# Multi-Agent Reinforcement Learning for Inverse Design in Photonic Integrated Circuits

**Yannik Mahlau, Maximilian Schier, Christoph Reinders, Frederik Schubert, Marco Bügling, Bodo Rosenhahn**

**Keywords:** Photonic Integrated Circuits, MARL, Discrete Optimization, Optical Computing

## Summary

Inverse design of photonic integrated circuits (PICs) has traditionally relied on gradient-based optimization. However, this approach is prone to end up in local minima, which results in suboptimal design functionality. As interest in PICs increases due to their potential for addressing modern hardware demands through optical computing, more adaptive optimization algorithms are needed. We present a reinforcement learning (RL) environment as well as multi-agent RL algorithms for the design of PICs. By discretizing the design space into a grid, we formulate the design task as an optimization problem with thousands of binary variables. We consider multiple two- and three-dimensional design tasks that represent PIC components for an optical computing system. By decomposing the design space into thousands of individual agents, our algorithms are able to optimize designs with only a few thousand environment samples. They outperform previous state-of-the-art gradient-based optimization in both two- and three-dimensional design tasks. Our work may also serve as a benchmark for further exploration of sample-efficient RL for inverse design in photonics.

## Contribution(s)

1. We introduce the design of photonic integrated circuit components as a discrete optimization problem, which we implement as a multi-agent reinforcement learning (MARL) environment. This bandit-like MARL environment tests the interaction of multiple thousand agents with very few samples.
   **Context:** Photonic integrated circuits enable optical computing, which is a new field for fast and energy efficient hardware accelerators (McMahon, 2023). Previous research in MARL mostly focused on a handful of agents using millions of training samples to learn an environment with many states (Rutherford et al., 2023). In contrast, we introduce a bandit setting with a single environment state, but multiple thousands of agents using only 10000 training samples. The sample efficiency is important because electromagnetic simulations for environment steps are time-consuming (Mahlau et al., 2025).
2. To solve the challenges of our new environment, we develop two multi-agent reinforcement learning algorithms. They are based on proximal policy optimization (Schulman et al., 2017) and an actor-critic approach with stochastic policies similar to the soft-actor-critic (Haarnoja et al., 2018). In extensive experiments, we show that our algorithms outperform previous state-of-the-art.
   **Context:** Inverse design in photonics has previously almost exclusively been performed using gradient-based optimization (Schubert et al., 2025), which can quickly find a decent solution, but its susceptibility to getting stuck in local minima during optimization impedes performance.
3. We publish the reinforcement learning environment and training algorithms as open-source.
   **Context:** We hope to facilitate reproducibility and further research.

# Multi-Agent Reinforcement Learning for Inverse Design in Photonic Integrated Circuits

**Yannik Mahlau[1], Maximilian Schier[1], Christoph Reinders[1], Frederik Schubert[1], Marco Bügling, Bodo Rosenhahn[1]**
mahlau@tnt.uni-hannover.de

[1]**Institute of Information Processing, Leibniz University Hannover, Germany**

## Abstract

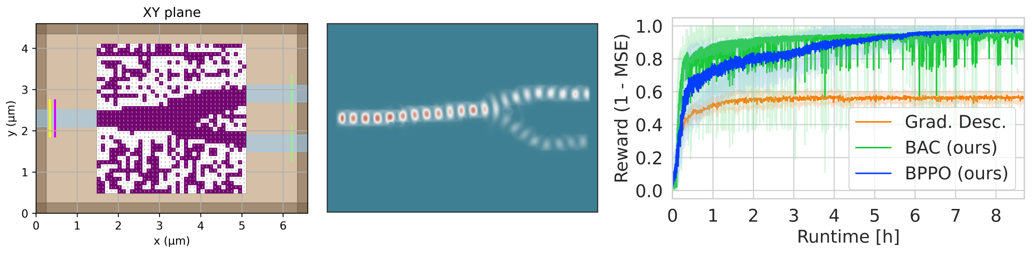
Inverse design of photonic integrated circuits (PICs) has traditionally relied on gradient-based optimization. However, this approach is prone to end up in local minima, which results in suboptimal design functionality. As interest in PICs increases due to their potential for addressing modern hardware demands through optical computing, more adaptive optimization algorithms are needed. We present a reinforcement learning (RL) environment as well as multi-agent RL algorithms for the design of PICs. By discretizing the design space into a grid, we formulate the design task as an optimization problem with thousands of binary variables. We consider multiple two- and three-dimensional design tasks that represent PIC components for an optical computing system. By decomposing the design space into thousands of individual agents, our algorithms are able to optimize designs with only a few thousand environment samples. They outperform previous state-of-the-art gradient-based optimization in both two- and three-dimensional design tasks. Our work may also serve as a benchmark for further exploration of sample-efficient RL for inverse design in photonics. [1]

## 1 Introduction

Modern computing and machine learning are fundamentally based on the representation and processing of information through digital electrical signals. This approach has driven technological advancement for decades. Although modern hardware has achieved significant improvements in computing power, particularly through parallel architectures like Graphics Processing Units (GPUs), fundamental limitations are being reached (Markov, 2014). Although the number of transistors continues to increase, the clock rate of individual processor cores has reached a plateau. This constraint persists even as GPU architectures leverage massive parallelization to achieve higher computational throughput. Consequently, this led to renewed consideration of analog computing for specialized applications (Haensch et al., 2019; Kazanskiy et al., 2022).

In optical computing, the digital representation is replaced by an analog encoding using electromagnetic light waves. This has the advantages of high bandwidth, operation speed, and energy efficiency (McMahon, 2023). Computation is performed on photonic integrated circuits (PIC), where optical components are connected for data input, output, and computation. PICs are especially interesting for neural network inference, whose energy consumption has increased drastically in the last years (Desislavov et al., 2023). To illustrate the potential speedups, our designs perform a small scalar-vector multiplication in about 150 femtoseconds, which is about 2500 times faster than a single clock cycle of a classical electrical computer.

[1]Our open-source implementation can be found at https://github.com/ymahlau/blend



(a) Environment Scene (b) Simulation (c) Optimization Results

Figure 1: Design task of a linear operation on a photonic integrated circuit. In (a), 65% of the incoming light emitted by a source (yellow) in the left waveguide (blue) should be routed to the top right waveguide (blue), while 35% of the light should go to the bottom right waveguide (blue). Transmission is measured as the ratio between output- (green) and input-detector (pink). The design task is a binary optimization problem for choosing silicon or air at every voxel. In (b), an electromagnetic simulation of the design is shown. During optimization (c), gradient descent gets stuck in a local minimum, while our BPPO and BAC show better exploration behavior.

However, analog computing requires high accuracy to work well, as errors through multiple operations accumulate. Designing a PIC component by hand is difficult, as designs are often counterintuitive and have a large number of parameters (Molesky et al., 2018). Therefore, inverse design has to be used, where a design is optimized automatically using an electromagnetic simulation. Since these simulations are differentiable, it has been very popular to optimize PIC components using gradient-based optimization (Molesky et al., 2018). However, gradient-based optimizations often get stuck in local minima. Therefore, it is necessary to find a better optimization algorithm.

We formulate the task of finding a good design for a PIC component as a discrete optimization problem. By discretizing the design space, the task can be formulated as placing either material or air at every voxel in 3D space. Since electromagnetic simulations are expensive, a learning algorithm must optimize a large design space using few samples. In extensive experiments, we show that our new multi-agent RL algorithms are able to deal with these challenges through the decomposition of the action space into multiple agents. The algorithms are based on proximal policy optimization (Schulman et al., 2017) and an actor-critic approach with stochastic policies similar to soft actor-critic (SAC) (Haarnoja et al., 2018). Through multiple design tasks, we illustrate that both algorithms significantly outperform gradient-based optimization, which was previously state-of-the-art in inverse design. In Figure 1, an example for optimizing a PIC design is shown.

## 2 Background

Inverse design of PIC components is based on electromagnetic simulations, which allows analysis by simulating light propagation through the component. The Finite-Difference Time-Domain (FDTD) method (Taflove & Hagness, 2005) is the most popular method for such a simulation (Dory et al., 2019; Augenstein & Rockstuhl, 2020). Light is characterized by an electric field $E$ and magnetic field $H$, which are three-dimensional vectors at every point in space. The propagation of light can be computed using Maxwell's equations (Maxwell, 1865)

$$\frac{\partial H}{\partial t} = -\frac{1}{\mu}\nabla \times E \qquad \text{and} \qquad \frac{\partial E}{\partial t} = \frac{1}{\varepsilon}\nabla \times H. \tag{1}$$

The electric and magnetic fields are updated in a leapfrog pattern, where the electric field is updated based on the magnetic field and vice versa. To efficiently compute these updates, space and time are discretized according to the Yee grid (Kane Yee, 1966). Specifically, the electric field is defined on whole integer time steps on the edges between spatial grid points. In contrast, the magnetic field is defined in between two integer time steps on the faces between four spatial grid points.

