Appendix

Anonymous Authors

APPENDIX

A. Overview of the Appendix

To guide the reader through the appendix, we outline the content and structure below.

- **Appendix A: Dataset Descriptions.** We describe the motivations behind the datasets choice, highlighting their characteristics, and the edge distributions both for the negatives and positives.
- **Appendix A: Experimental Set-up: Additional Details.** Here, we descibe additional details on our experiments, such as the hyperparameter space, and the details related to the hardware that we used and also on the optimization strategy.
- **Appendix C: Additional Results.** This section presents extended results from our experiments, including several examples of attraction and repulsion among gradients, and also bar plots to evaluate how the models predict heterophilic edges w.r.t. homophilic ones.
- **Appendix G: Extended Related Works.** Here we include a thorough discussion of link prediction methods, describing how the state-of-the-art practices have evolved during the years.

DATASETS DESCRIPTION

In this Section of the supplementary materials we provide information on the datasets that we used, how they have been split for training and evaluation.

We first start with a description of the datasets, that also help to understand why they are heterophilic graphs. **`Amazon Ratings:`** Nodes represent products, with edges linking items frequently bought together. Node classes denote product ratings. The goal in link prediction is to anticipate likely co-purchases.

`Roman Empire:` This graph is built from the Roman Empire's Wikipedia article [1], with nodes as non-unique words. Links exist if words are connected in dependency trees or appear sequentially in text, forming a nearly chain-like structure with shortcuts. Node classes indicate syntactic roles, and link prediction involves reconstructing chain structure and syntactic dependencies.

`Minesweeper:` This synthetic 100x100 grid has cells as nodes, each linked to at most 8 neighbors. Cells are classified as mines or traversable, and link prediction aims to capture grid structure amid varying cell types.

`Questions:` Based on a Q&A website, nodes are users connected by interaction over time. Users are classified as active or inactive. Link prediction consists of prediction what users are likely to interact.

`Tolokers:` This dataset is based on data from a crowdsourcing platform. Nodes represent workers that have participated in at least one of 13 selected projects. Labels for each node are binary, identifying what workers have been banned from a project. An edge connects two tolokers if they have worked on the same task. Thus, link prediction have the objective to predict what worker will likely collaborate together.

The modality of the experiments follow the transductive link prediction paradigm. Graphs have been split into training, validation, and test positive edges (N_{pos}) with 80%, 10%, and 10% percentages. Negative edges (N_{neg}) for each split were also sampled, and in the experiments, we adopted several ratios N_{neg}/N_{pos} since there is not a specific policy to follow when selecting N_{neg} [2]. We select negative edges through the random negative sampling routine implemented by PyTorch Geometric [3]. As concerns, the number of evaluation edges, and those contained in the training, validation and test sets Table I showcases the proportion of the edges in our experimental set-up. We defined the split in order to avoid data leakage among the edges in the set, in particular we followed the specifics established by [4]. The negatives were sampled more, as we see in Table I, since we use the number of negatives as an hyperparameter. Regarding the evaluation edges, we ensured an equal balance of negative and positive samples. The total number of edges and how they differ in terms of homophilic and heterophilic is displayed in Table II.

In this Section, we report additional details on the experimental set-up, such as the metrics, optimization algorithm, and the hyperparameters.

B. Implementation Details

We chose AUROC since is commonly used in link prediction tasks with GNNs [5], [6]. We trained them using negative log-likelihood. The loss function is optimized using Adam [7] with early stopping [8], with patience of 300 epochs, monitoring the AUROC. The experiments were run on a single Nvidia GeForce RTX 3090 Ti 24GB.

Identify applicable funding agency here. If none, delete this.

Dataset	Splits	Message Passing Edges	Positive Edges	Negative Edges
Minesweeper	**train**	50,436	6,304	1,260,800
	val	63,044	3,940	777,933
	test	70,924	3,940	771,958
Amazon Ratings	**train**	119,104	14,888	2,977,600
	val	148,880	9,305	1,851,634
	test	167,490	9,305	1,845,817
Questions	**train**	196,532	24,566	4,913,200
	val	245,664	15,354	3,064,559
	test	276,372	15,354	3,060,429
Roman Empire	**train**	42,150	5,268	1,053,600
	val	52,686	3,292	657,070
	test	59,270	3,292	656,211
Tolokers	**train**	664,320	83,040	16,608,000
	val	830,400	51,900	10,380,000
	test	934,200	51,900	10,380,000

TABLE I: Dataset Statistics for Message Passing Edges, Positive Edges, and Negative Edges.

Dataset	Positives $\mathcal{E}_{hm}$	Positives $\mathcal{E}_{ht}$	Negatives $\mathcal{E}_{hm}$	Negatives $\mathcal{E}_{ht}$	$\lvert\mathcal{E}_{pos}\rvert = \lvert\mathcal{E}_{neg}\rvert$
Amazon Ratings	3,507	5,798	2,483	6,822	9,305
Roman Empire	122	3,170	301	2,991	3,292
Minesweeper	2,672	1,268	2,534	1,406	3,940
Questions	12,906	2,448	14,456	898	15,354
Tolokers	30,731	21,169	33,354	18,546	51,900

TABLE II: Statistics for Positives and Negatives in $\mathcal{E}_{hm}$ and $\mathcal{E}_{ht}$ categories. This refers to the test edges for which we report the main results in the paper.

C. Hyperparameters

We considered several hyperparameters in our model, including the learning rate α, weight decay γ, and hidden dimension d_h, which was kept constant across all layers during the message-passing phase. We also adjusted the hidden dimension of the decoder d_{MLP} for the decoding phase. The dropout rates for the encoding and decoding phases, denoted as ρ and ρ_{MLP} respectively, were optimized. Additionally, we tuned the number of layers for message passing L and decoding L_{MLP}, examined the use of batch normalization in the decoder, and considered the ratio of negative to positive samples $\frac{N_{neg}}{N_{pos}}$. In the GRAFF-LP experiment we also considered the value of the step size τ. Our hyperparameter space was the following.

Additional Results

D. Additional results on `Tolokers`

For space constraints, we report here the AUROC performance for `Tolokers` in Table IV. We can see that GNN models surpass the MLP performance, underlining the informative nature of edges in `Tolokers` for the link prediction task. GRAFF-LP ranks among the top-3 models coherently with the other datasets. However, here GraphSAGE is able to outperform the other baselines.

In Table V, we also report the gradient separability performance. In this case all the models have high gradient separability, and using our readout improves such separability measure, even though in `Tolokers`, the performance with the gradient readout is slightly worse than the hadamard product. However, we use this metric to understand whether GNNs are learning to separate edge gradients, which is confirmed by the consistent high value of GS^T, and also the gradient separability trend

Hyperparameter	Value
α	{0.01, 0.001}
γ	{0, 0.01, 0.001}
d_h	{128, 256}
d_{MLP}	{32, 64}
ρ	{0.1, 0.3, 0.5}
ρ_{MLP}	{0.1, 0.3, 0.5}
L	{1, 3, 5, 7, 9, 12}
L_{MLP}	{0, 1, 2}
Batch Norm.	{Yes, No}
$\frac{N_{neg}}{N_{pos}}$	{0.25, 0.5, 1, 2, 4, 8}
τ	{0.1, 0.25, 0.5}

TABLE III: Hyperparameter Space for Experiments.

of the test edge shown in Figure 12. We can see, that even with 3 layers, GRAFF-LP improves the separability in terms of gradients.

