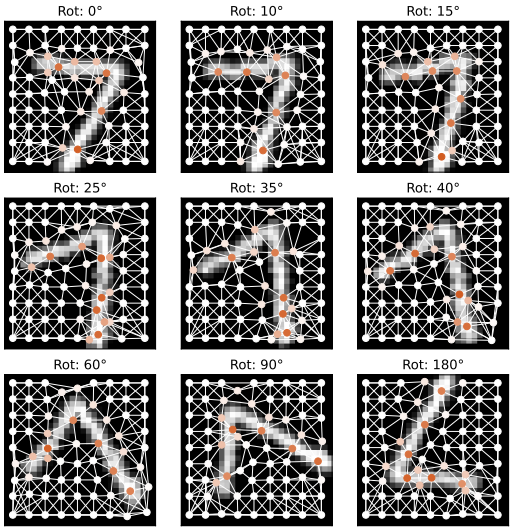


Figure 6: **Rotated Super-pixel MNIST.** Rotating images prior to creating super-pixels to leads to some structural distortion (Ding et al., 2021). However, we can see that the class-discriminative information is preserved, despite rotation. This allows for simulating different levels of graph structure distribution shifts, while still ensuring that samples are valid.

Tables 4 and 5 provided expanded results on the rotated image super-pixel graph classification task, discussed in Sec. 6.1.

In Table 7 we focus on the calibration results on this task for GPS variants alone. Across all levels of distribution shift, the best method is our strategy for applying G-ΔUQ to a pretrained model– demonstrating that this is not just a practical choice when it is infeasible to retrain a model, but can lead to powerful performance by any measure. Second-best on all datasets is applying G-ΔUQ during training, further highlighting the benefits of stochastic anchoring.

In addition to the structural distribution shifts we get by rotating the images before constructing super-pixel graphs, we also simulate feature distribution shifts by adding Gaussian noise with different standard deviations to the pixel value node features in the super-pixel graphs. In Table 8, we report accuracy and calibration results for varying levels of distribution shift (represented by the size of the standard deviation of the Gaussian noise). Across different levels of feature distribution shift, we also see that G-ΔUQ results in superior calibration, while maintaining competitive or in many cases superior accuracy.

Table 4: **RotMNIST-Accuracy.** Here, we report expanded results (accuracy) on the Rotated MNIST dataset, including a variant that combines G-ΔUQ with Deep Ens. Notably, we see that anchored ensembles outperform basic ensembles in both accuracy and calibration. (Best results for models using Deep Ens. and those not using it marked separately.)

MODEL	G-ΔUQ?	LPE?	Avg. Test (↑)	Acc. (10) (↑)	Acc. (15) (↑)	Acc. (25) (↑)	Acc. (35) (↑)	Acc. (40) (↑)
GatedGCN	✗	✗	0.947 ±0.002	0.918 ±0.002	0.904 ±0.005	0.828 ±0.009	0.738 ±0.009	0.679 ±0.007
	✓	✗	0.933 ±0.015	0.894 ±0.019	0.878 ±0.020	0.794 ±0.032	0.698 ±0.036	0.636 ±0.048
	✗	✓	0.949 ±0.002	0.917 ±0.004	0.904 ±0.005	0.829 ±0.007	0.744 ±0.007	0.685 ±0.006
	✓	✓	0.915 ±0.032	0.872 ±0.038	0.852 ±0.0414	0.776 ±0.039	0.680 ±0.037	0.631 ±0.033
GPS	✗	✓	**0.970** ±0.001	**0.948** ±0.001	**0.938** ±0.001	**0.873** ±0.006	**0.770** ±0.013	**0.688** ±0.009
	✓	✓	0.969 ±0.001	0.946 ±0.003	0.937 ±0.003	0.869 ±0.003	0.769 ±0.012	0.679 ±0.014
GPS (Pretrained)	✓	✓	0.967 ±0.002	0.945 ±0.004	0.934 ±0.005	0.864 ±0.009	0.759 ±0.010	0.674 ±0.002
GatedGCN-DENS	✗	✗	0.963 ±0.0002	0.943 ±0.001	0.933 ±0.001	0.874 ±0.002	0.794 ±0.002	0.731 ±0.002
	✓	✗	0.949 ±0.008	0.922 ±0.008	0.907 ±0.011	0.828 ±0.020	0.733 ±0.032	0.662 ±0.046
	✗	✓	0.965 ±0.001	0.943 ±0.001	0.933 ±0.001	0.873 ±0.001	0.792 ±0.004	0.736 ±0.003
	✓	✓	0.954 ±0.005	0.930 ±0.010	0.917 ±0.011	0.850 ±0.023	0.759 ±0.025	0.696 ±0.032
GPS-DENS	✗	✓	**0.980** ±0.000	**0.969** ±0.000	**0.961** ±0.000	**0.913** ±0.000	**0.834** ±0.000	**0.750** ±0.000
	✓	✓	0.978 ±0.001	0.963 ±0.000	0.953 ±0.001	0.905 ±0.000	0.822 ±0.002	0.736 ±0.003

Table 5: **RotMNIST-Calibration.** Here, we report expanded results (calibration) on the Rotated MNIST dataset, including a variant that combines G-ΔUQ with Deep Ens. Notably, we see that anchored ensembles outperform basic ensembles in both accuracy and calibration. (Best results for models using Deep Ens. and those not using it marked separately.)

MODEL	G-ΔUQ	LPE?	Avg.ECE (↓)	ECE (10) (↓)	ECE (15) (↓)	ECE (25) (↓)	ECE (35) (↓)	ECE (40) (↓)
GatedGCN-TS	✗	✗	0.035 ±0.001	0.054 ±0.002	0.062 ±0.003	0.118 ±0.007	0.185 ±0.006	0.233 ±0.008
	✗	✓	0.033 ±0.002	0.053 ±0.002	0.061 ±0.004	0.116 ±0.005	0.179 ±0.006	0.225 ±0.005
GatedGCN	✗	✗	0.038 ±0.001	0.059 ±0.001	0.068 ±0.340	0.126 ±0.008	0.195 ±0.012	0.245 ±0.011
	✓	✗	**0.018** ±0.008	0.029 ±0.013	**0.033** ±0.164	0.069 ±0.033	0.117 ±0.048	0.162 ±0.067
	✗	✓	0.036 ±0.003	0.059 ±0.002	0.068 ±0.340	0.125 ±0.006	0.191 ±0.007	0.240 ±0.008
	✓	✓	0.022 ±0.007	**0.028** ±0.014	0.034 ±0.169	**0.062** ±0.022	**0.109** ±0.019	**0.141** ±0.019
GPS-TS	✗	✓	0.024 ±0.001	0.041 ±0.001	0.049 ±0.001	0.102 ±0.006	0.188 ±0.012	0.261 ±0.008
GPS	✗	✓	0.026 ±0.001	0.044 ±0.001	0.052 ±0.156	0.108 ±0.006	0.197 ±0.012	0.273 ±0.008
	✓	✓	0.022 ±0.001	0.037 ±0.005	0.044 ±0.133	0.091 ±0.008	0.165 ±0.018	0.239 ±0.018
GPS (Pretrained)	✓	✓	0.021 ±0.001	0.032 ±0.003	0.039 ±0.116	0.083 ±0.002	0.153 ±0.007	0.217 ±0.012
GatedGCN-DENS	✗	✗	0.026 ±0.000	0.038 ±0.001	0.042 ±0.001	0.084 ±0.002	0.135 ±0.001	0.185 ±0.003
	✓	✗	**0.014** ±0.003	**0.018** ±0.005	**0.021** ±0.005	0.036 ±0.012	0.069 ±0.032	0.114 ±0.056
	✗	✓	0.024 ±0.001	0.038 ±0.001	0.043 ±0.002	0.083 ±0.001	0.139 ±0.004	0.181 ±0.002
	✓	✓	0.017 ±0.002	0.024 ±0.005	0.027 ±0.008	**0.030** ±0.004	**0.036** ±0.012	**0.059** ±0.033
GPS-DENS	✗	✓	0.016 ±0.001	0.026 ±0.002	0.030 ±0.000	0.066 ±0.000	0.123 ±0.000	0.195 ±0.000
	✓	✓	**0.014** ±0.000	0.023 ±0.002	0.027 ±0.003	0.055 ±0.004	0.103 ±0.006	0.164 ±0.006

Table 6: **Accuracy of GPS Variants on RotatedMNIST.** We focus on the accuracy results for GPS variants on rotated MNIST dataset. Using G-ΔUQ (with or without pretraining) remains close in accuracy to foregoing it, generally within the range of the standard deviation of the results.

MODEL	G-ΔUQ?	Avg. Test (↑)	Acc. (10) (↑)	Acc. (15) (↑)	Acc. (25) (↑)	Acc. (35) (↑)	Acc. (40) (↑)
GPS	✗	**0.970** ±0.001	**0.948** ±0.001	**0.938** ±0.001	**0.873** ±0.006	**0.770** ±0.013	**0.688** ±0.009
	✓	0.969 ±0.001	0.946 ±0.003	0.937 ±0.003	0.869 ±0.003	0.769 ±0.012	0.679 ±0.014
GPS (Pretrained)	✓	0.967 ±0.002	0.945 ±0.004	0.934 ±0.005	0.864 ±0.009	0.759 ±0.010	0.674 ±0.002

Table 7: **Calibration of GPS Variants on RotatedMNIST.** We focus on the calibration results for GPS variants on rotated MNIST dataset. Across the board, we see improvements from using G-ΔUQ , with our strategy of applying it to a pretrained model doing best.