This complicated arrangement ensures that the curl operation ($\nabla\times$) can be computed quickly and accurately since no interpolation is necessary.

PICs are fabricated using either silicon or polymer materials. These different fabrication methods admit different fabrication constraints. Silicon PICs are manufactured in a subtractive process, resulting in 2D designs with uniform extrusion in the third axis (Han et al., 2014; Hung et al., 2002). Although silicon can be fabricated at nanometer resolution (Cai et al., 2019), we restrict the resolution of our designs to an economically viable size of 80nm. In contrast to silicon, polymer can be fabricated into intricate 3D structures using the two-photon polymerization (2PP) process (O'Halloran et al., 2023). But, 2PP has other design constraints. Specifically, no material can float in the air, and a design cannot have enclosed air cavities. Furthermore, the resolution of 2PP is more coarse than that of silicon fabrication with a minimum feature size of 500nm.

## 3 Related Work

The application of reinforcement learning to photonic inverse design has been enabled by recent speedups in electromagnetic simulation (Mahlau et al., 2025; Flexcompute, 2022). Early work focused on 1D topologies, which inherently have a small design space. Jiang et al. (2021); Jiang & Yoshie (2022) proposed combining unsupervised learning and RL with genetic algorithms to optimize multilayer solar absorbers. Similarly, Seo et al. (2022) applied Deep Q-Networks to the design of 1D metasurfaces. Park et al. (2024) developed a combination of gradient-based optimization and Deep-Q learning to optimize 1D metagratings. Furthermore, a great deal of work focused on optimizing a small number of parameters of a fixed shape parameterization (Li et al., 2023; Yu & Hao, 2025; Shams et al., 2024; Witt et al., 2023). Some initial success in 2D designs was achieved by Butz et al. (2023), which optimized a mode converter with 2070 binary variables using an undisclosed RL algorithm.

To the best of our knowledge, large-scale 2D or fully 3D topology optimization has only been achieved using gradient-based optimization with the adjoint method (Dory et al., 2019; Mansouree et al., 2020), automatic differentiation (Schubert et al., 2025; Hughes et al., 2019; Tang et al., 2023), or a combination of both (Luce et al., 2024).

## 4 Reinforcement Learning for Inverse Design

We model the problem of designing a PIC component as a discrete optimization problem. For every voxel in the discretized design space, the designer has to make a decision wether to place material or leave this spot empty, i.e. place air. The discretization of the design space in a grid follows naturally from the discretization of the FDTD simulation. Mathematically, the design space is $\mathcal{A} = \{0,1\}^N$, where $N$ is the number of discretized voxels. We denote a joint action using the bold letter $\mathbf{a} = (a_1, \ldots, a_N)$. The objective is to find the best joint action $\mathbf{a}^* = \arg\max_{\mathbf{a}\in\mathcal{A}} R(\mathbf{a})$, where $R : \mathcal{A} \to \mathbb{R}$ is the payoff function of the environment. The payoff function performs an FDTD simulation, which is expensive. Therefore, we define a budget $T$ that specifies how often the payoff function can be queried during optimization. This formulation can be viewed as a multi-armed bandit, except in contrast to classical formulations, performance is only measured using the best reward, i.e. $\max_{t\in\{1,\ldots,T\}} R(\mathbf{a}_t)$.

### 4.1 Baseline Optimization Algorithms

In the past, most inverse design has been performed using gradient-based optimization, which calculates the gradient through the differentiable FDTD simulation and performs supervised optimization. During optimization, the material permittivity $\varepsilon$ is modeled as a continuous parameter. For simulation, the continuous permittivity is mapped to the closest material permittivity. This mapping ensures that the simulation is physically valid at every optimization step. Furthermore, it enforces fabrication constraints for polymer designs by removing floating material and filling enclosed cavities. However, the mapping introduces a non-differentiable operation, but this issue can be overcome with a straight-through estimator (Schubert et al., 2025).

Nevertheless, gradient computation is costly in electromagnetic simulations and optimization is prone to get stuck in local minima. An optimization procedure that does not require gradients is the evolutionary algorithm (Jin, 2003). In this algorithm, a population of random binary designs is initialized, which are randomly mutated and recombined based on their performance during optimization. Another well-studied optimization procedure from the bandit literature is the upper confidence bound. Specifically, the decoupled upper confidence bound for trees (DUCT) is often used to decompose the large joint action space of a multi-agent system into small individual action spaces (Tak et al., 2014; Mahlau et al., 2024). This idea can be applied here, so that every voxel individually keeps track of the rewards associated with placing material or air. Every voxel then individually chooses to place material or air in the next iteration based on the DUCT formula

$$a_n^* = \arg\max_{a\in\{0,1\}} \frac{w_n^a}{v_n^a} + c \cdot \frac{\sqrt{v_n^0 + v_n^1}}{v_n^a}, \tag{2}$$

where $v^a$ is the number of times action $a$ has been used and $w^a$ is the accumulated sum of rewards. The exploration constant $c$ is a hyperparameter that balances between exploration and exploitation.

### 4.2 Multi-Agent Reinforcement Learning for Bandits

We adapt two different reinforcement learning algorithms to the bandit setting of inverse design. Both algorithms make use of the same neural network architecture, where the parameters are shared between all agents. The input of the neural network is a positional encoding $O : \{1, \dots, N\} \to \mathcal{O}$, mapping the agent index to an encoding providing a structural bias (Vaswani, 2017). In addition, agents implicitly share information through the shared neural network architecture.

#### 4.2.1 Bandit Actor-Critic (BAC)

We implement a novel actor-critic approach with stochastic policies similar to SAC (Haarnoja et al., 2018) for the multi-agent bandit problem. Let $\pi : \mathcal{O} \to \mathcal{P}(\{0, 1\})$ be a stochastic policy conditioned on a positional encoding. We denote a single sampled action as $a_n \sim \pi(\cdot \mid O(n))$ and use $\mathbf{a} \sim \pi$ as the shorthand for a sampled joint action. Thus, the optimization objective becomes $\pi^* = \arg\max_\pi \mathbb{E}_{\mathbf{a}\sim\pi}[R(\mathbf{a})]$. Next, we introduce a centralized critic $\mathbf{C}_\psi : \mathcal{A} \to \mathbb{R}$, which predicts the expected payoff of a joint action. Its parameters $\psi$ are learned through gradient descent on the regression error of $\mathbf{C}$ with

$$J_{\mathbf{C}}(\psi) = \mathop{\mathbb{E}}_{(\mathbf{a},r)\sim\mathcal{D}} \left[(r - \mathbf{C}_\psi(\mathbf{a}))^2\right] . \tag{3}$$

Since $\mathbf{C}$ is a differentiable surrogate of $R$, the parameters $\theta$ of a parameterized policy $\pi_\theta$ can be learned by gradient ascent on $\mathbf{C}$ using the objective

$$J_\pi(\theta) = \mathop{\mathbb{E}}_{\mathbf{a}\sim\pi_\theta} [\mathbf{C}_\psi(\mathbf{a})] . \tag{4}$$

However, the sampling of $\mathbf{a} \sim \pi_\theta$ is not differentiable, since the action space is (multi-) discrete. For single agent RL, this is commonly solved by expressing the expectation as a sum $\mathbb{E}_{\mathbf{a}\sim\pi}[\mathbf{C}(\mathbf{a})] = \sum_{\mathbf{a}\in\mathcal{A}} \pi(\mathbf{a})\mathbf{C}(\mathbf{a})$ (Christodoulou, 2019; Vieillard et al., 2020). But in our bandit setting, this is not tractable due to the large joint action space of size $|\mathcal{A}| = 2^N$. We also considered calculating the closed form for any single agent while keeping the actions of other agents fixed, as done in MARL algorithms like COMA (Foerster et al., 2018). But for such an objective, the number of critic evaluations would scale linearly with the number of agents, which becomes excessive for tens of thousands of agents. Instead, we approximate the gradient of the expected value by straight-through estimation on one drawn sample (Bengio et al., 2013). Specifically, for any action $a_n \sim \pi_\theta(\cdot \mid O(n))$, we approximate the gradient as $\nabla a_n \approx \nabla \pi_\theta(a_n = 1 \mid O(n))$. Thus, we can calculate the gradient through all agents using a single evaluation of the critic.

