SGALAD: SUBGRAPH-GUIDED ADAPTATION FOR LOG ANOMALY DETECTION ACROSS DOMAINS

Anonymous authors
Paper under double-blind review

ABSTRACT

Cross-system log anomaly detection must handle changing event vocabularies, dependencies, and anomaly distributions with few target labels. We present SGALAD (Subgraph-Guided Adaptation for Log Anomaly Detection), which learns local event subgraphs and adapts a frozen graph representation. It selects seed events, refines neighborhoods, and fuses local features through attention. Target adaptation updates a projection, classifier, and domain discriminator while preserving the source-trained extractor. On BGL-to-Thunderbird and Thunderbird-to-BGL transfer, SGALAD achieves 10-shot F1 of $96.30 \pm 0.38\%$ and $89.80 \pm 0.42\%$, exceeding size-matched random-subgraph adaptation by 7.90 and 5.50 points. Across 5, 10, and 20 labeled windows per class, it outperforms full GNN fine-tuning. SGALAD updates 0.06 million parameters (4.8% of the full-GNN trainable count), with approximately 8.6–8.8 times shorter measured adaptation time. Fixed-representation discrepancy and fixed-feature estimation bounds motivate both design choices. These results support learned local selection for efficient cross-system adaptation.

1 INTRODUCTION

Operational logs support automated anomaly detection (Xu et al., 2009; Du et al., 2017; Le & Zhang, 2022). Transfer is difficult when event vocabularies, execution dependencies, and anomaly distributions change. Labeling every installation is costly, motivating adaptation with limited target supervision (Zhang et al., 2024; Wang et al., 2024; Oliner & Stearley, 2007).

Graph models capture dependencies beyond individual event semantics (Yan et al., 2023; Wan et al., 2021; Li et al., 2024b; Xie et al., 2022; Li et al., 2023), but whole-graph representations can mix anomaly-related patterns with environmental context. Aligning these representations may emphasize domain similarity without preserving anomaly discrimination. We therefore ask: *can learning which local event structures to retain improve both transfer quality and adaptation efficiency?*

SGALAD learns seed relevance, refines neighborhoods, and fuses local embeddings during source training. At the target, it freezes the structural extractor and trains a compact projection, anomaly classifier, and domain discriminator. Selection emphasizes predictive event patterns, while freezing limits parameters estimated from scarce target labels. A shared-structure hypothesis, motivated by invariant prediction (Carvalho et al., 2023; Fan et al., 2024; Chen et al., 2022), guides representation learning.

We evaluate four log datasets, BGL–TB transfer, OpenStack as an additional target, three support budgets, scarce anomalies, components, adaptation cost, and industrial data. Size-matched random subgraphs isolate learned selection; global features test local representation; full GNN fine-tuning tests optimization scope. At 10-shot, SGALAD improves over random selection by 7.90 and 5.50 F1 points while updating 0.06M parameters. Our contributions are:

- A supervised local event-subgraph representation integrating seed selection, edge pruning, local encoding, and attention fusion.
- Frozen-extractor adaptation supported by fixed-representation discrepancy and fixed-feature estimation analyses.
- Evidence across detection quality, label budgets, component contributions, hyperparameter sensitivity, and measured adaptation efficiency.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

2 RELATED WORK

Log detection and transfer. Sequence models such as DeepLog (Du et al., 2017) and LogAnomaly (Meng et al., 2019), semantic and language-model methods (Liao et al., 2023; Qi et al., 2023; He et al., 2024; Guan et al., 2024; Zhang et al., 2025; Sun et al., 2025; Ma et al., 2026; Zhang et al., 2026; Xiao et al., 2026; Shi et al., 2027), and graph models (Yan et al., 2023; Xie et al., 2022; Chu et al., 2025; Zhao et al., 2026; He et al., 2026; Han et al., 2026; Li et al., 2026; Khanal & Narayan, 2026) capture complementary log features. LogTransfer (Chen et al., 2020), LogTAD (Han & Yuan, 2021), LogGT (Wang et al., 2024), LogETA (Gong et al., 2024), LogDC (Tang & Guan, 2026), and the LLM-based LogSynergy (Sui et al., 2025) address cross-system shifts; MetaLog (Zhang et al., 2024) and LogMeta (Sun et al., 2026) study meta-learning for few-shot adaptation. SGALAD focuses on learning local structures before adaptation and tests this choice with matched internal controls.

Graph generalization. Graph adaptation handles shifts in structure, attributes, and labels (Wang et al., 2023; Ding et al., 2022; Yin et al., 2023; Liu et al., 2024a;b). StableGNN (Fan et al., 2024), CIGA (Chen et al., 2022), and invariant anomaly detection (Carvalho et al., 2023) motivate representations less dependent on environmental correlations. SGALAD couples supervised structural selection with target alignment, requiring the selected representation to retain label information.

Efficient adaptation. SPoT (Vu et al., 2022), LoRA (Hu et al., 2022), AdapterGNN (Li et al., 2024a), and GraphLoRA (Yang et al., 2025) restrict the parameters updated during transfer. SGALAD places its trainable projection after a frozen graph extractor. We evaluate the resulting performance–cost trade-off through F1, trainable parameters, adaptation time, and peak memory at fixed support budgets.

3 PROBLEM FORMULATION

A log window or session is represented by $G = (V, E, X)$ with label $y \in \{0, 1\}$, where 1 denotes an anomaly. Source and target distributions may differ: $P_s(G, y) \neq P_t(G, y)$. We use labeled source data D_s for pretraining, a labeled target support set D_t^l for adaptation, and available unlabeled target training data D_t^u for domain alignment. The target test set D_t^{test} is held out from training and threshold selection. Domain labels distinguish source from target and are separate from anomaly labels.

In a balanced K-shot setting, D_t^l contains K normal and K anomalous windows, for $2K$ support labels; we evaluate $K \in \{5, 10, 20\}$. This budget describes labeled target support. The objective is a target predictor $c_\omega \circ q_\phi \circ f_{\theta^*}$, where f_{θ^*} is the frozen source-trained extractor, q_ϕ is a trainable projection, and c_ω is the target classifier. Selected subgraphs should retain useful anomaly information while reducing dependence on system-specific context. The notation and extended analysis appear in Appendix B.

Figure 1 illustrates the motivating hypothesis: anomaly-related structures may be shared across systems even when their environmental context differs. SGALAD selects local event structures before adapting their representation, aiming to preserve predictive information while reducing dependence on system-specific variation.

4 METHOD

4.1 OVERVIEW AND EVENT GRAPHS

Figure 2 summarizes the two training stages. Source supervision trains the structural extractor and a source classifier. Target adaptation reuses the extractor and trains the downstream projection, target classifier, and domain discriminator. Each target system is adapted separately.

Drain maps messages to event templates. Within each window or session, distinct templates form nodes. We add a directed edge from an earlier event's template to a later event's template when their occurrences are separated by at most o sequence positions; edge weights count temporal co-occurrences. A pretrained Sentence-BERT encoder maps each template to a 768-dimensional vector, yielding $X \in \mathbb{R}^{|V| \times 768}$.

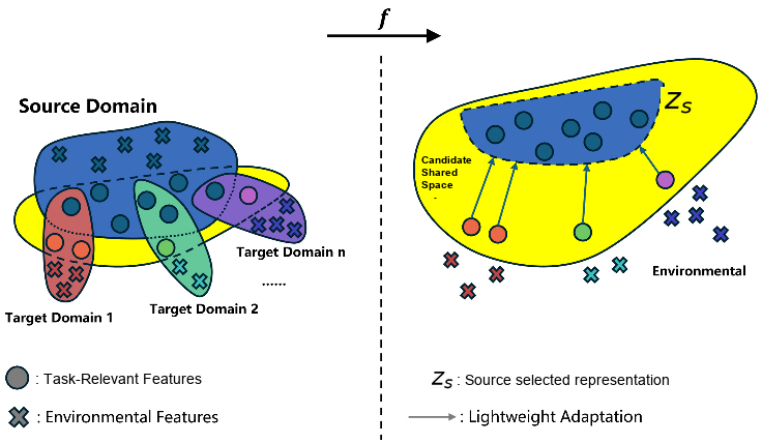

Figure 1: Conceptual source–target representation hypothesis. Circles denote candidate shared features and crosses denote environmental features; adaptation maps target examples toward a shared representation space.

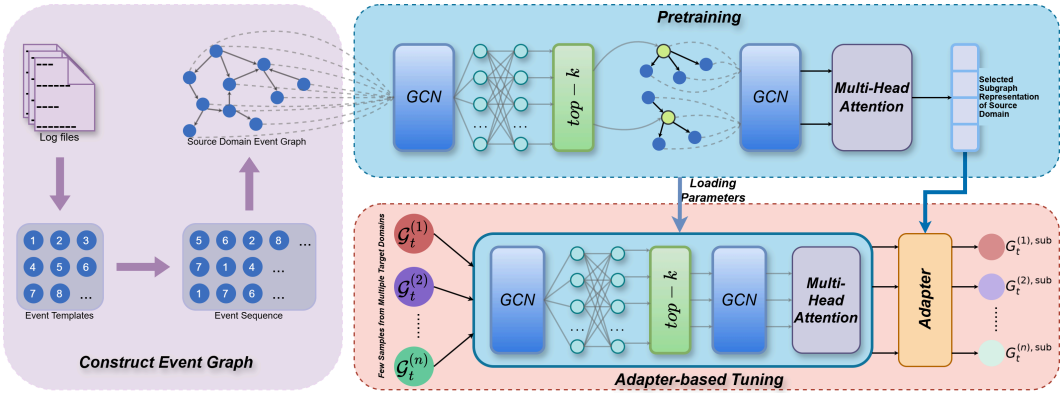

Figure 2: Overall SGALAD architecture. Event graphs feed a source-trained subgraph extractor; target adaptation reuses the extractor and updates downstream modules. The target examples illustrate separate applications to different systems. Superscript sub denotes the selected-subgraph representation.