MODEL	G-ΔUQ	Avg.ECE (↓)	ECE (10) (↓)	ECE (15) (↓)	ECE (25) (↓)	ECE (35) (↓)	ECE (40) (↓)
GPS-TS	✗	0.024 ±0.001	0.041 ±0.001	0.049 ±0.001	0.102 ±0.006	0.188 ±0.012	0.261 ±0.008
GPS	✗	0.026 ±0.001	0.044 ±0.001	0.052 ±0.156	0.108 ±0.006	0.197 ±0.012	0.273 ±0.008
	✓	0.022 ±0.001	0.037 ±0.005	0.044 ±0.133	0.091 ±0.008	0.165 ±0.018	0.239 ±0.018
GPS (Pretrained)	✓	**0.021** ±0.001	**0.032** ±0.003	**0.039** ±0.116	**0.083** ±0.002	**0.153** ±0.007	**0.217** ±0.012

Table 8: **MNIST Feature Shifts**. G-ΔUQ improves calibration and maintains competitive or even improved accuracy across varying levels of feature distribution shift.

				STD = 0.1		STD = 0.2		STD = 0.3		STD = 0.4	
MODEL	LPE?	G-ΔUQ?	Calibration	Accuracy (↑)	ECE (↓)	Accuracy (↑)	ECE (↓)	Accuracy (↑)	ECE (↓)	Accuracy (↑)	ECE (↓)
GatedGCN	✗	✗	✗	0.742±0.005	0.186±0.018	0.481±0.015	0.414±0.092	0.293±0.074	0.606±0.147	0.197±0.092	0.71±0.178
	✗	✓	✗	**0.773**±0.053	**0.075**±0.032	0.536±0.010	**0.160**±0.087	**0.356**±0.101	0.422±0.083	**0.249**±0.074	**0.529**±0.047
	✓	✗	✗	0.751±0.02	0.176±0.014	0.519±0.004	0.348±0.03	0.345±0.032	0.485±0.096	0.233±0.043	0.581±0.142
	✓	✓	✗	0.745±0.026	0.100±0.036	**0.541**±0.040	0.235±0.067	**0.355**±0.062	**0.408**±0.116	0.242±0.063	0.539±0.139

A.5 Stochastic Centering on the Empirical NTK of Graph Neural Networks

Using a simple grid-graph dataset and 4 layer GIN model, we compute the Fourier spectrum of the NTK. As shown in Fig. 7, we find that shifts to the node features can induce systematic changes to the spectrum.

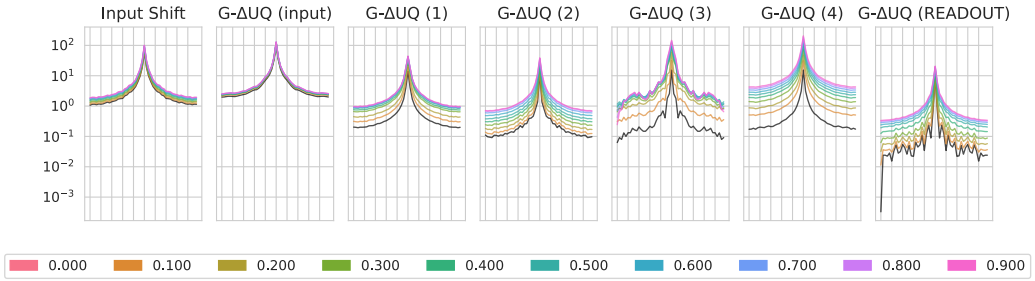


Figure 7: **Stochastic Centering with the empirical GNN NTK.** We find that performing constant shifts at intermediate layers introduces changes to a GNN's NTK. We include a vanilla GNN NTK in black for reference. Further, note the shape of the spectrum should not be compared across subplots as each subplot was created with a different random initialization.

A.6 SIZE-GENERALIZATION DATASET STATISTICS

The statistics for the size generalization experiments (see Sec. 5.1) are provided below in Table 9.

Table 9: **Size Generalization Dataset Statistics:** This table is directly reproduced from (Buffelli et al., 2022), who in turn used statistics from (Yehudai et al., 2021; Bevilacqua et al., 2021).

	NCI1			NCI109		
	ALL	SMALLEST 50%	LARGEST 10%	ALL	SMALLEST 50%	LARGEST 10%
CLASS A	49.95%	62.30%	19.17%	49.62%	62.04%	21.37%
CLASS B	50.04%	37.69%	80.82%	50.37%	37.95%	78.62%
# OF GRAPHS	4110	2157	412	4127	2079	421
AVG GRAPH SIZE	29	20	61	29	20	61

	PROTEINS			DD		
	ALL	SMALLEST 50%	LARGEST 10%	ALL	SMALLEST 50%	LARGEST 10%
CLASS A	59.56%	41.97%	90.17%	58.65%	35.47%	79.66%
CLASS B	40.43%	58.02%	9.82%	41.34%	64.52%	20.33%
# OF GRAPHS	1113	567	112	1178	592	118
AVG GRAPH SIZE	39	15	138	284	144	746

A.7 GOOD BENCHMARK EXPERIMENTAL DETAILS

For our experiments in Sec. 5.2, we utilize the in/out-of-distribution covariate and concept splits provided by Gui et al. (2022). Furthermore, we use the suggested models and architectures provided by their package. In brief, we use GIN models with virtual nodes (except for GOODMotif) for training, and average scores over 3 seeds. When performing stochastic anchoring at a particular layer, we double the hidden representation size for that layer. Subsequent layers retain the original size of the vanilla model.

When performing stochastic anchoring, we use 10 fixed anchors randomly drawn from the in-distribution validation dataset. We also train the G-ΔUQ for an additional 50 epochs to ensure that models are able to converge. Please see our code repository for the full details.

We also include results on additional node classification benchmarks featuring distribution shift in Table 12. In Table 14, we compare models without G-ΔUQ to the use of G-ΔUQ with randomly sampled anchors at the first or second layer.

Dataset	Shift	Train	ID validation	ID test	OOD validation	OOD test	Train	OOD validation	ID validation	ID test	OOD test
		Length									
GOOD-SST2	covariate	24744	5301	5301	17206	17490					
	concept	27270	5843	5843	15142	15944					
		Color									
GOOD-CMNIST	covariate	42000	7000	7000	7000	7000					
	concept	29400	6300	6300	14000	14000					
	no shift	42000	14000	14000	-	-					
		Base					Size				
GOOD-Motif	covariate	18000	3000	3000	3000	3000	18000	3000	3000	3000	3000
	concept	12600	2700	2700	6000	6000	12600	2700	2700	6000	6000
		Word					Degree				
GOOD-Cora	covariate	9378	1979	1979	3003	3454	8213	1979	1979	3841	3781
	concept	7273	1558	1558	3807	5597	7281	1560	1560	3706	5686
		University									
GOOD-WebKB	covariate	244	61	61	125	126					
	concept	282	60	60	106	109					
		Color									
GOOD-CBAS	covariate	420	70	70	70	70					
	concept	140	140	140	140	140					

Table 10: Number of Graphs/Nodes per dataset.

Dataset	Model	# Model layers	Batch Size	# Max Epochs	# Iterations per epoch	Initial LR	Node Feature Dim
GOOD-SST2	GIN-Virtual	3	32	200/100	–	1e-3	768
GOOD-CMNIST	GIN-Virtual	5	128	500	–	1e-3	3
GOOD-Motif	GIN	3	32	200	–	1e-3	4
GOOD-Cora	GCN	3	4096	100	10	1e-3	8710
GOOD-WebKB	GCN	3	4096	100	10	1e-3/5e-3	1703
GOOD-CBAS	GCN	3	1000	200	10	3e-3	8

Table 11: Model and hyperparameters for GOOD datasets.

A.8 GOOD DATASET ADDITIONAL RESULTS

We also include results on additional node classification benchmarks featuring distribution shift in Table 12. In Table 14, we compare models without G-ΔUQto the use of G-ΔUQwith randomly sampled anchors at the first or second layer.

Table 12: **Additional Node Classification Benchmarks.** Here, we compare accuracy and calibration error of G-ΔUQ and "no G-ΔUQ " (vanilla) models on 4 node classification benchmarks across concept and covariate shifts. First, we note that across all our evaluations, without any posthoc calibration, G-ΔUQ is superior to the vanilla model on nearly every benchmark for better or same accuracy (8/8 benchmarks) and better calibration error (7/8), often with a significant gain in calibration performance. However, due to the challenging nature of these shifts, achieving state-of-the-art calibration performance often requires the use of post-hoc calibration methods – so we also evaluate how these posthoc methods can be elevated when combined with G-ΔUQ (versus the vanilla variant). When combined with popular posthoc methods, we highlight that performance improves across the board, when combined with G-ΔUQ (including in WebKB and CBAS-Concept). For example, on WebKB – across the 9 calibration methods considered, "G-ΔUQ + calibration method" improves or maintains the calibration performance of the analogous "no G-ΔUQ + calibration method" in 7/9 (concept) and 6/9 (covariate). In CBAS, calibration is improved or maintained as the no-G-ΔUQ version on 5/9 (concept) and 9/9 (covariate). In all cases, this is achieved with little or no compromise on classification accuracy (often improving over "no G-ΔUQ" variant). We also emphasize that, across all the 8 evaluation sets (4 datasets x 2 shift types) in Table 10, the best performance is almost always obtained with a GDUQ variant: (accuracy: 8/8) as well as best calibration (6/8) or second best (2/8).

