Supplement

Anonymous Author(s)
Affiliation
Address
email

The equation numbers and result references without prefixes correspond directly to those presented in the main paper. Throughout this supplementary material, labels are prefixed with "S" to facilitate easy cross-referencing with the corresponding sections in this document.

In this supplement, we expand on material from the main paper, addressing each point as follows:

1. An in-depth comparison with related work is covered in Section S1.
2. Section S2 presents the datasets and details of the attackers, and includes additional experimental results, inference time, and model size to reinforce our model's effectiveness.
3. Theoretical proofs supporting assertions made in the main paper are presented in Section S3.
4. A concrete example to illustrate our claims is provided in Section Section S4.
5. The complete summary of the algorithm can be found in Section Section S5.
6. Lastly, we address the limitations of our work and discuss its broader impact.

S1 Related Work

In what follows, we briefly review a few concepts closely related to our work.

Graph Adversarial Attacks and Defenses. In *modification attacks*, adversaries can perturb a graph's topology by adding or removing edges [1–9]. To improve the modification attack performance, adversaries are also permitted to perturb node attributes [4–7, 10, 11]. In *injection attacks*, adversaries can only inject malicious nodes into the original graph [12–15] while the edges and nodes inside the original graph are not allowed to be perturbed. For *defense methods* against adversarial attacks, multiple robust GNN models have been proposed. Examples include RobustGCN [16], GRAND [17], ProGNN [18], GLNN [19], GAUGM [20], STABLE [21] and RWL-GNN [22]. In addition, preprocessing-based defenders including GNN-SVD [23] and GNNGuard [24] may help to improve GNN robustness.

In this paper, we use different attacks to test GNNs robustness. We compare graph neural flows with different stability settings with the above-mentioned defense methods as robustness benchmarks.

Stable Graph Neural Flow Networks. While traditional GNNs perform message passing on a simple, discrete and flat space, graph neural flows model the message passing as a continuous diffusion process that occurs on a smooth manifold. GRAND [25] and GRAND++ [26] use heat diffusion to achieve feature information exchange. BLEND [27] exploits the Beltrami diffusion where the nodes' positional information is updated along with their features. GraphCON [28] adopts the coupled oscillator model that preserves the graph's Dirichlet energy over time and thus mitigates the oversmoothing problem. In general, [29] shows that graph neural PDEs are Lyapunov stable and exhibit stronger robustness against graph topology perturbation than traditional GNNs.

While most of the above-mentioned graph neural flows are Lyapunov stable, whether the notion of Lyapunov stability leads to better adversarial robustness is an open question. In this paper, we argue that Lyapunov stability does not necessarily imply adversarial robustness.

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

Hamiltonian Neural Networks. Hamiltonian equations have been applied to conserve an energy-like quantity in (graph) neural networks. The references [30–32] train a neural network to infer the Hamiltonian dynamics of a physical system, where Hamiltonian equations are solved using neural ODE solvers. In [33], the authors propose to learn a Hamiltonian function of the system by a neural network to capture the dynamics of physical systems from observed trajectories. They shows that the network performs well on noisy and complex systems such as a spring-chain system. To forecast dynamics, the work [34] use neural networks that incorporate Hamiltonian dynamics to efficiently learn phase space orbits and demonstrate the effectiveness of Hamiltonian neural networks on several dynamics benchmarks. The paper [35] builds a Hamiltonian-inspired neural ODE to stabilize the gradients so as to avoid gradient vanishing and gradient exploding.

In this paper, inspired by existing Hamiltonian neural networks, we introduce several energy-conservative graph neural flows. We are neither simulating a physical system nor forecasting a forecast the dynamics for a physical problem. Instead, we combine the Hamiltonian mechanics concept with graph neural networks to develop a new robust GNN.

S2 More Experiments

S2.1 Data and Attackers

The datasets and attack budgets utilized in Table 2 and Table 3 are outlined in Table S1 and Table S2, respectively. These datasets span various domains and scales, thereby providing a diverse base for our study. The adopted attack budget aligns consistently with the specifications set out in the paper [36].

Table S1: Dataset Details

Dataset	# Nodes	# Edges	# Features	# Classes
Cora	2708	5429	1433	7
Citeseer	3327	4732	3703	6
PubMed	19717	44338	500	3
Coauthor	18,333	81,894	6,805	15
Computers	13,752	245,861	767	10
Ogbn-Arxiv	169343	1166243	128	40

Table S2: Attacks' budgets for GIA. * refers to targted GIA.

Dataset	max # Nodes	max # Edges
Cora	60	20
Citeseer	90	10
PubMed	200	100
Coauthor	300	150
Ogbn-Arxiv*	120	100
Computers*	100	150

S2.2 Implementation Details

The raw node features are compressed to a fixed dimension, such as 64, using a fully connected (FC) layer to generate the initial features $q(0)$ in (9). At time $t = 0$, $p(0)$ and $q(0)$ are initialized identically. For $t > 0$, both $q(t)$ and $p(t)$ undergo updates using a graph ODE. The ODE is solved using the solver from [37]. It is observed that different solvers deliver comparable performance in terms of clean accuracy. However, to mitigate computational expense, the Euler solver is employed in our experiments, with an ablation study on different solvers provided for further insight. The integral time T acts as a hyperparameter in our model. Interestingly, the performance of the model exhibits minimal sensitivity to this time T. For all datasets, we establish the time T as 3 and maintain a fixed step size of 1. This setup aligns with a fair comparison to three-layer GNNs.
All the baseline models presented in Table 2 and Table 3 are implemented based on the original work of [36]. The baseline model results in Table 4 are directly extracted from the paper [38]. This is done as we employ the same clean and perturbed graph datasets provided in their research [38].