TABLE IV: Performance of models on the Tolokers dataset with f_h and f_g. Asterisks indicate statistical significance (p-value = {* $\rightarrow$ 0.01, ** $\rightarrow$ 0.05, *** $\rightarrow$ 0.1}). Text color refers to the first, second, and third model according to the mean.

Models	f_h	f_g
MLP	92.97 ± 1.14*	92.37 ± 0.73*
GCN	98.13 ± 0.28*	97.95 ± 0.17
SAGE	98.60 ± 0.26	98.73 ± 0.19
GAT	96.97 ± 0.32*	96.50 ± 0.14*
ELPH	90.09 ± 0.81*	0.0 ± 0.0
NCNC	-	0.0 ± 0.0
GRAFF-LP	98.24 ± 0.19	97.76 ± 0.03

TABLE V: Comparison of GS^T on the Tolokers dataset, when the model is trained with f_h or f_g. We report the percentage variation between the results of f_h and f_g as Δ.

Model	f_h	f_g	Δ
MLP	87.50 ± 0.54	91.74 ± 0.76	+4.55%
GCN	90.52 ± 0.34	92.01 ± 0.28	+1.64%
SAGE	90.52 ± 0.20	92.22 ± 0.26	+1.87%
GAT	89.12 ± 0.56	88.61 ± 0.0038	-0.57%
ELPH	83.73 ± 0.65	0.0 ± 0.0	-
NCNC	0.0 ± 0.0	0.0 ± 0.0	-
GRAFF-LP	90.27 ± 0.33	90.69 ± 0.14	+0.46%

E. Additional results on Homophilic Datasets

For sake of completeness, we decided to assess how GRAFF-LP ranks in the recently proposed HeaRT benchmark [9]. This is a benchmark for link prediction under homophily, where the positive and negative edges used in the evaluation are chosen in a way that simple heuristics may fail to predict them, by sampling hard positives and negatives. This benchmark shed light on how node-based methods are not significantly worse than subgraph-based methods, or those approaches based on structural features. This gap reduction underlines how the datasets that are typically used in benchmarks do not require advanced methods like NCNC or ELPH for high performance, which is also what we observed in our experiments. We used the same hyperparameters set proposed in [9], and the best configuration result on the test set are presented in Table VI. We report the original table from [9], so we have additional baselines, and those that are affine such as GCN, GAT and GraphSAGE, are implemented differently from ours. Details on implementation can be found in our code, as well as the HeaRT repository to reproduce their experiments as well. In Table VI, we can see that also within homophilic link prediction GRAFF-LP achieve competitive performance, specifically in PubMed, where it sets the new state-of-the-art. While in the other datasets, GRAFF-LP ranks in the top-3 only in Citeseer, leaving a significant gap in Cora. Further research is required to better understand the meaning of homophily in link prediction, and how edge gradients separate in these settings.

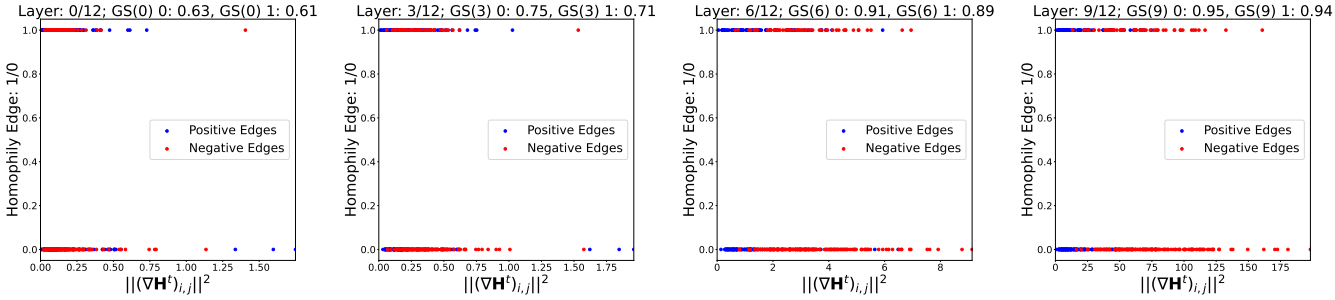
TABLE VI: Results on Cora, Citeseer, and Pubmed (%) under HeaRT. Highlighted are the results ranked **first**, **second**, and **third**.

Models	Cora		Citeseer		Pubmed	
	MRR	Hits@10	MRR	Hits@10	MRR	Hits@10
Heuristic						
CN	9.78	20.11	8.42	18.68	2.28	4.78
AA	11.91	24.10	10.82	22.20	2.63	5.51
RA	11.81	24.48	10.84	22.86	2.47	4.90
Shortest Path	5.04	15.37	5.83	16.26	0.86	0.38
Katz	11.41	22.77	11.19	24.84	3.01	5.98
Embedding						
Node2Vec	14.47 ± 0.60	32.77 ± 1.29	21.17 ± 1.01	45.82 ± 2.01	3.94 ± 0.24	8.51 ± 0.77
MF	6.20 ± 1.42	15.26 ± 3.39	7.80 ± 0.79	16.72 ± 1.99	4.46 ± 0.32	9.42 ± 0.87
MLP	13.52 ± 0.65	31.01 ± 1.71	22.62 ± 0.55	48.02 ± 1.79	6.41 ± 0.25	15.04 ± 0.67
GNN						
GCN	**16.61 ± 0.30**	**36.26 ± 1.14**	21.09 ± 0.88	47.23 ± 1.88	7.13 ± 0.27	15.22 ± 0.57
GAT	13.84 ± 0.68	32.89 ± 1.27	19.58 ± 0.84	45.30 ± 1.30	4.95 ± 0.14	9.99 ± 0.64
SAGE	14.74 ± 0.69	34.65 ± 1.47	21.09 ± 1.15	48.75 ± 1.85	**9.40 ± 0.70**	**20.54 ± 1.40**
GAE	**18.32 ± 0.41**	**37.95 ± 1.24**	25.25 ± 0.82	49.65 ± 1.48	5.27 ± 0.25	10.50 ± 0.46
GNN+Pairwise Info						
SEAL	10.67 ± 3.46	24.27 ± 6.74	13.16 ± 1.66	27.37 ± 3.20	5.88 ± 0.53	12.47 ± 1.23
BUDDY	13.71 ± 0.59	30.40 ± 1.18	22.84 ± 0.36	48.35 ± 1.18	7.56 ± 0.18	16.78 ± 0.53
Neo-GNN	13.95 ± 0.39	31.27 ± 0.72	17.34 ± 0.84	41.74 ± 1.18	7.74 ± 0.30	17.88 ± 0.71
NCN	14.66 ± 0.95	35.14 ± 1.04	**28.65 ± 1.21**	**53.41 ± 1.46**	5.84 ± 0.22	13.22 ± 0.56
NCNC	14.98 ± 1.00	**36.70 ± 1.57**	24.10 ± 0.65	**53.72 ± 0.97**	8.58 ± 0.59	18.81 ± 1.16
NBFNet	13.56 ± 0.58	31.12 ± 0.75	14.29 ± 0.80	31.39 ± 1.34	¿24h	
PEG	**15.73 ± 0.39**	36.03 ± 0.75	21.01 ± 0.77	45.56 ± 1.38	4.40 ± 0.41	8.70 ± 1.26
Physics-Inspired GNN						
GRAFF-LP (Hadamard)	15.73 ± 0.77	34.76 ± 1.13	**26.77 ± 1.1**	**51.76 ± 1.68**	**13.49 ± 0.8**	**27.20 ± 0.84**
GRAFF-LP (Gradient)	13.75 ± 0.66	31.56 ± 1.57	**25.7 ± 1.32**	49.98 ± 1.1	**12.29 ± 0.79**	**26.46 ± 2.69**

F. Additional Examples of Attraction and Repulsion through GS^T

Here we show some example where GRAFF-LP is able to learn separating the gradients. We have example both with f_h as well as with f_g. These examples helps to answer more profoundly to **RQ2**.

