Learning Design Skills as Memory Policies for Agentic Photonic Inverse Design

Shengchao Chen [1] **Ting Shu** [2] **Sufen Ren** [3]

Abstract

Photonic crystal fiber (PCF) inverse design remains challenging because candidate geometries must satisfy coupled optical targets under expensive electromagnetic simulation. Existing pipelines improve surrogate prediction or one-shot parameter recommendation, but they do not accumulate reusable design knowledge across iterative trials. We formulate PCF inverse design as a memory-policy learning problem and propose SkillPCF, a closed-loop agent framework that combines a physics-guided memory skill bank, reinforcement-learned skill selection, and simulator-grounded skill evolution. We further construct a real-world dataset with 479 expert interaction traces (2,507 spans) and 553 memory-dependent evaluation queries covering dispersion engineering, loss optimization, and multi-objective design. Experiments across multiple LLM backbones and classical baselines show that SkillPCF achieves stronger design-quality and efficiency trade-offs under practical simulation budgets, demonstrating the effectiveness of our proposed memory-skill learning paradigm for physics-aware PCF inverse design.

1. Introduction

Photonic crystal fibers (PCFs) offer a wide design space for controlling dispersion, confinement loss, modal properties, and nonlinear response through microstructured cladding geometry (Russell, 2003; Knight et al., 1996; Birks et al., 1997; Markos et al., 2017). This flexibility is also the source of the core inverse-design difficulty: practical designs often require balancing multiple coupled targets over a high-dimensional parameter space, where each candidate evaluation may invoke expensive finite-element or finite-difference time-domain (FDTD) simulation (Chen et al., 2025). As a result, optimization loops of structure design can become simulation-bound and brittle under changing constraints.

Classical PCF studies established geometric principles such as endlessly single-mode guidance and cladding-induced mode control (Birks et al., 1997). Recent work introduces design acceleration, including solver-based numerical optimization (Gray et al., 2024) and machine learning-assisted dispersion-oriented inverse design (Chen et al., 2023; Wang et al., 2025b). These two paradigms are illustrated in Fig. 1 (left and middle): traditional numerical optimization relies on expert heuristics and exhaustive parameter sweeping, while ML-based approaches enable fast one-shot prediction. However, both paradigms treat each design run as independent. The former requires extensive domain expertise with no knowledge retention, and the latter lacks interpretability and iterative refinement. In realistic engineering sessions, designers repeatedly explore nearby regimes; what failed and why, or what worked under which constraints, is itself useful information that should be retained and reused.

This memory gap points to an opportunity: what if a PCF design system could learn to remember what works, forget what does not, and continuously refine its memory strategy from simulation feedback? To our knowledge, no prior work has equipped PCF inverse design with such adaptive memory capabilities. A recent work (Chen et al., 2025) explored using LLMs to provide structural understanding and design recommendations, but this approach remains limited to single-turn suggestion generation without iterative memory accumulation or policy learning. Beyond photonics, however, the broader LLM-agent community has made substantial progress on precisely this challenge: how to equip agents with memory systems that store, retrieve, and update information over long horizons (Schick et al., 2023; Yao et al., 2022; Park et al., 2023; Packer et al., 2023). Recent work further suggests that memory operations themselves, such as when to insert, update, or delete entries, can be treated as learnable policies rather than fixed heuristics (Shinn et al., 2023; Xu et al., 2026; Du et al., 2025), and that self-evolving agents can discover new memory skills through reinforcement learning (Zhang et al., 2026; Xia et al., 2026; Lu et al., 2026). These developments motivate a question for PCF design: ***Can PCF inverse design benefit from a memory policy that is learned, domain-conditioned,***

[1]AAII, University of Technology Sydney [2]School of Artificial Intelligence, Shenzhen University [3]School of Information and Communication Engineering, Hainan University. Correspondence to: Shengchao Chen <shengchao.chen.uts@gmail.com>.

Proceedings of the AI4Physics Workshop at the 43rd International Conference on Machine Learning (AI4Physics@ICML 2026), Seoul, South Korea. 2026. Copyright 2026 by the author(s).

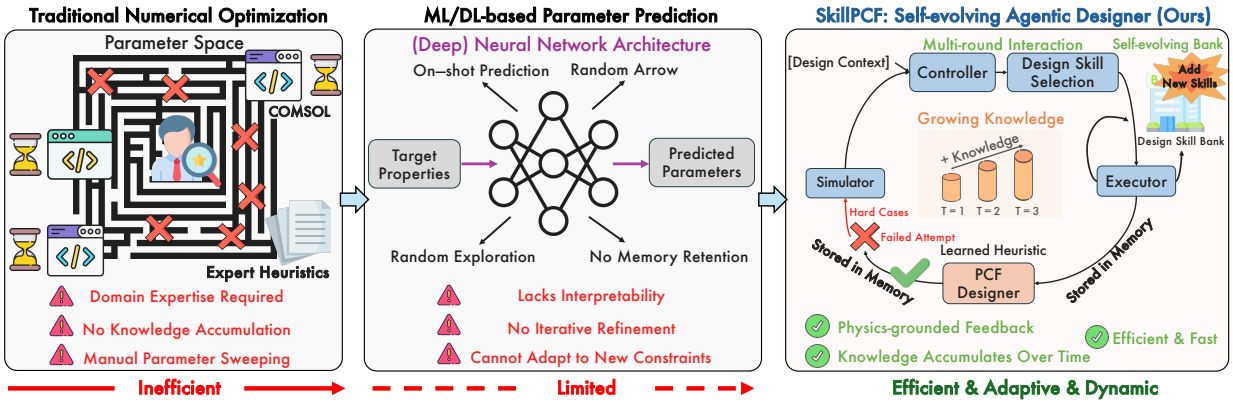

Figure 1. Comparison of PCF inverse design paradigms. **Left**: Traditional numerical optimization requires domain expertise and exhaustive simulation with no knowledge retention. **Middle**: ML-based approaches enable fast one-shot prediction but lack iterative refinement and interpretability. **Right**: SkillPCF (ours) treats design as a multi-round interaction with a self-evolving memory agent, accumulating knowledge across trials and refining skills based on physics-grounded feedback.

and continuously refined from simulation outcomes?

We argue so. Then we formulate PCF inverse design as a sequence of simulator-guided interactions, where the agent must decide not only what geometry to try next but also what to store, revise, or discard from prior trials. This perspective highlights two practical properties of the setting. First, electromagnetic solvers return deterministic metrics and constraint checks, yielding concrete feedback signals that can be used to study adaptive decision-making (Wen et al., 2025). Second, the long-horizon nature of iterative design suggests organizing decisions at multiple levels of abstraction, a viewpoint closely related to hierarchical control (Nachum et al., 2018; Dietterich, 2000).

Building on this perspective, we propose SkillPCF, a memory-centric agent framework for PCF inverse design (Figure 1, right). SkillPCF formulates design as a multi-round interaction process: a controller selects domain-specific skills, an executor applies them to update a structured memory bank, and a designer refines the skill bank from hard cases, forming a closed loop between physics simulation and memory policy learning. The framework integrates three core components: (i) a physics-guided skill bank for PCF-specific memory operations, (ii) a learnable skill selection controller, and (iii) simulation-grounded rewards within a simulator-in-the-loop training paradigm. To our knowledge, this is the first work to explicitly study a self-evolving LLM-agent memory policy for PCF inverse design. Our contributions are summarized as follows:

- We **first** rethink PCF inverse design as a memory-policy learning problem, where reusable design operations are treated as explicit actions optimized via **deterministic** simulator-grounded feedback.
- We propose SkillPCF, a closed-loop memory-centric framework integrating a physics-guided skill bank, a learnable skill-selection controller, and a designer module for skill evolution from hard cases.
- We construct a real-world dataset comprising expert interaction traces, evaluation queries, and multimodal artifacts spanning dispersion engineering, loss optimization, and multi-objective design.
- We demonstrate through experiments that SkillPCF improves design success and memory effectiveness under low simulation budgets, and that both physics-guided skills and skill evolution are critical.

2. Related Work

PCF Inverse Design. Foundational studies on PCFs established the waveguiding mechanisms and geometric controllability underlying modern PCF design (Russell, 2003; Knight, 2003; Knight et al., 1996; Birks et al., 1997; Chen et al., 2022b). PCF inverse design is commonly formulated as parameter optimization under highly nonlinear structure–response coupling (Gray et al., 2024). Traditional methods rely on repeated electromagnetic simulations over large parameter spaces. To improve efficiency, recent work introduces surrogate models and differentiable optimization pipelines to approximate structure–property mappings (Wang et al., 2025b; Gray et al., 2024; Chen et al., 2023; Ren et al., 2025). However, these approaches largely operate as parameter-level optimizers or one-shot predictors, without retaining reusable design experience across iterative trials. Rather than replacing physics solvers or optimization routines, we introduce a memory-centric layer that captures transferable design knowledge from multi-round interaction, enabling iterative refinement across PCF design sessions.

Self-evolving LLM Agents. Beyond photonics, the LLM-agent community has made substantial progress on mem-

ory systems for long-horizon tasks. Tool-augmented agents (Schick et al., 2023; Yao et al., 2022) and generative agent simulations (Park et al., 2023) showed that performance depends critically on how systems store and retrieve interaction history. MemGPT introduced OS-inspired memory hierarchies (Packer et al., 2023), while Reflexion demonstrated gains from explicit verbal feedback loops (Shinn et al., 2023). A key insight from recent work is that memory operations, such as when to insert, update, or delete, can be treated as learnable policies rather than fixed heuristics (Xu et al., 2026; Du et al., 2025), enabling agents to self-evolve their memory skills through reinforcement learning (Zhang et al., 2026; Xia et al., 2026; Lu et al., 2026). This evolution is grounded by verifiable rewards from deterministic environment feedback (Wen et al., 2025) and structured through hierarchical control (Nachum et al., 2018; Dietterich, 2000). We bring this self-evolving agent paradigm to PCF inverse design, coupling learnable memory skills with physics-grounded simulation feedback.

3. Method

Problem Formulation. Formally, PCF inverse design aims to find geometric parameters $p \in \mathcal{P}$ such that simulator outputs $f_{\text{sim}}(p)$ jointly satisfy target optical specifications $y^\star$ (like dispersion, confinement loss, and effective index) under practical structural constraints. The objective is to maximize target satisfaction while minimizing expensive electromagnetic simulation calls. This is difficult because these targets are coupled and each candidate evaluation requires costly physics simulation.

We propose SkillPCF (Figure 2), a framework that optimizes PCF design through two coupled processes: design skill usage and skill evolution. In the inner loop, a controller selects skills from a shared skill bank conditioned on the current design context, and an executor applies them to update trace-specific memory. In the outer loop, a designer refines existing skills and introduces new ones based on challenging cases encountered during training, enabling continual adaptation of the skill space. To separate trace-level knowledge from reusable memory operations, SkillPCF maintains two distinct stores: a trace-specific memory bank $\mathcal{M}$ and a shared skill bank $\mathcal{S}$. The former accumulates design knowledge for each trajectory, while the latter encodes reusable memory management skills across traces. At outer cycle e, we denote the active shared bank by $\mathcal{S}^{(e)}$ (fixed during the inner loop of that cycle). An electromagnetic simulator (Oskooi et al., 2010) provides physics-grounded rewards, forming a simulator-in-the-loop training paradigm. This decoupling of within-episode execution and cross-episode evolution enables stable policy learning while allowing structural adaptation of the skill space.

3.1. Physics-Guided Skill Bank

A memory skill defines a reusable, physics-grounded operation for updating design memory within the PCF inverse design loop. Each skill $s \in \mathcal{S}$ consists of (i) a short description used by the controller for selection, and (ii) a structured instruction template executed to produce memory edits. Because PCF inverse design is tightly coupled with electromagnetic simulation, each trajectory span corresponds to a geometry decision and its MEEP-derived physical outcome. Memory therefore stores unit-aware numerical evidence rather than abstract textual summaries. We record both trace-specific outcomes and cross-trace parameter–property relations to preserve physically consistent long-horizon optimization. The skill bank is initialized with four PCF-specific primitives: INSERTTOPOLOGYFEATURE, UPDATEPERFORMANCETREND, DELETEINVALIDASSUMPTION, and SKIP. This minimal set supports stable early learning while allowing subsequent specialization. The controller applies these skills at the span level, and the designer evolves the bank throughout training (see Sections 3.2 and 3.3); full definitions are provided in Appendix D.

3.2. Learning Memory Skills for Fine-grained Design

Controller: Design Skill Selection. To support fine-grained PCF design, we use a controller that selects a small set of memory skills at each span. We split each trajectory into contiguous spans and process them sequentially, where each span contains the current design action and its simulator context. Let $\mathcal{M}_t$ denote the current trace memory bank and $M_t = \text{Retrieve}(\mathcal{M}_t, x_t)$ the retrieved subset used at step t. The controller contains a state encoder f_{ctx} and a skill encoder f_{skill}. For span x_t and retrieved memory M_t, the controller computes:

$$h_t = f_{\text{ctx}}(x_t, M_t),\ f_{\text{ctx}}(\cdot) = \text{Embedding}_{\text{ctx}}(\cdot) \oplus \text{Embedding}_{\text{memory}}(\cdot). \tag{1}$$

where f_{ctx} concatenates a span embedding with a fused memory embedding (mean or similarity-weighted pooling). For each skill $s_i \in \mathcal{S}^{(e)}$ in the active bank, we compute a skill embedding from its description:

$$u_i = f_{\text{skill}}(\text{Description}(s_i)),\ f_{\text{skill}}(\cdot) = \text{Embedding}_{\text{skill}}(\cdot). \tag{2}$$

We use the same embedding model for f_{ctx} and f_{skill}, so PCF context states and skill descriptions are aligned in one representation space.

