Reinforcement Learning On Photonic Inverse Design

Li-Wei Su
Department of Electrical Engineering
National Taiwan University
williamsu0011@gmail.com

Hseuh-En Chang
Department of Electrical Engineering
National Taiwan University
b12901042@g.ntu.edu.tw

To-Chih Chen
Department of Electrical Engineering
National Taiwan University
ryan940214@gmail.com

Cian-Jhu Chen
Department of Electrical Engineering
National Taiwan University
b12901136@g.ntu.edu.tw

Abstract

Traditional photonic design often fails to fully exploit the available design space due to a reliance on tuning limited parameters within known structures. While mainstream gradient-based inverse design treats development as an optimization problem, it is often susceptible to getting trapped in local minima. We propose formulating photonic inverse design as a sequential, layer-by-layer progression. By restricting the action space to a single layer of pixels at each step, we significantly reduce dimensionality and simplify the optimization landscape. We demonstrate that by leveraging shared physical principles, the learned policy captures underlying wave dynamics, allowing models to be efficiently fine-tuned to generalize across different tasks. This transferability significantly reduces the computational overhead for novel specifications compared to training from scratch.

1 Introduction

Photonics enables a diverse array of technologies, ranging from optical interconnects and neural networks to quantum optics [Su et al., 2020]. Traditionally, these systems are designed by tuning a limited number of parameters within established geometric structures; however, this constrained approach fails to fully exploit the vast potential of the available design space. While inverse design addresses this by treating the process as a gradient-based optimization problem [Molesky et al., 2018], the high dimensionality of the design landscape remains a significant hurdle, often leading to convergence to local minima.

We propose formulating photonic inverse design as a sequential decision process—where material is added layer-by-layer to light—thereby restricting the action space to a single layer of pixels at a time. This sequential perspective significantly reduces dimensionality and simplifies the optimization landscape. Furthermore, the inherent exploration features of reinforcement learning (RL) reduce the likelihood of becoming trapped in local minima.

Our current work demonstrates the efficacy of this layer-by-layer approach and shows that, because optical systems share underlying physical principles, models can be effectively fine-tuned to generalize across different tasks. This success in transfer learning suggests that the agent learns a fundamental physics-aware policy rather than merely overfitting to a specific geometry.

Preprint.

2 Related Work

While the first computational methods proposed for photonic design were gradient-based, they often suffer from becoming trapped in local minima [Su et al., 2020]. Furthermore, the standard procedure of binarizing a continuous permittivity distribution into discrete material values (e.g., silica or silicon) typically results in significant performance degradation.

Reinforcement Learning (RL) can overcome these issues by utilizing its inherent exploration capabilities. By employing discrete algorithms like Proximal Policy Optimization (PPO) [Schulman et al., 2017], an agent can take discrete actions from the start, ensuring the final design is physically realizable without the performance loss associated with post-optimization binarization.

Recent studies have explored Multi-Agent Reinforcement Learning (MARL) for inverse design, where each agent is responsible for a single "pixel" Mahlau et al. [2025]. However, this MARL framework may merely shift the burden of high-dimensional action spaces to the coordination between thousands of individual agents. These authors also utilized a Multi-Armed Bandit (MAB) approach, but these lack the sequential depth required for complex light redirection.

Another study employed Deep Q-Learning [Mnih et al., 2013] and Proximal Policy Optimization (PPO) [Schulman et al., 2017] for the autonomous inverse design of nanophotonic laser cavities.[Li et al., 2023] While the researchers demonstrated that these reinforcement learning (RL) agents outperform human intuition, it remains unclear whether they offer any significant advantage over mainstream gradient-based optimization methods.

Finally, supervised learning methods have been proposed. [Tahersima et al., 2019] [So et al., 2020] [Liu et al., 2021] However, these approaches are often impractical due to the lack of sufficiently large datasets. Supervised models are fundamentally limited by the quality of their training data and lack the flexibility to generalize across different design tasks as effectively as RL-based agents.

3 Problem Formulation

We formulate the inverse design of an optical power splitter as a sequential decision-making problem. The agent constructs the device layer-by-layer along the propagation direction (x-axis) by assigning each pixel in the current layer to silicon or silica. For the overall framework of the RL setup, refer to Fig 1.

3.1 Design Environment

The design region $\mathcal{D} \subset \mathbb{R}^2$ is discretized into a grid of $N \times N$ pixels, where $N = 20$. Each pixel (i, j) corresponds to a physical region of size $\Delta x \times \Delta y$, with $\Delta x = \Delta y = 100$ nm. The material distribution is represented by a binary matrix $\mathbf{M} \in \{0, 1\}^{N \times N}$, defined as:

$$M_{i,j} = \begin{cases} 1 & \text{if pixel } (i,j) \text{ is silicon } (n_{\text{Si}}) \\ 0 & \text{if pixel } (i,j) \text{ is silica } (n_{\text{SiO}_2}) \end{cases}$$

The refractive indices are $n_{\text{Si}} = 3.45$ and $n_{\text{SiO}_2} = 1.45$ at the operating wavelength of 1550 nm.

