ANALOGQUAD: LEARNING ANALOG CIRCUIT REPRESENTATIONS THROUGH TEMPLATE–INSTANCE NEURAL-SYMBOLIC ALIGNMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Aligning analog circuit netlists with symbolic descriptions requires capturing both reusable structure and parameter-dependent behavior. Template-level alignment alone does not distinguish realizations with different device sizes and operating points. We introduce ANALOGQUAD, a graph–symbolic pretraining framework that connects template and sized-instance views in both modalities. Sizing injection adds device values to graph attributes and instantiates supplied symbolic expressions. A graph encoder and a hierarchical symbolic encoder share weights across domains within each modality, and symmetric contrastive learning aligns the resulting pairs at both levels. Experiments demonstrate improved cross-modal retrieval and transfer to downstream functional annotation and sizing-performance classification. [1]

1 INTRODUCTION

Analog circuits interface computation with the physical world, from sensing and communication to power delivery (Gray et al., 2009; Sansen, 2007). However, analog design remains labor-intensive, requiring expert judgment and repeated design iterations (Xu et al., 2024; Mehradfar et al., 2025). A key challenge is to learn reusable representations that support retrieval, functional annotation, and design analysis across circuits with related functions. Connectivity alone does not specify behavior: circuits with the same topology can have different gain, bandwidth, and stability because their transistor dimensions, passive values, and operating points differ. A useful representation must therefore connect reusable circuit structure to the parameter-dependent behavior of its concrete sizing realizations.

Graph neural networks support circuit annotation, generation, and synthesis (Kunal et al., 2020; Dong et al., 2023; Mehradfar et al., 2025), with pretraining signals drawn from operating points, graph augmentations, and electrical constraints (Hakhamaneshi et al., 2022; Lee et al., 2025; Xu et al., 2026). Symbolic circuit analysis offers complementary supervision by expressing gain, poles, zeros, and transfer functions in circuit quantities (Gielen et al., 1989; 1994; Daems et al., 2002). These expressions expose parameter dependence directly. We use symbolic records produced by the signal-path-based extraction method of Li & Shi (2026).

This symbolic view has a useful template–instance structure. A parameterized netlist G_t^T and parameterized symbolic record S_t^T describe a circuit template t. A named sizing assignment $\boldsymbol{\xi}$ gives rise to a sized graph and an instantiated symbolic record:

$$\left(G_t^T, S_t^T\right) \xrightarrow{\boldsymbol{\xi}} \left(G_{t,\boldsymbol{\xi}}^I, S_{t,\boldsymbol{\xi}}^I\right). \tag{1}$$

$S_{t,\boldsymbol{\xi}}^I$ contains numerical substitutions while retaining the symbolic schema and expression hierarchy of S_t^T. Figure 2 illustrates the corresponding circuit, graph, and symbolic views. Template matching can exploit circuit-family identity, whereas instance matching also depends on parameter values. When negatives mostly belong to different templates, graph–symbolic alignment may identify pairs from topology alone. This motivates the question: ***how can graph–symbolic pretraining preserve reusable circuit structure while distinguishing its sizing realizations?***

[1] We will open-source the dataset and code upon publication.

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.

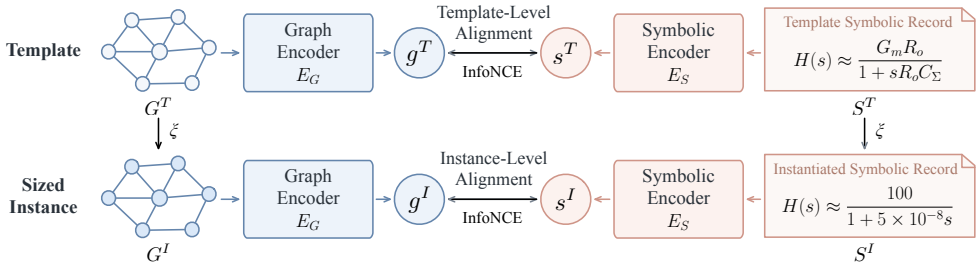


Figure 1: AnalogQuad overview. A sizing assignment instantiates the template graph and symbolic record. Modality-specific encoders share weights across the two levels, and symmetric InfoNCE objectives align each graph with its paired symbolic record.

ANALOGQUAD addresses this question through three operations, as shown in Figure 1. First, *sizing injection* constructs graph and symbolic instance views from their templates by adding device values to graph nodes and substituting circuit quantities into expressions. Second, a graph encoder and a hierarchical symbolic encoder share weights across domains within each modality; the symbolic encoder retains expression branches, entry metadata, and type counts. Third, symmetric contrastive learning aligns graph–symbolic pairs at both template and instance levels. We evaluate the resulting representations through *cross-modal retrieval* and *downstream tasks*. For retrieval, ANALOGQUAD achieves the highest MRR among the compared configurations in both directions and domains, while sizing adaptation improves matching over direct template-checkpoint transfer across all four backbones. For downstream tasks, pretrained graph encoders improve performance-prediction AUROC and functional-annotation macro-F1.

Our contributions are:

- **Template–instance graph–symbolic learning.** We formulate learning across two modalities and two abstraction levels, using deterministic sizing injection to construct paired graph and symbolic inputs.
- **Structured symbolic encoding and alignment.** We encode expression branches, entry metadata, and type counts hierarchically, and align the resulting symbolic records with circuit graphs through a shared symmetric contrastive objective.
- **Retrieval and downstream evaluation.** Experiments demonstrate improved cross-modal retrieval in both domains and effective transfer of pretrained graph encoders to functional annotation and sizing-performance prediction.

2 RELATED WORK

Circuit representation learning. Circuit graph encoders differ in the design information they learn to predict. GANA annotates functional structures in netlists (Kunal et al., 2020); CktGNN uses nested graph representations for topology generation and device sizing (Dong et al., 2023); FALCON learns a circuit-performance model for layout-constrained parameter optimization (Mehradfar et al., 2025). Pretraining provides reusable graph features through other signals: Hakhamaneshi et al. (2022) predict simulated DC node voltages, DICE contrasts graph augmentations without simulation (Lee et al., 2025), and KCLNet constrains message passing and embeddings through electrical equivalence and Kirchhoff's current law (Xu et al., 2026). We use GraphCL's graph-augmentation contrast as a structure-only reference (You et al., 2020). ANALOGQUAD introduces a complementary modality tailored to analog design: symbolic records that express small-signal behavior in circuit parameters.

Multimodal alignment. CLIP aligns global representations of paired images and text (Radford et al., 2021), while FILIP and FG-CLIP introduce finer interactions or supervision (Yao et al., 2022; Xie et al., 2025). For digital circuits, CircuitFusion fuses hardware code, structural graphs, and functional summaries and learns from equivalent transformations (Fang et al., 2025); UniRTL aligns RTL code and control data flow graphs through mutual masked modeling (Liu et al., 2026). AnalogRetriever learns a shared retrieval space for analog netlists, schematics, and functional descriptions using contrastive training (Wang et al., 2026). ANALOGQUAD aligns circuit graphs with parameter-dependent small-signal records at template and instance levels.

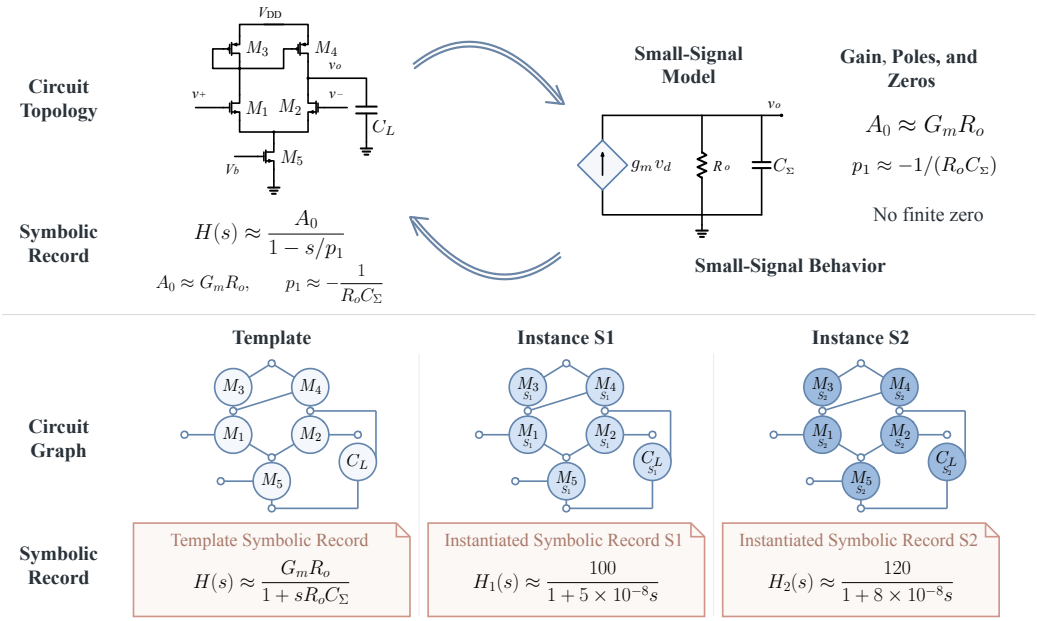


Figure 2: Circuit, graph, and symbolic views of a five-transistor operational transconductance amplifier. Small-signal abstraction connects the circuit topology to its parameterized symbolic behavior. The lower rows show the template and two illustrative sizing assignments; S1 and S2 in the artwork denote assignments rather than training stages.

Symbolic structure and parameter dependence. Symbolic circuit analysis derives parameter-dependent expressions for network behavior to support analysis and sizing (Gielen et al., 1989; 1994; Daems et al., 2002). Formula retrieval studies how to retain mathematical structure in learned representations (Amador & Zanibbi, 2025). DUET and Self-supervised Transformation Learning address a related representation question: retaining transformation information that invariant embeddings may discard (Suau et al., 2023; Yu et al., 2024). In AnalogQuad, sizing changes graph attributes and symbolic expressions together while preserving connectivity and record schema, enabling alignment at both template and instance levels.

3 Problem Formulation

Two modalities at two levels. A circuit template t comprises a parameterized netlist G_t^T and a parameterized symbolic record S_t^T with available descriptions, poles, zeros, and transfer functions. We denote a named assignment of circuit quantities by $\boldsymbol{\xi}$, reserving Φ for learned model parameters. Deterministic sizing-injection operators construct the corresponding instance views:

$$G^I_{t,\boldsymbol{\xi}} = \mathcal{I}_G(G_t^T; \boldsymbol{\xi}), \qquad S^I_{t,\boldsymbol{\xi}} = \mathcal{I}_S(S_t^T; \boldsymbol{\xi}). \tag{2}$$

The graph operator $\mathcal{I}_G$ populates device-sizing features while preserving connectivity. The symbolic operator $\mathcal{I}_S$ substitutes available circuit quantities into the supplied expressions, retaining their schema and expression hierarchy. Each operator uses the quantities relevant to its modality (Section 4.2). Thus, (G^T, S^T, G^I, S^I) denotes four observed view types: two template views and two corresponding instance views. The instance views are sizing-specific realizations nested under the template; $\boldsymbol{\xi}$ constructs them deterministically without changing connectivity or the symbolic schema.