Our experiment code is developed based on the following repositories:

- `https://github.com/tk-rusch/GraphCON`
- `https://github.com/twitter-research/graph-neural-pde`
- `https://github.com/LFhase/GIA-HAO`
- `https://github.com/ChandlerBang/Pro-GNN`

S2.3 White-box Attack

In our main study, we utilized black-box attacks. Now, we extend our experiments to incorporate white-box, injection, and evasion attacks. In the context of white-box attacks, the adversaries have full access to the target model, enabling them to directly attack the target model to generate a perturbed graph. This represents a significantly more potent form of attack than the black-box variant. Moreover, we execute inductive learning tasks the same as Table 2, with the corresponding results reported in Table S3. We observe that, under white-box attack conditions, all other baseline models exhibit severely reduced performance, essentially collapsing across all datasets. The classification accuracy of the HANG model experiences a slight decline on the Cora, Citeseer, and Pubmed datasets. However, its performance remains substantially superior to other diffusion models or GNN models. Intriguingly, our HANG-quad model remains virtually unaffected by the white-box attacks, maintaining a performance level similar to that observed under black-box attacks. This observation underscores the robustness of HANG-quad and reiterates the critical role that the energy conservation property plays in fortifying the model against adversarial attacks.

Table S3: Node classification accuracy (%) on graph **injection, evasion, non-targeted, white-box** attack in **inductive** learning. The best and the second-best result for each criterion are highlighted in red and blue respectively.

Dataset	Attack	HANG	HANG-quad	GraphCON	GraphCurv	GRAND	GAT	GraphSAGE	GCN
Cora	*clean*	87.13±0.86	79.68±0.62	86.27±0.51	86.13±0.51	87.53±0.59	87.58±0.64	86.65±1.51	88.31±0.48
	PGD	67.69±3.84	78.04±0.91	42.09±1.74	37.16±1.69	36.02±4.09	30.95±8.22	29.79±7.56	35.83±0.71
	TDGIA	64.54±3.95	77.35±0.66	19.01±1.45	15.46±1.98	14.72±1.97	4.81±1.19	17.83±6.62	33.05±1.09
Citeseer	*clean*	74.11±0.62	71.85±0.48	74.84±0.49	69.62±0.56	74.98±0.45	67.87±4.97	63.22±9.14	72.63±1.14
	PGD	67.54±1.52	72.21±0.71	42.78±1.54	32.24±1.21	38.57±1.94	25.87±6.69	29.65±4.11	30.69±2.33
	TDGIA	63.29±3.15	70.62±0.96	33.55±1.10	16.26±1.20	30.11±1.43	17.46±3.34	17.83±1.56	21.10±2.35
CoauthorCS	*clean*	96.16±0.09	95.27±0.12	95.10±0.12	93.93±0.48	95.08±0.12	92.84±0.41	93.0±0.39	93.33±0.37
	PGD	93.40±0.71	93.25±1.02	7.80±1.18	13.21±4.21	8.0±0.06	11.96±7.10	10.73±6.84	11.02±5.04
	TDGIA	93.38±0.71	94.12±0.43	7.35±1.61	10.38±1.07	4.53±1.33	1.35±0.55	2.89±1.65	3.61±1.77
Pubmed	*clean*	89.93±0.27	88.10±0.33	88.78±0.46	86.97±0.37	88.44±0.34	87.41±1.73	88.71±0.37	88.46±0.20
	PGD	68.62±2.82	87.64±0.39	36.86±2.63	39.34±0.77	39.52±3.35	38.04±4.91	38.76±4.58	39.03±0.10
	TDGIA	69.56±3.16	87.91±0.46	31.49±1.87	30.15±1.30	36.19±7.04	24.43±4.10	38.89±0.76	42.64±1.41

S2.4 Attack Strength

We assess the robustness of the HANG model and its variant under varying attack strengths, with the node classification results displayed in Table S4. Our analysis reveals that the HANG model demonstrates superior robustness as the attack budget escalates. It's noteworthy, however, that under larger attack budgets, HANG-quad may relinquish its robustness attribute in the face of PGD and Meta injection attacks. This result implies that the amalgamation of Lyapunov and Conservative stability facilitates enhanced robustness only under minor graph perturbations. On the contrary, the HANG model, which solely incorporates Conservative stability, exhibits a consistently high-performance level regardless of the increasing attack strength.

S2.5 Nettack

We further evaluate the robustness of our model under the targeted poisoning attack, Nettack [4]. Adhering to the settings outlined in [38], we select nodes in the test set with a degree greater than 10 to be the target nodes. We then vary the number of perturbations applied to each targeted node from 1 to 5, incrementing in steps of 1. It's important to note that Nettack only involves feature perturbations. The test accuracy in Table S5 refer to the classification accuracy on the targeted nodes. As demonstrated in Table S5, our HANG-quad model exhibits exceptional resistance to Nettack, thereby underlining its superior robustness. This suggests that the combination of Lyapunov stability and energy conservative stability significantly enhances robustness in the face of graph poisoning attacks, as Lyapunov stability has been shown to offer robustness against minor feature perturbations in the input graph [29]. Furthermore, our HANG model also displays superior resilience compared to other PDE models, reinforcing the fact that the energy conservative principle contributes to its robustness.

Table S4: Node classification accuracy (%) on graph **injection, evasion, non-targeted, black-box** attack in **inductive** learning under various attack strength.