3.2 Action Space

The design process advances along the x-axis. The total number of steps is $T = N$. At each timestep $t \in \{0, 1, \ldots, T-1\}$, the agent determines the material distribution for the current layer (row $i = t$).

The action at step t is denoted as a vector $\mathbf{a}_t = (a_{t,0}, a_{t,1}, \ldots, a_{t,N-1}) \in \mathcal{A} = \{0, 1\}^N$, where $a_{t,j} = M_{t,j}$ represents the material choice for pixel (t, j).

3.3 Observation Space

At step t, the agent receives an observation $\mathbf{o}_t \in \mathcal{O}$, defined as:

$$\mathbf{o}_t = [\mathbf{m}_t^{\text{flat}}, \mathbf{h}_t, t, \mathbf{a}_{t-1}]$$

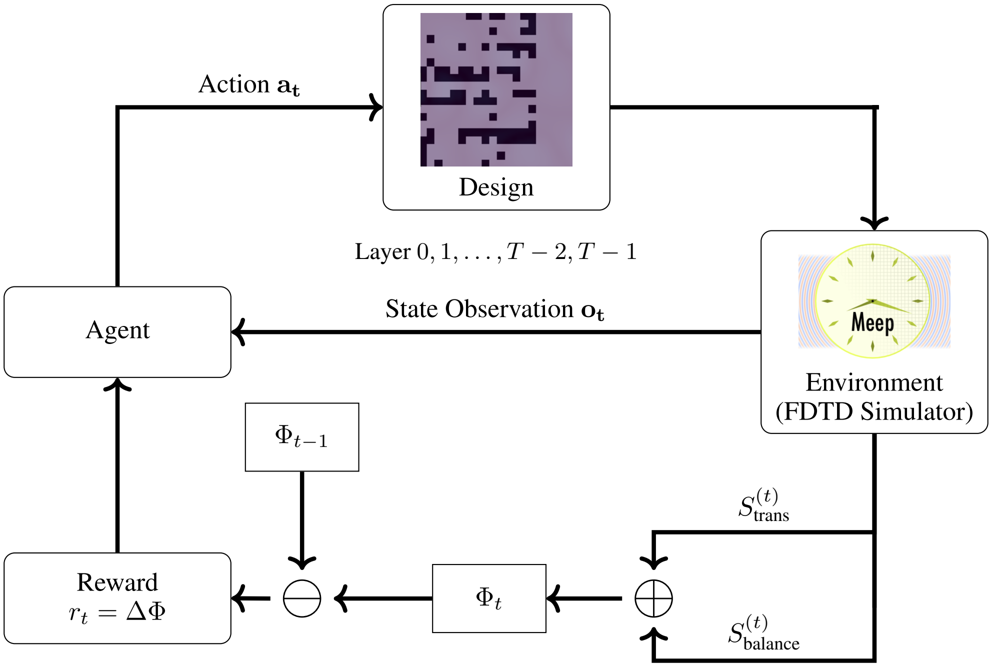

Figure 1: Reinforcement Learning framework for laye r-wise photonic structure generation with FDTD simulation

where the components are:

- $\mathbf{m}^{\text{flat}}_t \in \mathbb{R}^{N^2}$: The flattened material matrix $\mathbf{M}_t$ representing the design constructed up to the current step.
- $\mathbf{h}_t \in \mathbb{R}^{N_m}$: The normalized electromagnetic field measurements obtained from $N_m = 10$ monitors placed at predefined locations within the simulation region.
- $t \in \mathbb{N} \cup \{0\}$: The current layer index.
- $\mathbf{a}_{t-1} \in \{0, 1\}^N$: The material distribution of the previous layer. For $t = 0$, $\mathbf{a}_{-1}$ is initialized to match the input waveguide pattern.

3.4 Reward Function

The reward function is designed to optimize two objectives simultaneously: maximizing transmission efficiency and achieving the target power split ratio.

To ensure the device functions as a precise splitter, we define the Balance Score, $S^{(t)}_{\text{balance}}$, which measures how closely the power distribution matches the target ratio. It is defined as:

$$S^{(t)}_{\text{balance}} = \begin{cases} \max\left(1 - \frac{|r^{(t)}_{\text{actual}} - r_{\text{target}}|}{r_{\text{target}}}, 0\right) & \text{if } P^{(t)}_1 + P^{(t)}_2 > 0 \\ 0 & \text{if } P^{(t)}_1 + P^{(t)}_2 = 0 \end{cases}$$

Here, $r_{\text{target}} = 0.7$ represents the target split ratio (70% directed to port 1). The actual ratio is calculated as:

$$r^{(t)}_{\text{actual}} = \frac{P^{(t)}_1}{P^{(t)}_1 + P^{(t)}_2}$$

The score is maximized when the actual ratio equals the target and decays linearly as the deviation increases. Furthermore, to maintain consistent efficiency evaluation across different design iterations, we normalize the transmission using a fixed Reference Input Power, $P^{\text{ref}}_{\text{in}}$. This is computed once at initialization:

$$P_{\text{in}}^{\text{ref}} = 2 \cdot P_{\text{in}}^{\text{all-Si}}$$

where $P_{\text{in}}^{\text{all-Si}}$ is the input mode coefficient obtained from simulating a fully silicon-filled design region ($M_{i,j} = 1, \forall i, j$), and 2 is to make transmission score same as balance score.

