Supplementary Information for Coarse-graining network flow through statistical physics and machine learning

Zhang Zhang[1,2,3†], Arsham Ghavasieh[4†], Jiang Zhang[1,2], and Manlio De Domenico[3,5,6*]

[1]School of Systems Science, Beijing Normal University, Beijing, China
[2]Swarma Research, Beijing, China
[3]Department of Physics & Astronomy "Galileo Galilei", University of Padua, Italy
[4]Center for Complex Networks and Systems Research, Luddy School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN, USA
[5]Padua Center for Network Medicine, University of Padua, Italy
[6]Istituto Nazionale di Fisica Nucleare, Sez. Padova, Italy

† Equally contributing authors

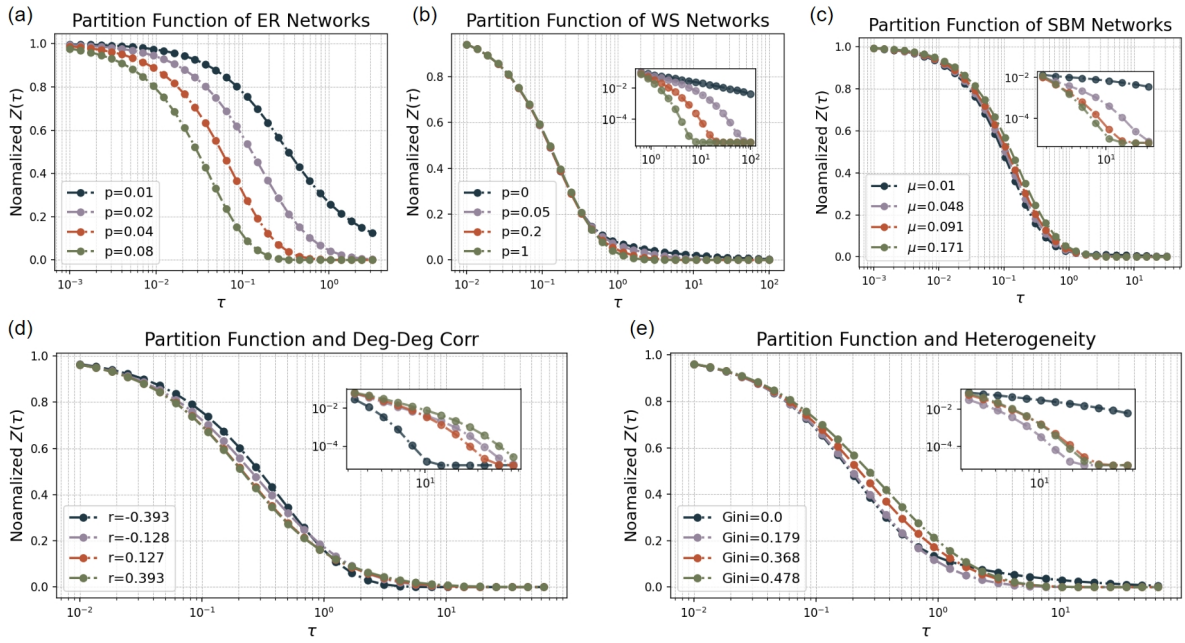
Corresponding author: manlio.dedomenico@unipd.it

1 Supplementary Note 1: Impact of network structure properties on partition function

Here, we focus on the effects of network density, clustering spectrum, community structure, degree distribution, degree-degree correlation, and homogeneity/heterogeneity on the network structure. Specifically, we observe the shape of the partition function by altering one topological property of the network while trying to keep other properties unchanged, to gain insights into the behavior of information flow on top of the network.

In subplot *a* of Supplementary Figure 1 we generated networks of different densities by changing the connection probability p in ER networks. We directly observed that density has a significant impact on the partition function: an increase in density causes the partition function to drop more quickly because higher density facilitates information diffusion across the network, making diffusion easier to complete. In subplot *b*, we observe the impact of the clustering spectrum on the partition function by changing the rewiring probability in small-world networks. When the rewiring probability of the small-world network is very low, each node is connected to a few neighbors, resulting in a high average clustering coefficient. Increasing the rewiring probability breaks these triangles, reducing the network's average clustering coefficient. With the total number of connections remaining unchanged, this affects the diffusion dynamics as follows: each node can initially diffuse information to its locality with the same difficulty, but at larger time scales, as the clustering coefficient decreases and long-range connections increase, global information diffusion becomes easier. Therefore, reducing the clustering coefficient makes large-scale information diffusion easier to complete. This is reflected in the partition function, meaning it approaches zero faster at large τ.

In subplot *c* of Supplementary Figure 1, we generated networks using the SBM model and altered the network's mixing parameter μ to observe the effect of inter-community connections on information diffusion. In the SBM model, $\mu = \frac{k_{out}}{k_{in}+k_{out}}$, in which k_{out} denotes the number of connections between communities and k_{in} denotes the number of connections inside the communities. By removing intra-community edges and adding inter-community edges, while keeping the network density constant, we notice that as the mixing parameter increases, the initial decline of the partition function slows down. This is due to the reduction of intra-community edges, which slows down the network's local (within-community) diffusion speed. Conversely, the latter part of the partition function declines faster because the increase in inter-community edges accelerates the global diffusion speed.



Supplementary Figure 1: Network topology and information flow: In this supplementary figure, we show how the network's partition function responds to changes in structural properties. In the subplot a, we set different network densities by adjusting the probability of edge creation for an Erdos Renyi graph with 400 nodes. In subplot b, we observe the effect of the small-world phenomenon on the partition function by adjusting the probability of rewiring each edge in a WS graph (with 300 nodes, each having 6 neighbors). In subplot c, we examine the impact of links between community structures on the partition function by setting the mixing parameter μ for a stochastic block network (with 3 communities, each containing 50 nodes). Here, $\mu = \frac{k_{out}}{k_{in}+k_{out}}$, where k_{in} denotes the number of connections within a community, and k_{out} denotes the number of connections between communities. In subplot d, we observe the impact of the network's degree-degree correlation on the partition function, which can be measured by the assortativity coefficient r. With constant network density, we increase the network's assortativity by adding edges between nodes of similar degrees and removing edges between nodes of differing degrees. In subplot e, we gradually rewire the edges of a regular network with 100 nodes, each with 4 neighbors, using the principle of preferential attachment to transition it into a scale-free network. During this process, homogeneity decreases while heterogeneity increases, and we use the Gini coefficient of the degree distribution to measure the level of heterogeneity.

In subplot d of Supplementary Figure 1, we measured the impact of degree-degree correlation on the network's partition function, which can be quantified by the assortativity coefficient r. When the assortativity coefficient is high, the initial decline of the partition function is faster. This is because the distribution of node degrees is relatively uniform, with each node having a sufficient number of neighbors, facilitating local information diffusion. When the assortativity coefficient is low, the decline in the latter part of the partition function is faster. This is because a few high-degree nodes connect to many low-degree nodes, acting as bridges and increasing the speed of global information diffusion.

