Graph Compression for Interpretable Graph Neural Network Inference At Scale

Yangxin Fan
Case Western Reserve University
yxf451@case.edu

Haolai Che
Case Western Reserve University
hxc859@case.edu

Mingjian Lu
Case Western Reserve University
mxl1171@case.edu

Yinghui Wu
Case Western Reserve University
yxw1650@case.edu

ABSTRACT

We demonstrate **ExGIS**, a parallel inference query engine to support explainable Graph Neural Network (GNNs) inference analysis in large graphs. (1) For a class of GNNs $\mathcal{M}^L$ with at most L layers, and a graph G, ExGIS performs an offline, once-for-all compression of G to a small graph G_c, such that for any inference query Q that requests the output of any GNN $M \in \mathcal{M}^L$ on any node v in G, G_c can be directly queried to yield correct output without decompression. (2) Given a workload W of inference queries that requests the output of GNNs from $\mathcal{M}$ over G, ExGIS perform fast online GNN inference and interpretation in parallel. It dynamically partitions W to balance workloads, and (a) executes inference that only consults compressed graph G_c without decompression, and (b) directly yields concise, explanatory subgraphs from G_c that can clarify the query output with high fidelity, all in parallel. Moreover, ExGIS integrates visual, interactive interfaces for query performance analysis, and a Large Language Models (LLMs)- enabled interpreter to support user-friendly, natural language explanation of query outputs. We demonstrate the compression rate and scalability of ExGIS, and its application in interpretable anomaly detection over bitcoin transaction networks and academic networks.

PVLDB Reference Format:
Yangxin Fan, Haolai Che, Mingjian Lu, and Yinghui Wu. Graph Compression for Interpretable Graph Neural Network Inference At Scale. PVLDB, 18(12): 5239 - 5242, 2025.
doi:10.14778/3750601.3750641

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/nicej1899/ExGIS-Demo.

1 INTRODUCTION

Graph Neural Networks (GNNs) have shown promising performance in various analytical tasks. Despite their promising performances, GNNs incur expensive inference cost when G is large [7]. The emerging need for large-scale, reliable data search, analysis and benchmarking require not only fast but also explainable inferences

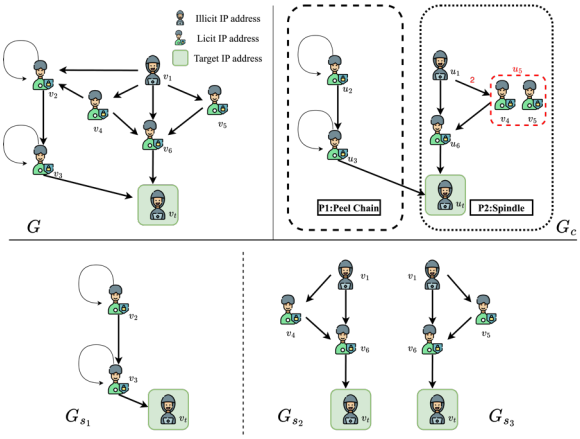

Figure 1: GNN-based anomaly detection in a bitcoin transaction network G. (1) ExGIS conducts an "once-for-all" compression to generate and distribute a compressed graph G_c (top right) for online parallel inference query workload. (2) ExGIS processes fast online inference that detects "illicit" IPs by only consulting G_c, without decompression. (3) For a user designated account v_t (in green box), its online, parallel explainer assembles three factual explanations that are well grounded by real-world strategies [5, 6].

of GNNs where graphs play critical roles. For example, money launderers who exploit Bitcoin blockchain transactions employ various techniques to conceal illicit cryptocurrency activities and evade detection by law enforcement agencies and AI-based monitoring systems [5, 6]. While GNNs have been applied to detect suspicious accounts and activities, it is critical to make GNN inference scalable and interpretable to track them in fast growing transactions.

Example 1: Figure 1 illustrates a bitcoin transaction network G, where a node is an account IP addresses with attributes such as the number and amount of transactions. An edge (v_1, v_2) means IP address v_1 committed a transaction to address v_2. Illicit accounts may utilize transaction patterns such as *Peel Chain* (transfering illicit assets along lengthy, numerous paths [5]) or *Spindle* (multiple small cryptocurrency transactions [6]) to obscure illegal transactions.