**Algorithm 1** Bandit Actor-Critic

```
Require: Simulation budget $T$, Critic gradient steps $U$, Policy gradient steps $G$, Critic learning rate $\lambda_C$, Policy learning rate $\lambda_\pi$, Critic batch size $B$
1:  $\mathcal{D} \leftarrow \{\}$
2:  Randomly initialize $\psi, \theta$
3:  repeat $T$ times
4:      $\mathbf{a} = (a_1, \ldots, a_N)$, with $a_i \sim \pi_\theta(\cdot|O(i))$          ▷ Sample from current policy
5:      $r \leftarrow R(\mathbf{a})$          ▷ Run simulation and observe pay-off
6:      $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{a}, r)\}$          ▷ Record experience
7:      repeat $U$ times          ▷ Train critic
8:          $\psi \leftarrow \psi - \lambda_C \nabla J_{\mathbf{C}}(\psi)$ using a random batch of size $B$ from $\mathcal{D}$          ▷ See Eq. (3)
9:      Randomly initialize $\theta$
10:     repeat $G$ times          ▷ Find new best policy
11:         $\theta \leftarrow \theta + \lambda_\pi \nabla J_\pi(\theta)$          ▷ See Eq. (4)
```

We present a short overview of BAC in Algorithm 1. In contrast to other actor-critic frameworks, we use a large number of gradient steps on the policy and critic networks per collected sample from the environment. For general Markov decision processes this would lead to estimation biases due to the temporal difference learning (Chen et al., 2021), but in the bandit setting this is not a concern. However, in the bandit setting, the lack of stochasticity may be problematic. The only stochasticity induced into the policy during optimization with Eq. (4) is through action sampling. The loss of entropy of the policy through optimization leads to a loss of entropy of the gradient ascent, which can impede training performance (Amir et al., 2021). We solve this in two ways. Firstly, we regularly reinitialize the policy to carry out a completely new gradient ascent starting from a policy with high entropy. Secondly, we mask a fraction $m$ of all agents for the gradient calculation in every gradient step. Thus, the optimization would retain randomness even with a fully deterministic policy.

### 4.2.2 Bandit Proximal Policy Optimization (BPPO)

Proximal Policy Optimization (PPO) (Schulman et al., 2017) has been one of the most successful reinforcement learning algorithms in recent years. However, basic PPO would be difficult to apply here because of the large action space and small number of samples. Following the idea of IPPO (De Witt et al., 2020) and MAPPO (Yu et al., 2022) to decompose the large action space into multiple agents, we implement Bandit Proximal Policy Optimization (BPPO). In the bandit setting, there is no notion of states, such that the PPO loss function for a single action $a_n$ can be written as

$$J(a_n, \theta_{\text{old}}, \theta) = \min\Big(\rho(a_n, \theta_{\text{old}}, \theta) A^{\pi_{\theta_{\text{old}}}}(a_n),\ \text{clip}\big(\rho(a_n, \theta_{\text{old}}, \theta), 1-\epsilon, 1+\epsilon\big) A^{\pi_{\theta_{\text{old}}}}(a_n)\Big),$$

where $\rho\left(a_n, \theta_{\text{old}}, \theta\right) = \frac{\pi_\theta(a_n|\,O(n))}{\pi_{\theta_{\text{old}}}(a_n|\,O(n))}$ is the policy ratio and $A^{\pi_{\theta_{\text{old}}}}(a_n)$ an advantage estimate for action $a_n$. In contrast to classical PPO, the advantage estimate is not dependent on any state, such that we can remove the critic completely and estimate the advantage as

$$A^{\pi_{\theta_{\text{old}}}}(a_n) = r_{a_n} - \mathop{\mathbb{E}}_{\mathbf{a} \sim \pi_{\theta_{\text{old}}}}\big[R(\mathbf{a})\big],$$

where $r_{a_n}$ is the payoff from a single sample associated with playing action $a_n$. The expected payoff of the old policy can be estimated using samples collected during rollout.

## 4.3 Environment Design

We introduce an reinforcement learning compatible environment for the design of PIC components. It includes three different scenarios, covering all of the major components necessary to build an optical computing system. The three different scenarios can be realized using silicon or polymer.

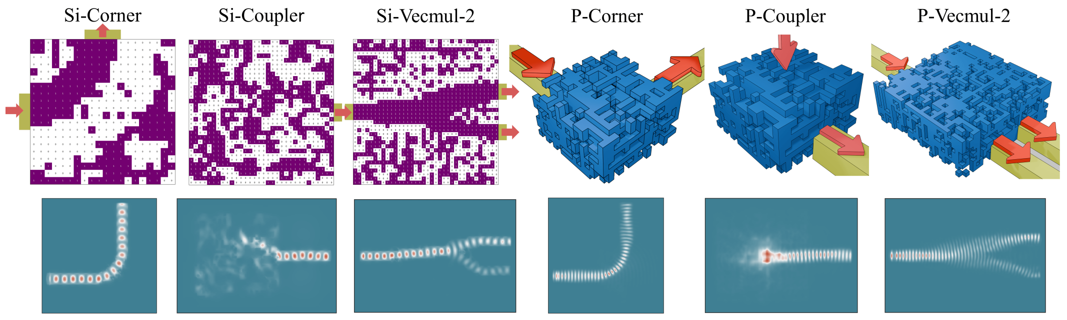


Figure 2: Optimized designs for the three different environments using either 2D silicon (purple) or 3D polymer designs (blue). The input and output waveguides are marked in green and with arrows. In the coupler environment, the input light comes from the top. The bottom row shows electromagnetic simulations of the designs.

In the first scenario, light needs to be transferred from an optical fiber to the chip as data input. Therefore, the challenge is to design a coupling element, which transfers light from free space into a waveguide. The fiber is placed vertically above the respective coupling element. The objective of the design is to transfer as much light as possible into the waveguide. This transfer can be measured using the poynting flux $P = E \times H$, which intuitively represents how much energy flows in a specific direction. Specifically, we would like to maximize the fraction of $P_x$ in the waveguide to $P_z$ below the source. Due to the time reversibility of Maxwell's equations, this design could also be used to transfer light from a waveguide into free space to measure computational output.

Secondly, light needs to be routed on the optical chip. Waveguides have low transmission loss on an optical chip as long as the waveguide is straight. However, for complex optical chips, it may be necessary to use sharp bends in the waveguides for efficient routing. These sharp bends introduce high transmission losses if the design is a simple round curve (Snyder & Love, 1983). Therefore, the second task is to design a component that connects two waveguides at a $90°$ angle. Again, the objective is to transfer as much light as possible measured as the fraction of poynting flux.

Lastly, a basic computational component is necessary to actually perform a calculation. Since we consider the standard linear form of Maxwell's equations, the computations are also restricted to linear operations. This restriction could be alleviated using nonlinear materials, which we leave to future work because it is a topic of active research in the photonics community (Bogdanov et al., 2024). For simplicity, we assume that the data is represented as directional energy in the waveguide, i.e. poynting flux. The simplest building block for a linear operation is a scalar-vector multiplication with a fixed vector, for example the trained weights of a neural network. By chaining multiple scalar-vector multiplications, one could also perform more complicated linear operations like a matrix-vector or even matrix-matrix multiplication. The scalar-vector multiplication can be implemented by distributing light from an input waveguide to multiple output waveguides according to the fixed vector. We measure the quality of a design as $1 - \text{MSE}$, where MSE is the mean squared error between the desired and actual poynting flux at the output waveguides. In Figure 2, designs of the six different setups are shown and Figure 1 shows more detailed analysis of the scalar-vector multiplication environment.

## 5 Experiments

We test the algorithms introduced above for the different PIC-components. As optimization in these environments is quite costly due to electromagnetic simulations, we devised a simple environment to optimize the various hyperparameters of all the algorithms presented above. The objective of this testing environment is to find a stable initial condition for Conways game of life (Gardner, 1970), which is a 2D binary grid optimization similar to our environments. As this environment is quick

| Environment | #Agents | Random | DUCT | Grad | EA | IQL | BAC (ours) | BPPO (ours) |
|---|---|---|---|---|---|---|---|---|
| Si-Corner | 400 | 27.5 ± 3.6 | 47.1 ± 12.1 | 74.4 ± 2.5 | 77.6 ± 6.2 | 80.9 ± 2.3 | <u>88.6 ± 0.8</u> | **91.7 ± 0.6** |
| Si-Coupler | 1024 | 9.7 ± 3.2 | 5.4 ± 0.5 | <u>41.5 ± 2.1</u> | 27.0 ± 2.3 | 13.0 ± 4.6 | 17.8 ± 0.8 | **51.6 ± 5.4** |
| Si-VecMul-2 | 1296 | 24.8 ± 2.3 | 44.2 ± 6.4 | 61.2 ± 3.0 | 73.60 ± 3.00 | 74.2 ± 4.4 | <u>96.1 ± 0.4</u> | **97.7 ± 1.4** |
| Si-VecMul-5 | 4356 | 7.4 ± 0.7 | 3.9 ± 0.4 | 34.7 ± 1.2 | 35.2 ± 2.9 | 63.1 ± 1.8 | **86.2 ± 4.1** | <u>76.2 ± 13.6</u> |
| P-Corner | 2560 | 3.9 ± 0.95 | 3.7 ± 0.4 | 7.1 ± 2.5 | 31.7 ± 2.9 | 13.1 ± 14.9 | <u>51.3 ± 2.4</u> | **55.9 ± 2.6** |
| P-Coupler | 6912 | 1.7 ± 0.13 | 1.8 ± 0.08 | 8.0 ± 1.7 | 21.5 ± 3.5 | 5.5 ± 4.7 | **37.0 ± 3.1** | <u>33.2 ± 8.3</u> |
| P-VecMul-2 | 7840 | 2.7 ± 0.49 | 6.4 ± 0.3 | 12.3 ± 1.5 | 43.6 ± 4.7 | 77.3 ± 5.0 | **95.6 ± 0.8** | <u>92.2 ± 7.2</u> |
| P-VecMul-5 | 27040 | 0.3 ± 0.0 | 0.7 ± 0.02 | 6.1 ± 0.4 | 5.3 ± 1.1 | 53.7 ± 1.4 | **89.0 ± 3.3** | <u>69.8 ± 24.1</u> |