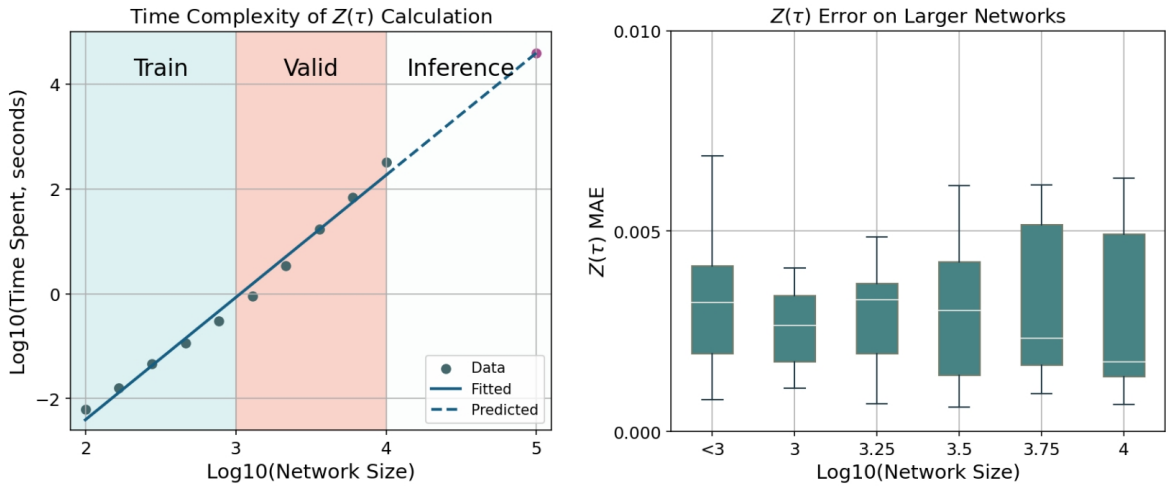
Finally, in subplot e of Supplementary Figure 1, we studied the impact of homogeneity/heterogeneity on the partition function. In a homogeneous network, different nodes have similar local structures(i.e. regular networks). In a heterogeneous network, the distribution of node degrees is broad, with certain nodes (referred to as central or hub nodes) having many more connections than others, leading to a diversity in the network's structure and connection patterns. The BA network is a typical example of a heterogeneous network. The homogeneity/heterogeneity of a network can be measured by various indicators, and here we chose the Gini coefficient of degree distribution to measure the heterogeneity. In economics, the Gini coefficient is used to measure income inequality; here, we use it to measure the unevenness of the network's degree distribution. The Gini coefficient ranges from 0 to 1, where 0 represents complete equality (total homogeneity) and 1 represents maximum inequality (total heterogeneity). In this experiment, we started with a regular network and rewired it with preferential attachment mechanism. After enough rewirings, this homogeneous regular network transformed into a heterogeneous scale-free network. We observe that in completely homogeneous networks, the partition function decreases more quickly at smaller τ but slows down at larger τ. Conversely, heterogeneous networks show the opposite pattern, with a slower decline at smaller τ and a faster decrease at larger τ. This difference can be understood from the dynamics of information diffusion. when τ is small, representing local information spread, homogeneous networks, where each node has the same number of neighbors, facilitate easy local information diffusion. In contrast, in heterogeneous networks, many nodes have only a few neighbors, making local information spread more challenging. When τ become larger, the network come to the global diffusion phase, the heterogeneity of networks, due to the presence of hub nodes, accelerates global information communication, whereas homogeneous networks lack such channels for global communication, hence slowing down at this stage.

2 Supplementary Note 2: Compression for network with 100,000 nodes

In this section, we will first introduce the challenges we might face when compressing large networks, then we will analyze the time complexity of different calculation processes, and include an experiment on compressing a network with 100,000 nodes.

When we train our model to compress a network with N nodes, it needs to compress the original network and calculate the error of the normalized partition function before and after compression as the loss to be optimized. Here, the most time-consuming part is calculating the partition function of the original network to serve as supervisory information, which has a time complexity of $O(N^3)$. Therefore, calculating the partition function for extremely large networks is almost impossible. To address this, we train the compression model on smaller network datasets, validate the model's generalizability on larger network datasets, and apply the model to extremely large networks (100,000 nodes). The time required to calculate the partition function for networks of different sizes and the sizes of the network data used for training and testing our model are shown in Supplementary Figure 2.

In the left subplot of Supplementary Figure 2, we show the time required to calculate the partition function for networks of various sizes. By performing linear fitting in log-log scale and extrapolating, we estimate that the time required to calculate the partition function for a network with 100K nodes exceeds 10^4 seconds, making repeated experiments at this scale nearly impossible, not to mention for larger networks. Therefore, we adopted the strategy of training on smaller networks and applying the trained model to larger networks. In the left subplot of Supplementary Figure 2, we can see that we train our model on networks of sizes 10^2 to 10^3. To assess the model's generalization ability, we validate the model's performance on networks sized 10^3 to 10^4. If the validation error is within a reasonable range, we can consider the model capable of compressing larger networks, and thus, apply it to the compression of larger large networks in the inference phase. The networks we used here are



Supplementary Figure 2: Time complexity of $Z(\tau)$ calculation and Generalization Results on Larger Network: In the left subplot, we present the relationship between network size in logarithmic space and the time consumed for calculating the partition function, and based on this relationship, we divide the model into three phases: training, validation, and inference. In the right subplot, we show the error variation during validation on models trained on networks smaller than 10^3 in size, and tested across networks of various sizes. We selected network sizes for testing that are similar to the training set size ($< 10^3$), and also chose different sizes of networks uniformly in the logarithmic space between 10^3 and 10^4 for testing. The model's testing error does not show a significant growth relationship with the size of the network.

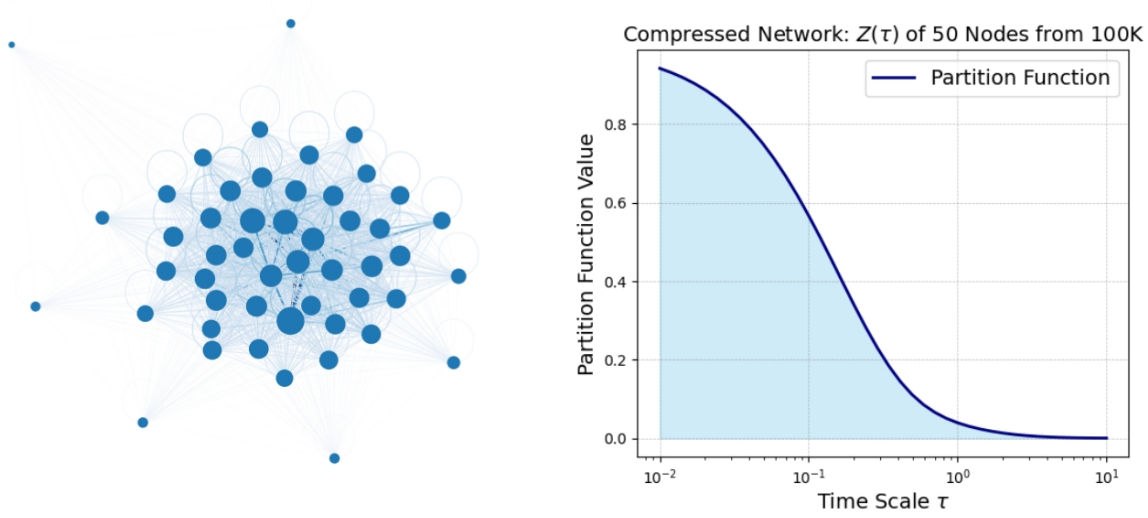
Watts-Strogatz (WS) networks with different sizes, their number of neighbors ranges from 2 to 8, and rewiring probabilities ranges from 0 to 0.6.

In the right subplot of Supplementary Figure 2, we showcase the training of our model on networks smaller than 10^3 in size, and we tested this model on both a test set of networks of the same size and larger networks (from 10^3 to 10^4) to compare the Mean Absolute Error (MAE) of the network's normalized partition function before and after compression. We observed that compared to smaller network data, the error did not significantly increase when our model was applied to larger networks. This indicates that the model has strong ability for generalization. Therefore, we further applied the model to the compression of a network with 100K nodes. Since the original network has 100K nodes and cannot be visualized, we can only visualize the network after compression. In Supplementary Figure 3, we display the structure of compressed network with 50 nodes and its partition function. Notably, while the computation of the original network's partition function would have taken more than 10^4 seconds, compressing the network first and then calculating the partition function within an acceptable range of partition function error took only a few seconds.