			Shift: Concept				Shift: Covariate			
			Accuracy (↑)		ECE (↓)		Accuracy (↑)		ECE (↓)	
Dataset	Domain	Calibration	No G-Δ UQ	G-Δ UQ	No G-Δ UQ	G-Δ UQ	No G-Δ UQ	G-Δ UQ	No G-Δ UQ	G-Δ UQ
WebKB	University	×	0.253±0.003	**0.281**±0.009	0.67±0.061	0.593±0.025	0.122±0.029	0.115±0.041	0.599±0.091	0.525±0.033
		CAGCN	0.253±0.005	0.268±0.008	0.452±0.14	0.473±0.12	0.122±0.018	0.092±0.161	0.355±0.227	0.396±0.161
		Dirichlet	0.229±0.018	0.22±0.022	0.472±0.06	0.472±0.03	0.244±0.105	**0.295**±0.044	**0.299**±0.092	0.328±0.044
		ETS	0.253±0.005	0.273±0.012	0.64±0.06	0.575±0.019	0.121±0.021	0.084±0.027	0.539±0.112	0.499±0.027
		GATS	0.253±0.005	0.273±0.01	0.608±0.008	0.485±0.02	0.122±0.018	0.079±0.029	0.455±0.057	0.376±0.029
		IRM	0.251±0.005	0.266±0.011	**0.342**±0.017	0.349±0.006	0.097±0.04	0.046±0.013	0.352±0.037	0.422±0.013
		Orderinvariant	0.253±0.005	0.27±0.01	0.628±0.026	0.564±0.024	0.122±0.018	0.106±0.065	0.545±0.079	0.47±0.065
		Spline	0.237±0.012	0.257±0.023	0.436±0.029	0.386±0.034	0.122±0.013	0.171±0.056	0.472±0.031	0.39±0.056
		VS	0.253±0.005	0.275±0.011	0.67±0.009	0.588±0.011	0.122±0.018	0.095±0.014	0.602±0.044	0.507±0.014
Cora	Degree	×	0.581±0.003	0.595±0.003	0.307±0.009	0.13±0.011	0.47±0.002	0.518±0.014	0.348±0.032	0.141±0.008
		CAGCN	0.581±0.003	**0.597**±0.002	0.135±0.009	0.128±0.025	0.47±0.002	0.522±0.025	0.256±0.08	0.231±0.025
		Dirichlet	0.534±0.007	0.551±0.004	0.12±0.004	0.196±0.003	0.414±0.007	0.449±0.01	0.163±0.002	0.356±0.01
		ETS	0.581±0.003	0.596±0.004	0.301±0.009	0.116±0.018	0.47±0.002	**0.523**±0.003	0.31±0.077	0.141±0.003
		GATS	0.581±0.003	0.596±0.004	0.185±0.018	0.229±0.039	0.47±0.002	0.521±0.011	0.211±0.004	0.308±0.011
		IRM	0.582±0.002	0.597±0.002	0.125±0.001	0.102±0.002	0.469±0.001	0.522±0.004	0.194±0.005	0.13±0.004
		Orderinvariant	0.581±0.003	0.592±0.002	0.226±0.024	0.213±0.049	0.47±0.002	0.498±0.027	0.318±0.042	0.196±0.027
		Spline	0.571±0.003	0.595±0.003	0.080±0.004	**0.068**±0.004	0.459±0.003	0.52±0.004	0.158±0.01	**0.098**±0.004
		VS	0.581±0.003	0.596±0.004	0.306±0.004	0.127±0.002	0.47±0.001	0.522±0.005	0.345±0.005	0.146±0.005
Cora	Word	×	0.607±0.003	0.628±0.001	0.284±0.009	0.111±0.013	0.603±0.004	0.633±0.031	0.263±0.004	0.118±0.019
		CAGCN	0.607±0.002	0.628±0.002	0.138±0.011	0.236±0.019	0.603±0.004	0.634±0.035	0.129±0.009	0.253±0.035
		Dirichlet	0.579±0.007	0.588±0.006	0.105±0.011	0.168±0.005	0.562±0.007	0.578±0.007	0.095±0.006	0.269±0.007
		ETS	0.607±0.002	0.628±0.002	0.282±0.002	0.11±0.003	0.603±0.004	0.634±0.013	0.243±0.023	0.106±0.013
		GATS	0.607±0.002	0.628±0.002	0.166±0.009	0.261±0.028	0.603±0.004	0.635±0.037	0.16±0.015	0.293±0.037
		IRM	0.608±0.001	0.63±0.002	0.115±0.002	0.088±0.003	0.602±0.003	0.635±0.004	0.106±0.002	0.098±0.004
		Orderinvariant	0.607±0.002	0.624±0.002	0.174±0.024	0.201±0.061	0.603±0.004	0.621±0.076	0.154±0.022	0.202±0.076
		Spline	0.598±0.005	0.629±0.002	0.073±0.002	**0.062**±0.005	0.591±0.002	0.635±0.004	0.063±0.006	**0.053**±0.004
		VS	0.607±0.001	**0.63**±0.002	0.283±0.003	0.111±0.003	0.603±0.004	**0.636**±0.003	0.261±0.005	0.119±0.003
CBAS	Color	×	0.83±0.014	0.829±0.011	0.169±0.013	0.151±0.014	0.703±0.015	0.746±0.027	0.266±0.02	0.169±0.018
		CAGCN	0.83±0.013	0.83±0.013	0.137±0.011	0.143±0.022	0.703±0.019	0.749±0.033	0.25±0.021	0.186±0.017
		Dirichlet	0.801±0.02	0.806±0.008	0.161±0.012	0.17±0.01	0.671±0.018	0.771±0.03	0.241±0.029	0.217±0.017
		ETS	0.83±0.013	0.827±0.014	0.146±0.013	0.164±0.007	0.703±0.019	0.76±0.037	0.28±0.023	0.176±0.019
		GATS	0.83±0.013	0.83±0.021	0.16±0.009	0.173±0.021	0.703±0.019	0.751±0.016	0.236±0.039	0.16±0.015
		IRM	0.829±0.013	0.839±0.015	0.142±0.009	**0.133**±0.006	0.72±0.019	0.803±0.04	0.207±0.035	**0.158**±0.017
		Orderinvariant	0.83±0.013	0.803±0.008	0.174±0.006	0.173±0.009	0.703±0.019	0.766±0.045	0.261±0.017	0.194±0.031
		Spline	0.82±0.016	0.824±0.011	0.159±0.009	0.16±0.014	0.683±0.019	0.786±0.038	0.225±0.034	0.179±0.035
		VS	0.829±0.012	**0.840**±0.011	0.166±0.011	0.146±0.012	0.717±0.019	**0.809**±0.008	0.242±0.019	0.182±0.014

Table 13: **Layerwise Anchoring for Node Classification Datasets with Intermediate Representation Distributions.** Here, we provide preliminary results for performing layerwise anchoring when performing node classification. We fit a gaussian distribution over the representations (similar to node feature anchoring) and then sample anchors from this distribution. We fit a gaussian distribution over the representations (similar to node feature anchoring) and then sample anchors from this distribution. We see that these alternative strategies do provide benefits in some cases, but overall, our original input node feature anchoring strategy is more performant.

			Shift: Concept						Shift: Covariate					
			Accuracy (↑)			ECE (↓)			Accuracy (↑)			ECE (↓)		
Dataset	Domain	Calibration	No G-Δ UQ	Random 1	Random 2	No G-Δ UQ	Random 1	Random 2	No G-Δ UQ	Random 1	Random 2	No G-Δ UQ	Random 1	Random 2
CBAS	Color	Dirichlet	0.801±0.02	0.765±0.012	0.839±0.023	0.161±0.012	0.301±0.018	0.234±0.027	0.671±0.018	0.74±0.023	0.689±0.032	0.241±0.029	0.349±0.04	0.381±0.029
		ETS	0.83±0.013	0.819±0.012	0.82±0.088	0.146±0.013	0.23±0.017	0.257±0.021	0.703±0.019	0.638±0.051	0.686±0.026	0.28±0.023	0.347±0.037	0.334±0.028
		IRM	0.829±0.013	0.821±0.019	0.885±0.026	**0.142**±0.009	0.219±0.012	0.206±0.066	0.72±0.019	0.617±0.084	0.693±0.026	**0.207**±0.035	0.363±0.03	0.299±0.036
		Orderinvariant	0.83±0.013	0.813±0.015	0.819±0.028	0.174±0.006	0.255±0.015	0.236±0.006	0.703±0.019	0.831±0.008	0.636±0.026	0.261±0.017	0.286±0.039	0.303±0.062
		Spline	0.82±0.016	0.814±0.022	0.839±0.035	0.159±0.009	0.235±0.017	0.196±0.036	0.683±0.019	0.621±0.052	**0.757**±0.026	0.225±0.034	0.312±0.026	0.331±0.024
		VS	0.829±0.012	0.817±0.017	**0.91**±0.006	0.166±0.011	0.251±0.012	0.259±0.021	0.717±0.019	0.593±0.038	0.695±0.051	0.242±0.019	0.38±0.037	0.359±0.02
Cora	Degree	Dirichlet	0.534±0.007	0.483±0.014	0.423±0.007	0.12±0.004	0.355±0.004	0.347±0.004	0.414±0.007	0.466±0.073	0.425±0.005	0.163±0.002	0.315±0.042	0.345±0.007
		ETS	0.581±0.003	0.562±0.01	0.496±0.002	0.301±0.009	0.297±0.009	0.289±0.006	0.47±0.002	0.498±0.119	0.34±0.076	0.31±0.077	0.511±0.005	0.329±0.008
		IRM	**0.582**±0.002	0.567±0.011	0.492±0.003	0.125±0.001	**0.072**±0.003	0.116±0.006	0.469±0.001	0.499±0.117	0.508±0.005	0.194±0.005	**0.094**±0.009	0.105±0.006
		Orderinvariant	0.581±0.003	0.566±0.004	0.495±0.002	0.226±0.024	0.151±0.015	0.14±0.008	0.47±0.002	0.499±0.107	0.108±0.034	0.318±0.042	0.506±0.005	0.093±0.009
		Spline	0.571±0.003	0.561±0.011	0.493±0.005	0.080±0.004	0.11±0.01	0.119±0.005	0.459±0.003	0.499±0.12	0.508±0.006	0.158±0.01	0.105±0.03	0.127±0.012
		VS	0.581±0.003	0.571±0.002	0.279±0.009	0.306±0.004	0.493±0.008	0.272±0.009	0.47±0.001	**0.511**±0.091	0.51±0.002	0.345±0.005	0.347±0.051	0.323±0.007
Cora	Word	Dirichlet	0.579±0.007	0.581±0.004	0.504±0.004	0.105±0.011	0.271±0.011	0.285±0.002	0.562±0.007	0.586±0.009	0.497±0.01	0.095±0.006	0.264±0.022	0.275±0.007
		ETS	0.607±0.002	0.641±0.003	0.575±0.003	0.282±0.002	0.352±0.012	0.328±0.007	0.603±0.004	0.633±0.003	0.567±0.004	0.243±0.023	0.377±0.023	0.374±0.006
		IRM	0.608±0.001	**0.642**±0.002	0.574±0.003	0.115±0.002	0.106±0.004	0.154±0.005	0.602±0.003	0.635±0.004	0.569±0.003	0.106±0.002	0.136±0.012	0.173±0.007
		Orderinvariant	0.607±0.002	**0.642**±0.004	0.573±0.004	0.174±0.024	0.109±0.011	0.107±0.01	0.603±0.004	**0.638**±0.004	0.566±0.004	0.154±0.022	0.087±0.006	0.073±0.004
		Spline	0.598±0.005	0.641±0.002	0.576±0.004	0.073±0.002	0.076±0.004	**0.068**±0.007	0.591±0.002	0.632±0.002	0.568±0.003	**0.063**±0.006	0.066±0.005	0.077±0.004
		VS	0.607±0.001	0.639±0.003	0.583±0.005	0.283±0.003	0.345±0.007	0.335±0.012	0.603±0.004	0.637±0.004	0.579±0.004	0.261±0.005	0.396±0.028	0.384±0.005
WebKB	University	Dirichlet	0.229±0.018	0.214±0.000	0.228±0.012	0.472±0.06	0.56±0.000	0.552±0.041	0.244±0.105		0.347±0.012	0.299±0.092		0.429±0.05
		ETS	0.253±0.005	**0.279**±0.000	0.234±0.01	0.64±0.06	0.437±0.000	**0.33**±0.022	0.121±0.021		0.225±0.013	0.539±0.112		0.258±0.028
		IRM	0.251±0.005	0.251±0.000	0.232±0.009	0.342±0.017	0.379±0.000	0.459±0.01	0.097±0.04		0.187±0.021	0.352±0.037		0.294±0.018
		Orderinvariant	0.253±0.005	**0.279**±0.000	0.237±0.01	0.628±0.026	0.568±0.000	0.53±0.049	0.122±0.018		0.221±0.026	0.545±0.079		0.321±0.061
		Spline	0.237±0.012	0.237±0.000	0.233±0.008	0.436±0.029	0.467±0.000	0.483±0.041	0.122±0.013		0.205±0.01	0.472±0.031		0.329±0.035
		VS	0.253±0.005	**0.279**±0.000	0.234±0.01	0.67±0.009	0.49±0.000	0.344±0.02	0.122±0.018		0.201±0.011	0.602±0.044		0.256±0.014