Law enforcement agencies aim to uncover money laundry activities, making it essential to detect illicit accounts and generate explanations in large transaction networks. For v_t as designated output nodes, we observe that the node v_4 and v_5 play a same "role" with same input features, and are "indistinguishable" for *any* inference query that involves a GNN with at most two layers and applies a same node update function (a GNN class $\mathcal{M}^2$) [1]. We can thus

Proceedings of the VLDB Endowment, Vol. 18, No. 12 ISSN 2150-8097.
doi:10.14778/3750601.3750641

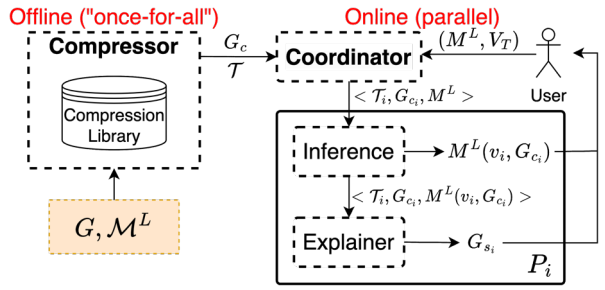

Figure 2: ExGIS Workflow: Overview

safely "merge" v_4 and v_5 and compress G to a smaller counterpart G_c, and only perform inference over G_c to infer v_t's label.

A GNN inference process can be performed over a fragmented compressed graph G_c, *in parallel*, to detected "illicit" accounts. Better still, a GNN explainer can readily generate explanatory subgraphs as factual explanations [2] of the output of v_t, also in parallel over fragments of G_c, to assemble a set of diversified and comprehensive explanations $G_{s_1}, G_{s_2}, G_{s_3}$ to clarify "Why" v_t is illicit, each grounded by a money laundry pattern, respectively. □

ExGIS. We demonstrate ExGIS, a parallel GNN inference and explanation engine to support large-scale and interpretable GNN inference. It has several unique features.

Inference-friendly compression. ExGIS adopts an *inference-friendly graph compression* scheme (IFGC) [1] to perform a "once-for-all" compression that compresses a large graph G into a smaller counterpart G_c with a memoization structure $\mathcal{T}$ that stores statistical neighborhood information of compressed nodes and edges. The compression scheme is provably guaranteed to preserve the output for *any* inference query for a GNN class that requests any GNN with layer L and a node update function of the same form [1], via an efficient inference process directly over G_c without decompression. This new feature is not addressed by prior GNN inference tools.

Interpretable GNN inference: At scale, and in "One-click". ExGIS effectively parallelizes a streamline of "inference-explanation" pipeline, which incorporates GNN inference and configurable explanation into an integrated online querying solution. Users only need to specify a graph, a set of designated test nodes V_T, and a GNN class, In one-click, it automatically creates, schedules, and distributes GNN inference and explanation "joblets" to assmeble the inference output along with explanations.

We demonstrate that ExGIS scale well and incurs a parallel cost that is only determined by the size of the compressed graph G_c and the number of processors, regardless of how large G is. While existing tools separate GNN inference and explanation as two independent task, ExGIS has made a first step parallelizing a holistic "inference-and-explanation" workflow with scalability guarantee.

User-friendly Interface. ExGIS provides an interactive, visual interfaces to allow users to inspect compressed graphs, inference results and explanatory subgraphs to understand GNN behavior. It also provides a natural language (NL) interface, empowered by Large Language Model agents (LLMs) to directly output NL interpretation of the interested output. The visual and textual interpretation provides a comprehensive understanding for GNN behavior in clarifying not only "What" (the labels are) but also "Why".

We invite users to experience the novel capability of ExGIS, as a configurable and scalable tool for explainable GNN inference, with all the above three new features integrated into a same system. No prior graph data systems can address these features simultaneously.

2 SYSTEM OVERVIEW

Workflow. ExGIS works with a coordinator and a set of processors. It performs two major steps (see Fig. 2): an offline, "once-for-all" graph compression, and an online parallelized "Inference-and-Explain" pipeline upon the receiving or inference query workload.

Offline Compression. For an input graph G and a class of GNN models $\mathbb{M}^L$ (see "Structural-preserving Compression"), the coordinator constructs a compression graph G_c and a corresponding memoization table $\mathcal{T}$ (which collects useful auxiliary data) "once-for-all", to be consulted by any inference query (M^L, v) that requests output $M^L(v, G)$ of any GNN M^L in GNN class $\mathbb{M}^L$, for any node v in G.