Training data. We use symbolic records produced by signal-path-based extraction (Li & Shi, 2026); Appendix A further describes their construction. We learn from a template-domain dataset $\mathcal{D}_T$ of graph–symbolic pairs and a sizing-domain dataset $\mathcal{D}_I$ of instantiated pairs. The training unit is a graph–symbolic pair within one domain, so template and instance pairs can be sampled independently. Multiple sizing records may share a topology. Each domain has its own topology partition, with disjoint train, validation, and test topologies within that domain.

Representation target. For domain $a \in \{T, I\}$, modality-specific encoders map an observed pair to a shared unit sphere:

$$\mathbf{z}_G^a = E_G(G^a), \qquad \mathbf{z}_S^a = E_S(S^a), \qquad \|\mathbf{z}_G^a\|_2 = \|\mathbf{z}_S^a\|_2 = 1. \tag{3}$$

Within each modality, the encoder shares weights across domains. For a graph query, the target is to rank its paired symbolic record above other records; the reverse direction ranks the paired graph for a symbolic query. A symmetric InfoNCE objective trains both retrieval directions. Retrieval evaluates pair correspondence, while downstream tasks evaluate the graph encoder as an initialization for annotation and sizing-performance classification.

4 METHOD

4.1 OVERVIEW

ANALOGQUAD connects template and sized-instance views of graphs and symbolic records through three components. Sizing injection constructs paired instance inputs (Section 4.2). A graph encoder and a hierarchical symbolic encoder produce global embeddings, sharing weights across domains within each modality (Section 4.3). Symmetric contrastive learning aligns graph–symbolic pairs at both levels to capture structural correspondence and sizing-dependent distinctions (Section 4.4).

4.2 CONNECTING TEMPLATES AND INSTANCES THROUGH SIZING INJECTION

We connect template views (G_t^T, S_t^T) to their sized-instance views $(G_{t,\boldsymbol{\xi}}^I, S_{t,\boldsymbol{\xi}}^I)$ by applying the same named sizing assignment to both modalities. Sizing enters graph attributes and symbolic expressions before encoding, as defined in Equation (2). Each assignment produces a paired realization of the template's connectivity and symbolic schema.

Graph attributes. As shown in Figure 3, template and sized graphs use the same node-feature format. Each node has base features describing device type, model type, and port count, followed by a sizing-value vector $\mathbf{p}_v$ and a presence-mask vector $\mathbf{m}_v$:

$$\mathbf{x}_v^T = [\mathbf{x}_v^{\text{base}}; \mathbf{0}_v; \mathbf{0}_v], \qquad \mathbf{x}_v^I = [\mathbf{x}_v^{\text{base}}; \mathbf{p}_v; \mathbf{m}_v]. \tag{4}$$

Here, $\mathbf{0}_v$ denotes an all-zero vector with the same shape as $\mathbf{p}_v$ and $\mathbf{m}_v$. For sized inputs, parameters are matched to device nodes by name. Each available positive value is converted to a fixed unit for its type and transformed with $\log_{10}$; its mask is set to one. Missing or inapplicable values and masks remain zero. Template inputs set both vectors to zero to indicate that numerical sizing is unspecified. Both domains therefore have the same input feature dimension, with connectivity unchanged by sizing injection.

Symbolic instantiation. The symbolic record retains the circuit parameters produced by the extraction procedure in Appendix A. We instantiate it by mapping record symbols to capacitance, resistance, and transconductance g_m values supplied with the instance, using nominal macromodel values for the remaining parameters. Unit conversion, substitution, and simplification yield numerical pole and zero expressions, which also form the transfer-function expression. The resulting $S_{t,\boldsymbol{\xi}}^I$ retains the schema and expression hierarchy of S_t^T. The device-to-symbol mapping supplies g_m directly from the instance record.

4.3 ENCODING GRAPHS AND SYMBOLIC RECORDS

Graph encoder. As shown in Figure 4, three message-passing layers propagate node features along device–net connections; mean pooling and projection yield a normalized 128-dimensional graph embedding. We use GCN (Kipf & Welling, 2017), four-head GAT (Veličković et al., 2018), edge-aware GINE (Xu et al., 2019; Hu et al., 2020), or GraphSAGE (SAGE) (Hamilton et al., 2017), using edge attributes where supported. Sizing-dependent differences enter through node features. Appendix D.8 and Table 3 specify the operators and architecture settings.

To characterize the supervision supplied by sizing inputs, decompose the graph encoder's affine input projection into base-feature and sizing-channel weights:

$$\mathbf{a}_v = W_b \mathbf{x}_v^{\text{base}} + W_s \mathbf{z}_v + \mathbf{b}, \qquad \mathbf{z}_v = [\mathbf{p}_v; \mathbf{m}_v]. \tag{5}$$

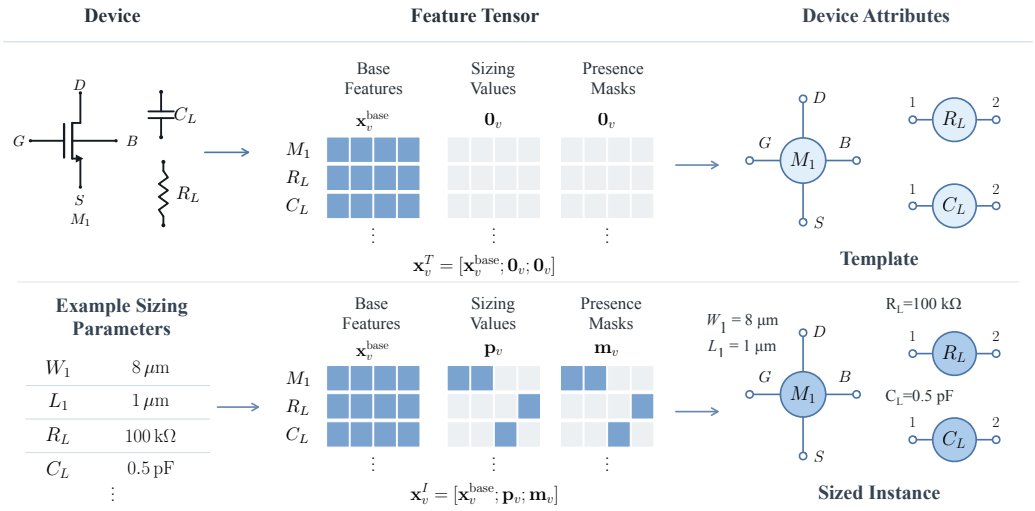


Figure 3: Graph-side sizing injection. A sizing assignment populates the sizing-value and presence-mask channels while preserving circuit connectivity; template inputs retain the same dimensional format with zeroed sizing channels.

Proposition 4.1 (Sizing-channel supervision)**.** *Fix all other parameters and stochastic encoder choices, and assume that W_s enters the model only through Equation* (5)*. On any template-only paired batch $\mathcal{B}_T$, the contrastive data loss $\mathcal{L}_{\mathcal{B}_T}$ defined by Equation* (8) *is independent of W_s and satisfies $\nabla_{W_s}\mathcal{L}_{\mathcal{B}_T} = \mathbf{0}$.*

Sized inputs can provide data gradients to these otherwise unsupervised input weights. The proof and the gradient expression for sizing-only and mixed-domain batches are given in Appendix B.1.

Hierarchical symbolic encoder. As shown in Figure 4, the symbolic encoder builds a global embedding from the structure of a supplied record. The parser identifies entries $S = \{R_j\}_{j=1}^{J}$, semantic types κ_j (e.g., pole or zero; Table 4), and pole/zero locations λ_j. Within an expression entry, each rational factor has numerator and denominator branches $b \in \{N, D\}$; descriptions, fallback content, and other fields are non-rational entries. We group rational branches by complete additive terms. A non-rational entry provides one grouped branch input, leaving the other branch empty. Groups are packed into chunks $\{C_{j,b,m}\}_{m=1}^{M_{j,b}}$ of at most 254 body tokens; only a term or non-rational entry exceeding this limit is split internally. This preserves every parser-emitted token within BERT's 256-token local input budget, including special tokens (Appendix B.3).

A shared local BERT encodes each chunk. Mean pooling summarizes each branch, and an entry MLP combines the branch summaries with a learnable characteristic embedding $\mathbf{e}^{\text{char}}_{\kappa_j}$, indexed by the entry's semantic type, and a location embedding $\mathbf{e}^{\text{loc}}_{\lambda_j}$:

$$\begin{aligned} \mathbf{q}_{j,b,m} &= \text{BERT}([\texttt{CLS}], C_{j,b,m}, [\texttt{SEP}])_{\texttt{CLS}}, \\ \mathbf{r}_{j,b} &= \frac{1}{M_{j,b}} \sum_{m=1}^{M_{j,b}} \mathbf{q}_{j,b,m}, \\ \mathbf{u}_j &= \text{MLP}([\mathbf{r}_{j,N}; \mathbf{r}_{j,D}; \mathbf{e}^{\text{char}}_{\kappa_j}; \mathbf{e}^{\text{loc}}_{\lambda_j}]). \end{aligned} \tag{6}$$

An empty branch uses a learned placeholder vector. A two-layer, position-free Transformer contextualizes the entry tokens, followed by attention pooling.