Dataset	Attack	# nods/edges injected	HANG	HANG-quad	GraphCON	GraphCurv	GRAND	GAT	GraphSAGE	GCN
Cora	PGD	80/40	75.78±3.46	78.41±0.71	33.60±2.62	33.74±2.84	34.04±1.87	32.83±3.51	26.79±6.53	31.31±0.06
	PGD	100/60	74.25±1.39	76.74±0.48	62.69±0.83	33.65±2.50	32.72±1.47	31.25±2.13	25.94±7.24	31.23±0.0
	PGD	120/80	71.01±2.60	70.69±1.83	33.46±1.95	34.88±3.66	32.21±0.83	31.28±0.15	25.57±7.28	31.23±0.0
	PGD	140/100	69.72±2.94	59.75±3.57	34.02±1.97	36.98±4.87	33.31±1.79	30.26±3.49	25.11±7.56	31.23±0.0
	PGD	160/120	68.12±2.93	45.06±4.77	33.34±2.27	39.78±4.15	33.07±1.34	31.28±0.15	24.87±7.80	31.23±0.0
	PGD	180/140	66.51±4.17	36.69±4.02	33.19±1.74	45.11±4.76	33.12±1.64	31.41±0.52	24.79±7.90	31.23±0.0
	PGD	200/160	66.01±4.10	29.58±5.11	32.99±2.40	50.51±3.94	32.18±0.89	31.47±0.70	24.74±7.96	31.23±0.0
	TDGIA	80/40	79.47±2.57	78.48±0.54	21.69±1.51	28.70±3.18	22.34±2.18	25.33±6.27	25.43±3.22	26.51±3.36
	TDGIA	100/60	78.73±1.20	78.74±0.94	24.32±4.16	36.07±2.27	27.10±2.04	27.67±5.38	30.63±0.62	30.68±0.34
	TDGIA	120/80	78.48±1.91	78.20±0.78	29.22±1.91	36.43±2.58	29.06±2.24	29.67±6.34	30.37±0.41	31.28±0.87
	TDGIA	140/100	79.40±2.20	77.42±0.71	23.54±1.49	25.54±4.26	27.04±3.25	26.99±6.05	28.85±2.75	31.94±3.26
	TDGIA	160/120	79.37±1.30	78.05±1.23	24.27±3.00	32.40±2.37	24.56±2.92	19.74±6.83	29.78±1.66	30.09±0.88
	TDGIA	180/140	78.49±2.34	76.90±0.57	31.20±1.34	41.67±6.30	31.44±0.32	30.15±4.67	31.23±0.0	31.23±0.0
	TDGIA	200/160	78.85±2.22	76.70±1.02	24.94±3.18	34.32±1.11	29.37±1.49	24.64±6.93	31.07±0.34	31.14±0.38
	MetaGIA	80/40	74.41±2.74	78.10±0.71	34.73±3.45	33.58±2.74	33.43±1.20	32.16±1.71	29.58±3.90	31.28±0.10
	MetaGIA	100/60	73.23±2.64	76.01±0.64	33.90±3.43	33.48±3.02	33.0±1.53	30.30±3.02	28.48±3.78	31.22±0.04
	MetaGIA	120/80	73.63±1.86	18.21±7.41	32.91±1.47	47.80±4.06	32.79±2.59	31.23±0.0	24.76±7.93	31.23±0.0
	MetaGIA	140/100	72.05±2.87	59.47±2.71	33.11±1.97	38.77±5.09	33.12±1.05	31.65±1.22	26.46±5.89	31.23±0.0
	MetaGIA	160/120	70.93±2.20	17.78±7.89	33.51±2.36	57.77±3.18	32.46±1.0	31.42±0.60	24.69±8.02	31.23±0.0
	MetaGIA	180/140	67.37±4.45	30.31±0.33	32.42±1.01	47.48±4.76	33.07±0.94	29.15±6.26	25.42±7.15	31.23±0.0
	MetaGIA	200/160	68.44±2.71	16.69±6.29	32.68±1.38	64.38±1.73	32.52±1.04	28.79±7.09	24.69±8.02	31.23±0.0
Citeseer	PGD	110/30	72.23±0.79	71.98±0.70	25.06±3.41	41.09±14.36	27.48±3.54	18.49±1.94	20.55±4.44	18.63±0.92
	PGD	130/50	71.61±0.84	71.44±0.72	22.90±4.73	43.05±13.60	26.42±6.16	18.33±1.75	19.16±2.95	17.85±1.66
	PGD	150/70	72.01±0.68	70.33±0.97	24.28±4.68	38.89±13.86	30.63±4.89	18.15±1.77	18.73±2.19	17.42±0.51
	PGD	170/90	71.22±0.67	67.81±1.11	23.79±4.70	26.14±4.72	21.49±2.54	19.09±2.83	17.51±0.41	17.65±0.75
	PGD	190/110	71.18±0.54	62.26±1.51	22.88±3.01	29.57±7.91	19.95±2.03	17.31±3.40	17.44±0.34	17.85±1.15
	PGD	210/130	71.13±0.72	50.15±1.88	24.90±3.30	31.26±4.51	20.65±1.23	17.33±3.40	17.39±0.33	17.67±0.90
	PGD	230/150	71.13±0.85	36.03±2.13	28.16±3.98	35.71±4.84	21.36±2.23	17.31±3.40	17.38±0.34	17.75±1.28
	TDGIA	110/30	72.48±0.67	72.30±0.82	24.30±1.73	26.31±1.64	23.85±1.25	19.26±3.59	20.24±1.97	18.80±2.45
	TDGIA	130/50	72.26±0.72	72.32±0.69	27.34±2.72	32.0±2.62	23.37±1.07	18.15±1.68	21.16±2.68	20.08±2.78
	TDGIA	150/70	72.15±0.58	72.41±0.86	26.41±1.68	27.39±1.25	22.21±1.82	19.26±3.08	19.54±2.25	20.39±2.18
	TDGIA	170/90	72.41±0.64	72.06±0.98	21.63±1.18	25.25±1.94	20.05±1.46	18.16±2.49	19.09±2.28	19.37±2.30
	TDGIA	190/110	72.80±0.85	72.39±0.65	24.56±2.34	26.97±2.04	19.94±1.44	18.04±1.59	20.13±1.72	21.53±2.92
	TDGIA	210/130	72.35±0.42	71.78±0.82	24.58±3.51	24.49±1.19	21.46±1.93	18.49±2.13	20.16±2.43	18.46±1.72
	TDGIA	230/150	71.77±0.71	71.76±0.81	19.01±2.74	28.85±1.65	19.28±1.72	18.97±2.65	20.76±4.0	18.76±2.02
	MetaGIA	110/30	72.22±1.38	72.43±0.67	22.48±2.28	20.95±2.29	30.29±4.58	21.77±2.91	22.18±2.88	18.72±0.27
	MetaGIA	130/50	72.18±1.17	71.96±0.86	23.28±4.32	19.81±1.44	25.91±3.65	20.44±2.63	20.67±2.39	18.16±0.16
	MetaGIA	150/70	71.83±0.98	71.05±0.93	24.68±4.59	19.72±1.82	26.82±2.76	19.64±2.85	20.24±3.80	18.19±0.07
	MetaGIA	170/90	71.76±1.22	68.81±0.77	22.48±2.22	20.80±1.97	29.07±8.81	18.82±2.68	19.20±2.75	18.27±0.0
	MetaGIA	190/110	71.85±0.60	64.54±0.78	23.29±4.28	22.91±2.98	26.28±2.84	18.59±2.01	18.70±2.24	18.28±0.03
	MetaGIA	210/130	71.39±0.72	56.44±2.42	22.56±3.09	24.97±2.53	26.84±3.41	18.71±2.64	18.61±2.09	18.27±0.0
	MetaGIA	230/150	71.52±0.65	13.48±3.49	24.13±3.08	48.37±2.85	26.81±3.78	18.58±1.83	18.59±2.06	18.27±0.0

Table S5: Node classification accuracy (%) under **modification, poisoning** targeted attack (Nettack) in **transductive** learning. The best and the second-best result for each criterion are highlighted in red and blue respectively.