Table 14: **Layerwise Anchoring for Node Classification Datasets with Random Shuffling.** Here, we provide preliminary results for performing layerwise anchoring when performing node classification. We use random shuffling (similar to the proposed hidden layer strategy) to create the interemediate representations. We see that these alternative strategies do provide benefits.

			Shift: Concept						Shift: Covariate					
			Accuracy (↑)			ECE (↓)			Accuracy (↑)			ECE (↓)		
Dataset	Domain	Calibration	No G-Δ UQ	Batch 1	Batch 2	No G-Δ UQ	Batch 1	Batch 2	No G-Δ UQ	Batch 1	Batch 2	No G-Δ UQ	Batch 1	Batch 2
CBAS	Color	Dirichlet	0.801±0.02	0.757±0.045	0.58±0.046	0.161±0.012	0.309±0.059	0.431±0.033	0.671±0.018	0.548±0.035	0.629±0.019	0.241±0.029	0.48±0.03	0.407±0.01
		ETS	**0.83**±0.013	0.699±0.036	0.637±0.014	0.146±0.013	0.265±0.013	0.258±0.015	0.703±0.019	0.562±0.087	0.507±0	0.28±0.023	0.37±0.021	0.333±0.02
		IRM	0.829±0.013	0.711±0.031	0.724±0.029	**0.142**±0.009	0.284±0.032	0.291±0.02	**0.72**±0.019	0.59±0.079	0.657±0.037	**0.207**±0.035	0.336±0.032	0.268±0.037
		Orderinvariant	**0.83**±0.013	0.788±0.007	0.574±0.051	0.174±0.006	0.268±0.023	0.208±0.055	0.703±0.019	0.61±0.011	0.5±0.019	0.261±0.017	0.334±0.035	0.249±0.037
		Spline	0.82±0.016	0.695±0.039	0.652±0.022	0.159±0.009	0.279±0.018	0.236±0.013	0.683±0.019	0.49±0.124	0.6±0.032	0.225±0.034	0.364±0.034	0.308±0.054
		VS	0.829±0.012	0.73±0.043	0.693±0.051	0.166±0.011	0.264±0.009	0.197±0.033	0.717±0.019	0.429±0.083	0.607±0.042	0.242±0.019	0.478±0.042	0.312±0.014
Cora	Degree	Dirichlet	0.534±0.007	0.515±0.003	0.442±0.012	0.12±0.004	0.304±0.01	0.315±0.004	0.414±0.007	0.507±0.004	0.419±0.006	0.163±0.002	0.28±0.006	0.338±0.004
		ETS	0.581±0.003	0.576±0.011	0.516±0.013	0.301±0.009	0.317±0.018	0.285±0.007	0.47±0.002	0.563±0.003	0.496±0.005	0.31±0.077	0.373±0.009	0.311±0.006
		IRM	**0.582**±0.002	0.579±0.009	0.523±0.008	0.125±0.001	**0.076**±0.004	0.129±0.004	0.469±0.001	0.562±0.004	0.494±0.004	0.194±0.005	**0.088**±0.011	0.098±0.003
		Orderinvariant	0.581±0.003	**0.582**±0.003	0.518±0.005	0.226±0.024	0.134±0.023	0.126±0.012	0.47±0.002	0.561±0.004	0.496±0.004	0.318±0.042	0.091±0.014	0.096±0.007
		Spline	0.571±0.003	0.58±0.006	0.518±0.011	0.080±0.004	0.093±0.007	0.092±0.007	0.459±0.003	**0.565**±0.004	0.496±0.005	0.158±0.01	0.091±0.009	0.128±0.012
		VS	0.581±0.003	0.581±0.005	0.529±0.005	0.306±0.004	0.313±0.006	0.294±0.004	0.47±0.001	0.562±0.005	0.498±0.008	0.345±0.005	0.368±0.016	0.308±0.003
Cora	Word	Dirichlet	0.579±0.007	0.575±0.004	0.491±0.013	0.105±0.011	0.28±0.007	0.282±0.012	0.562±0.007	0.586±0.009	0.507±0.006	0.095±0.006	0.264±0.022	0.249±0.007
		ETS	0.607±0.002	0.636±0.003	0.562±0.006	0.282±0.002	0.359±0.02	0.311±0.006	0.603±0.004	0.633±0.003	0.561±0.005	0.243±0.023	0.377±0.023	0.365±0.005
		IRM	0.608±0.001	0.632±0.004	0.562±0.006	0.115±0.002	0.124±0.006	0.16±0.005	0.602±0.003	0.635±0.004	0.557±0.006	0.106±0.002	0.136±0.012	0.176±0.007
		Orderinvariant	0.607±0.002	**0.639**±0.003	0.561±0.006	0.174±0.024	0.111±0.008	0.095±0.006	0.603±0.004	**0.638**±0.004	0.56±0.004	0.154±0.022	0.087±0.006	0.076±0.006
		Spline	0.598±0.005	0.633±0.004	0.561±0.007	0.073±0.002	0.077±0.005	**0.069**±0.004	0.591±0.002	0.632±0.002	0.56±0.006	**0.063**±0.006	0.066±0.005	0.08±0.004
		VS	0.607±0.001	0.633±0.006	0.574±0.007	0.283±0.003	0.368±0.009	0.32±0.005	0.603±0.004	0.637±0.004	0.573±0.008	0.261±0.005	0.396±0.028	0.373±0.006
WebKB	University	Dirichlet	0.229±0.018	0.231±0.015	0.234±0.007	0.472±0.06	0.562±0.014	0.534±0.022	0.244±0.105	0.242±0.166	**0.298**±0.077	0.299±0.092	0.468±0.092	0.483±0.055
		ETS	0.253±0.005	0.277±0.007	0.234±0.003	0.64±0.06	0.421±0.017	0.327±0.015	0.121±0.021	0.128±0.017	0.101±0.033	0.539±0.112	0.437±0.032	**0.293**±0.01
		IRM	0.251±0.005	0.265±0.019	0.232±0.014	0.342±0.017	0.377±0.015	0.438±0.015	0.097±0.04	0.118±0.033	0.093±0.034	0.352±0.037	0.482±0.02	0.435±0.016
		Orderinvariant	0.253±0.005	0.268±0.01	0.231±0.01	0.628±0.026	0.513±0.071	0.431±0.025	0.122±0.018	0.122±0.018	0.1±0.029	0.545±0.079	0.475±0.049	0.38±0.069
		Spline	0.237±0.012	0.242±0.01	0.228±0.014	0.436±0.029	0.415±0.042	0.484±0.035	0.122±0.013	0.129±0.024	0.097±0.013	0.472±0.031	0.478±0.033	0.425±0.013
		VS	0.253±0.005	**0.279**±0.007	0.232±0.005	**0.67**±0.009	0.441±0.021	0.323±0.015	0.122±0.018	0.132±0.01	0.101±0.033	0.602±0.044	0.455±0.041	0.297±0.008

Table 15: **Alternative Anchoring Strategies.** Here, we consider an alternative anchoring formulation for graph classification. Namely, instead of shuffling features across the batch (denoted Batch in the table), we perform READOUT anchoring by fitting a normal distribution over the hidden representations. We then randomly sample from this distribution to create anchors. Conceptually, this is similar to the node feature anchoring strategy. One potential direction of future work that is permitted by this formulation is to optimize the parameters of this distribution given a signal from an appropriate auxiliary task or loss. For example, we could perform an alternating optimization where the GNN is trained to minimize the loss, and the mean and variance of the anchoring distribution are optimized to minimize the expected calibration error on a separate calibration dataset. While a rigorous formulation is left to future work, we emphasize that the potential for improving the anchoring distribution, and thus controlling corresponding hypothesis diversity, is in fact a unique benefit of G-ΔUQ.