4.2 Supervised Local Subgraph Learning

Figure 3 details the source-learning pipeline, from node encoding and seed selection to local encoding, attention fusion, and anomaly classification. **Contextual encoding and seed selection.** An L-layer graph encoder produces contextual node features:

$$H^{(0)} = X, \qquad H^{(\ell+1)} = \sigma(\widehat{A}H^{(\ell)}W^{(\ell)}), \tag{1}$$

where $\widehat{A}$ denotes the normalized propagation operator. A two-layer scorer maps a node embedding h_v to its relevance score,

$$a_v = \mathrm{sigmoid}\big(w_2^\top \,\mathrm{ReLU}(W_1 h_v + b_1) + b_2\big)\,. \tag{2}$$

The top-k nodes become seeds. For each seed s, we extract its h-hop neighborhood.

Neighborhood refinement. Candidate edges are scored from their endpoint features:

$$p_{uv} = \mathrm{sigmoid}(\mathrm{MLP}_\psi([h_u \| h_v \| (h_u \odot h_v)]))\,. \tag{3}$$

Here $\|$ is concatenation and $\odot$ is elementwise multiplication. Up to K_e edges are retained in each neighborhood to form $G_s^r = (V_s^r, E_s^r)$. Seed selection controls which local contexts are examined, while edge pruning controls which interactions are passed to the local encoder.

Local encoding and fusion. A local graph encoder maps each retained neighborhood to node features r_v. Mean and max pooling capture complementary summaries, which are projected to a subgraph embedding:

$$z_s = \mathrm{MLP}_{pool}\left[\frac{1}{|V_s^r|}\sum_{v\in V_s^r} r_v \;\|\; \max_{v\in V_s^r} r_v\right]. \tag{4}$$

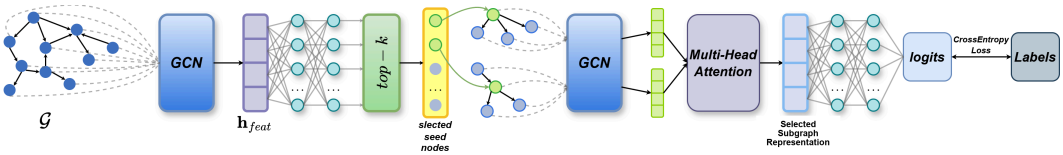

Figure 3: Source subgraph extraction. GCN features drive seed selection; selected neighborhoods are encoded and combined through multi-head attention before anomaly classification.

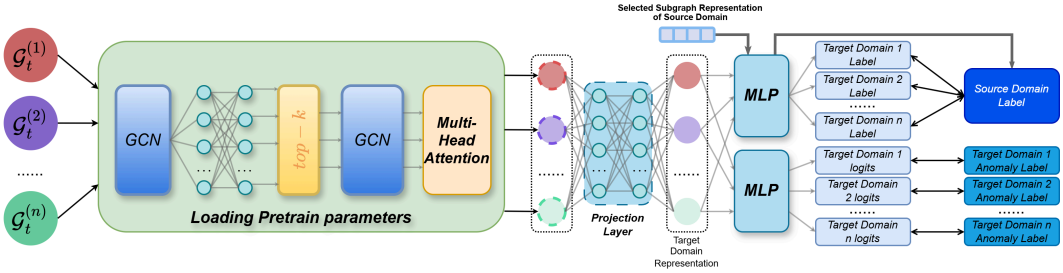

Figure 4: Target-domain adaptation. The green region reuses the source-trained extractor. The projection supplies the domain and anomaly prediction heads, whose objectives are defined in Eqs. (8)–(10).

We weight each pooled embedding by its continuous seed score, $\tilde{z}_s = a_s z_s$, and stack the gated embeddings as $Z = [\tilde{z}_1; \ldots; \tilde{z}_k]$. Weighting by a_s provides an explicit gradient path $\nabla_{\theta_{seed}} L_{src}$ to update the seed scorer during end-to-end source training, where θ_{seed} denotes its parameters. This path differentiates the continuous weights, not the discrete top-k membership decisions. SGALAD then models interactions among neighborhoods and reduces them to a graph vector:

$$U = \mathrm{FFN}(\mathrm{LayerNorm}(\mathrm{MHA}(Z) + Z)), \qquad z = \mathrm{Readout}(U). \tag{5}$$

A source classifier c_{ω_s} predicts the anomaly label from z. Its objective is

$$L_{src}(\theta, \omega_s) = \frac{1}{|D_s|} \sum_{(G,y)\in D_s} \mathrm{CE}(c_{\omega_s}(f_\theta(G)), y). \tag{6}$$

Source supervision learns task-relevant local representations and their aggregation. Algorithm 1 gives the source-stage procedure; implementation conventions for discrete selection are discussed in Appendix A.

4.3 Adaptation of the Frozen Representation

Figure 4 shows the frozen extractor and the downstream projection, domain-prediction branch, and anomaly-prediction branch. For each source or target graph, the frozen extractor produces $z = f_{\theta^*}(G)$. A shared two-layer projection maps these features into a bottleneck space:

$$u = q_\phi(z) = W_2\,\mathrm{ReLU}(W_1 z + b_1) + b_2, \qquad u \in \mathbb{R}^{d_p}. \tag{7}$$

The target classifier c_ω predicts anomalies from u, while the discriminator d_ν predicts the domain. Task supervision uses only labeled target support:

$$L_{task} = \frac{1}{|D_t^l|} \sum_{(G,y)\in D_t^l} \mathrm{CE}(c_\omega(q_\phi(f_{\theta^*}(G))), y). \tag{8}$$

Let B_s and B_t denote source and target alignment minibatches, with target examples drawn from the support and available unlabeled training sets. With $u_G = q_\phi(f_{\theta^*}(G))$, the domain loss uses separately normalized domain contributions:

$$L_{dom} = \frac{1}{|B_s|} \sum_{G\in B_s} \mathrm{CE}(d_\nu(u_G), 0) + \frac{1}{|B_t|} \sum_{G\in B_t} \mathrm{CE}(d_\nu(u_G), 1). \tag{9}$$

Domain labels allow unlabeled target windows to contribute to alignment. The optimization is

$$\min_{\phi,\omega} \max_{\nu} \left[L_{task} - \alpha L_{dom} \right]. \tag{10}$$

The discriminator minimizes L_{dom}; the projection minimizes $L_{task} - \alpha L_{dom}$; and the target classifier minimizes L_{task}. A gradient reversal layer implements the adversarial update to the projection.

The coefficient α increases with training progress according to the schedule in Appendix A. Throughout this stage, the graph extractor stays fixed. At inference, the discriminator is removed, leaving $c_\omega \circ q_\phi \circ f_{\theta^*}$. Algorithm 2 details the updates.

4.4 Analysis of structural selection and adaptation capacity

The two design choices address different parts of transfer. For a fixed representation f and binary predictor class $\mathcal{H}_z$, the standard domain adaptation bound (Ben-David et al., 2010) gives

$$R_t(h \circ f) \leq R_s(h \circ f) + \tfrac{1}{2} d_{\mathcal{H}_z \Delta \mathcal{H}_z}(P_s^z, P_t^z) + \lambda_f, \tag{11}$$

where P_s^z, P_t^z are representation distributions and λ_f is the minimum combined source and target risk in the predictor class. If a graph hypothesis class $\mathcal{H}_G$ contains all $h \circ f$ with $h \in \mathcal{H}_z$, then

$$d_{\mathcal{H}_z \Delta \mathcal{H}_z}(P_s^z, P_t^z) \leq d_{\mathcal{H}_G \Delta \mathcal{H}_G}(P_s^G, P_t^G). \tag{12}$$

Equation (12) bounds representation-level disagreement by that of the containing graph class. The adaptation bound also makes label retention essential through λ_f. SGALAD uses source-supervised selection to retain anomaly information and target alignment to address representation discrepancy.

For a norm-bounded linear predictor on fixed bounded features, the estimation term scales as $O(BR/\sqrt{n})$, with predictor norm B and feature bound R. This fixed-feature result motivates restricting target-stage trainable capacity. Appendix B states its assumptions and proof; random-selection and full-fine-tuning controls test representation quality and optimization scope for SGALAD's nonlinear adapter.

5 Experiments

5.1 Evaluation setup

Coverage and supervision. We evaluate single-domain detection on HDFS, BGL, Thunderbird (TB), and OpenStack (Xu et al., 2009; Du et al., 2017; Oliner & Stearley, 2007; Zhu et al., 2023), cross-system log transfer, and industrial detection on SWaT and WADI (Goh et al., 2017; Ahmed et al., 2017). HDFS uses block sessions, BGL uses six-hour windows with one-hour steps, OpenStack uses one-minute windows, and TB uses a sampled subset. The few-shot study uses $K \in \{5, 10, 20\}$ labeled windows per class, shared support sets and test splits, and three runs. Thus, 10-shot contains 20 support labels. We additionally evaluate BGL-to-OpenStack at 10-shot and vary the number of labeled target anomalies with ten normal support windows fixed. Test data are excluded from training and threshold selection (Appendix C).