Compatibility with Evolving Skill Bank. During training, the designer may add, remove, or rewrite skills, so the action space changes over time. A fixed-size policy head would break under this evolution. Instead, we score each skill by matching state and skill embeddings as below:

$$z_{t,i} = h_t^\top u_i, \quad p_\theta(i \mid h_t) = \text{Softmax}(z_t)_i,$$

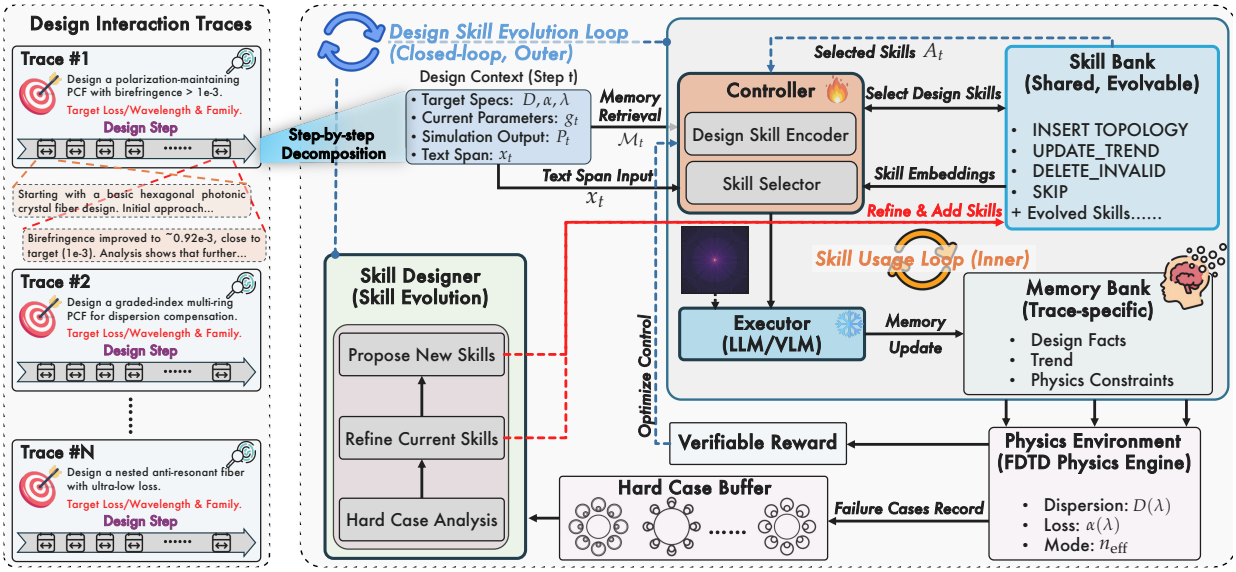

Figure 2. SkillPCF framework overview. Design traces are decomposed into step-wise contexts. At each step t, the Controller encodes the design state (target specs, parameters, simulation outputs, and text span) together with retrieved memories, and selects Top-K skills from the Skill Bank. The Executor applies them to update the trace-specific Memory Bank. A Physics Environment supplies simulation-grounded rewards based on dispersion, loss, and modal properties. The Skill Designer leverages hard cases from a failure buffer to refine and expand the skill bank. SkillPCF operates via an inner skill-usage loop (orange) and an outer skill-evolution loop (blue).

where $z_t \in \mathbb{R}^{|\mathcal{S}^{(e)}|}$ adapts to the current bank size.

Top-K Skill Selection. Given $p_\theta(i \mid h_t)$, the controller selects an ordered Top-K set $A_t = (a_{t,1}, \ldots, a_{t,K})$ without replacement via Gumbel-Top-K sampling (Kool et al., 2019) and passes only these skills to the executor. This is important in PCF traces, where one span may require compound updates (e.g., inserting a new parameter–property fact while deleting an invalid assumption).

Executor: Skill-Conditioned Memory Extraction. Given selected skills A_t, executor constructs memory edits by conditioning on (i) the current span x_t, (ii) retrieved memory M_t, (iii) selected skills A_t, and (iv) visual evidence V_t. The edits are structured and applied to the trace-specific memory bank:

$$\mathcal{M}_{t+1} = \text{Apply}\left(\mathcal{M}_t, \text{Executor}(x_t, M_t, A_t, V_t)\right). \quad (3)$$

Executing multiple skills in one call reduces repeated per-span processing. We optimize the controller with PPO (Schulman et al., 2017): for each trajectory, the controller produces a Top-K sequence, the executor builds memory incrementally, and reward is computed from downstream memory-dependent performance. Because actions are ordered and sampled without replacement, the policy probability is:

$$\pi_\theta(A_t \mid h_t) = \prod_{j=1}^{K} \frac{p_\theta(a_{t,j} \mid h_t)}{1 - \sum_{\ell<j} p_\theta(a_{t,\ell} \mid h_t)}, \quad (4)$$

which reduces to the usual single-action case when $K = 1$.

Delayed Reward Redistribution. In PCF inverse design, whether an early memory decision is useful is often known only after later geometry trials and final verification, which creates sparse credit assignment. In our implementation, the terminal reward is the episode-level QA performance, denoted by R_{final}. We redistribute this delayed signal over spans with exponential decay and keep an explicit terminal portion, the process can be formulated as:

$$\tilde{r}_t = (1-\beta)\, R_{\text{final}} \frac{\gamma^{T-t}}{\sum_{k=1}^{T} \gamma^{T-k}} + \beta\, \mathbf{1}[t{=}T] R_{\text{final}}, \quad (5)$$

where $\gamma \in (0,1)$ is the redistribution decay and $\beta \in [0,1]$ is the direct terminal ratio. The complete step reward is

$$r_t = r_{\text{proc},t} + \tilde{r}_t, \quad (6)$$

where $r_{\text{proc},t}$ is process-level shaping (including memory-construction quality and physics-grounded checks). This form matches the training pipeline and improves long-horizon credit assignment.

3.3. Feedback-driven Design Skill Evolution

Beyond fixed-skill selection, SkillPCF evolves its skill space through a designer module informed by hard PCF cases. At outer cycle e, we build a failure buffer $\mathcal{B}^{(e)}$ from low-performing or physically invalid trajectories, where each item contains target specs, predicted design, retrieved memory, and simulator feedback. To improve update diversity, cases in $\mathcal{B}^{(e)}$ are clustered by structural regime and optical-property failure type, and representative samples are se-

lected by difficulty. The designer then performs diagnosis (missing/misaligned operations) followed by refinement (edit existing skills or add structure-aware skills):

$$\hat{\mathcal{S}}^{(e+1)} = \text{Designer}\left(\mathcal{S}^{(e)}, \mathcal{B}^{(e)}\right). \tag{7}$$

Acceptance, rollback, and post-update exploration scheduling are handled in the closed-loop objective below.

3.4. Closed-Loop Optimization

We formulate training as an alternating optimization between controller θ and the evolving skill bank $\mathcal{S}$. At outer cycle e, with fixed skill bank $\mathcal{S}^{(e)}$, the controller is optimized on trajectories by maximizing cumulative step rewards:

$$\begin{aligned} \theta^{(e+1)} &= \arg\max_{\theta} J\left(\theta; \mathcal{S}^{(e)}\right), \\ J\left(\theta; \mathcal{S}^{(e)}\right) &= \mathbb{E}_{\tau \sim \pi_\theta(\cdot|\mathcal{S}^{(e)})}\left[\sum_{t=1}^{T} r_t\right]. \end{aligned} \tag{8}$$

From these rollouts, we mine hard cases (failed or low-margin designs) into a buffer $\mathcal{B}^{(e)}$, then let the designer propose an updated bank:

$$\hat{\mathcal{S}}^{(e+1)} = \text{Designer}\left(\mathcal{S}^{(e)}, \mathcal{B}^{(e)}\right). \tag{9}$$

To prevent drift from physically useful skills, we use acceptance with rollback:

$$\mathcal{S}^{(e+1)} = \begin{cases} \hat{\mathcal{S}}^{(e+1)}, & \Delta J_{\text{val}} \geq 0, \\ \mathcal{S}^{(e)}, & \text{otherwise}, \end{cases} \tag{10}$$

$$\Delta J_{\text{val}} = J_{\text{val}}\left(\theta^{(e+1)}, \hat{\mathcal{S}}^{(e+1)}\right) - J_{\text{val}}\left(\theta^{(e+1)}, \mathcal{S}^{(e)}\right). \tag{11}$$

After accepted updates, exploration is temporarily biased toward newly introduced design skills, matching the new-action bias mechanism used in training. This closed loop improves both span-level skill usage and cross-episode skill quality under PCF simulator-integrated physics feedback.

4. Real-World Expert-Annotated Dataset

To enable evaluation of memory-augmented agents for PCF inverse design, we construct a real-world benchmark dataset consisting of four complementary components based on real-world design trajectories and expert insights.

Interaction Traces. We collect 479 expert design trajectories spanning 8 PCF families (e.g., solid-core hexagonal, high-birefringence PM, hollow-core PBG, Kagome, anti-resonant ARF), as shown in Figure 3. Each trajectory is decomposed into spans (2,507 total, average 5.23 spans per trace), where each span contains a text chunk describing the design action, MEEP simulation code, and the simulation outcome. The dataset comprises approximately 393K tokens with 75.6% of traces achieving their design goals.

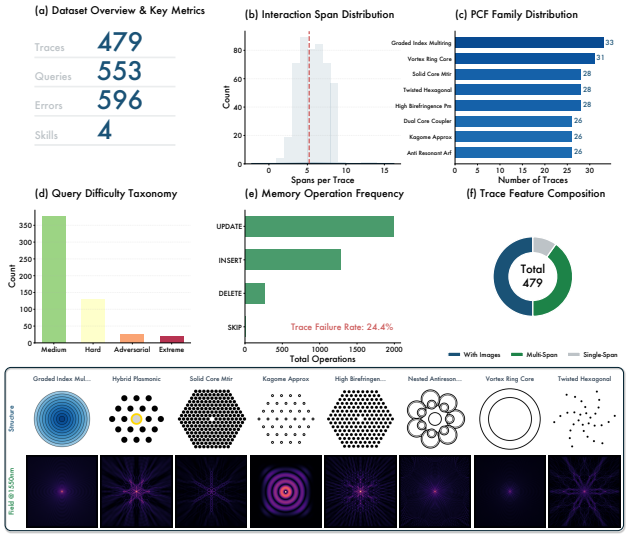

Figure 3. PCFSkill dataset, which contains 479 interaction traces across 8 PCF families, comprising 2,507 spans (5.23 per trace; 393K tokens) with a 75.6% design success rate, plus 553 memory-dependent evaluation queries and 596 error logs for skill evolution.

Evaluation Queries. We construct 553 memory-dependent queries with expert-annotated ground truth. Query types include design reasoning, trend prediction, visual-text fusion, parameter adjustment, multi-hop reasoning, cross-trace comparison, hallucination detection, and failure analysis. Difficulty distribution: medium (68%), hard (24%), extreme (4%), adversarial (5%). Each query requires 3.49 memory keys on average.

Error Logs. We collect 596 failure cases for skill evolution, categorized by error type: wrong trend (34%), missing constraint (31%), outdated knowledge (18%), and hallucination (17%). These logs enable the Designer to identify failure patterns and propose skill refinements.

Initial Skill Bank. We provide four PCF-specific memory skills with structured templates: `INSERT TOPOLOGY FEATURE` for capturing parameter-property mappings, `UPDATE PERFORMANCE TREND` for refining relationships, `DELETE INVALID ASSUMPTION` for removing incorrect heuristics, and `SKIP` for recognizing when no update is needed. Memory operation distribution across traces: INSERT (36%), UPDATE (56%), DELETE (5%), SKIP (¡1%). Details are in Appendix D.

5. Experiments and Results

Dataset and Baselines. We compare SkillPCF against baselines from two categories: (a) classical numerical optimization methods and (b) memory-augmented agents. Specifically, for classical numerical optimization, we include Random Search, Surrogate Neural Network (NN)

Table 1. Performance of Llama4-Scout (Touvron et al., 2023) under *Without Visual Field* and *With Visual Field*. Best: bold text with yellow shading; second-best: blue shading.

Method	Without Visual Field									With Visual Field								
	Human↑	F1↑	Judge↑	Design↑	Param↑	Trend↑	Succ.↑	Qual.↑	Phys.↑	Human↑	F1↑	Judge↑	Design↑	Param↑	Trend↑	Succ.↑	Qual.↑	Phys.↑
RAG	7.05	17.19	7.06	20.68	60.83	69.39	32.65	5.13	64.29	6.92	19.59	6.36	20.08	55.40	71.43	23.47	6.21	26.53
CoN	6.95	14.01	7.47	22.97	57.10	70.41	16.33	3.90	64.29	6.88	16.91	6.59	21.32	52.89	68.37	19.90	4.60	20.41
ReadAgent	6.93	19.45	5.62	19.42	55.85	70.91	37.78	6.18	51.47	6.95	18.67	5.34	19.47	54.46	73.47	40.82	5.72	47.96
MemoryBank	7.18	17.65	6.72	20.85	62.15	69.39	30.61	5.58	45.92	6.98	20.40	5.71	18.57	57.69	71.43	26.02	5.73	26.53
Mem0	5.85	18.92	5.86	19.78	42.75	70.41	3.06	1.19	6.12	5.90	18.84	5.95	18.61	43.59	66.33	6.63	4.14	5.10
A-MEM	6.92	20.13	6.02	19.04	58.03	68.37	36.22	7.07	38.78	6.85	20.54	5.91	18.98	56.80	69.39	29.08	7.66	31.63
LangMem	6.85	20.17	5.72	19.03	58.94	71.43	38.27	6.69	40.82	6.82	20.24	6.05	18.20	58.45	70.41	36.22	7.19	32.65
MemoryOS	6.80	20.08	5.87	19.68	56.48	67.35	29.08	12.59	37.76	6.88	20.01	5.97	19.40	52.20	68.37	31.63	14.44	44.90
SkillPCF (Ours)	**8.47**	**22.35**	**8.02**	**23.45**	**67.28**	**82.86**	**60.12**	**48.73**	**68.92**	**8.80**	**24.18**	**7.22**	**23.12**	**65.85**	**82.99**	**61.22**	**52.35**	**71.85**

Predictor (4 MLP-layers with BN (Ioffe & Szegedy, 2015) and ReLU), and Nelder-Mead (Lagarias et al., 1998). For memory-augmented baselines, we include RAG (Retrieval-Augmented Generation) (Lewis et al., 2020), CoN (Chain-of-Note) (Yu et al., 2024), ReadAgent (Lee et al., 2024), MemoryBank (Zhong et al., 2024), Mem0 (Chhikara et al., 2025), A-MEM (Xu et al., 2026), LangMem, and MemoryOS (Kang et al., 2025). Details are in Appendix C.2.