		Test Acc			*Test Cal*			*OOD Acc*			*OOD Cal*		
Shift Type	***Method***	***MPNN***	***Batch***	***Random***	***MPNN***	***Batch***	***Random***	***MPNN***	***Batch***	***Random***	***MPNN***	***Batch***	***Random***
GoodMotif, basis, concept	Dirichlet	0.995 ± 0.0007	0.994 ± 0.0002	0.996 ± 0.0009	0.040 ± 0.0037	0.036 ± 0.0016	0.035 ± 0.0058	0.924 ± 0.0069	0.923 ± 0.0117	0.942 ± 0.0034	0.080 ± 0.0153	0.102 ± 0.0071	**0.062 ± 0.0086**
	ETS	0.995 ± 0.0007	0.995 ± 0.0005	0.996 ± 0.0007	0.035 ± 0.0034	0.036 ± 0.0101	0.032 ± 0.0052	0.925 ± 0.0095	0.926 ± 0.009	0.935 ± 0.0068	0.095 ± 0.0098	0.096 ± 0.0128	0.087 ± 0.01451
	IRM	0.9954 ± 0.0007	0.9957 ± 0.0009	**0.9965 ± 0.0004**	**0.0198 ± 0.0089**	0.0229 ± 0.0105	0.0225 ± 0.0038	0.9251 ± 0.0096	0.9301 ± 0.0123	**0.9462 ± 0.0024**	0.0873 ± 0.0176	0.0966 ± 0.0103	0.0907 ± 0.0276
	OrderInvariant	0.995 ± 0.0007	0.995 ± 0.0005	0.995 ± 0.0005	0.033 ± 0.0094	0.028 ± 0.0047	0.032 ± 0.0009	0.925 ± 0.0095	0.928 ± 0.0104	0.935 ± 0.0027	0.090 ± 0.0092	0.093 ± 0.0070	0.0754 ± 0.0029
	Spline	0.995 ± 0.0007	0.995 ± 0.0007	0.9962 ± 0.0005	0.034 ± 0.0002	0.035 ± 0.0090	0.032 ± 0.0048	0.924 ± 0.0098	0.926 ± 0.0092	0.937 ± 0.0030	0.091 ± 0.0084	0.089 ± 0.0123	0.083 ± 0.0065
	VS	0.995 ± 0.0007	0.995 ± 0.0005	0.996 ± 0.000	0.035 ± 0.0034	0.036 ± 0.0087	0.033 ± 0.0098	0.925 ± 0.0094	0.926 ± 0.0095	0.936 ± 0.0053	0.094 ± 0.0096	0.095 ± 0.0133	0.082 ± 0.009
GoodMotif,basis, covariate	Dirichlet	0.999 ± 0.0003	0.999 ± 0.0004	0.999 ± 0.0002	0.017 ± 0.0054	0.017 ± 0.0019	0.014 ± 0.0004	0.685 ± 0.0504	0.650 ± 0.0450	**0.698 ± 0.0139**	0.336 ± 0.0667	0.371 ± 0.0474	0.320 ± 0.0140
	ETS	0.9997 ± 0.0004	0.999 ± 0.0005	0.999 ± 0.0002	0.0095 ± 0.0091	0.017 ± 0.0064	0.017 ± 0.0056	0.690 ± 0.0434	0.649 ± 0.0476	0.686 ± 0.0226	0.313 ± 0.0413	**0.3739 ± 0.0485**	0.334 ± 0.0167
	IRM	0.9997 ± 0.0004	0.999 ± 0.0006	0.999 ± 0.0003	**0.0085 ± 0.0032**	0.010 ± 0.0032	0.014 ± 0.0042	0.690 ± 0.0434	0.647 ± 0.0472	0.692 ± 0.0226	0.315 ± 0.0505	0.354 ± 0.0450	0.328 ± 0.0211
	OrderInvariant	0.9997 ± 0.0004	0.999 ± 0.0005	0.999 ± 0.0003	0.014 ± 0.0028	0.020 ± 0.0090	0.013 ± 0.0081	0.690 ± 0.0434	0.649 ± 0.0450	0.689 ± 0.0170	0.320 ± 0.0501	0.358 ± 0.0410	0.328 ± 0.0218
	Spline	0.9997 ± 0.0004	0.999 ± 0.0005	0.999 ± 0.0003	0.016 ± 0.0049	0.017 ± 0.0053	0.017 ± 0.0052	0.690 ± 0.0434	0.649 ± 0.0476	0.6923 ± 0.0199	0.324 ± 0.0548	0.3733 ± 0.0507	0.327 ± 0.0105
	VS	**0.9998 ± 0.0001**	0.999 ± 0.0003	0.999 ± 0.0002	0.011 ± 0.0053	0.014 ± 0.0034	0.012 ± 0.0016	0.682 ± 0.0561	0.650 ± 0.0546	0.682 ± 0.0251	0.325 ± 0.0568	0.371 ± 0.0591	0.337 ± 0.0264
GOODSST2,length,concept	Dirichlet	0.938 ± 0.0019	0.939 ± 0.0056	**0.942 ± 0.00180**	0.189 ± 0.01989	**0.165 ± 0.0179**	0.187 ± 0.0256	**0.694 ± 0.0193**	0.693 ± 0.0020	0.687 ± 0.0027	0.146 ± 0.0196	**0.133 ± 0.015**	0.169 ± 0.0168
	ETS	0.938 ± 0.0020	0.939 ± 0.0060	0.941 ± 0.0017	0.389 ± 0.0018	0.390 ± 0.0022	0.393 ± 0.0007	0.6940 ± 0.0193	0.692 ± 0.0019	0.687 ± 0.0034	0.214 ± 0.0098	0.216 ± 0.0033	0.220 ± 0.0057
	IRM	0.939 ± 0.0016	0.939 ± 0.0058	0.941 ± 0.0018	0.326 ± 0.0011	0.326 ± 0.0013	0.327 ± 0.0017	0.693 ± 0.0185	0.692 ± 0.0026	0.685 ± 0.0026	0.240 ± 0.0017	0.232 ± 0.0050	0.242 ± 0.0053
	OrderInvariant	0.938 ± 0.0020	0.939 ± 0.0060	0.941 ± 0.0022	0.314 ± 0.0014	0.315 ± 0.0029	0.315 ± 0.0012	0.6940 ± 0.0193	0.692 ± 0.0019	0.687 ± 0.0033	0.224 ± 0.0010	0.222 ± 0.0030	0.223 ± 0.0054
	Spline	0.938 ± 0.0026	0.938 ± 0.0044	0.941 ± 0.0010	0.329 ± 0.0021	0.329 ± 0.0019	0.328 ± 0.0012	0.692 ± 0.0190	0.692 ± 0.0022	0.687 ± 0.0035	0.234 ± 0.0052	0.231 ± 0.0044	0.243 ± 0.0034
	VS	0.938 ± 0.0027	0.939 ± 0.0057	0.941 ± 0.0018	0.290 ± 0.2099	0.484 ± 0.0008	0.487 ± 0.0007	0.693 ± 0.0184	0.693 ± 0.0018	0.687 ± 0.0031	0.331 ± 0.0484	0.375 ± 0.0022	0.382 ± 0.0048
GOODSST2,length,covariate	Dirichlet	**0.896 ± 0.0029**	0.893 ± 0.0009	0.895 ± 0.00095	0.196 ± 0.0155	**0.172 ± 0.0091**	0.1797 ± 0.0109	0.825 ± 0.0037	0.827 ± 0.0066	0.805 ± 0.0150	0.163 ± 0.0198	**0.141 ± 0.0087**	0.142 ± 0.0122
	ETS	**0.8966 ± 0.0023**	0.894 ± 0.0011	0.894 ± 0.0006	0.357 ± 0.0013	0.359 ± 0.0004	0.362 ± 0.0019	0.826 ± 0.0036	0.828 ± 0.0065	0.806 ± 0.0117	0.309 ± 0.0050	0.314 ± 0.0076	0.300 ± 0.0070
	IRM	0.895 ± 0.0019	0.893 ± 0.0003	0.894 ± 0.0007	0.307 ± 0.0004	0.307 ± 0.0003	0.306 ± 0.0020	0.826 ± 0.0040	**0.828 ± 0.0065**	0.809 ± 0.0152	0.276 ± 0.0046	0.277 ± 0.0061	0.265 ± 0.0078
	OrderInvariant	**0.896 ± 0.0023**	0.894 ± 0.0011	0.894 ± 0.0008	0.288 ± 0.0008	0.285 ± 0.0008	0.284 ± 0.0013	0.826 ± 0.0036	0.828 ± 0.0065	0.806 ± 0.0106	0.244 ± 0.0022	0.241 ± 0.0037	0.225 ± 0.0054
	Spline	0.894 ± 0.0016	0.890 ± 0.0009	0.892 ± 0.0040	0.309 ± 0.0024	0.307 ± 0.0009	0.307 ± 0.0022	0.822 ± 0.0026	0.822 ± 0.0092	0.801 ± 0.0110	0.275 ± 0.0043	0.276 ± 0.0063	0.264 ± 0.0063
	VS	0.8963 ± 0.0028	0.893 ± 0.0008	0.894 ± 0.0007	0.291 ± 0.1833	0.460 ± 0.0011	0.465 ± 0.0010	0.821 ± 0.0053	0.827 ± 0.0071	0.806 ± 0.0119	0.299 ± 0.1395	0.431 ± 0.0061	0.429 ± 0.0054