Baselines and controls. Single-domain and cross-system benchmarks quote recent methods (2024–2026) and LogTransfer, LogGT, and LogETA from their papers. The expanded few-shot comparison includes LogTransfer, LogDC (Tang & Guan, 2026), AdapterGNN (Li et al., 2024a), and GraphLoRA (Yang et al., 2025), alongside full GNN fine-tuning, random subgraphs, global features, and SGALAD without domain alignment. Random subgraphs match the selection size; global features replace local extraction; full fine-tuning also updates the extractor. Random-subgraph and global-feature controls are evaluated only at 10-shot. We report anomaly F1 and mean $\pm$ standard deviation (SD) for repeated evaluations. Efficiency measurements cover target adaptation rather than source pretraining or end-to-end deployment. Appendix F contains supplementary precision, recall, and label-scarcity metrics; Table 5 reports adaptation resources.

5.2 Detection across log and industrial datasets

Single-domain logs. Table 1 compares SGALAD with recent LLM- and graph-based detectors on HDFS, BGL, and TB. SGALAD has the highest listed TB F1, is the only listed method above 96% on all three datasets, and leads the transfer-capable LogDC by 4.12/0.81 points (BGL/TB). Appendix E reports OpenStack results and complete precision/recall for all four datasets.

Table 1: Single-domain log detection F1 (%). OpenStack results and complete precision/recall are in Appendix E.

Method	Transfer	HDFS	BGL	TB
LLMeLog (He et al., 2024)		94.90	98.30	92.30
LogLLM (Guan et al., 2024)		**99.70**	91.60	96.60
Lograph (Chu et al., 2025)		97.60	97.00	95.30
XRAGLog (Zhang et al., 2025)		83.72	93.15	94.87
LogSwift (Sun et al., 2025)		98.47	**99.05**	—
LogDC (Tang & Guan, 2026)	✓	—	92.19	95.97
SGALAD	✓	98.20	<u>96.31</u>	**96.78**

Quoted from the original papers or the benchmarks of Zhao et al. (2026); Zhang et al. (2026); Shi et al. (2027). Transfer: reports cross-system results. Bold: best; underline: best transfer-capable.

Table 2: Cross-system and industrial detection F1 (%). Panel (a) reports bidirectional log transfer; panel (b) reports detection on SWaT and WADI.

(a) Cross-system logs

Method	BGL→TB	TB→BGL
LogTransfer	79.2	78.8
LogGT	91.9	79.8
LogETA	94.7	88.0
LogDC	94.0	89.3
LogSynergy[†]	98.0	83.4
SGALAD	**96.3**	**89.8**

(b) Industrial time series

Method	SWaT	WADI
CTAD	84.10	48.16
SA2	88.95	62.63
AnoDual	84.60	62.26
ConFlow	90.21	**93.25**
SGALAD	**93.60**	92.66

Bold: best listed F1 per task. [†]Multi-source protocol (two source systems); not ranked. TB: Thunderbird.

Cross-system benchmarks. Table 2a includes the recent LogDC (Tang & Guan, 2026) and LogSynergy (Sui et al., 2025). Among single-source methods, SGALAD leads by 1.6 (BGL-to-TB) and 0.5 (TB-to-BGL) points. The multi-source LogSynergy reaches 98.0% on TB but 83.4% on BGL, 6.4 points below single-source SGALAD. Appendix D provides the complete source/target records. Table 3 further examines target performance across explicit support budgets with an expanded baseline set.

Industrial datasets. Table 2b compares CTAD, SA2, AnoDual, and ConFlow (Kim et al., 2024; Cai et al., 2024; Yao et al., 2025; Li et al., 2025). SGALAD obtains 93.60% F1 on SWaT and 92.66% on WADI, leading on SWaT and trailing ConFlow by 0.59 points on WADI. Additional transfer F1 is 82.7% for BGL-to-SWaT, 74.5% for BGL-to-WADI, 76.8% for SWaT-to-WADI, and 79.7% for WADI-to-SWaT. Appendix D.2 reports industrial precision and recall.

5.3 Few-shot transfer and component contributions

External comparisons and label budgets. Unlike the fully supervised detectors in Table 1, each run here uses only 10–40 target labels. At 10-shot, SGALAD exceeds GraphLoRA by 4.20 and 3.30 F1 points in the two directions. Across 5, 10, and 20 shots, SGALAD improves from 92.15% to 96.85% on BGL-to-TB and from 85.40% to 91.10% in reverse. Its margins over full fine-tuning range from 7.05 to 9.65 points and from 5.20 to 7.60 points, respectively. The advantage therefore persists across the evaluated support budgets; Figure G.1 visualizes these trends.

Selection and alignment. At 10-shot, SGALAD exceeds random subgraphs by 7.90/5.50 F1 points and global features by 6.10/4.70 points. The random-subgraph comparison matches the reported 0.06M trainable parameters; global features use 0.03M and therefore also change adapter capacity. Removing domain alignment reduces F1 by 8.20/6.40 points. The following analysis examines selection training and structural components; Section 5.6 examines hyperparameter sensitivity.

Selection-training variants. Table 4 compares ways to train the seed scorer. SGALAD's soft-gated readout improves over unweighted hard top-k by 3.90/3.00 points and over a detached scorer by 6.15/4.20 points, and performs on par with Gumbel-Softmax (within 0.40 points).

Table 3: Few-shot target F1 (%). External baselines and internal controls across three support budgets.

Method	5-shot	10-shot	20-shot
BGL → Thunderbird			
LogTransfer	71.80 ± 2.15	77.90 ± 1.60	79.10 ± 1.85
LogDC	85.10 ± 1.25	88.70 ± 0.95	91.80 ± 0.70
AdapterGNN	86.40 ± 1.10	89.20 ± 0.85	90.90 ± 0.90
GraphLoRA	87.90 ± 0.95	92.10 ± 0.75	93.60 ± 0.60
Full GNN fine-tuning	82.50 ± 1.40	87.50 ± 0.72	89.80 ± 0.85
Random subgraphs	—	88.40 ± 0.61	—
Global features	—	90.20 ± 0.52	—
SGALAD (w/o alignment)	83.90 ± 1.15	88.10 ± 0.80	91.20 ± 0.65
SGALAD	**92.15** ± 0.45	**96.30** ± 0.38	**96.85** ± 0.42
Thunderbird → BGL			
LogTransfer	69.40 ± 2.40	75.80 ± 1.80	77.20 ± 2.10
LogDC	81.60 ± 1.35	86.20 ± 0.90	88.40 ± 0.75
AdapterGNN	80.50 ± 1.20	84.60 ± 0.85	87.50 ± 0.80
GraphLoRA	82.10 ± 1.05	86.50 ± 0.80	89.10 ± 0.65
Full GNN fine-tuning	77.80 ± 1.50	82.60 ± 0.81	85.90 ± 0.75
Random subgraphs	—	84.30 ± 0.74	—
Global features	—	85.10 ± 0.65	—
SGALAD (w/o alignment)	79.10 ± 1.20	83.40 ± 0.75	86.90 ± 0.60
SGALAD	**85.40** ± 0.58	**89.80** ± 0.42	**91.10** ± 0.48

Mean ± SD over three runs. Each K-shot support set contains K normal and K anomalous windows. Bold: highest mean within each direction and budget. —: not evaluated.

Table 4: Selection-training variants at 10-shot. Target F1 (%).

Variant	BGL→TB	TB→BGL
Uniform random	88.40 ± 0.61	84.30 ± 0.74
Detached scorer	90.15 ± 0.85	85.60 ± 0.90
Hard top-k (unweighted)	92.40 ± 0.70	86.80 ± 0.65
Gumbel-Softmax	95.90 ± 0.65	**90.10** ± 0.55
Soft-gated readout (SGALAD)	**96.30** ± 0.38	89.80 ± 0.42

Mean ± SD over three runs. Bold marks the highest mean within each direction.

Structural components. On HDFS, the full model achieves 98.2% F1, compared with 92.7% without seed selection, 95.9% without edge pruning, and 85.7% without subgraph structure. These correspond to decreases of 5.5, 2.3, and 12.5 points, respectively. The single-domain ablations complement the cross-system controls by examining the contribution of each structural component to anomaly detection.

5.4 Additional target system and scarce anomaly labels

Figure 5a extends the transfer evaluation to a cloud-platform target. SGALAD achieves 87.23±0.85% F1 on BGL-to-OpenStack, compared with 83.78±1.05% for GraphLoRA, a difference of 3.45 points. Table F.1 provides the full precision/recall results.

Figure 5b keeps ten normal support windows and varies anomalous support from one to ten windows. With one anomalous window, SGALAD reaches 82.60±3.80% F1 versus 74.30±4.60% for GraphLoRA. SGALAD's mean increases to 96.30% with ten anomalous windows; Table F.2 gives all values. This experiment evaluates supervised adaptation with scarce anomalous support. The ten-anomaly condition matches the support composition of the 10-shot study.