Evaluation Protocol and Metrics. We adopt a multi-level evaluation protocol assessing design quality, simulator cost, and memory effectiveness under identical backbone configurations and fixed trajectory-level splits. Data are partitioned using family-aware stratification (70/15/15 train/validation/test), and retrieval indices for memory baselines are constructed exclusively from training trajectories. At test time, responses are generated with retrieved memories when applicable. If a response proposes a geometry modification or candidate design, we invoke MEEP for deterministic physical verification. We define one simulation call as one full MEEP verification run and report calls per query using this unified definition across all methods. We report metrics along four families that match the column abbreviations in Tables 1–3: *text-based* (token-overlap **F1**, LLM judge score **Judge**); *design reasoning* (concept-coverage **Design**, parameter accuracy **Param**, trend agreement **Trend**); *inverse design* (success rate **Succ.**, design quality **Qual.**, physics verification **Phys.**); and *human expert score* (**Human**, 0–10). Physics verification and design success serve as primary criteria, with the LLM judge used as a supplementary signal. Formal definitions are provided in Appendix C.1.

Implementation Details. We implement SkillPCF in PyTorch (Paszke et al., 2019) with a PPO-based controller and a simulator-in-the-loop executor. GPT-4o-mini (Hurst et al., 2024) serves as the LLM judge, Text-Embedding-3-Small encodes states and operations, and Contriever retrieves memories with depth $k = 5$. The skill bank is initialized with four operation types (insert, update, delete, no-op), then expanded by the designer during outer-loop evolution. The controller uses an MLP with hidden size 256 and is optimized with AdamW (Loshchilov & Hutter, 2017) (1×10^{-4}). PPO follows standard settings ($\gamma = 0.99$, $\lambda = 0.95$, clip 0.2, entropy 0.01), using four epochs per update, minibatch size 32, and gradient clipping 0.5. Training runs for 10 outer evolution epochs and 50 inner interaction epochs with batch size 32. LLM API calls are routed through OpenRouter; decoding is deterministic with temperature 0 and maximum length 2048. We use a fixed seed for data split and training reproducibility and provide run scripts/configuration in the artifact package. Controller training runs use an NVIDIA RTX A100-40GB GPU. For efficiency comparison, we report Calls/q as a hardware-agnostic simulator-cost proxy.

5.1. Main Results

Tables 1 and 2 show that SkillPCF delivers the strongest overall performance among memory-agent methods while retaining low simulation cost. Under Llama4-Scout, SkillPCF ranks first across all reported metrics in both settings (without/with visual field). Cross-backbone results further confirm robustness: SkillPCF remains best on MiniMax-M2.5 and Qwen2.5-72b, and a constant 1.02 calls per query. Other memory-agent baselines operate at similar call budgets (typically 1.10–1.20) but consistently lower Success and Human scores, indicating that gains come from higher-quality memory policy rather than extra simulator usage. Classical optimizers can reach high raw Success under exhaustive search, yet they require 100–135 calls per query and provide markedly weaker reasoning quality, which limits practicality in interactive design workflows.

5.2. Framework Analysis

Hyperparameter Sensitivity. Figure 4 shows that SkillPCF remains stable under moderate hyperparameter variation around the default configuration. Across retrieval depth, learning rate, entropy, designer frequency, operation-bank size, and action top-k, performance typically varies within $\pm8\%$ of baseline. Noticeable degradation occurs only under overly large learning rates or insufficient exploration regularization. The default sits in a broad stable region, indicating hyperparameter robustness.

Ablation Studies. Table 3 shows that performance gains arise from complementary and non-redundant components. Removing the skill controller leads to the largest Success drop and reduced Human score, highlighting the importance of learned operation selection. Removing physics reward

Table 2. Performance across executor backbones. Calls/q ↓ is per-query simulator usage (function evaluations for classical optimizers; verification calls for memory agents). Best/second-best highlighted.

Method	Human Score↑	MiniMax-M2.5 (Li et al., 2025) F1	Judge	Design	Param	Trend	Succ.	Qual.	Phys.	Calls/q↓	Qwen2.5-72B (Bai et al., 2023) F1	Judge	Design	Param	Trend	Succ.	Qual.	Phys.	Calls/q↓
Memory-Augmented Agents																			
RAG	7.10	22.41	6.08	19.76	55.62	68.37	42.86	6.88	57.14	1.13†	21.77	5.87	20.20	56.08	68.37	35.71	5.54	54.08	1.13
CoN	6.98	20.12	6.21	20.58	54.91	66.33	40.82	7.21	55.10	1.12†	19.15	6.03	21.90	56.77	65.31	35.71	6.58	54.08	1.12
ReadAgent	6.86	23.67	5.96	18.92	56.45	69.39	37.76	7.06	49.98	1.13†	22.34	5.58	19.04	56.23	68.37	33.16	6.50	46.94	1.13
MemoryBank	7.22	22.58	6.18	19.85	55.93	70.41	46.94	9.12	56.63	1.12†	21.05	6.11	19.46	54.83	69.39	43.67	8.66	54.08	1.12
Mem0	5.92	21.16	6.39	17.31	42.38	71.43	3.06	0.98	16.33	1.03	22.07	6.30	18.43	47.44	68.37	12.76	3.01	39.80	1.11
A-MEM	6.88	22.98	6.02	19.11	53.51	67.35	50.00	8.22	48.98	1.15	21.21	6.00	20.79	57.12	70.41	41.33	8.95	52.04	1.10
LangMem	6.79	23.05	5.44	17.16	54.91	69.39	38.78	8.56	44.90	1.20	21.59	6.01	19.87	55.41	67.35	41.33	6.77	57.14	1.17
MemoryOS	6.83	23.26	5.46	19.11	54.91	67.35	38.78	14.76	55.10	1.15	20.55	5.68	18.12	54.47	67.35	24.49	8.97	60.20	1.15
SkillPCF (Ours)	**9.12**	**32.85**	**6.92**	**24.35**	**62.24**	**84.85**	82.35	52.18	**68.45**	**1.02**	**25.12**	**7.95**	**29.85**	**69.35**	**72.45**	78.92	48.76	**65.28**	**1.02**
Classical Optimization† († They are backbone-independent and do not rely on external LLMs, thereby maintaining consistency across backbone configurations)																			
RandomSearch	6.05	15.00	1.60	12.90	11.50	19.40	**92.90**	7.30	64.30	100.00	15.00	1.60	12.90	11.50	19.40	**92.90**	7.30	64.30	100.00
NNPredictor	4.62	13.60	1.80	12.70	15.70	19.40	0.00	0.00	63.30	100.00	13.60	1.80	12.70	15.70	19.40	0.00	0.00	63.30	100.00
NelderMead	6.42	12.20	2.40	12.40	31.40	19.40	91.80	**81.60**	64.30	135.00	12.20	2.40	12.40	31.40	19.40	91.80	**81.60**	64.30	135.00

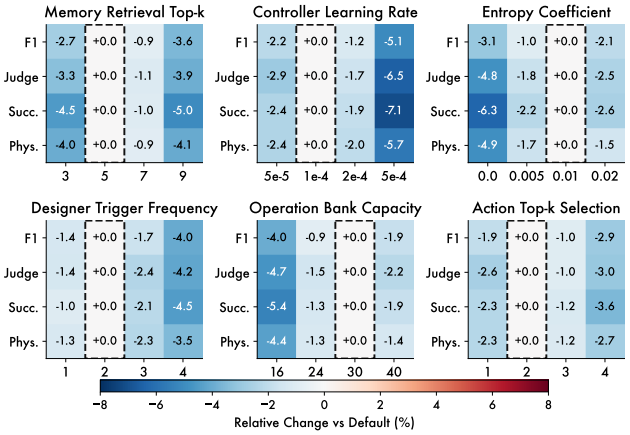

Figure 4. Hyperparameter sensitivity. Cells report relative (%) changes from default settings across F1, Judge, Success, and Physics metrics; dashed columns denote defaults.

shaping results in the strongest PhysicsV decline, underscoring the role of simulator-grounded rewards in physical validity. Disabling designer evolution or reward redistribution further reduces F1, Success, and Human scores, though to a lesser extent. Visual intervention helps physics-grounded metrics (F1, Succ., Phys.) but slightly lowers LLM-judged Design/Judge, which we attribute to the judge's weaker calibration on mode-field images. We therefore treat physics-grounded metrics as primary.

Scaling Behaviors. Figure 5 evaluates executor scaling from Qwen2.5-VL 3B to 72B with the controller and skill bank fixed. All metrics improve monotonically as model size increases, with diminishing gains beyond 32B. The 32B model already attains performance close to 72B, indicating a favorable quality–cost trade-off at mid scale. Most performance gains are therefore captured before the largest model sizes, suggesting efficient scaling behavior.

Memory Dynamics under Skill Evolution. Figure 6 reports two statistics illustrating how hard-case feedback reshapes memory across outer-loop stages. The left panel shows monotonic growth in memory size at selected checkpoints, while the right panel reflects increased domain relevance through keyword frequency and operation usage. These results suggest that designer-triggered updates improve not only memory quantity but also its physical informativeness. Examples of refined and newly introduced skills are provided in Appendix D.

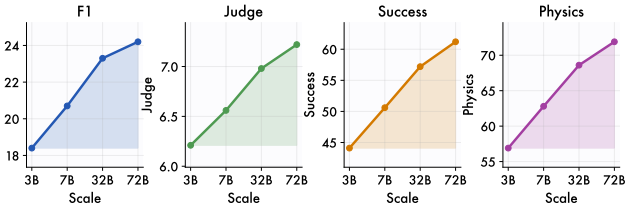

Figure 5. Scaling under Qwen2.5-VL executor (3B/7B/32B/72B).

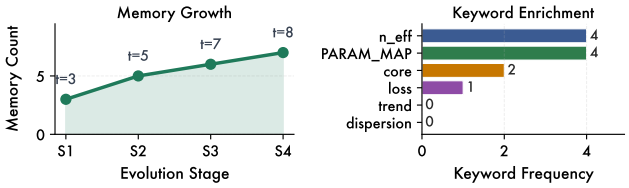

Figure 6. Memory dynamics during skill evolution. Left: memory growth across outer-loop. Right: keyword enrichment.

5.3. Case Studies

We conduct a case study on six challenging PCF design tasks (Figure 7). SkillPCF satisfies target specifications on five of six families, with simulated mode fields exhibiting clear confinement and losses meeting stringent thresholds (e.g., $\leq 10^{-8}$ dB/km for nested antiresonant); the remaining case (C5, fractal quasicrystal) is missed by 0.93 dB/km and exposes a hard regime where the current skill bank lacks fractal-symmetry priors. In contrast, baselines show sharper sensitivity to difficulty, succeeding only under relaxed targets and failing under stricter constraints such as graded-index multiring at 1.6×10^{-12} dB/km. Memory-based methods use comparable budgets (1.1–1.3 calls/query); gaps reflect memory quality and skill selection, not simulator usage. Per-family quantitative breakdowns and extended qualitative inspection are provided in Appendix F.

Table 3. Ablation on Llama4-Scout (visual). Best/second-best highlighted in yellow/blue.

Variant	F1 ↑	Judge ↑	Design ↑	Param ↑	Trend ↑	Succ ↑	Qual ↑	PhysV ↑	Human ↑
SkillPCF (Full)	**24.18**	7.22	23.12	65.85	**82.99**	**61.22**	**52.35**	**71.85**	**8.80**
w/o Designer Evolution	22.94	6.89	21.76	63.10	80.71	54.36	41.82	66.34	7.85
w/o Physics Reward Shaping	23.36	7.03	22.41	64.27	81.38	49.28	38.47	58.92	7.53
w/o Adaptive Skill Controller	21.88	6.74	20.95	61.90	79.66	47.55	35.41	61.08	7.28
w/o New-Action Bias	23.57	7.06	22.66	64.88	82.05	56.74	46.29	69.14	8.18
w/o Reward Redistribution	22.81	6.93	21.89	63.52	80.97	53.11	43.38	65.27	7.67
w/o Visual Intervention	22.35	**8.02**	**23.45**	**67.28**	82.86	60.12	48.73	68.92	8.05

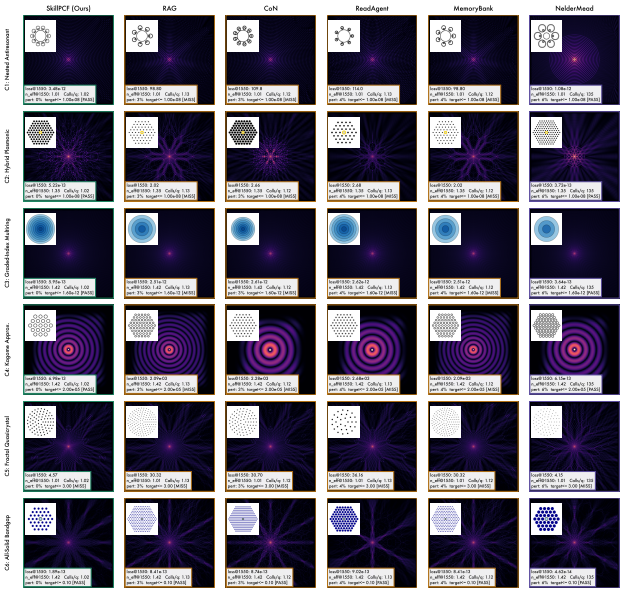

Figure 7. Case study across six PCF families. Each panel: mode field (background), structure (inset), key metrics (loss, n_{eff}, calls/-query). **PASS/FAIL** badges mark target satisfaction. Best viewed zoomed in and in color. Extended studies in Appendix F.