Dataset	Ptb	HANG	HANG-quad	GraphCON	GraphCurv	GRAND	GAT	GCN	RGCN	GCN-SVD	Pro-GNN
Cora	1	75.54±3.10	76.99±3.16	73.25±3.91	63.73±2.25	80.12±1.81	76.04±2.08	75.06±1.02	76.75±1.71	77.23±1.82	81.81±1.66
	2	73.73±3.64	76.51±2.60	67.83±3.0	62.41±2.94	76.27±1.79	70.24±1.43	70.60±1.10	70.96±1.14	72.53±1.60	75.90±1.43
	3	68.43±4.23	73.13±2.85	68.19±2.10	61.20±3.08	70.48±3.74	65.54±1.34	67.95±1.72	66.51±1.60	66.75±1.54	70.12±1.93
	4	66.02±2.21	72.53±2.14	57.59±2.34	56.51±2.72	65.30±2.40	61.69±0.90	61.57±1.47	59.28±2.68	60.72±1.63	65.66±1.35
	5	60.12±3.63	68.80±2.55	55.30±4.77	51.93±2.77	57.95±2.38	58.31±2.03	55.54±1.66	55.30±1.66	57.71±1.82	64.34±1.72
Citeseer	1	76.03±3.51	79.05±1.38	76.03±3.44	68.89±2.67	80.0±1.05	81.27±1.38	78.41±1.62	78.25±0.73	80.16±2.04	81.75±0.79
	2	74.76±2.50	77.94±2.29	68.73±6.62	67.62±3.11	74.28±7.47	77.43±4.89	74.92±3.54	75.40±2.04	79.84±0.73	81.27±0.95
	3	74.76±2.18	77.14±2.48	60.47±5.24	60.63±3.87	57.14±9.28	60.85±2.99	63.97±3.69	60.31±1.19	77.14±2.86	79.68±1.98
	4	73.49±3.02	78.41±1.62	55.55±6.23	53.17±6.48	59.84±2.75	61.59±4.64	55.40±2.60	55.49±1.75	69.52±3.31	77.78±2.84
	5	72.06±3.56	73.49±3.48	51.75±2.77	48.73±4.60	48.41±8.10	55.56±6.28	47.62±5.17	47.44±2.01	69.21±2.48	71.27±4.99
Polblogs	1	97.06±0.66	97.37±0.37	87.07±1.35	68.17±3.25	96.41±0.87	97.22±0.25	96.83±0.17	97.00±0.07	97.56±0.20	96.83±0.06
	2	96.39±1.16	96.89±0.16	82.92±1.53	65.48±2.85	92.93±4.21	96.11±0.65	95.61±0.20	95.87±0.23	97.12±0.09	97.17±0.12
	3	96.02±0.93	96.65±0.15	80.76±0.74	62.59±1.99	91.96±4.22	95.81±0.56	95.41±0.18	95.59±0.27	96.61±0.14	96.93±0.12
	4	93.81±3.86	96.26±0.53	77.46±1.59	58.68±0.40	86.83±6.28	94.80±0.66	94.24±0.24	94.37±0.26	96.17±0.19	96.89±0.16
	5	91.65±4.67	95.91±0.33	75.30±2.71	59.02±3.19	83.69±5.88	93.28±1.43	93.00±0.48	93.20±0.43	95.13±0.25	96.13±0.25

S2.6 ODE solvers

The results from various ODE solvers are depicted in Table S6. We consider fixed-step Euler and RK4, along with adaptive-step Dopri5, from [37], and Symplectic-Euler from [28]. The Symplectic-Euler method, being inherently energy-conserving, is particularly suited for preserving the dynamic properties of Hamiltonian systems over long times. Our observations suggest that while the choice of solver slightly influences the clean accuracy for some models, their performance under attack conditions remains fairly consistent. Consequently, there was no specific optimization for solver selection during our experiments. For computational efficiency, we opted for the Euler ODE solver in all experiments presented in the main paper.

Table S6: Node classification accuracy (%) on graph **injection, evasion, non-targeted, black-box** attack in **inductive** learning of Citeseer dataset.

Attack	Solver	HANG	HANG-quad	GraphCON	GraphCurv	GRAND
clean	Euler	74.11±0.62	71.85±0.48	74.84±0.49	69.62±0.56	74.98±0.45
	rk4	73.71±1.58	72.63±0.56	75.80±0.38	74.46±0.68	75.32±0.78
	symplectic euler	71.35±1.91	72.80±0.64	75.43±0.62	69.87±0.78	75.70±0.71
	dopri5	75.20±0.93	–	76.24±0.76	74.63±0.70	75.14±0.56
PGD	Euler	72.31±1.16	71.07±0.41	40.56±0.36	55.67±5.35	36.68±1.05
	rk4	71.85±2.04	72.38±0.52	41.26±0.89	41.51±1.76	41.21±1.57
	symplectic euler	70.57±1.74	72.46±0.53	40.87±2.62	49.09±7.82	39.72±1.54
	dopri5	73.59±0.38	–	42.20±2.21	40.07±0.87	40.53±1.14
TDGIA	Euler	72.12±0.52	71.69±0.40	36.67±1.25	34.17±4.68	36.67±1.25
	rk4	71.03±1.64	72.85±0.78	36.19±2.03	37.90±1.70	34.21±1.63
	symplectic euler	71.79±2.03	73.31±0.58	35.32±1.78	28.40±0.91	35.12±1.62
	dopri5	72.14±0.71	–	38.04±1.71	40.63±1.60	34.39±1.19
MetaGIA	Euler	72.92±0.66	71.60±0.48	48.36±2.12	45.60±4.31	46.23±2.01
	rk4	70.25±1.45	72.39±0.61	42.57±1.09	43.38±0.88	41.01±0.92
	symplectic euler	71.56±1.07	72.86±0.72	42.57±1.09	44.72±6.28	41.0±0.64
	dopri5	71.83±1.26	–	42.61±0.57	42.93±0.61	41.74±0.69

S2.7 Computation Time

The average inference time and model size for different models used in our study are outlined in Table S7. This analysis is performed using the Cora dataset, with all graph PDE models employing the Euler Solver, an integration time of 3, and a step size of 1. Additionally, for fair comparison, all baseline models are configured with 3 layers. Upon examination, it is observed that our HANG and HANG-quad models necessitate more inference time compared to other baseline models. This is primarily due to the requirement in these models to initially calculate the derivative. However, when compared to other defense models, such as GCNGUAD, our models are still more efficient, thus validating their practical utility.