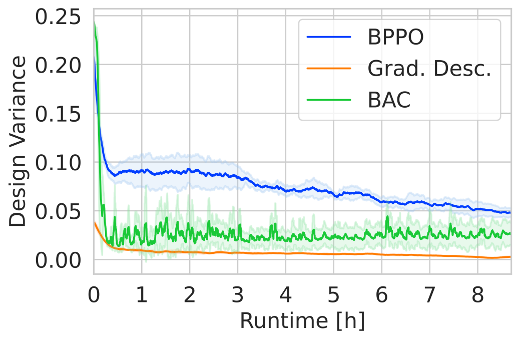
Table 1: Performance comparison across different environments and algorithms. For the corner and coupler environments, performance is the transmission efficiency, while for the scalar-vector multiplication performance is measured as $1 - \text{MSE}$. Mean and standard deviation are calculated over 5 seeds. The best performing algorithm is highlighted as **bold** and the second best is <u>underlined</u>.

to evaluate, we optimized all hyperparameters using this environment with Optuna (Akiba et al., 2019). Only the gradient-based optimization cannot be optimized with this environment, as it is non-differentiable. Gradient-based optimization has the learning rate as its only hyperparameter, which we optimized by a sweep over the silicon coupler environment.

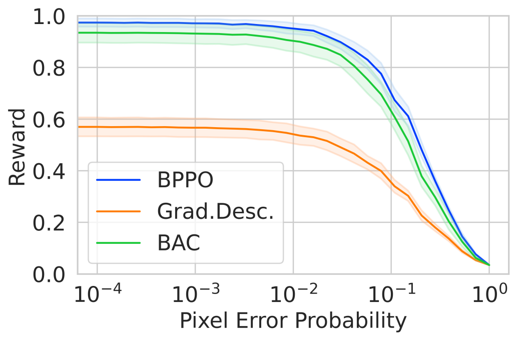
In addition to the algorithms presented above, we also tested independent Q-learning (IQL), which applies the standard Q-learning approach to each agent individually (Tan, 1993; Rutherford et al., 2023). In Table 1, the results of the evaluation are displayed. DUCT performed only little better than random search in most environments, as it lacks coordination between different agents. Gradient-based optimization performed better for the 2D designs of silicon than for the 3D polymer designs. In 3D designs, the mapping of latent parameters to a physically valid design can introduce gradient errors by the straight-through estimator. For example, when large parts of the design float in the air and are removed by the mapping, the gradient is calculated for a design with a large distance from the latent parameters. Evolutionary algorithms (EA) perform similar to gradient-based optimization with better results in some environments and worse results in other environments. The large number of agents and the small number of samples prevent EA from discovering an optimal solution. IQL performs better than gradient-based optimization in most environments, but the epsilon-greedy exploration sometimes leads to suboptimal exploration behavior. Our BAC and BPPO algorithms perform best, often with little difference. However, BAC seems to perform better in environments with many agents, while BPPO performs better in smaller environments. This indicates that the off-policy approach scales better to large designs, since all previous experience is contained in the replay buffer. The designs optimized by BPPO are shown in Figure 2.

For the corner environments, we can also compare the results against a naive circular corner design Bahadori et al. (2019). Measurement of this naive design in our environment yields a transmission of 88.4% for the silicon corner and 18.1% for polymer. In both cases, gradient-based optimization is unable to beat this simple baseline, while BAC and BPPO achieve better results.

To show that gradient descent gets stuck in local optima, we analyze the variance of designs during optimization in Figure 3. During optimization, the variance of designs produced by gradient descent quickly decreases, indicating that the optimization gets stuck a local optimum. In contrast, BPPO and BAC have higher design variance than gradient descent, indicating better exploration behavior.



(a) Design Variance



(b) Design Robustness

Figure 3: Comparison of BPPO, BAC and gradient descent regarding design variance and robustness in the Si-Vecmul2 environment. In (a), the variance is calculated over a window of 50 time steps during optimization. In (b), voxels are set to a random binary value with varying error probability.

Another important consideration is the robustness of the designs produced. For example, a high-performing design, whose performance collapses with a single voxel error, would be of little use in practice due to fabrication imperfections. For BPPO and BAC this is not the case as their stochastic policies implicitly optimize for randomness in the design voxels. Even if $10\%$ of the design voxels are randomly resampled, both algorithms still outperform the error-free designs of gradient-descent.

## 6 Conclusion and Future Work

We developed a formulation of the inverse design task for PIC components as a discrete optimization problem. For this bandit-like problem, we implemented environments representing the basic three components necessary to build an optical computing system. These components can be fabricated with either silicon or polymer, which leads to 2D or 3D design tasks respectively. The previous state-of-the-art gradient-based optimization can solve 2D silicon design tasks fairly well. However, we showed that it does not produce optimal results because it is prone to get stuck in local optima. Additionally, gradient descent struggles with 3D designs for polymer PICs. In contrast, our new BAC and BPPO algorithms show better exploration behavior, resulting in better performing designs.

In future work, we plan to extend the framework to nonlinear materials, which would alleviate the restrictions of Maxwell's linear equations. This would greatly increase the number of applications, for example building hardware accelerators for a trained neural network. Furthermore, there exist technologies for multi-material fabrication of polymer (Hu et al., 2022). Extending the action space from a binary choice to a class of three or more materials would be another extension of our framework that needs to be analyzed. Moreover, although our new algorithms outperform classical optimization algorithms by a large margin, they do not achieve perfect scores on our benchmarks. For building a real scalable optical computing system, even better designs are needed. We hope that the open-source implementation of the bandit-like environment serves as a benchmark for the development of new algorithms that can discover these designs.

### Broader Impact Statement

The research presented in this paper advances inverse design methodologies for PIC components. Although these innovations promise progress in optical computing and energy efficiency, we must carefully consider their potential impact on employment for human designers. We believe that optimal design outcomes emerge from collaborative processes that combine human expertise with automated systems. Critical to this approach is addressing questions of social acceptance and ensuring technological advancement proceeds by augmenting rather than replacing human capabilities.

**Acknowledgments**

This work was supported by the Federal Ministry of Education and Research (BMBF), Germany under the AI service center KISSKI (grant no. 01IS22093C), the European Union within the Horizon Europe research and innovation programme under grant agreement no. 101136006 – XTREME, the Lower Saxony Ministry of Science and Culture (MWK) through the zukunft.niedersachsen program of the Volkswagen Foundation and the Deutsche Forschungsgemeinschaft (DFG) under Germany's Excellence Strategy within the Cluster of Excellence PhoenixD (EXC 2122) and (RO2497/17-1). Additionally, this work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 517733257.

## References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2623–2631, 2019.

Idan Amir, Tomer Koren, and Roi Livni. Sgd generalizes better than gd (and regularization doesn't help). In *Conference on Learning Theory*, pp. 63–92. PMLR, 2021.

Yannick Augenstein and Carsten Rockstuhl. Inverse design of nanophotonic devices with structural integrity. *ACS Photonics*, 7(8):2190–2196, 2020. DOI: 10.1021/acsphotonics.0c00699.

Meisam Bahadori, Mahdi Nikdast, Qixiang Cheng, and Keren Bergman. Universal design of waveguide bends in silicon-on-insulator photonics platform. *J. Lightwave Technol.*, 37(13):3044–3054, Jul 2019.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Aditya Bhatt, Daniel Palenicek, Boris Belousov, Max Argus, Artemij Amiranashvili, Thomas Brox, and Jan Peters. Crossq: Batch normalization in deep reinforcement learning for greater sample efficiency and simplicity. In *International Conference on Learning Representations*, 2024.

Andrey A. Bogdanov, Sergey Makarov, and Yuri Kivshar. New frontiers in nonlinear nanophotonics. *Nanophotonics*, 13(18):3175–3179, 2024. DOI: doi:10.1515/nanoph-2024-0396.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: Composable transformations of Python+NumPy programs, 2018.

Marco Butz, Alexander Leifhelm, Marlon Becker, Benjamin Risse, and Carsten Schuck. A universal approach to nanophotonic inverse design through reinforcement learning. In *CLEO 2023*, pp. STh4G.3. Optica Publishing Group, 2023. DOI: 10.1364/CLEO_SI.2023.STh4G.3.