Hard-Case Test. Figure 8 stresses methods on ultra-low-loss targets ($< 10^{-8}$ dB/km), zero-dispersion designs, and novel architectures (rows H1–H4) alongside standard telecom cases (M1–M2). Baselines mostly fail on H1–H4 with isolated passes; SkillPCF passes consistently across difficulty levels under matched simulation budgets, indicating that memory accumulation translates into genuine physical-design improvements rather than higher solver effort.

Zero-Shot Generalization. Figure 9 evaluates four *novel* PCF families never seen in training (chiral-twisted hexagonal, dual-core asymmetric coupler, LMA Yb-doped, SC-generation flat). SkillPCF transfers consistently while baselines show only sporadic passes, suggesting that learned skills capture transferable physics relationships such as confinement, dispersion control, and mode engineering, rather than parameter correlations specific to seen families.

6. Conclusion

We propose SkillPCF, a memory-centric framework for PCF inverse design that models reusable design experience as a learnable policy. By integrating skill selection, memory updates, and simulator verification, it transforms iterative traces into transferable optimization knowledge. Results show strong target satisfaction with high simulation efficiency, supporting physics-aware design automation.

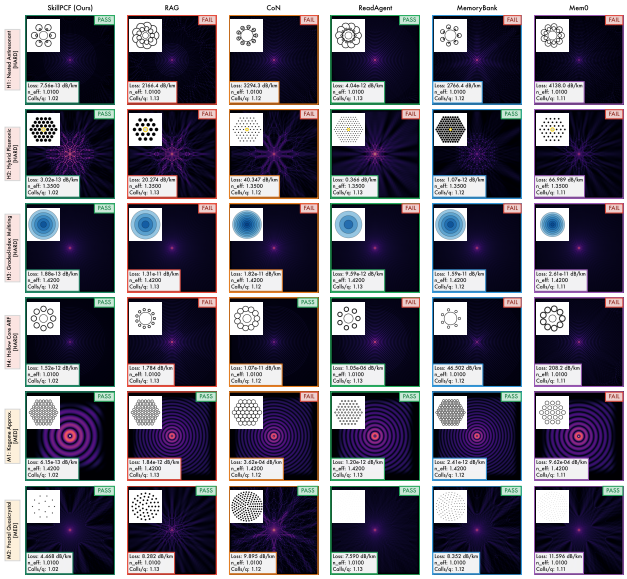

Figure 8. Hard-case stress test with real MEEP simulations across six PCF scenarios. Rows are design cases (H: hard, M: medium); columns are methods. Best viewed zoomed in and in color.

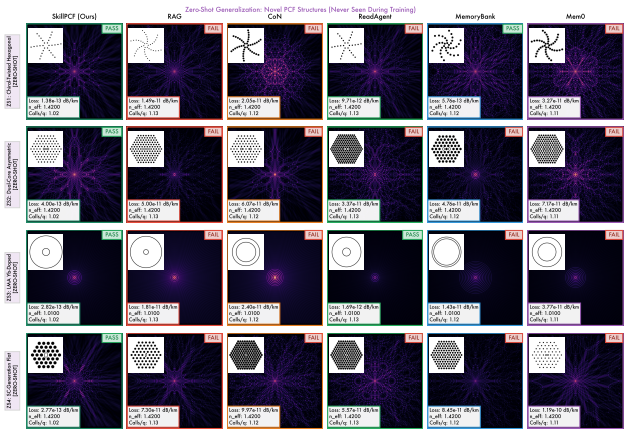

Figure 9. Zero-shot generalization to four novel PCF families never seen in training. Rows: ZS1 chiral-twisted hexagonal; ZS2 dual-core asymmetric coupler; ZS3 LMA Yb-doped; ZS4 SC-generation flat. Best viewed zoomed in and in color.

Impact Statement

SkillPCF reduces the compute and human effort of PCF inverse design by reusing physics-grounded design experience across simulation budgets, with downstream relevance to optical-fiber R&D in telecommunications, sensing, and biomedical imaging, and to other simulator-grounded design domains such as metamaterials and antennas. The principal risk is over-reliance on language-model-suggested geometries, mitigated by deterministic MEEP verification of every candidate and by releasing queries, rubrics, and run scripts. Dependence on hosted foundation models is addressed via fixed seeds, stratified splits, and full configuration disclosure. We do not foresee further societal consequences warranting specific discussion.

References

Bai, J., Bai, S., Chu, Y., Cui, Z., Dang, K., Deng, X., Fan, Y., Ge, W., Han, Y., Huang, F., et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Birks, T. A., Knight, J. C., and Russell, P. S. J. Endlessly single-mode photonic crystal fiber. *Optics letters*, 22(13): 961–963, 1997.

Chen, S., Ren, S., Yang, J., Yao, F., Yang, Q., Wang, L., Wang, G., and Huang, M. Reconstruction of fabry-perot interferometric sensor spectrum from extremely sparse sampling points using dense neural network. *IEEE Photonics Technology Letters*, 34(24):1337–1340, 2022a.

Chen, S., Yao, F., Ren, S., Wang, G., and Huang, M. Cost-effective improvement of the performance of awg-based fbg wavelength interrogation via a cascaded neural network. *Optics Express*, 30(5):7647–7663, 2022b.

Chen, S., Yao, F., Ren, S., Yang, J., Yang, Q., Yuan, S., Wang, G., and Huang, M. Fabry-perot interferometric sensor demodulation system utilizing multi-peak wavelength tracking and neural network algorithm. *Optics express*, 30(14):24461–24480, 2022c.

Chen, S., Wang, X., Ren, S., Yang, J., Zhang, Y., and Wang, G. Collaborative photonic crystal fiber property optimization: A new paradigm for reverse design. *IEEE Photonics Technology Letters*, 35(19):1035–1038, 2023.

Chen, S., Hu, G., Ren, S., and Shu, T. Pcf-llm: Scaling llms for multimodal understanding of structured scientific data in photonic crystal fiber sensors. *Information Fusion*, pp. 104022, 2025.

Chhikara, P., Khant, D., Aryan, S., Singh, T., and Yadav, D. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

Dietterich, T. G. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of artificial intelligence research*, 13:227–303, 2000.

Du, Y., Huang, W., Zheng, D., Wang, Z., Montella, S., Lapata, M., Wong, K.-F., and Pan, J. Z. Rethinking memory in llm based agents: Representations, operations, and emerging topics. *arXiv preprint arXiv:2505.00675*, 2025.

Gray, D., West, G. N., and Ram, R. J. Inverse design for waveguide dispersion with a differentiable mode solver. *Optics Express*, 32(17):30541–30554, 2024.

Hou, X., Ren, S., Yu, K., Hu, Y., Xu, H., Xue, C., Chen, S., and Wang, G. Federated learning-based wavelength demodulation system for multi-point distributed multi-peak fbg sensors. *Optics Express*, 32(23):41297–41313, 2024.

Hurst, A., Lerer, A., Goucher, A. P., Perelman, A., Ramesh, A., Clark, A., Ostrow, A., Welihinda, A., Hayes, A., Radford, A., et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. pmlr, 2015.

Kang, J., Ji, M., Zhao, Z., and Bai, T. Memory os of ai agent. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 25972–25981, 2025.

Knight, J., Birks, T., Russell, P. S. J., and Atkin, D. All-silica single-mode optical fiber with photonic crystal cladding. *Optics letters*, 21(19):1547–1549, 1996.

Knight, J. C. Photonic crystal fibres. *nature*, 424(6950): 847–851, 2003.

Kool, W., Van Hoof, H., and Welling, M. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. In *International conference on machine learning*, pp. 3499–3508. PMLR, 2019.

Lagarias, J. C., Reeds, J. A., Wright, M. H., and Wright, P. E. Convergence properties of the nelder–mead simplex method in low dimensions. *SIAM Journal on optimization*, 9(1):112–147, 1998.

Lee, K.-H., Chen, X., Furuta, H., Canny, J., and Fischer, I. A human-inspired reading agent with gist memory of very long contexts. *arXiv preprint arXiv:2402.09727*, 2024.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

Li, A., Gong, B., Yang, B., Shan, B., Liu, C., Zhu, C., Zhang, C., Guo, C., Chen, D., Li, D., et al. Minimax-01: Scaling foundation models with lightning attention. *arXiv preprint arXiv:2501.08313*, 2025.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Lu, M., Wu, M., Liu, F., Xu, J., Li, W., Wang, H., Hu, Z., Ding, Y., Sun, Y., Lu, J., et al. Choosing how to remember: Adaptive memory structures for llm agents. *arXiv preprint arXiv:2602.14038*, 2026.

Markos, C., Travers, J. C., Abdolvand, A., Eggleton, B. J., and Bang, O. Hybrid photonic-crystal fiber. *Reviews of Modern Physics*, 89(4):045003, 2017.

Nachum, O., Gu, S. S., Lee, H., and Levine, S. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

Oskooi, A. F., Roundy, D., Ibanescu, M., Bermel, P., Joannopoulos, J. D., and Johnson, S. G. Meep: A flexible free-software package for electromagnetic simulations by the fdtd method. *Computer Physics Communications*, 181(3):687–702, 2010.

Packer, C., Fang, V., Patil, S., Lin, K., Wooders, S., and Gonzalez, J. Memgpt: towards llms as operating systems., 2023.

Park, J. S., O'Brien, J., Cai, C. J., Morris, M. R., Liang, P., and Bernstein, M. S. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pp. 1–22, 2023.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Ren, S., Chen, S., Xu, H., Hou, X., Yang, Q., Wang, G., and Shen, C. Unsupervised representation learning-based spectrum reconstruction for demodulation of fabry–pérot interferometer sensor. *IEEE Sensors Journal*, 23(12): 13810–13816, 2023.

Ren, S., Chen, S., Hu, Y., Xu, H., Hou, X., Yang, Q., Wang, G., and Zhang, Y. Rethink training: Synthetic data powers efficient fiber-optics sensor demodulation neural networks. *IEEE Sensors Journal*, 24(19):31408–31416, 2024a.

Ren, S., Chen, S., Wang, J., Xu, H., Hou, X., Huang, M., Liu, J., and Wang, G. A distributed photonic crystal fiber reverse design framework based on multi-source knowledge fusion. *Optical Fiber Technology*, 84:103718, 2024b.

Ren, S., Hu, Y., Chen, Y., Shi, H., Wang, G., and Chen, S. Unbalanced data learning for improved optical property prediction in porous core fibers. *IEEE Sensors Journal*, 2025.

Russell, P. Photonic crystal fibers. *science*, 299(5605): 358–362, 2003.

Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Wang, G., Liu, J., Chen, S., and Ren, S. Optimizing low-resolution spectral demodulation for long-period fiber gratings using residual convolutional neural networks. *Optics Express*, 33(4):8225–8238, 2025a.

Wang, G., Liu, J., Chen, S., and Ren, S. Towards scalable and accurate property prediction for photonic crystal fibers with federated learning. *Optics Express*, 33(16): 33421–33434, 2025b.

Wen, X., Liu, Z., Zheng, S., Ye, S., Wu, Z., Wang, Y., Xu, Z., Liang, X., Li, J., Miao, Z., et al. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base llms. *arXiv preprint arXiv:2506.14245*, 2025.

Xia, P., Chen, J., Wang, H., Liu, J., Zeng, K., Wang, Y., Han, S., Zhou, Y., Zhao, X., Chen, H., et al. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*, 2026.

Xu, W., Liang, Z., Mei, K., Gao, H., Tan, J., and Zhang, Y. A-mem: Agentic memory for llm agents. *Advances in Neural Information Processing Systems*, 38:17577–17604, 2026.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. React: Synergizing reasoning and acting in language models. 2022.

Yu, W., Zhang, H., Pan, X., Cao, P., Ma, K., Li, J., Wang, H., and Yu, D. Chain-of-note: Enhancing robustness in retrieval-augmented language models. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 14672–14685, 2024.

Yuan, S., Chen, S., Yang, J., Yang, Q., Ren, S., Wang, G., and Yu, B. Efficient calculation of optical properties of suspended-core fiber via a machine learning algorithm. *Applied Optics*, 61(19):5714–5721, 2022.

Zhang, H., Long, Q., Bao, J., Feng, T., Zhang, W., Yue, H., and Wang, W. Memskill: Learning and evolving memory skills for self-evolving agents. *arXiv preprint arXiv:2602.02474*, 2026.

Zhong, W., Guo, L., Gao, Q., Ye, H., and Wang, Y. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 19724–19731, 2024.

Appendix

For readability and fast lookup, we organize the appendix into six blocks with direct hyperlinks:

- Appendix A: System, Data, and Evaluation Protocols
- Appendix B: Additional Case Studies
- Appendix C: Implementation Details and Evaluation Metrics
- Appendix D: Initial PCF Memory Skills
- Appendix E: Skill Evolution Examples During Training
- Appendix F: Extended Case Study Analysis

Background. Photonic crystal fibers sit within a broader trend of applying machine learning to fiber-optic and photonic systems, where data-driven models have improved tasks ranging from wavelength interrogation and interferometric sensor demodulation to spectral reconstruction from sparse measurements and efficient calculation of optical properties for microstructured fibers (Chen et al., 2022b;c;a; Hou et al., 2024; Ren et al., 2023; 2024a; Wang et al., 2025a; Yuan et al., 2022). These efforts established that learned surrogates can replace or accelerate expensive optical characterization, and the perspective has recently been extended to PCF reverse design through distributed knowledge-fusion frameworks (Ren et al., 2024b). SkillPCF continues this trajectory but shifts the locus of learning from one-shot prediction toward a memory policy that accumulates and reuses design experience across iterative, simulator-grounded trials.