The transmission efficiency at step t, denoted as $S_{\text{trans}}^{(t)}$, is normalized by the square of the input mode coefficient calculated from an all-silicon reference design ($P_{\text{in}}^{\text{ref}}$). It is defined as:

$$S_{\text{trans}}^{(t)} = \frac{P_1^{(t)} + P_2^{(t)}}{P_{\text{in}}^{\text{ref}}}$$

where $P_1^{(t)}$ and $P_2^{(t)}$ represent the power at the output ports. A composite score Φ_t is then calculated to combine the transmission and balancing objectives:

$$\Phi_t = w_{\text{trans}} \cdot S_{\text{trans}}^{(t)} + w_{\text{balance}} \cdot S_{\text{balance}}^{(t)}$$

Here, $w_{\text{trans}} = 10$ and $w_{\text{balance}} = 10$ are the weighting coefficients. The reward at step t, r_t, is defined as the incremental improvement in this composite score:

$$r_t = \begin{cases} \Phi_t - \Phi_{t-1} & \text{if } t > 0 \\ 0 & \text{if } t = 0 \end{cases}$$

3.5 Optimization Objective

The agent's goal is to maximize the expected cumulative reward over an episode:

$$\max_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right]$$

where $\pi_\theta : \mathcal{O} \to \mathcal{P}(\mathcal{A})$ is the policy parameterized by θ, τ is the trajectory, and $\gamma \in [0, 1]$ is the discount factor. By substituting the reward definition, the cumulative return can be expanded as:

$$\sum_{t=0}^{T-1} \gamma^t r_t = \Phi_{T-1} - \sum_{t=0}^{T-2} (1 - \gamma) \gamma^t \Phi_t$$

When the discount factor is close to unity (e.g., $\gamma = 0.99$) and the episode length is fixed at $T = 20$, the discounting penalty is minimal. Consequently, the objective approximately maximizes the final composite score of the completed device:

$$\max_\theta \mathbb{E}_{\tau \sim \pi_\theta} [\Phi_{T-1}] \approx \max_\theta \mathbb{E}_{\tau \sim \pi_\theta} \left[w_{\text{trans}} \cdot S_{\text{trans}}^{(T-1)} + w_{\text{balance}} \cdot S_{\text{balance}}^{(T-1)} \right]$$

4 Preliminary

4.1 Simulation time

Since our electromagnetic field simulations utilize the Finite-Difference Time-Domain (FDTD) method [Yee, 1966]—rather than the Finite-Difference Frequency-Domain (FDFD)[Rappaport and McCartin, 1991] method, which solves for the steady state directly—we must specify a simulation duration long enough for the field to propagate from the source in the input waveguide to the output waveguide. To determine an optimal simulation time, we monitored the magnetic field magnitude at the output waveguide as a function of time. We then selected the time at which the field reached saturation, ensuring that the system had transitioned from a transient state to a stable steady state, which is 200.

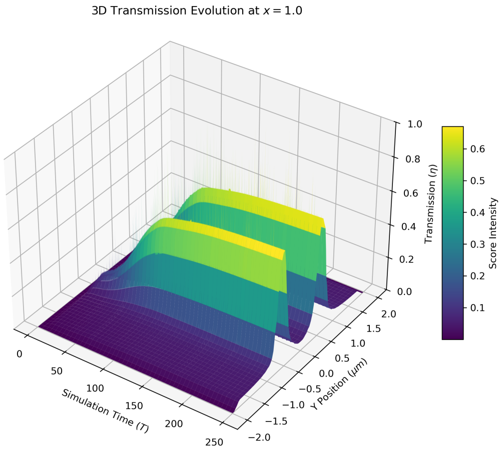

Figure 2: Transmission Evolution at $x = 1$

4.2 Pixel Size and Resolution

Other critical design parameters include the pixel size and the simulation resolution. For typical Silicon-on-Insulator (SOI) platforms used in optical communications at $\lambda = 1550$ nm, the pixel dimensions are constrained by fabrication limits (typically > 50 nm). Consequently, our design space involves a grid on the order of 10×10 pixels. [Shen et al., 2015, Piggott et al., 2017] The temporal resolution of the simulation is governed by Meep (our FDTD software package), which utilizes a stability criterion of $\Delta t = \Delta x/2$[Oskooi et al., 2010].

5 Method

5.1 Baseline Methods

To benchmark the proposed RL-based design method, we compare against a gradient-based inverse-design baseline based on adjoint optimization. The baseline solves the same splitter design problem in a continuous permittivity space, following standard topology-optimization formulations for nanophotonic devices Su et al. [2020].

We parameterize the design region by a continuous design variable $\boldsymbol{\rho} \in [0, 1]^{N\times N}$, defined on the same grid as the RL environment. The physical permittivity distribution is obtained via linear interpolation: $\varepsilon(\boldsymbol{\rho}) = \varepsilon_{\text{SiO}_2} + \boldsymbol{\rho}\,(\varepsilon_{\text{Si}} - \varepsilon_{\text{SiO}_2})$.

