RAL-Opt: Overcoming Parametric Limitations in Post-Layout Optimization via Retrieval-Augmented Learning

Jiajie Xu[1,3], Songlin Li[1,3], Leilei Jin[2], Zhaohan Wang[1,3], Xiangfei Hu[1,3], Hao Yan[1,3,*], and Longxing Shi[1,3]

[1]Southeast University [2]CUHK [3]National Technology Innovation Center for EDA

Abstract

Post-layout optimization is essential to modern chip design. As an inherently iterative process, it continuously produces evolving design knowledge across optimization iterations. However, existing machine learning (ML)-based approaches struggle to incorporate such knowledge without costly retraining and the risk of catastrophic forgetting. As a result, the knowledge accumulated during optimization remains largely unleveraged in guiding subsequent design improvements. To address this mismatch, we propose RAL-Opt, a retrieval-augmented learning framework that enables continual integration of evolving design knowledge without retraining. RAL-Opt decomposes circuit timing graphs into reusable chunks stored in a knowledge database, retrieves relevant knowledge for unseen designs, and continuously updates the database to enable progressive improvement. A knowledge distillation strategy is developed to mitigate topological inconsistencies between database and unseen-design chunks, while a retrieval-augmented graph learning scheme integrates knowledge across designs. Experimental results on multiple ML-based post-layout optimization frameworks demonstrate that RAL-Opt not only outperforms state-of-the-art task-specific methods but also enables continual performance improvement across optimization iterations.

1 Introduction

Post-layout optimization, typically implemented through Engineering Change Orders (ECO), is a critical step for improving power, performance, and area (PPA) metrics prior to chip signoff. However, achieving desirable optimization outcomes has become increasingly challenging in advanced technology nodes. On the one hand, increasingly complex higher-order physical effects hinder the accurate estimation of the impact of ECO optimizations. On the other hand, the rapidly growing design scale dramatically enlarges the ECO solution space. Consequently, heuristic-based commercial tools typically require numerous optimization iterations to meet PPA targets [1], leading to substantial computational overhead and runtime burden.

Recently, machine learning (ML) has emerged as a promising paradigm for improving the efficiency of ECO optimization in commercial tools [2–9]. Existing approaches can be broadly classified into two main categories. The first category employs surrogate models to replace conventional evaluators [2, 3], enabling fast feedback on candidate solutions and thereby accelerating the search for desirable optimization outcomes.

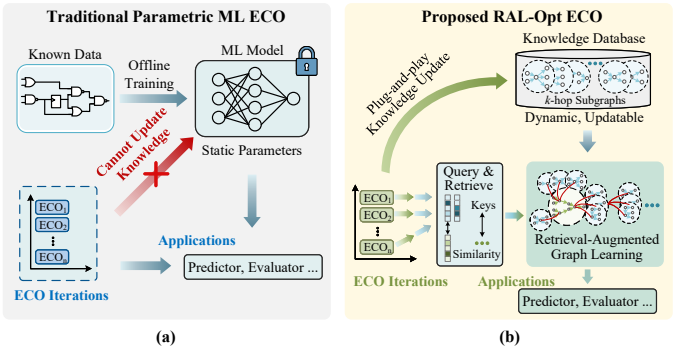

Figure 1: Comparison between (a) parametric ML and (b) our proposed RAL-Opt for ECO iterations.

The second category seeks to reduce the exploration burden of traditional algorithms over the ECO solution space. These methods either predict the optimization results of commercial tools to provide better initial solutions [4–8], or rank optimization targets by importance so as to narrow the search space explored by the optimizer [9].

Nevertheless, existing ML-based approaches generally follow the paradigm of parametric learning [10]. As illustrated in Figure 1 (a), once trained on labeled datasets, model parameters remain fixed during deployment. Incorporating new external knowledge typically requires retraining the model on additional data. However, retraining large-scale models incurs substantial computational overhead and may lead to catastrophic forgetting of previously learned knowledge. As a result, conventional ML methods struggle to incorporate evolving design knowledge. In contrast, ECO optimization is inherently an iterative process, where new optimization results and design knowledge are continuously generated throughout repeated design iterations. This mismatch between the static nature of parametric learning and the evolving knowledge produced during ECO iterations poses a significant challenge for existing ML-based ECO optimization methods.

Retrieval-Augmented Generation (RAG) has recently emerged as a prominent paradigm that enhances model adaptability beyond the limitations of static parametric learning by integrating a dynamic retrieval mechanism [11]. However, existing RAG techniques are primarily designed for text-based applications [12–15] and therefore cannot be directly applied to circuit optimization tasks. Unlike textual applications, circuit optimization operates on structured graph representations that capture circuit connectivity and design attributes. This modality mismatch prevents existing text-oriented chunking and retrieval mechanisms, as well as the associated retrieval-augmented learning strategies, from being directly applied to ECO optimization.

To address the above challenges, we propose **RAL-Opt**, a retrieval-augmented learning framework designed for continual knowledge integration in post-layout optimization. As shown in Figure 1 (b), unlike

*Corresponding author.

This work is licensed under a Creative Commons Attribution 4.0 International License.
ICCAD '26, San Jose, CA, USA
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2873-0/2026/11
https://doi.org/10.1145/3831252.3833935

existing ML-based ECO optimization methods, RAL-Opt enables seamless integration of evolving optimization knowledge from ECO iterations in a plug-and-play manner, without requiring model retraining. Specifically, RAL-Opt operates on circuit timing graphs, where seen designs are decomposed into graph chunks and stored in an external knowledge database. When optimizing unseen designs, relevant chunks are dynamically retrieved to enhance the effectiveness of ML-based ECO strategies. Moreover, optimization results obtained from unseen designs can be further decomposed and stored back into the database, enabling continual knowledge accumulation and progressive improvement across ECO iterations.

The main contributions of this work are summarized as follows:

- **Retrieval-augmented ML for ECO optimization.** We propose RAL-Opt, the first retrieval-augmented framework for ML-driven ECO optimization, enabling continual integration of evolving design knowledge without retraining.
- **Distillation-empowered retrieval.** We introduce a knowledge distillation strategy to address topological inconsistencies between database and unseen-design chunks, enabling reliable similarity matching.
- **Retrieval-augmented graph learning.** We develop a retrieval -augmented graph learning scheme that integrates logic-cone knowledge with local k-hop neighborhood information to enhance model inference.
- **Comprehensive evaluation.** We integrate RAL-Opt into multiple ML-based ECO optimization frameworks and validate its effectiveness through extensive experiments and ablation studies.

2 Preliminary

2.1 Post-layout Optimization

Fast and accurate timing and power optimization is essential for post-layout design closure. Traditional ECO methods formulate optimization problems using techniques such as sensitivity analysis [16], convex optimization [17], and Lagrangian relaxation [18, 19], and solve them with heuristic or greedy algorithms. However, accurate timing evaluation often requires physically aware ECO flows with repeated placement and routing (P&R), incurring substantial runtime overhead. ML-based methods mitigate this cost by replacing expensive P&R evaluations with fast inference. Trained on large-scale labeled datasets, these models either predict timing impacts [2, 3] or model optimizer behavior to accelerate optimization [4, 5, 7, 9]. Despite strong in-distribution performance, they often generalize poorly to unseen designs and cannot readily incorporate newly generated design knowledge, limiting their effectiveness in iterative ECO optimization.