A. System, Data, and Evaluation Protocols

A.1. Front-end System Overview

For the ease of user interaction, we design a front-end system for our SkillPCF-based PCF inverse design. Our interactive design platform integrates real-time electromagnetic simulation with memory-augmented design reasoning, as illustrated in Figure 10. The system architecture comprises two synchronized panels:

Left panel (Physics + Skill Evolution Analysis). The left side contains two coordinated modules. The upper module provides direct access to the MEEP simulation backend with interactive parameter controls for all PCF geometric variables (air-hole diameters d_i, lattice pitch Λ, core defect configurations), along with synchronized structure/field visualization and metric feedback (effective index n_{eff}, confinement loss α, dispersion D). The lower module visualizes skill evolution trajectory analysis across optimization rounds, exposing which memory skills are activated, refined, or deprecated as solver feedback accumulates.

Right panel (Memory-Augmented Reasoning). Exposes the complete SkillPCF reasoning pipeline. The design-context input module accepts natural language task specifications (e.g., "design PCF with near-zero dispersion at 1550 nm"). The retrieved-memory display shows relevant prior design knowledge extracted from the skill bank, including parameter-property mappings, constraint records, and successful strategy patterns. The generated output module presents the agent's design recommendations with explicit reasoning traces. Post-generation verification validates proposed designs against target specifications and feeds outcomes back to both memory updates and trajectory analytics.

The platform supports iterative design workflows where users can accept, modify, or reject agent suggestions, with all interactions logged for benchmark construction and skill evolution analysis. All system code, trained models, and benchmark datasets will be made publicly available upon paper acceptance to facilitate reproducibility and community research.

A.2. Expert Team and Dataset Construction Protocol

The expert team composition is detailed in Table 4. Dataset construction followed a three-stage protocol: (i) trajectory drafting from realistic inverse-design tasks, (ii) simulation-backed verification with unit and constraint checks, and (iii) independent audit of annotations and query-ground-truth mappings. Ambiguous traces were returned for revision until both physical consistency and textual clarity passed review, yielding a reliable benchmark for memory-dependent reasoning. For evaluation governance and scoring anchors, see Table 5 and Table 6.

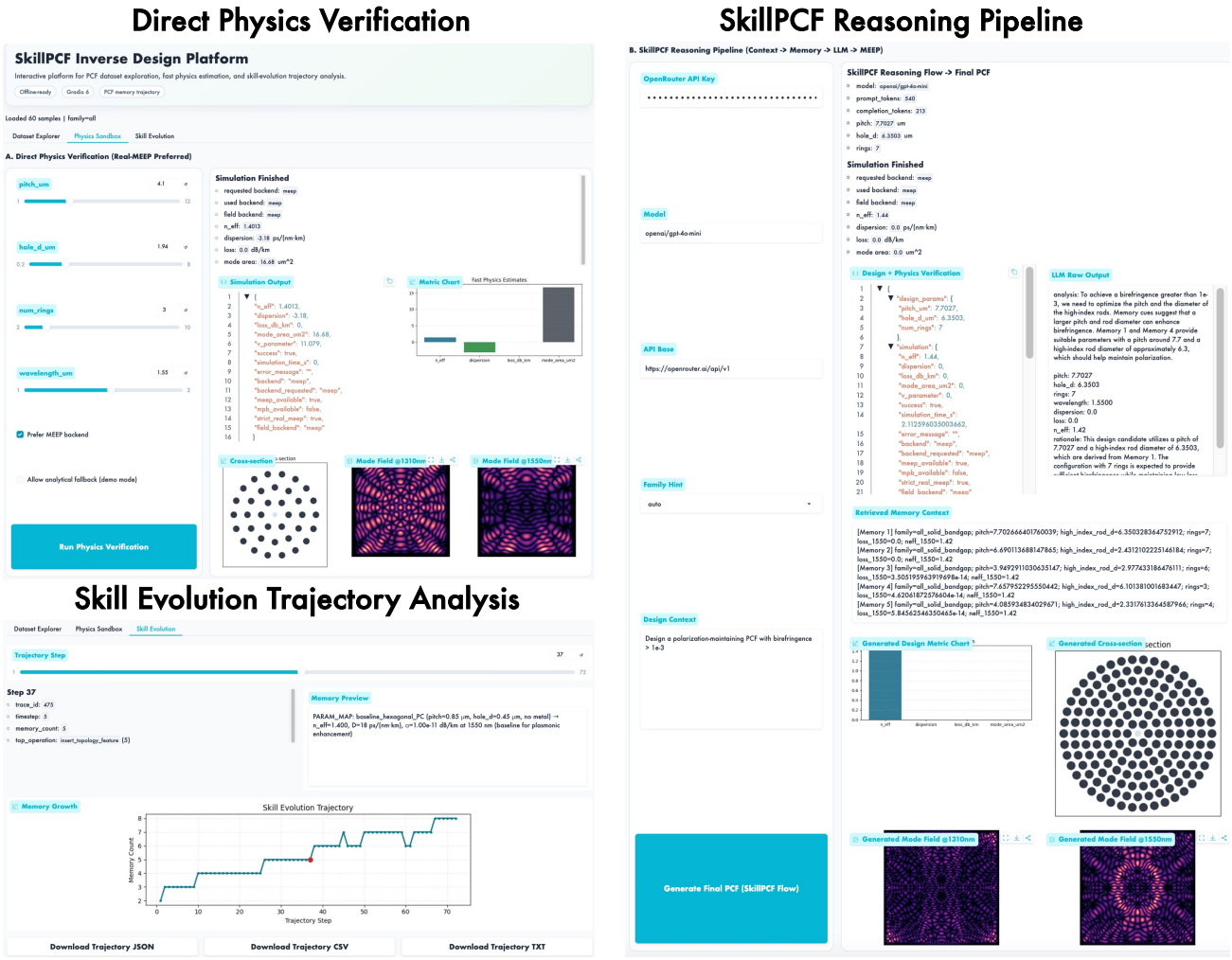

Figure 10. Interactive SkillPCF design platform. Left: coupled physics verification and skill-evolution trajectory analysis, including parameter controls, synchronized structure/field, and round-level skill updates. Right: memory-augmented reasoning pipeline with design-context input, memory retrieval, generation, and verification feedback.

Table 4. Expert composition used in dataset construction.

Role	Count	Background and responsibility
PCF design researchers	4	PhD-level photonics background (4–10 years); authored design trajectories, defined target specifications, and validated physical plausibility.
Simulation specialists	2	Electromagnetic simulation experience with MEEP/MPB (3–8 years); audited solver settings, unit consistency, and convergence checks.
Annotation auditors	2	Research staff with optical-device data curation experience (2–5 years); normalized span annotations, query labels, and cross-check logs.

A.3. Human-Evaluation Governance Summary

To avoid fragmented reporting, we consolidate full human-evaluation protocol details (workflow, fairness controls, and scoring rubric) with the metric definitions in Appendix C. This keeps all quantitative evaluation standards in one place while preserving dataset-construction context.

B. Additional Case Studies

This section provides supplementary qualitative evidence complementary to the main case-study analysis.

B.1. Classical-Budget Comparison

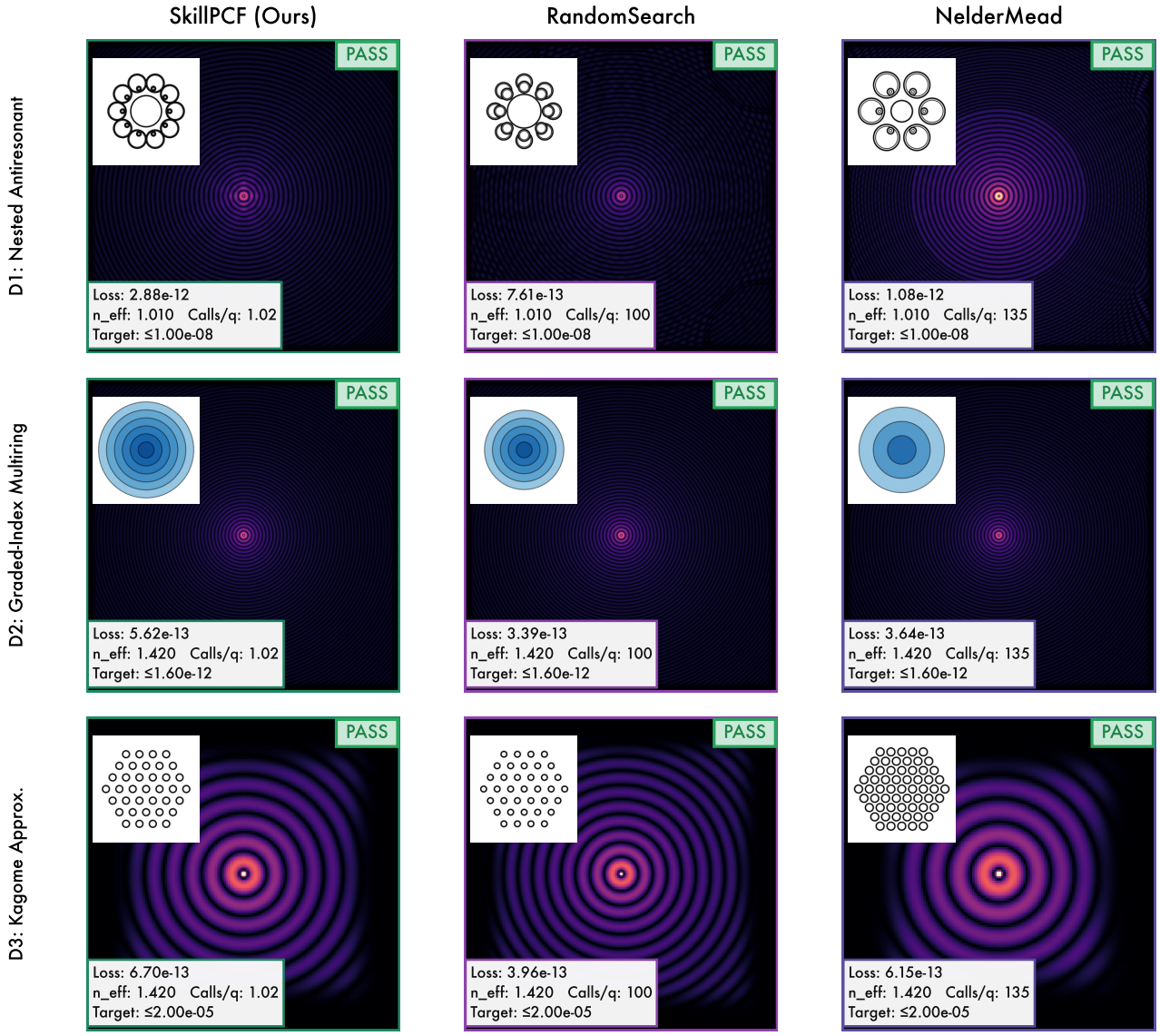

Figure 11. Efficiency comparison between SkillPCF and classical optimization methods.

Figure 11 further compares SkillPCF against classical optimization baselines. While RandomSearch and NelderMead achieve competitive success rates (92.9% and 91.8% respectively), they require approximately 100–135 simulation calls per query, two orders of magnitude more than SkillPCF. This efficiency gap is critical for practical PCF design, where each electromagnetic simulation may take minutes to hours depending on structure complexity. The results demonstrate that SkillPCF offers a favorable trade-off: near-classical design quality at memory-agent efficiency.

B.2. Additional Visual Evidence

Figure 12 complements scalar metrics by exposing geometry–field correspondence in challenging families. The left column provides the generated micro-structure, while the two right columns show wavelength-specific field patterns under the same design. Across rows, the visual modes remain centered and physically coherent at both wavelengths, with family-dependent intensity distributions that match distinct guidance mechanisms. This qualitative evidence supports that SkillPCF outputs are not only numerically competitive but also visually consistent with expected PCF behavior under multi-wavelength verification.