The baseline minimizes a composite objective function aiming for the target split ratio $r_{\text{target}} = 0.7$ and maximum efficiency:

$$\min_{\boldsymbol{\rho}} J(\boldsymbol{\rho}) = (r_{\text{actual}} - r_{\text{target}})^2 + (1 - r_{\text{actual}} - (1 - r_{\text{target}}))^2 - (P_{\text{total}})^2, \tag{1}$$

where P_{total} is the total transmitted power. Gradients $\nabla_{\boldsymbol{\rho}} J$ are computed using the adjoint method.

To address the challenge of creating fabrication-ready binary devices from continuous variables, we employ a standard two-stage continuation strategy involving sigmoid sharpening and projection. Detailed mathematical formulations of this binarization process and the specific hyperparameters used are provided in **Appendix A**.

5.2 Network Architecture

To effectively tackle the photonic inverse design problem, the agent must correlate the geometric material distribution with the resulting physical field behavior. We propose a dual-branch hybrid architecture (Figure 5.2) that processes these distinct modalities separately before fusion. The policy and value networks share a common feature extraction backbone.

Unlike standard CNN-based policies that rely solely on visual inputs, our architecture explicitly incorporates physical feedback via a **Scalar Branch**. This branch processes the electromagnetic field measurements ($\mathbf{h}_t$), the current layer index (t), and the previous layer distribution ($\mathbf{a}_{t-1}$), providing the agent with direct information about the light propagation status. Simultaneously, the **Spatial Branch** utilizes a CNN to extract local geometric features from the material history ($\mathbf{m}_t^{\text{flat}}$). These features are concatenated to form a physics-aware latent representation, enabling the policy to make informed material decisions based on both structural history and real-time optical performance.

The observation $\mathbf{o}_t$ is split into two streams:

1. **CNN Branch (Spatial Features)**: The flattened material matrix $\mathbf{m}_t^{\text{flat}}$ is reshaped to 20×20 and processed through a Convolutional Neural Network (CNN) to capture local spatial correlations. The architecture follows
   - **Layer 1**: Conv2d ($1 \to 16$ filters, 3×3, pad=1) + ReLU + MaxPool (2×2). Output: $10 \times 10 \times 16$.
   - **Layer 2**: Conv2d ($16 \to 32$ filters, 3×3, pad=1) + ReLU + MaxPool (2×2). Output: $5 \times 5 \times 32$.
   - **Layer 3**: Conv2d ($32 \to 64$ filters, 3×3, pad=1) + ReLU.
   - **Projection**: The tensor is flattened to 1600 dimensions and projected via a Linear layer to $\mathbf{f}_{\text{CNN}} \in \mathbb{R}^{128}$.
2. **Scalar Branch (Physical Features)**: The scalar components—field monitors, layer index, and previous layer distribution—are directly concatenated:

$$\mathbf{s}_{\text{scalar}} = [\mathbf{h}_t, t, \mathbf{a}_{t-1}] \in \mathbb{R}^{10+1+20} = \mathbb{R}^{31}$$

3. **Feature Fusion**: The spatial and scalar features are concatenated to form the latent representation $\mathbf{f}_t$:

$$\mathbf{f}_t = [\mathbf{f}_{\text{CNN}}, \mathbf{s}_{\text{scalar}}] \in \mathbb{R}^{159}$$

Finally, separate Multi-Layer Perceptron (MLP) heads process the fused features to generate the policy and value estimates:

- **Policy Head**: An MLP with hidden sizes $[256, 128]$ maps $\mathbf{f}_t$ to $\mathbb{R}^N$. A Sigmoid activation is applied to output the silicon probabilities $[p_{t,0}, \ldots, p_{t,N-1}]$.
- **Value Head**: An MLP with hidden sizes $[256, 128]$ maps $\mathbf{f}_t$ to a scalar $\mathbb{R}$, estimating the state value $V_\theta(\mathbf{o}_t)$.

All hidden layers utilize ReLU activations.

5.3 Sequential Design Process

While the problem formulation defines the state space, the core of our method lies in the integration between the RL agent and the FDTD solver as in Fig. 1. Unlike global optimization methods that update the entire structure simultaneously, our process is strictly causal:

1. **Step-wise Physics Update**: At each step t, after the agent fixes the material for the current layer, the simulation environment is updated, and the electromagnetic field is propagated forward through the newly added material.
2. **Real-time Feedback**: The field monitors capture the physical scattering effects created by the current partial structure. This data constitutes the $\mathbf{h}_t$ component of the observation.
3. **Closed-loop Decision**: This updated physical state allows the agent to react to wave interference patterns in real-time, effectively "guiding" the light path layer-by-layer based on actual physics rather than heuristic predictions.

This simulation-in-the-loop approach ensures that the policy learns to manipulate the wavefront dynamics directly.

A key challenge in inverse design is the sparse reward problem. To address this, we employ a dense reward shaping strategy. By defining the reward as the incremental improvement in the composite score ($r_t = \Phi_t - \Phi_{t-1}$), we provide the agent with immediate feedback at every layer, significantly accelerating convergence compared to sparse terminal rewards.