5.5 Adaptation efficiency

Figure 6 relates target F1 to adaptation time. SGALAD updates 0.06M parameters, compared with 1.25M for full fine-tuning, 0.08M for AdapterGNN, and 0.18M for GraphLoRA. Its measured time

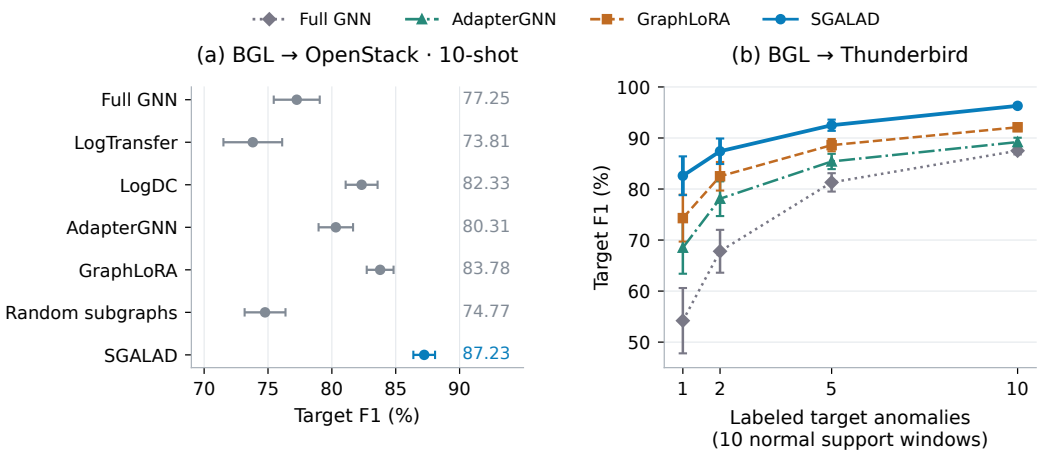

Figure 5: Additional transfer and scarce anomaly labels. (a) BGL-to-OpenStack F1 at 10-shot. (b) BGL-to-TB F1 with ten normal support windows and a varying number of anomalous support windows. Points show means; error bars denote one SD.

Table 5: Adaptation resources at 10-shot.

Method	M	BGL→TB (s)	MiB	TB→BGL (s)	MiB
Full GNN fine-tuning	1.25	32.6±1.5	1,420	34.2±1.8	1,450
LogTransfer	0.32	14.5±0.8	720	15.1±0.9	740
AdapterGNN	0.08	4.8±0.3	560	5.0±0.3	575
GraphLoRA	0.18	11.2±0.6	890	11.8±0.7	910
Global features	0.03	2.1±0.1	360	2.2±0.1	365
Random subgraphs	0.06	3.5±0.2	470	3.6±0.2	475
SGALAD	0.06	3.8±0.2	480	3.9±0.2	485

M: trainable parameters in millions. Time: mean ± SD over three runs. Memory: reported peak GPU memory. LogDC timing was not included in the resource summary.

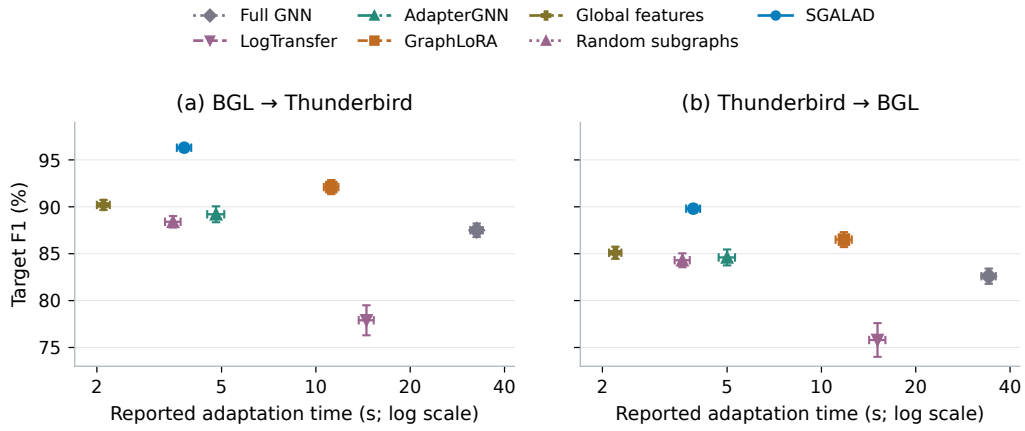

Figure 6: Performance and adaptation time at 10-shot. Horizontal and vertical error bars show one SD over three runs. Time uses a logarithmic scale. Only methods with reported timing measurements are shown; full parameter and memory counts appear in Table 5.

is 3.8/3.9 seconds in the two directions, compared with 32.6/34.2 seconds for full fine-tuning and 11.2/11.8 seconds for GraphLoRA. Peak memory is 480/485 MiB versus 1,420/1,450 MiB for full fine-tuning. Global features are fastest at 2.1/2.2 seconds, while SGALAD obtains higher F1. These measurements characterize adaptation-stage cost, not inference latency or complete deployment cost.

5.6 Hyperparameter sensitivity

Figure 7 examines seed count, neighborhood radius, and alignment strength.

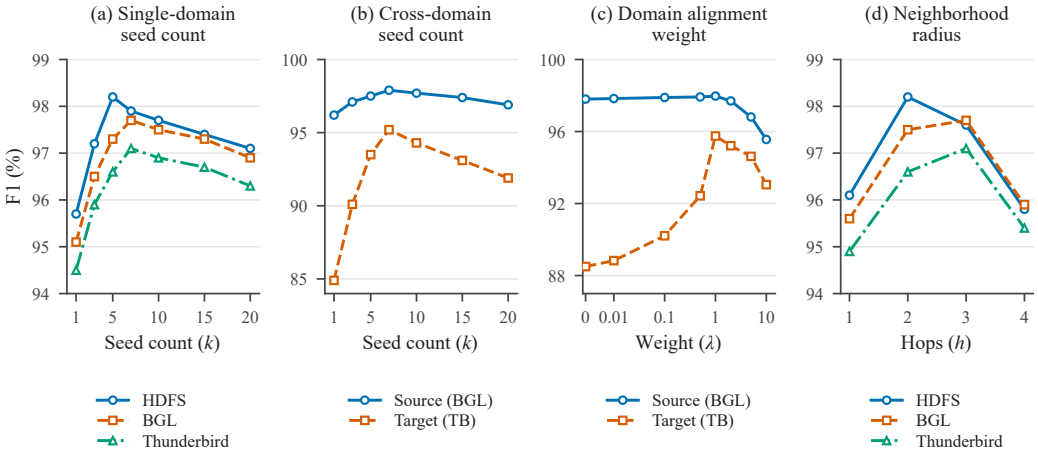

Figure 7: Sensitivity to seed count, alignment weight, and neighborhood radius. Panels (b,c) show source BGL and target TB. Panel (c) is linear from 0 to 0.01 and logarithmic above 0.01. Points show F1 estimates.

Zero alignment weight gives 88.5% F1, consistent with the repeated no-alignment result (88.10±0.80%). The radius peaks at two hops on HDFS and three on BGL/TB, so larger neighborhoods do not always help.

6 Discussion and limitations

SGALAD combines local structural selection with parameter-efficient adaptation for cross-system log detection. The few-shot experiments use fixed source–target pairs and predefined support budgets; continual shifts in event vocabularies remain an open evaluation setting. Future work will study drift-triggered selector updates, uncertainty-guided annotation, and multi-source training.

7 Conclusion

We presented SGALAD, which combines supervised local event-subgraph learning with adaptation of a frozen extractor. Controlled few-shot experiments show that learned selection improves over random subgraphs and global features, while frozen adaptation outperforms full GNN fine-tuning across the evaluated label budgets. Updating 0.06M parameters yields substantial measured time and memory savings relative to full fine-tuning. These findings support local structural selection as a useful foundation for efficient cross-system log detection and motivate extending selective adaptation to continuously evolving operational environments.

AI USE STATEMENT

All ideas and contributions presented in this work originate from the authors, with each author making a substantial contribution to the research. Generative AI was not used to develop the research ideas, methodology, experiments, analysis, or conclusions. AI-assisted tools were used exclusively for language editing and polishing to improve the clarity and readability of the manuscript.

REFERENCES

C. M. Ahmed, V. R. Palleti, and A. P. Mathur. WADI: a water distribution testbed for research in the design of secure cyber physical systems. in *Proceedings of the 3rd International Workshop on Cyber-Physical Systems for Smart Water Networks*, Pittsburgh Pennsylvania: ACM, Apr. 2017, pp. 25–28. doi: 10.1145/3055366.3055375., 2017.

S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. A theory of learning from different domains. *Machine Learning*, vol. 79, no. 1, pp. 151–175, May 2010, doi: 10.1007/s10994-009-5152-4., 2010.

J. Cai, Z. Wei, and J. Luo. ICS Anomaly Detection Based on Sensor Patterns and Actuator Rules in Spatiotemporal Dependency. *IEEE Transactions on Industrial Informatics*, vol. 20, no. 8, pp. 10647–10656, Aug. 2024, doi: 10.1109/TII.2024.3393528., 2024.