2.2 Retrieval-Augmented Generation

RAG enhances pretrained language models by retrieving external knowledge and incorporating it through prompt-based conditioning, thereby improving factual consistency, reliability, and interpretability [12–15, 20, 21]. Conventional RAG systems use retrievers to select relevant documents from large knowledge corpora and provide them as contextual input to reader models [22–24]. Although RAG has achieved strong results in natural language processing and multimodal visual–text tasks [25–27], its application to structured data remains limited. Existing graph-oriented extensions, such as KG-RAG, largely rely on textual node descriptions [28–30]. Circuit optimization, however, provides little textual supervision, creating a fundamental mismatch with text-based chunking and retrieval. While [11] introduces structure-aware retrieval without text, it primarily relies on topological similarity. Post-layout circuits exhibit strong physical dependencies beyond topology, making topology-only retrieval insufficient for reliable knowledge matching and netlist decomposition. In this work, we aim to extend the RAG paradigm to post-layout optimization to improve both generalization and continual knowledge integration.

2.3 Problem Formulation

To address the aforementioned limitations, we formally define the problem as follows:

Problem 1 (Retrieval-Augmented Learning for Post-layout Optimization). *Given seen post-layout designs $\mathcal{G}_S$ and their optimization results $\mathcal{R}$, the goal is to decompose them into reusable chunks and leverage these chunks through retrieval-augmented learning to improve optimization on unseen designs.*

3 Methodology Overview

This section presents an overview of RAL-Opt. The framework consists of three key components: knowledge database construction (Section 4 and Figure 2), retriever pretraining (Section 5.1), and retrieval-augmented learning (Section 5.2).

Knowledge Database (Section 4). The pipeline begins by constructing a knowledge database to support retrieval and learning. We extract timing graphs from seen designs and decompose them into reusable chunks. Each chunk is centered on the outpin of a logic cell and comprises its k-hop neighborhood. It is further augmented with metadata for efficient indexing and context differentiation.

Prior ML-based approaches often adopt large partitioning strategies, such as logic cone-based [31] or critical path-based [7] decompositions. While these methods preserve structural completeness, they reduce comparability across designs. Large partitions couple multiple functional contexts and introduce significant structural variability, making reliable correspondence between query and database samples difficult and degrading retrieval stability. In contrast, the proposed k-hop, outpin-centered decomposition provides a balanced granularity. It preserves local structural and timing dependencies while maintaining comparable representations across designs, enabling efficient indexing and fine-grained matching.

Retriever Pretraining (Section 5.1). Given the constructed knowledge database, the next step is to pretrain the encoder within the retriever. A fundamental challenge is the representation misalignment between database chunks and downstream task inputs. The database is composed of k-hop local neighborhoods, whereas downstream optimization tasks operate on larger subgraphs that capture global semantics, such as logic-cone structures. This cross-scale discrepancy leads to inconsistent representations and hinders the retriever from identifying semantically relevant knowledge.

To address this issue, we introduce a distillation-based pretraining framework as a cross-scale alignment mechanism. It enforces consistency between representations derived from local database chunks and those associated with task-level structures, thereby bridging the semantic gap across different granularities. As a result, the pretrained encoder produces embeddings that are both locally expressive and globally aligned, enabling reliable retrieval for downstream tasks.

Retrieval-Augmented Learning (Section 5.2). In the final stage, we develop an end-to-end retrieval-augmented learning framework. Given a query design, the pretrained retriever encodes its task-specific subgraphs and retrieves, for each outpin, the top-N most relevant chunks from the knowledge database based on semantic similarity. A lightweight decoder integrates the retrieved knowledge with the query representations through a learnable aggregation mechanism, and maps the resulting features to the final predictions.

The framework is modular and can be seamlessly integrated into existing ML-based ECO optimization pipelines, where it can function as a surrogate model or a predictive module. Unseen designs can be incrementally decomposed and incorporated into the database, enabling continual knowledge accumulation and progressive performance improvement across optimization iterations.

4 Knowledge Database Construction

Figure 2 illustrates the netlist preprocessing pipeline and the knowledge database architecture of RAL-Opt. We represent the circuit as a timing graph, which serves as the fundamental abstraction for performance evaluation and is widely adopted in post-layout optimization [7, 19, 32]. The graph is first structurally refined by removing constraint-related arcs to retain only timing-relevant dependencies. To enable fine-grained knowledge extraction, we decompose the refined graph into a set of k-hop subgraphs, each centered at the outpin of a logic cell (with $k = 3$ illustrated in the figure). Overlapping subgraphs are intentionally preserved to ensure full coverage of all logic cell outputs, allowing each outpin to serve as a semantic anchor for localized structural context.

These subgraphs are then organized into a knowledge database. To address the I/O bottleneck associated with large-scale graph data, we adopt a hierarchical storage architecture that decouples lightweight indexing from heavy tensor storage. Specifically, each subgraph is stored as an independent data object, enabling on-demand and fine-grained retrieval without loading the entire dataset. To enhance scalability, each subgraph is assigned a unique hash-based signature, and a prefix-based sharding strategy is employed to distribute data across storage buckets, effectively reducing I/O contention in large-scale scenarios. The index file maintains not only the mapping between signatures and storage locations, but also rich metadata describing subgraph identity, including the source design identifier and the corresponding center outpin. This design allows the retrieval process to operate primarily on lightweight index queries, enabling rapid and precise localization of relevant subgraphs while avoiding expensive access to large tensor files.

5 Algorithms

The overall workflow of RAL-Opt is illustrated in Figure 3. The training process consists of two stages. In the first stage, the retriever is pretrained to learn two task-specific encoders. These encoders not only achieve strong performance on downstream tasks but also produce mutually aligned representations, ensuring a consistent cross-source embedding space. Building upon this alignment, the second stage focuses on retrieval-augmented learning. Given the embeddings generated by the pretrained encoders, we train a stable decoder that effectively leverages retrieved knowledge to aggregate task-relevant information, thereby enhancing model performance on downstream tasks.

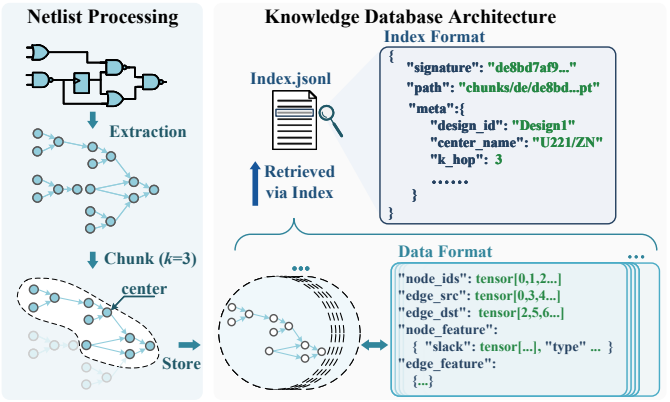

Figure 2: Netlist preprocessing and knowledge database architecture of RAL-Opt.

5.1 Stage 1: Retriever Pre-training

After constructing the knowledge database, the primary challenge lies in enabling effective retrieval. Unlike text-based RAG frameworks, where retrieval can be performed via lexical similarity, circuit chunks lack explicit textual semantics, rendering such approaches inapplicable. Instead, we adopt a representation-based retrieval paradigm, where each chunk is encoded into a latent embedding space, and retrieval is performed via semantic similarity.