Fig. 1: $||(\nabla \mathbf{H}^t)_{i,j}||^2$ evolution with a fully-trained 9-layers GRAFF-LP via f_g on `Amazon Ratings`.

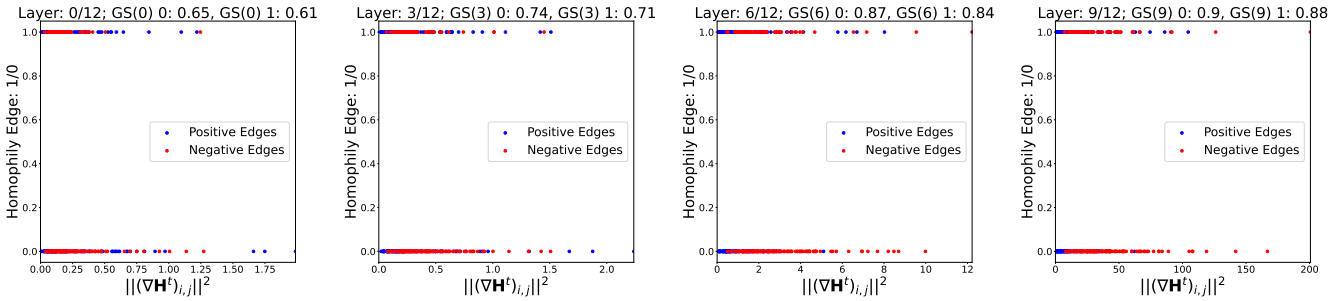


(a) $GS^0_{hm,hm}$, $GS^0_{ht,ht}$ (b) $GS^3_{hm,hm}$, $GS^3_{ht,ht}$. (c) $GS^6_{hm,hm}$, $GS^6_{ht,ht}$ (d) $GS^9_{hm,hm}$, $GS^9_{ht,ht}$

In Figures 1, 2, we notice that with f_g and even with f_h, GRAFF-LP have learnt to separate the edge gradients in `Amazon Ratings`. Of course, w.r.t. the paper results, f_h provide a lighter separability. We can see the same for `Minesweeper` in Figure 3. Another interesting behavior is GRAFF-LP with f_h in `Roman Empire`, where its GS^T is lower than 60%, and indeed we see that it evolves as in Figure 4. However, if we simply train it with f_g, we get $GS^T > 90\%$, as illustrated in Figure 5.

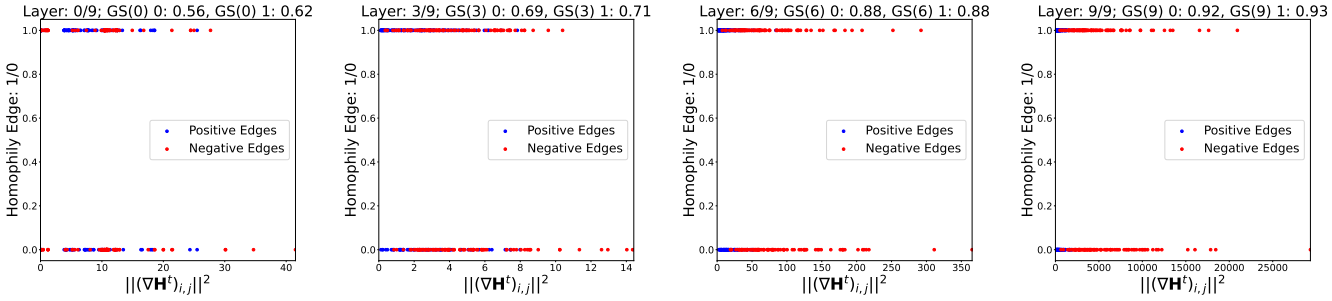
Unfortunately, even though GRAFF-LP presents the right inductive bias to learn this behavior, we have an instance where we do not find this. In particular we have this for `Questions`, as we see in Figure 6. Here we find that the negative gradients are pushed downward w.r.t. the positives that are pushed upward. From the Figure, we conjecture that it may be related to the edge gradient initialization, in Figure 13a, we have the positive edges that reach a maximum of 0.035, then this values

Fig. 2: $||(\nabla \mathbf{H}^t)_{i,j}||^2$ evolution with a fully-trained 9-layers GRAFF-LP via f_h on `Amazon Ratings`.



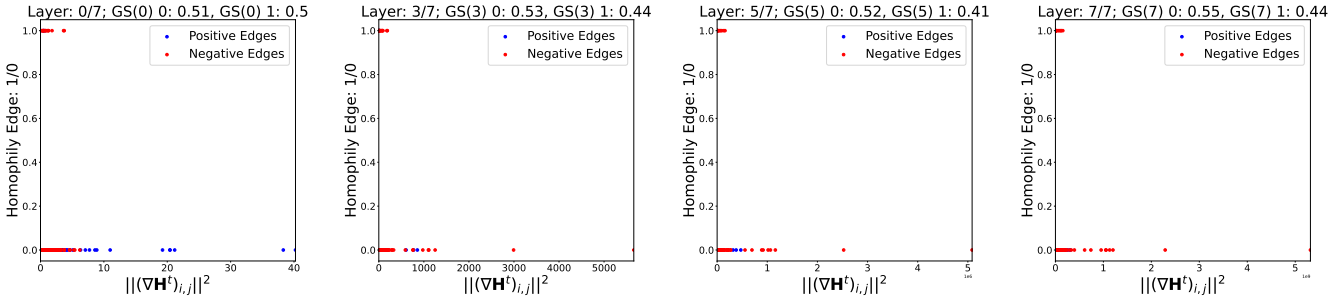
(a) $GS^0_{hm,hm}$, $GS^0_{ht,ht}$ (b) $GS^3_{hm,hm}$, $GS^3_{ht,ht}$. (c) $GS^6_{hm,hm}$, $GS^6_{ht,ht}$ (d) $GS^9_{hm,hm}$, $GS^9_{ht,ht}$

Fig. 3: $||(\nabla \mathbf{H}^t)_{i,j}||^2$ evolution with a fully-trained 9-layers GRAFF-LP via f_h on `minesweeper`.