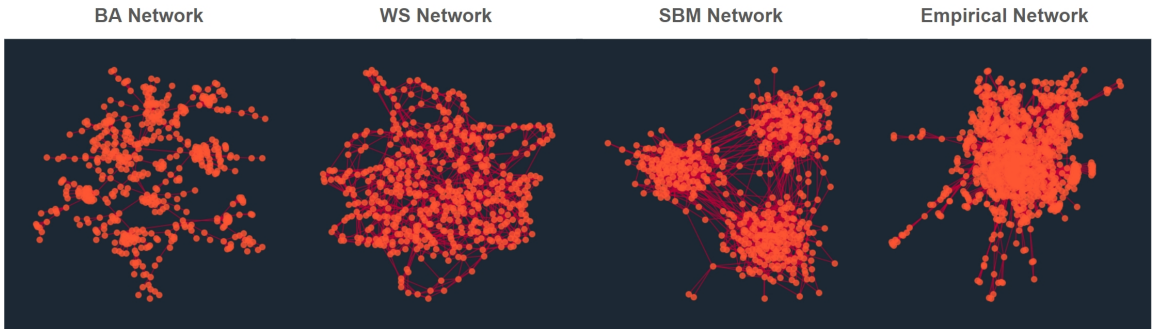
3 Supplementary Note 3: Detailed Comparision with Laplacian Renormalization Group Method

Here, we will demonstrate the differences between our method and the Laplacian Renormalization Group(LRG) method in terms of applicability, the mechanism of forming supernodes, the properties of the macroscopic network structure after compression and the ability to preserve the network entropy.

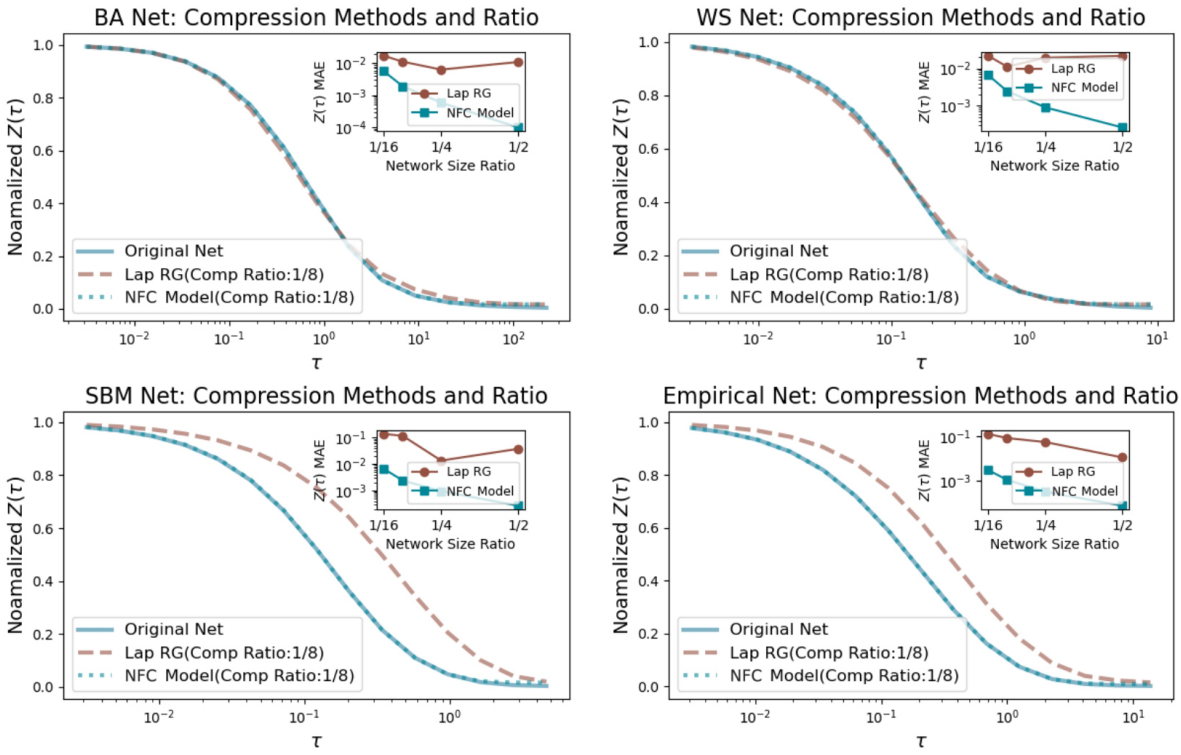
In Supplementary Figure 5, we show the partition functions of the compressed networks obtained at different compression ratios when applying our method and the LRG method to these 4 networks, respectively. We also present a subplot showing the relationship between the differences in the partition functions before and after compression changing with compression size. We chose to compress the networks to 50%, 25%, 12.5%, and 6.25% of the original network size. From Supplementary Figure 5, we can observe that among all the network data, the LRG method performs best on BA networks,



Supplementary Figure 3: Compressed WS Network: From 100K Nodes to 50 Nodes: Visualization of the structure of the compressed network(left subplot) and Partition Function of the compressed network(right subplot).



Supplementary Figure 4: A schematic diagram showing three typical synthetic networks and one empirical network: They are a BA network(N=500, m=1), a WS network(N=500, k=3, p=0.1), a SBM network(N=500, mixing parameter $\mu = \frac{k_{in}}{k_{in}+k_{out}} = 0.067$, in which k_{in} denotes the number of edges inside the communities and k_{out} denotes the number of edges between communities, and an empirical network data from the field of biology.