Ming Cai, Hyunwoo Park, Jackie Yang, Youseok Suh, Jun Chen, Yandong Gao, Lunwei Chang, John Zhu, S C Song, Jihong Choi, Gary Chen, Bo Yu, Xiao-Yong Wang, Vincent Huang, Gudoor Reddy, Nagaraj Kelageri, David Kidd, Paul Penzes, Wayne Chung, S.H. Yang, S.B. Lee, B.Z. Tien, Giri Nallapati, S.-Y. Wu, and P. R. Chidambaram. 7nm mobile soc and 5g platform technology and design co-development for ppa and manufacturability. In *2019 Symposium on VLSI Technology*, 2019.

Xinyue Chen, Che Wang, Zijian Zhou, and Keith Ross. Randomized ensembled double q-learning: Learning fast without a model. In *9th International Conference on Learning Representations, ICLR 2021*, 2021.

Petros Christodoulou. Soft actor-critic for discrete action settings. *arXiv preprint arXiv:1910.07207*, 2019.

Richard Courant, K. Friedrichs, and Hans Lewy. Über die partiellen differenzengleichungen der mathematischen physik. *Mathematische Annalen*, 100:32–74, 1928.

Christian Schroeder De Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.

Radosvet Desislavov, Fernando Martínez-Plumed, and José Hernández-Orallo. Trends in ai inference energy consumption: Beyond the performance-vs-parameter laws of deep learning. *Sustainable Computing: Informatics and Systems*, 38:100857, 2023. ISSN 2210-5379.

Constantin Dory, Dries Vercruysse, Ki Youl Yang, Neil V Sapra, Alison E Rugar, Shuo Sun, Daniil M Lukin, Alexander Y Piggott, Jingyuan L Zhang, Marina Radulaski, et al. Inverse-designed diamond photonics. *Nature communications*, 10(1):3309, 2019.

Timothy Dozat. Incorporating Nesterov Momentum into Adam. In *Proceedings of the 4th International Conference on Learning Representations*, pp. 1–4, 2016.

Flexcompute. Tidy3d: hardware-accelerated electromagnetic solver for fast simulations at scale. https://www.flexcompute.com/download-whitepaper/, 2022.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Ahmed Fawzy Gad. Pygad: An intuitive genetic algorithm python library. *Multimedia tools and applications*, 83(20):58029–58042, 2024.

Martin Gardner. Mathematical games. *Scientific american*, 222(6):132–140, 1970.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Wilfried Haensch, Tayfun Gokmen, and Ruchir Puri. The next generation of deep learning hardware: Analog computing. *Proceedings of the IEEE*, 107(1):108–122, 2019. DOI: 10.1109/JPROC.2018.2871057.

Hee Han, Zhipeng Huang, and Woo Lee. Metal-assisted chemical etching of silicon and nanotechnology applications. *Nano Today*, 9(3):271–304, 2014. ISSN 1748-0132.

Qin Hu, Graham A. Rance, Gustavo F. Trindade, David Pervan, Long Jiang, Aleksandra Foerster, Lyudmila Turyanska, Christopher Tuck, Derek J. Irvine, Richard Hague, and Ricky D. Wildman. The influence of printing parameters on multi-material two-photon polymerisation based micro additive manufacturing. *Additive Manufacturing*, 51:102575, 2022. ISSN 2214-8604. DOI: https://doi.org/10.1016/j.addma.2021.102575.

Tyler W Hughes, Ian AD Williamson, Momchil Minkov, and Shanhui Fan. Forward-mode differentiation of maxwell's equations. *ACS Photonics*, 6(11):3010–3016, 2019.

N.P Hung, Y.Q Fu, and M.Y Ali. Focused ion beam machining of silicon. *Journal of Materials Processing Technology*, 127(2):256–260, 2002. ISSN 0924-0136.

Anqing Jiang and Osamu Yoshie. A reinforcement learning method for optical thin-film design. *IEICE Transactions on Electronics*, 105(2):95–101, 2022.

Anqing Jiang, Liangyao Chen, and Osamu Yoshie. Otf gym: A set of reinforcement learning environment of layered optical thin film inverse design. In *CLEO: Science and Innovations*, pp. SM1Q–7. Optica Publishing Group, 2021.

Yaochu Jin. *Evolutionary Algorithms*, pp. 49–71. Physica-Verlag HD, Heidelberg, 2003. ISBN 978-3-7908-1771-3.

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, May 1966.

Nikolay L Kazanskiy, Muhammad A Butt, and Svetlana N Khonina. Optical computing: Status and perspectives. *Nanomaterials*, 12(13):2171, 2022.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

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.

Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with warm restarts. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

Alexander Luce, Rasoul Alaee, Fabian Knorr, and Florian Marquardt. Merging automatic differentiation and the adjoint method for photonic inverse design. *Machine Learning: Science and Technology*, 5(2):025076, 2024.

Yannik Mahlau, Frederik Schubert, and Bodo Rosenhahn. Mastering zero-shot interactions in cooperative and competitive simultaneous games. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 07 2024.

Yannik Mahlau, Frederik Schubert, Konrad Bethmann, Reinhard Caspary, Antonio Calà Lesina, Marco Munderloh, Jörn Ostermann, and Bodo Rosenhahn. A flexible framework for large-scale fdtd simulations: open-source inverse design for 3d nanostructures. In *Photonic and Phononic Properties of Engineered Nanostructures XV*, volume 13377, pp. 40–52. SPIE, 2025.

Mahdad Mansouree, Hyounghan Kwon, Ehsan Arbabi, Andrew McClung, Andrei Faraon, and Amir Arbabi. Multifunctional 2.5d metastructures enabled by adjoint optimization. *Optica*, 7(1):77–84, Jan 2020. DOI: 10.1364/OPTICA.374787.

Igor L Markov. Limits on fundamental limits to computation. *Nature*, 512(7513):147–154, 2014.

James Clerk Maxwell. Viii. a dynamical theory of the electromagnetic field. *Philosophical Transactions of the Royal Society of London*, 155:459–512, 1865.

Peter L. McMahon. The physics of optical computing. *Nature Reviews Physics*, 5(12):717–734, October 2023.

micro resist technology GmbH. ma-n 400 and ma-n 1400 - negative tone photoresists. https://www.microresist.com, 2025.

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

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *International conference on machine learning*, pp. 16828–16847. PMLR, 2022.

Seán O'Halloran, Abhay Pandit, Andreas Heise, and Andrew Kellett. Two-photon polymerization: Fundamentals, materials, and chemical modification strategies. *Advanced Science*, 10(7): 2204072, 2023.

Chaejin Park, Sanmun Kim, Anthony W. Jung, Juho Park, Dongjin Seo, Yongha Kim, Chanhyung Park, Chan Y. Park, and Min Seok Jang. Sample-efficient inverse design of freeform nanophotonic devices with physics-informed reinforcement learning. *Nanophotonics*, 13(8):1483–1492, 2024. DOI: doi:10.1515/nanoph-2023-0852.

J. Alan Roden and Stephen D. Gedney. Convolution pml (cpml): An efficient fdtd implementation of the cfs–pml for arbitrary media. *Microwave and Optical Technology Letters*, 27(5):334–339, 2000.

Alexander Rutherford, Benjamin Ellis, Matteo Gallici, Jonathan Cook, Andrei Lupu, Gardar Ingvarsson, Timon Willi, Akbir Khan, Christian Schroeder de Witt, Alexandra Souly, et al. Jaxmarl: Multi-agent rl environments in jax. *arXiv preprint arXiv:2311.10090*, 2023.

Maximilian Schier, Christoph Reinders, and Bodo Rosenhahn. Learned fourier bases for deep set feature extractors in automotive reinforcement learning. In *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 931–938. IEEE, 2023.

Maximilian Schier, Frederik Schubert, and Bodo Rosenhahn. Explainable reinforcement learning via dynamic mixture policies. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. To be published, 2025.

Frederik Schubert, Yannik Mahlau, Konrad Bethmann, Fabian Hartmann, Reinhard Caspary, Marco Munderloh, Jörn Ostermann, and Bodo Rosenhahn. Quantized inverse design for photonic integrated circuits. *ACS Omega*, 2025.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Dongjin Seo, Daniel Wontae Nam, Juho Park, Chan Y. Park, and Min Seok Jang. Structural optimization of a one-dimensional freeform metagrating deflector via deep reinforcement learning. *ACS Photonics*, 9(2):452–458, 2022. DOI: 10.1021/acsphotonics.1c00839.

Abdullah Bin Shams, Abdur Rahman Akib, and Stewart Aitchison. Deep transfer reinforcement learning in nanophotonics: A multi-objective inverse design approach. *2024 Conference on Lasers and Electro-Optics (CLEO)*, pp. 1–2, 2024.

A .W. Snyder and J. Love. *Optical Waveguide Theory*. Springer, 1 edition, 1983. ISBN 0412099500.

Allen Taflove and Susan C. Hagness. *Computational electrodynamics: the finite-difference time-domain method*. Artech House, Norwood, 3rd edition, 2005.

Mandy J. W. Tak, Marc Lanctot, and Mark H. M. Winands. Monte carlo tree search variants for simultaneous move games. In *2014 IEEE Conference on Computational Intelligence and Games*, pp. 1–8, 2014. DOI: 10.1109/CIG.2014.6932889.

Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pp. 330–337, 1993.

Rui Jie Tang, Soon Wei Daniel Lim, Marcus Ossiander, Xinghui Yin, and Federico Capasso. Time reversal differentiation of fdtd for photonic inverse design. *ACS Photonics*, 2023.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Nino Vieillard, Olivier Pietquin, and Matthieu Geist. Munchausen reinforcement learning. *Advances in Neural Information Processing Systems*, 33:4235–4246, 2020.

Donald Witt, Jeff Young, and Lukas Chrostowski. Reinforcement learning for photonic component design. *APL Photonics*, 8(10), 2023.

Ge Yang, Anurag Ajay, and Pulkit Agrawal. Overcoming the spectral bias of neural value approximation. In *International Conference on Learning Representations*, 2022.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Zhendi Yu and Ran Hao. Inverse design of high-q topological corner states nanocavities based on deep reinforcement learning. *Optics Communications*, 577:131402, 2025. ISSN 0030-4018. DOI: https://doi.org/10.1016/j.optcom.2024.131402.

# Supplementary Materials

*The following content was not necessarily subject to peer review.*

## A Algorithm Details

### A.1 Structural Priors through Positional Encoding

A major advantage of expressing the optimization problem as a multi-agent problem is the ability to introduce prior knowledge about the physical structure of the design and thus action space. We use an observation mapping $O : \{1, \dots, N\} \to \mathcal{O}$ that encodes this structure as a positional encoding (Vaswani, 2017). Specifically, for a discrete design space of size $N = |X| \cdot |Y| \cdot |Z|$ and $b$ bands, $O$ is defined as

$$O(n) = \begin{bmatrix} f(x(n)) \\ f(y(n)) \\ f(z(n)) \end{bmatrix} \text{, with} \tag{5}$$

$$f(a) = \begin{bmatrix} a \\ \sin(2^0 \cdot \pi \cdot a) \\ \dots \\ \sin(2^{b-1} \cdot \pi \cdot a) \\ \cos(2^0 \cdot \pi \cdot a) \\ \dots \\ \cos(2^{b-1} \cdot \pi \cdot a) \end{bmatrix}, \tag{6}$$

where $x : \{1, N\} \to [-1, 1]$, $y : \{1, N\} \to [-1, 1]$, and $z : \{1, N\} \to [-1, 1]$ are functions returning the position of an agent among the 3D grid axis, normalized to range $[-1, 1]$. We use positional encodings as they improve sample efficiency compared to regular multi-layer perceptrons, especially when applied to the critic (Yang et al., 2022; Schier et al., 2023).

The advantage of using such a positional encoding compared to an $N$-dimensional embedding layer is shown in Figure 4. We also compare to a flat actor, which is a simple learnable vector with one entry per agent or voxel, describing its activation probability. This represents a policy under the single agent paradigm acting on the joint action space. Both BAC and BPPO were trained with agents conditioned on the positional encoding or an equivalently sized embedding layer. The embedding layer does not contain any information about the structure of the 2D design space, which impedes performance. Interestingly, BPPO is more robust than BAC to the lack of a positional prior. This may be explained by the common observation that actor-critic algorithms are very sensitive to architecture changes to the critic rather than the policy (Bhatt et al., 2024) and our BPPO implementation not using a value network.

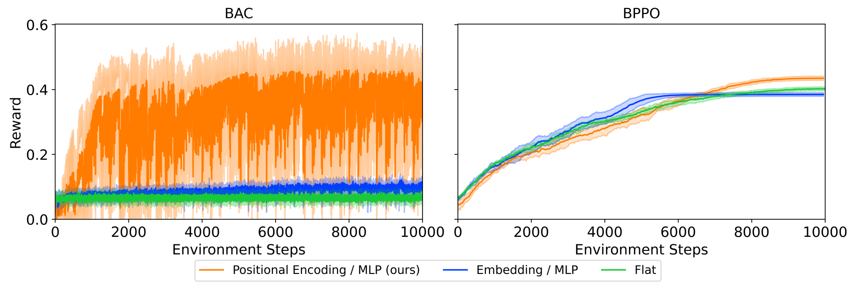


Figure 4: Performance of BAC and BPPO with and without a structural prior through the positional encoding. The experiment is performed on the testing environment presented in section A.3. Both algorithms achieve a better final design when agents are conditioned on the positional encoding.

### A.2 Independent Q-Learning (IQL)

A simple form of multi-agent learning in discrete action spaces is the independent application of Q-Learning to each agent (Tan, 1993; Rutherford et al., 2023; Schier et al., 2025). In our case, payoffs instead of state-action-values are independently learned. The critic $C_\psi : \mathcal{O} \times \{0,1\} \to \mathbb{R}$, parameterized by $\psi$, estimates the joint payoff given the action of a single agent. The critic is optimized by gradient descent on

$$J_C(\psi) = \underset{((a_1,\ldots,a_n),r)\sim\mathcal{D}}{\mathbb{E}} \left[ \frac{1}{n} \sum_{i=1}^{n} (C_\psi(O(i), a_i) - r)^2 \right],$$

where $\mathcal{D}$ is a buffer storing previously collected experiences of joint actions $(a_1, \ldots, a_N)$ and rewards $r$. The implicit greedy policy for evaluation is $\mu_\psi(i) := \arg\max_{a\in\{0,1\}} C_\psi(O(i), a)$. In order to carry out exploration during training, an $\epsilon$-greedy strategy is employed. Because the critic estimates global reward from local actions, the non-stationary actions of all other agents are indistinguishable from noise. Whenever the behavior of other agents in replayed experiences differs from the current implicit policy, either because the implicit policy has changed or because the exploratory policy was used, the global reward signal becomes biased. Therefore, IQL has no convergence guarantees, but we can reduce the bias of stale experiences by using a small buffer size.

### A.3 Game of Life Hyperparameter Optimization

We implemented a simple testing environment for hyperparameter optimization, which is very quick to evaluate, but still represents a similar structure to the bandit setting of PIC component design. To this end, we implemented an environment based on Conways game of life (Gardner, 1970). In this game, a 2D grid of cells is simulated, which can be dead or alive. At each step, a cell that is alive and has either two or three neighbors survives. Cells with more than three or less than two neighbors die of over- or under-population, respectively. Additionally, dead cells with exactly three neighbors become alive. The actions in this game determine the starting configuration for the game of life. The goal is finding a design, which has as many cells alive as possible, but is stable such that as few cells as possible change in a single step of the game. Therefore, performance is measured as the difference between the ratio of alive cells and the ratio of changed cells after a single game step.

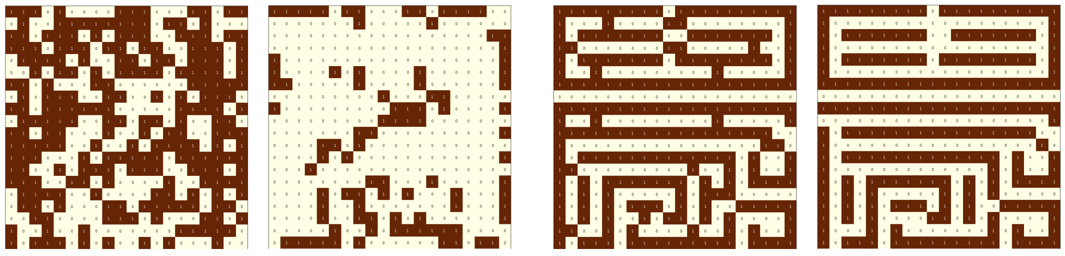

(a) Random Design (b) Optimized Design

Figure 5: Example designs for the game of life environment. In (a), a random design and in (b) a design optimized by BPPO are shown. For both (a) and (b), the left image displays the design and the right side the game of life grid after a single step.

In Figure 5, a random design and a design optimized by BPPO are shown. The random design achieves a score of $0.04$, because many cells change after the single game step. In contrast, the optimized design has many alive cells and the grid changes only a little after a game step, resulting in a score of $0.51$.

### A.4 Hyperparameters

Using the game of life environment described above, we optimized the various hyperparameters of our algorithms using Optuna (Akiba et al., 2019). Only gradient descent cannot be used in the game of life environment because it is not differentiable. For gradient-based optimization, we used the adam optimizer (Kingma & Ba, 2015) with nesterov momentum (Dozat, 2016). We used the standard parameters of $b_1 = 0.9$, $b_2 = 0.999$ and $\varepsilon = 10^{-8}$. For the learning rate, we used a cosine scheduling with linear warmup (Loshchilov & Hutter, 2017). We performed a sweep over the peak learning rate parameter using the silicon coupler environment. The results of this experiment are shown in Figure 6. We concluded that $0.01$ is the best learning rate, which we used for all the following experiments.