However, as aforementioned, to ensure cross-design comparability, the knowledge database is constructed using k-hop local subgraphs. Such localized representations, while efficient for storage and retrieval, are insufficient to fully capture the structural and contextual dependencies required in post-layout optimization. For instance, the optimal sizing strategy for a cell depends not only on its local neighborhood but also on its position within timing paths and its interactions with multiple paths. Consequently, downstream ML-driven ECO optimization typically requires more structurally complete representations, leading to a fundamental cross-scale discrepancy between database chunks and downstream task inputs.

To bridge this gap, we design a dual-encoder retriever architecture. For downstream tasks, we employ logic-cone-based subgraphs, which have been widely adopted in circuit foundation models [31] and performance prediction frameworks [33, 34]. Starting from each register, we backtrace all driving logic up to other registers to construct a logic cone that captures the full context affecting timing behavior. To effectively model both local interactions and global dependencies within the logic cone, we adopt a graph transformer (GFormer) [35], which combines a graph neural network (GNN) with a linear-complexity attention mechanism. The attention operation is defined as:

$$\text{LinearAttn}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \phi(\mathbf{Q})\big(\phi(\mathbf{K})^\top \mathbf{V}\big) \Big/ \phi(\mathbf{Q})\big(\phi(\mathbf{K})^\top \mathbf{1}\big), \tag{1}$$

where $\phi(\cdot)$ denotes a kernel mapping function that enables linear-time attention with respect to the number of nodes. In contrast, for the knowledge database, each chunk corresponds to a localized k-hop subgraph centered at an outpin. Since such subgraphs focus on local structural patterns, a lightweight GNN encoder is sufficient to aggregate neighborhood information efficiently.

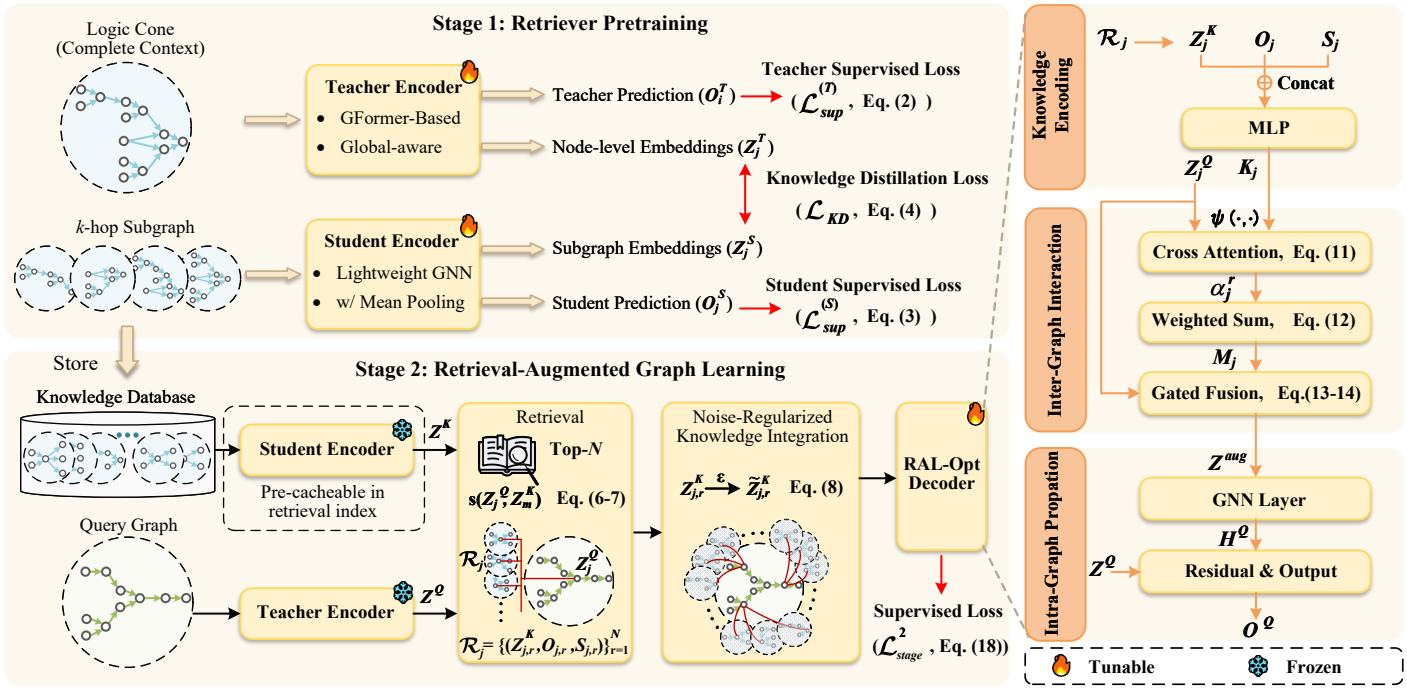

Figure 3: Overview of the RAL-Opt framework. Stage 1 aligns representations between database chunks and downstream inputs via knowledge distillation. Stage 2 enhances downstream performance through retrieval-augmented graph learning. The rightmost panel details the RAL-Opt decoder architecture.

To align these heterogeneous representations, we adopt a teacher–student knowledge distillation framework. As illustrated in the upper-left region of Figure 3, each training sample consists of a logic cone together with its k-hop subgraphs centered at outpins. The teacher encoder operates on the logic cone and learns globally informed node representations under node-level supervision. By contrast, the student encoder processes each outpin-centered k-hop subgraph independently. Specifically, it first performs message passing within the local subgraph, and then applies mean pooling to obtain a compact subgraph-level representation. This pooled representation is used for prediction, while the supervision signal is given by the label associated with the corresponding center outpin. In this way, the student learns an outpin-centered subgraph predictor that preserves retrieval efficiency while remaining aware of the local structural context.

Supervised Learning Objective. For the teacher encoder, supervision is applied at the node level over the logic cone. Let $\mathbf{O}_i^T$ denote the prediction at node i and $\mathbf{y}_i$ the corresponding ground truth. We adopt a task-dependent objective $\mathcal{L}_{sup}^{(T)}$ defined as:

$$\mathcal{L}_{sup}^{(T)} = \frac{1}{|\mathcal{V}_m|} \sum_{i \in \mathcal{V}_m} \ell(\mathbf{O}_i^T, \mathbf{y}_i), \tag{2}$$

where $\mathcal{V}_m$ denotes the set of valid nodes (excluding undefined labels), and $\ell(\cdot)$ represents a task-specific loss function (e.g., cross-entropy for classification or ℓ_2 regression objectives for timing prediction).

For the student encoder, supervision is imposed on each outpin-centered k-hop subgraph. Let $\mathbf{g}_j$ denote the k-hop subgraph centered at outpin j, and let $\mathbf{O}_j^S$ be the prediction generated from its pooled subgraph representation. The supervised objective is defined as:

$$\mathcal{L}_{sup}^{(S)} = \frac{1}{|\mathcal{P}_m|} \sum_{j \in \mathcal{P}_m} \ell(\mathbf{O}_j^S, \mathbf{y}_j), \tag{3}$$

where $\mathcal{P}_m$ denotes the set of valid outpins and $\mathbf{y}_j$ is the label associated with the center outpin of $\mathbf{g}_j$.