J. Carvalho, M. Zhang, R. Geyer, C. Cotrini, and J. M. Buhmann. Invariant anomaly detection under distribution shifts: a causal perspective. in *Advances in Neural Information Processing Systems*, vol. 36, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., Curran Associates, Inc., 2023, pp. 56310–56337. [Online]. Available: `https://proceedings.neurips.cc/paper_files/paper/2023/file/b010241b9f1cdfc7d4c392db899cef86-Paper-Conference.pdf`, 2023.

R. Chen, S. Zhang, D. Li, Y. Zhang, F. Guo, W. Meng, D. Pei, Y. Zhang, X. Chen, and Y. Liu. Log-Transfer: Cross-System Log Anomaly Detection for Software Systems with Transfer Learning. in *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, Oct. 2020, pp. 37–47. doi: 10.1109/ISSRE5003.2020.00013., 2020.

Y. Chen, Y. Zhang, Y. Bian, H. Yang, K. Ma, B. Xie, T. Liu, B. Han, and J. Cheng. Learning causally invariant representations for out-of-distribution generalization on graphs. in *Advances in Neural Information Processing Systems*, vol. 35, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., Curran Associates, Inc., 2022, pp. 22131–22148. [Online]. Available: `https://proceedings.neurips.cc/paper_files/paper/2022/file/8b21a7ea42cbcd1c29a7a88c444cce45-Paper-Conference.pdf`, 2022.

Guojun Chu, Jingyu Wang, Qi Qi, Haifeng Sun, Zirui Zhuang, Bo He, Yuhan Jing, Lei Zhang, and Jianxin Liao. Anomaly detection on interleaved log data with semantic association mining on log-entity graph. *IEEE Transactions on Software Engineering*, 51(2):581–594, 2025. doi: 10.1109/TSE.2025.3527856.

K. Ding, K. Shu, X. Shan, J. Li, and H. Liu. Cross-Domain Graph Anomaly Detection. *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 6, pp. 2406–2415, Jun. 2022, doi: 10.1109/TNNLS.2021.3110982., 2022.

M. Du, F. Li, G. Zheng, and V. Srikumar. DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning. in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, in CCS '17. New York, NY, USA: Association for Computing Machinery, Oct. 2017, pp. 1285–1298. doi: 10.1145/3133956.3134015., 2017.

S. Fan, X. Wang, C. Shi, P. Cui, and B. Wang. Generalizing Graph Neural Networks on Out-of-Distribution Graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 1, pp. 322–337, Jan. 2024, doi: 10.1109/TPAMI.2023.3321097., 2024.

J. Goh, S. Adepu, K. N. Junejo, and A. Mathur. A dataset to support research in the design of secure water treatment systems. in *Critical Information Infrastructures Security (CRITIS 2016)*, Lecture Notes in Computer Science, vol. 10242, G. Havarneanu, R. Setola, H. Nassopoulos, and S.

Wolthusen, Eds., Cham: Springer International Publishing, 2017, pp. 88–99. doi: 10.1007/978-3-319-71368-7_8., 2017.

K. Gong, S. Luo, L. Pan, L. Zhang, Y. Zhang, and H. Yu. LogETA: Time-aware cross-system log-based anomaly detection with inter-class boundary optimization. *Future Generation Computer Systems*, vol. 157, pp. 16–28, Aug. 2024, doi: 10.1016/j.future.2024.03.028., 2024.

Wei Guan, Jian Cao, Shiyou Qian, Jianqi Gao, and Chun Ouyang. LogLLM: Log-based anomaly detection using large language models. *arXiv preprint arXiv:2411.08561*, 2024.

X. Han and S. Yuan. Unsupervised Cross-system Log Anomaly Detection via Domain Adaptation. in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, Virtual Event Queensland Australia: ACM, Oct. 2021, pp. 3068–3072. doi: 10.1145/3459637.3482209., 2021.

Yehong Han, Lin Du, and Jun Zhao. Reinforced temporal graph neural networks for class-imbalanced log anomaly detection. *Expert Systems with Applications*, 301:130438, 2026. doi: 10.1016/j.eswa.2025.130438.

Jiewei He, Chao Song, Ruilin Hu, and Zheng Ren. LogHGAD: A hypergraph-based log anomaly detection method for AIoT systems. *Journal of Systems Architecture*, 170:103630, 2026. doi: 10.1016/j.sysarc.2025.103630.

Minghua He, Tong Jia, Chiming Duan, Huaqian Cai, Ying Li, and Gang Huang. LLMeLog: An approach for anomaly detection based on LLM-enriched log events. In *IEEE 35th International Symposium on Software Reliability Engineering (ISSRE)*, pp. 132–143, 2024. doi: 10.1109/ISSRE62328.2024.00023.

E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-Rank Adaptation of Large Language Models. in *International Conference on Learning Representations (ICLR)*, 2022. [Online]. Available: `https://openreview.net/forum?id=nZeVKeeFYf9`, 2022.

Aja Khanal and Apurva Narayan. State Machine Guided Multi-Relational Synthetic Data from Logs for Anomaly Detection. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pp. 2297–2306, 2026. doi: 10.1145/3770855.3818134.

H. Kim, S. Kim, S. Min, and B. Lee. Contrastive Time-Series Anomaly Detection. *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 10, pp. 5053–5065, Oct. 2024, doi: 10.1109/TKDE.2023.3335317., 2024.

V.-H. Le and H. Zhang. Log-based anomaly detection with deep learning: how far are we? in *Proceedings of the 44th International Conference on Software Engineering*, in ICSE '22. New York, NY, USA: Association for Computing Machinery, Jul. 2022, pp. 1356–1367. doi: 10.1145/3510003.3510155., 2022.

R. Li, Z. Liu, X. Zhu, L. Li, and X. Cao. Detecting Multivariate Time Series Anomalies With Cascade Decomposition Consistency. *IEEE Transactions on Instrumentation and Measurement*, vol. 74, pp. 1–14, 2025, doi: 10.1109/TIM.2025.3547479., 2025.

S. Li, X. Han, and J. Bai. AdapterGNN: Parameter-Efficient Fine-Tuning Improves Generalization in GNNs. *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 12, pp. 13600–13608, Mar. 2024, doi: 10.1609/aaai.v38i12.29264., 2024a.

Y. Li, Y. Liu, H. Wang, Z. Chen, W. Cheng, Y. Chen, W. Yu, H. Chen, and C. Liu. GLAD: Content-Aware Dynamic Graphs For Log Anomaly Detection. in *2023 IEEE International Conference on Knowledge Graph (ICKG)*, Dec. 2023, pp. 9–18. doi: 10.1109/ICKG59574.2023.00007., 2023.

Z. Li, J. Shi, and M. Van Leeuwen. Graph Neural Networks based Log Anomaly Detection and Explanation. in *Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings*, Lisbon Portugal: ACM, Apr. 2024, pp. 306–307. doi: 10.1145/3639478.3643084., 2024b.

Zhong Li, Jiayang Shi, and Matthijs van Leeuwen. Graph neural networks based log anomaly detection and explanation. *Data Mining and Knowledge Discovery*, 40:66, 2026. doi: 10.1007/s10618-026-01235-6.

L. Liao, K. Zhu, J. Luo, and J. Cai. LogBASA: Log Anomaly Detection Based on System Behavior Analysis and Global Semantic Awareness. *International Journal of Intelligent Systems*, vol. 2023, no. 1, p. 3777826, 2023, doi: 10.1155/2023/3777826., 2023.

M. Liu, Z. Fang, Z. Zhang, M. Gu, S. Zhou, X. Wang, and J. Bu. Rethinking Propagation for Unsupervised Graph Domain Adaptation. *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 12, pp. 13963–13971, Mar. 2024, doi: 10.1609/aaai.v38i12.29304., 2024a.

M. Liu, Z. Zhang, J. Tang, J. Bu, B. He, and S. Zhou. Revisiting, benchmarking and understanding unsupervised graph domain adaptation. in *Advances in Neural Information Processing Systems*, vol. 37, A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, Eds., Curran Associates, Inc., 2024, pp. 89408–89436. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2024/file/a2c548b2ed3141400d86a235eaa6fad0-Paper-Datasets_and_Benchmarks_Track.pdf, 2024b.

Jun Ma, Qingjun Xiao, Bingjin Wu, Hang Chen, Yongchao Zhang, and Guang Cheng. LogSnippet: Online frequent snippet mining and multi-event tokenization for compact log representation and anomaly detection. *Knowledge-Based Systems*, 341:115844, 2026. doi: 10.1016/j.knosys.2026.115844.

W. Meng, Y. Liu, Y. Zhu, S. Zhang, D. Pei, Y. Liu, Y. Chen, R. Zhang, S. Tao, P. Sun, and R. Zhou. LogAnomaly: Unsupervised Detection of Sequential and Quantitative Anomalies in Unstructured Logs. in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, Macao, China: International Joint Conferences on Artificial Intelligence Organization, Aug. 2019, pp. 4739–4745. doi: 10.24963/ijcai.2019/658., 2019.

A. Oliner and J. Stearley. What Supercomputers Say: A Study of Five System Logs. in *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)*, Jun. 2007, pp. 575–584. doi: 10.1109/DSN.2007.103., 2007.