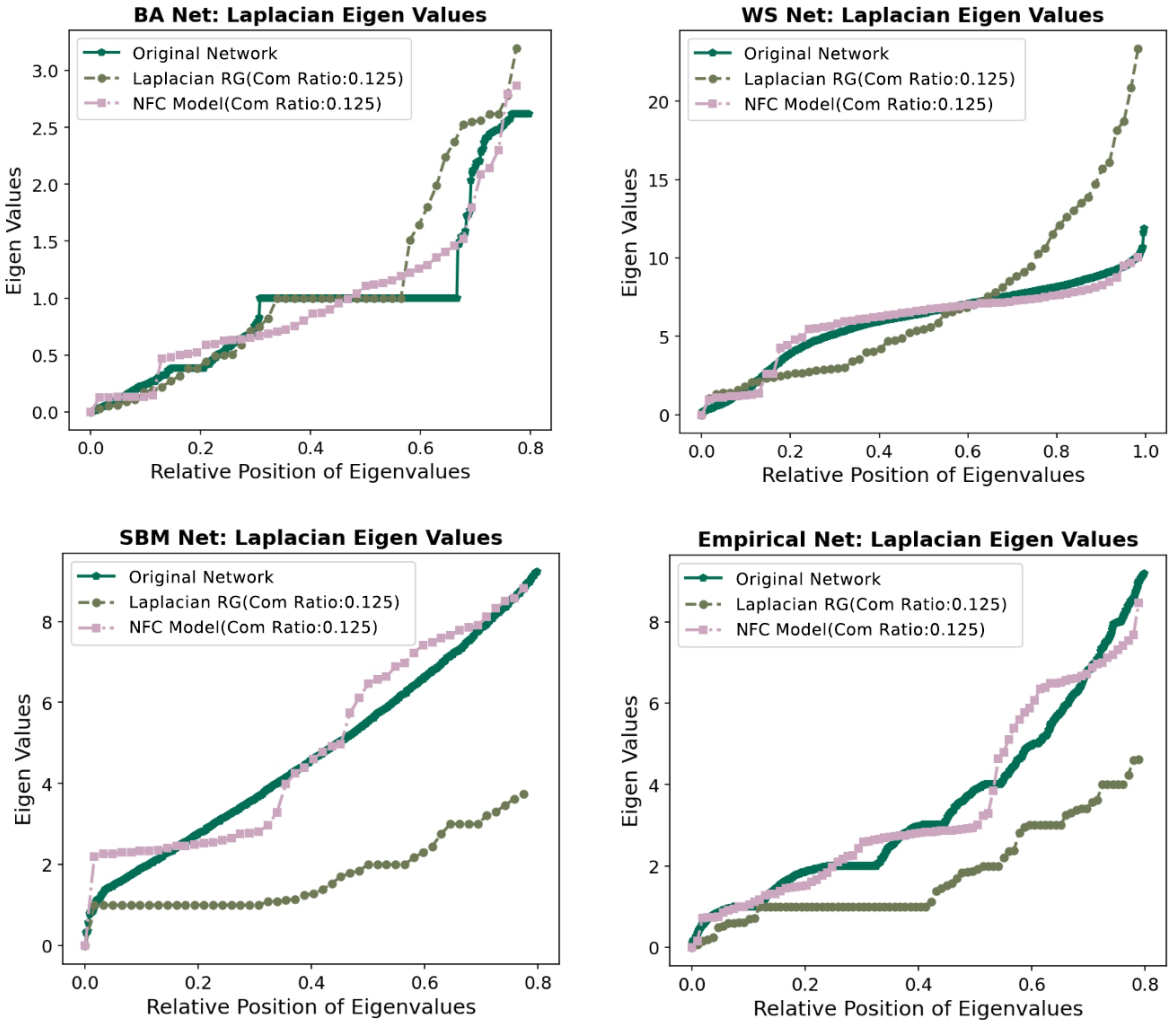
Supplementary Figure 5: Comparison for NFC Model and the LRG method across different networks: On each subplot, we plotted the partition function curve of the original network and the curves when the network is compressed to 12.5% of its original size by two methods. In the insets of each subplot, we show the mean absolute error of the normalized partition function of the original network after being compressed to 50%, 25%, 12.5%, and 6.25% by different methods

where the compressed network retains the partition function to the greatest extent, consistent with the effects claimed in their paper. However, when we move to other network structures, the performance of the LRG method starts to decline, with a larger change in the partition functions of the networks before and after compression. In contrast, our method ensures that the compressed network has a partition function similar to that of the original network in any type of tested network, even including BA network.

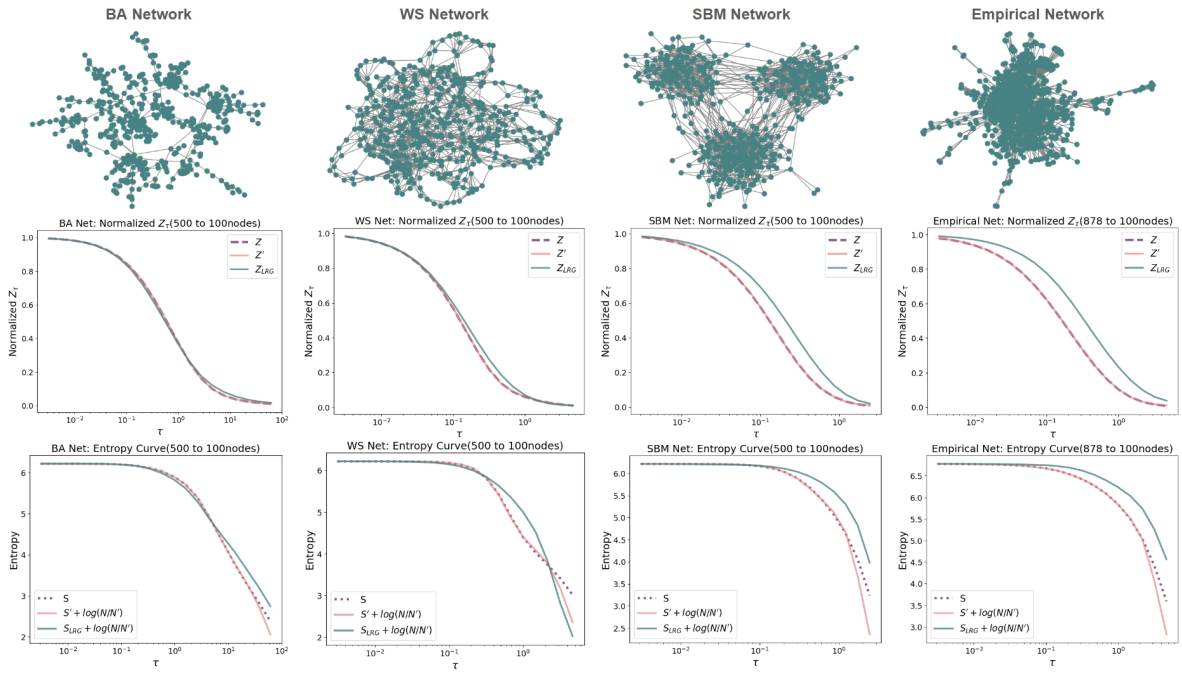
In Supplementary Figure 6, we display the distribution of eigenvalues of the Laplacian matrix for the original network and the networks compressed by the NFC Model and LRG methods. Since the network's partition function is defiened by $Z(\tau) = Tr(e^{-\tau L}) = \sum_{l=1}^{N} e^{-\tau\lambda_l}$, where λ_l denotes the Lth smallest eigen value of the laplacian matrix, all the eigenvalues are non-negative, $\lambda_l \geq 0$, and at least one of the eigenvalues is $\lambda_1 = 0$, the smaller eigenvalues will determine the network's diffusion behavior. For some networks, to highlight the differences in distributions, we only show the first 80% of the sorted eigenvalues. The figure shows that our method can more accurately maintain the consistency of the eigenvalue distribution while compressing the network size, thereby ensuring the consistency of information flow behavior.

In the maintext, we have shown analytically that our model can preserve the entropy while compressing the network size, in Supplementary Figure 7 we numerically demonstrate that our model preserve entropy better than LRG does.

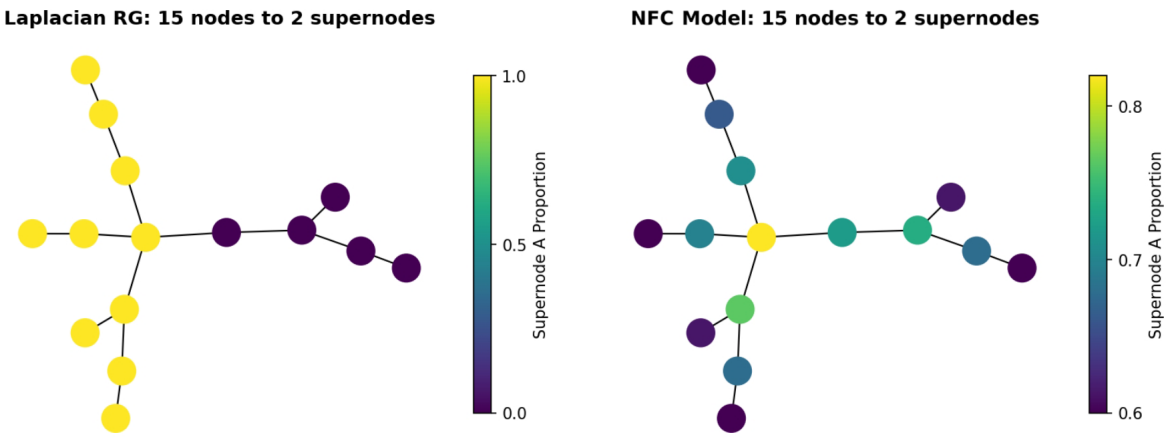
Additionally, LRG and NFC Model fundamentally differ in their basic mechanisms, that is, principles for selecting supernodes. LRG selects a subnetwork of the original network to form a supernode based on the results of the diffusion process, and this strategy is discrete, meaning each node can only be part of one supernode. In contrast, NFC Model selects nodes with similar local structures to form a supernode, and this strategy is continuous; for example, node i can enter supernode A with an 80% proportion and supernode B with a 20% proportion. As shown in Supplementary Figure 8,



Supplementary Figure 6: Distribution of Laplacian eigenvalues: We present the distribution of the Laplacian matrix's eigenvalues of networks, sorted from smallest to largest, on three synthetic networks and one empirical network. This includes the original network, networks compressed by the Laplacian RG method, and networks compressed by the NFC Model. Since the partition function is primarily determined by the smaller eigenvalues, to highlight the difference in distributions, we only display the smallest 80% of the eigenvalues for some networks.