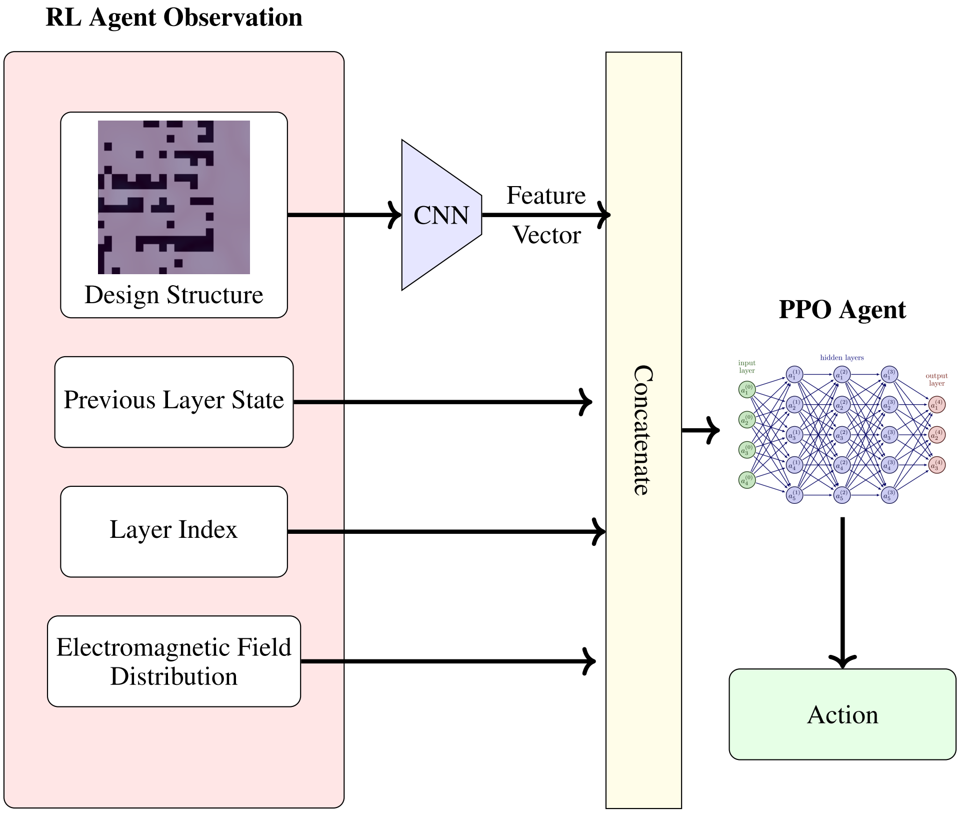

Figure 3: Overview of the PPO-based RL agent. The design structure is encoded by a CNN and concatenated with auxiliary observation features to generate the action

6 Experiment Results

6.1 Evaluation Setup

Tasks: The model is tasked with designing beam splitters that distribute light into two output waveguides with power ratios of 50:50 (1:1) and 70:30. The objective function prioritizes maximizing total transmission efficiency.

Baseline: We employ SPINS-B (Open source version of Stanford Photonic INverse design Software) as our primary baseline, which utilizes gradient-based adjoint optimization.

Metrics: We evaluate performance using the aforementioned composite score, Φ_t, which accounts for both transmission efficiency and power splitting accuracy.

Setup: To evaluate the design candidates, we execute Meep FDTD simulations for 200 time units to extract the electromagnetic field profile at the output plane. This simulated field is compared against a predefined target mode to calculate a balance and transmission score. The reward is then defined as the incremental difference between the current and previous scores, driving the agent toward iterative improvement.

Training Configuration: The agent is trained using the Proximal Policy Optimization (PPO) algorithm [Schulman et al., 2017], implemented via Stable-Baselines3. To ensure efficient exploration and stable convergence, we employ $N_{\text{envs}} = 16$ parallel environments. Detailed hyperparameters, including learning rates, coefficients, and the full training procedure, are provided in **Appendix B**.

6.2 Main Result

This experiment is conducted to evaluate whether our RL-based method can generate higher-quality 20×20 pixel designs and achieve a higher score than SPINS under the Meep simulation environment.

As shown in Fig. 4, our method begins to outperform the baseline after approximately 175 rollouts and attains a score that is about 3.1 points higher than the baseline at convergence.

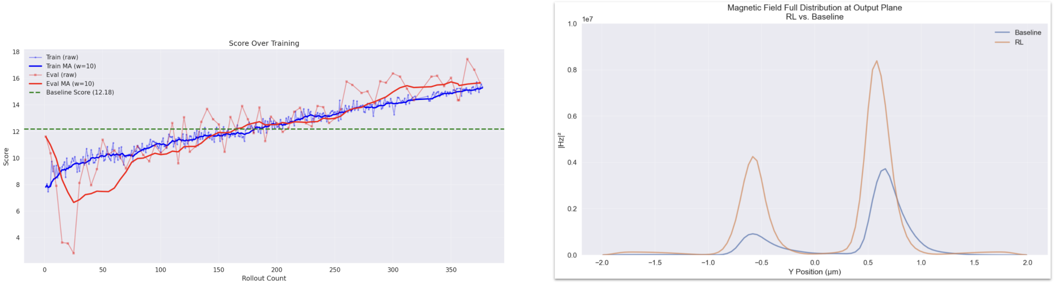

Figure 4: Experimental results. (Left) The score curve of the main setting during training. (Right) Comparison of the field distribution at the output plane between the agent's design and the gradient-based baseline.