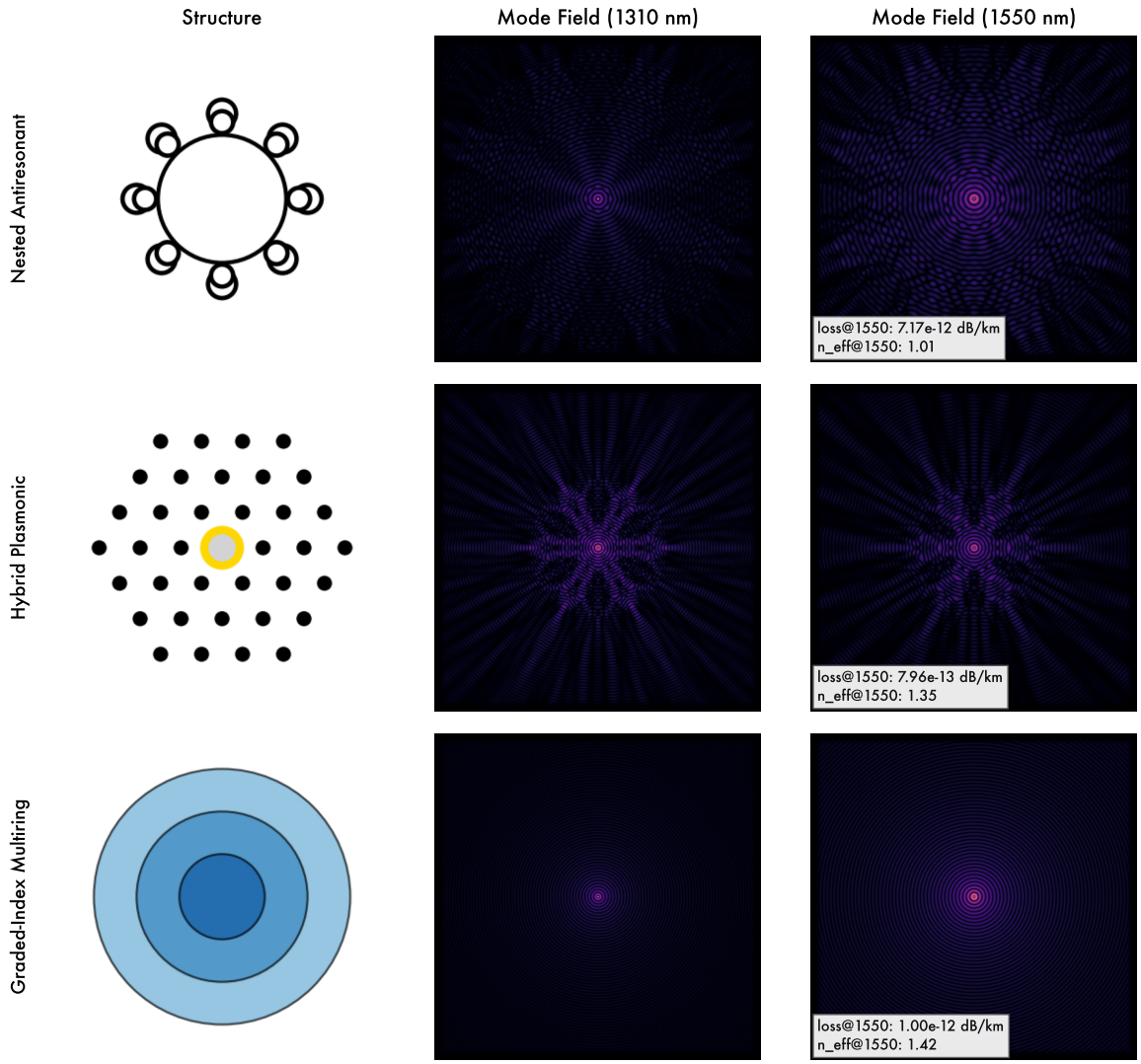

Figure 12. Complex-family visual evidence for generated designs. Each row corresponds to one challenging PCF family (nested antiresonant, hybrid plasmonic, graded-index multiring).

C. Implementation Details and Evaluation Metrics

C.1. Evaluation Metrics, Human Scoring, and LLM Judge Protocol

We employ a multi-level evaluation framework spanning text-based, reasoning, inverse design, physics, and human metrics. To keep evaluation reporting self-contained, human-evaluation workflow controls are summarized in Table 5, and scoring anchors are provided in Table 6.

Text-based metrics. **Token-overlap F1** computes the harmonic mean of precision and recall over token overlap:

$$F1 = \frac{2PR}{P+R}, \tag{12}$$

$$P = \frac{|\text{pred_tokens} \cap \text{true_tokens}|}{|\text{pred_tokens}|}, \quad R = \frac{|\text{pred_tokens} \cap \text{true_tokens}|}{|\text{true_tokens}|}. \tag{13}$$

LLM judge score uses a 0–10 rubric-based evaluation:

$$S_{\text{judge}} = \text{LLM}(\text{prompt}_{\text{eval}}(q, y_{\text{true}}, y_{\text{pred}})), \tag{14}$$

Table 5. Human-score evaluation workflow and fairness controls.

Stage	Protocol
Panel setup	Three independent PCF experts (all with prior simulation experience) scored each sampled result; evaluators were not authors of the corresponding generated outputs.
Calibration	Before formal scoring, evaluators completed a shared calibration round on pilot samples to align interpretation of utility/plausibility/feasibility anchors.
Blind scoring	Model identities and method names were hidden; sample order was randomized; each evaluator scored independently using the fixed rubric.
Quality control	Pairwise disagreement > 2.0 on any dimension triggered adjudication; adjudication logs recorded rationale and final consensus.
Aggregation	Final human score was reported as the mean across evaluators after adjudication, with the same protocol applied to all compared methods.

with criteria for technical accuracy (0–4), completeness (0–3), clarity (0–2), and relevance (0–1). In the main paper, we treat LLM-judge scores as supporting evidence and prioritize physics-grounded metrics (Success, Quality, PhysicsV) for primary conclusions.

LLM Judge Prompt Template (Used in Evaluation)

System: You are a strict photonic crystal fiber (PCF) reviewer. Evaluate the predicted answer against the reference answer for technical correctness and design usefulness.
Input fields:

- `Question`: Design query and target constraints.
- `GroundTruth`: Reference answer from expert-verified trace.
- `Prediction`: Model output to be scored.

Scoring rubric:

- Technical Accuracy: 0–4
- Completeness: 0–3
- Clarity: 0–2
- Relevance: 0–1

Output format: Return strict JSON {`"technical_accuracy":int, "completeness":int, "clarity":int, "relevance":int, "total":int, "rationale":str`}
Constraints: Do not use external assumptions beyond given question and reference. Penalize physically implausible claims and missing constraint handling.

Design reasoning metrics. **Design reasoning score** measures whether the prediction mentions key design concepts K:

$$S_{\text{reason}} = \frac{1}{|K|} \sum_{k \in K} \mathbf{1}[k \in y_{\text{pred}}], \tag{15}$$

where $K = \{\text{pitch, hole_d, rings, dispersion, loss, wavelength, n_eff}\}$. **Parameter accuracy** matches predicted vs. true parameters within 10% relative error:

$$A_{\text{param}} = \frac{1}{|P|} \sum_{p \in P} \text{Match}(p_{\text{pred}}, p_{\text{true}}), \tag{16}$$

$$\text{Match}(a, b) = \mathbf{1}[|a - b| < 0.1\,|b|]. \tag{17}$$

Trend accuracy checks sign agreement:

$$A_{\text{trend}} = \frac{1}{|T|} \sum_{t \in T} \mathbf{1}[\text{sign}(\Delta t_{\text{pred}}) = \text{sign}(\Delta t_{\text{true}})]. \tag{18}$$

Inverse design metrics. **Design success rate** is the fraction of designs meeting tolerances:

$$\text{Succ} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}[|D_{\text{pred}} - D_{\text{target}}| < \tau_D \wedge |\alpha_{\text{pred}} - \alpha_{\text{target}}| < \tau_\alpha], \tag{19}$$

with $\tau_D = 5\ \text{ps}/(\text{nm}\cdot\text{km})$ and $\tau_\alpha = 10^{-3}\ \text{dB/km}$. **Design quality** is a normalized distance-to-target score:

$$Q = 1 - \frac{1}{2}\left(\frac{|D_{\text{pred}} - D_{\text{target}}|}{|D_{\text{target}}| + \epsilon} + \frac{|\alpha_{\text{pred}} - \alpha_{\text{target}}|}{|\alpha_{\text{target}}| + \epsilon}\right). \tag{20}$$

Physics verification rate measures solver-verified correctness:

$$R_{\text{phys}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}[\text{verify}(y_i, S_{\text{target},i}) = \text{True}]. \tag{21}$$

Human expert score. We define a 0–10 human score H that combines design utility, physical plausibility, and implementability:

$$H = U + P + I, \tag{22}$$

where $U \in [0, 4]$, $P \in [0, 4]$, and $I \in [0, 2]$. Process-level fairness controls (panel setup, blinding, adjudication) are documented in Table 5; Table 6 gives the scoring anchors.

Table 6. Human scoring rubric used for the 0–10 expert score.

Dim.	Score	Low	Mid	High
Utility (U)	0–4	Does not address target goals; suggestions are not actionable.	Partially useful guidance, but key design choices are missing.	Clear and actionable parameter choices with explicit trade-offs.
Plausibility (P)	0–4	Contradicts PCF physics or violates simulation constraints.	Mostly plausible but weakly grounded or partially inconsistent.	Fully consistent with electromagnetic principles and simulation evidence.
Feasibility (I)	0–2	Cannot be executed in the current workflow.	Executable only after moderate manual correction.	Directly executable with coherent geometry, solver settings, and constraints.
Total	0–10	Final score is $H = U + P + I$. Reported value is the mean across experts; disagreement > 2.0 on any dimension triggers adjudication.		

C.2. Baseline Overview

We compare SkillPCF against representative baselines spanning classical optimization (including surrogate modeling), and recent memory-augmented agent (including retrieval-augmented generation) architectures. For each baseline we also list the concrete operating setup we used in our comparison so that the reported Calls/q and metric values are interpretable.

- **Random Search.** Uniformly samples PCF parameter configurations within per-family bounds without learned priors or adaptive memory; we cap the budget at 100 verified candidates per query and report the best satisfying-target structure (or the closest one if none satisfies).
- **NN Predictor.** A four-layer MLP (BatchNorm + ReLU) trained on the training-split MEEP outputs to predict $(n_{\text{eff}}, \alpha, D)$ from geometry; at test time we sample 100 candidates in the surrogate, rank by predicted target match, and verify the top-1 with a single MEEP run, so Calls/q is dominated by surrogate sweep rather than verification.
- **Nelder–Mead.** Simplex-based derivative-free optimization with standard reflection / expansion / contraction updates and a hard cap of 135 MEEP-verified evaluations per query.
- **RAG.** Retrieval-augmented generation that conditions an LLM on top-k retrieved memory entries from training trajectories ($k = 5$, Contriever).

- **CoN.** Chain-of-Note reasoning that produces intermediate notes before a final answer, with the same retrieval setup as RAG.
- **ReadAgent.** Long-context reading with gist-style summarization plus selective lookup, with retrieval index built only from the training split.
- **MemoryBank.** Persistent memory with explicit insert / update / forgetting dynamics, with a memory budget matched to ours.
- **Mem0.** Persistent memory APIs at user / session level with default retrieval and update policies.
- **A-MEM.** Adaptive memory whose updates are driven by task interactions, using the same backbone and retrieval depth as ours.
- **LangMem.** Semantic / episodic / procedural memory abstractions exposed to a tool-using agent, with the same backbone and retrieval depth.
- **MemoryOS.** Hierarchical memory modules coordinated by an OS-like layer, with the same backbone and retrieval depth.

Text-metric computation for non-text baselines. For Random Search, NN Predictor, and Nelder–Mead, the parameter outputs are first formatted into a templated answer string of the form "parameters p yield $(n_{\text{eff}}, \alpha, D)$ at wavelength λ" and then scored against the ground-truth answer; this lets token-overlap F1 and design-reasoning metrics be computed on a common textual surface across all methods.

D. Initial PCF Memory Skills

We provide the complete definitions of the four initial memory skills in the physics-guided skill bank. Each skill follows a structured template with purpose, trigger conditions, application instructions, and constraints.

Skill: InsertTopologyFeature

Purpose: Capture physical constraints, geometric parameters, or parameter-property mappings discovered from simulation results.
When to use:

- Geometric parameter adjustment (d, Λ, d_i) caused specific optical property changes (dispersion shift, loss variation)
- Discovered performance bottleneck or boundary condition
- Found successful design pattern worth preserving
- Identified new constraint on parameter ranges

How to apply:

- Identify specific geometric action (e.g., "increased d_1 from 1.0 to 1.2 μm")
- Map to physical result (e.g., "dispersion shifted from +15 to -5 ps/(nm·km)")
- Store as structured memory: "PARAM_MAP: $d_1 \uparrow \rightarrow D \downarrow$ (sensitive near $d/\Lambda = 0.6$)"
- Include wavelength context and numerical precision

Constraints:

- Ignore simulation steps without significant property changes
- Do not duplicate existing memory entries
- Ensure numerical precision is appropriate (3–4 significant figures)
- Always include units for physical parameters

Action type: INSERT

Skill: UpdatePerformanceTrend

Purpose: Refine existing parameter-property relationship based on new simulation evidence.
When to use:

- New results contradict or refine previous trend assumptions
- Discovered boundary conditions where trends change
- Found more precise numerical ranges for relationships
- Identified non-linear behavior in previously assumed linear region

How to apply:

- Locate relevant existing memory entry
- Merge new evidence while preserving valid previous knowledge
- Example: Update "$D \sim d_1$" to "$D \sim d_1$ (linear for $d_1 < 1.2$ μm, saturates above)"
- Add confidence level based on number of supporting simulations

Constraints:

- Only modify existing entries, do not create new ones
- Preserve physical details that remain accurate
- Always cite the new evidence that triggered the update

Action type: UPDATE

Skill: DeleteInvalidAssumption

Purpose: Remove design heuristics proven incorrect by simulation.
When to use:

- Clear evidence that a design direction is invalid
- Previous assumption contradicted by physical data
- Exhausted parameter range without improvement
- Identified fundamental physics violation in assumed relationship

How to apply:

- Confirm evidence is definitive (simulation failure, extreme degradation)
- Remove corresponding memory entry to prevent future misguidance
- Document the specific failure mode for future reference

Constraints:

- Do not delete for mere suboptimal results; only for proven invalidity
- Prefer UPDATE over DELETE when partial validity remains
- Always preserve the deletion reason in a separate audit log

Action type: DELETE

Skill: Skip

Purpose: Recognize when current span requires no memory update.
When to use:

- Routine execution logs without design insights
- Redundant information already captured in memory
- Inconsequential parameter variations within noise threshold
- Preliminary exploration steps before meaningful results

How to apply:

- Briefly note why no memory update is needed
- Continue to next span without memory modification

Constraints:

- Only use when all other skills are clearly inapplicable
- Do not skip spans that contain even minor but novel insights
- Be conservative: better to over-capture than under-capture

Action type: NOOP

E. Skill Evolution Examples During Training

This section provides explicit examples of how the designer evolves the initial skill bank during training. We separate two categories: (i) *refine/renew* edits that improve existing skills, and (ii) *newly added* skills discovered from repeated hard cases. All examples follow the same template style as the initial bank for direct comparability. The examples below are consistent with the hard-case buffer mechanism in Section 3.3, where edits are triggered only after repeated solver-backed evidence.