Supplementary Figure 7: Effectiveness of preserving Entropy after network compression: In this figure we demonstrate the preservation of the partition function and entropy after compressing 3 synthetic networks and a empirical network using our method and LRG. Here, Z, Z', and Z_{LRG} denote the partition functions of the original network, our compressed network, and the LRG-compressed network, respectively. Similarly, S, S', and S_{LRG} represent the entropies of the original network, our compressed network, and the LRG-compressed network, respectively.



Supplementary Figure 8: Differences in supernode composition between methods Laplacian RG and NFC Model: In this figure, we take a BA network with 15 nodes and compress it to two nodes using methods Laplacian RG and NFC Model: supernode A and supernode B. In the left subplot, Laplacian RG groups connected subnetworks into Supernodes. In the right subplot, each node belongs to Supernode A at a certain proportion.

we chose a typical BA network with 15 nodes and $m = 1$, and used LRG and NFC Model to compress it, compressing it into a macroscopic network with two supernodes (A and B). The left subplot shows LRG's grouping strategy for nodes, where supernodes correspond to two subnetworks formed by adjacent nodes. The right subplot shows NFC Model's strategy, where nodes with similar local structures are given similar grouping results that are continuous, thus finely controlling the structure of the macroscopic network.

Moreover, an intuitive display of the structure of the compressed network can also reflect the differences between the NFC Model and the Laplacian RG method, as shown in Supplementary Figure 9: The Laplacian RG method compresses the original network into a smaller unweighted graph without self-loops. In contrast, the NFC Model produces a weighted graph that may include self-loops.

4 Supplementary Note 4: U.S. Airports grouping results

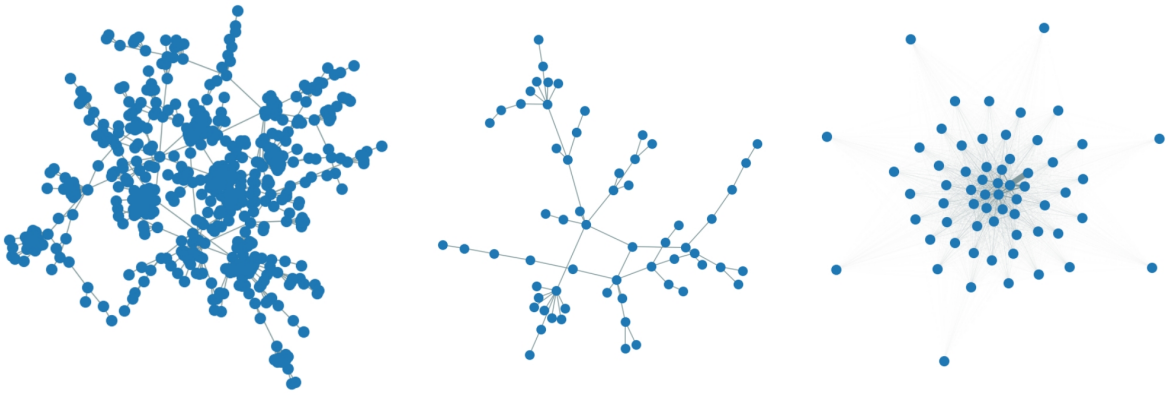
As mentioned in the main text, we have compressed the US Airport Network into two networks with 50 and 20 nodes, respectively. Each airport is assigned to a specific super-node. Here, we present the relationship between all airports and their corresponding super-nodes for interested readers, as shown in Supplementary Table 1. It's noteworthy that, despite with sufficient training, each airport nearly always has a clear group assignment; however, this does not imply that every super-node consists of distinct airport nodes. For instance, in a toy network with only three nodes, if airports A, B, and C have probabilities of 99%, 98%, and 99% respectively to be assigned to super-node 1, and probabilities of 1%, 2%, and 1% to be assigned to super-node 2, we will indicate that all nodes belong to super-node 1. Consequently, you may observe that the number of super-nodes displayed in Supplementary Table 1 is less than 50.

Supplementary Table 1: Relation between U.S. Airports and Supernodes:Using NFC Model, we have compressed the U.S. Airport Network into a macro network comprising 50 nodes. Every node in the original network corresponds to a super-node within this macro network. This Supplementary Table displays the relationship between the original nodes and the super-nodes. The left column lists the super-node identifiers, while the right column presents the corresponding airport IATA codes.