To retain entry-type multiplicities, we add a transformed count vector before projection:

$$\begin{aligned} (\widetilde{\mathbf{u}}_1, \ldots, \widetilde{\mathbf{u}}_J) &= \text{Transformer}(\mathbf{u}_1, \ldots, \mathbf{u}_J), \\ E_S(S) &= \text{norm}\big(p_S\big(\text{AttnPool}(\{\widetilde{\mathbf{u}}_j\}_{j=1}^{J}) + \rho(\log(1+\mathbf{c}))\big)\big)\,. \end{aligned} \tag{7}$$

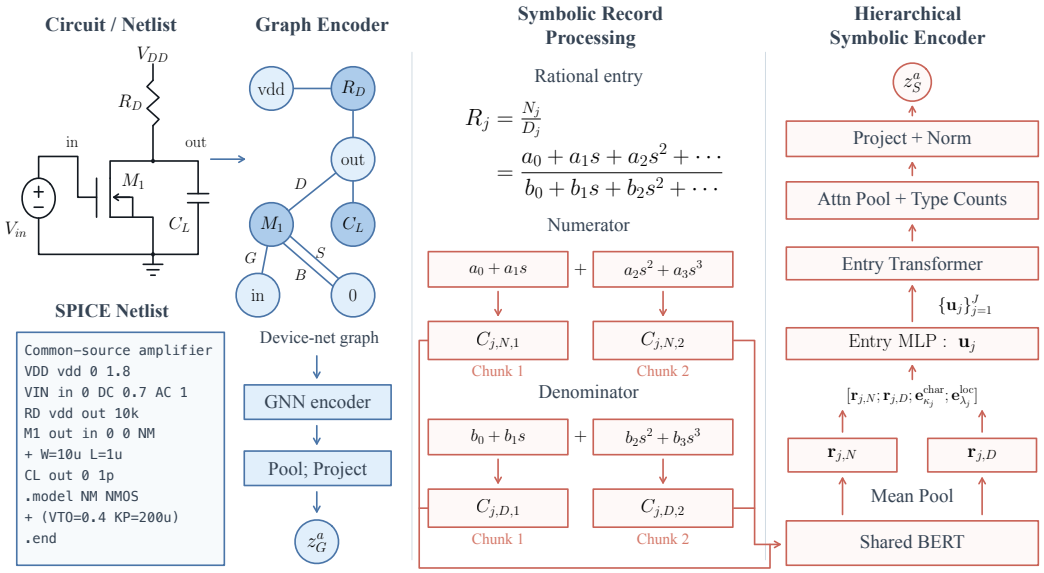


Figure 4: Graph neighborhood aggregation and hierarchical symbolic encoding. The symbolic hierarchy illustrates a rational entry, with metadata and type counts retained during aggregation.

Here, $\mathrm{norm}(\mathbf{x}) = \mathbf{x}/\|\mathbf{x}\|_2$, p_S is a projection head, $\mathbf{c}$ counts entries by type, and ρ maps these counts to the pooled representation's dimension. Training aligns the resulting record embedding with its paired graph embedding.

4.4 Aligning Template and Instance Views across Domains

Template and instance views carry complementary information about circuit structure and its sizing realizations. We align each graph with its symbolic record within the same domain, using shared encoders to connect learning across levels. As described in Section 3, template and instance pairs can be sampled independently.

Symmetric alignment. For a batch $\mathcal{B} = \{(G_i, S_i)\}_{i=1}^{B}$, write $X = (E_G(G_i))_{i=1}^{B}$ and $Y = (E_S(S_i))_{i=1}^{B}$. The score $Z_{ij} = \mathbf{x}_i^\top \mathbf{y}_j/\tau$ uses a learned positive temperature initialized to 0.07. The shared objective is

$$\ell_i^{X\to Y} = -\log \frac{\exp Z_{ii}}{\sum_{j=1}^{B} \exp Z_{ij}}, \qquad \ell_i^{Y\to X} = -\log \frac{\exp Z_{ii}}{\sum_{j=1}^{B} \exp Z_{ji}},$$
$$\mathcal{L}_{\mathrm{AQ}}(\mathcal{B};\Phi) = \mathcal{L}_{\mathrm{bi}}(X,Y) = \frac{1}{2B}\sum_{i=1}^{B}(\ell_i^{X\to Y} + \ell_i^{Y\to X}). \tag{8}$$

Each matched template pair (G_t^T, S_t^T) or instance pair $(G_{t,\boldsymbol{\xi}}^I, S_{t,\boldsymbol{\xi}}^I)$ forms a positive; all nonmatching records in the batch enter the denominators. Row and column normalization give equal weight to the two retrieval directions. Lemma B.1 in Appendix B.2 establishes $\mathrm{BiMRR}(\mathcal{B}) \geq \exp[-\mathcal{L}_{\mathrm{AQ}}(\mathcal{B};\Phi)]$ when the loss and ranks use the same score matrix and candidate set.

Training. Template and sized-instance pairs use the same contrastive objective. We train on template pairs followed by sizing adaptation, or on mixed-domain batches from initialization. Sizing adaptation inherits both encoders, projection heads, and temperature from the selected template checkpoint, resets the optimizer and learning-rate schedule, and updates the full model. Each mixed batch contains 20 template pairs and 12 sizing pairs. We compute the contrastive objective on their combined score matrix, using nonmatching records from either domain as negatives. Sizing pairs supply instance-specific values and masks to the otherwise unsupervised input weights identified in Proposition 4.1. Appendices D.2 and D.4 give the complete training schedules and checkpoint-selection rules.

5 EXPERIMENTS

We evaluate ANALOGQUAD on cross-modal retrieval and downstream tasks. Retrieval experiments examine graph–symbolic correspondence in the template domain and adaptation to the sizing domain. Downstream experiments assess the transfer of pretrained graph encoders to functional annotation and sizing-performance prediction.

5.1 EXPERIMENTAL SETUP

Datasets. Our paired data describe operational-amplifier circuits and their small-signal symbolic records. The template domain contains 8,115 graph–symbolic pairs across 1,616 topologies; the sizing domain contains 4,827 pairs across 17 topologies with specified device sizes. Downstream tasks are GANA functional annotation (Kunal et al., 2020), with 1,027 graphs and 33,781 nodes, and sizing prediction on the 4,827 sized circuits. Sizing prediction classifies whether a circuit belongs to the better-performing half of its topology for each of four performance measures; its targets are performance labels, not device dimensions. Device sizes enter graph attributes and symbolic substitutions as described in Section 4.2. Appendix D gives the splits and task definitions.

Methods. We evaluate ANALOGQUAD with the CLIP-style symmetric contrastive objective (Radford et al., 2021) and training schedules described in Section 4.4. In comparisons between training configurations, AnalogQuad-S1 and AnalogQuad-S2 denote the template and sizing-adapted checkpoints, and AnalogQuad (joint) denotes training with mixed-domain batches. We compare these configurations with the SigLIP-based multimodal baseline MM-S1/MM-S2 (Zhai et al., 2023). This retrieval comparison uses the same GINE graph architecture for all methods. Sequential retrieval and downstream transfer additionally cover GCN, GAT, and SAGE. Downstream initializations are Scratch, graph-only GraphCL-S1/GraphCL-S2, and AnalogQuad-S1/AnalogQuad-S2; S1/S2 identify the corresponding pretraining domains. Joint and MM checkpoints are evaluated only for retrieval. The comparison evaluates complete training configurations, whose learning rates and budgets differ as listed in Table 5.

Evaluation. Exact-pair retrieval ranks the matched target in both graph-to-symbolic and symbolic-to-graph directions against each domain's full test pool: 815 template pairs or 458 sizing pairs. We report Recall@K and mean reciprocal rank (MRR), averaging the two directions for bidirectional summaries. Each pool contains multiple topologies and supports exact-pair retrieval across the full domain test split. Appendix D.1 specifies ranking and tie handling. Retrieval uses one pretraining seed; downstream results are five-seed means with sample standard deviations. The main downstream comparison uses epoch 20, with learning curves at additional endpoints. Percentage improvements are relative to the stated baseline and computed from the corresponding mean scores. GraphCL and ANALOGQUAD use matched downstream training budgets; their pretraining compute differs. Hardware and software versions appear in Appendix D.7.

5.2 CROSS-MODAL RETRIEVAL IN THE TEMPLATE DOMAIN

As shown in Figure 12, template pretraining retrieves more than 99% of paired targets within the top ten candidates for every backbone. This coverage demonstrates graph–symbolic correspondence across all four architectures. Figure 11 summarizes mean bidirectional MRR across backbones.

As shown in Figure 5, AnalogQuad-S1 improves mean bidirectional MRR by 19.2% over MM-S1, with directional gains of 21.4% for graph-to-symbolic retrieval and 17.1% for symbolic-to-graph retrieval. AnalogQuad (joint) increases the mean improvement over MM-S1 to 24.2%, giving a further 4.2% gain over AnalogQuad-S1. Table 1 provides the underlying MRR values for both domains.

5.3 RETRIEVAL TRANSFER TO THE SIZING DOMAIN

As shown in Figure 11, sizing adaptation improves mean bidirectional MRR over direct template-checkpoint transfer by approximately 982%, 383%, 694%, and 513% for GCN, GAT, GINE, and SAGE, respectively. These gains are measured on the same sizing test pool for both checkpoints and

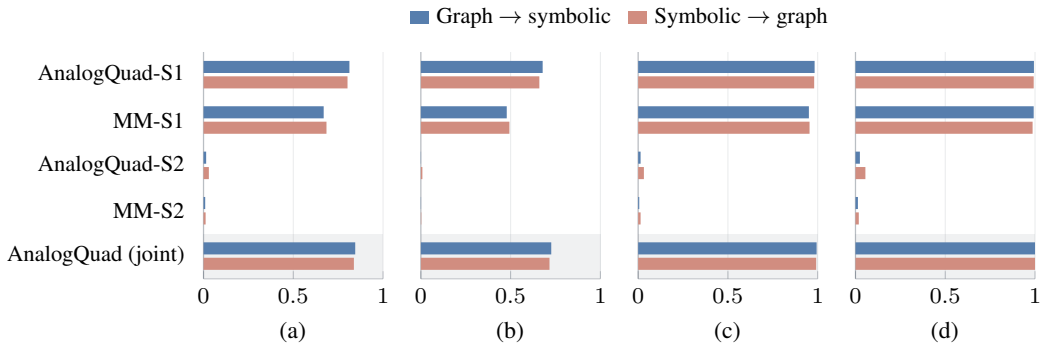


Figure 5: Template retrieval over 815 test candidates in both directions: (a) mean reciprocal rank (MRR); (b) R@1; (c) R@5; (d) R@10.

Table 1: Exact-pair retrieval in both domains, measured by MRR (G: graph; S: symbolic). Bold marks the best column; n/a denotes results unreported in this comparison.

Method	**Template** ($N = 815$)		**Sizing** ($N = 458$)	
	$G \to S$	$S \to G$	$G \to S$	$S \to G$
AnalogQuad-S1	0.8127	0.8021	n/a	n/a
MM-S1	0.6693	0.6849	n/a	n/a
AnalogQuad-S2	0.0147	0.0297	0.1065	0.1007
MM-S2	0.0094	0.0117	0.0836	0.0808
ANALOGQUAD (joint)	**0.8450**	**0.8375**	**0.1135**	**0.1123**

Table 2: Sizing retrieval after sequential S1→S2 training with hierarchical and flat symbolic encoders. The final column averages MRR across directions.

Symbolic encoder	$G \to S$	$S \to G$	Mean
Hierarchical (AnalogQuad-S2)	0.1065	0.1007	0.1036
w/o chunking (flat BERT)	0.1041	0.1149	0.1095

show the benefit of adapting to instance-level pairs across all four backbones. Figure 13 reports both retrieval directions.