Knowledge Distillation. To transfer global structural knowledge from the teacher to the student, we align their latent representations at the outpin level. Let $\mathbf{Z}_j^T$ and $\mathbf{Z}_j^S$ denote the embeddings produced by the teacher and student encoders, respectively, for outpin j. Let $\mathcal{P}$ denote the set of outpins within a given logic cone. We define a representation-level distillation loss as:

$$\mathcal{L}_{KD} = \frac{1}{|\mathcal{P}|} \sum_{j \in \mathcal{P}} \left\| \mathbf{Z}_j^S - \mathbf{Z}_j^T \right\|_2^2, \tag{4}$$

which encourages the student encoder to mimic the global-aware representations learned by the teacher.

Overall Pre-Training Objective. The overall training objective combines supervised learning and knowledge distillation:

$$\mathcal{L}_{\text{stage}}^1 = \mathcal{L}_{sup}^{(T)} + \mathcal{L}_{sup}^{(S)} + \lambda \mathcal{L}_{KD}, \tag{5}$$

where λ controls the trade-off between task supervision and representation alignment.

This design enables the student encoder to inherit richer global contextual knowledge from the logic-cone-based teacher, while maintaining the efficiency and locality required for scalable retrieval over the knowledge database.

5.2 Stage 2: Retrieval-Augmented Learning

After pretraining the retriever, the next challenge is how to effectively and stably exploit the knowledge stored in the database. To this end, we propose a retrieval-augmented graph learning paradigm. A lightweight decoder integrates retrieved subgraph knowledge with the query logic cone through collaborative graph propagation, thereby enhancing task-specific inference.

Retrieval and Knowledge Selection. Each training iteration starts from a query logic cone $\mathcal{G}_Q$ sampled from a design $\mathcal{D}_i$. To prevent information leakage, all database chunks originating from $\mathcal{D}_i$ are excluded via metadata filtering. For the remaining candidates, we compute similarity scores in the shared embedding space induced by Stage 1.

Specifically, let $\mathbf{Z}_j^Q \in \mathbb{R}^d$ denote the embedding of the j-th outpin in the query logic cone, and let $\mathbf{Z}_m^K \in \mathbb{R}^d$ denote the embedding of the m-th chunk in the knowledge database. For each query outpin, retrieval is performed by measuring its similarity to all candidate database chunks. We adopt the normalized inner product:

$$s(\mathbf{Z}_j^Q, \mathbf{Z}_m^K) = \frac{\langle \mathbf{Z}_j^Q, \mathbf{Z}_m^K \rangle}{\|\mathbf{Z}_j^Q\|_2 \|\mathbf{Z}_m^K\|_2}. \tag{6}$$

For each outpin j, we retrieve the top-N most relevant chunks:

$$\mathcal{R}_j = \text{Top-}N\Big(\{s(\mathbf{Z}_j^Q, \mathbf{Z}_m^K)\}_{m=1}^M\Big), \tag{7}$$

where M is the number of candidate chunks after filtering, and $\mathcal{R}_j$ denotes the retrieved knowledge set associated with outpin j. Notably, since the retriever is frozen in Stage 2, all database chunks can be precomputed, with embeddings stored and linked via index metadata, enabling direct lookup without on-the-fly encoding.

Noise-Regularized Knowledge Integration. Inspired by noise-based regularization techniques in retrieval-augmented learning [36–38], we introduce stochastic perturbations to the retrieved embeddings during training. This design aims to improve robustness against retrieval noise and mitigate over-reliance on exact embedding matches, which may be sensitive to distribution shifts across designs.

Specifically, we inject Gaussian noise into retrieved embeddings:

$$\tilde{\mathbf{Z}}_{j,r}^K = \mathbf{Z}_{j,r}^K + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{I}), \tag{8}$$

where $\mathbf{Z}_{j,r}^K \in \mathbb{R}^d$ denotes the r-th retrieved embedding for outpin j, and σ controls the noise scale. This stochastic perturbation encourages the decoder to learn more robust retrieval-conditioned representations, thereby improving generalization.

Knowledge Encoding. For each outpin j, the retrieved knowledge is represented as a set of triplets:

$$\mathcal{R}_j = \{(\tilde{\mathbf{Z}}_{j,r}^K, \mathbf{O}_{j,r}, \mathbf{S}_{j,r})\}_{r=1}^N, \tag{9}$$

where each triplet consists of a noise-perturbed embedding $\tilde{\mathbf{Z}}_{j,r}^K$, the corresponding student prediction $\mathbf{O}_{j,r}$, and the similarity score $\mathbf{S}_{j,r}$. These features are concatenated and projected into a unified representation via a multilayer perceptron (MLP):

$$\mathbf{K}_j = \text{MLP}\big(\text{Concat}\big[\tilde{\mathbf{Z}}_j^K, \mathbf{O}_j, \mathbf{S}_j\big]\big), \tag{10}$$

where $\mathbf{K}_j \in \mathbb{R}^{N \times d}$ is aligned with the query embedding space.

Inter-Graph Interaction. To integrate retrieved knowledge with the query graph, we introduce a cross-graph interaction mechanism that enables retrieval-conditioned feature modulation. For each outpin embedding $\mathbf{Z}_j^Q$, we compute its relevance to the retrieved knowledge $\{\mathbf{K}_{j,r}\}_{r=1}^N$ through a learnable compatibility function $\psi(\cdot,\cdot)$, implemented as a lightweight neural scorer. The attention weights are defined as:

$$\alpha_{j,r} = \frac{\exp\big(\psi(\mathbf{Z}_j^Q, \mathbf{K}_{j,r})\big)}{\sum_{r'} \exp\big(\psi(\mathbf{Z}_j^Q, \mathbf{K}_{j,r'})\big)}, \tag{11}$$

which quantify the relative importance of each retrieved subgraph with respect to the query outpin. The retrieved knowledge is then aggregated as a weighted combination:

$$\mathbf{M}_j = \sum_{r=1}^N \alpha_{j,r} \mathbf{K}_{j,r}, \tag{12}$$

yielding a retrieval-informed message $\mathbf{M}_j$.

To adaptively fuse retrieved knowledge with intrinsic query representations, we employ a gated fusion mechanism to obtain the augmented embedding $\mathbf{Z}_j^{\text{aug}}$:

$$\mathbf{Z}_j^{\text{aug}} = \gamma_j \odot \mathbf{M}_j + (1 - \gamma_j) \odot \mathbf{Z}_j^Q, \tag{13}$$

where the gate $\gamma_j \in (0,1)^d$ is dynamically generated as:

$$\gamma_j = \sigma\big(\mathbf{W}[\mathbf{Z}_j^Q \| \mathbf{M}_j]\big). \tag{14}$$

Here, $\mathbf{W}$ is a learnable parameter matrix, and $\sigma(\cdot)$ denotes the sigmoid function. This gating mechanism enables the model to adaptively balance the contribution of retrieved knowledge for each outpin and feature

Algorithm 1 Overall Training Methodology for RAL-Opt

Input: Logic-cone dataset $\mathcal{C}$, knowledge database $\mathcal{G}$, target variable y, top-N, distillation weight λ, noise scale σ
Output: Trained teacher encoder $f_T^\star$, trained student encoder $f_S^\star$, retrieval index $\mathcal{I}$, trained decoder $f_{\text{dec}}^\star$