Table S7: Average inference time and model size

Model	HANG	HANG-quad	GraphCON	GraphCurv	GRAND	GAT	GraphSAGE	GCN	GCNGUARD	RGCN
Inference Time (ms)	17.16	14.41	2.60	6.18	2.25	4.13	3.85	3.93	95.90	2.50
Model Size(MB)	0.895	0.936	0.732	0.734	0.732	0.381	0.380	0.380	0.380	0.735

S3 Proof of Theorem S1

.

In the main paper, the statement of Theorem S1 contains some superfluous assumptions, which we revise herein, supplying proofs as required.

Theorem S1. *Given that $\mathbf{A}_G$ is right stochastic, GRAND demonstrates both BIBO and Lyapunov stability for any $\alpha \geq 1$. Moreover, if $\alpha > 1$, it achieves asymptotic stability under any perturbation. For $\alpha = 1$, if we take the normalized adjacent matrix $\mathbf{A}_G = \mathbf{W}\mathbf{D}^{-1}$, where $\mathbf{D}$ is the diagonal node degree matrix, GRAND conserves a quantity that can be interpreted as energy. Additionally, when $\alpha = 1$, it attains asymptotic stability if the graph is aperiodic and strongly connected, and if the perturbation on $\mathbf{X}(0)$ preserves unchanged column summations.*

Proof. Recall that

$$\frac{\mathrm{d}\mathbf{X}(t)}{\mathrm{d}t} = (\mathbf{A}_G(\mathbf{X}(t)) - \alpha\mathbf{I})\mathbf{X}(t) := \overline{\mathbf{A}}_G(\mathbf{X}(t))\mathbf{X}(t), \tag{S1}$$

where $\mathbf{A}_G$ is right stochastic and $\alpha \geq 1$.

Without loss of generality, we assume $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}|\times 1}$ since the results can be generalized to $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}|\times r}$ component-wise.

As $\mathbf{A}_G(\mathbf{X}(t))$ is consistently right-stochastic and $\alpha \geq 1$, the eigenvalues of $\overline{\mathbf{A}}_G(\mathbf{X}(t)) = \mathbf{A}_G(\mathbf{X}(t)) - \alpha\mathbf{I}$ always have *non-positive real parts* [39][Theorem 8.1.22]. We define a Lyapunov function as $V(\mathbf{X}(t)) = \|\mathbf{X}(t)\|^2 \coloneqq \mathbf{X}(t)^\intercal\mathbf{X}(t)$ where $\|\cdot\|$ is the euclidean norm.

Take the derivative of V with respect to time, we have

$$\dot{V}(\mathbf{X}(t)) = \mathbf{X}^\intercal(t)\dot{\mathbf{X}}(t) + \dot{\mathbf{X}}^\intercal(t)\mathbf{X}(t) = \mathbf{X}^\intercal(t)\left(\overline{\mathbf{A}}_G(\mathbf{X}(t)) + \overline{\mathbf{A}}_G^\intercal(\mathbf{X}(t))\right)\mathbf{X}(t) \tag{S2}$$

Since the eigenvalues of $\overline{\mathbf{A}}_G(\mathbf{X}(t))$ possess negative real parts, we infer that $\dot{V}(\mathbf{X}(t)) \leq 0$. This shows that $V(\mathbf{X}(t))$ is non-increasing over time, thereby implying that $\mathbf{X}(t)$ remains bounded. In effect, we also prove Lyapunov stability concerning the equilibrium point $\mathbf{X} = 0$.

To prove asymptotic stability for the equilibrium point $\mathbf{X} = 0$ when $\alpha > 1$, we need to show that the system not only remains bounded but also approaches $\mathbf{X} = 0$ as $t \to \infty$. This is indicated by the fact that the eigenvalues of $\overline{\mathbf{A}}_G(\mathbf{X}(t))$ have negative real parts when $\alpha > 1$, which means $\dot{V}(\mathbf{X}(t)) < 0$ unless $\mathbf{X}(t) = 0$. This signifies that $V(\mathbf{X}(t))$ strictly declines over time unless $\mathbf{X}(t) = 0$, thereby showing that the system will converge to $\mathbf{X} = 0$ as $t \to \infty$.

We next show that if $\mathbf{A}_G$ is chosen as $\mathbf{W}\mathbf{D}^{-1}$, GRAND conserves a quantity that can be regarded as energy. The "energy" is the sum of the elements of $\mathbf{X}(t)$. This quantity is conserved if $\mathbf{1}^\intercal\overline{\mathbf{A}}_G = \mathbf{0}^\intercal$, where $\mathbf{1}$ is an all-ones vector, which is evident given that $\mathbf{W}\mathbf{D}^{-1}$ is column stochastic.

Finally, we aim to prove that GRAND is asymptotic stable concerning a specified equilibrium vector when $\alpha = 1$ and the graph is aperiodic and strongly connected.

Consider the matrix $\mathbf{W}\mathbf{D}^{-1}$; given that $\mathbf{W}\mathbf{D}^{-1}$ is column stochastic and the graph is strongly connected and aperiodic, the Perron-Frobenius theorem as stated in [39][Lemma 8.4.3., Theorem 8.4.4] confirms that the value 1 is the unique eigenvalue equal to the spectral radius. This infers that the modified adjacency matrix $\overline{\mathbf{A}}_G = \mathbf{W}\mathbf{D}^{-1} - \mathbf{I}$ has an eigenvalue of 0, with the rest of the eigenvalues having *strictly negative real parts*. Representing $\overline{\mathbf{A}}_G = \mathbf{S}\mathbf{J}\mathbf{S}^{-1}$ as the Jordan canonical form, we conclude that $\mathbf{J}$ contains a block of only a single 0 (for simplicity, we assume 0 is the first Jordan block).