As shown in Figure 6g, AnalogQuad-S2 improves mean bidirectional sizing MRR by 26.1% over MM-S2. AnalogQuad (joint) extends this improvement to 37.4%, with gains of 35.8% and 39.0% in the graph-to-symbolic and symbolic-to-graph directions, respectively. It also exceeds AnalogQuad-S2 by 9.0% in mean sizing MRR. As shown in Table 1, AnalogQuad (joint) achieves the highest MRR in both directions on both domains, preserving template retrieval while improving instance-level alignment.

5.4 COMPARING HIERARCHICAL AND FLAT SYMBOLIC ENCODERS

As shown in Table 2, flat BERT yields 5.7% higher mean sizing MRR than the hierarchical encoder after sequential S1→S2 training. The direction-specific comparison shows a 2.3% advantage for the hierarchical encoder over flat BERT in graph-to-symbolic retrieval, while flat BERT improves symbolic-to-graph MRR by 14.1% over the hierarchical encoder.

The flat baseline serializes each record into a single 512-token BERT input with CLS pooling; both runs use GINE and the same test pool. Because formatting, aggregation, and S1 training history differ, this comparison does not isolate chunking. Appendix D.6 gives checkpoint and evaluation details.

5.5 DOWNSTREAM TRANSFER

We transfer the pretrained graph encoder to functional annotation and sizing prediction. Figure 6 shows annotation curves for all four backbones and sizing curves for GCN and GAT; Figure 8 supplies

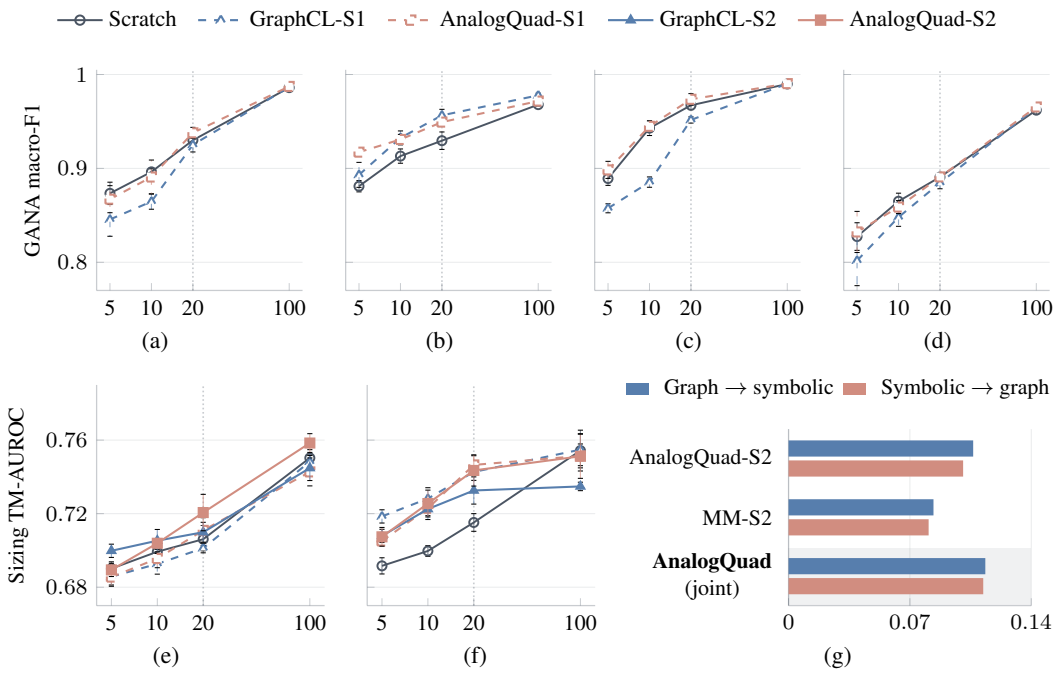


Figure 6: Downstream transfer and sizing retrieval. GANA test macro-F1: (a) GCN, (b) GAT, (c) GINE, and (d) SAGE. Sizing-prediction test topology-macro AUROC: (e) GCN and (f) GAT. Learning curves use logarithmic epoch axes and show five-seed means with sample-standard-deviation error bars. (g) Sizing retrieval reports MRR over 458 test candidates.

the complete trajectories. Dashed lines with open markers denote S1, solid lines with filled markers denote S2, and dotted guides mark epoch 20.

Functional annotation. As shown in Figure 6, AnalogQuad-S1 improves mean macro-F1 over Scratch at epoch 20 by 0.85%, 2.13%, and 0.66% for GCN, GAT, and GINE, respectively, and closely matches Scratch for SAGE. AnalogQuad-S1 leads the GCN and GINE comparisons, while GraphCL-S1 leads among the GAT initializations. These results demonstrate transfer of template-pretrained graph representations to node-level functional annotation. The gaps between initializations narrow by epoch 100.

Sizing prediction. Topology-macro AUROC averages valid per-measure AUROCs within each topology and then equally across topologies; measures with only one class and topologies without valid measures are excluded. As shown in Figures 6 and 8, both AnalogQuad checkpoints improve mean topology-macro AUROC over Scratch at epoch 20 across all four backbones. AnalogQuad-S2 yields relative gains of 2.05%, 3.96%, 0.45%, and 0.91% for GCN, GAT, GINE, and SAGE, respectively. AnalogQuad-S2 leads the GCN comparison, and AnalogQuad-S1 leads for GAT with a 4.38% improvement over Scratch. GraphCL-S2 leads the GINE and SAGE comparisons.

6 Conclusion

AnalogQuad connects analog circuit templates and sized instances through sizing injection, hierarchical symbolic encoding, and graph–symbolic alignment. For retrieval, AnalogQuad improves mean bidirectional MRR by 24.2% on templates and 37.4% on sized instances relative to MM-S1 and MM-S2, respectively. For downstream tasks at 20 fine-tuning epochs, AnalogQuad improves mean topology-macro AUROC by up to 4.4% and mean functional-annotation macro-F1 by up to 2.1% over training from scratch. All percentages denote relative improvements; downstream comparisons use five-seed means. These results demonstrate the value of aligning reusable circuit structure with parameter-dependent symbolic behavior for retrieval and downstream learning.

AI USE STATEMENT

The authors led the research, including problem formulation, method design, experimental decisions, and assessment of the findings. Generative AI tools assisted with code development and debugging, experiment execution and result analysis, manuscript drafting and editing, and initial figure drafts. The authors reviewed and verified the AI-assisted content and outputs and take full responsibility for the final text, methods, claims, and experimental results.

REPRODUCIBILITY STATEMENT

The symbolic-record extraction procedure and dataset splits are described in Appendix A and Table 3. The input representations, encoders, and contrastive objective are specified in Section 4 and Appendix D.8; training configurations and checkpoint-selection criteria are detailed in Appendices D.2 and D.4. Retrieval evaluation, sizing-prediction metrics, and the hardware and software environment are documented in Appendices D.1, D.5 and D.7.

ETHICS STATEMENT

This work uses circuit graphs, symbolic circuit records, and engineering data derived from circuit simulation. It does not involve human subjects or personal data.

REFERENCES

Bryan Amador and Richard Zanibbi. Math formula graph retrieval using contrastive learning over visual and semantic embeddings. In *Proc. ICTIR*, pp. 230–237, 2025.

Walter Daems, Georges Gielen, and Willy Sansen. A fitting approach to generate symbolic expressions for linear and nonlinear analog circuit performance characteristics. In *Proc. DATE*, pp. 268–273, 2002.

Zehao Dong, Weidong Cao, Muhan Zhang, Dacheng Tao, Yixin Chen, and Xuan Zhang. CktGNN: Circuit graph neural network for electronic design automation. In *Proc. ICLR*, 2023.

Wenji Fang, Shang Liu, Jing Wang, and Zhiyao Xie. CircuitFusion: Multimodal circuit representation learning for agile chip design. In *Proc. ICLR*, 2025.

Georges Gielen, Piet Wambacq, and Willy M. C. Sansen. Symbolic analysis methods and applications for analog circuits: A tutorial overview. *Proc. IEEE*, 82(2):287–304, 1994.

Georges G. E. Gielen, Herman C. C. Walscharts, and Willy M. C. Sansen. ISAAC: A symbolic simulator for analog integrated circuits. *IEEE JSSC*, 24(6):1587–1597, 1989.

Paul R. Gray, Paul J. Hurst, Stephen H. Lewis, and Robert G. Meyer. *Analysis and Design of Analog Integrated Circuits*. John Wiley & Sons, 2009.

Kourosh Hakhamaneshi, Marcel Nassar, Mariano Phielipp, Pieter Abbeel, and Vladimir Stojanović. Pretraining graph neural networks for few-shot analog circuit modeling and design. *arXiv preprint arXiv:2203.15913*, 2022.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proc. NeurIPS*, volume 30, 2017.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *Proc. ICLR*, 2020.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proc. ICLR*, 2017.

Kishor Kunal, Tonmoy Dhar, Meghna Madhusudan, Jitesh Poojary, Arvind Sharma, Wenbin Xu, Steven M. Burns, Jiang Hu, Ramesh Harjani, and Sachin S. Sapatnekar. GANA: Graph convolutional network based automated netlist annotation for analog circuits. In *Proc. DATE*, 2020.

Sungyoung Lee, Ziyi Wang, Seunggeun Kim, Taekyun Lee, Yao Lai, and David Z. Pan. DICE: Device-level integrated circuits encoder with graph contrastive pretraining. *arXiv preprint arXiv:2502.08949*, 2025.

Mingzhen Li and Guoyong Shi. A signal-path recognition approach to multistage Op-Amp pole-zero extraction with applications. *IEEE TCAD*, 45(7):3045–3058, 2026.

Yi Liu, Hongji Zhang, Lei Chen, Mingxuan Yuan, and Qiang Xu. UniRTL: Unifying code and graph for robust RTL representation learning. *arXiv preprint arXiv:2605.31040*, 2026.

Asal Mehradfar, Xuzhe Zhao, Yilun Huang, Emir Ceyani, Yankai Yang, Shihao Han, Hamidreza Aghasi, and Salman Avestimehr. FALCON: An ML framework for fully automated layout-constrained analog circuit design. *arXiv preprint arXiv:2505.21923*, 2025.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proc. ICML*, pp. 8748–8763, 2021.

Willy M. C. Sansen. *Analog Design Essentials*, volume 859. Springer, 2007.

Xavier Suau, Federico Danieli, T. Anderson Keller, Arno Blaas, Chen Huang, Jason Ramapuram, Dan Busbridge, and Luca Zappella. DUET: 2D structured and approximately equivariant representations. In *Proc. ICML*, volume 202 of *PMLR*, pp. 32749–32769, 2023.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *Proc. ICLR*, 2018.

Yihan Wang, Lei Li, Yao Lai, Jing Wang, and Yan Lu. AnalogRetriever: Learning cross-modal representations for analog circuit retrieval. *arXiv preprint arXiv:2604.23195*, 2026.