Stage 1: Retriever Pre-training
Initialize teacher encoder f_T and student encoder f_S
for each training epoch **do**
  for each paired mini-batch $(\mathcal{B}_Q, \mathcal{B}_K)$ with $\mathcal{B}_Q \subset \mathcal{C}, \mathcal{B}_K \subset \mathcal{G}$ **do**
    Encode $\mathcal{B}_Q$ with f_T and compute $\mathcal{L}_{sup}^{(T)}$ ▷ Eq. (2)
    Encode $\mathcal{B}_K$ with f_S and compute $\mathcal{L}_{sup}^{(S)}$ ▷ Eq. (3)
    Align teacher and student embeddings through $\mathcal{L}_{KD}$ ▷ Eq. (4)
    Update f_T and f_S using $\mathcal{L}_{\text{stage}}^1$ ▷ Eq. (5)
  end for
end for
Obtain the trained retriever $\{f_T^\star, f_S^\star\}$
Construct the retrieval index $\mathcal{I}$ with $f_S^\star$

Stage 2: Retrieval-Augmented Learning
Freeze the retriever $\{f_T^\star, f_S^\star, \mathcal{I}\}$ and initialize decoder f_{dec}
for each training epoch **do**
  for each sampled query mini-batch $\mathcal{B}_Q \subset \mathcal{C}$ **do**
    Encode $\mathcal{B}_Q$ with $f_T^\star$ to obtain $\mathbf{Z}_Q$
    Retrieve top-N knowledge chunks with $\mathcal{I}$ ▷ Eqs. (6)–(7)
    Apply noise-regularized retrieval ▷ Eq. (8)
    Encode the retrieved knowledge into $\mathbf{K}$ ▷ Eq. (10)
    Perform inter-graph interaction to obtain $\mathbf{Z}_Q^{\text{aug}}$ ▷ Eqs. (11)–(14)
    Propagate $\mathbf{Z}_Q^{\text{aug}}$ over $\mathcal{B}_Q$ and generate outputs ▷ Eqs. (15)–(17)
    Update f_{dec} using $\mathcal{L}_{\text{stage}}^2$ ▷ Eq. (18)
  end for
end for
Obtain the trained decoder $f_{\text{dec}}^\star$
return $\{f_T^\star, f_S^\star, \mathcal{I}\}$, and $f_{\text{dec}}^\star$

dimension, allowing flexible integration of external knowledge with the original query representation.

Intra-Graph Propagation. While the above interaction operates at the outpin level, the retrieved knowledge must be propagated across the entire logic cone to influence downstream predictions. To this end, we perform intra-graph propagation over the query graph:

$$\mathbf{H}^{\mathcal{Q}} = \mathrm{GNN}(\mathcal{G}^{\mathcal{Q}}, \mathbf{Z}^{\mathrm{aug}}), \tag{15}$$

where the GNN performs message passing along circuit connectivity. This step effectively broadcasts the retrieval-enhanced representations from outpins to all nodes, allowing the injected knowledge to interact with local structural dependencies and propagate along timing paths.

Training Objective. Finally, the propagated features are integrated with the original query representations through a residual connections:

$$\mathbf{Z}^{\mathcal{Q}}_{\mathrm{final}} = \mathbf{Z}^{\mathcal{Q}} + \mathbf{H}^{\mathcal{Q}}, \tag{16}$$

which preserves the original information while incorporating retrieval-conditioned updates. The resulting representations are then fed into a task-specific output head:

$$\mathbf{O}^{\mathcal{Q}} = f_{\mathrm{head}}(\mathbf{Z}^{\mathcal{Q}}_{\mathrm{final}}), \tag{17}$$

to produce final predictions $\mathbf{O}^{\mathcal{Q}}$.

The model is trained using a supervised objective:

$$\mathcal{L}^{2}_{\mathrm{stage}} = \frac{1}{|\mathcal{V}_m|} \sum_{i \in \mathcal{V}_m} \ell(\mathbf{O}_i, \mathbf{y}_i), \tag{18}$$

where $\mathcal{V}_m$ denotes the set of valid nodes and $\ell(\cdot)$ is a task-dependent loss function.

5.3 Overall Training Methodology

The overall training pipeline of RAL-Opt is summarized in Algorithm 1. Stage 1 establishes a cross-scale retriever, consisting of a teacher encoder and a student encoder, which align logic-cone queries and database chunks into a unified, task-aware embedding space. Using the trained student encoder, all knowledge database chunks are encoded offline, creating a retrieval index $\mathcal{I}$ that stores chunk embeddings alongside their predictive values and metadata. This index facilitates efficient relevance search during both Stage 2 training and inference, enabling rapid retrieval of task-related knowledge without the need for repeated encoding. Once the retriever and index are fixed, Stage 2 fine-tunes a lightweight decoder, which incorporates retrieved knowledge into query-graph reasoning to produce the final predictions.

6 Experimental Results

6.1 Experimental Setup

For data preparation, we compile a diverse set of open-source circuit designs from the IWLS benchmarks [39], OpenCores [40], and Chipyard [41]. All RTL designs are synthesized using Synopsys Design Compiler [42] with a commercial 12nm technology library. Physical design is performed in Synopsys IC Compiler II [43], and design quality is analyzed using Synopsys PrimeTime [1]. The statistical details of the dataset are provided in Table 1.

For implementation, RAL-Opt is developed in Python 3.12 and implemented using PyTorch 2.4.1. The GNN components employed in the teacher encoder and decoder modules are based on Graph Convolutional Network (GCN) [44], while the student encoder is built upon Graph Attention Network (GAT) [45]. The hidden dimensionality of both the encoder and decoder is set to 256. In Stage 1, the distillation loss weight λ is set to 0.5. The retriever is trained for 40 epochs, followed by 10 epochs of training for the decoder in Stage 2. For fair comparison, all baseline methods are trained for 50 epochs. All experiments are conducted on a Linux machine equipped with two AMD EPYC 7B13 CPUs, 768 GB of memory, two NVIDIA GeForce RTX 4090 GPUs, and two NVIDIA RTX 6000 Ada Generation GPUs.

Table 1: Benchmark design information.

Benchmarks	#Designs	Design Size Range		
		#K Cells	#K Pins	#K EndPoints
IWLS	14	2-78	9-263	1-17
OpenCores	4	16-354	51-1066	4-44
Chipyard	3	266-1127	884-3120	15-116

Table 2: Post-optimization timing results prediction accuracy comparison. The unit of "MAE" is *ps*.

Unseen designs (#Cells)	DAC23 [2]		TCAD24 [3]		**RAL-Opt (ours)**	
	MAE	R^2	MAE	R^2	MAE	R^2
Design1 (14.7k)	6.13	0.86	4.54	0.92	**3.89**	**0.94**
Design2 (27.2k)	4.04	0.71	3.43	0.84	**3.01**	**0.86**
Design3 (31.0k)	81.32	0.87	76.57	0.89	**58.40**	**0.91**
Design4 (39.2k)	17.99	0.75	9.53	0.92	**5.87**	**0.96**
Design5 (77.9k)	10.40	0.66	**5.62**	**0.87**	5.96	0.86
Design6 (81.9k)	9.77	0.85	11.21	0.81	**7.90**	**0.86**
Design7 (180.4k)	26.59	0.79	26.21	0.82	**24.40**	**0.87**
Design8 (1127.5k)	18.75	0.78	13.50	0.83	**11.68**	**0.89**
Average	21.87	0.79	18.83	0.86	**15.15**	**0.89**

6.2 Performance on Various Downstream Tasks

To comprehensively evaluate the effectiveness of the proposed RAL-Opt framework, we integrate it into multiple representative ML-based ECO optimization paradigms, including surrogate-driven and prediction-accelerated frameworks. We consider commonly adopted ECO operations, including gate sizing and buffering, and adapt them according to different baseline settings. We compare against a range of task-specific baselines within these paradigms. Experimental results consistently demonstrate that RAL-Opt achieves superior performance over state-of-the-art (SOTA) models across different settings, validating the effectiveness of retrieval-augmented learning in improving generalization on out-of-distribution designs.