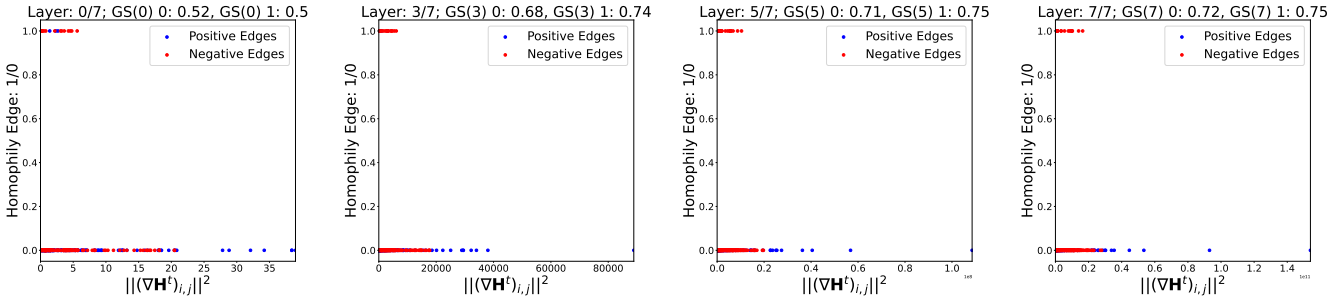
(a) $GS^0_{hm,hm}$, $GS^0_{ht,ht}$ (b) $GS^3_{hm,hm}$, $GS^3_{ht,ht}$. (c) $GS^6_{hm,hm}$, $GS^6_{ht,ht}$ (d) $GS^9_{hm,hm}$, $GS^9_{ht,ht}$

Fig. 4: $||(\nabla \mathbf{H}^t)_{i,j}||^2$ evolution with a fully-trained 7-layers GRAFF-LP via f_h on `Roman Empire`.



(a) $GS^0_{hm,hm}$, $GS^0_{ht,ht}$ (b) $GS^3_{hm,hm}$, $GS^3_{ht,ht}$. (c) $GS^5_{hm,hm}$, $GS^5_{ht,ht}$ (d) $GS^7_{hm,hm}$, $GS^7_{ht,ht}$

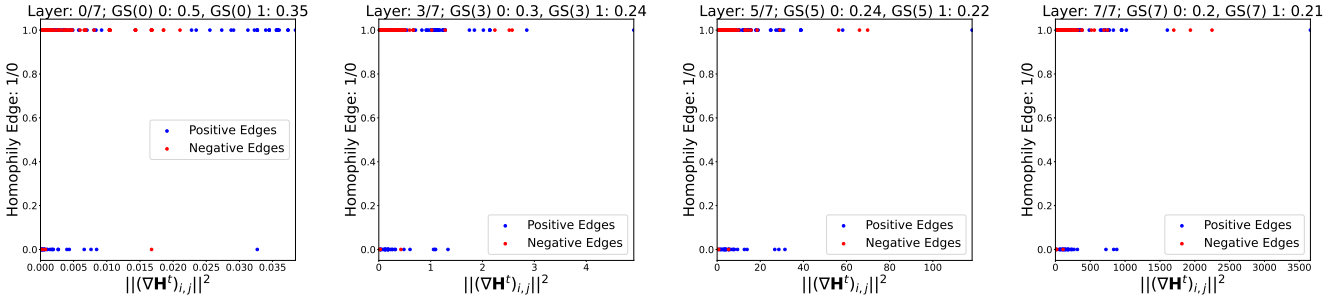
Fig. 5: $||(\nabla \mathbf{H}^t)_{i,j}||^2$ evolution with a fully-trained 7-layers GRAFF-LP via f_g on `Roman Empire`.



(a) $GS^0_{hm,hm}$, $GS^0_{ht,ht}$ (b) $GS^3_{hm,hm}$, $GS^3_{ht,ht}$. (c) $GS^5_{hm,hm}$, $GS^5_{ht,ht}$ (d) $GS^7_{hm,hm}$, $GS^7_{ht,ht}$

increases both for negatives as well as positives. To better visualize and understand this behavior, we report the distributions of the squared norm gradients according to the different layers of the GNN. We show the distribution for `Questions` in Figure 9. We see, that as inferred from Figure 6, the negative gradients are initialized to a lower score w.r.t. the positives, and as the network evolves the features they get separated accordingly. For sake of completeness, we report the distribution of the squared norm gradients of `Amazon Ratings` and `Roman Empire` in Figures 10, 11. Now we illustrate also some cases

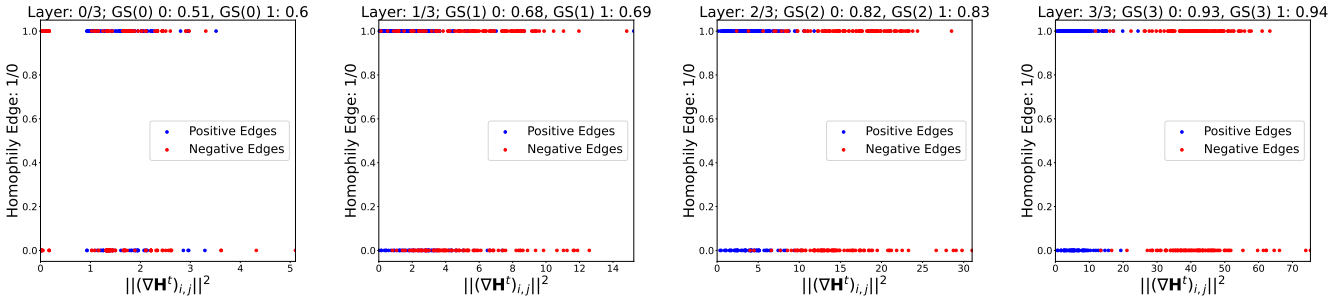
Fig. 6: $||(\nabla \mathbf{H}^t)_{i,j}||^2$ evolution with a fully-trained 7-layers GRAFF-LP via f_g on `Questions`.



(a) $GS^0_{hm,hm}$, $GS^0_{ht,ht}$ (b) $GS^3_{hm,hm}$, $GS^3_{ht,ht}$. (c) $GS^6_{hm,hm}$, $GS^6_{ht,ht}$ (d) $GS^7_{hm,hm}$, $GS^7_{ht,ht}$

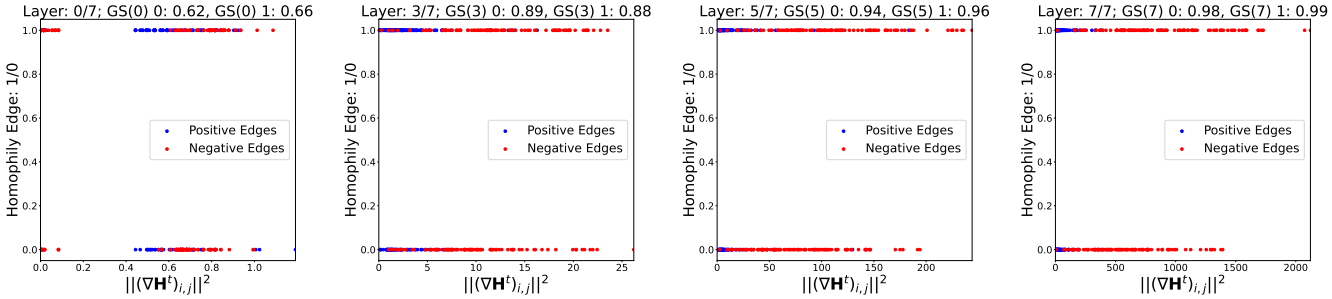
where the gradient separability is learned when the inductive bias is not present, but f_g let this happen anyway. These cases are interesting for GCN and GAT in `Minesweeper`. In Figure 7, we have GCN, while in Figure 8 we have GAT.

Fig. 7: $||(\nabla \mathbf{H}^t)_{i,j}||^2$ evolution with a fully-trained 3-layers GAT via f_g on `Minesweeper`.