Online Parallel "Inference-and-Explain". For each inference query (M^L, v_i) from a query workload (M^L, V_T), where V_T is a set of nodes to be queried in G, an online "Inference-and-Explain" module consults the compressed graph G_c and relevant fragments of memoization table to conduct online parallel inference and explanation for each node $v_i \in V_T$, at each processor P_j where v_i resides. P_j then registers inference output and the explanation graph to the coordinator in parallel to be eventually assembled and returned.

Inference-friendly Compression [1]. A GNN class $\mathbb{M}^L$ refers to any set of GNNs with layers up to L and adopt a node update function of the same form. Given a GNN model class $\mathbb{M}^L$ and a graph G, ExGIS induces a compressed graph G_c by merges the nodes that are computationally "*indistinguishable*" based on structural and embedding equivalence (determined by a similarity threshold α of node attribute values). Meanwhile, ExGIS dynamically derives a memoization table $\mathcal{T}$ to "remember" neighborhood statistics (such as node degrees, edge attentions) of merged nodes as "scaling factors" that are needed to restore node embeddings from G_c.

ExGIS employs structural-preserving compression [1]. A compression scheme specifies a pair $(\mathcal{C}, \mathcal{P})$. Compressor $\mathcal{C}$ computes G_c as the quotient graph of an equivalence relation R^S over G, which is the non-empty, maximum equivalence relation that captures node pairs (v, v') that are indistinguishable for inference queries for $\mathbb{M}^L$. $\mathcal{P}$ is an inference function that restores $M^L(v, G)$ by directly scaling up $M^L([v], G_c)$ in G_c (where the node $[v]$ is the equivalence class of v), with a matching scaling factor cached in $\mathcal{T}$. This ensures fast inference over G_c without decompression.

Structural equivalence. Given a graph $G=(X, A)$, a *structural equivalence* relation, denoted as R^S, is a non-empty binary relation such that for any node pair (v, v') in G, $(v, v') \in R^S$, if and only if:

- $X_v^0 = X_{v'}^0$, *i.e.,* v and v' have the same input features;
- for any neighbor u of v ($u \in N(v)$), there exists a neighbor u' of v' (u' in $N(v')$), such that $(u, u') \in R^S$; and
- for any neighbor u'' of v' in $N(v')$, there exists a neighbor u''' of v in $N(v)$, such that $(u'', u''') \in R^S$.

Example 2: Consider G and G_c in Fig. 1 and a GNN class with at most 2 graph convolutional layers that adopts a node

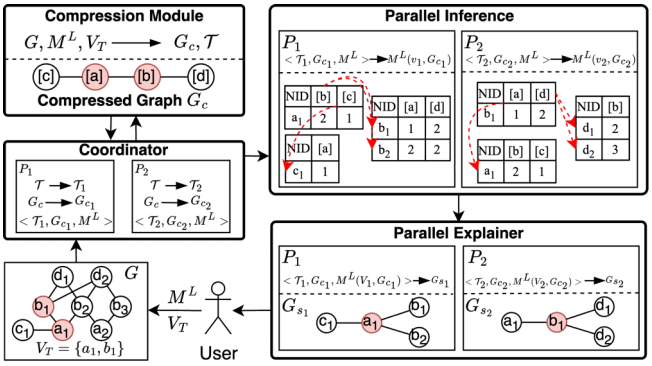

Figure 3: Parallel Inference and Explainer with a running example: $M^L \in \text{GCN}^2$; $V_T = \{a_1, b_1\}$; $\mathcal{T}_1$, $\mathcal{T}_2$: partitioned subsets of $\mathcal{T}$ for a_1 and b_1; G_{s_1}, G_{s_2}: explanation graphs for a_1 and b_1

update function that computes v's embedding at k-th layer as $\sigma(\Theta^k(\sum_{u \in \mathcal{N}(v)} \frac{1}{\sqrt{deg_u deg_v}} x_u^{k-1}))$. As v_4 and v_5 are merged into a node $u_5 = [v_4]$ in G_c, during inference, one just need to "recover" their original embeddings with a scaling factors corresponding to their original *in-degrees* at inference time. For example, for v_4, its embedding is directly rescaled as $\sigma(\Theta^k(\sum_{[u] \in \mathcal{N}([v_4])} \frac{1}{\sqrt{\text{deg}_{u_5}}} \mathcal{T}(\mathbf{v_4}, \mathbf{u_5}) x_{u_5}^{k-1}))$, in constant time. □