J. Qi, S. Huang, Z. Luan, S. Yang, C. Fung, H. Yang, D. Qian, J. Shang, Z. Xiao, and Z. Wu. LogGPT: Exploring ChatGPT for Log-Based Anomaly Detection. in *2023 IEEE International Conference on High Performance Computing & Communications, Data Science & Systems, Smart City & Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*, Dec. 2023, pp. 273–280. doi: 10.1109/HPCC-DSS-SmartCity-DependSys60770.2023.00045., 2023.

Qinxuan Shi, Sicong Shao, Zhanglong Yang, and Brian Terry. A Green Hybrid Transformer-based System Log Anomaly Detection Framework for System Security. *Future Generation Computer Systems*, 186:108695, 2027. doi: 10.1016/j.future.2026.108695.

Yicheng Sui, Xiaotian Wang, Tianyu Cui, Tong Xiao, Chenghao He, Shenglin Zhang, Yuzhi Zhang, Xiao Yang, Yongqian Sun, and Dan Pei. Bridging the Gap: LLM-Powered Transfer Learning for Log Anomaly Detection in New Software Systems. In *Proceedings of the IEEE 41st International Conference on Data Engineering (ICDE)*, pp. 4414–4427, 2025. doi: 10.1109/ICDE65448.2025.00331.

Mingyang Sun, Yuan Chang, Xuecong Wang, and Yan Sun. LogSwift: An efficient log anomaly detection method with lightweight semantic modeling and hierarchical attention. In *Thirteenth International Conference on Advanced Cloud and Big Data (CBD)*, pp. 163–168, 2025. doi: 10.1109/CBD69312.2025.00037.

Yicheng Sun, Jacky Wai Keung, Hi Kuen Yu, and Wenqiang Luo. LogMeta: A few-shot model-agnostic meta-learning framework for robust and adaptive log anomaly detection. *Journal of Systems and Software*, 235:112781, 2026. doi: 10.1016/j.jss.2026.112781.

Pan Tang and Yepeng Guan. Cross-system log anomaly detection based on domain adversarial adaptation and batch contrastive learning. *Future Generation Computer Systems*, 175:108022, 2026. doi: 10.1016/j.future.2025.108022.

T. Vu, B. Lester, N. Constant, R. Al-Rfou', and D. Cer. SPoT: Better Frozen Model Adaptation through Soft Prompt Transfer. in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, S. Muresan, P. Nakov, and A. Villavicencio, Eds., Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 5039–5059. doi: 10.18653/v1/2022.acl-long.346., 2022.

Y. Wan, Y. Liu, D. Wang, and Y. Wen. GLAD-PAW: Graph-Based Log Anomaly Detection by Position Aware Weighted Graph Attention Network. in *Advances in Knowledge Discovery and Data Mining (PAKDD 2021)*, Lecture Notes in Computer Science, vol. 12712, K. Karlapalem, H. Cheng, N. Ramakrishnan, R. K. Agrawal, P. K. Reddy, J. Srivastava, and T. Chakraborty, Eds., Cham: Springer International Publishing, 2021, pp. 66–77. doi: 10.1007/978-3-030-75762-5_6., 2021.

P. Wang, X. Zhang, Z. Cao, W. Xu, and W. Li. LogGT: Cross-system log anomaly detection via heterogeneous graph feature and transfer learning. *Expert Systems with Applications*, vol. 251, p. 124082, Oct. 2024, doi: 10.1016/j.eswa.2024.124082., 2024.

Q. Wang, G. Pang, M. Salehi, W. Buntine, and C. Leckie. Cross-Domain Graph Anomaly Detection via Anomaly-Aware Contrastive Alignment. *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 4, pp. 4676–4684, Jun. 2023, doi: 10.1609/aaai.v37i4.25591., 2023.

Ruizhi Xiao, Yabo Wang, Weilong Li, Jiakun Sun, and Shuyuan Jin. InterpLog: Interpretable log-based anomaly detection assisting troubleshooting for system reliability. *Journal of Systems and Software*, 240:112916, 2026. doi: 10.1016/j.jss.2026.112916.

Y. Xie, H. Zhang, and M. A. Babar. LogGD: Detecting Anomalies from System Logs with Graph Neural Networks. in *2022 IEEE 22nd International Conference on Software Quality, Reliability and Security (QRS)*, Dec. 2022, pp. 299–310. doi: 10.1109/QRS57517.2022.00039., 2022.

W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan. Detecting large-scale system problems by mining console logs. in *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, Big Sky Montana USA: ACM, Oct. 2009, pp. 117–132. doi: 10.1145/1629575.1629587., 2009.

L. Yan, C. Luo, and R. Shao. Discrete log anomaly detection: A novel time-aware graph-based link prediction approach. *Information Sciences*, vol. 647, p. 119576, Nov. 2023, doi: 10.1016/j.ins.2023.119576., 2023.

Zhe-Rui Yang, Jindong Han, Chang-Dong Wang, and Hao Liu. GraphLoRA: Structure-Aware Contrastive Low-Rank Adaptation for Cross-Graph Transfer Learning. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*, pp. 1785–1796, 2025. doi: 10.1145/3690624.3709186.

M. Yao, D. Tao, P. Qi, and R. Gao. Rethinking Discrepancy Analysis: Anomaly Detection via Meta-Learning Powered Dual-Source Representation Differentiation. *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 8579–8592, 2025, doi: 10.1109/TASE.2024.3486688., 2025.

N. Yin, L. Shen, M. Wang, L. Lan, Z. Ma, C. Chen, X.-S. Hua, and X. Luo. CoCo: A Coupled Contrastive Framework for Unsupervised Domain Adaptive Graph Classification. in *Proceedings of the 40th International Conference on Machine Learning*, in Proceedings of machine learning research, vol. 202. PMLR, Jul. 2023, pp. 40040–40053. [Online]. Available: `https://proceedings.mlr.press/v202/yin23a.html`, 2023.

C. Zhang, T. Jia, G. Shen, P. Zhu, and Y. Li. MetaLog: Generalizable Cross-System Anomaly Detection from Logs with Meta-Learning. in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, in ICSE '24. New York, NY, USA: Association for Computing Machinery, Apr. 2024, pp. 1–12. doi: 10.1145/3597503.3639205., 2024.

Lingzhe Zhang, Tong Jia, Mengxi Jia, Yifan Wu, Hongyi Liu, and Ying Li. XRAGLog: Resource-efficient and context-aware log-based anomaly detection method using retrieval-augmented generation. In *AAAI 2025 Workshop on Preventing and Detecting LLM Misinformation (PDLM)*, 2025.

San Zhang, Shiliang Zhou, Yuhang Ma, Qianwen Ding, Chunfang Yang, and Quanbin Du. LogPDGMoE: Log anomaly detection method based on MoE with parallel dual-noise gating network. *Information Sciences*, 748:123498, 2026. doi: 10.1016/j.ins.2026.123498.

Xudong Zhao, Xiaolong Xu, Haolong Xiang, Tong Gao, Lianyong Qi, Amin Beheshti, Xuyun Zhang, and Wanchun Dou. DyLogNet: A Dynamic Multi-Relational Graph Framework for Log Anomaly Detection. In *Proceedings of the ACM Web Conference 2026 (WWW '26)*, pp. 3755–3763, 2026. doi: 10.1145/3774904.3792263.

J. Zhu, S. He, P. He, J. Liu, and M. R. Lyu. Loghub: A Large Collection of System Log Datasets for AI-driven Log Analytics. in *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*, Oct. 2023, pp. 355–366. doi: 10.1109/ISSRE59848.2023.00071., 2023.

A TRAINING PROCEDURES AND IMPLEMENTATION CONVENTIONS

A.1 SOURCE LEARNING AND TARGET ADAPTATION

Algorithms 1 and 2 separate source learning from target adaptation. The source stage optimizes the extractor and a source classifier using source anomaly labels. The target stage freezes the extractor and fits a new projection, target classifier, and domain discriminator. Source and target features pass through the same projection during alignment.

Algorithm 1 Source pretraining

Require: Labeled source data D_s, source validation data, seed count k, neighborhood radius h, and edge budget K_e.
Ensure: Source-trained extractor f_{θ^*} and source classifier.
Initialize extractor and source classifier parameters.
while the source stopping criterion is not met **do**
  Sample a source minibatch and construct event graphs.
  Encode nodes and score seeds using Eqs. (1)–(2).
  Select top-k seeds and extract their h-hop neighborhoods.
  Score edges using Eq. (3); retain up to K_e edges per neighborhood.
  Encode local graphs and pool their features using Eq. (4).
  Weight pooled embeddings by seed scores: $\tilde{z}_s = a_s z_s$.
  Stack gated embeddings, fuse them, and read out using Eq. (5).
  Compute source predictions and binary cross-entropy loss.
  Backpropagate the source loss; seed-scorer gradients pass through $a_s z_s$.
end while
Select a checkpoint using source validation data.
return trained extractor and source classifier.