Task 1: Retrieval-Augmented Surrogate. In this task, RAL-Opt is employed to predict the timing impact of ECO operations. We compare it against a graph-based surrogate (DAC23 [2]) and a SOTA multi-modal predictor (TCAD24 [3]). As reported in Table 2, RAL-Opt consistently achieves the best overall performance, reducing the average mean absolute error (MAE) from 21.87 and 18.83 to 15.15, while improving R^2 score to 0.89. These results demonstrate its superior predictive accuracy across diverse unseen designs.

The performance gains can be primarily attributed to the retrieval-augmented paradigm. Unlike purely parametric models that rely solely on fixed learned representations, RAL-Opt explicitly retrieves task-relevant subgraph knowledge and integrates it into inference. This non-parametric augmentation effectively mitigates the distribution gap between training and unseen designs, enabling the model to leverage historical optimization patterns rather than extrapolating from limited data. Consequently, RAL-Opt achieves stronger generalization than both

Table 3: Optimization result comparison for Task 2. "WNS", "TNS", "NVE", and "Power" denote Worst Negative Slack, Total Negative Slack, Number of Violated Endpoints, and total power, respectively. The units of "WNS" and "TNS" are ns, and that of "Power" is mW.

Unseen designs	Before Optimization				Tool Reference (32 threads) [1]					**RAL-Opt + Commercial Tool (ours)**				
	WNS	TNS	NVE	Power	WNS	TNS	NVE	Power	speed-up	WNS	TNS	NVE	Power	speed-up
Design1	-0.022	-0.129	11	5.161	-0.019	-0.063	8	5.186	1.00	**-0.018**	**-0.061**	**7**	5.184	**1.75×**
Design2	-0.032	-2.051	176	23.582	-0.031	-1.725	160	22.272	1.00	**-0.030**	**-1.687**	**157**	22.298	**2.50×**
Design3	-3.333	-93.938	32	0.436	-3.231	-90.653	32	0.443	1.00	**-3.207**	**-90.025**	32	0.443	**1.22×**
Design4	-0.482	-336.001	1401	1.766	-0.429	**-335.594**	1401	1.754	1.00	-0.429	-335.710	1401	1.732	**1.60×**
Design5	-0.198	-2.065	30	48.385	-0.198	-0.304	4	48.386	1.00	-0.198	-0.304	4	48.386	**1.33×**
Design6	-0.064	-2.295	91	25.232	-0.063	-0.795	33	25.232	1.00	-0.063	**-0.794**	33	25.232	**1.50×**
Design7	-1.378	-75.033	64	5.415	-1.371	-74.76	64	5.415	1.00	-1.371	-74.76	64	5.415	**1.54×**
Design8	-1.889	-214.765	295	35.985	-1.768	-129.646	269	36.585	1.00	-1.768	**-129.644**	269	36.685	**1.75×**
Average	-	-	-	-	1.000	1.000	1.000	1.000	1.00	**0.986**	**0.991**	**0.979**	**0.998**	**1.64×**

graph-based and multi-modal parametric predictors, whose knowledge remains implicitly encoded and bounded by the training distribution.

Task 2: Retrieval-Augmented Predictor. In this task, RAL-Opt is applied to predict the optimization outcomes produced by commercial tools, following prior works [4–8]. These predictions are further utilized as high-quality initial solutions to accelerate downstream optimization. We compare RAL-Opt with representative models from TODAES23 [5], ISPD25 [7], and ASPDAC26 [8]. As shown in Table 4, RAL-Opt achieves the highest average F1 score (0.917), consistently outperforming all baselines, while maintaining a runtime comparable to TODAES23 and ISPD25 and significantly lower than ASPDAC26.

This advantage arises from the efficient integration of retrieval and inference. By leveraging a pre-encoded retrieval index, RAL-Opt avoids repeatedly processing the entire dataset and enables fast access to task-relevant knowledge. Combined with a lightweight decoder and efficient attention mechanisms, this design achieves a favorable efficiency–accuracy trade-off. When integrated into a commercial tool [1], the predicted results provide effective initialization, yielding up to 2.50× runtime speedup while preserving, or even slightly improving, optimization quality. These results highlight the practical value of RAL-Opt in accelerating real-world ECO flows.

Task 3: Retrieval-Augmented Planner. In this task, RAL-Opt serves as a planner to identify critical pins for ECO modification, following prior work [9]. By imposing an explicit optimization order over critical regions, it effectively constrains the search space of the downstream optimizer. We adopt the Lagrangian relaxation (LR)-based solver [19] as the backend and compare RAL-Opt with the heterogeneous graph neural network (HGNN)-based planner [9]. The results in Table 5 show that both ML-driven planners outperform the conventional LR solver, validating the importance of structured search space reduction.

The improvement is attributed to retrieval-augmented planning, which precisely localizes critical repair regions. In contrast, full-graph optimization introduces a large, highly coupled search space, often leading to oscillatory behaviors (e.g., the ping-pong effect) and suboptimal convergence. By leveraging retrieved historical patterns, RAL-Opt enables more accurate and stable decision-making, thereby reducing optimization complexity and improving solution quality. As a result, the RAL-Opt-guided solver achieves the best overall performance, delivering an average 1.49× speedup over the LR solver while improving WNS and TNS by 1.7% and 9.3%, respectively, with marginal power overhead.

Table 4: Optimization outcome prediction accuracy (F1 score) and runtime speedup refer to commercial tool.

Unseen designs	TODAES23 [5]		ISPD25 [7]		ASPDAC26 [8]		**RAL-Opt**	
	F1	speed-up	F1	speed-up	F1	speed-up	F1	speed-up
Design1	0.741	1285×	0.871	866×	0.895	378×	**0.935**	357×
Design2	0.647	1579×	0.932	1005×	0.883	554×	**0.926**	1127×
Design3	0.757	1672×	0.918	673×	0.895	182×	**0.939**	378×
Design4	0.695	1240×	0.856	949×	0.823	320×	**0.905**	454×
Design5	0.711	974×	0.862	696×	**0.865**	331×	0.858	352×
Design6	0.633	1453×	0.887	683×	0.906	322×	**0.938**	579×
Design7	0.656	1808×	0.914	944×	0.864	255×	**0.929**	582×
Design8	0.711	1423×	0.887	811×	0.846	397×	**0.902**	640×
Average	0.694	1429×	0.891	828×	0.872	342×	**0.917**	558×

6.3 Effectiveness of Retrieval for Continual Knowledge Integration

Figure 4 compares the performance gains achieved by incremental fine-tuning in parametric ML models and retrieval-based knowledge integration in RAL-Opt on Design6. In RAL-Opt, newly generated optimization knowledge is decomposed into graph chunks after each ECO iteration and appended to an external knowledge database. In contrast, parametric ML incorporates new knowledge by fine-tuning the teacher model GFormer on accumulated data.

As shown in Figure 4 (a), fine-tuning provides initial gains but becomes increasingly unstable as optimization proceeds. Its performance gradually degrades and, for Task 2, eventually falls below that of the initial model without fine-tuning. Moreover, Figure 4 (b) shows that fine-tuning does not yield consistent improvements across test cases and can cause substantial degradation, highlighting the susceptibility of parametric models to catastrophic forgetting under continual updates.