A.9 POST-HOC CALIBRATION STRATEGIES

Several post hoc strategies have been developed for calibrating the predictions of a model. These have the advantage of flexibility, as they operate only on the outputs of a model and do not require that any changes be made to the model itself. Some methods include:

- **Temperature scaling (TS)** (Guo et al., 2017) simply scales the logits by a temperature parameter $T > 1$ to smooth the predictions. The scaling parameter T can be tuned on a validation set.
- **Ensemble temperature scaling (ETS)** (Zhang et al., 2020) learns an ensemble of temperature-scaled predictions with uncalibrated predictions ($T = 1$) and uniform probabilistic outputs ($T = \infty$).
- **Vector scaling** (VS) Guo et al. (2017) scales the entire output vector of class probabilities, rather than just the logits.
- **Multi-class isotonic regression (IRM)** (Zhang et al., 2020) is a multiclass generalization of the famous isotonic regression method (Zadrozny & Elkan, 2002)): it ensembles predictions and labels, then learns a monotonically increasing function to map transformed predictions to labels.
- **Order-invariant calibration** (Rahimi et al., 2020) uses a neural network to learn an intra-order-preserving calibration function that can preserve a model's top-k predictions.
- **Spline** calibration instead uses splines to fit the calibration function (Gupta et al., 2021).
- **Dirichlet calibration** (Kull et al., 2019) models the distribution of outputs using a Dirichlet distribution, using simple log-transformation of the uncalibrated probabilities which are then passed to a regularized fully connected neural network layer with softmax activation.

For node classification, some graph-specific post-hoc calibration methods have been proposed. **CaGCN** (Wang et al., 2021) uses the graph structure and an additional GCN to produce node-wise temperatures. GATS (Hsu et al., 2022) extends this idea by using graph attention to model the influence of neighbors' temperatures when learning node-wise temperatures. We use the post hoc calibration baselines provided by Hsu et al. in our experiments.

All of the above methods, and others, may be applied to the output of any model including one using G-ΔUQ. As we have shown, applying such post hoc methods to the outputs of the calibrated models may improve uncertainty estimates even more. Notably, calibrated models are expected to produce confidence estimates that match the true probabilities of the classes being predicted (Naeini et al., 2015; Guo et al., 2017; Ovadia et al., 2019). While poorly calibrated CIs are over/under confident in their predictions, calibrated CIs are more trustworthy and can also improve performance on other safety-critical tasks which implicitly require reliable prediction probabilities (see Sec. 5). We report the top-1 label expected calibration error (ECE) (Kumar et al., 2019; Detlefsen et al., 2022). Formally, let p_i be the top-1 probability, c_i be the predicted confidence, b_i a uniformly sized bin in $[0, 1]$. Then,

$$ECE := \sum_{i}^{N} b_i \|(p_i - c_i)\|$$

.

A.10 Details on Generalization Gap Prediction

Accurate estimation of the expected generalization error on unlabeled datasets allows models with unacceptable performance to be pulled from production. To this end, generalization error predictors (GEPs) (Garg et al., 2022; Ng et al., 2022; Jiang et al., 2019; Trivedi et al., 2023a; Guillory et al., 2021) which assign sample-level scores, $S(x_i)$ which are then aggregated into dataset-level error estimates, have become popular. We use maximum softmax probability and a simple thresholding mechanism as the GEP (since we are interested in understanding the behavior of confidence indicators), and report the error between the predicted and true target dataset accuracy: $GEPError := ||\text{Acc}_{target} - \frac{1}{|X|}\sum_i \mathbb{I}(\text{S}(\bar{\text{x}}_i; \text{F}) > \tau)||$ where τ is tuned by minimizing GEP error on the validation dataset. We use the confidences obtained by the different baselines as sample-level scores, $\text{S}(\text{x}_i)$ corresponding to the model's expectation that a sample is correct. The MAE between the estimated error and true error is reported on both in- and out-of -distribution test splits provided by the GOOD benchmark.

A.11 Results on Generalization Error Prediction

GEP Experimental Setup. GEPs (Garg et al., 2022; Ng et al., 2022; Jiang et al., 2019; Trivedi et al., 2023a; Guillory et al., 2021) aggregate sample-level scores capturing a model's uncertainty about the correctness of a prediction into dataset-level error estimates. Here, we use maximum softmax probability for scores and a thresholding mechanism as the GEP. (See Appendix A.10 for more details.) We consider `READOUT` anchoring with both pretrained and end-to-end training, and report the mean absolute error between the predicted and true target dataset accuracy on the OOD test split.

GEP Results. As shown in Table 16, both pretrained and end-to-end G-ΔUQ outperform the vanilla model on 7/8 datasets. Notably, we see that pretrained G-ΔUQ is particularly effective as it obtains the best performance across 6/8 datasets. This not only highlights its utility as a flexible, light-weight strategy for improving uncertainty estimates without sacrificing accuracy, but also emphasizes that importance of structure, in lieu of full stochasticity, when estimating GNN uncertainties.

Table 16: **GOOD-Datasets, Generalization Error Prediction Performance**. The MAE between the predicted and true test error on the OOD test split is reported. G-ΔUQ variants outperform vanilla models on 7/8 datasets (GOODMotif(Basis,Covariate) being the exception). Pretrained G-ΔUQ is particularly effective at this task as it achieves the best performance overall on 6/8 datasets. Promisingly, we see that regular G-ΔUQ improves performance over the vanilla model on 6/8 datasets (even if it is not the best overall). We further observe that performing generalization error prediction is more challenging under covariate shift than concept shift on the GOODCMNIST, GOODMotif(Basis) and GOODMotif(Size) datasets. On these datasets, the MAE is almost twice as large than their respective concept shift counterparts, across methods. GOODSST2 is the exception, where concept shift is in fact more challenging. To the best our knowledge, we are the first to investigate generalization error prediction on GNN-based tasks under distribution shift. Understanding this behavior further is an interesting direction of future work.

	CMNIST (Color)		MotifLPE (Basis)		MotifLPE (Size)		SST2	
Method	**Concept(↓)**	**Covariate (↓)**	**Concept(↓)**	**Covariate(↓)**	**Concept(↓)**	**Covariate(↓)**	**Concept(↓)**	**Covariate(↓)**
Vanilla	0.200 ± 0.009	0.510 ± 0.089	0.045 ± 0.003	$\mathbf{0.570 \pm 0.012}$	0.324 ± 0.018	0.537 ± 0.146	0.117 ± 0.006	0.056 ± 0.044
G-ΔUQ	$\mathbf{0.190 \pm 0.010}$	0.493 ± 0.072	0.023 ± 0.003	0.572 ± 0.019	0.317 ± 0.007	0.528 ± 0.189	0.124 ± 0.016	0.054 ± 0.043
Pretr. G-ΔUQ	0.192 ± 0.005	$\mathbf{0.387 \pm 0.048}$	$\mathbf{0.018 \pm 0.012}$	0.573 ± 0.004	$\mathbf{0.307 \pm 0.016}$	$\mathbf{0.356 \pm 0.143}$	$\mathbf{0.114 \pm 0.004}$	$\mathbf{0.030 \pm 0.026}$

A.12 Additional Study on Pretrained G-ΔUQ

For the datasets and data shifts on which we reported out-of-distribution calibration error of pretrained vs. in-training G-ΔUQ earlier in Fig. 4, we now report additional results for in-distribution and out-of distribution accuracy as well as calibration error. We also include results for the additional GOODMotif-basis benchmark for completeness, noting that the methods provided by the original benchmark Gui et al. (2022) generalized poorly to this split (which may be related to why G-ΔUQ methods offer little improvement over the vanilla model.) Fig. 8 shows these extended results. By these additional metrics, we again see the competitiveness of applying G-ΔUQ to a pretrained model versus using it in end-to-end training.

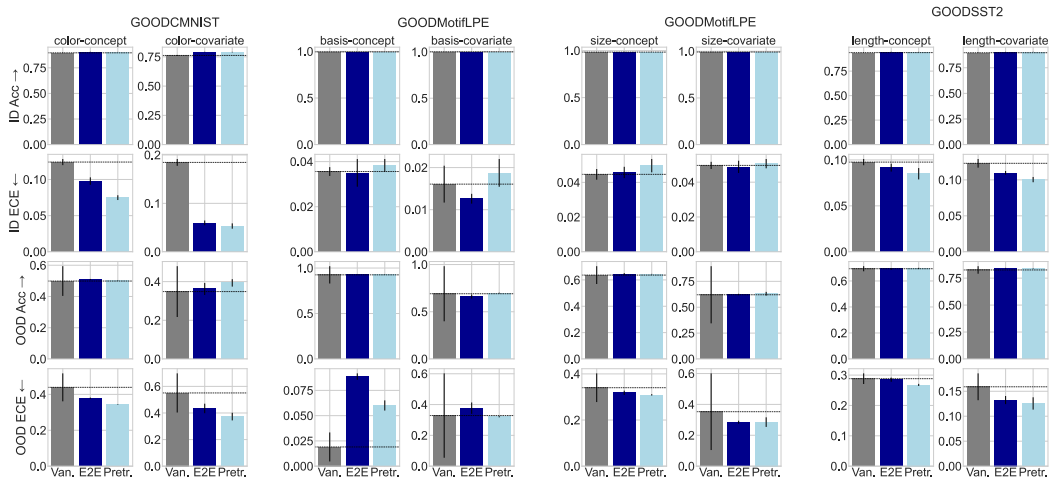


Figure 8: **Evalauting Pretrained G-ΔUQ.** Here, we report the performance of pretrained G-ΔUQ models vs. end-to-end and vanilla models with respect to in-distribution and out-of-distribution accuracy as well as expected calibration error. With the exception of the GOODMotif (basis) dataset, pretrained G-ΔUQ improves the OOD ECE over both the vanilla model and end-to-end G-ΔUQ at comparable or improved OOD accuracy on 7/8 datasets. Furthermore, pretrained G-ΔUQ also improves the ID ECE on all but the GOODMotif (size) datasets (6/8), where it performs comparably to the vanilla model, and maintains the ID accuracy. (We note that all methods are comparably better calibrated on the GOODMotif ID data than GOODCMIST/GOODSST2 ID data; we suspect this is because there may exist simple shortcuts available in the GOODMotif dataset that can be used on the ID test set effectively.) Overall, these results clearly demonstrate that pretrained G-ΔUQ does offer some performance advantages over end-to-end G-ΔUQ and does so at reduced training times (see Table. A.13). For example, on GOODCMNIST (covariate shift), pretrained G-ΔUQ is not only 50% faster than end-to-end G-ΔUQ , it also improves OOD accuracy and OOD ECE over both the vanilla and end-to-end G-ΔUQmodels.