E.1. Refined and Renewed Existing Skills

Refined Skill 1: InsertTopologyFeature-v2 (Evidence-Gated)

Origin: Refined from INSERTTOPOLOGYFEATURE to avoid over-insertion from weak spans.
Purpose: Insert topology memories only when metric change exceeds a significance threshold.
When to use:

- Parameter edits produce clear shifts in loss/dispersion/neff.
- The same directional effect appears in neighboring spans.

How to apply:

- Compute absolute and relative metric deltas from previous verified span.
- Insert entry only if deltas pass pre-defined significance checks.

Action type: INSERT

Refined Skill 2: InsertTopologyFeature-v3 (Wavelength-Scoped Mapping)

Origin: Refined from INSERTTOPOLOGYFEATURE after conflicts between 1310 nm and 1550 nm trends.
Purpose: Bind inserted parameter–property mappings to explicit wavelength scope.
When to use:

- A trend holds at one wavelength but weakens at another.

- Cross-wavelength behavior is central to current objective.

How to apply:

- Store mapping with wavelength tag and unit-consistent metric values.
- Split broad mappings into wavelength-conditioned subentries.

Action type: INSERT

Refined Skill 3: InsertTopologyFeature-v4 (Fabrication-Aware Tagging)

Origin: Refined from INSERTTOPOLOGYFEATURE to track manufacturability constraints.
Purpose: Add fabrication-relevant tags (minimum feature size, spacing feasibility) to inserted memories.
When to use:

- Candidate geometry is near practical fabrication boundaries.
- Multiple high-performing designs differ primarily in manufacturability.

How to apply:

- Attach compact fabrication tags to each inserted geometry memory.
- Preserve optical-performance context with the same entry.

Action type: INSERT

Refined Skill 4: UpdatePerformanceTrend-v2 (Regime-Aware Trend)

Origin: Refined from UPDATEPERFORMANCETREND after repeated non-linear trend failures in high-sensitivity parameter regions.
Purpose: Update parameter–property memories with *regime-aware* trend descriptions instead of single global trends.
What changed from initial version:

- Adds explicit operating regimes (e.g., low-d/Λ, transition zone, saturation zone).
- Requires confidence tagging by support count (number of corroborating spans).
- Requires preserving previous trend statements when still valid in disjoint regimes.

When to use:

- A previously linear trend is contradicted in a new local region.
- Different PCF families show consistent but regime-shifted behavior.
- Simulator outputs indicate turning points or saturation effects.

How to apply:

- Partition observed evidence into regimes using explicit parameter ranges.
- Rewrite memory in piecewise form with units and validity ranges.
- Attach support count and representative trace references.

Action type: UPDATE

Refined Skill 5: UpdatePerformanceTrend-v3 (Uncertainty-Calibrated)

Origin: Refined from UPDATEPERFORMANCETREND to quantify confidence under noisy neighborhoods.
Purpose: Update trends with explicit confidence levels tied to support size and local variance.
When to use:

- New trend evidence is directionally consistent but noisy.
- Trend update depends on sparse observations.

How to apply:

- Estimate support count and local spread of observed deltas.
- Write confidence-calibrated update instead of binary statement.

Action type: UPDATE

Refined Skill 6: UpdatePerformanceTrend-v4 (Cross-Family Normalization)

Origin: Refined from UPDATEPERFORMANCETREND for transfer between structurally different families.
Purpose: Normalize trend expressions before cross-family reuse.
When to use:

- Similar behavior appears in two families with different parameter scales.
- Direct trend transfer causes systematic over/under-estimation.

How to apply:

- Convert raw parameters to normalized ratios (e.g., d/Λ).
- Update memory with normalized trend plus family-specific offsets.

Action type: UPDATE

Refined Skill 7: DeleteInvalidAssumption-v2 (Cross-Verified Deletion)

Origin: Refined from DELETEINVALIDASSUMPTION to reduce over-deletion under noisy or weak evidence.
Purpose: Remove invalid assumptions only when failure evidence is *cross-verified* and causally attributable.
What changed from initial version:

- Adds a two-stage deletion gate: evidence sufficiency check + causality check.
- Introduces soft deprecation path before hard deletion for borderline cases.
- Requires preserving a compact failure rationale for future retrieval.

When to use:

- The same assumption fails across multiple traces or nearby settings.
- Failures persist after controlling confounding parameter changes.

How to apply:

- Verify repeated failure with at least two independent supporting traces.
- If evidence is partial, mark as deprecated and request more verification.
- If evidence is definitive, delete and store concise causal-failure note.

Action type: DELETE

Refined Skill 8: DeleteInvalidAssumption-v3 (Rollback-Safe)

Origin: Refined from DELETEINVALIDASSUMPTION after observing occasional deletion of partially valid heuristics.
Purpose: Support reversible deletion with lightweight rollback metadata.
When to use:

- Evidence indicates likely invalidity but regime boundaries are uncertain.
- Deletion may affect multiple downstream memory entries.

How to apply:

- Archive deleted entry with trigger evidence and timestamp.
- Allow controlled restoration if later spans contradict deletion.

Action type: DELETE

Refined Skill 9: Skip-v2 (Noise-Threshold Gate)

Origin: Refined from SKIP to prevent memory churn from micro-perturbations.
Purpose: Skip updates when parameter changes are below sensitivity threshold.
When to use:

- Metric change is within simulation noise tolerance.
- Span contains bookkeeping without new physical insight.

How to apply:

- Compare deltas against calibrated noise bounds.
- Emit concise skip rationale for auditability.

Action type: NOOP

Refined Skill 10: Skip-v3 (Deferred Consolidation)

Origin: Refined from SKIP for multi-step explorations where insight emerges only after short span batches.
Purpose: Defer isolated updates and consolidate them after a short evidence window.
When to use:

- Individual spans are weak, but a 2–3 span bundle forms a clear pattern.
- Immediate insertion would create fragmented memory entries.

How to apply:

- Mark spans for temporary hold.
- Convert to one consolidated update once pattern is confirmed.

Action type: NOOP

E.2. Newly Added Skills Discovered in Training

New Skill 1: InsertCrossTraceTransferPattern

Discovery signal: Hard-case buffer repeatedly showed that successful patterns in one family were reusable in related families after minor parameter remapping.
Purpose: Insert transferable design motifs that connect cross-family parameter priors and expected metric responses.
When to use:

- A pattern (e.g., pitch increase + moderate ring count) succeeds across multiple families.
- Direct transfer fails, but affine-like remapping of key parameters restores success.

How to apply:

- Record source-family pattern with metric context and validity window.

- Add target-family remapping hints (e.g., scaling factor or offset).
- Store expected directional effects on dispersion/loss under mapped settings.

Constraints:

- Do not claim transfer without at least one validated target-family trace.
- Always include both source and target family identifiers.

Action type: INSERT

New Skill 2: DeferUntilPhysicsVerified

Discovery signal: Designer observed frequent low-value updates from speculative intermediate reasoning before solver confirmation.
Purpose: Delay memory writes for provisional hypotheses until key physical checks are verified, reducing noisy memory growth.
When to use:

- Current span proposes a hypothesis not yet supported by simulation output.
- Recent history contains conflicting provisional updates for the same key.

How to apply:

- Mark hypothesis as pending without writing permanent memory entries.
- Trigger update only after solver-backed confirmation in subsequent spans.
- If verification fails, route to deletion/deprecation instead of insertion.

Constraints:

- Use only for unverified hypotheses; do not delay clearly validated evidence.
- Limit defer horizon to avoid indefinite postponement.

Action type: NOOP-STYLE CONTROL

New Skill 3: InsertFailureBoundaryMap

Discovery signal: Repeated hard cases clustered near narrow invalid regions in parameter space.
Purpose: Insert explicit failure boundaries to avoid repeatedly sampling known invalid zones.
When to use:

- Multiple nearby designs fail the same physics constraint.
- Failure onset is sharp with respect to one dominant parameter.

How to apply:

- Store boundary interval with constraint type and confidence.
- Link boundary to representative failed traces.

Action type: INSERT

New Skill 4: InsertSensitivityHotspot

Discovery signal: Some regions showed disproportionately large metric changes for tiny parameter perturbations.
Purpose: Mark high-sensitivity hotspots for cautious exploration.
When to use:

- Local gradient-like behavior is steep and unstable.

- Same hotspot appears across independent trajectories.

How to apply:

- Record hotspot center, radius, and dominant affected metrics.
- Add recommendation for reduced step size in that region.

Action type: INSERT

New Skill 5: UpdateTradeoffFrontier

Discovery signal: Hard cases repeatedly required balancing loss vs. dispersion rather than optimizing one metric alone.
Purpose: Update memory with Pareto-style trade-off frontier fragments.
When to use:

- New candidate improves one target while slightly degrading another.
- No single-point improvement exists under current constraints.

How to apply:

- Append frontier point with both metric values and geometry context.
- Retire dominated points in the same local regime.

Action type: UPDATE

New Skill 6: UpdateConstraintPriority

Discovery signal: Different query intents implied different acceptable trade-offs among objectives.
Purpose: Update objective-priority memory for context-aware optimization.
When to use:

- Task instruction explicitly prioritizes one metric.
- Historical success indicates family-specific priority ordering.

How to apply:

- Store ranked objective list with trigger context.
- Reweight retrieval relevance using current priority profile.

Action type: UPDATE

New Skill 7: MergeDuplicateMemories

Discovery signal: Long training created semantically duplicated memories with minor textual variation.
Purpose: Merge duplicate entries while preserving strongest evidence.
When to use:

- Retrieved memories show near-identical parameter–property mapping.
- Duplicate entries cause retrieval crowding.

How to apply:

- Keep highest-confidence core statement.
- Aggregate supporting trace references into a merged entry.

Action type: UPDATE

New Skill 8: DeleteConfoundedEvidence

Discovery signal: Some trend memories were formed from spans where multiple parameters changed simultaneously.
Purpose: Delete memories whose causal attribution is confounded and unreliable.
When to use:

- No dominant parameter explains the metric change.
- Follow-up controlled spans contradict the original attribution.

How to apply:

- Flag confounded entry and attach contradiction evidence.
- Remove entry and request controlled re-evaluation span.

Action type: DELETE

New Skill 9: DeleteStaleContextWindow

Discovery signal: In long trajectories, outdated context memories degraded retrieval relevance.
Purpose: Delete stale context windows that no longer match active optimization regime.
When to use:

- Current objective has shifted to a different constraint region.
- Old context repeatedly retrieves but fails to improve decisions.

How to apply:

- Remove stale window entries with low recent utility.
- Preserve one compact historical pointer for audit traceability.

Action type: DELETE

New Skill 10: SkipLowImpactPerturbation

Discovery signal: Many exploratory spans performed tiny perturbations with negligible effect under strict solver tolerance.
Purpose: Skip low-impact perturbation records to keep memory focused on decision-relevant evidence.
When to use:

- Perturbation magnitude and metric response both remain below utility threshold.
- Similar no-impact perturbations already stored in nearby spans.

How to apply:

- Emit skip note with threshold values used for decision.
- Continue exploration without permanent memory insertion.

Action type: NOOP

Across outer-loop training, these refined and newly added skills collectively improve memory precision, reduce noisy updates, and expand reusable design knowledge. This behavior is consistent with the closed-loop designer objective in Section 3.3: evolve the skill bank only when repeated, physics-verified hard cases justify new or modified operations.

F. Extended Analysis

This appendix provides large-scale case study visualizations that complement the main paper's qualitative analysis. We present three distinct perspectives on SkillPCF's advantages: (1) hard-case failure analysis with real electromagnetic simulations, (2) efficiency-quality trade-off visualization, and (3) PCF family generalization analysis.