Chunyu Xie, Bin Wang, Fanjing Kong, Jincheng Li, Dawei Liang, Gengshen Zhang, Dawei Leng, and Yuhui Yin. FG-CLIP: Fine-grained visual and textual alignment. In *Proc. ICML*, volume 267 of *PMLR*, pp. 68777–68793, 2025.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *Proc. ICLR*, 2019.

Peng Xu, Jintao Li, Tsung-Yi Ho, Bei Yu, and Keren Zhu. Performance-driven analog layout automation: Current status and future directions. In *Proc. ASPDAC*, pp. 679–685, 2024.

Peng Xu, Yapeng Li, Tinghuan Chen, Tsung-Yi Ho, and Bei Yu. KCLNet: Electrically equivalence-oriented graph representation learning for analog circuits. In *Proc. AAAI*, 2026.

Lewei Yao, Runhui Huang, Lu Hou, Guansong Lu, Minzhe Niu, Hang Xu, Xiaodan Liang, Zhenguo Li, Xin Jiang, and Chunjing Xu. FILIP: Fine-grained interactive language-image pre-training. In *Proc. ICLR*, 2022.

Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. In *Proc. NeurIPS*, volume 33, pp. 5812–5823, 2020.

Jaemyung Yu, Jaehyun Choi, Dong-Jae Lee, HyeongGwon Hong, and Junmo Kim. Self-supervised transformation learning for equivariant representations. In *Proc. NeurIPS*, volume 37, 2024.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proc. ICCV*, pp. 11975–11986, 2023.

A SYMBOLIC SMALL-SIGNAL EXTRACTION

We extract the dataset's symbolic records using the signal-path recognition method of Li & Shi (2026). Figure 7 traces the OTA in Figure 2 through graph abstraction, small-signal modeling, and symbolic analysis. These records provide the symbolic inputs to ANALOGQUAD.

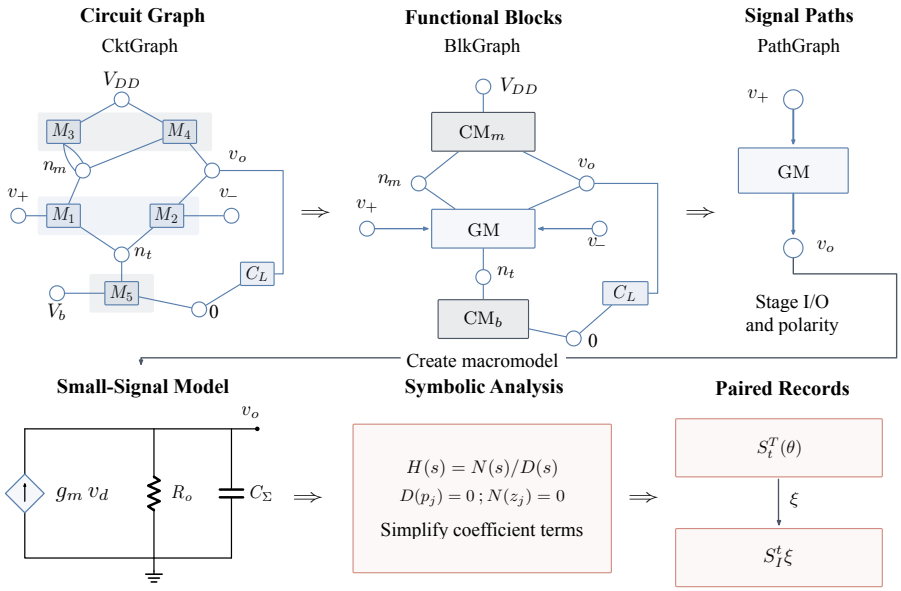


Figure 7: Signal-path-based symbolic extraction, illustrated for the OTA in Figure 2. Graph condensation yields a small-signal model, whose expressions form templates and sized records.

A.1 RECOGNIZING SIGNAL PATHS FROM NETLISTS

The extractor converts a netlist into a bipartite *CktGraph*, with device and net vertices connected by terminal-labeled edges. Bias paths and gate connections identify transconductance (GM), current-mirror (CM), and cascode (CAS) blocks. Contracting each block yields a *BlkGraph* that preserves inter-block connections and key nets entering GM blocks.

Tracing signal propagation between key nets identifies the primary amplification path and feedforward or feedback compensation. Pruning connections outside these paths yields a *PathGraph* that records stage boundaries, signal polarity, and compensation. In the illustrated OTA, the differential pair becomes GM, the mirror load becomes CM_m, and the bias device becomes CM_b. Multistage circuits preserve the connections between successive stages and their compensation paths.

A.2 DERIVING SMALL-SIGNAL SYMBOLIC EXPRESSIONS

Each gain stage is represented by an effective transconductance $G_{m,k}$ and output impedance $R_{o,k} \parallel C_{o,k}$. Passive compensation elements retain their circuit connections; active compensation paths contribute controlled sources and loading. Stage transconductance is related to the device-level g_m values of the recognized GM block. Output resistance and capacitance are obtained from a first-order approximation to the stage output admittance, $Y_{o,k}(s) \approx R_{o,k}^{-1} + sC_{o,k}$, with the stage inputs nulled. These quantities describe a local small-signal model around the operating point.

The graph-pair decision diagram (GPDD) engine derives a rational transfer function from this linear macromodel. With $\boldsymbol{\theta}$ collecting its circuit parameters,

$$H(s;\boldsymbol{\theta}) = \frac{N(s;\boldsymbol{\theta})}{D(s;\boldsymbol{\theta})}, \qquad D(p_j;\boldsymbol{\theta}) = 0, \qquad N(z_j;\boldsymbol{\theta}) = 0, \tag{9}$$

where p_j and z_j denote poles and zeros after common-factor cancellation. The extractor simplifies polynomial coefficients using technology-dependent magnitude orders of their product terms (Li

& Shi, 2026). Separated roots admit compact approximations; nearby roots may require quadratic factors to retain conjugate pairs.

For the illustrative OTA, a single-stage approximation makes the connection between the circuit and its expression explicit:

$$H(s) = \frac{G_m R_o}{1 + sR_o C_\Sigma}, \qquad A_0 = G_m R_o, \qquad p = -\frac{1}{R_o C_\Sigma}. \tag{10}$$

Here, $v_d = v_+ - v_-$, G_m includes the signal polarity, and C_Σ includes the output-node capacitance and external load. This first-order example has no finite zero; higher-order and compensated models can contain additional poles and zeros.

A.3 Constructing Template and Instance Records

The extracted expressions and descriptions form S_t^T. The parser retains available pole, zero, transfer-function, description, and fallback entries (Table 4). Numerator/denominator branches, entry types, and pole/zero locations supply the symbolic encoder's hierarchy and metadata.

For a sizing assignment $\boldsymbol{\xi}$, device-to-symbol mappings associate record parameters with available capacitance, resistance, and g_m values. Unit conversion, substitution, and simplification produce $S_{t,\boldsymbol{\xi}}^I$, using nominal macromodel values for absent parameters. Both levels retain the same schema: extraction supplies the analytic structure, sizing instantiates its parameters (Section 4.2), and the symbolic encoder encodes the resulting record (Section 4.3).

B Proofs for Input Representation and Training

We prove the sizing-channel supervision property stated in Proposition 4.1 and derive sizing-channel gradients for general paired batches. We then establish the retrieval bound in Lemma B.1 and token coverage for the structured chunker.

B.1 Supervision of Sizing Channels

Proof of Proposition 4.1. Let Ψ denote all parameters other than W_s, and write the contrastive data loss as $\mathcal{L}_{\mathcal{B}}(W_s, \Psi)$. Fix Ψ and the stochastic encoder choices. For every graph node in a template-only batch, Equation (4) gives $\mathbf{z}_v = \mathbf{0}$. The input projection in Equation (5) therefore reduces to

$$\mathbf{a}_v = W_b \mathbf{x}_v^{\text{base}} + \mathbf{b},$$

which is independent of W_s. Since W_s enters the model only through this projection, replacing it leaves every graph embedding unchanged. Symbolic embeddings depend only on the fixed parameters Ψ. All pairwise scores and hence the complete contrastive data loss are therefore unchanged for any W_s, W_s', giving

$$\mathcal{L}_{\mathcal{B}_T}(W_s, \Psi) = \mathcal{L}_{\mathcal{B}_T}(W_s', \Psi), \qquad \nabla_{W_s} \mathcal{L}_{\mathcal{B}_T} = \mathbf{0}. \tag{11}$$

□

For the general gradient expression, consider any paired batch $\mathcal{B}$, which may contain template graphs, sized graphs, or both. At differentiable points, write $\boldsymbol{\delta}_v = \partial \mathcal{L}_{\mathcal{B}} / \partial \mathbf{a}_v$. Each occurrence of a graph node contributes an input activation $\mathbf{a}_v$, and a perturbation dW_s changes it by $d\mathbf{a}_v = (dW_s)\mathbf{z}_v$. The chain rule gives

$$\begin{aligned} d\mathcal{L}_{\mathcal{B}} &= \sum_{(G,S)\in\mathcal{B}} \sum_{v \in V(G)} \boldsymbol{\delta}_v^\top (dW_s) \mathbf{z}_v \\ &= \left\langle \sum_{(G,S)\in\mathcal{B}} \sum_{v \in V(G)} \boldsymbol{\delta}_v \mathbf{z}_v^\top,\ dW_s \right\rangle_F, \end{aligned}$$

where $\langle \cdot, \cdot \rangle_F$ is the Frobenius inner product. Identifying the coefficient of dW_s yields

$$\nabla_{W_s} \mathcal{L}_{\mathcal{B}} = \sum_{(G,S)\in\mathcal{B}} \sum_{v \in V(G)} \boldsymbol{\delta}_v \mathbf{z}_v^\top. \tag{12}$$

The sums run over graph-node occurrences in the batch. The derivative $\boldsymbol{\delta}_v$ includes the effects of message passing, graph pooling, and all contrastive comparisons in both directions.

In a mixed batch, template nodes contribute zero outer products because $\mathbf{z}_v = \mathbf{0}$, while the derivatives at sized nodes incorporate comparisons with records from both domains. Thus, the same expression accounts for cross-domain negatives. The resulting gradient depends on both the sizing inputs and the backpropagated signals; weight decay is separate from the data gradients analyzed here.