GNN Explainers. ExGIS has a built-in library of post-hoc GNN explainers, which computes factual or counterfactual explanatory subgraphs for an output to be explained (*e.g.,* [2]). A factual explanatory subgraph for a node v_i refers to a subgraph $G_s \subseteq G$ such that the GNN model M^L still preserves the same prediction for v_i, *i.e.,* $M^L(v_i, G_s) = M^L(v_i, G)$. A counterfactual explanatory subgraph identifies a subgraph $G'_s \subseteq G$ such that the removal of G'_s leads to the alteration of the prediction, *i.e.,* $M^L(v_i, G \setminus G'_s) \neq M^L(v_i, G)$. These explanation subgraphs help identify the critical graph structures responsible for model predictions. For example, the three subgraphs $G_{s_1}, G_{s_2}, G_{s_3}$ in Fig. 1 are all factual explanation for the output of node v_t. Specially, G_{s_1} is grounded by a peel chain pattern $P1$, and G_{s_2} and G_{s_3} witness spindle pattern $P2$. These explanations not only highlight influential illicit accounts to GNN's decision making, but also reveals their interactions for further inspection.

3 PARALLELIZE "INFERENCE-AND-EXPLAIN"

The demonstration will go through three major modules and algorithms that parallelize the online "inference-and-explain" pipeline, as illustrated in Fig. 3: (1) **Coordinator**. Given the configuration tuple (M^L, V_T) provided by the user, coordinator conduct the following steps: (1) retrieves the corresponding G and $\mathcal{T}$ from the compressor; (2) generates parallel joblets $< \mathcal{T}_i, G_{c_1}, M^L >$ and sends them to processor P_i for all $v_i \in V_T$. (2) **Compressor**. Given a graph G and a class of GNNs $\mathcal{M}^L$, the compressor produces the compressed graph G_c and the memoization structure $\mathcal{T}$. It performs offline "once-for-all" graph compression by leveraging a built-in library of compression methods, including our structural-preserving compression as well as the state-of-the-art graph compression approaches [3, 4]. (3) **Parallel Inference and Explainer**. Given the joblet $< \mathcal{T}_i, G_{c_1}, M^L >$, the module executes the following procedure in parallel (1) computes the inference result $M^L(v_i, G_{c_i})$; and

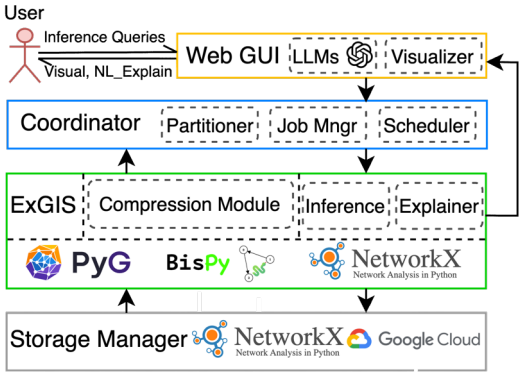

Figure 4: ExGIS Architecture

(2) utilizes $< \mathcal{T}_i, G_{c_i}, M^L(v_i, G_{c_i}) >$ to generate explanation graph G_{s_i}, providing an interpretation of the inference for v_i.

Coordinator. The coordinator optimizes query workload with three components. (1) *Partitioner*: Upon receiving an inference query (M^L, V_T), it initializes a set of processors $\mathcal{P}$. At each $P_i \in \mathcal{P}$, it partitions V_T, the memoization table $\mathcal{T}$ to $\mathcal{T}_i$ and $\mathcal{T}_i$, and induces G_{c_i} accordingly to $v_i \in V_T$. (2) *Load Balancer*: it assigns joblets to processors by distributing query workload among the processors, it minimizes the total time cost of inference and explanation on the entire set of V_T. (3) *Job Scheduler*: reschedules joblets $< \tau_i, G_{c_i}, M^L >$ based on estimated workload and current processor status, and allocate computational resources to dynamically rebalance the parallel "Inference-and-explain" computation.