(a) $GS^0_{hm,hm}$, $GS^0_{ht,ht}$ (b) $GS^1_{hm,hm}$, $GS^1_{ht,ht}$. (c) $GS^2_{hm,hm}$, $GS^2_{ht,ht}$ (d) $GS^3_{hm,hm}$, $GS^3_{ht,ht}$

Fig. 8: $||(\nabla \mathbf{H}^t)_{i,j}||^2$ evolution with a fully-trained 7-layers GAT via f_g on `Minesweeper`.



(a) $GS^0_{hm,hm}$, $GS^0_{ht,ht}$ (b) $GS^3_{hm,hm}$, $GS^3_{ht,ht}$. (c) $GS^6_{hm,hm}$, $GS^6_{ht,ht}$ (d) $GS^7_{hm,hm}$, $GS^7_{ht,ht}$

Finally we show that learning to separate gradients is not the only hypothesis that can be learnt to achieve high performance. ELPH has the highest score in `Minesweeper`, since it can extract the features that tells it that the graph is a grid, and then the link prediction task is easy. Since ELPH do not need to separate the gradient we expect a low GS^T, which is what we observe in Figure 13.

Fig. 9: `Questions`: Edge Gradients Distribution, at the end of each message-passing phase of a 7-layers fully-trained GRAFF-LP via f_g.

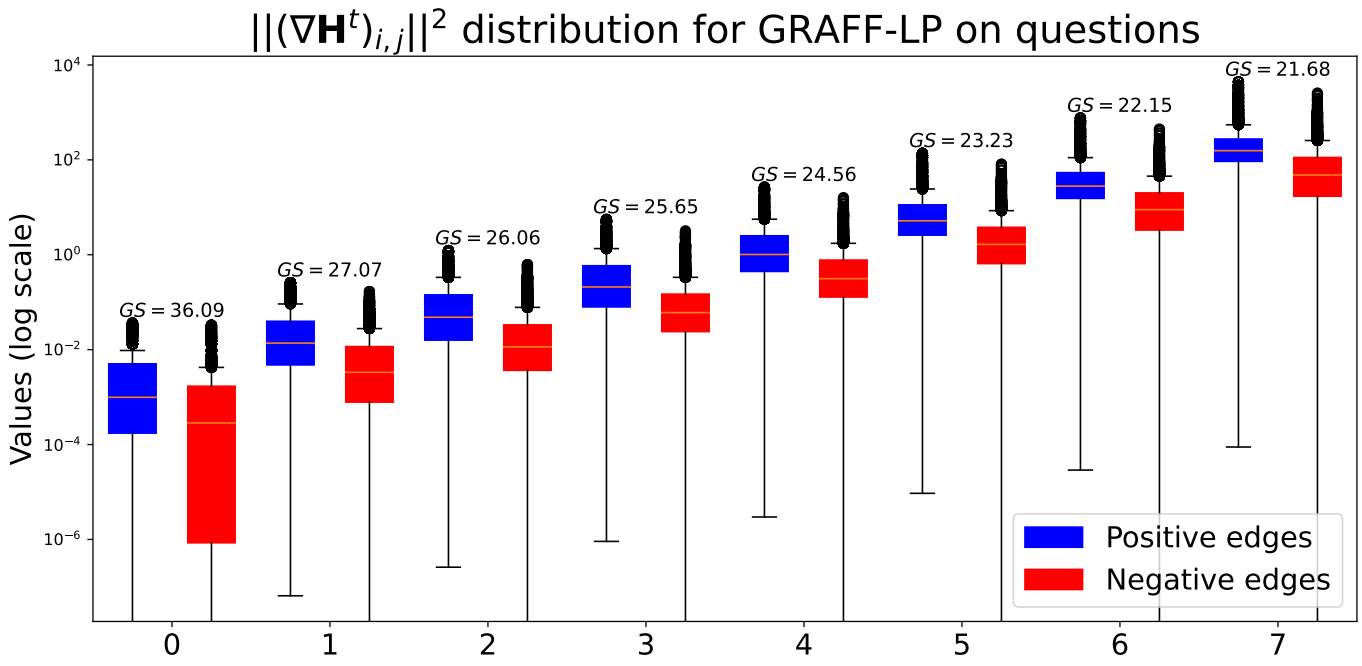


Fig. 10: `Amazon Ratings`: Edge Gradients Distribution, at the end of each message-passing phase of a 12-layers fully-trained GRAFF-LP via f_g.

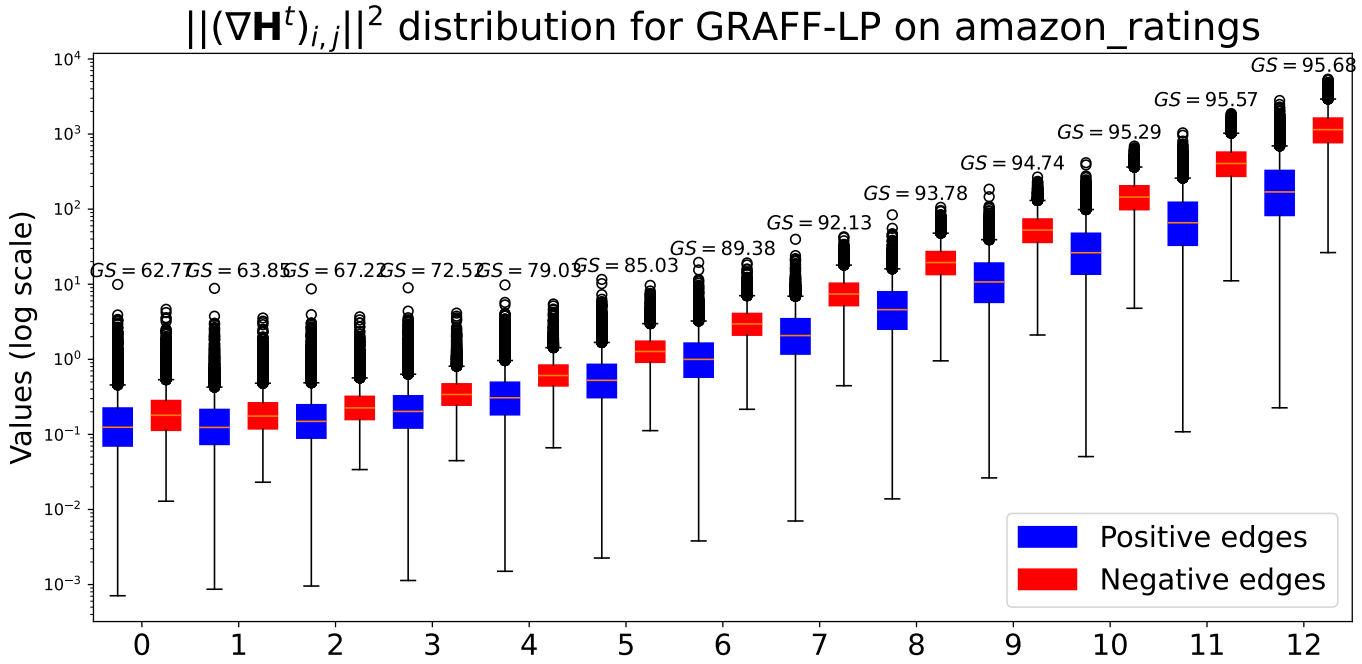


Fig. 11: `Roman Empire`: Edge Gradients Distribution, at the end of each message-passing phase of a 7-layers fully-trained GRAFF-LP via f_g.