Algorithm 2 Supervised target adaptation

Require: Source-trained extractor f_{θ^*}, labeled support D_t^l, unlabeled target training set D_t^u, and source adaptation data.
Ensure: Target predictor $c_\omega \circ q_\phi \circ f_{\theta^*}$.
Freeze θ^*; initialize ϕ, ω, and ν.
while the validation and stopping rule permits training **do**
  Sample labeled target, unlabeled target, and source minibatches.
  Compute frozen representations $z = f_{\theta^*}(G)$.
  Project source and target features using $u = q_\phi(z)$.
  Compute L_{task} on labeled target support using Eq. (8).
  Compute L_{dom} with source label 0 and target label 1.
  Set the domain coefficient α using Eq. (A.1).
  $\nu \leftarrow \nu - \eta \nabla_\nu L_{dom}$.
  $\omega \leftarrow \omega - \eta \nabla_\omega L_{task}$.
  $\phi \leftarrow \phi - \eta(\nabla_\phi L_{task} - \alpha \nabla_\phi L_{dom})$.
end while
Discard the discriminator and return $c_\omega \circ q_\phi \circ f_{\theta^*}$.

For normalized training progress $p \in [0, 1]$, the nonnegative domain coefficient is

$$\alpha(p) = \alpha_{max} \left(\frac{2}{1 + \exp(-\gamma p)} - 1 \right), \qquad \gamma > 0. \tag{A.1}$$

The coefficient is applied once to the reversed domain gradient for the projection. It does not reverse the discriminator's own minimization of domain cross-entropy. The extractor is unchanged throughout target adaptation.

A.2 SELECTION AND GRAPH REPRESENTATION

Top-k membership is discrete. SGALAD weights each pooled subgraph embedding by its seed score, $\tilde{z}_s = a_s z_s$, before the attention fusion in Eq. (5). The source loss updates the seed scorer through

these continuous weights, while the selected indices are treated as fixed during backpropagation. The graph-level readout reduces the k attention rows to one vector.

Repeated occurrences of a template are merged into one node, and temporal co-occurrences determine the event-graph edges. The resulting representation combines template semantics with local event dependencies.

B Analysis for fixed representations and bounded predictors

B.1 Notation

Table B.1: Notation for the model and analysis.

Symbol	Meaning
$G = (V, E, X)$	Event graph, temporal edges, and node features
D_s	Labeled source training examples
D_t^l, D_t^u	Target support and unlabeled target training examples
D_t^{val}, D_t^{test}	Target validation and held-out test examples
k, h, K_e	Seed count, neighborhood radius, retained edge budget
f_θ, z	Structural extractor and fused graph representation
q_ϕ, u	Adaptation projection and projected representation
c_ω, d_ν	Task classifier and domain discriminator
a_v, p_{uv}	Node and edge relevance scores
λ_f	Minimum joint source and target risk for representation f

B.2 Discrepancy under a fixed representation

Proposition 1. Let f be a fixed measurable representation, $\mathcal{H}_z$ a binary hypothesis class on the representation space, and $\mathcal{H}_G$ a graph hypothesis class containing every $h \circ f$ for $h \in \mathcal{H}_z$. For graph marginals P_s^G, P_t^G and their pushforward distributions P_s^z, P_t^z,

$$d_{\mathcal{H}_z \Delta \mathcal{H}_z}(P_s^z, P_t^z) \le d_{\mathcal{H}_G \Delta \mathcal{H}_G}(P_s^G, P_t^G), \tag{B.1}$$

where $d_{\mathcal{H}\Delta\mathcal{H}}(P, Q) = 2 \sup_{h,h' \in \mathcal{H}} |\Pr_P(h \neq h') - \Pr_Q(h \neq h')|$. The inequality also holds for empirical distributions formed from the same source and target samples transformed by f.

Proof. Every disagreement event between $h(f(G))$ and $h'(f(G))$ belongs to the disagreement class induced by $\mathcal{H}_G$. The supremum on the left is therefore over a subset of the events on the right. Class containment yields the stated bound without assuming strict contraction. □

For the same representation and predictor class, the adaptation bound (Ben-David et al., 2010) is

$$R_t(h \circ f) \le R_s(h \circ f) + \frac{1}{2} d_{\mathcal{H}_z \Delta \mathcal{H}_z}(P_s^z, P_t^z) + \lambda_f, \tag{B.2}$$

with $\lambda_f = \inf_{h \in \mathcal{H}_z} [R_s(h \circ f) + R_t(h \circ f)]$. Risks use binary zero-one loss and the same composed predictor. The representation must retain label information and admit a predictor that works in both domains: a constant representation has zero discrepancy but can have large joint risk. Conditional label stability on common support motivates a small joint-risk term. Adversarial training provides a discriminator-based alignment surrogate.

B.3 Estimation for a fixed feature map

Proposition 2. Let f be fixed independently of n independent labeled target examples and satisfy $\|f(G)\|_2 \le R$. Consider linear scores $g_w(G) = w^\top f(G)$ with $\|w\|_2 \le B$, and a loss bounded in $[0, 1]$ and L-Lipschitz in its score. With probability at least $1 - \delta$, simultaneously for this class,

$$R_\ell(g_w) \le \widehat{R}_\ell(g_w) + \frac{2LBR}{\sqrt{n}} + 3\sqrt{\frac{\log(2/\delta)}{2n}}. \tag{B.3}$$

Proof. Let $z_i = f(G_i)$ and let ϵ_i be independent Rademacher signs. The empirical complexity of the linear score class is bounded by

$$\widehat{\mathfrak{R}}_n = \frac{1}{n}\mathbb{E}_\epsilon \sup_{\|w\|_2 \le B} \sum_{i=1}^{n} \epsilon_i w^\top z_i \le \frac{B}{n}\mathbb{E}_\epsilon \left\| \sum_{i=1}^{n} \epsilon_i z_i \right\|_2$$
$$\le \frac{B}{n}\left(\sum_{i=1}^{n} \|z_i\|_2^2 \right)^{1/2} \le \frac{BR}{\sqrt{n}}.$$

Loss contraction multiplies this bound by L. Applying the bounded-loss uniform deviation inequality gives Eq. (B.3). A bias can be included by augmenting the feature vector with corresponding norm bounds. □

This proposition concerns a controlled linear class on frozen features. Extending it to SGALAD's trainable nonlinear projection and classifier requires explicit bounds on the layer norms and loss. Cross-entropy additionally requires a bounded-logit or clipped-loss restriction to satisfy the stated bounded-loss premise. These propositions provide a conditional analysis of representation discrepancy and estimation on fixed features.

C Evaluation context

Datasets and units. The log benchmarks cover HDFS (Xu et al., 2009; Zhu et al., 2023), BGL and Thunderbird (Oliner & Stearley, 2007; Zhu et al., 2023), and OpenStack (Du et al., 2017; Zhu et al., 2023). HDFS uses block sessions. The described BGL graph construction uses six-hour windows with one-hour steps, and OpenStack uses one-minute windows. Thunderbird uses a sampled subset. The industrial experiments use SWaT (Goh et al., 2017) and WADI (Ahmed et al., 2017).

Supervision and splitting. The few-shot study reports shared support sets and test splits for the compared methods. At 10-shot, each support set contains 20 windows in total. The shot count refers to labeled support windows. Test data are excluded from training and threshold selection.

Statistics and costs. The few-shot F1 and adaptation-time results report three-seed means and standard deviations. Benchmark and sensitivity results are point estimates from their respective settings. A standard deviation measures run-to-run variation and is not a confidence interval. Trainable parameter counts describe the optimized graph-model configurations. Time and peak memory characterize target adaptation; they do not represent complete deployment cost.

Comparison protocols. The controlled few-shot and resource comparisons use shared support sets, test splits, and repeated runs. Single-domain and cross-system baseline scores quoted from prior publications provide context under each publication's original windowing, splitting, and label-budget protocol.

Area metrics. The extended benchmark reports F1 and AUC. F1 is the primary metric for the controlled few-shot comparison.

D EXTENDED BENCHMARK RESULTS

Main-text Table 1 reports single-domain F1 on HDFS, BGL, and Thunderbird. This appendix supplies complete cross-system source/target results and industrial precision/recall; complete single-domain metrics, including OpenStack, follow in Appendix E.

D.1 COMPLETE CROSS-SYSTEM LOG RESULTS

Table D.1: Complete cross-system log benchmark results.

	Source		Target	
Method	F1	AUC	F1	AUC
BGL → Thunderbird				
LogTransfer	0.971	0.972	0.792	0.828
LogTAD	0.926	0.964	0.758	0.804
LogGT	0.995	0.964	0.919	0.908
LogETA	0.957	0.959	0.947	0.963
LogDC (Tang & Guan, 2026)	0.922	—	0.940	—
SGALAD	0.974	0.955	**0.963**	**0.965**
Thunderbird → BGL				
LogTransfer	0.995	0.995	0.788	0.833
LogTAD	0.788	0.797	0.845	**0.909**
LogGT	0.985	0.991	0.798	0.801
LogETA	0.997	0.996	0.880	0.898
LogDC (Tang & Guan, 2026)	0.960	—	0.893	—
SGALAD	0.986	0.976	**0.898**	0.899

Scores use a 0–1 scale. Bold: highest target value within each transfer direction. LogDC source F1 is its reported single-system F1 on the source dataset; —: AUC not reported. LogTAD is retained here for completeness. LogSynergy (Sui et al., 2025) uses a multi-source protocol and reports target F1 of 0.980 (TB) and 0.834 (BGL).

SGALAD has the highest listed target F1 in both directions. Its AUC on Thunderbird-to-BGL is 0.899, below LogTAD's 0.909, illustrating that thresholded F1 and an area metric need not rank methods identically.

D.2 ADDITIONAL INDUSTRIAL TIME-SERIES RESULTS

Table D.2: Industrial time-series detection. Precision, recall, and F1 in percent.