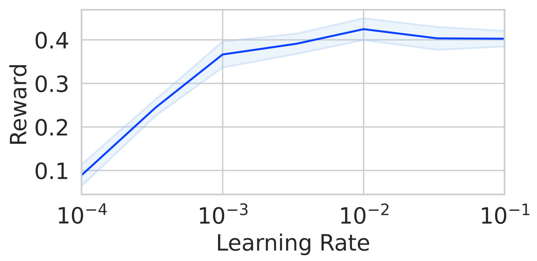


Figure 6: Influence of the learning rate hyperparameter on the performance of gradient-based optimization in the silicon coupler environment. Mean and standard deviation are calculated over five seeds.

Using DUCT, the problem arises that often all agents select the same actions as DUCT action selection is deterministic and all agents receive the same reward. Therefore, the first 50 of the 10000 actions during optimization were selected uniformly random. This ensures that the agents started using the DUCT formula with different initial values. Moreover, we slightly altered the standard formula by multiplying the exploration term with gaussian noise. The mean of this normal distribution as well as the exploration constants are hyperparameters, which we optimized in the game of life environment. The best hyperparameters we found were an exploration factor $c = 0.2145$ and a noise mean of $0.3242$.

| **Parameter** | **Value** | **Explanation** |
|---|---|---|
| Solutions per Population | 92 | Number of candidate solutions in each generation |
| #Parents mating | 8 | Number of solutions selected as parents for breeding |
| Keep parents | 2 | Number of best parents to include in next generation |
| Crossover type | uniform | Genes are randomly swapped with equal probability |
| Mutation type | swap | Mutation by exchanging positions |
| Gene mutation rate | 34% | Percentage of genes mutated in each offspring |

Table 2: Hyperparameters of the evolutionary algorithm.

For evolutionary algorithms, we used the PyGad library (Gad, 2024). The hyperparameter optimization resulted in the values listed in Table 2.

The IQL, BAC, and BPPO algorithm all used neural networks with a positional encoding as input. The hyperparameters used by all these algorithms are shown in Table 3. All of the algorithms use an MLP architecture. For IQL, we use a linear scheduling of the epsilon greedy exploration. Starting from $\varepsilon = 1$, the random probability is annealed to $\varepsilon = 0.05$ in the first 60% of the optimization and then is constant. For BPPO, we used a epsilon clip value of $\varepsilon_{\text{clip}} = 0.4978$ and an entropy loss coefficient of $0.005759$. BPPO performed 66 gradient updates between rollouts.

| Parameter | IQL | BAC | BPPO |
|---|---|---|---|
| #Sincos-Bands | 8 | 8 | 8 |
| Learning rate | $10^{-3}$ | $10^{-3}$ ($10^{-4}$) | $10^{-4}$ |
| Batch size | 32 | 32 | $10^4$ |
| Buffer size | 200 | $10^4$ | $32 \times$ #Agents |
| #Hidden layer | 2 | 2 | 4 |
| Hidden Dim. | 64 | 128 (256) | 126 |

Table 3: Common hyperparameters of the IQL, BAC and BPPO algorithms. For BAC, values in parentheses is for the critic, which is separate from the actor. BAC trains on every simulation result, such that the maximum buffer size is the number of simulations. BPPO performs 32 simulations between gradient updates, such that the number of data collected during rollouts is the number of agents multiplied by 32.

For BAC we used the following hyperparameters as shown in Algorithm 1. In our experiments, we used 128 critic gradient steps $U$, 1024 policy gradient steps $G$ and a batch size $B$ of 32. Furthermore, at the beginning of optimization, we collect $B$ experiences using uniformly sampled actions to start training the critic on a sufficiently filled experience buffer. We mask the gradients of 95% of agents for the optimization of the policy to maintain stochasticity $(m)$. Furthermore, to prevent loss of plasticity in the critic during optimization (Nikishin et al., 2022), we reinitialize the critic every 250 steps. After reinitialization, we perform 512 times as many critic gradient steps as normal to quickly minimize the regression error again.

## B Full Training Results

In Figure 7, the full training curves for all environments are shown. The first interesting observation is that BAC has instable training dynamics, because the critic is often retrained. In some environments, this may actually be beneficial as it increases exploration. Since BAC uses all previously collected data to regularly retrain the critic, instable training dynamics can never stop training progress completely. In contrast, BPPO has smooth training dynamic with a continuously increasing reward during training. However, in some environments, such as P-Vecmul-5, the variance between different seeds is high.

In the polymer coupler environment, the training collapse of gradient descent can be seen. In the first few environment steps, gradient descent is able to increase the reward quickly. However, at some turning point, structures floating in the air or enclosed cavities are removed by the mapping from latent parameters to a physically valid design. This leads to a large distance between the latent parameters and the actual design used in simulation, which introduces gradient errors. During optimization, gradient descent is never able to recover from these errors and consequently the performance goes to zero. In the other environments, the reward achieved by gradient descent also increases quickly in the first few training steps However, the optimization quickly gets stuck in a local minimum, where the reward remains relatively constant for the rest of the training.

## C Environment Details

Our environments perform an electromagnetic simulation to determine the quality of a design. For simulation, we use the FDTDX software (Schubert et al., 2025) written in JAX (Bradbury et al., 2018), which has been found to be the fastest open-source FDTD software currently available (Mahlau et al., 2025). To induce energy into the simulation with a light source, we use a total-field scattered-field definition (Taflove & Hagness, 2005), which allows unidirectional light input. We use a light source of wavelength 1550nm, which is the standard wavelength for telecommunication. Reflections at the simulation boundary are prevented through a convolutional perfectly matched layer (Roden & Gedney, 2000), which absorbs light directed at the boundary. In Figure 8, the simulation scenes for all environments are displayed. For silicon, we assumed a relative permittivity of 12.25 and for polymer 2.6326, which corresponds to the material of the ma-N-1400 series (micro resist technology GmbH, 2025).

| **Name** | **Resolution** | **Size [µm]** | **Sim. Time** | **Sim. Steps** | **Memory Req.** |
|---|---|---|---|---|---|
| Si-Corner | 20nm | $4 \times 4 \times 1.5$ | 100fs | 2623 | 188MB |
| Si-Coupler | 25nm | $6 \times 4.3 \times 2$ | 125fs | 2623 | 203MB |
| Si-Vecmul-2 | 25nm | $6.6 \times 4.6 \times 1.5$ | 106fs | 2222 | 193MB |
| Si-Vecmul-5 | 25nm | $9.6 \times 7.6 \times 1.5$ | 156fs | 3272 | 431MB |
| P-Corner | 100nm | $17 \times 17 \times 9$ | 200fs | 1049 | 161MB |
| P-Coupler | 100nm | $20 \times 15 \times 12$ | 150fs | 787 | 209MB |
| P-Vecmul-2 | 100nm | $24 \times 17 \times 8.5$ | 248fs | 1299 | 210MB |
| P-Vecmul-5 | 100nm | $36 \times 29 \times 8.5$ | 368fs | 1929 | 491MB |

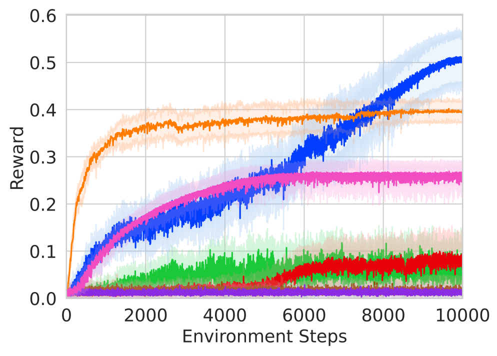
Table 4: Parameters for the electromagnetic FDTD simulations for the different environments. The abbreviation fs is the metrical unit femtosecond. The memory requirements are calculated based on the array sizes of electric and magnetic field, material properties and boundary states, but does not include intermediate values in the FDTD computation.

In Table 4, the detailed simulation parameters for the different environments are displayed. The resolution of the simulation has to be finer than the size of the design voxels to accurately calculate light propagation. For silicon, the voxel size is 80nm in the corner environment and 100nm in the coupler and vecmul-environments. Therefore, we used resolutions of 20nm and 25nm, respectively. The polymer designs have a voxel size of 500nm, such that we chose a simulation resolution of 100nm. The time discretization, i.e., the time passed per simulation step, is chosen such that the Courant-Friedrichs-Lewy stability conditions (Courant et al., 1928) are satisfied. The time discretization and the simulation time, which is usually in the order of a few hundred femtoseconds, determine the number of simulation steps that need to be performed. The number of simulation steps is the main factor that influences the computational runtime of the simulation. Because of the fine resolution, the silicon environments have a higher number of simulation steps than the polymer environments. With the memory requirements, the maximum number of parallel simulations on a graphics card can be calculated. However, the true VRAM usage on a graphics card is usually about twice as high as the calculated values in the table because of intermediate computational results in the simulation. Additionally, a single simulation is already very well parallelized due to its implementation in JAX, such that parallelization of multiple simulations yields little improvement.