A.13 RUNTIME TABLE

Table 17: **Runtimes.** We include the runtimes of both training per epoch (in seconds) and performing calibration. Reducing stochasticity can help reduce computation (L1 $\rightarrow$ L3). Cost can also be reduced by using a pretrained model.

	GOODCMNIST		GOODSST2		GOODMotifLPE	
Dataset	Training (S)	Inference (S)	Training (S)	Inference (S)	Training (S)	Inference (S)
Vanilla	18.5	25.8	10.8	18.5	3.8	4.5
Temp. Scaling	18.5	23.5	10.8	13.4	3.8	5.3
DEns (Ens Size=3)	18.456 x Ens Size	59.4	10.795 x Ens Size	29.0	3.8 x Ens Size	11.8
G-ΔUQ (L1, 10 anchors)	22.1	181.5	15.9	17.1	5.8	15.5
G-ΔUQ (L2, 10)	22.4	148.6	12.7	15.5	5.8	11.8
G-ΔUQ (HiddenRep, 10)	18.5	28.0	13.8	19.6	3.9	6.5
G-ΔUQ (Pretr. HiddenRep, 10)	8.6	27.8	6.8	16.0	2.5	6.4

A.14 MEAN AND VARIANCE OF NODE FEATURE GAUSSIANS

Table 18: **Mean and Variance of Node Feature Anchoring Gaussians.** We report the mean and variance of the Gaussian distributions fitted to the input node features. Because the input node features vary in size, we report aggregate statistics over the mean and variance corresponding to each dimension. For example, Min(Mu) indicates that we are reports the minimum mean over the d-dim set of means.

Dataset	Domain	Shift	Min (Mu)	Max (Mu)	Mean (Mu)	Std (Mu)	Min (Std)	Max (Std)	Mean (Std)	Std (Std)
GOODSST2	length	concept	-4.563	0.69	-0.011	0.278	0.163	0.803	0.242	0.049
GOODSST2	length	covariate	-4.902	0.684	-0.01	0.3	0.175	0.838	0.255	0.05
GOODCMNIST	color	concept	0.117	0.133	0.127	0.008	0.092	0.097	0.095	0.003
GOODCMNIST	color	covariate	0.087	0.131	0.102	0.025	0.108	0.109	0.108	0
GOODMotifLPE	size	covariate	0.003	0.021	0.011	0.008	0.835	1.728	1.248	0.377
GOODMotifLPE	size	concept	-0.006	0	-0.002	0.003	0.542	1.114	0.783	0.242
GOODMotifLPE	basis	concept	-0.011	0.015	0.001	0.011	0.721	1.464	1.09	0.304
GOODMotifLPE	basis	covariate	-0.007	-0.002	-0.004	0.002	0.808	1.913	1.251	0.469
GOODWebKB	university	concept	0	0.95	0.049	0.099	0.001	0.5	0.168	0.095
GOODWebKB	university	covariate	0	0.934	0.05	0.104	0.001	0.5	0.164	0.098
GOODCora	degree	concept	0	0.507	0.007	0.017	0.001	0.5	0.061	0.051
GOODCora	degree	covariate	0	0.518	0.007	0.017	0.001	0.5	0.061	0.052
GOODCBAS	color	covariate	0.394	0.591	0.471	0.093	0.142	0.492	0.403	0.174
GOODCBAS	color	concept	0.23	0.569	0.4	0.144	0.168	0.495	0.39	0.152

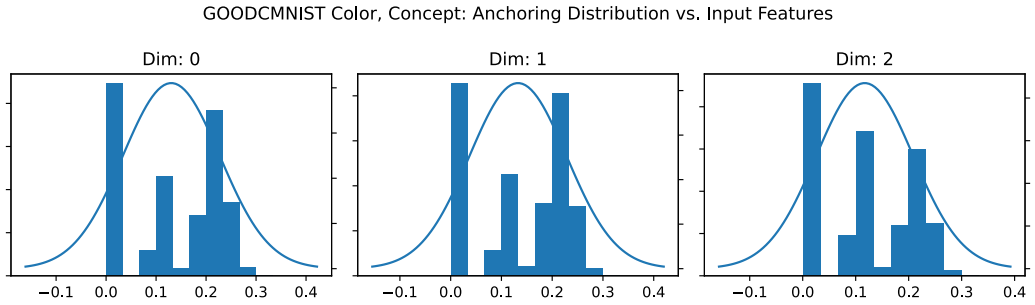


Figure 9: **GOODCMNIST, Concept, Anchoring Distribution.** We plot the mean and variance of the fitted anchoring distribution vs. the true feature distribution for each input dimension. We observe there is a mismatch between the empircal distribution and the fitted Gaussian. However, we did not find this mismatch to harm the effectiveness of G-ΔUQ.

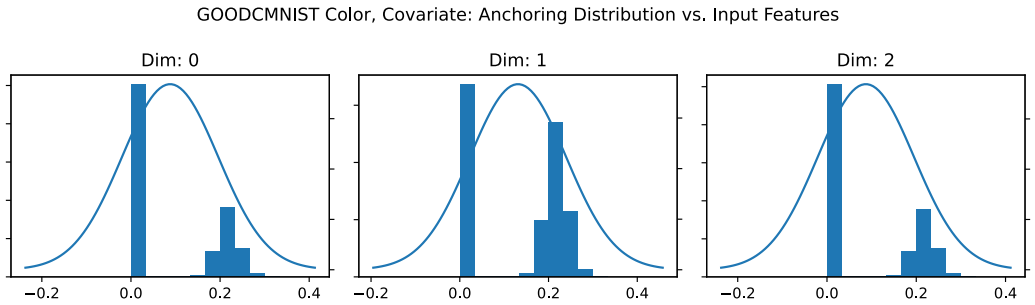


Figure 10: **GOODCMNIST, Covariate, Anchoring Distribution.** We plot the mean and variance of the fitted anchoring distribution vs. the true feature distribution for each input dimension. We observe there is a mismatch between the empircal distribution and the fitted Gaussian. However, we did not find this mismatch to harm the effectiveness of G-ΔUQ.

Table 19: **Number of Parmeters per Model.**We provide the number of parameters in the vanilla and modified parameter as follows. Note, that the change in parameters is architecture and input dimension dependent. For example, GOODCMNIST, and GOODSST2 use GIN MPNN layers. Therefore, when changing the layer dimension, we are changing the dimension of its internal MLP. It is not an error that intermediate layer G-ΔUQhave the same number of parameters, this is due to the architecture: these layers are the same size in the vanilla model. Likewise, GOODCora's input features have dimension is 8701, so doubling the input layer's dimension appears to add a signficant number of parameters. We do not believe this

Dataset	GOODCMNIST	GOODMotif	GOODSST2	GOODCORA	GOODWebKB	GOODCBAS
Baseline	2001310	911403	1732201	2816770	695105	185104
G-ΔUQ(NFA)	2003110	913803	2193001	5429770	1206005	186304
G-ΔUQ(L1)	2360110	1633203	2091001			
G-ΔUQ(L2)	2360110	1633203	2091001			
G-ΔUQ(L3)	2360110					
G-ΔUQ(L4)	2360110					
G-ΔUQ(Readout)	2004310	912303	1732501			

A.15 Expanded Discussion on Anchoring Design Choices

Below, we expand upon some of the design choices for the proposed anchoring strategies.

When performing node featuring anchoring, how does fitting a Gaussian distribution to the input node features help manage the combinatorial stochasticity induced by message passing?

Without loss of generality, consider a node classification setting, where every sample is assigned a unique anchor. Then, due to message passing, after l hops, a given node's representation will have aggregated information from its l hop neighborhood. However, since each node in this neighborhood has a unique anchor, we see that any given node's representation is not only stochastic due to its own anchor but also that of its neighbors. For example, if any of its neighbors are assigned a different anchor, then the given node's representation will change, even if its own anchor did not. Since this behavior holds true for all nodes and each of their respective neighborhoods, we loosely refer to this phenomenon having combinatorial complexity, as effectively marginalizing out the anchoring distribution would require handling any and all changes to all l-hop neighbors. In contrast, when performing anchored image classification, the representation of a sample is only dependent on its unique, corresponding anchor, and is not influenced by the anchors of other samples. To this end, using the fitted Gaussian distribution helps manage this complexity, since changes to the anchors of a node's l-hop neighborhood are simpler to model as they require only learning to marginalize out a Gaussian distribution (instead of the training distribution). Indeed, for example, if we were to assume simplified model where message passing only summed node neighbors, the anchoring distribution would remain gaussian after l rounds of message passing since the sum of gaussians is still gaussian (the exact parameters of the distribution would depend on the normalization used however).

REFERENCES

Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2021.