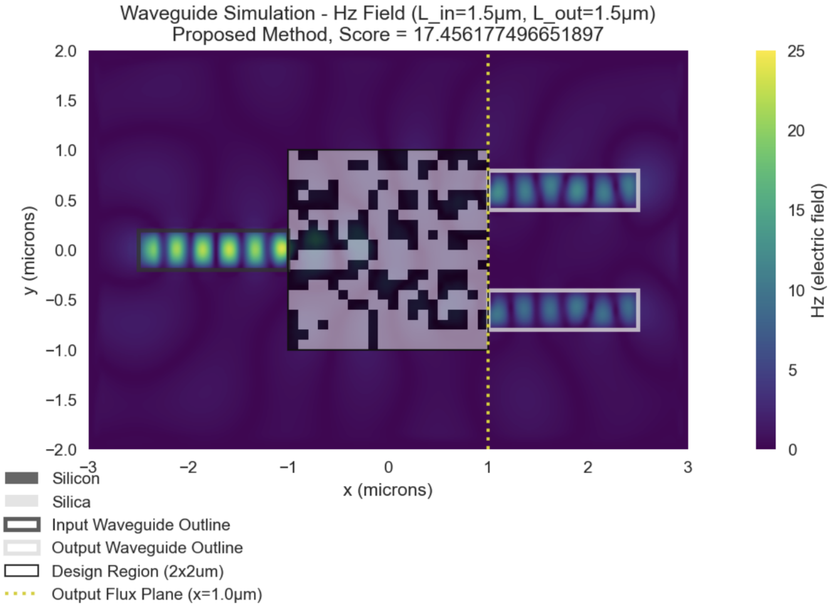

Figure 5: The design generated by agent

6.3 Adaptation Experiment

This experiment evaluates the model's ability to adapt to a modified task specification. We first train the model on a task with a target ratio of 50:50 until convergence. After convergence, we change the reward function to reflect a new target ratio of 65:35 and continue training using the same model initialization.

As shown in Fig. 6, the model requires approximately 378 rollouts to reach a score of around 20 during the initial training stage. In contrast, after the reward function is modified, the model achieves a comparable score within only 60 rollouts. This substantial reduction in required rollouts demonstrates that a pre-trained model can adapt to a closely related task more efficiently by updating its reward function.

7 Conclusion

We propose a reinforcement learning (RL)–based approach for photonic inverse design. By leveraging the exploratory capability of RL, our method discovers designs with higher performance than gradient-

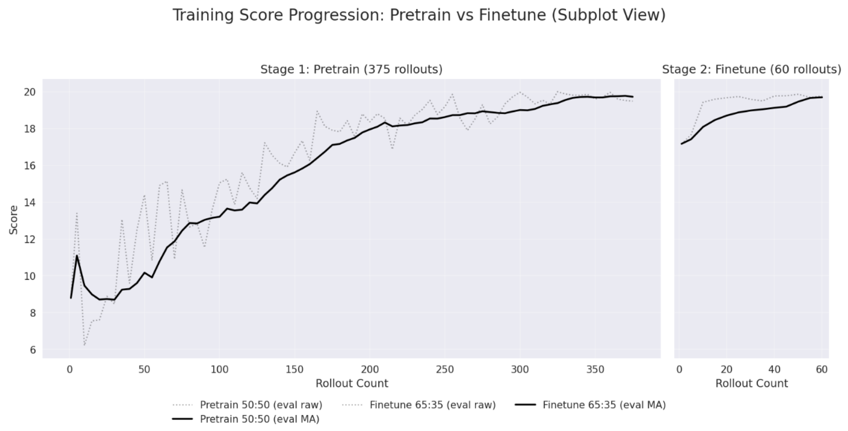

Figure 6: Finetune score curve

based approaches, which are often prone to local minima. Experimental results further indicate that the model captures underlying physical principles, enabling generalization across different inverse design tasks.

A primary limitation of the proposed approach is its computational inefficiency, as RL requires a large number of environment interactions. This issue is particularly pronounced when coupled with Finite-Difference Time-Domain (FDTD) simulations. However, in optical system engineering, the caliber of the final design is a more critical metric than the duration of the design phase.

Future work will explore extending the proposed framework to a meta-reinforcement learning (Meta-RL) setting [Finn et al., 2017]. By adopting a learning-to-learn paradigm, we aim to investigate whether a single model can adapt more efficiently to diverse photonic inverse design tasks, potentially reducing the need for costly retraining.

References

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. 2017.

Renjie Li, Ceyao Zhang, Wentao Xie, Yuanhao Gong, Feilong Ding, Hui Dai, Zihan Chen, Feng Yin, and Zhaoyu Zhang. Deep reinforcement learning empowers automated inverse design and optimization of photonic crystals for nanoscale laser cavities. *Nanophotonics*, 12(2):319–334, 2023. doi: doi:10.1515/nanoph-2022-0692. URL `https://doi.org/10.1515/nanoph-2022-0692`.