As our system of equations is a linear time-invariant ordinary differential equation (ODE), the solution to (S1) can be expressed as:

$$\begin{aligned}\mathbf{X}(t) &= e^{\overline{\mathbf{A}}_G t}\mathbf{X}(0)\\ &= \mathbf{S}e^{\mathbf{J}t}\mathbf{S}^{-1}\mathbf{X}(0)\end{aligned} \tag{S3}$$

Further, the Jordan canonical form of $\mathbf{W}\mathbf{D}^{-1}$ is represented as $\mathbf{S}\bar{\mathbf{J}}\mathbf{S}^{-1}$ where $\bar{\mathbf{J}} = \mathbf{J} + \mathbf{I}$ with the first Jordan block being 1 and the rest having eigenvalues *strictly less than* 1. Based on [39][Theorem 3.2.5.2.], we observe that $\lim_{k\to\infty}(\mathbf{W}\mathbf{D}^{-1})^k = \lim_{k\to\infty}\mathbf{S}\bar{\mathbf{J}}^k\mathbf{S}^{-1} = \mathbf{S}\mathbf{\Lambda}\mathbf{S}^{-1}$, where $\mathbf{\Lambda}$ is a diagonal matrix with the first element as 1 and all the others as 0:

$$\mathbf{\Lambda} = \begin{pmatrix} 1 & & & \\ & 0 & & \\ & & \ddots & \\ & & & 0 \end{pmatrix}$$

Since $\lim_{k\to\infty}(\mathbf{W}\mathbf{D}^{-1})^k$ maintains its column stochasticity and the rank of $\mathbf{S}\mathbf{\Lambda}\mathbf{S}^{-1}$ is 1, we deduce that the first row of $\mathbf{S}^{-1}$ is $a\mathbf{1}^\intercal$ with a being a scalar and $\mathbf{1}$ an all-ones vector. According to [39][3.2.2], it follows that

$$\lim_{t\to 0}\mathbf{X}(t) = \lim_{t\to 0}\mathbf{S}e^{\mathbf{J}t}\mathbf{S}^{-1}\mathbf{X}(0) = \mathbf{S}\mathbf{\Lambda}\mathbf{S}^{-1}\mathbf{X}(0) \tag{S4}$$

If $s \coloneqq \mathbf{1}^\intercal \mathbf{X}(0)$ remains constant, we have that $\lim_{t\to 0} \mathbf{X}(t) = sa\mathbf{s}$ where $\mathbf{s}$ is the first column of $\mathbf{S}$. We thus conclude that if the perturbation on $\mathbf{X}(0)$ maintains the column summations, i.e. the "energy", unchanged, the asymptotic convergence to the equilibrium vector $sa\mathbf{s}$ remains unaffected.

The proof is now complete. □

S4 Example 1

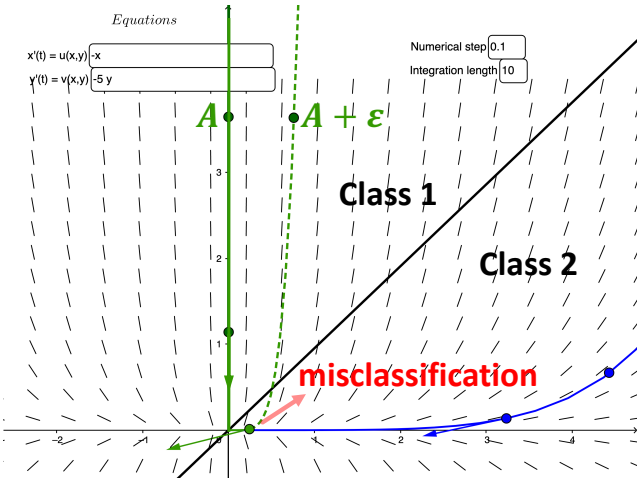

Figure S1: Lyapunov stability and adversarial robustness

We provide here an example to demonstrate our claim: consider the Lyapunov stable ODE

$$\dot{\mathbf{x}}(t) = \begin{pmatrix} -1 & 0 \\ 0 & -5 \end{pmatrix} \mathbf{x}(t) \tag{S5}$$

with initial condition $\mathbf{x}(0) = [x_1(0), x_2(0)]^\intercal$. The solution to this ODE is given by $\mathbf{x}(t) = x_1(0)e^{-t}[1,0]^\intercal + x_2(0)e^{-5t}[0,1]^\intercal$. For all initial points in $\mathbb{R}^2$, we have $\mathbf{x}(t) \to \mathbf{0}$ as $t \to \infty$. Furthermore, as $t \to \infty$, the trajectory $\mathbf{x}(t)$ for any initial point is approximately parallel to the x-axis. We draw the phase plane in Fig. S1.

Assume that the points on the upper half y-axis belongs to class 1 while we have a linear classifier that seperates class 1 and class 2 as shown in Fig. S1. We observe that for the initial point A belonging to class 1, the solution from a small perturbed initial point $A + \epsilon$ is misclassified as class 2 for a large enough t for any linear classifier. We see from this example that Lyapunov stability does not imply adversarial robustness in graph neural diffusion models.

S5 Complete Algorithm Summary

We present the complete algorithm of HANG in Algorithm 1, which unfortunately had been delayed in its inclusion within the main paper due to space constraints.

Algorithm 1: Graph Node Embedding Learning with HANG

Initialization: Initialize the network modules including Hamiltonian function network H_{net}, the raw node features compressor network FC, and the final classifier.

I. Training:

for *Epoch* 1 ***to*** N **do**

1) Perform the following to obtain the embedding $\mathbf{q}_k(T)$ for each node k:

Input: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with raw node features

Apply FC to compress raw features for each node and get $\{(\mathbf{q}_k(0), \mathbf{p}_k(0))\}_{k=1}^{|\mathcal{V}|}$. Here, we divide the $2r$ dimensions into two equal segments. The first half functions as the feature vector $\mathbf{q}_k(0) = (q_k^1, \ldots, q_k^r)$, while the second half acts as "momentum" vectors $\mathbf{p}_k(0) = (p_k^1, \ldots, p_k^r)$ that guide system evolution. For simplification, in code implementation, we set the output feature dimension of the compressor FC as r and designate $\mathbf{p}_k(0) = \mathbf{q}_k(0)$.