Rushil Anirudh and Jayaraman J. Thiagarajan. Out of distribution detection via neural network anchoring. In *Asian Conference on Machine Learning, ACML 2022, 12-14 December 2022, Hyderabad, India*, 2022.

Beatrice Bevilacqua, Yangze Zhou, and Bruno Ribeiro. Size-invariant graph representations for graph classification extrapolations. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2021.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2015.

Davide Buffelli, Pietro Liò, and Fabio Vandin. Sizeshiftreg: a regularization method for improving size-generalization in graph neural networks. In *Proc. Adv. in Neural Information Processing Systems (NeurIPS)*, 2022.

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Velickovic. Principal neighbourhood aggregation for graph nets. In *NeurIPS*, 2020.

Nicki Skafte Detlefsen, Jiri Borovec, Justus Schock, Ananya Harsh, Teddy Koker, Luca Di Liello, Daniel Stancl, Changsheng Quan, Maxim Grechkin, and William Falcon. Torchmetrics - measuring reproducibility in pytorch, 2022. URL https://github.com/Lightning-AI/torchmetrics.

Mucong Ding, Kezhi Kong, Jiuhai Chen, John Kirchenbauer, Micah Goldblum, David Wipf, Furong Huang, and Tom Goldstein. A closer look at distribution shifts and out-of-distribution generalization on graphs. In *NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications*, 2021.

Vijay Prakash Dwivedi, Chaitanya K. Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *CoRR*, 2020.

Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2022a.

Vijay Prakash Dwivedi, Ladislav Rampásek, Michael Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. Long range graph benchmark. In *Proc. Adv. in Neural Information Processing Systems NeurIPS, Datasets and Benchmark Track*, 2022b.

Saurabh Garg, Sivaraman Balakrishnan, Zachary C. Lipton, Behnam Neyshabur, and Hanie Sedghi. Leveraging unlabeled data to predict out-of-distribution performance. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2022.

Thomas Gaudelet, Ben Day, Arian R. Jamasb, Jyothish Soman, Cristian Regep, Gertrude Liu, Jeremy B. R. Hayter, Richard Vickers, Charles Roberts, Jian Tang, David Roblin, Tom L. Blundell, Michael M. Bronstein, and Jake P. Taylor-King. Utilising graph machine learning within drug discovery and development. *CoRR*, abs/2012.05716, 2020.

Shurui Gui, Xiner Li, Limei Wang, and Shuiwang Ji. GOOD: A graph out-of-distribution benchmark. In *Proc. Adv. in Neural Information Processing Systems (NeurIPS), Benchmark Track*, 2022.

Devin Guillory, Vaishaal Shankar, Sayna Ebrahimi, Trevor Darrell, and Ludwig Schmidt. Predicting with confidence on unseen distributions. In *ICCV*, 2021.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proc. of the Int. Conf. on Machine Learning, (ICML)*, 2017.

Kartik Gupta, Amir Rahimi, Thalaiyasingam Ajanthan, Thomas Mensink, Cristian Sminchisescu, and Richard Hartley. Calibration of neural networks using splines. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2021.

Arman Hasanzadeh, Ehsan Hajiramezanali, Shahin Boluki, Mingyuan Zhou, Nick Duffield, Krishna Narayanan, and Xiaoning Qian. Bayesian graph neural networks with adaptive connection sampling. In *ICML*, 2020.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2017.

Dan Hendrycks, Mantas Mazeika, and Thomas G. Dietterich. Deep anomaly detection with outlier exposure. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2019.

Dan Hendrycks, Nicholas Carlini, John Schulman, and Jacob Steinhardt. Unsolved problems in ML safety. *CoRR*, abs/2109.13916, 2021.

Dan Hendrycks, Steven Basart, Mantas Mazeika, Andy Zou, Joseph Kwon, Mohammadreza Mostajabi, Jacob Steinhardt, and Dawn Song. Scaling out-of-distribution detection for real-world settings. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2022a.

Dan Hendrycks, Andy Zou, Mantas Mazeika, Leonard Tang, Bo Li, Dawn Song, and Jacob Steinhardt. Pixmix: Dreamlike pictures comprehensively improve safety measures. In *Proc. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022b.

José Miguel Hernández-Lobato and Ryan P. Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2015.

Hans Hao-Hsun Hsu, Yuesong Shen, Christian Tomani, and Daniel Cremers. What makes graph neural networks miscalibrated? In *Proc. Adv. in Neural Information Processing Systems NeurIPS*, 2022.

Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Mach. Learn.*, 110(3), 2021.

Yiding Jiang, Dilip Krishnan, Hossein Mobahi, and Samy Bengio. Predicting the generalization gap in deep networks with margin distributions. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Konstantin Kirchheim, Marco Filax, and Frank Ortmeier. Pytorch-ood: A library for out-of-distribution detection based on pytorch. In *Workshop at the Proc. Int. Conf. on Computer Vision and Pattern Recognition CVPR*, 2022.

Boris Knyazev, Graham W. Taylor, and Mohamed R. Amer. Understanding attention and generalization in graph neural networks. In *Proc. Adv. in Neural Information Processing Systems (NeurIPS)*, 2019.

Ranganath Krishnan and Omesh Tickoo. Improving model calibration with accuracy versus uncertainty optimization. 2020.

Meelis Kull, Miquel Perelló-Nieto, Markus Kängsepp, Telmo de Menezes e Silva Filho, Hao Song, and Peter A. Flach. Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with dirichlet calibration. In *Proc. Adv. in Neural Information Processing Systems NeurIPS*, 2019.

Ananya Kumar, Percy Liang, and Tengyu Ma. Verified uncertainty calibration. In *Proc. Adv. in Neural Information Processing Systems NeurIPS*, 2019.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proc. Adv. in Neural Information Processing Systems (NeurIPS)*, 2017.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Proc. Adv. in Neural Information Processing Systems NeurIPS*, 2018.

Weitang Liu, Xiaoyun Wang, John D. Owens, and Yixuan Li. Energy-based out-of-distribution detection. In *Proc. Adv. in Neural Information Processing Systems NeurIPS*, 2020.

Matthias Minderer, Josip Djolonga, Rob Romijnders, Frances Hubis, Xiaohua Zhai, Neil Houlsby, Dustin Tran, and Mario Lucic. Revisiting the calibration of modern neural networks. In *Proc. Adv. in Neural Information Processing Systems (NeurIPS)*, 2021.

Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020. URL www.graphlearning.io.

Luis Müller, Mikhail Galkin, Christopher Morris, and Ladislav Rampásek. Attending to graph transformers. *CoRR*, abs/2302.04181, 2023.

Mahdi Pakdaman Naeini, Gregory F. Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Proc. Conf. on Adv. of Artificial Intelligence (AAAI)*, 2015.

Aviv Netanyahu, Abhishek Gupta, Max Simchowitz, Kaiqing Zhang, and Pulkit Agrawal. Learning to extrapolate: A transductive approach. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2023.

Nathan Ng, Neha Hulkund, Kyunghyun Cho, and Marzyeh Ghassemi. Predicting out-of-domain generalization with local manifold smoothness. *CoRR*, abs/2207.02093, 2022.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D. Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Proc. Adv. in Neural Information Processing Systems NeurIPS*, 2019.

Amir Rahimi, Amirreza Shaban, Ching-An Cheng, Richard Hartley, and Byron Boots. Intra order-preserving functions for calibration of multi-class neural networks. *Advances in Neural Information Processing Systems*, 33:13456–13467, 2020.

Mrinank Sharma, Sebastian Farquhar, Eric Nalisnick, and Tom Rainforth. Do bayesian neural networks need to be fully stochastic? In *AISTATS*, 2023.

Jayaraman J. Thiagarajan, Rushil Anirudh, Vivek Narayanaswamy, and Peer-Timo Bremer. Single model uncertainty estimation via stochastic data centering. In *Proc. Adv. in Neural Information Processing Systems (NeurIPS)*, 2022.

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *Proc. Int. Conf. on Learning Representations ICLR*, 2022.

Puja Trivedi, Danai Koutra, and Jayaraman J Thiagarajan. A closer look at scoring functions and generalization prediction. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2023a.

Puja Trivedi, Danai Koutra, and Jayaraman J. Thiagarajan. A closer look at model adaptation using feature distortion and simplicity bias. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2023b.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

Haoqi Wang, Zhizhong Li, Litong Feng, and Wayne Zhang. Vim: Out-of-distribution with virtual-logit matching. In *Proc. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022a.

Haorui Wang, Haoteng Yin, Muhan Zhang, and Pan Li. Equivariant and stable positional encoding for more powerful graph neural networks. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2022b.

Xiao Wang, Hongrui Liu, Chuan Shi, and Cheng Yang. Be confident! towards trustworthy graph neural networks via confidence calibration. In *Proc. Adv. in Neural Information Processing Systems NeurIPS*, 2021.

Olivia Wiles, Sven Gowal, Florian Stimberg, Sylvestre-Alvise Rebuffi, Ira Ktena, Krishnamurthy Dj Dvijotham, and Ali Taylan Cemgil. A Fine-Grained Analysis on Distribution Shift. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2022.

Andrew Gordon Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. In *Proc. Adv. in Neural Information Processing Systems NeurIPS*, 2020.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.

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*, pp. 11975–11986. PMLR, 2021.

Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 694–699, 2002.

Jize Zhang, Bhavya Kailkhura, and Thomas Yong-Jin Han. Mix-n-match : Ensemble and compositional methods for uncertainty calibration in deep learning. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2020.

Yingxue Zhang, Soumyasundar Pal, Mark Coates, and Deniz Üstebay. Bayesian graph convolutional neural networks for semi-supervised classification. In *AAAI*, 2019.

Yanqiao Zhu, Yuanqi Du, Yinkai Wang, Yichen Xu, Jieyu Zhang, Qiang Liu, and Shu Wu. A survey on deep graph generation: Methods and applications. In *Learning on Graphs Conference (LoG)*, 2022.