Supernode No	IATA
Supernode #1	PDX
Supernode #2	OAK, CLE, MIA, STL, SAN

Continued on next page

Supernode No	IATA
Supernode #3	MCI, CVG, RDU, AUS, MSY
Supernode #4	IAH, DTW, MSP, CLT, LAS
Supernode #5	ILI, PTU, PQI, MSS, JBR, CNM, ENA, HOM, TBN, AUG, OGS, VDZ, APF, BHB, RKD, PKB, GCN, AHN, CGI, IRK, RUT, SLK, SLN, UIN, WRL, ADK, GLH, PVC, SDP, MWA, PDT, UST, MBL, KUK, KWT, LAM, OGD
Supernode #6	SBY, CRP, SMX, BKG, HHH
Supernode #7	BTI, LUR, PIZ, ITO, BTT, FYU, TLJ, SNP, STG, HNM, DLG, HRO, HON, JNU, ACK, ADQ, MUE, HRL, BRW, BET, HVR, CDV, OME, SCC, TEB, IAG, IPL, OTZ, AOO, ELD, MKL, MKK, BFI, GAL, PBG, AKN, JHM, KTN, FAI, LNY, CDB, SIT, ISP, DUT, YNG, PUW, LWS, FKL, PSG, JST, MVY, HYA, MGW, ACV, BFD, BKW, CEC, CLM, DUJ, EWB, GGW, HOT, LBE, LEB, LMT, OTH, GST, SGY, HCR, HNS, KLG, MCG, MOU, ANI, VAK, WRG, KQA, ORH, AKP, ANV, ATK, GAM, HPB, KAL, KSM, KVL, MYU, RBY, SHH, SVA, WTK, ARC, DRG, IGG, KVC, PTH, TOG, YAK, AET, CYF, STS, TVF, PGD, JHW, SHD, SDY, GDV, OLF, SOW, FRD, ESD, EMK, UNK, SHX, CHU, NUI, EEK, KWK, MLL, RSH, CIK, HUS, HSL, NUL, VEE, WBQ, CEM, SHG, AGN, ELV, HNH, MTM, HYG, EGX, KPV, PIP, WSN, AKK, KYK, KLN, ABL, BKC, IAN, OBU, ORV, WLK, KTS, ELI, GLV, TLA, WAA, WMO, KKA, SMK, SKK, TNC, AKB, IKO, AUK, KPN, KFP, NLG, KLW, KWN, KOT, KYU, SCM, KKH, NIB, AIN, IRC, SLQ, MLY, PVU, MMU, NME, OOK, WNA, PKA
Supernode #8	TTN, LIH, BUR
Supernode #9	FSM, ICT, MSN, GGG, GNV, LFT, GCK, MOT, HLN, LCH, PUB, MDT, LNK, LAN, ELP, CAE, PNS, CHA, JAN, DLH, SHV, BPT, DAY, PHF, TUS, HIB, MAF, GRB, AGS, ISN, LIT, MLB, MQT, TYS, SPS, LRD, TLH, ACT, SJT, DRO, CLL, ABI, COU, DSM, EWN, MLU, ROW, BRO, DHN, FMN, HOB, SUX, MCN, TXK, GRK, MOB, SAF, CYS, SCK, COS, MFE, LBB, ART, AMA, FOE, ILM, BTR, TYR, AEX, TUL, CPR, VPS, FLO, GTF, ELM, DAB, AVL, GSO, FSD, CHO, ROA, LEX, EVV, BZN, TVC, JAC, BMI, GPT, AZO, TOL, FWA, DEC, CID, LSE, CWA, PIA, ATW, RST, CMI, MHK, GJT, SGU, SRQ, MLI, BIS, RAP, FNT, IDA, CAK, HSV, MGM, TRI, PAH, PGA, FCA, MBS, EGE, CSG, LAW, STC, GTR, CRW, AVP, BJI, FAR, GCC, SCE, MEI, SPI, CEZ, HDN, LBL, COD, SGF, JLN, ABE, XNA, SBN, OAJ, DBQ, ABR, ABY, ALO, APN, BFF, BQK, BRL, CIU, CMX, DDC, EAU, FAY, GRI, LAR, LBF, LYH, MKG, MSL, PIB, PIR, PLN, RKS, SHR, TUP, VCT, VLD, ESC, MTJ, RIW, LWB, PGV, ASE, GUC, DIK, CDR, AIA, MCK, ALS
Supernode #10	BGR, YUM, SWF, LUK, EYW, ITH, RFD, BGM, ERI
Supernode #11	GFK, ECP
Supernode #12	OGG, BIL, HTS
Supernode #13	BNA
Supernode #14	PHX, LAX, DAL, DCA, SLC, HOU, SEA, IAD, MDW, PHL, ANC
Supernode #15	DFW, ATL, DEN, ORD, SFB
Supernode #16	PIE, AZA
Supernode #17	INL, LCK, BLV, ATY, PSM, RHI, HGR, IMT, BRD
Supernode #18	BOS, EWR, KOA, FLL, LGB, TPA, BLI, JFK, BWI
Supernode #19	PRC, IPT, CDC, HVN, BTM, CLD, TWF, EKO, LNS, PIH, CNY, VEL, MMH, VIS, MCE
Supernode #20	OMA, GEG, MEM, BOI, MYR, ACY, BFL, PIT, IND, HPN, RIC, ORF, SAV, SAT, ROC, PVD, GRR, FAT, BTV, JAX, MKE, PBI, HNL, ONT, SYR, SJC, CMH, RSW, CHS, RNO, BHM, SMF, BUF, BDL, PSP, ILG, PWM, OKC, ALB, SNA, MRY, SBA, MHT, SDF, ABQ, GSP, EUG, MFR, RDM, MSO, PSC
Supernode #21	CIC, MOD, RDD, FLG, ALW, CKB, OWB, SBP, YKM, EAT
Supernode #22	SFO, LGA, MCO



Supplementary Figure 9: Visualization of network structures before and after compression: In the left subplot, we display a BA network with N = 500 and m = 1. In the middle subplot, we show the network structure obtained using the Laplacian RG method, while the network structure obtained with the NFC Model is presented in the right subplot.

5 Supplementary Note 5: Network Compression for more empirical networks from different domains

In Figure 5 of the main manuscript we conduct analysis on 25 real networks and attempt to understand the differences in compression outcomes across different domains. Here we demonstrate the basic information of the network data we use in in Supplementary Table 2.

6 Supplementary Note 6: Training one Compression model for Multiple Networks

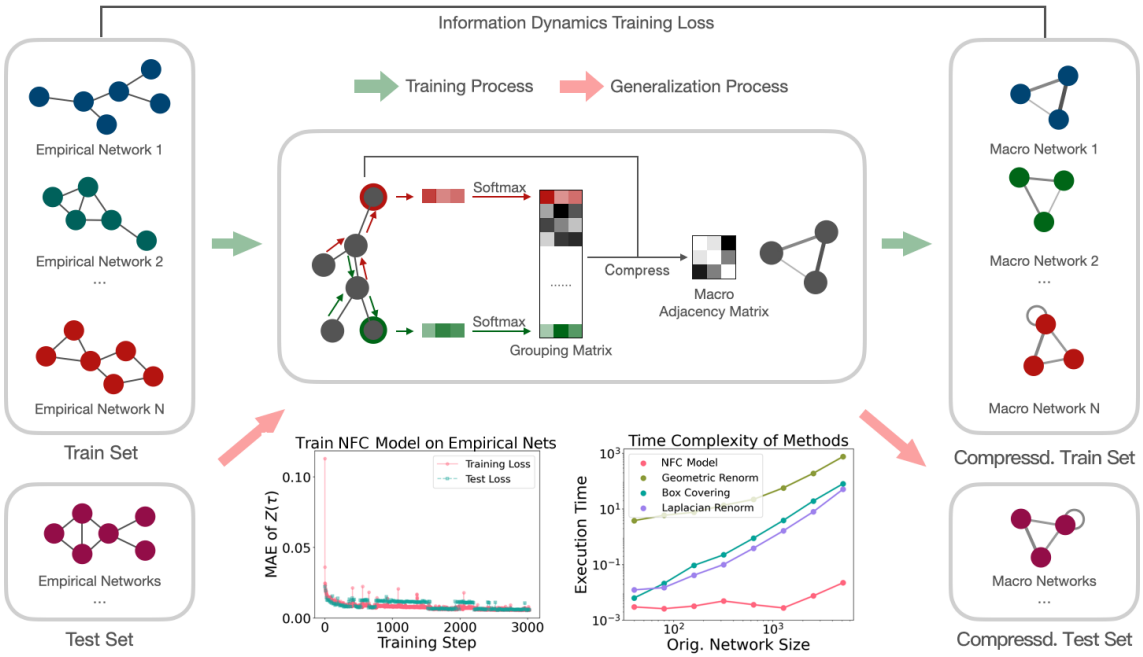
To swiftly compress networks we've not previously encountered, we trained a compression model that can adapt to multiple real-world networks. For this purpose, we downloaded 2,193 networks with node counts ranging from 100 to 2,495 from NetworkRepository[1]. These networks span multiple domains, including biology, social networks, and the brain. We selected 412 networks with more than 1,000 nodes as the test set, 1,000 networks with node counts between 100 and 1,000 for the training set, and an additional 375 as the validation set. (For more detailed public data and code, please see our GitHub repository.) The training process is depicted in Supplementary Figure 10. During each training instance, we compress a network, compare the discrepancy between normalized partition function of the compressed and the original network, and reduce this error using the gradient descent algorithm. By training across all networks in the training set, the model achieved a general compression capability for networks in our dataset. Supplementary Figure 10 displays the training pipeline and the error evolution curves for both the training and test sets.