B.2 Contrastive Loss and Bidirectional Retrieval

Lemma B.1 (Bidirectional retrieval bound)**.** *For a fixed batch of $B \geq 1$ matched pairs with finite scores, let $r_i^{X\to Y}$ and $r_i^{Y\to X}$ be the ranks of the paired targets among the same B candidates used in Equation* (8)*, with ties broken in any fixed order. Then*

$$\begin{aligned}\mathrm{BiMRR}(\mathcal{B}) &= \frac{1}{2B}\sum_{i=1}^{B}\left(\frac{1}{r_i^{X\to Y}} + \frac{1}{r_i^{Y\to X}}\right) \\ &\geq \exp[-\mathcal{L}_{\mathrm{AQ}}(\mathcal{B};\Phi)]\,.\end{aligned} \tag{13}$$

Proof of Lemma B.1. Fix the batch and its score matrix Z. For a graph query i, the pessimistic rank of the paired symbolic target is

$$\rho_i^{X\to Y} = 1 + \sum_{j\neq i}\mathbf{1}\{Z_{ij} \geq Z_{ii}\}.$$

It places every tied mismatch before the paired target, so any fixed tie-breaking order satisfies $r_i^{X\to Y} \leq \rho_i^{X\to Y}$. Exponentiating the directional loss in Equation (8) gives

$$\begin{aligned}e^{\ell_i^{X\to Y}} &= \sum_{j=1}^{B}\exp(Z_{ij} - Z_{ii}) \\ &\geq 1 + \sum_{j\neq i}\mathbf{1}\{Z_{ij} \geq Z_{ii}\} \geq r_i^{X\to Y}.\end{aligned}$$

The matched term contributes one, and every mismatch counted by the indicator contributes at least one. Applying the same argument to column i gives $e^{\ell_i^{Y\to X}} \geq r_i^{Y\to X}$. Taking reciprocals and averaging over all $2B$ queries yields

$$\mathrm{BiMRR}(\mathcal{B}) \geq \frac{1}{2B}\sum_{i=1}^{B}\left(e^{-\ell_i^{X\to Y}} + e^{-\ell_i^{Y\to X}}\right).$$

Jensen's inequality for the convex function $u \mapsto e^{-u}$ then yields

$$\begin{aligned}\mathrm{BiMRR}(\mathcal{B}) &\geq \exp\left[-\frac{1}{2B}\sum_{i=1}^{B}\left(\ell_i^{X\to Y} + \ell_i^{Y\to X}\right)\right] \\ &= \exp[-\mathcal{L}_{\mathrm{AQ}}(\mathcal{B};\Phi)]\,,\end{aligned}$$

which proves Equation (13). Every step uses the same score matrix for the loss and ranks, and applies equally to batches drawn from one domain or both. □

For a minibatch, the bound concerns retrieval within that minibatch. Applying it to a full test candidate pool requires computing both the loss and the ranks from that pool's score matrix.

B.3 Structured-Chunk Token Coverage

Proposition B.1 (Structured-chunk coverage)**.** *Let the ordered token occurrences emitted by the parser be partitioned into nonempty groups corresponding to additive terms or non-rational entries. For an integer budget $q \geq 1$, split every group longer than q into consecutive nonempty segments of length at most q. Greedily pack consecutive segments, starting a new chunk whenever appending the next segment would exceed q. The resulting chunk bodies contain each emitted token occurrence exactly once and have length at most q. With $q = 254$ and one* `[CLS]` *and one* `[SEP]` *per chunk, each local input has at most 256 tokens.*

Proof of Proposition B.1. Treat token occurrences as distinct by their field, record entry, branch, term, and position, even when their token values coincide. By assumption, the initial groups form a disjoint partition of these occurrences. Splitting an oversized group into consecutive segments preserves that partition and bounds every segment length by q.

Greedy packing replaces consecutive segments by their unions. At each step it either appends a segment to the current chunk within the budget, or starts a new chunk containing that segment alone. Because every segment has length at most q, both operations respect the budget. Each segment is consumed exactly once, so packing neither duplicates nor discards a token occurrence. The final chunks consequently form a disjoint partition of all parser-emitted occurrences. For the configured budget $q = 254$, adding two special tokens gives at most 256 tokens per local input. Additive terms remain intact whenever their group fits within the budget. □

C Complete Downstream Results

The downstream initializations are Scratch, GraphCL-S1, GraphCL-S2, AnalogQuad-S1, and AnalogQuad-S2. S1 denotes template pretraining; S2 denotes sizing adaptation from the same method's S1 checkpoint. All plots report five-seed means with sample standard deviations. Colors identify initialization families, and open/filled markers distinguish S1/S2. AnalogQuad (joint) and MM-S1/MM-S2 are used only in the retrieval comparison.

C.1 Functional-Annotation Trajectories

The upper row of Figure 8 expands the functional-annotation curves in Figure 6 at epochs 5, 10, 20, and 100. The three initializations share the downstream training budget within each backbone. At epoch 20, AnalogQuad-S1 leads on GCN and GINE, GraphCL-S1 leads on GAT, and AnalogQuad-S1 nearly ties Scratch on SAGE. The gaps between initializations narrow by epoch 100.

C.2 Sizing-Prediction Trajectories

The lower row of Figure 8 gives the complete four-backbone sizing trajectories, including the GINE and SAGE curves that complement the main-text panels. At epoch 20, both AnalogQuad checkpoints improve mean topology-macro AUROC over Scratch on all four backbones; GraphCL-S2 leads the GINE and SAGE comparisons.

C.3 Secondary Metrics at Fixed Endpoints

Figures 9 and 10 compare the secondary metrics at fixed downstream budgets. The backbone-wise comparisons and five-seed aggregation match the learning curves. TM-AUROC denotes topology-macro AUROC; the auxiliary rank and pooled-AUROC metrics are defined in Appendix D.5.

D Experimental Setup

D.1 Retrieval Evaluation and Directional Results

Each query has one exact paired target. Candidates comprise the full test split of the evaluated domain, rather than only samples sharing the query topology. We normalize graph and symbolic embeddings and rank candidates by decreasing dot-product similarity. Equal scores are resolved by the fixed candidate order, as in the stable sorting used by the evaluator. For normalized query $\mathbf{q}$, target $\mathbf{y}_i$, and candidates indexed in that order, the target rank is

$$r_i(\mathbf{q}, \mathcal{C}) = 1 + \sum_{j \neq i} \mathbb{1}[\mathbf{q}^\top \mathbf{y}_j > \mathbf{q}^\top \mathbf{y}_i] + \sum_{j < i} \mathbb{1}[\mathbf{q}^\top \mathbf{y}_j = \mathbf{q}^\top \mathbf{y}_i]. \tag{14}$$

Recall@K is the fraction of queries with $r_i \leq K$, and MRR is the mean of $1/r_i$ over queries. We evaluate both graph-to-symbolic and symbolic-to-graph retrieval; bidirectional means give equal weight to the two directions. The template and sizing test candidate pools contain 815 and 458 pairs, respectively.

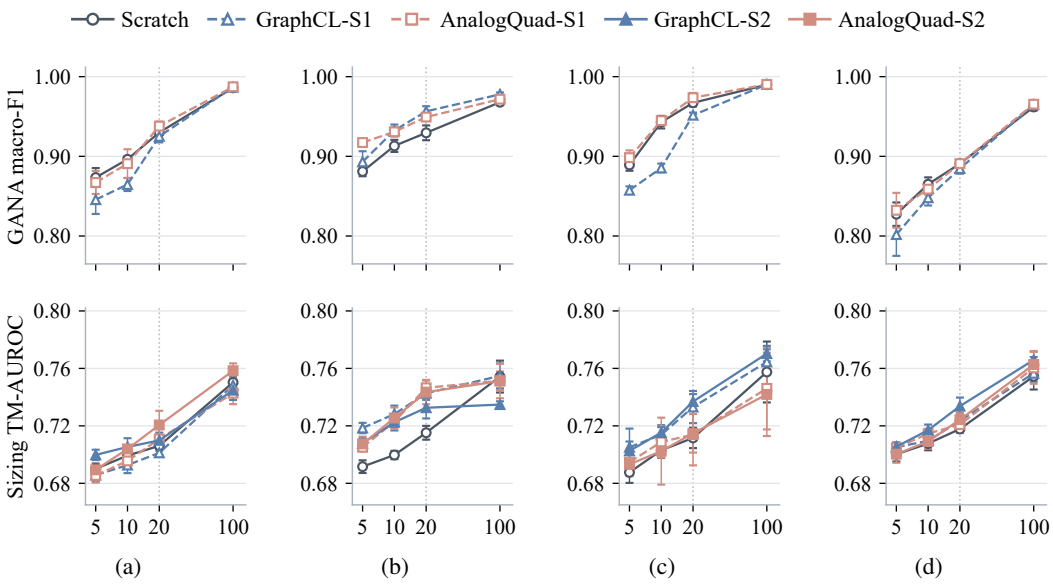


Figure 8: Downstream learning trajectories: (a) GCN; (b) GAT; (c) GINE; (d) SAGE. Upper/lower rows show GANA test macro-F1 and sizing-prediction test topology-macro AUROC, with a shared initialization legend and logarithmic epoch axes. S2 applies to sizing only. Points and error bars show five-seed means and sample standard deviations; dotted guides mark epoch 20.

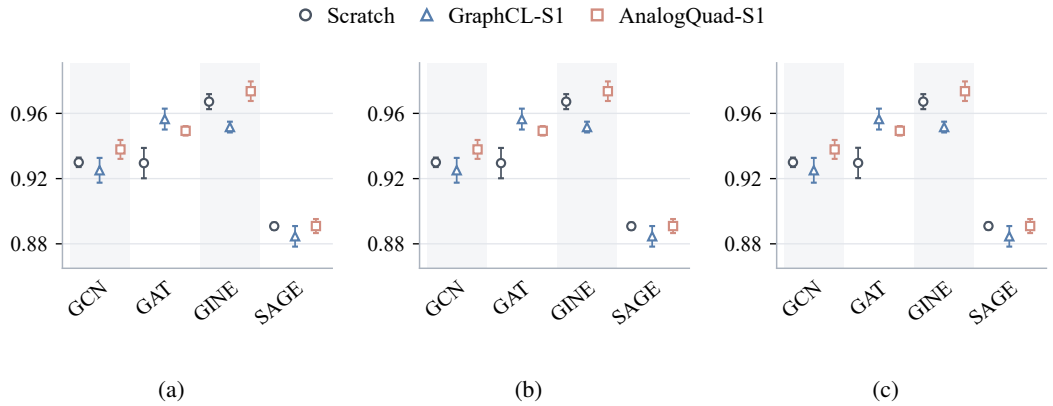


Figure 9: GANA test metrics at epoch 20: (a) accuracy; (b) macro-F1; (c) weighted-F1. Markers show five-seed means and vertical intervals show sample standard deviations. The three metrics share a score range and initialization legend; shaded columns separate backbone groups. Higher scores are better.

Figure 11 summarizes mean bidirectional MRR and sizing median ranks for AnalogQuad-S1 and AnalogQuad-S2 across four backbones. Figures 12 and 13 separate the two retrieval directions for the same checkpoints. The paired bars in Figure 11 compare checkpoints; vertical segments in Figure 13 denote checkpoint adaptation. Template and sizing results use different, explicitly labeled metric ranges.