Pipelining. At each processor P_i, given joblet $< \mathcal{T}_i, G_{c_i}, M^L >$, (1) it performs the inference for assigned $v_i \in V_T$ in parallel to generate output $M^L(v_i, G_{c_i})$. It next generates an explanatory task as a joblet $< \mathcal{T}_i, G_{c_i}, M^L(v_i, G_{c_i}) >$ and sends back to coordinator; (2) Upon receiving an assigned explanatory joblet $< \mathcal{T}_i, G_{c_i}, M^L(v_i, G_{c_i}) >$, it generates an explanation graph G_{s_i} in parallel. The explainer constructs G_{s_i} following the algorithm in [2] by default. The inference and explanation joblets follow a pipelined parallelism among all processes, to ensure that users receive explained output incrementally, rather than waiting or all inference queries to be evaluated.

Parallel Time Cost. The offline compression cost is $O(|G^L|)$ where G^L is the subgraph induced by L-hop neighbors of V_T. The parallel cost per inference query is $O(\frac{L|G_c|F^2}{n})$ where L is the number of layers and F is number of features per node in G_c, and n the number of processors. The parallel explanation cost is $O(|G^L|\frac{|V_T|}{n})$. The overall cost is independent with the original graph size G (see [1]).

System Architecture. The multi-tier ExGIS architecture is deployed on well established data systems, as illustrated in Fig. 4. (1) The *Web GUI* consists of (1) portals to collect metadata of configuration and receiving inference query workload; (2) LLM interfaces for NL interpretations; and (3) a visualizer to support visual analysis of compressed graphs, explanation graphs, and inference-time performance evaluation. (2) The *Coordinator* hosts the job scheduler to i) manage the allocations of computational resources; and ii) coordinate message exchanging among the compression module, explainer, GUI, and storage manager. (3) The *ExGIS Core* contains critical computational modules: offline Compression and online parallel Inference and Explainer, supported by built-in libraries *e.g.,*

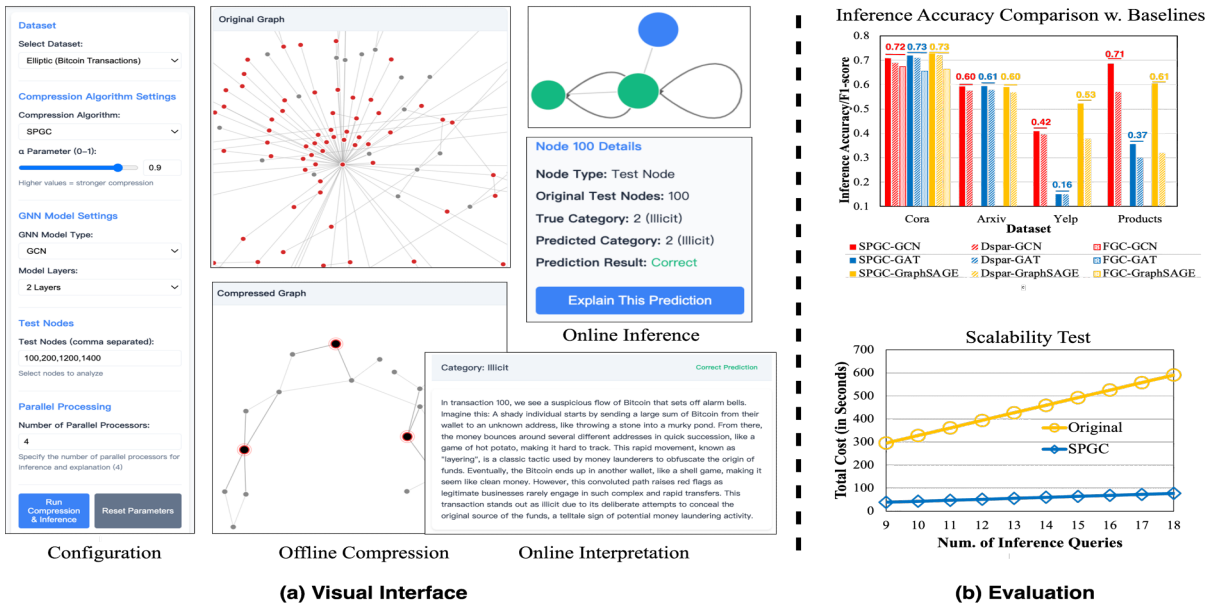

Figure 5: ExGIS: (a) Visual Interfaces; (b) Evaluation: Accuracy over G_c (top); Scalability vs. Workload size (down)

PyTorch Geometric, BisPy, NetworkX and LLMs including Llama 3.3 and GPT-4o. (4) The *Storage Layer* hosts all graphs as NetworkX objects, GNNs, and metadata as JSON objects.