7 Supplementary Note 7: Detailed Explanation of Time Complexity

We use Graph Isomorphism Networks (GIN) for network compression. To run a trained model we first need to calculate node features based on the topological structure, all node features used here are local characteristics of the nodes, with the most time-consuming feature being k-coreness. The computational time complexity for k-coreness is $O(N + E)$, where N and E are number of nodes and edges of the original network, respectively. The model then calculate the node grouping information by aggregating neighbor information and updating node information, which can be represented by Equation 1.

Supplementary Table 2: Information about the networks: Basic information for 25 typical networks from 8 different domains. All networks were downloaded from the Network Repository(https://networkrepository.com/).

Network Domain	Network	Nodes	Edges
Biology	yeast-protein-inter	1458	1948
	grid-plant	1272	2726
	grid-mouse	791	1098
	CE-GT	878	3181
	diseasome	516	1188
Economic	poli	2243	2667
	mahindas	1258	7513
Brain	mouse-kasthuri_graph_v4	987	1536
	macaque-rhesus_brain_1	242	3054
Collaboration	netscience	379	913
	CSphd	1024	1042
Email	univ	1133	5451
	dnc-corecipient	849	10384
Power	662-bus	662	906
	685-bus	685	1282
	bcspwr09	1723	2394
	inf-power	4941	6594
Road	euroroad	1039	1305
	minnesota	2640	3302
Social	pages-politician	5908	41706
	advogato	5054	39374
	pages-tvshow	3892	17239
	hamsterster	2000	16097
	wiki-Vote	889	2914
	pages-food	620	2091



Supplementary Figure 10

Training one Compression Model for Multiple Networks:: In this supplementary figure, we depict the training and testing processes of a model that possesses the capability for universal compression across various networks. The outcomes of the training and the time complexity are also showcased. By observing the error evolution curves for the training and test sets over time, we can ascertain the effectiveness of the training. By comparing the time required to compress networks of different sizes using various algorithms, we can confirm that our algorithm's time complexity is notably lower than others.

$$h_v^{(k)} = MLP^{(k)}((1+\epsilon^{(k)}) \cdot h_v^{(k-1)} + \sum_{u \in \mathcal{N}(v)} h_u^{(k-1)}), \tag{1}$$

in which $h_v^{(k)}$ denotes the vector representation of node v in layer k, $\mathcal{N}(v)$ denotes a set of node v's neighbors, $MLP^{(k)}$ means a multi-layer perceptron and $\epsilon^{(k)}$ denotes a learnable parameter.

Subsequently, the updated node information needs to be mapped to grouping information through a softmax operation, as represented by Equation 2.

$$S_v^i = \frac{e^{h_v^i}}{\sum_{j=1}^{M} e^{h_v^j}}, \tag{2}$$

in which S_v^i denotes the ratio that node v belongs to the super-node i and M denotes the number of a super-nodes, which is a pre-defined constant here.

Finally, the model uses the obtained grouping matrix to compress the original network and generate the adjacency matrix of the compressed network, as shown in Equation 3

$$\hat{A} = \frac{M}{N} S^T \cdot A \cdot S, \tag{3}$$

in which A and $\hat{A}$ denotes the micro and macro adjacency matrix, respectively.

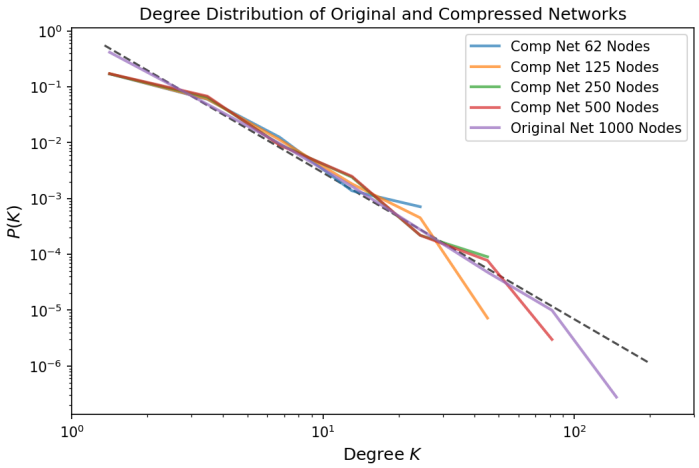
In the aggregation function of GIN represented by Equation 1, it involves two operations: aggregation of neighbor information and feature update. For the aggregation of neighbor information, it requires summing up the features of all neighbors for each node. The time complexity of this part is proportional to the number of edges E, as each edge contributes to one addition operation of neighbor features, that is, $O(E)$. For the feature update part, we use an MLP (Multi-Layer Perceptron) for updates, with the time complexity of each layer being $O(N * d_{in} * d_{out})$, where N is the number of nodes, d_{in} is the input dimension to the MLP, and d_{out} is its output dimension. Therefore, the total complexity of the GIN model is $O(E + N * d_{in} * dout)$. Since d_{in} and d_{out} are artificially defined as relatively small constants, the time complexity can generally be approximated as $O(N + E)$. This shows that the GIN model is highly efficient when processing large-scale graph data.

In the softmax operation represented by Equation 2, for each node, we need to perform M exponential operations, one summation operation of M elements, and M multiplication operations. Therefore, its time complexity is $O(N * 3M) = O(N)$.

In the process shown in Equation 3, which involves using a grouping matrix to compress the original adjacency matrix, there are two matrix multiplication operations. Since the original adjacency matrix is often represented sparsely, the time complexity of the first matrix multiplication is $O(M * E)$, and the time complexity of the second matrix multiplication is $O(M^2 * N)$. Given that M is an artificially set constant representing the macro network size, the final time complexity is $O(N + E)$. Therefore, considering these three computational steps together, compressing a network with a trained model ultimately requires time proportional to $O(N + E)$.

8 Supplementary Note 8: Preservation of Degree Distribution in BA network Compression

Although we have proven that the model can preserve information flow, we cannot guarantee the preservation of all structural features. While surprisingly, experiments have shown that we can maintain the scale-free nature of the degree distribution after compressing BA networks(see Supplementary Figure 11). We believe this is because scale-free networks have strong heterogeneity, and adjacent nodes generally do not cluster together, thus the compressed network almost does not form self-loops, avoiding the impact of self-loops. On the other hand, for the BA network, the degree distribution is a core structural attribute that determines its information propagation behavior, so our model preserves the original network's degree distribution during the compression process.



Supplementary Figure 11: Degree distribution of the original and compressed networks: This plot displays the degree distribution for a BA network with 1000 nodes (m=1) compressed to various sizes (from 1/2 to 1/16 of the original network). X-axis: Node degree, Y-axis: Frequency of degrees. As can be seen, the degree distribution remains the same before and after compression.

9 Supplementary Note 9: Implementation of alternative compression methods

In this paper, we primarily compare three methods: box-covering, geometric renormalization group, and laplacian renormalization group. Their implementations are as follows:

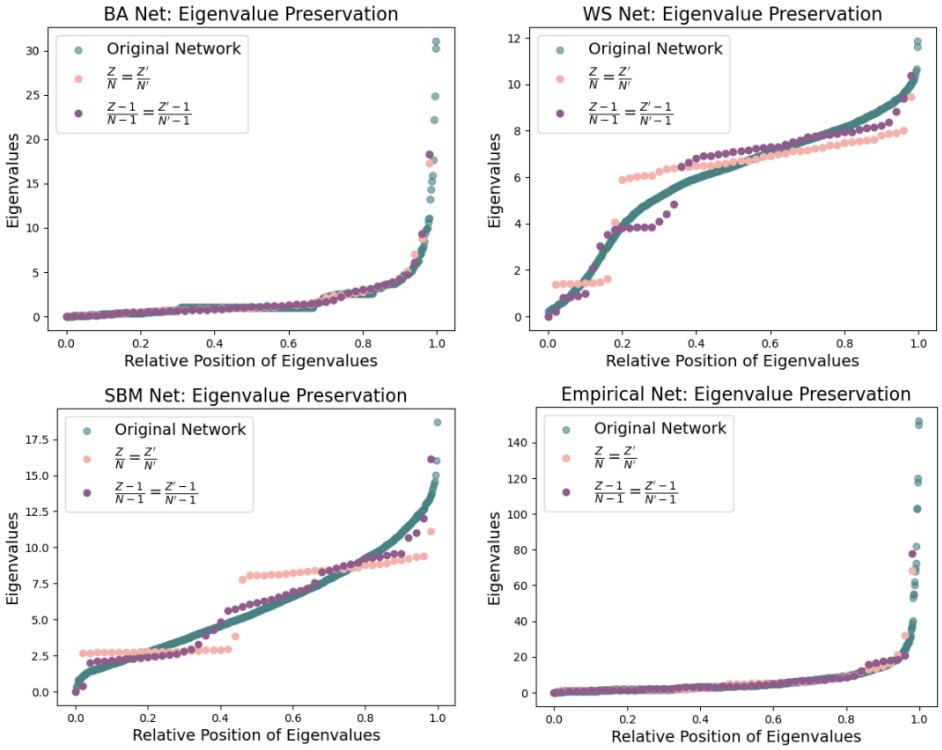
- **Box Covering:** Here we employ one of the most typical methods from the set provided by Kovács et al.[2]: the Greedy coloring method[3] to perform all comparative experiments. Their open-source code is available at https://github.com/PeterTKovacs/boxes.
- **Geometric Renormalization Group:** The core of the Geometric Renormalization Group lies in embedding the network structure into hyperbolic space. This part can be implemented using Mercator (https://github.com/networkgeometry/mercator). The remaining parts will be straightforward to implement.
- **Laplacian Renormalization Group:** Here, we use the official public code provided by the original authors, available at https://github.com/pvgongora/Laplacian-RG.

10 Supplementary Note 10: Alternative forms of Coarse-Graining

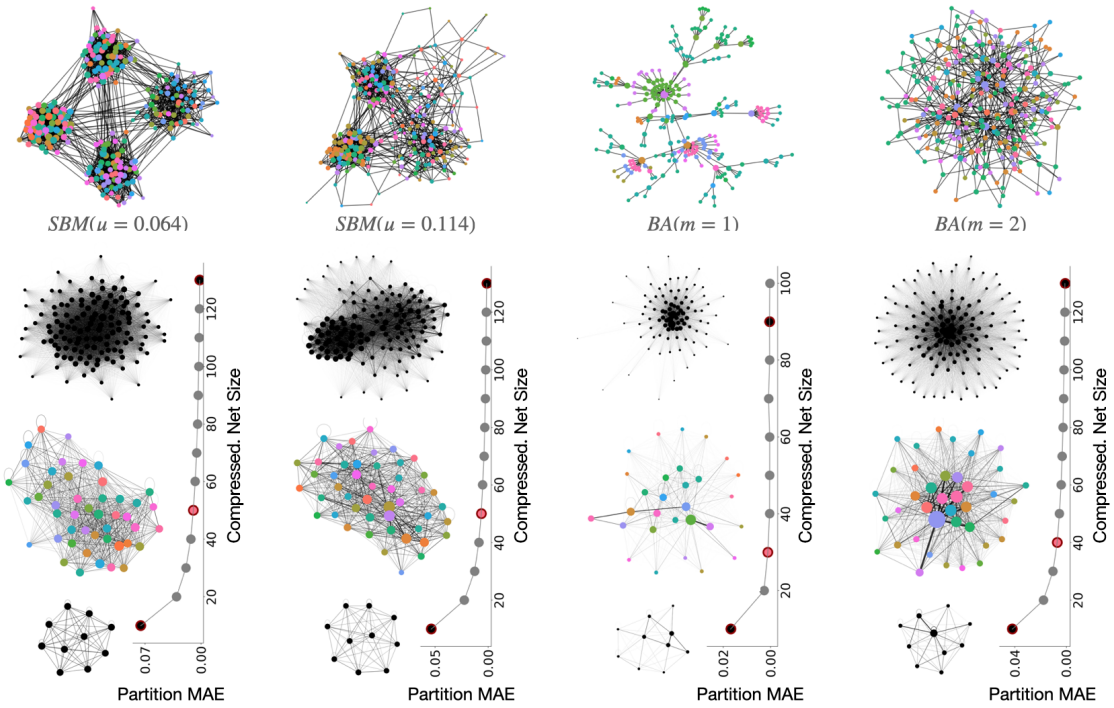
Our model's overall goal is to compress network size while ensuring the normalized partition function remains unchanged, with different normalization methods leading to different specific target functions. Assuming the original network has N nodes with a partition function Z, and the compressed network has N' nodes with a partition function Z', the two target functions are $\frac{Z}{N} = \frac{Z'}{N'}$ and $\frac{Z-1}{N-1} = \frac{Z'-1}{N'-1}$. Under all other equal conditions, the former target allows us to derive a relationship for the change in entropy as $S' = S - log(N/N')$, where S' denotes the entropy of compressed graph, meaning the shape of the entropy curve remains unchanged after compression. The latter ensures that the normalized results of the partition function range from 0 to 1, better preserving the original network's eigenvalue distribution (see Supplementary Figure 12).

11 Supplementary Note 11: Compress Synthetic Networks

As introduced in the main text and Supplementary Note 10, by setting two different target functions for the model, we can optimize for distinct compression strategies. Here, we present the results of using



Supplementary Figure 12: The ability of two target functions to preserve the eigenvalue distribution: In this figure, we show the eigenvalue distribution of networks compressed under two different target functions compared to the original networks. We conducted experiments on three synthetic networks with 500 nodes each and one real-world network with 878 nodes, compressing all networks to 50 nodes



Supplementary Figure 13: Compression for Synthetic Networks with alternative target function Visual representation of the compression process applied to two representative Stochastic Block Model (SBM) networks (mixing parameters μ of 0.064 and 0.114, respectively) and Barabasi-Albert (BA) networks (numbers of links per node m of 1 and 2). Each column delineates a particular network alongside its compressions. For every network, we select three distinct compression sizes to visualize: the error threshold(a stage at which the normalized partition function difference escalates by further compression), a point considerably below it, and another notably above it. At the error threshold and in the original network, we indicate the supernodes and their units with the same color. The deviation of the normalized partition function versus the macro network size is illustrated.

target function $\frac{Z'-1}{N'-1} = \frac{Z-1}{N-1}$ on four typical synthetic networks, as shown in Supplementary Figure 13.

As expected, for very large compressions, we find an increase in the deviation between partition functions— i.e., cost. However, interestingly, the increase is non-linear. All smaller compressions exhibit negligible errors in recovering the flow in the original networks, whereas, upon reaching a error threshold, further compressions lead to significantly higher costs.

We visualize three compressions (See Supplementary Figure 13): one at the error threshold, the other far below the error threshold, and the last one far above it. To depict the composition relationship, the nodes in the macro network and the corresponding nodes in the original network were assigned the same color.

Supplementary References

[1] Rossi, R. & Ahmed, N. The network data repository with interactive graph analytics and visualization. In *Proceedings of the AAAI conference on artificial intelligence*, vol. 29 (2015).

[2] Kovács, T. P., Nagy, M. & Molontay, R. Comparative Analysis of Box-Covering Algorithms for Fractal Networks. *Applied Network Science* **6**, 73 (2021).

[3] Song, C. *et al.* How to calculate the fractal dimension of a complex network: the box covering algorithm. *Journal of Statistical Mechanics: Theory and Experiment* **2007**, P03006 (2007).