D.2 Sequential Training and Checkpoint Selection

Let $\mathrm{Train}(\mathcal{D}, \mathcal{L}; \Phi, \mathcal{O})$ denote training on dataset $\mathcal{D}$ from model and optimizer states $\Phi, \mathcal{O}$. Template pretraining (S1), followed by sizing adaptation (S2), gives

$$\begin{aligned} \Phi_T &= \mathrm{Train}(\mathcal{D}_T, \mathcal{L}_{\mathrm{AQ}}; \Phi_0, \mathcal{O}_T), \\ \Phi_I &= \mathrm{Train}(\mathcal{D}_I, \mathcal{L}_{\mathrm{AQ}}; \Phi_T, \mathrm{ResetOpt}()). \end{aligned} \tag{15}$$

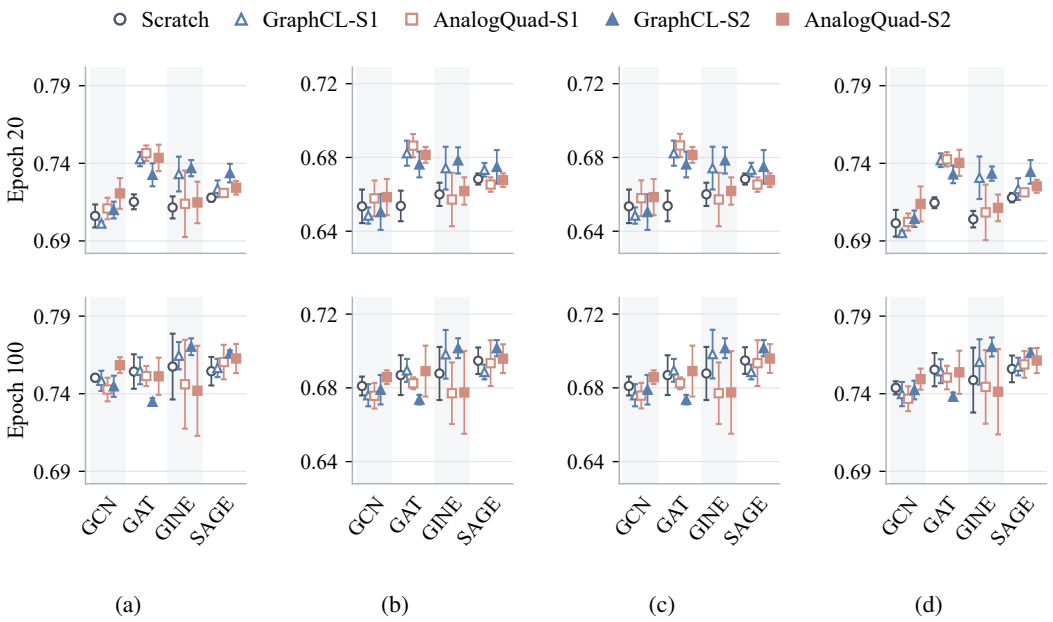


Figure 10: Sizing-prediction test metrics at epochs 20 (upper row) and 100 (lower row): (a) topology-macro AUROC; (b) rank macro-F1; (c) rank accuracy; (d) pooled AUROC. Points and vertical intervals show five-seed means and sample standard deviations, with shared scales across epochs. The shared legend identifies initializations; open/filled markers indicate S1/S2. Higher scores are better.

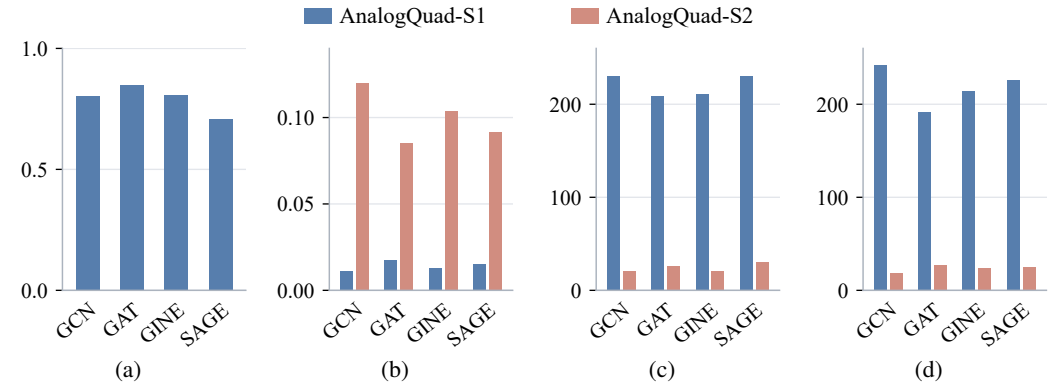


Figure 11: AnalogQuad retrieval with one shared checkpoint legend. (a) Template mean bidirectional MRR after S1, over 815 test candidates. (b) Sizing mean bidirectional MRR before and after S2, over 458 candidates. Sizing median target rank: (c) graph-to-symbolic; (d) symbolic-to-graph. Bars start at zero. Higher MRR and lower median rank are better. Template median ranks were not reported.

The initial state Φ_0 uses pretrained BERT weights and random initialization for the other modules. The selected template checkpoint Φ_T initializes sizing adaptation, which updates the full model to obtain Φ_I.

AnalogQuad-S1 checkpoints are selected by validation batch-local mean bidirectional template-retrieval accuracy.

AnalogQuad-S2 checkpoints are selected by global mean bidirectional sizing validation MRR over all 620 candidates. AnalogQuad-S2 restores the complete AnalogQuad-S1 model state for the same backbone, including both encoders, projection heads, and temperature, resets the optimizer and learning-rate schedule, and optimizes the symmetric InfoNCE objective in Equation (8) on shuffled sizing-only graph–symbolic pairs.

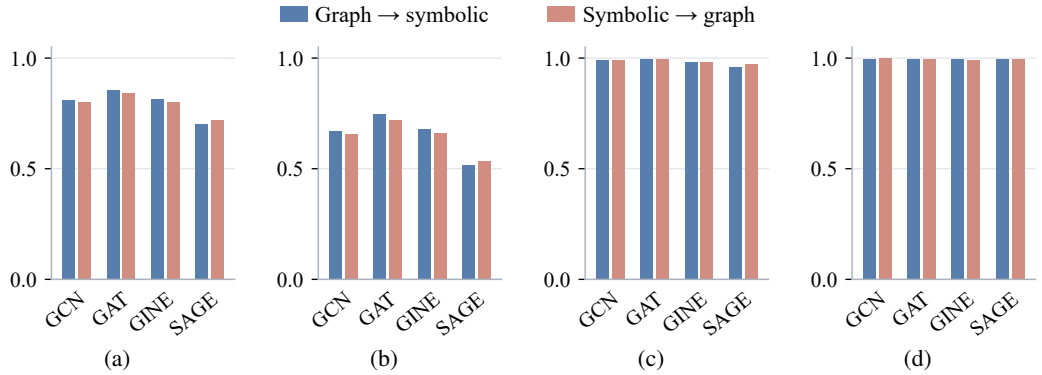


Figure 12: AnalogQuad-S1 template retrieval over 815 test candidates: (a) MRR; (b) R@1; (c) R@5; (d) R@10. Paired bars and the shared legend distinguish graph-to-symbolic and symbolic-to-graph retrieval. All panels share a zero-baseline scale; higher is better.

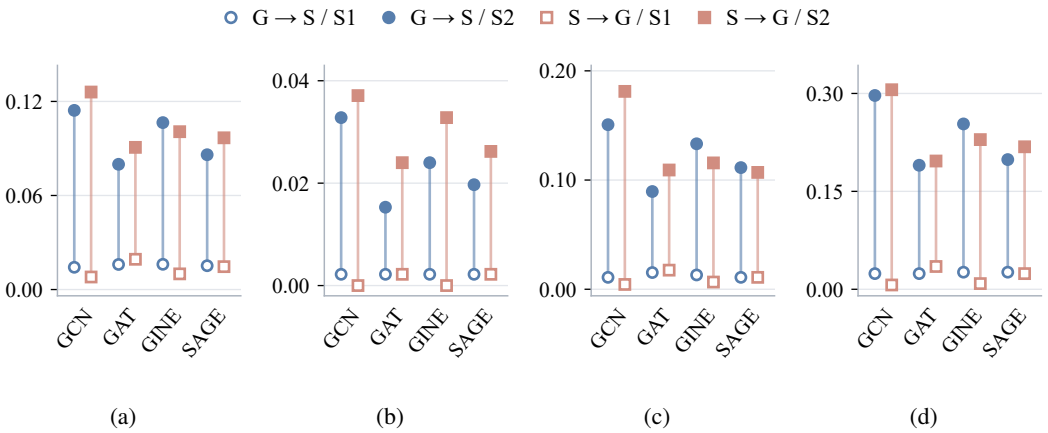


Figure 13: AnalogQuad sizing retrieval over 458 test candidates: (a) MRR; (b) R@1; (c) R@5; (d) R@10. Within each backbone and direction, segments connect AnalogQuad-S1 direct evaluation (open markers) to AnalogQuad-S2 (filled markers). The shared legend identifies directions and checkpoints. Higher is better; segments indicate adaptation, not uncertainty.

GraphCL-S1 trains on 6,490 template training graphs for 20 epochs with batch 128, learning rate 0.001, weight decay 0.0001, augmentation ratio 0.2, and temperature 0.2. GraphCL-S2 loads the full GraphCL-S1 encoder and projector, resets the optimizer, and trains for 20 more epochs on 3,749 sizing training graphs. Both use the final epoch without symbolic records or downstream labels during pretraining. Pretraining steps and encoder cost differ from ANALOGQUAD; downstream budgets are matched.

Table 3 summarizes sequential pretraining and downstream settings; the template topology split uses seed 42. Selected retrieval checkpoints appear in Figure 14.

D.3 SYMBOLIC RECORD ENTRY TYPES

The hierarchical symbolic encoder uses the five record-entry types in Table 4. An entry can contain a mathematical expression or non-rational content such as descriptive text. The parser determines κ_j from the record fields and entry names; the corresponding characteristic embedding $\mathbf{e}_{\kappa_j}^{\text{char}}$ is learned jointly with the encoders, without an additional type-classification objective. A record need not contain all five types.

Table 3: Sequential AnalogQuad pretraining and downstream settings. Dataset counts follow train/validation/test order; LR denotes learning rate.