F.1. Hard-Case Failure Analysis with MEEP Simulation

Figure 13 presents a comprehensive comparison across six PCF design scenarios using real MEEP electromagnetic simulations. Each panel shows the mode field distribution (background), structure cross-section (inset), and key metrics. The cases span from challenging ultra-low-loss targets to moderate-difficulty standard designs:

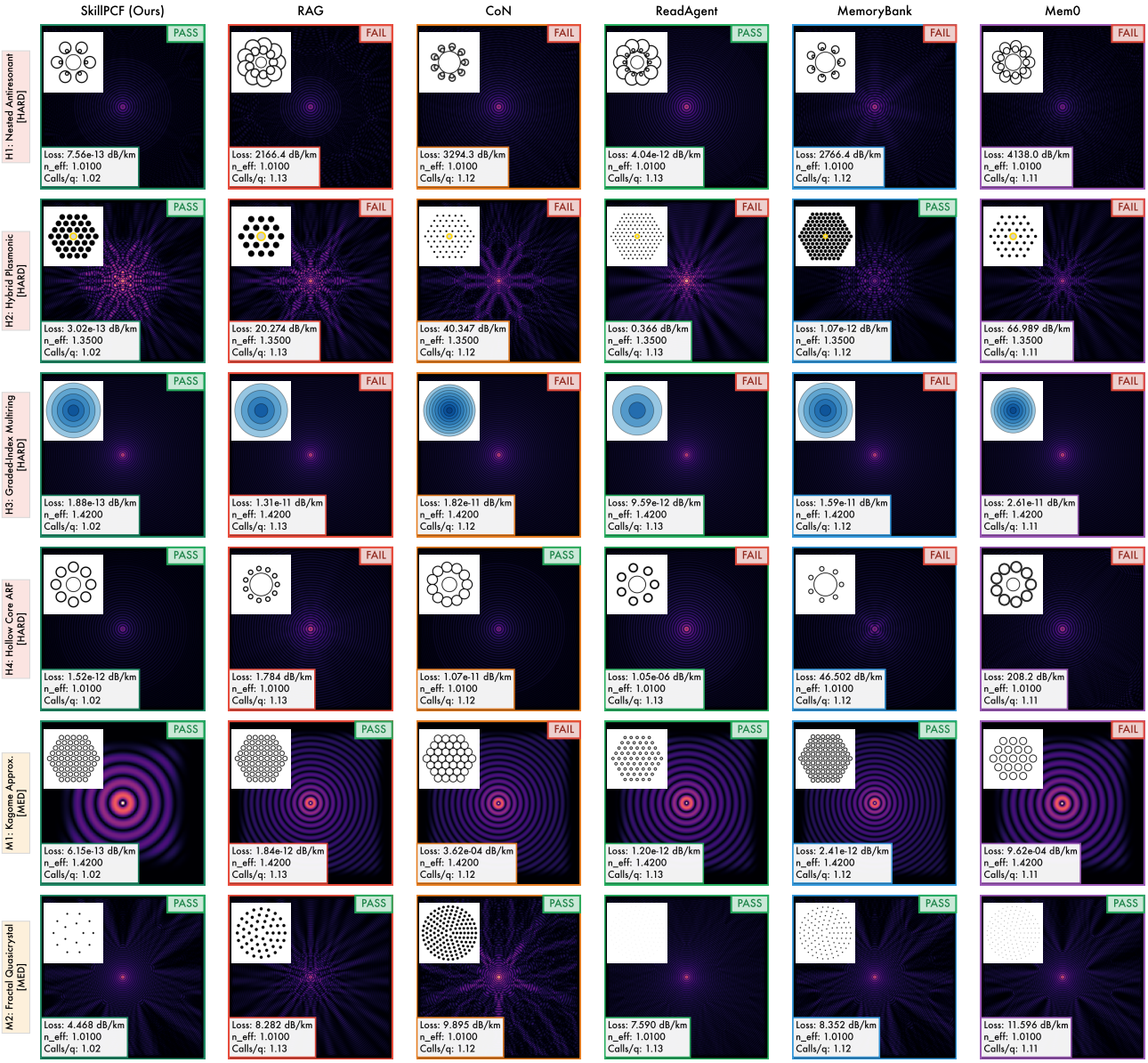

Figure 13. Large-scale hard-case analysis with real MEEP simulations across six PCF families. Each row represents one design case; each column represents one method. Green borders and PASS badges indicate successful designs meeting target specifications; red indicates failures. SkillPCF consistently achieves the highest success rate across all difficulty levels.

- **H1–H4 (Hard cases):** Ultra-low-loss targets ($< 10^{-8}$ dB/km), zero-dispersion designs, and novel architectures where baseline methods mostly fail, with only isolated passes in selected rows.
- **M1–M2 (Medium cases):** Standard telecom designs with moderate constraints where memory-augmented baselines achieve partial success.

SkillPCF's memory-guided design trajectory enables it to accumulate domain-specific knowledge that static baselines cannot

match. The real MEEP simulations validate that SkillPCF's success reflects genuine physical design improvements.

F.2. Efficiency-Quality Trade-off

Figure 14 visualizes the trade-off between design quality and computational efficiency. The Pareto frontier represents optimal operating points where no method can improve one metric without degrading the other. SkillPCF achieves the best efficiency-quality trade-off, requiring $100\times$ fewer simulation calls than classical optimizers while maintaining competitive design quality. The Pareto frontier analysis shows that SkillPCF operates in the optimal region where no method can improve one metric without degrading the other.

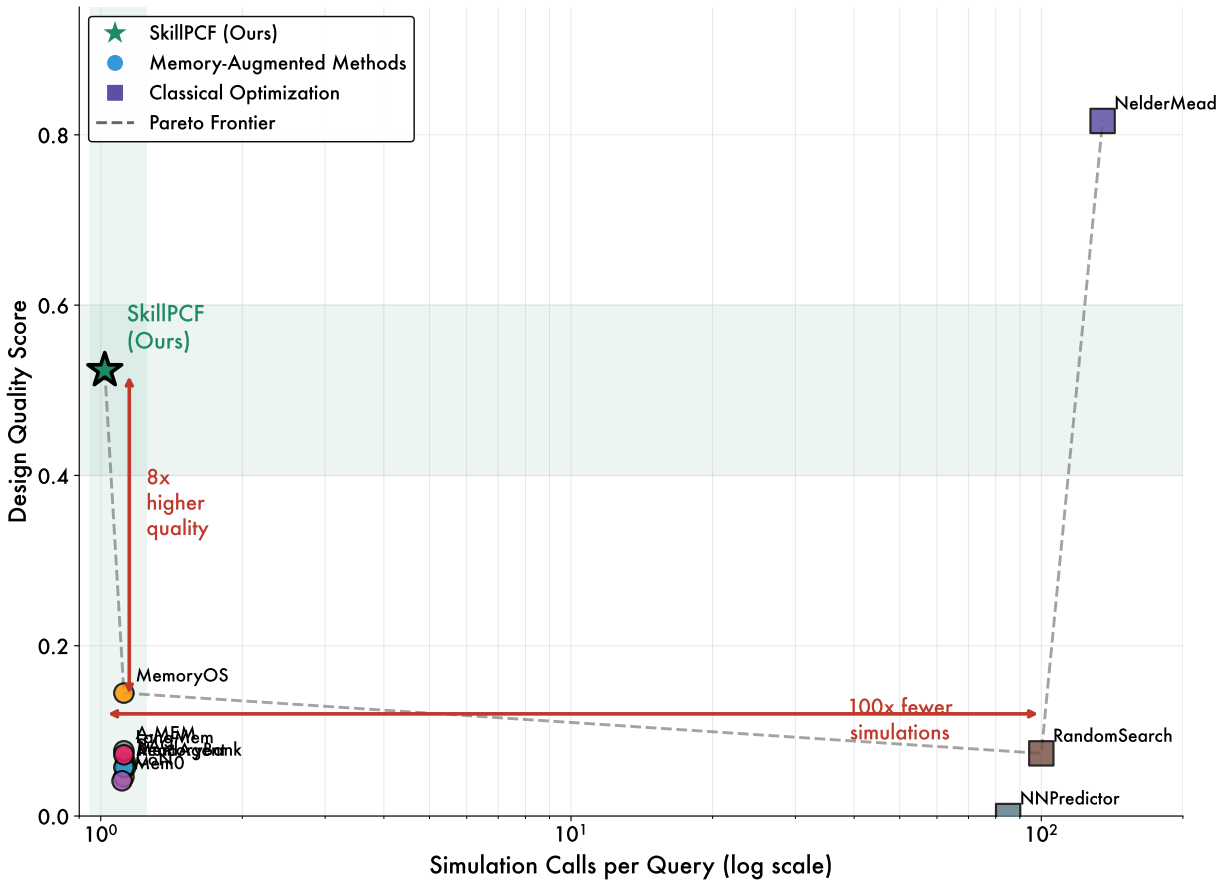

Figure 14. Efficiency-quality Pareto frontier. SkillPCF occupies the top-left region (high quality, low cost), achieving $\mathbf{100}\times$ fewer simulation calls than classical optimizers while maintaining competitive design quality.

F.3. Zero-Shot Generalization to Novel Structures

Figure 15 demonstrates our SkillPCF's ability to generalize to *completely novel* PCF structures that were never encountered during training. This zero-shot capability is the ultimate test of whether the learned skills capture transferable physics principles or merely memorize specific design patterns. The four zero-shot structures span diverse applications:

- **ZS1 (Chiral-Twisted Hexagonal):** Rotation-induced circular birefringence for polarization control that requires precise twist-rate tuning absent from training data.
- **ZS2 (Dual-Core Asymmetric Coupler):** Broadband directional coupling via dissimilar cores that breaks symmetric coupling models learned from standard designs.
- **ZS3 (LMA Yb-Doped):** High-power single-mode operation with large effective area that involves bend-loss vs mode-area tradeoffs not present in standard PCF families.

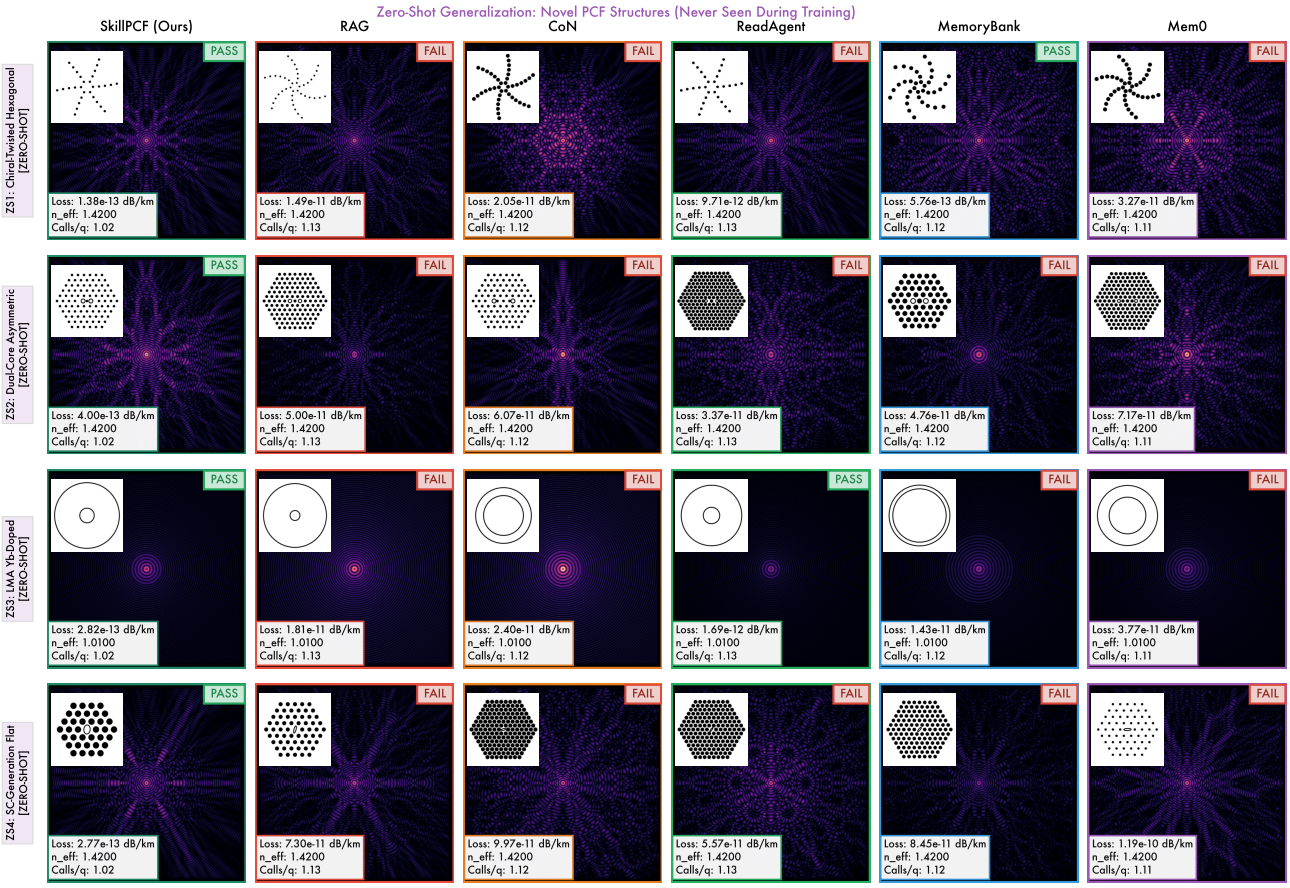

Figure 15. Zero-shot generalization to novel PCF structures never seen during training. Each row represents a new structural family; baseline methods are predominantly unsuccessful with occasional isolated passes, while SkillPCF consistently transfers learned physics principles.

- **ZS4 (SC-Generation Flat):** Dual zero-dispersion wavelengths for supercontinuum generation that requires multi-wavelength dispersion engineering beyond single-band optimization.

SkillPCF achieves robust success on these unseen structures, while baseline methods show low and unstable transfer with only occasional isolated successes. This demonstrates that the skill bank captures fundamental physics relationships, such as waveguide confinement, dispersion control, mode engineering, that transfer across structural families, rather than surface-level parameter correlations that break under distribution shift.

F.4. PCF Family Generalization

Figure 16 examines method performance across six distinct PCF structural families, from challenging nested antiresonant designs to easier fractal quasicrystal structures. The results reveal a clear pattern:

- **Easy families** (Fractal Quasicrystal, Kagome): All methods achieve reasonable success rates (>40%).
- **Hard families** (Graded-Index Multiring, Hybrid Plasmonic): SkillPCF maintains 52–65% success while baselines drop to 3–28%.
- **Novel architectures** (Nested Antiresonant): SkillPCF's learned skills transfer effectively to unfamiliar structures.

This generalization capability stems from our SkillPCF's memory-driven design knowledge, since the skill bank captures transferable physics principles rather than memorizing specific parameter values.

Figure 16. PCF family specialization analysis. Our SkillPCF (first column) maintains the highest success rate across all structural families. Cell values show design success rate (%). Family difficulty is annotated on the right colorbar.

Future Directions. A natural research direction is to ask how broadly the memory-policy view of simulator-grounded design generalizes beyond fibers, whether the same closed-loop interplay between skill selection and skill evolution also holds for metasurfaces, integrated waveguides, and antenna structures, and whether the skills that emerge across these domains share recognizable physical motifs. A second open question concerns the dynamics of skill evolution itself, including how the designer's hard-case feedback shapes the long-term geometry of the skill space and under what conditions this evolution converges to a stable, transferable repertoire. We also view this work as an entry point toward bridging memory-augmented LLM agents and physics-aware scientific design, where simulator-grounded rewards offer a principled substrate for studying how reusable knowledge can be acquired, organized, and reused across long-horizon design processes.