2) To better express the evolution dynamics mathematically, concatenate and relabel the node features as:

$$q(0) = \left(q^1(0), \ldots q^{r|\mathcal{V}|}(0)\right) = \left(\mathbf{q}_1(0), \ldots, \mathbf{q}_{|\mathcal{V}|}(0)\right).$$
$$p(0) = \left(p_1(0), \ldots p_{r|\mathcal{V}|}(0)\right) = \left(\mathbf{p}_1(0), \ldots, \mathbf{p}_{|\mathcal{V}|}(0)\right), \tag{S6}$$

Note that in the actual code implementation, the concatenation of node features is not necessary as H_{net} is realized as either (11) or (12). The concatenate operation is only for mathematical formulation.

3) The trajectory of feature evolution is modelled as per the following canonical Hamilton's equations:

$$\dot{q}(t) = \frac{\partial H_{\text{net}}}{\partial p}, \quad \dot{p}(t) = -\frac{\partial H_{\text{net}}}{\partial q}, \tag{S7}$$

with the initial features $(q(0), p(0)) \in \mathbb{R}^{2r|\mathcal{V}|}$.

Various ODE solvers as provided by [37] and the symplectic-euler solver from [28] can be employed to solve (S7). We refer readers to Section S2.6 for more details.

4) Acquire the evolved features at time T as $q(T)$, which is then decompressed into individual node features $\mathbf{q}_k(T)_{k=1}^{|\mathcal{V}|}$ for further utilization.

5) Utilize backpropagation to minimize the cross-entropy loss for node classification.

6) Perform validation over the validation split.

7) Save the model parameters.

II. Testing:

Load the model from the best validation epoch and perform **Step I.1-4).** to obtain the final feature embedding over the test split. Perform node classification.

Limitations

While our work on graph neural flows presents promising advancements in enhancing adversarial robustness of GNNs using Hamiltonian-inspired neural ODEs, it is not without limitations. As we demonstrated in the paper, the notions of stability borrowed from dynamical systems, such as BIBO stability and Lyapunov stability, do not always guarantee adversarial robustness. Our finding that energy-conservative Hamiltonian graph flows improve robustness is only one facet of the broader landscape of potential stability measures. It is possible that other notions of stability, not covered in this work, could yield additional insights into adversarial robustness. Our current Hamiltonian graph neural flows do not explicitly account for quasi-periodic motions in the graph dynamics. The Kolmogorov-Arnold-Moser (KAM) theory, a foundational theory in Hamiltonian dynamics, is renowned for its analysis of persistence of quasi-periodic motions under small perturbations in Hamiltonian dynamical systems. While the energy-conserving nature of our Hamiltonian-inspired model inherently offers some level of robustness to perturbations, an explicit incorporation of KAM theory could potentially further improve the robustness, particularly in the face of quasi-periodic

adversarial attacks. However, this is a complex task due to the high dimensionality of typical graph datasets and the intricacies involved in approximating quasi-periodic dynamics.

Broader Impact

This research, centered on enhancing adversarial robustness in graph neural networks (GNNs), carries implications for various sectors, such as social media networks, sensor networks, and chemistry. By improving the resilience of GNNs, we can boost the reliability of AI-driven systems, contributing to greater efficiency, productivity, and cost-effectiveness. The shift towards automation may displace certain jobs, raising ethical concerns about income disparity and job security. Moreover, while our models enhance robustness, potential system failures can still occur, with impacts varying based on the application. Lastly, the robustness conferred might be exploited maliciously. Our work underscores the importance of diligent oversight, equitable technology implementation, and continuous innovation in the development of AI technologies.

References

[S-1] J. Chen, Y. Wu, X. Xu, Y. Chen, H. Zheng, and Q. Xuan, "Fast gradient attack on network embedding," *ArXiv*, 2018.

[S-2] M. Waniek, T. P. Michalak, M. J. Wooldridge, and T. Rahwan, "Hiding individuals and communities in a social network," *Nature Human Behaviour*, vol. 2, no. 1, pp. 139–147, 2018.

[S-3] J. Du, S. Zhang, G. Wu, J. M. F. Moura, and S. Kar, "Topology adaptive graph convolutional networks," *ArXiv*, vol. abs/1710.10370, 2017.

[S-4] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial attacks on neural networks for graph data," in *Proc. Int. Conf. Knowl. Discovery Data Mining*, 2018.

[S-5] D. Zügner and S. Günnemann, "Adversarial attacks on graph neural networks via meta learning," in *Proc. Int. Conf. Learn. Representations*, 2019.

[S-6] Y. Ma, S. Wang, T. Derr, L. Wu, and J. Tang, "Graph adversarial attack via rewiring," in *Proc. Int. Conf. Knowl. Discovery Data Mining*, 2021, p. 1161–1169.

[S-7] Y. Sun, S. Wang, X. Tang, T.-Y. Hsieh, and V. Honavar, "Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach," in *Proc. Web Conf.*, 2020, p. 673–683.

[S-8] X. Wan, H. Kenlay, B. Ru, A. Blaas, M. A. Osborne, and X. Dong, "Adversarial attacks on graph classification via bayesian optimisation," *arXiv preprint arXiv:2111.02842*, 2021.

[S-9] S. Geisler, T. Schmidt, H. Şirin, D. Zügner, A. Bojchevski, and S. Günnemann, "Robustness of graph neural networks at scale," *Advances Neural Inf. Process. Syst.*, vol. 34, pp. 7637–7649, 2021.

[S-10] J. Ma, J. Deng, and Q. Mei, "Adversarial attack on graph neural networks as an influence maximization problem," in *ACM International Conference on Web Search and Data Mining*, 2022, pp. 675–685.

[S-11] B. Finkelshtein, C. Baskin, E. Zheltonozhskii, and U. Alon, "Single-node attacks for fooling graph neural networks," *Neurocomputing*, vol. 513, pp. 1–12, 2022.

[S-12] J. Wang, M. Luo, F. Suya, J. Li, Z. Yang, and Q. Zheng, "Scalable attack on graph data by injecting vicious nodes," *Data Mining Knowl. Discovery*, pp. 1 – 27, 2020.