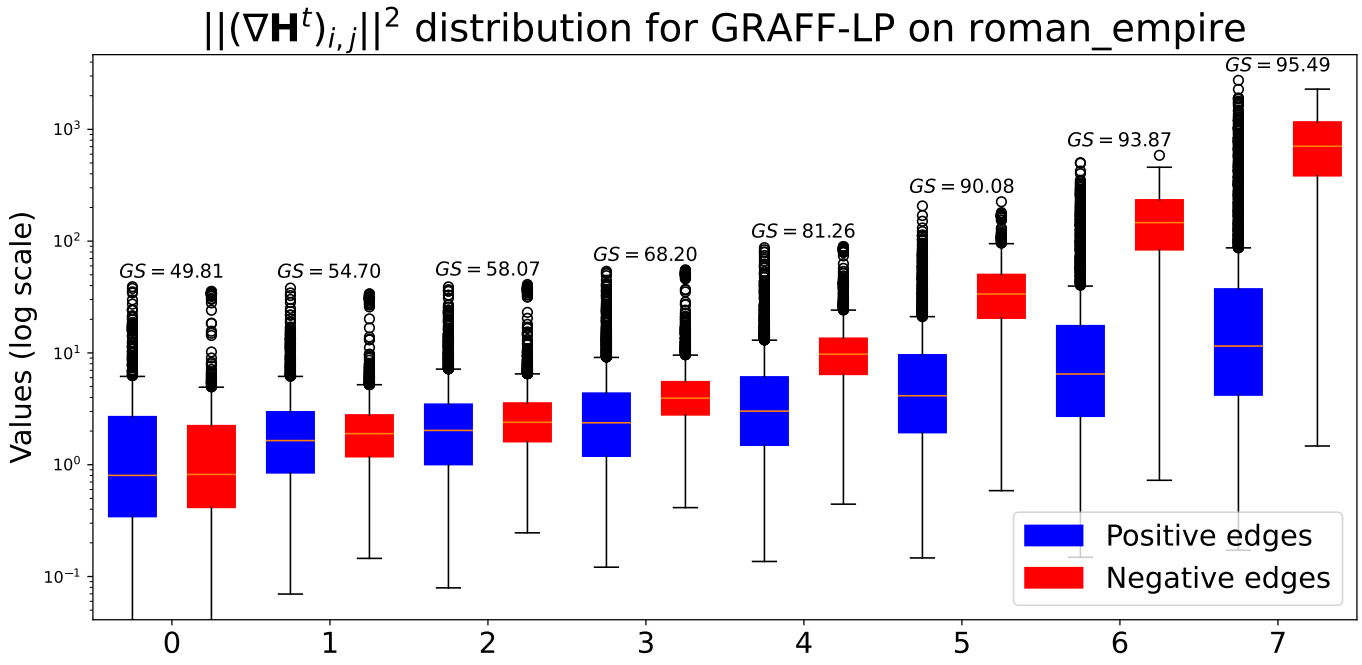


Fig. 12: `Questions`: Edge Gradients Distribution, at the end of each message-passing phase of a 3-layers fully-trained GRAFF-LP via f_g.

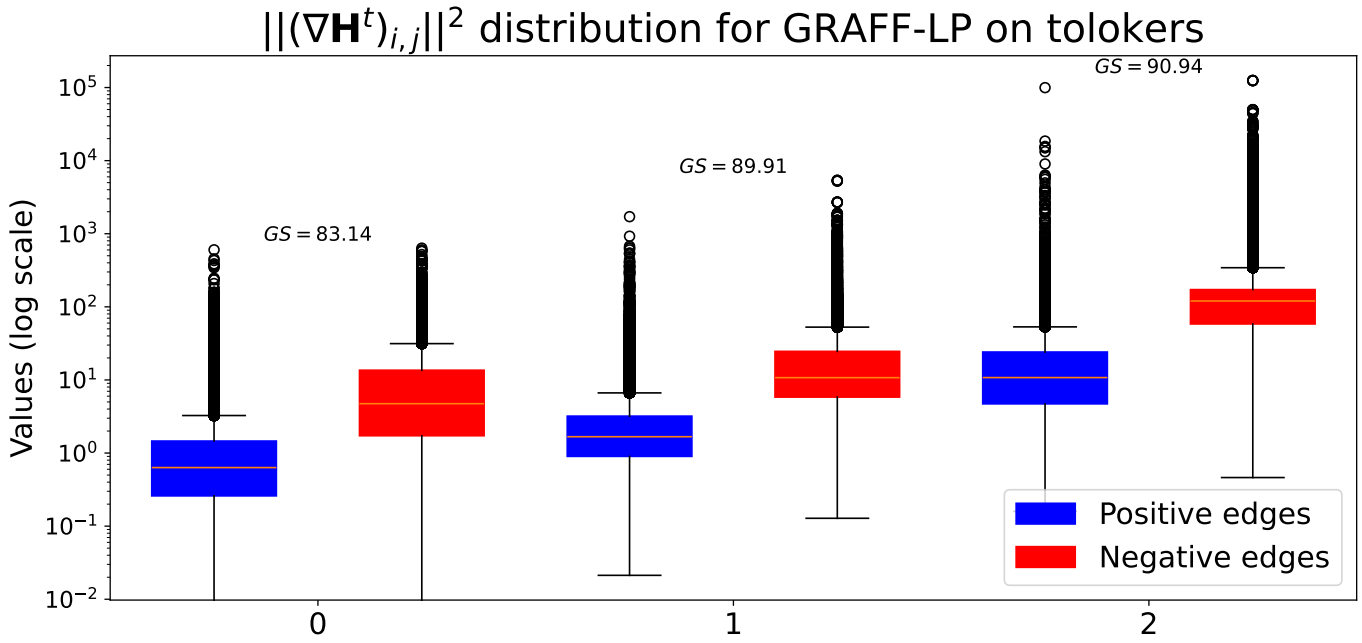
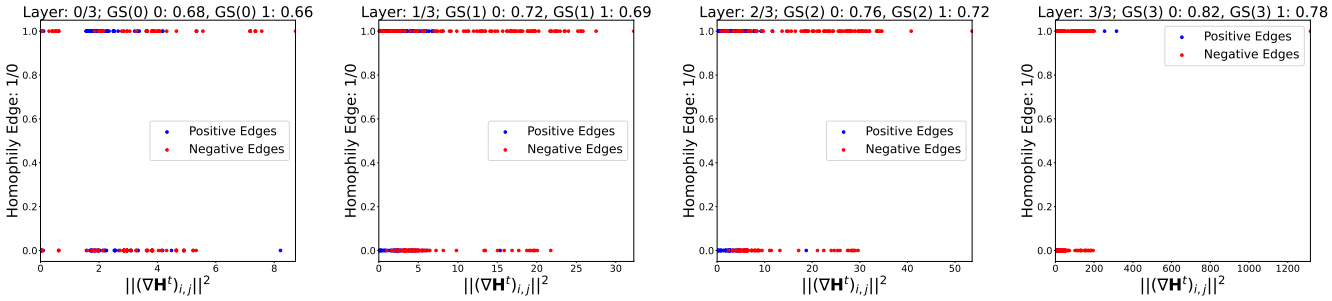