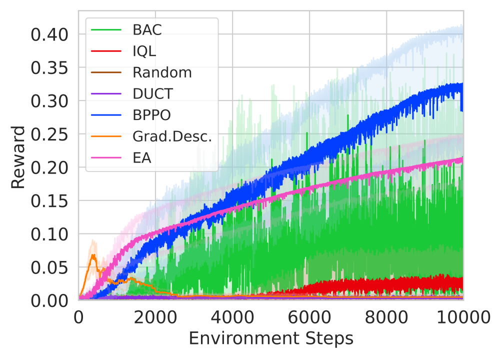
The actions in the environment are mapped to a physically valid design for the electromagnetic simulations described above. The physical validity is determined by the fabrication processes of silicon and polymer, which we describe in more detail here. Polymer fabrication enables rapid prototyping and small-batch production, while silicon-based PICs provide higher refractive index and smaller feature sizes.

Silicon PICs are fabricated in a subtractive process, where structures are patterned on a waiver. Common removal techniques include chemical etching (Han et al., 2014) and focused ion beam milling (Hung et al., 2002). This fabrication method limits structures to 2D designs with uniform extrusion in the third dimension. In other words, designs are limited to planar geometries without overhanging structures. Another important point to consider is the minimum feature size. Although single-digit nanometer resolution is possible in specialized facilities (Cai et al., 2019), such precision comes at a significant cost. For practical implementations, it is more economically viable to utilize fabrication facilities with less accurate machines. Our experiments use a minimum feature size of 80nm, which is easily achievable with economically viable fabrication methods. To ensure strict adherence to this fabrication constraint, we discretize the design space into square voxels of the corresponding size.

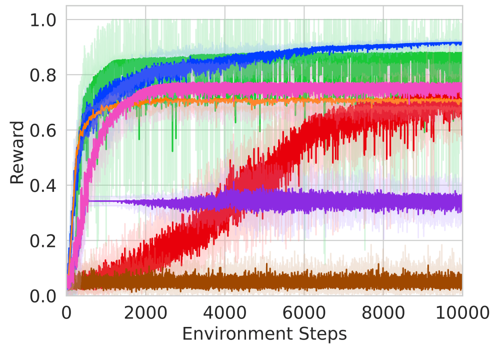
In contrast to silicon, polymer can be fabricated into intricate 3D shapes with overhangs and holes. In the two-photon polymerization (2PP) process (O'Halloran et al., 2023), liquid monomer resin is placed on a substrate and hardened using a femtosecond laser. The process relies on the simultaneous absorption of two near-infrared photons by the photosensitive resin, triggering localized polymerization only at the focal point where photon density is highest. After polymerization of the desired structures, the remaining liquid monomer is washed away. Any liquid resin remaining in the design would slowly polymerize over time. This leads to the fabrication constraint that designs with fully enclosed cavities cannot be produced, which would trap liquid resin that could not be washed away. In addition, all parts of the design must be connected to the ground, because no structures can float in the air. This constraint arises from the 3D fabrication capabilities and is not present in 2D silicon designs. In contrast to silicon, it is not possible to fabricate polymer at nanometer resolution. In our experiments, we restrict the design space to voxels of 500nm, which can be achieved by most modern 2PP printers.



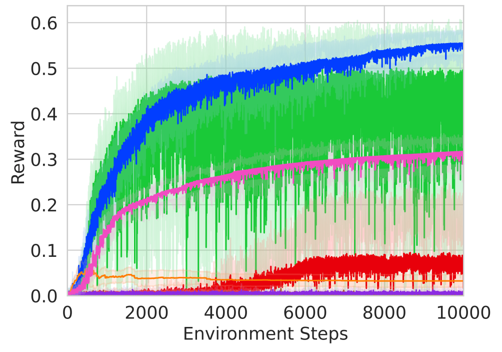
(a) Si-Coupler



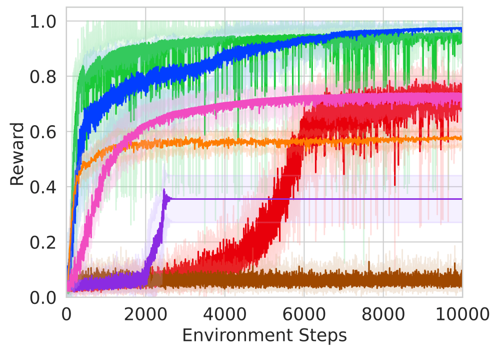
(b) P-Coupler



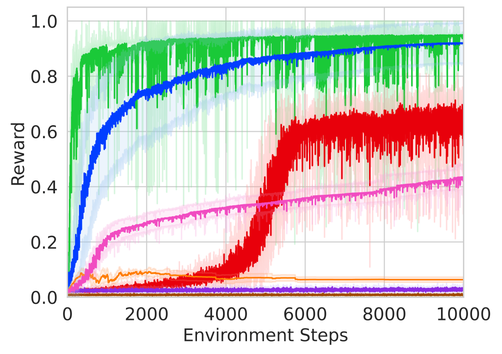
(c) Si-Corner



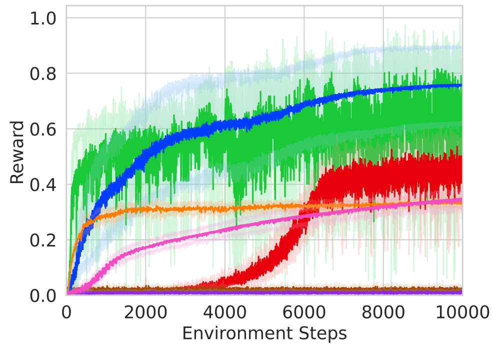
(d) P-Corner



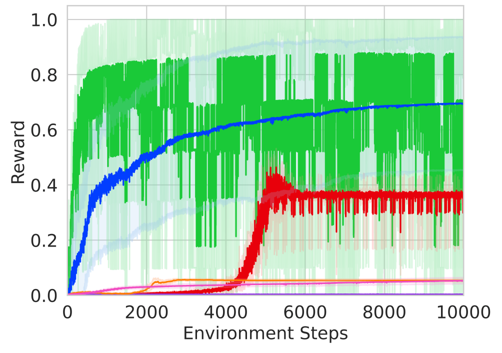
(e) Si-Vecmul-2



(f) P-Vecmul-2

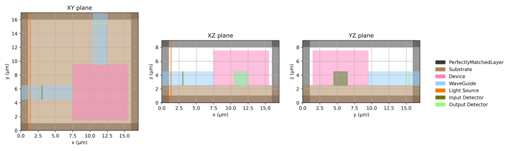


(g) Si-Vecmul-5

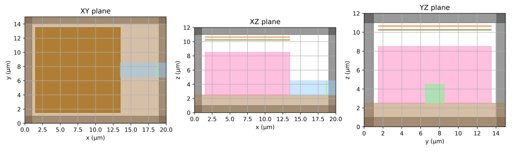


(h) P-Vecmul-5

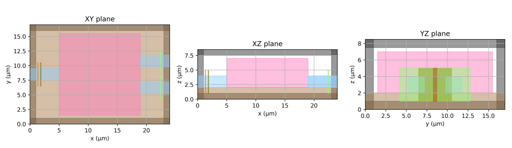
Figure 7: Learning curves for algorithms tested in our work in the different environments. Gradient descent performed fewer steps than the other algorithms because gradient computation required time. In this plot, the x-axis for gradient descent is stretched to allow a comparison with the other algorithms. Mean and standard deviation are calculated over five seeds.



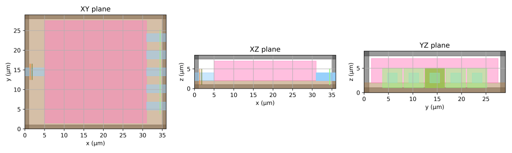
(a) P-Corner



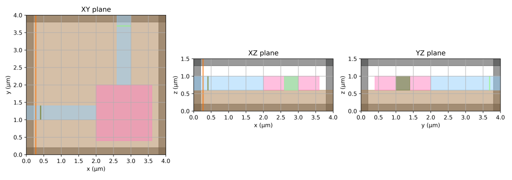
(b) P-Coupler



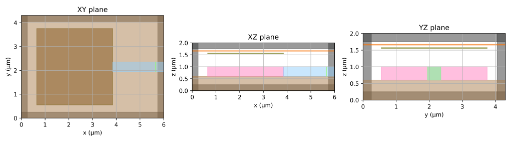
(c) P-Vecmul-2



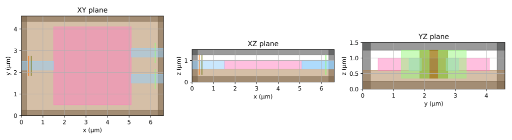
(d) P-Vecmul-5



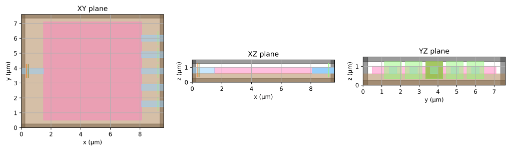
(e) Si-Corner



(f) Si-Coupler



(g) Si-Vecmul-2



(h) Si-Vecmul-5

Figure 8: Simulation Scenes for the different environments presented in our work.