4 DEMONSTRATION SCENARIO

Datasets. We demonstrate ExGIS with Elliptic Bitcoin transactions network (203,769 nodes, 234,355 edges), and Cora citation network (2,708 nodes, 5,429 edges). We report complete tests with two graph compression baseline methods across six diverse datasets [1].

"Inference & Explain" in One-Click. We start with a walkthrough of ExGIS. Users begin by selecting datasets, configuring datasets, compression parameters (compression scheme and parameters), mainstream GNN classes (*e.g.,* GCN, GAT, or GraphSAGE) with layer number, test nodes, and number of processors. With a single click on"Run Compression & Inference," the system automates offline compression to cold start, streamlines the workload optimization for parallel execution. Users can then request ad-hoc inference queries by simply choosing nodes in the compressed graph panel, and readily browse and inspect the visual output and explanations.

Accuracy & Scalability Analysis. Our demonstration highlights ExGIS's performance metrics. The Statistics Analysis tab shows that for Bitcoin transaction networks, ExGIS achieves compression ratios of 92.4% for nodes and 91.3% for edges while maintaining 100% inference accuracy with a 2-layer GCN. The time statistics reveal excellent efficiency: compression (0.039s), inference (0.003s), and explanation generation (0.007s). As shown in Fig. 5, ExGIS maintains competitive accuracy compared to baselines across datasets while offering superior scalability - execution time remains nearly constant for SPGC as queries increase, unlike the linear growth observed with original graphs. Due to limited space, we report more scalability tests and analysis in [1].

Interpretable Network Anomaly Detection. We invite users to experience user cases of ExGIS in two applications.

Illicit IP Account Detection. Using Elliptic Bitcoin transaction network, users can specify inference queries over designated IP account and visually inspect most influential illicit and licit accounts and transactions to GNN output, observe the impact of different distribution of labels to the decision making of GNNs, and be informed by natural language alerts on suspicious activities.

Topic Analysis in Citation Networks. Our second scenario uses the Cora academic citation network. Users can conveniently choose papers of interests and test GNNs of choices to infer potential topics. The explanation reveals influential citations that may influence GNN inference for the particular topic via a subnetwork of citations, hence in turn suggesting useful literature and collaboration.

ACKNOWLEDGMENTS

Che and Wu are supported by NSF OAC-2104007. Fan, Lu, and Wu are supported by DE-NA0004104.

REFERENCES

[1] 2025. Full. https://github.com/Yangxin666/SPGC/blob/main/SPGC_full.pdf
[2] Tingyang Chen, Dazhuo Qiu, Yinghui Wu, Arijit Khan, Xiangyu Ke, and Yunjun Gao. 2024. View-based explanations for graph neural networks. *Proceedings of the ACM on Management of Data* 2, 1 (2024), 1–27.
[3] Manoj Kumar, Anurag Sharma, Shashwat Saxena, and Sandeep Kumar. 2023. Featured graph coarsening with similarity guarantees. PMLR.
[4] Zirui Liu, Kaixiong Zhou, Zhimeng Jiang, Li Li, Rui Chen, Soo-Hyun Choi, and Xia Hu. 2023. DSpar: An Embarrassingly Simple Strategy for Efficient GNN Training and Inference via Degree-Based Sparsification. *TMLR* (2023).
[5] Shiyu Ouyang, Qianlan Bai, Hui Feng, and Bo Hu. 2024. Bitcoin money laundering detection via subgraph contrastive learning. *Entropy* 26, 3 (2024).
[6] Nadia Pocher, Mirko Zichichi, Fabio Merizzi, Muhammad Zohaib Shafiq, and Stefano Ferretti. 2023. Detecting anomalous cryptocurrency transactions: An AML/CFT application of machine learning-based forensics. *Electronic Markets* 33, 1 (2023), 37.
[7] Hongkuan Zhou, Ajitesh Srivastava, Hanqing Zeng, Rajgopal Kannan, and Viktor Prasanna. 2021. Accelerating large scale real-time GNN inference using channel pruning. *arXiv preprint arXiv:2105.04528* (2021).