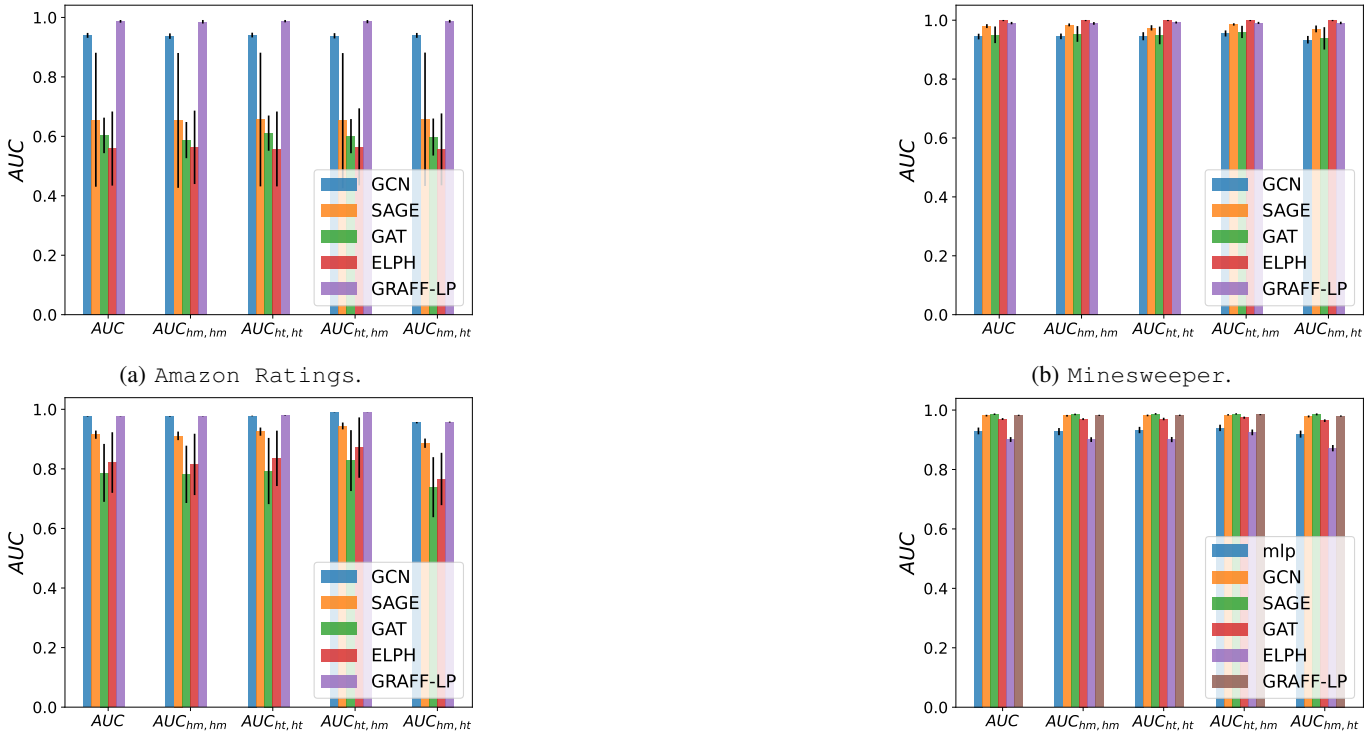
Fig. 13: $||(\nabla \mathbf{H}^t)_{i,j}||^2$ evolution with a fully-trained 3-layers ELPH via f_h on `Minesweeper`.



(a) $GS^0_{hm,hm}$, $GS^0_{ht,ht}$ (b) $GS^1_{hm,hm}$, $GS^1_{ht,ht}$. (c) $GS^2_{hm,hm}$, $GS^2_{ht,ht}$ (d) $GS^3_{hm,hm}$, $GS^3_{ht,ht}$

G. Additonal Examples of robustness to Heterophilic and Homophilic edges

In Figures 14a and 14b, 14c, we have additional evidences that all the models do not struggle explicitely to predict an edge because of the node classes associated.



(a) `Amazon Ratings`. (b) `Minesweeper`.

(c) `Questions`. (d) `Tolokers`.

Fig. 14: Comparison of model performance on different datasets. Here we expose the ability of the models to predict homophilic edges or heterophilic ones, both as negatives or positives.

In the main paper we reported the runtime analysis, and number of parameters required by each baselines, comprised of GRAFF-LP. We showd that GRAFF-LP have time complexity comparable with the node-based methods, and has advantage in terms of memory requirements thanks to the weight sharing property. For completeness we report the other analysis on the remaining datasets in Table VII. As we can observe there are not specific difference with the results presented in the main paper.

TABLE VII: Comparison of model performance across datasets, showing the number of parameters and runtime (in seconds) for each model. The inference time is averaged across 10 trials. OOM means out of memory.

Model	**Roman Empire**		**Questions**		**Tolokers**	
	Parameters	**Runtime (s)**	**Parameters**	**Runtime (s)**	**Parameters**	**Runtime (s)**
MLP	32256	0.0588 ± 0.01	32320	0.1539 ± 0.01	13696	0.218 ± 0.01
GCN	31680	0.0797 ± 0.01	31744	0.1514 ± 0.01	13120	0.3835 ± 0.04
GAT	32064	0.0503 ± 0.00	32128	0.1528 ± 0.01	13504	0.4612 ± 0.07
SAGE	43968	0.0766 ± 0.01	44032	0.1486 ± 0.01	25408	0.4705 ± 0.02
ELPH	40542	0.4953 ± 0.03	40606	1.6236 ± 0.06	21982	2.6042 ± 0.24
NCNC	27584	0.1231 ± 0.02	27648	0.5205 ± 0.01	OOM	OOM
GRAFF-LP (f_h)	23617	0.0633 ± 0.00	23681	0.1513 ± 0.01	5057	0.4453 ± 0.04
GRAFF-LP (f_g)	23617	0.0728 ± 0.02	23681	0.1470 ± 0.01	5057	0.4290 ± 0.02

EXTENDED RELATED WORKS

Graph Neural Networks for link prediction. Over the years, link prediction algorithms have evolved and can easily be distinguished between *non-neural-based* and *neural-based* methods. The former mainly consists of heuristics [10]–[13] that rely on strong assumptions on the link prediction process. On the other hand, neural-based methods imply the use of learning systems, in particular GNNs. Even though some GNNs cannot be as expressive as most of the heuristics [5], they can learn graph structure features and content features in a unified way, outperforming previous works. An instance of this class are the node-based methods, where the objective is to learn the node representations in a vector form, and then estimate the link existence accordingly. Graph Auto-Encoders are an example of this class [14], and several variants have been proposed in recent years [15]–[17].

More recently, the *subgraph-based* paradigm, pioneered by SEAL [5], pushed the state-of-the-art beyond node-based methods. Computing subgraph representations increases GNN expressivity but makes this approach inefficient and impractical for real-world graphs. [18] and [19] tried to alleviate the efficiency-related issues using subgraph features. Despite these efforts, the node-based baselines remain a more efficient and scalable solution. Moreover, [9] have shown that the performance gap between these two families of models is not so enhanced when the training, validation and test positive and negative edges are accurately selected. For these reasons, we focused our analysis leveraging the node-based paradigm.

Link prediction methods have predominantly been compared within homophilic benchmarks, biasing progress in that direction. The heterophilic scenario became a subject of interest for link prediction only recently when [20] proposed an 'ad hoc' method to handle link prediction under heterophily. This approach outperforms previous node-based baselines but poorly scales to larger datasets because of the multiple set of features associated with each node.