Zhaocheng Liu, Dayu Zhu, Lakshmi Raju, and Wenshan Cai. Tackling photonic inverse design with machine learning. *Adv. Sci. (Weinh.)*, 8(5):2002923, March 2021.

Yannik Mahlau, Maximilian Schier, Christoph Reinders, Frederik Schubert, Marco Bügling, and Bodo Rosenhahn. Multi-agent reinforcement learning for inverse design in photonic integrated circuits. 2025.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. 2013.

Sean Molesky, Zin Lin, Alexander Y Piggott, Weiliang Jin, Jelena Vucković, and Alejandro W Rodriguez. Inverse design in nanophotonics. *Nat. Photonics*, 12(11):659–670, November 2018.

Ardavan F Oskooi, David Roundy, Mihai Ibanescu, Peter Bermel, J D Joannopoulos, and Steven G Johnson. Meep: A flexible free-software package for electromagnetic simulations by the FDTD method. *Comput. Phys. Commun.*, 181(3):687–702, March 2010.

Alexander Y Piggott, Jan Petykiewicz, Logan Su, and Jelena Vučković. Fabrication-constrained nanophotonic inverse design. *Sci. Rep.*, 7(1):1786, May 2017.

C.M. Rappaport and B.J. McCartin. Fdfd analysis of electromagnetic scattering in anisotropic media using unconstrained triangular meshes. *IEEE Transactions on Antennas and Propagation*, 39(3): 345–349, 1991. doi: 10.1109/8.76332.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. 2017.

Bing Shen, Peng Wang, Randy Polson, and Rajesh Menon. An integrated-nanophotonics polarization beamsplitter with 2.4 × 2.4 μm2 footprint. *Nat. Photonics*, 9(6):378–382, June 2015.

Sunae So, Trevon Badloe, Jaebum Noh, Jorge Bravo-Abad, and Junsuk Rho. Deep learning enabled inverse design in nanophotonics. *Nanophotonics*, 9(5):1041–1057, 2020. doi: doi:10.1515/nanoph-2019-0474. URL `https://doi.org/10.1515/nanoph-2019-0474`.

Logan Su, Dries Vercruysse, Jinhie Skarda, Neil V Sapra, Jan A Petykiewicz, and Jelena Vučković. Nanophotonic inverse design with SPINS: Software architecture and practical considerations. *Appl. Phys. Rev.*, 7(1):011407, March 2020.

Mohammad H Tahersima, Keisuke Kojima, Toshiaki Koike-Akino, Devesh Jha, Bingnan Wang, Chungwei Lin, and Kieran Parsons. Deep neural network inverse design of integrated photonic power splitters. *Sci. Rep.*, 9(1):1368, February 2019.

Kane Yee. Numerical solution of initial boundary value problems involving maxwell's equations in isotropic media. *IEEE Transactions on Antennas and Propagation*, 14(3):302–307, 1966. doi: 10.1109/TAP.1966.1138693.

A Appendix A: Baseline Optimization Details

As discussed in Section 5.1, the gradient-based baseline utilizes a continuous design variable $\boldsymbol{\rho} \in [0, 1]$. To ensure the final design consists only of discrete materials (silicon or silica), we employ a two-stage optimization strategy:

Stage 1: Continuous Optimization. We first optimize $\boldsymbol{\rho}$ directly using the objective in Eq. 1. This stage explores the full design space, typically converging to a smooth, grayscale design.

Stage 2: Sigmoid Sharpening. To coerce the design toward binary values, we apply a sigmoid projection with progressively increasing sharpness:

$$\boldsymbol{\rho}_{\text{sig}} = \sigma\big(\beta(2\boldsymbol{\rho} - 1)\big), \tag{2}$$

where $\sigma(x) = 1/(1 + e^{-x})$ is the sigmoid function and β is a sharpening factor. We iteratively optimize the objective using $\boldsymbol{\rho}_{\text{sig}}$ while gradually increasing β (e.g., $\beta \in \{4, 8, 16, 32\}$). This progressive sharpening helps maintain optimization stability while strictly penalizing intermediate values.

Final Discretization. After convergence, a final thresholding is applied to obtain the binary mask:

$$\boldsymbol{\rho}_{\text{binary}} = \begin{cases} 1 & \text{if } \boldsymbol{\rho}_{\text{sig}} \geq 0.5 \\ 0 & \text{if } \boldsymbol{\rho}_{\text{sig}} < 0.5 \end{cases} \tag{3}$$

This binary design is then verified using a final forward simulation to report the performance metrics.

B Appendix B: Implementation Details

B.1 PPO Hyperparameters

We utilize the PPO algorithm with the hyperparameters listed in Table 1. These values were selected based on preliminary experiments to balance training stability and sample efficiency.