[S-13] Q. Zheng, Y. Fei, Y. Li, Q. Liu, M. Hu, and Q. Sun. Kdd cup 2020 ml track 2 adversarial attacks and defense on academic graph 1st place solution. Accessed: May 1, 2022. [Online]. Available: https://github.com/Stanislas0/KDD_CUP_2020_MLTrack2_SPEIT

[S-14] X. Zou, Q. Zheng, Y. Dong, X. Guan, E. Kharlamov, J. Lu, and J. Tang, "Tdgia: Effective injection attacks on graph neural networks," in *Proc. Int. Conf. Knowl. Discovery Data Mining*, 2021, p. 2461–2471.

[S-15] H. Hussain, M. Cao, S. Sikdar, D. Helic, E. Lex, M. Strohmaier, and R. Kern, "Adversarial inter-group link injection degrades the fairness of graph neural networks," *arXiv preprint arXiv:2209.05957*, 2022.

[S-16] D. Zhu, Z. Zhang, P. Cui, and W. Zhu, "Robust graph convolutional networks against adversarial attacks," in *Proc. Int. Conf. Knowl. Discovery Data Mining*, 2019, p. 1399–1407.

[S-17] W. Feng, J. Zhang, Y. Dong, Y. Han, H. Luan, Q. Xu, Q. Yang, E. Kharlamov, and J. Tang, "Graph random neural networks for semi-supervised learning on graphs," in *Proc. Advances Neural Inf. Process. Syst.*, 2020.

[S-18] W. Jin, Y. Ma, X. Liu, X. Tang, S. Wang, and J. Tang, "Graph structure learning for robust graph neural networks," in *Proc. Int. Conf. Knowl. Discovery Data Mining*, 2020, p. 66–74.

[S-19] X. Gao, W. Hu, and Z. Guo, "Exploring structure-adaptive graph learning for robust semi-supervised classification," in *2020 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2020, pp. 1–6.

[S-20] T. Zhao, Y. Liu, L. Neves, O. Woodford, M. Jiang, and N. Shah, "Data augmentation for graph neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 11 015–11 023.

[S-21] K. Li, Y. Liu, X. Ao, J. Chi, J. Feng, H. Yang, and Q. He, "Reliable representations make a stronger defender: Unsupervised structure refinement for robust gnn," in *Proc. Int. Conf. Knowl. Discovery Data Mining*, 2022, pp. 925–935.

[S-22] B. Runwal, S. Kumar *et al.*, "Robust graph neural networks using weighted graph laplacian," *arXiv preprint arXiv:2208.01853*, 2022.

[S-23] N. Entezari, S. A. Al-Sayouri, A. Darvishzadeh, and E. E. Papalexakis, "All you need is low (rank): Defending against adversarial attacks on graphs," in *Proc. Int. Conf. Web Search Data Mining*, 2020, p. 169–177.

[S-24] X. Zhang and M. Zitnik, "Gnnguard: Defending graph neural networks against adversarial attacks," in *Proc. Advances Neural Inf. Process. Syst.*, 2020.

[S-25] B. P. Chamberlain, J. Rowbottom, M. Goronova, S. Webb, E. Rossi, and M. M. Bronstein, "Grand: Graph neural diffusion," in *Proc. Int. Conf. Mach. Learn.*, 2021.

[S-26] M. Thorpe, T. M. Nguyen, H. Xia, T. Strohmer, A. Bertozzi, S. Osher, and B. Wang, "Grand++: Graph neural diffusion with a source term," in *Proc. Int. Conf. Learn. Representations*, 2021.

[S-27] B. P. Chamberlain, J. Rowbottom, D. Eynard, F. Di Giovanni, D. Xiaowen, and M. M. Bronstein, "Beltrami flow and neural diffusion on graphs," in *Advances Neural Inf. Process. Syst.*, 2021.

[S-28] T. K. Rusch, B. P. Chamberlain, J. Rowbottom, S. Mishra, and M. M. Bronstein, "Graph-coupled oscillator networks," in *Proc. Int. Conf. Mach. Learn.*, 2022.

[S-29] Y. Song, Q. Kang, S. Wang, K. Zhao, and W. P. Tay, "On the robustness of graph neural diffusion to topology perturbations," in *Advances Neural Inf. Process. Syst.*, New Orleans, USA, Nov. 2022.

[S-30] S. Greydanus, M. Dzamba, and J. Yosinski, "Hamiltonian neural networks," in *Advances Neural Inf. Process. Syst.*, 2019.

[S-31] Y. D. Zhong, B. Dey, and A. Chakraborty, "Symplectic ode-net: Learning hamiltonian dynamics with control," in *Proc. Int. Conf. Learn. Representations*, 2020.

[S-32] Y. Chen, T. Matsubara, and T. Yaguchi, "Neural symplectic form: Learning hamiltonian equations on general coordinate systems," in *Advances Neural Inf. Process. Syst.*, 2021.

[S-33] Z. Chen, J. Zhang, M. Arjovsky, and L. Bottou, "Symplectic recurrent neural networks," in *Proc. Int. Conf. Learn. Representations*, 2020.

[S-34] A. Choudhary, J. F. Lindner, E. G. Holliday, S. T. Miller, S. Sinha, and W. L. Ditto, "Physics-enhanced neural networks learn order and chaos," *Physical Review E*, vol. 101, no. 6, p. 062207, 2020.

[S-35] E. Haber and L. Ruthotto, "Stable architectures for deep neural networks," *Inverse Problems*, vol. 34, no. 1, pp. 1–23, Dec. 2017.

[S-36] Y. Chen, H. Yang, Y. Zhang, K. Ma, T. Liu, B. Han, and J. Cheng, "Understanding and improving graph injection attack by promoting unnoticeability," in *Proc. Int. Conf. Learn. Representations*, 2022.

[S-37] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," in *Advances Neural Inf. Process. Syst.*, 2018.

[S-38] W. Jin, Y. Ma, X. Liu, X. Tang, S. Wang, and J. Tang, "Graph structure learning for robust graph neural networks," in *Proc. Int. Conf. Knowl. Discovery Data Mining*, 2020, pp. 66–74.

[S-39] R. A. Horn and C. R. Johnson, *Matrix analysis*. New York: Cambridge university press, 2012.