Physics-Inspired vs. Physics-Informed. Physics-Inspired (PIrd) neural networks belong to the class of Physics-Informed (PI) neural networks. However, a preliminary distinction must be made for clarity's sake. Generally, the PI paradigm has the objective of providing priors to machine learning models to let them intuit the underlying physical process that can help to improve the task performance. These priors can benefit the neural network training in several ways: through better efficiency in training data requirements, faster training convergency, or the model's generalizability and interpretability [21]–[23]. The methodologies that have been employed to transfer such physical knowledge differ widely [21], [24]–[26], and take the form of different types of biases. Among this, we have a bias of the inductive type. Which is what we refer to as PIrd. Some of these have been proposed by [26], [27], [28], and [29].

Physics-Inspired Graph Neural Networks. The class of methods that can be associated with PIrd GNNs are models where the physics bias is encompassed within the network's architecture in the form of 'hard' constraints. From this perspective, we report some examples.

In the work by [28], the GNN is interpreted as a gradient flow, namely, its forward pass minimizes a parametrized energy functional, respecting the properties of gradient flows, thanks to symmetric weight matrices. In the paper by [29], GNNs resemble reaction-diffusion equations which are typically used to model the spatial and temporal change of one or more chemical substance concentrations. [30] proposed GNNs that behave as a non-dissipative system through the use of antisymmetric weight matrices, and followingly [31], it was shown that also a non-conservative behavior to retain the node information can be enabled

via architectural biases. These works are experimentally limited to node classification benchmarks, and no practical feedback on their application to link prediction is currently available in the literature. In our work, we provide the first perspective on PIrd biases applied in the context of link prediction.

REFERENCES

[1] Q. Lhoest, A. V. del Moral, Y. Jernite, A. Thakur, P. von Platen, S. Patil, J. Chaumond, M. Drame, J. Plu, L. Tunstall, J. Davison, M. Šaško, G. Chhablani, B. Malik, S. Brandeis, T. L. Scao, V. Sanh, C. Xu, N. Patry, A. McMillan-Major, P. Schmid, S. Gugger, C. Delangue, T. Matussière, L. Debut, S. Bekman, P. Cistac, T. Goehringer, V. Mustar, F. Lagunas, A. M. Rush, and T. Wolf, "Datasets: A community library for natural language processing," 2021.

[2] P. Awasthi, N. Dikkala, and P. Kamath, "Do more negative samples necessarily hurt in contrastive learning?" 2022.

[3] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," 2019.

[4] J. Zhu, Y. Zhou, V. N. Ioannidis, S. Qian, W. Ai, X. Song, and D. Koutra, "Pitfalls in link prediction with graph neural networks: Understanding the impact of target-link inclusion amp; better practices," in *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, ser. WSDM '24. ACM, Mar. 2024. [Online]. Available: http://dx.doi.org/10.1145/3616855.3635786

[5] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," 2018.

[6] T. Ucar, "Ness: Node embeddings from static subgraphs," 2023.

[7] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[8] F. Girosi, M. Jones, and T. Poggio, "Regularization Theory and Neural Networks Architectures," *Neural Computation*, vol. 7, no. 2, pp. 219–269, 03 1995. [Online]. Available: https://doi.org/10.1162/neco.1995.7.2.219

[9] J. Li, H. Shomer, H. Mao, S. Zeng, Y. Ma, N. Shah, J. Tang, and D. Yin, "Evaluating graph neural networks for link prediction: Current pitfalls and new benchmarking," 2023. [Online]. Available: https://arxiv.org/abs/2306.10453

[10] M. Zhang, "Graph neural networks: Link prediction," in *Graph Neural Networks: Foundations, Frontiers, and Applications*, L. Wu, P. Cui, J. Pei, and L. Zhao, Eds. Singapore: Springer Singapore, 2022, pp. 195–223.

[11] Barabasi and Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, oct 1999. [Online]. Available: https://doi.org/10.1126%2Fscience.286.5439.509

[12] L. A. Adamic and E. Adar, "Friends and neighbors on the web," *Social Networks*, vol. 25, no. 3, pp. 211–230, 2003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378873303000091

[13] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer Networks and ISDN Systems*, vol. 30, no. 1, pp. 107–117, 1998, proceedings of the Seventh International World Wide Web Conference. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016975529800110X

[14] T. N. Kipf and M. Welling, "Variational graph auto-encoders," 2016.

[15] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," 2019.

[16] T. R. Davidson, L. Falorsi, N. D. Cao, T. Kipf, and J. M. Tomczak, "Hyperspherical variational auto-encoders," 2022.

[17] C. Wang, S. Pan, G. Long, X. Zhu, and J. Jiang, "Mgae: Marginalized graph autoencoder for graph clustering," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, ser. CIKM '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 889–898. [Online]. Available: https://doi.org/10.1145/3132847.3132967

[18] B. P. Chamberlain, S. Shirobokov, E. Rossi, F. Frasca, T. Markovich, N. Hammerla, M. M. Bronstein, and M. Hansmire, "Graph neural networks for link prediction with subgraph sketching," 2023.

[19] Z. Zhu, Z. Zhang, L.-P. Xhonneux, and J. Tang, "Neural bellman-ford networks: A general graph neural network framework for link prediction," 2022.

[20] S. Zhou, Z. Guo, C. Aggarwal, X. Zhang, and S. Wang, "Link prediction on heterophilic graphs via disentangled representation learning," 2022.

[21] K. Kashinath, M. Mustafa, A. Albert, J. Wu, C. Jiang, S. Esmaeilzadeh, K. Azizzadenesheli, R. Wang, A. Chattopadhyay, A. Singh, A. Manepalli, D. Chirila, R. Yu, R. Walters, B. White, H. Xiao, H. Tchelepi, P. Marcus, A. Anandkumar, and M. Prabhat, "Physics-informed machine learning: Case studies for weather and climate modelling," *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, vol. 379, p. 20200093, 02 2021.

[22] C. Meng, S. Seo, D. Cao, S. Griesemer, and Y. Liu, "When physics meets machine learning: A survey of physics-informed machine learning," 2022.

[23] C. Banerjee, K. Nguyen, C. Fookes, and M. Raissi, "A survey on physics informed reinforcement learning: Review and open problems," 2023.

[24] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis, "Learning nonlinear operators via deeponet based on the universal approximation theorem of operators," *Nature Machine Intelligence*, vol. 3, no. 3, p. 218–229, Mar. 2021. [Online]. Available: http://dx.doi.org/10.1038/s42256-021-00302-5

[25] G. Karniadakis, Y. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," pp. 1–19, 05 2021.

[26] S. Greydanus, M. Dzamba, and J. Yosinski, "Hamiltonian neural networks," 2019.

[27] B. P. Chamberlain, J. Rowbottom, M. Gorinova, S. Webb, E. Rossi, and M. M. Bronstein, "Grand: Graph neural diffusion," 2021.

[28] F. Di Giovanni, J. Rowbottom, B. P. Chamberlain, T. Markovich, and M. M. Bronstein, "Understanding convolution on graphs via energies," 2023.

[29] J. Choi, S. Hong, N. Park, and S.-B. Cho, "Gread: Graph neural reaction-diffusion networks," 2023.

[30] A. Gravina, D. Bacciu, and C. Gallicchio, "Anti-symmetric DGN: a stable architecture for deep graph networks," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=J3Y7cgZOOS

[31] S. Heilig, A. Gravina, A. Trenta, C. Gallicchio, and D. Bacciu, "Injecting hamiltonian architectural bias into deep graph networks for long-range propagation," 2024. [Online]. Available: https://arxiv.org/abs/2405.17163