In contrast, RAL-Opt incorporates new knowledge without modifying model parameters. Newly generated knowledge is added to the external database in a plug-and-play manner and retrieved according to relevance, enabling stable performance gains across ECO iterations. Performance aggregated over all designs also remains stable or slightly improves, indicating that newly accumulated knowledge does not interfere with previously acquired representations.

Figure 5 further provides a qualitative visualization of the optimization process in Task 3 on Design12. As ECO iterations progress, the advantage of RAL-Opt becomes increasingly evident. Benefiting from

Table 5: Optimization result comparison for Task 3. The units of "WNS" and "TNS" are *ns*, and that of "Power" is *mW*.

Unseen designs (#Cells)	LR Solver-Only [19]				LR Solver + HGNN [9]					**LR Solver + RAL-Opt (ours)**				
	WNS	TNS	NVE	Power	WNS	TNS	NVE	Power	speed-up	WNS	TNS	NVE	Power	speed-up
Design9 (7.5K)	-0.032	-0.572	41	**4.270**	-0.032	-0.509	39	4.273	1.81×	-0.032	**-0.482**	**37**	4.273	**1.90×**
Design10 (11.2K)	-0.053	-0.344	11	**9.744**	-0.053	-0.245	9	9.745	1.12×	**-0.051**	**-0.242**	**9**	9.745	**1.20×**
Design11 (15.8K)	-0.601	-34.952	78	**1.584**	-0.598	-34.754	78	1.585	1.12×	**-0.562**	**-33.464**	78	1.585	**1.19×**
Design12 (55.7K)	-0.119	-3.644	118	**1.732**	-0.119	-3.623	119	1.754	1.45×	-0.119	**-3.452**	**112**	1.766	**1.48×**
Design13 (265.7K)	-1.495	-109.960	220	7.155	-1.494	-109.818	220	7.155	**1.54×**	**-1.493**	**-109.393**	**215**	7.156	1.53×
Design14 (354.3K)	-0.186	-61.985	1114	116.726	-0.186	-61.652	1099	**116.725**	**1.67×**	-0.186	**-61.582**	1099	116.784	1.65×
Average	1.000	1.000	1.000	**1.000**	0.999	0.931	0.961	1.002	1.45×	**0.983**	**0.907**	**0.939**	1.003	**1.49×**

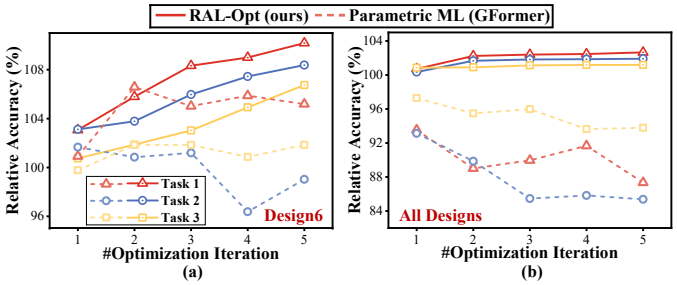

Figure 4: A case study of knowledge integration during ECO iterations on Design6, comparing RAL-Opt and parametric model fine-tuning, where (a) shows performance on Design6 and (b) shows performance aggregated across all designs.

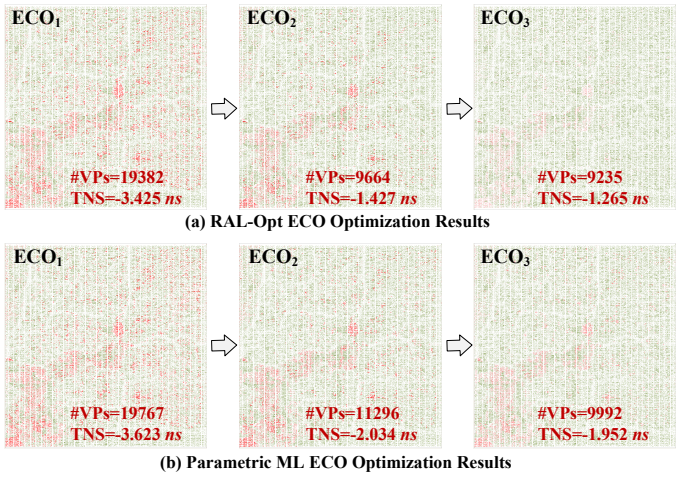

Figure 5: Visualization of optimization results across ECO iterations on Design12 under the Task 3 Paradigm for (a) RAL-Opt and (b) Parametric ML. Violated pins (VPs) are highlighted in red.

its retrieval-driven continual learning capability, RAL-Opt consistently achieves superior optimization outcomes compared to parametric ML. Specifically, the improvement in TNS grows from an initial 5.5% to over 35% in later iterations, clearly demonstrating the cumulative advantage of retrieval-based knowledge integration in iterative ECO optimization.

6.4 Ablation Study

We compare the complete RAL-Opt framework against four ablated variants: replacing our chunking scheme with the generic chunking method of [11] (w/ GC), removing retrieval (w/o Retrieval), knowledge distillation (w/o KD), or noise-regularized integration (w/o Noise). The results are reported in Figure 6.

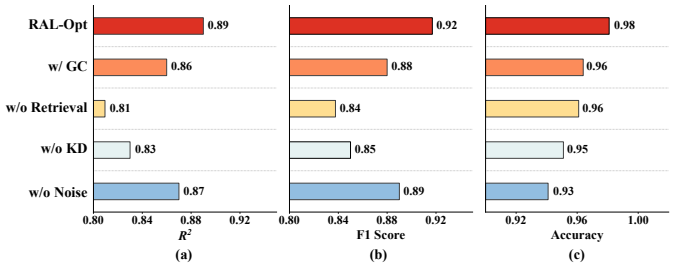

Figure 6: Ablation results of RAL-Opt on (a) Task 1, (b) Task 2, and (c) Task 3.

Retrieval yields the largest gains on Tasks 1 and 2, underscoring its importance for complex regression and classification tasks. By incorporating relevant historical subgraphs, it substantially improves generalization beyond the training distribution. Moreover, our outpin-oriented chunking strategy consistently outperforms generic chunking, demonstrating its greater suitability for circuit optimization. Knowledge distillation improves performance across all three tasks, suggesting that aligning retrieved chunks with task-level subgraphs enables more effective knowledge matching and higher-quality retrieval. By contrast, noise-regularized integration has the greatest impact on Task 3, indicating that, for tasks with limited label diversity, Gaussian perturbations improve robustness and prediction accuracy.

7 Conclusion

In this paper, we present RAL-Opt, a retrieval-augmented learning framework for continual knowledge integration in post-layout optimization. By maintaining an external knowledge database and incorporating retrieved knowledge during training and inference, RAL-Opt enables explicit reuse of historical optimization patterns and consistently improves existing ML-based methods across diverse ECO tasks. Its non-parametric design further supports plug-and-play integration of newly generated knowledge from iterative ECO processes without costly retraining. Future work will extend RAL-Opt to multimodal settings.

Acknowledgments

This work was supported by the NSFC (62474038, 62274034), the Key R&D Program of Jiangsu Province (BE2023003-2), the NSF of Jiangsu Province (BK20232005).