Data and schedule	**Template (S1)**	**Sizing (S2)**
Paired samples	6,490 / 810 / 815	3,749 / 620 / 458
Topologies	1,292 / 161 / 163	13 / 2 / 2
Batch size	32	32 (shuffled)
Epoch limit	100	50
Patience (epochs)	20	10
Shared encoders and optimization		
Graph encoder	Input/hidden/output: 22/128/256; three layers; dropout 0.1; mean pooling. GCN, four-head GAT, edge-aware GINE, or SAGE.	
Symbolic encoder	BERT-base-uncased: 12 layers, 12 heads, hidden 768. Local inputs: at most 256 tokens including special tokens; micro-batch 16.	
Record hierarchy	Entry type, numerator/denominator branch, additive term, bounded chunk. Two-layer position-free entry Transformer; attention pooling and multiplicity feature.	
Sizing inputs	Graph-node W/L/C/R log values and presence masks; template sizing entries set to zero; numerical substitutions in sizing symbolic records (Section 4.2).	
Projection and loss	Hidden/output: 256/128; symmetric graph–symbolic InfoNCE in both stages; learnable temperature initialized to 0.07.	
Learning rates	GNN, symbolic-structure, and projection modules: 10^{-4}; BERT: 2×10^{-5}.	
Optimizer schedule	Weight decay 0.01; warmup ratio 0.1; optimizer and schedule reset between stages.	
Downstream evaluation		
GANA data	Graphs: 616/205/206; nodes: 20,042/6,903/6,836 (train/val./test). Signature-grouped splits with zero exact-signature overlap.	
GANA training	Hidden 128; three layers; dropout 0.1; LR 0.001; weight decay 0.0001; full-batch; macro-F1. Batch normalization recalibrated at each endpoint on label-free training inputs only.	
Sizing prediction	Batch 64; hidden/head: 128/128; head/backbone LR: 0.001/0.0003; 59 optimizer steps per epoch; topology-macro AUROC.	
Reporting	Seeds 0–4; mean $\pm$ sample standard deviation. Test evaluation reloads fixed endpoint checkpoints from complete 100-epoch trajectories.	

Table 4: Symbolic record entry types and their meanings. The parser-assigned type indexes the learned characteristic embedding.

Entry type κ_j	**Meaning**
`description`	Descriptive text supplied with the symbolic record.
`pole`	An expression for a pole, identified by the entry name.
`zero`	An expression for a zero, identified by the entry name.
`transfer`	The overall transfer-function expression $H(s)$.
`fallback`	An entry with an unrecognized type, or the raw record retained when no other entry is extracted.

D.4 Joint Training and MM Configurations

AnalogQuad (joint) and MM use the GINE graph encoder, hierarchical BERT symbolic encoder, and data splits described above, with seed 42. BERT starts from pretrained BERT-base-uncased weights; other components are randomly initialized unless inherited from a preceding checkpoint. All runs use weight decay 0.01 and a warmup ratio of 0.1. Table 5 lists the configurations.

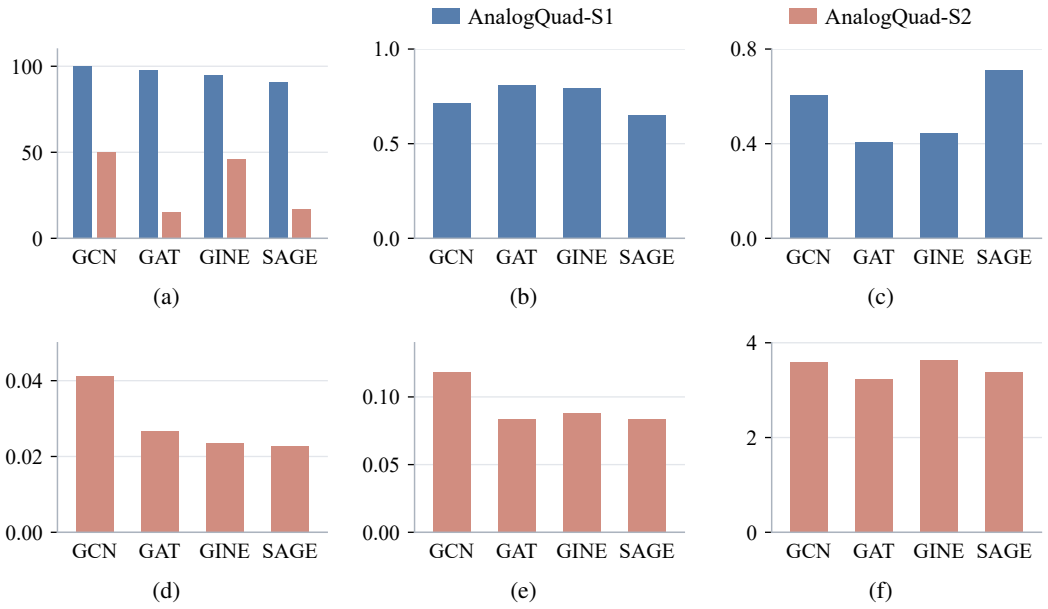


Figure 14: Selected AnalogQuad checkpoints and validation metrics: (a) epoch; (b) S1 accuracy; (c) S1 loss; (d) S2 R@1; (e) S2 MRR; (f) S2 loss. S1/S2 select by template accuracy/global sizing MRR. Bars start at zero; higher accuracy, R@1, and MRR and lower loss are better.

With the training operator defined in Appendix D.2, joint training starts from Φ_0 and optimizes mixed template and sizing batches:

$$\begin{aligned} \mathcal{B}_{\text{mix}} &= \mathcal{B}_T \cup \mathcal{B}_I, \qquad \mathcal{L}_{\text{joint}} = \mathcal{L}_{\text{AQ}}(\mathcal{B}_{\text{mix}}; \Phi), \\ \Phi_{\text{joint}} &= \text{Train}(\mathcal{D}_T \cup \mathcal{D}_I, \mathcal{L}_{\text{joint}}; \Phi_0, \mathcal{O}_{\text{joint}}). \end{aligned} \tag{16}$$

Each mixed batch contains 20 template pairs and 12 sizing pairs in a single contrastive score matrix. Nonmatching records from either domain serve as negatives, so the objective differs from the sum of two separate domain losses.

Table 5: AnalogQuad (joint) and MM pretraining configurations (GINE, seed 42). Learning rates list BERT / other encoder and projection modules; – denotes unreported settings.

Setting	**AnalogQuad (joint)**	**MM-S1**	**MM-S2**
Pretraining checkpoint	None	None	MM-S1, epoch 63
Objective	Symmetric InfoNCE	Pairwise sigmoid	Pairwise sigmoid
Global batch	32: 20 template + 12 sizing	32 template: 16 per GPU	32 sizing: one GPU
Learning rates	10^{-5} / 10^{-4}	2×10^{-5} / 10^{-4}	2×10^{-5} / 10^{-4}
Epoch limit	–	100	50
Completed epoch	100	83	43
Selected epoch	97	63	33
Patience	–	20	10

AnalogQuad (joint) trains from initialization with cross-domain negatives from each mixed batch. MM-S1 initializes its inverse temperature to 10 and learnable bias to -10. MM-S2 restores the full selected MM-S1 model, including temperature and bias, then resets the optimizer and schedule. MM-S1 selects by batch-local mean bidirectional validation accuracy; MM-S2 selects by global mean bidirectional sizing validation MRR over 620 candidates.

For joint training, checkpoint selection maximizes the smaller of the two domain validation MRRs after normalization by their fixed sequential-reference values:

$$\min \left\{ \frac{\text{MRR}_T}{0.8502192338}, \frac{\text{MRR}_I}{0.0881398616} \right\}. \tag{17}$$

The validation pools contain 810 template pairs and 620 sizing pairs. We evaluate the selected joint checkpoint independently on CPU using the same frozen test candidate pools as the baselines.

D.5 Sizing Prediction Task and Metrics

Sizing prediction is a four-label binary classification task on sized circuit graphs. The targets indicate whether a circuit belongs to the better-performing half of its topology in DC gain, gain–bandwidth product, phase margin, and power consumption. Higher values are preferred for the first three measures and lower values for power. For each topology and measure, we orient the values so that larger is better, sort them, and label values at or above the upper median as positive, including ties at the cutoff. Thus, tied values may produce more than half positive labels. These performance labels are used only for downstream supervision, not as graph–symbolic pretraining inputs.

Topology-macro AUROC first averages AUROC over valid performance measures within each test topology, then averages equally over topologies. A measure is valid when both classes are present; topologies with no valid measures are excluded. The auxiliary rank accuracy and macro-F1 use within-topology upper-median thresholding of predicted scores, with ties included, rather than a fixed probability threshold. The non-topology-macro AUROC pools test circuits before averaging over valid measures.

D.6 Flat Symbolic Encoder Comparison

The *w/o chunking (flat BERT)* variant in Section 5.4 and Table 2 encodes each serialized record with a 512-token limit and CLS pooling. Both runs use GINE and the same 458 test candidates; the hierarchical reference is AnalogQuad-S2. Flat BERT starts from a legacy S1 checkpoint whose graph input projection is expanded from 14 to 22 dimensions with zero-initialized additional columns, and its S2 checkpoint is selected by validation MRR. It is evaluated on CPU FP32, while the hierarchical model is evaluated on CUDA. This comparison covers the combined choices of symbolic formatting/content, aggregation, and S1 training history; it does not isolate the effect of chunking.

D.7 Hardware and Software

We conduct experiments on a server equipped with two 32-core Intel Xeon Gold 6430 CPUs and eight NVIDIA A100-SXM4 GPUs (40 GB each). Training uses a single GPU except for MM template pretraining, which uses two GPUs with a global batch size of 32. We use PyTorch 2.4.1 and CUDA 12.1. Graph operators are implemented with PyG 2.6.1, while symbolic encoders use Transformers 4.44.2.

D.8 Graph Encoder Details

The graph encoder processes $G = (V_D \cup V_N, E, X, A)$ with device nodes, net nodes, connectivity, node features, and optional edge attributes. Starting from $\mathbf{h}_v^{(0)} = \phi_{\text{in}}(\mathbf{x}_v)$, it aggregates neighborhood information and pools node representations:

$$\begin{aligned} \mathbf{m}_v^{(\ell)} &= \mathrm{AGG}_\ell\{\phi_\ell(\mathbf{h}_u^{(\ell)}, \mathbf{a}_{uv}) : u \in \mathcal{N}(v)\}, \\ \mathbf{h}_v^{(\ell+1)} &= \mathrm{UPDATE}_\ell(\mathbf{h}_v^{(\ell)}, \mathbf{m}_v^{(\ell)}), \\ E_G(G) &= \mathrm{norm}\left(p_G\left(\frac{1}{|V|}\sum_{v\in V}\mathbf{h}_v^{(L)}\right)\right), \end{aligned} \tag{18}$$

Here, $\mathrm{norm}(\mathbf{x}) = \mathbf{x}/\|\mathbf{x}\|_2$ and p_G is a projection head. The operators are GCN, four-head GAT, edge-aware GINE, or SAGE, with edge attributes used where supported. All backbones use three message-passing layers and project to a shared 128-dimensional space, as specified in Table 3. Sizing-dependent node features propagate along the circuit connections.