Table 1: PPO Training Hyperparameters

Hyperparameter	**Value**
Total timesteps	2×10^6
Parallel environments (n_{envs})	16
Steps per env per rollout (n_{steps})	240
Batch size	480
Epochs per update (n_{epochs})	20
Learning rate	1×10^{-4}
Discount factor (γ)	0.99
GAE parameter (λ)	0.95
Clip range (ϵ)	0.2
Entropy coefficient	0.005
Value loss coefficient	0.5
Max gradient norm	0.5
Evaluation frequency	Every 5 rollouts

B.2 Training Procedure

The training pipeline follows a standard iterative process:

1. **Trajectory Collection**: We utilize $n_{\text{envs}} = 16$ parallel environments to collect experience. Each environment executes $n_{\text{steps}} = 240$ steps, resulting in a total buffer size of $16 \times 240 = 3,840$ timesteps (approximately 192 completed episodes per rollout) before an update is triggered.

2. **PPO Update**: The collected trajectories are shuffled and divided into mini-batches of size 480. The policy and value networks are updated for $n_{\text{epochs}} = 20$ epochs using the clipped objective function with $\epsilon = 0.2$ and Generalized Advantage Estimation (GAE) with $\lambda = 0.95$.
3. **Periodic Evaluation**: To monitor performance, we evaluate the deterministic policy every 5 rollouts. During evaluation, gradients are disabled, and the agent's performance is recorded based on the transmission efficiency, balance score, and the composite reward Φ.
4. **Model Checkpointing**: We maintain a "best model" checkpoint. If the current evaluation score exceeds the previous best, the model weights are saved. A final checkpoint is also saved upon the completion of all 2×10^6 timesteps.

C Appendix C: Auxiliary Experiment

To evaluate the contribution of distinct observation components and architectural choices, we conducted five auxiliary experiments alongside the main configuration. All variants were trained under identical computational budgets and hyperparameter settings to ensure a fair comparison.

Experimental Variants: The experiments include the following configurations:

- **Main**: CNN-based architecture utilizing the full observation space (material matrix, monitors, index, previous layer).
- **CNN No Previous Layer**: CNN-based architecture excluding previous layer information.
- **MLP Full Matrix**: Non-CNN architecture including the full material matrix.
- **MLP Monitors Index Only**: Non-CNN architecture utilizing only monitors and the layer index (excluding the material matrix).
- **Similarity V1** and **Similarity V2**: The Main architecture augmented with a composite score component measuring the similarity between the current and previous layers.

In general, these variants exhibited faster initial training progress than the Main configuration. However, due to computational constraints, none of the experiments had fully converged at the time of evaluation; thus, these results reflect early-training behavior. Extended training horizons may reveal more distinct performance differences.

The slower convergence of the Main configuration is likely attributable to the higher complexity of the CNN-based architecture—which typically requires a longer training period—as well as the inherent stochasticity of the training process.

Preliminary results indicate no significant performance advantage across the different architectures or observation space designs. This suggests that the model does not rely exclusively on single observation components, but rather exhibits a degree of representational redundancy.

While the relatively narrow performance gaps suggest the RL agent possesses some robustness to architectural variations, the consistent underperformance of the *Main* configuration indicates that specific design choices may be impeding learning efficiency. Potential bottlenecks include the interaction between CNN feature extraction and the scalar branch, or the specific encoding of previous layer information in the *Main* setting.

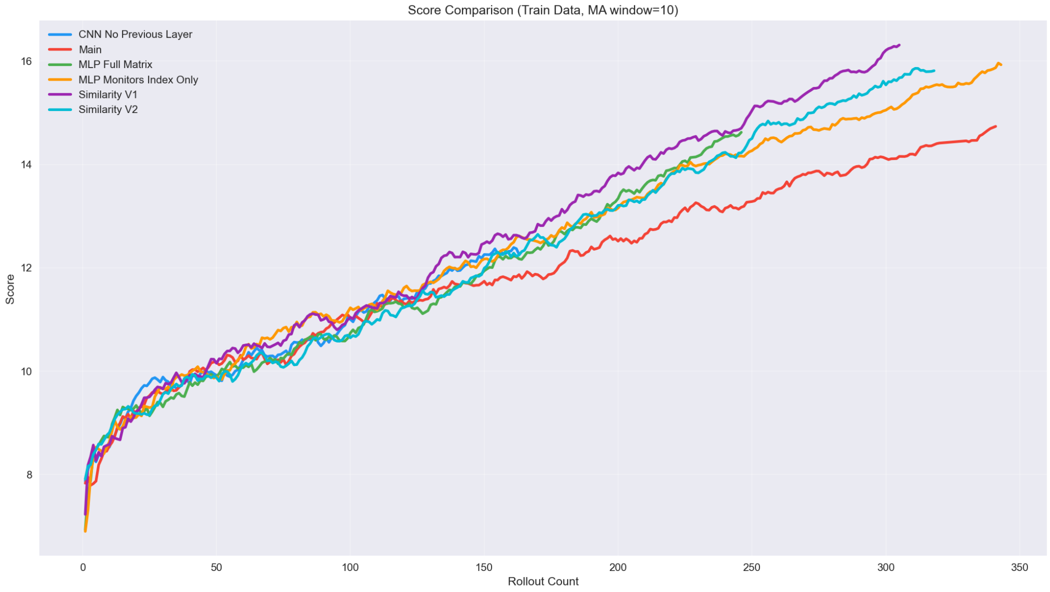

Figure 7: Score curve of different settings