References

[1] Synopsys, "Primetime User Guide", https://www.synopsys.com/implementation-and-signoff/signoff/primetime.html, 2024.

[2] P. Pham *et al.*, "Agd: A learning-based optimization framework for eda and its application to gate sizing," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2023, pp. 1–6.

[3] Y. Ye *et al.*, "Learning-driven physically aware large-scale circuit gate sizing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 44, no. 5, pp. 1901–1914, 2024.

[4] C.-K. Cheng *et al.*, "Dagsizer: A directed graph convolutional network approach to discrete gate sizing of vlsi graphs," *ACM Transactions on Design Automation of Electronic Systems*, vol. 28, no. 4, pp. 1–31, 2023.

[5] Y.-C. Lu *et al.*, "Eco-gnn: Signoff power prediction using graph neural networks with subgraph approximation," *ACM Transactions on Design Automation of Electronic Systems*, vol. 28, no. 4, pp. 1–22, 2023.

[6] S. Nath *et al.*, "Transsizer: A novel transformer-based fast gate sizer," in *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, 2022, pp. 1–9.

[7] Y.-C. Lu *et al.*, "Lego-size: Llm-enhanced gpu-optimized signoff-accurate differentiable vlsi gate sizing in advanced nodes," in *Proceedings of the 2025 International Symposium on Physical Design*, 2025, pp. 152–162.

[8] J. Ahn *et al.*, "A heterogeneous graph-based gate sizer integrating graph attention network and transformer," in *2026 31st Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2026, pp. 1442–1448.

[9] X. Zhou *et al.*, "Heterogeneous graph neural network-based imitation learning for gate sizing acceleration," in *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, 2022, pp. 1–9.

[10] F. Imam *et al.*, "Parametric and nonparametric machine learning techniques for increasing power system reliability: a review," *Information*, vol. 15, no. 1, p. 37, 2024.

[11] X. Jiang *et al.*, "Ragraph: A general retrieval-augmented graph learning framework," *Advances in Neural Information Processing Systems*, vol. 37, pp. 29 948–29 985, 2024.

[12] P. Lewis *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in neural information processing systems*, vol. 33, pp. 9459–9474, 2020.

[13] Y. Gao *et al.*, "Retrieval-augmented generation for large language models: A survey," *arXiv preprint arXiv:2312.10997*, vol. 2, no. 1, p. 32, 2023.

[14] Y. Yu *et al.*, "Rankrag: Unifying context ranking with retrieval-augmented generation in llms," *Advances in Neural Information Processing Systems*, vol. 37, pp. 121 156–121 184, 2024.

[15] Y. Pu *et al.*, "Customized retrieval augmented generation and benchmarking for eda tool documentation qa," in *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, 2024, pp. 1–9.

[16] J. Hu *et al.*, "Sensitivity-guided metaheuristics for accurate discrete gate sizing," in *Proceedings of the International Conference on Computer-Aided Design*, 2012, pp. 233–239.

[17] K. Kasamsetty *et al.*, "A new class of convex functions for delay modeling and its application to the transistor sizing problem [cmos gates]," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 7, pp. 779–788, 2000.

[18] G. Flach *et al.*, "Effective method for simultaneous gate sizing and v th assignment using lagrangian relaxation," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 33, no. 4, pp. 546–557, 2014.

[19] A. Stefanidis *et al.*, "Autonomous application of netlist transformations inside lagrangian relaxation-based optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 8, pp. 1672–1686, 2020.

[20] F. Zhu *et al.*, "Retrieving and reading: A comprehensive survey on open-domain question answering," *arXiv preprint arXiv:2101.00774*, 2021.

[21] X. Li *et al.*, "Chain-of-knowledge: Grounding large language models via dynamic knowledge adapting over heterogeneous sources," *arXiv preprint arXiv:2305.13269*, 2023.

[22] P. Xu *et al.*, "Retrieval meets long context large language models," in *The Twelfth international conference on learning representations*, 2023.

[23] J. Kim *et al.*, "Sure: Summarizing retrievals using answer candidates for open-domain qa of llms," *arXiv preprint arXiv:2404.13081*, 2024.

[24] K. Ma *et al.*, "Chain-of-skills: A configurable model for open-domain question answering," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023, pp. 1599–1618.

[25] Y. Zhu *et al.*, "Retrieval-augmented embodied agents," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 17 985–17 995.

[26] W. Lin and B. Byrne, "Retrieval augmented visual question answering with outside knowledge," in *Proceedings of the 2022 conference on empirical methods in natural language processing*, 2022, pp. 11 238–11 254.

[27] Z. Cheng *et al.*, "Retrieval-augmented hypergraph for multimodal social media popularity prediction," in *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, 2024, pp. 445–455.

[28] B. Jin *et al.*, "Heterformer: Transformer-based deep node representation learning on heterogeneous text-rich networks," in *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*, 2023, pp. 1020–1031.

[29] X. Jiang *et al.*, "Hykge: A hypothesis knowledge graph enhanced rag framework for accurate and reliable medical llms responses," in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025, pp. 11 836–11 856.

[30] Q. Huang, H. Ren, and J. Leskovec, "Few-shot relational reasoning via connection subgraph pretraining," *Advances in neural information processing systems*, vol. 35, pp. 6397–6409, 2022.

[31] W. Fang *et al.*, "Nettag: A multimodal rtl-and-layout-aligned netlist foundation model via text-attributed graph," *arXiv preprint arXiv:2504.09260*, 2025.

[32] Y.-C. Lu, Z. Guo, K. Kunal, R. Liang, and H. Ren, "Insta: An ultra-fast, differentiable, statistical static timing analysis engine for industrial physical design applications," in *2025 62nd ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2025, pp. 1–7.

[33] Z. Wang *et al.*, "Restructure-tolerant timing prediction via multimodal fusion," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2023, pp. 1–6.

[34] X. Zhang *et al.*, "Disentangle, align and generalize: Learning a timing predictor from different technology nodes," in *Proceedings of the 61st ACM/IEEE Design Automation Conference*, 2024, pp. 1–6.

[35] Q. Wu *et al.*, "Sgformer: Simplifying and empowering transformers for large-graph representations," *Advances in Neural Information Processing Systems*, vol. 36, pp. 64 753–64 773, 2023.

[36] F. Fang *et al.*, "Enhancing noise robustness of retrieval-augmented language models with adaptive adversarial training," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024, pp. 10 028–10 039.

[37] F. Cuconasu *et al.*, "The power of noise: Redefining retrieval for rag systems," in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024, pp. 719–729.

[38] O. Yoran *et al.*, "Making retrieval-augmented language models robust to irrelevant context," *arXiv preprint arXiv:2310.01558*, 2023.

[39] C. Albrecht *et al.*, "Iwls 2005 benchmarks," in *International Workshop for Logic Synthesis (IWLS)*, vol. 9, 2005.

[40] OpenCores: The reference community for Free and Open Source gateware IP cores, https://opencores.org/.

[41] A. Amid *et al.*, "Chipyard: Integrated design, simulation, and implementation framework for custom socs," *Ieee Micro*, vol. 40, no. 4, pp. 10–21, 2020.

[42] Synopsys,"Design Compiler User Guide", https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/dc-ultra.html, 2024.

[43] Synopsys,"IC Compiler II User Guide", https://www.synopsys.com/implementation-and-signoff/physical-implementation/ic-compiler.html, 2024.

[44] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[45] P. Veličković *et al.*, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.