	SWaT			WADI		
Method	P	R	F1	P	R	F1
CTAD (Kim et al., 2024)	**97.10**	74.20	84.10	79.20	34.60	48.16
SA2 (Cai et al., 2024)	96.20	82.71	88.95	77.77	52.43	62.63
AnoDual (Yao et al., 2025)	96.54	75.29	84.60	66.20	58.76	62.26
ConFlow (Li et al., 2025)	89.06	91.38	90.21	92.63	**93.88**	**93.25**
SGALAD	91.00	**96.36**	**93.60**	**93.79**	91.56	92.66

Bold: highest listed value in each column. Shade: SGALAD.

Table D.2 reports industrial single-domain results. Target F1 is 0.827 for BGL-to-SWaT, 0.745 for BGL-to-WADI, 0.768 for SWaT-to-WADI, and 0.797 for WADI-to-SWaT. These results extend the evaluation beyond system logs.

E COMPLETE SINGLE-DOMAIN LOG METRICS

OpenStack appears here because only three of the listed baselines report single-domain results on that dataset.

Table E.1: Single-domain detection on four log datasets. Precision, recall, and F1 (%).

	HDFS			BGL		
Method	P	R	F1	P	R	F1
DeepLog (Du et al., 2017)	92.98	90.09	91.51	92.31	90.91	91.60
LLMeLog (He et al., 2024)	92.50	97.50	94.90	97.40	99.30	98.30
LogLLM (Guan et al., 2024)	**99.40**	**100.00**	**99.70**	86.10	97.90	91.60
Lograph (Chu et al., 2025)	96.80	98.60	97.60	98.50	95.70	97.00
XRAGLog (Zhang et al., 2025)	92.32	76.59	83.72	87.18	**100.00**	93.15
LogSwift (Sun et al., 2025)	97.08	99.91	98.47	99.65	98.47	99.05
LogMeta (Sun et al., 2026)	**99.40**	99.60	99.50	99.30	99.20	99.30
LogSnippet (Ma et al., 2026)	99.30	99.30	99.30	99.10	98.00	98.50
LogPDGMoE (Zhang et al., 2026)	94.80	99.54	97.11	99.61	98.73	99.17
DyLogNet (Zhao et al., 2026)	98.50	99.20	98.80	99.60	97.60	98.60
GreenHT-SLAD (Shi et al., 2027)	97.82	**100.00**	98.90	**99.99**	**100.00**	**99.99**
LogDC* (Tang & Guan, 2026)	—	—	—	86.17	99.12	92.19
SGALAD*	97.50	98.91	98.20	93.52	<u>99.27</u>	<u>96.31</u>

	Thunderbird			OpenStack		
Method	P	R	F1	P	R	F1
DeepLog (Du et al., 2017)	69.20	80.31	74.34	64.94	92.87	76.43
LLMeLog (He et al., 2024)	93.50	91.30	92.30	—	—	—
LogLLM (Guan et al., 2024)	96.60	96.60	96.60	—	—	—
Lograph (Chu et al., 2025)	95.70	95.10	95.30	—	—	—
XRAGLog (Zhang et al., 2025)	94.87	94.87	94.87	—	—	—
LogSwift (Sun et al., 2025)	—	—	—	82.43	73.49	77.71
LogMeta (Sun et al., 2026)	99.30	99.00	99.20	—	—	—
LogSnippet (Ma et al., 2026)	**99.70**	99.50	**99.60**	—	—	—
LogPDGMoE (Zhang et al., 2026)	99.25	**99.54**	99.39	—	—	—
DyLogNet (Zhao et al., 2026)	97.60	96.20	96.80	—	—	—
GreenHT-SLAD (Shi et al., 2027)	—	—	—	**99.22**	**100.00**	**99.61**
LogDC* (Tang & Guan, 2026)	96.75	95.21	95.97	—	—	—
SGALAD*	94.39	99.30	<u>96.78</u>	95.40	97.63	96.50

Baseline values are reported by the original papers under their own protocols; LLMeLog and Lograph are taken from Zhao et al. (2026), LogLLM and XRAGLog from Zhang et al. (2026), and LogSwift from Shi et al. (2027). *Methods that also support cross-system transfer. Bold: highest listed value per column; underline: highest F1 among transfer-capable methods. Shade: SGALAD. —: not reported.

F ADDITIONAL TRANSFER AND LABEL-SCARCITY RESULTS

The following tables provide exact metrics underlying the additional target and label-scarcity figure. Adaptation resource measurements are reported in main-text Table 5. Means and SDs are reported as separate quantities; error bars in the corresponding figures denote SD rather than confidence intervals.

Table F.1: BGL-to-OpenStack transfer at 10-shot. Precision, recall, and F1 (%).

Method	Precision	Recall	F1
Full GNN fine-tuning	75.40 ± 2.10	79.20 ± 2.40	77.25 ± 1.80
LogTransfer	71.30 ± 2.50	76.50 ± 2.80	73.81 ± 2.30
LogDC	81.20 ± 1.40	83.50 ± 1.60	82.33 ± 1.25
AdapterGNN	78.60 ± 1.50	82.10 ± 1.70	80.31 ± 1.35
GraphLoRA	82.40 ± 1.10	85.20 ± 1.30	83.78 ± 1.05
Random subgraphs	73.20 ± 1.80	76.40 ± 2.00	74.77 ± 1.60
SGALAD	$\mathbf{86.10} \pm 0.90$	$\mathbf{88.40} \pm 1.10$	$\mathbf{87.23} \pm 0.85$

Mean $\pm$ SD over three runs.

Table F.2: BGL-to-TB target F1 (%) with scarce anomalous support.

Method	1 anomaly	2 anomalies	5 anomalies	10 anomalies
Full GNN fine-tuning	54.20 ± 6.40	67.80 ± 4.20	81.30 ± 1.80	87.50 ± 0.72
AdapterGNN	68.50 ± 5.10	78.10 ± 3.40	85.40 ± 1.50	89.20 ± 0.85
GraphLoRA	74.30 ± 4.60	82.50 ± 2.80	88.60 ± 1.20	92.10 ± 0.75
SGALAD	$\mathbf{82.60} \pm 3.80$	$\mathbf{87.40} \pm 2.50$	$\mathbf{92.50} \pm 1.10$	$\mathbf{96.30} \pm 0.38$

Ten normal support windows are fixed. Entries are mean $\pm$ SD. The ten-anomaly column has the same values as the corresponding 10-shot entries in Table 3.

G SUPPORT-BUDGET VISUALIZATION

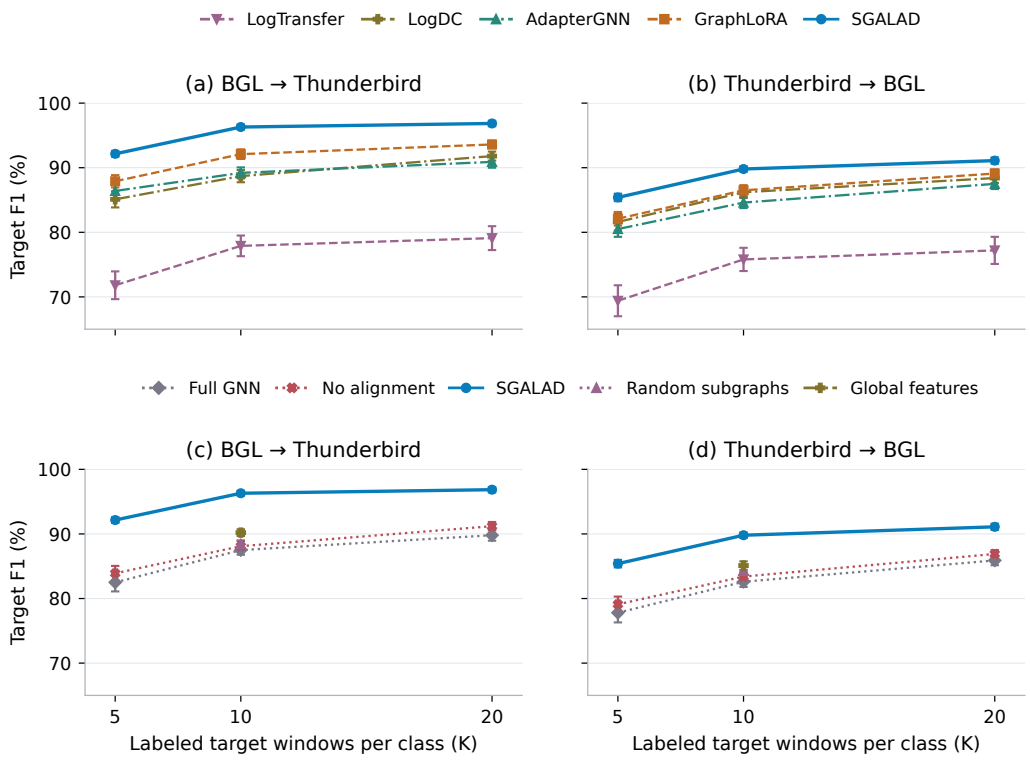

Figure G.1: Target F1 across support budgets. Panels (a,b) compare external baselines and SGALAD; panels (c,d) show internal controls. Error bars represent one SD over three runs. Random-subgraph and global-feature controls were evaluated only at $K = 10$.